# Technische Universität Ilmenau

**Fakultät für Informatik und Automatisierung**

**Institut für Praktische Informatik und Medieninformatik**

**Fachgebiet Telematik**

**Prof. Dr. Dietrich Reschke**


**Hauptseminar Informatik**

**im SS 2002**

# Grid Systeme

**Betreuer: Dipl. Inf. Thorsten Strufe**

**vorgelegt von:**


**Carsten Ginter**

**Jakob-Weil-Straße 13**

**99096 Erfurt**

**03677/463654**

**Carsten.Ginter@wi.stud.tu-ilmenau.de**

**Matrikel-Nr.: 27129**

# Inhaltsverzeichnis

# Glossar

| | |
|---|---|
| **API** | Application Programming Interface |
| **DRM** | Distributed Resource Management |
| **DUROC** | Dynamically-Updated Request Online Coallocator |
| **GSS** | Generic Security Service |
| **GRAM** | Globus Resource Allocation Manager |
| **GIIS** | Grid Index Information Service |
| **GRIP** | GRid Information Protocol |
| **GPT** | Grid Packaging Technology |
| **GRRP** | GRid Registration Protocol |
| **GRIS** | Grid Resource Information Service |
| **GSI** | Grid Security Infrastructure |
| **IDL** | Interface Definition Language |
| **IETF** | Internet Engineering Task Force |
| **LOID** | Legion object identifier |
| **LDAP** | Lightweight Directory Access Protocol |
| **MPL** | Mentat Programming Language |
| **MDS** | Monitoring and Discovery Service |
| **NCSA** | National Center for Supercomputing Applications |
| **RU** | Remote Unix |
| **RSL** | Resource Specification Language |
| **SSL** | Secure Socket Layer |
| **TLS** | Transport Layer Security |

# 1 Einleitung

## 1.1 Anforderungen an Grids

Seit einigen Jahren wird der Begriff „Grid“ häufig diskutiert. Foster und Kesselman definieren das Grid folgendermaßen: „A computational grid is hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities.“
Damit man später die Eignung verschiedener Systeme besser beurteilen kann, sollen die Anforderungen an dieser Stelle etwas näher erläutert werden. Unter dem Begriff „dependable“ versteht man, das Ressourcen nach bestimmten Vereinbarungen vorhersehbar und zuverlässig in einem bestimmten Umfang zur Verfügung stehen. Hinter dem Wort „consistency“ versteckt sich die Forderung nach Standarddiensten und standardisierten Schnittstellen. Ohne diese Standards wäre ein Grid nicht sinnvoll nutzbar. „Pervasive access“ bedeutet, das innerhalb einer bestimmten Umgebung von verschiedenen Systemen kontrollierter Zugriff auf das Grid möglich sein soll. Die letzte Anforderung, „inexpensive access“ braucht wohl nicht näher erläutert werden. Es muss natürlich für alle Nutzer des Grids ökonomisch sinnvoll sein sich an dem Grid zu beteiligen. Ansonsten wird das Grid nicht lange bestehen.[1]
Weiterhin soll ein Grid auch organisationsübergreifend in so genannten dynamischen virtuellen Organisationen arbeiten.[2] Organisationsübergreifend bedeutet, das sich mehrere Unternehmen an dem Grid beteiligen. Das bringt einige Probleme mit sich, die von der Grid Infrastruktur bewältigt werden müssen. Jedes der beteiligten Unternehmen hat eine eigene Nutzer- und Systemverwaltung. Diese kann durch das Grid nicht beeinflusst werden. Auch die Sicherheitskonzepte der einzelnen Beteiligten dürfen nicht verletzt werden. Hinzu kommt noch, das in den Unternehmen unterschiedliche Hard- und Softwaresysteme zum Einsatz kommen. Insgesamt stellt ein solches Grid also ein sehr heterogenes Gebilde dar. Um diese Anforderungen zu erfüllen ist eine umfangreiche Softwareinfrastruktur nötig.

[1] Vgl. [Foster und Kesselman 1999] S. 18
[2] Vgl. [Foster u. a. 2001] S. 2

## 1.2 Zielstellung

Diese Arbeit beschäftigt sich mit dieser Softwareinfrastruktur. Es sollen einige Softwaresysteme für die Realisierung von Grids vorgestellt werden. Aufgrund der großen Anzahl von Ansätzen für solche Softwareinfrastrukturen und dem beschränkten Umfang dieser Arbeit muss zunächst eine Auswahl vorgenommen werden. Die Auswahl beschränkt sich auf die bekanntesten, in der Literatur am häufigsten erwähnten Systeme. Dazu gehören Condor, die Sun Grid Engine, Legion und das Globus Toolkit. Da die Systeme teilweise recht unterschiedliche Ansätze aufweisen, sollen diese zunächst möglichst anschaulich vorgestellt werden. Anschließend soll anhand der oben dargestellten Anforderungen aufgezeigt werden, in welchem Umfang die Systeme zur Realisierung von Grids geeignet sind.

# 2 Condor

Das Condor Projekt begann 1988 an der Universität von Wisconsin-Madison. Es baut auf den Ergebnissen des Remote Unix (RU)-Projektes auf. Das Projekt arbeitet auf dem Gebiet des Distributed Resource Management (DRM).[3]
Condor selbst ist ein „workload management system“ für rechenintensive Aufgaben. Wie viele andere Stapelverarbeitungssystem besitzt auch Condor einen Warteschlangenmechanismus, eine Abarbeitungsstrategie, ein Prioritätsschema, eine Ressourcenüberwachung und ein Ressourcenmanagement. Nachdem ein Nutzer seinen Auftrag an Condor übermittelt hat, wird der Auftrag in eine Warteschlange gestellt. Condor entscheidet dann, wann und wo der Auftrag ausgeführt wird. Die Ausführung wird von Condor überwacht und das Ende des Auftrages dem Nutzer mitgeteilt.
Condor kann benutzt werden um einen Cluster von Rechnern zu verwalten. Condor kann die Verfügbarkeit einzelner Rechner erkennen. Sollte ein Rechner, auf dem gerade ein Auftrag bearbeitet wird, nicht länger verfügbar sein, so kann der Auftrag auf einen anderen Rechner migrieren. Der Zustand eines Auftrages wird deshalb in regelmäßigen Abständen in so genannten „checkpoints“ gespeichert. Ein Auftrag kann dann an dieser Stelle fortgesetzt werden.
Condor benötigt kein „shared file system“. Daten können im Auftrag des Nutzers transferiert werden. Es können aber auch alle I/O-Aktivitäten der Anwendung transparent auf den Rechner umgeleitet werden von dem der Auftrag übermittelt wurde.
Die Zuordnung von Aufträgen zu einzelnen Rechnern erfolgt durch den so genannten „ClassAd“-Mechanismus. Dabei werden die Anforderungen und Präferenzen eines Auftrages mit den Eigenschaften und Präferenzen des Rechners verglichen. Passen die Eigenschaften von Auftrag und Rechner zusammen, dann kann der Auftrag ausgeführt werden.
Auf diese Weise kann die gesamte Rechenleistung einer Organisation zu einer einzigen Ressource zusammengefasst werden.[4]
Darin liegt jedoch auch der entscheidende Nachteil von Condor im Hinblick auf Grids. Es ist ohne Hilfsmittel nicht organisationsübergreifend einsetzbar und kann nicht mit anderen Ressourcen, die nicht mit Condor ausgestattet sind, zusammenar-

[3] Vgl. [Condor Project 2002a]
[4] Vgl. [Condor Project 2002c,b]

beiten. Mit Hilfe von Condor's „flocking"-Mechanismus können verschiedene Condor-Cluster verbunden werden. Über Condor-G kann Condor in Globus-Grids integriert werden.[5]

# 3 Sun Grid Engine

Die Sun Grid Engine ist in zwei Versionen verfügbar. Die Standard Version ist kostenlos downloadbar. Auch der Sourcecode ist zu dieser Version verfügbar. Die Enterprise Version vertreibt Sun Microsystems kostenpflichtig.
Die Sun Grid Engine ist ein Teil einer Gesamtarchitektur, die Sun Microsystems für den Aufbau von Grids vorsieht. Obwohl die Grid Engine, zumindest in der Standard Version, auch für das Betriebssystem Linux verfügbar ist, sieht Sun deren Einsatz hauptsächlich in einer Umgebung aus Hard- und Software von Sun Microsystems vor. Die Palette reicht dabei von Servern und Speichersystemen über Betriebssystem, Ressource Management, Runtime Bibliotheken, Entwicklungstools bis hin zu System Management Software.[6]
Die Grid Engine ist in dieser Architektur für das Distributed Resource Management (DRM) verantwortlich. Sie dient der optimalen Ausnutzung der Software- und Hardwareressourcen. Ähnlich wie Condor besitzt auch die Grid Engine alle Funktionen eines klassischen Stapelverarbeitungssystemes. Als zusätzliche Funktion ist die so genannte „batch-aware shell" zu erwähnen. Sie ermöglicht das ausführen von interaktiven Anwendungen auf der Sun Grid Engine Software.[7]
Die Einsatzmöglichkeiten der Grid Engine sind allerdings beschränkt. Das liegt zum einen an der starken Ausrichtung an Hard- und Softwareumgebungen von Sun Microsystems und zum anderen an der Art der möglichen Grids. Sun unterteilt dabei drei Arten bzw. Ebenen. Die kleinste Ebene stellt das Cluster Grid dar. Dabei werden Workstations und Server zu einem Grid zusammengefasst, die typischerweise unter Kontrolle eines Projektes oder einer Abteilung eines Unternehmens stehen. Für diesen Einsatzzweck sieht Sun die Standard Version der Sun Grid Engine vor. Die zweite Ebene ist das Campus bzw. Enterprise Grid. Hier werden mehrere Cluster Grids zusammengefasst. Es beteiligen sich demnach mehrere Abteilungen eines

[5] Vgl. [Condor Project 2002c]
[6] Vgl. [SUN Microsystems, Inc. 2002a] S. 19-20
[7] Vgl. [SUN Microsystems, Inc. 2002a] S. 21-22

Unternehmens an dem Grid. Hierfür ist die Enterprise Edition der Sun Grid Engine gedacht. Die dritte Ebene nennt man Global Grid. Hierbei arbeiten Campus Grids verschiedener Unternehmen zusammen. Solche Grids können mit der Grid Engine allein nicht ermöglicht werden. Es wird zusätzliche Software wie das Globus Toolkit nötig.[8]

# 4 Legion

Legion wurde seit 1993 an der Universität von Virginia entwickelt. Die Entwickler bezeichnen es als objektorientiertes Metasystem. Es ist entworfen um Millionen von Hosts und Billionen von Objekten über Hochgeschwindigkeitsnetze zu einem großen System zusammenzufassen. Dem Nutzer gegenüber erscheint das System als ein einzelner Computer.
Legion bildet eine Zwischenschicht zwischen Betriebssystem und Anwendungen. Jede Ressource wird in Legion als Objekt repräsentiert. Verwaltet werden die Objekte von den „Legion object management services“. Zusätzlich stehen weitere Kerndienste wie das „Legion file system“, die „Context-space directory services“ und „Resource management services“ zur Verfügung. Damit Anwendungen auf die Dienste von Legion zugreifen können, werden von Legion Bibliotheken bereitgestellt. Die Anwendungen müssen diese Bibliotheken verwenden um mit Legion zusammenzuarbeiten. Das gesamte System ist erweiterbar. Sowohl die Kerndienste als auch die Bibliotheken können erweitert oder ausgetauscht werden.[9]
Die Kommunikation zwischen den Objekten erfolgt über Methodenaufrufe. Das Erstellen von Objekten ist mit verschiedenen Programmiersprachen wie C, Fortran, C++, Java und Mentat Programming Language (MPL) möglich. Die Schnittstellen der Objekte werden durch eine Interface Definition Language (IDL) beschreiben. Derzeit werden durch Legion zwei IDLs unterstützt, die Corba IDL und MPL. Ansprechbar sind die einzelnen Objekte über so genannte „context names“ die vom Ersteller des Objektes vergeben werden. Durch den schon erwähnten Directory Service werden diese Namen auf eindeutige systemnahe Identifikatoren, die so genannten LOID (Legion object identifier), abgebildet.[10]

[8] Vgl. SUN Microsystems, Inc. [2002a] S. 5 und SUN Microsystems, Inc. [2002b] S. 24
[9] Vgl. [Legion Research Group b]
[10] Vgl. [Grimshaw u. a. 1998]

Durch seine Sicherheitsarchitektur soll der Einsatz von Legion auch über administrative Grenzen hinweg nutzbar sein. Das heißt, es sind auch „Global Grids“ möglich. Durch das verwendete RSAREF 2.0 ist sowohl die Verschlüsselung als auch das Signieren der Kommunikation möglich. Legion läuft mit den Rechten des Users der es verwendet. Es kann also nur die Ressourcen verwenden die dem Nutzer zur Verfügung stehen. Der Nutzer kann also keine Sicherheitsrichtlinien verletzen. Wer auf die Ressourcen, die der Nutzer zur Verfügung stellt, zugreifen darf kann der Nutzer bestimmen.[11]
Der Nutzen durch den Produktiven Einsatz ist wohl eher gering. Das ganze Projekt hat eher wissenschaftlichen Charakter. Die Entwickler sehen Legion eher als eine Sammlung von Spezifikationen und Schnittstellen und weniger als eine Implementation.[12] Sie schließen daher auch die Fertigstellung eines Produktes aus. Dies soll anderen überlassen werden.[13] Ein Beispiel für die kommerzieller Vermarktung von Legion findet sich bei der Firma AVAKI.[14]

# 5 Globus Toolkit

## 5.1 Inhalt des Globus Projektes

Das Globus Projekt ist ein Forschungsprojekt. Es beschäftigt sich mit den Problemen bei der Erstellung von Gridinfrastrukturen und mit den Problemen bei der Entwicklung von Anwendungen, welche die Dienste des Grids verwenden.[15] Die Initiatoren des Projektes sind die „Mathematics and Computer Science Division“ des Argonne National Laboratory, das „Information Sciences Institue“ der „University of Southern California“ und das „Distributed Systems Laboratory“ der Universität von Chicago.[16]
Die Arbeit des Projektes hat eine Sammlung von Softwaretools hervorgebracht, die das Erstellen von Grids und gridbasierten Anwendungen erleichtert. Diese Tools werden unter dem Namen „Globus Toolkit“ zusammengefasst. Das Globus Toolkit ist momentan in der Version 2.0 verfügbar. Das Globus Toolkit besitzt eine offene

[11] Vgl. [Legion Research Group d]
[12] Vgl. [Legion Research Group a]
[13] Vgl. [Legion Research Group c]
[14] siehe http://www.avaki.com/
[15] Vgl. [Globus Project 2001a]
[16] Vgl. [Globus Project 2002a]

Architektur und ist als Sourcecode verfügbar. An der Entwicklung waren nicht nur die bereits erwähnten Organisationen beteiligt, sondern auch viele andere.[17]
Das Globus Toolkit bildet eine Middleware zwischen den Anwendungen und den spezifischen Ressourcen eines Rechners. Dazu stellt es den Anwendungen bestimmte Dienste und Schnittstellen bereit. Für alle Dienste stellt das Globus Toolkit ein entsprechendes Application Programming Interface (API) zur Verfügung. Darüber können Anwendungen die Dienste nutzen. Für die direkte Nutzung durch einen Nutzer stehen für viele Dienste Kommandozeilen-Tools zur Verfügung. Die Kommunikation zwischen den Diensten erfolgt durch bestimmte Protokolle. Das Globus Toolkit regelt auch die Verbindung zu den jeweiligen lokalen Betriebssystemen der beteiligten Rechner. Dabei kann das Globus Toolkit durchaus mit verschiedenen Architekturen umgehen.
Die Dienste des Toolkits lassen sich in drei Kategorien einteilen: „Resource Management“, „Information Services“ und „Data Management“. Alle drei Komponenten benutzen eine einheitliche Grid Security Infrastructure (GSI). Für die Auslieferung des Globus Toolkits wird die Grid Packaging Technology (GPT) verwendet.

## 5.2 Grid Security Infrastructure

Ziel der Grid Security Infrastructure (GSI) ist es, sichere Kommunikation zwischen den einzelnen Elementen des Grids zu ermöglichen. Da die Kommunikation dabei auch organisationsübergreifend erfolgen soll, ist kein zentral verwaltetes Sicherheitssystem möglich. Um die Verwendung des Grids für die Nutzer möglichst einfach zu gestalten, wird ein „single sign-on“ Mechanismus vorgesehen.[18]
Die GSI bietet Dienste für sichere Authentifizierung und Kommunikation über öffentliche Netzwerke. Diese Dienste können von Anwendungen und anderen Diensten des Globus Toolkits verwendet werden. Im folgenden werden die Möglichkeiten der GSI näher beschrieben.
Die GSI basiert auf einer Public Key Infrastruktur. Als Schlüssel werden dabei X.509 Zertifikate verwendet. Diese bestehen aus einem öffentlichen und aus einem privaten Schlüssel. Dabei ist der private Schlüssel durch ein Passwort gesichert. Zur Verschlüsselung von Datenübertragungen wird Transport Layer Security (TLS) bzw.

[17] Vgl. [Globus Project 2002g]
[18] Vgl. [Globus Project 2002i]

Secure Socket Layer (SSL) verwendet. Diese Standards wurden im Globus Toolkit erweitert um den bereits erwähnten „single sign-on“ Mechanismus zu realisieren.
Die Implementation der GSI baut auf die Generic Security Service (GSS)-API[19], eine Standard-API für Sicherheitssysteme die von der Internet Engineering Task Force (IETF) gefördert wird, auf. Diese API stellt einheitliche Schnittstellen zur Verfügung, die unabhängig von den darunter liegenden Mechanismen und Programmierumgebungen sind. Dadurch ist die GSI-Implementation unabhängig von den jeweiligen Sicherheitsmechanismen der verschiedenen Grid-Ressourcen. Diese können unterschiedlich sein und sich ändern, ohne das Änderungen an der GSI oder den Anwendungen nötig sind.[20]

## 5.3 Resource Management

Für das Ressourcenmanagement sieht das Globus Toolkit verschiedene Komponenten vor. Dazu zählen die Resource Specification Language (RSL), der Globus Resource Allocation Manager (GRAM) und der Coallocator.[21]
Die RSL ist eine Sprache für die Beschreibung von Ressourcen. Sie wird von verschiedenen Komponenten der „Globus Resource Management“ Architektur verwendet um deren Funktion in Zusammenarbeit mit anderen Komponenten zu erfüllen.[22]
Der GRAM ist die unterste Ebene der „Globus Resource Management“ Architektur. Er ermöglicht das Ausführen, das Überwachen und das Beenden eines Jobs auf einer bestimmten Ressource. Dazu nimmt er Anfragen in Form der RSL entgegen und parst sie. Anschließend reserviert er die nötigen Ressourcen und startet den Job. Während der Abarbeitung informiert er den Client über den Status des Auftrages und aktualisiert den MDS.
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.[23]

[19] Vgl. [Linn 1997, 2000]
[20] Vgl. [Foster u. a. 1998]
[21] Vgl. [Globus Project 2002b]
[22] Vgl. [Globus Project 2000]
[23] Vgl. [Globus Project 2002h,f]

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 Beispielimplementation liefert das Globus Toolkit den Coallocator „Dynamically-Updated Request Online Coallocator (DUROC)“ mit.[24]

## 5.4 Information Services

Die Informationsdienste des Globus Toolkits werden unter dem Namen Monitoring and Discovery Service (MDS) zusammengefasst. Der MDS besteht aus zwei Arten von Diensten, dem Grid Resource Information Service (GRIS) und dem Grid Index Information Service (GIIS). Deren spezifische Eigenschaften und Aufgaben werden später beschrieben. Zunächst sollen die Gemeinsamkeiten betrachtet werden.
Der MDS dient der Speicherung von Daten über die Art und den Zustand von Ressourcen in so genannten Verzeichnissen. Dabei handelt es sich um statisch und dynamische Informationen. Zur Abfrage und Speicherung von Informationen wird das Lightweight Directory Access Protocol (LDAP) verwendet. LDAP ist ein Standard der IETF.[25] Die Daten werden in einem oder mehreren LDAP-Servern gespeichert. Eine Vielfalt von LDAP-Server-Implementationen, sowohl kommerzielle als auch Open-Source, ist verfügbar und kann für den MDS genutzt werden. Im Globus Toolkit wird die frei verfügbare OpenLDAP-Implementation mitgeliefert. Die Struktur der Daten in einem solchen Verzeichnis wird durch Schemas beschrieben. Das Globus Toolkit enthält ein solches Schema zur Beschreibung von Grid-Ressourcen. Dieses Schema kann natürlich erweitert werden und an die jeweiligen Bedürfnisse angepasst werden.[26] Für den Datenaustausch mit den GRIS und GIIS sind zwei Protokolle vorgesehen. Das GRid Information Protocol (GRIP) dient dem Abfragen der Daten von GRIS und GIIS. Das GRid Registration Protocol (GRRP) dagegen ist dafür gedacht, um Informationen in die Verzeichnisdienste zu „pushen“.[27]

### 5.4.1 Grid Resource Information Service

Der GRIS läuft auf jeder Ressource. Das heißt, auf jedem Rechner der an das Grid angebunden ist, läuft ein LDAP-Server mit dieser Funktionalität. Ist die Adresse

[24] Vgl. [Globus Project]
[25] Vgl. [Yeong u. a. 1995]
[26] Vgl. [Globus Project 2001b]
[27] Vgl. [Czajkowski u. a. 2001]

eines Rechners bekannt, kann dieser Dienst über den Port 2135 erreicht werden.
In diesen Verzeichnissen sind Informationen über die Ressourcen des jeweiligen Rechners enthalten. Es können Informationen über Konfiguration, Fähigkeiten und Status der Ressource gezielt abgefragt werden. Es ist jedoch keine Suche vorgesehen.[28]

### 5.4.2 Grid Index Information Service

Der GIIS aggregiert die Informationen mehrerer GRIS. Dieser Dienst enthält also Informationen über mehrere Ressorcen in einem bestimmten Bereich des Grids. Dieser Bereich kann das gesamte Grid sein. Es ist aber auch möglich eine Hierarchie von GIIS aufzubauen. Dabei gibt es mehrere GIIS die für bestimmte Teilbereiche des Grids zuständig sind und einen oder mehrere GIIS die wiederum die Informationen der untergeordneten GIIS aggregieren. Ein GIIS erlaubt durch seine zusammen getragenen Informationen die Suche nach bestimmten Ressourcen.[29]

## 5.5 Data Management

Die Komponenten des Data Management sind speziell für so genannte Data Grids entwickelt. Sie sind aber natürlich auch für andere Arten von Grids sinnvoll. Deshalb sind sie auch im Globus Toolkit enthalten.
Als Datentransferprotokoll kommt GridFTP zum Einsatz. Außerdem gibt es noch zwei Komponenten namens „Globus Replica Management“ und „Globus Replica Catalog“. [30]
GridFTP ist ein erweitertes FTP-Protokoll. Der wesentlichste Unterschied besteht in der Integration der GSI. Weiterhin sind bei GridFTP mehrere Datenkanäle möglich. Das ermöglicht parallele Datentransfers. Außerdem besteht die Möglichkeit Dateien nur partiell zu transferieren. Das ist besonders nützlich bei besonders großen Dateien, die nur zu bestimmten Teilen benötigt werden. Weiterhin ist durch Server-zu-Server-Transfers die Möglichkeit zur Datenreplikation gegeben.[31]
Besonders bei wissenschaftlichen Anwendungen wird häufig mit großen Datenmengen gearbeitet. Dabei sind meistens auch mehrere Organisationen beteiligt. Da nicht jede Organisation die Möglichkeiten bzw. die Notwendigkeit hat alle Daten selbst

[28] Vgl. [Globus Project 2002k]
[29] Vgl. [Globus Project 2002k]
[30] Vgl. [Globus Project 2002c]
[31] Vgl. [Globus Project 2002j]

zu speichern, ist ein Mechanismus nötig der es ermöglicht herauszufinden wo welche Daten zu einem bestimmten Zeitpunkt lagern. Außerdem sollte es über diesen Dienst auch möglich sein festzulegen, welche Daten wo bereitgestellt werden sollen. Diese Aufgaben erfüllt das „Globus Replica Management“. Dazu nutzt es den Globus Replica Catalog und GridFTP.[32]
Der „Globus Replica Catalog“ unterstützt das „Globus Replica Management“. In diesem Verzeichnis werden Verknüpfungen zwischen logischen Namen und einer oder mehrerer Kopien physikalisch gespeicherter Dateien abgelegt. Realisiert ist dieses Verzeichnis im Globus Toolkit durch einen LDAP-Server. Die API des „Globus Replica Catalog“ ist jedoch so konzipiert, das das Speicherverfahren(hier der LDAP-Server) austauschbar ist. So könnten die Daten zukünftig beispielsweise in einer Datenbank gespeichert werden.[33]

## 5.6 Packaging Technology

Seit der Version 2.0 des Globus Toolkits wird eine neue Packaging Technologie verwendet. Diese basiert auf der Grid Packaging Technology (GPT) des National Center for Supercomputing Applications (NCSA).[34] Das Paketformat ist XML-basiert. Es ist möglich sowohl Binärpakete als auch Sourcecodepakete zu erstellen. Die Sourcecodepakete können so gestaltet werden, das daraus automatisch die Anwendungen erstellt werden können. Weiterhin können Abhängigkeiten zwischen Paketen festgelegt werden. Es ist zusätzlich auch ein Mechanismus vorgesehen, welcher Sourcecodes vor dem kompilieren auch noch „patchen“ kann.[35]

[32]Vgl. [Globus Project 2002e]
[33]Vgl. [Globus Project 2002d]
[34]siehe http://www.ncsa.uiuc.edu/Divisions/ACES/GPT/
[35]Vgl. [Blau und Bletzinger 2002]

# 6 Fazit

Die in der Arbeit gezeigten Ansätze für Gridinfrastrukturen sind nur einige mögliche Ansätze. Mit entsprechendem Zeitaufwand lassen sich noch viele weiter Projekte finden die sich mit diesem Thema befassen. Die vorgestellten Ansätze spiegeln jedoch ganz gut den Stand der derzeitigen Entwicklung wieder. Die meisten Projekte sind über einen Ansatz noch nicht hinausgekommen. Häufig entwickeln die Projekte auch Ansätze für sehr spezifische Einsatzzwecke. Das liefert zwar optimale Ergebnisse für einen bestimmten Zweck, lässt sich aber häufig nicht wiederverwenden.
Die untersuchten Projekte haben die Anforderungen aus der Definition von Foster und Kesselman weitestgehend erfüllt. Sehr unterschiedlich fällt jedoch die Eignung für den Einsatz in so genannten virtuellen Organisationen aus. Legion und das Globus Toolkit sind hier Vertreter für die wenigen Projekte, die einen solchen Einsatz ermöglichen.
Das Globus Projekt bietet einen sehr allgemeinen Ansatz. Dort wurden die allgemeinen Anforderungen von Grids analysiert und in Software umgesetzt. Die daraus hervorgegangene Software ist auf sehr vielen Soft- und Hardwareplattformen einsetzbar. Zudem ist des Globus Toolkit flexibel erweiterbar. Das hat wohl auch zu seiner großen Verbreitung geführt. Praktisch alle Versuche größere organisationsübergreifende Grids zu verwirklichen greifen in mehr oder weniger großem Umfang auf das Globus Toolkit zurück. Beispiele hierfür sind das NASA Information Power Grid (IPG)[36] und das DataGrid Projekt[37]. Allgemein kann man sagen, das sich das Globus Toolkit zu einem Quasi-Standard entwickelt hat.
Momentan gibt es noch recht wenige Versuche große Grids für den produktiven Einsatz zu errichten. Da solche Projekte sowohl durch private Unternehmen(z.B. IBM) als auch durch Einrichtungen der öffentlichen Hand gefördet werden(siehe European DataGrid), ist zu erwarten das sich die Anzahl solcher Projekte in den nächsten Jahren erhöhen wird. Besonders wissenschaftliche Institutionen könnten dadurch profitieren. Bis die Technologie der Grids ausgereift ist und Grids zum Alltag gehören, werden wohl aber noch einige Jahre vergehen.

[36] siehe http://www.nas.nasa.gov/About/IPG/ipg.html
[37] siehe http://www.eu-datagrid.org/

# Literatur

[Blau und Bletzinger 2002] Blau, Eric ; Bletzinger, Michael: *Grid Packaging Technology - Overview*. Februar 2002. – URL `http://www.ncsa.uiuc.edu/Divisions/ACES/GPT/Overview.html`. – Zugriffsdatum: 25.07.2002

[Condor Project 2002a] Condor Project: *How did the Condor project start?* 2002. – URL `http://www.cs.wisc.edu/condor/background.html`. – Zugriffsdatum: 23.07.2002

[Condor Project 2002b] Condor Project: *Overview of the Condor High-Throughput Computing System*. 2002. – URL `http://www.cs.wisc.edu/condor/overview/`. – Zugriffsdatum: 23.07.2002

[Condor Project 2002c] Condor Project: *What is Condor?* 2002. – URL `http://www.cs.wisc.edu/condor/description.html`. – Zugriffsdatum: 23.07.2002

[Czajkowski u. a. 2001] Czajkowski, Karl ; Fitzgerald, Steven ; Foster, Ian ; Kesselman, Carl: Grid Information Services for Distributed Resource Sharing. (2001). – URL `http://www.globus.org/research/papers/MDS-HPDC.pdf`. – Zugriffsdatum: 25.07.2002

[Foster und Kesselman 1999] Foster, Ian (Hrsg.) ; Kesselman, Carl (Hrsg.): *The Grid: Blueprint for a New Computing Infrastructure*. San Francisco : Morgan Kaufmann Publishers, Inc., 1999. – ISBN 1-55860-475-8

[Foster u. a. 1998] Foster, Ian ; Kesselman, Carl ; Tsudik, Gene ; Tuecke, Steven: A Security Architecture for Computational Grids. (1998). – URL `ftp://ftp.globus.org/pub/globus/papers/security.pdf`. – Zugriffsdatum: 04.07.2002

[Foster u. a. 2001] Foster, Ian ; Kesselman, Carl ; Tuecke, Steven: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. (2001). – URL `http://www.globus.org/research/papers/anatomy.pdf`. – Zugriffsdatum: 29.06.2002

[Globus Project ] Globus Project: *The Dynamically-Updated Request Online Coallocator - DUROC v0.8*. – URL `http://www.globus.org/duroc/function_reference.html`. – Zugriffsdatum: 25.07.2002

[Globus Project 2000] Globus Project: *The Globus Resource Specification Language RSL v1.0*. Mai 2000. – URL `http://www.globus.org/gt2/admin/guide-overview.html`. – Zugriffsdatum: 25.07.2002

[Globus Project 2001a] Globus Project: Grid Research. (2001), December. – URL `http://www.globus.org/research/default.asp`. – Zugriffsdatum: 25.07.2002

[Globus Project 2001b] Globus Project: Introduction to Grid Computing and the Globus Toolkit. (2001), Oktober. – URL `http://www.globus.org/training/grids-and-globus-toolkit/IntroToGridsAndGlobusToolkit.pdf`. – Zugriffsdatum: 05.07.2002

[Globus Project 2002a] Globus Project: Collabotators. (2002), Juni. – URL `http://www.globus.org/about/collaborators.html`. – Zugriffsdatum: 25.07.2002

[Globus Project 2002b] Globus Project: *Globus 2.0 Overview*. April 2002. – URL `http://www.globus.org/gt2/admin/guide-overview.html`. – Zugriffsdatum: 25.07.2002

[Globus Project 2002c] Globus Project: *The Globus Data Grid Effort: Software*. Januar 2002. – URL `http://www.globus.org/datagrid/software.html`. – Zugriffsdatum: 25.07.2002

[Globus Project 2002d] Globus Project: *The Globus Replica Catalog*. Januar 2002. – URL `http://www.globus.org/datagrid/replica-catalog.html`. – Zugriffsdatum: 25.07.2002

[Globus Project 2002e] Globus Project: *Globus Replica Management*. April 2002. – URL `http://www.globus.org/datagrid/replica-management.html`. – Zugriffsdatum: 25.07.2002

[Globus Project 2002f] Globus Project: *Globus Resource Allocation Manager*. Juli 2002. – URL `http://www.globus.org/gram/`. – Zugriffsdatum: 25.07.2002

[Globus Project 2002g] Globus Project: The Globus Toolkit. (2002), Juli. – URL `http://www.globus.org/toolkit/`. – Zugriffsdatum: 25.07.2002

[Globus Project 2002h] Globus Project: *GRAM Overview*. Mai 2002. – URL `http://www-fp.globus.org/gram/overview.html`. – Zugriffsdatum: 25.07.2002

[Globus Project 2002i] Globus Project: *Grid Security Infrastructure*. März 2002. – URL `http://www.globus.org/security`. – Zugriffsdatum: 04.07.2002

[Globus Project 2002j] Globus Project: *The GridFTP Protocol and Software*. Januar 2002. – URL `http://www.globus.org/datagrid/gridftp.html`. – Zugriffsdatum: 25.07.2002

[Globus Project 2002k] Globus Project: MDS 2.1 User,s Guide. (2002), Mai. – URL `http://www.globus.org/mds/mdsusersguide.pdf`. – Zugriffsdatum: 04.07.2002

[Grimshaw u. a. 1998] Grimshaw, Andrew S. ; Lewis, Michael J. ; Ferrari, Adam J. ; Karpovich, John F.: Architechtural Support for Extensibility and Autonomy in Wide-Area Distributed Object Systems. (1998), Juni. – URL `http://legion.virginia.edu/papers/CS-98-12.pdf`. – Zugriffsdatum: 02.07.2002

[Legion Research Group a] Legion Research Group: *Legion Architecture*. – URL `http://legion.virginia.edu/overview_arch.html`. – Zugriffsdatum: 25.07.2002

[Legion Research Group b] Legion Research Group: *Legion Overview*. – URL `http://legion.virginia.edu/overview.html`. – Zugriffsdatum: 25.07.2002

[Legion Research Group c] Legion Research Group: *The Legion Project*. – URL `http://legion.virginia.edu/index.html`. – Zugriffsdatum: 25.07.2002

[Legion Research Group d] Legion Research Group: *Legion Security*. – URL `http://legion.virginia.edu/overview_security.html`. – Zugriffsdatum: 25.07.2002

[Linn 1997] Linn, J.: *RFC 2743 - Generic Security Service Application Program Interface Version 2*. Januar 1997. – URL `http://www.ietf.org/rfc/rfc2078.txt?number=2078`. – Zugriffsdatum: 04.07.2002

[Linn 2000] Linn, J.: *RFC 2743 - Generic Security Service Application Program Interface Version 2, Update 1.* Januar 2000. – URL http://www.ietf.org/rfc/rfc2743.txt?number=2743. – Zugriffsdatum: 04.07.2002

[SUN Microsystems, Inc. 2002a] SUN Microsystems, Inc.: *Sun Cluster Grid Architecture: A technical white paper describing the foundation of Sun Grid Computing.* 2002. – URL http://wwws.sun.com/software/grid/SunClusterGridArchitecture.pdf. – Zugriffsdatum: 29.06.2002

[SUN Microsystems, Inc. 2002b] SUN Microsystems, Inc.: *Sun Grid Engine, Enterprise Edition 5.3 Administration and User's Guide.* Mai 2002. – URL http://www.sun.com/products-n-solutions/hardware/docs/pdf/816-4739-10.pdf. – Zugriffsdatum: 02.07.2002

[Yeong u. a. 1995] Yeong, W. ; Howes, T. ; Kille, S.: *Lightweight Directory Access Protocol.* März 1995. – URL http://www.ietf.org/rfc/rfc1777.txt?number=1777. – Zugriffsdatum: 05.07.2002