Grids
Beschreibung des Grid-Gedankens und Abgrenzungen

im Rahmen des Seminars
Grundlagen webbasierter Informationssysteme

Marco Kiltz

Sommersemester 2004

Inhaltsverzeichnis

1. Einleitung und Motivation

Grid-Technologie ist in den letzten Jahren zunehmend populär geworden. Sie soll die Nutzung von verteilt vorliegenden Ressourcen ermöglichen. Der Grid-Gedanke zielt darauf „Rechnerleistung aus der Steckdose“ zu erhalten, ähnlich der Versorgung mit Strom oder Wasser [ReSc04] in Analogie zum englischen Begriff *Power-* beziehungsweise *Utility-Grid*. Computing Grids zielen auf die Verwendung ungenutzter Ressourcen wie Prozessor, Rechner und Hauptspeicher. Bei Data Grids werden verteilte persistente Speicher genutzt, um größere Datenmengen verwalten und speichern zu können. Data Grids teilen sich die technischen Grundlagen mit den Computing Grids und werden im Rahmen dieses Seminars in einer eigenen Ausarbeitung behandelt. Die folgenden Arbeit konzentriert sich auf die Grid-Grundlagen, die Computing Grids und Data Grids gemeinsam haben.

Das Grid Computing entstand in den neunziger Jahren, um große wissenschaftliche Anwendungen mit hohem Rechenaufwand bearbeiten zu können. Dazu wurden zunächst Hochleistungsrechner verbunden, zunehmend wurde jedoch insbesondere für große Firmen das zusätzliche Verbinden von Arbeitsplatzrechnern interessant. Dies ist möglich, da Untersuchungen [Bers02] ergeben haben, dass in vielen Unternehmen Desktopcomputer nur zu 5 % ausgelastet sind, und auch die Server der Firmen selten voll genutzt werden.

Jeder Rechner, der an einem solchen Grid-System teilnimmt, kann einerseits vorhandene ungenutzte Rechenleistung anbieten, oder aber selbst Ressourcen auf entfernten Rechnern nutzen. Manche Probleme, die auf einem einzelnen Prozessor einige Tage Bearbeitungszeit benötigten [OGSI04], sind mit Grid Computing in wenigen Minuten ausgewertet, wenn das Problem einen hohen Grad an Nebenläufigkeit aufweist, problemlos in einzelne Teiljobs für die Grid-Knoten aufgeteilt werden kann und im Grid genügend Ressourcen vorhanden sind. Um Jobs und zu bearbeitende Daten auf die Grid-Knoten verteilen zu können und die schnelle Kommunikation zwischen voneinander abhängigen Teiljobs zu ermöglichen, müssen die Grid-Knoten mit ausreichender Bandbreite vernetzt werden.

Die Aufgaben des Grid beschränken sich nicht nur auf das Verwalten und Koordinieren von Prozessorzeit und Speicher, sondern auch von Software, Lizenzen und anderen Services. Teure Lizenzen können effizienter genutzt werden. Es ist ausreichend, den Auftrag an einen Grid-Knoten zu senden, der die entsprechende Lizenz besitzt. Auch teure spezielle Hardware, wie z.B. ein Hochleistungsmikroskop, kann durch das Grid von anderen Teilnehmern genutzt und so besser ausgelastet werden.

Durch schnellere Berechnungen verbesserte sich die Produktivität, und die Ressourcen des einzelnen Computers werden besser genutzt.

2. Grundlagen

Um wie zuvor beschrieben die ungenutzten verteilt vorliegenden Ressourcen effizient für Anwendungen nutzbar zu machen genügt es nicht, dem Entwickler einer Grid-Anwendung direkten Zugriff auf die Ressourcen zu geben. Der Entwickler einer Grid-Anwendung wäre so selbst für die Unterstützung der heterogenen Prozessor- und Rechnerarchitekturen, Betriebssysteme, Schnittstellen und Netzwerkprotokolle zuständig, wodurch die Entwicklung erheblich aufwendiger und fehlerträchtiger und damit teurer würde.
Es ist daher die Aufgabe der Grid-Software, diese Heterogenität zu überwinden, und dem Anwendungsentwickler eine einheitliche Schnittstelle zu den Ressourcen anzubieten. Im Idealfall erscheint das Grid aus Sicht des Entwicklers wie ein einziger sehr leistungsfähiger Rechner, so dass er sich um Details der Ressourcensuche und -zuordnung nicht zu kümmern braucht. In heute verfügbaren Grid-Systemen ist die Abstraktion jedoch noch nicht so weit fortgeschritten.
Für die Abstraktion von technischen Details der Ressourcen ist ihre Verwaltung durch das Grid erforderlich. So ist bei der Zuteilung der Jobs auf Grid-Knoten darauf zu achten, dass einzelne Rechner nicht extrem ausgelastet und andere untätig sind. Ein Grid benötigt dazu einen Scheduler[1], der aufgrund der heterogenen Ressourcen deutlich komplexer ausfällt, als innerhalb eines Ein- oder Mehrprozessorsystems.
Abbildung 1 zeigt am Beispiel einer CPU-Ressource den Ablauf bei der Nutzung des Grids. Eine Grid-Anwendung (Client) benötigt zusätzliche Ressourcen. Alle Ressourcen die in einem Grid zur Verfügung stehen sind in mehreren *Resource Registries* gespeichert, über die geeignete Ressourcen ermittelt werden (1.). Die Grid-Anwendung bittet nun um Zuweisung der Ressourcen beim zuständigen Ressourcen-Manager. Bei erfolgreicher Zuweisung erhält die Grid-Anwendung vom Ressourcen-Manager eine Referenz auf die angeforderten Ressourcen. Die Grid-Anwendung sendet daraufhin den Job und zusätzliche Eingabedaten an die Ressource und führt die Anwendung aus, während der Ressourcen-Manager die Resource Registries entsprechend der neuen Situation aktualisiert, damit die nun gestiegene Auslastung der zugeteilten Ressource für weitere Anforderungen berücksichtigt werden kann. Über die Referenz, die der Client vom Ressourcen-Manager erhalten hat, kann dieser nun die Ausführung seiner Anwendung überwachen und die Ergebnisse abrufen.

1 Unter Scheduling versteht man, die Auswahl eines Prozesses aus einer Menge ausführbarer Prozesse. Die Reihenfolge der Prozesse wird von der Scheduling-Strategie bestimmt.

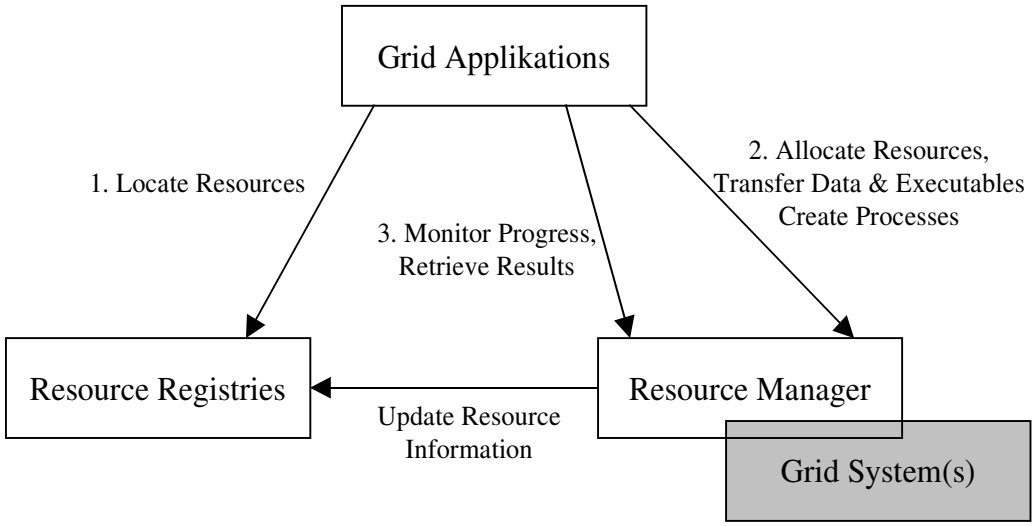

Abbildung 1: Ablauf zur Nutzung einer Ressource in einem Grid

Ein funktionierendes Grid stellt auch ein geeignetes Mittel für organisationsübergreifende Zusammenarbeit dar. Hierfür hat sich der Begriff der virtuellen Organisation (*Virtual Organisation,* VO) etabliert. Darunter ist eine Umgebung zu verstehen, bei der Personen an verschiedenen Orten in verschiedenen Recherumgebungen zusammenarbeiten und dazu Ressourcen teilen wollen. Ein Beispiel für eine VO ist eine Gruppe von Entwicklern aus verschiedenen Unternehmen, die an einem gemeinsamen Projekt arbeiten, oder ein Wetteramt, welches Informationen von ihren weit verteilten Messstationen einholt und die Ergebnisse ihrer Berechnungen an angeschlossene Unternehmen weiterleitet. Diese und viele weitere Beispiele zeigen, dass sich virtuelle Organisationen in ihrer Aufgabe, Größe, Lebensdauer und Struktur unterscheiden, jedoch kann man gemeinsame Anforderungen an die Rechnerarchitektur erkennen. Eine flexible Kontrolle der verteilten Ressourcen und geeignete Zugriffskontrollmechanismus zählen zu den gemeinsamen Anforderungen. Alle an einer solchen Kooperation teilnehmenden Organisationen steuern Mitarbeiter und Ressourcen bei und stellen sie im Grid zur Verfügung. Eine VO profitiert somit von einem Grid.

3. Abgrenzung des Grid-Begriffs

3.1 Merkmale eines Grid

Die Problematik der Einteilung was nun ein Grid ist und was nicht erinnert an die anfänglichen Diskussionen um die genaue Definition des Begriffs Internet. Wie dieser wird auch der Grid-Begriff einem ähnlichen Wandel unterliegen.

Die Definition des Grid-Begriffs soll daher auf der Grundlage der wichtigsten Merkmale erfolgen.
[Fost02] nennt drei wesentliche Kriterien, die ein Grid ausmachen:

- Verwaltung von Ressourcen ohne zentralen Kontrolle
- standardisierte Protokolle und standardisierte Schnittstellen
- Funktionen zur Zusicherung einer Dienstgüte (*Quality of Service* [2])

[BoDG01] setzt dagegen andere Schwerpunkte und nennt u.a.:

- Heterogenität: ein Grid sollte mit unterschiedlichen Hard- und Softwaresystemen lauffähig kommunizieren können
- Skalierbarkeit: ein Grid sollte aus einigen wenigen bis zu einigen Millionen Ressourcen bestehen.

Diese Liste ist nicht vollständig, umfasst jedoch alle wesentlichen Kriterien und soll daher im folgenden als Grundlage zu Bewertung von Systemen und Technologien dienen, die als Grid bezeichnet werden.

3.2 Cluster

Die Verbindung mehrerer Rechner durch ein lokales Netzwerk (LAN) und die Kopplung zu einem großen Rechner durch entsprechende Software nennt man Cluster. Die Rechner haben oft ähnliche oder identische Hardware. Sie gehören meist zu einer Organisation oder Abteilung, wodurch aufwendige Sicherheitsmaßnahmen nicht erforderlich sind.
Betrachtet man ein Cluster-System anhand obiger Kriterien, so stellt man fest, dass ein Cluster einen Knotenpunkt mit zentraler Kontrolle besitzt, und keine Heterogenität der Rechner vorliegt. Somit kann ein Cluster nicht als Grid angesehen werden.

3.3 Peer-to-Peer

Verteilte Systeme beruhen zumeist auf dem Client-Server-Prinzip, in diesem gibt es einen Server, der einen Dienst anbietet, und Clients, die den Dienst nutzen. Dem gegenüber steht des Prinzip des Peer-to-Peer [3] (P2P). Beim P2P ist die Rollenverteilung aufgehoben. Jeder Knoten kann Client und Server gleichzeitig sein. Das bekannteste Beispiel für P2P sind File-Sharing-Systeme. Ein Peer-to-Peer-System hat kein zentrales Kontrollsystem und die Peers sind oft heterogen, es fehlen jedoch Funktionen zur Zusicherung einer Dienstgüte und es existieren keine standardisierten Protokolle. P2P-Systeme erfüllen damit zentrale Anforderungen nicht und können nicht als Grid aufgefasst werden.

2 Dienstgüteparameter sind u.a. Antwortzeit, Durchsatz, Verfügbarkeit und Sicherheit
3 engl. Peer = Gleichgestellter, Ebenbürtiger

3.4 Intragrid

Von einem Intragrid spricht man, wenn die beteiligten Rechner bezüglich Architektur und Betriebssystem nicht mehr homogen sind. Die Knoten eines Intragrid sind oft über mehr als eine Organisationseinheit verteilt. Die Scheduling-Aufgaben innerhalb des Grid werden komplexer. Zudem müssen Sicherheitsmaßnahmen implementiert werden, die dafür sorgen, dass wichtige Daten in einer Abteilung vor Zugriff anderer geschützt sind und eine Authentifizierung vor Zugriff auf die Daten und Ressourcen muss erfolgen.

3.5 Intergrid

Während Cluster und Intragrid einen festen Nutzerkreis und eine feste Zielsetzung haben, entspricht die Vision des Intergrid in etwa dem, was das Inter- gegenüber einem Intranet ausmacht. Die Benutzer sind nicht auf Teilnehmer an bestimmten Projekten oder an Firmenzugehörigkeiten gebunden, statt dessen können global Ressourcen zur Verfügung gestellt und genutzt werden, so wie sich Webseiten an jeden interessierten Nutzer richten. Die in einem Intergrid eingesetzte Technologie kann identisch zu der von Intragrids sein, häufig ergänzt um restriktive Sicherheitsmaßnahmen und Abrechnungssysteme.

3.6 Projekte

Die Vorteile des Grid-Computing machen es zu einem interessanten Konzept für viele Anwendungsbereiche mit großen Anforderungen an Rechenleistung und Speicherplatz. Aufgrund fehlender Standardisierung sind viele Grid-Projekte unabhängig voneinander entstanden. In vielen wissenschaftlichen Bereichen wie z.B. der Biologie oder der Physik wurden wegen des anfänglichen Fehlens domänenunabhängiger, frei verfügbarer Grid-Software eigene Grids entwickelt.

3.6.1 Eurogrid

Ein Beispiel für ein solches Projekt ist das Eurogrid, welches von der Europäischen Kommision unterstützt wurde. Das Projekt hatte als Ziele die Etablierung eines europäischen Grid-Netzwerkes, die Unterstützung der Eurogrid-Software-Infrastruktur, die Entwicklung von wichtigen Softwarekomponenten sowie deren Integration in das Eurogrid und sollte einen Beitrag zu der internationalen Grid-Entwicklung leisten.
Basierend auf der im Rahmen von Eurogrid entstandenen Technologien entwickelten sich verschiedene domänenspezifische Unter-Projekte, u.a. das Bio-Grid, das Meteo-Grid und das CAE-Grid. Das Bio-Grid zielte auf die Entwicklung einheitlicher Schnittstellen für biologische Anwendungen und Datenbanken. Ziel

dieser Entwicklungen war auch die Integration bestimmter Applikationen in das Unicore-Projekt[4], um die Entwicklung benutzerfreundlicher Tools zu unterstützen, die durch das Unicore-Projekt etabliert werden sollten. Das Meteo-Grid soll eine präzisere Vorhersage des Wetters erzielen. Durch eine Einteilung des Globus in Einheiten mit einer Größe von 2-25 km konnte eine bessere Vorhersage des Wetters und des Klimas erreicht werden. Je kleiner die Einheiten, umso genauer die Klimasituation, aber umso größer ist der Rechenaufwand. Statt dedizierter Großrechner wie der Earth Simulator könnte das Grid die nötige Rechenleistung bereitstellen. Das CAE-Grid (Computer Aided Engineering) unterstützte die Durchführung physikalischer Simulationen, insbesondere zur Aerodynamik. Das Eurogrid nutzt die Netzwerkinfrastruktur des Internet mit einem sicheren Zugang für die Eurogrid-Nutzer. Inzwischen ist das Projekt abgeschlossen [Mall04].

3.6.2 Globus-Projekt

Das Globus-Projekt hat seine Schwerpunkte auf die Entwicklung von Sicherheitskonzepten und den schnellen und sicheren Datentransfer innerhalb eines Grids gelegt. Die Implementierung des Projektes heißt Globus Toolkit, und wird unter anderem in weiteren Grid-Projekten eingesetzt, u.a. dem National Technologie Grid, dem European Data Grid und dem NASA Information Powergrid.

3.6.3 Unicore-Projekt

Die Entwickler von Unicore (Uniform Interface to Computing Resources) zielten bei der Implementierung auf einen benutzerfreundlichen, flexiblen und dynamischen Zugang zu den Ressourcen im Grid. Hierzu wurde eine Drei-Schichten-Architektur entworfen. Der *User Layer* beinhaltet eine grafische Benutzeroberfläche, damit der Unicore-Nutzer seine Jobs verwalten kann. Die zweite Schicht, genannt *Server Layer* wandelt einen Job des Nutzers in einen oder mehrere Batch Jobs um, und gibt diese an die *Target Layers*, die sich auf den Rechnern der angeforderten Ressource befinden, und die Batch Jobs abarbeiten [RaWi01].

3.6.4 Grid Interoperability Project

Ein weiteres aktuelles Projekt ist das Grid Interoperability Project (GRIP). Es widmet sich drei großen Bereichen, zum einen der Implementierung einer interoperablen Softwareschicht, der Entwicklung von Grid-Anwendungen vornehmlich aus Domänen, wie sie auch beim Eurogrid untersucht wurden, also biotechnische und meteorologische Anwendungen, und der Entwicklung von Standards für Grids. Das Globus-Projekt und das Unicore-Projekt wurden im GRIP vereinigt, um zwischen den Projekten Interoperabilität zu ermöglichen, und um die Erfahrungen aus beiden Projekten für die Definition von

4 Uniform Interface to Computer Resources

Standards zu nutzen. Die Unicore-User können durch GRIP Globus-Anwendungen nutzen und die besseren Sicherheitstandards des Globus-Grid wurden in das Unicore-Projekt integriert. Die Selbstständigkeit der Projekte blieb bei den Änderungen jedoch erhalten.
Trotz dieser bestehenden Bemühungen um Konsolidierung und Standardisierung wurde die Vision eines einzigen, global umfassenden Grids, das als Grundlage für dynamische Zusammenarbeit dienen kann, bisher nicht verwirklicht.
Aufgrund der mangelnden Interoperabilität der unabhängig voneinander entstandenen Grids ist ein Standard notwendig, um eine von Implementierung und Anwendung unabhängige Basis für das Grid Computing zu schaffen.

4. Standardisierung durch Grid Services

4.1 Grundlagen von OGSI

Das Global Grid Forum ist im Juni 1999 im Rahmen des von der NASA Numerical Aerospace Simulation (NAS) und dem Information-Power-Grid-Programm veranstalteten ersten *Grid-Forum-Workshop* entstanden. Ursprünglich wurden fünfzig Gäste erwartet, tatsächlichen kamen jedoch 150 Personen aus fünfzig Organisationen und vier Ländern. Heute zählen zu den Mitgliedern des GGF über 400 Organisationen aus mehr als fünfzig Ländern. Unter den Teilnehmern sind neben großen Softwareunternehmen auch akademische und staatliche Forschungsinstitute.
Das Global Grid Forum definiert einen dienstbasierten Standard zur Implementierung von Grid-Systemen, die Open Grid Service Infrastructure (OGSI). OGSI ist keine fertige Software, sondern eine Spezifikation, die als Grundlage für die Implementierung standardkonformer Grids dient. Das Global Grid Forum nutzt als Grundlage für die Grid Services die bereits etablierte *Web-Service*-Technik. Web Services sind ein dienstbasierter Standard und eignen sich für den Einsatz in heterogenen System-Umgebungen. Web Services ermöglichen den Aufbau einer dienstbasierten Architektur (*Service-Oriented Architecture*, SOA). Im folgenden sollen diese beiden Begriffe erklärt werden.

4.1.1 Dienstorientierte Architektur

Eine dienstorientierte Architektur (SOA) hat das Ziel, die Kopplung zwischen den kommunizierenden Softwaresystemen gegenüber objektorientierten Architekturen zur Entwicklung verteilter Anwendungen wie JavaRMI, CORBA und DCOM zu reduzieren. Die einzelnen Softwaresysteme bleiben autonom und kommunizieren lediglich mit Hilfe von Nachrichten. Die Kopplung der Systeme wird dadurch reduziert und sie können so leichter unabhängig voneinander weiterentwickelt werden. Eine reduzierte Kopplung ermöglicht auch eine größere Flexibilität bezüglich der unterstützten Protokolle und der

Rechnerarchitekturen von Clients und Servern. Abbildung 2 zeigt den grundlegenden Ablauf zur Nutzung einer SOA. Der Service Provider ist der Besitzer oder Anbieter eines Dienstes, der in einer Service Registry Informationen seines Dienstes abgelegt. Zur Nutzung des Dienstes muss der Service Requester in der Registry den Dienst lokalisieren. Ist dies erfolgt, kann er den Dienst z.B. in Form eines entfernten Methodenaufruf oder eines Nachrichtenaustausch nutzen. Ein Dienst ist nach [SOA04]:

> „A service is a function that is well-defined, self-contained, and does not depend on the context or state of other services."

Ein Dienst ist nach obiger Definition eine Funktion, die nicht abhängig von dem Zustand eines anderen Dienstes ist. Die Funktion oder Operation muss in sich abgeschlossen sein, also keinen Bezug zu anderen Diensten, oder Aufruf anderer Operationen haben und wohldefiniert sein. Eine andere Definition lautet:

> „A service is a network-enabled entity that provides a specific capability, for example, the ability to move files, create processes or verify access rights. A service is defined in terms of the protocol one uses to interact with it and the behavior expected in response to various protocol message exchanges (i.e., „service = protocol + behavior")." [FoKT01]

Der Unterschied zur ersten Definition ist der, dass ein Service nicht als Operation angesehen wird, sondern als ein Entität mit spezifischen Fähigkeiten. Zudem betont diese Definition, dass ein Service durch den Ablauf und Inhalt der Interaktion mit ihm charakterisiert wird.
Der Requester eines Dienstes sendet eine Nachricht an diesen und erwartet ein bestimmtes Verhalten als Antwort. Eine Service-Definition beschreibt die zu sendende Nachricht und das externe Verhalten.

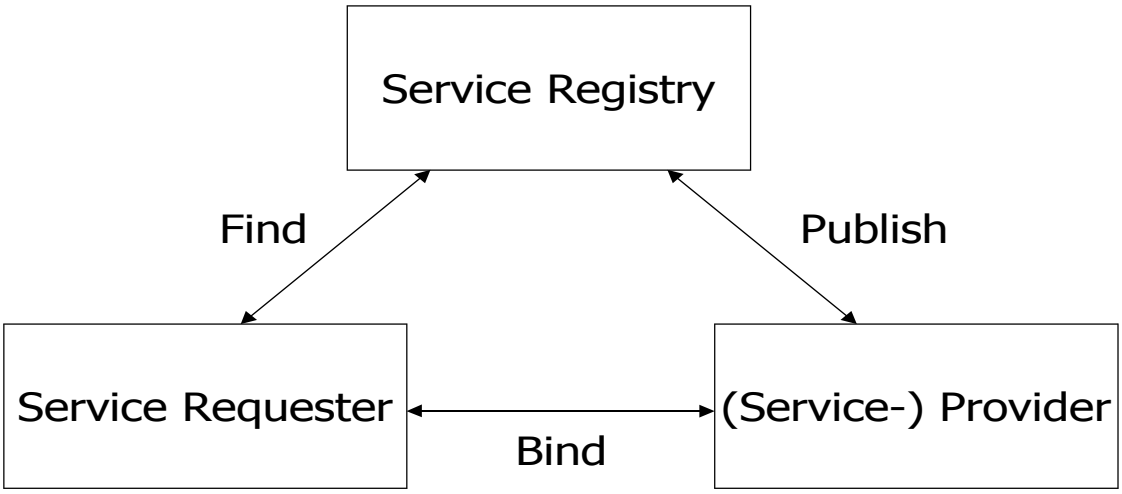

Abbildung 2: Rollen in einer dienstorientierten Architektur

Das SOA-Konzept macht keine Aussage über die technische Umsetzung der Teilnehmer und die Protokolle für deren Interaktion.

4.1.2 Web Service

Web Services sind eine technische Realisierung einer Service-Oriented Architecture. Web Services können verschiedene Techniken zur Umsetzung der Vorgänge Find, Bind, und Publish anbieten. Für das Binden wird bei den Web Services meist *SOAP* [5] benutzt [SOAP04]. Ein SOAP-Dokument hat einen oder mehrere Header, einen Body und einen Envelope, ist eine Einheit der Kommunikation und repräsentiert einen flexiblen Mechanismus um Daten darzustellen. Als Transportprotokoll für SOAP-Nachrichten wird meist das *Hypertext Transfer Protocol* (HTTP) eingesetzt, es können aber auch beliebig andere Transportprotokolle wie z.B. das *Simple Mail Transport Protocol* (SMTP) oder das *File Transfer Protocol* (FTP) genutzt werden. Der Anbieter eines Service publiziert die Schnittstelle seines Dienstes in Form eines WSDL-Dokuments. Die *Web Service Description Language* (WSDL) [CCM+04] ist eine plattform-, programmiersprachen- und protokollunabhängige XML-basierte Beschreibungssprache für die Schnittstellen von Web Services. Ein WSDL-Dokument beschreibt nur die Schnittstelle, es macht keine Vorgabe über die Implementierung des dahinterstehenden Dienstes. So kann ein Dienst in einer beliebigen Programmiersprache auf einer beliebigen Hardware- und Betriebssystem-Plattform implementiert sein. Erfolgt keine Änderung der Schnittstelle, kann die Implementierung jederzeit ausgetauscht werden.
Die wichtigsten Elemente einer solchen Schnittstellenbeschreibung sind die Elemente p*ortType*, *Operation* und *Message*. XML-Schema dient üblicherweise als Mittel, um die einfachen und komplexen *Types* eines *parts* zu beschreiben. Ein part beschreibt die Kommunikationsdaten einer Message. Eine Message kann mehrere parts enthalten. Eine Operation besteht aus einer oder mehreren Messages die als Eingabe-, Ausgabe- oder Fehlernachricht definiert werden können. Eine oder mehrere Operationen bilden einen Port Type der so die abstrakte Schnittstelle eines Web Service beschreibt. Welche Protokolle zur Kommunikation mit dem Web Service genutzt werden, beschreibt das *Binding*-Element. Das Binding definiert das Transferprotokoll (z.B. HTTP oder SMTP) und die Kodierung der Nachricht (z.B. SOAP oder XML-RPC). Der Name des Web Service und die Internetadresse, unter der er zu finden ist, wird durch das WSDL-Element *Port* beschrieben. Ein Service kann durch mehr als einen Port mit potentiell unterschiedlichen Bindings genutzt werden. Abbildung 3 zeigt das Zusammenwirken der WSDL-Elemente.
Das Suchen eines Web Services bei einer Service Registry wird ermöglicht durch UDDI (Universal Description, Discovery and Integration). UDDI dient zur Publizierung von Web Services. Die zugrundeliegende Registry speichert Informationen in Analogie zu Telefonteilnehmerverzeichnissen in *White Pages*, *Green Pages* und *Yellow Pages*. Die Green Pages enthalten im Wesentlichen die WSDL-Schnittstellen der angebotenen Web Services. Die White Pages beinhalten die Kontaktinformationen der Web-Service-Anbieter und eine allgemeine Service-Charakterisierung. Die Yellow Pages ermöglichen die Suche nach Diensten über Kategorien.

5 Ursprünglich stand SOAP für “simple object access protocoll”. Seit der Version 1.2 wird auf die Verwendung dieser Bezeichnung verzichtet.

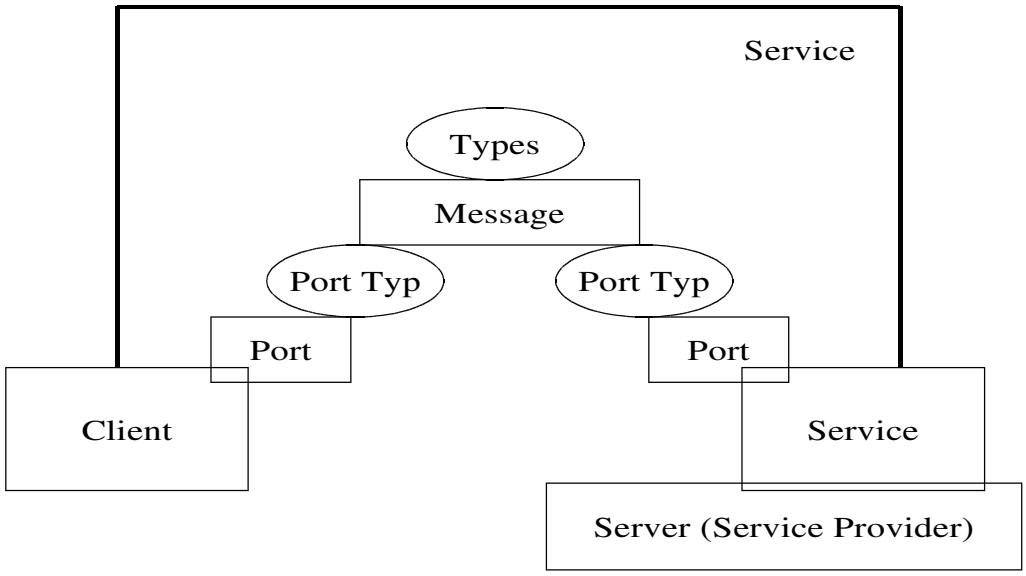

Abbildung 3: Zusammenspiel der WSDL-Elemente

Der grundsätzliche Ablauf bei der Nutzung eines Web Service beginnt mit der Suche des Service Requester in der Service Registry unter Verwendung von UDDI. Der Service Provider hat dazu seinen Service in der Registry veröffentlicht (in Form einer WSDL-Beschreibung). Der Provider kann für einen Dienst mehrere Kodierungs-/Protokollpaare anbieten (z.B. SOAP/HTTP, SOAP/SMTP), die Bindings. Nachdem der Service Requester den gesuchten Service gefunden hat bindet er sich an den Service, d.h. er wählt eines der Kodierungs-/Protokollpaare. Danach kann der Requester den Web Service durch den in WSDL beschriebenen Nachrichtenaustausch nutzen.

4.2 Grid Services

Web Services lösen bereits zahlreiche Interoperabilitätsprobleme bei der Interaktion heterogener verteilter Systeme. Sie bilden daher eine geeignete Grundlage für die Entwicklung dienstbasierter Grids. Ein Grid arbeitet mit Ressourcen, diese haben i.A. einen Zustand. Ein Web Service ist per Definition zustandslos beziehungsweise in der Schnittstellenbeschreibung von Web Services ist die Beschreibung eines Zustands des Service nicht vorgesehen. WSDL muss um diesen Aspekt vervollständigt werden. Daher wurde für die Schnittstelle eines Grid Service die Beschreibungssprache WSDL zur Grid Service Description Language (GWSDL) erweitert.

Die Erweiterung von WSDL sollte neben den erforderlichen Zustandsdaten auch eine Vererbung von Port Types beinhalten. GWSDL ermöglicht das Erstellen von Port Types durch die Beschreibung eigener Operationen, oder durch das Erben von Operationen von einem existierenden Port Type. Außerdem kann ein Port Type in GWSDL nicht nur Operationen beinhalten, sondern zusätzlich Service-Data-Elements (SDEs) beschreiben, die die Zustandsdaten repräsentieren. Die Beschreibung eines SDE beinhaltet neben dem

Namen und dem Typ auch die Informationen *minOccurs*, *maxOccurs*, *nilable*, *modifiable*, und *mutability*. MinOccurs und maxOccurs legen fest, wie oft der Wert auftreten darf. Der Wert *nil* ist ein spezieller Wert für einen Zustand, und die Bedeutung des Wertes ist abhängig vom SDE. Ist das Element modifiable mit true belegt, kann der Requester ein SDE verändern, ansonsten nicht. Mutability kann die Werte *static*, *constant*, *extendable*, und *mutable* annehmen. Ist das Attribut mutability static, wird der Wert des SDEs mit dem initialisiert, der in der WSDL-Beschreibung spezifiziert ist. Bei constant wird der Wert bei der Instanzierung des Dienstes für dessen Lebensdauer gesetzt. Extendable SDEs erlauben den Requestern neue Zustandsdaten hinzuzufügen (bis zum Wert von maxOccurs), jedoch nicht diese zu verändern. Dies erlaubt erst der Wert mutable. Die Anzahl der existierenden SDEs die erstellt werden dürfen hat OGSI nicht spezifiziert.

Um gewisse Basisfunktionalitäten einheitlich zu gestalten hat OGSI einige Standard-Port-Types definiert, die in Tabelle 1 dargestellt sind.

Schnittstelle	Operationen	Beschreibung
GridService	FindServiceData	Abfrage für einige Informationen der Grid-Service-Instanz (Handle, Reference). Abfragen von SDEs. *Abfragen sind erweiterbar.*
	SetTerminationTime	Setzen der Terminierungszeit für den Grid Service
	Destroy	Grid-Service-Instanz direkt beenden
NotificationSource	SubscribeToNotificationTopic	Anmelden für die Benachrichtigung für vom jeweiligen Dienst generierte Ereignisse
NotificationSink	DeliverNotification	Abmelden von der Benachrichtigung über Ereignisse
Registry	RegisterService	Registrieren von GSHs
	UnregisterService	Löschen von GSHs aus der Registrierung
Factory	CreateService	Erzeugen einer neuen Grid-Service-Instanz
HandleMap	FindByHandle	Gibt die Referenz zurück, die zu einem bestimmten Handle gehört

Tabelle1: OGSI Grid Service Schnittstellen [FKN02]

Somit kann der Nutzer eines Grid Service den aktuellen Zustand eines Dienstes abrufen, mit Operationen die im Port Type GridService beschrieben sind.

Abbildung 4 zeigt den Aufbau eines GWSDL-Dokuments in Form eines UML-Diagrammes, die Unterschiede beziehungsweise Neuerungen zu WSDL sind grau unterlegt. Neben den Operationen die ein Port Type enthalten kann, sind auch die Zustandsdaten (serviceData) spezifiziert. Die Vererbung, die in WSDL nicht beschrieben werden kann, wird in GWSDL durch das Attribut *extends* realisiert.

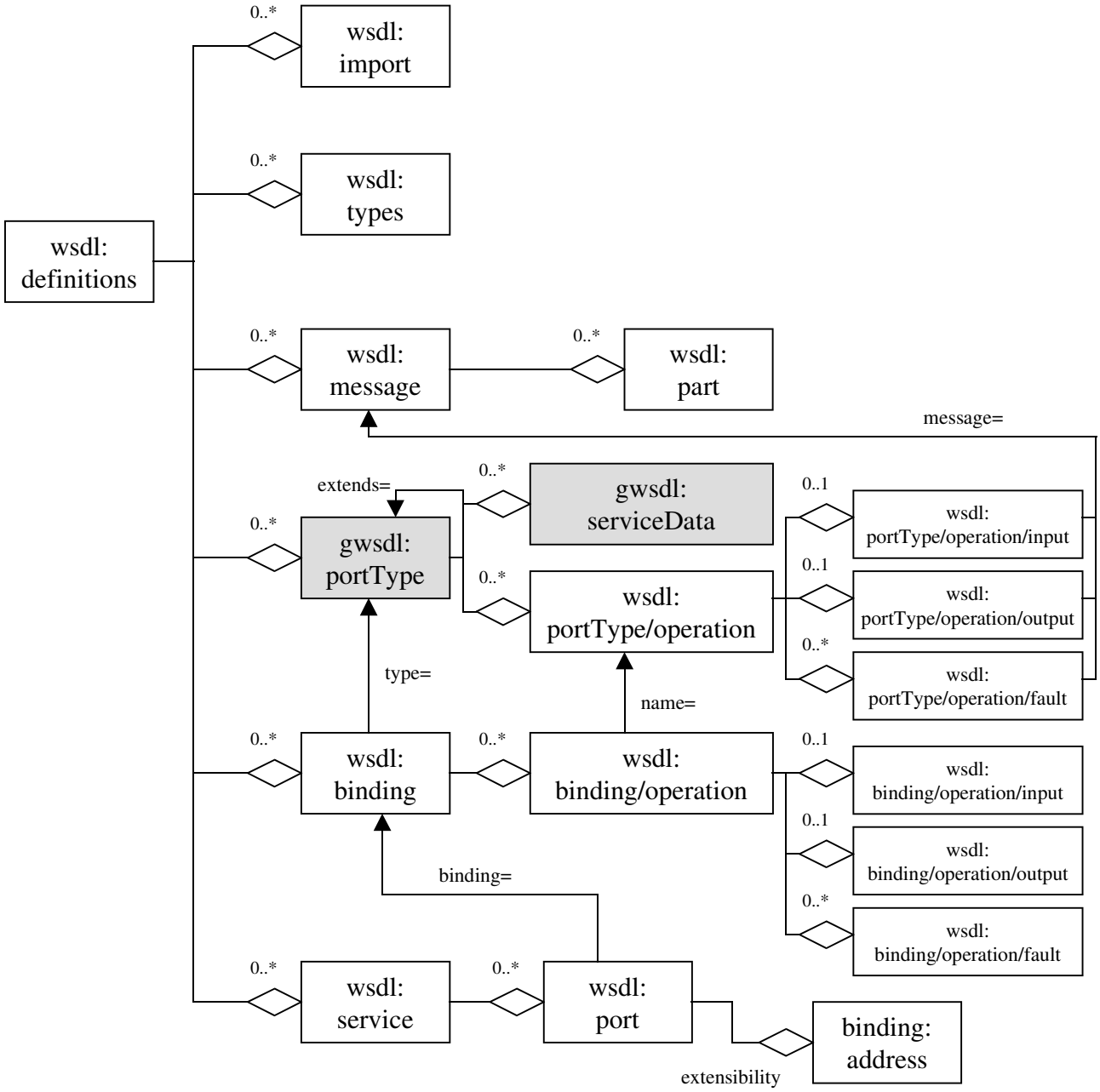

Abbildung 4: UML-Diagramm zur Beschreibung von GWSDL [OGSI04]

Nachdem der Zusammenhang zwischen Grid Services und Web Services beschrieben wurde, sollen nun zusätzliche Eigenschaften von Grid Services vorgestellt werden.

4.2.1 Eigenschaften von Grid Services

Jeder Grid Service ist ein Web Service, somit können Eigenschaften von Web Services auf den Grid Service übertragen werden. Das GGF hat zusätzlich zu den Eigenschaften von Web Services weitere in ihren Standard hinzugefügt [OGSI04]:

(1) Alle Grid Services erben vom Port Type GridService, der Operationen anbietet, um den Zustand einer Grid-Service-Instanz abzurufen, zu modifizieren oder die Instanz zu beenden.

(2) Web Services sind grundsätzlich nicht-transiente Dienste. Sie existiert auch, wenn sie nicht aktiv genutzt werden. Ein Grid Service dagegen kann auch transient sein, d.h. er wird nur für begrenzte Zeit benötigt und möglicherweise exklusiv für einen Client zur Verfügung.

(3) Um transiente Grid-Service-Instanzen zu erzeugen orientiert sich OGSI am bekannten Factory-Entwurfsmuster. Dazu existiert ein i.A. nicht-transienter Factory-Service, der den Standard-Port-Type Factory oder einen davon abgeleiteten Port Type implementiert.

(4) Für Grid Services existiert ein explizites und eine implizites Lebenszeit-Management. Transiente Grid Services die für einen Client erstellt werden, werden oft nur für eine gewisse Zeit benötigt. Eine Grid-Service-Instanz hat einen Zeitpunkt (kann auch *infinity* sein) bis zu dem er verfügbar ist (implizites Lebenszeit-Management). Ein Grid Services kann durch die *destroy*-Operation auch explizit beendet werden.

(5) Um einen Grid Service eindeutig zu identifizieren definiert OGSI ein zweistufiges Adressierungsschema:

   (a) Der eindeutige Identifikator den ein Client erhält, wenn er eine Instanz kreiert nennt man *Grid Service Handle* (GSH). Der GSH ist i.A. eine *Universal Resource Identifier* (URI[6]).

   (b) Um einen Grid Service nutzen zu können, muss man die GSH in eine *Grid Service Reference* (GSR) umwandeln. Ein GSH kann mit Hilfe eines *Handle Resolver* in eine oder mehrere GSRs überführt werden. Der Handle Resolver ist ein spezieller Service, der in GWSDL beschrieben ist. Eine GSR enthält alle nötigen Informationen, um ihn nutzen zu können. GSRs werden üblicherweise als GWSDL-Datei realisiert, OGSI erlaubt aber auch anders aufgebaute GSRs. Dies erlaubt verteilten Programmierarchitekturen wie CORBA ohne Änderungen ihre Schnittstelle als Grid Service gemäß OGSI zu beschreiben. Während ein GSH global eindeutig und dauerhaft gültig ist, haben GSRs eine begrenzte Gültigkeit.

   (c) Die GSHs und GSRs können in einem Bündel an die Clients gesendet werden. Dieses Bündel nennt man *Grid Service Locator*. Hat ein Client den Factory Service eines transienten Grid Service aufgerufen, sendet der Factory Service einen Locator mit einem Handle und gültige Referenzen an den Client zurück. Speichert der Client den Locator bevor er die Referenzen nutzt, können diese vor Gebrauch ungültig werden. Oft speichert die Anwendung daher auch nur den GSH persistent ab und erfragt erst bei der Nutzung des Dienstes eine gültige GSR, mit Hilfe eines Handle Resolvers.

6 URIs sind der Standard, um Objekte im Internet eindeutig zu identifizieren.

4.2.2 Nutzung eines Grid Services

Abbildung 5 zeigt den Ablauf zur Nutzung eines Grid Service. Grid-Anwendungen suchen zunächst in der Registry nach einer Factory, wenn sie noch keine GSH und/oder GSR für den Dienst besitzen. Dazu benötigt die Grid-Anwendung zumindest für die Registry und den Handle Resolver eine gültige Referenz. Die Grid-Anwendung erhält von der ausgewählten Factory eine GSH und eventuell mehrere GSRs auf eine Instanz. Diese kann entweder neu erzeugt worden sein, oder auch eine bereits existierende Instanz sein. Für spätere Referenzierungen trägt die Factory den Handle im Handle Resolver ein, und der instanzierte Grid Service meldet sich als Referenz des Handles beim Handle Resolver an. Die Grid-Anwendung kann den Grid Service über diese Referenz nutzen.
Eine Referenz auf eine Instanz kann ungültig werden. Der Grid Service kann nur weiter genutzt werden, wenn sich der Client an den Handle Resolver wendet, um eine neue Referenz zu erhalten.
Um die Nutzung eines zustandsbehafteten Grid Services darzustellen, soll das folgende Beispiel für eine Grid-Service-Schnittstelle dienen:

```
<wsdl:definitions
        xmlns:ogsi="http://www.gridforum.org/namespaces/2003/03/OGSI"
        xmlns:sd="http://www.gridforum.org/namespaces/3003/03/serviceData"
        xmlns:wsdl="http://schemas.xmlsoap.org/wsdl"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:counter="http://www.gridforum.org/namespaces/3003/05/ogsiprimer/counter/basic"
        targetNamespace="http://www.gridforum.org/namespaces/3003/05/ogsiprimer/counter/basic">
        <wsdl:types>
                <xs:schema/>
        </wsdl:types>
        <wsdl:message name="increaseMsg">
                <wsdl:part name="value" type="xs:positiveInteger">
        </wsdl:message>
        <wsdl:portType name="counterPortType" extends="ogsi:GridService">
                <wsdl:operation name="increase">
                        <wsdl:input message="increaseMsg">
                </wsdl:operation>
                <sd:serviceData name="counterValue"
                        type="xs:positiveInteger"
                        minOccurs="1"
                        maxOccurs="1"
                        mutability="mutable">
                </sd:serviceData>
        </wsdl:portType>
</wsdl:definitions>
```

Abbildung 4: Grid-Service-Schnittstelle

Abbildung 4 beinhaltet nicht die notwendigen binding und service-Elemente und beschreibt so nur eine abstrakte Schnittstelle. Der Grid Service ermöglicht das Erhöhen einer Variable *counterValue*. Hierzu steht die Operation *increase* zur Verfügung. Der Wert der Variable wird durch ein SDE repräsentiert. Auf den

Wert kann nun mit Hilfe von Operationen die der OGSI-Standard-Port-Type GridService zur Verfügung stellt zugegriffen werden. Die Aktualisierung der SDE kann auf unterschiedliche Arten erfolgen. OGSI hat hierfür Operationen spezifiziert, es können aber auch eigene Operationen definiert werden. hat. Ein SDE könnte immer dann aktualisiert werden, wenn eine Operation auf das SDE erfolgt, oder in einem bestimmten Zeitintervall. Das SDE counterValue wird im Beispiel durch die Operation increase geändert und aktualisiert. Der Wert des SDE ist genau einmal vorhanden, begrenzt durch minOccurs und maxOccurs.

4.3 Grundlagen von OGSA

OGSI ist eine formale Spezifikation auf der OGSA basiert. OGSA definiert Operationen, welche die grundlegenden Operationen von OGSI erweitern um den Applikationen eine umfangreiche Funktionalität anbieten zu können. Abbildung 4 veranschaulicht die Einordnung der OGSA Services. Zu den Services gehören die Grid Core Services, die Dienste zur Sicherheit, Dienst-Kommunikation, und Dienst-Koordination anbietet. Für den Zugriff auf CPU-Ressourcen werden Grid Program Executions Services zur Verfügung gestellt und für den Zugriff auf Daten die Grid Data Services. Zusätzlich existieren domänenspezifische Dienste.

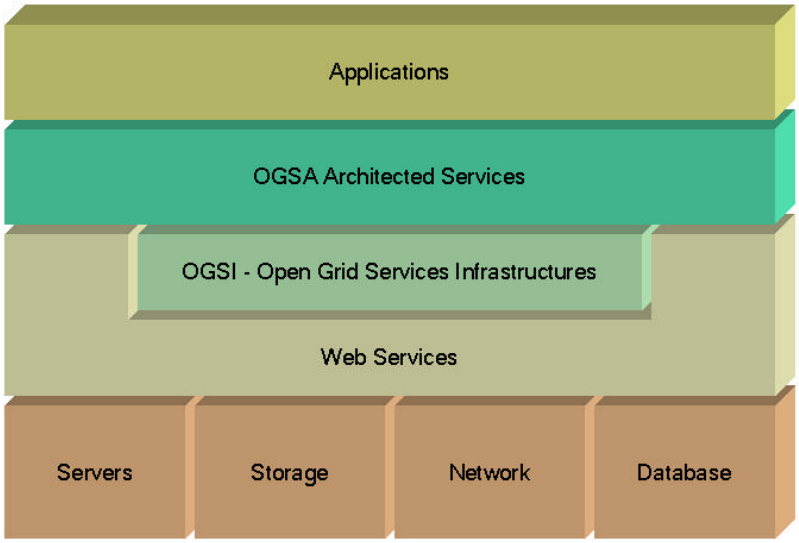

Abbildung 5: OGSA Architektur

Die OGSA Architected Services bauen nicht nur auf der vorhandenen OGSI-Spezifikation auf, die OGSA working groups benötigen für die Anforderungen an ein Grid erweiterte Web-Service-Funktionalitäten.

5. Zusammenfassung

Grid Computing beschreibt ein Konzept zur Nutzung heterogener und verteilter Ressourcen. Grids sind nicht zu verwechseln mit bereits vorhandenen Technologien wie Clustern oder P2P-Netzen. Projekte die den Grid-Gedanken realisieren sind getrennt voneinander entstanden, da ein Standard fehlte. Die Standardisierung erfolgt durch das Global Grid Forum, das die bereits vorhandene Web-Service-Technologie nutzt. Da Web Services zustandslos sind, mussten für Grid Services einige Ergänzungen erfolgen.

Grids werden i.A. keine Hochleistungsrechner ersetzen können. Viele Probleme benötigen außer einer Menge Rechenleistung auch geringe Latenzzeiten bei der Kommunikation, die nur eng vernetzte Hochleistungsrechner erbringen können. Die Grid-Software sollte vergleichbar mit dem Betriebssystem für einen einzelnen Rechner sein. Eine Abstraktion von Verteilungs- und Heterogenitätsaspekten sollte vorliegen, so dass Anwendungsentwickler und Endnutzer den Eindruck haben an einem einzigen leistungsfähigen Rechner zu arbeiten.

6. Ausblick

Bisher hat primär das Global Grid Forum an einer Spezifizierung und Beschreibungssprache für zustandsbehaftete Dienste gearbeitet. Hierbei entstand die OGSI-Spezifikation und mit GWSDL einer WSDL-Erweiterung, die u.a. Vererbung und SDEs ergänzt. An der Weiterentwicklung von Web Services beteiligten Unternehmen wie IBM oder HP haben den Bedarf für zustandsbehaftete Dienste unabhängig vom Grid Computing erkannt und eine funktional mit OGSI vergleichbare Spezifikation, das *Web Service Ressource Framework* (WSRF) entwickelt. Beide Spezifikationen beinhalten die Möglichkeit, Zustandsdaten zu beschreiben. Im WSRF wurde dies in der WS-ResourceProperties-Spezifikation beschrieben, analog zu den ServiceDataElements von OGSI. Die OGSI-ServiceGroup findet ihre Entsprechung in WS-ServiceGroup, das Pendant zu den NotificationSource- und NotificationSink-Schnittstellen bildet WS-Notification. Das Global Grid Forum plant, sich künftig an der Spezifikation des Web Service Resource Framework zu orientieren.

Die Entwicklung des Grid durchläuft wie auch andere Technologien die Phasen der anfänglichen Idee über die Spezifizierung und über den Feldtest bis zum produktiven Einsatz. Derzeit befindet sich das Grid in dem Übergang von der Phase der Feldtests zum Einsatz in der realen Umgebung.

Das Ziel der Aktivitäten zur Entwicklung eines global umfassenden Grids ist letztendlich das Intergrid, ein Utility-Grid für Computer-Ressourcen, an dem jeder teilnehmen kann, um Ressourcen anzubieten, oder bei Bedarf nachzufragen.

7 Literaturverzeichnis

[Bers02] Viktors Berstis: Fundamentals of Grid Computing, IBM Corp., http://www.redbooks.ibm.com/redpapers/pdfs/redp3613.pdf, 2002

[BoDG01] Bote-Lorenzo,Dimitriadis,Gómez-Sánchez: Grid Characteristics and Uses: a Grid Definition, University of Valladoid, 2001

[CCM+04] Christensen, Curbera, Meredith, Weerawarana: W3C, http://www.w3.org/TR/2001/NOTE-wsdl-20010315, Stand: Juni 2004

[FKN02] Foster, Kesselmann, Nick, Tuecke: The Physiology of the Grid, Globus Projekt, http://www.gridforum.org/ogsi-wg/drafts/ogsa_draft2.9_2002-06-22.pdf, 22. Juni 2002

[FoKT01] Foster,Kesselman,Tuecke: The Anatomy of the Grid, Enabling Scalable Virtual Organizations, http://www.globus.org/research/papers/anatomy.pdf, 2001

[Fost02] Foster: What is a Grid?, Globus Project, http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf, 20. Juli 2002

[Mall04] Mallmann: Application Testbed for European GRID Computing Juelich, http://www.eurogrid.org, Stand: Mai 2004

[OGSI04] o.V.: Open Grid Services Infrastructure Primer, Global Grid Forum, https://forge.gridforum.org/projects/ogsi-wg/document/draft-ggf-ogsi-gridserviceprimer/en/1, Stand: Juni 2004

[RaWi01] Michael Rambadt, Philipp Wieder: Unicorn-Globus: Interoperability of Grid Infrastructures, Forschungszentrum Jülich, http://www.grid-interoperability.org/unicore_globus_interop_paper_final.pdf, 2002

[ReSc04] Reinefeld/Schintke: Grid Services, Springer Verlag, Informatik Spektrum, 26. April 2004

[SOA04] o.V.: Service-oriented architecture (SOA) definition, http://www.service-architecture.com/web-services/articles/service-oriented_architecture_soa_definition.html, Stand: Juni 2004

[SOAP04] o.V.: SOAP Version 1.2 Part 0; Primer, http://www.w3.org/TR/2003/REC-soap12-part0-20030624/ Stand: Juni 2004