Models of computation for energy-efficient time-aware distributed embedded systems

Jeronimo Castrillon
Chair for Compiler Construction (CCC)
TU Dresden, Germany

International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC’21)
Singapore University of Technology, Singapore. December 22 2021
Incredible evolution over the last decades

SoCs: Long history of specialization and interaction with environment

Systems on Chip (SoC): Evolution

- Incredible evolution over the last decades
- SoCs: Long history of specialization and interaction with environment

1999

2020+


https://www.hpcwire.com/2017/04/10/nvidia-responds-google-tpu-benchmarking/
Models of computation for 2020+

- Need for dynamic graphs, adaptivity, time semantics


© Prof. J. Castrillon. MCS0C’21. Singapore, 2021
In this talk

- Introduction
- Quick background
- Runtime adaptivity
- CPS it’s about time
- Summary
Dataflow programming

- Graph representation of applications
  - Implicit repetitive execution of tasks
  - Good model for streaming applications
  - Good match for signal processing & multi-media

- Some MoCs allow reasoning about
  - Termination, deadlocks
  - Memory consumption
  - Efficient schedules / maximum throughput
  - Determinism
Classic MoC-based HW-SW co-design

- Model-based approach
  - Graph model of application
  - Graph model of target systems
  - Early estimation of performance/energy/…
  - Correct-by-construction code-generation

- Successful iterative co-design methodologies (since the 80s)
  - Application
  - Architecture
  - Mapping
Programming flows (static)

- Many: CAL, DOL, CPN, Daedalus, Ptolemy, CAPH, ...

- Information
  - Dataflow model: Rates, states, actions, traces, ...
  - Architecture model: Resources, interconnect, costs, ...

- Optimization: Mapping application to hardware
  - Multi-objective optimization (exact formulations, heuristics, meta-heuristics)
  - High-level simulation / cost models
In this talk

- Introduction
- Quick background
- **Runtime adaptivity**
- CPS it’s about time
- Summary
Hybrid mapping and scheduling

- Hybrid DSE: a compile and run-time approach
  - Enable adaptivity: malleable, multi-variant
  - Run-time predictability, robustness, resource aware

Design-time

- Application $\lambda_i$
- Architecture description
- Design Space Exploration

Operational points for $\lambda_1 \ldots \lambda_n$

Runtime

- Workload
- Architecture resources state
- Runtime Resource Manager

Need for speed, context, meta-information

Multiple heuristics and meta-heuristics to exploit structure, traces, …
Coping with complexity: Shape space

- Embeddings for the mapping space: Improved heuristics?
- Example: T-SNE Visualization for mappings space (8 tasks on Odroid XU4)

Smaller space, but inconclusive results on ODROID
Coping with complexity: Shape space (2)

- Embeddings for the mapping space: Improved heuristics?
- Recent results on the complex MPPA3 Coolidge platform

 Relative mapping quality

<table>
<thead>
<tr>
<th>Method</th>
<th>GBM</th>
<th>Static CFS</th>
<th>Random Walk</th>
<th>Sim. Annealing</th>
<th>Tabu Search</th>
<th>Genetic</th>
<th>Grad. Descent</th>
</tr>
</thead>
<tbody>
<tr>
<td>E3S</td>
<td>3.00</td>
<td>1.00</td>
<td>0.30</td>
<td>0.10</td>
<td>0.03</td>
<td>1.00</td>
<td>3.00</td>
</tr>
</tbody>
</table>

 Relative time to solution

<table>
<thead>
<tr>
<th>Method</th>
<th>GBM</th>
<th>Static CFS</th>
<th>Random Walk</th>
<th>Sim. Annealing</th>
<th>Tabu Search</th>
<th>Genetic</th>
<th>Grad. Descent</th>
</tr>
</thead>
<tbody>
<tr>
<td>E3S</td>
<td>10.0</td>
<td>1.00</td>
<td>0.10</td>
<td>0.01</td>
<td>0.01</td>
<td>1.00</td>
<td>10.0</td>
</tr>
</tbody>
</table>

© Prof. J. Castrillon. MCSoC'21. Singapore, 2021
Coping with complexity: Factoring symmetries out

- **Symmetries: Performance-invariant mapping transformations**
  - Exploit platform symmetries
  - Exploit application symmetries

Goens, PhD Thesis. 2021
Goens, ACM TACO 2017
Efficient algorithms for determining canonical variants, among others

Goens, ACM TACO 2017
Goens, IEEE TCAD 2021
Goens, PhD Thesis. 2021
Coping w/ complexity: Domain-specific knowledge

- Scalability for large graphs: Compose mappings of application phases
- Example: Dynamic graphs in 4/5G baseband
Coping w/ complexity: Domain-specific knowledge (2)

- Exploration time and results for actual 4G traces (graphs) on ODROID

Mapping algorithm: 
- genetic
- simulated annealing
- static cfs
- tabu search

Khasanov,
ACM TECS 2021
Run-time adaptability: Tetris

- Runtime manager aware of symmetries
- Transform mappings according to resources
- Less time and energy variations

(Linux running multiple instances of audio filter applications)
Run-time adaptability: Run-time (re-)mapping

- **Fast heuristics:** Multiple-choice Multi-dimensional Knapsack Problem (MMKP)
  - Energy and time of canonical mapping configurations computed offline
  - **Max Difference First (MDF):** Select jobs in decreasing order of energy savings
- **Synthetic applications with up to 4 tasks on ODROID** (exhaustive exploration)

---

**Graph:**

- **Relative energy (normalized to EX-MEM):**
  - MMKP-LR
  - MMKP-MDF

- **Tests:**
  - MMKP-MDF: 13.1% more energy efficient and only 3.5% off of optimum
  - Only 9% achieve optimum
  - Optimum for ~70% of tests

---

Khasanov, DATE 2020
Stress benchmark: 4/5G scheduling

- Multiple graphs per subframe: Deadlines of 0.5 and 2.5 ms!
- Comparing with work-stealing scheduler and variations of MMKP
- ODROID and ODROID extended with accelerators

![Graphs showing success rate and dynamic energy for different traces and scheduling strategies.]

<table>
<thead>
<tr>
<th>Work-Stealing</th>
<th>Dynamic scheduler</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMKP-LR</td>
<td>Hybrid, Lagrangian Relaxation</td>
</tr>
<tr>
<td>MMKP-MDF</td>
<td>Hybrid, heuristic</td>
</tr>
<tr>
<td>MMKP-MDF-J</td>
<td>Hybrid, Knapsack heuristic</td>
</tr>
</tbody>
</table>

Khasanov, ACM TECS 2021
Stress benchmark: 4/5G scheduling (2)

- Multiple graphs per subframe: Deadlines of 0.5 and 2.5 ms!
- Runtime overhead of prototype Python algorithms

<table>
<thead>
<tr>
<th>Trace</th>
<th>MMKP-LR</th>
<th>MMKP-MDF</th>
<th>MMKP-MDF-J</th>
</tr>
</thead>
<tbody>
<tr>
<td>trace0</td>
<td>508.27</td>
<td>665.62</td>
<td>1031.99</td>
</tr>
<tr>
<td>trace1</td>
<td>1116.17</td>
<td>913.81</td>
<td>1031.99</td>
</tr>
<tr>
<td>trace2</td>
<td>913.81</td>
<td>3.06</td>
<td>3.06</td>
</tr>
<tr>
<td>trace3</td>
<td>2.35</td>
<td>2.37</td>
<td>2.35</td>
</tr>
<tr>
<td>trace4</td>
<td>1.46</td>
<td>2.23</td>
<td>2.23</td>
</tr>
</tbody>
</table>

Khasanov, ACM TECS 2021
Simulation and tool prototyping

- A python-based exploration framework
  - Tool interfacing
  - Platform generation (w/ calibrated models), heuristics, representations, ...
  - Trace-driven simulation, visualization
  - Tool composition

Menard, RAPIDO 2021

In this talk

- Introduction
- Quick background
- Runtime adaptivity
- **CPS it’s about time**
- Summary
Reactors: Time-deterministic programming

- CPSs go beyond adaptive dataflow
  - Reactive behavior to external events
  - Time: crucial in the interaction with the physical
  - Distributed and heterogeneous
  - Safety criticality

- Reactors (collab. w/ UC Berkeley and others)
  - Timed semantics + dataflow
  - Allow reasoning about event ordering (determinism)

See also Prof. Lee’s talk:
https://www.youtube.com/watch?v=_jbdWky4Iys
Reactors: Time-deterministic programming

- Determinism: Given same initial state and inputs, behavior is unambiguously defined
  - Easier debugging and testing
  - Simulations are more representative
  - More tractable analysis and verification

- “Platforms”, e.g., ROS2 or Adaptive AUTOSAR, going asynchronous (publish-subscribe, SOA, actors)
Logical and physical time

- Logical time (as in synchronous models)
  - Discrete ticks: Execute dataflow graph
  - Absolute simultaneity
  - Total ordering possible (dense time)

- Physical time
  - Continuous, from the environment
  - Simultaneity: Not really
  - Impose deadlines on execution

- In reactors: logical time attempts to follow physical time
The reactor model

- **Actions** model events *(physical and logical)*
- Behavior in **reactions**, triggered by actions or ports, which can trigger actions
- A **reactor** carries state and contains reactions and reactors
- Reactions have priorities (enforce order)
- **Mutations** *(WiP)* allow adapting the application

Lohstroh, CyPhy 2019
A reactor program spawns a dependency graph across reactors

Example: Power-train control

- Reactors and dataflow

- **Angle**
  - Left Pedal (LP)
  - Right Pedal (RP)

- **Force**
  - Brake Control (BC)
  - Brakes (B)

- **Torque**
  - Engine Control (EC)
  - Engine (E)

- **Delay**
  - $d_{RP,a_{pol}} = 2$ ms

- **Connections**
  - Action
  - Connection
  - Ordered reaction
  - (anti)dependency

© Prof. J. Castrillon. MCSoC'21. Singapore, 2021
Distributed execution

- Order along logical time
- Leverage physical clocks for synchronization

Assumptions

- Network ports annotated with deadline (D): Relates to time for processing request
- Safe-to-process: Depends on maximum network latency and clock sync error bounds
Lingua Franca

- Polyglot programming: C, C++, TypeScript, Python, Rust, …
- IDE using Kieler
  https://www.rtsys.informatik.uni-kiel.de/en/archive/kieler/welcome-to-the-kieler-project
- Code generation and runtime system

Lohstroh, ACM TECS 2021
Automotive use case

- Adaptive AUTOSAR: Consortium with most automotive players
- Service-oriented architecture: C++, futures, concurrency, ...

Client Code

```cpp
int main() {
    s = ServiceProxy();
    s.set_value(1);
    s.add(2);
    result = s.get_value();
    std::cout << result.get();
    return 0;
}
```
Time determinism

- Emergency break assistant
  - Test application in AUTOSAR repo
  - Simple processing pipeline (no buffering)
  - Hidden assumption: data always read before next item is written

- Non-determinism leads to different non-deterministic executions!
  (Target system: 2 MinnowBoard Turbot3 boards connected Ethernet)

Menard, DATE 2020

© Prof. J. Castrillon. MCSoC'21. Singapore, 2021
Reactor integration

- Transactors convert reactor interfaces to AUTOSAR
- Proxies and skeletons are unmodified (standard compliant)
- Work arounds needed to handle timestamps
Reactors for automotive software

- Reactor integration
- Transactors convert reactor interfaces to AUTOSAR
- Proxies and skeletons are unmodified (standard compliant)
- Work arounds needed to handle timestamps

- Time annotations to ensure time-determinism (based on measurements)
- Removed all time-bugs with negligible overhead (not trivial to assess)
In this talk

- Introduction
- Quick background
- Runtime adaptivity
- CPS it’s about time
- Summary
Summary

- What can we achieve with formal MoCs?
  - Improving methods for optimization
  - Methods for energy-efficient adaptive execution
  - Adding time for time-determinism in distributed CPS

- Moving forward
  - Ongoing work on scaling methods to larger systems
  - Support for emerging architectures and interconnect
  - Optimization for reactor programs
  - Mutations in reactors
  - …
Thanks! & Acknowledgements

..., and previous members of the group (Andres Goens, Sebastian Ertel), and collaborators (E. Lee, M. Lohstroh, H. Härtig, G. Fettweis, A. Kumar)

Funded by

DFG Deutsche Forschungsgemeinschaft
TraceSymm (Number 366764507)

Bundesministerium für Bildung und Forschung
E4C (Number 16ME0189)

NATIONAL INSTRUMENTS

STAATSMINISTERIUM FÜR WISSENSCHAFT KULTUR UND TOURISMUS

© Prof. J. Castrillon. MCSoC'21. Singapore, 2021
References


