Universität Ulm

Fakultät für Informatik

Vermeidung von Überlastsituationen durch Replikation von Workflow-Servern in ADEPT

Thomas Bauer, Peter Dadam

Universität Ulm

Nr. 2000-09

Ulmer Informatik-Berichte

Juli 2000

Vermeidung von Überlastsituationen durch Replikation von Workflow-Servern in ADEPT

Thomas Bauer, Peter Dadam
Abteilung Datenbanken und Informationssysteme, Universität Ulm
{bauer, dadam}@informatik.uni-ulm.de, http://www.informatik.uni-ulm.de/dbis

In einem Workflow-Management-System (WfMS) entsteht durch die Ausführung von Workflows (WF) eine hohe Last. Dies betrifft sowohl die WF-Server, als auch das zugrunde liegende Kommunikationssystem. Im ADEPT-Projekt wurde ein Verteilungsmodell entwickelt, das hilft, eine Überlastung des Kommunikationssystems zu vermeiden. Dabei wird zur Reduzierung der Kommunikationslast die Kontrolle über einen WF ggf. von einem WF-Server zu einem anderen WF-Server in einem anderen Teilnetz verlagert. Bei den in [BD97, BD00] beschriebenen Verfahren wird angenommen, dass sich genau ein WF-Server in jedem Teilnetz befindet. Da für diesen die zu bewältigende Last möglicherweise zu hoch werden kann, wird in diesem Beitrag ein Verfahren zur Replikation von WF-Servern vorgestellt. Dieses erlaubt eine beliebige und veränderbare Aufteilung der Last auf die WF-Server eines Teilnetzes. Es zeichnet sich dadurch aus, dass keine zusätzliche Kommunikation benötigt wird.

1 Einleitung

WfMS ermöglichen die rechnerunterstützte Ausführung von Geschäftsprozessen in einer verteilten Systemumgebung. Der entscheidende Vorteil solcher Systeme ist, dass sie helfen, große Anwendungssysteme überschaubarer zu gestalten. Dazu wird der applikationsspezifische Code der verwendeten Anwendungen von der Ablauflogik des Geschäftsprozesses („Prozesslogik“) getrennt. Anstelle eines großen monolithischen Programmpakets erhält man nun einzelne Aktivitäten, welche die Anwendungsprogramme repräsentieren. Ihre Ablauflogik wird in einer separaten Kontrollflussdefinition festgelegt, welche die Ausführungsreihenfolge (Sequenz, Verzweigung, Parallelität, Schleifen) der einzelnen Aktivitäten bestimmt. Das WfMS sorgt dafür, dass nur solche Aktivitäten ausgeführt werden können, die der Ablauflogik zufolge zur Ausführung anstehen. Diese werden in die Arbeitslisten autorisierter Bearbeiter eingefügt. Welche Benutzer zur Bearbeitung einer bestimmten Aktivität autorisiert sind, wird meist durch eine Rolle festgelegt, die auch den entsprechenden Bearbeitern zugeordnet ist. Dabei ist es durchaus möglich, dass mehrere Benutzer ermittelt werden, die für die Bearbeitung einer bestimmten Aktivität in Frage kommen.

Ein gravierender Mangel heutiger WfMS ist ihre schlechte Skalierbarkeit bei wachsenden Benutzerzahlen. Die Last für die WF-Server und für das zugrunde liegende Kommunikationsnetzwerk kann sehr groß werden [BD99b, BD00]. Diese muss geeignet auf ausreichend viele Systemkomponenten verteilt werden, um Überlastsituationen zu vermeiden. In dieser Arbeit konzentrieren wir uns auf die Vermeidung von Überlastsituationen für die WF-Server. Dies soll allerdings so erfolgen, dass dadurch keine gravierenden Nachteile für das Kommunikationsverhalten entstehen, d.h., zusätzliche Kommunikation soll möglichst vermieden werden.

1.1 Verteilte Workflow-Ausführung

Wir betrachten eine unternehmensweite oder sogar unternehmensübergreifende WF-basierte Anwendung im operativen Einsatz. Durch die hohe Anzahl von Benutzern und gleichzeitig aktiven WF-Instanzen[1] wird eine sehr große Last erzeugt. Diese kann dazu führen, dass Komponenten des WfMS überlastet werden. Deshalb wird in ADEPT$_{distribution}$, der verteilten Variante des ADEPT-WfMS[2] [DKR+95, RD98], eine WF-Instanz nicht mehr nur von einem WF-Server kontrolliert, sondern sie wird partitioniert und abschnittsweise von verschiedenen Servern gesteuert [BD97] (vgl. Abb. 1). Wird bei der Ausführung von Instanzen dieses WF-Typs das Ende einer Partition erreicht, so wird die Kontrolle über diesen WF an den nächsten WF-Server weitergereicht (*Migration*). Damit dies möglich ist, muss eine Beschreibung des Zustands der WF-Instanz zu diesem Server transportiert werden.

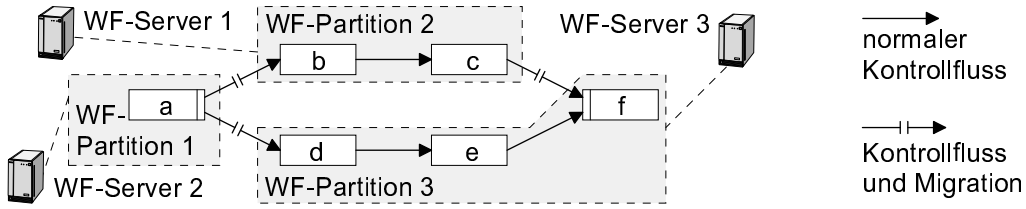

Abbildung 1 Partitionierung eines WF-Graphen und verteilte WF-Ausführung.

Eine solche Partitionierung des WF-Graphen und die zugehörigen Migrationen werden auch in anderen Ansätzen verwendet (z.B. [MWW+98]). Wir verfolgen in ADEPT zusätzlich das Ziel, die Kommunikationskosten zu minimieren. Der Grund dafür ist, dass konkrete Erfahrungen mit existierenden WfMS gezeigt haben, dass zwischen dem WF-Server und seinen Clients sehr viel kommuniziert wird und zum Teil auch große Datenmengen ausgetauscht werden. Dies kann dazu führen, dass das Kommunikationssystem überlastet wird. Deshalb wird in ADEPT der WF-Server für jede Aktivität so gewählt, dass der Kommunikationsaufwand minimiert wird. Dazu wird der WF-Server für eine Aktivität in der Regel so gewählt, dass er im Teilnetz[3] der Mehrzahl ihrer potentiellen Bearbeiter liegt. Dadurch wird in vielen Fällen teilnetzübergreifende Kommunikation zwischen dem WF-Server und seinen Clients vermieden. Außerdem werden die Antwortzeiten verbessert und die Verfügbarkeit erhöht, da bei der Ausführung von Aktivitäten kein Gateway oder WAN (Wide Area Network) zwischengeschaltet ist.

Bei diesem Ansatz wird also bereits zur Modellierungszeit eine (statische) Zuordnung von WF-Servern zu den Aktivitäten vorgenommen. Es gibt aber auch Fälle, in denen diese Vorgehensweise nicht ausreichend ist, um gute Resultate zu erzielen: Dies ist dann der Fall, wenn *abhängige Bearbeiterzuordnungen* (z.B. selber Bearbeiter wie Aktivität n) verwendet werden, weil die potentiellen Bearbeiter einer Aktivität dann vom Bearbeiter einer Vorgängeraktivität abhängen. Da sich die Menge der potentiellen Bearbeiter einer solchen Aktivität erst im Verlauf der WF-Ausführung ergibt, ist es vorteilhaft, ihren WF-Server erst zur Ausführungszeit der entsprechenden WF-Instanz festzulegen. Der Server kann dann in einem für die entsprechenden Bearbeiter günstigen Domain gewählt werden. Zu diesem Zweck wurden für ADEPT$_{distribution}$ so genannte *variable Serverzuordnungen* [BD99a, BD00] entwickelt. Dies sind Ausdrücke (z.B. "Server im Domain des Bearbeiters der Aktivität n"), die zur Ausführungszeit der WF-Instanz ausgewertet werden, um denjenigen WF-Server zu ermitteln, der die zugehörige Aktivitäteninstanz kontrollieren soll.

[1] In [KAGM96, SK97] werden Anwendungen beschrieben, bei denen die Zahl der Benutzer des WfMS bis auf einige zehntausend anwachsen kann oder mehrere zehntausend WF gleichzeitig im System sein können.

[2] ADEPT steht für Application Development Based on Encapsulated Pre-Modeled Process Templates.

[3] Ein Teilnetz wird zusammen mit den in ihm lokalisierten Clients und WF-Servern als *Domain* bezeichnet.

1.2 Problemstellung und Anforderungen

Der zentrale Aspekt von ADEPT$_{distribution}$ ist also, dass für jeden WF-Typ eine (bzgl. der bei der WF-Ausführung entstehenden Kommunikationslast) optimale Verteilung der Aktivitäten auf die WF-Server ermittelt wird. Das Teilnetz, in dem sich der WF-Server einer bestimmten Aktivitäteninstanz befindet, wird durch einen Serverzuordnungsausdruck vorgegeben. Um diese optimale Verteilung realisieren zu können, müssen alle Teilnetze über einen WF-Server verfügen. In den bisherigen Veröffentlichungen zu ADEPT$_{distribution}$ (z.B. [BD97, BD99a, BD00]) haben wir angenommen, dass sich in jedem Teilnetz genau ein WF-Server befindet. Das Problem dabei ist, dass dieser, wegen seiner begrenzten Leistungsfähigkeit, überlastet sein kann, obwohl das zugehörige Teilnetz noch nicht voll ausgelastet ist. Um dieses Problem zu lösen, könnte z.B. ein Hochleistungsrechner als WF-Server verwendet werden, oder es könnten mehrere WF-Server innerhalb eines Domains eingesetzt werden. Wie wir noch sehen werden, wird sich letzterer Ansatz als besser geeignet erweisen.

In dieser Arbeit soll ein Verfahren für die Verwendung mehrerer WF-Server innerhalb eines Domains entwickelt werden, welches die folgenden Anforderungen erfüllt:

1. Es muss möglich sein, die zu bewältigende Last auf beliebig viele WF-Server zu verteilen. Diese Lastaufteilung muss in einem beliebigen, vorzugebenden Verhältnis realisierbar sein. Es muss also z.B. vorgegeben werden können, dass sich die Last auf zwei zur Verfügung stehende Server im Verhältnis 3:2 aufteilt. Nur so kann bei einer beliebigen Menge an Server-Rechnern sichergestellt werden, dass keiner dieser Server überlastet wird.
2. Durch den Einsatz mehrerer WF-Server innerhalb eines Domains dürfen die durch das Verteilungsmodell von ADEPT erzielten Vorteile (z.B. die Reduzierung der Kommunikationslast) nicht konterkariert werden. Deshalb sollte das zu entwickelnde Verfahren möglichst keine (zusätzliche) Kommunikation zwischen den WF-Servern desselben Domains erfordern. Die Auswahl eines WF-Servers soll also ohne Abstimmungsprozess zwischen den verschiedenen Servern erfolgen.
3. Die Verwendung eines zentralen Servers zur Verteilung der Aufgaben verbietet sich, weil auch mit diesem kommuniziert werden müsste, und weil eine solche zentrale Komponente einen Flaschenhals darstellen und die Verfügbarkeit des WfMS beeinträchtigen würde. Es wird also ein verteiltes Verfahren zur Auswahl des Servers einer bestimmten Aktivitäteninstanz benötigt.
4. Um eine Integration in ADEPT zu ermöglichen, muss das Verfahren auch im Zusammenspiel mit variablen Serverzuordnungen funktionieren. Es muss also möglich sein, zur Festlegung des WF-Servers einer Aktivität, Ausdrücke (z.B. "Server im Domain des Bearbeiters der Aktivität n") zu verwenden, anstatt einen Server fest vorzugeben. Außerdem muss es im Falle einer dynamischen Änderung einer WF-Instanz [RD98, DRK00] ohne größeren Aufwand möglich sein, denjenigen Server eines Domains zu ermitteln, der die WF-Instanz tatsächlich kontrolliert (vgl. [RBD99]).
5. Es muss möglich sein, die Anzahl der WF-Server und das Verhältnis der Lastaufteilung zwischen ihnen im laufenden Betrieb zu verändern, ohne dass dieser dadurch beeinträchtigt wird.

Es ist also das Ziel, ein Verfahren zu entwickeln, das eine beliebige und veränderbare Verteilung der Last auf die WF-Server ermöglicht und dabei möglichst ohne Kommunikation auskommt.

1.3 Annahmen und Rahmenbedingungen

In diesem Abschnitt werden einige Aspekte der in diesem Beitrag zugrunde gelegten Systemumgebung erörtert. Da wir den operativen Einsatz von WfMS betrachten, kann von den folgenden Annahmen ausgegangen werden:

- Als WF-Server werden Rechner verwendet, die für diesen Zweck reserviert sind oder eine für diesen Zweck reservierte Bandbreite haben. Es ist also nicht anzunehmen, dass die Leistungsfähigkeit eines WF-Servers aufgrund anderer Aufgaben schwankt.
- Die Leistungsfähigkeit der WF-Server eines Domains ist etwas größer als die zu bewältigende Last erfordert (Reserve), so dass Lastschwankungen in einem gewissen Umfang ausgeglichen werden können.
- Bei Ausfall eines WF-Servers werden dessen Aufgaben von einem Backup-Server übernommen (vgl. [KAGM96]). Die Erhöhung der Verfügbarkeit ist kein primäres Ziel der in diesem Beitrag vorgestellten Verfahren.

Im nächsten Abschnitt werden einige prinzipiell mögliche Lösungsansätze diskutiert. Das Verfahren, mit dem der konkrete Server für eine Aktivitäteninstanz ausgewählt wird, wird in Abschnitt 3 vorgestellt. In Abschnitt 4 werden einige Fragestellungen betrachtet, welche die Veränderung einer gewählten Lastverteilung betreffen. Abschnitt 5 diskutiert die untersuchte Problemstellung im Kontext anderer verteilter WfMS-Ansätze. Der Beitrag schließt mit einer Zusammenfassung und einem Ausblick auf weitere Arbeiten.

2 Lösungsansätze

Im Folgenden wird untersucht, ob der in Abschnitt 1.2 angedachte Ansatz tatsächlich am besten geeignet ist, um die vorliegende Problemstellung zu lösen. Zu diesem Zweck werden zunächst verschiedene grundlegende Ansätze vorgestellt und analysiert.

2.1 Verwendung eines Hochleistungssystems

Die zu bewältigende Last wird bei diesem Ansatz nicht, wie in Abschnitt 1.2 skizziert, auf mehrere WF-Server verteilt. Stattdessen wird ein einziger, extrem leistungsstarker Rechner verwendet.

Der Vorteil dieses Ansatzes ist, dass die bisher für ADEPT entwickelten Verfahren unverändert weiterverwendet werden können. Allerdings entstehen für Hochleistungssysteme in der Regel sehr hohe Anschaffungskosten, so dass es sich lohnt, andere Alternativen zu betrachten. Außerdem kann auch ein solcher Rechner an seine Leistungsgrenzen stoßen.

2.2 Verändern der Serverzuordnungen

Bei diesem Ansatz befindet sich in jedem Domain weiterhin nur ein WF-Server. Um diesen bei Bedarf zu entlasten, können einige der von ihm kontrollierten Aktivitäten Servern anderer Domains zugeordnet werden. Dazu werden die Serverzuordnungen dieser Aktivitäten entsprechend verändert.

Durch diese Vorgehensweise werden die Kommunikationskosten erhöht, da ein Server ja gerade deshalb gewählt wurde, weil durch seine Verwendung die geringsten Kommunikationskosten anfallen. Dieses Vorgehen widerspricht damit der Kernidee von ADEPT. Außerdem ist das Verfahren überhaupt nicht anwendbar ist, wenn eine „globale Überlastung“ der WF-Server vorliegt, d.h., wenn insgesamt mehr Last zu bewältigen ist, als die aktuell vorhandenen WF-Server verarbeiten können. Da dieser Ansatz nicht vorsieht, dass mehrere WF-Server im selben Teilnetz eingesetzt werden, kann deren Gesamtkapazität nicht durch die Hinzunahme weiterer Server erhöht werden.

2.3 Aufspaltung eines Domains

Ist der WF-Server eines Domains überlastet, so wird dieser in mehrere Domains aufgeteilt. Die dadurch entstehenden Domains verfügen über jeweils einen WF-Server und ein eigenes Teilnetz. Bei diesem Ansatz werden also WF-Server und Teilnetze hinzugenommen. Eine Variante dieses Ansatzes ist die Bildung von logischen Domains, die zwar jeweils über einen eigenen WF-Server verfügen, denen aber dasselbe Teilnetz zugrunde liegt. Bei beiden Varianten werden die Benutzer des „Originaldomains“ auf die resultierenden Domains aufgeteilt.

Dieses Vorgehen ist immer dann empfehlenswert, wenn es eine „natürliche Zerlegung“ des Domains gibt, d.h., wenn der Domain in weitgehend disjunkte Teile (getrennte Aktivitäten und getrennte Benutzer) aufgespalten werden kann. Ist zudem auch das zugehörige Teilnetz stark belastet, so empfiehlt sich die Aufspaltung in Domains mit getrennten Teilnetzen. Dann wird allerdings durch die Veränderung der Domains ein Umbau des Kommunikationsnetzwerks notwendig.

Beide Varianten haben die folgenden gravierenden Nachteile: Häufig gibt es keine natürliche Zerlegung eines Domains, so dass keine sinnvolle Aufteilung der Benutzer auf die neu entstandenen Domains existiert. Außerdem ist es schwierig, die Last im gewünschten Verhältnis auf die Domains zu verteilen, weil die Belastung eines WF-Servers davon abhängt, wie viele und welche Bearbeiter und Aktivitäten seinem Domain zugeordnet sind. Da also nicht direkt vorgegeben werden kann, in welchem Verhältnis sich die Last auf die WF-Server verteilen soll, verletzt dies die Anforderung 1 (vgl. Abschnitt 1.2).

2.4 Mehrere Workflow-Server in einem Domain

Bei dem schon in Abschnitt 1.2 erwähnten Ansatz bleiben die Domains bezüglich ihrer Benutzer und Teilnetze unverändert. Bei Bedarf werden lediglich zusätzliche WF-Server in einem Domain verwendet. Wie der Server, der eine konkrete Aktivitäteninstanz kontrollieren soll, dann bestimmt werden kann, wird im Abschnitt 3 diskutiert.

Bei diesem Ansatz wird bei der Modellierung nur festgelegt, in welchem Domain der WF-Server einer Aktivität liegt. Welcher konkrete Server diese kontrollieren soll, ist ein physischer Aspekt, der bei der Modellierung nicht betrachtet wird. Der Einsatz mehrerer WF-Server je Domain ermöglicht im Prinzip eine beliebige Aufteilung der Last auf die WF-Server dieses Domains. Als positiver Nebeneffekt wird auch die Verfügbarkeit verbessert, da ein Benutzer jetzt von mehreren WF-Servern seines Domains bedient wird. Fällt einer von diesen vorübergehend aus, so kann der Benutzer zumindest mit den anderen weiterarbeiten.

Die Verwendung von mehreren WF-Servern in einem Domain kann allerdings Einfluss auf das Kommunikationsverhalten beim Aktualisieren der Arbeitslisten der Benutzer[4] haben. Ob die zu kommunizierende Datenmenge größer oder kleiner wird, hängt von der für das Aktualisieren der Arbeitslisten verwendeten Methode (vgl. [BD98]) ab. Wird die Arbeitsliste eines Bearbeiters in festen Zeitabständen aktualisiert, so wird durch das vorgestellte Verfahren eine höhere Anzahl von Kommunikationen nötig, da diese Aktualisierungen nun von mehreren Servern pro Domain durchgeführt werden müssen. Die kommunizierte Datenmenge verändert sich insgesamt gesehen jedoch nicht, da jeder WF-Server nur den von ihm verwalteten Teil der Gesamtarbeitsliste des Benutzers übertragen

[4] Ein Benutzer erhält in ADEPT seine Arbeitsliste von mehreren WF-Servern aus verschiedenen bzw. gleichen Domains. Dies ist aber für den Anwendungsentwickler transparent. Um dies zu erreichen, kapselt das API des WF-Client die verteilte Arbeitslistenverwaltung und bietet die Gesamtarbeitsliste des Benutzers an.

muss. Wird dagegen die Arbeitsliste eines Benutzers bei jeder Änderung stets sofort aktualisiert, so ändert sich die Anzahl der Kommunikationen nicht, da die Anzahl der Änderungen gleich bleibt. Da jeweils nur noch der zu diesem WF-Server gehörende Teil der Arbeitsliste (komplett) übertragen wird, wird die insgesamt kommunizierte Datenmenge sogar kleiner. Wird jeweils nur der neu hinzugekommene Eintrag übertragen (das „Delta“), so bleibt die kommunizierte Datenmenge unverändert. Zusammenfassend lässt sich also feststellen, dass die Verwendung von mehreren WF-Servern in einem Domain Auswirkungen auf das Kommunikationsverhalten beim Aktualisieren der Arbeitslisten haben kann. Diese sind nicht besonders gravierend, da die zur Aktualisierung der Arbeitslisten übertragene Datenmenge i.d.R. klein ist, verglichen mit der sonstigen Kommunikation eines WfMS (Start von Aktivitäten, Migrationen). Insbesondere ist es von der für die Aktualisierung verwendeten Methode abhängig, ob diese Auswirkungen positiv, negativ oder neutral sind.

Da die Verfahren aus Abschnitt 2.1 - 2.3 nicht geeignet sind, um das gegebene Problem zu lösen, wird nur der zuletzt vorgestellte Ansatz weiter verfolgt. Im nun folgenden Abschnitt wird beschrieben, wie bei diesem Verfahren die Auswahl eines konkreten WF-Servers für eine Aktivitäteninstanz ablaufen muss, um damit eine beliebige Verteilung der Last auf die Server erreichen zu können.

3 Auswahl des Workflow-Servers eines Domains

Im vorherigen Abschnitt haben wir uns dafür entschieden, den WF-Server eines Domains zu replizieren. Um diese Entscheidung umsetzen zu können, muss ein Verfahren entwickelt werden, welches für eine Aktivitäteninstanz einen Server des vorgesehenen Domains auswählt. Ein solches Verfahren muss die in Abschnitt 1.2 aufgestellten Anforderungen erfüllen, damit es nicht der Zielsetzung von ADEPT entgegenwirkt. Insbesondere muss es eine beliebige Lastaufteilung ermöglichen und die Konzepte von ADEPT, wie z.B. variable Serverzuordnungen [BD00], unterstützen. Außerdem sollte es möglichst ohne zusätzliche Kommunikationen realisiert werden. Um die Erfüllung der Anforderung 5, welche die Änderbarkeit der Lastaufteilung im laufenden Betrieb fordert, kümmern wir uns im Abschnitt 4. Im Folgenden werden nun einige mögliche Vorgehensweisen untersucht und ein Verfahren entwickelt, welches die Anforderungen 1-4 erfüllt.

3.1 Statische Festlegung des Workflow-Servers zur Modellierungszeit

Eine einfache Möglichkeit, die Last auf mehrere Server aufzuteilen, ist, den WF-Server für eine Aktivität (so wie schon bisher den Domain) im WF-Modell explizit festzulegen (z.B. Aktivität k wird von Server 2 des Domains 7 kontrolliert). Unterschiedliche Aktivitätentypen (aus unterschiedlichen Partitionen) können so von verschiedenen WF-Servern eines Domains kontrolliert werden.

Da konkrete Server für die Aktivitäten vorgegeben werden, ist es kaum möglich, ein gewünschtes Verhältnis der Lastaufteilung zu realisieren. Wie bei den Verfahren aus Abschnitt 2.3 ist damit die Anforderung 1 verletzt. Außerdem kann das Verfahren nicht in Verbindung mit variablen Serverzuordnungen eingesetzt werden, weil diese durch einen Ausdruck, der erst zur Ausführungszeit ausgewertet wird, den Domain einer Aktivität festlegen (z.B. "Server im Domain des Bearbeiters der Aktivität 1"). Eine explizite Festlegung eines WF-Servers innerhalb des resultierenden Domains ist deshalb nicht möglich.

3.2 Lastabhängige Auswahl des Workflow-Servers

Eine Aufteilung der Last auf die WF-Server lässt sich auch erreichen, indem derjenige WF-Server ausgewählt wird, der aktuell am wenigsten belastet ist (gemessen an seiner Soll-Last). Die Identifikation des entsprechenden WF-Servers erfordert, dass Lastinformation über dessen Domain verfügbar ist. Um diese zur Verfügung zu stellen, sind prinzipiell zwei Vorgehensweisen denkbar:

1. Die Lastinformation wird (periodisch oder falls möglich „Huckepack“ mit sonstigen Kommunikationen) an alle WF-Server des WfMS verteilt. Soll eine Migration durchgeführt werden, so ist die Lastinformation lokal vorhanden, so dass der Zielserver ausgewählt werden kann.
2. Die Lastinformation wird ausschließlich einem Dispatcher im jeweiligen Domain gemeldet. Soll der WF-Server für eine Aktivitäteninstanz ausgewählt werden, so führt der Dispatcher des jeweiligen Domains diese Auswahl durch.

Die erste Methode erfordert einen höheren Aufwand bei der Verteilung der Lastinformation, weil jeder WF-Server die aktuelle Belastung jedes anderen Servers kennen muss. Bei der zweiten Methode wird der Aufwand für die Verteilung der Lastinformation reduziert, da nur der Dispatcher diese Informationen benötigt. Andererseits wird ein größerer Aufwand für die Auswahl des WF-Servers nötig, da zuerst mit dem jeweiligen Dispatcher kommuniziert werden muss. Außerdem stellt der Dispatcher eine zentrale Komponente dar, welche die Verfügbarkeit seines Domains verschlechtert, da er bei jeder Migration benötigt wird. Beide Varianten verletzen also die Anforderung 2, die zweite Methode zusätzlich die Anforderung 3.

Der Vorteil von lastabhängigen Verfahren ist, dass Schwankungen in der Belastung der WF-Server ausgeglichen werden können. Diese können aufgrund WfMS-externer Ereignisse auftreten, oder weil die von einem Server verwalteten WF-Instanzen in einem bestimmten Zeitintervall besonders viel Last erzeugen. Der entscheidende Nachteil beider Verfahren ist der zusätzlich entstehende Aufwand für Kommunikation und Synchronisation. Dieser führt dazu, dass die von den WF-Servern bewältigte Last nicht mehr linear zu deren Leistungsfähigkeit wächst. Außerdem ist – wie schon von Scheduling-Verfahren bekannt ist [Gos91] – für den Erfolg eines lastabhängigen Verfahrens die Qualität der Lastinformation und eine geeignet gewählte Frequenz für den Lastinformationsaustausch äußerst kritisch. Dies gilt umso mehr, weil WF eine komplexe interne Struktur aufweisen und für mehrere Wochen am gewählten Server verweilen können. Zusammenfassend lässt sich also feststellen, dass lastabhängige Verfahren ein (akademisch) interessanter Ansatz sind, da sie potentiell die Möglichkeit bieten, Lastschwankungen auszugleichen. Sie sind aber schwer zu realisieren und erzeugen selbst zusätzliche Kommunikationslast.

3.3 Zufällige Auswahl des Workflow-Servers zur Laufzeit

Wenn für eine WF-Instanz mit der Ausführung einer Partition (in dem vorgegebenen Domain) begonnen wird (WF-Start oder Migration), dann kann zufällig ein WF-Server dieses Domains ausgewählt werden. Dazu wird eine Zufallszahl z aus dem Intervall $[0, 1)$ berechnet. Dieses Intervall wird in disjunkte Teilintervalle zerlegt, die den WF-Servern des Domains zugeordnet sind. Für die fragliche Partition wird derjenige WF-Server gewählt, in dessen Teilintervall z fällt. Die Wahrscheinlichkeit, dass ein bestimmter WF-Server gewählt wird (und damit sein Anteil an der Gesamtlast), entspricht der Länge seines Intervalls.

Ein solches Verfahren hat einige sehr schöne Eigenschaften: Es ermöglicht eine beliebige Aufteilung der Last. Diese ist zudem, durch eine Veränderung der Teilintervalle, leicht änderbar. Das Verfahren

ist auch für variable Serverzuordnungen anwendbar und es ist sogar möglich, die Last für die Steuerung eines einzigen Aktivitätentyps auf mehrere WF-Server zu verteilen. Wegen der großen Anzahl von WF-Instanzen ist es sehr unwahrscheinlich, dass, aufgrund von Unzulänglichkeiten des Zufallszahlengenerators, deutlich von der geplanten Lastverteilung abgewichen wird.

Das Verfahren hat allerdings einen entscheidenden Nachteil: Es führt zu einem Problem bei der Zusammenführung paralleler WF-Zweige. Dies soll an dem in Abb. 2 dargestellten Beispiel erläutert werden: Angenommen, für die aktuelle WF-Instanz findet die Migration $M_{b,c}$ vor $M_{d,e}$ statt. Dann wird bei der Ausführung von $M_{b,c}$ der konkrete WF-Server innerhalb des Domains 3 festgelegt. Dieser ist aber dem WF-Server des Domains 2 nicht ohne weiteres bekannt. Bei einer naiven Implementierung dieses Verfahrens könnte es deshalb vorkommen, dass dieser bei der Migration $M_{d,e}$ einen anderen WF-Server aus dem Domain 3 (zufällig) auswählt, was zu einem Problem bei der Zusammenführung der parallelen Zweige an der Aktivität f führen würde. Eine Abstimmung des Zielservers zwischen den verschiedenen Startservern der Migration erfordert jedoch zusätzliche Kommunikation, was der Anforderung 2 widerspricht.

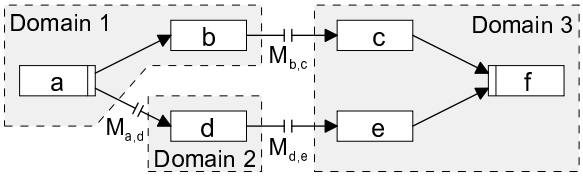

Abbildung 2 Problem bei der Zusammenführung paralleler Zweige bei zufälliger Serverwahl.

3.4 Pseudo-zufällige Auswahl des Workflow-Servers zur Laufzeit

Wir wollen die positiven Eigenschaften des vorherigen Verfahrens erhalten, jedoch das Problem bei der Zusammenführung paralleler Zweige vermeiden. Dies gelingt durch folgenden Trick: Der WF-Server wird nicht rein zufällig gewählt, sondern so, dass für eine WF-Instanz in einem Domain stets derselbe WF-Server ermittelt wird. Dazu wird ein WF-instanzspezifisches Datum (z.B. die WF-ID) mit einem Pseudo-Zufallszahlengenerator auf ein $z \in [0, 1)$ abgebildet. Dadurch ergibt sich für eine WF-Instanz stets dasselbe z und damit derselbe WF-Server.

Alle Vorteile des vorherigen Verfahren bleiben erhalten: Es wird eine beliebige Aufteilung der Last auf die WF-Server ermöglicht, alle Konzepte von ADEPT (inkl. variabler Serverzuordnungen) werden unterstützt und das Verfahren kommt ohne zusätzliche Kommunikation aus. Die Probleme bei der Zusammenführung paralleler Zweige werden vermieden, da innerhalb eines Domains für alle parallelen Zweige einer WF-Instanz derselbe Server ermittelt wird. Weil es ebenfalls keine Kommunikation erfordert, erfüllt das Verfahren die Anforderungen 1-4 aus Abschnitt 1.2. Für die Praxis ist zu erwarten, dass es besser funktioniert als lastabhängige Verfahren. Die Gründe dafür sind, dass keine zusätzliche Kommunikation notwendig wird, und dass das zu erwartende Systemverhalten leichter abschätzbar ist. Da die WF-Server über eine Lastreserve verfügen, spielt eine leichte Ungleichverteilung der Last keine große Rolle. Auch deshalb wird kein Verfahren benötigt, das eine solche Ungleichverteilung aktiv ausgleicht. Aus diesen Gründen wird das pseudo-zufällige Verfahren weiter verfolgt. Im Folgenden wird noch geklärt, wie die geforderte pseudo-zufällige Aufteilungsfunktion realisiert und die von den WF-Servern benötigte Information bereitgestellt werden kann. In Abschnitt 4 wird dann diskutiert, wie zusätzlich die Anforderung 5 (Änderbarkeit der Lastaufteilung) erfüllt werden kann.

3.4.1 Realisierung der Aufteilungsfunktion

Bei dem Verfahren wird ein instanzspezifisches Datum der WF-Instanz *Inst* verwendet, um z.B. bei einer Migration einen WF-Server des Zieldomains D auszuwählen. Diese Berechnung übernimmt die domainspezifische Aufteilungsfunktion $g^D(\mathit{Inst})$ (siehe Algorithmus 1). Das unveränderliche instanzspezifische Datum *InstDat* (z.B. die WF-ID) der WF-Instanz *Inst* wird mit Hilfe eines Pseudo-Zufallszahlengenerators auf einen Wert z im Intervall $[0, 1)$ abgebildet. Die Funktion $f^D(z)$ bildet diesen Wert z auf einen Server s_i des Domains D ab. Durch die Wahl dieser Funktion f lässt sich die Lastverteilung auf die WF-Server steuern. Da ein Pseudo-Zufallszahlengerator ausgehend vom selben Startwert *InstDat* stets dasselbe Ergebnis z berechnet, ergibt sich für eine WF-Instanz *Inst* stets derselbe WF-Server s_i im Domain D.

Algorithmus 1 (Berechnung der Aufteilungsfunktion $g^D(\mathit{Inst})$)
input
Inst: WF-Instanz, für die der WF-Server ermittelt werden soll
D: Domain, aus dem ein WF-Server gewählt werden soll
result
logische Server-ID des für die WF-Instanz *Inst* ausgewählten WF-Servers aus dem Domain D
begin
InstDat $= instanzspezifisches\_Datum(\mathit{Inst})$;
$z = pseudo\_random(\mathit{InstDat})$;
Funktion $f^D(z)$ definiert durch $a_1^D \ldots a_{n-1}^D$ und $s_1 \ldots s_n$:
case $z \in [0, a_1^D)$: **return** s_1;
case $z \in [a_1^D, a_2^D)$: **return** s_2;
...
case $z \in [a_{n-1}^D, 1)$: **return** s_n;
end.

Die Wahrscheinlichkeit für die Auswahl des WF-Servers s_i durch den Algorithmus 1 entspricht der Länge des zugehörigen Intervalls $[a_{i-1}, a_i)$. Dieser Aussage liegt die Annahme zugrunde, dass die Werte von z über $[0, 1)$ gleichverteilt sind. Dies kann durch den Einsatz eines „guten“ Pseudo-Zufallszahlengenerators bei der Berechnung der Funktion $pseudo\_random$ erreicht werden. In Algorithmus 1 kann als instanzspezifisches Datum die z.B. die ID der WF-Instanz verwendet werden. Stattdessen kann auch die Startzeit der WF-Instanz (oder ein speziell für diesen Zweck generierter Zufallswert) gewählt werden. Wird z.B. von der Startzeit nur der Wert für die Millisekunden als *InstDat* verwendet, so ist es realistisch anzunehmen, dass die Werte von *InstDat* gleichmäßig über das Intervall $[0, 1000)$ verteilt sind. Deshalb ist in diesem Fall die Funktion $pseudo\_random(\mathit{InstDat}) := \mathit{InstDat}/1000$ ausreichend, um eine Gleichverteilung von z über $[0, 1)$ zu erreichen.

3.4.2 Verteilung der benötigten Information

Beim Start von WF-Instanzen und bei deren Migration muss entschieden werden können, welcher Server s des Domains D für die WF-Instanz *Inst* verwendet werden soll. Dazu muss die Funktion $g^D(\mathit{Inst})$ allen WF-Servern des WfMS bekannt sein. Diese Funktion berechnet eine logische Server-ID, die mit einer anderen Funktion $h(s)$ auf eine physische Rechneradresse (z.B. IP-Adresse) abgebildet wird. Auch $h(s)$ muss allen Servern bekannt sein. Da diese beiden Funktionen selten geändert werden, werden sie auf allen WF-Servern repliziert (ähnlich wie die WF-Vorlagen). Aspekte, welche die Änderung dieser Funktionen betreffen, werden im nächsten Abschnitt diskutiert. Da solche Ände-

rungen eher selten auftreten, fallen die durch die Replikation entstehenden Kommunikationskosten nicht ins Gewicht.

4 Änderung der Lastaufteilung

Das in Abschnitt 3.4 vorgestellte, pseudo-zufällige Verfahren erfüllt die Anforderungen 1-4. In der oben beschriebenen Form wird die Anforderung 5, dass die Aufteilung der Last veränderbar sein muss, noch nicht erfüllt. Dies ist aber notwendig, damit eine existierende Überlastung behoben oder eine Systemkonfiguration verändert werden kann. Die Aufteilungsfunktion zu verändern ist allerdings nicht unproblematisch, weil eine WF-Instanz stets vom selben Server innerhalb eines Domain kontrolliert werden muss, um das in Abschnitt 3.3 beschriebene Problem beim Zusammenführen paralleler Zweige zu vermeiden. Ungünstigerweise erfordert eine Veränderung der Lastaufteilung jedoch eine Veränderung dieser Zuordnung. Im Folgenden wird untersucht, wie Server ausgetauscht, hinzugefügt und weggenommen werden können und wie die Lastaufteilung verändert werden kann, so dass die Anforderung 5 erfüllt wird, ohne dadurch die Anforderungen 1-4 zu verletzen.

4.1 Physische Server austauschen

Soll der WF-Server mit der logischen ID s_{change} durch einen anderen ersetzt werden, so wird dieser durch Algorithmus 2 zuerst gestartet. Nachdem die Möglichkeit zur Migration zu dem von der Änderung betroffenen Server gesperrt wurde, wird die Abbildungsfunktion $h(s)$ der logischen Server-IDs auf die Rechneradressen verändert. Dann werden die WF-Instanzen, welche vom stillzulegenden Server kontrolliert werden, zum neuen Server transportiert. Die neue Funktion $h_{(n+1)}(s)$, welche die bisher gültige Funktion $h_{(n)}(s)$ ersetzt, wird repliziert, so dass sie jedem Server des WfMS bekannt ist. Die Änderung der Funktion $h(s)$ bewirkt, dass am alten WF-Server zukünftig keine Migrationen mehr eingehen. Jetzt können die Sperren aufgehoben werden, so dass auch wieder Migrationen zum Server s_{change} stattfinden können. Schließlich kann der zu ersetzende Server gestoppt werden.

Algorithmus 2 (Austauschen eines WF-Servers)
input
s_{change}: logische ID des Servers, welcher ersetzt werden soll
adr: physische Adresse des Servers, welcher die logische ID s_{change} erhalten soll
$h_{(n)}(s)$: bisherige Abbildungsfunktion logischer IDs von WF-Servern auf deren physische Adresse
begin
starte den neuen WF-Server adr;
sperre (bei allen WF-Servern) die Migrationen zum (bisherigen) Server s_{change};
// ordne s_{change} die neue Server-ID zu und übernehme die anderen Einträge
$h_{(n+1)}(s_{change}) := adr$; $\forall s \neq s_{change}$: $h_{(n+1)}(s) := h_{(n)}(s)$;
transferiere alle WF-Instanzen vom Server $h_{(n)}(s_{change})$ zum Server $h_{(n+1)}(s_{change})$;
repliziere $h_{(n+1)}(s)$ an alle WF-Server;
gebe Migrationen zum WF-Server s_{change} wieder frei;
stoppe den zu deaktivierenden WF-Server $h_{(n)}(s_{change})$;
end.

Eigentlich müsste das Sperren, das Replizieren von $h_{(n+1)}(s)$ und das Freigeben der Sperren in einer verteilten Transaktion stattfinden. Dadurch wären aber Migrationen zum Server s_{change} nicht möglich, solange die Transaktion nicht bei allen Servern beendet wurde. Der Server s_{change} würde also durch

den Ausfall eines beteiligten Servers während der Ausführung von Algorithmus 2 blockiert werden. Außerdem würde ein solches Vorgehen die Ausführung eines teuren 2-Phasen-Commit-Protokolls erforderlich machen. Um diese Nachteile zu vermeiden, wird für den Algorithmus 2 und die nachfolgend beschriebenen Algorithmen der folgende Trick verwendet: Die Möglichkeit zur Migration wird weiterhin bei allen WF-Servern gesperrt. Nach Ausführung des entsprechenden „Sperrbefehls“ können bei s_{change} also keine neuen Migrationen mehr eingehen. Die Replikation der Funktion $h_{(n+1)}(s)$ wird nun mit dem Freigeben der Sperre zu einer einzigen Transaktion zusammengefasst. Sobald nun ein Server die neuen Daten erfolgreich (lokal) gespeichert hat, hebt er die Sperre für Migrationen zu s_{change} auf, so dass Migrationen zu s_{change} (auf Basis der neuen Funktion) von ihm durchgeführt werden können. Durch diese Vorgehensweise wird zwar die Sperre nicht bei allen Servern gleich lange gehalten, die Möglichkeit zur Migration ist aber entweder noch gesperrt oder es wird schon die neue Funktion $h_{(n+1)}(s)$ verwendet. Deshalb können keine Inkonsistenzen durch die Verwendung der unterschiedlichen Funktionen $h_{(n)}(s)$ und $h_{(n+1)}(s)$ entstehen. Der große Vorteil dieser Vorgehensweise ist, dass jeder WF-Server sofort nach der erfolgreichen (lokalen) Speicherung der Daten wieder ohne Einschränkung weiterarbeiten kann, auch wenn noch nicht alle Server die Änderung der Abbildungsfunktion (z.B. wegen eines lokalen Systemausfalls) vollzogen haben.

4.2 Lastverteilung ändern

Wenn zwar weiterhin dieselben WF-Server verwendet werden sollen, aber die Verteilung der Last zwischen ihnen geändert werden soll, so muss die bisher gültige Aufteilungsfunktion $g^D_{(n)}(\mathit{Inst})$ durch eine neue Aufteilungsfunktion $g^D_{(n+1)}(\mathit{Inst})$ ersetzt werden. Durch diese Veränderung können Probleme bei der Zusammenführung paralleler Zweige einer WF-Instanz entstehen: Wenn einzelne Zweige einer WF-Instanz die alte Funktion verwenden und andere die neue, so kann es vorkommen, dass eine WF-Instanz von verschiedenen WF-Servern eines Domains kontrolliert wird. Dies wollen wir jedoch ausschließen, um Kommunikation zwischen WF-Servern desselben Domains zu vermeiden (Anforderung 2). Zu diesem Zweck kann eines der folgenden Verfahren verwendet werden:

1. In Algorithmus 3 wird die neue Aufteilungsfunktion $g^D_{(n+1)}(\mathit{Inst})$ des Domains D mit einem Zeitstempel $T^D_{(n+1)}$ versehen, der mit ihr repliziert wird. Die neue Funktion $g^D_{(n+1)}(\mathit{Inst})$ wird dann nur für WF-Instanzen *Inst* verwendet, für die gilt: $Startzeitpunkt(\mathit{Inst}) > T^D_{(n+1)}$. Während der Algorithmus 3 ausgeführt wird, ist die Möglichkeit zur Migration in den Domain D gesperrt. Für WF-Instanzen, die nach dem Start von Algorithmus 3 erzeugt wurden, ist damit zum Zeitpunkt einer etwaigen Migration in den Domain D die neue Funktion $g^D_{(n+1)}(\mathit{Inst})$ und der zugehörige Zeitstempel bekannt, so dass die Migration auch wirklich auf Basis der neuen Funktion $g^D_{(n+1)}(\mathit{Inst})$ durchgeführt werden kann.

Algorithmus 3 (1. Verfahren zur Änderung von $g^D(\mathit{Inst})$)
input
D: Domain, in dem die Lastaufteilung verändert werden soll
$g^D_{(n+1)}(\mathit{Inst})$: neue Aufteilungsfunktion für den Domains D
begin
sperre (bei allen WF-Servern) die Migrationen in den Domain D;
$T^D_{(n+1)} = time()$; // Zeitstempel auf die aktuelle Zeit setzen
repliziere $(g^D_{(n+1)}(\mathit{Inst}), T^D_{(n+1)})$;
gebe Migrationen in den Domain D wieder frei;
end.

2. Bei dem in Algorithmus 4 dargestellten Verfahren wird die Menge $A^D_{(n+1)}$ der IDs von WF-Instanzen berechnet, die zur Zeit der Änderung in der Datenbank eines WF-Servers s des Domains D gespeichert sind und für die sich durch die Änderung der WF-Server ändert. Für eine WF-Instanz *Inst* mit $ID(Inst) \in A^D_{(n+1)}$ wird nicht die neue Funktion $g^D_{(n+1)}(Inst)$ verwendet, sondern weiterhin die alte Funktion $g^D_{(n)}(Inst)$ (bzw. $g^D_{(n-1)}(Inst)$, falls zusätzlich $ID(Inst) \in A^D_{(n)}$ gilt, usw.). Die Menge $A^D_{(n+1)}$ enthält also die IDs derjenigen WF-Instanzen, die zum Zeitpunkt der Änderung von einem WF-Server des Domains D kontrolliert werden und für die sich durch $g^D_{(n+1)}$ der Server ändern würde. Für diese WF-Instanzen wird weiterhin die „alte" Aufteilungsfunktion verwendet, so dass der entsprechende WF-Server alle parallelen Zweige kontrolliert. Deshalb entsteht für diese Instanzen kein Problem beim Zusammenführen paralleler Zweige. Für alle anderen WF-Instanzen ($ID(Inst) \notin A^D_{(n+1)}$) wird ausschließlich die „neue" Funktion $g^D_{(n+1)}$ verwendet, so dass auch hier kein Problem auftreten kann.

Algorithmus 4 (2. Verfahren zur Änderung von $g^D(Inst)$)
input
D: Domain, in dem die Lastaufteilung verändert werden soll
$g^D_{(n+1)}(Inst)$: neue Aufteilungsfunktion des Domains D
s_D: Menge der WF-Server des Domains D
begin
sperre (bei allen WF-Servern) die Migrationen in den Domain D;
$activeWFs = \bigcup_{s \in S_D} \{Inst \mid$ WF-Instanz *Inst* ist am WF-Server s aktiv$\}$;
// im Domain D aktive WF-Instanzen, für die sich der Server verändern würde:
$A^D_{(n+1)} = \{ID(Inst) \mid Inst \in activeWFs \wedge Server^D(Inst) \neq g^D_{(n+1)}(Inst)\}$;
repliziere $(g^D_{(n+1)}(Inst), A^D_{(n+1)})$;
gebe Migrationen in den Domains D wieder frei;
end.

Der Vorteil des ersten Verfahrens ist, dass es einen sehr geringen Aufwand erfordert. Allerdings wird die neue Funktion $g^D_{(n+1)}(Inst)$ erst dann für alle WF-Instanzen verwendet, wenn keine WF-Instanzen mehr existieren, die älter als diese Funktion sind. Da WF-Instanzen häufig mehrere Wochen laufen, kann es entsprechend lange dauern, bis die neu eingestellte Lastverteilung tatsächlich erreicht wird. Das zweite Verfahren erfordert wegen der Berechnung und Replikation von $A^D_{(n+1)}$ einen wesentlich größeren Aufwand als das erste Verfahren. Es hat aber den Vorteil, dass auch für schon lange laufende WF-Instanzen der Migrationszielserver auf Basis der neuen Aufteilungsfunktion ermittelt wird. Welches der beiden Verfahren besser geeignet ist, ist von der jeweiligen Situation abhängig. Soll die Lastaufteilung lediglich längerfristig angepasst werden, so ist die 1. Variante ausreichend. Sollen aber die Auswirkungen der Änderung möglichst ab sofort beobachtet und bewertet werden, so ist die lange Verzögerung nicht akzeptabel. Dann muss die 2. Variante verwendet werden. Die dabei entstehenden Kosten relativieren sich, da Veränderungen der Lastaufteilung i.d.R. doch eher selten sind.

4.3 Anzahl der Server eines Domains erhöhen

Soll die Anzahl der WF-Server eines Domains D erhöht werden, so muss eine neue logische Server-ID s_{new} definiert werden. Nachdem der Algorithmus 5 den neuen WF-Server gestartet hat, wird dessen logische ID s_{new} auf dessen physische Adresse adr_{new} abgebildet. Dies geschieht durch die Änderung und Replikation der Funktion $h(s)$. Für diesen Vorgang sind keine Sperren erforderlich, weil s_{new} noch kein Intervall der Aufteilungsfunktion $g^D(Inst)$ zugeordnet ist; der Server s_{new} wird also

überhaupt noch nicht verwendet. Damit er jedoch in Zukunft WF-Instanzen kontrollieren kann, muss zusätzlich die Funktion $g^D(\mathit{Inst})$, wie im vorherigen Abschnitt beschrieben, geändert werden. Dem neuen WF-Server muss also ein Intervall der Aufteilungsfunktion zugeordnet werden.

Algorithmus 5 (Neuen WF-Server einführen)
input
s_{new}: logische ID des neuen WF-Servers
adr_{new}: physische Adresse des neuen Servers
$h_{(n)}(s)$: aktuelle Funktion, die logische Server-IDs auf physische Server-Adressen abbildet
$g^D_{(n'+1)}(\mathit{Inst})$: neue Aufteilungsfunktion des Domains D, in der auch s_{new} berücksichtigt wird
begin
starte den WF-Server adr_{new};
$h_{(n+1)}(s_{new}) := adr_{new}$; $\forall s \neq s_{new}$: $h_{(n+1)}(s) := h_{(n)}(s)$;
repliziere $h_{(n+1)}(s)$;
ändere $g^D_{(n')}(\mathit{Inst})$ in $g^D_{(n'+1)}(\mathit{Inst})$ mit Algorithmus 3 oder 4;
end.

4.4 Anzahl der Server eines Domains reduzieren

Wenn die Anzahl der WF-Server des Domains D verringert werden soll, so verhindert der Algorithmus 6 zuerst durch eine Sperre weitere Migrationen zu dem stillzulegenden Server s_{old}. Dann werden alle bei ihm vorhandenen WF-Instanzen zu dem Server s_{new} transferiert, welcher dessen Aufgaben (vorläufig) übernimmt. Damit zukünftig keine Migrationen an dem stillzulegenden Server mehr eingehen, wird die logische Server-ID s_{old} auf die physische Adresse des übernehmenden WF-Servers „umgelenkt“, indem die Funktion $h(s)$ verändert wird. Nun wird die neue Funktion $h_{(n+1)}(s)$ repliziert und die Sperren werden freigegeben. Da der stillzulegende Server keine WF-Instanzen mehr kontrolliert und auch keine Migrationen mehr bei ihm eingehen können, wird er gestoppt. Um auch die logische ID des stillgelegten Servers auslaufen zu lassen, sollte nach Beendigung von Algorithmus 6 mit den Verfahren aus Abschnitt 4.2 die Aufteilungsfunktion $g^D(\mathit{Inst})$ so verändert werden, dass s_{old} kein Intervall mehr zugeordnet ist. Im Zuge dieser Veränderung kann zusätzlich die bisher von s_{old} bewältigte Last geeignet auf andere WF-Server verteilt werden, indem die Intervalle entsprechend angepasst werden.

Algorithmus 6 (WF-Server stilllegen)
input
s_{old}: logische ID des stillzulegenden WF-Servers
s_{new}: logische ID des Servers, der die Aufgaben von s_{old} übernehmen soll
$h_{(n)}(s)$: aktuelle Funktion, die logische Server-IDs auf physische Server-Adressen abbildet
begin
sperre (bei allen WF-Servern) die Migrationen zum Server s_{old};
$h_{(n+1)}(s_{old}) := h_{(n)}(s_{new})$; $\forall s \neq s_{old}$: $h_{(n+1)}(s) := h_{(n)}(s)$;
transferiere alle WF-Instanzen vom Server $h_{(n)}(s_{old})$ zum Server $h_{(n)}(s_{new})$;
repliziere $h_{(n+1)}(s)$;
gebe Migrationen zum WF-Server s_{old} wieder frei;
stoppe den WF-Server $h_{(n)}(s_{old})$;
end.

Wie wir gesehen haben, ist es also nicht nur möglich, die Aufteilungsfunktion der WF-Instanzen auf die WF-Server effizient und mit guten statistischen Eigenschaften zu realisieren, sie kann außerdem

im laufenden Betrieb beliebig verändert werden. Damit erfüllt das oben beschriebene Verfahren nicht nur – wie schon in Abschnitt 3 festgestellt wurde – die Anforderungen 1-4, sondern zusätzlich auch die Anforderung 5 aus Abschnitt 1.2.

5 Diskussion

Unseres Wissens gibt es bisher keine Arbeiten, die sich mit dem Problem der Überlastung des WF-Servers eines Domains befassen. Der vorliegende Beitrag untersucht also erstmalig die durch die Verwendung mehrerer WF-Server innerhalb desselben Domains resultierenden Probleme und die Fragestellung, wie dann ein geeigneter Server ausgewählt werden kann. Der nun folgende Abschnitt bietet einen kurzen Überblick über Verteilungsmodelle für WfMS. Eine detaillierte Diskussion zu diesem Thema findet sich in [BD98, BD99b]. Außerdem wird nun diskutiert, wie das vorgestellte Verfahren bei verschiedenen Verteilungsstrategien eingesetzt werden kann.

Einige Forschungsprototypen, die sich nicht primär um Skalierbarkeit kümmern (z.B. Panta Rhei [EG96], WASA [WHKS98]), und die meisten kommerziellen Systeme verwenden einen zentralen WF-Server. Da dieser einen potentiellen Flaschenhals darstellt, können solche WfMS nicht als skalierbar bezeichnet werden. Bei voll verteilten Ansätzen (z.B. Exotica/FMQM [AMG+95] und INCAS [BMR96]) übernimmt der Rechner jedes Benutzers Serverfunktionalität. Da es keine WF-Server im eigentlichen Sinn gibt, bei denen sich die Last konzentriert, können auch keine WF-Server überlastet werden. Deshalb besteht keine Notwendigkeit für die Verwendung des vorgestellten Verfahrens. Im Exotica-Projekt wurde ein Ansatz entwickelt, bei dem der WF-Server (Cluster) für eine WF-Instanz bei deren Start zufällig ausgewählt wird [AKA+94]. In diesem Cluster verbleibt der WF für seine gesamte Laufzeit, es gibt also keine Migrationen. Dies entspricht einem zentralen Ansatz mit repliziertem WF-Server. Die Serverwahl findet zufällig statt (vgl. Abschnitt 3.3). Da eine WF-Instanz den Cluster nie wechselt, treten keine Probleme bei der Zusammenführung paralleler Zweige auf, so dass die zufällige Serverwahl ausreichend ist.

Bei zahlreichen Ansätzen werden wie bei ADEPT mehrere WF-Server verwendet, wobei ebenfalls der Wechsel des Servers möglich ist. In MENTOR [MWW+98] und WIDE [CGS97] wird der WF-Server nahe bei den potentiellen Bearbeitern der Aktivität allokiert; bei METEOR$_2$ [DKM+97], CodAlf und BPAFrame (beide [SM96]) liegt er nahe bei der zur Aktivität gehörenden Anwendung. Bei all diesen Ansätzen kann ein WF-Server überlastet sein. Mit dem vorgestellten pseudo-zufälligen Verfahren wäre es möglich, ihn zu replizieren und die Last auf die Replikate zu verteilen.

In MOBILE [HS96] werden verschiedene WF-Typen von unterschiedlichen WF-Servern kontrolliert. Migrationen sind nicht vorgesehen. Allerdings können Subprozesse von einem anderen WF-Server kontrolliert werden [SNS99]. Dieser wird zur Laufzeit aufgrund verschiedener Kriterien (z.B. Rechte, Gewichte) ausgewählt. Auch bei diesem Ansatz können WF-Server überlastet sein, so dass die vorgestellten Verfahren angewendet werden können. Aufgrund der „Subprozess-Semantik“ werden parallele Zweige stets auf dem WF-Server zusammengeführt, auf dem sie sich aufgeteilt haben. Da deshalb das in Abschnitt 3.3 geschilderte Problem beim Zusammenführen paralleler Zweige nicht auftreten kann, ist das Verfahren mit zufälliger Serverwahl völlig ausreichend.

Die Anwendung des vorgestellten Verfahrens ist also nicht auf ADEPT beschränkt. Zahlreiche andere Ansätze können auch von ihm profitieren. Dabei genügt für Ansätze ohne Migrationen eine zufällige Serverwahl; ansonsten ist (wie bei ADEPT$_{distribution}$) die pseudo-zufällige Variante am besten geeignet.

6 Zusammenfassung und Ausblick

Die zahlreichen Aufgaben eines WF-Servers können zu dessen Überlastung führen. Um dies zu verhindern, kann ein WF-Server repliziert und die Last zwischen den entstehenden Replikaten aufgeteilt werden. In diesem Beitrag wurde ein Verfahren vorgestellt, bei dem für eine WF-Instanz pseudo-zufällig ein WF-Server ausgewählt wird. Dieses Verfahren realisiert eine Art „Hashing“ der WF-Instanzen auf die WF-Server eines Domains. Es ermöglicht eine beliebige Aufteilung der Last zwischen den WF-Servern, generiert im wesentlichen keine zusätzliche Kommunikation und erfordert nur sehr wenig Rechenaufwand. Weiter wurde gezeigt, wie es möglich ist, die Lastaufteilung zwischen den WF-Servern im laufenden Betrieb zu verändern. Damit ist das Problem der Überlastung der WF-Server gelöst, so dass eine Konzentration auf den Kommunikationsaspekt möglich ist, d.h. auf die geeignete Auswahl des Domains für eine Aktivität. Da bei der Modellierung nur Domains – und nicht konkrete Rechner – vorgegeben werden, können die verschiedenen WF-Server eines Domains als ein einziger leistungsstarker Server betrachtet werden („virtual powerful server“).

In diesem Beitrag wurde auch die Anwendbarkeit lastabhängiger Verfahren zur Serverauswahl untersucht. Diese haben zwar im Allgemeinen Nachteile gegenüber dem pseudo-zufälligen Verfahren, sind aber evtl. für bestimmte Spezialanwendungen interessant. Eine vertiefte Untersuchung dieser Verfahren bietet sich vor allem an, wenn die verfügbare Leistung der WF-Server stark schwankt, wenn der Ausfall von WF-Servern durch dieses Verfahren kompensiert werden soll, oder wenn ein WF-Server nur sehr wenige WF-Instanzen gleichzeitig kontrollieren kann, so dass schon kleine Ungleichverteilungen, wie sie beim pseudo-zufällige Verfahren entstehen können, zu Problemen führen würden. Für große WF-Anwendungen, die operativ eingesetzt werden, ist aber das vorgestellte Verfahren den lastabhängigen Verfahren vorzuziehen.

Danksagung: Wir danken unseren Kollegen Manfred Reichert und Clemens Hensinger für ihre hilfreichen Anregungen.

Literatur

[AKA+94] G. Alonso, M. Kamath, D. Agrawal, A. El Abbadi, R. Günthör und C. Mohan: *Failure Handling in Large Scale Workflow Management Systems.* Technischer Bericht RJ9913, IBM Almaden Research Center, November 1994.

[AMG+95] G. Alonso, C. Mohan, R. Günthör, D. Agrawal, A. El Abbadi und M. Kamath: *Exotica/FMQM: A Persistent Message-Based Architecture for Distributed Workflow Management*. In: *Proc. IFIP Working Conf. on Information Systems for Decentralized Organisations*, Trondheim, August 1995.

[BD97] T. Bauer und P. Dadam: *A Distributed Execution Environment for Large-Scale Workflow Management Systems with Subnets and Server Migration.* In: *Proc. 2nd IFCIS Conf. on Cooperative Information Systems*, S. 99–108, Kiawah Island, SC, Juni 1997.

[BD98] T. Bauer und P. Dadam: *Architekturen für skalierbare Workflow-Management-Systeme – Klassifikation und Analyse.* Ulmer Informatik-Berichte 98-02, Universität Ulm, Fakultät für Informatik, Januar 1998.

[BD99a] T. Bauer und P. Dadam: *Efficient Distributed Control of Enterprise-Wide and Cross-Enterprise Workflows.* In: *Proc. Workshop Enterprise-wide and Cross-enterprise Workflow Management: Concepts, Systems, Applications, 29. Jahrestagung der GI*, S. 25–32, Paderborn, Oktober 1999.

[BD99b] T. Bauer und P. Dadam: *Verteilungsmodelle für Workflow-Management-Systeme – Klassifikation und Simulation.* Informatik Forschung und Entwicklung, 14(4):203–217, Dezember 1999.

[BD00] T. Bauer und P. Dadam: *Efficient Distributed Workflow Management Based on Variable Server Assignments.* In: *Proc. 12th Conf. on Advanced Information Systems Engineering*, S. 94–109, Stockholm, Juni 2000.

[BMR96] D. Barbará, S. Mehrotra und M. Rusinkiewicz: *INCAs: Managing Dynamic Workflows in Distributed Environments.* Journal of Database Management, Special Issue on Multidatabases, 7(1):5–15, 1996.

[CGS97] S. Ceri, P. Grefen und G. Sánchez: *WIDE – A Distributed Architecture for Workflow Management.* In: *Proc. 7th Int. Workshop on Research Issues in Data Engineering*, Birmingham, April 1997.

[DKM+97] S. Das, K. Kochut, J. Miller, A. Sheth und D. Worah: *ORBWork: A Reliable Distributed CORBA-based Workflow Enactment System for METEOR$_2$.* Technical Report #UGA-CS-TR-97-001, Department of Computer Science, University of Georgia, Februar 1997.

[DKR+95] P. Dadam, K. Kuhn, M. Reichert, T. Beuter und M. Nathe: *ADEPT: Ein integrierender Ansatz zur Entwicklung flexibler, zuverlässiger kooperierender Assistenzsysteme in klinischen Anwendungsumgebungen*. In: *Proc. GI/SI-Jahrestagung*, S. 677–686, Zurich, September 1995.

[DRK00] P. Dadam, M. Reichert und K. Kuhn: *Clinical Workflows - The Killer Application for Process-oriented Information Systems?* In: *Proc. 4th Int. Conf. on Business Information Systems*, S. 36–59, Posen, April 2000.

[EG96] J. Eder und H. Groiss: *Ein Workflow-Managementsystem auf der Basis aktiver Datenbanken.* In: J. Becker, G. Vossen (Herausgeber): *Geschäftsprozeßmodellierung und Workflow-Management.* Int. Thomson Publishing, 1996.

[Gos91] A. Goscinski: *Distributed Operating Systems: The Logical Design.* Addison-Wesley, 1991.

[HS96] P. Heinl und H. Schuster: *Towards a Highly Scaleable Architecture for Workflow Management Systems.* In: *Proc. 7th Int. Workshop on Database and Expert Systems Applications*, S. 439–444, Zurich, September 1996.

[KAGM96] M. Kamath, G. Alonso, R. Günthör und C. Mohan: *Providing High Availability in Very Large Workflow Management Systems.* In: *Proc. 5th Int. Conf. on Extending Database Technology*, S. 427–442, Avignon, März 1996.

[MWW+98] P. Muth, D. Wodtke, J. Weißenfels, A. Kotz-Dittrich und G. Weikum: *From Centralized Workflow Specification to Distributed Workflow Execution.* Journal of Intelligent Information Systems, Special Issue on Workflow Management Systems, 10(2):159–184, März/April 1998.

[RBD99] M. Reichert, T. Bauer und P. Dadam: *Enterprise-Wide and Cross-Enterprise Workflow-Management: Challenges and Research Issues for Adaptive Workflows.* In: *Proc. Workshop Enterprise-wide and Cross-enterprise Workflow Management: Concepts, Systems, Applications, 29. Jahrestagung der GI*, S. 56–64, Paderborn, Oktober 1999.

[RD98] M. Reichert und P. Dadam: *ADEPT$_{flex}$ – Supporting Dynamic Changes of Workflows Without Losing Control*. Journal of Intelligent Information Systems, Special Issue on Workflow Management Systems, 10(2):93–129, März/April 1998.

[SK97] A. Sheth und K. J. Kochut: *Workflow Applications to Research Agenda: Scalable and Dynamic Work Coordination and Collaboration Systems.* In: *Proc. NATO Advanced Study Institute on Workflow Management Systems and Interoperability,* S. 12–21, Istanbul, August 1997.

[SM96] A. Schill und C. Mittasch: *Workflow Management Systems on Top of OSF DCE and OMG CORBA.* Distributed Systems Engineering, 3(4):250–262, Dezember 1996.

[SNS99] H. Schuster, J. Neeb und R. Schamburger: *A Configuration Management Approach for Large Workflow Management Systems.* In: *Proc. Int. Joint Conf. on Work Activities Coordination and Collaboration,* San Francisco, Februar 1999.

[WHKS98] M. Weske, J. Hündling, D. Kuropka und H. Schuschel: *Objektorientierter Entwurf eines flexiblen Workflow-Management-Systems.* Informatik Forschung und Entwicklung, 13(4):179–195, 1998.