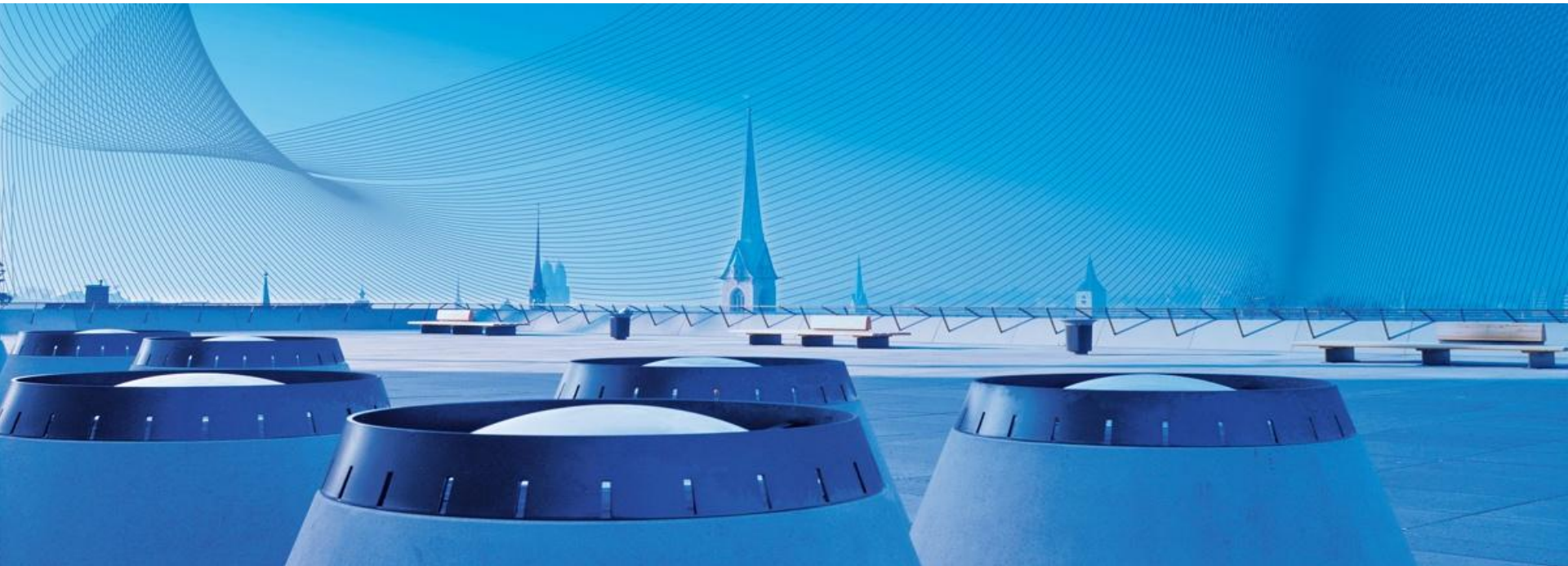
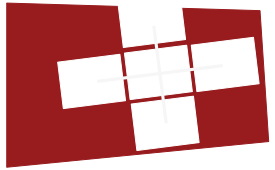


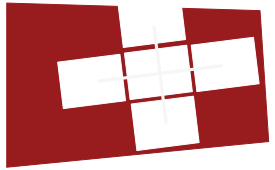
# The Multimed System

T.-I. Salomie, I. Subasu, J. Giceva, G. Alonso  
*Systems Group, ETH Zurich*

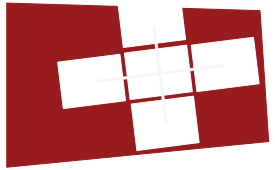




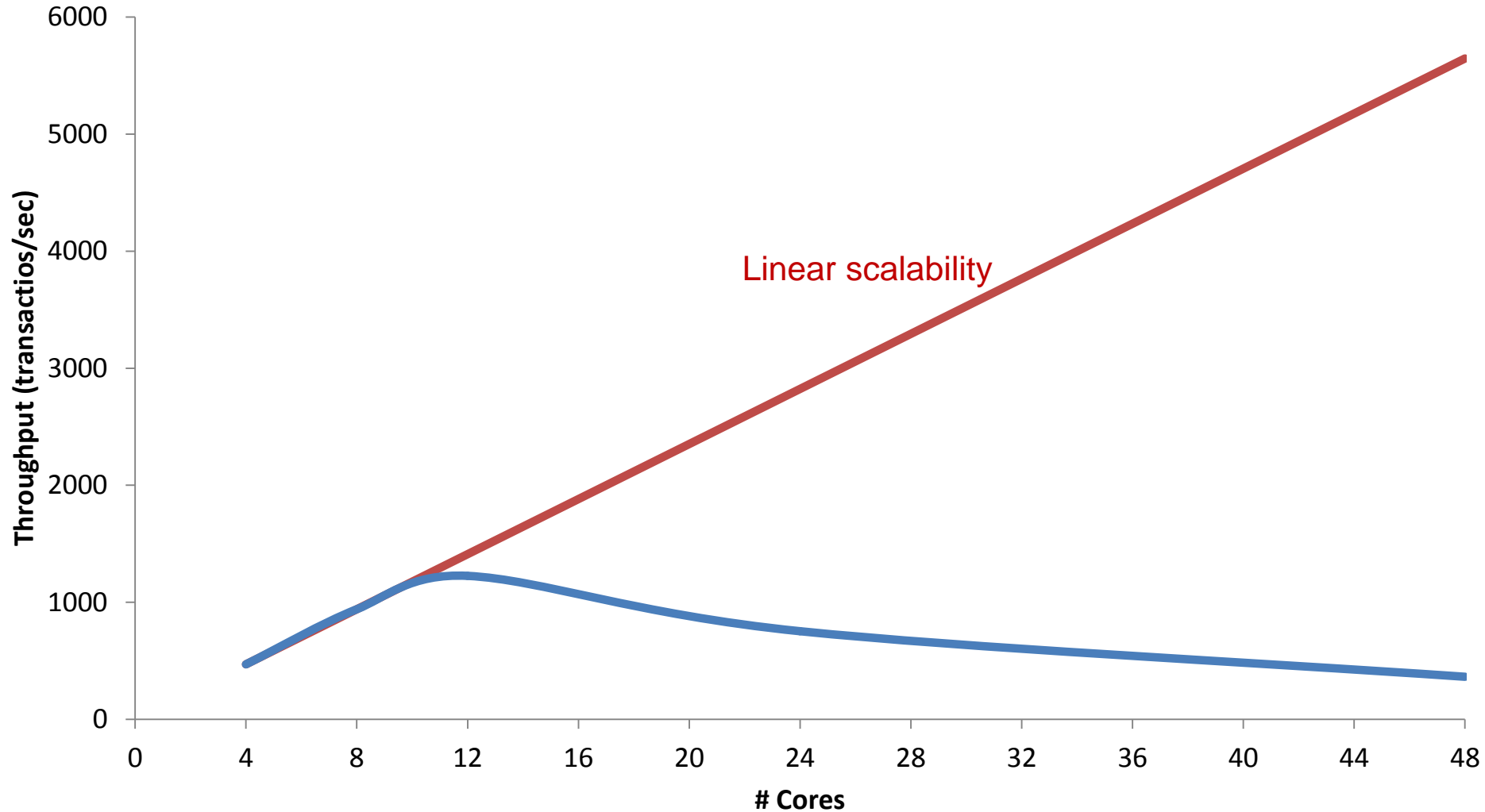
# A story about "Data, Data, Data!" ... and multicores

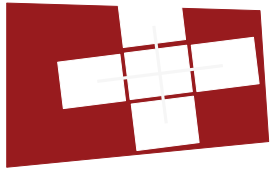


# Multicores – a challenge for system software

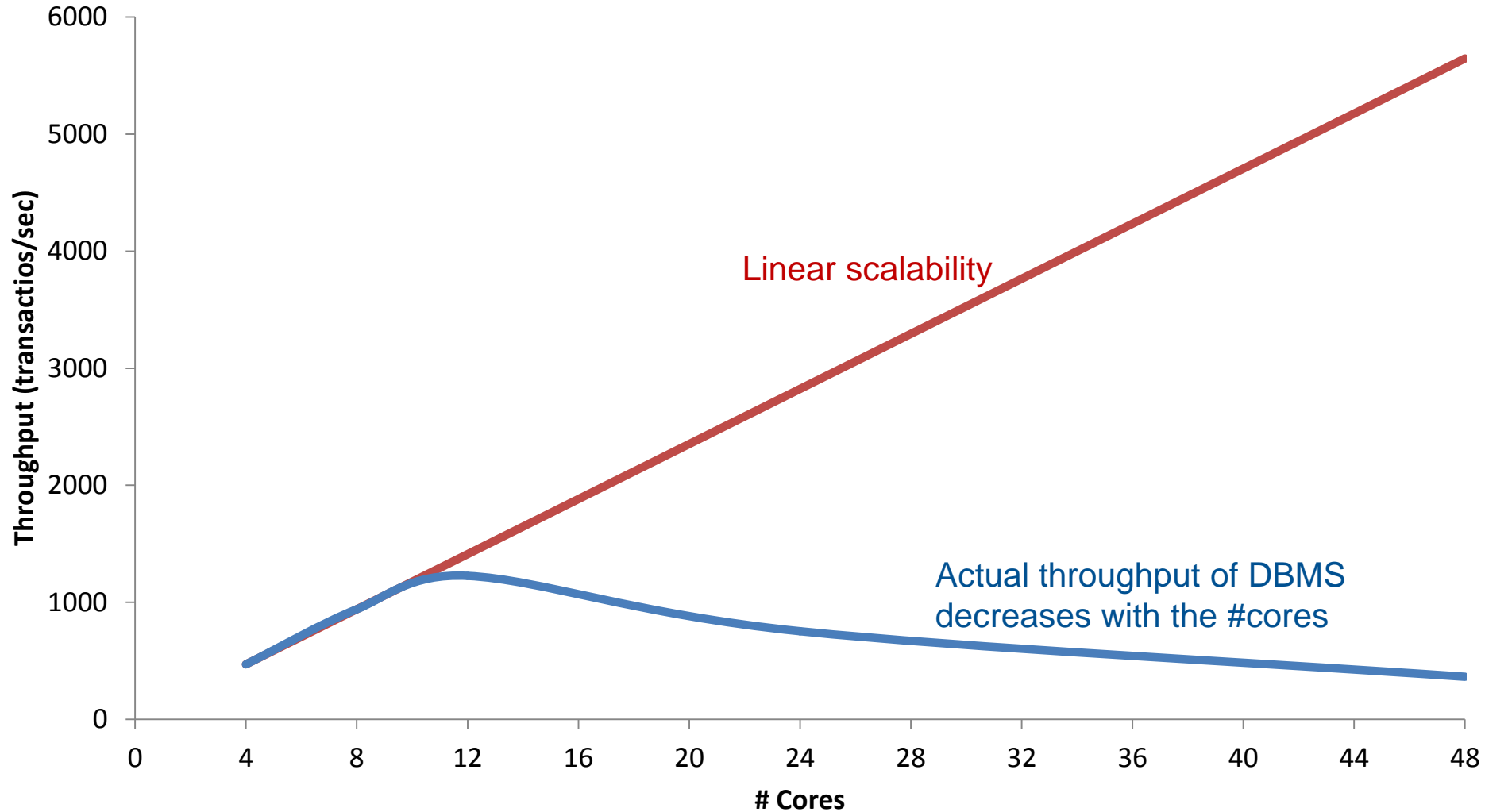


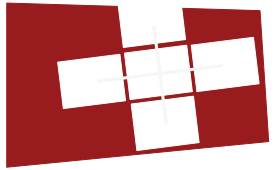
# Multicores – a challenge for system software



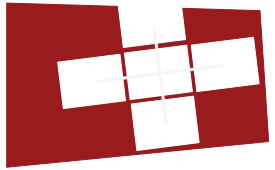


# Multicores – a challenge for system software

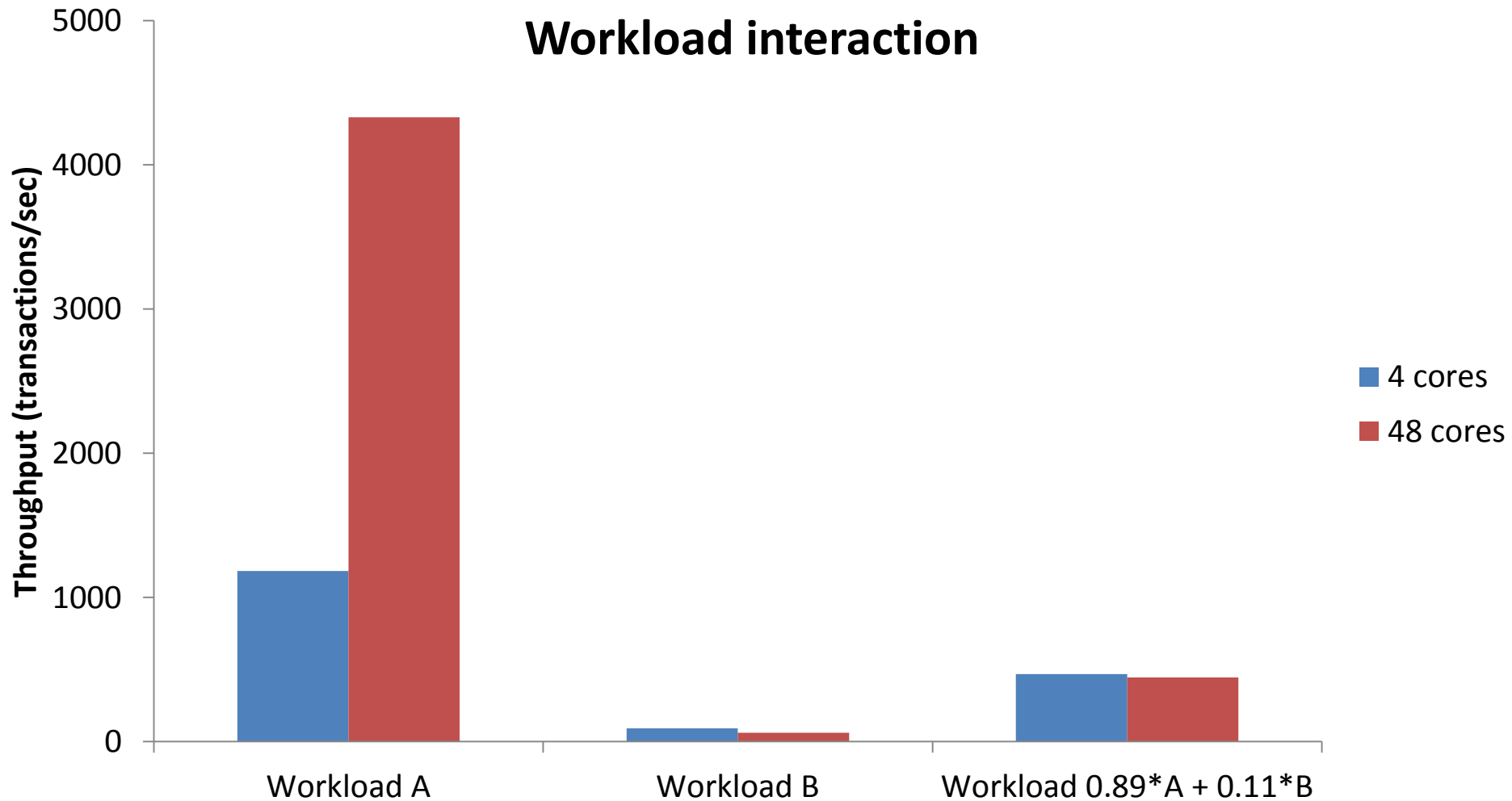


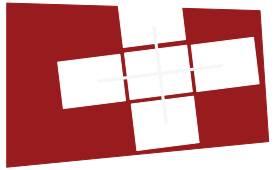


# Multicores – a challenge for system software

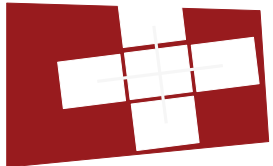


# Multicores – a challenge for system software





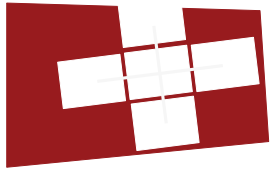
# Multicores – a challenge for system software



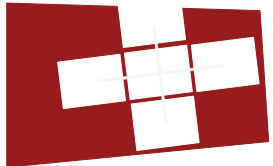
# Multicores – a challenge for system software

## 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 (Eurosyst'11)



# Database trends for multicores



# Database trends for multicores

- **Fix & pray!**

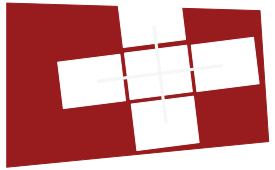
```
while (!scales) {  
    find_bottleneck();  
    fix_it();  
}
```

- Shore-MT, MCC-DB, Most of commercial DBMS, ...

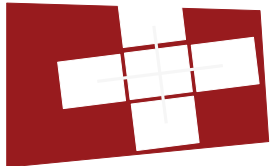
- **Throw out the old things!**

- 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

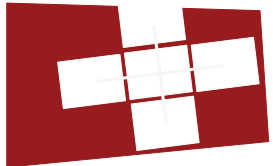
- View the multicore as a **distributed system**
  - Barrelfish (Baumann, SOSP'09), fos (Wentzlaff, ACM OSR'09)

... **partition resources** into clusters

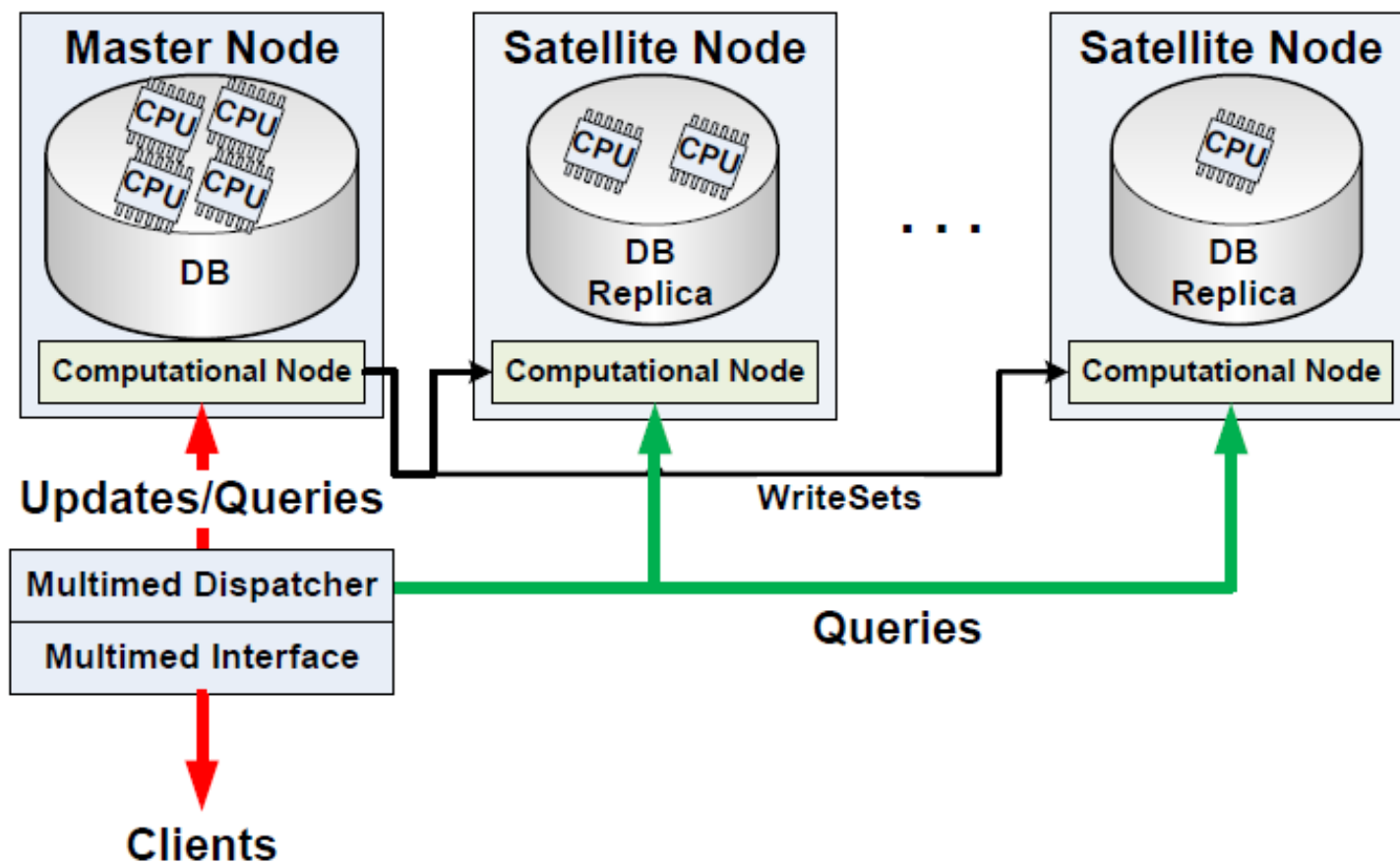
- Cerberus (Song, Eurosys'11)

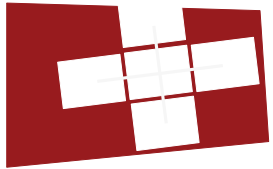
... and run a database on each partition

➔ **Non-intrusive** approach for scaling DBMSs to multicores

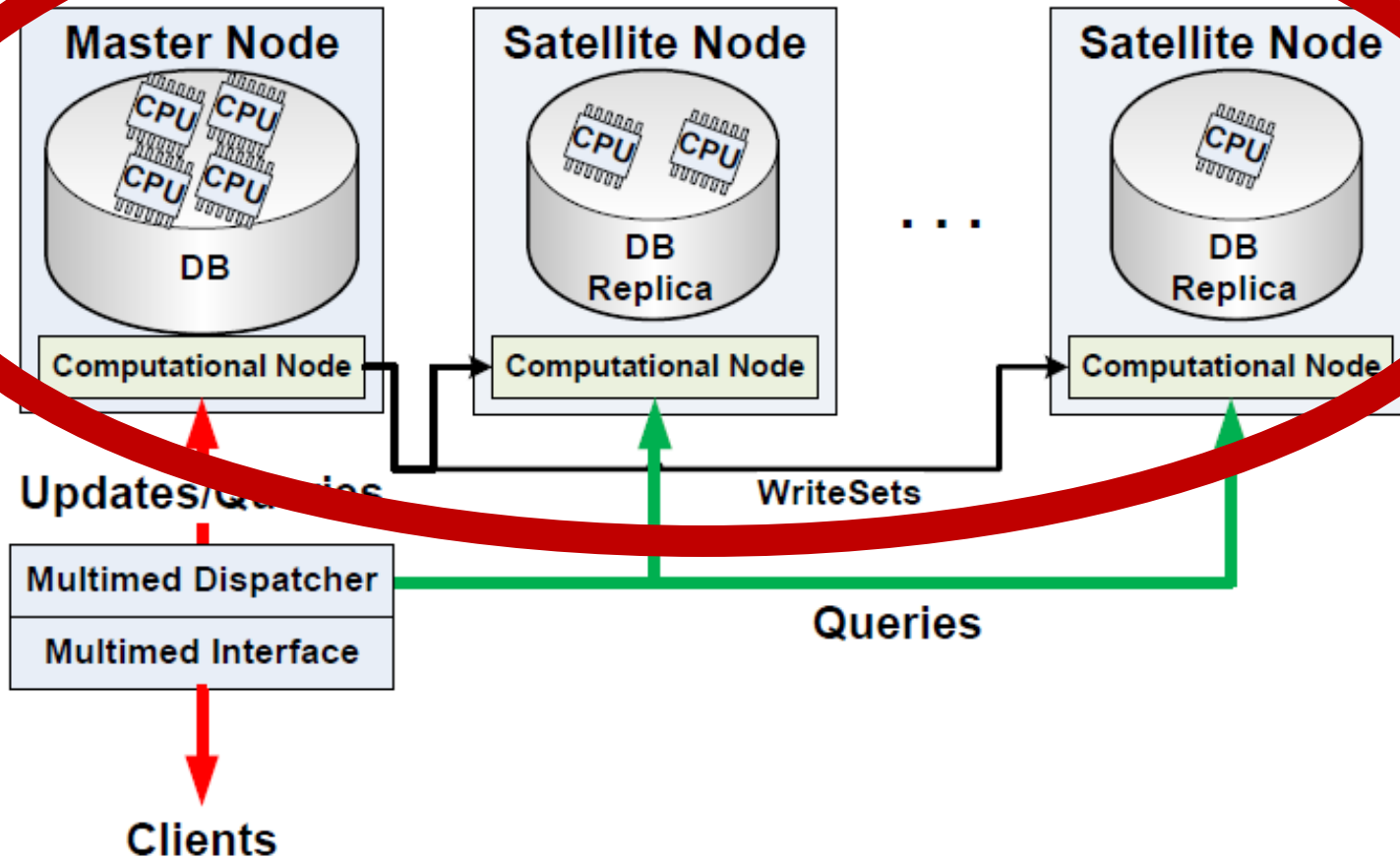


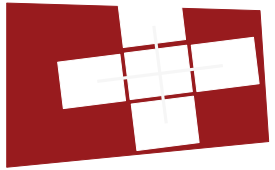
# Multimed architecture



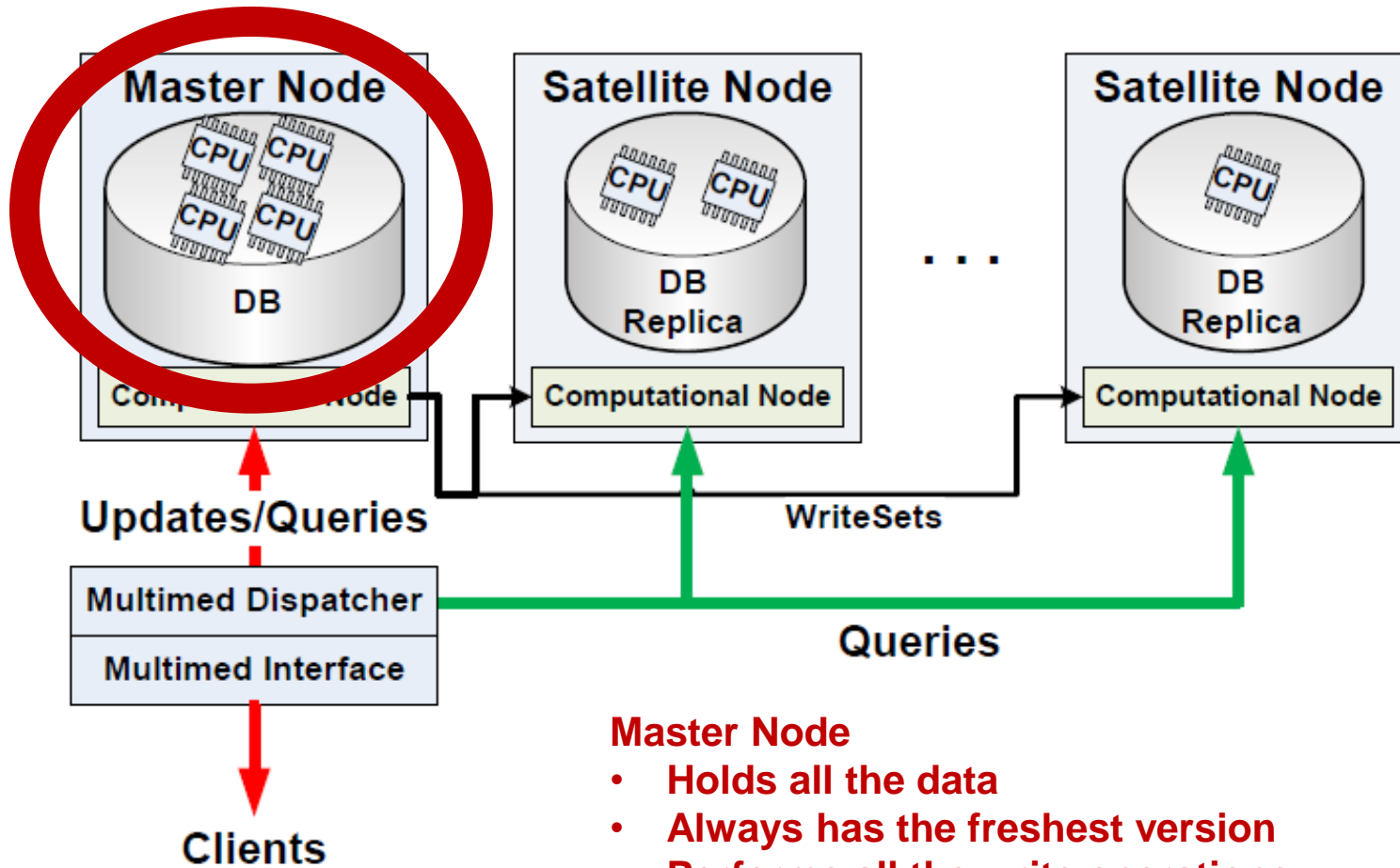


# Multimed architecture



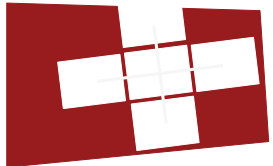


# Multimed architecture

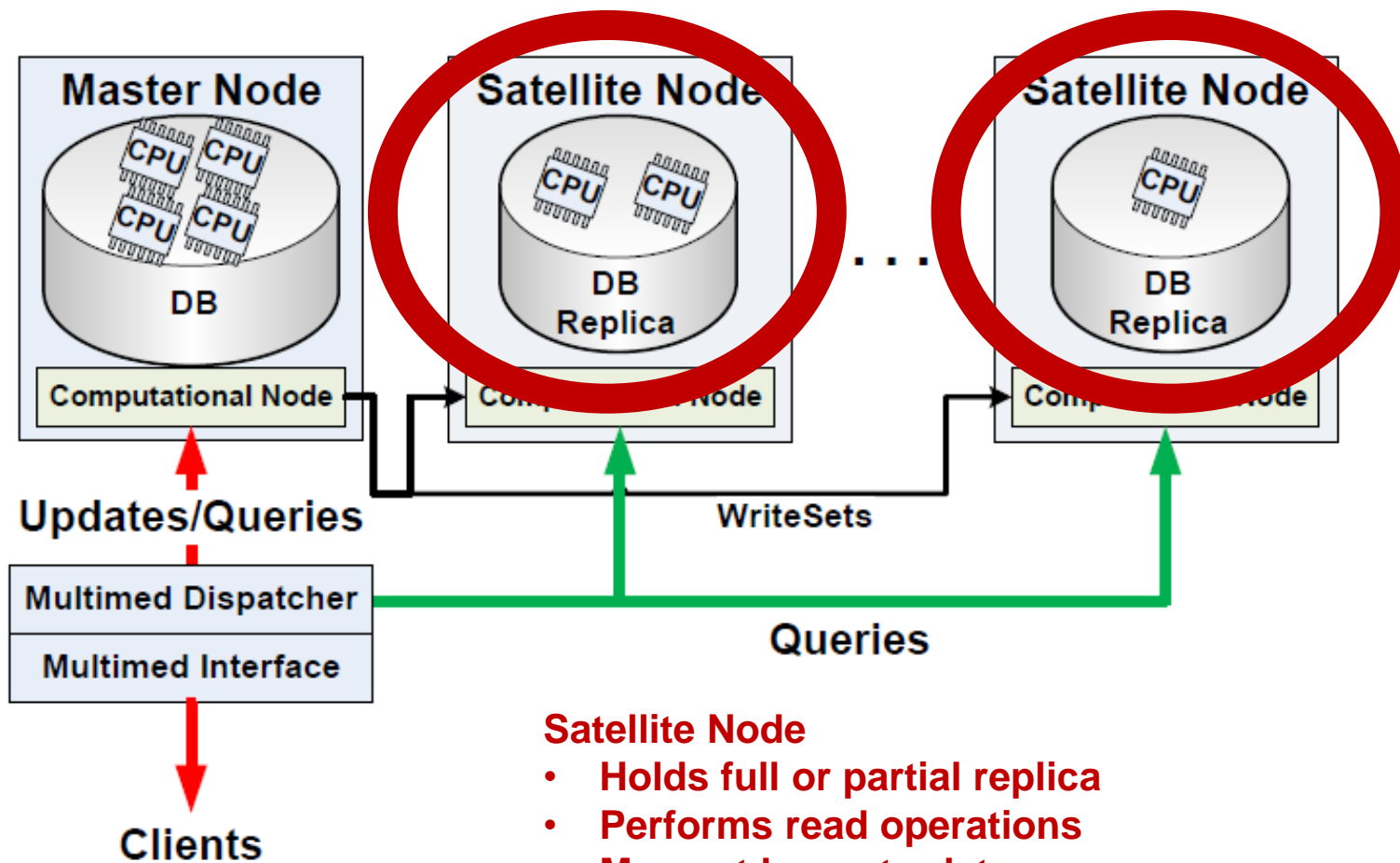


## Master Node

- Holds all the data
- Always has the freshest version
- Performs all the write operations

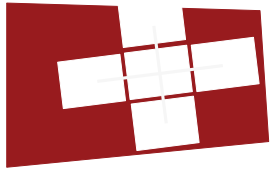


# Multimed architecture

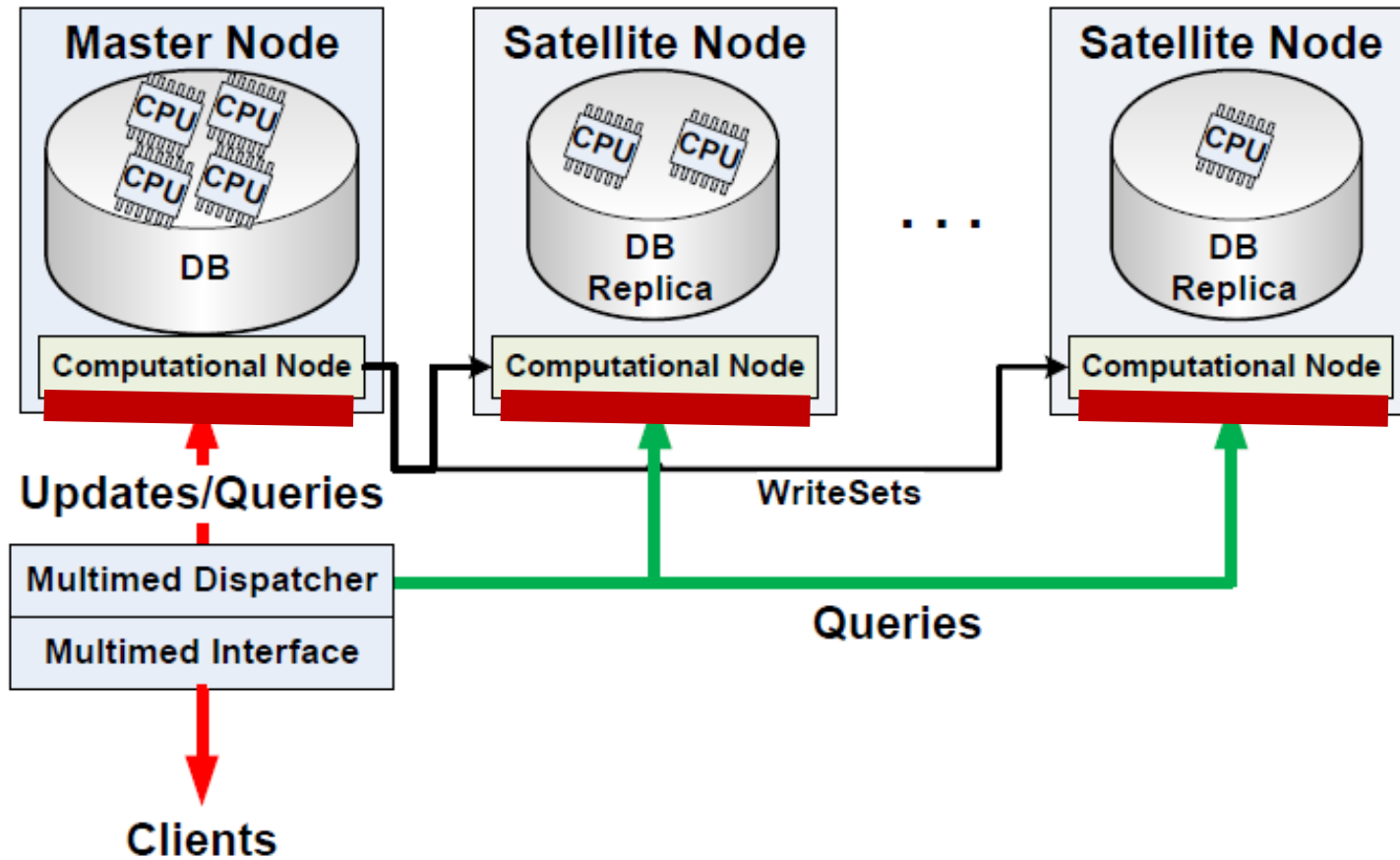


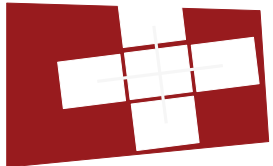
## Satellite Node

- Holds full or partial replica
- Performs read operations
- May not be up-to-date
- Many possible optimizations

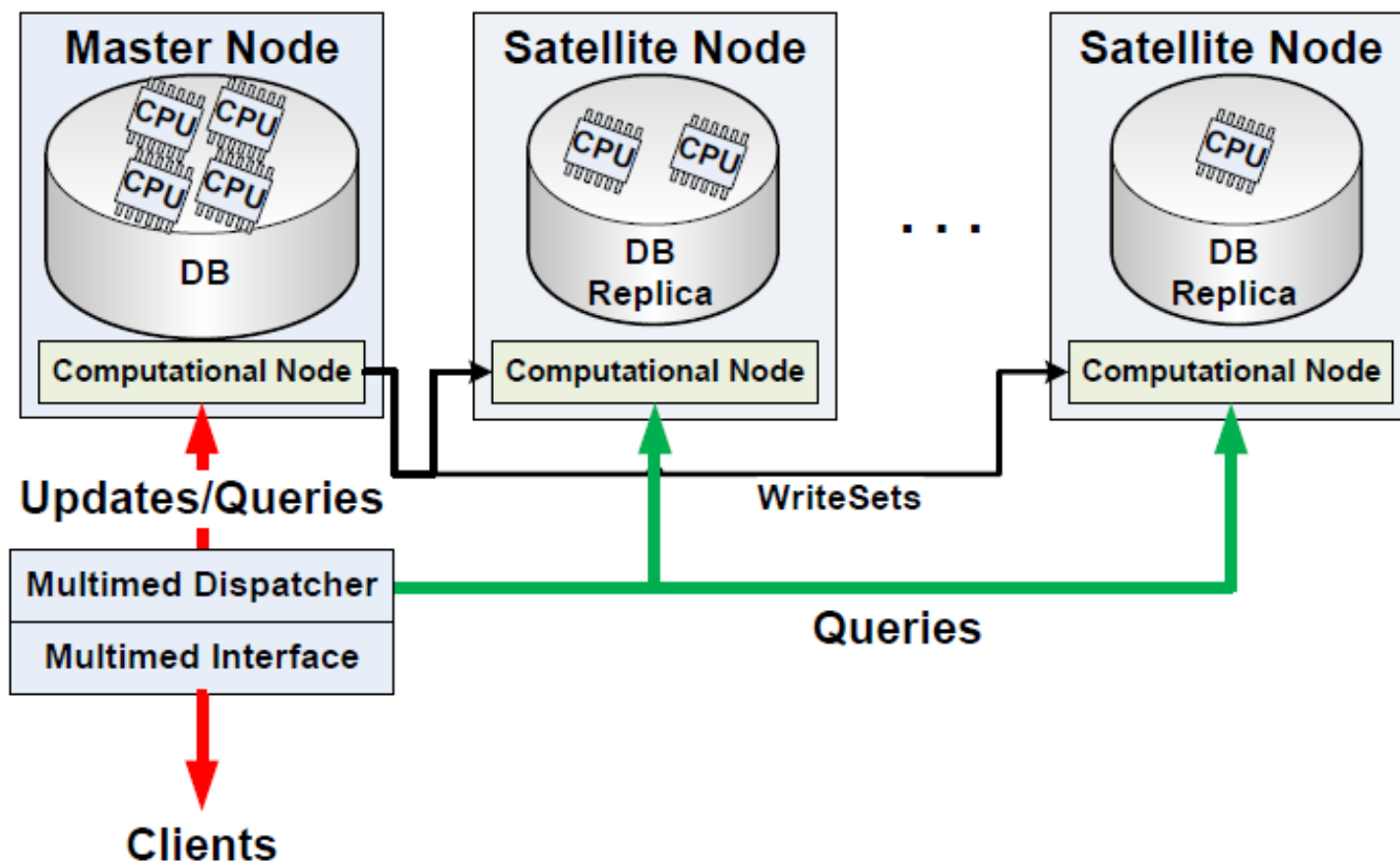


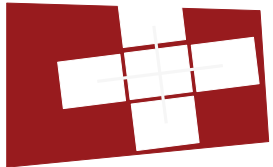
# Multimed architecture



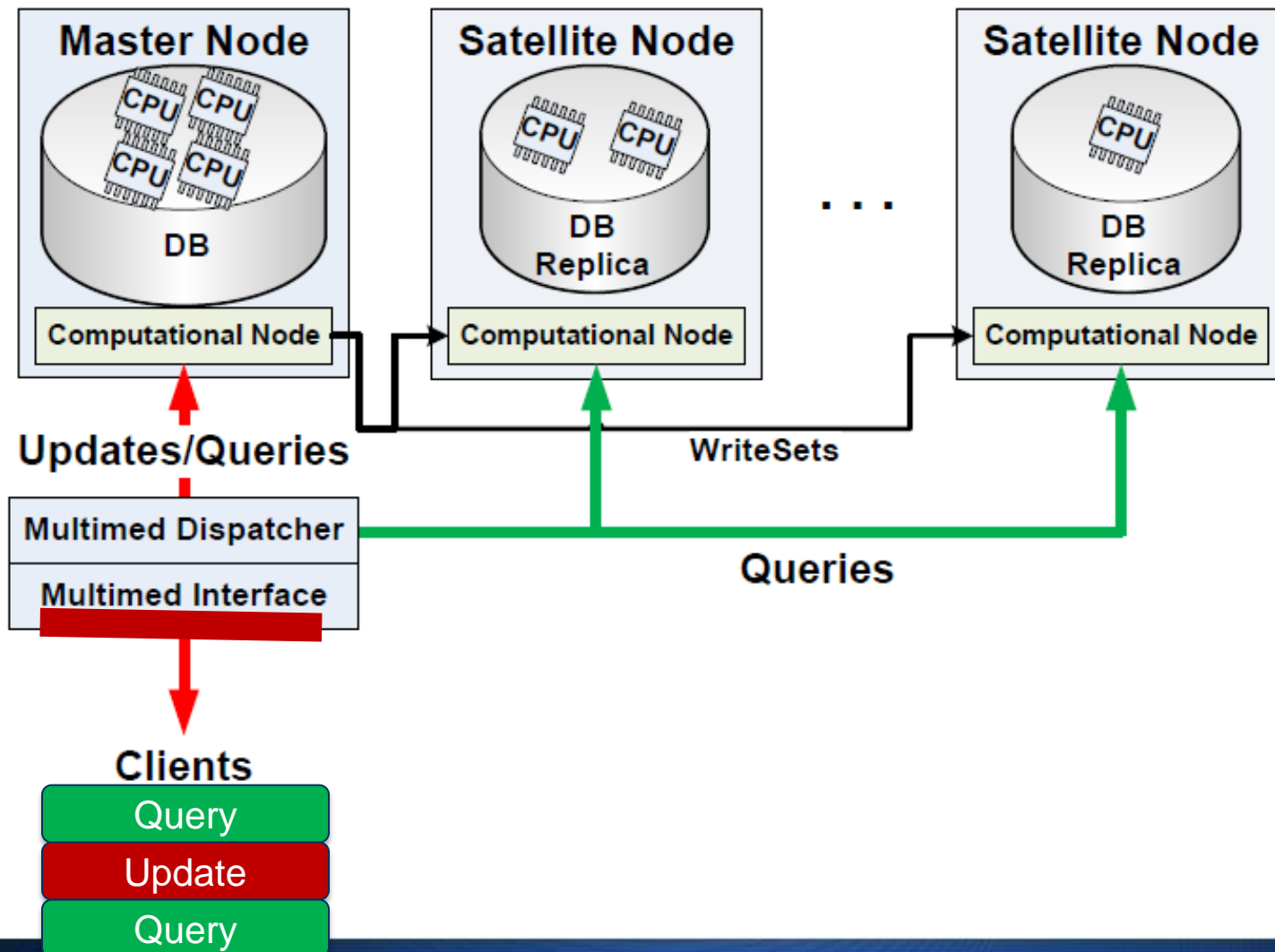


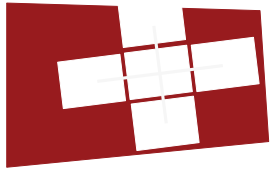
# Multimed architecture



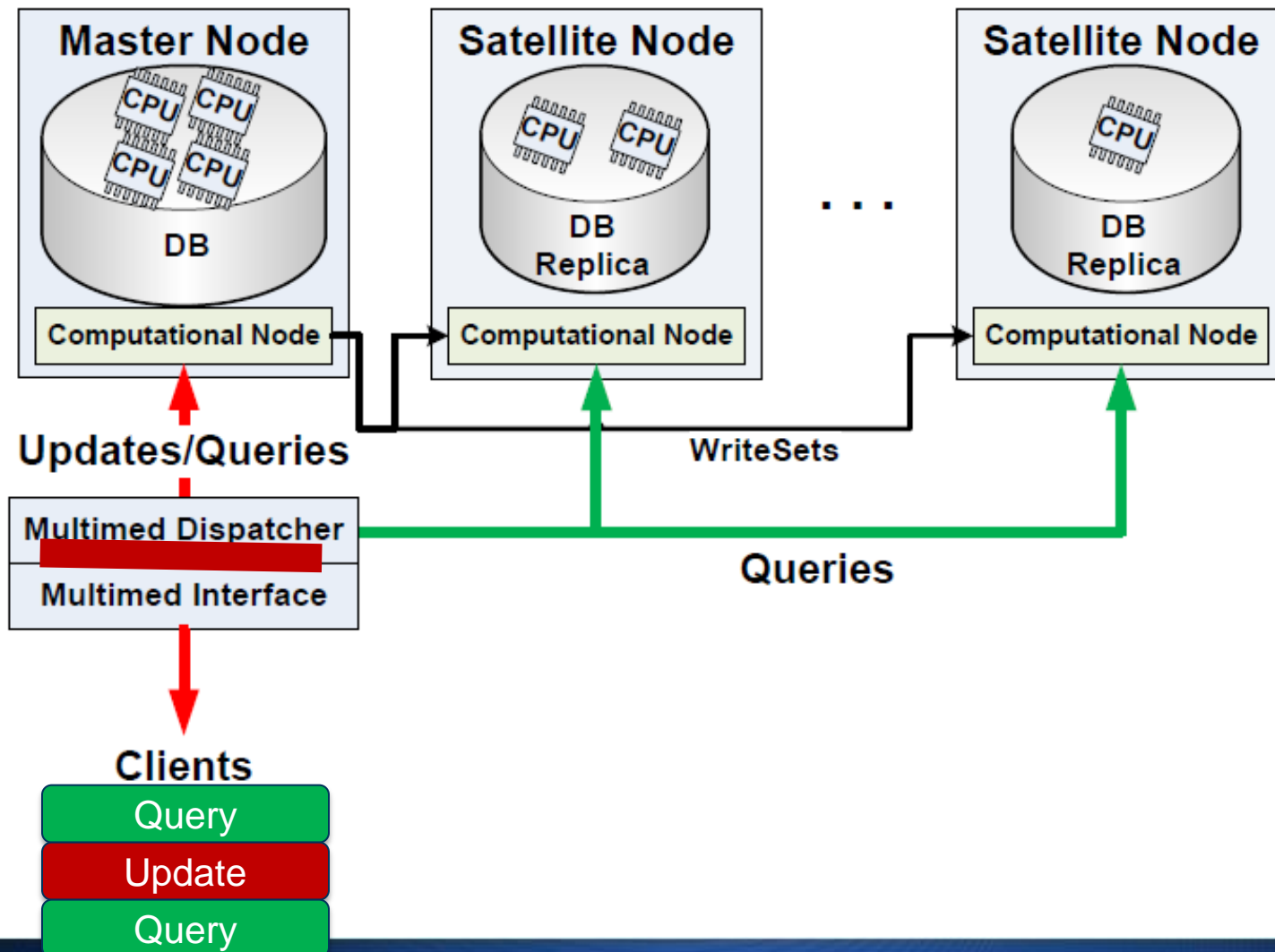


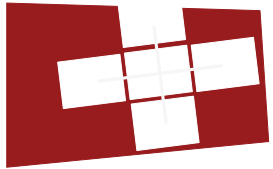
# Multimed architecture



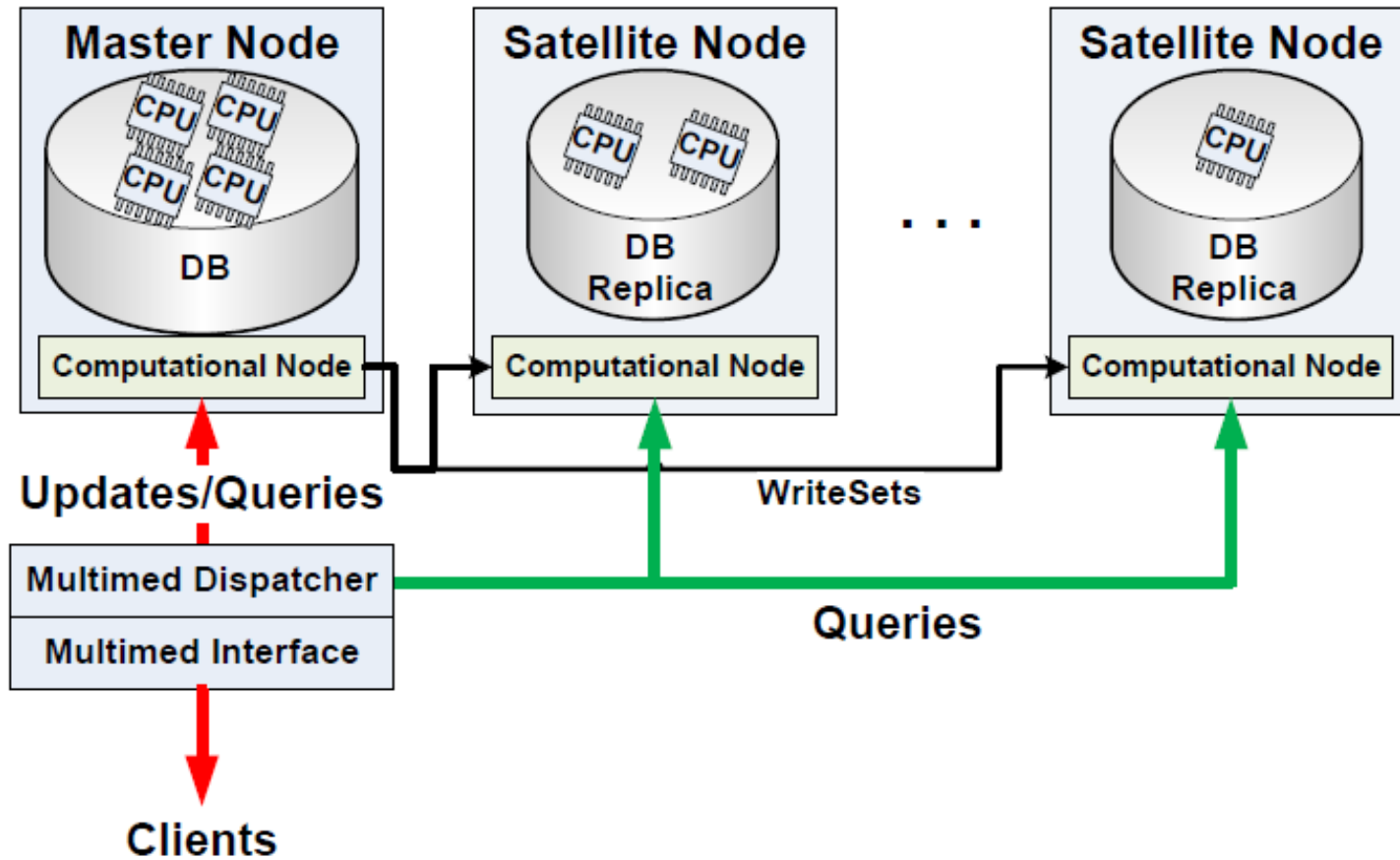


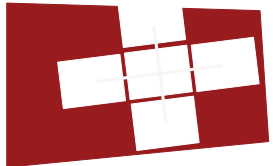
# Multimed architecture



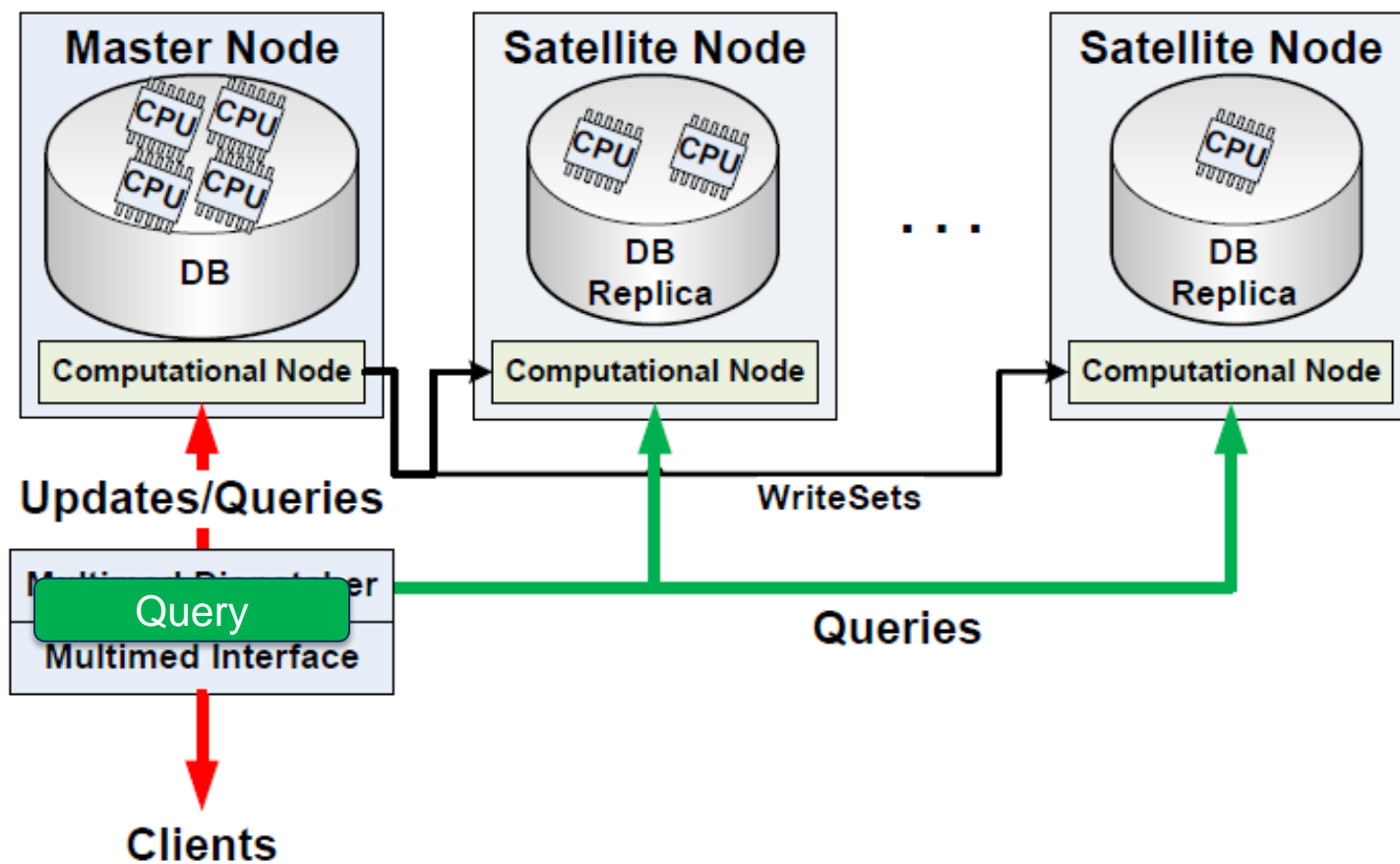


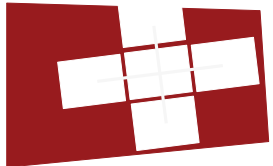
# Multimed execution



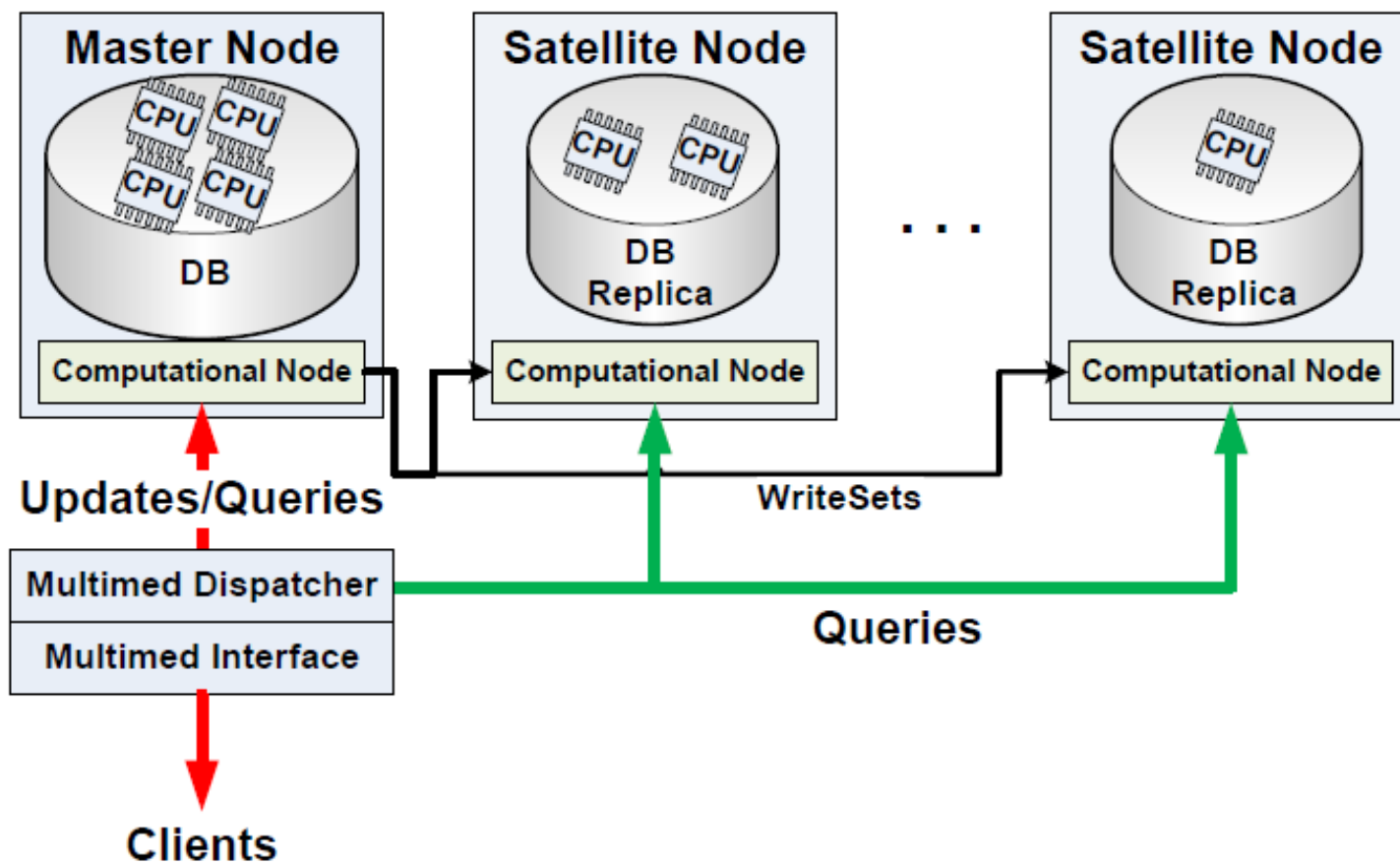


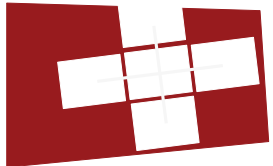
# Multimed execution



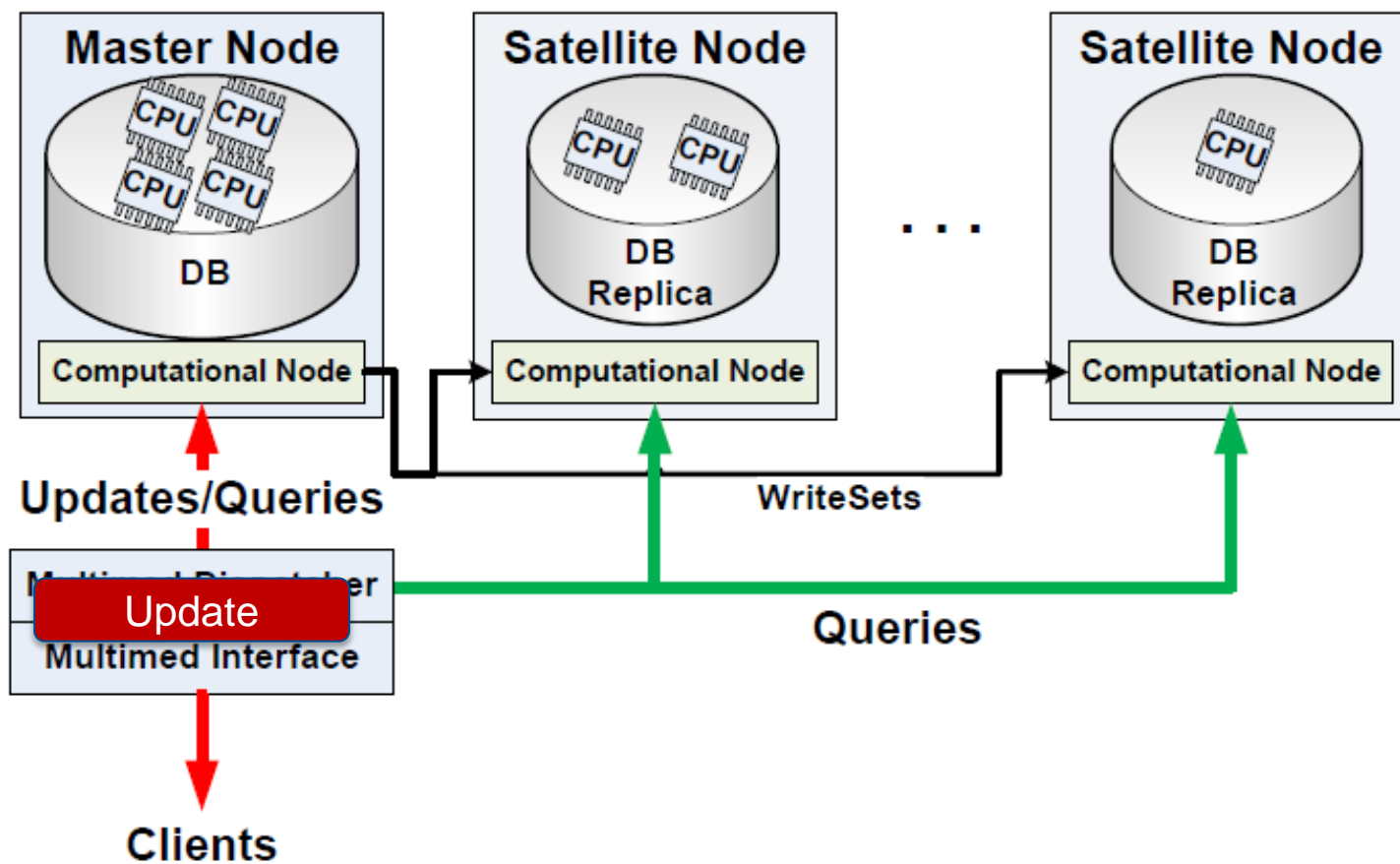


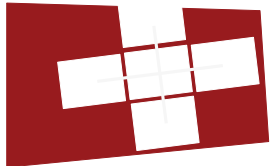
# Multimed execution



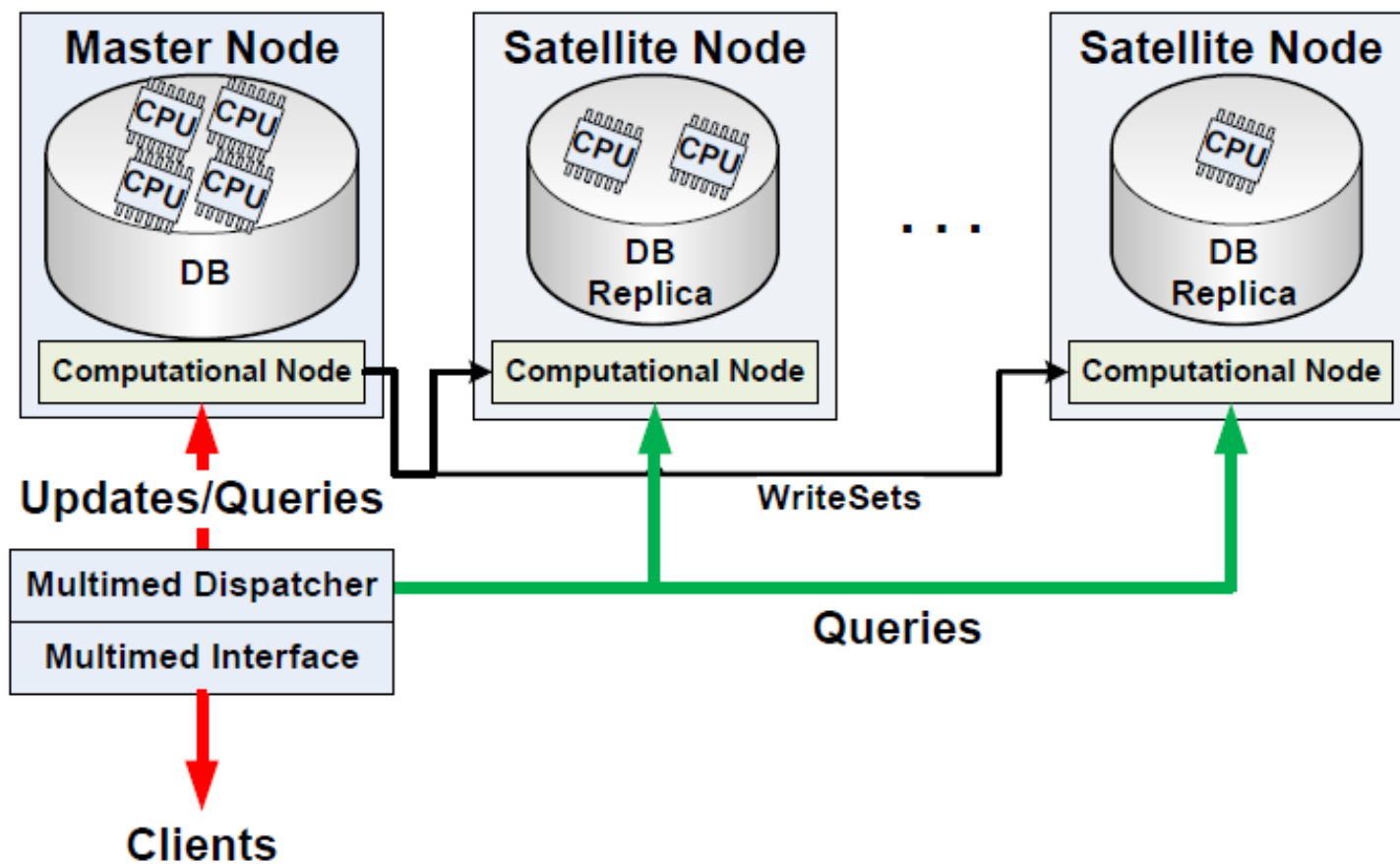


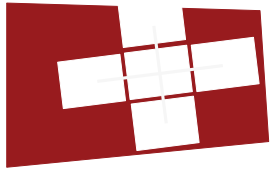
# Multimed execution



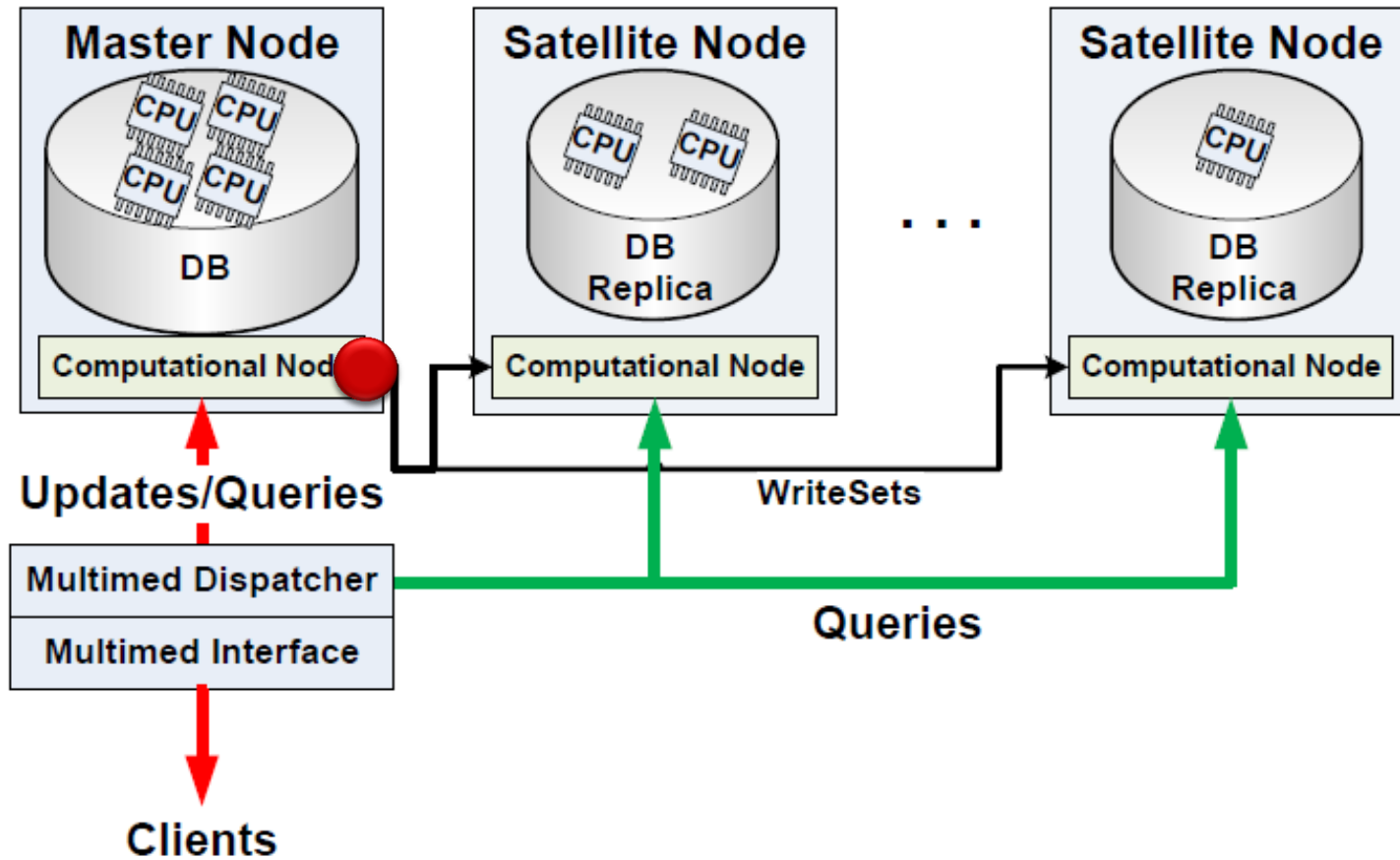


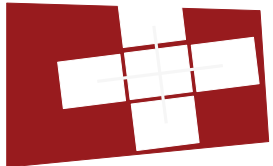
# Multimed execution



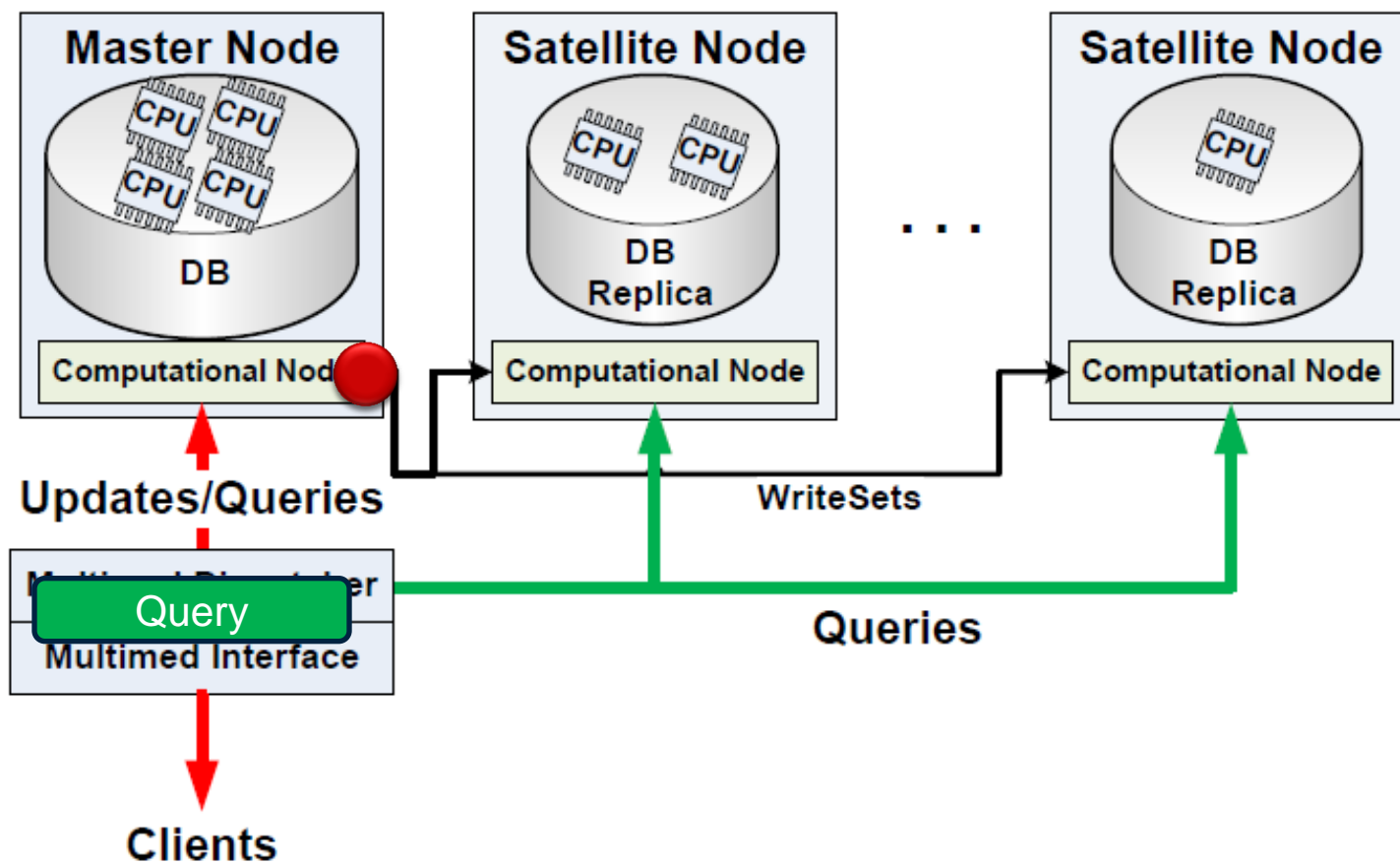


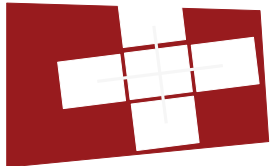
# Multimed execution



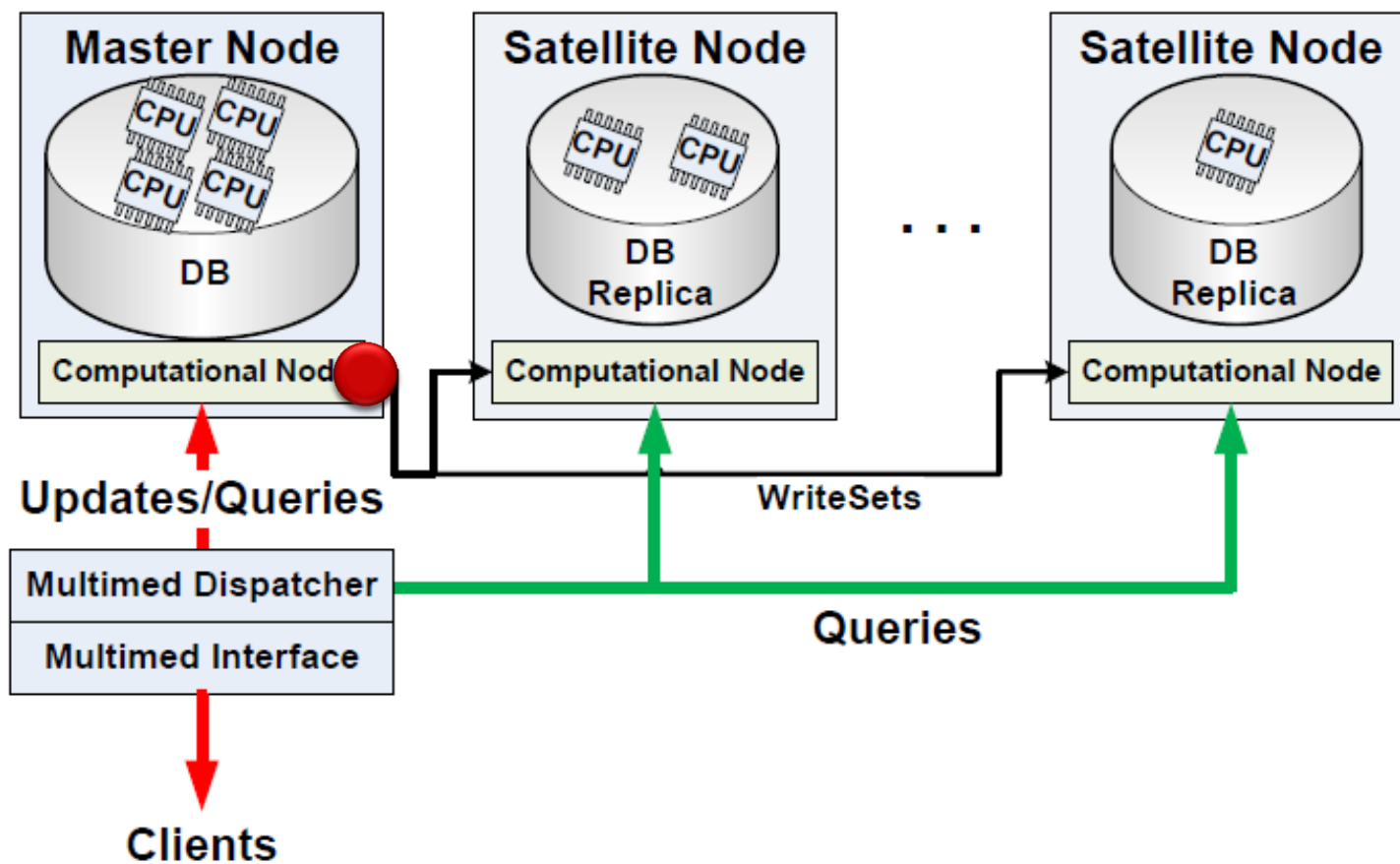


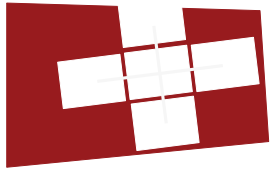
# Multimed execution



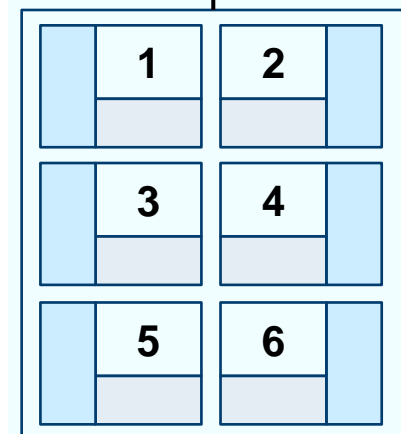
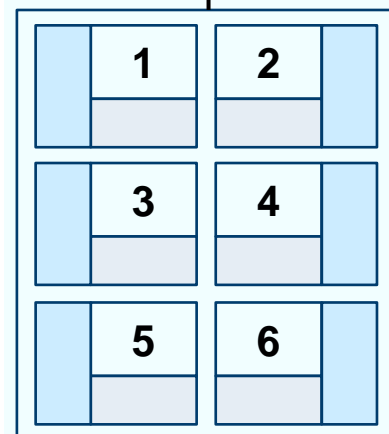
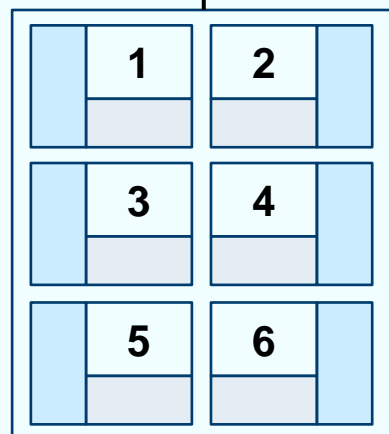
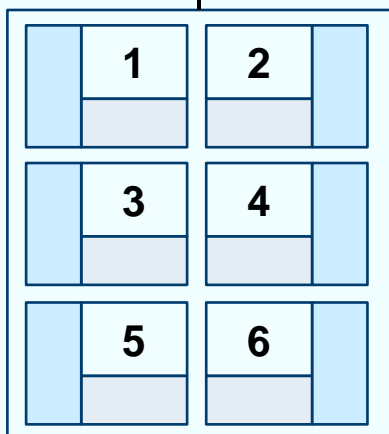
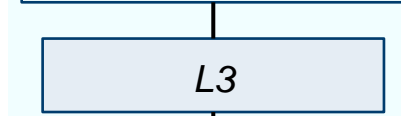
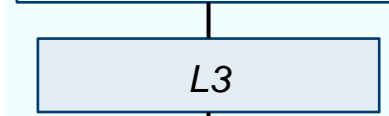
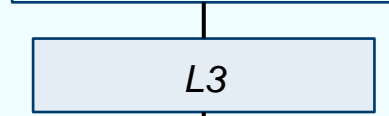
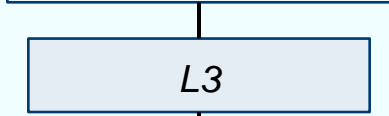
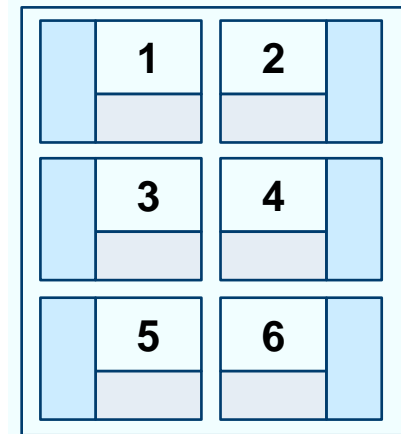
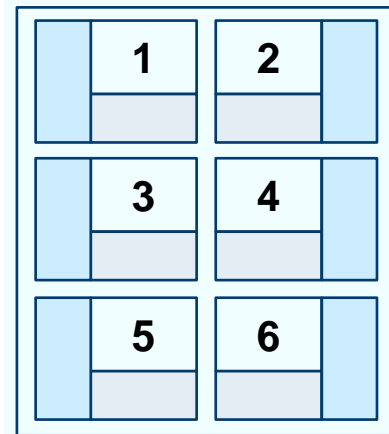
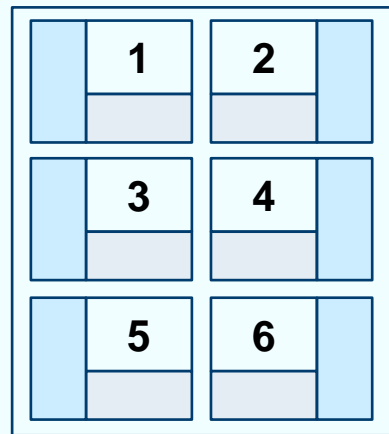
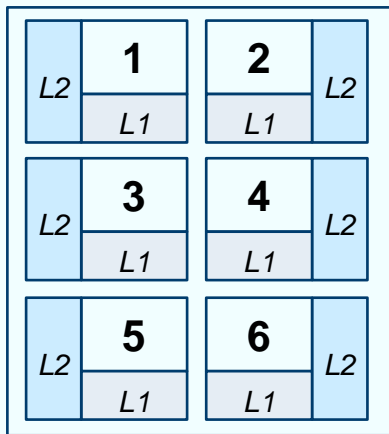


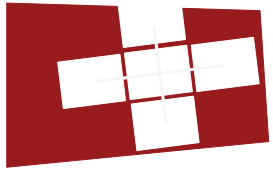
# Multimed execution



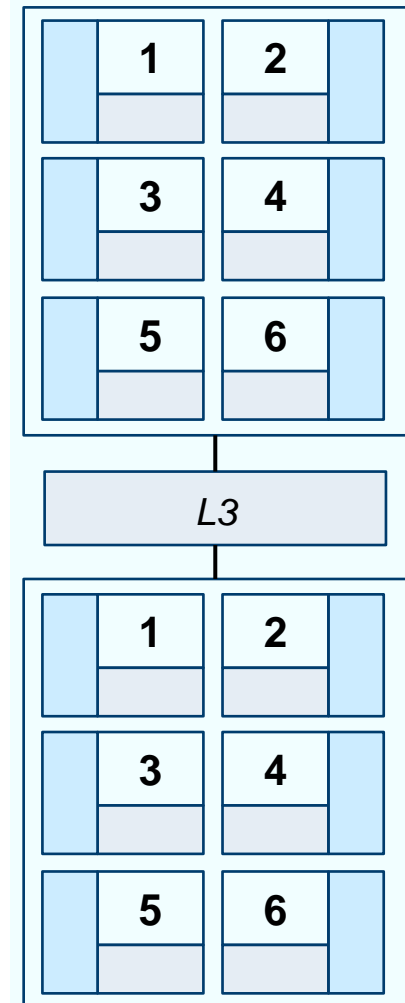
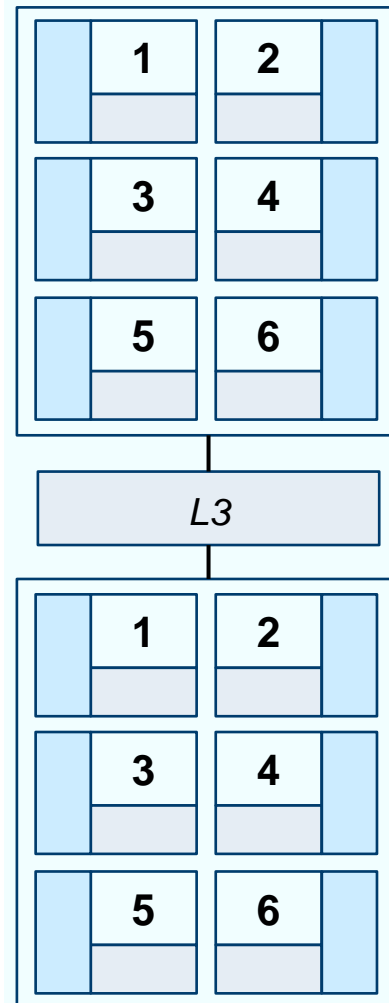
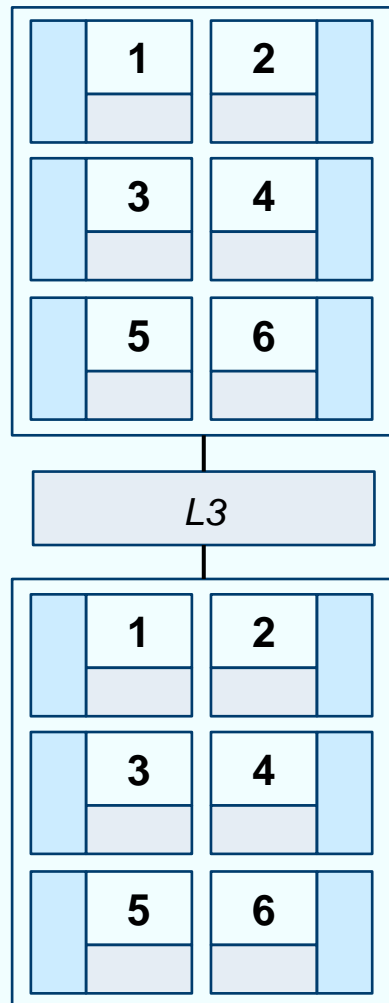
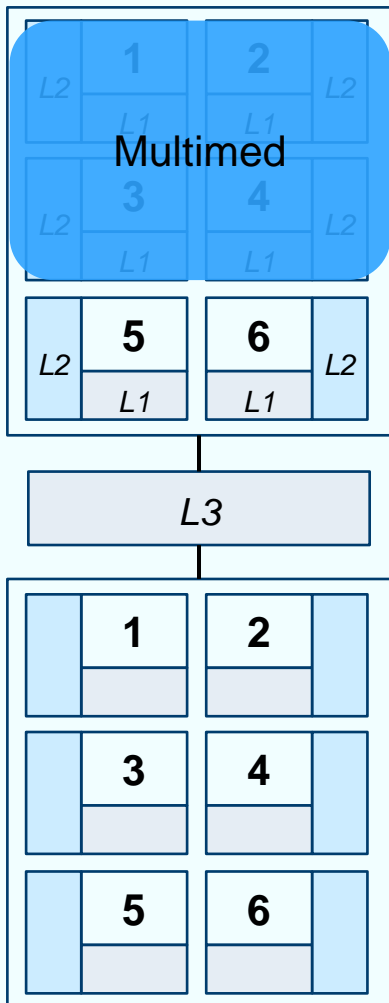


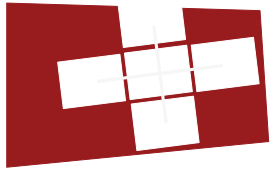
# System deployment, 48 core, 4 socket AMD Magny-Cours



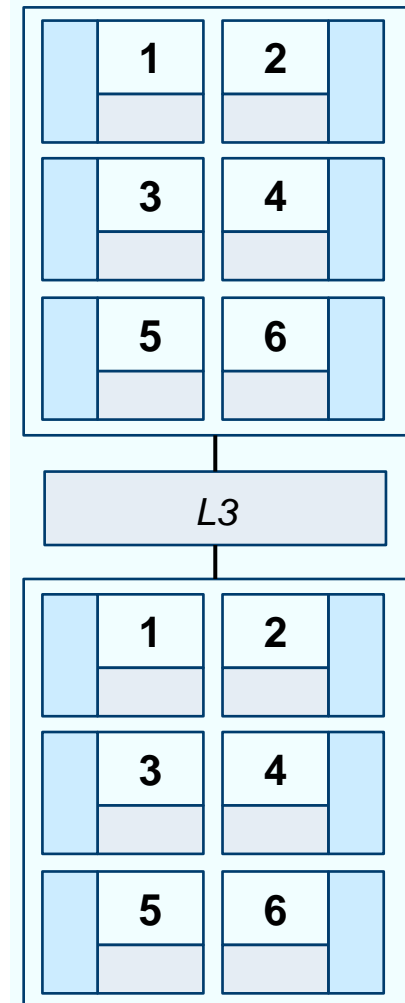
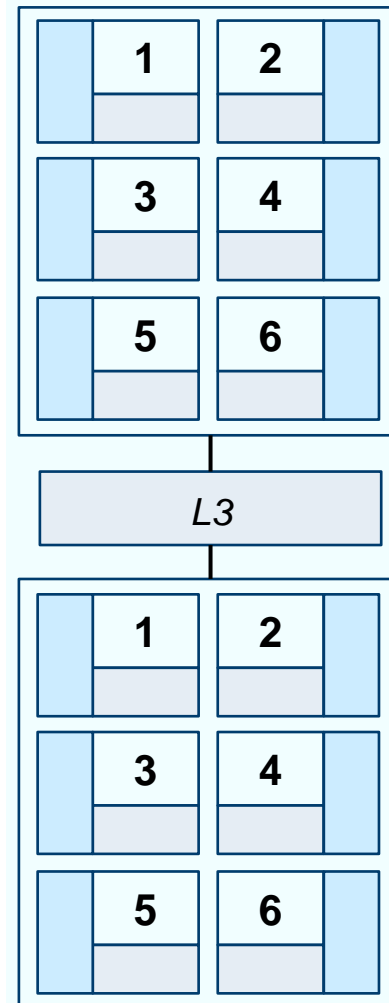
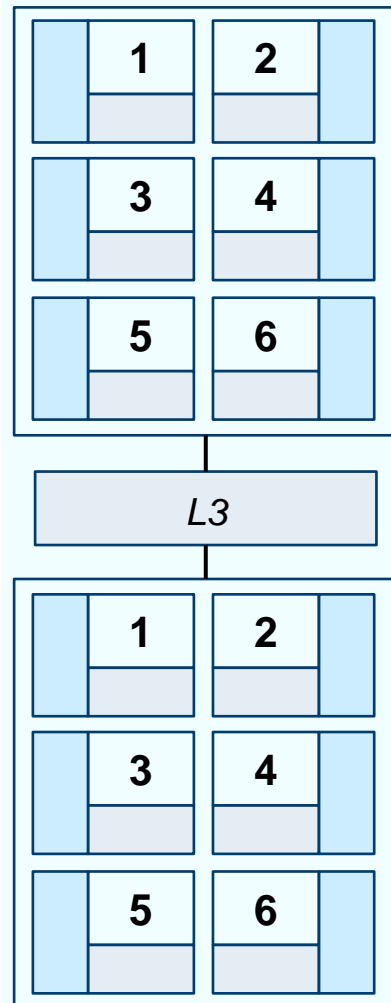
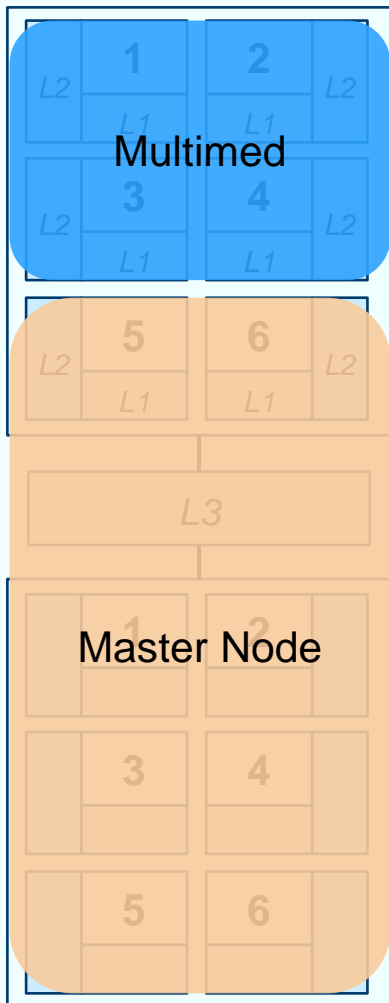


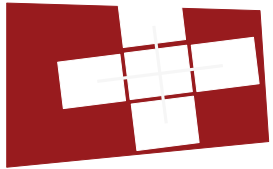
# System deployment, 48 core, 4 socket AMD Magny-Cours



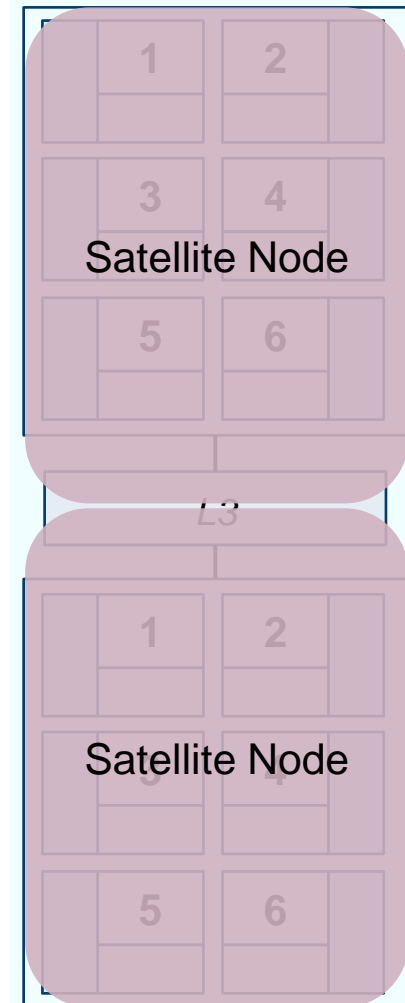
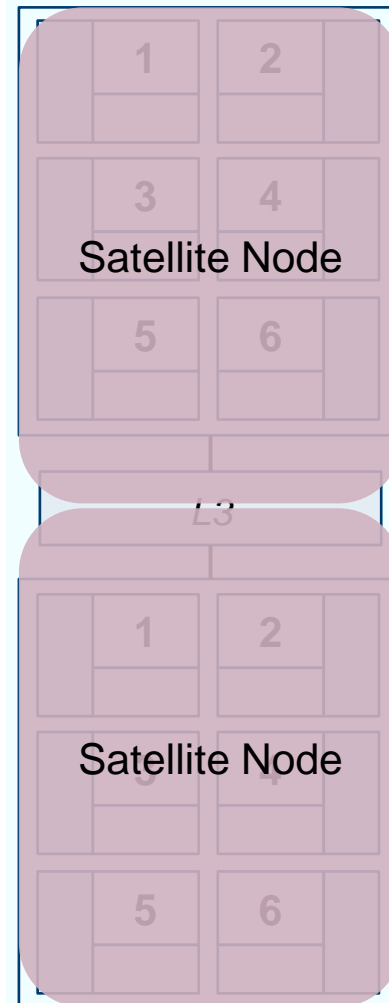
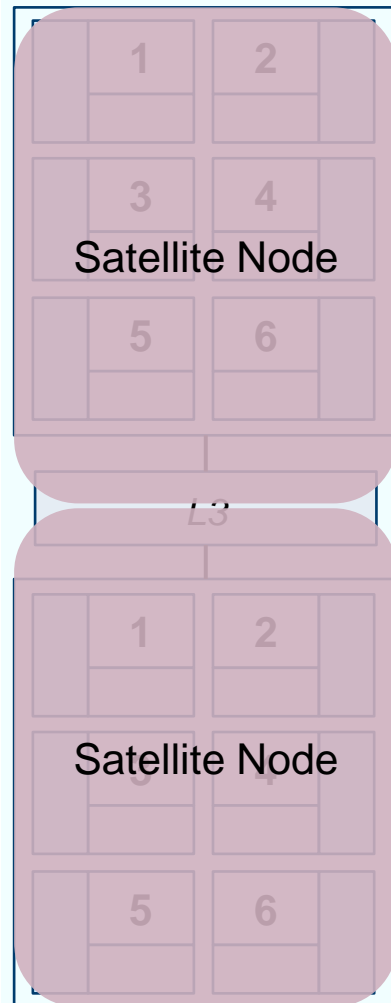
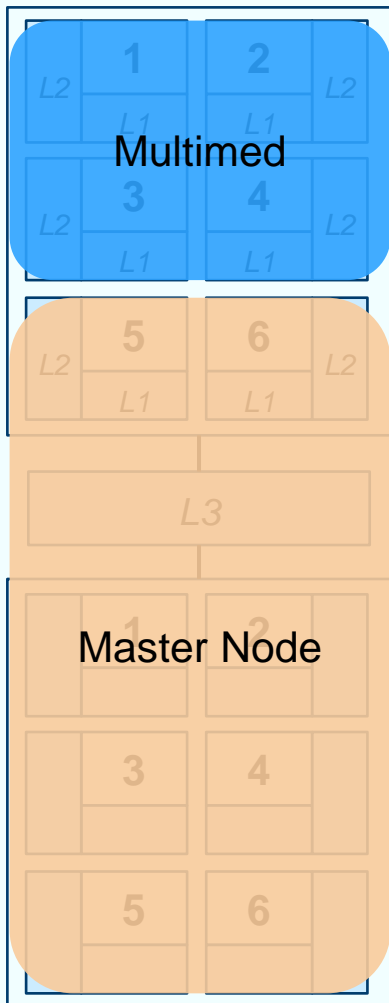


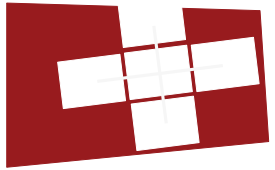
# System deployment, 48 core, 4 socket AMD Magny-Cours



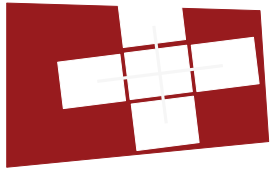


# System deployment, 48 core, 4 socket AMD Magny-Cours

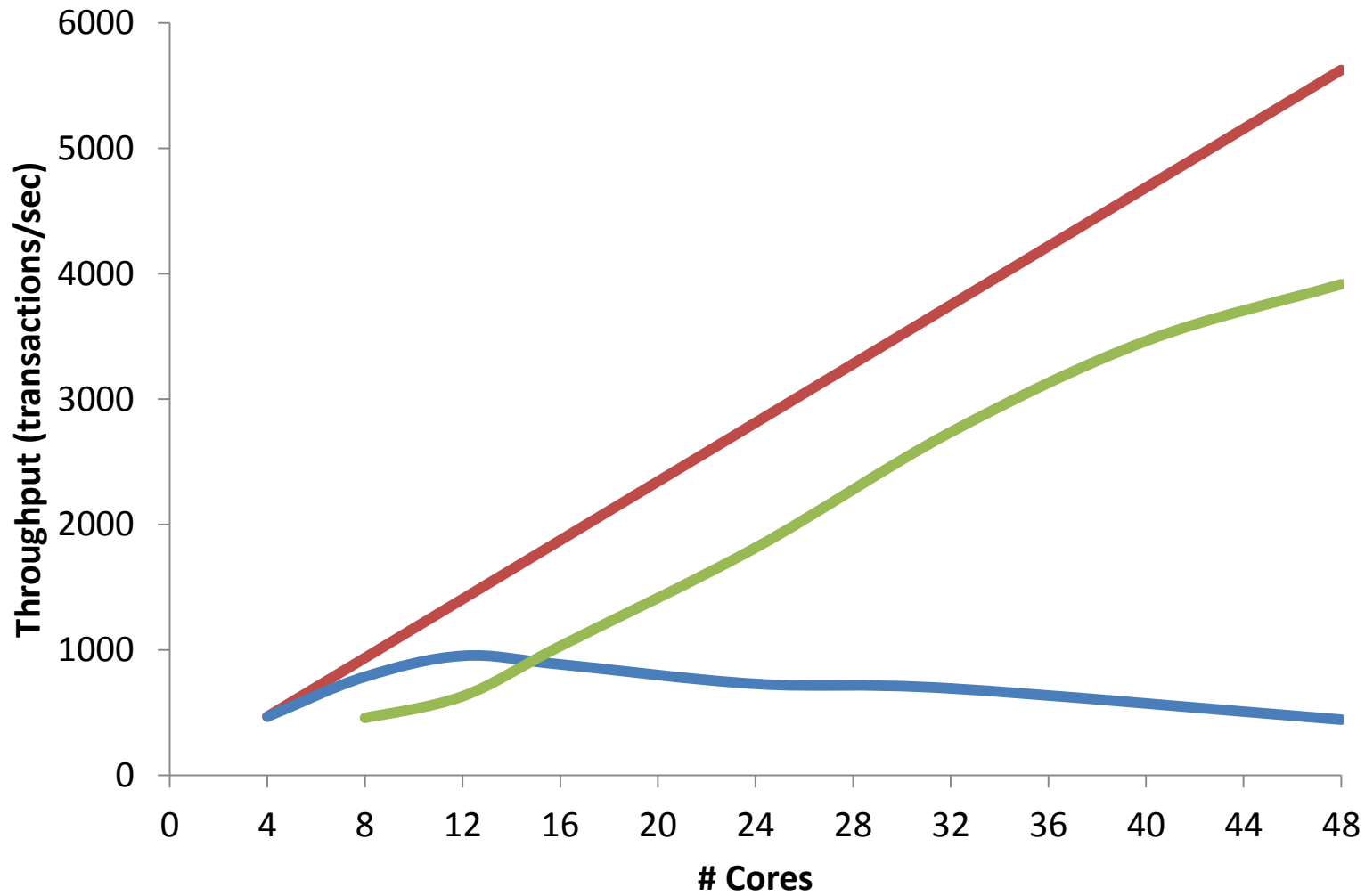


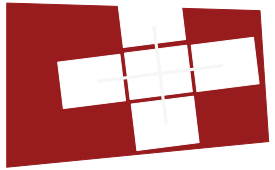


# Multimed's scalability

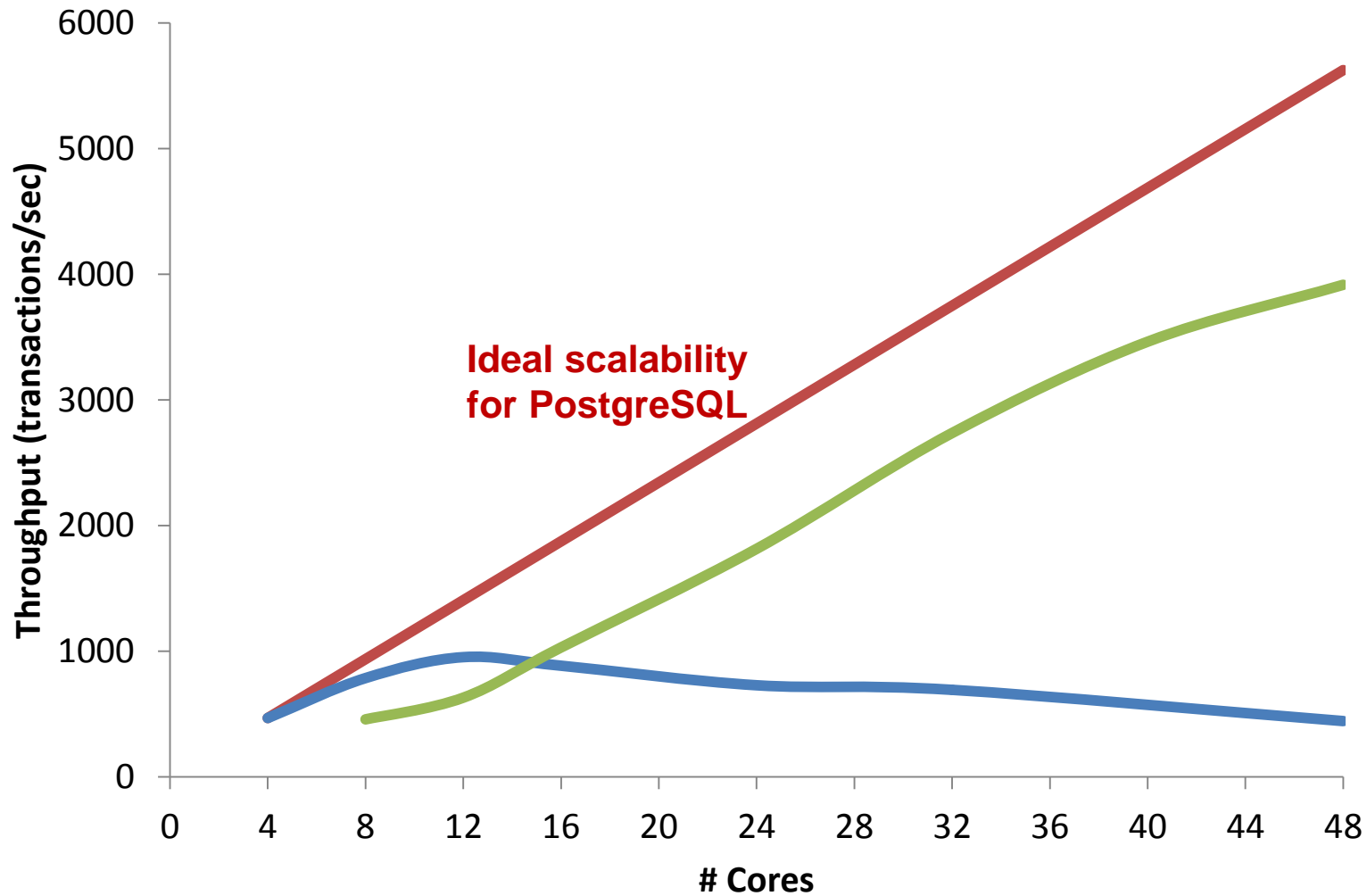


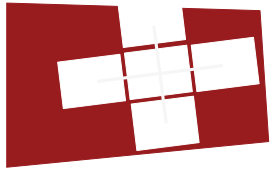
# Multimed's scalability



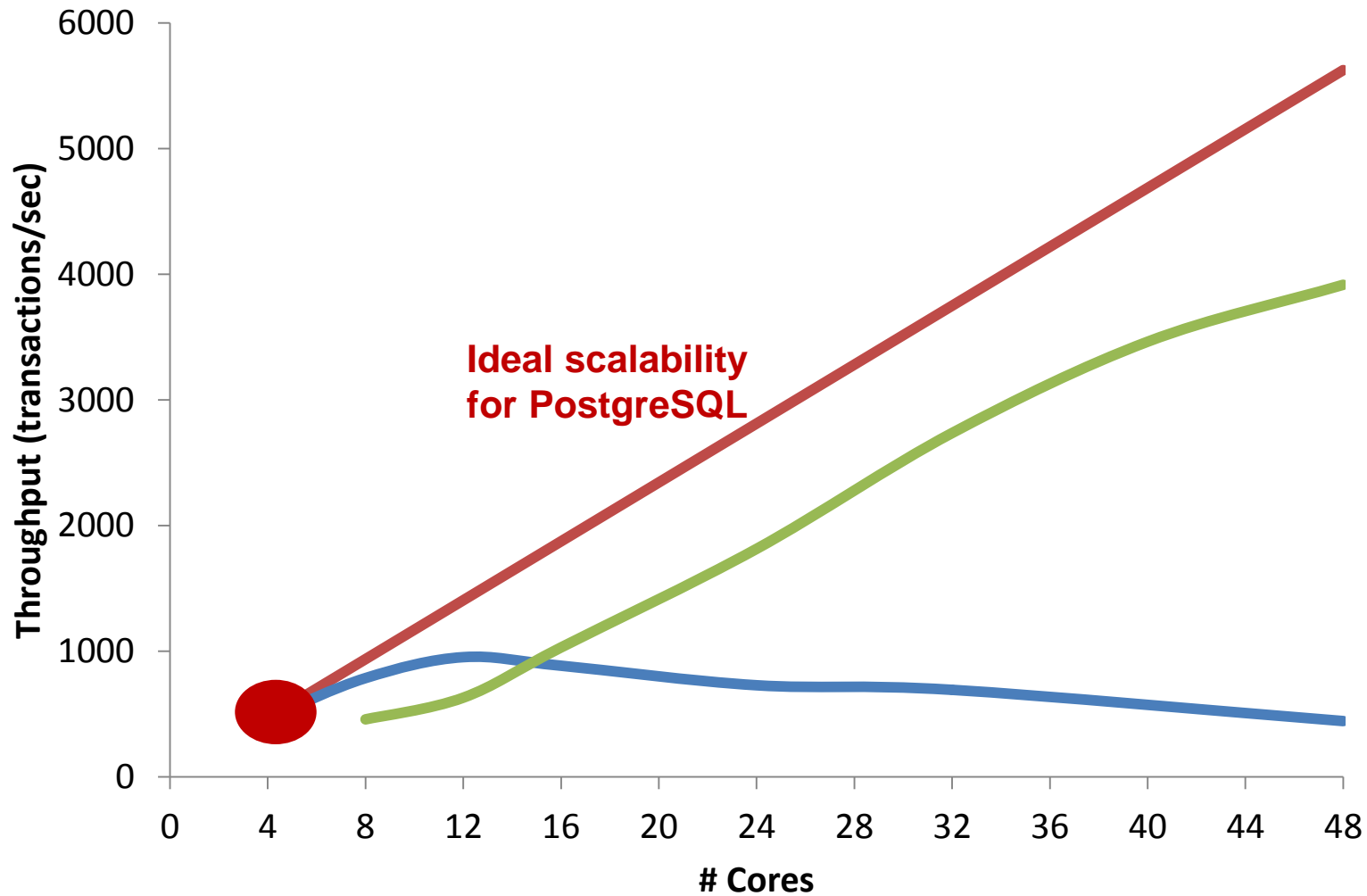


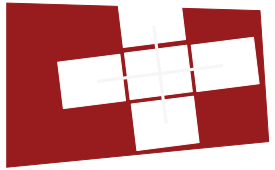
# Multimed's scalability



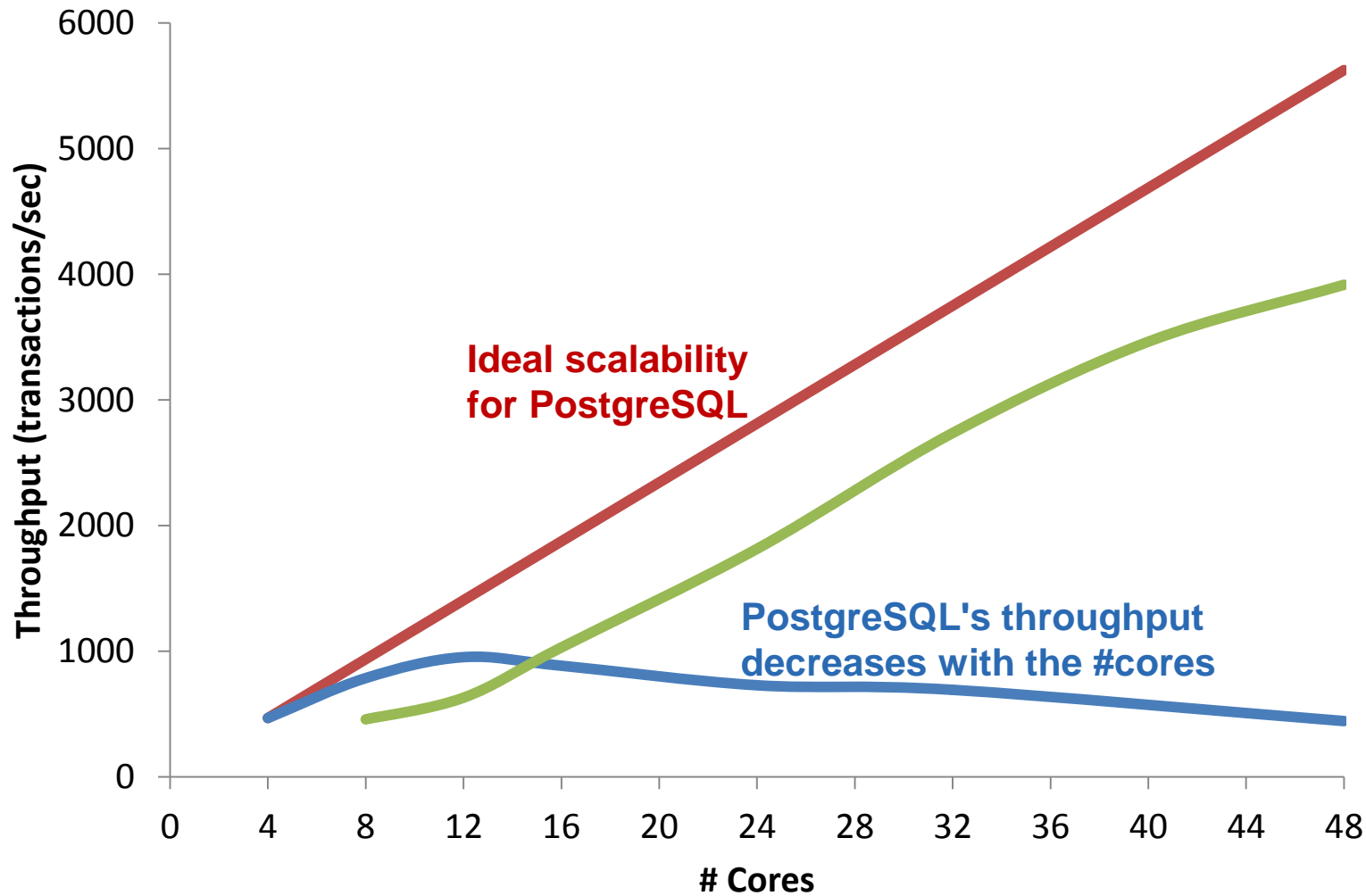


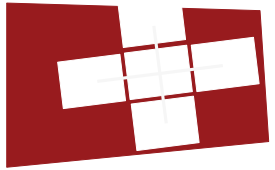
# Multimed's scalability



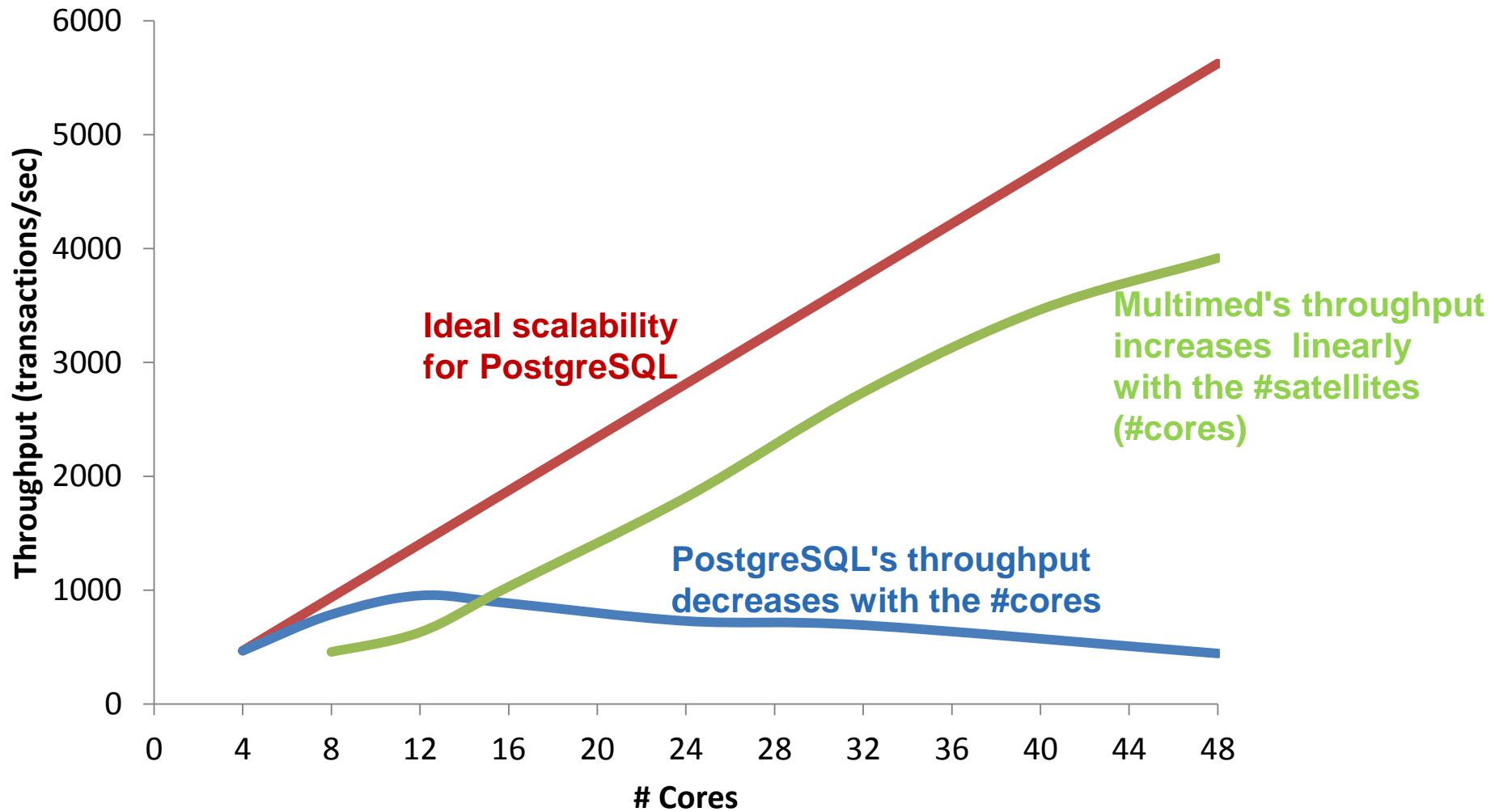


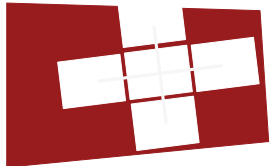
# Multimed's scalability



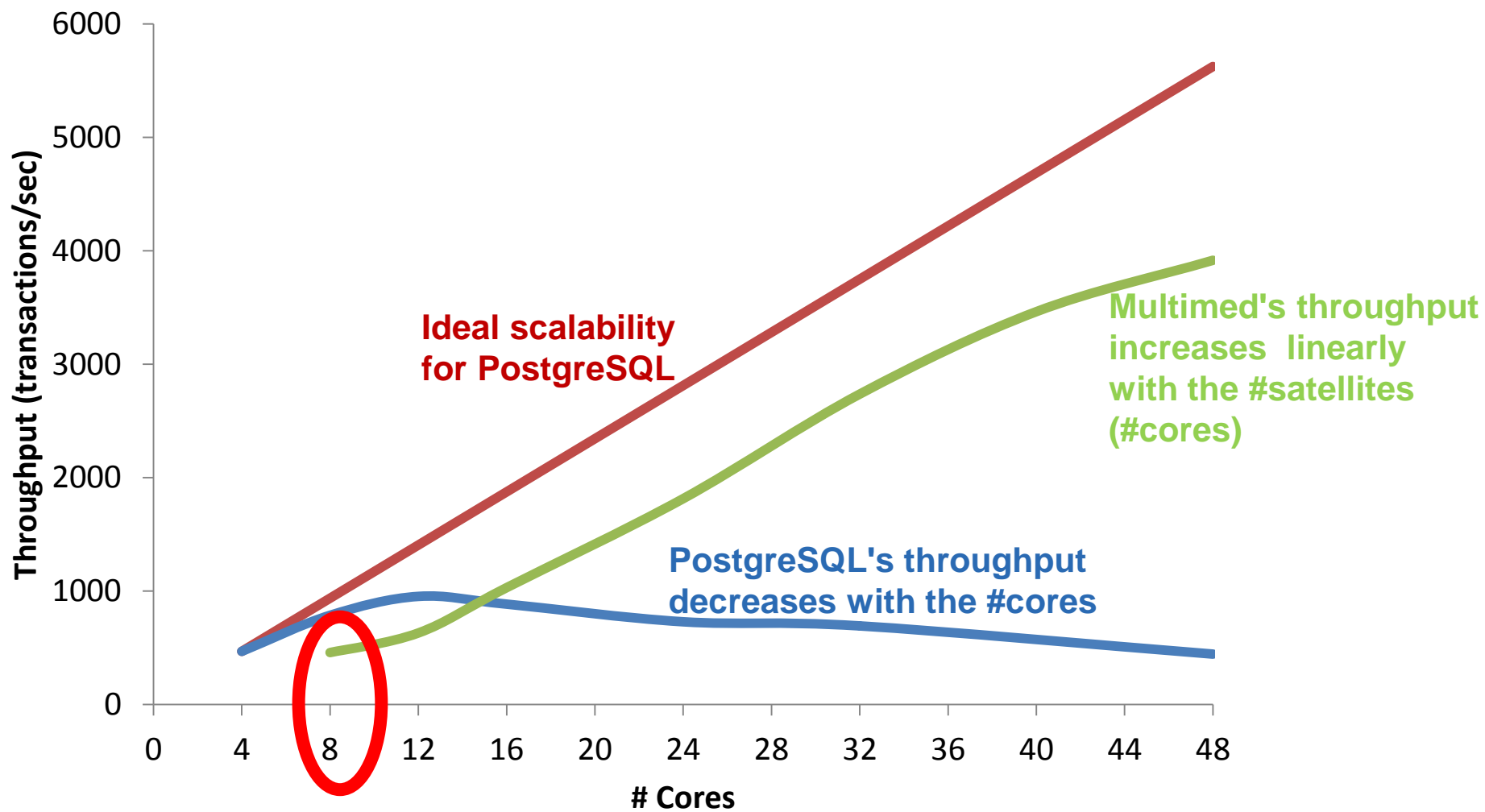


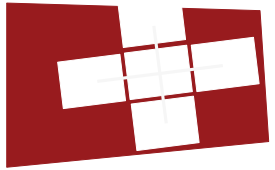
# Multimed's scalability



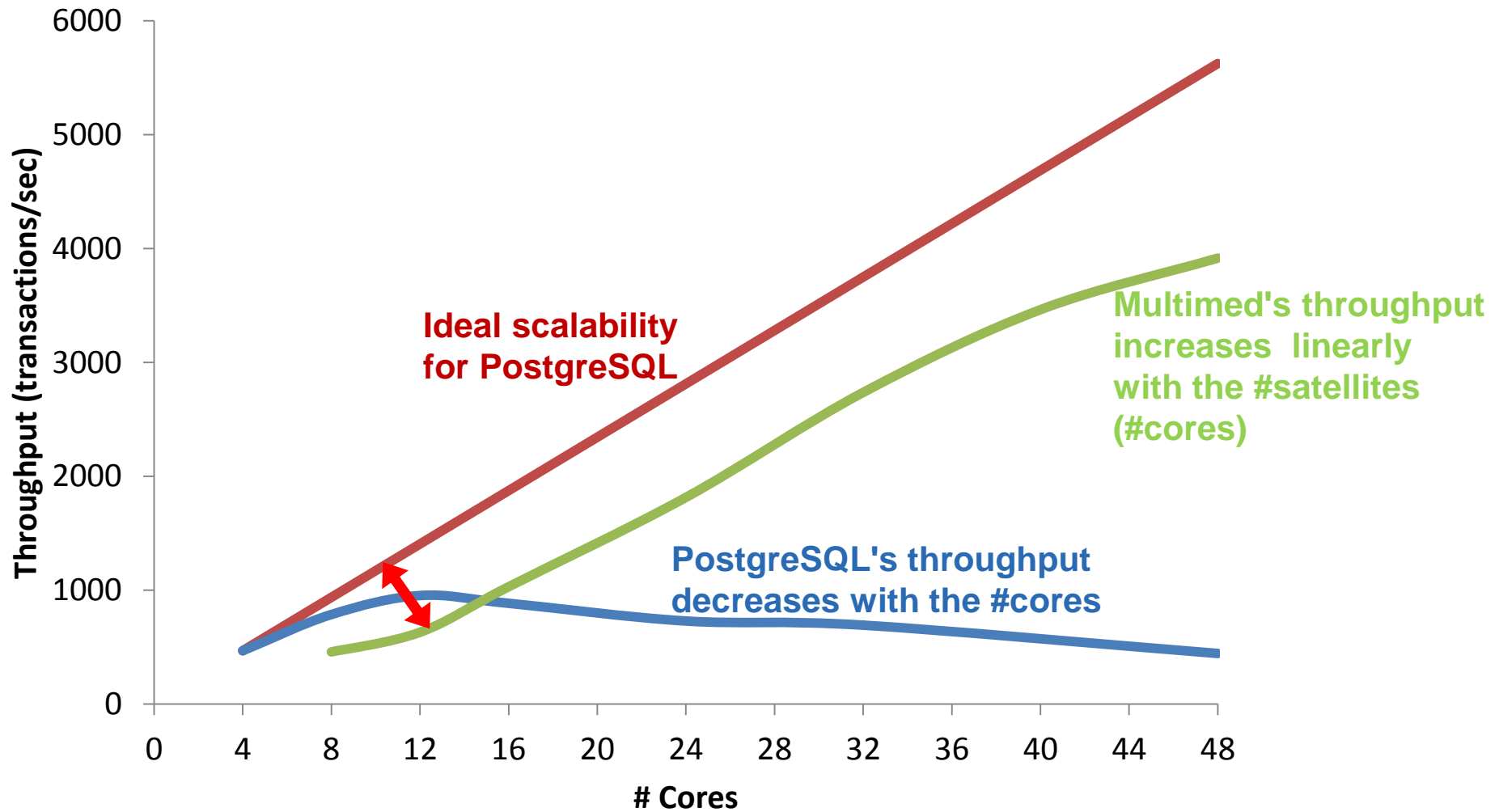


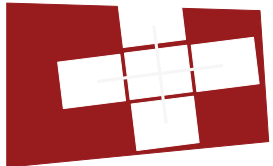
# Multimed's scalability



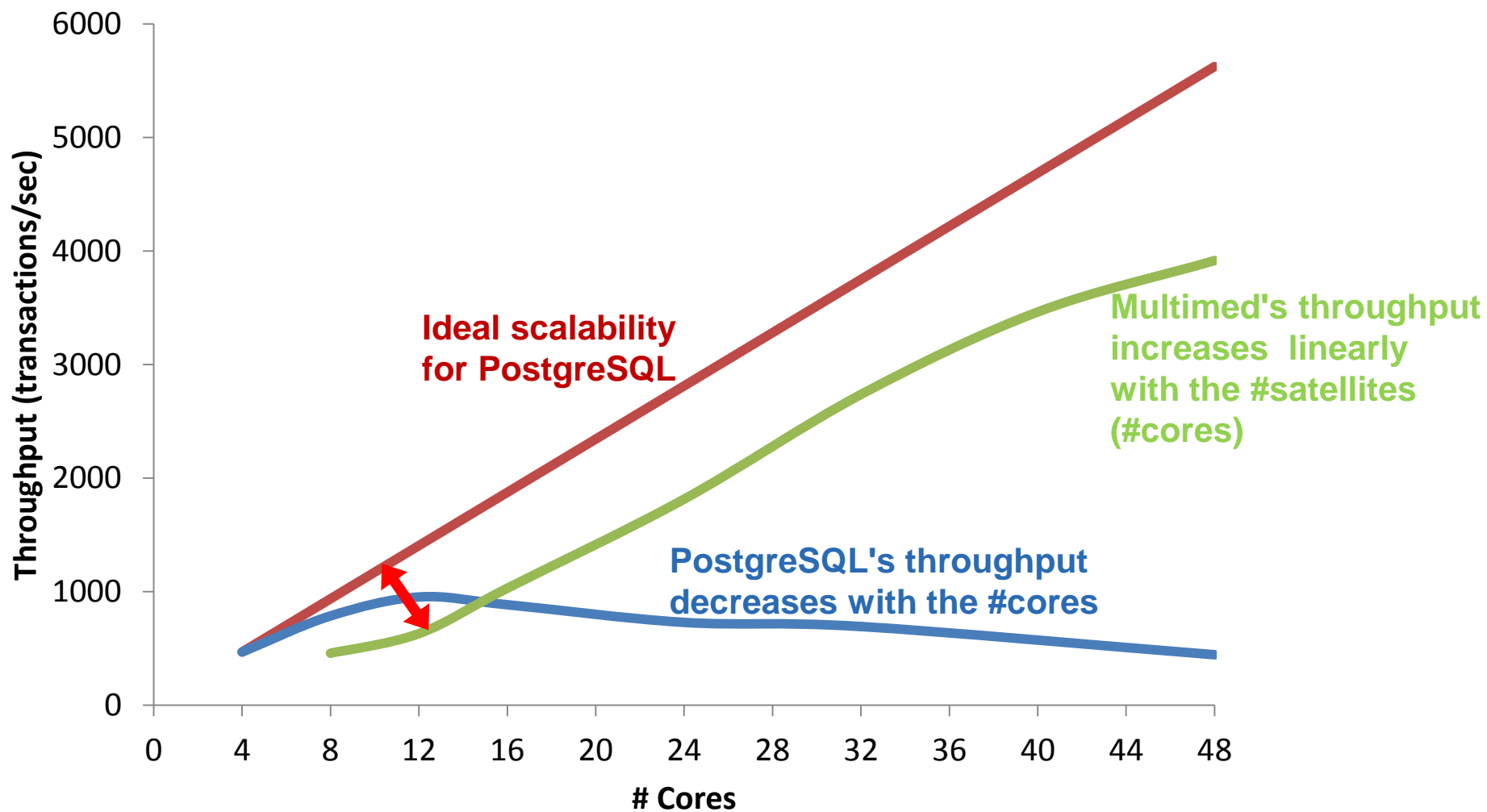


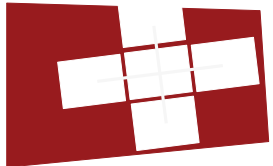
# Multimed's scalability



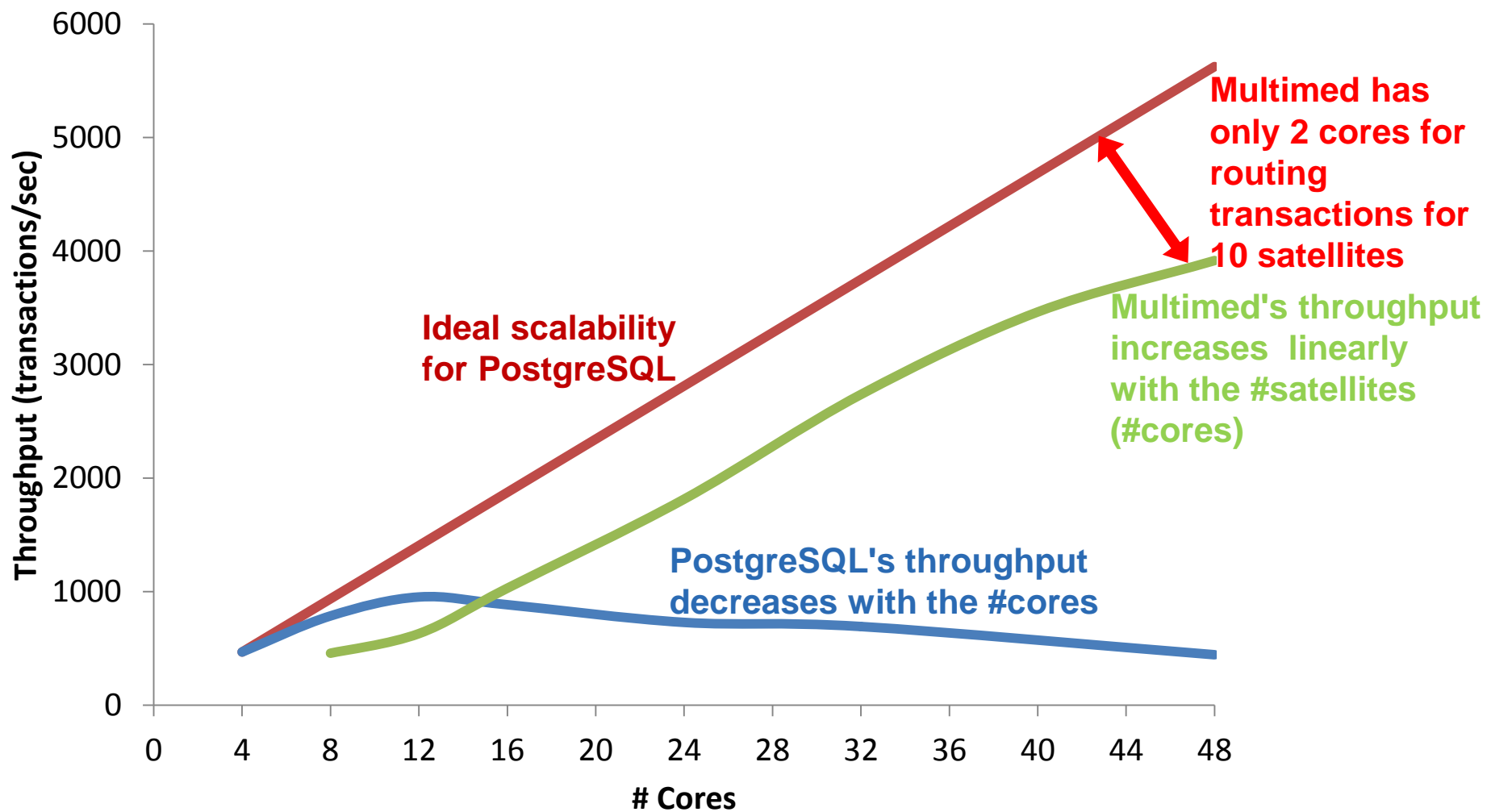


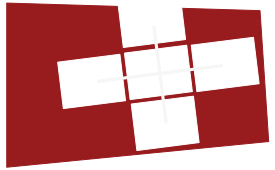
# Multimed's scalability



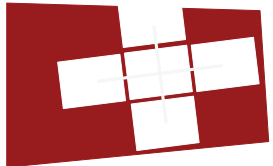


# Multimed's scalability



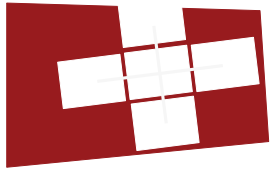


# Why does Multimed work?

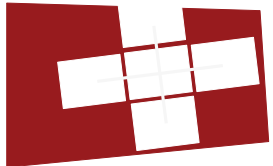


## Why does Multimed work?

- *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

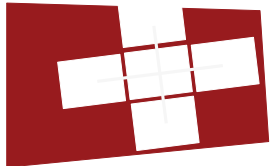


# Databases and workloads



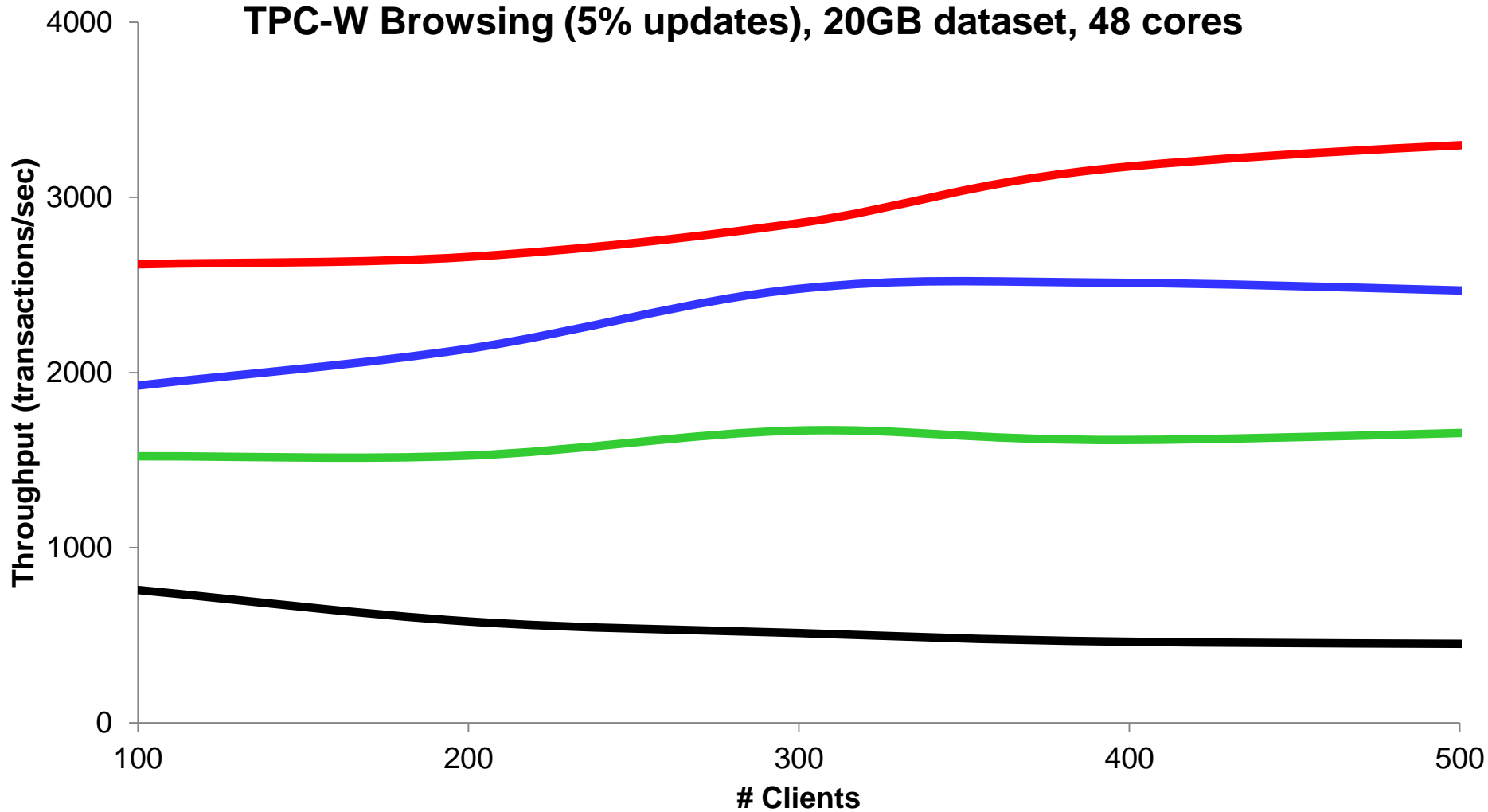
# Databases and workloads

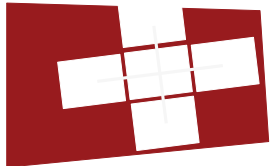
- Target workloads
  - Read-heavy workloads with updates (TPC-W)
    - TPC-W Browsing (5% Updates)
    - TPC-W Shopping (20% Updates)
  - Mainly main memory resident datasets



# Experimental results, PostgreSQL

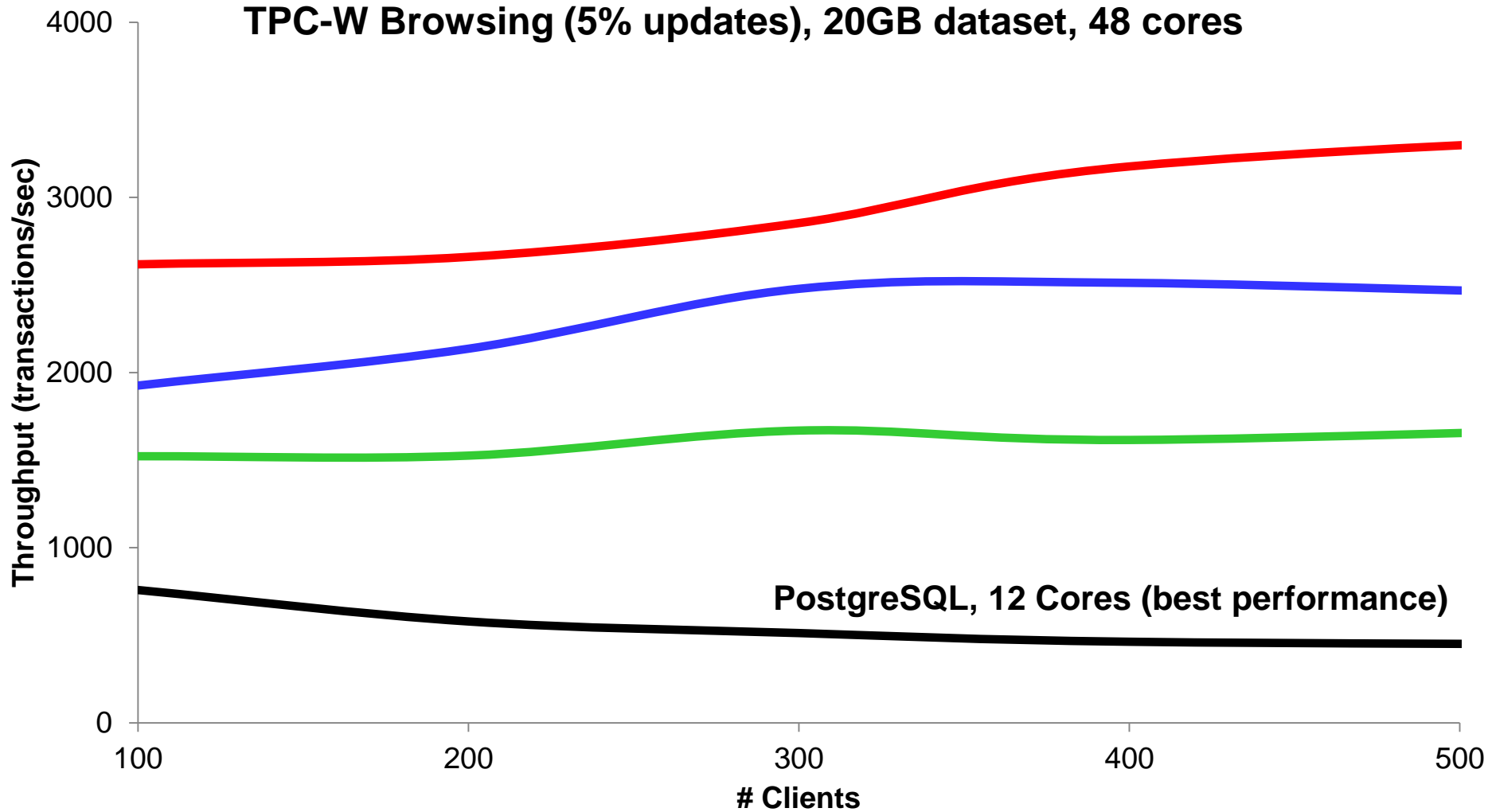
TPC-W Browsing (5% updates), 20GB dataset, 48 cores

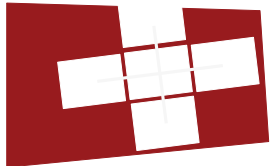




# Experimental results, PostgreSQL

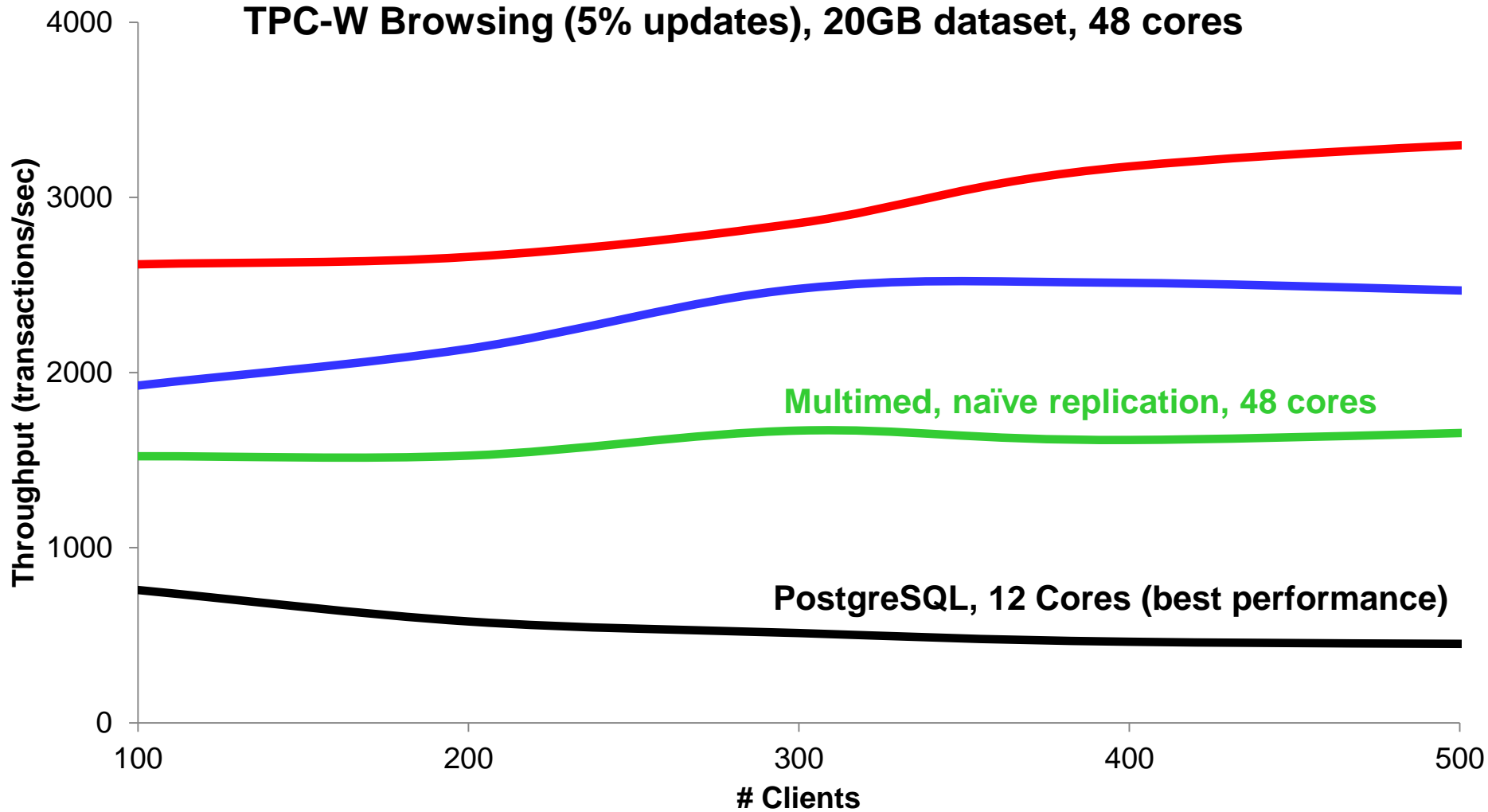
**TPC-W Browsing (5% updates), 20GB dataset, 48 cores**

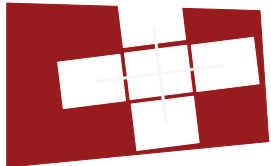




# Experimental results, PostgreSQL

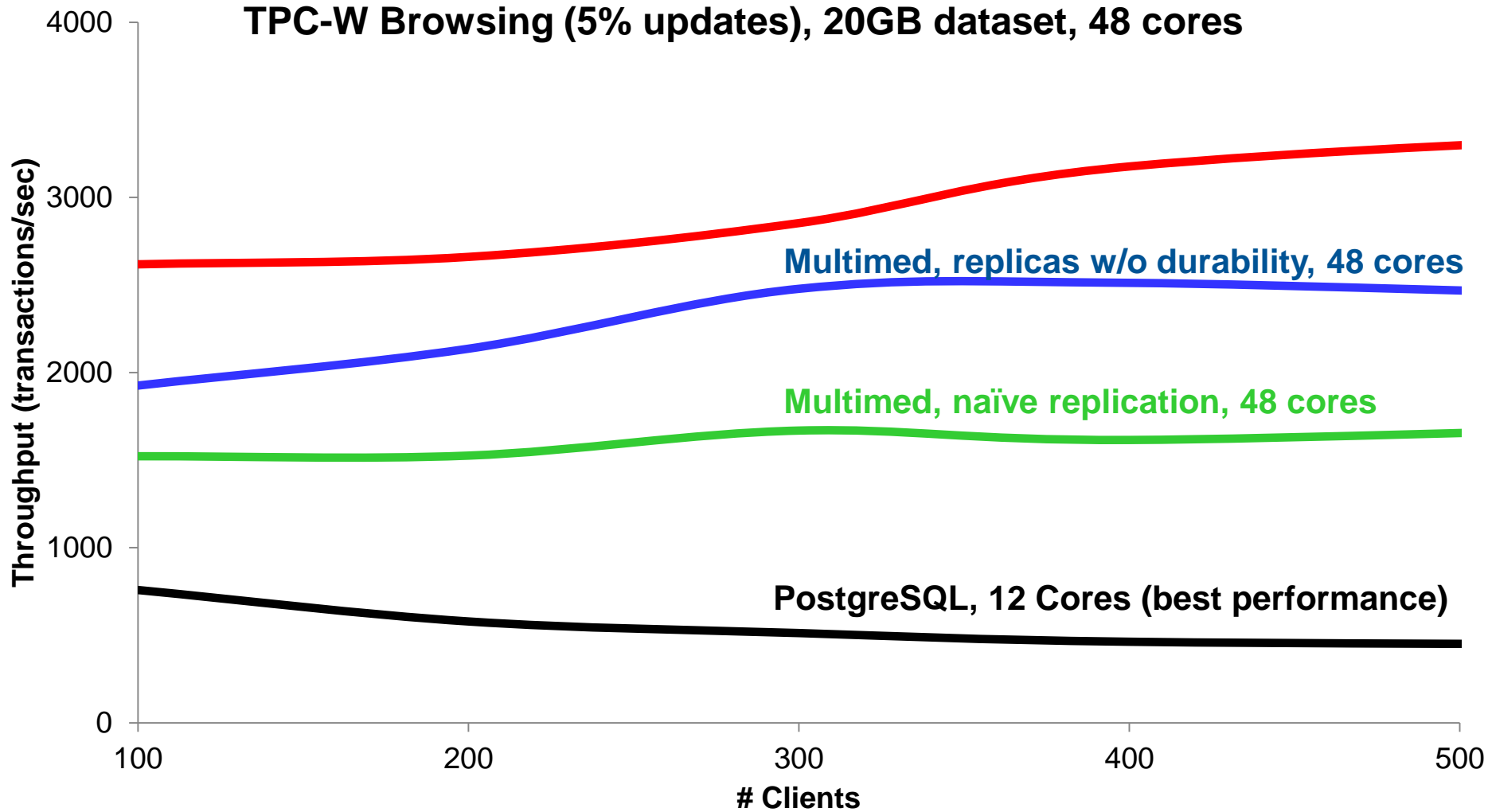
TPC-W Browsing (5% updates), 20GB dataset, 48 cores

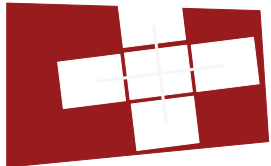




# Experimental results, PostgreSQL

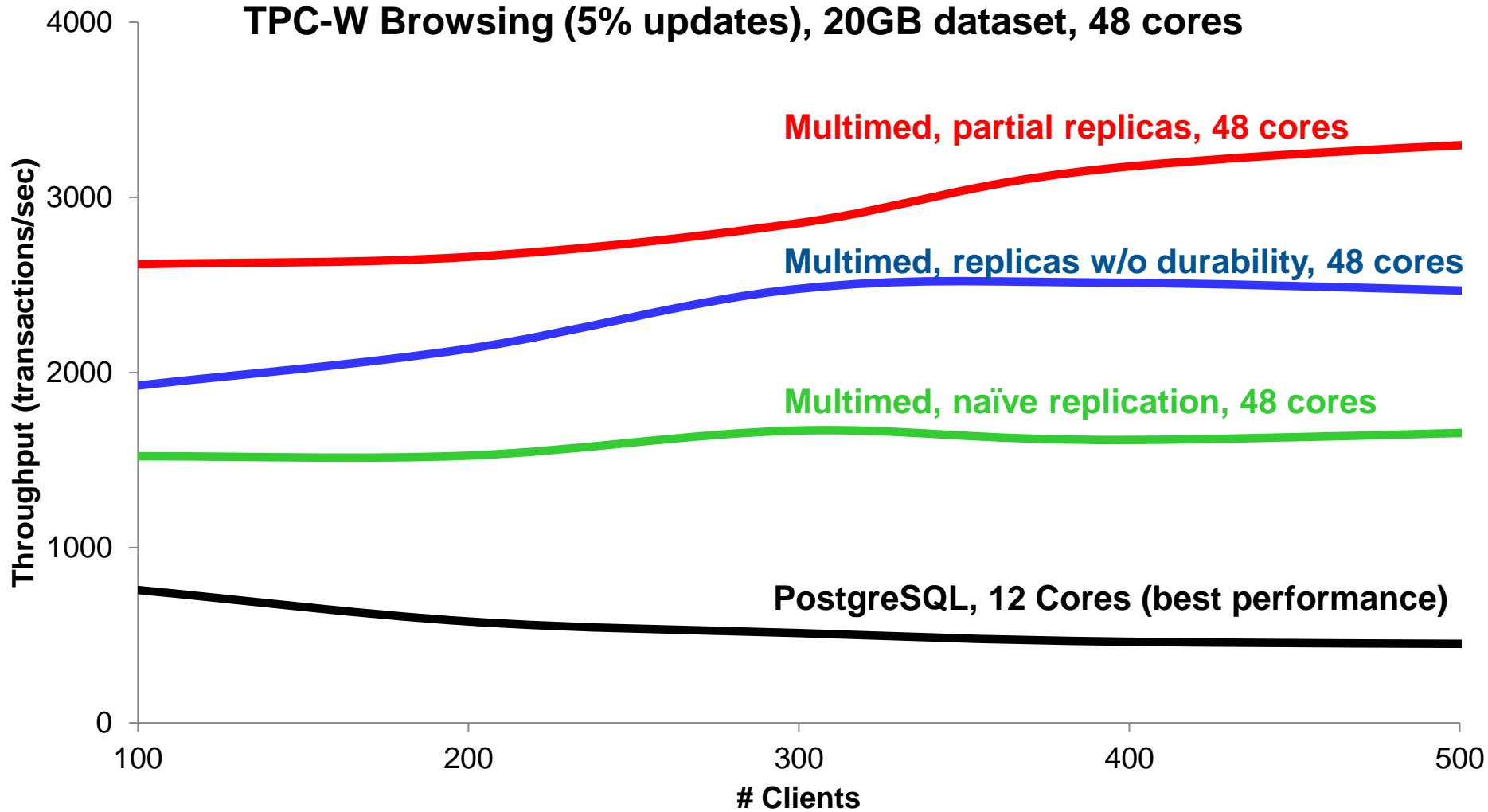
TPC-W Browsing (5% updates), 20GB dataset, 48 cores

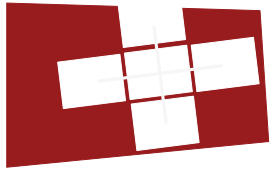




# Experimental results, PostgreSQL

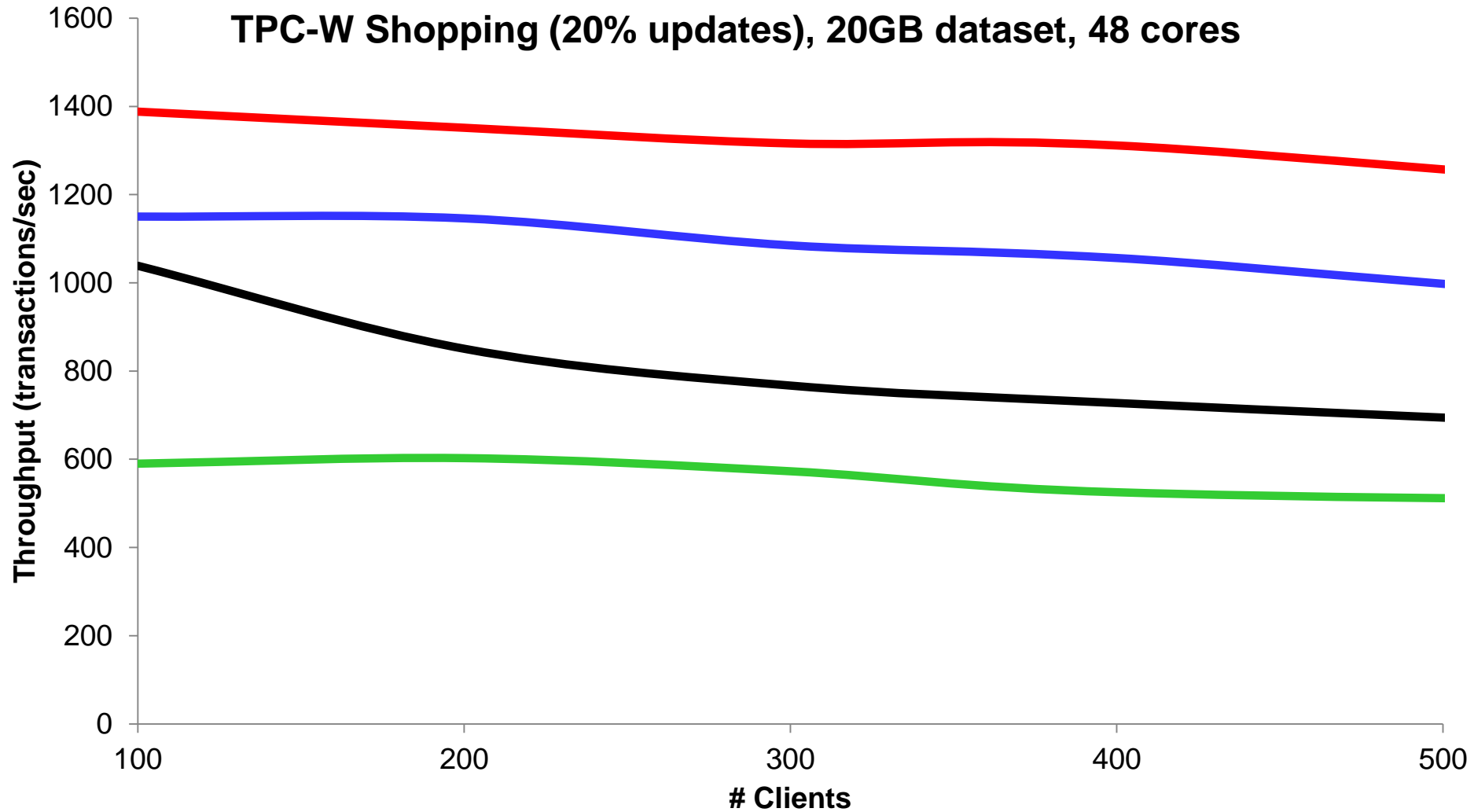
TPC-W Browsing (5% updates), 20GB dataset, 48 cores

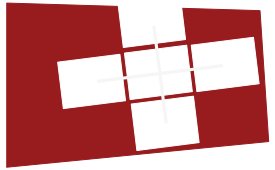




# Experimental results, PostgreSQL

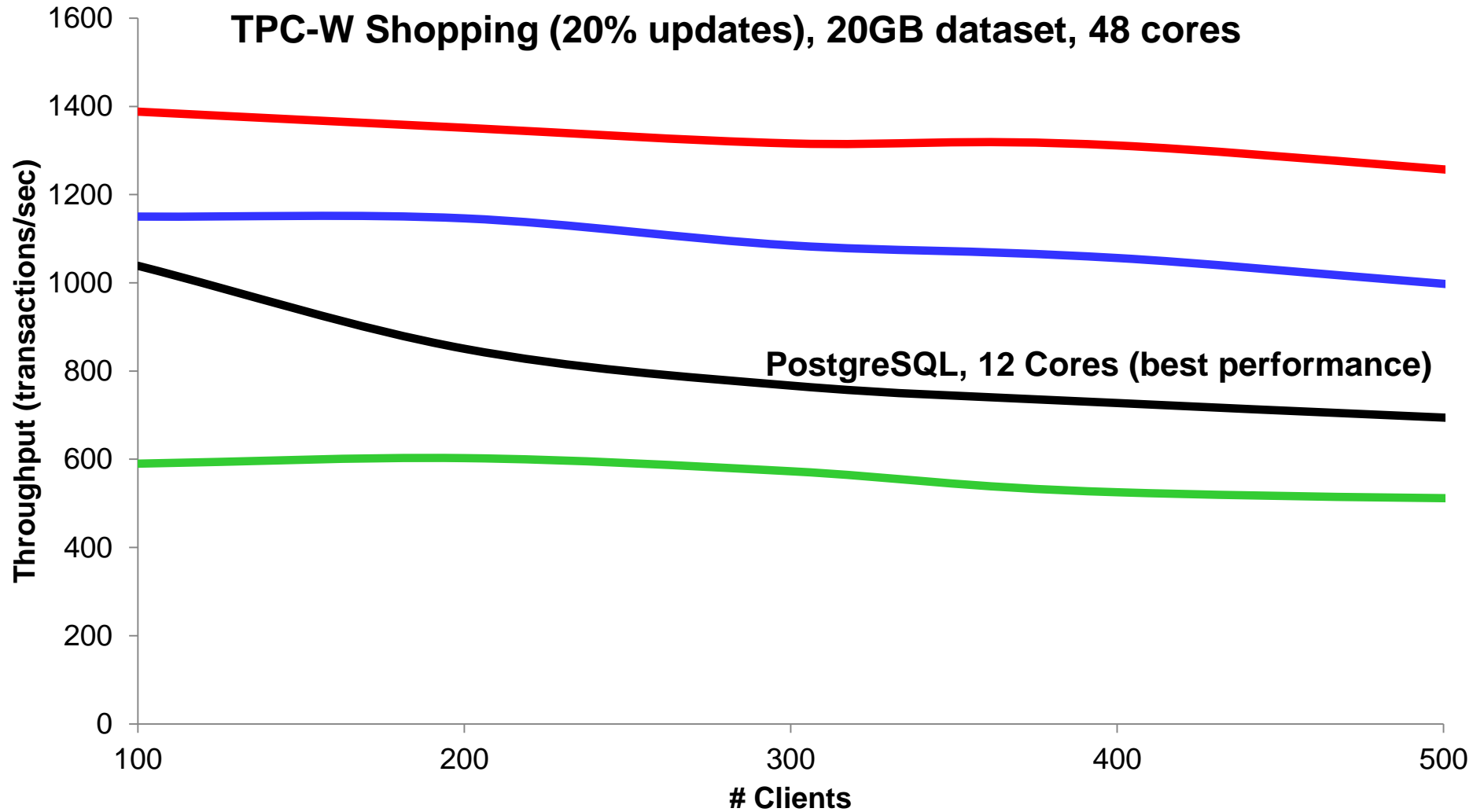
TPC-W Shopping (20% updates), 20GB dataset, 48 cores

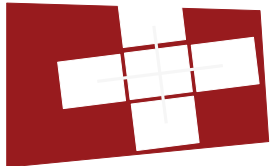




# Experimental results, PostgreSQL

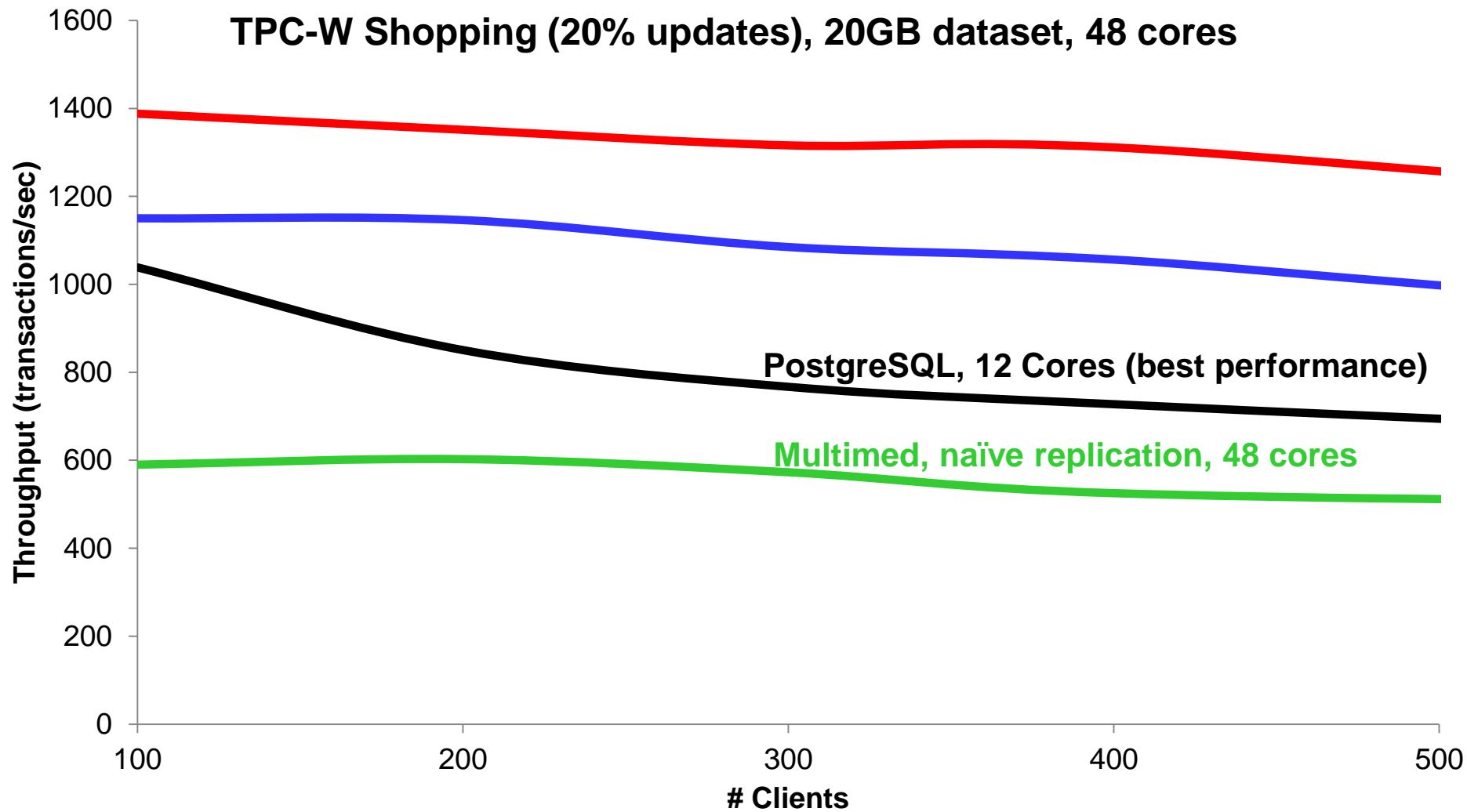
**TPC-W Shopping (20% updates), 20GB dataset, 48 cores**

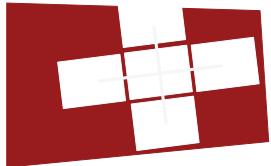




# Experimental results, PostgreSQL

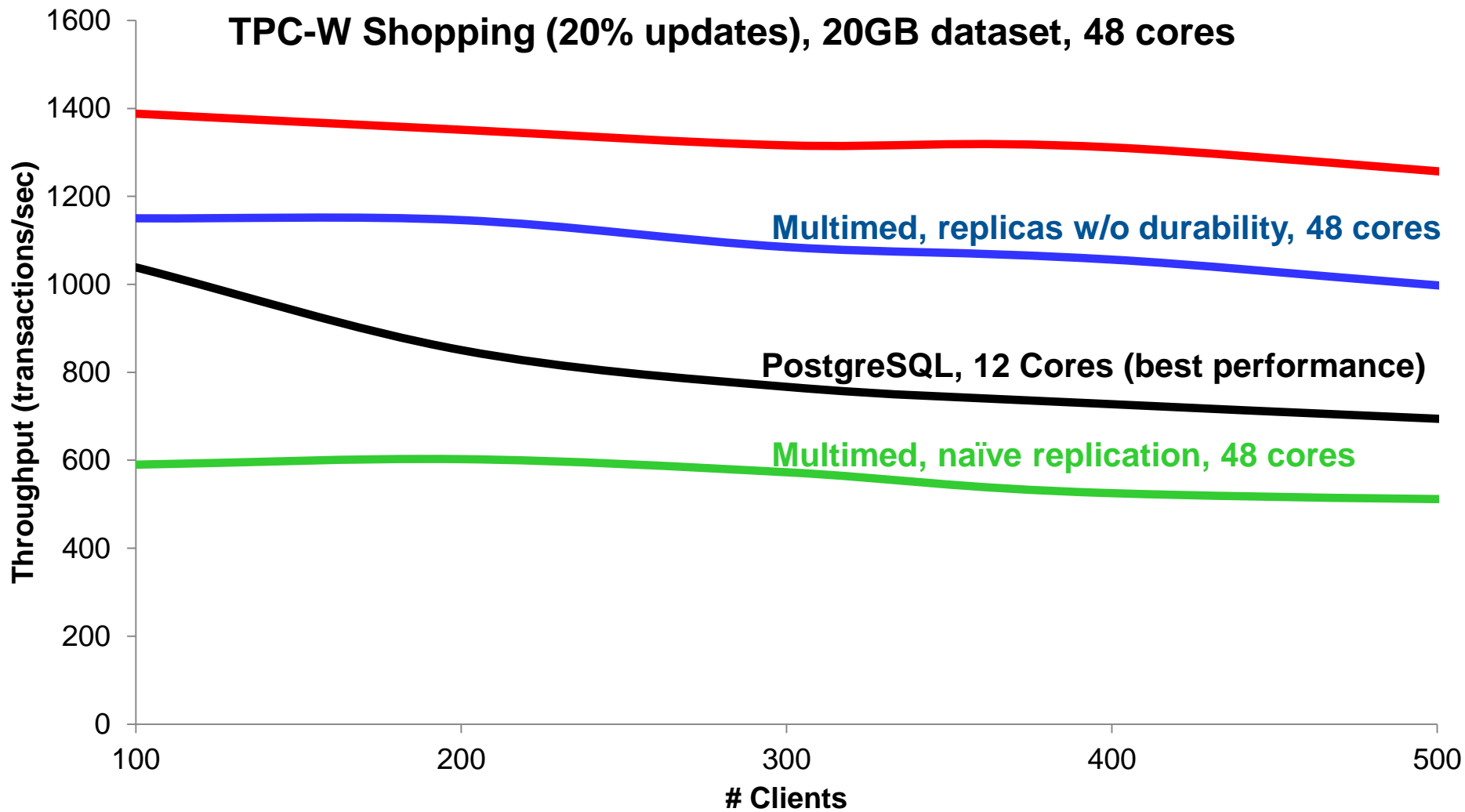
**TPC-W Shopping (20% updates), 20GB dataset, 48 cores**

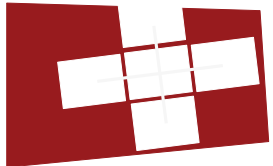




# Experimental results, PostgreSQL

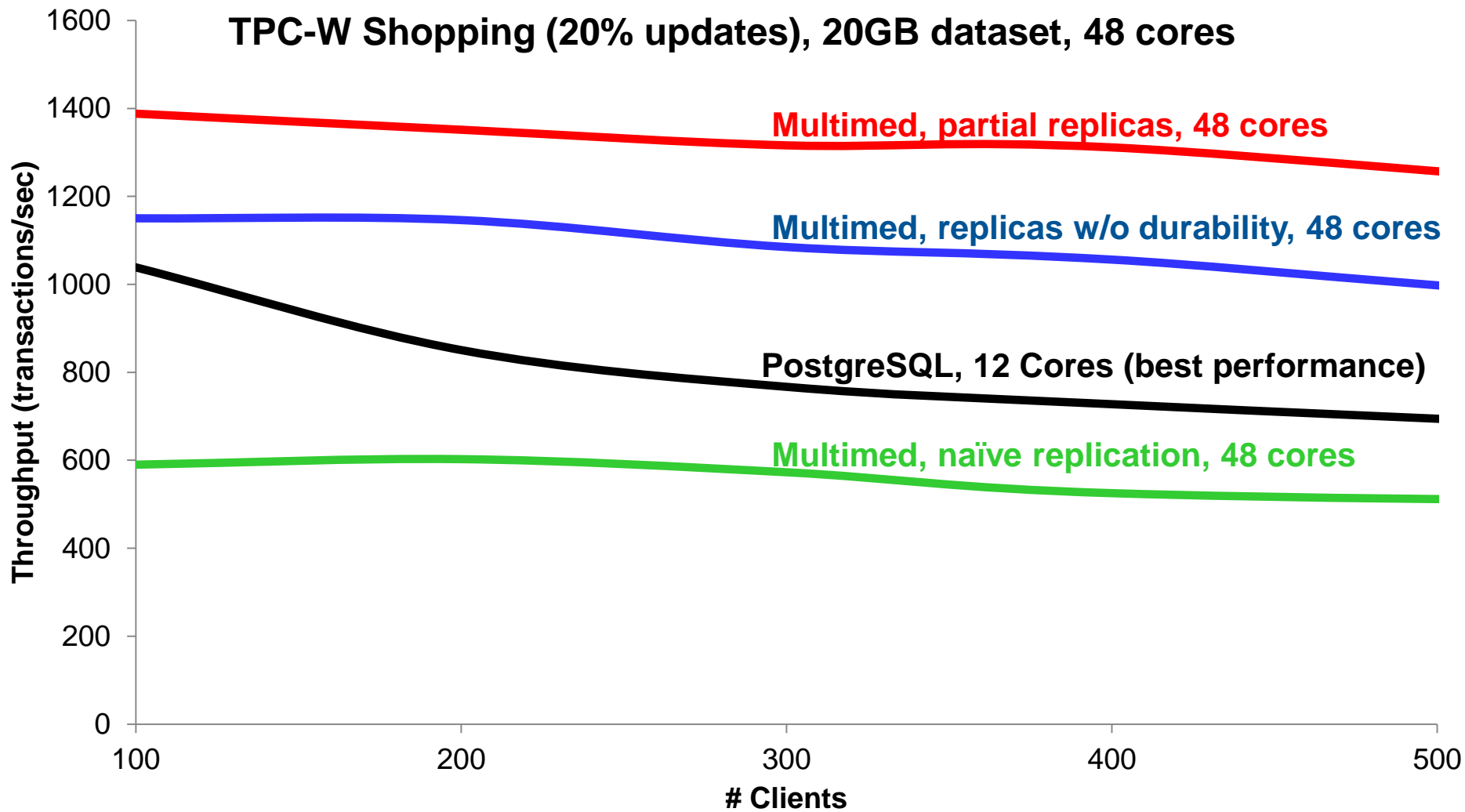
**TPC-W Shopping (20% updates), 20GB dataset, 48 cores**

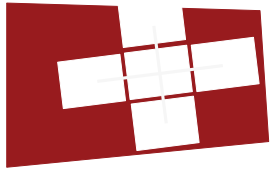




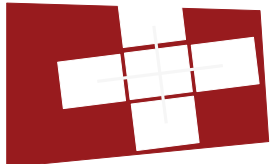
# Experimental results, PostgreSQL

TPC-W Shopping (20% updates), 20GB dataset, 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@inf.ethz.ch](mailto:tsalomie@inf.ethz.ch)