Emulating and Evaluating Hybrid Memory for Managed Languages on NUMA Hardware

Shoaib Akram (Ghent), Jennifer B. Sartor (Ghent and VUB), Kathryn S. Mckinley (Google), and Lieven Eeckhout (Ghent)

Shoaib.Akram@UGent.be

GHENT UNIVERSITY

VUB VRIJE UNIVERSITEIT BRUSSEL

Google

Hybrid DRAM-PCM memory

☺ More GB/$ with Phase Change Memory

☹ Higher latency *and* low endurance

Speed
Endurance

Capacity

DRAM

PCM

Managing DRAM-PCM memory

Mitigate PCM wear-out

Bridge the DRAM-PCM latency gap

Speed
Endurance

DRAM

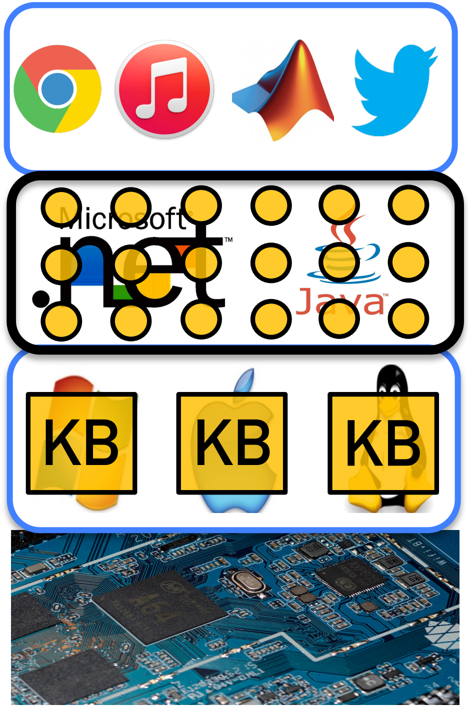
Capacity

PCM

Managing DRAM-PCM memory

PLDI Philadelphia 2018

Write-Rationing Garbage Collection for Hybrid Memories



Garbage collection

Proactive ☺

Fine-grained

objects

Operating System

Coarse-grained

pages KB

GC manages DRAM-PCM hybrid better than OS

Pros/cons of simulating DRAM-PCM

Gain insight

- What triggered the writeback to memory?

Study parameter sensitivity

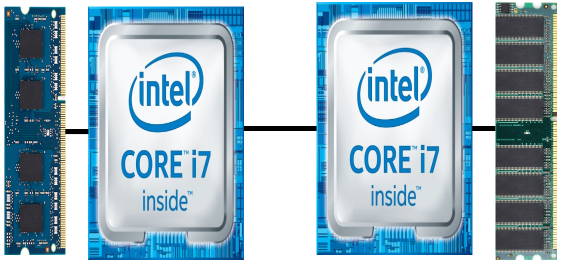
Slow process

- Page Rank over twitter → hours versus months!

Incomplete model

- Missing OS or proprietary hardware features

Emulation for hybrid memory



Multi-socket NUMA for emulating DRAM-PCM hybrid memory

Fast evaluation of emerging workloads

- Several co-running BIG graph analytic applications written in Java

Existing emulation platforms

Focus is to evaluate explicit memory management in C/C++

Focus is to model the latency of PCM

Contribution: Emulation platform

DRAM-PCM emulation for managed applications

Comparison with Sniper using write-rationing garbage collectors

Contribution: Analysis of PCM writes

PCM writes and write rates

- C++ versus Java
- Impact of multiprogramming
- Classic versus emerging applications

Is PCM practical as main memory?

Outline

Heap management

Kingsguard collectors

Comparison with simulation

Write analysis

Outline

Heap management

Kingsguard collectors

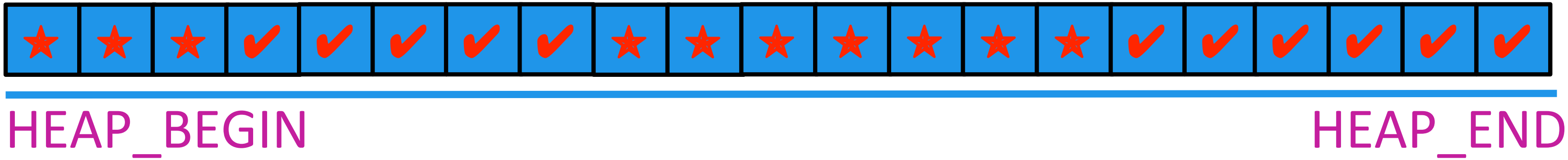
Comparison with simulation

Write analysis

DRAM heap management

Heap Tracker

✔ available ★ occupied

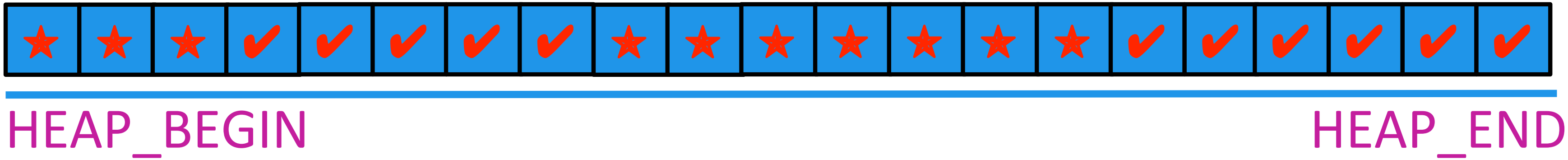


Heap Organization

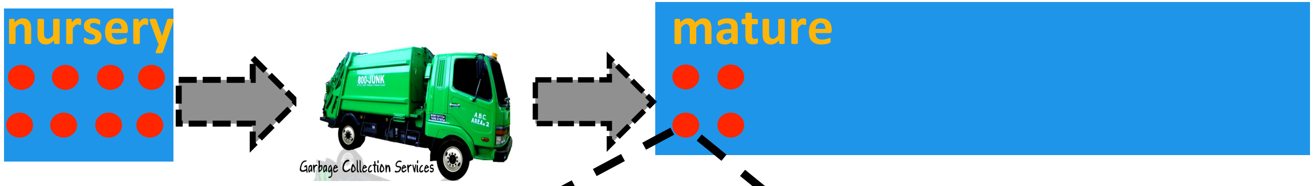
DRAM heap management

Heap Tracker

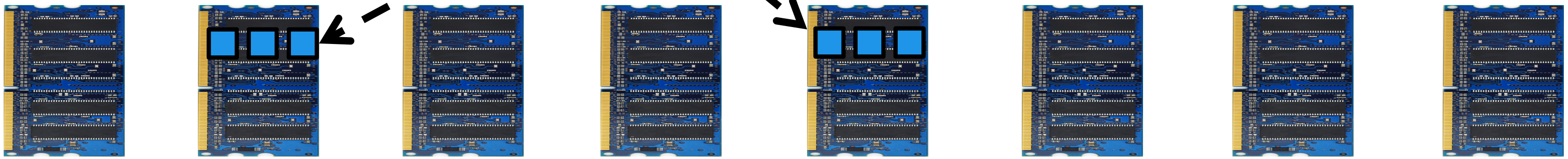
✔ available ★ occupied



Heap Organization

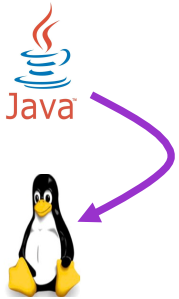


Physical Memory



DRAM-PCM heap management

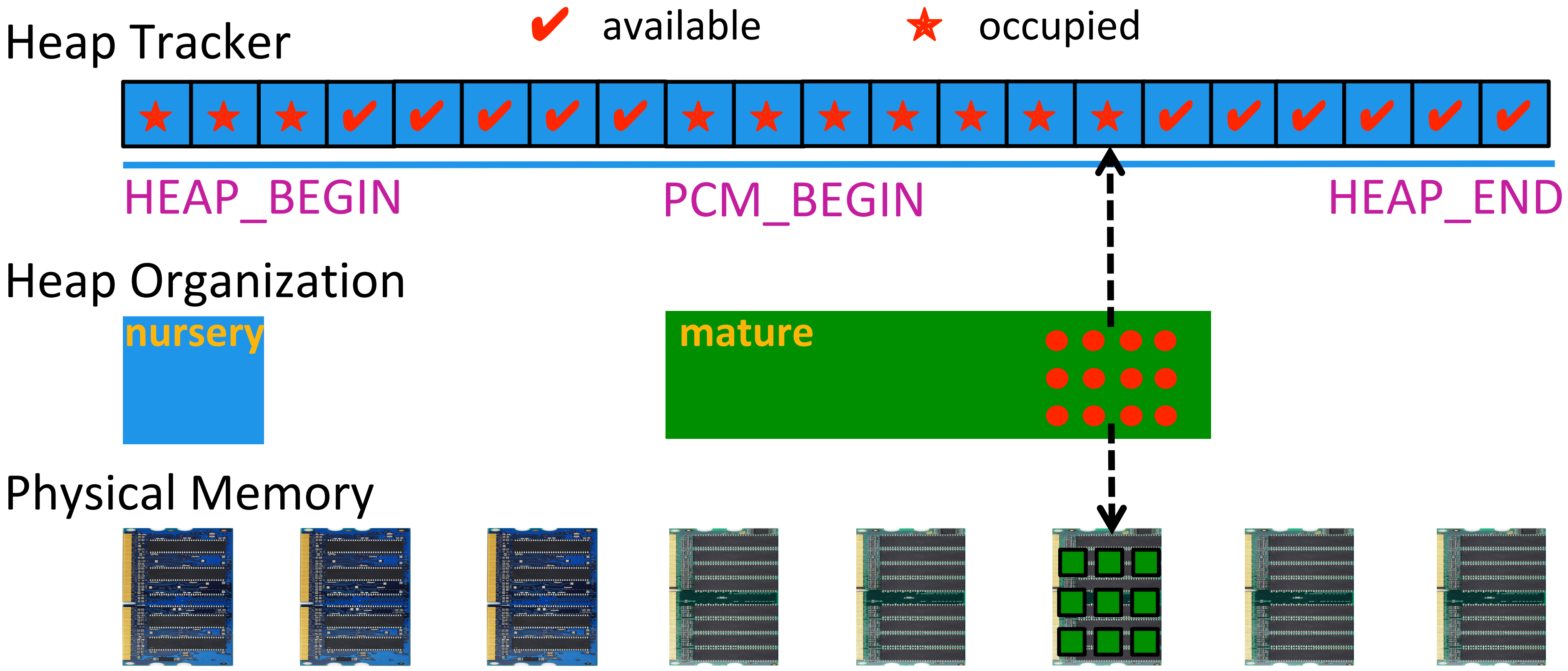
JVM uses mbind() to inform the OS to map a space in DRAM or PCM



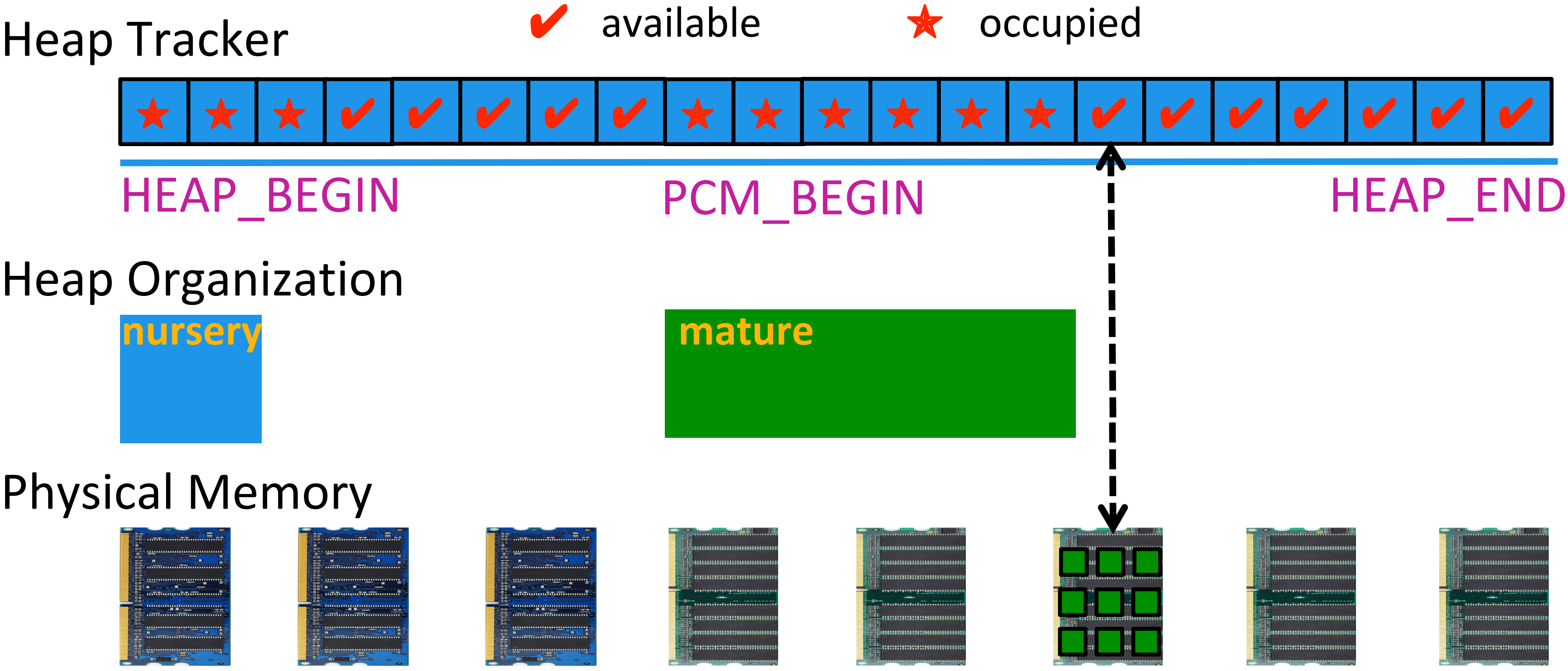
Anything else the JVM should do?

Next: Sanity check with a DRAM nursery and PCM mature

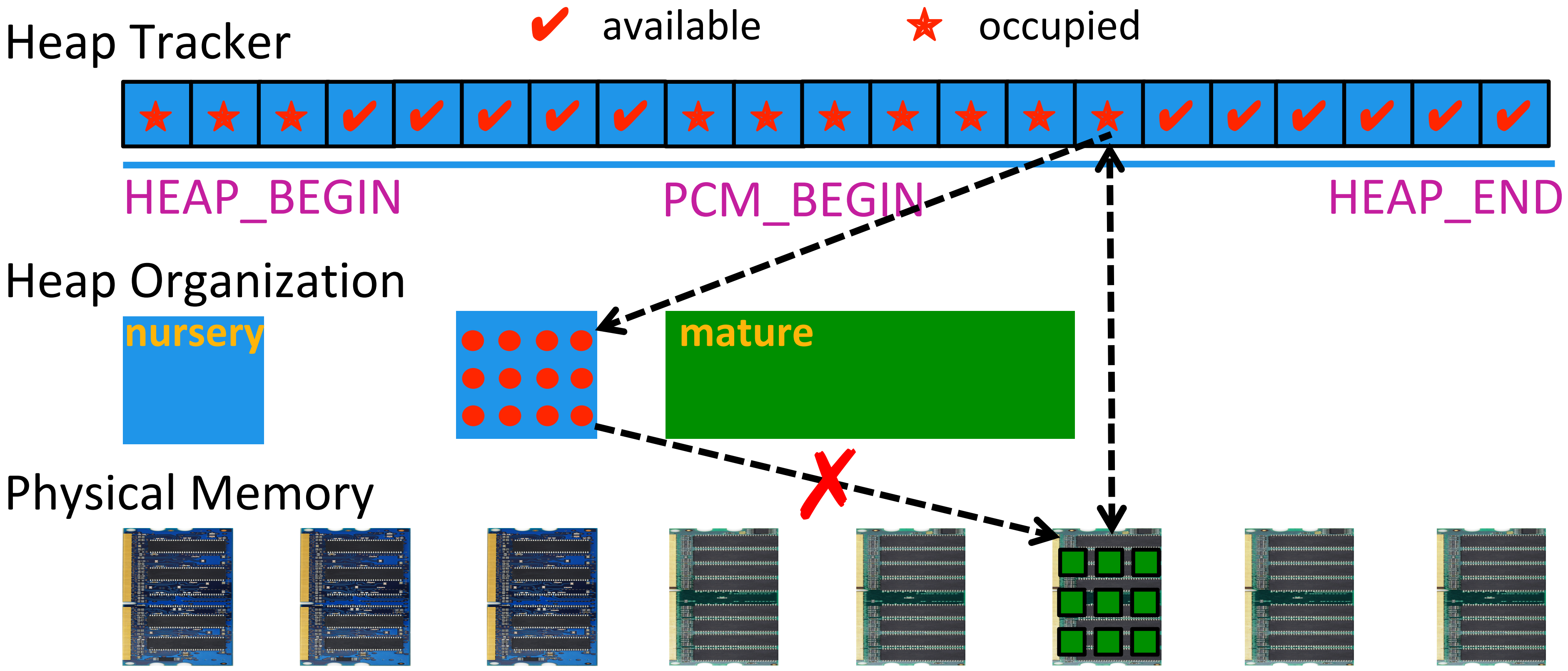
DRAM-PCM heap management



DRAM-PCM heap management



DRAM-PCM heap management

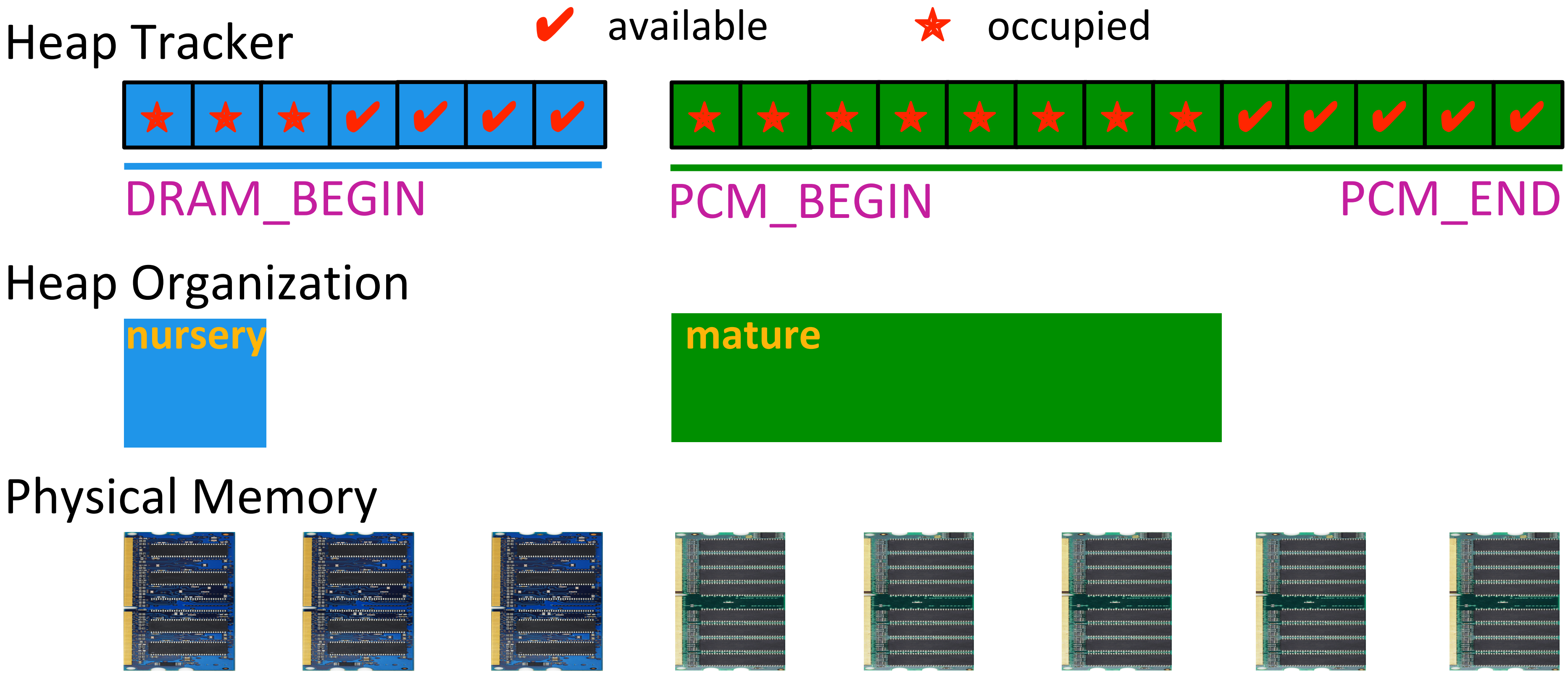


DRAM-PCM heap management

Options

- Map/unmap pages in physical memory whenever space grows/shrinks
- Two free lists ✔

DRAM-PCM heap management



Outline

Heap management

Kingsguard collectors

Comparison with simulation

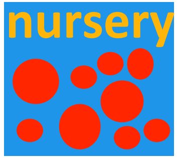
Write analysis

Kingsguard-Nursery (KG-N)

Write-rationing GC: concentrate writes in DRAM

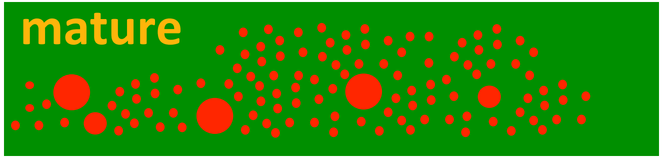
70%

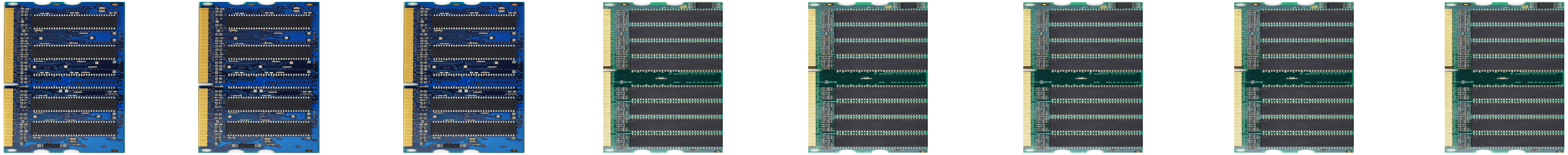
of writes



22%

to 2% of objects

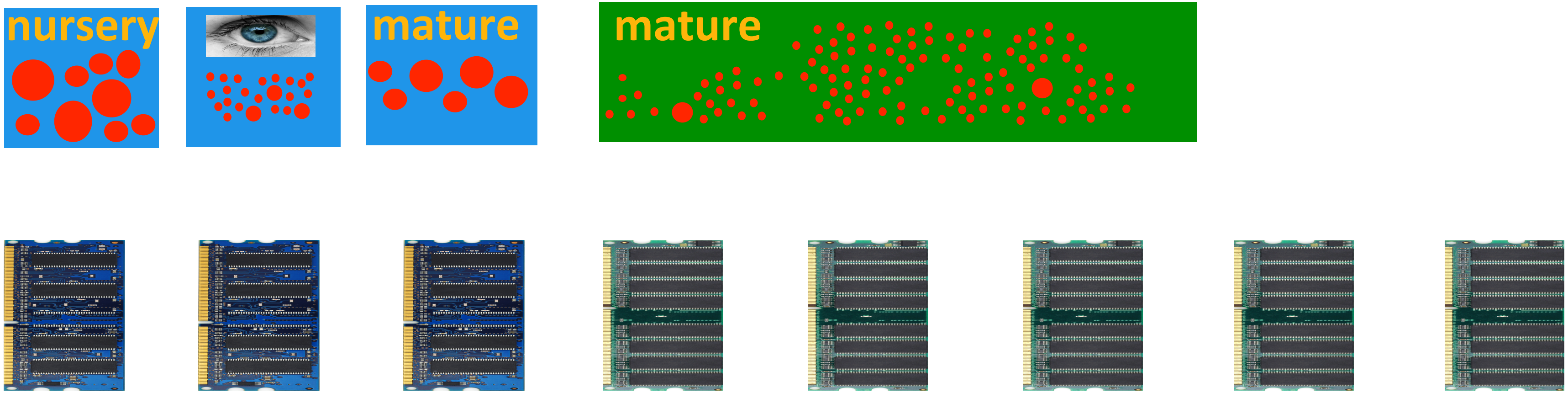




Kingsguard-Writers (KG-W)

KG-W monitors writes in a DRAM observer space

Trades off performance for better endurance



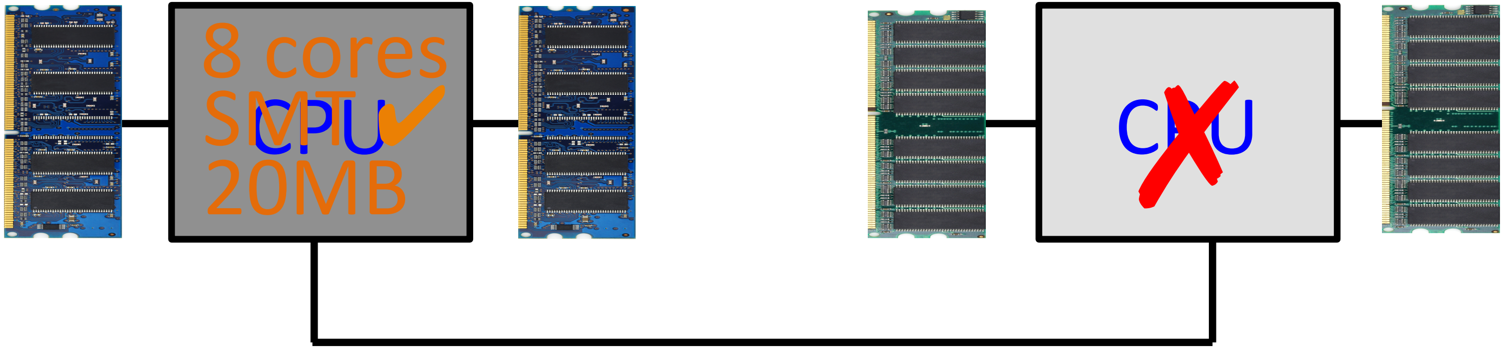
Emulation setup

App/Monitor

Jikes RVM

OS

Monitor: Intel pcm-memory.x
to get per-socket write rate



Emulation versus simulation

PCM write reduction with KG-N and KG-W versus PCM-Only

Execution time increase with KG-W versus KG-N

No OS in simulation
Faithfully model emulator

Reduction in PCM writes with KG-N and KG-W versus PCM-Only

Kingsguard collectors limit PCM writes

KG-W much better than KG-N

	Simulation	Emulation
KG-N	4%	8%
KG-W	62%	64%

Increase in execution time with KG-W versus KG-N

KG-W is slower than KG-N because it monitors writes to objects

	Simulation	Emulation
KG-W	+7%	+10%

Graph workload evaluation

GraphChi: Analyze BIG graphs on a single machine

- Both Java and C++ implementations

Page Rank *and* Connected Components

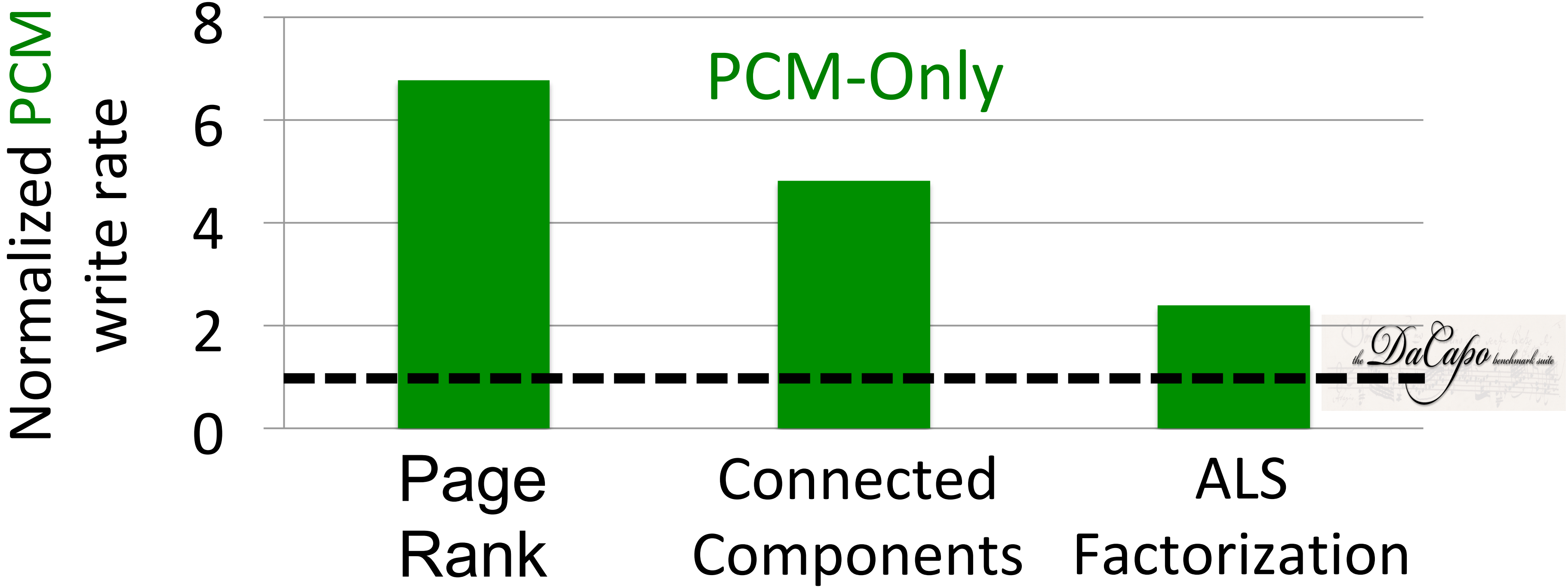
- LiveJournal social network

ALS Factorization

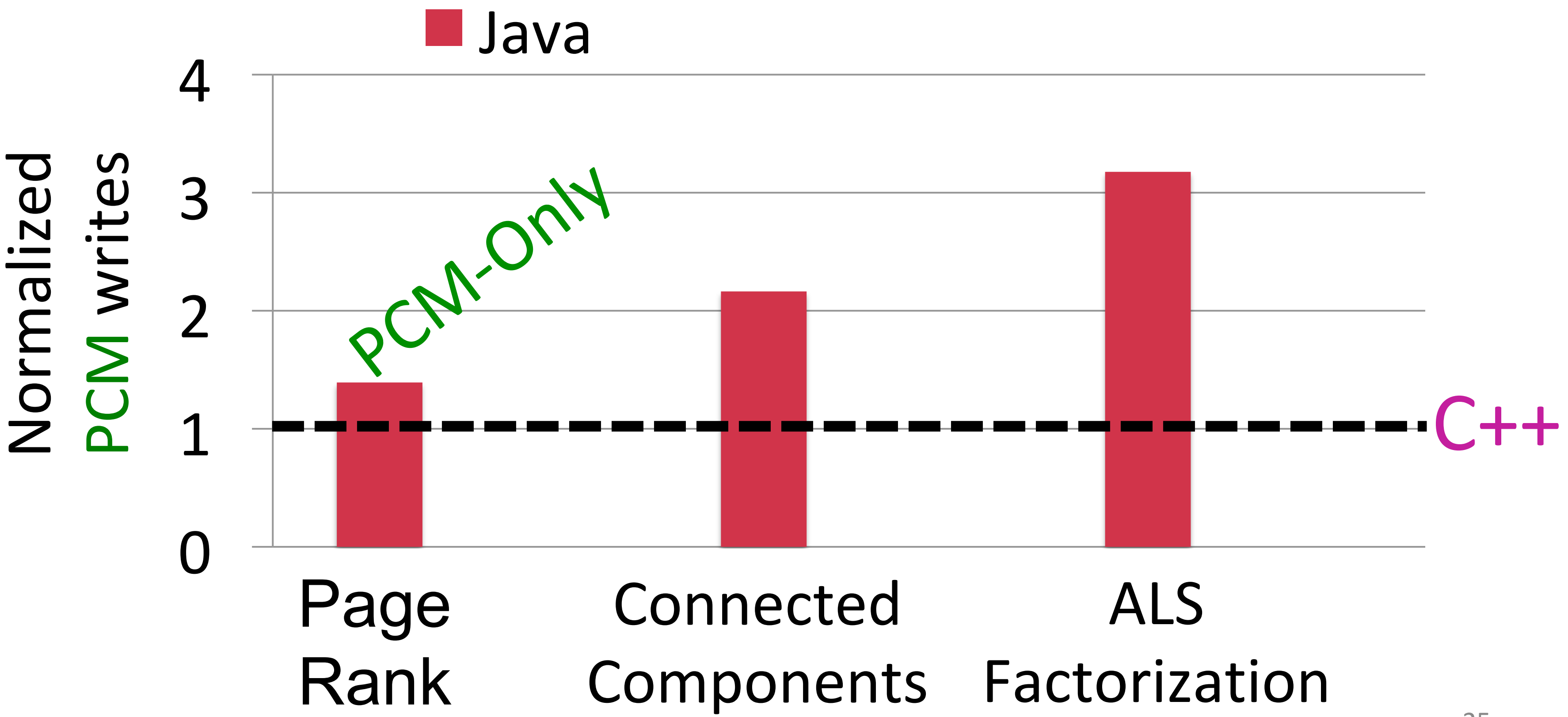
- Netflix challenge

Graph apps write more than DaCapo

Billions of vertices → Billions of objects



Java writes more to PCM than C++

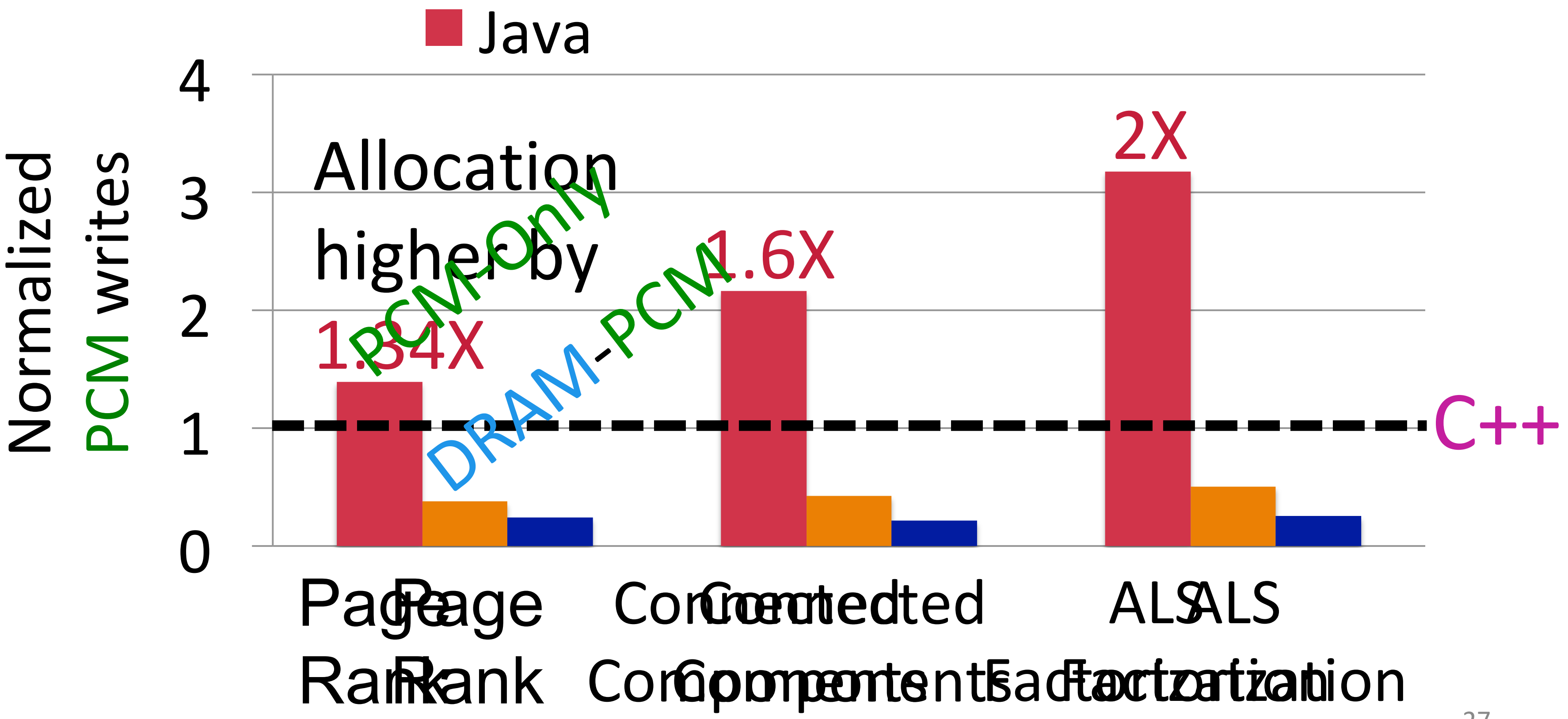


Java writes more to PCM than C++

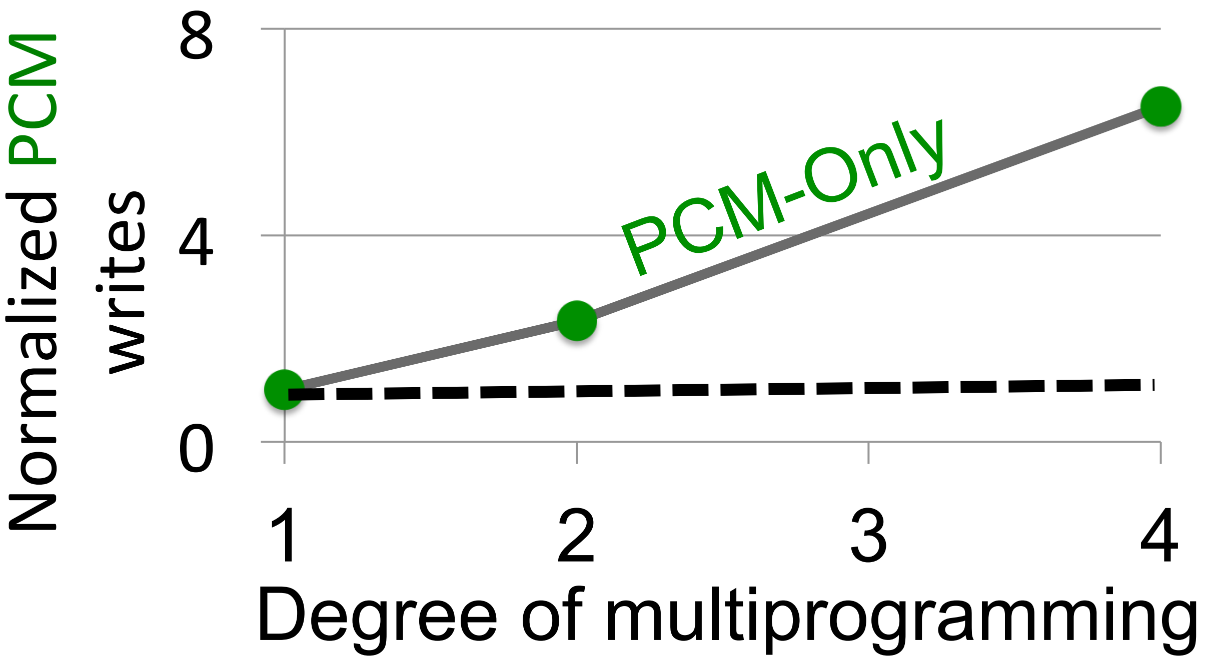
Reasons

- Higher allocation volume →
- Copying between heap spaces
- Zeroing to provide memory safety

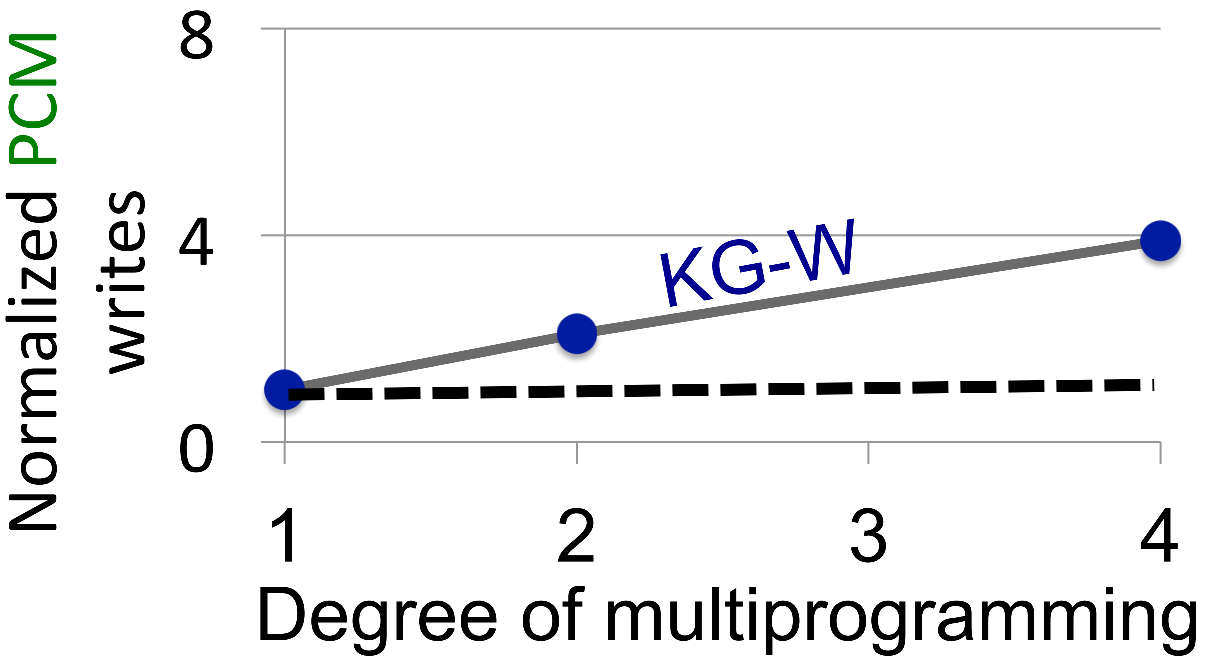
Java writes more to PCM than C++



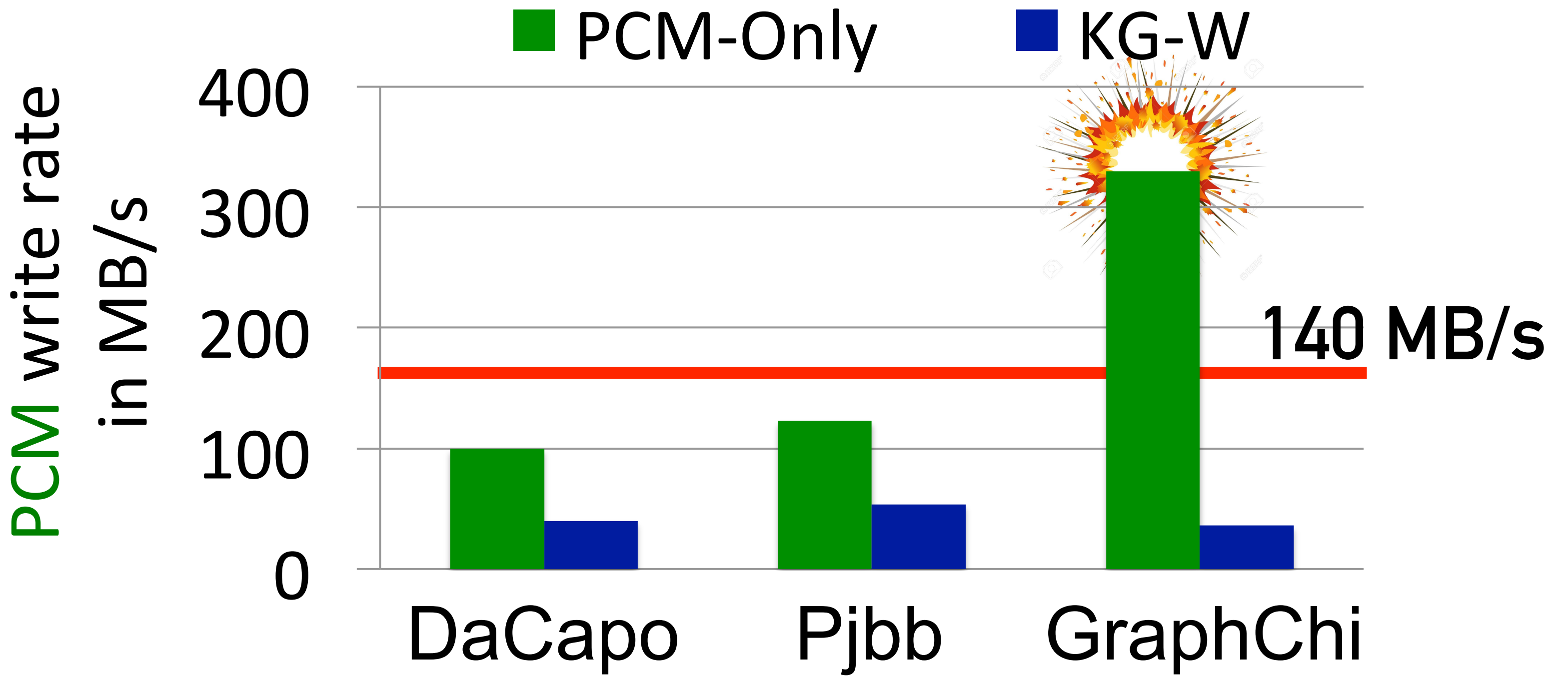
Writes increase **super-linearly** due to multiprogramming with PCM-Only



Writes increase linearly due to multiprogramming with KG-W

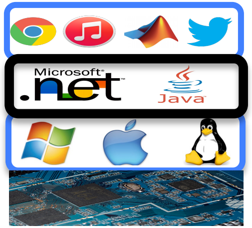


PCM-Only is not practical as main memory



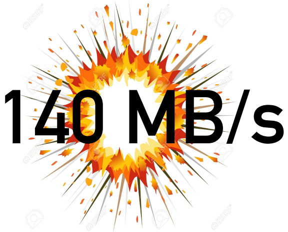
Conclusions

Across the stack emulation of hybrid memory



Similar outcomes with different evaluation methods

More research to make PCM practical as main memory



Model	Capacity	DWPD	TBW	GB/Day	Warranty
SSD 600p	1,000GB	0.32	576.0	316	5 Years
SSD 750	1,200GB	0.06	127.0	70	5 Years
DC P3320	1,200GB	0.68	1,480.0	811	5 Years
DC P3520	1,200GB	0.68	1,480.0	811	5 Years
DC D3600	1,000GB	3	5,475.0	3,000	5 Years
DC D3700	1,600GB	10	29,200.0	16,000	5 Years
DC P3500	1,200GB	0.3	657.0	360	5 Years
DC P3600	1,200GB	3	6,570.0	3,600	5 Years
DC P3700	1,600GB	17	49,640.0	27,200	5 Years
DC S3100	1,000GB	0.1	109.5	100	3 Years
Optane™ SSD DC P4800X	375GB	30	12,318.8	11,250	3 Years
Optane™ Memory	16GB	6.25	182.5	100	5 Years
Optane™ Memory	32GB	3.125	182.5	100	5 Years

Source: Objective Analysis, April 2017