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

<table>
<thead>
<tr>
<th></th>
<th>HDD</th>
<th>SSD</th>
<th>NVM</th>
<th>DRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency</td>
<td>7.1 ms</td>
<td>68 us</td>
<td>2-500 ns</td>
<td>100 ns</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>2.6 MB/s</td>
<td>250 MB/s</td>
<td>5 GB/s</td>
<td>64 GB/s</td>
</tr>
</tbody>
</table>
Non-volatile Memory is Promising

- Fast byte-addressable and persistent NVM technologies are coming
- NVM has good performance but still not enough
- The existing work already shows the big performance loss, using NVM as main memory [1,2]

<table>
<thead>
<tr>
<th></th>
<th>HDD</th>
<th>SSD</th>
<th>NVM</th>
<th>DRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency</td>
<td>7.1 ms</td>
<td>68 us</td>
<td>2-500 ns</td>
<td>100 ns</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>2.6 MB/s</td>
<td>250 MB/s</td>
<td>5 GB/s</td>
<td>64 GB/s</td>
</tr>
</tbody>
</table>

NVM-based Heterogenous Main Memory System

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

Which data should go to which memory?
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

```c
#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

Example code from the Heat benchmark

```c
#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
...
}
Using Tahoe with Heterogeneous Memory System

Task metadata
Task metadata
Task metadata
readyQueue

Task Profiling
Profiling DB
Performance modeling
DRAM Space Management
Data Migration
Using Tahoe with Heterogonous Memory System

I. Does this task type exist in database?

II. No. No data move movement

III. Do task profiling

IV. Update DB

Performance Modeling

DRAM Space Management

Data Migration

profiling DB
Using Tahoe with Heterogenous Memory System

I. Does this task type exist in database?

II. The task type is found

III. Making the data placement decision

Performance modeling

DRAM Space Management

Data Migration

Representative Task Profiling

Profiling DB

readyQueue

Task metadata

Task metadata

Task metadata
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

```c
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

```c
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
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

NVM

Task

DRAM

Complete data placement
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)

  – $\text{tot\_mem\_acc}$: last level cache miss rate
  – $\text{INS}_{\text{total}}$: 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

<table>
<thead>
<tr>
<th>Model Type</th>
<th>Multiple LR Model</th>
<th>ANN Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>NVM Bandwidth</td>
<td>1/4</td>
<td>1/8</td>
</tr>
<tr>
<td></td>
<td>1/4</td>
<td>1/8</td>
</tr>
<tr>
<td>Average training time per epoch (s)</td>
<td>25.3</td>
<td>23.5</td>
</tr>
<tr>
<td></td>
<td>32.4</td>
<td>31.7</td>
</tr>
<tr>
<td>Total training time (s)</td>
<td>207.2</td>
<td>191.4</td>
</tr>
<tr>
<td></td>
<td>254.9</td>
<td>249.6</td>
</tr>
<tr>
<td>Average prediction error</td>
<td>10.9%</td>
<td>26.4%</td>
</tr>
<tr>
<td></td>
<td>3.6%</td>
<td>4.1%</td>
</tr>
<tr>
<td>Prediction error variance</td>
<td>0.2</td>
<td>57.2</td>
</tr>
<tr>
<td></td>
<td>0.007</td>
<td>0.016</td>
</tr>
</tbody>
</table>

• 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

\[ T_p = (T_{c\_NVM} - T_{c\_DRAM}) \times \frac{p\_nvm\_acc}{tot\_mem\_acc} + T_{c\_DRAM} \]

- \( T_p \): execution time with the partial data placement
- \( p\_nvm\_acc \): number of NVM accesses with partial data placement
- \( tot\_mem\_acc \): 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

<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>FFT</th>
<th>BT</th>
<th>Strassen</th>
<th>CG</th>
<th>Heat</th>
<th>Random Access</th>
<th>SPECFE M3D</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_{nvm_acc}$</td>
<td>$5.7 \times 10^7$</td>
<td>$1.9 \times 10^8$</td>
<td>$7.7 \times 10^6$</td>
<td>$4.3 \times 10^7$</td>
<td>$5.2 \times 10^7$</td>
<td>$1.0 \times 10^8$</td>
<td>$7.4 \times 10^7$</td>
</tr>
<tr>
<td>$tot_mem_acc$</td>
<td>$1.2 \times 10^8$</td>
<td>$4.1 \times 10^8$</td>
<td>$1.6 \times 10^7$</td>
<td>$7.4 \times 10^7$</td>
<td>$2.2 \times 10^8$</td>
<td>$2.7 \times 10^8$</td>
<td>$1.45 \times 10^8$</td>
</tr>
<tr>
<td>$p_{nvm_acc}$</td>
<td>$0.48$</td>
<td>$0.46$</td>
<td>$0.48$</td>
<td>$0.58$</td>
<td>$0.24$</td>
<td>$0.37$</td>
<td>$0.51$</td>
</tr>
</tbody>
</table>

| 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

\[ \text{perf} = \max \text{perf}_i (1 \leq i \leq k) \]

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)
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.
- 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?