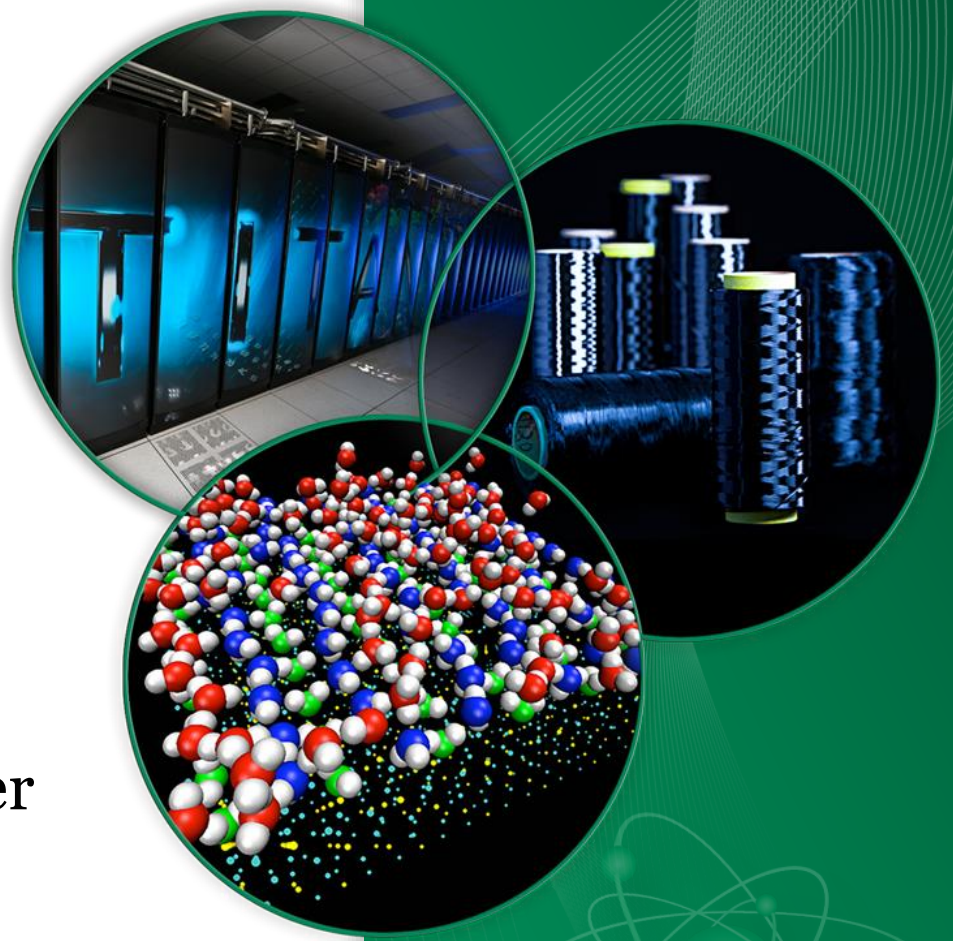
EqualChance: Addressing Intra-set Write Variation to Increase Lifetime of Non-volatile Caches



Sparsh Mittal, Jeffrey S. Vetter

INFLOW
OCTOBER, 2014
COLORADO, USA

Executive Summary

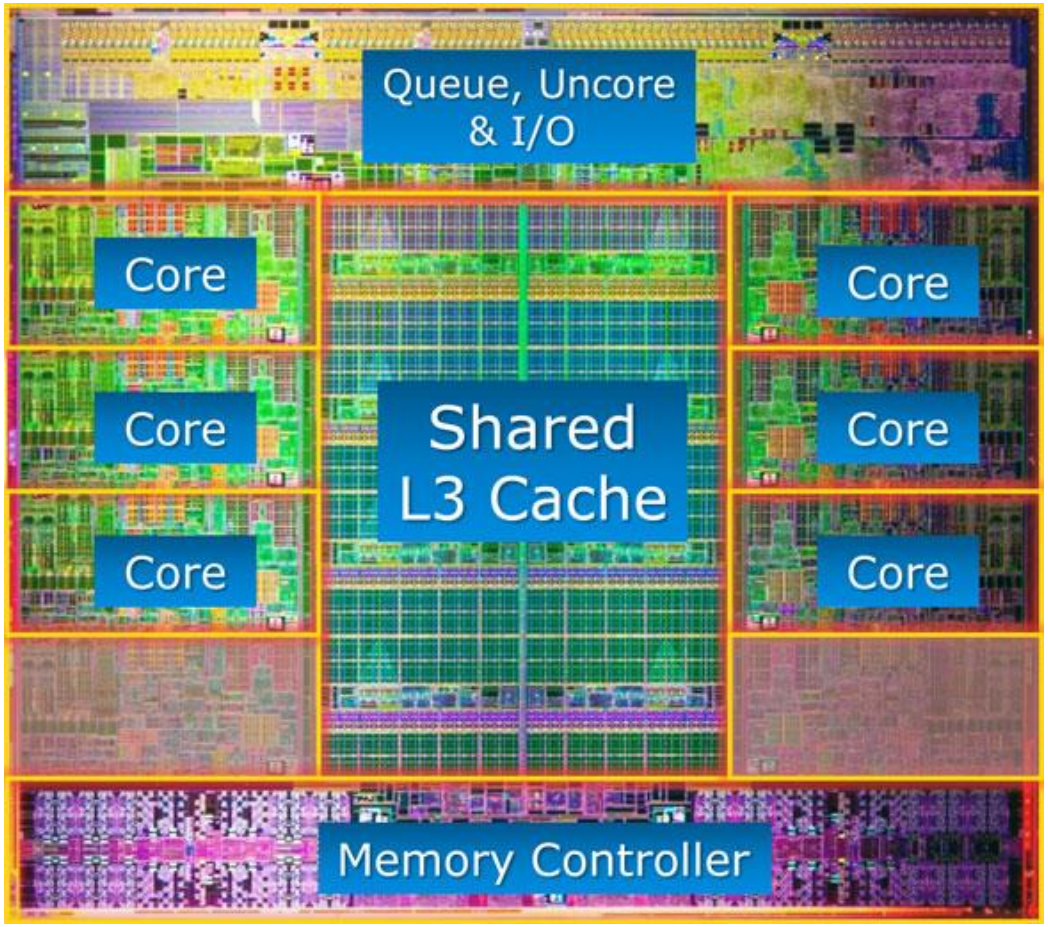
- Limited write endurance is a crucial limitation of NVMs
- Write-variation exacerbates this issue even further.

- We propose EqualChance, a technique to mitigate intra-set write-variation in on-chip last-level caches.
- It periodically changes the physical block location of a data-item to achieve wear-leveling.

- Single core experiments, SPEC2006 and HPC workloads
- Results: EqualChance improves cache lifetime by 4.29X
- It has very small implementation and performance overhead

Motivation: Processor Design Trends

- Core-count is increasing
- LLC size is increasing



Intel's 32nm Sandy Bridge Core i7-3960X 15MB LLC

Motivation: Need of SRAM Alternatives

- SRAM Limitations
  - Scalability challenges
  - Huge leakage power consumption
  - Low density
- SRAM caches consume huge chip area and leakage power
- Power consumption restricts performance scaling

We need SRAM alternatives!

NVMs vis-à-vis SRAM and eDRAM

	SRAM	eDRAM	STT-RAM (NVM)	ReRAM (NVM)	PCM (NVM)
Cell-size (F^2)	120-200	60-100	6-50	4-10	4-12
Write Endurance	**10^{16}**	**10^{16}**	**$4*10^{12}$**	**10^{11}**	**10^{8}-10^{9}**
Speed	Very fast	Fast	Fast read, slow write	Fast read, slow write	Slow read, very slow write
Leakage Power	High	Medium	Low	Low	Low
Retention Period	N/A	30-100 µs	N/A (unless relaxed)	N/A	N/A

NVMs vis-à-vis SRAM and eDRAM

	SRAM	eDRAM	STT-RAM (NVM)	ReRAM (NVM)	PCM (NVM)
Cell-size (F^2)	120-200	60-100	6-50	4-10	4-12
Write Endurance	**10^{16}**	**10^{16}**	**$4*10^{12}$**	**10^{11}**	**10^{8}-10^{9}**
Speed	Very fast	Fast	Fast read, slow write	Fast read, slow write	Slow read, very slow write
Leakage Power	High	Medium	Low	Low	Low
Retention Period	N/A	30-100 µs	N/A (unless relaxed)	N/A	N/A

The write endurance of NVMs is orders of magnitude smaller than that of SRAM/eDRAM!

Write-variation issue in caches

- Conventional cache management policies
  - Optimize performance and energy.
  - Do not account for limited write-endurance
  - May lead to high write-variation

Write-variation issue in caches

- Conventional cache management policies
  - Optimize performance and energy.
  - Do not account for limited write-endurance
  - May lead to high write-variation

- Example: on using LRU replacement policy, repeated writes happen to a hot-block
- This block may fail much early than remaining blocks
- Thus, actual lifetime may be much shorter than expected lifetime with uniform write distribution

An example from SPEC06 suite

- Lbm is most write-intensive among SPEC06 apps
- Povray has the highest intra-set and inter-set write-variation

- Write-magnitude of Lbm = 41X that of Povray
- Worst-case writes with Lbm = 1/20X that of Povary

- Clearly, *variation in writes* is more crucial issue than *magnitude of writes*.

EqualChance: Addressing Intra-set Write Variation to Increase Lifetime of Non-volatile Caches

EqualChance: Key Idea

- Conventional cache management policies aim to keep hot-data in cache as much as possible
- Issue: This increases writes to blocks storing those data

EqualChance: Key Idea

- Conventional cache management policies aim to keep hot-data in cache as much as possible
- Issue: This increases writes to blocks storing those data

- Idea: Periodically change block-location of those data
- Use counters to record writes on each set. After certain number of writes, swap hot data with another cold data
- The candidate for swap may be invalid (called I-shifting) or clean (called C-shifting)
- We do not swap with dirty data since this may itself be frequently written

EqualChance Wear-leveling Algorithm

Wear-leveling Algorithm 1 of 3

```
z= way-index of write-hit block
if( FlagBit[setId] is ON)
{
 p = way-index of least recent invalid block in setId
    if(p is found)
    {
      Swap data of ways z and p . Do not update LRU-information
    }
    else
    {
      q = way-index of least recent clean block in setId
       if(q is found)
          Swap data of ways z and q . Do not update LRU-information
       else
          ItIsNormalWrite = TRUE
    }
}
else
  ItIsNormalWrite = TRUE
```

Wear-leveling Algorithm 2 of 3

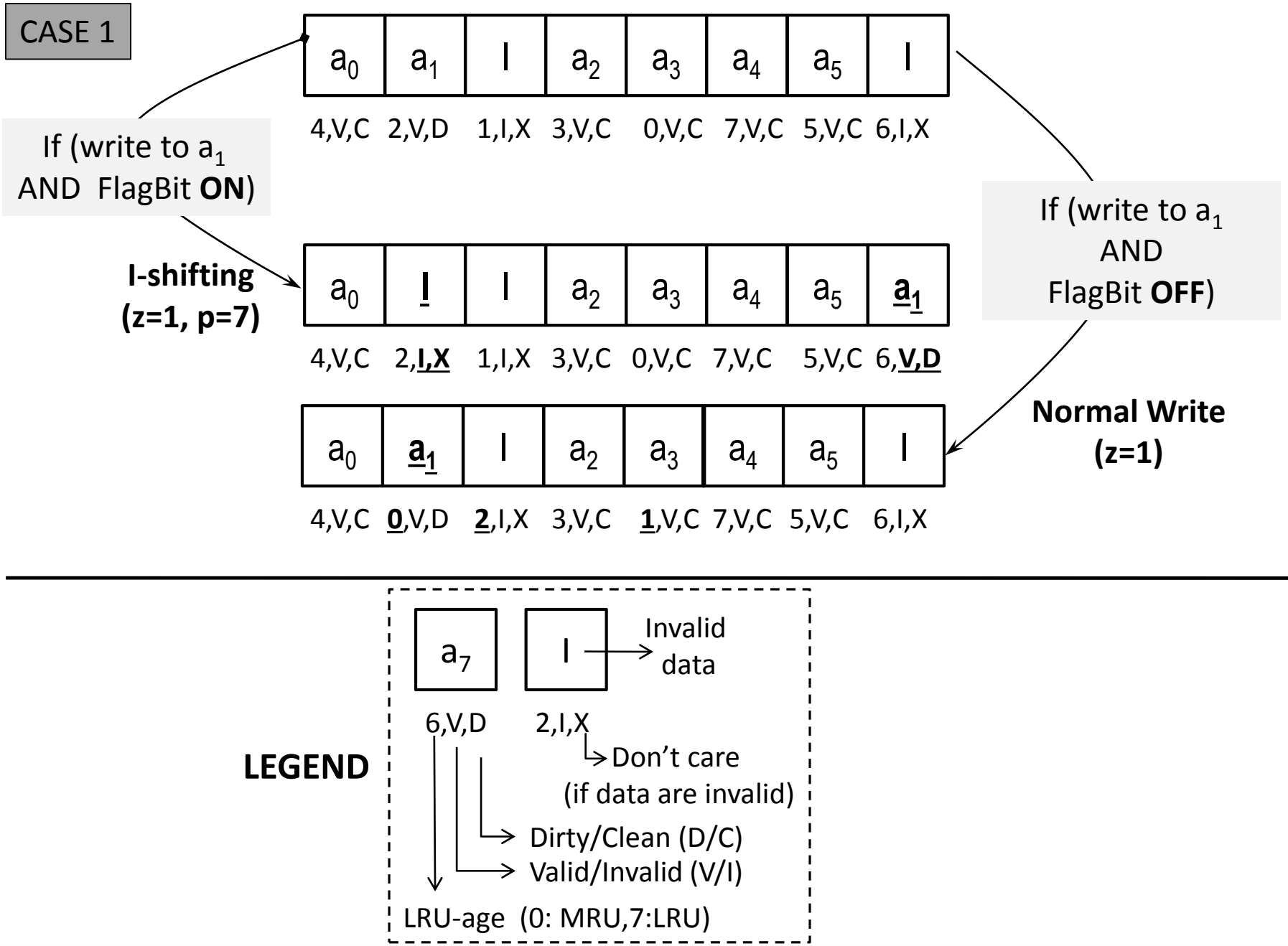
```
z= way-index of write-hit block
if(FlagBit[setId] is ON)
{
 p = way-index of least recent invalid block in setId
   if(p is found)
    {
      Swap data of ways z and p. Do not update LRU-information
    }
    else
    {
      q = way-index of least recent clean block in setId
       if(q is found)
          Swap data of ways z and q. Do not update LRU-information
       else
          ItIsNormalWrite = TRUE
    }
}
else
  ItIsNormalWrite = TRUE
```

1-shifting

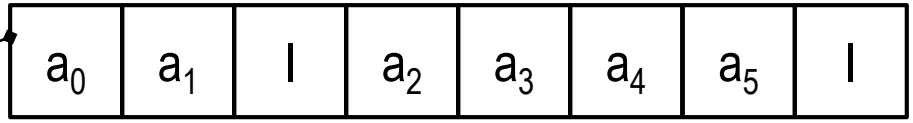
Wear-leveling Algorithm 3 of 3

```
z= way-index of write-hit block
if(FlagBit[setId] is ON)
{
 p = way-index of least recent invalid block in setId
   if(p is found)
    {
     Swap data of ways z and p. Do not update LRU-information
    }
   else
   {
    q = way-index of least recent clean block in setId
     if(q is found)
        Swap data of ways z and q. Do not update LRU-information
     else
        ItIsNormalWrite = TRUE
   }
}
else
  ItIsNormalWrite = TRUE
```

C-shifting



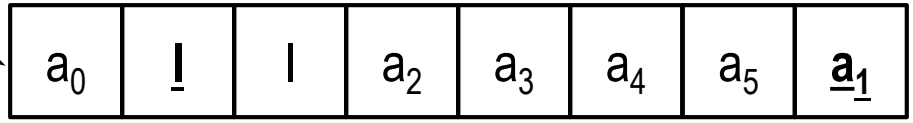
CASE 1



4,V,C 2,V,D 1,I,X 3,V,C 0,V,C 7,V,C 5,V,C 6,I,X

If (write to a_1 AND FlagBit **ON**)

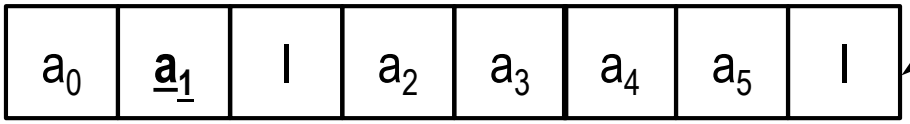
I-shifting (z=1, p=7)



4,V,C 2,**I,X** 1,I,X 3,V,C 0,V,C 7,V,C 5,V,C 6,**V,D**

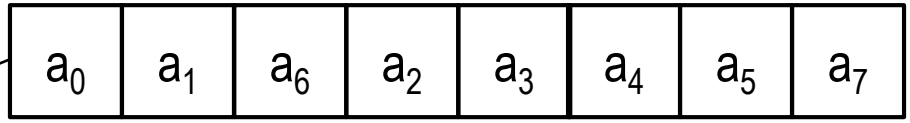
If (write to a_1 AND FlagBit **OFF**)

Normal Write (z=1)



4,V,C **0**,V,D **2**,I,X 3,V,C **1**,V,C 7,V,C 5,V,C 6,I,X

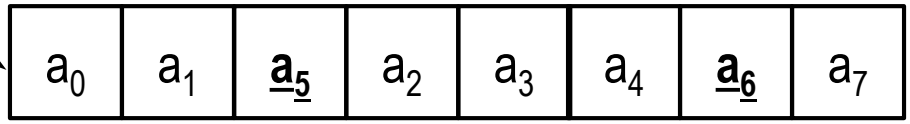
CASE 2



4,V,C 0,V,D 1,V,C 3,V,C 2,V,C 7,V,D 5,V,C 6,V,D

If (write to a_6 AND FlagBit **ON**)

C-shifting (z=2, q=6)



4,V,C 0,V,D 1,V,**C** 3,V,C 2,V,C 7,V,D 5,V,**D** 6,V,D

Overhead Estimation

- Overhead comes due to extra counters and swap-buffer (used for data-transfer)
- Let N= number of sets, M = associativity, L= block size, G= # of tag-bits

$$Overhead = \frac{(N \times 5) + (64 \times L)}{N \times M \times (L + G)} \times 100$$

- Overhead is < 0.15% of L2 cache size

- A small increase in LLC latency can be easily hidden

Salient Features

- Can be easily integrated with write-minimization techniques
- No offline profiling is required.
- Does not increase DRAM traffic, unlike data-invalidation techniques [1] ==> does not harm performance or energy
- Wear-leveling has the side benefit of reducing thermal density

[1]. J. Wang et al. "i²WAP: Improving non-volatile cache lifetime by reducing inter-and intra-set write variations," HPCA 2013.

Experiments

- Sniper x86-64 simulator, 300M instruction
- Single core simulations using 4MB L2.
- ReRAM (resistive RAM) L2, parameters from NVsim.
- Baseline: Shared LRU cache with no wear-leveling
- We measure energy of L2 cache, main memory and algorithm.

Evaluation Metrics

- We show results on:
  - Relative lifetime, relative performance and percentage energy loss
  - Coefficient of inter-set write-variation (InterV) and intra-set write-variation (IntraV) [1]

$$InterV = \frac{100}{W_{avg}} \sqrt{\frac{\sum_{i=1}^{N} \left(\sum_{j=1}^{M} w_{i,j}/M - W_{avg} \right)^2}{N-1}}$$

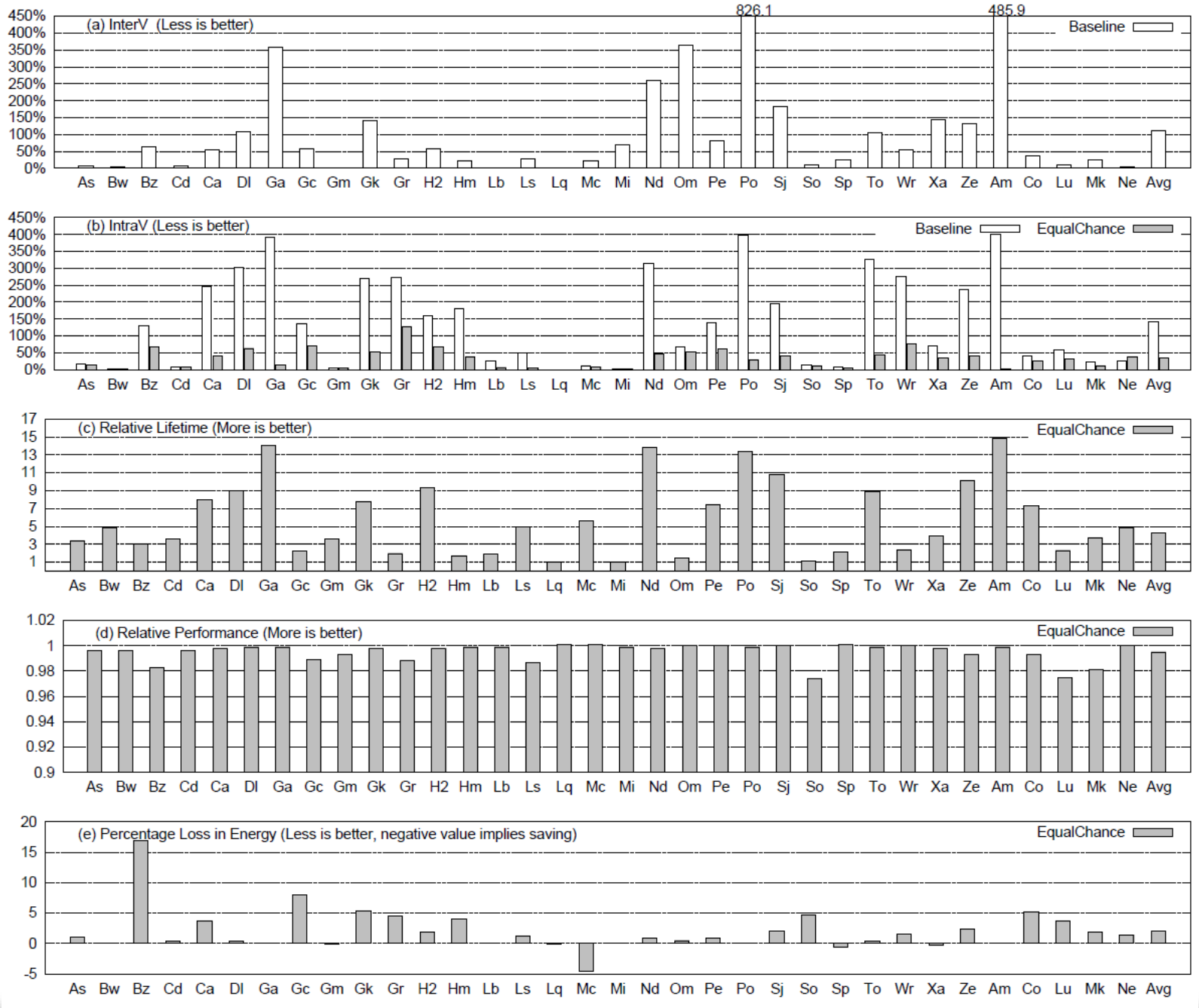
$$IntraV = \frac{100}{N \cdot W_{avg}} \sum_{i=1}^{N} \sqrt{\frac{\sum_{j=1}^{M} \left(w_{i,j} - \sum_{r=1}^{M} w_{i,r}/M \right)^2}{M-1}}$$

Workloads

34 Workloads
(all 29 SPEC2006 Benchmarks & 5 DoE applications)

As(astar), Bw(bwaves), Bz(bzip2), Cd(cactusADM), Ca(calculix)
Dl(dealII), Ga(gamess), Gc(gcc), Gm(gemsFDTD), Gk(gobmk)
Gr(gromacs), H2(h264ref), Hm(hmmer), Lb(lbm), Ls(leslie3d)
Lq(libquantum), Mc(mcf), Mi(milc), Nd(namd), Om(omnetpp)
Pe(perlbench), Po(povray), Sj(sjeng), So(soplex), Sp(sphinx)
To(tonto), Wr(wrf), Xa(xalancbmk), Ze(zeusmp), Am(amg2013)
Co(CoMD), Lu(LULESH), Mk(MCCK), Ne(Nekbone)

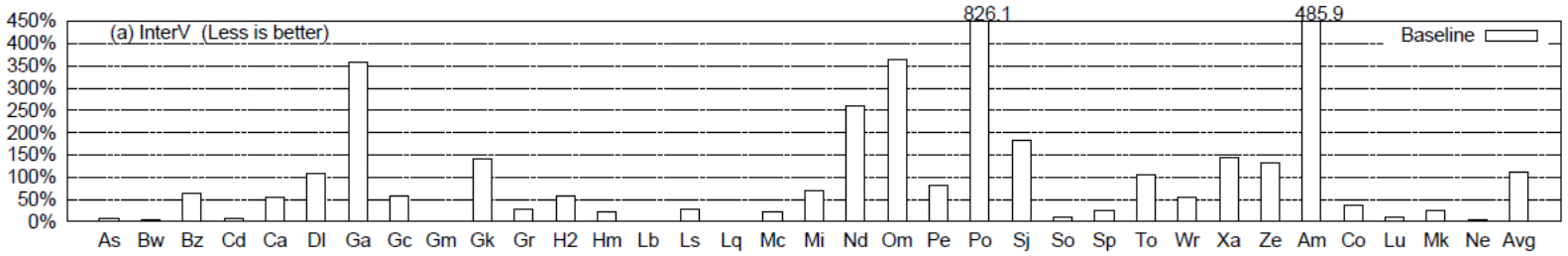
Results

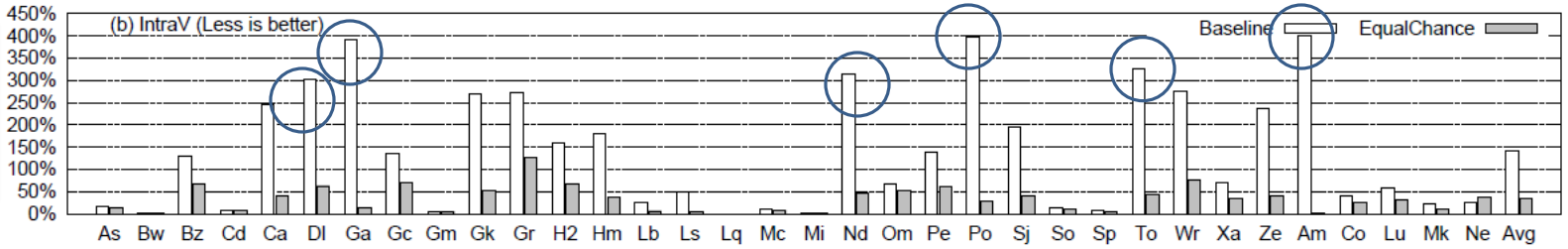


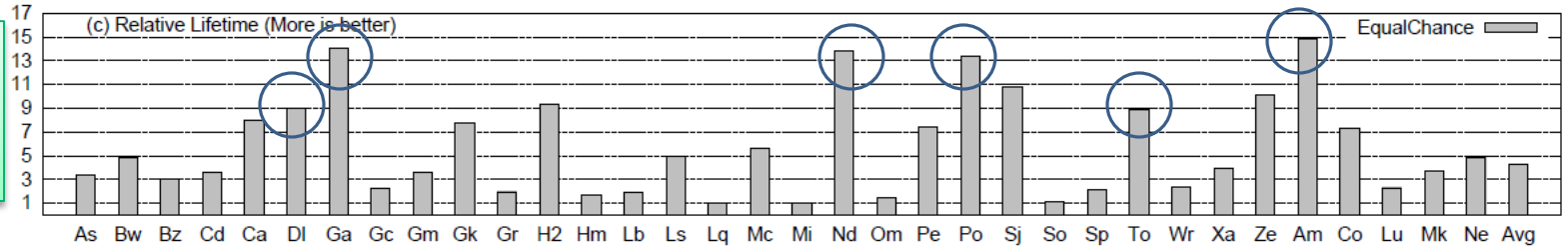
Large IntraV in several workloads

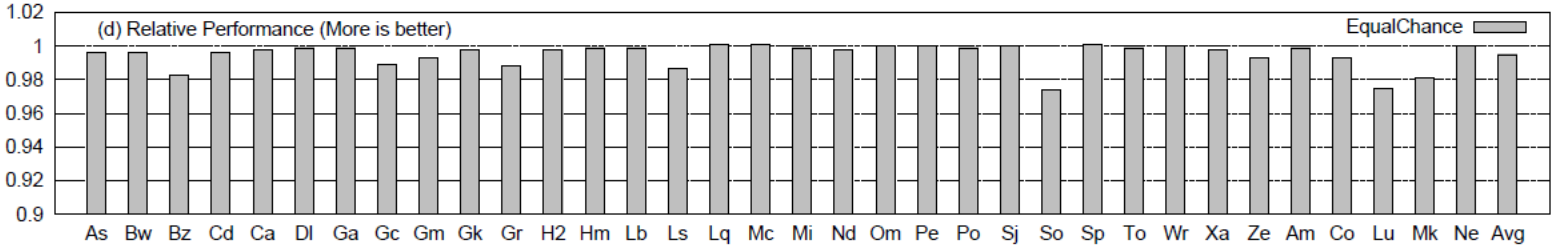
Corresponding lifetime improvement

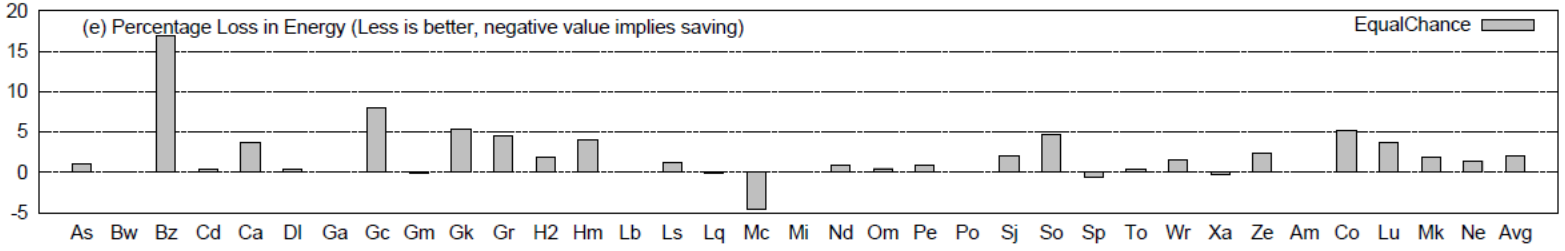
Performance and energy losses are negligible









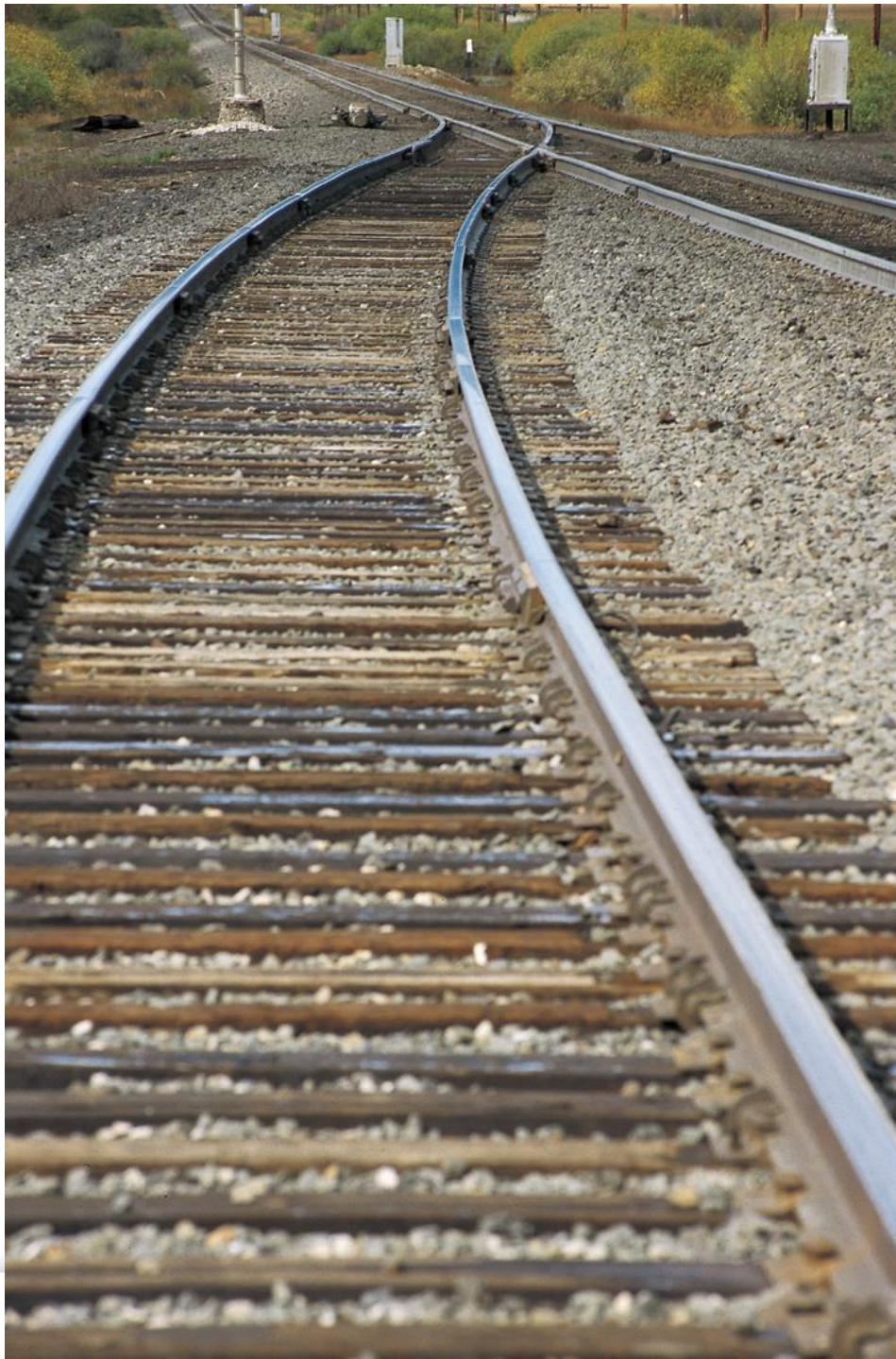


Result Analysis

- EqualChance improves lifetime by 4.29X and reduces IntraV significantly
- Lifetime improvement depends on the intra-set write variation present in baseline program.
- Some programs show > 10X lifetime improvement.
- Performance close to baseline and energy loss < 2%.
- **Parameter sensitivity results** show that it works well for wide range of algorithm and system parameters.

Conclusion and Future Work

- We presented EqualChance for improving lifetime of NVM caches.
- Future Work
  - Integration with other techniques, e.g. write-minimization.
  - Extending to processors with tens of cores
  - Evaluation with multithreaded workloads.




Thanks a lot!
http://ft.ornl.gov
mittals@ornl.gov
OAK RIDGE
National Laboratory

Extra Slides

Table 3: Number of I-and C-shifting operations

	I-shifting	C-shifting		I-shifting	C-shifting
As	120K	143K	Mi	298K	50K
Bw	1K	61K	Nd	76K	0
Bz	271K	0	Om	4K	2K
Cd	344	86K	Pe	11K	571
Ca	261K	0	Po	107K	0
Dl	63K	0	Sj	45K	0
Ga	133K	0	So	737K	577K
Gc	97K	35K	Sp	361	58K
Gm	217K	517K	To	120K	0
Gk	131K	0	Wr	33K	0
Gr	156K	6K	Xa	34K	66K
H2	47K	0	Ze	625K	0
Hm	176K	0	Am	82K	0
Lb	2580K	1K	Co	71K	33
Ls	51K	595K	Lu	134K	239K
Lq	0	201K	Mk	21K	251K
Mc	38K	361K	Ne	8K	0

Table 4: Parameter Sensitivity Results for EqualChance. Default values: $\Upsilon = 5$, overhead values shown in Section 4, 16-way, 4MB L2 cache (Perf. = performance).

	% InterV	% IntraV		Relative	Relative	% Loss in	I-shifting	C-shifting
	Baseline	Baseline	EqualChance	Lifetime	Perf.	Energy	operations	operations
Default	111.1	141.8	33.8	4.29	1.00	1.97	199K	96K
Υ=10	111.1	141.8	41.5	3.79	1.00	1.11	78K	69K
Υ=15	111.1	141.8	44.9	3.60	1.00	0.75	48K	48K
Higher overhead	111.1	141.8	33.8	4.29	0.99	2.01	199K	96K
8-way L2	154.2	111.7	28.8	3.09	0.99	1.55	148K	144K
32-way L2	74.0	171.6	36.3	5.02	1.00	3.29	251K	44K
2MB L2	71.9	108.6	25.9	3.38	0.99	2.96	157K	139K
8MB L2	152.3	181.5	45.1	6.20	0.99	2.74	239K	53K