UNIVERSITÄT LEIPZIG

INSTITUT FÜR INFORMATIK

ABTEILUNG DATENBANKEN

SEMINAR CLOUD DATA MANAGEMENT

Servicebasierte Datenintegration

Seminararbeit

Autor:
Christoph Aßmann
mam08ixo@studserv.uni-leipzig.de

Betreuer:
Michael Hartung
hartung@izbi.uni-leipzig.de

14. März 2010

Inhaltsverzeichnis

1 Einleitung

Sowohl im wissenschaftlichen wie auch im ökonomischen Kontext werden ständig Daten erhoben, deren integrierte Auswertung einen höheren Wert schafft, als die isolierte Auswertung der einzelnen Bestände. Servicebasierte Datenintegration beschreibt diesbezüglich den Prozess der transparenten, technologie- und ortsunabhängigen Auswertung verteilter Datenbestände. Um die Autonomie der Datenquellen weitestgehend zu erhalten und dezentrale Administration zu ermöglichen, werden die Datenbestände über definierte Schnittstellen verfügbar gemacht. Insbesondere beim Grid Computing, einer bereits seit längerem erforschten Form des verteilten Rechnens, wird dieses Konzept verfolgt.

Cloud Computing als ein aktueller Trend, Ressourcen nach Bedarf und Verfügbarkeit zu verteilen und somit die Auslastung zu erhöhen, wird derzeit vermehrt thematisiert und es stellt sich die Frage, ob für diese Form des verteilten Rechnens ebenfalls servicebasierte Datenintegration zum Einsatz kommt und frühere Entwicklungen aus dem Bereich Grid Computing übernommen werden.

Zur Beantwortung dieser Fragestellung wird einleitend ein Vergleich zwischen Grid und Cloud Computing durchgeführt, wobei auf den größeren Bedarf an Standardisierung beim Grid Computing eingegangen wird. Ausgehend von den Standardisierungsbemühungen des Open Grid Forum wird die Evolution von Grid-Systemen anhand einiger Erweiterungen der ursprünglichen dienstbasierten Standards und eines konkreten Anwendungsbeispiels aufgezeigt. Eine wichtige Rolle spielt dabei durchgehend das dienstorientierte Modell verteilter Systeme.

2 Begriffsbestimmung

Im Folgenden werden die zentralen Konzepte im Kontext *servicebasierte Datenintegration* vorgestellt. Der Begriff *Grid Computing* wird vom *Cloud Computing* abgegrenzt, *Datenintegration* erläutert und die Grundlagen *dienstbasierter Architekturen* werden genannt.

2.1 Grid Computing und Cloud Computing

Wesentliche Begriffe der Ausarbeitung sind Grid Computing und Cloud Computing. Im Folgenden wird dargestellt, was diese Formen verteilten Rechnens charakterisiert, welche Besonderheiten sie aufweisen und wodurch sie sich in Hinblick auf die einleitende Fragestellung unterscheiden.

2.1.1 Grid Computing

Laut [FKT01] existieren verschiedene Auslegungen des Konzepts *Grid*. In Abgrenzung zu Peer-to-Peer und verteiltem Rechnen wird Grid Computing als koordinierte Aufgabenbearbeitung und gemeinsame Nutzung von Ressourcen in dynamischen, multiinstitutionellen virtuellen Organisationen beschrieben. Nach dieser Definition werden beim Grid Computing sowohl Daten (Storage) als auch Rechenleistung (Computing) an verschiedenen Standorten über ein Netzwerk bereitgestellt. Die Gesamtheit der Anbieter und Konsumenten zuzüglich der notwendigen Inhaltsbestimmungen und Zugriffsregelungen werden dabei als virtuelle Organisation bezeichnet.

Ein wesentliches Merkmal von Grids ist deren Heterogenität. Verschiedene Standorte sind in der Regel verschiedenen (realen) Organisationen zugehörig und unterliegen demnach unterschiedlicher technischer Administration und strategischer Ausrichtung. Darüber hinaus wird in [Bó8] angemerkt, dass der jeweilige Inhalt der Kollaboration innerhalb der virtuellen Organisation im Voraus geregelt sein muss. Individuelles Anwendungs-Deployment zur unabhängigen Nutzung eines Grids durch einen beliebigen Konsumenten sei nicht vorgesehen. Tatsächlich existieren jedoch Ansätze, die Grid-Systeme um Deployment-Funktionalitäten erweitern ([LPP04, LPP05, TLO+03, GA05]).

2.1.2 Cloud Computing

Cloud Systeme hingegen haben eher eine kommerzielle als eine kollaborative Ausrichtung. Sie können als Dienstleistung eines Anbieters gegenüber Dienstkonsumenten betrachtet werden. Gegenstand des Cloud Computing sind wie bei Grid-Systemen sowohl Storage als auch Rechenleistung. Anwendungen können dabei sowohl durch den Dienstkonsumenten in die Cloud eingebracht werden (Deployment) als auch im Sinne des *Software as a Service* durch den Dienstanbieter bereitgestellt werden (vgl. [LKN+09]).

Die Bereitstellung der Cloud durch einen zentralen Anbieter impliziert Auswirkungen auf die im Kontext der Grid-Systeme erwähnten ökonomischen und technischen Aspek-

te. Cloud-Systeme können homogen aufgebaut werden, Administration und strategische Planung folgen einheitlichen vom Dienstanbieter vorgegebenen Richtlinien. Darüber hinaus ermöglicht die Homogenität der Systeme proprietäre Entwicklungen, die nicht per se abzuwerten sind. Vielmehr können daraus Impulse auf Standardisierungsbemühungen ausgehen und die allgemeine Entwicklung verteilten Rechnens beschleunigt werden.

2.1.3 Gemeinsamkeiten und Unterschiede

Ein Vergleich unter der Fragestellung, ob Cloud Computing lediglich eine Weiterentwicklung von Grid Computing sei, wird in [BÓ8] vorgenommen. Gemeinsam ist den beiden Ansätzen das Ziel, Rechenleistung und Speicher nach Verfügbarkeit und Bedarf zu verteilen. Es sollen somit ungenutzte Kapazitäten vermieden werden, die im Normalfall bei der permanenten dedizierten Zuteilung zu einem Konsumenten entstehen.

Die Unterschiede zwischen Grid und Cloud Computing finden sich in erster Linie bei deren Einsatzzweck und davon abgeleitet bei der Organisation der Netzwerke und den ökonomischen Rahmenbedingungen. So werden Grids hauptsächlich für wissenschaftliche Zwecke eingesetzt, sowohl um Daten zu einem Gegenstand an verschiedenen Orten zu sammeln und dennoch in ihrer Gesamtheit zur Verfügung zu stellen als auch um weniger gut ausgestatteten Institutionen aufwendige Berechnungen (Batch Jobs) zu ermöglichen. Grundlage der virtuellen Organisationen ist demnach die Kollaboration zum gegenseitigen Nutzen. Wissenschaftliche Grids unterliegen öffentlicher Trägerschaft, deren Mittel die Skalierbarkeit der Systeme maßgeblich bestimmt. Skalierbar sind Grids grundsätzlich vertikal durch Hinzufügen neuer Standorte oder horizontal durch Aufrüsten der einzelnen Standorte unter Hinzunahme weiterer Knoten.

Weniger problematisch als bei Grid-Systemen gestaltet sich die Skalierbarkeit der Cloud. Das System wird nicht durch öffentliche Mittel sondern durch die Dienstnutzer finanziert. Sieht das Abrechnungsmodell bei steigender Auslastung der Ressourcen durch den Dienstnutzer steigende “Mieten” vor, lässt sich die Cloud durch die Mehreinnahmen erweitern.

Beiden Formen der Ressourcenteilung gemein ist, dass kaum Sicherheit über die Dienstequalität besteht. Im Falle der Cloud-Systeme sollten entsprechende Angaben Bestandteil des *Service Level Agreement* sein, bei Grid-Systemen besteht hingegen kaum rechtlicher Anspruch auf Verfügbarkeit, Datenschutz und Datensicherheit, Leistungsfähigkeit etc. Insbesondere wird in [BÓ8] auch auf die Problematik der Rechtsverbindlichkeit bei internationalen Dienstanbietern sowie die Intransparenz proprietärer Cloud-Systeme hingewiesen.

2.2 Datenintegration

Eine Übersicht über verschiedene Definitionsansätze des Begriffs *Datenintegration* liefert [Jun06]. Darunter wird die Auffassung des Integrationsbegriffs nach Lenzerini aus dem Englischen übersetzt wiedergegeben:

> "Datenintegration ist das Problem der Kombination von Daten aus unterschiedlichen Quellen und der Bereitstellung einer einheitlichen Sicht (für die Benutzer) auf diese Daten."

Aus dieser Definition lassen sich für den Kontext der Grid-Systeme mehrere Problemfelder der Datenintegration ableiten.

- **Verschiedenheit der Datenquellen**
  Die Datenquellen sind, wie in Abschnitt 2.1.1 beschrieben, autonom, heterogen und geographisch verteilt. Erläuterungen zu diesen Aspekten sind bspw. [HH09b] zu entnehmen. Darüber hinaus bezeichnet [LN07] Informationssysteme als heterogen, die "nicht die exakt gleichen Methoden, Modelle und Strukturen zum Zugriff auf ihre Daten anbieten". Heterogenität wird darüber hinaus als häufige Folge von sowohl Verteilung als auch Autonomie identifiziert. Es werden folgende Arten der Heterogenität unterschieden:
  - Technische Heterogenität: Datenzugriff (Anfragemöglichkeit, Anfragesprache, Austauschformat, Kommunikationsprotokoll)
  - Syntaktische Heterogenität: syntaktisch unterschiedliche Darstellung semantisch gleicher Sachverhalte (Zahlenformat, Zeichenkodierung)
  - Datenmodellheterogenität: syntaktisch unterschiedliche Darstellung semantisch gleicher Modelle (objektorientiert, relational, XML)
  - Strukturelle Heterogenität: unterschiedliche Darstellung semantisch gleicher Ausschnitte der realen Welt in syntaktisch gleichem Datenmodell (Normalisierungsgrad in relationalen Datenbanken)
  - Schematische Heterogenität: unterschiedlicher Detaillierungsgrad semantisch gleicher Ausschnitte der realen Welt in syntaktisch gleichem Datenmodell
  - Semantische Heterogenität: verschiedene Interpretationen eines Namens (Synonyme, Homonyme, Hyperonyme)
- **Kombination der Daten**
  Ein weiterer Aspekt der Datenintegration ist das Data Cleaning. Anforderungen an die aus verschiedenen Datenquellen zusammengeführten Informationen sind Redundanzfreiheit, Korrektheit und Verständlichkeit. Zur Erfüllung dieser Anforderungen sind die Daten vor (offline) oder zur Laufzeit (online) der Anfrage zu bereinigen (vgl. [HH09a, BG09]). Dieser Transformationsschritt kann jedoch entfallen, wenn die einzelnen Bestandteile der virtuellen Organisation vollständig disjunkte Daten halten. Denkbar ist dies bspw. bei der wissenschaftlichen Auswertung von Messreihen zu einem Objekt, die an verschiedenen Orten durchgeführt und gespeichert worden sind.
- **Erstellung einer einheitlichen Sicht**
  Grundsätzlich ist die Architektur des Grid-Systems vor dem Anwender zu verbergen. Um die geforderte Transparenz zu gewährleisten, sind die Beziehungen zwischen den einzelnen Datenquellen zu modellieren. Nach [CT04] sind dazu mehrere

Varianten des Schema-Mappings zu identifizieren, darunter *Global As View* (GAV, Bottom-Up) und *Local As View* (LAV, Top-Down). Bei der Bottom-Up-Integration werden alle verfügbaren lokalen Schemas zu einem globalen Schema gemischt. Eine Anfrage gegen das globale Schema zieht das *Unfolding* der Query nach sich. Bei der Top-Down-Integration wird zunächst das auf den Anwendungszweck zugeschnittene globale Schema erstellt und anschließend das Mapping der lokalen Schemas auf das neue globale Schema vorgenommen. Ein *Rewriting* der Anfrage wird nötig.

2.3 Dienstbasierte Architekturen

[SS07] beschreibt das dienstorientierte Modell als eine Architektur verteilter Systeme, in deren Mittelpunkt "eine prozessorientierte Sicht mit Diensten als Basiskonzept, die in verteilten Systemen angeboten, gesucht und genutzt werden können" steht. Bei Diensten handele es sich demnach um grobgranulare Bausteine, die über wohldefinierte Schnittstellen Funktionalität und Daten kapseln. Ein Ziel dienstorientierter Systeme ist die technologieunabhängige Integration heterogener Systeme. Web Services als eine Anwendung dienstorientierter Systeme werden über ein WSDL[1]-Dokument plattform- und programmiersprachenunabhängig beschrieben.

Ein im Globus Toolkit[2] umgesetzter Ansatz, integrierte Daten technologieunabhängig aus Grid-Systemen heraus zur Verfügung zu stellen, ist der Einsatz von *Grid Services*, die Web Services insbesondere um folgende Eigenschaften erweitern:

- **Lebenszyklusmanagement**
  Grid Services sind im Gegensatz zu Web Services nicht von der Lebenszeit des Web Service Containers abhängig. Sie können individuell instanziiert und beendet werden, was neue Anforderungen wie Bereinigung ungenutzter Dienste mit sich bringt.
- **Service-Zustand**
  Grid Services sind zustandsbehaftet und können die durch einen Dienstkonsumenten erzeugten Daten bis zur nächsten Anfrage vorhalten.
- **Benachrichtigungen**
  Grid Services bieten Dienstkonsumenten die Möglichkeit, sich für Benachrichtigungen am Dienst zu registrieren. Welche Benachrichtigungen ein Dienst bereitstellt, ist dem Dienstanbieter überlassen und nicht standardisiert.
- **Vererbung**
  Grid Services können bereits bestehende Grid Services erweitern.

Um Methoden, Datentypen und o.g. neue Fähigkeiten Dienstkonsumenten darzulegen, ist allgemein eine Grid Service Reference (GSR) notwendig. Im speziellen Fall der Verwen-

[1] http://www.w3.org/TR/wsdl/
[2] http://www.globus.org/toolkit/

dung von SOAP als zugrundeliegendes Netzwerkprotokoll wird die *Web Services Description Language* (WSDL) eingesetzt. Da sich die Erweiterungen jedoch vor der Version 2.0[3] nicht beschreiben ließen, wurde als Übergangslösung die *Grid Services Description Language* (GSDL [vgl. [SSB04]] bzw. Grid-WSDL, GWSDL) eingeführt.

3 Architektur dienstbasierter Grid-Umgebungen

Gemäß der Definition von Grid-Systemen nach [Fos02] existieren eine Reihe von Grid-Implementationen, darunter DAS-3[4], DOE Science Grid[5] oder TeraGrid[6]. Daneben sind weitere nicht-generische *Metacomputing Frameworks* entwickelt worden, die auf bestimmte Anwendungszwecke zugeschnitten sind. In [SSB04] werden diese wie folgt kategorisiert:

- ausgerichtet auf spezielle Forschungsprojekte, z.B. UD Patriot Grid, SETI@Home[7]
- ausgerichtet auf zentralen administrativen Anbieter, z.B. MPICH[8], PVM[9]
- leichtgewichtige Systeme, z.B. H2O[10], JGrid[11]

Um die Interoperabilität unabhängiger, heterogener, geographisch verteilter Grid-Standorte zu gewährleisten, ist die Standardisierung der Systeme unerlässlich. Im Folgenden wird eine offene Architektur dienstbasierter Grid-Systeme beschrieben. Anschließend werden Grid-Implementationen vorgestellt, die auf Grundlage offener Architekturen konzipiert worden sind.

3.1 Standardisierung

Standardisierungsbemühungen im Bereich Grid Computing werden durch das *Open Grid Forum* (OGF) vorgenommen, einer Fusion aus *Global Grid Forum* (akademischer Hintergrund) und der *Enterprise Grid Alliance* (ökonomischer Fokus). Zahlreiche Arbeitsgruppen[12] sind mit der Weiterentwicklung der *Open Grid Services Architecture* (OGSA) in verschiedenen Bereichen beschäftigt. Für den Bereich *Daten* ist u.a. die Arbeitsgruppe *Database Access and Integration Services* (DAIS-WG) zuständig.

[3] http://www.w3.org/TR/wsdl20/
[4] http://www.cs.vu.nl/das3/
[5] http://www.doesciencegrid.org/
[6] http://www.teragrid.org/
[7] http://setiathome.berkeley.edu/
[8] http://www.mcs.anl.gov/research/projects/mpi/mpich1/, http://www.mcs.anl.gov/research/projects/mpich2/
[9] http://www.csm.ornl.gov/pvm/
[10] http://harness2.org/h2o
[11] http://www.ieeetcsc.org/jgrid.html
[12] http://www.ogf.org/gf/group_info/areasgroups.php

Ziel der Bemühungen ist, ein nach standardisierten Schnittstellen zusammengesetztes Grid-System mit standardisierten Funktionalitäten als ein einziges virtuelles System betrachten zu können, dessen Komponenten als Gesamtheit verwaltet, genutzt, überwacht und abgerechnet werden können (vgl. [BDG+06]).

Wesentliches Anliegen der dienstbasierten Datenintegration in Grid-Systemen ist der technologieunabhängige Zugriff auf heterogene Datenquellen. Heterogenität zeichnet sich dabei auf der einen Seite durch verschiedene Datenbankparadigmen aus (relational, objektorientiert, semistrukturiert, dateibasiert) sowie durch herstellerspezifisch unterschiedliche Umsetzungen eines Paradigmas auf der anderen Seite. Die Arbeitsgruppe DAIS gliedert ihre Standardisierungsbemühungen in folgende Kategorien (vgl. [AAK+06]):

- Data Description
- Data Access
- Data Factory

In der Kategorie **Data Description** sind Schnittstellen spezifiziert, anhand derer Metadaten über Datenquellen abgefragt werden können. Diese beinhalten Angaben zu Identifikation, Management, Zugriff, Anfragesprache / -version, Rechte (lesbar / schreibbar) und Transaktionsunterstützung. Fragt ein Dienskonsument die Metadaten eines Dienstes ab, so erhält er diese in Form eines *PropertyDocument*, einem speziellen XML-Dokumenttyp.

Data Access beschäftigt sich mit der Bereitstellung des direkten Zugriffs auf Datenquellen. Dies geschieht über einen *Data Access Service*, der den Zugriff auf vorhandene Datenquellen abstrahiert. Dieser Dienst stellt Schnittstellen zum Auflisten verfügbarer Datenquellen sowie zum Umsetzen eindeutiger abstrakter Ressourcen-Bezeichner in physische Adressen bereit.

Indirekter Zugriff hingegen wird in der Kategorie **Data Factory** behandelt. Dabei wird zunächst durch den Dienstkonsument eine Referenz auf einen passenden *Data Access Service* angefordert, der in der Lage ist, die Anfrage zu beantworten. Dieses Vorgehen erlaubt die transparente Allokation eines Zugriffsdienstes innerhalb des Grid-Systems.

3.2 Konzepte und Implementationen

Eine vollständige Umsetzung der Spezifikation durch alle Hersteller von Datenbanksystemen ist vorerst nicht zu erwarten, weshalb sich die aktuellen Implementierungsbestrebungen auf den Entwurf von Middleware fokussieren. Dieser Ansatz verspricht durch die angestrebte Wiederverwendbarkeit sowohl Kostenreduktion auf Datenbank- und Anwendungsseite als auch Qualitätssteigerung durch häufigeren Einsatz der Middleware.

3.2.1 OGSA-DAI

Wie in [AAB+05] beschrieben handelt es sich bei *Open Grid Services Architecture Data Access and Integration* (OGSA-DAI) um eine Java-basierte Referenzimplementation der durch die DAIS-WG aufgestellten Empfehlungen. Grid Services dienen dabei als sprachneutrale, plattformunabhängige Grundlage für den dienstbasierten Zugriff auf Datenquellen. Die Umsetzung sowie Erweiterungen der Spezifikation durch OGSA-DAI werden im Folgenden anhand ihrer Komponenten beschrieben.

OGSA-DAI-Dienste sind in Containern organisiert. Beim Start eines Containers wird zunächst eine Dienst-Registratur (*Grid Data Service Registry*, GDSR) bereitgestellt. Für jedes Datenbankmanagementsystem im Container wird außerdem eine Instanz einer *Grid Data Service Factory* (GDSF) gestartet, die sich jeweils mit charakteristischen Metadaten an der Registratur des Containers anmeldet.

Abbildung 1 zeigt schematisch einen einfachen möglichen Ablauf einer Anfrage an das Grid-System unter Nutzung der beschriebenen Diensttypen. Stellt ein Client eine Anfrage an das Grid-System, so ist zunächst über die Registratur eine passende Service Factory zu lokalisieren (2). Diese instanziiert anschließend einen *Grid Data Service* (GDS) für die aktuelle Session (3). Ein *Grid Service Handle* (GSH) steht damit eindeutig für eine Client-Datenbank-Verbindung und ermöglicht Transaktionen über mehrere Anfragen hinweg (4). Eine Alternative bzw. eine Möglichkeit, Verbindungskosten zu sparen, ist die Nutzung der dokumentorientierten Schnittstelle eines GDS. Hierbei werden alle auszuführenden Aktivitäten (“activity chain”) einmalig in Form eines *Perform Documents* übertragen. Die in XML beschriebenen Aktivitäten lassen sich gruppieren in das Ausführen von Anfragen gegen die Datenbank (*Database Activities*) und das Weiterreichen von Anfrageergebnissen (*Delivery Activities*) via *Grid Data Transport* (GDT) an weitere GDS oder auch andere Clients als den ursprünglich anfragenden. Ergebnisse einer Aktivität können auf diese Weise als Eingabe für folgende Aktivitäten weitergereicht werden. Darüber hinaus ist es auf einfache Weise möglich, weitere Aktivitäten zu definieren, da jede Aktivität auf eine Java-Klasse abgebildet wird.
Abschließend wird der GDS durch den Client oder nach einem Time-Out beendet.

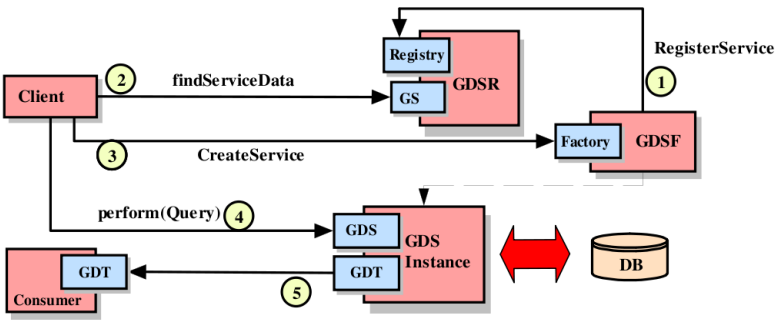

Abbildung 1: Ablauf einer OGSA-DAI-Anfrage (Quelle: [AMP+03])

Auf dieser Basis-Architektur aus Datenzugriffskomponenten setzt *OGSA Distributed Query Processing* (OGSA-DQP) als Datenintegrationskomponente auf. OGSA-DQP ist selbst ein Grid Service mit entsprechenden Schnittstellen und somit dynamisch erzeugbar, Anfragen sind zustandsbehaftet (vgl. [AMP+03]). OGSA-DAI wird demnach um die Dienste *Grid Distributed Query Service* (GDQS, Coordinator) und *Grid Query Evaluation Service* (GQES, Evaluator) erweitert.

Abbildung 2 zeigt die Grid-Anfrage erweitert um Distributed Query Processing. Statt die Anfrage direkt an einen GDS zu senden, werden die GSHs zunächst an den Coordinator weitergeben. Dieser importiert die Schemas der abzufragenden Datenquellen (4), die diese als Metadaten bereitstellen. Anschließend nimmt der Coordinator die Query entgegen (5), kompiliert, optimiert und partitioniert diese auf Grundlage der vorliegenden Schema-Informationen und gibt die entstandenen Teile an die instanziierten Evaluatoren weiter (6). Die Evaluatoren führen die zugewiesene Aktivität aus (7) und reichen das Teilergebnis zur Zusammenführung an den Coordinator zurück (8). Ein Nebeneffekt dieser Konfiguration ist, dass sich durch die Übergabe des Service Handles das Lebenszyklus-Management der Data Services vom Client hin zum Coordinator verschiebt.

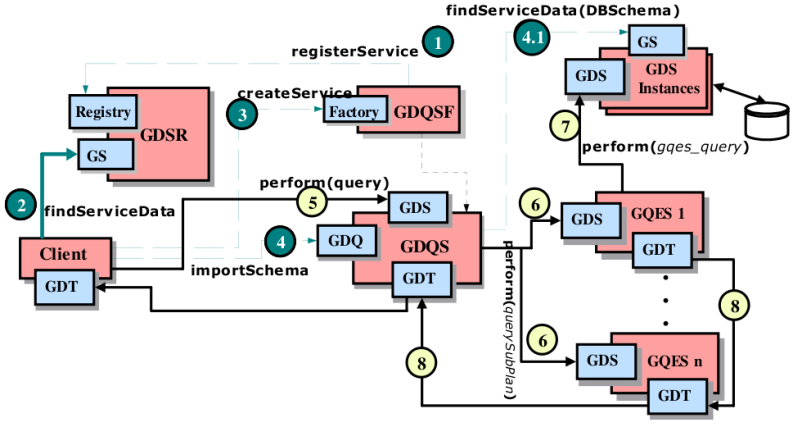

Abbildung 2: OGSA-DAI-Anfrage mit DQP (Quelle: [AMP+03])

3.2.2 GDIS

Das *Grid Data Integration System* (GDIS, [CT04]) nutzt OGSA-DAI und OGSA-DQP (Abschnitt 3.2.1) sowie das Globus Toolkit als Grundlage. Darüber hinaus adressiert das System eine wesentliche Anforderung an verteilte Systeme durch die Erweiterung der genannten Basisbestandteile um einen Dienst zur Schemaintegration: *OGSA Grid Data Integration* (OGSA-GDI). Ohne Schemaintegration ist auch bei verteilter Anfragebearbeitung Voraussetzung, dass die lokalen Schemas der beteiligten Datenquellen dem Dienstnutzer / anfragenden Client bekannt sind und als Metadaten abgefragt werden können.

Bei GDIS wird von der Prämisse ausgegangen, eine vollständige Schemaintegration nach Bottom-Up-Verfahren sei in Grid-Systemen aufgrund ihrer inhärenten Eigenschaften nicht praktikabel (Heterogenität, dezentrale Administration etc.). Daher sei ein Verfahren notwendig, das der starken Dynamik von Grid-Systemen gerecht wird und Schemainformationen neuer oder geänderter Datenquellen iterativ einem globalen (ggf. replizierten) Mapping-Katalog hinzufügt. Demnach wird bei Hinzufügen eines neues Knotens zum Grid-System zunächst wie in Abschnitt 3.2.1 beschrieben über Registratur und Factory ein Service-Handle auf den zugehörigen Data Service erlangt. Über die in GDIS zusätzlich definierte Schnittstelle *Grid Distributed Query* (GDQ) werden anschließend die Schemainformationen der neuen Datenquelle sowie über die Schnittstelle *Import Mappings* (IM) die bisherigen Schemainformationen aus dem globalen Katalog importiert. Ausgehend von lokalem und globalem Schema werden nun Mapping-Definitionen manuell (Schnittstelle MMC, *Manual Mappings Composition*) oder automatisch unter Hinzunahme von Matcher-Bibliotheken (Schnittstelle AMC, *Automatic Mappings Composition*) erzeugt und in den globalen Katalog zurückgespielt (Schnittstelle MU, *Mappings Update*).

Der Ablauf einer Anfrage gegen das erweiterte Grid-System verläuft wie in Abbildung 3 veranschaulicht: Der anfragende Client bezieht zunächst das integrierte Schema von einer *Mediator Node* und formuliert daraus eine Anfrage gegen die *Integration Node* (1). Diese formuliert die Anfrage unter Einbeziehung der durch eine *Mapper Node* (2) bereitgestellten Mappings aus dem globalen Katalog um und reicht sie an den DQP der *Processing Node* weiter (3). Die Anfrage wird partitioniert und an *Execution Nodes* verteilt (4). Jede beteiligte Execution Node greift anhand eines von *Wrapper Nodes* bereitgestellten Wrappers (5) auf Daten der *Data Node* zu und liefert die Ergebnisse zurück an die *Processing Node* (7,8,9). Dort werden die Teilergebnisse zusammengesetzt und das Gesamtergebnis über die *Integration Node* (10) zurück an die *Client Node* (11) gesendet.

3.2.3 In Silico Proteome Integrated Data Environment Resource (ISPIDER)

Eine konkrete Anwendung der erläuterten Architekturen wird in [ZFB+06] beschrieben. Das *ISPIDER Proteomics Grid* soll dazu dienen, Analysen über verschiedene Datenquellen aus dem Bereich Proteomik (Biochemie) durchzuführen. Dies erweitere die Datenbestände beträchtlich und helfe darüber hinaus dabei, die Datenqualität zu erhöhen, indem abweichende Werte im Vergleich als solche erkannt werden. In die Analyse einbezogen werden sollen die verteilten, autonomen Datenquellen *Global Proteome Machine Database* (gpmDB), *Proteome Experimental Data Repository* (PEDRo) und PepSeeker. Neben den Basistechnologien OGSA-DAI und OGSA-DQP soll außerdem AutoMed [BMT02] zur Schematransformation eingesetzt werden. Der Übergang (*Pathway*) von S_1 nach S_2 führt dabei über Zwischenschemas, an denen jeweils primitive Operationen `add`, `delete`, `extend`, `contract` oder `rename` auf einzelne Schemaelemente durchgeführt werden. Die Übergänge werden dabei jeweils in IQL ([PZ08]) ausgedrückt.

Bei der Konzeption des Grid Systems wurde entschieden, für das globale Schema eine Untermenge der Vereinigung aller beteiligten Schemas zu nutzen. Auf Grundlage der nach der Integration enthaltenen Informationen sollen übliche Analysen und vor allem Vergleiche möglich sein. Attribute, die nicht in allen Datenquellen vorhanden sind, ließen sich hinge-

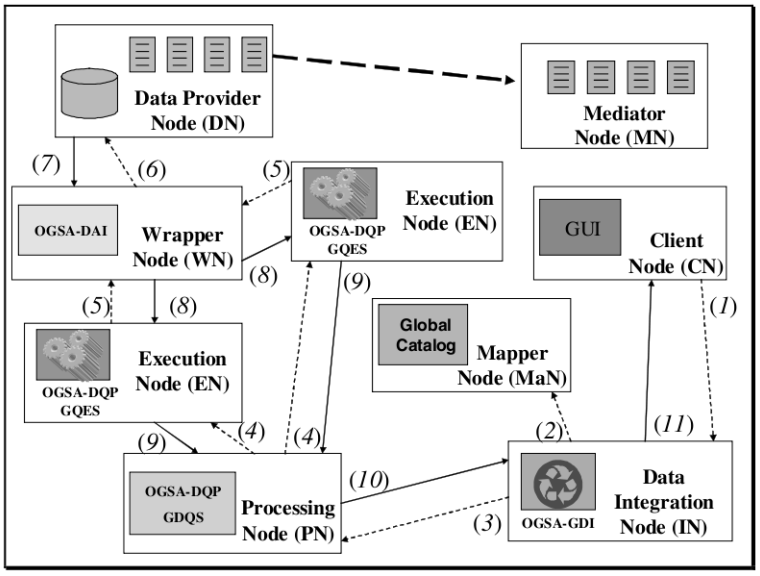

Abbildung 3: OGSA-GDI-Anfrage (Quelle: [CT04])

gen nicht vergleichen. Aufgrund seines hohen Detailgrads wurde das PEDRo-Schema als Basis für das globale Schema gewählt und davon ausgehend die Beziehungen zu den Schemas von gpmDB und PepSeeker hergestellt. Die Problematik der eindeutigen Identifikation von Objekten in der integrierten virtuellen Datenbank wurde durch hinzugefügte *Life Science Identifiers* (LSIDs) gelöst. Konflikte zwischen den numerischen Primärschlüsseln der Ursprungsdatenbanken wurden somit vermieden.

Aufgrund der übersichtlichen Anzahl beteiligter Schemas und teilweise kryptischer Bezeichner in den lokalen Schemas ist das initiale globale Schema manuell erstellt worden. Durch die Nutzung von AutoMed lassen sich dennoch lokale wie globales Schema verändern (Schemaevolution), es müssen lediglich die AutoMed-Transformationspfade erweitert werden.

Der Ablauf einer Anfrage gegen das Grid-System verläuft wie in Abbildung 4 dargestellt: Der Client formuliert eine IQL-Query gegen das globale Schema und sendet diese an den *AutoMed Query Processor* (AQP). Anhand des *Schemas & Transformations Repository* ist AutoMed in der Lage, diese Query entlang der Transformationspfade in Anfragen gegen die lokalen Schemas zu übersetzen. Die umformulierten Anfragen werden anschließend optimiert und, nach Übersetzung von IQL nach OQL durch den *AutoMed-DQP Wrapper*, an den OGSA-DQP weitergereicht. Nach Ablauf des regulären Ergebnisbeschaffungsprozesses wird die in Form eines XML-Antwort-Dokuments eintreffende Antwort durch den Wrapper in das IQL-Typsystem übersetzt und weiter evaluiert.

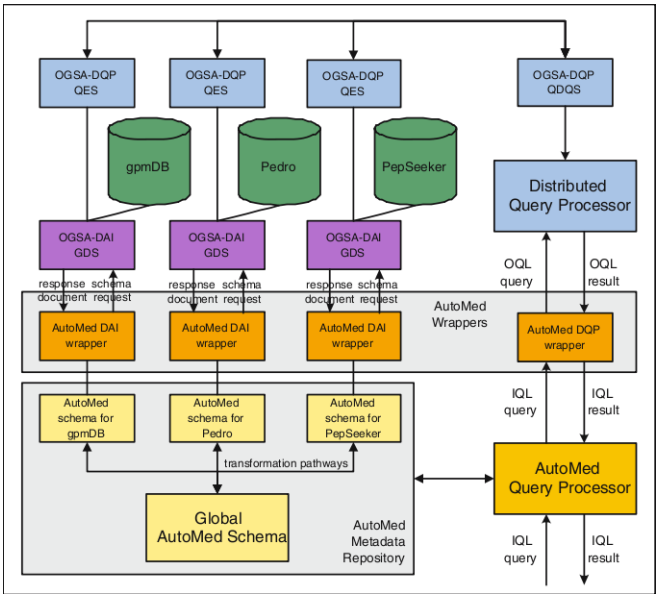

Abbildung 4: ISPIDER-Architektur (Quelle: [ZFB+06])

4 Dienstbasierte Datenintegration in der Cloud

Wie in Kapitel 2.1.2 beschrieben, unterscheiden sich Grids von Clouds insbesondere durch ihren Einsatzzweck und den Anbieter der Ressourcen. Während in Grid-Systemen die Anforderung besteht, an verschiedenen Standorten erzeugte Daten in einer einheitlichen Sicht bereitzustellen, ist die Integrationsproblematik in Clouds eine andere. Entweder Daten werden direkt in der Cloud erzeugt – bspw. durch SaaS-Anwendungen – und liegen damit bereits in einheitlicher Form vor oder Daten müssen zur weiteren Nutzung in die Cloud eingebracht werden. Integration in die Cloud bezeichnet folglich den Vorgang, Daten aus veschiedenen Legacy-Systemen des Dienstnehmers in ein einheitliches Schema auf Seiten des Cloud-Anbieters zu überführen. Um den Nutzer der Cloud-Dienste von der technischen Umsetzung dieses Problems weitestgehend zu entlasten, bieten Drittanbieter die Migration der Daten als Dienstleistung an.

Im Rahmen ihrer *Cloud Data Integration Solutions* [inf09] identifiziert die Informatica Corporation die notwendige Integration der Daten in die Cloud als größte Hürde für den Umstieg von Legacy-Systemen auf Cloud-basierte Anwendungen. Die Verlagerung einer Anwendung in die Cloud zieht in Bezug auf die Datenhaltung drei Anforderungen nach sich:

- Initiales Laden der Daten in die Cloud
- Regelmäßige Synchronisation der Daten
- Extrahieren der Daten für Datensicherung und Berichterstellung

Um diesen Anforderungen begegnen zu können, wurde eine Lösung geschaffen, lokale Datenquellen eines Unternehmens in die SaaS-Anwendung *Salesforce CRM*, die in der Cloud-Plattform *Force.com* läuft, zu migrieren. Dem Endanwender wird dabei die aufwendige Entwicklung einer Migrationslösung abgenommen, alle notwendigen Angaben zur Datenintegration werden über ein Web-Interface konfiguriert. Diese Angaben beinhalten:

- Verbindungsinformationen zur Legacy-Datenquelle
- Verbindungsinformationen zur Zieldatenbank in der Cloud
- Zuordnung der Felder aus Quell- und Zieldatenbank
- Filterdefinitionen (Einschränkung der zu übertragenden Daten)
- Transformationsregeln
- Ablaufplanung (Scheduling)

Festzustellen ist, dass es sich bei den Informatica Cloud Data Integration Solutions um eine Lösung handelt, die explizit auf einen Cloud-Anbieter zugeschnitten ist und sich um diesen als Dienstleistung gegenüber dessen Kunden ansiedelt.

5 Zusammenfassung

Der Vergleich der Konzepte Grid Computing und Cloud Computing zeigt, dass der wesentliche Unterschied in deren Einsatzzweck liegt. Beiden liegt das Bestreben nach besserer Verteilung von Ressourcen zugrunde. Grid Computing wird jedoch eher im wissenschaftlichen Kontext über unabhängige, geographisch verteilte, heterogene Datenquellen angewendet, während Cloud Computing bei gleichem Hintergrund die Bereitstellung der Cloud durch einen zentralen Anbieter vorsieht. Die Heterogenität der Datenquellen und die dezentrale Administration dieser stellt beim Grid Computing folglich weitaus höhere Anforderungen an die Standardisierung von Kommunikationsschnittstellen. Eine tragende Rolle bei der Spezifikation von dienstbasierten Grid-Systemen spielt das Open Grid Forum, welches die Open Grid Services Architecture (OGSA) in verschiedenen Bereichen entwickelt. Unter den derzeit acht Bereichen[13] ist unter anderem die Arbeitsgruppe *Database Access and Integration Services* für den Bereich Daten zuständig. Ergebnis dieser Arbeitsgruppe ist die Spezifikation WS-DAI, auf der weitere Dienste aufsetzen, die bestimmte Probleme des Grid Computing wie verteilte Datenbankanfragen oder Schemaintegration adressieren. Architekturen wie ISPIDER im Bereich Biochemie nutzen diese Standards und nach diesen entwickelte Middleware, um Datenquellen technologieunabhängig zu integrieren und in der Gesamtheit einen höheren Wert zu schaffen als durch die isolierte Auswertung der einzelnen Datenbestände.

Cloud-Systeme dagegen weisen eine davon abweichende Integrationsproblematik auf. Statt virtueller Integration heterogener Datenquellen steht hier die physische Integration in ein eher homogenes Cloud-System im Vordergrund. Servicebasierte Datenintegration im Cloud-Umfeld ist nicht gekennzeichnet durch standardisierte Web Services sondern durch die Dienstleistung eines (Dritt-) Anbieters gegenüber Nutzern des Cloud-Systems.

Angesichts der grundsätzlich verschiedenen Bedeutung servicebasierter Datenintegration in den Bereichen Grid Computing und Cloud Computing erscheint eine gegenseitige Beeinflussung bzw. Übernahme von Entwicklungen nicht sinnvoll. Einzelne Grid-Standorte ließen sich zwar als Cloud-System organisieren, deren Nutzer ist dann aber zugleich der Anbieter – die wissenschaftliche Einrichtung.
In Cloud-Systemen ist der Einsatz von Web Services als abstrahierende Schicht zum Verbergen der Heterogenität nicht notwendig, da Cloud-Systeme zentral administriert und homogenen aufgebaut sind.

[13] Applications, Architecture, Compute, Data, Infrastructure, Liaison, Management, Security

Literatur

[AAB+05] Mario Antonioletti, Malcolm P. Atkinson, Robert M. Baxter, Andrew Borley, Neil P. Chue Hong, Brian Collins, Neil Hardman, Alastair C. Hume, Alan Knox, Mike Jackson 0003, Amrey Krause, Simon Laws, James Magowan, Norman W. Paton, Dave Pearson, Tom Sugden, Paul Watson und Martin Westhead. The design and implementation of Grid database services in OGSA-DAI. *Concurrency - Practice and Experience*, 17(2-4):357–376, 2005.

[AAK+06] Mario Antonioletti, Malcolm Atkinson, Amy Krause, Simon Laws, Susan Malaika, Norman W Paton, Dave Pearson und Greg Riccardi. Web Services Data Access and Integration – The Core (WS-DAI) Specification, Version 1.0, July 2006.

[AMP+03] M. Nedim Alpdemir, Arijit Mukherjee, Norman W. Paton, Paul Watson, Alvaro A. A. Fernandes, Anastasios Gounaris und Jim Smith. Service-Based Distributed Querying on the Grid. In Maria E. Orlowska, Sanjiva Weerawarana, Mike P. Papazoglou und Jian Yang, Hrsg., *ICSOC*, Jgg. 2910 of *Lecture Notes in Computer Science*, Seiten 467–482. Springer, 2003.

[BÓ8] Marc-Elian Bégin. An EGEE comparative study: Grids and clouds - evolution or revolution. Bericht, CERN - Engeneering and Equipment Data Management Service, June 2008.

[BDG+06] D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, F. Siebenlist, R. Subramaniam, J. Treadwell und J. Von Reich. The Open Grid Services Architecture, Version 1.5, July 2006.

[BG09] Andreas [Hrsg.] Bauer und Holger [Hrsg.] Günzel. *Data-Warehouse-Systeme*. dpunkt, Heidelberg, 2009.

[BMT02] Michael Boyd, Peter McBrien und Nerissa Tong. The AutoMed Schema Integration Repository. In Barry Eaglestone, Siobhán North und Alexandra Poulovassilis, Hrsg., *BNCOD*, Jgg. 2405 of *Lecture Notes in Computer Science*, Seiten 42–45. Springer, 2002.

[CT04] Carmela Comito und Domenico Talia. GDIS: A Service-Based Architecture for Data Integration on Grids. In Robert Meersman, Zahir Tari und Angelo Corsaro, Hrsg., *OTM Workshops*, Jgg. 3292 of *Lecture Notes in Computer Science*, Seiten 88–98. Springer, 2004.

[FKT01] Ian T. Foster, Carl Kesselman und Steven Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *IJHPCA*, 15(3):200–222, 2001.

[Fos02] Ian Foster. What is the Grid? A Three Point Checklist. Bericht, Argonne National Laboratory & University of Chicago, 2002.

[GA05] Wojtek Goscinski und David Abramson. Application Deployment over Heterogeneous Grids using Distributed Ant. In *e-Science*, Seiten 361–368. IEEE Computer Society, 2005.

[HH09a] Steven Helmis und Robert Hollmann. *Data Cleaning*, Kapitel 4. In Webbasierte Datenintegration [HH09c], 1. Auflage, 2009.

[HH09b] Steven Helmis und Robert Hollmann. *Dimensionen und Architektur der Informationsintegration*, Kapitel 3. In Webbasierte Datenintegration [HH09c], 1. Auflage, 2009.

[HH09c] Steven Helmis und Robert Hollmann. *Webbasierte Datenintegration*. Vieweg + Teubner — GWV Fachverlage GmbH, Wiesbaden, 1. Auflage, 2009.

[inf09] Informatica Cloud Data Integration Solutions for Salesforce CRM and force.com. White Paper, November 2009.

[Jun06] Reinhard Jung. *Architekturen zur Datenintegration*, Kapitel 2. Deutscher Universitats-Verlag — GWV Fachverlage GmbH, Wiesbaden, 2006.

[LKN+09] Alexander Lenk, Markus Klems, Jens Nimis, Stefan Tai und Thomas Sandholm. What's inside the Cloud? An architectural map of the Cloud landscape. *Software Engineering Challenges of Cloud Computing, ICSE Workshop on*, 0:23–31, 2009.

[LN07] Ulf Leser und Felix Naumann. *Informationsintegration: Architekturen und Methoden zur Integration verteilter und heterogener Datenquellen.* dpunkt, 2007.

[LPP04] Sébastien Lacour, Christian Pérez und Thierry Priol. A Network Topology Description Model for Grid Application Deployment. *Grid Computing, IEEE/ACM International Workshop on*, 0:61–68, 2004.

[LPP05] Sébastien Lacour, Christian Pérez und Thierry Priol. Generic Application Description Model: Toward Automatic Deployment of Applications on Computational Grids. *Grid Computing, IEEE/ACM International Workshop on*, 0:284–287, 2005.

[PZ08] Alex Poulovassilis und Lucas Zamboulis. A Tutorial on the IQL Query Language. Bericht, Department of Computer Science and Information Systems, Birkbeck, University of London, 2008.

[SS07] Alexander Schill und Thomas Springer. *Verteilte Systeme - Grundlagen und Basistechnologien*, Kapitel 2. Springer-Verlag, Berlin / Heidelberg, 2007.

[SSB04] Gunther Stuer, Vaidy S. Sunderam und Jan Broeckhove. Towards OGSA Compatibility in Alternative Metacomputing Frameworks. In Marian Bubak, G. Dick van Albada, Peter M. A. Sloot und Jack Dongarra, Hrsg., *International Conference on Computational Science*, Jgg. 3036 of *Lecture Notes in Computer Science*, Seiten 51–58. Springer, 2004.

[TLO+03] H. Q. Thuan, D. Lim, Y. S. Ong, Students Ho, Quoc Thuan und Dudy Lim. Grid Application Deployment Kit, 2003.

[ZFB+06] Lucas Zamboulis, Hao Fan, Khalid Belhajjame, Jennifer A. Siepen, Andrew Jones, Nigel J. Martin, Alexandra Poulovassilis, Simon J. Hubbard, Suzanne M. Embury und Norman W. Paton. Data Access and Integration in the ISPIDER Proteomics Grid. In Ulf Leser, Felix Naumann und Barbara A. Eckman, Hrsg., *DILS*, Jgg. 4075 of *Lecture Notes in Computer Science*, Seiten 3–18. Springer, 2006.