Globus Grid Toolkit V3

Hanno Tersteegen, Oliver Zilken

Verteilte und Parallele Systeme I
Fachbereich Angewandte Informatik
Fachhochschule Bonn-Rhein-Sieg

16. Juni 2003

Prof. Dr. Rudolf Berrendorf

Inhalt

1. Grid Computing

1.1. Einführung

Der Begriff des Grid Computing wurde durch die Assoziation mit dem elektrischen Stromnetz (Power Grid) geprägt. Die ursprüngliche Motivation für Grid Computing war, dass Rechenkapazität in gleicher Weise universell und transparent zur Verfügung gestellt werden soll, wie dies für elektrische Energie der Fall ist.
Neuerdings wird Grid Computing umfassender definiert, als die sichere, flexible, koordinierte und gemeinsame Nutzung von Ressourcen innerhalb virtueller Organisationen. Als Ressourcen sind hier neben Rechenkapazität auch Daten und Geräte, sowie im weitergehenden Sinne, Personen zu verstehen. Virtuelle Organisationen bestehen dabei aus einer Menge von (global) verteilten Institutionen.
Beispiele für virtuelle Organisationen sind Application Service Provider oder Konsortien von Firmen die gemeinsam ein neues Produkt entwickeln (z.B. Auto oder Flugzeug) oder auch internationale wissenschaftliche Kooperationsprojekte. Solche kooperativen Problemlösungsumgebungen werden zukünftig in vielen Bereichen von Industrie und Wissenschaft eine signifikante Rolle spielen und damit einen wesentlichen Einfluss auf die Entwicklung globaler Informationsinfrastrukturen, insbesondere dem Internet, ausüben.

1.2 Vorgeschichte

Im Jahre 1965 träumten Multics-Designer am MIT von der Zukunft: „a computer facility operating like a power company or water company“. 1968 veröffentlichen Licklider und Taylor „The Computer as a Communications Device“, darin erinnern einige Szenarien an einen Grid. In den folgenden Jahrzehnten fand eine Entwicklung verschiedenster Ansätze für verteilte Systeme statt.

1. Generation
Jetzt, im Nachhinein wissen wir, dass Metacomputing die Anfänge des Grid-Computing waren. Begonnen hat die Entwicklung hierzu in den USA mit dem Vernetzen von Supercomputern, damals nannte man dies „Metacomputing“. Die Wurzeln liegen dabei im CASA –Projekt, eines der US Gigabit-Tests in den späten Achtzigern. [Catlett92]
1995 wurden bei dem Projekt FAFNER große Zahlen parallel faktorisiert, dem System reichten normale Workstations mit 4MByte Speicher. Da nach der Initialisierung der einzelnen Rechner keine bzw. kaum Kommunikation nötig war, benötigte man kein Hochleistungsnetz.
Im Gegensatz zum FAFNER-Projekt wurden beim I-WAY-Projekt 17 Rechenzentren (also Großcomputer) mit einem Netzwerk auf ATM-Basis miteinander verknüpft. Das Netzwerk bestand aus 10 Einzelnetzwerken mit variabler Bandbreite und Protokollen, wobei verschiedene Routing und Switching-Technologien verwendet wurden. [Evo]

Probleme der 1. Generation
Beiden Projekten fehlte es an der Skalierbarkeit, FAFNER benötigte eine große menschliche Einflussnahme für die Weiterverarbeitung der Ergebnisse. I-.WAY war durch das Design der Komponenten die I-POP und I-Soft unterstützten beschränkt. [Evo]

2. Generation
Bei der 2. Generation wurde der Fokus mehr auf die Middleware gelegt, um große Datenmengen bearbeiten zu können
Drei Hauptziele wurden dabei verfolgt:
Die Heterogenität: Ein Grid beinhaltet eine große Anzahl heterogener Ressourcen. Unter Umständen umspannt es eine Zahl von administrativen Domains, die über den ganzen Erdball verteilt sein können. Von daher ist Homogenität unmöglich. Nun galt es diese strukturellen Vorgaben so zu nutzen, dass aus ihnen ein vernünftig funktionierendes Grid entstand.
Ein Grid muss auf jedenfall sehr skalierbar sein, da es von einigen wenigen bis zu Millionen von Ressourcen wachsen kann. Anwendungen die eine große Anzahl an, unter Umständen geographische verteilten, Ressourcen benötigen, müssen größere Latenzzeiten verkraften können. Die einzelnen Ressourcen müssen auch eindeutig adressierbar sein und die Sicherheitsaspekte dürfen nun nicht mehr außer Acht gelassen werden.
Der dritte Punkt, der bei der 2. Generation beachtet werden sollte, war die Anpassungsfähigkeit. Da ein Ressourcenausfall in einem Grid von normaler Natur ist, muss es Ressourcenmanager geben oder die Applikationen müssen dynamisch arbeiten, damit das Maximum an Performance aus dem System herausgeholt werden kann. [Evo]

3. Generation
In der 3. Generation wird nun in die Richtung „distributed global collaboration“ eingelenkt.
Man versucht nun die Interfaces zu vereinheitlichen , außerdem soll mehr Automatisierung und Selbst-Organisation in den Grids stattfinden. Die Metadaten finden nun auch mehr Beachtung, die Ansätze dabei sind die Web-Services und Multiagentensysteme. [Fox]

1.3 Definition Grid Computing

Unter einem Computer Grid versteht man verteilte, möglicherweise heterogene Ressourcen, welche zusammen genutzt werden können, um rechenintensive Applikationen auszuführen. Ein Grid wird auch Metacomputer genannt (s. Vorgeschichte).
Die Charakteristika eines Grids sieht wie folgt aus:
Ein Grid hat keine zentrale Kontrolleinheit. Es stellt gewisse Dienste wie z.B. Sicherheit zur Verfügung, welche die Identifizierung, Authentifizierung und Autorisierung von Benutzern regelt. Weiter muss es so etwas wie ein Abrechnungssystem geben, um die Kosten und die damit verbundenen Zahlungen zu bestimmen.
Ein Grid nutzt Standardprotokolle, wie sie zum Beispiel heute die Internetprotokolle darstellen und es garantiert gewisse Qualitäten, wie die maximale Dauer der Antwortzeit und es macht Aussagen über den möglichen Durchsatz, also die maximale Menge der Daten die pro Zeiteinheit übertragen werden können.

1.4 Unterschied zwischen Grid-, Cluster – und Webcomputing

Cluster Computing konzentriert sich auf (meistens) homogene, verteilte Systeme, die in einer Domain verwaltet werden. Der Fokus liegt auf dem „high – throughput computing“. D.h. der Informationsaustausch, also die zur Verfügung stehende Bandbreite, ist äußerst wichtig für das System
Web Computing kann irgendwelche Kombinationen von Ressourcen enthalten, welche Namensserver, Suchmaschinen usw. unterstützen.

Grid Computing konzentriert sich auf heterogene verteilte Systeme, die nicht von einer zentralen Stelle verwaltet werden. Der Fokus liegt auf dem „high – performance computing“ bei gemeinsamer Ressourcennutzung. [unizh]

1.5 Gründe für Grid Computing

Heutige Applikationen wie wissenschaftliche Simulationen, zum Beispiel in der Chemie und der Physik, Diagnosen in der Medizin, Wettervorhersagen und das Betreuen von Aktienportfolios brauchen sehr viel Rechenleistung. Diese können mit heutigen Computern nur ungenügend gelöst werden.

Netzwerk vs. Computer Performance [unizh]

Zeit:	1986-2000	2001-2010
Entwicklung der Leistung:	Computer: x 500 Netzwerke: x 340.000	Computer: x 60 Netzwerke: x 4000

Die Netzwerkperformance verdoppelt sich im Moment alle 9, die Performance der Rechner allerdings nur alle 18 Monate. [Moore] Somit wird es mit der immer höheren Leistungsfähigkeit der Netzwerken möglich, die Berechnungen auf mehrere Ressourcen zu verteilen und so eine viel höhere Rechenkapazität zu erhalten. Auch ist es möglich, eine höhere Auslastung der einzelnen Computerressourcen zu erreichen, indem man nicht benötigte Kapazitäten anderen zur Verfügung stellen kann.

1.6 Nutzerstrukturen in einem Grid

Ein großes Interessen an Grid Computern haben sicherlich Unternehmen. Können sie doch ihre Betriebskosten senken, indem sie sonst leerlaufende Rechenkapazität besser ausnutzen.
Eine 2. Zielgruppe sind Mitarbeiter wissenschaftlicher Einrichtungen, bei denen große Datenmengen verarbeitet werden müssen. So zum Beispiel Wissenschafler die auf großen Gendatenbanken arbeiten.
Dabei ist zu beachten, das eine gewisse Menge an Know-How und auch ein finanzieller Spielraum nötig ist, um ein Grid Projekt zu Implementieren.

1.7 Grid Applikationen

Folgende 6 Arten von Applikationen werden heute unterschieden:

1. Distributed Supercomputing
Klasse von Applikationen dessen Anforderungen nur durch mehrere Ressourcen mit hoher Kapazität befriedigt werden können. Wird auch verteilter Supercomputer genannt, dieser bietet jedoch mehr Rechenkapazität als „nur“ ein Supercomputer. Mit einem solchen Grid ist es möglich komplexe Analysen durchzuführen. Applikationen auf Supercomputern brauchen in der Regel große Mengen an Ressourcen. Durch die eher seltene Benutzung dieser Anwendungen ist die Anschaffung eines Supercomputers eigens für diese Anwendung oft nicht rentabel genug.
Bsp. Astronomie (Sky Project), Klimaforschnung, Kampfsimulationen (SFExpress)

2. High – Throughput Computing
Ein solches Grid koordiniert die Nutzung von ungenutzten Ressourcen in schwach gekoppelten Systemen. Hier geht es um viele kleine, häufig auftretende Rechenaufgaben.
Bsp.: RSA keycracking, seti@home (Auffinden von außerirdischem Leben)

3. On – Demand Computing
Beim On-Demand Computimg werden Gridkapazitäten genutzt um Kurzzeit - Anforderungen zu befriedigen. Hier wird die Ressourcenteilung besonders aus Kostengründen realisiert. Die Ressourcen können aus Berechnungen, Software oder Daten bestehen.
Bsp.: Meteorologie

4. Data–Intensive Computing
Konzentriert sich auf die Synthetisierung von neuen Informationen aus großen verteilten Datenbeständen. Diese Applikationen gehen in die Richtung „Die Nadel im Heuhaufen finden".
Bsp: NILE (distributed system for high energy physics)

5. Collaborative Computing
Collaborative Computing ermöglicht die bessere Koordination von Mensch–zu–Mensch Interaktionen. Der Fokus liegt hier auf Echtzeit-Interaktion und virtuelle Realität.
Bsp: NICE (System für virtuelle Zusammenarbeit von Kindern)

6. Teleimmersion
Teleimmersion bezeichnet Systeme, bei denen die virtuelle Zusammenarbeit gefördert wird, sie werden auch "collaborative virtual enviroments" genannt.
Heute wird Teleimmersion vor allem in der interaktiven wissenschaftlichen Visualisierung, in der Ausbildung und im Training, in der Kunst, im Industriellen Design und in der Telepresence verwendet.

2. Die Grundlagen des Globus Grid Toolkit V3

2.1. Die Open Grid Service Architecture (OGSA)

Die Motivation der *Open Grid Service Architecture* (im weiteren OGSA genannt) war die Suche nach einer möglichst kompletten Einbindung von *Web Services* in die offenen Protokolle des Grid Computing [OGSA]. Um diese Motivation zu verstehen, ist es notwendig, den Inhalt der *Web Services* genauer zu betrachten.
Exkurs: Web Services
Der Begriff *Web Services* beschreibt im Allgemeinen eine Menge von Funktionen zur Kommunikation in heterogenen verteilten Systemen [OGSA]. Die Funktionen unterscheiden sich jedoch deutlich von den ‚klassischen' Paradigmen wie DCE, CORBA oder Java RMI.
Die Besonderheit der *Web Services* ist, das sie auf einfachen, aber weit verbreiten Standards wie zum Beispiel XML aufsetzen. Sie sind deshalb sowohl von der Programmiersprache, mit der sie implementiert werden, als auch von ihrer System-Plattform, unabhängig.
Die *Web Services* Standards sind hauptsächlich durch das W3C definiert.
Dadurch bilden sie eine stabile Basis, auf die viele Groß-Projekte der führenden IT - Industrie aufbauen. Einige Beispiele sind u. a. Microsoft (.NET), IBM (Dynamic EBusiness) und Sun (SunONE).

Zwei Begriffe haben in letzter Zeit die Bedeutung des Begriffs *Web Services* geprägt: *SOAP* und *WSDL*.
Das *simple Object Access Protokoll* (**SOAP**) bietet Möglichkeiten zum Transfer zwischen einer Dienstanbietenden und einer Anfragenden Komponente. Die zu versendenden Daten werden dabei in einen XML - konformen Umschlag eingebettet. Dieser Umschlag kann 2 verschiedene Arten von Botschaften enthalten: RPC-kompatible Nachrichten oder reine Datentransfers [heise].
Diese XML - Pakete können dann mittels http, FTP, JMS oder ähnlichem über das Netz verschickt werden.
Die Web *Service Description Language* (**WSDL**) wird benutzt, um detaillierte Informationen zu einem bestimmtem Dienst festzulegen.
Auf diese in XML formulierten Informationen können sich Applikationen beziehen, um genauere Angaben und benötigte Parameter des angeforderten Dienstes zu bekommen.
Ein *WSDL*-Dokument ist im Allgemeinen sehr komplex und enthält typischerweise Elemente wie Importvorgaben, Typenbeschreibungen, Schemas, Nachrichten, Ports und spezielle Bindungen [heise].
Aus dieser Verknüpfung der schon vorhandenen Grid – Computing Protokolle mit diesen Web Service Protokollen ergeben sich in der Praxis einige Vorteile.
Die Ressourcen lassen sich einfacher Virtualisieren.
Außerdem bietet sich so eine unabhängige Vereinheitlichung von Ressourcen, Diensten und Informationen. Dies bedeutet eine Schaffung von standardisierten Schnittstellen und Verhalten für ein effektives Management von verteilten Systemen.
Zudem wird eine Unterstützung von praxisnahen Dienst – Spezifikationen wie allgemeines Ressourcenmanagement, Datenbankmanagementsystemen, Arbeitsabläufen (workflow), Sicherheitsdiensten, etc. angeboten.
Als letzter großer Vorteil wäre die Tatsache aufzuführen, das mit den neuen Protokollen die volle Kompatibilität mit allen schon vorhandenen Protokollen des Globus Grid Toolkits gesichert werden kann.

2.2. Die Open Grid Service Infrastructure (OGSI)

Die Open Grid Service Infrastructure definiert Mechanismen zum Erschaffen, Managen oder zum Austausch von Information zwischen zwei Instanzen von *Web Services* [OGSI].
Es definiert im Allgemeinen ein Modell, das Web Services genau beschreibt und strukturiert.
Es bildet eine Erweiterung zur offiziellen WSDL – Spezifikation des W3C. Es wurde zusammen mit Entwicklern des W3C erarbeitet und bildet nun einen neuen Grid Standard namens **GSDL** [OGSI].
Mit diesen grundlegenden, WSDL – basierten Schnittstellen und Verhaltens-Maßregeln eines Grid Services kann eine uneingeschränkte Interoperabilität sichergestellt werden
Da ein Grid aber nicht nur aus persistenten Diensten besteht, mussten auch transiente, also unbeständige Dienste berücksichtigt werden. Dieses wurde durch Interfaces realisiert, die die Zustände von verteilten Aktivitäten überwachen und manipulieren können
Dadurch können auch temporäre Instanzen eines Services dynamisch adressiert und dynamisch erstellt oder zerstört werden.
Typische Beispiele von solch transienten Instanzen sind zum Beispiel dynamische Arbeitsabläufe, Video Konferenzen oder verteilte Datenanalysen.
Die Unterstützung dieser transienten Abläufe war höchst prioritär, da sie einen kritischen Punkt bei der Unterstützung von virtuellen Organisationen darstellt.
In der Praxis ist immer ein großer Teil der *OGSA* und des gesamten Grids für dieses Management von Dienst – Instanzen beschäftigt.

Version 1.0 der *OGSI*- Spezifikation ist erst vor kurzem fertigestellt worden und wurde am 5.April 2003 dem Globus Grid Forum zur Evaluation und Kommentierung frei gegeben. Mit einer offiziellen Veröffentlichung wird noch Ende Juni 2003 gerechnet [OGSA].

3. Das Globus Grid Toolkit V3

3.1. Einführung

Das Globus Grid Toolkit V3 baut fast vollständig auf der oben genannten *OGSA* und der *OGSI* auf. Dies ermöglicht das Starten von Grid - Anwendungen auf einer Vielzahl von verschiedenen, plattformunabhängigen Hosting Umgebungen [GT3]:
Standalone / Embedded Hosting Umgebungen, Servlet Engine Hosting Umgebungen, Enterprise Bean Hosting Umgebungen und Virtual Hosting Umgebung.
Diese 4 verschiedenen Arten, eine Grid – Anwendung aufzusetzen, werden nun genauer erläutert:

a) Standalone / Embedded Hosting Umgebung
Hierbei werden die Grid – Anwendungen auf einem einfachen *SOAP*/http Server gestartet. Optional kann die Anwendung auch in eine handelsübliche J2SE Java Virtual Machine eingebettet werden.
Diese Methode wird häufig für die Implementierung von Test Frameworks eingesetzt. Der Vorteil dieser Methode besteht darin, dass sich sämtliche relevanten Daten in einem Klassenlader befinden und einfach über das http 1.0 Protokoll angesprochen werden können.

b) Servlet Engine Hosting Umgebung
Hierbei wird ein standardisierter Servlet Dispatcher bereitgestellt. Dieser kann auf jedem Servlet - API kompatiblen Web Server eingesetzt werden. Als Besonderheit befindet sich hier der *OGSI* Container in einer einzigen Anwendung innerhalb einer Servlet Engine und hat einen separaten Klassenlader. Ein beispielhafter, schon als stabil laufend getesteter Anwendungsfall hierfür ist die Einbettung in einem Tomcat 4.0.6 Server.

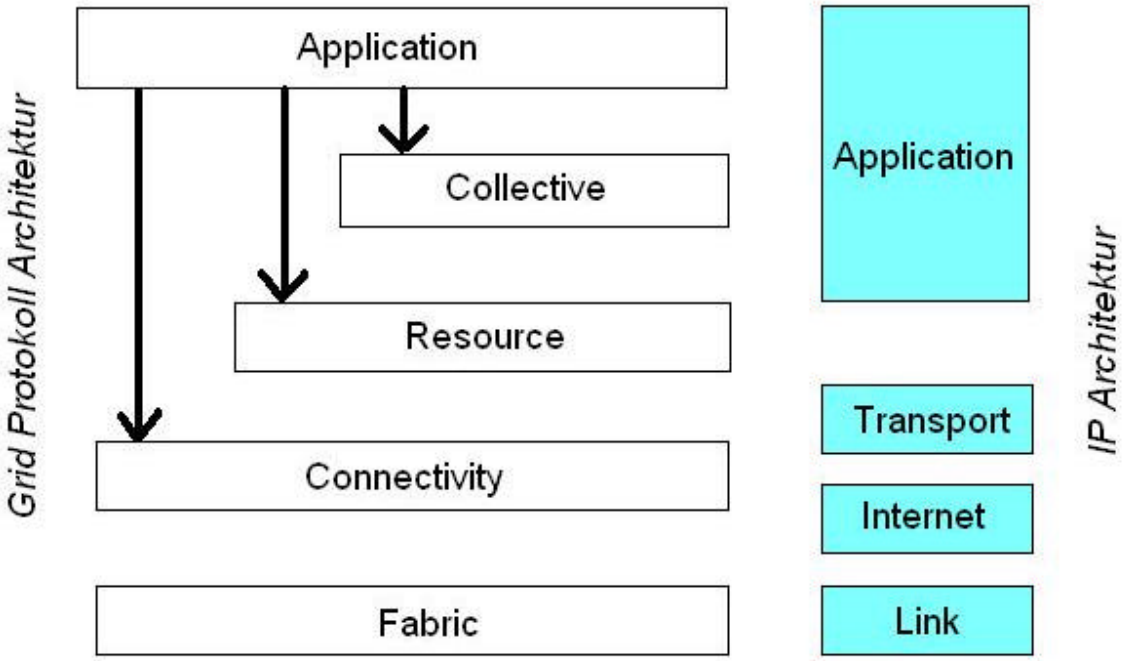
c) Enterprise Bean Hosting Umgebung
Diese neue Funktionalität erlaubt den Einsatz von Java Enterprise Beans in Grid Applikationen. Es werden sowohl Zustandsbehaftete wie auch zustandslose Session Beans und Entity Beans unterstützt.
Es wurden schon 2 Entwicklungsumgebung für diesen Einsatz von Globus zertifiziert: IBM WebSphere und das Open - Source Projekt JBoss.

d) Virtual Hosting Umgebung
Diese Umgebung stellt einen virtuellen Dienst zu Verfügung, um andere Hosting Umgebungen von einem (logisch oder physikalisch) entfernten Ort zu steuern.
Diese Art der Infrastruktur ist entscheidend für den dynamischen Aufbau von virtuellen Organisationen (*VO's*).

3.2. Architektur des GT3

3.2.1. Protokoll – Architektur

Dem GT3 liegt eine 5-Schichten- Architektur zugrunde, die dem IP- Schichtenmodell ähnelt [GT3]:



Diese einzelnen, hierarchisch geordneten Schichten werden nun erläutert:

3.2.1.1. Fabric Schicht

Die *Fabric Schicht* regelt den Zugriff auf lokale Ressourcen.
Mögliche Arten von Ressourcen wären Computerressourcen, Netzwerkressourcen, Sensoren, oder Speichersysteme.
Die angebotenen Funktionalitäten enthalten das Starten und Überwachen von Programmen, Statusinformationen, Holen und Speichern von Files, allgemeines Management (Reservation , Priorisierung).

3.2.1.2. Connectivity Schicht

Diese Schicht definiert Protokolle für verschiedene Kommunikations- und Authentifizierungsdienste.
Diese werden bei der Behandlung der Dienste – Architektur noch einmal genauer erläutert.

3.2.1.3. Ressource Schicht

Die *Ressource Schicht* regelt die sichere Verhandlung, das Initiieren,
Überwachen und die Kontrolle von gemeinsam benutzten Ressourcen. Es gibt Management - und Informationsprotokolle. Managementprotokolle werden gebraucht, um den Zugriff auf

Ressourcen zu regeln (*Quality of Service*). Informationsprotokolle werden zur Beschaffung von Informationen (*Reservation, Quality of Service*) über Ressourcen gebraucht.

3.2.1.4. Collective Schicht

Die *Collective Schicht* stellt Dienste und Protokolle für die Verwaltung globaler Zustände zur Verfügung. Die wichtigsten Dienste hierfür sind:

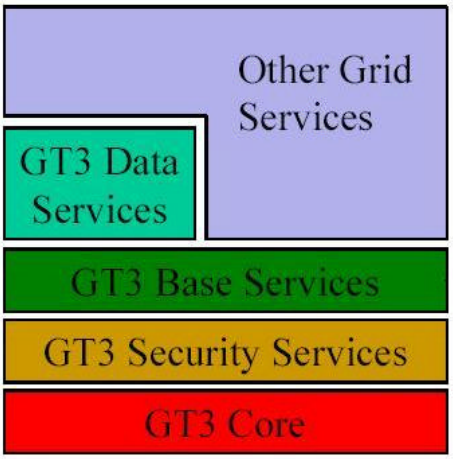
a) Directory Services: Feststellung der Existenz von Ressourcen
b) Scheduling and Brokering Services: Verwalten von Ressourcen
c) Monitoring and Diagnostics Services: Überwachen von Ressourcen
d) Workload Management Systems: Verteilung von Arbeitslast auf die Ressourcen
e) Accounting and payment services: Zahlung von Ressourcen.
f) Software Discovery Services
g) Detektion von Software in einem Grid.

3.2.1.5. Application Schicht

Auf der hierarchisch höchsten Schicht sind die Applikationen der User, die mit dem Grid interagieren sollen.

3.2.2. Dienste – Architektur

Die Protokoll – Architektur des GT3 wird mittels dieser Dienste implementiert. Die Dienste des GT3 sind ebenfalls hierarchisch aufeinander aufgebaut, um eine möglichst vollständige Kompatibilität und Interoperabilität zu gewährleisten [GT3].
Das Schichten – Modell der Dienste des Grids sieht wie folgt aus:



3.2.2.1. GT3 Core

Der Kern des GT3 enthält verschiedene Komponenten wie die Implementationen von grundlegenden Grid – Diensten, Notification Sink, Notification Description, Notification Source, Registration, Handle Resolver sowie Factories [GT3].

Die Basis – Dienste im Kern des GT3 wurden so designed, das sie die *OGSI* Version 1.0 Spezifikationen [OGSI] voll unterstützen und auch zum GT2 voll kompatibel sind.

Durch die Benutzung des Grid Service Container Framework wird die uneingeschränkte Portabilität zu beliebigen Hosting Umgebungen ermöglicht. Außerdem wurde dem Kern noch eine Entwicklungs- und Laufzeitumgebung beigegeben, damit es einfach möglich ist, neue Grid Dienste nach eigenen Wünschen beizufügen.

3.2.2.2. GT3 Security Services

Diese Schicht enthält als wichtigste Komponente den *Secure Conversation Service*, dem unter verschiedene Sicherungsmechanismen zur Verfügung stehen. Hier eine Auswahl der bedeutendsten:

a) TSL / SSL: Der *Transport Security Layer* bzw. der *Secure Socket Layer* sichern das neuartige *"httpg"* – Protokoll ab, bei dem es sich um ein *OGSI* – ermöglichendes, auf *http* basierendes Protokoll handelt [GT3].

b) SOAP Security: Dieser Sicherungsmechanismen basiert auf den offiziellen W3C – Standards *WS – Security, XML Encryption* und *XML Signature* [GT3].

c) X.509 Identitäts – Zertifikate: Diese von der *ITU* standardisierten Zertifikate regeln die Authentifizierung der User [ITU].

d) X.509 Proxy – Zertifikate: Diese Zertifikate sind notwenig zur Unterstützung von 2 wichtigen Kommunikations- und Authentifizierungsprotollen:

- *Single Sign on*: Ein User authentifiziert sich einmal und kann dann die verschiedenen Grid - Ressourcen und Dienste nutzen, ohne eine weitere Interaktion mit den Sicherungsmechanismen des Grids unterhalten zu müssen.
- *Delegation*: Ein Programm, welches von einem Anwender gestartet wird, erhält, automatisch und im gesamten Grid gültig, die Rechte die diesem User zugeordnet sind.

Dieses gegenüber dem GT2 verbessertes Sicherheitsmodell reduziert die Menge an privilegiertem Source Code, der von einem Dienst im Grid benötigt wird. Dadurch ist es einfacher, das GT3 in Firewall – geschützten Umgebungen einzusetzen.
Diese Funktionalitäten werden zusammengenommen als **GSI** (*Grid Service Infrastruture*) bezeichnet [GT3].

3.2.2.3. GT3 Base Services

Der Schicht der Basis – Dienste, die in einem Grid benötigt werden, wird hauptsächlich aus 4 Diensten gebildet:

a) Managed Job Service:

Dieser Dienst implementiert die **GRAM** Architektur in einer *OGSA* kompatiblen Form [GT3]. Das *GRAM* (*Grid Ressource Access and Management*).
Er ermöglicht das Ausführen, das Überwachen und das Beenden eines Jobs auf einer bestimmten Ressource. Dazu nimmt er Anfragen entgegen, reserviert die nötigen Ressourcen und startet den Job. Während der Abarbeitung informiert er den Client über den Status des Auftrages und aktualisiert den **MDS**, den Monitoring and Discovery Service, der eine überwachende, übergeordnete Instanz darstellt.
Der Zugriff auf den *GRAM* erfolgt über eine einheitliche Schnittstelle. Diese wird als *gatekeeper* bezeichnet. Realisiert wird der gatekeeper als Serverdienst, der über einen bestimmten Port angesprochen werden kann. Dieser Dienst muss in der Lage sein die lokalen Ressourcenmanagementmechanismen zu nutzen. Dadurch wird die Verbindung zu der ressourcenspezifischen Software hergestellt.
Ein Coallocator ist nötig, wenn ein Job über mehrere Ressourcen verteilt bearbeitet werden soll. Er ist dafür zuständig die Kommunikation mit den verschiedenen *GRAMs* zu koordinieren. Als Beispiel implementation liefert das Globus Toolkit den Coallocator *Dynamically Updated Request Online Coallocator* (**DUROC**) mit.

b) Index Service
Dieser Dienst wird benutzt, um einzelne Daten eines Dienstes zu sammeln, zu vereinigen oder anzufragen. Die Daten können so dynamisch auf Anfrage einer außenstehenden Instanz erstellt und manipuliert werden. Es werden zwei Arten von Eingaben unterschieden: Monitor Data Feeds und Poll Data Feeds.

c) Reliable File Transfer Service (RTFS)
Dieser Dienst baut auf dem **GridFTP** – Protokoll auf. Dieses ist ein erweitertes FTP – Protokoll, das den Anforderungen eines Grids angepasst ist. Der wesentlichste Unterschied besteht in der Integration der *GSI*. Weiterhin sind bei *GridFTP* mehrere Datenkanäle möglich. Dies ermöglicht parallele Datentransfers. Außerdem besteht die Möglichkeit, Dateien nur partiell zu transferieren. Das ist sehr nützlich bei besonders großen Dateien, die nur zu bestimmten Teilen benötigt werden. Zuletzt ist durch Server – zu – Server - Transfers die Möglichkeit zur Datenreplikation gegeben.
Der *RTFS* bietet so einen verlässlichen Kommunikationskanal zwischen zwei *GridFTP* Servern über eine dritte Instanz (sich selbst). Er bietet weiterhin eine gewisse Atomarität und ist nachvollziehbar und wiederherstellbar, um auch bei hochverfügbaren und zeitkritischen Transaktionen eine hohe Zuverlässlichkeit des Transfers zu gewährleisten.

d) File Transfer Service
Dieser Dienst biete ähnliche Funktionalitäten wie der *RTFS*, ihm fehlen nur die Atomaritäts- und Nachvollziehbarkeits- Möglichkeiten. Mit ihm lassen sich Transfers realisieren, die etwas schneller ablaufen können als beim *RFTS*.

3.2.2.4. GT3 Data Services
Die Funktionalitäten dieser Schicht können bei Bedarf auch durch selbst definierte Dienste in der nächsthöheren Schicht übernommen werden.
Diese Schicht wird hauptsächlich durch das **Replica Management** bestimmt.
Die Gründe für die Notwendigkeit dieser Funktion liegen darin begründet, dass es bei komplexen Anwendungen zu sehr großen Datenmengen kommen kann. Diese können in der Praxis nicht an einem physischen Ort gespeichert werde, sondern werden in beliebig viele Komponenten des Grids verteilt gespeichert.
Das Replica Management übernimmt die Verwaltung dieser verteilten Daten, in dem es festhält, welche Daten auf welchem Rechner zu einem bestimmten Zeitpunkt gespeichert sind. Außerdem ist es in diesem Dienst auch möglich festzulegen, welche Daten wo und für wen bereitgestellt werden können und soll. Dazu bedient sich das Replica Management dem **Globus Replica Katalog**. In diesem Verzeichnis werden die Verknüpfungen zwischen logischen Namen und einer oder mehrerer Kopien physikalisch gespeicherter Daten abgelegt. Im GT3 ist der *Globus Replica Katalog* durch einen *LDAP* – Server realisiert.

3.2.2.5. Other Grid Services
Diese Schicht kann andere, optionale oder auch selbst definierte Dienste enthalten, die dem Grid zur Verfügung gestellt werden.

3.3. Entwicklungsumgebungen für das GT3

Damit auch umfangreiche Projekt komfortabel zu realisieren sind, unterstützt das GT3 die Anbindung und Benutzung von vielen verschiedenen Entwicklungsumgebungen, um die einzelnen Komponenten der Grid – Anwendung zu erstellen.
Beispiele hierfür sind **Apache Axis** für die Erstellung der SOAP Engine, **Jakarta Tomcat** zum Bereitstellen einer Servlet Engine, **Junit** als alles erfassendes Test Framework, **Jakarta Log4J** als Debugging Tool und das **Globus Java Cog Kit** für Sicherheitskomponenten und GRAM Tooling.

3.4. Release Timeline für das GT3

Globus hatte für den Release des Grid Toolkit V3 im Januar 2003 folgende Timeline veröffentlicht [GT3]:

Die Versprechung, eine stabile und offiziell fertige Version im Juni 2003 herauszubringen, wird wahrscheinlich nicht in Erfüllung gehen. Zum Zeitpunkt der Erstellung dieses Aufsatzes (10.6.2003) war noch nicht einmal eine Beta-Version verfügbar. Ein genauerer Termin zur Veröffentlichung des Produktes ist nicht bekannt.

4. Zusammenfassung – Das Internet der Zukunft ???

Nach unserer Meinung sollte man die Macht des Grid Computing nicht überschätzen. Sehr oft wird diese neue Form des verteilten Rechnens als „Das Internet der Zukunft“ angepriesen. Zahlreich sind die kühnen Vorstellungen einiger Forscher, die in 20 – 30 Jahren den Bildschirm als einzige Schnittstelle zum Rechnen einer beliebigen Aufgabe sehen. Es wird sich vorgestellt, das man dem globalen Grid seinen Wunsch nennt, und dann in Ressourcen, die sich auf der ganzen Welt verteilt befinden, die Ergebnisse berechnet werden. Der Anwender hätte so gar keinen lokalen Rechenaufwand mehr. Computer würden nicht mehr gekauft, sondern Ressourcen würden gemietet und geleast. Der Computerbegriff würde sich zur dynamischen, kollaborativen Ansammlung von Speicher, Prozessoren, etc. ändern, und das Netzwerk würde immer mehr den Mittelpunkt der Computerwelt ausmachen.
Sicher geht die Entwicklung in diese Richtung, die Frage ist allerdings wie lange wird dies dauern und gibt es nicht vielleicht noch einen anderen Weg.

Aber trotz diesen Phantasien bestimmt sich das Durchsetzungsvermögen des Grid Computing durch die Akzeptanz in Forschung und Wirtschaft. Um zu verdeutlichen, dass das Grid Computing trotz der neuen, praktikablen Erweiterungen im GT3 wohl nur sehr selten über eine Existenz in Forschungsgruppen und Großrechnern hinauskommen wird, möchten wir einen Kommentar von Ralph Hülsenbusch anführen.

Ralph Hülsenbusch ist leitender Redakteur des renommierten Magazins für professionelle Informationstechnik **iX**. In der Ausgabe April 2003 schrieb er in seinem Leitartikel folgendes [IBM]:

„Abseits des Patriotismus sollte es allen eine Warnung sein, dass die Versuche, Application Service Providing (ASP) massenwirksam zu installieren, mehr oder weniger kläglich gescheitert sind. […] Standardapplikationen – im Sinne des fürs Tagesgeschäft üblichen - per Grid zu mieten, lohnt sich in der Regel kaum und birgt schwer durchschaubare Sicherheitsrisiken“

„Ob Grid vor Ungewünschtem zu schützen vermag, hängt vor allem von der Frage ab, wer die Fäden spinnt.“

5. Quellenverzeichnis

[Catlett92]
C. Catlett and L. Smarr, “Metacomputing,” Communications of the ACM, Juni 1992

[Evo]
„The Evolution of the Grid” David De Roure, Mark A. Baker, Nicholas R. Jennings and Nigel R. Shadbolt

[Foster]
Foster, Kesselman (eds.). „The Grid: Blueprint for a Future Computing Infrastructure“ . 1998: Morgan Kaufmann

[Fox]
Berman, Fox, Hey (eds.). „ Grid Computing: Making the Global Infrastructure a Reality“. 2003: Wiley. (*www.grid2002.org*)

[Globus]
www.globus.org
Die Seite zum Globus Grid Toolkit

[GT3]
http://twgrid.org/event/isgc2003/ISGC_pdf/ Globus_Tool_Kit_3.pdf
Globus Grid Toolkit V3

[heise]
http://www.heise.de/ix/artikel/2002/09/121/
Grid Computing

[IBM]
www.unicore.org/press/press_IBM.pdf
IBM Innovation Center für Kunden in Montpellier

[ITU]
http://www.webopedia.com/TERM/X/X_509.html
X.509 Internet Sicherheitsstandard der ITU

[IX]
iX Magazin für professionelle Informationstechnik, Ausgabe April 2003,
Heise Verlag, Hannover

[Moore]
Moore’s Law vs. storage improvements vs. optical improvements

[OGSA]
www.gridforum.org/ogsi-wg/drafts/ ogsa_draft2.9_2002-06-22.pdf
OGSA Standard

[OGSI]
http://www.org/ogsi-wg/drafts/draft-ggf-ogsi-gridservice-29_2003-04-05.pdf
OGSI Standard

[Tueb]
http://www-sr.informatik.uni-tuebingen.de/courses/GridSeminar_WS_02_03/
Seite zum Grid Seminar der Uni Tübingen

[unizh]
http://www.ifi.unizh.ch/richter/Classes/sem_cutting_edge/Slides_Grid.pdf.
Arbeit über Grid Computing