

# Transaction Logging Unleashed with NVRAM

Tianzheng Wang      Ryan Johnson



UNIVERSITY OF  
**TORONTO**

# No, I'm not talking about the syslog

```
[ 3.309626] [drm:cpt_set_fifo_underrun_reporting] *ERROR* uncleaned pch fifo underrun on pch transcoder A
[ 3.309627] [drm:cpt_serr_int_handler] *ERROR* PCH transcoder A FIFO underrun
[ 3.318433] iwlwifi 0000:03:00.0: L1 Enabled - LTR Disabled
[ 3.325377] iwlwifi 0000:03:00.0: Radio type=0x1-0x2-0x0
[ 3.601724] iwlwifi 0000:03:00.0: L1 Enabled - LTR Disabled
[ 3.608680] iwlwifi 0000:03:00.0: Radio type=0x1-0x2-0x0
[ 3.816624] e1000e 0000:00:19.0: irq 29 for MSI/MSI-X
[ 3.853617] Console: switching to colour frame buffer device 170x48
[ 3.857624] i915 0000:00:02.0: fb0: inteldrmfb frame buffer device
[ 3.857626] i915 0000:00:02.0: registered panic notifier
[ 3.917200] e1000e 0000:00:19.0: irq 29 for MSI/MSI-X
[ 4.669589] cfg80211: Calling CRDA to update world regulatory domain
[ 4.744350] cfg80211: Calling CRDA to update world regulatory domain
[ 4.843255] cfg80211: Calling CRDA to update world regulatory domain
[ 5.734478] random: nonblocking pool is initialized
[ 6.746837] cfg80211: Calling CRDA to update world regulatory domain
[ 6.943294] cfg80211: Calling CRDA to update world regulatory domain
[ 7.216206] wlp3s0: authenticate with ec:88:8f:23:f2:80 (capab=0x31)
[ 7.217838] wlp3s0: send auth to ec:88:8f:23:f2:80 (capab=0x31)
[ 7.220438] wlp3s0: authenticated
[ 7.220586] iwlwifi 0000:03:00.0 wlp3s0: disabled to WEP/TKIP use
[ 7.221327] wlp3s0: associate with ec:88:8f:23:f2:80 (capab=0x31)
[ 7.226971] wlp3s0: RX AssocResp from ec:88:8f:23:f2:80 (capab=0x31 status=0 aid=1)
[ 7.232019] wlp3s0: associated
[ 7.232100] cfg80211: Calling CRDA to update world regulatory domain
[ 25.982046] systemd-journald[160]: File /var/log/journal/6f2bb100-ec42d485a0337320399845/user-1000.journal c
[ 27.439588] fuse init (API version 7.23)
[ 50.199436] tun: Universal TUN/TAP device driver, 1.6
[ 50.199443] tun: (C) 1999-2004 Max Krasnyansky <maxk@qualcomm.com>
[ 2005.927649] cfg80211: Calling CRDA to update world regulatory domain
[ 3266.976748] capability: warning: `VirtualBox' uses 32-bit capabilities (legacy support in use)
[ 3271.625164] vboxdrv: Found 4 processor cores.
[ 3271.625484] vboxdrv: fAsync=0 offMin=0xe2 offMax=0x11e0
[ 3271.625555] vboxdrv: TSC mode is 'synchronous', kernel timer mode is 'normal'.
[ 3271.625557] vboxdrv: Successfully loaded version 4.3.20_OSE (interface 0x001a0008).
[14880.309571] perf interrupt took too long (2521 > 2495), lowering kernel.perf_event_max_sample_rate to 50100
[44736.638945] cfg80211: Calling CRDA to update world regulatory domain
```

# 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



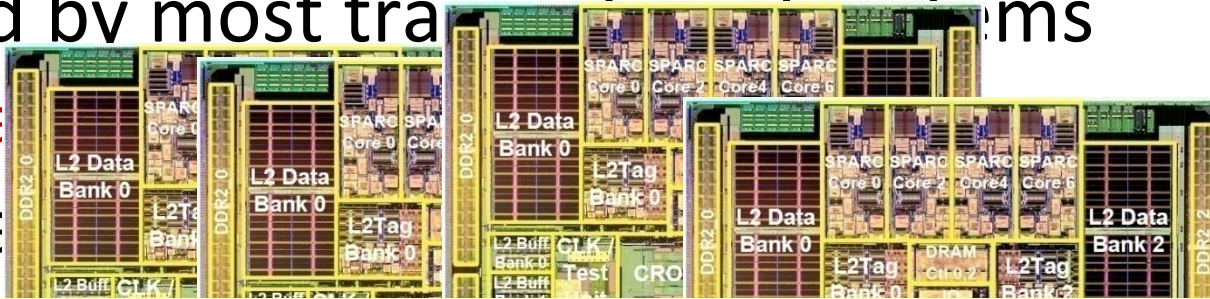
- Performance
  - Buffer log records in DRAM
  - Disk/storage friendly long, sequential writes

# Write-ahead logging

- Used by most transactional systems

- Data

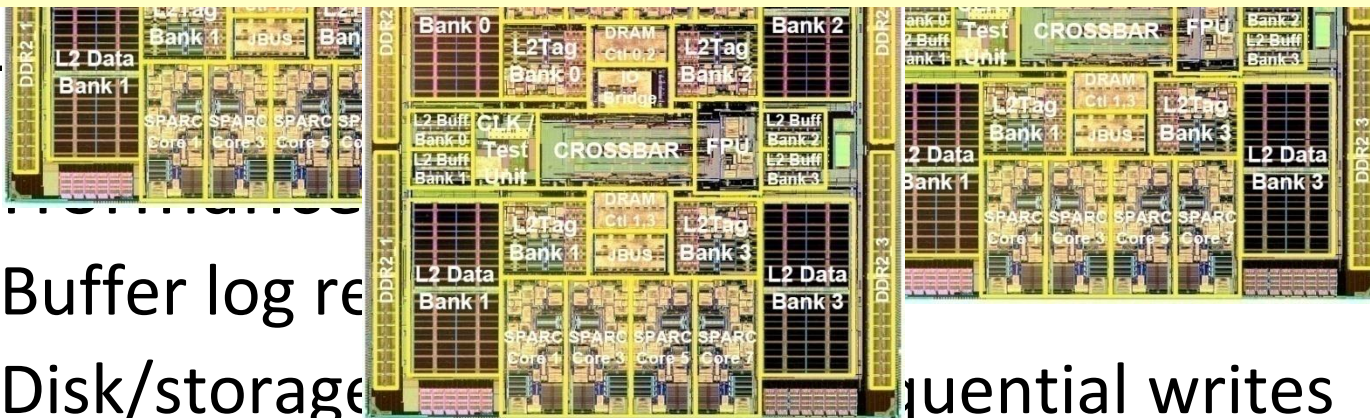
- Reliability



All was good until we had  
massively parallel hardware

- Performance

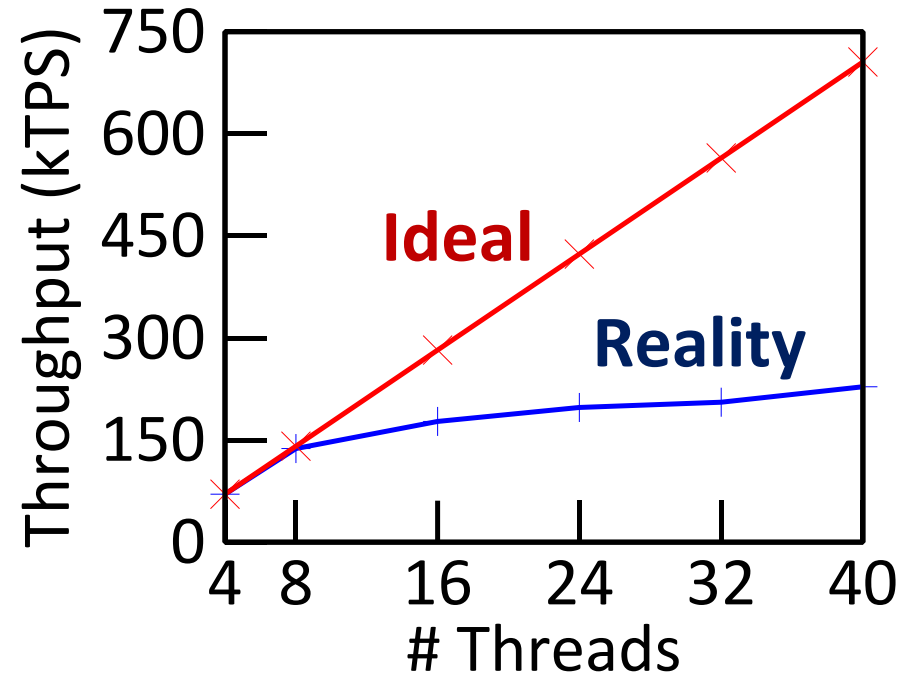
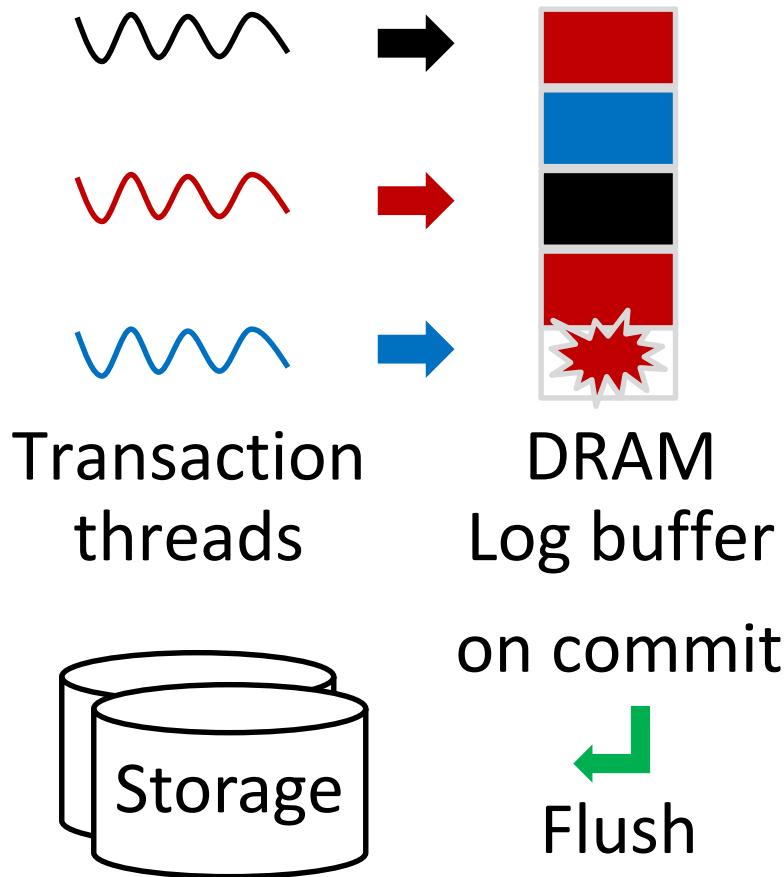
- Buffer log re...
  - Disk/storage



change it

sequential writes

# Centralized log: a serious bottleneck



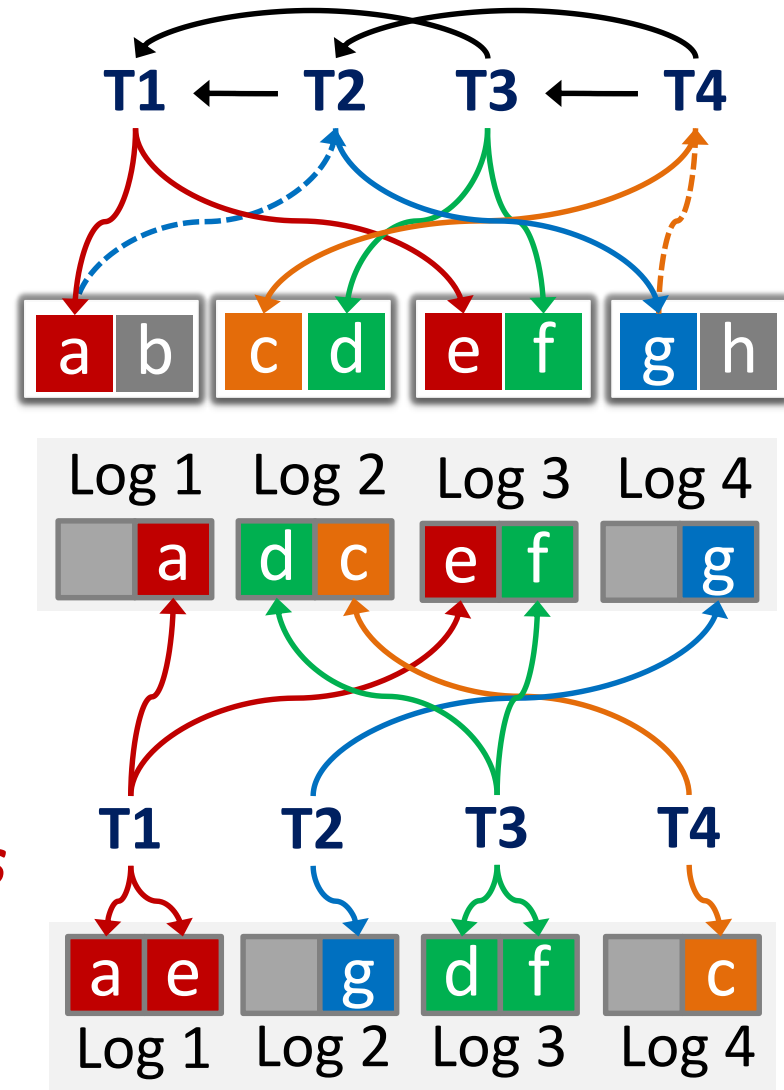
**Why not distribute the log?**

**Sure!**

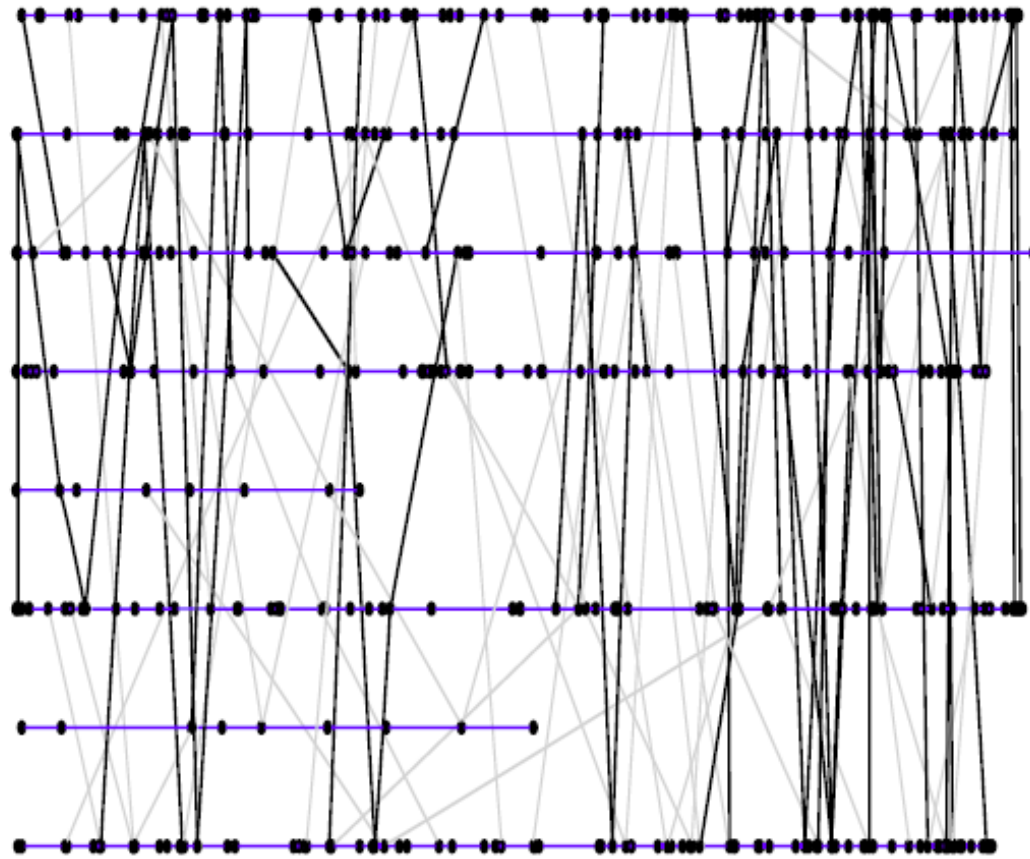
**But need the help of  
byte-addressable, non-volatile memory (NVRAM).**

# The (impractical) distributed log

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



# The (impractical) distributed log

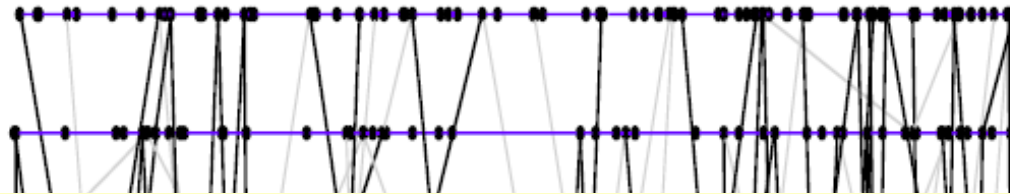


**Figure 13.** Inter-log dependencies for 1ms of TPC-C (8 logs, ~100kB, ~30 commits).

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



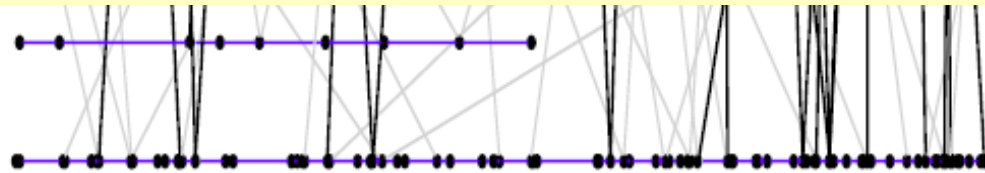
# The (impractical) distributed log



**Heavy dep. tracking + slow I/O**

**=**

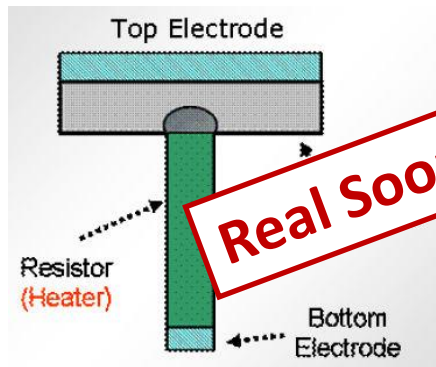
**showstoppers**



**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

# NVRAM to the rescue

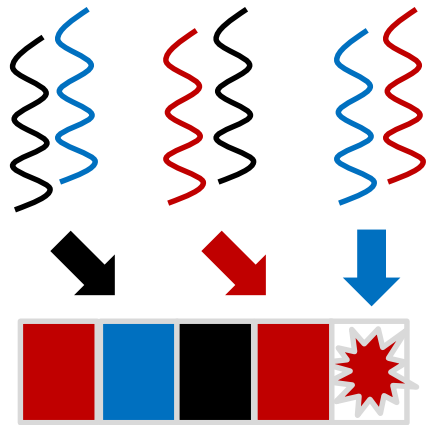


- NVRAM as *log buffers* for distributed logging
  - Log records durable once written
  - No dep tracking or flush-before-commit

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

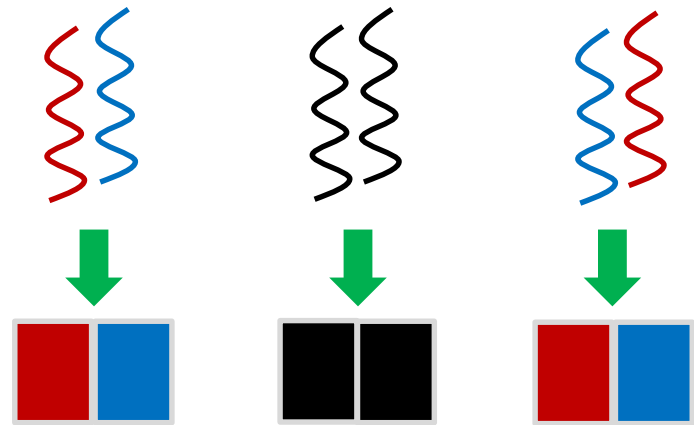
# System architecture

Before:



Log buffer (DRAM)

After:



Log buffers (NVRAM)

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

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

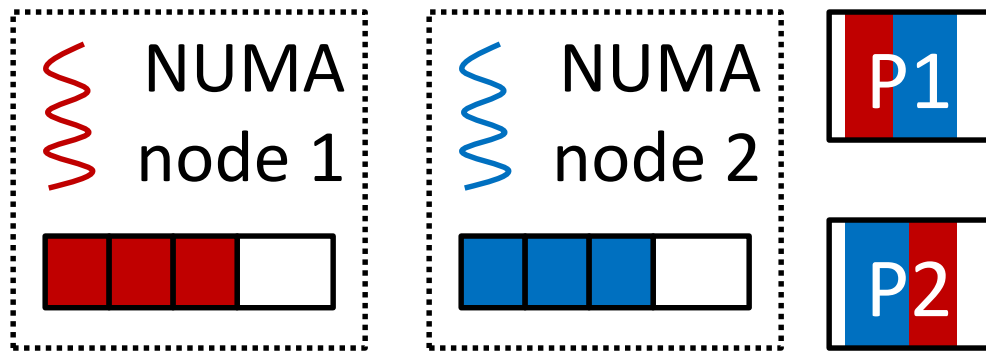
# Challenges

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

# Problem #1: NUMA effects

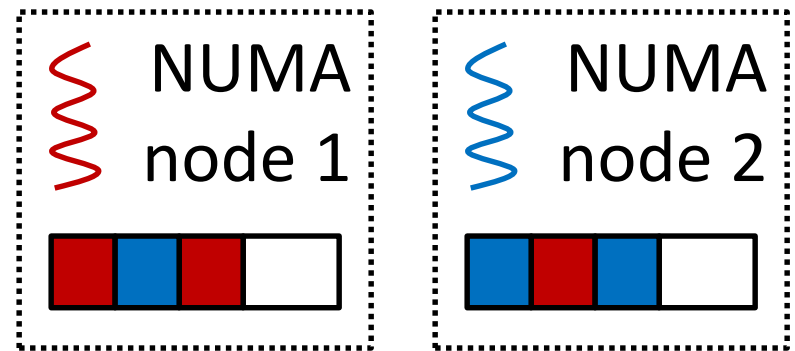
- Partition-by-page => easier/simpler recovery
- Threads prefer to access *local* NVM node

*Transaction level:*



☺ NUMA-friendly

*Page level:*



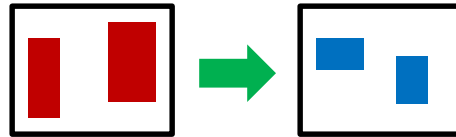
☹ Cross NUMA boundary

**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

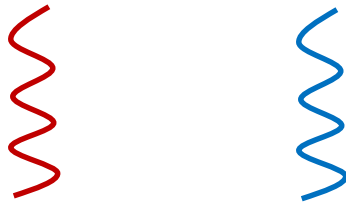
The **same** page being modified:



Recovery manager:



Transaction threads:



**Same LSNs, whom first?**

Log buffers:

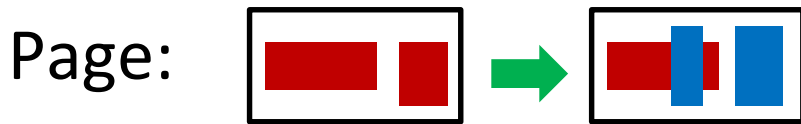


smaller  $\neq$  earlier!

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

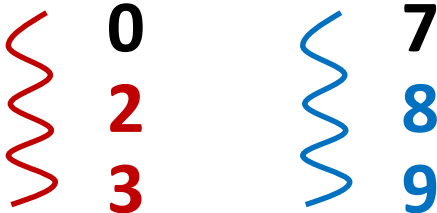
# Solution #2: global sequence number

- Based on Lamport's clock, no extra contention



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

Pg GSN: 1 – 2 – 3 3 – 8 – 9

Tx GSN: 

Log bufs: 

GSN:	Page	Transaction	Log
EX-latch	$\max(\text{pg's}, \text{tx's}) + 1$		/
SH-latch	/	$\max(\text{pg's}, \text{tx's})$	/
Log ins.	$\max(\text{pg's}, \text{tx's}, \text{log's}) + 1$		

**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.

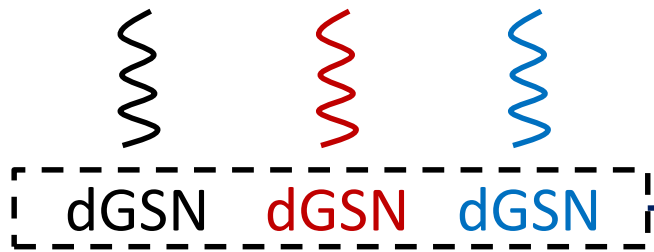


The demonstration of a breakout game



# 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  $\leq$  8

TXN	dGSN
<del>Xct 1</del>	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

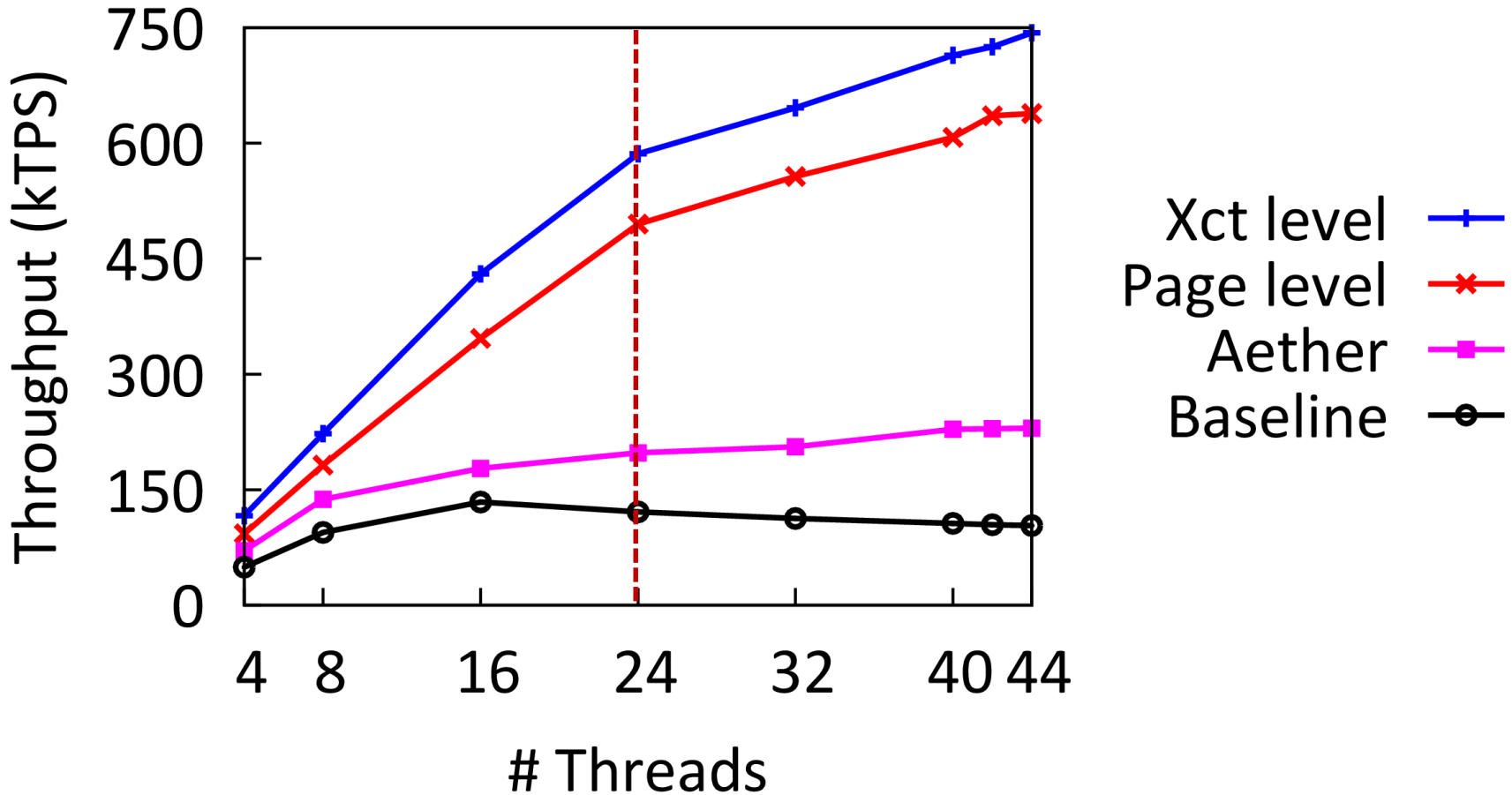
- Workloads

- Shore-MT, with Aether\*
- TPC-C: online transaction processing
- TATP: telecom database applications

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

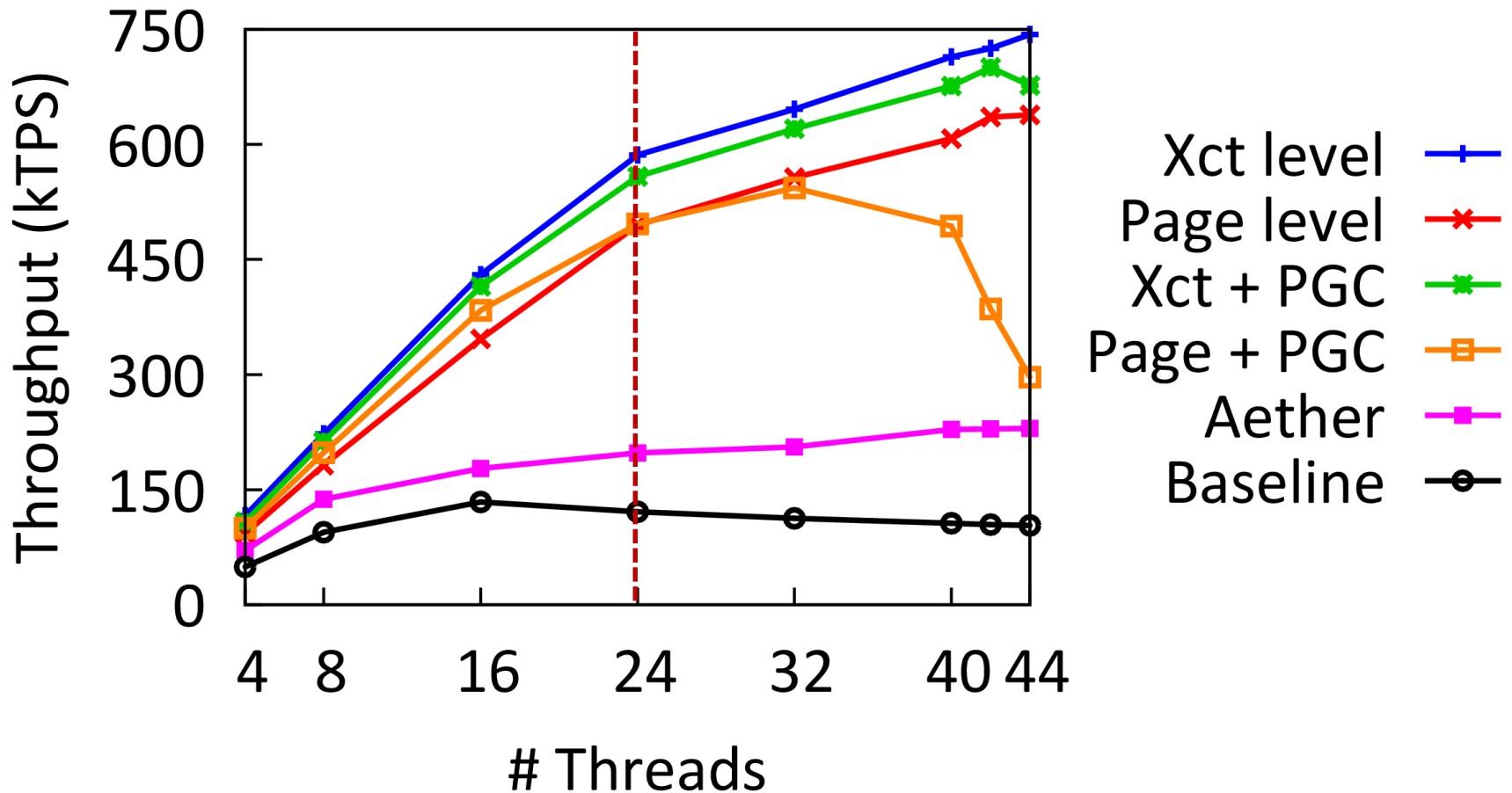
# TATP – write intensive

- Distributed vs. centralized logging



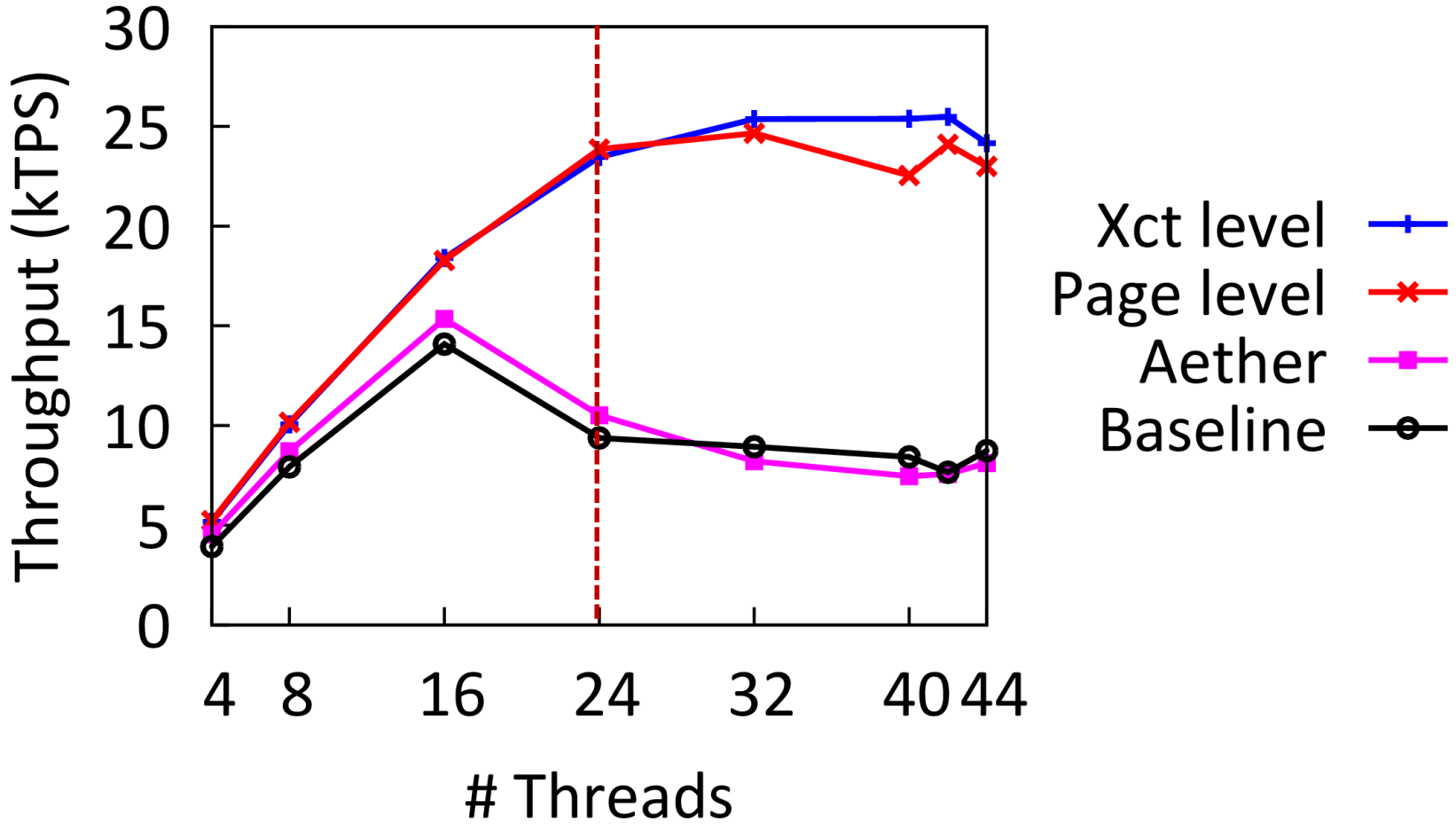
# TATP – write intensive

- Passive group commit



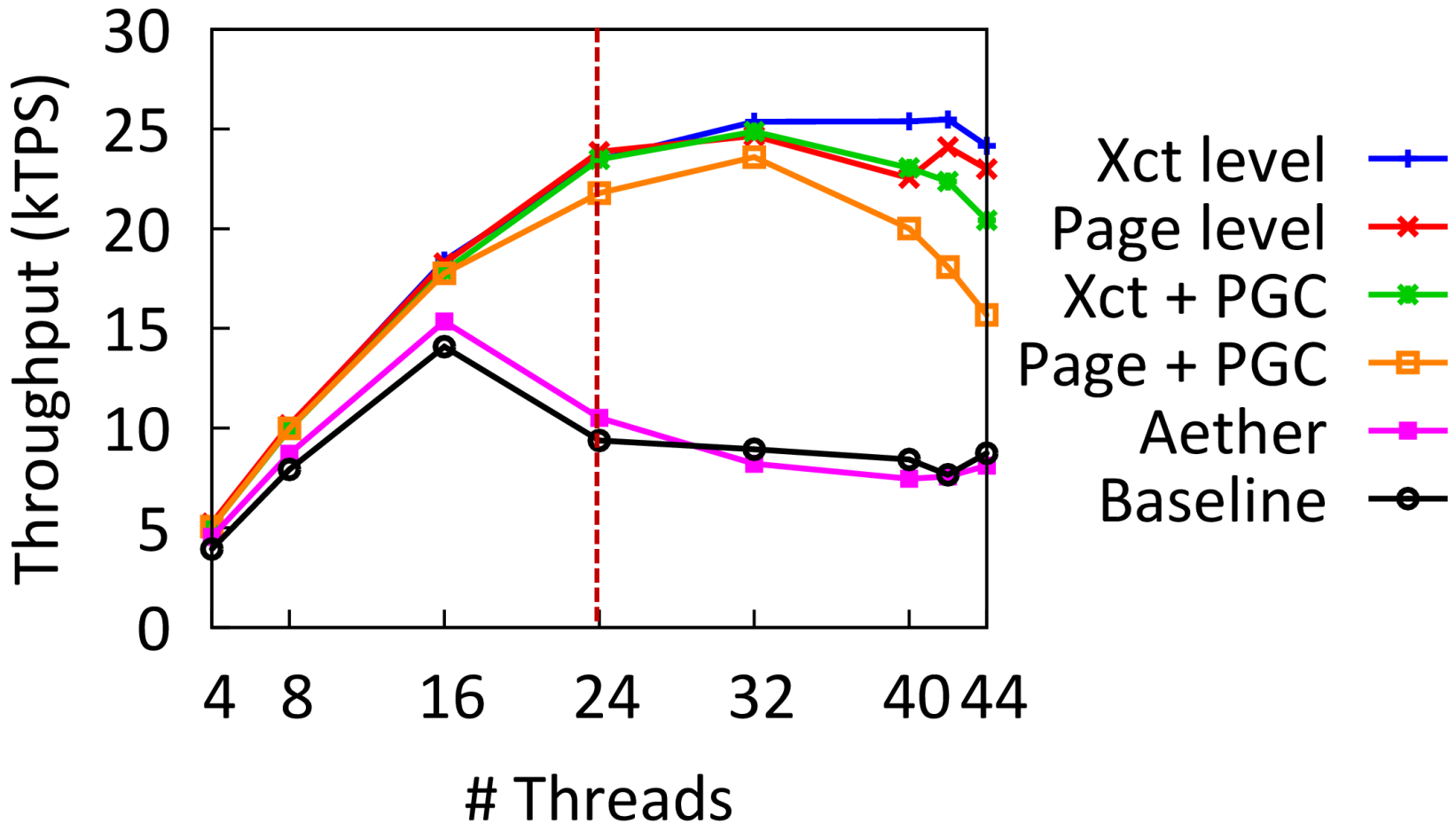
# TPC-C – full transaction mix

- 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!*