Performance Balancing: Software-based On-chip Memory Management for Effective CMP Executions

Naoto Fukumoto, Kenichi Imazato, Koji Inoue, and Kazuaki Murakami

Kyushu University, Japan
Outline

• Limitation factor of CMP performance
• Performance balancing for effective CMP executions
• Evaluation
• Summary
Chip MultiProcessors

- CMPs (Chip MultiProcessors)
  - Have several processor cores integrated in a chip
  - Achieve high performance by parallel processing
  - Are widely used from server to embedded systems

Cell Broadband Engine
(Sony/Toshiba/IBM)

Core i7 (Intel)


http://ascii.jp/elem/000/000/124/124281/img.html

(scale is not correct)
Target CMP Model

Parallel program

Chip

ScratchPad Memory: Software managed on-chip memory

On-chip interconnect

Main memory
What is Limitation Factor of CMP Performance?

• “Memory Wall Problem”: Total performance is limited by low memory performance
  – Speed gap between processor and main memory
  – Will be getting more critical in CMPs

Program: Barnes

Program: Cholesky

High scalability

Memory limits the total performance
Outline

- Limitation factor of CMP performance
- Performance balancing for effective CMP executions
- Evaluation
- Summary
Motivation of Performance Balancing

• Conventional execution to exploit all cores is not always effective
  – From 6 to 8 cores, performance is slightly improved
• Improving memory performance may largely improve total performance

Aim to improve total performance, we use the part of cores for memory performance improvement
Performance Balancing
~ Suitably Utilize the Processor Cores ~

Define two kinds of core

• Helper cores
  – Do not execute threads
  – Provide SPMs to main core
  – Hold data for main cores

• Main cores
  – Execute parallelized threads
  – Quickly obtain the data by on-chip data transfer to exploit helper cores’ SPM
Example of Advantage from Helper Cores

- If helper cores exist
  - # of main memory accesses = 1
  - # of on-chip data transfers = 2

- Helper cores replace main memory accesses to high-speed on-chip data transfers

Example code

```c
func() {
    DMAGET(dataA);
    DMAGET(dataB);
    DMAGET(dataC);
    ...
}
```
Importance of the Number of Helper Cores

• \((\text{#of main cores} + \text{#of helper cores}) = \text{constant}\)

• Increase \#of helper cores
  – Improves memory performance
  – Degrades computation performance

Need to find the appropriate number of helper cores
Changing the Number of Helper Cores According to Application Characteristic

Program: Barnes

- Ideal memory
- 256KB private memory

Program: Cholesky

- Ideal memory
- 256KB private memory

Performance improvement by executing parallelized threads by all cores

Large Improvement of memory performance
Small degradation of computing performance
1. Perform pre-execution only one time with one main core to obtain profile information
2. Decide data mapping to helper cores’ SPM and estimate the best number of helper cores
3. Generate the objective source code supporting performance balancing
How to Decide the Data Mapping to Helper Cores’ SPM

- Objective: Find the best data mapping that minimizes # of accesses to main memory by main cores
- Count # of accesses for every address interval
- Choose the data in decreasing order of the count
How to Predict the Best #of Helper Cores

- Objective: Find the best #of helper cores that maximizes the total performance
- Predict the performance for all combinations
- Find the best #of helper cores

Pre execution with 1 main cores

Performance modeling

Find the best # of helper cores

- Fraction of exe. time parallelized
- #of on-chip data transfers ...

The predicted # of helper cores
Outline

• Limitation factor of CMP performance
• Performance balancing for effective CMP executions
• Evaluation
• Summary
Experimental setup

• Hardware platform
  – Cell Broadband Engine
  – # of cores: seven

• Benchmark programs
  – Scientific domain: Himeno benchmark, LU (SPLASH-2), FFT (Cell SDK), Matrix multiplication (Cell SDK)
  – Embedded domain: Susan (MiBench)
  – Use the same input data for profiling and evaluation

• Evaluation models
  – CONV: Conventional approach (execute by 7 cores)
  – PB-PREDICT: Performance Balancing (PB) with the predicted number of helper cores
  – PB-IDEAL: Performance Balancing (PB) with the best number of helper cores
Results of Evaluation

6 main cores and 1 helper core
## Results of Evaluation

<table>
<thead>
<tr>
<th>Helper Cores</th>
<th>SS</th>
<th>SSS</th>
<th>HIMENO</th>
<th>SUSAN</th>
<th>FFT</th>
<th>LU</th>
<th>MATRIX</th>
<th>MUL</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Helper Cores</td>
<td>4:3</td>
<td>5:2</td>
<td>7:0</td>
<td>0</td>
<td>1</td>
<td>512</td>
<td>256</td>
<td></td>
</tr>
<tr>
<td># of on-chip transfers</td>
<td>80%</td>
<td>60%</td>
<td>40%</td>
<td>20%</td>
<td>0%</td>
<td>100%</td>
<td>100%</td>
<td></td>
</tr>
<tr>
<td># of DMA transfers</td>
<td>60%</td>
<td>80%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td></td>
</tr>
</tbody>
</table>
Results of Evaluation

![Graph showing performance results for different benchmarks.

- **CONV**: Light blue bars.
- **PB-PREDICT**: Red bars.
- **PB-IDEAL**: Green bars.

**Benchmarks**:
- **HIMENO**
- **SUSAN**
- **FFT**
- **LU**
- **MATRIX MUL**

**Execution Time**:
- **Optimal** configuration.
- **Predicted** configuration.

**Speed Up**:
- **SS**
- **SSS**
- **HIMENO**
- **SUSAN**
- **FFT**
- **LU**
- **MATRIX MUL**

**Note**: The graph illustrates the performance comparison between different execution configurations for various benchmarks, highlighting the speedup and execution time for each configuration.
Results of Evaluation

Our technique improves performance to make a good balance between comp. and mem. performance.
Outline

• Limitation factor of CMP performance
• Performance balancing for effective CMP executions
• Evaluation
• Summary
Summary and Future Work

• *Performance balancing* for effective CMP executions
  – Remedies the memory-wall problem by reducing the bandwidth pressure and #of main memory accesses
  – Achieves an appropriate balance between computation and memory performance to change #of helper cores
  – Significantly improves performance to improve memory performance at the expense of computing performance

• Future work
  – More detailed evaluation (e.g. energy consumption, parameter sensitivity)
  – Improving data mapping of helper cores
Thank you

Any Questions?

~Please speak slowly~