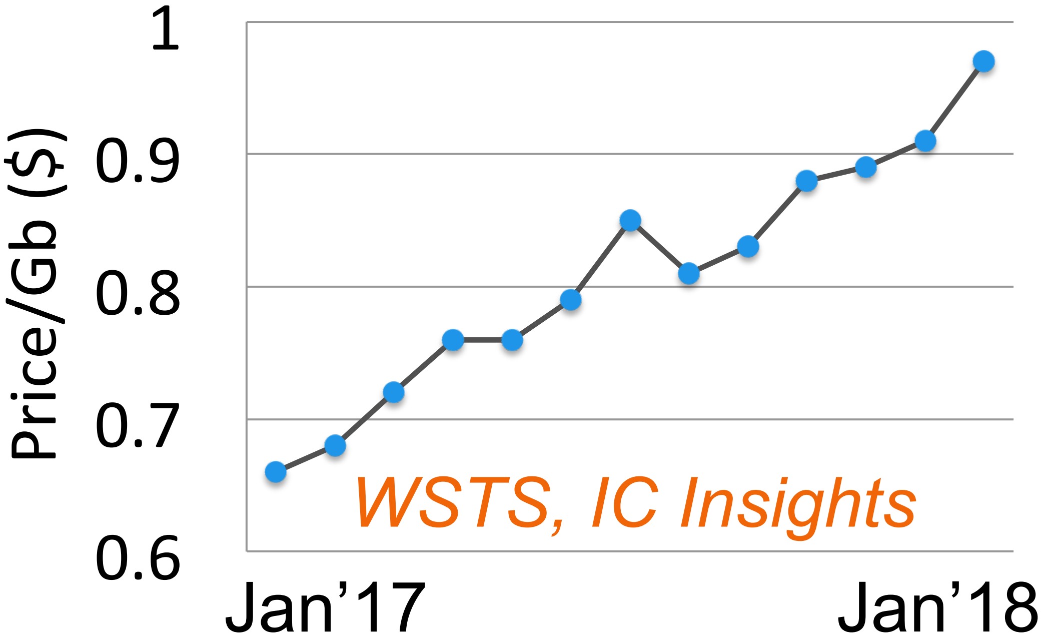
Exploiting Managed Language Semantics to Mitigate Wear-out in Persistent Memory

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)

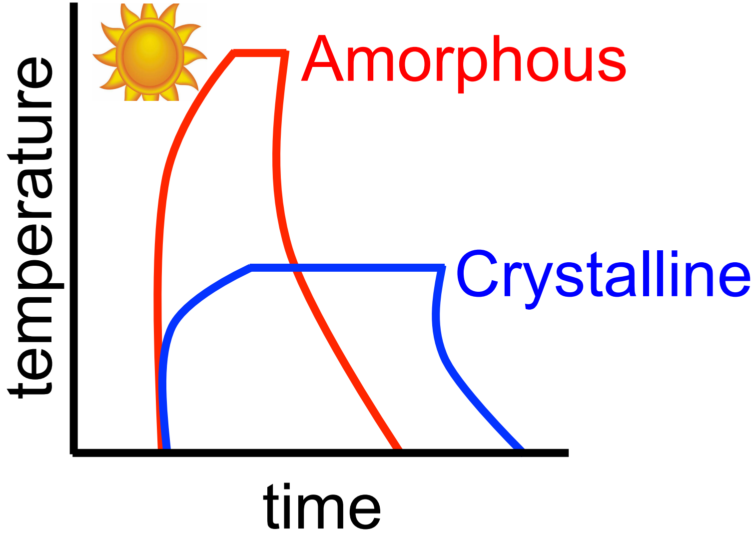
- 🙂 Scalable → More Gb for the same price
- Byte addressable like DRAM
- Latency closer to DRAM
- 🙁 Low write endurance

Why PCM has low write endurance?

Store information as change in resistance

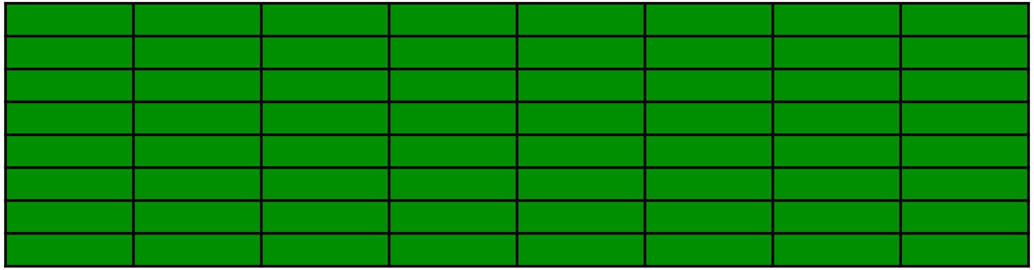
Crystalline is set & Amorphous is reset

Electric pulses to program PCM cells wear them out



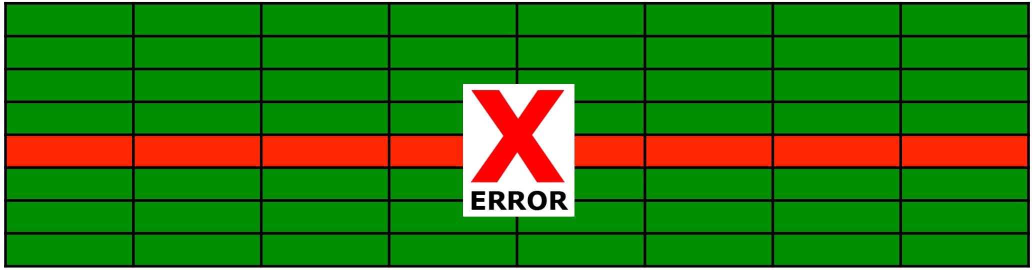
Mitigating PCM wear-out

Wear-leveling to spread writes across PCM



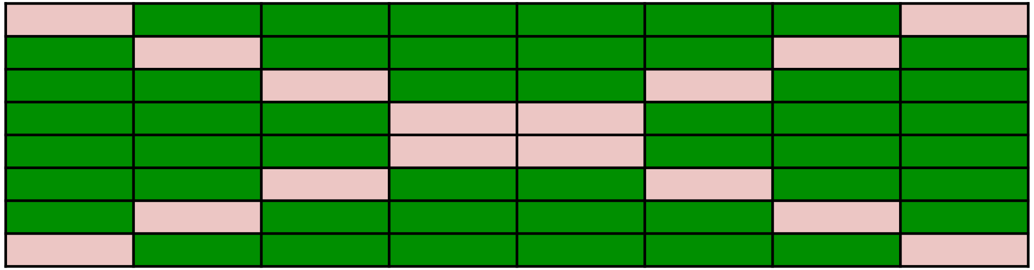
Mitigating PCM wear-out

Wear-leveling to spread writes across PCM



Mitigating PCM wear-out

Wear-leveling to spread writes across PCM



Problem: PCM-Only with wear-leveling wears out in a few months

Flash Memory Summit

Hybrid DRAM-PCM memory

Endurance

Capacity
Persistence

DRAM

PCM

This talk → Use DRAM to limit PCM writes

OS to limit PCM writes

Page migrations hurt performance and PCM lifetime

Managed runtimes

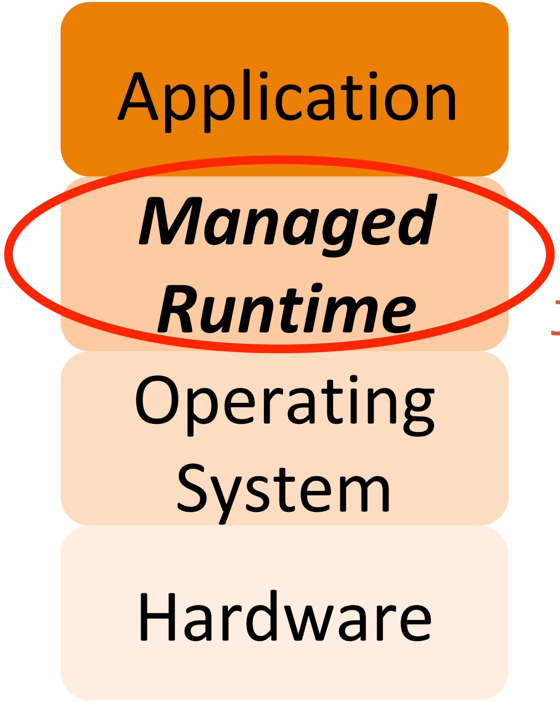
Platform independence

Abstract hardware/OS

→ Aka *Virtual Machine*

Ease programmer's burden

Garbage collection (GC)



GC to limit PCM writes

GC aware of heap semantics
→ Pro-active allocation

GC operates with objects
→ Fine-grained mgmt.

Application

Operating System

Hardware

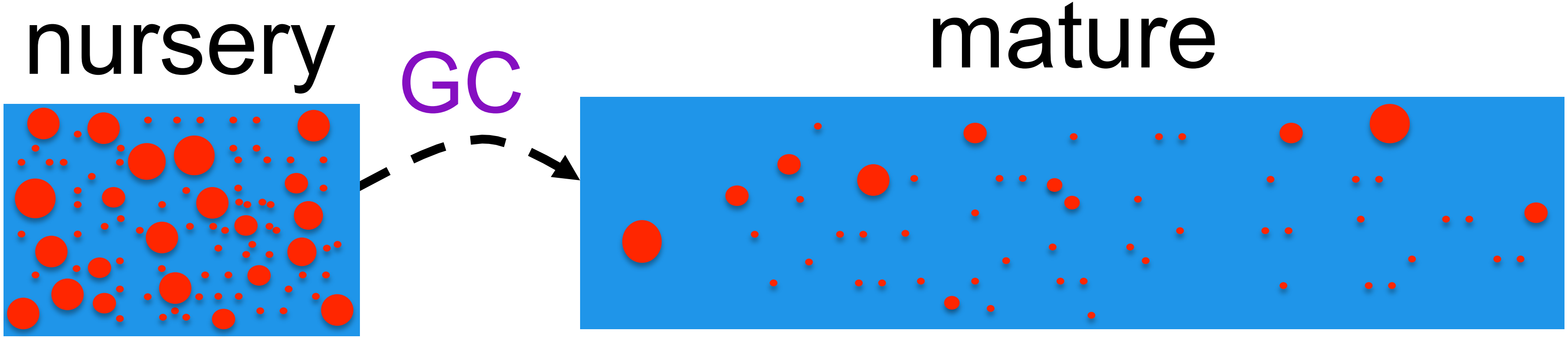
Java

Write Distribution in GC heap

70%

of writes

Write Distribution in GC heap



70%

of writes

22%

to 2% of objects

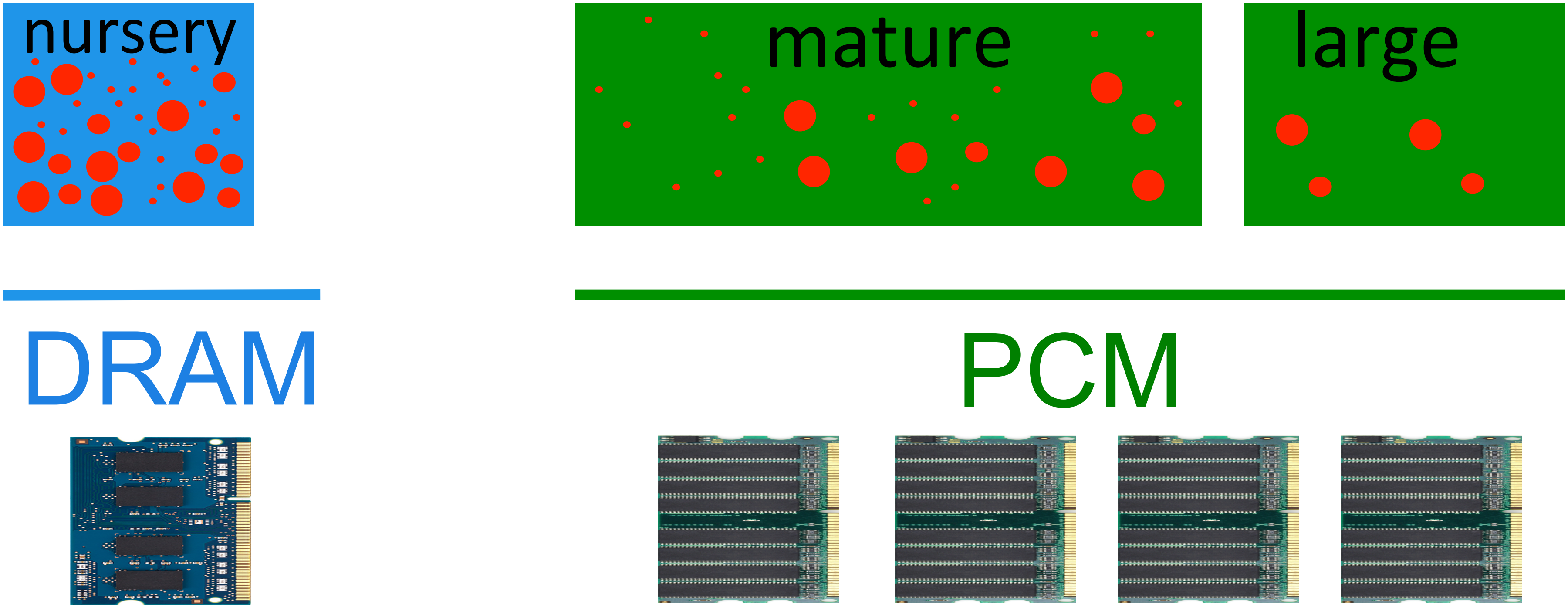
Write-Rationing Garbage Collection

Limit PCM writes by discovering highly written objects

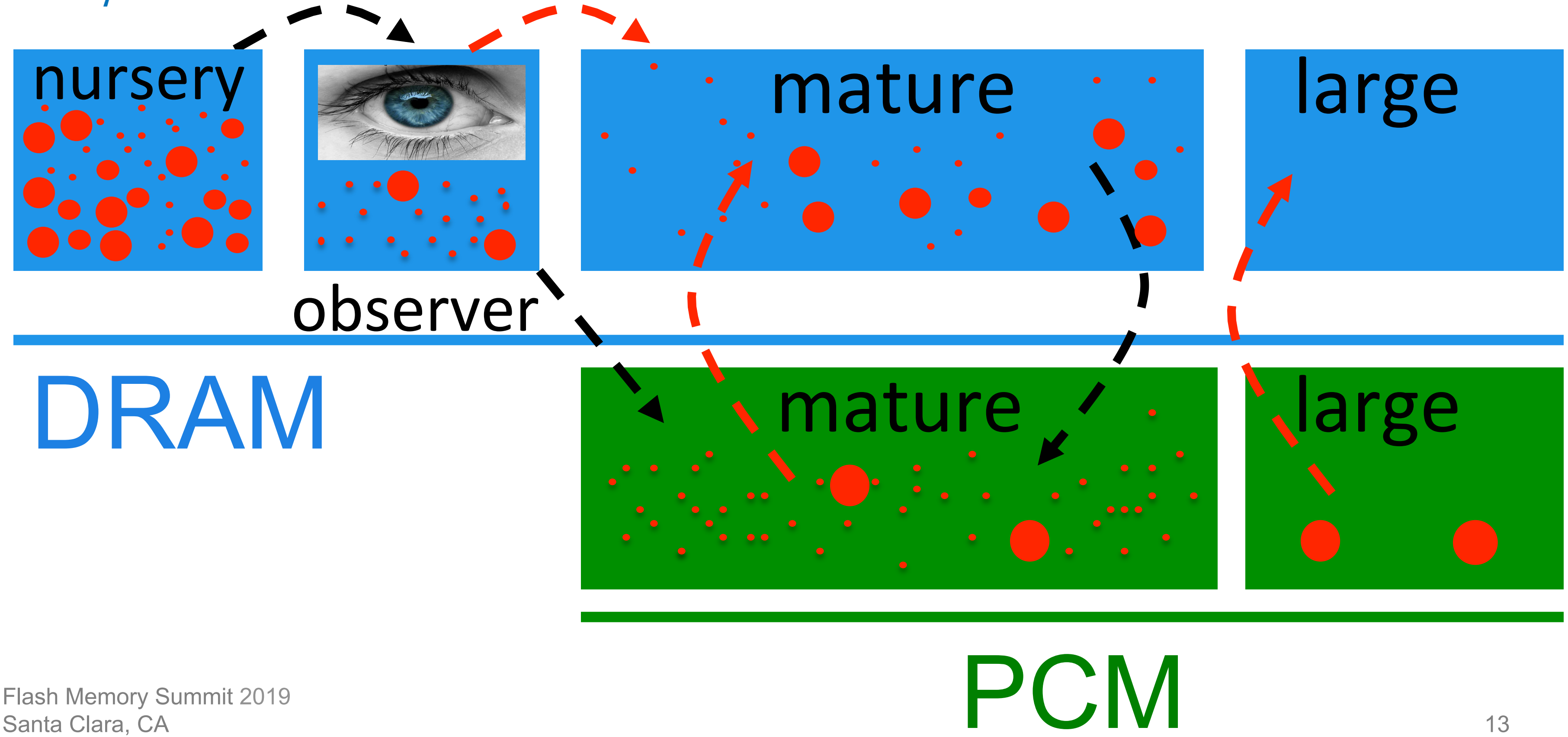
Kingsguard → dynamic monitoring

Crystal Gazer → prediction

Kingsguard-Nursery (KG-N)



Kingsguard-Writers (KG-W)



Metadata optimization

Full-heap GC: Mark a bit in meta of all live objects

Meta Opt: Place object meta-data in DRAM

KG-W drawbacks

Monitoring overhead

Limited opportunity to predict writes

Fixed DRAM consumption

Write-Rationing Garbage Collection

Limit PCM writes by discovering highly written objects

Kingsguard → monitoring

Crystal Gazer → prediction

Allocation site as a write predictor

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

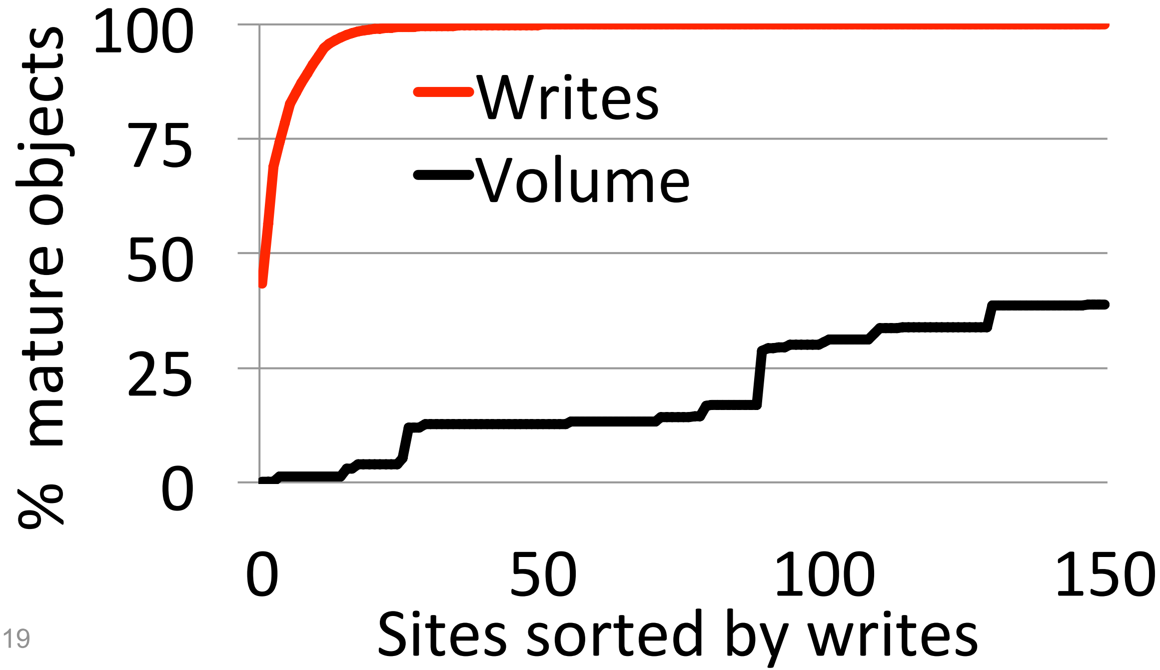
→ Produces highly written objects

Uniform distribution 🙁

Skewed distribution 🙂

Write distribution by allocation site

Few sites capture majority of writes



Crystal Gazer operation

Application Profiling → Advice Generation → Bytecode Compilation

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

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

Advice generation

Generate <alloc-site, advice> pairs

- advice → DRAM or PCM
- input is a write-intensity trace

Two heuristics to classify allocation sites as DRAM

DRAM allocation sites

Frequency: More than a threshold writes

- ✔Aggressively limits writes
- ✗ 1 Byte and 1024 Byte object treated similarly

Density: More than a threshold *write-density*

- ✔Optimizes for writes and DRAM capacity

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

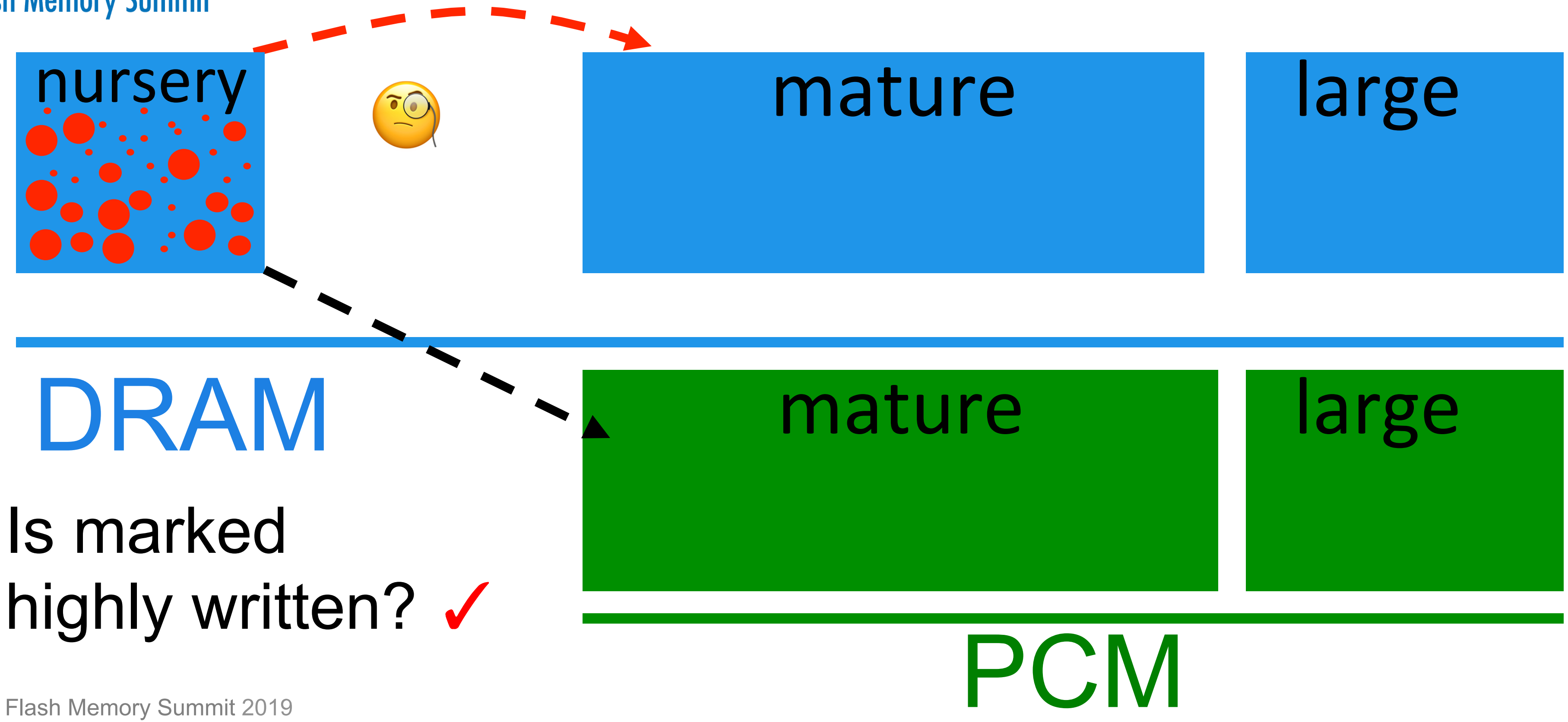
Object Identifier	# Writes	# Bytes	Allocation site
O1	0	4	A() + 10
O2	0	4	A() + 10
O3	128	4	A() + 10
O4	128	4096	B() + 4

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? ✓

Persistence

Persistent parent → copy child objects to PCM

VM startup → Move highly-written to DRAM

Write barrier tracks writes & persistent candidates

Evaluation methodology

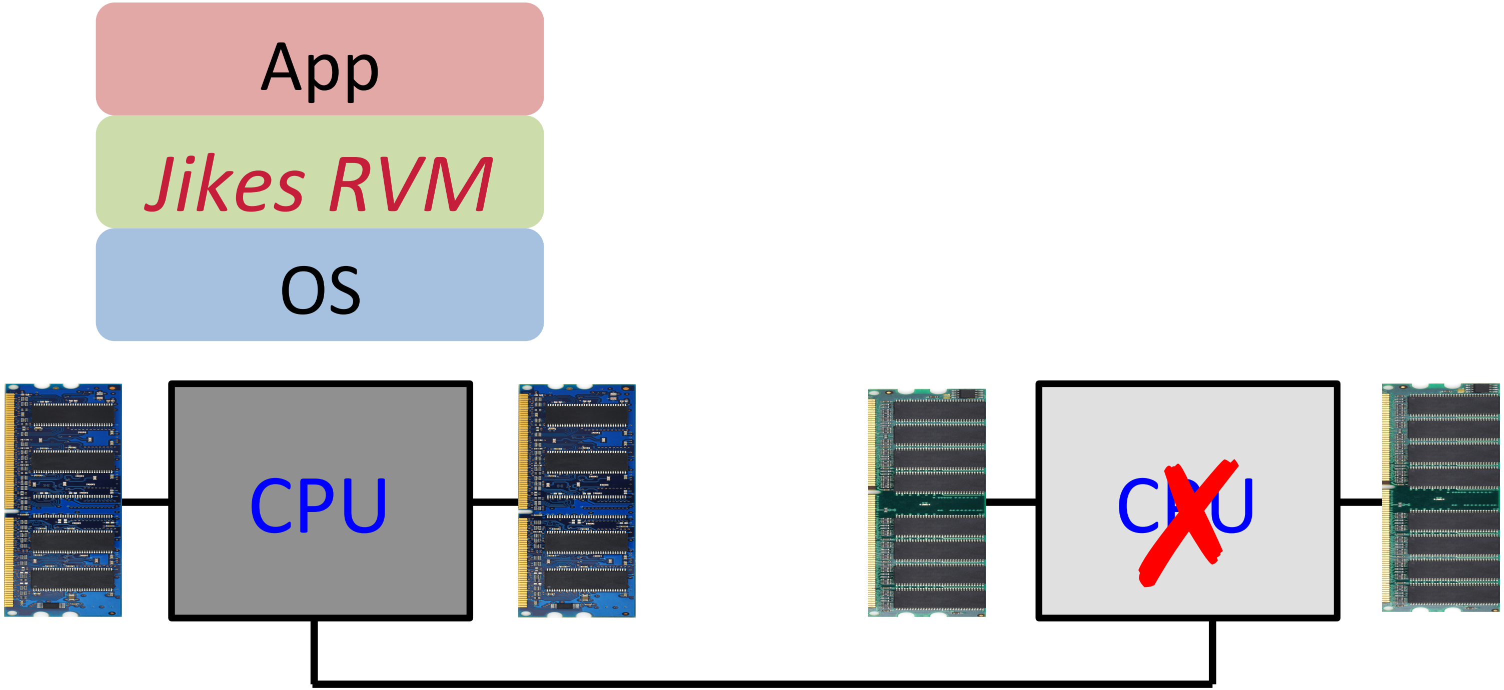
15 Applications → DaCapo, GraphChi, SpecJBB

Medium-end server platform

Different inputs for production and advice

Jikes RVM

Emulation platform



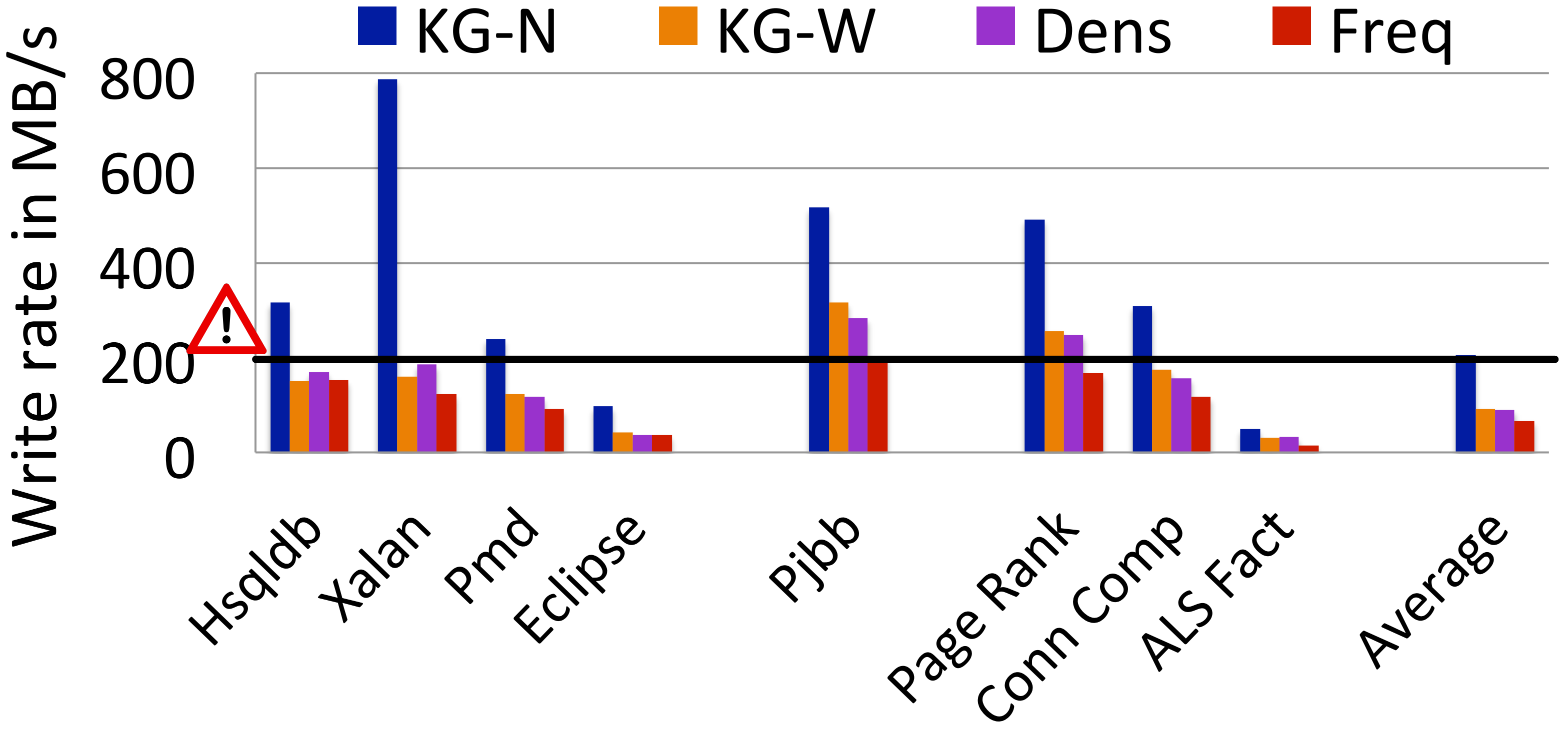
PCM write rates → lifetime

PCM-Only write rate is up to 1.8 GB/s

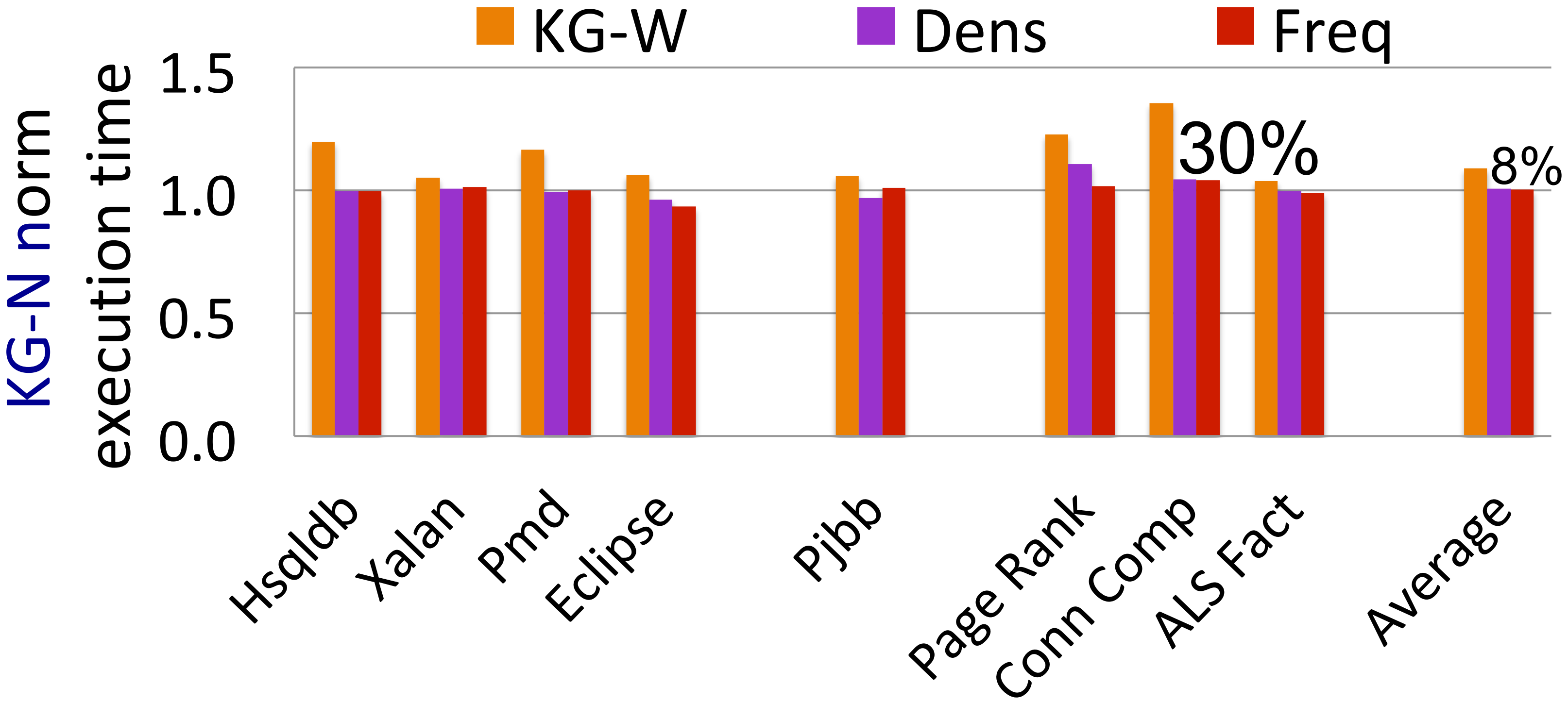
Safe operation is 200 MB/s for 5-10 year lifetime

Flash Memory Summit

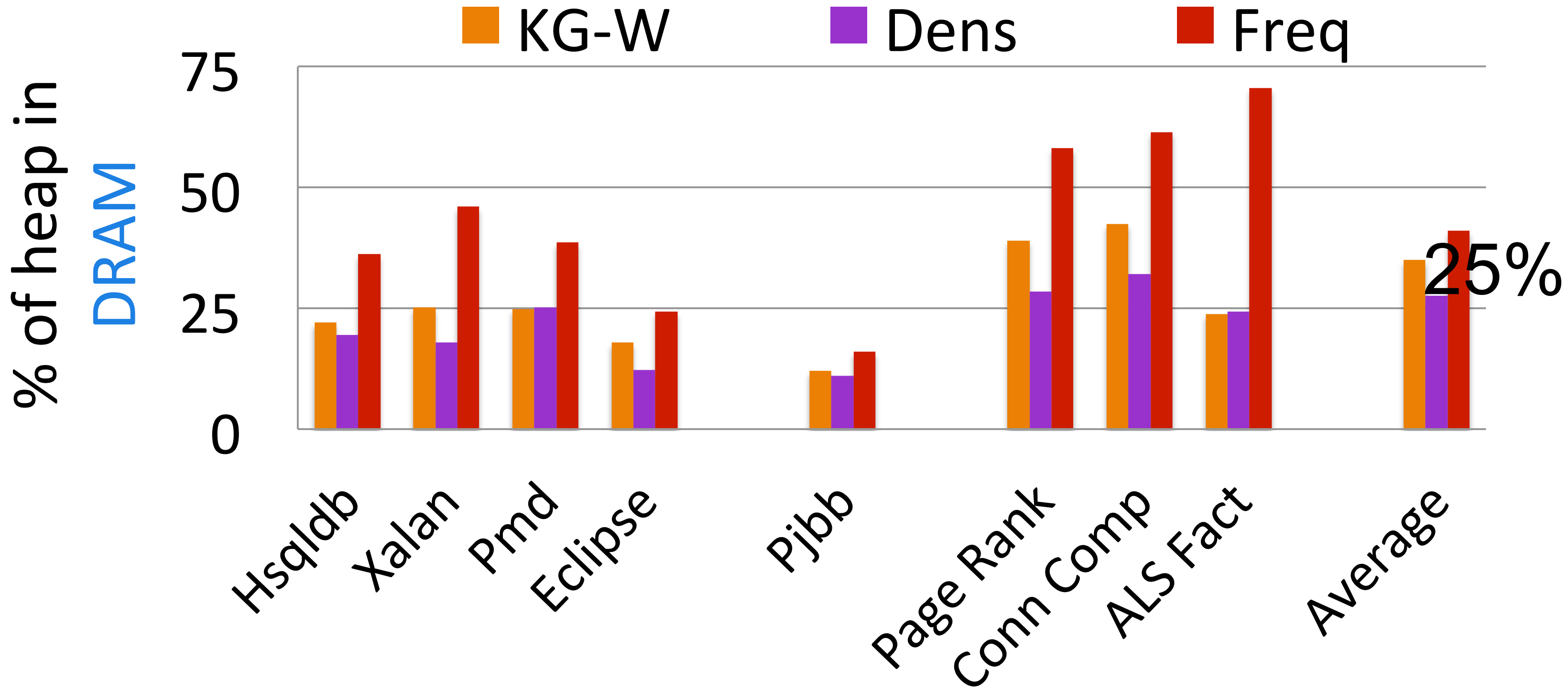
PCM write rates



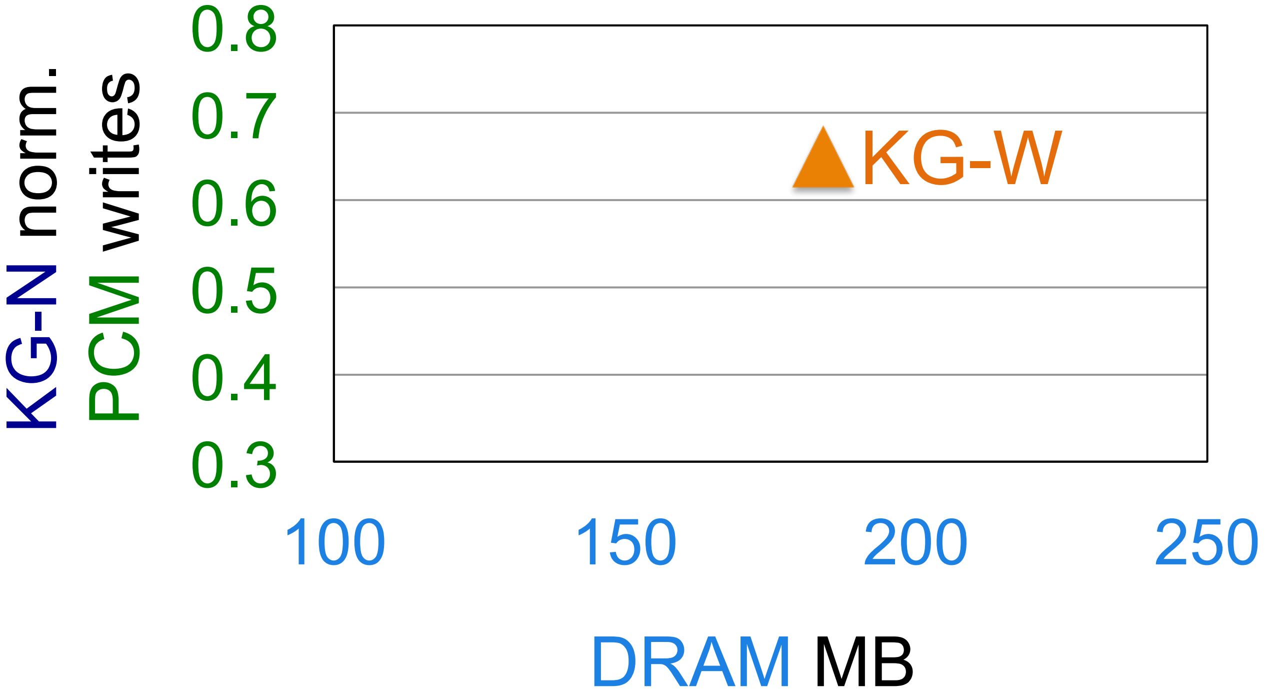
Performance



DRAM capacity

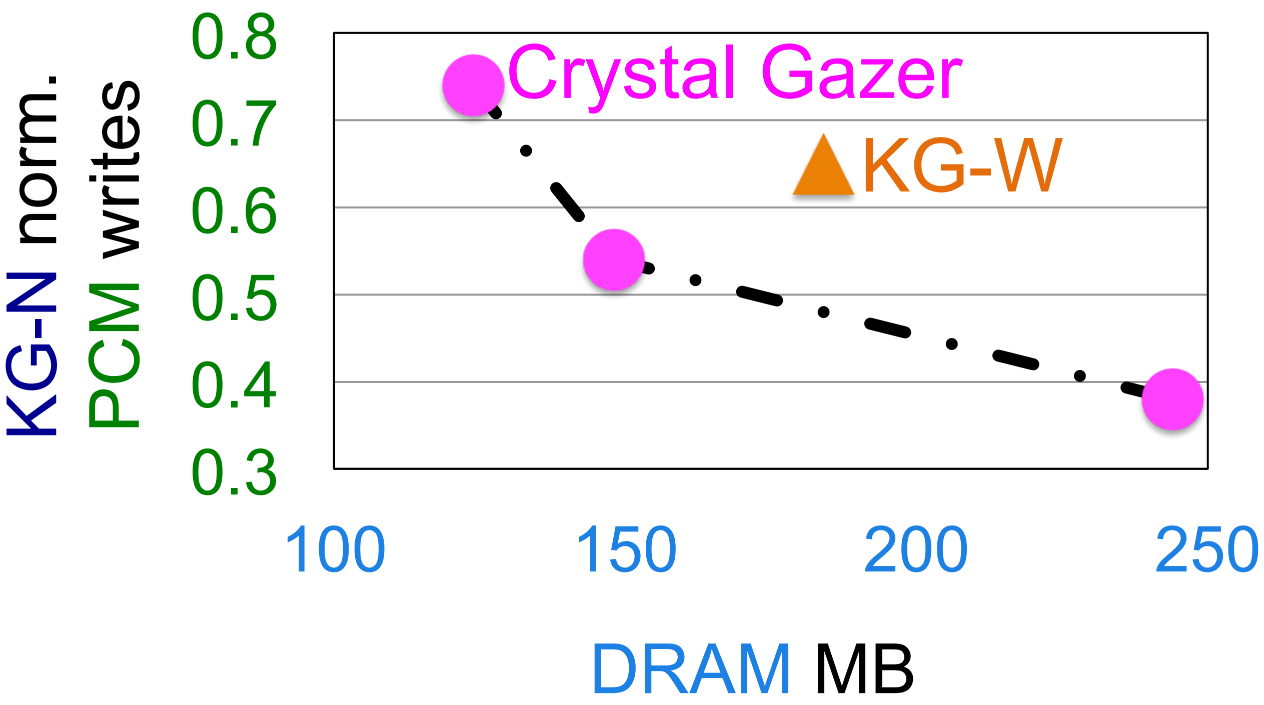


KG-W versus Crystal Gazer



KG-W versus Crystal Gazer

Crystal Gazer opens up Pareto-optimal trade-offs



Write-rationing garbage collection

Hybrid memory is inevitable

DRAM PCM

Each layer can play a role in wider adoption

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

More information

PLDI 2018 → Write-rationing garbage collection for hybrid memories

SIGMETRICS 2019 → Crystal Gazer: Profile-driven write-rationing garbage collection for hybrid memories

ISPASS 2019 → Emulating and evaluating hybrid memory for managed languages on NUMA platform