

Dienste und Standards für das Grid Computing

Alexander Reinefeld und Florian Schintke

Zuse-Institut Berlin (ZIB)
Takustraße 7, 14195 Berlin
{reinefeld,schintke}@zib.de

Abstract:

In der noch jungen Geschichte des Grid Computing wurden bereits mehrere Standards für serviceorientierte Grids vorgeschlagen und diskutiert: OGSA, OGSF und WSRF, sowie Grid-spezifische Erweiterungen der Web Services. Sie alle haben das Ziel, zustandsbehaftete, transiente Dienste zur kooperativen Nutzung im Grid freizugeben. Ortstransparenz wird durch Virtualisierung ermöglicht: Nicht der interne Aufbau oder gar die Implementation eines Dienstes stehen im Vordergrund, sondern die Definition der Schnittstellen und Protokolle, über die der Dienst angesprochen werden kann.

Dieser Artikel soll das gewachsene und scheinbar undurchdringliche Dickicht der verschiedenen Grid-Architekturen und -Standards etwas lichten und auf diese Weise zu einem besseren Verständnis der wichtigsten Konzepte des Grid Computing sowie ihrer Hintergründe beitragen.

1 Grid Computing

Grid Computing, das sich mit der kooperativen Nutzung geographisch verteilter Ressourcen beschäftigt, ist nicht, wie man vielleicht zunächst denken mag, eine bloße Neuauflage der traditionellen verteilten Systeme, sondern vielmehr der Versuch, auf höherer Ebene heterogene Ressourcen über Verwaltungsdomänen hinweg gemeinschaftlich zu nutzen. Es geht also nicht darum, eine einzelne Applikation oder Anwendungsgruppe mit verteilten Prozessen, die auf einen Anwendungskontext zugeschnitten sind, zu realisieren, sondern um die Schaffung einer umfassenden Infrastruktur zur Nutzung verteilter Ressourcen.

Entstanden ist das Grid Computing in den frühen neunziger Jahren durch die Kopplung von Hochleistungsrechnern über schnelle Datenleitungen, um damit speicher- oder rechenintensive Aufgaben lösen zu können. Die damals gebräuchlichen Bezeichnungen *Metacomputing* oder *Hypercomputing* verweisen schon auf das Ziel der koordinierten Ressourcennutzung für große Rechenaufgaben. Zwar ist die verfügbare Rechen- und Speicherleistung einzelner Systeme seitdem um ein Vielfaches angewachsen, das Verlangen nach immer leistungsfähigeren Ressourcen ist aber dennoch geblieben und ist, allerdings neben anderen Aspekten, weiterhin eine wichtige Motivation zum Einsatz von Grid Technologien.

Was verteilte Hochleistungsrechner gemeinsam nutzbar macht, kann auch in kleinerem

Maßstab von Vorteil sein, und so gesellte sich neben das Hochleistungsrechnen auch das Cluster- und schließlich das Desktop-Computing, um dem Nutzer an seinem PC die geballte Rechenleistung entfernter Computer auf Abruf bereitzustellen. Analog zum Stromnetz, dem „Power Grid“, kam die Metapher des „Information Power Grid“ [9] auf – der „Rechenleistung aus der Steckdose“.

Spätestens ab diesem Zeitpunkt, wo das Grid Computing in Abkehr von seinem ehemals elitären Ziel des gekoppelten Hochleistungsrechnens die breite Masse der Cluster- und Desktop-Systeme erreichte und damit eine potentiell große Nutzergemeinde bekam, wurde klar, dass die Dienste, die für das Grid Computing notwendig sind, möglichst unabhängig von Implementationen und Applikationen standardisiert werden müssen, um das Grid Computing auf eine solide Basis zu stellen. Zu dieser Aufgabe treffen sich Forscher und Anwender im *Global Grid Forum (GGF)* [15], dem weltweiten Standardisierungsgremium aller Gridforscher.

Grid-Systeme weisen, in Abgrenzung zu anderen Organisationsformen, vier charakteristische Eigenschaften auf:

- *Lokale Autonomie*: Die Ressourcen sind über mehrere lokale Verwaltungseinheiten hinweg verteilt und werden von den jeweiligen Eigentümern nach deren Verfahrensregeln (policies) in den Betrieb eingestellt und wieder herausgenommen. Die Grid-Software hat keinen Einfluß auf lokale Entscheidungen, sie muss sich vielmehr daran anpassen.
- *Heterogenität der Ressourcen*: Die Ressourcen im Grid sind sehr vielfältig. Sie umfassen Rechner verschiedener Bauart und Leistung, Datenspeichersysteme, Netzwerke sowie Spezialgeräte wie beispielsweise Satellitendatenquellen, medizinische Apparaturen oder Sequenziermaschinen. Auch Softwarepakete und Expertenwissen („human in the loop“) sind in das Grid eingebunden.
- *Skalierbarkeit*: Der Umfang von Grid-Systemen reicht von einigen lokal verbundenen Knoten bis zu weltumspannenden Systemen. Bei Bedarf müssen Grid-Dienste ohne größere Synchronisations- oder Koordinationsverluste von mehreren Knoten gleichzeitig erbracht werden, d.h. sie müssen mit den Anforderungen wachsen.
- *Dynamik und Adaptivität*: Aufgrund der hohen Anzahl beteiligter Komponenten und der inhärenten Unzuverlässigkeit der Netzwerke und Knoten, können Einzelausfälle im Grid nicht verhindert werden. Grid-Software muss in der Lage sein, flexibel auf Fehlersituationen jeder einzelnen Komponente zu reagieren.

2 OGSA – Grundlegende Systemarchitektur für das Grid Computing

In einer serviceorientierten Architektur werden die Komponenten des verteilten Systems durch ihre Schnittstellen (API, GUI) und ihr Verhalten definiert. Die Implementation spielt dabei nur eine untergeordnete Rolle. Die Interaktion zwischen den einzelnen Diensten regeln definierte Protokolle.

Die *Open Grid Service Architecture (OGSA)* [11, 12] ist eine erweiterbare Systemarchitektur für das Erbringen, das Zusammenspiel und den Konsum von Diensten im Grid. OGSA wurde von der gleichnamigen Working Group des Global Grid Forum bereits im Jahr 2002 als Standardisierungsvorschlag vorgelegt und befindet sich – aufgrund seiner Komplexität und der weitreichenden Konsequenzen – immer noch in der Diskussionphase.

OGSA definiert ein Komponentenmodell, das es Anwendungen ermöglicht, auf einfache Weise im Grid angebotene Dienste zu nutzen und miteinander zu koppeln. Es geht besonders auf die grundlegenden Probleme in Grid-Umgebungen ein, die bisher zum Teil für jedes Projekt individuell gelöst werden mussten: die Identifikation, Authentifikation und Autorisation von Teilnehmern und Diensten, das Auffinden und der Aufruf von Diensten, das Aushandeln von Verfahren, die Überwachung und Kontrolle verteilter Anwendungen und die Einrichtung und Unterstützung dynamischer Arbeitsgruppen (virtual organizations).

Der OGSA-Standardisierungsvorschlag [12] beschreibt neben einigen typischen Anwendungsszenarien die wichtigsten Aspekte, die bei der Konzeption von Grid-Systemen beachtet werden müssen:

- Unterstützung von heterogenen Ressourcen (Auffinden, Anfrage und Lebenszeit-Management von verteilten Diensten und Ressourcen),
- Handhabung der Verfahrensregeln lokaler Verwaltungseinheiten (Abbildung der Verfahrensregeln, Transparenz, Sicherheit, Abrechnung),
- Nutzung der Ressourcen (Reservierung, Überwachung, Kontrolle, Nutzungsprofile, Prognosen),
- Job-Ausführung und Qualitätsüberwachung (Co-Scheduling, Job-Überwachung, Workflow-Management, Dienste-Komposition, Dienstgütevereinbarungen),
- Management verteilter Daten (Datenzugriff, Datenintegration, Metadatenverwaltung, Replikation, Caching, Staging, Platzierung),
- Sicherheit (Authentifikation, Autorisation, Isolation, Delegation, Umgang mit Firewalls),
- Skalierbarkeit (Vermeidung zentraler Komponenten, dynamisches Hinzufügen und Entfernen von Ressourcen),
- Verfügbarkeit (Umgang mit unzuverlässigen Komponenten, transparente Ausfallbehandlung).

Für diese und weitere Aspekte definierten Arbeitsgruppen des Global Grid Forum generische Schnittstellen, die der OGSA-Architektur genügen. Die konkreten Implementationen sind austauschbar, was den Aufbau modularer Systeme ermöglicht.

3 Web Services – Standard für Dienste in Verteilten Systemen

Web Services [21] wurden als Programmier- und Systemparadigma für verteilte Systeme mit autonomen Teilnehmern und Diensteanbietern entwickelt. Sie haben mittlerweile eine recht weite Verbreitung gefunden, nicht nur im akademischen Bereich, sondern auch im kommerziellen Umfeld, hier insbesondere in B2B-Anwendungen. Die Abgrenzung, welche Methoden und Techniken zu Web Services zu zählen sind, fällt schwer. Allgemein versteht man unter Web Services ein Bündel von Technologien zur Beschreibung und Implementation von Schnittstellen, Dateiaustauschformaten, Qualitätseigenschaften des Datenaustauschs, zur Registrierung und Komposition von Komponenten sowie zur Beschreibung der Sicherheit im Austausch mit Komponenten [21].

Zentrales Anliegen der Web Services ist die Unterstützung der Interaktion zwischen geographisch verteilten Computern. Wichtige Bestandteile sind daher die Definition von Schnittstellen (WSDL) und Kommunikationsprotokollen (SOAP), die im allgemeinen über Standard-Internetprotokolle (HTTP, SMTP) übertragen werden. Um einen Web Service aufrufen zu können, muß er adressierbar sein, d.h. es muß einen gültigen URI (uniform resource identifier) geben, über den der Dienst angesprochen werden kann. Eine weitere wichtige Eigenschaft von Web Services ist ihre Autonomie: Es gibt grundsätzlich keine Möglichkeit, die service-interne Verarbeitung von außen zu beeinflussen, was natürlich die Definition und Einhaltung von Qualitätseigenschaften verkompliziert.

3.1 Web Services für Dienste im Grid?

Prinzipiell könnten Dienste im Grid mit Hilfe von Web Services realisiert werden – und sind es teilweise auch. Grid-Dienste können mit der Web Services Definition Language (WSDL) definiert werden, sie können sich mittels SOAP untereinander verständigen (service binding) und den Web Services Security Layer nutzen.

Allerdings gibt es zwei wesentliche Unterschiede zwischen den Diensten, die von Web Services einerseits bzw. Grid-Services andererseits bereitgestellt werden: Web Services sind persistent und zustandslos, während Grid-Dienste oft transient und zustandsbehaftet sind.

Daher existieren wesentliche Unterschiede zwischen den beiden Standards in den jeweiligen Methoden zur Adressierung, zum Lebenszeit-Management und zur Handhabung von Zustandsinformationen. So hat man sich beispielsweise bei den Grid-Diensten – sowohl in OGSF als auch in WSRF – auf ein zweistufiges Namensschema für benannte Dienste geeinigt, das bei OGSF aus abstrakten Grid Service Handles (GSH) und konkreten Grid Service References (GSR) besteht und im WSRF-Standard durch das Namensschema des WS-Adressing-Pakets definiert ist. Die Datenelemente (*ServiceData* in OGSF bzw. *WS-ResourceProperties* in WSRF) enthalten für jeden Dienst Zustandsinformationen und weitere Metainformationen. Hinzu kommen Schemata zur einheitlichen Instanziierung und für das Lebenszeit-Management.

Oberflächlich betrachtet sind dies nur kleine syntaktische Erweiterungen der Web Ser-

vices. Sie bilden jedoch die Grundlage für die Realisierung zuverlässiger, ortstransparenter Dienste im Grid über die Grenzen lokaler Verwaltungsdomänen hinweg.

4 OGSi – Standard für Dienste im Grid

Alle Dienste in OGSi sind zunächst einmal Web Services. Darüber hinaus müssen sie einige Basisfunktionen unterstützen, die im *Open Grid Service Infrastructure (OGSi)*-Standard [25] festgelegt sind [22]. Dazu gehören Aspekte wie: maschinenlesbare Schnittstellendefinition von Diensten (Funktionen, Datenelemente), Erweiterbarkeit von Diensten, Verfügbarkeitszeitraum (*lifetime*) von Diensten, Auffinden und Referenzieren von Diensten (*HandleResolver*), einheitliche Fehlerbehandlung in Diensten, Benachrichtigung über Änderungen an Diensten, Instanziierung von Diensten (*Factory*) und die Verwaltung von Gruppen von Diensten.

Bei näherer Betrachtung des OGSi-Standards fallen Analogien zu den bekannten Entwurfsmustern (design patterns) der objektorientierten Programmierung auf [14]. Dort existiert auch das Konzept der „Factory“, die zur Instanziierung von Diensten genutzt wird. Die Übertragung anderer Entwurfsmuster, wie z.B. Adapter, Bridge, Mediator, Observer, Proxy, oder Strategy auf Web- und Grid-Dienste erscheint ebenfalls sinnvoll und vielversprechend, um Grid-Dienste einfacher flexibel und interoperabel zu gestalten.

4.1 Schnittstellen

Dienst-Schnittstellen werden mit Hilfe der *Web Service Description Language (WSDL)* definiert. Bei OGSi-Diensten können Funktionen und Datenelemente angeboten werden. Ein Dienst mit mehreren Funktionen und Datenelementen wird durch einen sogenannten *portType* beschrieben. Dieser Begriff ist schon Teil der WSDL-Spezifikation und ist eine Beschreibung eines Dienstes, der über einen Netzwerk-Port angesprochen werden kann. Die konkrete Implementation eines *portType* entspricht dann einem Daemon, der auf Anfragen an einem Netzwerk-Port wartet.

4.2 Zustandsinformationen

OGSi-Dienste können einen Zustand besitzen, der sich in den Datenelementen des Dienstes widerspiegelt. Für diese Datenelemente (*serviceData*) kann über die Klassen *static*, *constant*, *extendable* und *mutable* festgelegt werden, ob die Zustanswerte statisch im WSDL-Dokument definiert sind, ob sie nur bei der Instanziierung des Dienstes gesetzt werden, ob weitere Statuswerte hinzugefügt werden können oder ob Werte im Laufe des Dienstes entfernt werden können.

Um Datenelemente auch über ihren Namen auffindbar zu machen, stellt jeder OGSi-Dienst

eine Reflektion (introspection) der Datenelemente und eine Methode *queryByServiceDataNames* zur Verfügung.

Ähnlich der Vererbungshierarchie in der objektorientierten Programmierung [14] können Dienste als Erweiterung von anderer Dienste beschrieben werden, wobei auch Erweiterungen von Kompositionen (Mehrfachvererbung) möglich sind.

4.3 Lebenszeit-Management

Um mit der Dynamik in Grid-Umgebungen besser umgehen zu können, gibt es ein explizites und ein implizites Lebenszeit-Management für OSGI-Dienste.

Beim impliziten Lebenszeit-Management werden für OSGI-Dienste Zeitpunkte angegeben, bis zu denen sie zugreifbar sein werden. Als globales Zeitformat wird die GMT-Zeit verwendet, das um den Wert *infinity* erweitert wurde. Jeder Dienst besitzt zwei definierte Zeitpunkte, *after* und *before*. Ein Dienst terminiert frühestens nach dem Zeitpunkt *after* aber stets vor dem Zeitpunkt *before*.

Mit der *destroy* Methode kann ein OSGI-Dienst auch explizit beendet werden. Neue Dienstinstanzen werden über eine Factory erzeugt.

4.4 Dienste-Referenzierung

Zur Adressierung von Diensten in dynamischen Umgebungen nutzt OSGI ein zweischichtiges Referenzierungssystem. Ein sogenannter *Grid Service Handle* (GSH) ist ein abstrakter Verweis auf einen Dienst, unabhängig von einer konkreten Dienste-Instanz. Ein GSH ist unabhängig davon, wie die Semantik eines Dienstes erbracht wird. Verschiedene Implementationen des gleichen GSH können den Dienst auf unterschiedliche Arten (z.B. verteilt oder zentral) erbringen.

Ein GSH kann über den *HandleResolver* in eine oder mehrere *Grid Service Reference* (GSR) umgewandelt werden, die auf Implementationen des Dienstes verweisen und alle nötigen Informationen enthalten, um diesen Dienst nutzen zu können. Das Format des GSR ist abhängig vom zu nutzenden Aufrufmechanismus. Es kann zum Beispiel für einen SOAP-Dienst eine WSDL-Beschreibung oder für einen Corba-Dienst eine IOR-Beschreibung sein.

Grid-Anwendungen nutzen in ihrem Programmtext GSHs, um persistent auf zu nutzende Dienste zu verweisen. Ähnlich wie virtuelle Speicheradressen in einem Computer in reale Speicheradressen umgesetzt werden, wird hier zur Laufzeit festgestellt, wo der geforderte Dienst verfügbar ist, indem der GSH in GSRs aufgelöst wird. Die Applikation nutzt dann die GSRs, um den Dienst tatsächlich anzusprechen.

Um Informationen über Dienste und Dienste-Referenzen gemeinsam austauschen zu können, wurde in OSGI die Struktur des *ServiceLocator* geschaffen, die mehrere GSHs, GSRs und auch Definitionen von *portTypes* aufnehmen kann.

OGSI schafft also zunächst die Basis zur Beschreibung von Grid-Diensten und definiert zusätzlich eine erste Infrastruktur für Grid-Dienste (Factory, HandleResolver, usw.).

4.5 Grid-Services im Zusammenspiel

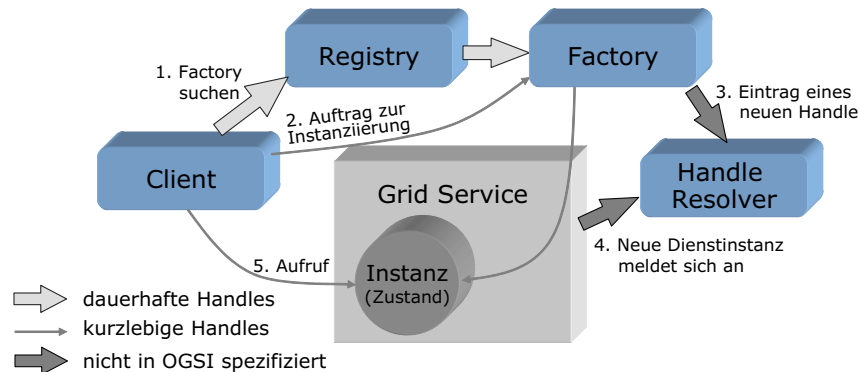


Abbildung 1: Die Hauptkomponenten des OGSI-Rahmenwerkes in einem einfachen Szenario.

Ein einfaches Szenario für den Aufruf eines OGSI-Dienstes ist in Abbildung 1 gezeigt. Grid-Anwendungen, die einen OGSI-Service nutzen wollen, suchen zunächst in einem Dienstverzeichnis nach einer Factory für diesen Dienst (1). Die Factory wird anschließend beauftragt eine Instanz des gewünschten OGSI-Dienstes zu erzeugen, oder eine passende bereits vorhandene Instanz auszuwählen (2). Für die spätere Referenzierung trägt die Factory einen Handle des Dienstes im Handle Resolver ein (3) und der instanziierte OGSI-Dienst meldet sich als Referenz des Handles beim Handle Resolver an (4). Schließlich kann der OGSI-Dienst von der Anwendung über diese Referenz genutzt werden (5). Das Lebenszeit-Management legt die Dauer der Gültigkeit kurzlebiger Handles über die Lebenszeit des entsprechenden Dienstes fest. Ist diese Zeit abgelaufen, muss sich eine Anwendung an den Handle Resolver wenden, um einen aktuellen Verweis auf den Dienst zu finden, denn in der Zwischenzeit könnte der Dienst auf andere Ressourcen verlagert worden sein, weil sich beispielsweise die Verfügbarkeit oder Last der Ressourcen geändert hat. Durch den Mechanismus des Handle Resolvers werden langlebige Gridanwendungen von solchen Änderungen nicht negativ beeinflusst.

5 WSRF – Der neue Standard für Dienste im Grid

Als Nachfolger des (noch recht jungen) OGSI-Standards wurde *Web Services Resource Framework (WSRF)* [4] vorgeschlagen [5], der, im Gegensatz zu OGSI auf Strukturänderungen im WSDL verzichtet und stattdessen die gleiche Funktionalität wie OGSI

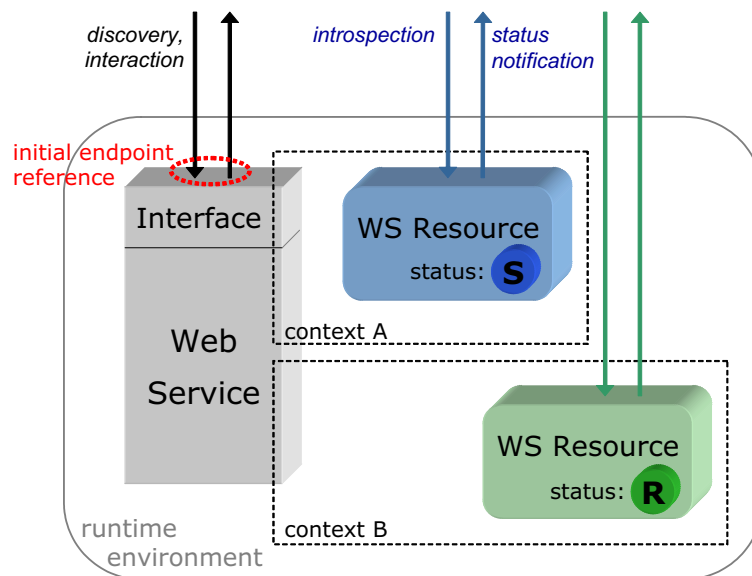


Abbildung 2: Dienste-Aufrufsszenario in WSRF.

mit anderen Mitteln realisiert. Anstatt einen umfassenden Standard vorzulegen wurde die Spezifikation in fünf unabhängige Module aufgeteilt (Tabelle 1), die auch separat nutzbar sind.

Die Problematik ist immer noch die gleiche. Zustandslose Web Services allein reichen für das Grid Computing nicht aus. Zu oft werden Zustandsinformationen auch außerhalb der sie steuernden Web Services benötigt. Hierfür soll WSRF ein von den Diensten unabhängig einsetzbares Konzept liefern. Beim Job-Management beispielsweise durchläuft jeder Job während seiner Bearbeitung eine Reihe von Zuständen, wie *submitted*, *pending*, *running*, *done*. In Ausnahmesituationen können auch andere Zustände wie *canceled* oder *failed* erreicht werden. WSRF definiert ein einheitliches Schema, wie mit solchen Zustandsinformationen in Web Services umzugehen ist.

Abbildung 2 illustriert ein WSRF-Szenario, in dem zwei Benutzer-Jobs über einen Web Service kontrolliert werden. Dabei wird zunächst der Web Service über eine sogenannte *endpoint reference* aufgerufen. Diese besteht initial aus der Adresse des Web Service. Wenn der Web Service erstmalig einen neuen Job zur Bearbeitung erhält, kreiert er einen neuen Kontext, der unter anderem auch den Zustand des Jobs enthält, und weist ihm eine *resource identification* zu, die zusammen mit der Adresse des Web Service eine neue *endpoint reference* bildet, unter der in Zukunft dieser Job angesprochen werden kann.

Bei weiteren Aufrufen bekommt der Web Service die *resource identification* mitgeliefert und weiß daher, auf welchen Daten er operieren soll. Der Web Service selbst ist also zustandslos, kann aber über die *resource identification* auf mehreren verschiedenen Datensätzen arbeiten.

Name	Beschreibung
WS-ResourceProperties	Beschreibt die den Web Services assoziierten Zustandsinformation sowie Abfrage- und Änderungsmechanismen.
WS-ResourceLifetime	Erlaubt es, eine WS-Ressource sofort oder zu einem späteren Zeitpunkt freizugeben.
WS-RenewableReferences	Bietet persistente Referenzen auf Dienste unabhängig von deren konkreter Diensteinstanz.
WS-ServiceGroup	Erzeugt eine Gruppe zusammengehöriger Web Services für die Referenzierung und Nutzung.
WS-BaseFault	Definiert die Basisstruktur von Fehlermeldungen, mit der Fehler unterschiedlicher Dienste einheitlich behandelt werden können.
WS-Addressing	Definiert die Adressierung von Web Services in dynamischen Umgebungen.
WS-Base Notification, WS-Brokered Notification	Stellen ein Benachrichtigungssystem auf der Grundlage des Publish/Subscriber-Modells für Web Services zur Verfügung.

Tabelle 1: Die fünf Spezifikationen der WSRF-Familie [29, 30, 31, 32, 33] und unterstützende Standards [2, 16, 17].

Der Zustand selbst wird über sogenannte *WS-ResourceProperties* [32] modelliert, so daß die Werte auf einheitliche Art und Weise abgefragt, beobachtet und geändert werden können. Im Zusammenspiel mit den Methoden der Web Services Notification [16, 17, 26, 27] können sich andere Dienste automatisch über Zustandsänderungen informieren lassen und dann gegebenenfalls weitere geeignete Aktionen veranlassen.

6 Grid-Dienste im praktischen Einsatz

Effizienz. Dienste können im Grid nur sinnvoll genutzt werden, wenn sie – durch geeignete Definition von Schnittstellen und Protokollen – auf beliebigen Knoten erbracht oder aufgerufen werden können. Diese Plattformunabhängigkeit hat einen hohen Preis. Verglichen mit anderen Möglichkeiten des entfernten Methodenaufrufs, wie Corba, JavaRMI, oder MPI, sind Web- und Grid-Dienste wesentlich aufwendiger und damit langsamer: Messungen mit verschiedenen SOAP-Implementationen waren mehr als zehnmal langsamer [6].

In der Praxis ist allerdings nicht die Aufruf-Latenzzeit das wichtigste Kriterium, sondern vielmehr die verbesserte Interoperabilität. Dienste im Grid sollen keinesfalls die bisherigen effizienten Kommunikationsmechanismen innerhalb verteilter Anwendungen

ablösen [28], sondern auf höherer Ebene allgemeine, applikationsübergreifende Infrastrukturdienste zur Verfügung stellen, die es verteilten Anwendungen erlauben, mehr über ihre heterogene Umgebung zu erfahren und diese einheitlicher, flexibler und besser zu nutzen.

Komplexität der Installation. Nach wie vor ist die Schwelle für den Einsatz von Grid-Systemen relativ hoch. Die Installation und Konfiguration der prototypischen Implementationen kann mit dem Komfort kommerzieller Anwendungen noch nicht konkurrieren. Allerdings besteht mittlerweile in einzelnen Anwendungsfeldern, wie der Teilchenphysik, der Klima- und Umweltforschung oder den Lebenswissenschaften, ein dringender Bedarf nach Unterstützung durch geeignete Grid-Middleware. So ist in den letzten Jahren in internationalen F&E-Projekten Software entwickelt worden, die aber viele Ecken und Kanten, insbesondere in der Installation, Wartung und Nutzungsergonomie aufweist.

Parallel zu den eher akademisch orientierten Grid-Entwicklungen hat sich im kommerziellen Umfeld der Web Services-Standard für B2B-Anwendungen durchgesetzt. Der Erfolg der Web Services war letztlich für die Akteure im Global Grid Forum ausschlaggebend, den als zu komplex erachteten OGSII-Standard durch das stärker an die erfolgreichen Web Services angelehnte WS-Resource Framework zu ersetzen. Es besteht die Hoffnung, dass mit den ersten WSRF-Implementationen auch die Komplexität der Installation reduziert wird.

Zuverlässigkeit und Skalierbarkeit. Mit dem Einsatz von Grid-Systemen im produktiven Betrieb rückten zwei Aspekte in den Vordergrund: Skalierbarkeit und Zuverlässigkeit. Um beides zu erhöhen, wurden zentrale, leistungs- und ausfallkritische Komponenten, wie z.B. Informationsdienste, durch hierarchische Implementationen ersetzt.

Ein weitergehender Ansatz, der bislang noch nicht vollständig realisiert wurde, liegt in der Nutzung von Techniken des *Peer-to-Peer Computing (P2P)* für das Grid Computing [3, 8, 13, 23, 24]. Die flachen, nicht-hierarchischen P2P-Systeme versprechen eine höhere Skalierbarkeit und eine bessere Ausfallsicherheit durch Redundanz. Jeder „Peer“ führt denselben verteilten Algorithmus aus und handelt nach denselben Regeln. Die Gesamtheit der Peers erbringen den Dienst.

Grid-Systeme sind in der Regel sehr umfangreich und bieten eine Vielzahl von Diensten an. Wenn jeder Dienst durch ein einzelnes, speziell dafür ausgelegtes P2P-System erbracht wird, erhält man ein Komponentenmodell, in dem jede (verteilte) Komponente einem P2P-Dienst entspricht [23, 24]. Ähnlich dem erfolgreichen UNIX-Toolset-Ansatz wird dadurch die Komplexität der einzelnen Dienste reduziert, was eine erhöhte Robustheit und bessere Software-Wartbarkeit zur Folge hat.

Überwachung und Selbst-Optimierung. Die Verwendung separater, überschaubarer P2P-Komponenten für jeden zu erbringenden Dienst verbessert zwar die Software-Wartbarkeit, sie ändert aber nichts an der Problematik der Laufzeitüberwachung und -optimierung, die nach wie vor durch Systemadministratoren erfolgen muss. In Grids ist diese Aufgabe besonders schwierig, da Komponenten dynamisch hinzugefügt oder ent-

fernt werden, oder an einzelnen Stellen unvorhersehbare Leistungsanforderungen auftreten können.

Durch den Einsatz von Konzepten des Autonomic Computing [20] wird es möglich, selbst-optimierende Grid-Systeme zu entwerfen. Autonome Verfügbarkeitsdienste überwachen das Vorhandensein einer ausreichenden Anzahl von Dienstekopien, und instanzieren bei steigender Last selbständig neue Kopien [24]. Da in P2P-Systemen keine globale Sicht des Gesamtsystems existiert, entscheidet jede Instanz anhand seiner lokalen Information. Alle Komponenten agieren völlig autonom. Der Informationsaustausch geschieht indirekt durch Beobachtung und Reaktion auf Veränderungen des Systems, wodurch es zwischen den Komponenten zu gegenläufigen Regelprozessen (thrashing) kommen kann, was jedoch durch eine selbständige, dezentrale Schwellwertanpassungen unterdrückt wird.

Analog funktionieren die Optimierungskomponenten für die Lastbalancierung, Dienstplatzierung oder Nutzungsvorhersage. Mehrere derartige Ansätze befinden sich derzeit in der Entwicklung [23]. Um zu den oben genannten Standards kompatibel zu sein, erfolgt die Implementation mit Web Services bzw. nach dem neuen WSRF-Standard.

7 Zusammenfassung

Durch den zunehmenden Trend zur Nutzung verteilter Dienste und Ressourcen wird das Grid Computing weiter an Bedeutung gewinnen. Mehrere Grid-Systeme sind bereits im produktiven Einsatz – nicht nur im akademischen, sondern auch im kommerziellen Bereich.

Parallel zu den ersten Implementationen begannen im Global Grid Forum Bemühungen zur Standardisierung: Zunächst wurde ein (nahezu) reiner Grid-Standard (OGSI) vorgeschlagen, der sich aber als zu umfangreich und zu komplex herausgestellt hat. Anfang dieses Jahres (2004) wurde er durch den *WS Resource Framework* ersetzt, der sich an die im kommerziellen Bereich erfolgreichen Web Services angelehnt. Damit steht nun in Form der WSRF-Standard-Familie (Tab. 1) eine geeignete Erweiterung der Web Services für die Belange der Dienste im Grid zur Verfügung.

Als nächstes steht nun die Konvergenz von Grid- und P2P-Systemen auf der Agenda [7]. Dabei soll die ursprünglich aus den privaten Tauschbörsen stammende P2P-Technologie auf Grid-Systeme übertragen werden [24], um sie skalierbar, selbst-organisierend und vor allen Dingen zuverlässiger zu machen. Hierarchische Grid Systeme werden durch Architekturen mit verteilten Hash-Tabellen, Skip-Lists und Gossip-Algorithmen ersetzt – dies natürlich auf Basis der genannten Standards, um die Kompatibilität und Erweiterbarkeit zu gewährleisten.

Literatur

- [1] T. Banks (Edt.). Open Grid Service Infrastructure Primer (Entwurf), 11.03.2004. <http://www.ggf.org/ogsi-wg/>
- [2] A. Bosworth et al.. Web Services Addressing (WS-Addressing). Joint specification by BEA Systems, IBM and Microsoft, March 2003
- [3] J. Crowcroft, T. Moreton, I. Pratt, A. Twigg. Peer-to-Peer Technologies. In: Foster, Kesselmann (Hrsg.), *GRID2 Blueprint for a New Computing Infrastructure*, pp. 593–622, Morgan Kaufmann Publishers, 2004.
- [4] K. Czajkowski et al. The WS-Resource Framework. Version 1.0, 05.03.2004, <http://www.globus.org/wsrf/>
- [5] K. Czajkowski et al. From Open Grid Services Infrastructure to WS-Resource Framework: Refactoring and Extension. Version 1.1, 05.03.2004, <http://www.globus.org/wsrf/>
- [6] D. Davis, M. Parashar. Latency Performance of SOAP Implementations. In: *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid2002)*, pp. 407–412, Mai 2002.
- [7] EU Expert Group. Next Generation Grid(s) – European Grid Research 2005-2010. Technical Report. Juni 2003. <http://www.cordis.lu/>
- [8] I. Foster, A. Iamnitchi. On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing. 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03), pp. 118–128, February 2003.
- [9] I. Foster, C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*, 2nd Edition, Morgan Kaufmann Publishers 2003.
- [10] I. Foster, C. Kesselman, S. Tuecke. The Anatomy of the Grid - Enabling Scalable Virtual Organizations. *J. Supercomputer Applications*, 2001.
- [11] I. Foster, C. Kesselman, J. M. Nick, S. Tuecke. The Physiology of the Grid - An Open Grid Services Architecture for Distributed Systems Integration, 2002. <http://www.globus.org/research/papers/ogsa.pdf>
- [12] I. Foster, H. Kishimoto. The Open Grid Service Architecture, Version 1.0 Draft, 17.05.2004. <http://www.ggf.org/ogsa-wg/>
- [13] G. Fox et al. Peer-to-Peer Grids. In: Berman, Fox, Hey (Hrsg.), *Grid Computing*, pp. 471–490, Wiley, 2003.
- [14] E. Gamma, R. Helm, R. Johnson, J. Vlissides. *Design Patterns, Elements of Reusable Object-Oriented Software*. Addison Wesley, 1995.
- [15] Global Grid Forum. <http://www.ggf.org/>
- [16] S. Graham et al. Web Services Base Notification (WS-Base Notification) v. 1.0. Joint specification by Bea Systems, IBM and Microsoft, March 2004.
- [17] S. Graham et al. Web Services Brokered Notification (WS-Brokered Notification) v. 1.0. Joint specification by Bea Systems, IBM and Microsoft, March 2004.
- [18] H.-G. Hegering, W. Hiller, R. Maschuw, A. Reinefeld, M. Resch. D-Grid: Auf dem Weg zur e-Science in Deutschland. <http://www.d-grid.de/>

Draft. Final version in: J. von Knop, W. Haferkamp (Hrsg.), 18. DFN Arbeitstagung über Kommunikationsnetze, Düsseldorf, Lecture Notes in Informatics, Series of the German Informatics Society (GI), 2004, vol. P-55, pp. 293 - 304.

- [19] H.F. Hoffmann, A. Putzer, A. Reinefeld. Vom World Wide Web zum World Wide Grid: Eine neue Informations-Infrastruktur für wissenschaftliche Anwendungen. *Physikalische Blätter* 57 (2001) Nr. 12, pp. 39–44.
- [20] IBM Systems Journal. *Autonomic Computing*. Volume 42 Issue 1, 2003.
- [21] D. Kossmann, F. Leymann. Web Services. *Informatik Spektrum*, vol. 26, 2004, pp. 117–128.
- [22] A. Reinefeld, F. Schintke. Grid Services – Web Services zur Nutzung verteilter Ressourcen. *Informatik Spektrum*, vol. 26, 2004, pp. 129–135.
- [23] A. Reinefeld, F. Schintke, T. Schütt. Scalable and Self-Optimizing Data Grids. In: *Annual Review of Scalable Computing*, vol. 6, 2004, Singapore University Press.
- [24] F. Schintke, T. Schütt, and A. Reinefeld, A Framework for Self-Optimizing Grids Using P2P Components. 1st Intl. Workshop on Autonomic Computing Systems (ACS), DEXA'03, pp. 689–693, Sept. 2003.
- [25] S. Tuecke et al. Open Grid Service Infrastructure (OGSI) version 1.0, GGF Proposed Recommendation, Juli 2003. <http://www.ggf.org/ogsi-wg/>
- [26] S. Vinoski Web Services Notifications. *IEEE Internet Computing*. vol. 8, no 2, pp. 86–90, March/April 2004.
- [27] S. Vinoski More Web Services Notifications. *IEEE Internet Computing*. vol. 8, no 3, pp. 90–93, May/June 2004.
- [28] W. Vogels. Web Services Are Not Distributed Objects. *IEEE Internet Computing*. 7(6):59–66. Nov/Dez 2003.
- [29] WS-BaseFaults. Draft March 2004, <http://www.globus.org/wsrf/>
- [30] WS-RenewableReference. Draft March 2004, <http://www.globus.org/wsrf/>
- [31] WS-ResourceLifetime. Draft March 2004, <http://www.globus.org/wsrf/>
- [32] WS-ResourceProperties. Draft March 2004, <http://www.globus.org/wsrf/>
- [33] WS-ServiceGroup. Draft March 2004, <http://www.globus.org/wsrf/>