Exploiting Managed Language Semantics to Mitigate Wear-out in Persistent Memory

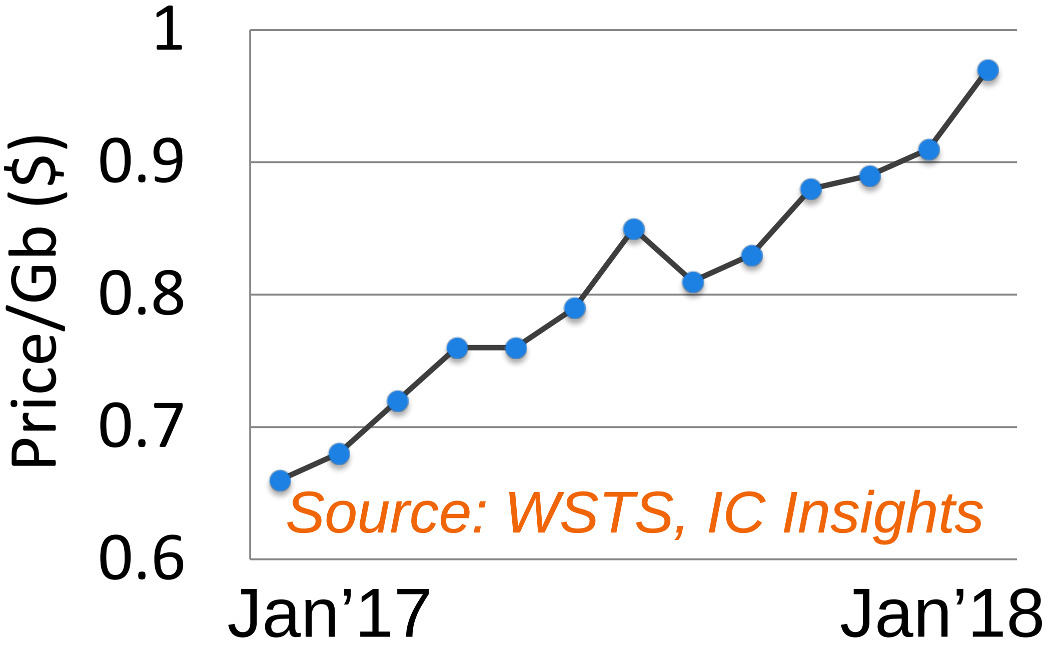
Shoaib Akram

Ghent University, Belgium

Main memory capacity expansion

Charge storage in DRAM a scaling limitation

Manufacturing complexity makes DRAM pricing volatile



Phase change memory (PCM)

Scalable → More Gb for the same price

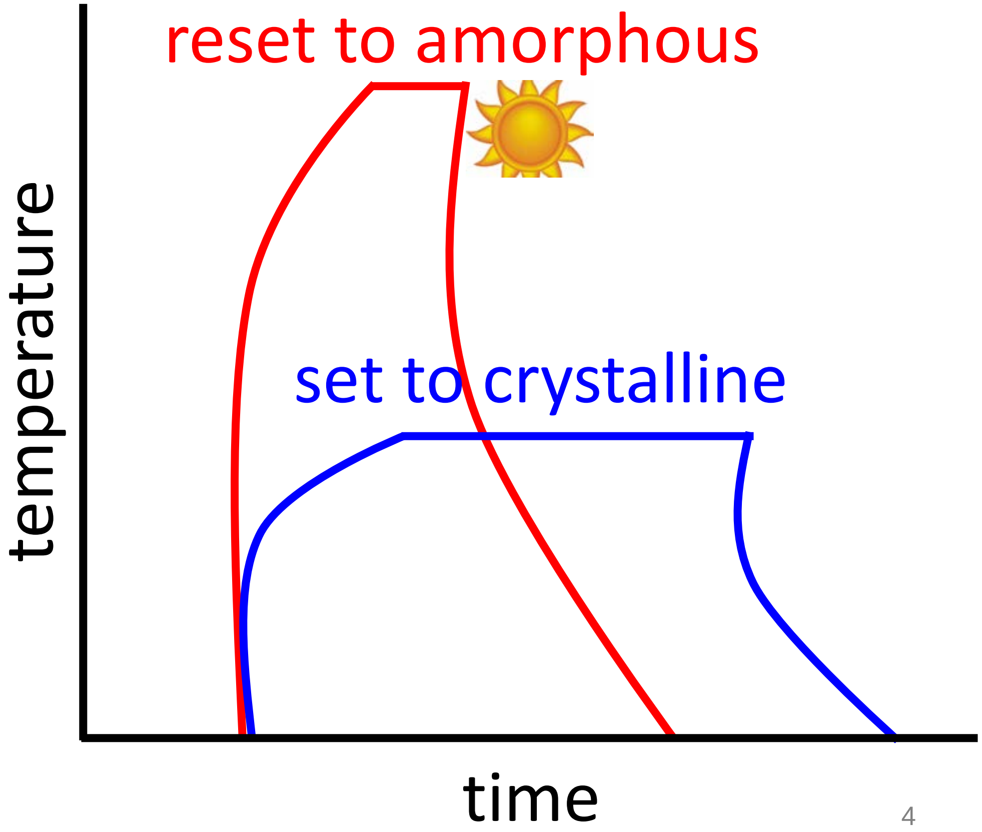
Byte addressable like DRAM

Latency closer to DRAM

☹ Low write endurance

Why PCM has low write endurance?

Electric pulses to program PCM cells wear them out over time



Hybrid DRAM-PCM memory

PCM alone as a DRAM replacement wears out in a few months for popular Java applications

DRAM	PCM
Endurance	Capacity Persistence

Mitigating PCM wear-out

Wear-leveling to spread writes across PCM

This talk → Use DRAM to limit PCM writes

OS to limit PCM writes

Coarse-grained page migrations hurt application performance and PCM lifetime

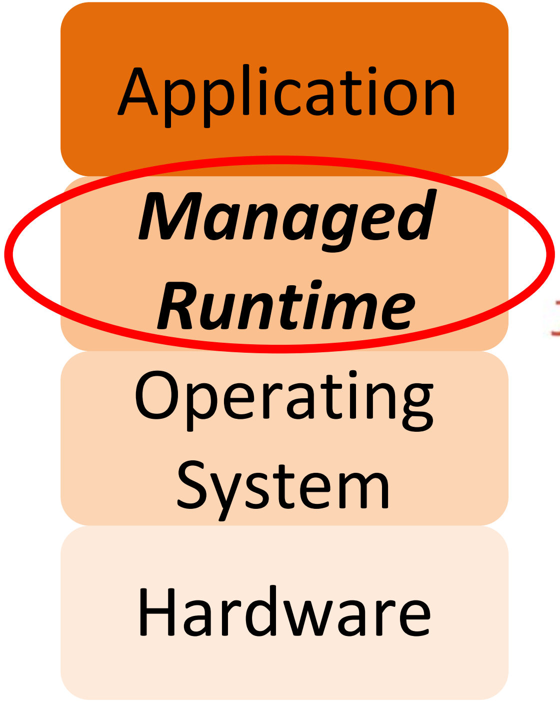
Managed runtimes

Platform independence

- Abstract hardware/OS
- → Aka *Virtual Machine*

Ease programmer’s burden

- Garbage collection
- Security



GC to limit PCM writes

GC understands memory semantics

GC approaches are *pro-active* and *fine-grained*

Application

Operating System

Hardware

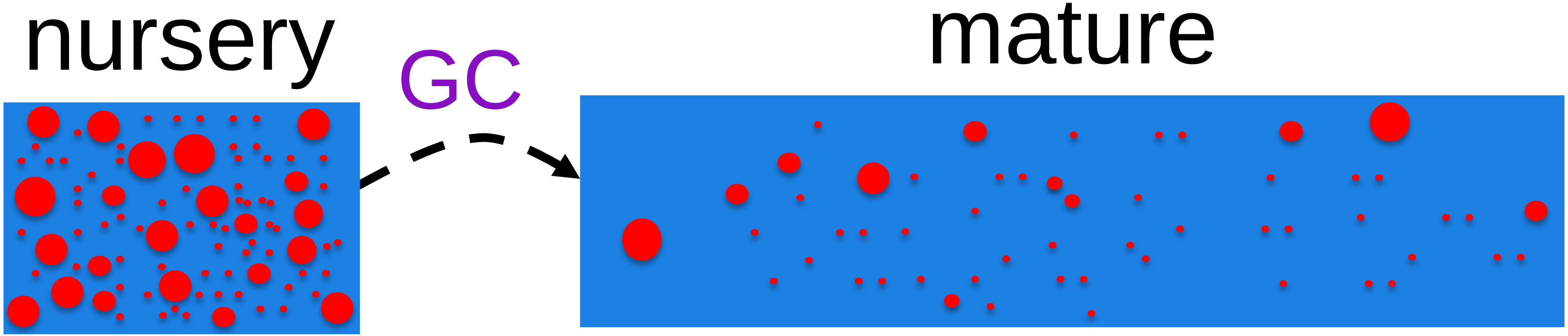
Write Distribution in GC heap

70%

of writes

Flash Memory Summit

Write Distribution in GC heap



70%

of writes

22%

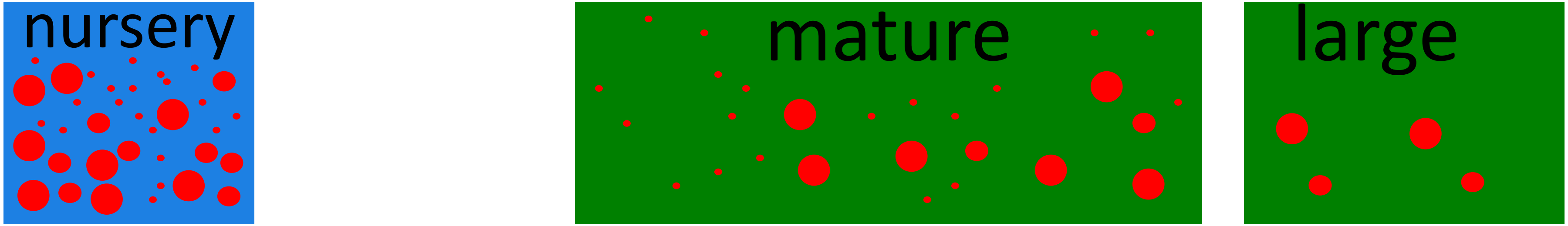
to 2% of objects

Write-Rationing Garbage Collection

Limit PCM writes by discovering highly written objects

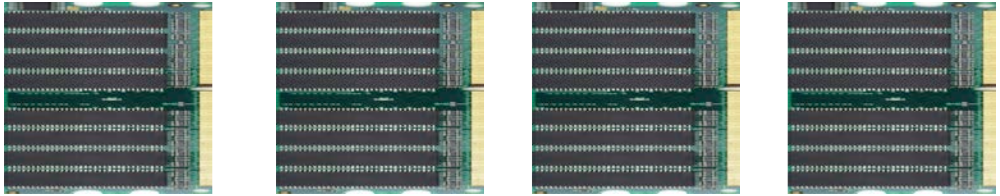
Kingsguard dynamically monitor writes

Kingsguard-Nursery (KG-N)

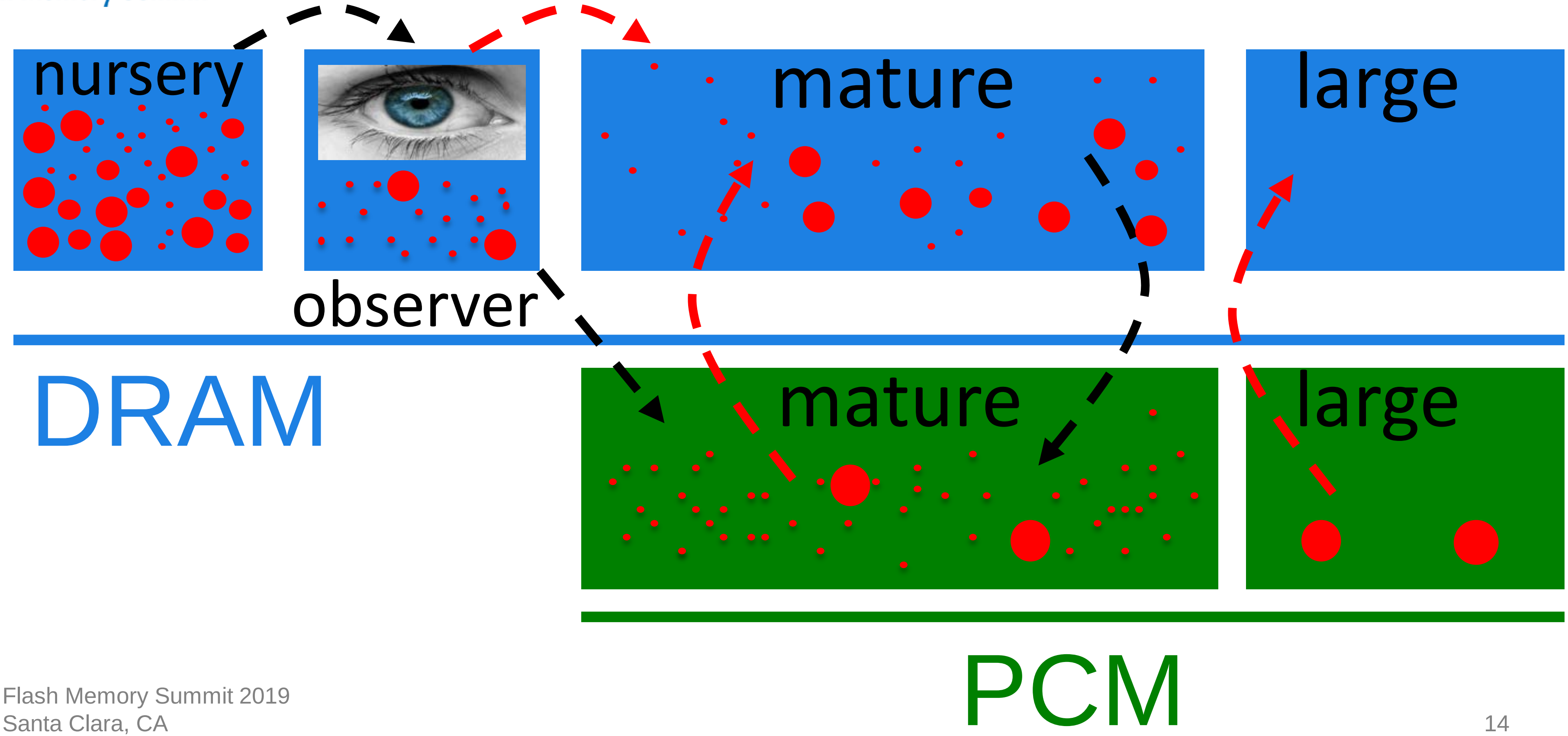


DRAM

PCM



Kingsguard-Writers (KG-W)



KG-W drawbacks

Overhead of dynamic monitoring

Limited time window to predict write intensity

Excessive & fixed DRAM consumption

Write-Rationing Garbage Collection

Limit PCM writes by discovering highly written objects

Kingsguard dynamically monitor writes

Crystal Gazer statically profiles objects

Allocation site as a write predictor

```
a = new Object()
b = new Object()
c = new Object()
d = new Object()
```

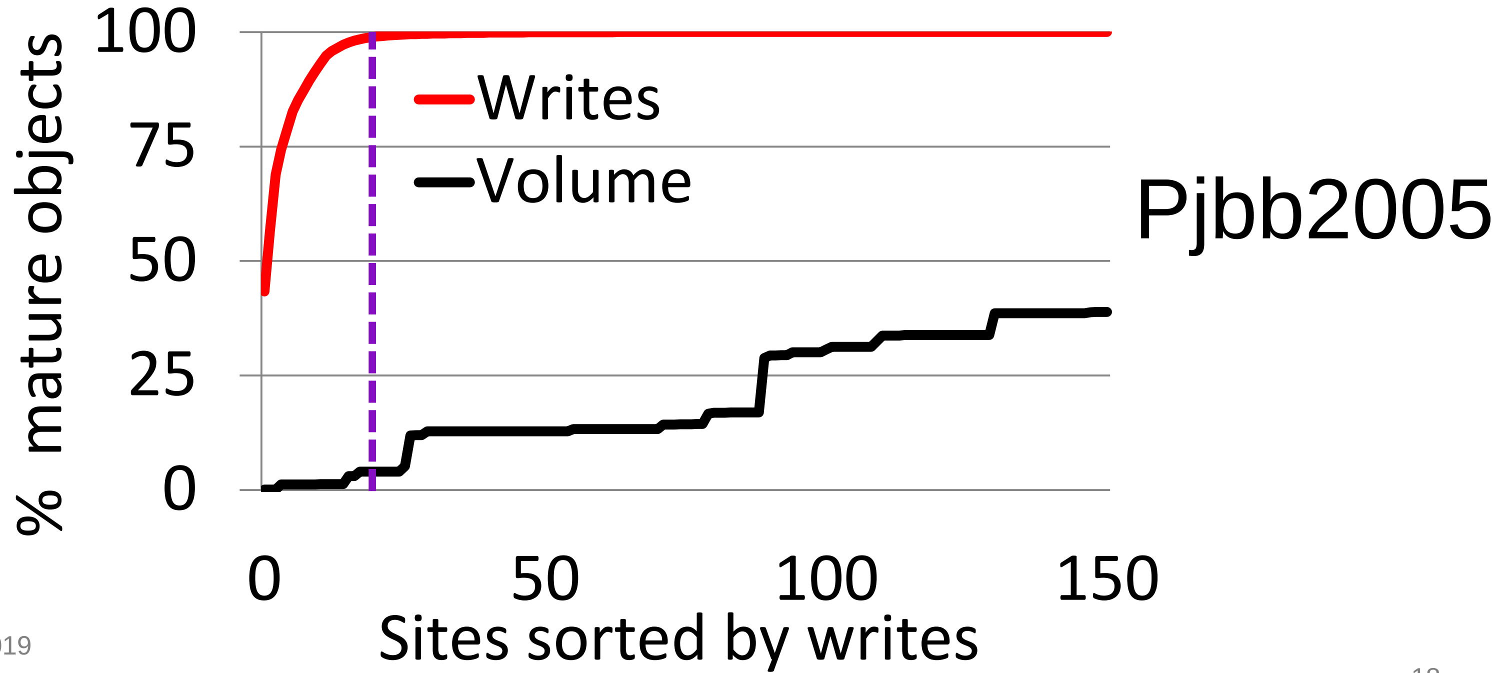
```
a = new Object()
b = new Object()
c = new Object()
d = new_dram Object()
```

Uniform distribution ☹

Skewed distribution ☺

Write distribution by allocation site

Few sites capture majority of writes



Crystal Gazer operation

Application Profiling → Advice Generation → Bytecode Compilation Object Placement

```
a = new Object()
…
b = new Object()
```

```
a = new Object()
…
b = new_dram Object()
```

Advice generation

Goal: Generate <alloc-site, advice> pairs

advice → DRAM or PCM

input is a write-intensity trace

Two heuristics to classify allocation sites as DRAM or PCM

Classification heuristic (1)

Freq: A *threshold* % of objects from a site get more than a *threshold* # writes → DRAM

☺ Aggressively limits PCM writes

☹ No distinction based on object size

Classification heuristic (2)

Write density → Ratio of # writes to object size

Dens: A *threshold* % of objects from a site have more than a *threshold* write density → DRAM

Classification thresholds

Homogeneity threshold → 1%

Frequency threshold → 1

Density threshold → 1

Classification examples

Frequency threshold = 1

PCM writes = ?, DRAM bytes = ?

Object Identifier	# Writes	# Bytes	Allocation site
O1	0	4	A() + 10
O2	0	4	A() + 10
O3	128	4	A() + 10
O4	128	4096	B() + 4

Classification examples

Frequency threshold = 1

PCM writes = ?, DRAM bytes = ?

	Object Identifier	# Writes	# Bytes	Allocation site
	O1	0	4	A() + 10
	O2	0	4	A() + 10
→	O3	128	4	A() + 10
→	O4	128	4096	B() + 4

Classification examples

Frequency threshold = 1

PCM writes = 0/256, DRAM bytes = 5008

Object Identifier	# Writes	# Bytes	Allocation site
O1	0	4	A() + 10
O2	0	4	A() + 10
→ O3	128	4	A() + 10
→ O4	128	4096	B() + 4

Classification examples

Density threshold = 1

PCM writes = ?, DRAM bytes = ?

Object Identifier	# Writes	# Bytes	Allocation site
O1	0	4	A() + 10
O2	0	4	A() + 10
O3	128	4	A() + 10
O4	128	4096	B() + 4

Classification examples

Density threshold = 1

PCM writes = ?, DRAM bytes = ?

Object Identifier	# Writes	# Bytes	Allocation site
O1	0	4	A() + 10
O2	0	4	A() + 10
O3	128	4	A() + 10
O4	128	4096	B() + 4

→ 32

Classification examples

Density threshold = 1

PCM writes = ?, DRAM bytes = ?

Object Identifier	# Writes	# Bytes	Allocation site
O1	0	4	A() + 10
O2	0	4	A() + 10
O3	128	4	A() + 10
O4	128	4096	B() + 4

→ <1

Classification examples

Density threshold = 1

PCM writes = 128/256, DRAM bytes = 12

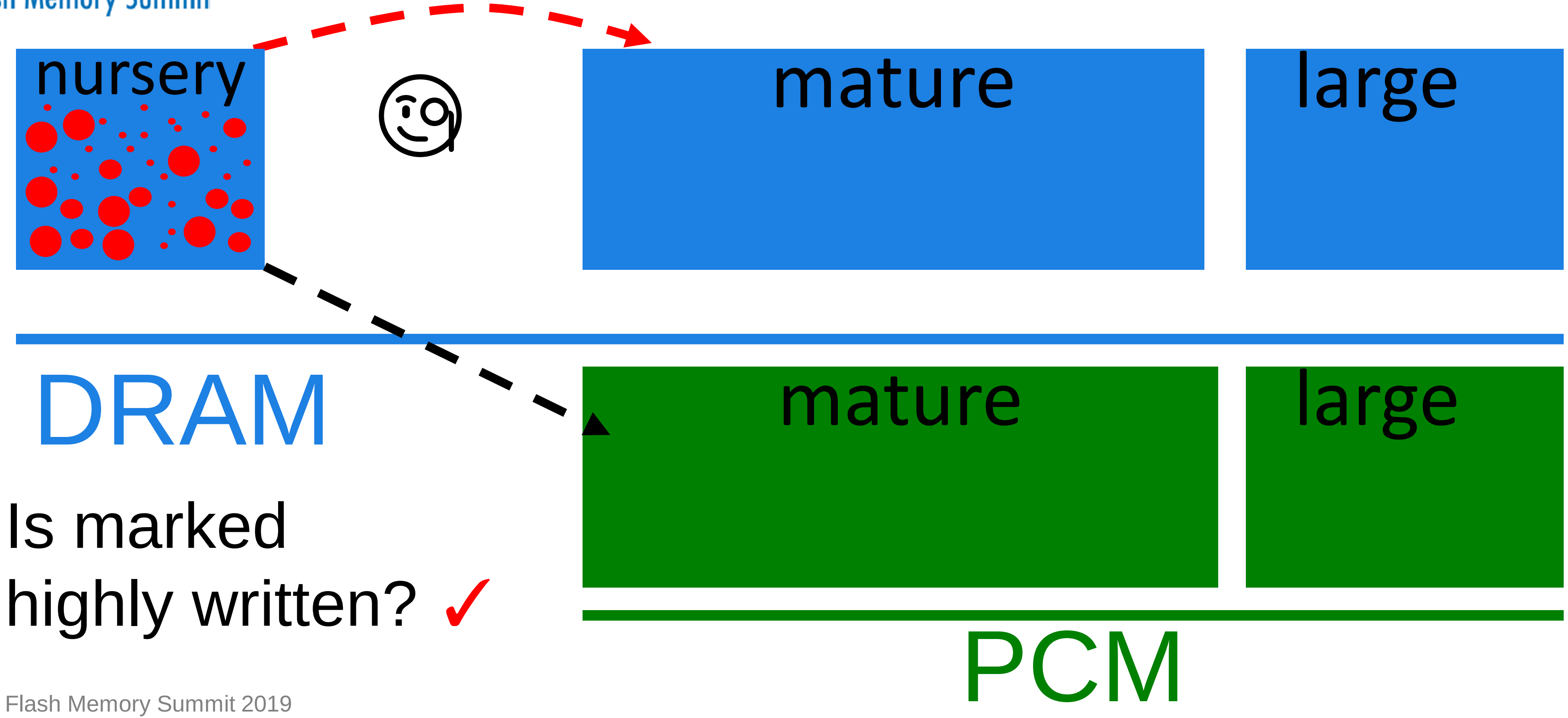
Object Identifier	# Writes	# Bytes	Allocation site
O1	0	4	A() + 10
O2	0	4	A() + 10
O3	128	4	A() + 10
O4	128	4096	B() + 4

Object placement in Crystal Gazer

new_dram() → Set a bit in the object header

GC → Inspect the bit on nursery collection to copy object in DRAM or PCM

Object placement in Crystal Gazer



Is marked highly written? ✓

Key features of Crystal Gazer

Eliminates overhead of dynamic monitoring

Less mispredictions due to pro-active nature

Pareto optimal trade-offs b/w capacity and lifetime

Evaluation methodology

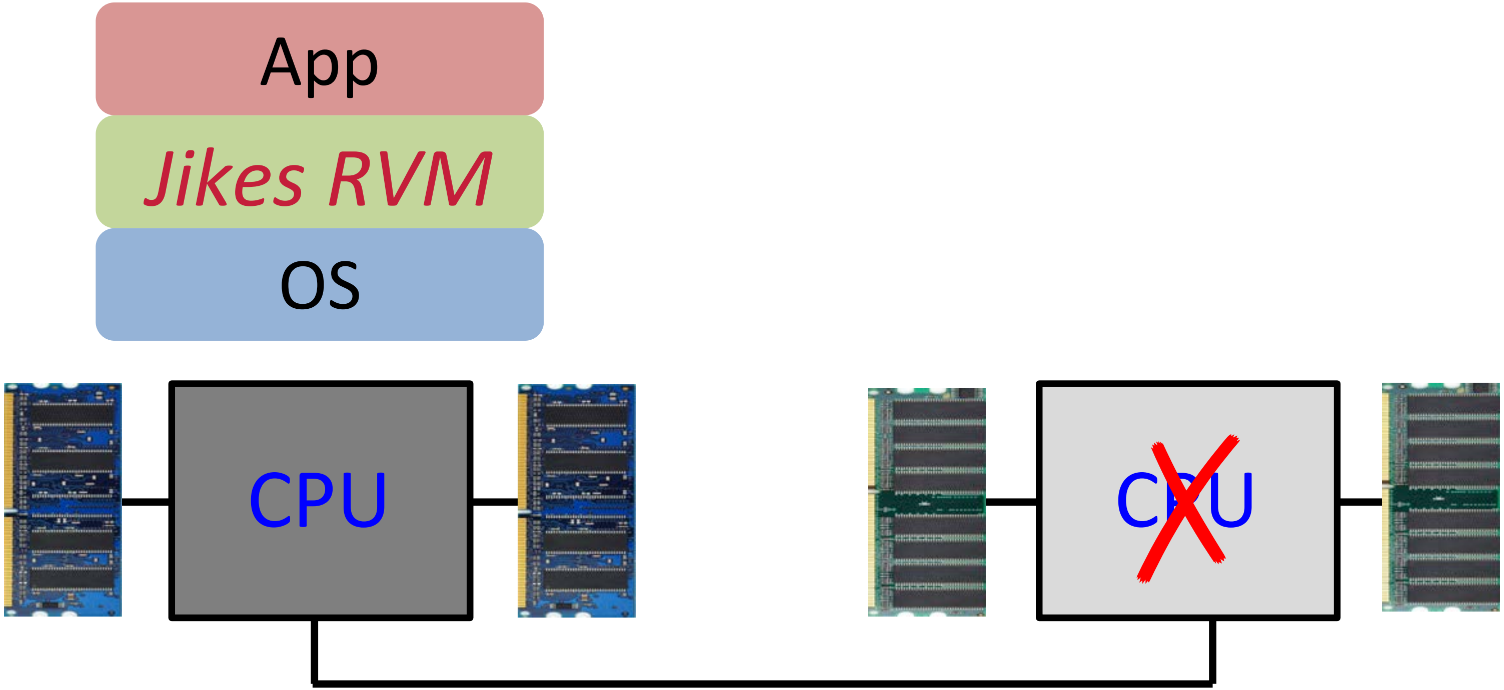
15 Applications → DaCapo, GraphChi, SpecJBB

Medium-end server platform

Different inputs for production and advice

Jikes RVM

Emulation platform

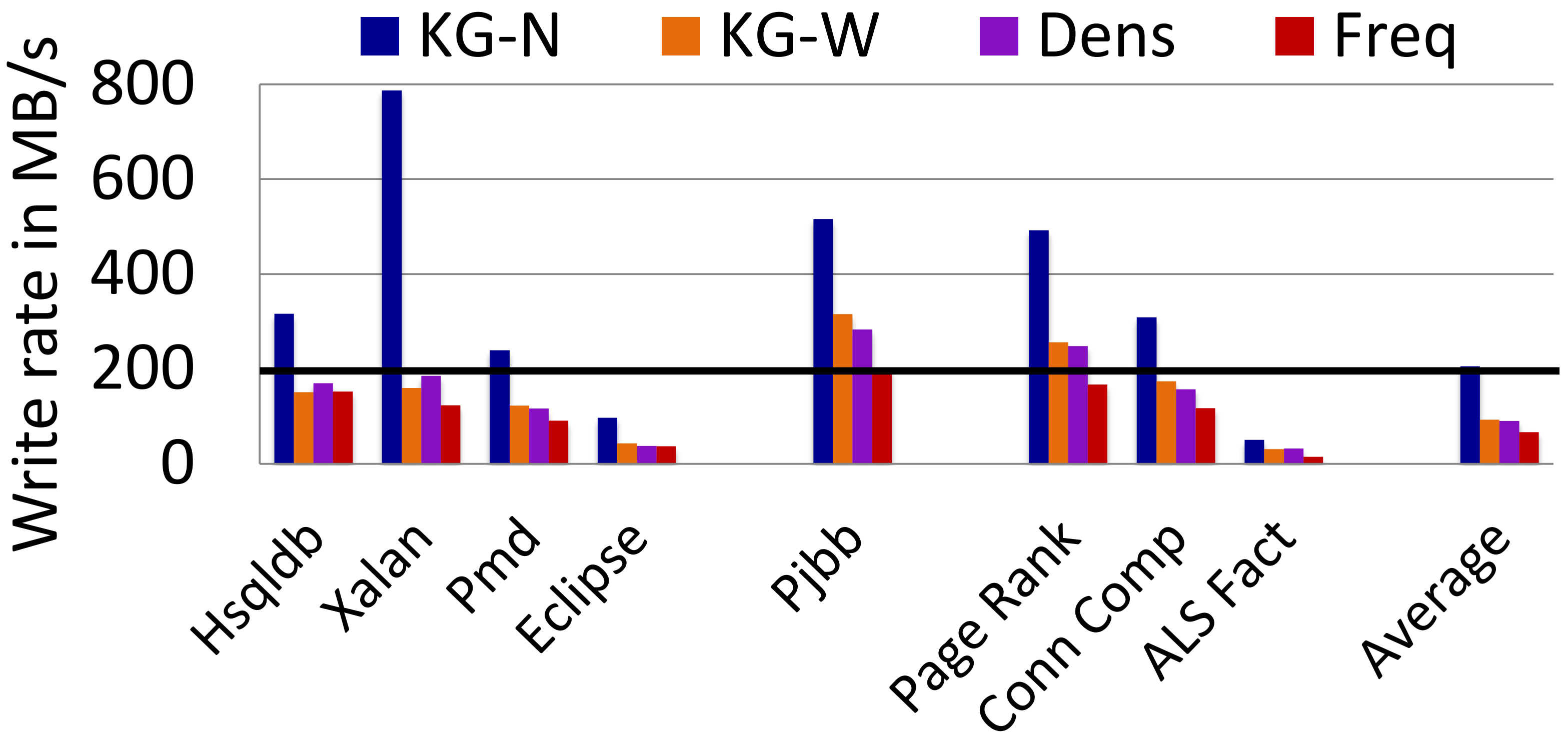


PCM write rates

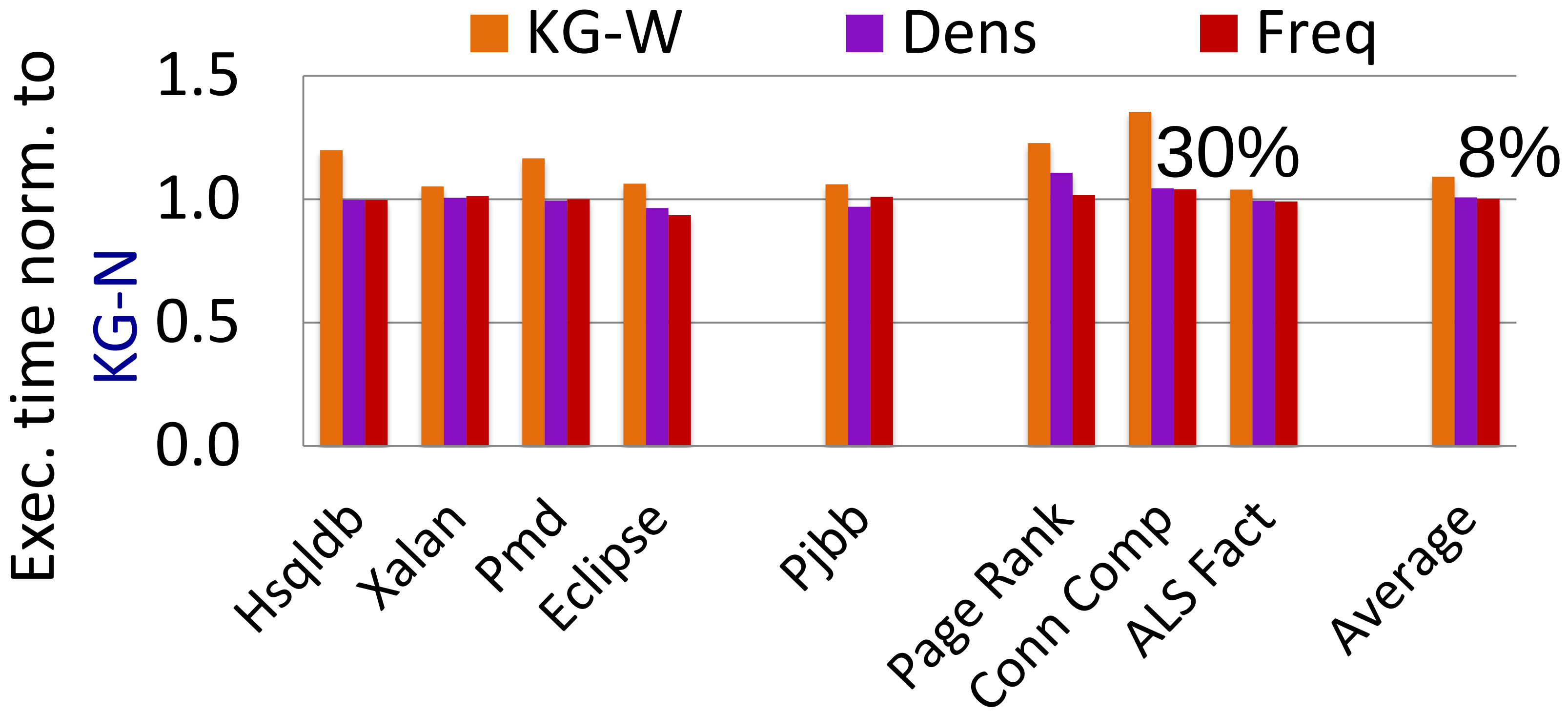
PCM-Only write rate is above 1 GB/s on average

Safe operation is 200 MB/s

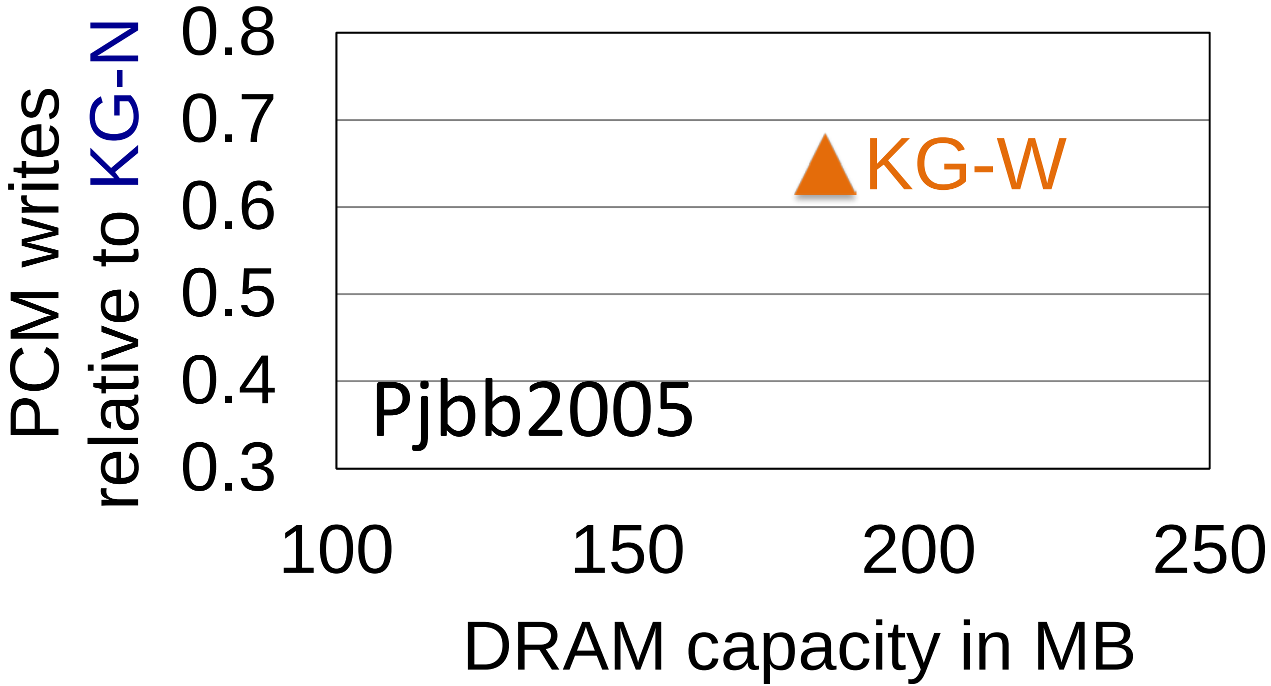
PCM write rates



Execution time

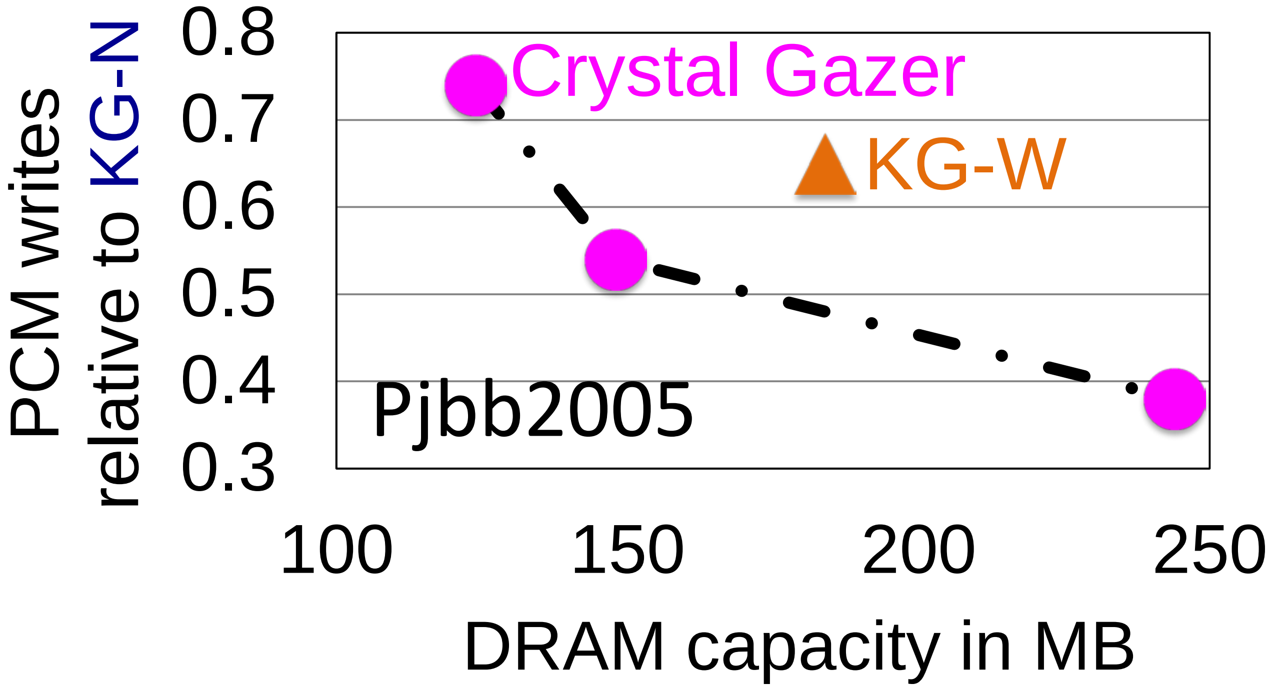


KG-W versus Crystal Gazer



KG-W versus Crystal Gazer

Crystal Gazer opens up Pareto-optimal trade-offs



Write-rationing garbage collection

Hybrid memory is inevitable

DRAM | PCM

All layers can contribute to manage hybrid memory

Write-rationing GC is pro-active and fine-grained