Sensordatenverarbeitung mit dem Open Source Datenstrommanagementframework Odysseus

André Bolles[1], Dennis Geesen[1], Marco Grawunder[1],
Jonas Jacobi[1], Daniela Nicklas[2], H.-Jürgen Appelrath[1]

Abteilung Informationssysteme[1] / Datenbanken & Internettechnologien[2]
Department für Informatik
Universität Oldenburg
26129 Oldenburg

Abstract: Die Verarbeitung von kontinuierlichen Datenströmen, bspw. aus Sensoren, gewinnt zunehmend an Bedeutung. Flexible Möglichkeiten zur Verarbeitung solcher Daten werden durch Konzepte des Datenstrommanagements (DSM) bereitgestellt. Bisherige prototypische Datenstrommanagementsysteme (DSMS) wurden jeweils zur Evaluation bestimmter Verarbeitungskonzepte entwickelt. Der Vergleich oder die kombinierte Betrachtung dieser verschiedenen Konzepte ist bisher nur unter großem Aufwand möglich, da hierzu verschiedene DSMS integriert werden müssen. An der Universität Oldenburg wird daher ein Datenstrommanagementframework (DSMF) namens Odysseus entwickelt, welches es ermöglicht, nahezu beliebige Aspekte des DSM zu implementieren und zu evaluieren. Dieses Framework wird noch 2010 als Open Source DSMF veröffentlicht. Die vorliegende Arbeit stellt Odysseus und dessen Erweiterbarkeit vor und erläutert Fragestellungen, die bei der Veröffentlichung auftreten.

1 Einleitung

Sensoren sind in immer mehr Anwendungsfeldern, wie der Überwachung von Energienetzen oder zur Objektverfolgung in Fahrerassistenzsystemen, verbreitet. Die von Sensoren produzierten Datenströme müssen kontinuierlich verarbeitet werden. Konzepte des Datenstrommanagements (DSM, auch als complex event processing (CEP) bekannt) bieten hierzug analog zur Anfrageverarbeitung in Datenbanken flexible Möglichkeiten. Auf Grund der Heterogenität verschiedener, in den vergangenen Jahren entwickelter DSMS ist eine Kombination oder auch ein Vergleich der verschiedenen, dort implementierten Konzepte nur unter großem Aufwand möglich. Häufig ist eine Neuimplementierung in einem anderen DSMS nötig. An der Universität Oldenburg wurde daher mit Odysseus [BG+09] ein Datenstrommanagementframework (DSMF) entwickelt, welches als Plattform zur Evaluation nahezu beliebiger Konzepte des DSM dient. Odysseus wird noch 2010 als Open Source DSMF veröffentlicht, um auch anderen Forschungs- und Entwicklungseinrichtungen die Umsetzung ihrer Konzepte zur kontinuierlichen Verarbeitung von Sensordaten zu ermöglichen. In dieser Arbeit werden die Vorteile von Odysseus vorgestellt und Fragestellungen erläutert, die bei der Veröffentlichung als Open Source DSMF auftreten.

Hierzu werden im Kapitel 2 verwandte Arbeiten im Bereich DSM vorgestellt. Odysseus selbst wird mit seiner modularen Architektur in Kapitel 3 vorgestellt. Aufbauend darauf wird in Kapitel 4 beispielhaft beschrieben, wie eigene Konzepte der kontinuierlichen Datenverarbeitung in Odysseus integriert werden können. Fragestellungen, die bei der Veröffentlichung von Odysseus als Open Source DSMF auftreten, werden in Kapitel 5 erläutert, bevor abschließend diese Arbeit in Kapitel 6 zusammengefasst wird.

2 Verwandte Arbeiten

Obwohl DSM ein relativ junges Forschungsfeld ist, existieren bereits einige prototypische DSMS wie STREAM [AB+03], Borealis [AA+05], TelegraphCQ [CC+03] oder auch PIPES [KS09]. Diese DSMS dienen in erster Linie dazu bestimmte Verarbeitungskonzepte von Datenströmen zu evaluieren. So werden in STREAM Datenströme für die Verarbeitung zunächst in Relationen und anschließend zurück in Datenströme transformiert. Auch die Umsetzung des sogenannten Fensterkonzeptes unterscheidet sich in den einzelnen Systemen. Fenster werden dazu eingesetzt Ausschnitte auf Datenströmen zu definieren, um nicht alle Daten des Datenstroms für die Verarbeitung einlesen zu müssen. Diese Fenster werden in [AA+05, CC+03] bspw. an die Operatoren angehängt, damit diese entscheiden können, welche Elemente des Datenstroms aktuell zur Verarbeitung betrachtet werden müssen. In PIPES existieren dagegen eigene Fensteroperatoren, die sogenannte Gültigkeitsintervalle an die Elemente eines Datenstroms anhängen. Diese Gültigkeitsintervalle können von den nachfolgenden Operatoren ausgelesen werden, so dass auch hier entschieden werden kann, welche Elemente aktuell gültig sind. Ein alternativer Ansatz zu den Gültigkeitsintervallen wird wiederum in Aurora verwendet, wo die Gültigkeit mit sogenannten positiven Elementen beginnt und durch entsprechende negative Elemente beendet wird. Neben diesen Aspekten unterscheiden sich die Systeme auch in den verwendeten Datenmodellen. Während die zuvor genannten Systeme auf das relationale Modell setzen, werden in [GG+04] XML-Datenströme verarbeitet. Natürlich gibt es weitere Unterschiede zwischen den Systemen, die an dieser Stelle aber nicht alle genannt werden können.

Im Vergleich zu den genannten DSMS wird mit Odysseus jedoch nicht das Ziel verfolgt, bestimmte Konzepte des DSM evaluieren zu können, sondern vielmehr eine Plattform bereitzustellen, mit der eben nahezu beliebige Konzepte evaluiert werden können. So können mit Odysseus bspw. verschiedene Datenmodelle untersützt werden oder aber auch beide zuvor genannten Fensterrepräsentationen. Die Architektur, die für ein solches DSMF notwendig ist, wird im folgenden Kapitel beschrieben.

3 Odysseus

Odysseus ist ein Datenstrommanagementframework (DSMF), welches vollständig in Java implementiert ist. Es besitzt eine komponentenorientierte, OSGi-basierte Architektur (Abbildung 1), die es erlaubt DSMS für verschiedene Anwendungsdomänen zu entwickeln.

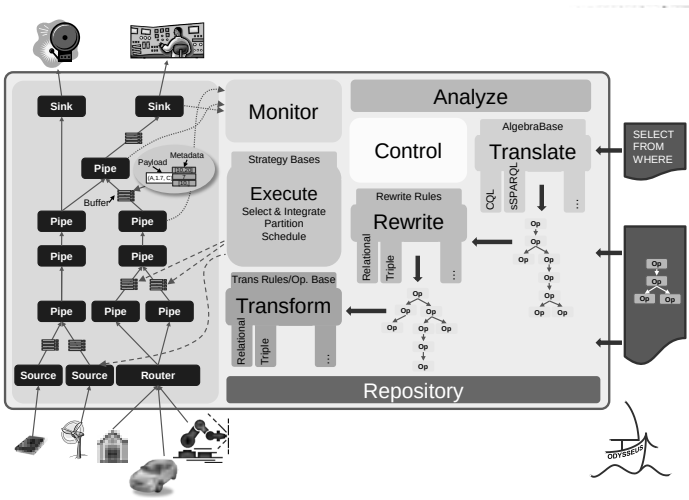

Abbildung 1: Architektur von Odysseus

Odysseus' Architektur kann dabei in sogenannte Fix- und Variationspunkte unterteilt werden. Fixpunkte sind die Schnittstellen, die Odysseus für verschiedene Aufgaben zur Verfügung stellt. Mit den Variationspunkten ist eine Anpassung der einzelnen Komponenten an verschiedene Bedürfnisse möglich.

In Abbildung 1 ist rechts zu erkennen, dass verschiedene Möglichkeiten existieren, Anfragen in Odysseus zu installieren. Neben verschiedenen Anfragesprachen wie Stream-SQL oder Stream-SPARQL [BG+08] (*SELECT, FROM, WHERE*) können Anfragepläne auch direkt über eine prozedurale Anfragesprache in Form eines Operatorbaums beschrieben werden. Über eine einheitliche in Odysseus implementierte Parser-Schnittstelle (*TRANSLATE*) (Fixpunkt) können diese Anfragesprachen in logische Anfragepläne übersetzt werden. Auch andere Anfragesprachen, wie die Esper Query Language (http://esper.codehaus.org), können so in Odysseus integriert werden.

Logische Anfragepläne stellen Informationen über eine Anfrage, wie bspw. Join-Prädikate bereit. Odysseus stellt abstrakte logische Operatoren als Fixpunkt bereit, die die Verknüpfung mehrerer Operatoren zu ganzen Anfrageplänen erlauben. Als Variationspunkt können eigene logische Operatoren von diesen abstrakten Operatoren ableiten und so spezifische Metadaten zur Verfügung stellen.

Logische Operatoren enthalten nur die Metadaten, aber noch keine Algorithmen zur Ausführung einer Anfrage. Daher werden logische Anfragepläne durch eine Regelmaschine in physische, ausführbare Anfragepläne übersetzt. Als Fixpunkt stellt Odysseus hier eine Schnittstelle zu einer Regelengine[1] bereit. Mit eigenen Regeln kann die Übersetzung in verschiedene physische Anfragepläne gesteuert werden (*REWRITE, TRANSFORM*, Variationspunkt).

[1] Drools, http://www.jboss.org/drools, zuletzt besucht: 04.05.2010

Für physische Anfragepläne stellt Odysseus ein Metadatenframework bereit, über das beliebige Metadaten an einzelne Datenstromelemente angehängt werden können. So kann bspw. sowohl der in Kapitel 2 genannte Positiv/Negativ-Ansatz als auch der Intervallansatz umgesetzt werden. Auch abstrakte, physische Operatoren sind zunächst datenmodellunabhängig, so dass eigene Operatoren zur Verarbeitung neuer Datenmodelle entwickelt werden können.

Es gibt weitere Komponenten, mit denen die Ausführung und Überwachung von Anfragen gesteuert werden kann. Die *EXECUTE*-Komponente bietet eine Schnittstelle (Fixpunkt) zur Ausführung physischer Anfragepläne, wie sie links in der Abbildung dargestellt sind. Verschiedene Ausführungsstrategien bilden einen Variationspunkt. Mit *CONTROL* kann bei Veränderungen der Anfrageverarbeitung, die über *MONITOR* und *ANALYZE* identifiziert werden, korrigierend eingegriffen werden. Weitere Details sind in [BG+09] dargestellt.

Durch die OSGi-basierte Architektur werden alle genannten Komponenten in eigene Bundles ausgelagert. Auch die Entwicklung neuer Konzepte bspw. mit neuen Metadaten, die an den Datenstromelementen angehängt werden, kann in eigene Bundles ausgelagert werden, so dass der Rest des Frameworks unverändert bleibt. Dies erlaubt zudem, nur bestimmte Bundles für bestimmte Anwendungsdomänen bereitzustellen, so dass Entwickler, die Odysseus für ihre Anwendung nutzen wollen, nicht das vollständige Framework mit allen spezialisierten Bundles installieren müssen.

4 Erweiterbarkeit von Odysseus

Dieses Kapitel illustriert die Erweiterbarkeit von Odysseus anhand des Beispiels der Objektverfolgung in einem Fahrerassistenzsystem.

Objektverfolgung Bei der Objektverfolgung sollen Objekte die bspw. durch eine Bildverarbeitung in einem Videobild erkannt wurden, in späteren Videobildern wiedererkannt werden. Beispielhaft passiert hierbei Folgendes: Ein Objekt o_0 sei zum Zeitpunkt t_0 an Position x_0 erkannt worden. Zum Zeitpunkt t_1 sei ein ähnliches Objekt o_1 an Position x_1 erkannt worden. Jetzt soll festgestellt werden, ob $o_0 = o_1$ ist. Hierzu wird mit einem Bewegungsmodell die Position $\hat{x}_0$ von o_0 zum Zeitpunkt t_1 geschätzt. Dann werden $\hat{x}_0$ und x_1 verglichen. Liegen diese dicht beieinander, kann davon ausgegangen werden, dass $o_0 = o_1$ gilt. Jetzt muss jedoch noch die tatsächliche Position x_{neu} aus $\hat{x}_0$ und x_1 geschätzt werden. Hierzu werden Filter, wie der Kalman-Filter eingesetzt. Anschließend beginnt dieser Prozess von vorne, diesmal jedoch mit der neuen Position x_{neu}.

Zyklische Anfragen Will man diesen Prozess der Objektverfolgung in einem DSMS abbilden, so muss die Möglichkeit geschaffen werden, zyklische Anfragen formulieren und ausführen zu können. Weiterhin müssen immer die aktuellen Zustände der Objekte (die Position im zuvor genannten Beispiel) zwischengespeichert werden.

Zur Formulierung zyklischer Anfragen bedarf es der Erweiterung einer Anfragesprache. Dies sei hier am Beispiel von StreamSQL gezeigt. Will man einen Zyklus ausdrücken, so muss es möglich sein, Daten aus einer Quelle auszulesen und auch wieder in diese Quel-

le hineinzuschreiben. Während mit SELECT $*$ FROM <name> bereits Daten aus einer Quelle ausgelesen werden können, wird mit SELECT $*$ INTO <name> FROM <name> eine Erweiterung der Grammatik vorgenommen, die auch das Schreiben in eine Quelle ausdrückt.

Eine solche Quelle soll die Speicherung und Auslieferung von Objektzuständen übernehmen und wird als sogenannter Broker-Operator in Odysseus integriert. Die Integration dieses Operators erfolgt an verschiedenen Stellen im System. Ein logischer Broker erlaubt den Aufbau zyklischer Anfragepläne. In einem physischen Broker ist der Algorithmus zur Verteilung der Objektzustände enthalten. Bei der Verteilung muss u.a. darauf geachtet werden, dass Objektzustände nicht weitergeleitet werden, bevor eine Aktualisierung abgeschlossen ist.

Die Komponenten für zyklische Anfragen befinden sich in einem eigenen OSGi-Bundle, so dass die Unabhängigkeit des Odysseus-Kerns von diesem neuen Bundle gewährleistet wird.

5 Open Source Release

Es ist noch innerhalb dieses Jahres eine Veröffentlichung von Odysseus als Open Source geplant. Es ist das Ziel anderen Forschergruppen ein Framework für DSMS zur Verfügung zu stellen. Hierbei wird nicht eine Weiterentwicklung des Odysseus-Kerns angestrebt, sondern vielmehr die Bereitstellung neuer Module durch andere Forschergruppen. Außerdem erhoffen wir uns eine Verbreitung des Themas Datenstrommanagement. Wir prüfen aktuell noch unterschiedliche Alternativen, wie die Art der Veröffentlichung aussehen kann und welche Lizenzmodelle auf Grund von verwendeten Bibliotheken notwendig sein werden. Als mögliche Verbreitungsmechanismen sind geplant:

Die Veröffentlichung von Odysseus als Eclipse Rich Client Anwendung Dabei wird ein Teil der Software direkt zum Download zur Verfügung gestellt und weitere Module und Aktualisierungen können mit Hilfe des in Eclipse eingebauten Update-Mechanismus vertrieben werden. Es ist auf diese Weise sehr einfach möglich, unterschiedliche Arten von Paketen zu definieren, von einem einfachen relationalen Odysseus bis hin zu einem kompletten Odysseus, ohne die einfache Updatemöglichkeit zu verlieren.

Der Zugriff auf das SVN Derzeit wird Odysseus durch das Versionsverwaltungssystem SVN verteilt. Solange nur eine beschränkte und bekannte Anzahl von Nutzern zugreift, ist dies die beste Lösung. Bei einer Veröffentlichung würden wir hier vermutlich zunächst nur lesenden Zugriff auf das Repository erlauben.

Download von Releases Die letzte angedachte Möglichkeit besteht darin, den vollständigen Sourcecode in Form eines Archivs zum Download zur Verfügung zu stellen. Dies hätte den Nachteil, dass nicht immer die aktuellste Version zur Verfügung stehen würde und Fehler u.U. erst später korrigiert werden.

Odysseus verwendet eine Reihe von externen Bibliotheken. Vor der finalen Veröffentlichung ist hier zunächst noch einmal zu testen, welche Lizenzmodelle in Frage kom-

men können. Da wir möglichst wenig einschränken und eine weite Verbreitung fördern möchten, präferieren wir aktuell Lizenzmodelle wie sie z.B. von der Apache Foundation angewendet werden. Da einige Bibliotheken (u.a. JEP und Drools) von uns speziell für Odysseus erweitert wurden muss hier zunächst der rechtliche Rahmen geklärt werden.

6 Zusammenfassung

In dieser Arbeit wurde Odysseus als ein Framework zur Entwicklung von Datenstrommanagementsystemen vorgestellt. Die Architektur und die Erweiterungsmöglichkeiten von Odysseus wurden sowohl abstrakt beschrieben als auch anhand der Erweiterung um zyklische Anfragen dargestellt. Die Integration neuer Konzepte in Odysseus kann bei Einhaltung der vorgegebenen Schnittstellen mit relativ geringem Aufwand erfolgen, da nur neue Konzepte angefasst werden müssen und bestehende Konzepte wie z. B. der Scheduler wiederverwendet werden können. Abschließend wurde aufgezeigt, welche Fragestellungen bei der Veröffentlichung von Odysseus als Open Source Framework auftreten. Verschiedende Veröffentlichungsmechanismen wie eine Eclipse Rich Client Anwendung, ein SVN-Zugriff oder Release-Downloads sind vorstellbar. Außerdem müssen Lizenzmodelle, insbesondere hinsichtlich der Verwendung von Bibliotheken untersucht werden.

Literatur

[AA+05] Daniel J. Abadi, Yanif Ahmand et al. The Design of the Borealis Stream Processing Engine. In *Second Biennial Conference on Innovative Data Systems Research (CIDR 2005)*, Januar 2005.

[AB+03] Arvind Arasu, Brian Babcock et al. STREAM: The Standford Stream Data Manager. *IEEE Data Engineering Bulletin*, 26(1), 2003.

[BG+08] Andre Bolles, Marco Grawunder et al. Streaming SPARQL - Extending SPARQL to Process Data Streams. In *ESWC 2008*, 2008.

[BG+09] Andre Bolles, Marco Grawunder et al. Odysseus: Ein Framework für maßgeschneiderte Datenstrommanagementsysteme. In *Informatik 2009 - Im Fokus das Leben*, 9 2009.

[CC+03] Sirish Chandrasekaranand, Owen Cooper et al. TelegraphCQ: Continuous Dataflow Processing for an Uncertain World. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*, 2003.

[GG+04] Todd J. Green, Ashish Gupta et al. Processing XML streams with deterministic automata and stream indexes. *ACM Trans. Database Syst.*, 29(4), 2004.

[KS09] Jürgen Krämer und Bernhard Seeger. Semantics and implementation of continuous sliding window queries over data streams. *ACM Trans. Database Syst.*, 34(1), 2009.