P2P-Datenmanagement für e-Science-Grids

T. Scholl B. Gufler J. Müller A. Reiser A. Kemper
Lehrstuhl für Datenbanksysteme, Technische Universität München, Fakultät für Informatik
Boltzmannstr. 3, 85748 Garching bei München

Über die bereits vorhandenen Datenvolumina hinaus, stellt insbesondere die antizipierte Datenflut neuer e-Science-Projekte Forscher vor neue Herausforderungen. Im Rahmen des AstroGrid-D-Projektes der deutschen Grid-Initiative (D-Grid) erforschen wir mit dem HiSbase-System ein durchsatzoptimiertes Datenmanagement für fachspezifische Forschungsverbünde, das wir in Zusammenarbeit mit den führenden Zentren der Astrophysik innerhalb Deutschlands entwickeln. HiSbase ermöglicht die verteilte Datenverwaltung für e-Science-Forschungsverbünde, indem die jeweils vorherrschenden Dateneigenschaften (z. B. Schieflagen in der Datenverteilung) und Anfragemuster in die Optimierung einbezogen werden. Durch die Kombination von Ansätzen zur Datenpartitionierung und Replikation sowie von P2P Technologien kann HiSbase interessante Themenkomplexe wie Datenlastbalancierung, Berücksichtigung von Anfrage-Hotspots und die Anpassung sowohl an kurzfristige als auch an langfristige Lastverschiebungen adressieren.

1 Einleitung

E-Science-Projekte zahlreicher Fachbereiche, wie etwa Biologie, Geografie, Hochenergiephysik oder Astrophysik, sind mit immensen Datenmengen aus aktuellen Experimenten konfrontiert. In Anbetracht anstehender astrophysikalischer Projekte wie z. B. dem *Panoramic Survey Telescope and Rapid Response System (Pan-STARRS)* mit einer Datenrate von mehreren Terabytes pro Tag ist absehbar, dass der bisher verfolgte Ansatz einer zentraler Datenhaltung [Szalay et al. 2002] an seine Grenzen stoßen wird.

Häufig ist in e-Science-Applikationen die Korrelation von Daten mehrerer Experimente von hohem Interesse, da sich aus solchen Kombinationen neue Einsichten gewinnen und Ergebnisse ableiten lassen. Bisher werden meistens die Experimentergebnisse auf den Rechnern des Instituts abgespeichert, welches das Experiment durchführt. Diese Strategie ist für das gerade beschriebene Anwendungsszenario kontraproduktiv, da alle Datenquelle einzeln befragt und Zwischenergebnisse über das Netzwerk transferiert werden müssen.

Problemstellung. Um der enormen Datenmenge Herr werden zu können, schließen sich die Wissenschaftler innerhalb eines Forschungsverbundes zu *Virtuellen Organisationen* zusammen und schaffen Infrastrukturen für ihre Föderationen, sogenannte Data Grids [Venugopal et al. 2006]. Innerhalb dieser Infrastrukturen werden dedizierte Ressourcen über Breitband-Netzwerke miteinander verknüpft, um den Forschern nun zu ermöglichen die unterschiedlichen Forschungsergebnisse miteinander zu verknüpfen. Um die Reproduzierbarkeit der Ergebnisse zu gewährleisten, werden einmal veröffentlichte Ergebnisse nicht mehr geändert. Stattdessen werden zusätzliche, erweiterte Versionen öffentlich zugänglich gemacht. Zudem stellt die zunehmende Popularität der Daten innerhalb der Forschungsverbünde hohe Anforderungen an die verschiedenen Architekturen, wie z. B. das Erzielen eines hohen Durchsatzes. Weitere herausfordernde Aspekte sind in diesem Zusammenhang *Schieflagen der Datenverteilung* innerhalb der Archive sowie *Anfrage-Hotspots*.

Forschungsverbünde im Bereich e-Science benötigen Konzepte für Plattformen, die wissenschaftlichen Datenaustausch unterstützen und dabei 1) direkt eine Datenmenge von mehreren Terabyte oder Petabyte verarbeiten können, 2) existierende Verbundnetzwerke und Rechenkapazitäten mit einigen hundert Rechnern integrieren und 3) einen hohen Durchsatz liefern, um mit der ständig steigenden Anzahl an Nutzern skalieren zu können. Unter Berücksichtigung dieser Kriterien entwerfen wir ein skalierbares Datenmanagement, das sowohl Daten- als auch Anfragelastbalancierung ermöglicht.

Unser Ansatz. HiSbase bietet einen dezentralen und skalierbaren Datenmanagementansatz, der zur Verfügung stehende Ressourcen – sowohl Speicher als auch Rechenkapazität – innerhalb einer Forschungs-Community nutzt und sich damit den skizzierten Herausforderungen (Durchsatzoptimierung bei großen Datenmengen und Korrelation verteilter Datenquellen) stellt. Aufbauend auf verteilten Hashtabellen (distributed hashtables, DHT), werden Daten anstatt abhängig von ihrer Herkunft in HiSbase aufgrund häufiger Anfragemerkmale gemäß einer Community-spezifischen Verteilungsfunktion partitioniert und – falls notwendig – auch repliziert. Im Falle mehrdimensionaler Daten sorgen raumfüllende Kurven für das Erhalten logischer Nähe. Mit Hilfe der Partitionierung, die auf Datenverteilungshistogrammen basiert, gleichen wir Schieflagen (*skew*) in den verwalteten Daten aus und behalten die Anfragelokalität bei.

Anwendungsbeispiel. In der Astrophysik wie in anderen Wissenschaftsbereichen erwarten wir neben dem gewaltigen schon existierenden Datenvolumen exponentielle Wachstumsraten. Außerdem wird das Bedürfnis nach einer skalierbaren und effizienten Datenhaltung durch höhere Zugriffsraten verstärkt, da Forscher zunehmend mit diesen Informationssystemen arbeiten. In vielen Applikationen spielt die Fusion und Kombination von Beobachtungsdaten aus verschiedenen Datenquellen (die bspw. unterschiedliche Frequenzbereiche abdecken) eine Schlüsselrolle, um neue wissenschaftli-

che Erkenntnisse zu gewinnen. Das Erstellen von Wahrscheinlichkeitslandkarten für Galaxienhaufen [Schücker et al. 2004, Carlson et al. 2007] oder die Klassifikation von spektralen Energieverteilungen [Kuntschke et al. 2006] sind derartige Anwendungen. Hierbei finden insbesondere Bereichsanfragen (z. B. auf bestimmte Himmelsregionen) ihre Anwendung. Im Rahmen von AstroGrid-D [Enke et al. 2007], eines Teilprojekts der deutschen e-Science und Grid-Computing-Initiative D-Grid, wirken wir am Aufbau einer Plattform zur verteilten Informationsverarbeitung für die deutsche Astrophysik mit [Kuntschke et al. 2004, Scholl et al. 2007c, Scholl et al. 2007a, Scholl et al. 2007d].

2 Verwandte Arbeiten

Aufgrund schnell wachsender Datenmengen gewinnen verteilte Lösungen zur Datenverarbeitung immer mehr an Bedeutung. So werden in [Venugopal et al. 2006] verschiedene Designvarianten für Data Grids in einer Taxonomie zusammengefasst. Diese wird auf existierende Data Grids anhand ihrer Organisation, der Umsetzung von Datentransport, Replikation, Resourcenzuordnung und Scheduling abgebildet.

GIME [Zimmermann et al. 2006] stellt einen Ansatz zum dezentralen Management geotechnischer Informationen verschiedener Institutionen in föderierten Data Grids vor. Besonderer Wert wird auf Autonomie, standardisierten Zugriff und Zusammenarbeit zur effizienten Anfragebearbeitung gelegt. Letztere realisiert man mittels eines globalen Index basierend auf Quadtrees oder R-Trees, der die Regionen der beteiligten Archive verwaltet. Auf diese Weise reduziert sich die Anzahl der zu versendenden Nachrichten, da Anfragen nur an die Archive gesendet werden, deren Region die Anfrageregion schneiden. Allerdings kann es zu Problemen bei der Lastbalancierung kommen, falls ein Archiv für eine sehr große Region zuständig ist.

Unter den Ansätzen zur dezentralen Datenverarbeitung haben besonders P2P-Systeme aufgrund ihrer Skalierbarkeit viel Aufmerksamkeit erhalten. Insbesondere DHT-basierte System, wie z. B. Chord [Stoica et al. 2001] und Pastry [Rowstron, Druschel 2001], bieten gut skalierende und fehlertolerante Ansätze, die auch bei Schieflagen in den Datenverteilungen effiziente Punktanfragen ermöglichen. Allerdings zerstören die zur Datenverteilung auf die Peers verwendeten Hashfunktionen jegliche Nachbarschaftsbeziehungen zwischen den Daten. Dies verhindert eine effiziente Unterstützung von Bereichsanfragen, die aber in vielen Anwendungen, besonders in der Astrophysik, eine große Rolle spielen.

P-Ring [Crainiceanu et al. 2007] hingegen ermöglicht Bereichsanfragen bei gleichzeitiger Garantie eines logarithmischen Suchaufwands, insbesondere auch bei Schieflagen in der Datenverteilungen. Zur Realisierung einer garantierten Balancierung der Datenlast verwendet P-Ring Aushilfsknoten, sog. *helper peers*, die aufgrund vieler Einfügeoperationen überlastete Knoten unterstützen. Der überlastete Knoten teilt seine Daten zwischen sich und seinen Aushilfsknoten auf. Daten unterbelasteter Knoten werden entweder neu verteilt oder komplett abgegeben und somit neue Aushilfsknoten geschaffen. Im Gegensatz zu HiSbase geht P-Ring von einer gleichverteilten Anfragelast aus und betrachtet keine Anfrage-Hotspots.

HotRoD [Pitoura et al. 2006] behandelt Anfrage-Hotspots auf eindimensionalen Daten. Diese werden mittels einer lokalitätserhaltenden Hashfunktion auf einem Ring verteilt, um so auch Bereichsanfragen zu unterstützen. Zur möglichst gleichmäßigen Verteilung der Anfragelast auf alle Knoten speichern die Knoten Anfragehäufigkeiten und Bereichsgrenzen der Anfragen. Häufig angefragte Daten überlasteter Knoten werden repliziert und auf zusätzlichen rotierten virtuellen Ringen gespeichert. Schiefe Datenlasten werden in HotRoD nicht berücksichtigt.

Auch BORG [Klan et al. 2008] nutzt mehrere virtuelle Ringe, um Replikate von Datenobjekten und deren Updates dezentral zu verwalten. Im Unterschied zu HotRoD werden jedoch alle Replikate eines Objekts in einem eigenen Replikationsring verwaltet. Knoten gehören den Ringen an, zu denen sie Daten speichern. Updates in BORG können von jedem Knoten, der ein Replikat speichert, initiiert werden, wobei ein temporärer Master innerhalb des Replikationsrings den eigentlichen Updatevorgang steuert. Schieflagen in der Daten- und Anfrageverteilung werden nicht angesprochen.

In [Datta et al. 2007] wird gezeigt, dass Caching in DHT-Systemen zum Umgang mit Hotspots und Ungleichgewichten der Anfragelast nicht ausreicht. Die zwecks Fehlertoleranz redundant vorhandenen Routen im Netzwerk können zusätzlich zur Anfragelastbalancierung verwendet werden. Eine Gewichtsfunktion, die sowohl die Weiterleitungslast als auch die Anfragebearbeitungslast der Knoten berücksichtigt, dient dabei zur Auswahl der am wenigsten ausgelasteten Route. Es ist offen, ob sich die beschriebenen Ansätze auf datenintensive Szenarien anwenden lassen.

Der SD-Rtree [du Mouza et al. 2007] ist eine skalierbare verteilte Datenstruktur, die insbesondere für Punkt- und Bereichsanfragen auf große Mengen räumlicher Daten ausgelegt ist. Basierend auf dem R-Baum wird im Falle einer Überlast dem Netzwerk ein neuer Server hinzugefügt und die betroffenen Daten werden gleichmäßig verteilt. Anwender interagieren mit dem SD-Rtree über eine Sicht, die durch Verteilungsoperationen möglicherweise veraltet ist, aber durch spezielle Nachrichten inkrementell angepasst wird. Replikationsaspekte sind in dieser Datenstruktur nicht berücksichtigt.

3 HiSbase-Architektur

Abbildung 1 zeigt die wesentlichen Aspekte der HiSbase Architektur aus Anwendersicht. Verteilte Archive speisen ihre Daten über einen ihnen bekannten, möglichst nahegelegenen HiSbase-Knoten in das gemeinsame Netzwerk und jeder Knoten bietet die Möglichkeit Anfragen zu stellen. Im Folgenden nehmen wir an, dass die verwendeten Daten entweder bereits vorverarbeitet und ihre Schemata bekannt sind oder zumindest über ein gemeinsames Teilschema verfügen. Orthogonale Ansätze wie Datenfusion [Naumann et al. 2006] oder Schema-Matching [Rahm, Bernstein 2001] zur Aufbereitung wissenschaftlicher Daten können hierfür im Vorfeld eingebracht werden. Die Daten werden im Netzwerk so verteilt, dass logisch verwandte Daten (dargestellt durch die geometrische Form) dem gleichen Knoten zugeordnet werden, auch wenn sie ur-

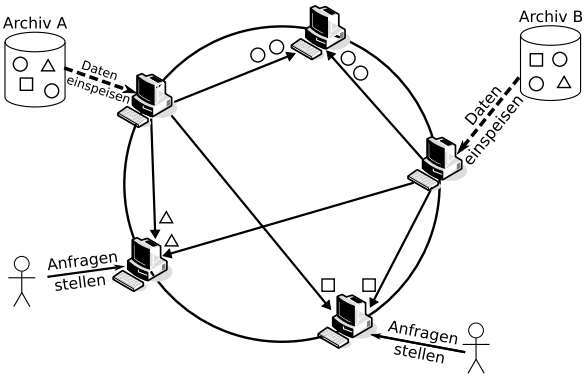


Abbildung 1: Die HiSbase-Architektur

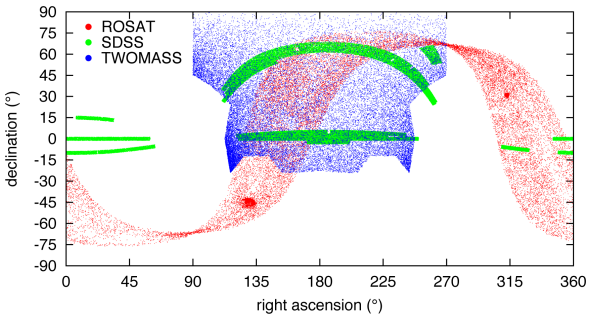


Abbildung 2: Datenausschnitte aus den drei astrophysikalischen Katalogen ROSAT, SDSS und TWOMASS

sprünglich von unterschiedlichen Datenquellen stammen. Damit kann die Korrelation dieser Daten direkt auf einem Knoten erfolgen.

Die DHT-Struktur *Pastry* [Rowstron, Druschel 2001] verwaltet die HiSbase-Stationen und wickelt den Nachrichtenaustausch im Overlay-Netzwerk ab. Wie in Chord [Stoica et al. 2001] verteilt Pastry die Daten gleichmäßig auf einen eindimensionalen, ringförmigen Schlüsselraum. Im Vergleich zu anderen DHT-Systemen ([Stoica et al. 2001, Ratnasamy et al. 2001]) optimiert Pastry das Routing. In den ersten Phasen des Routings senden Stationen bevorzugt Nachrichten an physische Nachbarn und beschleunigen damit die Kommunikation über das Overlay-Netzwerk.

3.1 Mehrdimensionale Daten und Schieflage der Datenverteilung

In vielen Bereichen der Wissenschaft werden mehrdimensionale Daten verwendet, z. B. in der Klimaforschung, in der Medizin und insbesondere auch in der Astrophysik. Objekte aus Beobachtungsdaten, die eine benachbarte Position im astronomischen Koordinatensystem haben, werden hier als logisch benachbart betrachtet. Gängig ist die Verwendung des sphärischen Koordinatensystems, das die Koordinaten in Rektaszension und Deklination bezogen zum Erdmittelpunkt angibt.

Abbildung 2 zeigt Datenausschnitte dreier astrophysikalischer Projekte, die uns durch unsere Kooperationspartner zur Verfügung gestellt wurden. Hier sieht man deutlich eine Schieflage der Datenverteilung, da sowohl Himmelsregionen existieren, die Daten von allen drei Archiven enthalten, als auch Bereiche, die in unserem Datenausschnitt keine Daten enthalten.

Da manche Bereiche intensiver untersucht wurden oder dichter besiedelt sind, treten derartige Schieflagen in der Datenverteilung häufig auf. Wählt man eine geeignete Verteilungsfunktion und ein entsprechendes Datenpartitionierungsschema, können diese Schieflagen berücksichtigt und ausgeglichen werden.

Im Folgenden illustrieren wir anhand der Abbildung 3 das Vorgehen in HiSbase mit einem kleinen vereinfachten Beispiel. In Abbildung 3(i) sind die Datenobjekte des zweidimensionalen Datenraums als Punkte dargestellt.

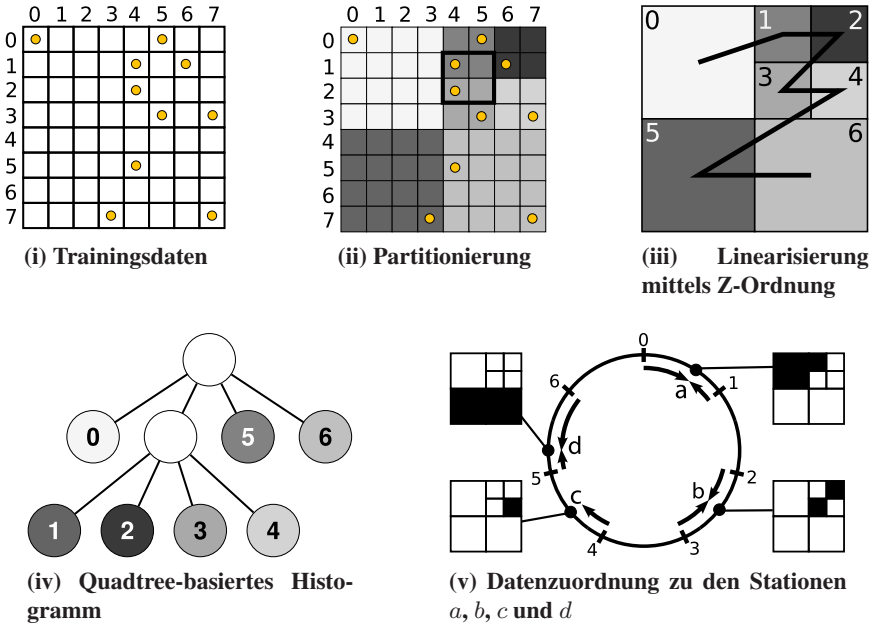
3.2 Datenlastbalancierung durch Histogramme

Das Partitionierungsschema wird in einer Trainingsphase bestimmt. Als Trainingsdaten können hierfür die Gesamtdaten oder eine zufällige, repräsentative Auswahl dienen. Aufgrund ihrer Ähnlichkeit mit Equi-Depth-Histogrammen [Poosala et al. 1996] bezeichen wir die verwendete Verteilungsfunktion und die entsprechende Datenstruktur in HiSbase als *Histogramm*. Dieses Histogramm ist allen Stationen bekannt.

Um eine gleichmäßige Datenverteilung zu realisieren, bieten sich Quadtrees [Finkel, Bentley 1974, Samet 1990] als Datenstruktur für das Histogramm an, da sie sich an die Datenverteilung anpassen und datenreiche Regionen häufig unterteilen. Ein weiterer Vorteil ist die regelmäßige Struktur der Partitionen (Rechtecke im zweidimensionalen Fall), da so die Berechnung relevanter Regionen im Zuge der Anfragebearbeitung einfach realisierbar ist. An anderer Stelle [Scholl et al. 2007d] haben wir eine ausführliche Studie der Trainingsphase durchgeführt, weitere Datenstrukturen untersucht und eine Auswahl möglicher Bewertungskriterien vorgestellt.

Die Histogrammdatenstruktur selbst erstellen wir durch rekursive Dekomposition des Datenraums. Wir beginnen mit einem einblättrigen Quadtree, der den gesamten Datenraum abdeckt und die repräsentativen Datenausschnitte aller Archive enthält. Im Anschluss unterteilen wir jeweils das Blatt mit den meisten Punkten so lange, bis die gewünschte Anzahl an Datenbereichen erzielt ist. Abbildung 3(ii) zeigt die Quadtree-basierte Aufteilung mit sieben Partitionen, die im weiteren auch als *Regionen* bezeichnet werden.

Um bei der Abbildung der Datenbereiche auf den eindimensionalen Datenring der P2P-Infrastruktur die Datenlokalität zu erhalten, verwendet HiSbase *raumfüllende Kurven* wie die Hilbertkurve [Hilbert 1891] oder die Z-Ordnung [Orenstein, Merrett 1984]. Raumfüllende Kurven wurden als Indexstruktur für mehrdimensionale Datenbanken bereits intensiv erforscht [Orenstein, Merrett 1984, Markl, Bayer 2000]. Abbildung 3(iii) zeigt, wie die Bereiche des Datenraums mit Hilfe der Z-Kurve linearisiert werden. Bei Quadtrees ist die Verwendung der Z-Ordnung intuitiv, da sie einer die Blätter gemäß einer Tiefensuche durchnummeriert (siehe Abbildung 3(iv)).



(i) Trainingsdaten **(ii) Partitionierung** **(iii) Linearisierung mittels Z-Ordnung**

(iv) Quadtree-basiertes Histogramm **(v) Datenzuordnung zu den Stationen a, b, c und d**

Abbildung 3: Verarbeitung eines illustrierenden Datenbeispiels mit einem Quadtree-basiertem Histogramm

3.3 Regionenzuordnung

Den Regionen werden Identifikatoren der DHT-Struktur zugeordnet, die gleichmäßig über den Schlüsselraum verteilt sind, während die Stationen zufällig auf diesem angeordnet werden. Die Wahrscheinlichkeit, dass ein Peer einer Region zugeordnet wird, ist somit für alle Regionen gleich, auch wenn sie unterschiedlich große Datenbereiche abdecken. Abbildung 3(v) illustriert eine Zuordnung der sieben Regionen (0–6) zu vier Stationen (a–d). Die Identifikatoren der DHT-Struktur sind nicht dargestellt. Die Stationen kommunizieren über die Regionenidentifikatoren, um immer den verantwortlichen Peer zu erreichen. HiSbase nutzt das Histogramm somit nicht nur als Verteilungsfunktion, sondern auch als Routing-Index.

3.4 Anfragebearbeitung

Jede HiSbase-Station kann regionenbasierte Anfragen stellen bzw. entgegennehmen und berechnet bei der Anfrageanalyse die relevanten Regionen. Die Station, über die eine Anfrage eingespeist wird, koordiniert die Anfragebearbeitung, falls sie selbst eine relevante Region verwaltet. Sonst wird aus den verantwortlichen Peers ein Koordinator für die Anfragebearbeitung bestimmt. Der Koordinator kontaktiert die anderen relevanten Peers und fusioniert deren Ergebnisse. Angenommen, die Beispielanfrage (das dick umrandete Rechteck) aus Abbildung 3(ii) wird durch Station a gestellt. Station a übernimmt nun die Koordination, da sie selbst eine der beiden relevanten Regionen (1 und 3) verwaltet und leitet die Anfrage zudem an Station b weiter. Nachdem die Ergebnisse der eigenen Datenbank mit denen der Station b kombiniert wurden, liefert Station a das Endergebnis zurück.

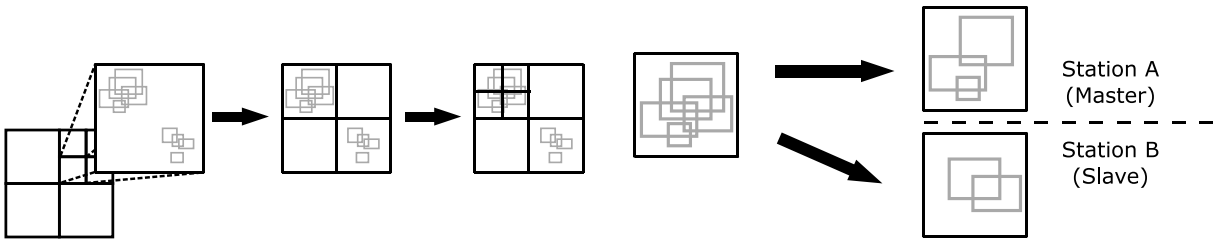
3.5 Einspeisung und Verwaltung der Daten

Alle Stationen beziehen ihre Datenobjekte direkt von den Datenquellen (Archiven), die in HiSbase integriert werden. Die Community-spezifische Verteilungsfunktion bestimmt, auf welchen Peers die Daten (unabhängig vom Ursprungsarchiv) abgelegt werden. Jeder Peer hält die Daten aus allen Archiven, die in die von ihm verwalteten Regionen fallen.

Bei der Datenverwaltung der einzelnen Stationen abstrahiert HiSbase von konkreten Datenbanksystemen. So können sowohl traditionelle als auch hauptspeicherbasierte Datenbanksysteme eingesetzt werden. Die HiSbase-Implementierung verwendet Standard-SQL, das die eingesetzten Systeme unterstützen müssen.

3.6 Stand der Implementierung

Die Implementierung des Forschungsprototypen von HiSbase ist in Java realisiert und verwendet *FreePastry* (http://freepastry.org). FreePastry ist eine OpenSource Pastry-Implementierung, die am Max-Planck-Institut für Softwaresysteme (http://mpi-sws.mpg.de) betreut und weiterententwickelt wird. Die HiSbase Software ist innerhalb des AstroGrid-D Testbetts sowie im PlanetLab-Forschungsnetzwerk (http://planet-lab.org) installiert um dort Messungen durchzuführen. Weitere Implementierungsdetails sind [Scholl et al. 2007a] zu entnehmen. Erste Messungen innerhalb unseres Lehrstuhl-Netzwerkes haben gezeigt, dass wir eine super-lineare Verbesserung des Durchsatzes im Vergleich zu einer einzelnen Station erzielen können [Scholl et al. 2007b].



(i) Datenpartitionierung **(ii) Master-Slave-Verfahren**

Abbildung 4: Behandlung von Anfrage-Hotspots in HiSbase (Anfrageregionen sind grau dargestellt)

4 Lastbalancierung im HiSbase-Netzwerk

Die Lastbalancierung durch Partitionierung, Replikation und Parallelisierung ist ein zentraler Aspekt der HiSbase-Architektur. Unseren Ansatz zur *Datenlastbalancierung* haben wir im vorangegangen Abschnitt genauer beschrieben.

Sind die Daten eines astrophysikalischen Projektes einmal publiziert, werden diese nicht mehr verändert. Dies ist schon aus Gründen der Reproduzierbarkeit wissenschaftlicher Experimente, die auf diesen Daten beruhen, notwendig. Deshalb werden für die einzelnen Kataloge (meist in einem jährlichen Zyklus) neue Versionen parallel zu den bisherigen Daten angeboten. Zusätzliche Daten (neue Kataloge bzw. neue Versionen) können jederzeit in HiSbase integriert werden, sobald alle Stationen deren Schemainformation erhalten haben.

Allerdings ist dann die Datenlastbalancierung möglicherweise nicht mehr optimal. Beim Einfügen neuer Daten bietet sich es deshalb an zu überprüfen, ob das Histogramm – und damit auch die Datenverteilung – weiter verbessert werden kann.

Empirische Untersuchungen [Singh et al. 2006] zeigen, dass es nicht nur Schieflagen in der Datenverteilung gibt, sondern auch *Anfrage-Hotspots* auftreten können, d. h. die Anfragen sind nicht uniform verteilt. Die Entdeckung eines interessanten Phänomens kann beispielsweise dazu führen, dass sich sehr viele Forscher für diese Himmelsregion(en) interessieren. Dadurch würde die Anfrageintensität in einer oder sogar mehreren Regionen ansteigen und eine Überlastsituation bei den verantwortlichen Stationen hervorrufen. Neben diesen kurzfristigen Hotspots können natürlich auch längerfristig zu bereits existierenden Anfrageherden neue – möglicherweise stärkere – Hotspots entstehen.

P2P-Datenmangementsysteme, die auf verteilten Baumstrukturen basieren (z. B. [du Mouza et al. 2007]), stoßen bei der Behandlung von Anfrage-Hotspots an ihre Grenzen, da sie Daten nur partitionieren und nicht replizieren. Abbildung 4(i) illustriert ein derartiges Szenario an einem Beispiel mit zwei Anfrage-Hotspots (Anfrageregionen sind grau dargestellt). Ein dynamisches P2P-Netzwerk könnte zwar mit der ersten Unterteilung die beiden Hotspots trennen; bei einer weiteren Unterteilung müsste jedoch der Ansatz unter Umständen eine größere Zerstückelung der Bereichsanfragen und damit auch höhere Kommunikationskosten in Kauf nehmen, da nun Anfragen mehrere Regionen abdecken. In HiSbase können wir diese Zerstückelung umgehen, indem wir, wie in Abbildung 4(ii) angedeutet, populäre Datenregionen mit Hilfe unseres Master-Slave-Verfahrens (siehe Abschnitt 4.2) replizieren, die Anfragen des Hotspots auf die replizierten Daten verteilen und damit weiterhin die Anfragelokalität beibehalten.

Im Folgenden gehen wir zuerst auf die langfristigen Balancierungsaspekte ein, bevor wir Ansätze für kurzfristige Überlastsituationen beschreiben.

4.1 Langfristige Lastbalancierung

In Abschnitt 3.2 basierte das Training nur auf repräsentativen Daten, und es wurde immer die datenreichste Partition aufgeteilt. Falls bereits Anfragehistorien von den unterschiedlichen Datenquellen vorliegen, ist es vorteilhaft diese ebenfalls während der Trainingsphase zu verwenden. Hierbei gilt es nun die beiden widerstrebenden Ziele Datenvolumenbalancierung und Anfragelokalität gegeneinander abzuwägen.

Anfragehistorien in der Trainingsphase. Eine einfache und sehr effiziente Möglichkeit ist es, bei der Partitionierung diejenigen Regionen bevorzugt aufzuteilen, die sowohl viele Datenobjekte enthalten als auch viele Anfragen verarbeiten müssen. Gibt es in einem Bereich des Datenraums mehrere kleinere Hotspots, werden diese auf mehrere Regionen verteilt und somit die Last auf der einzelnen Station reduziert. Dies entspricht der Vorgehensweise in Abbildung 4(i).

Bei einer sehr großen Anfragenkonzentration wird dieser Ansatz aber ab einer bestimmten Regionengröße kontraproduktiv, da durch weitere Unterteilungen die Regionen zu klein werden und damit die Anfragelokalität verloren gehen kann. Deshalb verfolgen wir in HiSbase einen erweiterten Ansatz, der, sobald ein Großteil der Anfragen mehr als einen bestimmten Prozentsatz der Regionenfläche abdeckt, diese Regionen nicht mehr weiter unterteilt und stattdessen wie unten beschrieben repliziert (Abbildung 4(ii)).

Monitoringdaten bezüglich der Anfragen und der Auslastung der einzelnen Stationen, die im laufenden Betrieb gesammelt werden, können auch bei der Fortschreibung des Histogramms genutzt werden. Diese sind nicht nur für das Erkennen von kurzfristigen Anfrage-Hotspots nützlich, sondern auch für die Trainingsphase zur Anpassung des Partitionierungsschemas an neue Gegebenheiten.

Reorganisation des Histogramms. In HiSbase durchläuft jedes Histogramm die drei Entwicklungsstufen, die in Abbildung 5 dargestellt sind. Gemäß eines *neuen* Histogramms werden neue Datenpartitionen aufgebaut und im Netzwerk verteilt

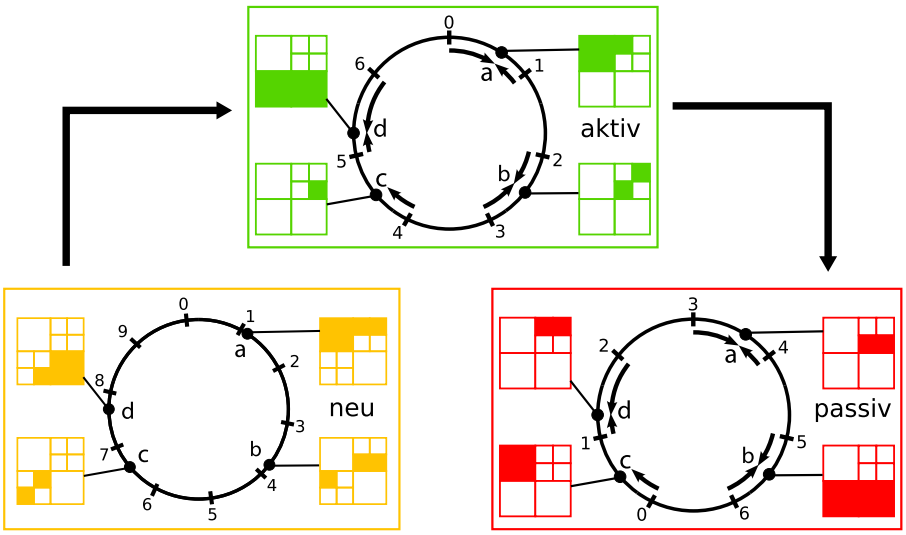


Abbildung 5: Entwicklungsstufen eines Histogramms innerhalb HiSbase

(siehe Abschnitt 3.5). Ist die Datenverteilung abgeschlossen, werden die Daten und das Histogramm *aktiviert* und HiSbase verwendet sie für die Anfragebearbeitung. Zudem hält sich jede HiSbase-Station ein *passives* Histogramm. Wir nutzen das passive Histogramm primär, um eine erhöhte Datenverfügbarkeit zu gewährleisten, und können es darüberhinaus für die Anfragelastbalancierung verwenden.

Für die Bestimmung des passiven Histogramms und dessen Daten sehen wir zwei zentrale Designoptionen: 1) das »veraltete« aktive Histogramm wird zusammen mit den entsprechenden Daten weiterverwendet oder 2) es wird ein weiteres (leicht modifiziertes) Histogramm erstellt. Für letzteres müssen die Daten erneut verteilt werden. Abhängig von der Ursache für die Fortschreibung des Histogramms – seien es nun eine neue Datenquelle oder zusätzliche Anfrage-Hotspots – wird sich die Wahl einer der beiden Alternativen auf das System entsprechend auswirken. Betrachten wir zunächst die Variante, wenn das aktive Histogramm als passives Histogramm weiterverwendet wird.

Wurde eine neue Datenquelle zusammen mit dem neuen Histogramm in das HiSbase-Netzwerk integriert, sind die Daten in dem veralteten Histogramm nicht verfügbar. Somit könnte das passive Histogramm in den aktualisierten Abschnitten nur unvollständige Antworten liefern. Wurden die neuen Daten bereits in das veraltete Histogramm eingefügt, ist zwar die Datenlastbalancierung gemäß diesem Histogramm nicht mehr optimal, (da sie ja ausschlaggebend für die Fortschreibung des Histogramms war) zumindest sind aber alle Daten über das veraltete Histogramm zugreifbar. Verursachten neue Anfrage-Hotspots eine Weiterentwicklung des Histogramms, balanciert das veraltete Histogramm zwar wieder nicht die Anfragebearbeitung optimal, es beinhaltet aber alle notwendigen Daten.

Wenn wir das neue Histogramm ebenfalls für die passive Kopie verwenden, wird die selbe Histogrammstruktur zweimal verwendet und die Daten werden doppelt verteilt. Durch die Verschiebung des Ursprungs der Linearisierungsfunktion (wie in Abbildung 5) oder die Anwendung einer anderen raumfüllenden Kurve kann man eine doppelte Datenverfügbarkeit erzielen. Diese Variante ist zum Beispiel auch für die erste Initialisierung einer HiSbase-Instanz geeignet, da keine zusätzlichen Daten oder Statistiken zur Verfügung stehen werden. Sind aktives und passives Histogramm (sowie deren Daten) identisch, können beide Versionen direkt zur Lastbalancierung genutzt werden, indem die Anfragen anhand der Lastinformationen oder im Round-Robin-Verfahren mit beiden Histogrammen verarbeitet werden.

Bei der Datenverteilung zur Fortschreibung des Partitionierungshistogramms bieten sich ebenfalls zwei Alternativen, wobei die eine Variante die zeitige Datenredundanz und die andere eine frühzeitige Anfragebearbeitung unterstützt.

Die erste Option verteilt die Daten bereits doppelt, einmal gemäß der neuen Histogrammstruktur und einmal gemäß einer »rotierten« Variante. Dadurch wird die gleichzeitige Ersetzung von aktivem und passivem Histogramm vorbereitet und bei erfolgreichem Abschluss vollzogen. Dies kann eine hohe Netzwerkbelastung für die einspeisenden Datenarchive zur Folge haben, da die Daten über die gleichen Netzwerkverbindungen doppelt verteilt werden.

Im Vergleich dazu gibt es die Möglichkeit, zuerst die Daten einmal gemäß des neuen Histogramms zu verteilen. Ist dies erfolgreich abgeschlossen, kann diese neue Kopie bereits zur Anfrageverarbeitung genutzt werden. Die neuen Daten stehen somit früher zur Verarbeitung bereit, wenn auch ohne zusätzliche Replikation. Zur erneuten Replikation der Daten kann die erste, bereits verteilte Kopie genutzt werden, indem jede Station die Daten der ihr zugeordneten neuen Partitionen gemäß dem »rotierten Histogramm« einspeist.

4.2 Kurzfristige Lastbalancierung

Für kurzfristige Anfrage-Überlastsituationen verwenden wir einen Master-Slave-Ansatz, bei dem überlastete Stationen einen Teil ihrer Daten auf Slave-Knoten replizieren. Die Zuordnung von »Sklaven« zu Master-Knoten

wird mit Hilfe einer Multicast-Gruppe arrangiert. Auch für Multicast-Kommunikation gibt es, etwa mit Scribe [Rowstron, Druschel 2001], P2P-basierte Implementierungen, so dass HiSbase auch für diese Komponente nicht auf zentrale Rechner angewiesen ist und sie effizient realisieren kann. In diesem Zusammenhang sind auch die Statistiken, die zur Fortschreibung der Histogramme gesammelt werden, nützlich, um zu stark beanspruchte Stationen zu identifizieren.

Ist eine Station überlastet, gilt es eine geeignete Auswahl an Daten für die Replikation zu bestimmen, eine notwendige Anzahl an Replikaten zu finden und geeignete Stationen für die Verwaltung der Replikate auszumachen.

Für diese drei Kriterien benötigen wir sowohl Lastinformationen als auch Informationen über die Ausfallhäufigkeit einer Station. Im Kontext von Community-Netzwerken wie HiSbase ist vorerst die Lastinformation von primärem Interesse, da wir von einer stabilen Netzwerkinfrastruktur und dedizierten Stationen ausgehen. Dies ist eine sinnvolle Voraussetzung für eine Verteilung von derartig großen Datenmengen. Haben zwei Rechner das gleiche Lastprofil, d. h. sie sind auf Grund ihrer Lastsituation beide Kandidaten für eine Replikation, sollte man zuerst den Rechner replizieren, der eine höhere Ausfallquote hat.

Die Anzahl der Replikate kann proportional zu der Überlast des Peers im Verhältnis zu der Last seiner Nachbarn gewählt werden. Hat ein Peer dreimal soviel Last wie seine Nachbarn, sollten seine Daten dreimal repliziert werden, um so das Lastniveau der einzelnen Rechner wieder aneinander anzugleichen.

Die Auswahl des Rechners, der ein Replikat verwalten soll, kann auf Grund mehrerer Kriterien erfolgen. Im allgemeinen Fall können entweder »logische« Nachbarn (Stationen, die in dem Overlay-Netzwerk nebeneinander liegen) oder »physikalische« Nachbarn (Stationen, die im tatsächlichen Netzwerk benachbart sind) verwendet werden. Hierbei müssen wir abwägen zwischen einer zeitnahen Replikation, die eine hohe Bandbreite benötigt (die physikalische Nachbarn besser leisten können), oder der Beibehaltung der Anfragelokalität (um Anfragen, die mehrere Histogrammregionen überlasten, komplett auf andere Rechner zu verschieben), die durch die Replikation auf logischen Nachbarn erzielt wird.

Bei unserem Master-Slave-Ansatz verwenden wir Knoten, die physikalisch benachbart sind, da die Replikation zwischen diesen Knoten deutlich schneller durchgeführt werden kann, anstatt die Daten erst über ein Wide-Area-Netzwerk zu verteilen. Betreffen Anfragen benachbarte Regionen, die auf mehreren Peers liegen, reduziert die Verwendung von physikalischen Nachbarn auch den Kommunikationsaufwand.

5 Zusammenfassung und Ausblick

Der vorliegende Beitrag beschreibt HiSbase, eine Architektur für datenintensive Anwendungen in e-Science-Community-Grids. HiSbase verbindet etablierte Konzepte aus der Datenbankforschung mit P2P-Technologien innerhalb des Grid-Computing, um eine durchsatzoptimierte Datenhaltung zu realisieren. Anwendungsspezifische Dateneigenschaften und Anfragemuster werden berücksichtigt, indem HiSbase eine Histogrammdatenstruktur aufbaut, die zudem Schieflagen in der Datenverteilung ausgleicht und somit die Daten gleichmäßig verteilt. Das Histogramm dient ferner als Routing-Index, um die Flexibilität der Architektur zu erhöhen.

Wir sind auf wichtige Aspekte der Lastbalancierung eingegangen, insbesondere auch die Behandlung von Anfrage-Hotspots. Wir haben mehrere Designalternativen aufgezeigt, die derartige Schieflagen sowohl langfristig als auch kurzfristig reduzieren. Langfristige Anfrage-Hotspots können während der Erstellung des Partitionierungshistogramms sowie bei dessen Fortschreibung berücksichtigt werden. Um kurzfristig auf eine neue Anfragelast reagieren zu können, verwendet HiSbase eine Master-Slave-Hierarchie, bei der durch Replikation die Last von einer überlasteten Station auf mehrere unterlastete Stationen verteilt wird.

Erste Messungen haben gezeigt, dass wir durch eine erhöhte Parallelität und eine hohe Cache-Lokalität auf den einzelnen Knoten eine super-lineare Verbesserung des Durchsatzes im Vergleich zu einer einzelnen Station erzielen können. Momentan realisieren wir die unterschiedlichen Ansätze zur Lastbalancierung, um sie dann im AstroGrid-D-Testbett sowie mit Hilfe einer Simulationsumgebung miteinander zu vergleichen. Weiterführende Arbeiten umfassen sowohl Aspekte wie die verteilte Ausführung von Data-Mining-Anwendungen über die HiSbase-Architektur als auch die Anbindung an Schnittstellen, die durch die *International Virtual Observatory Alliance (IVOA)* standardisiert werden. Im Vordergrund steht hierbei die *Astronomical Data Query Language (ADQL)*, die auf SQL basiert und eine Anfragesprache für astronomische Datenarchive definiert.

Danksagung

Diese Arbeiten sind Teil des AstroGrid-D-Projekts in der D-Grid-Initiative und werden durch das Bundesministerium für Bildung und Forschung (BMBF) unter Vertrag 01AK804F und durch Microsoft Research Cambridge (MSRC) unter Vertrag 2005-041 gefördert.

Literatur

[Carlson et al. 2007] Carlson, A., Böhringer, H., Scholl, T., Voges, W. (2007). Finding Galaxy Clusters using Grid Computing Technology. In *Proc. of the IEEE Intl. Conf. on e-Science and Grid Computing (demo)*, Bangalore, India.

[Crainiceanu et al. 2007] Crainiceanu, A., Linga, P., Machanavajjhala, A., Gehrke, J., Shanmugasundaram, J. (2007). P-Ring: An Efficient and Robust P2P Range Index Structure. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data*, pages 223–234, Beijing, China.

[Datta et al. 2007] Datta, A., Schmidt, R., Aberer, K. (2007). Query-load balancing in structured overlays. In *Proc. of the IEEE/ACM Int. Symposium on Cluster Computing and the Grid*, pages 453–460, Rio de Janeiro, Brazil.

[du Mouza et al. 2007] du Mouza, C., Litwin, W., Rigaux, P. (2007). SD-Rtree: A Scalable Distributed Rtree. In *Proc. of the Intl. Conf. on Data Engineering*, pages 296–305, Istanbul, Turkey.

[Enke et al. 2007] Enke, H., Steinmetz, M., Radke, T., Reiser, A., Röblitz, T., Högqvist, M. (2007). AstroGrid-D: Enhancing Astronomic Science with Grid Technology. In *Proc. of the German e-Science Conference*, Baden-Baden, Germany.

[Finkel, Bentley 1974] Finkel, R. A., Bentley, J. L. (1974). Quad Trees: A Data Structure for Retrieval on Composite Keys. *Acta Informatica*, 4:1–9.

[Hilbert 1891] Hilbert, D. (1891). Über die stetige Abbildung einer Linie auf ein Flächenstück. *Math. Ann.*, 38:459–460.

[Klan et al. 2008] Klan, D., Sattler, K.-U., Hose, K., Karnstedt, M. (2008). Decentralized Managing of Replication Objects in Massively Distributed Systems. In *Proc. of the Intl. Workshop on Data Management in P2P Systems*, Nantes, France.

[Kuntschke et al. 2006] Kuntschke, R., Scholl, T., Huber, S., Kemper, A., Reiser, A., Adorf, H.-M., Lemson, G., Voges, W. (2006). Grid-based Data Stream Processing in e-Science. In *Proc. of the IEEE Intl. Conf. on e-Science and Grid Computing*, page 30, Amsterdam, The Netherlands.

[Kuntschke et al. 2004] Kuntschke, R., Stegmaier, B., Häuslschmid, F., Reiser, A., Kemper, A., Adorf, H.-M., Enke, H., Lemson, G., Voges, W. (2004). Datenstrom-Management für e-Science mit StreamGlobe. *Datenbank-Spektrum*, 4(11):14–22.

[Markl, Bayer 2000] Markl, V., Bayer, R. (2000). Processing Relational OLAP Queries with UB-Trees and Multidimensional Hierarchical Clustering. In *Proc. of the Intl. Workshop on Design and Management of Data Warehouses*, page 1, Stockholm, Sweden.

[Naumann et al. 2006] Naumann, F., Bilke, A., Bleiholder, J., Weis, M. (2006). Data Fusion in Three Steps: Resolving Schema, Tuple, and Value Inconsistencies. *IEEE Data Engineering Bulletin*, 29(2):21–31.

[Orenstein, Merrett 1984] Orenstein, J., Merrett, T. (1984). A Class of Data Structures for Associative Searching. In *Proc. of the ACM SIGACT-SIGMOD Symp. on Principles of Database Sys.*, pages 181–190, Waterloo, Ontario, Canada.

[Pitoura et al. 2006] Pitoura, T., Ntarmos, N., Triantafillou, P. (2006). Replication, Load Balancing, and Efficient Range Query Processing in DHT Data Networks. In *Proc. of the Intl. Conf. on Extending Database Technology*, pages 131–148, Munich, Germany.

[Poosala et al. 1996] Poosala, V., Ioannidis, Y. E., Haas, P. J., Shekita, E. J. (1996). Improved Histograms for Selectivity Estimation of Range Predicates. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data*, pages 294–305, Montreal, Quebec, Canada.

[Rahm, Bernstein 2001] Rahm, E., Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350.

[Ratnasamy et al. 2001] Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S. (2001). A Scalable Content-Addressable Network. In *Proc. of the ACM SIGCOMM Intl. Conf. on Data Communication*, pages 161–172.

[Rowstron, Druschel 2001] Rowstron, A. I. T., Druschel, P. (2001). Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems. In *Proc. of the IFIP/ACM Intl. Conf. on Distributed Systems Platforms (Middleware)*, pages 329–350, Heidelberg, Germany.

[Samet 1990] Samet, H. (1990). *The Design and Analysis of Spatial Data Structures*. Addison Wesley.

[Scholl et al. 2007a] Scholl, T., Bauer, B., Gufler, B., Kuntschke, R., Weber, D., Reiser, A., Kemper, A. (2007a). HiSbase: Histogram-based P2P Main Memory Data Management. In *Proc. of the Intl. Conf. on Very Large Data Bases (demo)*, pages 1394–1397, Vienna, Austria.

[Scholl et al. 2007b] Scholl, T., Bauer, B., Gufler, B., Kuntschke, R., Weber, D., Reiser, A., Kemper, A. (2007b). Histogram-based P2P Main Memory Database for Locality-Aware Data (Poster). http://www-db.in.tum.de/research/publications/conferences/HiSbase-DemoPoster-VLDB2007.pdf.

[Scholl et al. 2007c] Scholl, T., Bauer, B., Kuntschke, R., Weber, D., Reiser, A., Kemper, A. (2007c). HiSbase: Informationsfusion in P2P Netzwerken. In *Proc. of the GI Conference on Database Systems for Business, Technology, and Web (demo)*, pages 602–605, Aachen, Germany.

[Scholl et al. 2007d] Scholl, T., Kuntschke, R., Reiser, A., Kemper, A. (2007d). Community Training: Partitioning Schemes in Good Shape for Federated Data Grids. In *Proc. of the IEEE Intl. Conf. on e-Science and Grid Computing*, pages 195–203, Bangalore, India.

[Schücker et al. 2004] Schücker, P., Böhringer, H., Voges, W. (2004). Detection of X-ray Clusters of Galaxies by Matching RASS Photons and SDSS Galaxies within GAVO. *Astronomy & Astrophysics*, 420:61–74.

[Singh et al. 2006] Singh, V., Gray, J., Thakar, A., Szalay, A., Raddick, J., Boroski, B., Lebedeva, S., Yanny, B. (2006). SkyServer Traffic Report – The First Five Years. Technical Report MS-TR-2006-190, Microsoft Research, Microsoft Cooperation, Redmond, WA, USA.

[Stoica et al. 2001] Stoica, I., Morris, R., Karger, D. R., Kaashoek, M. F., Balakrishnan, H. (2001). Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proc. of the ACM SIGCOMM Intl. Conf. on Data Communication*, pages 149–160, San Diego, CA, USA.

[Szalay et al. 2002] Szalay, A. S., Gray, J., Thakar, A. R., Kunszt, P. Z., Malik, T., Raddick, J., Stoughton, C., vandenBerg, J. (2002). The SDSS skyserver: public access to the sloan digital sky server data. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data*, pages 570–581, Madison, WI, USA.

[Venugopal et al. 2006] Venugopal, S., Buyya, R., Ramamohanarao, K. (2006). A Taxonomy of Data Grids for Distributed Data Sharing, Management, and Processing. *ACM Computing Surveys*, 38(1):3.

[Zimmermann et al. 2006] Zimmermann, R., Ku, W.-S., Wang, H., Zand, A., Bardet, J.-P. (2006). A Distributed Geotechnical Information Management and Exchange Architecture. *IEEE Internet Computing*, 10(5):26–33.