M. Husemann, D. Rathig und N. Ritter

Transaktionskontrolle im Grid-Data-Computing

ZUSAMMENFASSUNG

Grid-Data-Services sind auf Web-Service-Technologien beruhende Dienste, die dynamisch in komplexe, datenintensive Verarbeitungsvorgänge einbezogen werden können (Grid-Data-Computing), die wiederum eines Transaktionsschutzes bedürfen. Aufgrund der Nähe zu Web-Services kann eine Transaktionskontrolle auf Basis der Spezifikationen WS-Coordination und WS-Transaction vorgenommen werden. Diese erfüllen jedoch die Anforderungen des Grid-Data-Computing nur unzureichend, da häufig auftretende Ablaufmuster nicht angemessen unterstützt werden. Beispielsweise ist es nicht möglich, eine Transaktion durch eine Anfrage an einen Grid-Data-Service zu initiieren, die Ergebnisse zur Verarbeitung an einen dritten Teilnehmer weiterzuleiten und diesen die Transaktion beenden zu lassen. Ebenso wird der dynamischen Fortpflanzung von Verarbeitungsvorgängen über vielfältige Grid-Data-Services hinweg nicht konsequent Rechnung getragen. Wir schlagen in diesem Beitrag ein auf WS-Coordination und WS-Transaction aufbauendes Ablaufmodell für das Grid-Data-Computing vor, das die beschriebenen Defizite durch die Einführung spezieller Dienste und Protokolle ausräumt.

1 EINLEITUNG

Das Paradigma des *Grid-Data-Computing* umfasst Datenzugriffe und das Datenmanagement in Grid-Umgebungen. Zentrale Bestandteile von Grid-Umgebungen sind die *Grid-Services*, erweiterte Web-Services mit einheitlichen Schnittstellen etwa für das Erzeugen, Benennen, Entdecken, Aufrufen und Beenden von Diensten. Die *Open-Grid-Services-Architecture* (*OGSA*, [FKT01], [FKN+02]) beschreibt die Semantik der Schnittstellen von Grid-Services; die *Open-Grid-Services-Infrastructure* (*OGSI*, [TCF+03]) definiert die Schnittstellen von Grid-Services auf der Basis der *Web-Services-Description-Language* (*WSDL*) und beschreibt Konventionen für den Umgang von Klienten mit Grid-Services.

Grid-Data-Services (*GDS*) sind Grid-Services, die durch eine einheitliche Schnittstelle zu unterschiedlichen Typen von Datenquellen eine *Daten-Virtualisation*, das heißt eine Abstraktion von konkreten Datenquellen, anbieten [FTU+03]. In der Spezifikation der Grid-Data-Services [AAM+03] wird eine Ablaufkontrolle auf der Grundlage von Web-Services-Coordination, Web-Services-Atomic-Transaction und Web-Services-Business-Activity nahe gelegt. In einem alternativen Ansatz wird eine auf der OGSA-Plattform und dem X/Open-DTP-Modell basierende Grid-Transaction-Processing-Architektur vorgestellt [QYT04]. Durch diese Architektur können Zugriffe von Grid-Anwendungen auf Datenbanksysteme über verteilte Transaktionen koordiniert werden. Dabei wird ausgenutzt, dass die populären RDBMS (beispielsweise IBM DB2, ORACLE, Sybase) die XA-Schnittstelle des X/Open-DTP-Modells unterstützen und dadurch als Resource-Manager im Sinne von X/Open-DTP auftreten können.

Die genannten Modelle für die Ablaufkontrolle sind nicht im Hinblick auf Grid-Data-Services entwickelt worden; ihre Flexibilität bezüglich der Abläufe verteilter Transaktionen in Grid-Umgebungen ist begrenzt. In diesem Beitrag stellen wir daher einen erweiterten Ansatz für die Transaktionskontrolle im Grid-Data-Computing vor, der Konzepte bewährter Modelle wie X/Open-DTP und CORBA Object-Transaction-Service aufgreift und auf der Basis von WS-Coordination an die Bedürfnisse des Grid-Data-Computing anpasst.

In Abschnitt 2 betrachten wir typische Interaktionsarten und mögliche Ablaufstrukturen des Grid-Data-Computing. Wir identifizieren daraus resultierende Anforderungen an ein Modell für die Transaktionskontrolle in Abschnitt 3 und untersuchen in Abschnitt 4 existierende Modelle auf ihre Eignung für die Anwendung in Grid-Umgebungen. In Abschnitt 5 stellen wir unseren erweiterten Ansatz vor und erläutern die entscheidenden Neuerungen anhand von Beispielen. Abschnitt 6 gibt eine Zusammenfassung über die gewonnenen Erkenntnisse.

2 INTERAKTIONEN UND ABLAUFSTRUKTUREN IN GRID-UMGEBUNGEN

Es gibt drei grundlegende Arten der Interaktion zwischen Klienten und Grid-Data-Services: einfache Anfragen, Aufrufe mit Auslieferung der Anfrage-Ergebnisse an eine dritte Partei und automatische Ausführung von Anfragen in regelmäßigen Zeitabständen oder abhängig von bestimmten Ereignissen mit Auslieferung der Anfrage-Ergebnisse an vorher bestimmte Knoten. Letztere Interaktionsart wird im Folgenden *Abonnement* genannt. Abb. 1 A - C veranschaulichen diese Interaktionsarten:

A. einfache Anfrage: Client 1 stellt eine Anfrage an den GDS [1a] und erhält die Ergebnisse seiner Anfrage [1b].
B. Anfrage-Ergebnisse für dritte Partei: Client 1 stellt eine Anfrage an den GDS [1a] und erhält eine Bestätigung, dass die Anfrage ausgeführt wurde [1b]. Die Ergebnisse der Anfrage werden an Client 2 geschickt [2].
C. Abonnement: Client 1 registriert sich bei dem GDS für die regelmäßige Ausführung einer bestimmten Anfrage, beispielsweise täglich zu einer bestimmten Uhrzeit [1a]. Client 1 erhält vom GDS regelmäßig die Ergebnisse dieser Anfrage [1b].

Beispiel-Szenarien für diese Interaktionsarten werden in [PAD+02], [ADG+03] und [GGM+03] vorgestellt.

In Grid-Umgebungen können sich Ablaufstrukturen während einer Anfrage, also auch innerhalb einer Transaktion, verändern. Abhängig von den Rahmenbedingungen der Anfrage kann es also zu statischen oder dynamischen Ablaufstrukturen kommen.

Statische Ablaufstrukturen liegen vor, wenn der die Transaktion startende Knoten die Struktur festlegt, nach der die Transaktion ablaufen soll. Er bestimmt, welche Knoten an der Transaktion teilnehmen und wie sie miteinander interagieren. Diese Ablauf-

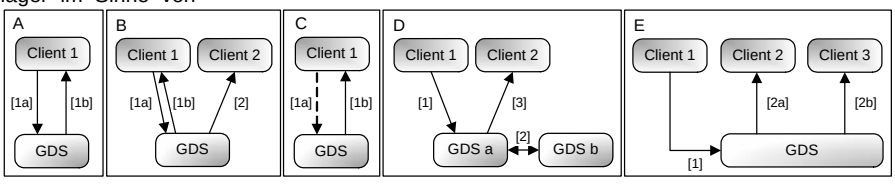

Abb. 1: Arten der Interaktion zwischen Klienten und Grid-Data-Services und Beispiele für statische und dynamische Ablaufstrukturen.

struktur ist während der Transaktion nicht veränderbar und erfordert einige Kenntnisse des Initiators über den Ablauf der Transaktion. Die bereits beschriebenen Ablaufstrukturen aus Abb. 1 A - C sind statisch.

Bei *dynamischen Ablaufstrukturen* kann nach der die Transaktion initiierenden Anfrage des Klienten die Ablaufstruktur dynamisch an die Erfordernisse der Teilnehmer angepasst werden. Ein Klient muss dynamische Ablaufstrukturen zulassen, wenn er nicht sicher sein kann, dass die beteiligten Knoten zur Erfüllung ihrer Aufgaben keine anderen Knoten in die Transaktion einbeziehen müssen. Jeder zu beteiligende Knoten kann selbst entscheiden, welche anderen Knoten er in die Transaktion einbeziehen möchte und welche Ablaufstruktur sich daraus letztendlich ergibt. Abb. 1 D und E zeigen zwei mögliche dynamische Ablaufstrukturen:

D. GDS a benötigt zur Bearbeitung der Anfrage von Client 1 den GDS b [2] und liefert die Ergebnisse der Anfrage an Client 2 [3].
E. Von dem Ergebnis der Anfrage, die Client 1 an den GDS stellt [1], hängt es ab, ob Client 2 oder Client 3 die Ergebnisdaten der Anfrage erhält [2a und 2b].

3 ANFORDERUNGEN

Das Grid-Data-Computing soll maximale Dynamik und Flexibilität bei der verteilten Verarbeitung datenintensiver Vorgänge in Grid-Umgebungen ermöglichen. Folglich gilt es, statische Abhängigkeiten und vordefinierte Beziehungen zwischen den Knoten zu vermeiden. Idealerweise werden Anfragen ohne vorherige Festlegung von Ablaufstrukturen im Grid durchgeführt. Diese sollen sich dynamisch bei der Verarbeitung einer Anfrage ergeben, etwa durch die Propagierung von Aufrufen über Knoten im Grid hinweg: Jeder Knoten kann für die Beantwortung einer Anfrage Dienste anderer Knoten in Anspruch nehmen, so dass es zu einer für den Anfrager transparenten Fortpflanzung der Anfrage kommt (vgl. Distributed-Query-Service [PAD+02] und Virtual-Database-Systems [Wat01] zur Weiterleitung von Anfragen an Grid-Data-Services). Trotz der dynamischen Organisation der Anfrageverarbeitung sind dabei die Anforderungen an transaktionale Datenmanipulation zu erfüllen.

Die sich dynamisch entwickelnden Ablaufstrukturen basieren auf den bereits vorgestellten Interaktionsarten. Speziell die Kombinationen der Interaktionsarten „Aufrufe mit Auslieferung der Anfrage-Ergebnisse an eine dritte Partei“ und „Abonnement“ ermöglichen die Erzeugung komplexer Abläufe von Aufrufen zur Verarbeitung von Anfrageergebnissen. Innerhalb dieser komplexen und dynamischen Ablaufstrukturen lassen sich Knoten mit verschiedenen funktionalen Rollen unterscheiden, etwa die Rolle des *demarkationsberechtigten Knotens* sowie die des *Koordinators* bei verteilten Transaktionen.

Demarkationsberechtigte Knoten verfügen über das Recht, die laufende Transaktion abzuschließen. Dieses Recht liegt üblicherweise ausschließlich bei dem transaktionsinitiierenden Knoten. Die in Grid-Umgebungen mögliche dynamische Anfrage-Fortpflanzung sowie die Interaktionsarten „Aufrufe mit Auslieferung der Anfrage-Ergebnisse an eine dritte Partei“ und „Abonnement“ führen zwangsläufig zu Situationen, in denen Knoten Transaktionen zwar anstoßen, die Entscheidung über den Abschluss jedoch anderen Knoten überlassen sollten, die stärker in die komplexen Verarbeitungsabläufe involviert sind. Erst durch die Möglichkeit des Abtretens oder Teilens des Rechts zur Demarkation einer Transaktion kann die Flexibilität, die durch dynamische Anfrage-Fortpflanzung geboten wird, umfassend ausgeschöpft werden.

Der Koordinator übernimmt die Abwicklung der Transaktion, nachdem ein entsprechender Aufruf durch einen demarkationsberechtigten Knoten erfolgt ist. Üblicherweise ist die Übergabe der Koordinator-Rolle an einen anderen Knoten innerhalb einer Transaktion nicht möglich. Verschiedene Situationen können die Übergabe der Koordination an einen anderen Knoten erforderlich machen: etwa eine Überlastung des Koordinator-Knotens mit Einbußen in seiner Performanz oder der Beitritt eines Knotens in eine laufende Transaktion, der selbst die Koordinator-Rolle übernehmen kann, so dass die Kosten für die Benutzung des Koordinations-Services verringert werden. Nur durch die Übergabe der Koordinator-Rolle kann auch zur Laufzeit einer Anfrage flexibel auf sich verändernde Umgebungsbedingungen reagiert werden.

An ein umfassendes Modell für die Transaktionskontrolle im Grid-Data-Computing sind daher folgende Anforderungen zu stellen:

1. Gewährleistung transaktionaler Datenmanipulation, insbesondere der Atomarität verteilter Bearbeitungsschritte, über einer Menge von Knoten, die entweder
   a. statisch ist und mit Beginn der Transaktion feststeht oder
   b. in der laufenden Transaktion durch die Fortpflanzung von Aufrufen bzw. Anfragen dynamisch erweitert werden kann;
2. Möglichkeit zur Einnahme unterschiedlicher Rollen durch an einer Aktivität beteiligte Knoten:
   a. zwischen aufeinander folgenden Aktivitäten oder
   b. innerhalb einer Aktivität.

Anforderung 2a beschreibt dabei die bloße Fähigkeit eines Knotens, verschiedene Rollen einnehmen zu können. Sie ist damit Voraussetzung für Anforderung 2b.

4 EXISTIERENDE ANSÄTZE

Unter besonderer Berücksichtigung dieser Anforderungen werden im Folgenden aktuelle Ansätze der Ablaufkontrolle für das Grid-Data-Computing betrachtet.

4.1 X/OPEN DTP

Das *Distributed-Transaction-Processing-Modell* (X/Open DTP) der Open Group liegt seit 1996 in Version 3 vor [XOp96]. Es beschreibt eine Software-Architektur für die Koordination verteilter Zugriffe auf gemeinsam genutzte Ressourcen. Hierfür werden vier funktionale Komponenten definiert. Eine *Applikation* legt die Demarkationen von Transaktionen fest und initiiert die Datenoperationen, die innerhalb der Transaktionen ausgeführt werden sollen. *Resource-Manager* wie Datenbanken oder Dateisysteme führen die eigentlichen Zugriffe auf Datenquellen durch. Ein *Transaktions-Manager* weist Transaktionen eindeutige Kennungen zu, überwacht ihren Fortschritt und koordiniert den erfolgreichen Abschluss bzw. im Fehlerfall die Rücksetzung der Transaktionen. *Kommunikations-Manager* kontrollieren die Kommunikation zwischen verteilten Applikationen, die mehrere Transaktions-Manager einbeziehen, und bedienen sich dabei geeigneter Kommunikationsprotokolle.

Die Interaktion der funktionalen Komponenten erfolgt über spezifizierte Schnittstellen, die die Signatur und Semantik der benötigten Methoden festlegen. Von zentraler Bedeutung für die übergeordnete Ablaufkontrolle sind das *TX-Interface* zwischen Applikation und Transaktions-Manager sowie das *XA-Interface* zwischen Transaktions-Manager und Resource-Manager.

Eine Transaktion wird gestartet, indem die Applikation die Methode *tx_begin()* des zugehörigen Transaktions-Managers aufruft. Der Transaktions-Manager weist daraufhin die teilnehmenden Resource-Manager an, Anfragen der Applikation innerhalb der Transaktion durchzuführen. Alternativ können Resource-Manager auch erst im Moment der ersten Anfrage der Applikation der Transaktion beitreten (*dynamische Registrierung*). Die Applikation schließt die Transaktion ab, indem sie die Methode *tx_commit()* des Transaktions-Managers aufruft. Der Transaktions-Manager wickelt dann die Beendigung der Transaktion mit den Resource-Managern über das *Zwei-Phasen-Commit-Protokoll* (*2PC*, [Gra78]) ab.

Nimmt mehr als eine Applikation an einer Transaktion teil, so sind jeder Applikation eigene Resource-Manager, ein eigener Transaktions-Manager und ein eigener Kommunikations-Manager zugeordnet. Eine solche Gruppe von Komponenten wird als *Instanz* bezeichnet. Instanzen werden in einer baumartigen Hierarchie organisiert. Der Transaktions-Manager der initiierenden Instanz kontrolliert als übergeordneter Transaktions-Manager den Gesamtablauf, indem er Aufrufe an die untergeordneten Transaktions-Manager propagiert und die Abwicklung der Transaktion zwischen den Instanzen gemäß dem 2PC-Protokoll durchführt. Die Kommunikation zwischen den einzelnen Instanzen erfolgt über die einzelnen Kommunikations-Manager.

Das X/Open-DTP-Modell erfüllt die Anforderungen 1a und 2a an Modelle für die Transaktionskontrolle in Grid-Umgebungen. Allerdings zeigt es mit Blick auf die anderen formulierten Anforderungen einige Schwächen. So ist die Möglichkeit zur Fortpflanzung von Anfragen innerhalb von Transaktionen nur für Applikationen beschrieben. Sie können zur Bearbeitung einer Anfrage gegebenenfalls auf andere Applikationen zurückgreifen. Ob Resource-Manager diese Möglichkeit ebenfalls haben oder jeweils nur auf die von ihnen verwaltete Datenquelle zugreifen können, bleibt unklar. Damit ist die Anforderung 1b nur teilweise erfüllt.

Das Recht zur Demarkation der Transaktion liegt ausschließlich bei der initiierenden Applikation. Weiterhin obliegt die Koordination der Transaktion stets dem Transaktions-Manager der initiierenden Instanz. Diese funktionalen Rollen können während einer Transaktion nicht weitergegeben werden, so dass jeder Knoten auf die Rolle festgelegt ist, mit der er der Transaktion beigetreten ist. Die Anforderung 2b bleibt somit ebenfalls unerfüllt. Diese Schwächen des X/Open-DTP-Modells hinsichtlich Anfragefortpflanzung und flexibler Rollenverteilung erschweren eine dynamische Entwicklung der Ablaufstrukturen.

4.2 WS-Coordination und WS-AtomicTransaction

Durch die Komposition von Web-Services können komplexe Einheiten entstehen. Die Gesamtheit der in einer solchen Einheit von den Teilnehmern ausgeführten Aktionen wird als *Aktivität* bezeichnet. Die Spezifikation der *Web-Services-Coordination* (*WS-Coordination*, [CCC+03a]) definiert ein erweiterbares Framework für die Koordination dieser verteilten Aktionen. Die Koordination hat das Ziel, dass alle Teilnehmer an der Aktivität eine Übereinkunft über das Ergebnis der verteilten Aktionen erzielen. Zentrale Bestandteile des Frameworks sind ein Koordinations-Service sowie anwendungsspezifische Koordinations-Typen, die den erweiterbaren Teil der Spezifikation bilden. *Web-Services-Atomic-Transaction* (*WS-AtomicTransaction*, [CCC+03b]) definiert einen Koordinations-Typ, der den Teilnehmern an einer kurzlebigen Aktivität mit Alles-oder-Nichts-Eigenschaften ermöglicht, eine Übereinkunft über das Ergebnis der Aktivität zu erzielen. Das *Web-Services-Business-Activity-Framework* (*WS-BusinessActivity*, [CCC+04]) beschreibt im Gegensatz dazu einen Koordinations-Typ für langlebige Aktivitäten wie Workflows, die nicht im Fokus dieses Artikels liegen und daher im Folgenden nicht weiter berücksichtigt werden.

WS-Coordination

Nach der WS-Coordination-Spezifikation besteht ein Koordinations-Service (kurz Koordinator) aus den folgenden Komponenten:

Die *Coordination-Protocol-Services* ermöglichen die Koordination nach den vom Koordinations-Service implementierten Koordinations-Protokollen. Sie werden in den Spezifikationen der jeweiligen Koordinations-Typen, etwa WS-AtomicTransaction (siehe unten), definiert.

Der *Activation-Service* reagiert auf die *CreateCoordinationContext*-Nachricht einer Anwendung durch die Erzeugung einer neuen Aktivität und die Rückgabe eines entsprechenden *Coordination-Context* (Koordinations-Kontextes). Die Anwendung gibt diesen an die anderen Anwendungen, die koordiniert werden sollen, weiter. Er enthält die notwendigen Informationen, um sich für die jeweilige Aktivität und den gewünschten Koordinations-Typ registrieren zu können. Die genaue Semantik der CreateCoordinationContext-Schnittstelle wird durch die Spezifikation des jeweiligen Koordinations-Typs bestimmt.

Der *Registration-Service* definiert eine *Register*-Schnittstelle, an der sich Web-Services für die Koordination nach einem Protokoll des vom Koordinator unterstützten Koordinations-Typs anmelden können. Dazu wird eine *Register*-Nachricht, die eine Referenz auf den Coordination-Protocol-Service des Teilnehmers enthält, an den Registration-Service gesendet. Die an den sich registrierenden Teilnehmer zurückgesendete *RegisterResponse*-Nachricht enthält eine Referenz auf den Coordination-Protocol-Service des Koordinators. Damit ist eine Verbindung zwischen den beiden Coordination-Protocol-Services hergestellt und die protokollspezifischen Nachrichten können entsprechend adressiert werden.

Eine Anwendung, die einen Koordinations-Kontext erhält, kann den Registration-Service der aufrufenden Anwendung oder den eines anderen vertrauenswürdigen Koordinators benutzen; letzterer muss sich dann bei dem Registration-Service der aufrufenden Anwendung registrieren. Dadurch kann eine Baumstruktur von Koordinations-Services entstehen, bei der jeweils der übergeordnete Knoten seine untergeordneten Koordinatoren kontrolliert. Dem Koordinator des die Aktivität initiierenden Web-Services obliegt somit die Kontrolle über die gesamte Aktivität. Er kann daher als *Super-Koordinator* bezeichnet werden.

WS-AtomicTransaction

Die Spezifikation beschreibt drei Protokolle für die Koordination von Aktivitäten als atomare Ablaufeinheit. Teilnehmer, die sich für *Completion* registrieren, können den Koordinator auffordern, die laufende Transaktion festzuschreiben oder abzubrechen. Das Resultat dieser Abwicklung wird dem initiierenden Teilnehmer mitgeteilt. Das *Two-Phase-Commit-Protokoll* (*2PC*) koordiniert die Entscheidung der Teilnehmer über den Abschluss einer verteilt durchgeführten Aktivität und gewährleistet, dass jeder Teilnehmer über die Entscheidung informiert wird. Es wird unterteilt in die beiden Varianten *Volatile 2PC* und *Durable 2PC* für Teilnehmer, die flüchtige beziehungsweise persistente Ressourcen verwalten. Ein Teilnehmer kann sich für mehrere Protokolle registrieren, indem er unterschiedliche *Register*-Nachrichten versendet.

Das WS-Coordination-Framework in Verbindung mit dem Koordinations-Typ WS-AtomicTransaction erfüllt die Anforderungen 1a, 1b und 2a an Modelle für die Transaktionskontrolle in Grid-Umgebungen. Allerdings wird aus der Spezifikation von WS-AtomicTransaction nicht deutlich, welche Teilnehmer an der Aktivität sich für das Completion-Protokoll registrieren können und daher das Recht zur Demarkation der Transaktion haben. Aus diesem Grund ist nicht zu beantworten, ob Knoten im Laufe der Transaktion die Rolle eines demarkationsberechtigten Knotens einnehmen können. Unklar bleibt insbesondere, ob die initiierende Applikation beeinflussen kann, inwiefern andere Knoten sich für das Completion-Protokoll anmelden können.

Unabhängig von der Flexibilität, die durch das Completion-Protokoll auf der Ebene des Koordinations-Typs potenziell ermöglicht wird, legt die Reihenfolge, in der die Teilnehmer der Aktivität aufgerufen werden, die Struktur auf der Ebene des Koordinations-Service fest. Der Super-Koordinator bildet zwangsläufig die Wurzel einer Baumstruktur, die sich aus der schrittweisen Registrierung der Koordinatoren anderer Teilnehmer ergibt und eine Koordinations-Hierarchie zur Folge hat. Die Rolle des Super-Koordinators kann daher innerhalb einer Transaktion nicht von einem anderen Knoten übernommen werden.

Die Kombination aus dem WS-Coordination-Framework und dem Koordinations-Typ WS-AtomicTransaction erlaubt die Fortpflanzung von Anfragen und ermöglicht eine flexiblere Rollenverteilung als das X/Open-DTP-Modell. Aufgrund der Unklarheiten im Zu-

sammenhang mit dem Completion-Protokoll und der vorgegeben Koordinator-Hierarchie wird Anforderung 2b jedoch nicht erfüllt. Im Gegensatz zu X/Open DTP und WS-Coordination/WS-AtomicTransaction erfüllt unser im nächsten Abschnitt vorgestellter Ansatz alle in Abschnitt 2 beschriebenen Anforderungen.

5 DAS MODELL: TracG

Der vorgeschlagene Ansatz zur transaktionalen Aktivitätskontrolle in Grid-Umgebungen (***Tr**ansactional **A**ctivity **C**ontrol for the **G**rid, TracG*) gestattet die dynamische Entwicklung von Ablaufstrukturen durch die Einführung von Regeln, die den Rahmen für mögliche Abläufe festlegen. Die Grundlage der möglichen Ablaufstrukturen bilden die Kombinationen unterschiedlicher Interaktionsarten in Grid-Umgebungen, beispielsweise einfache Anfragen, Aufrufe mit Auslieferung der Anfrage-Ergebnisse an eine dritte Partei und das automatische Ausführen von Anfragen in regelmäßigen Zeitabständen oder abhängig von bestimmten Ereignissen mit Auslieferung der Anfrage-Ergebnisse an vorher bestimmte Knoten.

5.1 Koordination der Transaktion

Im Zentrum unseres Ansatzes steht eine Koordinationsinstanz, die den Ablauf der Transaktionen nach dem 2PC-Protokoll koordiniert. Die Benutzung des 2PC-Protokolls garantiert, dass entweder alle an der Transaktion beteiligten Instanzen ihre Aufgaben erfolgreich ausführen oder keine.

Für die Koordination der Transaktion wird ein *Koordinations-Service*, der analog zu WS-Coordination einen *Aktivierungs-Service* und einen *Registrierungs-Service* umfasst, benötigt. Dieser Service kann entweder durch einen speziellen Grid-Service oder auch von einem an der Transaktion beteiligten Knoten, der die notwendigen Schnittstellen implementiert, erbracht werden. Der initiierende Knoten sucht für die zu startende Transaktion einen Koordinations-Service (oder kennt bereits den entsprechenden *Grid-Service-Handle, GSH*). Der Knoten schickt eine *CreateCoordinationContext*-Nachricht an den Aktivierungs-Service des Koordinators. Diese Nachricht kann eine Liste von GSHs der Knoten, die an der Transaktion teilnehmen sollen, enthalten. Der Aktivierungs-Service legt daraufhin einen neuen Koordinations-Kontext an und gibt einen Transaktions-Kontext an den Knoten zurück. Die in der *CreateCoordinationContext*-Nachricht berücksichtigten Knoten erhalten eine RegisterResponse-Nachricht, um über die erfolgte Registrierung beim Koordinator informiert zu werden.

Der *Transaktions-Kontext* enthält eine Transaktions-ID und den GSH des aktuellen Koordinators**.** Der *Koordinations-Kontext* enthält eine Koordinations-Kontext-ID, die entsprechende Transaktions-ID und eine Liste von GSHs zu koordinierender Knoten sowie Attribute zur Steuerung der Rechteweitergabe.

Anforderung: transaktionale Datenmanipulation

Knoten senden nach Erhalt des Transaktions-Kontextes eine *Register*-Nachricht an den Registrierungs-Service des Koordinators, um sich für die Transaktion zu registrieren. Diese Nachricht enthält den eigenen GSH und die entsprechende Transaktions-ID. Der GSH wird im Koordinations-Kontext zur Liste der an der jeweiligen Transaktion teilnehmenden Knoten hinzugefügt. Ein Knoten kann ähnlich dem *Transactional-Server* des *CORBA Object-Transaction-Service* der Object Management Group [OMG03] selbst nicht-rücksetzbar sein, aber trotzdem den Transaktions-Kontext an andere zurücksetzbare Knoten weitergeben.

Durch die Weitergabe des Transaktions-Kontextes beim Einbeziehen neuer Knoten in die Transaktion und die Registrierung neuer Knoten beim Koordinator umfasst die transaktionale Kontrolle alle an der Aktivität beteiligten Knoten, so dass Anforderung 1b, insbesondere natürlich auch Anforderung 1a, erfüllt sind.

Anforderung: Einnahme unterschiedlicher Rollen

Der die Transaktion initiierende Knoten kann die Einnahme funktionaler Rollen durch andere Knoten über die Weitergabe von Rechten mithilfe von Parametern der CreateCoordinationContext-Nachricht beeinflussen. Der resultierende Koordinations-Kontext enthält die Attribute *CommitRightPropagation* und *ChangeCoordinatorRightPropagation.* Sie geben an, ob das Recht zur Demarkation der Transaktion beziehungsweise zur Bestimmung eines neuen Koordinators nicht weitergegeben werden kann (Attributwert *NotAllowed*), nur abgegeben werden kann (Attributwert *HandOver*) oder geteilt wird (Attributwert *Share*). Letzteres ermöglicht, dass auch mehrere Instanzen gleichzeitig das Recht zur Demarkation der Transaktion beziehungsweise Bestimmung des Koordinators haben können. Die Knoten können die jeweiligen Attribut-Werte durch geeignete Operationen des Koordinators abfragen.

Im Koordinations-Kontext gibt es zu jedem Knoten ein Attribut *CommitRight*, das beschreibt, ob ein Knoten über das Recht zur Demarkation der Transaktion verfügt. Das Recht eines Knotens, einen neuen Koordinator zu bestimmen, wird durch ein Attribut *ChangeCoordinatorRight* ausgedrückt. Analog zu *CommitRight* wird dieses Attribut für jeden an der Transaktion beteiligten Knoten im Koordinations-Kontext verwaltet. Diese Attribute können jeweils drei Werte annehmen: *Granted*, *NotGranted* und *Never*. Die Werte *Granted* und *NotGranted* zeigen den Status eines Knotens bezüglich eines Rechtes an. Der Wert *Never* bietet dem die Transaktion initiierenden Knoten die Möglichkeit, ausdrücklich zu verhindern, dass ein Knoten das Recht zur Demarkation der Transaktion oder zur Bestimmung eines neuen Koordinators erhalten kann.

Die Inhaber eines Rechtes können die Attribute *ReleaseCommitRight* bzw. *ReleaseChangeCoordinatorRight* des Koordinations-Kontextes benutzen, um allen Knoten die Möglichkeit zu geben, das jeweilige Recht zu erhalten. Wenn diese Attribute über eine *ReleaseCommitRight*- bzw. *ReleaseChangeCoordinatorRight*-Nachricht gesetzt wurden, kann jeder Knoten (außer denen, deren *CommitRight*- bzw. *ChangeCoordinatorRight*-Attribute auf den Wert *Never* gesetzt sind) durch eine *RequestCommitRight*- bzw. *RequestChangeCoordinatorRight*-Nachricht an den Koordinator das jeweilige Recht anfordern. Bei Erhalt dieser Nachricht muss der Koordinator bei geforderter Abgabe des jeweiligen Rechtes, also *CommitRightPropagation.HandOver* oder *ChangeCoordinatorRightPropagation.HandOver,* allen anderen Klienten die Möglichkeit, das entsprechende Recht zu erhalten, nehmen. Dazu entfernt er das *ReleaseCommitRight*- bzw. *ReleaseChangeCoordinatorRight*-Attribut aus dem Koordinations-Kontext.

Um das Recht zur Demarkation der Transaktion beziehungsweise zur Bestimmung eines neuen Koordinators gezielt an andere Knoten weiterzugeben, sendet ein Rechteinhaber eine *GrantCommitRight*- beziehungsweise eine *GrantChangeCoordinatorRight*-Nachricht an den Koordinator. Diese enthält den GSH des Knotens, der das Recht erhalten soll. Die Weitergabe der Rechte erfolgt nur, wenn das zugehörige Attribut *CommitRightPropagation* beziehungsweise *ChangeCoordinatorRightPropagation* nicht auf *NotAllowed* gesetzt ist. In diesem Fall wird in dem Koordinations-Kontext das Recht des Knotens (Attributwert *Granted*) vermerkt, der Knoten, der das Recht erhält, entsprechend informiert und dem weitergebenden Knoten das Ergebnis der Rechteweitergabe mitgeteilt.

Knoten haben auch in einer bereits laufenden Transaktion die Möglichkeit, bei Bedarf den Koordinations-Service aufzufordern, die Koordination der Transaktion an einen anderen Koordinations-Service abzutreten. Dazu muss der Knoten eine *ChangeCoordinator*-Nachricht mit dem GSH des neuen Koordinators an den aktuellen Koordinator senden. Der aktuelle Koordinator überprüft, ob der GSH zu einem anderen Koordinations-Service gehört, und gibt daraufhin den Koordinations-Kontext an den Knoten mit dem entsprechenden GSH weiter. Der Wechsel des Koordinators wird den an der Transaktion teilnehmenden Knoten mitgeteilt; dazu sendet der alte Koordinator an alle Transaktionsteilnehmer einen geänderten Transaktions-Kontext, der den GSH des neuen Koor-

dinators und die alte Transaktions-ID enthält. Welcher Knoten die Koordination letztlich durchführt, steht also erst beim Aufruf einer Operation zur Demarkation der Transaktion fest.

Knoten, die das Recht zur Demarkation der Transaktion haben, können den Koordinator durch eine *Start2PC*-Nachricht zum Starten des Zwei-Phasen-Commit-Protokolls veranlassen. Diese Nachricht enthält die jeweilige Transaktions-ID und den GSH des sendenden Knotens.

Alle an der Transaktion beteiligten Knoten können im Fehlerfall den Koordinator durch eine *Rollback*-Nachricht zum Abbruch der Transaktion auffordern. Die *Rollback*-Nachricht muss die jeweilige Transaktions-ID enthalten.

Die beschriebenen Mechanismen ermöglichen die Einnahme der Rolle eines demarkationsberechtigten Knotens und eines Koordinators innerhalb eines komplexen Ablaufs. Insbesondere ist damit auch die flexible, dynamisch änderbare Rollenverteilung unter den beteiligten Knoten gewährleistet. Somit erfüllt der von uns beschriebene Ansatz die Anforderungen 2a und 2b.

5.2 Fähigkeiten der Knoten

Die verschiedenen Rollen im Zusammenhang mit der transaktionalen Durchführung einer Aktivität können prinzipiell von beliebigen Knoten, Grid-Services und auch Web-Services, übernommen und während der Laufzeit der Transaktion gewechselt werden. Ein Knoten kann jedoch bei der Teilnahme an einer Transaktion Rollen nur im Rahmen seiner Fähigkeiten übernehmen. Er muss dazu jeweils Schnittstellen anbieten, etwa für die Weitergabe des Transaktions-Kontextes, für die Entgegennahme und Weitergabe des Rechtes zur Transaktions-Demarkation oder für die Abwicklung der Transaktion gemäß dem 2PC-Protokoll. Dabei müssen die Fähigkeiten des Knotens für andere Knoten erkennbar – also durch entsprechende Operationen abfragbar – sein, damit er korrekt und seinen Möglichkeiten entsprechend in die Transaktion eingebunden werden kann.

5.3 Szenario

Zur Verdeutlichung des vorgeschlagenen Ansatzes wird anhand eines Beispiel-Szenarios die dynamische Entwicklung einer Ablaufstruktur erläutert (siehe Abb. 3 und Abb. 4). Das Szenario abstrahiert zu Gunsten der Übersichtlichkeit von einigen Details, wie Antwort- bzw. Bestätigungsnachrichten, etwa bezüglich der Registrierung der Knoten, die von Anfang an Teilnehmer der Transaktion sind, und der Weitergabe des Rechts zur Demarkation der Transaktion an Client 2, sowie der eigentlichen Durchführung des 2PC-Protokolls.

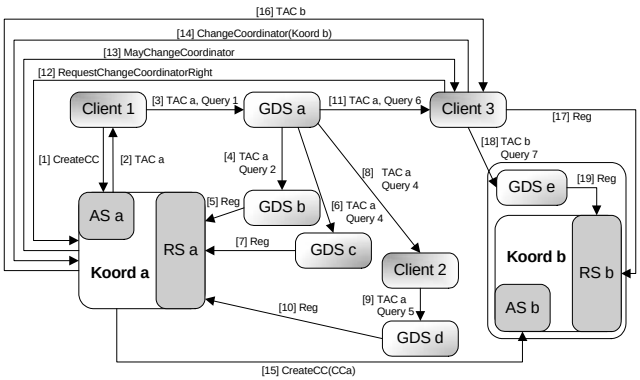

Abb. 3: Beispiel-Szenario für die dynamische Entwicklung einer Ablaufstruktur mit Weitergabe des Rechts zur Demarkation der Transaktion und Weitergabe des Koordinations-Kontextes: Initialisierung der Koordination und Registrierungsvorgang.

Ausgangssituation

Kooperationen wissenschaftlicher und/oder kommerzieller Einrichtungen können das Grid-Data-Computing nutzen, um ihre lokal erhobenen Daten gemeinsam verwalten zu können. Dazu müssen diese vor dem Einbringen in den gemeinsamen Datenbestand validiert werden. Das hier vorgestellte fiktive Szenario betrifft einen Zusammenschluss von Wissenschaftlern für die Erforschung der Gen-Sequenzen von Hefepilzen.

Client 1 will die Ergebnisse seiner Untersuchungen zu Gen-Sequenzen von Hefepilzen in den dafür vorgesehenen Grid-Data-Service GDS a schreiben. Dieser GDS wird von mehreren Forschungsgruppen für die gegenseitige Überprüfung und den Austausch der Forschungsergebnisse benutzt. Die Daten dürfen erst nach erfolgreicher Überprüfung von dem Grid-Data-Service übernommen werden. Client 1 sendet seine komplexe Update-Anfrage an den GDS a, der zur Erledigung der Anfrage GDS b und GDS c aufruft. Client 1 veranlasst die Weiterleitung der Ergebnisse einer Überprüfungsabfrage durch den GDS a an Client 2, der die Überprüfung mit Hilfe des GDS d durchführen kann. Client 1 überträgt das Recht zur Demarkation der Transaktion an Client 2, das dieser zur Durchführung der Überprüfung ausüben kann. Hintergrund für diese Übertragung des Demarkationsrechtes ist, dass Client 1 bekannt ist, dass seine Daten erst nach erfolgreicher Prüfung durch Client 2 in GDS a festgeschrieben werden und daher Client 2 besser den Zeitpunkt für die Demarkation der Transaktion bestimmen kann. Client 3 hat sich bei GDS a angemeldet, um im Falle von Update-Operationen ebenfalls zur Überprüfung herangezogen und in die Transaktion einbezogen zu werden. Client 3 nutzt dafür den Datenbestand von GDS e.

Ablauf

Die Initialisierung der Koordination und den Registrierungsvorgang zeigt Abb. 3: Client 1 startet eine Transaktion, indem er eine CreateCoordinationContext-Nachricht an den Aktivierungs-Service des Koordinators a als dedizierten Koordinations-Service sendet [1]. Die Nachricht enthält die GSHs von GDS a und Client 2, die in die Transaktion aufgenommen werden sollen. In der Nachricht ist das *CommitRight*-Attribut von Client 2 auf den Wert *Granted* gesetzt, so dass dieser Knoten das Recht zur Demarkation der Transaktion erhält. Die Nachricht enthält weiterhin die Parameter *CommitRightPropagation.HandOver* und *ChangeCoordinatorRightPropagation.Share* sowie *ReleaseChangeCoordinatorRight*. Damit kann ein neuer Koordinations-Kontext erzeugt werden, der die Liste der GSHs von Client 1, Client 2 sowie GDS a enthält, die Abgabe des Rechtes zur Demarkation sowie die Teilung des Rechtes zur Bestimmung des Koordinators erlaubt und Knoten gestattet, das Recht zur Bestimmung eines neuen Koordinators anzufordern. Der Koordinator a sendet den erzeugten Transaktions-Kontext TAC a an Client 1 [2]. Client 1 übergibt den Transaktions-Kontext an GDS a und stellt seine komplexe Update-Anfrage [3]. GDS a zieht für die Bearbeitung der Anfrage GDS b und GDS c heran und übergibt ihnen den Transaktions-Kontext [4, 6]. GDS b und c registrieren sich beim Registrierungs-Service des Koordinators a [5, 7]. GDS a informiert Client 2 über die an ihn gestellte Anfrage und übergibt den Transaktions-Kontext [8]. Client 2 übergibt den Transaktions-Kontext an GDS d und stellt seine Anfrage [9]. GDS d registriert sich bei Koordinator a [10]. GDS a informiert Client 3 über die an ihn gestellte Anfrage und übergibt den Transaktions-Kontext [11]. Client 3 fordert vom Koordinator a das Recht zur Bestimmung eines neuen Koordinators an [12], welches ihm erteilt wird [13]. Daraufhin bestimmt Client 3 Koordinator b als neuen Koordinator [14]. Dieser zweite Koordinations-Service wird von einem Knoten implementiert, der auch den von Client 3 zu benutzenden GDS e beherbergt. Aus Effizienz- und Kostengründen zieht Client 3 daher diesen Koordinator vor. Der Koordinator a übergibt den Koordinations-Kontext an Koordinator b [15]. Client 3 erhält von Koordinator a nach der Übertragung der Koordination auf Koordinator b den neuen Transaktions-Kontext TAC b, der den GSH von Koordinator b enthält [16]. Auch alle anderen Transaktionsteilnehmer erhalten den neuen Transaktionskontext und werden so über den Koordinatorwechsel informiert (in Abb. 3 nicht dargestellt). Client 3 registriert sich bei Koordinator b [17], übergibt den Transaktions-Kontext

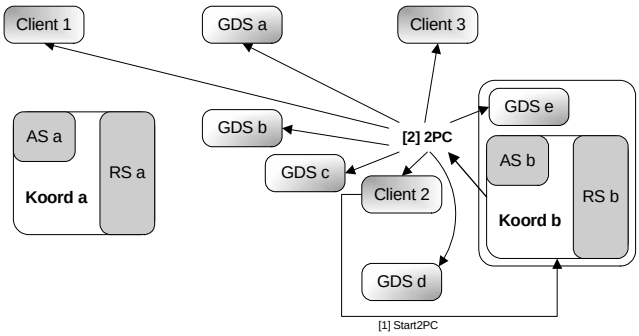

Abb. 4: Beispiel-Szenario für die dynamische Entwicklung einer Ablaufstruktur mit Weitergabe des Rechts zur Demarkation der Transaktion und Weitergabe des Koordinations-Kontextes: Demarkation und Koordination der Transaktion.

an GDS e und stellt seine Anfrage [18]. GDS e registriert sich beim neuen Koordinator b [19]. Damit sind die Initialisierung der Koordination und der Registrierungsvorgang abgeschlossen.

Abb. 4 zeigt die anschließende Demarkation und Koordination der Transaktion: Client 2 übt sein Recht zur Demarkation der Transaktion aus und sendet eine *Start2PC*-Nachricht an Koordinator b [1]. Koordinator b führt das 2PC-Protokoll mit den registrierten Knoten durch [2]. Nur wenn die Überprüfungen der Daten von Client 1 durch Client 2 und Client 3 positiv verlaufen, werden die Daten in GDS a festgeschrieben und Client 1 entsprechend unterrichtet. Anderenfalls werden alle Änderungen zurückgenommen.

Unser TracG-Modell ermöglicht die Bewahrung der transaktionalen Kontrolle bei dynamischer Fortpflanzung von Aufrufen und Anfragen zwischen Knoten im Grid. Im Szenario wird dies beispielsweise bei den Schritten [4] und [5] deutlich. In diesem Fall kann GDS a die an ihn gerichtete Anfrage nur bearbeiten, indem er GDS b und GDS c befragt. Die Weitergabe des Demarkationsrechtes erlaubt es, die Demarkation durch einen Teilnehmer vornehmen zu lassen, der hierzu besser qualifiziert ist als der Initiator der Aktivität. Der Wechsel des Koordinators, im Szenario in den Schritten [14] bis [16] dargestellt, gestattet die Wahl des optimalen Koordinators durch einen entsprechend befugten Transaktions-Teilnehmer.

6 FAZIT

Grid-Data-Computing stellt ein neues Verarbeitungs-Paradigma dar, das sich insbesondere durch sehr hohe Flexibilität der Einbeziehung von Web-Services bzw. Grid-Data-Services in datenintensive, verteilte Verarbeitungsvorgänge auszeichnet. Derartige Abläufe können ausgehend von speziellen Anwendungsanforderungen sowohl statische Muster, wie z.B. Auslieferung der Ergebnisse einer Anfrage zur Weiterverarbeitung an eine dritte Partei, als auch sich vollkommen dynamisch entwickelnde, beliebig komplexe Strukturen annehmen. Existierende Ansätze der Transaktionskontrolle für solche Vorgänge, wie z.B. X/OPEN DTP oder WS-Coordination und WS-AtomicTransaction unterstützen die angesprochenen Muster unzureichend und/oder tragen der inhärenten Dynamik unzureichend Rechnung. Obwohl diese Ansätze eine Kontrolle sich dynamisch fortpflanzender Vorgänge ermöglichen, führt Letzteres unnötigerweise dazu, dass eine Weitergabe des Demarkationsrechtes für Transaktionen bzw. ein Wechseln der Rolle des Koordinators nur unzureichend unterstützt werden.

Dieses Defizit wird durch das in diesem Beitrag vorgeschlagene Modell TracG ausgeräumt. TracG beschreibt entsprechende Konzepte und zugehörige Protokolle, die sich für den Einsatz sowohl mit Web-Services als auch mit Grid-Data-Services eignen. Insgesamt wird mit TracG ein Ansatz vorgeschlagen, der den grundlegenden Prinzipien des Grid-Data-Computing besser angepasst ist als die existierenden Ansätze, da er Flexibilität und Dynamik konsequenter berücksichtigt. Es wäre wünschenswert gewesen, das Modell als Koordinationstyp für den nahe liegenden Standard WS-Coordination zu konzipieren. Eine derartige Vorgehensweise hätte jedoch die Übernahme der Einschränkungen von WS-Coordination bezüglich der angeführten Anforderungen impliziert.

Durch die Verwendung des 2PC-Protokolls kann es bei der Fortpflanzung von Aufrufen im TracG-Modell zu unerwünschten Situationen kommen, etwa wenn sich zum Zeitpunkt der Demarkation der Transaktion noch nicht alle Knoten beim Koordinator registriert haben. Diese Probleme sollen zukünftig durch eine Erweiterung des 2PC-Protokolls berücksichtigt werden. Nach weiterer Ausgestaltung des Modells ist es beabsichtigt, eine vollständige Spezifikation von TracG in die Diskussion des OGSA-DAI-Projekts [OGSA-DAI] einfließen zu lassen.

Danksagung. Die Autoren danken den anonymen Gutachtern für die äußerst hilfreichen Anmerkungen.

LITERATUR

[AAM+03] Antonioletti, M., M. Atkinson, S. Malaika, S. Laws, N. Paton, D. Pearson und G. Riccardi (2003): Grid Data Service Specification. http://www.cs.man.ac.uk/grid-db/papers/draft-ggf-dais-spec-ggf9.pdf

[ADG+03] Atkinson, M., V. Dialani, L. Guy, I. Narang, N. Paton, D. Pearson, T. Storey und P. Watson (2003): Grid Database Access and Integration: Requirements and Functionalities. http://www.cs.man.ac.uk/grid-db/papers/DAIS_GGF7RF.pdf

[CCC+03a] Cabrera, F., G. Copeland, W. Cox, M. Feingold, T. Freund, J. Johnson, C. Kaler, J. Klein, D. Langworthy, D. Orchard, I. Robinson, J. Shewchuck und T. Storey (2003a): Web Services Coordination (WS-Coordination). ftp://www6.software.ibm.com/software/developer/library/ws-coordination.pdf

[CCC+03b] Cabrera, F., G. Copeland, W. Cox, M. Feingold, T. Freund, J. Johnson, C. Kaler, J. Klein, D. Langworthy, A. Nadalin, I. Robinson, T. Storey und S. Thatte (2003b): Web Services Atomic Transaction (WS-AtomicTransaction). ftp://www6.software.ibm.com/software/developer/library/ws-atomictransaction.pdf

[CCC+04] Cabrera, F., G. Copeland, W. Cox, T. Freund, J. Klein, D. Langworthy, I. Robinson, T. Storey und S. Thatte (2004): Web Services Business Activity Framework (WS-BusinessActivity). ftp://www6.software.ibm.com/software/developer/library/ws-busact.pdf

[FKN+02] Foster, I., C. Kesselmann, J. Nick und S. Tuecke (2002): The Physiology of the Grid - An Open Grid Services Architecture for Distributed Systems Integration. http://www.globus.org/research/papers/ogsa.pdf

[FKT01] Foster, I., C. Kesselman, S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International J. Supercomputer Applications, 15(3), 2001.

[FTU+03] Foster, I., S. Tuecke und J. Unger (2003): OGSA Data Services. http://www.cs.man.ac.uk/grid-db/papers/draft-ggf-dais-dataservices-ggf9.pdf

[GGM+03] Gawlick, D., V. Gogate, C. Madsen, S. Mishra, I. Narang und M. Subramanian (2003): Data Distribution in the Grid Environment. http://www.cs.man.ac.uk/grid-db/papers/draft-ggf-dais-dist-ggf9-2.pdf

[Gra78] Gray, J. N. (1978): Notes on Data Base Operating Systems. In Operating Systems: An Advanced Course, Springer-Verlag Lecture Notes in Computer Science 60:393 - 481, 1978.

[OGSA-DAI] Website des OGSA-DAI-Projektes: http://www.ogsadai.org.uk

[OMG03] OMG (2003): Transaction Service Specification, Version 1.4. http://www.omg.org/cgi-bin/apps/doc?formal/03-09-02

[PAD+02] Paton, N., M. Atkinson, V. Dialani, D. Pearson, T. Storey und P. Watson (2002): Database Access and Integration Services on the Grid. http://www.cs.man.ac.uk/grid-db/papers/dbtf.pdf

[QYT04] Qi, Z., J. You, Y. Jin und F. Tang (2004): GridTP Services for Grid Transaction Processing. Wird erscheinen in: Proc. of GCC2003.

[TCF+03] Tuecke, S., K. Czaijkowski, I. Foster, J. Frey, S. Graham, C. Kesselmann, T. Maquire, T. Sandholm, D. Snelling und P. Vanderbilt (2003): Open Grid Services Infrastructure (OGSI). http://www.globus.org/research/papers/Final_OGSI_Specification_V1.0.pdf

[Wat01] Watson, P. (2001): Databases and the Grid. Technical Report CS-TR-755, University of Newcastle, 2001.

[XOp96] X/Open 1996: Distributed Transaction Processing: Reference Model, Version 3. http://www.opengroup.org/online-pubs?DOC=009249599&FORM=PDF