INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Fortgeschrittenenpraktikum

Adaptive Manageability Services im Grid

Tobias Abt

Aufgabensteller: Prof. Dr. Heinz-Gerd Hegering

Betreuer: Michael Schiffers
Nils genschen Felde
Dr. Helmut Heller

Abgabetermin: 27. Februar 2006

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Fortgeschrittenenpraktikum

Adaptive Manageability Services im Grid

Tobias Abt

Aufgabensteller: Prof. Dr. Heinz-Gerd Hegering

Betreuer: Michael Schiffers
Nils genschen Felde
Dr. Helmut Heller

Abgabetermin: 27. Februar 2006

Zusammenfassung

In dieser Arbeit werden das Globus Toolkit 4 und die darauf aufbauenden Manageability Services behandelt.

Zuerst wird dazu ein kurzer Überblick über die Funktionalität des Globus Toolkits in der Version 4 gegeben. Das Globus Toolkit besteht aus eine Vielzahl von Programmen, welche zum großen Teil als Web Services umgesetzt wurden, und ist für den Einsatz in einer globalen Gridinfrastruktur konzipiert. In der Gridszene besitzt das Globus Toolkit den Status des de facto Standards. Mit Hilfe dieser Grundlage wird dann eine Installation und die dazugehörige Konfiguration des Globus Toolkits in der Version 4.0.1 auf vier Rechnern beschrieben. Zu den einzelnen Abschnitten der Konfiguration werden jeweils spezielle Tests aufgeführt, mit denen die Funktionstüchtigkeit überprüft wurde.

Das zweite große Thema dieser Arbeit, die Manageability Services, werden darauf erst kurz eingeführt und dann exemplarisch für das Globus Toolkit 4 implementiert. Anhand dieser Implementierung wird nicht nur die Funktionsweise von Manageability Services beschrieben, sondern auch die Funktionsweise von Web Services, auf denen Manageability Services aufbauen, aufgezeigt. Desweiteren werden anhand der Manageability Services verschiedene Stufen der Sicherheit, wie zum Beispiel Transport Level Security oder Message Level Security, welche von dem Globus Toolkit zur Verfügung gestellt werden, umgesetzt.

Inhaltsverzeichnis

Inhaltsverzeichnis i

Abbildungsverzeichnis iii

Tabellenverzeichnis iv

1 Einführung 1
- 1.1 Einleitung . . . 1
- 1.2 Motivation . . . 1
- 1.3 Vorgehensweise . . . 2

2 Einführung in das Globus Toolkit 3
- 2.1 Aufgaben des Globus Toolkits . . . 3
- 2.2 Historische Entwicklung . . . 3
  - 2.2.1 Neue Komponenten des Globus Toolkits 4 . . . 4
  - 2.2.2 Geänderte Komponenten des Globus Toolkits 4 . . . 4
- 2.3 Aufbau des Globus Toolkits . . . 4
  - 2.3.1 Sicherheit: Grid Security Infrastructure (GSI) . . . 4
  - 2.3.2 Data Management . . . 8
  - 2.3.3 Common Runtime . . . 8
  - 2.3.4 Execution Management . . . 9
- 2.4 Verwandte Arbeiten . . . 10

3 Installation des Globus Toolkits Version 4.0.1 11
- 3.1 Vorbereitung . . . 11
- 3.2 Installation . . . 11
- 3.3 Konfiguration . . . 12
  - 3.3.1 Sicherheit . . . 12
  - 3.3.2 GridFTP . . . 14
  - 3.3.3 Java WS Core . . . 15
  - 3.3.4 Reliable File Transfer (RFT) . . . 16
  - 3.3.5 WS GRAM . . . 17

4 Einführung in Manageability Services 20
- 4.1 Definiton . . . 20
- 4.2 Wichtigkeit von Web Services . . . 20
- 4.3 Dynamische Systemintegration . . . 20
- 4.4 Beispiel: Linux OS Service . . . 21

5 Manageability Services 23
- 5.1 Service zum Überwachen der Prozessorleistung im Grid . . . 23
  - 5.1.1 Implementierung des Interfaces . . . 23

5.1.2 Schreiben des Services . . . 23
5.1.3 Definieren von Deploymentmarametern . . . 24
5.1.4 Erzeugen des gar-files . . . 24
5.1.5 Schreiben eines Clients . . . 24
5.2 Erweiterung des Beispiels . . . 24
5.2.1 Clientseite . . . 24
5.2.2 Serverseite . . . 25
5.3 Test des Dienstes . . . 27
5.3.1 Client . . . 27
5.3.2 Server . . . 27
5.3.3 Sicherheit . . . 27
5.4 Zusammenfassung . . . 29

6 Abschließende Betrachtung **30**
6.1 Zusammenfassung . . . 30
6.2 Weiterführende Arbeiten . . . 30
6.3 Danksagung . . . 31

A Hardware der Rechner **33**
A.1 projekt-ext10 . . . 34
A.2 projekt-ext11 . . . 34
A.3 projekt-ext12 . . . 34
A.4 projekt-ext13 . . . 34

B Installierte Software **35**

Literaturverzeichnis **39**

Abbildungsverzeichnis

2.1 Bestandteile des Globus Toolkits, aus [Soto 05] 5
2.2 Ablauf der Authentifizierung mittels Zertifikaten 7
2.3 Schichten der Web Service Architektur, aus [Fost 05] 9
2.4 Höhere Schichten eines Web Services, mit dem Weg der Nachricht, aus [Fost 05] 9
2.5 Verschiedene Zustände eines Jobs in GRAM, aus [Fost 05] 10

4.1 Graphische Oberfläche des Linux OS Service, aus [WBMH 03] 22

A.1 Anordnung der Rechner und ihre Hardware 33

Tabellenverzeichnis

5.1 Konstanten und ihre Funktion 25

Kapitel 1

Einführung

1.1 Einleitung

In den Zeiten von totaler Vernetzung und Rechnern, deren Leistung die meiste Zeit brach liegt, liegt es nahe, die nicht genutzten Ressourcen in irgendeiner Form unter bestimmten Umständen der Allgemeinheit zur Verfügung zu stellen. Aber auch immer komplexe Aufgaben, wie zum Beispiel das Verwalten von sehr großen Datenbeständen oder das Berechnen von genauen Simulationen, etwa eines Erdbebens, erfordern mehr Leistung, als es ein einziger Rechner zur Verfügung stellen kann. Anstatt nun immer größere und schnellere Rechner zu bauen, kann man nun mehrere „kleine“ Rechner „zusammenschließen“ und so mehr Leistung erhalten, als diese jeweils einzeln bieten würden.
An diesem Punkt kommt das Grid-Computing ins Spiel. Das Grid-Computing hat zum Ziel, aus einzelnen, vernetzten Rechnern ein Netz zu erstellen, in dem Ressourcen global verfügbar gemacht und verwaltet werden können. In [IF 01] wird das daraus resultierende Problem, das *Grid-Problem*, wie folgt definiert:
„(...) coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations.“
Es ist hierbei wichtig, dass man kontrollieren kann, welche Ressourcen verfügbar gemacht werden und wem der Zugriff auf jene erlaubt ist. Eine Gruppe, die über bestimmte Rechte verfügt, wird dabei *Virtuellen Organisationen* genannt.
Das Globus Toolkit stellt als Middleware die benötigten Funktionalitäten zur Verfügung, um mehrere Rechner zusammenzuschließen und verfügbare Ressourcen wie Anwendungen und Daten, Rechenleistung und Speicherkapazitäten oder individuelle Komponenten, aufzudecken. Des weiteren bietet es die Voraussetzungen, um die in der heutigen Zeit unverzichtbare Sicherheit zu gewährleisten und den Zugriff auf die Ressourcen zu koordinieren.
Das Globus Toolkit gilt dank seiner fortschrittlichen Technologie, seinem Open-Source Status, seiner engagierten Klientel und seiner kontinuierlichen Weiterentwicklung schon als der de facto Standard der Grid-Szene.

1.2 Motivation

Die Aufgabe dieses FoPras ist die Installation des Globus Toolkits in der Version 4.0.1 (siehe Anhang B), die Überprüfung der Funktionstüchtigkeit der grundlegenden Dienste und die exemplarische Entwicklung eines Anwendungsfalles für „Manageability Services“, der sich an den IBM Manageability Services [WBMH 03] orientiert. Zu beachten ist dabei, dass diese für die Globus Toolkit Version 3.0.2 entwickelt wurden und dass das Globus Toolkit in der aktuellen Version einige grundlegende Änderungen erfahren hat, auf die später noch genauer eingegangen wird. Desweiteren ist die verwendete Version des Toolkits recht

neu, und es wird sich zeigen müssen, wie weit sie ausgereift und wie gut die vorhandene Dokumentation ist.

1.3 Vorgehensweise

Zuerst wird das Globus Toolkit 4 auf den vier Testrechnern, die in dieser Arbeit anhand des ersten Labels ihres Fully Qualified Domain Name (FQDN) bezeichnet werden, also projekt-ext1{0,1,2,3}, installiert werden. Davor müssen diese für das Toolkit vorbereitet werden, also die benötigte Software installiert und ein spezieller Benutzer angelegt werden. Darauf wird die Funktionstüchtigkeit des Toolkits anhand verschiedener Tests sichergestellt und ein exemplarischer „Manageability Service“ geschrieben werden. Dieser dient auch dazu, verschiedene Sicherheitskonzepte darzustellen.

Kapitel 2

Einführung in das Globus Toolkit

2.1 Aufgaben des Globus Toolkits

Das Globus Toolkit ist eine Menge von Software Komponenten, welche zum einen Teil durch Web Services implementiert, zum anderen Teil durch sogenannte pre-Web Services realisiert wurden, um *Verteilte Systeme* aufzubauen. Dabei wurde besonderen Wert auf *Sicherheit* und *Skalierbarkeit* gelegt:
Durch die Implementierung von aktuellen Sicherheitsstandards wie zum Beispiel einer *Public Key Infrastructure* (PKI) kann die Interaktion der einzelnen Parteien über ein offen zugängliches Medium, also das Internet, erfolgen und durch den modularen Aufbau des Toolkits muss nicht zwangsweise das komplette Software Paket installiert und konfiguriert werden, sondern es ist möglich, sich ein seinen individuellen Bedürfnissen angepasstes System zusammenstellen.
Dadurch ist Globus Toolkit die grundlegende Software, um *Ressourcen* sicher online zur Verfügung zu stellen. Ressourcen sind dabei zum einen Rechenleistung und Datenbanken, zum anderen Programme, welche in der Form von Web Services implementiert wurden. Das Globus Toolkit bietet dabei eine Vielzahl von Web Services an und hat auch die Grundlage geschaffen, um eigene Web Services zu implementieren. Dazu bietet es noch *API*s und *Kommandozeilen*-Programme, um die verschiedenen Dienste erreichbar zu machen.
Zu dem Globus Toolkit gehört allerdings nicht nur das Toolkit (also die Software), sondern auch der Globus Teil:
Dieser besteht zum einen aus der *Gemeinschaft* von Benutzern, welche die Entwicklung des Toolkits weiter vorantreiben, zum anderen aus der *Infrastruktur*, zu finden unter globus.org, welche die Gemeinschaft z.B. durch Email-Listen und Repositories unterstützt.

2.2 Historische Entwicklung

Die Version 1.0.0 des Globus Toolkits ist am 29.10.1998 erschienen.
Die Version 2.4 hatte sich dann schon zum de facto Standard der Grid Szene entwickelt und ist auch heute noch weit verbreitet. Diese Version hatte zwar bereits Open Source Status, allerdings baute es im Großen und Ganzen auf keinen *formalen* technischen Spezifikationen auf. Die verwendeten Techniken wurden selbst entwickelt und nicht öffentlich diskutiert oder geprüft.

Der Kern der Version 3.0 basierte erstmals auf der *Open Grid Services Infrastructure (OGSI)*. Abgeschlossen wurde die 3.x Reihe mit der Version 3.2, der neue Kern des Toolkit ist mit Hilfe von neuen Techniken oder besser gesagt Standards entwickelt worden.

Auf die Unterschiede der 3.x Reihe zu der für diese Arbeit verwendeten Version des Toolkits in der 4.x Reihe wird im Folgenden kurz eingegangen.

2.2.1 Neue Komponenten des Globus Toolkits 4

Neu im Globus Toolkit sind der *C WS Core* und der *WS A&A Delegatioln Service*.
Der *C WS Core* wird in Kapitel 2.3.3 beschrieben, der *WS A&A Delegation Service* wird in Kapitel 2.3.1 näher ausgeführt.

2.2.2 Geänderte Komponenten des Globus Toolkits 4

Java WS Core: Als der Hauptbestandteil des Globus Toolkits hat diese Komponente einige grundlegende Änderungen erfahren. So basierte das Globus Toolkit 3.2 auf GWDSL das nun in der aktuelle Version durch den aktuellen Standard *WSDL* abgelöst wurde. Des weiteren war der Standard für Web Services OGSI (Open Grid Services Infrastructure), welcher inzwischen veraltet ist und vom W3C durch das *Web Services Resource Framework (WSRF)* ersetzt worden ist. Deshalb verwendet auch das Globus Toolkit 4 nun WSRF für die Implementierung von Web Services. Desweiteren benutzte das Globus Toolkit 3.2 sogenannte *wrapped* Formatierung von WSDL, das nun durch die sogenannte *document* Formatierung abgelöst wurde.
In Globus Toolkit 3.2 waren die *Logik* und der *Zustand* des Programmes noch nicht getrennt, in Globus Toolkit 4 wurde die Logik in eine Dienst-Klasse und die Zustandsinformationen in eine Ressource-Klasse aufgespalten. Das führt auch zu der getrennten Konfiguration von Dienst und Ressourcen (siehe Kapitel 3.3.3).

RFT: Die Implementierung wurde komplett geändert, da der zugrunde liegende Globus Kern geändert wurde. RFT wird in Kapitel 2.3.2 näher beschrieben.

2.3 Aufbau des Globus Toolkits

Abbildung 2.1 gibt einen Überblick über die einzelnen Bestandteile des Globus Toolkits: Die „WS Components“ sind dabei der Teil, der mithilfe von Web Services implementiert wurden, die „Non-WS Components“ dementsprechend auf „normale“ programmatische Art.

In dieser Arbeit kann natürlich nicht auf jede einzelne Komponente bis in letzte Detail eingegangen werden, das Folgende ist mehr als ein grober Überblick zu verstehen. Im Folgenden werden also die vier großen funktionalen Einheiten des Globus Toolkits, namentlich die *Sicherheitsinfrastruktur*, das *Datenmanagement*, das *Execution Mangagement* und die *Common Runtime* Umgebung (siehe Abbildung 2.1), vorgestellt.

2.3.1 Sicherheit: Grid Security Infrastructure (GSI)

Bevor wir zu GSI kommen, werfen wir kurz einen Blick auf die Grundlagen der Sicherheit:
Damit die Kommunikation zwischen zwei Parteien als sicher erachtet werden kann, müssen folgende drei Punkte beachtet worden sein, die sozusagen die Grundlage jeglicher Überlegung sein sollten:

Vertraulichkeit: Nur Befugte haben Zugriff auf die Daten.

Integrität: Der Empfänger muss sich sicher sein können, dass die Nachricht unverfälscht bei ihm ankommt.

Authentizität: Die beiden Kommunikationspartner sind auch wirklich die, welche sie vorgeben zu sein. Es muss also die Identität beider Seiten überprüft werden können.

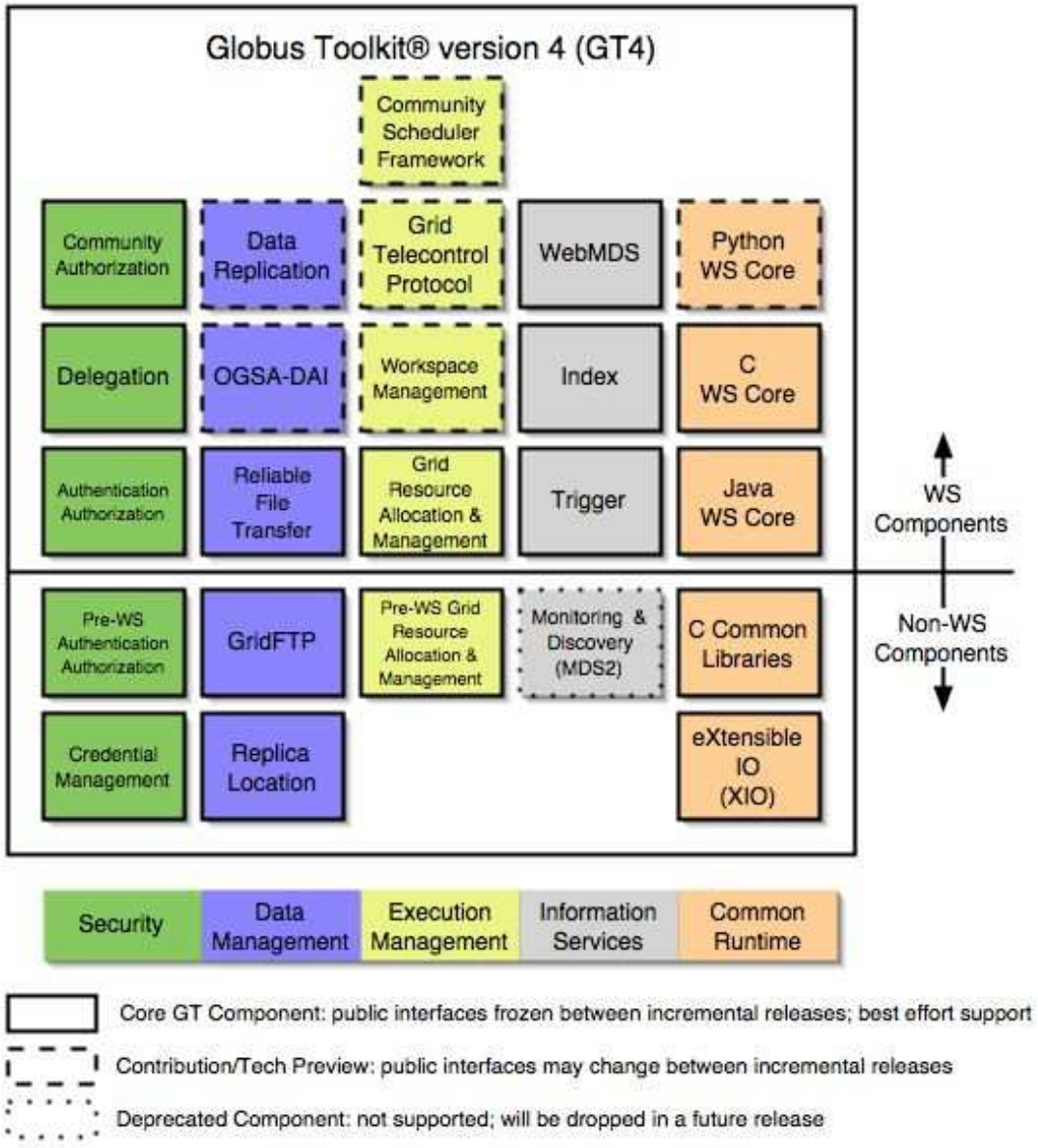

Abbildung 2.1: Bestandteile des Globus Toolkits, aus [Soto 05]

Ein weiteres Merkmal der Sicherheit, das allerdings nicht zu den absoluten Basics gehört, ist die **Autorisierung.** Dadurch kann man den Zugriff auf bestimmte Dienste und Ressourcen nur bestimmten Personen oder Objekten erlauben.

Die *Grid Security Infrastructure* bietet die notwendigen Grundlagen für *sichere Kommunikation*, *Delegation*, *Authentifizierung* und *Autorisierung*. Diese vier Punkte werden im Folgenden vorgestellt.

Sichere Kommunikation

GSI stellt verschiedene Konzepte zur Gewährleistung der Sicherheit bereit:

- Transport-Level Security: Die komplette Kommunikation wird mit Hilfe von TLS verschlüsselt. Da-

bei wird der Schutz und die Integrität der Kommunikation sowie die anonyme Authentifizierung unterstützt, auf *„Delegation“* muss aber verzichtet werden. Diese Methode bietet die beste Performanz und wird standardmäßig verwendet.

- Message-Level Security: Nicht der gesamte Datenverkehr, sondern nur die „Inhalte“ der verschickten Pakete werden verschlüsselt. Hierfür gibt es zwei Ausprägungen, zum einen *GSI Secure Message*, zum anderen *GSI Secure Conversation*.
  GSI Secure Message setzt auf dem WS-Security Standard auf und bietet Verschlüsselung und die Sicherstellung der Integrität. Die Performanz ist gut, wenn wenig Nachrichten geschickt werden.
  GSI Secure Conversation setzt auf der WS-SecureConversation Spezifikation auf. Neben den schon bei GSI Secure Message erwähnten Funktionalitäten, ermöglicht diese Technik außerdem noch anonyme Authentifizierung und *„Delegation“*, auf das später noch eingegangen wird. Da für GSI Secure Conversation vor dem eigentlichen Nachrichten-Austausch Sicherheitsparameter vereinbart werden, die alle nachfolgenden Nachrichten benutzen können, ist die Performanz erst bei mehreren Nachrichten gut.

Delegation

Der Delegation Service dient zur temporären Weitergabe von Rechten in Form von Zertifikaten. Dies wird im Folgenden anhand eines Beispiels erläutert:
A will von B einen Dienst in Anspruch nehmen. B stellt fest, dass er die Anforderungen von A nicht allein erfüllen kann und will den Auftrag (oder einen besser einen Unterauftrag) von A weiter an C leiten. Nun kann es passieren, dass C zwar A traut, aber nicht B. Eine Option wäre nun, dass C an A eine Anfrage betreffs des Ursprungs des Auftrags stellt. Diese Methode ist aber für A nicht wünschenswert, da bei großen Aufträgen A von Nachfragen leicht überschwemmt werden könnte. Eine Lösung wäre es, wenn A sein Zertifikat an B „ausleiht“, um die Aussagen von B zu beglaubigen. In der Praxis ist dies natürlich nicht machbar, da das Sicherheitsrisiko viel zu hoch wäre (wer weiß was B mit dem Zertifikat sonst noch alles anstellt...). Ein vertretbares Risiko dagegen ist B ein temporäres Zertifikat auszustellen, genannt *Proxy-Zertifikat*. Dieses ist nur kurze Zeit gültig – bei dem Globus Toolkit sind es zwölf Stunden – und ermöglicht B das Handeln an As statt, genannt Delegation.

Authentifizierung

Die Authentifizierung kann auf zweit Arten erfolgen: Zum einen die altbekannte Methode mit Benutzername und Passwort, die aber eher selten zum Einsatz kommt, zum anderen mit **X.509 Zertifikaten**. Ein Zertifikat ist Teil einer Public Key Infrastructure (PKI) und besteht aus einem
Subjekt, das den Benutzer oder das Objekt identifiziert,
einem öffentlichen *Schlüssel*, der zu dem Subjekt gehört,
die Identität einer *Certification Authority* (CA), die das Zertifikat ausgestellt hat,
und die *Signatur* der CA, mit der die Echtheit eines Zertifikates bestätigt/überprüft werden kann.
Die Certification Authority muss dabei eine dritte Autorität darstellen, welche die Identität des Zertifikatsinhabers überprüft – so muß man, um vom LRZ ein Zertifikat zu bekommen, dort seinen Ausweis vorlegen – und außerdem müssen beide Parteien jeweils der CA des anderen vertrauen.
Zu dem Zertifikat gehört der *geheime* Schlüssel. Was mit dem geheimen Schlüssel verschlüsselt wurden, kann mit dem öffentlich entschlüsselt und vice versa. Die Kombination aus geheimen Schlüssel und Zertifikat wird *credential* genannt.

Die gegenseitige Authentifizierung zwischen zwei Parteien, in dem Folgenden Beispiel zwischen A und B, läuft nun wie in dem in Abbildung 2.2 dargestellten Sequenzdiagramm ab.
Falls nun die entschlüsselte Nachricht x von A mit der generierten Nachricht x von B übereinstimmt, kann sich B sicher sein, dass A der ist, der er vorgibt zu sein. Nun wird der Prozess noch einmal andersherum

durchgeführt, und beide Parteien haben der Gegenseite ihre Identität bewiesen.

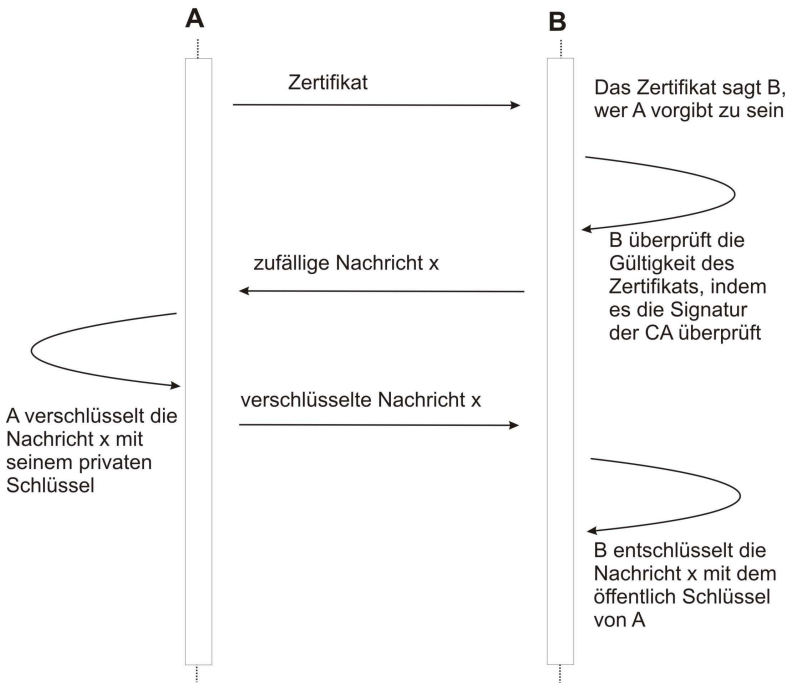

Abbildung 2.2: Ablauf der Authentifizierung mittels Zertifikaten

Autorisierung

Hierbei sind Client und Server getrennt zu betrachten: Beide sind gleichberechtigt in der Wahl des jeweiligen Interaktions-Partners. Betrachten wir zuerst die Möglichkeiten, die sich dem Server bieten:

- **none**: Man führt keine Autorisierung durch, verständlicherweise die einfachste Art.
- **self**: Der Zugriff wird erlaubt, wenn der Client die selbe Identität wie der Dienst aufweist.
- **host**: Der Client wird zugelassen, falls Credentials vorweisen kann, die einer Vorgabe entsprechen. Es werden also nur Verbindungen von einem Host erlaubt
- **gridmap**: Nur Benutzer im grid-mapfile werden zugelassen. In einem grid-mapfile werden Subjekte von Zertifikaten lokalen Identitäten (user-IDs) zugeordnet und stellt eine Access Control List (ACL) dar.
- **identity**: Der Client muss eine bestimmte Identität vorweisen. Entspricht einem grid-mapfile mit nur einem Eintrag, wird aber im Programm-Code definiert.
- **userName**: Es ist möglich, eine Autorisierung anhand von Benutzername und Passwort durchzuführen. Diese Methode wird allerdings kaum verwendet.
- **SAML**: Man kann die Autorisations-Entscheidung an einen Dienst weiterreichen. SAML ist so ein Dienst, er benutzt eine XML-Syntax, so genannte „assertions“, anhand derer Entscheidungen definiert werden können.

SAML steht für „Security Assertion Markup Language“ und ist ist als Vorschlag für eine Recommendation bei der „Global Grid Forum OGSA Security Working Group“ eingereicht worden. Der Sinn von SAML ist den Austausch von Autorisierung- und Authentifizierung-Informationen zu erleichtern.

Der Client besitzt nicht so viele Arten, den Host zu autorisieren, und trotz Namensgleichheit einiger Punkte, bieten sie in einem Fall andere Funktionalität.

- **none**: Keine Autorisierung.
- **self**: Die Identität des Servers muss mit der des Clients übereinstimmen.
- **host**: Der Server muss gültige Credentials besitzen.
- **identity**: Der Server/Dienst muss bestimmte Credentials besitzen.

2.3.2 Data Management

Die in Graphik 2.1 unter der Spalte des *Data Managements* aufgeführten Punkte werden im Folgenden beschrieben:

RLS: Dienst zum Registrieren und Entdecken von Informationen über Kopien.

GridFTP: sicheres, zuverlässiges Hochgeschwindigkeits-Protokoll zum Übertragen von Dateien, das für große Bandbreiten von WANs optimiert wurde. Das GridFTP Protokoll basiert auf FTP. Zusätzlich zu bereits definierten IETF RFCs wurden einige weitere Features eingebaut, um den heutigen Anforderungen gerecht zu werden.

Einige Besonderheiten von GridFTP sind:

- striping: Man hat mehrere Endpunkte (Rechner) an der Sender, Empfänger oder an beiden Seiten. So können große Dateien schneller übertragen werden, als ein Host allein fähig wäre, indem die Datei gleichzeitig von mehreren Hosts herunter- oder hochgeladen wird.
- GSI Security, siehe Kapitel 2.3.1
- Zuverlässigkeit: In bestimmten Intervallen (default: fünf Sekunden) werden vom empfangenden Server Marken gesetzt, bis zu der die Übertragung erfolgreich war. Bricht der Transfer ab, kann er anhand dieser Marken wieder aufgenommen werden, ohne komplett neu starten zu müssen.
- Third-party Transfer: Ein Client kann die Übertragung zwischen zwei Servern vermitteln.

RFT: Reliable File Transfer ist ein Web Service, der es ermöglicht, zuverlässig Dateien zu übertragen. Im Gegensatz zu GridFTP ist hier der Transfer auch vor einem Fehler auf Clientseite geschützt. Der Service funktioniert folgender weise: Der Client sagt dem Dienst, was wohin soll indem er ihm die URLs der jeweiligen Endpunkte gibt. Der Dienst schreibt diese in eine Datenbank und führt die Übertragung für den Client aus. Falls die Übertragung abbricht, kann sie dadurch wieder aufgenommen werden.

2.3.3 Common Runtime

Die Runtime-Componenten bieten Web und Pre-Web Services durch Bibliotheken und Tools, die es dem Globus Toolkit ermöglichen, diese Services plattformunabhängig anzubieten. Im Einzelnen bestehen sie aus dem *Java WS Core*, dem *C WS Core*, dem *Python WS Core* und *XIO*. Die ersten drei sind zum Implementieren von Diensten in der jeweiligen Sprache geeignet, XIO hingegen bieten den Zugriff auf viele grundlegende Protokolle.
Im folgenden wird das Konzept des WS Core kurz darstellt.
Zuerst einmal zu der Frage, was ein Web Service ist: Ein Web Service ermöglicht die Zusammenarbeit von Maschinen im Netzwerk. Dazu stellt er ein Interface in einem für Maschinen lesbaren Format dar (WSDL), andere Maschinen kommunizieren durch SOAP Nachrichten in der durch das Interface beschriebenen Art.

Als Protokoll wird gewöhnlicherweise HTTP verwendet. Die Funktionalitäten für Sicherheit (siehe Kapitel 2.3.1) und Management werden auf höheren Schichten bereitgestellt. Abbildung 2.3 verdeutlicht den Aufbau eines Web Services.

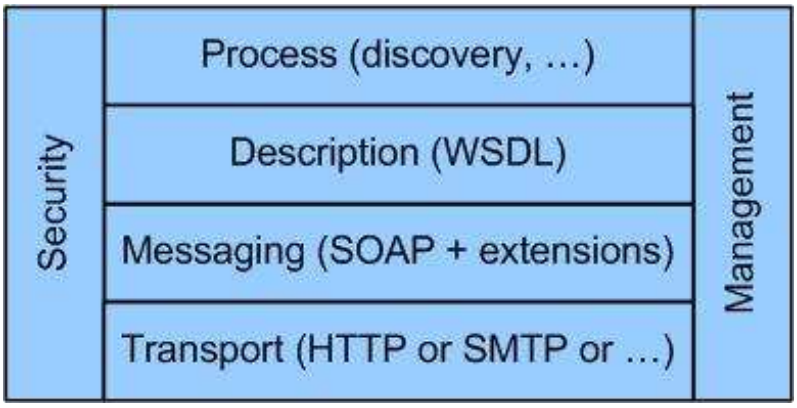

Abbildung 2.3: Schichten der Web Service Architektur, aus [Fost 05]

Bei der Implementierung eines Web Services wird zwischen zwei Teilen unterschieden:

1. Der *plattformunabhängige* Teil, der die SOAP-Nachrichten empfängt und identifiziert, um das entsprechenden Programm, das die Nachricht bearbeitet, aufzurufen
2. Der *plattformabhängige* Teil, also das Programm / Web Service, das die Nachricht bearbeitet

Diese Trennung hat den Vorteil, dass bei dem Schreiben von Web Services nur auf den plattformabhängigen Teil geachtet werden muss.

Abbildung 2.4 gibt einen Überblick über beide Komponenten und stellt dazu noch den Weg einer Nachricht bei Web Services dar.

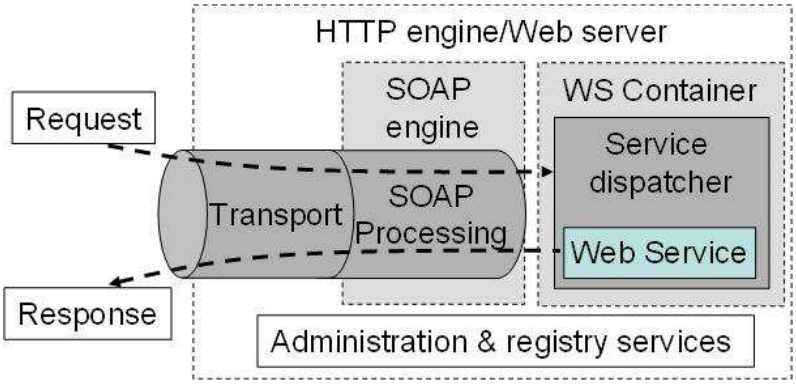

Abbildung 2.4: Höhere Schichten eines Web Services, mit dem Weg der Nachricht, aus [Fost 05]

2.3.4 Execution Management

Um einen Benutzer ein Programm auf einem entfernten Rechner ausführen zu lassen, bietet das Globus Toolkit zwei Möglichkeiten: Zum einen kann man einen Web Service schreiben, durch den das Programm aufgerufen werden kann, zum anderen kann man das Programm mittels *GRAM* (Grid Resource Allocation & Management) direkt aufrufen. *GRAM* ist die Hauptkomponente des *Execution Managements*.

Web Services bieten dem Benutzer keine Kontrolle über die Ausführung eines Programms, alles was er machen kann, ist den Web Service mit eventuellen Parametern aufrufen um dann ein Ergebnis zurückzubekommen.
Mit *GRAM* hingegen kann der Benutzer auf vielfältige Weise mit dem Programm (*Job*) interagieren:

- Die auszuführenden Programme können *selbst bereitgestellt* werden.
- Ausführbare Dateien, Eingabe sowie Ausgabe können sehr groß sein und/oder von verschiedenen Aufrufen genutzt sein. Es wird die Funktionalität bereitgestellt, um die Bereitstellung von Daten *verwalten* zu können (*staging*).
- Auf die Ausgabe kann zugegriffen werden, während der Job läuft (*streaming*).
- Der Benutzer kann den Job jederzeit *abbrechen*.
- Ein *Scheduler* sorgt für die optimale Ausführung aller eingereichten Jobs.
- Ein Job kann verschieden Zustände annehmen, die von dem Benutzer *eingesehen* werden können. Abbildung 2.5 zeigt die verschiedenen möglichen Zustände und ihre Zusammenhänge.

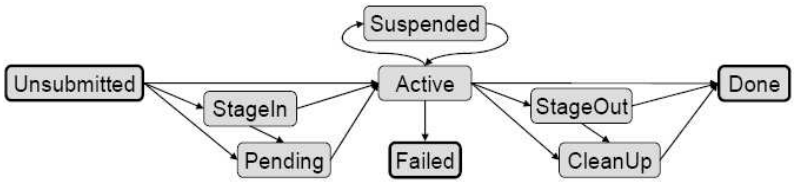

Abbildung 2.5: Verschiedene Zustände eines Jobs in GRAM, aus [Fost 05]

GRAM bezeichnet eigentlich zwei Komponenten:

1. Eine Suite von „pre-Web Services“.
2. Eine Suite von Web Services. Wenn man dieses GRAM meint, spricht man meist von *WS GRAM*.

Es wird natürlich auch die sichere Interaktion der Benutzer mit GRAM ermöglicht.
WS GRAM benutzt WSRF Funktionalität um Benutzer zu authentifizieren und bösartige Beeinträchtigungen der Jobs zu verhindern, während „pre-Web Service GRAM“ direkt auf GSI zugreifen kann.
Desweiteren wird bei GRAM durch Delegation (siehe Kapitel 2.3.1) die Möglichkeit geschaffen, dass ein Benutzer Rechte weitergeben kann, um zum Beispiel auf bestimmte Ressourcen zugreifen zu können.

2.4 Verwandte Arbeiten

Von dem Globus Team gibt es selbstverständlich eine Installationsanleitung ([glo 05]) für ihr Toolkit. Diese Installationsanleitung ist schon sehr ausgereift, befindet sich allerdings noch in der Entwicklung.
Zur Entwicklung von Web Services gibt es das Tutorial von Borja Sotomayor ([Soto 05]). Es führt aber nicht nur in die Web Services ein, sondern versorgt auch mit dem nötigen Hintergrundwissen über das Globus Toolkit und die verwendeten Konzepte. Auch dieses Tutorial befindet sich noch in der Entwicklung – laut Aussagen des Autors deckt es erst 10% von dem ab, was das Toolkit anzubieten hat – und so ist noch geplant, das Kapitel über *Sicherheit* weiter auszubauen, desweiteren sollen noch Kapitel über *Information Services*, *Data Management* und *Execution Management* folgen.
Die Grundlage für Manageability Services ist [WBMH 03]. Da diese Arbeit sich aber auf die Globus Toolkit Version 3.2 bezieht, die inzwischen veraltet ist, dient sie mehr als Orientierungshilfe und Vorlage für Konzepte.

Kapitel 3

Installation des Globus Toolkits Version 4.0.1

Die Installation bzw. die Konfiguration des Globus Toolkits ist gut dokumentiert ([glo 05]) und die einzelnen Abschnitte sind mit Tests ausgestattet, welche die Funktionalität des vorher Konfigurierten überprüfen. Bei der folgenden Arbeit wurde [glo 05] als Vorlage verwendet.
Das Toolkit wurde dabei auf vier Rechnern (siehe Anhang A) installiert, die im Folgenden, wie schon einmal erwähnt, mit dem ersten Label ihres FQDN bezeichnet werden, also *projekt-ext1{0,1,2,3}*. Der FQDN lautet dementsprechend projekt-ext1{0,1,2,3}.nm.ifi.lmu.de. Das Betriebssystem auf allen vier Rechnern ist *SuSe Linux 8.2*.

3.1 Vorbereitung

Der erste Schritt für die Installation des Globus Toolkits ist die Installation der für das Toolkit benötigter Software. *In Anhang B sind die benötigten Programme und ihre verwendeten Versionen sowie die Bezugsquelle im Netz aufgelistet.* Dabei wurde immer in [glo 05] empfohlenen Programme verwendeten, meist in der aktuellen Version, jedoch immer in der geforderten Version. So gab es zwischen den verwendeten Programmen keine Probleme.
Als zweiter Schritt wird ein spezieller Benutzer angelegt, welcher der Eigentümer des Globus Toolkits wird und unter dessen Account die Toolkit-spezifischen Anwendungen laufen werden. Der Benutzer wird *globus* genannt. Des weiteren muss noch das Verzeichnis für die Installation angelegt werden (`/usr/local/globus-4.0.0`)[1], dessen Eigentümer auch der globus-Account wird. Der Pfad zu diesem Ordner ist wichtig für das Globus Toolkit sowie für Programme, die das Toolkit verwenden wollen, daher wird er in der globalen Variable $GLOBUS_LOCATION gespeichert. Diese Variable *muss* vor der Installation der Toolkits existieren.

3.2 Installation

Für die Installation des Globus Toolkits wurde die Source-Distribution in der Version 4.0.1 (siehe Anhang B) verwendet.
Die Installation an sich läuft unter dem globus-Account ab. Dabei kommen die unter Linux üblichen Befehle zum Einsatz:

[1] Der Ordner für die Globus-Installation trägt aus historischen Gründen noch die Bezeichnung der Version 4.0.0

- `./configure --prefix=$GLOBUS_LOCATION`
- `make`
- `make install`

Mit dem „./configure“ - Befehl können noch einige weitere optionale Pakete und Features installiert oder ausgeschlossen werden. Für dieses Projekt wurden, wie oben zu erkennen ist, die Standardeinstellungen hergenommen. Bei Bedarf lassen sich aber eventuell benötigte Pakete jederzeit nachträglich installieren.
Der „make“ - Befehl dauert recht lange, auf unseren Rechnern waren es mehrere Stunden.

Bei der Installation trat das erste Problem auf: Auf den Rechnern waren schon Host-Zertifikate installiert, allerdings sind diese nicht richtig konfiguriert worden. Dadurch brach die Installation ab, da hier schon die Funktionalität der Zertifikate überprüft wird, falls diese vorhanden sind. Wie Zertifikate korrekt installiert werden, ist in Kapitel 3.3.1 beschrieben. Wenn keine Host-Zertifikate vorhanden sind, funktioniert die Installation natürlich auch (oder gerade deswegen...).

3.3 Konfiguration

3.3.1 Sicherheit

Die Sicherheits-Mechanismen und -Merkmale des Globus Toolkit 4 werden in Kapitel 2.3.1 beschrieben.

Zertifikate

Der erste Schritt bei der Konfiguration der Sicherheitskomponenten ist die Anforderung von Zertifikaten, mit denen Personen und Ressourcen in einer Public Key Infrastructure (PKI) authentifiziert und autorisiert werden. Dieser Prozess kann recht langwierig sein, vor allem da Fehler meist nicht sofort zu erkennen sind und sich so erst im Laufe des Amtsweges offenbaren.
Also wird im Folgenden kurz die Vorgehensweise beschrieben.

Die Beantragung teilt sich in zwei Abschnitte auf:

1. Erstellen eines Requests-Files am Rechner
2. Ausfüllen eines schriftlichen Antrags

Der erste Teil läuft wie folgt ab:
Man besorgt sich das Zertifikat einer Certification Authority (CA), in unserem Fall ist das die DFN-CERT Services GmbH (http://www.dfn-cert.de/). Der DFN-CERT stellt die Zertifikate aus, und durch das Vorhandensein eines Zertifikates einer CA signalisiert man dem Globus Toolkit, dass man den von dieser CA ausgestellten Zertifikaten traut. Des weiteren werden mit den beim CA-Zertifikat mitgelieferten Konfigurationsdateien die Host- und User-Zertifikats-Requests erstellt. Die CA-Zertifikate (man kann natürlich mehreren CAs trauen) und zugehörige Dateien werden im Verzeichnis $/etc/grid-security/certificates$ gespeichert.
Um ein Zertifikatsantrag zu erzeugen, bringt das Globus Toolkit einen eigenen Befehl mit, die Syntax zum Beantragen eines Host-Zertifikats sieht dann wie in Listing 3.1 aus:

Listing 3.1: grid-cert-request Befehl: Syntax für Host-Zertifikat

```
grid-cert-request -int -host projekt-ext10.nm.ifi.lmu.de -prefix projekt-ext10
```

Wichtig sind der „-int“ und der „-host“ Schalter, durch ersteren ist man in der Lage, die für das *Subjekt* des Zertifikates benötigten Angaben zu machen, der zweite zeigt an, dass ein Request für ein Host-Zertifikat

erstellt wird und benötigt den FQDN des Hosts als Parameter. Bei der Abfrage, die durch den „-int“ Schalter ausgelöst wird, gibt man dann folgende Werte ein:

- Level 0 Country [DE]:““ (d.h. einfach Enter drücken, dadurch wird der default-Wert übernommen)
- Level 1 State [Bayern]:“.“ (der Punkt läßt den Eintrag wegfallen)
- Level 2 Locality [Muenchen]:“.“
- Level 3 Organisation [Leibniz-Rechenzentrum]:“GridGermany“
- Level 4 Organisation Unit [Grid Computing]:“Leibniz-Rechenzentrum“
- Name (e.g., John M. Smith) []:“.“

Wichtig ist er Eintrag „GridGermany“, der Eintrag „Leibniz-Rechenzentrum“ kann dem Namen der eigenen Einrichtung angepasst werden.
Der Schalter „-prefix projekt-ext10“ ist praktisch, wenn man mehrere Host-Zertifikate generiert, dadurch wird im Namen „host“ durch das angegebene Präfix ersetzt und so kann man gleich alle Zertifikate in einem Ordner speichern.
Mit diesem Befehl wurden nun die drei Dateien

- projekt-ext10cert.pem
- projekt-ext10cert_request.pem
- projekt-ext10key.pem

im Verzeichnis $/etc/grid-security/$ erstellt. Die Datei „projekt-ext10cert.pem“ wird später einmal das Zertifikat enthalten, ist nun aber noch leer. „projekt-ext10key.pem“ enthält den Schlüssel und darf nur von „root“ gelesen werden (wichtig!). Der Request (projekt-ext10cert_request.pem) wird nun an das LRZ, namentlich Herrn Richter (richter at lrz.de), geschickt. Das LRZ fungiert für den DFN-CERT als so genannte *Registration Authority (RA)*, eine Einrichtung, die berechtigt ist, die Identität eines Antragsstellers festzustellen.
Denn zusätzlich zu dem Request muss ein *schriftlicher Antrag* abgegeben werden, der bei einem Host-Zertifikat-Request von einem Verantwortlichen – wie etwa einem Administrator – ausgefüllt werden muss und der seinen Ausweis zur Feststellung seiner Identität bei einer RA persönlich vorlegen muss.
Entgegen der Information, die man von dem „grid-cert-request“ Befehl erhält, muss man sich selbst mit dem LRZ in Verbindung setzten, um dem selbigen seinen Ausweis zur Feststellung der Identität vorzulegen, natürlich nachdem man das Request-File abgeschickt hat.

Für die Beantragung eines User-Zertifikat sieht das ganze fast genauso aus, der Aufruf erfolgt wieder mit „grid-cert-request“ Befehl, diesmal aber nur mit dem Schalter „-int“. Zusätzlich muss man nun seine email-Adresse und seinen Namen angeben, so wie er im Ausweis steht. Spitznamen oder Ähnliches werden nicht angenommen!
Außerdem wird das Zertifikat mit einem Passwort geschützt sein, das auch hier angegeben werden muss.
Zu bemerken ist noch, dass der Befehl hängen bleibt, wenn der „-int“ Schalter weggelassen wird.
Den schriftlich Antrag trägt man auch hier zum LRZ und läßt dort bei der Gelegenheit gleich seine Identität feststellen.
Das Zertifikat bekommt man dann per Email zugeschickt und ersetzt damit den Dummy, der vorher als Platzhalter angelegt wurde.

Firewall

Da die Rechner alle an das Internet angeschlossen sind, wurde auf jedem eine einfache Firewall installiert, die von außen für folgende TCP-Ports oder Dienste offen ist:

- SSH

- 2811 für GridFTP
- 3000-3095 für die Globus-Dienste (wird auch vom LRZ normalerweise verwendet)
- 8080 für den unsicheren Globus-Container
- 8443 für den sicheren Globus-Container

Damit die Globus-Dienste den hiermit freigeschalteten Portrange nutzen, wird noch die globale Variable `GLOBUS_TCP_PORT_RANGE` angelegt, die mit dem Wert `3000,3095` initialisiert wird.
In der Richtung von innen nach außen sowie für das interne Interface sind alle Verbindungen erlaubt.

3.3.2 GridFTP

Die Funktionsweise von GridFTP wird in Kapitel 2.3.2 beschrieben.

Konfiguration von GridFTP

Die Konfiguration eines einfachen GridFTP-Servers ohne besondere Merkamle – wie etwa striping – gestaltet recht einfach. Man hat dabei zunächst die Wahl, ob man den Server unter (x)inetd oder als daemon starten will. Es wurde die daemon-Variante gewählt, da hier mitgeloggte Informationen leichter zu erreichen sind und man den Server nur dann laufen lassen kann, wenn man ihn wirklich braucht. Die Konfiguration des Servers wird in der Datei `$GLOBUS_LOCATION/etc/gridftp.conf` vorgenommen und bietet zahlreiche Einstellungen. Für die einfachen Zwecke hier sieht die Datei auch dementsprechend einfach aus (Listing 3.2):

Listing 3.2: gridftp.conf

```
port 2811
allow_anonymous 1
anonymous_user gridftp
banner "gridprojekt1 test"
log_level ALL
```

Hierbei wird lediglich der Port, unter dem der Dienst erreichbar sein wird, auf 2811 festgelegt, der anonyme Zugang erlaubt und definiert, ein kleines Banner ausgegeben und alles mitgeloggt, was so anfällt.

Test von GridFTP

Etwas diffiziler wurde es, als die Funktionalität durch die mitgelieferten Tests überprüft werden sollte. Mit folgenden drei Schritten sollten die Tests durchlaufen:

1. gültigen Proxy generieren: `tobi@projekt-ext10:~> grid-proxy-init`
2. Die Globus Benutzer Umgebung „sourcen“:
   `tobi@...:~> source $GLOBUS_LOCATION/etc/globus-user-env.sh`
3. `$GLOBUS_LOCATION/test/globus_ftp_client_test/TESTS.pl` ausführen

Darauf sollte man folgende Ausgabe erhalten:

Listing 3.3: gridFTP Testsuite Ergebnis

```
./TESTS.pl 2>&1 | tee $GL/gridftp.log
Started server at port 35556
Running sanity check
Server appears sane, running tests
```

```
globus-ftp-client-caching-get-test..........ok
globus-ftp-client-caching-transfer-test......ok
globus-ftp-client-create-destroy-test........ok
globus-ftp-client-exist-test................ok
globus-ftp-client-extended-get-test..........ok
globus-ftp-client-extended-put-test..........ok
        1/100 unexpectedly succeeded
globus-ftp-client-extended-transfer-test.....ok
globus-ftp-client-get-test...................ok
globus-ftp-client-lingering-get-test.........ok
globus-ftp-client-multiple-block-get-test....ok
globus-ftp-client-partial-get-test...........ok
globus-ftp-client-partial-put-test...........ok
globus-ftp-client-partial-transfer-test......ok
globus-ftp-client-plugin-test................ok
globus-ftp-client-put-test...................ok
        1/124 unexpectedly succeeded
globus-ftp-client-size-test..................ok
globus-ftp-client-transfer-test..............ok
globus-ftp-client-user-auth-test.............ok
All tests successful (2 subtests UNEXPECTEDLY SUCCEEDED).
Files=18, Tests=2567, 1909 wallclock secs (516.44 cusr + 60.69 csys = 577.13 CPU)
```

Probleme bei dem Testen von GridFTP

Im Folgenden werden die Fehler und die Lösung beschrieben, soweit eine gefunden wurde.

- Tests können nicht als „root“ ausgeführt werden. (Der Server dagegen muss mit root-Rechten laufen)
- Fehlermeldung: „Unable to create gridmap-file“. Die Ursache ist ein abgelaufenes Proxy-Zertifikat.
- sporadisch kam noch die Fehlermeldung
  `Sanity check of source (gsiftp://localhost/etc/group) failed.`
  Woran das gelegen hat, lässt sich mit absoluter Gewissheit nicht sagen, ab und zu aber half es, die Globus Benutzer Umgebung zu „sourcen“.
- Falls man den Test ohne das Startskript „TESTS.pl“ startet, schlägt genau der letzte der 2567 Tests (*globus-ftp-client-user-auth-test*, besteht selbst wiederum aus zwei Tests) fehl. Um nun nicht immer alle 2567 Tests durchlaufen lassen zu müssen, um den letzten Test mit seinen zwei Untertests auszuführen, wurde der Test ohne das Skript einzeln gestartet. Das separate Ausführen der einzelnen Tests führt aber zu recht chaotischen Ergebnissen, wie leider recht spät herauskam, denn beim *globus-ftp-client-user-auth-test* mit seinen zwei Untertests, von denen auch noch einer das Scheitern der Authentifizierung testet, also quasi immer erfolgreich ist, war das nicht so ganz ersichtlich. Erst als einer von den anderen Tests, die aus mehr Untertests bestehen, ausgeführt wurde, ist das Problem offenkundig geworden.

3.3.3 Java WS Core

Der Java WS Core wird in Kapitel 2.3.3 vorgestellt.

Konfiguration des Java WS Core

Die Konfiguration spielt sich hier auf drei Ebenen ab:

- Container-Ebene, was der globalen Konfiguration entspricht.

- Service-Ebene, durch welche die globale Konfiguration überschrieben werden kann.
- Ressource-Ebene, welche die unterste Schicht ist.

Die Konfiguration der ersten beiden erfolgt deskriptiv mithilfe von den beiden Dateien „server-config.wsdd“, welche Informationen über den Web Service enthält, und „jndi-config.xml“, welche Informationen über das Ressourcen-Management enthält. Falls der Dienst Sicherheit unterstützt, kann noch der Security Descriptor – „security-config.xml“ – vorhanden sein. Die Dateien befinden sich für den Container in `$GLOBUS_LOCATION/etc/globus_wsrf_core/`, für jeden Dienst in seinem eigenen Verzeichnis in dem Ordner `$GLOBUS_LOCATION/etc/`. Diese Art der Konfiguration hat den Vorteil, dass sie leicht zu ändern ist.
Die Konfiguration von Ressourcen selbst erfolgt auf programmatische Art.

Test des Java WS Core

Um die Funktionalität zu überprüfen, wurde auch hier wieder eine Testsuite mitgeliefert. Die Tests liefen zwar alle erfolgreich ab (laut Ausgabe), jedoch verwirren die geworfenen Fehlermeldungen etwas. Wahrscheinlich werden einfach Fehler – oder Fehlerbehandlungen – getestet.

3.3.4 Reliable File Transfer (RFT)

RFT wird in Kapitel 2.3.2 beschrieben.

Konfiguration des RFT

Da RFT eine Datenbank benötigt (siehe 2.3.2), muss man als erstes eine JDBC-kompatible Datenbank einrichten. Da PostgreSQL mit dem Globus Toolkit getestet wurden, lag es nahe, diese auch her zunehmen und wurde von der SuSe Linux 8.2 CD installiert (siehe Anhang B).
Die Konfiguration von PostgreSQL teilt sich in folgende Schritte auf:

1. Postmaster deamon so konfigurieren, dass er TCP-Verbindungen annimmt.
   Dazu wird in der Konfigurationsdatei `$PGDATA/postgresql.conf` der Eintrag „`tcpip_socket = true`“ hinzugefügt. In [glo 05] wird ein anderer Weg vorgeschlagen (postmaster script ändern), um den TCP Support zu erlangen. Beide Wege führen zum selben Ziel, es ist Geschmackssache, welche Methode man wählt.

2. PostgreSQl Benutzer erstellen
   Dazu führt man unter dem postgres-Account den Befehl
   `postgres@projekt-ext10:/> creatuser globus -P` aus. Wichtig ist der Schalter „-P“, dadurch kann man gleich ein Passwort für den Benutzer eingeben. Falls man einen Benutzer wieder löschen will geschieht dies mit dem Kommando „dropuser“.

3. Sicherheit für die Datenbank einschalten
   Dazu wird in der Datei `$PGDATA/pg_hba.conf` der Eintrag `host rftDatabase globus 10.153.51.150 255.255.255.255 md5` hinzugefügt. Die IP-Adresse ist dabei diejenige des internen Interfaces, so ist die Datenbank nicht von außen zu erreichen und man muss die Firewall nicht extra anpassen.

   Die nächsten Schritte werden als globus Benutzer ausgeführt!

4. Datenbank für RFT erstellen
   `globus@projekt-ext10:~> createdb rftDatabase`

5. Die Datenbank mit passenden Schemas füllen
   ```
   globus@projekt-ext10:~> psql -d rftDatabase -f
     $GLOBUS_LOCATION/share/globus_wsrf_rft/rft_schema.sql
   ```
6. In globus die Parameter für Datenbank einstellen
   Dazu wird in der Datei
   `$GLOBUS_LOCATION/etc/globus_wsrf_rft/jndi-config.xml`
   unter dem Eintrag `userName` Wert „globus“ eingetragen und unter dem Eintrag `password` das unter 2. eingestellte Passwort. Desweiteren wird unter `connecionString` der Wert „jdbc:postgresql://gridprojekt0.lab.ifi.lmu.de/rftDatabase“ eingetragen.

Dies schließt die grundlegende Konfiguration von RFT ab.

Test des RFT

Auch zu RFT wird eine Testsuite mitgeliefert. Diese Test waren mit die unangenehmsten, da sie recht spärlich beschrieben sind, und jede Menge Fehler lieferten.
Den Hauptteil der Fehler kann man ausmerzen, indem man in den *.xfr - Dateien in `$GLOBUS_LOCATION/share/globus_wsrf_rft_test/` anstatt „localhost“ als Adresse für den Container den FQDN einträgt. Dasselbe gilt für die Datei „test.properties“ im selben Verzeichnis. Der FQDN wird deshalb benötigt, da die Adresse, über die der Dienst aufgerufen wird, mit der Adresse im Host-Zertifikat verglichen wird, und dort steht nun mal der FQDN des Hostes. Ein weiteres Manko ist noch, dass im Java-Code des Testes noch so ein „localhost“ vorkommt, der natürlich nicht ohne weiteres gegen den FQDN ausgetauscht werden kann. So ist ein (Unter-)Test immer zum Scheitern verurteilt. Trotz sorgfältiger Überprüfung aller Einstellung und einschlägigen Internetseiten blieben jedoch noch zwei Fehler (failure) bei den Tests „testDeleteDir“ und „testAllOrNone“ übrig (bei dem Rechner *projekt-ext13* nur bei „testAllOrNone“), dessen Ursache nicht geklärt werden konnte. Es wird sich in der Praxis zeigen müssen, ob der Fehler am Test lag, oder ob wirklich ein Fehler in der Konfiguration oder im Programm daran schuld ist.
Ein weiteres Problem ist, dass PostgreSQL von Zeit zu Zeit irgendwie seine Datenbank beschädigt und im Zuge dessen nicht mehr starten kann. Dies ist bis jetzt zweimal passiert, der Ursache dafür ist nicht nachvollziehbar. Nachdem die Datenbank neu angelegt wurde, lief PostgreSQL jedenfalls wieder.

3.3.5 WS GRAM

Eine kurze Einführung in WS GRAM wird in Kapitel 2.3.4 gegeben.

Konfiguration

Für *WS GRAM* muss *sudo* konfiguriert werden. Dies geschieht in der Datei `/etc/sudoers`, in der die in Listing 3.4 aufgeführten Einträge hinzugefügt werden:

Listing 3.4: Eintrag für /etc/sudoers

```
# Globus GRAM entries
   globus  ALL=(username1,username2) NOPASSWD:
     /usr/local/globus-4.0.0/libexec/globus-gridmap-and-execute
     -g /etc/grid-security/grid-mapfile
     /usr/local/globus-4.0.0/libexec/globus-job-manager-script.pl *

   globus  ALL=(username1,username2) NOPASSWD:
     /usr/local/globus-4.0.0/libexec/globus-gridmap-and-execute
     -g /etc/grid-security/grid-mapfile
```

```
/usr/local/globus-4.0.0/libexec/globus-gram-local-proxy-tool *
```

Zu beachten ist dabei dass die Einträge von „globus“ bis „*“jeweils eine große Zeile bilden.
Desweiteren darf beim Verweis auf die Globus-Dateien nicht wie sonst üblich die Variable `$GLOBUS_LOCATION` verwendet werden.
Die Einträge bewirken, dass GRAM nur Programme von den Konten ausführt, die in dem *grid-mapfile* aufgeführt werden. Dazu muss allerdings der Globus-Container unter dem root-Account laufen.

Tests

Die Funktionalität von GRAM wurde durch zwei einfachen Test überprüft:
Zuerst wurde einfach ein kleiner Job abgearbeitet, welcher erfolgreich war, wie Listing 3.5 zeigt.

Listing 3.5: Einfacher GRAM Test

```
tobi@projekt-ext13:~> globusrun-ws -submit -c /bin/touch touched_it
Submitting job...Done.
Job ID: uuid:b2573222-a4d0-11da-b7ec-00300503948c
Termination time: 02/25/2006 00:58 GMT
Current job state: CleanUp
Current job state: Done
Destroying job...Done.
tobi@projekt-ext13:~> ls -l ~/touched_it
-rw-r--r--    1 tobi   users    0 Feb 24 01:58 /home/tobi/touched_it
```

Hierbei wird lediglich im Home-Verzeichnis die leere Datei `touched_it` angelegt.

Als nächsten Test wurde nun noch die Job-Beschreibung aus einer (XML-)Datei ausgelesen. Die Datei (`test_super_simple.xml`) ist in Listing 3.6 beschrieben, der Testdurchlauf ist in Listing 3.7 dargestellt.

Listing 3.6: XML-Datei für GRAM Test

```
<job>
    <executable>/bin/echo</executable>
    <argument>this is an example_string </argument>
    <argument>Globus was here</argument>
    <stdout>${GLOBUS_USER_HOME}/stdout</stdout>
    <stderr>${GLOBUS_USER_HOME}/stderr</stderr>
</job>
```

Listing 3.7: GRAM Test mit Job-Beschreibung in XML-Datei

```
tobi@projekt-ext13:~> globusrun-ws -submit -f test_super_simple.xml
Submitting job...Done.
Job ID: uuid:faed6604-a4d1-11da-98f6-00300503948c
Termination time: 02/25/2006 01:07 GMT
Current job state: Active
Current job state: CleanUp
Current job state: Done
Destroying job...Done.
```

Durch diesen Test werden im Home Verzeichnis die beiden Dateien „`stderr`“ und „`stdout`“ angelegt. Die Datei „`stdout`“ enthält dabei den Satz „`this is an example_string  Globus was here`“.

Mit dem Einrichten von GRAM ist die Konfiguration des Globus Toolkits abgeschlossen, in dem nächsten Kapitel werden nun die *Manageability Services* behandelt.

Kapitel 4

Einführung in Manageability Services

4.1 Definiton

Der Begriff Manageability Services wird in „Grid-Enabled Manageability Services for Linux Resources“ ([WBMH 03]) eingeführt:
Manageability Services bilden das Bindeglied zwischen Management Software und Ressourcen. Sie haben dabei das Ziel, auf der *vorhanden Infrastruktur* aufzubauen und sich *dynamisch* in ein System einzugliedern.

4.2 Wichtigkeit von Web Services

Zur Realisierung von Manageability Services werden Web Services verwendet. D.h. es werden die on-demand und selbst beschreibenden Merkmale der Web Services verwendet, um in *Autonomen-* und/oder *Grid*-Systemen Ressourcen zu entdecken, mit ihnen zu interagieren, sie zu überwachen und Information von ihnen erhalten zu können.
Es ist dabei wichtig, dass ein *offener (Industrie-)Standard* zum Einsatz kommt, also zum Beispiel WSRF, da dadurch die Heterogenität der vielen verstreuten Systeme überwunden werden und so eine einheitliche Schnittstelle geschaffen werden kann, um in vielfältigster Weise mit den Ressourcen zu interagieren.
In [WBMH 03] wird noch OGSI als Standard verwendet. Das geschah wegen dem damals aktuellen Globus Toolkit 3.2, für das OGSI als Standard gedient hat. Dieser ist allerdings inzwischen veraltet und in Globus Toolkit 4 durch WSRF ersetzt worden, wie in Kapitel 2.2 schon erwähnt worden ist.

4.3 Dynamische Systemintegration

Ein weiterer wichtiger Punkt bei den Manageability Services ist die dynamische Systemintegration. Sie kommt mit der Verwendung des Globus Toolkits und dem darin zugrunde liegenden WSRF. Dadurch wird eine dienstorientierte Infrastruktur auf die vorhandenen Betriebssystem aufgesetzt. So können die zugrunde liegenden Ressourcen nach außen verfügbar und durch eine einheitliche Schnittstelle zugreifbar gemacht werden.

Ein Administrator könnte so zum Beispiel eine normale Management Konsole verwenden, welche die verfügbaren Dienste entdecken und sie sich an binden kann, um eine ganze Serverfarm zentral zu managen,

ohne sich über Systemupdates und Veränderungen in der Verwaltungs-Software sorgen zu müssen.

4.4 Beispiel: Linux OS Service

Ein in [WBMH 03] vorgestelltes Beispiel ist der *Linux OS Service*. In dem Linux Betriebssystem sind viele Parameter durch das /proc leicht zugänglich und veränderbar. Der Dienst nutzt dies aus und bietet einen mit „OS Parameter“ einen portType mit den zugehörigen *get* und *set* Funktionen, um auf die System-Parameter zugreifen zu können. Für alle anderen Operationen ist der „Linux OS“ portType definiert worden.
So können folgende veränderbaren Parameter, die sogenannten Effektoren gesteuert werden:

- Kernel Parameters
- File System Parameters
- Virtual Memory Parameters
- Network Core Parameters
- Network IPv4 Parameters
- Network IPv4 Route Parameters
- Network Unix Parameters
- Debug Parameters
- ABI Parameters

Abbildung 4.1 zeigt die GUI des Services.
In dieser wird die Adresse des Dienstes eingeben (1.) und es können verschiedene Sicherheitsmerkmale ausgewählt werden (2.).
Unter (3.) können die oben genannten Parameter aus einer Liste gewählt , der aktuelle Wert angeschaut und eventuell neu gesetzt werden. Dies wird durch den „OSParameter“ portType umgesetzt.
Desweiteren kann der Service jeden beliebigen Befehl in dem System ausführen, das Betriebssystem neu starten und herunter fahren. Dies wird von dem „LinuxOS“ portType erledigt, durch den auch die durchschnittliche Systembelastung der letzten, der letzten fünf und der letzen zehn Minuten angezeigt werden kann (4.).

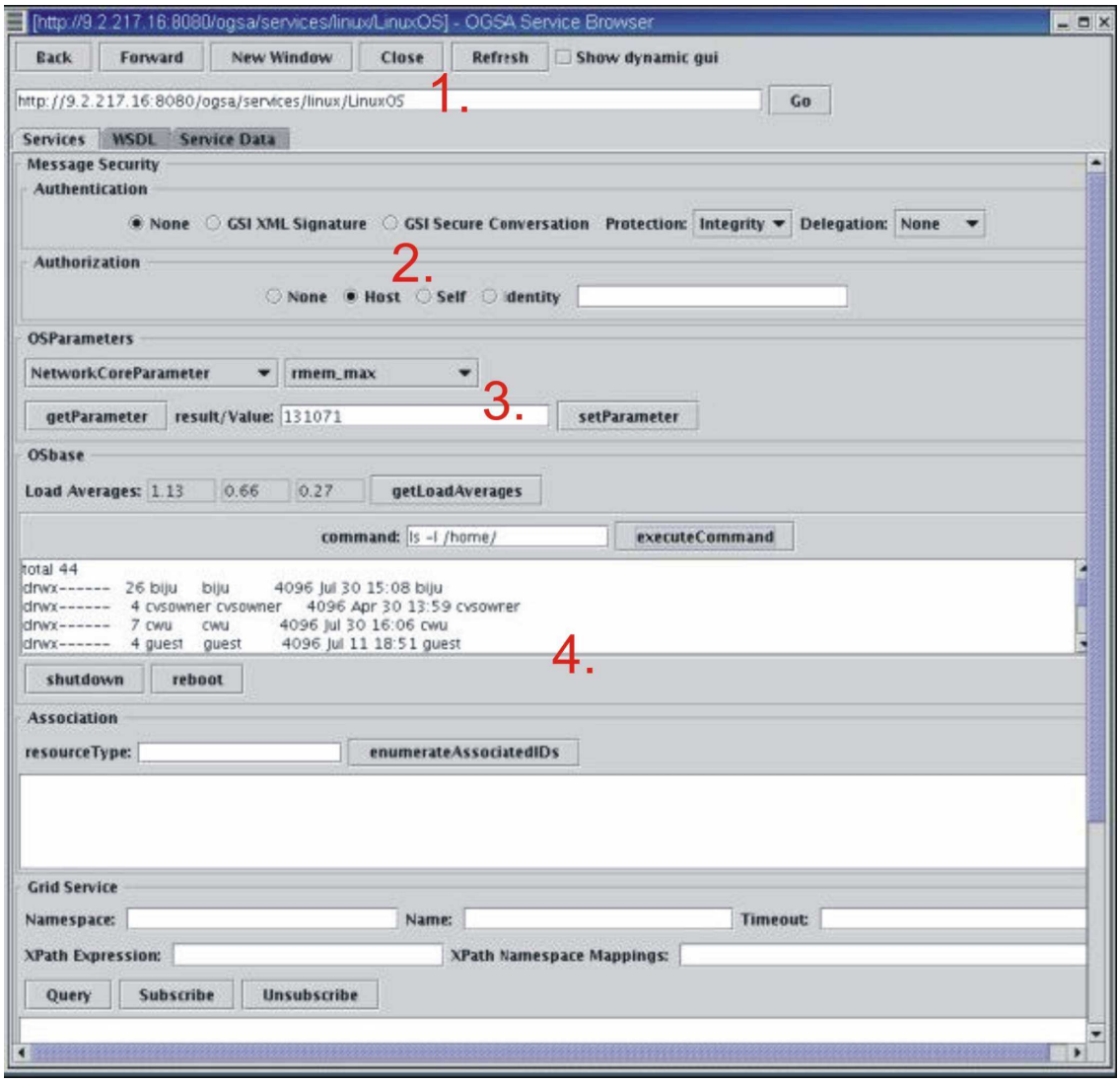

Abbildung 4.1: Graphische Oberfläche des Linux OS Service, aus [WBMH 03]

Kapitel 5

Manageability Services

Eine kurze Einführung in die Manageability Services wird in Kapitel 4 gegeben.

5.1 Service zum Überwachen der Prozessorleistung im Grid

Der Anfang in Bezug auf Webservices wird hier mit einem einfachen Dienst gemacht, durch den man die durchschnittliche Prozessorauslastung der letzten Minute, der letzten fünf Minuten und der letzten zehn Minuten abfragen kann. Dazu wird die Linux-Variable `loadavg` ausgelesen und zurückgegeben. Bei dem Dienst wurde noch keine Rücksicht auf Trennung von Service und Resource genommen. Des weiteren war auch nur eine Instanz dieses Dienstes möglich. Es hätte natürlich auch wenig Sinn gemacht diese Funktionen bei einem derart simplen Beispiel zu implementieren. Auf den globus-Seiten findet sich ein sehr gutes Tutorial von Borja Sotomayor ([Soto 05]), allerdings befindet es sich noch in einem Beta-Stadium, die hier verwendete Version ist 0.2.1.

Das Schreiben eines Web Services spaltet sich in kleine Abschnitte auf, in dem jeweils eine bestimmte Funktion eingebaut wird, und also werden im Folgenden die einzelnen Schritte anhand der resultierende Dateien/Programme dargestellt.

5.1.1 Implementierung des Interfaces

Im Interface werden unsere Methoden abstrakt definiert. Dies geschieht WSDL. WSDL hat den Vorteil, dass es plattformunabhängig ist. So kann der Service in einer beliebigen Sprache geschrieben werden und durch das Interface mit einer anderen beliebigen Sprache angesprochen werden. Die Methoden im Interface werden als portTypes bezeichnet. Aus dem WSDL-file werden später zwei Arten von *Stubs* generiert, der *Client-Stub* und der *Server-Stub*:

- Der Client-Stub generiert SOAP-Anfragen und interpretiert die SOAP-Antworten des Servers.
- Der Server-Stub dementsprechend interpretiert SOAP-Anfragen und generiert SOAP-Antworten, die zum Client geschickt werden.

5.1.2 Schreiben des Services

Der Dienst selbst in Java geschrieben, das Globus Toolkit bietet auch einen C- und Python WS Core, also könnte man auch eine dieser Sprachen verwenden.

5.1.3 Definieren von Deploymentmarametern

Um dem Container mitzuteilen, wie er den Web Service anbieten soll, und um noch einige Parameter für den Dienst festzulegen, verwenden wir zwei Dateien. Die erste nennt sich „deploy-server.wsdd“ und die andere „deploy-jndi-config.xml“.
Die erste ist im WSDD-Format, dem „Web Service Deployment Descriptor“-Format, geschrieben. In ihr wird z.B. die URI des Dienstes festgelegt.
Die zweite ist in XML geschrieben und hat bei unserem simplen Beispiel so gut wie keine Bedeutung. Wir benötigen sie aber um mit den Regeln konform zu bleiben.

5.1.4 Erzeugen des gar-files

Mit diesen vier Dateien ist der Web Service noch lange nicht komplett. Glücklicherweise können die noch benötigten Teile aus den bis jetzt geschriebenen erzeugt werden. Dazu kommt das Tool *Ant* zum Einsatz, das die Früchte unserer Arbeit der drei vorherigen Schritte zusammenfasst, soweit nötig kompiliert und die noch fehlenden Teile erzeugt. *Ant* wird dabei über einen Skript von dem Globus Toolkit angesprochen, in diesem Fall sieht die Syntax wie in Listing 5.1 aus.

Listing 5.1: Syntax von *globus-build-service.sh*

```
./globus-build-service.sh -d first/prozess -s schema/prozess/Prozess.wsdl
```

„first/prozess“ spiegelt dabei die Ordnerstruktur wider, die Datei „Prozess.wsdl“ ist das in Kapitel 5.1.1 angesprochene Interface.
Das Ergebnis ist ein so genanntes „Grid-Archive“, oder kurz GAR - Datei. Mithilfe des globus-Befehls „`globus-deploy-gar <gar-file>`“ werden dann alle benötigten Dateien an den richtigen Ort geschrieben (Deployment). Falls der Dienst wieder entfernt werden soll, kann dies mittels des Befehles „`globus-undeploy-gar <gar-id>`“ geschehen. `<gar-id>` ist dabei der Name der GAR-Datei ohne die .gar Endung.

5.1.5 Schreiben eines Clients

Unser Client wurde in Java geschrieben und bietet noch keine großen Besonderheit, er kann auf den Dienst zugreifen und das war´s.

5.2 Erweiterung des Beispiels

Nun werden noch Sicherheitsmerkmale in das Beispiel integriert werden. In Abschnitt 2.3.1 wurden die Sicherheitskonzepte des Globus Toolkits vorgestellt. Nun wollen wir in medias res gehen und die Implementierung der Konzepte betrachten.

5.2.1 Clientseite

Als erstes wird der Client sicher gemacht, indem im Stub bestimmten Konstanten gesetzt werden. In Tabelle 5.1 sind die möglichen Konstanten und ihre Wirkung aufgelistet.

Eine der ersten drei Konstanten wird immer mit einer der zweiten beiden kombiniert.
Es ist möglich, mehrere dieser Paare in Reihe zu schalten, um besondere Funktionen – wie etwa Delegation – der jeweiligen Mechanismen zu kombinieren.
`GSI_ANONYMOUS` steht für sich allein, eine Kombination wäre auch wenig sinnvoll.

Konstante	Aktiviertes Merkmal
GSI_SEC_CONV	GSI Secure Conversation
GSI_SEC_MSG	GSI Secure Message
GSI_TRANSPORT	GSI Transport
ENCRYPTION	Vertraulichkeit
SIGNATURE	Integrität
GSI_ANONYMOUS	keine Authentifizierung

Tabelle 5.1: Konstanten und ihre Funktion

Die Identität des Client wird durch setzen dieser Konstanten (außer GSI_ANONYMOUS) dabei automatisch übermittelt.
Das Setzen der Eigenschaft auf dem Stub kann nun z.B. so aussehen wie in Listing 5.2.

Listing 5.2: Sicherheit - Client

```
((Stub) prozess)._setProperty(Constants.GSI_SEC_CONV, Constants.SIGNATURE);
```

Die Autorisation auf Clientseite erfolgt durch die Konstante AUTHORIZATION in Kombination mit einer der Klassen NoAuthorization, SelfAuthorization, HostAuthorization, IdentityAuthorization. Die vier Schemas werden in 2.3.1. beschrieben.
Im Quellcode sieht das ganze dann z.B. für die Host-Autorisation so aus: 5.3

Listing 5.3: Autorisation - Client

```
((Stub) prozess)._setProperty(Constants.AUTHORIZATION,
        HostAuthorization.getInstance());
```

5.2.2 Serverseite

Die Sicherheitseinstellungen auf der Serverseite wird im Gegensatz zu der Clientseite deskriptiv erledigt. Mann kann zwar auch dem Client auf diese Weise Sicherheit beibringen, aber es wird im allgemeinen auf die im vorigen Kapitel (5.2.1) vorgestellte programmatische Art gemacht.
Die Datei, in der die Sicherheitseinstellungen beschrieben werden, ist eine XML-Datei, die wie in diesem Beispiel aussehen kann (Listing 5.4):

Listing 5.4: XML-Datei der Sicherheitseinstellungen

```
<securityConfig xmlns="http://www.globus.org">

<!-- -1-  define grid-map-file -->
 <gridmap value="etc/secure_prozess/grid-map-test"/>

<!-- -2-  authorisation method -->
 <authz value="host"/>

<!-- -3-  authentication method, for each operation seperatly defined -->
 <method name="getLoad1Min">
        <auth-method>
               <GSISecureConversation>
                      <protection-level>
                             <integrity/>
                      </protection-level>
               </GSISecureConversation>
        </auth-method>
 </method>
```

```
 <method name="getLoad5Min">
        <auth-method>
                <GSISecureMessage>
                        <protection-level>
                                <privacy/>
                        </protection-level>
                </GSISecureMessage>
        </auth-method>
 </method>

 <method name="getLoad10Min">
        <auth-method>
                <GSITransport/>
        </auth-method>
 </method>

<!-- -4-  default for any other method (there aren´t any in our case...) -->
        <auth-method>
                <GSITransport/>
        </auth-method>

</securityConfig>
```

zu -1- Im Grid-Map-File wird festgelegt, welche Benutzer Zugriff auf den Dienst haben, in Kapitel 2.3.1 wurde die Funktionsweise schon einmal beschrieben.

zu -2- Das `<authz/>` Element legt durch das *value* Attribut die Autorisations-Methode fest, hier wurde die *Host* Methode gewählt. Dazu muss natürlich noch irgendwo der Host, von dem man die Aufrufe erlauben will, ausgewählt werden. Dies geschieht in der Datei „deploy-server.wsdd“ durch den Eintrag `<parameter name="hostAuthz-url" value="projekt-ext13.nm.ifi.lmu.de"/>`. Die restlichen Möglichkeiten zur Autorisierung wurden in Kapitel 2.3.1 dargestellt.

zu -3- Zu jeder Funktion kann man separat die Authentifizierungs-Methode bestimmen. Zu den drei verfügbaren Methoden (siehe Kapitel 2.3.1), von denen auch mehrere unter dem `<auth-method/>` Element stehen können, kann jeweils in dem `<protection-level/>` Subelement eine bestimmte Sicherheitsstufe festgelegt werden:

- `<integrity/>`: Die Nachrichten muss signiert sein (Integer, d.h unverfälscht).
  Bei `<GSITransport/>` ist dies automatisch vorgegeben.

- `<privacy/>`: Die Nachrichten müssen vertraulich sein (also verschlüsselt).

Falls kein `<protection-level/>` Subelement angegeben wird, kann der Client alle verfügbaren Optionen verwenden.
Falls mehrere Authentifizierungs-Methoden angegeben wurden, stehen dem Client all jene zur Verfügung.

zu -4- Hier wird der Sicherheitsstandard für die übrigen Funktionen festgelegt.

5.3 Test des Dienstes

5.3.1 Client

Der Aufruf des Clients erfolgt über die Kommandozeile, wie in Listing 5.5 dargestellt wird. An der Adresse des Servers (`https://...`) ist zu erkennen, dass *TLS* verwendet wird.

Listing 5.5: Client: Aufruf und Ausgabe

```
java -classpath ./build/stubs/classes/:$CLASSPATH
 secure.client.ProzessService_instance.ClientConvSig
 https://projekt-ext13.nm.ifi.lmu.de:8443/wsrf/services/secure/ProzessService

Processload in the past minute: 0.32
Processload in the past five minutes: 0.13
Processload in the past ten minutes: 0.04
```

5.3.2 Server

Der Container kann so eingestellt werden, dass er einige Informationen mitloggt. In Listing 5.6 ist eine solche Ausgabe dargestellt. Man erkennt, dass

1. die Methode `getLoad1Min` aufgerufen wurde.
2. der Aufruf mit dem Zertifikat von Tobias Abt gemacht wurde.
3. der Service auf dem Host `projekt-ext13.nm.ifi.lmu.de` ausgeführt wird.

Die beiden letzteren Informationen stammen aus dem User- bzw. Host-Zertifikat, die bei 2. und 3. beschriebene Zeile entspricht dem *Subjekt* des jeweiligen Zertifikates.

Listing 5.6: Container: Log

```
2006-02-20 22:41:17,981 INFO  impl.ProzessService
                    --- 1. ---
 [ServiceThread-10,logSecurityInfo:156] SECURITY INFO FOR METHOD 'getLoad1Min'
2006-02-20 22:41:17,984 INFO  impl.ProzessService
 [ServiceThread-10,logSecurityInfo:160] The caller is:
                    ---- 2. ---
 /C=DE/O=GridGermany/OU=Leibniz-Rechenzentrum/CN=Tobias Abt
2006-02-20 22:41:17,986 INFO  impl.ProzessService
 [ServiceThread-10,logSecurityInfo:164] INVOCATION SUBJECT
2006-02-20 22:41:17,988 INFO  impl.ProzessService
 [ServiceThread-10,logSecurityInfo:165] Subject:
        Principal:
                    --- 3. ---
/C=DE/O=GridGermany/OU=Leibniz-Rechenzentrum/CN=host/projekt-ext13.nm.ifi.lmu.de
Private Credential: org.globus.gsi.gssapi.GlobusGSSCredentialImpl@1d47f59
```

5.3.3 Sicherheit

Aus den obigen Ausgaben kann man nicht erkennen, ob der Client auch wirklich auf sichere Art und Weise, also verschlüsselt, mit dem Server kommuniziert. Deshalb schauen wir im Folgenden uns den Datenverkehr zwischen beiden mit einem Protokollanalysator an. Dazu wird dieser zwischen Client und Server

geschaltet. Damit man in dem Protokollanalysator sinnvolle Informationen mitlesen kann, muss noch TLS deaktiviert werden, der Service also in einem unsicheren Container gestartet werden. Der Zugriff auf den unsicheren Container erfolgt dann über `http`, im Gegensatz zu dem in Listing 5.5 dargestellten Aufruf über `https`.
Auf Clientseite wird dazu *GSI Secure Conversation* und *Vertraulichkeit* verwendet. So kann die Steuerinformation der Nachricht gelesen werden, der eigentliche in Inhalt der Nachricht ist jedoch verschlüsselt. In Listing 5.7 ist die Ausgabe des Protokollanalysators – etwas gekürzt – dargestellt. Der verschlüsselte Inhalt der Nachricht ist ganz unten unter dem Tag `<xenc:CipherData>` zu erkennen.

Listing 5.7: Ausgabe des Protokollanalysators

```
==============
Listen Port: 8081
Target Host: localhost
Target Port: 8080
==== Request ====
POST /wsrf/services/secure/ProzessService HTTP/1.1
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime,
 multipart/related, text/*
User-Agent: Axis/1.2RC2
Host: localhost:8081
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: "http://www.globus.org/namespaces/se
cure/ProzessService_instance/ProzessPortType/getLoad1MinRequest"
Transfer-Encoding: chunked
Connection: close

a57
<soapenv:Envelope xmlns:soapenv=
    "http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/03/addressing"
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Header>

<!-- ... -->

<!-- die Adresse des Dienstes -->

      <wsa:To soapenv:mustUnderstand="0">
   http://projekt-ext13.nm.ifi.lmu.de:8081/wsrf/services/secure/ProzessService
      </wsa:To>

<!-- die aufgerufene Methode -->

      <wsa:Action soapenv:mustUnderstand="0">
   http://www.globus.org/namespaces/secure/ProzessService_instance/
   ProzessPortType/getLoad1MinRequest
      </wsa:Action>

<!-- ... -->

   </soapenv:Header>

   <soapenv:Body>
```

```
      <xenc:EncryptedData Id="EncDataId-15595312"
        Type="http://www.w3.org/2001/04/xmlenc#Content">
         <xenc:EncryptionMethod Algorithm=
            "http://www.globus.org/2002/04/xmlenc#gssapi-enc">
         </xenc:EncryptionMethod>
         <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
            <wsse:SecurityTokenReference xmlns:wsse=
             "http://docs.oasis-open.org/wss/2004/01/
␣␣␣␣␣␣␣␣␣␣␣␣␣oasis-200401-wss-wssecurity-secext-1.0.xsd">
               <wsse:Reference URI="#SecurityContextToken-22375698"
                ValueType="http://www.globus.org/ws/2004/09/
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣security/sc#GSSAPI_CONTEXT_TOKEN">
               </wsse:Reference>
            </wsse:SecurityTokenReference>
         </ds:KeyInfo>

<!-- verschlüsselte Nachricht -->

         <xenc:CipherData>
            <xenc:CipherValue>FwMAAYhvH9YgN/v2Fn+52aJRjt99dyn
            4RachH5t+KmSNIfRsNh5xASTv2wCVVxcrAgbeWrFdQwrPDLMa
            xPBJ1TrZcl3DFntY7hHasLUXa6WubY0TEjZk2o1uhakMMVE5K
            G1+b3rF4msMoS+fApU+1HRqZxQGtdyXPhj5sPWjchUpOU6iu3
            2ozh1QrdD7obkISyMY3lpIA9zJnbeOH2PPzTbIOkF1a96BMGa
            T2QGISFcoSbzc/bQFQ4QgfQPiupgLRUdomYm22ZcaqfO4QFcX
            oAOSwXXO7I3Cwbema7lZNraErPDvEpxzS1dHKHv8RU4JEv4Ia
            U+dRA7WHrnsvDHBx8lI2aZoiaPE4ArMWdLo+PvhdLy+t4n87G
            ryNe0T15r1UI09ylNEMDZ/og5+qejazF50AbB3sArZhhRvli+
            FRpp41qP8sAKzAsqFpg/ry67tKNEwjJsOsXwArHMU18EUxX91
            r4BUogWH1aQRwdJ1oorp5IeooiVR9t77NLgl/DGHYb8rhWHvt
            /ugIb5p2w==
          </xenc:CipherValue>
         </xenc:CipherData>
      </xenc:EncryptedData>
   </soapenv:Body></soapenv:Envelope>
0
```

Somit wurde sichergestellt, dass die eingestellte Verschlüsselung auch wirklich funktioniert.

5.4 Zusammenfassung

In diesem Kapitel haben wir einen Service vorgestellt, mit dessen Hilfe man die `loadavg` - Variable des `/proc` Dateisystems eines Rechners im Grid von einem beliebigen Rechner aus ermitteln kann. Der Aufruf des Services erfolgt dabei wie in Listing 5.5 dargestellt. Anhand des Dienstes wurden desweiteren verschiedene Sicherheitsfeatures dargestellt und getestet.

Kapitel 6

Abschließende Betrachtung

6.1 Zusammenfassung

In dieser Arbeit wurde die Installation sowie die Konfiguration des Globus-Toolkits in der Version 4.0.1 durchgeführt und die Funktionsweise zum einen anhand der mitgelieferten Tests und zum anderen durch die Implementierung eines Web-Services überprüft.
Die Installation verlief im im Großen und Ganzen problemlos ab, wenn man davon absieht, dass durch eine falsche Konfiguration der vorhandenen Zertifikate anfangs die Installation abbrach.
Die möglichen Einstellungen für die Konfiguration sind in der guten Anleitung auf der Webseite [glo 05] meist anhand von Beispielen ausführlich erklärt. Dadurch war auch die Konfiguration kein großes Problem.
Die Funktionstüchtigkeit der Konfiguration konnte man durch die mitgelieferten Tests sicherstellen. Dabei traten die ersten Probleme auf: Für die richtige Konfiguration und Ausführung der Tests war oft mehr Arbeit nötig, als für die Konfiguration, die man testen will. In der Anleitung gibt es zwar immer ein paar Fehlerszenarien, jedoch sind diese bei weitem nicht vollständig und sind auch nicht für die Fehler der Tests, sondern die Fehler der eigentlichen Konfiguration gedacht. Weiterführende Hilfe kann man in solchen Fällen von der Globus Gemeinschaft (Globus Mailinglisten) bekommen.
Für die Programmierung von Web Services ist das Tutorial von Borja Sotomayor ([Soto 05]) sehr hilfreich, hier wird ausführlich der Einstieg in die Welt der Web Services beschrieben. Allerdings ist das Tutorial noch nicht ganz komplett, es ist noch ein Kapitel über „Information Services“ geplant. Auch kommt das letzte Kapitel über die Sicherheit nicht ganz an die Qualität der vorherigen Teile heran.
Abschließend kann man sagen, wenn man die meist erfolgreichen Tests in Betracht zieht, dass die neue Version des Toolkits zwar einige große Änderungen erfahren, aber durchaus praxistauglich ist.

6.2 Weiterführende Arbeiten

Es wird festgestellt werden müssen, inwieweit die Fehler der Tests sich in der Praxis bemerkbar machen, oder ob sie sich überhaupt bemerkbar machen.
Für das Globus Toolkit wird weitere Software installiert werden müssen, um zum Beispiel Szenarien wie *Virtuelle Organisationen* aufzubauen. Aber auch bei der vorhandenen Installation kann man noch mehrere Anwendungsfälle, wie zum Beispiel einen„Striped Server“(siehe Kapitel 2.3.2) oder „Delegation“ (siehe Kapitel 2.3.1), testen.
Der Rechner „projekt-ext11“ stürzt nach unbestimmter Zeit ab. Mögliche Ursachen dafür waren nicht ersichtlich, werden aber wohl noch festgestellt werden müssen, wenn man mit dem Rechner von außen aus, also zum Beispiel per SSH, arbeiten will. Auch die für den RFT (siehe Kapitel 3.3.4) auf Rechner projekt-ext10 verwendete Datenbank PostgreSQL verweigerte ohne ersichtlichen Grund schon ein paar mal den Dienst. Das Neuanlegen der Datenbank behob zwar das Problem, die Ursache für das Versagen

wird aber noch festzustellen sein.
Auch die weitere Entwicklung des Globus Toolkits wird man im Auge behalten müssen, da noch nicht alle Bereiche vollständig sind, wie zum Beispiel die Dokumentation, und natürlich die Entwicklung des Globus Toolkits weiter voran getrieben werden wird.

6.3 Danksagung

Ich möchte mich bei meinen Betreuern Herrn Michael Schiffers und Herrn Nils Otto v. d. gentschen Felde von der LMU sowie Herrn Dr. Helmut Heller von dem LRZ für ihre freundliche Unterstützung bedanken. Auch Herrn Dr. Anton Frank vom LRZ, der nicht direkt mit der Betreuung meiner Arbeit beauftragt war, möchte ich für seine hilfreichen Tipps und seine Geduld danken.

Anhang A

Hardware der Rechner

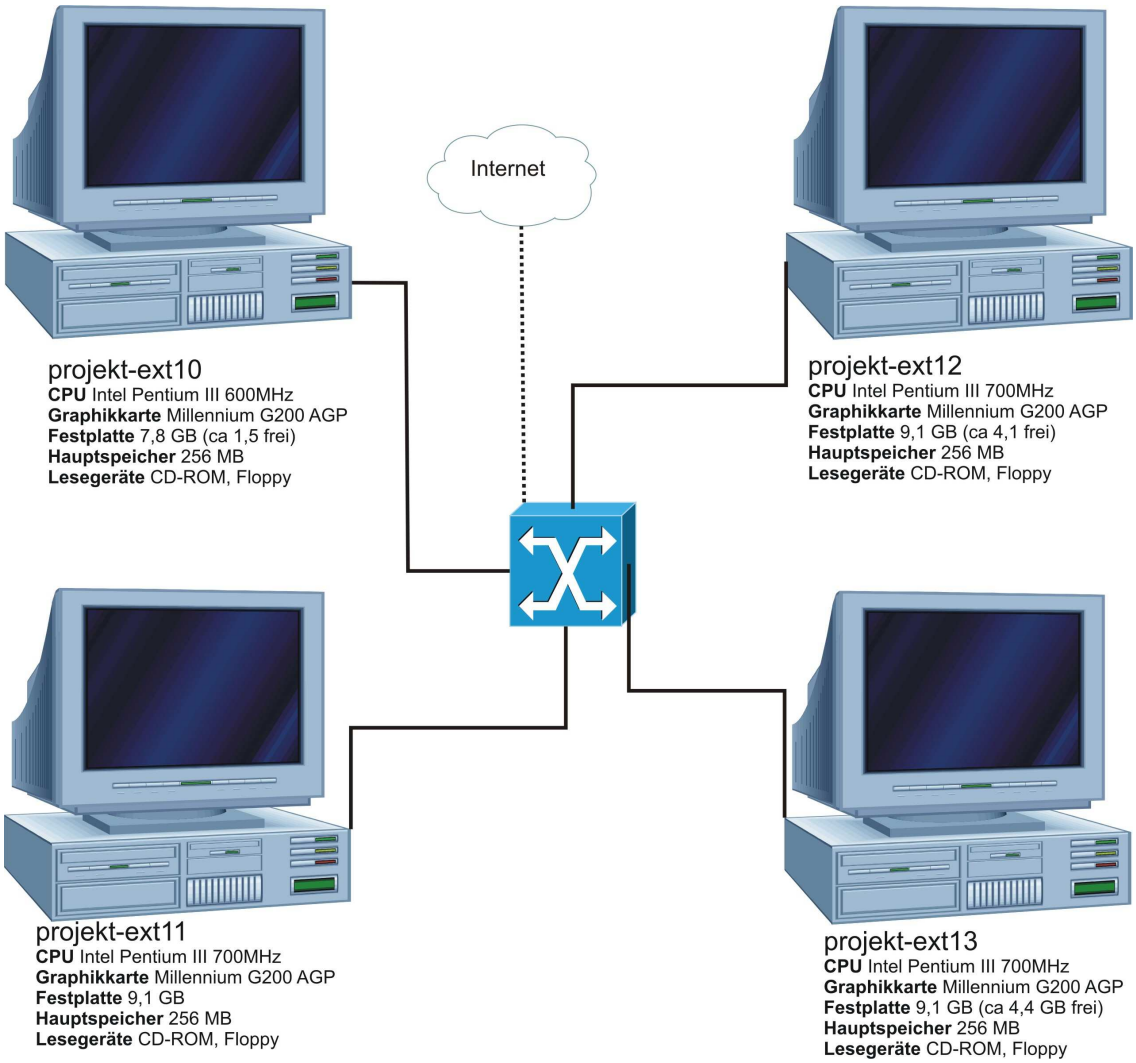

Abbildung A.1: Anordnung der Rechner und ihre Hardware

A.1 projekt-ext10

CPU Intel Pentium III 600MHz
Graphikkarte Millennium G200 AGP
Festplatte 7,8 GB (ca 1,5 frei)
Hauptspeicher 256 MB
Lesegeräte CD-ROM, Floppy

A.2 projekt-ext11

CPU Intel Pentium III 700MHz
Graphikkarte Millennium G200 AGP
Festplatte 9,1 GB
Hauptspeicher 256 MB
Lesegeräte CD-ROM, Floppy

A.3 projekt-ext12

CPU Intel Pentium III 700MHz
Graphikkarte Millennium G200 AGP
Festplatte 9,1 GB (ca 4,1 frei)
Hauptspeicher 256 MB
Lesegeräte CD-ROM, Floppy

A.4 projekt-ext13

CPU Intel Pentium III 700MHz
Graphikkarte Millennium G200 AGP
Festplatte 9,1 GB (ca 4,4 GB frei)
Hauptspeicher 256 MB
Lesegeräte CD-ROM, Floppy

Anhang B

Installierte Software

Es wurden auf allen Rechnern die gleichen Softwarepakete verwendet.

- Globus Toolkit Version 4.0.1 Link: http://www-unix.globus.org/toolkit/survey/index.php?download=gt4.0.1-all-source-installer.tar.gz
- Java J2SE Software Development Kit 1.5.0_04 Link:http://java.sun.com/j2se/1.5.0/download.jsp
- Apache Ant 1.6.2 Link:http://archive.apache.org/dist/ant/source/apache-ant-1.6.2-src.zip
- Apache Ant 1.6.5 Link:http://archive.apache.org/dist/ant/source/apache-ant-1.6.5-src.zip, installiert auf gridprojekt3
- sudo 1.6.8p9 http://www.courtesan.com/sudo/dist/sudo-1.6.8p9.tar.gz
- zlip 1.2.3 http://www.zlib.net/zlib-1.2.3.tar.gz
- ntp 4.2.0 http://ntp.isc.org/Main/DownloadViaHTTP?file=ntp4/ntp-4.2.0.tar.gz
- SuSe Linux 8.2 Installiert von CD.

Die restliche hier aufgeführte Software stammt von der SuSe Linux 8.2 Installations-Cd.

- GNU tar 1.13.25
- GNU make 3.80
- PostgreSQL 7.3.2 – Zu beachten ist hier, dass auf grid0 das Verzeichnis der Datenbank „/home/postgres/data“ ist, auf den anderen das Standardverzeichnis „/var/lib/pgsql/data“. Die globale Variable $PGDATA zeigt auf jedem Rechner den Pfad zur Datenbank korrekt an.

Glossar

$GLOBUS_LOCATION Variable, die auf das Installationsverzeichnis des Globus Toolkits zeigt. Wichtig für viele Programme die das Toolkit benutzen, sowie für das Toolkit selbst.

ACL Access Control List

Credential Kombination aus Zertifikat und geheimen Schlüssel

Deployment Prozess des Einbindens eines Dienstes in einen Container

FQDN Full Qualified Domain Name

GAR Grid-Archive, enthält alle für das Deployment eines Dienstes notwendigen Dateien und Informationen

IETF Internet Engineering Task Force

JDBC JAVA Data Base Connectivity - mit X/Open SQL CLI ausgestattete Datenbankschnittstelle in Java mit flexibler, dynamischer Schnittstellenverwaltung (siehe sun.com)

OGSI Open Grid Services Infrastucture

PKI Public Key Infrastructure

portType Die durch das WSDL-Interface bereitgestellten Operationen

RFC Request for Comment, Standards der IETF

RLS Replica Location Service

SAML Security Assertion Markup Language

Security Assertion Markup Language Sprache, die zum Ziel hat, den Austausch von Autorisations- und Authentifizierungs-Informationen zu erleichtern.

SOAP Simple Object Access Protokoll, ein vom W3C definierter Standard zum Austausch von Informationen im XML-Format in einer verteilten Umgebung. Für gewöhnlich baut SOAP auf http auf.

Stub Der für die Interpretation und Generierung von SOAP-Nachrichten zuständige Teil eines Web Services

WAN Wide Area Network

WSDL Web Service Description Language

WSRF Web Services Resource Framework

Literaturverzeichnis

[Fost 05] FOSTER, IAN: *GT4 Primer*. 2005, http://www.globus.org/toolkit/docs/4.0/key/GT4_Primer_0.6.pdf .

[glo 05] *GT4 Admin Guide*. World Wide Web page, 2005, http://www.globus.org/toolkit/docs/4.0/admin/docbook/ .

[IF 01] IAN FOSTER, CARL KESSELMAN, STEVEN TUECKE: *The Anatomy of the Grid: Enabling Scalable Virtual Organizations*. Int. J. High Perform. Comput. Appl., 15(3):200–222, 2001.

[Soto 05] SOTOMAYOR, BORJA: *The Globus Toolkit 4 Programmer's Tutorial*, November 2004-2005, http://gdp.globus.org/gt4-tutorial/ .

[WBMH 03] WU, C. ERIC, HARIHARAN BALAKRISHNAN, BIJU T. MANIAMPADAVATHU und WILLIAM P. HORN: *Grid-Enabled Manageability Services for Linux Resources*. IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, 2003.