Proc. Workshop "Methoden und Werkzeuge zur Datenbankadministration", Darmstadt, März 1996, in: Datenbank-Rundbrief Nr. 17, GI-Fachgruppe Datenbanken, Mai 1996

Automatisches, zielorientiertes Performance Tuning von Transaktionssystemen

Erhard Rahm

Univ. Leipzig, Institut für Informatik, 04109 Leipzig
rahm@informatik.uni-leipzig.de

1 Einführung

Die Überwachung und Steuerung des Leistungsverhaltens derzeitiger Transaktionssysteme und Datenbanksysteme weist eine Reihe gravierender Schwächen auf:

- weitgehend manuelles Performance Tuning durch Systemverwalter (z.B. DBA)
- komplizierte Systemverwaltung durch Vielzahl von internen Parametern
- mangelnde Abstimmung zwischen Resource-Managern, insbesondere zwischen DBS, TP-Monitor und Betriebssystem (z.B. verwenden TP-Monitore Transaktionsprioritäten, Betriebssysteme dagegen Prozeß-Prioritäten; das DBS kennt meist gar keine Prioritäten)*
- unzureichende Unterscheidung verschiedener Lastgruppen im DBS bei der Zuteilung von Betriebsmitteln (Sperren, Pufferplatz, etc.).

Diese Probleme stellen sich bereits im Falle zentralisierter Transaktionssysteme, welche u.a. aus einem DBS, einem TP-Monitor sowie den Anwendungsprogrammen bestehen. Eine Verschärfung der Lage tritt in verteilten Transaktionssystemen ein, wenn Transaktions- und Datenbankfunktionen auf mehreren Verarbeitungsrechnern auszuführen sind. Hier sind zusätzliche Festlegungen zu treffen, insbesondere bezüglich der Rechnerzuordnung von Transaktionsaufträgen (Lastverteilung), von Anwendungsprogrammen sowie der Daten, so daß die Kapazität aller Rechner gut genutzt werden kann. Ein weiteres Problem liegt in der steigenden Komplexität und Heterogenität der Lastprofile, da neben einfacheren OLTP-Anwendungen zunehmend daten- und berechnungsintensive Anfragen für Decision Support auf denselben Daten zu bearbeiten sind.

Die Bewältigung dieser Probleme erfordert einen automatischen, selbstoptimierenden Ansatz zur Performance-Kontrolle von Transaktions- und Datenbanksystemen. Dabei wird die Verarbeitung systemseitig ständig überwacht und analysiert, so daß Problemsituationen unmittelbar erkannt werden können. Kontrollparameter des Systems sind automatisch einzustellen und in Abhängigkeit des aktuellen Systemzustands anzupas-

* Eine Konsequenz daraus ist, daß die DB-Operationen von Transaktionen niedriger Priorität auch die üblicherweise hohe Priorität von DBS-Prozessen erhalten und somit wichtigere Transaktionen stark behindern können.

sen, z.B. zur Behandlung von Leistungsproblemen. Die Realisierung eines solchen Ansatzes ist sehr aufwendig und wirft eine Vielzahl von Problemen auf, u.a.

- Wie werden Leistungsprobleme automatisch erkannt?
- Wie werden die Ursachen für Leistungsprobleme ermittelt?
- Kann durch Änderung von Kontrollparametern eine Verbesserung der Situation erwartet werden?
- Welche Kontrollparameter werden wann und wie angepaßt?

Daneben muß gewährleistet sein, daß der automatische Kontrollansatz auch in Überlastsituationen stabil arbeitet und nur vergleichsweise geringen Overhead verursacht.

Eine Reihe von Forschungsarbeiten befaßte sich in letzter Zeit mit dieser Thematik, allerdings meist nur bezüglich eines relativ kleinen Teilbereiches (z.B. Anpassung des Multiprogramming Levels zur Begrenzung von Sperrkonflikten). Unser Ansatz zur automatischen Performance-Kontrolle ist u.a. durch folgende Merkmale gekennzeichnet:

- Zur Vereinfachung der Systemadministration wird ein *zielorientierter* Kontrollansatz verfolgt (goal-oriented resource allocation) [Ra89, NFC92, FNGD93, BMCL94, BCL96]. Dabei spezifiziert der Systemverwalter lediglich *was* die Performance-Ziele sind, jedoch *nicht* mehr, *wie* diese Ziele erreicht werden sollen. Performance-Ziele entsprechen externen Leistungsanforderungen für einzelne Lastgruppen (z.B. Antwortzeitrestriktionen von Transaktionstypen) und werden automatisch in geeignete Einstellungen für interne Kontrollparameter abgebildet.
- Es soll ein umfassender Ansatz unterstützt werden, der eine integrierte Behandlung unterschiedlicher Engpaßtypen ermöglicht. Damit soll auch eine koordinierte Betriebsmittelvergabe zwischen einzelnen Teilsystemen wie TP-Monitor, DBS und Betriebssystem ermöglicht werden.
- Bei der Behandlung von Leistungsproblemen wird unterschieden zwischen vollständigen und partiellen Überlastsituationen. Im ersten Fall leiden praktisch alle aktiven Lastgruppen unter der Überlast, während im zweiten Fall die Probleme auf eine Teilmenge der Lastgruppen beschränkt ist. Die Berücksichtigung partieller Überlastsituationen verlangt zwar komplexere Kontrollverfahren, eröffnet jedoch auch gezieltere Steuerungsmöglichkeiten.

Im nächsten Kapitel beschreiben wir eine Grobarchitektur für eine derartige Performance-Kontrolle. Nähere Untersuchungen erfolgten bisher für die Realisierung im zentralen Fall, wozu auch ein erstes Simulationsmodell entwickelt wurde. Auf die dabei verwendeten Verfahren wird in Kap. 3 eingegangen; Kap. 4 beschreibt erste Simulationsergebnisse.

2 Kontrollarchitektur

Zur Transaktions- und Datenbankverarbeitung unterstellen wir ein lokal verteiltes Server-System, bestehend aus mehreren über ein Hochgeschwindigkeitsnetz lose gekoppelten Verarbeitungsrechnern. Auf jedem Verarbeitungsrechner seien u.a. ein TP-Monitor und DBS zur Bearbeitung von Transaktionsaufträgen bzw. DB-Operationen aktiv. Die Verwaltung von Hardware-Ressourcen wir CPU, Hauptspeicher etc. erfolgt durch entsprechende Resource-Manager des Betriebssystems (CPU-Dispatcher, Speicherverwaltung, ...). Die DBS-Instanzen der einzelnen Rechner bilden ein Paralleles DBS gemäß dem Shared-Nothing- oder Shared-Disk-Ansatz [DG92]. Durch eine enge Kooperation der DBS-Instanzen wird Anwendungsprogrammen sowie Endbenutzern eine DB-Verarbeitung wie im zentralen Fall ermöglicht (single system image).

Wie in Abb. 1 gezeigt, erweitern wir zur Performance-Kontrolle eine solche Umgebung vor allem um zwei Funktionen: eine lokale Performance-Kontrolle in jedem Verarbeitungsrechner sowie eine globale Performance-Kontrolle. In beiden Fällen handelt es sich im wesentlichen um ein Kontrollprogramm, das in periodischen Zeitabständen ausgeführt wird bzw. wenn bestimmte Ausnahmebedingungen eintreten. Das Kontrollprogramm greift auf Monitordaten über den aktuellen Systemzustand (Istzustand) zu und wertet diese aus, um zu entscheiden, ob eine Abweichung vom Sollzustand vorliegt (z.B. Verfehlung von Leistungszielen). Liegt eine solche Situation vor, wird versucht, die Ursache dafür zu ermitteln und das Problem durch Einleiten entsprechender Korrekturmaßnahmen zu beheben. Die Wirkung der ergriffenen Maßnahmen kann durch Analyse neuer Monitordaten überprüft werden. Die Vorgehensweise entspricht somit der Verwendung von Regelkreisen (feedback loops) zur adaptiven Systemkontrolle.

Aufgabe der *lokalen Performance-Kontrolle* ist, die Verarbeitung innerhalb eines Rechners zu überwachen. Dazu steht diese Komponente mit Monitor- und Scheduling-Komponenten in Verbindung, die in existierenden Teilsystemen (Betriebssystem, TP-Monitor, DBS) typischerweise bereits zum Großteil vorhanden sind. Für diese Komponenten ist jetzt jedoch eine geeignete Schnittstelle zur lokalen Performance-Kontrolle zu unterstützen, über die ein Austausch von Monitordaten und Kontrollanweisungen erfolgt. Insbesondere liegt es in der Verantwortung der lokalen Performance-Kontrolle, die einzelnen Teilsysteme über diese Schnittstellen aufeinander abzustimmen. Die lokale Performance-Kontrolle steht daneben mit der globalen Performance-Kontrolle in Verbindung, um sie mit Informationen über den lokalen Systemzustand zu versorgen bzw. um Probleme, die durch Anpassung lokaler Kontrollparameter nicht behoben werden konnten, weiterzumelden.

Die *globale Performance-Kontrolle* ist verantwortlich für die Überwachung des gesamten Mehrrechnersystems. Wir gehen von einer zentralisierten Realisierung dieser Funktion aus, das heißt, die globale Kontrollschleife wird in einer Komponente (z.B. in

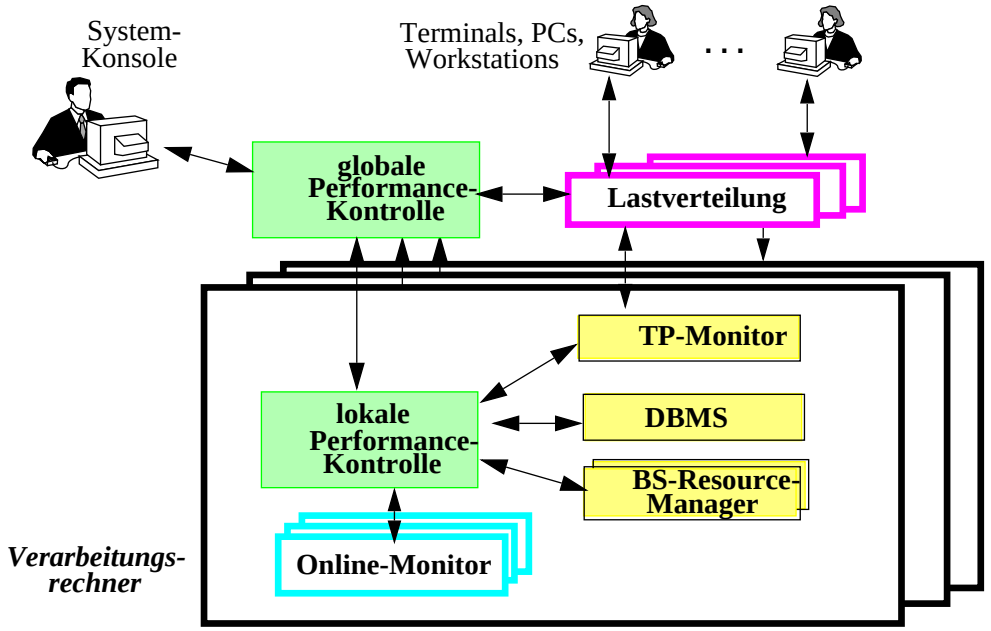

Abb. 1: Grobarchitektur zur automatischen Performance-Kontrolle

einem Prozeß) ausgeführt. Im Gegensatz zu einer verteilten Realisierung wird damit die Entscheidung über globale Korrekturmaßnahmen erheblich vereinfacht. Daneben kann die globale Performance-Kontrolle die zentrale Schnittstelle zur Systemverwaltung bilden, so daß über eine Systemkonsole Kommandos bezüglich des gesamten Systems spezifiziert werden können (Änderung von Leistungszielen, Anforderung von Informationen, etc.). Die Schnittstelle zur lokalen Performance-Kontrolle jedes Rechners wird zur Propagierung von Systemkommandos bzw. eigenen Kontrollanweisungen sowie zum Empfang lokaler Statusinformationen verwendet.

Neben der globalen Systemüberwachung ist auch die Lastverteilung (Transaktions-Routing) eine Aufgabe der globalen Performance-Kontrolle. Diese Teilfunktion wurde in Abb. 1 separat gezeigt, da sie durchaus verteilt realisiert werden kann. Die Lastverteiler können dabei, wie auch die globale Kontrollkomponente, auf dedizierten Front-End-Rechnern oder auf den Verarbeitungsrechnern selbst ablaufen. Aufgabe der Lastverteilung ist vor allem Ein- und Ausgabenachrichten zwischen Terminals und Verarbeitungsrechnern zu transferieren. Die Zuordnung von Transaktionsaufträgen zu Verarbeitungsrechnern erfolgt nach einer gemeinsamen Routing-Strategie, die von der globalen Kontrollkomponente angepaßt werden kann. Die Lastverteiler können daneben die Ankunftsraten sowie die Transaktionsbearbeitungszeiten bestimmen und diese Information der globalen Kontrollkomponente übergeben.

Der skizzierte Architekturvorschlag sieht also Kontrollmaßnahmen auf drei Ebenen vor: durch teilsystemspezifische Scheduling-Komponenten, eine lokale Performance-Kontrolle pro Verarbeitungsrechner sowie eine globale Performance-Kontrolle*. Diese Komponenten kooperieren eng miteinander, um globale Leistungsziele erreichen zu können. Ein adaptiver Ansatz wird durch Nutzung von Feedback-Schleifen innerhalb der lokalen und globalen Performance-Kontrolle erreicht. Wesentlich für die Effizienz des Ansatzes ist die Beschränkung hinsichtlich der Informationen, die gesammelt und ausgewertet werden. Ein kritischer Parameter ist auch, in welcher Häufigkeit die lokalen und globalen Kontrollschleifen ausgeführt werden. Hier gilt es einen guten Kompromiß zwischen Effizienz (seltene Ausführung) und Reagibilität (häufige Ausführung) zu finden. Eine einfache Administration wird durch die weitgehend automatische Anpassung von Kontrollparametern sowie der Bereitstellung einer zentralen Administrationsschnittstelle unterstützt.

3 Lokale Performance-Kontrolle

Zur Evaluierung unterschiedlicher Strategien zur Engpaßbestimmung und -behebung wurde ein erstes Simulationsmodell eines zentralisierten Transaktionssystems entwikkelt, wobei die in Kap. 2 eingeführte Kontrollarchitektur unterstützt wird. Zur lokalen Performance-Kontrolle werden zur Zeit vier Engpaßtypen berücksichtigt, nämlich CPU-, Sperr-, Platten- und MPL-Engpässe. Ein MPL-Engpaß bezieht sich dabei auf initiale Wartezeiten in einer Eingangswarteschlange, die entstehen, wenn bei Eintreffen einer Transaktion bzw. Anfrage die vorgesehene maximale Anzahl gleichzeitiger Aktivierungen (Multiprogramming Level) bereits erreicht ist. Als Performance-Ziele werden mittlere Antwortzeitschranken pro Transaktions- bzw. Anfragetyp verfolgt. Folgende Daten werden u.a. dynamisch zur Performance-Kontrolle gesammelt:

- pro Lastgruppe mittlere Antwortzeiten sowie Zusammensetzung der Antwortzeiten (Anteile für CPU, Lock-Wait, Plattenzugriffe, MPL-Queueing)
- Auslastung von CPUs und Platten
- MPL-Auslastung und Länge der Eingangswarteschlange
- Sperrstatistiken (s.u.).

Systemanalyse und Engpaßerkennung

Bei jeder Aktivierung der lokalen Performance-Kontrolle wird durch Vergleich der Bearbeitungszeiten von kürzlich beendeten sowie laufenden Transaktionen geprüft, ob die mittleren Antwortziele eingehalten werden. Solange dies der Fall ist, erfolgen keine An-

* Für zentralisierte Transaktionssysteme entfällt natürlich die globale Performance-Kontrolle; Aufgaben wie die Realisierung einer zentralen Administrationsschnittstelle werden dann von der lokalen Performance-Kontrolle übernommen.

passungen der Kontrollparameter. Liegen Zielverfehlungen vor, wird ermittelt, ob das Problem auf eine kleinere Teilmenge der Lastgruppen beschränkt ist (*partielle Überlast*) oder praktisch alle Lastgruppen ihre Leistungsziele verfehlen (*vollständige Überlast*).

Zur Engpaßerkennung werden zwei unterschiedliche Ansätze verfolgt, die im Rahmen der Simulationsstudie miteinander verglichen werden sollen:

1. *Verwendung ressourcen-spezifischer, lastgruppenunabhängiger Indikatoren (CPU-, Platten- und MPL-Auslastung bzw. Warteschlangenlängen)*
   Hier repräsentiert eine bestimmte Ressource einen Engpaß, wenn der entsprechende Indikator einen kritischen Schwellwert übersteigt (z.B. CPU-Auslastung > 90%). Zur Indikation eines Sperrengpasses werden verschiedene Metriken eingesetzt, u.a. die sogenannte Konfliktrate aus [MW92], welche als Verhältnis zwischen der Anzahl aller gesetzten Sperren und der Anzahl von aktiven Transaktionen gesetzten Sperren definiert ist. Nach [MW92] soll ein Wert von über 1.3 einen Sperrengpaß signalisieren, unabhängig von der jeweiligen Datenbank und Anwendungslast.
2. *Analyse der Antwortzeitzusammensetzung*
   Für die "leidenden" Lastgruppen wird die Antwortzeitzusammensetzung analysiert. Übersteigt für eine Ressource der entsprechende Antwortzeitanteil einen bestimmten Schwellwert (z.B. Antwortzeitanteil für Sperren > 30%), wird diese Ressource als Engpaß eingestuft.

Ansatz 2 erlaubt vor allem die gezielte Behandlung partieller Überlastsituationen, während bei vollständiger Überlast Ansatz 1 eine schnelle Engpaßerkennung verspricht.

Eine Komplikation bei der Engpaßbestimmung besteht darin, daß zu einem Zeitpunkt mehrere Engpässe vorliegen können. Dies ist häufig darauf zurückzuführen, daß ein primärer Engpaß - z.B. Plattenüberlastung - einen Folgeengpaß verursachen kann, z.B. einen MPL-Engpaß. Hier ist es wesentlich, den primären Engpaß zu erkennen und zu behandeln*. Um dies zu erreichen, behandeln wir die Engpässe in der folgenden Reihenfolge:

1. Plattenengpässe
2. Sperrengpässe
3. CPU-Engpässe
4. MPL-Engpässe.

Somit werden Sperrengpässe nur behandelt, wenn kein Plattenengpaß (mehr) vorliegt; ein MPL-Engpaß wird nur behandelt - durch Erhöhen des MPLs - wenn keiner der drei

* Dies wäre nicht möglich bei unkoordinierten Kontrollaktionen in den einzelnen Teilsystemen. Z.B. würde die Behandlung eines MPL-Engpasses im TP-Monitor durch Erhöhen des MPL-Wertes den primären Plattenengpaß verschärfen statt reduzieren.

anderen Engpässe mehr gegeben ist. Diese Vorgehensweise berücksichtigt typische Abhängigkeiten zwischen den einzelnen Engpaßtypen (ein Plattenengpaß kann einen MPL-Engpaß sowie - aufgrund verlängerter Antwortzeiten - einen Sperrengpaß verursachen, jedoch nicht umgekehrt).

Korrekturmaßnahmen

Die wichtigsten Maßnahmen zur Behebung von Engpässen sind zur Zeit die Einstellung von MPL-Werten sowie die Anpassung von Transaktionsprioritäten.

Bei der MPL-Anpassung erfolgt eine Reduzierung im Falle von Platten-, Sperr- oder CPU-Engpässen sowie eine Erhöhung für (ausschließliche) MPL-Engpässe. Dabei besteht die Gefahr, daß abwechselnd eine Reduzierung und Erhöhung von MPL-Werten erfolgt, so daß der primäre Engpaß nur verlagert, jedoch nicht behoben wird. Um dies zu verhindern, erfolgt eine MPL-Erhöhung generell nur, wenn die Auslastungsindikatoren bezüglich CPU und Platten sowie die gewählte Sperrmetrik einen kritischen Wert nicht überschreiten.

Zur MPL-Anpassung kann entweder ein Gesamt-MPL für die Anzahl Aktivierungen aller Lastgruppen eingestellt oder eine sogenannte *MPL-Matrix* eingesetzt werden. In dieser Matrix wird für je zwei Lastgruppen i und j die maximale Anzahl gleichzeitiger Aktivierungen angegeben, wobei der Eintrag für i=j das Maximum innerhalb einer Gruppe festlegt. Mit der MPL-Matrix kann gezielt auf partielle Überlastprobleme reagiert werden, z.B. aufgrund von Sperrengpässen. Denn wenn die Sperrkonflikte nur zwischen Aktivierungen desselben Transaktionstyps auftreten, kann dem durch Absenken des typspezifischen MPL-Wertes Rechnung getragen werden. Bestehen die Konflikte vor allem mit einem anderen Transaktionstyp, erfolgt ein Absenken des gemeinsamen MPL-Wertes*. Die Reduzierung des Gesamt-MPL-Wertes ist demgegenüber wesentlich ungenauer und kann zu unnötigen Verzögerungen für unbeteiligte Transaktionstypen führen.

Die Anpassung von Transaktionsprioritäten ist neben der Verwendung lastgruppenspezifischer MPL-Werte ein weiterer Ansatz zur Behandlung partieller Überlastsituationen. Dabei wird die Priorität von Transaktionen "leidender" Lastgruppen gegenüber Lastgruppen, die ihre Ziele erreicht haben, erhöht. Prioritäten werden zur Zeit bei der Eingangswarteschlange sowie der CPU-Zuteilung berücksichtigt. Zur Prioritätenvergabe und -anpassung wurden verschiedene Verfahren realisiert, die simulativ miteinander verglichen werden sollen.

---

* Zur Bestimmung der wesentlichen Konfliktpartner wird in einer Sperrmatrix die Häufigkeit und Dauer von Sperrkonflikten zwischen den einzelnen Lastgruppen geführt.

4 Erste Simulationsergebnisse

In einem ersten Simulationsexperiment wird lediglich die Behandlung von CPU- und MPL-Engpässen für eine homogene Last mit einem Transaktionstyp betrachtet. Dabei entfallen im Mittel pro Transaktion 40 Objektzugriffe und etwa 330 000 Instruktionen. Zur Erleichterung von CPU-Engpässen wurden eine geringe CPU-Leistung eingestellt (2 Prozessoren mit je 2,5 MIPS) sowie durch eine entsprechende Konfigurierung Sperr- und Plattenengpässe umgangen. Im Einbenutzerbetrieb beträgt die mittlere Antwortzeit etwa 0,5 s; das Antwortzeitziel (Goal) für den Mehrbenutzerbetrieb wurde auf 1,5 s eingestellt. Die lokale Performance-Kontrolle wird alle 2 s aktiviert und erfordert einen Bearbeitungs-Overhead von durchschnittlich 100 000 Instruktionen sowie zusätzlichen 100 000 Instruktionen falls Korrekturmaßnahmen eingeleitet werden. Die Engpaßerkennung erfolgt über eine Analyse der Antwortzeitzusammensetzung. Ein CPU- bzw. MPL-Engpaß wird unterstellt, wenn das Antwortzeitziel verfehlt wird und CPU-Wartezeiten bzw. MPL-Wartezeiten (in der Eingangswarteschlange) 20% der Antwortzeit übersteigen. Im folgenden werden zunächst Ergebnisse für "stabile" Lastkonstellationen mit einer festen Ankunftsrate betrachtet; danach eine "variable" Lastsituation, bei der zu bestimmten Zeitpunkten Lastspitzen mit Überlastung des Systems auftreten.

Abb. 2 zeigt die Ergebnisse mit statischer und dynamischer (adaptiver) MPL-Einstellung für zwei Ankunftsraten, nämlich 12 und 15 TPS (Transaktionen pro Sekunde). Die erste Ankunftsrate bewirkt eine CPU-Auslastung von ca. 80%; bei 15 TPS wird das System fast vollständig ausgelastet (99%). Die Ankunftsrate von 12 TPS (oder niedriger) ist auch bei statischer MPL-Einstellung weitgehend unproblematisch. Nur MPL-Werte kleiner als 10 bewirken, daß die CPU-Leistung nicht voll genutzt wird und MPL-Engpässe entstehen. Die dynamische Einstellung wurde für unterschiedliche MPL-Startwerte analysiert und konnte unabhängig von dem gewählten Startwert (zwischen 1 und 100) das

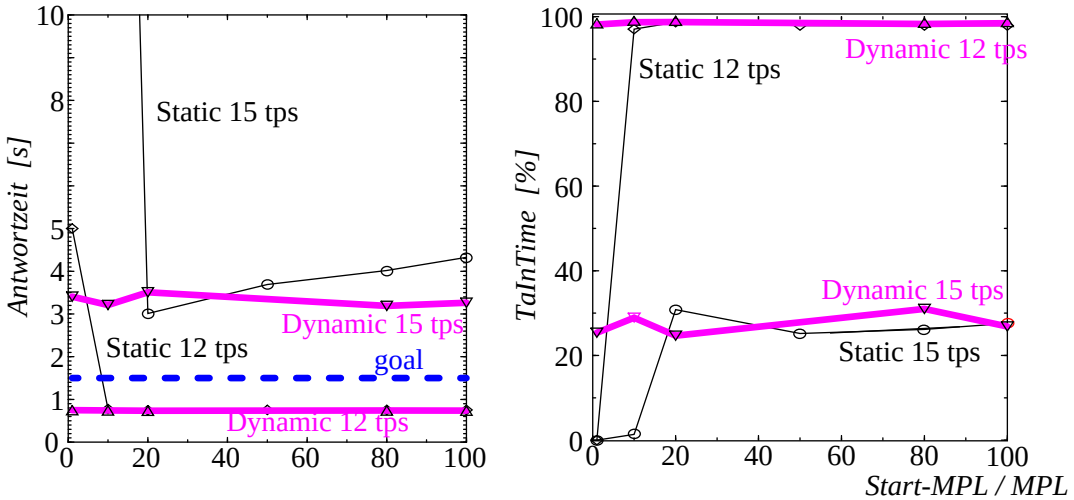

Abb. 2: Stabile Last: mittl. Antwortzeiten und In-Time-Anteile

optimale Ergebnis erreichen. Der Anteil von Transaktionen, welche ihr Antwortzeitziel erreichten (rechtes Diagramm in Abb. 2), war praktisch 100%. Im Hochlastfall von 15 TPS konnte dagegen das Antwortzeitziel meist nicht erreicht werden. Aber auch hier war die dynamische MPL-Einstellung in der Lage, unabhängig vom Startwert das mit optimaler statischer MPL-Einstellung erzielte Ergebnis zu erreichen*. Natürlich ist es in der Praxis schwierig, den optimalen statischen MPL-Wert zu finden, so daß meist mit schlechteren Antwortzeiten als mit dynamischer MPL-Einstellung zu rechnen ist. Die statische Einstellung "optimaler" MPL-Werte ist noch schwieriger bei schwankendem Lastprofil und/oder heterogenen Lasten mit mehreren Lastgruppen, so daß in diesen Fällen die Vorteile einer dynamischen MPL-Kontrolle stärker zur Geltung kommen sollten.

Zur Untersuchung von Lastschwankungen wurden für den betrachteten Transaktionstyp eine Grundlast von 12 TPS eingestellt sowie mehrere Lastspitzen von 25 TPS, welche zu einer völligen Überlastung des Systems führen. Abb. 3 zeigt die Entwicklung der Antwortzeiten sowie MPL-Einstellungen über einen Beobachtungszeitraum von 100 s. Man erkennt, daß der zunächst eingestellte MPL-Wert von 100 (rechte y-Achse) unverändert bleibt, solange das Antwortzeitziel von 1,5 s eingehalten wird. Die erste Lastspitze zum Zeitpunkt 100 führt zu einem CPU-Engpaß, dessen Behandlung in wenigen Schritten eine MPL-Reduzierung auf 17 bewirkte. Diese schnelle MPL-Anpassung war möglich, da hierfür die (niedrige) MPL-Auslastung der zurückliegenden Beobachtungs-

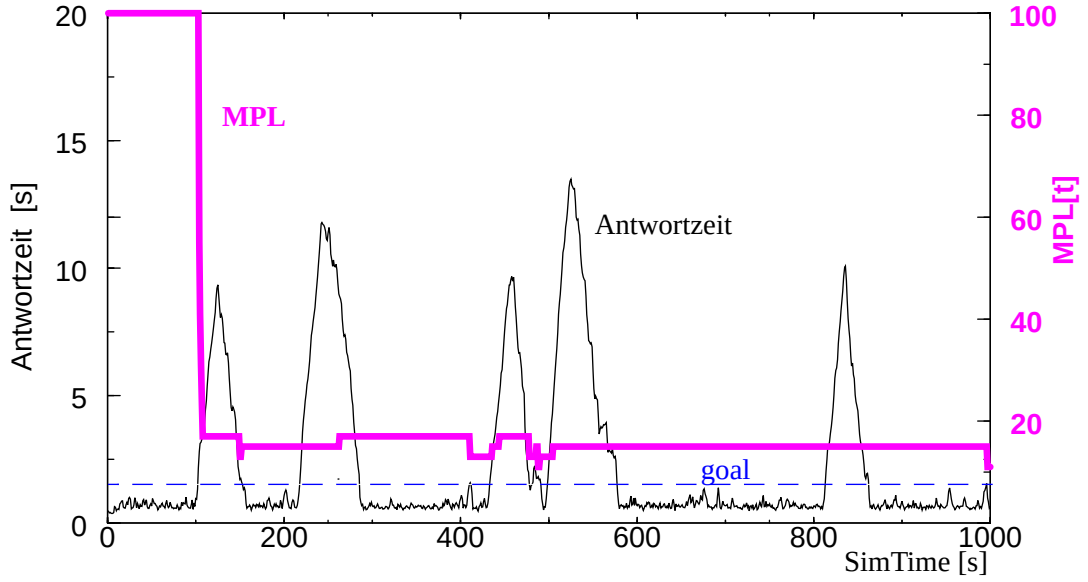

Abb. 3: Variable Last: Antwortzeit- und MPL-Entwicklung (MPL-Startwert 100)

* Die geringfügig schlechtere mittlere Antwortzeit gegenüber dem optimalen statischen MPL-Wert 20 geht auf den CPU-Overhead von knapp 2% für die dynamische Performance-Kontrolle zurück.

periode zugrundegelegt wurde. Bei höherer MPL-Auslastung erfolgen MPL-Änderungen dagegen in kleinen Schritten, um Überreaktionen zu vermeiden. Man erkennt, daß im weiteren Verlauf nur noch wenige MPL-Anpassungen trotz weiterer Lastspitzen vorkommen. Der Grund hierfür liegt darin, daß keine weitere MPL-Reduzierung erfolgt, wenn die Antwortzeiten bereits von MPL-Wartezeiten dominiert sind. Eine MPL-Erhöhung erfolgt dagegen auch nur, wenn die CPU-Auslastung einen kritischen Schwellwert (hier: 98%*) unterschreitet, was jedoch während einer Lastspitze kaum der Fall war. Ein weiterer Grund für die Stabilität und Effektivität der dynamischen Performance-Kontrolle liegt darin, daß nur bei Überschreiten der Antwortzeitziele Korrekturmaßnahmen erfolgen.

In weiteren Experimenten gilt es, die Verfahren zur Behandlung zusätzlicher Engpaßtypen und für heterogene Lastprofile zu evaluieren und zu verfeinern.

Literatur

BMCL94 Brown, K.P.; Mehta, M.; Carey, M.J.; Livny, M.: *Towards Automated Performance Tuning for Complex Workloads*. **Proc. 20th Int. Conf. on Very Large Databases**, 72-84, 1994

BCL96 Brown, K.P.; Carey, M.J.; Livny, M.: *Goal-Oriented Buffer Management Revisited*. **Proc. ACM SIGMOD conf.**, 1996

DG92 DeWitt, D., Gray, J.: *Parallel Database Systems: The Future of High Performance Database Systems*. **Comm. ACM** 35 (6), 85-98, 1992

FNGD93 Ferguson, D., Nikolaou, C., Georgiadis, L., Davies, K.: *Satisfying Response Time Goals in Transaction Processing Systems*. **Proc. 2nd Int. Conf. on Parallel and Distributed Information Systems (PDIS-93)**, 138-147, 1993

MW92 Mönkeberg, A., Weikum, G.*: Performance Evaluation of an Adaptive and Robust Load Control Method for the Avoidance of Data-Contention Thrashing.* **Proc. 18th Int. Conf. on Very Large Data Bases**, 432-443, 1992

NFC92 Nikolaou, C., Ferguson, D., Constantopoulus, P.: *Towards Goal-Oriented Resource Management.* IBM Research Report RC 17919, IBM T.J. Watson Research Center, Yorktown Heights, 1992

Ra89 Rahm, E., Ferguson, D., Georgiadas, L., Nikolaou, C., et al.: *Goal-Oriented Workload Management in Locally Distributed Transaction Systems*. IBM Research Report RC 14712, IBM T.J. Watson Research Center, Yorktown Heights, 1989

Ra96 Rahm, E.*: Goal-Oriented Performance Control for Transaction Processing.* Techn. Bericht (in Vorbereitung), Univ. Leipzig, Institut f. Informatik, 1996

WHMZ94 Weikum, G., Hasse, C., Mönkeberg, A., Zabback, P.: *The Comfort Automatic Tuning Project*. **Information Systems** 19 (5), 381-432, 1994

* Obwohl bei diesem hohen Auslastungswert starke CPU-Wartezeiten in Kauf genommen werden, führt ein geringerer Wert zu MPL-Verzögerungen, die noch stärker ins Gewicht fallen. Dies geht u.a. darauf zurück, daß die mittleren CPU-Belegungszeiten sehr viel kleiner sind als die "MPL-Belegungszeiten" (= Antwortzeiten).