Runtime Data Management on Non-volatile Memory-based Heterogeneous Memory for Task-Parallel Programs

Kai Wu Jie Ren Dong Li

University of California, Merced

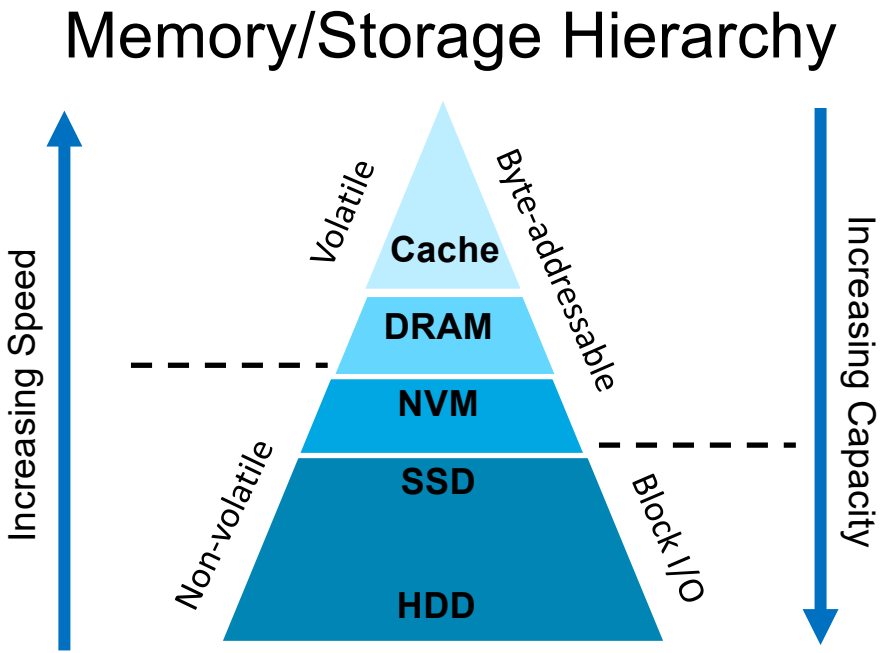
PASA Lab

SC'18

Non-volatile Memory is Promising

- Fast byte-addressable and persistent NVM technologies are coming
  - NVM has good performance

	HDD	SSD	NVM	DRAM
Latency	7.1 ms	68 us	2-500 ns	100 ns
Bandwidth	2.6 MB/s	250 MB/s	5 GB/s	64 GB/s

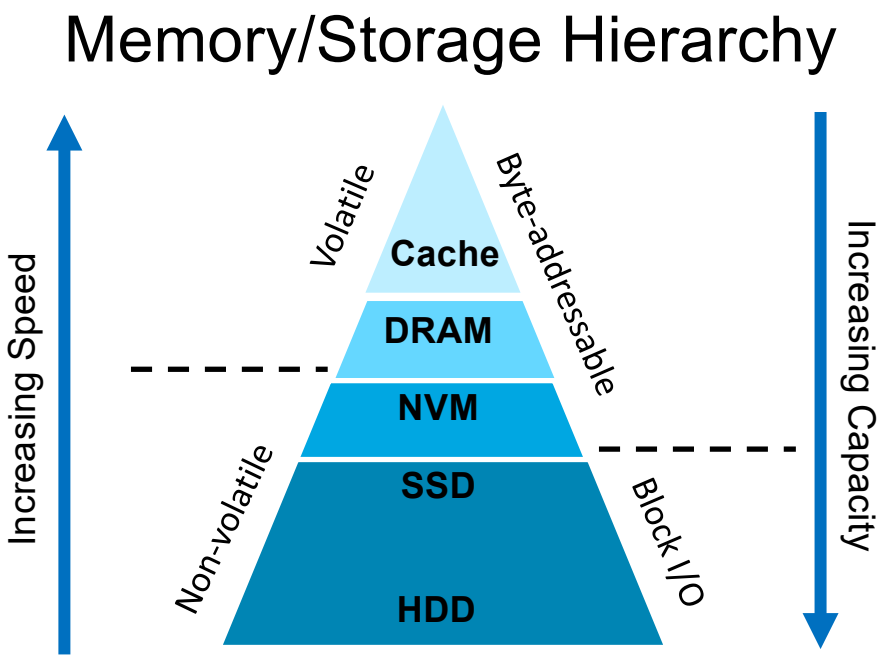


Non-volatile Memory is Promising

- Fast byte-addressable and persistent NVM technologies are coming
  - NVM has good performance but still not enough

	HDD	SSD	NVM	DRAM
Latency	7.1 ms	68 us	2-500 ns	100 ns
Bandwidth	2.6 MB/s	250 MB/s	5 GB/s	64 GB/s

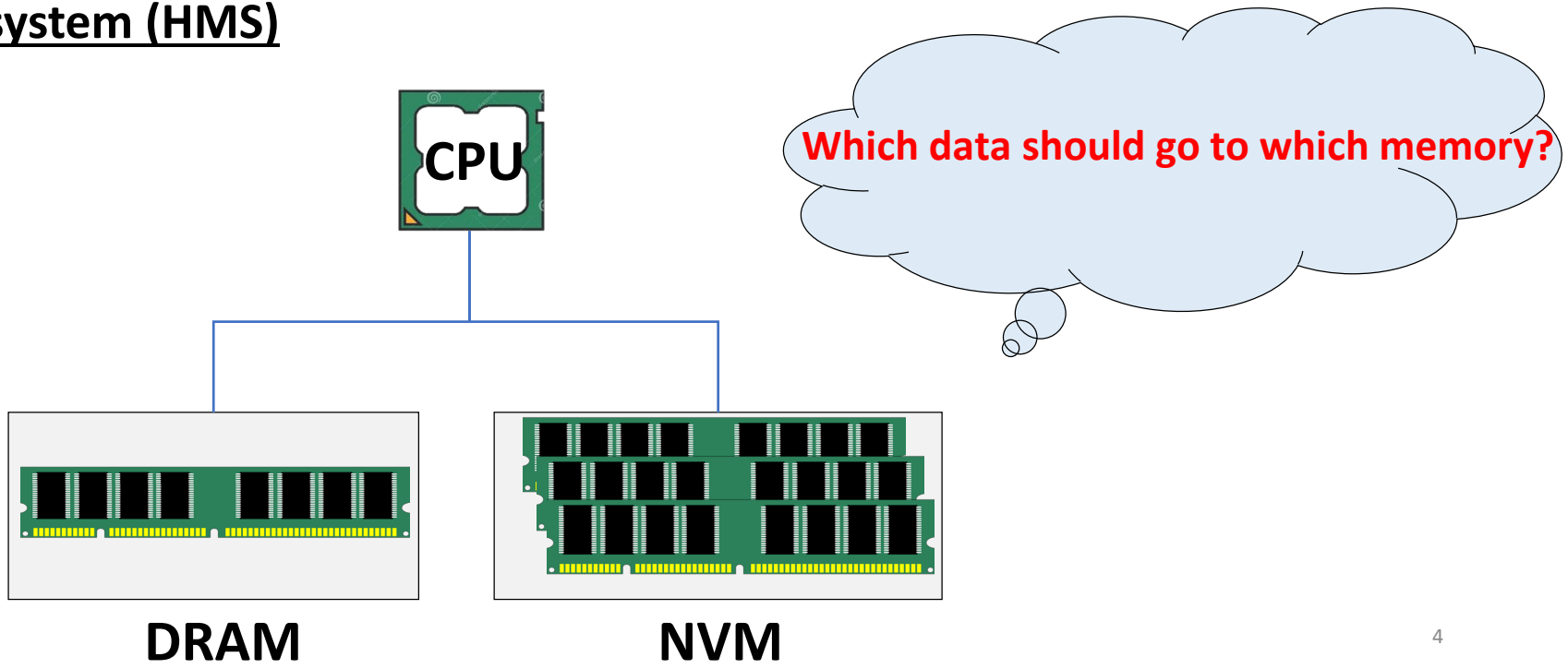
- The existing work already shows the big performance loss, using NVM as main memory [1,2]



[1] S. R. Dulloor, A. Roy, Z. Zhao, N. Sundaram, N. Satish, R. Sankaran, J. Jackson, and K. Schwan, "Data Tiering in Heterogeneous Memory Systems,"EuroSys'16
[2]K. Wu, Y. Huang, and D. Li, "Unimem: Runtime Data Management on Non-volatile Memory-based Heterogeneous Main Memory," SC'17.

NVM-based Heterogenous Main Memory System

- We must pair NVM with DRAM to build a **heterogeneous memory system (HMS)**



Task-parallel Programs

- We target the task-based programming model
  - Particularly, the OmpSs programming model (similar to OpenMP task)
- Tasks are independent code regions that can be executed in parallel
- Programmers express data dependencies between tasks

```
#pragma omp task \
    in(([realN]oldPanel)[1;BS][1;BS] ...) out (...)
void jacobi(long realN, long BS, \
        double newPanel[realN][realN], \
        double oldPanel[realN][realN])
{
    ...
}
```

Research Challenges

- First, how to capture and characterize memory access patterns for each task?
  - Different tasks in a task-parallel program often work on different data (with different memory addresses)

- Second, how to maximize the performance benefit?
  - How to estimate the performance benefit when data of a task is distributed among DRAM and NVM?

- Third, how to minimize the impact of data movement on application performance?

Story in a Nutshell

- Tahoe: a runtime system for task-parallel programs to manage data placement on NVM-based HMS
  - No hardware/application modification
- Characterize memory access information across tasks
  - Profiling memory access pattern of some tasks
  - Predicting the performance of other tasks that have no page sharing with the profiled tasks
- Hybrid performance model to drive data placement decisions
  - Combine machine learning and analytical models
  - Avoid modeling complexity and introduce modeling flexibility

Background Information

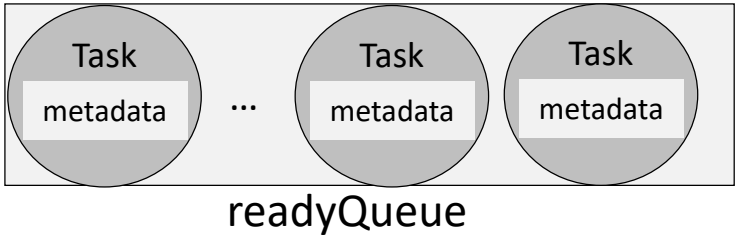
- Task metadata information
  - Task dependence information
  - Task execution state (Initialized, Ready, Active, Completed)
  - Input/output data object information
- Task type
  - Tasks running the same code region with the same input data size have the same task type

```
#pragma omp task \
    in(([realN]oldPanel)[1;BS][1;BS] ...) out (...)
void jacobi(long realN, long BS, \
        double newPanel[realN][realN], \
        double oldPanel[realN][realN]) {
  for (int i=1; i <= BS; i++) {
    for (int j=1; j <= BS; j++) {
      newPanel[i][j] = 0.25 * (oldPanel[i-1][j] \
          + oldPanel[i+1][j] + oldPanel[i][j-1] \
          + oldPanel[i][j+1]);
} } }
```

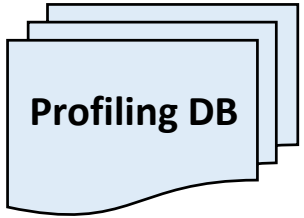
```
void main(){
...
#pragma omp taskwait
for (int iters=0; iters<L; iters++) {
  int currentPanel = (iters + 1) % 2;
  int lastPanel = iters % 2;
  for (long i=BS; i <= N; i+=BS) {
    for (long j=BS; j <= N; j+=BS) {
      jacobi(realN, BS, \
            (m_t) &A[currentPanel][i-1][j-1], \
            (m_t) &A[lastPanel][i-1][j-1] );
} } }
#pragma omp taskwait
...
}
```

Example code from the Heat benchmark

Using Tahoe with Heterogonous Memory System



Task Profiling

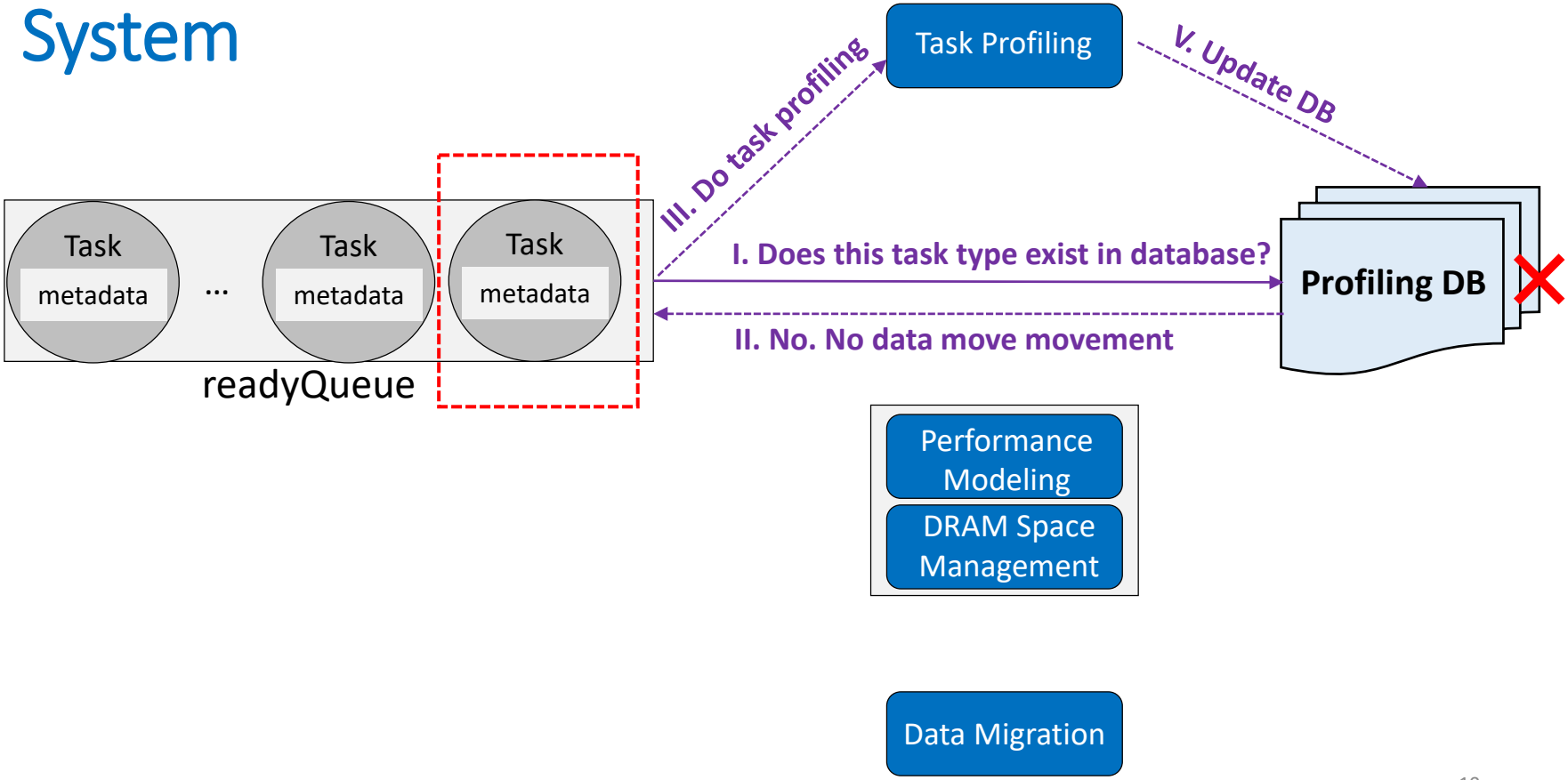


Performance modeling

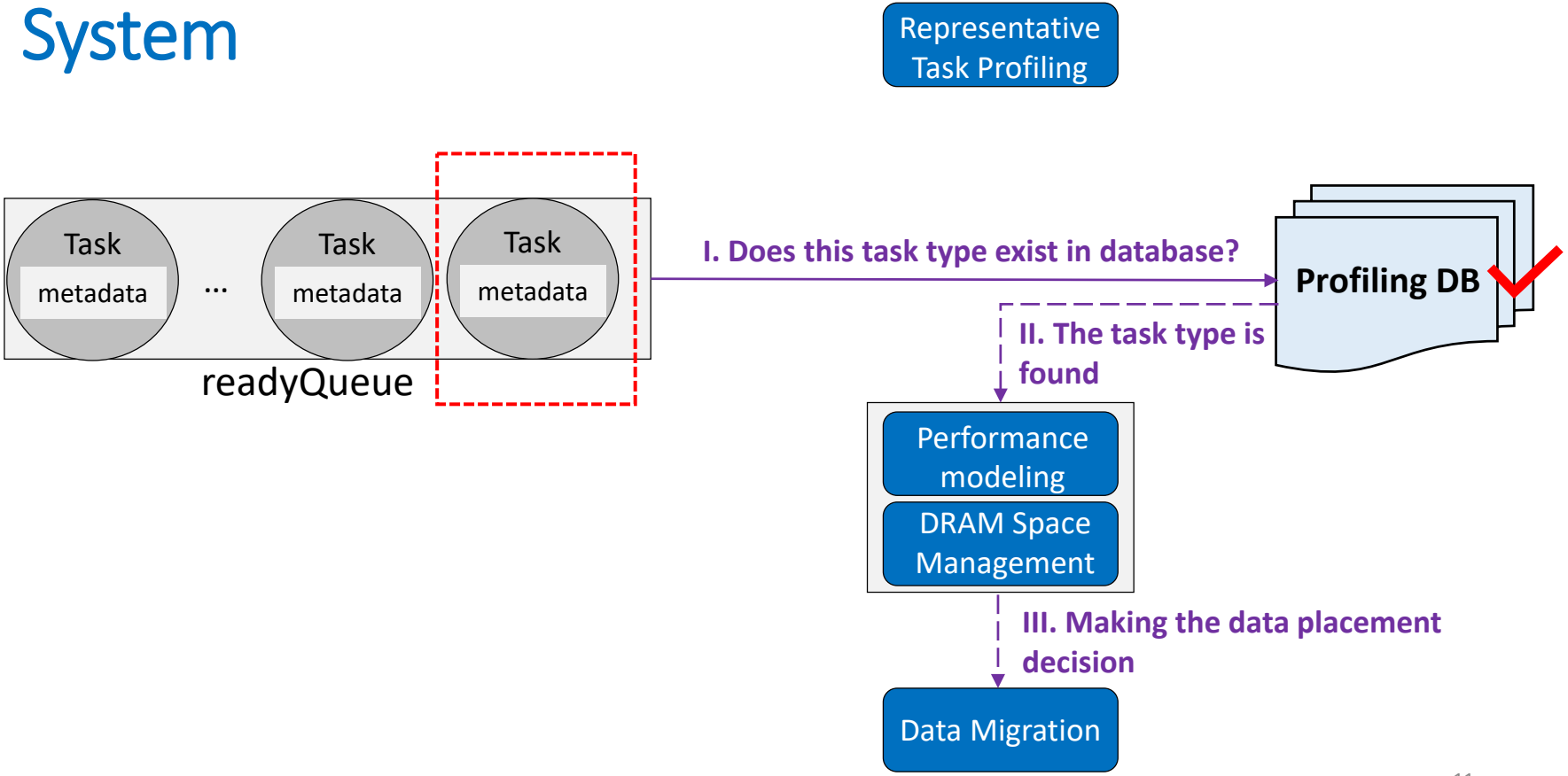
DRAM Space Management

Data Migration

Using Tahoe with Heterogonous Memory System



Using Tahoe with Heterogonous Memory System



Task Profiling

- Our goal: collect main memory access events of the **first instance of each task type** and decide which memory pages to migrate for each task

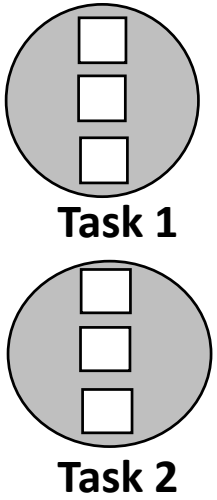
- Memory access events: number of instructions, last-level cache misses and execution time

- Use **sampling-based** hardware performance counters
  - Map the last-level cache miss events to memory pages via memory addresses

Task Mapping

- The memory access information of the profiled task cannot be directly used by other tasks to decide data placement
  - Different tasks use different virtual addresses for their data objects

- Page-level -> Data object level

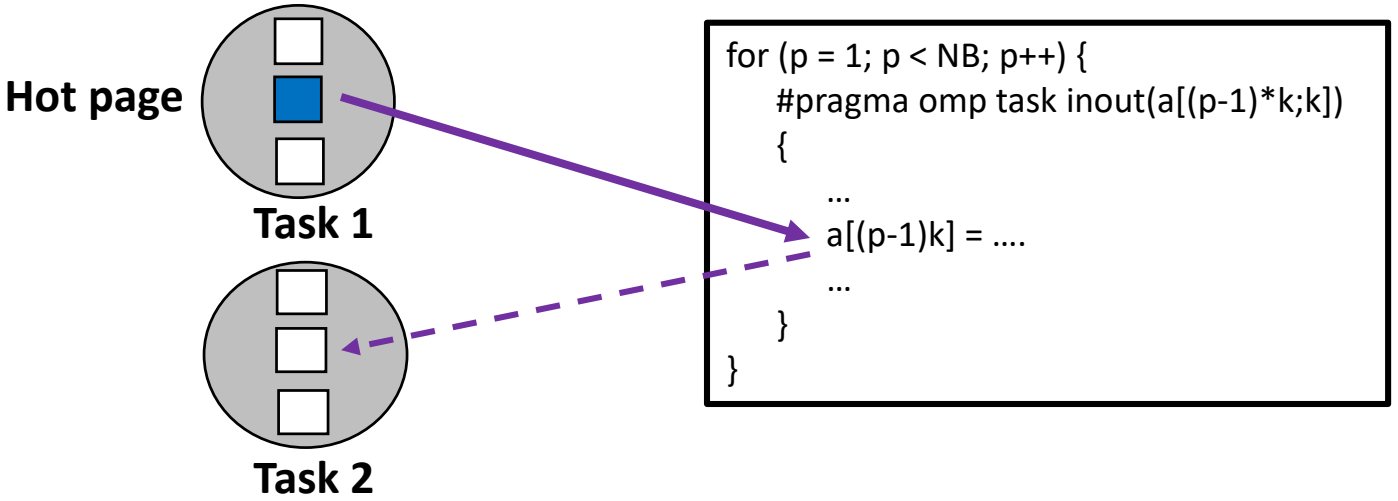



```
for (p = 1; p < NB; p++) {
    #pragma omp task inout(a[(p-1)*k;k])
    {
        …
        a[(p-1)k] = ….
        …
    }
}
```

Task Mapping

- The memory access information of the profiled task cannot be directly used by other tasks to decide data placement
  - Different tasks use different virtual addresses for their data objects

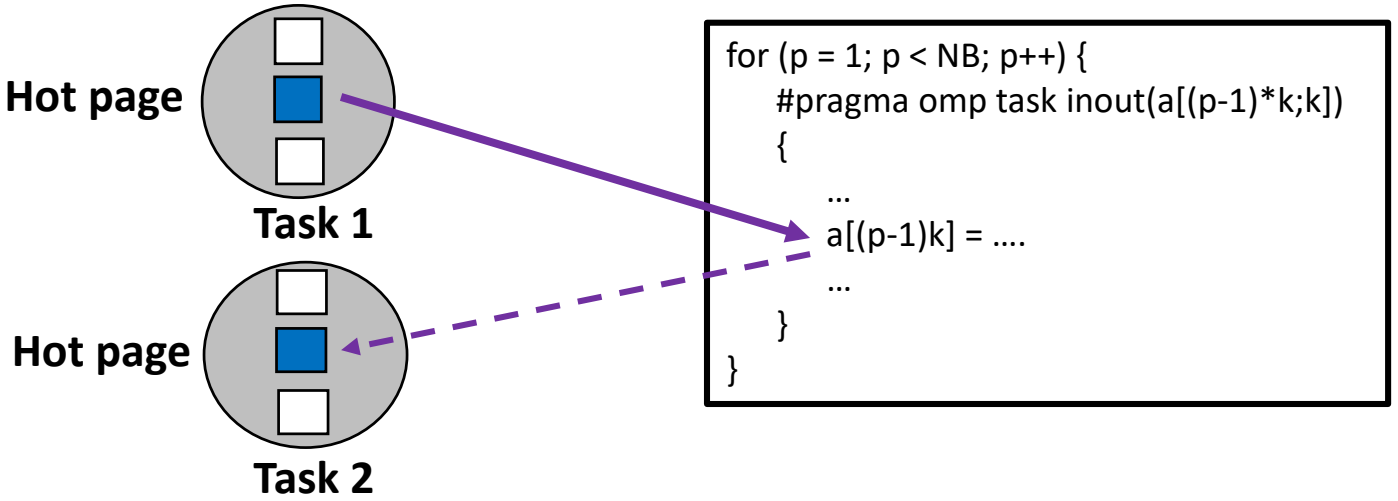
- Page-level -> Data object level



Task Mapping

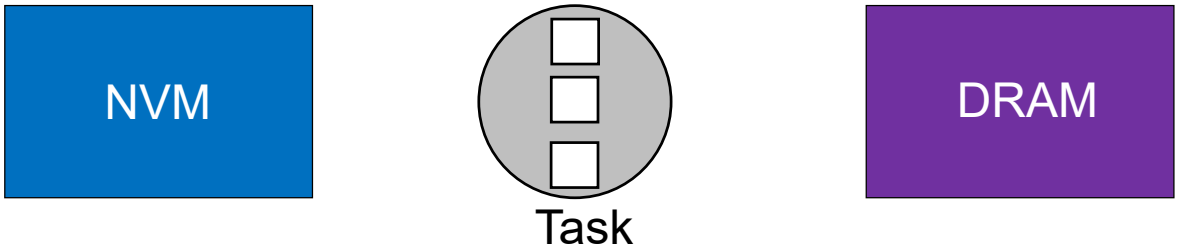
- The memory access information of the profiled task cannot be directly used by other tasks to decide data placement
  - Different tasks use different virtual addresses for their data objects

- Page-level -> Data object level



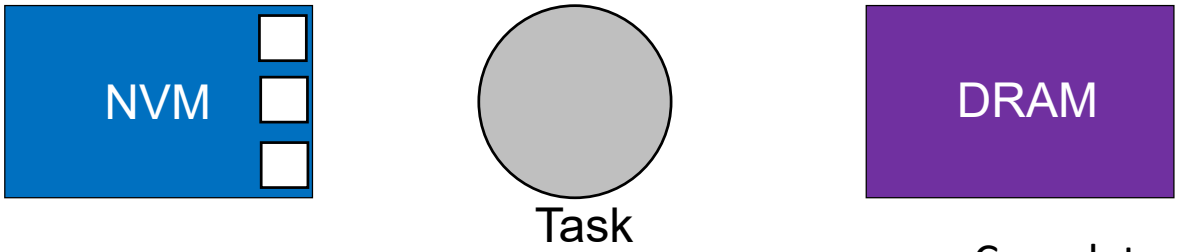
Performance Modeling

- Goal: Decide DRAM space partition between multiple tasks when those tasks are ready to be run by multiple processing elements



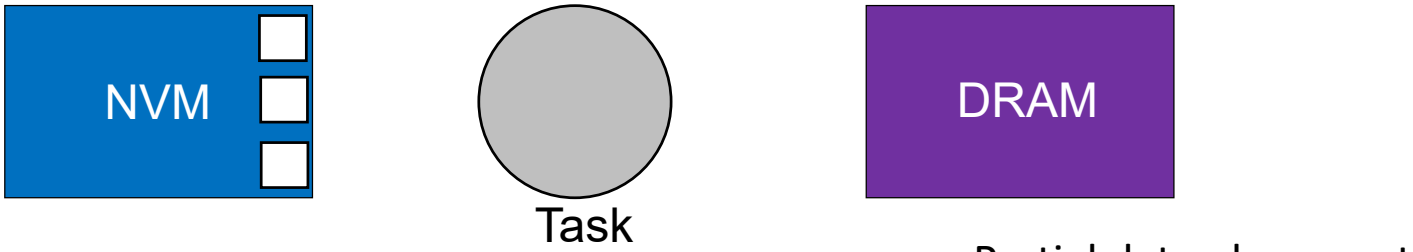
Performance Modeling

- Goal: Decide the DRAM space partition between multiple tasks when those tasks are ready to be run by multiple processing elements



Performance Modeling

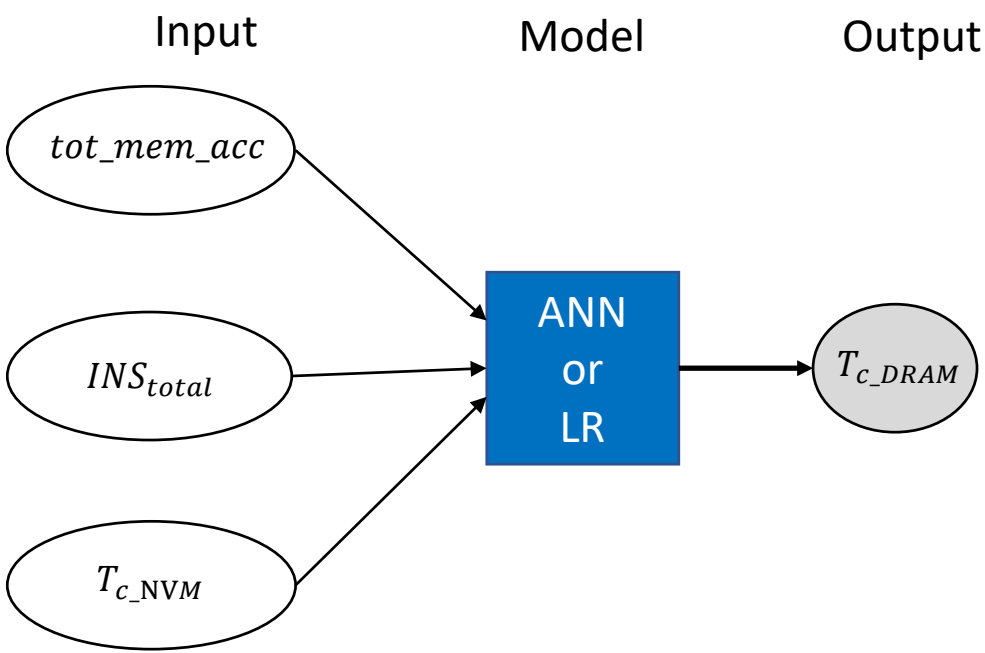
- Goal: Decide the DRAM space partition between multiple tasks when those tasks are ready to be run by multiple processing elements



- Hybrid performance model
  - Machine learning based-model to predict performance for complete data placement
  - Analytical based-model to predict performance for partial data placement

Performance Modeling for Complete Data Placement

- Analytical modeling is hard to capture the sophisticated relationship between execution time and performance events
- Modeling techniques
  - Linear regression analysis (LR)
  - Artificial neural network (ANN)



  - tot_mem_acc: last level cache miss rate
  - INS_{total}: total instruction number
  - T_{c_NVM}: execution time on NVM
  - T_{c_DRAM}: Estimated execution time on DRAM

Performance Modeling for Complete Data Placement

- Prediction accuracy and training time with various memory bandwidth
  - Seven benchmarks from BSC application repository
  - Cross-validation

Model Type	Multiple LR Model			ANN Model		
NVM Bandwidth	1/4	1/8	1/16	1/4	1/8	1/16
Average training time per epoch (s)	25.3	23.5	22.4	32.4	31.7	33.8
Total training time (s)	207.2	191.4	195.0	254.9	249.6	262.3
Average prediction error	10.9%	26.4%	45.9%	3.6%	4.1%	5.1%
Prediction error variance	0.2	57.2	4700	0.007	0.016	0.017

- ANN model performs better (less than 6% prediction error on average)
- Use ANN model in the Tahoe

Performance Modeling for Partial Data Placement

- The machine learning model needs to increase the number of parameters (lacks flexibility)

- Analytical modeling

$$T_p = (T_{\mathrm{c\_NVM}} - T_{c\_DRAM}) \times \frac{p\_\mathrm{nvm}\_acc}{tot\_mem\_acc} + T_{c\_DRAM}$$

  - $\mathrm{T_p}$: execution time with the partial data placement
  - $\mathrm{p\_nvm\_acc}$: number of NVM accesses with partial data placement
  - $\mathrm{tot\_mem\_acc}$: total number of memory accesses with complete data placement

Performance Modeling for Partial Data Placement

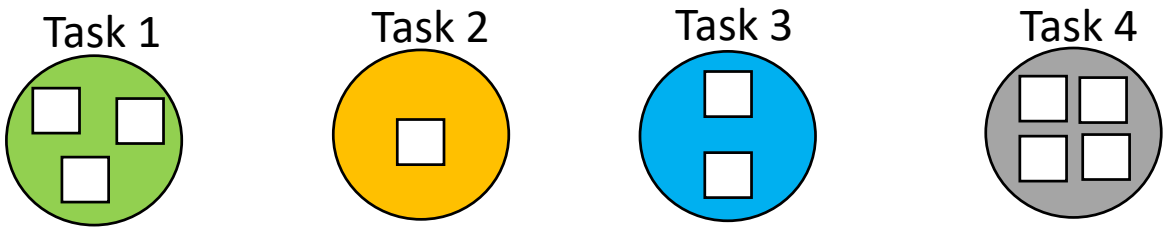
- Performance prediction error
  - Three configurations: (1) NVM-only, (2) memory is allocated using a round robin approach on both NVM and DRAM, and (3) DRAM-only

Benchmarks	FFT	BT	Strassen	CG	Heat	Random Access	SPECFE M3D
p_nvm_acc	$5.7 \times \times 10^7$	$1.9 \times \times 10^8$	$7.7 \times \times 10^6$	$4.3 \times \times 10^7$	$5.2 \times \times 10^7$	$1.0 \times \times 10^8$	$7.4 \times \times 10^7$
tot_mem_acc	$1.2 \times \times 10^8$	$4.1 \times \times 10^8$	$1.6 \times \times 10^7$	$7.4 \times \times 10^7$	$2.2 \times \times 10^8$	$2.7 \times \times 10^8$	1.45×10^8
$\frac{p\_nvm\_acc}{tot\_mem\_acc}$	0.48	0.46	0.48	0.58	0.24	0.37	0.51
Prediction error	6.9%	3.6%	3.0%	1.5%	3.0%	3.0%	6.5%

- The prediction error is less than 7%

Data Migration for Multiple Tasks

Case 1: tasks with different types co-run



How many pages on DRAM?

m_1 m_2 m_3 m_4

Estimated execution time

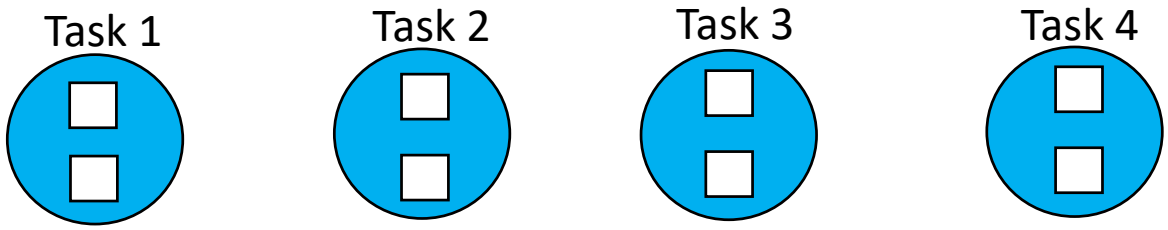
$perf_1$ $perf_2$ $perf_3$ $perf_4$

$$perf = \max perf_i (1 \le i \le k)$$

Dynamic programming !

Data Migration for Multiple Tasks

Case 2: tasks with the same type co-run



Evenly partition the available DRAM space

DRAM Space Management

- Records which memory pages are in DRAM
- Migrate pages from DRAM to NVM when DRAM runs out of space and there is a task pending to be executed
  - LRU policy (Expensive)

DRAM Space Management

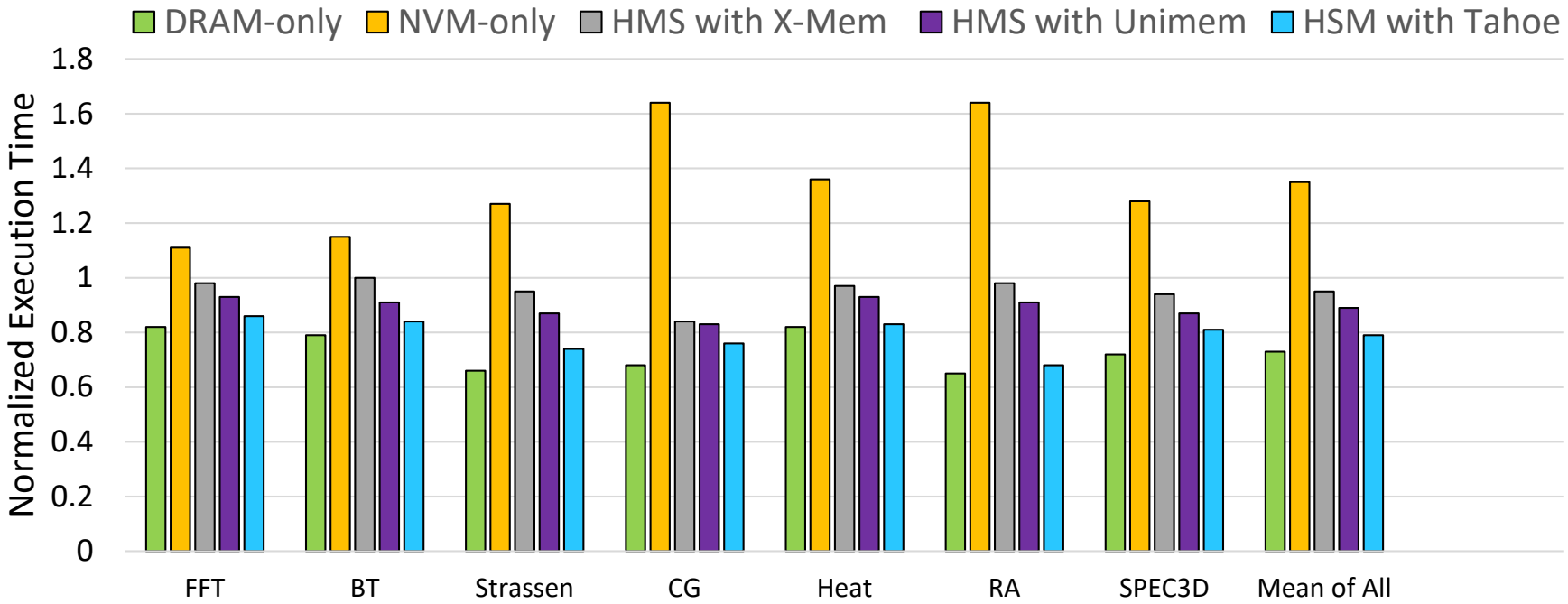
- Records which memory pages are in DRAM

- Migrate pages from DRAM to NVM when DRAM runs out of space and there is a task pending to be executed
  - ~~LRU policy~~ (Expensive)
  - FIFO policy based on tasks execution order

Performance Evaluation

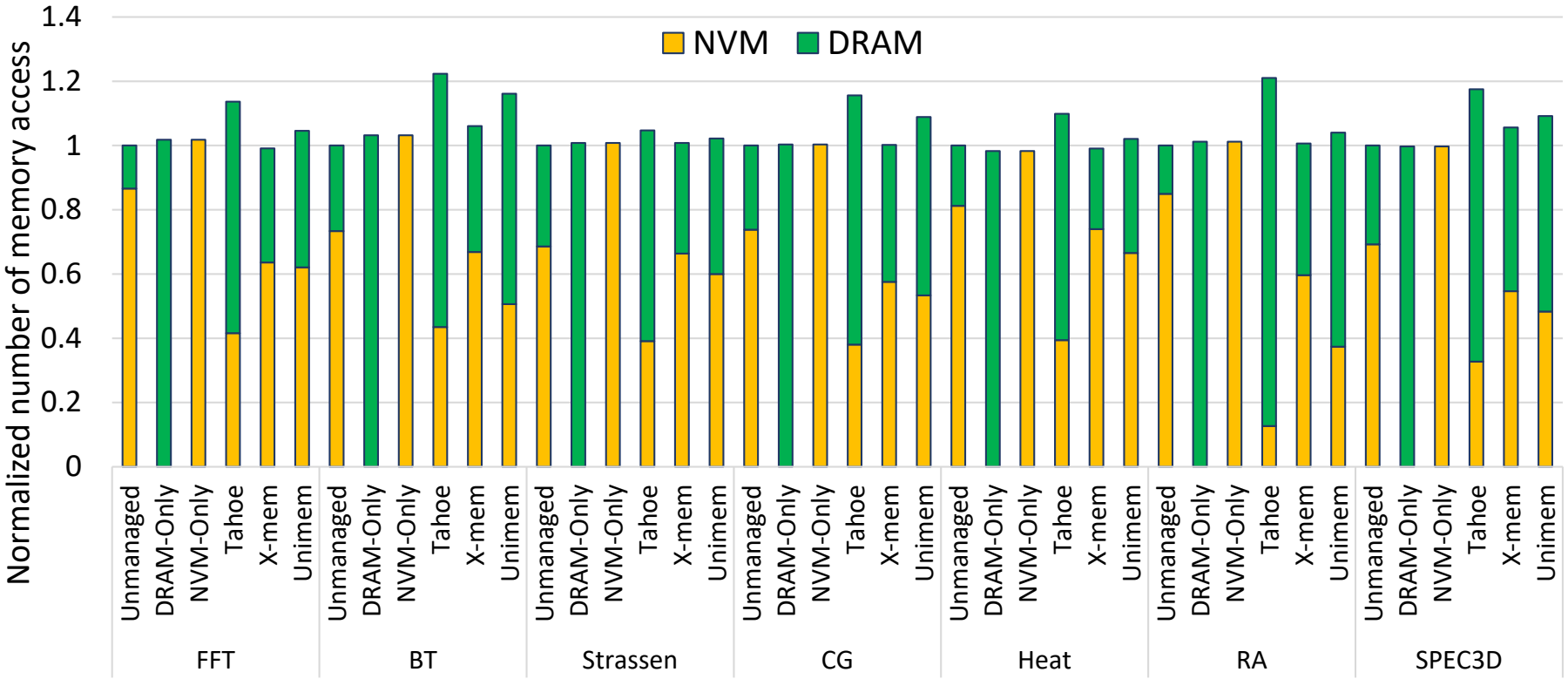
- NVM emulator
  - Quartz(Hewlett Packard): enables the emulation of NVM latency and bandwidth characteristics
- Workloads
  - FFT, BT-MZ, Strassen, CG, Heat, RandomAccess(RA) from BSC application repository
  - SPECFEM3D(SPEC3D)
- Comparisons
  - Existing work:
    - X-Mem (EuroSys'16)
    - Unimem (SC'17)
  - HMS-oblivious (baseline)

Basic Performance Tests with 1/4 DRAM Bandwidth



- X-mem, Unimem and Tahoe reduce execution time by 5%, 11% and 21% on average respectively (using HMS-oblivious as the baseline)
- Tahoe outperforms X-mem and Unimem by 16% and 10% on average

Memory access breakdowns with 1/4 DRAM Bandwidth



- Tahoe has larger numbers of DRAM memory accesses than other systems
  - Make best use of DRAM for performance

Conclusions

- Using runtime of a programming model to direct data placement on heterogenous memory system is promising

- Tahoe is a runtime system for task-parallel programs to manage data placement on NVM-based HMS
  - leverage task metadata and collect the memory access information of limited tasks
  - use a hybrid performance model to make data placement decisions

- Tahoe achieves higher performance than a conventional HMS-oblivious runtime (24% improvement on average) and two state-of-the-art HMS-aware solutions (16% and 11% improvement on average, respectively)

Thank you! Question?