NICTA

Operating System Support for the Heterogeneous OMAP4430:

A tale of two micros

Etienne Le Sueur and Simon Rodgers
with Aaron Carroll and Bernard Blackham

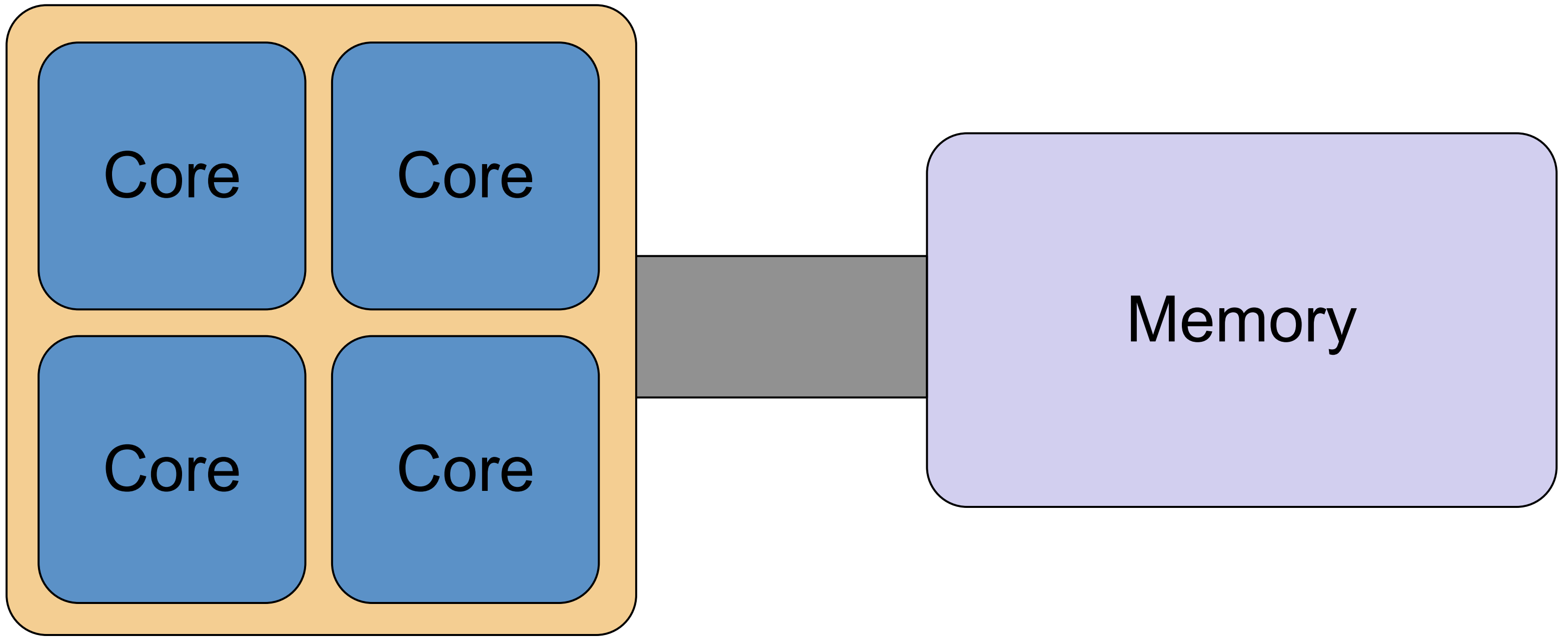
Linux.conf.au, Ballarat
17 January 2012

Australian Government
Department of Broadband, Communications and the Digital Economy
Australian Research Council

NICTA Funding and Supporting Members and Partners

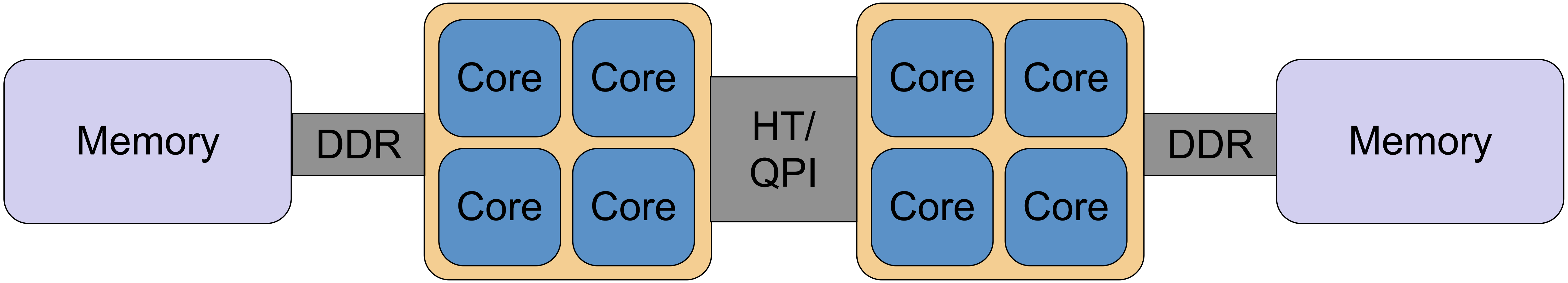
Traditional chip-multi-processors

- Symmetric Multi-Processing (SMP)
  - – two or more identical processors / cores



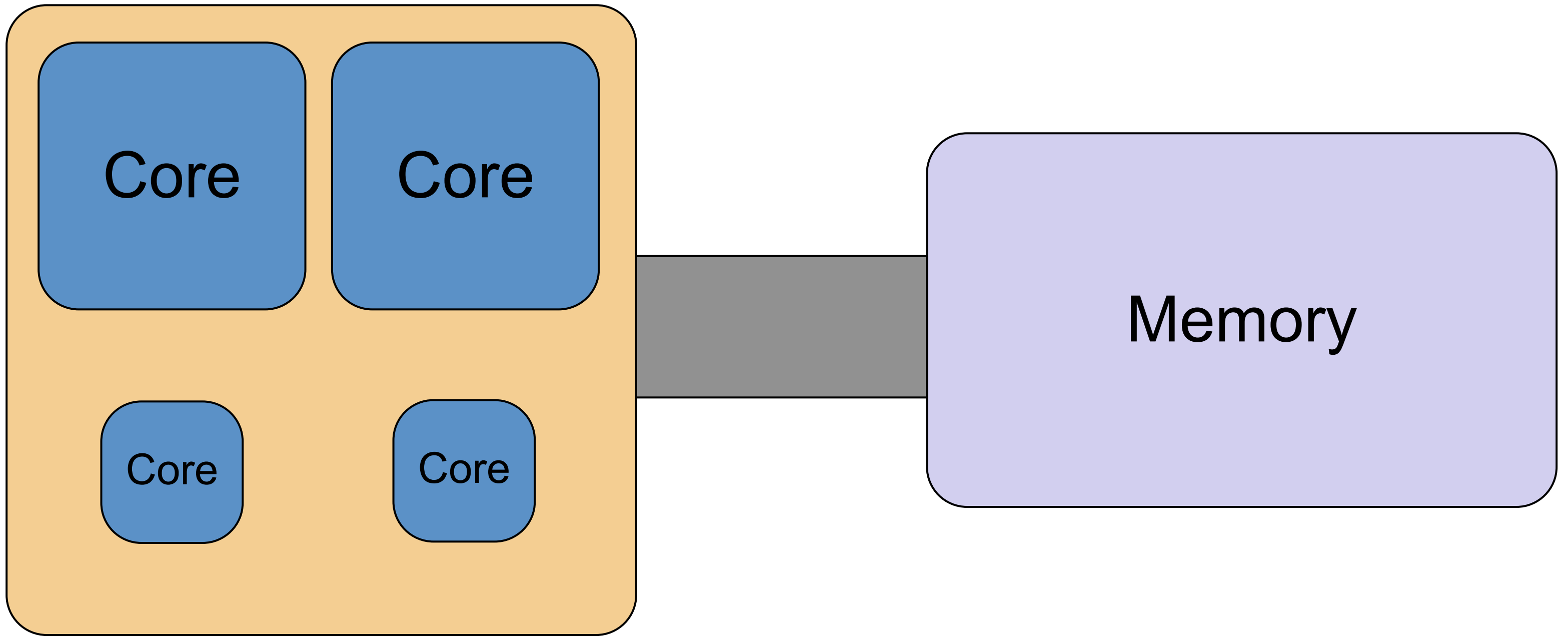
Traditional chip-multi-processors

- Symmetric Multi-Processing (SMP)
  - – two or more identical processors / cores

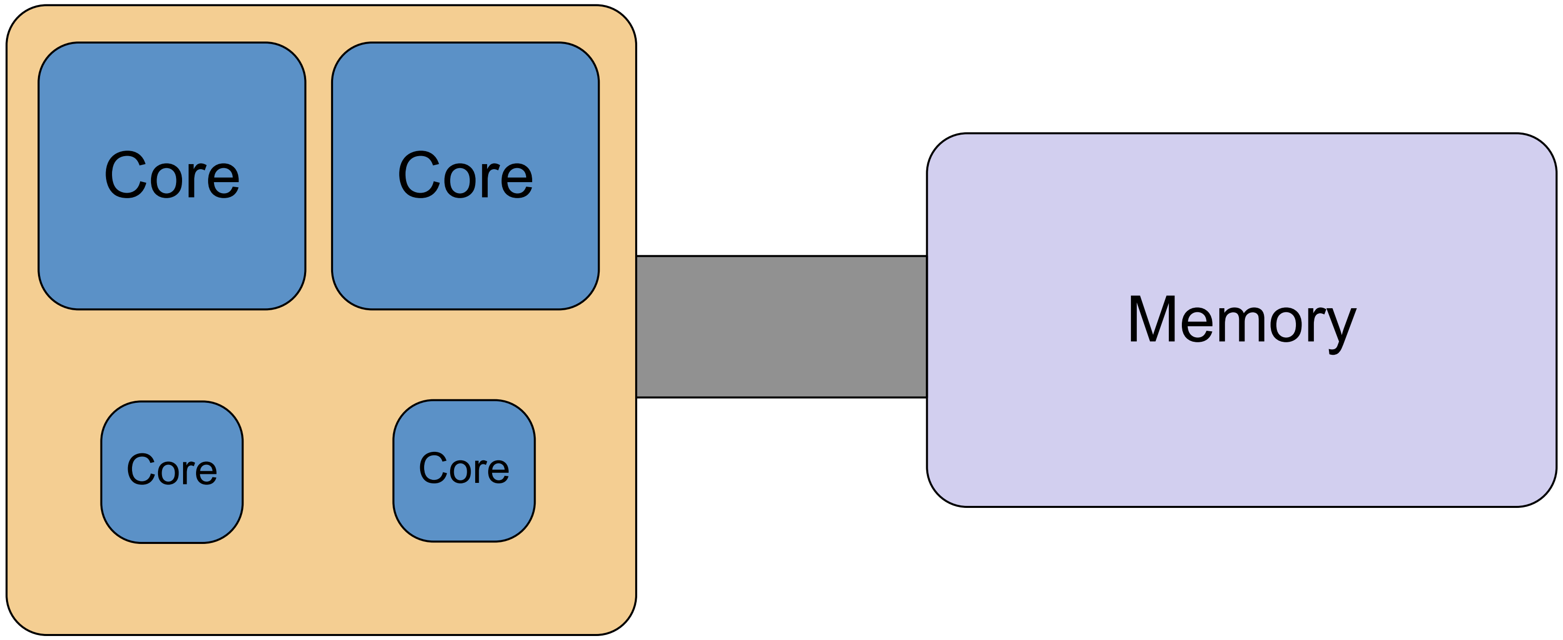


Heterogeneous chip-multi-processors

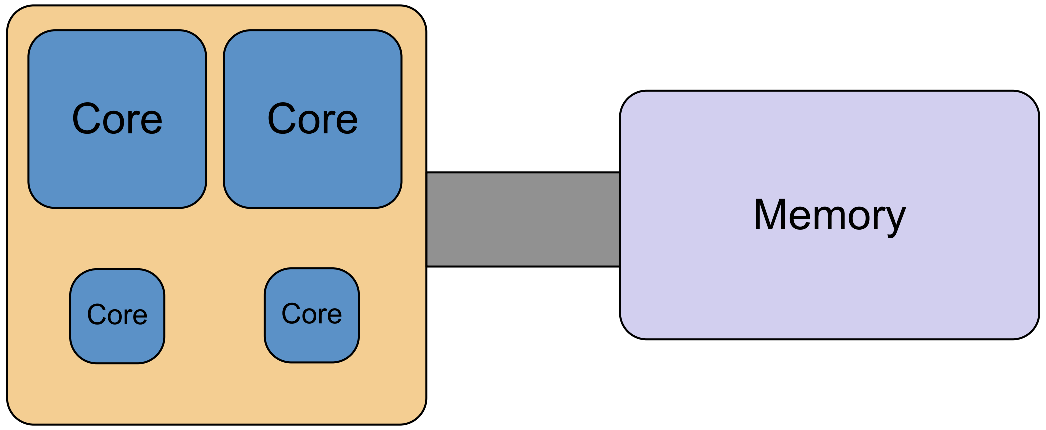
- ‘Asymmetric Multi-Processing’ (AMP)
  - – several different processors / cores



Core asymmetry

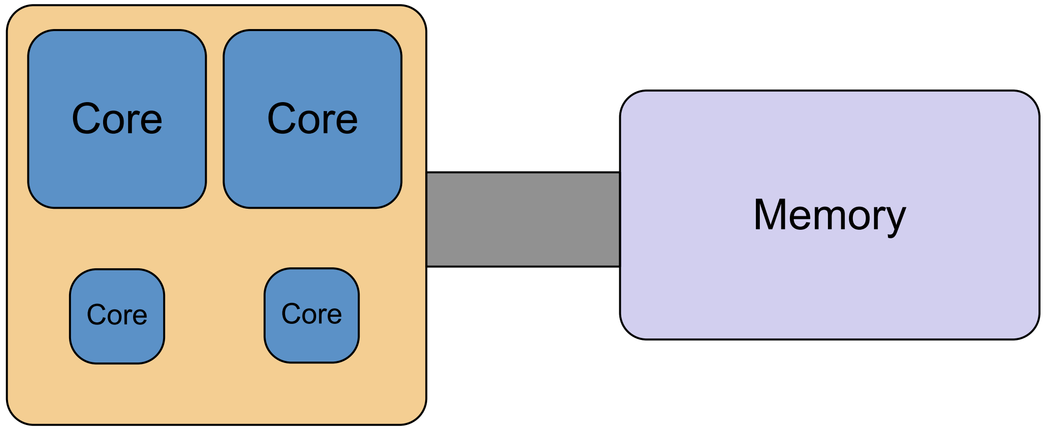


Core asymmetry



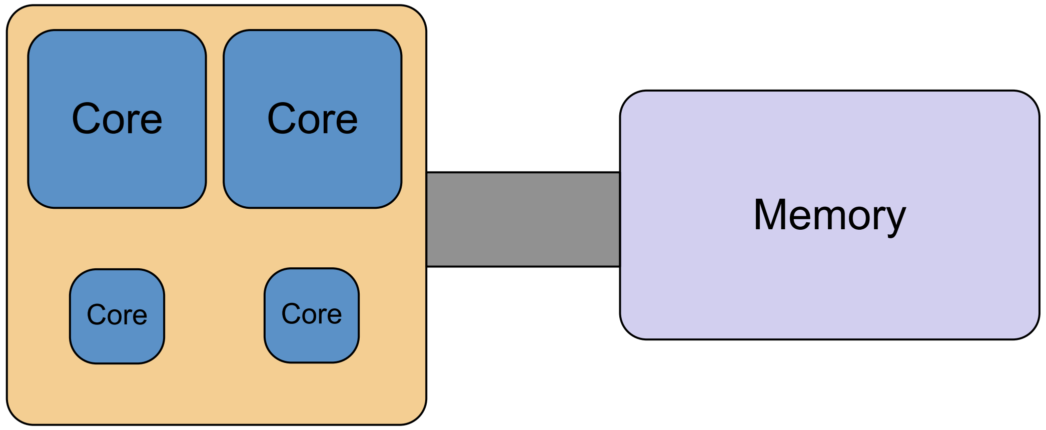
Core asymmetry

- Performance
  - – different frequency
  - – different pipelines
  - – different size caches/TLBs
  - – etc.



Core asymmetry

- Performance
  - different frequency
  - different pipelines
  - different size caches/TLBs
  - etc.
- Instruction Set Architecture
  - ARMv7 vs Thumb
  - SSE vs no SSE



Benefits of heterogeneous systems

- Energy efficiency
  - small cores have a small die area
  - low-power off-load allows big cores to sleep while small cores work
- Computational efficiency in general
  - can fit more small cores in a given area giving greater parallel performance
  - single-threaded workloads can still get performance on a big core

OS design for heterogeneous processors

- Models
  - restrictive
  - hybrid
  - unified
  - distributed

Restrictive model

- Restrict all programs to the sub set of features supported by both types of cores
- Limited to a subset of features
  - performance may not be as good as it could be

Hybrid model

- Allow user programs to interrogate the heterogenous capabilities of the system
- Allows user programs to execute on the cores that provide the features they need.
  - on Intel, CPUID
  - sched_setaffinity(target_core)

Unified model

- Allow programs to use the combined feature set of the two types of cores
- Fault-and-migrate when an unsupported feature is requested
- *Proxy* instructions in light-weight processes
- Requires a lot of OS trickery

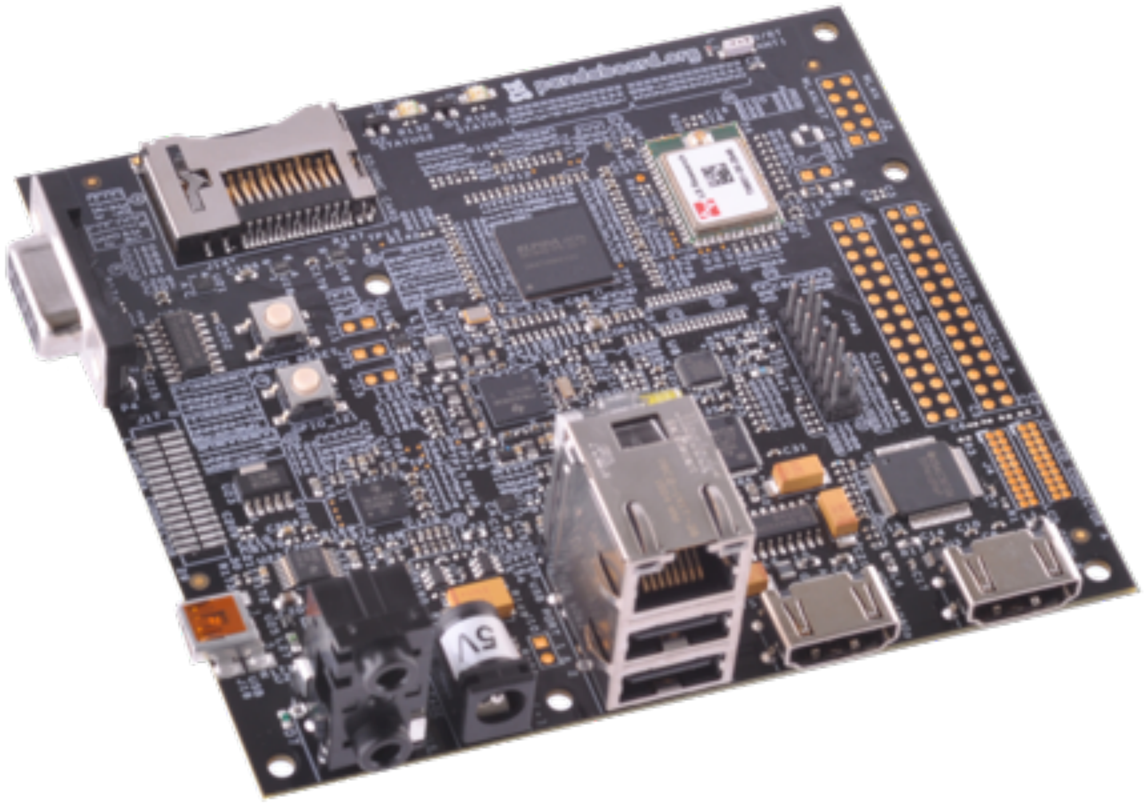
Distributed shared-memory model

Distributed shared-memory model

- Simply provide a mechanism for loading and running code on different cores
  - SPUfs
    - IBM Cell processor
    - filesystem based, at least it fits the Unix model!
  - TI SysLink
    - provides mechanism to load software into co-processors
    - runs within the TI SYS/BIOS OS framework

Pandaboard

- USB, DVI/HDMI, Ethernet, WiFi, Bluetooth, SD-card, etc…



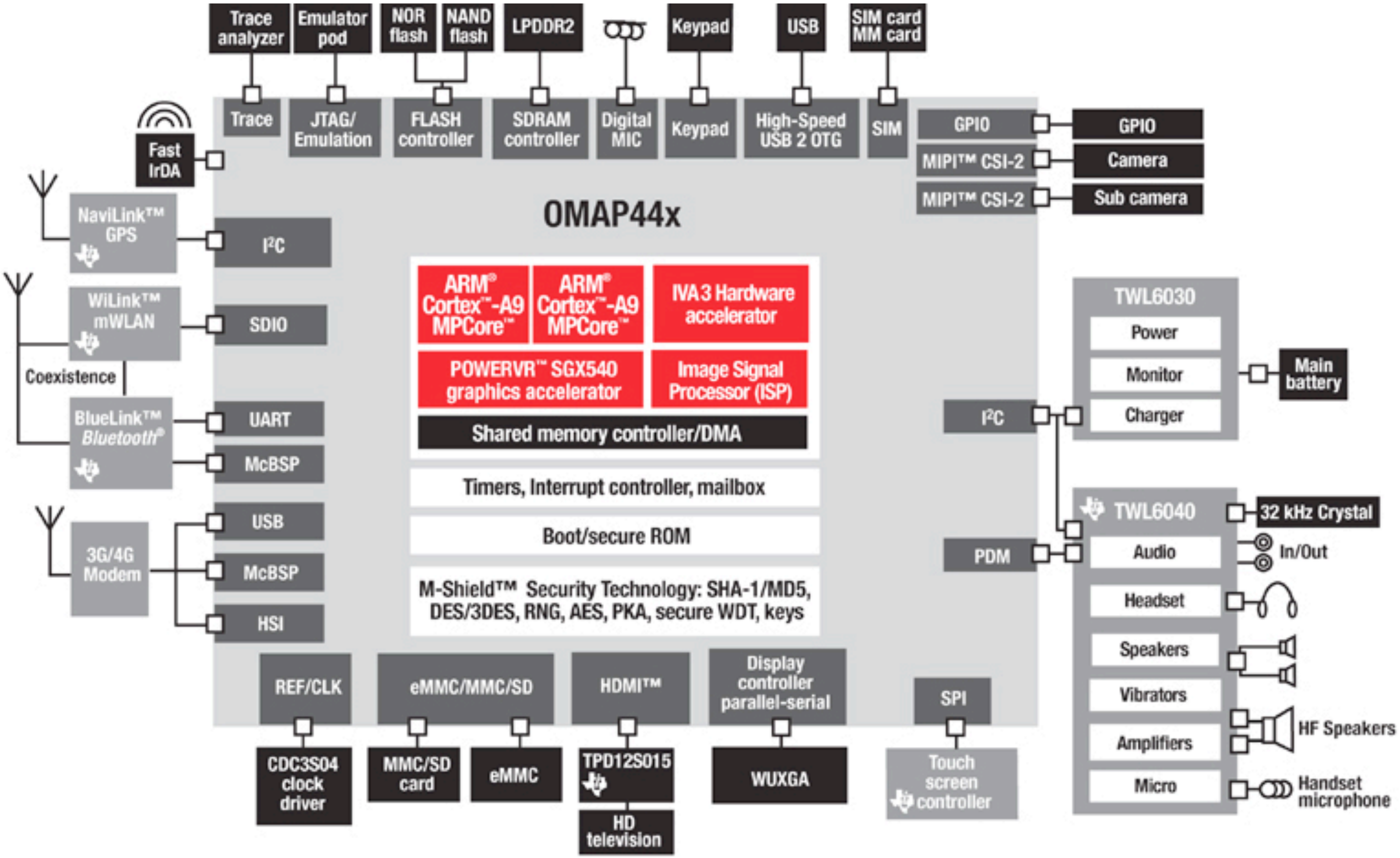
The first OMAP4430 hardware platform

PEAP project

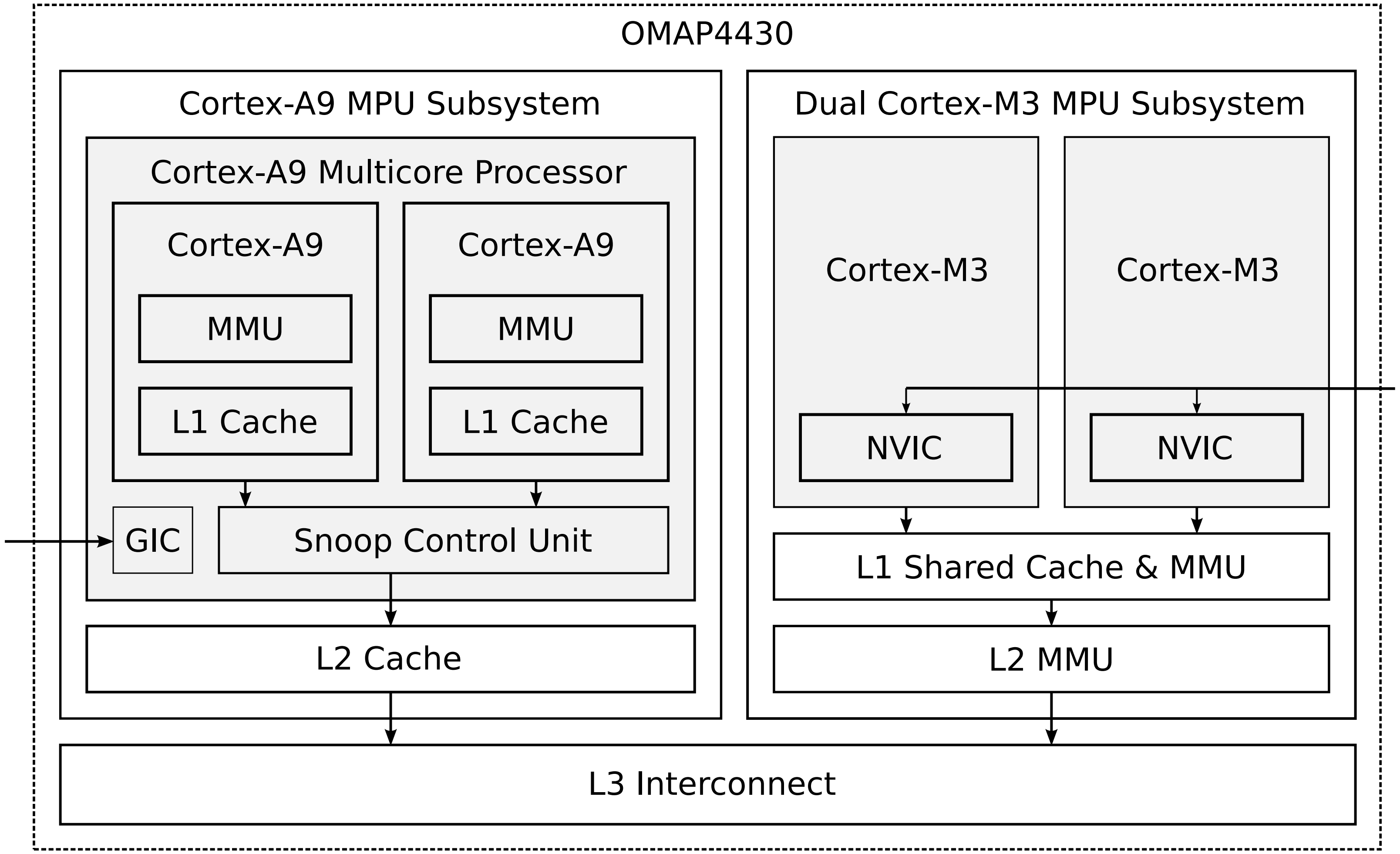
- TI gave us a free Pandaboard!
  - *Pandaboard Early Adopter Program* (PEAP)
  - project chosen from about 50 potentials
- The plan was...
  - a single Linux image running on both architectures
  - treat both types of core as general-purpose
  - examine effects on
    - Energy consumption
    - Efficiency

TI OMAP 4430

NICTA



What cores?



	ARM Cortex-A9 Core	ARM Cortex-M3 Core
Architecture	ARM v7-A	ARM v7-M
ISA Support	ARM, Thumb-2, floating-point, NEON, DSP, Jazelle	Thumb-2
Memory Protection	Memory Management Unit	Optional 8 region MPU
Clock Speed	1.0 GHz	266 MHz

A common instruction (sub)set exists

A common instruction (sub)set exists

- Both A9 and M3 support Thumb-2, but...

A common instruction (sub)set exists

- Both A9 and M3 support Thumb-2, but...
- Neither support all the features of Thumb-2

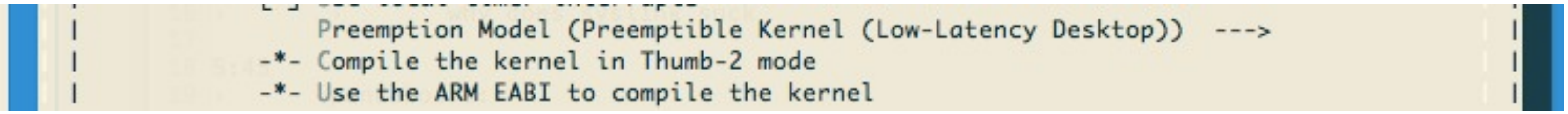
A common instruction (sub)set exists

- Both A9 and M3 support Thumb-2, but...
- Neither support all the features of Thumb-2
- No strict subset of user-visible ISA between A9 / M3
  - SDIV / UDIV (M3 only)
  - UMAAL, SSAT16, USAT16, SETEND (A9 only)
  - optional DSP extension (A9 only)

A common instruction (sub)set exists

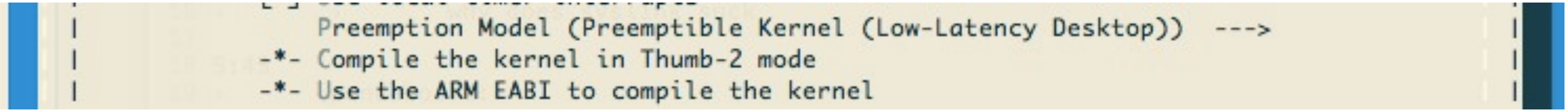
- Both A9 and M3 support Thumb-2, but...
- Neither support all the features of Thumb-2
- No strict subset of user-visible ISA between A9 / M3
  - SDIV / UDIV (M3 only)
  - UMAAL, SSAT16, USAT16, SETEND (A9 only)
  - optional DSP extension (A9 only)
- System ISA is also completely different

Thumb-2 support for kernel



```
|               Preemption Model (Preemptible Kernel (Low-Latency Desktop))  --->
|           -*- Compile the kernel in Thumb-2 mode
|           -*- Use the ARM EABI to compile the kernel
```

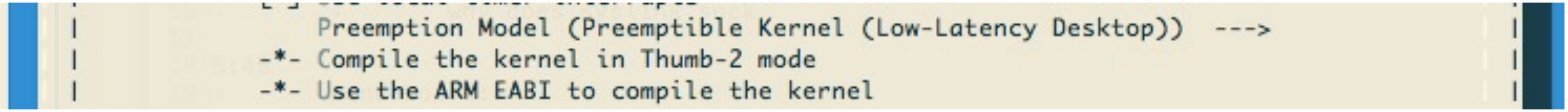
Thumb-2 support for kernel



```
|            Preemption Model (Preemptible Kernel (Low-Latency Desktop))  --->
|         -*- Compile the kernel in Thumb-2 mode
|         -*- Use the ARM EABI to compile the kernel
```

- An easy first step
  - – a small amount of assembly hacking
  - – found bug in OMAP init routines, booting second core in ARM mode
  - – userspace-helper functions still compiled as ARM
    - ABI defines them as ARM
    - glibc tries to put the CPU in ARM mode
    - patch glibc! more later...

Thumb-2 support for kernel



```
        Preemption Model (Preemptible Kernel (Low-Latency Desktop))  --->
    -*- Compile the kernel in Thumb-2 mode
    -*- Use the ARM EABI to compile the kernel
```

- An easy first step
  - a small amount of assembly hacking
  - found bug in OMAP init routines, booting second core in ARM mode
  - userspace-helper functions still compiled as ARM
    - ABI defines them as ARM
    - glibc tries to put the CPU in ARM mode
    - patch glibc! more later...

- It works!

A second step...

A second step...

...Linux on the Cortex-M3!

A second step...

...Linux on the Cortex-M3!

- Standard Linux does not support the Cortex-M3 **:-(**
  - M3 core is designed for small embedded systems without a MMU

A second step...

...Linux on the Cortex-M3!

- Standard Linux does not support the Cortex-M3 **:-(**
  - – M3 core is designed for small embedded systems without a MMU
- uClinux support exists
  - – fork of Linux designed to support small micro-controllers

A second step...

...Linux on the Cortex-M3!

- Standard Linux does not support the Cortex-M3 **:-(**
  - M3 core is designed for small embedded systems without a MMU
- uClinux support exists
  - fork of Linux designed to support small micro-controllers
- Our plan
  - take the support from uClinux and put it into standard Linux
  - Linux can't directly boot an M3 core, so...
  - partition memory in two
  - bootstrap M3 Linux from A9 Linux

Problems with Linux on the Cortex-M3

- Memory management
- Exception handling
- Toolchain

Memory management

Memory management

- Page table
  - – virtual-to-physical memory mappings

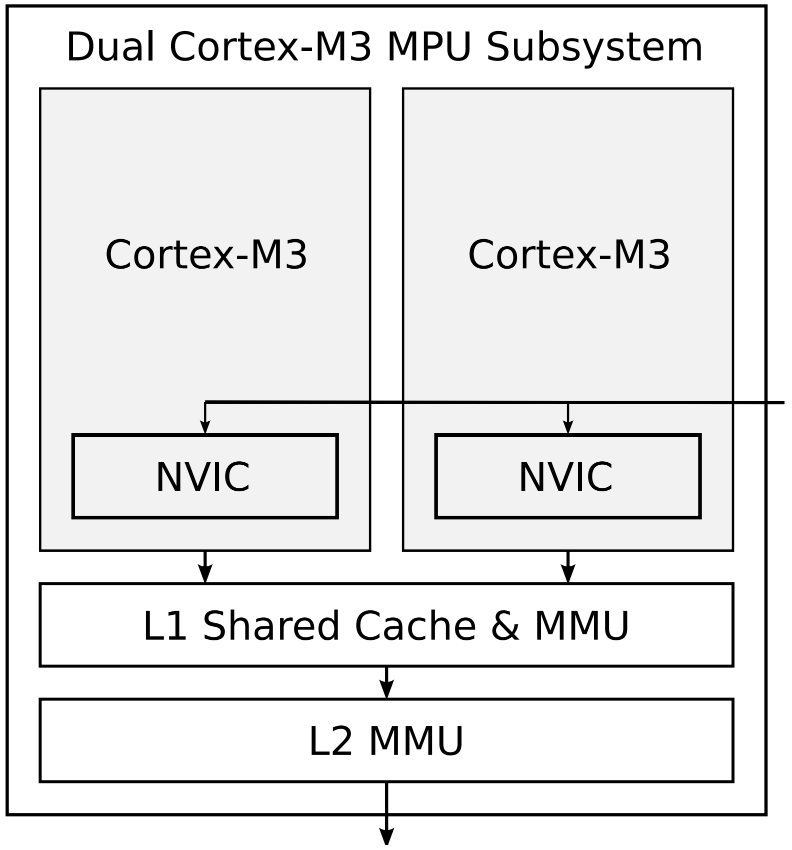
Memory management

- Page table
  - virtual-to-physical memory mappings
- Memory management unit (MMU)
  - translation
  - permissions (read-only, execute-only)

Memory management

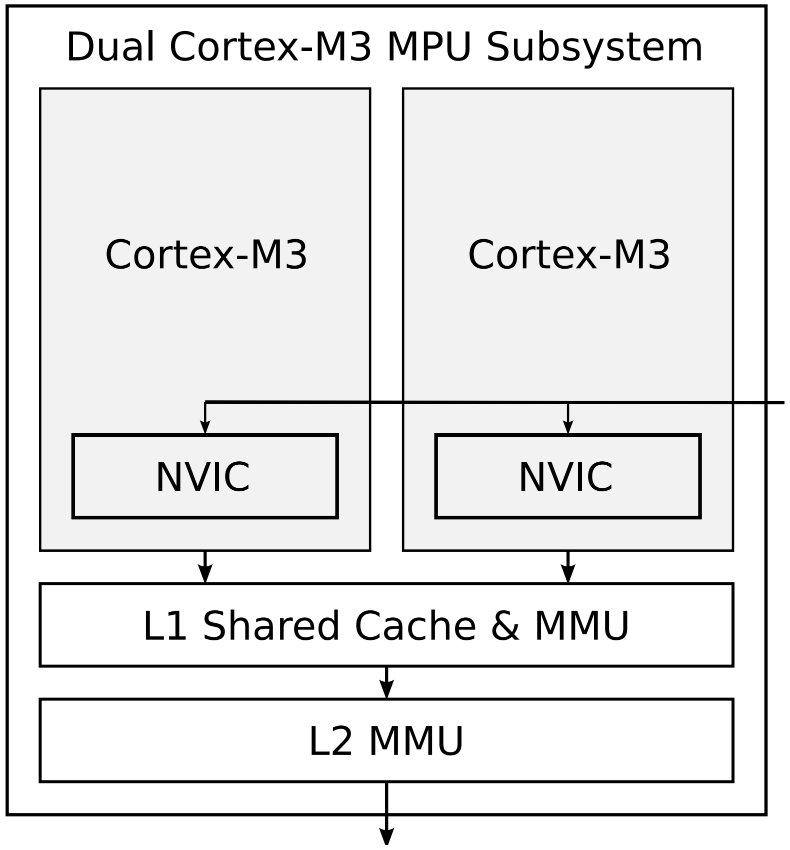
- Page table
  - virtual-to-physical memory mappings
- Memory management unit (MMU)
  - translation
  - permissions (read-only, execute-only)
- Translation look-aside buffer (TLB)
  - cache for virtual memory mappings
  - software loaded
  - hardware pagetable walker

Memory management on the Cortex-M3



Memory management on the Cortex-M3

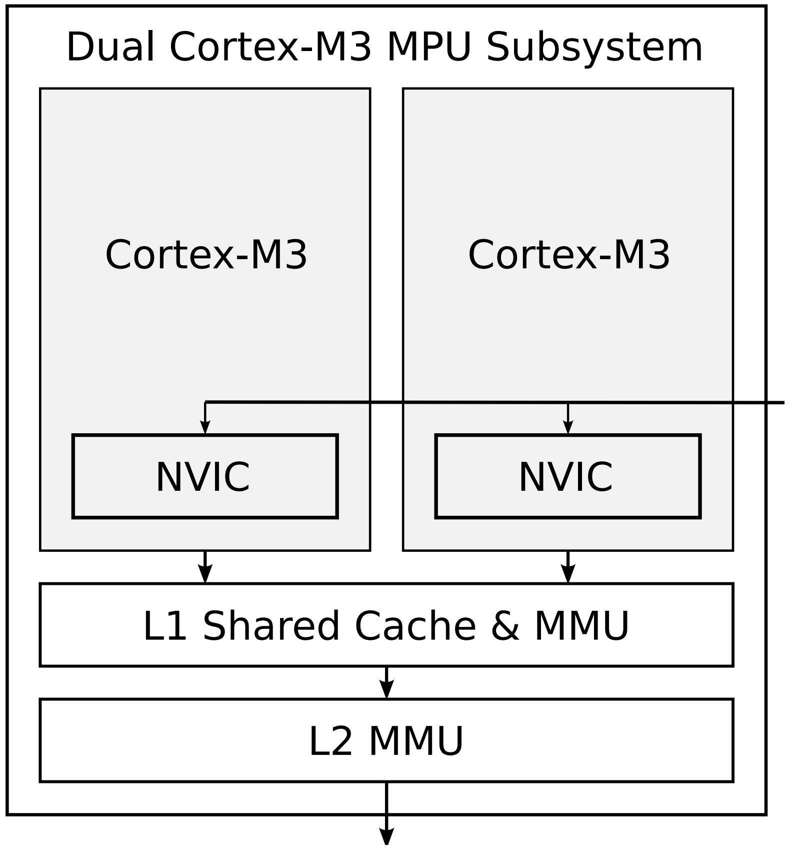
- Subsystem's shared MMUs
  - L1 shared cache & MMU
    - 10 entry TLB
    - read-only & execute-only permissions
    - software loaded
  - L2 MMU
    - 32 entry TLB
    - hardware walker (ARMv6 without permissions)



Memory management on the Cortex-M3

NICTA

- Subsystem's shared MMUs
  - L1 shared cache & MMU
    - 10 entry TLB
    - read-only & execute-only permissions
    - software loaded
  - L2 MMU
    - 32 entry TLB
    - hardware walker (ARMv6 without permissions)



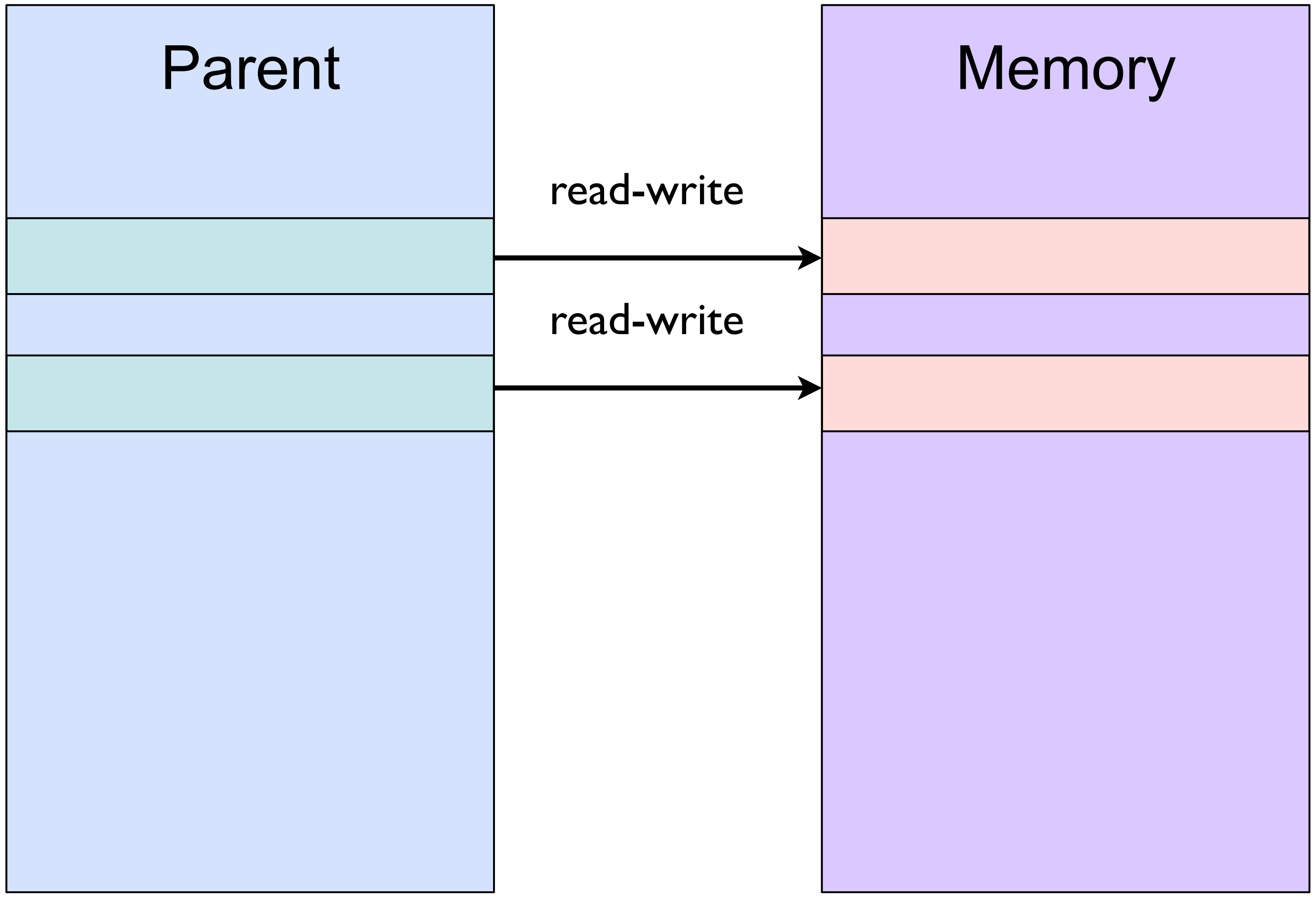
- Limitations
  - no supervisor-mode permissions - separate kernel page table
  - no tagged TLB - flush the TLB on every context switch

Copy on write

- Used throughout Linux
  - shared pages, fork

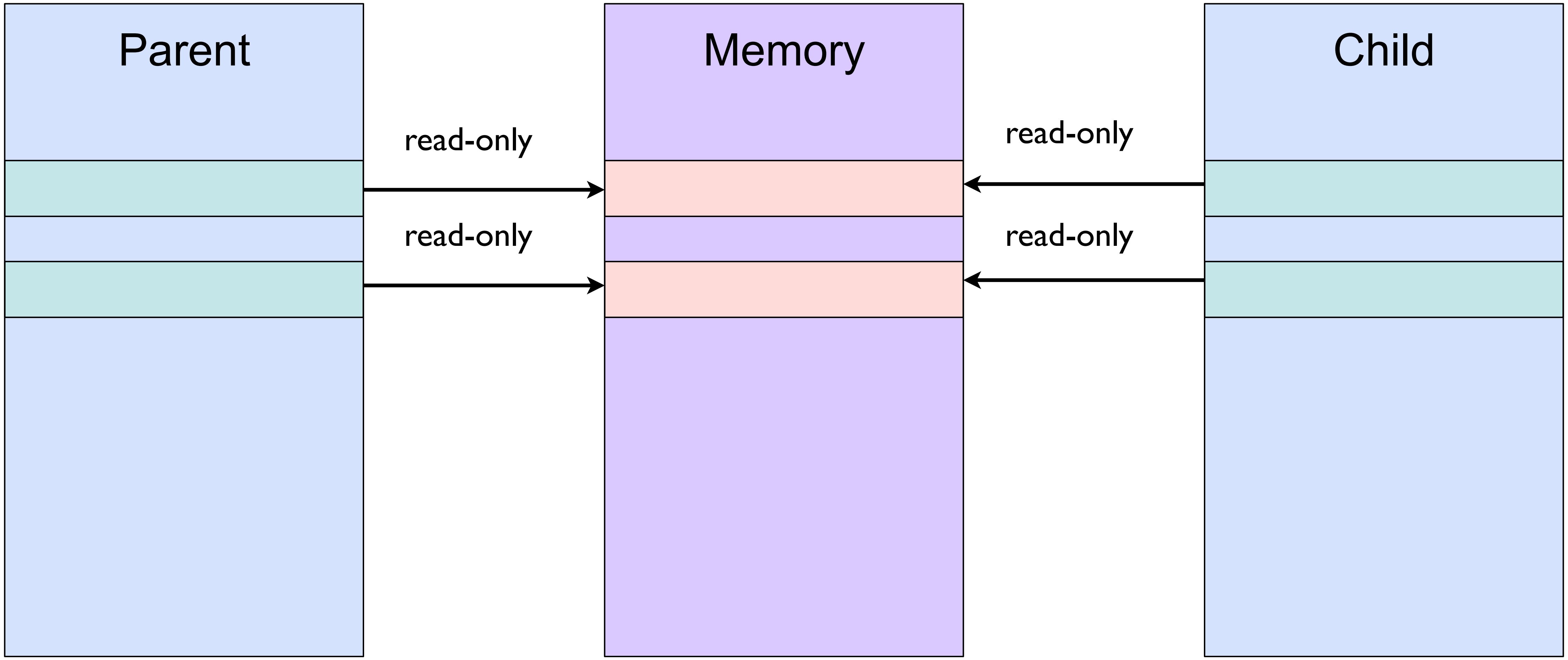
Copy on write

- Used throughout Linux
  - shared pages, fork



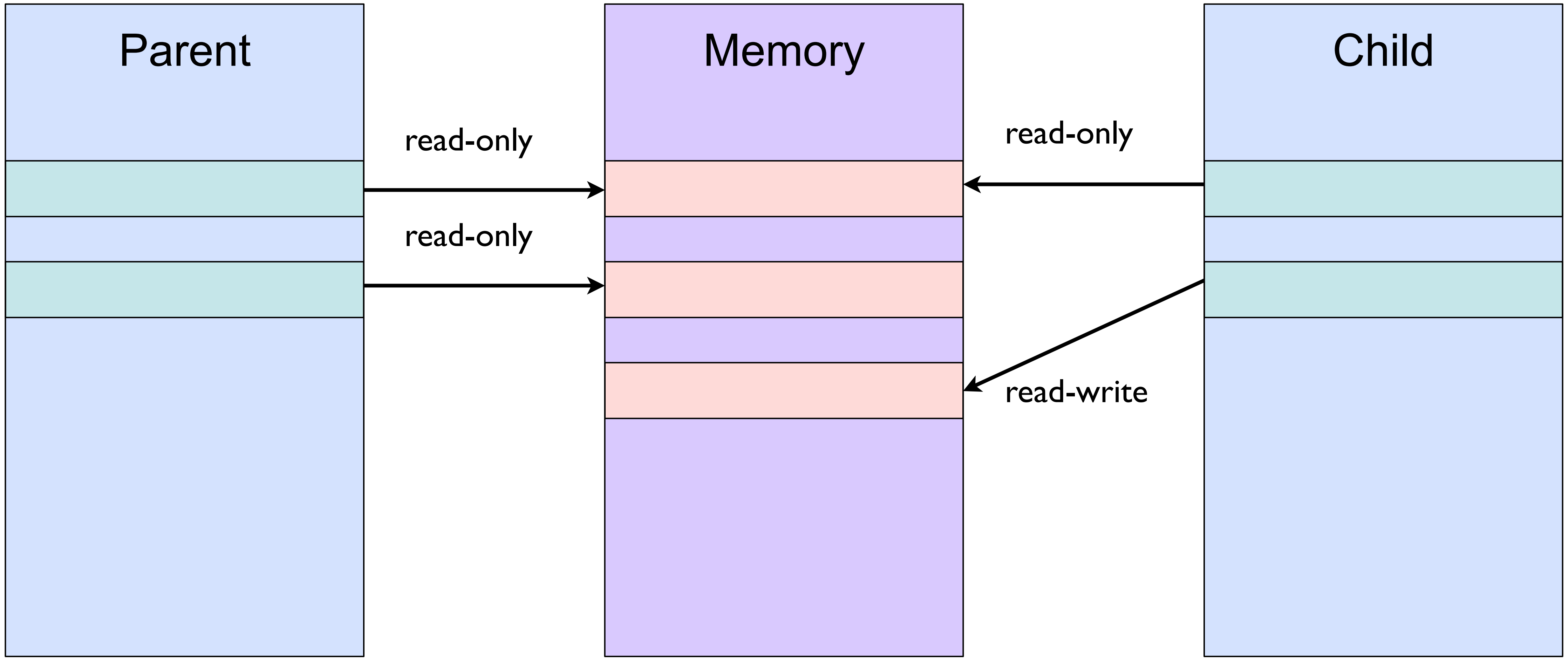
Copy on write

- Used throughout Linux
  - – shared pages, fork



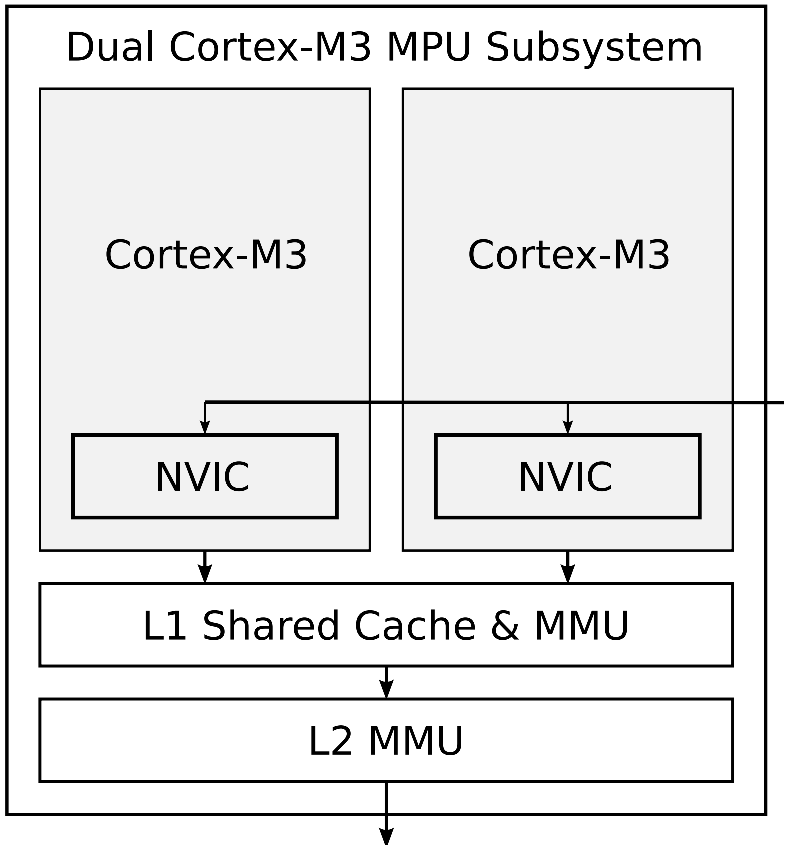
Copy on write

- Used throughout Linux
  - – shared pages, fork



Memory management on the Cortex-M3

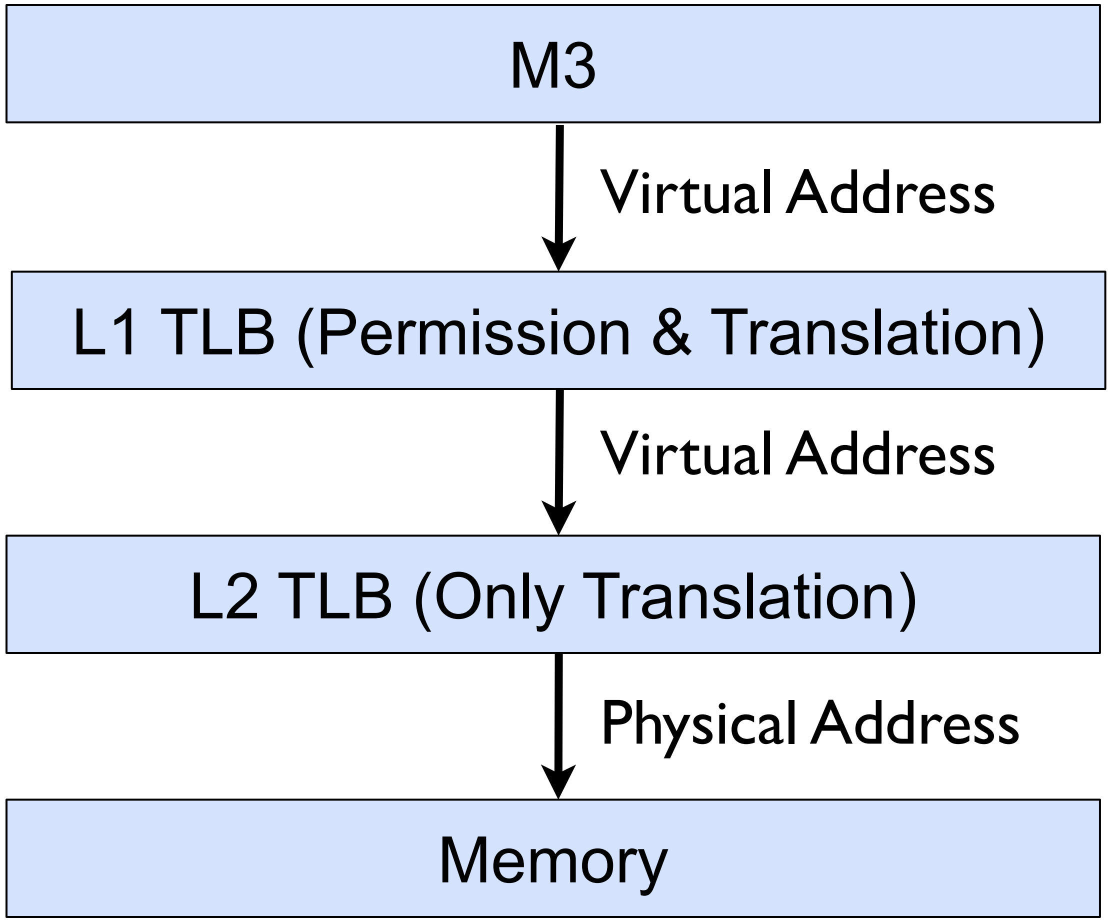
- Subsystem's shared MMUs
  - L1 shared cache & MMU
    - 10 entry TLB
    - read-only & execute-only permissions
    - software loaded
  - L2 MMU
    - 32 entry TLB
    - hardware walker (ARMv6 without permissions)



- Limitations
  - no supervisor-mode permissions - separate kernel page table
  - no tagged TLB - flush the TLB on every context switch

Read-only pages

- Hidden from L2 walker
  - marked invalid in the pagetable
  - causes a fault when access
- Manually loaded into L1
  - with correct permissions
  - no translation
- L2 kept in sync with the L1
  - MMUs in series, double translation
  - avoid L2 faulting



M3

Virtual Address

L1 TLB (Permission & Translation)

Virtual Address

L2 TLB (Only Translation)

Physical Address

Memory

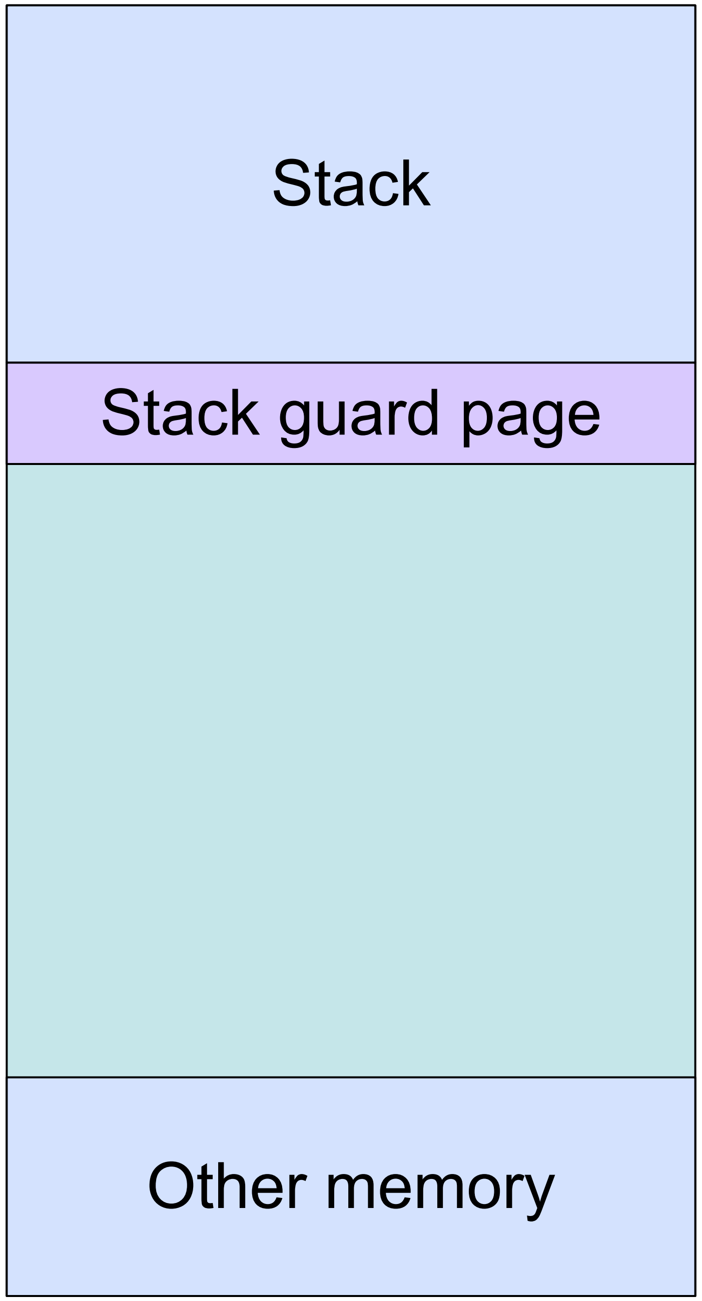
Exception handling on the Cortex-M3

Exception handling on the Cortex-M3

- M3 exception entry behaviour
  - core saves its state to memory pointed to by the current stack pointer

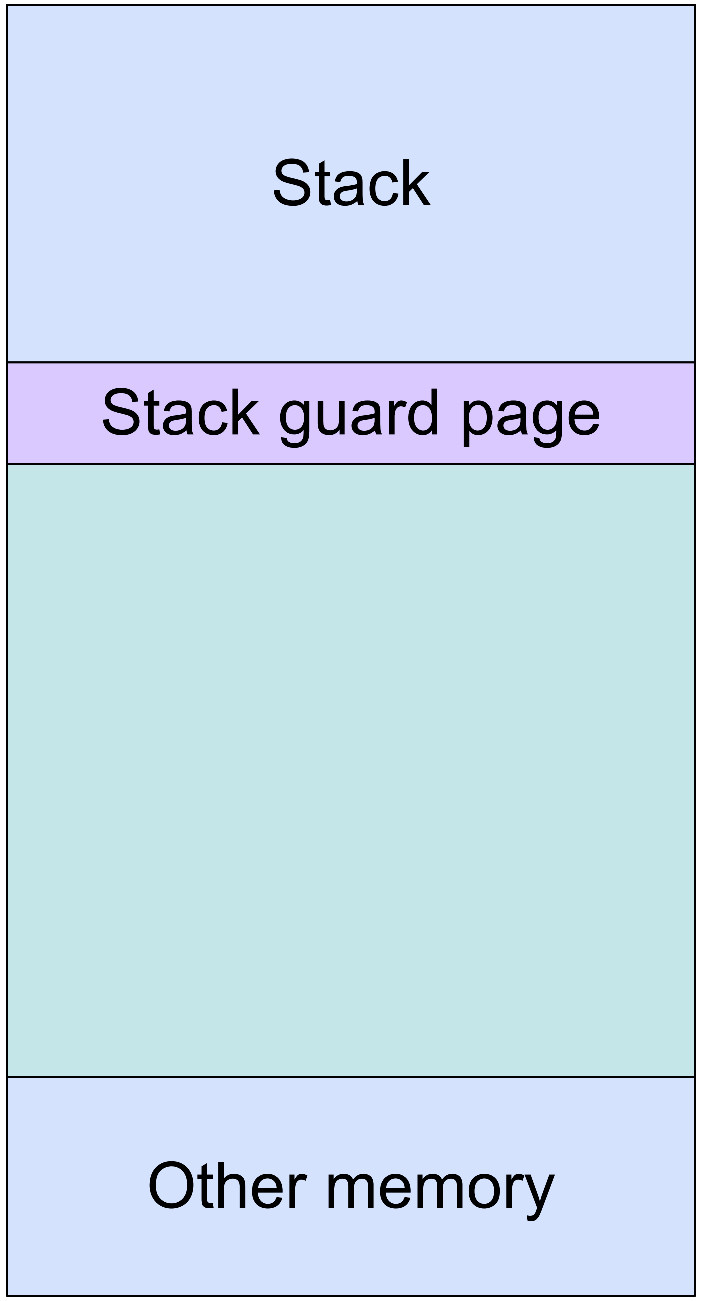
Exception handling on the Cortex-M3

- M3 exception entry behaviour
  - core saves its state to memory pointed to by the current stack pointer
- Dynamic stack allocation
  - access past the end of the stack results in a fault
  - kernel catches the fault, more stack is allocated



Exception handling on the Cortex-M3

- M3 exception entry behaviour
  - core saves its state to memory pointed to by the current stack pointer
- Dynamic stack allocation
  - access past the end of the stack results in a fault
  - kernel catches the fault, more stack is allocated
- Stack faults on M3 are unrecoverable
  - preallocate and pin entire stack
  - no dynamically resizing the stack



Toolchain for userspace applications

Toolchain for userspace applications

- Glibc does not produce pure Thumb-2
  - userspace-helpers
  - hand coded ARM assembly, e.g. memcpy implemented in ARM assembly

Toolchain for userspace applications

- Glibc does not produce pure Thumb-2
  - userspace-helpers
  - hand coded ARM assembly, e.g. memcpy implemented in ARM assembly
- Binutils
  - Procedure Linkage Table (PLT) used for dynamic binding shared libraries implemented with ARM

Toolchain for userspace applications

- Glibc does not produce pure Thumb-2
  - userspace-helpers
  - hand coded ARM assembly, e.g. memcpy implemented in ARM assembly
- Binutils
  - Procedure Linkage Table (PLT) used for dynamic binding shared libraries implemented with ARM
  - stick with static binaries for now

Linux now works on the M3 and supports userspace...

Linux now works on the M3 and supports userspace…

… beside an A9

Modifying Linux to support the A9s and M3s

- Unified model
  - performance overhead of migrations
- Hybrid model
  - no forced restriction of features
  - allow the user to interrogate system
- Restrictive model
  - restrict to subset of features
  - allows any process to run on any core

Implementing this in Linux

- Compiling for the subset of Thumb-2
- Producing a single image
- Synchronisation
- Supporting live migration
- Interrupts

Compiling for multiple architectures

Compiling for multiple architectures

- Compile a single image which can boot either A9 or M3

Compiling for multiple architectures

- Compile a single image which can boot either A9 or M3
- Patched binutils
    - compile C to common subset of Thumb-2

Compiling for multiple architectures

- Compile a single image which can boot either A9 or M3
- Patched binutils
  - compile C to common subset of Thumb-2
  - allow for both architecture's special register/co-processor instructions
    - cp15 (A9 co-processors for system control, cache, MMU)
    - PRIMASK, FAULTMASK, BASEPRI (M3 mask registers)

Single kernel image

Single kernel image

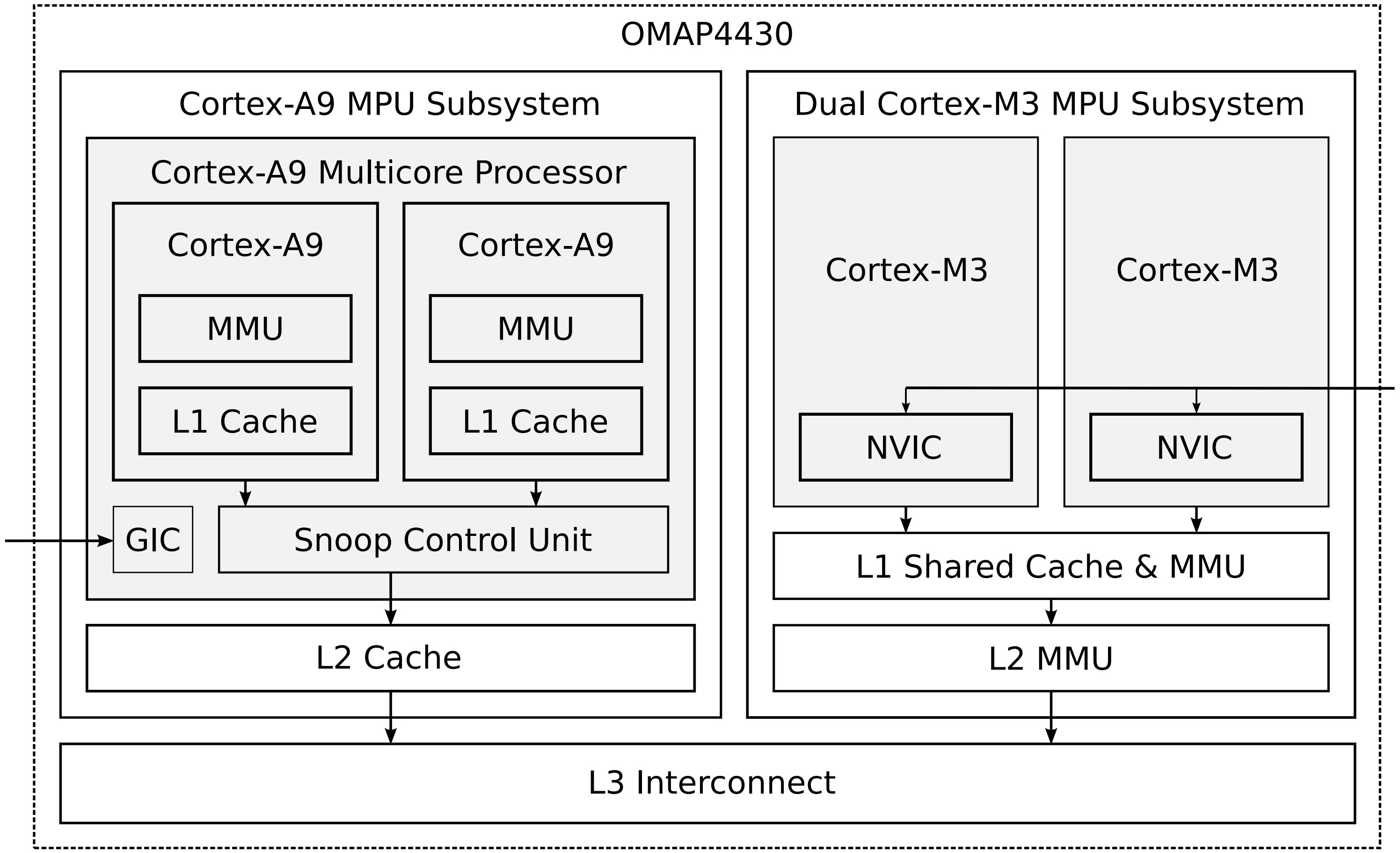
- ARM Linux can be compiled for multiple processors
  - multiple processor abstraction (proc_info)
    - allows compiling support for multiple processors into a single image
  - extended to incorporate architectural (system ISA) differences
    - interrupt enabling/disabling, co-processors, exception handling

Single kernel image

- ARM Linux can be compiled for multiple processors
  - – multiple processor abstraction (proc_info)
    - allows compiling support for multiple processors into a single image
  - – extended to incorporate architectural (system ISA) differences
    - interrupt enabling/disabling, co-processors, exception handling
- Running the kernel on both A9 and M3
  - – per core (A9/M3) MMU mapping for proc_info struct, each core can see its own functions

Synchronisation

- Cross-subsystem synchronisation
  - locks are implemented using an atomic operation
  - ARM's *exclusive monitor* won't work (LDREX, STREX)
  - implement synchronisation primitives with hardware spin-locks

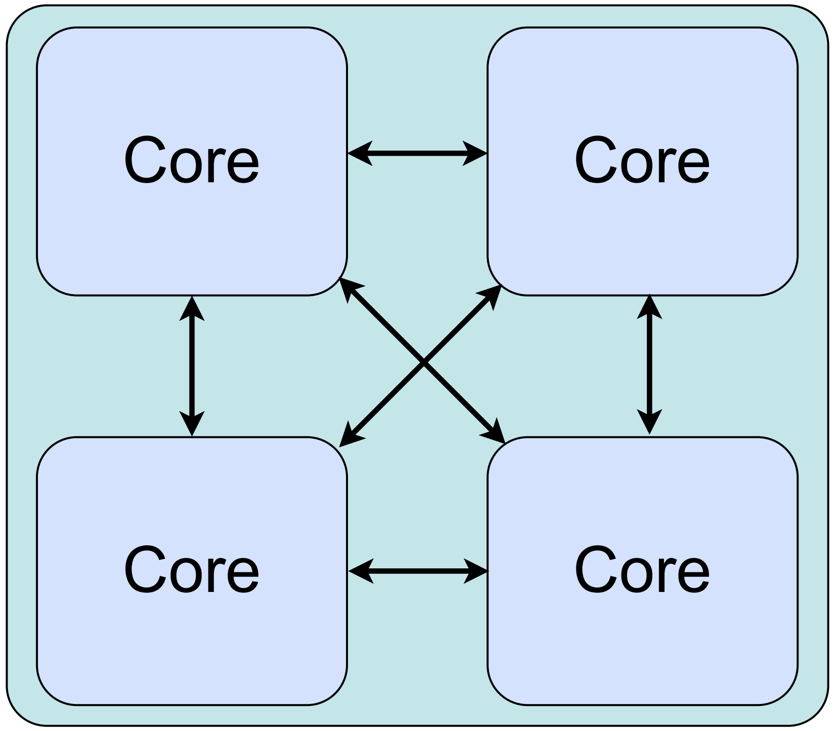


Synchronisation

- Inter-processor interrupts
  - trigger, signal completion
  - no direct interrupts between A9 and M3
  - OMAP's mailbox

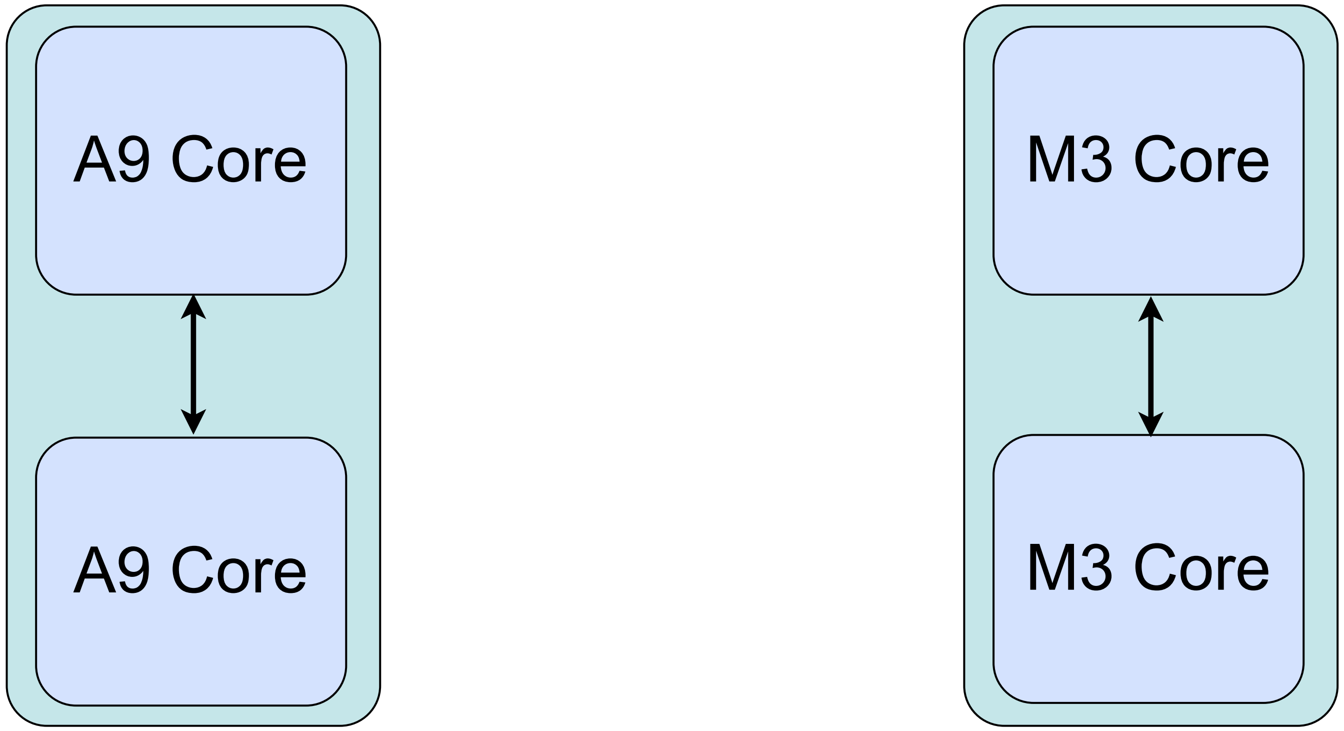
Synchronisation

- Inter-processor interrupts
  - trigger, signal completion
  - no direct interrupts between A9 and M3
  - OMAP's mailbox



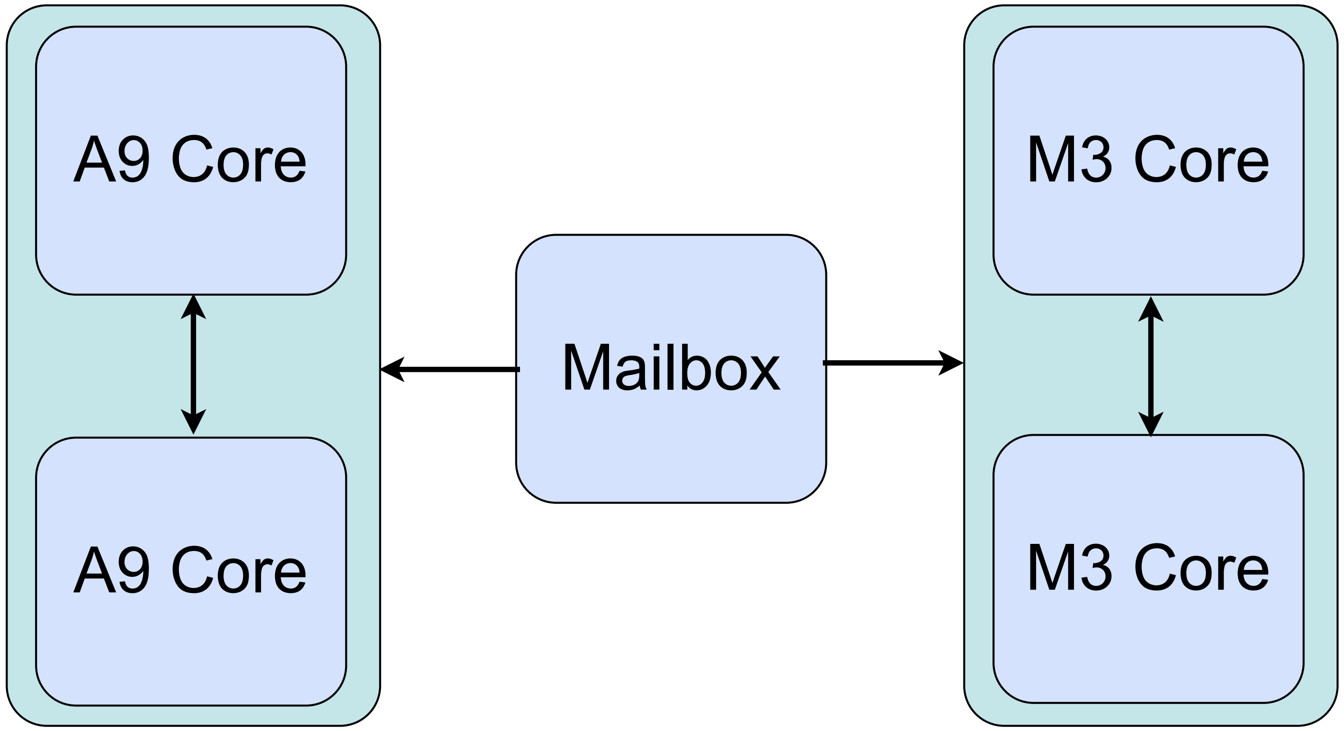
Synchronisation

- Inter-processor interrupts
  - trigger, signal completion
  - no direct interrupts between A9 and M3
  - OMAP's mailbox



Synchronisation

- Inter-processor interrupts
  - trigger, signal completion
  - no direct interrupts between A9 and M3
  - OMAP's mailbox



Supporting migration

Supporting migration

- Page-tables
  - both subsystems use ARMv6 pagetable format
  - read-only mapping must be invalidated when running on the M3

Supporting migration

- Page-tables
  - both subsystems use ARMv6 pagetable format
  - read-only mapping must be invalidated when running on the M3
- Exception handling
  - manipulate status registers and saved registers to consistent format
  - return to the correct exception return path

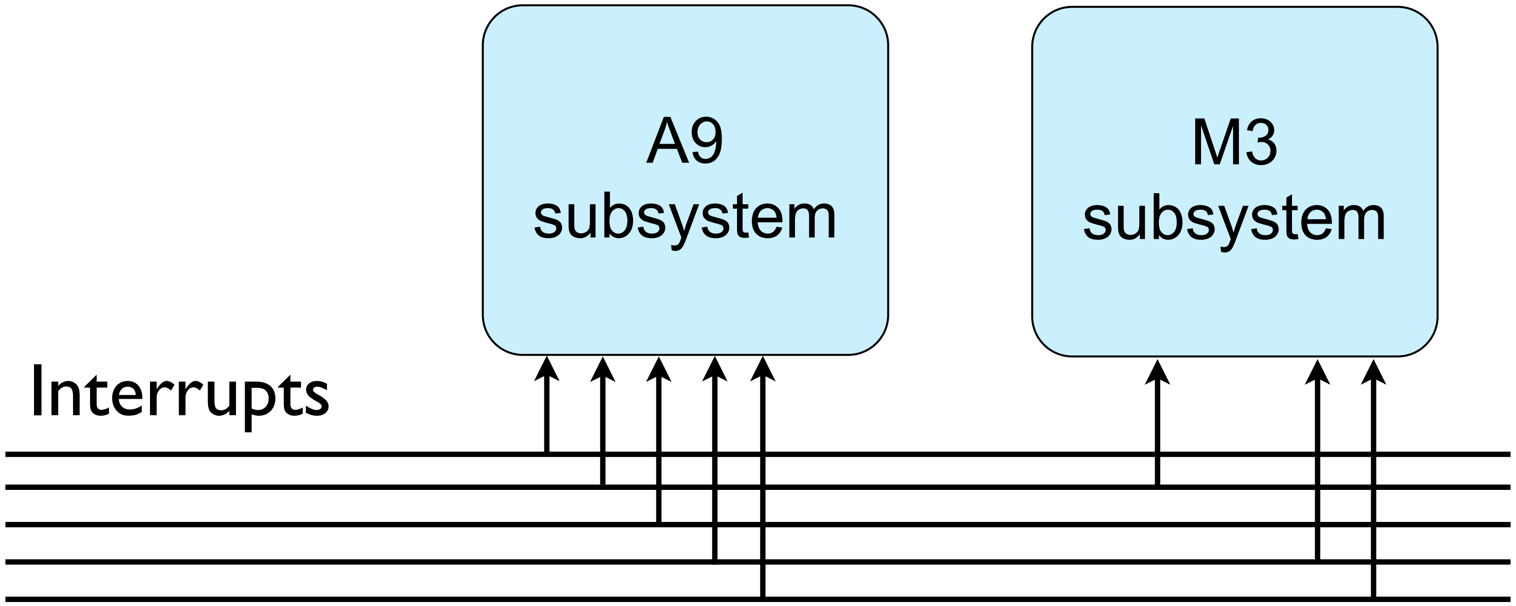
Supporting migration

- Page-tables
  - both subsystems use ARMv6 pagetable format
  - read-only mapping must be invalidated when running on the M3
- Exception handling
  - manipulate status registers and saved registers to consistent format
  - return to the correct exception return path
- Live migration
  - taskset, sched_setaffinity

Interrupts

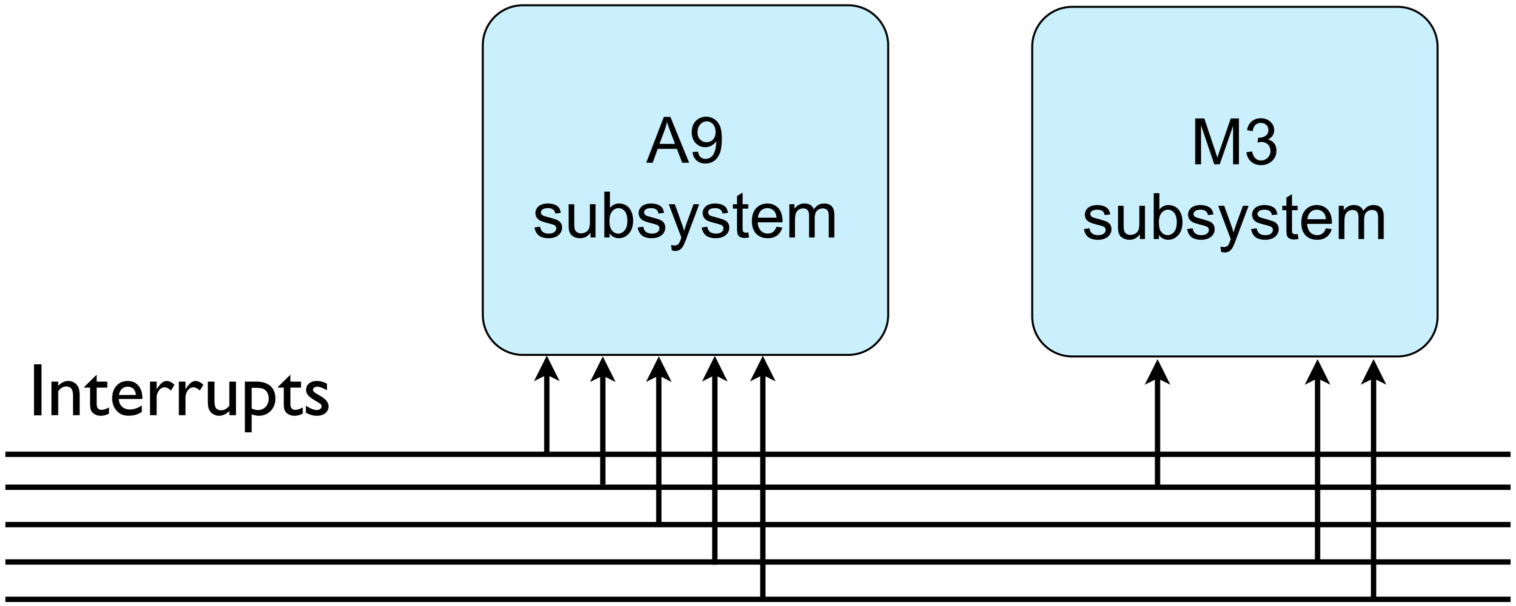
Interrupts

- Interrupt controller
  - the M3 and A9 cannot access each other's interrupt controllers
  - masking certain interrupts can only be done on certain cores
  - interrupts are not easily distributed, some interrupts are not mapped to the M3 subsystem



Interrupts

- Interrupt controller
  - the M3 and A9 cannot access each other's interrupt controllers
  - masking certain interrupts can only be done on certain cores
  - interrupts are not easily distributed, some interrupts are not mapped to the M3 subsystem



- This means that Linux can not completely run on the M3.

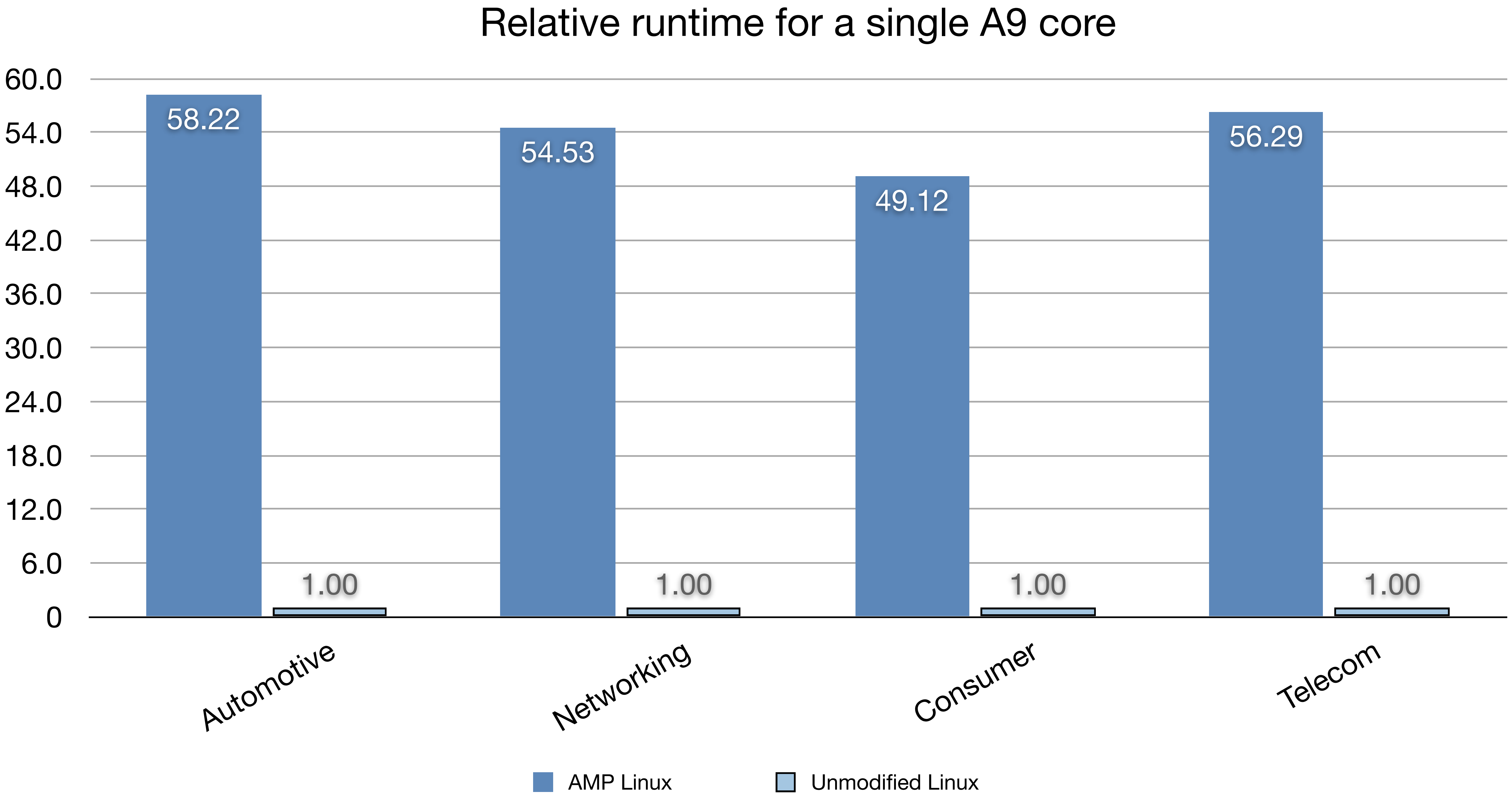
Now, Linux runs with both the M3 and A9, and we can migrate tasks between them!

Awesome! But what about performance?

- Investigate the overheads of our changes
- EEMBC
  - embedded benchmarking suite
  - wide range of workloads
    - automotive
    - telecommunications
    - networking
    - ‘consumer’

Awesome! But what about performance?

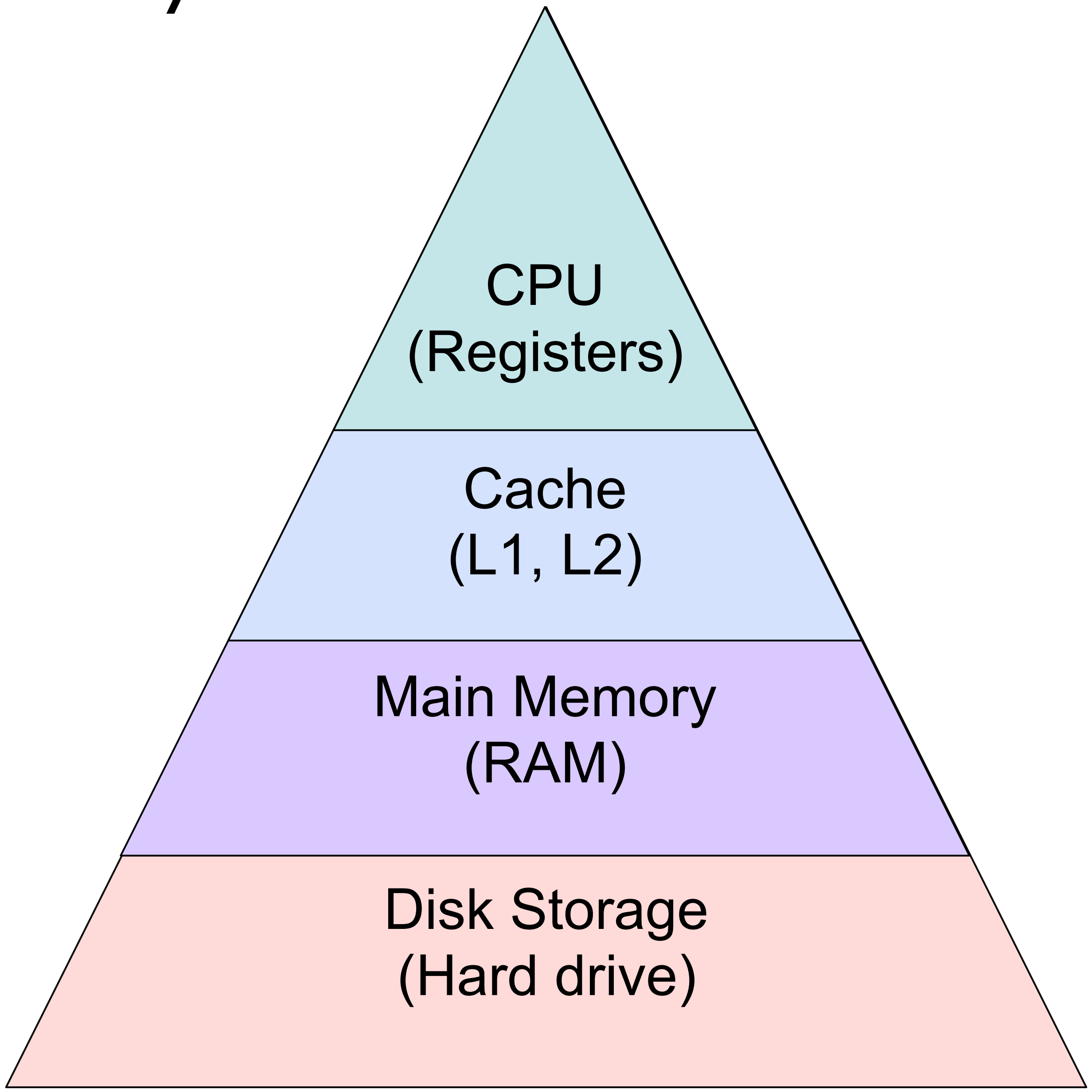
NICTA



- Performance is *really* bad

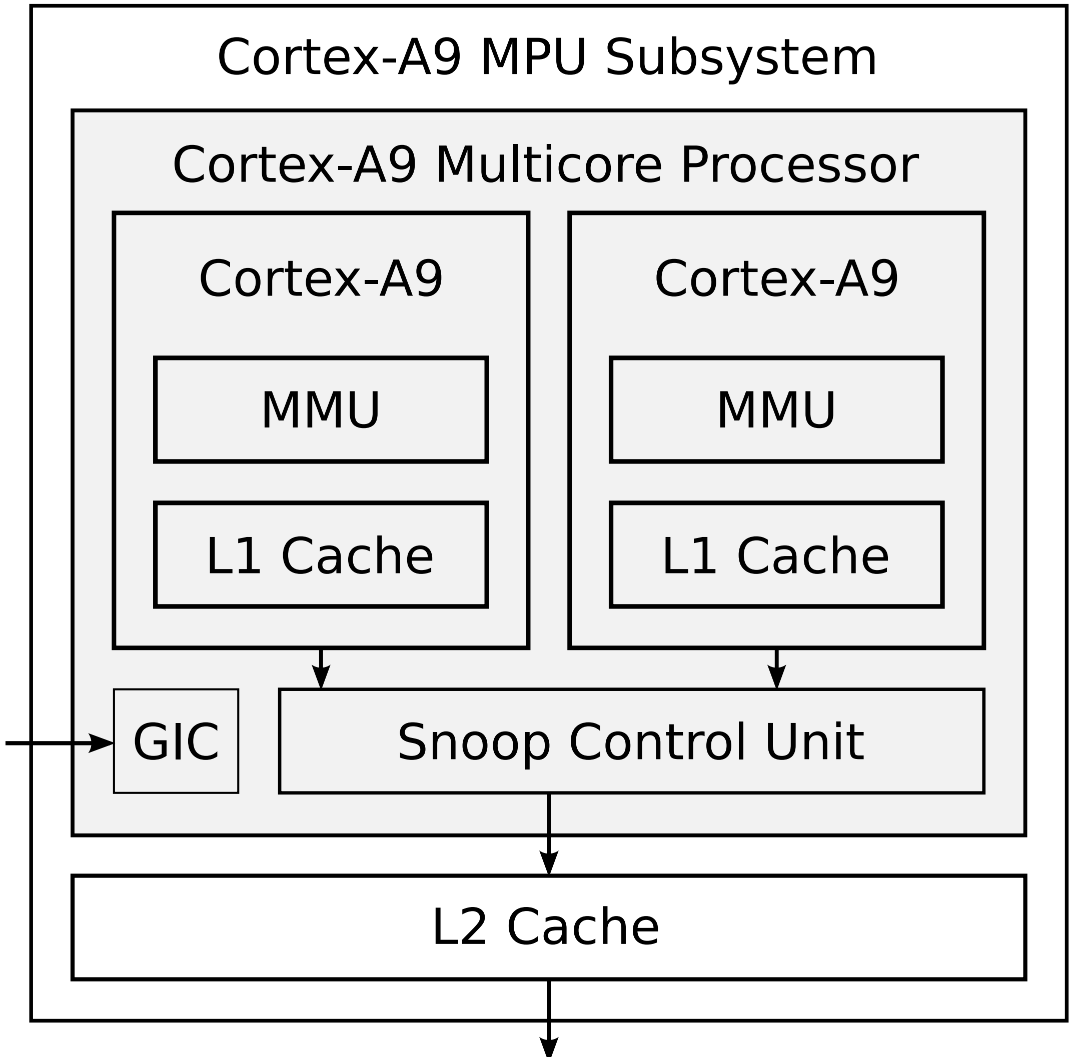
Caches are off!

- Caches provide improved latency



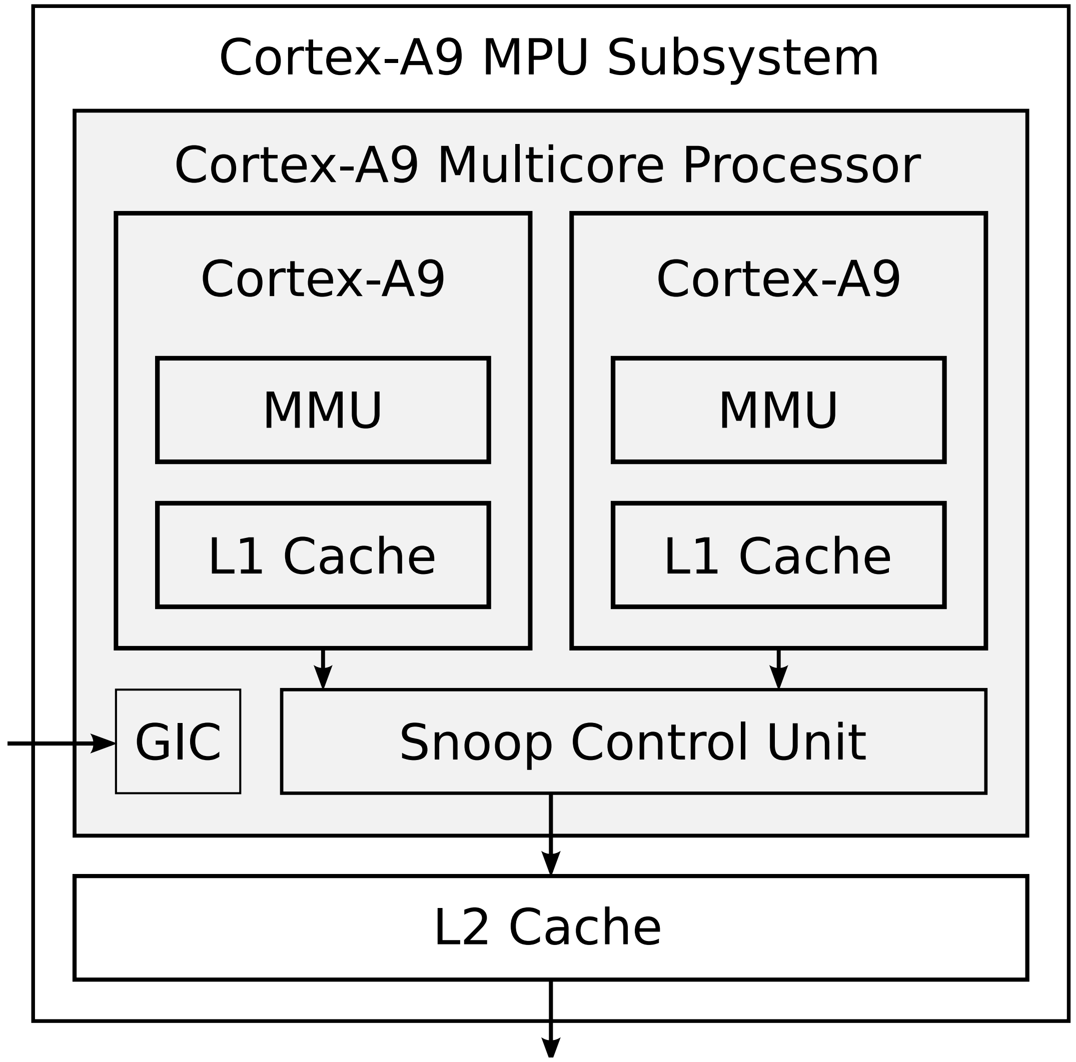
Caching on a multi-core system

- Shared caches
  - high latency
  - shared memory
- Local caches
  - low latency
  - less cache per CPU
  - cache coherency issues
- Combination

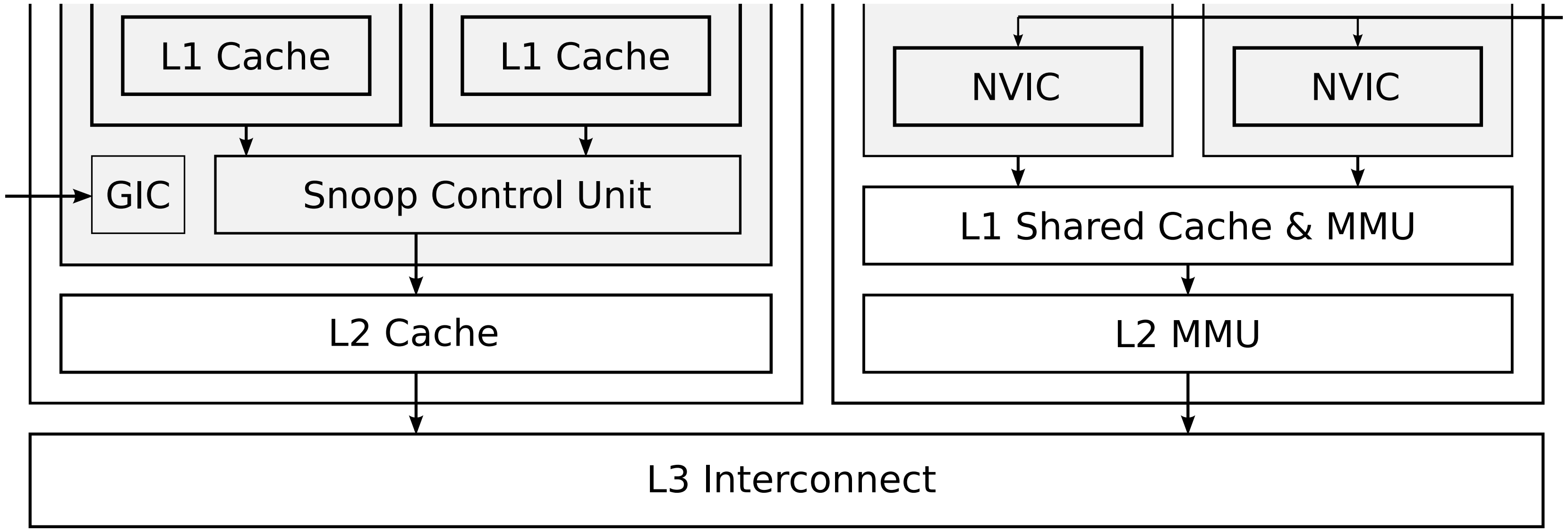


Cache coherency

- Sharing memory
  - out of date data
  - notify other cores of changes to data
- Cache coherency protocols
  - snooping (MOESI protocol)
    - requires hardware support

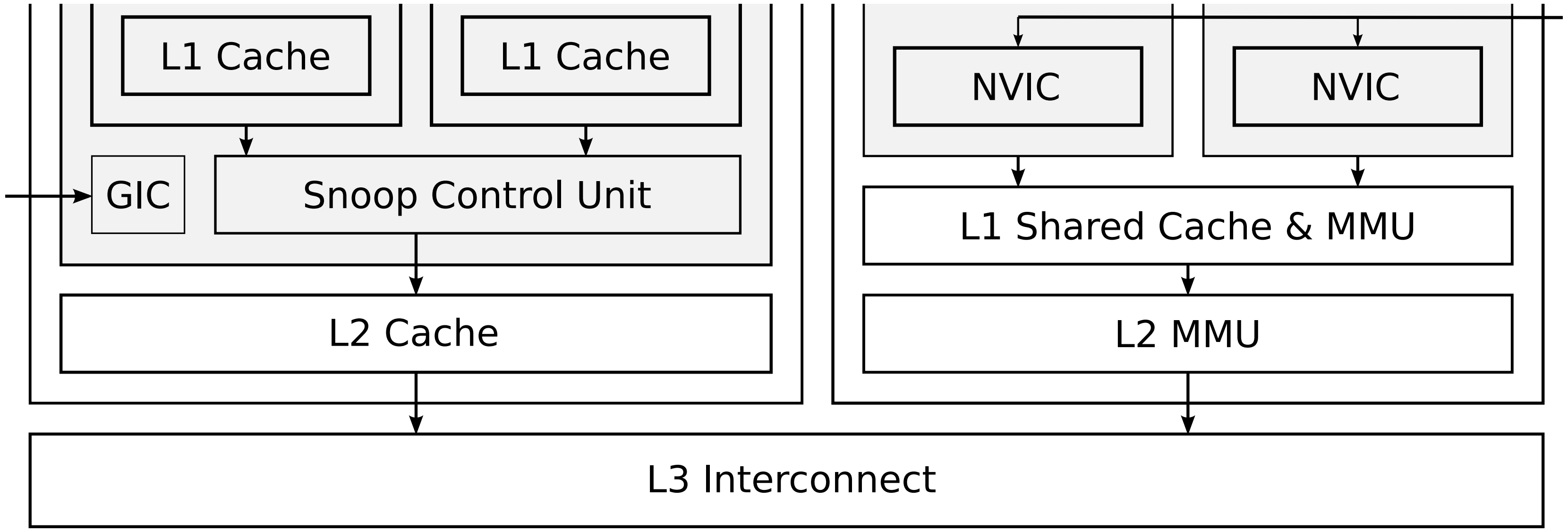


Enabling caches



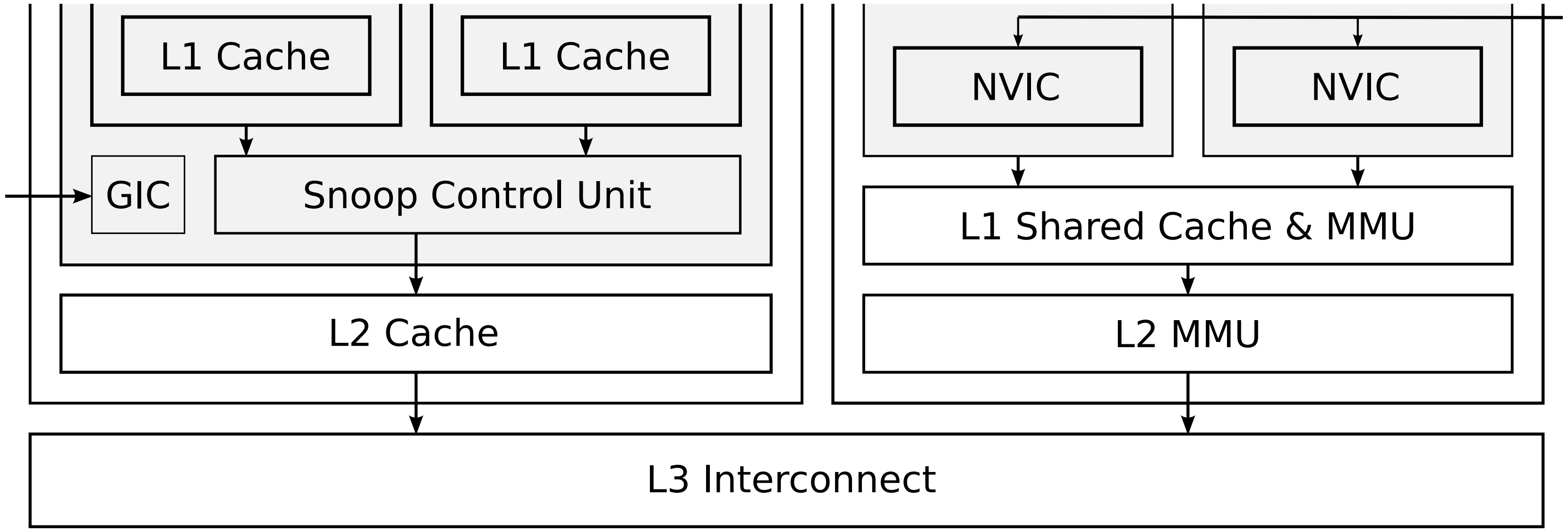
Enabling caches

- No shared cache between the A9 and M3s
  - sharing must occur at main memory



Enabling caches

- No shared cache between the A9 and M3s
  - – sharing must occur at main memory
- No hardware support for cross-subsystem cache coherency
  - – efficient cache coherency requires hardware support (snooping)



Enabling caches

Enabling caches

- Big lock based coherency - introducing Linux 2.0!

Enabling caches

- Big lock based coherency - introducing Linux 2.0!
  - – restrict to one CPU in the kernel (lots of waiting)

Enabling caches

- Big lock based coherency - introducing Linux 2.0!
  - – restrict to one CPU in the kernel (lots of waiting)
  - – flush all caches on acquire/release (lots of flushing)

Enabling caches

- Big lock based coherency - introducing Linux 2.0!
  - restrict to one CPU in the kernel (lots of waiting)
  - flush all caches on acquire/release (lots of flushing)
  - interrupt for signalling contention

Now we have caching!

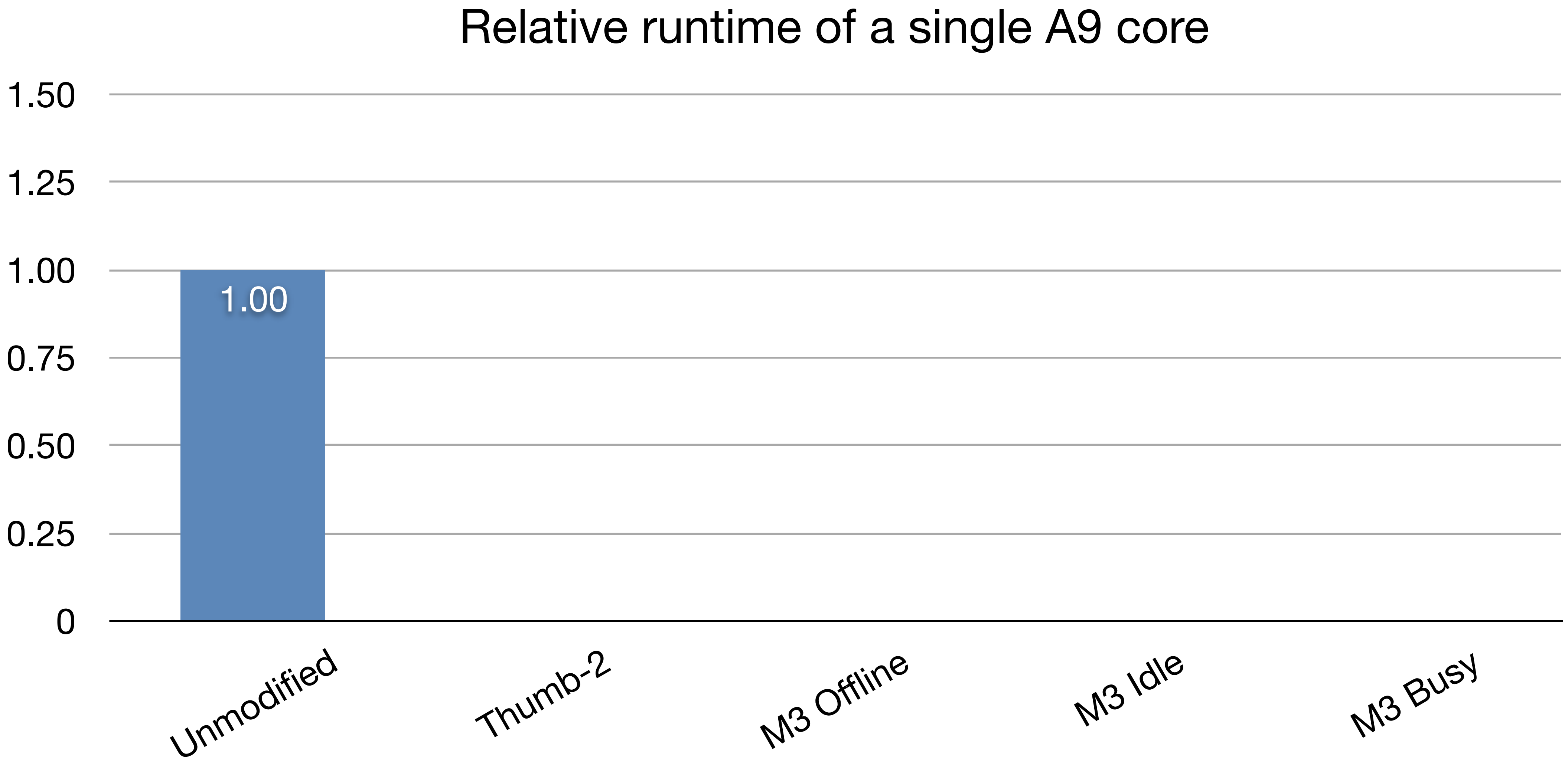
Lets ignore the BKL for now ;-)

Overhead

- Compare performance of **just** an A9 core
- Vary what the M3 core is doing

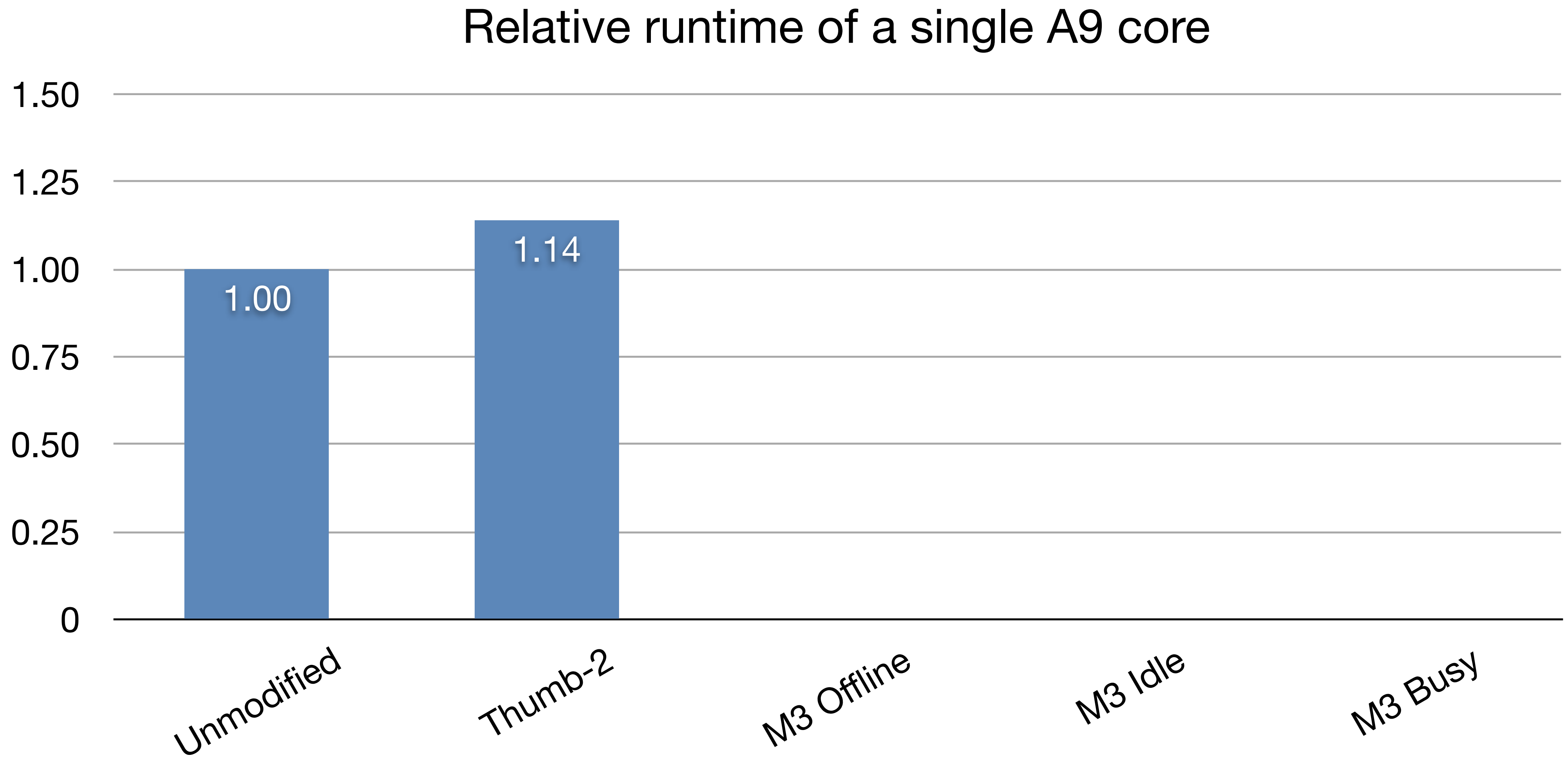
Overhead

- Compare performance of **just** an A9 core
- Vary what the M3 core is doing



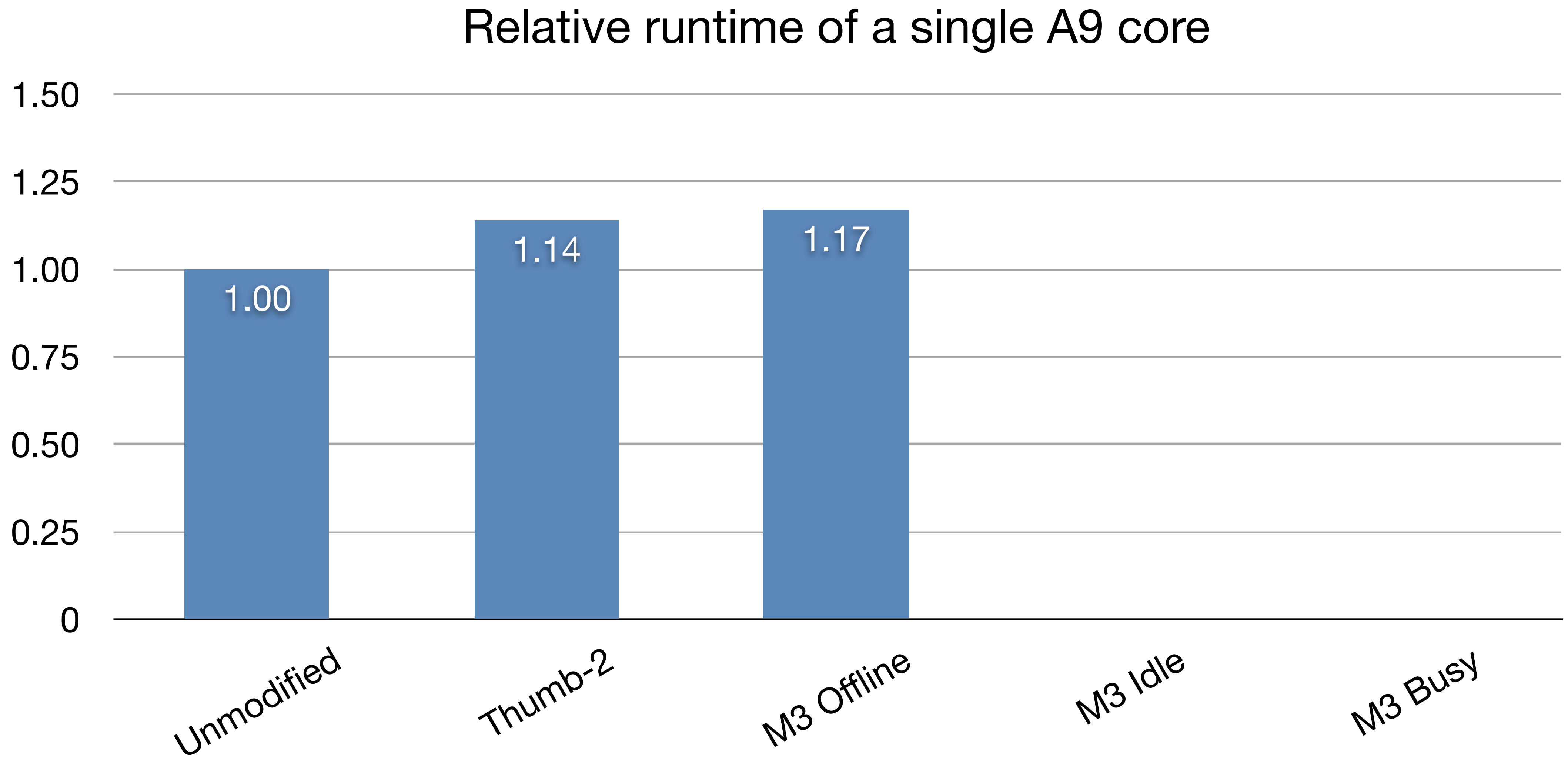
Overhead

- Compare performance of **just** an A9 core
- Vary what the M3 core is doing



Overhead

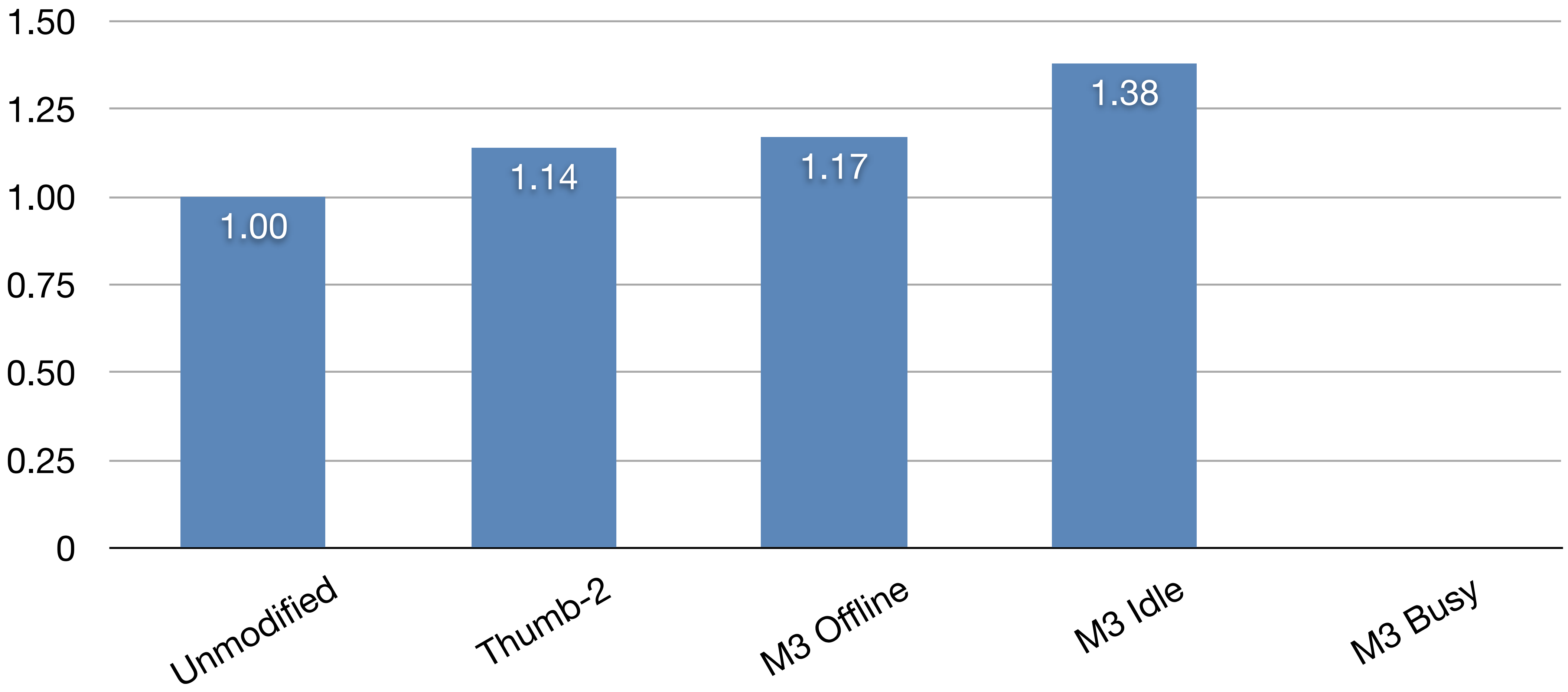
- Compare performance of **just** an A9 core
- Vary what the M3 core is doing



Overhead

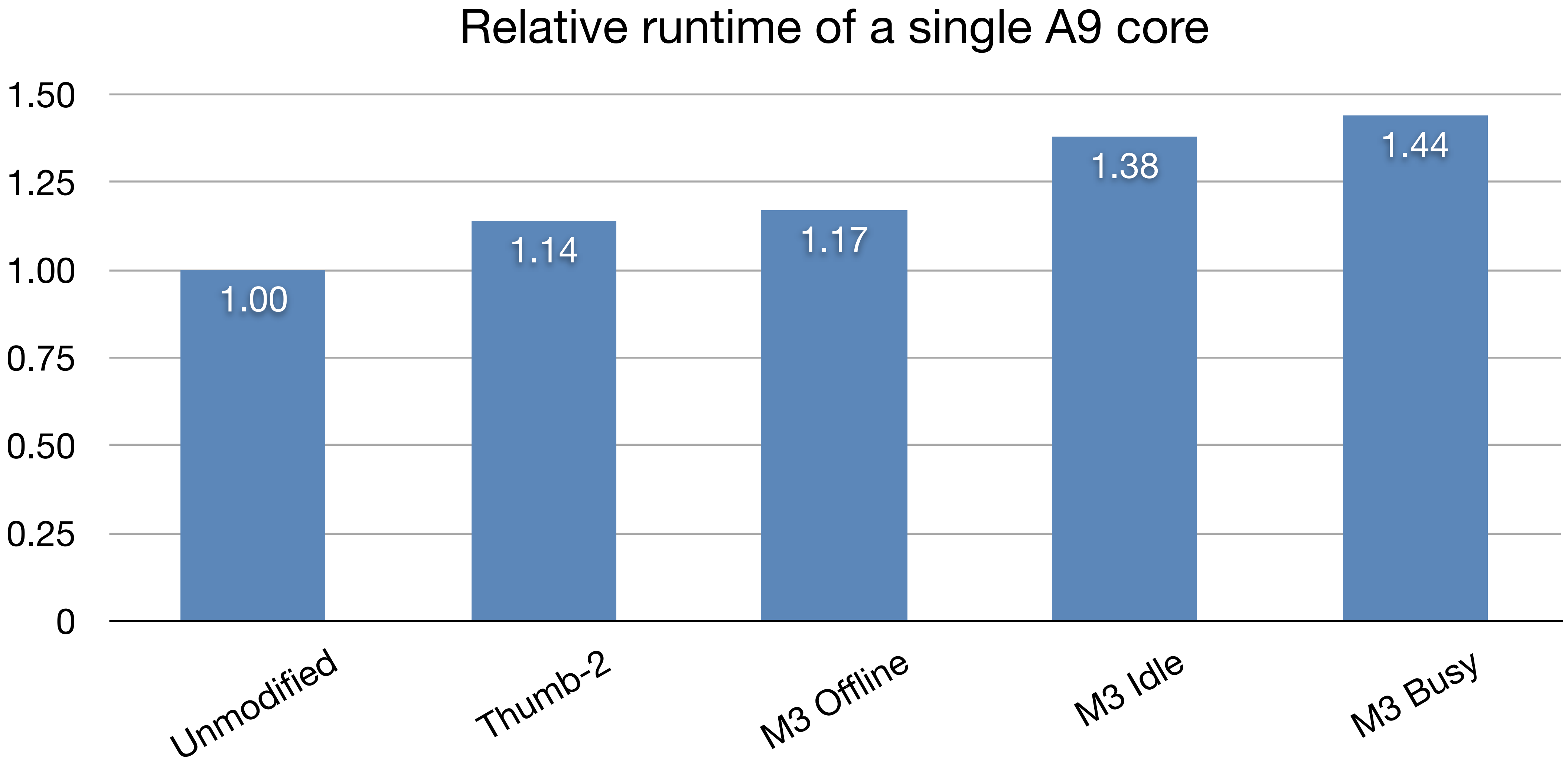
- Compare performance of **just** an A9 core
- Vary what the M3 core is doing

Relative runtime of a single A9 core



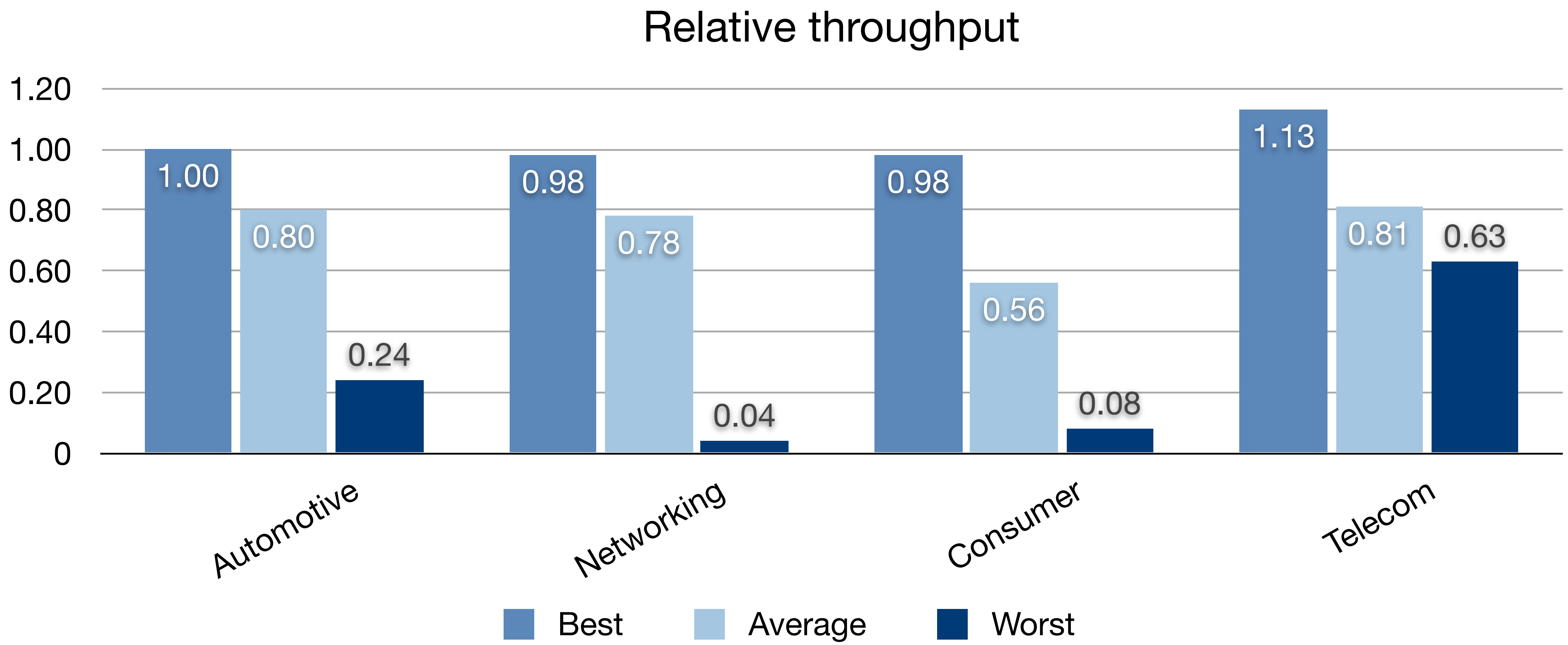
Overhead

- Compare performance of **just** an A9 core
- Vary what the M3 core is doing

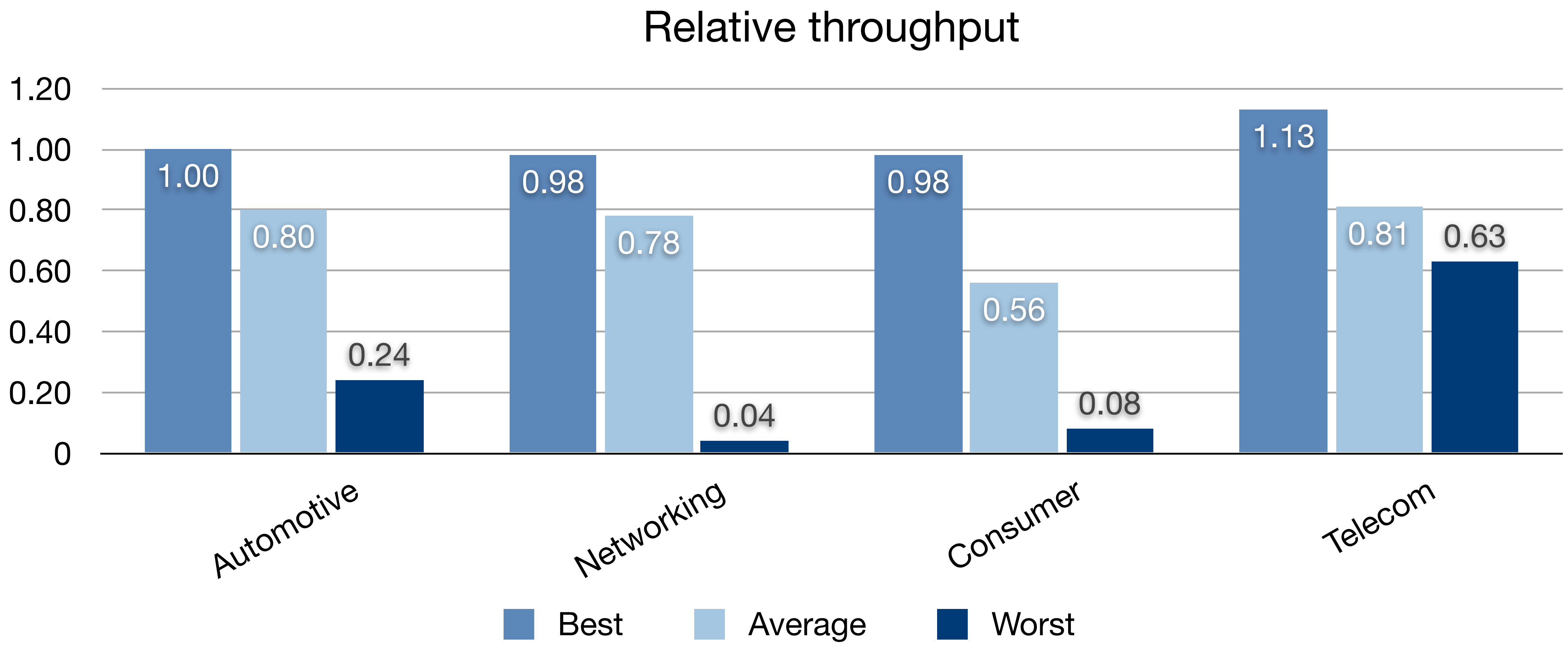


System throughput

NICTA



System throughput

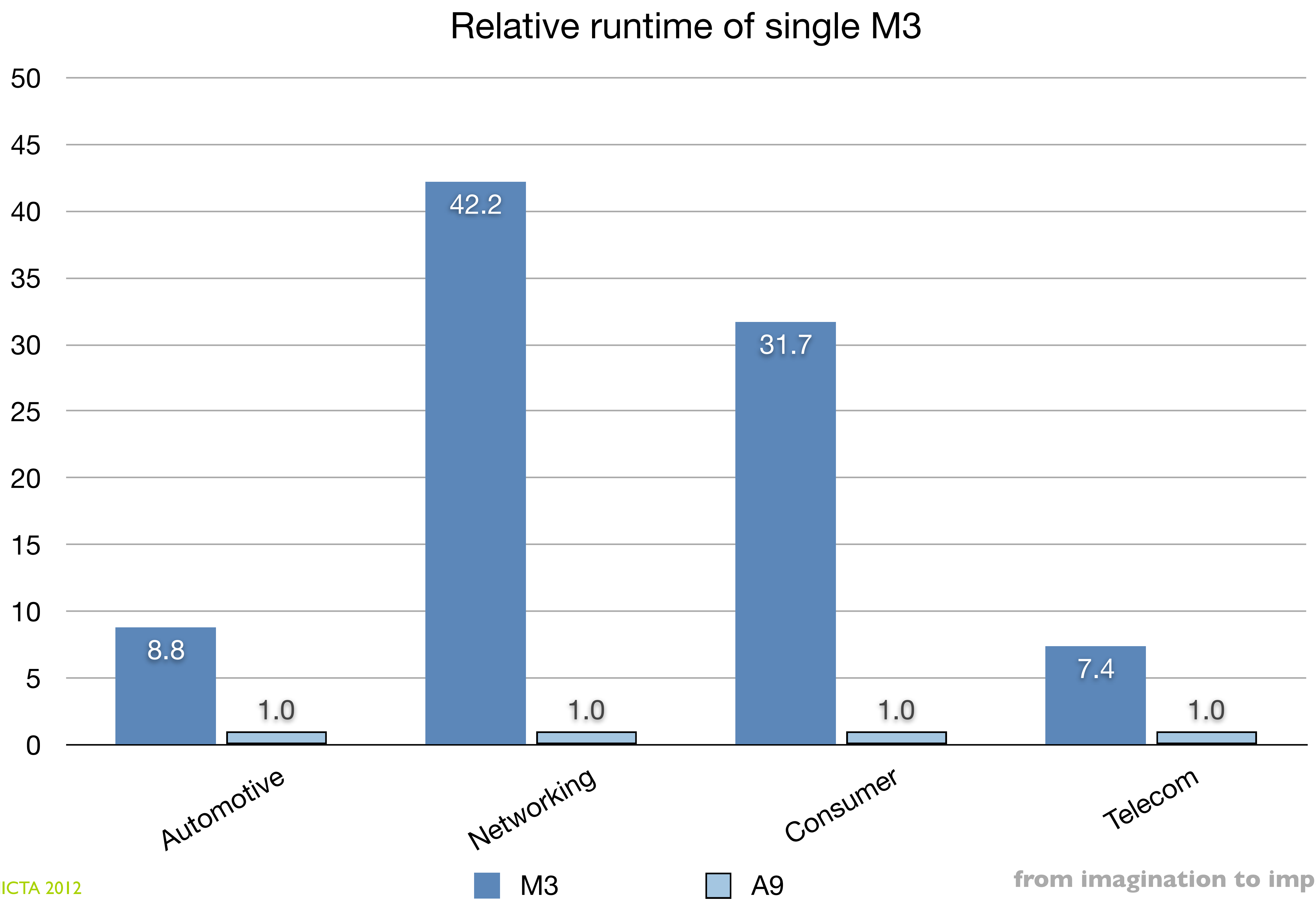


- Performance is still not great
  - M3 doesn't make up for overheads
  - worst case due to high L1 TLB (software loaded) miss rate, as the M3 spends most of its time refilling the L1 TLB, locking the A9 out of the kernel

Using the system

M3 vs A9

NICTA



Can the system be used in its current state?

Can the system be used in its current state?

- Can energy-efficiency be improved by using the M3s?
  - – performance overheads negate any savings

Can the system be used in its current state?

- Can energy-efficiency be improved by using the M3s?
  - performance overheads negate any savings
- How can the system know how each core will perform?
- How can scheduling decisions be made?

Modelling the performance of each core

Modelling the performance of each core

- Run on the A9 for a short time, measure some predictors
  - cache misses
  - instructions executed
  - branches correctly predicted
  - TLB miss rate *
  - etc

Modelling the performance of each core

- Run on the A9 for a short time, measure some predictors
  - – cache misses
  - – instructions executed
  - – branches correctly predicted
  - – TLB miss rate *
  - – etc
- Plug these numbers into a model

$$T_{\mathrm{M3}} = \alpha T_{\mathrm{A9}} + \gamma_0 C_0 + \cdots + \gamma_n C_n$$

Modelling the performance of each core

- Run on the A9 for a short time, measure some predictors
  - cache misses
  - instructions executed
  - branches correctly predicted
  - TLB miss rate *
  - etc
- Plug these numbers into a model

$$T_{\mathrm{M3}} = \alpha T_{\mathrm{A9}} + \gamma_0 C_0 + \cdots + \gamma_n C_n$$

- Decide whether it’s worth migrating...

Modelling the performance of each core

- Run on the A9 for a short time, measure some predictors
  - – cache misses
  - – instructions executed
  - – branches correctly predicted
  - – TLB miss rate *
  - – etc
- Plug these numbers into a model

$$T_{\mathrm{M3}} = \alpha T_{\mathrm{A9}} + \gamma_0 C_0 + \cdots + \gamma_n C_n$$

- Decide whether it's worth migrating...
- Prediction is within about 10% error for a wide range of workloads from EEMBC

Conclusion

- Linux can now schedule tasks on both A9 or M3 cores
  - overheads are high mostly due to lack of hardware support
  - with a bit of support from the hardware, the system should be usable
- With the right counters, performance prediction is accurate
  - again, hardware support would help, either provide performance counters on the M3s or better performance counters on the A9s.
- It only took 8500 lines to do.
  - No, we haven't pushed it upstream.
  - If you're interested in the details, look out for a potential Usenix ATC publication - fingers crossed.

Questions?

Questions?

NICTA

From **imagination** to **impact**

NICTA

From **imagination** to **impact**

Expected Questions

- Will we push the changes upstream?
  - a lot of changes to linux for not much gain atm.
  - very specific to OMAP4430, which is not very useful.