Grid Computing

Proseminar Softwarearchitekturen
Alina Lenz
Studiengang: B.Sc. Informatik - Mathematik
alina.lenz@student.uni-siegen.de

Abstract: Diese Seminararbeit beschäftigt sich mit der Funktionsweise des sogennanten Grid Computings an Hand eines Beispielkonzepts und einer dazu gehörenden Implementierung.

1 Einleitung und Motivation

Eines der bekanntesten Grid Computing Projekte war Anfang des 21. Jahrhunderts das *Seti@home* Projekt, bei dem Internetnutzen aufgerufen wurden, mit ihren Privatrechner an den Auswertungen von Radio-Signalen (mit dem Ziel, außerirdisches Leben zu finden) teilzuhaben. Doch wie genau funktionierte dieses Projekt?
Dieser Begriff, **Grid Computing**, wurde Ende des 20. Jahrhundert geprägt, als vor allem in der Forschung der Bedarf nach flexiblen Systemen mit hoher Rechenleistung groß war. Denn damals wurde auf Grund intensiver internationalen Zusammenarbeit der Wunsch nach solchen Systemen, vor allem für komplizierte Berechnungen, aber auch für das Bereitstellen von virtuellen Besprechungsräumen und Laboratorien groß.
Da gleichzeitig viele Rechner nur zu einem Bruchteil ihrer Leistung genutzt wurden, kam die Idee, Computer ähnlich wie beim *Cluster Computing* mit einander zu einem großen virtuellen Comuter zu vernetzen, wobei der Unterschied zum Clustering die Tatsache darstellt, dass die einzelnen Rechner räumlich weit voneinander getrennt sind. So entstand eine Netz-Architektur (*engl. Grid*), bei der die beteiligten Rechner (*Nodes*) Ressourcen (Speicherplatz und Rechenleistung u.a.) teilen, miteinander über offene Protokolle kommunizieren können und außerdem die Möglichkeit haben, gewisse Dienste (Services) anzubieten. Die Nodes werden dabei auf physikalischer Ebene über eine Software Schnittstelle, eine sog. *Middleware*, zu virtuellen Nodes zusammengefügt und strukturiert.

Diese Arbeit wird, um die Grundkonzepte einer Grid Computing Realisierung herauszuarbeiten, im Folgenden vor allem auf das vom Global Grid Forum (GGF) eingeführte Open Grid Service Architecture-Modell (OGSA) eingehen und im Anschluß eine Implementierung der OGSA, das Globus Toolkit, betrachten.

2 OGSA

Im Folgenden wird eine der ersten Grid-Architekturen näher betrachtet. Dazu werden die einzelnen Teile der Implementierung im Detail erörtert.

2.1 Architektur

Die OGSA ist eine von der GGF vornehmlich für Unternehmens- und Forschungs-Zwecke entwickelte Architektur, die vor allem den Informationsaustausch zwischen den am Grid beteiligten Rechnern in einem heterogenen Netz und die Verfügbarkeit der angebotenen Ressourcen sicherstellt. Ursprünglich wurde die OGSA als eine Abwandlung der damals aufkommenden *Web Services Architecture* von Ian Foster konzeptuell angedacht, wurde dann aber von der GGF verfeinert und veröffentlicht. Sie lässt sich grob in drei Kernbereiche aufteilen:

- die **OGSI** (Open Grid Services Infrastructure)
- die eigentliche **OGSI** (Open Grid Services Architecture)
- die **OGSA Schemata**

Die OGSI stellt eine beschreibende Schnittstelle sowie Management Schnittstellen zur Verfügung, in denen sämtliche angebotene Services beschrieben sind. Diese Beschreibungen erfolgen über eine *Standard Interface Definition Language* (SDL) wie, zum Beispiel, die *Web Service Description Language* (WSDL). Zu den durch die Services ausgeführten Aufgaben der OGSI gehören:

- *Grid Services beschreiben*, dazu zählen vor allem die Beschreibung der Funktion eines Services

- *Grid Services erzeugen*
- *Grid Services benennen*, die Namen sind für das Erreichen der jeweiligen Grid Services notwendig
- Die *Lebenszeit von Grid Services verwalten und überwachen*, denn verschiedene Grid Services haben je nach ihrer Anforderung kurze oder lange Lebenszeiten.
- *Grid Services überwachen*, darunter fällt die Überwachung von Zugriffsrechten und der Auslastung der verschiedenen Grid Services
- *Grid Services gruppieren*, zusammenhängende Services werden gruppiert, um den Verwaltungsausfwand zu vermindern. So können z.B. Aufträge direkt an eine Services-Gruppe übergeben werden.
- *Informationsaustausch zwischen den Grid Services*

Außerdem beinhaltet die OGSI Schnittstellen zur Auffindung von Grid Services. Dadurch brauchen nur die minimalen Mechanismen, die für den grundlegenden Betrieb eines Grids benötigt werden, in der OSGi bereitstellt werden, falls Nutzer im späteren Betrieb Spezialanforderungen benötigen, können diese leicht hinzuimplementiert werden. Die OGSI stellt also die unterste Schicht eines OGSA Sytems dar, auf der die höheren Modelle und OGSA Services aufsetzen können.
Die in der OGSI definierten Services reichen nur bedingt für den grundlegenden Betrieb aus, daher erweitert die OGSA das System mit höheren Funktionalitäten, dazu zählen die Folgenden:

- Die Auflösung und das Auffinden von Grid Services Namen in einem Grid Netz
- Das Managen und die Erstellung ganzer Grid Service Domains, also die Gruppierung äquivalenter Grid Services zu einer Service Domain
- Die Sicherheit der Grid Services, was besonders bei einem ans Internet angeschloßenem Grid in Hinsicht auf Datensicherheit sehr wichtig ist
- Die Rechteverwaltung der Grid Services, daher die Zugriffsberechtigungen der Clients auf die Services

- Das Logging und die Kommunikation der Grid Services, sowohl Serviceintern als auch zu den Clients
- Grid Services müssen fähig sein, auf Statusänderungen zu reagieren
- es muss eine genaue Überwachung möglich sein, um zu bestimmen welche Clients welche Ressource benutzen, damit eventuelle Abrechnungen gemacht werden können

Als oberste Schicht befinden sich die OGSA Schemata, die eine Beschreibungsprache der Services darstellen. Denn weder die OGSI noch die OGSA bieten genaue Definitionen zur konkreten Realisierung der Services oder zu deren Schnittstellen nach außen und untereinander.
Deshalb bieten die OGSA Schemata eine Beschreibung, die angibt, wie ein realisierter Grid nach außen hin aussieht und welche Schnittstellen er bereit stellt. Die Beschreibung nach außen wird meist in allgemeingültigen Formaten wie XML gehalten und die konkrete interne Beschreibung erfolgt dann in dem vorhin erwähnten WSDL.
Über diese Beschreibung ist es möglich, seine Grid Services zu formulieren, ohne sich über die später mit diesem Grid Service komunizierenden Grids Gedanken machen zu müssen. Daher muss man sich nicht schon im Vorfeld an Konventionen über Protokolle, Schnittstellen und Funktionen oder andere Restriktionen halten, wodurch der Programmierer in seiner Arbeit die größtmögliche Freiheit genießt. Es können also jegliche Art von Services neu definiert werden und ohne Probleme in ein bestehenede Grid Service System eingepflegt werden. [FK04]

2.2 Resourcen

Da der große Vorteil eines Grid-Systems die Erweiterung der Arbeitskapazitäten ist, müssen in diesem System vorhandene Ressourcen auch neu definiert werden. Vor allen da die Kapazitäten so weit von einander getrennt sind, müssen sie streng überwacht und verwaltet werden, da es sonst zu unnötigen Engpässen und ähnlichem kommen kann. Denn anders als bei normalen Systemen ist der Ressourcenbegriff in einer Grid Umgebung viel weiter gefasst. So ist nicht nur der Computer, Speicherplatz, etc eine Ressource, sondern auch die vom Grid angebotene Service Infrastruktur. Aus diesem Grund muss das Ressourcenmanagement viel großflächigere Vorraussetzungen erfüllen, um das ihm zu Grunde liegende System richtig unterstützen zu können:

- **Auftragseingabe**, dabei muss das Management sicherstellen, dass die Resource einen Auftrag erfüllt, sei es die Ausführung eines Programms oder einer Datenbankabfrage. Es muss sichergestellt werden, dass die Ressourcen ihren Anforderungen gerecht werden.
- **Workload Management**, ein Client hat die Möglichkeit, die für einen Auftrag benötigten Resourcen selbst zu bestimmen, diese Freiheit wird vom Workload Management zur Verfügung gestellt, da dieser die nötigen Ressourcen organisiert und zuweist.
- **On-Deman Access**, kurzfristige Zugriffsberechtigung auf Ressourcen
- **Co-Scheduling**, Bei vielen Aufträgen ist es notwendig, dass verschiedene Ressourcen miteinander kombiniert werden müssen. So muss bei Dateitransfers sowohl der Speicherort des Originalfiles sowie der neue Speicherort koordiniert werden
- **Resource Brokering** Da in einem Grid große Mengen an Ressoucen zur Verfügung stehen, braucht man einen "Zwischenhändler", der für jeden Auftrag die am besten passende Resource auswählt. Dies wird von *Broker* realisert, er teilt anhand bestimmter Regeln jedem Auftrag die benötigte Resource zu.

Um diese Vorraussetzungen erfüllen zu können, wird ein allgemeines Ressource Management Framework erstellt, dessen grundlegender Bestandteil

die sogenannten *Service Level Agreements* (SLA) sind. Diese SLA stellen eine Dienstleistungvereinbarung zwischen dem Cient und der Node dar. Die SLA bieten dem Client bei Anfrage auf eine Ressource die Aufschlüsselung, was die Ressource kann und inwiefern sie den Auftrag des Clients bearbeiten kann, ohne ihm aber einen genauen Einblick in die Arbeitsweise der Ressource zu geben. Eine einzige SLA kann diesen geraumen Aufwand an möglichen Ressourcen-Daten nicht verwalten, deshalb gliedern sich die SLA in drei spezialisierte Unterformen auf:

- **TSLA** (Task Service-Level Agreements): diese SLAs vereinbaren die Performance eines Auftrags. Sie werden dann erstellt, wenn eine Auftragsbeschreibung an ein Queing System gesendet wird
- **RSLA** (Resource Service-Level Agreements): diese SLAs vereinbaren, wer eine Ressource verwenden darf. Dabei spielt die Art der Verwendung keine Rolle.
- **BSLA**(Binding Service-Level Agreements): diese SLAs verinbaren, wer eine Ressource zu einem bestimmten Auftrag hinzufügen darf, zB. wenn eine RSLA eine bestimmte Netzwerkbandbreite an einem bestimmten Port zusagt.

Durch diese drei grundlegenden SLA Variantionen können auch komplizierte Ressourcen Anfragen sicher umgesetzt werden.

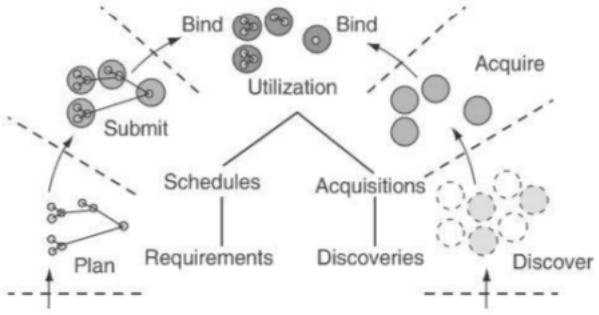

Abbildung 1: Recourcen Zuordnung. Quelle: [FK04, S.264]

Nachdem die Ressourcenzugriffe nun definiert wurden, müssen noch die Ressourcen selbst definiert werden, dies geschieht über eine Beschreibung

in der *Recource Description Language* (RDL). Die RDL bietet die Möglichkeit, für alle Ressourcen gewisse Eigenschaften festzulegen, um den Umgang mit diesen realisierbar zu machen. Die wichtigsten Eigenschaften sind hierbei:

- Beschreiben, welche Ressourcen vom Verbrauchen benötigt werden und wofür diese gebraucht werden
- Beschreiben, welche Ressource ein Anbieter überhaupt besitzt und unter welchen Vorraussetzungen diese benutz werden können
- Ressourcenattribute müssen gesetzt werden können, dazu gehört zum Beispiel die maximale Bandbreite einer Leitung.
- Lebenszeiten der Ressourcen müssen definiert werden können
- das System muss die Möglichkeit unterstützen, neue Resourcen durch das Recource Management System anzukündigen
- Ressourcenkombinationen müssen definiert werden können, um Basiskompositionen zu komplizierteren und mächtigeren Kompositionen zusammenfassen zu können.

All diese Aspekte müssen von einer RDL realisiert werden können, damit das Grid fehlerfrei funktioniert.
Derzeit sind zwei RDLs aktiv in Benutzung, die *Recource Specification Language* (RSL) und die *Common Information Model* (CIM). Die RSL wird derzeit im Globus Toolkit genutzt, welches im dritten Kapitel dieser Arbeit näher erläutert wird.
Bis jetzt hat das Ressourcenmanagement die Möglichkeit, Ressourcen zu definieren und die Zugriffe dazu zu verwalten, jedoch fehlt dem System noch die Möglichkeit, die SLAs selbst zu verwalten, daher ist ein elementarer Teil des Managementsystems das sogenannte *Task Management*, dieses übernimmt folgende Aufgaben:

- Eine SLA beenden, im Fehlerfall muss das Sytem fähig sein, eine SLA vorzeitg zu beenden, um eine Systemblockade zu verhindern
- Die Lebenszeit einer SLA verlängern
- SLAs verändern, um etwa Nutzungsbedingungen anzupassen

- Neue SLAs erzeugen

Die Implementierung der Service Vereinbarungen und der Datenerfassung des Ganzen findet in der OGSA in dem sogennanten *Service Negotiation and Acquision Protocol* (SNAP) statt. [FK04]

2.3 Monitoring

Aufgrund des weit verteilten Systems ist die Fehlersuche in einem Grid sehr aufwändig und sollte nach Möglichkeit vermieden werden. Daher ist eine intensive Überwachung des Systems notwendig, bei der alle notwendigen Informationen zu Vorgängen und Auftragsabläufen gesammelt werden, um eventuelle Fehler finden zu können. Diesen Vorgang der Überwachung und Beobachtung nennt man *Monitoring*.
Die Schwierigkeit beim Grid-Monitoring besteht darin, dass die Monitoringeinheit, ebenso wie jede andere Grid-Komponente auch, im Netz verteilt ist, es daher mehrere Instanzen von ihr gibt, die untereinander abgeglichen werden müssen, um Redundanzen zu vermeiden.
Weitere Aufgaben des Monitoring sind:

- Betriebssicherheit: im Falle eines Komponentenausfalls muss gewährleistet werden, dass der Rest des Systems stabil weiterlaufen kann
- Zeitnähe: die gesammelten Informationen müssen zeitnah zugreifbar sein
- Skalierbarkeit: Das System muss skalierbar sein, da die Monitoring Event Rate stark variieren kann. So können zum Beispiel tausend Events in einer Minute auftreten oder auch nur wenige im Jahr
- Sicherheit: Das System muss sichern, dass sensible Daten nicht von außerhalb abgegriffen werden können

Ein solches Monitoring ist in der OSGA in Form eines Modells realisiert. In diesem Modell ist eine *Notification Message* eine benannte Struktur mit einem Zeitstempel, die mit Informationen beschrieben werden kann. Diese Informationen gehören zu einer oder mehreren Ressourcen, wie etwa zum Arbeitsspeicher oder zum Prozessor. Die Komponente, die diese Notification Message aussendet, wird *Notification Source* genannt. Die Komponete,

die solche eine Nachricht bekommt und weiterverarbeitet, wird *Notification Sink* genannt. Zudem gibt es eine Komponente, die diese Informationen speichert und für spätere Zeitpunkte abrufbar macht, diese wird *OGSA Registry* genannt. Dieses Modell erlaubt sowohl eine *Request/Response Situation*, als auch eine *Subscription Situation* [FK04]

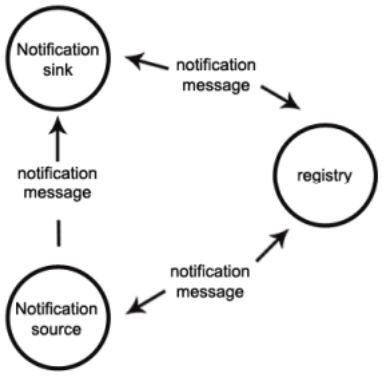

Abbildung 2: Schematische Darstellung des Monitoring in OGSA

2.4 Sicherheit

Mit hoher Wahrscheinlichkeit ist jede Grid Node an das Internet angeschlossen, womit die Frage nach der Sicherheit im Grid Netz groß wird. Deshalb müssen durch das System auch Vorkehrungen zur Abschottung von sensiblen Daten getroffen werden. Ein Grundkonzept sind hierbei die sogennannten *Virtual Organisations* (VO). Diese sind Zusammenschlüsse von Personen, Nodes und Services, in denen Handlungen und Aufträge nur für bestimmte Personen erlaubt sind, das sind also "lose" Zusammenschlüsse von Komponenten, in denen ein gewisses Maß an Sicherheit herrscht. Um Sicherheit garantieren zu können und spätere Probleme zu vermeiden, müssen zwischen den teilnehmenden Komponenten im Vorfeld einige Punkte des Sicherheitskonzeptes definiert und realisiert werden:

- Die Integration und Interaktion von verschiedenen Sicherheitssystemen muss gewährleistet werden können. Da die einzelnen VOs ver-

schiedenene Sicherheitskonzepte aufweisen können, muss im umliegenden Grid-Sicherheitssystem gewährleistet sein, dass die VO-übergreifende Kommunikation möglich ist und dass später hinzukommende VOs auch nachträglich ins Grid eingepflegt werden können.

- Es muss möglich sein, dynamische *Trust Domains* zu erstellen und zu verwalten. Das bedeutet, dass VOs in der Lage sein müssen, Vertrauensregeln zwischen sich selbst und den Benutzern oder Ressourcen zu erstellen und zu verwalten. Diese müssen auch dynamisch genug sein, um unerwartete neue Teilnehmer problemlos aufnehmen zu können und bei derem Ausscheiden diese Lücke wieder zu schließen.

- Benutzer und Ressourcen müssen dynamisch fähig sein, Services zu erstellen und dem Grid hinzuzufügen, diese neuen Services müssen dann auch den restlichen Teilnehmern und Komponenten zur Verfügung gestellt werden.

Der Großteil der Organisation, vor allem das Anlegen der Services, muss ohne jegliche Administration erlaubt sein, daher muss auch ein normaler Benutzer befähigt sein, solche Interaktionen vorzunehmen. Hier sieht man auch den größten Unterschied zu den gängigen Netzwerk-Sicherheitskonzepten, bei denen nur der *Amdministrator* weitläufige Rechte hat und die normalen Nutzer nur die für Ihre Arbeit notwendigen Aufträge erledigen dürfen. Grids benötigen aufgrund ihres verteilten Aufbaus ein User-basiertes Sicherheitssytem.
Bei der Implementierung des Grid-Sicherheitskonzepts ist wichtig, dass bestehende Systeme genutzt und nicht ungewollt ersetzt werden können. Deshalb gibt es auf *OSI* Schichtebene einige Anpassungen an die am Netzwerkverkehr beteiligten Schichten:

- Transportschicht: es muss einen Transport Mechanismus für Nachrichten geben, der die Konvertierung in ein anderes Format erlaubt. So kann zum Beispiel *HTTPS* verwendet werden, um sichere Nachrichten zu übertragen und *HTTP* für die Übertragung von unsicheren Nachrichten

- Nachrichtenschicht: es müssen Mechanismen zum Nachrichtenaustausch angeboten werden. Dabei ist wichtig, dass die Nachrichten bei

den Konversationspartnern auch ausgewertet und verarbeitet werden kann und vor allem: nur bei den Kommunikationspartnern findet eine Auswertung statt.

- Serviceschicht: in der Serviceschicht muss es möglich sein, Sicherheitsvorgaben für den Nachrichtentransport festzulegen. Zudem müssen verschiedene Sicherheitsvorgaben von unterschiedlichen Mitgliedern einer Konversation vergleichbar gemacht werden.

- Policy Layer: es muss sichergestellt werden, dass Namen für bestimmte Services, Benutzer etc. in verschiedenen Zusammenhängen unterschiedlich gedeuted werden können, abhängig davon, in welcher VO und in welchem Kontext sich die verwendete Ressource gerade befindet

All dies sind low-level Sicherheitsmaßnahmen, die sich in ein bestehendes Sicherheitskonzept schnell einpflegen lassen. Zusätzlich gibt es noch high-level Sicherheitsmaßnahmen, die von einem Grid Security Model erfüllt werden müssen:

- Authentifizierung: diese darf nicht technologiegebunden sein sondern plugin fähig sein, denn da es bei der Authentifizierung in einem bestehenden System unterschiedliche Ansätze gibt, darf diese nur als "Vermittlungsstelle" agieren.

- Delegation: es gibt Anwendungsfälle, bei denen das Recht, einen bestimmten Service nutzen zu dürfen, an Dritte weitergegeben werden müssen. Dazu muss sichergestellt sein, dass diese Funktion nicht nach Belieben und nur in dem für den Auftrag nötigen Rahmen genutzt werden kann.

- Einfacher Login: Es muss Nutzern möglich sein, sich einmal nur am Grid anzumelden und den Vorgang nicht für jeden einzelnen Service wiederholen zu müssen. Dieser Login muss dann eine Lebendauer zugewiesen bekommen und im Grid weiter gereicht werden.

- Berechtigungsnachweise: Diese Nachweise sind sowohl für die Delegierung von Rechten als auch für die Benutzeranmeldung von Bedeutung. Auch sie dürfen nur eine begrenzte Lebenszeit haben und dynamisch erneuerbar sein. Denn der Benutzer muss die Möglichkeit

haben, sich nach Ablauf einer Berechtigung eine neue zu holen, da er sonst angefangene Aufträge nicht zu Ende führen könnte.

- Autorisierung: jede Benutzung eines Grid Service muss auf einer vorhergehenden Authentifizierung beruhen und gesetzten Authorisierungsregeln unterliegen.

- Datenschutz: sowohl der Service Benutzer als auch der Service Anbieter muss die Möglichkeit haben, bestimmte Daten einsehen zu können, jedoch müssen die selben Daten vor Dritten geschütz bleiben.

- Diskretion: die Nachrichtentransportmeachanismen auf der Netzwerkebene müssen entsprechen abgesichert sein, damit Nachrichten nicht abgehört weren können. Das betrifft sowohl den Point-to-Point-Transport, wie auch den Store-and-Forward-Transport von Nachrichten.

- Nachrichten Integrität: die unerlaubte Veränderung von Nachrichten oder Dokumenten muss festgestellt werden können. Hierzu können die zahlreichen bekannten Prüfverfahren (z.B. Hash) verwendet werden.

- Richtlinien Austausch: der Richtlinien Austausch zwischen Service Anbietern und Service Nutzern muss dynamisch möglich sein, damit diese ihre Sicherheitsregeln, Fuktionalitäten, usw bei Bedarf untereinander austauschen können.

- Logging: Benutzerlogins und ähnliche Vorgänge am Grid-Netz müssen protokolliert werden. Diese Protokolle müssen sowohl zuverlässig als auch sicher aufgezeichnet werden.

- Sicherheitslevel: es muss möglich sein, die Sicherheitsvorkehrungen in einer VO abschätzen und bewerten zu können. Dazu gehören Informationen wie eine Virenscanner, Firewalls etc. Dies ist auch für Nutzer wichtig, damit sie wissen, in welcher VO sie welchen Service sicher in Anspruch nehmen können.

- Verwaltung: die Sicherheitseinstellungen, inklusive der high-level Sicherheit müssen verwaltbar sein.

- Firewall: Firewalls sind ein großes Problem bei Grids. Sie können den Datenverkehr stark einschränken, deshalb muss eine Grid in der Lage sein, die lokalen Firewalls auf seine Anforderungen anzupassen, ohne dabei die lokalen Sicherheitseinstellungen zu verletzen.

Hier sieht man noch eine graphische Aufbereitung der Sicherheitsstruktur eines Grids:

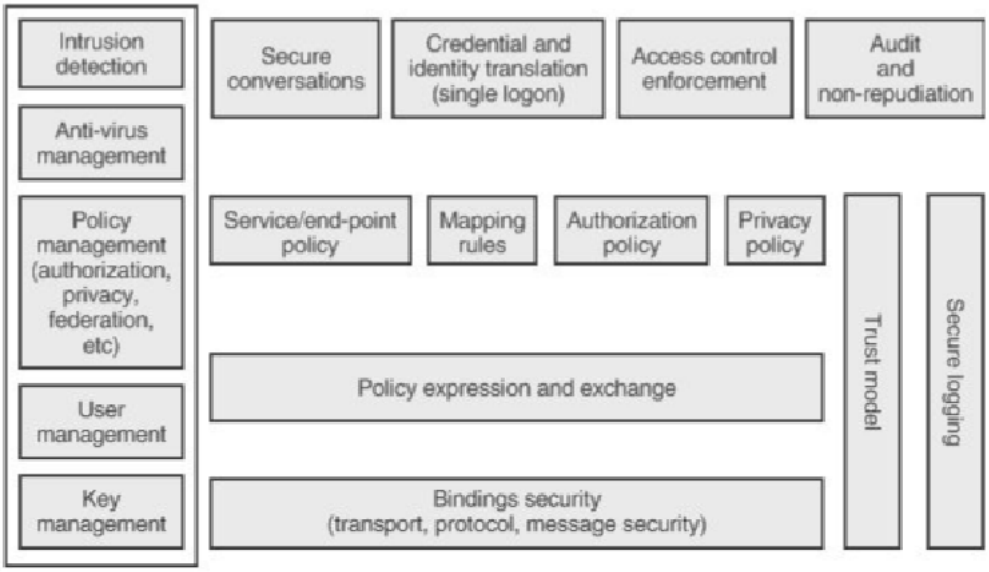

Abbildung 3: Das Grid Security Modell. Quelle: [FK04, S.375]

3 Globus Toolkit

3.1 Service - Übersicht

Das Globus Toolkit ist ein Open Source Toolkit der Globus Allience. Es stellt Schnitstellen un d Komponenten bereit, um ein Grid in Betrieb zu nehmen. Das Globus Toolkit deckt mit seinen Komponenten einen Großteil des von der OGSA geforderten Spektrums ab und spielt in dem damit aufgebauten Grid die Rolle der Middleware.[GT]

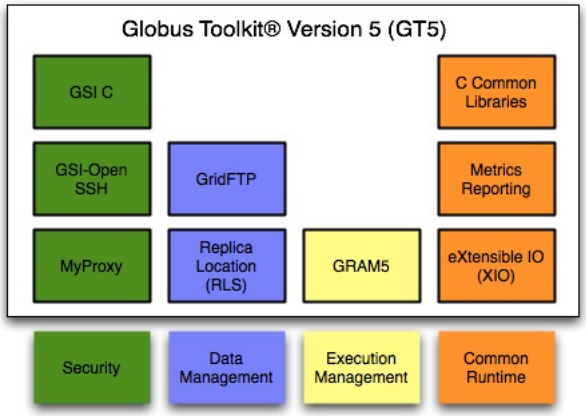

Abbildung 4: Grundlegendes Schema des Globus Toolkit. Quelle: [GT]

Für die Implementierung der Sicherheitskonzepte bietet das Toolkit gleich mehrere Konzepte, darunter auch MyProxy und GSI-Open SSh, dadurch lässt es dem Benutzer die größtmögliche Auswahl und damit auch den Komfort, sich sein System selbst zu konfigurieren und nicht an enge Vorgaben gebunden zu sein. Das Ressourcenamanagement basiert im Toolkit vor allem auf GridFTP, wobei bei dieser Anwendung der Schwerpunkt auf dem Dateinmanagement liegt. Ebenfalls ist GRAM5 als Realisierung der Monitoring Vorgaben vertreten, mit Hilfe von GRAM5 kann man die aktuellen Jobs sowohl verwalten als auch überwachen. Leider bietet es aber keine Scheduling Algorithmen an. Applikationen werden im Globus Toolkit über diverse Komponenten ausgeführt, ein großer Vertreter darunter ist eXtensible IO. [GT] [Wik01]

3.2 Implementierung des Sicherheitsmodells

Das Globus Toolkit bietet eine Implementierung des von der OGSA beschriebenen Sicherheitsmodells an. Die Implementierung durch das Globus Toolkit geschieht dabei in mehreren einzelnen Modulen, welche zusammen dem Sicherheitsmodell der OGSA entsprechen.
Zur Authentifizierung nutzt das Globus Toolkit eine sogennante "Public Key Cryptography". Dabei werden basierend auf dem X.509 Zertifikat Format sogennante CAs (Certificate Authority) erstellt. So besteht die Möglichkeit, dass diese Zertifikate später auf den entsprechenden Endpunkten, oder in entsprechenden Zwischenknoten in andere Zertifikate umgewandelt werden können.
Ein X.509 Zertifikat kann beispielsweise in ein Kerberos Zertifikat umgewandelt werden und umgekehrt. Dies alles kann ohne größeren Aufwand geschehen. Zudem bietet das X.509 Zertifikat noch die Möglichkeit, dass dynamisch zwischen zwei Endknoten neue Zertifikate erstellt und unterzeichnet werden, wobei neue Beziehungen zwischen Nodes erstellt werden können, ohne die ganze Architektur, welche die Nodes umgibt, mit einzubeziehen.
Neben dem reinen Zertifikat für die Authentifizierung muss natürlich auch die Sicherheit der im Netzwerk übertragenen Pakete gewährleistet sein. Zu diesem Zweck nutzt das Globus Toolkit TLS (Transport Layer Security) als Standard, um einen verschlüsselten Nachrichtenaustausch zu gewährleisten. [GSI]
Neben den beiden oben gennanten Verfahren bietet das Globus Toolkit aber auch einen Proxy an. Dieser Proxy erleichtert zum einen die Bildung von VOs, zum anderen macht er aber auch ein "Single Sign On" möglich. Das heißt, wenn sich ein Benutzer einmal im Grid authentifiziert hat, macht der Proxy es möglich, dass seine Authentifizierung an andere Nodes und Resourcen weitergeleitet wird, und sich der Client so entsprechend authentifizieren kann, ohne sich noch einmal komplett neu am Grid anzumelden. Hierzu werden am Proxy entsprechende Proxy Zertifikate erstellt und benutzt. Damit ist es möglich, dynamisch VOs zu bilden, und es erleichtert das Nutzen von mehreren Resourcen enorm. [FK04]

4 Beispielanwendungen

Eine bekannte Beispielanwendung für das Grid Computing ist das WLCG (Worldwide Large Hadron Collider Computing Grid). Dieses Grid wurde am CERN (Europäische Organisation für Kernforschung) entwickelt und hat sich seitdem auf Standorte auf der ganzen Welt ausgebreitet. Durchschnittlich fallen bei den Experimenten im CERN jährlich etwa 15 PByte an Daten an, eine Datenmenge, die für einen normalen Menschen kaum vorstellbar ist. Diese Datenmengen können dabei kaum, oder genauer gesagt, gar nicht von Mitarbeitern und Forschern in der Nähe des CERNs verarbeitet werden. Deshalb wurden Forscher auf der ganzen Welt aufgerufen, durch das WLCG auf die Daten des CERNs zuzugreifen, und bei den entsprechenden Auswertungen zu helfen. Jede andere Architektur hätte bei den enormen Datenmengen, die es in solch einem Netzwerk zu übertragen gilt, Probleme bekommen, aber die Gridarchitektur eignet sich dafür hervorragend. Dabei werden die Daten auf vielen Rechnern bzw. Rechenzentren (Nodes) auf der ganzen Welt verteilt und gespeichert. Wenn dann die entsprechenden Auswertungen der Daten erfolgen sollen, wird einfach die am nächsten zum Client liegende Node genutzt. Zusammenfassend kann man sagen, dass die Kernaufgabe dieses Grids in einem Workload Management System liegt.
Das WLCG ist hierarchisch aufgebaut. Es gibt mehrere Ebenen (Tiers), in denen jeweils die Nodes untereinander vernetzt sind.

- Tier-0: Es besteht nur aus dem Rechenzentrum vom CERN.
- Tier-1: Dazu zählen 11 Rechenzentren, diese sind verteilt auf die folgenden Länder: Kanada, Frankreich, Deutschland, Italien, Niederlande, Spanien, Taipeh, Nord-Europa, England, sowie zwei in den USA. Diese Rechenzentren können die Roh-Daten verarbeiten und analysieren.
- Tier-2: Dazu zählen 160 Rechenzentren, die auf der ganzen Welt verteilt sind. Diese können etwa die Hälfte der Daten verarbeiten, die das LHC jährlich produziert.

In den einzelnen Segmenten dieses Grids wird unterschiedliche Middleware eingesetzt. Dennoch ist es aufgrund einer einheitlichen Spezifikation der OGSA möglich, alle einzelnen teilnehmenden Grids zu einem großen

Grid, dem WLCG zusammenzufassen. Europa und Asien benutzen gLite Middleware, welche vom EGEE (Enabling Grids for E-sciencE) entwickelt wurde. Nord Europa benutzt ARC (Advanced Resource Connector), und in den USA wird hauptsächlich ein Virtual Data Toolkit von der OSG (Open Science Grid) benutzt. Dabei nahmen sich fast alle eingesetzten Middleware-Lösungen ein Vorbild am Globus Toolkit, und teilweise sind die Kernkomponenten auch vom Globus Toolkit komplett übernommen. Der Dateitransfer zwischen einzelnen Nodes wird von einem sogennanten Grid File Transfer Service übernommen, dieser ist vom EGEE Projekt entwickelt worden. Mit dessen Hilfe ist es möglich, sowohl sicher als auch effizient Dateien zwischen den Nodes zu übertragen. [WLCG]

5 Zusammenfassung

Grid Computing ist eine moderne Rechner Architektur, die es erlaubt, die Internetarchitektur zu erweitern und dadurch bestehende Kapazitäten in ihrem vollen Ausmaß zu nutzen. Dabei steht vor allem die Kommunikation und Zusammenarbeit von örtlich verteilten Clients im Vordergrund. Grid Computing ist noch eine sehr neue Architektur, die sich in der heutigen IT-Landschaft erst beweisen und erproben muss. Trotzdem werden Grid Computing Systeme erfolgreich und produktiv eingesetzt, wie das Cern Beispiel eindrucksvoll beweist. In Zunkunft wird das Grid System sich weiter entwickeln und es wird in immer mehr Projekten eingesetzt. So dass irgendwann vielleicht auch der Durchschnittsbürger am Heimcomputer von diesem System profitieren kann, indem er rechen- oder zeitintensive Arbeiten einfach in eine Grid auslagert.
So könnte dieses System die Bewegungsfreiheit von Usern, ob die eines Laboranten oder die eines Heimnutzers, revolutionieren.

Literatur

[BA06] T. Barth, A. Schüll. *Grid Computing: Konzepte - Technologien - Anwendungen* Vieweg Verlag, Wiesbaden, 2006.

[FK04] I. Foster, C. Kesselmann. *The Grid: Blueprint for a new Computing Infrastrukture*. Morgan Kaufmann Publishers, San Francisco, 2004.

[GSI] Grid Security Infrastructure in C (GSI C). About the Globus Toolkit. `http://www.globus.org/toolkit/docs/5.0/5.0.0/security/gsic/`. Stand: 10.06.2011.

[GT] Globus Toolkit. About the Globus Toolkit. `http://www.globus.org/toolkit/about.html`. Stand: 10.06.2011.

[Wik01] Wikipedia(EN). Globus Toolkit. `http://en.wikipedia.org/wiki/Globus_Toolkit`. Stand: 10.06.2011.

[WLCG] WLCG. Further Information. `http://lcg.web.cern.ch/LCG/public/outreach.htm`. Stand: 10.06.2011.