

## Tuningfall 2:

### DB2- und AP-Tuning zur Kostensenkung auf DB2(MIPS)

#### Aufgabenstellung:

Im folgenden Fall wurde ein System, das rein auf dynamischen DB2 Zugriffen basiert, untersucht. Die Zugriffshäufigkeit pro Tag lag bei **ca. 8 Mio bis 10 Mio Zugriffen pro Tag**. Nicht nur, dass damit das zentrale DB2-System stark belastet wurde, es gab eine Reihe von Faktoren, die das System aufwendig und teuer werden ließen.

Die Messungen wurden im normalen DB2 DBA Umfeld durchgeführt.

Die **hohen Accountingkosten pro Monat** bei der Berechnung des Verbrauchs nach MIPS-Stunden galt es in den folgenden Vorgehensschritten signifikant zu senken:

1. Analyse des Applikationssystems
2. Erarbeiten und Planung der Lösungsmaßnahmen
3. Test und Implementieren der Lösungen
4. Vergleich der Ergebnisse und weitere Maßnahmen

#### Vorgehen/Ablauf

Im Schritt 1 kristallisierten sich **bestimmte Aktionen, die vom BEA-Server und dem genutzten System TOPLink automatisch** ausgelöst wurden heraus, die zusammen zwischen **40% und 50% des kritischen Zeitverbrauchs** (CPU und „elapsed“) verursachten. Zudem waren einige ineffiziente Query-Formulierungen und häufige Zugriffe auf statische "Stammdaten" an dem Desaster schuld. Die Lösung sollte eine **Beschleunigung und Minimierung der Zugriffe** durch folgende Maßnahmen sein:

1. **Eliminieren unnötiger Zugriffe** auf das zentrale DB2:
  - a. Die Anzahl der Abfragen zur Bestätigung von existierenden Connects wurde halbiert.
  - b. Die Zugriffe auf die "Stammdaten" wurden auf dem Server "gecached"
  - c. TOPLink wurde entfernt
  - d. Die dynamischen Queries wurden mit Parametermarkern formuliert und
  - e. Das "dynamic cacheing" auf dem zOS-Rechner eingeschaltet.
2. **Vermeiden unnötiger COMMITS** (auch beim Lesen), die von BEA und/oder TOPLink ausgelöst worden waren
3. Senken der DB2-Zeiten durch **Umschreiben der kritischen Queries**
4. Unterstützung der Query-Formulierungen über **zusätzliche Indizes**
5. **Unterstützung kritischer Datenmodifikationen** durch 21 neu erstellter "stored procedures" anstatt dynamischer SQL-Zugriffe aus JAVA

#### Ergebnis

Durch die gebündelten Massnahmen in Pkt. 1 bis Pkt. 4 wurden nach Einsatz der Änderungen im September die **CPU-Kosten von ca. 50 - 60 TEUR auf 12 – 16 TEUR in den Monaten Oktober bis Dezember** gesenkt. Dies wurde erreicht durch die **Senkung der CPU-Kosten um ca. 80% !**

Die Analyse brachte noch **weitere Tuningpotentiale / Defects** zu Tage. Sie sollten sich weiter auf **CPU-Zeiten** und auch auf die „elapsed time“ niederschlagen. Sie brachten auf jeden Fall noch **Vorteile in „wait“- und Antwortzeiten** des gesamten Applikationskomplexes.

Diese Maßnahmen waren notwendig, da **die Anwendung erst 30%** ihrer geplanten Last erhalten hatte. Weitere 70% Lastzuwächse waren in den Jahren 2005 und 2006 noch geplant.

Der Aufwand für diese Maßnahmen betrug insgesamt **ca. 6 Mannmonate ( ca. 70.000 € )** und wurde von der Firma **S.K. Consulting Services GmbH** zusammen mit dem Entwicklungsteam des AG geleistet.