Mimer and Schedeval: Tools for Comparing Static Schedulers for Streaming Applications on Manycore Architectures

Nicolas Melot  Johan Janzén  Christoph Kessler

¹Linköping University
Dept. of Computer and Inf. Science
Linköping, Sweden

November 27, 2015
Outline

1. Introduction
   - Streaming
   - Evaluation

2. Mimer evaluation framework

3. Schedeval streaming framework
   - Description
   - Testing overhead
   - Evaluating computation time
   - Studying energy consumption

4. Mimer & Schedeval environment
   - Data structures
   - Programming
   - Experiment

5. Conclusion
   - Conclusion
   - Future work
Introduction

High-Performance Computing on Streams

- Optimize for time and energy
- Scheduling Problems

Numerous publications

- On-chip Pipelining ([Keller et al. [2012]])
- Scheduling sequential tasks ([Pruhs et al. [2008]])
- Crown Scheduling ([Melot et al. [2015]])
Streaming

**Straightforward**: communication through shared off-chip main memory

- Easy to implement
- Main memory bandwidth is performance bottleneck

**On-chip pipelining**: communication through small on-chip memories ([Keller et al. [2012]]).

- Trade core to core communication for reduced off-chip memory accesses
Optimize for time and energy

Scheduling for performance

- Compromise between time and energy
- Many features to take into account
  - Voltage and frequency
    - Cores grouped in islands
    - Impact dependent on stalls due to memory accesses
  - Off-chip memory accesses
  - On-chip communications
  - Static/dynamic power
- Behavior differences between execution platforms

\[
P_{dyn} \approx V \cdot f^2
\approx f^3
\]

\[
E_{dyn} \approx P_{dyn} \cdot p \cdot t
\approx f^3 \cdot p \cdot t
\]
Testing Scheduling techniques

Difficulty to test scheduling techniques

- Simulator: imperfect, slow
- Real platform: high development efforts

Difficulty to compare between techniques

- Access to experimental datasets
  - Tailor-made for a paper (Xu et al. [2012])
  - Lack of adaptability (Kasahara [2004])
- Access to raw results
- Access to result processing and representation tools

Mimer: Testing and interpreting framework, raw and structured data
Schedeval: run a streaming application on real architectures
Flexible formats

- GraphML (XML) Task graphs
- XML Schedules
- AMPL-base platforms
- Flexible C++ backend API

Open-data

- Keep raw data
- Publication of data processing scripts (R)
- Structure data in CSV (Comma-Separated Values)
Schedeval: run a streaming application on a real architecture

- Co-developed with Janzén [2014] as Master thesis work.
- Integrates as a schedule evaluator in Mimer workflow
Testing the Schedeval framework

Framework overhead: ping-pong application

Vary the number of tasks
- With a single pair
- With several pairs

Vary the mapping
- Mapped to several cores
- Mapped to several tiles
- Mapped to a single core

Monitor
- Average ping round trip time
- Proportion of non data-ready task instances scheduled
Schedeval overhead

- Roundtrip time penalized by distance and overhead
- Latency hidden with increasing ping-pong pairs
- Decreasing rate of non data-ready task scheduled
Schedeval overhead

- Roundtrip time penalized by distance and overhead
- Latency hidden with increasing ping-pong pairs
- Decreasing rate of non data-ready task scheduled
Evaluating the Schedeval framework

Computation time: mergesort

Schedules over 6 cores *(Melot et al. [2012])*

- Mapped per level (simple)
- Mapped per block (reduce communications)

Test schedule with a unique core

Test a variant with no frequency scaling mechanism
Schedeval computation time

Results described by Janzén [2014]

Vary implementations

- Depth-first task execution
- Distribute presort phases

Observations

- Low performance difference in streaming phase
- High performance difference for initial sort
Studying energy consumption

Usefulness for energy consumption studies

- Use a fixed application: StreamIt implementation of FFT (Thies et al. [2002])
- Run with 11 different schedules
- Vary deadline tightness to constrain frequency
- Monitor Energy consumption
- Compare with simple energy models

\[ P_{dyn} \approx V \cdot f^2 \]
\[ E_{dyn} \approx P \cdot p \cdot t \]
\[ \approx f^3 \cdot p \cdot t \]
Measure energy with Schedeval

Usefulness for energy consumption studies

Analytic evaluation
- All schedules are equivalent
- No voltage islands taken into account

Observations through Mimer / Schedeval
- Analytical model fails to capture architectural details
- More fast tasks mapped to more voltage islands
Under the hood
Mimer & Schedeval Data structures: Taskgraph

...<node id="n30">
  <data key="v_name">t31</data>
  <data key="v_module">merge</data>
  <data key="v_workload">1</data>
  <data key="v_max_width">1</data>
  <data key="v_efficiency">exprtk: 1 / p</data>
</node>
<edge source="n30" target="n14"/>
<edge source="n29" target="n14"/>
...
Mimer & Schedeval Data structures: Platform

# Number of cores

\[ \text{param } p := 4; \]

# Frequency levels

\[ \text{set } F := 100000 \ 106000 \ 114000 \ 123000 \ 133000 \ 145000 \ 160000 \ 178000 \ 200000 \ 228000 \ 266000 \ 320000 \ 400000 \ 533000 \ 800000; \]
Mimer & Schedeval Data structures: Schedule

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<schedule name="Generated from Algebra"
    appname="Generated from Algebra">
    <core coreid="1">
        <task name="t1" start="0" frequency="8e5"
            width="1" workload="32" />
    </core>
    <core coreid="2">
        <task name="t2" start="0" frequency="8e5"
            width="1" workload="16" />
    </core>
    ... 
    </core>
    <core coreid="3"> ... </core>
    <core coreid="4"> ... </core>
</schedule>
```
Schedeval refactored: Drake

```c
#include <drake/node.h>
#include <drake/link.h>
#include <drake/platform.h>

int drake_init(task_t *task, void* aux) {
    return 1;
}

int drake_start(task_t *task) {
    return 1;
}

int drake_run(task_t *task) {
    return 1;
}

int drake_destroy(task_t *task) {
    return 0;
}
```
### Mimer experiment

<table>
<thead>
<tr>
<th>platform</th>
<th><code>scc_8.dat</code></th>
<th><code>scc_32.dat</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>taskgraph</td>
<td><code>fft2_streamit_loose</code></td>
<td><code>fft2_streamit_tight</code></td>
</tr>
<tr>
<td>evaluation</td>
<td><code>static_dynamic_busy</code></td>
<td><code>drake_eval_scc_emulator</code></td>
</tr>
<tr>
<td>scheduler</td>
<td><code>crown_ltlg_height</code></td>
<td><code>crown_binary_ltlg_height</code></td>
</tr>
<tr>
<td>analysis</td>
<td><code>parco2015-nmelot</code></td>
<td></td>
</tr>
</tbody>
</table>
Demo
Investigate more

Scheduling by minimizing the number of core to use and switch off unused cores.

All you need is a new scheduler implementation

Already done!
Presented at PARCO 2015, Edinburgh, on Thursday, September 3rd.
Conclusion

Mimer & Schedeval to facilitate scheduling experimentation and data publication

**Mimer**
- Simplifies the process through automation
- Allows the publication of raw and structured results as well as how to process them into figures in articles.

**Schedeval (refactored into Drake)**
- Schedeval performance scales well with number of tasks
- Shows schedule differences in time and energy

http://www.ida.liu.se/labs/pelab/mimer/
Future work

Future work: lots of missing features

- Task microbenchmarking
- Improve platform description
- Implement more Schedeval applications
- Investigate core-to-core communications
- Support for SDF
- Support for stream reconfiguration
- Port to Xeon, Tilera, MPPA

Questions & answers

http://www.ida.liu.se/labs/pelab/mimer/
www.liu.se


