The Multimed System

T.-I. Salomie, I. Subasu, J. Giceva, G. Alonso

Systems Group, ETH Zurich
A story about
"Data, Data, Data!"
… and multicores
Multicores – a challenge for system software
Multicores – a challenge for system software

Throughput (transactions/sec)

# Cores

Linear scalability
Multicores – a challenge for system software

Throughput (transactions/sec) vs. # Cores

- Linear scalability
- Actual throughput of DBMS decreases with the #cores
Multicores – a challenge for system software
Multicores – a challenge for system software

Workload interaction

Throughput (transactions/sec)

- Workload A
- Workload B
- Workload 0.89*A + 0.11*B

4 cores
48 cores
Multicores – a challenge for system software
Multicores – a challenge for system software

- Increasing number of hardware contexts

- Higher contention on shared data structures
  - Spinning and blocking locks (Johnson, DaMoN'09)
  - MCC-DB (Rubao, VLDB'09)

- Impact of updates and large scans
  - Crescando (Unterbrunner, VLDB'09)

- Design for concurrent operations, yet not for parallelism
  - Shore-MT (Johnson, EDBT'09)

- Increasing load interaction
  - Multimed (Eurosys'11)
Database trends for multicores
Database trends for multicores

- **Fix & pray!**
  
  ```java
  while (!scales) {
    find_bottleneck();
    fix_it();
  }
  ```
  
  - Shore-MT, MCC-DB, Most of commercial DBMS, …

- **Throw out the old things!**
  - Research: column stores, main memory, shared scans, FPGA/GPU
  - Appliances: Teradata, Netezza/IBM

- **Be smart! 😊**
Multimed’s approach
Multimed’s approach

- Based on **single-master data replication**
  - Ganymed (Plattner, Middleware'04), Byzantium (Garcia, Eurosys'11)
  - Proven approach for clusters
  - Not a universal solution for all workloads

... within the same machine

- View the multicore as a **distributed system**
  - Barrelfish (Baumann, SOSP'09), fos (Wentzlaff, ACM OSR'09)

... **partition resources** into clusters
  - Cerberus (Song, Eurosys'11)

... and run a database on each partition

**Non-intrusive** approach for scaling DBMSs to multicores
Multimed architecture
Multimedi architecture
Multimed architecture

Master Node
- Holds all the data
- Always has the freshest version
- Performs all the write operations
Multimed architecture

**Satellite Node**
- Holds full or partial replica
- Performs read operations
- May not be up-to-date
- Many possible optimizations
Multimed architecture
Multimed architecture
Multimed architecture
Multimed architecture
Multimed execution

- **Master Node**: DB (Database), Computational Node.
- **Satellite Node**: DB Replica, Computational Node.
- **Multimed Dispatcher**
- **Multimed Interface**
- **Updates/Queries**
- **WriteSets**
- **Queries**
- **Clients**

System architecture diagram showing the flow of updates, queries, and responses between master and satellite nodes.
Multimed execution

- Master Node
- Satellite Node
- Satellite Node
- Computational Node
- DB
- DB Replica
- Updates/Queries
- WriteSets
- Queries
- Clients
- Multimed Interface
- Query
Multimed execution
Multimed execution
Multimed execution
Multimed execution
Multimed execution

Diagram showing the execution process involving Master Node, Satellite Nodes, Computational Nodes, DB Replicas, Updates/Queries, WriteSets, and Clients.
Multimed execution
System deployment, 48 core, 4 socket AMD Magny-Cours
System deployment, 48 core, 4 socket AMD Magny-Cours
System deployment,
48 core, 4 socket AMD Magny-Cours

Multimed

Master Node

L3

L3

L3
System deployment, 48 core, 4 socket AMD Magny-Cours

Multimed

Master Node

Satellite Node

Satellite Node

Satellite Node

Satellite Node

Satellite Node
Multimed's scalability
Multimed's scalability

Throughput (transactions/sec) vs. # Cores

- Red line: Linear scalability
- Green line: Non-linear scalability
- Blue line: Initial peak with subsequent linear scalability
Multimed's scalability

Ideal scalability for PostgreSQL

Throughput (transactions/sec)

# Cores

0 4 8 12 16 20 24 28 32 36 40 44 48

0 1000 2000 3000 4000 5000 6000
Multimed's scalability

Ideal scalability for PostgreSQL
Multimed's scalability

Throughput (transactions/sec) vs # Cores

Ideal scalability for PostgreSQL

PostgreSQL's throughput decreases with the #cores
Multimed's scalability

Ideal scalability for PostgreSQL

Multimed's throughput increases linearly with the #satellites (#cores)

PostgreSQL's throughput decreases with the #cores
Multimed's scalability

Throughput (transactions/sec) vs. # Cores

- Ideal scalability for PostgreSQL
- Multimed's throughput increases linearly with the #satellites (#cores)
- PostgreSQL's throughput decreases with the #cores

Key points:
- Multimed's scalability is superior to PostgreSQL's.
- Ideal scalability is achieved in PostgreSQL up to 8 cores.
- Beyond 8 cores, PostgreSQL's throughput decreases.
- Multimed's throughput remains high and linear up to 48 cores.
Multimed's scalability

Ideal scalability for PostgreSQL

Multimed's throughput increases linearly with the #satellites (#cores)

PostgreSQL's throughput decreases with the #cores
Multimed's scalability

Ideal scalability for PostgreSQL

Multimed's throughput increases linearly with the #satellites (#cores)

PostgreSQL's throughput decreases with the #cores
Multimed's scalability

Ideal scalability for PostgreSQL

PostgreSQL's throughput decreases with the #cores

Multimed has only 2 cores for routing transactions for 10 satellites

Multimed's throughput increases linearly with the #satellites (#cores)
Why does Multimed work?
Why does Multimed work?

- **Intuitively:**
  - Reduces contention by replication
  - Solves the problems of load interaction by running "heavy" transactions on specific satellite nodes
  - Each engine runs on a small #cores

- **Counter intuitively:**
  - Routing layer adds latency,
  - Replication adds latency, **but**
  - Requests are answered faster by each satellite due to less load interaction and contention, compensating for these latencies
  - Shared caches
Databases and workloads
Databases and workloads

- Target workloads
  - Read-heavy workloads with updates (TPC-W)
    - TPC-W Browsing (5% Updates)
    - TPC-W Shopping (20% Updates)
  - Mainly main memory resident datasets
Experimental results, PostgreSQL

TPC-W Browsing (5% updates), 20GB dataset, 48 cores
Experimental results, PostgreSQL

TPC-W Browsing (5% updates), 20GB dataset, 48 cores

Throughput (transactions/sec)

PostgreSQL, 12 Cores (best performance)
Experimental results, PostgreSQL

TPC-W Browsing (5% updates), 20GB dataset, 48 cores

- Multimed, naïve replication, 48 cores
- PostgreSQL, 12 Cores (best performance)
Experimental results, PostgreSQL

TPC-W Browsing (5% updates), 20GB dataset, 48 cores

- **Multimed, replicas w/o durability, 48 cores**
- **Multimed, naïve replication, 48 cores**
- **PostgreSQL, 12 Cores (best performance)**
Experimental results, PostgreSQL

TPC-W Browsing (5% updates), 20GB dataset, 48 cores

- Multimed, partial replicas, 48 cores
- Multimed, replicas w/o durability, 48 cores
- Multimed, naïve replication, 48 cores
- PostgreSQL, 12 Cores (best performance)
Experimental results, PostgreSQL

TPC-W Shopping (20% updates), 20GB dataset, 48 cores

Throughput (transactions/sec)

# Clients

100 200 300 400 500

0 200 400 600 800 1000 1200 1400 1600
Experimental results, PostgreSQL

TPC-W Shopping (20% updates), 20GB dataset, 48 cores

Throughput (transactions/sec)

# Clients

PostgreSQL, 12 Cores (best performance)
Experimental results, PostgreSQL

TPC-W Shopping (20% updates), 20GB dataset, 48 cores

Throughput (transactions/sec)

# Clients

- PostgreSQL, 12 Cores (best performance)
- Multimed, naïve replication, 48 cores
Experimental results, PostgreSQL

TPC-W Shopping (20% updates), 20GB dataset, 48 cores

- Multimed, replicas w/o durability, 48 cores
- PostgreSQL, 12 Cores (best performance)
- Multimed, naïve replication, 48 cores
Experimental results, PostgreSQL

TPC-W Shopping (20% updates), 20GB dataset, 48 cores

- Multimed, partial replicas, 48 cores
- Multimed, replicas w/o durability, 48 cores
- PostgreSQL, 12 Cores (best performance)
- Multimed, naïve replication, 48 cores
The morale of the story
The morale of the story

Why parallelize when you can distribute?

* for the PowerPoint version that was presented at EuroSys 2011, please send an email to tsalomie at inf.ethz.ch