Java-Entwicklung im Mainframe- und Batchumfeld

Ein Praxisbericht

Burkhard Seck
Tricept Informationssysteme AG

Ingo Federenko
KfW Bankengruppe

Tricept Informationssysteme

kfw BANKENGRUPPE

Themenüberblick

1. Kurze Vorstellung
2. Was macht Batchjobs so besonders?
3. Typische Schwierigkeiten für Java-Entwickler. Fehler, die häufig wiederkehren.
4. Generelle Strategien zur Vermeidung von Fehlern und zur Lösung bestimmter Probleme.

1 Kurze Vorstellung

Burkhard Seck

Informatikstudium in Paderborn mit anschließender WHK-Tätigkeit

Seit 1999 bei der Tricept AG

Kernkompetenz im Banking- und Finance-Umfeld

Große Architekturen mit Mainframe-Anbindung

Über C, C++ und Smalltalk zu Java

2003 - erster Kontakt mit Java Batch in klassischer Batchumgebung am Mainframe unter z/OS

Betreuung und Erweiterung eines projektspezifischen Java-Batch-Frameworks
(seit 2004 produktiv)

1 Kurze Vorstellung

Ingo Federenko

Studium an der Universität der Bundeswehr 1990 bis 1994

Seit 2005 bei der KfW Bankengruppe

Teamleiter der IT in der Darlehensbuchhaltung (Nebenbuch) und FiBu (Hauptbuch)

Seit 1994 im Großrechnerumfeld, seit 2003 in heterogenen Systemen

Seit 1997 Java als zweite OO-Sprache und siebte Programmiersprache

Seit 2001 als Experte für DB2-Optimierung tätig

Seit 2010 mit Java Batch im Mainframe Umfeld für ein Großprojekt

Themenüberblick

1. Kurze Vorstellung
2. Was macht Batchjobs so besonders?
3. Typische Schwierigkeiten für Java-Entwickler. Fehler, die häufig wiederkehren.
4. Generelle Strategien zur Vermeidung von Fehlern und zur Lösung bestimmter Probleme.

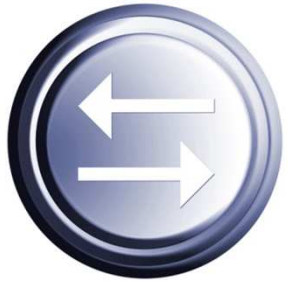
2 Was macht Batchjobs so besonders?

Batchjobs sind doch eigentlich ganz einfach:

- *Datensatz lesen*
- *Daten irgendwie verarbeiten (meist ein einfacher Algorithmus)*
- *Ergebnis wegschreiben*

Und das Ganze in einer Schleife, bis alle Datensätze verarbeitet sind.

2 Was macht Batchjobs so besonders?



In der Regel

- *werden Daten massenhaft verarbeitet,*
- *sind Batchjobs zeit- und performance-kritisch,*
- *läuft ein Batchjob im Verbund mit anderen Batchjobs, mit denen er sich Ressourcen teilen muss,*
- *handelt es sich bei einem Batchjob um einen kritischen Prozess, der durch einen Abbruch das ganze System zum Stehen bringen kann.*

2 Was macht Batchjobs so besonders?

- *Nach einem Abbruch muss ein Batchjob schnell und einfach wiederaufgesetzt werden können.*
- *Batchjobs laufen meist auf Großrechnern.*

- *Die Ausführung eines Batchjobs ist teuer.*

Themenüberblick

1. Kurze Vorstellung
2. Was macht Batchjobs so besonders?
3. **Typische Schwierigkeiten für Java-Entwickler. Fehler, die häufig wiederkehren.**
4. Generelle Strategien zur Vermeidung von Fehlern und zur Lösung bestimmter Probleme.

3 Typische Schwierigkeiten

Java-Entwickler sind für das Designen & Programmieren
von Batchjobs nicht ausgebildet

- *Das Bewusstsein für besondere Anforderungen im Batch fehlt.*
- *Java-Menschen entwickeln objektorientiert.*
- *Entscheidungen in der Designphase wirken sich auf Laufzeitverhalten aus.*
- *Man kann sich schnell im Objekt-/Komponentendesign verlieren.*

Grundregel: Design follows Performance, schnell vor schön

3 Typische Schwierigkeiten

Java-Entwickler sind für Arbeiten am Mainframe nicht mehr ausgebildet

- *Wie bediene ich grundsätzlich eine 3270-Maske?*
- *Wo finde ich was?*
- *Gibt es das auch mit vernünftiger Oberfläche?*
- *Wie starte/stoppe ich einen Batchjob?*
- *Wo sehe ich, wie der Batchjob abläuft?*
- *Wer kann mir das alles erklären?*

3 Typische Schwierigkeiten

Zwei Welten prallen aufeinander

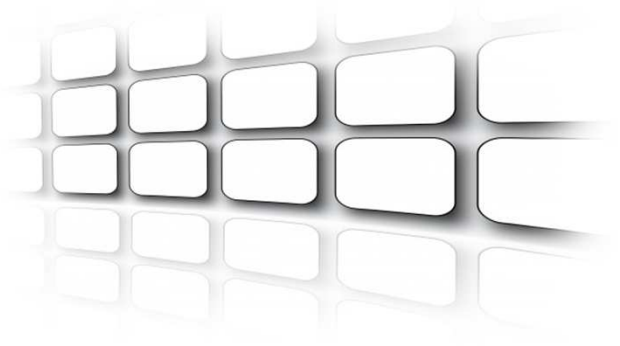
- *Der typische Host-Entwickler*
- *Der typische Java-Entwickler*
- *Altersunterschiede*
- *Unterschiede in der Dynamik beider Kulturen*
- *Völlig unterschiedliche Herangehensweisen an Probleme*
- *Beide Welten haben ihre eigene Sprache.*
- *Reibungsverluste*

➢ *Das kann ziemlich ins Geld gehen…*

3 Beliebte Fehler

Unterschätzen äußerer Einflüsse, wie etwa

- *Einfluss anderer Batchprozesse auf selbst genutzte Ressourcen*
- *Unterschiede im Entwicklungs-, Test- und Produktionssystem*
- *Implementierungsaufwand für Produktionsbetreuung*
  - *Monitoring*
  - *Logging*
  - *Abbruch/Restart-Verhalten*
- *Aufwand für Performance-Tests*



3 Beliebte Fehler

Unterschätzen äußerer Einflüsse, wie etwa

- *Fehlende Tool-Unterstützung, z.B. bei Entwicklung und Fehlersuche*
  - *Profiling & Monitoring*
- *Technische Aspekte*
  - *Startaufwand der Java VM bei kurz laufenden Batchprozessen*
  - *Wirksamkeit des Hotspot-Compilers*
  - *Initialisierung von JPA sehr teuer*

3 Beliebte Fehler

- *Einsatz existierender Eigenimplementierungen ohne Prüfung*
- *Einsatz beliebiger Java-Bibliotheken ohne Prüfung*
  - *Open Source und kommerziell*
- *Verwendung von Default-Parameter-Einstellungen*
  - *Beispiele:*
    - *Startparameter Java-VM*
    - *Garbage Collect*
    - *Caching-Parameter JPA/Hibernate*
    - *…*

3 Beliebte Fehler

Unterschätzen der Datenversorgung eines Batchjobs

- *Datenbanktreiber erzeugt die meisten Objekte*
- *Treiberparameter*
- *Verwaltungsstrukturen in der Persistenzschicht*
- *Commitfrequenz*
- *Sortierte Bearbeitung für C/R-Verfahren*

3 Beliebte Fehler

Technische Optimierungsversuche an fachlich nicht optimalen Batchprozessen

- *Verändern der Parameter der VM*
- *Zuschalten von Prozessoren*
- *Erhöhen der Checkpoint-Frequenz*
- *DB-Maßnahmen (Re-Organisation, Vergrößerung der Bufferpools)*
- *Index-Strukturen Online optimiert ➢ für Batches evtl. kontraproduktiv*

Themenüberblick

1. Kurze Vorstellung
2. Was macht Batchjobs so besonders?
3. Typische Schwierigkeiten für Java-Entwickler. Fehler, die häufig wiederkehren.
4. Generelle Strategien zur Vermeidung von Fehlern und zur Lösung bestimmter Probleme.

4 Strategien & Lösungen

Personal / Ausbildung

- *Andere Philosophie eines Batches*
  - *Besonderheiten der Massendatenverarbeitung*
- *DB-Techniken*
  - *Bufferpool-Besonderheiten bei Massendaten*
  - *Clustering und Indizes bei Massen-Updates und sortierten Inserts*
- *C/R-Frameworks*
  - *Checkpoint-Frequenz vs. Logische Transaktion*
  - *Aufwand/Overhead bei Checkpoint*
- *Kooperation bei Ressourcen*
  - *Monopolisierung vs. Ko-Existenz*
  - *Parallele Änderungen, Dirty Reads, Locking, Deadlocks*

4 Strategien & Lösungen

Technik (DB-nahe Anpassungen)

- *Geringe Auswirkungen auf Einzelzugriff*
- *Multiplizieren sich im Batch*
- *Packages für Zugriff auf DB2 mit KEEPDYNAMIC(Y) binden*
- *Prepared Statement Cache aktivieren (DSNZPARM)*
  - *Wirkung mit Performance-Monitor überprüfen*
  - *Auswirkung: Statement found steigt, Keepdynamic/Implicit Prepare bzw. Prepare Avoidance steigt, Statement not found sinkt*
- *Anpassen der Checkpoint/Commit-Frequenz*
  - *Korreliert mit den Class-3-Zeiten der DB2*
    - *Log Write I/O*
    - *Updates/Commit*

Strategien & Lösungen

Design

Design

- *Blockweises Lesen statt synchroner einzelner Leseoperationen*
  - *Spezieller Batch-Lesemodus*
  - *Asynchrones, multithreaded Read-Ahead*
- *Einsatz von Eager-Joins*
  - *Multiple Ergebniszeilen müssen gefiltert werden*
  - *Nur bei Datengruppen, die bei allen Anwendungsfällen benötigt werden*
- *Einsatz von Lazy-Read*
  - *Enormer Vorteil vor allem beim blockweisen Lesen und Asynchronen Read-Ahead*
- *Parallelisierung bei disjunkten Datengruppen*

Strategien & Lösungen

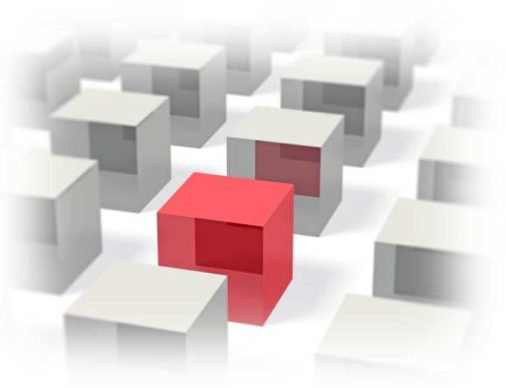
JPA-Besonderheiten

- *Keine ausschließlich generierten Pojos verwenden*
  - *Besser: Pojos generieren, danach nicht benötigte Spalten entfernen*
- *Regelmäßiges „Clear“ der Daten*
  - *„Clear“ kann je Persistence-Unit erfolgen*
- *Nach einem „Clear“ können Lazy-Reads auf eine Exception laufen*
  - *Beim asynchronen Lesen können alle lazy gelesenen Datengruppen initialisiert werden, bevor mit „Clear“ der Buffer gelöscht wird*

4 Strategien & Lösungen

Test-Besonderheiten

- *Lokale Rechner (meist Intel/Windows) haben schnelle Prozessoren, aber langsamen DB-Zugriff*
- *Ziel-Umgebung hat schnellen DB-Zugriff, aber langsame, nicht exklusive Prozessoren*
- *Instanziierung von Objekten auf dem Host teurer als auf Intel-Maschinen*
- *Andere Strategien für Garbage-Collector*
- *Mengen-Besonderheiten beachten*
  - *Insert/Update verhält sich anders bei Non-Clustered Indizes*
  - *Dirty-Prüfung erst bei größeren Datenmengen bemerkbar*



Fragen & Antworten

Vielen Dank für Ihre Aufmerksamkeit!

Kontaktdaten

Burkhard Seck

burkhard.seck@tricept.de

Ingo Federenko

ingo.federenko@kfw.de