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

Nicolas Melot Johan Janzén Christoph Kessler

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

November 27, 2015

Outline

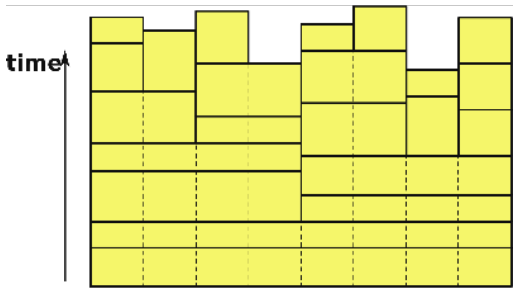
Introduction

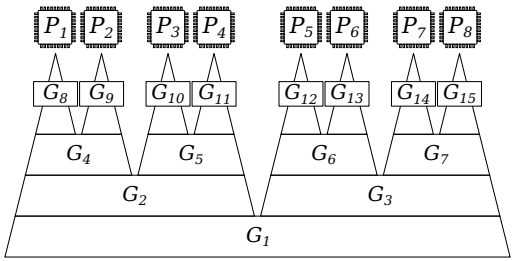
High-Performance Computing on Streams

- 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])

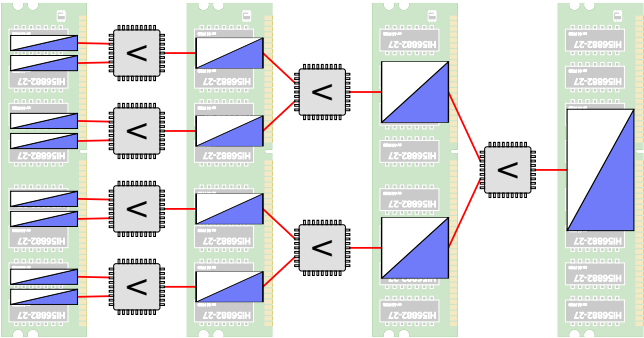




Streaming

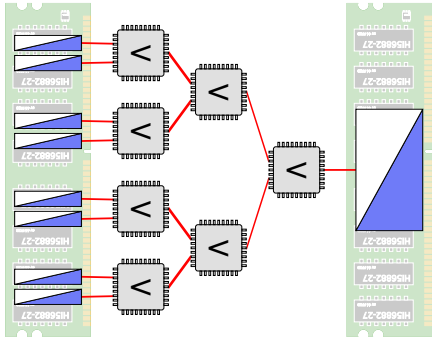
Straightforward: communication through shared off-chip main memory

- 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

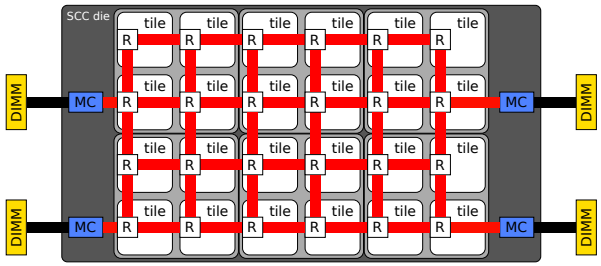


Optimize for time and energy

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

$$
\begin{aligned}
P_{dyn} &\approx V \cdot f^2 \\
&\approx f^3 \\
E_{dyn} &\approx P_{dyn} \cdot p \cdot t \\
&\approx f^3 \cdot p \cdot t
\end{aligned}
$$



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

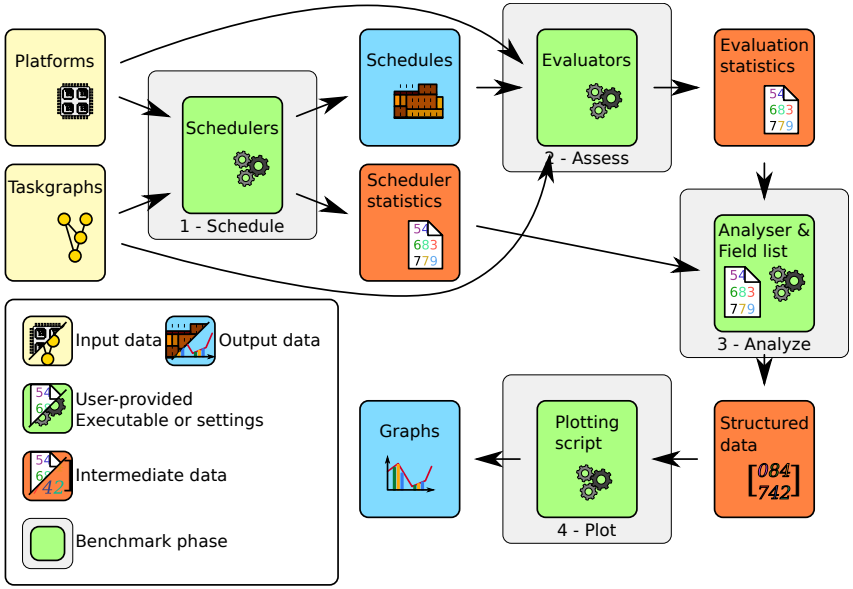
Mimer

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)



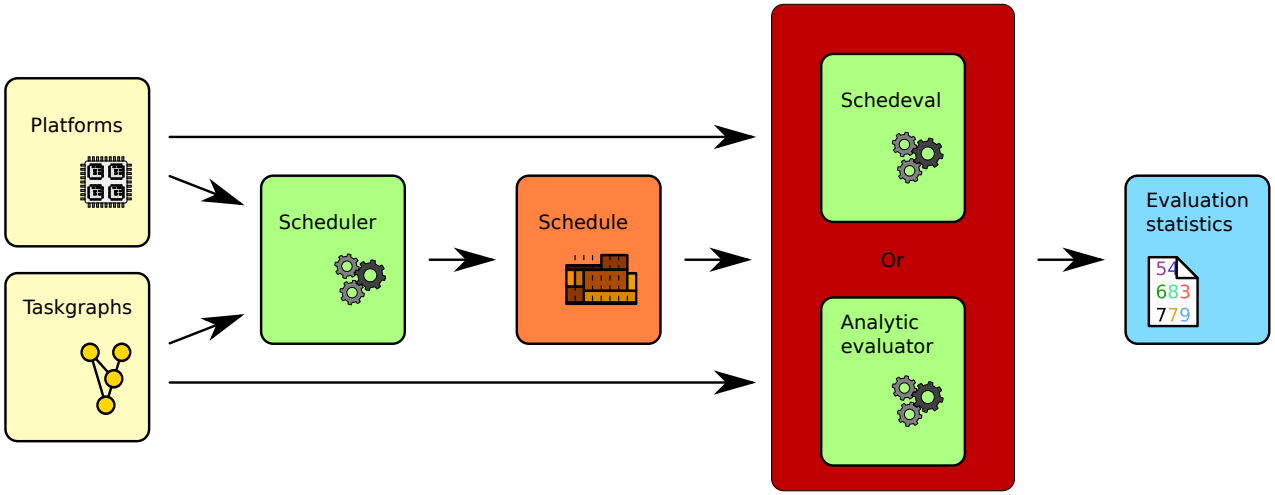
4 steps

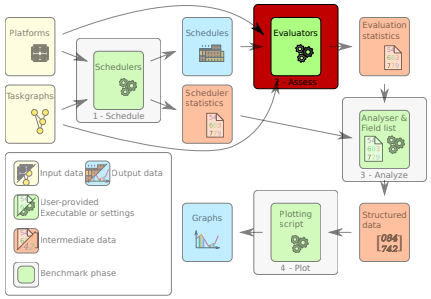
- Schedule
- Evaluate
- Analyze
- Plot

Schedeval

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

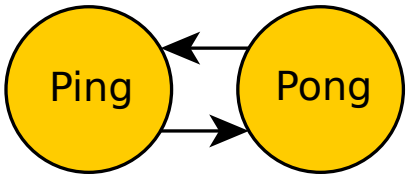
- With a single pair
- With several pairs

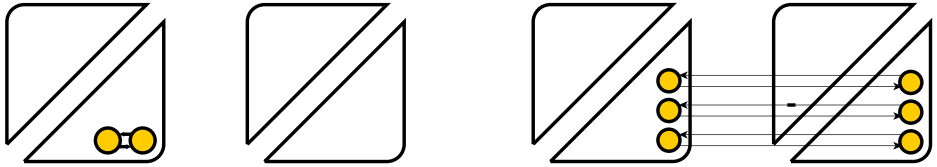
Vary the mapping

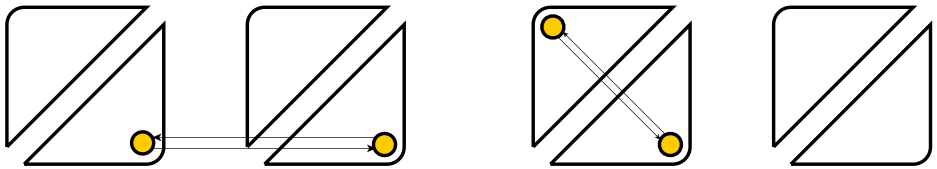
- Mapped to several cores
- Mapped to several tiles
- Mapped to a single core

Monitor

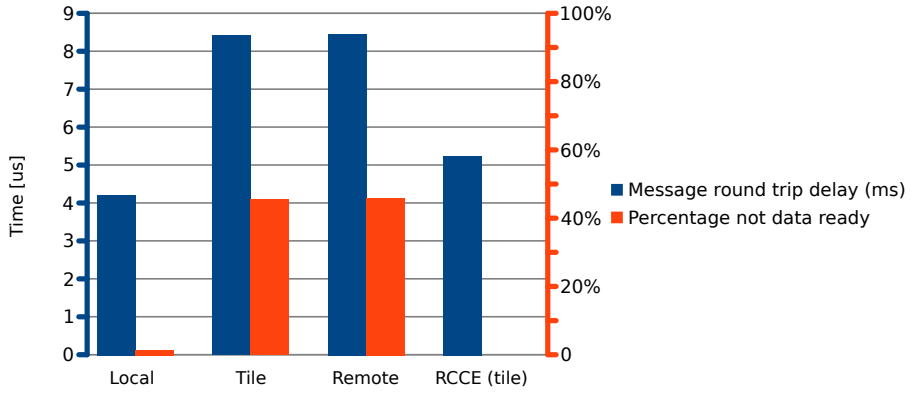
- Average ping round trip time
- Proportion of non data-ready task instances scheduled

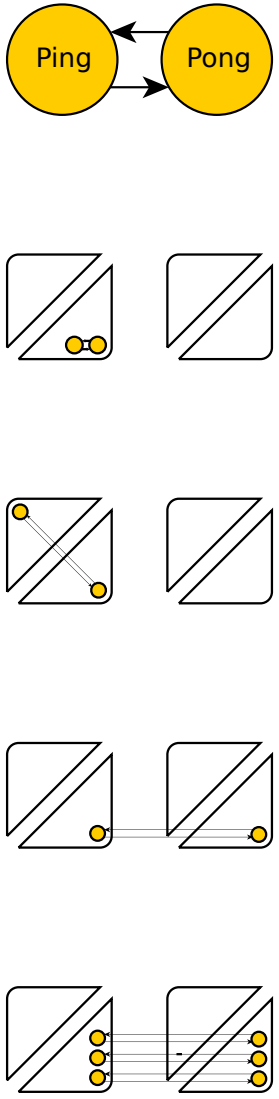






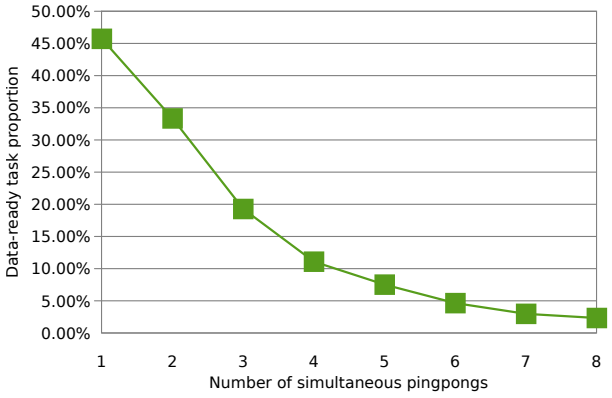
Schedeval overhead

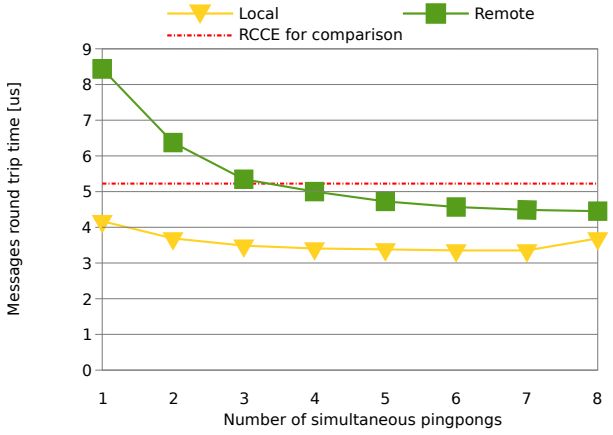


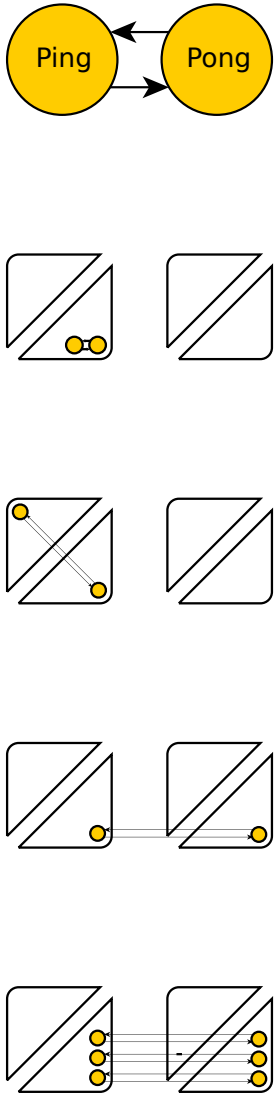


- Roundtrip time penalized by distance and overhead
- Latency hidden with increasing ping-pong pairs
- Decreasing rate of non data-ready task scheduled

Schedeval overhead







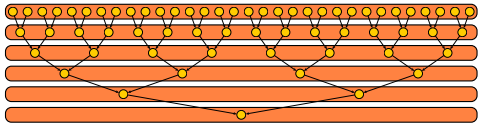
- Roundtrip time penalized by distance and overhead
- Latency hidden with increasing ping-pong pairs
- Decreasing rate of non data-ready task scheduled

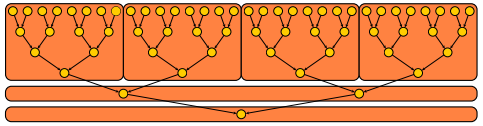
Evaluating the Schedeval framework

Computation time: mergesort

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

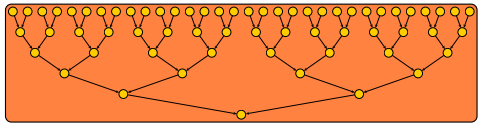
- Mapped per level (simple)
- Mapped per block
  (reduce communications)





Test schedule with a unique core

Test a variant with no frequency scaling mechanism

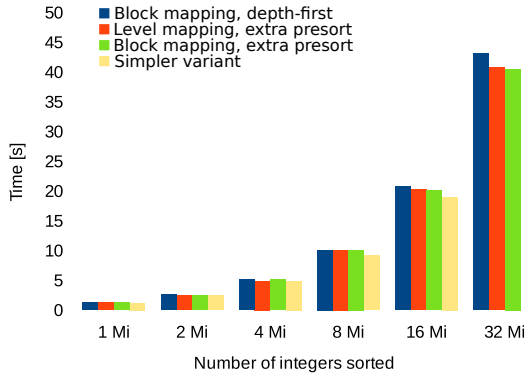


Schedeval computation time

Results described by Janzén [2014]

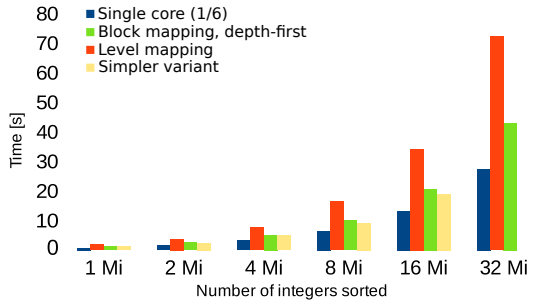
Vary implementations

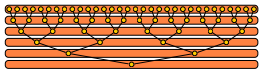
- Depth-first task execution
- Distribute presort phases

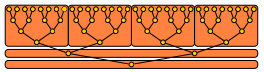


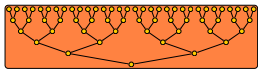
Observations

- Low performance difference in streaming phase
- High performance difference for initial sort







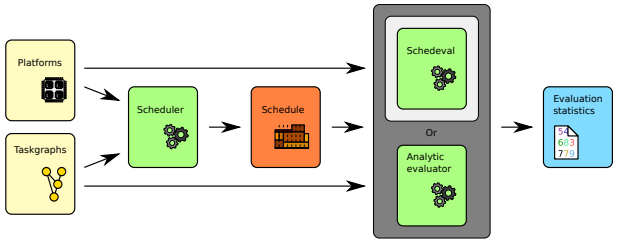


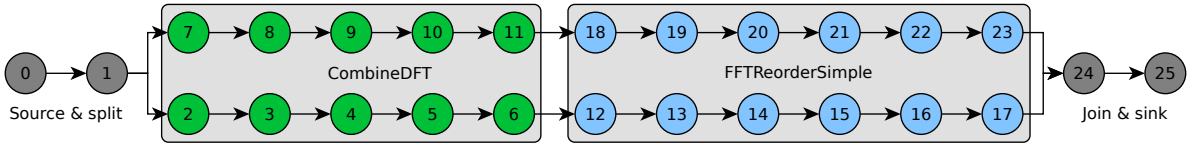
Studying energy consumption

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

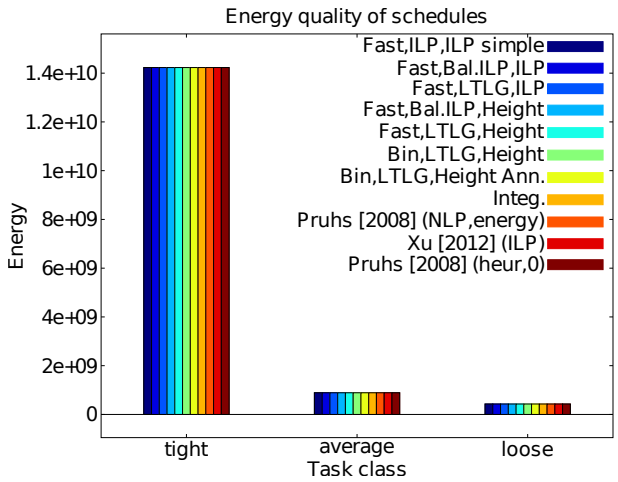
$$
\begin{aligned}
P_{dyn} &\approx V \cdot f^2 \\
&\approx f^3 \\
E_{dyn} &\approx P \cdot p \cdot t \\
&\approx f^3 \cdot p \cdot t
\end{aligned}
$$

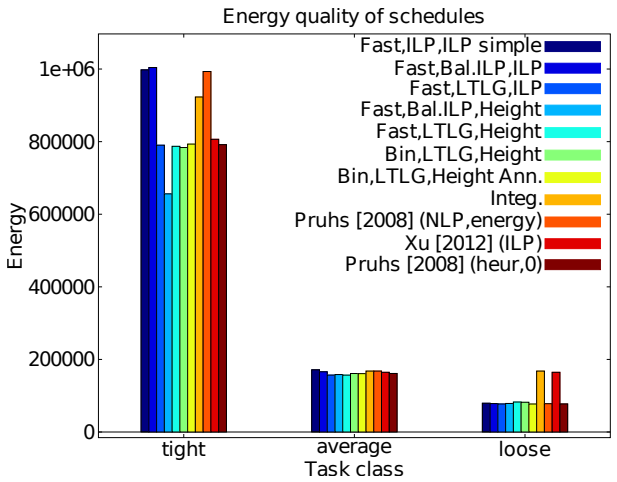


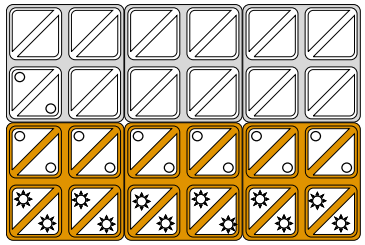


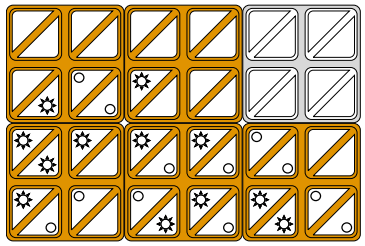
Measure energy with Schedeval

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
param p := 4;

# Frequency levels
set F := 100000 106000 114000 123000 133000\
  145000 160000 178000 200000 228000 266000\
  320000 400000 533000 800000 ;
```

Mimer & Schedeval Data structures: Schedule

```
<?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 coreid="3"> ... </core>
 <core coreid="4"> ... </core>
</schedule>
```

Schedeval refactored: Drake

```
#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

```
platform="scc_8.dat scc_32.dat"
taskgraph="fft2_streamit_loose \
fft2_streamit_tight"
evaluation="static_dynamic_busy \
drake_eval_scc_emulator"
scheduler="crown_ltlg_height \
crown_binary_ltlg_height"
analysis="parco2015-nmelot"
```

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

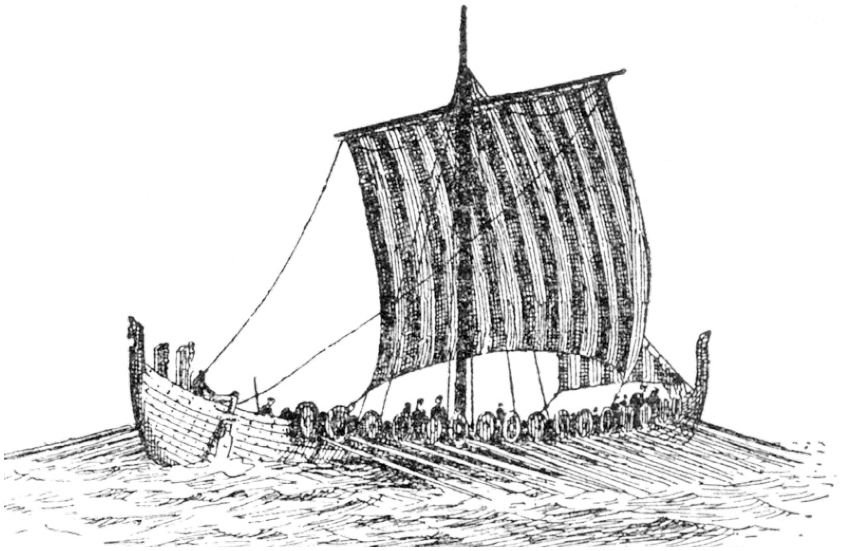
http://www.ida.liu.se/labs/pelab/mimer/

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

Bibliography

Johan Janzén. Evaluation of energy-optimizing scheduling algorithms for streaming computations on massively parallel multicore architectures. Master's thesis, Linköping University, 2014. URL http://liu.diva-portal.org/smash/record.jsf?pid=diva2%3A756758.

Kasahara. Standard task graph set, 2004. URL http://www.kasahara.elec.waseda.ac.jp/schedule/index.html.

Jörg Keller, Christoph Kessler, and Rikard Hulten. Optimized on-chip-pipelining for memory-intensive computations on multi-core processors with explicit memory hierarchy. *Journal of Universal Computer Science*, 18(14):1987–2023, 2012.

Nicolas Melot, Christoph Kessler, Kenan Avdic, Patrick Cichowski, and Jörg Keller. Engineering parallel sorting for the intel SCC. *Procedia Computer Science*, 9(0):1890 – 1899, 2012. doi: http://dx.doi.org/10.1016/j.procs.2012.04.207. Proc. of the Int. Conf. Computational Science, ICCS 2012.

Nicolas Melot, Christoph Kessler, Jörg Keller, and Patrick Eitschberger. Fast Crown Scheduling Heuristics for Energy-Efficient Mapping and Scaling of Moldable Streaming Tasks on Manycore Systems. *ACM Trans. Archit. Code Optim.*, 11(4): 62:1–62:24, January 2015. ISSN 1544-3566.

Kirk Pruhs, Rob van Stee, and Patchrawat Uthaisombut. Speed Scaling of Tasks with Precedence Constraints. *Theory of Computing Systems*, 43(1):67–80, July 2008.

William Thies, Michal Karczmarek, and Saman Amarasinghe. Streamit: A language for streaming applications. In *Compiler Construction*, volume 2304 of *Lecture Notes in Computer Science*, pages 179–196. Springer Berlin Heidelberg, 2002. doi: 10.1007/3-540-45937-5_14.

Huiting Xu, Fanxin Kong, and Qingxu Deng. Energy minimizing for parallel real-time tasks based on level-packing. In *18th Int. Conf. on Emb. and Real-Time Comput. Syst. and Appl. (RTCSA)*, pages 98–103, Aug 2012.