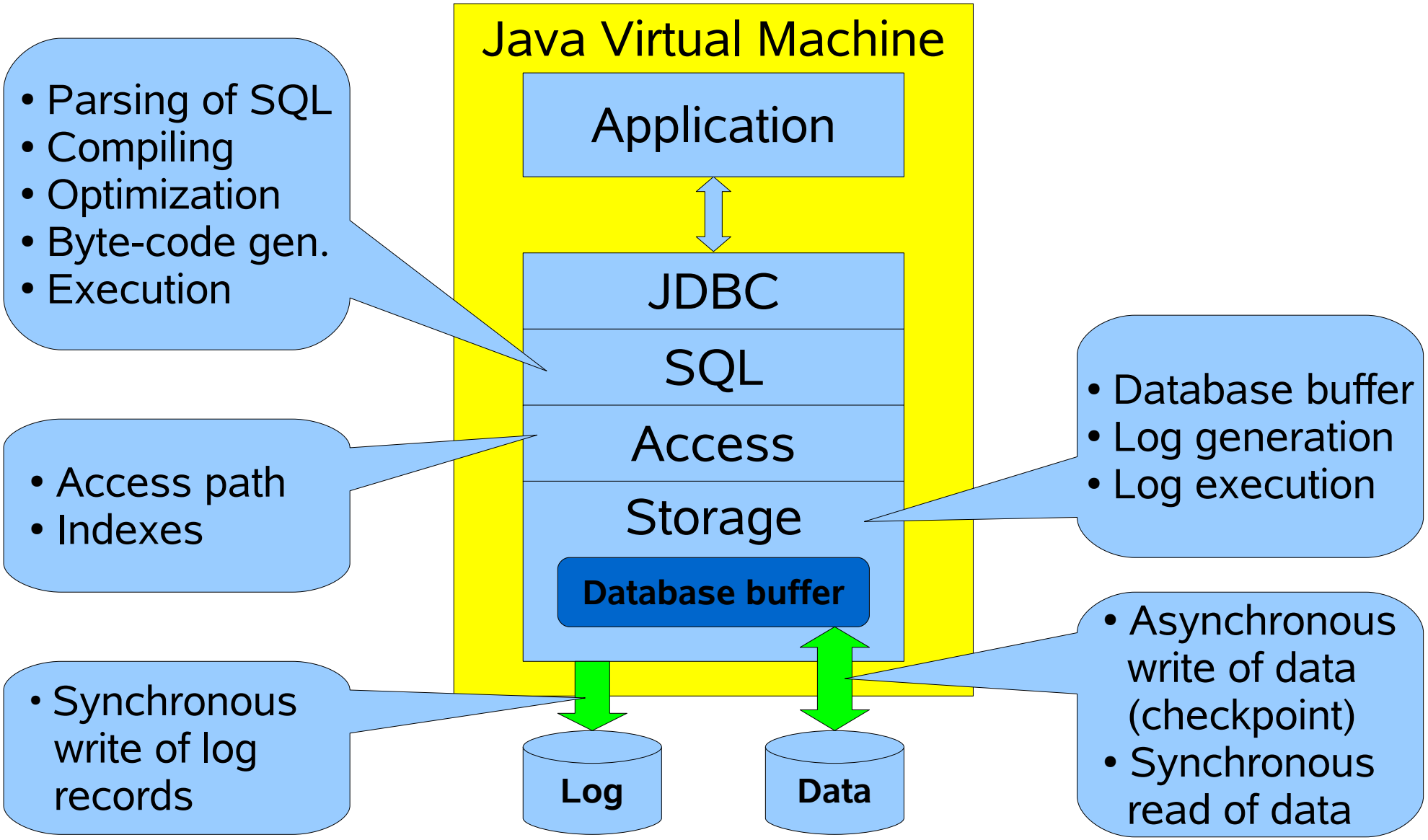
Apache Derby Performance

Olav Sandstå, Dyre Tjeldvoll, Knut Anders Hatlen
Database Technology Group
Sun Microsystems

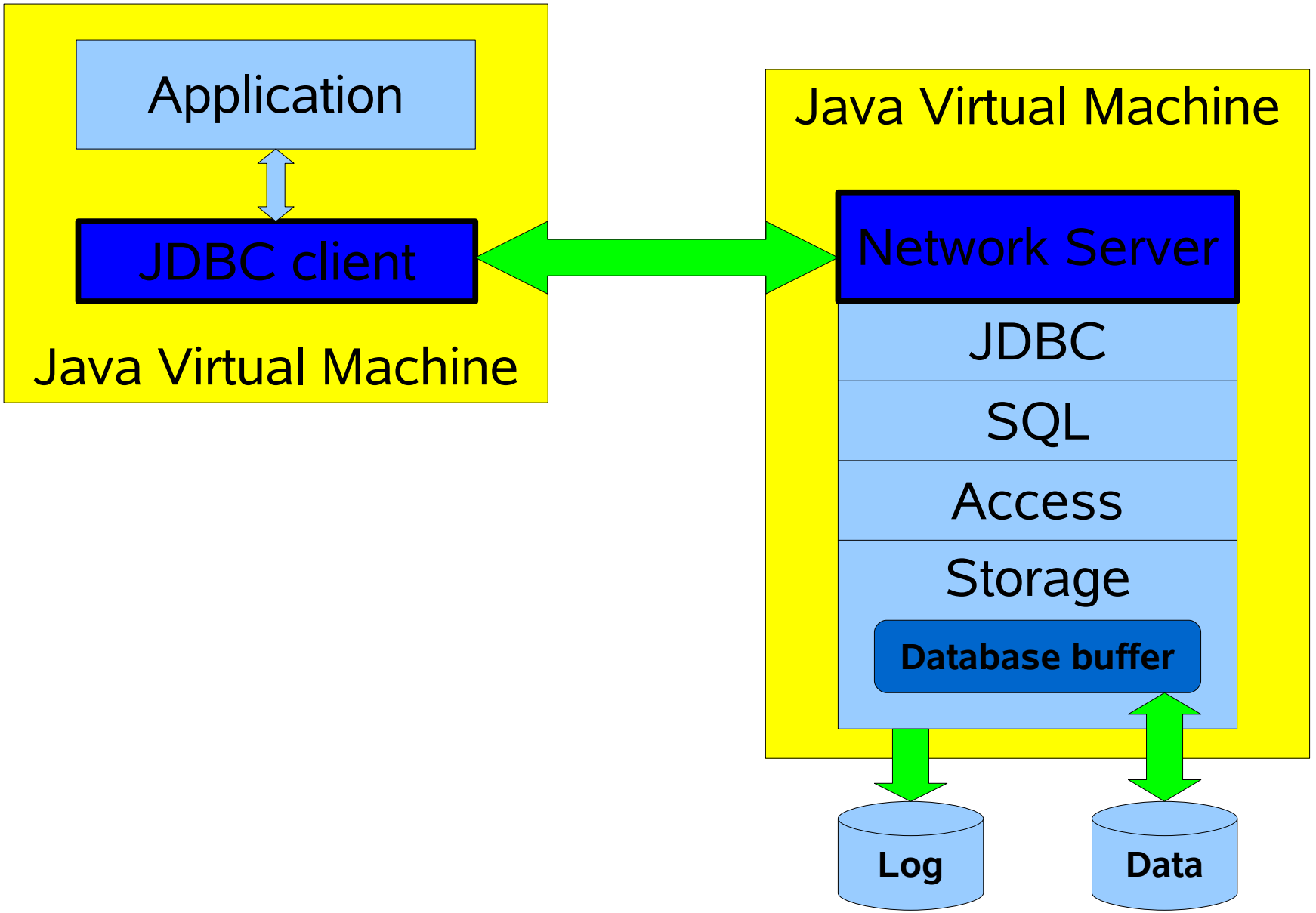
Overview

- Derby Architecture
- Performance Evaluation of Derby
- Performance Tips
- Comparing Derby, MySQL and PostgreSQL

Derby Architecture: Embedded



Derby Architecture: Client-Server



Performance Evaluation of Derby

Overview:

- Evaluation of disk and file system configurations:
  - Disk write cache
  - Database and transaction log on separate disks
- Database configurations:
  - Size of database buffer
- Comparing Embedded and Client-Server

What Is “Performance”?

- How to measure database performance?
  - Throughput
  - Response time
  - Scalability
- Which configuration?
  - Out-of-the-box configuration
  - Carefully tuned
- How to compare database systems with different properties and different tuning possibilities?

Test Configuration

Load clients:

1. TPC-B like load:

- 3 UPDATES
- 1 SELECT
- 1 INSERT

2. Single-record SELECT:

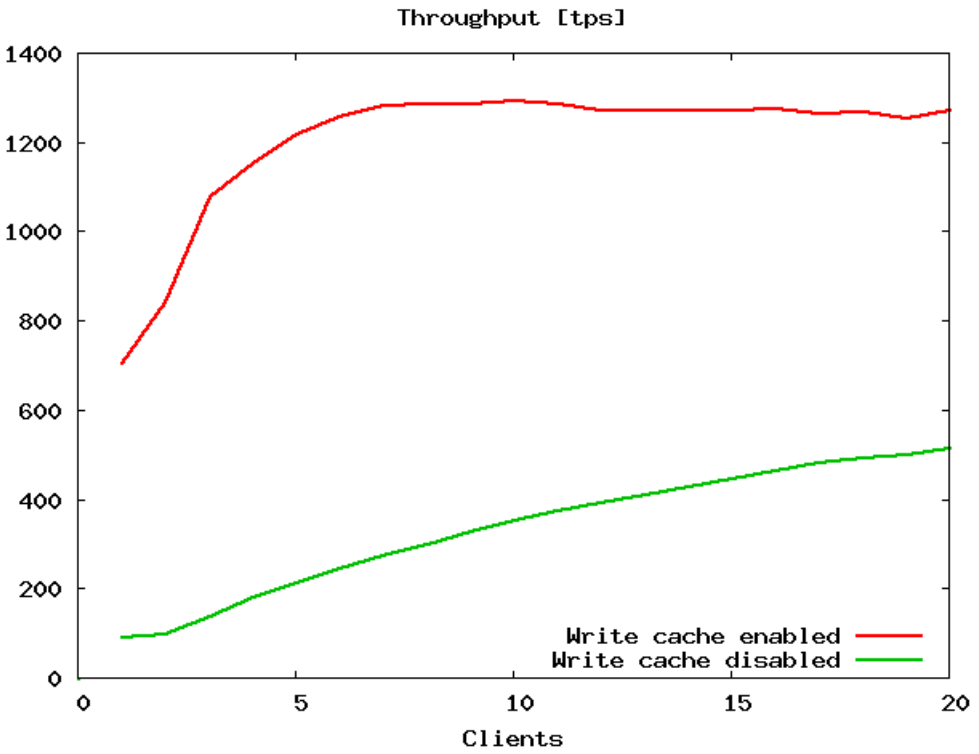
- Select of one record on primary key

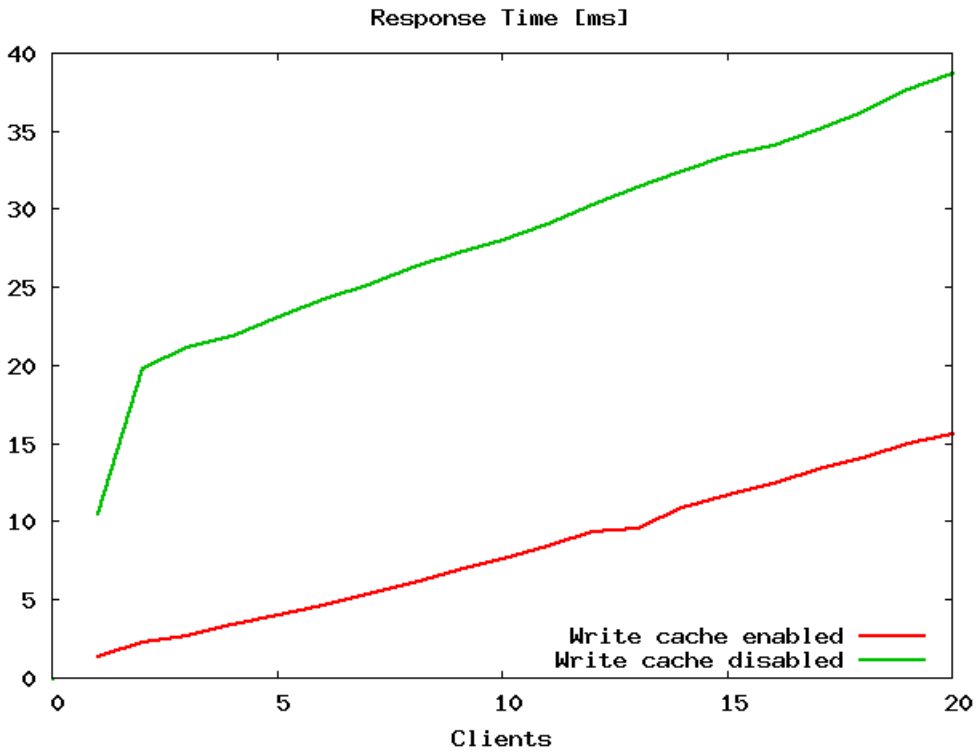
Test platform:

- Sun JVM 1.5.0
- Solaris 10 x86
- 2 x 2.4 GHz AMD Opteron
- 2 GB RAM
- 2 SCSI disks
- UFS file system

Effect of Write Cache on Disk

TPC-B like load:





WARNING: The write cache reduces probability of successful recovery after power failure

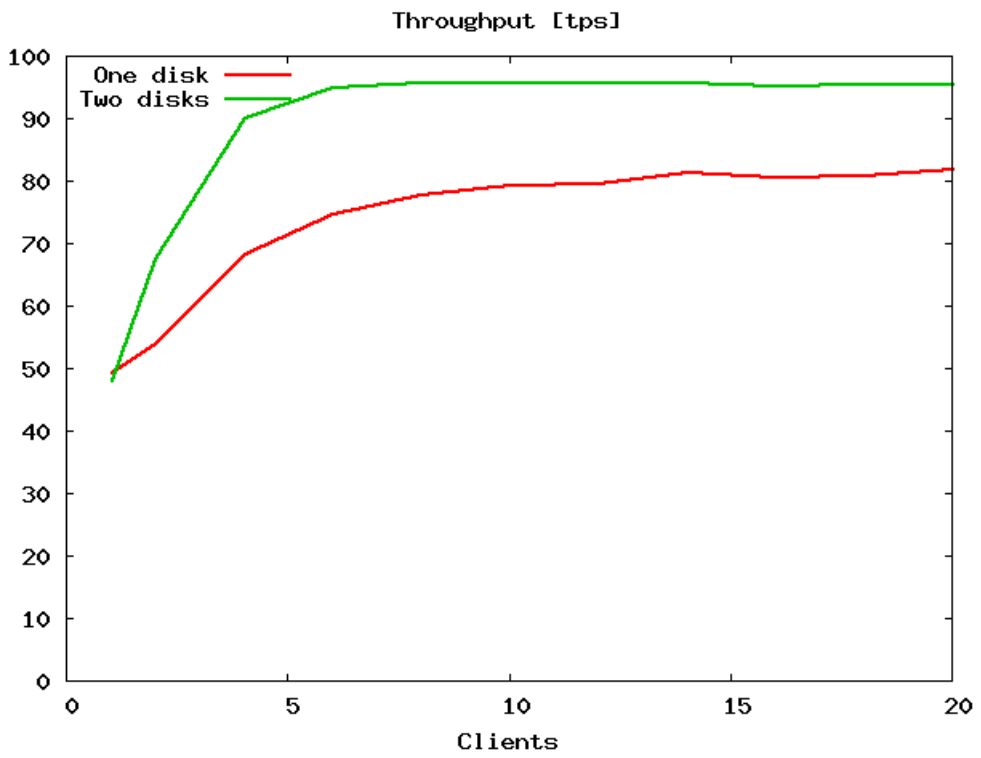
Separate Data and Log Devices

Log device:

- Sequential write of transaction log
- Synchronous as part of commit
- Group commit

Data device:

- Data in database buffer regularly written do disk as part of checkpoint
- Data read from disk on demand

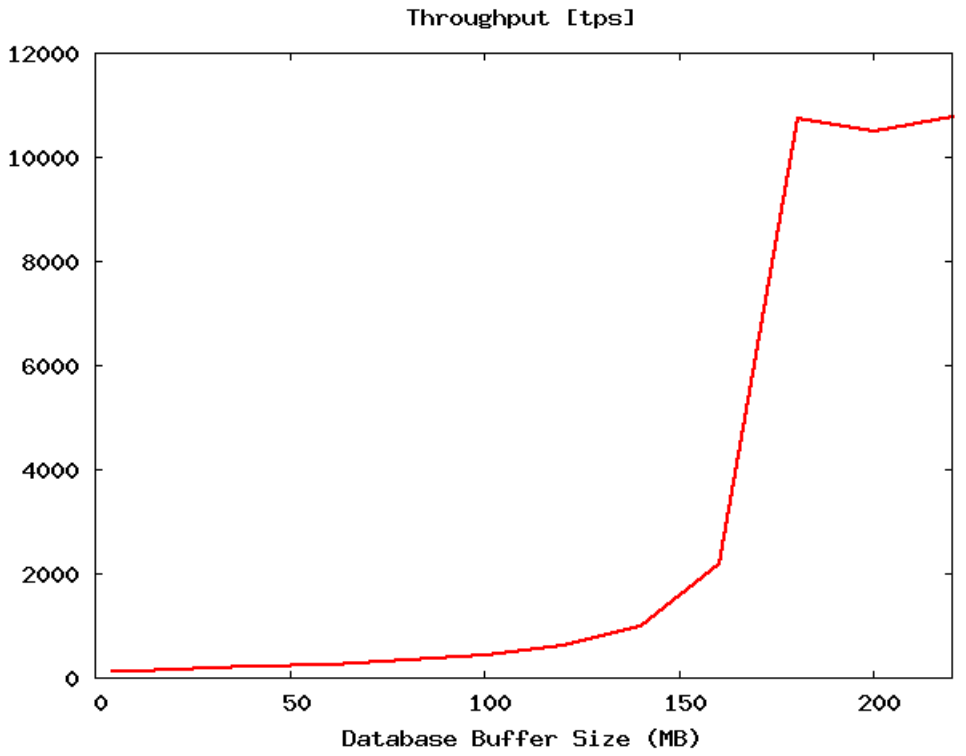


Tip: use separate disks for data and log device

Database Buffer

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

Example:
single-record select:

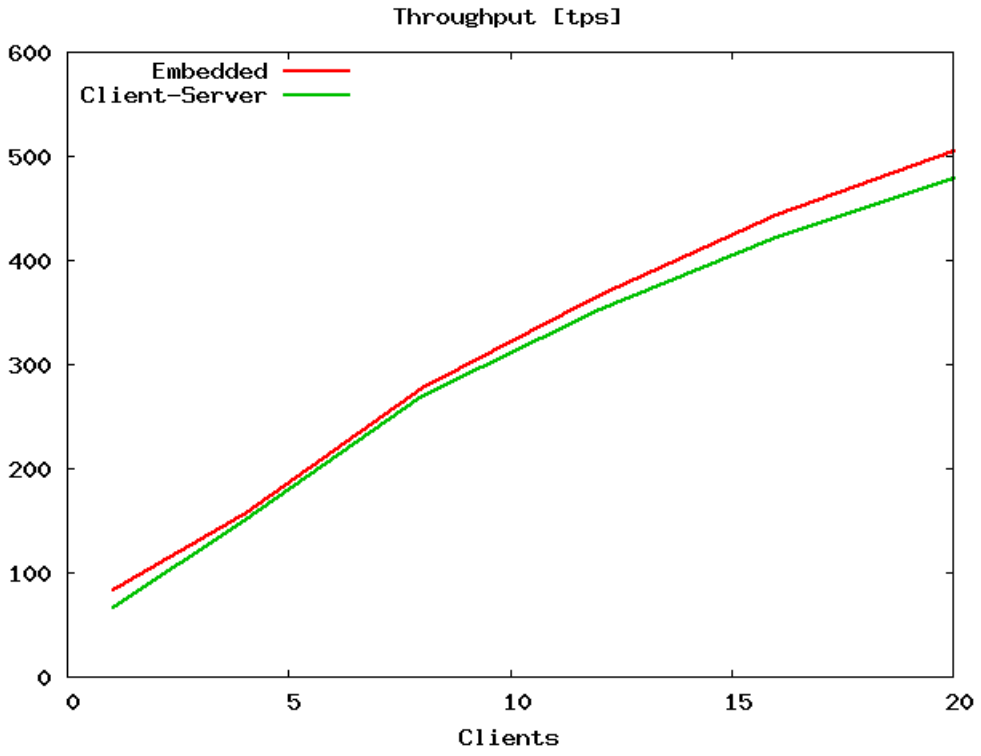


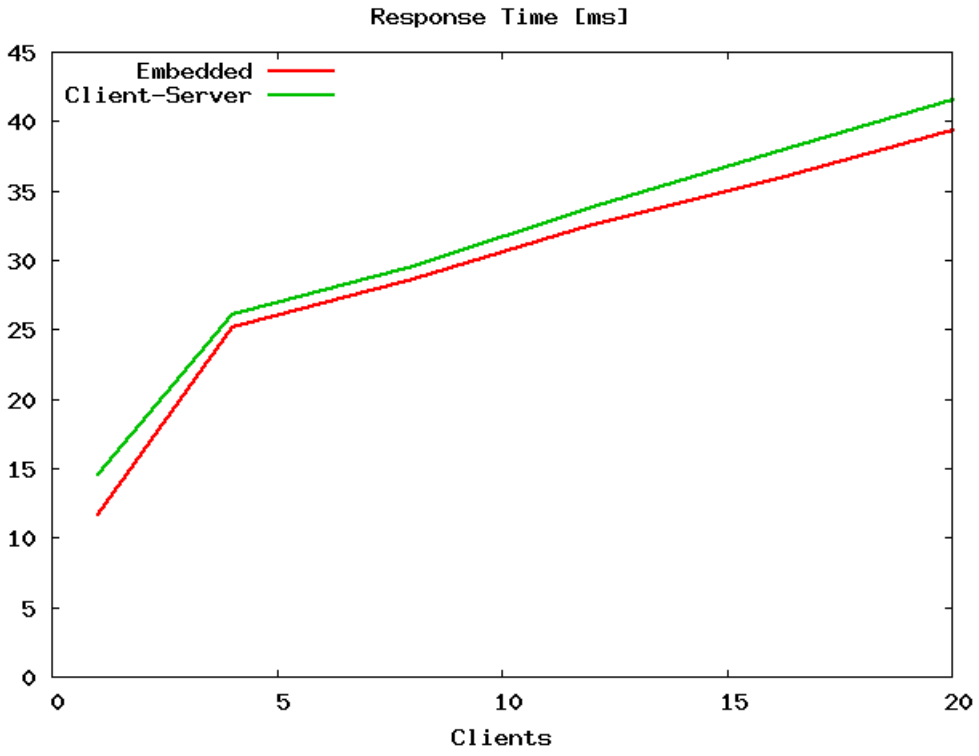
Performance tip:

- increase the size of the database buffer to get frequently accessed data in memory

Comparing Embedded and Client-Server

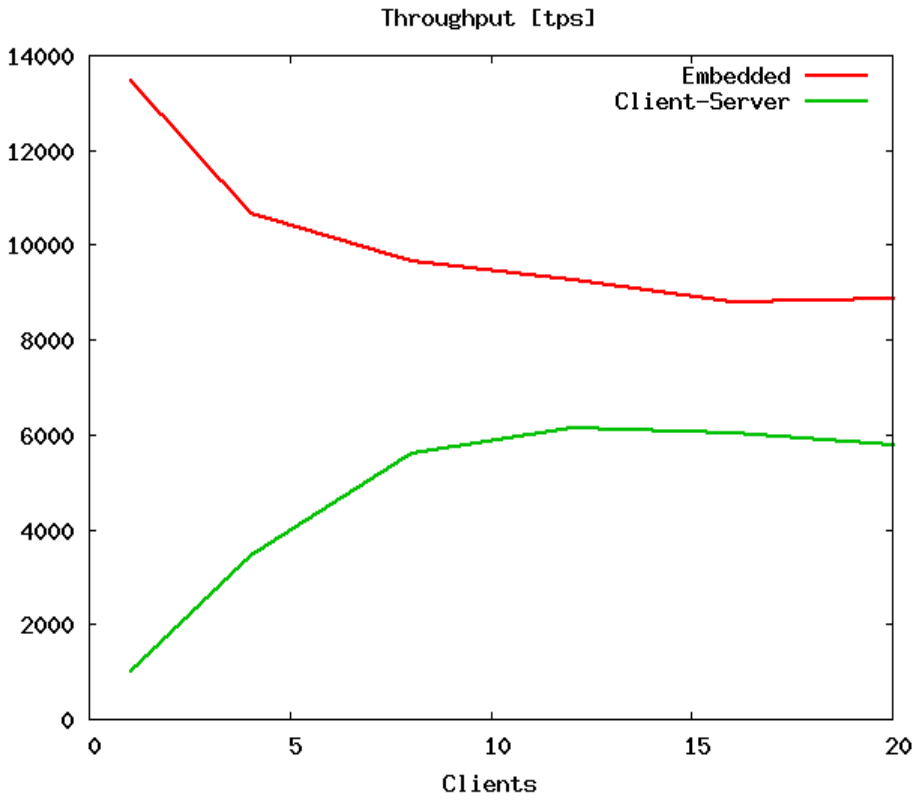
- TPC-B like load:

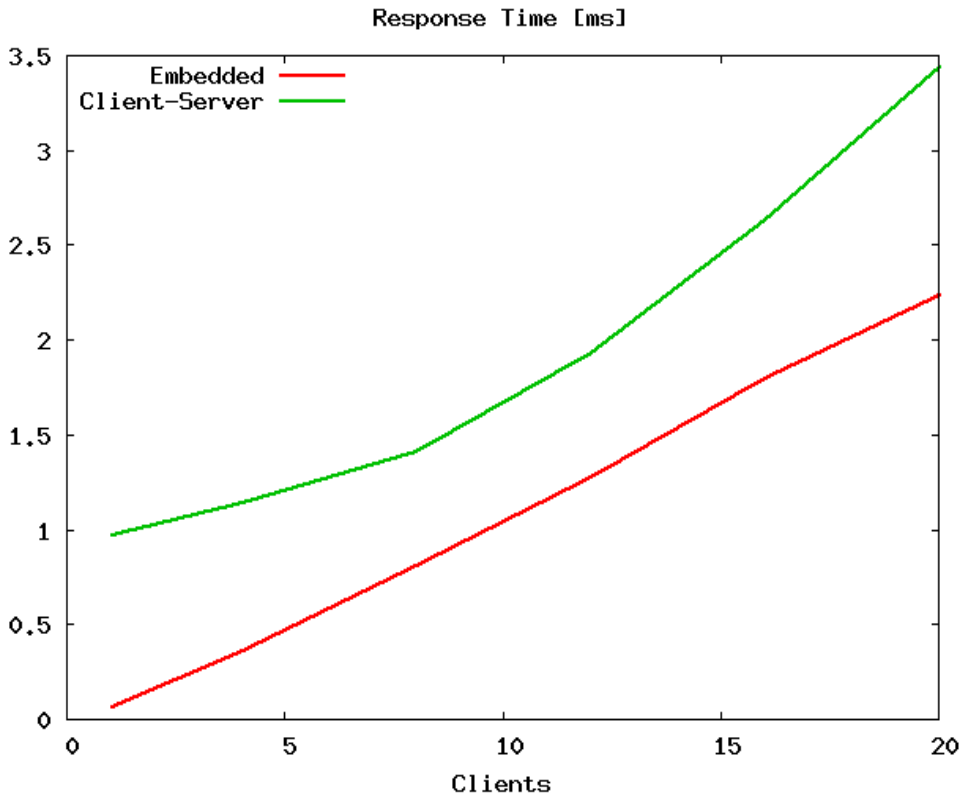




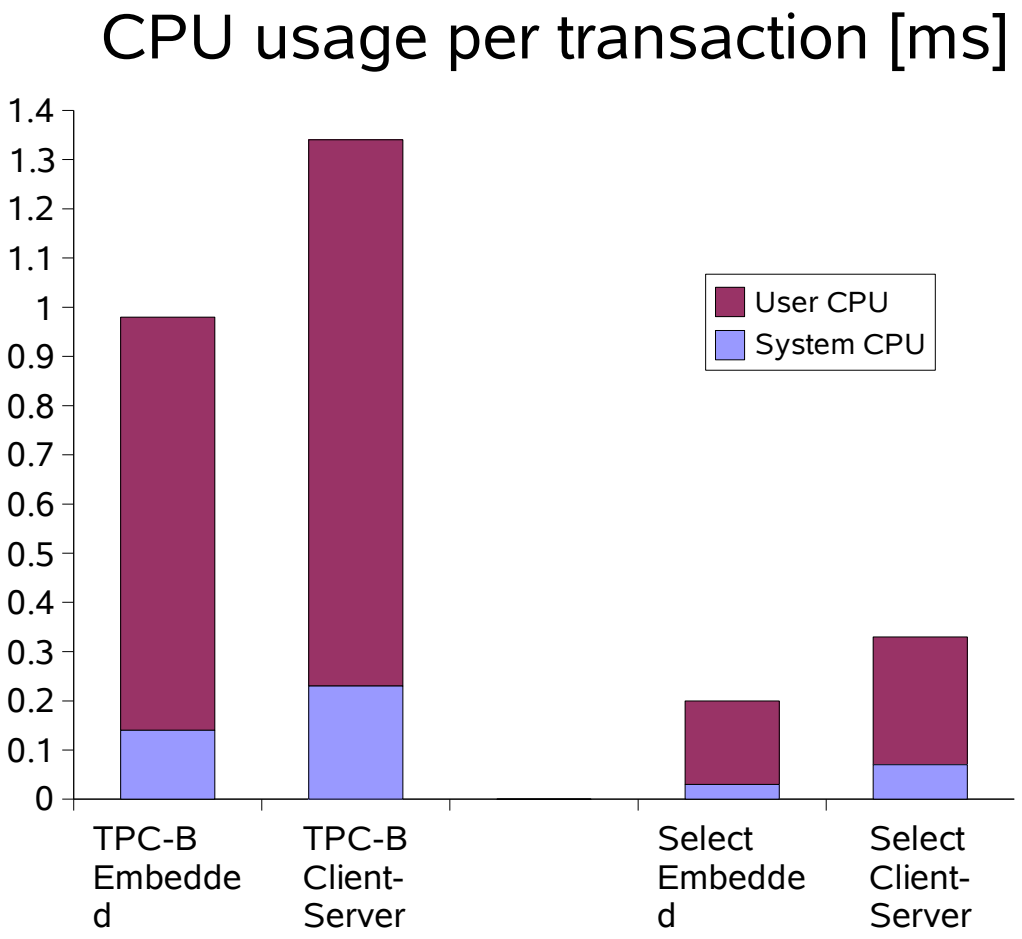
Comparing Embedded and Client-Server

- Single-record SELECT:





Comparing Embedded and Client-Server:
CPU Usage

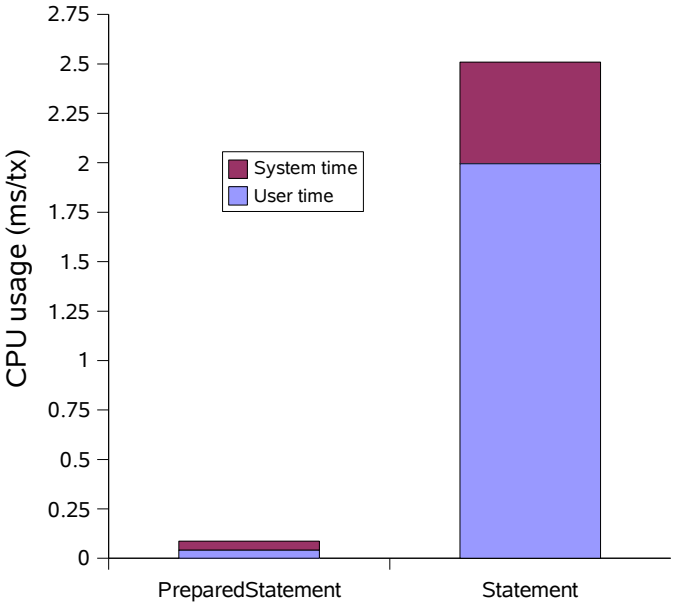


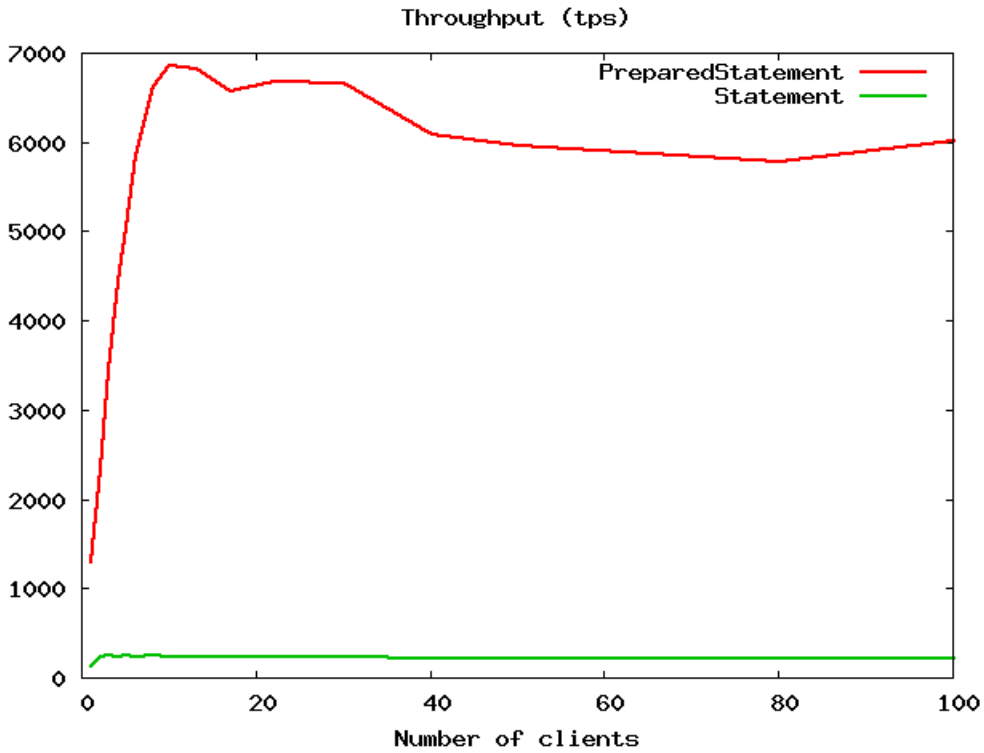
- Network Server add 30%-50% to the CPU usage
- System CPU usage:
  - Message sending and receiving
- User CPU usage:
  - Message parsing
  - Character set conversions

Performance Tips

Performance Tips 1:
Use Prepared Statements

- Compilation of SQL statements is expensive:
  - Derby generates Java byte code and loads generated classes
- Prepared statements eliminate this cost





Tip:

- **USE** prepared statements
- and **REUSE** them

Performance Tips 2:
Avoid Table Scans

- Use indexes to optimize much used access paths:
  - `CREATE INDEX....`
- Check the query plan:
  - derby.language.logQueryPlan=true
- Use RunTimeStatistics:
  - SYSCS_UTIL.SYSCS_SET_RUNTIMESTATISTICS(1)
  - SYSCS_UTIL.SYSCS_GET_RUNTIMESTATISTICS()

Tip:

- Use indexes
- Use Derby's tools to understand the query execution

Comparing the Performance of MySQL, PostgreSQL and Derby

Performance Evaluation:
MySQL, PostgreSQL and Derby

Evaluated performance of:

- MySQL/InnoDB (version 5.0.10)
- PostgreSQL (version 8.0.3)
- Derby Embedded (version 10.1.1.0)
- Derby Client-Server

Database Configurations

Configurations:

- "Out-of-box" performance
- No tuning, except:
  - > size of database buffer
  - > database and transaction log on separate disks
- → No Benchmark

Load:

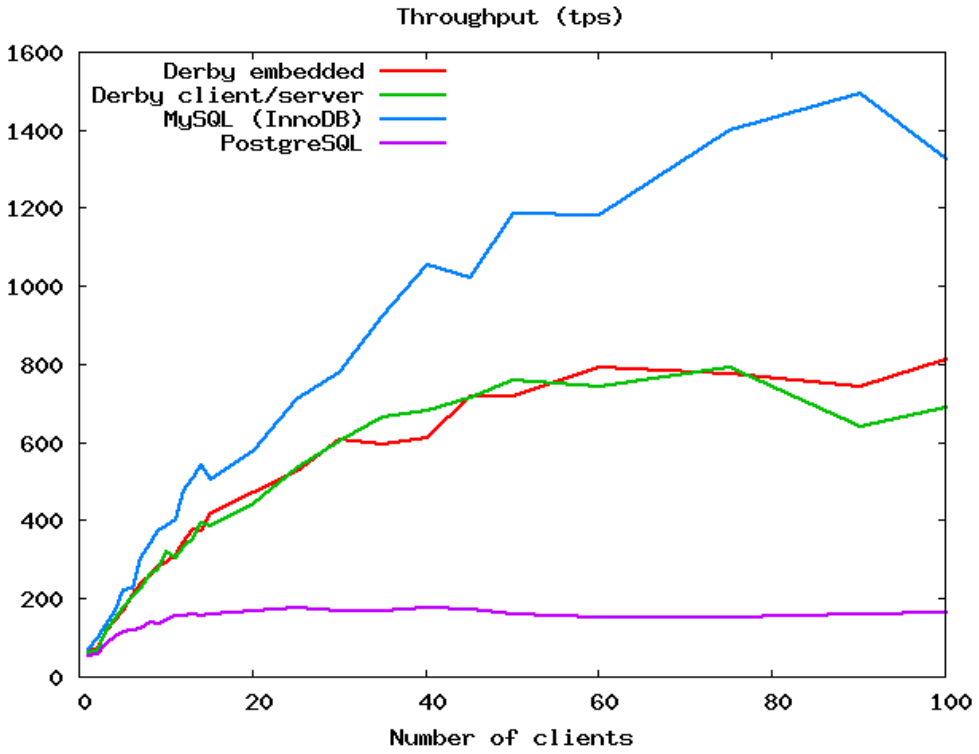
- > 1-100 concurrent clients

Databases:

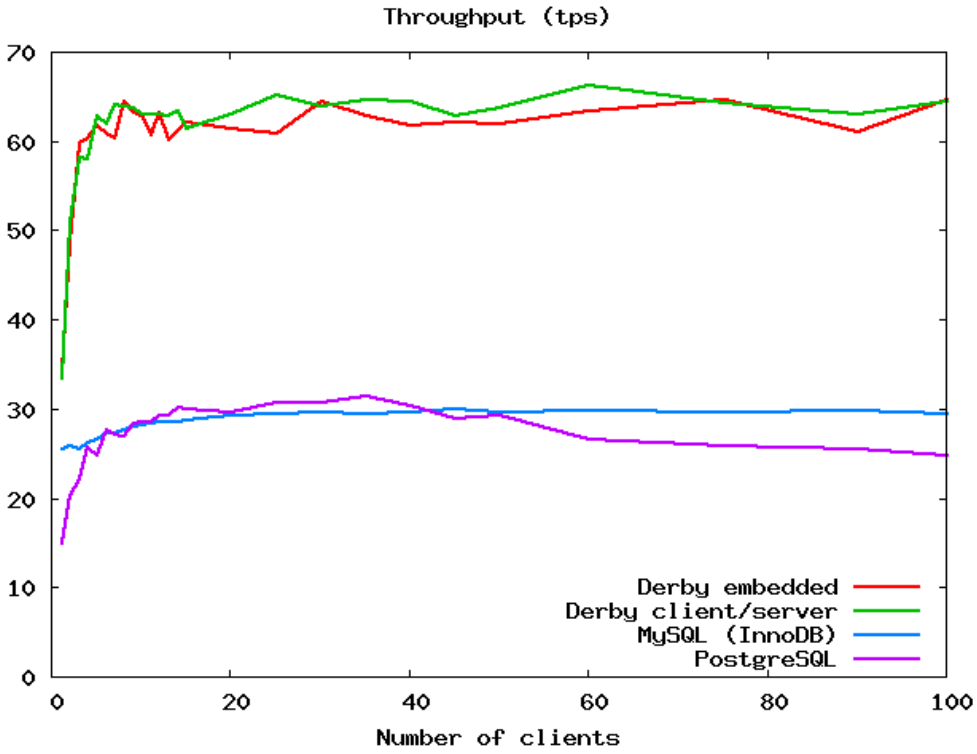
1. Main-memory database:
   - > 10 MB user data
   - > 64 MB database buffer
2. Disk database:
   - > 10 GB user data
   - > 64 MB database buffer

Throughput: TPC-B like load

Main-memory database (10 MB):

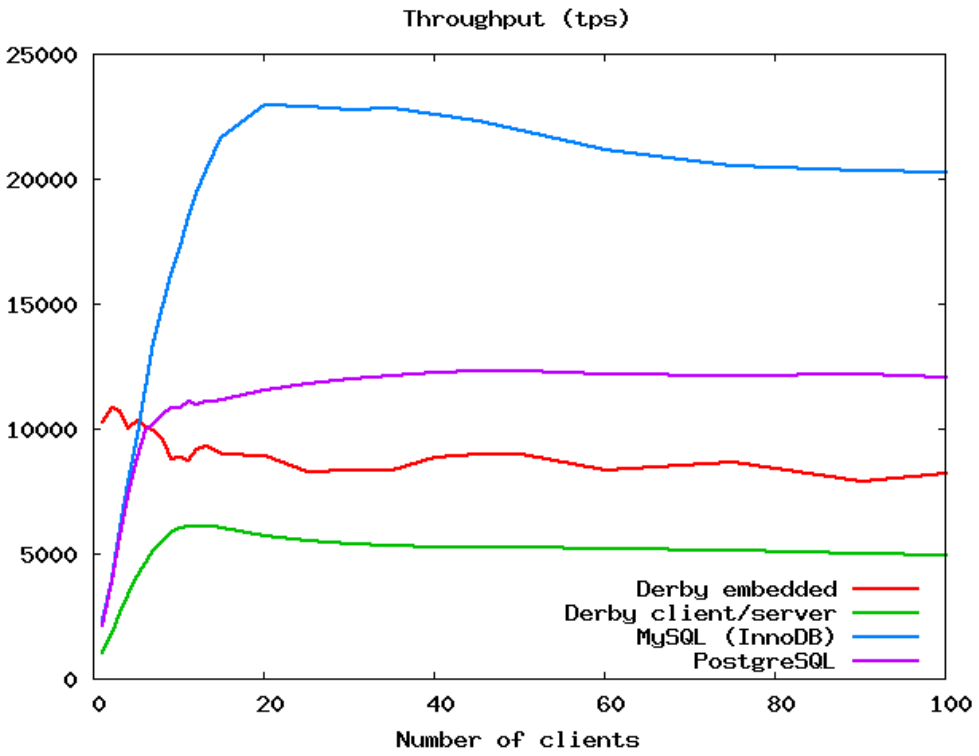


Disk-based database (10 GB):

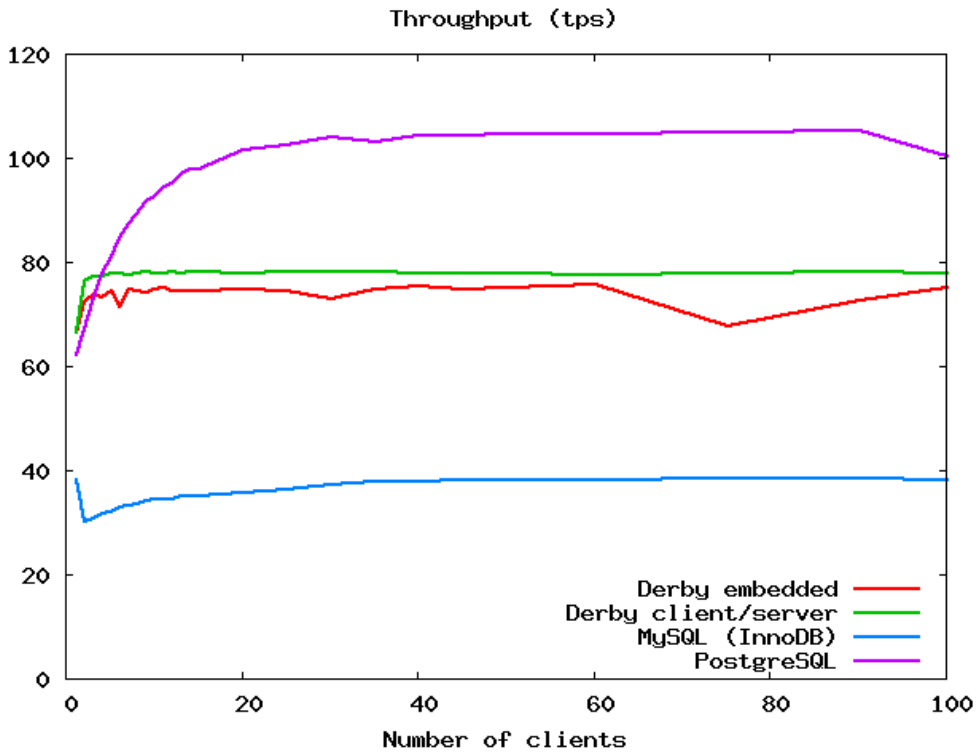


Throughput: Single-record Select

Main-memory database (10 MB):



Disk-based database (10GB):

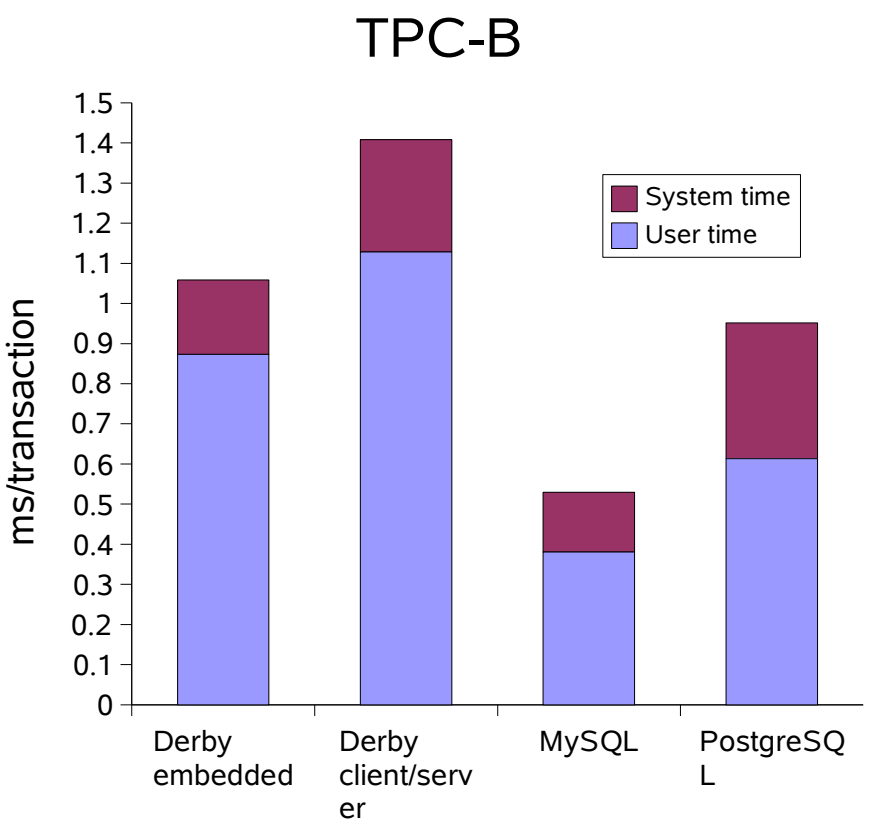


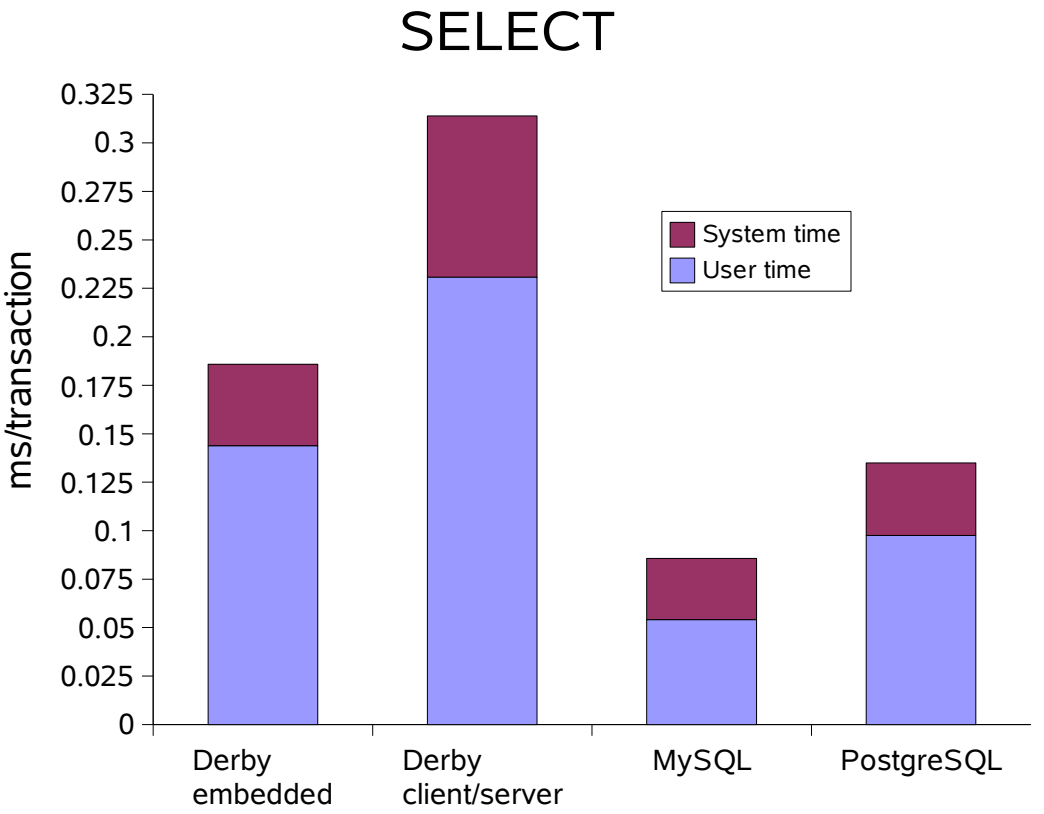
Observations

- Derby outperforms MySQL on disk-based databases
  - > Derby has 100% higher throughput than MySQL
- MySQL performs better on small main-memory databases
  - > Update-intensive load: Derby has 20-50% lower throughput
  - > Read-intensive load: Derby has 50% lower throughput
- PostgreSQL performs best on read-only databases, and has lowest throughput on update-intensive databases

Why?

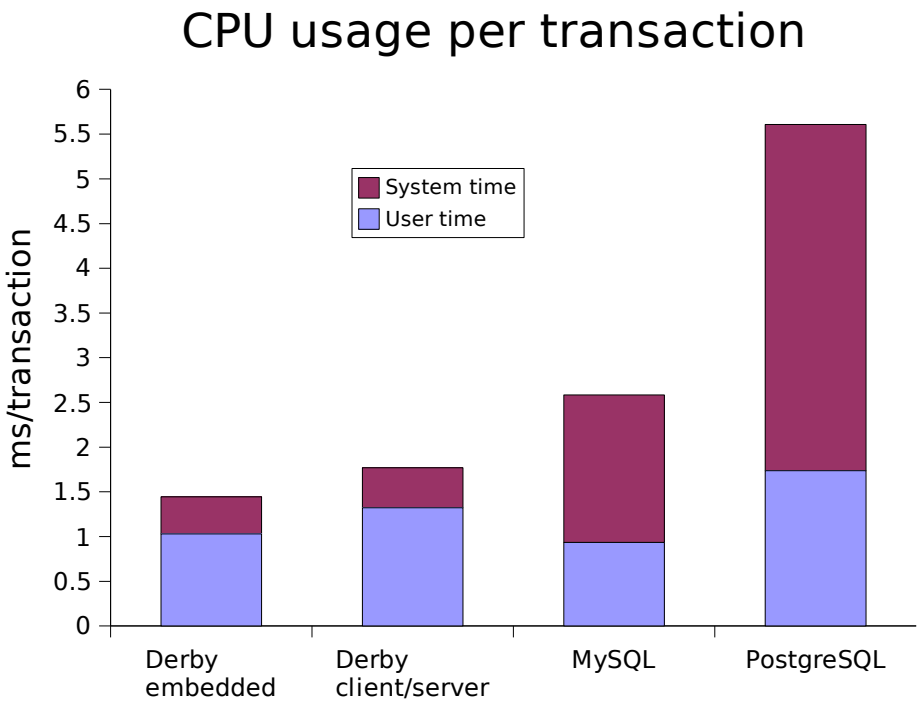
CPU Usage Main-Memory Database



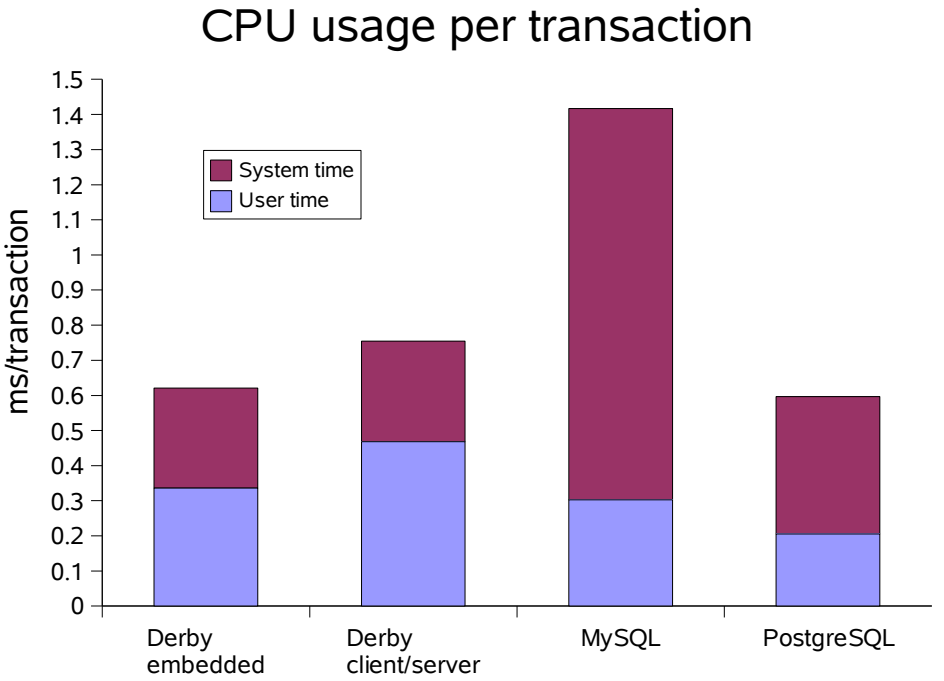


CPU Usage Disk-Based Database

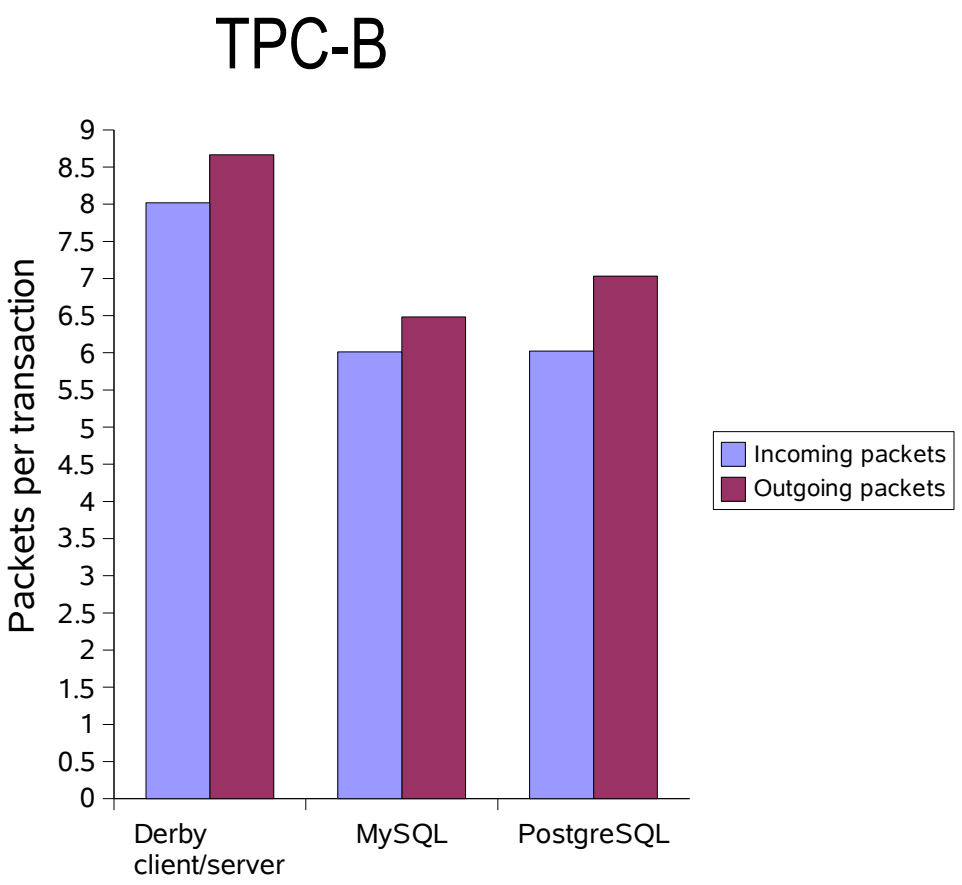
TPC-B

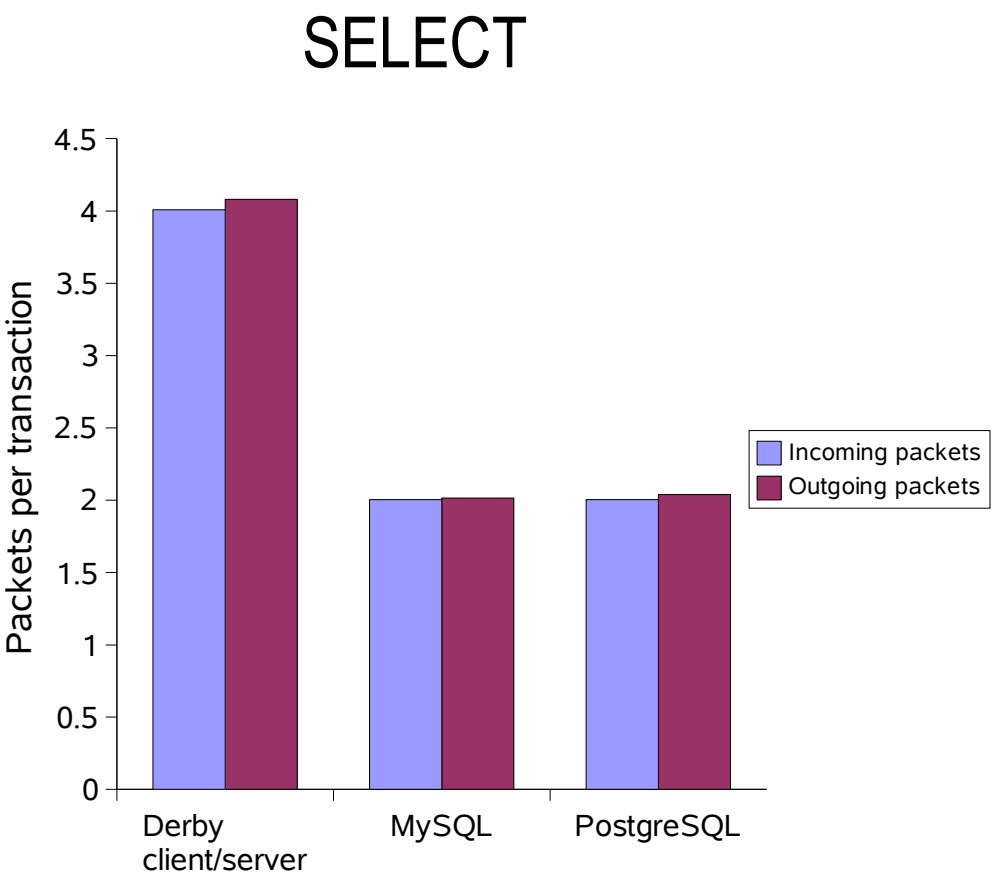


SELECT



Network I/O

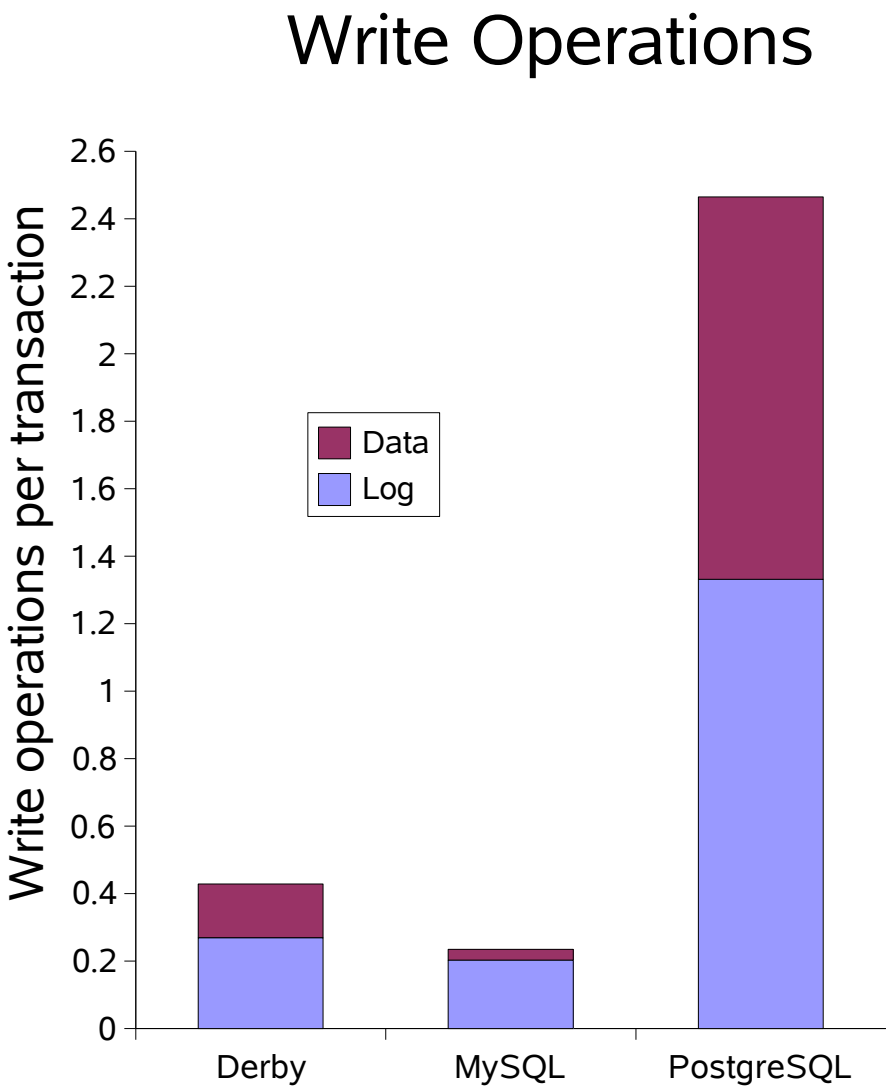


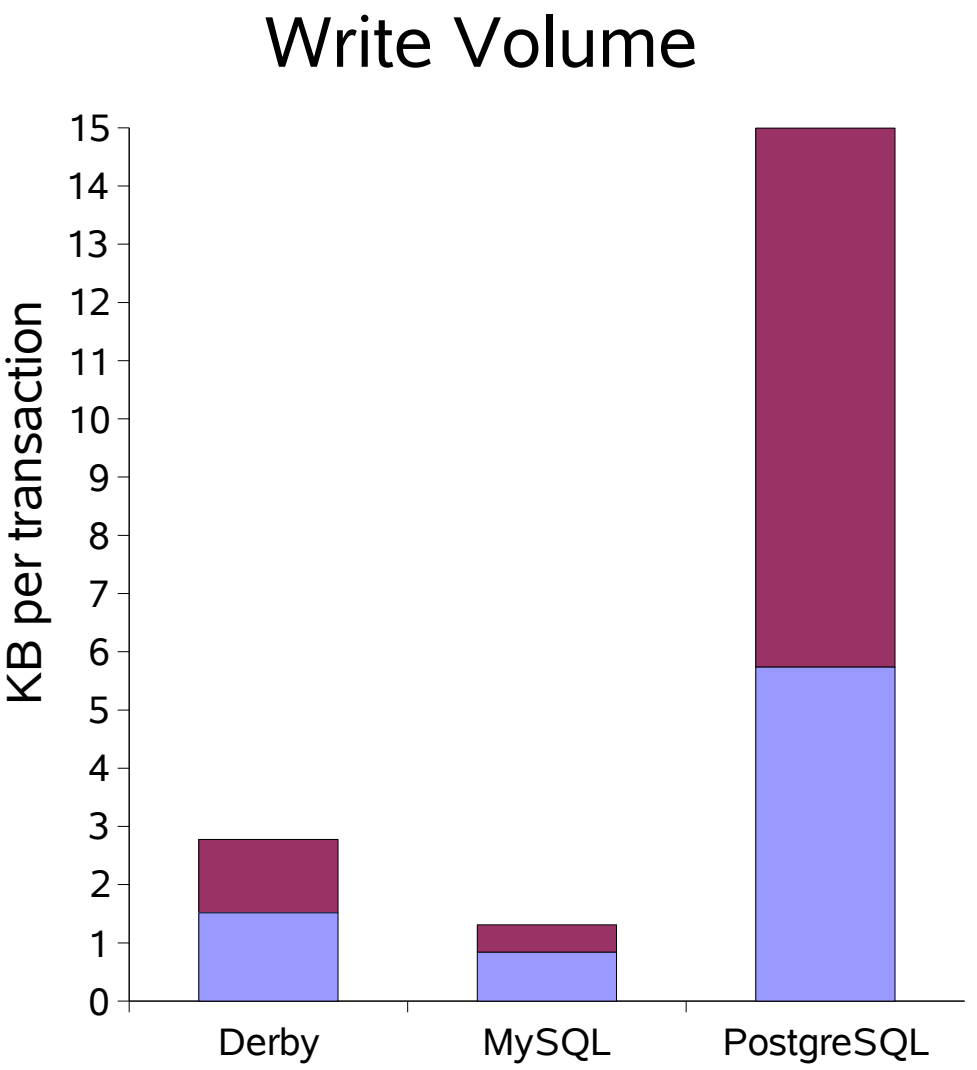


Observation:

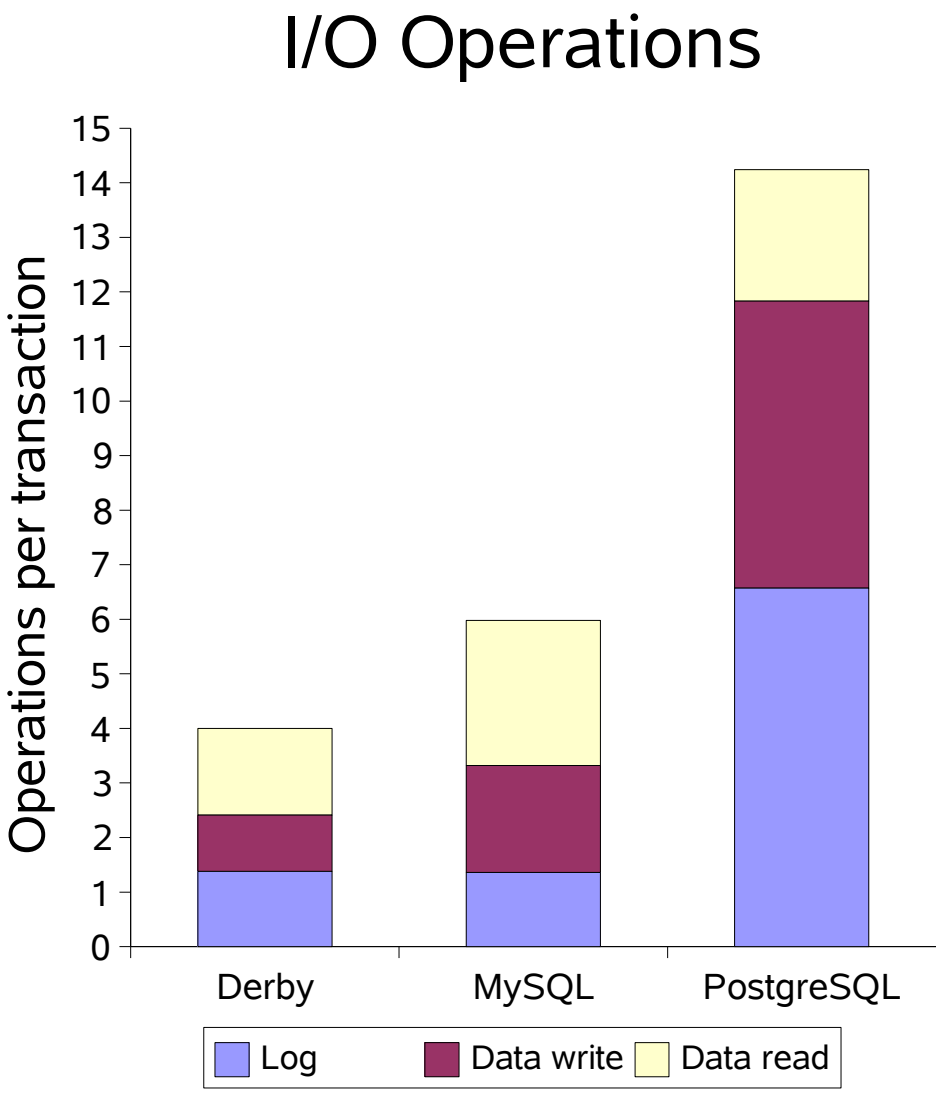
- Derby has two extra round-trips for SELECT operations

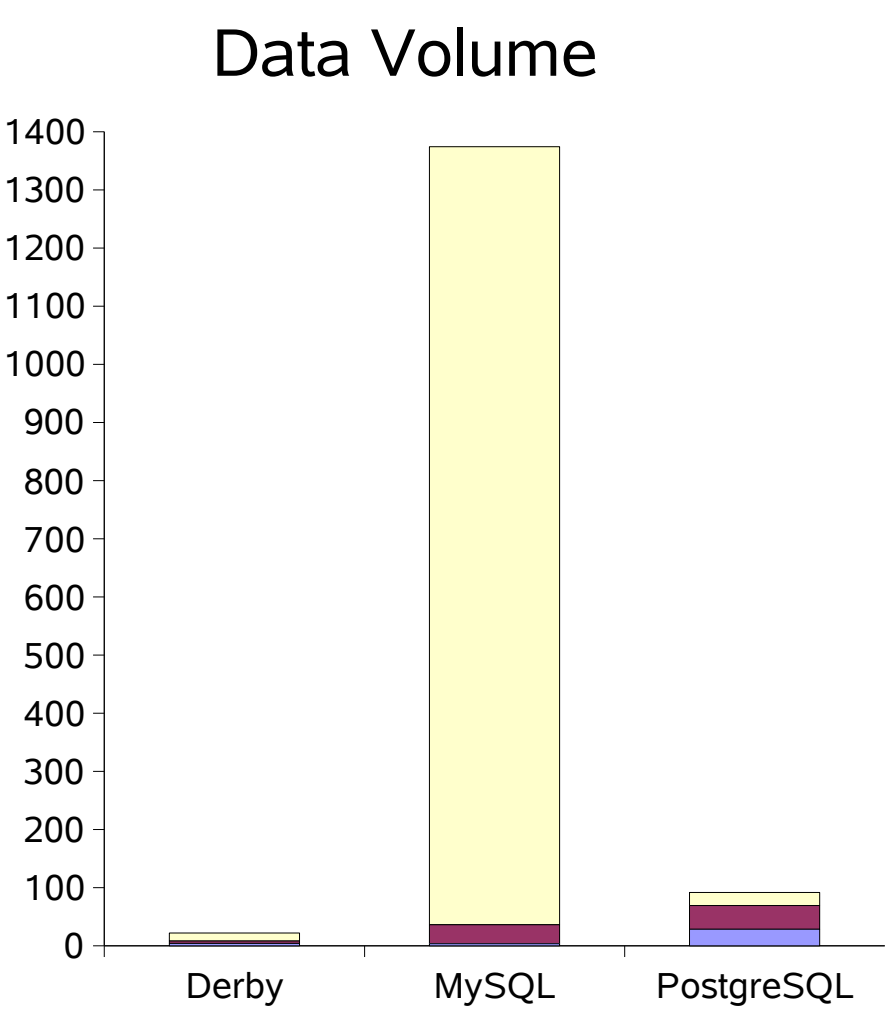
Disk I/O: Main-Memory Database (TPC-B)



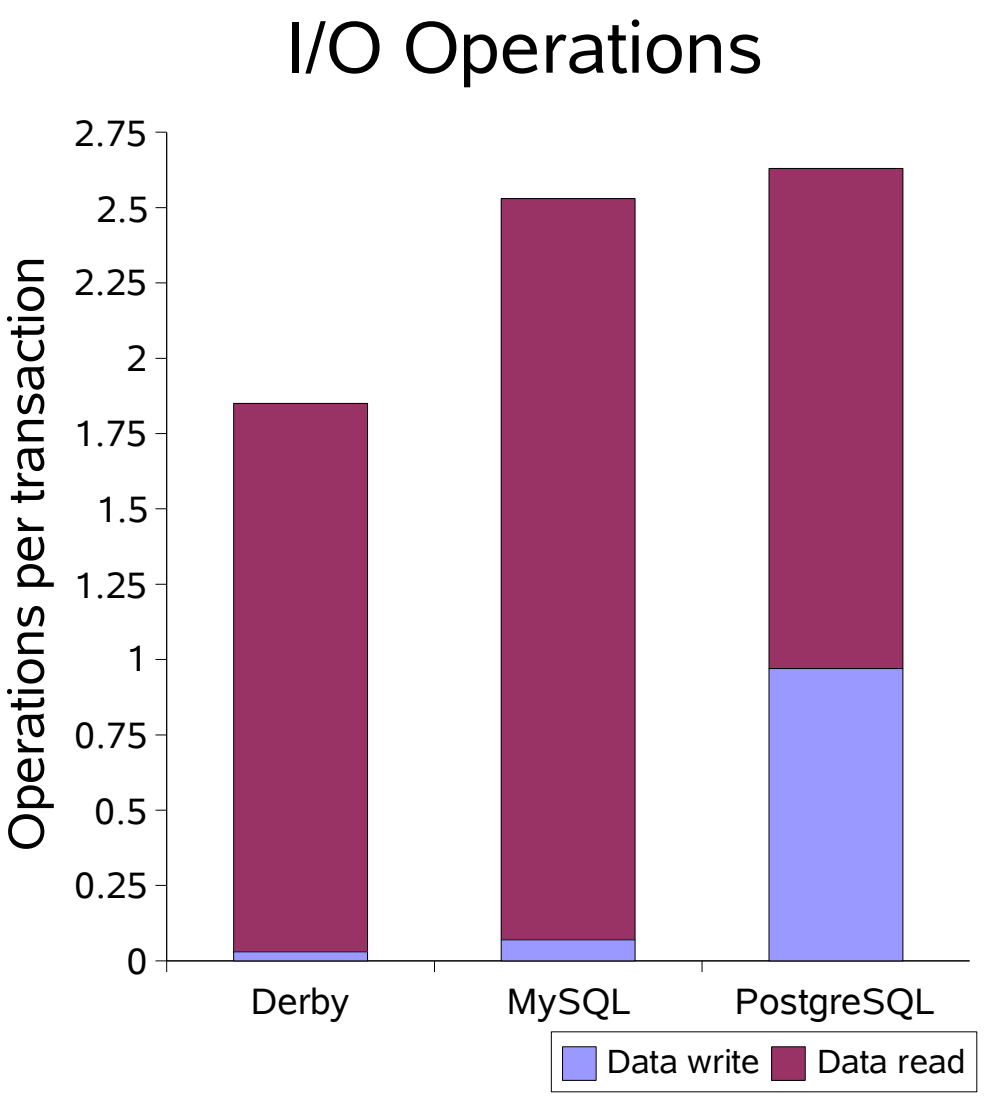


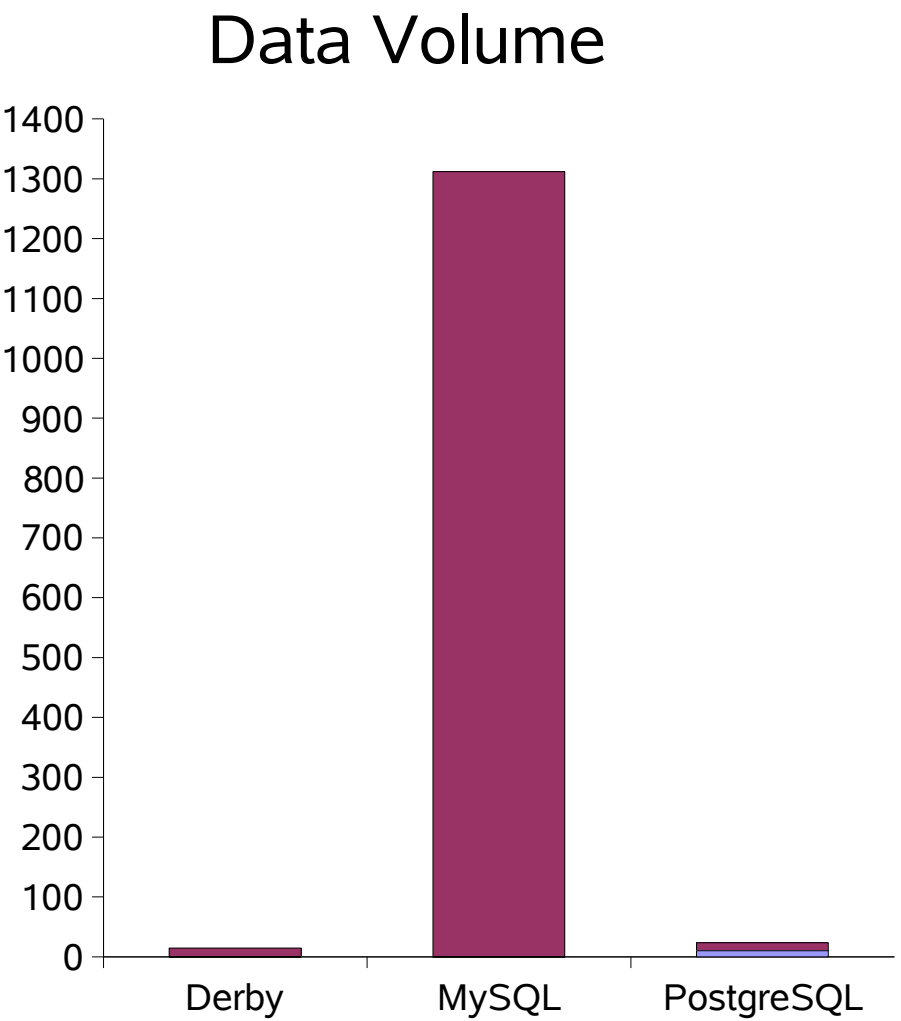
Disk I/O: Disk-Based Database (TPC-B)





Disk I/O: Disk-Based Database (SELECT)





Conclusions: Resource Usage

- MySQL performs better than Derby when
  - The database is small and fits in the database buffer
  - Throughput becomes CPU-bound
  - Derby uses more CPU and sends more messages over the net
- Derby performs better than MySQL when
  - The database is large and does not fit in the database buffer
  - Throughput becomes I/O-bound
- PosgreSQL performs best with read-only load
  - Update-intensive load results in much disk I/O

Performance Improvement Activities

CPU usage:

- High CPU usage during initialization of database buffer (DONE)
- Reduce CPU usage in network server (message parsing and generation)
- Improve use of hash tables and reduce lock contention

Network IO:

- Reduce number of messages sent and received

Disk IO:

- Improve fairness of scheduling of I/O requests (DONE)
- Allow concurrent read operations on data files

Performance tip:

- Next version of Derby will have even better performance

Summary

- **Disk Write Cache:**
  - > Improves the performance, but.....
  - > WARNING: durability and the probability of successful recovery after a power failure are reduced
- **Data and log devices:**
  - > Keep the transaction log and database on a separate devices
- **Database buffer:**
  - > Keep frequently accessed data in the database buffer
- **Client-Server versus Embedded:**
  - > Throughput: down by 5% (TPC-B) to 30% (SELECT)
- **Use:**
  - > Prepared statements and indexes

Questions?

Olav Sandstå, Dyre Tjeldvoll, Knut Anders Hatlen

Olav.Sandstaa@sun.com, Dyre.Tjeldvoll@sun.com,
Knut.Hatlen@sun.com