DB-Aspekte des E-Commerce
Schwerpunkt: Anwendungen
Application Server Benchmarks

Mit der wachsenden Bedeutung des *E-Commerce* in Unternehmen rücken die Eigenschaften der hierfür eingesetzten Rechnersysteme, oft als *E-Commerce-Lösung* bzw. *–Plattform* bezeichnet, zunehmend in den Mittelpunkt der Aufmerksamkeit. Neben schwer meßbaren Qualitäten wie der Zuverlässigkeit und reinen Kostenaspekten bei Anschaffung und Betrieb, bleibt die Performance dieser Systeme als bedeutendes Unterscheidungsmerkmal. Mit Hilfe von *Benchmarks* läßt sich die Performance in einfach zu verstehende Werte fassen, die insbesondere auch technischen Laien ein grundlegendes Verständnis für die Leistungsfähigkeit schaffen können. So haben *Benchmarks* einen bedeutenden Einfluß auf die Entscheidungsfindung bei der Anschaffung von Rechnersystemen.

Die heute übliche Aufteilung von *E-Commerce-Lösungen* in drei Schichten stellt eine Herausforderung bei der Entwicklung geeigneter *Benchmarks* dar. So kann sich jede Schicht als Engpaß für die Leistung des Gesamtsystems erweisen und bedarf einer gesonderten Betrachtung. Insbesondere die mittlere Schicht der Applikationslogik, die in modernen *E-Commerce-Lösungen* durch Applikationsserver vertreten ist, stellt im Gegensatz zu Datenhaltung- und Präsentationsschicht eine relativ neue Technologie dar und bereitet daher beim *Benchmarking* noch Schwierigkeiten.

Ziel dieser Ausarbeitung ist eine Vorstellung und ein anschließender Vergleich existierender *Benchmarks* im Hinblick auf ihre Eignung für die Leistungsbeurteilung von Applikationsservern.

Nach einer allgemeinen Einführung in die Begriffswelt des *Benchmarking* und der mit ihm verbundenen Einschränkungen folgt eine kurze Motivation für die Notwendigkeit der Leistungsbewertung von *E-Commerce-Lösungen* im allgemeinen und Applikationsservern im speziellen.

Um einen objektiven und fairen Vergleich der existierenden *Benchmarks* zu ermöglichen, werden dann Vergleichskriterien beschrieben, die bei der anschließenden Vorstellung der ausgewählten *Benchmarks* als Grundlage für eine objektive Beurteilung dienen sollen. Zum Abschluß folgt eine Übersicht über die aktuelle Situation und die verbleibenden Defizite bei der Leistungsbewertung von Applikationsservern.

Inhalt

1. Benchmarking – Eine Einleitung

Mit der Entwicklung der ersten *Benchmarks* ist eine eigene Begriffswelt entstanden. Die wichtigsten dieser Grundbegriffe sollen zunächst an dieser Stelle erläutert werden.

1.1 Definition des Begriffs *Benchmark* [1]

Der Begriff *Benchmark*, was wörtlich übersetzt soviel wie „Bezugswert“ bedeutet, hat im Bereich der Informationstechnologie eine besondere Bedeutung. Man versteht darunter einen Test, um die Leistungsfähigkeit (*Performance*) von Computersystemen oder von Teilsystemen bei der Ausführung definierter Aufgaben zu bestimmen und mit Hilfe von *Metriken* vergleichbar zu machen [2]. Diese Tests sind im allgemeinen in Form von Software realisiert, die auf dem zu testenden System (*System under test*, kurz *SUT*) ausgeführt wird. Die simulierte Belastung des *SUT* wird als Arbeitslast oder *Workload* bezeichnet.

Der Versuch, die Leistungsfähigkeit eines Rechnersystems auf einen Wert zu reduzieren, scheitert jedoch an der Vielfalt von möglichen Anwendungsgebieten. Daher entstanden im Laufe der Entwicklung der Informationstechnologie eine Vielzahl von *Benchmarks*, die jeweils auf bestimmte Bereiche zugeschnitten sind.

1.2 Klassen von *Benchmarks*

Bereits aus der Definition des *Benchmark*-Begriffs geht bereits eine grobe Einteilung der *Benchmarks* in zwei Kategorien hervor: Man unterscheidet *Benchmarks*, deren Ziel die möglichst isolierte Bewertung eines einzelnen oder einiger weniger Teilsysteme ist und die im folgenden als *Komponentenbenchmarks* bezeichnet werden, von den sogenannten *Systembenchmarks*, welche die *Performance* des Gesamtsystem evaluieren sollen.

Eine genaue Trennung dieser beiden Kategorien ist nicht immer möglich und auch vom jeweiligen Standpunkt des Anwenders abhängig. Ein *Benchmark*, der für einen einzelnen Rechner einen Leistungswert ermittelt, kann als *Systembenchmark* verstanden werden, wenn es darum geht diesen Rechner als Einzelsystem zu bewerten. Geht es jedoch darum, die Funktion des gleichen Rechners als Teil eines komplexeren verteilten Systems zu beurteilen, müßte man ihn als *Komponentenbenchmark* einordnen.

1.2.1 *Komponentenbenchmarks*

Die meisten klar einzuordnenden Beispiele für *Komponentenbenchmarks* testen einzelne Hardware-Komponenten eines Rechnersystems. Beispiele hierfür sind:

CPU-Leistung: Ziel dieser *Benchmarks* ist die Ermittlung der reinen Rechenleistung eines Prozessors, häufig getrennt nach Ganzzahl- und Fließkomma-Einheit, die in allen modernen Prozessoren als mehr oder weniger stark getrennte Einheiten realisiert sind. Die Performance wird häufig in „Millionen Operationen pro Sekunde“ gemessen. Zu dieser Kategorie gehört unter anderem der weit verbreitete CPU2000-Benchmark [3] der *System Performance Evaluation Corporation (SPEC)*.

Leistung des Hauptspeichersubsystems: Neben der durchschnittlichen und der maximalen Transferrate sind hier besonders die Verzögerung für das Ausliefern eines Datums relevant, da die CPU ggf. in dieser Zeit untätig bleibt. Aufgrund der stetig größer werdenden Lücke zwischen Prozessor- und Speichergeschwindigkeit kommt der Hauptspeicherleistung eine zunehmend größere Rolle für die Leistungsfähigkeit des Gesamtsystems zu. Trotz starker Entkopplung der CPU vom RAM durch große und schnelle Zwischenspeicher (*Caches*) wird die Systemleistung häufig durch die geringe Speichergeschwindigkeit begrenzt.

I/O-Leistung: Hierunter faßt man die Geschwindigkeit von Ein-/Ausgabe-Technologien zusammen, beispielsweise von System- und Peripheriebussen. Ziel dieser *Benchmarks* ist es, die in der Praxis von der zu testenden Technologie erreichbaren Maximaltransferraten zu ermitteln und in Bezug zu den theoretischen Maximalwerten zu setzen. Diese *Benchmarks* werden beispielsweise zum Aufspüren von Flaschenhälsen in Rechnersystemen eingesetzt.

Sekundärspeichergeschwindigkeit: Diese *Benchmarks* ermitteln die Leistungsfähigkeit von nichtflüchtigen Speichermedien wie Festplatten oder optischen Laufwerken. Neben den erreichbaren Transferraten sind hier insbesondere die Zugriffszeiten von Interesse, da sie wesentlich die Leistungsfähigkeit des Mediums beispielsweise im Datenbankeinsatz bestimmen.

Grafikleistung: Mit der zunehmenden Verbreitung von graphischen Benutzeroberflächen seit dem Ende der 80er Jahre wurde die Leistung des Graphiksubsystems entscheidend für die Nutzbarkeit eines Rechnersystems. Nachdem Graphiksubsysteme im 2D-Bereich jedoch einen Leistungsstand erreicht haben, der für jede denkbare Anwendung „schnell genug“ erscheint, wurde mit dem Aufkommen der ersten 3D-Beschleuniger der

Schwerpunkt des *Benchmarking* in diesen Bereich verlagert. Technische Anwendungen wie CAD- und Visualisierungssysteme profitieren in erheblichem Maße von Unterstützung durch 3D-Hardware, aber auch die Unterhaltungsindustrie hat sich als Motor für die Entwicklung leistungsfähiger und zugleich bezahlbarer 3D-Graphikkarten hervorgetan. *Benchmarks* in diesem Bereich haben inzwischen auch die Funktion von Technologiedemonstrationen übernommen, die dem Anwender und möglichen Kunden in optisch opulenter Weise die Vorzüge der jeweils aktuellen Produktgeneration verdeutlichen sollen.

1.2.2 Systembenchmarks

Die Leistung einzelner Komponenten kann im allgemeinen schon für eine Aussage über die Leistungsfähigkeit des System bei bestimmte Anwendungen genutzt werden. Oft ist es jedoch wünschenswert, spezielle *Benchmarks* zur Ermittlung der Tauglichkeit des Gesamtsystems für bestimmte Anwendungen heranzuziehen. Diese *Systembenchmarks* bewerten dabei nicht nur die Performance der Hardware, sondern auch das reibungslose Zusammenspiel der Komponenten und die Leistungsfähigkeit der eingesetzten Anwendungs- und Systemsoftware. Für das *Benchmarking* von Standard-PCs unter Windows hat sich hier der SysMark der BAPCo (Business Applications Performance Corporation) [4] als Referenz etabliert, der momentan in der Version 2001 erhältlich ist. Er ermittelt die Leistungsfähigkeit des Systems in zwei Szenarien, *Office Productivity* und *Internet Content Creation*, mit Hilfe von realen Anwendungsprogrammen. Diese werden für einen Benchmarklauf nebenläufig von simulierten Anwendern genutzt und die vom System benötigte Reaktionszeit gemessen. Aus dem Mittel aller während eines Szenarios gemessenen Zeiten wird dann ein Benchmarkresultat ermittelt.

1.3 Schwierigkeiten des *Benchmarking*

Ein grundlegendes Problem bei der Erstellung und Durchführung von *Komponentenbenchmarks* besteht in der eingeschränkten Möglichkeit zur getrennten Betrachtung der zu testenden Komponente vom Rest des Systems. Da die wenigstens Komponenten physisch isoliert bewertet werden können, müssen sie notwendigerweise innerhalb einer Systemumgebung eingesetzt werden. Deren Einfluß auf die Testergebnisse sollte möglichst minimal gehalten werden, um die Vergleichbarkeit und Plausibilität der Ergebnisse zu gewährleisten. Wo dies nicht durch entsprechende Realisierung des *Benchmarks* möglich ist, kann man durch exakte Beschreibung der zu verwendenden Systemumgebung eine Vergleichbarkeit sicherstellen, handelt sich aber dabei Probleme mit der Anpassung an neue Technologien ein, die zur Referenzumgebung inkompatibel sind.

Für *Benchmarks* besteht generell das Problem, daß sie meistens synthetisch sind und damit nur eine Abstraktion von der tatsächlichen Nutzung des Computersystems darstellen. Häufig prüfen *Benchmarks* nur bestimmte, sehr eng eingegrenzte Nutzungsszenarien, die nicht notwendigerweise denen realer Anwendungen entsprechen müssen. Dieses Problem versuchen einige *Benchmarks* wie der oben erwähnte *BAPCo SYSmark* dadurch zu umgehen, daß sie die Anwendungssoftware selbst für die Bewertung heranziehen, indem sie die Nutzung dieser Anwendung simulieren und die dazu benötigte Zeit messen. Doch auch hier verbleiben Ungenauigkeiten, da die simulierten Nutzungsprofile nicht notwendigerweise denen des späteren Anwenders entsprechen müssen.

1.4 *Benchmarking* als Entscheidungsgrundlage

Ist man sich dieser Einschränkungen bewußt, können *Benchmarks* dennoch einen relativ einfachen und objektiven Vergleich verschiedener Computersysteme oder Teilsysteme ermöglichen. So läßt sich mit dem eigenen Nutzungsprofil entsprechenden *Benchmarks* der Entscheidungsprozeß für den Erwerb eines Rechnersystems durch meßbare Fakten unterstützen.

Hersteller, deren System bei populären *Benchmarks* gut abschneidet, können mit entsprechend steigenden Absatzzahlen rechnen. Insbesondere im Privatkundenbereich ist dieser Zusammenhang teilweise sehr deutlich, da sich dieser Kundenkreis außer vom Preis hauptsächlich durch die sicht- und meßbare Leistung lenken läßt. Im Geschäftskundenbereich werden dagegen auch weniger gut in Zahlen faßbare Kriterien für die Kaufentscheidung herangezogen, beispielsweise Serviceleistungen, Ausfallsicherheit und Kompatibilität zu bereits im eigenen Unternehmen existierenden Systemen. Diese Aspekte werden mit der zunehmenden Bedeutung von Rechnersystemen und *E-Commerce-Lösungen* immer wichtiger und unter dem Stichwort *total cost of ownership* zusammengefaßt. Dennoch sind auch hier die Auswirkungen von gutem Abschneiden bei *Benchmarks* auf die Einkaufspolitik nicht zu unterschätzen, insbesondere da diese als Rechtfertigung und zur Absicherung von Entscheidungen dienen. Zudem bieten insbesondere *Benchmarks* aus dem geschäftlichen Umfeld neben einem reinen Leistungswert auch einen Wert an, der die vom jeweiligen Computersystem gelieferte Leistung und die Kosten dieses Systems zueinander in Beziehung setzt.

1.4 *Benchmarking* vs. *Benchmarketing*

Mit dem Wissen um die Bedeutung von *Benchmarks* für den Markterfolg eines Produkts und der zunehmenden wirtschaftlichen Bedeutung von Computern für Unternehmen kam es zu zwei parallelen Entwicklungen: Zum einen begannen die Hersteller von Hardware und Software, ihre Produkte für Benchmarks zu „optimieren“. Neben durchaus wünschenswerten Maßnahmen, die tatsächlich zu einer Steigerung der Leistung in realen Anwendungen führten, war es nicht unüblich, daß auch Modifikationen mit weniger lauteren Zielen vorgenommen wurden. Diese häufig als *Benchmark Specials* bezeichneten Aktivitäten umfassen beispielsweise das Ausnutzen von Wissen über das Nutzungsprofil des Benchmarks, oder die Verwendung von Methoden, welche die Nutzung eines Systems auf das *Benchmarking* beschränken.

Andere Tätigkeiten werden weniger euphemistisch als Schummeln (*Cheating*) bezeichnet. Dazu gehören beispielsweise Systeme, die einen *Benchmark* anhand seines charakteristischen Zugriffsmusters erkennen und diese Anfragen dann mit hoher Geschwindigkeit bedienen, ohne dabei jedoch wirklich Arbeit zu verrichten.

Gleichzeitig begann eine regelrechte Inflation von *Benchmarks*. So entwickelten zahlreiche Hardwarehersteller eigene *Benchmarks*, die natürlich die Überlegenheit der eigenen Produkte gegenüber denen der Mitbewerber demonstrierten, indem sie explizit die eigenen Stärken „testeten“, bzw. die Schwächen der Konkurrenz ausnutzten. Zudem wurden diese Hausmacher-*Benchmarks* auch benutzt, um die Vorteile einer neuen Produktgeneration zu untermauern, insbesondere wenn sie in der Realität weniger ersichtlich waren.

Als Gegenbewegung entwickelten Dritthersteller oder Konsortien eigene *Benchmarks*, die aufgrund ihrer größeren Unabhängigkeit und der dadurch erhofften Objektivität deutlich größere Akzeptanz erfahren haben. Aber auch hier bleibt das Problem, das die vom Hersteller für das Prospekt ermittelte und die beim Kunden dann tatsächlich erzielbare Leistung stark voneinander abweichen, wofür sich recht schnell der Begriff des *Benchmarketing* eingebürgert hat.

2. Benchmarking von E-Commerce-Lösungen

Nachdem nun grundlegende Begriffe zum Thema Benchmarking erläutert wurden, soll nun die besondere Problematik bei der Leistungsbewertung von *E-Commerce-Lösungen* verdeutlicht werden.

2.1 Die Komponenten einer *E-Commerce-Lösung*

Abb.1: Struktur einer E-Commerce-Lösung

Mit der zunehmenden Bedeutung des *E-Commerce* auch für Unternehmen der *Old Economy* entstanden neue Rechnersysteme, welche die erforderliche Funktionalität realisieren. Da *E-Commerce* und traditionelle Geschäftsbereiche häufig auf gemeinsamen Datenbeständen agieren, bauen diese Systeme auf der bereits existierenden IT-Infrastruktur auf, die häufig mit dem Unternehmen gewachsen und dementsprechend sehr heterogen ist. Zudem erfolgt der Zugriff auf *E-Commerce-Systeme* oft über mehrere Medien oder *Kanäle*, was zusammen mit der Heterogenität der Datenhaltungssysteme eine Strukturierung der *E-Commerce-Lösung* in drei Schichten oder Komponenten erfordert.

Der Begriff der *E-Commerce-Plattform* faßt dabei neue und alte (*Legacy-)* Systeme zusammen, und schließt auch die zugehörige Software, welche die Realisierung der Geschäftlogik und die Integration übernehmen muß mit ein.

Im folgenden sollen diese Komponenten insbesondere im Bezug auf eine Leistungsbewertung kurz vorgestellt werden.

2.1.1 Präsentationsschicht

Die Präsentationsschicht hat zunächst die Aufgabe, die über die zahlreichen und z.T. sehr unterschiedlichen Medien eingehenden Anfragen in ein einheitliches Format umzusetzen, um so den tieferen Schichten eine vom verwendeten Medium unabhängige Weiterverarbeitung zu ermöglichen. Umgekehrt erfolgt hier auch die Konvertierung der Antworten in die Formate des verwendeten Mediums oder Kanals, man spricht daher auch von *Kanalanspassungsschicht* [21].

Da *E-Commerce* insbesondere im *B2C*-Bereich fast ausschließlich über das Internet abgewickelt wird, ist die Präsentationsschicht im allgemeinen in Form von Webservern vertreten. In vielen Unternehmen existieren bereits Webserver, die ursprünglich nur die „Visitenkarte“ der Firma im Internet beheimateten. Im allgemeinen können sie für die Verwendung innerhalb der *E-Commerce-Lösung* angepaßt werden. Ihre Performance ist ein entscheidender Faktor für die Nutzbarkeit der *E-Commerce-Lösung* Aufgrund der langen Existenz der Webserver-Technologie existieren hier zahllose *Benchmarks*, wie z.B. der frei erhältliche WebBench 3.0 der Ziff Davis eTesting Labs [6], oder Mindcrafts WebStone [7]. Auch die SPEC hat sich mit SPECWeb99 [8] dieses Themas angenommen.

2.1.2 Datenhaltungsschicht (Enterprise Information Systems)

Unter diesem Oberbegriff werden Systeme zur persistenten Datenhaltung zusammengefaßt. Dazu gehören neben den allgegenwärtigen Datenbanken auch komplexe Unternehmenssoftware wie *Enterprise Ressource Planning* (ERP)-Systeme und sogenannte *Legacy*-Systeme wie Mainframes. Gelegentlich wird diese Schicht auch als *Backend-System* bezeichnet.

Die neuen Anwendungen setzen nun auf dieser Datenbasis auf. Die Zugriffe werden über geeignete generische Schnittstellen gekapselt, so daß die Anwendung zu einem gewissen Grad unabhängig vom verwendeten *Enterprise Information System* bleibt.

Die ständige Verfügbarkeit und der schnelle Zugriff auf die hier gespeicherten Daten ist essentiell für die Leistungsfähigkeit der *E-Commerce-Lösung*. Für das *Benchmarking* von Datenbanksystemen hat sich insbesondere das *Transaction Processing Performance Council* (TPC, [9]) hervorgetan, das schon 1989 mit der Veröffentlichung des TPC-A Benchmarks einen Quasi-Standard für die Leistungsbewertung geschaffen hat. Unter Berücksichtigung der jeweils aktuellen Nutzungsszenarien werden regelmäßig neue Versionen entwickelt.

2.1.3 Anwendungsschicht (*Application Server)*

Die Verbindung von Datenhaltung und Präsentation erfolgt durch die Ebene der *Business Logic* oder Geschäftslogik, welche die Geschäftvorgänge realisiert.

Bei den ersten E-Commerce-Anwendungen war die Geschäftlogik häufig noch so einfach, daß sie mit existierenden Technologien für die Erzeugung dynamischer Inhalte wie *Server Side Includes* oder *CGIs* auf dem Webserver realisiert werden konnte, evtl. unterstützt von intelligenten Datenbanken, die mit Hilfe *von Stored Procedures* auch komplexere Anfragen effizient ausführen können.

Mit steigenden Ansprüchen der Kunden stieg jedoch auch die Komplexität der Anwendungslogik so weit, daß diese Ansätze zunehmend an ihre Grenzen stießen. So entstand die Notwendigkeit, eine eigene Schicht für die Anwendungslogik vorzusehen. Diese sogenannten *Application Server* bilden den zentralen Bestandteil einer voll ausgebauten *E-Commerce-Lösung*. Neben proprietären Ansätzen kann die Geschäftslogik auch mit Hilfe von entsprechenden *Frameworks* wie Suns *Java 2 Enterprise Edition* [10] oder *.NET* von Microsoft [11] mit Hilfe verteilter Komponenten realisiert werden. Diese *Frameworks* bieten eine Infrastruktur, welche die

Implementierung der Anwendung erheblich erleichtert, da sich die Entwickler nicht um systemnahe Dinge wie Transaktionsverwaltung, Kommunikation zwischen verschiedenen Teilen eines verteilten Systems oder Caching kümmern müssen.

Während anfangs häufig einer der beiden anderen Komponenten der *E-Commerce-Plattform* den Engpaß des Systems dargestellt hat, wird mit der stetig steigenden Komplexität der Geschäftslogik zunehmend der Applikationsserver zum Flaschenhals für die Leistungsfähigkeit. Trotz der Neuheit dieser Technologien sind auch für Applikationsserver bereits verschiedene Ansätze zur Leistungsbewertung entstanden, die im folgenden vorgestellt und bezüglich verschiedener Kriterien verglichen werden sollen. Zunächst soll jedoch der Performance-Begriff in dieser Domäne genauer definiert werden.

2.2 Performance eines Servers

Der Begriff „Geschwindigkeit“ bezieht sich bei großen Serversystemen, wie sie *E-Commerce-Plattformen* und die hier betrachteten Applikationsserver darstellen, zumeist auf den erreichbaren Durchsatz, also die Anzahl von durchführbaren Geschäftsvorfällen oder Transaktionen pro Zeiteinheit, und die dabei erzielte durchschnittliche und maximale Antwortzeit, also die Verzögerung zwischen Nutzerinteraktion und Reaktion des Systems. Diese beiden Ziele sind schwierig zu vereinbaren, da eine Optimierung auf hohen Durchsatz zu einer Erhöhung der Antwortzeiten führt, während kurze Antwortzeiten aufgrund der häufigeren Kontextwechsel den Durchsatz reduzieren. Hier müssen die Hersteller von Serversystemen einen Kompromiß finden, der beide Anforderungen berücksichtigt.

Neben der Geschwindigkeit, wie sie beim eingangs beschriebenen *Benchmarking* einzelner Komponenten im Vordergrund steht, kommen bei großen Serversystemen weitere Aspekte hinzu.

Der Begriff der *Skalierbarkeit* wird für die Beschreibung von zwei verschiedenen Aspekten verwendet. Zum einen beschreibt er, wie gut ein Serversystem auf steigende Last reagiert, wie gut es sich seinem theoretischen Maximaldurchsatz annähern kann bzw. wie stark Durchsatz und Antwortzeit leiden, wenn das System an seine Belastungsgrenze gerät.

Zu guter *Skalierbarkeit* gehört aber auch die optimale Reaktion des Gesamtsystem auf Erweiterungen. Insbesondere Server, die häufig als Multiprozessormaschinen realisiert sind, reagieren je nach verwendeter Hardware- und Softwarearchitektur mit unterschiedlich hohem Leistungsgewinn auf Erweiterung wie zusätzliche Prozessoren oder mehr Speicher. Dies ist besonders wichtig, wenn ein Rechnersystem mit den Anforderungen des Unternehmens wachsen soll, hier kann gute *Skalierbarkeit* den Unterschied ausmachen zwischen einem Neukauf, der oft auch mit enormen Kosten für Umstellung und Umschulungen verbunden ist, und dem preisgünstigeren und einfach zu realisierenden Zukauf neuer Hardware.

Alle diese Leistungswerte sollen im folgenden vereinfachend unter dem Begriff „Performance“ zusammengefaßt werden.

Weitere Aspekte, die bei unternehmenskritischen Rechnersystemen von nicht unerheblichem Interesse sind, wie z.B. Ausfallssicherheit, können im allgemeinen nicht im Rahmen eines *Benchmarks* bestimmt werden, da sie nur über einen langen Zeitraum und unter den tatsächlichen Einsatzbedingungen ermittelt werden können. Dies ist jedoch äußerst aufwendig und kostenintensiv, so daß man hier im allgemeinen auf Erfahrungswerte angewiesen ist.

2.3 Herausforderungen des *Application Server Benchmarking*

Zunächst besteht hier wie bei allen *Benchmarks* zur Bewertung der Performance für bestimmte Anwendungen das Problem, die tatsächliche Arbeitslast der Anwendung möglichst genau nachzubilden, um praxisrelevante Ergebnisse zu erhalten. Insbesondere rein synthetische *Benchmarks* können hier durch zu starke Abstraktion von der realen Systemnutzung verfälschte Ergebnisse liefern, weshalb für das *Benchmarking* häufig realitätsnahe Anwendungsszenarien entworfen werden, deren Nutzung dann durch Treibersysteme (*Driver*) simuliert und die dabei erzielte Geschwindigkeit gemessen wird.

Eine weitere Schwierigkeit entsteht durch den oben beschriebenen mehrschichtigen Aufbau *von E-Commerce-Lösungen*, da dieser die isolierte Betrachtung einer Teilkomponente erheblich erschweren kann. Insbesondere den Einfluß der Geschwindigkeit der Datenhaltungsschicht auf die Benchmarkergebnisse eines Applikationsservers kann erheblich sein, so daß es notwendig ist, diesen Einfluß beim *Benchmarking* zu minimieren.

Ein weiteres Problem, das generell beim Test von leistungsfähigen Serversystemen auftritt, liegt in der Erzeugung der Arbeitslast. Hierzu ist im allgemeinen ein ähnlich schnelles System notwendig, oder die Lastgenerierung muß auf mehrere Systeme verteilt werden.

3. Application Server Benchmarks

Mit dem Aufkommen immer komplexerer E-Commerce-Applikationen sind die für die Gesamtperformance einer *E-Commerce-Lösung* bisher wenig beachteten Applikationsserver verstärkt in den Mittelpunkt des *Benchmarking* gerückt. Hierbei sind nicht nur Vergleiche der Serverhardware gefordert, die aufgrund der großen Vielfalt eine enorme Bandbreite vom einfachen Ein-Prozessor-Rechner auf PC-Basis bis hin zum 64-Prozessor-Server abdeckt.

Vielmehr besteht auch großes Interesse an einem quantitativen Vergleich der verschiedenen Technologien, die zur Realisierung der auf dem Applikationsserver beheimateten Applikationen zur Verfügung stehen. Dieser führte zur Entstehung der ersten *Benchmarks* für E-Commerce-Lösungen und Applikationsserver, von denen im folgenden eine Auswahl näher vorgestellt werden soll.

3.1 Vergleichskriterien für *Benchmarks*

Um die verschiedenen zur Zeit verfügbaren *Benchmarks* für Applikationsserver zielgerichtet vergleichen zu können, sollen zunächst einige Kriterien beschrieben werden, denen besondere Beachtungen zu schenken ist.

3.1.1 Technologieunabhängigkeit

Um Applikationsserver unabhängig von der zugrundeliegenden Softwaretechnologie vergleichen zu können, sollte ein idealer *Benchmark*, ggf. nach geringfügiger Anpassung, die jedoch die Ergebnisse nicht verfälschen darf, auf Servern mit beliebiger Technologie verwendbar sein. Nur so ist es möglich, neben der Performance der Hardware auch einen quantitativen Vergleich verschiedener Applikationsserver-Technologien zu machen.

Ein Beispiel für einen klassischen CPU-*Benchmark*, der das Kriterium der Technologieunabhängigkeit nahezu perfekt erfüllt, ist der SPEC CPU2000-*Benchmark*, der im Quelltext vorliegt und so für die jeweilige Zielplattform kompiliert werden kann.

3.1.2 Aufwand

Dieses Kriterium umfaßt sowohl den zeitlichen als auch den finanziellen Aufwand, der für einen Benchmarklauf notwendig aus.

Je nach der Form, in welcher der *Benchmark* vorliegt, kann sich der Aufwand für eine Messung erheblich unterscheiden. Handelt es sich lediglich um eine Spezifikation, ist vor dem Testlauf zunächst die Implementierung notwendig, was je nach Umfang einen erheblichen Aufwand bedeutet. Liegt der *Benchmark* dagegen als fertiges Produkt vor, genügt im Idealfall die Installation auf dem Testrechner. Zwischen diesen beiden Extremen finden sich *Benchmark-Frameworks*, die prinzipiell vollständig sind, aber an die jeweilige Zielumgebung angepaßt werden müssen.

Ein weiterer Faktor, der den Aufwand beim *Benchmarking* beeinflußt, sind die Anforderungen an das Testsystem. So genügt hier manchmal der Applikationsserver, während in anderen Fällen eine komplette *E-Commerce-Plattform* zur Verfügung stehen muß. Zusätzlich sind eventuell noch leistungsstarke Treibersysteme notwendig, um die gewünschte Arbeitslast zu erzeugen. Die Kosten für die Bereitstellung und die Konfiguration dieser Systeme sind unter Umständen größer als die für den Applikationsserver selbst.

3.1.3 Isolation

Wie zuvor erläutert wurde, bestehen *E-Commerce-Plattformen* aus mehreren Komponenten, die bezogen auf die Gesamtleistung des Systems stark voneinander abhängig und für das Funktionieren des Systems nicht zu trennen sind. Es ist nun zu klären, welche dieser Komponenten Einfluß auf die Ergebnisse des *Benchmarks* haben bzw. wie gut der *Benchmark* die Komponenten voneinander isoliert bewerten kann.

3.1.4 Performanceaspekte

Da die Leistungsfähigkeit eines Servers durch mehrere Dimensionen charakterisiert wird, sollte ein *Benchmark* möglichst viele für die Ermittlung des Resultats berücksichtigen. Die wichtigsten sind Durchsatz, Antwortzeit und Skalierbarkeit.

3.1.5 Unabhängigkeit

Wie bei den einleitenden Erläuterungen zu *Benchmarks* erwähnt wurde, nutzen viele Hersteller Benchmarks, um die Überlegenheit ihrer Produkte hervorzuheben. Die Relevanz der Ergebnisse hängt dabei nicht zuletzt von der Unabhängigkeit des *Benchmarks* vom jeweiligen Hersteller ab. So existieren *Benchmarks*, die von großen Konsortien ausgearbeitet wurden, in denen viele große Hersteller aus dem jeweiligen Gebiet zusammenarbeiten, deren Produkte aber auf dem Markt Konkurrenten sind. Hier kann man von größerer Objektivität und Neutralität der *Benchmarks* ausgehen, da bei entsprechender Struktur des Konsortiums eine Einflußnahme einzelner Hersteller minimiert werden kann.

Ein anderes Extrem wäre ein *Benchmark*, der vom Hersteller des Applikationsservers selbst stammt. Hier muß die Unabhängigkeit grundsätzlich als gering angesehen werden, und vor einem Vergleich verschiedener Produkte oder Plattformen auf dieser Basis gilt es sich zu vergewissern, daß die Fairness gewahrt bleibt.

3.1.6 Domäne

Da sich E-Commerce-Anwendungen aus verschiedenen Geschäftsumfeldern oder Szenarien unter Umständen stark in ihrem Arbeitlast-Profil unterscheiden, sind die Ergebnisse, die innerhalb eines solchen Umfeldes ermittelt wurden, nur sehr bedingt auf andere übertragbar. Um die Aussagekraft eines *Benchmarks* zu erhöhen, wäre es daher wünschenswert, er nicht nur ein einzelnes Szenario abdecken würde.

3.1.7 Popularität

Ein *Benchmark* wird als Grundlage und Hilfsmittel für Entscheidungsträger um so bedeutungsvoller, je größer seine Akzeptanz auf dem jeweiligen Markt ist. Ein Entscheidung, die sich auf die Ergebnisse eines weit verbreiteten *Benchmarks* stützt, läßt sich damit problemloser rechtfertigen als eine Entscheidung auf Basis eines weniger populären *Benchmarks*.

Weiterhin bedeutet große Akzeptanz auch das Vorhandensein von Benchmarkresultaten für viele Systeme, die so ohne großen Aufwand für die Entscheidungsfindung genutzt werden können, während andernfalls die jeweiligen Systeme selbst bewertet werden müßten, was im allgemeinen nur für einen kleinen Ausschnitt der in Frage kommenden Systeme machbar und unter Umständen gar nicht möglich ist.

3.2 *Benchmarks*

Anhand der vorgestellten Kriterien vergleichen wir im folgenden verschiedene Benchmarkansätze, die im weiteren Sinne das *Benchmarking* von Applikationsservern ermöglichen.

3.2.1 Transaction Processing Performance Council – TPC-W [12]

Der *Transaction Processing Performance Council* (TPC) bezeichnet sich als „non-profit corporation“, in zur Zeit 26 große und kleine Soft- und Hardwarehersteller als Vollmitglieder vertreten sind, darunter auch Hersteller von Applikationsservern wie BEA, Sun, Microsoft und IBM. Jedem Mitglied steht im *Full Council* eine Stimme zu, so daß Entscheidungen grundsätzlich auf demokratischen Wege mit einer Zwei-Drittel-Mehrheit zu Stande kommen. Das TPC wurde 1988 mit acht Mitgliedern gegründet, und sollte neben der reinen Entwicklung von *Benchmark*-Spezifikationen auch deren Einhaltung überwachen. Dies war in einer Situation entstanden, in der zwar einige breit akzeptierte *Benchmarks* für die damals aufkommenden OLTP-Systeme wie TP1 und DebitCredit existierten, aber viele Firmen deren Spezifikation nach eigenem Gutdünken interpretierten und abänderten. So kamen Ergebnisse zu Stande, die mit tatsächlichen Anwendungen unerreichbar blieben. Diese Situation sollte der TPC als zentrale Kontrollinstanz beseitigen helfen.

Der TPC kann auf eine lange Reihe von erfolgreichen *Benchmarks* im Bereich von Transaktionssystemen zurückblicken. Die inzwischen überholten *Benchmarks* TPC-A und –B zielten auf *OLTP-Systeme* und wurden inzwischen vom TPC-C abgelöst. Weiterhin entstanden mit TPC-D und TPC–H Benchmarks *für Decision-Support-Systeme*. Generell liegen TPC-Benchmarks nicht in Form eines Produkts vor, sondern spezifizieren lediglich die Anforderung an eine Implementierung des *Benchmarks*. Entsprechend aufwendig ist die Veröffentlichung eines Ergebnisses. Dennoch erreichten die meisten der TPC-*Benchmarks* große Bedeutung im Datenbankmarkt.

Allgemeine Richtlinien

Nachdem einige Firmen die ersten TPC-*Benchmarks* intensiv und erfolgreich in ihrem Marketing genutzt hatten, tauchte zunehmend die Frage auf, wie man möglichen Betrugsversuchen vorbeugen könnte. In Folge wurden

„allgemeine Richtlinien für die Implementierung" entworfen, die seither jeder TPC-*Benchmark*-Spezifikation vorangestellt sind.

Zu den zentralen Forderungen gehört, daß sämtliche *Benchmarks* nur mit Systemen, Produkten und Technologien durchgeführt werden dürfen die allgemein verfügbar sind und die für das vom jeweiligen TPC-Benchmark angesprochene Marktsegment eine ausreichende Bedeutung haben bzw. von den Nutzern des Marktsegments mit hoher Wahrscheinlichkeit eingesetzt werden würden. Damit werden neue Technologien nicht von vorneherein ausgeschlossen.

Explizit verboten ist die Verwendung von Produkten und Technologien, die nur die *Benchmark*-Ergebnisse, nicht aber die Performance in realen Anwendungen verbessern, die Richtlinien sprechen hier von einem „*Benchmark Special*". Anhand von mehreren Kriterien sollen diese identifiziert werden:

- Die Implementierung weißt Restriktionen auf, die ihren Einsatz für andere Anwendungen als den *Benchmark* beschränken.
- Die Implementierung nutzt das Wissen über das Nutzungsprofil durch den *Benchmark* aus.
- Die Implementierung erfordert außergewöhnliche Erfahrung auf Seiten der Administratoren und Anwender.
- Die Preisbildung der Implementierung weicht deutlich von üblichen Geschäftspraktiken ab.

Bei vermuteten Verletzungen dieser Bestimmungen oder der eigentlichen Benchmarkspezifikation wurde zunächst das *Technical Advisory Board* (*TAB*) eingeführt, das auf Anfrage und bei konkreten Verdachtsfällen tätig wird. Da dies jedoch erst nach einer Veröffentlichung eines *Benchmarks* geschieht, könnten auf nicht den Bestimmungen entsprechender Weise ermittelte Benchmarkergebnisse nicht nur die Glaubwürdigkeit des jeweiligen Herstellers, sondern auch die des TPC gefährden. Deshalb wurde ein Auditing-Verfahren eingeführt, in dem neue Implementierungen noch vor ihrer Veröffentlichung durch eine Gruppe zertifizierter Auditoren hinsichtlich der Einhaltung der Bestimmungen überprüft werden.

TPC-W

Der 1999 veröffentlichte TPC-W-*Benchmark* nimmt sich als einer der jüngsten Produkte des TPC der Leistungsbewertung von *E-Commerce-Lösungen* an, indem er eine Arbeitslast innerhalb einer kontrollierten Umgebung abarbeitet, welche die Aktivitäten eines transaktionalen Webservers simuliert. Als Szenario wurde ein Online-Buchhandel gewählt, dessen Seiten von simulierten Kunden genutzt werden. Daraus ergeben sich vielfältige Anforderungen an das *E-Commerce-System*, wie die Verwaltung von mehreren Browser-Sitzungen, die Erzeugung dynamischer Webseiten nebst zugehöriger Datenbankzugriffe und das nebenläufige Abarbeiten verschieden komplexer Online-Transaktionen.

Die Spezifikation

Wie bei alle bisherigen TPC-*Benchmarks* handelt es sich auch beim TPC-W nicht um ein einsatzbereites Produkt, sondern um eine Spezifikation, welche die Anforderungen an eine Implementierung beschreibt. Die genaue Realisierung bleibt dabei weitestgehend frei, es werden keine Vorschriften über die Anzahl der einzusetzenden Systeme gemacht, so daß mögliche Implementierungen die komplette Bandbreite zwischen Ein-Server-Lösung und komplexem *Multi-Tier-System* abdecken können.

Das TPC-W-Datenbankschema

Da das TPC ursprünglich aus der Welt der Datenbanken kommt, liegt dem System eine Datenbank mit zahlreichen Relationen zugrunde, deren Größe und Attributanzahl ein breites Spektrum abdecken. Die Erfüllung der ACID-Kriterien ist Grundvoraussetzung.

Als Minimalanforderung sind acht Tabellen vorgesehen, die jedoch um weitere Tabellen ergänzt werden können, um flexibel auf die Fähigkeiten der zu testenden Produkte einzugehen. Anzahl und Typ der Attribute sind vorgegeben.

Über zwei Parameter, die Kardinalität der ITEMS-Relation und die Anzahl der emulierten Browser (*Emulated Browser, EB*), bestimmen sich zum einen die Kardinalitäten aller anderen Relationen. Sie dienen gleichzeitig zur Skalierung des *Benchmarks*.

Das Befüllen der Datenbank mit Testdaten ist in der Spezifikation genau beschrieben und erfolgt zum größten Teil mit Zufallsdaten. Inzwischen bietet die TPC mit WGEN ein Tool an, das kritische Teile des Füllvorgangs übernimmt.

Die Relationen dürfen sowohl horizontal als auch vertikal partitioniert werden, allerdings muß dies transparent für die Anwendung geschehen, die für den Datenbankzugriff nur generische Zugriffsverfahren nutzen darf. Somit soll sichergestellt sein, daß eine beliebige andere Applikation ebenfalls auf die Daten zugreifen kann.

Die Replikation von Daten ist generell zulässig, allerdings müssen diese Replikate allen Anforderungen des ACID-Paradigmas genügen, es darf also nicht zu Inkonsistenzen kommen.

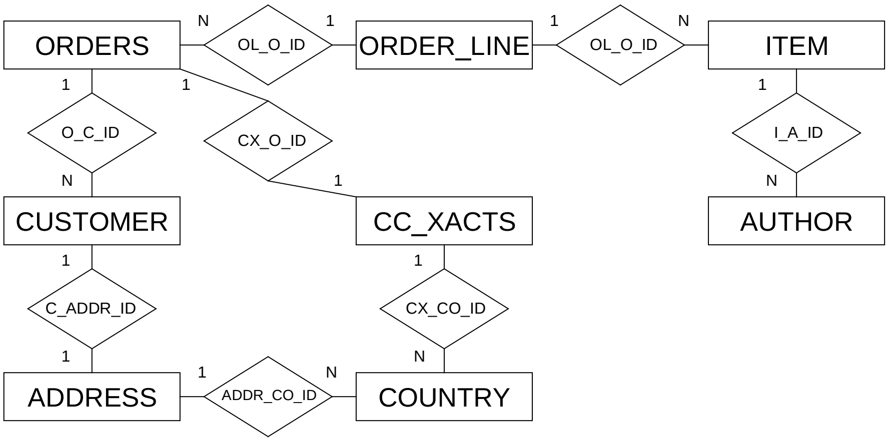

Abb. 2: TPC-Datenbankschema in ERM-Schreibweise

Workload

Die Arbeitslast wird mit Hilfe der bereits erwähnten *Emulated Browsers* erzeugt, welche die Interaktion der Nutzer mit der Anwendung über HTTP nachbilden. Dabei kommt neben ungesicherter Kommunikation auch Datenverschlüsselung mit SSL zum Einsatz.

Eine Nutzerinteraktion beginnt grundsätzlich entweder mit der Identifikation als Kunde oder mit der Neuregistrierung, die auch den Start einer Nutzersitzung beinhalten, welche vom System verwaltet werden müssen.

Anschließend wählt der *emulated browser* aus einer Menge von möglichen *web interactions*. Diese unterteilen sich in zwei Klassen:

Das reine *Browsing*, umfaßt neben der Anzeige der Hauptseite (*Home*) des simulierten Internetshops auch die Anzeige neuer Produkte (*New Products*) und Bestseller (*Best Sellers*), die Anzeige von *Produktdetails (Product Detail*) und die Suche nach Produkten (*Search Request*) sowie das Anzeigen der *Suchergebnisse (Search Display*).

Der Kaufvorgang (*Ordering*) beinhaltet die initiale Kundenregistrierung (*Customer Registration*), das Anzeigen des Inhalts des Warenkorbs (*Cart*), Kauf (*Buy Request*) und Kaufbestätigung (*Buy Confirm*), sowie *Anforderung (Order Inquiry*) und Anzeige (*Order Display*) der letzten Bestellung. Hinzu kommt die Aktualisierung von Produktdaten durch den Administrator (*Admin Request* und *Admin Confirm*).

Die *emulated browser* messen jeweils die Zeit zwischen Anfrage und Erhalt der Seite (Antwortzeit, web interaction response time, WIRT), warten dann eine zufällige Zeit ab, um die Denkzeit des Nutzers nachzubilden. Sie ist exponentialverteilt und liegt im Mittel bei 7 bis 8 Sekunden. Mit Hilfe von vorgegebenen Navigationspfaden und Zufallswerten wird anschließend die nächste Navigationsoperation im Webshop ausgewählt und durchgeführt.

Metrik

Die Geschwindigkeitsmetrik, die der TPC-W-*Benchmark* ermittelt, ist die Anzahl der pro Sekunde verarbeiteten *web interactions* (*web interactions per second*, *WIPS*). Um den für die erzielten Ergebnisse betriebenen Aufwand zu erfassen, wird dieser Wert zusätzlich mit den Gesamtkosten des Systems in Relation gesetzt. Die Ermittlung dieser Kosten ist Teil der Spezifikation, die so errechnete Preis-Performance-Rate trägt die Einheit

Dollar pro WIPS ($/WIPS). Der Vergleich dieser Werte ist nur bei gleich großen Datenbanken zulässig, also einer identischen Menge an Produkten in der ITEMS-Relation.

Um verschiedene Nutzungsschwerpunkte zu bewerten, führt der Benchmark drei unterschiedliche Meßdurchläufe durch, die sich in der Gewichtung der oben aufgeführten *web interactions* unterscheiden, den *web interaction mixes*:

- Der *shopping mix* gewichtet Browsen und Kaufen im Verhältnis 4:1 und liefert den WIPS-Wert.
- Der *browsing mix* betont die Nutzung des Systems für die Informationsgewinnung und besteht zu 95% aus *web interactions* aus der *browse*-Gruppe, mit ihm wird der WIPSb-Wert ermittelt. Dadurch wird besonders das Web-Front End des Systems belastet.
- Um auch die Webshops zu berücksichtigen, die das Glück haben, daß sich ihre Besucher hauptsächlich mit Kaufen beschäftigen, wird der *ordering mix* zur Ermittlung des WIPSo-Wertes herangezogen. Hier werden Browsen und Kaufen gleich gewichtet, was in anbetracht der komplexeren Transaktionstypen beim Kauf zu einer verstärkten Belastung des Datenbanksystems führt.

Um die Auswirkungen von Meßungenauigkeiten zu minimieren und reproduzierbare Resultate zu garantieren, muß jeder Meßvorgang mindestens 30 Minuten dauern.

Die Gewinnung der eigentlichen Meßwerte erfolgt durch Dividieren der Anzahl erfolgreich ausgeführter *Web Interactions* durch die Meßdauer. Damit eine *Web Interaction* zur Bestimmung dieses Wertes herangezogen werden darf, muß sie zusätzlich eine obere Schranke für die maximale Antwortzeit einhalten, die je nach Interaktionstyp zwischen 3 und 20 Sekunden liegt.

Die Ergebnisse müssen zusätzlich zum reinen Meßwert auch in graphischer Form dargestellt werden.

Um eine Einschätzung der Ergebnisse zu ermöglichen, wird ferner die Veröffentlichung von weiteren Meßwerten gefordert, die während der Meßläufe zu ermitteln sind. Dazu gehören die CPU- und Speichernutzung der beteiligten Server, die Paging-Aktivität, Datenbank- und Applikationsserver- Ein- /Ausgabe-Aktivität und Webserverstatistiken.

TPC-W-Implementierungen mit Java-Technologie

Bisherige Implementierungen des TPC-W nutzten fast ausschließlich proprietäre Technologien und verzichten alle auf die Verwendung eines Applikationsservers, vom TPC als *Commerce Server* bezeichnet, für die Implementierung der Geschäftslogik. Der Hauptgrund für diese Entscheidung ist vermutlich bei der einfachen Applikationslogik des TPC-W zu suchen, die noch leicht mit Hilfe von etablierten, aber für die Implementierung komplexer Projekte kaum geeigneten Technologien realisiert werden kann.

Mit der wachsenden Bedeutung von Java für Server (*serversided Java*) haben sich inzwischen wissenschaftliche Einrichtungen mit der Realisierung der TPC-W-Spezifikation mit Java-Technologie beschäftigt.

Aufgrund der Einfachheit der Applikationslogik des TPC-W-*Benchmarks* wurde dazu bisher hauptsächlich Java-Webtechnologie wie *Servlets* und *Java Server Pages (JSPs)* benutzt [13].

Einer der wenigen existierenden Versuche, Teile des TPC-W mit *Enterprise Java Beans (EJBs)* zu verwirklichen, ist unter [14] zu finden. Im Vergleich der EJB-Implementierung mit einer Variante, die direkt auf JDBC aufsetzt, haben sich große Performanceschwächen von EJBs gezeigt, durchschnittlich wurde eine um Faktor drei schlechtere Leistung beobachtet.

Bewertung

Wenn man sich auf der TPC-Webseite [12] nach Benchmarkresultaten für den TPC-W umschaut fällt auf, daß bisher nur sehr wenige Ergebnisse vorliegen. Mag dies zunächst aufgrund des beträchtlichen Aufwands, der für die Implementierung nötig ist, zunächst noch verständlich sein, so erscheint TPC-W im Vergleich zu anderen TPC-Benchmarks bisher nur geringe Bedeutung erlangt zu haben. So brachte es z.B. der TPC-A innerhalb seiner knapp 6-jährigen Lebenszeit von 1989 bis 1995 auf über 300 veröffentlichte Ergebnisse. Über mögliche andere Ursachen für die mangelnde Popularität könnte hier nur gemutmaßt werden.

Die Eignung des TPC-W als reiner Applikationsserverbenchmark erscheint aufgrund seines ganzheitlichen Ansatzes zur Bewertung des Gesamtsystems eher gering. Zwar ist mit Hilfe der zusätzlich zu den WIPS-Werten veröffentlichten Meßwerte ein Rückschluß auf die Performance der einzelnen Server möglich [15], wegen des relativ datenbanklastigen Nutzungsprofils und nicht zuletzt auch wegen der eher einfachen Applikationslogik dürfte es jedoch kaum möglich sein, den Applikationsserver an seine Leistungsgrenzen zu bringen. Letzteres führte auch dazu, daß die bisher existierenden Implementierungen die Anwendungslogik nicht mit Hilfe *von E-*

Commerce-Technologien wie J2EE oder .NET realisieren, sondern i.a. mit proprietären Anwendungen, die möglichst nahe am Front-End plaziert sind. Für komplexere Anwendungen sind derartige maßgeschneiderte Systeme jedoch zu aufwendig in der Realisierung.

Wegen der hohen Datenbankabhängigkeit bezeichnen viele Quellen, darunter auch Sun [18], den TPC-W nicht als Applikations- sondern als Datenbankserverbenchmark.

Als großer Vorteil des TPC-W kann die völlige Unabhängigkeit von bestimmten Technologien gelten, weshalb er sich gut zu deren Vergleich bei ansonsten identischen Systemumgebungen eignen würde, für die natürlich jeweils eine eigene Implementierung erstellt werden muß.

3.2.2 SPECjbb2000 [16]

Die System Performance Evaluation Corporation (SPEC) ist wie die TPC eine „non-profit"-Organisation, in der Hardwarehersteller, Systemintegratoren, aber auch Forschungseinrichtungen wie Universitäten zusammengeschlossen sind. Sie wurde 1988 von einigen Workstation-Herstellern gegründet, welche die Notwendigkeit für einen standardisierten und anerkannten Benchmark erkannt hatten. Inzwischen hat die Organisation 60 zahlende Mitglieder, und seit ihrem Bestehen wurde eine beeindruckende Liste von *Benchmarks* veröffentlicht. Insbesondere die SPEC CPU-*Benchmarks* in ihren zahlreichen Versionen haben große Bedeutung in der Industrie erlangt und sind dort der de facto Standard für den Leistungsvergleich von Workstations. Das Programm umfaßt aber z.B. auch Web- und Mailserverbenchmarks.

Nach der Veröffentlichung des jvm98-Benchmarks zum Vergleich verschiedener *Java Virtual Machines* trat die SPEC mit JBB2000 (*Java Business Benchmark*) zum ersten Mal mit einem Benchmark zur Leistungsbewertung von serverseitigem Java an die Öffentlichkeit. Die Grundlage stammt vom pBOB (Portable Business Object Benchmark) von IBM ab, der hauptsächlich für IBM-interne Tests verwendet wurde, aber auch außerhalb des Unternehmens einige Popularität erlangte. Die SPEC hat den ursprünglichen Code jedoch stark modifiziert, so daß die Ergebnisse nicht vergleichbar sind.

SPECifikation

Der mit JBB2000 verfolgte Ansatz unterscheidet sich deutlich von dem des TPC-W. Der Benchmark ist als eigenständiges Java-Programm erhältlich, und erleichtert so die Leistungsbewertung erheblich, da er schnell und unkompliziert auf beliebigen Systemen eingesetzt werden kann. Er ist vollständig unabhängig und benötigt zur Ausführung lediglich ein *Java Runtime Environment* für die Zielplattform, auch Testdaten und Datenstrukturen zu deren Verwaltung werden automatisch generiert.

JBB2000 simuliert ein komplettes 3-Schichten-System, realisiert jedoch nur die mittlere Schicht mit der Applikationslogik vollständig. Als Anwendungsszenario wurde ein Großhandelsunternehmen gewählt, das über mehrere Lagerhäuser verfügt. Jedes Lagerhaus wird im System durch eine Datenmenge von ca. 25 Megabyte repräsentiert, die im Speicher in Form von Java-Objekten gehalten wird. Auf diesen Objekten können nun die Nutzer verschiedene Operationen ausführen, z.B. neue Bestellungen aufgeben oder den Status existierender Bestellungen abfragen. Hinzu kommen Operationen, die innerhalb der Firma selbst generiert werden, wie das Verarbeiten der Bestellung für die Lieferung, das Überprüfen von Mindestwarenbeständen und das Eintragen von bezahlten Rechnungen. Pro Warenlager wird ein solcher Nutzer durch genau einen Thread repräsentiert, der diese Operationen gemäß festgelegter Wahrscheinlichkeiten auswählt und ausführt.

Da sowohl die Nutzer als auch die Datenhaltung auf dem SUT stattfinden, sind weder Datenbank noch Frontend (z.B. Webserver) für einen Benchmarklauf notwendig. Zudem wird dadurch eine Beeinflussung der Ergebnisse durch Plattenaktivitäten oder Netzwerkverkehr vermieden. SPECjbb2000 orientiert sich unter anderem bei Schema, Operationsprofil und der Generierung der Eingaben am TPC-C-Benchmark, ist jedoch nicht mit diesem vergleichbar.

Benchmarklauf und Performance-Metrik

Jeder Benchmarklauf besteht aus einer Menge von „Punkten" (*points*). Jeder Punkt repräsentiert ein Meßintervall mit einer bestimmten Anzahl von Warenlagern und den durch Threads simulierten Nutzern. Während der Messung wird die Anzahl der Operationen pro Sekunde ermittelt,

Ein kompletter Benchmarklauf besteht nun aus mehreren dieser Meßintervalle mit zunehmender Anzahl N von Lagerhäusern und damit auch Nutzern, und zwar mindestens von 1 Warenlager bis zu 2N Warenlagern, wobei N diejenige Anzahl von Warenlagern ist, bei welcher der Höchstwert des Durchsatzes gemessen wurde. Als Minimalanforderungen müssen Meßwerte für N von 1 bis 8 vorliegen.

Der SPECjbb2000 Meßwert wird aus den Meßwerten bestimmt, indem der Durchschnitt aller „Punkte“ von N bis 2N ermittelt wird. Sollte ein System nicht in der Lage sein, bis zu 2N Warenlager verwalten zu können, kann dennoch ein gültiges Resultat ermittelt werden, indem für die fehlenden Meßwerte ein Ergebnis von 0 Operationen pro Sekunde angenommen wird.

Da dieses Anwendungsszenario einen hohen Grad an Parallelität erlaubt, wird der Punkt N des maximalen Durchsatzes theoretisch dann erreicht, wenn N gleich der Anzahl der Systemprozessoren ist.

Bewertung

Der augenscheinliche Vorteil von SPECjbb2000 ist zunächst die einfache und unmittelbare Durchführbarkeit der Leistungsmessung, es sind im Gegensatz zum TPC-W keine enormen Anfangsinvestitionen in die Implementierung notwendig. Zudem erlaubt der lediglich simulierte Datenbankzugriff eine isolierte Betrachtung der Applikationsserverleistung ohne Beeinflußung durch ein eventuell unterdimensioniertes Datenbanksystem. Dies zusammen mit dem Renommee der SPEC hat sicher nicht unerheblich zur im Vergleich zum TPC-W großen Popularität beigetragen.

Allerdings muß als Kritik der Datenbanksimulation bemerkt werden, daß sie auf jegliche Wartezeiten verzichtet, so daß es zu unrealistisch hohen Durchsatzwerten kommen kann. Zudem werden bei der Ermittlung des Meßwertes die Antwortzeiten nicht berücksichtigt.

Aufgrund der proprietären Implementierung der Anwendungslogik sind auch keine Vergleiche *verschiedener E-Commerce-Technologien* auf identischer Hardware mit SPECjbb2000 möglich, was allerdings ausdrücklich nicht die Zielsetzung bei der Entwicklung war. Vielmehr versteht die SPEC den jbb2000-Benchmark als Möglichkeit zur Evaluierung der verwendeten *Java Virtual Machine*, des *Just-In-Time-Compilers*, der *Garbage-Collection* und in Teilen auch des verwendeten Betriebssystems.

Ein Kritikpunkt ist zudem das Profil der erzeugten Workload, welche einen unrealistisch hohen Parallelitätsgrad erlaubt, so daß die gemessen Skalierbarkeit des Systems im allgemeinen zu optimistisch sein dürfte. Weiterhin wird unter anderem in [17] darauf hingewiesen, daß der verfügbare Speicher bzw. die Größe des Heap der JVM einen enormen Einfluß auf die Ergebnisse hat, der sich in diesem Ausmaß nicht auf reale Anwendungen übertragen läßt.

3.2.3 ECPerf [18],[19]

Mit der Veröffentlichung von *Java 2 Enterprise Edition (J2EE)* durch Sun stand zum ersten Mal eine vollständige Plattform für die Entwicklung serverbasierter Anwendungen mit Java zur Verfügung. Mit Hilfe verteilter Komponenten, den *Enterprise Java Beans*, ist es möglich, komplexe Anwendungen zu implementieren, ohne sich um systemnahe Details wie Transaktionsverwaltung, Kommunikation oder Lastbalancierung kümmern zu müssen. Der Entwickler kann sich so voll auf die Geschäftslogik selbst konzentrieren, während ihm die J2EE-Laufzeitumgebung , der sogenannte *Container*, diese Aufgaben auf Wunsch abnimmt. Beim *Container* handelt es sich im wesentlichen um eine erweiterte *Java Virtual Machine*, welche zusätzlich die notwendige Infrastruktur für die Geschäftobjekte zur Verfügung stellt. Für eine kurze Einführung in *Enterprise Java Beans* sei hier auf [5] verwiesen.

Motivation für einen *EJB-Benchmark*

Da *J2EE* im Gegensatz zu vergleichbaren Frameworks wie .NET lediglich eine Spezifikation ist, welche die Anforderungen an die vom Container zu realisierende Infrastruktur beschreibt, kann jeder Hersteller J2EE-konforme Applikationsserver implementieren. In Folge ist eine große Vielfalt von J2EE-Servern entstanden. Der Vorteil von Java auch auf Servern besteht in der Plattformunabhängigkeit. Für den Kunden bedeutet dies, daß eine Anwendung, die auf die Verwendung proprietärer Erweiterungen verzichtet, auf jedem beliebigen dieser Server mit nur minimalem Anpassungsaufwand einsetzbar ist. Dadurch wird der Kunde unabhängig vom Hersteller des Servers, und ein offener und freier Markt für Komponenten kann sich entwickeln.

Die Unterschiede zwischen den Servern sind also weniger funktionaler Art, da alle zunächst die Spezifikation erfüllen müssen. Natürlich versuchen viele Hersteller, durch eigene Erweiterungen, die möglicherweise Schwächen der Spezifikation ausbügeln und so für die Entwickler durchaus angenehm sind, sich von der Konkurrenz abzuheben und auch die Kunden an ihre Plattform zu binden. Wird der Einsatz dieser Erweiterungen jedoch konsequent vermieden, bleibt als Unterscheidungsmerkmal für J2EE-Server letztlich primär die Performance.

Da Sun die zum Zeitpunkt der Veröffentlichung von J2EE verfügbaren *Benchmarks* für wenig tauglich zum Vergleich verschiedener J2EE-Implementierungen befand, sah man sich als Initiator des Standards gezwungen, selbst einen Benchmark zu entwickeln.

Ziel der Entwicklung war es zum einen, den Käufern eine Hilfsmittel zum Verständnis von Skalierbarkeit und Tuning der Server zu liefern, und zum anderen den Server-Herstellern eine Möglichkeit zu bieten, die Leistung ihrer Produkte zu verbessern und herauszustellen. Dazu wird besonders die Fähigkeit der Server bzgl. Speicherverwaltung, *Connection Pooling*, *Caching* und *Loadbalancing* durch Aktivierung und Passivierung von Komponenten beansprucht. ECPerf zielt nicht auf eine Bewertung von Datenbank oder Präsentationsschicht ab.

Die simulierte Unternehmung

Um dieser Anforderung gerecht zu werden, war eine Geschäftslogik erforderlich, die anspruchsvoller als die aus anderen Benchmarks bekannte ist. Insbesondere wollte man den zunehmenden Trend hin zum *Supply Chain Management*, der automatisierten Fabrikation, Lagerverwaltung und der Integration der Systeme der Zulieferer in das eigene Unternehmen berücksichtigen, die für funktionierende *Just-In-Time*-Fertigung und Lieferung unabdingbar sind.

In Folge entstand das Modell einer global agierenden Unternehmung, die aus vier stark integrierten Geschäftsbereichen oder –domänen besteht: Kunden-, Fertigungs-, Zulieferer- und Unternehmensführungsbereich.

Kundenbereich

Der Kundenbereich (*Customer Domain*) ist zuständig für die Interaktion mit den Kunden. Zu den simulierten Funktionalitäten gehören die Verwaltung von Warenkörben, das Aufgeben von neuen und Ändern sowie Einsehen von alten Bestellungen. Die Kunden werden grob in die Klassen Einzelkunden und Wiederverkäufer unterteilt, die sich hauptsächlich in der durchschnittlichen Auftragsgröße unterscheiden. Direkte Verbindungen bestehen zum einen zur Unternehmensführung, die für die Ermittlung von Rabatten und die Überprüfung der Zahlungsfähigkeit zuständig ist, zum anderen zum Fertigungsbereich, wo die Bestellungen u.U. neue Fertigungsaufträge auslösen.

Fertigungsbereich

Der Schwerpunkt des Fertigungsbereichs (M*anufacturing Domain*) liegt in der Verwaltung der zahlreichen Fabriken mit ihren Produktionsanlagen, von denen es zwei Typen gibt: P*lanned Lines* arbeiten gemäß vorheriger Planung, während *Largerorder Lines* als zusätzliche Kapazitäten für Großaufträge genutzt werden. Die Anlagen werden erst mit dem Eintritt eines Arbeitsauftrags (*Workorder*) aktiv, die jeweils die Produktion einer bestimmten Menge eines Produkts steuert. Bei *Planned Lines* werden die Aufträge aufgrund einer Vorhersage über den zu erwartenden Absatz erzeugt, während Aufträge für L*argerorder Lines* durch eingehende Großaufträge generiert werden. Als erster Schritt bei der Abarbeitung eines Auftrags kommt die Entnahme der für die Produktion benötigten Rohstoffe und Halbfertigerzeugnisse gemäß der Stücklisten für das Produkt. Dadurch kommt es zu Interaktionen mit dem Zulieferbereich, in dem zur Neige gehende Lagerbestände Nachbestellungen auslösen können. Anschließend wird die Produktion simuliert, indem der Weg der Produkte durch die Fertigungsanlage simuliert wird. Der Auftrag „begleitet“ die Produkte dabei und sein Status wird auf jeder Station aktualisiert.

Zuliefererbereich

Im Zuliefererbereich (*Supplier Domain*) werden je nach benötigten Vorprodukten und Dringlichkeit geeignete Zulieferer ausgewählt, an die dann Kaufaufträge versandt werden. Beim Eintreffen einer Lieferung wird der Fertigungsbereich darüber informiert und die Lagerverwaltung aktualisiert.

Unternehmensführungsbereich

Die Verwaltung der Kundendaten wird aus Sicherheits- und Datenschutzgründen ausschließlich an zentraler Stelle im Bereich der Unternehmensführung (*Corporate Domain*) vorgenommen. Hierzu gehört unter anderem die vom Kundenbereich benötigte Information über Kreditrahmen und Rabatte für die einzelnen Kunden.

Implementierung

Jeder Unternehmensbereich ist durch eine Reihe von *Enterprise Java Beans* im System repräsentiert, die mit Hilfe der vom Container zur Verfügung gestellten Infrastruktur zusammenarbeiten, um die geforderte Funktionalität zu erfüllen. Die Datenhaltung erfolgt in logisch getrennten Datenbanken für jede Domäne.

ECPerf kann sowohl zentral auf Einzelrechnern als auch auf verteilten Systemen eingesetzt werden. Im zentralen Fall können alle Teile der Anwendung in einem einzigen *J2EE-Container* oder auch in mehreren installiert werden. Bei beiden Varianten kann eine einzelne Datenbank eingesetzt werden, so daß auf ein XA 2-Phasen-Commit verzichtet werden kann.

Der verteilte Fall prüft die Leistung der Anwendung, wenn die einzelnen Systeme der globalen Unternehmung auch tatsächlich verteilt sind und über ein Netzwerk zusammenarbeiten. Für Transaktionen, die sich über mehrere Domänen erstrecken, ist dann ein XA 2-PC erforderlich.

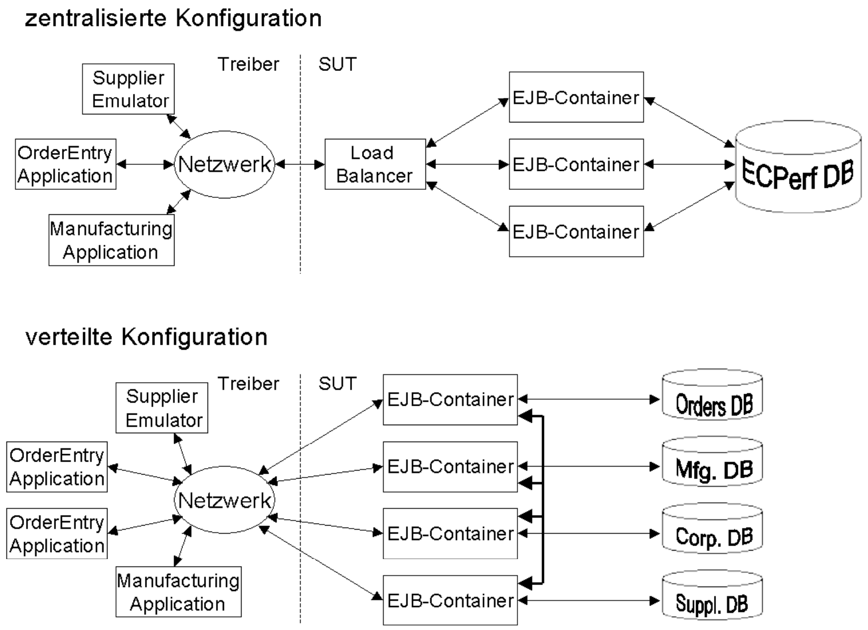

Abb.3: Beispiele für eine zentralisierte bzw. verteilte Konfiguration beim ECPerf

Testumgebung und Treiber

Das *System under test* beinhaltet neben dem oder den J2EE-Server(n) auch die beteiligten Datenbanken und die darauf eingesetzte Software, sowie die zwischen diesen Systemen eingesetzte Netzwerkhardware, inklusive Systemen zur Lastbalancierung.

Zunächst sind vor dem Testlauf die Datenbanken mit Testdaten zu füllen, was mit Hilfe eines beim ECPerf mitgelieferten Tools geschieht.

Nicht zum SUT gehören die Komponenten, welche die Systeme simulieren, die mit der Anwendung interagieren. Zu diesen Treibersystemen gehört der *Supplier Emulator*, welcher die Rolle der Zulieferer übernimmt Hierbei handelt es sich um ein Servlet, das auf einem eigenen Server abläuft. An ihn richtet der Zuliefererbereich seine Bestellungen, die nach einer Wartezeit dann in Form einer Lieferung beantwortet werden.

Während der *Supplier Emulator* eine passive Rolle spielt und nur auf Veranlassung des Systems aktiv wird, ist die *OrderEntry*-Applikation für die Stimulation des Systems zuständig, indem sie zunächst zufällig einen Kunden auswählt oder ggf. einen neuen Kunden im System anmeldet, und dann nach einer vorgegebenen Verteilung zufällig Kundeninteraktionen betreibt. 50% der Aktivitäten sind neue Bestellungen, je 20% entfallen auf das Abfragen des Status und das Ändern von alten Bestellungen, die verbleibenden 10% auf das Abfragen des Kundenstatus. Für jede dieser Aktivitäten ermittelt der Treiber die Antwortzeit des Systems, die einen Schwellwert nicht überschreiten darf, 90% der Anworten müssen innerhalb von zwei Sekunden vorliegen und der Mittelwert der Anwortzeiten darf nicht größer als 2,1 Sekunden sein.

Die Nachbildung der Produktionsanlagen erfolgt durch die *Manufacturing Application,* die für jede Fabrik je eine *Planned Line* und eine *LargeOrder Line* simuliert. Dazu verwaltet sie die momentan bearbeiteten Workorders, die Anzahl der hergestellten Produkte und die benötigte Zeit für die Abarbeitung einer Workorder. Die Durchschnittszeit darf nicht über 5,1 Sekunden liegen, und 90% der Workorders müssen innerhalb von 5 Sekunden abgeschlossen werden.

Testlauf

Die simulierte Belastung der Anwendung wird durch die Aktivität der Treiber gesteuert, welche durch die frei wählbare *injection rate* bestimmt wird. Aus der I*injection Rate* ergibt sich die Anzahl der benötigten Instanzen von OrderEntry- und Manufacturing-Applikation. Um zudem ein sinnvolles Verhältnis zwischen I*njection Rate* und Datenbankgröße zu gewährleisten, sind die Kardinalitäten vieler Relationen funktional durch diese bestimmt.

Die Treiber werden zu Beginn des Testlaufs gestartet. Nach einer mindestens zehnminütigen Anfahrzeit muß ein stabiler Durchsatz erreicht sein, um mit dem eigentlichen Meßintervall, welches mindestens 30 Minuten umfaßt zu beginnen. Im Anschluß muß dieser stabile Durchsatz für weitere fünf Minuten gehalten werden.

Metrik

Für den Kundenbereich wird die Geschwindigkeitsmetrik in Transaktionen pro Minute ermittelt, indem die Anzahl der erfolgreichen Transaktionen durch die Dauer des Meßintervalls dividiert wird. Analog erfolgt für den Produktionsbereich die Bestimmung der Metrik Workorders pro Minute.

Aus diesen beiden Werten wird dann die eigentlichen Metrik für das Gesamtresultat ermittelt, der ECPerf-Wert. Er ergibt sich aus der Summe der beiden Teilergebnisse, und wird in Benchmark Business Operationen pro Minute (BBops) gemessen. Je nach Realisierung des Systems wird dem Ergebnis ein @Std im zentralen Fall oder @Dist für ein verteiltes System nachgestellt. Um die Reproduzierbarkeit des erzielten Ergebnisses zu demonstrieren, muß es durch ein Vergleichsmessung von identischer Länge bestätigt werden, deren Ergebnis höher ausfällt. Zu einem vollständigen Testbericht gehört auch die Veröffentlichung der Verteilung der Anwortzeiten sowohl aus dem Kunden- als auch aus dem Fertigungsbereich, sowie Messungen des Momentandurchsatzes über die gesamte Dauer des Meßintervalls.

Anschließend kann aus dem ECPerf-Wert ein Preis-Leistungswert ermittelt werden, in dem der Gesamtpreis des SUT durch das ECPerf-Ergebnis dividiert wird. Die Einheit ist folglich Dollar pro BBops ($/BBops). Der Gesamtpreis wird nach strikten Regeln ermittelt, und enthält neben Hard- und Software auch die Kosten der Supportleistungen für ein Jahr.

Bewertung

Da die erste Version von ECPerf erst im Juni 2001 vorgestellt wurde, läßt sich wenig über seine zukünftige Bedeutung sagen. Von den zahlreichen Serverherstellern, die bereits Messungen in Suns Testeinrichtungen gemacht haben, hat bisher nur Borland ein Ergebnis veröffentlicht [20]. Über die Gründe dieser Zurückhaltung läßt sich zum jetzigen Zeitpunkt nur spekulieren.

Da ECPerf vom Entwickler des J2EE-Standards selbst kommt, hat er jedoch ganz klar das Potential, eine enorme Bedeutung zu erlangen und auf dem Markt für J2EE-kompatible Applikationsserver zum Standardbenchmark zu avancieren. Allerdings ist ECPerf nicht unabhängig von der Datenbankperformance, so daß für aussagekräftige Ergebnisse ein leistungsfähiger DB-Server zur Verfügung stehen muß. Da Sun den Herstellern von Applikationsservern jedoch die nötigen Systeme für Tests zu Verfügung stellt, wiegt dieser Nachteil weniger schwer. Ein großer Pluspunkt im Vergleich zu TPC-W oder dem JBB2000 ist das äußerst aufwendige Anwendungsszenario, das eine viel höhere Belastung des Applikationsservers verursacht, so daß dieser den Engpaß im Gesamtsystem darstellt und an seine Leistungsgrenze gebracht werden kann.

3.2.4 Brokat-Ansatz [21]

Die *Brokat Technologies AG* ist ein Anbieter von Software und Diensten für *E-Business-Lösungen*, insbesondere im Finanzbereich. Nach Eröffnung des Insolvenzverfahrens am 3. Dezember 2001 wurde nun der Verkauf des Kerngeschäfts an die Firma DataDesign bekannt gegeben [22]. Ein Produkt des Unternehmens ist *der Brokat Twister Applikationsserver*, der unter der Bezeichnung *Brokat Server Technologies* vertrieben wurde. Er hat insbesondere im Bereich des Online-Banking einige Bedeutung erlangt und wird beispielsweise von der Advance Bank eingesetzt. Als proprietäre Technologie nutzt Twister ein eigenes Objektmodell, *das Repository Defined Object (RDO).*

Um Aussagen über die Leistungsfähigkeit, Skalierbarkeit und Zuverlässigkeit machen zu können, benötigte man einen geeigneten Applikationsserverbenchmark. Existierende Benchmarkanwendungen erschienen ungeeignet, da sie aus Sicht der Entwickler von Brokat einige Defizite aufweisen. Insbesondere sind sie alle domänenspezifisch, und zudem ermitteln sie die Leistung der gesamten *E-Commerce-Lösung* und ermöglichen keine isolierte Betrachtung des Applikationsservers. So entstand die prototypische Implementierung eine *Benchmark*-Anastzes für den *Twister*-Server, deren Ziel die Beseitigung dieser Defizite war.

Kernidee Parametrisierbarkeit

Um die Simulation beliebiger Anwendungsfälle zu erlauben, kann der *Benchmark* durch Variation von Parametern modifiziert werden. Sie beziehen sich auf drei Bereiche, den Anfragen, der Session-Verwaltung und der Interaktion mit der von Brokat als *Backend-Schicht* bezeichneten Datenhaltung.

Anfragen werden durch Anzahl und Länge der Ein- und Ausgabeparameter beschrieben, durch das aufzurufende Anwendungsobjekt sowie durch die aus der Anfrage resultierenden Interaktionen mit dem Backend.

Mit *Session*-Typen werden Interaktionsfolgen beschrieben, also eine Reihe von Anfragen, die ein Benutzer zusammenhängend stellt. Durch Gewichtung der *Sessions* kann konfiguriert werden, wie oft ein *Session*-Typ während eines Benchmarklaufs aufgerufen wird.

Die *Backend*-Interaktion wird durch Angabe des verwendeten Informationssystems, der Art der Interaktion, also Lese- oder Schreibzugriff sowie die Datenmenge beschrieben. Um die geforderte Isolierung des Applikationsservers von der Datenhaltung zu ermöglichen, kann hier zusätzlich angegeben werden, ob die Interaktionen tatsächlich durchgeführt oder nur simuliert werden sollen.

Realisierung der Benchmarkanwendung

Um die zur Parametrisierung notwendige Flexibilität zu erreichen, wir die Applikationslogik in vier abstrakte Elementarfunktionen unterteilt, die zur Vereinfachung als separate und nicht verzahnte Aktionen angenommen werden. Als erster Schritt erfolgt die *Überprüfung der Anfrageparameter*, um herauszufinden, ob es eine gültige Anfrage im Sinne des durch Parameter definierten simulierten Szenarios ist.

Anschließend erfolgt die Nachbildung der *Datenverarbeitung*, indem für Applikationsserver typische Operationen durchgeführt werden. Dazu gehören einfache Ganzzahl- und Fließkommaberechnungen, Speicherzugriffe auf Hashtabellen, String- und Tabellenoperationen oder das Aufbauen und Traversieren von Binärbäumen. Der Umfang dieser Operationen, also die benötigte CPU-Zeit, wird ebenfalls durch Parameter vorgegeben.

In Schritt drei wird ermittelt, welche Interaktionen mit dem Backend für die Beantwortung der Anfrage notwendig sind. Je nach Parametrisierung werden sie anschließend an die Backend-Schicht weitergereicht oder nur simuliert, indem die Backend-Integrationsschicht für eine einstellbare Zeit wartet und dann zufallsgenerierte Rückgabewerte liefert.

Der letzte Schritt generiert die Antwort aus den berechneten bzw. aus dem Backend erhaltenen Daten und gibt diese dann an die Kanalanpassungsschicht zurück.

Testumgebung

Da die Aufmerksamkeit des *Benchmarks* dem Applikationsserver gilt, sind Präsentations- und Backendschicht nicht Teil des SUT.

Als Kanal zum Nutzer kommt zum einen HTTP/HTML zum Einsatz, um solche Benutzer zu simulieren, die über Webbrowser auf die Anwendung zugreifen. Als zweiter Kanal wird direkt TCP/IP verwendet, was den Zugriff über proprietäre Anwendungen nachbildet. Die nötige Anpassung erfolgt für HTTP in der durch *eine Java Server Page (JSP)* repräsentierten Präsentationsschicht bzw. durch ein *Kanalanpassungsobjekt (channel alignment object, CAO)* im Applikationsserver. Als *Backend*-System kommt eine relationale Datenbank und ein *LDAP*-Verzeichnisdienst zum Einsatz, falls auf eine Simulation dieser Schicht verzichtet wird. Die Parametrisierung des *Backends* umfaßt das Datenbankschema, wobei außer dem als Integer realisierten Primärschlüssel alle Attribute vom Typ *varchar* sind. Hinzu kommt die Definition der Anfragen, die aus dem Tabellennamen, der Art des Zugriffs und der Anzahl der betroffenen Zeilen besteht

Lastgenerierung

Um die für die Auslastung eines leistungsfähigen Servers notwendige Last erzeugen zu können, ist entweder ein ähnlich leistungsfähiges Treibersystem notwendig, oder die Lasterzeugung ist auf mehrere Maschinen zu verteilen. Dazu wurde ein *Framework* entwickelt, welches das synchrone Initialisieren und Starten der Treibersysteme erlaubt, sowie das anschließende zentrale Auswerten der Ergebnisse. Dieses *Distributed Test Environment (DTE)* wurde zum großen Teil in Java implementiert.

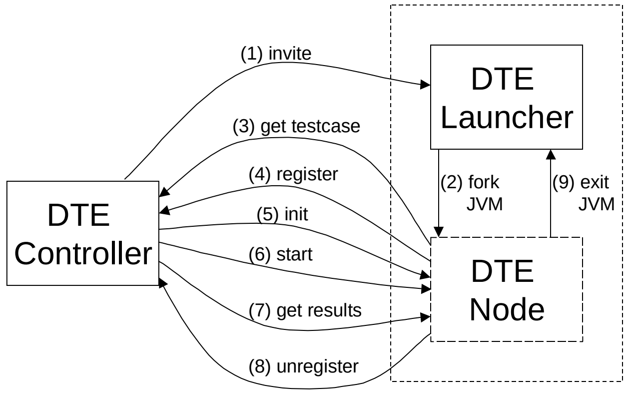

Abb. 4: DTE-Protokoll

Der *DTE-Controller* initiiert den Testlauf durch eine *invite*-Nachricht (1) per *IP-Multicast* innerhalb des LANs. Alle Rechner, auf denen *DTE* aktiv ist, starten daraufhin eine neue *Java Virtual Machine* (2), die beim *Controller* den aktuellen Testfall abholt (3) und sich anschließend als bereit (4) registriert. Anschließend überträgt der *Controller* weitere Parameter des Testlaufs wie Testdauer und Zielsystem (5), wodurch die Initialisierung abgeschlossen ist. Per Start-Signal an alle *DTE-Knoten* wird der Testlauf gestartet (6). Im Anschluß holt der *Controller* die Testdaten bei allen beteiligten Knoten ab (7), die sich dann beim *Controller* abmelden (8) und die JVM beenden (9).

Beispielszenario

Da viele *Twister*-basierte Applikationen aus dem Finanzumfeld kommen, wurde für einen Benchmarklauf die Parametrisierung so gewählt, daß eine typische Online-Banking-Sitzung nachgebildet wird.

Dazu wurden zunächst die vier Anfragen „Anmeldung“, „Kontostandsabfrage“ „Überweisung“ und „Abmeldung“ definiert, die dann zu einer Sitzung zusammengestellt wurde, in der sich der simulierte Nutzer zunächst natürlich anmeldet, seinen Kontostand abfragt, vier Überweisungen tätigt, schließlich noch einmal den Kontostand überprüft und sich dann wieder abmeldet. Jede Anfrage erfordert einen Zugriff auf das *Backend*-System, das für diesen Lauf jedoch nur simuliert wurde, um die volle Leistung des Applikationsservers prüfen zu können.

Testlauf und Metriken

Bei dem im *Sun Benchmark Center Deutschland* durchgeführten Testlauf diente als Hardwareplattform für den Applikationsserver eine *Sun Enterprise E6500* mit 28 400MHz- Prozessoren und 32 Gigabyte Hauptspeicher. Die Lasterzeugung erfolgt auf einer einzelnen Maschine, einer *Sun Enterprise E6000* mit 30 336MHz-Prozessoren und 13 Gigabyte Hauptspeicher. Das *DTE* kam dennoch zum Einsatz, da mehrere Prozesse für die Lasterzeugung synchron gestartet werden mußten.

Die Anzahl der simulierten Benutzer wurde von 100 bis 10000 in Schritten von 500 erhöht und jeweils die durchschnittliche Antwortzeit sowie der Durchsatz gemessen. Um die Skalierbarkeit des Servers zu ermitteln, wurden diese Messungen mit verschiedenen Prozessorzahlen wiederholt.

Die Meßwerte zeigten eine sehr gute Übereinstimmung mit den in Kundenprojekten gewonnenen Erfahrungen.Die Graphik zeigt ein typisches Bild für Durchsatz und Antwortzeit bei großen Serversystemen: bis zur Auslastungsgrenze N* steigt der Durchsatz nahezu linear an, während die Antwortzeit konstant niedrig bleibt. Beim Überschreiten der Auslastungsgrenze steigt die Antwortzeit dann annähernd linear, während der Durchsatz nicht konstant bleibt, sondern wegen des Overheads für die Verwaltung der wartenden Anfragen wieder leicht sinkt.

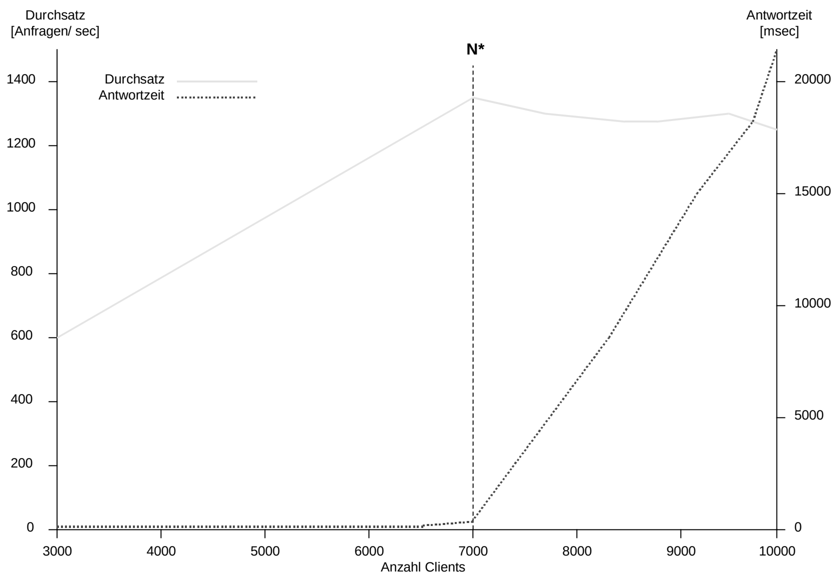

Abb. 5: Durchsatz und Antwortzeit bei steigender Nutzerzahl

Bewertung

Aufgrund seines Prototypenstatus und der verwendeten proprietären Technologie läßt sich der von Brokat entwickelte *Benchmark* nur bedingt mit den anderen vergleichen. Er wurde hier viel mehr wegen seiner interessanten Ansätze vorgestellt, die viele der zuvor erkannten Probleme beim *Benchmarking* von Applikationsservern zu lösen versuchen. Insbesondere die Möglichkeit zur Simulation der Datenhaltungsschicht (*Backend*) durch geeignet gewählte Wartezeiten erlaubt kostengünstigere Messungen, da keine leistungsfähigen Datenbanksysteme zur Verfügung stehen müssen. Auch die unbeeinflußte Bewertung des Applikationsservers durch Verzicht auf diese Wartezeiten ist somit möglich. Durch die Parametrisierung können nicht nur beliebige Nutzungsszenarien simuliert werden, sondern auch gezielt Schwachstellen der Systeme aufgedeckt werden.

Auch wenn die Entwicklung eines Standardbenchmarks nicht das Ziel der Entwickler von Brokat war, wird die Portierbarkeit auf andere Plattformen als recht gut bewertet, da das System weitgehend in Java erstellt wurde.

4. Zusammenfassung

Die Bedeutung von Applikationsservern als mittlerer Schicht in E-Commerce-Lösungen nimmt mit der stetig komplexer werdenden Anwendungslogik zu. Dadurch steigt auch die Bedeutung von geeigneten *Benchmarks*, die eine objektive Leistungsbewertung dieser Systeme insbesondere hinsichtlich Durchsatz, Antwortzeiten und Skalierbarkeit erlauben.

Die vorgestellten Applikationsserverbenchmarks zeigen die Vielfalt der möglichen Ansätze, um die Leistungsfähigkeit der mittleren Schicht einer *E-Commerce-Lösung* zu evaluieren. Jeder hat unterschiedliche Vorzüge und Nachteile aufzuweisen, die seinen Einsatz je nach geplanten Anwendungsszenario und möglichen Aufwand mehr oder weniger sinnvoll erscheinen lassen.

4.1 Gegenüberstellung der Benchmarks

Anhand der zu Beginn aufgeführten Vergleichskriterien sollen nun die verschiedenen Benchmarkansätze direkt gegenübergestellt werden.

4.1.1 Technologieunabhängigkeit

Der *TPC-W-Benchmark* weißt aufgrund seines Spezifikationscharakters naturbedingt die größte Technologieunabhängigkeit auf, da prinzipiell jede für Applikationsserver erhältliche Technologie für die Implementierung verwendet werden kann. SPECjbb2000 hat hier den Nachteil, daß er Performancemessungen mit einer proprietären mitgelieferten Applikationsservertechnologie durchführt, so daß lediglich Vergleiche von Hardwareplattformen, Betriebsystem und *Java Virtual Machine* möglich sind. Der ECPerf-*Benchmark* ist in seiner Zielsetzung auf J2EE-Applikationsserver ausgerichtet und eignet sich daher nicht, um J2EE mit anderen konkurrierenden Technologien zu vergleichen. Der lediglich als Prototyp auf der proprietärten Twister-Technologie existierende Benchmarkansatz von Brokat ermöglicht ebenfalls nur einen Vergleich von Hardware und Laufzeitumgebung, seine Portierbarkeit auf J2EE wird jedoch als gut eingestuft.

4.1.2 Aufwand

Durch Verfügbarkeit als vollständiges Produkt läßt sich der SPECjbb2000-Benchmark relativ problemlos und mit niedrigem Aufwand einsetzen. Durch die Erzeugung der *Workload* auf dem *SUT* und die Simulation der Präsentations- und Datenhaltungsschicht sind zudem keine weiteren Rechner wie Treibersysteme neben dem Applikationsserver notwendig. ECPerf erfordert wie jede J2EE-Applikation eine Anpassung an den jeweiligen Applikationsserver durch Modifikation des *Deployment Descriptors*, der Aufwand hierfür ist jedoch vergleichsweise gering. Die Erstellung einer vollständigen Implementierung der TPC-W-Spezifikation bedeutet nichts weniger als die Entwicklung eines kompletten Softwaresystems, was je nach verwendeter Technologie und Erfahrung der Entwickler einen hohen zeitlichen und finanziellen Aufwand bedeutet. Der nicht für die Allgemeinheit zugängliche Brokat-*Benchmark* kann hier nicht direkt verglichen werden. Allerdings bedeutet die Möglichkeit zur Simulation des *Backend* eine Verringerung des Hardwareaufwands beim Benchmarking, da keine Datenbank zur Verfügung gestellt werden muß.

4.1.3 Isolation

Die Isolation des Applikationsservers gelingt SPECjbb2000 sehr gut, da sämtliche Datenbankaktivitäten nur simuliert stattfinden. ECPerf verwendet zwar ein wenig DB-lastiges Nutzungsprofil und verzichtet weitgehend auf Präsentationsaspekte, kann so aber den Applikationsserver nicht vollständig unabhängig vom Rest des Systems bewerten. Der TPC-W-*Benchmark* weist ein Lastprofil auf, das relativ stark auf die Datenbankleistung angewiesen ist Durch die äußerst einfache Applikationslogik ist demzufolge ein sehr leistungsfähiger DB-Server erforderlich, um den Applikationsserver auszulasten ohne daß die DB zum Engpaß wird.

Der Brokat-Benchmarkansatz ermöglicht durch die optionale Simulation der DB-Zugriffe und den Verzicht auf Präsentationsaspekte im SUT eine gute Isolation des Applikationsservers.

4.1.4 Performanceaspekte

Die Geschwindigkeitsmetrik des TPC-W-Benchmarks betont insbesondere den Durchsatz des Gesamtsystems. Die Antwortzeit geht nur indirekt in die Wertung ein, indem sie bestimmte Werte nicht überschreiten darf, die aber für den verwöhnten Anwender bereits etwas hoch sein dürfen. Durch Variation der Datenbankgröße lassen sich auch Rückschlüsse auf die Skalierbarkeit machen.

Neben der Ermittlung des Durchsatzes sieht SPECjbb2000 bereits bei einem Standard-Benchmarklauf die Messung der Skalierbarkeit vor, indem eine wachsende Zahl von *Threads* gestartet wird.

Der zentrale ECPerf Messwert berücksichtigt den Durchsatz des Systems, die erzielten Antwortzeiten sind in Form eines Graphen jedoch Teil eines vollständigen Berichts und werden ähnlich dem TPC-W als *Constraints* in die Bewertung mit einbezogen..

4.1.5 Unabhängigkeit

Das TPC wurde von einem Zusammenschluß von Datenbankherstellern gegründet, mit dem Ziel der Schaffung einer unabhängigen Organisation für Datenbankbenchmarking. Durch die Kontrollinstanzen innerhalb des TPC, in der jedes Mitglied eine Stimme hat, wird weitestgehend sichergestellt, daß keine Benachteiligung einzelner Produkte entsteht. Die große Akzeptanz früherer TPC-*Benchmarks* untermauert deren Unabhängigkeit.

ECPerf dagegen stammt vom J2EE-Initator Sun, da es jedoch von vornherein nur für den Vergleich verschiedener J2EE-Server gedacht war, ist dies nicht notwendigerweise ein Zeichen für mangelnde Unabhängigkeit. Allerdings vertreibt Sun auch selbst den *iPlanet* Applikationsserver und ist einer der größten Hersteller von Serverhardware, so daß eine gewisse Befangenheit nicht auszuschließen ist.

Ähnlich wie das TPC hat auch die SPEC durch die Entwicklung zahlreicher populärer *Benchmarks* ein enormes Renommee erlangt. Auch hier existieren Kontrollorgane, welche die Einhaltung alle Vorschriften für die Veröffentlichung von Ergebnissen sicherstellen, jedoch erscheint die Arbeit der SPEC weniger transparent als die des TPC.

Der Brokat-Ansatz scheint ein klassisches Beispiel für einen *Benchmark* zu sein, den ein Hersteller seinem Produkt auf den Leib geschneidert hat. Er wurde jedoch nicht für die Verwendung zu Marketingzwecken vorgesehen, sondern für interne Optimierungsarbeiten, so daß die Ergebnisse daher durchaus eine gewisse Aussagekraft haben, aber für einen Vergleich mit anderen Produkten nicht geeignet sind.

4.1.6 Domäne

Die komplexe globale Unternehmung, die der ECPerf simuliert, deckt eine Vielzahl von möglichen Nutzungsszenarien ab und erzeugen eine hohe Belastung des Applikationsservers. Im Vergleich dazu erscheint der aus der Datenbankwelt kommende TPC-W mit seiner einfachen Anwendungslogik und Datenbanklastigkeit weniger für die Bewertung von Applikationsservern geeignet. Auch SPECjbb2000 zeigt sich auf ein einzelnes Szenario beschränkt, so daß seine Ergebnisse nur schlecht auf andere Nutzungsmuster übertragbar sind. Der Brokat-*Benchmark* erreicht mit seinem vielversprechenden Konzept der Parametrisierung einen bisher nicht gekannten Grad an Domänenunabhängigkeit. Allerdings erschwert die Parametrisierbarkeit auch den Vergleich der Ergebnisse, so daß entsprechender Standardszenarien notwenig wären, um hier eine unüberschaubare Vielfalt von Szenarien zu vermeiden.

4.1.7 Popularität

Aufgrund der einfachen Einsetzbarkeit eines fertigen Produkts hat SPECjbb2000 eine große Verbreitung erfahren. Seit der Vorstellung im Juni 2000 wurden bisher 82 Ergebnisse veröffentlicht. Der TPC-W hat im Gegensatz zu SPECjbb2000, und auch im Vergleich mit anderen TPC-*Benchmarks* nur wenig Anklang gefunden. Bisher wurden lediglich sechs Ergebnisse für eine Datenbank mit 10000 Produkten und nur vier für die große Datenbank mit 100000 Büchern veröffentlicht. Als Hauptgrund mag der enorme Aufwand für eine vollständige Implementierung gelten, obwohl dies bei anderen TPC-*Benchmarks* die Akzeptanz nicht geschmälert hat. Vielmehr dürfte die Verfügbarkeit eines fertigen, unkompliziert und kostengünstig einzusetzenden *Benchmarks* wie SPECjbb2000 aus Sicht vieler Unternehmen die Investition in den TCP-W unnötig erscheinen lassen.

Obwohl ECPerf erst im Juni 2001 veröffentlicht wurde, verwundert zunächst, das bisher lediglich ein Ergebnis publiziert wurde, obwohl bereits zahlreiche Hersteller Benchmarkläufe durchgeführt haben. Spekulationen über die Gründe für dieses Verhalten soll hier jedoch kein Raum geboten werden. Der Brokat-Ansatz kann als proprietärer *Benchmark* hier nicht verglichen werden. Als Prototyp für neue Konzepte war er hauptsächlich für interne Verwendung vorgesehen.

4.2 Fazit

Die vorgestellten *Benchmarks* zeigen das ganze Spektrum von Ansätzen für das *Benchmarking* von Applikationsservern, aber auch zugleich die Defizite der bisher existierenden Lösungen.

So kranken viele Benchmarks daran, daß sie keinen Vergleich von verschiedenen *E-Commerce-Technologien ermöglichen*, sondern lediglich Hardware und evtl. das Betriebsystem mit einbeziehen. So bringt SPECjbb2000 seine eigene Laufzeitumgebung und Objektinfrastruktur mit, so daß hauptsächlich die verwendete Java Virtual Machine die Ergebnisse beeinflußt. ECPerf ist von vornherein auf J2EE-Applikationsserver festgelegt. Lediglich der TPC-W würde einen Vergleich verschiedener Technologien erlauben, seine mangelnde Verbreitung, der hohe Aufwand für Implementierung und Testläufe sowie die einfache Anwendungslogik lassen ihn jedoch wenig geeignet erscheinen.

Somit ist also die momentan sehr interessante Evaluierung verschiedener Technologien wie Microsofts .NET und Suns J2EE auf vergleichbarer Hardware und mit identischer Workload bisher nicht praktikabel durchführbar, ein nicht zu unterschätzendes Problem im bevorstehenden Ringen um Marktanteile bei *den E-Commerce-Plattformen.*

Für den Vergleich von J2EE-Servern dürfte sich trotz gewisser Anlaufschwierigkeiten ECPerf in absehbarer Zeit als Standard etablieren, insbesondere da Sun den Herstellern die nötige Infrastruktur für die Messungen zur Verfügung stellt.

Ein weiteres Manko vieler *Benchmarks* liegt darin begründet, daß sie domänenspezifisch sind. Die so erzielten Ergebnisse sind daher nur bedingt auf andere Anwendungsszenarien übertragbar. ECPerf versucht dieses Problem abzuschwächen, indem die simulierte Anwendung sehr viele Szenarien zugleich abdeckt. Als Ideallösung erscheint jedoch ein weitgehend parametrisierbarer *Benchmark* wie ihn Brokat entwickelt hat. Eine Übertragung dieses Konzepts auf einen portablen *Benchmark* würde die Nachbildung von beliebigen Nutzungsszenarien ermöglichen. Allerdings würde der Vergleich von Ergebnissen enorm erschwert, da die so gewonnenen Werte nur mit den jeweils verwendeten Parametern Sinn machen und diese mitveröffentlicht werden müßten. Um eine unüberschaubare Menge von nicht vergleichbaren Ergebnissen zu vermeiden, müßten folglich Standardszenarien entwickelt werden, so daß der Vorteil der Parametrisierbarkeit wieder weitgehend verloren ginge. Für interne Leistungsbewertung durch Kunden oder Serverhersteller ist dieser Ansatz jedoch von großem Interesse, allerdings machen die wirtschaftlichen Schwierigkeiten der Firma Brokat eine Weiterverfolgung dieses vielversprechenden Konzeptes unwahrscheinlich.

Die Isolation der Applikationsservers von den anderen Komponenten der E-Commerce-Lösung stellt eine weitere Herausforderung dar. SPECjbb2000 versucht durch Simulation der DB-Zugriffe im Hauptspeicher dieses Problem zu beheben, durch den Verzicht auf Wartezeiten werden dabei jedoch unrealitisch hohe Ergebnisse erreicht. Auch hier zeigt sich der Brokat-Ansatz innovativ, indem der Datenbankzugriff durch wählbare Wartezeiten realistischer simuliert wird.

Auf dem Weg zu einem Standard-Benchmark für Applikationsserver bleiben also noch ungelöste Probleme, die aufgrund der stetig wachsenden Bedeutung dieser Technologie gelöst werden sollten, um Herstellern und Kunden eine einfache, verläßliche und allgemein akzeptierte Möglichkeit für den Leistungsvergleich der Produkte und Technologien zu erlauben.

Quellenverzeichnis

Alle URLs beziehen sich auf den Stand von Januar 2002.

[1] Eine Linksammlung zu allem was irgendwie mit Benchmarks zu tun hat: http://www.benchmarkresources.com/

[2] Eine kleine FAQ zum Thema Benchmarking: http://www.sysopt.com/benchfaq.html

[3] Website der Standard Performance Evaluation Corporation zum CPU2000 Benchmark: http://www.spec.org/osg/cpu2000/

[4] Website der Business Applications Performance Corporation http://www.bapco.com/

[5] Java 2 Enterprise Edition - Enterprise Java Beans & Enterprise Information Systems http://wwwdbis.informatik.uni-kl.de/courses/seminar/SS2001/ausarbeitung6.ps

[6] http://www.etestinglabs.com/benchmarks/webbench/webbench.asp

[7] http://www.mindcraft.com/webstone/

[8] Website der Standard Performance Evaluation Corporation zum web99 Benchmark: http://www.spec.org/osg/web99/

[9] Website des Transaction Processing Performance Council: http://www.tpc.org

[10] Alles über J2EE vom Erfinder Sun: http://java.sun.com/j2ee/

[11] The Windows .NET Server Application Environment: http://www.msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnetserv/html/windowsnetserver.asp

[12] Website des Transaction Processing Performance Council zum TPC-W Benchmark: http://www.tpc.org/tpcw/

[13] Eine Implementierung des TPC-W mit Java Technologie vom Department of Electrical and Computer Engineering and Computer Sciences Department der Universität von Wisconsin - Madison http://www.ece.wisc.edu/~pharm/tpcw.shtml

[14] Eine Vergleich der Leistung von TPC-W-Implementierungen mit Enterprise Java Beans bzw. direkter Verwendung von J2EE: http://gi-akvo.decrc.abb.de/Workshop2000/HettelZinkEJBPerformance.pdf

[15] TPC-W: Benchmarking an E-Commerce Solution - Wayne D. Smith, Intel Corporation

[16] Website der Standard Performance Evaluation Corporation zum SPECjbb2000 Benchmark: http://www.spec.org/osg/jbb2000/

[17] Hewlett Packard zu den SPEC JBB2000 und SPEC JVM98 Benchmarks: http://www.hp.com/products1/unix/java/infolibrary/jdb2000_jvm98.html

[18] Website von Sun zum ECPerf Benchmark: http://java.sun.com/j2ee/ecperf/

[19] The Server Side: http://ecperf.theserverside.com/ecperf/

[20] Borland veröffentlicht das erste ECPerf Ergebnis http://www.borland.com/about/news/ecperf_result.html

[21] Leistungsbewertung für Application Server-Technologie: ein parametrisierbarer Benchmark-Ansatz am Beispiel des Brokat Twister Application Servers
Tagungsband der GI/OCG-Jahrestagung Informatik 2001, Wien, Sept. 2001, pp. 909-915

[22] DataDesign übernimmt Brokats Bankensoftware http://www.heise.de/newsticker/data/anw-07.01.02-001/