Configuring Apache Derby for Performance and Durability

Olav Sandstå
Database Technology Group
Sun Microsystems
Trondheim, Norway
Overview

• Background
  > Transactions, Failure Classes, Derby Architecture

• Configuring Derby
  > Durability of data
  > Performance

• Performance Tips

• Derby Performance
  > Comparing Derby, MySQL and PostgreSQL
Properties of Transactions

Atomicity - “all or nothing”

Consistency - “from one valid state to another valid state”

Isolation - “independent of other running transactions”

Durability - “no committed transaction will be lost”
Failure Classes

• Process:
  > Derby or the JVM crashes

• Operating System:
  > the operating system crashes

• Hardware:
  > CPU, memory or disks fail

• Site:
  > fire, earthquakes, etc

• “Drunken DBA”:
  > DBA accidentally deletes or changes data

“If anything can go wrong, it will”  
Murphy's Law
Derby Architecture: Client-Server
Durability and Performance
Durability and Performance

Durability:

“Want all changes to data to be written to disk”

Performance:

“Want read and write access to data at memory speed”
Data and Log Devices

Log device:
> Sequential write of transaction log
> Synchronous as part of commit
> Group commit

Data device:
> Data in database buffer regularly written to disk as part of checkpoint
> Data read from disk on demand
Performance: Separate Data and Log Devices

Log on separate disk:

- utilize sequential write bandwidth on disk
- Configuration:
  JDBC connection url:
  \[\text{logDevice}=<\text{path}>\]

Performance tip:
Use separate disks for data and log device
Disk Activity

Data and log on one disk:

Data and log on separate disks:

Disk head movement for 5 seconds of database activity
Performance and Durability: Log Device Configuration

**Durability:**
- Log to disk before commit

**Performance:**
- A disk write is “slow” (3-10 ms)

**Options:**
- Disk's write cache:
  - disabled
  - enabled
- Disable durability:
  - derby.system.durability = test
Effect of Disk Log Configurations

**WARNING:** Write cache reduces probability of successful recovery after power failure
Crash Recovery

Process crash
- Durability = test
- Write cache
- No write cache

Power failure
- Durability = test
- Write cache
- No write cache

Durability tip:
Disable the disk's write cache on the log device
Durability: Preparing for Disk Failures

Log device:
- mirror log on two disks (RAID-1)
- must use OS support for mirroring

Data device:
- backup
Backup

Offline backup:
• Stop Derby database
• Copy database files

Online backup:
• Backup while Derby server is running
• New in Derby 10.2: non-blocking online backup
• Supports archiving of log files
Online Backup

- **Backup:**
  - SYSCS_UTIL.SYSCS_BACKUP_DATABASE('/home/backup/061012')

- **Backup and archive log:**
  - SYSCS_UTIL.SYSCS_BACKUP_AND_ENABLE_LOG_ARCHIVE_MODE('/home/backup/061012', 1)
Restore and Roll-Forward Recovery

• Situation:
  > Database is corrupted
  > Disk with database has errors

• Restore and roll-forward recovery using:
  > JDBC connection url:
    - 'jdbc:derby:myDB;rollForwardRecoveryFrom=/home/backup

Help!!
Backup and Restore Strategy

• Define it
  > Derby configuration
    – Mirrored disks for log?
  > Backup configuration
    – Online or offline?
    – Archived log?
  > Restore strategy

• Implement it
  > Ensure it runs regularly

• TEST IT!
  > One day you will need it!!
## Failure Classes: Summary

<table>
<thead>
<tr>
<th>Category</th>
<th>Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process crash</td>
<td>Automatic recovery</td>
</tr>
<tr>
<td>OS crash</td>
<td>Automatic recovery</td>
</tr>
<tr>
<td>Hardware failures</td>
<td>Backup, mirrored log disks, archive log</td>
</tr>
<tr>
<td>Site failures</td>
<td>Backup</td>
</tr>
<tr>
<td>“Drunken DBA”</td>
<td>Backup</td>
</tr>
</tbody>
</table>
Performance Tips
Performance Tips 1: Database Buffer

• Cache of frequently used data pages in memory
• Cache-miss leads to a read from disk (or file system cache)
• Size:
  > default 4 MB
  > derby.storage.pageCacheSize

Performance tip:
• increase the size of the database buffer to get frequently accessed data in memory
Performance Tips 2: Use Prepared Statements

- Compilation of SQL statements is expensive:
  - Derby generates Java byte code and loads generated classes.

- Prepared statements eliminate this cost.

Performance tip:
- **USE** prepared statements
- and **REUSE** them.
Performance Tips 3: Avoid Table Scans

Two ways of locating data:

• Table scan: reads the entire table
• Index: finds the data by reading a few blocks

Avoid table scans:

• Use indexes to optimize frequently used access paths:
  – CREATE INDEX....
• BUT: indexes are not free – needs to be maintained

Performance tip:

• Create and use indexes
Performance Tips 4: Use the Derby Tools

• Know the load on the database:
  > derby.language.logStatementText=true

• Check the query plan:
  > derby.language.logQueryPlan=true

• Use run-time statistics:
  > SYSCS_UTIL.SYSCS_SET_RUNTIMESTATISTICS(1)
  > SYSCS_UTIL.SYSCS_GET_RUNTIMESTATISTICS()

• Optimizer Overrides (New in 10.2)

Performance tip:
• Use Derby's tools to understand the query execution
Performance of Apache Derby 10.2
Apache Derby 10.2

Performance improvements:

- **Client-server:**
  - reduced number of round-trips between client and server
  - reduced CPU usage in Derby network server
  - improved streaming of LOBs

- **SQL Optimizer:**
  - improved optimization
  - support for Optimizer Overrides

30-70% increased throughput on simple queries
Comparing Performance

Databases:
- Derby 10.1.2.1
- Derby 10.2.1.6
- MySQL 5.0
- PostgreSQL 8.0

Load clients:
1. “TPC-B like” load:
   > 3 updates, 1 insert, 1 select
2. Single-record SELECT:
   > one record by primary key

Test platform:
- 2 x 2.4 Ghz AMD Opteron
- Solaris 10
- Sun Java SE 6
Throughput: TPC-B

Main-memory database (10 MB): Disk-based database (10 GB):
Throughput: Single-record Select

Main-memory database (10 MB):  Disk-based database (10GB):

![Graph showing throughput for main-memory database and disk-based database](image)
Performance Improvement Activities

General:
• SQL optimizer improvements

CPU usage:
• Improve use of synchronization to reduce lock contention
• Reduce object allocations/garbage collection

Client-Server:
• Improve LOB streaming

Disk IO:
• Allow concurrent read/write operations on data files
• Reduce number of disk updates during log write
Summary

Performance:
- Separate data and log on different disks
- Configure database buffer to keep most used data
- Use indexes
- Use the Derby tools:
  - query plan
  - optimizer overrides
  - timing statistics

Durability:
- Write log to two disks
  - write cache on disk is dangerous
- Backup regularly
  - include archive log
- Have a strategy for backup and recovery
  - TEST IT!!
Configuring Apache Derby for Performance and Durability

Olav Sandstå
Olav.Sandstaa@sun.com