Aufbau eines Grids an der Universität Kassel

Abschlussarbeit zum Diplom I
an der Universität Kassel

Abgabe: 04.01.2007

vorgelegt von
Mirko Dietrich

Gutachter:
Prof. Dr. Claudia Leopold
Prof. Dr. Kurt Geihs
Universität Kassel
Fachbereich 16 – Elektrotechnik / Informatik
Fachgebiet Programmiersprachen/-methodik
Wilhelmshöher Allee 73
34121 Kassel

Erklärung

Nach §13 Absatz 7 der Prüfungsordnung für den Diplomstudiengang Informatik im Fachbereich Elektrotechnik der Universität Kassel versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

_____ , Kassel den 3. Januar 2007

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einführung

Nach dem rapiden Aufstieg der Webtechnologie in den 90er Jahren, wird nun seit einigen Jahren von einer neuen, sich herausbildenden „Zukunftstechnologie“ gesprochen: dem *Grid*. Ihm werden ähnlich schnelle Entwicklungsschübe prognostiziert. Allerdings wird sich seine Ausbreitung vorerst weniger auf private Bereiche ausdehnen, wie es beim Web der Fall war. Vielmehr werden wohl Wissenschaftler und Unternehmen von den Möglichkeiten profitieren. Als Analogie auf das Stromnetz (power grid) [22] soll es den Zugang zu Rechen- und Speicherkapazitäten, sowie anderen Ressourcen, wie z.B. speziellen wissenschaftlichen Apparaturen, insbesondere aber zu Supercomputern leichter machen. Hürden, wie verschiedene Hardwarearchitekturen, eine Vielzahl an unterschiedlichen, oftmals inkompatiblen Betriebssystemen oder die räumliche Ferne, werden überbrückt. Der Anwender, der Zugang zu Ressourcen benötigt, erlangt diese über standardisierte Schnittstellen, ohne dass er sich mit den technischen Details auseinander setzen muss. Darüber hinaus werden weitere Aufgaben, wie die Lastverteilung, Umsetzung von Sicherheit, Abrechnung, usw. transparent von der *Grid Middleware* übernommen.

1.1 Struktur der Arbeit

Im ersten der sieben Kapitel wird ein kurzer Überblick über die Ergebnisse dieser Diplomarbeit gegeben. Das Kapitel 2 erklärt grundlegende Begriffe und erläutert die Aufgaben einer Grid-Middleware. Im darauf folgenden Kapitel 3 werden die wichtigsten Standardisierungsorganisationen und die *Open Grid Services Architecture* vorgestellt. Die drei Middleware-Systeme Globus Toolkit, UNICORE und gLite werden im Kapitel 4 im Detail besprochen. Im Kapitel 5 wird die die bisherige Situation der Computerressourcen am Hochschulrechenzentrum der Universität Kassel geschildert. Außerdem gibt es eine

Dokumentation der Middleware-Installation. Das letzte Kapitel enthält eine Zusammenfassung der Arbeit.

1.2 Ergebnisse

Diese Ausarbeitung entstand während der Durchführung meiner Arbeiten im Rechenzentrum der Universität Kassel. Ich habe eine Grid-Middleware in dem vom Rechenzentrum der Universität Kassel betreuten Linux- und AIX-Cluster installiert. Folgende Ergebnisse werden in der Arbeit näher beschrieben:

1. **Bestandsaufnahme der aktuellen Situation der Cluster am Rechenzentrum der Universität Kassel**

   Die bisherige Situation der Cluster am Rechenzentrum der Universität Kassel wurde beschrieben. Dabei wurde auf die Art des Betriebssystems und die verfügbare Software eingegangen besonders in Hinsicht auf das vorhandene Batch-System. Augenmerk wurde gelegt auf die Zusammenarbeit zwischen dem Betriebssystem, dem Batch-System und in Frage kommenden Grid-Middlewares. Besonders interessant war die Bedienung der Cluster und die Art des Benutzerzugangs.

2. **Ein Vergleich der existierenden Middleware-Systeme in Hinblick auf die Eignung für die Universität Kassel**

   Momentan existieren viele verschiedene Ansätze für die Umsetzung einer Grid-Middleware. Symptomatisch dafür gibt es eine Vielzahl verschiedener Entwicklungen für Grid-Systeme, die teilweise unterschiedliche Herangehensweisen realisieren. Eine Hauptaufgabe der Diplomarbeit war es, einen Überblick der bekanntesten Systeme Globus Toolkit, gLite und UNICORE zu geben (siehe Kapitel 4). Dabei war es wichtig auf die Eignung für die Universität Kassel einzugehen.

   Nach einer ersten Phase der Recherche wurden die Ergebnisse präsentiert. Daraufhin wurde in Abstimmung mit dem Hochschulrechenzentrum die Entscheidung getroffen, das Globus Toolkit zu installieren.

3. **Installation auf den zentralen Rechenknoten des Hochschulrechenzentrums**

   Die ausgewählte Software Globus Toolkit habe ich auf den zentralen Servern des Rechenzentrums installiert. Dafür wurde zuerst eine Testinstallation auf zwei isolierten Clusterrechnern durchgeführt. Nachdem diese Installation erfolgreich war, wurden die Rechner wieder in das Cluster integriert und die Installation auf dem gesamten Linux-Cluster und danach auch auf dem AIX-Cluster durchgeführt. Zum Erproben der Grid-Middleware habe ich ein geeignetes Testprogramm geschrieben, welches Ein- und Ausgabedaten besitzt und die wichtigsten Funktionen der Middleware ausnutzt.

Kapitel 2

Begriffsklärung

2.1 Der Begriff Grid

Für den Begriff Grid sind die Definitionen bisher sehr unterschiedlich. Foster kreierte den Begriff [22] als eine Analogie auf das Stromnetz. So einfach wie beim Stromnetz Elektrizität zur Verfügung steht, so einfach soll im Grid der Zugriff auf Rechen- und Speicherressourcen sein. Dieser Vergleich soll verdeutlichen, wie man sich die Verteilung von Ressourcen zukünftig vorstellt, nämlich transparent und einfach. Die zukünftigen „Konsumenten", also Wissenschaftler, die Leistungen in Anspruch nehmen, sollen über eine geeignete Infrastruktur Ressourcen benutzen können, ohne sich dabei Gedanken um technische Details machen zu müssen. In [22] beschreiben die Autoren die Grundidee des Grids folgendermaßen:

> „A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities."

Doch etwa ein Jahr später wurde diese Definition in [21] erweitert. Es wurde vom Konzept der *Virtuellen Organisationen* gesprochen, als ein mit dem Begriff des Grids verbundendes Paradigma. Der Abschnitt 2.2 soll näher darauf eingehen. Es existiert eine Prüfliste [20], die drei Punkte formuliert anhand denen man ein Grid erkennen kann. Demzufolge werden folgende Anforderungen vorrausgesetzt:

1. **Dezentrale Organisation**
   Ein Grid soll nur dann als solches angesehen werden, wenn es nicht unter einer zentralen Kontrolle steht, sondern einen Verbund von dezentral organisierten Ressourcen darstellt. Im Gegensatz zu einem lokalen

Rechnerverbund kann ein Grid geographisch und institutionell verteilt sein.

2. **Verwendung von standardisierten, offenen und allgemeinen Protokollen**
   Die Basis eines Grids sollen standardisierte und offene Protokolle sein, die auf keine Spezialisierung auf ein gewisses Anwendungsgebiet abzielen.

3. **Dienstgüte**
   Ein Grid soll erhöhte Qualität in Bezug auf Antwortzeit, Durchsatz, Speicherplatz, Rechenkapazität oder anderen Faktoren aufweisen, um die Ansprüche des Benutzers zufrieden zu stellen.

Damit sind vor allem größere, wissenschaftliche Grid-Projekte gemeint, die sich auf die Verwendung von gemeinsamen Protokollen geeinigt haben. Systeme, wie File-Sharing-Netze oder verteilte Programme, wie das SETI@Home-Projekt [7], die den Leerlauf von Desktop-PCs ausnutzen werden in dieser Definition nicht als Grid bezeichnet. Ihnen fehlt eine allgemeine Protokollschnittstelle und sind in ihrer Aufgabe zu speziell, um unter den Begriff Grid zu fallen.

Eine bedeutende, wirtschaftliche Motivation für die Entwicklung des Gridparadigmas ist der Versuch, brachliegende Ressourcen auszunutzen. Dafür werden – im Gegensatz zu einfachen Clustern – Ressourcen zusammengeschaltet, die geographisch und administrativ verteilt sein können. Dadurch kann eine bessere Nutzung der Rechenressourcen erreicht werden. Wenn die vorhandene Hardware optimal genutzt wird, machen sich die Investitionen schneller bezahlt [28].

2.2 Virtuelle Organisationen

Eine weitere wichtige Komponente im Grid-Computing sind *Virtuelle Organisationen*. Eine Gruppe von Personen oder Einrichtungen entscheidet bestimmte Ressourcen, wie Speicherplatz, Rechenkapazität oder auch wissenschaftliche Instrumente untereinander zu teilen. Meist arbeiten sie dabei an einem gemeinsamen Ziel oder einem konkreten Projekt. Sie werden beispielsweise durch Vertragsschlüsse oder Forschungsprojekte gegründet, besitzen oftmals nur eine bestimmte Laufzeit und sind geographisch und administrativ dezentral organisiert. Die Partner sind i.d.R. rechtlich unabhängige Organisationen. Eine Virtuelle Organisation kann gezielt Ressourcen von verschiedenen Stellen bündeln, so dass insgesamt eine Steigerung der Produktivität

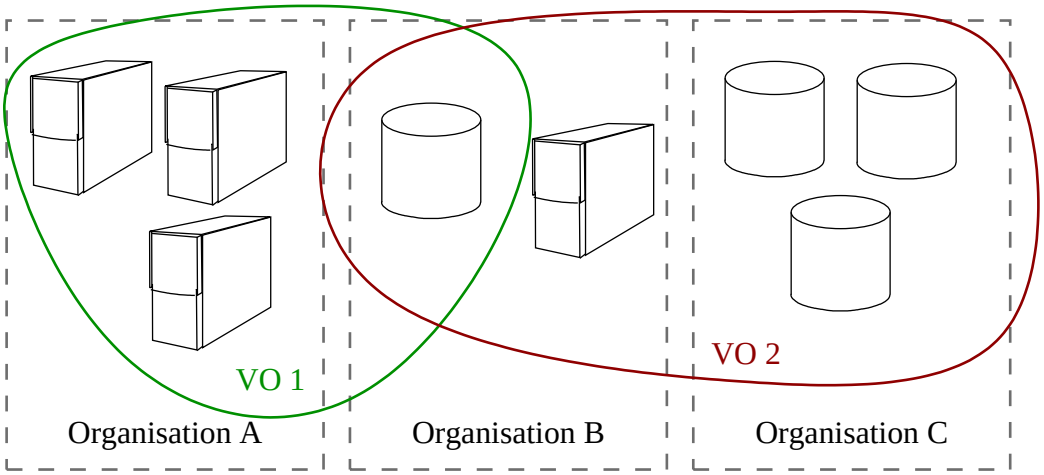

Abbildung 2.1: Schematische Darstellung Virtueller Organisationen

eintritt. Für die technische Unterstützung solcher Kooperationen wird eine spezielle Software verwendet. Die Grid-Middleware stellt die Basisdienste zur Verfügung die von einer Virtuellen Organisation benötigt werden, um gemeinsam arbeiten zu können. Abbildung 2.1 zeigt zwei Virtuelle Organisationen, die von drei Organisationen gebildet werden. Dabei teilen sie sich Speicher- und Rechenressourcen.

2.3 Grid-Middleware

2.3.1 Sicherheit

Public Key Infrastructure

Für das Arbeiten in einer Virtuellen Organisation ist eine funktionierende *Autorisations- und Authentifikationsinfrastruktur* (AAI) unumgänglich. Um ein sicheres Arbeiten über Unternehmens- und Institutsgrenzen hinweg zu gewährleisten, muss für ausreichend Sicherheit gesorgt werden, denn die Verbindungen werden oft über unsichere Netzwerke, wie das Internet geleitet. Eine Verschlüsselung der Verbindung ist wichtig, damit Informationen auf dem Transport nicht von Dritten ausgespäht werden können. Außerdem muss sichergestellt sein, dass nur autorisierte Personen und Dienste Zugriff auf die Ressourcen bekommen, um einen Missbrauch zu vermeiden und es muss genau definierbar sein, wer auf welche Ressourcen und Daten Zugriff hat.

Eine solche AAI ist i.d.R. durch eine *Public key infrastructure* (PKI) mittels *X.509-Zertifikate* realisiert. Diese Technik ist weit verbreitet und es existieren ausgiebig getestete und zuverlässige, freie Implementierungen. Durch die Signierung von Zertifikaten lassen sich sehr unkompliziert Vertrauensverhältnisse zwischen Organisationen ausdrücken. Damit zwei Stellen miteinander kommunizieren können, müssen sie in einer vertrauensvollen Beziehung zueinander stehen. Dies wird erreicht, indem beide Teil einer gemeinsamen Vertrauenshierarchie sind. Abschnitt 3.3 stellt eine für Grid-Anwendungen etablierte Vertrauenskette vor.

Proxyzertifikate

Allerdings gibt es auch Fähigkeiten, welche die PKI nicht unterstützt [34]:

- Wenn ein Grid-Benutzer mit einer Grid-Umgebung interagiert, z.B. Jobs absetzt, Daten kopiert, usw., dann tritt er in Kontakt mit vielen einzelnen Diensten dieser Umgebung. Jeder dieser Dienste verlangt eine Authentifizierung des Benutzers. So müsste er sich bei jedem einzelnen wieder und wieder persönlich anmelden. Normalerweise sind private Schlüssel mit einem Passwort geschützt. Dieses müsste bei jeder Anmeldung erneut eingegeben werden, da eine Speicherung des privaten Schlüssels in Klartext ein Sicherheitsrisiko darstellt. Die ständige Eingabe des Passworts ist für ein bequemes Arbeiten auszuschließen. Besonders im Hinblick auf Stapelverarbeitung, wo der Benutzer nicht zugegen ist, wird deutlich, dass eine andere Technik benutzt werden sollte. Wünschenswert ist eine einmalige Eingabe des Passwort. Danach muss das Passwort für eine bestimmte Zeit nicht erneut eingegeben werden. Hier spricht man vom *Single-Sign-On*.

- In einer Grid-Umgebung kann es aufgrund der Natur des Grids passieren, dass ein Auftrag oder Teile davon an andere Stellen weitergereicht werden. Beispielsweise möchte ein Benutzer im Zuge der Bearbeitung eines Auftrags Daten von einer bestimmten Datenbank holen lassen. Der Teil der Software, der für die Abarbeitung des Auftrags zuständig ist, muss in der Lage sein, im Namen des Benutzers agieren zu können, ohne dass der Benutzer dafür anwesend sein muss und sich persönlich bei der Datenbank authentifiziert. Diese *Delegation* kann prinzipiell über beliebig viele Ebenen stattfinden.

Eine Lösung für dieses Problem stellen *Proxyzertifikate* dar. X.509-Benutzerzertifikate werden von Zertifizierungsstellen signiert. Proxyzertifikate werden vom Benutzer bzw. von einem anderen Proxyzertifikat signiert. Eine

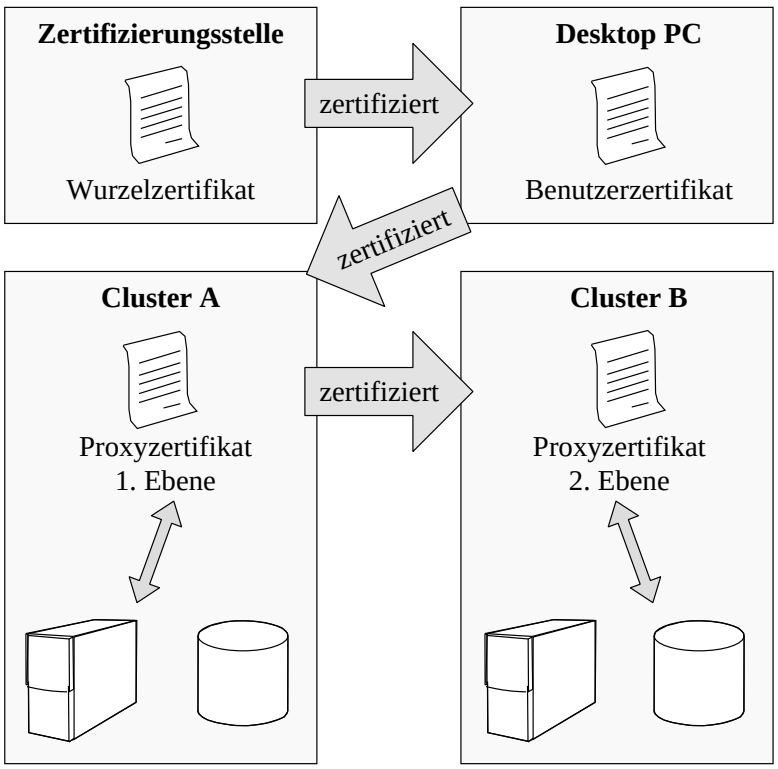

Abbildung 2.2: Vertrauenskette durch Proxyzertifikate

Kompromittierung eines Proxyzertifikats ist weitaus weniger problematisch, als die eines normalen Benutzerzertifikats: Die Gültigkeit des Proxyzertifikats verfällt innerhalb weniger Stunden und wird für den Angreifer nutzlos. Auf Basis eines Proxyzertifikats können weitere Proxyzertifikate erstellt werden, die wieder weitere Proxyzertifikate signieren können, usw., so dass die Delegationstiefe theoretisch beliebig fortgesetzt werden kann. Proxyzertifikate sind normale X.509-Zertifikate mit zusätzlichen Feldern. Dort kann der Aussteller des Proxyzertifikats definieren, bis zu welcher Ebene weitere Proxyzertifikate erstellt werden können.

Der Benutzer authentifiziert sich nicht mehr direkt bei einem Dienst, sondern lässt von vornherein ein Proxyzertifikat erstellen. Dieses wird dann automatisch verwendet, sobald ein Dienst eine Authentifizierung verlangt. Dadurch ist das Single-Sign-On realisiert. Braucht eine Aufgabe länger, als das Proxyzertifikat gültig ist, so muss der Benutzer dafür Sorge tragen, dass er rechtzeitig ein weiteres Proxyzertifikat erstellt, welches das vorherige ersetzt.

Sollen Benutzerprivilegien delegiert werden, so wird an der Stelle, die Privilegien empfangen soll, ein weiteres Proxyzertifikat erstellt, welches vom

ersten Proxyzertifikat signiert wird. Dadurch entsteht eine Vertrauenskette, die vom Benutzer ausgeht. Abbildung 2.2 zeigt eine solche Kette inklusive der Zertifizierungsstelle und einer Delegation.

2.3.2 Ausführen von Jobs

Eine zentrale Funktion einer Grid-Middleware ist die Ausführung von Programmen. Der Benutzer schickt einen Auftrag an eine Middleware, die dann die Ausführung des Programms übernimmt. Die Middleware entscheidet auf welcher Maschine die Ausführung stattfindet etwa anhand der momentanen Auslastung oder speziellen Anforderungen des Auftrags. Aufträge können in Unteraufgaben unterteilt sein, die gegenseitige Abhängigkeiten besitzen. Diese Abhängigkeiten sinnvoll aufzulösen ist auch Aufgabe der Middleware. Entweder wird das Scheduling der Jobs von einem lokalen Batch-System erledigt, die Middleware erledigt das Scheduling selbst oder es wird eine Kombination aus beiden Varianten durchgeführt.

2.3.3 Verteilung der Daten

Ein weitere Anforderung an eine Grid-Middleware ist die Verteilung von Daten. Hiermit wird das Kopieren von beliebigen Datenmengen zwischen verschiedenen Systemen bezeichnet. Klassischerweise besteht ein Auftrag aus Eingangs- und Ausgangsdaten. Zuerst müssen die Daten von dem System, von dem aus der Auftrag abgeschickt wird, zu dem Cluster kopiert werden, auf dem der Auftrag ausgeführt wird. Bedarfsweise kann auch das auszuführende Programm selber im Vorfeld auf das Zielsystem kopiert werden. Dies ist die einfachste Variante. Die Middleware muss also irgendeine Art Dateiübertragungsdienst anbieten. Desweiteren kann es zu Übertragungen zwischen verschiedenen Clustern kommen. Oft sind die Eingabedaten nicht auf dem Rechner des Benutzers, sondern in einem externen Datenspeicher abgelegt. So muss das Cluster, welches den Auftrag bekommt, eigenständig eine Verbindung zu der Datenbank aufbauen. Auch die Ergebnisse können auf diese Weise an einem bestimmten Ort abgelegt werden.

Außerdem werden Replikationsmechanismen benötigt. Sollen an zwei oder mehreren Stellen die gleichen Daten vorgehalten werden, sei es aus Sicherungs- oder Performancegründen, so muss für eine konsistente und möglichst automatische Abgleichung der Datenbestände gesorgt werden. Auch hier kann die Middleware geeignete Dienste anbieten. Besonderen Wert wird auf die Skalierbarkeit, Sicherheit und Performance eines solchen Systems gelegt.

2.3.4 Monitoring

Setzt ein Benutzer Rechenaufträge ab, so kann er erstmal nicht direkt überprüfen, wie weit diese abgearbeitet sind oder ob sie überhaupt schon in der Bearbeitung sind. Hierfür muss die Middleware geeignete Dienste anbieten, die dem Nutzer ermöglichen, festzustellen, ob ein Auftrag fertig oder noch in Bearbeitung ist und eventuell auch in welchem Stadium der Bearbeitung er sich gerade befindet. So hat der Benutzer immer Überblick über seine aktuellen Aufträge. Dieser Dienst kann dem Benutzer über eine Webschnittstelle, GUI oder auch Kommandozeile angeboten werden.

2.3.5 Service Discovery

Unter *Service Discovery* versteht man die Möglichkeit Informationen über die verfügbaren Dienste und deren Eigenschaften in einer Grid-Umgebung zu erhalten. Dienste sind z.B. die Möglichkeit Jobs abzusetzen und zu überwachen oder Daten zu speichern und abzurufen. Dabei kann u.a. abgefragt werden, welche Dienste ein Cluster anbietet, die Anzahl der Rechenknoten, verfügbarer Speicherplatz, momentane Auslastung, usw. Die Informationen sollten in einer maschinenlesbaren Form ausgegeben werden. Die Idee dabei ist, dass zukünftig sogenannte *Resource Broker* eigenständig in der Lage sein sollen, Entscheidungen zu treffen, beispielsweise auf welchem Cluster sie bestimmte Aufträge ausführen. Grundlage für diese Entscheidungen können sein: Kostenminimierung (verschiedene Anbieter verlangen unterschiedliche Preise), die Ausstattung an verschiedenen Clustern ist unterschiedlich und evtl. ungenügend für bestimmte Probleme, usw. Durch Service Discovery kann der Resource Broker feststellen, welche Eigenschaften und Besonderheiten und welche Ressourcen überhaupt ein Cluster zur Verfügung stellt. Eine Schnittstelle für den Benutzer kann dann z.B. als Webplattform implementiert werden.

2.3.6 Accounting/Billing

Eine Vereinbarung in einer Virtuellen Organisation kann sein, dass die Partner alle gegenseitig genutzten Dienstleistungen finanziell ausgleichen. Dafür muss genau messbar sein, wie viel Speicherplatz, Rechenzeit und andere Ressourcen verwendet wurden. Die Middleware sollte also in der Lage sein, alle Aktivitäten zu protokollieren und eine Aufschlüsselung nach Benutzer, Gruppe und Virtueller Organisation vornehmen können.

Kapitel 3

Standardisierung

3.1 Standardisierungsorganisationen

Um die Standardisierung und damit die Zusammenarbeit im Bereich Grid-Computing zu verbessern, haben sich Organisationen gebildet, die versuchen gemeinsame Entwürfe auszuarbeiten und Standards zu entwickeln.

3.1.1 Global Grid Forum

Noch vor dem Jahr 2000 trafen sich in den USA eine Gruppe von Wissenschaftlern, um die Standardisierung im Grid-Bereich voranzutreiben. Nachdem in Europa und Japan ähnliche Bestrebungen begonnen hatten, traf man sich zum ersten Treffen des *Global Grid Forums* Anfang 2001. Gemeinsam sollten fortan Standards geschaffen werden in einer Weise, die dem Vorgehen bei der *Internet Engineering Task Force* ähnelt. Die wichtigsten bisher veröffentlichten Standards sind im Abschnitt 3.2 näher beschrieben.

3.1.2 Enterprise Grid Alliance

Die *Enterprise Grid Alliance* ist ein Verbund von Unternehmen, die in der Entwicklung von Grid-Produkten involviert sind. Sie war bisher der industrielle Gegenpart zum Global Grid Forum, welches eher wissenschaftliche und akademische Interessen vertrat. Unter anderen sind Unternehmen, wie *Fujitsu Siemens Computers*, *Hewlett-Packard*, *Intel* und *Sun Microsystems* Mitglieder.

3.1.3 Open Grid Forum

Im Juni 2006 haben sich das Global Grid Forum und die Enterprise Grid Alliance zum *Open Grid Forum* zusammen geschlossen. Diese neu gegründete Vereinigung bringt industrielle, sowie akademische Organisationen an einen Tisch und will so die Entwicklung und den Einsatz von Grid-Systemen weiter vorantreiben.

3.2 Grid-Standards

3.2.1 Open Grid Services Architecture

Um die Interoperabilität zwischen den verschiedenen Grid-Systemen zu erhöhen hat das Open Grid Forum (damals noch Global Grid Forum) im Jahre 2003 einen Standard entwickelt und herausgegeben. Die *Open Grid Services Architecture* (OGSA) [18] ist ein auf Web Services aufbauendes System, welches Basisdienste für die gängigsten Aufgaben in einer Gridumgebung beschreibt. Die Basisdienste der OGSA bieten die in Abschnitt 2.3 erläuterten Grundfunktionalitäten einer Grid-Middleware. Ziel ist es, die Schnittstellen zu standardisieren und so die einzelnen Komponenten verschiedener Middleware-Systeme kompatibel untereinander zu machen. Mittelfristig soll die Kommunikation zwischen verschiedenen Systemen und auch innerhalb einer Middleware hauptsächlich über Web Services funktionieren und alte Protokolle werden nach und nach ersetzt. Folgende Basisdienste wurden beschrieben [19]:

- **Infrastructure Services:**

  Alle von der OGSA definierten Dienste funktionieren über einheitliche Schnittstellen. Als Basis wurden *Web Services* ausgewählt. Sie sind unabhängig vom Betriebssystem und der Programmiersprache, außerdem basieren sie auf einfachen Protokollen und beherrschen von sich aus Merkmale wie Verschlüsselung und Authentifizierung. Ein weiterer Nutzen von Web Services ist, dass sie großen Rückhalt in der Industrie genießen. Die *Web Services Description Language* (WSDL) ist ein Weg, Dienste formal zu beschreiben. Auf dieser Grundlage wurden die weiteren Grid-Dienste spezifiziert. Allerdings gibt es auch Anforderungen, denen Web Services bisher nicht genügten. Vor allem die Zustandslosigkeit der Web Services war ein Problem. Bei Grid-Anwendungen sollen bestimmte Informationen über mehrere Verbindungen hinweg gespeichert werden. Hierfür sollte eine Erweiterung der Web Services stattfin-

den. Das Global Grid Forum entwickelte die *Open Grid Services Infrastructure* (OGSI), eine statusorientierte Erweiterung der Web Services [31]. Doch OGSI wurde obsolet, nachdem die OASIS, ein Konsortium für die Standardisierung im Bereich Web Services, eine eigene Erweiterung herausbrachte. Das *Web Services Resource Framework* (WSRF) spezifiziert die nötigen Eigenschaften und ersetzt OGSI.

- **Execution Management Services:**

  Eine der Hauptaufgaben eines Grids ist die Ausführung von Arbeitsaufträgen. Diese werden über das Netzwerk abgeschickt und an einem i.d.R. entfernten Ort ausgeführt. Die Aufgabe dieses Dienstes ist es, Jobs anzunehmen und zu entscheiden, wo sie ausgeführt werden. Dabei können verschiedenste Angaben die Entscheidung beeinflussen. Beispielsweise die Auslastung eines Clusters oder die Anzahl der Prozessoren. Ein Job kann aus Unteraufgaben bestehen, die wiederum Abhängigkeiten untereinander besitzen können. Diese aufzulösen und eine sinnvolle Abarbeitungsreihenfolge herzustellen ist auch Aufgabe dieses Dienstes. Bricht ein Job während der Abarbeitung ab, so muss entschieden werden, ob dieser neu gestartet werden soll. Bei einem Cluster gibt es meistens eine Warteschlange für die abzuarbeitenden Jobs. Eine effiziente Reihenfolge zu finden, bei der die Ressourcen möglichst optimal ausgenutzt werden, ist eine weitere wichtige Aufgabe.

- **Data Services:**

  Bei Datendiensten geht es darum, Möglichkeiten anzubieten, entweder auf Daten zuzugreifen oder Daten abzuspeichern. Neben Dateien gibt es auch noch andere Formen, in denen die Daten vorliegen können, u.a. Datenströme, relationale Datenbanksysteme oder auch Geräte, wie z.B. Sensoren. Dabei kann es sich in der einfachsten Variante darum drehen, Dateien von einem Standort zu einem anderen zu kopieren. Doch es existieren komplexere Szenarien. Bei einem Job werden typischerweise Eingangsdaten verarbeitet. Diese zum Ort der Ausführung zu übermitteln und die Resultate an einen geeigneten Ort zu kopieren wird unter dem Begriff *File Staging* zusammengefasst. Auch die auszuführende Datei selber kann im Vorfeld übertragen werden.

  Eine weitere wichtige Anwendung sind Replikationsdienste. Dabei existieren in einem Grid mehrere Stellen, an denen die selben Daten vorhanden sind. Die Datendienste müssen sich um einen konsistenten Zustand der Daten kümmern und dafür Sorge tragen, dass Änderungen

zügig weiterverteilt werden. Desweiteren gibt es Szenarien, in denen zusammengehörige Daten über viele Stellen verteilt sind, aber dem Anwender als ein einheitliches virtuelles Dateisystem dargestellt werden. Er weiß u.U. gar nicht, wo sich die Daten in Wirklichkeit befinden. Für ihn erscheinen sie, wie eine einzelne lokale Festplatte. Dabei muss der Datendienst sich um das Zusammenfügen der einzelnen Fragmente kümmern.

- **Security Services:**

  Sicherheit ist eine ernstzunehmende Aufgabe, denn die Anforderungen sind, je nach Unternehmen, Einrichtung und Virtueller Organisation sehr verschieden. Oftmals überlagern sich die Sicherheitsanforderungen, z.B. wenn ein Unternehmen Teil einer Virtuellen Organisation ist. Dann muss die Sicherheitsinfrastruktur die Anforderungen des Unternehmens sowie der Virtuellen Organisation in sich vereinen.

  Wie auch immer das Sicherheitsmodell implementiert ist, muss es eine Reihe von Aspekten berücksichtigen. Durch die Authentifikation wird die Identität einer Person, eines Computer oder eines Dienst sichergestellt und dadurch Missbrauch durch Dritte vermieden. Jede Identität wird durch einen *Distinguished Name* (DN) ausgedrückt. Diese Form kommt aus dem Verzeichnisdienst X.500 und hat folgenden Aufbau [33]:

  ```
  /C=DE/O=GridGermany/OU=Universitaet Kassel
  /OU=Hochschulrechenzentrum/CN=Max Mustermann
  ```

  Dabei bezeichnet jedes einzelne Feld eine organisatorische Einheit. So besitzt jede Instanz im Grid einen eindeutigen und unterscheidbaren Namen. Zu jedem dieser Namen gehört ein X.509 Zertifikat. Abschnitt 3.3 berichtet ausführlich über eine existierende Vertrauensstruktur in Grid-Umgebungen. Es muss außerdem sichergestellt sein, dass die verschiedenen Verfahren der Authentifikation untereinander funktionieren. Dafür gibt es eine sogenannte *credential conversion*, bei der eine Form der Authentifikation in eine andere überführt werden kann, damit verschiedene Systeme miteinander kommunizieren können. Unter Autorisation versteht man die Möglichkeit verschiedenen Individuen verschiedene Rechte und Zugriffsmöglichkeiten zuzuordnen. Schließlich muss

dafür gesorgt werden, dass private Daten, die zu einer Virtuellen Organisation gehören, geschützt sind vor fremden Zugriff. Alle Zugriffe und Ereignisse im Sicherheitssystem werden protokolliert.

- **Resource Management Services:**

  Ein Aspekt in der OGSA ist die Fähigkeit eines Grids sich selbst zu steuern. Dabei kann es um physische Vorgänge gehen, wie z.B. Steuerung von wissenschaftlichen Geräten, Starten und Herunterfahren eines Rechenknotens oder die Konfiguration von Netzwerken. Außerdem können die Grid-Dienste und -Ressourcen selber verwaltet werden, z.B. das bedarfsweise Starten und Beenden von einzelnen Diensten aufgrund einer Anfrage oder auch im Fehlerfall. Diese Klasse von Diensten bietet die Mechanismen Ressourcen zu verwalten, während die im nächsten Abschnitt beschriebenen Self-management Services Entscheidungen treffen, in welchen Situationen welche Maßnahmen durchgeführt werden.

- **Self-management Services:**

  Eine Grid-Umgebung soll in der Lage sein, die eigenen Zustände und Fehler eigenständig zu entdecken und zu beheben. Dafür müssen geeignete Überwachungsdienste verwendet werden und aufgrund bestimmter Regeln Entscheidungen getroffen werden. Beispielsweise bei einem Eintreffen eines Jobs, der Zugang zu einem relationalen Datenbanksystem benötigt, muss dieses wahlweise gestartet und danach wieder beendet werden. Oder einzelne Rechenknoten fallen aus, so müssen sie neugestartet, ganz abgeschaltet oder aber durch andere Ressourcen ersetzt werden. In Spitzenzeiten, wo die Systemlast sehr hoch ist, können dynamisch Ressourcen dazugeschaltet werden und bei Verminderung der Last Ressourcen abgeschaltet oder anderen Aufgaben zugeteilt werden, um Kosten zu sparen. Im Gegensatz zu den Resource Management Services geht es bei den Self-management Services eher um die Selbstdiagnose im Fehlerfall und die Anpassung an veränderte Umgebungsbedingungen, wie dynamische Umverteilung von Ressourcen. Sie greifen auf die Information Services zu, um Daten für die Entscheidungsfindung zu sammeln und lösen dann Vorgänge aus, indem sie Resource Management Services verwenden.

- **Information Services:**

Die OGSA bietet Möglichkeiten Informationen in einer maschinenlesbaren Form abzufragen. Diese beziehen sich auf den Gesamtzustand eines Grids, die einzelnen Cluster und teilnehmende Virtuelle Organisationen. Beispielsweise kann ein Benutzer oder ein Dienst mit einer spezifischen Anforderung über einen zentralen Informationsdienst Datenbanksysteme finden, welche seine Aufgaben lösen können und dann einen der Kandidaten wählen. Oder es werden bei der Ausführung einer Aufgabe periodisch Statusprotokolle generiert, die dann ausgegeben werden können, um nachzuvollziehen, in welchem Zustand sich die Abarbeitung gerade befindet. Dienste die einen Informationsdienst nutzen sind u.a. das Job Management, der Resource Broker, das Self Management oder aber eine Überwachungsschnittstelle für den Benutzer.

3.2.2 Job Submission Description Language

Um Jobs spezifizieren zu können, verwenden viele Grid-Middlewares spezielle Beschreibungssprachen. Bisher besitzt jede Middleware eine eigene Sprache für diesen Zweck, was eine Zusammenarbeit zwischen den verschiedenen Middlewares erschwert. Die *Job Submission Description Language* ist eine auf XML basierende Sprache um Aufgaben für Batch-Systeme zu beschreiben und wurde – wie die OGSA auch – vom Open Grid Forum entwickelt. Es lassen sich auch Anforderungen ausdrücken. Dazu zählen u.a. Speicherbedarf, Prozessorzeit und Hardwarearchitektur. Mit Hilfe dieser Beschreibungssprache können Aufgaben und Unteraufgaben definiert werden. Außerdem können Abhängigkeiten zwischen den einzelnen Aufgaben ausgedrückt werden. Unteraufgaben können z.B. auch das Übertragen der Eingangs- und Ausgangsdaten sein [9].

3.3 International Grid Trust Federation

Die *International Grid Trust Federation* (IGTF) ist eine vom Open Grid Forum anerkannte Organisation und bildet die Grundlage für die weltweite, sichere Bildung von wissenschaftlichen Gridgemeinschaften. Sie hat sich das erste Mal beim siebten Global Grid Forum in Tokio getroffen und trifft sich weiterhin bei Zusammenkünften des Grid Forums. Die IGTF besitzt drei *Policy Management Authorities* (PMA) als Mitglieder. Diese sind eigenständige Organisationen, welche die Wurzelzertifikate für Grid-Projekte der jeweiligen Regionen herausgeben und damit Vertrauensstellen für die Grid-Gemeinschaft bilden. Über die International Grid Trust Federation sind

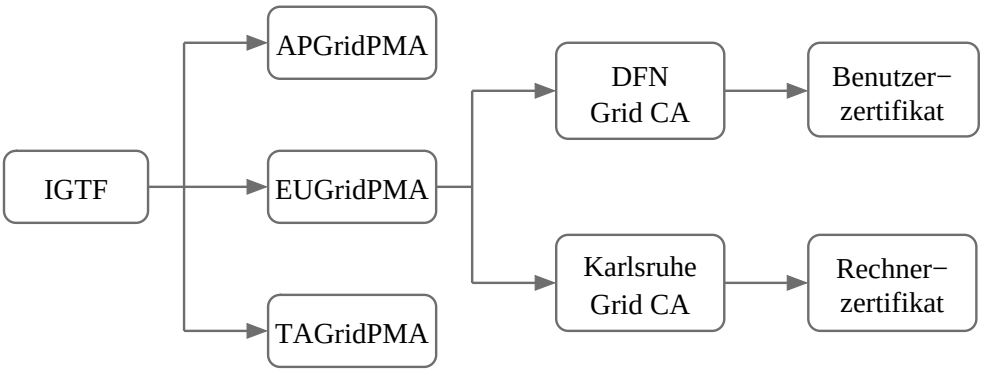

Abbildung 3.1: Hierarchie der Vertrauensstellen (Ausschnitt)

diese drei Organisationen in einem Vertrauensverhältnis untereinander verbunden. Die Mitglieder sind:

1. Asia Pacific Grid PMA (APGridPMA)
2. European Grid PMA (EUGridPMA)
3. The Americas Grid PMA (TAGridPMA)

Diese geben Zertifikate an andere Authentifizierungsstellen aus. Die EUGridPMA selbst stellt Zertifikate für Institutionen innerhalb Europas aus. Für Deutschland sind das zur Zeit die *GermanGrid Certificate Authority* (CA), eine an der Universität Karlsruhe ansässige Zertifizierungsstelle, und die CA des DFN. Die IGTF schreibt vor, dass alle ausgestellten Zertifikate durch persönliche Vorstellung mit Lichtbildausweis erfolgen müssen. Da es aber nur zwei Zertifizierungsstellen in Deutschland gibt, wurde eine weitere Ebene eingeführt: die *Registry authority* (RA). Sie ist berechtigt ein Schlüsselpaar von einer Person entgegenzunehmen, deren Identität per Lichtbildausweis zu bestätigen und die Signierungsanforderung dann zu der CA weiterzuleiten. Diese schickt das signierte Zertifikat dann direkt zurück an die jeweilige Person. RA werden momentan an allen beteiligten Instituten gebildet. Abbildung 3.1 zeigt die Zertifikatshierarchie der EUGridPMA.

Kapitel 4

Vergleich aktueller Grid Middleware Systeme

4.1 Globus Toolkit

Das *Globus Toolkit* gilt als die bedeutendste Grid-Middleware. 1995 wurde auf der *Supercomputing Conference* das *I-WAY Projekt* [13] vorgestellt, ein US-amerikanisches Projekt, welches versuchte nationale Computerressourcen miteinander zu verbinden und eine abstrakte Serviceschicht zu implementieren. Aus diesem Projekt wuchs das Globus Toolkit hervor, welches 1998 als Version 1.0 veröffentlicht wurde. Das Programm unterstützt praktisch alle Merkmale, die von einer Grid-Middleware verlangt werden. Es gilt als de facto Standard und wird von der *Globus Alliance* entwickelt. Da es unter der *Apache License* veröffentlicht wurde, ist es quellenöffentlich verfügbar. Es wird weltweit stark eingesetzt und ist daher sehr erprobt. Abbildung 4.1 zeigt die wichtigsten Komponenten im Globus Toolkit.

4.1.1 Versionen

Die im Jahr 2002 veröffentlichte Version 2.0 des Toolkits benutzt eigene Protokolle und ist daher vollständig inkompatibel zu anderen Grid-Middlewares. Mit der Version 3.0 des Globus Toolkits sollten erstmal standardisierte Protokolle zum Einsatz kommen. Diese Version basiert bereits auf der OGSA und dem von Global Grid Forum spezifizierten OGSI (siehe auch Abschnitt 3.2.1). Nachdem OGSI durch WSRF ersetzt wurde, hat man eine weitere Anpassung vorgenommen. Die Globus Toolkit Version 4.0, die im Jahre 2005 erschien, ist nun konform mit den Standards der Web Service Community. Obwohl weite Teile der Software nun auf Web Services aufbauen, sind in der Software nach wie vor die sog. *Pre-WS-Komponenten*, also die Teile der

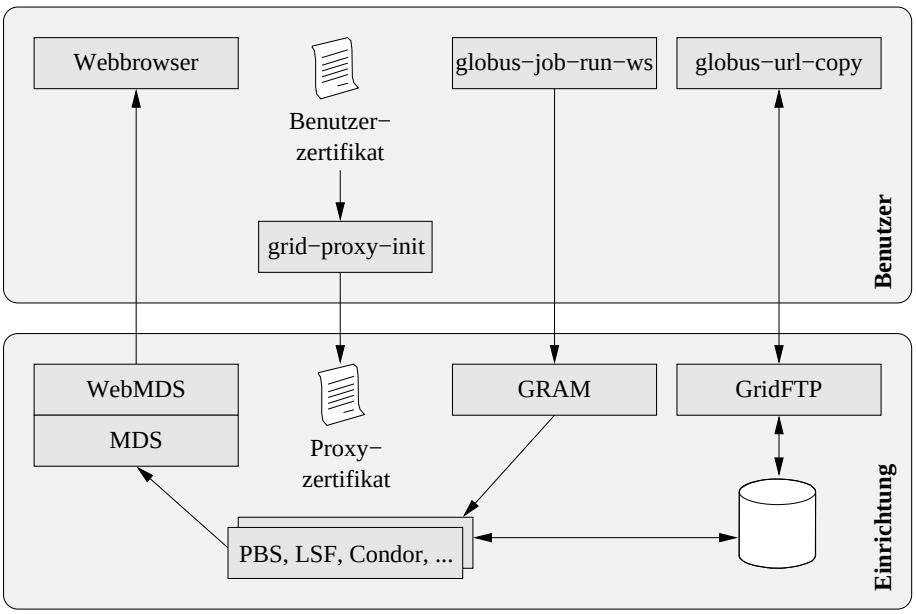

Abbildung 4.1: Komponenten im Globus Toolkit

Software die auf eigenen Protokollen aufbauen, enthalten. Dadurch soll der Übergang zur neuesten Version einfacher werden, da viele im Produktionseinsatz befindlichen Systeme noch Globus in der Version 2 verwenden. Das GT unterstützt zur Zeit die vom Open Grid Forum vorgeschlagenen Standards am besten von allen Grid-Middlewares.

4.1.2 Funktionsumfang

Die Sicherheitsschicht des Programmpakets, die sog. *Grid Security Infrastructure* (GSI), basiert auf *asymmetrischer Verschlüsselung*, auch *Public-key cryptography* genannt. Durch X.509 Zertifikate kann Vertrauen ausgedrückt werden. Dabei werden typischerweise Zertifikate der in Abschnitt 3.3 beschriebenen Hierarchie verwendet. Globus ist dann in der Lage, eine gegenseitige Authentifizierung durchzuführen. Wenn also ein Benutzer sich bei einem Dienst anmeldet, so kann der Server sicherstellen, dass der Benutzer derjenige ist, der er vorgibt zu sein und andersherum, ist der Benutzer in der Gewissheit, dass es sich wirklich um den Server handelt, mit dem er kommunizieren möchte. Es kann eine gesicherte Übertragung stattfinden, die allerdings einen Overhead erzeugt. Auf diese Weise kann die Kommunikation nicht abgehört werden. Außerdem wird eine Integritätsprüfung vorgenommen. Damit wird sichergestellt, dass übertragene Informationen auf dem

Weg nicht verändert wurden.

Bei Globus werden alle Authentifikationen vom Benutzer über Proxyzertifikate geregelt. Sie erlauben Single-Sign-On, sowie Delegation. Vor jeder Interaktion mit der Grid-Middleware muss also erstmal ein Proxyzertifikat erzeugt werden. Der private Schlüssel selbst ist i.d.R. im Home-Verzeichnis des Benutzers gespeichert und mit einem symmetrischen Verfahren verschlüsselt, so dass er ein Passwort eingeben muss, um den Schlüssel freizuschalten. Außerdem gibt es die Möglichkeit den privaten Schlüssel auf einer sog. *Smartcard* abzuspeichern. So wird der Schlüssel über ein Kartenlesegerät eingegeben, wodurch die Wahrscheinlichkeit einer Kompromittierung verringert wird. Dann gibt es noch ein Softwareprojekt namens *MyProxy*. Dabei wird ein Server aufgesetzt, der die privaten Schlüssel von Benutzern zentral speichert. Der Benutzer kann sich dann von jedem Ort aus bei dem Server anmelden und seinen privaten Schlüssel über einen sicheren Kanal beziehen. Das hat den Vorteil, dass der Schlüssel nicht im Home-Verzeichnis des Benutzers liegen muss. So ist er vor Diebstahl geschützt und der Benutzer kann von überall aus auf seinen Schlüssel zugreifen. Allerdings bildet die zentrale Schlüsselverwaltung auch eine Schwachstelle im System (single point of failure).

Für die Autorisierung innerhalb einer Virtuellen Organisation reicht die bloße Zuordnung zwischen dem DN, also dem X.500-Namen aus dem Zertifikat, und dem lokalen Benutzer oft nicht, denn die Zugriffsrechte sind an verschiedenen Einrichtungen unterschiedlich oder bestimmte Benutzerkonten existieren nicht. Um die Nutzerrechte innerhalb einer Virtuellen Organisation konsistent zu halten, wurde der *Community Authorization Service* (CAS) entwickelt. Durch diesen Dienst können Zugriffsrechte für jeden DN definiert werden und nicht wie bisher für die einzelnen Benutzerkonten. Der Dienst kann von verschiedenen Einrichtungen zentral angesprochen werden [27].

Der *Grid Resource Allocation and Management* (GRAM) ist das Programmteil in der Globus Software, dass Jobs annimmt, weiterdelegiert, überwacht und verwaltet. Es existiert in einer Variante, die Web Services verwendet, sowie einer Implementation die noch vom Globus Toolkit 2 stammt. Die ankommenden Jobs werden entweder direkt als Unterprozess ausgeführt oder an ein lokales Batch-System, wie OpenPBS weitergeleitet.

Die Jobs werden mittels der *Resource Specification Language* (RSL) definiert. Dort wird genau festgelegt, welche Daten wann kopiert werden müssen, welche Programme ausgeführt werden sollen. Außerdem können Unteraufgaben, Abhängigkeiten und bestimmte Anforderungen an Hauptspeicher und Prozessorzeit definiert werden. Die vom Global Grid Forum für diesen Zweck spezifizierte Sprache JSDL (siehe Abschnitt 3.2.2) wird derzeit noch nicht unterstützt. Die nächste Version des Toolkits soll aber eine Unterstützung

beinhalten.

Außerdem besteht die Möglichkeit, eine Delegation an einen anderen GRAM-Server vorzunehmen. Allerdings ist das nicht dynamisch möglich, sondern muss in der Jobspezifikation statisch definiert werden. Es existiert allerdings experimentelle Software, die diese Aufgabe dynamisch erfüllen können. Der *Grid Service Broker* [32], der *GridWay Metascheduler* [24] sowie das *GridLab Resource Management System* (GRMS) [25] sind experimentelle Projekte, die Aufgaben eines Resource Broker übernehmen sollen.

Für das einfache Kopieren von Daten wird *GridFTP* genutzt. Dies ist eine Erweiterung des *File Transfer Protocols* um die Authentifikationsmechanismen des Globus Toolkits. Der Dienst *Reliable File Transfer* (RFT) basiert auf GridFTP und kann zusätzlich die Metadaten einer Dateiübertragung in einer Datenbank speichern und bietet eine Web Services Schnittstelle an. Dadurch muss während der Dateiübertragung keine Verbindung vom FTP Client aktiv sein. RFT kümmert sich dabei auch um die Wiederaufnahme der Übertragung im Fehlerfall.

Für die Replikation von Daten bietet das Globus Toolkit den *Replica Location Service* (RLS) an. Dieser dient als Katalog für logische Dateinamen, die auf physikalische Dateien im Grid abgebildet werden. Dabei meint der logische Dateiname einen Bezeichner für eine Datei, die aber physikalisch mehrfach im Grid abgespeichert sein kann. Dadurch können Dateien in einem Grid verteilt werden und sind dennoch für den Benutzer transparent zugreifbar. Der *Data Replication Service* (DRS) kümmert sich um eine automatische Konsistenz der Daten zwischen verschiedenen Standorten.

Die *Data Access and Integration* (DAI) ist ein Projekt, welches den Zugriff auf relationale Datenbanksysteme und XML Dokumente innerhalb eines Grids vereinfachen soll [10]. Es basiert bisher noch auf der OGSI aus der Globus Toolkit Version 3, wird aber demnächst auf volle Web Services Konformität umgestellt.

Die Implementation für Informationsdienste im Globus Toolkit ist das *Monitoring and Discovery System* (MDS) und existiert jeweils als Version mit und ohne Web Services. Die letztere wird mit der Zeit komplett durch die Web Services Variante ersetzt werden. Der Dienst stellt Informationen in Form von XML Dokumenten bereit. Die Quelle für die Informationen stammen vom GRAM, sowie vom RFT Service. Zusätzlich können externe Monitoring Systeme angeschlossen werden. Dazu zählt *Hawkeye* für die Überwachung von Condor Systemen sowie *Ganglia* für die Überwachung von Batch-Systemen wie OpenPBS. Durch die Tomcat Anwendung *WebMDS* lassen sich die Informationen über eine Web-Schnittstelle menschenlesbar anzeigen. Ein sog. Trigger Service ist in der Lage bei bestimmten Zuständen Aktionen auszulösen. So kann beispielsweise beim Ausfall eines Rechenknotens

eine E-Mail an den Administrator geschickt werden oder andere Aufgaben automatisch ausgeführt werden.

Zum Abrechnen von Rechenzeit und anderen Ressourcen gibt es viele Ansätze, von denen sich bisher keiner durchgesetzt hat. Eine Lösung namens *SweGrid Accounting System* (SGAS) kann in das Globus Toolkit integriert werden, ist aber bisher noch als Beta-Software gekennzeichnet.

4.1.3 Bedienung

Das Globus Toolkit besitzt vom WebMDS abgesehen keine Benutzeroberfläche, sondern wird ausschließlich per Kommandozeile bedient. Um einen Job abzusetzen wird die Job Spezifikation an einen GRAM Dienst abgeschickt. Dazu muss auf dem eigenen Rechner die Globus Client Software installiert sein. Vor dem Abschicken muss das Proxyzertifikat über einen Befehl generiert werden. Der Status des Auftrags kann dann über weitere Kommandozeilenbefehle überwacht werden.

Allerdings existiert mit der der Tomcat Anwendung *GridSphere* eine webbasierte Oberfläche, die allerdings nicht Teil des Globus Toolkits ist und zusätzlich installiert werden muss. Über diese Schnittstelle lassen sich bequem Jobs abschicken und überwachen und der Status eines Clusters einsehen.

Im Rahmen des Globus Toolkit Projekts wurde die OpenSSH Software modifiziert. *GSI-OpenSSH* erweitert die Software um die Sicherheitsschicht von Globus. So lassen sich sichere Terminalverbindungen aufbauen. Dafür muss der installierte SSH-Server, sowie das clientseitige SSH-Programm die Erweiterungen beinhalten.

Ein weiterer Weg auf die Globus Dienste zuzugreifen, ist eine der Programmierschnittstellen zu verwenden. Es existieren APIs für die Programmiersprachen C, Java sowie Python. So können Entwickler Programme entwickeln, die direkt mit den Web Services vom Globus Toolkit interagieren können.

4.1.4 Installation

Die Installation des Globus Toolkits gestaltet sich als verhältnismäßig komplex, da sie aus vielen Einzelkomponenten besteht. Die Software ist nicht als Komplettlösung zu sehen, sondern eher als eine Sammlung von zusammengehörenden Modulen, mit spezifischen Aufgaben. Daher muss vor der Installation bestimmt werden, welche der Module gebraucht werden und welche nicht. Dieser Ansatz ist jedoch sehr flexibel, da er durch die modulare Struktur alle Möglichkeiten der Erweiterung bietet. Funktionen, wie der Resource

Broker, eine graphische Benutzeroberfläche oder das Accounting sind nicht im Globus Toolkit enthalten. Leider existiert im Gegensatz zu gLite (siehe Abschnitt 4.3) nur ein großes Paket, welches alle Komponenten beinhaltet und manuell installiert werden muss.

4.1.5 Interoperabilität

Durch den starken Rückhalt im Open Grid Forum und die rasante Entwicklung gilt das Globus Toolkit gewissermaßen als die Referenzimplementation der OGSA. Da es bisher die Middleware ist, welche die Spezifikationen am besten umsetzt, werden sich andere Systeme an dieser Middleware orientieren und eine Konformität zur OGSA mit der Zusammenarbeit mit dem Globus Toolkit gleichsetzen.

4.2 UNICORE

Bereits 1997, noch bevor der Begriff des Grids geprägt war, wurde begonnen am *Uniform Interface to Computing Resources*, kurz UNICORE, zu arbeiten. Das Projekt wurde in zwei Phasen vom *Bundesministerium für Bildung und Forschung* gefördert. Die erste Phase lief von 1997 bis 1999, das Folgeprojekt namens UNICORE Plus wurde von 2000 bis 2002 gefördert [23].

Es sollte deutschen Supercomputern ermöglichen ihre Rechen- und Speicherkraft zu bündeln. Es wurde besonderes Augenmerk auf die Benutzerfreundlichkeit gelegt. Eine intuitive und leicht bedienbare, grafische Oberfläche mittels der Arbeitsaufträge abgesetzt werden können, sollte geschaffen werden. Forscher sollten nicht durch die Hürden der Bedienbarkeit von ihrer eigentlichen wissenschaftlichen Tätigkeit abgehalten werden [16].

UNICORE wird besonders in Deutschland, aber teilweise auch im europäischen Ausland, sowie im asiatischen Raum genutzt. Einige deutsche Einrichtungen setzen UNICORE auch parallel zu Globus Toolkit oder anderen Middleware-Systemen ein.

Abbildung 4.2 zeigt die drei Schichten aus der die UNICORE Software besteht. Die Benutzerschicht besteht aus dem UNICORE Client, der auf dem Benutzerrechner läuft. Mit *UNICORE site* (Usite) ist eine Einrichtung gemeint, wo ein UNICORE Gateway läuft. Er ist der Einstiegspunkt und leitet die Aufträge vom Benutzer weiter und versorgt diesen mit Statusinformationen. Innerhalb einer Usite können mehrere Virtual sites (Vsites) existieren. Sie bezeichnen ein Computercluster oder einen Supercomputer. Der *Network Job Superviser* verarbeitet die einkommenden Benutzeraufträge, stellt sicher dass die Benutzer die erforderlichen Rechte besitzen und leitet dann die Auf-

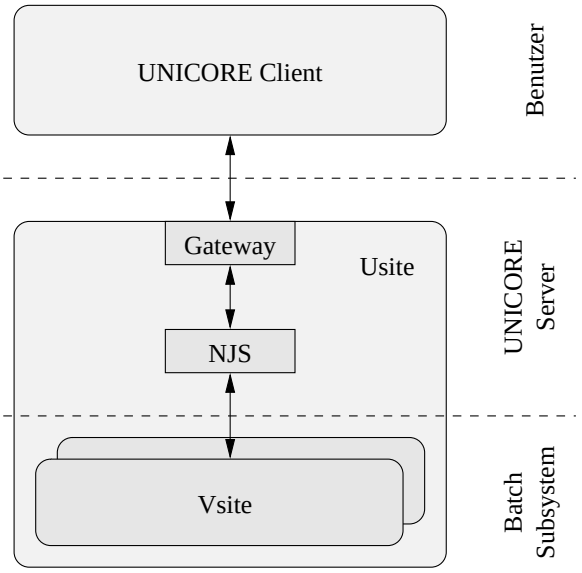

Abbildung 4.2: Benutzer-, Server- und Batch-System-Schicht der UNICORE Architektur

träge an die entsprechenden Vsites weiter [30]. Die Benutzer- und Serverschicht ist in Java implementiert. Die Anbindung an das lokale Batchsystem erfolgt über die Programmiersprache Perl.

4.2.1 Versionen

Im Jahr 2000 schlossen sich UNICORE-Nutzer und -Entwickler zum *UNICORE Forum e.V.* zusammen. Sie entschieden die Freigabe der Software unter einer Open-Source-Lizenz. Seitdem wird der Quellcode bei Sourceforge verwaltet. Auf der Liste der Mitglieder finden sich viele bekannte Namen, u.a. sind die Unternehmen EADS, Fujitsu Siemens, Hewlett Packard, IBM, Intel, Sun Microsystems sowie die Einrichtungen RWTH Aachen, Leibniz-Rechenzentrum, Universität Karlsruhe, Universität Paderborn und viele andere vertreten. Die erste einsatzfähige Version 3.5 war 2002 verfügbar. Die aktuelle Version von UNICORE für den Produktionseinsatz ist die Version 5. Sie beinhaltet jedoch noch keine Umstellung auf Web Services, so wie es die OGSA verlangt (siehe Abschnitt 3.2.1). Doch für die Zukunft ist eine Version UNICORE 6 angekündigt. Sie soll das Projekt um die nötigen Schnittstellen ausbauen und eine Zusammenarbeit mit OGSA-konformen Systemen, wie dem Globus Toolkit ermöglichen.

4.2.2 Funktionsumfang

Die Sicherheit basiert auf X.509 Zertifikaten und einer Public-key Infrastruktur. Die Benutzerzertifikate werden in einem *PKCS#12* Container gespeichert. Das ist dasselbe Format in dem Webbrowser persönliche Zertifikate ablegen. Bei der UNICORE-Architektur wird die komplette Benutzerinteraktion über eine einzige graphische Benutzerschnittstelle durchgeführt. Die Java-Anwendung entschlüsselt die Zertifikate nach der Eingabe des persönlichen Passworts und kann diese dann solange verwenden, wie das Programm nicht geschlossen wird. Dadurch wird ein Single-Sign-On realisiert. Allerdings ist eine echte Delegation nicht möglich, da keine Proxyzertifikate , wie in Abschnitt 2.3.1 beschrieben, zum Einsatz kommen.

Die Benutzer werden immer auf der ausführenden Seite auf lokale Systembenutzer abgebildet. Der Benutzer muss also immer auch als lokaler Benutzer auf den Systemen vorhanden sein.

Das UNICORE Gateway ist die Schnittstelle eines UNICORE Clusters zum Internet. Die Verbindung zum UNICORE Gateway, die i.d.R. über das unsichere Internet erfolgt, erledigt der *UNICORE Protocol Layer*, eine auf SSL aufbauende Protokollschicht.

Die Jobbeschreibungen werden bei UNICORE *Abstract Job Object* (AJO) genannt. Sie werden bequem über den UNICORE Client zusammengestellt und beinhalten Beschreibungen von rechenintensiven Aufträgen, sowie Datenübertragungen. Auch werden Unteraufgaben, bedingte Abarbeitung und Schleifen angeboten.

Die Ausführung von Jobs erfolgt in Kombination von drei Komponenten des UNICORE Systems. Der Nutzer schickt den Job über den UNICORE Client ab. Dieser nimmt Kontakt mit dem UNICORE Gateway auf und wird authentifiziert. Die Jobbeschreibung wird an den Network Job Superviser weitergeleitet. Er bildet das Bindeglied zum lokalen Batchsystem und übersetzt die AJOs in Aufträge für das jeweilige Batch-Subsystem.

Ein Resource Broker ist bisher nicht im Paket enthalten. Es existiert aber ein experimentelles Plug-In für UNICORE [5].

Daten werden über den *UNICORE Protocol Layer* kopiert. Letztendlich werden hier Objekte serialisiert. Es kann eine Datenübertragung zwischen dem UNICORE Client und dem Gateway oder direkt zwischen zwei UNICORE Clustern geschehen.

Der Status der verbundenen Cluster werden im UNICORE Client direkt angezeigt. Dabei stammen die Daten vom UNICORE Gateway. Es werden die aktuellen Jobs angezeigt, die der Nutzer auf dem jeweiligen Cluster abgeschickt hat. Wenn ein Job aus mehreren Unteraufgaben besteht, so werden diese durch einen Graphen angezeigt und farblich markiert. Ist eine Aufgabe

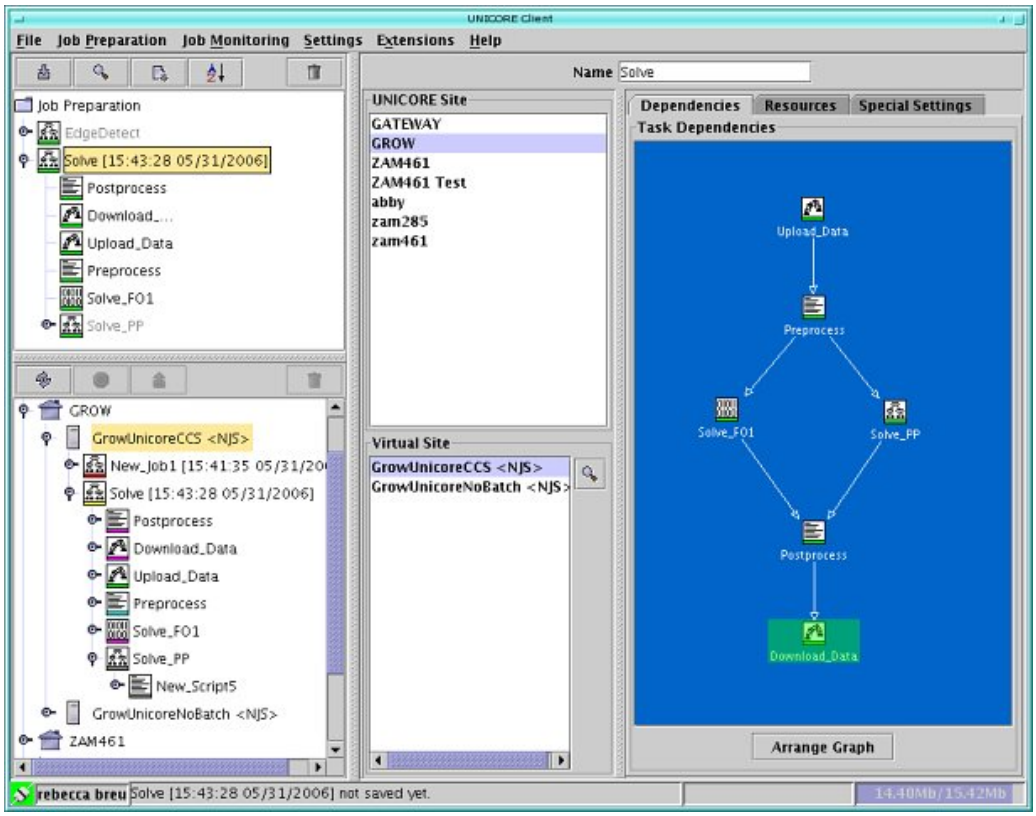

Abbildung 4.3: Überwachung von Jobs mit dem UNICORE Client

abgearbeitet, so kann die Ausgabe mit dem UNICORE Client angezeigt werden. Ansonsten existieren keine weiteren Informationsdienste im UNICORE Projekt.

Es werden keine Abrechnungsprotokolle im UNICORE Projekt definiert. Es existiert lediglich eine Zeichenkette bei der Jobdefinition, in der zukünftige Abrechnungsinformationen eingetragen werden können.

4.2.3 Bedienung

Mittels der java-basierten Benutzeroberfläche, dem UNICORE Client genannten Programm, lassen sich alle Benutzeraktionen durchführen. Erstmal muss das Zertifikat, welches von der Zertifizierungsstelle signiert wurde, importiert werden und alle Angaben zum Benutzer eingetragen werden. Nachdem die sogenannten Usites, also Server auf denen das UNICORE Gateway läuft, eingestellt wurden, kann eine erste Verbindung zu diesen aufgebaut werden. Mittels der Oberfläche lässt sich eine Job-Beschreibung erzeugen

und abschicken. Abbildung 4.3 zeigt einen Job, der aus mehreren Unterjobs besteht.

4.2.4 Installation

Das UNICORE Projekt wählte im Gegensatz zum Globus Toolkit ein anderes Softwaredesign. Es ist eher als „Komplettpaket“ zu sehen. Dies bringt den Vorteil mit sich, dass es relativ leicht zu installieren und benutzen ist. Allerdings schrumpft durch die eher monolithische Struktur auch die Flexibilität und Interoperabilität zu anderen Gridumgebungen. Das Benutzerprogramm lässt sich sehr einfach installieren, da es eine einzelne Java-Applikation ist.

4.2.5 Interoperabilität

Die Zusammenarbeit mit anderen Middleware-Systemen ist bisher kaum gegeben, da die UNICORE Software einen wenig modularen Ansatz wählt und die Protokolle gänzlich inkompatibel mit anderen Middlewares sind. Vor allem die eigenwillige und unflexible Sicherheitsschicht schließt eine Kommunikation mit anderen Systemen bisher aus. Erst die angekündigte Version 6 mit der erhofften OGSA Konformität kann hier eine Wende herbeiführen. Das Projekt unterstützt die OGSA bisher also nicht. Allerdings wurde im *GRid Interoperability Project* (GRIP) [29] versucht eine solche Konformität herzustellen. Das Projekt ist allerdings ausgelaufen und die OGSA Spezifikation hat sich mittlerweile weiterentwickelt.

4.3 gLite

Das Projekt *Enabling Grids for E-sciencE* (EGEE) ist ein von der Europäischen Union gefördertes Projekt zur Stärkung der wissenschaftlichen Nutzung von Grids in Europa. Zur Zeit sind etwa 70 Organisationen in 27 Ländern einbezogen [26]. Aus diesem Projekt ist die Middleware *gLite* entstanden. Sie wird vor allem im Hochenergiephysikbereich eingesetzt, z.B. am CERN in Genf. gLite versteht sich als leichtgewichtige Middleware, welche die besten Komponenten bestehender Systeme in sich vereint.

4.3.1 Versionen

Ursprünglich wurde 2003 die Grid-Middleware *Large Hadron Collider Computing Grid* (LCG) vom *European DataGrid* (EDG) entwickelt, welche die

riesigen Datenmengen des großen Teilchenbeschleuniger am CERN verwalten sollte. Diese Middleware baut weitgehend auf dem Globus Toolkit auf. Parallel dazu ist ab 2005 die gLite Middleware entwickelt worden. In der Version 3.0, die seit 2006 verfügbar ist, wurden Teile der LCG Software, gLite 2 sowie anderen Lösungen, wie Condor und AliEn zu einer neuen Software zusammengesetzt. Sie sollte die besten Ansätze der verschiedenen Lösungen in sich vereinen.

4.3.2 Funktionsumfang

Die Basis der Sicherheitsschicht wurde dem Globus Toolkit entliehen. Sie beherrscht also auch Proxyzertifikate, Single-Sign-On sowie Delegation. Außerdem kommt der in Abschnitt 4.1 erläuterte Ablageserver für Zertifikate namens MyProxy zum Einsatz. Die Benutzerzertifikate werden also nicht bei den Anwendern gespeichert, sondern von einem zentralen Server ausgegeben. Der Virtual Organization Membership Service (VOMS) [6] ist ein Dienst, der Zugehörigkeiten zu Virtuellen Organisationen und Gruppen zentral speichert. So können Rollen organisationsübergreifend modelliert werden. Die Rechte, die einzelnen Rollen letztendlich gewährt werden, sind immer abhängig von der lokalen Einrichtung und können nicht durch VOMS erteilt werden. Dadurch liegt die Kontrolle der Privilegien immer bei den lokalen Systemadministratoren einer Organisation.

Durch Isolation werden die Privilegien der Benutzerjobs so eingeschränkt, dass sie ihre Aufgaben erfüllen können, allerdings ohne Zugriff auf weitere Ressourcen zu bekommen.

Das *Workload Management System* kümmert sich um die Abarbeitung der Jobs und eine effiziente Aufteilung auf die vorhandenen Ressourcen. Der *Workload Manager* nimmt den Job vom Benutzer entgegen und entscheidet an welches *Compute Element* er weitergeleitet werden soll. Dabei trifft der Workload Manager die Entscheidung aufgrund von Kriterien, die in der Jobbeschreibung definiert wurden. Er nimmt also die Rolle eines Resource Brokers ein. Die Compute Elements selber sind entweder Cluster oder Supercomputer, welche die eigentliche Rechenkraft besitzen. Dort läuft ein Batch-System, wie PBS oder Condor, dass die Jobs annimmt.

In der gLite Software kommt eine Beschreibungssprache für Jobs namens *Job Description Language* zum Einsatz, die ihre Wurzeln im Condor System hat. Mit ihrer Hilfe lassen sich komplexe Anforderungen ausdrücken, wie z.B. das Vorhandensein bestimmter Anwendungssoftware, Anzahl an Prozessoren, Hauptspeicher usw. Außerdem kann das Kopieren von Daten aus einem Speicher angefordert und Abhängigkeiten in einem azyklischen gerichteten Graphen ausgedrückt werden. Darüber hinaus werden neben der Stapelverarbei-

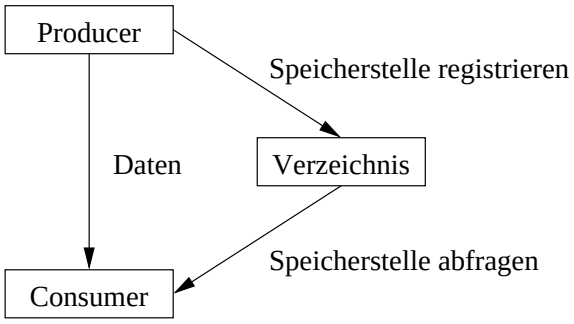

Abbildung 4.4: R-GMA

tung auch interaktive und MPI-basierte Jobs unterstützt. Aufgaben können zerteilt werden, so dass sie per Checkpointing auf andere Rechner verschoben oder angehalten und wieder aufgenommen werden können.

Das Datenmanagement ist besonders ausgeklügelt, da die Middleware konzipiert wurde, um die großen Datenaufkommen des Large Hadron Collider am CERN zu bewältigen. Alle Benutzer im Grid können Dateien aufgrund eines logischen Dateinamens identifizieren, unabhängig davon, wo sich die Datei geographisch befindet. Dabei gibt es ein virtuelles Dateisystem inklusive Unterordner. Dies macht es dem Anwender einfach auf Daten zuzugreifen. Eine logische Datei kann in der Praxis an mehreren physikalischen Speicherorten gespiegelt sein. Für das Ablegen physikalischer Dateien können mehrere Systeme zum Einsatz kommen. Sie müssen die Spezifikationen eines *Storage Resource Managers* (SRM) erfüllen, eine Abstraktionsschicht um den einheitlichen Zugriff auf heterogene Speichersysteme zu ermöglichen. Der *Disk Pool Manager* ist die leichtgewichtigste Variante. Er organisiert die Dateien in einer oder mehreren Festplatten und speichert Metainformationen in einer relationalen Datenbank. Neben SRM bietet er eine GridFTP und eine Remote File I/O (RFIO) Schnittstelle an. RFIO ist ein einfaches Netzwerkprotokoll für die Dateiübertragung, welches am CERN entwickelt wurde. *Castor* [11] ist ein am CERN entwickeltes System, dass für die Archivierung von großen Datenmengen auf Bandlaufwerken bestimmt ist. Es ist ein verhältnismäßig riesiges und schwergewichtiges System. *dCache* [15] ist für kleine bis große Systeme ausgelegt und wurde am DESY in Hamburg entwickelt. Um die logischen Dateinamen mit den wirklichen Daten zu verknüpfen kommt ein Dateikatalog zum Einsatz. Dies ist ein auf einer relationalen Datenbank aufbauender Dienst, der die Anfragen nach logischen Dateinamen transparent auflöst.

Das Informations- und Monitorsystem von gLite baut auf *R-GMA* [12]

auf. Dies ist ein Consumer/Producer-Modell basierend auf der Abfragesprache für relationale Datenbanksysteme SQL. Jede Virtuelle Organisation besitzt eine virtuelle relationale Datenbank. Es besteht aus drei Arten von Komponenten: dem Producer, dem Consumer und einem Verzeichnis. Producer wird jede Art von Informationsquelle genannt. Dies können beispielsweise die Statusinformationen eines Batch-Systems oder die momentane Speicherauslastung eines Massenspeichers sein. Die Informationsquellen registrieren sich bei einem Verzeichnis. Dabei benutzen sie einen CREATE-TABLE-Befehl. Consumer können Benutzerclients, aber auch automatisierte Dienste, wie ein Resource Broker sein. Der Consumer fragt das Verzeichnis ab und kann dadurch feststellen, welche Informationen vorhanden sind und welcher Producer diese bereitstellt. Die Abfrage geschieht durch den SQL-Befehl SELECT. Abbildung 4.4 zeigt den Zusammenhang zwischen den beteiligten Komponenten. Der Verzeichnisdienst ist durch einen LDAP-Server realisiert. Die gLite Software enthält außerdem das schon im Abschnitt 4.1.2 vorgestellte Monitoring and Discovery System des Globus Toolkit.

Das *Distributed Grid Accounting System* [8] ist eine Abrechnungslösung für die gLite Middleware. In jeder Virtuellen Organisation gibt es normalerweise ein *Home Location Register* (HLR). Es funktioniert wie eine Bank. Dort existiert für jeden Benutzer und für jede Ressource der Virtuellen Organisation ein Konto. In diesem werden virtuelle Währungseinheiten verwaltet. Sie werden *GridCredits* genannt. Allerdings ist keine Umrechnung in reale Währungen vorgesehen. Das System dient also nur zur Optimierung der Ressourcenauslastung und nicht zum finanziellen Ausgleich [17]. Weiterhin existieren *Price Authorities*. Sie geben Auskunft über die Nutzungskosten von Ressourcen in einer Virtuellen Organisation. Schickt ein Benutzer nun einen Auftrag ab, so wird er vom Workload Management System angenommen und der Ressource Broker ermittelt anhand von Anforderungen, die in der Jobbeschreibung angegeben sind, das am besten passende Compute Element. Dafür wird für jeden Kandidaten eine Kostenprognose auf Grundlage der Informationen aus der Jobbeschreibung und der Price Authority erstellt. Nach Beendigung des Auftrags werden die exakten Kosten berechnet und vom Auftraggeber abgebucht.

4.3.3 Bedienung

Die Bedienung der gLite Software wird per Kommandozeile durchgeführt. Es gibt eine Vielzahl verschiedener Kommandos. Beim Abschicken eines Jobs kann dieser durch die Beschreibungssprache JDL (siehe Abschnitt 4.3.2) genau spezifiziert werden. Dabei sind die wichtigsten Informationen der auszuführende Befehl und die Ein- und Ausgabedaten. Die auszuführende Datei

ist meistens ein Shell-, Python- oder Perlskript. Die Eingabedaten kommen vom eigenen PC oder können von einer anderen Stelle geholt werden. Die Ausgabedaten können nach der Beendigung des Jobs heruntergeladen werden.

Allerdings haben einzelne Grids auch webbasierte Lösungen entwickelt, mit denen sich die Hauptaufgaben im Grid, wie Überwachen des Clusters, Abschicken und Kontrollieren von Jobs usw. durchführen lassen können. Das italienische *National Institute of Nuclear Physics* (INFN) besitzt mit dem Projekt *Grid Infn Laboratory for Dissemination Activities* (GILDA) [3] eine ausgefeilte Oberfläche zum Kontrollieren der beteiligten Ressourcen im Grid. Auch an der Universität Wuppertal gibt es Bestrebungen den Zugang zum Grid durch ein Webfrontend zu erleichtern [4].

Außerdem existieren Schnittstellen für die Programmiersprachen C, Java, Perl und Python.

4.3.4 Installation

Die gLite Software wird über die Debian-Paketverwaltung APT installiert. Auf RedHat Systemen, z.B. Scientific Linux, wo gLite normalerweise zum Einsatz kommt, ist APT nicht installiert. Allerdings lässt es sich schnell mit einem RPM Paket nachrüsten. Im Gegensatz zum Globus Toolkit besteht die Installation aus einer Vielzahl von Paketen, welche die einzelnen Programmteile von gLite beinhalten. So lassen sich beispielsweise die Benutzerprogramme einzeln installieren. Die Konfiguration verläuft ähnlich wie beim Globus Toolkit. Es muss ein lokales Batch-System ausgewählt, die entsprechenden Zertifikate installiert und die Datenablage konfiguriert werden. Dabei existieren eine Vielzahl von Varianten, da es allein für die Datenspeicherung drei verschiedene Lösungen gibt.

4.3.5 Interoperabilität

Die gLite Software wird hauptsächlich im EGEE Projekt verwendet. Daher ist vor allem die Zusammenarbeit mit allen im Projekt beteiligten Softwarekomponenten wichtig. Dies wird erreicht, indem projektweite Standards eingehalten werden. Beispielsweise halten Komponenten zur Datenverwaltung die Spezifikationen eines Storage Resource Manager ein. So ist der Zugriff auf entfernte Dateien immer gleich, unabhängig davon, welches System wirklich dahinter steckt.

Teilweise werden auch die Bestimmungen der OGSA eingehalten. Doch im Gegensatz zum Globus Toolkit wird hier im Hinblick auf die Zusammenarbeit innerhalb des Projekts eine sanfte Umstellung auf Web Services

durchgeführt. Werden neue Protokolle und Schnittstellen entworfen, so werden diese unter Berücksichtigung der OGSA entworfen, doch die bereits im Einsatz befindlichen und ausgiebig getesteten Komponenten laufen weiter, auch ohne OGSA-Konformität.

Kapitel 5

Installation am Rechenzentrum der Universität Kassel

5.1 Derzeitige Situation an der Universität Kassel

Momentan gibt es zwei Rechencluster am Hochschulrechenzentrum der Universität Kassel. Auf dem neueren Cluster läuft das Betriebssystem *RedHat Enterprise Linux 4 AS 64 Bit*. Er besteht aus 57 Rechenknoten, von denen die meisten AMD Opteron 248 Doppelprozessorsysteme mit vier GB Hauptspeicher sind. Außerdem gibt es einige Rechner mit acht sowie 16 GB Hauptspeicher. Auf dem Cluster ist das Batchsystem *OpenPBS* installiert.

Außerdem existiert noch ein älteres Cluster. Es besteht aus drei Rechnern mit jeweils 4 PowerPC-Prozessoren mit 1000 MHz. Sie verfügen über 2 bzw. 16 GB Hauptspeicher. Es läuft das AIX Betriebssystem und das Batchsystem *LoadLeveler* von IBM.

Wer ein Benutzerkonto des Hochschulrechenzentrums besitzt, kann Rechendienste in Anspruch nehmen. Das Cluster steht allen Studenten und Mitarbeitern der Hochschule zur Verfügung. Der Zugang erfolgt per Secure Shell (SSH). Um auf einem Cluster Jobs auszuführen, zu überwachen und die Ergebnisse abzufragen, müssen spezielle Kommandos erlernt werden. Diese unterscheiden sich je nachdem, welches Cluster benutzt wird.

Das D-Grid Projekt [1] ist ein nationales Förderprogramm zur Unterstützung der Grid-Infrastruktur in Deutschland. Vor allem Universitäten nehmen teil, um sich zu einem großen Rechenverbund zusammenzuschließen. Den inneren Kern dieses Projekts bilden namhafte Einrichtungen, wie das Forschungszentrum Jülich, die Max-Planck-Gesellschaft uvm. Doch nach und nach sollen weitere Universitäten und Institute angeschlossen werden. Um Er-

fahrungen in diesem Bereich zu sammeln und so zukünftige Bemühungen zu unterstützen, eine Integration in vorhandene Grid-Strukturen durchzuführen, wurde eine Grid-Middleware auf den beiden Clustern des Rechenzentrums installiert und getestet.

5.2 Durchführung

5.2.1 Auswahl der Middleware

Nach einer eingehenden Recherche wurde die aktuelle Version der im Abschnitt 4.1 vorgestellten Middleware Globus Toolkit für die Installation ausgewählt. Für diese Entscheidung waren verschiedene Punkte maßgeblich. Vor allem die Interoperabilität zu anderen Systemen der Software war ausschlaggebend. Das Globus Toolkit besitzt die beste Unterstützung für die OGSA und die Sicherheitsschicht ist ausgelegt, um mit anderen Systemen zusammenzuarbeiten. UNICORE hat diesbezüglich Schwächen, da es in der aktuellen Version eigene Konzepte für eine Sicherheitsschicht implementiert hat, die sich sehr schlecht mit anderen Middlewares vertragen [14]. Die Sicherheitsschicht der gLite Middleware baut zwar auf dem Globus Toolkit auf, doch ist das Produkt insgesamt sehr stark auf die Arbeit im Umfeld der Hochenergiephysik ausgerichtet. Auch ist die Unterstützung für die OGSA noch nicht weit fortgeschritten.

5.2.2 Rechnerzertifikate

Damit die Sicherheitsschicht des Globus Toolkit einwandfrei funktioniert und die Anbindung an weitere Cluster möglich ist, musste ein Rechnerzertifikat auf jedem Cluster abgelegt werden. Dieses Zertifikat wurde von einer zentralen Stelle signiert (siehe Abschnitt 3.3). Dafür wurde ein entsprechender Antrag ausgefüllt und bei der Registry Authority des Rechenzentrums vorgelegt. Sie leitete den Antrag an die Zertifizierungsstelle des DFN weiter, wo er signiert und per E-Mail zurückgeschickt wurde. Dieser Vorgang muss für jeden Rechner und jeden Benutzer durchgeführt werden, der die Grid-Middleware nutzen möchte. Insgesamt mussten zwei Rechnerzertifikate beantragt werden, die innerhalb weniger Tage zur Verfügung standen.

5.2.3 Installation der Middleware

Im Vorfeld der eigentlichen Installation wurde eine Testinstallation vom Globus Toolkit auf zwei abgesonderten Maschinen des Linux Clusters durch-

geführt. Dadurch konnten Erfahrungen gesammelt werden, ohne den Produktionsbetrieb des Clusters zu stören.

Für den Betrieb des Globus Toolkits wurde ein Benutzerkonto `globus` angelegt, der rechenzentrumsweit existiert. Die Benutzerrechte der Prozesse sowie die Dateiberechtigungen laufen hauptsächlich mit diesem Benutzerkonto. In der folgenden Installationsbeschreibung werden Befehle auf der Kommandozeile in der folgenden Form angegeben:

```
(root) mkdir /opt/globus-4.0.3
```

Hier ist der Benutzer in Klammern vor dem eigentlichen Befehl angegeben. In diesem Beispiel ist es der Benutzer `root`. Danach kommt der eigentliche Befehl inklusive Parameter. Die Installation wurde mit Hilfe der Dokumentation des Globus Toolkit angefertigt [2].

Linux Cluster

Für die Installation wurden ein bis zwei GB Speicher zeitweise benötigt. Die endgültige Installation nahm etwa 150 MB in Anspruch. Auf dem Hauptrechner des Linux Clusters existierten genügend Festplattenspeicher um das Globus Toolkit zu installieren. Anfangs wurde das entsprechende Verzeichnis erzeugt und dem Benutzer `globus` übertragen. Er sollte später Besitzer der Globus Software sein.

```
(root) mkdir /opt/globus-4.0.3
(root) chown globus.globus /opt/globus-4.0.3
```

Folgende Befehle mussten an die systemweite Konfigurationsdatei `/etc/-profile` angehängt werden:

```
export JAVA_HOME=/usr/java/j2sdk1.4.2_09
export ANT_HOME=/opt/ant
export GLOBUS_LOCATION=/opt/globus-4.0.3
sh $GLOBUS_LOCATION/etc/globus-user-env.sh
export PATH=$PATH:$GLOBUS_LOCATION/bin:$GLOBUS_LOCATION/sbin:\
$GLOBUS_LOCATION/etc
export LD_LIBRARY_PATH=$GLOBUS_LOCATION/lib
export PBS_HOME=/var/spool/PBS
export GLOBUS_CERT_DIR=/etc/grid-security
```

Diese Angaben sind Umgebungsvariablen, die vom Globus Toolkit ausgelesen werden und vorher unbedingt gesetzt werden müssen. Sie stellen die

Pfade ein, die von System zu System unterschiedlich sein können. Da die Variablen systemweit eingetragen wurden, müssen sie nicht vom Benutzer jedes Mal per Hand eingestellt werden.

Damit auch alle Benutzer und Rechner mit der Globus Installation kommunizieren können, müssen die entsprechenden Zertifikate vorhanden und als vertrauenswürdig eingestuft sein. Dies wurde erreicht, indem die entsprechenden Zertifikate im Verzeichnis `/etc/grid-security` abgelegt wurden. Für diesen Zweck habe ich ein Archiv erzeugt, welches bereits die entsprechenden Zertifikate beinhaltet. Die importierten Zertifikate sind die der GridKA CA, also der Zertifizierungsstelle des Forschungszentrums Karlsruhe und des Deutschen Forschungsnetzes. Außerdem beinhaltet das Archiv noch Konfigurationsdateien für OpenSSL, die definieren, wie Zertifikatsanträge generiert werden müssen. Das hat den Sinn, dass Benutzer sich mit einem einzigen Befehl gültige Anträge generieren können, die dann auch automatisch per E-Mail an die richtige Stelle geschickt werden.

```
(root) source /etc/profile
(root) mkdir $GLOBUS_CERT_DIR
(root) cd $GLOBUS_CERT_DIR
(root) scp USER@HOST:path/to/certificates_uniks.tgz .
(root) tar xzvf certificates_uniks.tgz
(root) ln -s certificates/globus-host-ssl.conf.fe102e03 \
       globus-host-ssl.conf
(root) ln -s certificates/globus-user-ssl.conf.34f8e29c \
       globus-user-ssl.conf
(root) ln -s certificates/grid-security.conf.fe102e03 \
       grid-security.conf
```

Der erste Befehl war nur nötig, damit auch wirklich sichergestellt ist, dass die zuvor gemachten Änderungen an der Datei `/etc/profile` ausgeführt wurden. Danach wurde das Archiv `certificates_uniks.tgz` auf die Maschine kopiert und ausgepackt. Es enthält die oben angegebenen Konfigurationsdateien und Zertifikate. Zum Schluss wurden noch symbolische Links zu den richtigen Konfigurationsdateien erstellt.

Nun konnte begonnen werden, die Software zu kompilieren und installieren. Im Installationsverzeichnis vom Globus Toolkit `/opt/globus-4.0.3/` wurde das aktuelle Installationsarchiv heruntergeladen und entpackt.

```
(globus) cd /opt/globus-4.0.3/
(globus) wget http://www-unix.globus.org/ftppub/gt4/4.0/4.0.3/\
         installers/src/gt4.0.3-all-source-installer.tar.bz2
(globus) tar xjvf gt4.0.3-all-source-installer.tar.bz2
```

Im daraufhin entstandenen Verzeichnis `gt4.0.3-all-source-installer` habe ich die Kommandos zum Kompilieren und Installieren ausgeführt. Das Kommando `configure` überprüft die benötigten Bibliotheken und erzeugt ein Makefile. Dabei bezeichnet der Parameter `--prefix=$GLOBUS_LOCATION` das Installationsverzeichnis und `--enable-wsgram-pbs` sorgt dafür, dass die Komponente für die Anbindung an das lokale Batchsystem OpenPBS erstellt wird. Der Befehl `make` kompiliert das Globus Toolkit. Dieser Vorgang dauerte mehrere Stunden. Die Angaben `2>&1 | tee build.log` sorgten dafür, dass das Protokoll des Vorgangs abgespeichert wurde und nicht verloren ging. Der letzte Befehl installierte die fertig kompilierte Software schließlich in das gewünschte Verzeichnis.

```
(globus) cd gt4.0.3-all-source-installer
(globus) ./configure --prefix=$GLOBUS_LOCATION \
         --enable-wsgram-pbs
(globus) make 2>&1 | tee build.log
(globus) make install
```

Damit das Globus Toolkit die Jobs an das lokale Batchsystem weiterreicht, musste ein symbolischer Link richtig gesetzt werden.

```
(globus) rm /opt/globus-4.0.3/etc/grid-services/jobmanager
(globus) ln -s \
         /opt/globus-4.0.3/etc/grid-services/jobmanager-pbs \
         /opt/globus-4.0.3/etc/grid-services/jobmanager
```

Die Authentifizierung des Clusters geschieht durch das signierte Rechnerzertifikat und den zugehörigen privaten Schlüssel. Beides musste an die richtige Stelle kopiert werden. Außerdem mussten die Dateirechte angepasst werden. Auf den privaten Schlüssel hat nur der Eigentümer Leserechte, da er geheim gehalten werden muss.

```
(root) cd $GLOBUS_CERT_DIR
(root) scp USER@HOST:path/to/hostkey.pem .
(root) chmod 400 hostkey.pem
(root) scp USER@HOST:path/to/cert-154965601.pem hostcert.pem
(root) chmod 644 hostcert.pem
```

Nun mussten die Schlüssel noch einmal kopiert werden. Die Web Services Komponenten vom Globus Toolkit erwarten, dass die Schlüssel unter einem anderen Namen abgelegt sind. Außerdem laufen diese Komponenten unter dem Benutzer `globus`. Daher werden die Schlüssel diesem Benutzer übertragen.

```
(root) cp hostkey.pem containerkey.pem
(root) cp hostcert.pem containercert.pem
(root) chown globus.globus containerkey.pem containercert.pem
```

In den Zertifikaten der Benutzer des Globus Toolkits stehen X.500 Namen wie in Abschnitt 3.2.1 beschrieben. Auf dem System werden Berechtigungen über Unix-Benutzerkonten vergeben. Daher müssen die Namen aus den Zertifikaten auf die Benutzerkonten abgebildet werden. Der folgende Befehl erzeugt einen entsprechenden Eintrag in der Konfigurationsdatei für das Unix-Benutzerkonto `user`.

```
(root) grid-mapfile-add-entry -dn \
       "/C=DE/O=GridGermany/OU=Universitaet Kassel/\
       OU=Hochschulrechenzentrum/CN=Max Mustermann" \
       -ln user
```

Der genaue X.500 Name kann durch das Zertifikat des Benutzers herausgefunden werden. Die Ausgabe des folgenden Kommandos ist der Name, auf den das Zertifikat ausgestellt ist.

```
(user) grid-cert-info -subject
```

Nun war es bereits möglich als Benutzer ein Proxyzertifikat anzufordern (siehe Abschnitt 2.3.1). Der folgende Befehl schließt mit der Ausgabe `VERIFY OK` erfolgreich ab.

```
(user) grid-proxy-init -verify -debug
```

Ein wichtiger Dienst zur Datenübertragung ist GridFTP (siehe Abschnitt 4.1.2). Als erstes wurde ein Verzeichnis für die Protokolldateien erstellt. Das zweite Kommando startet den Dienst auf dem dafür vorgesehenen Port 2811.

```
(root) mkdir -p /var/log/gridftp/
(root) globus-gridftp-server -p 2811 -S -L /var/log/gridftp/
```

Nachdem der GridFTP Server gestartet wurde, ist es möglich testweise eine Datei zu kopieren.

```
(user) globus-url-copy -vb \
       file:///home/hrz/user/testdatei \
       gsiftp://host.de/tmp/testdatei
```

Der Reliable File Transfer (RFT) ermöglicht die sichere Speicherung von Übertragungszuständen (siehe auch Abschnitt 4.1.2). Er benötigt zum Speichern der Zustände eine SQL Datenbank. *PostgreSQL* konnte bequem über die Paketverwaltung von RedHat Linux installiert werden.

```
(root) cd /redhat/rhel4-as-x86_64-u1/RedHat/RPMS
(root) rpm -U postgresql-7.4.7-2.RHEL4.1.x86_64.rpm \
(root) postgresql-libs-7.4.7-2.RHEL4.1.x86_64.rpm \
(root) postgresql-server-7.4.7-2.RHEL4.1.x86_64.rpm
```

Nun musste die Datenbank noch konfiguriert werden. Im Verzeichnis `/var/lib/pgsql` sollten die Daten liegen. Die ersten drei Kommandos legen das Verzeichnis an, übertragen die Benutzerrechte dem `postgres` Benutzer und initialisieren die Datenbank. Das vierte Kommando startet den Datenbankserver und das letzte erstellt einen SQL Benutzer `globus`.

```
(root) mkdir /var/lib/pgsql
(root) chown postgres.postgres /var/lib/pgsql/
(postgres) /usr/bin/initdb -D /var/lib/pgsql/data/
(postgres) /usr/bin/pg_ctl start -D /var/lib/pgsql/data/ \
           -l postmaster.log -o "-i"
(postgres) /usr/bin/createuser globus
```

Sicherheitshalber wird der Zugriff von fremden IP-Adressen ausgeschlossen. Auch darf nur der Benutzer `globus` auf die Datenbank zugreifen. Dafür wird in die Datei `/var/lib/pgsql/data/pg_hba.conf` folgende Zeile eingetragen:

```
host rftDatabase globus 141.51.13.86 255.255.255.255 trust
```

Der Benutzer `globus` kann nun in der bereits initialisierten Datenbank das Datenbankschema ablegen. Dieses wurde von der Globus Toolkit Installation mitgeliefert.

```
(globus) createdb rftDatabase
(globus) psql -d rftDatabase -f \
         /opt/globus-4.0.3/share/globus_wsrf_rft/rft_schema.sql
```

Damit der RFT Dienst die Datenbank ansprechen kann, muss der Rechnername, der Benutzername sowie das Passwort in die Konfigurationsdatei `/opt/globus-4.0.3/etc/globus_wsrf_rft/jndi-config.xml` eingetragen werden.

Um RFT manuell zu verwenden muss eine spezielle Konfigurationsdatei geschrieben werden, in der alle nötigen Informationen stehen. Unter `/opt/globus-4.0.3/share/globus_wsrf_rft_test/transfer.xfr` ist eine Beispieldatei zu finden, in der nur noch der Rechnername angepasst werden muss. Dazu wurde die Datei nach `/tmp/rft.xfr` kopiert und alle Vorkommnisse von `localhost` in der Datei durch den entsprechenden Namen ersetzt. Der erste Befehl kopiert die Konfigurationsdatei für die Übertragung, der zweite legt eine Testdatei an, die kopiert werden soll und der letzte Befehl führt die eigentliche Kopieraktion durch. Nach dem erfolgreichen Abschluss der Kopie existiert eine Datei mit dem Namen `/tmp/rftTest_Done.tmp` mit demselben Inhalt wie die Quelldatei.

```
(user) cp \
       /opt/globus-4.0.3/share/globus_wsrf_rft_test/\
       transfer.xfr /tmp/rft.xfr
(user) echo Teststring123 >/tmp/rftTest.tmp
(user) rft -h host.de -f /tmp/rft.xfr
```

Um interaktive Benutzersitzungen zu ermöglichen muss ein SSH-Server mit Unterstützung für die Globus Sicherheitsschicht gestartet werden. Er kann mit dem folgenden Kommando auf dem Port 2222 gestartet werden. Bei Bedarf kann auch der alte SSH-Server vollständig ersetzt werden, da der mitgelieferte SSH-Server auch die alte Funktionalität bietet. Die Verbindung von einem Client Rechner erfolgt, wie mit einem normalen SSH Programm.

```
(root) /opt/globus-4.0.3/sbin/sshd -p 2222
```

Um die Web Service Komponenten vom Globus Toolkit, den sog. Container zu starten und zu stoppen werden folgende beiden Befehle benutzt. Er arbeitet dann als Hintergrunddienst.

```
(root) globus-start-container-detached
(root) globus-stop-container-detached
```

Außerdem kann auch die Pre-WS-Version des Gatekeeper gestartet werden.

```
(root) globus-gatekeeper \
       -conf /opt/globus-4.0.3/etc/globus-gatekeeper.conf
```

AIX Cluster

Die Installation des Globus Toolkits auf dem AIX Cluster gestaltete sich schwieriger, da benötigte Software erst installiert werden musste. Die GNU-Versionen von make, tar und sed sowie eine aktuelle Version von Ant wurden installiert. Ansonsten verlief die Installation ähnlich, wie beim Linux Cluster auch. Allerdings mussten an vielen Stellen Besonderheiten beachtet werden. Beispielsweise musste das Globus Toolkit mit dem Parameter `--with-flavor=vendorcc32` installiert werden. Für RFT wurde nicht erneut ein Datenbankserver aufgesetzt, sondern die bereits vorhandene Installation des Linux Clusters verwendet. Dafür wurde eine neue Datenbank angelegt und auch das Schema eingespielt. Außerdem wurden die Berechtigungen für die Datenbank so erweitert, dass der AIX Cluster Zugriff erhält.

Client Rechner

Damit Benutzer das Globus Toolkit verwenden können, müssen sie die Client-Programme auf ihrem Rechner installiert haben. Das umfasst vor allem Programme zum Abschicken und Überwachen von Jobs sowie die Globus FTP und SSH Client-Programme. Ein komfortabler Weg, die Programme auf vielen Rechnern zu installieren, besteht darin ein Debian-Paket zu erstellen. Dazu wird das Globus Toolkit wie gewohnt heruntergeladen und in einem Verzeichnis entpackt. Der Unterschied liegt nun darin, dass durch den Aufruf des Programms `checkinstall` ein entsprechendes Debian-Paket erzeugt wird, dass sich einfach auf anderen Rechnern installieren lässt. Durch die Parameter `wsgram gridftp gsi-openssh` und `install` werden nur die Client-Programme des Globus Toolkits erzeugt. Das Programm `checkinstall` besitzt eine interaktive Menüführung. Es lassen sich neben Debian-Paketen auch RPM-Pakete erstellen. Außerdem muss ein Titel, eine Beschreibung und eine Versionsnummer angegeben werden.

```
(user) ./configure
(user) checkinstall make wsgram gridftp gsi-openssh install
```

Wie im Abschnitt *Linux Cluster* schon beschrieben, müssen die Zertifikate des DFN importiert und die Umgebungsvariablen in die Datei `/etc/profile` eingetragen werden.

Benutzerzertifikate werden durch dieselbe Prozedur wie Rechnerzertifikate erlangt (siehe 5.2.2). Der persönliche Schlüssel und das Zertifikat des Benutzers werden an die richtige Stelle im Home-Verzeichnis kopiert. Es werden Berechtigungen gesetzt, die das Ausspähen des eigenen Schlüssels verhindern.

```
(user) mkdir ~/.globus
(user) scp host:path/to/usercert.pem ~/.globus/
(user) scp host:path/to/userkey.pem ~/.globus/
(user) chmod 644 ~/.globus/usercert.pem
(user) chmod 600 ~/.globus/userkey.pem
```

5.2.4 Testprogramm

Das Testprogramm zeigt vor allem, dass die Hauptaufgaben der Grid-Middleware funktionieren: die Sicherheitsschicht, das Ausführen von Programmen und die korrekte Verteilung der Daten. Der Job wird mit Hilfe einer XML-Datei beschrieben (siehe weiter unten). So kann angegeben werden, welche Datei ausgeführt werden soll, und welche Parameter übergeben werden (`executable`, `argument`). Im Beispiel wird ein Perlskript namens `process_data.pl` angegeben, welches noch besprochen wird. Die Ausgaben des Programms können in Dateien umgeleitet werden, die sich der Benutzer nach der Ausführung anschauen kann (`stdout`, `stderr`). Unter `fileStageIn` und `fileStageOut` kann festgelegt werden, welche Dateien vor bzw. nach der Ausführung des Programms kopiert werden sollen. Dabei kann eine Kopie zwischen dem lokalen PC, dem Cluster oder anderen Stellen beliebig erfolgen. Im Beispiel wird zu Beginn eine Datei mit Eingangsdaten namens `in.data` und das auszuführende Perlskript selbst vom Rechner des Benutzers zum Cluster kopiert und nach Beendigung des Programms die Ergebnisdatei `out.data` wieder zurückkopiert. Mittels `fileCleanUp` können noch etwaige temporäre Dateien nach der Programmausführung gelöscht werden. Es sollen die Eingangs- und Ausgangsdatei sowie das Perlskript gelöscht werden, so dass keine Spuren von dem Auftrag zurückbleiben.

```
<job>
  <executable>
    ${GLOBUS_USER_HOME}/work/process_data.pl
  </executable>
  <argument>${GLOBUS_USER_HOME}/work/in.data</argument>
  <argument>${GLOBUS_USER_HOME}/work/out.data</argument>
  <stdout>${GLOBUS_USER_HOME}/grid/gt_test_job/stdout</stdout>
  <stderr>${GLOBUS_USER_HOME}/grid/gt_test_job/stderr</stderr>
  <fileStageIn>
    <transfer>
      <sourceUrl>
        gsiftp://host.de:2811/home/hrz/user/grid/gt_test_job/in
.data
```

```
      </sourceUrl>
      <destinationUrl>
        file:///${GLOBUS_USER_HOME}/work/in.data
      </destinationUrl>
    </transfer>
    <transfer>
      <sourceUrl>
        gsiftp://host.de:2811/home/hrz/user/grid/gt_test_job/pr
ocess_data.pl
      </sourceUrl>
      <destinationUrl>
        file:///${GLOBUS_USER_HOME}/work/process_data.pl
      </destinationUrl>
    </transfer>
  </fileStageIn>
  <fileStageOut>
    <transfer>
      <sourceUrl>
        file:///${GLOBUS_USER_HOME}/work/out.data
      </sourceUrl>
      <destinationUrl>
        gsiftp://host.de:2811/home/hrz/user/grid/gt_test_job/ou
t.data
      </destinationUrl>
    </transfer>
  </fileStageOut>
  <fileCleanUp>
    <deletion>
      <file>file:///${GLOBUS_USER_HOME}/work/out.data</file>
    </deletion>
     <deletion>
      <file>file:///${GLOBUS_USER_HOME}/work/in.data</file>
    </deletion>
    <deletion>
      <file>
        file:///${GLOBUS_USER_HOME}/work/process_data.pl
      </file>
    </deletion>
  </fileCleanUp>
</job>
```

Der Auftrag selber ist ein Perlskript, welches Zahlen aus der Eingangsdatei erhöht und in die Ausgangsdatei schreibt, um so das Verarbeiten von Daten zu simulieren. Während jedem Schritt wird eine Sekunde gewartet. Anstatt einem Perlskript könnte auch ein Shell-, Pythonskript oder ein für die Zielarchitektur kompiliertes Binary verwendet werden. Die oben vorgestellte Jobbeschreibungsdatei wird als Parameter `-f` an das Kommando `globusrun-ws` weitergereicht. Das Argument `-submit` gibt den Befehl zum Abschicken. Mit `-F` wird der Globus Jobmanager angegeben. An dieser URL muss sich ein entsprechender Web Service befinden. Das Programm soll vom lokalen Batchsystem OpenPBS ausgeführt werden, daher wird `-Ft PBS` angehängt. Durch den Parameter `-b` wird der Job im Batch-Modus ausgeführt. So wird die EPR-Datei für den Job direkt ausgegeben. Ich leite die Informationen direkt in eine Datei um. Sie wird später gebraucht, um wieder auf den Job zuzugreifen. Das Programm `globusrun-ws` terminiert im Batch-Modus direkt nach dem Abschicken des Jobs. Der Parameter `-S` schließlich bewirkt, dass eine Delegation des eigenen Benutzerzertifikats erfolgt. Dies ist notwendig, damit das File Staging funktioniert, da die eigenen Berechtigungen per Proxyzertifikat an den GridFTP-Server weitergegeben werden müssen.

```
globusrun-ws -submit \
 -F https://host.de:8443/\
 wsrf/services/ManagedJobFactoryService \
 -Ft PBS -b -S -f simple_globus_job.xml >simple_globus_job.epr
```

Nachdem der Job abgeschickt wurde, erfolgt eine kurze Meldung, ob die Operation erfolgreich war. Ob der Auftrag in der Abarbeitung, schon fertig ist oder sich noch in der Warteschlange befindet kann nun jederzeit mit einem ähnlichen Kommando überprüft werden. Dabei wird nun als Befehl `-status` verwendet und die mit dem vorherigen Kommando gewonnene EPR-Datei als Parameter für `-j` angegeben. Außerdem lässt sich mit `-kill` ein Job abbrechen.

```
globusrun-ws -status -j simple_globus_job.epr
globusrun-ws -kill -j simple_globus_job.epr
```

Kapitel 6

Zusammenfassung

In dieser Arbeit wurden die wichtigsten Begriffe und Prinzipien für Grid-Middlewares beschrieben. Zu den wichtigsten Aufgaben einer Grid-Middleware zählt die Implementierung einer Sicherheitsschicht, das Ausführung von Benutzeraufträgen, die Verteilung von Daten, das Monitoring, Service Discovery sowie Accounting und Billing.

Mit dem Open Grid Forum gibt es eine Standardisierungsorganisation, die Unternehmen, Universitäten und Institute vereinigt und so die Entwicklung kompatibler Grid-Systeme vorantreiben will. Das wichtigste Produkt dieser Bemühungen ist die Open Grid Services Architecture, eine auf Web Services aufbauende Beschreibung für die wichtigsten Dienste in Grid Umgebungen. Durch die klare Spezifikationen von Schnittstellen sollen Grid-Middlewares zukünftig untereinander agieren können und die Inkompatibilitäten, die bisher noch vorherrschen mittelfristig ausgeräumt werden.

Die International Grid Trust Federation bietet eine Vertrauenshierarchie für weltweite Grid-Projekte, die auf einer Public-key Infrastruktur beruhen. Kernidee dabei ist, dass übergeordnete Stellen Zertifikate signieren und damit Vertrauen aussprechen. So können Benutzer und Rechner im Grid ihre Identität gegenseitig kontrollieren und sicheres, gemeinsames Arbeiten über Institutsgrenzen hinweg wird möglich. Darüber hinaus können Benutzer zeitlich begrenzte Proxyzertifikate ausstellen. Dies ist eine Voraussetzung für Delegation, also das Weiterreichen von Arbeitsaufträgen an dritte Stellen und somit eine wichtige Funktion von Grids.

Das Globus Toolkit als die bedeutendste und verbreitestete Middleware wurde vorgestellt. Sie besitzt umfangreiche Unterstützung für die OGSA, verfügt aber über keinen Resource Broker und damit über keine dynamische Zuteilung von Jobs sowie keine Accountinglösung. UNICORE besitzt im Gegensatz zum Globus Toolkit und zur dritten vorgestellten Grid-Lösung gLite eine graphische Benutzeroberfläche, doch zählt zu den größten Schwächen

von UNICORE die ungenügende Sicherheitsschicht. Sie besitzt keine Flexibilität und keine Unterstützung für erweiterte Techniken, wie Proxyzertifikate und Delegation von Benutzerrechten. Auch wird die OGSA bisher nicht implementiert. Die Middleware gLite ist als Middleware für das große EGEE Projekt entstanden und wird sehr intensiv im Produktionseinsatz genutzt. Sie ist besonders zugeschnitten auf die Bedürfnisse der Hochenergiephysik. Sie besitzt umfangreiche Lösungen in fast allen Aufgabenbereichen des Grid, ist allerdings mit der Umsetzung der OGSA noch nicht soweit fortgeschritten wie das Globus Toolkit.

Die bisherige Situation am Rechenzentrum der Universität Kassel besteht aus zwei Clustern auf denen lokale Batchsysteme installiert sind. Die Entscheidung für die Installation der Globus Middleware fiel aufgrund verschiedener Aspekte. UNICORE kann leider bisher nicht mit anderen Grid-Middlewares zusammen betrieben werden. Die gLite Software ist stark ausgerichtet auf eine Anwendungsdomäne. Das Globus Toolkit besitzt durch die sehr gute Unterstützung für die OGSA und die zügige Weiterentwicklung ausgezeichnete Zukunftsaussichten bezüglich der Interoperabilität und spätere Eingliederung in andere Grid-Umgebungen.

Nach der Installation des Globus Toolkit auf zwei abgesonderten Maschinen eines Clusters und der Signierung der Rechnerzertifikate durch eine zentrale Stelle, wurde die Grid-Middleware erst auf dem Linux-, dann auf dem AIX-Cluster des Rechenzentrums installiert. Außerdem gibt es eine Beschreibung für die Erstellung von Paketen für Client-Rechner. Das Testprogramm simuliert einen Verarbeitungsjob inklusive Transfer der Ein- und Ausgangsdaten.

Index

Literaturverzeichnis

[1] D-Grid Initiative. *http://www.d-grid.de/.*

[2] Globus Toolkit Official Documentation. *http://www.globus.org/toolkit/docs/.*

[3] Grid Infn Laboratory for Dissemination Activities. *https://gilda.ct.infn.it/.*

[4] LHC Computing Grid Web-GUI. *http://www.grid.uni-wuppertal.de/webgui/.*

[5] UNICORE Resource Broker. *http://uombroker.sourceforge.net/.*

[6] R. Alfieri, R. Cecchini, V. Ciaschini, L. dell'Agnello, A. Frohner, A. Gianoli, K. Lorentey, and F. Spataro. VOMS, an Authorization System for Virtual Organizations. *Grid Computing, First European Across Grids Conference*, 2004.

[7] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer. SETI@home: An experiment in public-resource computing. *Communications of the ACM*, 45, November 2002.

[8] C. Anglano, S. Barale, L. Gaido, A. Guarise, S. Lusso, and A. Werbrouck. An Accounting System for the Datagrid Project. February 2003.

[9] A. Anjomshoaa, F. Brisard, M. Drescher, D. Fellows, A. Ly, S. McGough, D. Pulsipher, and A. Savva. Job Submission Description Language (JSDL) Specification, Version 1.0. 2005.

[10] M. Antonioletti, M.P. Atkinson, R. Baxter, A. Borley, N.P. Chue Hong, B.Collins, J. Davies, D. Fitzgerald, N. Hardman, A.C. Hume, M. Jackson, A. Krause, S. Laws, N. W. Paton, T. Sugden, P. Watson, M. Westhead, and D. Vyvyan. OGSA-DAI Status Report and Future Directi-

ons. *Proceedings of the UK e-Science All Hands Meeting 2004*, September 2004.

[11] J.P. Baud, B. Couturier, C. Curran, J.D. Durand, E. Knezo, S. Occhetti, and O. Bärring. CASTOR status and evolution. 2003.

[12] R. Byrom, B. Coghlan, Cooke A, R. Cordenonsi, L. Cornwall, A. Datta, A. Djaoui, L. Field, S. Fisher, S. Hicks, S. Kenny, J. Magowan, W. Nutt, D. O'Callaghan, M. Oevers2, N. Podhorszki, J. Ryan, M. Soni, P. Taylor, A. Wilson, and X. Zhu. R-GMA: A Relational Grid Information and Monitoring System. December 2002.

[13] Thomas A. DeFanti, Ian Foster, Michael E. Papka, Rick Stevens, and Tim Kuhfuss. Overview of the I-WAY: Wide-area visual supercomputing. *The International Journal of Supercomputer Applications and High Performance Computing*, 10(2/3):123–131, Summer/Fall 1996.

[14] T. Dussa, U. Epting, B. Filipovic, G. Foest, J. Glowka, J. Götze, C. Grimm, M. Hillenbrand, C. Kohlschütter, R. Lohner, S. Makedanz, P. Müller, M. Pattloch, S. Piger, T. Straub, and J. Wiebelitz. Analyse von AA-Infrastrukturen in Grid-Middleware Version 1.1. *D-Grid Integrationsprojekt*, 2006.

[15] M. Ernst, P. Fuhrmann, M. Gasthuber, T. Mkrtchyan, and C. Waldman. dCache, a distributed storage data caching system. 2001.

[16] Dietmar W. Erwin and David F. Snelling. UNICORE: A Grid computing environment. *Lecture Notes in Computer Science*, 2150:825–834, 2001.

[17] J. Falkner, W. Müller, and M. Mucha. Bewertung bestehender Billing-Ansätze, Anforderungen an das D-Grid-Billing-Framework. June 2006.

[18] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, 2002.

[19] I. Foster, H. Kishimoto, A. Savva, D. Berry, A. Grimshaw, B. Horn, F. Maciel, F. Siebenlist, R. Subramaniam, J. Treadwell, and J. Von Reich. The Open Grid Services Architecture, Version 1.5. 2006.

[20] Ian Foster. What is the Grid? A Three Point Checklist. *GRIDToday*, 2002.

[21] Ian Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the Grid: Enabling scalable virtual organizations. *Lecture Notes in Computer Science*, 2150:1–??, 2001.

[22] Ian T. Foster and Carl Kesselman. *The Grid, Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, San Francisco, 1999.

[23] V. Huber. UNICORE: A Grid Computing Environment for Distributed and Parallel Computing. *Proceedings of 6th Internatioanl Conference on Parallel Computing Technologies (PaCT-2001), Novosibirsk, Russia, Springer, LNCS 2127*, pages 258–266, 2001.

[24] E. Huedo, R.S. Montero, and I.M. Llorente. The GridWay Framework for Adaptive Scheduling and Execution on Grids. *Scalable Computing - Practice and Experience*, 6:1–8, 2005.

[25] K. Kurowski, B. Ludwiczak, J. Nabrzyski, T. Piontek, and J. Pukacki. Report on the final release. 2006.

[26] E. Laure, S. Fisher, A. Frohner, C. Grandi, P. Kunszt, A. Krenek, O. Mulmo, F. Pacini, F. Prelz, J. White, M. Barroso, P. Buncic, F. Hemmer, A. Di Meglio, and A. Edlund. Programming the Grid with gLite. March 2006.

[27] L. Pearlman, V. Welch, I. Foster, C. Kesselman, and S. Tuecke. A Community Authorization Service for Group Collaboration. *Proceedings of the IEEE 3rd International Workshop on Policies for Distributed Systems and Networks*, 2002.

[28] Pawel Plaszczak and Richard Wellner. *Grid Computing: The Savvy Manager's Guide*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[29] M. Rambadt and P. Wieder. UNICORE – Globus: Interoperability of Grid Infrastructures. *Cray User Group Summit 2002 Proceedings*, 2002.

[30] M. Romberg. The UNICORE Architecture: Seamless Access to Distributed Resources. *Proceedings of the eighth IEEE International Symposium on High Performance Distributed Computing*, pages 287–293, 1999.

[31] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman an T. Maquire, T. Sandholm, D. Snelling, and P. Vanderbilt. Open Grid Services Infrastructure (OGSI). 2003.

[32] S. Venugopal, R. Buyya, and L. Winton. A Grid Service Broker for Scheduling e-Science Applications on Global Data Grids. *Concurrency and Computation: Practice and Experience*, 18:685–699, May 2006.

[33] C. Weider, J. Reynolds, and S. Heker. Technical Overview of Directory Services Using the X.500 Protocol. RFC 1309 (Informational), March 1992.

[34] V. Welch, I. Foster, C. Kesselman, O . Mulmo, L. Pearlman, S. Tuecke, J. Gawor, S. Meder, and F. Siebenlist. X. 509 proxy certificate for dynamic delegation. *3rd Annual PKI R&D Workshop*, 2004.