Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Using Heterogeneous Memory Systems

Antonio J. Peña

Leading Researcher & Group Manager

AccelCom
Accelerators & Communications for HPC

Feb. 25, 2026

Barcelona

Agenda

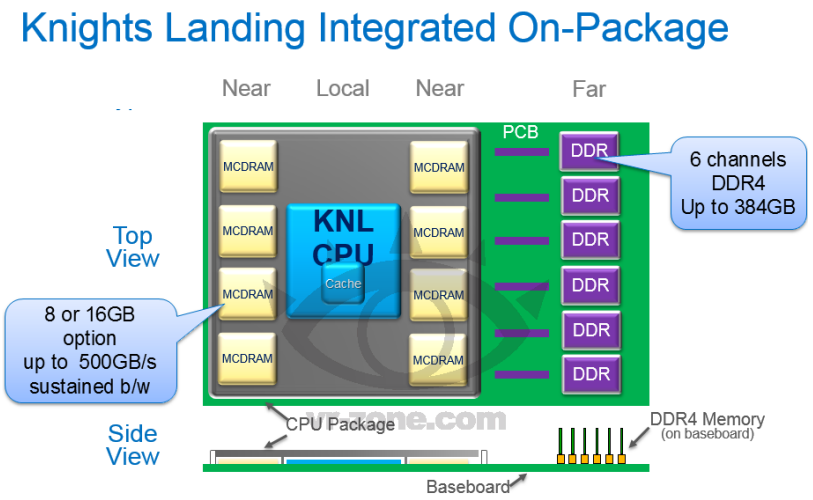
- Heterogeneous Memory Systems
  - Examples
  - Production Programming
  - Research Approaches
- Summary

Heterogeneous Memory Systems

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

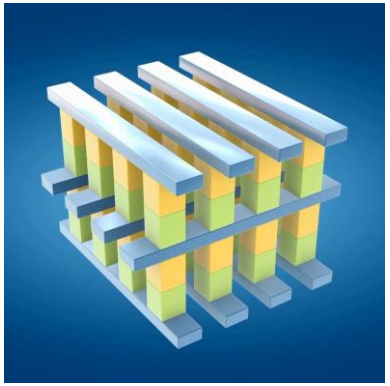
Motivation

- Heterogeneity in computing explored:
  - Heterogeneous processing ✓
  - Heterogeneous memory …
- Different memory technologies within computers already a reality
  - Scratchpad
    - Embedded processors
    - GPUs
  - High Bandwidth Memory
    - Intel KNL, Sapphire Rapids
    - GPUs
  - (Byte-addressable) NVRAM
    - HP's "The Machine"
    - Intel 3D Xpoint / Optane
- We expect more memory heterogeneity:
  - **CXL: memory expansion, memory pools**



Integrated on-package MCDRAM brings memory nearer to CPU for higher memory bandwidth and

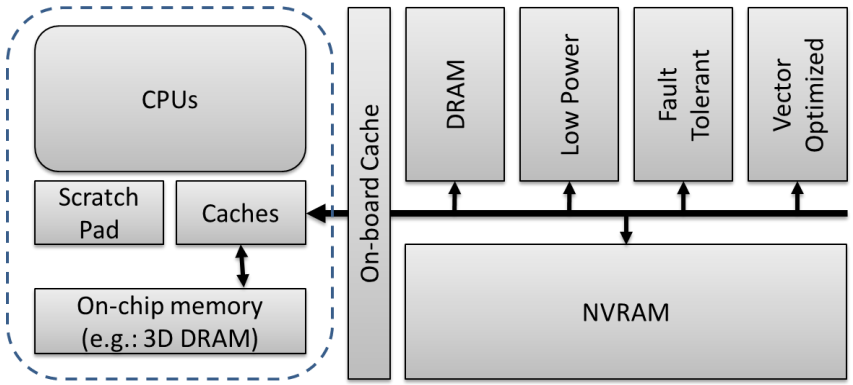
Intel KNL memory architecture



Intel 3D XPoint Technology

Motivation

- Different features:
  - Size, resilience, access patterns, energy, persistency…
- Examples:
  - Scratchpad:
    - Cachelike speeds, small sizes
  - Vector-specialized (e.g.: GDDR)
    - High bandwidth if cont. accesses
  - Low-power memory
    - Increased energy/speed ratio
  - ECC-enabled memory
    - Fault tolerance; speed & size ↑
  - I/O class (e.g.: NVRAM)
    - Large; reduced speeds & energy
    - Faster reading than writing



Heterogeneous Memory Systems

- Deep Memory != Heterogeneous Memory
  - Rationale: “deep” implies hierarchy (i.e., cache-like)
  - (own view – many authors don’t follow this distinction)
  - In some cases, deep memory may work well (high locality)
- Heterogeneous Memory Methodologies
  - Page level
    - Leverages OS’s view
    - Can monitor hot vs. cold pages, # of allocations, total size, global status
    - Easy migrations
  - Object granularity (object: variable, static array, heap buffer, etc.)
    - Leverage object semantics
    - Usually same access pattern across entire object
    - User-friendly – user may hint / control
  - …
  - Combinations?

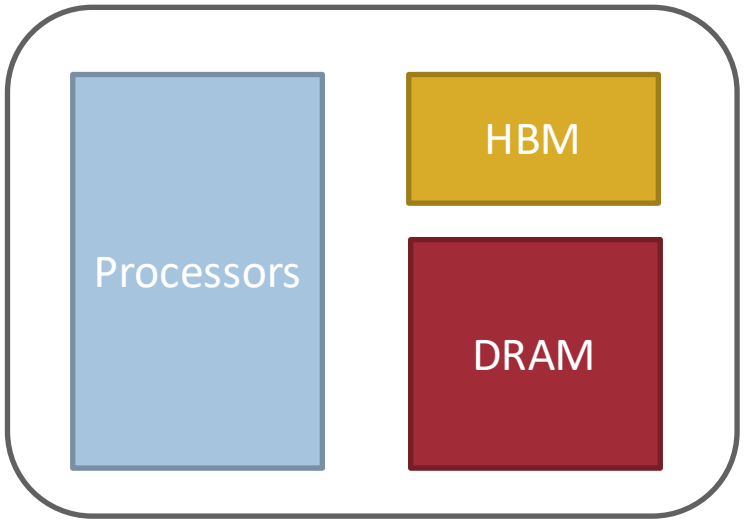
e.g., Sapphire Rapids, Optane

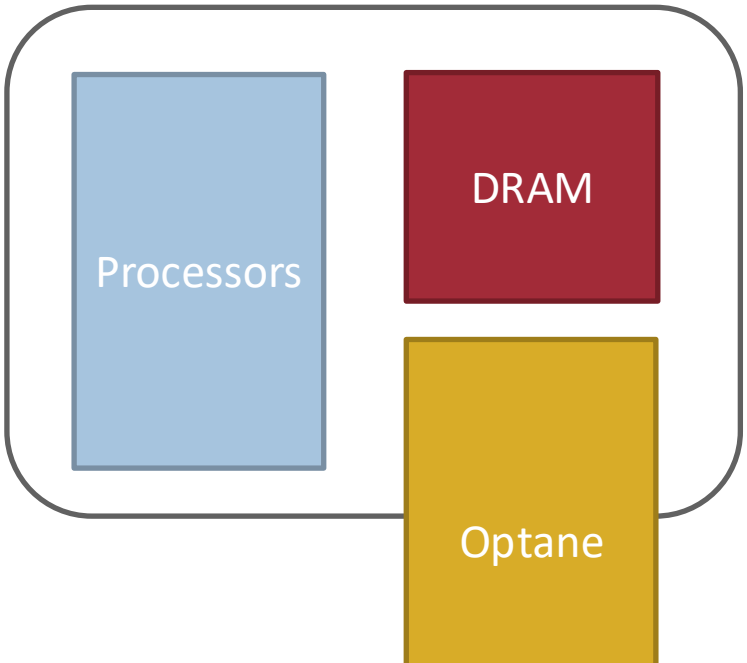
Sapphire Rapids

- Two levels of memory
  - Main memory (DRAM)
    - Regular DRAM latency/bandwidth
  - HBM
    - High bandwidth
    - + latency than DRAM
  - Default allocation: DRAM (*slower*)

Optane

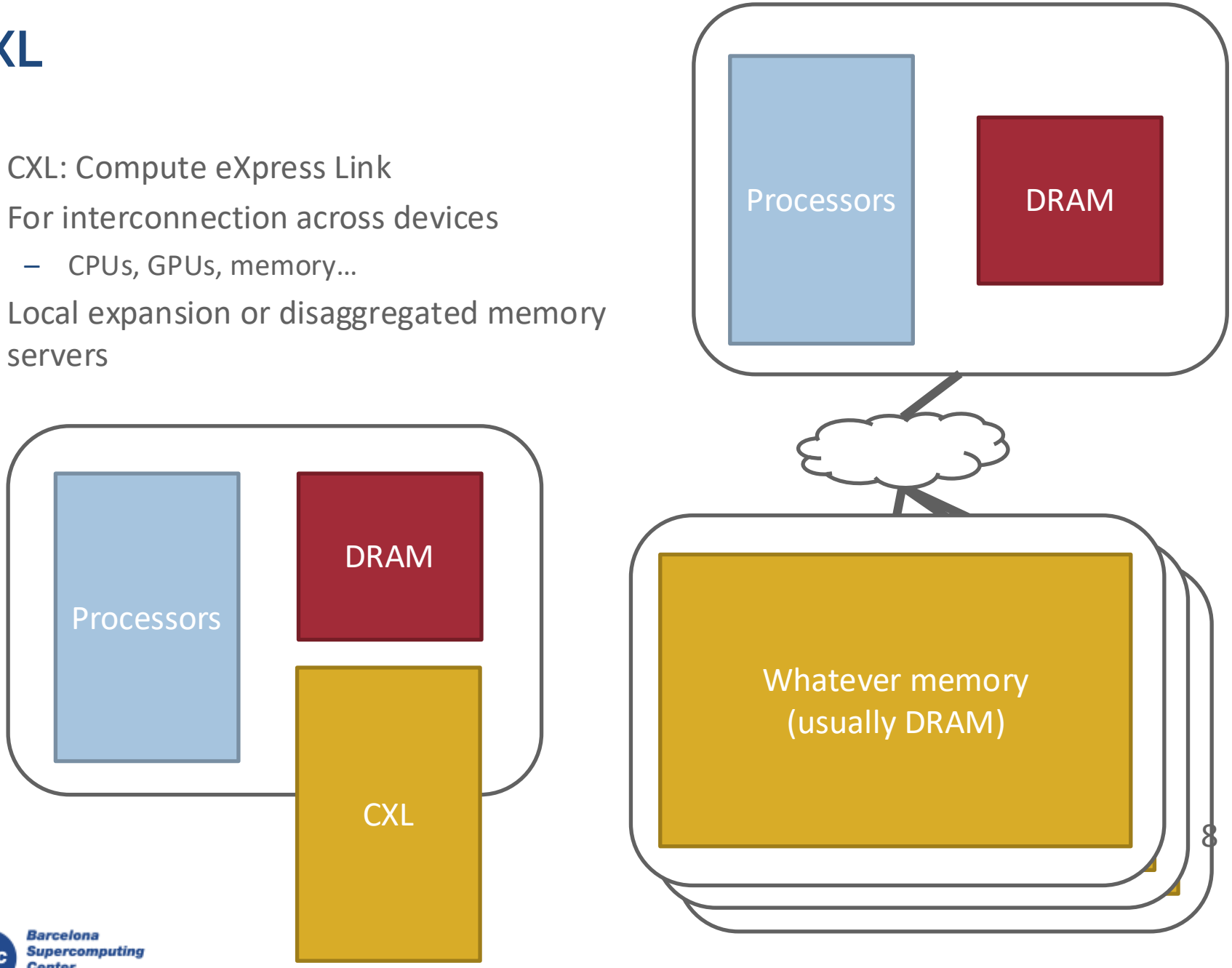
- Two levels of memory
  - Main memory (DRAM)
    - Regular DRAM latency/bandwidth
  - Intel Optane DC Persistent Memory
    - Very high capacity + persistency
    - Higher latency, but better than SSDs
  - Default allocation: DRAM (*faster*)





CXL

- CXL: Compute eXpress Link
- For interconnection across devices
  - CPUs, GPUs, memory…
- Local expansion or disaggregated memory servers



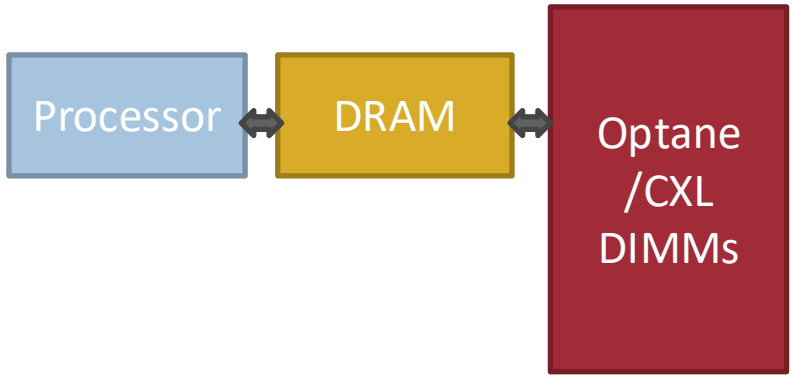
Memory Modes: Deep Memory

Sapphire Rapids

- HBM cache for DRAM
- Only DRAM address space
- Done in hardware (applications don't need to be modified)
- Misses more expensive (HBM and DRAM access)

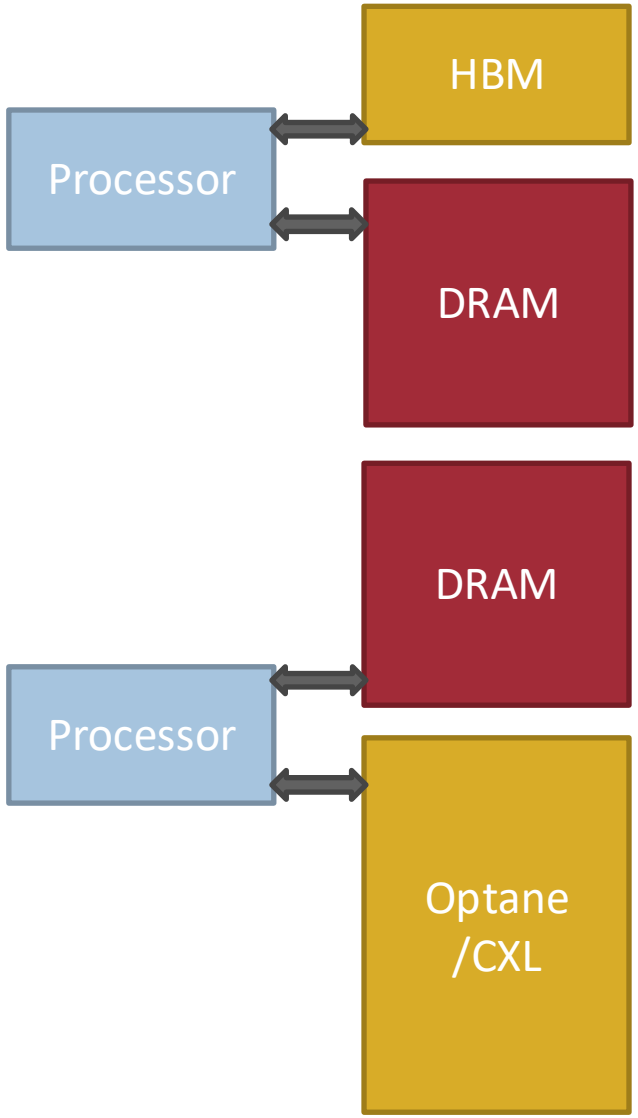
Optane / CXL

- DRAM as cache for other DIMMs
- Only large DIMMs address space
- Done in hardware (applications don't need to be modified)
- Misses more expensive (DRAM and Optane/CXL access)



Memory Modes: Heter. Memory

- a.k.a. Flat Mode / App. Direct
- HBM/Optane/CXL and local DRAM are all available
- More overall memory available
- Software managed:
  - Software needs to handle itself allocation placement
  - Regular loads/stores afterwards



Using Heterogeneous Memory Modes

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

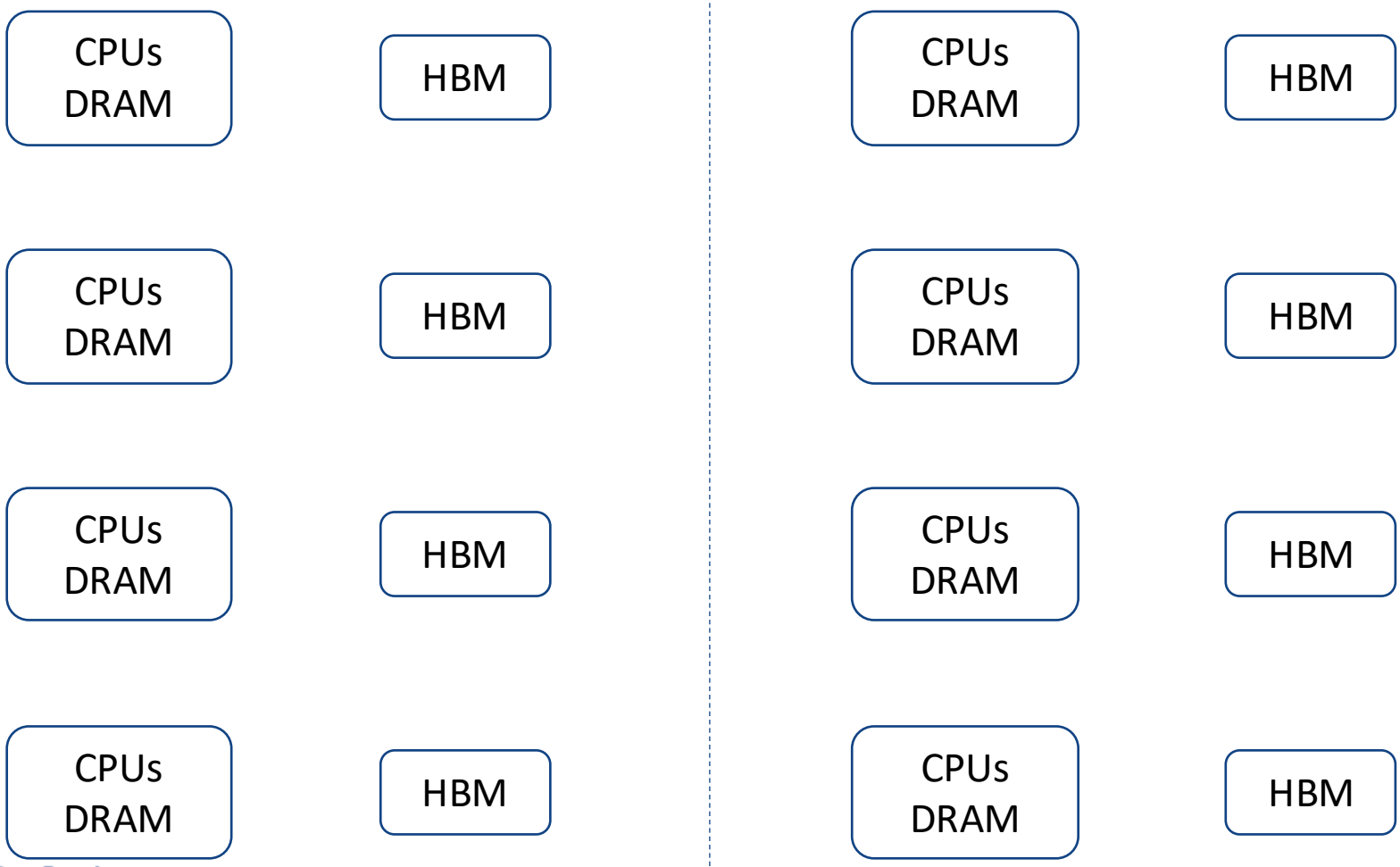
Using Heterogeneous Memory Modes

- Two memory spaces are available
  - By default, local DRAM is used
- Using HBM/Optane/CXL without changing an application
  - Requires kernel support
  - Set bulk memory policy
    - Preferred or enforced memory for application
    - **HBM/Optane/CXL exposed as NUMA nodes**
    - Use *numactl* program
      - numatcl –H [List of available NUMA nodes]
      - numactl --membind 1 my_app [Fails if it exhausts memory]
      - numactl --preferred 1 my_app [Falls back if it exhausts memory]
- **Automatic page movement / tiering** since Linux kernel 5.5
  - Not default: It has to be compiled/enabled by sysadmin
  - Conceptually analogous to disk swapping

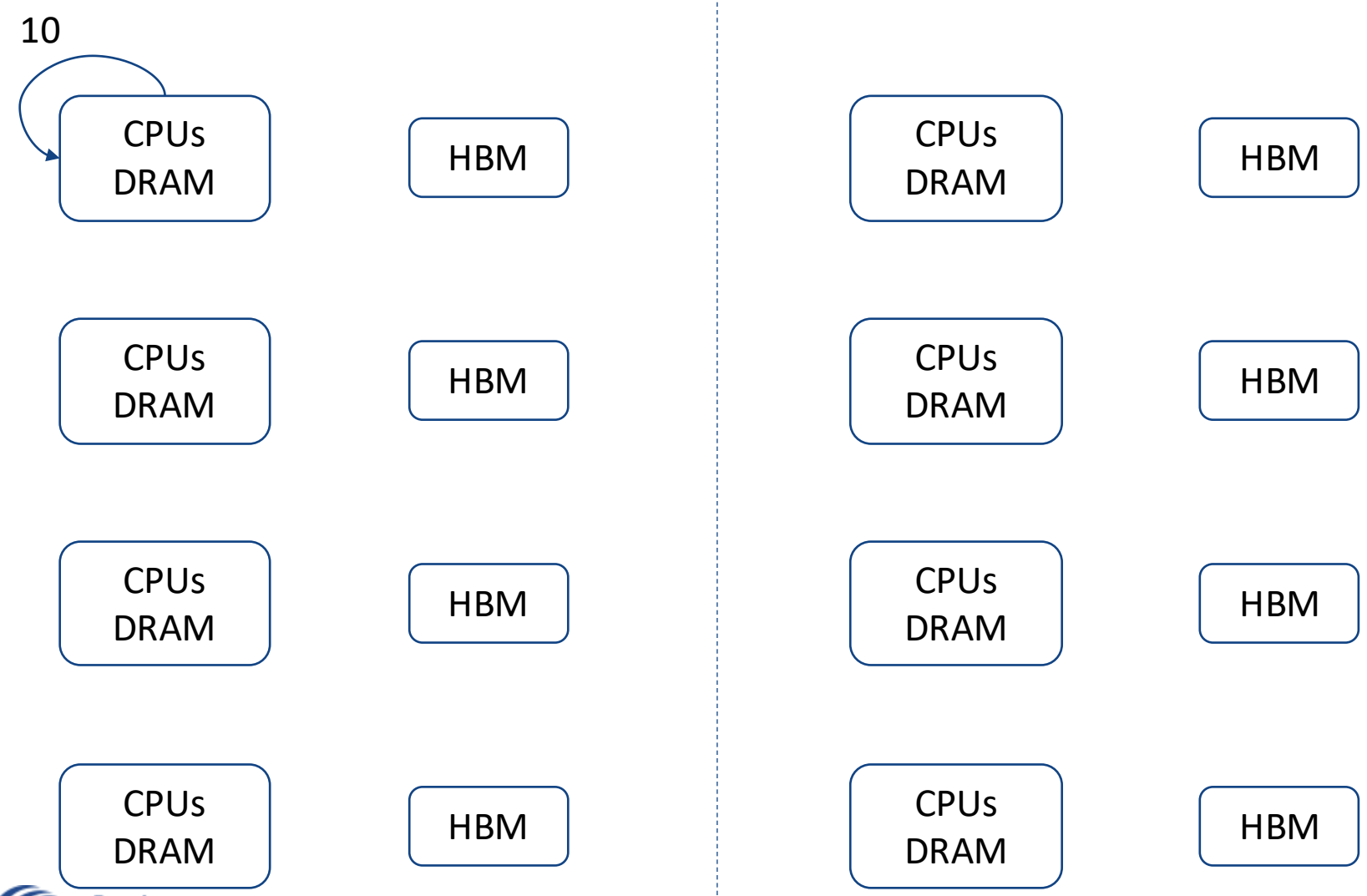
Using Heterogeneous Memory Modes

- Two memory spaces are available
  - By default, local DRAM is used
- Using HBM/Optane/CXL without chan
  - Requires kernel support
  - Set bulk memory policy
    - Preferred or enforced memory for application
    - **HBM/Optane/CXL exposed as NUMA nodes**
    - Use *numactl* program
      - numatcl –H [List of available NUMA nodes]
      - numactl --membind 1 my_app [Fails if it exhausts memory]
      - numactl --preferred 1 my_app [Falls back if it exhausts memory]
- **Automatic page movement / tiering** since Linux kernel 5.5
  - Not default: It has to be compiled/enabled by sysadmin
  - Conceptually analogous to disk swapping

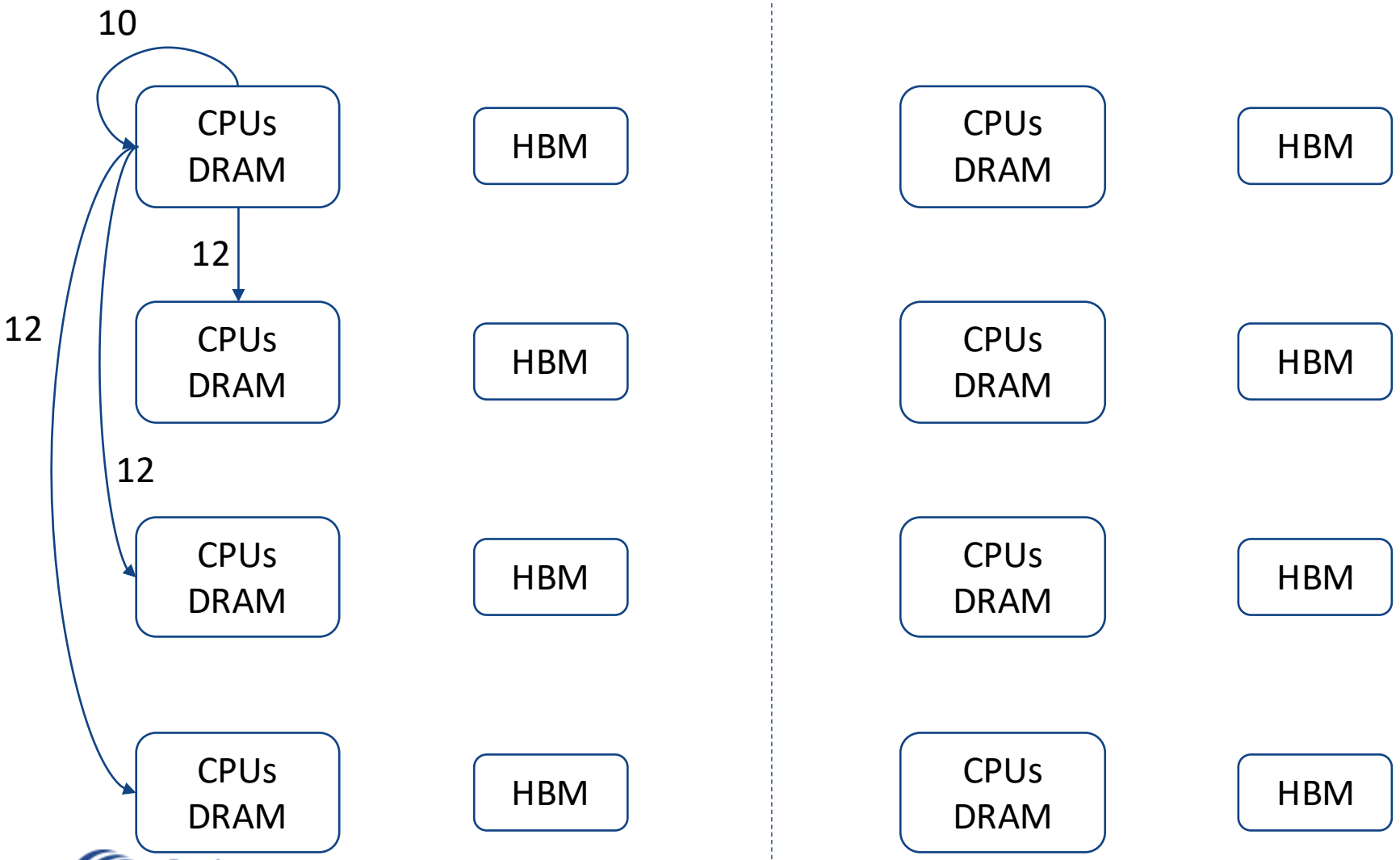
E.g., NUMA Node Dist. in Sapphire Rapids



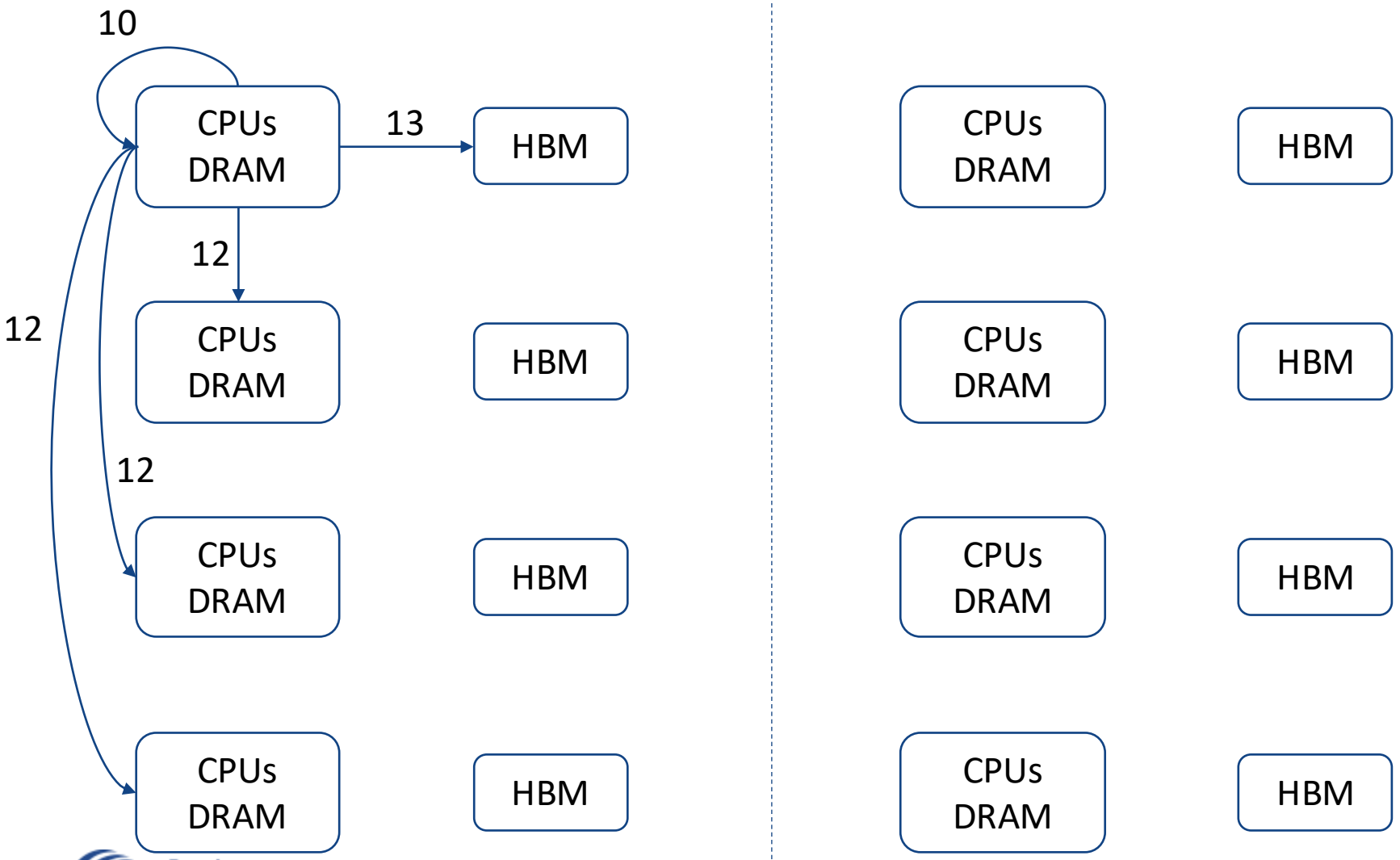
E.g., NUMA Node Dist. in Sapphire Rapids



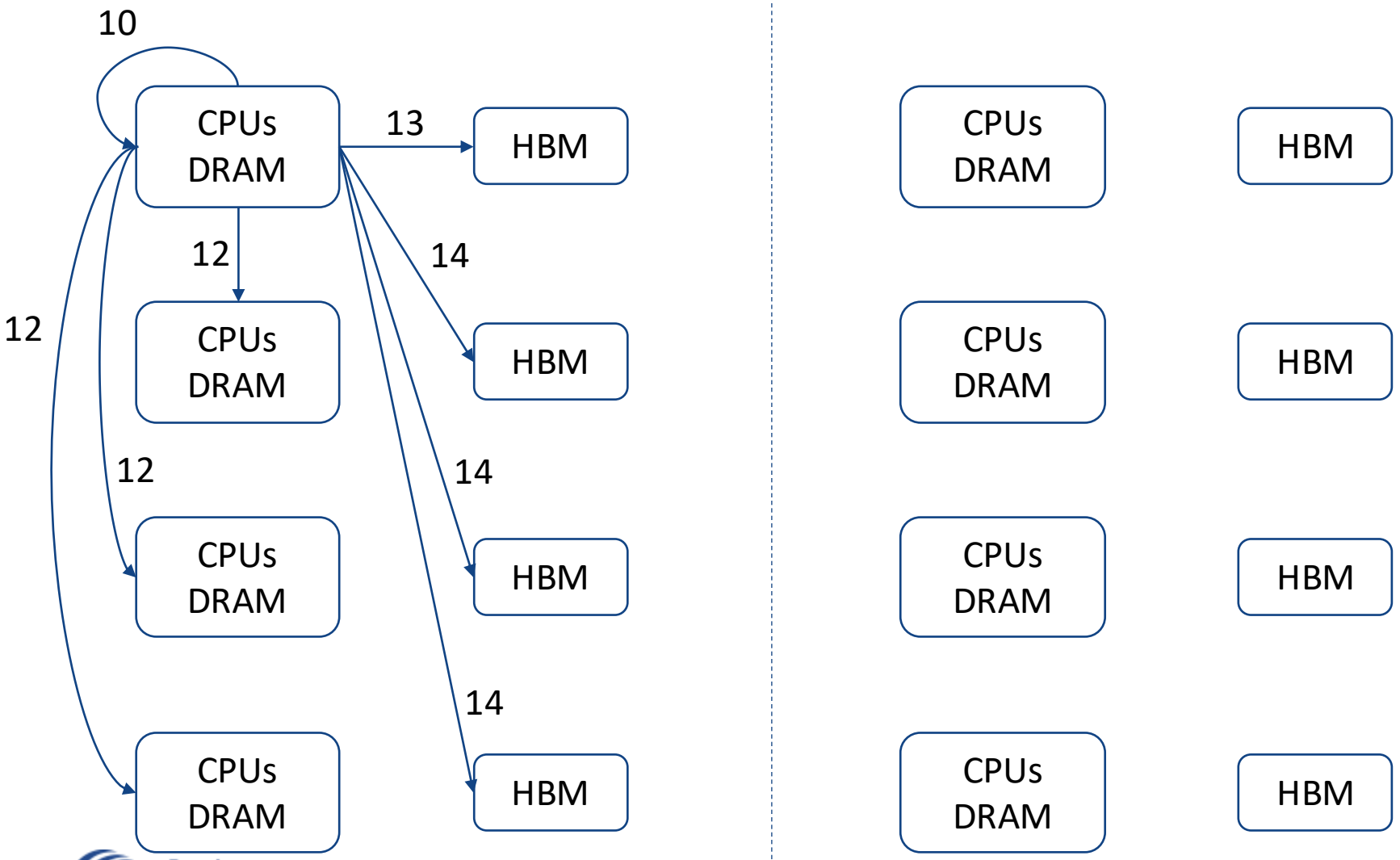
E.g., NUMA Node Dist. in Sapphire Rapids



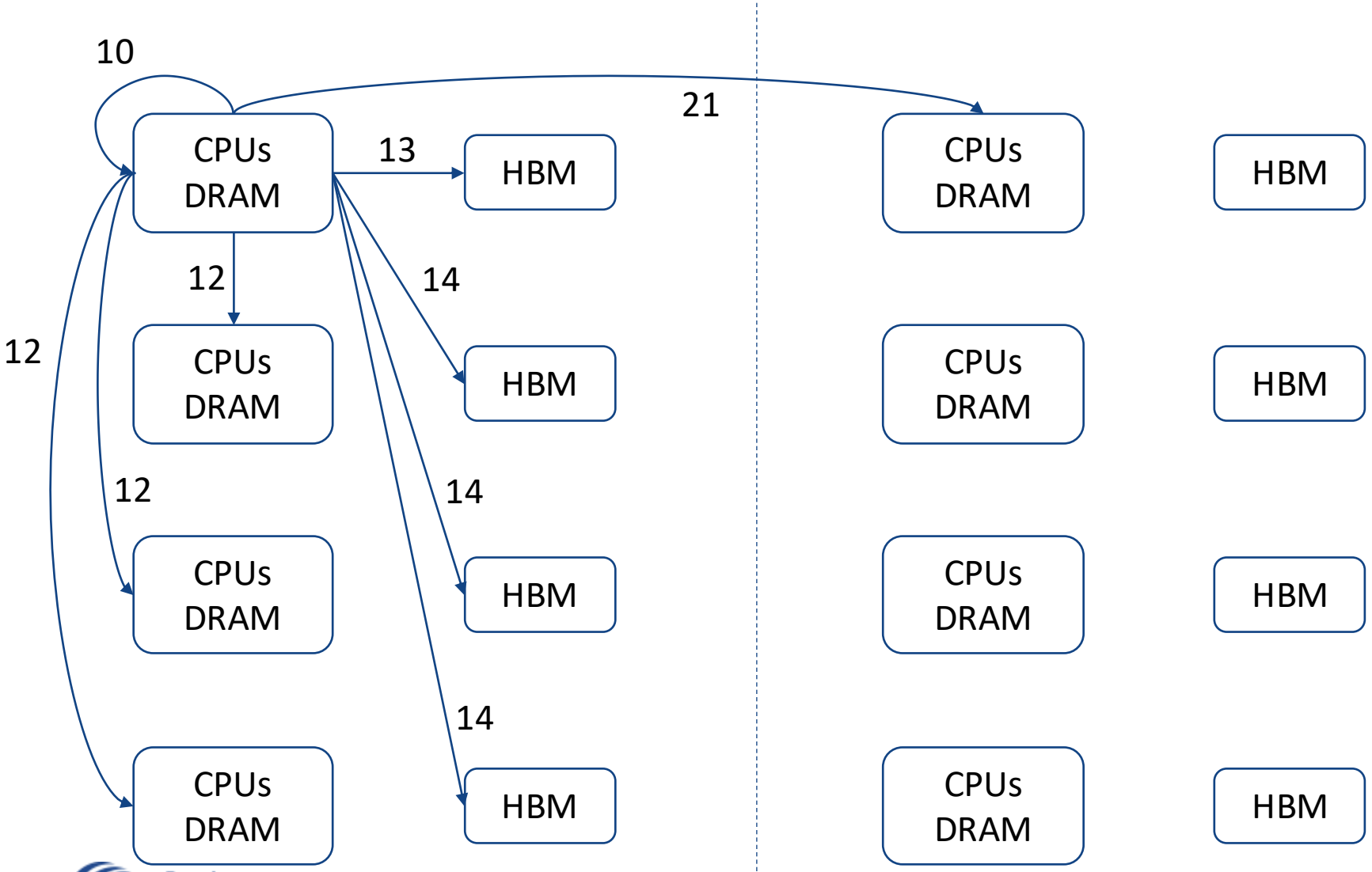
E.g., NUMA Node Dist. in Sapphire Rapids



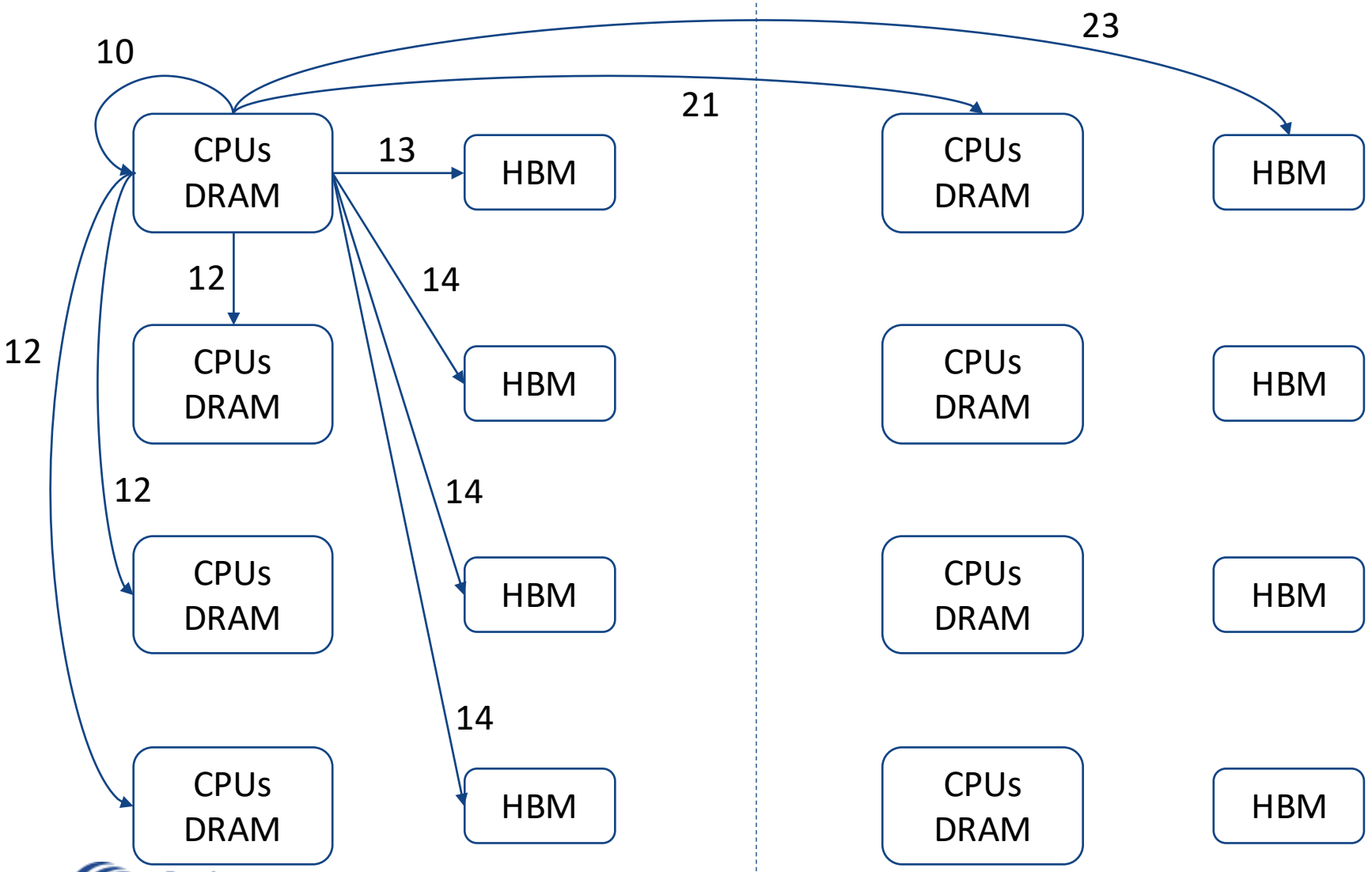
E.g., NUMA Node Dist. in Sapphire Rapids



E.g., NUMA Node Dist. in Sapphire Rapids



E.g., NUMA Node Dist. in Sapphire Rapids



memkind → UMF

- Open-source software that can manage allocations in different memory subsystems
  - https://github.com/memkind (deprecated)
  - https://github.com/oneapi-src/unified-memory-framework
    - Part of Intel's OneAPI

- Constructing allocators and memory pools
- Broadly useful abstractions and utilities for memory management

UMF

- Create an OS memory provider
  - For allocating memory from NUMA nodes visible to the OS
  - Allocations are made with *mmap*

res = umfMemoryProviderCreate(provider_ops, params, &provider);

- Memory allocation (replacement for malloc)

res = umfMemoryProviderAlloc(provider, alloc_size, alignment, &ptr_provider);

Het. Memory Placement Mechanisms

- numactl – too coarse grained
  - Place everything in NUMA node's memory N while it fits, FCFS
- UFM allocator and derivatives (Intel)
  - User specifies memory per allocation
  - What about statically-allocated objects (i.e., stack)?
- Kernel page movement
  - Reactive and page granularity
- OpenMP syntax: **#pragma omp allocate** …
- MPI Standard: **mpi_memory_alloc_kinds** info key

Assisting Users on Het. Memory Mode

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Assisting Users: Background

```
for (i = 0; i < N; i++) {
    a[i] = b[i];
    c[i] = d[f(i)];
}
```

Assisting Users: Background

- Code-oriented profiler: Cost attributed to code locations

```
for (i = 0; i < N; i++) {          5 %
    a[i] = b[i];                  20 %
    c[i] = d[f(i)];               75 %
}                                  -
```

Assisting Users: Background

- Code-oriented profiler: Cost attributed to code locations
- Data-oriented profiler: Cost attributed to memory objects

`for (i = 0; i < N; i++) {`	5 %	a ← 5%
`  a[i] = b[i];`	20 %	b ← 20%
`  c[i] = d[f(i)];`	75 %	c ← 5%
`}`	-	d ← 70%

Data-Oriented Profiling (I)
EVOP: Instrumentation + Cache Simulation

- To enable the **differentiation of memory** objects:
  - Locate the memory object comprising a given memory address
  - Store its associated access data
- Added support to be used from tools
- During the execution of a profiled application:
  - Every data access causing a **last-level cache miss** is checked against matching object
- To tackle profiling overhead for production-sized runs, limit to a region of interest
  - Modifying the code to be profiled
- **Output:**
  - **Per-object LL cache misses during the profiled portion of interest**

```
simulate(max_iter):  /* max_iter = 2 */
  CALLGRIND_START_INSTRUMENTATION
  for i in 1..max_iter:
    /* Simulation original code */
    if i = 1:
      CALLGRIND_ZERO_STATS
  CALLGRIND_STOP_INSTRUMENTATION
```

A. J. Peña and P. Balaji. “A framework for tracking memory accesses in scientific applications”, in P2S2 2014

Data-Oriented Profiling (II)
Extrae: Sampling HW Counters

- Extrae is BSC's monitoring package
- Parallel programming models
  - MPI, OpenMP, pthreads, OmpSs, CUDA, OpenCL, OpenSHMEM, GASPI
- Support for in-production binaries
  - LD_PRELOAD preferred among others (DynInst, re-linking, …)
- Sampling capabilities
  - HWC- (and alarm-) based
- Link to source code
  - Callstack at MPI, pthread, OmpSs, CUDA, OpenCL, OpenSHMEM, malloc()-related routines
  - ~~Need debugging information to translate addresses into source-code references~~
- Intercept I/O calls

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Servat *et al.* "Integrating memory perspective into the BSC performance tools", in P2S2 2017

Tradeoff

- EVOP uses instrumentation and runs on an emulator
  - Detailed but suffers high overhead
  - Simulated cache

- Extrae uses sampling & hardware counters
  - Runs on real target platform
  - Low overhead but lossy

Tradeoff

- EVOP uses instrumentation and runs on an emulator
  - Detailed but suffers high overhead
  - Simulated cache
- Extrae uses sampling & hardware counters
  - Runs on real target platform
  - Low overhead but lossy
- In practice both yield to similar placements
  - Hence we prefer Extrae for performance and simplicity

Data-Oriented Profiling…

- Instrumenting-based solutions (slow)
  - EVOP, MemSpy, SLO, MACPO, Intel Advisor
- HW-based solutions (lossy)
  - Extrae, Sun ONE Studio, HPCToolkit, Intel Vtune Amplifier, MemAxes, ProfDP, …
- Those give data access information to developers
  - Anything more automated possible?

hmem_advisor (BSC)

Maximize the value of the content of a (set of) knapsack(s) given a set of items of different values and sizes

- ***Multiple knapsack*** problem:
  - Knapsacks: memory subsystems
  - Knapsack capacity: memory size
  - Items: memory objects
  - Item weight: size
  - Item value: number of LL load cache misses
    - **CPU stall cycles**

- Not exactly a textbook problem:
  - The different knapsacks modify the value of their items:
    - Multiply cache misses by a different factor: **average latency**

Methodology
Assumptions and Current Known Limitations

- Average latency estimations for the different memory subsystems
- No memory migrations nor reuse of freed space
  - (other than by the same memory object)

Sample Output

dmem_advisor, a memory object distribution tool for heterogeneous memory systems
Copyright (C) 2015, Argonne National Laboratory
Author: Antonio J. Pena <apenya@anl.gov>

-- SP - 8388608 bytes --
niters [HPCCG.cpp:72] - 1 loads - 8 bytes
normr [main.cpp:170] - 1 loads - 8 bytes
t2 [HPCCG.cpp:82] - 1 loads - 8 bytes
t3 [HPCCG.cpp:82] - 1 loads - 8 bytes
rtrans [HPCCG.cpp:94] - 1 loads - 8 bytes
8 - 1 loads - 96 bytes
--
6 objects; 136 bytes (0.00162124633789%); 1080 saved

-- DRAM - 34359738368 bytes --
--
0 objects; 0 bytes (0.0%); 0 saved

-- 3D - 8589934592 bytes --
52 - 12566534 loads - 134217728 bytes
48 - 6291459 loads - 134217728 bytes
56 - 4194306 loads - 134217728 bytes
12 - 1048577 loads - 67108864 bytes
16 - 2097153 loads - 134217728 bytes
20 - 2097153 loads - 134217728 bytes
28 - 2097153 loads - 134217728 bytes
44 - 28090945 loads - 1811939328 bytes
40 - 56181888 loads - 3623878656 bytes
--
9 objects; 6308233216 bytes (73.4375%); 7453235920 saved

-- WHEREVER --
niters [main.cpp:169] - 4 bytes
ruse [mytimer.cpp:104] - 144 bytes
--
2 objects; 148 bytes

Sample Output

dmem_advisor, a memory object distribution tool for heterogeneous memory systems
Copyright (C) 2015, Argonne National Laboratory
Author: Antonio J. Pena <apenya@anl.gov>

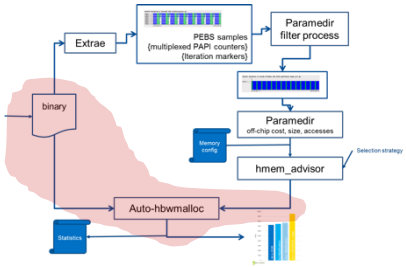
-- SP - 8388608 bytes --
niters [HPCCG.cpp:72] - 1 loads - 8 bytes
normr [main.cpp:170] - 1 loads - 8 bytes
t2 [HPCCG.cpp:82] - 1 loads - 8 bytes
t3 [HPCCG.cpp:82] - 1 loads - 8 bytes
rtrans [HPCCG.cpp:94] - 1 loads - 8 bytes
8 - 1 loads - 96 bytes
--
6 objects; 136 bytes (0.00162124633789%); 1080 saved

-- DRAM - 34359738368 bytes --
--
0 objects; 0 bytes (0.0%); 0 saved

-- 3D - 8589934592 bytes --
52 - 12566534 loads - 134217728 bytes
48 - 6291459 loads - 134217728 bytes
56 - 4194306 loads - 134217728 bytes
12 - 1048577 loads - 67108864 bytes
16 - 2097153 loads - 134217728 bytes
20 - 2097153 loads - 134217728 bytes
28 - 2097153 loads - 134217728 bytes
44 - 28090945 loads - 1811939328 bytes
40 - 56181888 loads - 3623878656 bytes
--
9 objects; 6308233216 bytes (73.4375%); 7453235920 saved

-- WHEREVER --
niters [main.cpp:169] - 4 bytes
ruse [mytimer.cpp:104] - 144 bytes
--
2 objects; 148 bytes

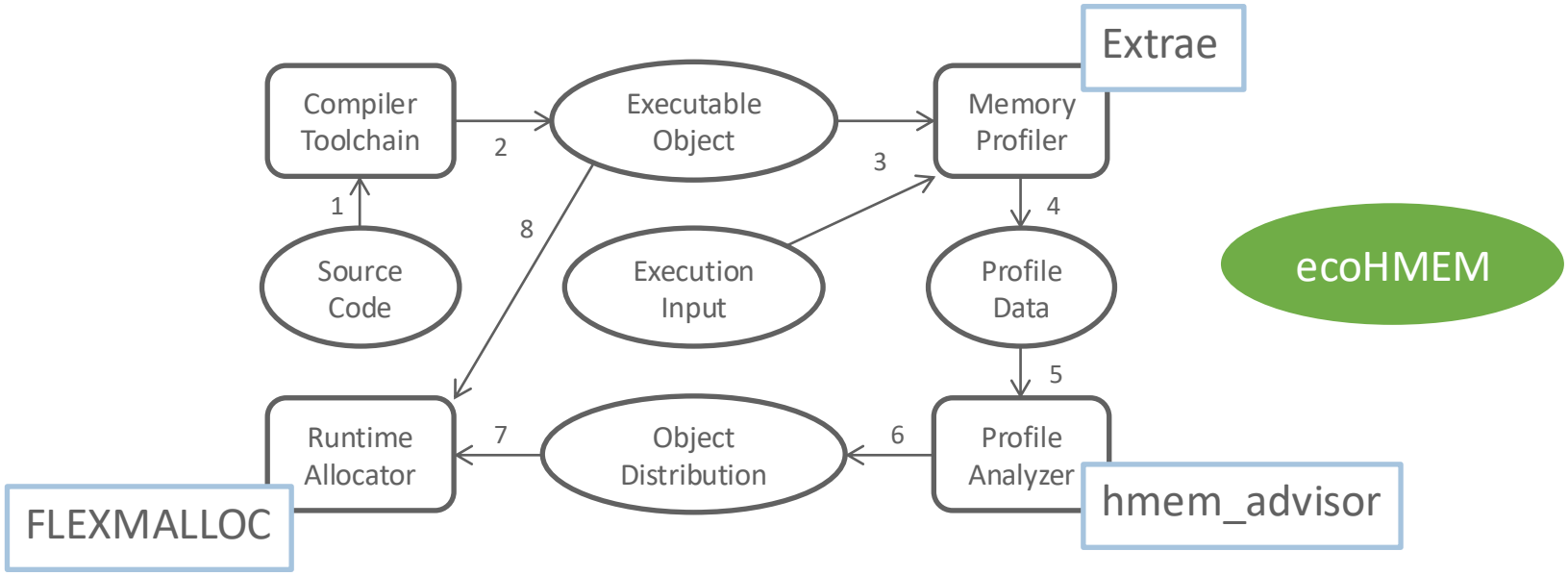
FlexMalloc (by H. Servat, Intel)



- Runtime component
- For each malloc/realloc/free call check if it is to be substituted
- Use in-production binaries (the same as profiling)
  - Recompiled binaries may not work
    - Will not match (find) memory objects
  - Inlining may reduce effectiveness providing false positive locations in deeper call-stacks but still work
- Only applies to dynamic allocations
  - Static allocations need to be handled manually (source code change)

Methodology

- Object-differentiated data-oriented profiling + distribution algorithm (analysis):
  1. Profile to determine per-object last-level cache misses / avg. access time
  2. Assess the optimal distribution of the different objects among the memory subsystems
     - Minimize processor stall cycles

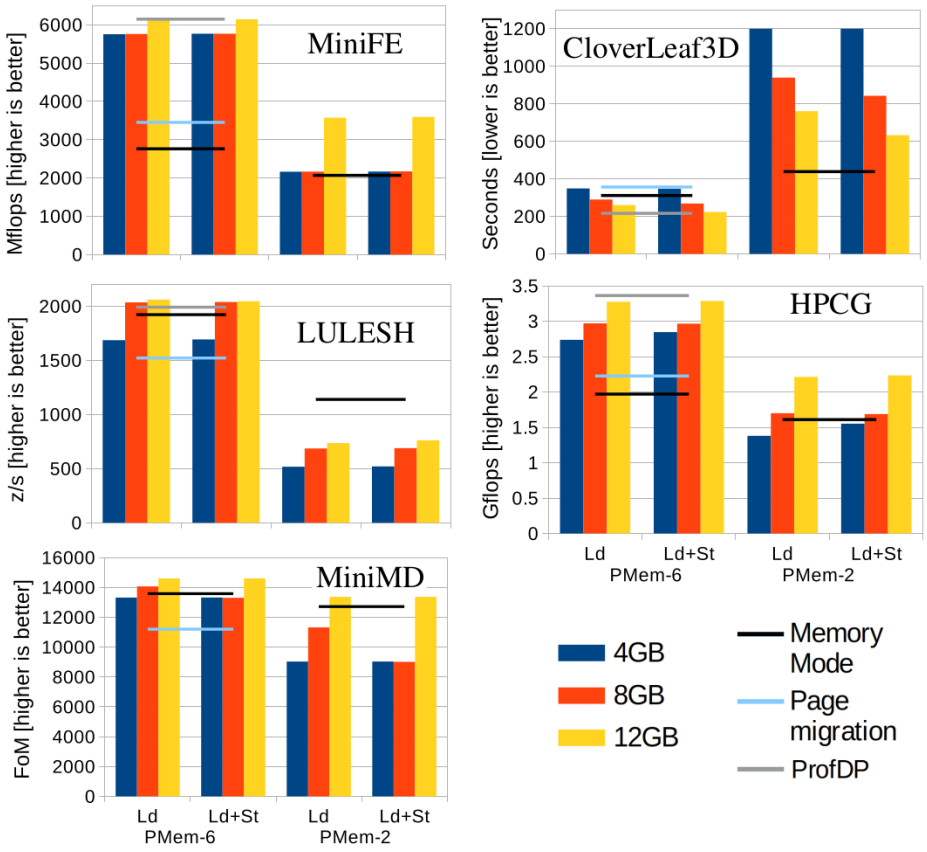


M. Jordà *et al.*, "ecoHMEM: Improving object placement methodology for hybrid memory systems in HPC", in *IEEE Cluster 2022*

Some Results

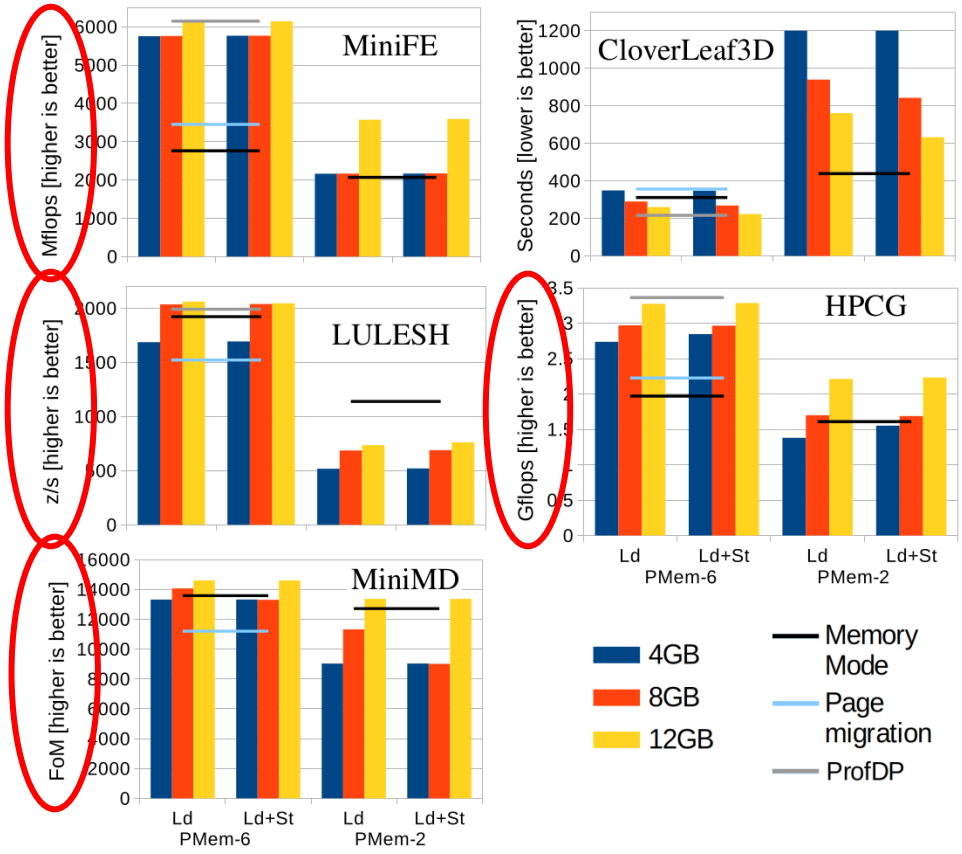
Some Results

Base algorithm



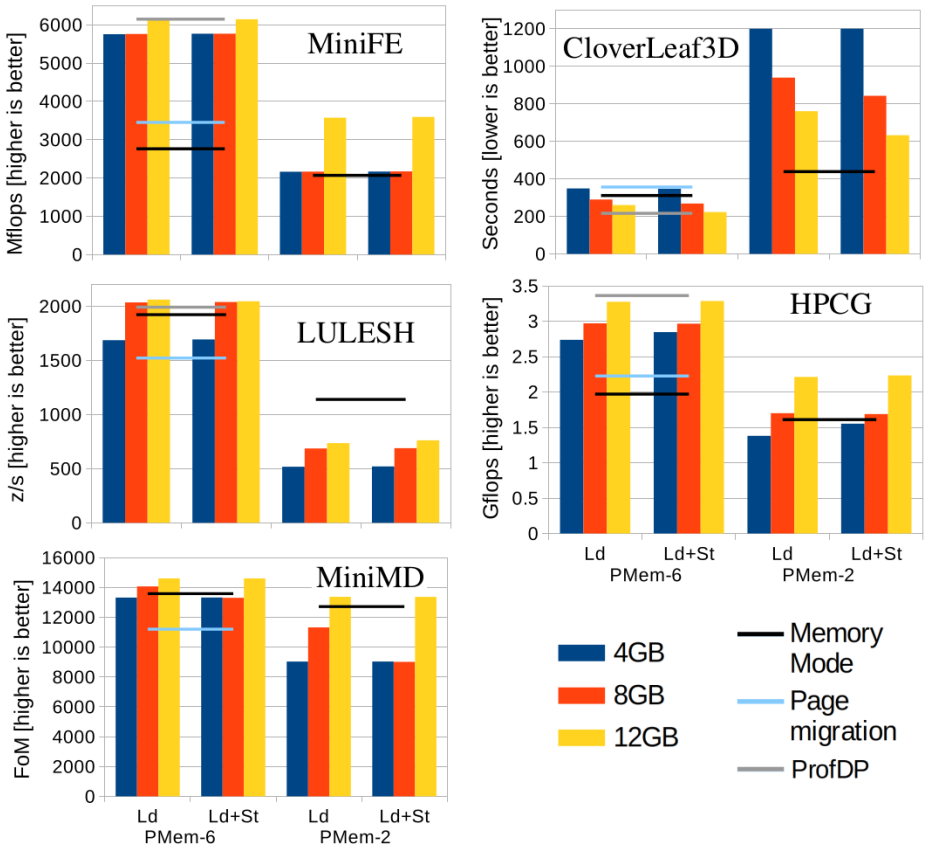
Some Results

Base algorithm



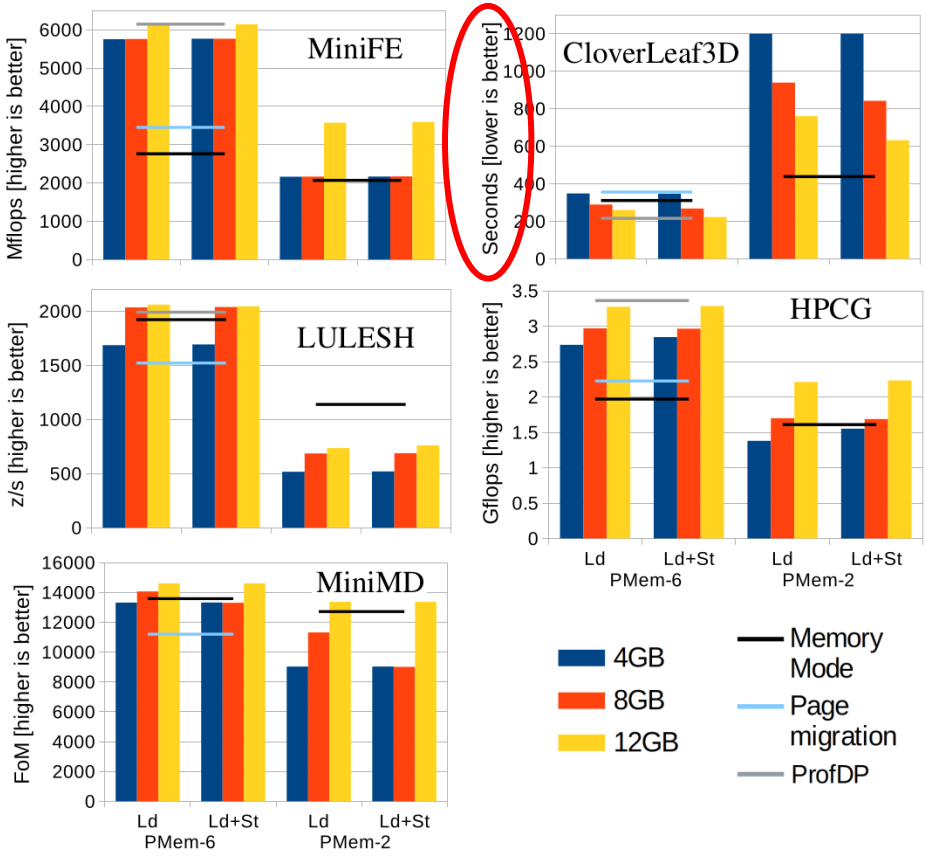
Some Results

Base algorithm



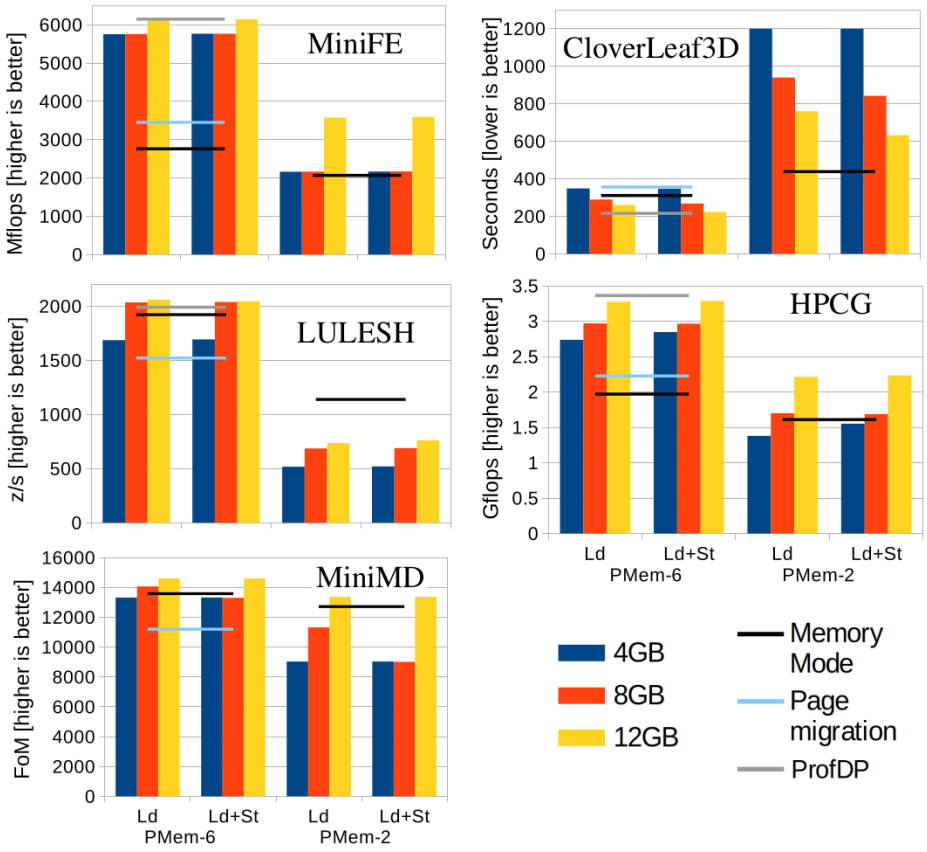
Some Results

Base algorithm



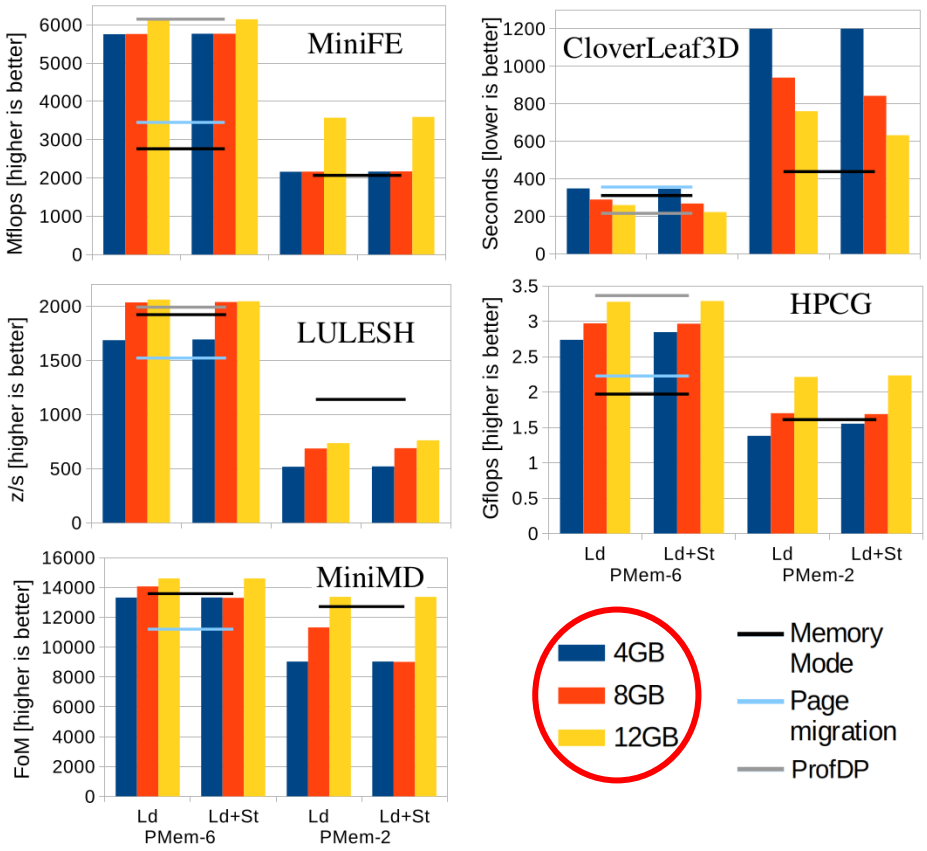
Some Results

Base algorithm



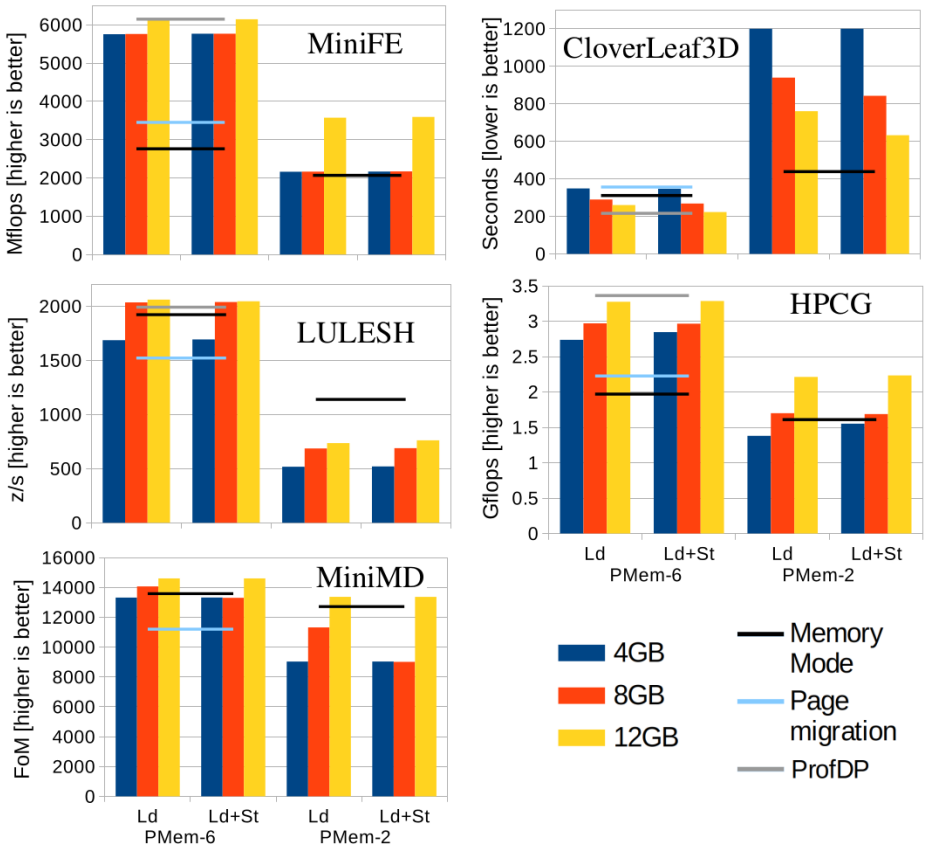
Some Results

Base algorithm



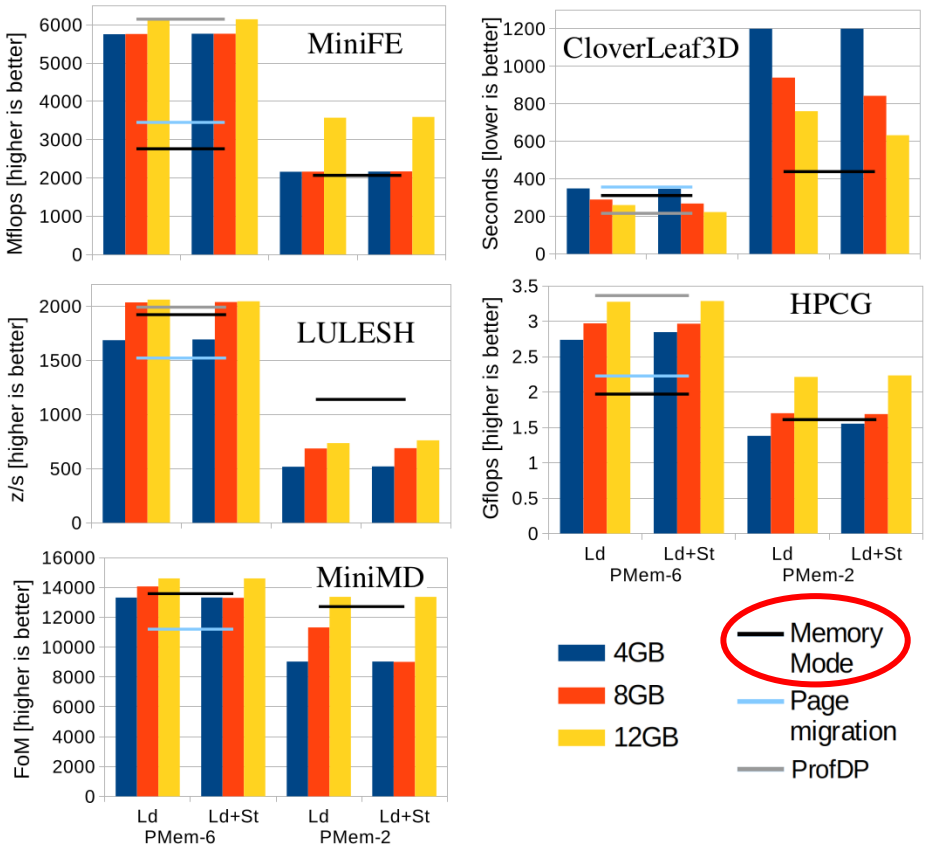
Some Results

Base algorithm



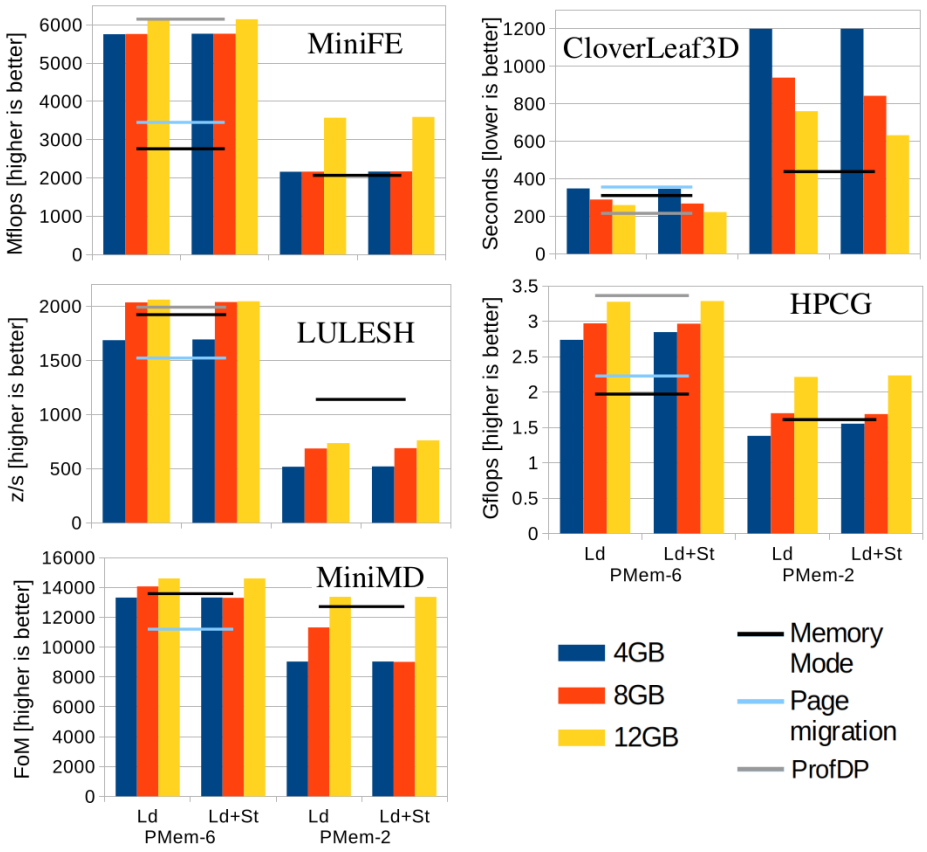
Some Results

Base algorithm



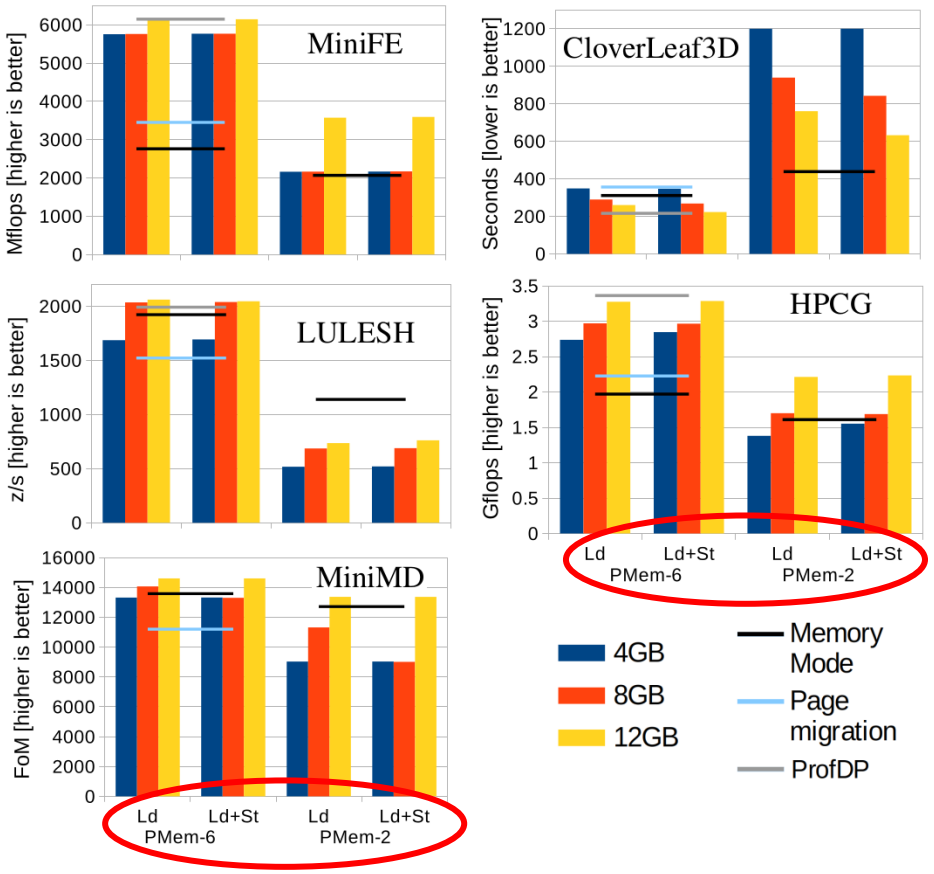
Some Results

Base algorithm



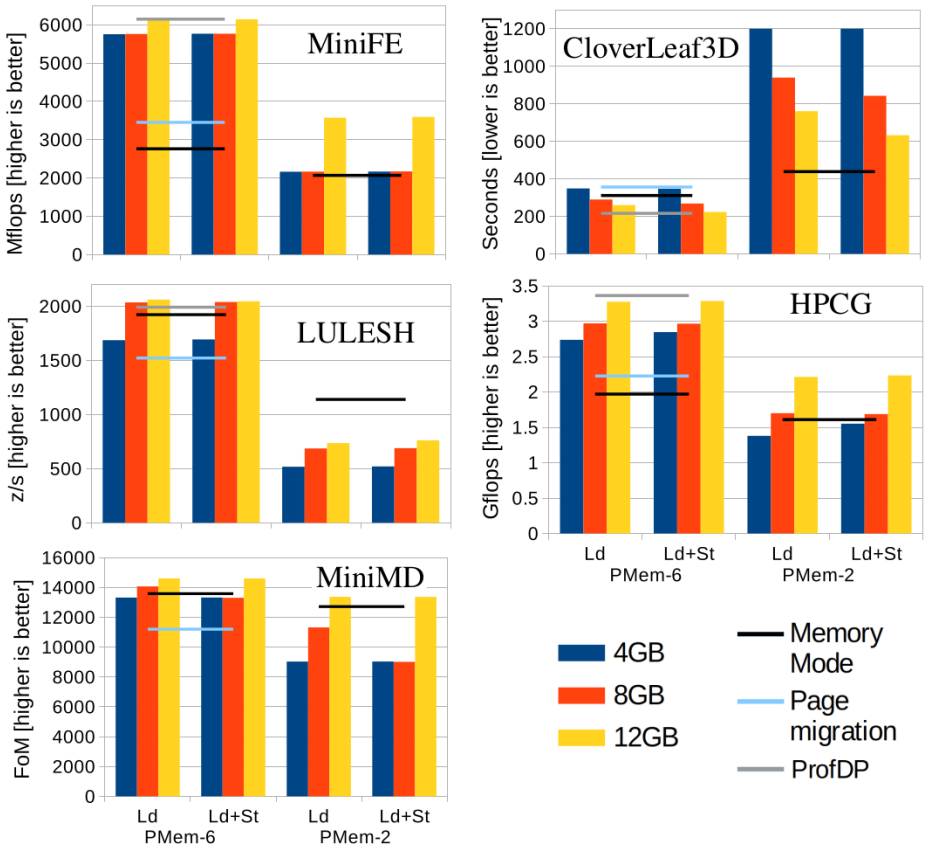
Some Results

Base algorithm



Some Results

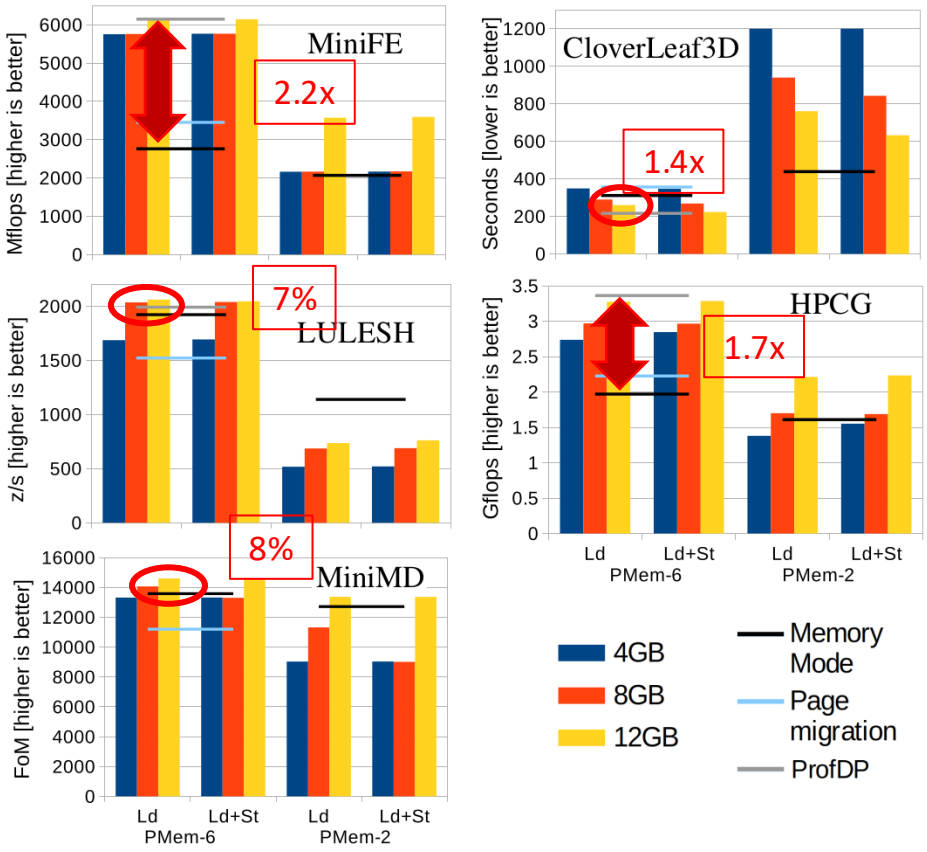
Base algorithm



Some Results

Base algorithm

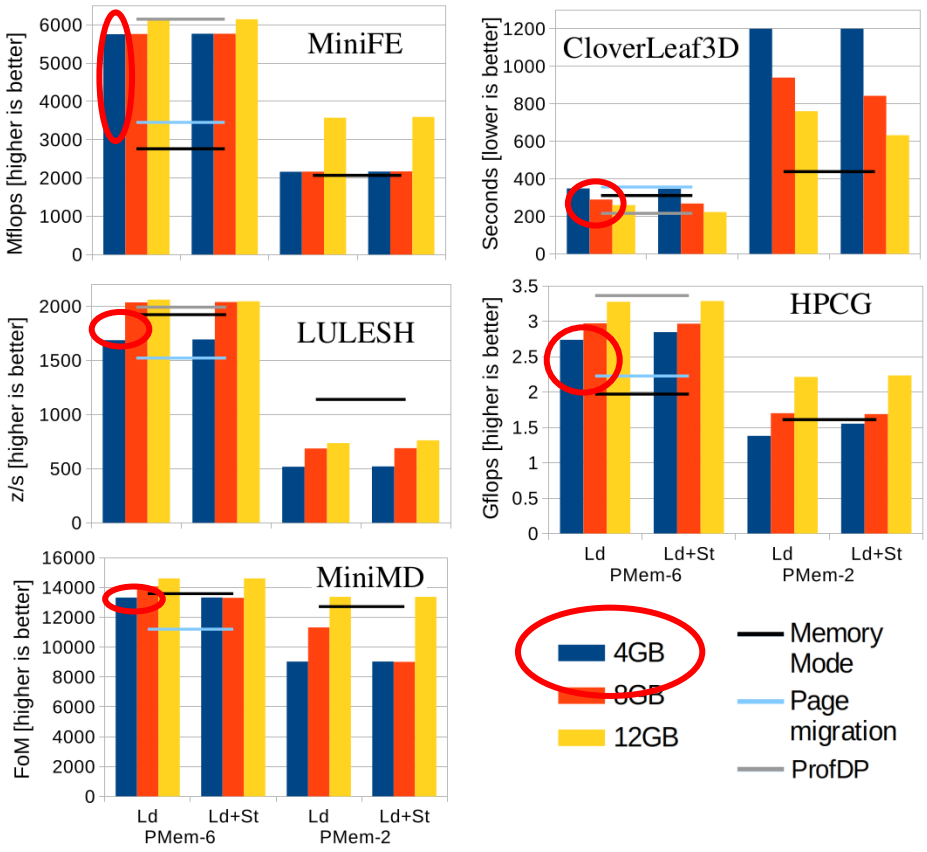
- With 12GB, in **all PMem-6** and most Pmem-2 cases, our framework performs better than MM and KPM
  - Reasons: MM cache misses, memory bound
  - Breakdown & further details in the paper



Some Results

Base algorithm

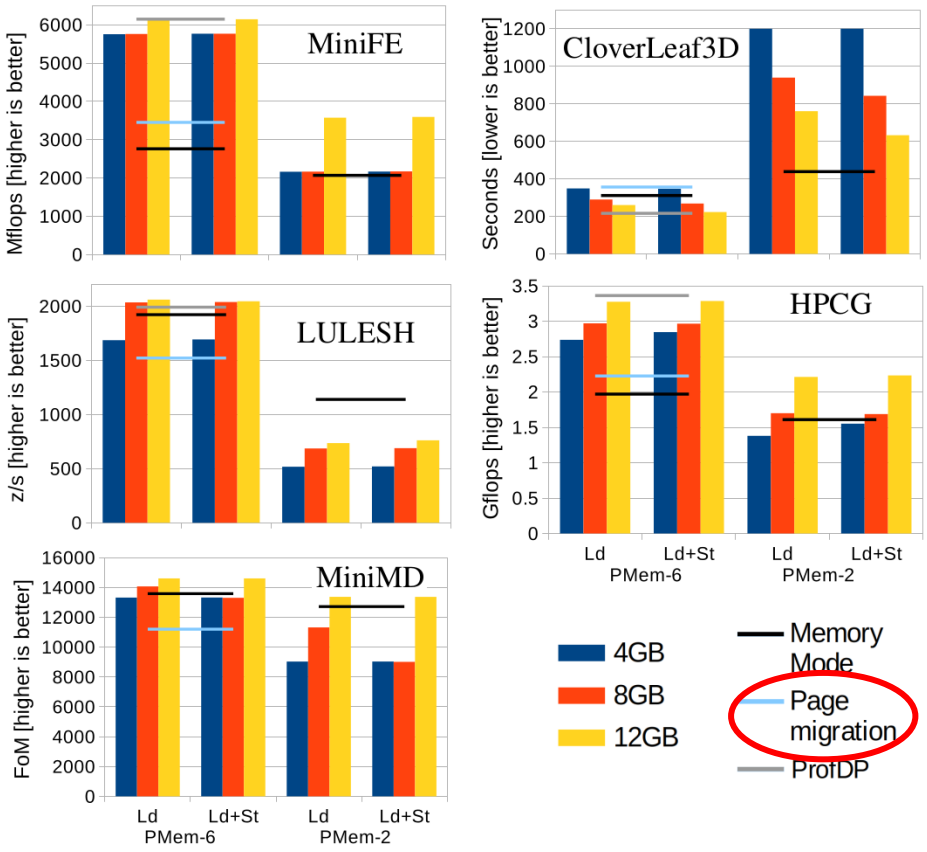
- With 12GB, in **all PMem-6** and most Pmem-2 cases, our framework performs better than MM and KPM
  - Reasons: MM cache misses, memory bound
  - Breakdown & further details in the paper
- ¼ DRAM: Little performance drop



Some Results

Base algorithm

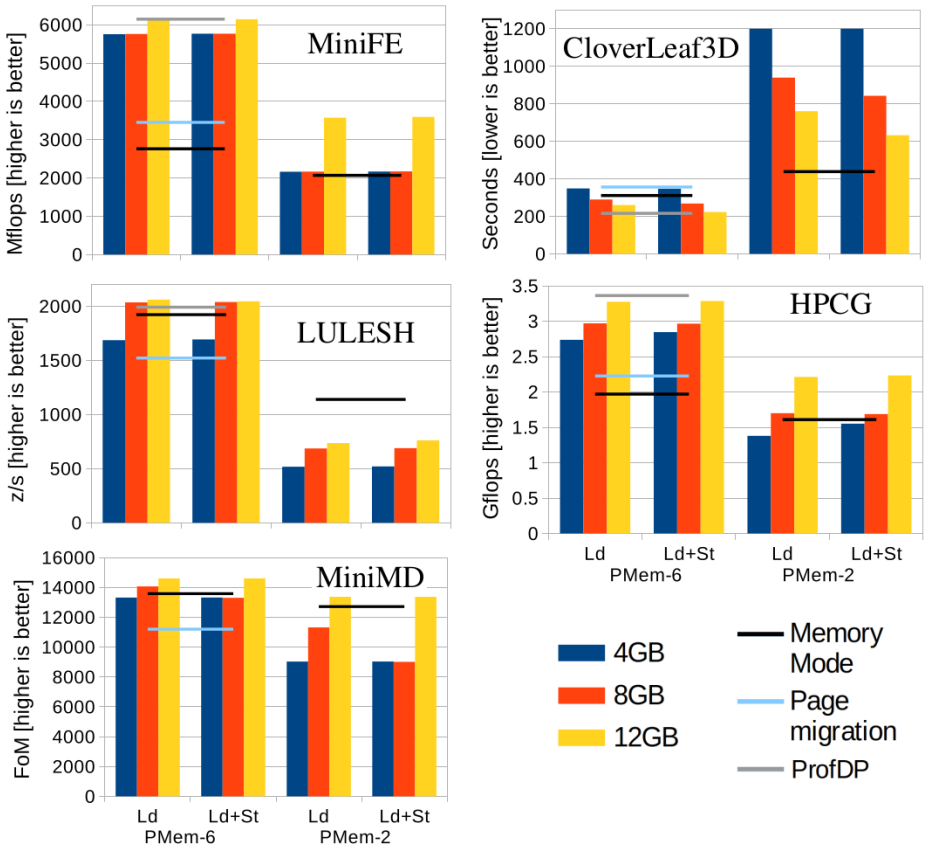
- With 12GB, in **all PMem-6** and most Pmem-2 cases, our framework performs better than MM and KPM
  - Reasons: MM cache misses, memory bound
  - Breakdown & further details in the paper
- ¼ DRAM: Little performance drop
- Kernel Page Migration (Intel *tiering*)
  - Lower performance
  - DRAM cost for page management metadata proportional to PMem size (15GB in our case); limits DRAM left for app.



Some Results

Base algorithm

- With 12GB, in **all PMem-6** and most Pmem-2 cases, our framework performs better than MM and KPM
  - Reasons: MM cache misses, memory bound
  - Breakdown & further details in the paper
- ¼ DRAM: Little performance drop
- Kernel Page Migration (Intel *tiering*)
  - Lower performance
  - DRAM cost for page management metadata proportional to PMem size (15GB in our case); limits DRAM left for app.
- LAMMPS: within negligible performance difference

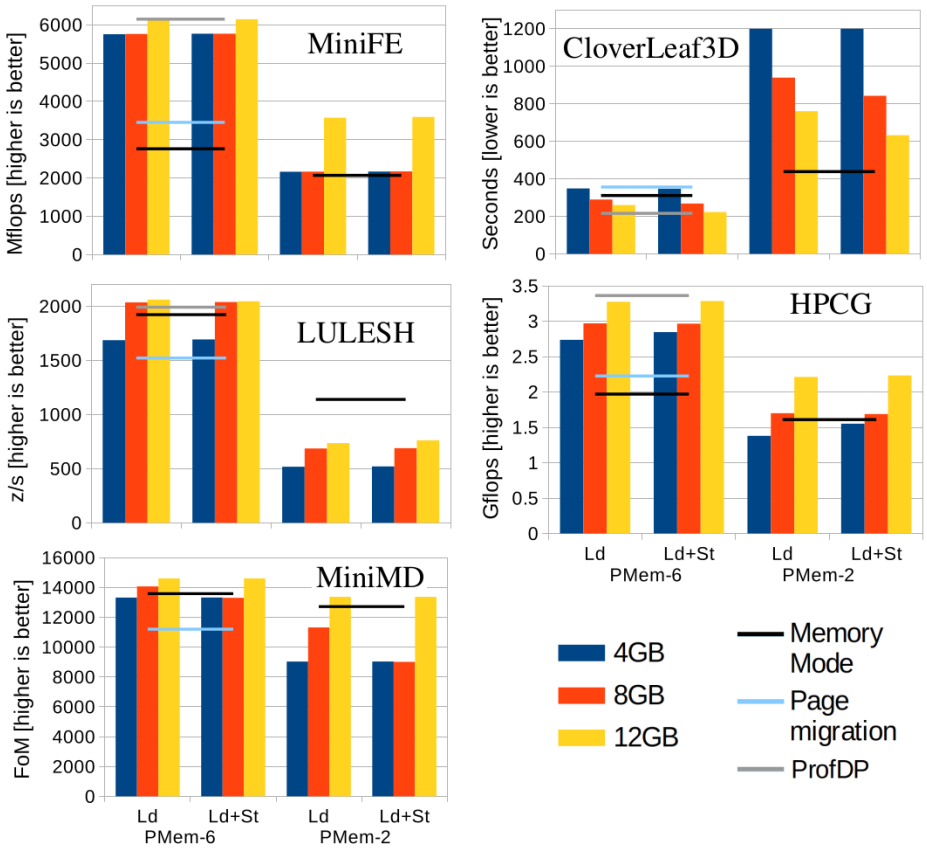


Some Results

Base algorithm

- With 12GB, in **all PMem-6** and most Pmem-2 cases, our framework performs better than MM and KPM
  - Reasons: MM cache misses, memory bound
  - Breakdown & further details in the paper
- ¼ DRAM: Little performance drop
- Kernel Page Migration (Intel *tiering*)
  - Lower performance
  - DRAM cost for page management metadata proportional to PMem size (15GB in our case); limits DRAM left for app.
- LAMMPS: within negligible performance difference

BW-Aware Algorithm



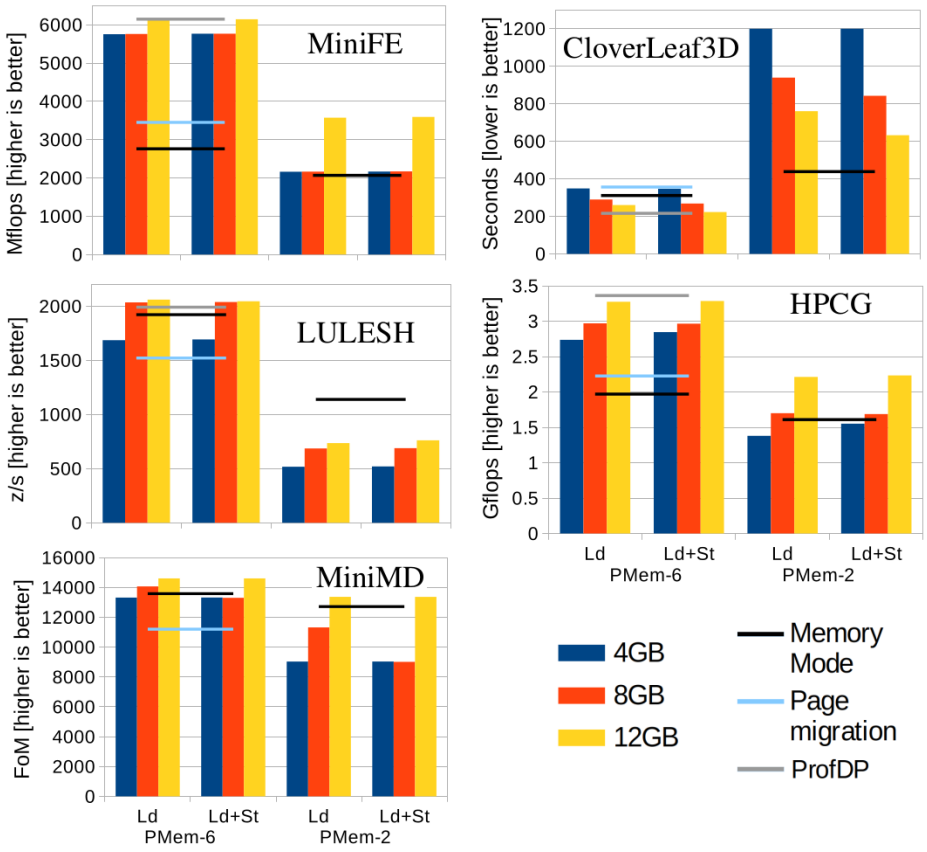
Some Results

Base algorithm

- With 12GB, in **all PMem-6** and most Pmem-2 cases, our framework performs better than MM and KPM
  - Reasons: MM cache misses, memory bound
  - Breakdown & further details in the paper
- ¼ DRAM: Little performance drop
- Kernel Page Migration (Intel *tiering*)
  - Lower performance
  - DRAM cost for page management metadata proportional to PMem size (15GB in our case); limits DRAM left for app.
- LAMMPS: within negligible performance difference

BW-Aware Algorithm

- LULESH: 7% → 19% improvement



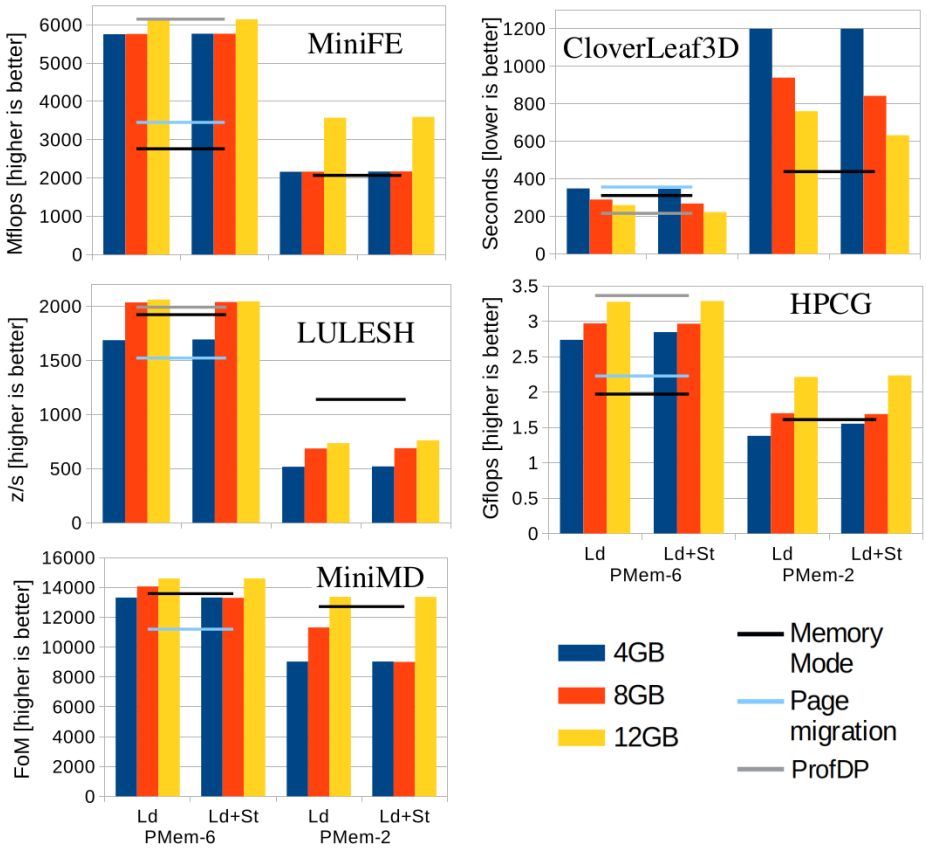
Some Results

Base algorithm

- With 12GB, in **all PMem-6** and most Pmem-2 cases, our framework performs better than MM and KPM
  - Reasons: MM cache misses, memory bound
  - Breakdown & further details in the paper
- ¼ DRAM: Little performance drop
- Kernel Page Migration (Intel *tiering*)
  - Lower performance
  - DRAM cost for page management metadata proportional to PMem size (15GB in our case); limits DRAM left for app.
- LAMMPS: within negligible performance difference

BW-Aware Algorithm

- LULESH: 7% → 19% improvement
- OpenFOAM: 35% overhead → 6% speedup



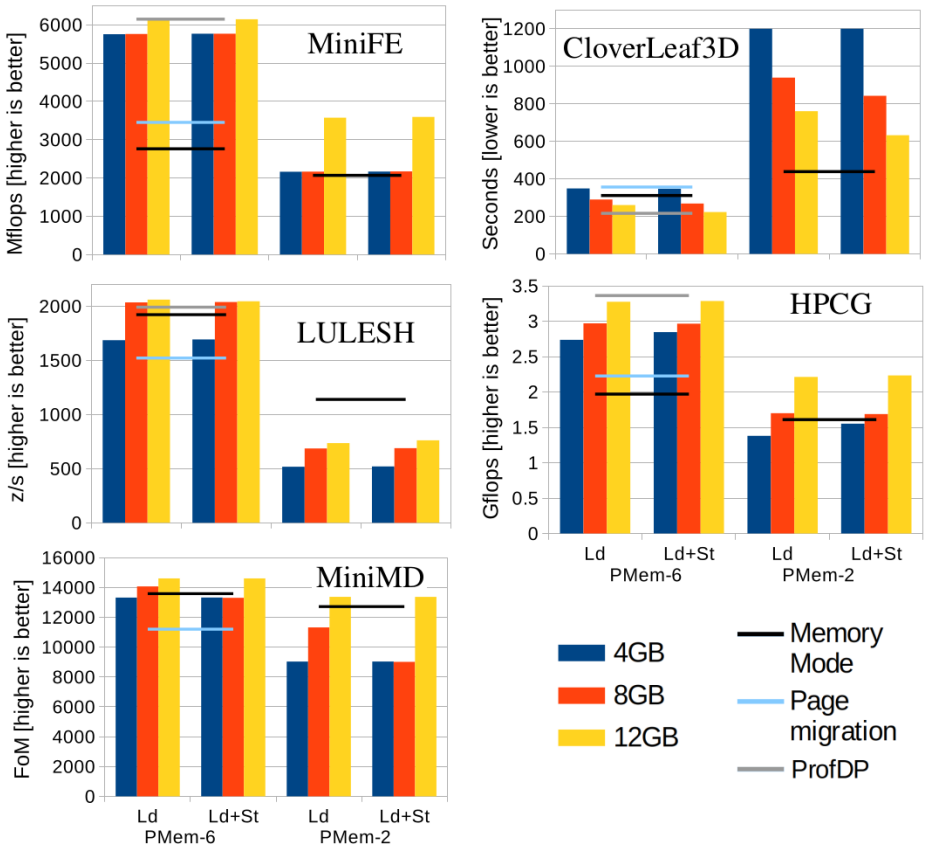
Some Results

Base algorithm

- With 12GB, in **all PMem-6** and most Pmem-2 cases, our framework performs better than MM and KPM
  - Reasons: MM cache misses, memory bound
  - Breakdown & further details in the paper
- ¼ DRAM: Little performance drop
- Kernel Page Migration (Intel *tiering*)
  - Lower performance
  - DRAM cost for page management metadata proportional to PMem size (15GB in our case); limits DRAM left for app.
- LAMMPS: within negligible performance difference

BW-Aware Algorithm

- LULESH: 7% → 19% improvement
- OpenFOAM: 35% overhead → 6% speedup
- BSIT: 1.87x w.r.t. I/O ☺



Only 6%? Why would I care?

Only 6%? Why would I care?

- Didn’t find any use case with non-negligible performance drop-down

Only 6%? Why would I care?

- Didn’t find any use case with non-negligible performance drop-down

- Improvements from 0% to 2x
  - In return for just a profiling run
  - No source code mods or recompilation!

Only 6%? Why would I care?

- Didn’t find any use case with non-negligible performance drop-down
- Improvements from 0% to 2x
  - In return for just a profiling run
  - No source code mods or recompilation!
- Great results for DRAM reduction: ¼ DRAM with little or no penalty

Some Results in Intel Sapphire Rapids

- Compared to DRAM-only (default) executions:
  - LULESH: 12.4% speedup
  - miniFE: 11.3% speedup
- Seamless use of HBM with >10% gains ☺

ecoHMEM

- FlexMalloc contributed by Intel
  - https://github.com/intel/flexmalloc

- Download tarball from BSC SW repository:
  - https://www.bsc.es/discover-bsc/organisation/scientific-structure/accelerators-and-communications-hpc/team-software

- Source code currently available from EPEEC's Project repository:
  - https://github.com/epeec/ecoHMEM

Highlight Use Case on Deep Memory

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

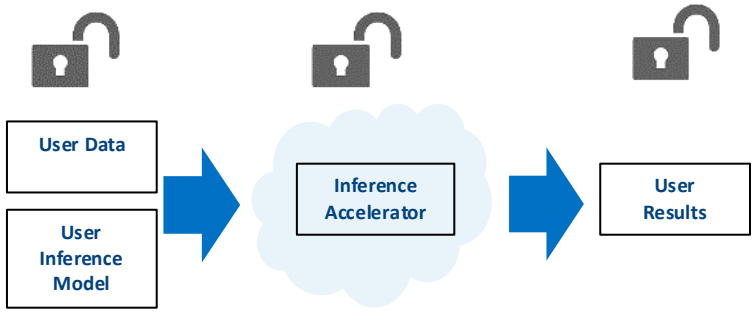
Homomorphic Encryption (HE)

- HE allows inference on **encrypted data** using **encrypted models.**
- Allows user to submit inference jobs externally without disclosing proprietary models & data
- Why is HE relevant to HPC?
  - >100x memory requirements
  - >100x compute time

Homomorphic Encryption (HE)

- HE allows inference on **encrypted data** using **encrypted models.**
- Allows user to submit inference jobs externally without disclosing proprietary models & data
- Why is HE relevant to HPC?
  - >100x memory requirements
  - >100x compute time

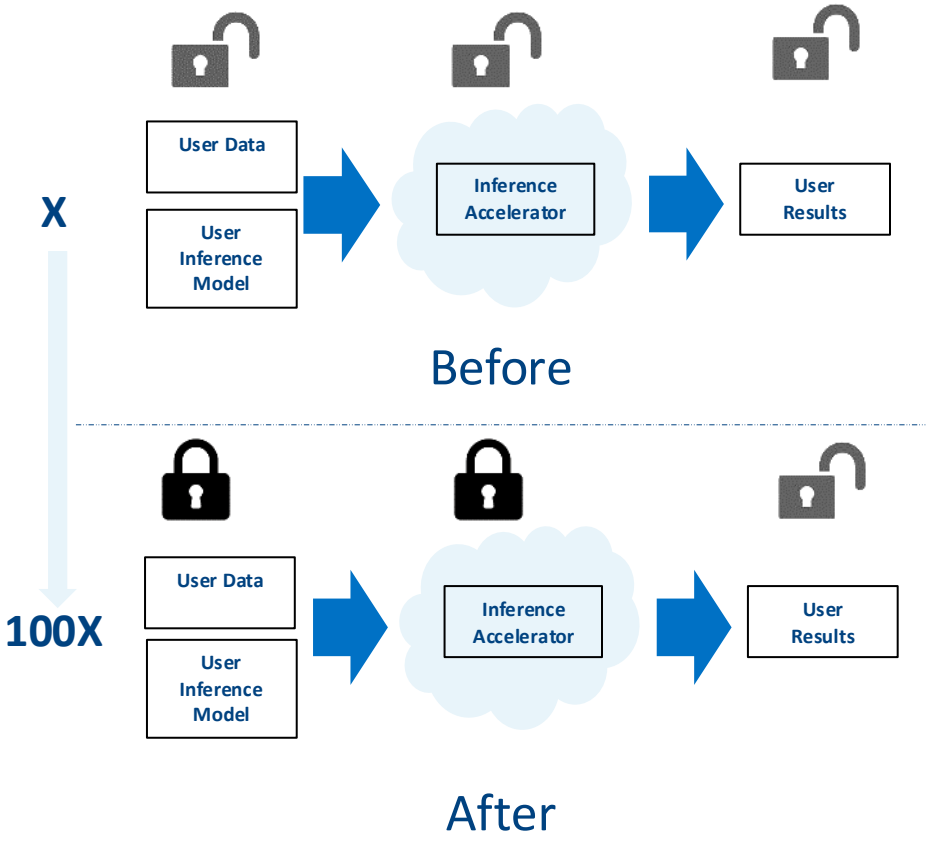
X



Before

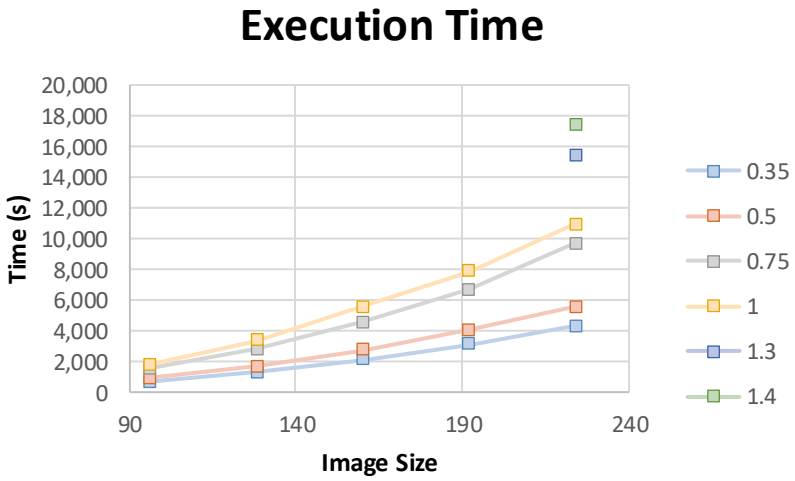
Homomorphic Encryption (HE)

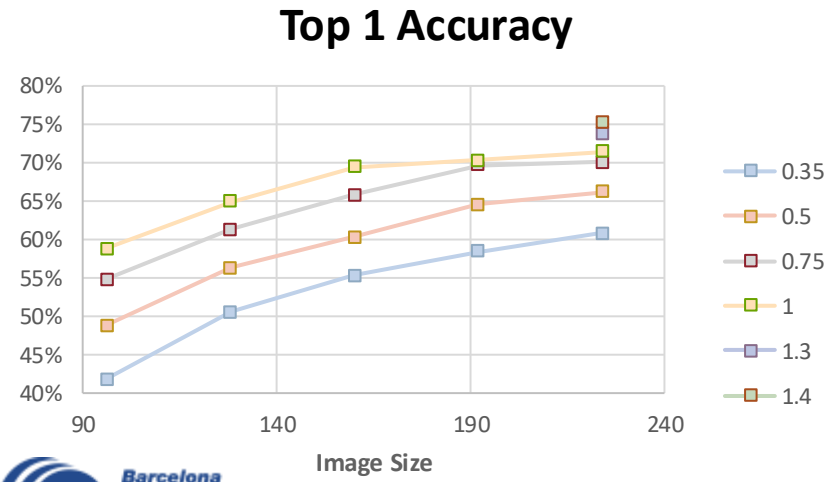
- HE allows inference on **encrypted data** using **encrypted models.**
- Allows user to submit inference jobs externally without disclosing proprietary models & data
- Why is HE relevant to HPC?
  - >100x memory requirements
  - >100x compute time

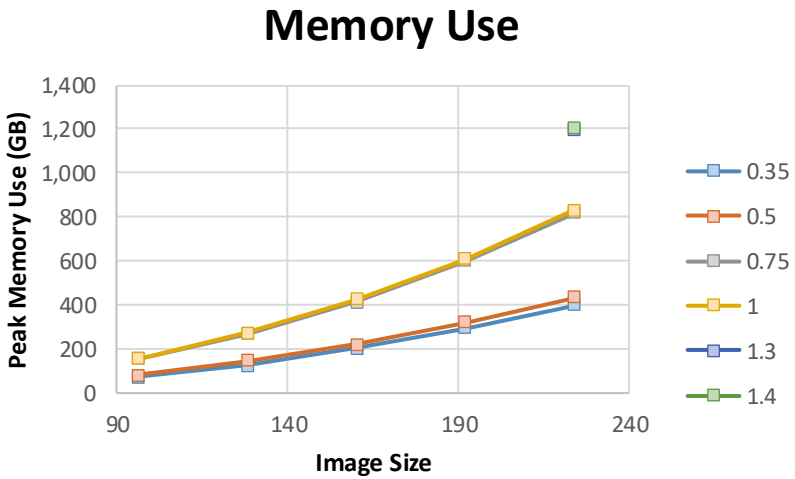


Unprecedented Accomplishment

- We ran the largest (plaintext) model at that date using HE input data
- Biggest to date was MobileNetV2 at expansion factor of 0.35
- Limited by DRAM sizes
- We run MobileNetV2 at max. expansion factor (x3 previous size)
- Also ResNet-50, 1st non-toy DNN
- Thanks to Intel Optane DIMMs
- Memory mode yields high efficiency







Barcelona Supercomputing Center
Centro Nacional de Supercomputación

G. Lloret-Talavera *et al.*, "Enabling homomorphically encrypted inference for large DNN models", Transactions on Computers, IEEE, May 2022.

Conclusion

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Summary

Summary

- Heterogeneity is here to stay and for good reasons

Summary

- Heterogeneity is here to stay and for good reasons
- Not only heterogeneous processing elements

Summary

- Heterogeneity is here to stay and for good reasons
- Not only heterogeneous processing elements
  - Also memory and others

Summary

- Heterogeneity is here to stay and for good reasons
- Not only heterogeneous processing elements
  - Also memory and others
- Heterogeneous memory management APIs in production

Summary

- Heterogeneity is here to stay and for good reasons
- Not only heterogeneous processing elements
  - Also memory and others
- Heterogeneous memory management APIs in production
  - Little help on deciding where to place data

Summary

- Heterogeneity is here to stay and for good reasons
- Not only heterogeneous processing elements
  - Also memory and others
- Heterogeneous memory management APIs in production
  - Little help on deciding where to place data
- Research efforts on automatic/guided data distribution

Summary

- Heterogeneity is here to stay and for good reasons
- Not only heterogeneous processing elements
  - Also memory and others
- Heterogeneous memory management APIs in production
  - Little help on deciding where to place data
- Research efforts on automatic/guided data distribution
  - Besides automatic HW and kernel memory movements

hmem-workshop

Sixth Workshop on Heterogeneity and Memory Systems (HMEM 2025)

In conjunction with SC'25, St. Louis, MO, November 17th, 2025

Overview and scope

Heterogeneity is ubiquitous, not only in terms of processing units but also memories and networks. As heterogeneity increases, memory subsystems play an even more important role to attain performance, from their technology to the system architecture to the software management and programming model. While CPU-only compute nodes are becoming rare instances, heterogeneous memory architectures have recently emerged and revolutionized the traditional memory hierarchy. Today's and upcoming architectures may well comprise multiple memory technologies next to DRAM, accelerators with dedicated memories, or even specific expansion cards hosting memory alone, such as: 3D-stacked memory, high-bandwidth multi-channel RAM, unified/shared memory on accelerators, Compute Express Link (CXL)-based architectures, persistent memory, or MRDIMMs.

As in previous years, the Workshop on Heterogeneous Memory Systems, now rebranded as Heterogeneity and Memory Systems (HMEM), will bring together different research efforts and expertise to the end of integrating different approaches and democratizing the use of resource heterogeneity from a memory perspective, to benefit applications not only in terms of performance, but also energy efficiency and cost trade-offs. The main goal of the workshop is to push the research frontiers forward by exchanging knowledge and debating ideas through featured talks, technical paper presentations, and interactive discussions. Overall, topics of interest include, but are not limited to:

- Resource heterogeneity (e.g., accelerators) and memory implications, including memory designs, data layouts, etc.
- Data allocation and placement techniques in heterogeneous memory systems
- Caching for heterogeneous memory systems
- Programming models and tools for complex/heterogeneous memory hierarchies
- Software-defined far memories
- Disaggregated memory and in-memory computing
- Data movement in heterogeneous memory systems
- Memory consistency and persistency models
- Data structures for heterogeneous memory infrastructures
- Abstractions and support for failure-atomicity in persistent memory

Last, but not Least…

- Open to advise MS Theses, Final Year Projects, host internships, collaborations, …

  https://www.bsc.es/discover-bsc/organisation/scientific-structure/accelerators-and-communications-hpc

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Accelerators and Communications for HPC

Group Manager: Antonio J. Peña

Resource Heterogeneity

Sergio Iserte (R3)

Marc Jordà (RE3)

Mariano Benito (R2)

Michele Esposito (R2)

Muhammad Usman (RE2)

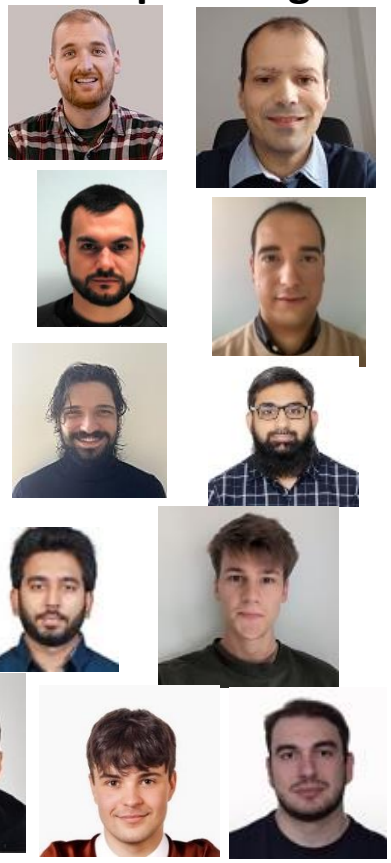
Nikhil Molugu (RE2)

Petter Sandas (R1)

Patrick Cerka (RE1)

Pablo Saura (RE1)

Íñigo Aréjula (RE1)



Homomorphic Encryption

Zaira Pindado (R3)

Lena Martens (RE3)

Xin Gao (R2)

Thomas Spendlhofer (R2)

Priyam Mehta (RE2)

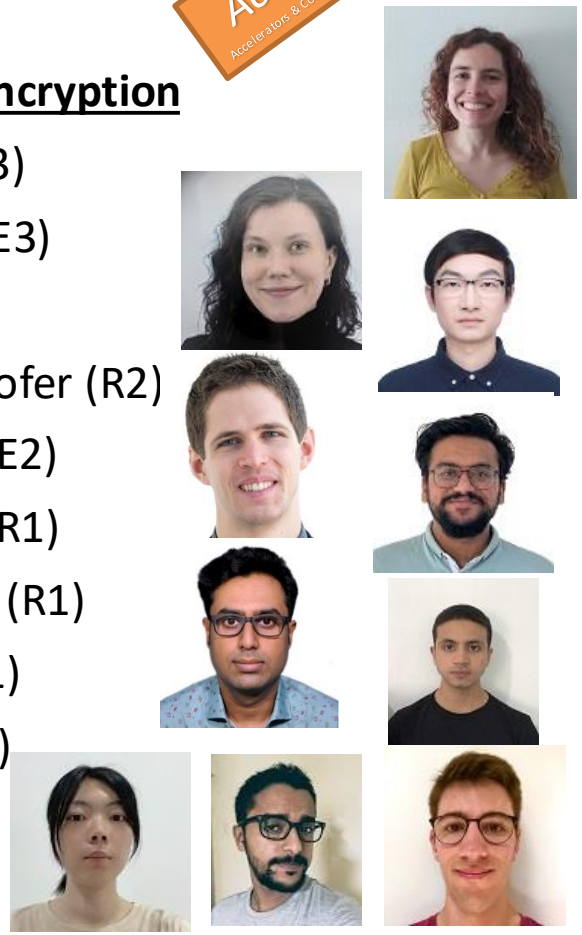
Tathagata Barik (R1)

Mohamed Allam (R1)

Shuxin Zheng (R1)

Gouri Sankar (R1)

Hugo Sanz (RE1)



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Thank you

antonio.pena@bsc.es