On the Scalability of Time-predictable Chip-Multiprocessing

Wolfgang Puffitsch
wolfgang.puffitsch@onera.fr

Martin Schoeberl
masca@imm.dtu.dk

JTRES’12, October 25, 2012
Motivation

- Growing complexity of real-time systems
- Growing performance demands
- Performance vs predictability
- Time-predictable computer architecture
  - Simple cores
  - Performance through thread-level parallelism
  - Does it scale?
Java Optimized Processor

- Directly executes Java bytecode
- Designed for time-predictability
  - Low-level part of WCET analysis easy
- Timing of bytecodes composable
- Specialized memory hierarchy
  - Stack cache
  - Method cache
  - Further caching on CMPs necessary
  - Predictable memory arbitration
JOP Chip Multi-Processor

JOP core 0
- pipeline
- memory interface
- Split $

JOP core 1
- pipeline
- memory interface
- Split $

JOP core N−1
- pipeline
- memory interface
- Split $

arbiter

FPGA

shared SRAM
Memory Arbitration

- Time-division multiple access arbitration
  - Every core has a slot of $N$ cycles
  - Slot reserved even if core does not access
- Round-robin arbitration
  - Slot reduced to 1 cycle if core does not access
  - Reducing to 0 cycles too costly in hardware
- Round-robin better average case
- TDMA better for worst-case
Arbitration Examples

- Arbitration under contention

![Diagram showing arbitration under contention]
Arbitration Examples

- TDMA behavior independent of other cores
- Some accesses known not to be worst case

```
Core 0
Core 1
Core 2
Core 3
Memory
```
Arbitration Examples

- Round-robin anomaly
- Every access might be worst case

![Diagram showing arbitration examples]

Core 0

Core 1

Core 2

Core 3

Memory

rd0A

nop

rd0B

rd1

rd2

rd3

rd0A nop rd0B
Split Caches I

- Memory areas have different behavior
- Use cache that suits WCET analysis
- JVM allows distinction

<table>
<thead>
<tr>
<th>Area</th>
<th>Addresses</th>
<th>Spatial Locality</th>
<th>Coherence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods</td>
<td>Partially known</td>
<td>High</td>
<td>No</td>
</tr>
<tr>
<td>Stack</td>
<td>Pattern</td>
<td>High</td>
<td>No</td>
</tr>
<tr>
<td>Constants</td>
<td>Known</td>
<td>Low</td>
<td>No</td>
</tr>
<tr>
<td>Type dependent</td>
<td>Partially known</td>
<td>Low</td>
<td>No</td>
</tr>
<tr>
<td>Statics</td>
<td>Known</td>
<td>Low</td>
<td>Yes</td>
</tr>
<tr>
<td>Heap (objects)</td>
<td>Unknown</td>
<td>Low</td>
<td>Yes</td>
</tr>
<tr>
<td>Heap (headers)</td>
<td>Unknown</td>
<td>Medium</td>
<td>Partial</td>
</tr>
<tr>
<td>Heap (arrays)</td>
<td>Unknown</td>
<td>High</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Split Caches II

- Method $\$: Whole methods, misses only on calls/returns
- Stack $\$: Caches all stack data, always hit
- Constant $\$: Direct-mapped, no cache coherence
- Static data $\$: Direct-mapped, cache coherence
- Object $\$: Fully associative, exploits object layout
- Fully associative $\$: Small, caches everything except array fields
Cache Coherence

- Invalidate caches on
  - Reads from volatiles
  - monitorenter
- Reduces cache hits
- Simple hardware
- Easy for WCET analysis
Evaluation

- JopCMP, 1-8 cores
- 90 MHz on Cyclone II FPGA
- 2 MB SRAM, 3 cycles access
- 27 cycle worst-case latency with 8 cores and TDMA arbitration
Benchmarks

**LiftCMP**  Control industrial lift, independent threads

**Matrix**  Parallel matrix multiplication, memory bandwidth

**Raytrace**  Floating point, at most 6 threads

**Queens**  N-Queens problem, considerable synchronization

**Kfl**  Real-world application, cache performance

**Udplp**  UDP/IP stack, cache performance
Cache Performance

▸ Eight cores, TDMA arbitration
Scalability - LiftCMP

![Graph showing scalability with number of cores and speedup.](image)
Scalability - Matrix
Scalability - Raytrace

The diagram above illustrates the speedup of Raytrace algorithms as the number of cores increases. The x-axis represents the number of cores, while the y-axis represents the speedup. The graph shows the performance of two scheduling algorithms: RR and TDMA. The speedup improves as the number of cores increases, indicating better scalability for both algorithms. The performance difference between RR and TDMA is minimal, suggesting that both algorithms scale similarly with increasing core counts.
Scalability - Queens

![Graph showing scalability with number of cores and speedup for different protocols (RR and TDMA). The graph plots speedup on the y-axis and number of cores on the x-axis, with linear trends for both protocols.]
Comparison with SHAP - LiftCMP

- SHAP DE2: 60 MHz, 16-bit SRAM, 2 cycles
- SHAP ML505: 80 MHz
  SSRAM 3 cycles, 1 cycle pipelined
Conclusion

- JopCMP scales well
- Split caches are effective
- Can compete with non-time-predictable architecture
- Future work: Prefetch arrays to exploit spatial locality
Thank you for your attention!