System Impact of Distributed Multicore Systems

December 5th 2012

Software Systems Division & Data Systems Division

Final Presentation Days

Mathieu Patte (Astrium Satellites)

Alfons Crespo (UPV)

All the space you need

ASTRIUM
AN EADS COMPANY

Outline

- Context, objectives and state of the art
- Xtratum porting on NGMP
  - Approach
  - Main issues
  - Scheduling policies
  - Configuration file
  - Evaluation
- Demonstration application
- Guidelines and way forward

Study's context and objectives

- First multi core platform for space are available
- Need to analyze the impacts at system level of the switch to multi core
- Identify first use cases for multi core processor and recommend best practices

Multi core processors use in spacecrafts

- Platform:
  - No actual needs in current reference architecture
  - Emerging need: Integrated Modular Avionics for Space
- Payload:
  - More processing power, means more powerful flexible payload data processing units
  - Main users: scientific missions

Multi core architecture survey

- Multi core means shared resources
  → competition for access to shared resources (memory)
- Key element is memory architecture
  - Multi level cache
  - Advanced cache coherency management techniques (inclusive, exclusive caches)
  - Layered buses, crossbar switches
  - Multi port cache/memory controller

NGMP architecture assessment

- NGMP main features:
  - Independent L1 caches, shared L2 cache
  - Shared bus to L2 → competition between the cores
  - IOMMU allows partitioning I/O hardware resources
  - Hardware Support for ASMP
  - No hardware support for virtualization

Software techniques on multi core

- Symmetric Multi Processing:
  - One single OS is managing all the cores
  - Pros: performance
  - Cons: all applications have to use the same OS, complex OS to qualify
- ASymmetric Multi Processing:
  - Different OS on different cores
  - Pros: OS diversity
  - Cons: all the OS needs to be qualified, space separation difficult to enforce

Software techniques for multi core (cont'd)

- SMP hypervisor:
  - Lightweight SMP kernel providing virtualization services
  - Pros: each application can run its own OS
  - Cons: performance is lower than native SMP

Parallelism programming model

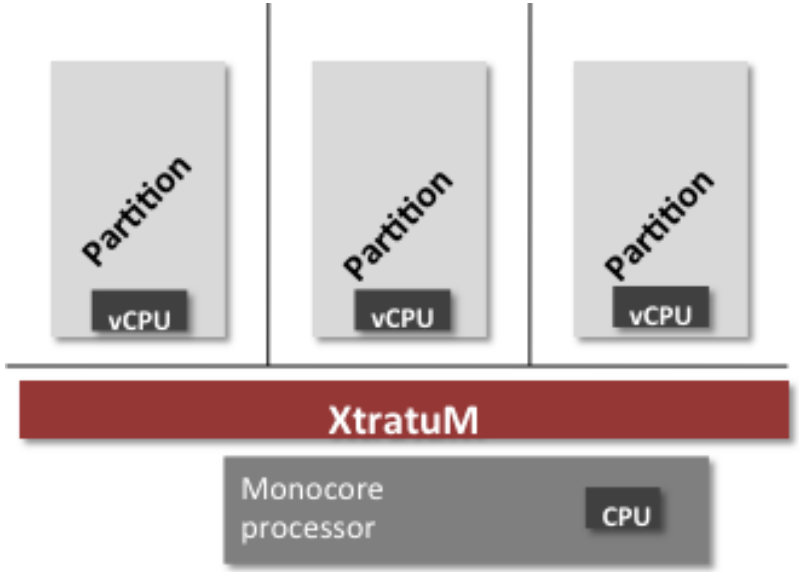
- Data parallelism: same code, different bits of data

Parallelism programming model (cont'd)

- Task parallelism: different code, same data

XtratuM: Approach

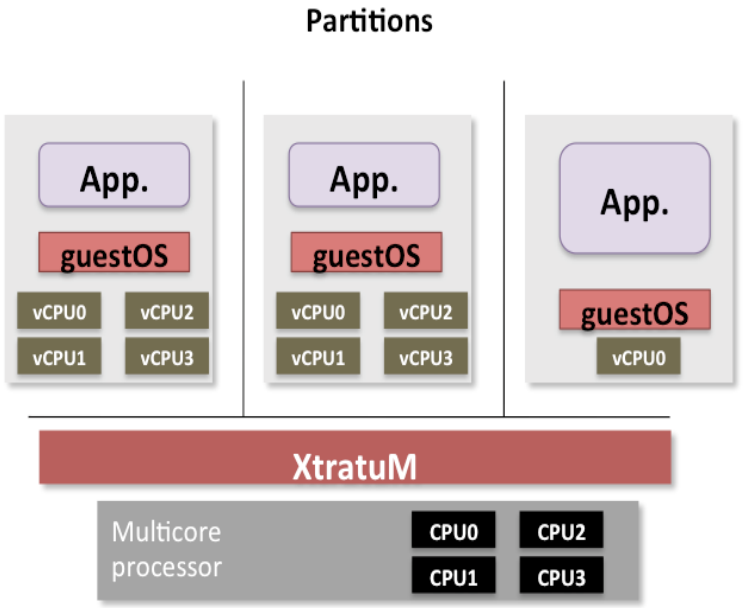
- XtratuM was initially designed for monocore architectures: LEON2 and LEON3
- It offers virtual machines (vCPU) to execute partitions



XtratuM: Hypervisor Multicore approach

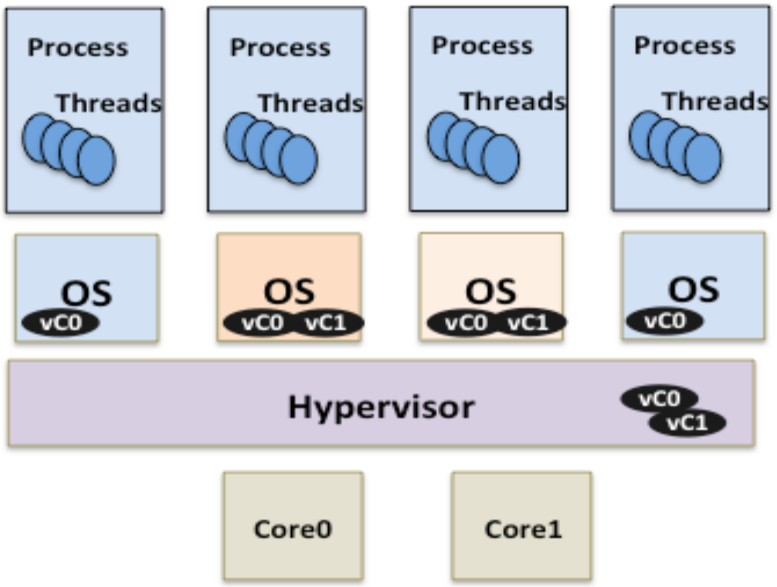
Hypervisor based system permits to build partitioned systems where partitions :

- Are **Temporal and Spatial isolated**
- **Can have different level of criticality**
- Use the **appropriated OS** for each application
- Requires an **Independent validation**
- Take advantage of the multicore to execute **mono-core OSs in a multicore platform**



XtratuM: Hypervisor based Multicore approach

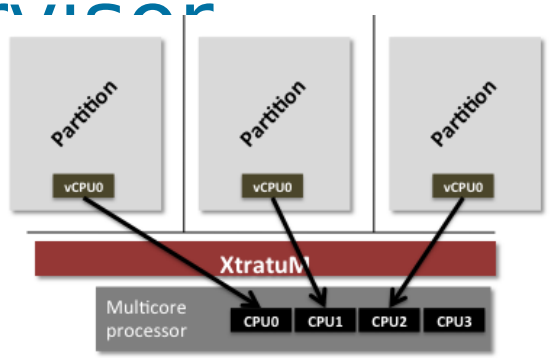
Hypervisor based Multicore Software architectures can provide a more flexible AMP or SMP view



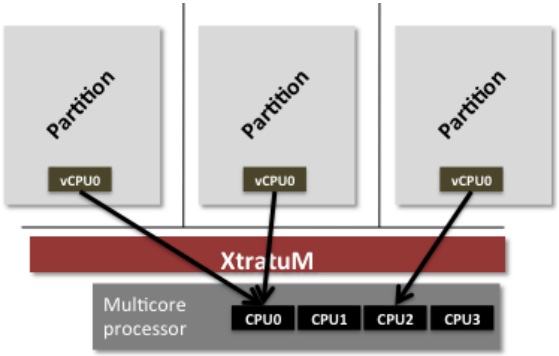
XtratuM: Multi-core hypervisor

Several schemes are possible

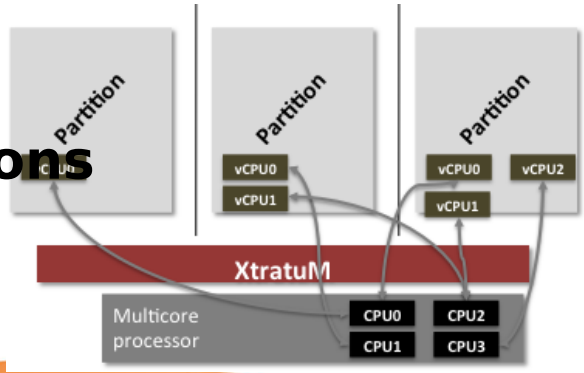
- **Monocore partitions on different real CPUs**



- **Monocore partitions on the same/different real CPUs**



- **Mono/Multi-core partitions**



XtratuM: Issues related to Multicore

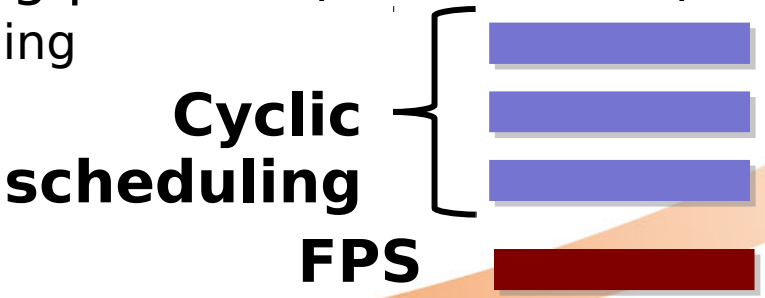
- Impact of Multicore on the services provided by the hypervisor:
  - Interrupts; Partition management; Health Monitor;
- Virtualised resources
  - Clock and Timers
  - Interrupt management (Set up/use the multiprocessor interrupt controller with extended ASMP)
  - IPI's management (Emulate IPIS through interrupts)
  - Memory management
- Scheduling
  - Main aspect to be analysed

XtratuM: Issues related to Multicore

- Virtual CPUs:
  - Inclusion of the virtual CPU (**vCPU**) concept
  - Each partition has one or more **vCPUs** (multi-core)
  - Each **vCPU** has a local partition control table
  - The clock is shared among the **vCPUs**
- New hypercalls are required tp deal with vCPUs
  - Get status
  - Start-up/resume/suspend/halt vCPU
- XML extension
  - Each partition shall define the number of VCPUs supported
  - Each slot shall indicate the VCPUID (omission means VCPUID=0)

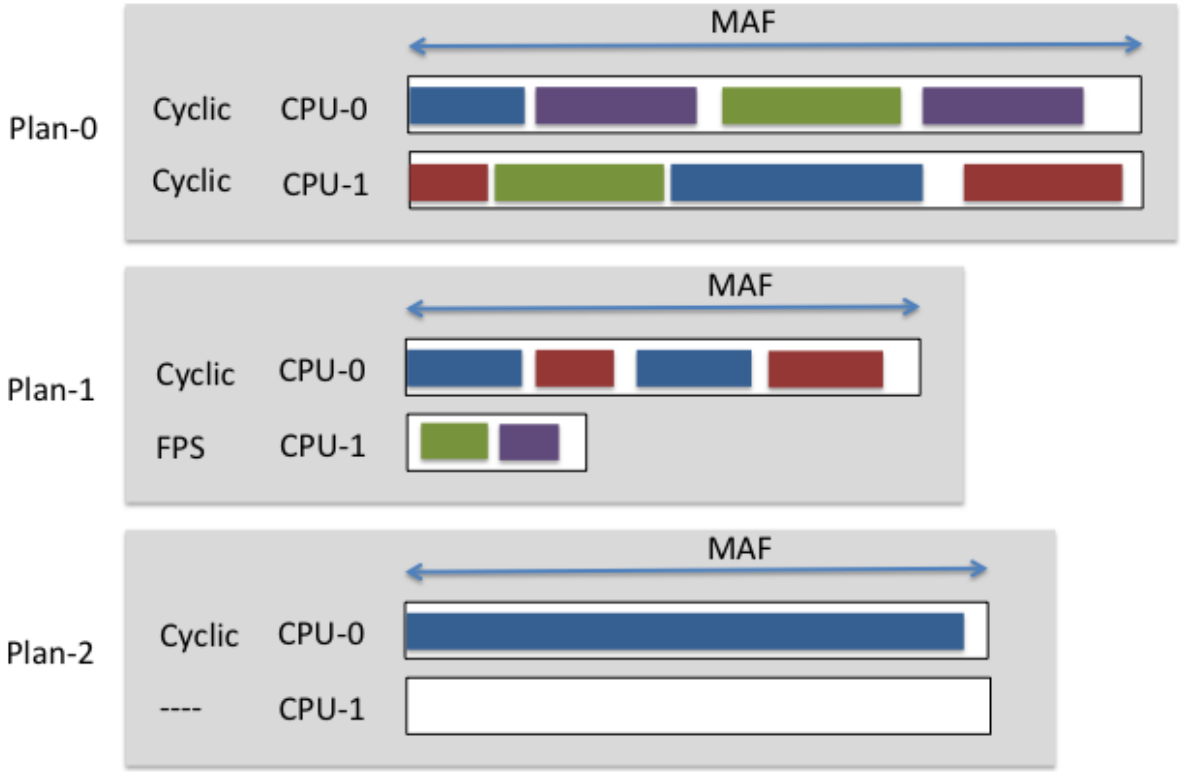
XtratuM: Scheduling Policies

- Each core can be scheduled with a defined scheduling policy
  - Specified in the configuration file
  - i.e.: 3 cores under a cyclic scheduling; 1 core FPS
- Several scheduling policies
  - Basic scheduling policy: **Cyclic scheduling**
  - Alternative scheduling policies (IO activities)
    - Fixed Priority Scheduling



XtratuM: Plan management in Multicore

- **Definition of Scheduling Plans**
  - **Multiple schedules**
  - **Plan switch: system partition**



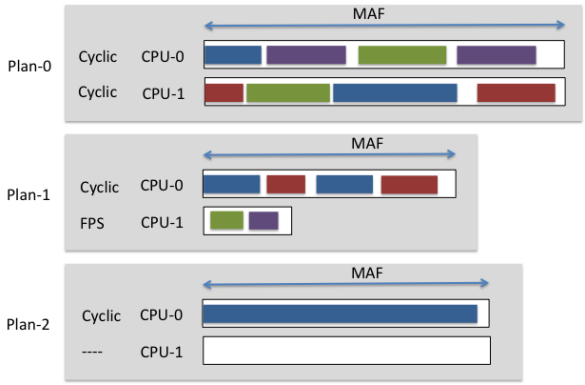
XtratuM: Multicore Scheduling Plan

- Plan specification: configuration file

```
<Processor id="0" frequency="50Mhz">
      <CyclicPlanTable>
        <Plan id="0" majorFrame="400ms">
          <Slot id="0" start="0ms" duration="200ms" partitionId="0"
vCpuId="0"/>
              <Slot id="1" start="200ms" duration="200ms"
partitionId="0" vCpuId="1"/>
        </Plan>
        ..............
      </CyclicPlanTable>
  </Processor>

  <Processor id="1" frequency="50Mhz">
      --------
      <FixedPriority>
            <Partition id="0" vCpuId="1" priority="10"/>
            <Partition id="2" vCpuId="0" priority="5"/>
      </FixedPriority>
  </Processor>

<PartitionTable>
  <Partition id="0" name="Partition1" flags="system" console="uart" noVCpus="4">
...
```



XtratuM: IOMMU description

- Similar to the MMU but at the AMBA bus level I/O.
- Provides capabilities for:
  - Protecting I/O pages (4KB-512KB page-size)
  - Translating I/O pages (4KB-512KB page-size)
- Solves the problem DMA memory protection problem
- Statically defined

XtratuM: IO-MMU: Configuration

- Configured statically during XML stage:

```
<XMHypervisor ... >
...
  <IoMmu>
     <AhbMst id="0" partitionId="0" busRouting="processor" vendorId="0xc" deviceId="0x0f" />
     <AhbMst id="1" partitionId="1" busRouting="memory"  vendorId="0xc" deviceId="0x1f" />
  </IoMmu>
</XMHypervisor>
```

- Where
  - Id: master ahb id; identifies the Master configuration register
  - PartitionId: bit [3:0] (GROUP) GRIOMMU Master configuration register
  - BusRouting: bit [4] (BS) GRIOMMU Master configuration register
  - VendorId: bits [31:24] (VENDOR) GRIOMMU Master configuration register
  - DeviceId: bits [23:12] (DEVICE) GRIOMMU Master configuration register

XtratuM: Test Suites

- Functional tests adapted from mono-core
  - CNES tests for LEON2 and LEON3
    - Spatial & temporal isolation; Partition Mngmt; Interrupt Mgmt; Health Monitor; IPC; ….
- SIMDS functional requirements
    - Boot Mngmt; Cache; Scheduling; IO; IPC; IRqs,; ….
- Performance evaluation
    - Dhrystone benchmark
    - CoreMark benchmark
    - IPC measurements

HAPS-54 Board

LEON4

4 Cores @ 50MHz

XtratuM: Performance Evaluation

- Dhrystone on native HW vs Dhrystone on a XM partition
- CoreMark on native HW vs CoreMark on a XM partition
- Use 1 core
- In the Partitioned target the benchmark is completed in 1 slot

Dhrystone	Native HW	XM Partition
Number of iterations	100000	100000
Time	2006149	2006315
Performance Lost (%)		0,008%

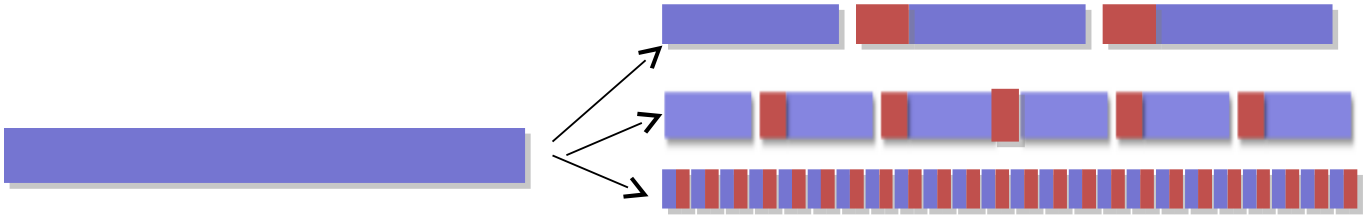
Clock Management (1sec)	20

CoreMark	Native HW	XM Partition
Number of iterations	1200	1200
Time	15436453	15604193
Performance Lost (%)		1,087%

WindowOverflowTrap	2235
WindowUnderflowTrap	2235
Clock Management (1sec)	14

XtratuM: Performance Evaluation

- CoreMark on a XM partition
  - Executed with different slot durations



Slot duration	No Slots	Time (s)	Perf. Lost	CoreMark/MHz
30 sec	1	15,604194		1,538048
1000	16	15,606633	0,0156%	1,537808
500	32	15,609788	0,0358%	1,537497
100	157	15,634318	0,1931%	1,535085
10	1592	15,918195	2,0123%	1,507709

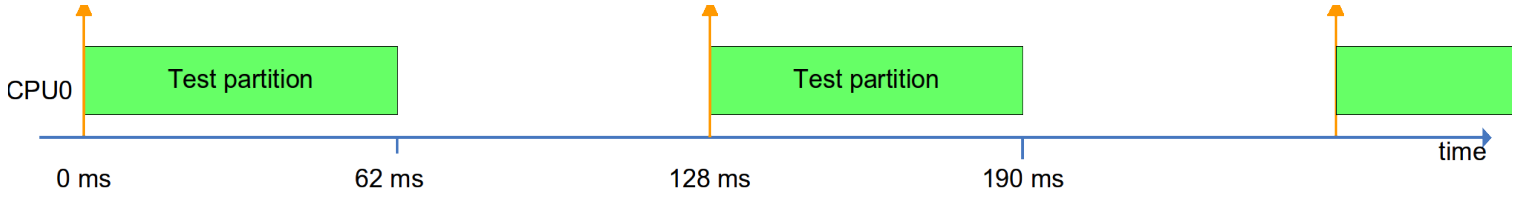
XtratuM: Performance Evaluation

- CoreMark on a XM partition
  - Executed on several cores

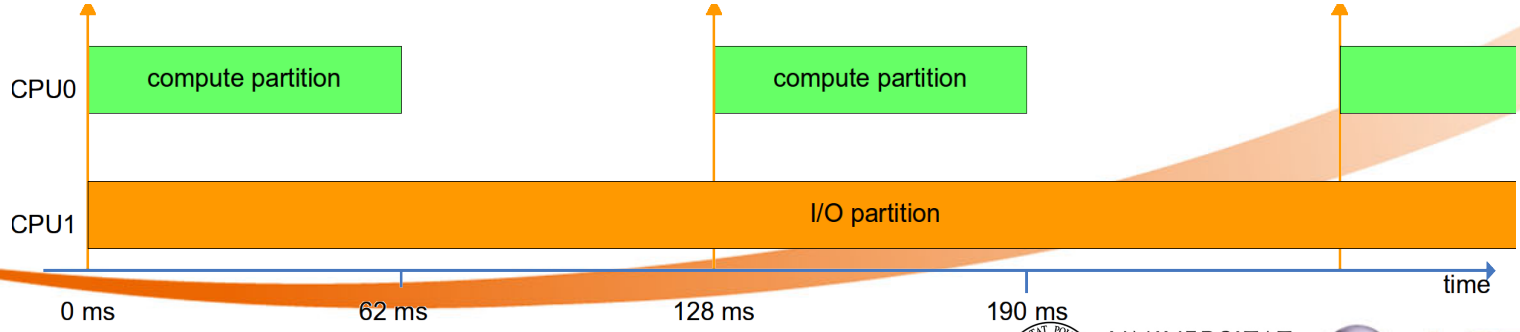
Core id	Time(s)	Perf. Lost
Core 0	15,60663	
Core 0	15,60967	0,0195%
Core 1	15,60968	**0,0195%**
Core 0	15,61191	**0,0338%**
Core 1	15,61185	0,0334%
Core 2	15,61186	0,0335%

Demonstration application

- Simple application: MD5 computation of incoming SpW packets
- Benchmark different I/O management techniques
  - Cyclic scheduling on one core

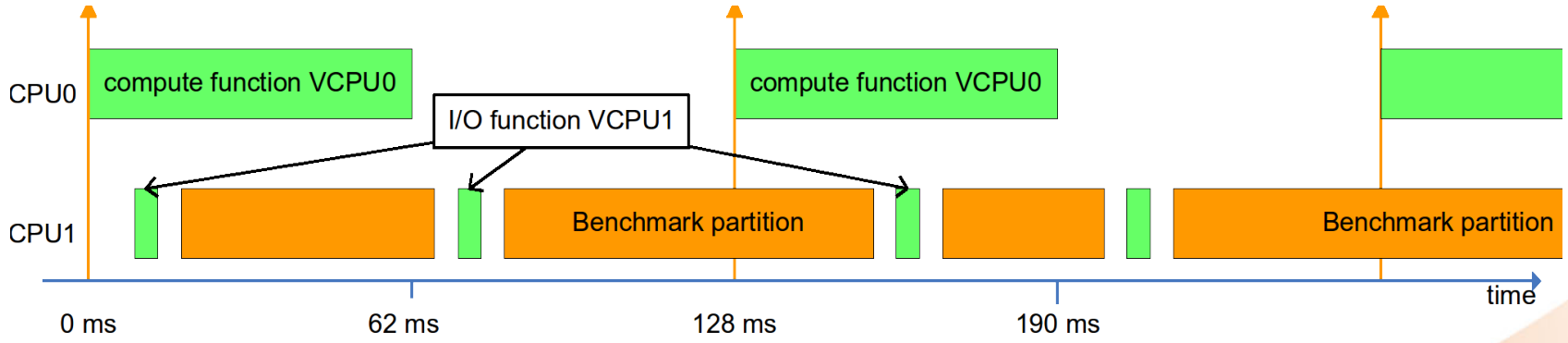


  - Cyclic scheduling and I/O partition

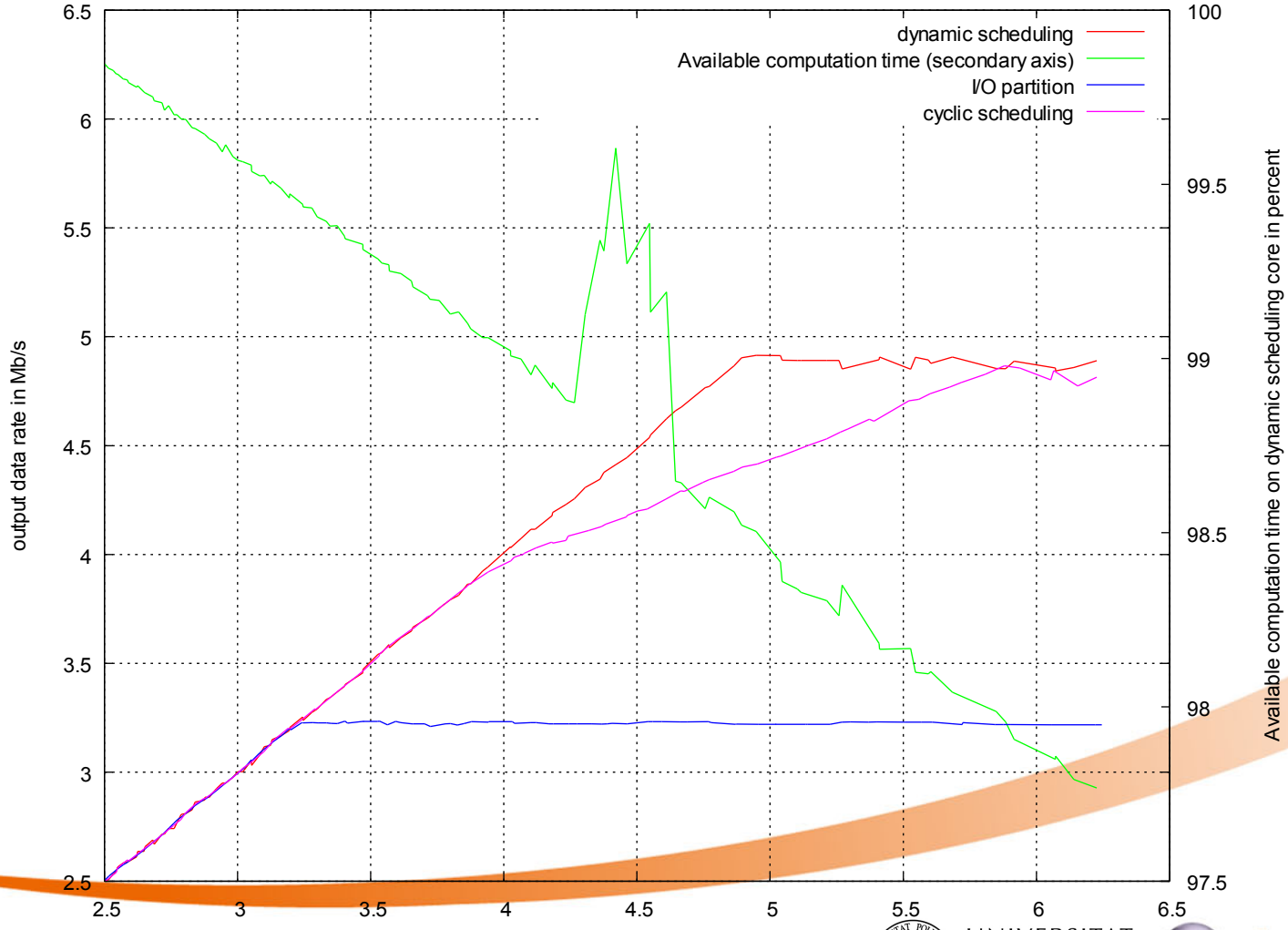


Demonstration application (con't d)

- Benchmark different I/O management techniques
  - Cyclic scheduling and fixed priority scheduling



Demonstration application results



Guidelines and way forward

- Multi core architecture
  - Minimize hardware coupling (need for system level simulation to asses the best techniques)
  - Optimize hardware for virtualization
  - Improve hardware debug facilities (real time tracing)
  - Provide comprehensive simulation environment

Guidelines and way forward (con't d)

- Hypervisor software
  - Improve development and validation environment
  - Provide a multithreading API to partition programmers
  - Improve fixed priority scheduling policy to ensure better timing isolation (priority server)
- Flexible data processing use case
  - Use task parallelism programming model
  - Possible integration of data processing tasks with instrument control tasks using IMA

Guidelines and way forward (con't d)

- IMA on multi core
  - Multi core can ease IMA implementation
  - Use direct I/O allocation (IOMMU)
  - Use fixed priority scheduling to implement interrupt handlers or allocate one core for the I/O partition
  - Modify RTOS to support concurrent execution of interrupt handlers
  - **Hardware coupling can impact time partitioning!**

Conclusions

- Demonstration of multi core techniques on NGMP
  - SMP hypervisor
  - Direct I/O allocation with IOMMU
  - VCPUs and fixed priority scheduling for interrupt handlers
- Multi core can efficiently be used for IMA and flexible data processing
- Prerequisites:
  - Multithreading software support (at hypervisor or RTOS level)
  - Multi core hardware improvements
  - Multi core WCET and scheduling analysis tools (ProArtis, ProCXim)