In: Proceedings of the 14th Workshop 'Workshop über Grundlagen von Datenbanken'. Halbinsel Fischland-Darß-Zingst, Germany, May 21-24, 2002, pp. 96--100.

OSIRIS – Eine generische Plattform für die verteilte Ausführung von Geschäftsprozessen

Christoph Schuler Heiko Schuldt

Datenbankforschungsgruppe, Institut für Informationssysteme
Eidgenössische Technische Hochschule (ETH)
CH–8092 Zürich, Schweiz
Email: {schuler,schuldt}@inf.ethz.ch

Zusammenfassung

Die zunehmende Verbreitung von Protokollen wie z.B. SOAP und WSDL bzw. Repositorys wie UDDI ermöglicht die einfache Nutzung bestehender Dienste in Form von Web-Services oder E-Services, auch über Netzwerkgrenzen hinweg. Prozesse bieten die Möglichkeit, verschiedene Aufrufe solcher Diensten zu einem gemeinsamen Ganzen in komplexen, verteilten Anwendungen zu kombinieren. Eine wichtige Rolle nehmen dabei Umgebungen ein, die die Ausführung von Prozessen, basierend auf existierenden Diensten, ermöglichen. Die Autonomie der Service-Anbieter und die Dynamik des Systems (neue Provider können hinzukommen, existierende können verschwinden, das Angebot pro Provider kann sich ändern, etc.) stellen besondere Anforderungen, die sich mit einem zentralen Prozess–Koordinator schwerlich realisieren lassen.

Die vorliegende Arbeit zeigt, wie die Ausführung von Prozessen in einem hoch dynamischen Umfeld realisiert werden kann. Dabei wird die Funktionalität des Prozess-Koordinators verteilt implementiert. Es wird zur Laufzeit entschieden, auf welcher Komponente der nächste Schritt im Prozess ausgeführt wird. Zur Verteilung und Konsistenzsicherung der für die Prozessausführung benötigten Meta-Information kommen Publish/Subscribe–Techniken zum Einsatz. Die konsequente Anwendung dieser Techniken auch zur Laufzeit erlaubt die weitestgehende Entkopplung der einzelnen Komponenten und Dienste. Durch das Publizieren eines Auftrags zur Ausführung eines Prozess-Schrittes kann zur Laufzeit dynamisch ein Provider ausgewählt werden, der den mit dem Auftrag verbundenen Dienst zuvor durch eine Registrierung angeboten hatte. Diese Indirektion lässt zudem die Realisierung weiterer Konzepte wie Service Discovery, Load Balancing und erweitertes Routing wie zum Beispiel die Auswahl des günstigsten Angebots zu.

1 Motivation

In einer modernen Umgebung sind viele Anwendungen und Dienste durch offene Schnittstellen —auch von ausserhalb der Systemgrenzen— nutzbar. Diese Schnittstellen kapseln die vorhandenen Dienste in einer einheitlichen Weise. Dadurch wird die Wiederverwendung der Funktionalität dieser Dienste in einem grösseren Zusammenhang, z.B. in Form von Geschäftsprozessen, ermöglicht. Die Geschäftspartner sind relativ einfach austauschbar, da durch standardisierte Schnittstellen (z.B. SOAP [W3C]) die Dienste oft kompatibel sind. Durch die Definition von Prozessen über diesen als Web– oder E–Service publizierten Diensten lassen sich basierend auf diesen einzelnen Bausteinen komplexe Abläufe zusammensetzen. Durch die Definition von Abhängigkeiten zwischen einzelnen Dienstaufrufen können solche Abläufe, mit Hilfe einer geeigneten Infrastruktur, in korrekter Weise koordiniert werden.

Prozess–Koordinatoren oder Workflow–Management–Systeme, die eine solche Infrastruktur bereitstellen, sind meistens zentral implementiert. Doch in einer modernen Umgebung,

wo die Zusammenarbeit verschiedener Partner über Web–Services über Systemgrenzen sowie über grosse geografische Distanzen hinweg immer wichtiger wird, sind zentrale Ansätze oft ungenügend und behindern die einzelnen Parteien in ihrer Autonomie. Deshalb ist ein verteilter Ansatz für die Ausführung dieser Prozesse anzustreben. Um diese verteilte Prozessausführung zu ermöglichen, muss Metainformation (z.B. Prozessmodelle) im System repliziert werden. Da sich diese Metadaten ändern können, muss auch sichergestellt werden, dass die Daten überall auf dem gleichen Stand sind. Mittels Publish/Subscribe–Methoden lässt sich dieses Ziel erreichen. Ähnliches gilt auch für Load-Information, die im System verteilt wird, um die lastbalanzierte Ausführung mehrerer Prozesse im System zu ermöglichen.

Prozesse erfordern zumeist dezidierte Ausführungsgarantien, z.B. flexible Fehlerbehandlung. Dies bedeutet, das die bekannten Konzepte aus dem zentralen Fall erweitert und verallgemeinert werden müssen, um sie auf eine verteilte Infrastruktur für die Prozessausführung anwenden zu können. Im Falle von flexibler Fehlerbehandlung bedeutet dies, dass jeder Prozess in einem konsistenten Zustand terminiert, z.B. durch alternative Ausführungen, die im Fehlerfall durchgeführt werden. Als Konsequenz davon muss ein in der Praxis verwendbarer Prozess–Koordinator erweiterte Datenbankfunktionalität [SBG+00] für die auszuführenden Prozesse bereitstellen.

Das an der ETH Zürich entwickelte Prototypsystem OSIRIS (Open Services Infrastructure for Reliable and Integrated Process Support) stellt eine solche Infrastruktur bereit. OSIRIS implementiert die benötigten Konzepte mittels Peer-to-Peer und Publish/Subscribe–Methoden. Das Ergebnis ist ein generisches System, das je nach konkreter Anwendung optimal konfiguriert werden kann. So wurde OSIRIS bereits erfolgreich in mehreren Anwendungsszenarien eingesetzt: der Erhaltung der Konsistenz von verteilten Datenbeständen, der effizienten Ausnutzung eines Clusters sowie zur garantierten Abwicklung verteilten Zahlungstransaktionen (Payment Prozesse) im Internet. Auf der Basis der verteilten Infrastruktur zur Prozessausführung haben wir auch erweiterte Konzepte für das Monitoring laufender Prozessinstanzen in das OSIRIS System eingebaut, um so die inhärente Verteilung für den Benutzer transparent zu machen.

2 System–Umgebung

Das Osiris System setzt auf einer heterogen, verteilten und autonomen Systemumgebung auf. Jeder Service–Provider unterhält sein eigenes autonomes System, auf welchem er seinen Dienst über eine standardisierte Schnittstelle (z.B. SOAP [W3C]) zur Verfügung stellt. Die Menge der angemeldeten Provider und Dienste ist nicht statisch. Neue Provider und Dienste können dynamisch zum System hinzukommen oder bestehende Komponenten können das System wieder verlassen.

Um Prozesse in dieser dynamischen Umgebung unterstützen zu können, braucht es zwei Arten von Systemkomponenten: zum einen zentrale Systemkomponenten, welche die Konfiguration überwachen bzw. Metainformation sammeln und verteilen und zum anderen eine zusätzliche Softwareschicht, die bei jedem Dienstanbieter installiert ist und die die Interaktion mit dem Gesamtsystem gewährleistet.

2.1 Dynamische Systemkonfiguration

Zu den zentralen Osiris–System–Diensten gehört ein Verzeichnis aller im System vorhandenen Dienste und Komponenten. Aufbauend auf Publish/Subscribe–Methoden liefert dieses Verzeichnis die Grund-Infrastruktur für die gesamte Kommunikation im System. Jeder Dienst meldet sich als Subscriber für sein Start-Event beim zentralen Service an. Auf diese Weise startet ein Publizieren der dieses Event beinhaltenden Nachricht irgendwo im System genau diesen Dienst. Somit ist eine Entkoppelung in der Kommunikation erreicht. Die zentrale Komponente dient lediglich als Verzeichnis; die Kommunikation zwischen zwei Komponenten geschieht in einer Peer-to-Peer-Weise und arbeitet mit einem lokalen Cache dieser Subscriber-Datenbank. Die

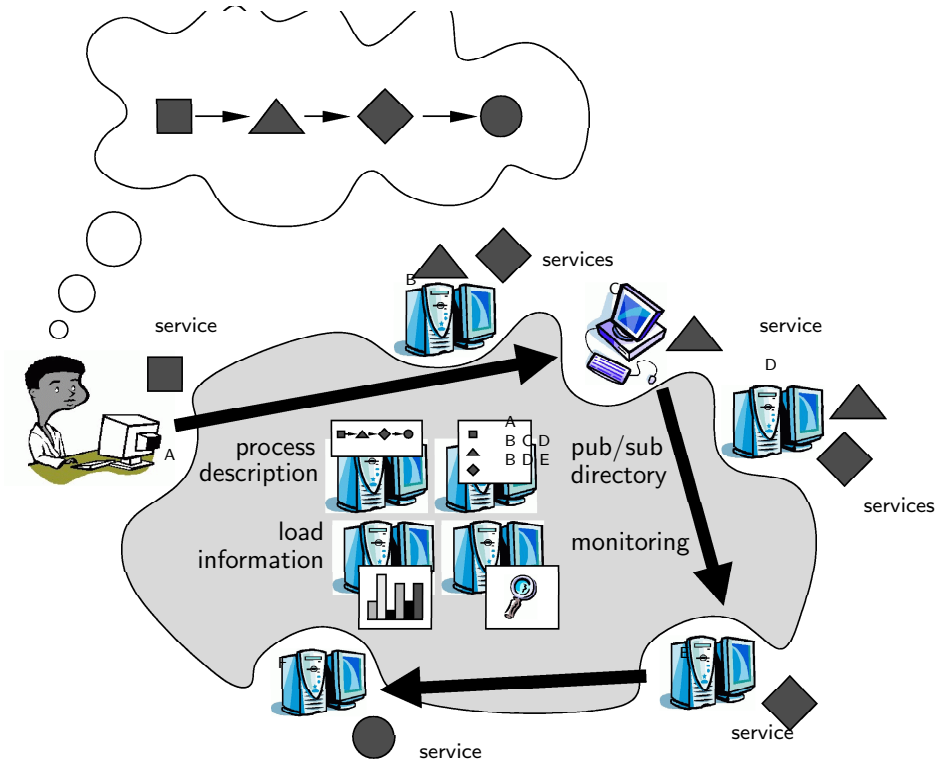

Abbildung 1: *Ausführung von Prozessen im verteilten Umfeld*

replizierten Daten werden lokal auf jeder Komponente innerhalb der Osiris–Schicht auf den Komponenten verwaltet. Aktualisiert werde diese Daten wiederum durch eine Publikation jeder Änderung.

2.2 Prozessausführung

Eine wichtige System–Komponente verwaltet alle definierten Prozessmodelle. Neben einer zentralen Kopie dieser Metadaten, werden Teile der Prozessbeschreibung an die Anwendungskomponenten weitergeleitet, die den jeweiligen Prozessauschnitt ausführen können. Dadurch erhalten diese Komponenten das Wissen über den jeweils nachfolgenden Schritt im Prozess und damit auch die Möglichkeit, nach der Ausführung eines Prozessschrittes auf direktem Weg die Kontrolle an die betreffende Komponente des nächsten Schrittes zu senden (siehe Abb. 1), ohne auf eine Indirektion für die Bestimmung des Zieles für die Ausführung des nächsten Schrittes zu verzichten, denn durch eine implizierte Registrierung für Änderungen an diesem Teilen des Prozessmodells werden alle Änderungen automatisch zu den Replikaten weitergeleitet. Auf diese Weise wird ein konsistenter Zustand der lokalen Kopien erreicht [SSS01].

2.3 Replikation von Metadaten

Die Konsistenzhaltung der Metadaten erfordert bei unserem verteilten Ansatz mehr Aufwand als bei einer zentralistischen Lösung. Ein naiver Ansatz würde jede Änderung der Metadaten in einer verteilten Transaktion durchführen. Die dadurch bedingte Blockierung des Systems hätte gravierende Auswirkungen auf die Performance des Gesamtsystems. Durch eine Analyse der unterschiedlichen Arten von Metadaten erkennt man jedoch, dass nur bestimmte Daten immer konsistent gehalten werden müssen und einige auch in einer möglicherweise leicht veralteten Version ausreichen.

Es lassen sich dabei drei Typen von Metadaten identifizieren:

- **Zentral verwaltete Daten** wie Einträge über gesperrte Ressourcen werden nicht repliziert, da diese Daten zentral ausgewertet werden müssen.
- **Verteilte Metadaten** wie Prozessdefinitionen müssen auf allen Rechnern aktuell und konsistent zur Verfügung stehen. Mittels Zeitstempel-Verfahren lassen sich diese Daten auf ihre Konsistenz testen. Eine verteilte Datenhaltung macht nur bei Metadaten Sinn, die sich verhältnismässig selten ändern, z.B. Prozessdefinitionen.
- **Leicht veraltete Zustandsdaten** können genügen, wenn es um die Entscheidung bei Load Balancing oder zur Service Discovery geht.

2.4 Korrektheit und Fehlertoleranz

Prozesse erfordern wie klassische Datenbank–Transaktionen Garantien. Die bekannten Konzepte aus dem zentralen Fall müssen dabei erweitert werden. Da Prozesse oft lang dauernde Vorgänge beschreiben, ist es nicht sinnvoll, den gesamten Prozess als eine Transaktion im klassischen Sinne zu betrachten. Vielmehr können die einzelnen Schritte des Prozesses als einzelne Transaktionen betrachtet werden. Die korrekte Ausführung des gesamten Prozesses garantiert der Prozess–Koordinator durch eine Realisierung der Konzepte transaktionaler Prozesse [SABS02]. Dabei wird garantiert, aufbauend auf dem Modell der flexiblen Transaktionen [ZNBB94], dass ein Prozess immer in einem konsistenten Zustand terminiert. Dies erreicht man durch Einführen von Alternativen im Kontrollfluss und durch Kompensation einzelner Schritte, die im Fehlerfall ausgeführt werden.

Um eine korrekte Ausführung zu garantieren, müssen Metainformationen zur Ausführung einer Prozess–Instanz persistent gespeichert werden. Durch eine lokale Datenbank und durch persistente Übertragung der Instanzdaten zur nächsten Komponente wird die Dauerhaftigkeit eines jeden Prozesses garantiert. Zusätzlich muss für die parallele Ausführung von Prozessen, die auf gemeinsame Resourcen zugreifen, eine Koordination durchgeführt werden. Zu diesem Zweck unterhält der Concurrency Control Service eine entsprechende Sperrtabelle, die nach einem klassischen Verfahren den Zugriff auf gemeinsame Ressourcen serialisiert. Der Kontrollfluss wird dazu von der Osiris–Schicht mittels einer Indirektion über diese zentrale Komponente geleitet, bevor der Service auf dem lokalen System gestartet werden kann.

3 Der OSIRIS Prototyp

Die verteilte Infrastruktur zur Ausführung von Prozessen besteht, neben den wenigen zentralen Komponenten, im wesentlichen aus einem Daemon auf jeder Komponente, die Dienste bereitstellt. Zusätzlich zur Basisfunktionalität durch den Aufruf lokaler Dienste steht innerhalb der Osiris–Softwareschicht eine lokale Datenbank für die Haltung von persistenten Daten zur Verfügung. Darauf aufbauend bietet die Infrastruktur eine eine garantierte Kommunikation zwischen zwei Systemen durch persistente Queues an.

Durch eine sehr modulare Architektur dieses Softwarelayers lassen sich die geforderten Dienste wie Publish/Subscribe–Funktionalität, Load Balancing und transparentes Routing von Prozessen auf einfache Weise realisieren. Mittels diesem Baukastensystem lässt sich die generische Infrastruktur an konkrete Anforderungen der Systemumgebung anpassen, so dass sich das System sowohl für verteilte Internet-Transaktionen als auch für aufwändige Berechnungen im Clusterumfeld und für die Konsistenzsicherung von Datenbeständen gleichermassen eignet.

Um der inhärenten Verteiltheit des Systems gerecht zu werden, bietet unser Prototyp auch die Möglichkeit einer Gesamtsicht des Systems, zum einen zur Prozessmodellierung, wo Prozesse gesamthaft definiert werden können und zum anderen durch eine Möglichkeit, die laufenden Prozess–Instanzen in einer Monitoring–Anwendung darzustellen und zu überwachen.

4 Verwandte Arbeiten

Es gibt verschiedene Ansätze, um Dynamik während einer Prozessausführung zu unterstützen. Einige dieser Ansätze, wie $ADEPT_{FLEX}$ [RD98] ändern die Definition von Prozessen und migrieren laufende Prozess–Instanzen vom alten Schema zum neuen. Dabei müssen die Instanzen oft durch regelbasierte Algorithmen umgeformt werden. Dieses Vorgehen wird vor allem im medizinischen Umfeld angewandt, wo es wichtig ist, dass alle Instanzen möglichst auf der aktuellsten Migrationsstufe sind [MR99]. Eine weitere Methode, um Dynamik von Prozessen zu unterstützen basiert darauf, während der Ausführung eines an sich statischen Prozess–Modells für jeden Schritt jeweils den geeignetsten Provider zu suchen. Man findet diese Ansätze, die dem OSIRIS-System in Bezug auf dessen Flexibilität ähneln, jedoch weniger Unterstützung für Ausführungsgarantien bieten, auch bei CrossFlow [GALH01] und bei *e*Flow [CIJ+00, CDS01].

Kommerziell verfügbare Systeme wie IBM MQSeries Workflows [MQS00] benutzen ebenfalls Publish/Subscribe Methoden und persistente Queues, allerdings nur um Aktivitäten auf entfernten System zu starten; die eigentliche Prozess-Koordination erfolgt immer zentral.

5 Zusammenfassung

Durch die Verwendung von Prozessen lassen sich aus bestehenden Diensten neue kombinierte Dienste zusammensetzen. Dadurch lassen sich auch komplexere Vorgänge automatisieren und korrekt koordinieren. Unser OSIRIS–System trägt durch eine verteilte Implementierung der Dynamik und der Autonomie der Dienstanbieter Rechnung. Durch Erweiterung der Konzepte für den zentralen Fall lassen sich Garantien für die Ausführung der Prozesse abgeben. Bedingt durch den modularen Aufbau bietet die OSIRIS–Infrastruktur eine Plattform, die sich durch Konfiguration in verschiedensten Anwendungsbereichen einsetzen lässt.

Literatur

[CDS01] F. Casati, U. Dayal, and M. Shan. E–Business Applications for Supply Chain Automation: Challenges. In *Proc. of the 17th International Conference on Data Engineering*, Heidelberg, Germany, April 2001.

[CIJ+00] F. Casati, S. Ilnicki, L. Jin, V. Krishnamoorthy, and M. Shan. Adaptive and Dynamic Service Composition in eFlow. In *Proc. Conf. on Advanced Information Systems Engineering*, Stockholm, 2000.

[GALH01] P. Grefen, K. Aberer, H. Ludwig, and Y. Hoffner. CrossFlow: Cross–Organizational Workflow Management for Service Outsourcing in Dynamic Virtual Enterprises. *IEEE Data Engineering Bulletin*, 24:52–57, 2001.

[MQS00] MQSeries Publish/Subscribe User's Guide. IBM Red Book, No. GC34-5269-05, 2000. IBM, International Business Machines Corporation.

[MR99] R. Müller and E. Rahm. Rule-Based Dynamic Modification of Workflows in a Medical Domain. In *Proceedings of Datenbanksysteme in Büro, Technik und Wissenschaft (BTW'99)*, Informatik Aktuell, pages 429–448, Freiburg, Germany, March 1999. Springer Verlag.

[RD98] M. Reichert and P. Dadam. $ADEPT_{flex}$ – Supporting Dynamic Changes of Workflows Without Losing Control. *Journal of Intelligent Information Systems*, 10(2):93–129, 1998.

[SABS02] H. Schuldt, G. Alonso, C. Beeri, and H.-J. Schek. Atomicity and Isolation for Transactional Processes. *ACM Transactions on Database Systems (TODS)*, 27(1), March 2002. To appear.

[SBG+00] H.-J. Schek, K. Böhm, T. Grabs, U. Röhm, H. Schuldt, and R. Weber. Hyperdatabases. In *Proceedings of the 1st International Conference on Web Information Systems Engineering (WISE'00)*, pages 14–23, Hong Kong, China, June 2000. IEEE Computer Society Press.

[SSS01] C. Schuler, H. Schuldt, and H.-J. Schek. Supporting Reliable Transactional Business Processes by Publish/Subscribe Techniques. In *Proceedings of the 2nd International Workshop on Technologies for E–Services (TES'01)*, pages 118–131, Rome, Italy, September 2001. Springer LNCS, Vol. 2193.

[W3C] W3C. Simple Object Access Protocol. `http://wwww.w3.org/TR/SOAP`.

[ZNBB94] A. Zhang, M. Nodine, B. Bhargava, and O. Bukhres. Ensuring Relaxed Atomicity for Flexible Transactions in Multidatabase Systems. In *Proc. ACM SIGMOD*, pages 67–78, Minneapolis, May 1994.