쓰기 참조의 특성과 SCM 기반 메모리 관리

Write reference characteristics and SCM-based memory management

Hyokyung Bahn

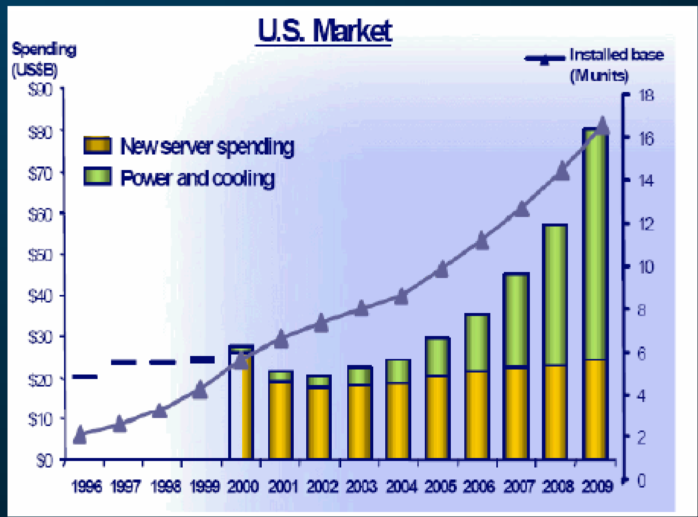
2011.4.19
NVRAMOS 2011

EWHA WOMANS UNIVERSITY

Storage Class Memory (SCM)

- **SCM Characteristics**
  - **Nonvolatile, Byte-addressable**
    - **eg. PCM (Phase Change Memory), FeRAM, STT-RAM (MRAM)**

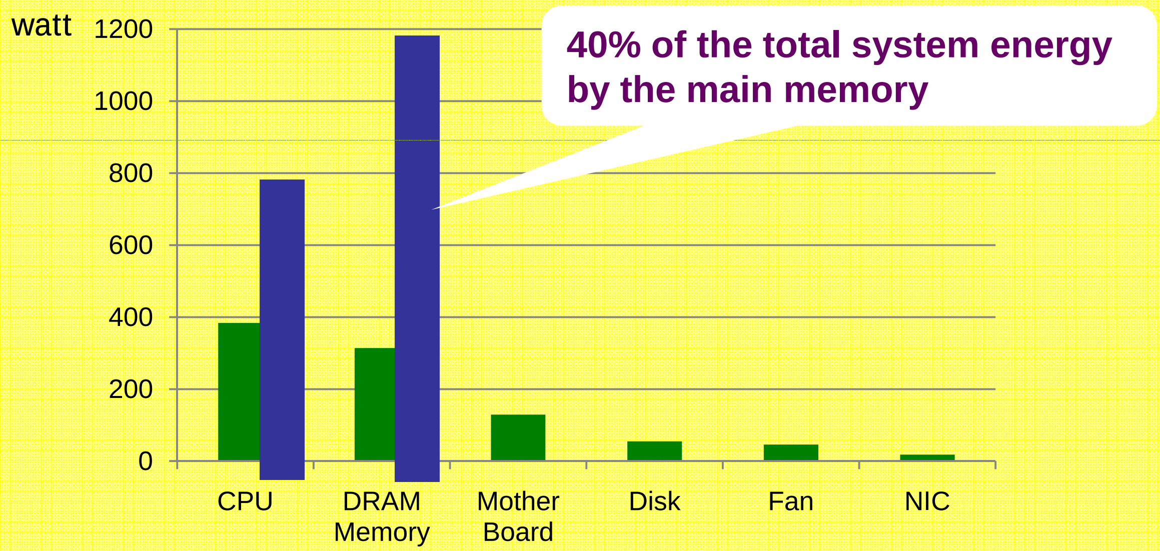
- **SCM Perspectives**
  - **Widely deployed in data center by 2012**
  - **Promisingly replace HDD by 2020**
    - **No more than 3-5x cost of HDD (<$1/GB in 2012)**
    - **< 1usec Access time**
    - **> 10^5 Read ops. Per second**
    - **> 100MB / sec**
    - **10x lower power than HDD**



(IBM Almaden Research Center, USENIX FAST Tutorial, 2009)

Why DRAM main memory need to change?

Multi-core system, More concurrency, Larger working set
→ an enormous need for increased memory
eg) 4GB/32-bit processors, 16EB/64-bit processors (1E = 10^{18})

Density (cost/bit)	❖ DRAM scaling to small technology is challenge
Power Consumption	

(Source: Intel Labs, 2008)

Phase Change Memory (PCM)

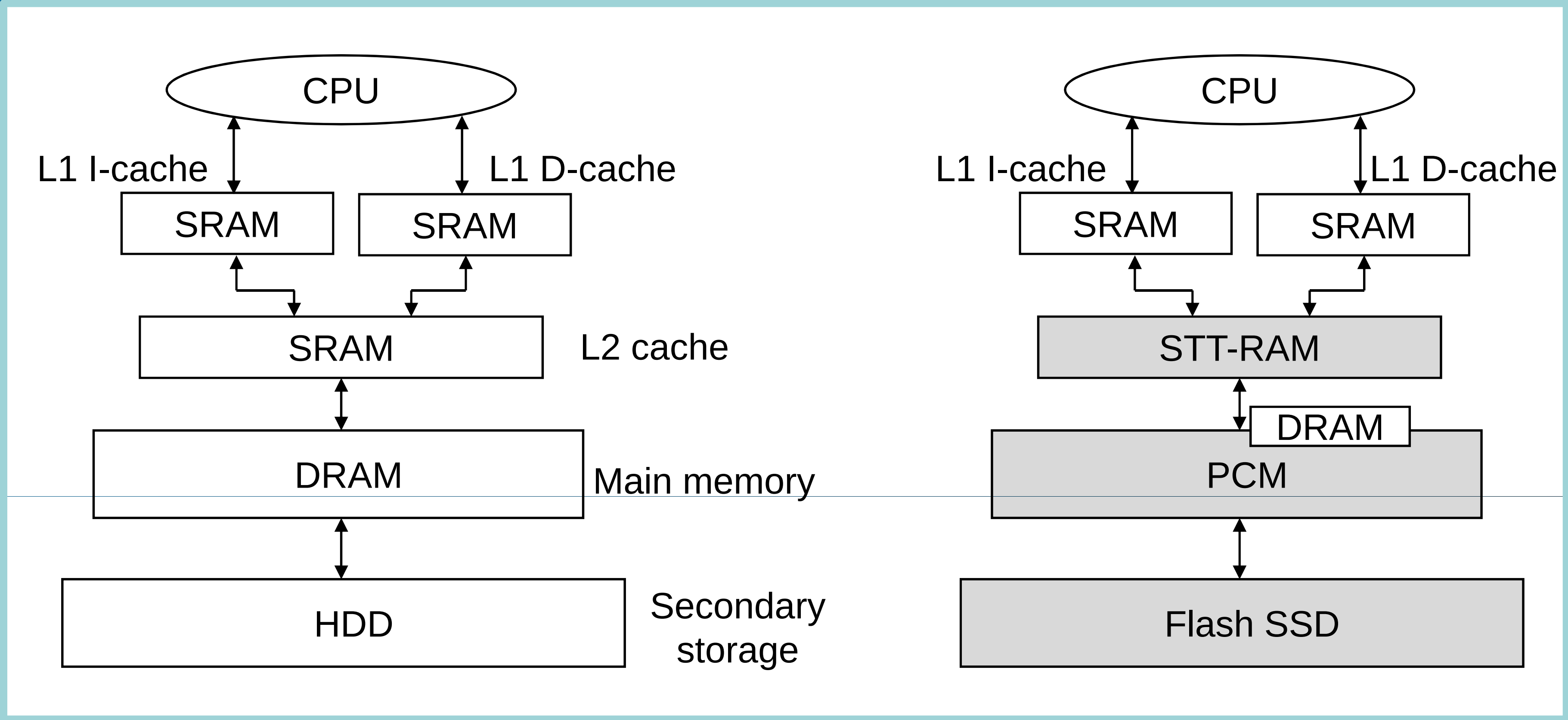
		DRAM (DRAM-DDR3 1.35V)	PCM (High Speed PCM '10)
Non-Volatile		NO	YES
Density		1X	2X ~ 4X
Power (Energy)	Read(J/GB)	0.7	1
	Write(J/GB)	1.1	6
	Static power (mW/GB)	100	1

PCM Challenges

		DRAM (DRAM-DDR3 1.35V)	PCM (High Speed PCM '10)
Non-Volatile		NO	YES
Density		1X	2X ~ 4X
Power	Read(J/GB)	0.7	1
	Write(J/GB)	1.1	6
	Idle state (mW/GB)	100	1
Latency	Read	1X	1X~ 2X
	Write	1X	7X ~ 8X
Endurance **		10^{15}	10^{7} ~10^{8}

** SRAM 10^{15}, STT-RAM 10^{15}, FeRAM 10^{12}, SLC Flash 10^{5}, MLC Flash 10^{4}

Memory & Storage Architectures



- STT-RAM, PCM, Flash SSD: write is slower than read

Estimating Future Writes

1. Find a good estimator for future write references

   Issue i. Considering read and write history together or considering write history alone
   Issue ii. Which is better? Temporal locality or Frequency based estimation

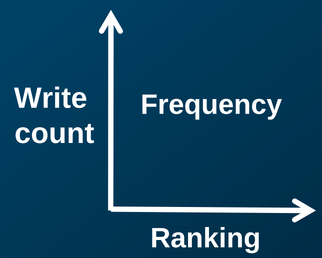
2. Store pages likely to be re-written on DRAM.

1. Temporal Locality
- Only write history
- Total (read+write) history

2. Frequency
- Only write history
- Total (read+write) history

3. Comparing
Temporal Locality & Frequency
Based Estimation

- by (read + write) recency
- by write recency



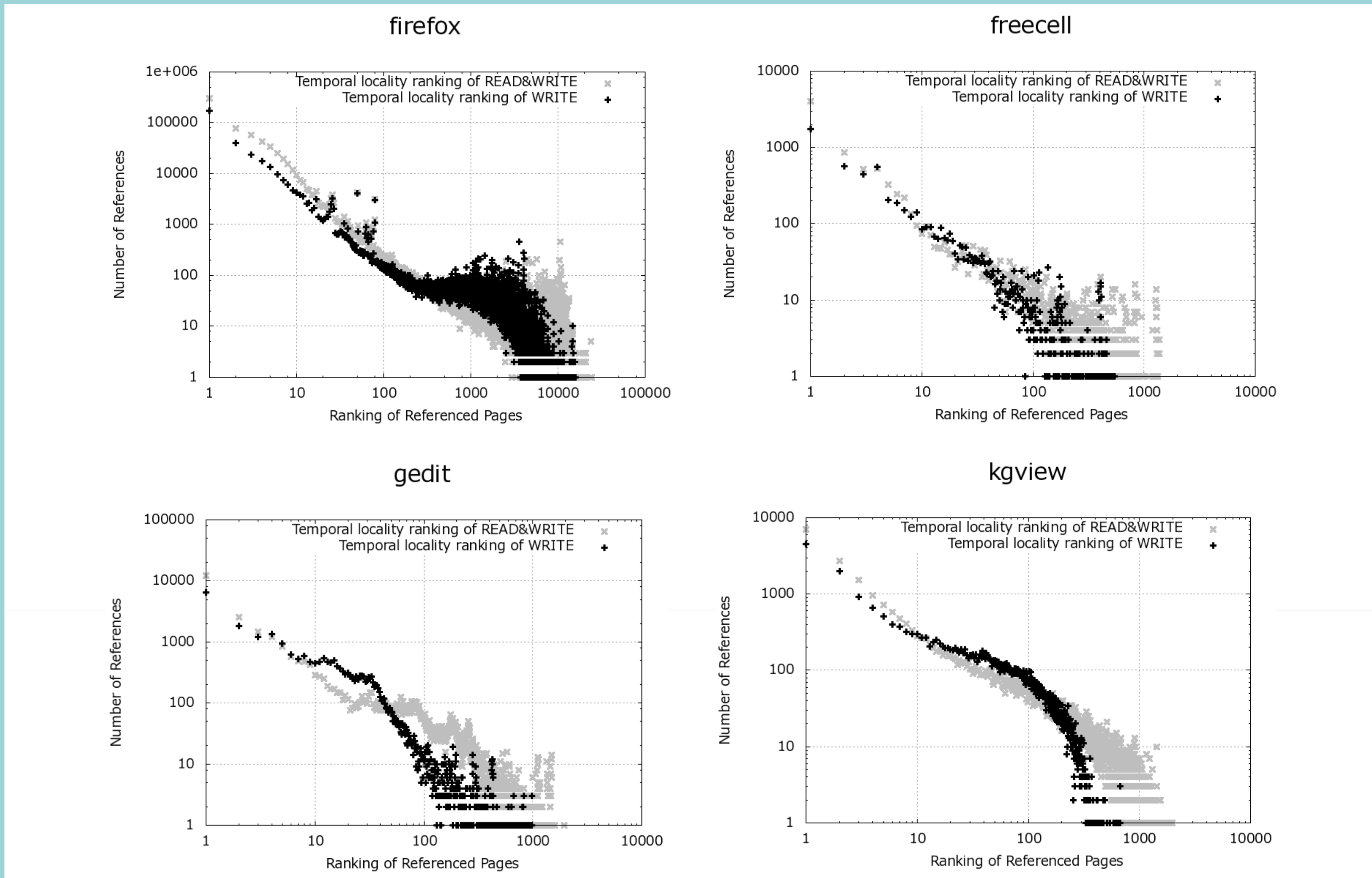
- by (read + write) frequency
- by write frequency

- by recency
- by frequency

Virtual Memory Traces Used

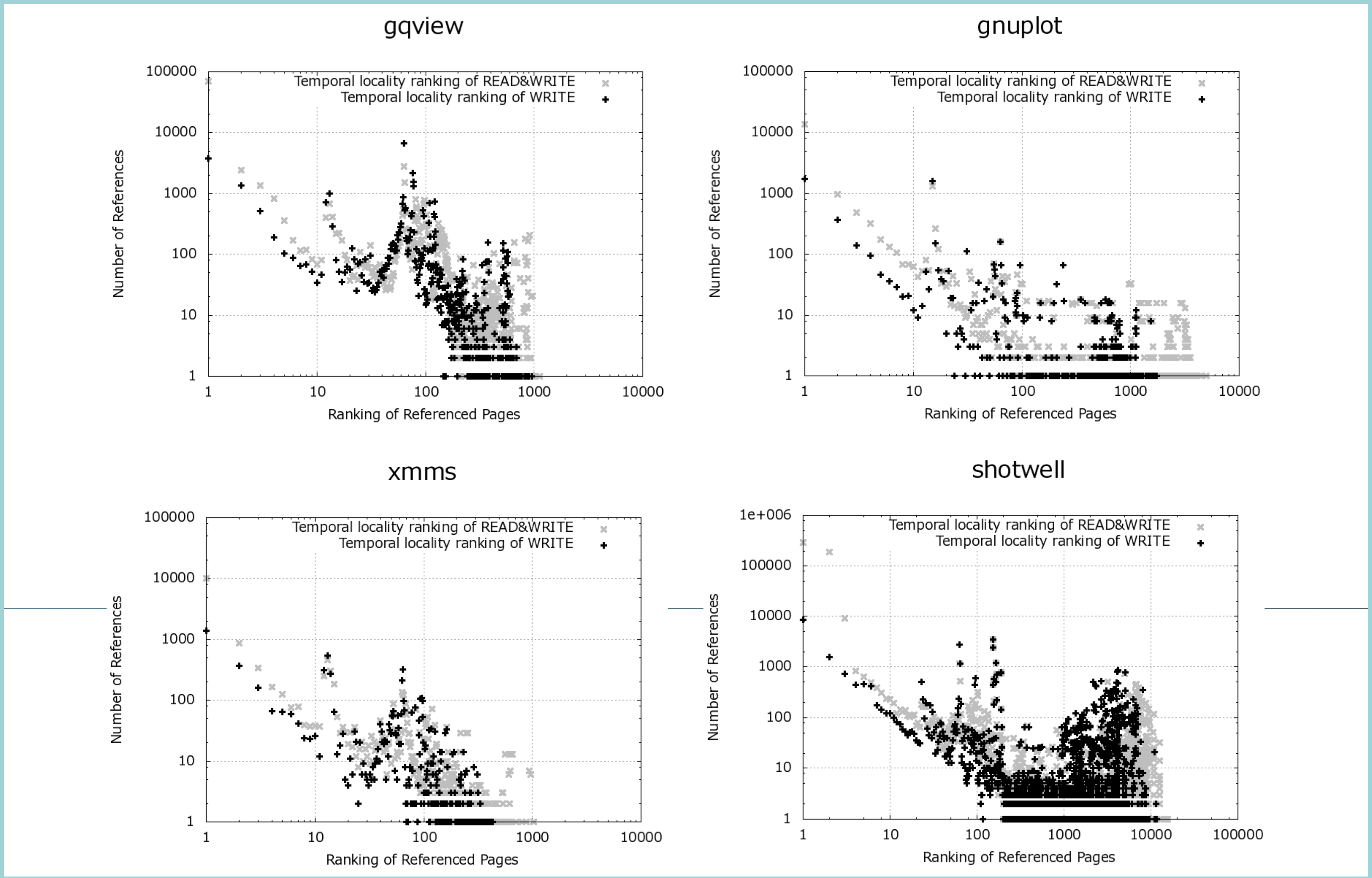
Workload	Contents	Memory footprint(KB)	Ratio of operations (data reads : data writes)	Memory access count			
				total	Instruction read	Data read	Data write
xmms	Mp3 player	8,052	1 : 7.79	1,169,310	65,413	125,653	978,244
gqview	Image viewer	7,428	1 : 2.01	611,142	93,653	172,044	345,445
shotwell	Photo management S/W	88,228	1 : 1.04	15,090,070	528,549	7,124,101	7,437,420
gnuplot	Graphing utility	21,132	1 : 1.10	220,240	47,551	82,110	90,579
firefox	Web browser	101,520	1.88 : 1	12,648,471	2,392,952	6,690,045	3,565,474
freecell	Game	10,084	5.26 : 1	490,700	114,750	315,906	60,044
gedit	Word processor	14,460	7.16 : 1	1,736,440	652,154	951,450	132,836
kghostview	PDF file viewer	17,388	10.26 : 1	1,548,820	373,260	1,062,008	103,552

Temporal Locality



- Using both read & write history estimates future writes better within top 10 rankings.
- Beyond top rankings, using write history alone may be better estimates of future writes.
- Overall, both estimators show similar results.

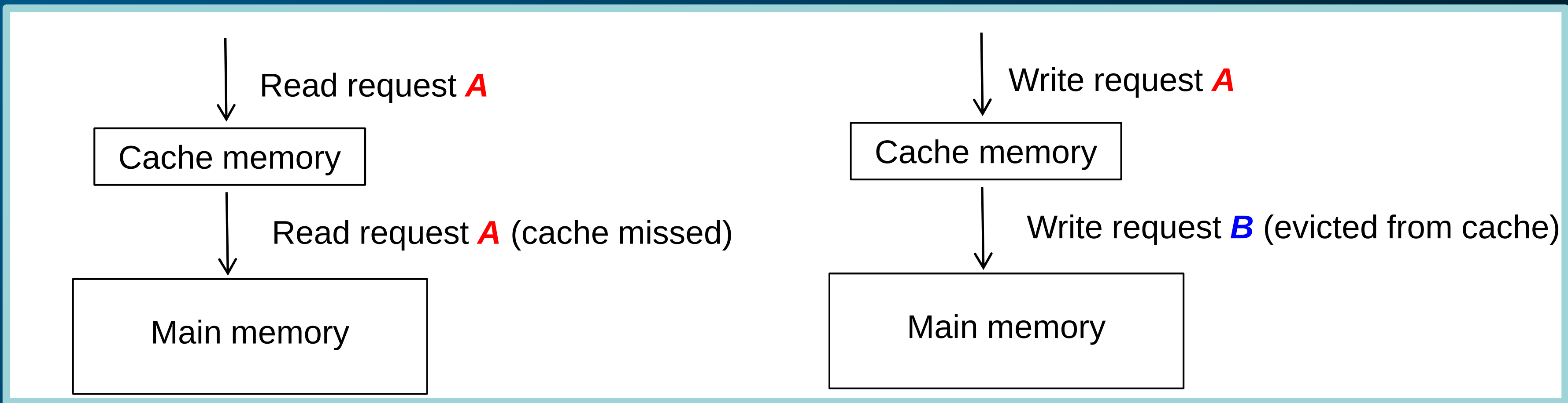
Temporal Locality



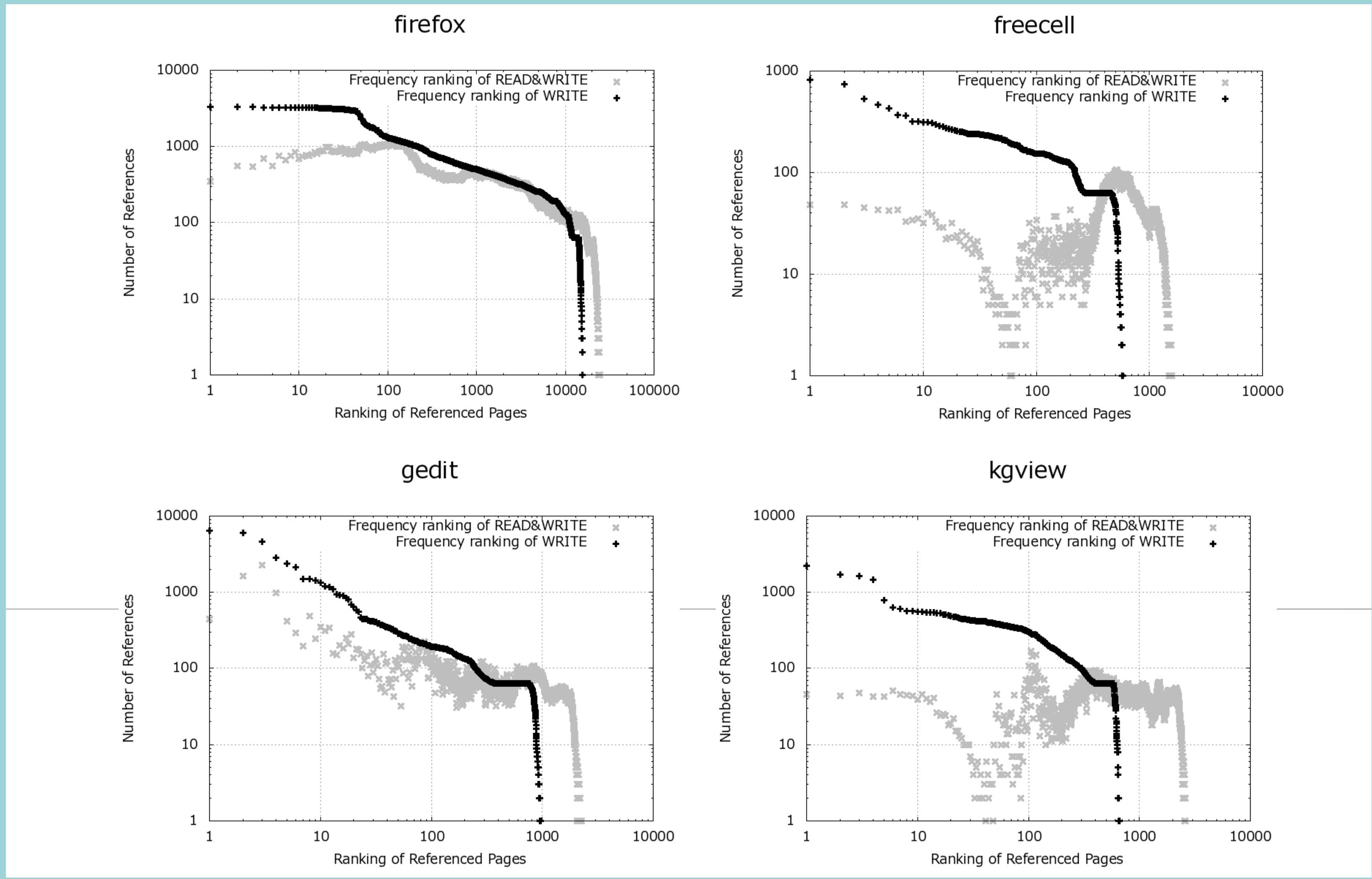
- Temporal locality for relatively write intense workloads are rather irregular (Ranking inversion)
- Temporal locality alone may not be sufficient to estimate the likelihood of future writes.

Why temporal locality of write irregular?

- Maybe due to write-back operation of cache memory
  - page references observed at VM contain only cache-missed ones
  - In case of read,
    - cache-missed requests are directly propagated to VM
      $\rightarrow$ Even though temporal locality becomes weak, it is not damaged seriously
  - In case of write,
    - cache-missed requests are not propagated directly to VM
    - but just written to the cache memory.
    - requests are delivered to VM only after evicted from cache memory.
    - time a write request arrives $\neq$ time the request is delivered to VM

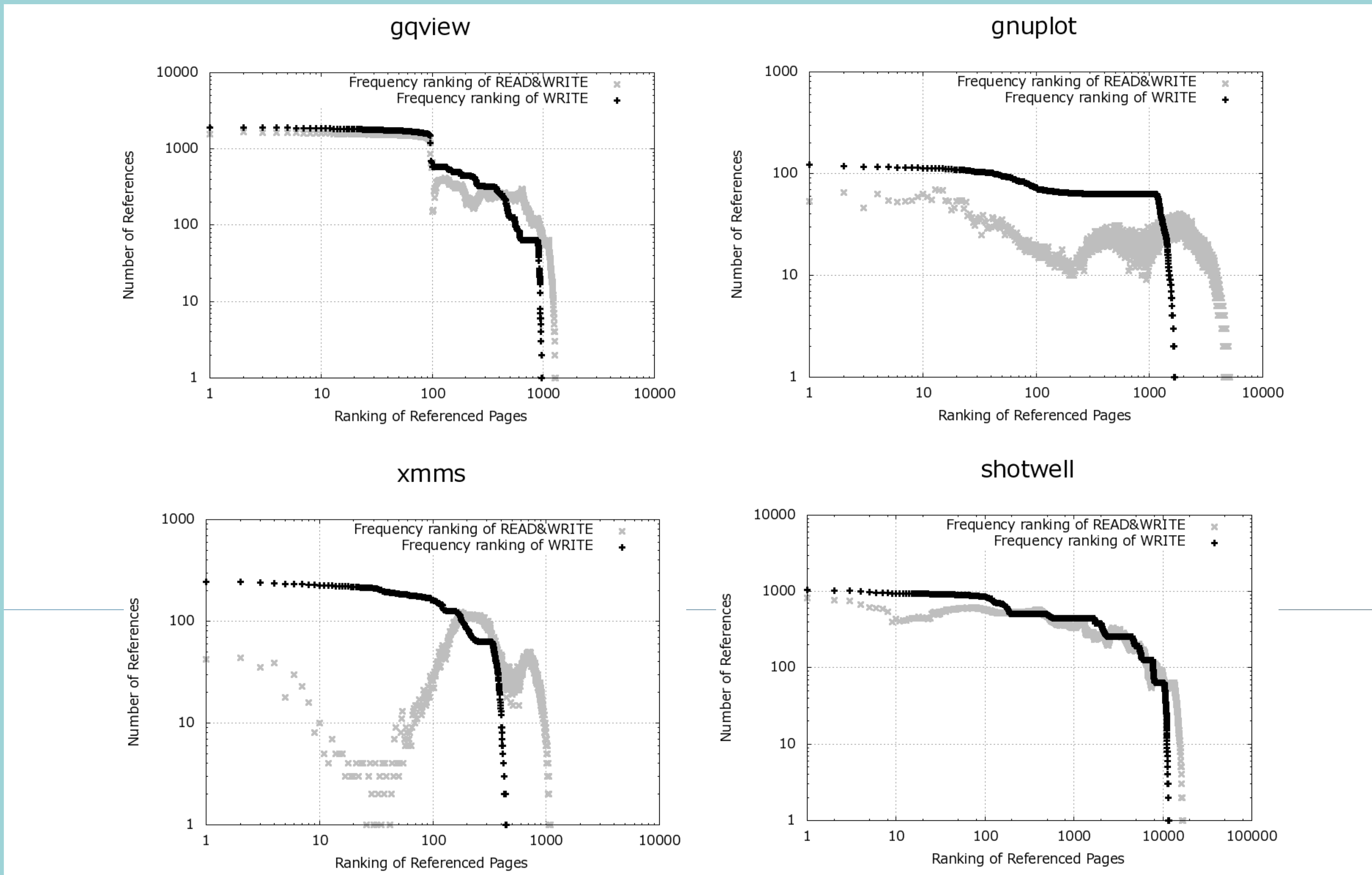


Frequency



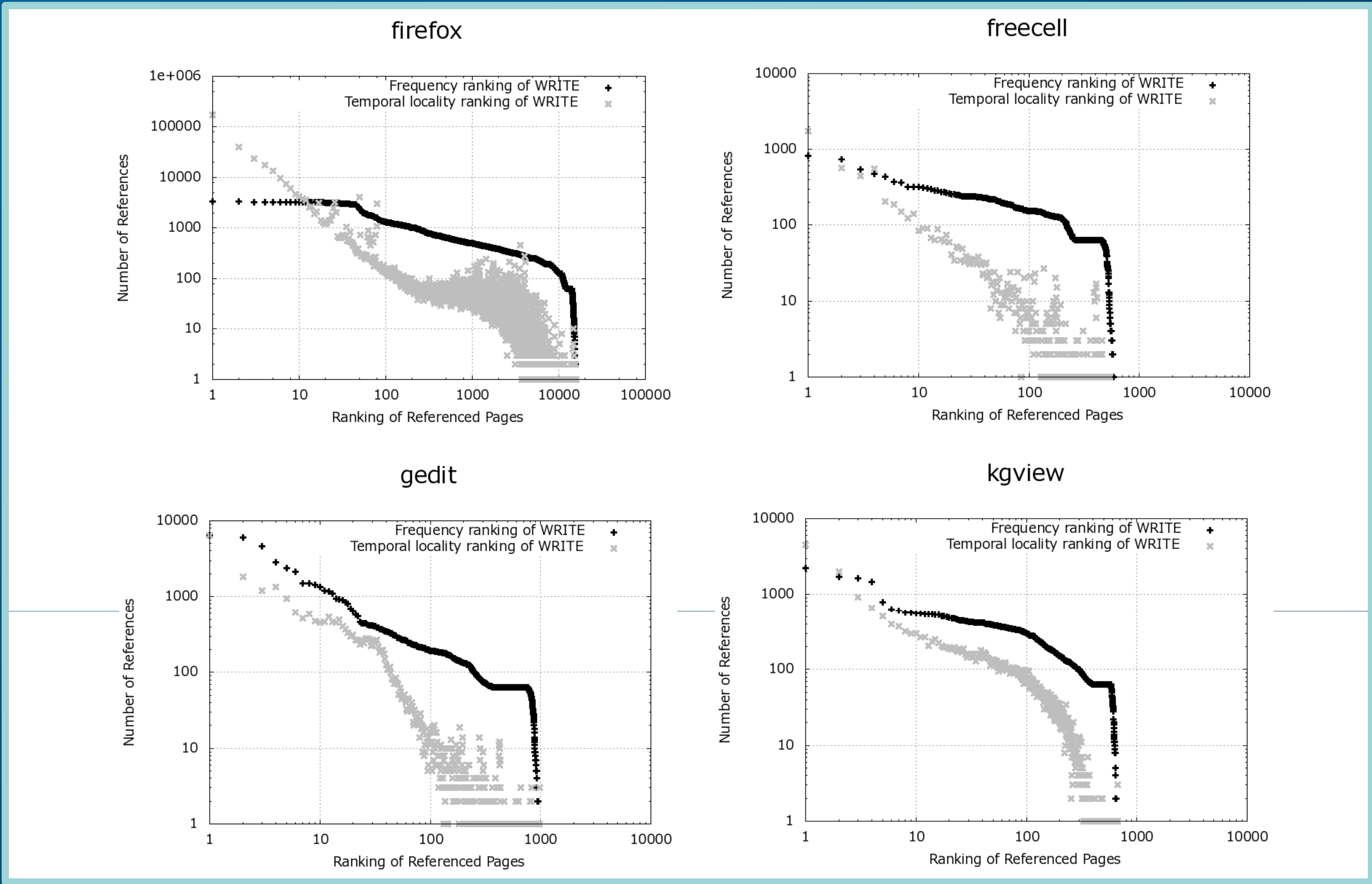
- Write frequency alone is more effective than frequency counted by both reads and writes

Frequency



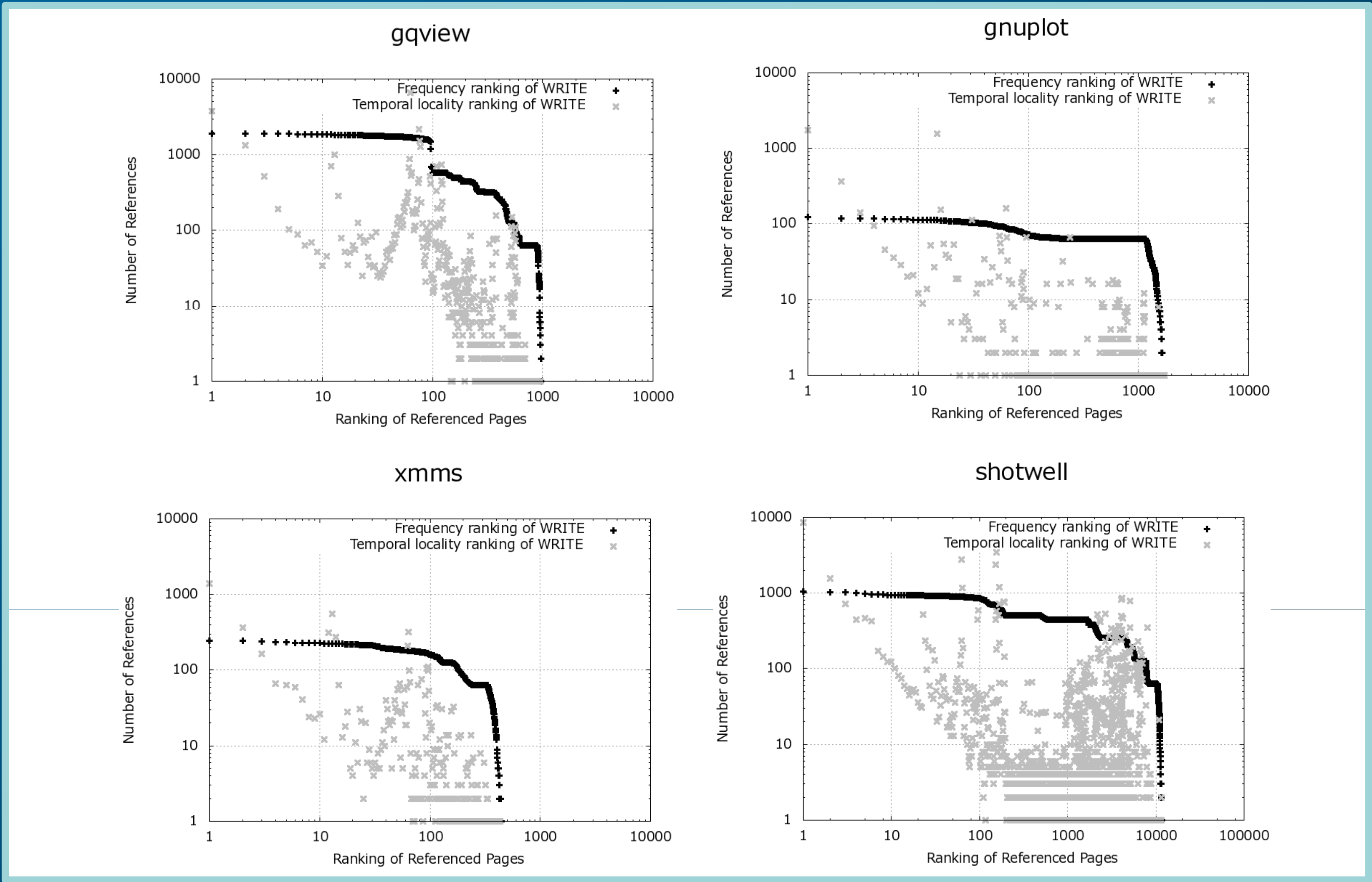
- Write frequency alone is more effective than frequency counted by both reads and writes

Temporal Locality vs. Frequency



- Frequency is more effective than temporal locality for most cases.
- However, at least the most recent reference history must be considered.

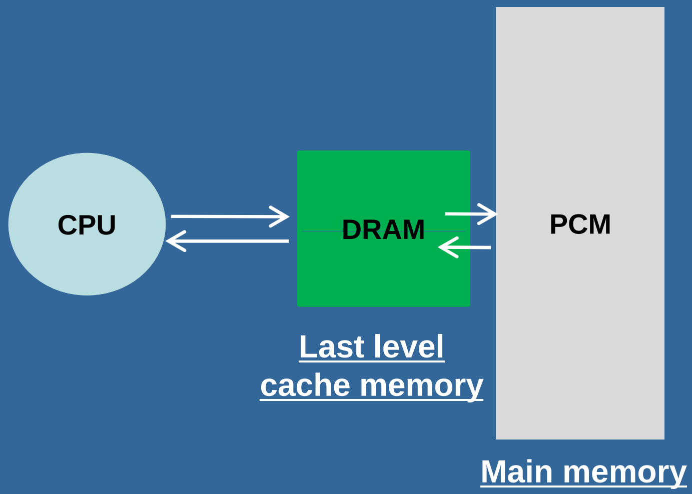
Temporal Locality vs. Frequency



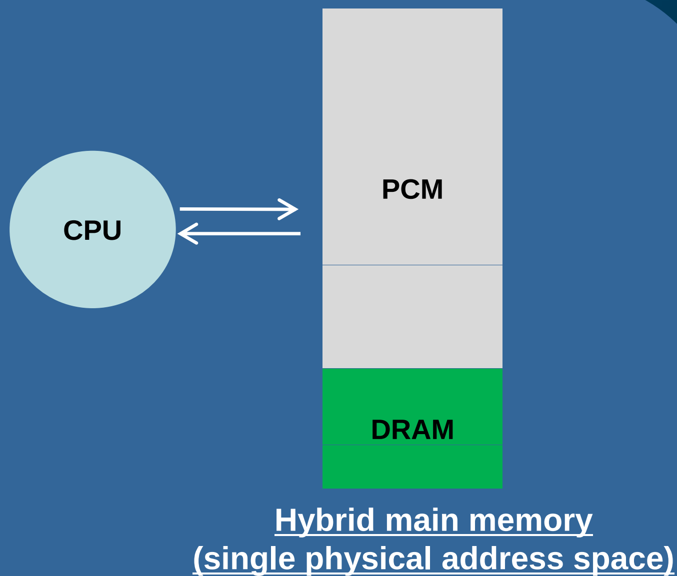
- Frequency is more effective than temporal locality for most cases.
- However, at least the most recent reference history must be considered.

Memory Architecture

✓ Write latency & Endurance problem of PCM
→ Use a small amount of DRAM along with PCM.



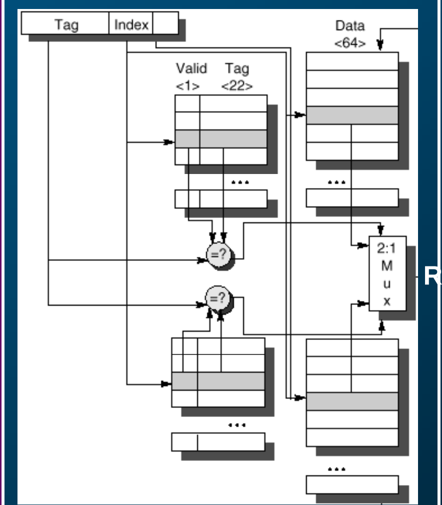
- DRAM cache miss → PCM access
- DRAM cache is hidden to the OS
  → H/W implementation,
  Fully associative placement is difficult!
  Collision may degrade space efficiency

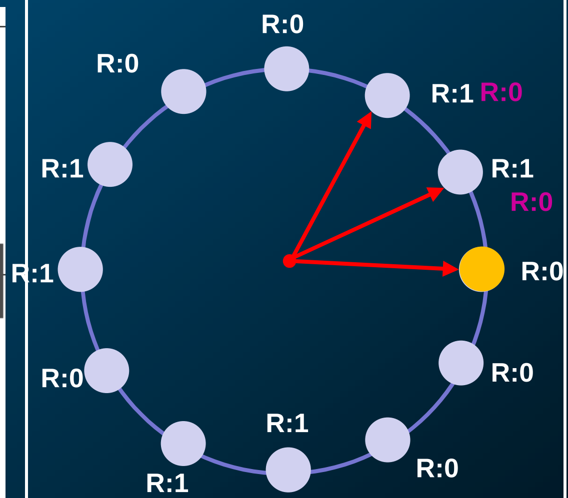


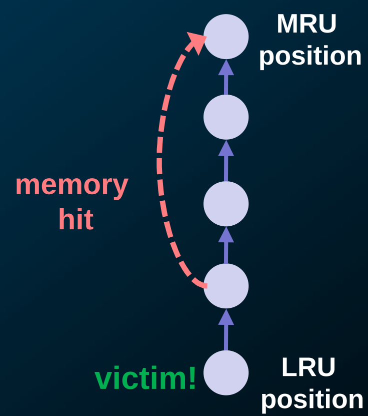
- Address translation through page table
- DRAM can be managed by OS
  → Fully associative placement is possible
  Limited reference information
  (eg. reference bit)

Comparison of Cache Replacement Problems in Each Layer

		Cache Memory	Virtual Memory System	File I/O Buffer Cache
Who manages hits/misses?	Hit	H/W	H/W	OS
	Miss	H/W	OS	OS
Representative Algorithms		Random / LRU	CLOCK	LRU
Replacement manager		H/W	OS	OS
How to Implement?		H/W implementation (Logical timestamp or bit shifting for each reference in a set)	S/W implementation supported by H/W (reference bit)	S/W implementation

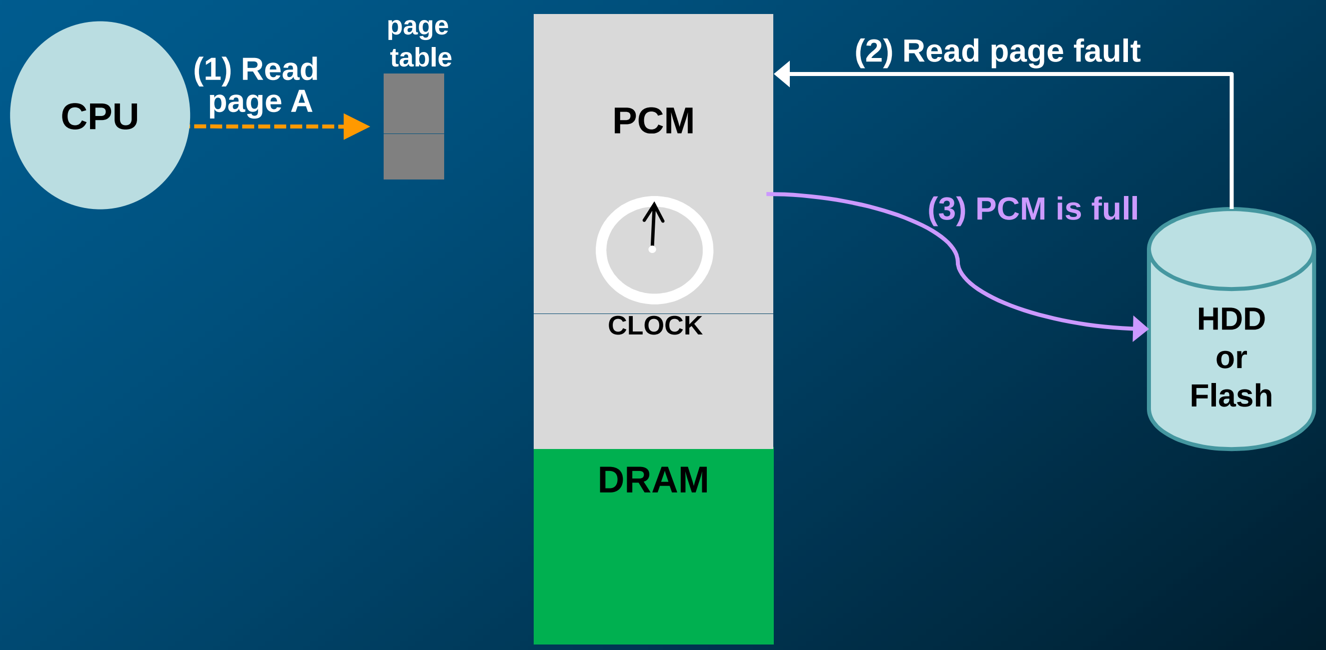






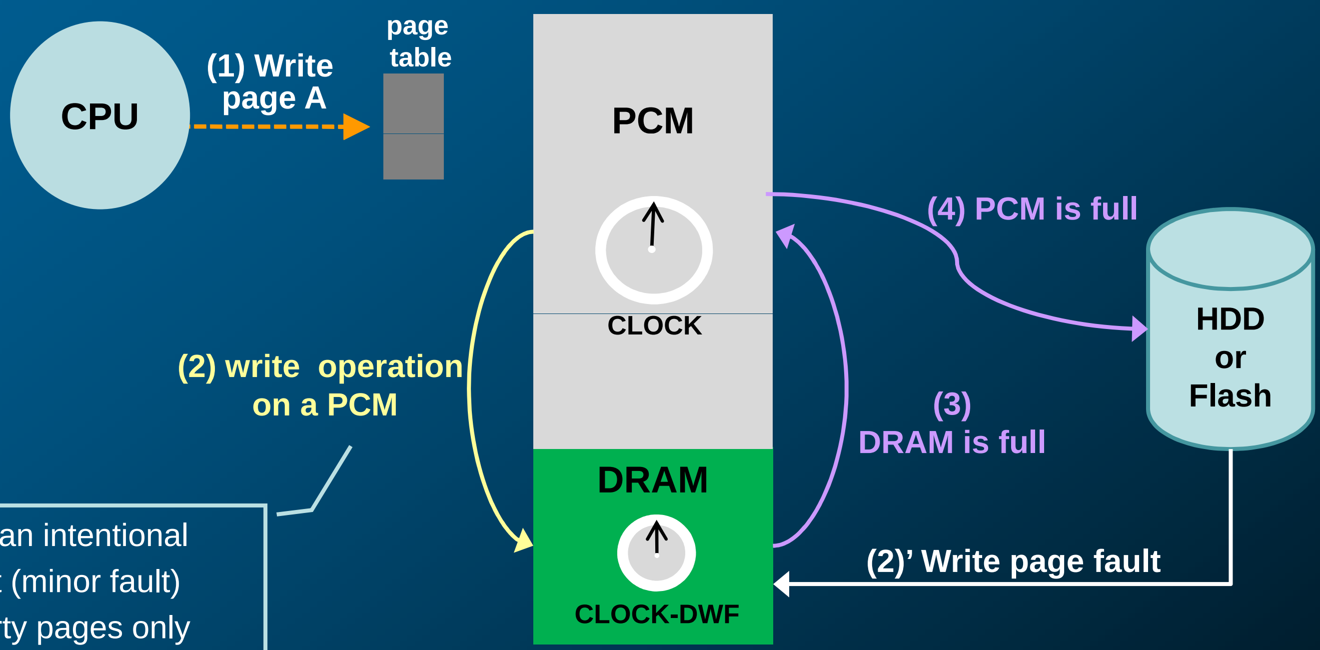
CLOCK-DWF
(Clock with Dirty bits and Write Frequency)

- CLOCK-DWF
  - Allocate read-intensive pages to PCM, write-intensive pages to DRAM.



CLOCK-DWF
(Clock with Dirty bits and Write Frequency)

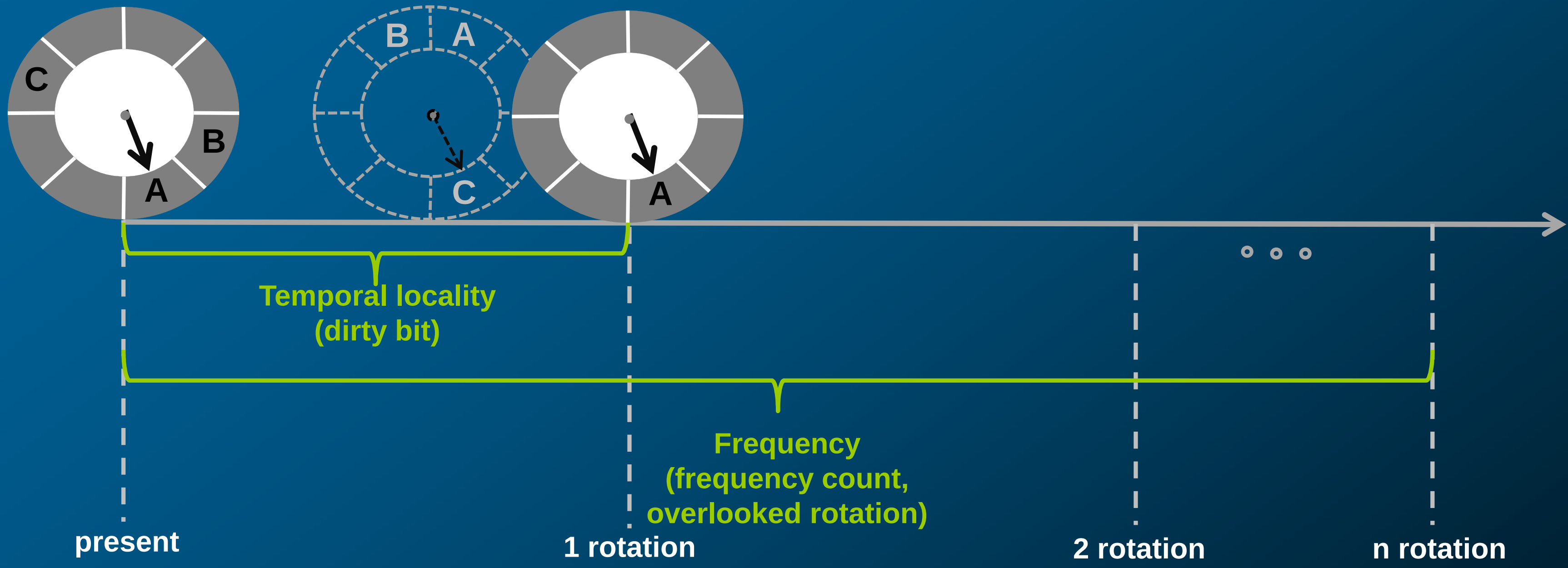
- CLOCK-DWF
  - Allocate read-intensive pages to PCM, write-intensive pages to DRAM.



- generate an intentional page fault (minor fault)
- DRAM: dirty pages only
  PCM: clean & dirty pages

CLOCK-DWF

(Clock with Dirty bits and Write Frequency)



	A	B				C			A	B				C		
Dirty	0	0				1			0	0				0		
Frequency count	5	6				4			5	6				5		
Overlooked rotation	6	1				3			7	2				0		

- **frequency count** does not indicate the real frequency but **a reset count of a dirty bit.**
  → considering *correlated references*

CLOCK-DWF
(Clock with Dirty bits and Write Frequency)

- ✓ Each page in DRAM has a dirty bit, frequency count and overlooked rotation count.
  - Dirty bit: set to 1 when a write operation occur, reset to 0 by CLOCK-DWF
  - Frequency count: increased when dirty bit become zero.
  - Overlooked rotation count: keep track of *how many times the page was overlooked.*

❖ **Victim Selection**

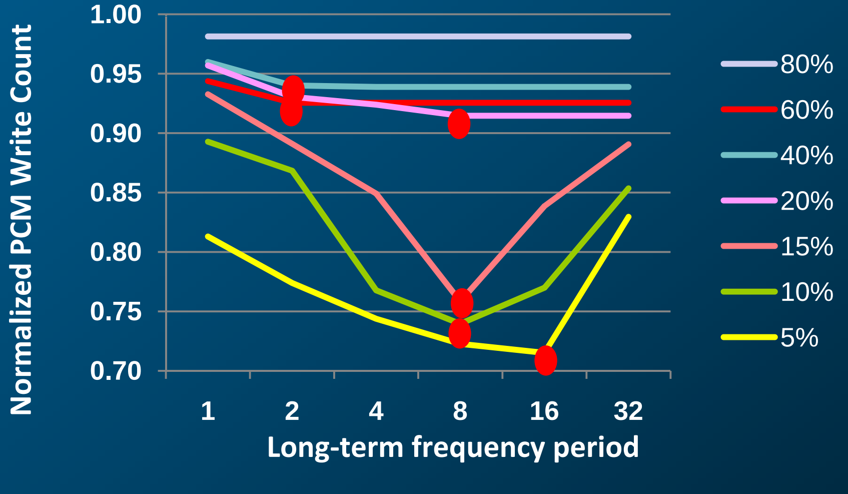
```
if  dirty_bit(page) is 0
      if  frequency(page) > Threshold   &  overlooked_rotation (page) < Expiration
            overlooked_rotation(page)++;
      else
            set dirty_bit (page)  to 1 and evict it
      end if
else  /* dirty_bit(page) is 1 */
      dirty_bit(page) = 0 ;  frequency(page)++;  overlooked_rotation(page) = 0;
end if
```

Parameter setting

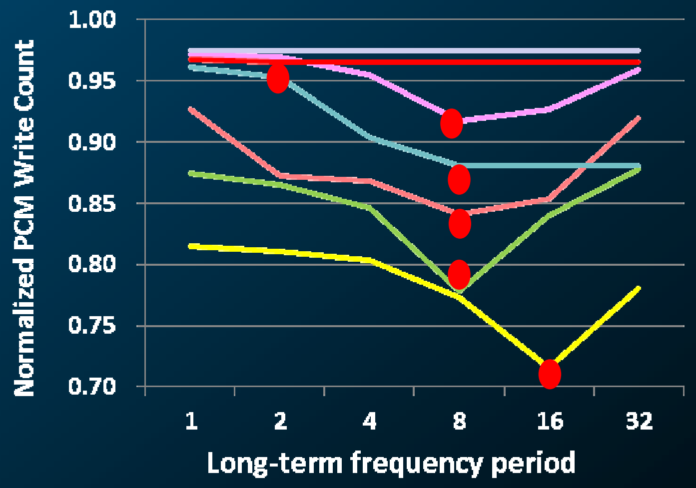
- *hot_page_threshold*
  - Determines the number of writes required for a page to be considered as a hot page.

***hot_page_threshold* ← { *hot_page_threshold* x ($SIZE_{DRAM}$ – 1) + *frequency*(p) } / $SIZE_{DRAM}$**

- *long-term frequency period*
  - Number of rotations that can be overlooked for hot pages despite not being re-written
  - When the memory size becomes large,
    - Optimal value becomes small.
    - Performance is less sensitive.



gqview trace



xmms trace

Experimental Setup

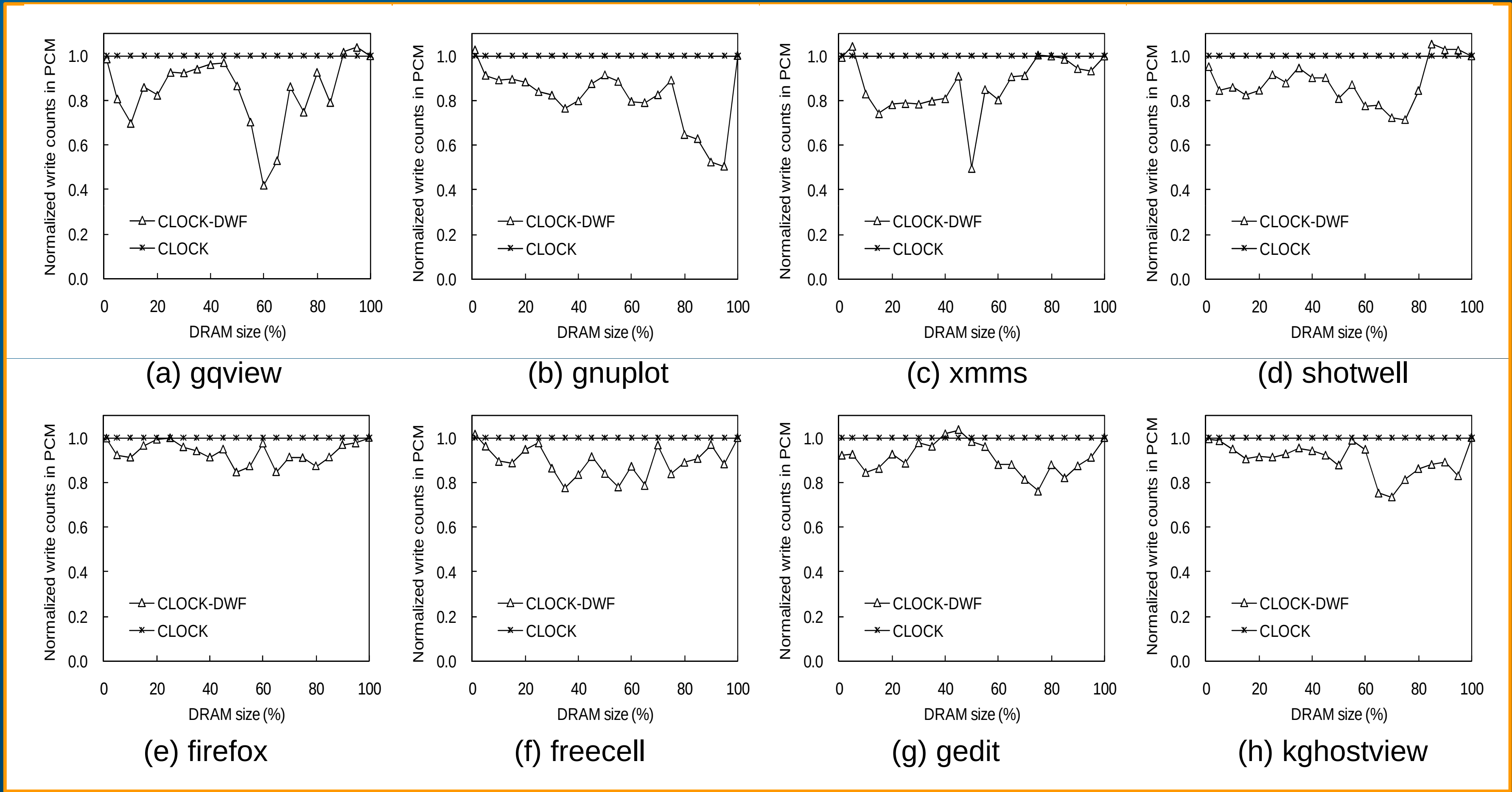
- Baseline Configuration
  - Page size: 4KB
  - Processor core: 4-core, each core runs at 2.66GHz
  - L1 I-Cache & D-Cache: 32KB, 64-byte lines, 8-way set associative
  - L2 Cache: 6MB, 64-byte lines, 24-way set associative
  - Main memory: 4GB, 8 ranks of 8 banks each
  - Hard disk drive: 5ms average access time

	DRAM	PCM
Read / Write Latency	50 / 50 ns	50 or 100 / 350 ns
Read / Write Energy	0.1 / 0.1 nJ/bit	0.2 / 1.0 nJ/bit
Static Power	1 W/GB	0.1 W/GB
Endurance	N/A	10^7

CLOCK-DWF vs. CLOCK

PCM write count

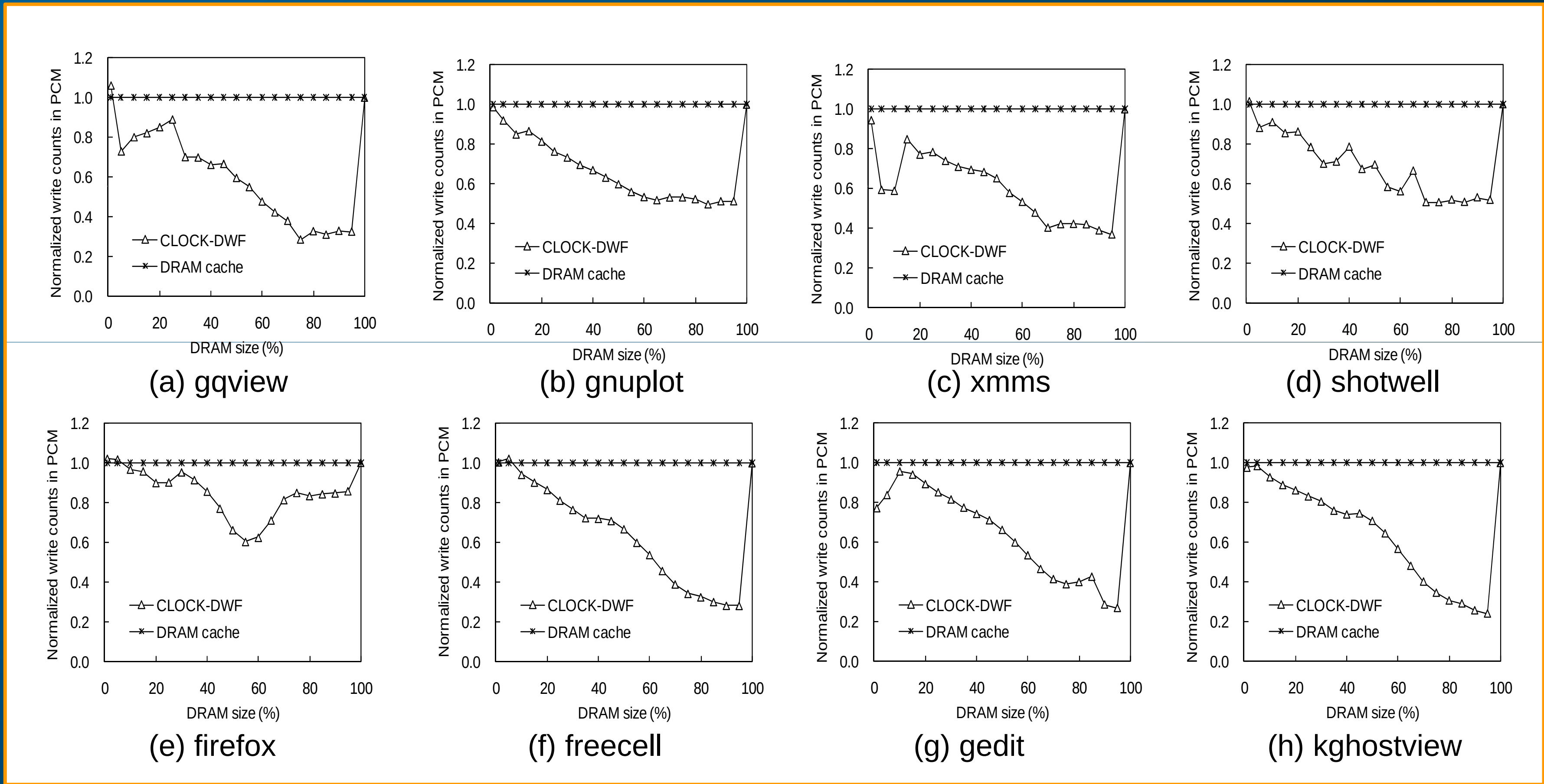
- *x*-axis: DRAM size of the maximum write memory usage of the workloads.
- *y*-axis: PCM writes of CLOCK-DWF normalized to that of CLOCK.



CLOCK-DWF VS. DRAM Cache

PCM write count

- DRAM Cache: 16-way set associative LRU
- *x*-axis: DRAM size relative to total memory footprint
- *y*-axis: # of PCM writes normalized to that of DRAM Cache



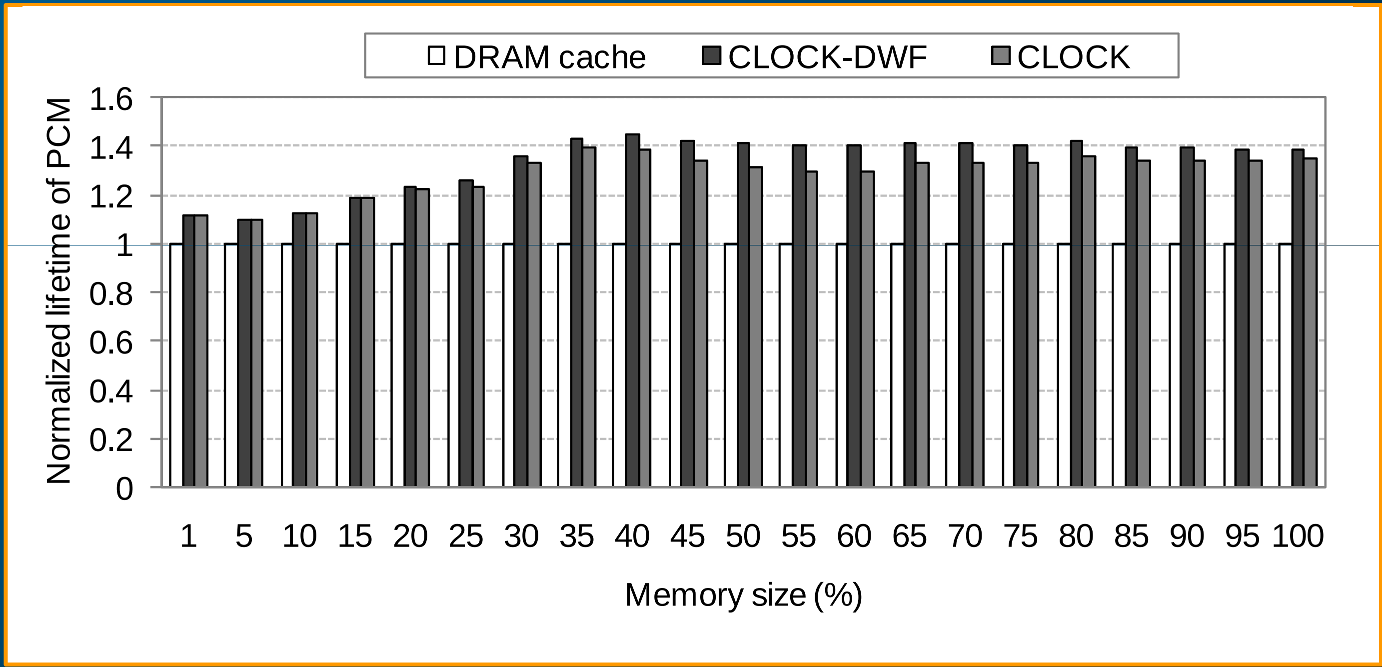
(a) gqview (b) gnuplot (c) xmms (d) shotwell

(e) firefox (f) freecell (g) gedit (h) kghostview

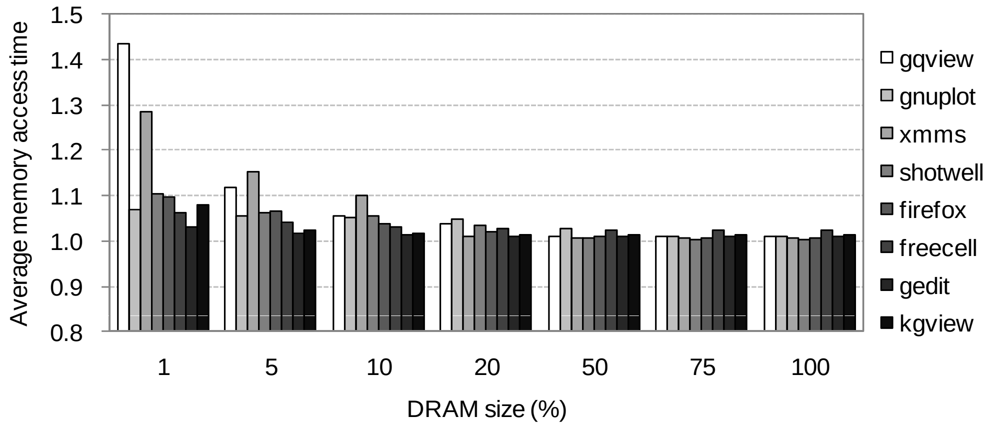
PCM Lifetime

- Sequentially execute the 8 workloads repeatedly until the write limit of PCM

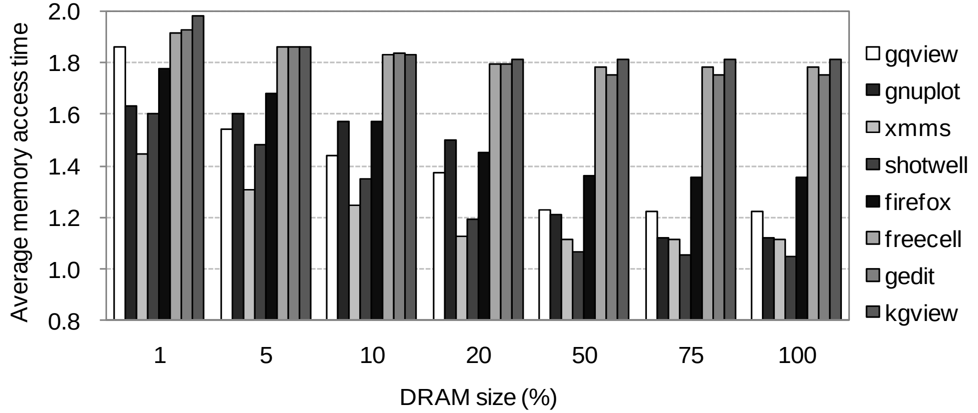
- DRAM Cache → CLOCK-DWF: 30% memory size, 4.7 years → 6.7 years
- CLOCK → CLOCK-DWF: 40~80% memory size, 5.8% extended.



CLOCK-DWF vs. Conventional System
Average memory access time



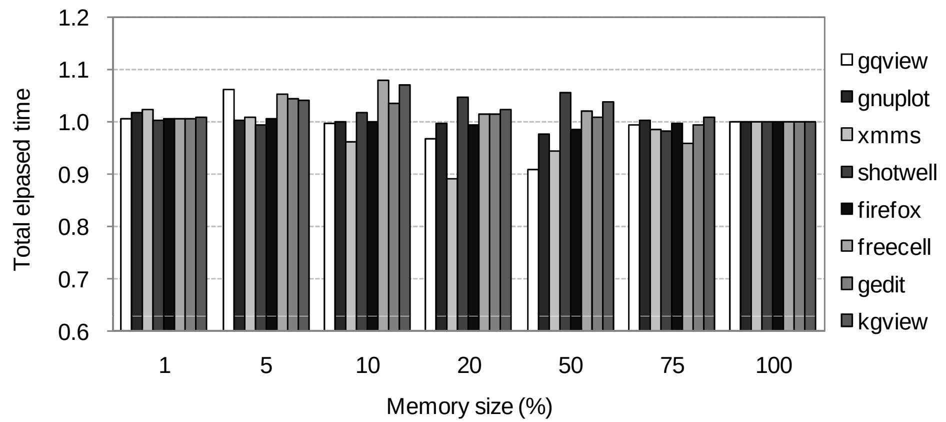
(a) *read_access_time*$_{PCM}$ = *read_access_time*$_{DRAM}$



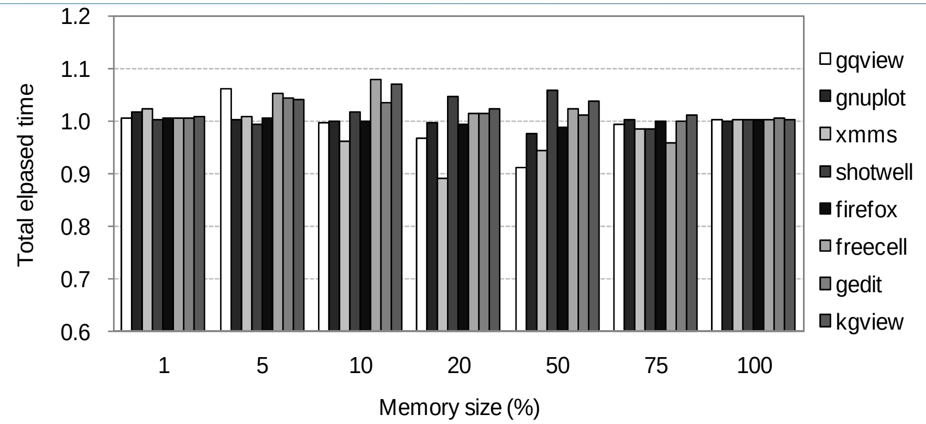
(b) *read_access_time*$_{PCM}$ = 2 x *read_access_time*$_{DRAM}$

- *x*-axis
  DRAM size of CLOCK-DWF
- *y*-axis
  Performance normalized to conventional system
- Performance degradation
  - Case (a)
    - smaller than 10%.
  - Case (b)
    - Read-intensive: 74.6%
    - Write-intensive: 31.8%

CLOCK-DWF vs. Conventional System
Total elapsed time



(a) $read\_access\_time_{PCM} = read\_access\_time_{DRAM}$



(b) $read\_access\_time_{PCM} = 2 \times read\_access\_time_{DRAM}$

- *x*-axis
  - CLOCK-DWF
    - DRAM:PCM = 1:9
  - Conventional system
    - DRAM only
- *y*-axis

  Performance normalized to conventional system
- Performance degradation
  - less than 8%
  - due to large page fault overhead

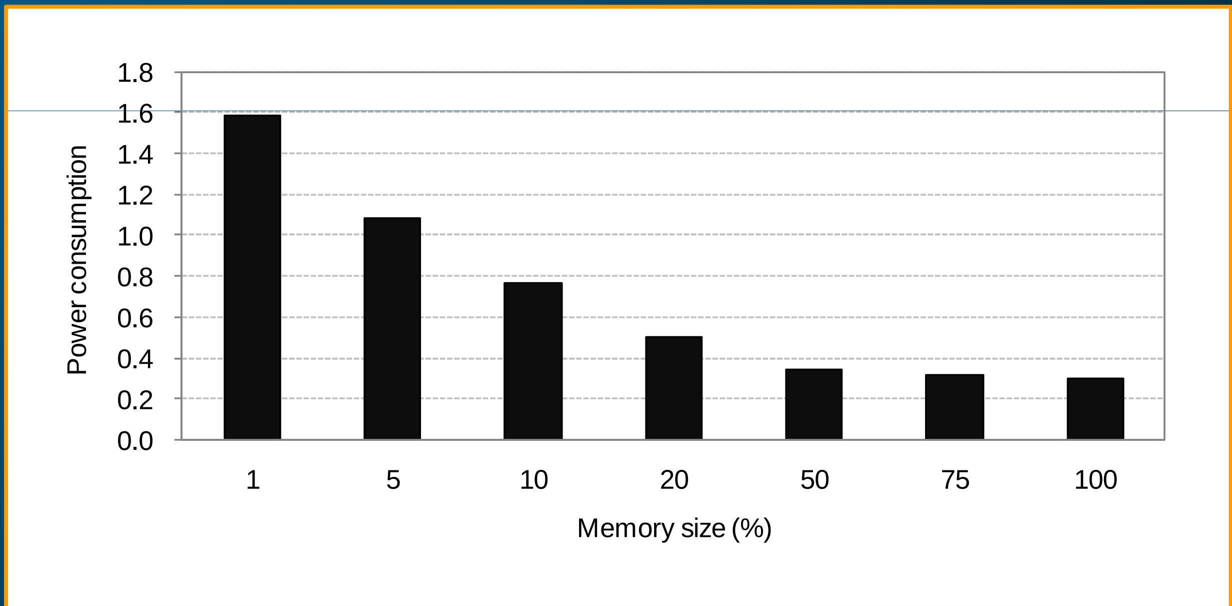
CLOCK-DWF vs. Conventional System

Power consumption

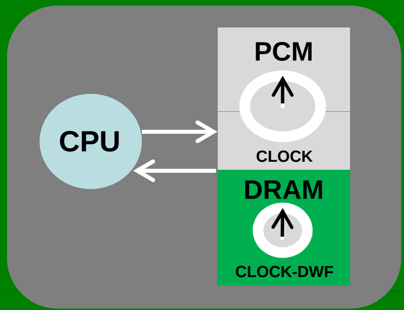
- Power consumption

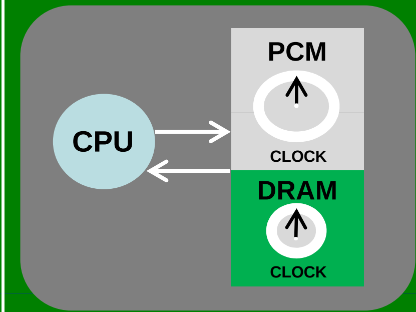
	DRAM	PCM
Read / Write Energy	0.1 / 0.1 nJ/bit	0.2 / 1.0 nJ/bit
Static Power	1 W/GB	0.1 W/GB

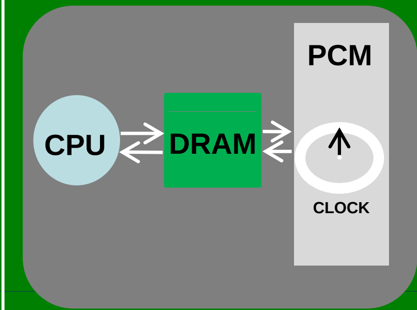
- Power-savings become large as memory size increases.
  → Static power accounts for a large portion.



Summary







	CLOCK-DWF	CLOCK	DRAM Cache
Memory architecture	DRAM + PCM memory	DRAM + PCM memory	DRAM Cache, PCM memory
DRAM usage	write	write	read / write
DRAM Replacement Policy	CLOCK-DWF (fully associative)	CLOCK (fully associative)	LRU (16-way set associative)
Temporal locality	O	O	O
Frequency	O	X	X
Write counts on PCM	0.65~0.24	0.76~0.57	1

Access Information

- If you want to cite this material, please contact the following information.
  - http://home.ewha.ac.kr/~bahn
  - bahn@ewha.ac.kr