Search Space Reduction for Parameter Tuning of a Tsunami Simulation on the Intel Knights Landing Processor

Kazuhiko Komatsu, Takumi Kishitani, Masayuki Sato, Akihiro Musa, Hiroaki Kobayashi
Tohoku University
13 September, 2018
MCSoC2018 Special Session ATMG2018
Background

• Various designs of recent computing systems
  – Multi/many-core technology
    • 10~100 cores in a processor
  – Vector processing technology
    • simultaneous calculations of multiple elements
      – The number of elements tends to be increased
  – Hybrid memory architecture
    • High bandwidth memories with the conventional DDR memory
      – MCDRAM&DDR in KNL, HBM in Volta and SX-Aurora TSUBASA

Performance tuning is mandatory to exploit the potential
Motivation

• Performance tuning for complex computing systems
  – A number of system parameters to be tuned due to its complexity
    • Difficult to find the best parameters in a practical time
      – 300+ patterns for all parameter combinations in KNL
  – Long time of each application execution
    • Execution time of an HPC application tends to increase due to more advanced, detailed, precise simulations
    • Each execution time drastically affects tuning times
      – An app is executed many times to find an appropriate parameter sets

A lot of time needs to be spent for performance tuning
Objective and approach

• Objective
  – Reduce the time for performance tuning
    • An appropriate parameter combination is found in a practical time

• Approach
  – Narrow a search space of system parameters
    • Predict a bottleneck
    • Select system parameters to solve the bottleneck
  – Reduce time of each application run
    • Select application parameters
      – Simulation period, input data, and so on
Search space reduction

1. System parameter survey
   – Understand characteristics of system parameters
     • Relationship between each system parameter and performance
     • This information used for system parameter selections in Step 3

2. Bottleneck prediction
   – Identify bottleneck considering both a computing system and an application

3. System parameter selections
   – Select **system parameters that only effective to solve** the predicted bottleneck
   → Narrow search space of system parameter combinations
Step 1: System parameter survey

• Preliminary evaluation of benchmarks with various system parameters
  – **Only once** when a system is installed
  – GEMM, HPL
    • Identify contribution of each system parameter to **computational** performance
  – Stream
    • Identify contribution of each system parameter to **memory bandwidth** performance
  – Any other benchmarks
    • Other information can be used for system parameter selection
Step2: Bottleneck prediction

• Identify bottleneck
  – Bottleneck is utilized to reduce search space of parameter combinations
  – How to identify?
    • This paper uses **Bytes/Flop** ratios of a system and an app
      – Code B/F < System B/F  =>  **Computational bound**
      – Code B/F > System B/F  =>  **Memory bandwidth bound**
        * Code B/F = (necessary data in Byte) / (# floating operations)
        * System B/F = (memory bandwidth) / (peak performance)
Step 3: System parameter selections

• What system parameters should be selected to narrow search space?
  – Select *system parameters that only effective to solve* the predicted bottleneck
    • Comp bound => parameters effective to computation
    • Mem bound => parameters effective to memory
    • Characteristics of system parameters are clarified in Step 1
Reduce time of each application run

- **Step 4: Setting application parameters**
  - Input data
    - Use smaller data to shorten execution time
      - Generally, the input data affects the length of an app execution
      - The characteristics of an application should not be changed
  - Simulation period
    - Make simulation period shorter
      - Until the characteristics of application do not change
      - The simulation period tends to be proportional to execution time
    - How to find an appropriate simulation period?
      - Shorting the period in the binary search manner until the simulation period and the execution time are proportional
  - Other application parameters
Case study: KNL with TSUNAMI

- Intel Xeon Phi 7290 (KNL)
  - 72 cores
    - Up to four threads can be used by SMT
  - Hybrid memory system
    - MCDRAM
      - High bandwidth memory
    - DDR
      - Conventional memory
- Tsunami numerical simulation [Koshimura, et al 2009]
  - Ground fluctuation
  - Tsunami inundation prediction

<table>
<thead>
<tr>
<th>Processor</th>
<th>Intel Xeon Phi 7290B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of cores</td>
<td>72</td>
</tr>
<tr>
<td>Peak performance</td>
<td>6.912 Tflop/s (single precision)</td>
</tr>
<tr>
<td>Memory subsystem</td>
<td>MCDRAM and DDR4-2400</td>
</tr>
<tr>
<td>MCDRAM capacity / BW</td>
<td>16 GB / 446.3 GB/s (stream)</td>
</tr>
<tr>
<td>DDR capacity / BW</td>
<td>96 GB / 115.2 GB/s (peak)</td>
</tr>
</tbody>
</table>
Configurable system parameters (1)

- **Cluster** mode
  - Decides how to logically divides tiles and memory into virtual regions
  - *all-to-all*
    - Not divided into virtual regions
    - UMA
  - *Quadrant, Hemisphere*
    - Divides into 4 or 2 virtual regions
    - UMA
    - Smaller latency than all-to-all
  - *Sub-NUMA Cluster*(SNC)-4, SNC-2
    - Divides into 4 or 2 virtual regions
    - NUMA
      - Nume-aware optimization is necessary
Configurable system parameters (2)

- **Memory** mode
  - Decides how to use MCDRAM and DDR
  - **Flat** mode
    - MCDRAM and DDR have the same address space
      - An application needs to explicitly allocate data in MCDRAM
  - **Cache** mode
    - All MCDRAM is treated as a cache of DDR
      - Cache is hardware-managed, so no explicit programming:
  - **Hybrid** mode
    - Combination of flat mode and cache mode
      - 25% or 50% of MCDRAM is used as cache.
      - Remaining MCDRAM is used as allocatable memory
Configurable system parameters (3)

- **Thread affinity**
  - *Compact*
    - A thread is assigned to a core as close as possible to its adjacent thread
    - Suitable for **computation-intensive** applications
  - *Scatter, balanced*
    - Threads are distributed across all cores as much as possible
    - Balanced affinity assigns a close thread to the same core
    - Suitable for **memory-intensive** applications

- **Number of threads**
  - Up to 4 threads can be assigned to one core
    - 72, 144, 216, 288 are candidates to use full cores in KNL

The number of combinations of system parameters reaches 300
Overview of TSUNAMI simulation

• TSUNAMI numerical simulation
  – Mainly consists on ground fluctuation and Tsunami inundation prediction

• Application parameters
  – Input data
    • Regional data to predict tsunami
  – Simulation period
    • How many time steps should be simulated?
Step 1: System parameter survey of KNL

- Stream benchmark with various system parameters
  - Less than 16 GB memory requirement to check MCDRAM performance
- Other benchmarks
  - stream (> 16GB)
  - GEMM
Step 2: Bottleneck prediction of KNL and TSUNAMI

- This paper predicts the bottleneck using Bytes/Flop ratios
  - System B/F of KNL
    - (memory BW) / (peak flops) = 446.3 GB/s / 6912 Gflop/s = 0.065
  - Code B/F of TSUNAMI
    - (necessary data) / (# flops) ≒ 1.85
→ Code B/F > System B/F
  - The TSUNAMI simulation is predicted as a Memory bandwidth bound application
Step3: System parameter selections of KNL

- Parameter selections to solve the predicted memory bottleneck
  - **Cluster mode**
    - *Quadrant* for short memory accesses
    - 5 candidates are reduced to just 1
  - **Memory mode**
    - *flat* mode to exploit MCDRAM bandwidth
    - 5 candidates to just 1 candidate
  - **Thread affinity and # of threads**
    - Balanced/scatter with 72, 144, 216, 288
    - Compat with 288
    - 12 candidates to 9 candidates
Step 4: Application parameter setting

• Limiting the simulation period
  – The shorter simulation period discovered by the binary search
  • Until the short period does not change the characteristics of application runs

By applying all reduction, 77.6 hours of full search can be reduced to 670 seconds.
Conclusion

• Toward reduction in performance tuning time
  – As the number of both system and application parameters increases, the long time is necessary for tuning

• Approach
  – Narrow a search space of system parameters
    • Predict a bottleneck and select appropriate system parameters
  – Reduce time of each application run
    • Select application parameters
    → 180 parameter combinations can be reduced to just 9
      • 77.6 hours for full search to 670 seconds by our approach

• Future work
  – More detailed evaluations
    • Not only the full search but also the conventional tuning algorithms
    • Other application such as computation intensive