The Traditional RDBMS Wisdom is All Wrong

by

Michael Stonebraker

DBg Database Group
MIT Computer Science and Artificial Intelligence Lab

Traditional RDBMS Wisdom

- Data is in disk block formatting (heavily encoded)
- With a main memory buffer pool of blocks
- Query plans
  - Optimize CPU, I/O
  - Fundamental operation is read a row
- Indexing via B-trees
  - Clustered or unclustered

Traditional RDBMS Wisdom

- Dynamic row-level locking
- Aries-style write-ahead log
- Replication (asynchronous or synchronous)
  - Update the primary first
  - Then move the log to other sites
  - And roll forward at the secondary (s)

Traditional RDBMS Wisdom

- Describes MySQL, DB2, Postgres, SQLServer, Oracle, …

- Focus of most college-level DBMS courses
  - Including M.I.T.
- Focus of most DBMS textbooks

Traditional RDBMS Wisdom

- Is obsolete
- i.e. completely wrong

DBMS Market (about third-sies)

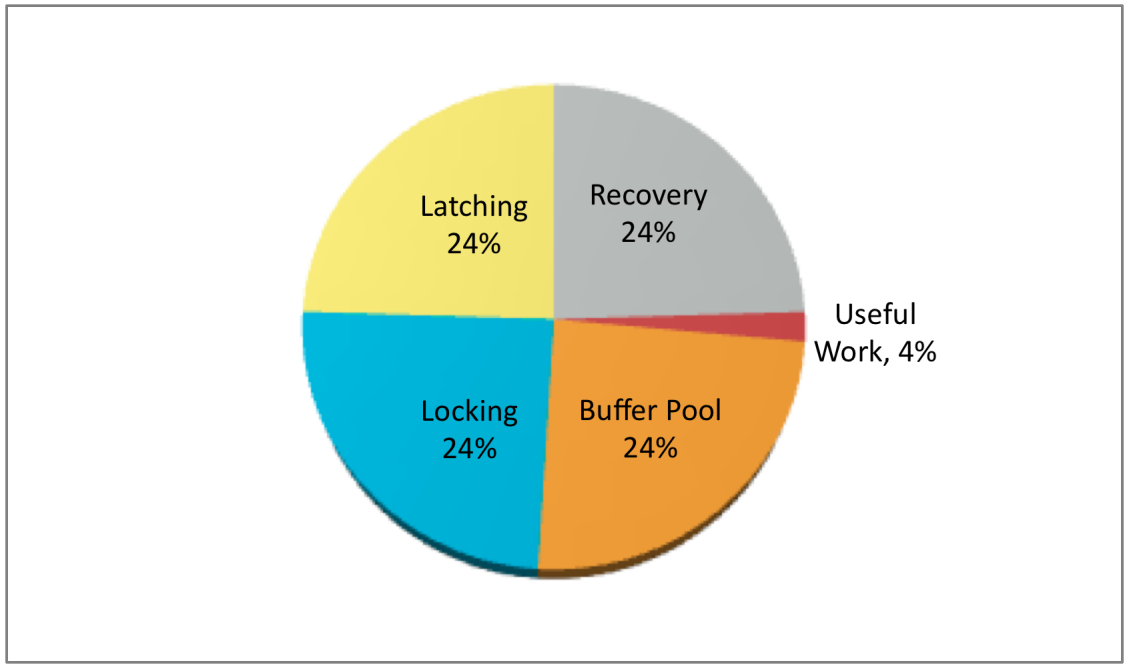
- Data Warehouses
  - Column stores will take over and don't look like the traditional wisdom
- Everything else
  - Hadoop, Graph-stores, No-SQL, array-stores,…
- OLTP
  - Focus of this talk!

Reality Check on OLTP Data Bases

- TP data base size grows at the rate transactions increase
- 1 Tbyte is a really big TP data base
- 1 Tbyte of main memory buyable for around $30K (or less)
  - (say) 64 Gbytes per server in 16 servers
- If your data doesn't fit in main memory now, then wait a couple of years and it will…..
- Facebook is an outlier

Reality Check – Main Memory Performance

- TPC-C CPU cycles
- On the Shore DBMS prototype
- "Elephants" should be similar



Motivated H-Store/VoltDB

- Main memory Linux SQL DBMS
- multi-node and sharded
- Stored procedure interface
- Pure ACID
- Fast
  - ~100X the elephants on TPC-C
  - ~10X No-SQL without giving up ACID
  - Scales to 3M TPC-C's per second
- Biggest use case is game state!

OLTP Data Bases -- 4 Big Decisions

- Main memory vs. disk orientation
  - Anti-caching is the answer
- Recovery strategy
  - Aries is dead; long live transaction logging
- Replication strategy
  - Active-active is the answer
- Concurrency control strategy
  - Determinism wins; nobody uses row level locking

To Go Fast

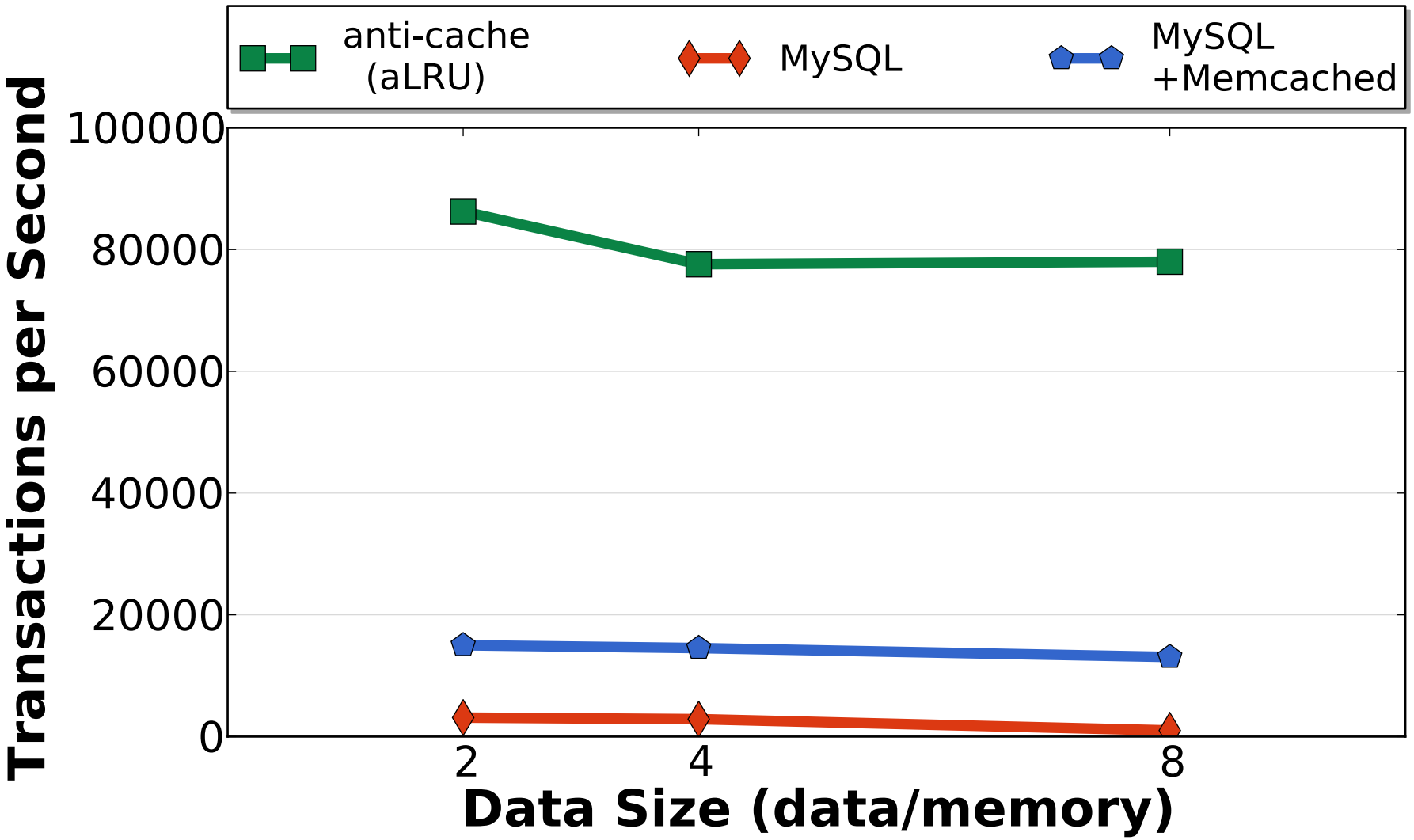
- Must focus on overhead
  - Better B-trees affects a small fraction of the path length
- Must get rid of all four pie slices
  - Anything less gives you a marginal win
- You cannot run a disk-based DBMS with a buffer pool!!!!

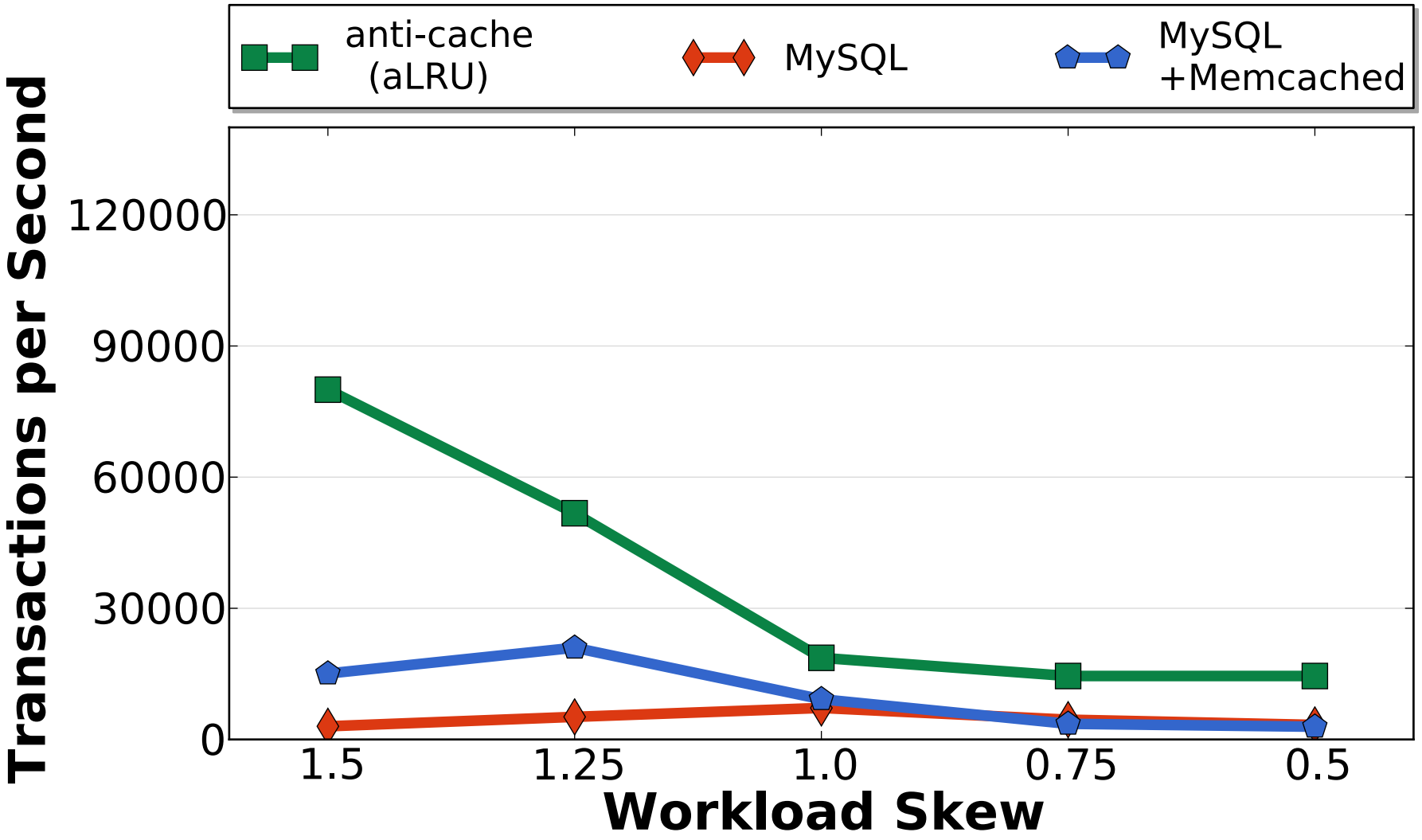
What if My Data Doesn't Fit?

- Use a disk-based DBMS and go slow
- Use Anti-caching

Anti-Caching (VLDB ‘14)

- Main memory format for data
- When memory fills, gather cold tuples and write to an archive (in main memory format)
- When a transaction has a “miss”, abort it but continue with “fake processing” to find all the absent data
- Get and “pin” the needed data
- Reschedule transaction when all needed data in main memory
- Numbers from H-Store implementation


anti-cache
(aLRU)
MySQL
MySQL
+Memcached
Transactions per Second
100000
80000
60000
40000
20000
0
2
4
8
Data Size (data/memory)



Advantages

- Better main memory management
  - 1 hot tuple won't force 99 cold tuples to stay in main memory with it
- No conversion of data back and forth between main memory and disk format

Disadvantage

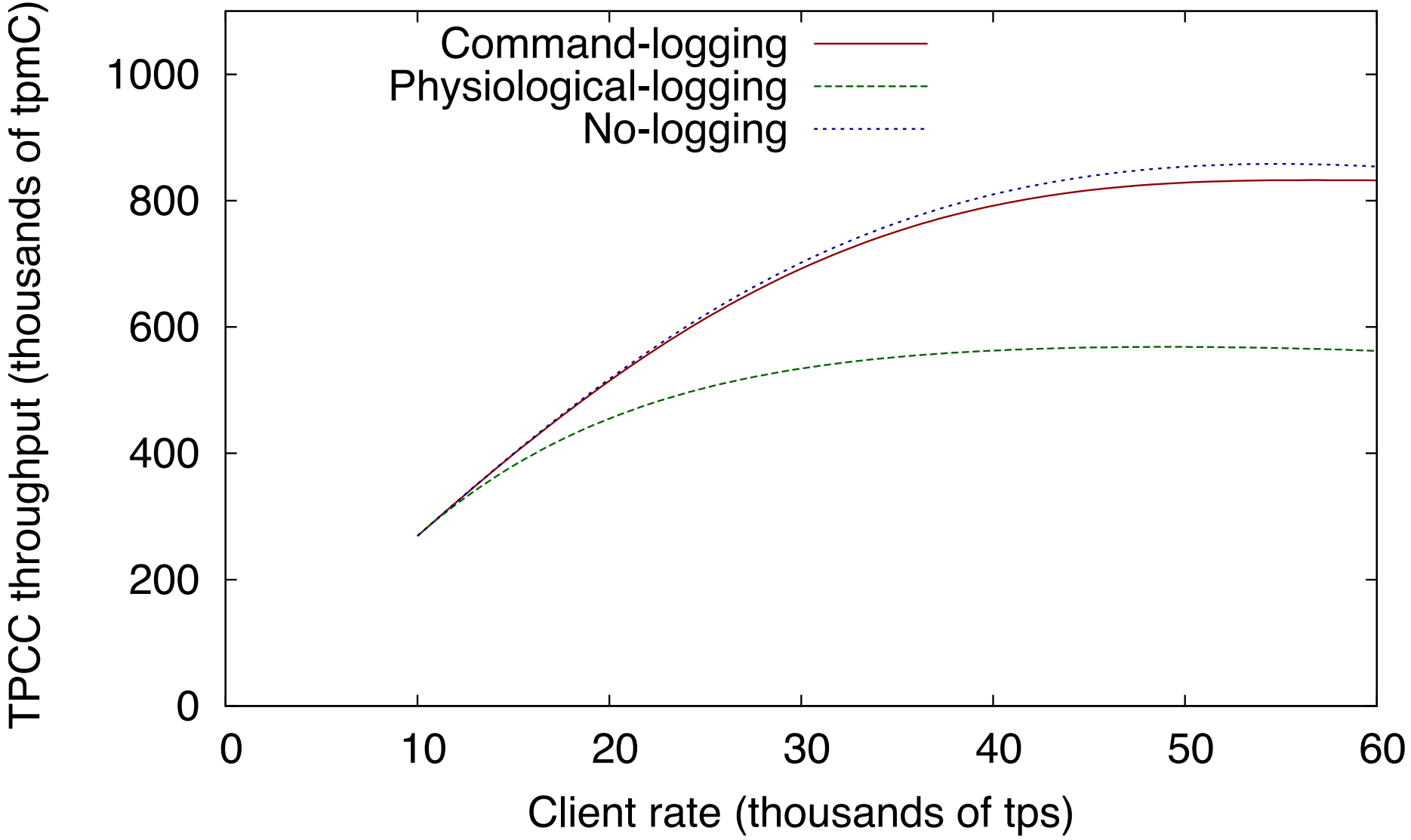
- Largest query (and all indexes) must still fit in main memory at one time
  - This is not a data warehouse!!
- Easy to fix with time travel

Conclusion

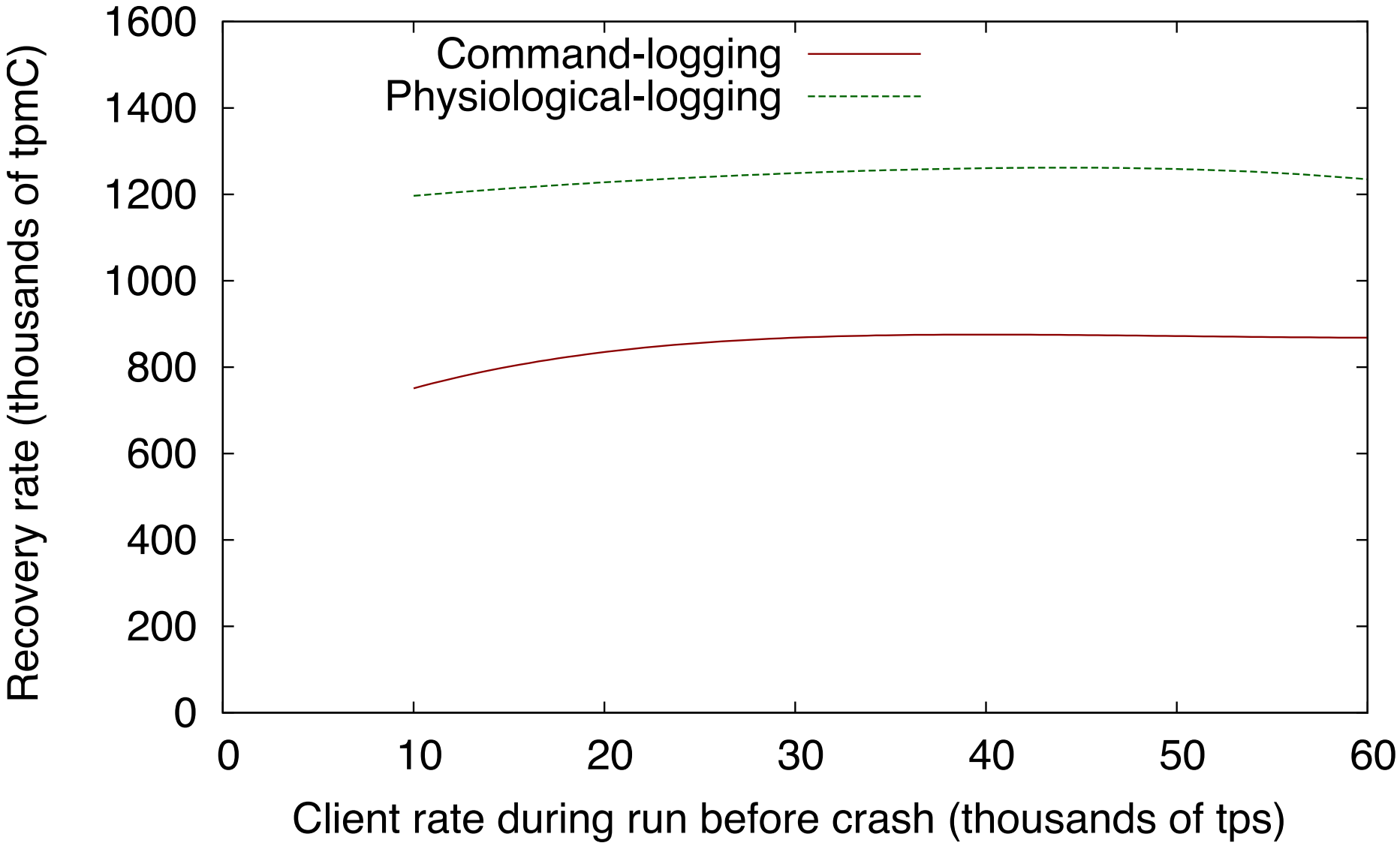
- There may be corner cases where anti-caching loses to a disk architecture
  - But we can't find one
- Main memory DBMSs are the answer!!!!
  - Hekaton, Hana, SQLFire, MemSQL, VoltDB, …

Some Data From Nirmesh Malvaiya

- Implemented Aries in VoltDB
- Compared against the VoltDB scheme
  - Asynchronous checkpoints
  - Command logging



DBg Database Group
MIT Computer Science and Artificial Intelligence Lab



DBg Database Group
MIT Computer Science and Artificial Intelligence Lab

Some Data From Nirmesh Malvaiya

- 1.5 X run-time performance gain
- 1.5 X penalty at recovery time

- Almost all OLTP applications demand HA
- Only run recovery for cluster-wide failures
  - E.g. power outage
- Bye-bye Mohan

How to Implement HA

- Active-Passive
  - As in the traditional wisdom
- Active-Active
  - Send update transactions to all copies
  - Each executes transaction logic

How to Implement HA

- Active-Passive
  - Write Nirmesh's data log over the network and roll forward at the backup node
- Active-Active
  - Send only the transaction, not the effect of the transaction
  - Allows read-queries to be sent to any replica

My Intuition – Active-Active will Cream Active-Passive

- Extend Nirmesh numbers to network traffic
  - 1.5 becomes 2 or 3 at run time
  - Roll forward stays at 1.5
- I.e. active-active will win
- Would be nice to prove this!!!

Concurrency Control

- MVCC popular (NuoDB, Hekaton)
- Time stamp order popular (H-Store/VoltDB)
- Lightweight combinations of time stamp order and dynamic locking (Calvin, Dora)
- I don't know anybody who is doing normal dynamic locking
  - It's too slow!!!!

The Nail in the Coffin

- Time stamp order compatible with active-active
  - As are any deterministic CC schemes
- Row-level locking and MVCC are not
  - Need a 2 phase commit between the replicas
  - Slow, slow, slow

Net-Net on OLTP

- Main memory DBMS
  - With anti-caching
  - And command logging
- Deterministic concurrency control
- HA via active-active

- Has nothing to do with the traditional wisdom!!!

Summary

- What we teach out DBMS students is all wrong
- Legacy implementations from the elephants are all wrong