Universität Ulm

Fakultät für Ingenieurwissenschaften und Informatik

Scheduling rechen- und datenintensiver Anwendungen in einem hierarchischen, verteilten Umfeld

Dissertation

zur Erlangung des Doktorgrades Dr. rer. nat. der Fakultät für Ingenieurwissenschaften und Informatik der Universität Ulm

vorgelegt von

Matthias Röhm

aus Reutlingen

2014

Amtierende Dekanin:	Prof. Dr. Tina Seufert
1. Gutachter:	Prof. Dr. Franz Schweiggert
2. Gutachter:	Prof. Dr. Manfred Reichert
Tag der Promotion:	12. Februar 2014

Danksagung

Ich möchte mich hier bei einigen Menschen bedanken, die mich bei der vorliegenden Arbeit besonders unterstützt haben.

Zuallererst möchte ich mich ganz herzlich bei Herrn Prof. Dr. Schweiggert für die großartige Unterstützung in all den Jahren bedanken! Erst durch seine Hinweise und Anmerkungen konnte ich aus den unterschiedlichen Teilen meiner Arbeit eine Dissertation formen.

Auch Dr. Matthias Grabert möchte ich an dieser Stelle ganz herzlich für die vielen Anmerkungen und Verbesserungsvorschläge danken. Seine Unterstützung hat mir sehr geholfen, den langen Weg zur fertigen Dissertation durchzuhalten.

Weiterhin möchte ich Herrn Prof. Dr. Manfred Reichert für seine Bereitschaft danken, meine Arbeit so zeitnah zu begutachten.

Ein weiterer Dank gilt Sina Wuhrer. Die Lektüre dieser Arbeit wäre ohne ihre Hilfe sicherlich keine erfreuliche Angelegenheit.

Meinen Eltern möchte ich für ihre Unterstützung während des Studiums und der Promotion danken, wobei insbesondere ihre Funktion als Korrekturleser sowie Probepublikum zum Feinschliff dieser Arbeit beitrug.

Natürlich danke ich an dieser Stelle auch Dir Martina. Du warst sowohl vor als auch während meiner Promotion stets für mich da. Ich hoffe, dass dies auch nach der Promotion so bleibt.

Zusammenfassung

Sowohl im wissenschaftlichen wie auch im industriellen Umfeld steigen die gespeicherten Datenvolumina und mit ihnen der Bedarf nach geeigneten Datenanalysesystemen. Die bestehenden, meist auf rechenintensive Anwendungen zugeschnittenen High Performance Computing Systeme, erfüllen die Anforderungen aktueller, datenintensiver Anwendungen jedoch nur unzureichend.

Die fortschreitende Globalisierung und die damit verbundene Notwendigkeit der gemeinsamen Nutzung von Ressourcen über Organisationen hinweg stellt eine weitere neue Herausforderung in der Datenanalyse dar. Darüber hinaus gewinnt die Integration externer Rechen- und Speicherressourcen, etwa von Cloud-Providern, unter Berücksichtigung der Kosten eine immer größere Bedeutung.

Ziel dieser Arbeit ist es daher, eine effiziente Ausführung sowohl von datenintensiven als auch rechenintensiven Anwendungen in globalen, organisationsübergreifenden Umgebungen zu ermöglichen. Die zentrale Problemstellung bildet hierbei die Suche nach einer Zuordnung der Anwendungen auf die vorhandenen Ressourcen, welche einerseits kostengünstig ist und andererseits eine möglichst kurze Verarbeitungsdauer bietet.

Der entwickelte Scheduling-Algorithmus ist speziell für organisationsübergreifend vernetzte kostenbehaftete Ressourcen konzipiert. Die kostengünstige und schnelle Ausführung der Anwendungen erreicht der Algorithmus durch eine möglichst geringe Distanz zwischen den zu verarbeitenden Daten und der Ausführungsressource. Durch die Ausnutzung der hierarchischen Netzwerkstruktur benötigt der Scheduler nur die einfach zu ermittelnden Ressourcenkosten und Ressourcengeschwindigkeiten, sodass die von bestehenden Verfahren benötigten, schwer bestimmbaren Informationen, wie Anwendungslaufzeiten und Netzwerkauslastungen, nicht benötigt werden. Erste Simulationen haben gezeigt, dass der gewählte Ansatz den bestehenden Verfahren in der vorgestellten Umgebung überlegen ist. Des Weiteren wird eine einfachere Variante des Algorithmus bereits erfolgreich im FDA-Miner innerhalb der Daimler AG zur Verarbeitung der Daten der Brennstoffzellenfahrzeugflotte verwendet.

Die gesteigerte Verarbeitungsleistung ermöglicht den Einsatz komplexer Methoden zur Analyse des Alterungsverhaltens der Brennstoffzellensysteme der gesamten Flotte. Mit den im Rahmen dieser Arbeit angepassten und neu entwickelten Alterungszustandsbestimmungsverfahren lässt sich die Degradation eines Brennstoffzellensystems direkt aus den Fahrtdaten berechnen. Aufwendige Labor- und Prüfstandstests können so vermieden werden. Darüber hinaus wird in dieser Arbeit eine Methode zur Bestimmung der dem Alterungsprozess zugrunde liegenden Mechanismen und Einflussfaktoren aus Felddaten vorgestellt.

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Die Anforderungen an die Verarbeitungsleistung von Computersystemen steigen im wissenschaftlichen wie industriellen Umfeld unaufhörlich. Einerseits werden immer aufwendigere Berechnungen durchgeführt, etwa um die Marktentwicklung eines Produkts, die Auswirkungen des Klimawandels oder die Interaktion von Molekülen vorherzusagen. Andererseits steigen auch die digital gespeicherten Datenvolumina, wobei nicht nur die Speicherung, sondern insbesondere die zeitnahe Extraktion aller relevanten Informationen aus diesen Daten in vielen Anwendungsgebieten von entscheidender Bedeutung ist. So generiert allein das internationale Kernforschungszentrum CERN mit dem Large Hadron Collider 15 Petabyte pro Jahr[1]. Internetfirmen wie Google, Yahoo und Facebook arbeiten ebenfalls mit Datenmengen im zweistelligen Petabyte-Bereich[2]. Banken und Versicherungen hingegen vereinen eine Vielzahl unterschiedlicher Datenquellen und nutzen diese zur Unternehmensbewertung, Risikoanalyse oder zum Kreditscoring. Doch auch in anderen Industriezweigen fallen mittlerweile enorme Datenmengen an, etwa Sensorinformationen von Versuchsträgern oder Produktionsanlagen.
Die aus den Daten gewonnen Erkenntnisse fließen wiederum in Simulationsmodelle ein, wodurch die Simulationen exakter, aber auch komplexer und aufwendiger zu berechnen werden. Der Bedarf nach effizienten Verarbeitungssystemen für rechen- *und* datenintensive Applikationen ist daher im wissenschaftlichen wie industriellen Bereich enorm.

Einzelne Organisationen verfügen häufig nicht über die Mittel, die für diese Anwendungen benötigten Ressourcen vorzuhalten, so dass die benötigten Ressourcen von spezialisierten Anbietern angemietet werden oder sich Organisationen zusammenschließen, um ihre Ressourcen gemeinsam zu nutzen. Des Weiteren führt die fortschreitende Globalisierung sowohl im kommerziellen wie auch wissenschaftlichen Umfeld dazu, dass Aufgabengebiete, Zuständigkeiten, Experten, Applikationen sowie Daten geografisch und organisatorisch verteilt sind.
Die bestehenden Systeme und Verfahren sind für die Ausführung rechen- und daten-

[1] http://home.web.cern.ch/about/computing
[2] http://www.facebook.com/notes/facebook-engineering/under-the-hood-hadoop-distributed-filesystem-reliability-with-namenode-and-avata/10150888759153920

intensiver Anwendungen in organisationsübergreifenden Umgebungen nur bedingt geeignet, da sie entweder für die effiziente Berechnung komplexer Simulationen oder die Verarbeitung lokaler Daten konzipiert wurden.

Es werden daher neue Techniken und Methoden benötigt, die sowohl eine effiziente Ausführung komplexer Berechnung als auch die Verarbeitung der stetig wachsenden, verteilten Datenmengen in globalen, organisationsübergreifenden Umgebungen ermöglichen. Die zentrale Herausforderung besteht hierbei darin, daten- und rechenintensive Anwendungen so auf die zu Verfügung stehenden Ressourcen unterschiedlicher Organisationen zu verteilen, dass alle Anwendungen, die um die Ressourcen konkurrieren, zeitnah und kostengünstig verarbeitet werden. Der nächste Abschnitt konkretisiert die Problemstellung des Schedulings ressourcenintensiver Anwendungen. Anschließend wird der Beitrag dieser Arbeit zusammengefasst und der weitere Aufbau der Arbeit erläutert.

1.1 Problemstellung

Ressourcenintensive Applikationen werden heutzutage in den unterschiedlichsten kommerziellen wie wissenschaftlichen Anwendungsfeldern eingesetzt. In den letzten Jahrzehnten lag der Fokus im Forschungsumfeld auf der Weiterentwicklung von Systemen und Verfahren, mit denen sich möglichst viele Prozessoren für die Berechnung komplexer Modelle nutzen lassen. Die aktuelle Generation von Clustercomputern verfügt über mehr als hunderttausend Prozessorkerne, die über spezielle Verbindungen miteinander kommunizieren können[3]. Zur Speicherung der Zwischenergebnisse sowie der eigentlichen Resultate der Berechnungen können die Clustermaschinen auf spezielle Speichersysteme zurückgreifen, die auch mit mehreren Tausend parallelen Zugriffen zurechtkommen und mehr als hundert GB/s Durchsatz bieten[4]. Ein Cluster besteht somit aus auf Berechnungen optimierten Rechenressourcen und aus Speicherressourcen, die über ein Netzwerk mit hoher Bandbreite miteinander verbunden sind. Im wissenschaftlichen Umfeld wurden in den letzten Jahren mehrere Initiativen gestartet, die eine Kooperation unterschiedlicher Einrichtungen zur Verarbeitung ressourcenintensiver Anwendungen zum Ziel hatten. Das Baden-Württemberg Grid[5] als Teil der D-Grid Initiative ist hierbei nur ein Beispiel. Im Rahmen dieser Initiative wurden Cluster der Universitäten Freiburg, Heidelberg, Tübingen, Mannheim, Esslingen und der Verbund Ulm/Konstanz sowie das Karlsruher Institut für Technologie und das Höchstleistungsrechenzentrum (HLRS) Stuttgart den Anwendern für wissenschaftliche Berechnungen zur Verfügung gestellt. Im Allgemeinen verfügen die einzelnen Einrichtungen aber über mehrere, eigenständige Cluster; das HLRS etwa verfügt aktuell über drei Cluster. Die Einrichtungen, beziehungsweise die Cluster, sind untereinander mit bis zu 10 Gb/s Netzwerkverbindungen verbunden.

Die Kooperation unterschiedlicher Einrichtungen ist im universitären Umfeld

[3] http://www.hlrs.de/systems/platforms/cray-xe6-hermit
[4] http://www.lrz.de/services/compute/supermuc/systemdescription
[5] http://www.bw-grid.de

deutlich einfacher zu realisieren als für Unternehmen, bei denen mögliche Kooperationspartner auch häufig Konkurrenten sind. In Unternehmen ist daher vielmehr die Kooperation der eigenen Standorte zur Ausführung ressourcenintensiver Anwendungen interessant. Für Unternehmen, die für ihre eventuell nur sporadisch benötigten rechenintensiven Anwendungen nicht genug eigene Ressourcen vorhalten wollen, wird von unterschiedlichen Anbietern eine komplette Infrastruktur (Infrastructre as a Service, IaaS) bestehend aus Rechenressourcen, Speicherressourcen sowie Netzwerk angeboten, die stundenweise gemietet werden kann. Diese angemieteten Ressourcen müssen zusammen mit den internen Ressourcen, die über mehrere Standorte verteilt sein können, zur Ausführung ressourcenintensiver Anwendungen geeignet kombiniert werden. Die bei Nutzung der externen Ressourcen entstehenden Kosten müssen hierbei ebenso betrachtet werden wie die Kosten für die Netzwerkverbindungen zwischen den Standorten und den externen Anbietern.

Neben rechenintensiven Anwendungen gewinnen extrem datenintensive Anwendungen sowohl in der Wissenschaft als auch im kommerziellen Umfeld immer mehr an Bedeutung. Die Speicherung und Auswertung der Daten aus dem Large Hadron Collider des CERN, der Daten aus dem Next Generation Sequencing oder der Bilder, Beiträge und Nutzungsprofile in sozialen Netzwerken sind hierbei nur einige Beispiele. Aber auch in der Industrie fallen immer größere Datenmengen an. So werden mehrere hundert Sensordaten der weltweit eingesetzten Versuchsfahrzeuge der Daimler Brennstoffzellenflotte, wie in Kapitel 5 näher beschrieben, aufgezeichnet. Zur Vorbeugung von Schäden und zur effizienten Weiterentwicklung müssen die Daten dabei möglichst schnell ausgewertet werden. Deshalb müssen die bei der Alterungsanalyse eingesetzten datenintensiven Analyseapplikationen sowie rechenintensiven Simulationen möglichst effizient ausgeführt werden.

Diese Beispiele zeigen die aktuellen Herausforderungen, die in ressourcenintensiven Anwendungsfeldern auftreten. So können die Ressourcen über mehrere Organisationen verteilt sein und bei der Benutzung können Kosten anfallen. Des Weiteren unterscheiden sich die Bandbreite, die Latenz sowie die Kosten der Netzwerkverbindungen innerhalb eines Clusters und zwischen Clustern oder Organisationen erheblich. Darüber hinaus reicht es nicht mehr aus, nur rechen- *oder* datenintensive Anwendungen effizient auszuführen. Viele neue Anwendungsgebiete bestehen aus datenintensiven *und* rechenintensiven Anwendungen, die effizient unter diesen Umgebungsbedingungen ausgeführt werden müssen.

Für eine effiziente Ausnutzung der Ressourcen und eine zeitnahe Verarbeitung der Anwendungen ist daher eine Ressourcenverwaltung essenziell, da die Benutzer in diesen organisationsübergreifenden Umgebungen nicht mehr in der Lage sind, die für sie am besten geeigneten Ressourcen auszuwählen. Zur Ausführung rechenintensiver Anwendungen muss die Ressourcenverwaltung möglichst schnelle Rechenressourcen auswählen, ohne dabei andere Anwendungen zu vernachlässigen. Für datenintensive Anwendungen hingegen sind die Speichersysteme sowie die Topologie, die Bandbreite und die Kosten des Netzwerkes wichtige Faktoren, die bei der Auswahl des Ausführungsortes von der Ressourcenverwaltung berücksichtigt werden müssen, da die Übertragung von großen Datenmengen über Weitverkehrsverbindungen die ei-

gentliche Verarbeitungsdauer um ein Mehrfaches übersteigen kann. Die Ressourcenverwaltung muss daher einen geeigneten *Scheduler* bereitstellen, der beliebige daten- und rechenintensive Anwendungen auf die Ressourcen unterschiedlicher Organisation verteilt und eine zeitnahe und kostengünstige Ausführung gewährleistet.

Beim Scheduling ressourcenintensiver Anwendungen müssen sowohl die Eigenschaften der Anwendungen als auch die der Umgebung betrachtet werden. Um ein breites Anwendungs- und Umgebungsspektrum abdecken zu können, wird für die nachfolgende Betrachtung ressourcenintensiver Anwendungen ein allgemeines Ausführungs- und Umgebungsmodell verwendet. Ressourcenintensive Anwendungen bestehen hierbei aus einem oder mehreren Programmen, die jeweils eine unterschiedliche Anzahl von Eingabedaten verarbeiten und die in einer definierten Reihenfolge ausgeführt werden müssen. Die Kombination aus *einem* Programm p und den von diesem Programm zu verarbeitenden Eingabedaten $d_1, d_2, ...$ wird im Folgenden als ein *Auftrag* $a = (e, p, d_1, d_2, ...)$ bezeichnet, wobei e den Eintreffzeitpunkt des Auftrags bezeichnet.
Ein Programm benötigt eine bestimmte Anzahl von Prozessorkernen auf *einer* Rechenressource, wobei eine Rechenressource üblicherweise mehrere Prozessorkerne besitzt. Das Programm kann zusätzliche Anforderungen an die Rechenressource definieren, etwa Prozessortyp, benötigter Hauptspeicher oder Betriebssystem, so dass nicht alle Rechenressourcen für die Ausführung eines Programmes geeignet sind. Vor der Ausführung müssen alle Eingabedaten vollständig auf dieser Rechenressource vorhanden sein. Sind die zu verarbeitenden Daten nicht dort gespeichert, müssen diese zunächst von anderen Ressourcen, den *Speicherressourcen*, über Netzwerkverbindungen (*Netzwerkressourcen*) kopiert werden. Nicht alle Rechen- und Speicherressourcen können direkt miteinander Daten austauschen, so dass für Datentransfers ein oder mehrere Zwischenstationen erforderlich sein können. Bei den Zwischenstationen handelt es sich um Speicherressourcen, die beim Transfer einen Datensatz zunächst vollständig zwischenspeichern, bis sie den Transfer zur nächsten Zwischenstation einleiten. Beim Transfer eines Datensatzes können somit mehrere Speicher- und Netzwerkressourcen involviert sein. Es wird angenommen, dass der kürzeste Pfad zwischen zwei Ressourcen bekannt ist und nur dieser für den Datentransfer verwendet werden kann.
Es ist wichtig darauf hinzuweisen, dass eine Komponente in diesem Modell sowohl Rechen- als auch Speicherressource sein kann.

Ein Auftrag benötigt sowohl Rechen- als auch Speicherressourcen, wobei für die beiden Ressourcentypen im Allgemeinen mehrere unterschiedliche Ressourcen für den Auftrag zur Auswahl stehen. So kann ein Programm meist auf mehreren Rechenressourcen ausgeführt werden und ein Datensatz ist auf unterschiedlichen Speicherressourcen redundant gespeichert. Zur Ausführung eines Auftrags muss somit vom *Scheduler* für das Programm eine geeignete Rechenressource und für jeden Datensatz eine Speicherressource gewählt werden. Die zur Übertragung benötigten Zwischenstationen und Netzwerkressourcen ergeben sich dabei implizit aus der Angabe der Rechen- und Speicherressource. Die Zuordnung eines Auftrags auf eine geeignete Rechenressource c und Speicherressourcen $s_1, s_2, ...$ wird mit $z = (w, c, s_1, s_2, ...)$

bezeichnet, wobei w die Wartezeit zwischen dem Eintreffen des Auftrags und des Ausführungsstartes unter dieser Zuordnung bezeichnet. Die Aufgabe des Schedulers besteht nun darin, alle Aufträge A, unter Beachtung deren Abhängigkeiten, den zu Verfügung stehenden Ressourcen zuzuordnen

$$\text{Scheduler} : A \to Z, \quad mit\ Z \in \mathbf{Z}$$

wobei $\mathbf{Z}$ Zuordnungsmöglichkeiten existieren und eine vorgegebene Zielfunktion $\varphi(Z)$ minimiert werden soll:

$$\text{Finde } Z^* \text{ mit } \varphi(Z^*) = \min_{Z \in \mathbf{Z}} \{\varphi(Z)\}$$

Zu den am häufigsten verwendeten Zielfunktionen zählen die mittlere oder maximale Ausführungsdauer τ ($\sum_{z \in Z} \tau_z$, $\max_{z \in Z} \tau_z$) oder die mittleren Kosten κ ($\sum_{z \in Z} \kappa_z$) über alle Aufträge [Pin12] [Bru09].[6] Bei den Kosten κ_z kann es sich prinzipiell um eine beliebige, mit der Ausführungsdauer monoton steigende Funktion handeln. Häufig werden die zeitabhängigen Benutzungskosten der verwendeten Ressourcen (Speicher-, Netzwerk- und Rechenressourcen) als Kostenfunktion κ_z verwendet.

Die Wahl der optimalen Zuordnung für beliebig viele über die Zeit eintreffende Aufträge ist, abhängig von der Zielfunktion, im Allgemeinen sehr schwierig. So ist schon das Problem für Aufträge *ohne* Datensätze auf *einer* Rechenressource die mittlere Ausführungsdauer zu minimieren NP-Vollständig [LRKB77]. Dies legt nahe, dass die erweiterte Problemstellung mit beliebig vielen Datensätzen und mehreren Rechenressourcen für ähnliche Zielfunktionen ebenfalls nicht effizient gelöst werden kann.
Wie Groß die Anzahl möglicher Zuordungen ist aus der die Schedulingalgorithmen wählen können, zeigt die nachfolgende Abschätzung: Sei a die Anzahl der Aufträge, d die mittlere Anzahl der Datensätze pro Auftrag, s die mittlere Anzahl der Speicherressourcen pro Datensatz und r die mittlere Anzahl geeigneter Rechenressourcen für einen Auftrag. Dann ergeben sich $a!$ mögliche Ausführungsreihenfolgen für die Aufträgen. Jeder dieser Aufträg kann einer von r Rechenressourcen zugewiesen werden und für jeden Datensatz eines Auftrages gibt es s mögliche Speicherorte. Die Anzahl der möglichen Zuordnungen kann kann somit wie folgt abgeschätzt werden [PRS05]:

$$|\mathbf{Z}| = a! \cdot r^a \cdot s^d$$

Aufgrund der Größe des Suchraums müssen Schedulingalgorithmen geeignete Methoden einsetzen, um die, unter der vorgegebenen Zielfunktion, optimale Lösung möglichst effizient und exakt zu approximieren. Hierbei können insbesondere die Eigenschaften der Anwendungen und der Umgebung verwendet werden.

[6] Da es sich um Optimierungsprobleme handelt, kann der konstante Term $|Z|$ bei der mittleren Ausführungsdauer sowie den mittleren Kosten vernachlässigt werden

1.2 Beitrag dieser Arbeit

Wie die im vorherigen Abschnitt aufgeführten Beispiele aus dem wissenschaftlichen wie kommerziellen Sektor zeigen, besteht ein großer Bedarf nach Schedulern, die eine effiziente Ausführung komplexer Berechnungen sowie die Verarbeitung der stetig wachsenden, verteilten Datenmengen in globalen, organisationsübergreifenden Umgebungen ermöglichen. Die zentrale Herausforderung besteht hierbei darin, daten- und rechenintensive Anwendungen so auf die zur Verfügung stehenden Ressourcen unterschiedlicher Organisationen zu verteilen, dass alle Anwendungen, die um die Ressourcen konkurrieren, zeitnah, fair und kostengünstig verarbeitet werden. Obwohl bereits etliche Scheduler für diese Problemstellung entwickelt wurden, lassen sich diese in der Praxis schwer einsetzen. Dies liegt unter anderem an den von den bestehenden Schedulern verwendeten, einfachen Zielfunktionen und den benötigten Informationen. Insbesondere benötigen diese Scheduler Informationen über die Kosten, die Geschwindigkeit sowie über die aktuelle und zukünftige Auslastung aller Netzwerkverbindungen. Darüber hinaus erfordern die bestehenden Scheduler auch die Ausführungsdauer der auszuführenden Anwendungen. Unter realen Bedingungen sind diese Informationen kaum verfügbar und können nur geschätzt werden.

Der in dieser Arbeit entwickelte *Dual Objective Hierarchical Scheduling (DOHS)* Algorithmus adressiert genau diese Schwachstellen. Der Algorithmus benötigt weder Netzwerkinformationen noch die Ausführungsdauer der Anwendungen. Statt dessen ist der Schedulingalgorithmus auf die Eigenschaften aktueller, organisationsübergreifender Umgebungen abgestimmt und ist in der Lage, mit den wenigen verlässlichen Anwendungs- und Umgebungsinformationen, die in realen Umgebungen zur Verfügung stehen, eine effiziente Ausführung von organisationsübergreifenden, rechen- und datenintensiven Anwendungen zu realisieren. Der DOHS Algorithmus nutzt hierfür die hierarchische Struktur des Netzwerks sowie die Eigenschaften der gewählten Zielfunktion.
Auf Basis des entwickelten Algorithmus wurde der FDA-Miner implementiert, ein Verarbeitungssystem zur Qualitätsanalyse der Daimler Brennstoffzellenfahrzeugflotte, welches erstmals die Datenanalyse der weltweit verteilt agierenden Versuchsfahrzeuge ermöglicht. Durch die gesteigerte Verarbeitungsleistung des entwickelten Verarbeitungssystems können neue, genauere Analysemethoden entwickelt werden. Die in dieser Arbeit weiterentwickelten Verfahren zur Bestimmung des Alterungszustandes sind mit dem FDA-Miner in der Lage, die Daten der gesamten Flotte zu nutzen und sind speziell auf die verteilte Berechung abgestimmt. Darüber hinaus wurde ein Verfahren zur Analyse der Alterungsursachen von Brennstoffzellen im Fahrzeugeinsatz entwickelt.

1.3 Aufbau der Arbeit

Das nachfolgende Kapitel beschreibt zunächst die aktuellen Ansätze zur Ausführung von rechen- und datenintensiven Anwendungen in organisationsübergreifenden

Umgebungen. Anschließend werden spezialisierte Konzepte und Systeme für die parallele Verarbeitung großer, verteilter Daten vorgestellt. Das darauf folgenden Kapitel stellt den in dieser Arbeit entwickelten DOHS Algorithmus im Detail vor. Hierfür wird zuerst ein geeignetes Umgebungs- sowie Schedulingmodell definiert. Auf deren Basis wird die Schedulingfunktion hergeleitet, welche dem DOHS Algorithmus zugrunde liegt. Darauf aufbauend wird der DOHS Algorithmus für das Scheduling von daten- und rechenintensiven Anwendungen vorgestellt. Im anschließenden Kapitel wird anhand von Simulationen der entwickelte Schedulingalgorithmus mit den bestehenden Verfahren verglichen. Darüber hinaus wird gezeigt, wie die entwickelte Simulationsumgebung zur Anpassung der Parameter des DOHS Algorithmus an reale Umgebungen genutzt werden kann. Das folgende Kapitel widmet sich dem FDA-Miner, einem im Rahmen dieser Arbeit implementierten Verarbeitungssystem zur Analyse der Sensordaten der Daimler Brennstoffzellenfahrzeuge, das die Anwendungen nach einer ersten, einfachen Version des DOHS-Algorithmus auf die Ressourcen verteilt und damit eine effiziente Ausführung der daten- sowie rechenintensiven Analyseapplikationen ermöglicht. Ein weiteres Kapitel erläutert ein in dieser Arbeit entwickeltes Analyseverfahren, das die Bestimmung des Alterungszustandes sowie der möglichen Alterungsursachen allein aus den Sensordaten ohne zusätzliche, aufwändige Laboruntersuchungen ermöglicht.
Mit einer Zusammenfassung der im Rahmen dieser Arbeit entwickelten Algorithmen und Analyseverfahren sowie einer Reihe von sinnvollen Erweiterungen für zukünftige, weiterführende Arbeiten schließt diese Arbeit.

Kapitel 2

Grundlagen und bestehende Verfahren

Sowohl im wissenschaftlichen wie auch im kommerziellen Umfeld haben sich Anwendungen etabliert, die sehr hohe Rechenleistung sowie Speichergeschwindigkeit und -kapazität benötigen. Um die Anforderungen dieser Anwendungen zu erfüllen, wurden verschiedenste spezielle Hard- und Software-Systeme entwickelt. Mit der steigenden Leistungsfähigkeit der entwickelten Verarbeitungssysteme erschließen sich auch immer neue Anwendungsfelder, die wiederum nach noch mehr Leistung verlangen, so dass die bestehenden Systeme und Verarbeitungsverfahren stets weiterentwickelt werden müssen. Die stetig steigenden Anforderungen erfordern neue Konzepte, um die Effizienz der Ressourcennutzung zu erhöhen. Dies kann unter anderem durch die zeitlich begrenzte Nutzung externer Ressourcen sowie der Zusammenarbeit unterschiedlicher Organisationen bei ressourcenintensiven Anwendungsfeldern erzielt werden.

Für die effiziente Ausführung von ressourcenintensiven Anwendungen wurden bereits unterschiedliche, spezialisierte Verarbeitungssysteme entwickelt. Insbesondere für rechenintensive Anwendungen existieren zahlreiche Systeme, wobei oftmals, wie im nächsten Abschnitt erläutert, Berechnungscluster eine zentrale Rolle spielen. Durch die immer schneller wachsenden Datenmengen wurden in den letzten Jahren auch unterschiedliche Systeme zur effizienten Verarbeitung großer Datenmengen entwickelt, von denen die Wichtigsten im Folgenden vorgestellt werden. Im Vergleich zu rechenintensiven Verarbeitungssystemen sind die Systeme zur effizienten, skalierbaren Verarbeitung großer Datenmengen noch relativ jung. Neben der Verbesserung der Verarbeitungsleistung für rechen- sowie datenintensive Anwendungen liegt der Fokus aktueller Entwicklungen vor allem darauf, das Kosten/Nutzen-Verhältnis noch weiter zu optimieren, und die Nutzung externer Ressourcen sowie die Zusammenarbeit unterschiedlicher Organisationen bei ressourcenintensiven Anwendungsfeldern zu erleichtern.

Die zentrale Aufgabe der Verarbeitungssysteme besteht somit für ressourcenintensive Anwendungen im *Scheduling*: Die auszuführenden Anwendungen so auf die zur Verfügung stehenden Ressourcen unterschiedlicher Organisationen zu vertei-

len, dass alle Anwendungen, die um die Ressourcen konkurrieren, zeitnah, fair und kostengünstig verarbeitet werden. Die größte Herausforderung für den Schedulingalgorithmus ist dabei eine effiziente Ausführung ressourcenintensiver Anwendungen in dynamischen, heterogenen, organisationsübergreifenden Umgebungen zu ermöglichen.

Die nächsten Abschnitte erläutern die Konzepte für Verarbeitungssysteme, die eine Zusammenarbeit unterschiedlicher Organisationen sowie die Integration externer Ressourcen ermöglichen und eine möglichst effiziente Ausführung von rechen- und datenintensiven Anwendungen anstreben. Anschließend werden aktuelle Schedulingverfahren für diese Umgebungen detailliert erläutert und auf ihre Eignung für reale Umgebungen geprüft. Das Kapitel endet mit einem Überblick über bestehende Konzepte und Verfahren zur skalierbaren Verarbeitung großer Datenmengen mit ihren Vor- und Nachteilen in Bezug auf rechen- und datenintensive Anwendungen.

2.1 Grid-Computing

In den letzten Jahrzehnten wurden zahlreiche Verarbeitungssysteme insbesondere für rechenintensive Anwendungen entworfen. Da die Leistungsfähigkeit einzelner Rechenressourcen - einzelner Prozessorkerne - aufgrund physikalischer Grenzen[SL05] nicht beliebig gesteigert werden konnte, bestehen aktuelle Verarbeitungssysteme für rechenintensive Anwendungen immer aus mehreren, verbundenen Ressourcen. Hierbei haben sich Berechnungscluster zum Standard für rechenintensive Anwendungsfelder entwickelt.[1] Ein Berechnungscluster besteht aus mehreren Rechnern, die über ein Hochgeschwindigkeitsnetzwerk (InfiniBand, Myrinet)[2] miteinander verbunden sind. Die niedrige Latenz und hohe Bandbreite dieser Netzwerke ermöglicht eine effiziente Kommunikation zwischen den Rechenressourcen, so dass einzelne Anwendungen mehrere Ressourcen nutzen können. Eine Kommunikationsmiddleware (MPI, PVM)[3] stellt Bibliotheken für die Entwicklung und Ausführung verteilter, insbesondere paralleler, Anwendungen zur Verfügung. Eine Ressourcenverwaltung regelt den Zugriff auf die Rechenressourcen und gewährleistet eine möglichst optimale Auslastung der Ressourcen sowie die zeitnahe Verarbeitung der Anwendungen. Anwendungen melden sich mit ihrem Ressourcenbedarf vor der Ausführung bei der Ressourcenverwaltung an und bekommen, falls vorhanden, geeignete Rechenressourcen zugeteilt. Die Ein- und Ausgabedaten der Anwendungen werden auf einem Speichersystem gespeichert, auf das alle Rechenressourcen direkten Zugriff haben. Anstelle eines großen Speichersystems werden häufig Speichersysteme eingesetzt, bei denen mehrere Speicherressourcen durch ein Paralleles Verteiltes Dateisystem (Lustre, GPFS)[4] zu einem großen, performanten System zusammengeschlossen werden. Ein Hochdurchsatznetzwerk (10 GigEthernet, InfiniBand) verbindet die einzelnen

[1] Beispiele aktueller Cluster sind der SuperMuc (http://www.lrz.de/services/compute/supermuc) oder Hermit (http://www.hlrs.de/systems/platforms/cray-xe6-hermit)
[2] InfiniBand: http://www.infinibandta.org, Myrinet: http://www.myricom.com
[3] MPI: http://mpi-forum.org, PVM: http://www.csm.ornl.gov/pvm
[4] Lustre: http://www.lustre.org, GPFS: http://www.ibm.com/systems/software/gpfs

Speicherressourcen und die Rechenressourcen.
Die Funktionsweise eines solchen Berechnungsclusters soll an folgendem Beispiel erläutert werden:

- Die Anwendung übermittelt das auszuführende Programm und dessen Anforderungen an die Ressourcenverwaltung.
- Die Ressourcenverwaltung wählt und reserviert die geeigneten Rechenressourcen und startet das Programm.
- Aufbau der Interprozess- oder Interressourcenkommunikation durch die Kommunikationsmiddleware.
- Abfrage des Speicherorts der Daten vom Verteilten Dateisystem. Lesen der Eingabedaten von den entsprechenden Speicherressourcen.
- Durchführung der Berechnung mit oder ohne Kommunikation zwischen den Rechenressourcen.
- Abfrage der Speicherorte für die Ergebnisse vom Verteilten Dateisystem. Schreiben der Ergebnisse auf den ausgewählten Speicherressourcen.

Der vorgestellte Aufbau ist speziell auf die Anforderungen rechenintensiver Anwendung zugeschnitten. Durch die konsequente Fokussierung auf rechenintensive Anwendungen können einfache Rechen- sowie Speicherressourcen eingesetzt werden und die Anzahl der Ressourcen exakt auf die Anforderungen angepasst werden. Dieser Ansatz trägt durch die geringeren Anschaffungs- als auch Wartungskosten zu einem optimierten Kosten/Nutzen-Verhältnis der Berechnungscluster bei. Aus den Bemühungen das Kosten/Nutzen-Verhältnis noch weiter zu optimieren und die Zusammenarbeit unterschiedlicher Organisationen bei rechenintensiven Anwendungsfeldern zu erleichtern entstand das *Grid-Computing* [FK99]. Grids ermöglichen insbesondere einen einheitlichen Zugriff auf mehrere Cluster unterschiedlicher Organisationen. Damit stehen einzelnen Anwendungen einerseits mehr Ressourcen zur Verfügung und andererseits kann die Auslastung insgesamt verbessert werden. Darüber hinaus gewinnt mit der zunehmenden Globalisierung die organisationsübergreifende, weltweite Zusammenarbeit in vielen ressourcenintensiven Anwendungsgebieten immer größere Bedeutung. Gekapselte Netze, heterogene Hard- und Softwarelandschaften sowie unterschiedliche Richtlinien sind hierbei nur einige Herausforderungen, die das Grid-Computing adressiert [RBJS03].

Im Folgenden wird genau auf die einzelnen Aspekte des Grid-Computings eingegangen, um eine klare Abgrenzung zu anderen verteilten Systemen herzustellen. Anschließend wird die oben genannte Grundidee präzisiert und die Herausforderungen des Schedulings in Grid-Umgebungen vorgestellt.

2.1.1 Eigenschaften des Grid-Computings

Der Begriff Grid-Computing wurde als Analogie zum Stromnetz, dem *Power Grid*, gewählt [FK99]. Ein Computer-Grid-System stellt nach dieser Vorstellung den Nutzern verschiedenste Ressourcen so einfach zur Verfügung, wie Strom aus der Steckdose. Um dies zu erreichen, wird eine weltweite Vernetzung von verschiedenen Computerressourcen, wie Supercomputer, Arbeitsplatzrechner, Speichersysteme oder Sensorsystemen, angestrebt. Diese Idee ist jedoch keineswegs neu, denn schon 1969 schrieb Len Kleinrock[5]:

> „We will probably see the spread of 'computer utilities', which, like present electric and telephone utilities, will service individual homes and offices across the country."

Aufgrund dieser fast allumfassenden Grundidee und einer ebenso wagen (ersten) Definition, entstand der Eindruck, Grid-Computing sei nichts Weiteres als eine Marketingstrategie[Fos02]. In jüngster Zeit haben die Marketingabteilungen einen weiteren Namen für eine ähnliche Idee geprägt: *Cloud-Computing*. Als Cloud-Computing werden Anwendungs-, Plattform- und Infrastrukturdienste bezeichnet, die von einem Anbieter über das Internet bereitgestellt werden [AFG+09] und meist auf virtuellen Plattformen basieren. Cloud-Computing wird häufig als neues, eigenständiges Konzept disjunkt zum Grid-Computing betrachtet. Hierbei werden jedoch die im Cloud-Computing beschriebenen Funktionen mit den ersten Definitionen des Grid-Computing verglichen. Vergleicht man Cloud-Computing hingegen mit der aktuellen, nachfolgend beschriebenen Grid-Definition, stellt man fest, dass Cloud-Computing und Grid-Computing, insbesondere im Bereich Scheduling, fast identische Problemstellungen aufweisen [FZRL08].

Grid-Systeme bauen auf den Ideen von etablierten Verteilten Systemen auf und erweitern diese um zusätzliche Eigenschaften. Die bekannten Clusterverwaltungssysteme (Condor, LSF, PBS, ...)[6][7][8] können als direkte Vorgänger von Grid-Verwaltungssystemen angesehen werden. Der Übergang von komplexen Clusterumgebungen zu Grids ist allerdings fließend. Wie die meisten Technologien hat sich auch das Grid im Laufe der Zeit weiterentwickelt. Das Ziel der ersten Projekte war die Skalierung der Prozessornutzung über große Distanzen und die Entwicklung eines virtuellen Supercomputers. Die Erfahrungen aus diesen Projekten führten zu neuen Anforderungen, die in nachfolgenden Projekten aufgegriffen wurden. In den nächsten Abschnitten werden die Evolutionsstufen des Grids genauer betrachtet, wobei sich die Evolution auch in den Definitionen widerspiegelt.

Schon bevor der Begriff Grid-Computing eingeführt wurde, existierten Projekte, die sich mit einigen der oben genannten Probleme auseinandersetzten. Die Projekte I-WAY und FAFNER können als die ersten Grid-System angesehen werden

[5] http://www.lk.cs.ucla.edu/data/files/Kleinrock/A%20Vision%20for%20the%20Internet.pdf
[6] Condor: http://research.cs.wisc.edu/htcondor
[7] LSF: http://www.ibm.com/systems/technicalcomputing/platformcomputing/products/lsf
[8] http://PBS: www.pbsworks.com

[RBJS03]. Obwohl beide unterschiedliche Ziele verfolgten, befassten sie sich mit ähnlichen Problemen. Die Hauptprobleme waren hierbei die Skalierung, die Sicherheit sowie die Heterogenität der lokalen Ressourcen. Ein Teil der Lösung dieser Probleme bestand aus einer neuen Abstraktionsschicht oberhalb der Rechencluster. Diese stellte nach außen eine einheitliche Schnittstelle für Authentifizierung, Ressourcenverwaltung, Prozessverwaltung und Kommunikationsfunktionen bereit. Nach innen bildeten sie die Funktionen auf das lokale System ab.

Die Erfolge der ersten Generation steigerten das Interesse an Grid-Systemen. Aus den Erfahrungen bei der Umsetzung der ersten Grid-Systeme und der Weiterentwicklung des Internets ergaben sich neue Anforderungen. Die Grid-Systeme der zweiten Generation wurden daher nicht mehr für spezielle Anwendungsgebiete entwickelt, sondern für die Vernetzung beliebiger Ressourcen über das Internet. Hierbei wurden vier wichtige Eigenschaften herausgearbeitet (nach [Fos02]):

- *Geografische Ausdehnung* beschreibt die Eigenschaft, dass die Ressourcen eines Grids über die ganze Welt verteilt sein können. Durch die globale Verteilung und die Integration verschiedenster Netzwerke, entstehen unter anderem Laufzeit- und Bandbreitenprobleme. Die globale Größe impliziert auch, dass ein Grid viele Ressourcen verwalten muss.

- *Administrative Ausdehnung* beschreibt die Eigenschaft, dass die Ressourcen eines Grids in verschiedenen administrativen Bereichen liegen können. Diese administrativen Bereiche können Abteilungen innerhalb einer Organisation aber auch unterschiedliche Organisationen sein, die innerhalb ihres Bereiches vollkommen eigenständig bleiben. In einem Grid existiert somit keine zentrale Kontrollinstanz.

- *Heterogenität* beschreibt die Eigenschaft, dass ein Grid aus vielen verschiedenen Ressourcen bestehen kann. Ein Grid-System muss in der Lage sein, alle verfügbaren Ressourcen zu integrieren. Das Spektrum reicht hierbei von gewöhnlichen Arbeitsplatzrechnern über Supercomputer bis hin zu Laborgeräten oder anderen technischen Systemen.

- *Dynamik* beschreibt die Eigenschaft, dass Ressourcen eines Grids sehr unterschiedliche Lebenszyklen besitzen. Ressourcen können für Jahre oder nur für Minuten zur Verfügung stehen. Ebenso werden ständig Ressourcen hinzugefügt oder entfernt. Auch Fehler sind in einem Grid nicht die Ausnahme, sondern die Regel. Das Grid-System muss Fehler erkennen und die Auswirkung auf andere Ressourcen so gering wie möglich halten.

Zu den Grid-Systemen der zweiten Generation, die diese Anforderungen erfüllen, zählen unter anderem Condor, die Sun Grid Engine und das Globus Toolkit 2 [RBJS03].

Die Grid-Systeme der zweiten Generation boten die geforderte Skalierbarkeit und wurden erfolgreich eingesetzt. Doch schon während der Entwicklung kristallisierten sich neue Anforderungen für die aktuelle dritte Generation heraus. Um die

Entwicklung neuer Grid-Applikationen zu erleichtern, sollte es möglich sein, vorhandene Applikations- und Systemkomponenten wiederzuverwenden und diese in flexibler Weise zu kombinieren. Des Weiteren sollten die neuen Systeme eine gewisse Autonomie besitzen, so dass sie leichter administrierbar und wartbar sind, um eine globale Ausdehnung überhaupt zu ermöglichen. Die Lösungen dieser neuen Anforderungen umfassen ein serviceorientiertes Modell, virtuelle Organisationen, standardisierte Protokolle und Schnittstellen sowie den Einsatz von Meta-Daten. Ein erstes Grid-System, das als Referenzimplementierung der dritten Generation angesehen werden kann, ist das Globus Toolkit 4 [RBJS03].

Der erste Versuch, ein Grid zu definieren wurde in [FK99] unternommen:

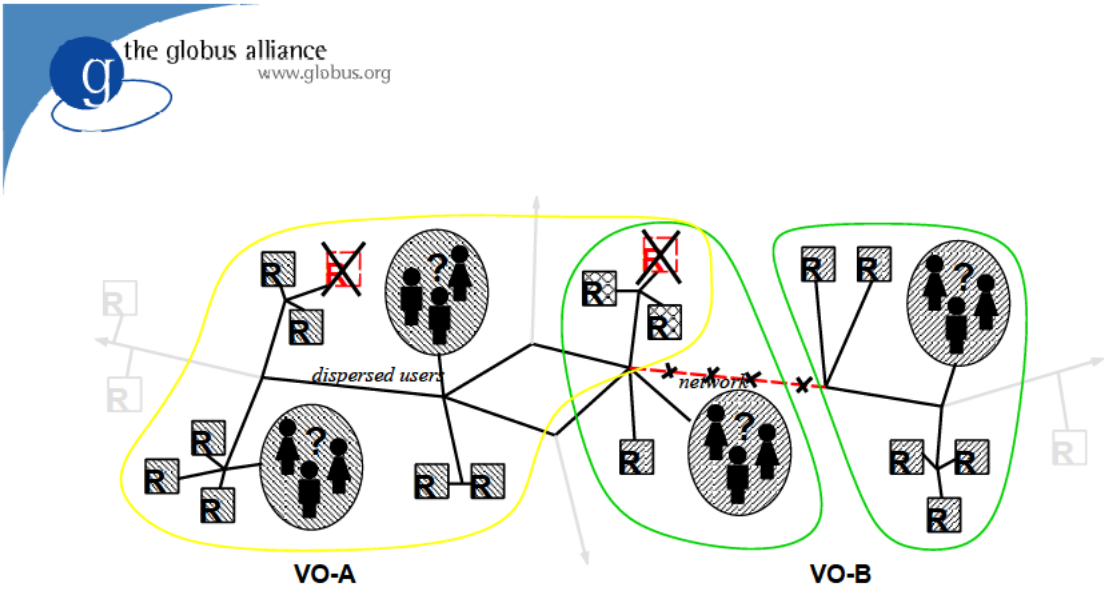


Abbildung 2.1: Das Konzept der Ressourcenteilung in Virtuellen Organisationen (aus [Fos05]).

Diese Definition setzt sich nur mit der Bereitstellung von Prozessoren auseinander. Wie leicht zu erkennen ist, umfasst diese Definition fast alle aktuellen verteilten Systeme. Dies wurde auch von den Autoren erkannt, worauf die Definition überarbeiteten und zusätzliche Aspekte, wie Skalierbarkeit, Sicherheit und Interoperabilität in die Definition eingearbeitet wurden. Darüber hinaus wurde der Begriff *Virtuelle Organisation* (VO) eingeführt. Eine VO ist ein dynamischer Zusammenschluss von Individuen und/oder Organisationen, welche die gemeinsamen Ressourcen nach einem wohl definierten Regelwerk nutzen. Abbildung 2.1 zeigt wie sich unterschiedliche Virtuelle Organisationen zusammenschließen, um ihre Ressourcen (Personen, Computer, Messgeräten) gemeinsam zu nutzen. Wie die Abbildung zeigt, können hierbei Ressourcen jederzeit ausfallen oder neu hinzukommen.

Die folgende Definition fasst die Aspekte der Ressourcenteilung mit dem Konzept der VO zusammen [FKT01]:

> „Grid computing is concerned with coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations.“

Eine ausführlichere Formulierung dieser Definition findet sich ebenfalls in [FKT01], in der insbesondere die Interoperabilität als wichtiger Aspekt hervorgehoben:

> „The sharing that we are concerned with is not primarily file exchange but rather direct access to computers, software, data, and other resources,[...]. This sharing is, necessarily, highly controlled, with resource providers and consumers defining clearly and carefully just what is shared, who is allowed to share, and the conditions under which sharing occurs. A set of individuals and/or institutions defined by such sharing rules form what we call a virtual organization.[...]Interoperability is thus the central issue to be addressed. In a networked environment, interoperability means common protocols.“

Fasst man die wichtigsten Aspekte der obigen Definitionen zusammen, ist ein Grid nach [Fos02] ein Verteiltes System, welches:

1. *Ressourcen koordiniert, die keiner zentralen Kontrolle unterliegen...* Ein Grid integriert und koordiniert Ressourcen und Nutzer, welche in unterschiedlichen Kontrolldomänen angesiedelt sind und übernimmt gleichzeitig weitere Aufgaben wie Sicherheit, Bezahlung, Einhaltung von vorgegebenen Regeln,...

2. *... in dem es standardisierte, offene, allgemeine Protokolle und Schnittstellen benutzt ...* Ein Grid baut auf standardisierten, offenen, allgemeinen Protokollen und Schnittstellen auf, welche für grundlegende Aufgaben wie Authentifizierung, Autorisierung, Ressourcenlokation und Ressourcenzugriff zuständig sind. Sind diese Protokolle und Schnittstellen nicht offen und standardisiert, ist die Interoperabilität stark eingeschränkt und es handelt sich um ein anwendungsspezifisches System.

3. *... um nichttriviale Dienstgüte (Quality of Service; QoS) zu bieten.* Ein Grid erlaubt es seinen Teilnehmern, die Ressourcen in koordinierter Form zu nutzen, um dadurch verschiedenste Arten von erhöhter Dienstgüte zu erzielen (wie beispielsweise Durchsatz, Verfügbarkeit oder Sicherheit), so dass das kombinierte System einen signifikant höheren Nutzen bietet als die Summe der einzelnen Einheiten.

2.1.2 Grid-Ressourcenverwaltung

Die zentrale Aufgabe eines Grid-Systems ist es, die Ressourcen aller beteiligten Organisationen zu verwalten. Die im vorherigen Abschnitt erläuterten Eigenschaften von Grids - Heterogenität und Dynamik der Ressourcen sowie geografische und administrative Ausdehnung - erfordern geeignete Methoden, um alle Ressourcen mit

ihrem aktuellen Zustand zu erfassen sowie eingehenden Ressourcenanfragen effizient auf passende Ressourcen abzubilden [TTL03].

Aus Sicht ressourcenintensiver Anwendungen zählen die Rechen- und Speicherressourcen sowie die Netzwerkverbindungen zu den wichtigsten Ressourcen. Durch die Dynamik eines Grids können Rechen- und Speicherressourcen jederzeit nicht mehr verfügbar sein oder neu ins System kommen. Diese Dynamik ist auch der administrativen Unabhängigkeit der einzelnen Organisationen geschuldet, so dass diese *ihre* Ressourcen zu jedem Zeitpunkt anderen Aufgaben zuteilen können. Die Auslastung der Netzwerkverbindungen kann ebenfalls nur schwer ermittelt werden. Zum einen ist hierbei die geografische Ausdehnung eines Grids ein wichtiger Faktor, so dass die Netzwerkverbindungen zwischen Organisationen oder sogar innerhalb von Organisationen über das Internet führen. Zum anderen werden die Netzwerkverbindungen einer Organisation nicht nur für die Verarbeitung von Berechnungsaufträgen verwendet, sondern auch für andere Applikationen und Dienste der Organisation.

Die Vielzahl an Benutzern sowie unterschiedlichen Benutzergruppen eines Grids und die damit einhergehende große Bandbreite der eingesetzten Anwendungen führen zu einer weiteren Herausforderung für den der Ressourcenverwaltung zugrunde liegenden Schedulingalgorithmus. So lässt sich die Anzahl der zu verarbeitenden Aufträge, die in einem Zeitintervall eintreffen werden, kaum vorhersagen. Aufgrund der vielen unterschiedlichen Anwendungen sowie Anwendungsfeldern kann darüber hinaus die Laufzeit einer konkreten Ausführung nur schwer abgeschätzt werden. Die von Scheduler zu bestimmende, effiziente Zuordnung der Verarbeitungsanfragen auf die Ressourcen gestaltet sich durch die inexakten Ressourcen- sowie Anwendungsinformationen äußerst schwierig.

Eine exakte Bestimmung der Verfügbarkeit der zu verarbeitenden Aufträge sowie der zukünftigen Auslastung der Ressourcen ist damit kaum möglich. Die Ressourcenverwaltung, insbesondere der Schedulingalgorithmus, muss diese besonderen Eigenschaften von Grid-Umgebungen berücksichtigen.

2.1.3 Bestehende Grid-Scheduler

In den letzten Jahren wurden eine Reihe unterschiedlicher Verfahren zum Scheduling in Grids entwickelt [AD06]. Der Fokus dieser Entwicklungen hat sich im Laufe der Zeit von reinen rechenintensiven Anwendungsfeldern immer mehr auf gemischte, daten- und rechenintensive Anwendungen verschoben [Kos12]. Die nachfolgend beschriebene Auswahl an für diese Arbeit besonders relevanten Grid-Schedulern wurden meist für bestimmte Einsatzzwecke entwickelt und unterscheiden sich in ihrer Behandlung von verteilten Berechnungen, der Berücksichtigung von Daten sowie komplexen Abhängigkeiten von Aufträgen. Insbesondere wird im folgenden Abschnitt die Eignung der Scheduler zur effizienten Verarbeitung von daten- *und* rechenintensiven Anwendungen in heterogenen, dynamischen, organisationsübergreifenden Umgebungen untersucht.

Die Grundidee der ersten Grids - rechenintensive Anwendungen effizient über Organisationsgrenzen hinweg auszuführen - führte zur Entwicklung von Schedulern,

die mit unterschiedlichen Heuristiken die auszuführenden Anwendungen auf die zur Verfügung stehenden Rechenressourcen verteilen [KDD+03] [BAG00] [FTL+02]. Eine erste Weiterentwicklung bilden Scheduler, die auch Verteilte Berechnungen auf Basis von MPI unterstützen [ABD+01] [HML04] [QPTGG+12] [GVBB13]. Bei diesen Schedulern werden die Daten im Scheduling nicht weiter betrachtet. Es wird lediglich die Möglichkeit zum Datentransfer angeboten.

Die zunehmende Bedeutung von datenintensiven Anwendungen forcierte die Entwicklung von Schedulingverfahren unter Berücksichtigung von Daten. Die gewählten Ansätze unterscheiden sich insbesondere im Hinblick auf das gewählte Umgebungsmodell, die aktive Replikation von Daten, der Anzahl der erlaubten Datensätze pro Auftrag sowie dadurch, ob die Abhängigkeiten zwischen Aufträgen beachtet werden.

Ein Zweig der Entwicklungen befasst sich primär mit der Verteilung der Daten und überlässt anderen Algorithmen das eigentliche Scheduling[TMX+13]. Das Ziel dieser Verfahren ist, eine Verteilung und Replikation der Daten zu erzeugen, so dass ein geeignetes Scheduling die Ausführung der Anwendungen beschleunigen kann[CDL+07].

Ein Vergleich unterschiedlicher Strategien für das getrennte Scheduling von Daten und Aufträgen findet sich in [RF02a]. Ein Auftrag umfasst hierbei ein Programm, das einen Prozessorkern belegt und einen Datensatz verarbeitet. Eine Organisation besteht aus Sicht des Schedulers aus einer Rechenressource mit mehreren Prozessorkernen und einer Speicherressource. In den durchgeführten Simulationen werden die Aufträge randomisiert, lastabhängig oder in Abhängigkeit des Datenspeicherortes auf die Organisationen verteilt. Gleichzeitig werden die vorhandenen Datensätze nicht repliziert oder randomisiert, beziehungsweise lastabhängig auf andere Organisationen repliziert. Die Autoren kommen zu dem Schluss, dass ein Auftrag am besten in einer Organisation ausgeführt werden sollte, welche den zu verarbeitenden Datensatz vorhält und dass häufig verwendete Datensätze repliziert werden sollten.

Das in [ME04] beschriebene Schedulingverfahren eignet sich für Verteilte Berechnungen, die mehrere Prozessorkerne und eine Eingabedatei benötigen, die auf mehreren Organisationen repliziert vorliegen kann. Eine Organisation besteht aus Sicht des Schedulers aus einer Rechenressource mit mehreren Prozessoren und einer Speicherressource. Der Scheduler versucht einen Berechnungsauftrag zunächst einer Organisation zuzuordnen, die sowohl genügend freie Prozessoren als auch den zu verarbeitenden Datensatz bereitstellt. Ist dies nicht möglich, wird eine Rechenressource mit genügend freien Prozessoren gewählt, für die die geschätzte Transferdauer minimal ist. Die Transferdauer wird auf Basis der Datengröße sowie der Bandbreite der Netzwerkverbindung zwischen den Organisationen bestimmt.

Ein weiterer Ansatz für verteilte Berechnungen, die mehrere Prozessorkerne benötigen und eine Eingabedatei verarbeiten, ist das in [PK03] beschriebene Schedulingverfahren. Das Verfahren versucht die Kosten der Ausführungen zu minimieren. Eine Organisation besteht aus Sicht des Schedulers aus einer Rechenressource mit mehreren Prozessorkernen und einer Speicherressource, die durch ein lokales Netzwerk miteinander verbunden sind. Der Scheduler bestimmt die beste Zuordnung für einen Auftrag, indem er für jede Kombination dieser virtuellen Speicher- und Re-

chenressourcen die Kosten berechnet. Das gewählte Kostenmodell basiert auf der Datengröße der Anwendung, der Ein- und Ausgabedaten, der Bandbreite der Netzwerkverbindungen innerhalb und zwischen den Organisationen sowie der benötigten Prozessorzeit der Anwendung.

Eine Erweiterung des in [HML04] beschriebenen GridWay-Schedulers erlaubt es den Benutzern, über Anforderungs- und Rangparameter die Auswahl der Rechen- und Speicherressourcen für beliebig viele Dateien in Abhängigkeit der Datengröße und Datenposition selbst zu bestimmen[DPHHL10]. Dem Benutzer stehen für jeden Datensatz die Parameter `CLOSE_DATA, HAS_CLOSE_DATA, SIZE_CLOSE_DATA` zur Verfügung, welche die Lokalität und die Größe des Datensatzes wiederspiegeln. Auf jeder potentiellen Rechenressource wird ein auf Basis dieser Parameter frei definierbarer, arithmetischer Ausdruck ausgewertet. Der Scheduler wählt diejenige Ressource, für die der Ausdruck maximal ist.
Eine weitere Erweiterung des GridWay-Schedulers ordnet die Aufträge den Ressourcen anhand deren geschätzten Ausführungs- sowie Transferdauern zu [TCR+12].

Der in [MAS+07] beschriebene Scheduler erfordert, im Gegensatz zum zuvor vorgestellten Scheduler, keine Benutzerangaben, um eine Anwendung mit beliebig vielen Datensätzen auf die Ressourcen eines Grids zu verteilen. Der Scheduler agiert dabei in drei Schritten. Zunächst werden die besten n - im Beispiel fünf - Organisationen aufgrund der verfügbaren Rechenleistung und der aktuellen Auslastung sowie ihrer Datenspeicherkosten ausgewählt. Anschließend bestimmt der Scheduler für jede dieser Organisationen die Kosten (für die Benutzung einer Ressource fallen Kosten pro Zeiteinheit an) für die Ausführung des Auftrags. In die Kostenabschätzung fließen die Datengröße der Anwendung sowie der Ein- und Ausgabedaten, die Netzwerkverbindung zwischen den Organisationen sowie der Berechnungsaufwand des Auftrags mit ein. Zuletzt wird die Organisation mit den geringsten Kosten gewählt.
In dem entwickelten Verfahren nehmen die Netzwerkverbindungen zwischen den Organisationen eine zentrale Rolle ein. Auf Basis von Latenz, verworfenen Paketen, Durchsatz sowie der Varianz der Laufzeit von Datenpaketen werden die Kosten zur Datenübertragung zwischen den Organisationen abgeschätzt. Nachdem eine Organisation gewählt wurde, nutzt der Scheduler die berechneten Netzwerkosten, um diejenigen Repliken der zu verarbeitenden Daten zu ermittelten, welche die geringsten Transferkosten erzeugen.

Ein ähnlicher Ansatz wird in [VB05] und [VB06] beschrieben. Der Schedulingalgorithmus erzeugt eine möglichst gute Zuordnung für jeden einzelnen Auftrag und ändert die Ausführungsreihenfolge der Aufträge, um die vorgegebene Zielfunktion für alle Aufträge zu minimieren. Als Zielfunktionen stehen entweder die Kosten oder die maximale Dauer der Ausführung zur Verfügung. Die Zuordnung eines Auftrags zu einer Organisation erfolgt unter Berücksichtigung der Datengröße der Eingabedaten, der geschätzten Bandbreite der Netzwerkverbindung zwischen den Organisationen sowie des Berechnungsaufwands.

Anstelle eines heuristischen Ansatzes wird in [PRS05] ein Genetischer Algorithmus für das Scheduling von datenintensiven Anwendungen verwendet. Der Scheduler führt dabei ein gleichzeitig ein Scheduling (Co-Scheduling) von Daten und Aufträgen

durch. Hierbei ordnet der Algorithmus die Aufträge den Rechenressourcen zu und bestimmt gleichzeitig eine Replikation der zu verarbeitenden Datensätze. Diese Problemstellung wird als Gensequenz kodiert. Das Gen eines Individuums besteht aus der Reihenfolge der Aufträge, der Zuordnung der Aufträge zu den Rechenressourcen und den Zuordnungen Datensatze zu den Speicherressourcen. Jedes Individuum repräsentiert damit eine mögliche Zuordnung inklusive der Datenreplikation.
Der evolutionäre Prozess startet mit mehreren randomisiert erzeugten Individuen. Für jedes Individuum werden die Kosten auf Basis der Datengröße der Eingabedaten, der geschätzten Bandbreite der Netzwerkverbindung zwischen den Organisationen sowie des Berechnungsaufwands bestimmt. Durch wiederholte Rekombination und Mutation werden die Kosten minimiert.
Bei a Aufträgen, r Rechenressourcen, d Datensätzen und s Speicherressourcen ergeben sich $a! \cdot r^a \cdot s^d$ mögliche Kombinationen. Die Größe des Suchraums erfordert daher eine entsprechend große Population sowie eine Vielzahl an Generationen.

Neben den soeben vorgestellten Verfahren zur Ausführung beliebiger Anwendungen existiert eine Vielzahl von spezialisierten Schedulern. Die Anwendung der *Divisible Load Theory* für datenintensive Problemstellungen umfasst Verfahren, die zunächst die zu verarbeitenden Daten aufteilen und dann auf eine Reihe von Ressourcen zur Verarbeitung verteilen [LZ02] [CM10]. Die Daten werden hierbei von einer Ressource - der Wurzel - auf die anderen Ressourcen transferiert und dort verarbeitet. Die Aufgabe des Schedulers besteht dabei darin, eine zulässige Aufteilung der Daten sowie eine Zuordnung auf die Ressourcen zu finden, die unter einer vorgegebenen Zielfunktion optimal ist [BGR03]. Die Verfahren verwenden neben der Datengröße und der zulässigen Datenaufteilung auch Informationen über die Netzwerkverbindung sowie den Berechnungsaufwand. Mehrere Arbeiten befassen sich dabei mit den speziellen Herausforderungen von Grids, wobei insbesondere die Optimierung des Netzwerktransfers im Vordergrund steht [ZTV08], [TRA+06], [ZK08].

2.1.4 Einschränkungen der Grid-Scheduler

Die im vorherigen Abschnitt vorgestellten, bestehenden Grid-Schedulingverfahren basieren auf einer oder mehreren der folgenden Annahmen:

- Dem Schedulingalgorithmus sind alle Verarbeitungsaufträge zu Beginn des Schedulings bekannt.
- Dem Schedulingalgorithmus stehen gute Schätzungen der Geschwindigkeit von Netzwerkverbindungen, Speichersystemen und Rechnern sowie zur Ausführungsdauer der Programme zur Verfügung.
- Der Schedulingalgorithmus minimiert eine einfache Zielfunktion, etwa die mittlere Ausführungsdauer oder die Gesamtkosten (€ oder „virtuelle“ Kosten).
- Nur die Netzwerkverbindungen zwischen den einzelnen Organisationen sind für das Scheduling relevant.

Diese Annahmen treffen jedoch, wie nachfolgend erläutert, in realen Umgebungen nicht immer zu.

Dem Schedulingalgorithmus sind alle Verarbeitungsaufträge zu Beginn des Schedulings bekannt

Bei den bestehenden Grid-Schedulern handelt es sich meist um *Offline-Scheduler*. Beim Offline-Scheduling sind dem Schedulingalgorithmus alle Verarbeitungsaufträge zu Beginn des Schedulings bekannt. Die Aufgabe des Schedulers besteht darin, *alle* Aufträge A in einem Schritt so auf die zur Verfügung stehenden Ressourcen zuzuordnen und die Ausführungsreihenfolge der Aufträge festzulegen (einen Ausführungsplan $Z \in \mathbf{Z}$ zu erzeugen), dass eine vorgegebene Zielfunktion $\varphi(Z)$ minimiert wird [Pin12].

Häufig wird die Annahme, alle Aufträge seien bekannt, dahin gehend relativiert, dass nur alle Aufträge innerhalb eines gewissen Zeitraums bekannt sein müssen. Der Scheduler wird hierbei in jedem Intervall $i \in I$ (t_i bezeichnet die Länge der Intervalle i) einmal ausgeführt und minimiert die Zielfunktion $\varphi(Z_i)$ nur für die Aufträge, die innerhalb dieses Intervalls eingetroffen sind. Begründet wird dieses Modell damit, dass die Anzahl der wartenden Aufträge viel größer sei, als die Anzahl der zur Verfügung stehenden Ressourcen und darüber hinaus stetig neue Aufträge eintreffen. Somit sind dem Scheduler zwar nicht alle, aber *alle aktuell relevanten* Aufträge bekannt, die in einem Intervall auf die Ressourcen verteilt werden sollen, so dass die erzeugte Zuordnung für die Aufträge des Intervalls annähernd identisch ist mit derjenigen, die alle Aufträge betrachtet.

Da das Problem, eine optimale Zuordnung zu finden im Allgemeinen sehr schwierig ist, werden, wie die zuvor vorgestellten Verfahren zeigen, häufig heuristische Ansätze oder metaheuristische Optimierungsverfahren für das Offline-Scheduling eingesetzt [AD06]. Bei den heuristischen Offline-Schedulern werden nicht nur die Ressourcenzuordnungen heuristisch gewählt, sondern auch die Ausführungsreihenfolge der Aufträge wird durch eine Heuristik bestimmt. Dies soll am Beispiel der drei folgenden Heuristiken [Mah99] gezeigt werden, die versuchen, die maximale Ausführungsdauer zu minimieren:

- *Min-Min.* Der Auftrag mit der kürzesten Ausführungsdauer wird der Ressourcen zugeteilt, die den Auftrag am schnellsten abarbeitet.
- *Max-Min.* Der Auftrag mit der längsten Ausführungsdauer wird der Ressourcen zugeteilt, die den Auftrag am schnellsten abarbeitet.
- *Sufferage.* Der Auftrag, der am meisten *leiden* müsste, wenn er nicht die für ihn beste Ressourcenzuordnung erhalten würde, wird bevorzugt. Der Leidenswert wird dabei aus der Differenz der Ausführungsdauer zwischen der besten und der zweitbesten Ressourcenzuordnung gebildet.

Die Heuristiken versuchen darüber hinaus, die Umgebungseigenschaften zur weiteren Vereinfachung des Problems zu nutzen.

Offline-Scheduler eignen sich insbesondere für Anwendungsfelder, bei denen blockweise eine große Anzahl von Aufträgen auf einmal eintreffen und auf eine kleine Anzahl statischer Ressourcen verteilt werden sollen. Treffen jedoch die Aufträge einzeln nacheinander über die Zeit ein und sollen vielen, dynamischen Ressourcen zugeordnet werden, wie dies in vielen Anwendungsfeldern der Fall ist, sind Offline-Scheduler nicht immer geeignet. So ist bei Zielfunktionen, welche die Ausführungsdauer berücksichtigen, das Verhältnis zwischen der Länge des Schedulingintervalls t_i und der mittleren Ausführungsdauer der Aufträge t_a von entscheidender Bedeutung. Treffen über die Zeit viele Aufträge ein, deren Ausführungsdauer deutlich kürzer als das Schedulingintervall ist ($t_a \ll t_i$), und sind Ressourcen frei, so kann die Wartezeit bis zur nächsten Schedulingphase größer sein als die eigentliche Ausführungsdauer der Aufträge. Um diesem Problem entgegen zu wirken, kann das Schedulingintervall verkürzt werden. Bei kurzen Intervallen stehen dem Offline-Scheduler jedoch nur sehr wenige Aufträge - im Extremfall nur ein Auftrag - zur Verfügung. Dies steht offensichtlich im Widerspruch zum eigentlichen Grundgedanken des Offline-Scheduling, bei dem durch geeignete Ressourcenwahl und Ausführungsreihenfolge für *alle* Aufträge gemeinsam die Zielfunktion minimiert werden soll. Da insbesondere die Heuristiken auf die effiziente Ressourcenwahl und die Bestimmung der Ausführungsreihenfolge für viele Aufträge abgestimmt sind, gelten die den Heuristiken zugrunde liegenden Annahmen nur bedingt für kurze Intervalle mit wenigen Aufträgen.
Ein weiteres Problem des Offline-Scheduling ergibt sich aus der Dynamik der Umgebung, so dass eine vom Offline-Scheduler festgelegte Zuordnung eines Auftrags zu einer Ressource hinfällig werden kann, da die Ressource zum avisierten Startzeitpunkt des Auftrages nicht mehr zur Verfügung steht.

In dynamischen Anwendungsgebieten mit stark unterschiedlichen Aufträgen und nicht vorhersehbaren Auftragsaufkommen ist daher das Offline-Schedulingmodell nur bedingt geeignet. Für diese Umgebungen ist ein *Online-Scheduling* [Gra69] [Sga97], bei dem vom Scheduler jeder Auftrag einzeln den Ressourcen zugewiesen wird, besser geeignet. Die einzelnen Aufträge werden hierbei vom Online-Scheduler so auf die Ressourcen verteilt, dass die vorgegebene Zielfunktion nicht für den einzelnen Auftrag, sondern über viele Aufträge minimiert wird. Im Gegensatz zum Offline-Scheduler setzen Online-Scheduler im Allgemeinen keinerlei Informationen über die Auftragsdauer oder das Auftragsaufkommen voraus und eignen sich damit insbesondere für dynamische, organisationsübergreifende Umgebungen.

Dem Schedulingalgorithmus stehen genaue Informationen zur Verfügung

Offline-Scheduler minimieren die Zielfunktion, indem sie sowohl die Aufträge den vorhandenen Ressourcen zuweisen als auch die Ausführungsreihenfolge der Aufträge festlegen. Um eine möglichst optimale Zuordnung und Reihenfolge der Aufträge auf den Ressourcen bestimmen zu können, muss die Ausführungsdauer der Aufträge bekannt sein. Bei ressourcenintensiven Anwendungen setzt sich die Ausführungsdauer eines Auftrages aus der Startverzögerung, der Transferdauer sowie der Berechnungs-

dauer zusammen. Die Startverzögerung eines Auftrags ergibt sich aus der Differenz des Eintreffzeitpunkts des Auftrages und dem Ausführungsstart. Die Transferdauer beschreibt die Zeit, welche benötigt wird, alle Eingabedaten des Auftrages zur Ausführungsressource zu übertragen. Die Zeit, die zur Verarbeitung der Eingabedaten benötigt wird, entspricht der Berechnungsdauer.

Zur genauen Bestimmung von Startverzögerung, Transfer- und Berechnungsdauer müssen sowohl exakte Umgebungs- als auch Auftragsinformationen vorliegen. Für die Transferdauer müssen unter anderem die aktuell verfügbaren Geschwindigkeiten der beim Transfer beteiligten Speicherressourcen sowie aller Netzwerkverbindungen bekannt sein. Die Ausführungsdauer lässt sich aus der Prozessorgeschwindigkeit der Rechenressource sowie der Anzahl der Rechenoperation die das Programm zur Verarbeitung der Eingabedaten benötigt bestimmen. Die Startverzögerung ergibt sich wiederum aus der Summe der Ausführungsdauern aller Aufträge, die vor diesem Auftrag auf der zugeordneten Ressource ausgeführt werden.

Diese Informationen stehen in realen Umgebungen selten exakt zur Verfügung, so dass diese geschätzt werden müssen [WSH99]. Für ressourcenintensive Anwendungen werden insbesondere gute Schätzverfahren für die Speicher- und der Netzwerkgeschwindigkeiten sowie für die Anzahl der Rechenoperation eines Auftrages benötigt.
In organisationsübergreifenden Umgebungen mit vielen unterschiedlichen Benutzergruppen und Anwendungsfeldern sind verlässliche Schätzungen dieser Informationen kaum möglich. So erschwert die große Anzahl an Anwendungen und Anwendungsgebieten die genaue Abschätzung der benötigten Rechenoperation für einen Auftrag. Schätzungen der Benutzer sind hierbei ebenso unzuverlässig wie heuristische Verfahren, die aus vorherigen Anwendungsausführungen die Ausführungsdauer bestimmen. Dies gilt vor allem für Anwendungen bei denen sich kleine Änderungen an der Konfiguration oder den Eingabedaten sehr stark auf die Ausführungsdauer auswirken, wie etwa bei Simulationsberechnungen.
Auch die Umgebungsinformationen, insbesondere die Netzwerkinformationen, können in organisationsübergreifenden Umgebungen nicht exakt bestimmt werden. Mit der Komplexität des Netzwerks innerhalb (Intranet) und zwischen den Organisationen (Internet) steigt auch die Komplexität der Bestimmung der Netzwerkgeschwindigkeiten. Bei der Betrachtung der Netzwerkgeschwindigkeit muss beachtet werden, dass das Netzwerk nicht nur für die Verarbeitung der Berechnungsaufträge, sondern für alle Anwendungen und Dienste der Organisationen verwendet wird. Die aktuelle Auslastung des Netzwerks hängt somit von vielen unterschiedlichen Faktoren mehrerer unabhängiger Organisationen ab.

Der Schedulingalgorithmus minimiert eine einfache Zielfunktion

Die bestehenden heuristischen Offline-Scheduler nutzen die Eigenschaften einfacher Zielfunktionen, um nicht alle möglichen Zuordnungen der Aufträge auf die Ressourcen zur Approximation des optimalen Offline-Scheduling betrachten zu müssen. Die am Häufigsten in der Literatur anzutreffende Zielfunktion ist die mittlere oder maxi-

male Ausführungsdauer ($\sum_{z \in Z} \tau_z$, $\max_{z \in Z} \tau_z$)[9] oder die mittleren Kosten ($\sum_{z \in Z} \kappa_z$) über alle Aufträge [AD06]. Diese Zielfunktionen erlauben es den Heuristiken, unter Hinzunahme zusätzlicher Annahmen, den Suchraum anhand der Dauer *oder* der Kosten einzugrenzen. So werden häufig zu teure oder zu langsame Ressourcen einfach vernachlässigt oder nur die nach einem vorgegebenen Kriterium bestimmten, besten n Ressourcen betrachtet.

Ein entscheidender Nachteil dieser einfachen Zielfunktionen besteht darin, dass sie nur eine Dimension eines mehrdimensionalen Problems erfassen [TKB09]. Häufig soll eine Zielfunktion sowohl die Anforderungen der Benutzer als auch die der Betreiber abdecken: Aus Sicht der Benutzer ist eine möglichst schnelle Ausführung der Aufträge erwünscht, wohingegen die Organisationen an einer möglichst kostengünstigen Ausführung interessiert sind. Diese scheinbar gegensätzlichen Anforderungen lassen sich mit den einfachen Zielfunktionen kaum abbilden, denn ein Scheduler, welcher nur die Kosten minimiert, wird ein Ausführungsplan erzeugen, der günstige und im Allgemeinen langsame Ressourcen bevorzugt. Auf der anderen Seite führt ein Scheduling nur auf Basis der Ausführungsdauer zur Verwendung der schnellsten und damit teuersten Ressourcen.

Es wird also eine Zielfunktion benötigt, die beide Anforderungen, Ausführungsdauer und Kosten, gemeinsam abbildet und damit zu einer *effizienten* Ausführung der Aufträge führt. Die von den bestehenden Heuristiken verwendeten Auswahlkriterien zur Einschränkung des Suchraums sind für komplexere Zielfunktionen, die sowohl die Kosten als auch die Ausführungsdauer beinhalten, jedoch nicht mehr geeignet.

Nur die Netzwerkverbindungen zwischen den einzelnen Organisationen sind für das Scheduling relevant

Die bestehenden Verfahren basieren auf einem Umgebungsmodell, welches alle Rechen- und Speicherressourcen einer Organisation, wie in Abbildung 2.2 dargestellt, zu einer virtuellen Ressource bündelt. Folglich umfasst das Umgebungsmodell nur die Netzwerkverbindungen, die diese virtuellen Ressourcen der unterschiedlichen Organisationen miteinander verbinden. Beim Scheduling einer Anwendung betrachten die bestehenden Schedulingalgorithmen somit nur die virtuellen Rechen- und Speicherressourcen sowie die Netzwerkverbindungen zwischen den Organisationen.

Das von den bestehenden Verfahren genutzte, *flache* Umgebungsmodell impliziert, dass alle Netzwerkverbindungen innerhalb einer Organisation unendliche Bandbreite, keine Latenz und keine Kosten besitzen. Diese Annahme ist bei genauer Betrachtung aktueller realer Umgebungen nicht aufrechtzuerhalten, da sich die Ressourcen auf mehreren Kommunikationsebenen mit unterschiedlichen Bandbreiten und Kosten befinden können.

Eine Organisation - ein Konzern oder eine Universität - besteht häufig aus meh-

[9] Da es sich um ein Optimierungsproblem handelt, kann der konstante Term $|Z|$ bei der mittleren Ausführungsdauer vernachlässigt werden

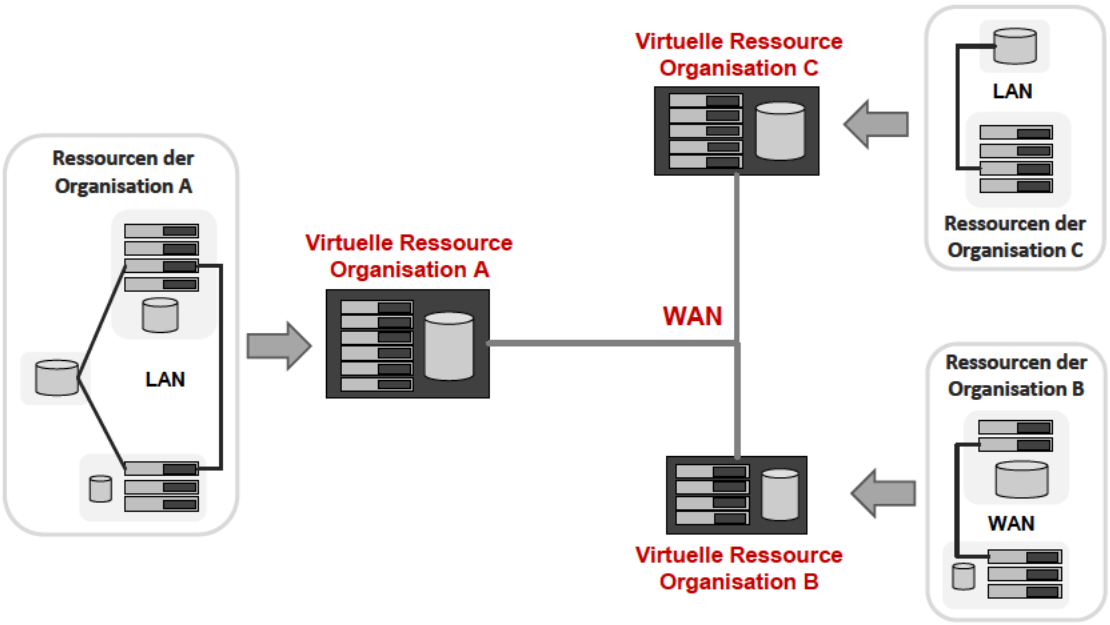


Abbildung 2.2: Das Umgebungsmodell aktueller Grid-Scheduler.

reren Standorten, die über LAN-[10] oder WAN-Verbindung[11] miteinander verbunden sind. Jeder dieser Standorte kann zur Ausführung ressourcenintensiver Anwendungen über eigene Ressourcen verfügen. Hierbei handelt es sich meist um Berechnungscluster die mehrere Rechen- sowie Speicherressourcen über ein lokales Netzwerk mit hoher Bandbreite und geringer Latenz zusammenfassen. Da eine direkte Verbindung aller Clusterressourcen mit allen anderen Ressourcen des Standortes mit enorm hohen Kosten verbunden wäre, erfolgt die Anbindung eines Clusters an die übrige Netzwerkinfrastruktur über Cluster-Gateways - ausgewählte Ressourcen des Clusters oder Netzwerkswitche. Die Kommunikation mit Ressourcen außerhalb des Clusters erfolgt somit nicht direkt und alle Ressourcen eines Clusters teilen sich die zur Verfügung stehende Geschwindigkeit der Gateways für die Kommunikation mit Ressourcen anderer Cluster oder Organisationen.[12] Des Weiteren muss beachtet werden, dass die Netzwerkverbindungen, welche die Cluster-Gateways eines Standortes oder einer Organisation miteinander verbinden, auch für unterschiedliche andere, eventuell unternehmenskritische, Dienste und Applikationen der Organisation verwendet werden können. Bei Verbindung zwischen Organisationen handelt es sich meist um WAN-Leitung mit einer deutlich geringeren Bandbreite und höher Latenz verglichen mit den Cluster- oder Interclusterverbindungen. Diese Verbindungen werden von allen Anwendungen, Diensten und Personen für die externe Kommunikation genutzt, so dass diese eine hohe Auslastung aufweisen. Die Kommunikation zwischen den Ressourcen der Organisationen kann einerseits über spezielle Organisations-Gateways

[10] Lokales Netzwerk (LAN). Häufig mit einer Bandbreite von 1Gb/s bis 10 Gb/s
[11] Weitverkehrsnetzwerk (WAN). Häufig mit einer Bandbreite von deutlich weniger als 1Gb/s
[12] Etwa das Leibniz-Rechenzentrum (http://www.lrz.de/services/compute/supermuc/systemdescription)

oder andererseits direkt über die Cluster-Gateways erfolgen. Abbildung 2.3 verdeutlicht die hierarchische Netzwerkstruktur noch einmal. Die Kosten der verschiedenen

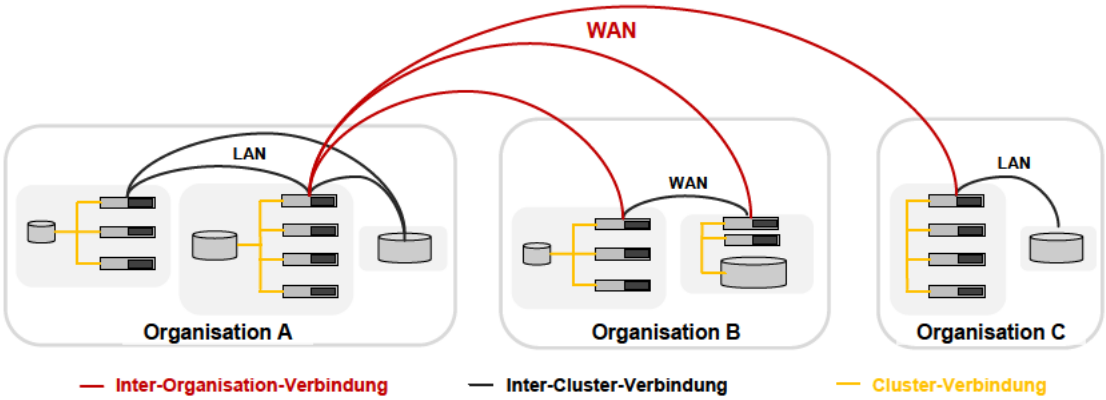


Abbildung 2.3: Die hierarchische Netzwerkstruktur von organisationsübergreifenden Umgebungen.

Ressourcen sind ein weiterer wichtiger Aspekt bei der Betrachtung der Umgebung. Neben den Kosten für Rechen- und Speicherressourcen sind insbesondere die Kosten der Netzwerkverbindungen der unterschiedlichen Ebenen von entscheidender Bedeutung. Kosten können hierbei einerseits monetärer Art, etwa die Mietkosten einer Verbindung oder die Kosten pro übertragene Dateneinheit, oder andererseits virtuelle Kosten, etwa interne Verrechnungskosten, zur Vermeidung von Überbelastungen sein.
Innerhalb eines Clusters können die Verbindungen als kostenlos betrachtet werden, da diese Verbindungen ausschließlich für die vom Scheduler auszuführenden Anwendungen genutzt werden. Neben dem Datentransfer wird das Clusternetzwerk jedoch auch für die Interprozesskommunikation verteilter Anwendungen genutzt. Um zu vermeiden, dass datenintensive Anwendungen das Netzwerk mit Datentransfers unnötig belasten und so die Interprozesskommunikation anderer Anwendungen beeinflussen, können auch den Clusterverbindungen geringe Kosten zugeordnet werden.
Bei Verbindungen zwischen Clustern innerhalb einer Organisation müssen zwei Fälle unterschieden werden. Handelt es sich um eine Verbindung innerhalb eines Standortes, fallen wie bei den Clustertransfers ebenfalls meist nur virtuelle Kosten an. Diese dienen primär dazu, dass diese Verbindungen nicht mit Datentransfers zwischen den Clustern überlastet werden und dadurch wichtige Anwendungen und Dienste der Organisation, die ebenfalls diese Verbindungen verwenden, beeinträchtigt werden. Verglichen mit den Kosten von Verbindungen innerhalb eines Clusters, sind die Kosten von Interclusterverbindungen innerhalb eines Standortes deutlich höher.
Verbindet eine Interclusterverbindungen Cluster unterschiedlicher Standorte, handelt es sich, wie bei Verbindungen zwischen Organisationen, meist um angemietete WAN-Verbindungen. Diese Verbindungen verursachen zumeist hohe Kosten in Form von Miet- und/oder Transferkosten. Darüber hinaus werden diese von allen Anwen-

dungen, Diensten sowie Personen für die externe, zum Teil geschäftskritische Kommunikation genutzt. Zur Vermeidung einer Überlastung dieser sensiblen Ressource können daher reale und virtuelle Kosten für die Verwendung der Interorganisationsverbindungen anfallen, so dass diese Verbindungsart als die teuerste Ressource angesehen werden kann.

Das von den bestehenden Verfahren verwendete flache Umgebungsmodell passt nicht zu diesen Gegebenheiten, so dass für eine effiziente Ausführung, insbesondere von datenintensiven Anwendungen, die hierarchische Struktur der realen Umgebungen beachtet werden muss.

2.2 Datenintensive Verarbeitungssysteme

Die Verwaltung und Verarbeitung großer Datenmengen gewinnt mit der zunehmenden digitalen Datenaufzeichnung - Bilder, Dokumente, Sensorinformationen, ... - immer mehr an Bedeutung. Obwohl die Leistungsfähigkeit einzelner Komponenten immer weiter zunimmt, sind diese bei Weitem nicht in der Lage Datenmengen im Tera- und Petabyte-Bereich zu speichern und zu verarbeiten. Daher wurden in den letzten Jahren eine Reihe unterschiedlicher Konzepte zur verteilten Datenhaltung und Verarbeitung entwickelt, mit der sich mehrere Ressourcen gemeinsam nutzen lassen. In diesem Abschnitt werden verschiedene verteilte Verarbeitungstechnologien vorgestellt und auf ihre Eignung für daten- und rechenintensive Anwendungen in globalen, heterogenen Umgebungen untersucht. Auch wenn die vorgestellten Verfahren und Technologien nicht direkt für die in dieser Arbeit untersuchte Problemstellung verwendet werden können, helfen die gewonnenen Erkenntnisse bei der Entwicklung eines geeigneten Schedulingalgorithmus.
Zunächst werden die Entwicklungen im Bereich der Relationalen Datenbanksysteme erläutert. Anschließend werden aktuelle, auf schnellen Datenzugriff und Skalierbarkeit optimierte, verteilte Verfahren und Systeme vorgestellt.

2.2.1 Relationale Datenbanksysteme

Relationale Datenbanksysteme bilden in vielen Bereichen die Basis von Datenverarbeitungssystemen. Für die Speicherung und Auswertung großer, verteilter Datenmengen wurden diese zu Verteilten Datenbanksystemen weiterentwickelt [OV11] [Dad96]. Verteilte Datenbanksysteme ermöglichen die Integration verteilter Daten sowie die Partitionierung der Daten auf mehrere Komponenten, wodurch sie in der Lage sind große Datenmengen zu speichern und zu verarbeiten. Durch den Einsatz von Transaktionen kann darüber hinaus die Konsistenz der Daten bei Veränderungen garantiert werden [Dad96].

Seit mehr als 30 Jahren befasst sich eine Vielzahl von Arbeiten mit den unterschiedlichen Herausforderungen von Verteilten Datenbanksystemen [Uma88]. Ein zentraler, und insbesondere für diese Arbeit relevanter Forschungsbereich befasst sich mit Verfahren zur Datenverteilung und der Anfrageoptimierung. Die Verteilung

der Daten auf die Komponenten dient unter anderem der Lastverteilung und der Verbesserung der Verfügbarkeit von Daten. Eine Verteilung der Daten zu finden, die bezüglich vorgegebener Anwendungen kostenoptimal ist, ist jedoch NP-Vollständig [LY80].
Die Aufgabe der Anfrageoptimierung besteht darin, auf Grundlage einer gegebenen Datenverteilung die einzelnen Datenzugriffe so zu planen, dass die Kosten minimal werden. Hierbei werden insbesondere die Prozessorkosten, die Kommunikationskosten sowie die Ein- und Ausgabekosten betrachtet [JUMS83]. Die Bestimmung eines kostenoptimalen Ausführungsplanes ist ebenfalls NP-Vollständig [HY79][IK84] Es existiert eine große Anzahl an unterschiedlichen Verfahren für die Anfrageoptimierung in Verteilten Datenbanksystemen [Kos00] [AAO05]. Die Minimierung der Netzwerkkosten, insbesondere die Vermeidung von WAN-Transfers, ist hierbei ein zentrales Thema [CI86] [RM97] [Gra87].

Abgrenzung

Das Problem der Anfrageoptimierung in Verteilten Datenbanksystemen entspricht in Teilen dem der im vorherigen Abschnitt betrachteten Grid-Scheduler. Beiden gemeinsam ist die Herausforderung, die Ausführung so zu planen, dass die Berechnung, der Datenzugriff sowie der Datentransfer möglichst effizient durchgeführt werden. Im Gegensatz zu den Grid-Schedulern, die beliebige Anwendungen zur Ausführung bringen müssen, kann sich die Anfrageoptimierung in Verteilten Datenbanksystemen auf die relationale Abfrageverarbeitung fokussieren. Mithilfe statistischer Informationen über die Daten lassen sich dadurch der Berechnungsaufwand sowie die Größe der Ein- und Ausgabedaten gut vorhersagen. Die Abschätzung der Netzwerktransfers, insbesondere in komplexen Netzwerken, bleibt aber ein Problem beider Gebiete.

Obwohl Verteilte Datenbanksysteme auf die Datenallokation und Ausführungsplanung von Relationen spezialisiert sind und dadurch nicht für beliebige ressourcenintensive Anwendungen geeignet sind, helfen die entwickelten Ansätze bei der Entwicklung neuer Verfahren. Hierbei sind vor allem die Erkenntnisse im Bereich der WAN-Transfers hervorzuheben, wo in mehreren Veröffentlichungen eine Vermeidung dieser Transferart vorgeschlagen wird [CI86] [Gra87].

2.2.2 Skalierbare Datenverarbeitungssysteme

Der enorme Zuwachs an Daten in unterschiedlichen Anwendungsfeldern hat zur Entwicklung neuer Methoden geführt, die auf die skalierbare Verarbeitung von Daten optimiert sind. Das Ziel sind Verfahren und Systeme, deren Datenverarbeitungsleistung linear mit der Anzahl von Ressourcen skaliert. Aktuelle skalierbare Datenverarbeitungssysteme basieren auf den Ansätzen von Berechnungsclustern und adaptieren diese auf die Analyse großer Datenmengen[DG04]. Die Architektur von Berechnungsclustern teilt, wie bereits erläutert, die Speicher- und Berechnungsaufgaben, so dass spezialisierte Cluster zur Datenspeicherung und zur Berechnung eingesetzt werden. Die Haupteinsatzgebiete dieser Cluster sind Simulation und Berechnungen komple-

xer Modelle. Obwohl hierbei sehr große Datenmengen anfallen können, sind diese Systeme auf eine hohe Rechenleistung optimiert, so dass die Berechnungscluster weit mehr Komponenten als die Datencluster umfassen. Bei datenintensiven Filter- und Analyseprogrammen führt dieses Ungleichgewicht zu einer schlechten Auslastung der Berechnungscluster, da aufgrund der Unterscheidung zwischen Speicher- und Rechenressourcen die Daten immer transferiert werden müssen.

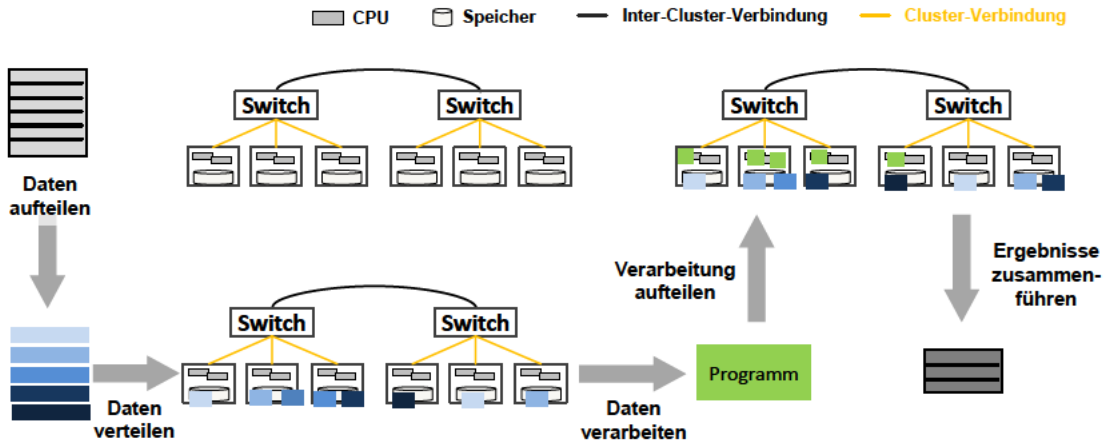


Abbildung 2.4: Allgemeine Architektur von skalierbaren Datenverarbeitungssystemen.

Für Szenarien, in denen der Fokus auf der Verarbeitung großer Datenmengen liegt, wurden daher die Daten- und Berechnungscluster kombiniert. Der Aufbau von Verarbeitungssystemen für diese kombinierten Cluster folgt meist der in Abbildung 2.4 dargestellten Architektur. Die über ein Standardnetzwerk miteinander verbundenen Ressourcen speichern jeweils einen Teil der zu verarbeitenden Daten lokal. Die Verwaltung der Daten übernimmt ein verteiltes Dateisystem, das insbesondere den Speicherort aller Datensätze verwaltet. Die Ausführung von Analyseprogrammen wird von der Ausführungsverwaltung gesteuert. Diese nutzt die Positionsinformationen des verteilten Dateisystems, um das auszuführende Programm auf den Ressourcen auszuführen, die die zu verarbeitenden Datensätze vorhalten. Die Applikationen sollten in der Lage sein, einzelne, disjunkte Datensätze der zu verarbeitenden Daten unabhängig voneinander verarbeiten zu können, damit möglichst viele Ressourcen parallel eingesetzt werden können.

Das bekannteste System dieser Art wird von Google betrieben und besteht aus mehreren Tausend Standard PCs, die sowohl zur Datenspeicherung als auch zur Verarbeitung verwendet werden. Die Verwaltung dieses Clusters übernimmt das *MapReduce*-Framework[DG04], von dem es mit *Hadoop*[13] auch eine freie Implementierung gibt. Die Basis des MapReduce-Frameworks bildet ein verteiltes Dateisystem, das Google File System (GFS) beziehungsweise das Hadoop Distributed File System (HDFS). Das Dateisystem besteht aus einem zentralen Verwaltungsknoten und beliebig vielen Datenknoten zur Speicherung der Daten. Anstelle von Dateien speichern

[13] http://hadoop.apache.org/

die Datenknoten einzelne Datenblöcke, so dass eine Datei über mehrere Datenknoten verteilt sein kann. Die Zuordnung von Datenblöcken zu Dateien ist die zentrale Aufgabe des Verwaltungsknotens. Darüber hinaus ist der Verwaltungsknoten für die Ausführung von Operationen auf dem Namensraum (öffnen, schließen, umbenennen von Dateien und Verzeichnissen) und die Zugriffskontrolle zuständig [Bor07].

Die Ausführungsverwaltung des Frameworks, bestehend aus einem Master- und mehreren Worker-Prozessen, nutzt das verteilte Dateisystem, um MapReduce-Aufträge möglichst in der Nähe der zu verarbeitenden Daten auszuführen[Läm08]. Ein Auftrag besteht hierbei aus einer `Map` und einer `Reduce` Funktion sowie den Datenblöcken der zu verarbeitenden Daten. Der von Google entwickelte MapReduce-

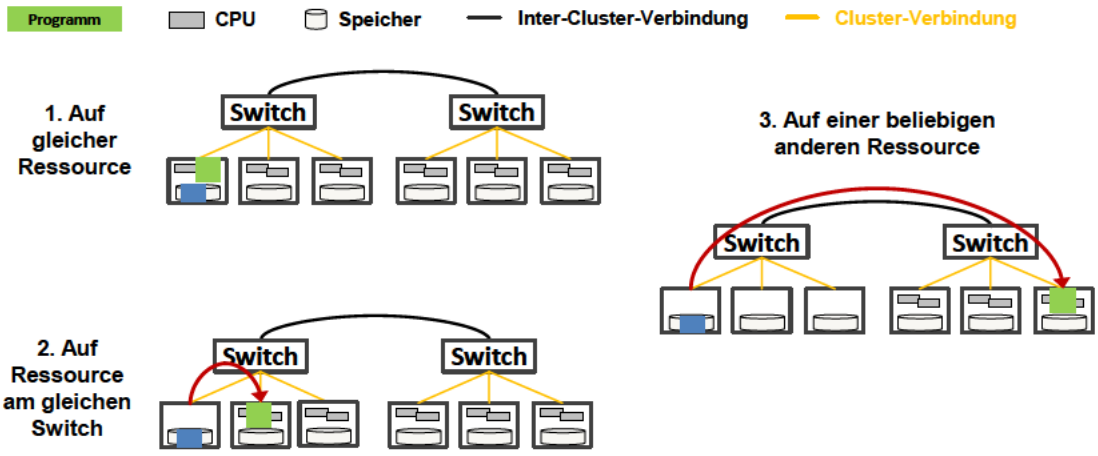


Abbildung 2.5: Das hierarchische Scheduling der MapReduce-Frameworks.

Scheduler [DG04] versucht, die Map-Funktionen möglichst *nahe* bei den zu verarbeitenden Datenblöcken auszuführen. Diesem Ansatz liegt die Annahme zugrunde, dass das Netzwerk, nach den von Google gemachten Erfahrungen, eine knappe Ressource ist, die geschont werden sollte. Des Weiteren wird angenommen, dass der Transferaufwand mit der Distanz zwischen Quelle und Ziel zunimmt. Es wird daher unterschieden, ob Daten lokal verarbeitet und damit nicht transferiert werden müssen, zwischen zwei Komponenten am gleichen Netzwerkswitch oder über Switche hinweg transferiert werden. Betrachtet man aktuelle Netzwerkinfrastrukturen, ist diese Vorgehensweise auch einfach nachvollziehbar. So sind moderne Switche häufig nicht symmetrisch ausgelegt, so dass deutlich mehr Anschlüsse und Bandbreite für Komponenten als für die Verbindung zu anderen Switchen zu Verfügung stehen. Damit steht einem Transfer zwischen zwei Komponenten am gleichen Switch die volle Bandbreite zur Verfügung, wohingegen sich bei Transfers über Switchgrenzen hinweg alle an den Switchen angeschlossen Komponenten die verfügbare Bandbreite teilen.

Auf Basis dieser Annahmen ermittelt der Scheduler für eine MapReduce-Berechnung zunächst für jeden Datenblock vom verteilten Dateisystem (GFS,

HDFS) die Position aller Repliken. Anschließend versucht der Scheduler eine Zuordnung zu generieren, bei der die Map-Funktionen auf den Komponenten ausgeführt werden, welche die zu verarbeitenden Datenblöcke vorhalten. Ist dies für einen Datenblock nicht möglich, versucht der Scheduler die Ausführung auf einer Komponente am gleichen Netzwerkswitch zu starten. Ist auch dies nicht möglich, wird eine beliebige andere Komponente zur Ausführung gewählt. In den beiden letzten Fällen werden die Datenblöcke zur Ausführungskomponente transferiert. Dieses hierarchische Scheduling wird in Abbildung 2.5 noch einmal verdeutlicht. Nachdem alle Map-Funktionen erfolgreich ausgeführt wurden, veranlasst der Master einen oder mehrere Worker die Reduce-Funktion auf die Zwischenergebnisse anzuwenden.

Im Folgenden soll das MapReduce-Verarbeitungsprinzip kurz vorgestellt werden, um die Vor- und Nachteile von MapReduce-Systemen aufzeigen zu können. Der Name MapReduce wurde in Anlehnung an die Map/Reduce Funktionen funktionaler Programmiersprachen gewählt und beschreibt die beiden zentralen Funktionen `Map` und `Reduce`. Die Map-Funktion nimmt ein Schlüssel/Wert-Paar entgegen und erzeugt daraus eine Menge neuer Schlüssel/Wert-Paare (Beispiel aus [DG04]):

$$Map: \ (k, v) \mapsto [(l_1, w_1), (l_2, w_2), \ ..., (l_n, w_n)]$$

In einem Zwischenschritt werden alle Paare mit identischem Schlüssel l_j gruppiert: $(l_j, [w_x, w_y, ...])$. Die Reduce-Funktion nimmt einen Schlüssel mit allen zugeordneten Werten entgegen und erzeugt daraus eine Liste von Ergebniswerten:

$$Reduce: \ (l_j, [w_x, w_y, ...]) \mapsto (l_j, [r_x, r_y, ...])$$

Aus diesen Definitionen ist leicht zu erkennen, dass die Anwendung der Map- und Reduce-Funktionen auf einen einzelnen Datensatz das gleiche Ergebnis liefert, wie die mehrfache Anwendung auf einen geteilten Datensatz. Mit dem MapReduce-Verfahren können so sehr große Datenmengen in kleinere Einheiten aufgeteilt und verarbeitet werden.

Die Funktionsweise des MapReduce-Verfahrens lässt sich Anhand des in [DG04] beschrieben Beispiels, der Bestimmung der Worthäufigkeit von Texten, anschaulich erläutern. In Auflistung 2.1 sind die Map- und Reduce-Funktion in Pseudocode für diese Aufgabe dargestellt.

```
map(String key, String value):
// key: document name; value: document contents
for each word w in value:
  EmitIntermediate(w, "1");

reduce(String key, Iterator values):
// key: a word; values: a list of counts
int result = 0;
for each v in values:
  result += ParseInt(v);
Emit(key, result);
```

Listing 2.1: Beispiel für Map- und Reduce-Funktionen(nach [DG04])

Die Map-Funktion gibt für jedes gefundene Wort (`key`) die Häufigkeit 1 (`value`) aus. Im Zwischenschritt wird für jedes Auftreten eines Wortes eine Liste aus Einsen erzeugt ($(wort, [1, 1, ...])$). Die Reduce-Funktion ermittelt die Worthäufigkeit durch einfaches Aufsummieren der Einsen.

Abgrenzung

Das MapReduce-Verfahren eignet sich trotz der vermeintlichen Einschränkung auf die Map- und Reduce-Funktionen für eine Vielzahl an Problemstellungen im Umfeld datenintensiver Anwendungen. Wie aus dem zuvor beschriebenen Beispiel hervorgeht, können Anwendungen im MapReduce-Verfahren umgesetzt werden, die in *einem* Schritt alle Teilmengen des Datensatzes *unabhängig, parallel* verarbeiten und in einem weiteren Schritt verdichten. Allerdings zeigt dieses Beispiel auch die Nachteile des MapReduce-Verfahrens: Rechenintensive Anwendungen, insbesondere Anwendungen die Interprozesskommunikation erfordern, lassen sich kaum effizient durch Map- und Reduce-Funktionen abbilden. Weitere Einschränkungen ergeben sich aus den aktuellen Implementierungen wie Hadoop oder Disco[14]. So unterstützt die Ausführungsverwaltungen der Frameworks nur MapReduce-Berechnungen, so dass alle Berechnungen der Applikationen zwingend im Framework implementiert werden müssen. Die Integration bestehender Programme ist nicht oder nur teilweise möglich. Rechenintensive Anwendungen, etwa Simulationen, können entweder gar nicht oder aufgrund des nicht für diese Aufgaben geeigneten Schedulings nicht auf der optimalen Rechenressource ausgeführt werden.
Des Weiteren entfalten die Frameworks ihre volle Leistungsfähigkeit nur im Zusammenspiel mit den eigenen, proprietären verteilten Dateisystemen, welche die Daten selbstständig aufteilen. Datensätze, die nicht in diesem Dateisystem gespeichert sind, können daher nicht direkt verarbeitet werden. Für weltweit verteilte Daten, die von unterschiedlichen Organisationen verwaltet werden, eigenen sich die aktuellen Frameworks somit nicht, da sie eine enge Kopplung der Rechner voraussetzen und keine Mechanismen für die Kommunikation über Organisationsgrenzen bereitstellen. Ansätze wie [SL09] lösen einige dieser Probleme, insbesondere die organisationsübergreifende Kommunikation, indem sie das MapReduce-Verfahren auf Grids erweitern. Die anderen zuvor beschriebenen Einschränkungen für rechenintensive Anwendungen bleiben jedoch erhalten.

[14] http://discoproject.org/

Kapitel 3

Der DOHS-Algorithmus

Ressourcenintensive Anwendungen haben heutzutage sowohl im wissenschaftlichen wie auch im kommerziellen Umfeld eine weite Verbreitung. Die stetig steigenden Datenmengen haben dabei zur Entwicklung spezieller Verarbeitungssysteme geführt, die in der Lage sind, Datensätze im Petabyte-Bereich zu verarbeiten [TSA+10]. Ebenso sind hochparallele Verarbeitungssysteme für rechenintensive Anwendungen entstanden, mit denen sich immer komplexere und genauere Berechnungen durchführen lassen. Die Verarbeitungsleistung dieser Systeme ermöglichte die Entwicklung unterschiedlicher neuer Anwendungen, die mit traditionellen Verarbeitungsmethoden nicht realisiert werden konnten. Die über die Zeit gewonnenen Erfahrungen beim Einsatz dieser Systeme führt aktuell zu immer neuen Anwendungsgebieten bei denen sowohl daten- als auch rechenintensive Programme benötigt werden. Dies erfordert ein Verarbeitungssystem, welches den Anwendungen eine Kombination aus hoher Rechenleistung und schneller Datenzugriffe bereitstellt.

Wie in den vorangegangenen Kapiteln bereits erwähnt, nimmt die globale Zusammenarbeit unterschiedlicher Organisationen im wissenschaftlichen und kommerziellen Umfeld stetig zu. Neben der gemeinsamen Arbeit an Problemstellungen ist die effiziente Nutzung der vorhandenen Ressourcen eine der wichtigsten Triebfedern dieser Entwicklung. Eine kostengünstige sowie schnelle Ausführung der Anwendungen erfordert eine geeignete Zuordnung der einzelnen Verarbeitungsaufträge auf die global verteilten Ressourcen der einzelnen Organisationen. Das *Scheduling* ist daher die zentrale Aufgabe eines Verarbeitungssystems für daten- und rechenintensive Anwendungen in globalen, organisationsübergreifenden Umgebungen.

In den nachfolgenden Abschnitten wird der in dieser Arbeit entwickelte *Dual Objective Hierarchical Scheduling (DOHS)* Algorithmus zur effizienten Ausführung von rechen- und datenintensiven Anwendungen in organisationsübergreifenden Umgebungen [RGS10] [RGS11] [RGS12] detailliert vorgestellt. Zunächst wird das in dieser Arbeit verwendete Modell von organisationsübergreifenden Umgebungen erläutert. Anschließend wird die vom Scheduler zu minimierende Zielfunktion auf Basis dieser Umgebung eingeführt. Durch das sukzessive Entfernen aller schwer zu ermittelnden Informationen wird aus dieser Zielfunktion eine Schedulingfunktion abgeleitet. Diese bildet die Grundlage des entwickelten DOHS-Algorithmus, der abschließend detail-

liert erläutert wird.
In den nächsten Kapiteln wird der Schedulingalgorithmus anhand von Simulationen mit bestehenden Verfahren verglichen und ein Verarbeitungssystem auf Basis des DOHS-Algorithmus für organisationsübergreifende, ressourcenintensive Anwendungen vorgestellt.

3.1 Umgebungs- und Schedulingmodell

Die effiziente Ausführung komplexer Berechnungen sowie die Verarbeitung großer Datenmengen sind aktuell zentrale Themen im wissenschaftlichen wie kommerziellen Umfeld. In Umgebungen mit global verteilten Ressourcen, die mehreren eigenständigen Organisationen zugeordnet sein können, ergeben sich neue Herausforderungen und Rahmenbedingungen für das Scheduling. Im Folgenden wird daher zunächst die Umgebung detailliert vorgestellt und ein Modell entwickelt, welches diese möglichst exakt abbildet. Auf Basis dieses Modells sowie den unterschiedlichen Anforderungen der Betreiber und der Benutzer wird anschließend das Schedulingmodell erläutert und eine geeignete Schedulingfunktion definiert.

3.1.1 Das Umgebungsmodell

Das Umgebungsmodell umfasst einerseits die Ressourcen mit ihren Zuständen sowie andererseits die Anwendungen mit ihren Ausführungseigenschaften. Damit bildet das Umgebungsmodell die Grundlage des Schedulings. Je genauer das Modell hierbei die Realität abbildet, desto besser kann der Scheduler die Eigenschaften der Umgebung nutzen, um die Anwendungen effizient - im Sinne der gewählten Zielfunktion - auf die Ressourcen zu verteilen.

Wie in Abschnitt 1.1 bereits angesprochen, benötigen daten- und rechenintensive Anwendungen Prozessoren und Datenspeicher sowie eventuell Netzwerkverbindungen für ihre Ausführung. Daher wird im Folgenden nur auf diese drei Ressourcengruppen näher eingegangen.
Betrachtet man aktuelle Computer, verfügen diese über einen oder mehrere Prozessoren, die wiederum aus mehreren Prozessorkernen bestehen. Dadurch können mehrere Anwendungen parallel ausgeführt werden oder eine Anwendung kann mehrere Prozessorkerne nutzen. Für die Ausführung relevant sind darüber hinaus unter anderem der Prozessortyp, die Größe des Hauptspeichers sowie das verwendete Betriebssystem. Zusätzlich sind häufig selbst in Computern, die primär für Berechnungsaufgaben genutzt werden, eine oder zwei Festplatten verbaut, deren Kapazität sich aktuell zwischen 500GB und 3TB bewegt.
Für die Speicherung der Daten sind zurzeit eine Vielzahl unterschiedlicher Konzepte und Systeme im Einsatz, etwa Network Attached Storage oder Storage Area Networks. Im Clusterumfeld sind aktuell Speichersysteme Standard, die aus mehreren Computern bestehen, an die jeweils eine größere Anzahl von Festplatten an-

geschlossen sind. Ein Verteiltes Dateisystem, etwa IBM GPFS, FraunhoferFS[1] oder Lustre, integriert diese zu einem Speichersystem. Dieser Aufbau, aus mehreren einfachen Komponenten ein leistungsfähiges Speichersystem zusammenzusetzen, folgt dem Ansatz von Berechnungsclustern. Die so aufgebauten Speichersysteme sind in der Lage, Hunderte Petabyte zu speichern und bieten mehr als 100 GB/s an Durchsatz[2]. Obwohl die Hauptaufgabe der Computer in diesen Speichersystemen darin besteht, die angeschlossenen Festplatten zu verwalten, verfügen diese auch über eine nicht zu vernachlässigende Rechenleistung.

Für die weitere Betrachtung werden die Begriffe *Rechenressource* und *Speicherressource* als Abstraktion der zugrunde liegenden Computer und Speichersysteme verwendet. Eine Rechenressource umfasst hierbei die Prozessoren eines Computers sowie die weiteren berechnungsrelevanten Eigenschaften wie Prozessortyp, Hauptspeicher und Betriebssystem. Die beiden zentralen Eigenschaften einer Rechenressource sind die Prozessorgeschwindigkeit und die Kosten. Die Kosten pro Zeit der Rechenressource werden vom Betreiber vorgegeben und können sich für unterschiedliche Anwender oder Anwendungen unterscheiden. Zur Bestimmung der Prozessorgeschwindigkeit kann einerseits auf bestehende Benchmarks für den Prozessortyp zurückgegriffen werden oder ein Benchmark vor der ersten Verwendung durchgeführt werden.[3]
Eine Speicherressource umfasst dem entsprechend alle Speichereinheiten und Speichereigenschaften eines Speichersystems. So wird ein Speichersystem, das aus mehreren Komponenten bestehen kann, zu einer Speicherressource zusammengefasst. Auch für Speicherressourcen bilden die Geschwindigkeit[4] und die Kosten die beiden zentralen Eigenschaften. Als Messkriterium für die Geschwindigkeit wird die maximale Lese- und Schreibgeschwindigkeit in MB/s verwendet.

In globalen, organisationsübergreifenden Umgebungen, die keiner zentralen Kontrolle unterliegen, kann der Zustand einer Ressource nicht einfach von der Belegung des Schedulers abgeleitet werden, da die Ressourcen auch für andere Zwecke innerhalb einer Organisation verwendet werden können. Des Weiteren können jederzeit Ressourcen aus dem Gesamtsystem entfernt oder hinzugefügt werden. Die Zustandsbestimmung muss diese Dynamik berücksichtigen und geeignete Methoden und Messkriterien für die einzelnen Ressourcen bereitstellen. Für Rechen- und Speicherressourcen lässt sich der Zustand jedoch über die von den gängigen Betriebssystemen bereitgestellten Funktionen gut erfassen [BC05]. Bei Rechenressourcen kann der Zustand etwa über den Betriebssystemparameter `load` oder die Anzahl der Benutzerprozesse ermittelt werden. Zur Bestimmung der Auslastung von Speicherressourcen können ebenfalls die vom Betriebssystem angebotenen Funktionen, etwa `iostat`, verwendet werden.

Insbesondere in globalen Umgebungen spielt die dritte Ressourcengruppe, das

[1] http://www.fhgfs.com
[2] http://www.lrz.de/services/compute/supermuc/systemdescription
[3] http://www.spec.org
[4] Bei Speicher- und Netzwerkressourcen werden die Begriffe Bandbreite und Geschwindigkeit synonym verwendet

Netzwerk, eine zentrale Rolle. Aus Sicht des Schedulers ist die zentrale Aufgabe der Netzwerkressourcen die Datenübertragung zwischen den Rechen- und Speicherressourcen zu ermöglichen. Wie bereits in Abschnitt 1.1 erläutert, können nicht alle Rechen- und Speicherressourcen direkt miteinander Daten austauschen, so dass für Datentransfers mehrere Zwischenstationen erforderlich sein können.
Obwohl sich die Netzwerkumgebungen beim Zusammenschluss mehrerer Organisationen im Detail unterscheiden, weisen diese ähnliche hierarchische Strukturen auf. Dies liegt vor allem an der historischen Entwicklung der Rechnerlandschaften zur Ausführung ressourcenintensiver Anwendungen, die von Großrechnern über Rechencluster zum Zusammenschluss mehrerer Rechencluster verlief.
Rechnercluster sind heutzutage der Standard für ressourcenintensive Anwendungen und bestehen aus mehreren Rechen- und Speicherressourcen, die über ein lokales Netzwerk mit hoher Bandbreite und geringer Latenz, wie 10 GigEthernet oder Infiniband, miteinander verbunden sind. Dieser Aufbau ermöglicht einerseits die Interprozesskommunikation zwischen Rechenressourcen und andererseits eine hohe Bandbreite beim Zugriff auf die Speicherressourcen. Das Netzwerk innerhalb eines Clusters kann damit als unterste Ebene - die *Clusterebene* - des Netzwerks angesehen werden. Die Anbindung eines Clusters an die übrige Netzwerkinfrastruktur erfolgt über ausgewählte Ressourcen des Clusters oder Netzwerkswitche.
Innerhalb einer Organisation können mehrere Cluster, etwa an unterschiedlichen Standorten, vorhanden sein, die über diese Gateways mit LAN- oder WAN-Verbindung miteinander verbunden sind. Diese *Interclusterebene* verbindet die Cluster eines Standorts oder einer Organisation miteinander, kann aber auch für unterschiedliche andere Dienste und Applikationen der Organisation verwendet werden. Die Bandbreite dieser Verbindungen, auf kurzen Distanzen häufig 10 GigEthernet, bei WAN-Verbindungen noch deutlich niedriger, teilen sich alle Clusterressourcen für die Interclusterkommunikation mit den anderen Diensten und Applikationen. Vergleicht man die Bandbreite innerhalb und zwischen Clustern, stellt man fest, dass für die Kommunikation zweier Ressourcen aus unterschiedlichen Clustern nur ein Bruchteil der Bandbreite einer clusterinternen Verbindung zur Verfügung steht. Schließen sich mehrere Organisationen zusammen, erfolgt die Kommunikation auf *Interorganisationsebene* meist über WAN-Verbindungen. Hierbei handelt es sich häufig um angemietete Leitungen, die nach Bandbreite oder Nutzung verrechnet werden. Diese Verbindungen werden von allen Anwendungen, Diensten und Personen für die externe Kommunikation genutzt, so dass diese äußerst wichtig für die Organisation sind und starke Nutzungsschwankungen aufweisen können. Die Kommunikation zwischen den Ressourcen unterschiedlicher Organisationen kann einerseits über spezielle Organisations-Gateways oder andererseits direkt über die Cluster-Gateways erfolgen.

Auch die Kosten für die Benutzung der Netzwerkverbindungen verändern sich mit jeder Ebene. So kann die Nutzung des Netzwerks auf Clusterebenen als nahezu kostenlos betrachtet werden, da außer den Anschaffungs- und Wartungskosten für die Organisation keine Kosten für die Benutzung anfallen. Da das Clusternetzwerk sowohl für die Interprozesskommunikation verteilter Anwendungen als auch für den

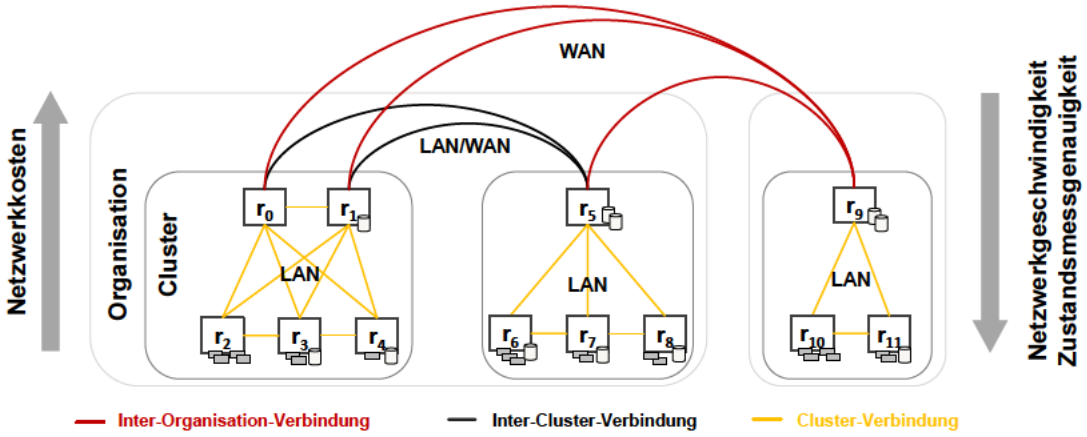


Abbildung 3.1: Das entwickelte hierarchische Umgebungsmodell.

Datentransfer genutzt wird, können allerdings geringe virtuelle Nutzungskosten eingeführt werden, um eine unnötige Belastung zu vermeiden.
Bei Verbindungen auf Interclusterebene muss zwischen LAN- und WAN-Verbindungen unterschieden werden:
Handelt es sich um LAN-Verbindungen innerhalb eines Standortes, fallen wie im Cluster keine Nutzungskosten an. Es können jedoch, wie auf Clusterebene, virtuelle Kosten eingeführt werden, damit diese Verbindungen nicht mit Datentransfers zwischen den Clustern überlastet werden und dadurch wichtige Anwendungen und Dienste der Organisation beeinträchtigt werden. Verglichen mit den Kosten von Clusterverbindungen, sind die Kosten der LAN-Verbindungen auf Interclusterebene daher deutlich höher. Sind die Cluster über ein WAN verbunden, kommen noch die mit dem Anbieter vereinbarten Miet- oder Nutzungskosten hinzu.
Gleiches gilt für die Verbindungen zwischen den Organisationen, wobei diese Verbindungen auch von anderen, teils geschäftskritischen Anwendungen und Diensten der Organisationen genutzt werden. Zur Vermeidung einer Überlastung dieser sensiblen Ressourcen können noch virtuelle Kosten für die Verwendung dieser Netzwerkverbindungen eingeführt werden, so dass die Gesamtkosten deutlich höher sind als für Verbindungen innerhalb einer Organisation. Die Interorganisationsverbindungen können damit als die teuerste Netzwerkressource angesehen werden.

Die Komplexität organisationsübergreifender, globaler Netze macht die Zustandsbestimmung der einzelnen Netzwerkkomponenten äußerst anspruchsvoll [WSH99]. Hierbei erschweren insbesondere die Vielzahl an Verbindungen sowie die unterschiedlichen Netzwerktechnologien die Bestimmung und Vorhersage der Netzwerkauslastungen. Darüber hinaus muss bei der Betrachtung der Netzwerkauslastung beachtet werden, dass die Netzwerkverbindungen zwischen Clustern und Organisation nicht nur für die Verarbeitung der Berechnungsaufträge, sondern für andere Anwendungen und Dienste der Organisationen, mit eventuell höherer Priorität, verwendet werden. Die Auslastung einer Netzwerkverbindung hängt somit von vielen

unterschiedlichen Faktoren ab, wobei der Aufwand der Zustandsbestimmung mit jeder Ebene der Netzwerkhierarchie steigt und die Genauigkeit entsprechend abnimmt. Abbildung 3.1 verdeutlicht den Zusammenhang zwischen Netzwerkgeschwindigkeit, Netzwerkzugriffskosten sowie Zustandsmessgenauigkeit und der Hierarchieebene noch einmal.

3.1.2 Das Schedulingmodell

Das Umgebungsmodell und die in Abschnitt 1.1 vorgestellte Auftragsdefinition bilden die Basis des in dieser Arbeit entwickelten Schedulingmodells. Die Zielfunktion stellt hierbei das zentrale Element des Schedulingmodells dar, da der Scheduler anhand dieser eine für das auszuführende Programm geeignete Rechenressource und für jeden Datensatz eine Speicherressource mitsamt den zur Datenübertragung benötigten Netzwerkressourcen wählt. Für das Scheduling ressourcenintensiver Anwendungen werden aktuell am Häufigsten die mittlere oder maximale Ausführungsdauer *oder* die mittleren Kosten über alle Aufträge als Zielfunktionen verwendet [AD06]. Diese spiegeln zwei unterschiedliche Sichten auf die Ressourcennutzung wieder. Aus Sicht der Benutzer ist eine zeitnahe Verarbeitung ihrer Aufträge erwünscht, so dass die Ausführungsdauer das wichtigste Kriterium für diese Interessengruppe darstellt. Dem gegenüber stehen die Betreiber und Verantwortlichen der Organisationen, die eine möglichst effiziente Ressourcennutzung anstreben. Die Minimierung der Kosten ist aus dieser Sicht das primäre Schedulingziel.

Anstatt sich, wie die bestehenden Verfahren, auf eine der Dimensionen - Kosten *oder* Dauer - zu beschränken, berücksichtigt die in dieser Arbeit entwickelte Zielfunktion beide Dimensionen. Das Ziel des Schedulings ist daher die Minimierung des mittleren *Verarbeitungsaufwandes*[5]

$$\varphi(Z) = \sum_{z \in Z} \tau(z) \cdot \kappa(z) \tag{3.1}$$

über alle Aufträge, wobei sich der Verarbeitungsaufwand einer Zuordnung $z = (w, c, s_1, ..., s_m)$ für einen Auftrag $a = (e, p, d_1, ..., d_m)$ aus der Multiplikation der Ausführungsdauer $\tau(z)$ mit den Ausführungskosten $\kappa(z)$ ergibt. Die Aufgabe des Schedulers besteht somit darin, für alle n Aufträge eine Zuordnung Z^* zu finden, so dass die Zielfunktion $\varphi(Z)$ minimiert wird:

$$\text{Finde } Z^* \text{ mit } \varphi(Z^*) = \min_{Z \in \mathbf{Z}} \{\varphi(Z)\} \tag{3.2}$$

wobei $Z = (z_1, z_2, ..., z_n)$ eine Zuordnungsmöglichkeit für alle n Aufträge und $\mathbf{Z} = \{Z_1, Z_2, ...,\}$ alle unterschiedlichen Zuordnungsmöglichkeiten für diese n Aufträge beschreiben.

Die Minimierung des mittleren Verarbeitungsaufwandes garantiert eine zeit- und kostenoptimale Ausführung der Aufträge. Ein Scheduling mit dem Ziel, die mittlere

[5] Da es sich um ein Optimierungsproblem handelt kann der konstante Term $|Z|$ des mittleren Verarbeitungsaufwands $1/|Z| \cdot \sum_{z \in Z} \tau(z) \cdot \kappa(z)$ vernachlässigt werden

oder maximale Ausführungsdauer zu minimieren würde hingegen schnelle Ressourcen bevorzugen. Dies würde zu deutlich höheren Kosten führen, da schnelle Ressourcen im Allgemeinen teurer sind. Wäre das Ziel des Schedulings andererseits die Minimierung der mittleren Kosten, würden günstige Ressourcen gewählt, wodurch die Ausführungsdauern steigen würden.
Ein Scheduling zur Minimierung des mittleren Verarbeitungsaufwandes bevorzugt Zuordnungen mit einem ausgewogenen Verhältnis zwischen Ausführungsdauer und Ausführungskosten. Die Mittelung des Aufwandes $\tau(z) \cdot \kappa(z)$ der einzelnen Aufträge garantiert des Weiteren, dass nicht der Aufwand einzelner Aufträge auf Kosten anderer Aufträge minimiert wird.

Im Folgenden werden die Ausführungsdauer und die Ausführungskosten auf Grundlage des vorgestellten Umgebungsmodelles definiert. In Tabelle 3.1.2 werden die nachfolgend verwendeten Bezeichner eingeführt. Die Ausführungsdauer $\tau(z)$ einer Zuordnung z

$$\tau(z) = \tau_{wait}(z) + \max_{1 \leq i \leq m} \{\tau_{tra}(d_i, s_i, c)\} + \tau_{cpu}(p, c, d_1, ..., d_m) \tag{3.3}$$

setzt sich aus der Wartezeit τ_{wait}, der längsten Transferdauer τ_{tra} und der Berechnungsdauer τ_{cpu} zusammen. Die Berechnungsdauer

$$\tau_{cpu}(p, c, d) = o(p, d)/g_{cpu}(c) \tag{3.4}$$

ergibt sich aus der Anzahl der benötigten Berechnungsoperationen $o(p, d_1, ..., d_m)$, die für die Verarbeitung der Eingabedaten benötigt werden sowie aus der Prozessorgeschwindigkeit $g_{cpu}(c)$ der Rechenressource.

Wie aus dem Umgebungsmodell hervorgeht, ist aufgrund der Struktur des Netzwerks eine direkte Kommunikation zwischen zwei Ressourcen nicht immer möglich. Ist keine direkte Kommunikation möglich, etwa zwischen zwei Ressourcen aus unterschiedlichen Clustern, werden die Daten über definierte Gateways geleitet. Prinzipiell können unterschiedliche Komponenten als Gateways eingesetzt werden, wobei in dieser Arbeit - wie auch in verwandten Arbeiten [Ven06] [MAS+07] - nur Speicherressourcen als Gateways fungieren können. Beim Transfer über mehrere Stationen muss der Datensatz auf einer Station zunächst vollständig angekommen sein, bevor dieser zur nächsten Station transferiert werden kann. Darüber hinaus erfolgt die Datenübertragung zwischen zwei Ressourcen immer auf dem kürzesten Pfad. Dies entspricht dem Verhalten aktueller Grid-Implementierungen und stellt den allgemeinsten Fall des Datentransfers dar. Seien nun $s_1, ..., s_m$ die unter der Zuordnung z den Datensätzen $d_1, ..., d_m$ zugeordneten Speicherressourcen und bezeichne für jeden der $1 \leq i \leq m$ Transfers $s_{i_0}, ..., s_{i_l}, s_{i_{l+1}}$ den kürzesten Pfad zwischen $s_{i_0} = s_i$ und $s_{i_{l+1}} = c$, dann ergibt sich die Dauer des Transfers eines Datensatzes zu

$$\tau_{tra}(d_i, s_i, c) = \sum_{j=0}^{l} \max\{\tau_{hdd}(s_{i_j}, d_i), \tau_{net}(s_{i_j}, s_{i_{j+1}}, d_i)\} \tag{3.5}$$

wobei τ_{hdd} die benötigte Speicherzugriffsdauer und τ_{net} die Netzwerktransferdauer beschreiben. Die Speicherzugriffsdauer $\tau_{hdd}(s, d)$ erfasst die Zeit, die die Speicherres-

source s benötigt um d zu lesen oder zu schreiben. Entsprechend erfasst die Netzwerktransferdauer $\tau_{net}(s_1, s_2, d)$ die Zeit, die benötigt wird, um d über die Verbindung s_1, s_2 zu transferieren. Es wird davon ausgegangen, dass ein Datensatz blockweise transferiert wird. Ein Block des Datensatzes wird hierbei von der Speicherressource gelesen und anschließend über die Netzwerkressource übertragen.[6] Die Transferdauer eines Datensatzes zwischen zwei Stationen wird somit von der langsameren, der Speicher- oder der Netzwerkressource, bestimmt. Die aktuelle Geschwindigkeit der Ressourcen lässt sich wiederum aus den Geschwindigkeiten und der Auslastung der Speicherressource beziehungsweise der Netzwerkverbindung ermitteln.

Die Bestimmung der Kosten einer Zuordnung orientiert sich ebenfalls am Umgebungsmodell sowie den aktuellen Verrechnungsmodellen von Ressourcenanbietern wie Amazon, Google oder der Deutschen Telekom. Aktuelle Verrechnungsmodelle basieren häufig primär auf der Nutzungsdauer[7], so dass auch in dieser Arbeit den Rechen-, Speicher- und Netzwerkressourcen zeitabhängige Benutzungskosten zugeordnet werden, etwa €/sec. Zusätzliche Fixkosten, wie monatliche Grundkosten, können anteilig auf die Nutzungskosten addiert werden. Die bei der Verwendung der Ressourcen entstehenden Kosten müssen nicht unbedingt an externe Anbieter entrichtet werden, sondern können auch innerhalb einer Organisation entstehen.

Die Kosten einer Zuordnung für einen Auftrag ergeben sich somit zu

$$\kappa(z) = \sum_{i=1}^{m} \kappa_{tra}(d_i, s_i, c) + \kappa_{cpu}(p, c, d_1, ..., d_m) \tag{3.6}$$

mit den Transferkosten κ_{tra} und Berechnungskosten κ_{cpu}. Fallen für die Verwendung einer Rechenressource k_{cpu} Kosten pro Zeiteinheit an, ergeben sich die Berechnungskosten abhängig von der Berechnungsdauer τ_{cpu} zu

$$\kappa_{cpu}(p, c, d_1, ..., d_m) = \tau_{cpu}(p, c, d_1, ..., d_m) \cdot k_{cpu}(c) \tag{3.7}$$

Entsprechend ergeben sich die Kosten für den Transfer aus den Kosten pro Zeiteinheit k_{hdd} und k_{net} für den Zugriff auf die Speicher- und Netzwerkressourcen.

$$\kappa_{tra}(d_i, s_i, c) = \sum_{j=0}^{l} \max\{\tau_{hdd}(s_{i_j}, d_i), \tau_{net}(s_{i_j}, s_{i_{j+1}}, d_i)\} \cdot (k_{hdd}(s_{i_j}) + k_{net}(s_{i_j}, s_{i_{j+1}})) \tag{3.8}$$

Da sowohl die Netzwerk- als auch die Speicherressource während der Transferdauer belegt sind, fallen auch für beide für diese Zeit Kosten an.

[6] Dies entspricht dem Verhalten aktueller Betriebsysteme, wobei auch mehrere Blöcke auf einmal gelesen werden können[Tan07]

[7] http://aws.amazon.com/ec2/pricing

P	Alle Programme inklusive ihrer Anforderungen
D	Alle verfügbaren Datensätze
n, n_o, n_c	Die Anzahl der Ressourcen, Organisationen, Cluster
R	Alle n Ressourcen im System $(r_1, r_2, \ldots, r_n)$
N	$(r_1, r_2) \in R \times R$, wobei r_1 und r_2 Daten direkt austauschen können
R_d	Alle $r \in R$ die $d \in D$ speichern
$a = (e, p, d_1, ..., d_m)$	Auftrag mit Eintrittszeitpunkt e, Programm p, und Daten $d_1, ..., d_m$
$A = \{a_1, ...a_{n_a}\}$	Alle n_a zu verarbeitenden Aufträge
$z = (w, c, s_1, ..., s_m)$	Zuordnung mit Wartezeit w, Rechen- und Speicherressourcen $c, s_1, ..., s_m$
$Z = \{z_1, ...z_{n_a}\}$	Die n_a Zuordnungen für alle Aufträge
$\varphi : Z \rightarrow \Re$	Verarbeitungsaufwand der Zuordnungen für alle Aufträge
$\kappa(z)$	Ausführungskosten der Zuordnung z
$\kappa_{tra}(d, r_1, r_2)$	Transferkosten für d von r_1 nach r_2
$\kappa_{cpu}(p, r, d_1, ..., d_m)$	Berechnungskosten von $d_1, ..., d_m$ mit p auf r
$\tau(z)$	Ausführungsdauer der Zuordnung z
$\tau_{cpu}(p, c, d_1, ..., d_m)$	Berechnungsdauer von $d_1, ..., d_m$ mit p auf c
$\tau_{wait}(z)$	Wartezeit der Zuordnung z
$\tau_{tra}(d, s, c)$	Transferdauer für d von s nach c
$\tau_{hdd}(d, r)$	Speicherzugriffsdauer für d auf r (**h**ard **d**isk **d**rive)
$\tau_{net}(d, r_1, r_2)$	Netzwerktransferdauer für d von r_1 nach r_2
$k_{net}(r_1, r_2)$	Netzwerkzugriffskosten von r_1 nach r_2
$k_{hdd}(r)$	Speicherzugriffskosten von r
$k_{cpu}(r)$	Prozessornutzungskosten von r
$g_{net}(r_1, r_2)$	Netzwerkgeschwindigkeit zwischen r_1 und r_2
$g_{hdd}(r)$	Speichergeschwindigkeit von r
$g_{cpu}(r)$	Prozessorgeschwindigkeit von r
$o(p, d_1, ..., d_m)$	Anzahl der Berechnungsoperationen für die Verarbeitung der Eingabedaten $d_1, ..., d_m$ mit p

Aus der so definierten Ausführungsdauer und Ausführungskosten lässt sich der Verarbeitungsaufwand einer Zuordnung bestimmen. Neben der Zielfunktion bildet der Schedulingmodus die zweite zentrale Komponente des Schedulingmodells. Wie im vorherigen Kapitel bereits erwähnt, unterscheidet man zwischen Offline- und Online-Schedulern. Beim Offline-Scheduling geht man davon aus, dass dem Scheduler alle Aufträge - oder zumindest alle relevanten Aufträge - bekannt sind und der Scheduler diese auf die vorhandenen Ressourcen verteilen muss. Offline-Scheduler eignen sich daher insbesondere für statische Umgebungen mit einer Vielzahl an Aufträgen. Ein Online-Scheduler erfährt hingegen von einem Auftrag erst zu dessen Eintrittszeitpunkt. Der Online-Scheduler muss daher für jeden einzelnen Auftrag eine Ressourcenzuordnung erzeugen. Hierbei muss aber genauso wie beim Offline-Scheduling die Zielfunktion über alle Aufträge minimiert werden. Offensichtlich sind Online-Scheduler deutlich besser für dynamische Umgebungen, wie Grids, geeignet als Offline-Scheduler, so dass sich die nachfolgenden Betrachtungen in dieser Arbeit auf das Online-Scheduling konzentrieren.

Beim klassischen Online-Scheduling, auch *List-Scheduling* genannt, werden die Aufträge streng in der Reihenfolge ihres Eintreffens abgearbeitet. Sind beim Eintreffen keine Ressourcen frei oder ist die vom Scheduler gewählte Rechenressource belegt, wird der Auftrag bis zur nächsten Schedulingphase an eine Warteliste angefügt. Beim nächsten Scheduling wird diese Liste streng der Reihe nach abgearbeitet. Sollen Aufträge priorisiert werden können, kann dies einfach durch eine Sortierung dieser Liste erfolgen, ohne das der eigentliche Schedulingalgorithmus geändert werden muss. Bei der nachfolgenden Vorstellung des entwickelten Schedulers wird daher die Priorisierung nicht weiter betrachtet, kann jedoch, falls dies in einem Anwendungsfall erforderlich ist, leicht integriert werden. Eine weitere Designfrage bei Online-Schedulern ist die Häufigkeit des Schedulings. Im gewählten Modell wird ein Scheduling beim Eintreffen eines neuen Auftrags, beim frei werden einer Rechenressource sowie in festgelegten Abständen durchgeführt.

Das Ziel des in dieser Arbeit entwickelten DOHS-Schedulingalgorithmus ist es, eine Zuordnung für jeden einzelnen Auftrag zu erzeugen, so dass der Verarbeitungsaufwand 3.1 über alle Aufträge minimiert wird. Im Folgenden werden alle Annahmen und Rahmenbedingungen der Umgebung sowie des Schedulings noch einmal zusammengefasst:

- Es können mehrere Repliken eines Datensatzes vorhanden sein. Es dürfen vom Scheduler keine zusätzlichen Repliken erzeugt werden. Werden Datensätze auf andere Ressourcen transferiert, werden diese dort nach der Verarbeitung gelöscht.
- Ein Verarbeitungsauftrag besteht aus einem Programm und beliebig vielen Eingabedatensätzen. Ein Auftrag wird dem System erst bei dessen Eintrittszeitpunkt bekannt. Die Anzahl der Aufträge ist nicht bekannt.
- Das System verfügt über keinerlei Informationen über die auszuführenden Programme. Die Berechnungsdauer eines Auftrags ist somit nicht bekannt. Die Größe sowie die Speicherorte der Datensätze sind bekannt.
- Ein Programm kann unterschiedliche Anforderungen an eine Rechenressource stellen, etwa ein bestimmtes Betriebssystem oder die minimale Hauptspeichergröße. Ein Programm benötigt eine bestimmte Anzahl Prozessorkerne auf *einer* Rechenressource. Aufträge, die *mehr als eine* Ressource benötigen, werden in dieser Arbeit *nicht* betrachtet.
- Eine Mehrfachbelegung von Prozessorkernen ist nicht erlaubt. Ebenso kann ein einmal gestarteter Auftrag nicht mehr angehalten und auf eine andere Ressource übertragen werden - nicht unterbrechbare (non-preemptive) Ausführung.
- Die Verarbeitung eines Auftrags erfolgt in drei Abschnitten: Zuweisung des Auftrags auf eine Rechenressource; Transfer aller Datensätze von den Speicherressourcen zu der Rechenressource; nach Beendigung aller Transfers startet die Ausführung des Programmes.
- Besteht beim Datentransfer keine direkte Verbindung zwischen den Ressourcen, muss der Datensatz auf allen Zwischenstationen zunächst vollständig angekommen sein, bevor er weitergeleitet werden kann.
- Speicher- sowie Netzwerkressourcen können für mehrere Transfers gleichzeitig genutzt werden. Die parallelen Transfers teilen sich hierbei die zur Verfügung stehende Bandbreite gleichmäßig.
- Ressourcen können jederzeit aus dem System entfernt werden und es können neue Ressourcen hinzukommen.
- Es stehen keine exakten Ressourceninformationen zur Verfügung.
- Globaler Scheduler. Alle Aufträge werden durch einen zentralen Scheduler auf die Ressourcen verteilt. Sind zusätzlich lokale Scheduler im System vorhanden, muss zumindest gewährleistet sein, dass die entstehende Ressourcennutzung für den globalen Scheduler ersichtlich ist.

3.2 Herleitung der Schedulingfunktion

In die Entwicklung des Schedulingalgorithmus flossen neben dem Umgebungsaufbau auch die Zuverlässigkeit der Ressourceninformationen sowie die Besonderheiten des Online-Scheduling und die Erkenntnisse der im vorherigen Kapitel vorgestellten bestehenden Verfahren mit ein. Die Grundlage des Algorithmus bildet eine aus der vorgegebenen Zielfunktion φ (3.1) abgeleitete *Schedulingfunktion* ϕ. Der entwickelte Scheduler bestimmt mit dieser Schedulingfunktion aus einer Reihe von möglichen Zuordnungen die aufwandsoptimale Zuordnung für den aktuellen Auftrag. Die Ableitung der Schedulingfunktion erfolgt nach den folgenden vier zentralen Eigenschaften:

- *Daten- und Rechenaufwand.* Wie die in Abschnitt 2.1.3 beschriebenen Scheduler gezeigt haben, muss für daten- und rechenintensive Anwendungen neben dem Berechnungsaufwand auch der Datentransfer beim Scheduling berücksichtigt werden.

- *Netzwerkstruktur.* Die in Abschnitt 2.2.2 erläuterten Entwicklungen im Bereich der skalierbaren Datenverarbeitung sehen im Netzwerk eine zentrale, beschränkte Ressource. Um eine effiziente Ausführung von datenintensiven Anwendungen zu gewährleisten, muss diese Ressourcengruppe geschont und die Struktur des Netzwerkes berücksichtigt werden.

- *Effiziente Ressourcennutzung.* Wie für rein rechenintensive Anwendungen gezeigt werden konnte [Sga97], müssen beim Online-Scheduling Ressourcen für nachfolgende Aufträge frei gehalten werden. Damit für nachfolgende Aufträge noch genügend Ressourcen zur Verfügung stehen, sollte daher eine Zuordnung gewählt werden, die eine gute Balance zwischen Verarbeitungsaufwand und Ressourcennutzung bietet. Betrachtet man die Ressourcenanforderung für einen Auftrag, benötigt dieser eine Rechenressource und für jeden Datensatz eine Speicherressource. Müssen die Daten transferiert werden, kommen noch Netzwerkressourcen und mit steigender Distanz noch weitere Speicherressourcen als Zwischenstationen hinzu. Somit lässt sich die Anzahl der für einen Auftrag verwendeten Ressourcen verringern, wenn eine Rechenressource gewählt wird, welche möglichst nahe bei den Speicherressourcen der Datensätze liegt. Der Ressourcenbedarf einer Zuordnung hängt damit direkt mit der Distanz zwischen Rechen- und Speicherressourcen zusammen. Die entwickelte Schedulingfunktion bevorzugt daher Zuordnungen, die nicht nur den Aufwand für den aktuellen Auftrag minimieren, sondern auch möglichst ressourcenschonend sind.

- *Informationsqualität.* Die dynamische und komplexe Umgebung erschwert die Erhebung exakter Informationen über die Ressourcen und Aufträge. Bei Schedulern, die von exakten Informationen ausgehen, können die ungenauen Ressourcen- und Auftragsinformationen erheblichen Einfluss auf die erzeugten Zuordnungen nehmen. So kann die Überschätzung der Ausführungsdauer eines

Auftrags dazu führen, dass auf das Freiwerden einer schnelleren Rechenressource gewartet sowie ein größerer Transferaufwand in Kauf genommen wird. Wie die im nachfolgenden Kapitel durchgeführten Simulationen zeigen, führen insbesondere ungenaue Netzwerkinformationen sowie Ausführungsdauern bei diesen Schedulern zu einem deutlich erhöhten Verarbeitungsaufwand. Der Schedulingalgorithmus sollte daher mit ungenauen Ressourcen- und Auftragsinformationen umgehen können und möglichst nur wenige, einfach zu ermittelnde, qualitative Informationen verwenden.

Die Schedulingfunktion ϕ muss somit einerseits mit unvollständigen Umgebungs- und Auftragsinformationen möglichst gut den Verarbeitungsaufwand φ (3.1) der Ressourcenzuordnungen für *einen* Auftrag approximieren. Andererseits muss die Funktion ϕ so gewählt werden, dass für die unbekannte Anzahl nachfolgender Aufträge noch genügend Ressourcen zur Verfügung stehen.
In der nachfolgend vorgestellten Approximation der Zielfunktion φ durch die Schedulingfunktion ϕ erfolgt keine Fehlerabschätzung, da unter realen Bedingungen keine exakte Bestimmung aller in der Funktion φ enthaltenen Terme möglich ist und somit einer Fehlerabschätzung der Approximation wenig Aussagekraft zukommen würde.

Die dem entwickelten Schedulingalgorithmus zugrunde liegende Schedulingfunktion wird nachfolgend in drei Schritten hergeleitet. Hierbei werden die in der Zielfunktion φ (3.1) verwendeten, schwer zu ermittelnden Informationen, insbesondere die Netzwerkinformationen sowie die Berechnungsoperationen eines Auftrags, sukzessive entfernt. Zunächst wird der einfachste Fall, eine Zuordnung für einen Datentransfer innerhalb eines Cluster mit nur einem Datensatz, betrachtet. Die daraus abgeleitete Approximation der Zielfunktion wird im zweiten Schritt auf Datentransfers zwischen Clustern und Organisationen erweitert. Im letzten Schritt wird die Schedulingfunktion auf den allgemeinsten Fall, Zuordnungen mit mehr als einem Datensatz, vervollständigt.

3.2.1 Datentransfer innerhalb eines Clusters

Zur Herleitung wird im Folgenden zunächst die in 3.1 definierte Zielfunktion einer Zuordnung z_{clu} für *einen* Auftrag mit *einem* Datensatz d und einem Transfer innerhalb eines Clusters betrachtet. Der Datensatz d ist hierbei auf der Speicherressource s gespeichert und die Ausführung wird auf der Rechenressource c durchgeführt. Ohne Beschränkung der Allgemeinheit wird die Wartezeit der Ausführung nachfolgend auf $\tau_{wait} = 0$ festgelegt. Für die Zielfunktion dieser Zuordnung z_{clu}

$$\varphi(z_{clu}) = \tau(z_{clu}) \cdot \kappa(z_{clu}) \tag{3.9}$$

mit der entsprechenden Dauer nach 3.3

$$\begin{aligned} \tau(z_{clu}) &= \tau_{tra}(d, s, c) + \tau_{cpu}(p, c, d) \\ &= \max\{\tau_{hdd}(s, d), \tau_{net}(s, c, d)\} + \tau_{cpu}(p, c, d) \end{aligned}$$

und den Kosten nach 3.6

$$\begin{aligned}\kappa(z_{clu}) &= \kappa_{tra}(d,s,c) + \kappa_{cpu}(p,c,d) \\ &= \max\{\tau_{hdd}(s,d), \tau_{net}(s,c,d)\} \cdot (k_{hdd}(s) + k_{net}(s,c)) \\ &\quad + \tau_{cpu}(p,c,d) \cdot k_{cpu}(c)\end{aligned}$$

wird nachfolgend eine Approximation vorgestellt.

Wie in Abschnitt 3.1.2 bereits erläutert, wird die Transferdauer τ_{tra} zwischen zwei Ressourcen entweder durch die Speicherzugriffsdauer τ_{hdd} oder durch die Netzwerktransferdauer τ_{net} bestimmt, je nachdem welche von beiden länger dauert. Aufgrund der vorhergegangenen Betrachtung der Umgebung kann angenommen werden, dass bei Transfers innerhalb eines Clusters nicht das günstige, schnelle Clusternetzwerk, sondern die Geschwindigkeit und die Kosten der Speicherressource die bestimmenden Faktoren sind. Unter diesen Annahmen kann beim Transfer innerhalb eines Clusters die Netzwerktransferdauer τ_{net} vernachlässigt werden, so dass man die Transferdauer für einen Clustertransfer nur durch die Speicherzugriffsdauer τ_{hdd} approximieren kann:

$$\tau_{tra}(d,s,c) = \max\{\tau_{hdd}(s,d), \tau_{net}(s,c,d)\} \approx \tau_{hdd}(s,d) \tag{3.10}$$

Übertragt man die approximierte Transferdauer auf die Transferkosten und vernachlässigt zusätzlich auch die Netzwerkzugriffskosten k_{net}, erhält man die folgende Approximation für die Transferkosten innerhalb eines Clusters:

$$\begin{aligned}\kappa_{tra}(d,s,c) &= \max\{\tau_{hdd}(s,d), \tau_{net}(s,c,d)\} \cdot (k_{hdd}(s) + k_{net}(s,c)) \qquad (3.11) \\ &\approx \tau_{hdd}(s,d) \cdot k_{hdd}(s)\end{aligned}$$

Multipliziert man die Zielfunktion 3.9 für den Transfer im Cluster aus und setzt

$$\begin{aligned}\varphi(z_{clu}) &= \tau(z_{clu}) \cdot \kappa(z_{clu}) \qquad (3.12) \\ &= (\tau_{tra}(d,s,c) + \tau_{cpu}(p,c,d)) \cdot (\kappa_{tra}(d,s,c) + \kappa_{cpu}(p,c,d)) \\ &= \tau_{tra}(d,s,c) \cdot \kappa_{tra}(d,s,c) + \tau_{tra}(d,s,c) \cdot \kappa_{cpu}(p,c,d) \\ &\quad + \tau_{cpu}(p,c,d) \cdot \kappa_{tra}(d,s,c) + \tau_{cpu}(p,c,d) \cdot \kappa_{cpu}(p,c,d)\end{aligned}$$

die zuvor beschriebenen Approximationen $\tau_{hdd}(s,d)$ für $\tau_{tra}(d,s,c)$ (3.10) sowie $\tau_{hdd}(s,d) \cdot k_{hdd}(s)$ für $\kappa_{tra}(d,s,c)$ (3.11) ein, ergibt sich:

$$\begin{aligned}\varphi(z_{clu}) &\approx \tau_{hdd}(s,d) \cdot \tau_{hdd}(s,d) \cdot k_{hdd}(s) + \tau_{hdd}(s,d) \cdot \kappa_{cpu}(p,c,d) \qquad (3.13) \\ &\quad + \tau_{cpu}(p,c,d) \cdot \tau_{hdd}(s,d) \cdot k_{hdd}(s) + \tau_{cpu}(p,c,d) \cdot \kappa_{cpu}(p,c,d)\end{aligned}$$

Ersetzt man nun noch $\kappa_{cpu}(p,c,d)$ durch $\tau_{cpu}(p,c,d) \cdot k_{cpu}(c)$ (3.7) erhält man:

$$\begin{aligned}\varphi(z_{clu}) &\approx \tau_{hdd}(s,d) \cdot \tau_{hdd}(s,d) \cdot k_{hdd}(s) + \tau_{hdd}(s,d) \cdot \tau_{cpu}(p,c,d) \cdot k_{cpu}(c) \\ &\quad + \tau_{cpu}(p,c,d) \cdot \tau_{hdd}(s,d) \cdot k_{hdd}(s) + \tau_{cpu}(p,c,d) \cdot \tau_{cpu}(p,c,d) \cdot k_{cpu}(c) \\ &= \tau_{hdd}(s,d)^2 \cdot k_{hdd}(s) \\ &\quad + \tau_{hdd}(s,d) \cdot \tau_{cpu}(p,c,d) \cdot (k_{cpu}(c) + k_{hdd}(s)) \\ &\quad + \tau_{cpu}(p,c,d)^2 \cdot k_{cpu}(c)\end{aligned} \tag{3.14}$$

Bei datenintensiven Anwendungen dominiert der *Transferterm* $\tau_{hdd}(s,d)^2 \cdot k_{hdd}(s)$, wohingegen für rechenintensive Anwendungen der *Berechnungsterm* $\tau_{cpu}(p,c,d)^2 \cdot k_{cpu}(c)$ die bestimmende Größe darstellt. Da die beiden Terme alle in 3.14 enthaltene Dauern (τ_{hdd}, τ_{cpu}) und Kosten (k_{hdd}, k_{cpu}) beinhalten, bieten diese Terme auch eine geeignete Approximation für Anwendungen, die sowohl daten- als auch rechenintensiv sind. Reduziert man die approximierte Zielfunktion 3.14 auf diese beiden Terme und ersetzt noch $\tau_{cpu}(p,c,d)$ durch $o(p,d)/g_{cpu}(c)$ (3.4) erhält man folgende Approximation:

$$\begin{aligned}\varphi(z_{clu}) &\approx k_{hdd}(s) \cdot \tau_{hdd}(s,d)^2 + k_{cpu}(c) \cdot \tau_{cpu}(p,c,d)^2 \\ &= k_{hdd}(s) \cdot \tau_{hdd}(s,d)^2 + k_{cpu}(c) \cdot o(p,d)^2/g_{cpu}(c)^2\end{aligned} \tag{3.15}$$

Diese Approximation könnte bereits als Schedulingfunktion für den Vergleich unterschiedlicher Zuordnungen eines Auftrags genutzt werden. Wie eingehend diskutiert, können die Auslastung, die Kosten und die Geschwindigkeit einer Speicherressource sowie die Prozessorgeschwindigkeit und die Kosten einer Rechenressource einfach und genau bestimmt werden, so dass $\tau_{hdd}(s,d)$, $k_{hdd}(s)$, $g_{cpu}(c)$ und $k_{cpu}(c)$ aus 3.15 dem Scheduler zur Verfügung stehen. Im Allgemeinen sind die von einer Anwendung benötigten Berechnungsoperationen $o(p,d)$ jedoch nicht bekannt und müssen daher aus der Approximation eliminiert werden.

Würde man nur die Rechenressourcen beim Scheduling betrachten könnten die Berechnungsoperationen beim Scheduling *eines* Auftrags einfach vernachlässigt werden. Für den Vergleich unterschiedlicher Zuordnungen wäre nur der Berechnungsterm $k_{cpu}(c) \cdot o(p,d)^2/g_{cpu}(c)^2$ relevant und da die Berechnungsoperationen $o(p,d)$ für alle Rechenressourcen identisch sind, wären diese für einen Vergleich irrelvant:

$$k_{cpu}(c) \cdot o(p,d)^2/g_{cpu}(c)^2 \approx k_{cpu}(c)/g_{cpu}(c)^2 \tag{3.16}$$

Die daraus resultierende Schedulingfunktion ϕ_1

$$\phi_1(z_{clu}) = \tau_{hdd}(s,d)^2 \cdot k_{hdd}(s) + k_{cpu}(c)/g_{cpu}(c)^2 \tag{3.17}$$

enthält nun nur noch einfach zu ermittelnde Informationen. Allerdings lässt sie auch keinen quantitativen Vergleich zwischen Transferaufwand und Rechenaufwand mehr

zu, da durch den Wegfall von $o(p,d)$ die Rechenaufwandsinformation verloren geht. Die Addition der beiden Terme ergibt damit nicht mehr den Gesamtaufwand, so dass die Funktion 3.17 nicht mehr für den Vergleich des Gesamtaufwandes unterschiedlicher Zuordnungen geeignet ist. Ein Beispiel mit Werten aus den in Kapitel 4 durchgeführten Simulationen verdeutlicht, wie sich die einfache Elimination der Berechnungsoperationen $o(p,d)$ auf die Schedulingfunktion auswirkt:

$$\begin{aligned} 20^2 \cdot 0,1 + 0,2 \cdot (10000/1000)^2 &= 60 \text{ nach } 3.15 \\ 20^2 \cdot 0,1 + 0,2 \cdot (1/1000)^2 &= 40,0000002 \text{ nach } 3.17 \end{aligned}$$

Mit $\tau_{hdd}(s,d) = 20$, $k_{hdd}(s) = 0,1$, $k_{cpu}(c) = 0,2$, $o(p,d) = 10000$ und $g_{cpu}(c) = 1000$. Wie das Beispiel zeigt, dominiert nach der Elimination der Berechnungsoperationen der Transferterm die Schedulingfunktion, so dass der Berechnungsterm für das Scheduling bedeutungslos geworden ist.

Durch eine Transformation des Transferterms und des Berechnungsterms aus ϕ_1 auf ein Intervall $[0;1]$ kann dieser Problematik entgegen gewirkt werden. Anstelle des Gesamtaufwandes wird durch diese Transformation der *relative* Verarbeitungsaufwand eines Auftrags in Bezug auf die möglichen Zuordnungen bestimmt. Seien ψ_t und ψ_c zwei beliebige Funktionen, welche den Transferterm beziehungsweise den Berechnungsterm auf das Intervall $[0;1]$ abbilden, wobei $\psi_t = 1$ für die Zuordnung mit minimalem Transferterm und $\psi_c = 1$ für die Zuordnung mit minimalem Rechenterm gilt, dann ergibt sich die Schedulingfunktion zu:

$$\phi_2(z_{clu}) = \psi_t \left(\tau_{hdd}(s,d)^2 \cdot k_{hdd}(s)\right) + \psi_c \left(k_{cpu}(c)/g_{cpu}(c)^2\right) \tag{3.18}$$

Die Transformation ermöglicht nun den Vergleich unterschiedlicher Zuordnungen, wie groß der absolute Beitrag des Transfer- beziehungsweise des Rechenaufwandes am Gesamtaufwand ist, ist jedoch nicht mehr ersichtlich. Solange der Rechenaufwand und der Transferaufwand sich nicht allzu stark unterscheiden, hat dies für das Scheduling eines Auftrags auch nicht unbedingt negative Auswirkungen. Für extrem daten- oder rechenintensive Aufträge ist es allerdings von Vorteil, wenn der Einfluss eines Terms durch eine Gewichtung verändert werden kann. Dies lässt sich durch die Einführung von Gewichtungsfaktoren realisieren. Mit den Faktoren β_1 und β_2 können Applikationen oder Benutzer dem Scheduler Hinweise geben, ob es sich um einen datenintensiven $\beta_1 > \beta_2$ oder rechenintensiven $\beta_2 > \beta_1$ Auftrag handelt:

$$\phi_3(z_{clu}) = \beta_1 \cdot \psi_t \left(\tau_{hdd}(s,d)^2 \cdot k_{hdd}(s)\right)) + \beta_2 \cdot \psi_c \left(k_{cpu}(c)/g_{cpu}(c)^2\right) \tag{3.19}$$

Im Gegensatz zu den bestehenden Schedulern, die möglichst exakte Schätzungen der Ausführungsdauer durch die Benutzer oder Anwendungen benötigen, reicht es hierfür vollkommen aus, das grobe Verhältnis zwischen Transfer- und Rechenaufwand zu kennen.

3.2.2 Datentransfer außerhalb eines Clusters

Aus der Betrachtung des Datentransfers innerhalb eines Clusters konnte die Schedulingfunktion 3.19 zum Vergleich unterschiedlicher Ressourcenzuordnungen für einen

Auftrag abgeleitet werden. Diese soll anhand des nachfolgenden Beispiels auf Datentransfers zwischen Clustern oder Organisationen erweitert werden.

Hierbei muss sowohl der mehrstufige Transferprozess wie auch die Netzwerkhierarchie beachtet werden. Nachfolgend wird die Zuordnung z_{org} verwendet, bei der ein Datensatz d von der Speicherressource s_g einer anderen Organisation über die Organisationsgateways s_h und s_i zur Rechenressource c transferiert wird.

Die Zielfunktion φ (3.1) ergibt sich bei organisationsübergreifenden Transfers für einen Datensatz zu:

$$\varphi(z_{org}) = (\tau_{tra}(d, s_g, c) + \tau_{cpu}(p, c, d)) \cdot (\kappa_{tra}(d, s_g, c) + \kappa_{cpu}(p, c, d)) \tag{3.20}$$

mit der aus den einzelnen Transferabschnitten ($s_g \rightarrow s_h \rightarrow s_i \rightarrow c$) zusammengesetzten Dauer

$$\begin{aligned}
\tau(z_{org}) &= \tau_{tra}(d, s_g, c) + \tau_{cpu}(p, c, d) \\
&= \max\{\tau_{hdd}(s_g, d), \tau_{net}(s_g, s_h, d)\} \\
&\quad + \max\{\tau_{hdd}(s_h, d), \tau_{net}(s_h, s_i, d)\} \\
&\quad + \max\{\tau_{hdd}(s_i, d), \tau_{net}(s_i, c, d)\} \\
&\quad + \tau_{cpu}(p, c, d)
\end{aligned}$$

und den entsprechenden Kosten:

$$\begin{aligned}
\kappa(z_{org}) &= \kappa_{tra}(d, s_g, c) + \kappa_{cpu}(p, c, d) \\
&= \max\{\tau_{hdd}(s_g, d), \tau_{net}(s_g, s_h, d)\} \cdot (k_{hdd}(s_g) + k_{net}(s_g, s_h)) \\
&\quad + \max\{\tau_{hdd}(s_h, d), \tau_{net}(s_h, s_i, d)\} \cdot (k_{hdd}(s_h) + k_{net}(s_h, s_i)) \\
&\quad + \max\{\tau_{hdd}(s_i, d), \tau_{net}(s_i, c, d)\} \cdot (k_{hdd}s(s_i) + k_{net}(s_i, c)) \\
&\quad + \tau_{cpu}(p, c, d) \cdot k_{cpu}(c)
\end{aligned}$$

Aufgrund der vorhergegangenen Betrachtung der Umgebung kann angenommen werden, dass bei Transfers zwischen Organisationen die Transferdauer durch die Netzwerkverbindungen maßgeblich bestimmt wird. Ebenso sind die Übergtragungskosten dieser Interorganisationsverbindungen meist höher als die Speicherzugriffskosten. Damit dominieren bei Transfers zwischen Clustern oder Organisationen die Netzwerktransferdauer (τ_{net}) und die Netzwerkkosten (k_{net}) die Speicherzugriffsdauer (τ_{hdd}) und die Speicherzugriffskosten (k_{hdd}). Andererseits sind für Transfers innerhalb eines Clusters, wie im vorherigen Abschnitt erläutert, nicht das Netzwerk, sondern die Geschwindigkeit und die Kosten der Speicherressource bestimmend.

Analog zur Approximation 3.10 können unter diesen Annahmen auch bei Transfers über Clustergrenzen hinweg die Transferdauer

$$\tau_{tra}(d, s_g, c) \approx \tau_{hdd}(s_g, d) + \tau_{net}(s_h, s_i, d) + \tau_{hdd}(s_i, d) \tag{3.21}$$

sowie die Transferkosten

$$\begin{aligned} \kappa_{tra}(d, s_g, c) \quad \approx \quad & \tau_{hdd}(s_g, d) \cdot k_{hdd}(s_g) \\ & + \tau_{net}(s_h, s_i, d) \cdot k_{net}(s_h, s_i) \\ & + \tau_{hdd}(s_i, d) \cdot k_{hdd}(s_i) \end{aligned} \tag{3.22}$$

auf die bestimmenden Größen reduziert werden. Hierbei wird für Transfers innerhalb eines Clusters die Speicherzugriffsdauer τ_{hdd} und die Speicherzugriffskosten k_{hdd} verwendet und für Transfers zwischen Clustern oder Organisationen werden die Netzwerktransferdauer τ_{net} und die Netzwerktransferkosten k_{net} genutzt.

Die entstehenden Terme für die Transferdauer 3.21 und 3.22 beinhalten sowohl die Netzwerktransferdauer $\tau_{net}(s_h, s_i, d)$ als auch die Netzwerkkosten $k_{net}(s_h, s_i)$. Die Berechnung der Netzwerktransferdauer erfordert jedoch Informationen über die aktuelle Auslastung und die Netzwerkgeschwindigkeit der Verbindungen zwischen den Organisationen. Wie bereits erläutert, sind diese Informationen insbesondere für Weitverkehrsnetzwerkverbindungen kaum zuverlässig bestimmbar. Ebenso sind die genauen Kosten dieser Verbindungen nicht immer bekannt.

Es liegt nahe, diese ungenauen Informationen durch die einfach zu ermittelnde Dauer und Kosten der entsprechenden Speicherressource s_h zu ersetzen. Dies führt zu einer Approximation der Transferdauer, die ausschließlich Informationen über die beteiligten Speicherressourcen verwendet:

$$\tau_{tra}(d, s_g, c) \approx \tau_{hdd}(s_g, d) + \tau_{hdd}(s_h, d) + \tau_{hdd}(s_i, d) \tag{3.23}$$

Entsprechendes gilt für die Transferkosten:

$$\begin{aligned} \kappa_{tra}(d, s_g, c) \quad \approx \quad & \tau_{hdd}(s_g, d) \cdot k_{hdd}(s_g) \\ & + \tau_{hdd}(s_h, d) \cdot k_{hdd}(s_h) \\ & + \tau_{hdd}(s_i, d) \cdot k_{hdd}(s_i) \end{aligned} \tag{3.24}$$

Die angepasste Transferdauer 3.23 sowie die entsprechenden Transferkosten 3.24 können zwar einfach und zuverlässig berechnet werden, ihnen fehlen aber wichtige Geschwindigkeits- und Kosteninformationen der Netzwerkverbindungen zwischen den Clustern beziehungsweise Organisationen. Aus den Eigenschaften der hierarchischen Netzwerkstruktur organisationsübergreifender Umgebungen lassen sich jedoch Faktoren ableiten, welche das Verhältnis der Geschwindigkeit und der Kosten zwischen den Netzwerkebenen beschreiben. Die Grundlage dieser Faktoren bildet die

Beobachtung, dass mit der Distanz der Netzwerkübertragung einerseits die Kosten steigen und andererseits die zur Verfügung stehende Bandbreite sinkt, so dass das Verhältnis Kosten zu Geschwindigkeit mit jeder Ebene steigt. Den relativen Unterschied dieses Verhältnisses zwischen den vier Ebenen kann man mit den Faktoren α_1, α_2, α_3 und α_4 abbilden.

Der lokale Zugriff bildet die unterste Ebene der Netzwerkhierarchie. Da beim lokalen Zugriff auf die Daten offensichtlich kein Netzwerk benötigt wird, wird dieser Ebene der neutrale Faktor $\alpha_1 = 1$ zugeordnet. Datentransfers innerhalb eines Clusters stellen die nächste Ebene dar. Aufgrund des schnellen und günstigen Netzwerkes verursachen diese nur einen geringen Zusatzaufwand durch den Netzwerktransfer verglichen mit einem lokalen Zugriff. Dieser zusätzliche Aufwand kann mit dem Faktor $\alpha_2 \geq \alpha_1$ erfasst werden. Wie in Abschnitt 3.1 ausführlich diskutiert, bieten die Netzwerkverbindungen zwischen Clustern innerhalb und außerhalb einer Organisation einerseits nur eine geringe Bandbreite, andererseits entstehen bei der Benutzung hohe Kosten. Den Transfers zwischen Clustern innerhalb und außerhalb einer Organisation werden entsprechend $\alpha_3 \geq \alpha_2$ beziehungsweise $\alpha_4 \geq \alpha_3$ zugeordnet.

Mithilfe der α-Faktoren können unterschiedliche Zuordnungen für einen Auftrag, auch ohne exakte Informationen zur Netzwerkgeschwindigkeit und -kosten, miteinander verglichen werden. Für das obige Beispiel eines Transfers zwischen Organisationen erhält man die Transferdauer $\hat{\tau}$

$$\begin{aligned} \tau_{tra}(d, s_g, c) &\approx \alpha \cdot (\tau_{hdd}(s_g, d) + \tau_{hdd}(s_h, d) + \tau_{hdd}(s_i, d)) \\ &= \hat{\tau}_{tra}(d, s_g, c) \end{aligned} \tag{3.25}$$

sowie die Transferkosten $\hat{\kappa}$

$$\begin{aligned} \kappa_{tra}(d, s_g, c) &\approx \alpha \cdot (\tau_{hdd}(s_g, d) \cdot k_{hdd}(s_g) \\ &\quad + \tau_{hdd}(s_h, d) \cdot k_{hdd}(s_h) \\ &\quad + \tau_{hdd}(s_i, d) \cdot k_{hdd}(s_i)) \\ &= \hat{\kappa}_{tra}(d, s_g, c) \end{aligned} \tag{3.26}$$

und damit die Schedulingfunktion

$$\phi_4(z_{org}) = \beta_1 \cdot \psi_t \left(\hat{\tau}_{tra}(d, s_g, c) \cdot \hat{\kappa}_{tra}(d, s_g, c)\right) + \beta_2 \cdot \psi_c \left(k_{cpu}(c)/g_{cpu}(c)^2\right) \tag{3.27}$$

Die α-Faktoren dienen nicht nur dazu die fehlenden Netzwerkinformationen zu kompensieren, sondern können auch dafür verwendet werden, die knappen Netzwerk- und Speicherressourcen zu schonen, indem Netzwerktransfers über große Distanzen, die viele Ressourcen benötigen, mit zusätzlichem Aufwand belegt werden. Hierzu kann einfach der Abstand zwischen den α-Faktoren erhöht werden.

3.2.3 Mehrere Datensätze

In der bisherigen Betrachtung wurde stets nur ein Datensatz berücksichtigt. Im Folgenden wird der allgemeine Fall eines Auftrages mit m Datensätzen $d_1, d_2, ..., d_m$ betrachtet. Wie man aus der Funktion für die Dauer 3.3 erkennen kann, fließt nur die maximale Transferdauer der m Transfers in die Gesamtdauer eines Auftrags mit ein. Bei den Transferkosten werden hingegen, wie in Funktion 3.8 definiert, alle entstehenden Kosten aufsummiert.

Unter Berücksichtigung dieser Definitionen ergibt sich der Transferterm t einer Zuordnung zu

$$t(c, s_1, \ldots, s_m) = \max_{1 \leq i \leq m} \{\hat{\tau}_{tra}(d_i, s_i, c)\} \cdot \sum_{i=1}^{m} \hat{\kappa}_{tra}(d_i, s_i, c) \tag{3.28}$$

mit der Transferdauer 3.25 und den Transferkosten 3.26, welche die Grundlage der zuvor hergeleiteten Schedulingfunktion 3.27 darstellen. Der Berechnungsterm b bleibt unverändert:

$$b(c) = k_{cpu}(c)/g_{cpu}(c)^2 \tag{3.29}$$

Auf Basis dieses Transferterms ergibt sich die Schedulingfunktion

$$\phi(z) = \beta_1 \cdot \psi_t \left(\max_{1 \leq i \leq m} \{\hat{\tau}_{tra}(d_i, s_i, c)\} \cdot \sum_{i=1}^{m} \hat{\kappa}_{tra}(d_i, s_i, c) \right) + \beta_2 \cdot \psi_c \left(k_{cpu}(c)/g_{cpu}(c)^2 \right) \tag{3.30}$$

welche dem nachfolgend beschriebenen Schedulingalgorithmus zugrunde liegt.

3.3 Der entwickelte Schedulingalgorithmus

Der in dieser Arbeit entwickelte Schedulingalgorithmus führt die Erkenntnisse aktueller Grid-Scheduler, Verteilter Datenbanksystem sowie MapReduce-Frameworks zu einem praxistauglichen Scheduler für daten- und rechenintensive Anwendungen in globalen, organisationsübergreifenden Umgebungen zusammen. Wie in den vorherigen Abschnitten bereits mehrfach erwähnt, ist ein zentraler Entwicklungsaspekt hierbei, nur die Informationen für das Scheduling zu verwenden, die sich auch in der Praxis zuverlässig und genau ermitteln lassen. Des Weiteren werden die hierarchische Netzwerkstruktur und die Eigenschaften der Zielfunktion verwendet, um die eintreffenden Aufträge so auf die Ressourcen zuzuordnen, so dass der benötigte Verarbeitungsaufwand über alle Aufträge möglichst gering ist.

p_{cpus}	Anzahl der von Programm p benötigten CPUs
$\|d\|$	Datengröße des Datensatzes d
$\overline{s}$	die mittlere Anzahl an Speicherressourcen für einen Datensatz
$cpus(r)$	Anzahl der Prozessorkerne von r
$clu(r)$	Eindeutiger Clusterbezeichnung von r
$org(r)$	Eindeutiger Organisationsbezeichnung von r
$u_{cpu}(r)$	Anzahl der belegten Prozessorkerne von r
$u_{hdd}(r)$	Anzahl der Speicherzugriffe auf r

Tabelle 3.3 fasst die in den nachfolgend beschriebenen Funktionen und im DOHS-Algorithmus zusätzlich zu Tabelle 3.1.2 verwendeten Definitionen zusammen.

Die zuvor hergeleitete Schedulingfunktion 3.30 bildet die Grundlage des entwickelten Schedulingalgorithmus. Die Eingabe des Schedulers umfasst alle noch nicht ausgeführten Aufträge A sowie die Parameter der Schedulingfunktion $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ und β_1, β_2. Darüber hinaus benötigt der Scheduler noch die Informationen über die verfügbaren Rechen- und Speicherressourcen R sowie die Positionsinformationen der Datensätze. Die Zuordnung eines Auftrages durch den in Abbildung 1 skizzierten Algorithmus lässt sich dabei in vier Teile gliedern. In einem ersten Schritt werden alle Rechenressourcen ausgewählt, die zur Ausführung des Auftrags infrage kommen. Anstatt einfach alle Rechenressourcen des Systems in das Scheduling mit einzubeziehen, kann durch Ausnutzung der Eigenschaften der Schedulingfunktion ein Großteil der Rechenressourcen ausgeschlossen werden. So müssen, wie nachfolgend detailliert erläutert, nur die besten Rechenressourcen jedes Clusters sowie alle Ressourcen, die einen zu verarbeitenden Datensatz speichern, betrachtet werden. Für diese möglichen Ausführungskandidaten werden im zweiten Schritt alle Speicherressourcen für die zu verarbeitenden Datensätze bestimmt, welche einen möglichst geringen Transferaufwand in Bezug auf die Schedulingfunktion ergeben. Auf Basis des Transfer- und Berechnungsaufwandes wird im dritten Schritt der Rang aller Kandidaten bestimmt. Die Zuordnung mit dem höchsten Rang wird für die Ausführung des Auftrags gewählt und reserviert. Zuletzt wird die Auslastung der verwendeten Ressourcen angepasst und der Scheduler bearbeitet den nächsten Auftrag.

Der entwickelte Algorithmus verwendet zur Bestimmung der Ausführungskandidaten, des Transfer- und Berechnungsaufwandes sowie der Ränge, die Schedulingfunktion $\phi(z)$ 3.30. Die Schedulingfunktion definiert jedoch nicht, wie sich diese Terme mit den real verfügbaren Ressourceninformationen bestimmen lassen. Für die beiden zentralen Komponenten der Funktion, den Transferterm 3.28 und den Berechnungsterm 3.29, müssen daher geeignete Funktionen definiert werden. Darüber hinaus muss auch ein geeignetes Verfahren zur Bestimmung der Speicherressourcen für die m Datensätze eines Auftrags gewählt werden, da für die Minimierung

Algorithmus 1: Pseudo-Code des entwickelten Schedulers

Eingabe : Noch nicht ausgeführte Aufträge A, Parameter $\alpha_1, \alpha_2, \alpha_3, \alpha_4, \beta_1, \beta_2$
Ausgabe : Zuordnungen Z

foreach $a = (p, d_1, ..., d_m) \in A$ **do**

`// Bestimme die Kandidaten` C `zur Ausführung von` p
C = Die effizientesten Rechenressourcen jedes Clusters
und alle Rechenressourcen die einen der Datensätze speichern

`// Bestimme für jeden Kandidaten die beste Zuordnung`
foreach $c \in C$ **do**
Bestimme $s_1, ..., s_m$ mit möglichst geringem Transferaufwand
unter Verwendung der Parameter $\alpha_1, \alpha_2, \alpha_3, \alpha_4$
T_c = Transferaufwand von $s_1, ..., s_m$ nach c
b_c = Berechnungsaufwand von p auf c
$z_c = (c, s_1, ..., s_m)$
end

Bestimme den Rang rg jedes Kandidaten $c \in C$ mit T_c und b_c sowie β_1, β_2

Wähle Zuordnung $z^* = (c^*, s_1^*, ..., s_m^*)$ mit höchstem Rang

Passe die Auslastung aller Komponenten entsprechend an

end

des Transferaufwandes ansonsten alle möglichen Kombinationen $R_{d_1} \cdot R_{d_2} \cdot \ldots \cdot R_{d_m}$ betrachtet werden müssten. Zuletzt müssen auch noch geeignete Transformationsfunktionen ψ_t und ψ_c für die Berechnung der Ränge gewählt werden.

3.3.1 Der Berechnungsaufwand

Der in der Schedulingfunktion verwendete Berechnungsterm $b = k_{cpu}(c)/g_{cpu}(c)^2$ (3.29) basiert auf den Kosten und der Prozessorgeschwindigkeit der Rechenressource c. Sowohl die Kosten als auch die Prozessorgeschwindigkeit lassen sich einfach bestimmen, so dass nicht sofort ersichtlich ist, welche Problematik bei der Bestimmung des Berechnungsterms vorliegt. Betrachtet man jedoch eine Umgebung mit belegten und freien Rechenressourcen, stellt sich die Frage, wie die belegten Ressourcen in das Scheduling integriert werden sollen. Bei den bestehenden Grid-Schedulern existiert diese Problematik nicht, da angenommen wird, dass die Ausführungsdauer der Aufträge bekannt ist und somit die Ausführungsdauer eines Auftrages sowohl für belegte als auch für freie Rechenressourcen einfach bestimmt werden kann. Dem entwickelten Scheduler ist die Ausführungsdauer, wie bereits erläutert, hingegen nicht bekannt. Eine Möglichkeit den Berechnungsaufwand ohne die Ausführungsdauer zu bestimmen, ist, nur die freien Rechenressourcen in das Scheduling mit einzubeziehen. Dadurch kann der Berechnungsterm unverändert übernommen werden.
Diese Vorgehensweise hat jedoch erhebliche Nachteile: Sind etwa alle Ressourcen

belegt und wird eine Ressource frei, erhält diese immer den nächsten Auftrag. Ein Scheduling findet in stark genutzten Umgebungen mit dieser Vorgehensweise de facto nicht mehr statt. Sollen daher auch belegte Rechenressourcen in das Scheduling mit einbezogen werden, bleibt die Frage, welche Prozessorgeschwindigkeit belegte Rechenressourcen besitzen.

Die im Algorithmus 2 dargestellte Funktion $\delta_{cpu}(p, c)$ zur Bestimmung des Berechnungsaufwands eines Programmes p auf der Rechenressource c zeigt den in dieser Arbeit entwickelten Ansatz, die Prozessorgeschwindigkeit einer belegten Rechenressource zu definieren. Die Funktion prüft zunächst, ob die Rechenressource über-

Algorithmus 2: Der Berechnungsaufwand $\delta_{cpu}(p, c)$

Eingabe : $p \in P$, $c \in R$
Ausgabe : Berechnungsaufwand

`// Überprüfe die Anforderungen von` p
1 **if** *c erfüllt nicht alle Anforderungen von p* **then**
2 | **return** ∞
3 **end**

`// Bestimme die langsamste frei Rechenressource`
`// Eine belegte Rechenressource ist immer langsamer`
`// als die langsamste freie Rechenressource`
4 $g_{cpu}^{min} = \min\{\, g_{cpu}(r) \mid r \in R \wedge cpus(r) > u_{cpu}(r) + p_{cpus} \,\}$
5 $g = 0$

`// Genügend freie CPUs`
6 **if** $cpus(c) > u_{cpu}(c) + p_{cpus}$ **then**
7 | $g = g_{cpu}(c)$
8 `// Keine freie CPU, aber es gibt Ressource mit freier CPU`
9 **else if** $u_{cpu}(c) \geq cpus(c) \ \wedge \ \exists\, r \in R$ *mit* $cpus(r) > u_{cpu}(r) + p_{cpus}$ **then**
10 | $g = g_{cpu}^{min} \cdot cpus(c)/(cpus(c) + u_{cpu}(c))$
11 `// Keine freie CPU im gesamten System`
12 **else**
13 | $g = g_{cpu}(c) \cdot cpus(c)/(cpus(c) + u_{cpu}(c))$
14 **end**
15 **return** $k_{cpu}(c)/g^2$

haupt alle Anforderungen - Prozessortyp, Betriebssystem, Hauptspeicher,... - des Programmes erfüllt (Zeile 1). Ist dies der Fall, wird die Geschwindigkeit der Rechenressource abhängig von der Auslastung des Gesamtsystems bestimmt. Verfügt die Ressource über genügend freie Prozessorkerne - die Anzahl der Prozessorkerne der Ressource ist größer als die Anzahl der belegten und angeforderten Prozessorkerne $cpus(c) > u_{cpu}(c) + p_{cpus}$ - wird die Prozessorgeschwindigkeit der Ressource

direkt verwendet (Zeilen 6 bis 7).
Sind andererseits alle Prozessorkerne der Ressource belegt und es existieren auch keine anderen freien Ressourcen im System, ergibt sich die Geschwindigkeit der Ressource zu $g_{cpu}(c) \cdot cpus(c)/(cpus(c) + u_{cpu}(c))$ (Zeilen 8 bis 9). Dieser Term bewirkt, dass die Geschwindigkeit einer voll belegten Ressource $cpus(c) = u_{cpu}(c)$ auf die Hälfte der eigentlichen Geschwindigkeit abfällt. Mit zunehmender Belegung sinkt die Geschwindigkeit der Ressource immer weiter. Durch die Berücksichtigung der Prozessoranzahl sinkt hierbei die Geschwindigkeit von Ressourcen mit vielen Prozessorkernen bei steigender Nutzung langsamer als die Geschwindigkeit von Ressourcen mit wenigen Prozessorkernen. Dies spiegelt die Wahrscheinlichkeit wieder, dass ein wartender Auftrag auf einer Rechenressource mit vielen Prozessorkernen schneller einen Prozessor zugeteilt bekommt als auf einer Ressource mit wenigen Prozessorkernen.
Für den Fall, dass die Ressource keine freien Prozessorkerne besitzt, aber im System noch mindestens eine geeignete, freie Rechenressource zur Verfügung steht, wird die Geschwindigkeit der Ressource fast identisch ermittelt. Anstatt der Geschwindigkeit der Ressource $g_{cpu}(c)$ wird jedoch die Geschwindigkeit der *langsamsten, freien* Rechenressource g_{cpu}^{min} verwendet (Zeilen 12 bis 13). Hiermit wird sichergestellt, dass der Berechnungsaufwand auf einer belegten Ressource immer höher ist als auf einer freien Ressource. Dadurch werden einerseits eine Überlastung schneller Ressourcen mit vielen Prozessoren und andererseits das Brachliegen langsamer Ressourcen vermieden.
Der von der Funktion $\delta_{cpu}(p, c)$ zurückgegebene Berechnungsaufwand ergibt sich aus der so ermittelten Prozessorgeschwindigkeit g und den Kosten der Rechenressource $k_{cpu}(c)$ (Zeile 15).

3.3.2 Der Transferaufwand

Zusammen mit dem Berechnungsterm bildet der Transferterm 3.28 die zentralen Komponenten der Schedulingfunktion 3.30. Der Transferterm beschreibt dabei den Transferaufwand einer vorgegebenen Zuordnung. Im DOHS-Algorithmus wird für alle möglichen Ausführungskandidaten eines Auftrags - die mithilfe des Berechnungsaufwandes bestimmt werden - eine Abbildung der zu verarbeitenden Datensätze auf die Speicherressourcen gesucht. Diese Abbildung soll einen möglichst minimalen Transferterm erzeugen. Genauer, besteht die Aufgabe der Transferaufwandsfunktion beim Scheduling eines Auftrags $a = (p, d_1, ..., d_m)$ darin, für einen möglichen Ausführungskandidaten c eine Zuordnung für alle m Datensätze zu den zur Auswahl stehenden Speicherressourcen $R_{d_1}, R_{d_2}, \ldots, R_{d_m}$

$$Z_{a,c} \;=\; \{\; (c, s_1, ..., s_m) \mid s_1 \in R_{d_1} \wedge s_2 \in R_{d_2} \wedge \ldots \wedge s_m \in R_{d_m} \;\}$$

mit möglichst geringem Transferterm

$$t(c, s_1, \ldots, s_m) = \max_{1 \leq i \leq m} \{\hat{\tau}_{tra}(d_i, s_i, c)\} \cdot \sum_{i=1}^{m} \hat{\kappa}_{tra}(d_i, s_i, c)$$

zu bestimmen:

$$\text{Finde } z^* \in Z_{a,c} \text{ mit } t(z^*) = \min_{z \in Z_{a,c}} \{t(z)\}$$

Das Problem, eine Zuordnung mit minimalem Transferaufwand zu finden, ähnelt dem „Unsplittable Flow Problem“ [CCGK02] welches NP-Vollständig ist. Zur Ermittlung einer Zuordnung mit minimalem Transferaufwand müssen daher alle $R_{d_1} \cdot R_{d_2} \cdot \ldots \cdot R_{d_m}$ möglichen Kombinationen betrachtet werden.
Dies lässt sich leicht an folgendem Beispiel eines Auftrags mit 2 Datensätzen zeigen: Für den Datensatz d_1 stehen die Speicherressourcen s_1 und s_2 mit den entsprechenden Aufwänden 9 und 10 zur Auswahl. Der Datensatz d_2 ist auf den Ressourcen s_1 und s_3 verfügbar, wobei die Aufwände 10 und 15 betragen. Wird nun zuerst der Datensatz d_1 betrachtet, wird diesem die Speicherressource s_1 zugeordnet. Dadurch steigt jedoch die Auslastung von s_1, so dass sich der Aufwand von d_2 auf s_1 auf 20 verdoppelt. Für d_2 wird daher s_3 gewählt. Der Gesamtaufwand der Zuordnung $(d_1, d_2) \rightarrow (s_1, s_3)$ beträgt damit 24, wohingegen die Zuordnung $(d_1, d_2) \rightarrow (s_2, s_1)$ nur einen Aufwand von 20 erzeugen würde.
In realen Umgebungen, wo auch $m > 20$ vorkommen kann, ist die Betrachtung aller Kombinationen offensichtlich nicht in angemessener Zeit durchführbar. Doch selbst wenn alle möglichen Kombinationen für einen Auftrag betrachtet werden, garantiert dies keinen minimalen Transferaufwand, da es zu ähnlichen Überschneidungen mit nachfolgenden Aufträgen kommen kann. Aus Sicht des Schedulers, dessen Ziel die Minimierung der Zielfunktion über alle Aufträge ist, sind daher Heuristiken zur Minimierung des Transferaufwandes völlig ausreichend.

Zur Bestimmung des Transferaufwandes ist die Transferdauer $\hat{\tau}_{tra}$, und hier insbesondere die Speicherzugriffsdauer τ_{hdd} von entscheidender Bedeutung. Wie in Abschnitt 3.1.1 bereits erwähnt, teilen sich alle Zugriffe auf eine Speicherressource die verfügbare Speicherzugriffsgeschwindigkeit gleichmäßig. Nimmt man einen diskreten Zeitverlauf an - etwa eine Taktung von einer Sekunde - kann die Auslastung einer Speicherressource r und damit die Speicherzugriffsdauer $\tau_{hdd}(r, d) = x_1 - x_0$ für einen Datensatz d mit dem Transferstart x_0 durch folgende Gleichung ermittelt werden

$$|d| = \sum_{x=x_0}^{x_1} g_{hdd}(r)/u_{hdd}(r, x)$$

wobei $|d|$ die Datengröße von d, $g_{hdd}(r)$ die Speichergeschwindigkeit und $u_{hdd}(r, x)$ die Auslastung zum Zeitpunkt x beschreiben. Offensichtlich benötigt die Berechnung der Speicherzugriffsdauer Wissen über die zukünftige Auslastung der Speicherressource. Obwohl die Auslastungen der Speicherressourcen, im Gegensatz zum Netzwerk, fast ausschließlich vom Scheduler gesteuert werden, kann nicht ausgeschlossen werden, dass weitere unabhängige Zugriffe erfolgen oder nachfolgende Aufträge diese Speicherressource benötigen. Doch selbst wenn alle Informationen zur zukünftigen Auslastung bekannt wären, müssen bei der Ermittlung der Auslastung einer Speicherressource alle anderen Transfers, bei denen die Speicherressource als Zwischenstation auftreten kann, berücksichtigt werden. Der dadurch entstehende Aufwand kann bei großen Umgebungen mit vielen Aufträgen das Scheduling sehr aufwendig

Algorithmus 3: Die Transferaufwand $\delta_{tra}(c, d_1, \ldots, d_m)$

Eingabe : $c \in R$ und $d_1, \ldots, d_m \in D, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \beta_1, \beta_2$
Ausgabe : Speicherzuordnung, Transferaufwand

$t_{max} = 0\,, \quad K = 0\,, \quad S = \emptyset$
$\forall\, r \in R:\ \hat{u}_{hdd}(r) = 0$
foreach $d \in \{d_1, \ldots, d_m\}$ **do**
  $t^* = \infty\,, \quad k^* = \infty\,, \quad e^* = \infty\,, \quad s^* = \emptyset, \quad u_{tmp} = \emptyset$
  foreach $s \in R_d$ **do**
    $\alpha = 0$
    if $s == c$ **then** $\alpha = \alpha_1$
    else if $clu(s) == clu(c)$ **then** $\alpha = \alpha_2$
    else if $org(s) == org(c)$ **then** $\alpha = \alpha_3$
    else $\alpha = \alpha_4$
    // $(s, s_1), \ldots, (s_l, c) \in N$ `sei Pfad zwischen` $s_0 = s$ `und` c
    $t = \alpha \cdot |d| \cdot \sum_{i=0}^{l} (u_{hdd}(s_i) + \hat{u}_{hdd}(s_i) + 1) \,/\, g_{hdd}(s_i)$
    $k = \alpha \cdot |d| \cdot \sum_{i=0}^{l} k_{hdd}(s_i) \cdot (u_{hdd}(s_i) + \hat{u}_{hdd}(s_i) + 1) \,/\, g_{hdd}(s_i)$
    if $(t \cdot k) < e^*$ **then**
      $e^* = (t \cdot k)$
      $t^* = t$
      $k^* = k$
      $s^* = s$
      $u_{tmp} = s, s_1, \ldots, s_l$
    end
  end
  if $t^* > t_{max}$ **then**
    $t_{max} = t^*$
  end
  $K = K + k^*$
  $S = S \cup \{s^*\}$
  $\forall\, r \in u_{tmp}:\ \hat{u}_{hdd}(r) = \hat{u}_{hdd}(r) + 1$
end
return $(S,\ (\, t_{max} \cdot K\,)\,)$

gestalten. Daher wird im entwickelten Scheduler als Schätzer für die Speicherzugriffsdauer die aktuelle Auslastung $u_{hdd}(r)$ der Ressource verwendet, so dass sich die im Schedulingalgorithmus verwendete Zugriffsdauer zu

$$|d| \cdot u_{hdd}(r)/g_{hdd}(r)$$

ergibt.

Die in Algorithmus 3 dargestellte Funktion $\delta_{tra}(c, d_1, \ldots, d_m)$ ermittelt auf Basis dieser Überlegungen eine Zuordnung mit möglichst geringem Transferaufwand. Anstatt alle möglichen Zuordnungen zu berücksichtigen, wird für jeden einzelnen Datensatz nacheinander die Speicherressource mit minimalem Transferaufwand zur Rechenressource c bestimmt. Diese Heuristik benötigt somit nur eine Iteration über die Datensätze sowie über die entsprechenden Speicherressourcen - es werden nur $R_{d_1} + R_{d_2} + \ldots + R_{d_m}$ mögliche Zuordnungen betrachtet (Zeilen 3 bis 20). Für jede dieser Zuordnungen wird mithilfe der zuvor definierten Speicherzugriffsdauer der Aufwand für den Transfer des Datensatzes von der vorgegebenen Speicherressource s zur Rechenressource c bestimmt. Die Transferdauer ergibt sich, wie in Gleichung 3.25 definiert, aus der Summe der Speicherzugriffsdauern über alle beim Transfer beteiligten Speicherressourcen. Diese Summe wird noch mit dem entsprechenden Distanzfaktor α multipliziert, um die fehlenden Netzwerkinformationen auszugleichen und Weitverkehrstransfers zu vermeiden (Zeile 11). Die Transferkosten werden nach Gleichung 3.26 aus der Summe des Produkts aus Speicherzugriffsdauer und Speicherzugriffskosten berechnet (Zeile 12). Der Transferaufwand für den Transfer eines Datensatzes ergibt sich schließlich aus der Multiplikation der Transferdauer mit den Transferkosten. Aus allen möglichen Speicherressourcen für einen Datensatz wird diejenige mit dem niedrigsten Aufwand gewählt (Zeile 13). Der Transferaufwand, die Transferdauer, die Transferkosten sowie die beteiligten Zwischenstation werden zwischengespeichert (Zeilen 14 bis 19). Die so ermittelte beste Speicherressource für diesen Datensatz wird der Ergebniszuordnung S hinzugefügt (Zeile 25). Die benötigten Transferkosten werden zu den Gesamtkosten addiert - dies entspricht $\sum_{i=1}^{m} \hat{\kappa}_{tra}(d_i, s_i, c)$ der Schedulingfunktion 3.30 (Zeile 24). Zusätzlich wird überprüft, ob es sich um die bislang längste Transferdauer handelt - entsprechend $\max_{1 \leq i \leq m}\{\hat{\tau}_{tra}(d_i, s_i, c)\}$ (Zeile 21). Des Weiteren werden die beim Transfer entstehenden zusätzlichen Speicherzugriffe für die Betrachtung der nachfolgenden Datensätze gespeichert (Zeile 26). Nachdem für alle Datensätze auf diese Weise die Speicherressourcen ausgewählt wurden, wird die Gesamtzuordnung sowie der Gesamtaufwand - wie in der Schedulingfunktion 3.30 definiert, die maximale Transferdauer multipliziert mit den Gesamtkosten - zurückgegeben (Zeile 28).

3.3.3 Der DOHS-Algorithmus

Mit den Funktionen zur Bestimmung des Berechnungsaufwandes $\delta_{cpu}(p, c)$ sowie des Transferaufwandes $\delta_{tra}(c, d_1, \ldots, d_m)$ lässt sich der in Algorithmus 1 skizzierte Schedulingalgorithmus genau definieren. Algorithmus 4 zeigt die Details des in dieser Arbeit entwickelten Schedulingalgorithmus für einen Auftrag mit beliebig vielen

Datensätzen. Der Algorithmus teilt sich in vier Abschnitte: (1) Ermittlung der Ausführungskandidaten; (2) Bestimmung des Transferaufwandes für alle Kandidaten; (3) Berechnung des minimalen Transfer- und Berechnungsaufwandes; (4) Ermittlung der Ränge für alle Kandidaten und Auswahl der besten Zuordnung.

Da die Betrachtung aller Rechenressourcen in realen Umgebungen mit Tausenden Ressourcen sehr aufwendig wäre, werden im ersten Schritt die Eigenschaften der Schedulingfunktion ausgenutzt, um die Anzahl der vom Scheduler zu untersuchenden Rechenressourcen so gering wie möglich zu halten. Zur Bestimmung der Ausführungskandidaten C, teilt man zunächst alle Rechenressourcen in zwei Kategorien auf. Die erste Kategorie enthält alle Ressourcen, die keinen der zu verarbeitenden Datensätze speichern, so dass alle Datensätze vor einer Ausführung auf diese Ressource transferiert werden müssen. Betrachtet man die Schedulingfunktion für diese Fälle, stellt man fest, dass der Transferaufwand - aufgrund der Vernachlässigung der Netzwerkverbindungen - für alle Ressourcen innerhalb eines Clusters identisch ist. Für das Scheduling muss daher aus der ersten Kategorie nur die beste Rechenressource jedes Clusters berücksichtigt werden. Heutzutage bestehen Cluster aus mehreren Hundert Rechenressourcen, so dass sich die Anzahl der vom Scheduling zu betrachtenden Rechenressourcen in diesen Umgebungen um mindestens einen Faktor 100 verringert.
Die zweite Kategorie enthält alle Rechenressourcen, die gleichzeitig auch Speicherressourcen sind und einen der zu verarbeitenden Datensätze vorhalten. Alle Ressourcen der zweiten Kategorie, die zusätzlich alle Anforderungen des Programmes erfüllen, sind offensichtlich auch geeignete Ausführungskandidaten, da sie im Unterschied zu den anderen Ressourcen im Cluster durch die auf der Ressource verfügbaren Datensätze einen spezifischen Transferaufwand aufweisen.

Nachdem alle möglichen Ausführungskandidaten bestimmt sind, kann im nächsten Schritt eine Zuordnung der Datensätze zu den Speicherressourcen und der daraus resultierende Transferaufwand für diese Rechenressourcen mit der Funktion $\delta_{tra}(c, d_1, \ldots, d_m)$ ermittelt werden. Die anschließende Berechnung des minimalen Transfer- und Berechnungsaufwandes (T_{min}, b_{min}) wird für die in der Schedulingfunktion erforderliche Normierung benötigt. Die Normierungsfunktionen ψ_t und ψ_c ergeben sich damit zu

$$\psi_t(\delta_{tra}(c, d_1, \ldots, d_m)) = T_{min}/\delta_{tra}(c, d_1, \ldots, d_m)$$

beziehungsweise

$$\psi_c(\delta_{cpu}(p, c)) = b_{min}/\delta_{cpu}(p, c)$$

mit den Wertebereichen $[1, 0)$. Im letzten Schritt werden die Ränge aller Kandidaten mit den Parametern β_1 und β_2 bestimmt und die Zuordnung mit dem höchsten Rang zurückgegeben.

Der DOHS-Algorithmus betrachtet für die Bestimmung der Ränge möglichst wenige Zuordnung, so dass nur für die besten Rechenressourcen eines Clusters sowie die Ressourcen die auch einen Datensatz speichern, der Transferaufwand bestimmt werden muss. Für einen Auftrag mit m Datensätzen lässt sich die Anzahl der vom

Algorithmus 4: Scheduling eines Auftrages mit mehreren Datensätzen

Eingabe : $p \in P$, $d_1, \ldots, d_m \in D$, $\alpha_1, \alpha_2, \alpha_3, \alpha_4$, β_1, β_2
Ausgabe : Zuordnung mit höchstem Rang z^*

`// Bestimme die möglichen Ausführungskanditaten`
$C = \emptyset$
`// Die beste Rechenressource jedes Clusters`
C_{best} = beste Rechenressourcen aller n_c Cluster
`// Ressource speichert einen Datensatz`
$C_{data} = \{ c \mid (c \in R_{d_1} \vee \ldots \vee c \in R_{d_m}) \wedge \delta_{cpu}(p, c) < \infty \}$
$C = C_{best} \cup C_{data}$
`// Bestimme Transferaufwand für alle Kandidaten`
$Z = \emptyset$
foreach $c \in C$ **do**
  $(S, T) = \delta_{tra}(\, c, d_1, \ldots, d_m \,)$
  $Z = Z \cup \{(\, c,\ S,\ T,\ \delta_{cpu}(p, c) \,)\}$
end
`// Bestimme minimalen Transfer- und Berechnungsaufwand`
$T_{min} = \min\{\, T \mid (c, S, T, b) \in Z \}$
$b_{min} = \min\{\, b \mid (c, S, T, b) \in Z \}$
`// Bestimme Zuordnung mit höchstem Rang`
$rg^* = 0$
$z^* = \emptyset$
foreach $(c, S, T, b) \in Z$ **do**
  $rg = \beta_1 \cdot \frac{T_{min}}{T} + \beta_2 \cdot \frac{b_{min}}{b}$
  if $rg > rg^*$ **then**
    $rg^* = rg$
    $z^* = (c, S, T, b)$
  end
end
return (z^*)

Scheduler zu betrachtenden Zuordnungen durch $(n_c+\overline{s})\cdot(R_{d_1}+\ldots+R_{d_m})$ abschätzen, wobei n_c die Anzahl der Cluster und $\overline{s}$ die mittlere Anzahl an Speicherressourcen für einen Datensatz bezeichnen.
Für Aufträge, die nur einen Datensatz verarbeiten sollen, lässt sich der Schedulingaufwand noch deutlich senken. Dies ist insbesondere für MapReduce und ähnlichen Anwendungen interessant, da diese eine große Anzahl an Aufträgen mit einem zu verarbeitenden Datensatz erzeugen. Der angepasste DOHS-Algorithmus 5 muss für Aufträge mit einem Datensatz nur aus $4 \cdot R_d$ Zuordnungen wählen. Denn aus Sicht einer Speicherressource, die einen der Datensätze speichert, gibt es nur 3 weitere mögliche Ausführungsmaschinen, die aufgrund der $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ infrage kommen. Dies ist die beste Rechenressource im gleichen Cluster, in der gleichen Organisation sowie außerhalb der Organisation. Alle anderen Rechenressourcen mit der gleichen Distanz können ausgeschlossen werden, da sie den gleichen Transferaufwand wie die besten Rechenressourcen auf der gleichen Ebene benötigen, aber einen höheren Berechnungsaufwand δ_{cpu} erfordern.

Algorithmus 5: Scheduling eines Auftrages mit einem Datensatz

Eingabe : $p \in P$, $d \in D$, $\alpha_1, \alpha_2, \alpha_3, \alpha_4$, β_1, β_2
Ausgabe : Zuordnung mit höchstem Rang z^*

`// Die besten Rechenressourcen`
C = beste Rechenressourcen aller n_c Cluster
O = beste Rechenressourcen aller n_o Organisationen
q = die beste Rechenressource
$Z = \emptyset$
foreach $s \in R_d$ **do**
  $Z = Z \cup \{ (s,\ s,\ \delta_{tra}(s,s,d),\ \delta_{cpu}(p,s)),$
    $(s,\ C[clu(s)],\ \delta_{tra}(s,C[clu(s)],d),\ \delta_{cpu}(p,C[clu(s)])),$
    $(s,\ O[org(s)],\ \delta_{tra}(s,O[org(s)],d),\ \delta_{cpu}(p,O[org(s)])),$
    $(s,\ q,\ \delta_{tra}(s,q,d),\ \delta_{cpu}(q,b)) \}$
end
$t_{min} = \min\{ t \mid (c,s,t,b) \in Z \}$
$b_{min} = \min\{ b \mid (c,s,t,b) \in Z \}$
$rg^* = 0$, $z^* = \emptyset$
foreach $(c,s,t,b) \in Z$ **do**
  $rg = \beta_1 \cdot \frac{t_{min}}{t} + \beta_2 \cdot \frac{b_{min}}{b}$
  if $rg > rg^*$ **then**
    $rg^* = rg$
    $z^* = (c,s,t,b)$
  end
end
return (z^*)

Wie aus den Funktionsdefinitionen δ_{cpu} und δ_{tra} sowie aus der Beschreibung des DOHS-Algorithmus hervorgeht, benötigt der in dieser Arbeit entwickelte Schedulingalgorithmus nur die Auslastung, die Geschwindigkeit sowie die Kosten der Speicher- und Rechenressourcen. Die in realen Umgebungen kaum zu ermittelnde Netzwerkgeschwindigkeit und Ausführungsdauer, die von den bestehenden Schedulern benötigt werden, werden durch die Berücksichtigung der hierarchischen Netzwerkstruktur in Form der Distanzparameter $\alpha_1, \alpha_2, \alpha_3$ *und* α_4 sowie die Anpassung der Prozessorgeschwindigkeit von belegten Rechenressourcen kompensiert. Die Distanzfaktoren dienen darüber hinaus auch dazu die knappen Speicher- und Netzwerkressourcen zu schonen, die für die Übertragung zwischen Clustern oder Organisationen benötigt werden. Diese freien Ressourcen können so von nachfolgenden Aufträgen, die auf diese Ressourcen angewiesen sind, genutzt werden. Die effiziente Bestimmung der Zuordnungen für die Aufträge ist neben der Güte der erzeugten Zuordnung ein weiterer Vorteil des entwickelten Schedulers. So betrachtet der DOHS-Algorithmus für einen Auftrag mit m Datensätzen nur ungefähr $(n_c + \overline{s}) \cdot (R_{d_1} + \ldots + R_{d_m})$ mögliche Zuordnungen. Dadurch kann der Scheduler auch in Umgebungen mit Hunderten Organisationen und Clustern eingesetzt werden.

Kapitel 4

Evaluierung des DOHS

Im vorherigen Kapitel wurde der *Dual Objective Hierarchical Scheduling* Algorithmus zur effizienten Zuteilung von rechen- und datenintensiven Anwendungen in organisationsübergreifenden Umgebungen detailliert vorgestellt. In den nachfolgenden Abschnitten soll die Güte der vom DOHS-Algorithmus generierten Zuordnungen im Bezug auf die Zielfunktion φ untersucht werden. Der Vergleich unterschiedlicher Scheduler kann hierbei auf verschiedene Weise durchgeführt werden. Als Vergleichswert kann etwa die erwartete mittlere Abweichung der Zielfunktion unter verschiedenen Bedingungen oder die maximal mögliche Abweichung des Wertes der Zielfunktion vom optimalen Wert verwendet werden [Sga97].
Häufig wird für Aufträge *ohne* Datensätze die maximalen Abweichung vom optimalen Wert σ verwendet und als Vergleichswert das Vielfache des optimalen Wertes $x \cdot \sigma$ angegeben. So ergibt sich für den von Graham[Gra66] entwickelten List-Scheduler zur Minimierung der maximalen Ausführungsdauer für m identische Rechenressourcen ein maximaler Abstand von $2 - \frac{1}{m}$ vom Optimum.[1]
Zur Ermittlung der mittleren oder maximalen Abweichung vom Optimum muss die optimale Zuordnung für die ungünstigste Umgebungs- und Auftragslage bestimmbar sein. Dies ist schon für komplexe Problemstellungen von Aufträgen *ohne* Datensätze sehr aufwendig und für die in dieser Arbeit untersuchte Problemstellung mit Datensätzen kaum umsetzbar. So konnte in der Literatur keine Untersuchung auf Basis der mittleren oder maximalen Abweichung für diese oder ähnliche Problemstellungen gefunden werden. Der Vergleich unterschiedlicher Schedulingalgorithmen erfolgt stattdessen, wie auch bei den in Abschnitt 2.1.3 vorgestellten Schedulern, durch Simulationen.

Die Eignung des DOHS-Algorithmus soll anhand der in diesem Kapitel vorgestellten Simulationen sowie des im nachfolgenden Kapitel beschriebenen, produktiv genutzten Verarbeitungssystems gezeigt werden. Die entwickelte Simulationsumgebung ermöglicht eine umfassende Evaluierung des entwickelten Algorithmus in den unterschiedlichsten Ausprägungen von organisationsübergreifenden Umgebungen. Die Simulationsumgebung dient einerseits dem Vergleich mit bestehenden Schedulern,

[1] Optimale Ausführungsdauer σ: Längste vom List-Scheduler erzeugte Ausführungsdauer ist $(2 - \frac{1}{m}) \cdot \sigma$

erlaubt andererseits aber auch die Modellierung realer Umgebungen zur Anpassung der Parameter des DOHS auf deren spezielle Anforderungen.

In den nachfolgenden Abschnitten wird zunächst die entwickelte Simulationsumgebung detailliert vorgestellt. Anschließend werden die zum Vergleich mit dem DOHS-Algorithmus herangezogenen Schedulingverfahren erläutert. Das Kapitel endet mit einer Diskussion der Ergebnisse der durchgeführten Simulationen.

4.1 Die Simulationsumgebung

Die entwickelte Simulationsumgebung basiert auf dem in Abschnitt 3.1 beschriebenen Umgebungs- und Schedulingmodell. Die Rechen- und Speicherressourcen sowie die Netzwerkressourcen mit ihren entsprechenden Eigenschaften bilden die Grundlage der Simulationsumgebung. Aus diesen Ressourcentypen können beliebige Umgebungen zur Simulation aufgebaut werden. Die Eigenschaften der Rechen-, Speicher- und Netzwerkressourcen sind ebenso wie die Struktur frei wählbar. Darüber hinaus stehen unterschiedliche Netzwerkauslastungsmodelle zur Verfügung. Die Anzahl und die Eigenschaften der auszuführenden Aufträge können gleichfalls frei gewählt werden. Ein Simulationslauf führt die Aufträge, wie vom gewählten Schedulingalgorithmus vorgegeben, in einer so erzeugten Umgebungsinstanz aus.

4.1.1 Ressourcen und Aufträge

Die Simulationsumgebung erfordert eine konkrete Ausprägung des in Abschnitt 3.1 beschriebenen Umgebungsmodells, wobei insbesondere die Verfügbarkeit und Genauigkeit der Ressourceninformationen definiert werden muss. Des Weiteren wird für die Simulation eine Beschreibung der Aufträge benötigt, die dem beschriebenen Schedulingmodell genügt.

Die Ressourcen

Eine Rechenressource besteht in der Simulationsumgebung aus mindestens einem Prozessorkern. Die beiden zentralen Eigenschaften einer Rechenressource sind die Prozessorgeschwindigkeit und die Kosten. Die Kosten sowie die Prozessorgeschwindigkeit der Rechenressource sind fest und unterscheiden sich nicht für unterschiedliche Anwender oder Anwendungen. Die Verwendung eines Prozessors wird in Cent pro Sekunde abgerechnet und die Geschwindigkeit in Operationen pro Sekunde. Ein Prozessorkern kann nur von einem Programm genutzt werden. Eine geteilte Nutzung ist nicht zulässig. In den durchgeführten Simulationen sind die weiteren Eigenschaften einer Rechenressource wie Prozessortyp, Hauptspeicher und Betriebssystem identisch.

Die Geschwindigkeit und die Kosten sind auch die entscheidenden Eigenschaften

einer Speicherressource. Die Geschwindigkeit[2] wird in MB (MegaByte) pro Sekunde angegeben und die Lese- und Schreibgeschwindigkeit sind identisch. Die Kostenabrechnung erfolgt wie bei den Rechenressourcen in Cent pro Sekunde. Alle parallelen Zugriffe auf eine Speicherressource teilen sich die zur Verfügung stehende Bandbreite gleichmäßig, so dass bei drei gleichzeitigen Zugriffen jedem Klient ein Drittel der Bandbreite zur Verfügung steht. Es wird angenommen, dass die Speicherressourcen eine unendliche Kapazität besitzen, so dass immer genügend Kapazität für eine Zwischenspeicherung zur Verfügung steht.

Wie bei den Rechen- und Speicherressourcen sind die Geschwindigkeit und die Kosten die entscheidenden Eigenschaften einer Netzwerkressourcen, wobei die Geschwindigkeit in MB pro Sekunde und die Kosten in Cent pro Sekunde angegeben werden. Die Bandbreite einer Verbindung kann asymmetrisch sein, so dass bei der Geschwindigkeits- und der Kostenbetrachtung die Transferrichtung beachtet werden muss. Alle Transfers in eine Richtung teilen sich, wie bei den Speicherressourcen, die zur Verfügung stehende Bandbreite gleichmäßig.

Alle Rechen- und Speicherressourcen im System sind eindeutig einem Cluster zugeordnet. Die Cluster werden wiederum eindeutig Organisationen zugeordnet. Innerhalb eines Clusters können alle Ressourcen direkt miteinander kommunizieren. Für die Kommunikation mit Ressourcen außerhalb des Clusters muss jedoch zwingend ein Gateway verwendet werden. In jedem Cluster ist hierfür genau eine Speicherressource als Gateway ausgewiesen. Die Gateways aller Cluster können direkt - über Organisationsgrenzen hinweg - miteinander kommunizieren.

Rechen- und Speicherressourceninformationen

Sowohl die Kosten als auch die Geschwindigkeit der Rechen- und Speicherressourcen stehen den Schedulern exakt zur Verfügung. Darüber hinaus wird angenommen, dass die Rechen- und Speicherressourcen ausschließlich vom Scheduler verwaltet werden, so dass die Auslastung der Ressourcen bekannt ist. Diese Annahme dient der Vereinfachung der Auswertung, stellt jedoch keine Beschränkung der Allgemeinheit dar, da eine externe Nutzung einfach festgestellt werden könnte und in die Auslastungsinformationen der Ressource integriert werden könnte.

Netzwerkressourceninformationen

Bei den Netzwerkressourcen stehen den Vergleichsschedulern die Informationen über die Geschwindigkeit und die Kosten exakt zur Verfügung. Die Auslastung einer Netzwerkressource ist, im Gegensatz zu den Rechen- und Speicherressourcen, im Allgemeinen jedoch nicht exakt ermittelbar. Dies liegt vor allem, wie bereits erläutert, an der Nutzung des Netzwerks durch eine Vielzahl weiterer Applikationen und Benutzer. Zur Modellierung dieser externen Netzwerkauslastung stehen in der entwickelten Simulationsumgebung zwei unterschiedliche Verfahren zur Verfügung.

[2] Bei Speicher- und Netzwerkressourcen werden die Begriffe Bandbreite und Geschwindigkeit synonym verwendet

Beim ersten Verfahren handelt es sich um ein randomisiertes, zeitlich begrenzt statisches Auslastungsmodell. Die Auslastung einer Verbindung wird, je nach Verbindungsart (Cluster-, Organisations- oder Interorganisationsverbindung), randomisiert aus einem vorgegebenen Wertebereich gewählt und bleibt anschließend in einem gewissen Intervall konstant. Nach Ablauf des Intervalls wird der Vorgang wiederholt. Dem eingesetzten Scheduler können die so ermittelten Auslastungen exakt, leicht oder stark fehlerbehaftet zur Verfügung gestellt werden. Bei leicht fehlerbehafteter Auslastung wird randomisiert aus dem näheren Wertebereich des exakten Wertes ein neuer Wert gewählt. Ein potenziell größerer Fehler entsteht, wenn aus dem gesamten Wertebereich ein neuer Wert generiert wird.

Das zweite Verfahren erzeugt die Auslastung auf Basis von Poissonprozessen. Die Modellierung eines Datentransfers auf Basis der Poisson-Wahrscheinlichkeitsverteilung $P_\lambda(k) = \frac{\lambda^k}{k!} \cdot e^{-\lambda},\ k \in N$ ist eine weitverbreitete Methode. Der Parameter $\lambda \in \Re^+$ ist hierbei sowohl der Erwartungswert als auch die Varianz der Verteilung. Wie in [PF95] erläutert wird, ist die Modellierung mit einem Poissonprozess jedoch nicht immer geeignet. Daher wird die Auslastung einer Netzwerkressource - getrennt für beide Richtungen - in der entwickelten Simulationsumgebung nicht durch einen Poissonprozess, sondern durch mehrere Poissonprozesse erzeugt. Hierbei wird die Zeit in Intervalle aufgeteilt - in den durchgeführten Simulationen beträgt die Intervalllänge 1000 Sekunden - und die Auslastung jedes Intervalls wird durch einen eigenen Poissonprozessen beschrieben. Dies folgt den Erkenntnissen aus [FM94], wo gezeigt werden konnte, dass sich die Netzwerkauslastung in einem begrenzten Zeitraum annähernd exakt mit einem Poissonprozess modellieren lässt.

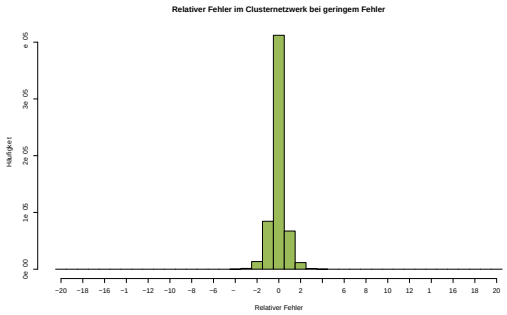
Dem eingesetzten Scheduler können die so ermittelten Auslastungen ebenfalls exakt, leicht oder stark fehlerbehaftet zur Verfügung gestellt werden. Die exakten Auslastungswerte für eine Netzwerkverbindung einer Simulation ergeben sich aus dem dieser Verbindung im entsprechenden Zeitintervall zugeordneten Poissonprozess. Die Parameter λ dieser Prozesse werden für jede Verbindungsart (Cluster-, Organisations- oder Interorganisationsverbindung) aus einem definierten Bereich ($[c_{min}, c_{max}]$, $[o_{min}, o_{max}]$, $[i_{min}, i_{max}]$) zufällig gewählt. Einer Verbindung v zwischen zwei Clustern innerhalb einer Organisation wird zu Beginn jedes Intervalles ein Poissonprozess wie folgt zugeordnet[3]:

$$\lambda_v = rand(\ o_{max} - o_{min}\ )\ +\ o_{min}$$

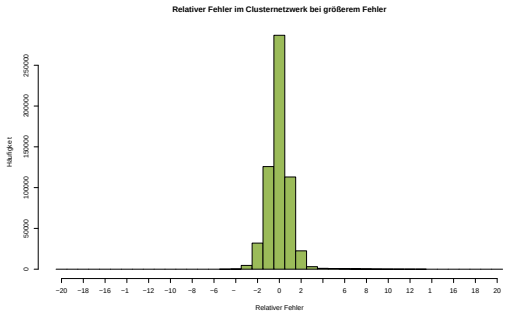
Bei geringem Fehler erhält der Scheduler für die Dauer eines Intervalls die korrekten Auslastungswerte. Anschließend wird die Auslastung von einem Poissonprozess mit einem randomisiert aus dem entsprechenden Bereich gewählten λ-Parameter generiert. Hierfür wird die randomisierte Bestimmung einfach noch einmal durchgeführt. Soll ein größerer Fehler erzeugt werden, wird von Beginn an die randomisierte Bestimmung des λ-Parameters des Poissonprozesses wiederholt.

Abbildung 4.1 zeigt die zufällig erzeugten *relativen* Fehler einer Simulationsumgebung über drei Intervalle. Ein positiver Wert entspricht dabei einer Überschätzung

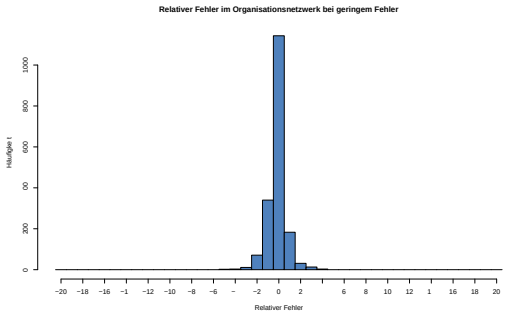
[3] rand(x) ist eine gleicherverteilte Zufallsvariable die Werten aus dem Intervall $[0; x]$ annimmt



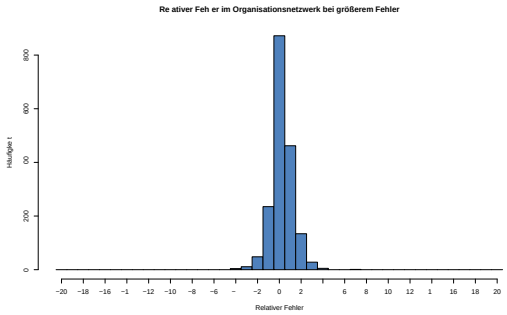
(a) Cluster - geringer Fehler



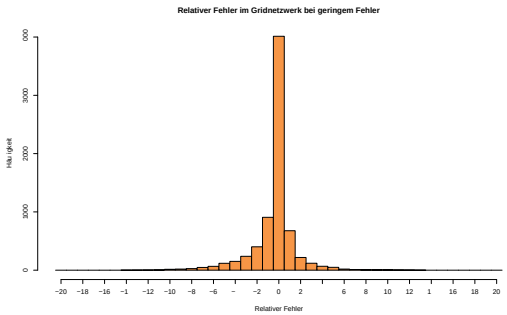
(b) Cluster - größerer Fehler



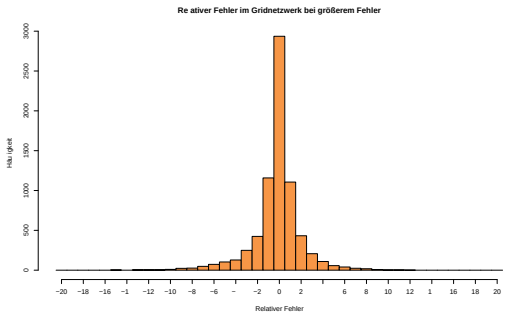
(c) Organisation - geringer Fehler



(d) Organisation - größerer Fehler



(e) Inter-Org. - geringer Fehler



(f) Inter-Org. - größerer Fehler

Abbildung 4.1: Histogramme der zufällig erzeugten, *relativen* Netzwerkfehler beim Poissonprozess.

der Auslastung und ein negativer Wert einer Unterschätzung. Wie aus den Histogrammen hervorgeht, steigen die von der Simulationsumgebung erzeugten relativen Fehler mit der Distanz die eine Netzwerkverbindung überbrückt.

Die Unterschiede zwischen den beiden Netzwerkauslastungsmodellen wurden in mehreren Simulationen mit den nachfolgend beschriebenen Vergleichsschedulern sowie dem DOHS untersucht. Es zeigt sich, dass das Netzwerkauslastungsmodell nur einen geringen Einfluss auf die von den unterschiedlichen Schedulern generierten Verarbeitungsaufwände hat. Daher wird für die nachfolgende Simulationen und die Evaluierung nur das realitätsnähere Poissonmodell verwendet.

Die zu verarbeitenden Aufträge

Die Verarbeitungsaufträge, die in der Simulation vom eingesetzten Scheduler auf die Ressourcen verteilt werden sollen, werden durch ihren Eintrittszeitpunkt, die zu verarbeitenden Datensätze, sowie die Anzahl der Operation pro MB definiert. Die Berechnungsdauer kann somit direkt aus der Anzahl der Operationen und der Gesamtgröße der Daten bestimmt werden.
Die Definition der Berechnungsdauer eines Auftrags über die Anzahl der Operationen pro MB ist für datenintensive Anwendungen naheliegend. Bei rechenintensiven Anwendungen ist die Berechnungsdauer allerdings häufig unabhängig von der Größe der Eingabedaten. Mit der gewählten Definition lassen sich trotzdem auch rechenintensive Anwendungen simulieren, indem die entsprechenden Aufträge nur kleine Eingabedaten umfassen und eine große Anzahl an Operationen pro MB aufweisen. Der dadurch entstehende, künstlich erzeugte Zusammenhang zwischen Eingabedatengröße und Ausführungsdauer hat jedoch auf die Simulationen keinen Einfluss.

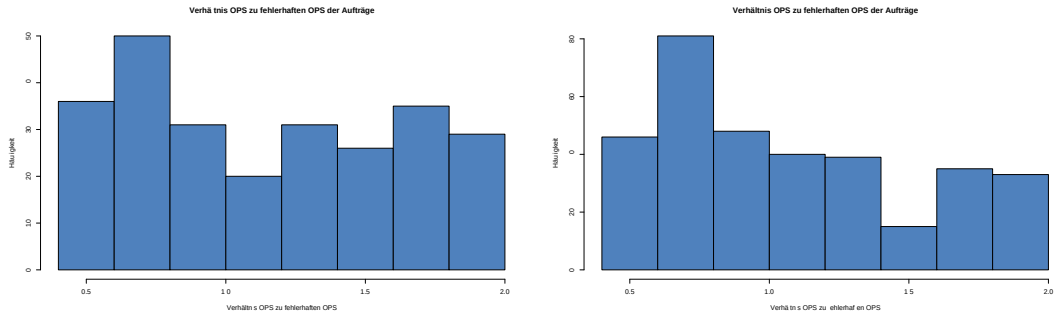


Abbildung 4.2: Histogramme des *relativen* Fehlers der benötigten Operationen.

Die Anzahl der von einem Auftrag benötigten Operationen ist, wie bereits erläutert, dem Scheduler im Allgemeinen nicht bekannt. Dem Scheduler kann daher entweder eine exakte o_c oder verfälschte Anzahl o_e von benötigten Operationen für das Scheduling zur Verfügung gestellt werden. Die Höhe des Fehlers der Operationen kann über einen Parameter γ gesteuert werden, wobei der verfälschte Wert randomisiert aus einem Intervall $[\, o_c/\gamma \,,\, o_c \cdot \gamma \,]$ gewählt wird. Für die nachfolgenden

Simulationen wurde mit $\gamma = 2$ ein geringer Fehler gewählt, so dass die fehlerbehafteten Operationen aus dem Intervall $[\, o_c/2 \,,\, o_c \cdot 2 \,]$ gewählt werden. Abbildung 4.2 zeigt beispielhaft den relativen Fehler (o_c/o_e) zweier unterschiedlicher Simulationen.

4.1.2 Ablauf der Simulation

Ein Simulationslauf führt, mit einem vorgegebenen Schedulingalgorithmus, das Scheduling der zu verarbeitenden Aufträge durch und simuliert die Ausführung der erzeugten Zuordnungen auf den Ressourcen der Umgebung. Die Eingabe der Simulation bilden die Ressourcen und deren Zughörigkeit zu Clustern und Organisationen sowie die zu verarbeitenden Aufträge.

Die Simulation startet zum Zeitpunkt 0 und wird in definierten Zeitschritten von Δ_i Sekunden - in den durchgeführten Simulationen wurde $\Delta_i = 10$ verwendet - weitergeführt. Zu Beginn jedes Zeitschrittes wird ein Scheduling mit dem vorgegebenen Schedulingalgorithmus durchgeführt. Dem Scheduler stehen alle, eventuell fehlerbehafteten, Ressourcen- und Auftragsinformationen für das Scheduling zur Verfügung. Sind noch wartende Aufträge in der Warteschlange oder ist ein neuer Auftrag im letzten Zeitschritt hinzugekommen, werden für diese, in der Reihenfolge ihres Eintreffens, nacheinander Zuordnungen generiert. Alle Aufträge die keiner freien Ressource zugeordnet werden konnten, werden wieder an die Warteschlange angefügt. Für Zuordnungen die eine freie Rechenressource enthalten und damit sofort ausgeführt werden können, wird die Rechenressource reserviert und der Transfer der Datensätze gestartet. Am Ende eines jeden Zeitschrittes wird der Fortschritt aller aktuell laufenden Transfers und Berechnungen bestimmt und die Auslastung der beteiligten Ressourcen angepasst. Wurde ein Auftrag innerhalb eines Zeitschritts beendet, wird, falls vorhanden, der oberste Auftrag der Warteliste der auf diese Rechenressource wartet gestartet.

Das Verhalten eines Schedulers bei fehlerhaften Informationen kann mit dem gleichen Ablauf untersucht werden. Es werden dem Scheduler jedoch nicht die exakten Informationen, sondern fehlerbehaftete Informationen während des Schedulings übergeben. Wie aus dem in Auflistung 6 dargestellten Algorithmus hervorgeht, ist die Simulation der Ausführung von den fehlerbehafteten Informationen nicht betroffen.

4.2 Vergleichsscheduler

Ein zentrales Ziel der durchgeführten Simulationen ist, herauszufinden, wie gut der DOHS-Algorithmus den Verarbeitungsaufwand in unterschiedlichen Umgebungen und Auftragszusammensetzungen minimiert. Hierfür sollte der vom DOHS erzeugte Verarbeitungsaufwand mit dem minimalen Verarbeitungsaufwand verglichen werden. Wie bereits in Abschnitt 1.1 beschrieben, müssten zur Bestimmung des minimalen Verarbeitungsaufwands $|\mathbf{Z}| = a! \cdot r^a \cdot s^d$ mögliche Zuordnungen betrachtet werden. Dies ist für realitätsnahe Umgebungen, auch in einer Simulation, nicht möglich.

Algorithmus 6: Simulation

Eingabe : Scheduler *Scheduler*, Alle Aufträg A, Umgebung R, Zeitschritt Δ_i
Ausgabe : $\varphi(Z)$

$\varphi = 0$
$i = 0$
$Z = \emptyset$

`// Simulation des aktuellen Zeitabschnitts`
while *nicht alle Aufträge beendet* **do**
  `// Scheduling aller noch nicht ausgeführter Aufträge`
  foreach *nicht ausgeführter Auftrag a mit Eintrittszeit* $> i$ **do**
    `// Sched erhält korrekte oder fehlerbehaftete Informationen`
    $z = Scheduler(a,\ i,\ R)$
    if *z kann ausgeführt werden* **then**
      $Z = Z \cup z$
    end
  end
  `// Simuliere Ausführung auf den Ressourcen`
  foreach $z \in Z$ **do**
    `// Simuliere Datentransfer und Ausführung von` z
    `// Speichere entstande Dauer und Kosten`
    $(\tau_z, \kappa_z) += Sim(\ z, i, \Delta_i\ )$
    if *Ausführung von z beendet* **then**
      $\varphi = \varphi + \tau_z \cdot \kappa_z$
      $Z = Z \setminus z$
    end
  end
  $i = i + \Delta_i$
end
return φ

Als Referenz wird daher ein Online-Scheduling durchgeführt, das für jeden einzelnen Auftrag a den Verarbeitungsaufwand für alle $|Z_a| = r \cdot s^d$ möglichen Zuordnungen bestimmt. Gewählt wird die Zuordnung mit minimalem Verarbeitungsaufwand.
Dies kann als das bestmögliche *Online*-Scheduling angesehen werden, wobei nicht

$$\min_{Z \in \mathbf{Z}} \{\varphi(Z)\}$$

sondern

$$\sum_{a \in A} \min_{z \in Z_a} \{\varphi(z)\}$$

bestimmt wird.
Der entsprechende Referenzalgorithmus für das Scheduling eines Auftrags wird in Auflistung 7 dargestellt, wobei R alle Ressourcen mit ihrem aktuellen Status, t den aktuellen Zeitpunkt und e den Eintrittszeitpunkt des Auftrags beschreiben. Als Schedulingfunktion wird der Verarbeitungsaufwand $Sched(z) = \varphi(z)$ für eine Zuordnung verwendet.

Algorithmus 7: Online-Schedulingalgorithmus

Eingabe : $a = (e, p, d_1, \ldots, d_m)$, t, R und $Sched(z)$
Ausgabe : Beste Zuordnung z^*

```
1  z* = ∅
2  w* = ∞
   // Bestimme für alle Ressourcen...
3  foreach c ∈ R do
       // ...und alle möglichen Speicherzuordnungen...
4      foreach (s_1, s_2, ..., s_m) ∈ (R^{d_1} × R^{d_2} × ... × R^{d_m}) do
           // ...die gewählte Zielfunktion
5          w = Sched(e, c, s_1, s_2, ..., s_m)
6          if w < w* then
7              w* = w
8              z* = (c, s_1, s_2, ..., s_m)
9          end
10     end
11 end
12 return z*
```

Ein weiteres Ziel der durchgeführten Simulationen ist der Vergleich des DOHS-Algorithmus mit bestehenden Schedulern. Anstatt den DOHS mit allen in Kapitel 2 vorgestellten bestehenden Schedulingalgorithmen, die sich meist nur in bestimmten Details unterscheiden, zu vergleichen, werden ähnliche Scheduler zusammengefasst und durch die allgemeinste Implementierung repräsentiert. Diese Vorgehensweise erlaubt darüber hinaus auch den Vergleich des DOHS mit Schedulern, die in dieser Arbeit nicht betrachtet werden.

Betrachtet man die in Abschnitt 2.1.3 vorgestellten bestehenden Grid-Scheduler, lassen sich diese in deterministische Heuristiken und stochastische, metaheuristische Optimierungsverfahren unterteilen. Die heuristischen Verfahren unterscheiden sich primär dadurch, welche Zuordnungen beim Scheduling betrachtet werden, wie die Ressourceninformationen bestimmt und eingesetzt werden sowie durch die verwendete Zielfunktion. Der Algorithmus 7 kann als allgemeinste Form dieser Heuristiken angesehen werden, da dieser alle möglichen Zuordnungen und Ressourceninformationen berücksichtigt sowie die Verwendung beliebiger Zielfunktionen ermöglicht. Da, im Gegensatz zu den bestehenden Verfahren, immer alle möglichen Zuordnungen betrachtet werden, ist die von diesem Algorithmus gewählte Zuordnung für einen Auftrag besser oder mindestens gleichwertig im Vergleich zu einer Zuordnung eines bestehenden Schedulers.
Wie bereits erwähnt, wird dieser Algorithmus mit der Schedulingfunktion $Sched(z) = \varphi(z)$ als Referenz verwendet. Als Repräsentanten der bestehenden Grid-Scheduler die auf das in dieser Arbeit entwickelte, erweiterte Umgebungsmodell angepasst wurden, aber weiterhin nur die Kosten *oder* die Dauer der Ausführung betrachten, werden als Schedulingfunktion $Sched(z) = \kappa(z)$ beziehungsweise $Sched(z) = \tau(z)$ verwendet. Die bestehenden Grid-Scheduler, die auf dem klassischen Umgebungsmodell[4] basieren, werden durch die Schedulingfunktionen $Sched(z) = \tilde{\varphi}(z) = \tilde{\tau}(z) \cdot \tilde{\kappa}(z)$, $Sched(z) = \tilde{\tau}(z)$ sowie $Sched(z) = \tilde{\kappa}(z)$ repräsentiert. Diese Funktionen folgen der Annahme der bestehenden Scheduler, dass die Transferdauer und damit auch die Transferkosten innerhalb einer Organisation vernachlässigt werden können:

$$\tilde{\tau}_t(d, s, c) = \begin{cases} 0 & \text{falls } c, s \text{ in der gleichen Org.} \\ \tau_t(d, s, c) & \text{sonst} \end{cases}$$

Der Algorithmus 7 kann auch für ein, auf die Verarbeitung mehrerer Datensätze, erweitertes MapReduce-Scheduling eingesetzt werden. Wie auch beim MapReduce-Scheduler entscheidet die Distanz zwischen der Rechenressource und den Speicherressourcen über die Wahl der Zuordnung. Die Schedulingfunktion $Sched(z) = MR(z)$ des erweiterten MapReduce-Schedulings ergibt sich für m Datensätze zu

$$MR(z) = \sum_{i=1}^{m} mr(d, s_i, c) \quad mit$$

$$mr(d, s, c) = \begin{cases} \mu_1 & \text{falls } c = s \\ \mu_2 & \text{falls } c, s \text{ im gleichen Cluster} \\ \mu_3 & \text{falls } c, s \text{ in der gleichen Organisation} \\ \mu_4 & \text{sonst} \end{cases}$$

wobei die Parameter μ_1 bis μ_4 zur Gewichtung der Distanz verwendet werden.

Die stochastischen, metaheuristischen Optimierungsverfahren werden von einem Genetischen Algorithmus repräsentiert. Bei dem in Auflistung 8 beschriebenen

[4] Alle Ressourcen einer Organisation werden zu einer virtuellen Ressource zusammengefasst

Algorithmus 8: Genetischer-Schedulingalgorithmus

Eingabe : t, e, $p \in P$, $d_1, \ldots, d_m \in D$, n_i, n_e
Ausgabe : Beste Zuordnung z^*

$z[n_i] = (\emptyset, ..., \emptyset)$
// Erzeuge n_i Individuen
for $i = 0$ **to** n_i **do**
  $z[i] = (rand(R), rand(R^{d_1}), \ldots, rand(R^{d_m}))$
end
$z^* = \emptyset$
$w^* = \infty$
// Simuliere über n_e Generationen
for $epoch = 0$ **to** n_e **do**
  $w[n_i] = (\infty, ..., \infty)$
  // Bestimme Aufwand für jedes Individuum
  for $i = 0$ **to** n_i **do**
    $w[i] = \varphi(z[i])$
    if $w[i] < w^*$ **then**
      $w^* = w[i]$
      $z^* = z[i]$
    end
  end
  // Rekombiniere die Individuen
  Recombine(z)
  // Mutiere die Individuen
  Mutate(z)
end
return z^*

Online-Algorithmus handelt es sich um einen einfachen Genetischen Algorithmus ohne spezielle Anpassungen. Die Zuordnungen für einen Auftrag mit m Datensätzen werden als Gen der Form $(R, R^{d_1}, \ldots, R^{d_m})$ beschrieben. Zu Beginn werden n_i Individuen randomisiert erzeugt. Diese entwickeln sich in n_e Generationen über Mutation und Rekombination weiter, wobei Individuen mit einem niedrigen Verarbeitungsaufwand bei der Rekombination bevorzugt werden. Der Algorithmus gibt die Zuordnung des Individuums mit minimalem Verarbeitungsaufwand zurück.
Wie Gut der Genetische Algorithmus den Verarbeitungsaufwand minimiert, hängt offensichtlich von der Anzahl der Individuen und der Generationen sowie der Mutations- und Rekombinationsfunktionen ab. Im Allgemeinen kann erwartet werden, dass der Verarbeitungsaufwand einer von einem Genetischen Algorithmus erzeugten Zuordnung für einen Suchraum der Größe $r \cdot s^d$ - bei nicht allzu großen s und d - beinahe identisch mit dem Referenz-Online-Scheduler ist.

Bei den nachfolgend beschriebenen Simulationen wurde auch ein Scheduler der rein randomisierte Zuordnungen der Form

$$z = (\ rand(R), rand(R^{d_1}), \ldots, rand(R^{d_m})\ )$$

erzeugt, als Vergleichsscheduler eingesetzt.

4.3 Die generierten Simulationen

Zur realitätsnahen Evaluierung des DOHS-Algorithmus wurden auf Basis der entwickelten Simulationsumgebung mehrere Simulationen durchgeführt. Mithilfe der Simulationen wird einerseits der Einfluss der Parameter $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ und β_1, β_2 auf den DOHS-Algorithmus untersucht, andererseits kann anhand der durchgeführten Simulationen der DOHS-Algorithmus mit bestehenden Schedulern verglichen werden.

Da sich der Aufbau realer, organisationsübergreifender Umgebungen stark unterscheiden kann, wurde die Simulationsumgebung für jede durchgeführte Simulation randomisiert erzeugt. Tabelle 4.3 zeigt die für die Evaluierung des DOHS und den Vergleich mit den bestehenden Schedulern generierten 10 Simulationen, mit der unterschiedlichen Anzahl an Ressourcen, Clustern sowie Organisationen. Die Anzahl der Aufträge wird für jede Simulation randomisiert erzeugt und liegt im Bereich von 100 bis 350 Aufträgen. Die ebenfalls randomisiert generierten Umgebungen bestehen aus 80 bis 120 Ressourcen, die auf 3 bis 6 Cluster sowie 3 oder 4 Organisationen verteilt sein können.

Sim	$\|A\|$	$\|R\|$	n_c	n_o
1	337	97	3	3
2	304	116	4	3
3	335	97	4	3
4	258	90	4	3
5	318	114	4	3
6	261	82	3	3
7	337	80	3	3
8	334	104	6	4
9	273	111	6	4
10	301	97	3	3

Die 10 erzeugten Simulationen umfassen sowohl Umgebungsstrukturen mit mehreren Clustern pro Organisation als auch Umgebungen mit nur einem Cluster pro Organisation - entsprechend dem Umgebungsmodell der bestehenden Verfahren. Dies ermöglicht eine Evaluierung des DOHS und den Vergleich mit bestehenden Verfahren ohne das Ergebnis durch die Einschränkung auf eine bestimmte Umgebungsstruktur zu verzerren.

Umgebungsstruktur

Die Zuordnung der Ressourcen zu den Clustern sowie die Zuordnung der Cluster zu den Organisationen erfolgt gleichfalls zufällig. Abbildung 4.3 zeigt beispielhaft die so erzeugte Umgebungsstruktur der 9. Simulation, inklusive der Geschwindigkeit sowie der Kosten der Netzwerkverbindungen. Die Verbindungen innerhalb eines Clusters sind grün dargestellt. Verbindungen zwischen Clustern einer Organisation sind blau gefärbt. Die Verbindungen zwischen Organisationen werden orange dargestellt. Die Breite einer Linie spiegelt die angegebene Bandbreite wieder.

Rechen- und Speicherressourcen

Zusätzlich zur Umgebungsstruktur werden auch alle Ressourcen- und Auftragseigenschaften randomisiert generiert. Bei den Rechenressourcen wird die Anzahl der Prozessorkerne sowie die Geschwindigkeit und die Kosten zufällig erzeugt. Die Geschwindigkeit einer Rechenressource wird für die Simulation in Operationen pro Sekunde angegeben und bezieht sich auf einen Prozessorkern der Ressource. Die Geschwindigkeit wird aus einem Bereich von 400 bis 1000 Operationen pro Sekunde gewählt. Die schnellsten Ressourcen sind somit pro Prozessorkern 150% schneller

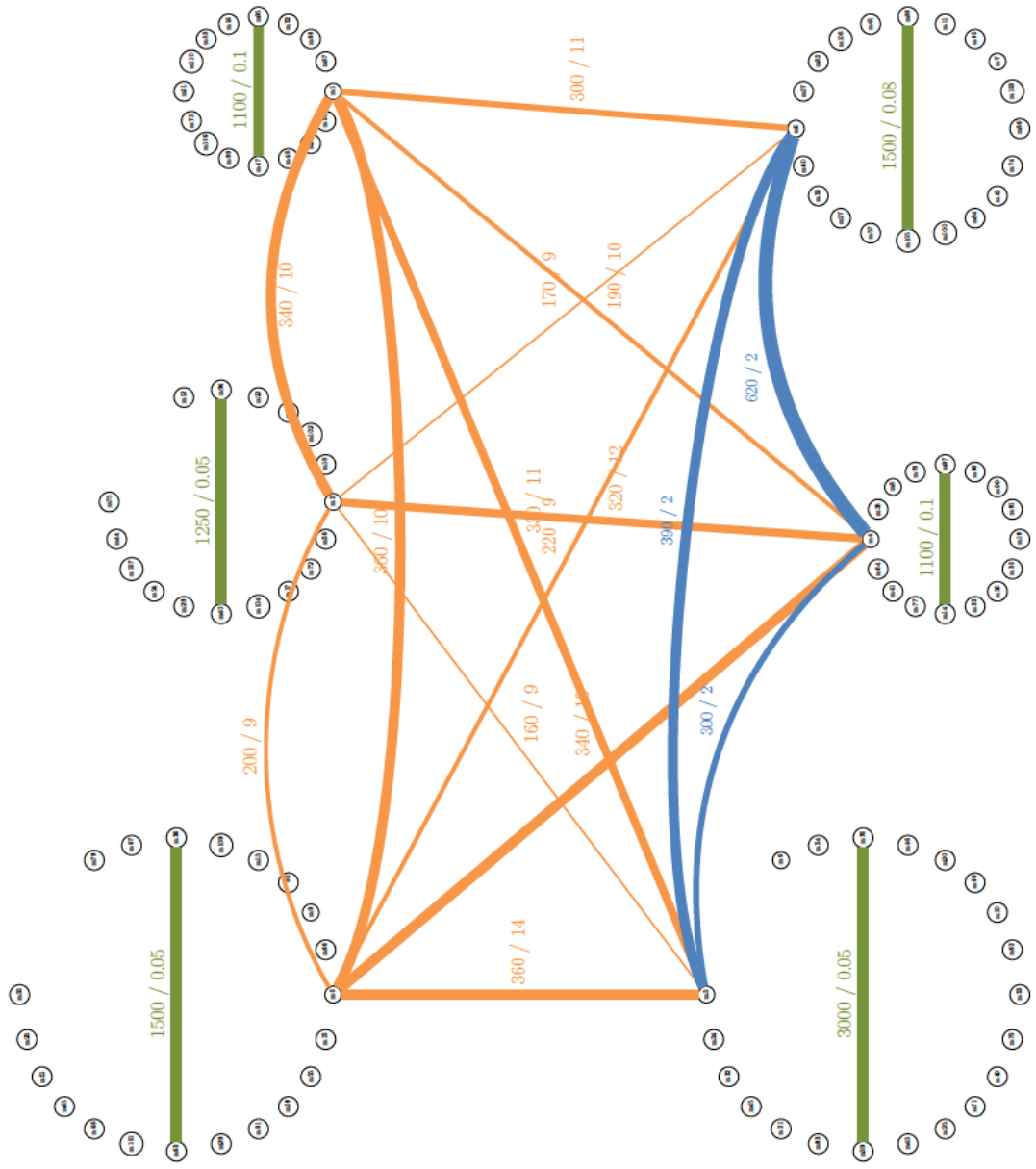
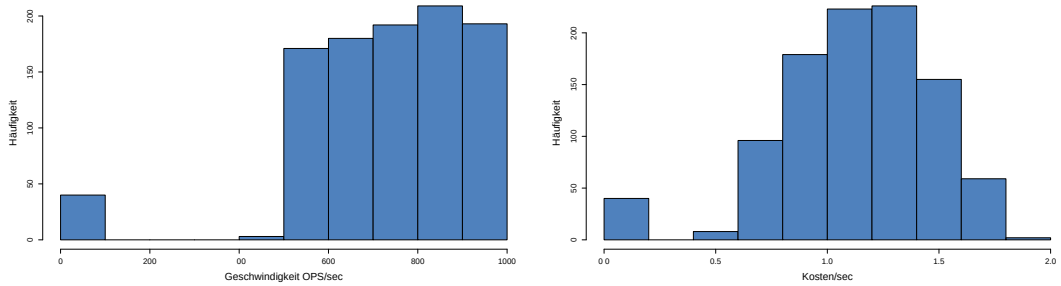


Abbildung 4.3: Umgebungsstruktur der 9. Simulation.

als die langsamsten Ressourcen. Dies entspricht in etwa dem Geschwindigkeitsunterschied zwischen der aktuellen und der letzten Prozessorgeneration. Da nicht alle Speicherressourcen auch als Rechenressourcen genutzt werden können, wird deren Rechengeschwindigkeit auf 0 gesetzt. Die Kosten werden hierbei in Abhängigkeit der Rechengeschwindigkeit bestimmt, wobei gilt, dass schnellere Ressourcen tendenziell teurer sind als langsame. Die in Abbildung 4.4 dargestellten Histogramme zeigen die Geschwindigkeit und die Kosten der Rechenressourcen über alle Simulationen.



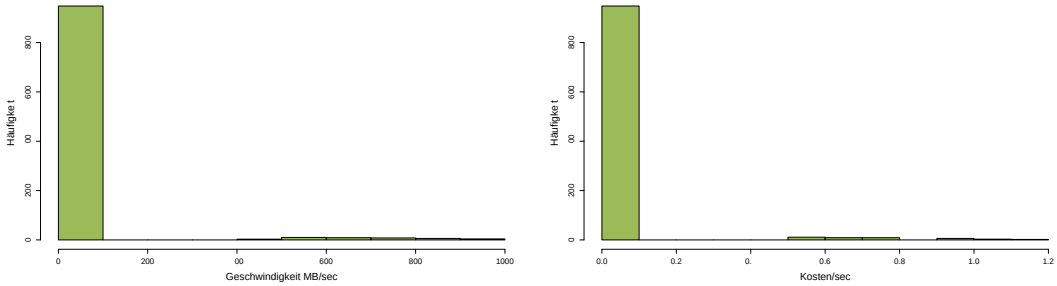
(a) Rechengeschw. (b) Rechenkosten

Abbildung 4.4: Histogramme der Rechengeschwindigkeit und -kosten über alle Simulationen.

Im Gegensatz zu den Rechenressourcen existieren bei den Speicherressourcen zwei unterschiedliche Ausprägungen. Die eine Gruppe von Speicherressourcen repräsentiert die speziellen Clusterspeichersysteme aktueller Rechencluster. Diese Speichersysteme bieten, wie in Abschnitt 2.1 beschrieben, durch ihren Aufbau einen hohen Durchsatz. Von diesen schnellen Speichersystemen sind nur wenige, meist nur eines, in einem Cluster vorhanden. Die Geschwindigkeit dieser Speichersysteme hängt insbesondere von der Anzahl der Rechenressourcen im Cluster ab. In realen Umgebungen wird die Geschwindigkeit eines Speichersystems so gewählt, dass jedem Prozessorkern ein gewisser Durchsatz zur Verfügung steht. So liegt dieser Wert bei Europas derzeit schnellstem Clustercomputer, dem SuperMuc[5], bei ungefähr $1,3$, wobei in anderen Anwendungsfeldern auch deutlich höhere Speichergeschwindigkeiten pro Prozessorkern möglich sind. Bei den durchgeführten Simulationen mit maximal 120 Ressourcen ergibt sich in der Simulationsumgebung ein Geschwindigkeitsbereich der Clusterspeichersysteme zwischen 400 und 1000 MB pro Sekunde.
Die andere Gruppe von Speicherressourcen bilden die wesentlich langsameren Speichereinheiten der Rechenressourcen. In aktuellen Rechenressourcen ist meist nur eine günstige Festplatte verbaut, deren Geschwindigkeit in der Simulationsumgebung mit 30 bis 100 MB pro Sekunde angenommen wird.
Die Geschwindigkeit und die Kosten der Speicherressourcen werden ebenfalls randomisiert erzeugt, wobei die Geschwindigkeit sowie die Kosten entsprechend dem

[5] http://www.lrz.de/services/compute/supermuc/systemdescription, 155000 Prozessorkerne und 200000 MB/s

Bereich der Ausprägung gewählt werden. Auch für die Speicherressourcen gilt, dass schnellere Ressourcen im Allgemeinen teurer als langsame sind. Abbildung 4.5 zeigt



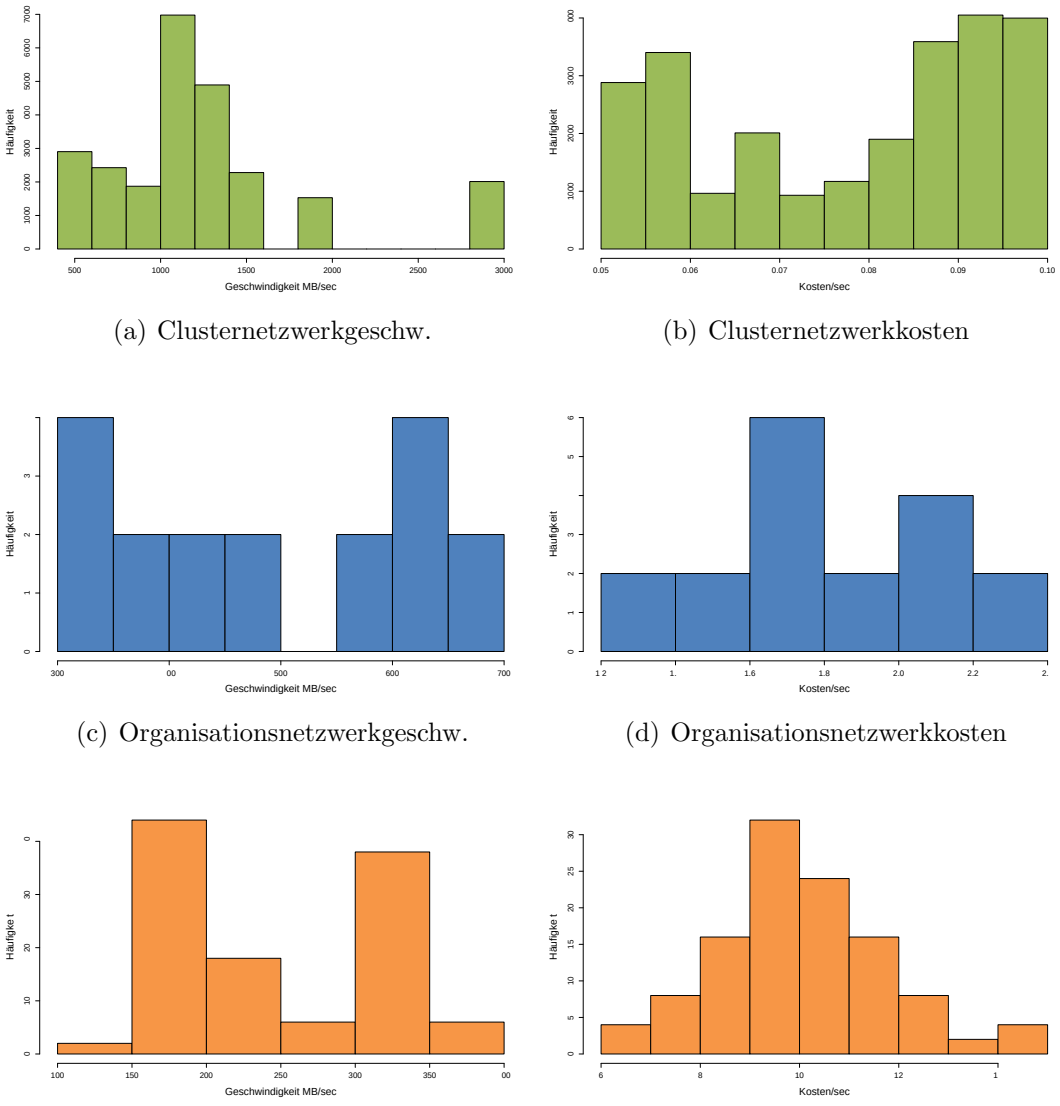
(a) Speichergeschw. (b) Speicherkosten

Abbildung 4.5: Histogramme der Speichergeschwindigkeit und -kosten über alle Simulationen.

die Histogramme der Geschwindigkeit und Kosten der Speicherressourcen über alle Simulationen.

Netzwerkressourcen

Die Generierung der Geschwindigkeiten und der Kosten der Netzwerkressourcen berücksichtigt die hierarchische Struktur des Netzwerks, wobei zwischen Cluster-, Organisations- und Interorganisationsnetzwerkressourcen (kurz Grid-Netzwerk) unterschieden wird. Wie bei den anderen Ressourcen, werden die Geschwindigkeit und die Kosten aller Netzwerkressourcen randomisiert erzeugt, wobei die speziellen Eigenschaften jeder Ebene beachtet werden. So sind alle Ressourcen innerhalb eines Clusters direkt über ein schnelles, kostengünstiges Netzwerk miteinander verbunden. Die Geschwindigkeit und die Kosten aller Netzwerkressourcen eines Cluster sind somit identisch.
Wie aus dem Umgebungsmodell hervorgeht, kann es sich bei Netzwerkverbindungen zwischen Clustern innerhalb einer Organisation entweder um schnelle, günstige LAN-Verbindungen oder um langsamere, teurere WAN-Verbindungen handeln. Dies spiegelt sich in den erzeugten Geschwindigkeiten und Kosten der Organisationsnetzwerkressourcen wieder.
Für die Netzwerkressourcen innerhalb eines Clusters oder Organisation gilt, wie für die Rechen- und Speicherressourcen, dass schnellere Ressourcen im Allgemeinen teurer als langsame sind. Dies gilt nicht unbedingt bei Interorganisationsverbindungen, da bei diesen WAN-Verbindungen neben der Geschwindigkeit noch weitere Faktoren, etwa die Distanz, die Kosten beeinflussen. Entsprechend dem Umgebungsmodell verfügen die Interorganisationsverbindungen über eine geringe Bandbreite und hohe Kosten. Die Histogramme aus Abbildung 4.6 zeigen die von der Simulationsumgebung generierten Netzwerkgeschwindigkeiten und -kosten über alle Simulationen.



(a) Clusternetzwerkgeschw. (b) Clusternetzwerkkosten

(c) Organisationsnetzwerkgeschw. (d) Organisationsnetzwerkkosten

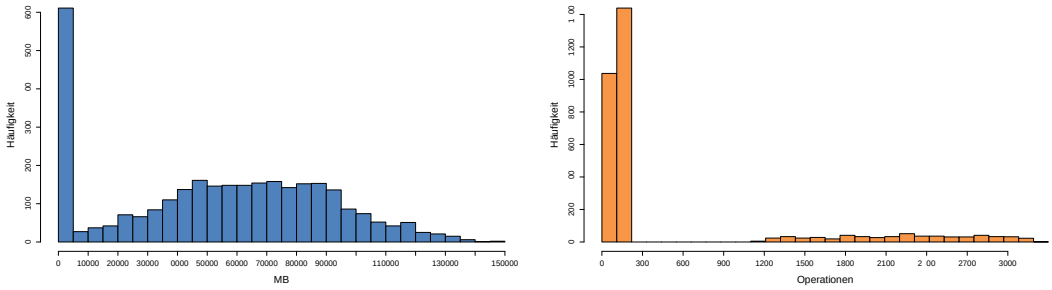
(e) Grid-Netzwerkgeschw. (f) Grid-Netzwerkkosten

Abbildung 4.6: Histogramme der Netzwerkgeschwindigkeiten und -kosten.

Daten und Aufträge

Neben den Ressourcen werden auch die Aufträge und die von diesen zu verarbeitenden Datensätze randomisiert erzeugt. Für eine Simulation werden 1000 Datensätze mit einer Größe von 1MB bis zu 50GB generiert, wobei mindestens 50 kleine Datensätze mit einer Größe unter 50MB erzeugt werden. Ein Datensatz kann bis zu n_c Repliken besitzen, wobei in den Simulationen ein Datensatz im Mittel auf $2,8$ Speicherressourcen vorhanden ist. Bei der Zuweisung der Repliken auf die Speicherressourcen wird sichergestellt, dass jeder Datensatz auf mindestens einem schnellen Clusterspeichersystem abgespeichert ist.

Bei der Generierung der Aufträge werden sowohl die zu verarbeitenden Datensätze als auch die benötigten Rechenoperationen pro Datenmenge randomisiert bestimmt. Die Simulationsumgebung unterscheidet hierbei zwischen drei Klassen von Aufträgen: Datenintensive, rechenintensive und gemischt intensive Aufträge. Für jede dieser Klassen kann eine Wahrscheinlichkeit angegeben werden, mit der ein Auftrag in die entsprechende Klasse fällt. Bei den durchgeführten Simulationen wurden folgende Wahrscheinlichkeiten verwendet: 30% datenintensive, 20% rechenintensive und 50% gemischt intensive Aufträge.



(a) Datenmenge (b) Operationen pro MB

Abbildung 4.7: Histogramme der Datenmenge und der Operationen pro MB der Aufträge.

Wird ein Auftrag als datenintensive markiert, werden bis zu 4 Datensätze - im Mittel $2,75$ - zufällig ausgewählt. Des Weiteren wird die Anzahl der benötigten Berechnungsoperationen pro MB zufällig aus dem Bereich $0,001$ bis $1,5$ bestimmt. Diese Werte repräsentieren typische, datenintensive Anwendungen, wie Datenfilterung oder Aggregation, in denen teilweise nur ein kleiner Teil des Datensatzes betrachtet wird.

Einem rechenintensiven Auftrag werden ebenfalls bis zu 4 Datensätze zufällig zugeordnet, wobei diese allerdings nur bis zu 50MB groß sein dürfen. Dies folgt aus der Beobachtung, dass rechenintensive Anwendungen häufig nur kleinere Eingabedaten verarbeiten. Ein rechenintensiver Auftrag benötigt zwischen 1000 und 4000 Operation für die Verarbeitung eines MB der Eingabedaten.

Auch für einen gemischt intensiven Auftrag werden bis zu 4 beliebige Datensätze

ausgewählt. Die benötigten Berechnungsoperationen liegen für diesen Auftragstyp jedoch zwischen $1,5$ und 50 Operationen pro MB. Durch die Auswahl von bis zu vier 50GB großer Datensätze und bis zu 50 Operationen pro MB können gemischt intensive Aufträge gleichzeitig aufwendige Datentransfers und aufwendige Berechnungen erfordern.

Abbildung 4.7 zeigt die unter diesen Vorgaben in den 10 Simulationen entstandenen Histogramme der zu verarbeitenden Datenmenge und der Berechnungsoperationen der Aufträge.

4.4 Simulationsergebnisse

Die Evaluierung des DOHS-Algorithmus auf Basis der im vorherigen Abschnitt beschriebenen 10 randomisiert generierten Simulationsumgebungen erleichtert die Übertragbarkeit der gewonnenen Erkenntnisse auf ähnliche, reale Umgebungen. Die unterschiedlichen Umgebungsstrukturen und Auftragscharakteristiken ermöglichen dabei einerseits den Einfluss der Parameter $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ und β_1, β_2 auf den DOHS-Algorithmus detailliert zu untersuchen, andererseits kann der DOHS-Algorithmus mit bestehenden Schedulern verglichen werden, ohne dabei die Ergebnisse auf eine bestimmte Problemstellung einschränken zu müssen. Im Folgenden werden zunächst die Ermittlung der Parameter sowie die Einflüsse der Parameter auf den DOHS-Algorithmus untersucht. Anschließend wird der DOHS-Algorithmus mit den bestehenden Schedulern detailliert verglichen.

4.4.1 Die Parameter des DOHS

Wie im vorherigen Kapitel erläutert, sind die Parameter $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ und β_1, β_2 entscheidend für die Güte der vom DOHS-Algorithmus erzeugten Zuordnungen. Mithilfe der Parameter β_1, β_2 können Applikationen oder Benutzer dem DOHS-Algorithmus das Verhältnis zwischen Transfer- und Berechnungsaufwand mitteilen. Die Parameter $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ dienen sowohl zur Kompensation der fehlenden Netzwerkinformationen als auch zur Schonung der Netzwerk- und Speicherressourcen. Insbesondere für die α-Parameter stellt sich die Frage, wie die Parameter für reale Umgebungen ermittelt werden können und wie sich eine Veränderung der Parameter auf die vom DOHS erzeugten Zuordnungen auswirkt.

Jeweils 10 Simulationen mit	
β_1	1
β_2	1
α_1	1
α_2	1,5
α_3	1,5 / 2
α_4	1,5 / 2 / 2,5 / 3 / 3,5 / 4
min $\overline{\varphi}$	452823528
max $\overline{\varphi}$	942148359
α_3	5 / 10 / 15 / 20 / 25
α_4	75 / 100 / 150 / 200 / 250 / 300 / 500
min $\overline{\varphi}$	391217871
max $\overline{\varphi}$	410804707
$SD\ \overline{\varphi}$	5505184

Wie aus der Definition der in Abschnitt 3.2 vorgestellten Schedulingfunktion hervorgeht, sollten die α-Parameter das Verhältnis der Kosten sowie der Auslastungen zwischen den Netzwerkebenen widerspiegeln. Damit nachfolgende Aufträge noch genügend freie Ressourcen, insbesondere Netzwerk- und Speicherressourcen, zur Verfügung stehen, sollte die mittlere Anzahl an Ressourcen die sich eine Verbindung auf einer Netzwerkebene teilen ebenfalls in die Bestimmung der α-Parameter mit einfließen.
Auf Basis dieser Überlegungen lassen sich die α-Parameter für die 10 generierten Simulationsumgebungen wie folgt abschätzen:

1. α_1 ist laut Definition der Schedulingfunktion 1.

2. Nimmt man an, dass das Netzwerk innerhalb von Clustern eine hohe Bandbreite und geringe Kosten besitzt und jede Netzwerkverbindung nur zwei Ressourcen miteinander verbindet, bietet es sich an, α_2 nur geringfügig größer als α_1 zu wählen - etwa $\alpha_2 = 1,25$ oder $\alpha_2 = 1,5$.

3. Bei Verbindungen zwischen Clustern innerhalb einer Organisation kann es sich um LAN- oder WAN-Leitungen handeln, die stark unterschiedliche Geschwindigkeiten und Kosten besitzen. Zur Abschätzung von α_3 ist eine Betrachtung der typischen Auslastung dieser Verbindungsart hilfreich. Zum einen werden diese Verbindungen häufig von anderen Diensten und Applikationen der Organisation genutzt. Zum anderen ist anzunehmen, dass manche Aufträge auf

Transfers innerhalb einer Organisation angewiesen sind. Die Betrachtung der Clustergrößen kann für eine Abschätzung dieser Auslastung genutzt werden. Bei maximal 120 Ressourcen in bis zu 6 Clustern ergibt sich eine Clustergröße von 20 bis 40 Ressourcen in den Simulationen. Nimmt man an, dass zwischen zwei Clustern dieser Größe immer zwei Ressourcen Daten austauschen oder andersweitig eine Auslastung erzeugt wird und auch Kosten für die Verwendung anfallen können, kann man etwa $\alpha_3 = 5$ oder $\alpha_3 = 15$ wählen.

4. Die Bestimmung von α_4 kann nach den gleichen Überlegungen erfolgen, wobei für die 10 Simulationsumgebungen Werte wie $\alpha_4 = 50$ oder $\alpha_4 = 100$ den erhöhten Transferaufwand zwischen Organisationen passend beschreiben.

Ein wichtiges Ziel des entwickelten DOHS-Algorithmus ist, auf Informationen zu verzichten, die sich nicht oder nur schwer exakt bestimmen lassen. Betrachtet man die α-Parameter, könnte man einwenden, dass diese ebenfalls nicht exakt bestimmt werden können. Eine exakte Bestimmung der Parameter ist jedoch gar nicht notwendig, da die α-Parameter nur den relativen Abstand des Transferaufwandes zwischen den Netzwerkebenen beschreiben. Solange die Parameter, insbesondere α_3 und α_4, nicht zu klein gewählt werden, erzeugt der DOHS-Algorithmus unabhängig von den absoluten Werten der α-Parameter „lokale“ Zuordnungen. Diese Zuordnungen besitzen eine möglichst geringe Distanz zwischen der Masse der Daten und den Rechenressourcen und führen somit zu einer effizienten Nutzung der Ressourcen unter Vermeidung langer, aufwändiger Transfers. Dies bestätigen auch die in Tabelle 4.4.1 aufgeführten Simulationen, die zeigen, wie der DOHS-Algorithmus auf eine Veränderung der α-Parameter reagiert.

In den 470 Simulationen wurden 47 unterschiedliche Kombinationen der Parameter α_3 und α_4 untersucht. Für jede dieser Kombinationen wurden die Aufträge der 10 unterschiedlichen Simulationsumgebungen mit dem DOHS auf die Ressourcen verteilt und die Ausführung simuliert. Anschließend wurde der Mittelwert $\overline{\varphi}$ aus den 10 Verarbeitungsaufwänden für jede Kombination ermittelt.

Die ersten 12 Kombinationen zeigen die Ergebnisse des DOHS bei der Wahl kleiner Werte für die Parameter α_3 und α_4. Den schlechtesten mittleren Verarbeitungsaufwand von 942148359 erzeugten die Parameter $\alpha_3 = 1,5$ und $\alpha_4 = 1,5$. Mit steigenden Werten der Parameter senkt sich der Verarbeitungsaufwand auf 452823528 ($\alpha_3 = 2,\ \alpha_4 = 4$).
Die Ergebnisse dieser kleinen Parameterwerte sind erwartungsgemäß deutlich schlechter, als die der weiteren 35 untersuchten Kombinationen, deren α_3 und α_4 Werte auf Basis der zuvor besprochenen Überlegungen gewählt wurden. Wie aus der Tabelle hervorgeht, beträgt der minimale Verarbeitungsaufwand 391217871 ($\alpha_3 = 10,\ \alpha_4 = 75$) und der maximale Verarbeitungsaufwand 410804707 ($\alpha_3 = 5,\ \alpha_4 = 150$) für die entsprechend gewählten Parameter α_3 und α_4. Die Standardabweichung des mittleren Verarbeitungsaufwandes der 35 Kombinationen mit passend gewählten Parametern beträgt lediglich 5505184, was ungefähr nur 1% des mittleren Verarbeitungsaufwandes entspricht.
Die durchgeführten Simulationen zeigen, wie zu erwarten war, dass ab einer gewis-

sen Größe, die Veränderung der α-Parameter nur zu einer geringen Veränderung der generierten Zuordnungen führt.

Neben den α-Parametern ermöglichen die Parameter β_1, β_2 Einfluss auf die vom DOHS erzeugten Zuordnungen zu nehmen. Die beiden Parameter bieten Applikationen oder Benutzer die Möglichkeit, dem DOHS-Algorithmus das Verhältnis zwischen Transfer- und Berechnungsaufwand eines Auftrags mitzuteilen. Um den Einfluss der β-Parameter zu untersuchen, wurden die 350 Simulationen mit den gleichen 35 Kombinationen der größeren α-Parameter wiederholt. Zusätzlich wurden für jeden Auftrag die Parameter β_1 zur Gewichtung des Transferaufwandes und β_2 zur Gewichtung des Berechnungsaufwandes wie folgt angepasst:

$$\beta_1 = \begin{cases} 0,5 & \text{falls Operation pro MB} > 100 \\ 1 & \text{sonst} \end{cases}$$

$$\beta_2 = \begin{cases} 0,5 & \text{falls Operation pro MB} < 0.25 \\ 0,75 & \text{falls Operation pro MB} < 2 \\ 1 & \text{sonst} \end{cases}$$

Der beste mittlere Verarbeitungsaufwand $\overline{\varphi}$ sank im Vergleich zum vorherigen Versuch mit festen β von 391217871 auf 387220587 ($\alpha_3 = 10,\ \alpha_4 = 75$). Die relativ geringe Verbesserung lässt sich auf den großen Einfluss der gemischt intensiven Aufträge in den 10 Simulationsumgebungen zurückführen. Dieser Auftragstyp benötigt sowohl einen hohen Transfer- als auch Berechnungsaufwand und es wird daher $\beta_1 = \beta_2 = 1$ gewählt.

Die durchgeführten Simulationen bestätigen die erwarteten Eigenschaften des DOHS-Algorithmus, im Bezug auf das Verhalten und den Einfluss der Parameter $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ und β_1, β_2. Der Algorithmus verhält sich, insbesondere bezüglich der Parameter α, äußerst robust, so dass der Algorithmus selbst in sehr unterschiedliche Umgebungen unverändert eingesetzt werden kann.

4.4.2 Der Verarbeitungsaufwand des DOHS

Das primäre Ziel der durchgeführten Simulationen ist die Güte der vom DOHS erzeugten Zuordnungen in realitätsnahen Umgebungen zu evaluieren. In den zuvor durchgeführten Simulationen konnten bereits die besten Parameter für die 10 unterschiedlichen Simulationsumgebungen sowie die entsprechenden Verarbeitungsaufwände ermittelt werden. Wie gut die vom DOHS erzeugten Zuordnungen sind, soll durch einen Vergleich der Verarbeitungsaufwände mit dem Online-Referenzalgorithmus 7 - im Folgenden als DK (Dauer $\cdot$ Kosten) abgekürzt - untersucht werden. Für die Untersuchung wird der DOHS mit den Parametern $\alpha_1 = 1$, $\alpha_2 = 1,5$, $\alpha_3 = 10$, $\alpha_4 = 75$ und $\beta_1 = \beta_2 = 1$ mit dem mittleren Verarbeitungsaufwand von $\overline{\varphi} = 391217871$ verwendet. Die Wahl von $\beta_1 = \beta_2 = 1$ entspricht hierbei aktuellen Umgebungen, in denen die Benutzer dem Scheduler häufig keine Informationen über die Aufträge zukommen lassen. Unter Verwendung des Online-Referenzalgorithmus DK ergab sich in den 10 Simulationen ein mittlerer Verarbei-

tungsaufwand von 249947115. Damit liegt der mittlere Verarbeitungsaufwand des DOHS ungefähr 57% über dem Wert des Referenzalgorithmus.

Abbildung 4.8 stellt den Unterschied der einzelnen Verarbeitungsaufwände der 10 Simulationen zwischen dem DOHS und dem *DK* dar. Wie aus der Abbildung

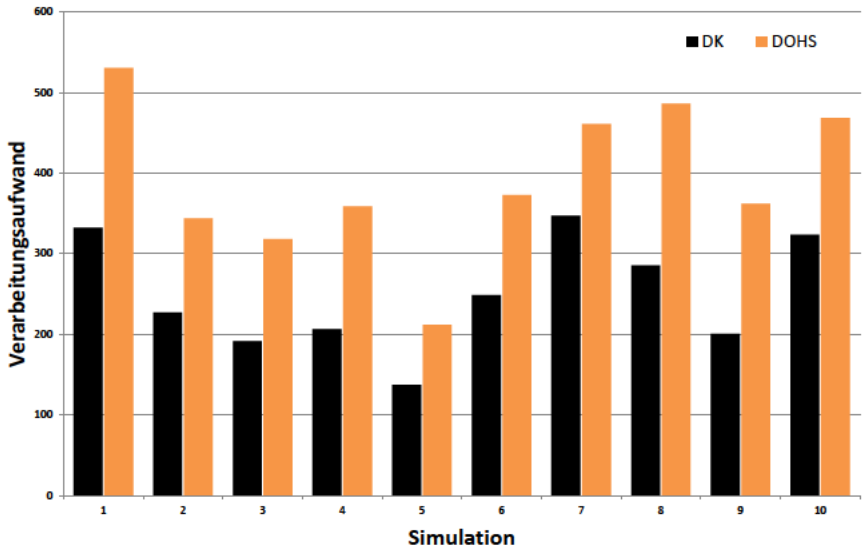


Abbildung 4.8: Vergleich des Verarbeitungsaufwands DOHS mit *DK*.

hervorgeht, sind die Unterschiede der Verarbeitungsaufwände über die Simulationen relativ ähnlich, so dass sich ein mittlerer Unterschied von 58% und eine Standardabweichung der Unterschiede von nur 14% ergeben. Dies zeigt, dass die Güte der vom DOHS erzeugten Zuordnungen weitgehend unabhängig von einer bestimmten Umgebung und Auftragszusammensetzung ist.

Der Unterschied zwischen dem Online-Referenzalgorithmus und dem DOHS lässt sich auf unterschiedliche Ursachen zurückführen. Eine Ursache sind die fehlenden Netzwerkinformationen, die dazu führen, dass bei Transfers über Cluster- oder Organisationsgrenzen hinweg nicht immer die beste Speicherressource gewählt wird. Dadurch steigen sowohl die Transferdauer als auch die Transferkosten. Die wichtigste Ursache sind jedoch die nicht vorhandenen Informationen über die Ausführungsdauer der Aufträge. Ohne diese Informationen ist es nicht möglich zu entscheiden, ob ein Auftrag auf einer freien Rechenressource mit höherem Transferaufwand ausgeführt werden soll oder ob der Auftrag auf eine aktuell belegte Rechenressource mit geringerem Transferaufwand warten soll. Die fehlenden Informationen kompensiert der DOHS durch die in Abschnitt 3.3.1 vorgestellte Funktion zur Bestimmung des Berechnungsaufwands eines Auftrags. Der DOHS versucht, mit dieser Funktion eine Balance zwischen der Nutzung freier Ressourcen und des Wartens auf belegte Ressourcen herzustellen.

Anhand der in Abbildung 4.9 dargestellten Grafiken lassen sich Rückschlüsse auf das vom DOHS erzeugte Verhältnis zwischen Warten auf belegte Ressourcen und

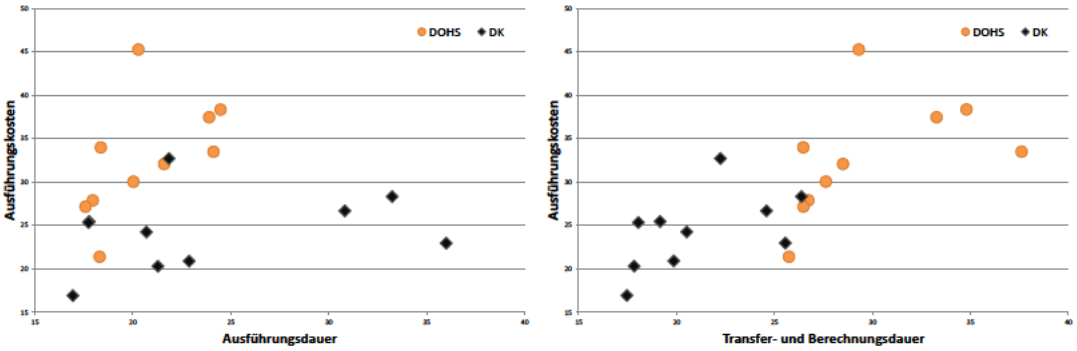


Abbildung 4.9: Vergleich der Dauer und Kosten zwischen DOHS und DK.

schneller Ausführung ziehen. Die Grafik auf der linken Seite zeigt die Summe der Ausführungsdauern und -kosten des DOHS und des DK über alle 10 Simulationen. Betrachtet man die Grafik, stellt man fest, dass die Summe der Ausführungsdauern vom DOHS häufig geringer ist als die des DK - im Mittel um 15%. Die Ausführungskosten des DOHS sind hingegen deutlich höher als die des DK. Auf der rechten Seite wird anstatt der Ausführungsdauern die Summe der Transfer- und Berechnungsdauern dargestellt. Diese liegen beim DOHS ebenfalls über den Werten des DK.
Wie aus den beiden Grafiken hervorgeht, bringt der DOHS die Aufträge im Allgemeinen schneller zur Ausführung, wobei etwas höhere Transfer- und Berechnungsdauern sowie Ausführungskosten entstehen. Das insgesamt gute Ergebnis des DOHS - nur 57% über dem Wert des Referenzalgorithmus - zeigt, dass der DOHS eine gute Balance zwischen Nutzung freier Ressourcen und des Wartens auf bessere, belegte Ressourcen findet.

4.4.3 Vergleich des DOHS mit bestehenden Schedulern

In den beiden vorhergehenden Abschnitten konnte gezeigt werden, dass der DOHS robust bezüglicher seiner Parameter sowie den Umgebungs- und Auftragszusammensetzungen ist. Ein weiterer Aspekt der Evaluierung und der durchgeführten Simulationen ist der Vergleich des DOHS-Algorithmus mit bestehenden Schedulern unter realen Bedingungen. Da in realen Umgebungen den Schedulern meist keine exakten Informationen zur Verfügung stehen, erfolgt der Vergleich des DOHS mit den bestehenden Schedulern auch im Hinblick auf den Einfluss von ungenauen Auftrags- sowie Netzwerkinformationen auf die Vergleichsscheduler.

Der DOHS wird hierbei nicht mit allen unterschiedlichen Schedulingalgorithmen verglichen, sondern ähnliche Scheduler werden, wie in Abschnitt 4.2 beschrieben, durch die allgemeinste Implementierung repräsentiert. Die heuristischen Grid-Scheduler sowie der erweiterte MapReduce-Scheduler basieren auf dem Algorithmus 7 und unterscheiden sich nur durch die verwendete Schedulingfunktion. Der Algorithmus berechnet für jeden einzelnen Auftrag die Schedulingfunktion für alle $r \cdot s^d$ möglichen Zuordnungen. Gewählt wird die Zuordnung mit minimalem Wert der

Schedulingfunktion. Es werden folgende Schedulingfunktionen für den Vergleich mit dem DOHS verwendet:

- *DK - Dauer und Kosten.* Als Repräsentanten der auf das neue Umgebungs- und Ausführungsmodell angepassten, bestehenden Grid-Scheduler wird der Online-Referenzalgorithmus $Sched(z) = \varphi(z)$ verwendet.

- *D - Dauer.* Als Vertreter der Scheduler die auf das neue Umgebungsmodell angepasst sind aber nur die Ausführungsdauer der Ausführung betrachten, wird als Schedulingfunktion $Sched(z) = \tau(z)$ verwendet.

- *K - Kosten.* Als Vertreter der Scheduler die auf das neue Umgebungsmodell angepasst sind aber nur die Kosten der Ausführung betrachten, wird als Schedulingfunktion $Sched(z) = \kappa(z)$ verwendet.

- *DKG - Dauer und Kosten im klassischen Grid.* Die bestehenden Grid-Scheduler, die auf dem klassischen Umgebungsmodell basieren, werden durch die Schedulingfunktionen $Sched(z) = \tilde{\varphi}(z)$ repräsentiert.

- *DG - Dauer im klassischen Grid.* Die bestehenden Grid-Scheduler, die auf dem klassischen Umgebungsmodell basieren und nur die Ausführungsdauer betrachten, werden durch die Schedulingfunktionen $Sched(z) = \tilde{\tau}(z)$ repräsentiert.

- *KG - Kosten im klassischen Grid.* Die bestehenden Grid-Scheduler, die auf dem klassischen Umgebungsmodell basieren und nur die Ausführungskosten betrachten, werden durch die Schedulingfunktionen $Sched(z) = \tilde{\kappa}(z)$ repräsentiert.

- *MR - MapReduce im Grid.* Für das erweiterte MapReduce-Scheduling wird die Schedulingfunktion $Sched(z) = MR(z)$ verwendet.

Des Weiteren wird ein Scheduling mit dem Genetischen Schedulingalgorithmus GA mit der Schedulingfunktion $\varphi(z)$ sowie dem einfachen, randomisierten Scheduler $RAND$ durchgeführt.

Mit exakten Informationen erzeugt der Referenzalgorithmus DK ein optimales Online-Scheduling. Mit inexakten Informationen dient der Algorithmus als Repräsentant der auf das neue Umgebungs- und Ausführungsmodell angepassten, bestehenden Grid-Scheduler. Abbildung 4.10 zeigt, wie sich die mittlere Ausführungsdauer und Ausführungskosten über die 10 Simulationen bei ungenauen Umgebungs- und Auftragsinformationen verändern. Erhält der Referenzalgorithmus nicht die exakten Netzwerkgeschwindigkeiten, sondern Netzwerkinformationen mit einem geringen Fehler, nach der Definition aus Abschnitt 4.1.1, erhöhen sich - dunkelrot dargestellt - primär nur die Ausführungskosten. Sind dem Scheduler darüber hinaus auch die benötigten Berechnungsoperationen nicht exakt bekannt, erhöhen sich - rot dargestellt - sowohl die Ausführungsdauer als auch die Ausführungskosten.

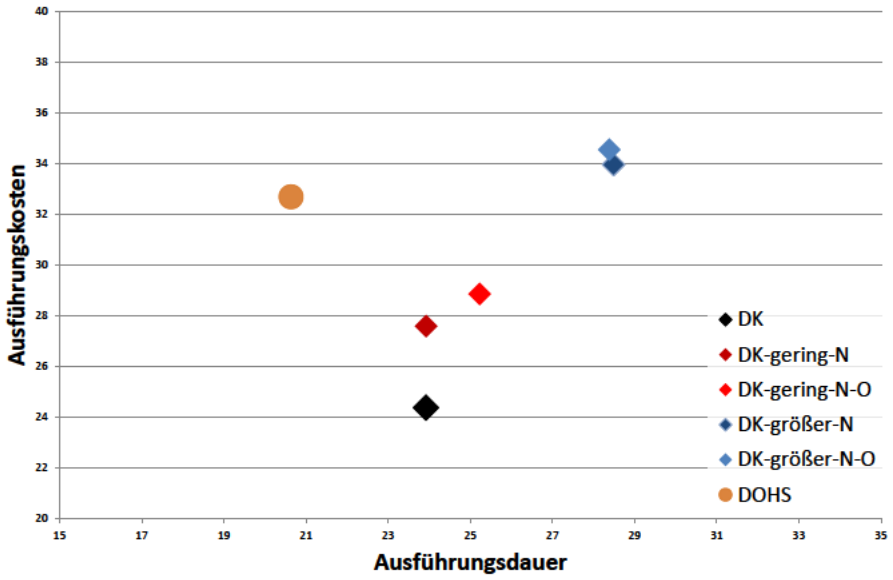


Abbildung 4.10: Einfluss der ungenauen Informationen auf den DK.

Ein größerer Fehler bei den Netzwerkinformationen - dunkelblau dargestellt - steigert die Ausführungsdauer und die Ausführungskosten erheblich. Sind zusätzlich auch nur ungenaue Informationen über die benötigten Berechnungsoperationen - blau dargestellt - vorhanden, führt dies zu einer weiteren Verschlechterung des Verarbeitungsaufwandes.

Wie aus der Abbildung hervorgeht, steigt der Verarbeitungsaufwand dieser vier unterschiedlichen Szenarien mit der Ungenauigkeit der Netzwerk- und der Auftragsinformationen. Schon ein geringer Netzwerkfehler - dem Scheduler stehen bei geringem Fehler zunächst exakte Informationen zur Verfügung - erhöht den Verarbeitungsaufwand bereits um 23% auf 306528131. Ein größerer Netzwerkfehler steigert den Verarbeitungsaufwand sogar um 115% auf 538160420. Sind zusätzlich nur ungenauen Informationen zur Anzahl der benötigten Berechnungsoperationen der Aufträge - die Berechnungsoperationen werden aus dem Intervall $[\, o_c/2 \,,\, o_c \cdot 2 \,]$ um den wahren Wert o_c geschätzt - verfügbar, erhöht sich der Verarbeitungsaufwand um weitere 10% (332249609) beziehungsweise bleibt bei größerem Netzwerkfehler annähernd unverändert (535319345).

Ungenaue Informationen haben offensichtlich einen erheblichen Einfluss auf die Güte der vom DK-Scheduler erzeugten Zuordnungen. Dies gilt, wie aus Abbildung 4.11 hervorgeht, für alle Scheduler, die rein auf Basis dieser Informationen die Aufträge auf die Ressourcen verteilen. Die Abbildung zeigt für jeden Vergleichsscheduler fünf Säulen (von links nach rechts): (1) exakte Informationen; (2) geringer Netzwerkfehler; (3) geringer Netzwerkfehler und fehlerhafte Berechnungsoperationen; (4) größerer Netzwerkfehler; (5) größerer Netzwerkfehler und fehlerhafte Berechnungsoperationen. Wie die Werte des Genetischen Algorithmus GA zeigen, ändert das eingesetzte Schedulingverfahren nichts an diesem Einfluss.

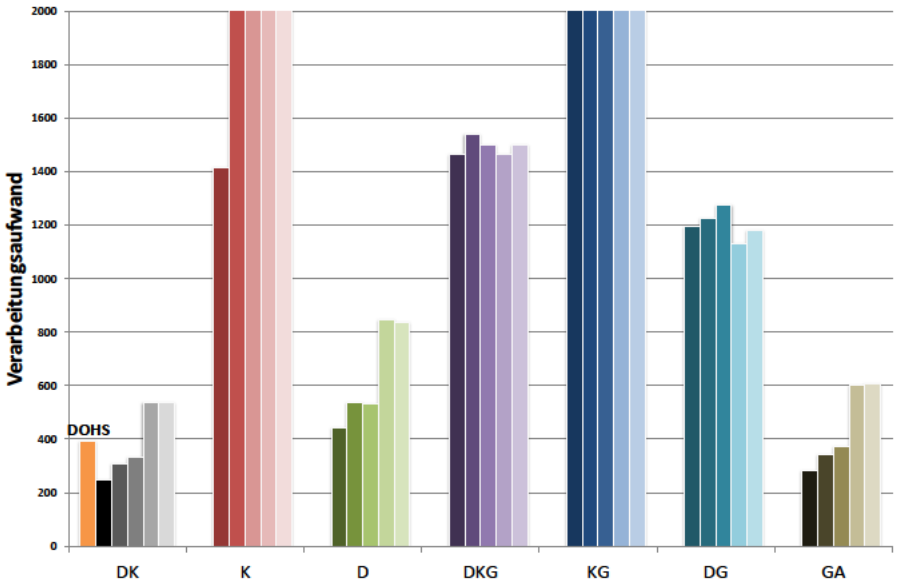


Abbildung 4.11: Veränderung des Verarbeitungsaufwandes durch fehlerhafte Informationen.

Das einem Scheduler zugrunde liegende Umgebungsmodell hat, wie die Werte des DKG, KG, DG, verdeutlichen, ebenfalls erhebliche Auswirkungen auf die Güte der erzeugten Zuordnungen.

Vergleicht man den Verarbeitungsaufwand des DOHS mit den Werten der besten Vergleichsscheduler DK und GA stellt sich heraus, dass der DOHS schon bei kleineren Fehlern der Netzwerk- und Auftragsinformationen vergleichbar gute Ergebnisse erzielt. Eine so genaue Schätzung der Netzwerkinformationen (erst nach 1000 Sekunden kann die geschätzte Geschwindigkeit abweichen; die Kosten sind immer exakt verfügbar) ist in realen Umgebungen jedoch äußerst schwierig und dürfte nur in seltenen Fällen möglich sein. Darüber hinaus ist die Schätzung der Anzahl der Berechnungsoperationen für einen Auftrag (die Schätzung unterscheidet sich um maximal einen Faktor 2 vom exakten Wert) im Allgemeinen nicht so genau möglich, wie in den Simulationen angenommen. Betrachtet man den Verarbeitungsaufwand der Vergleichsscheduler mit größeren Netzwerkfehlern, die realen Umgebungen eher entsprechen, liegen diese sogar deutlich über dem Verarbeitungsaufwand des DOHS.

Der Verarbeitungsaufwand der ähnlichen Vergleichsscheduler K, D, DKG, KG und DG liegt selbst bei genauen Umgebungsinformationen über dem Wert des DOHS. Bei den Schedulern K und D kann dies auf die einseitige Zielfunktion zurückgeführt werden, so dass der mittlere Verarbeitungsaufwand 460% (1413418264) beziehungsweise 77% (444575541) über dem Referenzalgorithmus liegt. Die für einen niedrigen Verarbeitungsaufwand nötige Balance aus Ausführungsdauer und Ausführungskosten kann mit den einfachen Zielfunktionen nicht erreicht werden. Dies verdeutlicht auch, dass bestehende Scheduler, die eine solche Balance durch die Möglichkeit der Auswahl zwischen einem Scheduling nach Ausführungsdauer *oder* Aus-

führungskosten erreichen möchten, dies nur sehr begrenzt umsetzen können. Bei den Schedulern DKG, KG und DG zeigt sich, wie ein nicht geeignetes Umgebungsmodell die effiziente Zuordnung der Aufträge auf die Ressourcen erschwert. Der Verarbeitungsaufwand liegt selbst bei exakten Umgebungsinformationen 500% (1506784851, 1480761022, 1479133984) über dem Referenzalgorithmus. Hierbei ist nicht nur die Vernachlässigung des Netzwerks innerhalb einer Organisation problematisch, sondern auch die Annahme, dass nur strikt getrennte Rechen- und Speicherressourcen existieren die zu virtuellen Ressourcen zusammengefasst werden können.

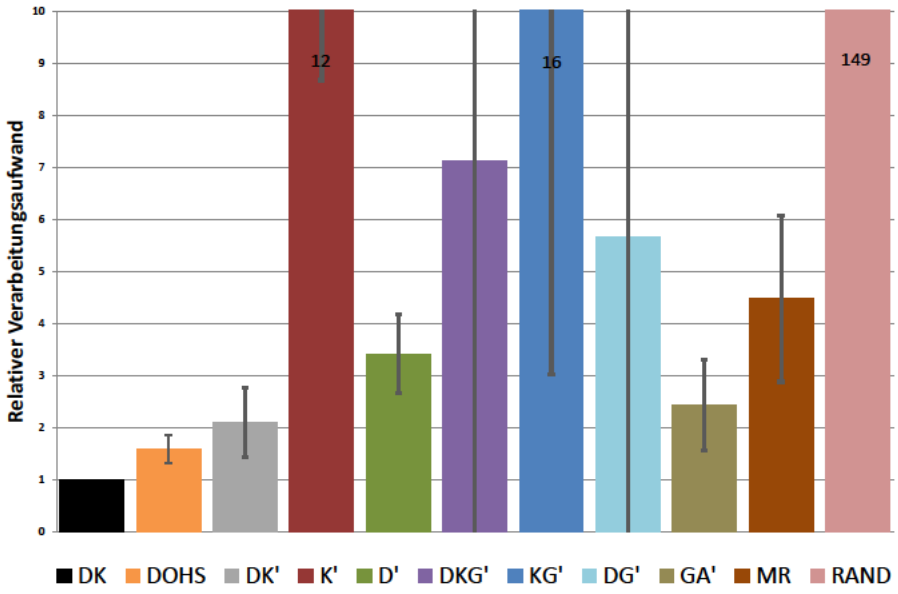


Abbildung 4.12: Mittlerer relativer Verarbeitungsaufwand mit entsprechendem Konfidenzintervall.

Neben dem DOHS erzeugt auch der erweiterte MapReduce-Scheduler MR gute Zuordnungen, ohne dabei auf die ungenauen Umgebungs- und Auftragsinformationen angewiesen zu sein. Der mittlere Verarbeitungsaufwand des MR-Schedulers in den 10 Simulationen liegt nur 331% (1077744713) über dem Wert des Referenzalgorithmus und 175% über dem DOHS. Die Variation der Parameter zwischen $\mu_1 = 1, \mu_2 \in [2; 10], \mu_3 \in [3; 100], \mu_4 \in [4; 1000]$ resultierte in einer leichten Schwankung des Verarbeitungsaufwandes zwischen 1243188518 und 1077744713. Der relativ geringe Verarbeitungsaufwand des MR-Schedulers lässt sich unter anderem auf den hohen Anteil an datenintensiven Aufträgen - 30% datenintensive und 50% gemischt intensive Aufträge - zurückführen. In Umgebungen mit einem höheren Anteil an rechenintensiven Aufträgen kann die Vernachlässigung der Rechenressourcen beim Scheduling allerdings zu deutlich schlechteren Ergebnissen führen.

Mit den durchgeführten Simulationen kann weiterhin gezeigt werden, dass eine rein randomisierte Zuordnung *Rand* von Aufträgen mit Datensätzen keine zufriedenstellenden Ergebnisse erzielt. Der mittlere Verarbeitungsaufwand der so erzeugten Zuordnungen liegt beim 156-fachen des Referenzalgorithmus.

Abbildung 4.12 stellt die mittlere Abweichung der Vergleichsscheduler vom Referenzalgorithmus über die 10 Simulationen sowie die Breite der entsprechenden Konfidenzintervalle dar. Für einen realitätsnahen Vergleich mit dem DOHS basieren die dargestellten Werte auf größeren Netzwerkfehlern sowie fehlerhaften Berechnungsaufwänden - gekennzeichnet durch ein „′“ nach dem Bezeichner des Vergleichsschedulers. Diese Darstellung veranschaulicht den Einfluss der unterschiedlichen Umgebungs- und Auftragsstrukturen auf das Schedulingergebnis. Aus der geringen Breite der Konfidenzintervalle von DK' (Konf. $0,67$), D' (Konf. $0,76$) sowie GA' (Konf. $0,87$) zeigt sich, dass der Verarbeitungsaufwand, der auf die Umgebung angepassten Scheduler, weitgehend unabhängig von der Umgebung ist. Einzig der Scheduler K' scheint negativ auf bestimmte Eigenschaften der Umgebung und der Aufträge zu reagieren.
Die Scheduler auf Basis des klassischen Grid-Umgebungsmodells DKG' (Konf. $4,2$), KG' (Konf. $5,1$) sowie DG' (Konf. $4,5$) zeigen die Abhängigkeit von diesem Modell in den stark unterschiedlichen Abweichungen vom Referenzscheduler. Bei genauerer Betrachtung erzeugen die Vergleichsscheduler für die Simulationen, die eine Umgebung nach dem klassischen Grid-Modell besitzen (Simulationsläufe 1, 6, 7 und 10), Zuordnungen mit einer deutlich geringeren Abweichung. Für die anderen Umgebungen steigt die Abweichung jedoch erheblich.
Im Vergleich zu den anderen Schedulern zeigt der DOHS mit einer mittleren Abweichung von nur $1,58$ und einer Konfidenzintervallbreite von $0,26$, dass der Algorithmus für unterschiedliche Umgebungs- und Auftragsstrukturen geeignet ist.

Die Art der Nutzung der Netzwerkressourcen ist eines der zentralen Unterscheidungsmerkmale zwischen den Schedulern. Die unterschiedlichen Strategien der Netzwerknutzung der Scheduler soll im Folgenden durch die von den Schedulern in der 9. Simulation erzeugten Netzwerkauslastungen veranschaulicht werden. Die Abbildungen 4.13, 4.14, 4.15 und 4.16 zeigen die Nutzung der einzelnen Netzwerkverbindungen des DK, des DK mit ungenauen Netzwerkinformationen, des DKG mit ungenauen Netzwerkinformationen sowie des DOHS. In den Abbildungen werden die Verbindungen innerhalb eines Clusters grün dargestellt. Zwischen Clustern einer Organisation sind die Verbindungen blau und zwischen Organisationen orange. Die Breite einer Linie spiegelt die angegebene Nutzung wieder, wobei eine breite Linie einer starken Nutzung entspricht. Ist eine Ressource gelb markiert, erfolgte der Zugriff auf die Daten lokal.

Aus den Abbildungen wird einerseits die Veränderung der vom Scheduler erzeugten Netzwerkauslastung unter ungenauen Netzwerkinformationen ersichtlich, andererseits zeigen die Abbildungen die Auswirkung eines unpassenden Umgebungsmodells. Die Strategie des DOHS, möglichst freie Rechenressourcen zu nutzen und die Transferdistanz gering zu halten, lässt sich an den in Abbildung 4.16 dargestellten Netzwerkauslastungen gut nachvollziehen.

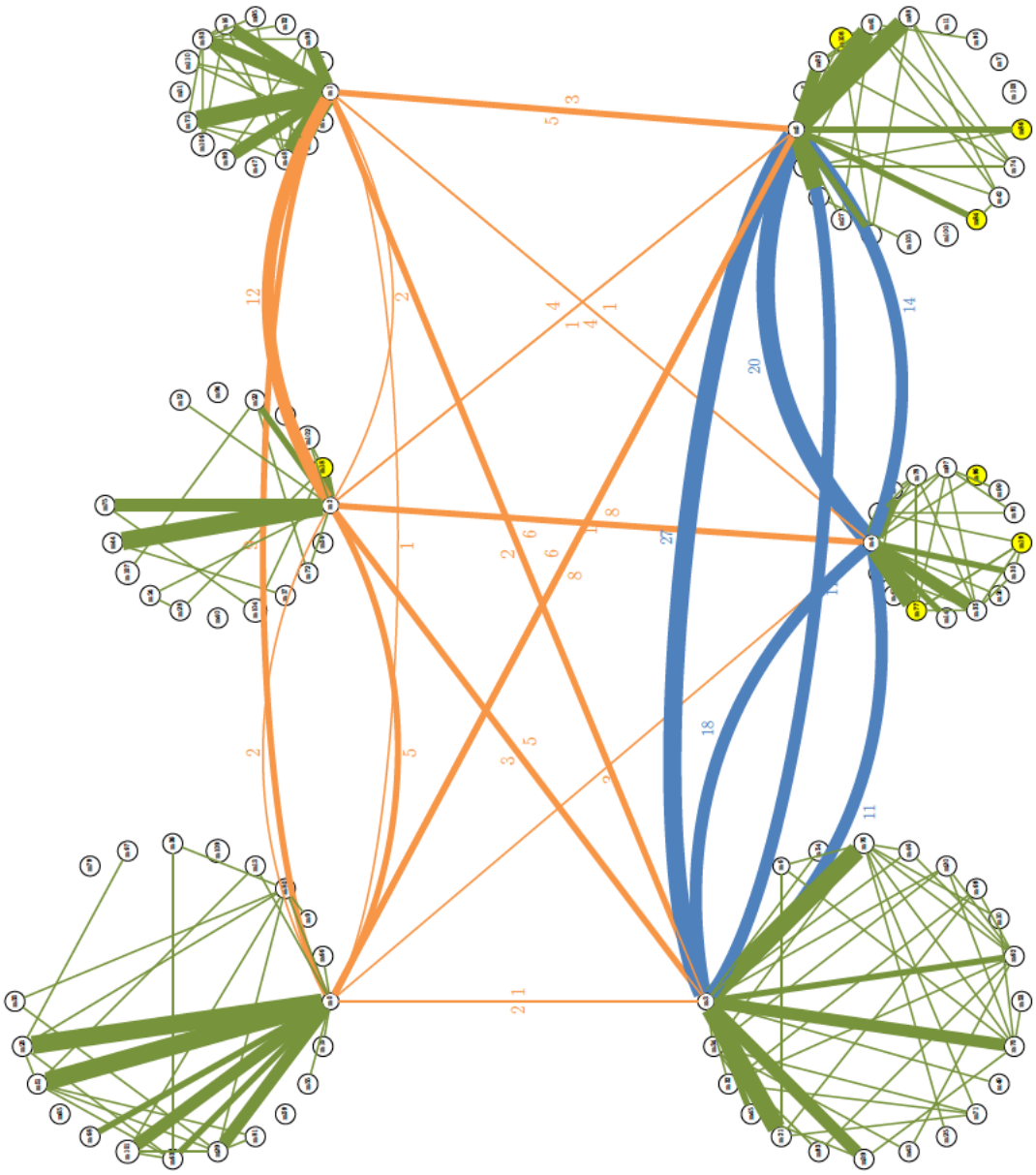


Abbildung 4.13: Netzwerknutzung des *DK* am Beispiel der 9. Simulation.

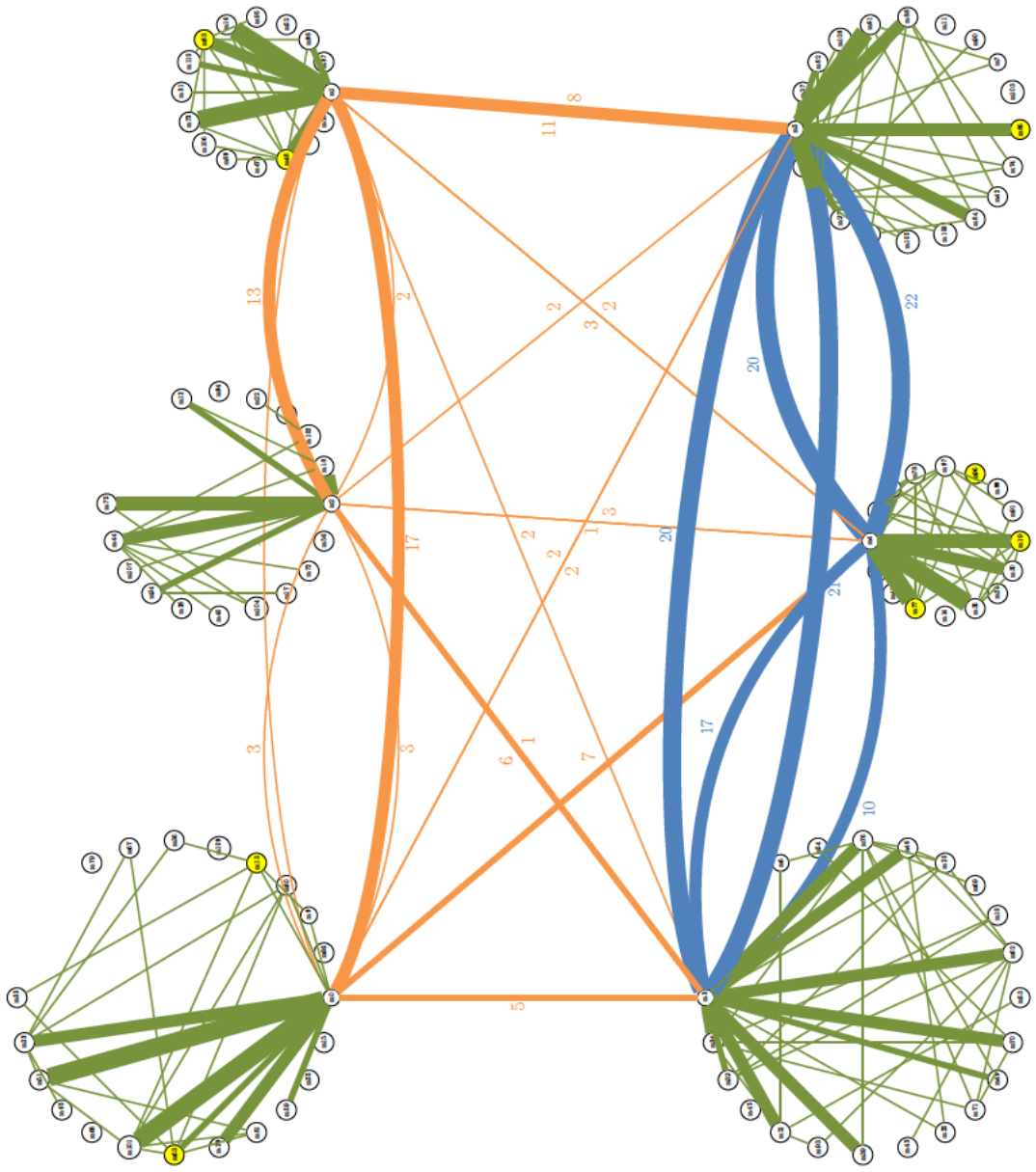


Abbildung 4.14: Netzwerknutzung des *DK* mit Fehler am Beispiel der 9. Simulation.

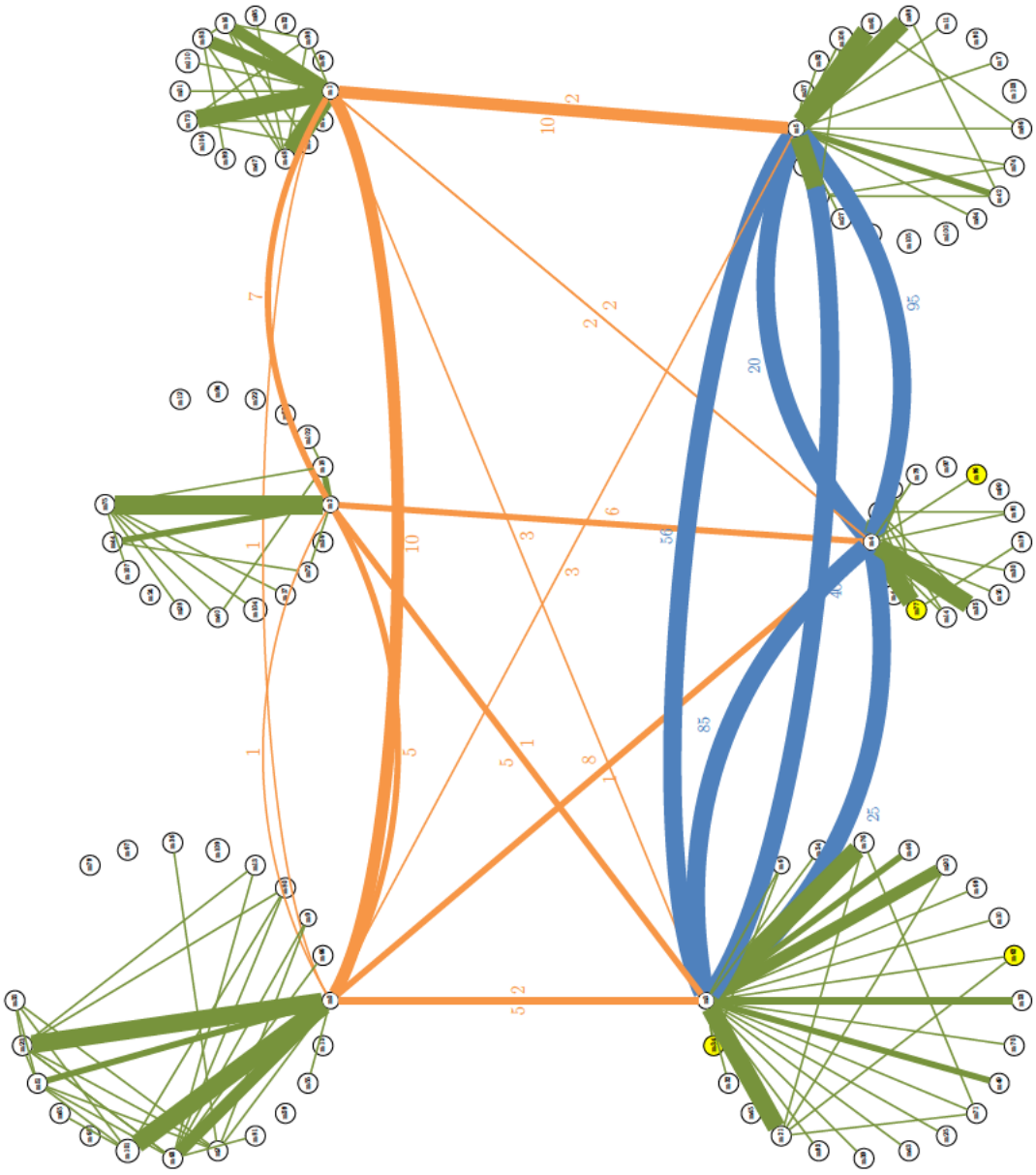


Abbildung 4.15: Netzwerknutzung des *DKG* mit Fehler am Beispiel der 9. Simulation.

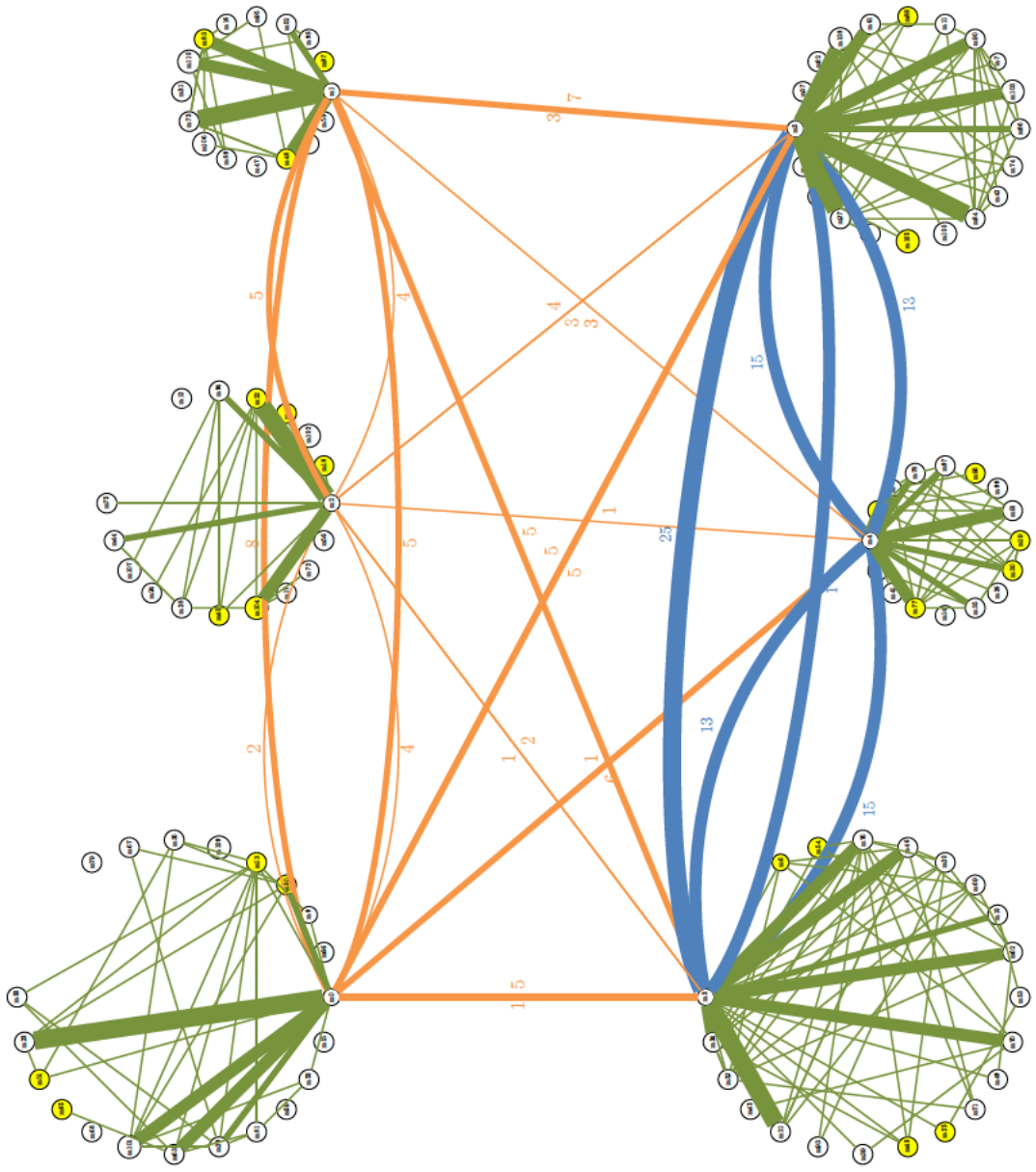


Abbildung 4.16: Netzwerknutzung des DOHS am Beispiel der 9. Simulation.

Kapitel 5

Der FDA-Miner

Das absehbare Ende der fossilen Brennstoffvorräte und die weltweit steigende Umweltbelastung durch Fahrzeugemissionen machen die Erforschung alternativer Antriebskonzepte zu einem der wichtigsten Themenfelder im Automobilbereich. Die Daimler AG arbeitet seit über 15 Jahren an der Brennstoffzellentechnologie und hat mit mehr als 100 Brennstoffzellenfahrzeugen die größte Flotte an emissionslosen Fahrzeugen weltweit [FFPW99] [Tru03]. Die Fahrzeuge der Fuel-Cell (F-Cell) Flotte werden von ausgewählten Kunden unter verschiedenen klimatischen Einflüssen sowie unterschiedlichen Verkehrs- und Nutzungsbedingungen betrieben. Bis heute wurden von der Flotte über vier Millionen Kilometer zurückgelegt. Die aus diesem Feldtest gewonnen Informationen helfen bei der Weiterentwicklung der Brennstoffzellenfahrzeuge und der benötigten Infrastruktur. Ein zentrales Ziel dieser Bemühungen ist dabei die Ermittlung des Alterungszustands und der Alterungsursachen der Brennstoffzellensysteme beim Einsatz in Fahrzeugen. Um für die Entwicklung der nächsten Generationen von Brennstoffzellenfahrzeugen möglichst viele Erfahrungen zu sammeln, wurde das Fleet Data Acquisition (FDA) System zur Aufzeichnung aller relevanten Informationen entworfen. Alle im Feld befindlichen Fahrzeuge zeichnen hierfür während der Fahrt eine Vielzahl an Sensorinformationen auf und übermitteln diese am Ende des Tages an einen lokalen Server.

Zur Aufbereitung und Analyse dieser Daten wurde im Rahmen dieser Arbeit der *FDA-Miner*[FSN+08] entwickelt. Der FDA-Miner implementiert eine umfassende Analyseumgebung zur Verarbeitung organisationsübergreifend, global verteilter Daten auf Basis aktueller Grid-Standards. Durch den Einsatz des in den vorherigen Kapiteln vorgestellten DOHS-Algorithmus, ist der FDA-Miner in der Lage, die zur Analyse erforderlichen daten- und rechenintensiven Data-Mining Anwendungen effizient auszuführen.

Der nachfolgende Abschnitt stellt zunächst das Fleet Data Acquisition System vor. Anschließend wird das im Rahmen dieser Arbeit entworfenen Konzepte zur Integration und Ausführung von organisationsübergreifenden Data-Mining Anwendungen[SSK+08] vorgestellt. Darauf folgend wird die Architektur und die Details der zentralen Komponenten des FDA-Miners für das Scheduling und die Ausführung von ressourcenintensiven Anwendungen beschrieben. Im Anschluss wird die

Benutzerschnittstelle des FDA-Miners vorgestellt, die den Benutzern unter anderem die Erzeugung individueller Reports und die Analyse der Fahrtdaten mit beliebigen Analyseverfahren ermöglicht. Das Kapitel endet mit dem Vergleich der Verarbeitungsleistung häufig verwendeter Analyseverfahren im FDA-Miner beim Scheduling mit dem DOHS und mit bestehenden Grid-Schedulern.

5.1 Das Fleet Data Acquisition System

Die F-Cell Flotte dient primär dem Test der Brennstoffzellenfahrzeuge im realen Kundeneinsatz und der Informationsgewinnung für die Entwicklung neuer Modelle. Um eine möglichst breite Palette an Umgebungsbedingungen, wie Klimazonen, Verkehrsaufkommen und Fahrverhalten, abzudecken, werden die Fahrzeuge bei Kunden in den USA (Californien und Michigan), Japan, Singapur und Deutschland eingesetzt. Die beim Einsatz der Brennstoffzellenfahrzeuge anfallenden Daten sind wichtige Informationsquellen für die Weiterentwicklung der Brennstoffzellentechnologie. Für die durchgängige Aufzeichnung und Speicherung aller relevanten Daten wurde

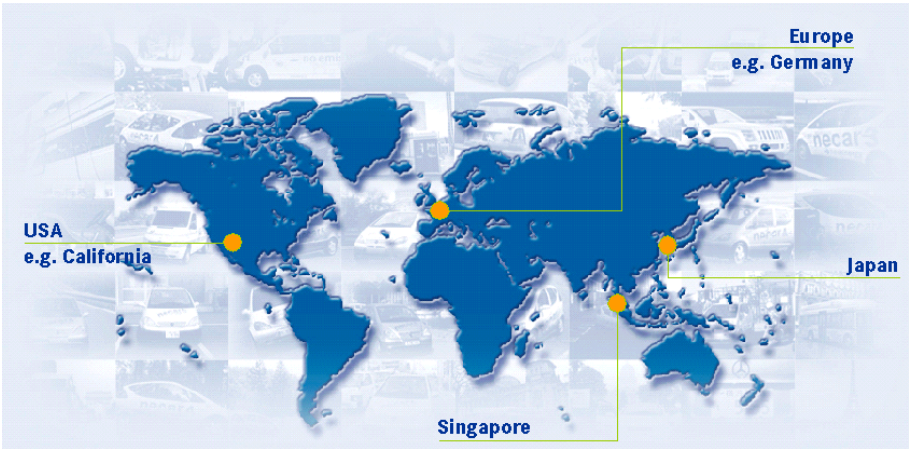


Abbildung 5.1: Die Einsatzorte der F-Cell Flotte: USA (Californien und Michigan), Japan, Singapur und Deutschland [NSW+05].

das FDA-System[WFNR05] entwickelt. Auf Basis dieser Daten können Ingenieure den aktuellen Status der Fahrzeuge sowie den Einfluss unterschiedlicher Umgebungsparameter auf das Brennstoffzellensystem bestimmen.

Das FDA-System besteht im Wesentlichen aus zwei Komponenten: einem Datenaufzeichnungssystem im Fahrzeug und einem Datenverwaltungssystem. Jedes Brennstoffzellenfahrzeug der F-Cell Flotte ist mit einem eigens entwickelten Aufzeichnungsgerät ausgestattet. Das Aufzeichnungsgerät überwacht das „Controller Area Network“ des Fahrzeugs und speichert alle Nachrichten und Signale in einem Ringpuffer. Die Nachrichten und Signalen enthalten sowohl Diagnose- als auch Betriebsinformationen der im Fahrzeug enthaltenen Sensoren. Benutzerdefinierte Filter

speichern die ausgewählten Nachrichten und Signale in einem internen, persistenten Speicher. Für jede Fahrt wird eine neue Datei auf dem Speichermedium erzeugt. Die im internen Speicher des Aufzeichnungsgerätes abgelegten Fahrtdaten werden regelmäßig ausgelesen. Das Aufzeichnungsgerät überträgt die Daten über W-LAN an Speicherressourcen der lokalen Servicestationen. Die Daten werden zu einem späteren Zeitpunkt zusätzlich an einen zentralen FDA-Server gesendet. Dieser speichert die Daten aller Fahrzeuge und stellt den Benutzern eine Schnittstelle zum Herunterladen der Daten zur Verfügung.

Die vom FDA-System vorgehaltenen Daten wurden zu Beginn des Projekts von den Ingenieuren manuell heruntergeladen und analysiert. Die Analyse beschränkte sich meist auf einzelne Fahrten oder kleinere Aufzeichnungszeiträume eines Fahrzeugs. Komplexere Analysen, wie etwa die Auswertung der gesamten Daten eines Fahrzeugs oder der ganzen Flotte, konnten jedoch nicht durchgeführt werden, da diese von einem einzelnen Arbeitsplatzrechner nicht in angemessener Zeit bearbeitet werden können. Durch die Verzögerung des Transfers zum zentralen Server konnten darüber hinaus keine zeitnahen Analysen durchgeführt werden, um etwa kritische Systemstatus oder Systemschäden vorzubeugen.

Die Masse an Daten und die häufig identischen Vorverarbeitungsschritte bei der Analyse führten zur Entwicklung des ersten FDA-Miner Prototypen. Im Gegensatz zum bis dahin genutzten FDA-Server, der aufgrund der Datenmengen nur die Daten der letzten drei Monate vorhalten konnte, sollte der FDA-Miner die Daten aller Fahrzeuge über den *gesamten* Lebenszyklus der Fahrzeuge *kostengünstig* bereitstellen. Die erste Version des FDA-Miners speicherte die Fahrtdaten aller Fahrzeuge auf einem zentralen Server, dem *Manager*. Alle neuen Daten wurden regelmäßig vom FDA-Server heruntergeladen, überprüft und gespeichert. Die Web-Benutzeroberfläche bot den Benutzer aktuelle Informationen zu den Fahrzeugen sowie eine Zusammenfassung einzelner Fahrtdatensätze. Über die Oberfläche konnten die Benutzer die Fahrtdaten eines oder mehrerer Fahrzeuge auswählen, mit vorgegebenen Programmen verarbeiten und die Resultate herunterladen. Die Verarbeitung der Daten erfolgte auf mehreren dedizierten *Rechenknoten*. Der Manager transferierte die zu verarbeitenden Daten vor der Ausführung an die Rechenknoten. Diese wendeten, die vom Benutzer ausgewählten Programme, nacheinander auf die Daten an und sendeten die Ergebnisse im CSV-Format an den Manager zurück. Die einzelnen Ergebnisse wurden vom Manager zusammengefasst und dem Benutzer bereitgestellt. Die so vorverarbeiteten Fahrtdaten wurden anschließend von den Benutzern mit unterschiedlichen Programmen (Excel, Matlab, ...) manuell ausgewertet.

Mit zunehmenden Benutzerzahlen, steigendem Datenaufkommen und immer komplexeren Verarbeitungsmethoden stieß der Prototyp schnell an seine Grenzen, so dass im Rahmen dieser Arbeit der FDA-Miner von Grund auf neu entworfen und implementiert wurde. Anstatt eines rein zentralen Systemes sollte die Verwaltung und Auswertung der Daten auch auf den lokalen Servern der einzelnen Standorte der FDA-Flotte vor Ort durchgeführt werden können. Mit diesem Ansatz können die Daten zeitnah, vor Ort verarbeitet werden, um kritische Systemstatus oder Systemschäden vorzubeugen. Darüber hinaus wäre es möglich auf eine Übertragung

und zentrale Speicherung der Daten komplett zu verzichten, so dass die Kosten für ein zentrales Data-Warehouse eingespart werden könnten. Dies ist auch im Hinblick auf die Integration von Daten weiterer Versuchsflotten, die von unterschiedlichen Organisationseinheiten betrieben werden, eine effiziente und skalierbare Alternative. Zusätzlich zur Speicherung und Verwaltung der Daten, muss der FDA-Miner diese auch effizient verarbeiten, wobei eine Betrachtung der eingesetzten Verarbeitungsszenarien für die Anforderungsanalyse äußerst hilfreich ist. Im FDA-Miner und ähnlichen Analysesystemen können allgemein drei wesentliche Szenarien mit unterschiedlichen Verarbeitungsprofilen unterscheiden werden:

- *Filtern.* Beim Filtern werden sehr große Datenmengen nach vordefinierten Regeln durchsucht. Die Rechenintensität dieses Vorgangs ist meist gering, so dass eine Übertragung der Eingabedaten die Verarbeitungsdauer erheblich beeinflusst.

- *Analysieren.* Bei der Analyse werden Modelle aus mittleren bis großen Datenmengen berechnet. Dieser Vorgang ist ebenso daten- wie rechenintensive, so dass die Verarbeitungsdauer durch eine Vermeidung der Datenübertragung deutlich gesenkt werden kann und eine Übertragung der Eingabedaten vermieden werden sollte.

- *Simulationen.* Simulationen benötigen meist nur kleine Eingabedaten erzeugen dafür jedoch sehr hohe Rechenlasten. Durch die Auswahl geeigneter Rechenressource kann die Simulation deutlich beschleunigt werden.

Bei den Anwendungen die aktuell im FDA-Miner ausgeführt werden, liegt der Fokus auf dem datenintensiven Filtern und Analysieren der Fahrtdaten. Häufig werden diese beiden Funktionen nacheinander angewendet. Im ersten Schritt werden alle für die nachfolgende Analyse benötigten Datensätze aus den Fahrtdaten gefiltert. Die so aufbereiteten Daten bilden die Eingabe der Analyseprogramme des zweiten Verarbeitungsschrittes. Weitere Schritte, etwa zur grafischen Aufbereitung oder zum Vergleich mehrerer Analyseergebnisse, können folgen. Obwohl der FDA-Miner hauptsächlich für datenintensive Anwendungen genutzt wird, werden auch Simulationen durchgeführt.

Durch den Einsatz der im Rahmen dieser Arbeit entwickelten Verfahren im FDA-Miner konnte einerseits die Benutzerfreundlichkeit erhöht und andererseits die Verarbeitungsdauer erheblich gesenkt werden.

5.2 Implementierungsaspekte des FDA-Miners

Die Analyse der Daten im FDA-Miner erfolgt durch unterschiedliche Data-Mining Applikationen, die oftmals eine große Anzahl an Rechen- und Speicherressourcen benötigen. Die neueste Generation von Grid-Systemen eigenen sich durch ihre organiststationsübergreifende und auf globale Vernetzung ausgelegte Kommunikations-

und Verwaltungsinfrastruktur ideal für die Umsetzung des FDA-Miners. Das Globus Toolkit (GT)[1] stellt ab Version 4 bereits spezielle Dienste für die Verwaltung von Rechen- und Speicherressourcen zur Verfügung, was GT 4 als Basis für ein globales Data-Mining System prädestiniert.

Data-Mining und andere Applikationen lassen sich prinzipiell auf zwei unterschiedliche Arten in ein Grid-Systemen integrieren. Eine in der Literatur wie auch in der Praxis häufig eingesetzte Methode ist die Implementierung anwendungsspezifischer Grid-Services[AT05][CTT07][CTT03]. Jede Applikation wird hierbei durch einen Grid-Service repräsentiert. Obwohl sich damit schnell Data-Mining Applikationen in ein Grid-System einbinden und einfach kombinieren lassen, hat diese Vorgehensweise den entscheidenden Nachteil, dass nur die lokalen Ressourcen eines Computers zur Verfügung stehen. Wenn gleichzeitig mehrere rechen- und/oder datenintensive Programme ausgeführt werden, kann dies daher zum Stillstand der Ressource führen, während andere Ressourcen nicht ausgelastet sind.

Für Applikationen mit langlaufenden und ressourcenintensiven Programmen, wie im Data-Mining üblich, bietet es sich an, für die Ausführung der Programme alle Rechenressourcen im Grid zu nutzen. Dies erzeugt zwar einen zusätzlichen Verwaltungsaufwand, erlaubt aber beim Einsatz eines geeigneten Schedulers, die effiziente Ausnutzung aller im Grid vorhandenen Ressourcen. Damit, wie im FDA-Miner erforderlich, beliebige Data-Mining Programme durch die Grid-Ausführungsschicht ausgeführt werden können, muss das System eine generische Schnittstelle zur Integration dieser Programme bereitstellen. Da nicht nur die Ressourcen, sondern auch die Benutzer über viele Standorte verteilt sein können, sind nicht alle Applikationen und Daten jedem Benutzer bekannt. Damit Data-Mining Programme auch organisationübergreifend von unterschiedlichen Benutzern eingesetzt werden können, wird darüber hinaus eine Data-Mining-spezifische Beschreibung der Applikationen benötigt, die es unter anderem ermöglicht, anhand bestimmter Kategorien, wie Clustering oder Klassifikation, nach geeigneten Applikationen zu suchen.

Aus dieser Diskussion lassen sich die zentralen Komponenten und ihre Eigenschaften ableiten, die für die Realisierung des FDA-Miners benötigt werden:

- *Applikationsverwaltung.* Die Grundlage der Applikationsverwaltung bildet die in dieser Arbeit entwickelte Applikationsbeschreibung[SSK+08] mit der sich beliebige Data-Mining Programme in ein Grid integrieren lassen. Die Applikationsbeschreibung ermöglicht einerseits geeignete Data-Mining Verfahren über ihre Eigenschaften auszuwählen ohne dabei die Implementierungsdetails oder die einzelnen Komponenten (Programm, Bibliotheken,...) kennen zu müssen. Andererseits bietet die entwickelte Applikationsbeschreibung auch die Möglichkeit die zu verarbeitenden Datensätze über ihre Eigenschaften zu spezifizieren, ohne den Speicherort oder den Dateinamen angeben zu müssen.

- *Informationsverwaltung.* Die Informationsverwaltung integriert die Zustandsinformationen aller Ressourcen und stellt diese den anderen Komponenten zur

[1] http://www.globus.org

Verfügung. Das „Monitoring and Discovery System“ (MDS)[Fit01] des Globus Toolkit bildet die zentrale Informationsquelle für die Zustandsinformationen der Ressourcen. Die zweite wichtige Informationsquelle bildet die im Rahmen dieser Arbeit entwickelte Datenverwaltung, in der die Speicherorte aller Datensätze sowie weitere Zusatzinformationen über die Daten verwaltet werden. Die Datenverwaltung verwaltet nur die Meta-Daten der Datensätze und überlässt die eigentliche Datenspeicherung den Datenbanken oder Dateisystemen.

- *Ausführungsverwaltung.* Der in dieser Arbeit entwickelte DOHS-Algorithmus bildet die Basis der Ausführungsverwaltung und ordnet die Verarbeitungsaufträge des FDA-Miners auf die zur Verfügung stehenden Ressourcen zu. Der DOHS nutzt die Informationsverwaltung für die Abfrage des aktuellen Zustands der Ressourcen sowie zur Lokalisation der Replika der zu verarbeitenden Datensätze. Die entwickelte Ausführungskomponente nutzt die Funktionen von Condor-G[FTL+02] und des Globus Toolkit, um die vom Scheduler erzeugten Zuordnungen auszuführen.

Die in Abbildung 5.1 dargestellte Architektur zeigt, wie sich diese zentralen Komponenten in das Gesamtsystem einfügen. Die Architektur des FDA-Miners besteht aus fünf aufeinander aufbauenden Schichten, die über anwendungsunabhängige Schnittstellen miteinander kommunizieren.

1. *Applikationsschicht.* Die oberste Schicht bildet die Schnittstelle zu den Benutzern des Systems. Eine intuitive Bedienoberfläche ist essenziell für die Akzeptanz der Benutzer. Durch die offenen Schnittstellen und die generische Applikationsbeschreibung lassen sich beliebige Benutzerschnittstellen implementieren. Die Applikationsbeschreibung bildet auch die zentrale Kommunikationseinheit zwischen den Komponenten der Applikationsschicht und den Systemdiensten.

2. *Systemdienste.* Diese Schicht bildet das Rückgrat des Systems und besteht im Wesentlichen aus drei Komponenten: der Applikationsverwaltung, der Datenverwaltung und der Ausführungsverwaltung.
   Die in Abschnitt 5.4 erläuterte Datenverwaltung speichert die Meta-Daten der im System vorhandenen Daten und stellt Schnittstellen für die Verwaltung dieser Informationen zur Verfügung. Die globale Verwaltung und Ausführung von Aufträgen übernimmt die Ausführungsverwaltung. Der Aufbau und die Funktionsweise der Ausführungsverwaltung werden in Abschnitt 5.5 vorgestellt.
   Die Verwaltung der im System vorhandenen Programme und Applikationen übernimmt die Applikationsverwaltung. Diese bietet Funktionen zur Erstellung von ADS-Dokumenten und zur Integration ins Grid.

3. *Grid-Middleware.* Das Globus Toolkit 4 bildet die Basis der Systemdienste. GT 4 stellt die Kommunikationsinfrastruktur für die Vernetzung der unterschiedlichen Organisationen bereit, verwaltet die Ressourcen der einzelnen Organisationen und bietet eine Reihe von Basisdiensten für die Sicherheit, das Informationsmanagement (MDS), die Ausführungs- (WS-GRAM) und die Datenverwaltung (Reliable File Transfer, RFT) [FKTT98][FFM07].

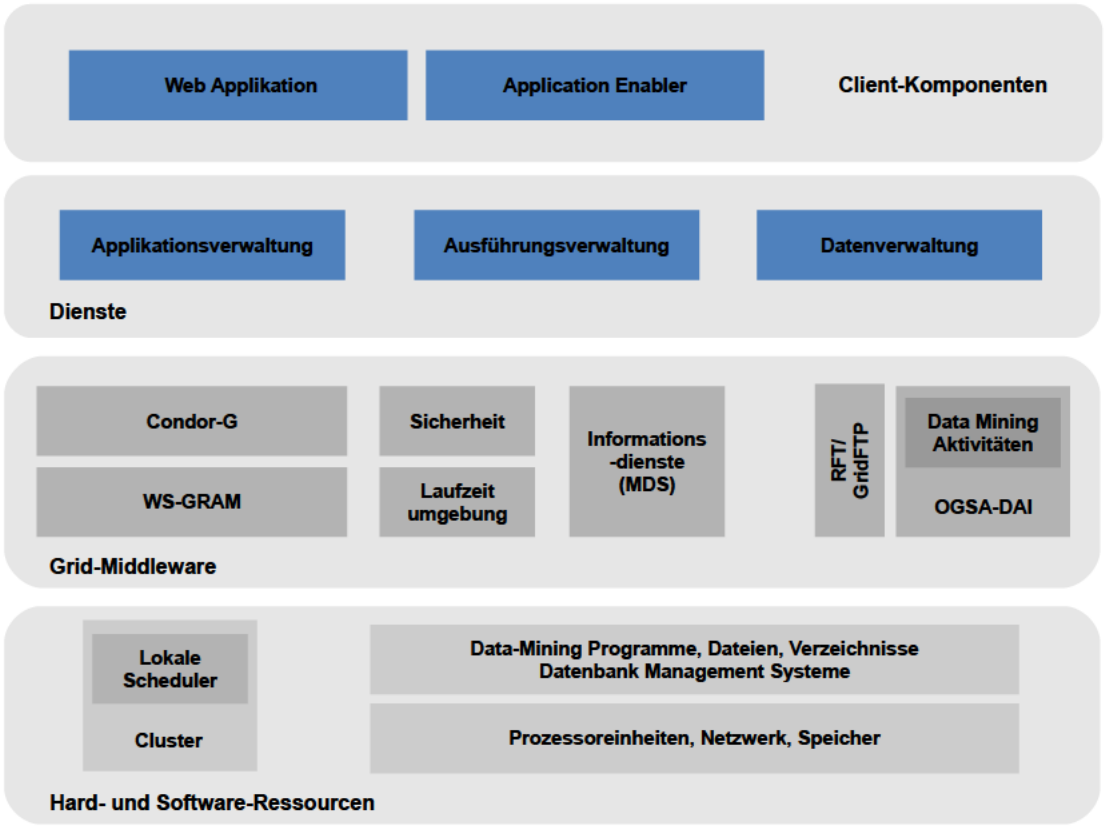


Abbildung 5.2: Die Architektur des Analysesystems.

4. *Clusterverwaltungssysteme.* Die Verwaltung der lokalen Cluster erfolgt durch spezialisierte Clusterverwaltungssysteme. Diese verwalten die Ausführung von Programmen auf mehreren lokalen Ressourcen einer Organisation. Alle Clusterverwaltungssysteme einer Organisation werden vom zugehörigen WS-GRAM gesteuert und ins Grid integriert.

5. *Ressourcen.* Die wichtigsten Ressourcen eines Data-Mining Systems bilden die Rechen- und Speicherressourcen. Auf Hardware-Ebene werden diese von Servern, PCs oder speziellen Speichersystemen bereitgestellt. Verschiedene Softwarekomponenten bieten Funktionen zum Speichern von Daten (Dateisysteme, Datenbanken) und Ausführung von Programmen (Betriebssysteme, Virtuelle Maschinen).

Die nachfolgenden Abschnitte stellen die Applikationsbeschreibung sowie die Daten- und Ausführungsverwaltung des FDA-Miner vor.

5.3 Applikationsbeschreibung

Die im Rahmen dieser Arbeit entwickelte Data-Mining Applikationsbeschreibung (Application Description Schema, ADS)[SSK+08] erfasst alle benötigten Informationen, um ein Data-Mining Programm in ein Grid zu integrieren. Das XML-Schema unterteilt sich in einen allgemeinen Teil, der eine Beschreibung des Programms enthält, einen Data-Mining-spezifischen Teil und eine Beschreibung des Ausführungszustandes.

Im Gegensatz zu bestehenden Grid-Ausführungsbeschreibungen, wie etwa JSDL[2] oder GRAM4-JDS[3], berücksichtigt das ADS die speziellen Eigenschaften von Data-Mining Applikationen. Das System nutzt diese Informationen unter anderem, um die Nutzer beim Auffinden geeigneter Algorithmen oder den bereits berechneten Ergebnissen zu unterstützen. Zusätzliche Elemente ermöglichen die Generierung programmspezifischer Konfigurationsoberflächen, die den Nutzern eine intuitive Bedienung ermöglichen. Darüber hinaus werden der aktuelle Status der Ausführung und die berechneten Ergebnisse durch das ADS erfasst. Die Beschreibung deckt damit den kompletten „Lebenszyklus“, von der Auswahl des Algorithmus bis zur Beendigung der Ausführung, eines Data-Mining-Programms ab.

Allgemeiner Teil

Im allgemeinen Teil werden Informationen gespeichert, die für das Auffinden und die Auswahl des Programms im Grid genutzt werden können:

- Textuelle Beschreibung der Applikation zu der dieses Programm gehört.
- Ausführliche Beschreibung des Algorithmus, dessen Parameter, Ein- und Ausgabedaten.
- Eine eindeutige ID (diese wird vom System vergeben).
- Hersteller des Programms.
- Versionsinformationen.

Weitere Elemente enthalten Informationen zur Ausführung des Programms.

- Programmtyp: Java, C, Bash Shell, Python.
- Das ausführbare Programm, die benötigten Bibliotheken und Subprogramme.
- Weitere Informationen zur Ausführungsumgebung, z.B. soll das Programm in einem Unterverzeichnis ausgeführt werden.

[2] https://forge.gridforum.org/sf/projects/jsdl-wg
[3] http://www-unix.globus.org/toolkit/docs/latest-stable/execution/gram4/schemas/gram_job_description.html

- Kommandos und Argumente zur Ausführung des Interpreters, z.B. „python“.

Die Optionen eines Programms werden in einzelnen Elementen definiert. Jedes Element enthält den Parameternamen und den dazugehörigen Wert, die zur Ausführung zum Kommandozeilenaufruf zusammengefügt werden. Um den Nutzern die korrekte Auswahl der Parameterwerte zu erleichtern, wird dem Parameter ein Datentyp zugewiesen und eine kurze textuelle Beschreibung angefügt. Zusätzlich können Standardwerte und/oder erlaubte Werte angegeben werden.

Weitere Elemente enthalten Informationen über die Ein- und Ausgabedaten des Programms. Zur genaueren Definition des Datums enthalten beide Elemente einen Namen für das Datum, eine kurze Beschreibung, den Status des Datums und um was für ein Datum es sich handelt (Datei, Verzeichnis, Parameterdatei). Neben Dateien und Verzeichnissen können auch Abfragen nach *logischen Objekten* (siehe Abschnitt 5.4) spezifiziert werden. Die Auflösung der logischen Namen in die entsprechenden Datensätze erfolgt vor der Ausführung durch die Ausführungskomponente. Ist ein Transfer der Eingabedaten nötig, können die Datensätze von entfernten Rechnern an die Ausführungsmaschine übertragen werden. Hierzu wird das Transferprotokoll, der Port, der Rechnername, der lokale Pfad und der Dateiname angegeben.

Die benötigten Umgebungsvariablen, die vor der Ausführung des Programms auf der Ausführungsmaschine gesetzt werden sollen, können diese in einem Element spezifiziert werden.

Die Anforderungen des Programms können ebenfalls erfasst werden. Die Hardwareanforderungen des Programms können durch Mindestanforderungen an Festplattenplatz und Hauptspeicher sowie die Einschränkung auf eine bestimmte Prozessorarchitektur (intel, x64, itanium,...) beschrieben werden. Ist das Programm nur auf einem Betriebssystem (Windows, Linux, ...) lauffähig, kann dieses spezifiziert werden. Die Details der Ausführung im Grid-System, wie der verwendete Ausführungsmanager oder die Auswahl bestimmter Ausführungsmaschinen, können ebenfalls angegeben werden.

Im Data-Mining werden häufig die Werte eines Parameters variiert, um eine möglichst exakte Lösung zu finden. Anstatt für jeden einzelnen Wert eine neue Instanz der Applikationsbeschreibung zu erzeugen, können mehrere unterschiedliche Werte für einen Parameter (Parameterschleifen/Parameterlisten) definiert werden. Eine Liste numerischer Wert wird durch Angabe eines Startwertes, eines Endwertes und einer Schrittweite (1, 2, 3, 4, 5, ...) bestimmt. Listen mit alphanumerischen Zeichen können ebenfalls genutzt werden. Die in den Listen enthaltenen Parameterwerte können in anderen Elementen referenziert werden. Das Ausführungssystem erzeugt für jeden Wert der Liste einen eigenen Auftrag und fasst die einzelnen Ergebnisse zusammen.

Data-Mining Teil

Der Data-Mining-spezifische Teil der Applikationsbeschreibung beinhaltet die relevanten Informationen eines Data-Mining Programms und folgt der in [CC03] be-

schriebenen Ontologie. Im Einzelnen werden im Element *applicationType* folgende Informationen zusammengefasst:

- Die Applikationsdomäne (momentan immer Data-Mining) .
- Name der Applikation, zu der dieses Programm gehört.
- Name des Algorithmus, der von diesem Programm implementiert wird.
- Der Problemtyp, den dieser Algorithmus löst (Klassifikation, Regression, ...).
- Eine Beschreibung der vom Algorithmus eingesetzten Technik (hierarchisches, partitionierendes Clustering).
- Die CRISP-DM-Phase, zu der dieser Algorithmus gehört (Business Understanding, Data Understanding, Data Preparation, Modelling, Evaluation, Deployment).

In Kombination mit einem geeigneten Suchdienst ermöglichen diese Informationen ein schnelles und einfaches Auffinden passender Algorithmen/Programme im Grid.

Ausführungszustand

Die Informationen, die den aktuellen Zustand der Ausführung eines Programms umfassen:

- Der Start- sowie der Endzeitpunkt der Ausführung wird für Laufzeitanalysen oder zeitbasierte Ergebnisvergleiche gespeichert.
- Der aktuelle Status (abgeschickt, in Ausführung, beendet, ...) des gesamten Jobs und aller Teiljobs inklusive eventueller Fehlerbeschreibung.
- Die URI des Ergebnisses der Ausführung des gesamten Jobs.

Auf Basis dieser Informationen können die Benutzer oder Applikationen den Ausführungszustand überwachen.

5.3.1 Applikationsverwaltung

Die Verwaltung der Applikationen umfasst die Speicherung und die Recherche von beliebigen Data-Mining Programmen. Ein Data-Mining Programm besteht aus Sicht der Applikationsverwaltung aus ausführbaren Programmen, Bibiliotheken sowie Konfigurationsdateien und einer ADS-Beschreibung.

Die Integration der ADS-Instanz in ein Grid-System ist dabei auf verschiedene Arten realisierbar. Eine einfache Methode ist der Einsatz eines speziellen Dienstes zur Abfrage und Speicherung von ADS-Instanzen. Obwohl einfach und schnell zu

implementieren, führt der Ausfall dieses einen Services zum Ausfall des Gesamtsystems. Der gewählte Ansatz nutzt daher den „Monitoring and Discovery System“ (MDS)[Fit01] für die Speicherung und die Abfrage von ADS-Instanzen. Der MDS garantiert eine robuste und dezentrale Speicherung der Informationen, so dass der Ausfall einzelner Maschinen die Funktion des Systems nicht gefährdet. Die Übergabe und die Aktualisierung der Informationen im MDS erfolgt über einen oder mehrere Applikationsdienste, welche auf dem MDS Aggregator Framework basieren. Die Applikationsservices lesen periodisch aus einem definierten Verzeichnis des Grid-Servers alle ADS-Beschreibungen, überprüfen deren Korrektheit und speichern die Informationen im MDS. Neue Applikationsbeschreibungen und Änderungen an vorhandenen Beschreibungen werden so zeitnah im System sichtbar. Für die Suche nach Data-Mining Programmen im Grid können die vom MDS bereitgestellten Funktionen genutzt werden.

Bevor ein Data-Mining Programm im Grid registriert und genutzt werden kann, muss eine ADS-Instanz erzeugt und die ausführbaren Programme, Bibiliotheken sowie Konfigurationsdateien in das Grid transferiert werden. Der im Rahmen dieser Arbeit entwickelte *Data-Mining Application Enabler* unterstützt die Applikationsentwickler hierbei. Die Web-Oberfläche verbirgt die komplexe XML-Syntax und bietet für die einzelnen Teile des ADS, wie in Abbildung 5.2 dargestellt, Web-Formulare. Aus den eingegebenen Informationen wird eine korrekte ADS-Instanz erzeugt und mit allen benötigten ausführbaren Dateien und Bibliotheken auf einen angegebenen Grid-Server transferiert.

5.4 Datenverwaltung

Die Datenverwaltung ist ein integraler Bestandteil jedes Data-Mining Systems. Insbesondere in organisationsübergreifenden Umgebungen sind Benutzer auf die Datenverwaltung angewiesen, um die vorhandenen Datensätze und ihre Repliken verwalten zu können. Darüber hinaus bildet die Datenverwaltung die Grundlage zur Realisierung des DOHS-Schedulingverfahrens für daten- und rechenintensive Anwendungen im FDA-Miner.

Die Heterogenität und die Souveränität der unterschiedlichen Organisationen erschweren den Einsatz eines gemeinsamen Datenspeichersystems, etwa eines einheitlichen, verteilten Dateisystems, erheblich. Die entwickelte Datenverwaltung konzentriert sich daher auf die Verwaltung der Meta-Daten der Datensätze und überlässt die eigentliche Datenspeicherung den Datenbanken oder Dateisystemen der einzelnen Organisationen. Im Gegensatz zu klassischen Grid-Systemen, bei denen nur Daten auf speziellen Grid-Servern verwaltet werden können, dürfen die Datensätze auf beliebigen Ressourcen des Grids abgelegt werden. Darüber hinaus, ermöglicht das flexible Meta-Datenformat beliebige Informationen über die Datensätze zu erfassen und zu durchsuchen.

Die Datenverwaltung besteht aus einer relationalen Objektdatenbank, in der die Meta-Daten abgelegt werden, und Datendiensten, die einen einheitlichen Zugriff auf

Abbildung 5.3: Der Data-Mining Application Enabler.

diese Daten bereitstellen. Die Architektur der Datenverwaltung basiert auf den Konzepten des GT4 Replica Location Service [RF02b], wobei die entwickelte Datenverwaltung weitaus mächtigere Funktionen bietet. In den nachfolgenden Abschnitten werden die Objektdatenbank und die Datendienste mit ihren Schnittstellen detailliert erläutert.

5.4.1 Objektdatenbank

In den Tabellen der Objektdatenbank können beliebige Meta-Daten zu den im System vorhandenen Daten gespeichert werden. Die Objektdatenbank bildet damit die Grundlage der gesamten Datenverwaltung. Aufbauend auf der Architektur und den Interna des RLS wurde ein auf die Data-Mining Anforderungen abgestimmtes Datenmodell entworfen.

Der RLS nutzt logische und physische Dateinamen (LFN, PFN) zur Verwaltung von Dateien [CPB+04]. LFNs und PFNs können beliebig viele beschreibende Attribute zur Identifikation zugeordnet werden. Aus der Implementierung des Local Replica Catalogs und des Replica Location Index ergeben sich die zwei entscheiden-

den Nachteile des RLS [CCF04]: (1) Die interne Datenverwaltung des LRC nutzt einzelne Datenbanktabellen für die Speicherung der Attribute, so dass es mit den bereitgestellten Programmen nicht möglich ist, gleichzeitig über mehrere Attribute zu suchen. Prinzipiell ließe sich eine solche Suchoption durch Joins realisieren, wobei für jedes Attribut eine zusätzliche Join-Bedingung nötig ist. Dies würde bei Anfragen die viele Attribute einbeziehen, unvermeidlich zu Problemen führen. (2) Den vollen Funktionsumfang des RLS erhält man erst beim Einsatz des RLI für die Verwaltung mehrerer LRCs. Der RLI erlaubt jedoch nur die Verwaltung von LFNs ohne zusätzliche Attribute. Eine Suche nach Daten über deren Attribute ist daher nicht möglich.

In vielen Data-Mining Anwendungen werden jedoch häufig mehrere Attribute (Aufzeichnungsdatum, System, Parameter, ...) zur Auswahl der Daten benötigt. Das entwickelte Datenmodell wird diesen Anforderungen gerecht und ist nicht nur in der Lage Datenobjekten beliebige Attribute zuzuordnen, sondern auch darauf optimiert, über all diese zu suchen. Die Meta-Daten eines Datenobjekts, die Objektinformationen, werden hierzu einer Objektklasse zugeordnet. Diese legt die zur Beschreibung der Datenobjekte benötigten Attribute fest. Es wird, ähnlich dem RLS, zwischen logischen und physischen Objektinformationen unterschieden. Logische Objekte dienen der abstrakten Beschreibung der Daten (zugeordnete Applikation, Messverfahren, Parameter, Aufzeichnungsdatum, ...). Physische Objekte enthalten Informationen über die eigentlichen Daten (Speicherort, Größe, letzter Zugriff, ...). Logische und physische Objekte können einander beliebig zugeordnet werden. Die Definition von logischen und physischen Objektattributen ermöglicht die Verwaltung beliebiger Daten (Dateien, Datenbanktabellen, ...). Für Dateien und Verzeichnisse lassen sich damit etwa POSIX-konforme physische Attribute erzeugen.

Soll ein neues Datenobjekt angelegt werden, werden ein logisches und ein physisches Objekt angelegt. Diesen Objekten werden mindestens ein logischer Objektname (LON) und ein physischer Objektname (PON), sowie beliebig vielen weitere Attribute zugeordnet. LONs und PONs müssen innerhalb einer Objektklasse eindeutig sein, wobei im FDA-Miner die MD5-Summe des Datensatzes zur Bestimmung der Eindeutigkeit verwendet wird. Jeder LON und PON wird mit den dazugehörigen logischen und physischen Attributen der Objektklasse in einer Datenbanktabelle verwaltet. Die Zuordnung zwischen den logischen und physischen Objektinformationen erfolgt über eine zusätzliche Zuordnungstabelle, so dass sich beliebig viele LONs und PONs einander zuordnen lassen.

Mit dem beschriebenen Datenmodell können die Meta-Daten persistent in einem Relationalen Datenbankmanagementsystem gespeichert werden. Die aktuelle Implementierung nutzt hierfür eine MySQL-Datenbank, wobei beliebige RDMS genutzt werden können. Die Verwaltung der logischen und physischen Attribute einer Objektklasse in einer eigenen Tabelle vereinfacht die Abfrage über mehrere Attribute erheblich. Darüber hinaus können über einen einfachen SQL-Join-Operationen Zuordnung zwischen logischen und physischen Objekten hergestellt werden.

5.4.2 Datendienst

Der Datendienst stellt den Applikationen und Systemdienste eine einheitliche Schnittstelle für die Verwaltung von Meta-Daten zur Verfügung. Der Datendienst bietet hierfür öffentliche Funktionen zur Manipulation von Objektklassen und Objektinformationen. Die Schnittstelle ermöglicht beliebigen Applikationen und Systemdiensten Zugriff auf die öffentlichen Funktionen des Datendienstes mit denen Objektklassen und Objektinformationen erzeugt, verändert und gelöscht werden können:

- Erzeuge Objektklasse. Für das Anlegen einer neuen Objektklasse wird ein eindeutiger Name, eine Beschreibung, die logischen und die physischen Attribute benötigt. Ist eine Objektklasse mit dem angegebenen Namen bereits vorhanden, wird die Anfrage verworfen.

- Verändere Objektklasse. Die ausgewählten Attribute werden der angegebenen Objektklasse hinzugefügt oder entfernt. Die entsprechenden Datenbanktabellen werden angepasst.

- Lösche Objektklassen. Die gesamte Objektklasse wird entfernt. Es werden auch alle Meta-Daten dieser Klasse gelöscht.

- Erzeuge logische/physische Objektinformationen. Es wird ein neues logisches/-physisches Objekt in der ausgewählten Objektklasse mit dem angegebenen LON/PON und den übermittelten Attributwerten erzeugt. Ist der LON/PON bereits vorhanden, wird die Anfrage verworfen.

- Zuordnung logische/physische Objektinformationen. Die LONs/PONs der Objektklasse werden einander zugeordnet.

- Verändere logische/physische Objektinformationen. Die ausgewählten logischen/physischen Attribute der Objektklasse werden hinzugefügt oder entfernt.

- Lösche logische/physische Objektinformationen. Die ausgewählten logischen/-physischen Objekte der Objektklasse werden gelöscht.

Die zentrale Funktion des Datendienstes ist die Suche nach Datenobjekten. Im einfachsten Fall ermittelt eine Suchanfrage die Zuordnung zwischen vorgegebenen LONs oder PONs einer Objektklasse. Komplexere Abfragen suchen eine Zuordnung von LONs/PONs in Abhängigkeit mehrerer logischer/physischer Attributbedingungen. Als Ergebnis einer Suche erhält der Aufrufer eine Liste von Objekten. Jedem Objekt dieser Liste können mehrere Attribut/Wert-Paare und andere Objekte zugeordnet sein.

5.5 Ausführungsverwaltung

Die Ausführungsverwaltung ist für die sichere Ausführung der von den Benutzern oder Applikationen erzeugten Aufträgen verantwortlich. Der entwickelte Dienst implementiert den DOHS-Algorithmus und nutzt die Datenverwaltung, den Condor-G Scheduler[FTL+02] und die Dienste der Grid-Infrastruktur[Fos05], insbesondere den RFT, den WS-GRAM, den MDS sowie die Authentifizierungs- und Autorisierungsfunktionen.

Die Ausführung eines Data-Mining Programmes mit einem Parametersatz sowie Ein- und Ausgabedaten ergeben einen *Auftrag*. Für die effiziente Ausführung eines Auftrags müssen insbesondere folgende Funktionen vom Ausführungsdienst zur Verfügung gestellt werden:

- *Ressourcenanfragen auf Ressourcenangebote abbilden (*Scheduling*).* Eine der zentralen Aufgaben der Ausführungsverwaltung ist es, Ressourcenanfragen (Aufträge in Form von ADS-Beschreibungen) auf die vorhandenen Ressourcen abzubilden und dabei einen möglichst geringen Verarbeitungsaufwand zu generieren. Der DOHS benötigt hierfür die aktuellen Zustandsinformationen der Ressourcen sowie die Positionsinformationen der Datensätze.

- *Transfer der Daten und Programme zur Ausführungsressource.* Sind die zur Verarbeitung benötigten Daten und Programme nicht auf der im vorherigen Schritt ausgewählten Rechenressource vorhanden, muss die Ausführungsverwaltung alle benötigten Daten zu dieser Ressource transferieren. Anschließend kann der Auftrag gestartet werden.

- *Überwachung der Ausführung.* Ein Auftrag kann während seiner Ausführung verschiedene Zustände, wie wartend, aktiv, beendet oder fehlerhaft, annehmen. Der Benutzer muss den Status der Ausführung ständig kontrollieren können, um entsprechend auf den aktuellen Status reagieren zu können (wiederholen, abbrechen).

- *Transfer der Ergebnisse.* Die Ausführung eines Programms kann in einem Grid auf Ressourcen anderer Organisationen erfolgen. Wie lange diese Maschinen noch zur Verfügung stehen, wird durch die verantwortliche Organisation bestimmt. Eine Speicherung der Ergebnisse auf der Ausführungsressource ist daher nicht immer erwünscht. Nach der Beendigung der Ausführung müssen deshalb die Ergebnisse von der Ausführungsressource zur Benutzermaschine oder einem vom Benutzer bestimmte Speicherressource transferiert werden können. Alle temporär erzeugten Daten müssen auf der Ausführungsressource gelöscht werden, um den Speicherplatz wieder freizugeben.

Grid-Systeme stellen bereits eine Reihe von Basisdiensten für die Ressourcenverwaltung, die Ausführungsverwaltung auf einzelnen Knoten sowie den Datentransfer bereit, so dass diese vom Ausführungsdienst genutzt werden können. Es werden daher nur die fehlenden Funktionen, etwa die Verarbeitung einer ADS-Beschreibung oder das Scheduling, durch den Ausführungsdienst implementiert.

Die Kommunikation der Clients mit dem Ausführungsdienst erfolgt über eine definierte Schnittstelle. Die Schnittstelle enthält Funktionen zur Annahme eingehender Ausführungsanfragen, zur Statusabfrage und zum Abbruch einer Ausführung. Ein einzelner Auftrag wird dabei durch eine ADS-Beschreibung definiert. Beim Eintreffen eines neuen Auftrags initiiert der Ausführungsdienst das Scheduling sowie die Ausführung des Auftrags und stellt den Benutzern alle benötigten Statusinformationen zur Verfügung.

Die Ausführungsverwaltung besteht, wie in Abbildung 5.3 dargestell, aus mehreren Komponenten. Erhält der Ausführungsdienst einen neuen Auftrag, wird zunächst die ADS-Beschreibung verarbeitet. Enthält die ADS logische Datenobjekte oder eine Suchanfrage nach logischen Datenobjekten, ermittelt der Ausführungsdienst zunächst den physischen Speicherort der Daten mithilfe des Datendienstes. Anschließend erzeugt der DOHS-Algorithmus eine Zuordnung für den aktuellen Auftrag. Der DOHS erhält hierfür die benötigten Ressourcenstatusinformationen sowie Dateninformationen. Die vom DOHS-Algorithmus erzeugte Zuordnung wird zur eigentlichen Ausführung vom Ausführungsdienst in einen Condor-G Auftrag überführt und zur Ausführung an Condor-G übergeben. Condor-G übernimmt die Kommunikation mit der Grid-Middleware sowie die Überwachung der Ausführung. Im Gegensatz zu den Ausführungsdiensten einer bestimmten Grid-Middleware, hier GT4, bietet Condor-G eine einheitliche Schnittstelle für die Integration beliebiger Grid- und Cluster-Middlewares, was die Portierbarkeit sowie die Adaption auf neue Middlewares erheblich vereinfacht. Darüber hinaus bietet Condor-G mit den *Class-Ads*[RLS98][RLS00] eine flexible Schnittstelle, um die vom DOHS-Algorithmus erzeugten Zuordnungen in einer heterogenen Grid-Umgebung auszuführen.

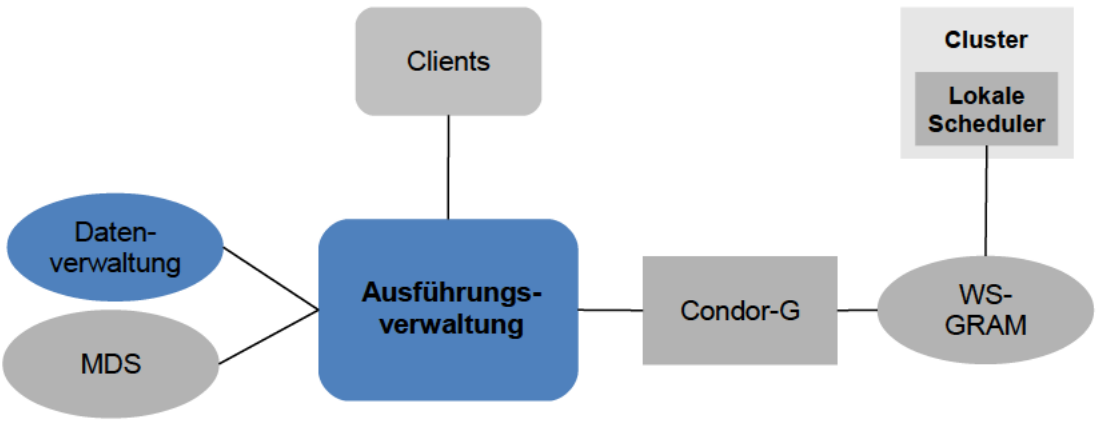


Abbildung 5.4: An der Ausführungsverwaltung beteiligte Komponenten.

Die Funktionsweise des Ausführungsdiensts soll anhand eines einfachen Beispiels erläutern werden. Abbildung 5.4 zeigt die einzelnen Schritte bei der Vorbereitung und Ausführung eines Auftrags. In diesem Szenario sollen Daten der Organisation A mit einem Data-Mining Programm der Organisation D verarbeitet werden. Der

Client befindet sich in der Organisation C und es stehen mehrere Rechenressourcen in den Organisationen A, B, und D zur Verfügung.

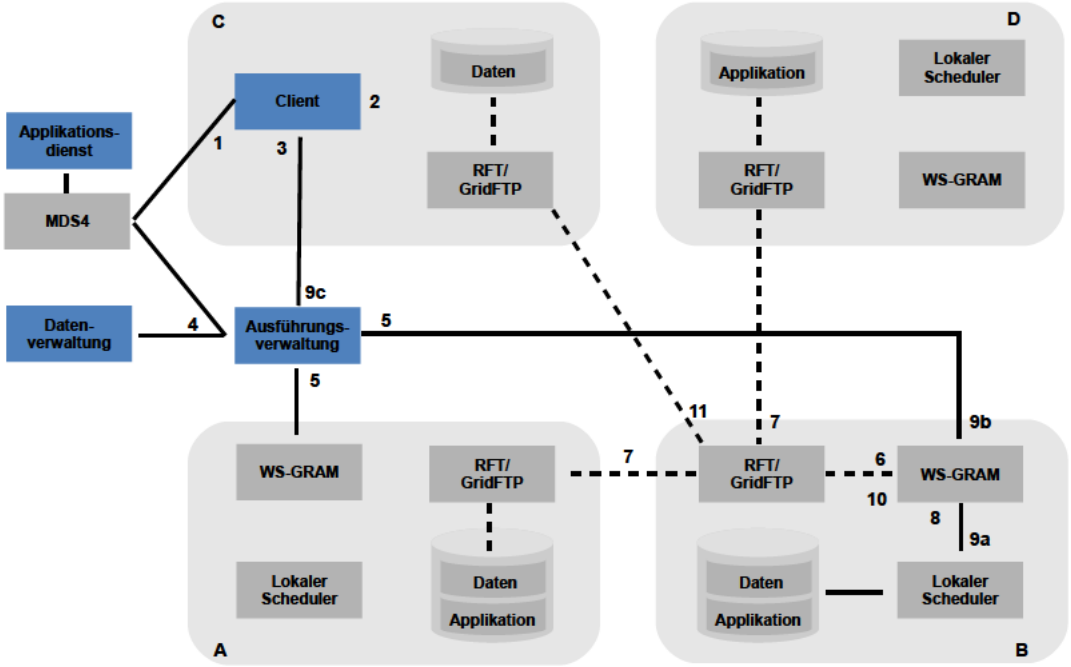


Abbildung 5.5: Interaktion der Komponenten bei der Ausführung eines Auftrags.

Schritte 1-3: Vorbereitung des Data-Mining Programmes

Suche und Auswahl eines geeigneten Algorithmus zur Analyse der Daten aus den im MDS gespeicherten ADS-Beschreibungen. Definition der Parameter des ausgewählten Data-Mining Programms und der Ein- und Ausgabedaten (hier in C). Für die resultierende ADS-Instanz wird ein Auftrag erzeugt und an den Ausführungsdienst weitergeleitet. Des Weiteren werden die Sicherheitsinformationen des Benutzers übergeben.

Schritte 4-8: Vorbereitung und Start der Ausführung

Beim Eingang einer Anfrage bezieht der Ausführungsdienst vom MDS zunächst die aktuellen Informationen über die verfügbaren Ressourcen, deren Eigenschaften und Auslastung. Anschließend werden die in der ADS-Instanz enthaltenen logischen Datenbeschreibungen ausgewertet und die Position der Daten mithilfe der Datendienste ermittelt (hier A und B). Anschließend wird für den generierten Auftrag durch den DOHS eine Zuordnung erzeugt. Diese wird in einen Condor-G Auftrag übersetzt und die Ausführung initiiert. Condor-G kommuniziert mit den entsprechenden WS-GRAM Ausführungsdiensten von GT 4 (hier B). Die Auftragsbeschreibung enthält

neben den Informationen zum auszuführenden Programm, auch die lokale Rechenressource, auf der das Programm gestartet werden soll sowie die URLs des Programmes und der Ausgabedaten. Der WS-GRAM koordiniert so den Transfer der Programmdateien mithilfe des RFT-Dienstes. Im Beispiel werden das Data-Mining Programm aus der Organisation D und ein Teil der Eingabedaten aus der Organisation A zur Organisation B transferiert. Sind die Daten und das Programm lokal vorhanden, startet der WS-GRAM die Ausführung des Programms auf der ausgewählten Rechenressource, die die Eingabedaten vorhält.

Schritt 9: Überwachung der Ausführung

Wie bereits erwähnt, können aus einer einzelnen ADS-Instanz mehrere Aufträge erzeugt werden. Nach dem Start wird die Ausführung der einzelnen Aufträge auf unterschiedlichen Ebenen überwacht. Auf unterster Ebene wird der Verlauf der Ausführung eines Auftrags von den Clusterverwaltungssystemen kontrolliert. Die WS-GRAMs nutzen diese Informationen, um alle ihnen zugeordneten Aufträge zu verwalten. Condor-G integriert die Statusinformationen aller bei der Ausführung beteiligter WS-GRAMs. Der Ausführungsdienst überwacht die Ausführung aller Aufträge, indem er regelmäßig Condor-G kontaktiert und den Status erfragt. Tritt bei der Ausführung ein Fehler auf, wird dies dem Client mitgeteilt.

Schritte 10-11: Ende der Ausführung und Ergebnistransfer

Nach der Beendigung eines Auftrags werden die Ergebnisse vom WS-GRAM zu einem temporären Verzeichnis transferiert. Sind alle Aufträge beendet, erzeugt der Ausführungsdienst aus den Teilergebnissen eine Ergebnisdatei, die anschließend an eine vorgegebene Speicherressource übertragen wird. Im Beispiel wird zur Speicherung der Teilergebnisse und der Ergebnisse die gleiche Speicherressource (C) verwendet.

5.6 Benutzeroberfläche

Der FDA-Miner stellt für die Entwicklung neuer Methoden und Modelle aus den Daten der Brennstoffzellenfahrzeuge eine flexible webbasierte Benutzeroberfläche zur Verfügung. Die Benutzer interagieren mit dem FDA-Miner ausschließlich über diese Web-Oberfläche, die neben klassischen Reportingfunktionen auch individualisierte Datenaufbereitungs- und Analyseverfahren bereitstellt. Die in Abbildung 5.5 dargestellte Analyseoberfläche bietet vielfältige Funktionen zum Filtern, berechnen zusätzlicher Werte und visualisieren der Daten der gesamten Flotte. Die Verarbeitung steuert der Benutzer über mehrere Menüs und Einstellungsfenster:

1. Der Benutzer wählt zunächst die zu verarbeitenden Fahrzeuge oder Stacks.
2. Wenn nicht der gesamte Zeitraum betrachtet werden soll, kann ein Zeitintervall angegeben werden.

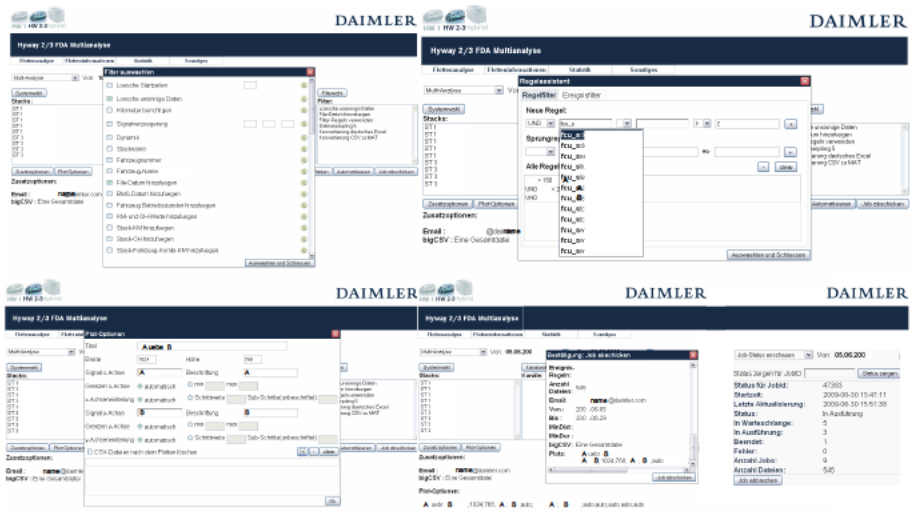


Abbildung 5.6: Die Analyseoberfläche des FDA-Miners.

3. Über eine Kanalwahl wird festgelegt, welche der aufgezeichneten Kanäle für diese Analyse betrachtet werden sollen.

4. Mit dem Regelassistenten lassen sich beliebig komplexe Regeln zur Datenfilterung erzeugen.

5. Sollen weitere Aufbereitungs- oder Analyseprogramme auf die Daten angewendet werden, können diese aus einer Liste mit allen für diese Daten verfügbaren Programme gewählt werden.

6. Weitere Einstellungen erlauben verschiedene Ausgabeformate sowie eine grafische Darstellung der Ergebnisse.

7. Ist ein Verarbeitungsauftrag vollständig konfiguriert, kann dieser zur späteren Verwendung gespeichert oder direkt ausgeführt werden. Darüber hinaus können Verarbeitungsaufträge regelmäßig (täglich, wöchentlich, monatlich) automatisiert ausgeführt werden.

Aus der vom Benutzer getroffenen Auswahl werden ein oder mehrere Aufträge mit den entsprechenden ADS-Beschreibungen erzeugt und an die Ausführungsverwaltung übergeben. Der Benutzer kann den Status der Ausführung jederzeit überwachen und die Ausführung bei Bedarf beenden.
Zusätzlich zur Analyseoberfläche stehen den Benutzern weitere spezialisierte Oberflächen zur Verfügung. Hierzu zählen etwa zwei Oberflächen für die Alterungszustandsanalysen aus Kapitel 6. Diese erlauben sowohl die Konfiguration der Analysen als auch die Darstellung der Ergebnisse.

Die Visualisierung der gesamten Flottendaten ist mit herkömmlichen Visualisierungs- und Auswertungsprogrammen kaum möglich. Die Weboberfläche

bietet den Benutzern daher die wichtigsten Visualisierungs- und Reportingfunktionen für die gesamte Flotte. Abbildung 5.6 zeigt einige Funktionen, wie die Statusan-

Abbildung 5.7: Visualisierungs- und Reportingfunktionen des FDA-Miners.

zeige, die dynamischen Histogramme, die Lastverteilung, die Korrelationsmatrix und die Datenübersicht. Neben den vorgegebenen Standardanzeigen können sich die Benutzer auch individualisierte Reports mit weiteren Informationen und Darstellungen generieren.

Die dynamische, interaktive Web-Oberfläche besteht aus einer Reihe speziell für die Flottenanalyse entwickelter Widgets auf Basis des Google Web Toolkits. Die Widgets ermöglichen eine einfache Weiterentwicklung bestehender Oberflächen wie auch neuer Web-Applikationen für weitere Flottenanalysesysteme. Eine Reihe generischer Java-Komponenten bietet den Widgets Funktionen für die Interaktion mit dem FDA-Miner Verarbeitungssystem. Durch diese Komponenten können Web-Applikationen transparent auf die Funktionen des Analysesystems zugreifen.

5.7 Evaluierung des FDA-Miners

Mithilfe der aktuellen, produktiven Version des FDA-Miners sind bis jetzt bereits mehrere Tausend Aufträge mit Millionen an Datensätzen verarbeitet worden. Die nachfolgende Evaluation des Systems dient daher nicht dem Nachweis der prinzipiellen Funktionsfähigkeit des Systems, sondern vielmehr dem Vergleich der entwickelten Verfahren und deren Implementierung mit ähnlichen gridbasierten Ansätzen.

Die Messungen wurden auf dem aktuellen FDA-Miner System durchgeführt,

das derzeit aus neun Ressourcen besteht. Auf einer Ressource werden ausschließlich die Datenintegration, die Ausführungs- und die Datenverwaltung sowie die Visualisierungs- und Reportingfunktionen ausgeführt. Die verbleibenden acht Ressourcen sind zugleich Rechen- und Speicherressourcen. Die Flottendaten sind über diese Ressourcen verteilt gespeichert, wobei jeder Datensatz auf mindestens zwei Ressourcen abgelegt ist. Die Verarbeitung der Datensätze kann daher ohne Datenübertragung erfolgen. Ein GigaBit-Ethernet Netzwerk verbindet die einzelnen Ressourcen.

Die Verarbeitungsleistung des aktuellen FDA-Miners wird im Folgenden anhand einiger typischen Aufgaben evaluiert. Der Fokus liegt dabei auf dem Vergleich mit der Ausführung auf einer einzelnen Ressource - zum Vergleich mit einer Auswertung auf einer Workstation eines Ingenieurs - sowie der Ausführung auf mehreren Ressourcen mit einer strikten Trennung zwischen Rechen- und Speicherressourcen - wie es in klassischen Grid-Systemen der Fall ist.
Neben dem Umgebungsaufbau können bei dieser Evaluierung auch die unterschiedlichen Schedulingstrategien zwischen dem DOHS und bestehenden Grid-Schedulern unter realen Bedingungen verglichen werden. Da sich die Ressourcen innerhalb einer Organisation befinden, kann aus Sicht der klassischen Grid-Scheduler die Transferdauer vernachlässigt werden. Der DOHS beachtet hingegen die Einflüsse des Datentransfers innerhalb einer Organisation und versucht unnötigen Datentransfer zu vermeiden. Im aktuellen Aufbau des FDA-Miner können die Kosten der einzelnen Ressourcen vernachlässigt werden, so dass der Fokus bei der Evaluierung auf der Ausführungsdauer der Aufträge liegt.

Die in den Tabellen 5.1 und 5.2 aufgeführten Messergebnisse zeigen das Verhalten zweier Analyseverfahren bei der Variation der Datensätze und der Anzahl der verwendeten Prozessorkerne im aktuellen FDA-Miner. Die verteilte Berechnung der beiden Analyseverfahren, das Optimierungsverfahren und die Künstlichen Neuronalen Netze (KNN) aus Abschnitt 6.3.2, unterscheiden sich dabei in ihrer Parallelität erheblich. Bei der Interpretation der nachfolgend vorgestellten Werte muss natürlich beachtet werden, dass die zu verarbeitenden Datensätze unterschiedlich groß sind. Dennoch können aus den ermittelten Werten Rückschlüsse auf das Verhalten der einzelnen Methoden bei der verteilten Ausführung gezogen werden.

Berechnungsdauer des KNN-Verfahrens

Bei dem KNN-Verfahren wird aus einer Eingabedatei, die alle aufbereiteten Daten eines Fahrzeugs enthält, ein Fahrzeugmodell auf einem Prozessorkern berechnet. Damit hängt die Skalierung der Modellberechnung nur von der Anzahl der zu verarbeitenden Fahrzeuge und der vorhandenen Prozessorkerne ab. Die Modellberechnung eines einzelnen Fahrzeugs wird bei diesem Verfahren nicht beschleunigt. Die vor der Modellberechnung notwendige Datenaufbereitung kann jedoch für jeden Datensatz separat erfolgen, so dass durch ein geeignetes Scheduling eine Übertragung der Eingabedaten vermieden werden kann. Die Verarbeitungsdauer dieses Vorverarbeitungsschrittes verhält sich somit annähernd linear zur Anzahl der eingesetzten Pro-

zessorkerne. Bei dieser verteilten Berechnung müssen die aufbereiteten Datensätze in einem Zwischenschritt zu einer Eingabedatei zusammengefügt und zur Modellberechnung an eine Rechenressource übertragen werden. Dieser zusätzliche Aufwand muss bei der Betrachtung der Gesamtdauer mitberücksichtigt werden und wird in der Spalte "Kombination" dargestellt.

Test	**Aufbereitung**	**Kombination**	**Modellbildung**
1 Fahrzeug, 1 CPU	138	-	433
1 Fahrzeug, 10 CPUs	12:46	0:31	437
1 Fahrzeug, 20 CPUs	6:02	0:31	438
10 Fahrzeuge, 1 CPU	1274	-	4681
10 Fahrzeuge, 10 CPU	116	3:14	465
10 Fahrzeuge, 20 CPUs	54	3:11	464
10 Fahrzeuge, 40 CPUs	29	3:19	460
10 Fahrzeuge, 60 CPUs	17	3:02	467

Tabelle 5.1: Verarbeitungszeiten der KNN-Analyseverfahren (in Minuten).

Mit den nachfolgend beschriebenen Messergebnissen wird das Verhalten des Systems bei der Variation der Anzahl der eingesetzten Prozessorkerne sowie der Anzahl der Datensätze bei der Berechnung des KNN-Verfahrens untersucht. Hierbei bezeichnet $t_{i,j}$ die Ausführungsdauer der Berechnung von i Fahrzeugen auf j Prozessorkernen. Betrachtet man das Verhältnis der Ausführungsdauer auf einem Prozessorkern zu n Prozessorkernen (10, 20, 40 und 60) bei der Berechnung von 10 Fahrzeugen, zeigen sich folgende Skalierungseigenschaften:

$$\begin{aligned} t_{10,1}/t_{10,10} &= (1274 + 4641)/(116 + 3,1 + 465) = 10,1 \\ t_{10,1}/t_{10,20} &= (1274 + 4641)/(54 + 3,1 + 464) = 11,3 \\ t_{10,1}/t_{10,40} &= (1274 + 4641)/(29 + 3,3 + 460) = 12,0 \\ t_{10,1}/t_{10,60} &= (1274 + 4641)/(17 + 3 + 467) = 12,1 \end{aligned}$$

Da die Modellberechnung nicht von mehr als 10 Prozessorkernen profitieren kann, sinkt die Gesamtberechnungsdauer mit mehr Prozessorkernen kaum noch.

Vergleicht man die Berechnungsdauer für 1 Fahrzeug auf 1 Prozessorkern mit 10 Fahrzeugen auf 10 Prozessorkernen ergibt sich erwartungsgemäß ein sehr gutes, annähernd lineares Verhältnis:

$$t_{1,1}/t_{10,10} = \frac{138 + 433}{116 + 3,1 + 465} = 0,98$$

Berechnungsdauer des Optimierungsverfahrens

Beim Optimierungsverfahren wird, im Gegensatz zum KNN-Verfahren, für jeden Datensatz ein eigenes Modell berechnet. Die einzelnen Datensätze eines Fahrzeugs können so unabhängig voneinander verarbeitet werden. Die Verarbeitungsdauer sinkt daher annähernd linear mit der Anzahl der verwendeten Prozessorkerne, wobei die einzelnen Ergebnisse bei der verteilten Berechnung noch in einem nachfolgenden Schritt kombiniert werden müssen.

Test	**Verarbeitungszeit**	**Kombination**
1 Fahrzeug, 1 CPU	342	-
1 Fahrzeug, 10 CPUs	38:47	0:12
1 Fahrzeug, 20 CPUs	15:54	0:12
1 Fahrzeug, 40 CPUs	8:32	0:15
1 Fahrzeug, 60 CPUs	5:08	0:14
10 Fahrzeuge, 1 CPU	2982	-
10 Fahrzeuge, 10 CPUs	349	0:39
10 Fahrzeuge, 20 CPUs	162	0:46
10 Fahrzeuge, 40 CPUs	90	0:51
10 Fahrzeuge, 60 CPUs	57:44	0:59

Tabelle 5.2: Verarbeitungszeiten des Optimierungsanalyseverfahrens (in Minuten).

Betrachtet man das Verhältnis der Ausführungsdauer auf einem Prozessorkern zu n Prozessorkernen (10, 20, 40 und 60) für das Optimierungsverfahren, ergeben sich die erwarteten annähernd linearen Beschleunigungen:

$$\begin{aligned} t_{10,1}/t_{10,10} &= 2982/350 = 8,5 \\ t_{10,1}/t_{10,20} &= 2982/163 = 18 \\ t_{10,1}/t_{10,40} &= 2982/91 = 33 \\ t_{10,1}/t_{10,60} &= 2982/59 = 51 \end{aligned}$$

Der Vergleich der Berechnungsdauer von 1 Fahrzeug auf 1 Prozessorkern mit 10 Fahrzeugen auf 10 Prozessorkernen erfüllt ebenfalls die Erwartungen:

$$t_{1,1}/t_{10,10} = \frac{342}{350} = 0,97$$

Berechnungsdauer einer Datenfilterung

Die Messergebnisse aus Tabelle 5.3 zeigen die unterschiedlichen Verarbeitungsdauern in den verschiedenen Umgebungsmodellen und zwischen dem Scheduling des

DOHS und klassischen Grid-Schedulern bei der Datenfilterung von unterschiedlich vielen Datensätzen. Dieser Test dient dem Vergleich mit der ersten Version des FDA-Miners und klassischen Grid-Systemen, in denen alle Daten einer Organisation auf einer Speicherressource gespeichert sind und die Ausführung auf getrennten Rechenressourcen erfolgt. Diese Umgebungen erfordern zwingend den Transfer der Daten vor der Verarbeitung von der Speicherressource auf die Ausführungsressourcen. Da sich die Ressourcen in einer Organisation befinden, erzeugen diese Transfers für klassische Grid-Scheduler jedoch keine zusätzlichen Aufwand.
Der Einsatz von gemischten Rechen- und Speicherressourcen und die Verteilung der Daten auf mehrere Ressourcen sowie das Scheduling durch den DOHS ermöglichen die Vermeidung unnötiger Datentransfers. Die Messwerte zeigen, dass bei großen Datensätzen eine dezentrale Speicherung und die Vermeidung der Datenübertragung, auch innerhalb eines Clusters oder einer Organisation, die Verarbeitungsgeschwindigkeit durch den DOHS enorm beschleunigt werden kann. Ebenso verdeutlichen

Test	**Anzahl CPUs**	**Grid**	**DOHS**
11694 Datensätze	10	16:56	15:32
11694 Datensätze	20	8:21	7:25
11694 Datensätze	40	7:28	4:43
11694 Datensätze	60	6:43	3:09
256530 Datensätze	10	465	419
256530 Datensätze	20	244	201
256530 Datensätze	40	211	118
256530 Datensätze	60	189	66

Tabelle 5.3: Verarbeitungszeiten mit und ohne Datenübertragung in Minuten.

die Messungen die weitaus bessere Skalierung der kombinierten Speicher- und Rechenressourcen sowie der Verarbeitung unter Vermeidung von Datentransfers, wobei sich die Verarbeitungsdauer bei 60 Prozessorkernen annähernd halbiert. Die schlechtere Skalierung in klassischen Grid-Umgebungen in den zwingend Daten transferiert werden müssen und die Transferdauer beim Scheduling innerhalb einer Organisation vernachlässigt wird, kann auf die sinkenden Transferraten der Speicherressource bei vielen parallelen Verbindungen und der Beschränkungen durch die Netzwerkbandbreite zurückgeführt werden. Eine Beschleunigung der Verarbeitung mit Datenübertragung wäre nur über zusätzliche Speicherressourcen zu erreichen, wodurch die Kosten erheblich gesteigert würden.

Kapitel 6

Alterungsanalysen aus Sensordaten

Lebensdaueruntersuchungen von neuen, komplexen Systemen werden bis heute häufig im Labor oder unter kontrollierten, stationären Bedingungen durchgeführt. Eine Übertragung dieser Untersuchungen auf den späteren, produktiven Einsatz ist aufgrund der Vielzahl von veränderbaren Umgebungsbedingungen kaum möglich. Daher ist die Untersuchung der Lebensdauer und der Verlässlichkeit der Systeme unter realen Bedingungen von großer Bedeutung.

Die Lebensdaueruntersuchung eines Systems erfordert eine regelmäßige Zustandsbestimmung. Idealerweise führt man hierzu für jede Einheit in regelmäßigen Abständen einen kontrollierten Test auf einem speziellen Prüfstand durch. Befinden sich die einzelnen Systeme jedoch im produktiven Betrieb in Kundenhand, scheiden solche Prüfstandstests zur Lebensdaueruntersuchung aus. Für diese Szenarien wurden Methoden entwickelt, die in regelmäßigen Abständen ein möglichst exaktes Modell des aktuellen Betriebszustandes des Systems aus den aufgezeichneten Sensordaten generieren. Die Alterung wird bei diesen Verfahren durch die Abweichung des Betriebsmodelles zu den Werten eines neuen Systems beschrieben.

Zur Anwendung der bestehenden Verfahren werden die Daten aller zu untersuchenden Systeme benötigt, so dass die meist lokal aufgezeichneten Daten zunächst in ein zentrales Data-Warehouse transferiert werden müssen. Dieser Ansatz erfordert einerseits einen aufwendigen Prozess, um die Daten zeitnah zu sammeln und zu verwalten. Andererseits wird die Datenmenge durch die zusätzlich zentrale Speicherung verdoppelt. Eine dezentrale Verwaltung und Verarbeitung der Daten, wie es das in dieser Arbeit entwickelte Umgebungsmodell und das Scheduling mit dem DOHS ermöglichen, eliminiert diese Problematiken. Im Rahmen dieser Arbeit wurden bestehende Verfahren auf die verteilte Verarbeitung adaptiert, sowie eine neue Methode zur Lebensdaueruntersuchung aus realen Sensordaten entworfen, die nicht den Betriebszustand, sondern direkt den Alterungszustand modelliert. Die Verfahren wurden dabei speziell für die effiziente Verarbeitung aller verfügbaren Sensordaten über Organisationsgrenzen hinweg konzipiert [FSN+08].

Die Lebensdaueruntersuchung eines Systemes ist mit der Alterungszustandsbe-

stimmung jedoch nicht abgeschlossen. Für die Weiterentwicklung der Systeme ist die Ermittlung der Lebensdauer beeinflussenden Faktoren beim Betrieb unter realen Bedingungen von entscheidender Bedeutung. Mit dem in dieser Arbeit entwickelten Verfahren lassen sich Hinweise auf mögliche Alterungsursachen aus den aufgezeichneten Sensordaten ermitteln [RKH07].

Die Verfahren zur Ermittlung des Alterungszustandes sowie zur Alterungsursachenanalyse können für beliebige Systeme eingesetzt werden. Im nachfolgenden Abschnitt werden zunächst die drei unterschiedlichen Verfahren zur Zustandsbestimmung von Systemen aus Sensordaten erläutert. Anschließend wird ein Verfahren zur Ermittlung der Alterungsursachen aus Sensordaten beschrieben. Darauf folgend wird die Anwendung dieser Methoden zur Alterungsanalyse von Brennstoffzellenfahrzeugen der Daimler AG vorgestellt.

6.1 Alterungszustandsbestimmung

Die Bestimmung des Alterungszustands eines Systems aus dem produktiven Betrieb aufgezeichneten Sensordaten kann mit unterschiedlichen Methoden erfolgen. Eines der gebräuchlichsten Verfahren lehnt sich an die Vorgehensweise bei der Alterungsbestimmung im Labor an. Dort wird das System in regelmäßigen Abständen detailliert untersucht, beziehungsweise auf einem speziellen Prüfstand unter kontrollierten Bedingungen betrieben. Dabei wird eine definierte Anzahl von Messgrößen, die *Leistungsindikatoren*, aufgezeichnet, welche die Leistungsfähigkeit und damit die Alterung des Systems beschreiben. Die Alterung kann anschließend als Differenz zwischen den Messwerten eines neuen Systems und den Werten dieser Zustandsmessungen errechnet werden.

Bei der Übertragung dieses Verfahrens auf die Alterungszustandsbestimmung aus im produktiven Einsatz aufgezeichneten Sensordaten muss insbesondere beachtet werden, dass meist keine Untersuchung und kein kontrollierter Prüfstandslauf zur Zustandsmessung genutzt werden können. Ein primitiver Ansatz zur Zustandsmessung besteht in der Suche nach *übereinstimmenden* Messwerten für alle Messgrößen, außer den Leistungsindikatoren, in den aufgezeichneten Sensordaten. Aus den übereinstimmenden Messungen können wieder die Differenzen zwischen den Leistungsindikatoren eines neuen und eines gealterten Systems berechnet werden. Für reale Systeme mit Hunderten von Messgrößen ist die Anzahl der übereinstimmenden Messungen jedoch so gering, dass dieser Ansatz nicht praktikabel ist.

Ein weitaus flexibler Ansatz nutzt ein Modell, das den Betriebszustand des Systems möglichst genau aus definierten Eingabemessgrößen approximiert. Aus den Eingabemessgrößen approximiert das Modell die Leistungsindikatoren. Die Menge der Eingabemessgrößen und der Leistungsindikatoren müssen dabei disjunkt sein. Das Modell wird in regelmäßigen Abständen oder fortlaufend auf die aufgezeichneten Sensordaten adaptiert und spiegelt somit immer den aktuellen Status des Systems wieder. Die Zustandsmessung erfolgt durch einen „virtuellen Prüfstandslauf“, bei dem vorgegebene Werte für alle Eingabemessgrößen in das Modell eingespeist wer-

den. Die Differenzen zwischen den Modellausgaben zu Beginn der Datenaufzeichnung und den aktuellen Messwerten ergeben den Verschleiß des Systems.

Die Bestimmung des Modells aus den aufgezeichneten Sensordaten kann in Abhängigkeit des gewählten Verfahrens sowohl datenintensiv als auch rechenintensiv sein. Die nachfolgend vorgestellten Verfahren zur Bestimmung des Alterungszustands sind daher speziell für die skalierbare, dezentrale Verarbeitung der Sensordaten in organisationsübergreifenden Umgebungen entwickelt worden. Das erste Verfahren bestimmt den Zustand eines Systems anhand theoretischer, physikalischer Modelle, deren Parameter aus den Daten mithilfe numerischer Optimierungsverfahren ermittelt werden. Das zweite Verfahren nutzt Künstliche Neuronale Netze zur Modellierung der Systeme.

Anstatt den aktuellen Betriebszustand eines Systems zu approximieren und anschließend die Abweichung zu bestimmen, kann, wie das dritte Verfahren zeigt, auch direkt die Alterung eines Systems modelliert werden. Bei diesem Verfahren fließen in das Modell neben bestimmten Eingabemessgrößen auch Zeitinformationen, etwa die Betriebszeit, ein. Aus den aufgezeichneten Sensordaten werden gezielt Messbereiche ausgewählt, bei denen die Leistungsindikatoren die deutlichsten Veränderungen, etwa unter Spitzenlasten, zeigen. Das Modell erlernt aus den Eingabemessgrößen und den zugehörigen Zeitinformationen eine Approximation der Leistungsindikatoren dieser Messbereiche. Die Alterung des Systems kann mit diesem Modell einfach durch die Modellausgabe zu unterschiedlichen Zeitpunkten bestimmt werden.

6.1.1 Zustandsbestimmung mit Betriebsmodellen

Die Alterungszustandsbestimmung in organisationsübergreifenden Umgebungen erfordert die Anpassung der bestehenden Verfahren, um eine dezentrale Verarbeitung zu ermöglichen. Bei der Alterungszustandsbestimmung auf Basis verteilter Sensordaten wird zunächst ein Modell für die Systeme definiert, anschließend werden die Parameter des Modells für alle Datensätze jedes Systems berechnet und zuletzt wird die Alterung anhand der generierten Modelle bestimmt.
Die Berechnung der Betriebsmodelle kann, wie in nachfolgend erläutert, durch Künstliche Neuronale Netze oder Optimierungsverfahren erfolgen, wobei beide Methoden eigene Vor- und Nachteile aufweisen.

Optimierungsverfahren für Betriebsmodelle

Für die Analyse komplexer Systeme sind Modelle ein wichtiges Hilfsmittel. Ein Modell approximiert das reale System und beschreibt dessen Verhalten durch eine möglichst geringe Menge von Parametern. Allgemein lässt sich dies wie folgt beschreiben:

- Das Verhalten des Systems kann durch k Messgrößen ($a = [a_1, a_2, ..., a_k]^T$) beschrieben werden.
- Es wurden m Messungen durchgeführt. Es werden also $m \times k$ Messwerte aufgezeichnet.

- Das Modell $M(p, a)$ mit den Parametern ($p = [p_1, p_2, ...]^T$) beschreibt den Zusammenhang zwischen den k Messgrößen und dem wahren Verhalten O: $M(p, a) = O + e$, wobei e den Fehler des Modells bezeichnet.

Die Anpassung eines Modells an das System erfolgt durch die Berechnung der Modellparameter auf Basis der Messwerte des Systems. Ziel dieser Parameterbestimmung ist es nun, den Fehler e zu verringern, indem ein Fehlerkriterium, etwa die Summe der Fehlerquadrate, aller Messungen minimiert wird:

$$F(p) = \frac{1}{2} \sum_{i=1}^{m} (M(p, a_i) - O_i)^2 \quad \longmapsto \quad min! \tag{6.1}$$

Es existieren unterschiedliche Ansätze zur Lösung dieses Minimierungsproblems, wobei im Folgenden das Levenberg-Marquardt Verfahren [Lev44][Mar63] eingesetzt wird. In vielen realen Problemstellungen unterliegen die Parameter p des Modells physikalisch vorgegebenen Restriktionen. Für die nachfolgenden Anwendungen sind hierbei insbesondere Wertebereiche ($p_i \in [x; y]$) von Bedeutung. Das Levenberg-Marquardt Verfahren berücksichtigt allerdings keine Nebenbedingungen, so dass aus den verschiedenen Verfahren zur nichtlinearen Optimierung unter Nebenbedingungen [KYF02], die Projektionsmethode von Rosen [Ros61] gewählt wurde.

Die Anwendung des Levenberg-Marquardt Optimierungsverfahrens zur Alterungszustandsbestimmung auf verteilten Datensätzen erfolgt in mehreren Schritten. Im ersten Schritt definiert der Benutzer ein beliebiges, meist physikalisch motiviertes, Modell M des Systems mit den Modellparametern p, den ausgewählten Eingabemessgrößen aus den Sensordaten a und den aufgezeichneten Ausgabedaten O. Im zweiten Schritt werden die Parameter des Modells mithilfe des Levenberg-Marquardt Verfahrens aus den Daten bestimmt. Die Berechnung erfolgt für jeden Datenblock separat, so dass mehrere Datenblöcke auf unterschiedlichen Ressourcen gleichzeitig verarbeitet werden können. Für jeden Datenblock werden die ermittelten Parameter mit den zugehörigen Start- und Endzeitpunkten gespeichert.

Im letzten Schritt kann aus den berechneten Modellparametern mit einem virtuellen Prüfstandslauf der Zustand des Systems bestimmt werden. Der Prüfstandslauf wird hierbei über eine Reihe vordefinierte Werte $s = (s_1, s_2, \ldots, s_n)$ für die Eingabemessgrößen festgelegt. Die Alterung zu einem Zeitpunkt t lässt sich somit als die Differenz $D(t, s)$ zwischen den virtuellen Prüfstandsläufen des neuen $M(p_0, s_i)$ und des gealterten $M(p_t, s_i)$ Systems beschreiben,

$$\begin{aligned} D(t, s_1) &= M(p_0, s_1) - M(p_t, s_1) \\ D(t, s_2) &= M(p_0, s_2) - M(p_t, s_2) \\ &\ldots \\ D(t, s_n) &= M(p_0, s_n) - M(p_t, s_n) \end{aligned}$$

wobei p_0 die Parameterwerte zum Startzeitpunkt und p_t die berechneten Parameterwerte zum Zeitpunkt t bezeichnen.

Eine weitere Methode der Alterungszustandsbestimmung betrachtet anstatt des gesamten Modells nur die Entwicklung einzelner Parameter über die Zeit. Beschreibt der Parameter eine zentrale Eigenschaft des Systems, zeigt dieses Verfahren den Zustand der zugehörigen Komponenten und lässt so Rückschlüsse auf mögliche Alterungsursachen zu.

Künstliche Neuronale Netze für Betriebsmodelle

Das im vorherigen Abschnitt vorgestellte Verfahren bietet eine Zustandsbestimmung auf Basis von theoretisch fundierten Modellen. Neben der reinen Zustandsbestimmung lassen diese Modelle zum Teil auch direkte Rückschlüsse auf Degradationsursachen zu. Die Formulierung eines solchen Modells ist jedoch aufgrund der komplexen Zusammenhänge der einzelnen Komponenten eines Systems sehr aufwendig. Die Integration weiterer potenzieller Einflussgrößen in ein Modell gestaltet sich ebenfalls kompliziert.

In der vorgestellten Arbeit wurde daher ein Verfahren entwickelt, das aus beliebig vielen potenziellen Einflussgrößen ein Betriebsmodell generiert. Die Wechselwirkungen und Abhängigkeiten der einzelnen Einflussgrößen werden bei diesem Verfahren durch Künstliche Neuronale Netze (KNN) abgebildet. Ein KNN erlernt, aus vorgegebenen Ein- und Ausgabesignalen, ein Modell des diesen Daten zugrunde liegenden Mechanismus. Im Gegensatz zu den Optimierungsverfahren werden hierbei nicht die Parameter eines vorgegebenen Modells aus den Daten bestimmt, sondern das gesamte Modell wird aus den Daten abgeleitet. Darüber hinaus kann gezeigt werden, dass KNN in der Lage sind jede beliebige Funktion (Modell) beliebig genau zu approximieren [Cyb88]. Aufgrund ihrer hohen Abbildungsleistung erzeugen KNN zwar sehr genaue Modelle, diese sind jedoch kaum interpretierbar, so dass KNN-Modelle keine Rückschlüsse auf mögliche Degradationsursachen zulassen.

Eine gebräuchliche Methode KNN-Modelle zur Zustandsbestimmung zu verwenden, besteht darin, das KNN nur auf den Daten der ersten Betriebsstunden aller zu untersuchenden Systeme zu trainieren. Das so erzeugte KNN-Modell repräsentiert somit das Verhalten eines neuen Systems. Anschließend wird das Modell in regelmäßigen Abständen auf die aufgezeichneten Daten der einzelnen Systeme angewandt. Weicht das Modell zu stark von den aufgezeichneten Daten ab, wird es auf die neuen Daten dieses Systems adaptiert. In regelmäßigen Abständen wird ein simulierter Prüfstandslauf zur Ermittlung des Systemzustandes durchgeführt. Hierzu werden die aus realen Prüfstandsläufen ermittelten Werte aller verwendeten Eingabesignale in das KNN eingespeist. Der aktuelle Zustand des Systems wird, wie bei den im vorherigen Abschnitt vorgestellten Verfahren, anhand der Degradation der Leistungsindikatoren bestimmt. Dies entspricht der Differenz zwischen den Ausgabesignalen eines neuen Modells M_0 und eines Modells zum Zeitpunkt t des simulierten Prüfstandslaufs $s = (s_1, s_2, \ldots, s_n)$:

$$D(t, s_i) = M_0(s_i) - M_t(s_i)\ , \quad 1 \leq i \leq n \tag{6.2}$$

Der Vorteil dieses Verfahrens liegt in den moderaten Prozessoranforderungen der

Simulation eines Prüfstandslaufs und der Adaption des Modells. Existiert ein allgemeines KNN-Modell des Systems, eignet sich das Verfahren auch für die Implementierung in einem Steuergerät, solange die Adaption nur selten erfolgt. Dem spricht entgegen, dass das Modell so häufig wie möglich adaptiert werden sollte, um möglichst viele Sensordaten in die Modellbildung zu integrieren und eine gleichbleibende Genauigkeit der simulierten Prüfstandsläufe zu garantieren. Ein weiteres Problem stellt die Überanpassung des Modells an die aktuellen Sensordaten dar, wobei zum einen der Fehler bei den Simulationen steigt und zum anderen häufige Adaptionen durchgeführt werden müssen. Darüber hinaus müssen zur Generierung des KNN-Modells die Daten aller Systeme zentral gesammelt und verarbeitet werden.

Das in dieser Arbeit entwickelte Verfahren zur Bestimmung des Betriebszustandes auf Basis von KNN vermeidet diese Nachteile, indem es alle Daten mit einbezieht. Der Ablauf entspricht hierbei weitgehend dem des Optimierungsverfahrens und ist ebenfalls speziell auf die skalierbare, dezentrale Verarbeitung angepasst. Im ersten Schritt werden der Aufbau sowie die Eingabesignale des KNN vom Benutzer spezifiziert. Anschließend wird für die gewählten Eingabesignale der virtuelle Prüfstandslauf definiert, der meist aus den Werten der Eingabesignale eines realen, aufbereiteten Prüfstandslauf basiert. Für jeden Datensatz eines zu untersuchenden Systems wird separat ein KNN trainiert. Im darauf folgenden Schritt wird der vorgegebene Prüfstandslauf auf allen generierten KNN simuliert. Die resultierenden Ausgabewerte des Prüfstandslaufs werden mit den zugehörigen Start- und Endzeitpunkten gespeichert. Die Alterung kann damit als Differenz der Modelle zu unterschiedlichen Zeitpunkten, analog zum Optimierungsverfahren, bestimmt werden. Das hier vorgestellte Verfahren ist deutlich aufwendiger als das zuvor beschriebenen Verfahren. Dafür nutzt es alle Daten zum Training, so dass alle vorhandenen Informationen bei der Modellbildung berücksichtigt werden.

Ein entscheidender Nachteil der vorgestellten Betriebsmodelle sind die unter bestimmten Umständen auftretenden Überanpassungs- und Extrapolationsprobleme. Beide Modelle nutzen zur Adaption, beziehungsweise zum Training, nur Ausschnitte der gesamten Daten eines Systems, so dass eine Überanpassung des Betriebsmodells an bestimmte Messgrößen oder Messwertzustände aus diesen Datensätzen erfolgen kann. Es ergeben sich dabei insbesondere große Fehler, wenn die zum Training / zur Adaption verwendeten Datensätze keine Messwerte im Wertebereich des simulierten Prüfstandslaufs enthalten, so dass die KNN in diesem Fall extrapolieren müssen. Anstelle der Spannungsdegradation wird in diesem Fall der Unterschied zwischen den Modellen M_0 und M_t und nicht die Degradation gemessen. Das nachfolgend beschriebene Alterungszustandsmodell umgeht diese Problematik, indem es alle Daten eines Systems zur Modellbildung verwendet.

6.1.2 Alterungszustandsmodell

Die Alterungszustandsbestimmung von komplexen Systemen folgt meist dem oben beschriebenen Verfahren, bei dem zunächst ein möglichst genaues Modell des Betriebszustandes generiert und anschließend die Abweichung zum Startwert ermittelt

wird. Anstatt den aktuellen Betriebszustand zu approximieren und anschließend die Abweichung zu bestimmen, kann, wie das in dieser Arbeit entwickelte Verfahren zeigt, ein KNN auch direkt zur Modellierung des Alterungszustandes verwendet werden. Die von den beschriebenen Verfahren verwendete Alterungsdefinition beschreibt die Alterung zum Zeitpunkt t bei einem definierten Prüfstandslauf s, als Differenz zwischen der vom Modell ermittelten Werte zum Betriebsstart und dem Zeitpunkt t.

Das im Folgenden beschriebene Verfahren zeigt, wie sich diese Alterungsdefinition direkt durch ein KNN abbilden lässt. Die Grundidee besteht darin, einem KNN neben den Sensorinformationen auch Zeitinformationen (Betriebszeit, Lebenszeit, ...) als Eingabesignale zu übergeben. Das KNN lernt damit sowohl die Zusammenhänge der Sensorinformationen als auch deren zeitliche Abhängigkeiten. Der Zustand eines Systems zu einem gewissen Zeitpunkt lässt sich mit diesem Modell einfach ermitteln. In das Modell werden vorgegebene Sensorinformationen eines aussagekräftigen Betriebszustandes b ein Mal mit den Zeitinformationen des Betriebsstarts und ein Mal mit den Zeitinformationen eines späteren Zeitpunkts t eingegeben und die Differenz gebildet:

$$D(b,t) = M(b,0) - M(b,t) \tag{6.3}$$

Die Umsetzung dieses Verfahrens erfolgt in drei Schritten. Vor der Berechnung wird ein bestimmter Betriebszustand b der Sensorinformationen (a_1^b, a_2^b, ...) sowie die zu verwendenden Zeitinformationen definiert.
Im ersten Schritt der Verarbeitung werden alle aufgezeichneten Sensordaten eines Systemes parallel aufbereitet. Da der gewählte Betriebszustand sehr selten exakt auftritt, werden die ausgewählten Sensorinformationen für das Training um den definierten Betriebszustand extrahiert ($a_1^b \pm \Delta a_1,\ a_2^b \pm \Delta a_2,\ ...$). In einem Kombinationsschritt werden die Resultate der einzeln verarbeiteten Datensätze zu einem Datensatz pro System zusammengefasst.
Anschließend wird für jedes System ein KNN trainiert. Das Training erfolgt durch Kreuzvalidierung, so dass unterschiedliche Daten eines Datensatzes abwechselnd zum Training und zur Validierung verwendet werden. Ein entscheidender Vorteil gegenüber den zuvor vorgestellten Betriebsmodellen besteht in der Ausnutzung aller vorhandenen Daten im Trainingsprozess, so dass Überanpassungen auf kurzzeitige Anomalien vermieden werden.
Nach Beendigung des Trainings erfolgt die Alterungszustandsbestimmung. In das erzeugte KNN-Modell werden hierbei die Sensorinformationen des ausgewählten Betriebszustands b und Zeitinformationen t eingespeist. Die Zeitinformationen werden über die gesamte Lebenszeit des Systems variiert, während die Sensorinformationen konstant gehalten werden. Das Ergebnis dieser Berechnung ist eine Zeitreihe der Leistungsindikatoren beim Betriebszustand b. Aus dieser Zeitreihe kann die Leistungsdegradation zwischen zwei Zeitpunkten direkt abgelesen werden.
Das berechnete KNN kann darüber hinaus in begrenztem Umfang auch zur Vorhersage des Degradationsverhaltens verwendet werden. Hierzu werden dem KNN zukünftige Zeitinformationen übergeben. Die so ermittelten Werte sind jedoch kritisch zu bewerten, da die Güte der Extrapolation eines KNN nur schwer abschätzbar

ist[Mit97].

6.2 Ursachenanalyse

Viele Arbeiten zur Untersuchung des Alterungsverhaltens von Systemen im produktiven Einsatz beschränken sich auf Modelle zur Zustandsbestimmung. Auf die zentrale Frage nach den Ursachen der Alterung bieten diese Verfahren jedoch keine Antworten. Die in dieser Arbeit entwickelte Ursachenanalyse stellt eine Methode zur Berechnung möglicher Alterungsursachen aus den aufgezeichneten Sensordaten zur Verfügung [RKH07].

6.2.1 Degradationsursachenmodell

Die zentrale Fragestellung bei der Untersuchung der Degradation eines Systems ist, wie sich der aktuelle, durch die Sensordaten beschriebene Systemzustand auf das Alterungsverhalten auswirkt. In der nachfolgenden Betrachtung des Degradationsursachenmodells umfasst der Systemzustandsvektors $x(t)$ alle Sensorinformationen zum Zeitpunkt t. Die gemessene oder berechnete Degradation kann aus mehreren Degradationsmessgrößen bestehen, so dass der Degradationszustandsvektor $w(t)$ die Degradation zum Zeitpunkt t beschreibt.

Betrachtet man den Degradationsprozess eines Systems im produktiven Einsatz, so sind, im Gegensatz zu Laborversuchen, vor allem die speziellen Umgebungs- und Systemzustände von Interesse. Unter diesem Gesichtspunkt ist es für die Ursachenanalyse naheliegend, den Einfluss des Systemzustandsvektors x auf den Degradationszustandsvektor w zur Beschreibung des Degradationsprozesses zu verwenden. Dieser Prozess kann durch eine Differenzialgleichung modelliert werden:

$$\dot{w}(t) = f(w(t), x(t)) \tag{6.4}$$

Die Abhängigkeit der Degradationsveränderung vom Degradationszustandsvektor spiegelt dabei die Möglichkeit einer nicht-linearen Degradation wieder. Die Degradationsrate für einen bestimmten Systemzustand kann damit für verschiedene Degradationsstadien unterschiedlich sein.

Die in einem Zeitintervall $[t_0, t_1]$ entstehende Degradation ergibt sich mit diesem Modell zu:

$$\Delta w = w(t_1) - w(t_0) = \int_{t_0}^{t_1} f(w(t), x(t))\, dt \tag{6.5}$$

Mit der a priori unbekannten Degradationsfunktion f, die aus den Messwerten ermittelt werden muss.

Im Allgemeinen ist es schwierig, die Funktion f direkt aus den beobachteten Systemzustands- und Degradationsvektoren zu identifizieren. Unter bestimmten Annahmen ist diese Aufgabe jedoch einfacher,

$$\dot{w}(t) = f(w(t), x(t)) = g(w(t)) \sum_j a_j h_j(x_j(t)) \tag{6.6}$$

wobei die Funktion $f(w(t), x(t))$ in diesem Fall aus einer Funktion des Degradationszustandes $g(w(t))$ sowie einer Linearkombination der Zustandsvariablen x_j mit unbekannten Koeffizienten a_j besteht. Die Funktion beschreibt somit den *linearen* Einfluss der Zustandsvariablen x_j auf die Degradationsrate $\dot{w}$ in Abhängigkeit des Degradationszustandes $g(w)$. Die vorgegebenen Funktionen h_j können dabei, etwa zur Normierung, auf die einzelnen Zustandsvariablen angewendet werden.

Setzt man 6.6 in 6.5 ein, erhält man:

$$\Delta w = w(t_1) - w(t_0) = \int_{t_0}^{t_1} g(w(t)) \sum_j a_j h_j(x_j(t))\, dt \tag{6.7}$$

Die direkte Bestimmung des Integrals 6.7 mit der unbekannten Funktion g und den unbekannten Koeffizienten a ist äußerst schwierig. Für die Zusammenhangsanalyse zwischen Degradation und Systemzustandsvariablen hilft die genaue Betrachtung des Degradationsprozesses, der verglichen mit der Änderung des Systemzustandes, relativ langsam ist. Für Paare von Degradationsmessungen $w(t_{0i}), w(t_{1i})$ in kurzen Zeitabschnitten $\Delta t = t_{1i} - t_{0i}$ kann die Degradation in diesen Intervallen als konstant angesehen werden. Die Funktion $w(t)$ lässt sich in diesem Fall durch die Konstante $w_i = (w(t_{0i}) + w(t_{1i}))/2$ substituieren. Das Integral 6.5 kann mit den Funktionsannahmen aus 6.7 wie folgt zerlegt werden:

$$\Delta w_i = w(t_{1i}) - w(t_{0i}) = \int_{t_{0i}}^{t_{1i}} f(w(t), x(t))\, dt = g(w_i) \sum_j a_j \int_{t_{0i}}^{t_{1i}} h_j(x_j(t))\, dt \tag{6.8}$$

Anstelle des Integrals der Funktion f genügt es unter dieser Annahme, die Integrale der Funktionen der Zustandsvariablen $h_j(x_j(t))$ zwischen den Zeitpunkten t_{0i} und t_{1i} zu bestimmen. Im Folgenden werden diese Integrale mit

$$h_{ji} = \int_{t_{0i}}^{t_{1i}} h_j(x_j(t))\, dt$$

bezeichnet.

Die Aufgabe besteht nun darin, die Funktionen $g(w)$, in Form einer geeigneten Funktionsapproximation, sowie den besten Koeffizientenvektor a für

$$\Delta w_i = g(w_i) \sum_j a_j h_{ji} \tag{6.9}$$

mit den aufgezeichneten bzw. berechneten konstanten Vektoren Δw, w_i und h_{ji}, zu finden.

Im Falle einer einzigen Degradationsmessgröße wird aus dem Vektor w eine skalare Konstante. Wählt man, im einfachsten Fall, eine lineare Funktionsapproximation $g(w_i) = bw_i + c$ ergibt dies folgendes Approximationsproblem:

$$\Delta w_i = (bw_i + c) \sum_j a_j h_{ji} \tag{6.10}$$

Die Schätzung der freien Parameter b, c und a_j kann entweder simultan, etwa mithilfe von Maximum Likelihood Methoden, oder mit einem Fixpunktalgorithmus, der abwechselnd die Funktion und den Koeffizientenvektor schätzt, erfolgen.

Der in dieser Arbeit verwendete Fixpunktalgorithmus besteht aus folgenden Schritten:

1. Starte mit der initialen Schätzung $b = 0$ und $c = 1$. Dies entspricht der Annahme, dass der aktuelle Degradationszustand w_i keinen Einfluss auf die Degradationsrate hat.

2. Halte b und c konstant und berechne die Komponenten a_j des Koeffizientenvektors durch Multivariate Lineare Regression.

3. Halte die a konstant und berechne b und c durch Lineare Regression.

4. Bestimme die Approximationsgenauigkeit der k-ten Iteration: $\delta_{ki} = \Delta w_i - (bw_i + c) \sum_j a_j h_{ji}$

5. Falls $\sum_i \delta_{ki}^2 - \sum_i \delta_{k-1i}^2 \geq \epsilon$ für eine definierte Fehlerschranke ϵ gilt, gehe erneut zu Schritt 2. Ansonsten ist der Algorithmus beendet.

Mithilfe dieses Degradationsmodells lassen sich direkt Rückschlüsse auf degradationsverursachende Zustandsvariablen ziehen. Wählt man die oben beschriebene Approximation 6.10, bieten die Korrelationskoeffizienten

$$r_j = Corr(\frac{\Delta w_i}{bw_i + c}, h_{ji}) \tag{6.11}$$

unmittelbar Hinweise auf mögliche Degradationsursachen.

Es ist wichtig darauf hinzuweisen, dass die Daten zur Bestimmung der freien Parameter der Approximation 6.10 mehrere unterschiedliche Objekte, repräsentieren müssen.
Für einzelne Objekte, mit notwendigerweise nicht überlappenden Zeitintervallen $[t_{0i}, t_{1i}]$, kann die Approximation 6.9 perfekt bestimmt werden, wenn eine Funktion mit der Eigenschaft $g(w_i) = \Delta w_i$ gewählt wird. Die Approximation ist damit unabhängig von den ausgewählten Zustandsvariablen und deren Werten, so dass keinerlei Rückschlüsse auf Degradationsursachen mehr möglich sind.
Im Allgemeinen kann für mehrere Objekte keine solche Funktion gefunden werden, da die Degradation in einem bestimmten Zeitintervall für unterschiedliche Objekte unterschiedlich groß ist.

6.3 Alterungsanalysen von Brennstoffzellenfahrzeugen

Die Suche nach Alternativen zu herkömmlichen Antriebskonzepten auf Basis fossiler Rohstoffe sowie die weltweite Verschärfung von Emissionsschutzgesetzen fördert

die Entwicklung von emissionslosen Fahrzeugtechnologien. Brennstoffzellenfahrzeuge sind dabei eine vielversprechende Alternative zu den heutigen Verbrennungsmotoren. Die F-Cell Flotte der Daimler AG dient der Weiterentwicklung und Vorbereitung der Brennstoffzellenfahrzeuge hin zur Großserienproduktion. Eines der zentralen Themen ist hierbei die gleiche Lebensdauer und Verlässlichkeit aktueller Verbrennungsmotoren zu erreichen.

Lebensdaueruntersuchungen von Brennstoffzellen werden bis heute fast ausschließlich im Labor oder unter kontrollierten, stationären Bedingungen durchgeführt[LLA98]. Eine Übertragung dieser Untersuchungen auf Brennstoffzellen in Fahrzeugen ist aufgrund der Vielzahl sich permanent verändernden Umgebungsbedingungen kaum möglich. Daher ist die Untersuchung der Lebensdauer und der Verlässlichkeit der Fahrzeuge der F-Cell Flotte für die Weiterentwicklung der Brennstoffzellentechnologie in Fahrzeugen von großer Bedeutung.

Die Lebensdaueruntersuchung einer Brennstoffzelle erfordert eine regelmäßige Zustandsbestimmung des gesamten Brennstoffzellensystems. Idealerweise führt man hierzu für jedes Fahrzeug in regelmäßigen Abständen einen kontrollierten Test auf einem speziellen Prüfstand durch. Da sich die Fahrzeuge der F-Cell Flotte größtenteils in Kundenhand befinden, scheiden solche Prüfstandstests zur Lebensdaueruntersuchung der gesamten Flotte aus. Die Daimler AG als auch andere Forschungseinrichtungen befassen sich daher mit Lebensdaueruntersuchungen von Brennstoffzellen anhand der beim Kundenbetrieb aufgezeichneten Daten. Die von der Daimler AG bislang entwickelten Methoden basieren alle auf Betriebsmodellen [Fri09][NSW+05][NSWP05] und wurden auf die Implementierung in Steuergeräten optimiert.
Im Rahmen dieser Arbeit wurden die in den vorhergegangenen Abschnitten beschrieben Verfahren im FDA-Miner umgesetzt. Die Verfahren wurden dabei speziell für die effiziente Verarbeitung der Daten der gesamten Fahrzeugflotte konzipiert. Darüber hinaus nutzen die in dieser Arbeit vorgestellten Verfahren alle aufgezeichneten Fahrtdaten, so dass im Gegensatz zu den erwähnten Ansätzen alle vorhandenen Informationen in die Analyse einfließen[FSN+08].

Die Lebensdaueruntersuchung eines Brennstoffzellenfahrzeugs ist mit der Alterungszustandsbestimmung jedoch nicht abgeschlossen. Für die Weiterentwicklung der Brennstoffzellentechnologie ist die Ermittlung der Lebensdauer beeinflussenden Faktoren beim Betrieb einer Brennstoffzelle in einem Fahrzeug von entscheidender Bedeutung. Die Bestimmung der Alterungsursachen stellt eine neue bislang ungelöste Herausforderung bei der Analyse der Brennstoffzellenfahrzeuge dar. Mit dem in dieser Arbeit beschriebenen Verfahren lassen sich Hinweise auf mögliche Alterungsursachen aus den aufgezeichneten Fahrtdaten ermitteln [RKH07].

Im nachfolgenden Abschnitt wird der Aufbau der Daimler Brennstoffzellenfahrzeuge sowie die unterschiedlichen Alterungsmechanismen von Brennstoffzellen erläutert. Im darauf folgenden Abschnitten werden die Implementierung der drei unterschiedlichen Verfahren zur Zustandsbestimmung von Brennstoffzellen aus den Fahrtdaten der F-Cell Flotte erläutert. Abschließend wird die Umsetzung des Verfahrens zur Ermittlung der Alterungsursachen von Brennstoffzellenstacks im Fahrzeugbe-

trieb beschrieben.

6.3.1 F-Cell Brennstoffzellenfahrzeuge

Die im Rahmen dieser Arbeit untersuchten Brennstoffzellenfahrzeuge der F-Cell Flotte basieren auf der Mercedes-Benz A- und B-Klasse. Wie in Abbildung 6.1 dargestellt, wird bei den Fahrzeugen der herkömmliche Antriebsstrang durch die Komponenten des Brennstoffzellensystems ersetzt. Das Brennstoffzellenmodul, be-

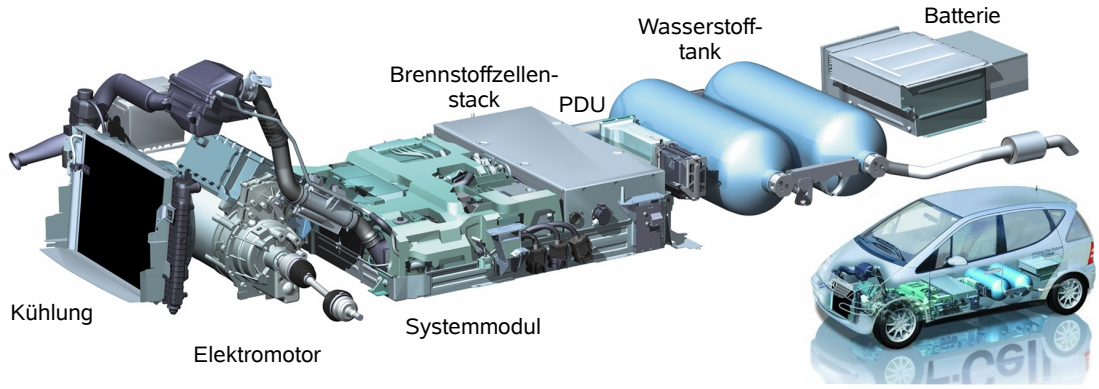


Abbildung 6.1: Antriebsstrang eines Brennstoffzellenfahrzeugs [Tru03].

stehend aus Brennstoffzellenstack (kurz: Stack), Kompressor und Steuerungselektronik bildet die zentrale Komponente des Antriebstrangs. Der benötigte Brennstoff, hier Wasserstoff, wird in speziellen Drucktanks gespeichert. Die Power Distribution Unit (PDU) verteilt die elektrische Energie zwischen der Batterie, dem Brennstoffzellenstack und dem Elektromotor. Die Batterie dient hierbei einerseits der Rückgewinnung von Bremsenergie sowie andererseits als Puffer zwischen Stack und Motor. Den eigentlichen Antrieb des Fahrzeugs übernimmt anstatt eines Verbrennungsmotors ein Elektromotor, der ohne Getriebe auskommt. Über das Kühlsystem werden sowohl der Elektromotor als auch der Stack im optimalen Temperaturbereich gehalten.

Erst durch das koordinierte Zusammenwirken all dieser Komponenten ist der Betrieb eines Brennstoffzellenfahrzeuges möglich. Das Brennstoffzellenmodul und insbesondere der Stack sind bei der Lebensdaueruntersuchung von entscheidender Bedeutung. Daher werden nachfolgend der Aufbau, die Funktionsweise, sowie die unterschiedlichen Alterungsphänomene von Brennstoffzellenstacks erläutert.

Aufbau und Funktion einer Brennstoffzelle

Brennstoffzelle sind effiziente elektrochemische Energieumwandler, die aus Wasserstoff und Sauerstoff Strom, Wasser und Wärme erzeugen [Sch05]. Es existieren verschiedene Brennstoffzellentypen, wobei die Proton Exchange Membrane Fuel Cell

(PEMFC) die gebräuchlichste für den Einsatz in Brennstoffzellenfahrzeugen ist. PEMFCs werden bei relativ niedrigen Temperaturen (zwischen 60° und 100° Grad Celsius) betrieben und erreichen eine Leistungsdichte von mehr als 1 KW pro KG Brennstoffzellengewicht. Die Leistung aktueller Brennstoffzellenstacks für Fahrzeuge reicht von 75 KW bis zu mehr als 100 KW.

Eine Brennstoffzelle besteht prinzipiell aus einem Elektrolyt, der mit einer Anode und einer Kathode gekoppelt ist. Auf der Anodenseite wird der Brennstoffzelle über eine Gasdiffusionselktrode Wasserstoff, auf Kathodenseite Luft oder reiner Sauerstoff zugeführt. Der Elektrolyt besteht typischerweise aus einer festen Polymermembran (Nafion) und ist beidseitig mit einem Katalysator (Platin) beschichtet. Auf der Anodenseite werden die H_2-Moleküle unter Abgabe von zwei Elektronen zu zwei Protonen oxidiert. Die frei gewordenen Elektronen gelangen über einen elektrischen Verbraucher zur Kathode. Die Protonen diffundieren durch die Polymermembran und verbinden sich auf Kathodenseite mit den Elektronen und dem Luftsauerstoff zu Wasser.
Die Protonleitfähigkeit der Nafion-Polymermembran hängt aufgrund der chemischen Eigenschaften von Nafion stark vom Feuchtigkeitsgrad ab. Die Gase werden aus diesem Grund üblicherweise vor ihrer Einleitung in die Brennstoffzelle befeuchtet.

Eine einzelne Brennstoffzelle liefert theoretisch eine maximale Spannung von $1,23V$. In der Praxis erreichen Brennstoffzellen Spannungen um $1V$, so dass für die in Fahrzeugen benötigte Spannung ($> 100V$) mehrere Zellen in Reihe geschaltet werden. Der aus dieser Reihenschaltung resultierende *Brennstoffzellenstack* versorgt das komplette Fahrzeug, etwa den Elektromotor und die Klimaanlage, mit elektrischer Energie. Eine nicht leitende Kühlflüssigkeit zirkuliert durch den gesamten Stack und führt die bei der Oxidation entstehende Wärme ab.

Die Polarisationskurve einer Brennstoffzelle

Aus dem charakteristischen Zusammenhang zwischen Strom und Spannung einer Brennstoffzelle, der Polarisationskurve, lässt sich der aktuelle Zustand einer Brennstoffzelle bestimmen. Die in Abbildung 6.2 schematisch dargestellte Polarisationskurve zeigt, dass mit steigender Stromentnahme die Spannung, aufgrund von Polarisationseffekten, immer weiter sinkt. Dem entsprechend bricht die Leistung der Brennstoffzelle bei sehr hoher Stromentnahme ein. Die Polarisationskurve (Spannung U in Abhängigkeit des Stroms I) einer Brennstoffzelle lässt sich mathematisch durch folgende Gleichung beschreiben [LCH+01]:

$$U(I) = U_0 - \mu(I) - \omega(I) - \eta(I) \tag{6.12}$$

Die Ruhespannung U_0 ist hierbei die maximale Spannung der Zelle ohne Last. $\mu(I)$, $\omega(I)$ und $\eta(I)$ beschreiben die einzelnen Spannungsverluste beim Strom I.

Der Aktivierungsverlust beeinflusst die Spannung insbesondere bei niedrigen Strömen und ist auf den limitierten Ladungstransfer und andere Aktivierungsprozesse zurückzuführen

$$\mu(I) = \frac{G \cdot T}{a \cdot z \cdot F} \cdot ln(I) \tag{6.13}$$

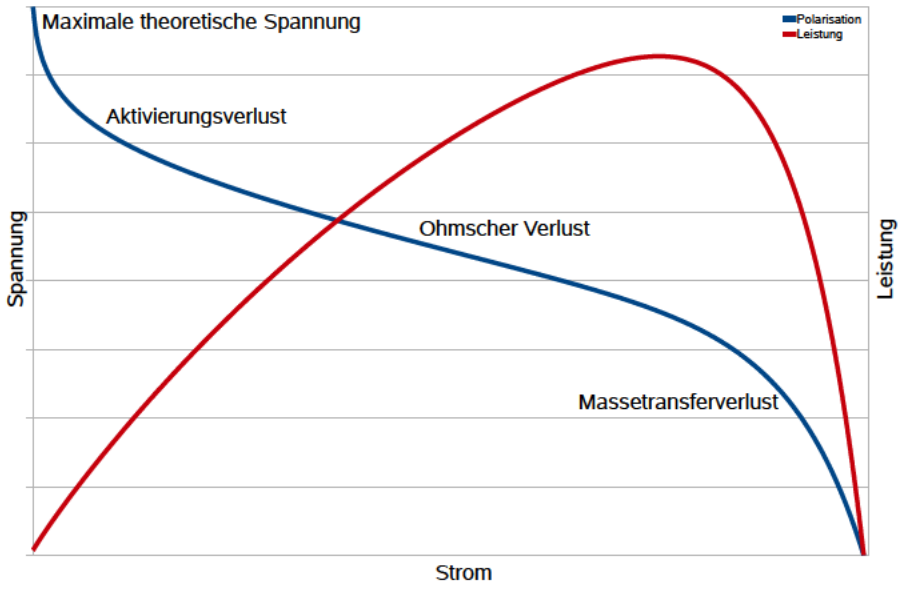


Abbildung 6.2: Polarisations- und Leistungskurve einer Brennstoffzelle, nach [LCH+01].

wobei I die Stromdichte (in mA/cm^2, G die universelle Gaskonstante ($8.3143 J/mol \cdot K$), T die Zellentemparatur (in Kelvin), a der Austauschkoeffizient (0.5), z die Anzahl der Elektronen pro Wasserstoffmolekül (2) und F die Faradaykonstante ($96485 C/mol$) darstellen.

Der ohmsche Verlust

$$\omega(I) = R_m \cdot I \tag{6.14}$$

beschreibt die Auswirkung des Widerstands der Nafion-Polymermembran R_m (in $kOHM/cm^2$) auf die Zellspannung. Der Membranwiderstand wird maßgeblich von der Protonenleitfähigkeit der Membran beeinflusst, welche wiederum stark vom Feuchtigkeitsgrad abhängt. Dieser beruht auf einer Vielzahl von Faktoren wie Temperatur, Gasfeuchte, Flussrate und Gasverteilung.

Der Massentransferverlust ist direkt mit dem Abfall der Gaskonzentration verbunden und führt zu einem abrupten Einbruch der Spannung:

$$\eta(I) = m \cdot e^{n \cdot I} \tag{6.15}$$

Die Gaskonzentration ist somit umgekehrt proportional zur Anzahl n der entstehenden Nebenprodukte (z.B. Stickstoff). Der Faktor m dient der Anpassung an die speziellen Eigenschaften der zu beschreibenden Brennstoffzelle.

Alterungsphänomene von Brennstoffzellen

Brennstoffzellen unterliegen, wie Verbrennungsmotoren und Batterien, einem Verschleiß- bzw. Alterungsprozess. Der Alterungszustand einer Brennstoffzelle kann

auf verschiedene Arten ermittelt werden. Im Labor kann etwa der Membran- oder der Katalysatorzustand exakt gemessen und dadurch der Verschleiß und dessen Ursachen bestimmt werden. Die Brennstoffzelle muss hierfür natürlich zerlegt werden, so dass dieses Verfahren für im Feld eingesetzte Brennstoffzellenfahrzeuge ausscheidet. Der Zustand einer Brennstoffzelle kann auch anhand der noch verfügbaren Spannung des Brennstoffzellenstacks bestimmt werden. Die Alterung wird bei dieser Methode durch die Spannungsdegradation des Stacks beschrieben.

Über die Lebenszeit reduziert sich die absolute Spannung einer Brennstoffzelle durch eine Vielzahl interner und externer Prozesse. Die Schädigungen der Brennstoffzellen können auf die Kontamination des Wasserstoffes oder der Luft aber auch auf Einschlüsse von schädlichen Stoffen bei der Produktion der Brennstoffzellenkomponenten zurückgeführt werden. Ebenso führen Veränderungen der internen mechanischen, chemischen und physikalischen Eigenschaften, wie Verdünnung der Membran, Löcher oder Risse, zur Degradation der Brennstoffzellenspannung. Diese Veränderungen hängen nicht zuletzt mit den Umgebungs- und Betriebsbedingungen der Brennstoffzelle zusammen. In stationären Einsatzszenarien, etwa als Notstromaggregat in Krankenhäusern, altern Brennstoffzellen deutlich langsamer als im mobilen Einsatz[WXP+04]. Dies könnte unter anderem auf den Einfluss interner Komponenten und deren Steuerung, wie Luftkompressor, Betriebstemperatur, hohe Lastdynamik oder die variierenden externen Einflüsse wie Fahrverhalten, Nutzungsbedingungen (Stadt- oder Überlandfahrten), Umgebungstemperatur, zurückzuführen sein.
Ein weiterer wichtiger Alterungsmechanismus hängt mit dem Aufbau von Brennstoffzellenstacks zusammen. Durch die Reihenschaltung der einzelnen Zellen führt schon der Defekt einer einzelnen Zelle zum Ausfall des gesamten Stacks.

Auch wenn noch alle Zellen funktionsfähig sind, ist ab einem gewissen Zeitpunkt die Spannung so gering, dass der Stack nicht mehr in der Lage ist, die Fahrzeugkomponenten, insbesondere den Elektromotor, mit ausreichend Spannung zu versorgen. Ebenso steigt der Wasserstoffverbrauch des Fahrzeugs mit abnehmender Spannung, da der Stack höhere Ströme liefern muss, um die gleiche Leistung zu erzeugen ($P = U \cdot I$). Um einen effizienten Fahrzeugbetrieb sicherzustellen, wird daher meist ein *EoL* (End of Life) Kriterium definiert, bei dem das Fahrzeug noch funktionsfähig ist, jedoch bereits eine bestimmte Spannungs- bzw. Leistungsdegradation aufweist, die keinen effizienten Betrieb mehr ermöglicht.

Mithilfe der Polarisationskurve kann die Spannungsdegradation und damit der aktuelle Zustand einer Brennstoffzelle bestimmt werden. Anhand der Form der Polarisationskurve lässt sich darüber hinaus ablesen, welche Bereiche der Brennstoffzelle Defekte aufweisen [Fow02]:

- Der Verlust an katalytischer Aktivität kann durch die Abnahme katalytischer Oberfläche oder katalytischen Materials sowie Membraninaktivierung oder Membranverschmutzung hervorgerufen werden. Der Verlust an katalytischer Aktivität zeigt sich in einer Parallelverschiebung der Polarisationskurve nach unten.

- Der Verlust an Protonen und elektrischer Leitfähigkeit kann durch Zersetzung der Membran, Austrocknung der Membran, Ablösung der Membran von Katalysator oder Korrosion der Leiterplatten verursacht werden. In der Polarisationskurve spiegelt sich der Verlust an Leitfähigkeit in einem steileren Abfall wieder.
- Eine Veränderung des Wasserabflussverhaltens und der damit verbundene Massentransferverlust führen zu einem früheren Abknicken der Polarisationskurve.

Die beschriebenen Effekte treten im Allgemeinen zusammen auf. Abbildung 6.3 zeigt die Veränderung der Polarisationskurve von einer neuen Zelle (Begin of Life, BoL) bis zum Ausfall der Zelle.

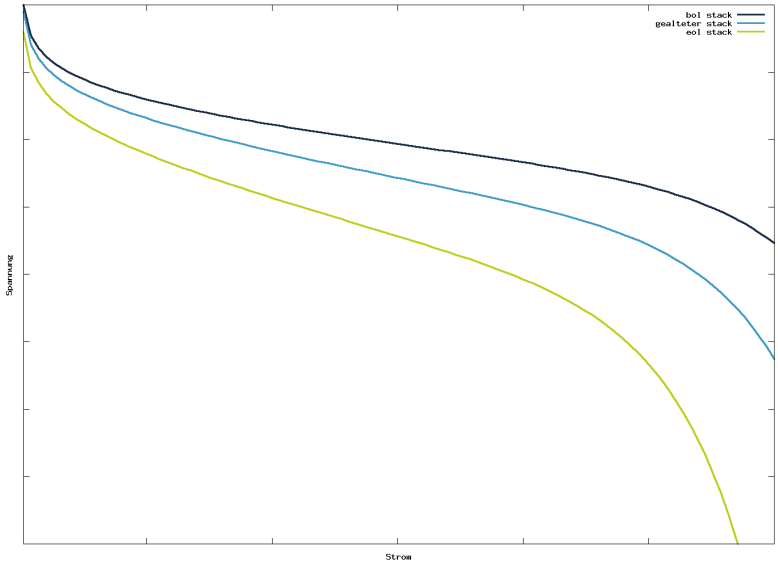


Abbildung 6.3: Veränderung der Polarisationskurve einer Brennstoffzelle.

Leider können Polarisationskurven, aufgrund unterschiedlicher Umgebungsbedingungen sowie der dynamischen Nutzung der Brennstoffzelle im Fahrzeugeinsatz, nicht direkt aus realen Fahrtdaten abgelesen werden, da das Strom/Spannungs-Diagramm einer solchen Fahrt im Allgemeinen keine Kurve ist, sondern, wie Abbildung 6.4 zeigt, vielmehr eine Punktwolke, in der zu einer bestimmten Stromstärke mehrere unterschiedliche Spannungen existieren. Die Variationsbreite der Spannung bei der gleichen Stromstärke ist für höhere Ströme deutlich größer als die beobachtbare Spannungsdegradation. Diese Spannungsschwankungen können im Wesentlichen auf die dynamische und statische Hysterese zurückgeführt werden [LCH$^+$01]. Der dynamische Hystereseffekt entsteht durch unterschiedliche Gasdrucke auf Anoden- und Kathodenseite bei der gleichen Stromentnahme. Dieser Effekt wird zum einen dadurch hervorgerufen, dass die Gasdrucksteuerung nicht schnell genug auf die Stromanforderung des Fahrers reagieren kann. Zum anderen führt die

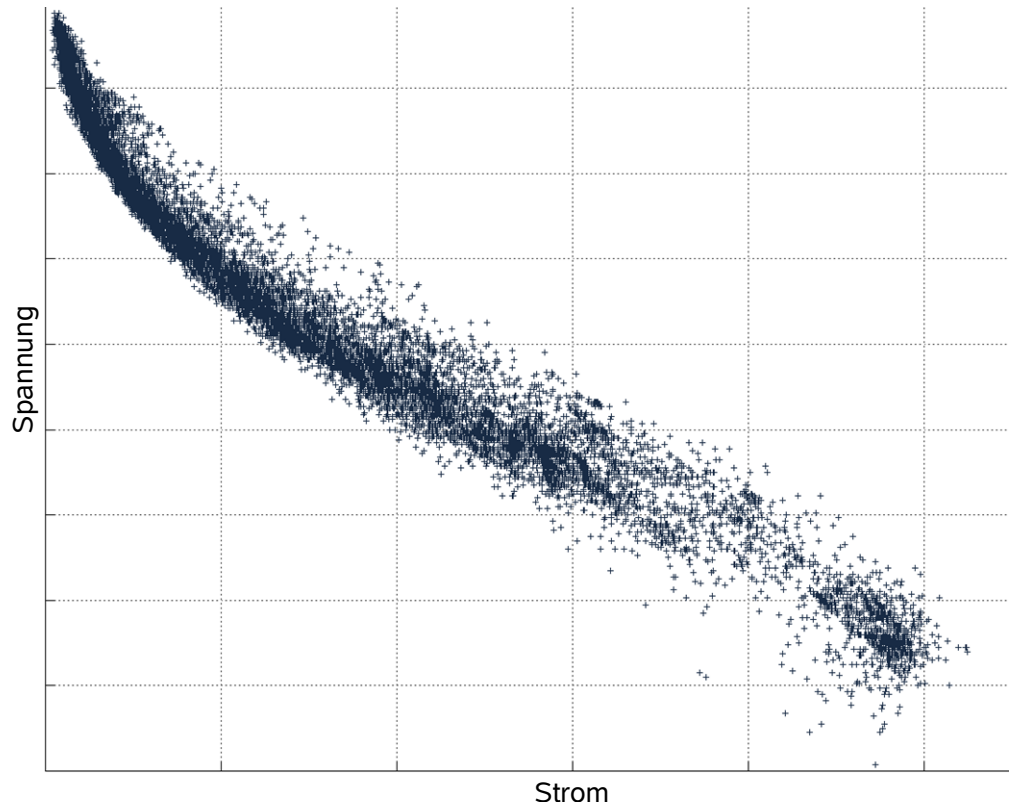


Abbildung 6.4: Strom/Spannungs-Diagramm einer Brennstoffzelle im Fahrzeugbetrieb.

Stromentnahme zum Ausfließen der Gase an der Anoden- und Kathodenseite, wobei sich ebenfalls der Gasdruck reduziert.
Der statische Hystereseffekt ist hauptsächlich auf unterschiedliche Betriebstemperaturen und der damit verbundenen Änderung der Membranfeuchtigkeit zurückzuführen. Dieser Effekt tritt beispielsweise nach einem Kaltstart auf, wenn sich die Brennstoffzelle bis zu ihrer spezifizierten Betriebstemperatur aufheizt.
Beide Hystereseffekte treten meist gemeinsam auf und können sich auch überlagern. Zur Ermittlung der Polarisationskurve müssen daher beide Effekte berücksichtigt werden, um ein geeignetes Modell erstellen zu können. Anhand dieses Modells kann die Polarisationskurve zu verschiedenen Zeitpunkten ermittelt und durch den Vergleich mehrerer Kurven die Degradation bestimmt werden. Darüber hinaus können, wie zuvor erläutert, auch mögliche Alterungsursachen aus der Polarisationskurve abgeleitet werden. Der Bestimmung der Polarisationskurve aus den Sensordaten kommt daher eine besondere Bedeutung zu.

6.3.2 Zustandsbestimmung im FDA-Miner

Im Rahmen dieser Arbeit wurden die in Abschnitt 6.1.1 beschriebenen Verfahren im FDA-Miner implementiert, um Polarisationskurven aus realen Fahrtdaten zu be-

stimmen. Ebenso wurde das in Abschnitt 6.1.2 vorgestellte Verfahren umgesetzt. Die Implementierungen dieser Verfahren bilden die Grundlage der im nächsten Abschnitt beschriebenen Ursachenanalyse.

Bestimmung der Polarisationkurve mit Optimierungsmethoden

Die Polarisationskurve einer Brennstoffzelle eignet sich, wie bereits in Abschnitt 6.3.1 erläutert, zur Bestimmung des Alterungszustands. Kann die Polarisationskurve durch einen kontrollierten Labortest ermittelt werden, lässt sich die Alterung einfach anhand der Verschiebung der Kurve ablesen. Ist dies, wie bei den Fahrzeugen der F-Cell Flotte, nicht möglich, muss die Polarisationskurve aus den Fahrtdaten abgeleitet werden.

Prinzipiell kann der Zusammenhang zwischen Strom und Spannung, nichts anderes ist die Polarisationskurve, mit unterschiedlichen Funktionen modelliert werden. Simple Approximationsverfahren, wie die lineare Regression oder die Polynominterpolation, führen jedoch nicht zu den gewünschten, detaillierten und interpretierbaren Ergebnissen. Die Grundlage des entwickelten Verfahrens bildet daher die in Abschnitt 6.3.1 vorgestellte Gleichung 6.12. Zur Zustandsbestimmung können die aus den realen Fahrtdaten ermittelten Parameter der Polarisationskurven zu unterschiedlichen Zeitpunkten miteinander verglichen und Veränderungen einfach abgelesen werden. Ein entscheidender Vorteil von theoretisch fundierten Gleichungen gegenüber anderen, heuristischen Modellen, liegt in der Interpretierbarkeit der errechneten Parameter. Diese lassen, aufgrund ihrer physikalischen Bedeutung, Rückschlüsse auf die Ursachen der Veränderungen zu.

Die Parameter dieser nicht-linearen Gleichung lassen sich mithilfe numerischer Optimierungsverfahren aus den Fahrtdaten berechnen. Da es sich bei den Parametern um physikalische Größen handelt, können diese nur Werte aus einem physikalisch sinnvollen Wertebereich annehmen. So müssen Temperaturen immer größer als 0 Kelvin, elektrische Widerstände größer 0 Ohm und die theoretische Maximalspannung eines Stacks im Bereich $[U_{min}; U_{max}]$ liegen. Bei der Optimierung müssen diese Nebenbedingungen/Restriktionen berücksichtigt werden, um korrekte Ergebnisse zu erhalten.

Die Anwendung des in Abschnitt 6.1.1 vorgestellten Optimierungsverfahrens zur Alterungszustandsbestimmung von Brennstoffzellenstacks aus den Fahrtdaten erfolgt in mehreren Schritten: Definition des Modells der Polarisationskurve, Datenaufbereitung, Berechnung der Parameter des Modells für jeden Datenblock und Bestimmung der Alterung anhand der Modellparameter.

Im ersten Schritt definiert der Benutzer ein beliebiges mathematisches Modell der Polarisationskurve des Brennstoffzellenstacks. Der FDA-Miner stellt für die Modelldefinition eine Web-Oberfläche bereit mit der das Modell aus beliebig vielen Parametern und Messgrößen der Fahrtdaten erstellt werden kann. Die Bestimmung der Parameter erfolgt durch das oben beschriebene Levenberg-Marquardt Verfahren. Aus den Angaben wird vom FDA-Miner automatisch ein entsprechender C-Quellcode generiert und ein ausführbares Programm erzeugt.

Im zweiten Schritt werden die Daten aufbereitet und die Parameter des Modells für alle vom Benutzer ausgewählten Fahrtdaten berechnet. Der FDA-Miner sendet die zur Datenaufbereitung benötigten Vorverarbeitungsprogramme und das Optimierungsprogramm zu den ausgewählten Datensätzen, die sich typischerweise auf unterschiedlichen Rechnern befinden. Die Vorverarbeitungsprogramme bereiten zunächst die Datensätze nach den Vorgaben des Modells auf. Hierbei werden unvollständige oder fehlerhafte Datensätze entfernt, die Daten anhand der Rahmenbedingungen gefiltert und in die vorgegebenen Datenblöcke aufgeteilt, wobei ein Datenblock einer Fahrt oder einer gewissen Betriebszeit entspricht. Sind in einem Datenblock zu wenige gültige Datensätze vorhanden, wird dieser von der weiteren Verarbeitung ausgeschlossen.
Anschließend werden die Parameter des Modells mithilfe des Optimierungsprogramms aus den vorverarbeiteten Datenblöcken bestimmt. Die Berechnung erfolgt für jeden Datenblock separat, so dass mehrere Datenblöcke gleichzeitig verarbeitet werden können. Für jeden Datenblock werden die ermittelten Parameter mit dem zugehörigen Zeitstempel (Betriebszeit, Lebenszeit oder gefahrene Kilometer) sowie eine grafische Darstellung, siehe Abbildung 6.5, der resultierenden Polarisationskurve gespeichert. Sind alle Berechnungen abgeschlossen, kombiniert der FDA-Miner die einzelnen Ergebnisse.

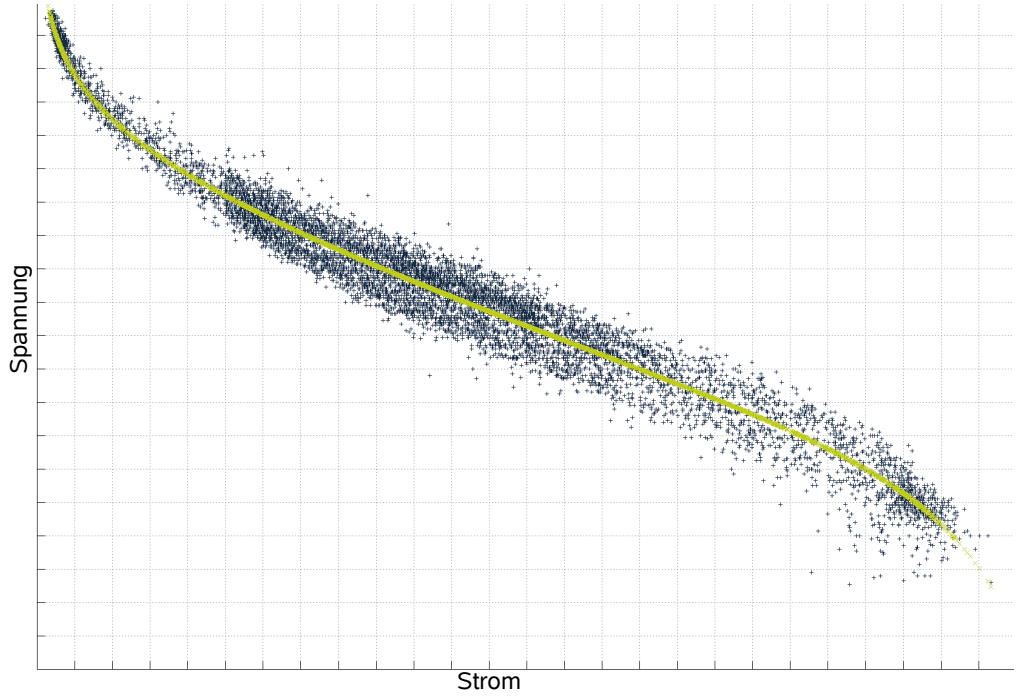


Abbildung 6.5: Aus den Daten berechnete Polarisationskurve eines Brennstoffzellenstacks.

Aus den berechneten Modellparametern kann im letzten Schritt der Zustand des Brennstoffzellenstacks bestimmt werden. Beim in dieser Arbeit entwickelten Ver-

fahren werden hierzu die berechneten Polarisationskurven aller Datenblöcke miteinander verglichen. Zur besseren Darstellung des Alterungsverlaufs kann anstelle der gesamten Polarisationskurve auch nur die Spannungsdifferenz bei einer Referenzstromstärke I_{ref} bestimmt werden.

Eine weitere Methode der Alterungszustandsbestimmung auf Basis der Polarisationskurvenmodelle betrachtet anstatt des gesamten Modells nur die Entwicklung einzelner Parameter über die Zeit. Beschreibt der Parameter eine zentrale Eigen-

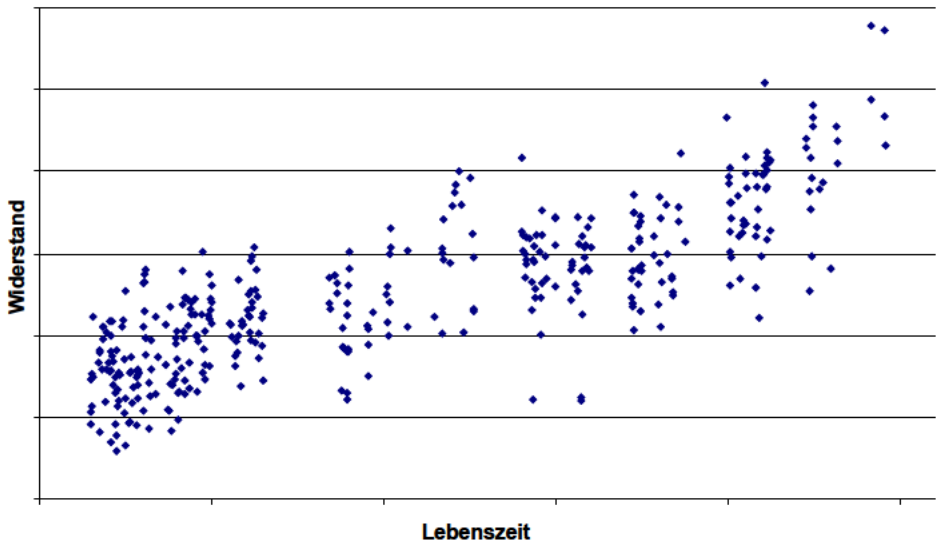


Abbildung 6.6: Der berechnete Innenwiderstand eines Fahrzeugs über dessen Betriebszeit.

schaft des Brennstoffzellenstacks, etwa den Innenwiderstand, zeigt dieses Verfahren den Zustand der zugehörigen Komponenten und lässt so Rückschlüsse auf mögliche Alterungsursachen zu. Abbildung 6.6 zeigt den Verlauf des Innenwiderstands eines Fahrzeugs über die Betriebszeit, wobei jeder Punkt dem ermittelten Innenwiderstandsparameter der Polarisationskurve einer Fahrt entspricht.
Im Rahmen dieser Arbeit wurden die zwei nachfolgenden Polarisationskurvenmodelle entworfen und auf die Fahrtdaten angewendet.

Das *Labormodell* orientiert sich an der in Abschnitt 6.3.1 beschriebenen mathematischen Beschreibung der Polarisationskurve einer Brennstoffzelle. Eine direkte Umsetzung der in Gleichung 6.12 beschriebenen Funktion ist aufgrund fehlender Sensorinformationen und der Anwendung auf den gesamten Stack anstelle einer Einzelzelle, nicht möglich. Das entworfene Modell M_l aus Gleichung 6.16 ersetzt die fehlenden Informationen durch die freien Parameter ϵ, μ, ω, θ und ϑ:

$$M_l = \epsilon - \mu \cdot ln(I) - \omega \cdot I - \theta \cdot e^{\vartheta \cdot I} \tag{6.16}$$

ϵ entspricht der aktuellen maximalen Spannung des Stacks, μ fasst die einzelnen Elemente des Aktivierungsverlusts zusammen, ω repräsentiert den Innenwiderstand

und θ sowie ϑ beschreiben den Massentransferverlust.
Die Wertebereiche der Parameter ergeben sich aus den von ihnen repräsentierten physikalischen Eigenschaften:

- $0 \leq \epsilon \leq U_{max}$. Die maximale Spannung des Stacks ϵ ist ungefähr gleich der Anzahl an Zellen im Stack (bei 1V pro Zelle) und kleiner als die theoretische Maximalspannung U_{max}.
- $\mu > 0$. Der Aktivierungsverlust $\mu \cdot ln(I)$ ist für $I \geq 1$ nur dann positiv, wenn μ positiv ist.
- $\omega > 0$. Der Innenwiderstand des Stacks muss positiv sein.
- $0 \leq \theta \leq \vartheta$. Die Restriktion $\theta \leq \vartheta$ gilt, da $\lim_{\vartheta \to 0} \theta \cdot e^{\vartheta \cdot I} \to \theta$ eine weitere lineare Komponente ergeben würde. In diesem Fall könnte θ den Parameter ω überlagern wodurch die Gleichung nicht mehr physikalisch interpretierbar wäre.

Mit diesen Restriktionen beschreibt das Modell die aufgezeichnete Spannung eines Datensatzes, nur auf Basis des Stroms, mit einem mittleren quadratischen Fehler von ungefähr 3%.

Neben der Bestimmung der Spannungsdegradation lassen die ermittelten Parameter auch Rückschlüsse auf mögliche Degradationsursachen zu. Beobachtet man etwa die Veränderungen des Innenwiderstandes ω über die Zeit, erhält man damit Hinweise auf den Zustand der Membran.

Das Labormodell M_l beschreibt den Brennstoffzellenstack unter statischen Laborbedingungen. Die in Abschnitt 6.3.1 beschriebenen Hystereseffekte, wie sie beim Betrieb im Fahrzeug auftreten, werden in diesem Modell nicht erfasst. Die statischen Hystereseffekte können weitgehend auf die Stacktemperatur und die dynamischen Hystereseffekte auf die Stromdynamik sowie den Gasdruck zurückgeführt werden. Das *Hysteresemodell* M_h erweitert daher M_l um die Temperaturabweichung T_d, die Gasdruckabweichung P_d und die Ableitung des Stroms I_d:

$$M_h = \epsilon + \tau \cdot T_d + \rho \cdot P_d - \delta \cdot I_d - \mu \cdot ln(I) - \omega \cdot I - \theta \cdot e^{\vartheta \cdot I} \tag{6.17}$$

Der Einfluss der statischen Hystereseffekte wird durch $\epsilon + \tau \cdot T_d + \rho \cdot P_d$ beschrieben. Die maximale Stackspannung ist damit keine feste Größe wie in Modell M_l, sondern ist abhängig von der aktuellen Temperatur und dem Gasdruck. Bei einem idealen Betriebszustand entspricht die Temperatur und der Gasdruck der definierten Normtemperatur T_{norm} und dem Normgasdruck P_{norm}. In diesem Zustand sind beide Abweichungen T_d ($T_d = T - T_{norm}$) und P_d ($P_d = P - P_{norm}$) Null. Weicht der Gasdruck oder die Temperatur von der Norm ab, verändert sich die Maximalspannung des Brennstoffzellenstacks entsprechend. Die Terme der Temperatur $\tau \cdot T_d$ und die Gasdruckabweichung $\rho \cdot P_d$ beschreiben den Effekt, dass mit Zunahme der Temperatur oder des Gasdrucks die Spannung des Stacks steigt.
Die im Fahrzeugbetrieb entstehende Dynamik, etwa die bei der Leistungsanforderung auftretende schnelle Steigerung der Stromentnahme, führt zu einer Reduzierung der Spannung. Der Einfluss der Stromdynamik wird in M_h durch $-\delta \cdot I_d$

($I_d = I_t - I_{t-1}$) beschrieben.
Die Grenzen der neu hinzugekommenen Parameter ergeben sich ebenfalls aus deren physikalischen Eigenschaften: $\tau > 0$, $\rho > 0$ und $\epsilon > 0$.
Das Modell beschreibt die aufgezeichnete Spannung eines Datensatzes, unter Beachtung der Restriktionen, mit einem mittleren quadratischen Fehler von ungefähr 2%.

Die zusätzlichen Parameter verbessern nicht nur die Genauigkeit des Modells, sondern liefern auch Hinweise auf den Einfluss der Hystereseffekte auf den Brennstoffzellenstack. Die aus der Analyse dieser Effekte gewonnenen Informationen können zur Verbesserung der Betriebsstrategie des Brennstoffzellenfahrzeugs genutzt werden.

Bestimmung der Polarisationkurve mit Künstlichen Neuronalen Netzen

Die im FDA-Miner realisierte Implementierung des in Abschnitt 6.1.1 vorgestellten KNN-Verfahrens erhält mehrere Eingabesignale aus den aufgezeichneten Daten und berechnet daraus die aktuelle Polarisationskurve des Stacks. Im Gegensatz zu den in [NSWP05] und [NSW04] beschriebenen Verfahren, die für die Zustandsbestimmung ein KNN auf den Daten der ersten Betriebsstunden trainieren und anschließend auf die später aufgezeichneten Daten anwenden, ermöglicht das Verfahren eine dezentrale Verarbeitung unter Berücksichtigung aller Fahrzeugdaten.

Der Ablauf der Berechnung im FDA-Miner entspricht hierbei weitgehend dem des Optimierungsverfahrens:
Im ersten Schritt werden der Aufbau sowie die Eingabesignale des KNN spezifiziert. Anschließend wird für die gewählten Eingabesignale der zur Simulation der Polarisationskurve verwendete Prüfstandslauf erzeugt. Dieser besteht aus den Werten der Eingabesignale eines realen, aufbereiteten Prüfstandslauf.
Beim Start der Berechnung, versucht der DOHS-Algorithmus eine Zuordnung zu erzeugen, die eine Ausführung der Vorverarbeitungsprogramme sowie der KNN Trainings- und Simulationsprogramme auf den Ressourcen die den ausgewählten Datensätzen vorhalten ermöglichen. Von den Vorverarbeitungsprogrammen werden unvollständige oder fehlerhafte Datensätze entfernt, die Daten anhand der für den Prüfstandslauf notwendigen Rahmenbedingungen gefiltert und in die vorgegebenen Datenblöcke aufgeteilt. Ein Datenblock entspricht bei diesem Verfahren einer Fahrt, einer gewissen Betriebs- oder Lebenszeit. Sind in einem Datenblock zu wenige gültige Datensätze vorhanden, wird dieser von der weiteren Verarbeitung ausgeschlossen. Für jeden vorverarbeiteten Datensatz wird eine KNN trainiert. Im darauf folgenden Schritt wird der vorgegebene Prüfstandslauf mit dem generierten KNN simuliert. Die resultierende Polarisationskurve wird mit dem zugehörigen Zeitstempel (Betriebszeit, Lebenszeit oder gefahrene Kilometer) sowie eine grafische Darstellung der resultierenden Polarisationskurve gespeichert. Sind alle Berechnungen abgeschlossen, kombiniert der FDA-Miner die einzelnen Ergebnisse und übergibt sie dem Benutzer zur weiteren Auswertung.

Das hier vorgestellte Verfahren ist deutlich aufwendiger als das in [NSWP05] und

[NSW04] beschriebenen Verfahren. Dafür nutzt es alle Daten zum Training, wobei alle vorhandenen Informationen zur Modellbildung berücksichtigt werden. Ebenso können durch die vom FDA-Miner zur Verfügung gestellten Ressourcen auch rechenintensive Trainingsverfahren, wie Kreuzvalidierung, genutzt werden, um die Qualität der Modelle zu erhöhen.

Alterungszustandsmodell im FDA-Miner

Alle gängigen Verfahren zur Alterungszustandsbestimmung von Brennstoffzellen folgen den oben beschriebenen Verfahren, bei dem zunächst ein möglichst genaues Modell des Stackbetriebs generiert und anschließend die Abweichung zum Startwert ermittelt wird. Das in Abschnitt 6.1.2 beschriebene Verfahren ermöglicht hingegen die direkte Modellierung des Alterungszustandes einer Brennstoffzelle durch ein KNN. Die Umsetzung dieses Verfahrens im FDA-Miner erfolgt in drei Schritten.

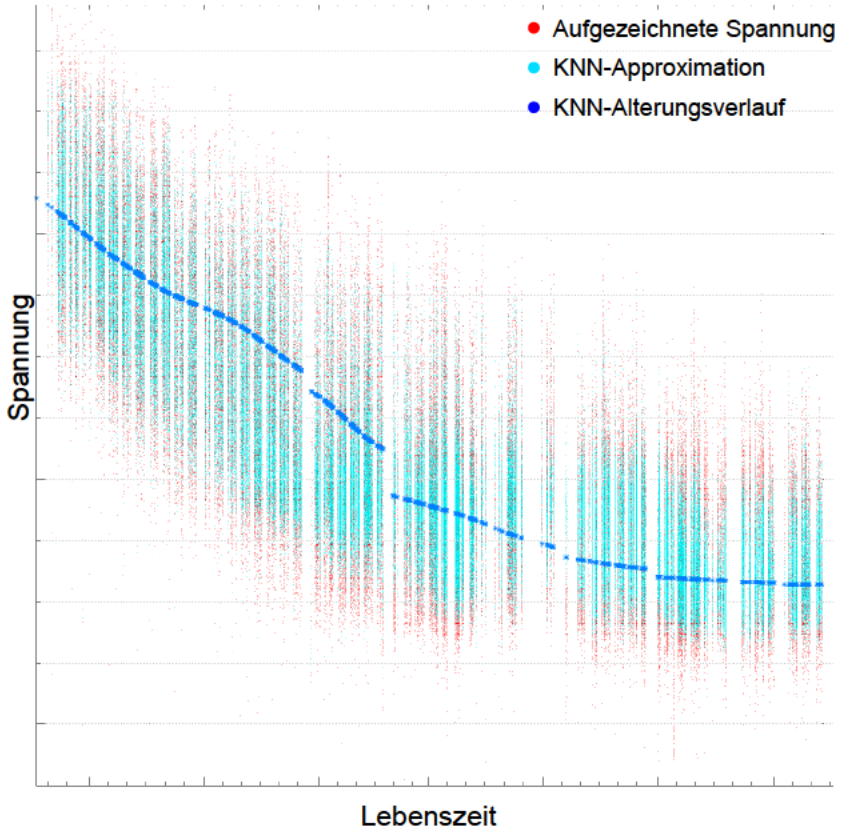


Abbildung 6.7: Mit Alterungszustandsmodell berechneter Alterungsverlauf eines Brennstoffzellenstacks.

Vor der Berechnung wird ein bestimmter Betriebszustand b der Sensorinformationen ($Strom = I_b$, $Temperatur = T_b$, $Gasdruck = P_b$, ...) sowie die zu verwendenden Zeitinformationen definiert.

Im ersten Schritt der Verarbeitung werden alle aufgezeichneten Fahrtdaten eines Fahrzeugs aufbereitet. Da der gewählte Betriebszustand sehr selten exakt auftritt, werden die ausgewählten Sensorinformationen für das Training um den definierten Betriebszustand herum extrahiert ($I = I_b \pm \Delta I,\ T = T_b \pm \Delta T,\ P = P_b \pm \Delta P,\ ...$). In einem Kombinationsschritt werden die Resultate der einzeln verarbeiteten Fahrtdaten zu einem Datensatz pro Fahrzeug zusammengefasst.
Im zweiten Schritt werden die Datensätze mit den KNN-Programmen auf mehrere Rechner verteilt. Anschließend wird pro Datensatz - ein Datensatz pro Fahrzeug - ein KNN trainiert. Das Training erfolgt durch Kreuzvalidierung, so dass unterschiedliche Daten eines Datensatzes abwechselnd zum Training und zur Überprüfung verwendet werden.
Nach Beendigung des Trainings erfolgt die Alterungszustandsbestimmung auf Basis des ausgewählten Betriebszustands b und Zeitinformationen t. Das Ergebnis dieser Berechnung ist eine Zeitreihe der Spannung beim Betriebszustand b. Aus dieser Zeitreihe kann, wie in Abbildung 6.7 dargestellt, die Spannungsdegradation zwischen zwei Zeitpunkten direkt abgelesen werden. Die Zeitreihe sowie diese grafische Darstellung des Spannungsverlaufs wird den Benutzern in einer Reportoberfläche zur Verfügung gestellt.

6.3.3 Ursachenanalyse für Brennstoffzellenfahrzeuge

Die bisherigen Arbeiten zur Untersuchung des Alterungsverhaltens von Brennstoffzellenstacks in Fahrzeugen [NSWP05][NSW04][Fri09][FSN+08] beschränken sich auf Modelle zur Zustandsbestimmung. Auf die zentrale Frage nach den Ursachen der Alterung bieten diese Verfahren jedoch keine Antworten. Die in dieser Arbeit entwickelte Ursachenanalyse stellt erstmals eine Methode zur Berechnung möglicher Alterungsursachen aus den aufgezeichneten Fahrdaten zur Verfügung [RKH07]. Das in Abschnitt 6.2 vorgestellte Verfahren wird im FDA-Miner zur Alterungsursachenanalyse der Fahrzeugbrennstoffzellenstacks der F-Cell Flotte angewandt. Die Analyse wird regelmäßig durchgeführt und die Ergebnisse in einem Report zusammengefasst.

Die Degradation wird in der aktuellen Implementierung, wie bei den Verfahren zur Zustandsbestimmung, allein durch den Spannungsabfall bei hohen Lasten definiert. Die Berechnung der möglichen degradationsverursachenden Zustandsvariablen erfolgt in mehreren Schritten.

Im ersten Schritt werden die aufgezeichneten Fahrtdaten bereinigt und die Summe aller Messwerte $\sum_{j=t_{0i}}^{t_{1i}} h_j(x_j(t))$ für die vorgegebenen Zeitintervalle $[t_{0i}, t_{1i}]$ bestimmt. Die Zeitintervalle sind hierbei variabel und liegen üblicherweise im Bereich einiger Betriebsstunden. Die Bereinigung und Summierung der Messdaten erfolgt für jede Fahrt separat, so dass die Berechnung auf mehreren Rechnern parallel ausgeführt werden kann. Die Teilergebnisse werden in einem Kombinationsschritt für jedes Zeitintervall zusammengefasst.

Im zweiten Schritt werden der Degradationszustand $w_i = (w(t_{0i}) + w(t_{1i}))/2$ und die Degradationsrate $\Delta w_i = w(t_{1i}) - w(t_{0i})$ für die einzelnen Zeitintervalle bestimmt. Wie in Abschnitt 6.3.1 ausgeführt, kann die Spannungsdegradation zu einem gewis-

sen Zeitpunkt $w(t)$ nicht einfach aus den Messungen abgelesen werden. Daher wird die Spannungsdegradation mit den im vorherigen Abschnitt beschriebenen Verfahren zur Alterungszustandsbestimmung berechnet.

Die Parameter a_j, b und c des Approximationsproblems 6.10 werden im dritten Schritt mit dem vorgestellten Fixpunktalgorithmus bestimmt. Anschließend wird der Lineare Korrelationskoeffizient sowie der Rangkorrelationskoeffizient für alle Messgrößen berechnet. Die so entstandenen Korrelationsmatrizen dienen als Grundlage für weitergehende Analysen der Alterungsursachen der Brennstoffzellenfahrzeuge.

Kapitel 7

Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurde mit dem DOHS-Algorithmus ein Schedulingverfahren entwickelt, um den aktuellen Herausforderungen der daten- und rechenintensiven Anwendung in organisationsübergreifenden Umgebungen zu begegnen. Bei der Entwicklung wurde insbesondere auf die Anwendbarkeit des Algorithmus unter realen Bedingungen geachtet. Die Grundlage hierfür bilden das entworfene Umgebungsmodell und die Zielfunktion φ. Der Verarbeitungsaufwand als Zielfunktion berücksichtigt sowohl die Interessen der Betreiber als auch der Benutzer und spiegelt damit die aktuellen Anforderungen an das Scheduling in organisationsübergreifenden Umgebungen wieder. Das Umgebungsmodell wiederum ermöglicht die Modellierung des Scheduling in den unterschiedlichsten Umgebungen, wie sie in aktuellen Grid- oder MapReduce-Umgebungen auftreten. Der DOHS-Algorithmus ist in der Lage in diesen unterschiedlichen Umgebungsstrukturen die Aufträge so auf die Ressourcen zu verteilen, dass der Verarbeitungsaufwand über alle Aufträge sehr gering bleibt. Dies erreicht der Algorithmus einerseits durch die ausschließliche Verwendung von Ressourceninformationen, die auch in realen Umgebungen zur Verfügung stehen, sowie durch die Integration der Umgebungsstruktur in das Scheduling. Im Vergleich zu den bestehenden Schedulern, die nur schwer bestimmbare Informationen wie die Netzwerkauslastung und die Berechnungsdauer benötigen, zeigen die durchgeführten Simulationen, dass der DOHS diesen unter realen Bedingungen deutlich überlegen ist.

Der DOHS-Algorithmus bildet die Grundlage des FDA-Miners und ermöglicht damit erstmals die zeitnahe Analyse der gesamten Daten einer Fahrzeugflotte. Die zentrale Herausforderung bei der Entwicklung des FDA-Miners war die effektive und effiziente Verarbeitung großer, global verteilter Datenmengen sowie die Ausführung rechenintensiver Anwendung zu kombinieren. Die in dieser Arbeit völlig neu entworfenen Daten- und Ausführungsdienste ermöglichen die Ausführung der Analyseprogramme auf beliebigen Ressourcen, sodass die Daten vor der Verarbeitung nicht aufwendig transferiert werden müssen. Aufgrund der neuen Verarbeitungsmöglichkeiten und der Leistungsfähigkeit des neuen FDA-Miners wird dieser aktuell für die erste und zweite Generation der FCell-Flotte Daimler AG eingesetzt. Neben den Simulationen zeigen sich die Vorteile des DOHS in den mehreren Tausend Ver-

arbeitungsaufträgen mit Millionen Datensätzen die bis jetzt im produktiven Betrieb verarbeitet wurden.

Die neu gewonnene Rechenkapazität ermöglicht den Einsatz komplexer Methoden zur Analyse des Alterungsverhaltens von Brennstoffzellensystemen in Fahrzeugen. Mit den im Rahmen dieser Arbeit angepassten und neu entwickelten Alterungszustandsbestimmungsverfahren lässt sich die Degradation eines Brennstoffzellenstacks direkt aus den Fahrtdaten berechnen. Aufwendige Labor- und Prüfstandstests können so vermieden werden. Im Gegensatz zu den bisher in der Literatur bekannten und bei der Daimler AG eingesetzten Verfahren nutzen die in dieser Arbeit vorgestellten Verfahren die gesamte Datenbasis und erreichen dadurch eine höhere Genauigkeit. Die flexible und schnelle Berechnung der Modelle im FDA-Miner bietet den Ingenieuren zusätzlich die Möglichkeit, unterschiedliche Konfigurationen zu untersuchen.
Darüber hinaus wurde in dieser Arbeit erstmals eine Methode zur Bestimmung der dem Alterungsprozess zugrunde liegenden Mechanismen und Einflussfaktoren aus Felddaten vorgestellt. Die mithilfe dieser Methode gewonnen Informationen können für die Weiterentwicklung und Optimierung der Brennstoffzellensysteme im Fahrzeugeinsatz verwendet werden, sodass diese Antriebstechnologie weiter zur Marktreife entwickelt werden kann. Die entwickelten Alterungszustands- und Alterungsursachenanalyseverfahren können jedoch auch für beliebige andere Anwendungsgebiete eingesetzt werden. So werden die KNN bereits zur Alterungszustandsbestimmung für andere Komponenten des Brennstoffzellensystems, wie etwa für den Luftkompressor, eingesetzt.

Der entwickelte DOHS-Algorithmus wurde speziell für daten- *und* rechenintensive Anwendungen entwickelt. In der aktuellen Form ist der Algorithmus allerdings nicht für das Scheduling von Anwendungen geeignet, die mehr als eine Rechenressource benötigen. In einigen äußerst rechenintensiven Anwendungsgebieten, etwa für komplexe Simulationen, können parallele Anwendungen enorm von der Ausführung auf mehreren Rechenressourcen profitieren. Damit wäre das Scheduling von parallelen Anwendungen eine lohnenswerte Weiterentwicklung des DOHS-Algorithmus. Neben Weiterentwicklungen für sehr rechenintensive Anwendungsgebiete wäre für zusammengesetzte, datenintensive Anwendungen eine Berücksichtigung der Ausgabedaten im Scheduling des DOHS von Vorteil. Dies gilt insbesondere in Anwendungsgebieten, in denen mehrere datenintensive Anwendungen nacheinander ausgeführt werden müssen, wobei die Ausgabedaten der einen Anwendung die Eingabedaten der nachfolgenden Anwendung bilden.

Literaturverzeichnis

[AAO05] Aljanaby, Alaa ; Abuelrub, Emad ; Odeh, Mohammed: A Survey of Distributed Query Optimization. In: *Int. Arab J. Inf. Technol.* 2 (2005), Nr. 1, S. 48–57

[ABD+01] Allen, Gabrielle ; Benger, Werner ; Dramlitsch, Thomas ; Goodale, Tom ; Hege, Hans-Christian ; Lanfermann, Gerd ; Merzky, André ; Radke, Thomas ; Seidel, Edward: Cactus Grid Computing: Review of Current Development. In: Sakellariou, Rizos (Hrsg.) ; Gurd, John (Hrsg.) ; Freeman, Len (Hrsg.) ; Keane, John (Hrsg.): *Euro-Par 2001 Parallel Processing* Bd. 2150. Springer Berlin / Heidelberg, 2001, S. 817–824

[AD06] Akl, Selim G. ; Dong, Fangpeng: Scheduling Algorithms for Grid Computing: State of the Art and Open Problems / School of Computing — Queen's University. Version: January 2006. `http://techreports.cs.queensu.ca/papers/view/2006-504`. Kingston, Ontario, Canada, January 2006 (2006-504). – Forschungsbericht. – Online–Ressource. – 55 S

[AFG+09] Armbrust, Michael ; Fox, Armando ; Griffith, Rean ; Joseph, Anthony D. ; Katz, Randy H. ; Konwinski, Andrew ; Lee, Gunho ; Patterson, David A. ; Rabkin, Ariel ; Stoica, Ion ; Zaharia, Matei: Above the Clouds: A Berkeley View of Cloud Computing / EECS Department, University of California, Berkeley. 2009. – Forschungsbericht

[AT05] Ali, Ali S. ; Taylor, Ian J.: Web Services Composition for Distributed Data Mining. In: *ICPPW '05: Proceedings of the 2005 International Conference on Parallel Processing Workshops*. Washington, DC, USA : IEEE Computer Society, 2005. – ISBN 0–7695–2381–1, S. 11–18

[BAG00] Buyya, Rajkumar ; Abramson, David ; Giddy, Jonathan: Nimrod/G: An architecture for a resource management and scheduling system in a global computational Grid. In: *Proceedings of the 4th International Conference on High Performance Computing in Asia-Pacific Region*, 2000, S. 283–289

[BC05] Bovet, Daniel ; Cesati, Marco: *Understanding The Linux Kernel*. Oreilly & Associates Inc, 2005. – ISBN 0596005652

[BGR03] Bharadwaj, Veeravalli ; Ghose, Debasish ; Robertazzi, Thomas G.: Divisible Load Theory: A New Paradigm for Load Scheduling in Distributed Systems. In: *Cluster Computing* 6 (2003), Nr. 1, S. 7–17

[Bor07] Borthakur, Dhruba: The Hadoop Distributed File System: Architecture and Design / The Apache Software Foundation. 2007. – Forschungsbericht

[Bru09] Brucker, Peter: *Scheduling Algorithms*. 5rd. Springer, 2009

[CC03] Cannataro, Mario ; Comito, Carmela: A data mining ontology for grid programming. In: *Proc. 1st Int. Workshop on Semantics in Peer-to-Peer and Grid Computing, in conjunction with WWW2003*, 2003, S. 113–134

[CCF04] Cai, Min ; Chervenak, Ann ; Frank, Martin: A Peer-to-Peer Replica Location Service Based on a Distributed Hash Table. In: *SC '04: Proceedings of the 2004 ACM/IEEE conference on Supercomputing*. Washington, DC, USA : IEEE Computer Society, 2004. – ISBN 0–7695–2153–3, S. 56

[CCGK02] Chakrabarti, Amit ; Chekuri, Chandra ; Gupta, Anupam ; Kumar, Amit: Approximation Algorithms for the Unsplittable Flow Problem. In: *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization*, Springer-Verlag, 2002, S. 51–66

[CDL+07] Chervenak, A. ; Deelman, E. ; Livny, M. ; Su, Mei-Hui ; Schuler, R. ; Bharathi, S. ; Mehta, G. ; Vahi, K.: Data placement for scientific applications in distributed environments. In: *Grid Computing, 2007 8th IEEE/ACM International Conference on*, 2007, S. 267–274

[CI86] Chung, Chin-Wan ; Irani, Keki B.: An Optimization of Queries in Distributed Database Systems. In: *Journal Parallel Distributed Comput.* 3 (1986), Nr. 2, S. 137–157

[CM10] C. Marinescu, Chen Y.: Algorithms for Divisible Load Scheduling of Data-intensive Applications. In: *Journal of Grid Computing* 8 (2010), Nr. 1, S. 133–155

[CPB+04] Chervenak, Ann L. ; Palavalli, Naveen ; Bharathi, Shishir ; Kesselman, Carl ; Schwartzkopf, Robert: Performance and Scalability of a Replica Location Service. In: *HPDC '04: Proceedings of*

the 13th IEEE International Symposium on High Performance Distributed Computing. Washington, DC, USA : IEEE Computer Society, 2004. – ISBN 0–7803–2175–4, S. 182–191

[CTT03] Congiusta, A. ; Talia, D. ; Trun?o, P.: VEGA: A visual environment for developing complex grid applications. In: *Proc. of 1st Int. Workshop on Knowledge Grid and Grid Intelligence*, 2003, S. 56–66

[CTT07] Congiusta, Antonio ; Talia, Domenico ; Trunfio, Paolo: Distributed data mining services leveraging WSRF. In: *Future Generation Computer Systems* 23 (2007), Nr. 1, S. 34–41

[Cyb88] Cybenko, G.: Continuous valued neural networks with two hidden layers are sufficient / Department of Computer Science, Tufts University, Medford. 1988. – Forschungsbericht

[Dad96] Dadam, Peter: *Verteilte Datenbanken und Client/Server-Systeme - Grundlagen, Konzepte und Realisierungsformen*. Springer, 1996

[DG04] Dean, Jeffrey ; Ghemawat, Sanjay: MapReduce: Simplified Data Processing on Large Clusters, 137–150

[DPHHL10] Delgado Peris, A. ; Hernandez, J. ; Huedo, E. ; Llorente, I.M.: Data location-aware job scheduling in the grid. Application to the GridWay metascheduler. In: *17th International Conference on Computing in High Energy and Nuclear Physics (CHEP09)* Bd. 219, 2010

[FFM07] Feller, Martin ; Foster, Ian ; Martin, Stuart: *GT4 GRAM: A functionality and performance study*. 2007

[FFPW99] Friedlmeier, G. ; Friedrich, J. ; Panik, F. ; Weiss, W.: First experiences with fuel cell demonstration vehicles. In: *GPC 1999 Proceedings: Advanced Propulsion Systems* 10 (1999), S. 84–94

[Fit01] Fitzgerald, Steven: Grid Information Services for Distributed Resource Sharing. In: *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing*, IEEE Computer Society, 2001 (HPDC '01)

[FK99] In: Foster, Ian ; Kesselman, Carl: *The Grid: Blueprint for a New Computing Infrastructure*. Morgan-Kaufman

[FKT01] Foster, Ian ; Kesselman, Carl ; Tuecke, Steven: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. In: *International J. Supercomputer Applications* (2001). `http://www.globus.org/alliance/publications/papers/anatomy.pdf`

[FKTT98] Foster, Ian T. ; Kesselman, Carl ; Tsudik, Gene ; Tuecke, Steven: A Security Architecture for Computational Grids. In: *ACM Conference on Computer and Communications Security*, 83-92

[FM94] Frost, V.S. ; Melamed, B.: Traffic modeling for telecommunications networks. In: *IEEE Communications Magazine* 32 (1994), Nr. 3, S. 70–81

[Fos02] Foster, Ian: What is the Grid? A Three Point Checklist. In: *GRIDToday* (2002). `http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf`

[Fos05] Foster, Ian: *A Globus Primer Or, Everything You Wanted to Know about Globus, but Were Afraid To Ask.* : , 2005. `http://www.globus.org/toolkit/docs/4.0/key/`

[Fow02] Fowler, M.: Issues associated with Voltage Degradation in a PEMFC. In: *Journal of New Materials for Electrochemical Systems* 5 (2002)

[Fri09] Frisch, Thomas: *Eine Methode zur Bestimmung des Verschleiß- und Alterungszustands einer PEM-Brennstoffzelle auf Basis eines modellbasierten Onboard-Diagnosesystems*, Ruhr Universität Bochum, Diss., 2009

[FSN+08] Friedrich, J. ; Schamm, R. ; Nitsche, C. ; Keller, J. ; Rehfus, B. ; Frisch, T. ; Röhm, M.: Advanced On-/Offboard Diagnostics for a Fuel Cell Vehicle Fleet. In: *Society of Automotive Engineers SAE World Congress 2008*, 2008

[FTL+02] Frey, James ; Tannenbaum, Todd ; Livny, Miron ; Foster, Ian ; Tuecke, Steven: Condor-G: A Computation Management Agent for Multi-Institutional Grids. In: *Cluster Computing* 5 (2002), July, Nr. 3, S. 237–246

[FZRL08] Foster, I. ; Zhao, Yong ; Raicu, I. ; Lu, Shiyong: Cloud Computing and Grid Computing 360-Degree Compared. In: *Grid Computing Environments Workshop, 2008. GCE '08*, 2008, S. 1–10

[Gra66] Graham, R. L.: Bounds for certain multiprocessing anomalies. In: *Bell System Technical Journal* 45 (1966), S. 1563–1581

[Gra69] Graham, R. L.: Bounds on Multiprocessing Timing Anomalies. In: *SIAM JOURNAL ON APPLIED MATHEMATICS* 17 (1969), Nr. 2, S. 416–429

[Gra87] Gray, J.: *Transparency in Its Place.* 1987

[GVBB13] Garg, Saurabh K. ; Venugopal, Srikumar ; Broberg, James ; Buyya, Rajkumar: Double auction-inspired meta-scheduling of parallel applications on global grids. In: *Journal of Parallel and Distributed Computing* 73 (2013), Nr. 4, S. 450 – 464

[HML04] Huedo, Eduardo ; Montero, Ruben S. ; Llorente, Ignacio M.: A framework for adaptive execution in grids. In: *Softw. Pract. Exper.* 34 (2004), Nr. 7, S. 631–651. – ISSN 0038–0644

[HY79] Hevner, A.R. ; Yao, S.B.: Query Processing in Distributed Database System. In: *Software Engineering, IEEE Transactions on* 5 (1979), Nr. 3, S. 177–187

[IK84] Ibaraki, Toshihide ; Kameda, Tiko: On the optimal nesting order for computing N-relational joins. In: *ACM Trans. Database Syst.* 9 (1984), Nr. 3, S. 482–502. – ISSN 0362–5915

[JUMS83] Jantz, Diane ; Unger, E. A. ; McBride, R. ; Slonim, Jacob: Query processing in a distributed data base. In: *Proceedings of the 1983 ACM SIGSMALL symposium on Personal and small computers*, ACM, 1983 (SIGSMALL '83), S. 237–244

[KDD+03] Kelly, Gabrielle A. ; Davis, Kelly ; Dolkas, Konstantinos N. ; Doulamis, Nikolaos D. ; Goodale, Tom ; Kielmann, Thilo ; Merzky, André ; Nabrzyski, Jarek ; Pukacki, Juliusz ; Radke, Thomas ; Russell, Michael ; Shalf, John ; Taylor, Ian: Enabling Applications on the Grid – A GridLab Overview. In: *International Journal of High Performance Computing Applications* 17 (2003), S. 449–466

[Kos00] Kossmann, Donald: The state of the art in distributed query processing. In: *ACM Comput. Surv.* 32 (2000), Nr. 4, S. 422–469

[Kos12] Kosar, Tevfik: *Data Intensive Distributed Computing: Challenges and Solutions for Large-scale Information Management.* 1. IGI Global, 2012

[KYF02] Kanzow, Christian ; Yamashita, Nobuo ; Fukushima, Masao: *Levenberg-Marquardt Methods for Constrained Nonlinear Equations with Strong Local Convergence Properties.* 2002

[LCH+01] Laurencelle, F. ; Chahine, R. ; Hamelin, J. ; Agbossou, K. ; Fournier, M. ; Bose, T. K. ; Laperrière, A.: Characterization of a Ballard MK5-E Proton Exchange Membrane Fuel Cell Stack. In: *Fuel Cells* 1 (2001), S. 67

[Lev44] Levenberg, K.: A Method for the Solution of Certain Problems in Least Squares. In: *Quart. Appl. Math. 2*, 1944, S. 164–168

[LLA98] Lee, J.H. ; Lalk, T.R. ; Appleby, A.J.: Modeling electrochemical performance in large scale proton exchange membrane fuel cell stacks. In: *Journal of Power Sources* 70 (1998), Nr. 2, S. 258 – 268

[Läm08] Lämmel, Ralf: Google's MapReduce programming model - Revisited. In: *Science of Computer Programming* 70 (2008), Nr. 1, S. 1 – 30

[LRKB77] In: Lenstra, J. K. ; Rinnooy Kan, A. H. G. ; Brucker, P.: *Annals of Discrete Mathematics*. Bd. 1: *Complexity of Machine Scheduling Problems*. Elsevier, 1977, S. 343–362

[LY80] Lam, K. ; Yu, C. T.: An approximation algorithm for a file-allocation problem in a hierarchical distributed system. In: *Proceedings of the 1980 ACM SIGMOD international conference on Management of data*, ACM, 1980 (SIGMOD '80), S. 125–132

[LZ02] Li, Guodong ; Zhang, Defu: Distributing and scheduling divisible task on parallel communicating processors. In: *J. Comput. Sci. Technol.* 17 (2002), November, Nr. 6, S. 788–796. – ISSN 1000–9000

[Mah99] Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems. In: *Journal of Parallel and Distributed Computing* 59 (1999), Nr. 2, S. 107 – 131

[Mar63] Marquardt, D.: An Algorithm for Least-Squares Estimation of Nonlinear Parameters. In: *SIAM J. Appl. Math. 11*, 1963, S. 431–441

[MAS+07] McClatchey, Richard ; Anjum, Ashiq ; Stockinger, Heinz ; Ali, Arshad ; Willers, Ian ; Thomas, Michael: Data Intensive and Network Aware (DIANA) Grid Scheduling. In: *Journal of Grid Computing* 5 (2007), S. 43–64. – ISSN 1570–7873

[ME04] Mohamed, H. H. ; Epema, D. H. J.: An Evaluation of the Close-to-Files Processor and Data Co-Allocation Policy in Multiclusters. In: *In Proceedings of the 2004 IEEE International Conference on Cluster Computing*, IEEE Society Press, 2004, S. 287–298

[Mit97] Mitchell, T.: *Machine Learning (Mcgraw-Hill International Edit)*. 1st. McGraw-Hill Education (ISE Editions), 1997. – ISBN 0071154671

[NSW04] Nitsche, C. ; Schroedl, S. ; Weiss, W.: Onboard Diagnostics Concept for Fuel Cell Vehicles using Adaptive Modelling. In: *IEEE Intelligent Vehicles Symposium*, 2004

[NSW+05] Nitsche, C. ; Schroedl, S. ; Weiss, W. ; Pucher, E. ; Fosmoe, R.: Predictive Maintenance and Diagnostics Concept for a Worldwide Fuel Cell Vehicle Fleet. In: *Electric Vehicle Symposium*, 2005

[NSWP05] Nitsche, C. ; Schroedl, S. ; Weiss, W. ; Pucher, E.: Rapid (practical) Methodology for Creation of Fuel Cell Systems Models with Scalable Complexity. In: *Journal of Power Sources* 145 (2005), Nr. 2, S. 383 – 391

[OV11] Özsu, M. T. ; Valduriez, Patrick: *Principles of distributed database systems*. Bd. 3. Springer, 2011

[PF95] Paxson, V. ; Floyd, Sally: Wide area traffic: the failure of Poisson modeling. In: *IEEE/ACM Transactions on Networking* 3 (1995), Nr. 3, S. 226–244

[Pin12] Pinedo, Michael L.: *Scheduling: Theory, Algorithms, and Systems*. 4rd. Springer, 2012

[PK03] Park, Sang-Min ; Kim, Jai-Hoon: Chameleon: A Resource Scheduler in A Data Grid Environment. In: *Proceedings of the 3st International Symposium on Cluster Computing and the Grid*. Washington, DC, USA : IEEE Computer Society, 2003 (CCGRID '03), S. 258–266

[PRS05] Phan, Thomas ; Ranganathan, Kavitha ; Sion, Radu: Evolving Toward the Perfect Schedule: Co-scheduling Job Assignments and Data Replication in Wide-Area Systems Using a Genetic Algorithm. In: *Job Scheduling Strategies for Parallel Processing* Bd. 3834, 2005, S. 173–193

[QPTGG+12] Quezada-Pina, Ariel ; Tchernykh, Andrei ; González-García, José Luis ; Hirales-Carbajal, Adán ; Ramírez-Alcaraz, Juan M. ; Schwiegelsohn, Uwe ; Yahyapour, Ramin ; Miranda-López, Vanessa: Adaptive parallel job scheduling with resource admissible allocation on two-level hierarchical grids. In: *Future Gener. Comput. Syst.* 28 (2012), Nr. 7, S. 965–976

[RBJS03] *Kapitel* The Evolution of the Grid. In: Roure, David D. ; Baker, Mark A. ; Jennings, Nicholas R. ; Shadbolt, Nigel R.: *Grid Computing: Making The Global Infrastructure a Reality*. John Wiley & Sons

[RF02a] Ranganathan, Kavitha ; Foster, Ian: Decoupling Computation and Data Scheduling in Distributed Data-Intensive Applications. In: *HPDC '02: Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing*, IEEE Computer Society, 2002, S. 352–358

[RF02b] Ripeanu, Matei ; Foster, Ian: A Decentralized, Adaptive Replica Location Mechanism. In: *HPDC '02: Proceedings of the 11th IEEE*

International Symposium on High Performance Distributed Computing. Washington, DC, USA : IEEE Computer Society, 2002. – ISBN 0–7695–1686–6, S. 24

[RGS10] Röhm, Matthias ; Grabert, Matthias ; Schweiggert, Franz: A Generalized MapReduce Approach for Efficient mining of Large data Sets in the GRID. In: *1. International Conference on Cloud Computing, GRIDs, and Virtualization*, 2010

[RGS11] Röhm, Matthias ; Grabert, Matthias ; Schweiggert, Franz: An Integrated Approach for Data- and Compute-intensive Mining of Large Data Sets in the GRID. In: *International Journal On Advances in Intelligent Systems* 4 (2011), Nr. 3

[RGS12] Röhm, Matthias ; Grabert, Matthias ; Schweiggert, Franz: Scheduling Data- and Compute-intensive Applications in Hierarchical Distributed Systems. In: *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA*, 2012

[RKH07] Röhm, Matthias ; Keller, Jörg ; Hrycej, Tomas: Data Mining Fuel Cell Fleet Data for Stack Degradation Analysis. In: *Fuel Cell Seminar, San Antonio*, 2007

[RLS98] Raman, Rajesh ; Livny, Miron ; Solomon, Marvin: Matchmaking: Distributed Resource Management for High Throughput Computing. In: *Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing (HPDC7)*. Chicago, IL, July 1998

[RLS00] Raman, Rajesh ; Livny, Miron ; Solomon, Marvin: Resource Management through Multilateral Matchmaking. In: *Proceedings of the Ninth IEEE Symposium on High Performance Distributed Computing (HPDC9)*. Pittsburgh, PA, August 2000, S. 290–291

[RM97] Rho, Sangkyu ; March, SalvatoreT.: Optimizing distributed join queries: A genetic algorithm approach. In: *Annals of Operations Research* 71 (1997), S. 199–228

[Ros61] Rosen, J. B.: The Gradient Projection Method for Nonlinear Programming. Part II. Nonlinear Constraints. In: *Journal of the Society for Industrial and Applied Mathematics* 9 (1961), Nr. 4, S. 514–532

[Sch05] Schamm, R.: Recent Development of Fuel Cell Cars at Daimler-Chrysler. In: *Symposium on Renewable Energy Systems in the 2005 World Exposition*, 2005

[Sga97] Sgall, Jirí: *On-Line Scheduling - A Survey*. 1997

[SL05] Sutter, Herb ; Larus, James: Software and the Concurrency Revolution. In: *Queue* 3 (2005), September, Nr. 7, S. 54–62

[SL09] Sandholm, Thomas ; Lai, Kevin: MapReduce optimization using regulated dynamic prioritization. In: *Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*, ACM, 2009 (SIGMETRICS '09). – ISBN 978–1–60558–511–6, S. 299–310

[SSK+08] Stankovski, Vlado ; Swain, Martin ; Kravtsov, Valentin ; Niessen, Thomas ; Wegener, Dennis ; Röhm, Matthias ; Trnkoczy, Jernej ; May, Michael ; Franke, Jürgen ; Schuster, Assaf ; Dubitzky, Werner: Digging Deep into the Data Mine with DataMiningGrid. In: *IEEE Internet Computing* 12 (2008), Nr. 6, S. 69–76. – ISSN 1089–7801

[Tan07] Tanenbaum, Andrew S.: *Modern Operating Systems*. 3rd. Prentice Hall Press, 2007. – ISBN 9780136006633

[TCR+12] Tomás, Luis ; Caminero, Agustín C. ; Rana, Omer ; Carrión, Carmen ; Caminero, Blanca: A GridWay-based autonomic network-aware metascheduler. In: *Future Generation Computer Systems* 28 (2012), Nr. 7, S. 1058 – 1069

[TKB09] Talukder, A. K. M. Khaled A. ; Kirley, Michael ; Buyya, Rajkumar: Multiobjective differential evolution for scheduling workflow applications on global Grids. In: *Concurr. Comput. : Pract. Exper.* 21 (2009), September, Nr. 13, S. 1742–1756. – ISSN 1532–0626

[TMX+13] Tu, Manghui ; Ma, Hui ; Xiao, Liangliang ; Yen, I-Ling ; Bastani, Farokh B. ; Xu, Dianxiang: Data Placement in P2P Data Grids Considering the Availability, Security, Access Performance and Load Balancing. In: *Journal Grid Computing* 11 (2013), Nr. 1, S. 103–127

[TRA+06] Tchernykh, Andrei ; Ramírez, Juan M. ; Avetisyan, Arutyun ; Kuzjurin, Nikolai ; Grushin, Dmitri ; Zhuk, Sergey: Two level job-scheduling strategies for a computational grid. In: *Proceedings of the 6th international conference on Parallel Processing and Applied Mathematics*, Springer-Verlag, 2006 (PPAM'05), S. 774–781

[Tru03] Truckenbrodt, A.: DaimlerChrysler Fuel Cell Vehicles - First Vehicles in Customer Hands. In: *F-Cell Symposium*, 2003

[TSA+10] Thusoo, Ashish ; Shao, Zheng ; Anthony, Suresh ; Borthakur, Dhruba ; Jain, Namit ; Sen Sarma, Joydeep ; Murthy, Raghotham ; Liu, Hao: Data warehousing and analytics infrastructure at facebook. In: *Proceedings of the 2010 international conference on*

Management of data. New York, NY, USA : ACM, 2010 (SIGMOD '10). – ISBN 978–1–4503–0032–2, S. 1013–1020

[TTL03] Thain, Douglas ; Tannenbaum, Todd ; Livny, Miron: Condor and the Grid. In: Berman, Fran (Hrsg.) ; Fox, Geoffrey (Hrsg.) ; Hey, Anthony (Hrsg.): *Grid Computing: Making the Global Infrastructure a Reality*. John Wiley & Sons Inc., April 2003

[Uma88] Umar, Amjad: Distributed Database Management Systems Issues and Approaches / The University of Michigan. 1988. – Forschungsbericht

[VB05] Venugopal, Srikumar ; Buyya, Rajkumar: A Deadline and Budget Constrained Scheduling Algorithm for eScience Applications on Data Grids. In: *6th International Conference on Algorithms and Architectures for Parallel Processing*, Springer-Verlag, 2005, S. 60–72

[VB06] Venugopal, Srikumar ; Buyya, Rajkumar: A Set Coverage-based Mapping Heuristic for Scheduling Distributed Data-Intensive Applications on Global Grids. In: *In proceedings of the 7th IEEE/ACM International Conference on Grid Computing(Grid06*, IEEE CS press, 2006

[Ven06] Venugopal, Srikumar: *Scheduling Distributed Data-Intensive Applications on Global Grids*, University of Melbourne, Australia, Diss., July 2006

[WFNR05] Weiss, W. ; Fosmoe, R. ; Nitsche, C. ; Roche, T.: Data Acquisition System for World Wide Fuel Cell. In: *Society of Automotive Engineers SAE World Congress 2005*, 2005

[WSH99] Wolski, Rich ; Spring, Neil T. ; Hayes, Jim: The network weather service: a distributed resource performance forecasting service for metacomputing. In: *Future Gener. Comput. Syst.* 15 (1999), Oktober, Nr. 5-6, S. 757–768

[WXP+04] Wood, D. L. ; Xie, J. ; Pacheco, S. D. ; Davey, J. R. ; Borup, R. L. ; Garzon, F. ; Atanassov, P.: Fuel Cell Seminar, San Antonio. In: *Durability Issues of the PEMFC GDL and MEA Under Steady-State and Drive-Cycle Operating Conditions*, 2004

[ZK08] Zikos, Stylianos ; Karatza, Helen D.: Resource Allocation Strategies in a 2-Level Hierarchical Grid System. In: *Proceedings of the 41st Annual Simulation Symposium*, IEEE Computer Society, 2008 (ANSS-41), S. 157–164

[ZTV08] Zheng, Qin ; Tham, Chen-Khong ; Veeravalli, Bharadwaj: Dynamic Load Balancing and Pricing in Grid Computing with Communication Delay. In: *Journal of Grid Computing* 6 (2008), Nr. 3, S. 239–253