Scheduling DAGs of Multi-version Multi-phase Tasks on Heterogeneous Real-time Systems

Julius Roeder, Benjamin Rouxel and Clemens Grelck

UNIVERSITY OF AMSTERDAM

This work is supported and partly funded by the European Union Horizon-2020 research and innovation programme under grant agreement No. 779882 (TeamPlay).

December 22, 2021
Motivation

Industry Survey\(^1\):

- By which year did you begin using multi-core embedded processors?
- By which year did you begin using heterogeneous multi-cores with different types of CPUs, GPUs, and other accelerators?

Industry Survey$^1$:

- By which year did you begin using multi-core embedded processors?
- By which year did you begin using heterogeneous multi-cores with different types of CPUs, GPUs, and other accelerators?

What kind of boards?

Odroid-XU4 by Hardkernel.

Jetson Nano by Nvidia.
High-performance Embedded Systems

- Heterogeneous CPU & GPU
- Binary Incompatible
- Unclear which target is best for which task
  → Multi-version tasks
Accelerator Workload

1. ...
2. cl_command_queue queue = get_queue();
3. cl_context context = get_context();
4. ...
5. cl_mem buffera = clCreateBuffer(context ...);
6. ...
7. cl_kernel kernel = clCreateKernel(program, "mult_matrix_gpu", NULL);
8. clSetKernelArg(kernel, 0, sizeof(int), (void*) &num_rows);
9. ...
10. cl_event event = NULL;
11. clEnqueueNDRangeKernel(queue, kernel, 2, ..., &event);
12. clWaitForEvents(1, &event);
13. clEnqueueReadBuffer(queue, bufferout, CL_TRUE, 0, sizeof(matrix_int_t)
14.   , (void*) &out_matrix, 0, NULL, NULL);
15. clReleaseMemObject(buffera);
...
State of the art

Reserve CPU&GPU
Wastes resources
Task model for multi-version multi-phase dependent tasks
Heterogeneous FLS (hFLS) heuristic
Task Model

- Directed Acyclic Graph
- Multiple versions & multiple phases
- Communication costs in task timing information
Scheduling Technique

hFLS Heuristic

- Off-line
- Partitioned
- Static
- Time-triggered
- Fixed preemption points
### Forward List Scheduling (FLS)

1. Sort the tasks (4 different ways)
2. While there are tasks
3. For all task versions
4. For all compute units
5. Shift task forward until no overlap or incomparability
6. Select best Version & Core combination
Scheduling Heuristic

**eFLS**

1. Sort the tasks (4 different ways)
2. While there are tasks
3. For all task versions
4. For all phases
5. For all compute units
6. Shift task forward until no overlap or incomparability
7. Select best Version & **Best Phase-Core combination**
BECAUSE CACHES
Cache related delays: migration

- GPU
- CPU Y
- CPU X

- T0-V0-P1
- T0-V0-P2
- T0-V0-P3

Another task

Cache load

- Shared cache lines

- Cache Related Preemption Delay (CRPD)
Cache related delays: entanglement

Cache Related Preemption Delay (CRPD)

J. Roeder, B. Rouxel, and C. Grelck
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sort the tasks (4 different ways)</td>
</tr>
<tr>
<td>2</td>
<td>While there are tasks</td>
</tr>
<tr>
<td>3</td>
<td>For all task versions</td>
</tr>
<tr>
<td>4</td>
<td>For all phases</td>
</tr>
<tr>
<td>5</td>
<td>For all compute units</td>
</tr>
<tr>
<td>6</td>
<td>Add CRPD to WCET</td>
</tr>
<tr>
<td>7</td>
<td>Shift task forward until no overlap or incomparability</td>
</tr>
<tr>
<td>8</td>
<td>Propagate interferences</td>
</tr>
<tr>
<td>9</td>
<td>Select best Version &amp; Best Phase-Core combination</td>
</tr>
</tbody>
</table>
FLS: Multi-version

Task 0
Task 3
Task 6
Task 1
Task 4
Task 7
Task 8
FLS: Multi-version

Task 0
Task 5
Task 4
Task 3
Task 2
Task 7
Task 6
Task 8
Task 1

L - 3
L - 2
B - 1
B - 0 T0-V0

Time
FLS: Multi-version

Task 0
Task 5
Task 4
Task 3
Task 2
Task 7
Task 6
Task 8
Task 1

L - 3
L - 2
B - 1 T0-V0
B - 0

Time
FLS: Multi-version

Task 0
Task 5
Task 4
Task 3
Task 2
Task 7
Task 6
Task 8
Task 1

L - 3
L - 2
B - 1
B - 0

t0 - V0

Time
FLS: Multi-version

Task 0
Task 5
Task 4
Task 3
Task 2
Task 7
Task 6
Task 8
Task 1

L - 3
L - 2
B - 1
B - 0

TO: V0

Time
FLS: Multi-version

Task 0
Task 5
Task 4
Task 3
Task 2
Task 7
Task 6
Task 8
Task 1

L - 3
L - 2
B - 1
B - 0

Time

Task 0 to V1
FLS: Multi-version

Task 0
Task 5
Task 4
Task 3
Task 2
Task 7
Task 6
Task 8
Task 1

L - 3
L - 2
B - 1
B - 0

Time

Task 0 is highlighted with a red cross, indicating a failure or error state. The diagram shows a timeline with tasks and their dependencies, indicating a multi-version scheduling approach.
FLS: Multi-version

Task 0
Task 5
Task 4
Task 3
Task 2
Task 7
Task 6
Task 8
Task 1

L - 3
L - 2
T0-V1
B - 1
B - 0

Time
FLS: Multi-version

Task 0
Task 5
Task 4
Task 3
Task 2
Task 7
Task 6
Task 8
Task 1

L - 3
L - 2
B - 1
B - 0
T0-V1

Time
FLS: Multi-version

Task 0
Task 5
Task 4
Task 3
Task 2
Task 7
Task 6
Task 8
Task 1

L - 3
L - 2
B - 1
B - 0
T0-V0

Time
FLS: Multi-version

Task 0
Task 5
Task 4
Task 3
Task 2
Task 7
Task 6
Task 8
Task 1

L - 3
L - 2
B - 1
B - 0
T0-V0
T5-V0
T7-V0
T8-V0
T6-V0
T2-V1
T3-V1
T4-V0

Time
Adding Phases
Adding a GPU
FLS: Multi-version & Multi-Phase

Task 0
Task 5
Task 4
Task 3
Task 2
Task 7
Task 6
Task 8
Task 1

L - 3
L - 2
B - 1
B - 0
T0-V0
T5-V0
GPU

Time
FLS: Multi-version & Multi-Phase

Task 0
Task 5
Task 4
Task 3
Task 2
Task 7
Task 6
Task 8
Task 1

L - 3
L - 2
B - 1
B - 0
T0-V0
T5-V1

Time
FLS: Multi-version & Multi-Phase

Task 0
Task 5
Task 4
Task 3
Task 2
Task 7
Task 6
Task 8
Task 1

L - 3
L - 2
B - 1
B - 0
T0-V0
T5-V2-P0
GPU

Time
FLS: Multi-version & Multi-Phase

Task 0
Task 5
Task 4
Task 3
Task 2
Task 7
Task 6
Task 8
Task 1

L - 3
L - 2
B - 1
B - 0
T0-V0
T5-V2-P0
GPU
T5-V2-P1

Time
FLS: Multi-version & Multi-Phase

Task 0
Task 5
Task 4
Task 3
Task 2
Task 7
Task 6
Task 8
Task 1

L - 3
L - 2
B - 1
B - 0
T0-V0 T5-V2-P0
GPU
T5-V2-P1
T5-V2-P2

Time
FLS: Multi-version & Multi-Phase

Task 0
Task 5
Task 4
Task 3
Task 2
Task 7
Task 6
Task 8
Task 1

L - 3
L - 2
B - 1
B - 0
GPU

T4-V2-P0
T0-V0 T5-V2-P0
T5-V2-P1
T5-V2-P2

Time
FLS: Multi-version & Multi-Phase

Task 0
Task 5
Task 4
Task 3
Task 2
Task 7
Task 6
Task 8
Task 1

L - 3
L - 2
B - 1
B - 0
T0-V0
T5-V2-P0
T5-V2-P2
GPU
T5-V2-P1
T4-V2-P1

Time
FLS: Multi-version & Multi-Phase

Task 0
Task 5
Task 4
Task 3
Task 2
Task 7
Task 6
Task 8
Task 1

L - 3
L - 2
B - 1
B - 0
GPU

T0-V0
T5-V2-P0
T4-V2-P0
T5-V2-P1
T4-V2-P1
T3-V1
T6-V1
T4-V2-P2
T2-V0
T5-V2-P2
T7-V0

Time
FLS: Multi-version & Multi-Phase

Task 0
Task 5
Task 4
Task 3
Task 2
Task 7
Task 6
Task 8
Task 1

L - 3
L - 2
B - 1
B - 0
GPU

T0-V0  T5-V2-P0  T8-V0
T5-V2-P2  T7-V0
T5-V2-P1
T4-V2-P1

T3-V1  T6-V1
T4-V2-P2  T2-V0

Time
FLS: Multi-version & Multi-Phase

Task 0
Task 5
Task 4
Task 3
Task 2
Task 7
Task 6
Task 8
Task 1

L - 3
L - 2
B - 1
B - 0
T0-V0
T5-V2-P0
T8-V0
T5-V2-P2
+CRPD
T7-V0
GPU
T5-V2-P1
T4-V2-P1

T3-V1
T6-V1
T4-V2-P2
T2-V0

Time
Experiments

Research Questions:

- How do our hFLS generated solutions compare against the optimal solutions?
- How does our hFLS approach compare against a state-of-the-art scheduling heuristics?
- Which task sorting algorithm performs the best?
Setup

- 1,000 task graphs (TGFF CODES/CASHE'1998)
- 1 quad-core CPU & 1 GPU
Experiment 1: hFLS vs. Optimal Solution

- hFLS scales well
- ILP solver does not scale well

J. Roeder, B. Rouxel, and C. Grelck
Experiment 1: hFLS vs. Optimal Solution

\[
\text{utilisation range} = \frac{\sum_{t \in \tau} (\max(WCET_t))}{D}
\]

- Completion time = 100; Deadline 10; Utilisation = 10
Experiment 1: hFLS vs. Optimal Solution

Schedulability
- ILP slightly better for higher utilisation range
- hFLS: 2.4% lower schedulability rate
Setup
- 10,000 task graphs (TGFF CODES/CASHE'1998)
- Heterogeneous octa-core CPU and GPU

Comparison heuristics
- eFLS [Roeder et al. 2021]
- HEFT [Topcuoglu et al. 2002] adjusted for multi-version
Experiment 2: hFLS vs. eFLS vs. HEFT

- hFLS significantly higher schedulability rates
- High utilisation: 11% higher schedulability rate
Experiment 3: hFLS Sorting

FLS: No sorting strategy consistently the best

- HEFT ranking best in 73.6%
- BFS best in 25.6%
- DFS best in 0.8% and only 2.6% better makespan
- DFS leads to more entanglement and higher CRPD costs
Experiment 3: hFLS Sorting

FLS: No sorting strategy consistently the best

<table>
<thead>
<tr>
<th>hFLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>HEFT ranking best in 73.6%</td>
</tr>
<tr>
<td>BFS best in 25.6%</td>
</tr>
<tr>
<td>DFS best in 0.8% and only 2.6% better makespan</td>
</tr>
<tr>
<td>DFS leads to more entanglement and higher CRPD costs</td>
</tr>
</tbody>
</table>
Conclusion

hFLS Fact sheet
- hFLS achieves higher utilisation for heterogeneous systems
- hFLS outperforms eFLS by up to 24%
- hFLS outperforms HEFT by up to 11%
- Performs well in comparison to ILP (2.4% degradation)
- Less sorting algorithms required

Future Work
- Combine hFLS and eFLS