Connector/J Performance and Debugging Features

Mark Matthews,
Development Manager – Client Connectivity
MySQL AB

MySQL Users Conference, 2005
April 18-21, 2005
Who is Mark Matthews?

Wrote the JDBC driver for MySQL

Team lead for 'Connectors Team'
  Me (Java)
  Georg Richter (php, SAP)
  Hakan Kuecuekyilmaz (php, SAP)
  Eric Herman (Java, SAP)
  Reggie Burnett (.Net)
  Patrick Galbraith (DBD)
  Peter Harvey (ODBC)
  Vasily 'Wax' Kishkin (Windows CE)

Live/work in suburb of Chicago, IL, USA
How About You?

• How many of you....
  – Use O/R Tools (Hibernate, JDO, EJB CMP)?
  – Use Struts, Spring, Xwork or Tapestry?
  – Use Tomcat, Jetty, Jonas Jboss, Resin, Weblogic, WebSphere?
  – Use an IDE?
Why Performance?

- Related to scalability
- Easier problem to solve (sometimes)
- Applies to all applications
- End-user/customer metric
- Yes, we’ll cover some scalability features/issues too!
J2EE Queuing Model

Client Request Queue → Execution Queue → JDBC/JMS/JCA Connection Pool → Transactional Resource

Cost of Resource
\( N = X \cdot R \)

Little’s Law

\( X_i = V_i \cdot X_0 \)

Forced Flow Law

\( U_i = X_i \cdot S_i \)

Utilization Law
What’s Important Then?

• Arrival Rate
• Service Times, Service Times, Service Times
• Concurrency == Bottlenecking
Fast Response Time vs Max Throughput
Slower Response Time vs Max Throughput
Application Segmentation

• Complexity/Size (data volume)
  – Determines applicability of performance features
  – Drives necessity for architectural changes

• Complex Architectures
  – You ain’t gonna need it (YAGNI)
  – Hopefully you thought ahead, though…
What Kind of Application?

MySQL®

100 users/100 TPS (32-bit)
400 users/200 TPS (64-bit)

100-500GB (32-bit platforms)
20-50GB (64-bit platforms)

small/low

50-100 (> 16-way joins)

Number of Tables/Entities

Concurrency/TxRate

Database Volume
Low Volume/Complexity

- Adding/Altering Indices
- Tuning MySQL Parameters
- Fixing some queries
- Increase heap avail to JVM
Moderate Volume/Complexity

- Everything from low complexity
- Application-level caching
- Increase server ‘headroom’
High Volume/Complexity

- Everything from medium complexity
- Architecture (replication, horizontal scaling)
Replication to Scale

- Writes and Time-sensitive Reads
- Read-only transaction

MySQL

Master

Slaves

Read-only transaction
public static void main(String[] args) throws Exception {
    ReplicationDriver driver = new ReplicationDriver();
    Properties props = new Properties();
    props.put("autoReconnect", "true");
    props.put("roundRobinLoadBalance", "true");
    props.put("user", "foo");
    props.put("password", "bar");

    // Looks like a normal MySQL JDBC url, with a comma-separated list
    // of hosts, the first being the 'master', the rest being any number
    // of slaves that the driver will load balance against
    Connection conn = driver.connect("jdbc:mysql://master,slave1,slave2,slave3/test", props);

    // Do something on the master
    conn.setReadOnly(false);
    conn.setAutoCommit(false);
    conn.createStatement().executeUpdate("UPDATE some_table ....");
    conn.commit();

    // Now, do a query from a slave, the driver automatically picks one
    // from the list
    conn.setReadOnly(true);

    ResultSet rs = conn.createStatement().executeQuery("SELECT a,b,c FROM some_other_table");
}
Extremely High Volume/Complexity

- Everything from previous slide(s)
- Hardware changes
  - (64 bit!)
  - Storage hardware
Continuous Performance Testing

- Test early, test often
- Low-load tests during development
  - Often, problems can show up under $n > 10$ users
- Connector/J Performance Monitors
  - Usage Advisor
  - Profiler
- Production Class/QA Load Tests at “Strawman” Stage
While Tuning – Be methodical!

• Script everything
• Reproducibility of test case is paramount
• Sit on your hands (only change one thing at a time)
• Log every configuration change, and their results (version control is ideal)
Sizing Your Connection Pool

Connection Pool Size vs. Actual Utilization
300 Virtual Users on a Typical Java Web Application

- Max Idle
- Max Active

Number of Connections

Connection Pool Size

MySQL
Use the Latest Stable Driver

LoadStorePerfTest Performance
TX consists of 3 queries -- 2 SELECTs, 1 UPDATE
of an entity with 16 fields of assorted types.

MySQLo
Work Efficiently!

• Release JDBC resources as soon as possible
• Use as few queries as possible to get the job done
• Only retrieve data you’ll actually use
• Cache what you can, if you can
Cache What Cache Can

- **PreparedStatement/CallableStatement**
  - cachePreparedStatements=true
  - cacheCallableStatements=true
- **Server Configuration**
  - cacheServerConfig=true
- **Session State**
  - elideSetAutoCommits=true
  - useLocalSessionState=true
  - alwaysSendSetIsolation=false
Finding Database Bottlenecks
MySQL Server Bottleneck Hunting

• Start with Connector/J Perf Metrics
• Follow up with MySQL server tools and techniques
• Everything (almost) you learn in other sessions this week is applicable with J2EE apps
• Connector/J perf metrics show you where to fix your code
MySQL Server Tools/Techniques

- Identify slow queries (with help from JDBC driver)
- Use explain with slow queries to determine why they are slow
- Fix why they are slow
  - Add indexes (key/possible_keys: NULL, compound indexes for sorting/grouping)
  - Rewrite
  - Update statistics or force index – (type:ALL)
- Use MySQL status variables ‘show ....’
  - SHOW INNODB STATUS
  - SHOW STATUS (key_reads, key_read_requests)
Connector/J Instrumentation

• Connector/J 3.1.1 and newer have most instrumentation
• More application-level information available than database server alone
• Functionality
  – Logs slow queries
  – Reports aggregate performance metrics
  – Advises on best practices (usage advisor)
Logging Slow Queries

- logSlowQueries=true
- 'auto explain' -> explainSlowQueries=true
- slowQueryThresholdMillis=n (2000 default)

- Wed Apr 14 05:38:02 PDT 2004 WARN: at testsuite.simple.ConnectionTest.testSavepoint(
  ConnectionTest.java:376) Slow query (exceeded 50 ms): DROP TABLE IF EXISTS testSavepoints
Slow Queries Demo
Reporting Performance Metrics

- gatherPerfMetrics=true
- reportMetricsIntervalMillis=n (30s default)
Performance Metrics Demo
Using the Usage Advisor

• useUsageAdvisor=true
• ‘Abandoned’ objects
• Un-used columns in SELECTs
• Incomplete result set traversal
• Costly type conversions – Prepared Statements
Usage Advisor Demo
Part I – “Take Aways”

• Learn how to use EXPLAIN
• Indexing is important
• Milliseconds count!
• Connector/J pinpoints problem areas
• Deal with performance issues early
• No premature optimizations!
Lunch!
Part II – Debugging Features
Help Us Help You

• Bug reports
  – More detail the better
  – Less work for you, the better

• Technical Support
  – Ditto
Filing Good Issue Reports

• If you are in doubt about stating something, state it

• Include version #'s of everything involved
  – OS, JVM, MySQL Server, C/J, Appserver

• Standalone, repeatable testcases if possible (we like Junit testcases!)
Available Debugging Features

• profileSQL=true
• traceProtocol=true
• enablePacketDebug=true (packetDebugBufferSize)
• dumpQueriesOnException=true
• Use trace-enabled driver (*.bin-g.jar)
  – Connector/J 3.1.8 or newer
  – Requires Aspect/J Runtime
  – Very, Very Verbose
Questions?