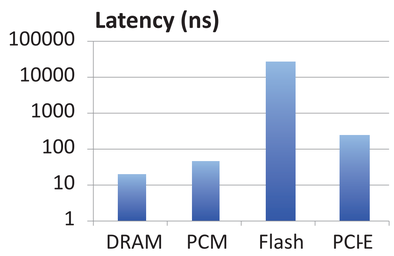
PERSISTENT, PROTECTED AND CACHED: BUILDING BLOCKS FOR MAIN MEMORY DATA STORES

Iulian Moraru, David G. Andersen, Michael Kaminsky, Niraj Tolia, Nathan Binkert, Reinhard Munz

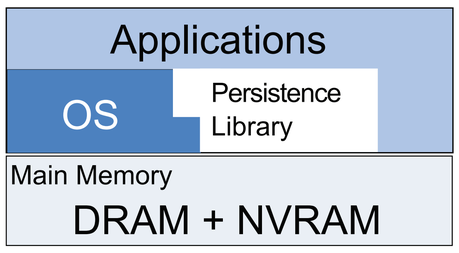
NVRAM

	DRAM	PCM	Flash
Read Latency	20-50 ns	50-100 ns	26 µs
Write Latency	20-50 ns	150-200 ns	26 µs
Read bandwidth	2 GB/s/die	1 GB/s/die	65 MB/s/die
Write bandwidth	2 GB/s/die	50-100 MB/s/die	45 MB/s/die
Write cycles	> 10^{15}	10^{8}	10^{6}

WHY THE MEMORY BUS?



NVRAM INTERFACE



CHALLENGE

- Fundamental challenge:
  - High throughput and strong durability guarantees, at the same time
- Prevent data loss
- Maintain "main memory" benefits

SOURCES OF DATA LOSS

- Wear-out
- Erroneous writes
- Incomplete updates
  - Power failures
  - Application crashes

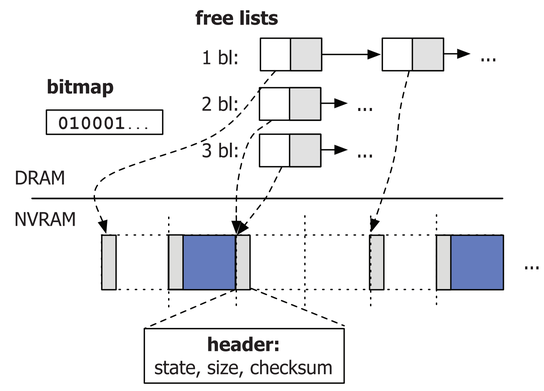
HARDWARE WEAR LEVELING

- Not a complete solution
  - High overhead (for attack prevention)
- Adaptive schemes
  - Penalize applications that write often to one location
- Needs software's help

MEMORY ALLOCATORS FOR DRAM

- Unsuitable for NVRAM
- Caching freed blocks for quick reuse
- Metadata embedded in allocated / free blocks
  - More writes than necessary
  - Frequent writes to one location

NVMALLOC

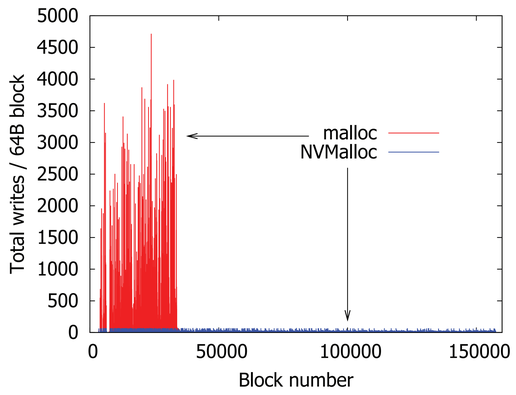


ROBUST MEMORY ALLOCATOR

- Metadata corruption can cause data loss
  - Solution: Add header checksum, over
    - Size
    - State
    - Location
  - Detect and contain corruptions

NVMALLOC RESULTS

WEAR LEVELING



Malloc performs 30-50% more writes than NVMalloc

FRAGMENTATION

#Free/#Allocate ratio	NVMalloc frag. (total/external)	malloc frag. (total/external)
1/3	2.14% (0.26%)	1.63% (0.32%)
1/2	2.29% (0.41%)	2.08% (0.51%)
1/1	10.45% (8.75%)	59.18% (10.68%)

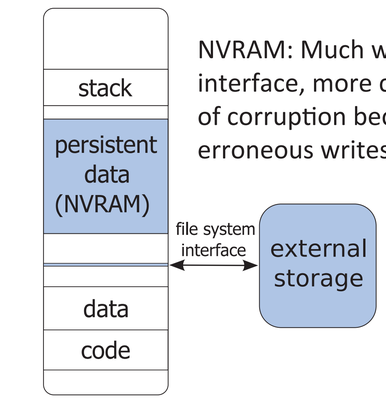
One million allocations/deallocations

OVERHEAD

Memcached, one million operations (75% inserts, 25% deletes)

Allocator	Allocation size	Avg. time [ms] (std. err.)
Memcached slab	10 B - 4 KB	1914 (7)
`malloc`	10 B - 4 KB	1997 (13)
NVMalloc	10 B - 4 KB	1856 (4)
Memcached slab	1 KB	1200 (5)
`malloc`	1 KB	1279 (6)
NVMalloc	1 KB	1258 (5)

ERRONEOUS WRITES



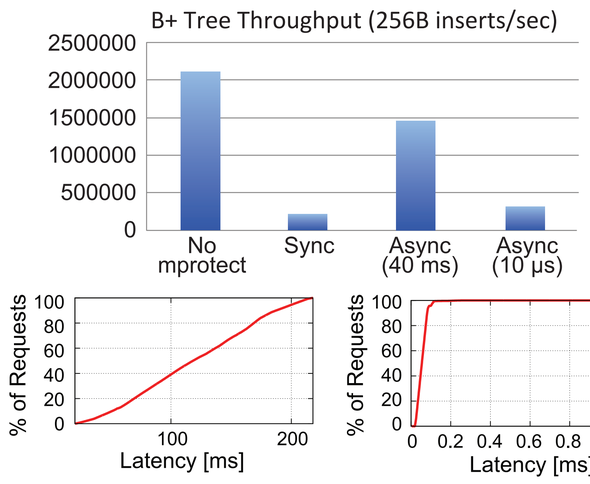
NVRAM: Much wider interface, more chances of corruption because of erroneous writes

Intel Science & Technology Center for Cloud Computing

Carnegie Mellon University

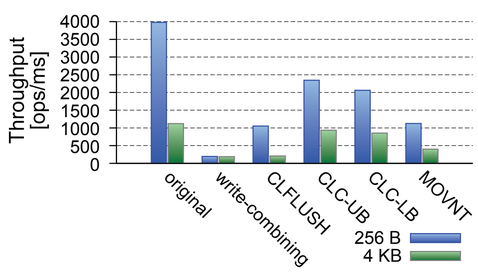
ASYNCHRONOUS MPROTECT

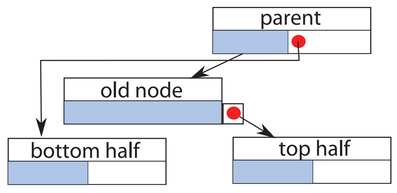
- No waiting, no system call overhead



CACHE LINE COUNTERS

- Consistent updates w/o sacrificing CPU cache use
- Make applications aware of the state of their updates:
  - Given a group of updates, count how many cache lines dirtied by these updates have not yet reached NVRAM
  - Modified B+ Tree:





Georgia Tech · intel · PRINCETON UNIVERSITY · UC Berkeley