Doming the Beast: achieving predictability on Drive PX2

Marko Bertogna
Univeristy of Modena, Italy
marko.bertogna@unimore.it
http://hipert.unimore.it/

Paolo Gai
Evidence Srl
pj@evidence.eu.com
http://www.evidence.eu.com

GTC Europe
Munich
October 10, 2017
Next-generation Embedded Systems

- Complex workload-intensive tasks
  - Perception, planning, ML/DNN
- Latency-critical control tasks
  - Cyber-physical interaction
  - Tight actuation loops
Performance vs Predictability

✓ Performance-oriented world
  – Highly parallel data crunching applications
  – General purpose operating systems
  – Wide and dynamic library support
  – Optimized for the average case

✓ Predictability-oriented world
  – Safety critical routines
  – Hard real time requirements
  – Functional safety standards (ISO 26262, DO178c)
  – Qualification/certification (ASIL, DAL)
  – Strict programming practices (MISRA C, AUTOSAR)
  – Optimized for the worst-case
The Predictability Challenge

✓ Multiple factors may affect predictability:
  – Dynamic control flows (conditional branches, loops, etc.)
  – Concurrency (mutual exclusion, synchronization, shared resources)
  – OS interference (interrupts, I/O, preemptions, context switches, etc.)
  – Hardware (caches, memory hierarchy, OoO engines, pipeline, etc.)

✓ Lot of established results for single-core technologies
  – ASIL D platforms/RTOS, RT schedulers, Timing analyzers, etc.

✓ Multi-core systems are inherently unpredictable
  – Lots of functional and timing inter-dependencies
  – Strongly NP-hard problems
  – Need for Freedom from Interference!
  – Getting worse with increasing number of cores

✓ What about GPU?
Core-level Memory Interference

Sequential read, sequential interference

Latency [ns]

WSS [B]

- Alone
- Interf 1
- Interf 2
- Interf 3
- Cache limit

From: Capodieci, Cavicchioli, Bertogna @ IEEE ETFA 2017
Combined Interference

From: Capodieci, Cavicchioli, Bertogna @ IEEE ETFA 2017
The **HERCULES** Project:

- **Industrial-grade framework** to provide **predictable performance** on top of cutting-edge embedded COTS platforms (e.g., DRIVE PX)
- **Homogeneous programming interface** for **high performance real-time** application on top of heterogeneous COTS platforms

EU funded H2020 2016 – 2018
[hercules2020.eu](http://hercules2020.eu)
Challenges

✓ Understand **predictability** bottlenecks
  - Hardware: caches, DRAM, memory controller, bus, etc.
  - Software: OS, scheduler(s), memory accesses, task dependencies, etc.

✓ Predictably **master contention** on concurrently accessed resources
  - Memory/Computing co-scheduling support (LLC, DRAM, host complex, GPU)
  - Compiler support for predictable execution model (PREM)

✓ Manage **multi-OS settings** with heterogeneous real-time and safety requirements
  - Predictable hypervisor support (Jailhouse and NVIDIA hypervisor)
  - Enhanced Linux support for soft real-time components
  - ERIKA Enterprise RTOS for critical partitions

✓ **Simplify programmability** of next-generation embedded devices
  - OpenMP + CUDA with Real-Time extensions
Automotive Use-Case

✓ Outdoor Valet Parking System
  - Autonomous driving in structured setting
  - Real-time obstacle detection and avoidance
  - Path planning and parking maneuver
Avionic Use-Case

✓ Computer Vision for Aerial Application
  - Online machine learning for object tracking
  - Exacopter drone with guidance, navigation and control
  - Obstacle detection and avoidance

Avionic Use-Case: Doming the Beast: achieving predictability on Drive PX2
Memory is the main problem for next-generation real-time platforms
- And it will get worse as number of cores increases...

Schedule memory accesses to master contention on architectural bottlenecks
- Coordinated scheduling of memory and execution phases
- Ensure tasks may promptly access memory resources based on real-time requirements

PREM Execution Model
- Divide tasks into Memory/Computation phases
- Size depending on LLC capacity
- Execute each M/C phase non-preemptively

Compiler support to automatically rearrange tasks
- Both for host and GPU tasks
Co-scheduling Support

 ✓ Real-Time tasks modeled as recurring conditional/parallel DAGs
   - Each node consists of an M/C phase
   - Executed non-preemptively

 ✓ Memory arbitration enforced at hypervisor level
   - Component-level memory budget mechanism
   - Hypervisor intervenes only when a component (core/partition/GPU) misbehaves
   - Reduced overhead

 ✓ Compatible to non-PREM code
   - Performance vs. predictability

 ✓ Predictable inter-component communication

 Melani, Bertogna et al. – cp-DAG @ IEEE Transactions Computers 2017
 Serrano, Melani, Bertogna – cp-DAG with limited preemptions @ DATE 2016

 Biondi, Buttazzo, Bertogna – BROE server @ IEEE Transactions Comp. 2017
Software architecture

- AUTOSAR
  - RTE
- Host-side
  - PREM compiler
- OpenMP
  - RUNTIME
- NVIDIA:
  - CUDA
- GPU-side
  - PREM compiler

Programming model(s) abstraction

- Linux
- "Big.LITTLE-like" core complex
  - E.g., 4 cortex A57 + 2 Denver
- Hypervisor abstraction
- GPU management / firmware
- GPU
- ISA subdomain #1
- ISA subdomain #2

"Dorming the Beast: achieving predictability on Drive PX2"
Some words about ERIKA3

http://www.eriaka-enterprise.com

✓ Minimal RTOS implementing OSEK/VDX and AUTOSAR OS 4.3
✓ open-source license with double licensing options
✓ Available for Nvidia Drive PX2 on top of Vibrante
✓ Used by various industries and research projects around Europe

Pre-release available starting Nov 6, 2017!
Performance measurements on NVIDIA Drive PX2

Nvidia Vibrante configuration:
✓ We considered ERIKA3 pinned on one of the Cortex A57
✓ Linux on the other 3 A57 cores
✓ Other VMs moved to Denver when possible

We are interested in the following measurements
✓ ISR Latency with the CPU idle or «busy» doing RTOS primitives
✓ AUTOSAR Task wakeup Latency
✓ Linux `clock_nanosleep` periodic task latency
✓ Variability when other CPUs are executing memory intensive tasks
ISR Latency Timings

- **ISR Latency, ERIKA3 idle**
  - 9-10.5 µs

- **ISR Latency, ERIKA3 «busy»**
  - 11-14 µs

Cumulative distribution of the execution times

---

Doming the Beast: achieving predictability on Drive PX2
ISR/Task Latency Timings

ISR Latency, ERIKA3 idle
9-10.5 µs

AUTOSAR Task Latency,
26-28 µs

Cumulative distribution of the execution times
ISR/Task Latency Timings vs. Linux

**ERIKA3 ISR Latency**
- ERIKA3 idle: 9-10.5 µs

**AUTOSAR Task Latency**
- 26-28 µs

**Linux clock_nanosleep**
- 57-235 µs

✓ Cumulative distribution of the execution times
ISR Latency Timings

Degradation due to memory intensive load on other CPUs

Cumulative distribution of the execution times
Conclusions

✓ Next-gen RT applications will run on heterogeneous multi-core systems
  – Fundamental lack of predictable solutions

✓ HERCULES aims at filling this gap providing a industrial-grade framework covering the whole architectural stack
  – RTOS, Hypervisor, Compiler support, Programming model, Real-time co-scheduling algorithms, Predictable execution model and runtime
  – Test on challenging automotive and avionic use cases

✓ ERIKA3 AUTOSAR OS
  – We proposed a configuration which uses the ERIKA3 AUTOSAR OS with a proper memory bandwidth management to provide hard real-time guarantees on NVIDIA Drive PX2
Thank you!

Marko Bertogna
marko.bertogna@unimore.it

http://hipert.unimore.it

Paolo Gai
pj@evidence.eu.com

http://www.evidence.eu.com