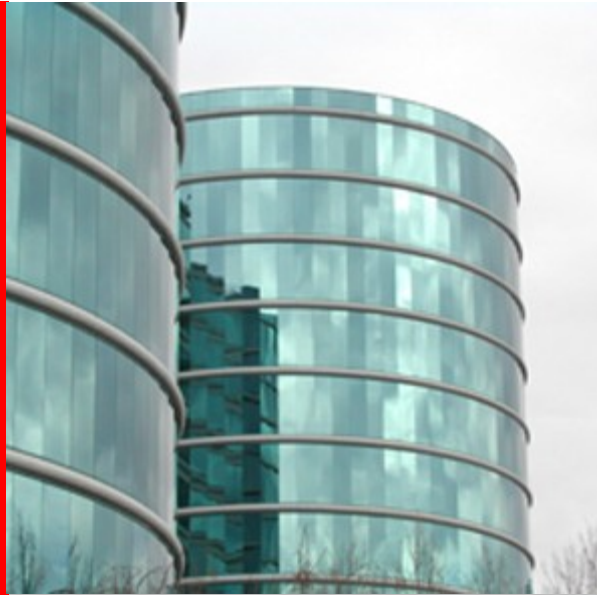


**ORACLE®**



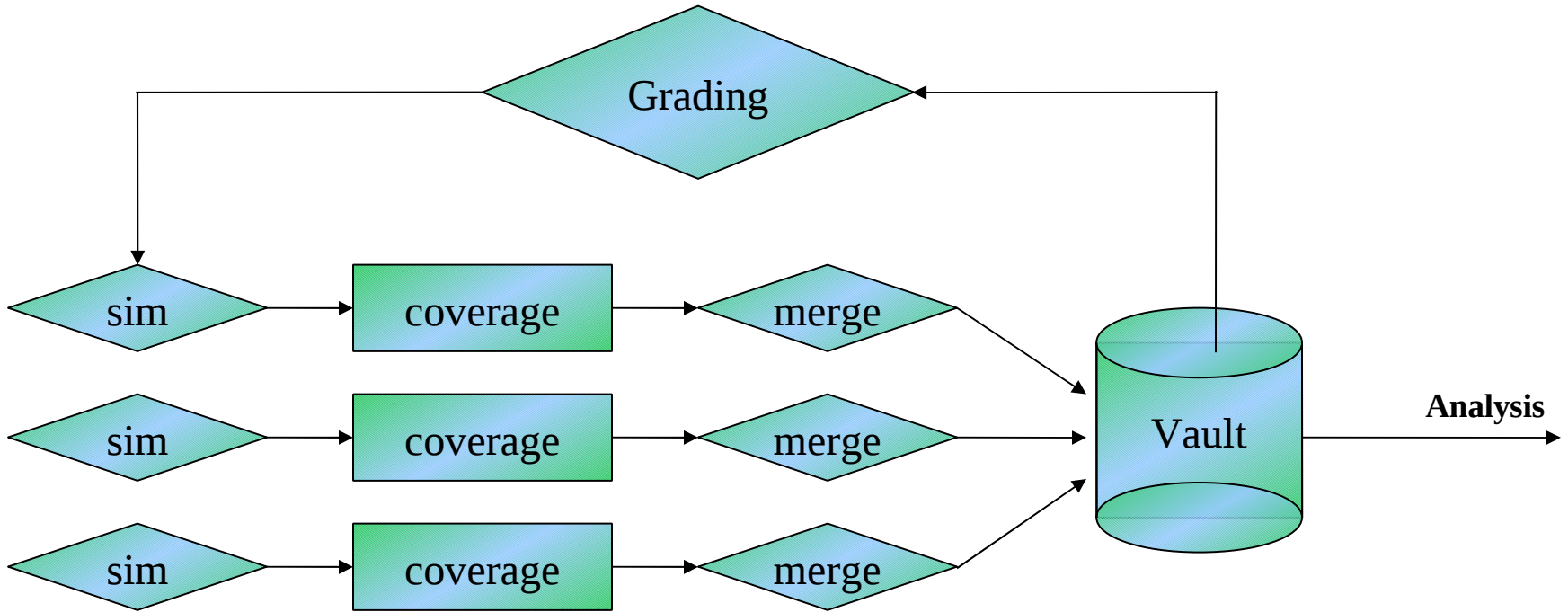
**ORACLE<sup>®</sup>**



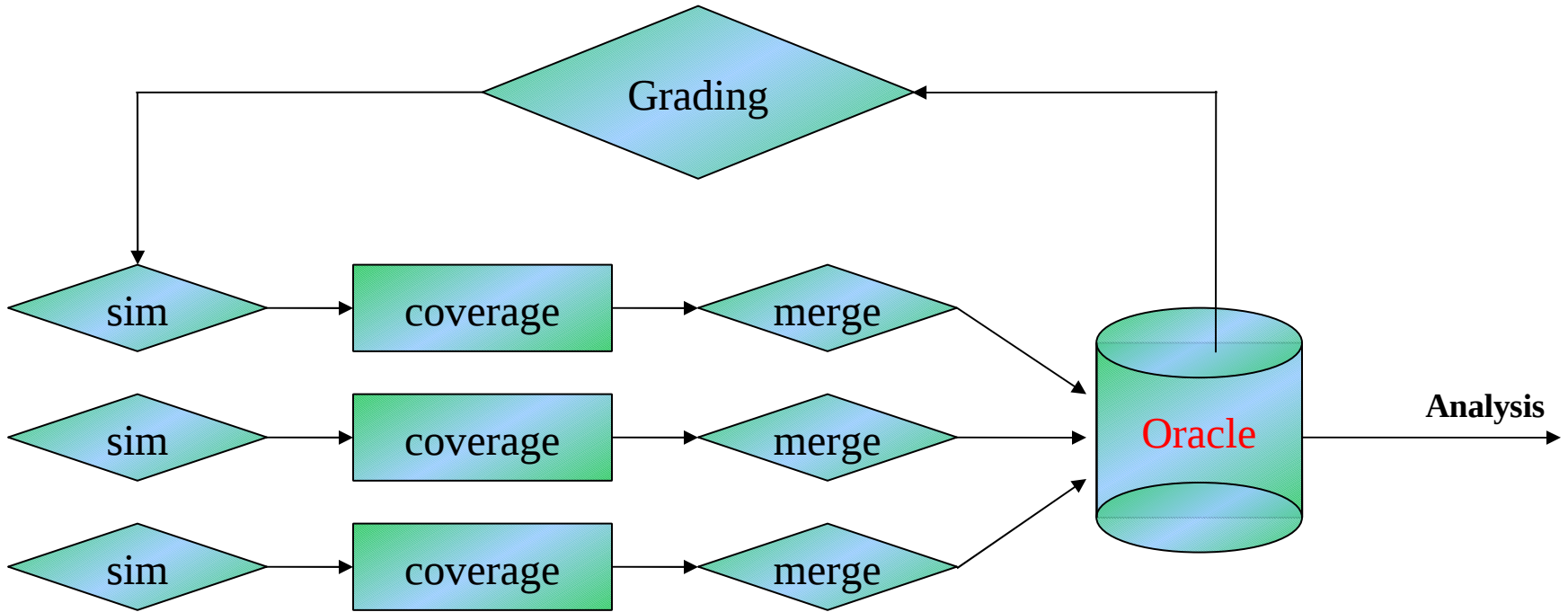
# **RDBMS-based Coverage Collection and Analysis**

James Roberts

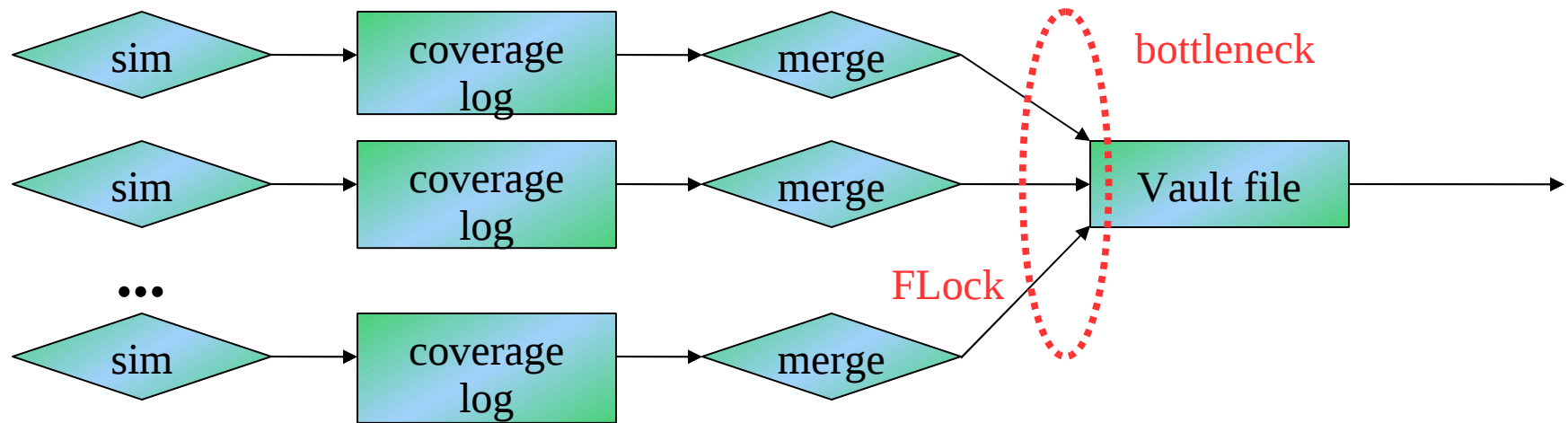
# Coverage-driven Flow



# Coverage-driven Flow



# Baseline: File-based flow

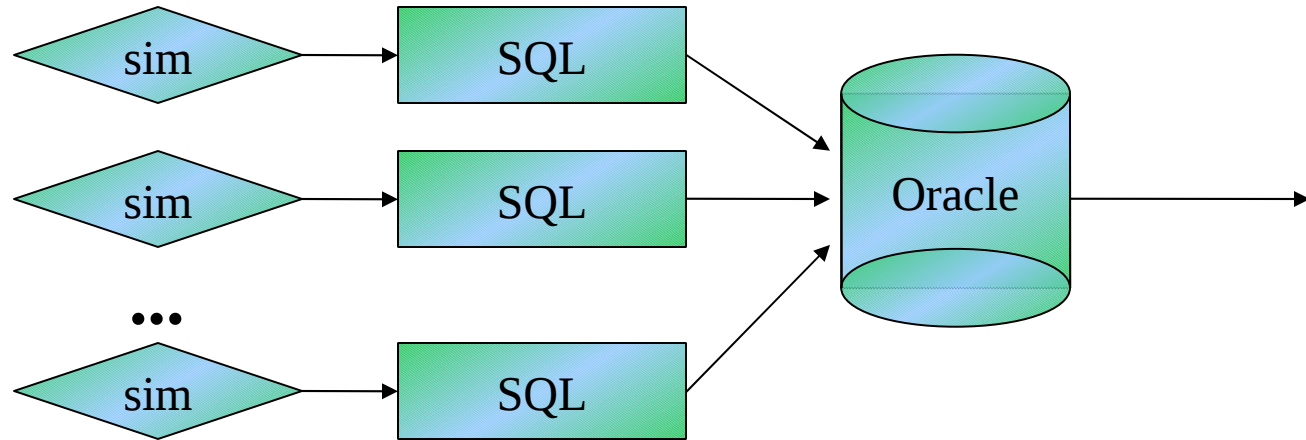


- Each sim writes out a log of its hit coverage
- Each log is diff-and-merged with the vault

## - *BAD:*

- *The vault can easily exceed 200,000 coverage objects and 300 Megabytes*
- *Diff-and-merge can exceed 10 minutes just for a single sim*
- *Only one sim can hold the file lock on the vault at one time*
- *These can be thousands of sims in parallel*
- *The vault is rewritten in its entirety after every merge*
  - *that means non-stop disk activity!*

# Coverage using a database



- Instead of a flat file, the vault is an Oracle database
- Instead of coverage logs, they are SQL queries to the database
- GOOD:
  - Each sim no longer requires a file lock on the entire vault
    - Merges can be parallel
    - Oracle database server is already very familiar with this kind of multiple-parallel-transaction situation
  - Only subset of coverage actually touched by sim is diff-and-merged
  - A rich set of SQL commands is available to do diag & coverage analysis on the vault
  - 3X speedup over file-based flow (and it's parallel)

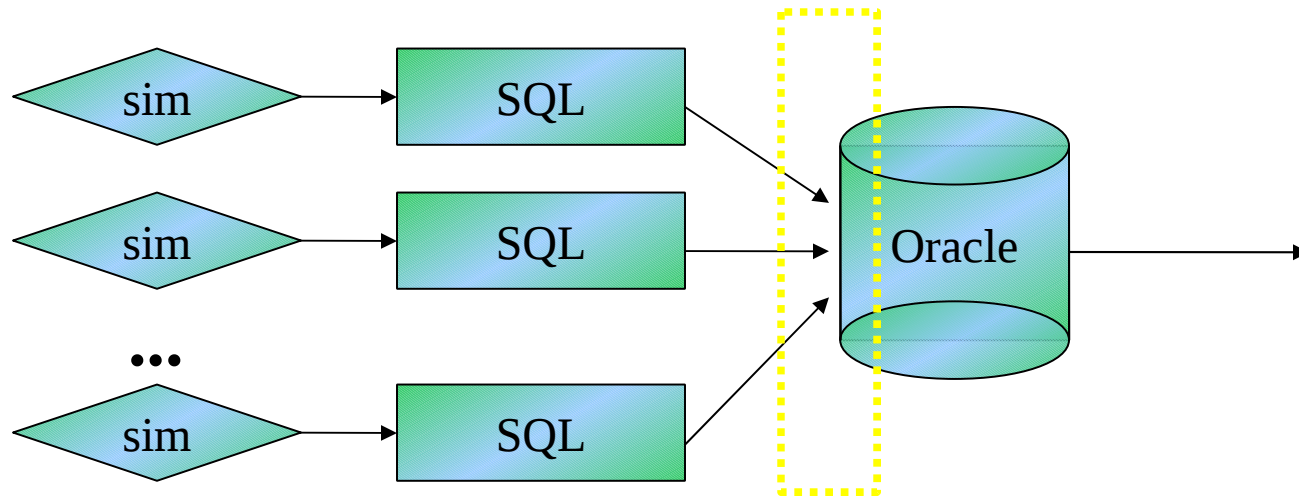
# A relational database table

## Vault

CId	Name	Block	Hits	(lock)
1	l2_hit_after_miss	L2	0	
2	ack_stg_2	SERDES	0->1	
3	clk_off	PADIO	1000	
4	bit1_carry_bit0	ALU	1	
...				
200,000				

Row-level locking

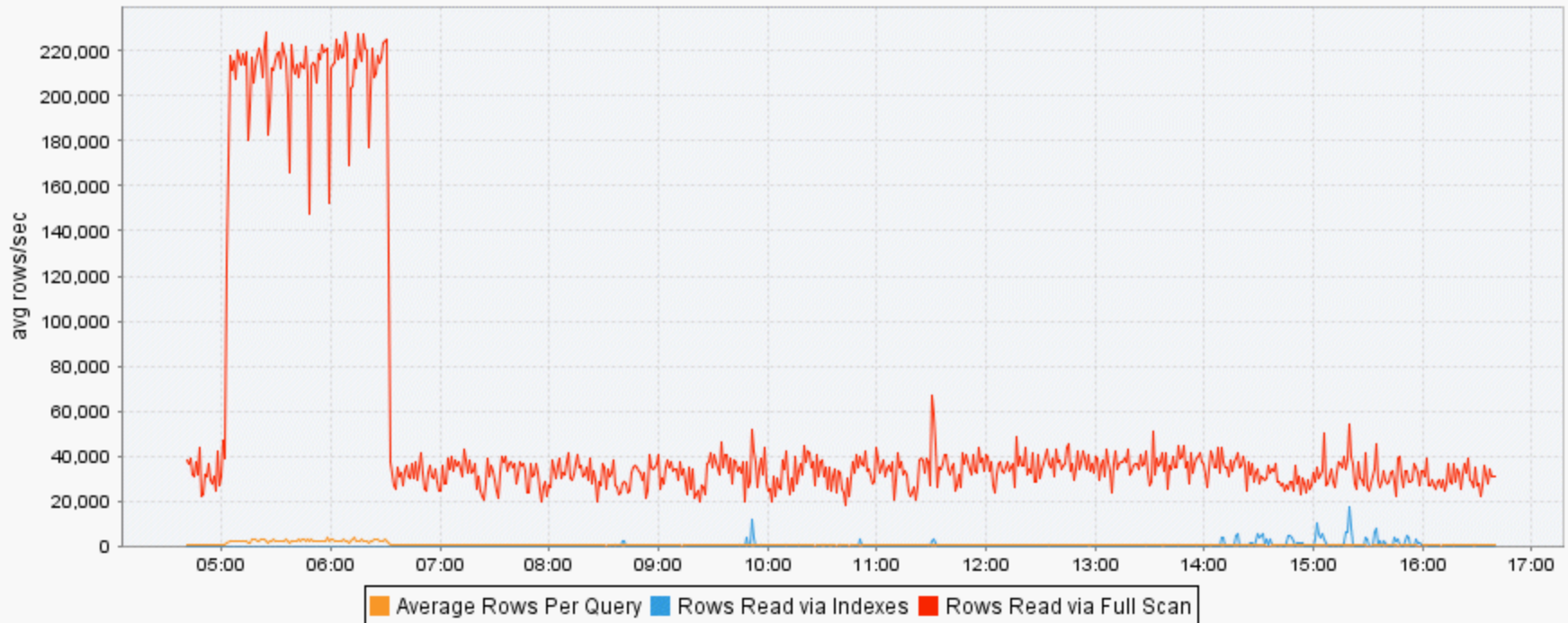
# Bottleneck: Oracle database flow



- + Oracle database server is multithreaded, and recognizes when coverage objects are unrelated and can be merged in parallel
- The fundamental problem remains, though: thousands of simulations contending for access to a single disk
- + We address this later by moving the merge to RAM

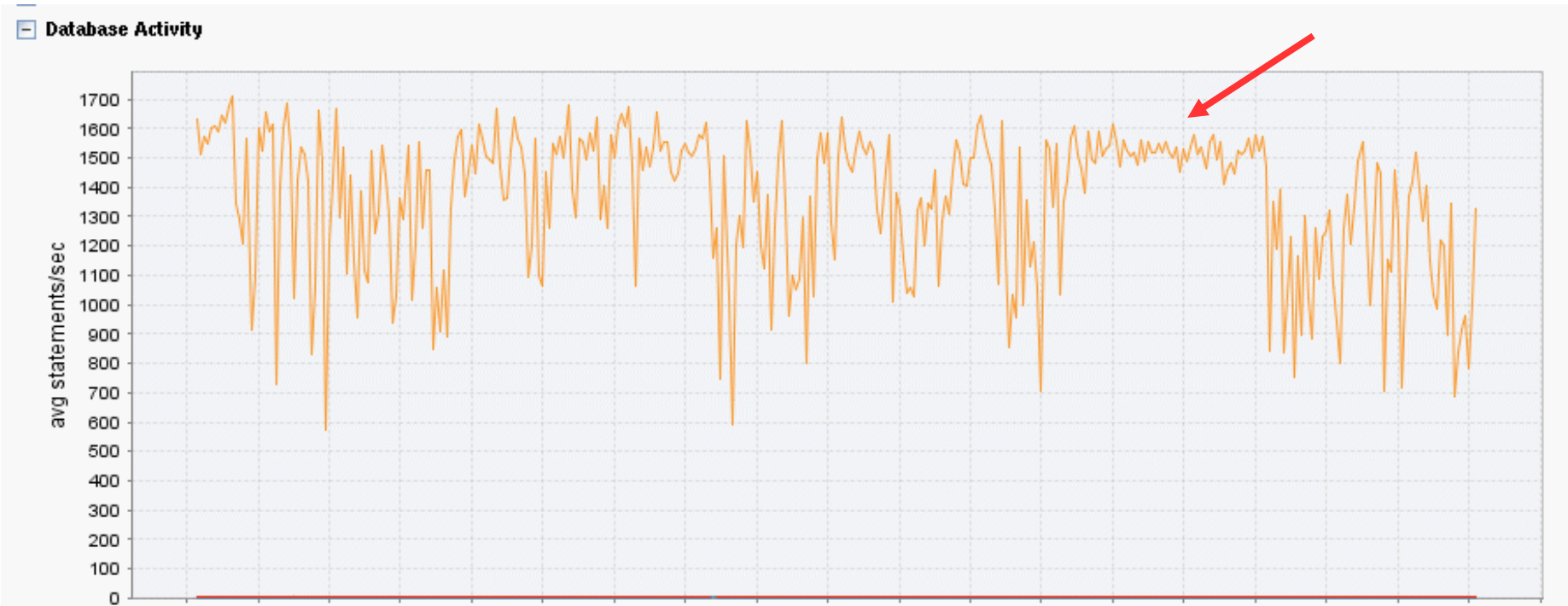
# Oracle database flow

## Row Accesses



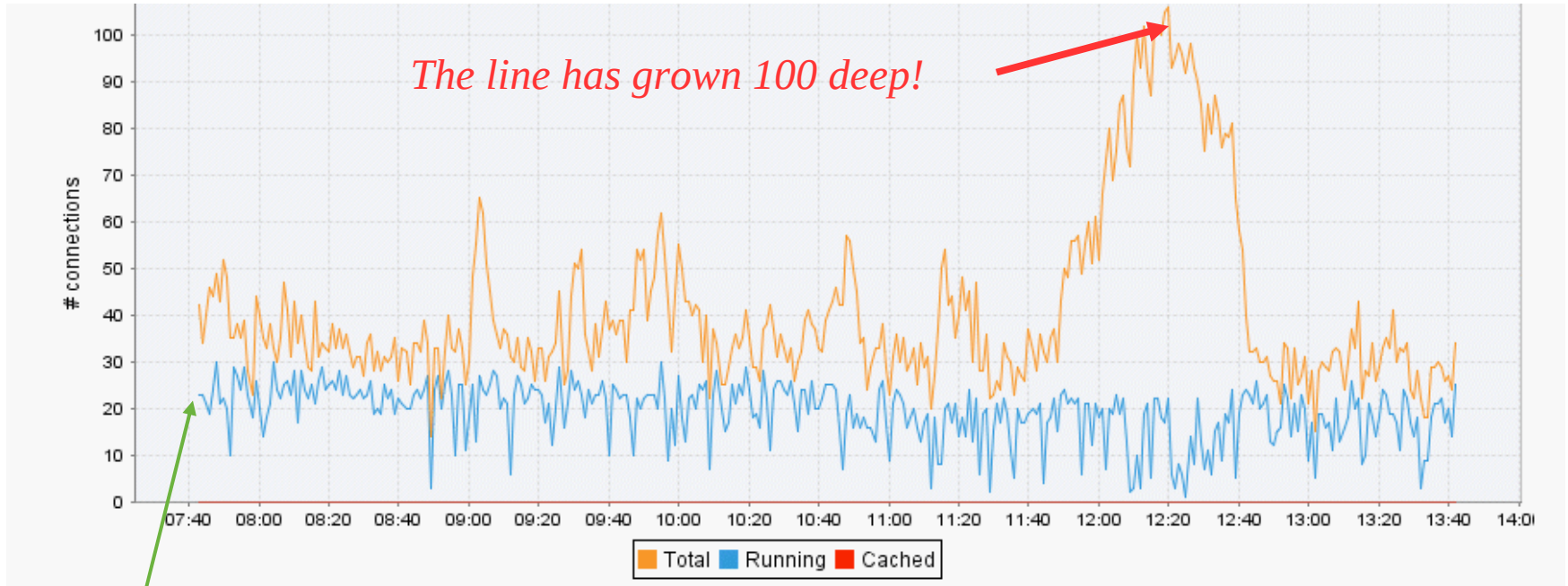
220,000 objects accessed per second,  
yet the entire database only has 170,000 objects

# Database activity



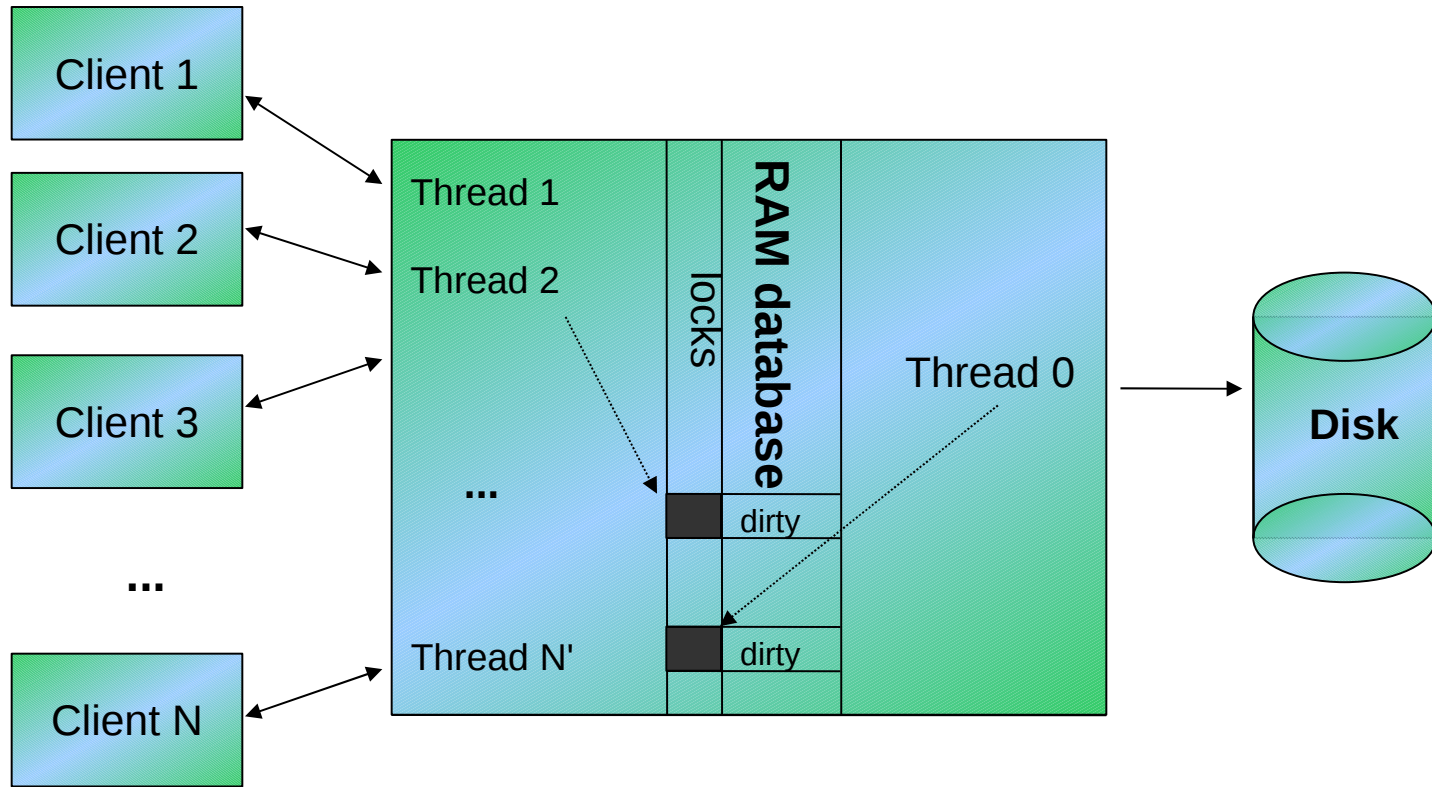
- Database server is bumping against a wall at around 1600 statements/second

# The Line Grows Longer



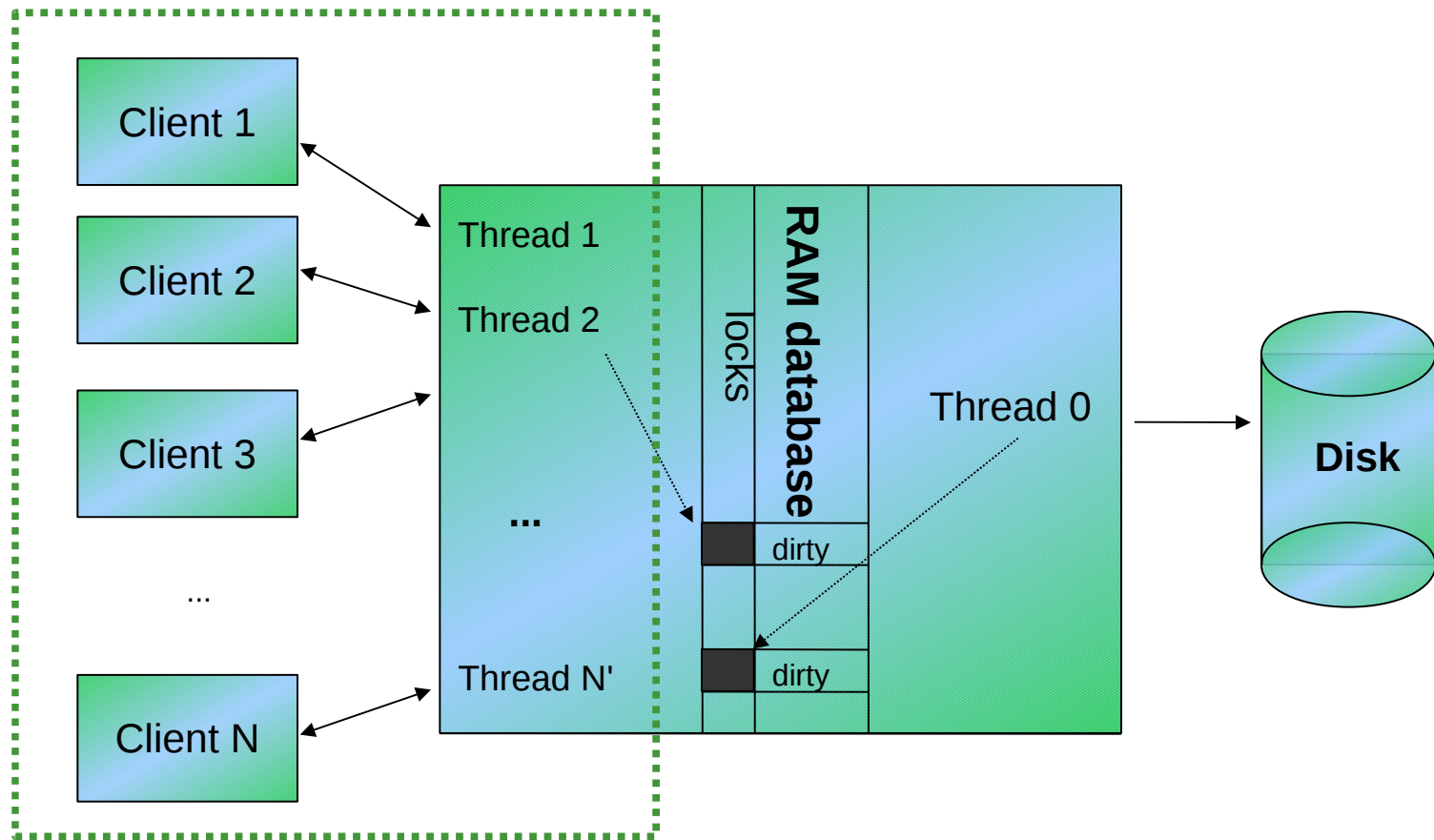
- + We're achieving a parallelism of 20. (this is good)
- As the database server cannot keep up with requests, the line of simulations grows longer and longer
- Workaround: throttle your simulations to acceptable levels

# Memory-resident database



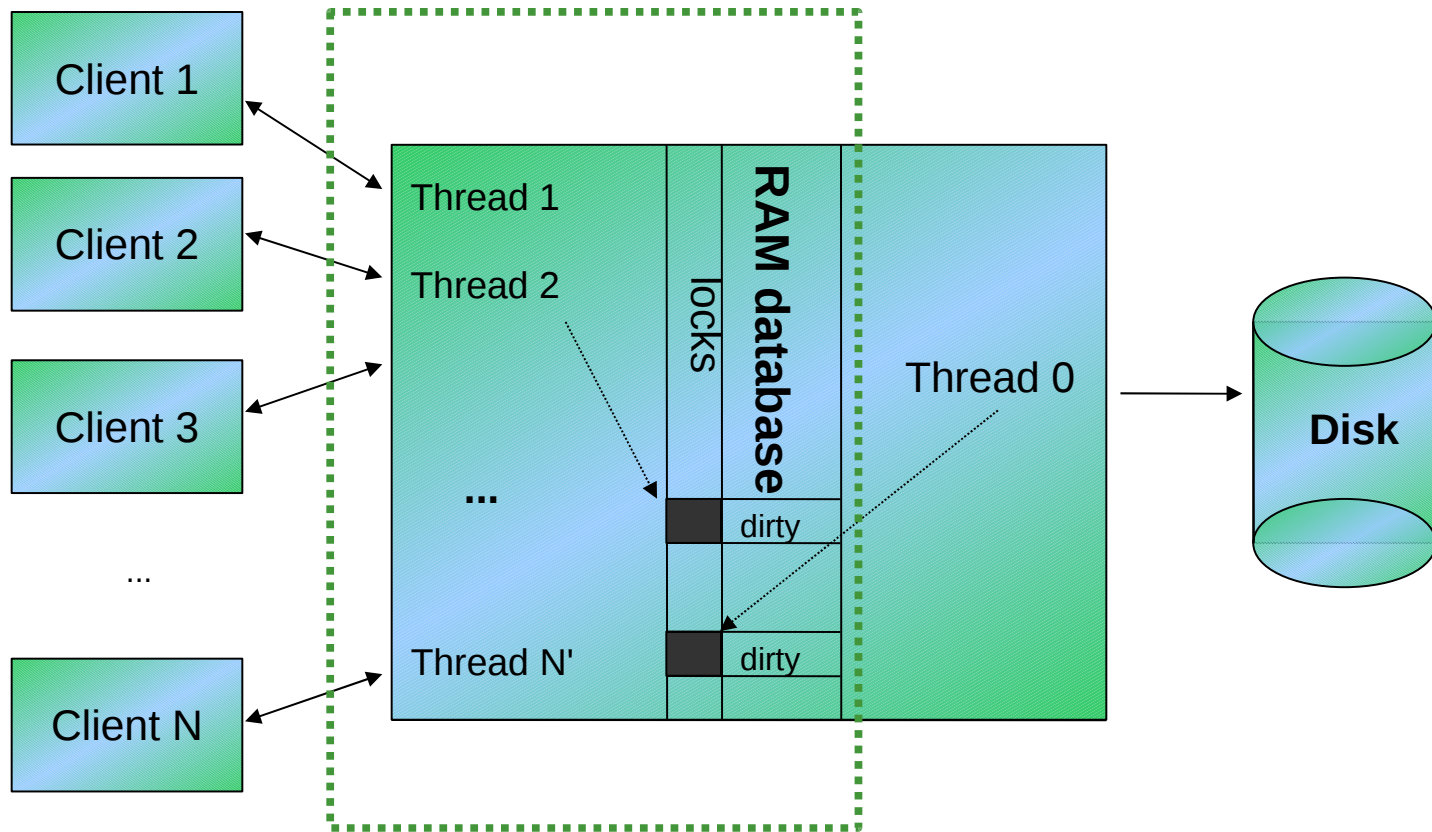
- Store main coverage database 100% in the RAM
- Multithreaded RAM database server
- Each simulation is a client

# Memory-resident database



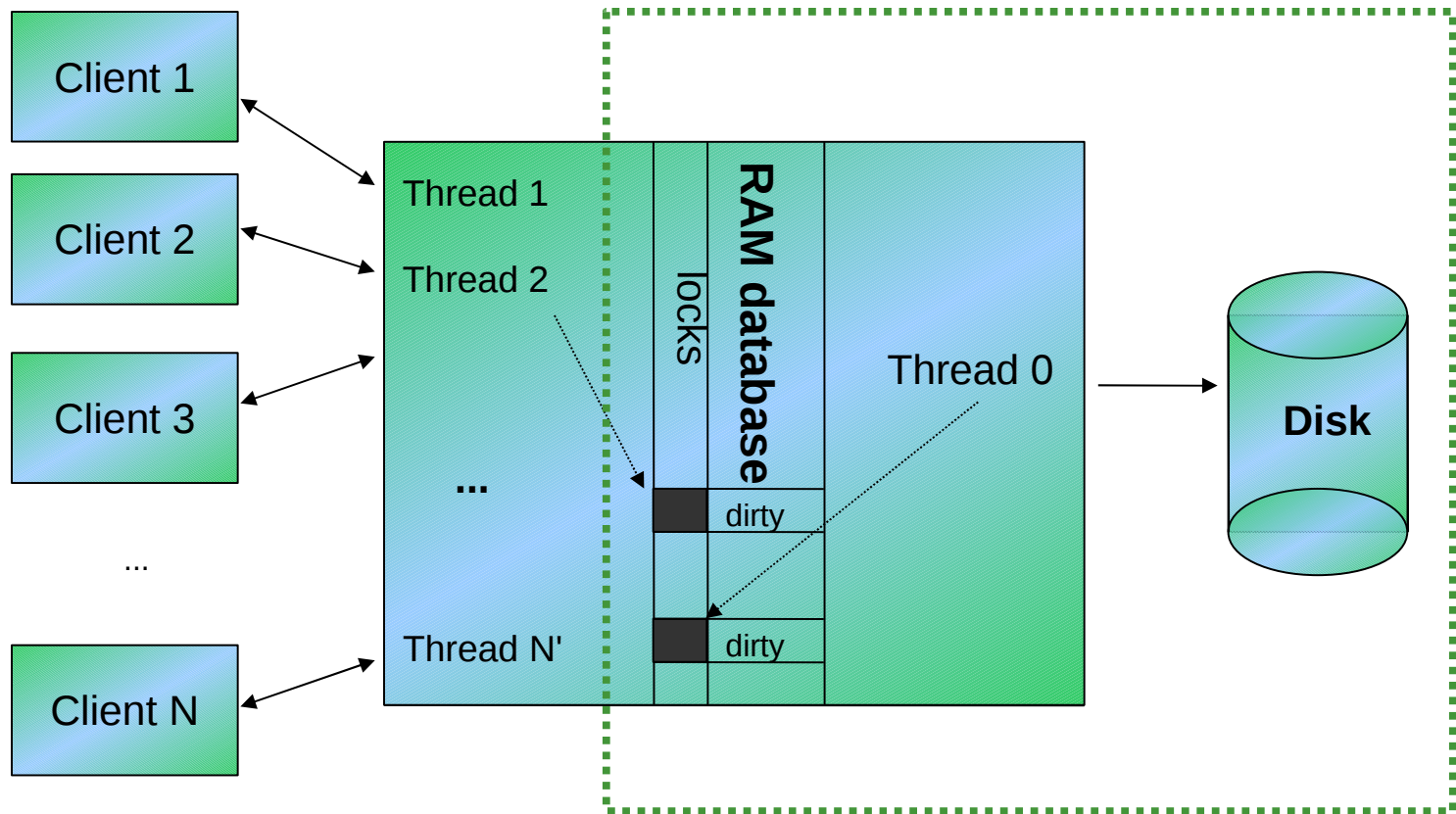
- simulations connect to server via TCP sockets, RPC
- coverage data transmitted directly to server
  - no disk involved

# Memory-resident database



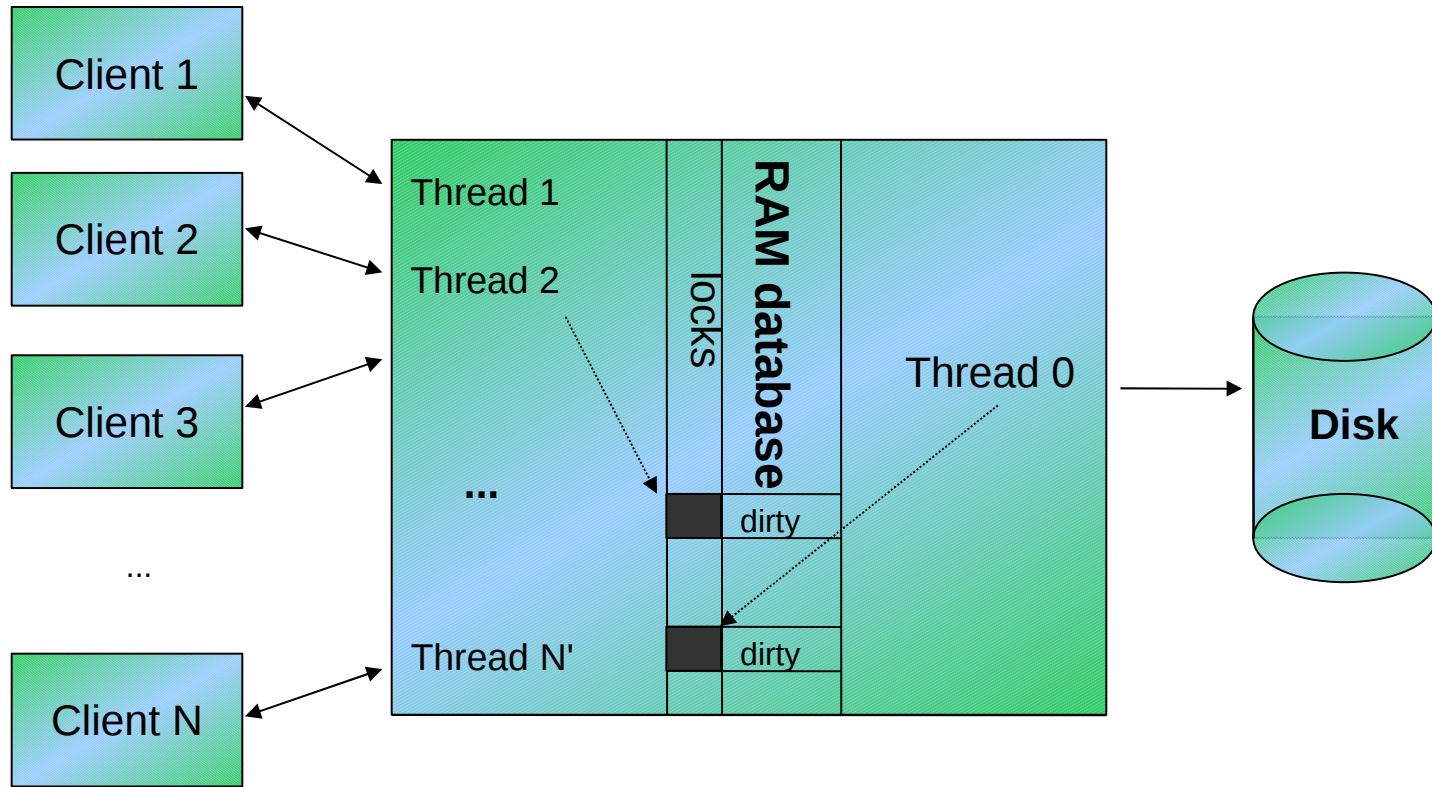
- Locking in RAM database to prevent contention across threads
- Dirty bits to flag updates
  - Updating dirty (but unlocked) objects perfectly legal

# Memory-resident database



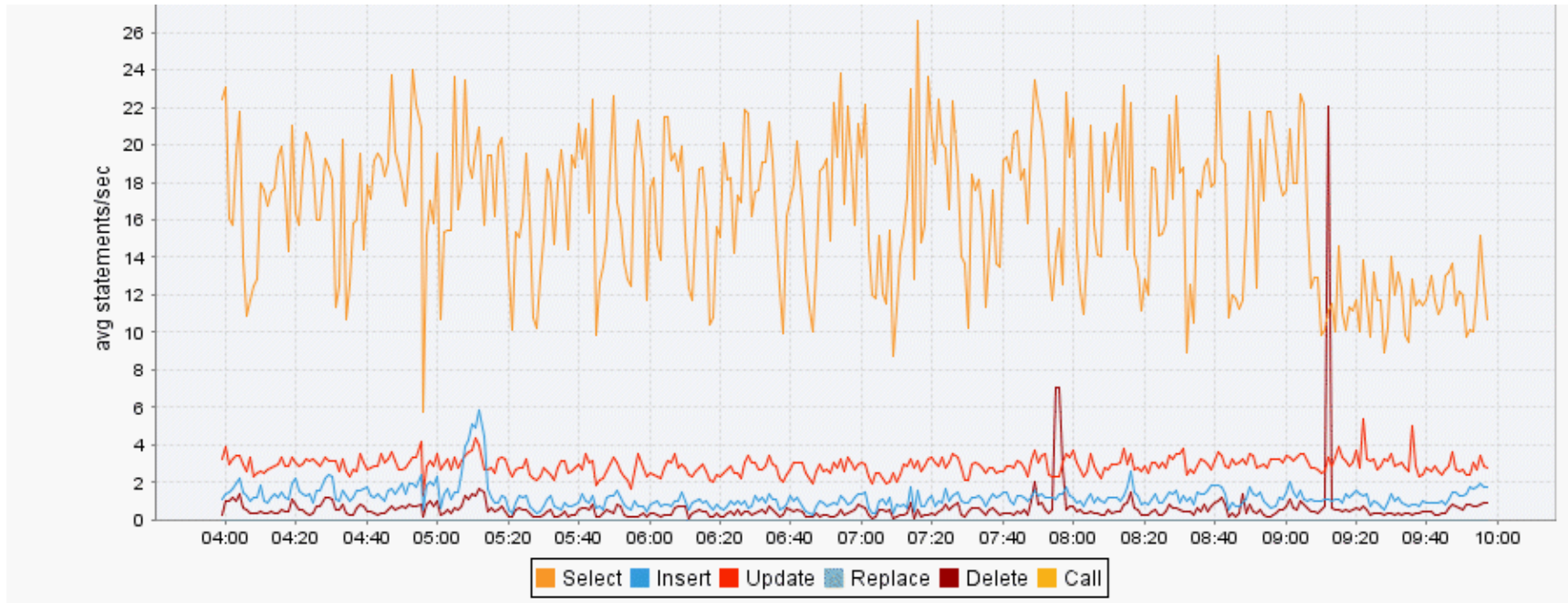
- Dedicated thread to flush updates to disk
  - Clears dirty bits
- + Disk never gets overloaded

# Memory-resident database



- Disk flush is completely decoupled from merge
- + *Client access time reduced from 30 minutes to 2 seconds!!! (900X speedup)*

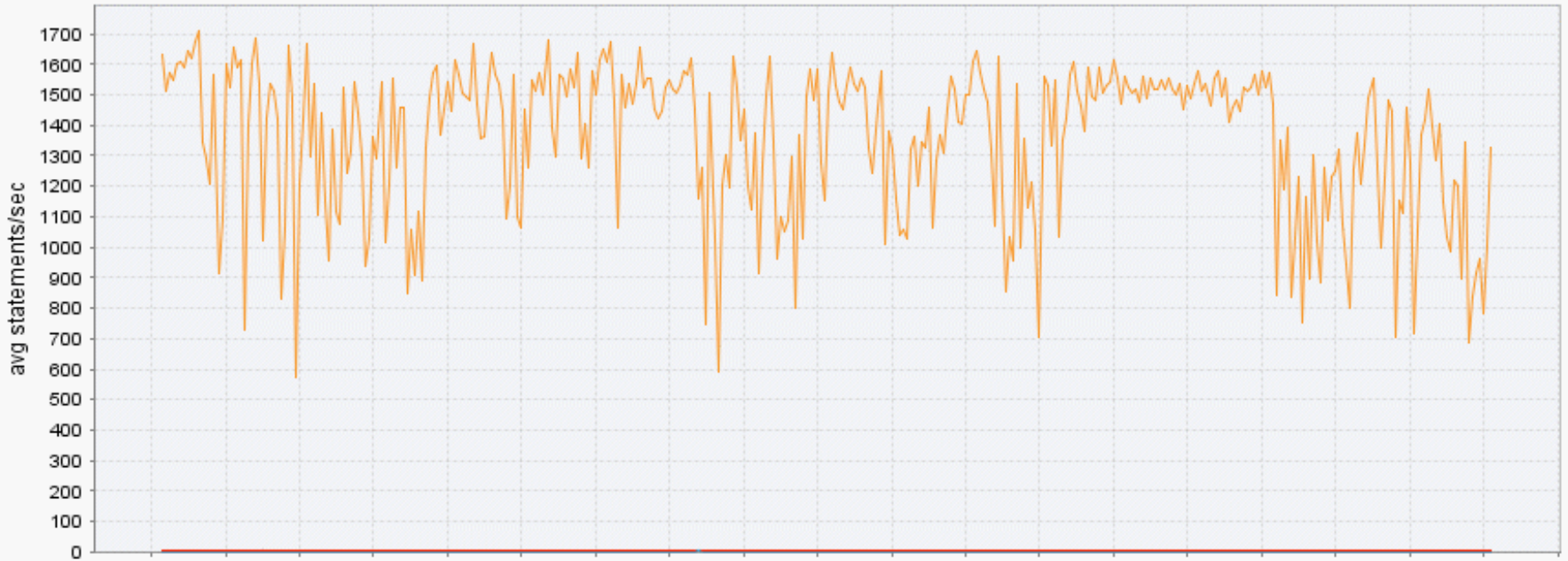
# Disk activity w/ Memory-resident DB



- Disk traffic reduced from 1600 statements/second to 18 statements/second
- *88X reduction in disk load!!!*

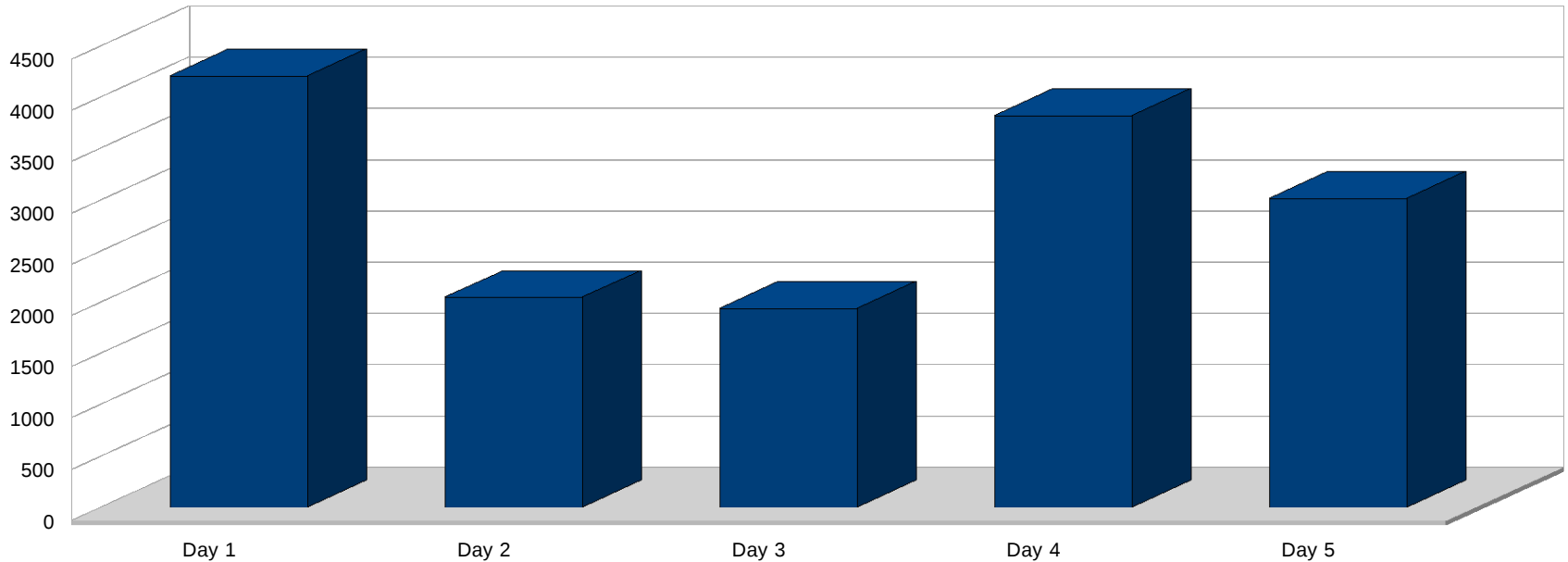
# Before: Non-Memory-Resident DB

Database Activity



- Compare with old results:  
1600 statements/second vs. 18 statements/second

# Disk I/O – 90% coverage population



- RAM cache filters out all the read accesses
  - at >90% coverage population, most database traffic is read-only
- 15030 disk writes over 5 days: or 1 disk I/O every 165 seconds
- + *Disk use is infinitesimal at >90% coverage*

# Conclusions

- Merge no longer the bottleneck in coverage flow
  - Unlimited simulations, where previously we had to throttle it
- 88X disk I/O reduction worst-case
  - Virtually zero I/O at >90% coverage
- 30-minute latency reduced to 2 seconds
- > 4-week uptimes for RAM server process
- Results have been positive to the point that people sometimes question our data

# Metrics

