In-memory Data Management Systems — Challenges and Opportunities

Zhang Hao
zhangh@comp.nus.edu.sg
National University of Singapore

23 Feb. 2016@RUC

Outline

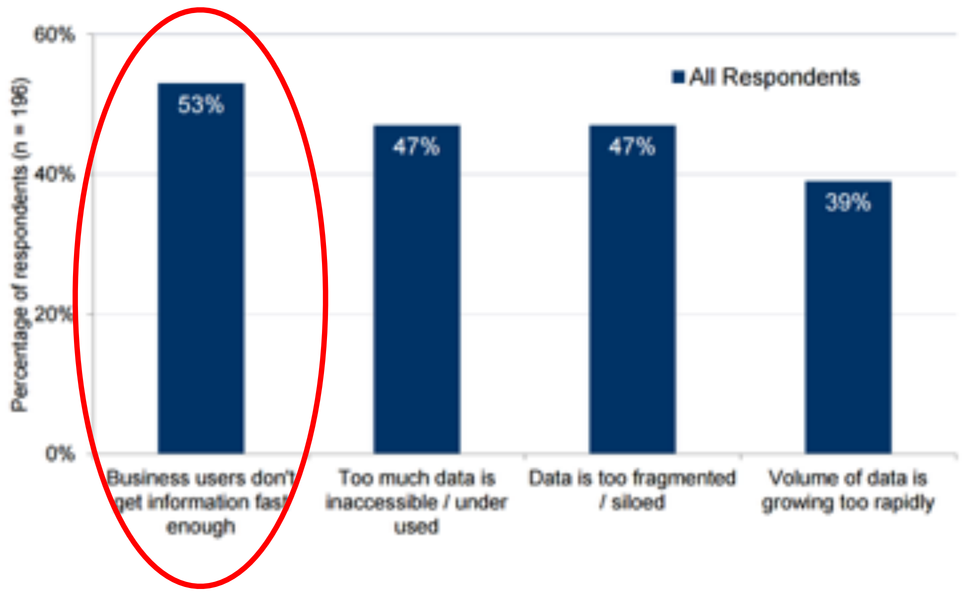
- Introduction
  - why in-memory?
- Hareware Innovation
  - NUMA
  - HTM
  - RDMA
- System Calls
- Anti-caching

Outline

- Introduction
  - why in-memory?
- Hareware Innovation
  - NUMA
  - HTM
  - RDMA
- System Calls
- Anti-caching

“Memory is the new disk, disk is the new tape”

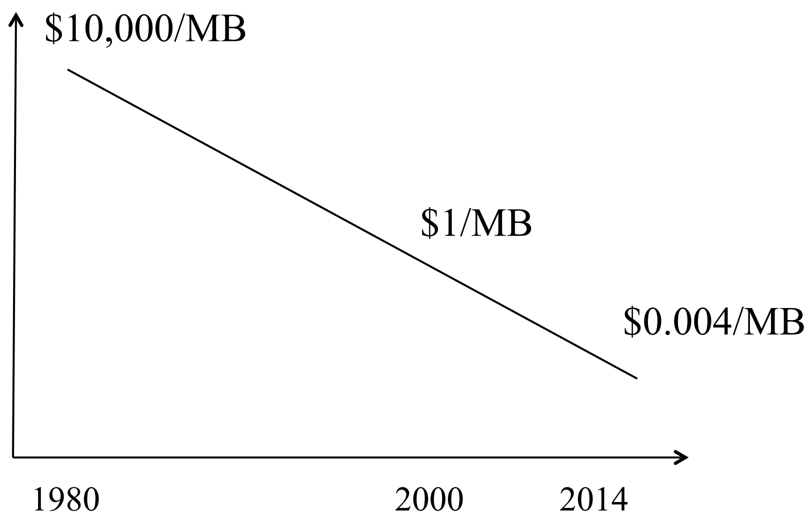
- **SPEED** is **EVERYTHING** in business
- Low-latency and Real-time
  - Disk I/O KILLs everything



Pain Points of Big Data
(Source: Aberdeen Group Survey)

“Memory is the new disk, disk is the new tape”

- DRAM becomes **BIGGER** and **CHEAPER**
- CPU is much **STRONGER**



Multi-Core Architecture (8 x 8core CPU per blade)

Massive parallel scaling with many blades

One blade ~$50.000 = 1 Enterprise Class Server

64bit address space – 2TB in current servers

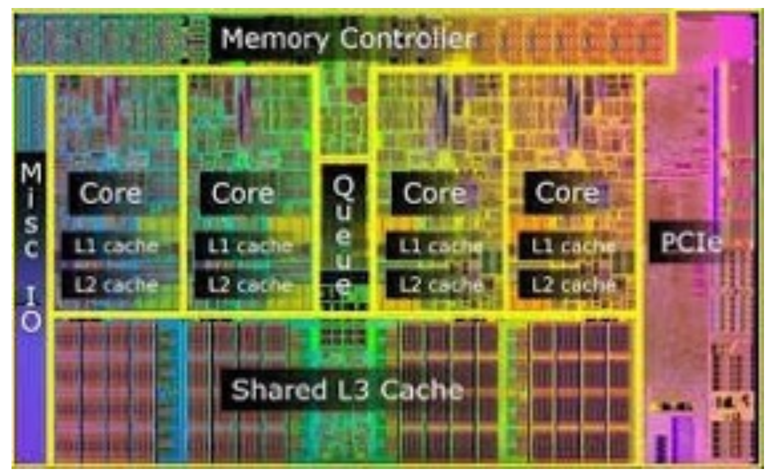
100GB/s data throughput

Dramatic decline in price/performance

Speed

CPU

Core / Core	
L1 Cache / L1 Cache	1ns
L2 Cache / L2 Cache	4ns
L3 Cache	7ns
Main Memory	100ns
NVM	200ns-1 50K
Disk	4M ns



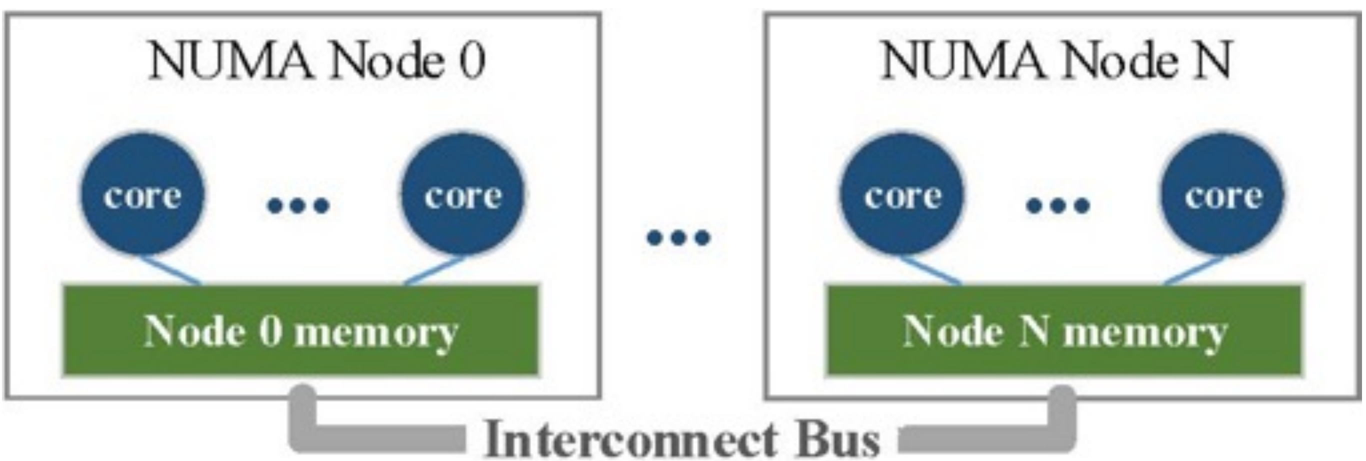
Intel Core i5

Outline

- Introduction
  - why in-memory?
- Hareware Innovation
  - NUMA
  - HTM
  - RDMA
- System Calls
- Anti-caching

HW Innovation – NUMA

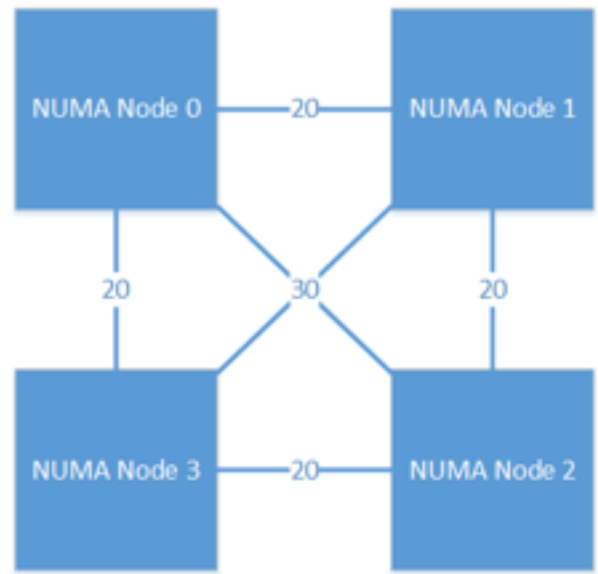
- NUMA node owns its local memory
- Different access speed to local/remote memory



HW Innovation – NUMA

Example: Topology of our epic server (likwid-topology) (epic.d1.comp.nus.edu.sg)

```
********************************************************************
NUMA Topology
********************************************************************
NUMA domains: 4
--------------------------------------------------------------------
Domain 0:
Processors:  0 4 8 12 16 20 24 28 32 36 40 44
Relative distance to nodes:  10 20 30 20
Memory: 2910.11 MB free of total 16003 MB
--------------------------------------------------------------------
Domain 1:
Processors:  1 5 9 13 17 21 25 29 33 37 41 45
Relative distance to nodes:  20 10 20 30
Memory: 10862.3 MB free of total 16126.2 MB
--------------------------------------------------------------------
Domain 2:
Processors:  2 6 10 14 18 22 26 30 34 38 42 46
Relative distance to nodes:  30 20 10 20
Memory: 12426.8 MB free of total 16126.2 MB
--------------------------------------------------------------------
Domain 3:
Processors:  3 7 11 15 19 23 27 31 35 39 43 47
Relative distance to nodes:  20 30 20 10
Memory: 13866.8 MB free of total 16125.8 MB
--------------------------------------------------------------------
```



HW Innovation – NUMA

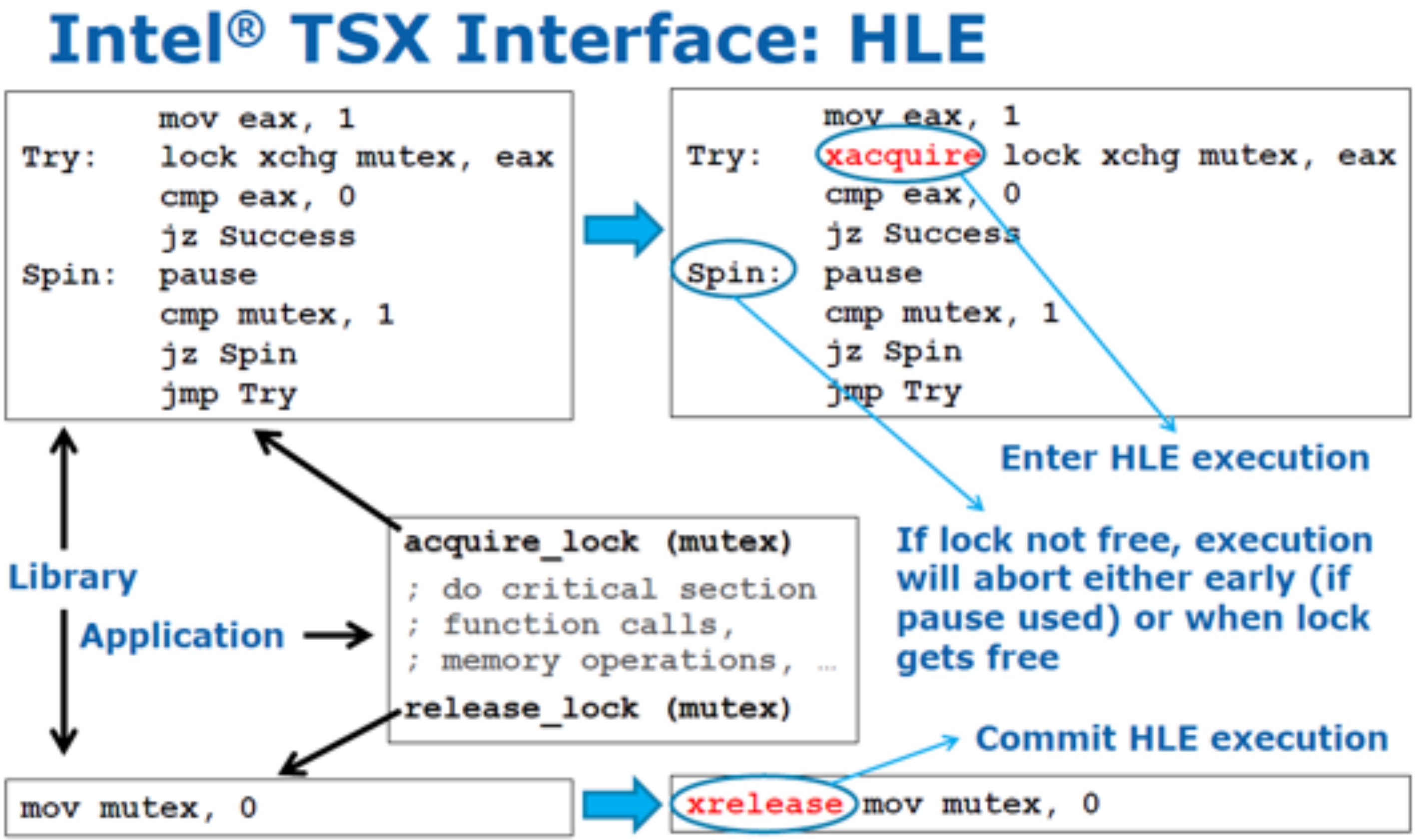
- Related works – NUMA-aware
  - shared-nothing architecture within one NUMA server, e.g., Bubba [1], Gamma [2]
  - Hardware islands with UNIX sockets [3]
  - Data shuffling: ring-shuffling [4]
  - Scheduling: marsel query execution [5]
  - Indexing: Buzzard indexing [6]
  - Specific algorithms: sort-merge join [7]

Outline

- Introduction
  - why in-memory?
- Hareware Innovation
  - NUMA
  - HTM
  - RDMA
- System Calls
- Anti-caching

HW Innovation – HTM

- Optimistic concurrency control in HW level



HW Innovation – HTM

- Limitations
  - The transaction size is limited to the size of L1 data cache.
  - Cache associativity makes it more prone to false conflicts.
  - HTM transactions may be aborted due to interrupt events.

HW Innovation – HTM

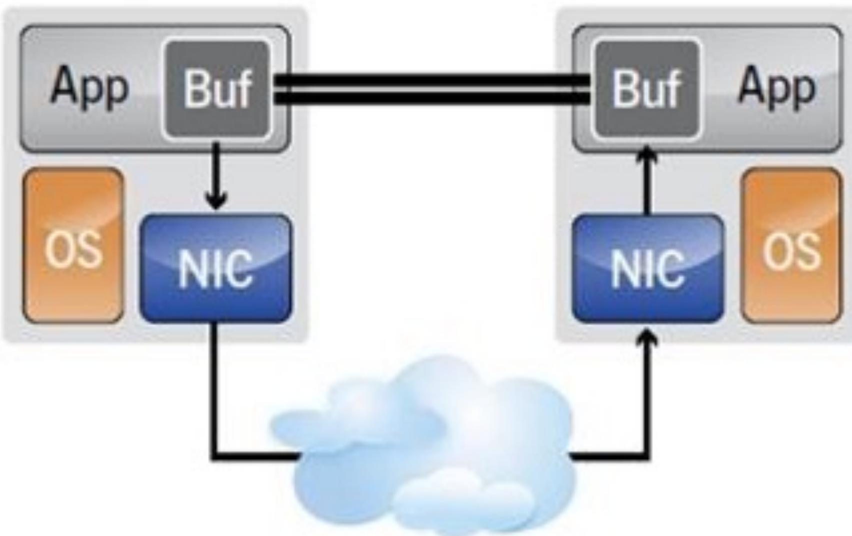
- Related works
  - A database transaction is divided into a set of relatively small HTM transactions with timestamp ordering (TSO) concurrency control and minimizing the false abort probability via data/index segmentation [8].
  - protects single data read, and validate/write phases using HTM transactions [9]

Outline

- Introduction
  - why in-memory?
- Hareware Innovation
  - NUMA
  - HTM
  - RDMA
- System Calls
- Anti-caching

HW Innovation – RDMA

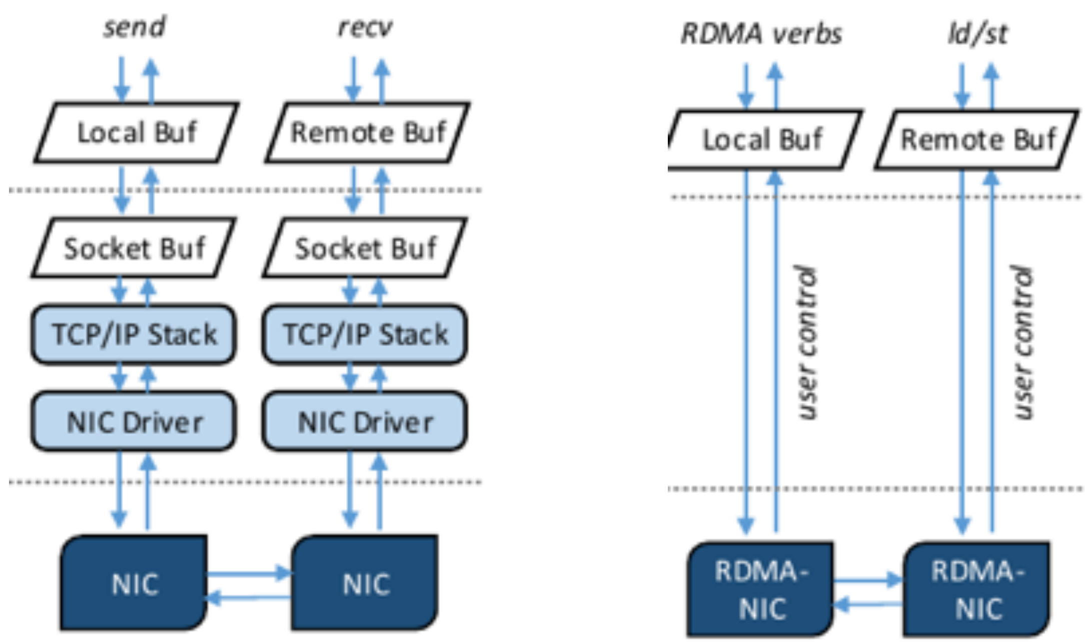
- Remote Direct Memory Access



OS or CPU is not involved!

HW Innovation – RDMA

- Comparison with Ethernet TCP/IP

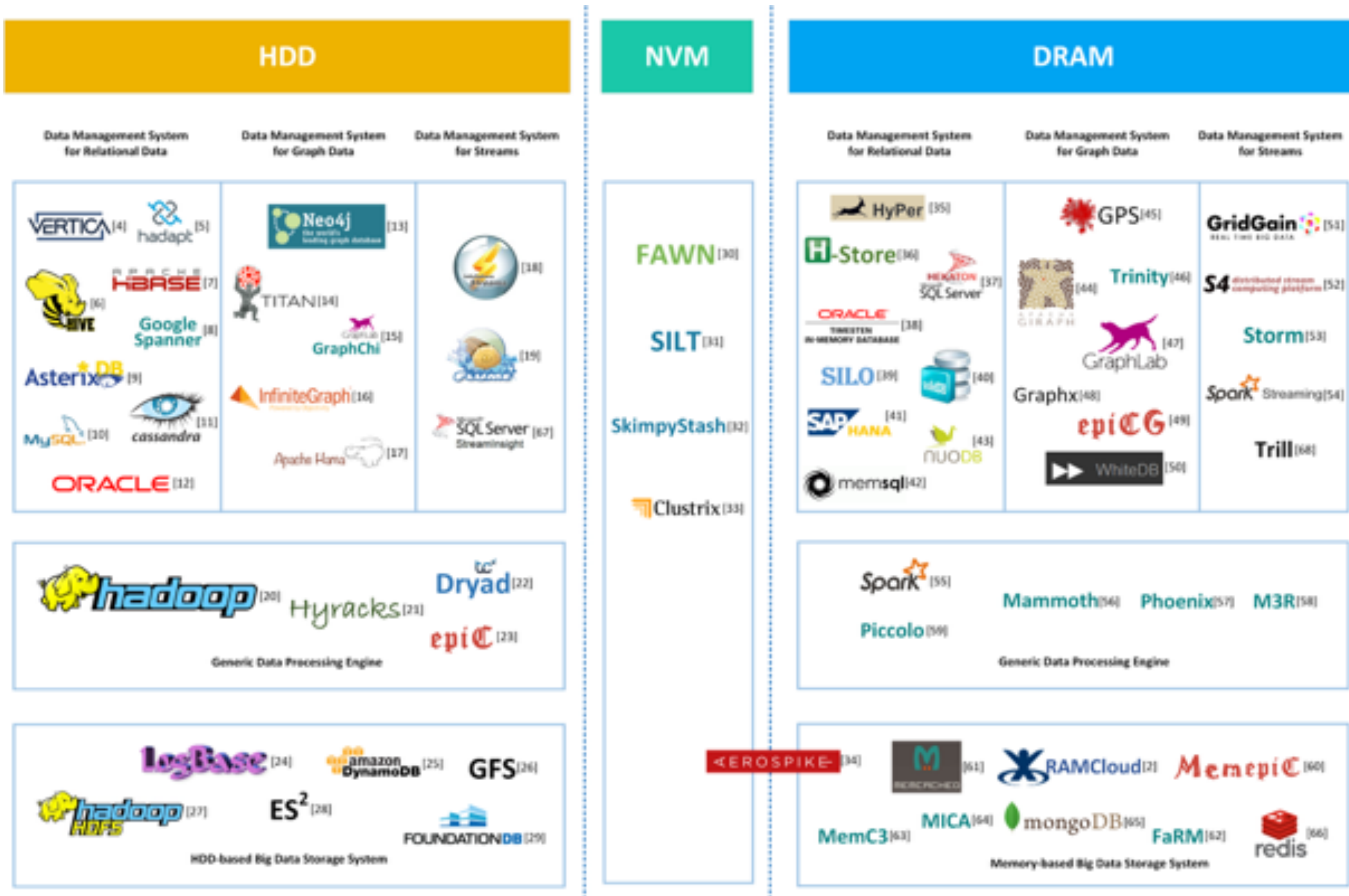


TCP/IP RDMA

HW Innovation – RDMA

- Related works
  - Pilaf [10]: multiple one-sided RDMA READ with self-verifying data structures for GET operations
  - HERD [11]: reducing latency (RDMA WRITE from client, and SEND from server); RDMA-specific features (e.g., inlining, selective signaling)

In-Memory Big Data Management and Processing: A Survey [12]



Outline

- Introduction
  - why in-memory?
- Hareware Innovation
  - NUMA
  - HTM
  - RDMA
  - NVRAM
- System Calls
- Anti-caching

Facts

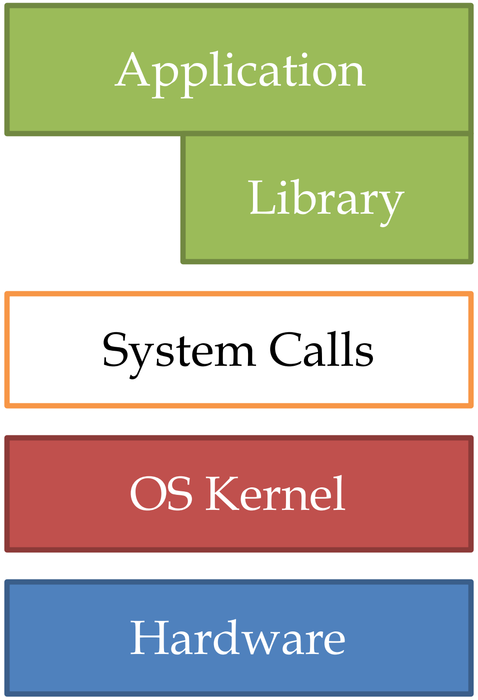
Simply moving the storage layer from disk to memory will not enable the DB to take full advantage of the memory performance.

Many reasons:

1) Pointer chasing
2) Cache unfriendly data structures
3) **System calls**
4) …

The Problem with System Calls

A system call is required every time an application requires "service" from the OS.



1. File management
2. Device management
3. Communication
4. Process control
5. Information maintenance

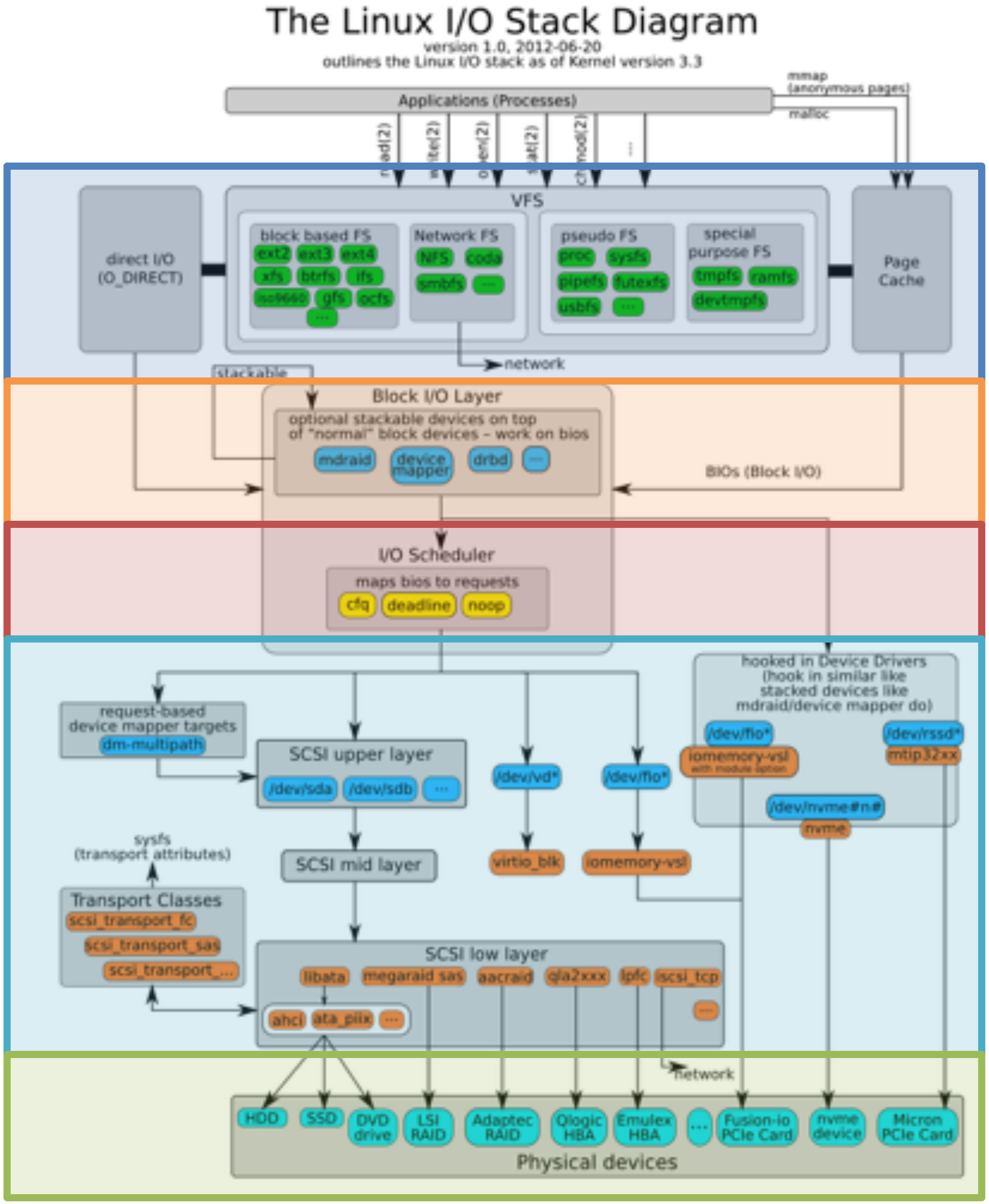
Virtual File System

Block I/O Layer

I/O Scheduler

Device Drivers

Hardware

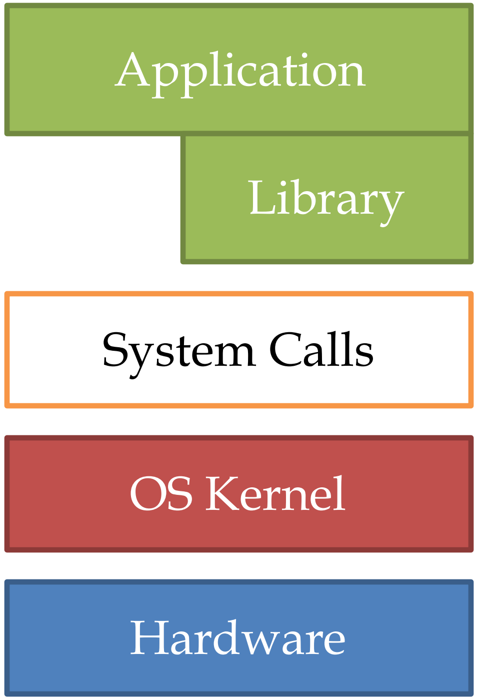


The Problem with System Calls

Syscalls have two main problems

1. Introduce latency

2. **Unsuitable abstraction for accessing memory**
   - **E.g. "read" syscall can read from a file (disk/SSD/NFS) or a network socket or arbitrary file-mapped device**



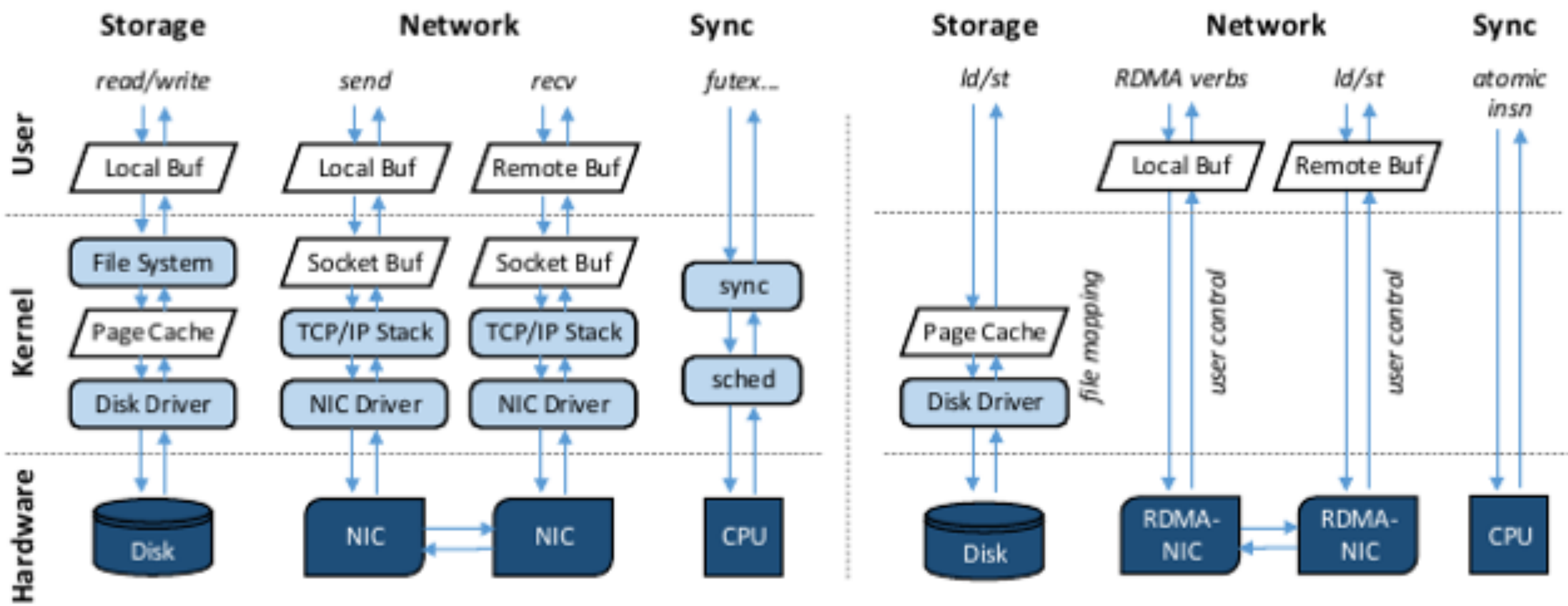
System Calls in Databases

Four sources of system calls:

1. Data accesses
   - open/close/**read**/**write**/stat …
2. Communication among workers
   - socket/listen/accept/connect/**sendmgs**/**recvmsg** …
3. Synchronization among workers
   - **pthread_mutex_lock**/**unlock**, **sem_wait**/**sem_post** …
4. Fault tolerance and recovery
   - a mixture of the above

There are methods to replace *most* system calls during the basic operation of an in-memory database.

Towards No Syscalls



Traditional DB using syscalls MemepiC with minimal syscalls

Outline

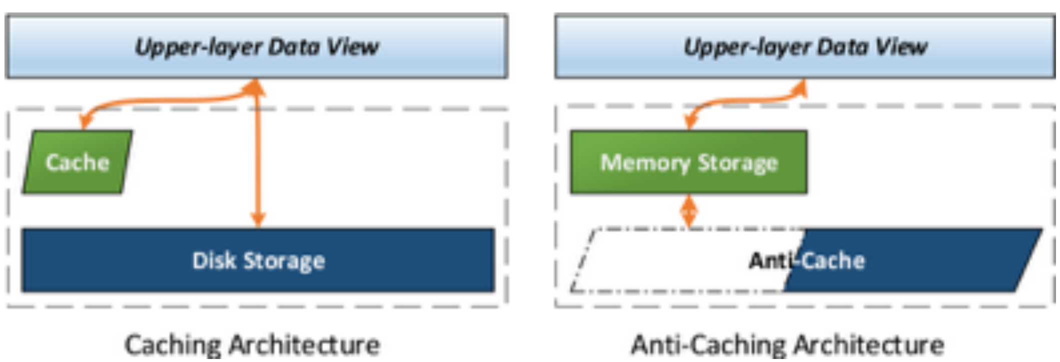
- Introduction
  - why in-memory?
- Hareware Innovation
  - NUMA
  - HTM
  - RDMA
  - NVRAM
- System Calls
- Anti-caching

Facts

- Memory never enough
  - Memory is still relatively scarce compared to HDD
  - Energy consumption
    - Memory is a significant contributor to the total system power
  - N-minute rule
    - cheaper to put the data in memory if it is accessed every N-minute
    - Cold data – stay on disk
    - Hot data – resident in memory

Caching vs. "Anti-Caching"

- Common
  - Deal with the same level of storages
- Difference
  - Assumption about the memory size
  - Different primary data locations
  - Target for different types of systems



Components of anti-/caching

- Access tracking
  - Granularity: Tuple vs page
- Eviction Strategy
  - LRU, MRU, CLOCK, WSCLOCK
- Book-keeping
  - indexes, filters, page table, etc.
- Swapping strategy
  - How much, and when

State-of-the-art Approaches

Approaches	Access Tracking	Eviction Strategy	Book-keeping	Data Swapping
H-Store anti-caching	Tuple-level tracking	LRU	Evicted table and index	Block-level swapping
Hekaton Siberia	Tuple-level access logging	Offline classification	Bloom and range filter	Tuple-level migration
Spark	N/A	LRU based on insertion time	Hash table	Block-level swapping
Cache Systems	Tuple-level tracking	LRU, approximate LRU, etc	N/A	N/A
Buffer Management	Page-level tracking	LRU, MRU, CLOCK, etc	Hash table	Page-level swapping
OS Paging	h/w-assisted page-level tracking	LRU, NRU, WSCLOCK, PPRA, etc	Page table	Page-level swapping
Efficient OS Paging	Tuple-level access logging	Offline classification and OS Paging	OS-dependent	OS-dependent
Access Observer in Hyper	h/w-assisted page-level tracking Memory protection	N/A	N/A	N/A

Access tracking - insights

- If the average tuple size is less than 4-KB for doubly-linked LRU list, their memory overheads are much higher than that of page-table-based method.

Eviction strategy - insights

- OS-based eviction approaches suffer from poor accuracy
  - Coarser-granularity
  - Lack of semantics information
- Access-logging based offline classification do well

Book-keeping - insights

- Index and eviction table: higher space overhead
- Bloom and other filters: quite space efficient
- Page table: hardware support

Swapping - insights

- Block/page-level swapping is efficient in terms of disk I/O throughput

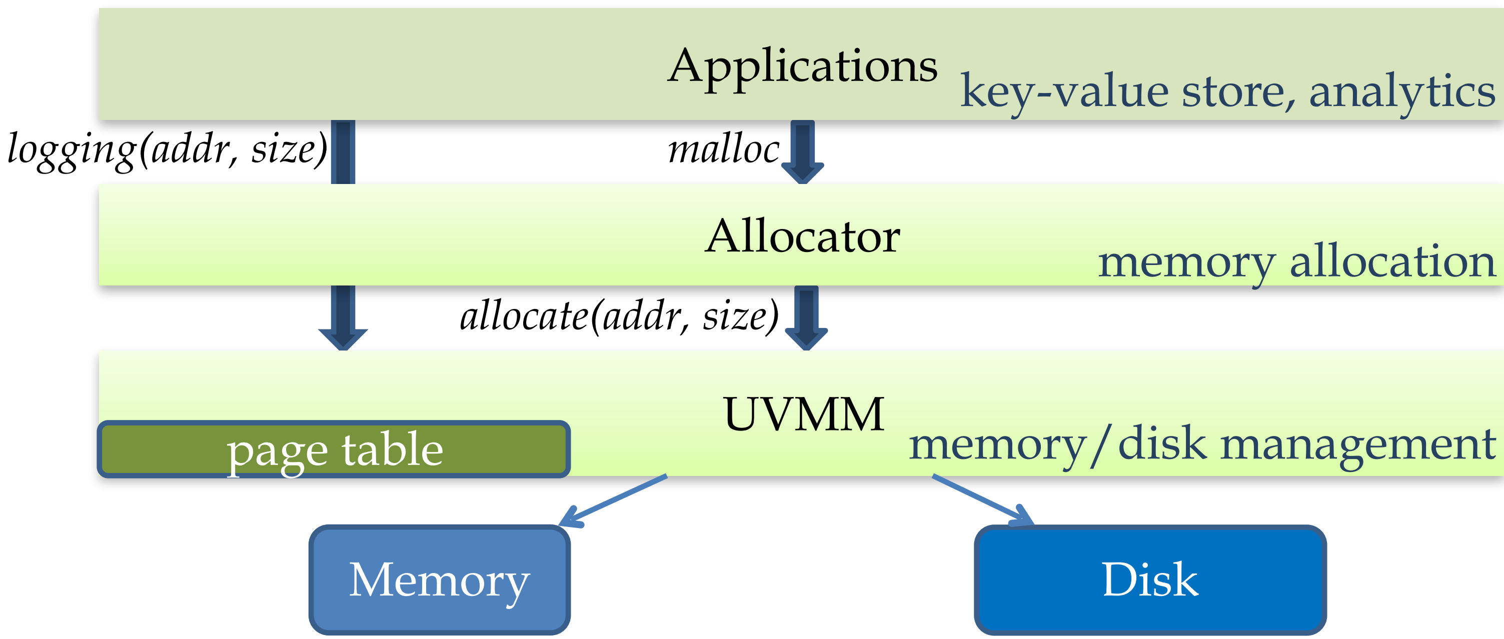
User-space vs kernel-space

- At user/application level
  - More semantics information
  - Flexible granularities (tuple, column, row, tables, page)
  - Platform-independence (possible)
- At kernel level
  - Directly use **hardware**
  - General
  - Only know **pages**

Towards An Efficient General Approach

– User-space Virtual Memory Management (UVMM) [13]

- Three-layer Hierarchy



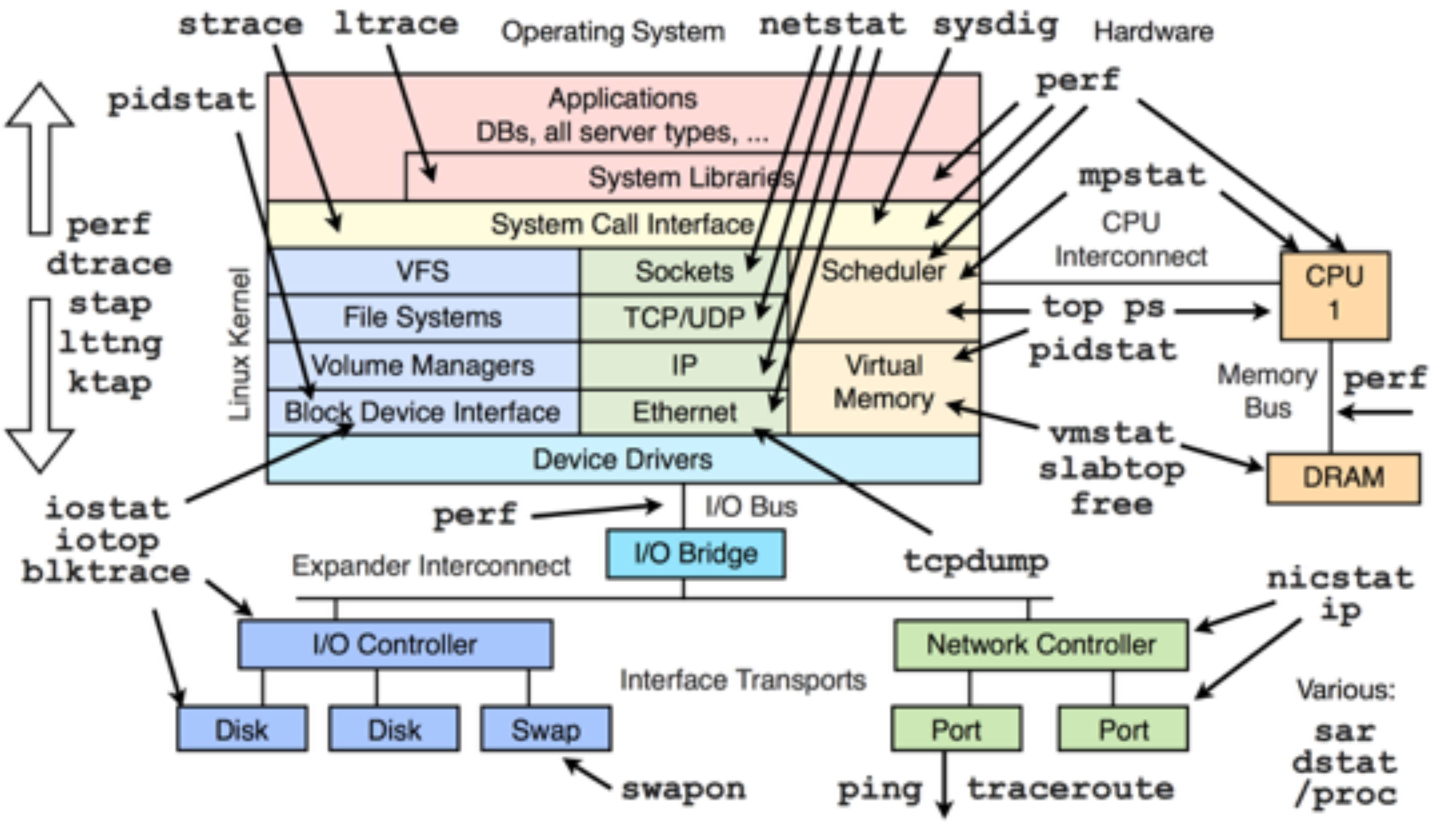
From Jeff Dean (2012)

Numbers Everyone Should Know

L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns
Mutex lock/unlock	25 ns
Main memory reference	100 ns
Compress 1K bytes with Zippy	3,000 ns
Send 2K bytes over 1 Gbps network	20,000 ns
Read 1 MB sequentially from memory	250,000 ns
Round trip within same datacenter	500,000 ns
Disk seek	10,000,000 ns
Read 1 MB sequentially from disk	20,000,000 ns
Send packet CA->Netherlands->CA	150,000,000 ns

Useful Linux Tools

- Measure lower-level numbers



Resources

- K.-L. Tan, Q. Cai, B. C. Ooi, W.-F. Wong, C. Yao, H. Zhang: In-memory Databases – Challenges and Opportunities -- From Software and Hardware Perspectives. ACM SIGMOD Record, Special Issue on Visionary Ideas in Data Management, Vol. 44, No. 2, 35 – 40, June 2015.
- H. Zhang, G. Chen, B. C. Ooi, K. L. Tan, M. Zhang: In-Memory Big Data Management and Processing: A Survey. IEEE Transactions on Knowledge and Data Engineering (TKDE), Vol. 27, No. 7, 1920 – 1948, July 2015.
- D. Loghin, B. M. Tudor, H. Zhang, B. C. Ooi, Y. M. Teo: A Performance Study of Big Data on Small Nodes. 41st Int'l Conference on Very Large Data Bases (VLDB), 762 – 773, 2015.
- H. Zhang, G. Chen, B. C. Ooi, W.-F. Wong, S. Wu, Y. Xia: "Anti-Caching"-based Elastic Memory Management for Big Data. 31st IEEE International Conference on Data Engineering (ICDE), 1268 – 1279, 2015.
- H. Zhang, et al., "Memepic: Towards a database system architecture without system calls," NUS Technical Report, 2014.
- H. Zhang, B. M. Tudor, G. Chen, B. C. Ooi: Efficient In-memory Data Management: An Analysis. 40th Int'l Conference on Very Large Data Bases (VLDB), 833 – 836, 2014.

- Slides: http://www.comp.nus.edu.sg/~a0095627

References

[1] H. Boral, W. Alexander, L. Clay, G. P. Copeland, S. Danforth, M. J. Franklin, B. E. Hart, M. G. Smith, and P. Valduriez, "Prototyping bubba, a highly parallel database system," IEEE Trans. Knowl. Data Eng., vol. 2, no. 1, pp. 4–24, Mar. 1990.
[2] D. J. Dewitt, S. Ghandeharizadeh, D. A. Schneider, A. Bricker, H.-I. Hsiao, and R. Rasmussen, "The gamma database machine project," IEEE Trans. Knowl. Data Eng., vol. 2, no. 1, pp. 44–62, Mar. 1990.
[3] D. Porobic, E. Liarou, P. Tozun, and A. Ailamaki, "ATraPos: Adaptive transaction processing on hardware islands," in Proc. IEEE 30th Int. Conf. Data Eng., 2014, pp. 688–699.
[4] Y. Li, I. Pandis, R. Muller, V. Raman, and G. M. Lohman, "Numa-aware algorithms: the case of data shuffling," in Proc. CIDR, 2013.
[5] V. Leis, P. Boncz, A. Kemper, and T. Neumann, "Morsel-driven parallelism: A NUMA-aware query evaluation framework for the many-core age," in Proc. ACM SIGMOD Int. Conf. Manag. Data, 2014, pp. 743–754.
[6] L. M. Maas, T. Kissinger, D. Habich, and W. Lehner, "BUZZARD: A NUMA-aware in-memory indexing system," in Proc. ACM SIGMOD Int. Conf. Manag. Data, 2013, pp. 1285–1286.
[7] M.-C. Albutiu, A. Kemper, and T. Neumann, "Massively parallel sort-merge joins in main memory multi-core database systems," Proc. VLDB Endowment, vol. 5, pp. 1064–1075, 2012.
[8] V. Leis, A. Kemper, and T. Neumann, "Exploiting hardware transactional memory in main-memory databases," in Proc. Int. Conf. Data Eng., 2014, pp. 580–591.
[9] Z. Wang, H. Qian, J. Li, and H. Chen, "Using restricted transactional memory to build a scalable in-memory database," in Proc. 9th Eur. Conf. Comput. Syst., 2014, pp. 26:1–26:15.
[10] C. Mitchell, Y. Geng, and J. Li. Using one-sided rdma reads to build a fast, cpu-efficient key-value store. In USENIX ATC '13, pages 103–114, 2013.
[11] A. Kalia, M. Kaminsky, and D. G. Andersen. Using rdma efficiently for key-value services. In SIGCOMM '14, pages 295–306, 2014.
[12] H. Zhang, G. Chen, B. C. Ooi, K. L. Tan, M. Zhang: In-Memory Big Data Management and Processing: A Survey. IEEE Transactions on Knowledge and Data Engineering, Vol. 27, No. 7, 1920-1948, July 2015.
[13] H. Zhang, G. Chen, B. C. Ooi, W.-F. Wong, S. Wu, Y. Xia: "Anti-Caching"-based Elastic Memory Management for Big Data. 31st IEEE International Conference on Data Engineering (ICDE), 1268 – 1279, 2015.

Thanks

Q/A