Datenstrom-Management für e-Science mit StreamGlobe

R. Kuntschke B. Stegmaier F. Häuslschmid A. Reiser A. Kemper
TU München, Lehrstuhl für Datenbanksysteme
H.-M. Adorf* H. Enke+ G. Lemson* W. Voges*
*Max-Planck-Institut für extraterrestrische Physik +Astrophysikalisches Institut Potsdam

Aktuelle Forschungsaktivitäten im Bereich der Datenstromverarbeitungssysteme untermauern die zunehmende Bedeutung von Datenströmen, etwa im Kontext von Sensornetzwerken und im Bereich e-Science. Mit der zunehmenden weltweiten Vernetzung experimentell arbeitender Wissenschaftler ist die Entwicklung von Datenstrom-Management-Systemen (DSMS) zur Informationsgewinnung aus im Netz veröffentlichten Messdaten zu einer großen Herausforderung geworden. In dieser Arbeit beschreiben wir die adaptive Anfragebearbeitung und Optimierung im Rahmen unseres StreamGlobe Projekts, das sich effiziente Anfragebearbeitung auf Datenströmen in verteilten, heterogenen Umgebungen zum Ziel setzt.

1 Einleitung

In den letzten Jahren haben Peer-to-Peer (P2P-)Netzwerke sowohl in den Medien als auch in der Forschung große Aufmerksamkeit erlangt. Dies ist einerseits zurückzuführen auf den Erfolg von File-Sharing Systemen, wie z. B. Napster und Gnutella, andererseits aber auch auf den Grad an Flexibilität, den diese Netzwerke bieten. Beispielsweise können sie benutzt werden, um Sensornetzwerke (vgl. [Stegmaier et al. 2004]) oder Netze zum Austausch und zur Analyse experimenteller Daten im e-Science Bereich aufzubauen. An derartigen Netzwerken können sich Datenquellen anmelden, um ihre Daten kontinuierlich in Form von Datenströmen an das Netz zu liefern. Neben den Datenlieferanten ist entsprechend auch die Registrierung von Anfragen im Netz möglich, welche die im Netzwerk verfügbaren Datenströme verarbeiten. In der Vergangenheit wurden verschiedene Ansätze zur Auffindung von Informationen, etwa Dokumenten, Dateien usw., in P2P-Netzwerken untersucht und unterschiedliche Topologien für derartige Netzwerke entwickelt. Ein Beispiel dafür sind Super-Peer-Netzwerke [Yang, Garcia-Molina 2003], wie sie auch bei [Löser et al. 2003] zum Einsatz kommen. Im Umgang mit Datenströmen ist jedoch das Auffinden von Peers, welche die gewünschten Informationen liefern, nicht die einzige Aufgabe. Zusätzlich muss ein kontinuierlicher Datenfluss von den Datenquellen zu den jeweiligen Empfängern hergestellt werden. Eine interessante Herausforderung, die sich in dieser hochdynamischen Umgebung ergibt, ist die Entwicklung eines verteilten, selbstorganisierenden DSMS für effiziente Anfragebearbeitung und Verteilung von Datenströmen im Netzwerk. Aufbauend auf den Techniken des Vorgängerprojekts ObjectGlobe [Braumandl et al. 2001], das die verteilte Bearbeitung persistenter Daten im Netzwerk realisierte, verfolgen wir mit *StreamGlobe* dieses Ziel. Andere DSMS, wie etwa TelegraphCQ [Chandrasekaran et al. 2003], Aurora [Cherniack et al. 2003], ONYX [Diao et al. 2004] und Cougar [Yao, Gehrke 2002], konzentrieren sich auf spezielle Aspekte der Anfragebearbeitung, z. B. kontinuierliche Anfragen auf Datenströmen, Lastbalancierung oder Quality-of-Service Management. Unserem Ansatz näher ist PIPES [Krämer, Seeger 2004]. Der hauptsächliche Beitrag von StreamGlobe ist darin zu sehen, dass es Datenströme nicht nur auf effiziente Weise lokalisiert und auswertet, sondern die Anfragebearbeitung adaptiv in das Netzwerk – hin zu den Datenquellen – verlagert, um dadurch den Datenfluss im Netz zu optimieren. In diesem Zusammenhang wird auch die Wiederverwendung von (Teil-)Datenströmen – insbesondere von bereits berechneten Ergebnissen von (Teil-)Anfragen – zur Beantwortung anderer Anfragen möglich, was sowohl das Datenvolumen im Netzwerk als auch die Auslastung der Peers reduziert.

Als Beispielarchitektur für eine mögliche Anwendung von StreamGlobe betrachten wir Abbildung 1. Das dargestellte Netzwerk ist als Super-Peer-Netzwerk organisiert, wobei SP_0 bis SP_3 einen sogenannten Super-Peer-Backbone bilden und fünf möglicherweise mobile Peers P_0 bis P_4 mit dem Backbone verbunden sind. Die Peers P_0 und P_2 registrieren Anfragen im Netzwerk. Die Peers P_1 und P_4 stellen hingegen Sensoren dar, die ihre Daten in Form von XML-Datenströmen an das Netzwerk liefern. Peer P_3 repräsentiert eine Datenbank, die persistent gespeicherte Daten enthält und im Netzwerk zur Verfügung stellt. Wir legen ein konkretes Anwendungsszenario aus dem Bereich der Astronomie zugrunde und sehen P_1 und P_4 als satellitengestützte Teleskope an, welche Sensormessdaten über registrierte Photonen an das Netzwerk liefern. Dabei handelt es sich um die Vision eines möglichen zukünftigen Szenarios, da die Datenverarbeitung und -analyse in der Astronomie heute noch meist auf persistenten Datenbeständen durchgeführt wird. Mit steigendem Datenvolumen wird dies in Zukunft jedoch häufig unpraktikabel werden, weshalb ein Übergang zur direkten Verarbeitung von Messdatenströmen sowohl sinnvoll als auch notwendig erscheint. Auch im Bereich der virtuellen Observatorien, denen in jüngster Zeit in der Astronomie gesteigerte Aufmerksamkeit zuteil wurde, sind derartige Überlegungen aktuell. Eine weitere Anwendung der Datenstromverarbeitung in diesem Bereich stellen sogenannte Roboterteleskope

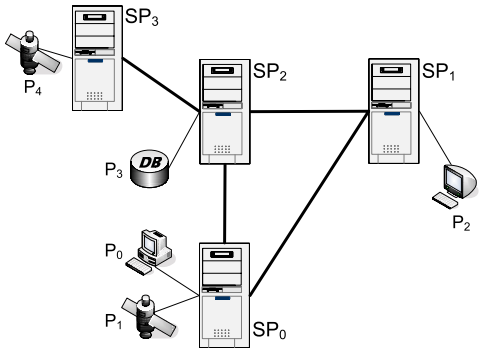

Abbildung 1: Beispielszenario

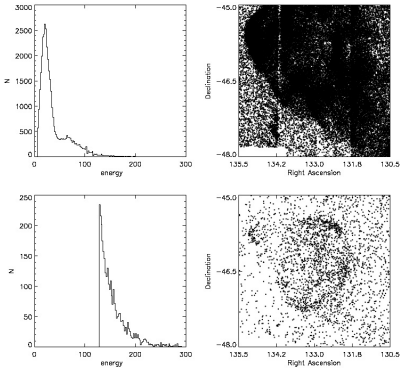

Abbildung 2: Visualisierung der Photonendaten

dar, deren Steuerung in Echtzeit auf bestimmte Ereignisse und Messwerte reagiert, um das Teleskop automatisch auf einen bestimmten Himmelsabschnitt auszurichten.

Wir beginnen mit einem vereinfachten Beispiel, das wir in Kapitel 3 vertiefen werden. Das Anwendungsszenario stellt die Vision eines Einsatzes von Datenstromverarbeitungstechniken im German Astrophysical Virtual Observatory (GAVO) [GAVO 2004] des Max-Planck-Instituts für extraterrestrische Physik (MPE) in Garching bei München und des Astrophysikalischen Instituts Potsdam (AIP) dar und basiert auf realen Daten des von dem Satelliten ROSAT durchgeführten ROSAT All-Sky Survey (RASS) [Voges et al. 1999]. Betrachten wir zunächst den Datenstrom, der von dem Satelliten P_4 geliefert wird. Der Photonendetektor des Satelliten erzeugt für jedes erkannte Photon eine Reihe von Messwerten und speist diese in das Netzwerk ein. Eine solche Messung besteht aus zeitlichen, räumlichen und spektralen Aspekten und umfasst die Position am Himmel, an der das Photon auftrat, beschrieben durch Rektaszension (ra) und Deklination (dec) in Grad, sowie Angaben über die Abweichung der bestimmten Position des Photons von dessen tatsächlicher Position (pos_error), Informationen über den Detektorpuls bei der Registrierung des Photons (phc), die daraus berechnete Energie des gemessenen Photons in keV (en), die Nummer des RASS Feldes (field_id), in dem das Photon am Himmel auftrat (das RASS Feld ist eine künstliche Einteilung des Himmels in 1378 Bereiche von etwa sechs mal sechs Grad Größe), die Zeit, zu der das Photon gemessen wurde, in Sekunden seit Beginn der Messungen (det_time) und die x- und y-Koordinaten des Detektorpixels, an dem das Photon aufgetroffen ist (dx, dy). Aus Gründen der kürzeren Darstellung verwenden wir die folgende vereinfachte DTD, um obigen Datenstrom, welcher als »photons« bezeichnet wird, zu beschreiben, obwohl StreamGlobe XML Schema verwendet.

```
<!ELEMENT photon (ra, dec,
                  pos_error, phc,
                  en, field_id,
                  det_time,
                  dx, dy)>
<!ELEMENT ra (#PCDATA)>
...
```

Alle übrigen Elemente haben analoge DTD Einträge. Seien nun P_0 bzw. P_2 Geräte, die von verschiedenen Gruppen von Astronomen genutzt werden. Die erste Gruppe will einen Überblick über alle Photonen im Bereich des *Vela Supernova Überrests* gewinnen. An P_0 sollen deshalb für jedes gemessene Photon dessen Koordinaten, Energie und Messzeitpunkt ausgegeben werden. Um den Energiewert gegebenenfalls überprüfen und neu berechnen zu können, soll auch der jeweils gemessene Detektorpuls zurückgeliefert werden. Dazu wird an P_0 die folgende Anfrage in XQuery registriert, in der die Funktion »stream(...)« eine XQuery-Erweiterung darstellt, die analog zur Funktion »doc(...)« den Strom »photons« als Eingabe für die Anfrage spezifiziert.

```
for $p in stream("photons")/photon
where $p/ra > 120.0
  and $p/ra < 138.0
  and $p/dec > -49.0
  and $p/dec < -40.0
return
  <vela>
    {$p/ra} {$p/dec} {$p/phc}
    {$p/en} {$p/det_time}
  </vela>
```

Eine zweite Gruppe interessiert sich hingegen für hochenergetische Photonen in dem kleineren Bereich des *RXJ0852.0-4622 Supernova Überrests* [Aschenbach 1998]. Auf P_2 soll deshalb nur dann eine Meldung erscheinen, wenn innerhalb dieses Bereichs am Himmel ein Photon entdeckt wird, das einen außergewöhnlich hohen Energiewert von mehr als 1.3 keV besitzt. Dabei sind hier auch die Auftreffpunkte der gemessenen Photonen auf dem Detektor von Bedeutung, um daraus gegebenenfalls fehlerhafte Werte für die Rektaszension und die Deklination neu berechnen oder den Betrachtungsbereich nachträglich auf bestimmte Sektoren der Detektorfläche einschränken zu können. Um diese Informationen zu gewinnen, wird an P_2 die folgende Anfrage in XQuery registriert.

```
for $p in stream("photons")/photon
where $p/en > 1.3
  and $p/ra > 130.5
  and $p/ra < 135.5
  and $p/dec > -48.0
  and $p/dec < -45.0
return
  <rxj>
    {$p/ra} {$p/dec} {$p/en}
    {$p/det_time} {$p/dx} {$p/dy}
  </rxj>
```

Der Effekt der Selektion von Photonen mit Energien größer als 1.3 keV wird in Abbildung 2 deutlich. Hier ist die Zahl der gemessenen Photonen in Abhängigkeit ihrer Energie und die Verteilung der Photonen in dem betrachteten Himmelsabschnitt gezeigt. Die obere Bildhälfte stellt die Situation ohne Selektion auf die Energie dar. In der Verteilung der Photonen lässt sich keine neue Struktur erkennen. Führt man die Selektion wie in der unteren Bildhälfte gezeigt durch, so wird der Umriss des auf diese Weise

entdeckten RXJ0852.0-4622 Supernova Überrests sichtbar.

StreamGlobe behandelt das skizzierte Szenario wie folgt. Die Daten von P_4 werden an SP_3 geliefert und dort möglichst früh gefiltert, so dass nur die Elemente im Strom verbleiben, die von beiden Anfragen benötigt werden. Wie aus den Anfragen leicht zu ersehen ist, enthält der selektierte Himmelsbereich der Anfrage an P_0 den in der Anfrage an P_2 selektierten Bereich vollständig, da sich RXJ0852.0-4622 in dem Gebiet von Vela befindet. Deshalb können alle Datenelemente, deren Koordinaten außerhalb des in der Anfrage an P_0 selektierten Bereichs liegen, bereits an SP_3 ausgefiltert werden. Weiterhin können alle Elemente entfernt werden, auf die keine der beiden Anfragen projiziert. Weitere Filterungen, z. B. basierend auf einer Selektion bezüglich der Energie, sind zu diesem Zeitpunkt noch nicht möglich, da die Anfrage an P_0 keinerlei Einschränkungen bezüglich des Energiewerts eines Photons trifft. Der an SP_3 gebildete Datenstrom enthält damit die kombinierten Informationen zur Beantwortung der beiden Anfragen an P_0 und P_2. Er wird nun weiter zu SP_2 geleitet, wo ihn das System repliziert und beide entstehenden Ströme entsprechend filtert, so dass sie jeweils nur noch die von P_0 bzw. P_2 benötigten Daten enthalten. Dazu führt das System für den an P_2 zu liefernden Datenstrom eine Selektion auf Energiewerte größer als 1.3 keV durch und schränkt den Datenstrom durch eine im Vergleich zur Selektion an SP_3 restriktivere Selektion (Nachfilterung) auf den in der Anfrage an P_2 selektierten Himmelsbereich ein. Weiterhin muss auf beiden Datenströmen noch jeweils eine Projektion auf die in der »return«-Anweisung der jeweiligen Anfrage angegebenen Elemente erfolgen. Die resultierenden Ströme werden schließlich über SP_0 bzw. SP_1 zu P_0 bzw. P_2 geleitet. Durch dieses Routing der Datenströme wird der Netzwerkverkehr im Vergleich zu herkömmlichen Systemen erheblich verringert, da redundante Übertragungen vermieden werden. Der Ansatz der Verlagerung der Anfragebearbeitung von den Klienten ins Netzwerk zur Reduzierung der Netzwerklast ist ein Hauptmerkmal von StreamGlobe und unterscheidet unser System von bisherigen Arbeiten, z. B. im Multicast-Bereich.

Der verbleibende Teil des Artikels gliedert sich wie folgt. In Kapitel 2 beschreiben wir kurz die zugrunde liegende Systemarchitektur von StreamGlobe, bevor wir in Kapitel 3 ausführlicher auf Anfragebearbeitung und Optimierung in unserem System eingehen. Außerdem greifen wir dort das Anwendungsbeispiel aus der Einleitung weiterführend auf. Kapitel 4 schließt unsere Ausführungen mit einer Zusammenfassung, einem kurzen Bericht über den Status der Implementierung unseres StreamGlobe Prototyps und einem Ausblick auf zukünftige Forschungsvorhaben.

2 StreamGlobe-Architektur

StreamGlobe stellt eine Föderation von Peers dar, welche je nach ihren Fähigkeiten Aufgaben im Netzwerk übernehmen. Abbildung 3 zeigt die Architektur von StreamGlobe auf den Peers. Gestrichelt gekennzeichnete Komponenten sind in Abhängigkeit der Fähigkeiten der Peers in unterschiedlichem Umfang vorhanden, z. B. besitzen einfache Peers oft nur rudimentäre Möglichkeiten zur Anfragebearbeitung und keine Optimierungskomponente. Im Nachfolgenden geben wir einen kurzen Einblick in die Architektur von StreamGlobe. Nähere Details finden sich in [Stegmaier et al. 2004].

StreamGlobe setzt auf der *Open Grid Services Architecture (OGSA)* und deren Referenzimplementierung, dem *Globus Toolkit*, auf. Die in Abbildung 3 gezeigten Schichten werden als kooperierende Dienste in dieser Architektur realisiert. Grid Computing wurde bereits in GridDB [Liu, Franklin 2004] im Zusammenhang mit Datenbanken und der Analyse wissenschaftlicher Daten untersucht und eingesetzt. Auch hier findet sich u. a. ein Anwendungsbeispiel aus der Astronomie. Allerdings arbeitet GridDB im Gegensatz zu StreamGlobe ausschließlich auf persistenten Daten.

OGSA erlaubt die beliebige Kommunikation zwischen Diensten, was im Hinblick auf mobile Peers nicht immer sinnvoll ist – z. B. sollen mobile Sensoren nur über bestimmte Knoten mit dem Netzwerk kommunizieren können. Daher wird eine zusätzliche P2P-Schicht etabliert, welche wie folgt definiert ist. Das Netzwerk besteht aus einer Menge von *Peers*. Jeder Peer besitzt Nachbar-Peers, mit denen er direkt kommunizieren kann. Zum Transfer von Daten zwischen zwei beliebigen Peers muss ein *Transferpfad* von Peers im Netzwerk aufgebaut werden. Konkret könnte das Overlay-Netzwerk beispielsweise durch ein Super-Peer-Netzwerk [Yang, Garcia-Molina 2003] realisiert werden.

Zur Integration verschiedenster Peers – von kleinen, mobilen Geräten mit wenig Rechenleistung bis hin zu leistungsfähigen, stationären Servern – in ein Informationsnetzwerk werden Peers nach ihrer Leistungsfähigkeit klassifiziert. *Thin-Peers* stellen Geräte mit relativ wenig Rechenleistung dar, die im Allgemeinen nicht in der Lage sind, aufwendige Anfragebearbeitung durchzuführen. Dagegen sind *Super-Peers* stationäre Rechner mit hoher Rechenleistung. Diese Super-Peers bilden einen Backbone und übernehmen im Netzwerk Anfragebearbeitungsaufgaben, welche andere (Thin-)Peers nicht ausführen können. Als Metadaten-Verwaltung (MDV) kommt eine Weiterentwicklung der MDV von ObjectGlobe [Keidl et al. 2002] zum Einsatz, welche auf den *Service Data* und *Service Discovery* Mechanismen des Globus Toolkit basiert. Die MDV verwaltet u. a. die Nachbarschaftsbeziehungen und Fähigkeiten der Peers, vorhandene Subskriptionsregeln, Datenströme, sowie Statistiken über Datenströme, wie etwa Größe und Frequenz der enthaltenen Elemente.

Benutzer registrieren *Subskriptionsregeln* zur Informationsgewinnung in XQuery an ihren jeweiligen Peers (Arbeitsstationen). In unserem Kontext sind Subskriptionen echte Anfragen – im Gegensatz zu Ansätzen aus der Literatur, bei welchen lediglich die zu einer Anfrage passenden Dokumente zugestellt werden – und ermöglichen ausdrucksstarke Transformationen der Datenströme, um sie flexibel auf individuelle Bedürfnisse »zuschneiden« zu können. Analog registrieren Datenquellen ihre Datenströme an bestimmten Peers. Eine Datenquelle kann ihre Daten mit zugehörigem XML Schema als individuellen Datenstrom registrieren, welcher damit im StreamGlobe Netzwerk unter einer eindeutigen Kennung zur Verfügung steht. Eine andere Möglichkeit ist die Registrierung der Daten als Teil eines

virtuellen Datenstroms. Hierbei werden die Datenströme aller beteiligten Datenquellen, welche jeweils mit dem Schema das virtuellen Stroms konform sind, gebündelt und unter einer eindeutigen Kennung von einem Peer zur Verfügung gestellt. Die Einspeisung von Daten in das Netzwerk wird von speziellen Operatoren, den *Wrappern*, übernommen, welche auf den verantwortlichen Peers ausgeführt werden.

In StreamGlobe wird eine verteilte, hierarchisch organisierte Optimierung eingesetzt, da aus Effizienzgründen eine zentrale Optimierungskomponente nicht realisierbar ist. Das Netzwerk wird dazu in Segmente partitioniert. In jedem Segment übernimmt ein *Speaker-Peer* die Optimierung und die Koordination mit Nachbarsegmenten. Die Aufgabe der Optimierung in StreamGlobe beinhaltet die Bestimmung der Peers, auf denen (Teil-)Anfragen bzw. Subskriptionen ausgeführt werden, und der dazu nötigen Transferpfade für Datenströme. Dabei werden folgende Ziele verfolgt:

1. Registrierung beliebiger Subskriptionen an den Peers, ungeachtet der Fähigkeiten des jeweiligen Peers.
2. Erzielung eines möglichst guten Datenflusses in Bezug auf das Transfervolumen im Netzwerk, ohne das Netzwerk mit redundanten Übertragungen zu belasten.
3. Gemeinsame Optimierung der Ausführung vieler Subskriptionen (Multi-Query Optimierung).

Die Optimierung ist ein kontinuierlicher Prozess, welcher alle Änderungen, z. B. An- und Abmeldungen von Subskriptionen und Datenquellen, berücksichtigt. Damit wird ein sich selbst optimierendes und rekonfigurierendes Netzwerk angestrebt, welches sich jederzeit in einem effizienten Zustand befindet. Weitere Details zur Optimierung werden im nächsten Kapitel dargelegt.

StreamGlobe zielt auf die Bearbeitung von Datenströmen ab und beinhaltet deshalb *push-basierte* Auswertungsstrategien – im Gegensatz zu traditionellen DBMS, wo Daten von untergeordneten Operatoren *pull-basiert* angefordert werden. Zur Auswertung der (Teil-)Subskriptionen setzen wir neue Optimierungstechniken (FluX [Koch et al. 2004]) für die Bearbeitung von XQuery-Anfragen auf Datenströmen ein. Sie minimieren den Speicherbedarf bei der Anfragebearbeitung und erlauben so die skalierbare Auswertung von Subskriptionsregeln. Bestimmte Subskriptionen können inhärent nicht skalierbar auf Datenströmen ausgeführt werden, z. B. wenn Joins enthalten sind. In solchen Fällen wird unendliches Zwischenspeichern verhindert, indem die Benutzer zur Spezifikation von Fensterbedingungen gezwungen werden. Damit wird die Ausführbarkeit auf unendlichen Datenströmen sichergestellt. Die (z. B. zeitliche) Konsistenz individueller Datenfenster obliegt den ausführenden Operatoren und ist Gegenstand aktueller Forschung.

3 Subskriptionsauswertung

Nachfolgend beschreiben wir unsere Ansätze zur Optimierung des Netzwerkverkehrs und zur effizienten Bearbeitung von Anfragen in StreamGlobe.

3.1 Optimierung

Zunächst behandeln wir die zentralen Vorgehensweisen zur Erreichung der in Abschnitt 2 beschriebenen Optimierungsziele. Das erste Ziel wird durch geeignetes Verschieben der Subskriptionsauswertung in das Netzwerk erreicht. Dazu führt das System jede Subskription ganz oder teilweise an einem oder mehreren geeigneten Peers auf den Transferpfaden von den Datenquellen zum die Subskription registrierenden Peer aus. Ein geeigneter Peer ist dabei ein Peer, der in der Lage ist, die Subskription zu verarbeiten, d. h. ausreichend Rechenleistung besitzt und vom Anfrageoptimierer unter Berücksichtigung von Optimierungszielen, wie z. B. Reduzierung des Netzwerkverkehrs, ausgewählt wurde. Um flexible und mächtige Subskriptionsregeln zu ermöglichen, kommt *mobiler Code* zum Einsatz. Neben der von den Peers bereitgestellten grundlegenden Funktionalität sind Benutzer in der Lage, benutzerdefinierten Code, beispielsweise Prädikate, Aggregationsoperatoren usw., in ihre Subskriptionsregeln aufzunehmen. Dieser wird anschließend an dem Peer, der den entsprechenden Teil der Subskription bearbeitet, instanziiert.

Das zweite Ziel wird durch die Verwendung zweier einander ergänzender Techniken erreicht. Bei der ersten dieser beiden Techniken handelt es sich um das *Filtern* von Datenströmen. Filterung erfolgt entweder durch die Verwendung von Projektionen (strukturelle Filterung) oder von Selektionen (inhaltsbasierte Filterung) oder einer Kombination aus beidem und basiert auf *Filteroperatoren*. Diese Operatoren werden an Peers ausgeführt, die sich auf dem Pfad des Datenstroms und zugleich möglichst nah an der Datenquelle befinden. Dadurch wird die über das Netzwerk zu transportierende Datenmenge reduziert. Die zweite Technik ist die *Ballung (Clustering) von Datenströmen*. Damit wird die Kombination mehrerer ähnlicher oder gleicher Datenströme im Netzwerk zu einem einzigen Strom, der mehrere Empfänger bedient, bezeichnet. Die Ballung von Datenströmen in StreamGlobe funktioniert wie folgt. Im Zuge der Registrierung einer neuen Anfrage analysiert das System die Anfrage und identifiziert ihre Eigenschaften. Diese beinhalten die Identifikatoren der zur Beantwortung der Anfrage benötigten Datenströme, die Operatoren (z. B. Projektionen, Selektionen, Joins usw.) die zur Verarbeitung der Eingabeströme verwendet werden und die von diesen Operatoren benötigten Bedingungen (Projektionsattribute, Selektions- und Joinprädikate usw.). Alle transformierten Datenströme im System werden entsprechend durch die zugehörigen Eigenschaften der jeweils erzeugenden Anfragen repräsentiert. Ursprüngliche Datenströme, die von einer Datenquelle an einem Super-Peer registriert wurden, werden durch einen eindeutigen Identifikator repräsentiert. Der Grund für die Entscheidung, Datenströme anhand ihrer Eigenschaften darzustellen, liegt in einer Erhöhung der Abstraktionsebene im Vergleich zur Schemarepräsentation. Dies erleichtert den Vergleich von Datenströmen und die Suche nach wiederverwendbaren Teilströmen im Netzwerk im Zuge der Optimierung.

Während der eigentlichen Ballung von Datenströmen überprüft der Speaker-Peer des betroffenen Teilnetzes die relevanten Metadaten der in seinem Teilnetz existierenden Datenströme und vergleicht ihre Eigenschaften mit denen der neu registrierten Anfrage. In einem ersten Greedy-Ansatz wählt der Speaker-Peer diejenigen Datenströme

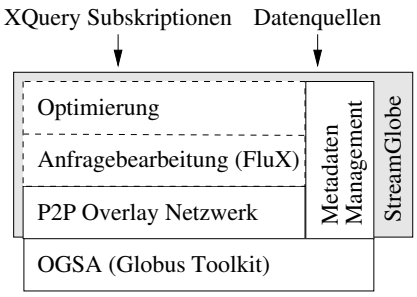

Abbildung 3: Architekturübersicht

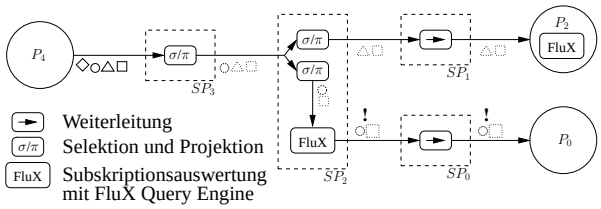

Abbildung 4: Auswertungsplan des Beispielszenarios

als Eingabeströme für die neue Anfrage aus, welche die benötigten Daten zur Beantwortung der Anfrage enthalten, am wenigsten unbenötigte Daten enthalten und über die geringste Zahl von Peers im Netzwerk geleitet werden müssen, um zum Empfänger zu gelangen. Natürlich muss auch die Entscheidung getroffen werden, wo bestimmte Anfrageoperatoren, z. B. Joins, im Netzwerk ausgeführt werden sollen. Dies ist Teil zukünftiger Arbeiten und wird auf einem geeigneten Kostenmodell basieren. Weiterhin werden wir Strategien zur Reorganisation des Netzwerks untersuchen. Damit lassen sich mögliche Effizienzeinbußen aufgrund lokaler Änderungen am Netzwerk bzw. an den Subskriptionen adaptiv korrigieren.

Die Ballung von Datenströmen, wie sie oben beschrieben wurde, trägt auch zur Erfüllung des dritten Ziels einer effektiven Multi-Query Optimierung bei. In jedem Teilnetz analysiert der Speaker-Peer die registrierten Subskriptionen und identifiziert gemeinsame Teilausdrücke. Diese werden im betreffenden Teilnetz einmalig ausgewertet, indem das System eine Subskriptionsregel, die dem jeweiligen Teilausdruck entspricht, an einem geeigneten Peer ausführt. Statt den Teilausdruck für jede der ursprünglichen Subskriptionen individuell auszuwerten, werden die Subskriptionen geeignet umgeschrieben, so dass sie den neu generierten speziellen Datenstrom, der aus der Auswertung der Subskription für den gemeinsamen Teilausdruck stammt, verwenden. Diese Vorgehensweise wurde bereits in dem Beispiel in Kapitel 1 demonstriert. Abgesehen von einer Lastverringerung auf den betroffenen Peers kann dadurch auch der Netzwerkverkehr weiter reduziert werden. Eine gängige Aufgabe in der Praxis wird beispielsweise die Aggregation von Sensordaten sein. Anstatt die gesamten Detaildaten zu jedem Peer zu senden, um überall die gleiche Aggregation durchzuführen, wird die Aggregation bereits nahe an der Datenquelle durchgeführt und nur die aggregierten Ergebnisse, welche im Allgemeinen ein weitaus geringeres Datenvolumen darstellen, werden an die entsprechenden Peers geliefert. Außerdem können bereits vorhandene aggregierte Datenströme im System zur Berechnung allgemeinerer Aggregate wiederverwendet werden, ähnlich zu den Roll-Up- und Cube-Operatoren im Data Warehousing Bereich.

Abbildung 4 zeigt die Anfrageauswertungsstrategie anhand des Beispielszenarios aus Kapitel 1. Die Symbole an den Netzwerkverbindungen repräsentieren Gruppen von Elementen. Die Raute steht für die Elemente »pos_error« und »field_id«, der Kreis für »phc«, das Dreieck für »dx« und »dy« und das Rechteck für »ra«, »dec«, »en« und »det_time«. Durch Projektionen verschwinden Symbole, da die entsprechenden Elemente aus dem Datenstrom herausgefiltert werden. Selektionen entfernen bestimmte Instanzen von Elementen, die das Selektionsprädikat nicht erfüllen. Dies wird durch gepunktete Symbole angedeutet. Ein Ausrufezeichen bezeichnet einen Wechsel in der Datenrepräsentation, z. B. die Einführung des Elements mit dem Namen »vela« an SP_2 im Ergebnis der Anfrage an P_0. In unserem Beispiel erfolgt die Einführung des Elements »rxj« in der Antwort auf die Anfrage an Peer P_2 an diesem Peer selbst und erscheint deshalb nicht im Netzwerk. Die Entscheidung, ob die Subskriptionsauswertung an P_2, SP_1 oder SP_2 durchgeführt werden soll, wird vom Optimierer getroffen und basiert auf überwachten Parametern, wie z. B. Rechenleistung und gegenwärtige Auslastung von Peers.

Der Auswertungsplan in Abbildung 4 zeigt die Situation, nachdem der Datenstrom und die beiden Anfragen aus Kapitel 1 im Netzwerk von Abbildung 1 registriert worden sind. Der Anfrageoptimierer hat die Anfragen bereits optimiert und in das System integriert. Zunächst werden die Daten durch einen Selektionsoperator auf den in der Anfrage an P_0 selektierten Himmelsbereich eingeschränkt und die Elemente »pos_error«und »field_id« durch einen Projektionsoperator aus dem Strom entfernt. Wie bereits in der Einleitung erläutert, enthält der in der Anfrage an P_0 selektierte Ausschnitt des Himmels den in der Anfrage an P_2 selektierten Ausschnitt vollständig. Um den Netzwerkverkehr zu reduzieren, entscheidet der Optimierer, den entsprechenden Selektions- und Projektionsoperator so nah wie möglich an der Datenquelle auszuführen. Da die Datenquelle P_4 ein einfacher Sensor ohne Anfragebearbeitungskapazitäten ist und daher die Selektion und die Projektion nicht selbst durchführen kann, werden die Operatoren im Netzwerk an Super-Peer SP_3 installiert und ausgeführt. Der aus Selektion und Projektion resultierende Datenstrom wird nur einmal (als ein zusammengefasster Strom) zu SP_2 geleitet. Die Entscheidung, auf welchen Wegen ein Strom durch das Netzwerk geleitet werden soll, basiert derzeit auf dem Ziel der Minimierung der Anzahl an Zwischenstationen (Peers), über die der Strom auf dem Weg von seiner Quelle zu den Empfängern im Netzwerk geleitet werden muss. Ausgefeiltere Optimierungsziele und Strategien zur Weiterleitung der Ströme im Netzwerk werden wir in künftigen Arbeiten untersuchen. An SP_2 wird der Strom mit dem Ziel

P_2 erneut gefiltert, indem ein Selektionsoperator die Energie und die Koordinaten der zu betrachtenden Photonen auf den in der Anfrage spezifizierten, kleineren Bereich einschränkt und ein Projektionsoperator das Element »phc« entfernt. Der verbleibende Strom wird über den kürzesten Weg zu P_2 weitergeleitet – in diesem Fall über SP_1. Die restliche Anfragebearbeitung, bestehend aus der Einführung des Elements »rxj«, wird an P_2 selbst ausgeführt. Der Strom mit dem Ziel P_0 wird ebenfalls an SP_2 gefiltert, und zwar unter Verwendung einer Projektion zur Entfernung der Elemente »dx« und »dy«, wie von der entsprechenden Anfrage gefordert. Außerdem wird das neue Element »vela« im Ergebnis der Anfrage bereits an SP_2 eingeführt. Eine erneute Selektion ist hier nicht mehr notwendig. Der sich ergebende Strom wird wiederum auf dem kürzesten Weg, der diesmal über SP_0 führt, zu P_0 weitergeleitet. Im Allgemeinen ist der kürzeste Weg nicht eindeutig und hängt von der zugrunde liegenden Netzwerktopologie ab. Im Zuge seiner Auswahl können auch weitere Optimierungsziele, wie z. B. Lastbalancierung auf Peers, berücksichtigt werden.

3.2 Anfragebearbeitung

In diesem Abschnitt werden die grundlegenden Konzepte, welche für die Anfragebearbeitung im Netzwerk eingesetzt werden, dargelegt. Betrachten wir zunächst die Ausführung von Filteroperationen. Diese Operationen projizieren einen Datenstrom auf die benötigten Teile des gesamten Schemas und selektieren Datenobjekte des Datenstroms anhand der Prädikate der Subskriptionsregeln. Das ursprüngliche Schema eines Datenstroms bleibt dabei – bis auf die Entfernung unnötiger Teile – gleich, insbesondere ändert sich die Reihenfolge der enthaltenen Elemente nicht. Projektionen können deshalb ausgeführt werden, ohne den Datenstrom zwischenzuspeichern. Die Ausführung von Selektionen ist schwieriger, da Daten nicht propagiert werden können, bevor ein Prädikat ausgewertet wurde, und somit ein Zwischenspeichern der Daten in der Regel nicht vermieden werden kann. Wir beschränken daher Filteroperationen auf Prädikate, die sich lokal auf ein Datenobjekt des Stroms beziehen. Damit muss maximal das aktuelle Datenobjekt zwischengespeichert werden. Unter diesen Voraussetzungen können Filteroperationen effizient und skalierbar mittels automatenbasierter Techniken [Koch, Scherzinger 2003] oder mit Hilfe der nachfolgend beschriebenen FluX Query-Engine [Koch et al. 2004], welche in Zusammenarbeit mit unserer Gruppe entwickelt wurde, ausgewertet werden.

FluX ist eine neue Optimierungstechnik zur Reduzierung des Speicherverbrauchs bei der Ausführung einer XQuery-Anfrage auf Datenströmen. Es stellt eine Zwischensprache zur Erweiterung der XQuery-Syntax um ereignisgesteuerte Ausführungsanweisungen dar, welche den kontrollierten Einsatz von (Hauptspeicher-)Puffern ermöglichen. Die zentrale Idee von FluX ist das *process-stream* Konstrukt `{ ps` $\$x$`: ` ζ `}` zur ereignisgesteuerten Verarbeitung des der Variable $\$x$ zugewiesenen Teildatenstroms. Dieser Datenstrom wird mit einer Menge von *Eventhandlern* ζ bearbeitet. Jeder Eventhandler ist von einer der beiden folgenden Formen

- `on` a `as` $\$y$ `return` α
- `on-first past(`S`) return` α

wobei α ein beliebiger Teilausdruck, a der Name eines XML-Tags und S eine Menge von Namen von XML-Tags ist. Ein »`on` a«-Handler wird ausgeführt, wenn vom Datenstrom $\$x$ ein öffnendes Tag mit Namen a gelesen wird. Die nachfolgenden (bzw. im Sinne des XML-Datenmodells untergeordneten) Elemente des Datenstroms werden als Datenstrom $\$y$ bezeichnet und an den Teilausdruck α, welcher wiederum ein FluX-Ausdruck sein kann, zur weiteren Auswertung übergeben. Ein »`on-first`«-Handler wird ausgeführt, wenn kein Element s mit $s \in S$ mehr im Datenstrom vorkommt, und stößt damit die Ausführung der Teilanfrage α an. Mit Hilfe von Informationen aus der DTD wird ein derartiger Handler so früh wie möglich zum passenden Zeitpunkt ausgeführt. In der Regel können XQuery-Anfragen nicht ohne Zwischenspeichern auf einem Datenstrom ausgeführt werden, z. B. wenn sich die Reihenfolge der Elemente in der Ausgabe der Anfrage von der Reihenfolge im Datenstrom unterscheidet. Eine FluX-Anfrage besteht daher im Allgemeinen aus einem (oder mehreren) ereignisgesteuerten process-stream Konstrukt(en) und darin eingebetteten XQuery-Teilanfragen, welche auf (vorher) gepufferten Teilen des Datenstroms ausgeführt werden.

Die Optimierung besteht nun darin, eine XQuery-Anfrage in eine entsprechende FluX-Anfrage umzuschreiben, welche die ursprüngliche Anfrage soweit wie möglich mit den ereignisgesteuerten Erweiterungen ausführt und den Bedarf an Hauptspeicherpuffern minimiert. In [Koch et al. 2004] wird ein Algorithmus vorgestellt, der dies mit Hilfe von Reihenfolgeinformationen über die Elemente aus der DTD des Datenstroms erreicht. Das Umschreiben von XQuery in FluX basiert auf der Generierung einer *sicheren* FluX-Anfrage. Eine FluX-Anfrage ist sicher, wenn alle in ihr enthaltenen XQuery-Teilanfragen, welche auf gepufferten Daten arbeiten, nur Datenobjekte im Strom referenzieren (z. B. durch Variablen oder Pfadausdrücke), welche nach dem Zeitpunkt der Ausführung einer derartigen Teilanfrage nicht mehr im Datenstrom vorkommen. Bei der Bearbeitung können so die benötigten Teile des Stroms leicht vollständig gepuffert und den auf den Puffern arbeitenden XQuery-Teilanfragen zur Verfügung gestellt werden. Die erste Anfrage des einleitenden Beispiels wird damit unter der gegebenen DTD in folgende FluX-Anfrage umgeschrieben (die »where«-Bedingung wird aus Gründen der besseren Lesbarkeit durch »C« abgekürzt).

```
{ps stream("photons")
 on photon as $p return
  {ps $p:
   on-first past(ra, dec)
    return
     {if (C) then <vela>}
     {for $r in $p/ra return
       {if (C) then {$r}}}
     {for $d in $p/dec return
       {if (C) then {$d}}};
   on en as $en return
    {if (C) then {$en}};
   on phc as $phc return
    {if (C) then {$phc}};
   on det_time as $d return
    {if (C) then {$d}};
   on-first past(*) return
    {if (C) then </vela>};};}
```

Diese (sichere) FluX-Anfrage wird wie folgt ausgewertet: Beim Lesen eines öffnenden »photon«-Tags befindet man sich im zweiten process-stream `{ps $p: ...}`. Die Query-Engine erkennt, dass im ersten »on-first«-Handler und

in »C« die beiden Pfade »$p/ra« und »$p/dec« referenziert werden und legt deshalb jeweils einen Puffer für die beiden Pfade und ein Boole'sches Flag für das Ergebnis der Bedingung an. Beim Lesen der Elemente »ra« und »dec« werden diese gepuffert und laufend die Bedingung ausgewertet. Nachdem »dec« gelesen wurde, kann aus der DTD gefolgert werden, dass diese beiden Elemente nicht mehr auftreten können und der erste »on-first«-Handler wird ausgeführt, welcher je nach Ergebnis der Bedingungsauswertung »<vela>« und den Inhalt der Puffer der beiden Pfade oder nichts ausgibt. Die übrigen Elemente können nun bei erfüllter Bedingung direkt, ohne sie zu puffern, ausgegeben werden – die richtige Reihenfolge und die korrekte Auswertung der Bedingung wird durch die DTD gewährleistet. Das Umschreiben der »where«-Bedingung in mehrere »if«-Anweisungen wirkt sich nicht negativ auf die Performanz aus, da die Bedingung nur einmalig ausgewertet und später nur noch das Ergebnis weiterverwendet wird. »on-first past(*)« ist eine Abkürzung für die Menge S der Namen aller möglichen Elemente und wird daher am Ende dieses Teilstroms, nachdem alle anderen Inhalte ausgegeben wurden, ausgeführt.

FluX ermöglicht die Auswertung von XQuery-Anfragen auf Datenströmen mit sehr geringem Speicherbedarf und somit auch die skalierbare Auswertung von Subskriptionsregeln. Details über FluX sowie eine experimentelle Bewertung finden sich in [Koch et al. 2004].

3.3 Anwendungsbeispiel

Kommen wir nun zu dem Beispiel aus Kapitel 1 zurück. Wir nehmen an, dass die zweite Gruppe von Astronomen eine weitere Anfrage an Peer P_2 in Abbildung 1 registriert. An P_2 soll dabei nur dann eine Meldung erscheinen, wenn der Mittelwert der Energien aller in den letzten 60 Sekunden gemessenen Photonen den Wert 1.3 keV übersteigt. Dabei interessieren nur Photonen aus dem Bereich des RXJ0852.0-4622 Supernova Überrests, wie in der Anfrage an P_2 in Kapitel 1. Um die gewünschten Informationen zu gewinnen, wird an P_2 die folgende Aggregationsanfrage in XQuery registriert.

```
let $p := stream("photons")/photon
  [ra > 130.5 and ra < 135.5 and
   dec > -48.0 and dec < -45.0]
  |$p/det_time diff 60 step 15|
let $a := avg($p/en)
where $a > 1.3
return
  <avg_energy>
    {$a}
  </avg_energy>
```

Die Zeile »|$p/det_time diff 60 step 15|« ist Teil unserer XQuery/XPath-Erweiterung zur Unterstützung fensterbasierter Operatoren. Sie spezifiziert, dass im Abstand von 15 Sekunden jeweils ein neuer Aggregatwert basierend auf den Energien der in den letzten 60 Sekunden im selektierten Himmelsbereich gemessenen Photonen berechnet wird. Außerdem reicht die Anfrage den Aggregatwert nur dann an P_2 weiter, falls er 1.3 keV übersteigt.

Aufgrund der Anfrage aus Kapitel 1 verfügt P_2 bereits über einen Teil der zur Beantwortung obiger Anfrage nötigen Daten. Es fehlen allerdings noch die Daten aller Photonen mit einer Energie von 1.3 keV und weniger. Um diese zu erhalten, kann der von SP_2 an P_2 gelieferte Datenstrom um die noch fehlenden Photonen erweitert werden, was einer Entfernung der Selektion »en > 1.3« an SP_2 gleich kommt. Diese Selektion, die zur Beantwortung der ersten Anfrage an P_2 benötigt wird, könnte dann erst an P_2 selbst ausgeführt werden. Dadurch würde allerdings der auf der frühen Selektion basierende Vorteil des verringerten Transfervolumens im Netzwerk verloren gehen. Eine bessere Alternative stellt die Verlagerung der Aggregatberechnung in das Netzwerk dar. Hierfür bietet sich SP_2 an, da dort bereits alle benötigten Daten vorhanden sind. Das zusätzliche Datenaufkommen ist dabei relativ gering, da auf der Datenverbindung zwischen SP_2 und P_2 nur einzelne Aggregatwerte, die das Selektionsprädikat erfüllen, zusätzlich übertragen werden. Außerdem muss die Selektion des zu betrachtenden Himmelsbereichs, die für beide Anfragen an P_2 identisch ist, nicht zweimal ausgeführt werden. Sie wird, wie in Kapitel 1 beschrieben, einmalig an SP_2 berechnet und dann für beide Anfragen jeweils wiederverwendet. Die Ausnutzung derartiger Optimierungsmöglichkeiten ist Aufgabe des StreamGlobe Anfrageoptimierers.

Mit einer letzten Beispielanfrage demonstrieren wir nun noch die Berechnung eines Joins eines Datenstroms mit einer persistenten Datenquelle in StreamGlobe. Dazu nehmen wir an, dass an P_2 eine Anfrage registriert werden soll, welche für jedes neu gemessene Photon die Energiedifferenz zwischen diesem neuen Photon und den Photonen einer früheren Messung berechnet. Dabei sollen die zu vergleichenden Photonen räumlich möglichst dicht zusammen liegen, also möglichst an derselben Stelle am Himmel aufgetreten sein, damit sie mit hoher Wahrscheinlichkeit von derselben Quelle stammen. Weiterhin sollen die Messzeitpunkte der früheren und der aktuellen Messung um etwa 90 Minuten, die Dauer einer Erdumrundung des Satelliten, auseinander liegen. Die Daten der früheren Messungen sind persistent in der Datenbank an Peer P_3 gespeichert. Die aktuellen Messwerte werden hingegen von dem satellitengestützten Teleskop an Peer P_1 als kontinuierlicher Datenstrom an das Netzwerk geliefert. Die Daten beider Datenquellen sollen derselben DTD genügen wie der in der Einleitung dargestellte Datenstrom von Peer P_4. Zur Berechnung der Paare von Messwerten der beiden Eingaben, die obigen Bedingungen bestmöglich entsprechen, bietet sich der Einsatz des BestmatchJoin-Operators (BMJ) an [Kemper, Stegmaier 2002]. Die an P_2 zu registrierende Anfrage hat dann die folgende Form in XQuery.

```
for $p1 in
  stream("photons")/photon
  for $p2 in
    document("photons_db")/photon
  where $p1 lobmj $p2 (
    abs($p1/det_time -
        ($p2/det_time + 5400))
      min 250 and
    abs($p1/ra - $p2/ra) min 1 and
    abs($p1/dec - $p2/dec) min 1)
  return
    <energy_diff>
      {$p1/en - $p2/en}
    </energy_diff>
```

Wir verwenden hier einen Left-Outer-BestmatchJoin (LOBMJ), um für jedes aktuell gemessene Photon ein passendes Photon einer früheren Messung zu finden. Wiederum stellen das Schlüsselwort »lobmj« und die übrigen Formulierungen im »where«-Teil der Anfrage unsere XQuery-Erweiterungen zur Unterstützung des BestmatchJoins dar. Sie formulieren die oben bereits beschriebene Join-Bedingung und geben weiterhin an, dass nur Paare von Photonen in das Ergebnis aufgenommen werden können, deren Messzeitpunkte um höchstens 250

Sekunden und deren Koordinaten um höchstens 1 Grad bezüglich Rektaszension und Deklination voneinander abweichen.

Der optimale Join-Auswertungsplan kann von mehreren Faktoren abhängen, z. B. von der Größe der Eingabedaten und des Join-Ergebnisses, von der aktuellen Netzbelastung auf bestimmten Verbindungen oder davon, wo sonst im Netz noch Teile der Eingaben oder des Ergebnisses benötigt werden. Ist das Join-Ergebnis beispielsweise relativ klein im Vergleich zu den Eingabedaten, so macht es Sinn, den Join so früh wie möglich zu berechnen, etwa an SP_0 oder SP_2. Ist das Join-Ergebnis hingegen groß verglichen mit den Eingaben, sollte er möglichst spät ausgeführt werden, also entweder direkt an P_2 oder, wenn P_2 dazu nicht in der Lage ist, an SP_1. Sollte sich allerdings ein neuer Peer an SP_3 registrieren und eine Anfrage stellen, die dasselbe Join-Ergebnis oder zumindest einen Teil davon benötigt, könnte es sinnvoller sein, den Join wiederum an SP_2 zu berechnen und dann das Ergebnis über SP_1 bzw. SP_3 an P_2 bzw. den neu an SP_3 registrierten Peer weiterzuleiten.

4 Zusammenfassung

In dieser Arbeit haben wir die adaptive Anfragebearbeitung und Optimierung auf Datenströmen in StreamGlobe skizziert. Ziel des Projekts ist ein effektives selbstorganisierendes DSMS, das die effiziente Informationsgewinnung in heterogenen Netzwerken ermöglicht. Im Gegensatz zu bestehenden P2P-Systemen lokalisiert StreamGlobe nicht nur Daten und führt anschließend die Anfragebearbeitung durch, sondern soll durch intelligente Ausnutzung von Anfragebearbeitungskapazitäten im Netzwerk einen möglichst effizienten Datenfluss erzielen. Um dies zu erreichen, kommen Operatoren zur strukturellen und inhaltlichen Filterung, sowie die Ballung von Datenströmen zum Einsatz. StreamGlobe baut auf dem Globus Toolkit auf und erweitert dieses um Fähigkeiten zur Anfragebearbeitung auf Datenströmen.

Die bereits implementierten Teile unseres StreamGlobe Prototyps umfassen die grundlegende Infrastruktur für den Aufbau des P2P-Netzwerks sowie eine eigenständige Implementierung der strombasierten XQuery-Engine FluX, welche zur Zeit in StreamGlobe integriert wird. Die Optimierungstechniken, wie sie in Abschnitt 3.1 vorgestellt wurden, befinden sich aktuell in der Entwicklung. Der Prototyp dient als Forschungsplattform für unsere zukünftigen Arbeiten.

Diese Arbeiten beinhalten u. a. weitere Untersuchungen im Bereich der Anfragebearbeitung auf Datenströmen, der Lastbalancierung sowie Quality-of-Service Aspekte [Braumandl et al. 2003] in einem verteilten DSMS. Im Detail wird dies die Erweiterung des FluX Anfragebearbeiters um die Unterstützung fensterbasierter Operatoren, wie z.B. Aggregationen, beinhalten. Außerdem beabsichtigen wir, die Optimierungskomponente durch Einbeziehung von Reorganisation zu verbessern und Prädikatvergleiche im Kontext der Ballung von Anfragen zu behandeln. In Zusammenarbeit mit Partnern aus der Astrophysik werden diese Konzepte in einer anwendungsspezifischen e-Science Domäne exemplarisch validiert.

Literatur

[Aschenbach 1998] Aschenbach, B. (1998). Discovery of a young nearby supernova remnant. *Nature*, 396(6707):141–142.

[Braumandl et al. 2001] Braumandl, R., Keidl, M., Kemper, A., Kossmann, D., Kreutz, A., Seltzsam, S., Stocker, K. (2001). ObjectGlobe: Ubiquitous query processing on the Internet. *The VLDB Journal*, 10(1):48–71.

[Braumandl et al. 2003] Braumandl, R., Kemper, A., Kossmann, D. (2003). Quality of Service in an Information Economy. *ACM Transactions on Internet Technology*, 3(4):291–333.

[Chandrasekaran et al. 2003] Chandrasekaran et al. (2003). TelegraphCQ: Continuous Dataflow Processing for an Uncertain World. In *Proc. of the Conf. on Innovative Data Systems Research*, Asilomar, CA, USA.

[Cherniack et al. 2003] Cherniack, M., Balakrishnan, H., Balazinska, M., Carney, D., Çetintemel, U., Xing, Y., Zdonik, S. B. (2003). Scalable Distributed Stream Processing. In *Proc. of the Conf. on Innovative Data Systems Research*, Asilomar, CA, USA.

[Diao et al. 2004] Diao, Y., Rizvi, S., Franklin, M. J. (2004). Towards an Internet-Scale XML Dissemination Service. In *Proc. of the Intl. Conf. on Very Large Data Bases*, pages 612–623, Toronto, Canada.

[GAVO 2004] GAVO (2004). German Astrophysical Virtual Observatory. http://www.g-vo.org.

[Keidl et al. 2002] Keidl, M., Kreutz, A., Kemper, A., Kossmann, D. (2002). A Publish & Subscribe Architecture for Distributed Metadata Management. In *Proc. of the IEEE Intl. Conf. on Data Engineering*, pages 309–320, San José, CA, USA.

[Kemper, Stegmaier 2002] Kemper, A., Stegmaier, B. (2002). Evaluating Bestmatch-Joins on Streaming Data. Technical Report MIP-0204, Universität Passau.

[Koch, Scherzinger 2003] Koch, C., Scherzinger, S. (2003). Attribute Grammars for Scalable Query Processing on XML Streams. In *Proc. of the Intl. Workshop on Database Programming Languages*, pages 233–256, Potsdam, Germany.

[Koch et al. 2004] Koch, C., Scherzinger, S., Schweikardt, N., Stegmaier, B. (2004). Schema-based Scheduling of Event Processors and Buffer Minimization on Structured Data Streams. In *Proc. of the Intl. Conf. on Very Large Data Bases*, pages 228–239, Toronto, Canada.

[Krämer, Seeger 2004] Krämer, J., Seeger, B. (2004). PIPES - A Public Infrastructure for Processing and Exploring Streams. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data*, pages 925–926, Paris, France.

[Liu, Franklin 2004] Liu, D. T., Franklin, M. J. (2004). GridDB: A Data-Centric Overlay for Scientific Grids. In *Proc. of the Intl. Conf. on Very Large Data Bases*, pages 600–611, Toronto, Canada.

[Löser et al. 2003] Löser, A., Siberski, W., Wolpers, M., Nejdl, W. (2003). Information Integration in Schema-Based Peer-To-Peer Networks. In *Proc. of the Intl. Conf. on Advanced Information Systems Engineering*, pages 258–272, Klagenfurt/Velden, Austria.

[Stegmaier et al. 2004] Stegmaier, B., Kuntschke, R., Kemper, A. (2004). StreamGlobe: Adaptive Query Processing and Optimization in Streaming P2P Environments. In *Proc. of the Intl. Workshop on Data Management for Sensor Networks*, pages 88–97, Toronto, Canada.

[Voges et al. 1999] Voges et al. (1999). The ROSAT All-Sky Survey Bright Source Catalogue. *Astronomy and Astrophysics*, 349(2):389–405.

[Yang, Garcia-Molina 2003] Yang, B., Garcia-Molina, H. (2003). Designing a Super-Peer Network. In *Proc. of the IEEE Intl. Conf. on Data Engineering*, pages 49–60, Bangalore, India.

[Yao, Gehrke 2002] Yao, Y., Gehrke, J. (2002). The Cougar Approach to In-Network Query Processing in Sensor Networks. *ACM SIGMOD Record*, 31(3):9–18.