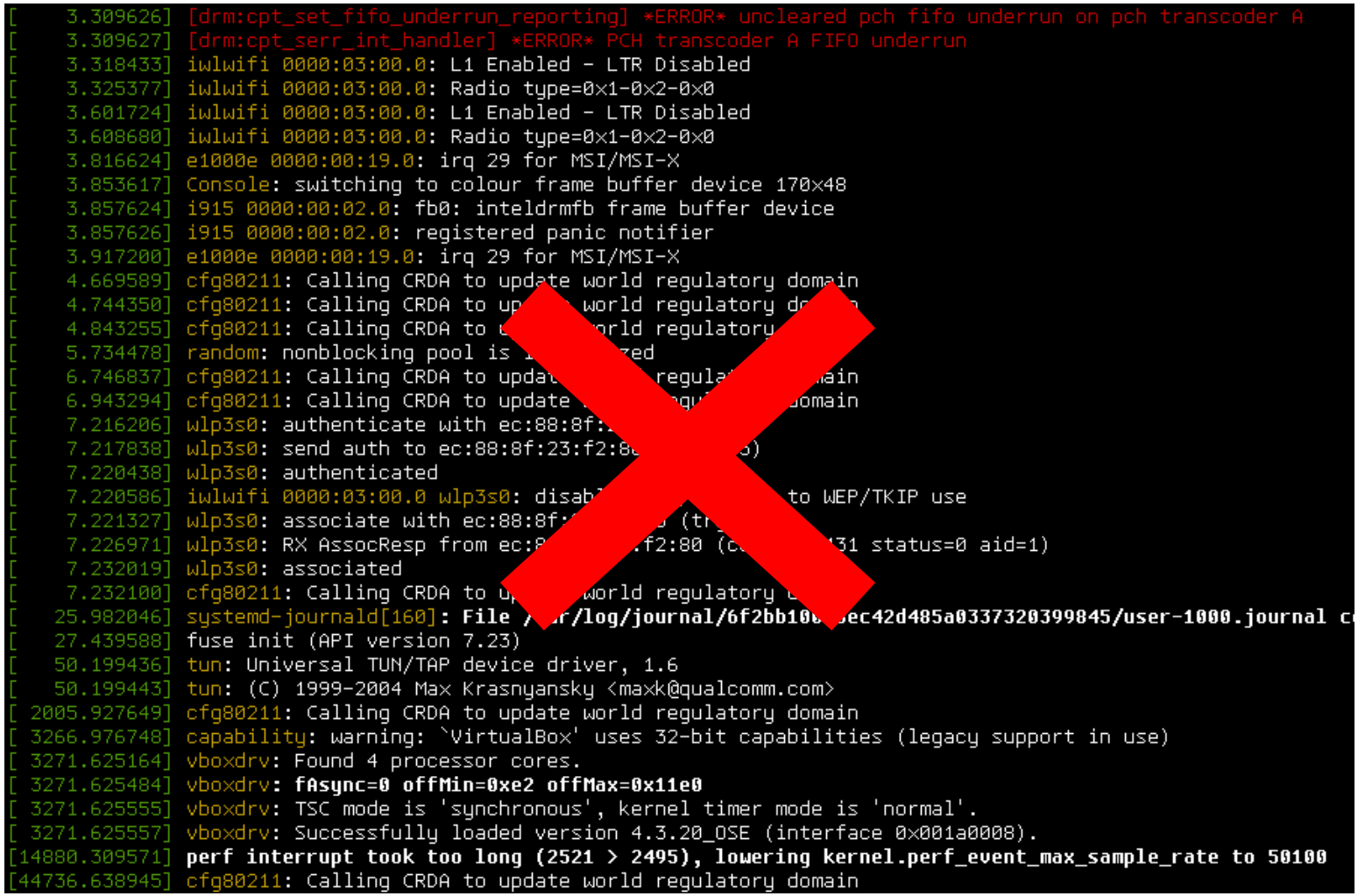
Transaction Logging Unleashed with NVRAM

Tianzheng Wang Ryan Johnson

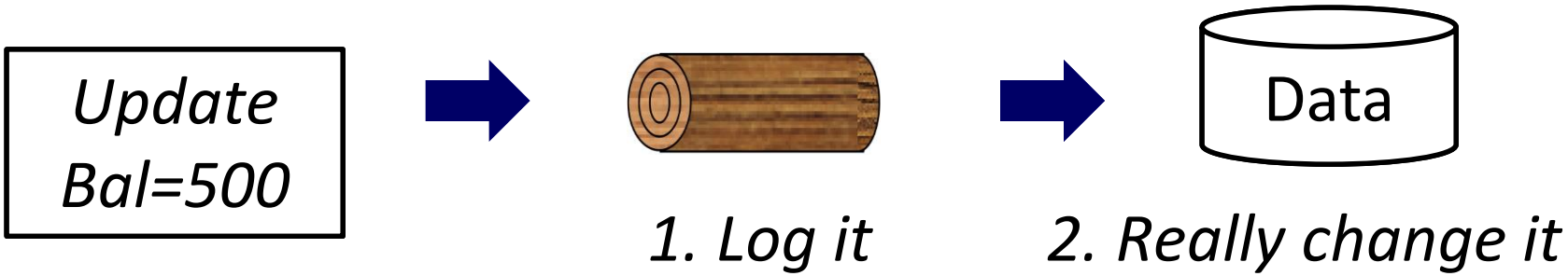
UNIVERSITY OF TORONTO

No, I’m not talking about the syslog



Write-ahead logging

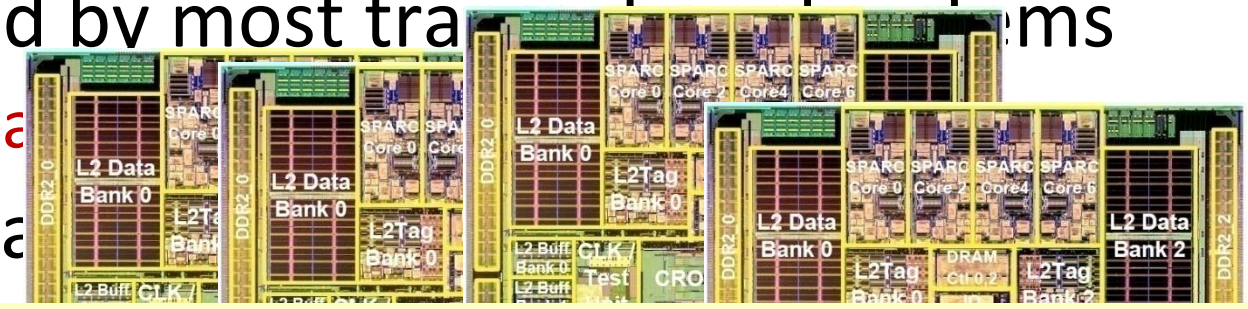
- Used by most transactional systems
  - Databases, file systems…
- Reliability
  - Everything goes to the log first, then the real place
  - Replay winners, rollback losers



- Performance
  - Buffer log records in DRAM
  - Disk/storage friendly long, sequential writes

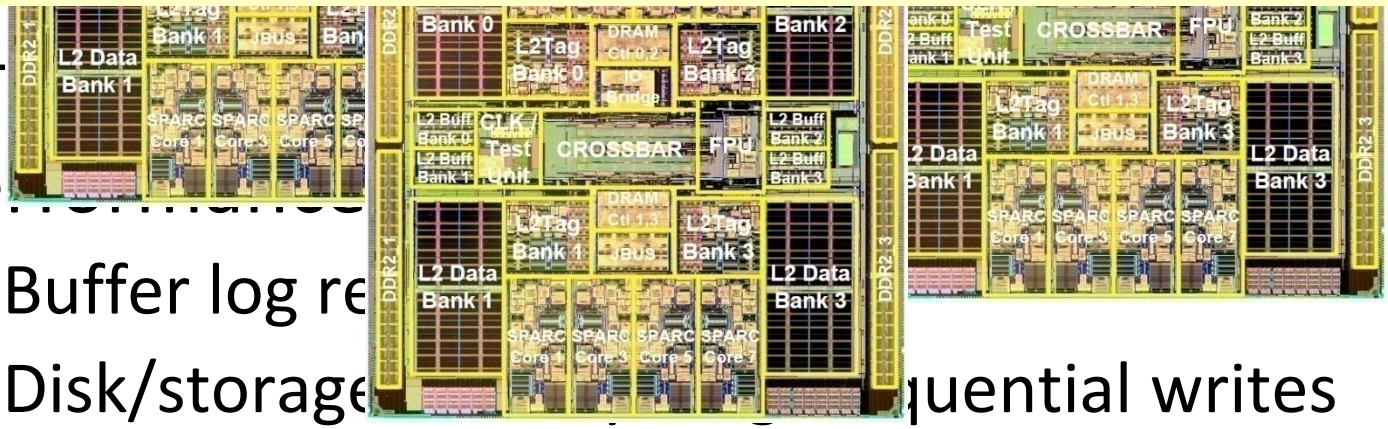
Write-ahead logging

- Used by most tra ms
  - Da
- Relia



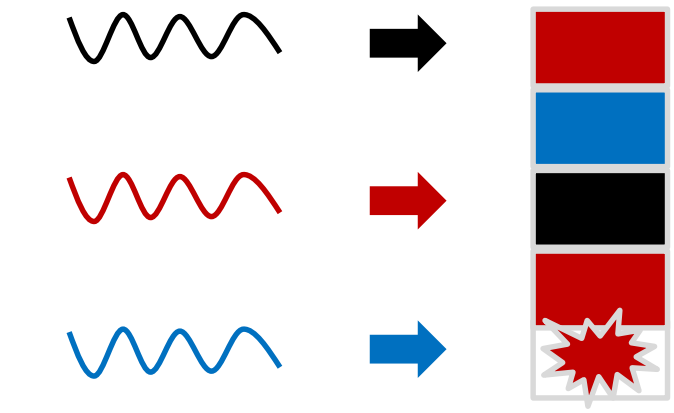
All was good until we had massively parallel hardware

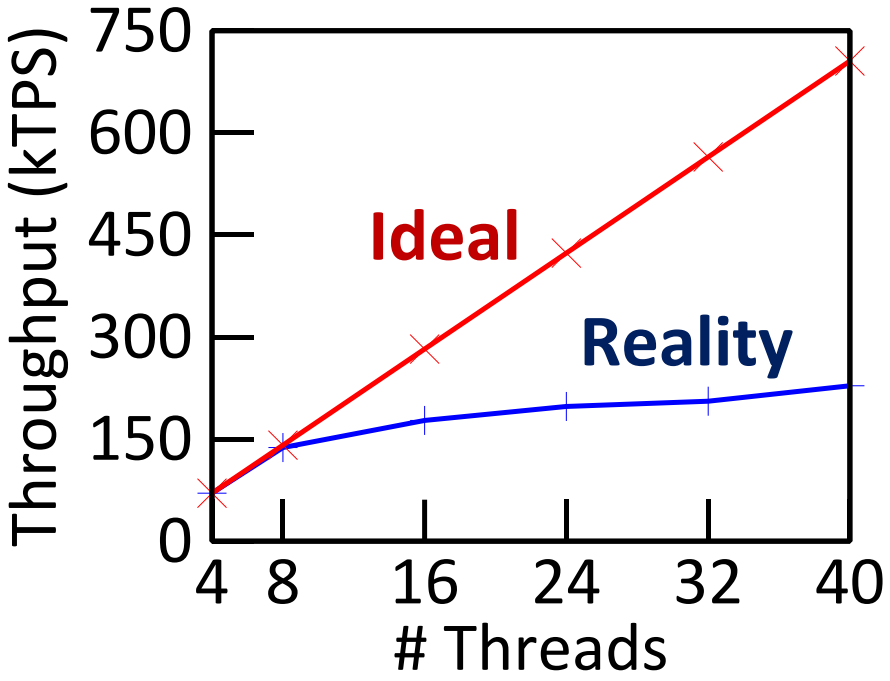
L *ange it*



- Pe
  - Buffer log re
  - Disk/storage quential writes

Centralized log: a serious bottleneck





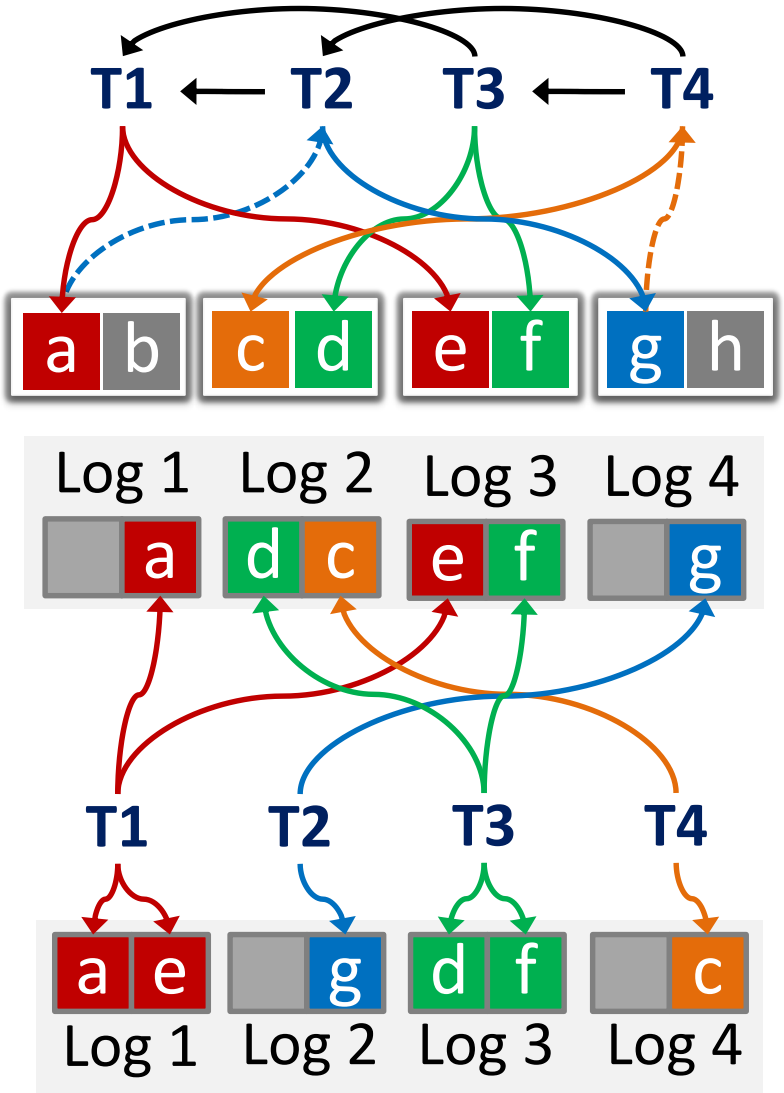
Why not distribute the log?

Sure!

But need the help of
byte-addressable, non-volatile memory (NVRAM).

The (impractical) distributed log

- Log space partitioning
  - by page or xct?
  - Impacts locality and recovery
- Dependency tracking
  - Direct xct deps: ***T4*** → ***T2***
  - Direct page deps: ***T4*** → ***T3***
  - Transitive deps: ***T4*** → {***T3***, ***T2***} → ***T1***
  - ***Easily end up flushing all logs***
- Storage is slow
  - System becomes I/O bound



The (impractical) distributed log

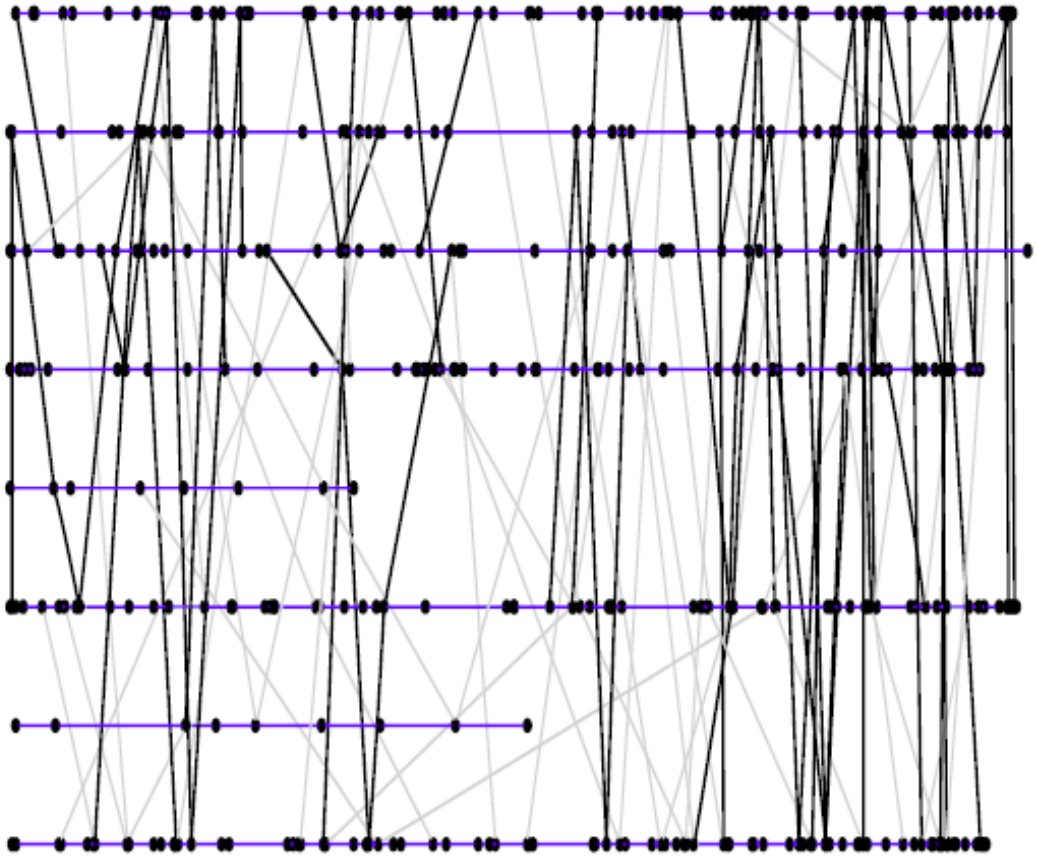


Figure 13. Inter-log dependencies for 1ms of TPC-C (8 logs, ~100kB, ~30 commits).

* R. Johnson etc., “Aether: a scalable approach to logging”, PVLDB 2010

The (impractical) distributed log

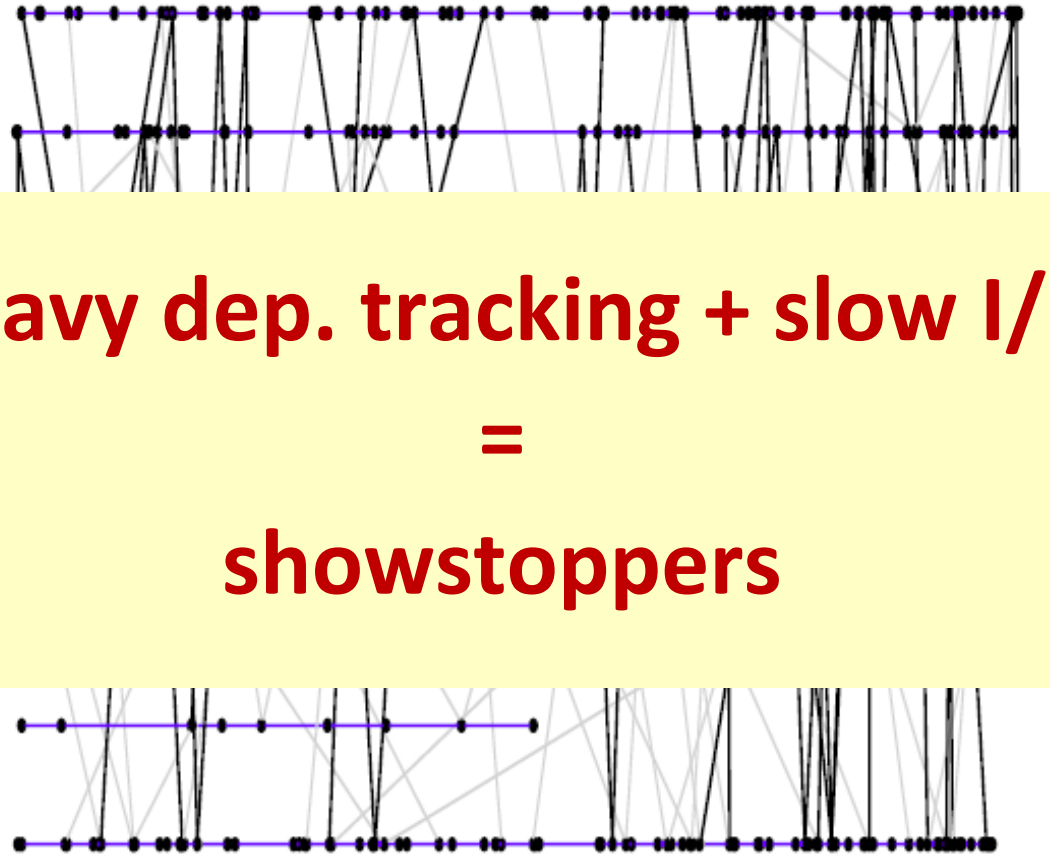


Figure 13. Inter-log dependencies for 1ms of TPC-C (8 logs, ~100kB, ~30 commits).

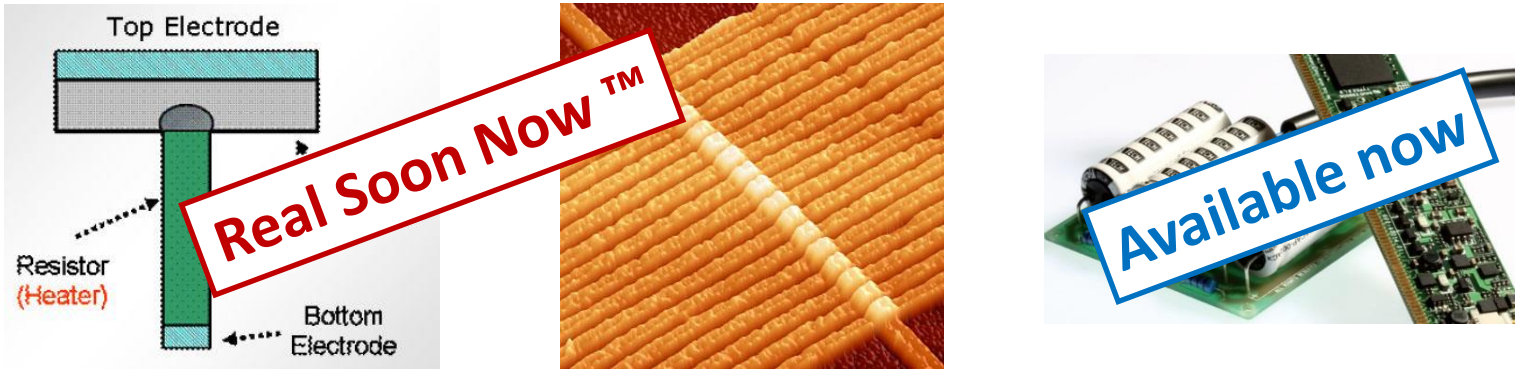
Heavy dep. tracking + slow I/O

=

showstoppers

* R. Johnson etc., “Aether: a scalable approach to logging”, PVLDB 2010

NVRAM to the rescue

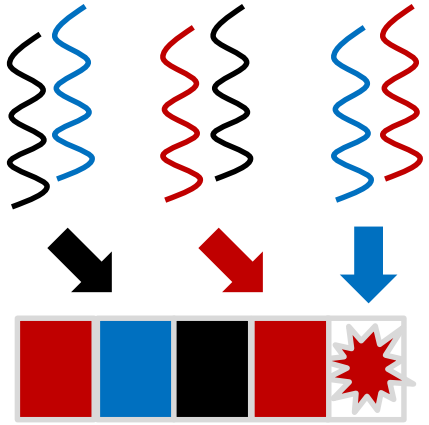


- NVRAM as ***log buffers*** for distributed logging
  - Log records durable once written
  - No dep tracking or flush-before-commit

Heavy dep. tracking + slow I/O = (*SOLVED*)

System architecture

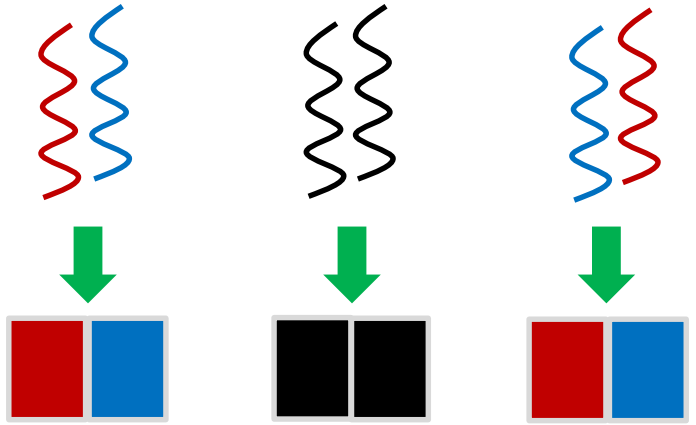
Before:



Log buffer (DRAM)

- Contend on a single log buffer
- Flush on commit or timeout

After:



Log buffers (NVRAM)

- Less or no contention
- Flush when buffers are full or timeout

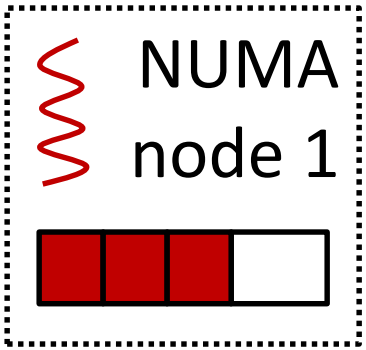
Challenges

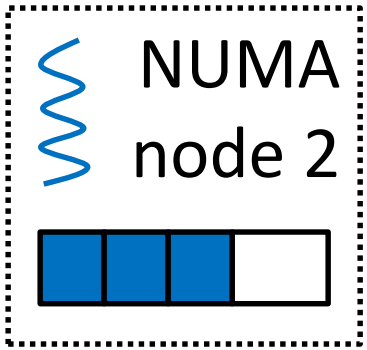
- NUMA effects
- Durability – processor cache is volatile
- Database system implications
  - Ordering
  - Uniqueness of log records
  - Recovery
  - Checkpointing
  - …

Problem #1: NUMA effects

- Partition-by-page => easier/simpler recovery
- Threads prefer to access ***local*** NVM node

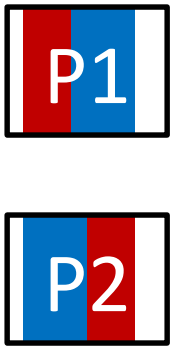
Transaction level:

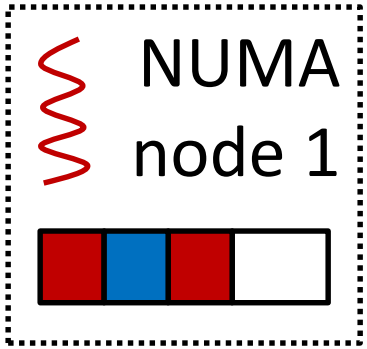


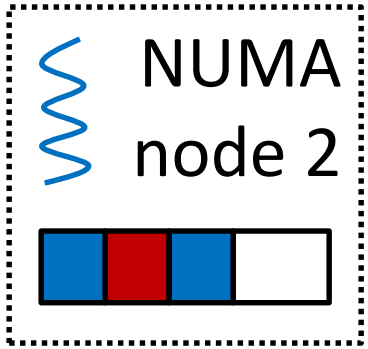


☺ **NUMA-friendly**

Page level:







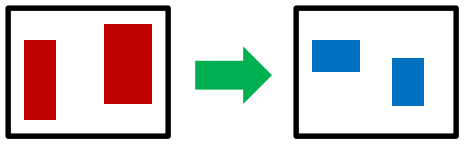
☹ **Cross NUMA boundary**

Prefer to partition by xct

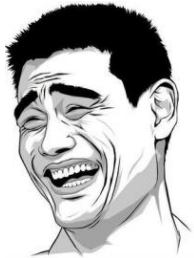
Problem #2: LSN gives partial order

- Log sequence numbers only good in any ***one*** log
- Recovery needs total order in any log/xct/page

The **same** page being modified:



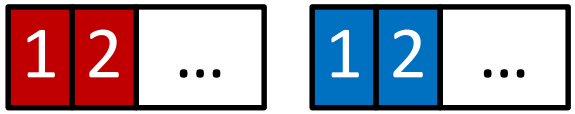
Recovery manager:



Transaction threads:

Same LSNs, whom first?

Log buffers:

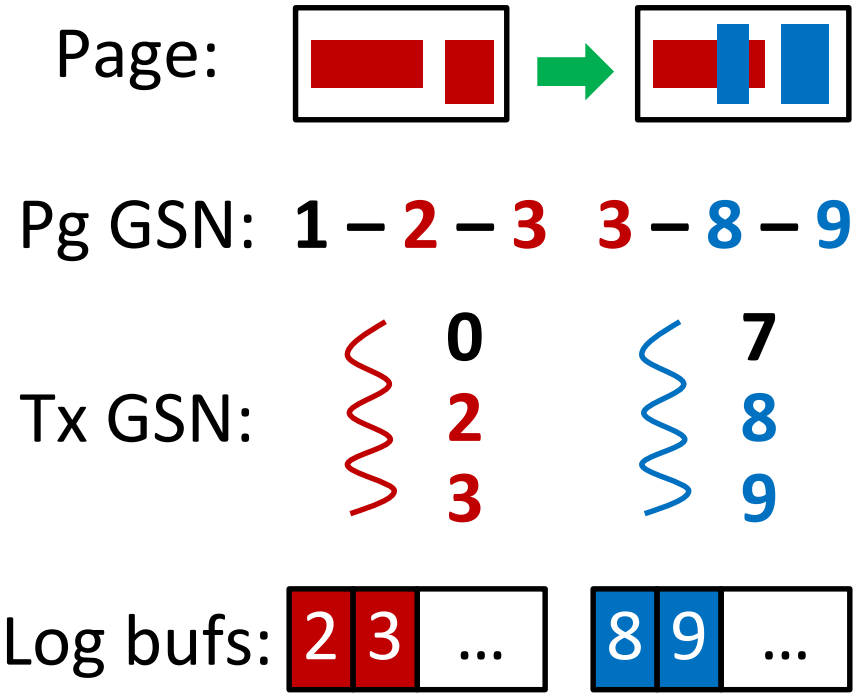


smaller ≠ earlier!

By-xct d-log needs global ordering of log records

Solution #2: global sequence number

- Based on Lamport's clock, no extra contention



How? Bump GSNs when the transaction latches pages and inserts log records

GSN:	Page	Transaction	Log
EX-latch	max(pg's, tx's) **+ 1**		/
SH-latch	/	max(pg's, tx's)	/
Log ins.	max (pg's, tx's, log's) **+ 1**		

GSN gives a partial, global order in each page, tx and log

Problem #3: Volatile CPU caches

- Log records must leave CPU cache before commit, preferably without dependency-tracking
- The ultimate solution: ***durable processor cache***
  - Candidates: FeRAM, SRAM + Supercapacitor…
  - Kiln [MICRO-46]
  - Whole system persistence [ASPLOS ’12]
  - Rohm nonvolatile CPU

But not available on the market

[CEATEC] Rohm Demonstrates Nonvolatile CPU, Power Consumption Cut by 90%

Motoyuki Ooishi, Nikkei Electronics — Oct 4, 2007

Like Share 0 Tweet 0 Print

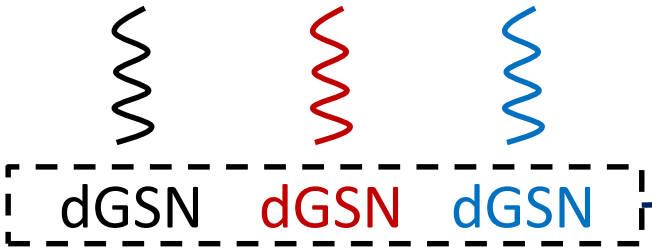
Rohm Co. Ltd. prototyped a nonvolatile CPU and exhibited it at CEATEC Japan 2007. The company prototyped an 8-bit microcomputer and made it nonvolatile by adding ferroelectric memory chips to all of the about 300 registers.

This time, the company demonstrated a breakout game by using a nonvolatile CPU, comparing with the case in which an existing CPU is used.

The demonstration of a breakout game

Problem #3: Volatile CPU caches

- Log records must leave CPU cache before commit, preferably without dependency-tracking
- Stop-gap solution: ***passive group commit***



on commit:

1. Flush local caches
2. Update ***local dGSN***
3. Enqueue transaction

Passive group commit daemon

Get ***min*** dGSN: 8

Dequeue xct with dGSN <= 8

TXN	dGSN
~~Xct 1~~	~~5~~
Xct 2	10

Commit queue

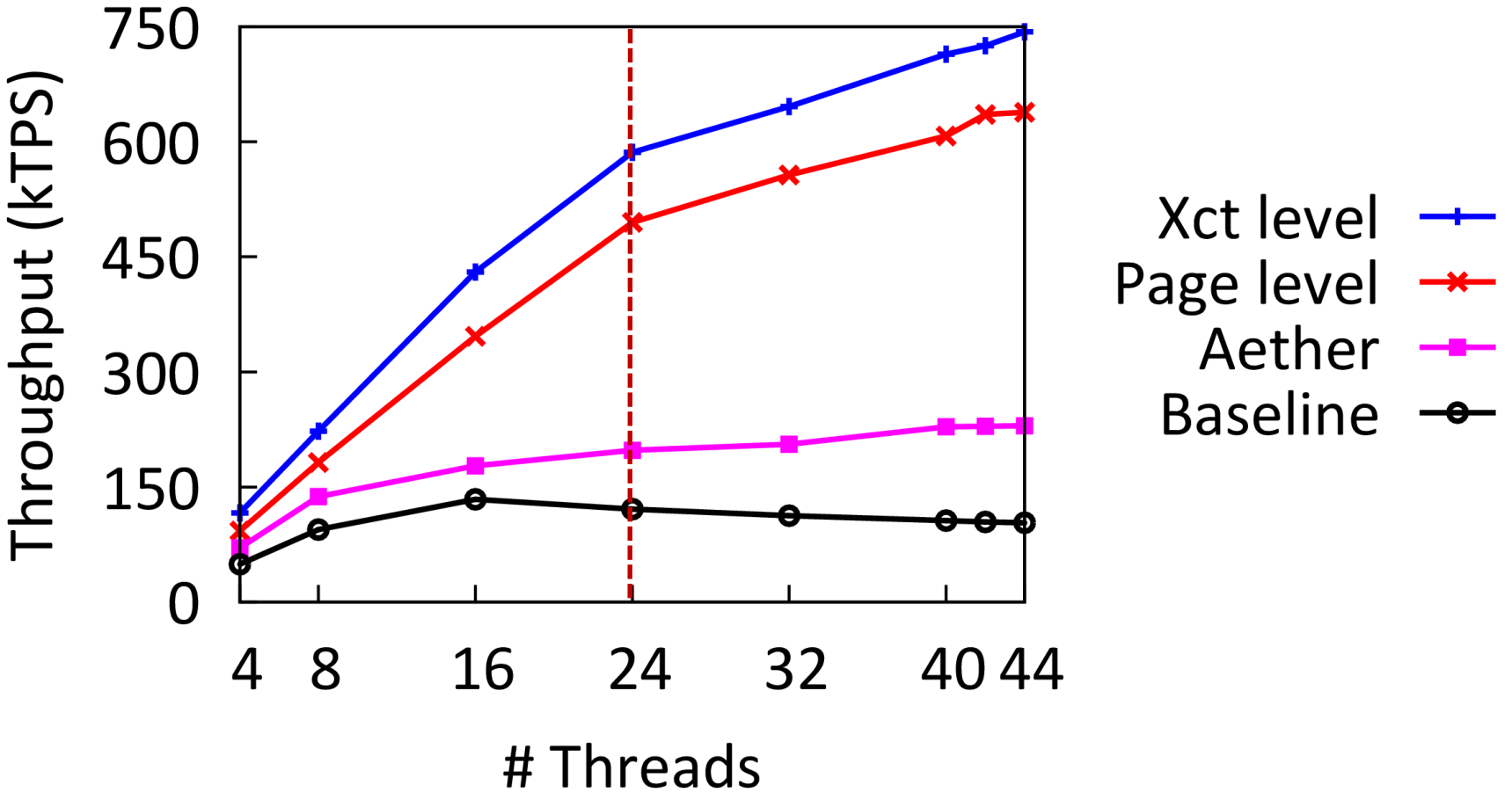
Evaluation

- Setup
  - 4-socket, 6-core Xeon E7- 4807 @ 1.8GHz
  - 24 physical cores, 48 “CPUs” with hyper threading
  - 64GB DRAM
  - NVM: flash/super-capacitor backed DRAM
- Workloads
  - Shore-MT, with Aether*
  - TPC-C: online transaction processing
  - TATP: telecom database applications

* R. Johnson etc., “Aether: a scalable approach to logging”, PVLDB 2010

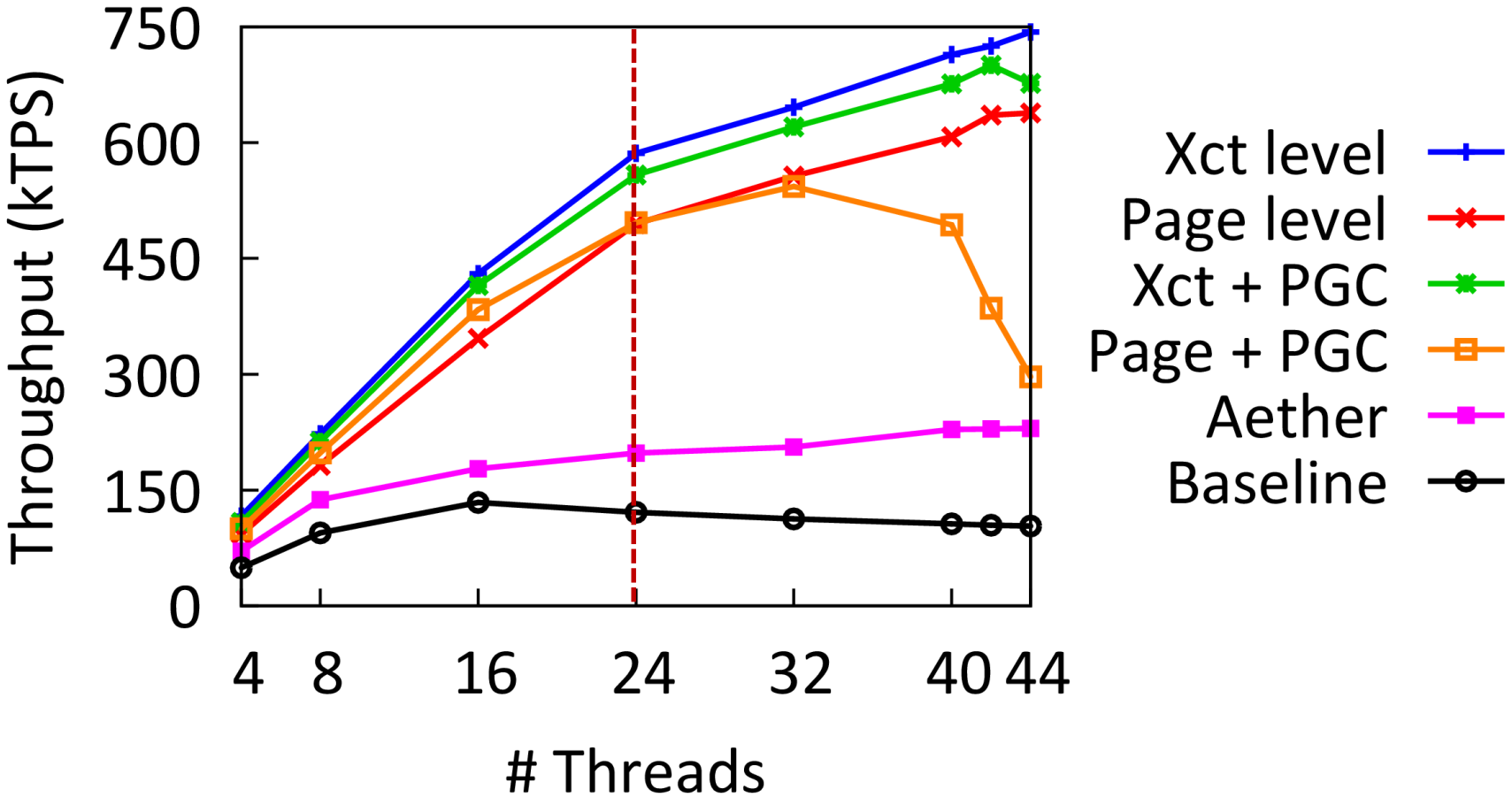
TATP – write intensive

- Distributed vs. centralized logging



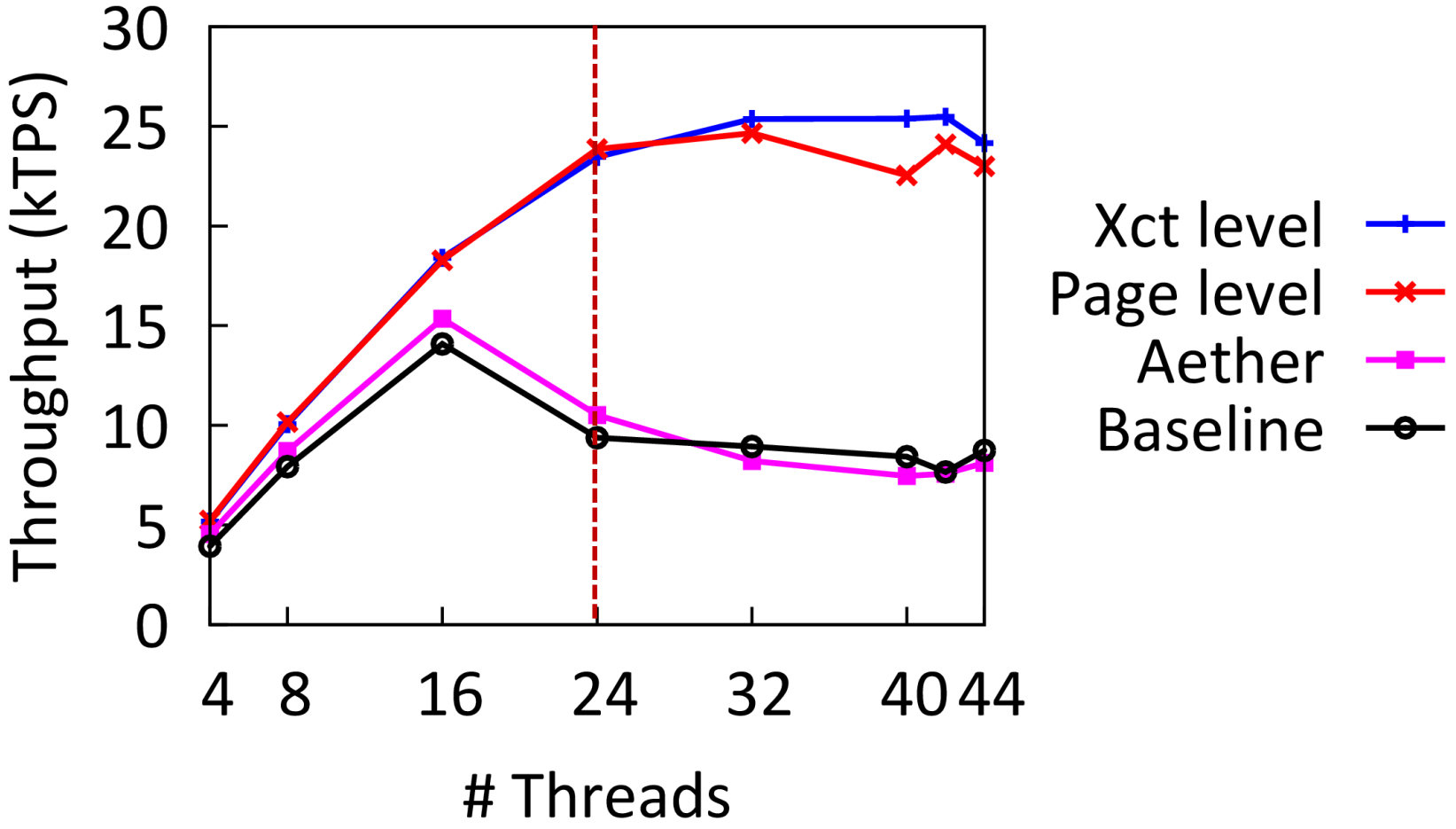
TATP – write intensive

- Passive group commit



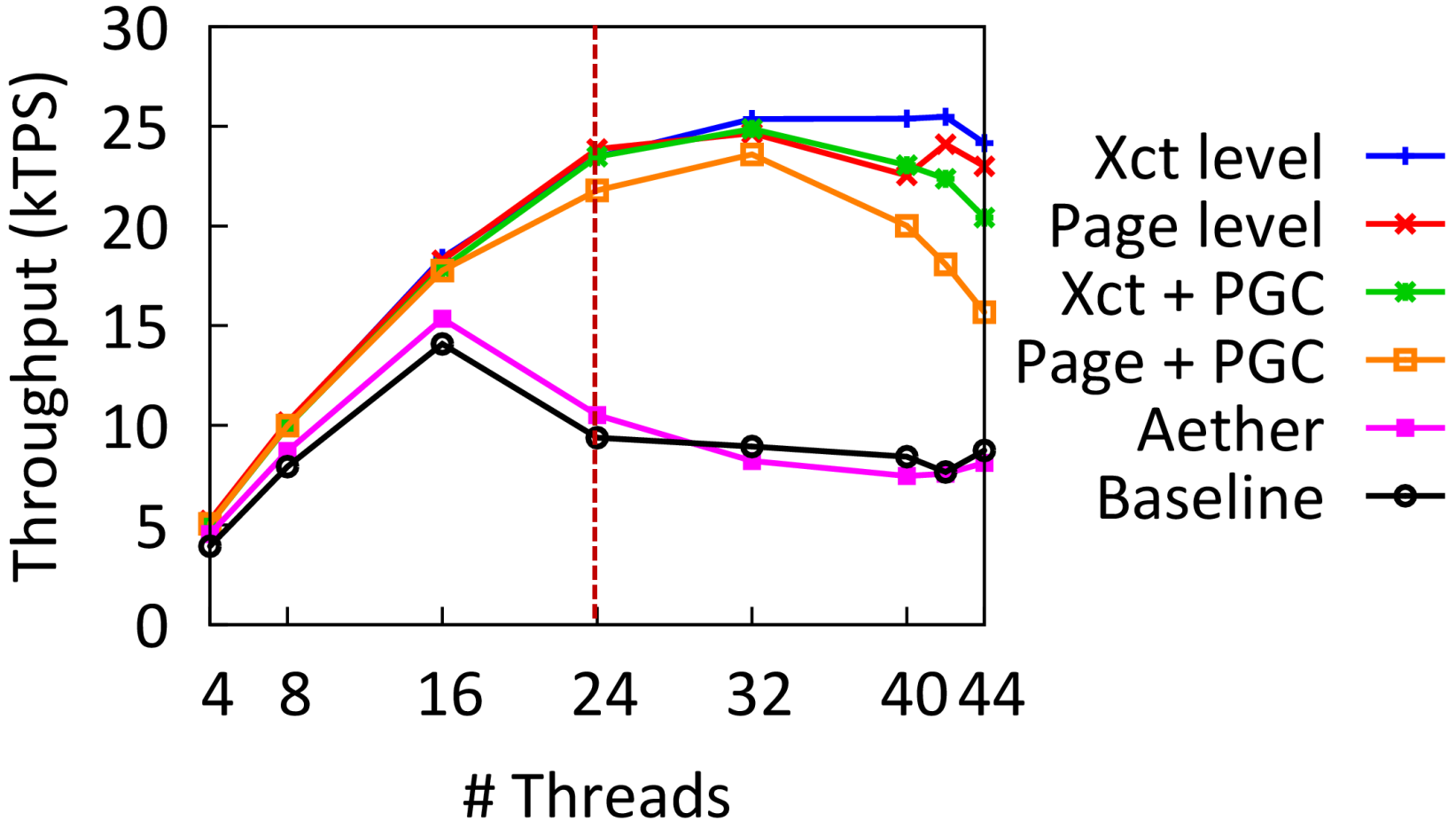
TPC-C – full transaction mix

- Distributed vs. centralized logging



TPC-C – full transaction mix

- Passive group commit



Conclusion

- Centralized logging is a serious bottleneck
- NVRAM resurrects d-log to scale databases
- Practical distributed log ***today***
  - Passive group commit
  - Flash/super-capacitor backed DRAM (NVDIMM)

Find out more in our VLDB paper:

Scalable Logging through Emerging Non-Volatile Memory

http://www.vldb.org/pvldb/vol7/p865-wang.pdf

Thank you!