Performance Measurement und Tuning einer eCommerce-Plattform

Andreas Badelt
Technology Architect **andreas_badelt@infosys.com**

Infosys

Infosys ist ein weltweit führender Anbieter von IT und Consulting Dienstleistungen. Wir definieren, entwickeln und implementieren IT-basierte Business Lösungen für unsere Kunden.

Infosys entwickelte als erstes Unternehmen das Global Delivery Model (GDM), das zu radikalen Veränderungen der Branche führte und dem Offshore Outsourcing zum Durchbruch verhalf.

Infosys gibt es seit über 10 Jahren in Deutschland und ist anerkannter Partner führender Unternehmen des Landes.

Kurzinformation:

Hauptgeschäftssitz: Bangalore, Indien

Büros in über 30 Ländern und 76 großen Städten

Development Center in: Australien, Brasilien, USA, China, Indien..

Anzahl Mitarbeiter: 150,000+

Unsere Werte:

Unsere Vision: Wir werden ein weltweit respektiertes Unternehmen sein.

Unsere Mission: Wir wollen unsere Ziele unter Beachtung von Fairness, Ehrlichkeit und Verbindlichkeit gegenüber unseren Kunden, Mitarbeitern, Lieferanten und der Gesellschaft insgesamt erreichen.

Wir haben das Global Delivery Model entwickelt

- **150,000+ Mitarbeiter aus über 80 Nationen**
- **Geschäftstätigkeit in 76 Städten über 33 Länder hinweg**

Einleitung

- Das Messen und Tunen der Performance großer eCommerce-Systeme – vielseitig und anspruchsvoll:
  - Verschiedene Sichten auf das System
  - Anforderungen selten von Beginn an klar
  - Nutzer sind unbekannt, lassen sich nur statistisch erfassen
  - System ist meist in ein komplexes Netz anderer Systeme eingebettet.

- Erfahrungen / „best practices“ aus Entwicklung, Test und Tuning großer, auf ATG Commerce basierender Systeme

- Vorgehensweise anhand des Six Sigma-Zyklus: „Definieren - Messen - Analysieren - Verbessern - Steuern“

Wie kann Performance definiert werden?

- Zusammenspiel verschiedener Stakeholder, insbesondere
  - Fachabteilung: Mengengerüste, Budget->verfügbare Ressourcen, nichtfunktionale Anforderungen
  - User Experience-Spezialisten: Antwortverhalten, Flows
  - Betrieb: Ressourcennutzung, nichtfunktionale Anforderungen
- Applikationsentwicklung muss diese „unter einen Hut“ bringen.
- Zentral: Nutzungsmodell für die zu bewältigende Last

Wie kann Performance definiert werden?

- Nutzungsmodell
  - Nie fertig - wächst mit jedem Release und mit der Zeit
  - Ggf. Vergleichsdaten vorhanden
  - Sollte sobald möglich mit Produktionsdaten angereichert werden.
  - Ziel: Grundlage für aussagekräftige Tests und im Idealfall direkte Vorhersagen aus dem Modell heraus

- Modell ergibt sich aus
  - Annahmen (Fachabteilung, UX – „Brainstorming“)
  - Repräsentativen Tests (geschlossenen Nutzergruppe)
  - Logfile-Analyse / Web Analytics

Wie kann Performance definiert werden?

- Fertiges Modell:
  - Relevante „Flows“ und Sammlung von Statistiken (Requests / Sessions pro Zeiteinheit, Peaks etc.)
  - Daten zu Hintergrundjobs (u.a.: Scheduling)
  - Definierter Sicherheitspuffer („Jahres-Spitzenlast plus X“)

  → Grundlage für Lasttests

Wie kann Performance definiert werden?

- Antwortzeiten
  - Obergrenze für das Laden von Seiten (3-5 Sekunden oder weniger)
  - Wo wird gemessen?
    - Antwortzeit im App / Web Server? Aus UX-Sicht irrelevant
    - Anzeige relevanter Informationen
    - Laden der vollständigen Seite
    - Seite ist wieder „aktiv“ nutzbar
  - Exakte Definition von Messklammern und KPIs (pro Seite). Können als Perzentile festgelegt werden („90% innerhalb 2 Sekunden“).
  - Auch für asynchrone Requests („partial page refresh“).
  - Berücksichtigung von Abhängigkeiten (Backend Calls) – SLAs und Maßnahmen bei Verletzung

Wie kann Performance definiert werden?

- Langzeitstabilität
  - System muss zeitlich unbegrenzt stabil sein
  - Wirklich? Wie lässt sich das testen?

- Ressourcen und Hardware-Sizing
  - Beschränkungen
    - Budget
    - Infrastruktur (CPU, Speicher, Netzwerk, …)
    - Services (z.B. DB, Web Services)
    - Voraussagekraft von Testsystemen
  - -> Frühzeitige Tests von Teilen des Systems und Extrapolation

Wie kann Performance gemessen werden?

- Arten von Tests
  - Lasttests: Fähigkeit, eine bestimmte Last zu verarbeiten
  - Stabilitätstests (Last über längeren Zeitraum)
  - Stresstests: Decken Lücken der Lasttests ab
    - extreme Last oder „ungewöhnliche“ Nutzung)
    - eher unabhängig vom Modell, basierend auf Erfahrungen des Entwicklungs- / Testteams)
  - Kapazitätstests („Abrisskantentests“): Wie viele gleichzeitige Nutzer, Requests etc.

Wie kann Performance gemessen werden?

- Test-Tools
  - Aufzeichnen von Flows
  - Anpassung der Aufzeichnung (Skriptsprache o.ä.) – Flexibilität, Wiederverwendung
  - Messung von Zeiten und Prüfen der Funktionalität
  - Ramp-Up, Ramp-Down-Phasen, Think Time, Timings / Zufallssteuerung, flexible Parallelität …
  - Reporting-Funktion (oder Aufzeichnen und externes Reporting)

Wie kann Performance gemessen werden?

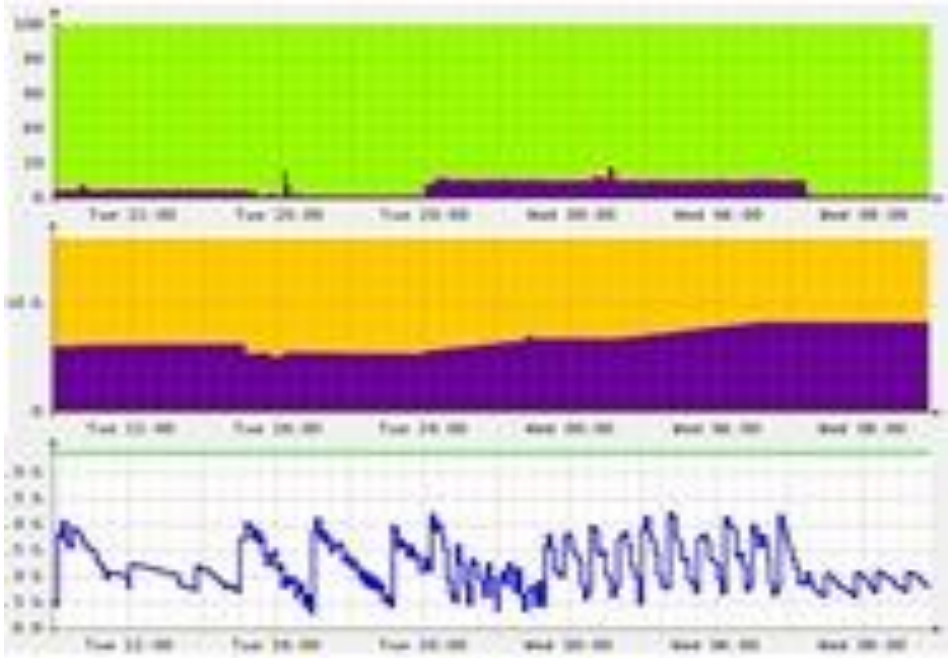
- Was ist zu beachten?
  - Automatisierung der Testsuiten; Limits: Batch-Jobs
    -> „Regieplan“
  - Aufzeichnen und Archivieren der Daten zu allen Tests: Testdefinition, Start-/Endzeit, Regieplan, Instanzen inkl. Startup-Parameter, Testergebnisse und weitere Systemkennzahlen:
    - Host
    - JVM
    - Application Server
    - Applikation
    - Backend-Systeme

Wie kann Performance gemessen werden?

- Was ist zu beachten?
  - Vergleichbarkeit von Umgebungen – insbesondere Skalierung
    - Vollkommene Skalierung meist unmöglich
    - Failover-Tests müssen möglich sein
  - Zusätzlich zum Messen Prüfung der Funktionalität (zumindest rudimentär)

Wie kann Performance analysiert und verbessert werden?

- Aufbereiten von Kennzahlen
  - Kennzahlen aus Datenbank „auf Knopfdruck“ verfügbar machen
  - Graphische Aufbereitung!
  - Zeitliche Korrelationen liefern erste Anhaltspunkte

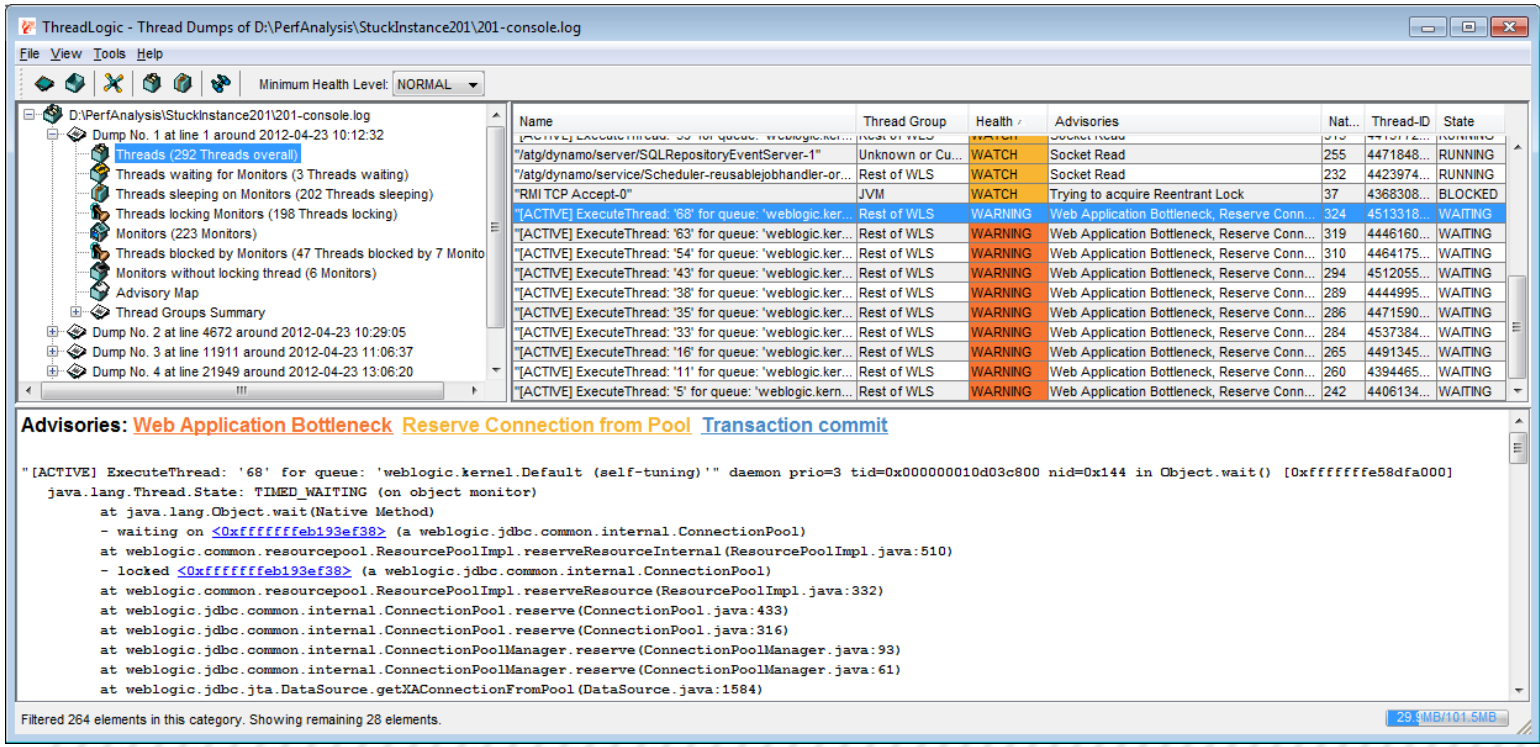


Wie kann Performance analysiert und verbessert werden?

- Datenbankzugriff
  - Connection Pools ausreichend dimensionieren, initiale=maximale Größe
  - Oracle DB? -> Reports aus dem Automated Workload Repository
  - ATG: SQL Repository Logging, alternativ:
  - JDBC-Treiber-Logging, oder:
  - JDBC-Analyse mit Profiling Tools. Aber: Vorsicht vor Schrödingers Katze (gilt nicht nur hier)!

Wie kann Performance analysiert und verbessert werden?

- Threads
  - Application Server Console: Threads und Request Dispatching (WeLo: WorkManager)
  - Thread Dumps am besten regelmäßig ziehen zur Vergleichbarkeit
  - ThreadDumpAnalyzer, ThreadLogic

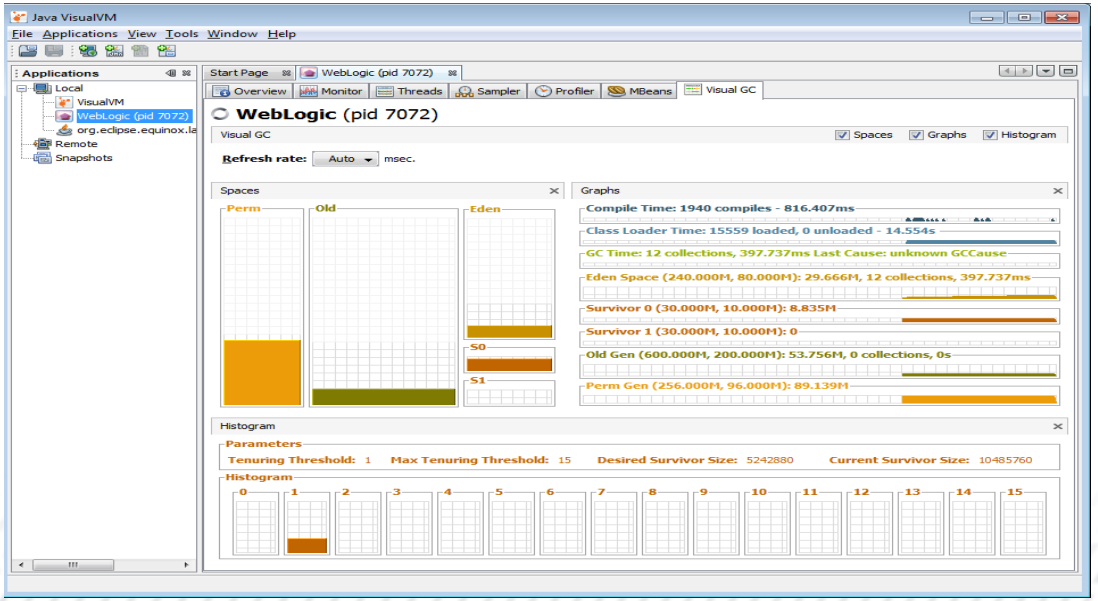


Wie kann Performance analysiert und verbessert werden?

- CPU-Nutzung und Zeit
  - Testtransaktionen (/ „Business Activity Monitoring“)
  - ATG: PerformanceMonitor. Leider: Nicht für hohe Last geeignet!
  - Profiling Tools: JProfiler, JProbe, VisualVM & Co.
  - Profiling unter Last!?
  - Anfangen mit Sampling (Raster am besten >= 10ms)
    -> „Breitensuche“
  - Wenn nötig: Analyse mit Instrumentierung für begrenzte Anzahl „verdächtiger“ Klassen (aus Sampling)
  - Offline Profiling – am besten routinemäßig mitlaufen lassen

Wie kann Performance analysiert und verbessert werden?

- Speicher und Garbage Collection
  - Wieder: Profiling Tools. Oder der Spezialist: MemoryAnalyzer
  - Garbage Collection (Tuning) hat signifikante Auswirkungen
  - Weniger ist oft mehr!
  - Für Webapplikationen:
    - „Old generation“-Wachstum vermeiden (Sessions).
    - Wenn Aufräumen nötig, dann so „leise“ wie möglich: CMS Collector, Parallel Collector
  - VisualGC!
    Auch als Plugin
    für VisualVM.



Wie kann Performance analysiert und verbessert werden?

- Caches
  - ATG: Insbesondere Repository Caches
  - Mindestens einmal pro Release: Alle Cacheeinstellungen prüfen!
  - Bei häufig genutzten Daten: Lieber großzügigen Puffer lassen.

Wie kann Performance analysiert und verbessert werden?

- Load Balancer
  - Von Round Robin über komplexe Lastverteilungsalgorithmen bis hin zum Redirect auf statische „Sorry“-Seite kurz vor Überlast.
  - Aber: Auch Load Balancer machen mal Fehler. Diese sind leider schwer zu finden.
  - Idealtypisch: Oszillierender Verlauf von Überlast einzelner Instanzen.
  - Generell: Wenn die Ideen ausgehen, einfach mal den Load Balancer ersetzen (mod_wl o.ä.)

Wie kann das Erreichen und Halten der Performance gesteuert werden?

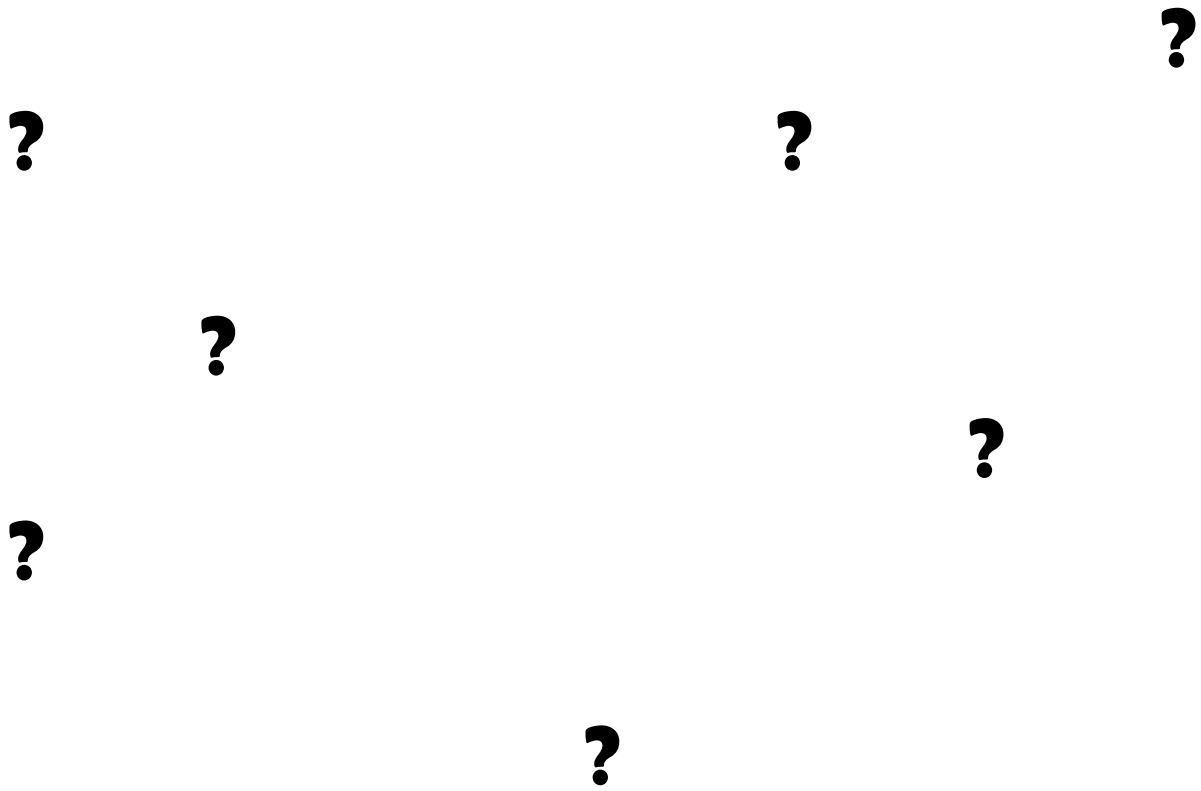
- Frühzeitige Tests während der Entwicklung:
  - Stresstests
  - Offline Profiling
  - Regelmäßiges Prüfen von Testtransaktionen o.ä.
  - Einfache Tools für regelmäßiges Sammeln von Daten im Continuous Build (URLHammer, Performance Unit Tests etc.)

- Mindestens einmal pro Release Last- und Stabilitätstests auf produktionsnaher Plattform

- Bei jeder Abweichung sofort reagieren

Performance Measurement und Tuning einer eCommerce-Plattform

Fragen?



Vielen Dank!

www.infosys.com