



Provable Security and Safety

The seL4 Microkernel and its Use in Critical Systems

Gernot Heiser | gernot.heiser@data61.csiro.au | @GernotHeiser

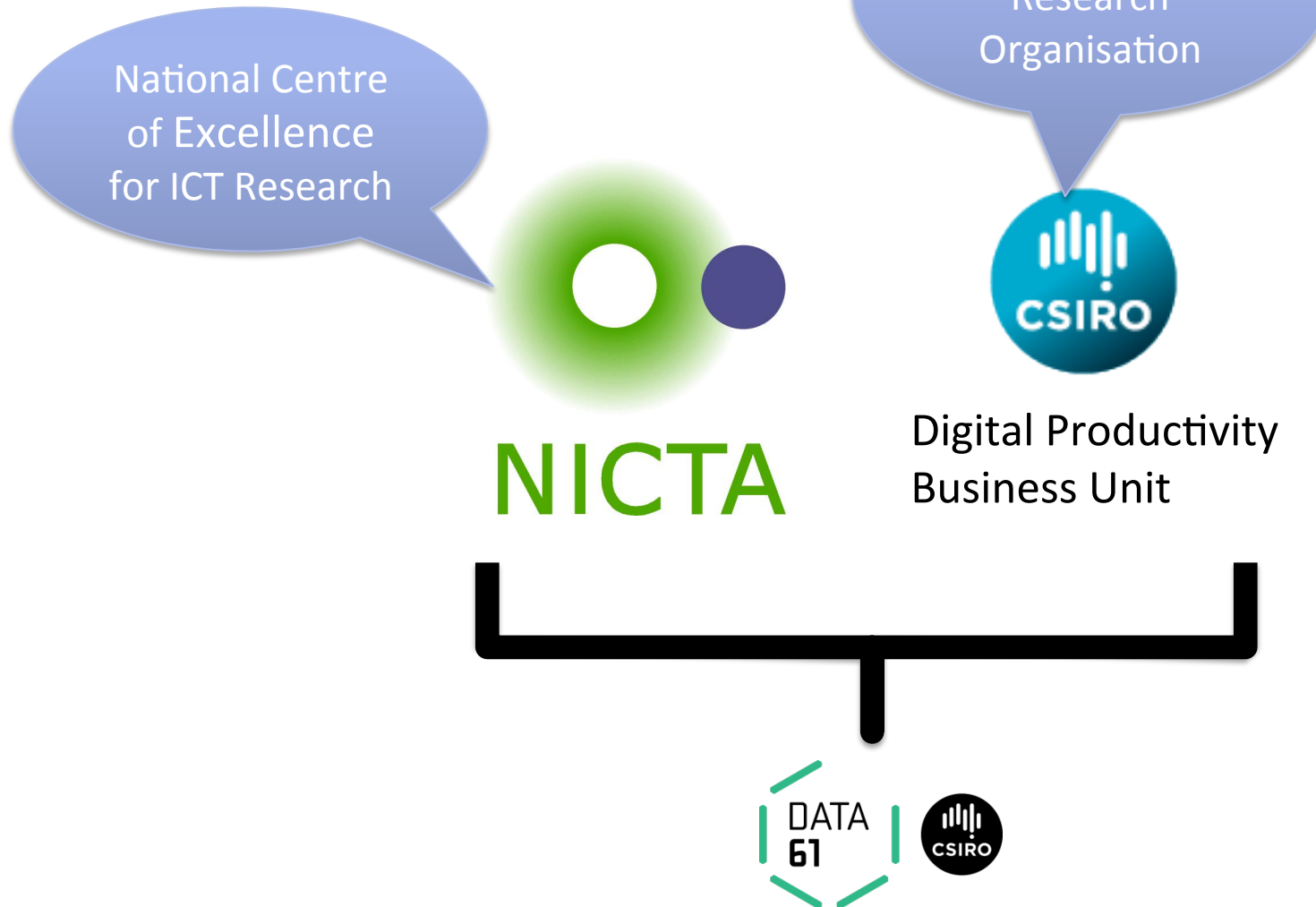
<http://microkernel.dude.wordpress.com>

February 2016

<https://trustworthy.systems/>



FAQ: What is Data61?



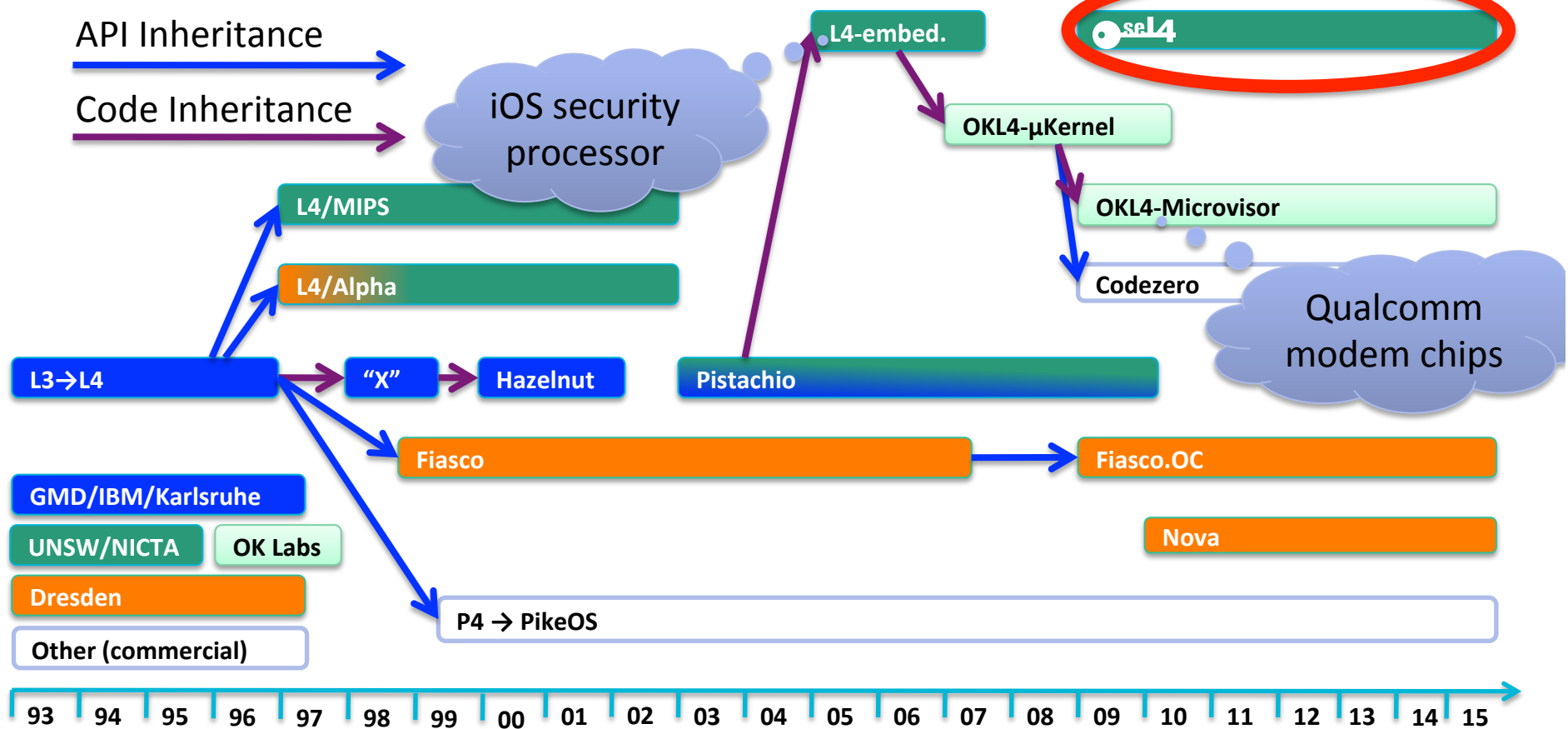
Mesa, AZ, 24 July 2015



se14 Inside!

L4 Family Tree

seL4: The latest (and most advanced) member of the L4 microkernel family – 20 years of history and experience



What is seL4?



seL4: The world's most (only?) **secure**
OS kernel – provably!

GPLed
2014-07-29

Philosophy Underlying seL4

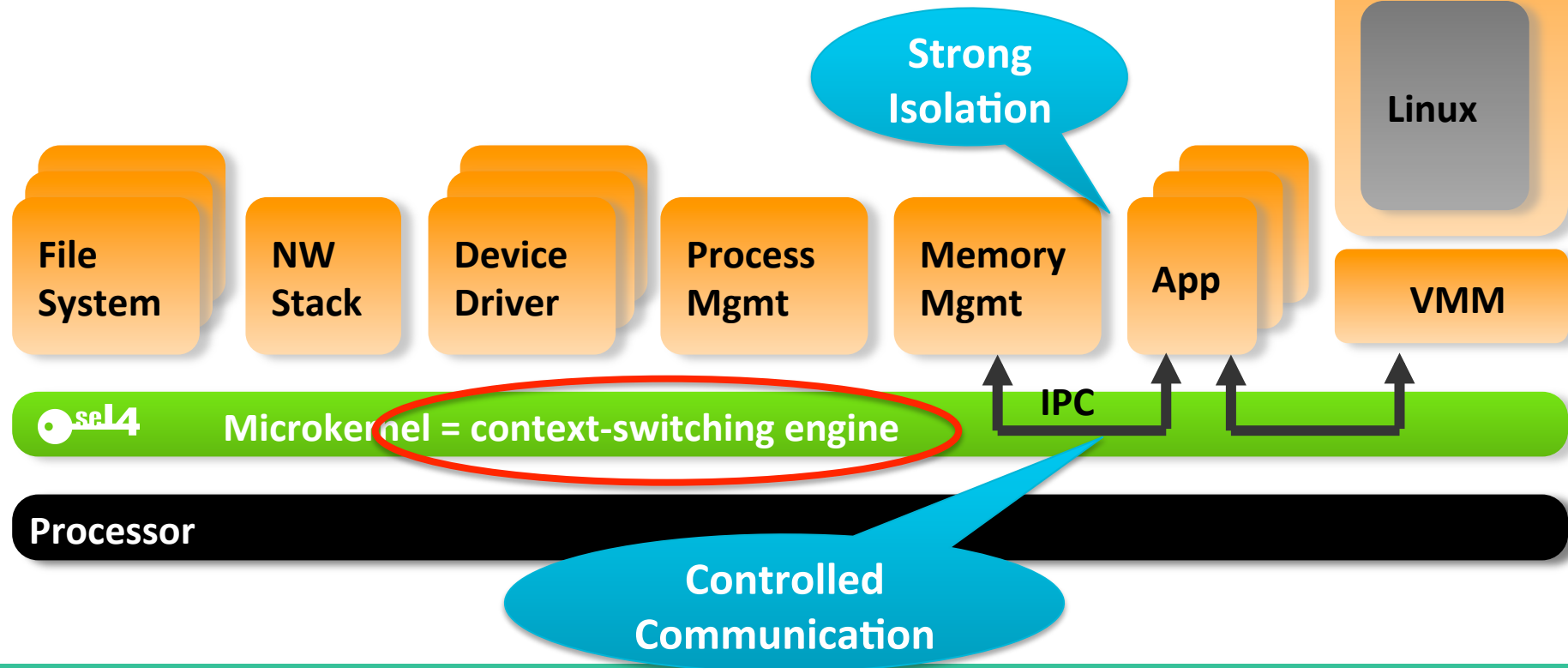


1. Security is paramount and drives design
2. Security is no excuse for bad performance
3. General-purpose platform for wide range of use cases

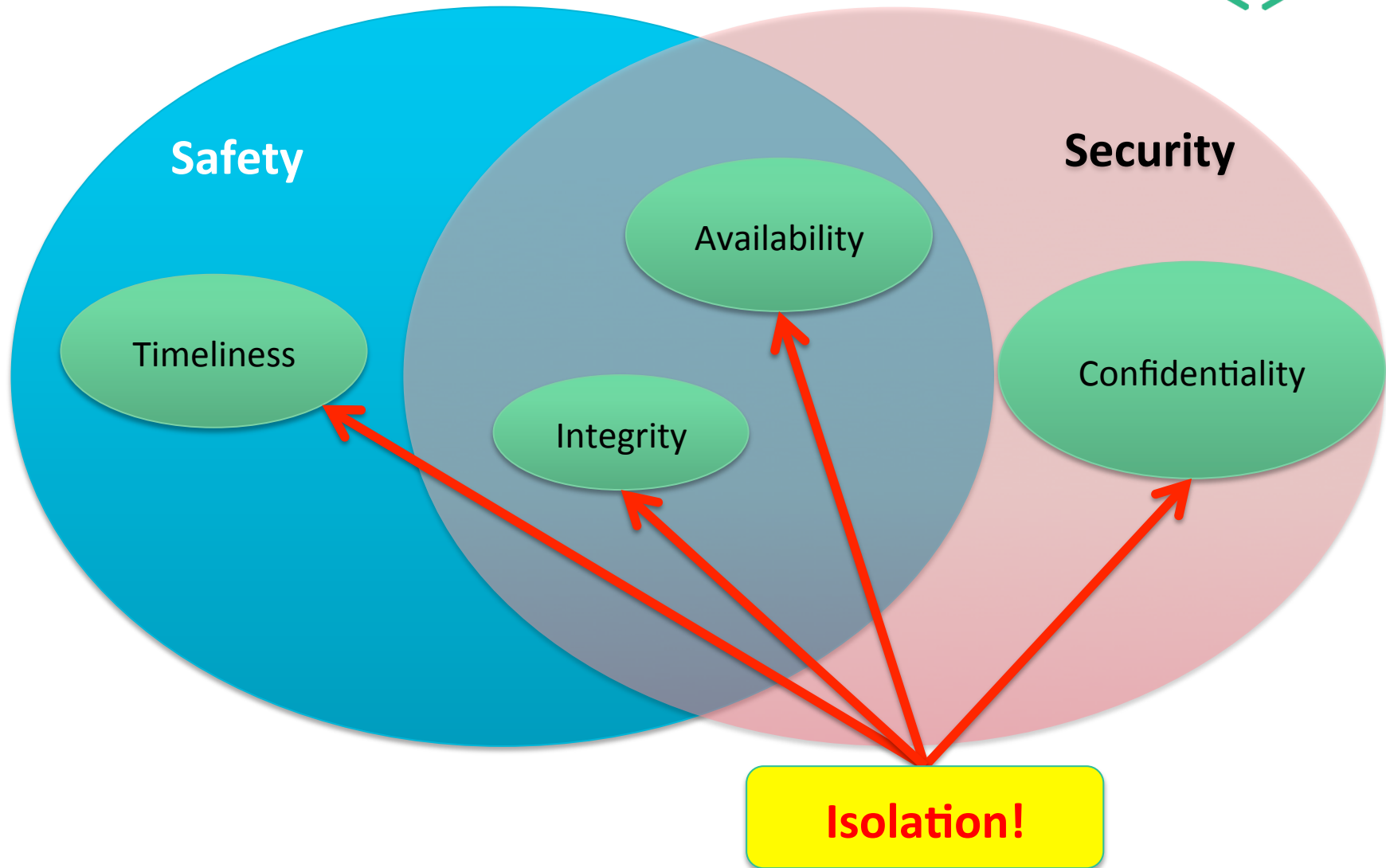
What seL4 Is Not: An Operating System



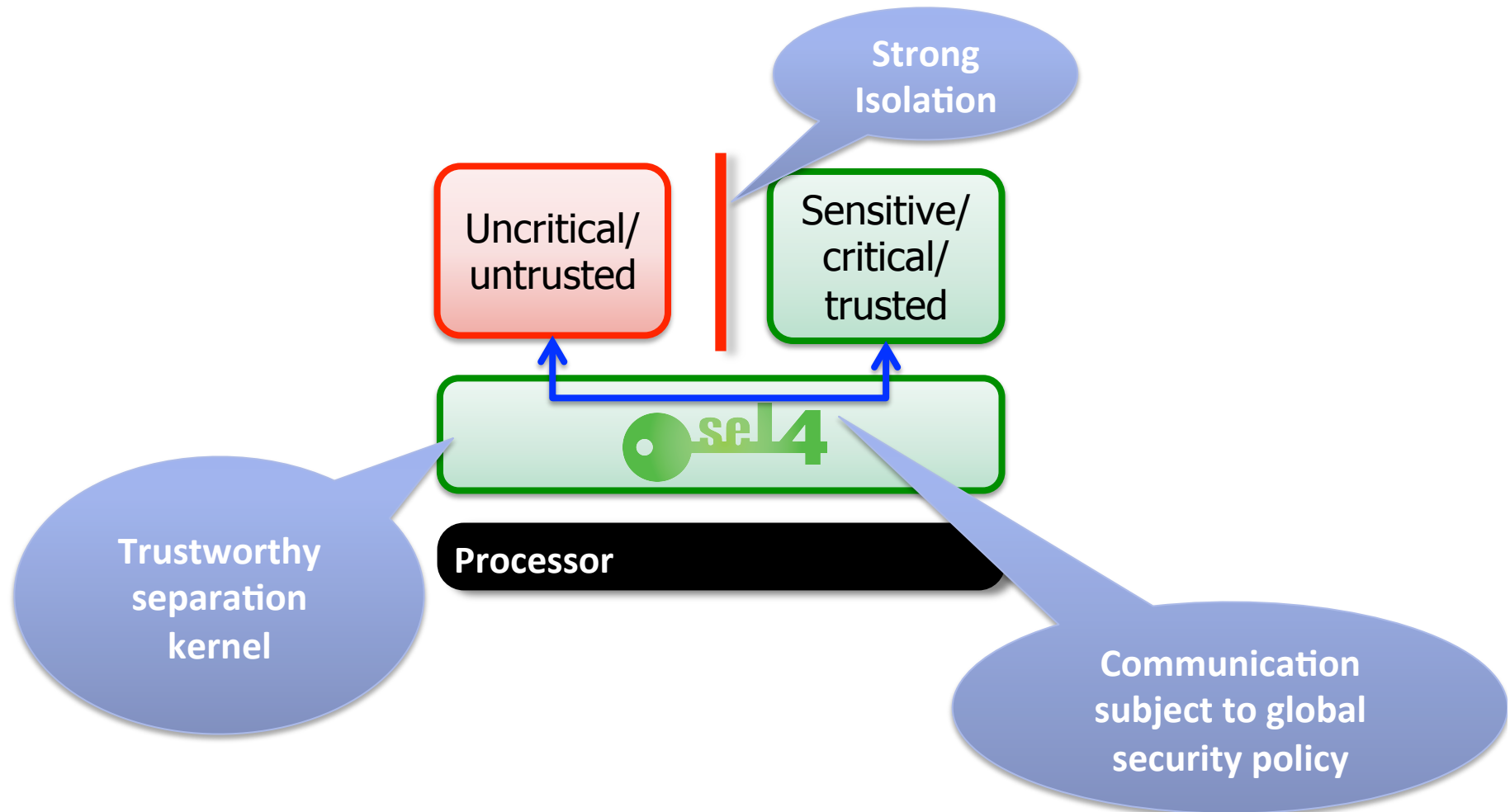
All device drivers, OS services, VMM are usermode processes



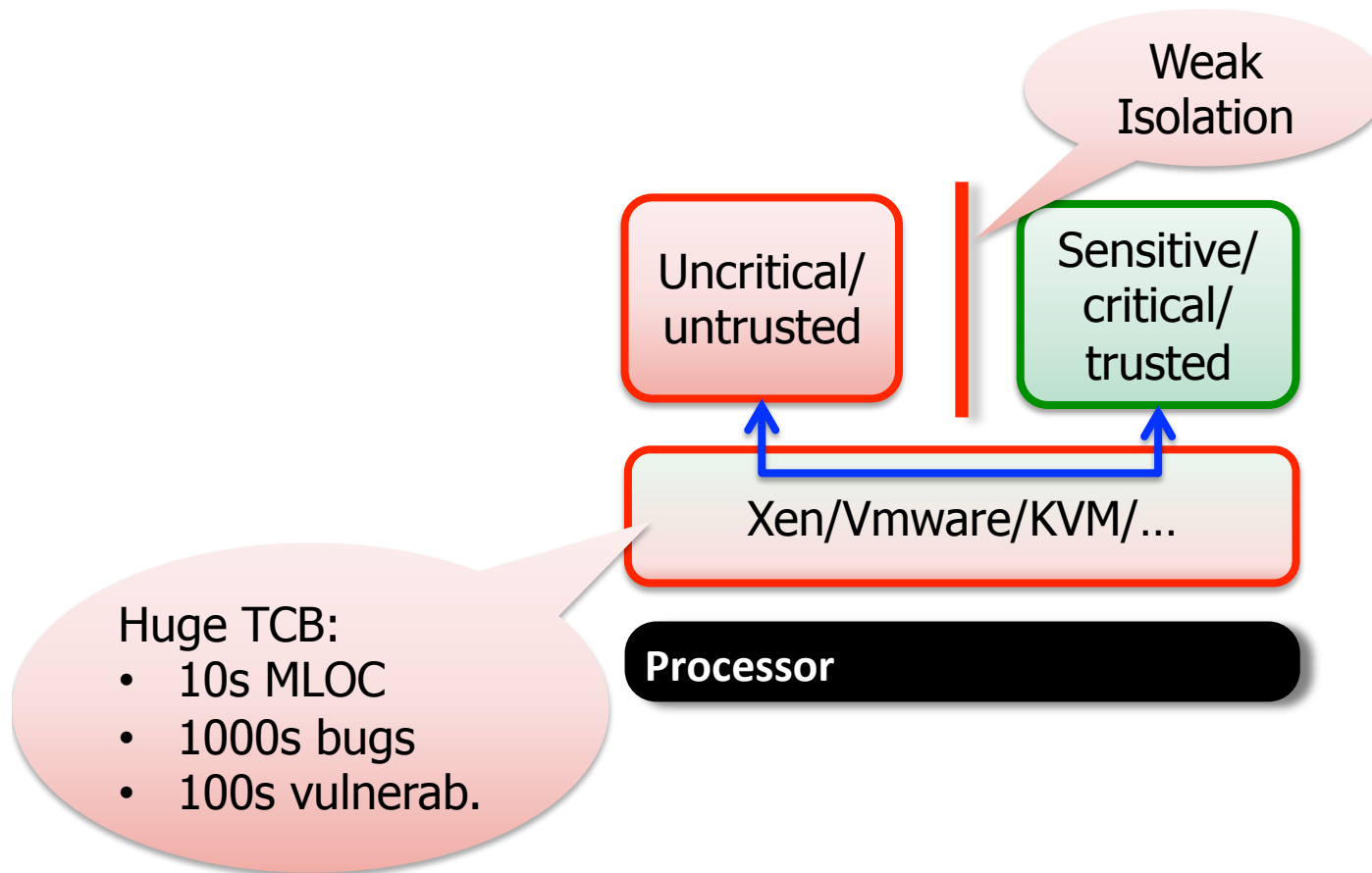
Requirements for Trustworthy Systems



Fundamental Requirement: Isolation



“High Assurance” *Bad* Practice



So, why don't we
prove
trustworthiness?

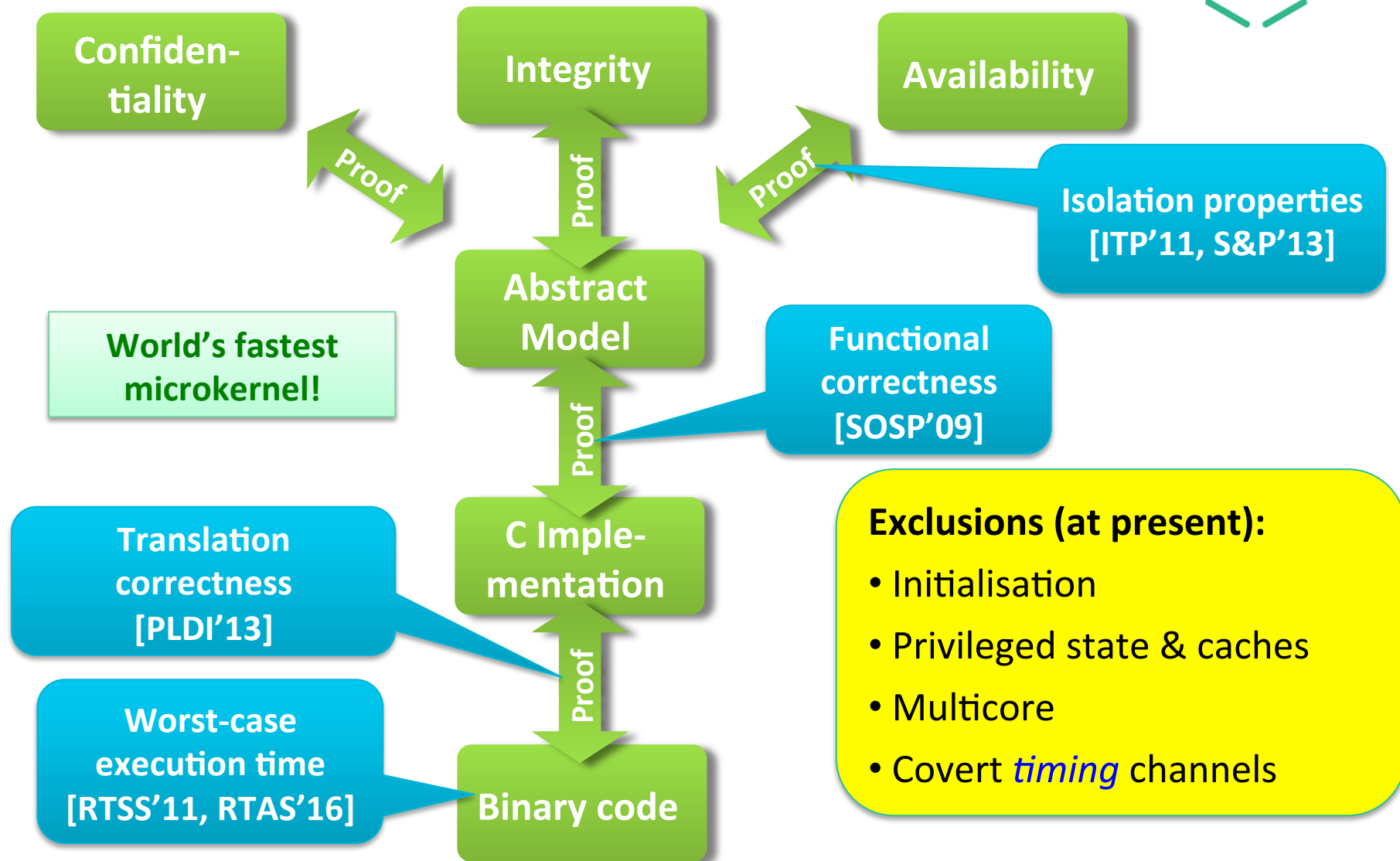
Claim: •

A system must be considered *untrustworthy*
unless *proved* otherwise!

Corollary [with apologies to Dijkstra]:

Testing, code inspection, etc. can only show
lack of trustworthiness!

seL4: Provable Isolation



Fundamental Design Decisions for seL4



1. Memory management is user-level responsibility

- Kernel never allocates memory (post-boot)
- Kernel objects controlled by user-mode servers

Isolation

2. Memory management is fully delegatable

- Supports hierarchical system design
- Enabled by *capability-based access control*

Performance

3. “Incremental consistency” design pattern

- Fast transitions between consistent states
- Restartable operations with progress guarantee

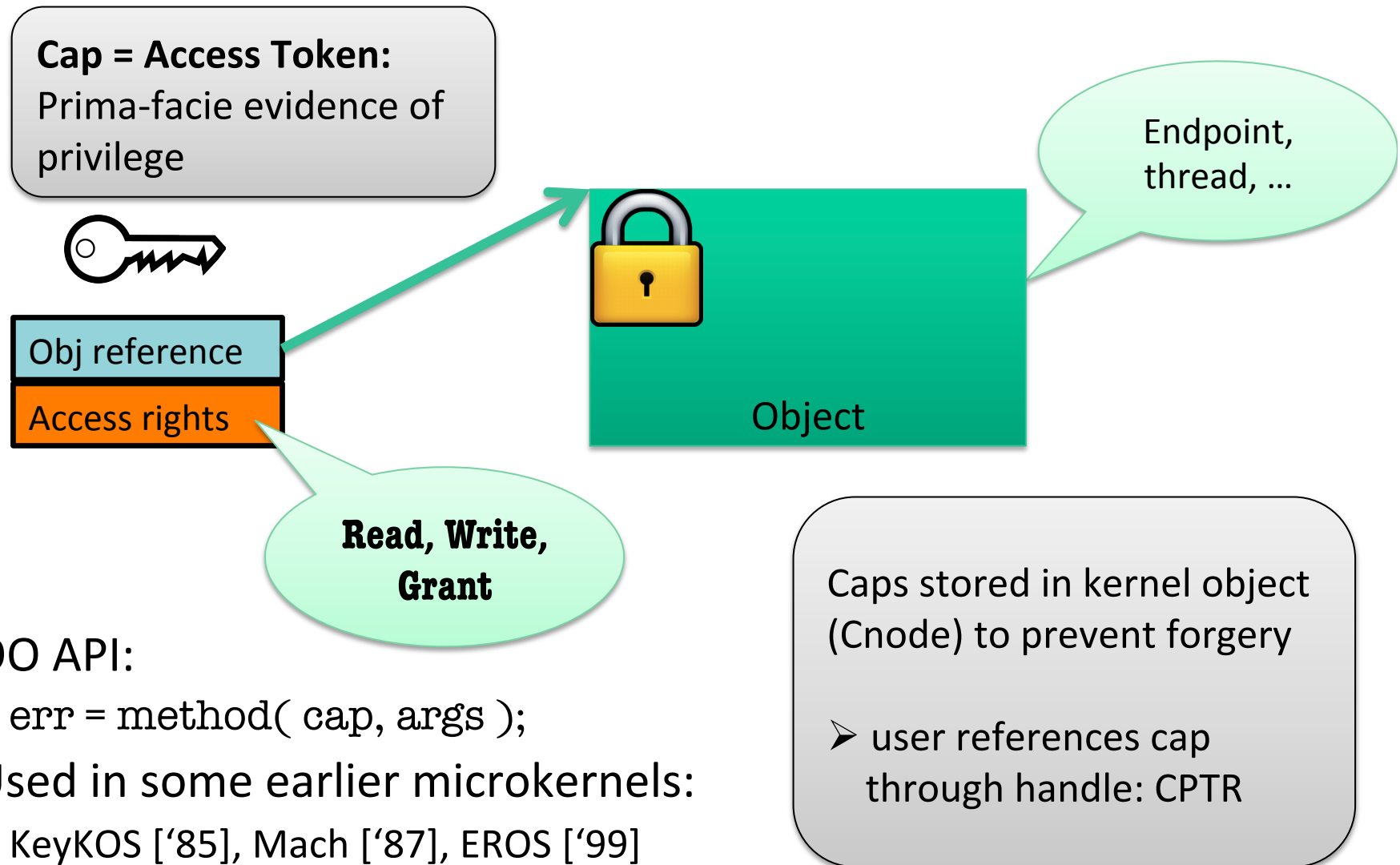
Real-time

4. No concurrency in the kernel

- Interrupts never enabled in kernel
- Interruption points to bound latencies
- Clustered multikernel design for multicores

Verification,
Performance

Key Mechanism: seL4 Capabilities



- OO API:
`err = method(cap, args);`
- Used in some earlier microkernels:
 - KeyKOS ['85], Mach ['87], EROS ['99]

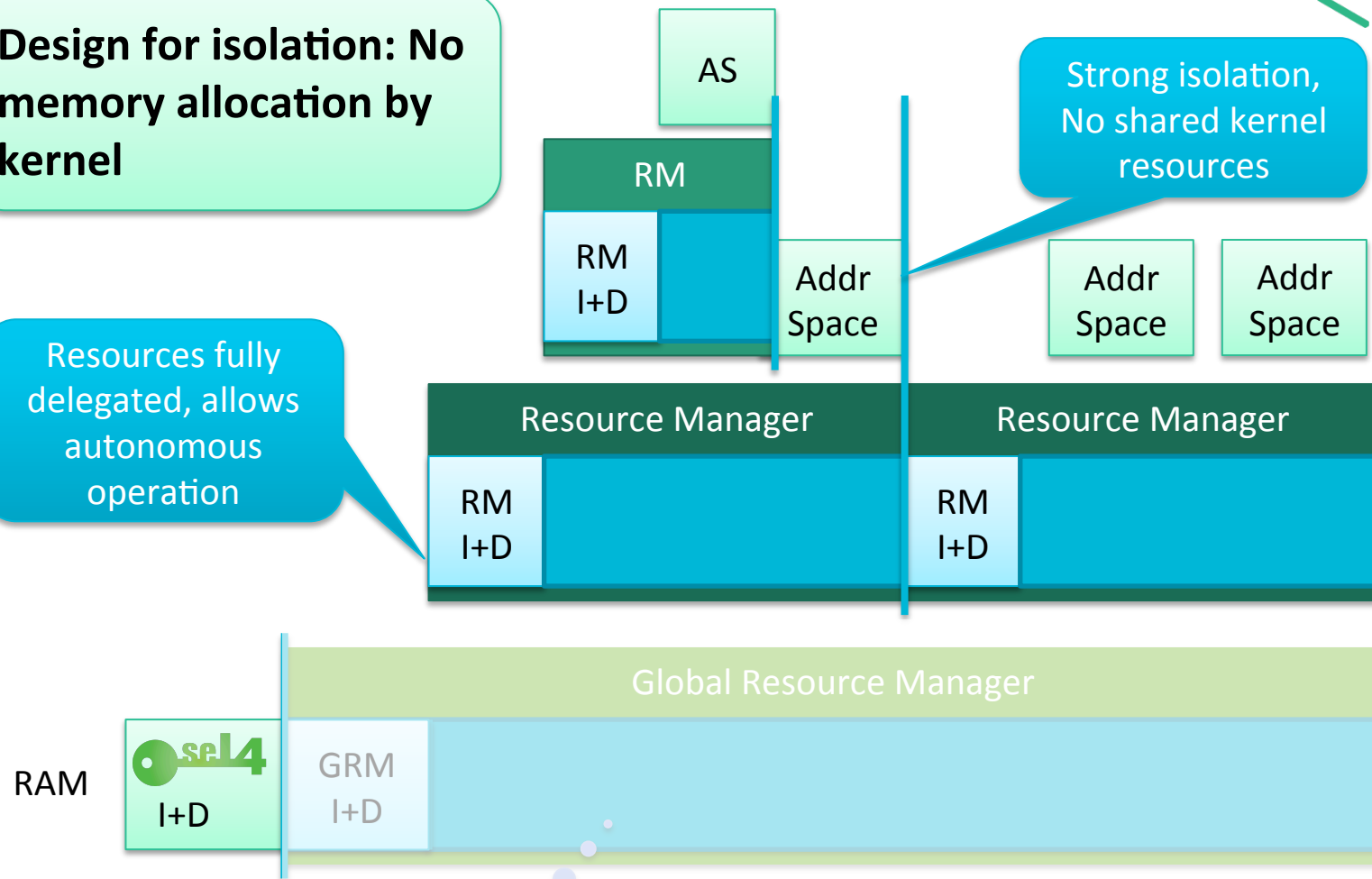
What's Different to Other Microkernels?



Design for isolation: No memory allocation by kernel

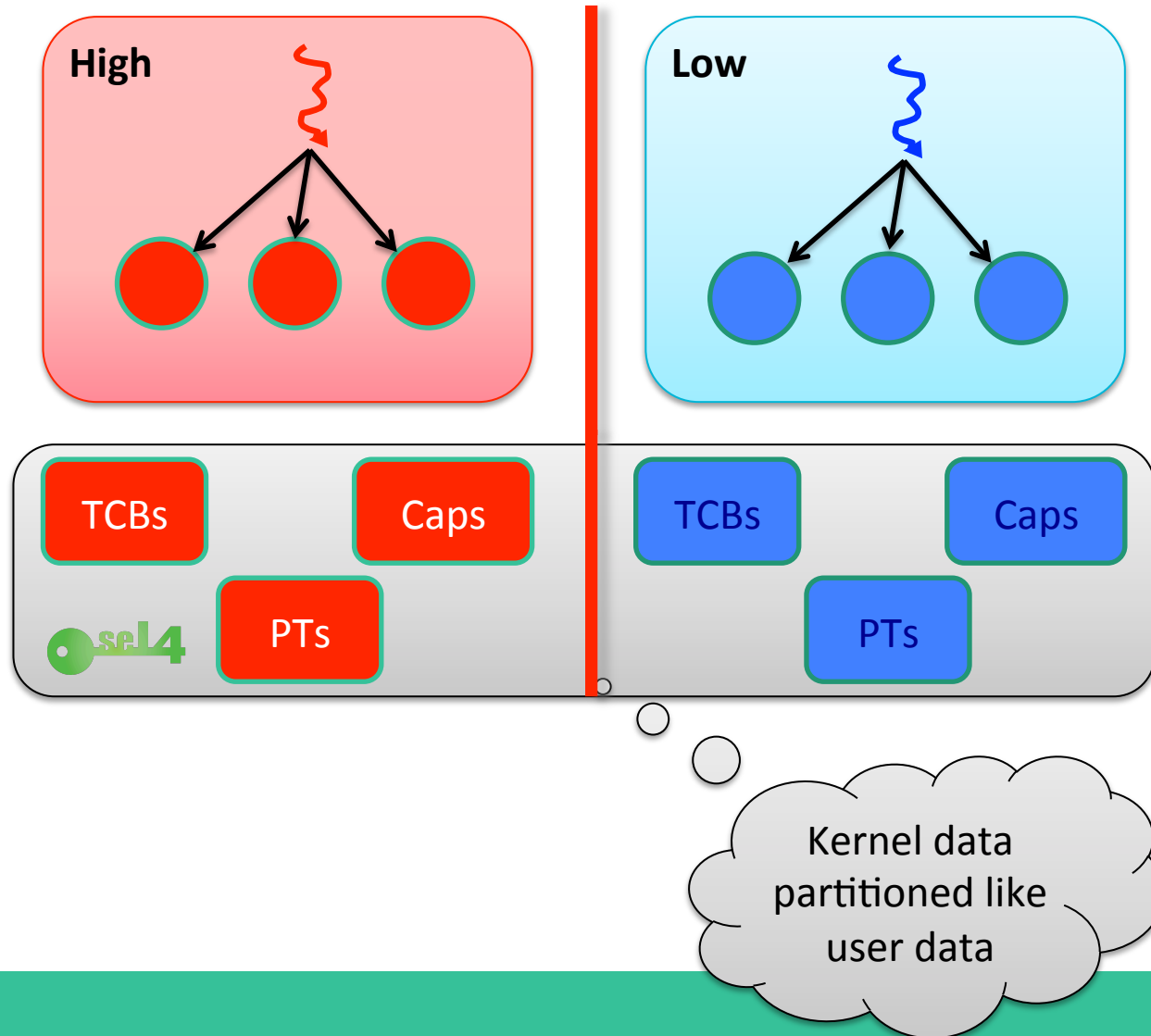
Resources fully delegated, allows autonomous operation

Strong isolation, No shared kernel resources



"Untyped" (unallocated) memory

seL4 Isolation Goes Deep



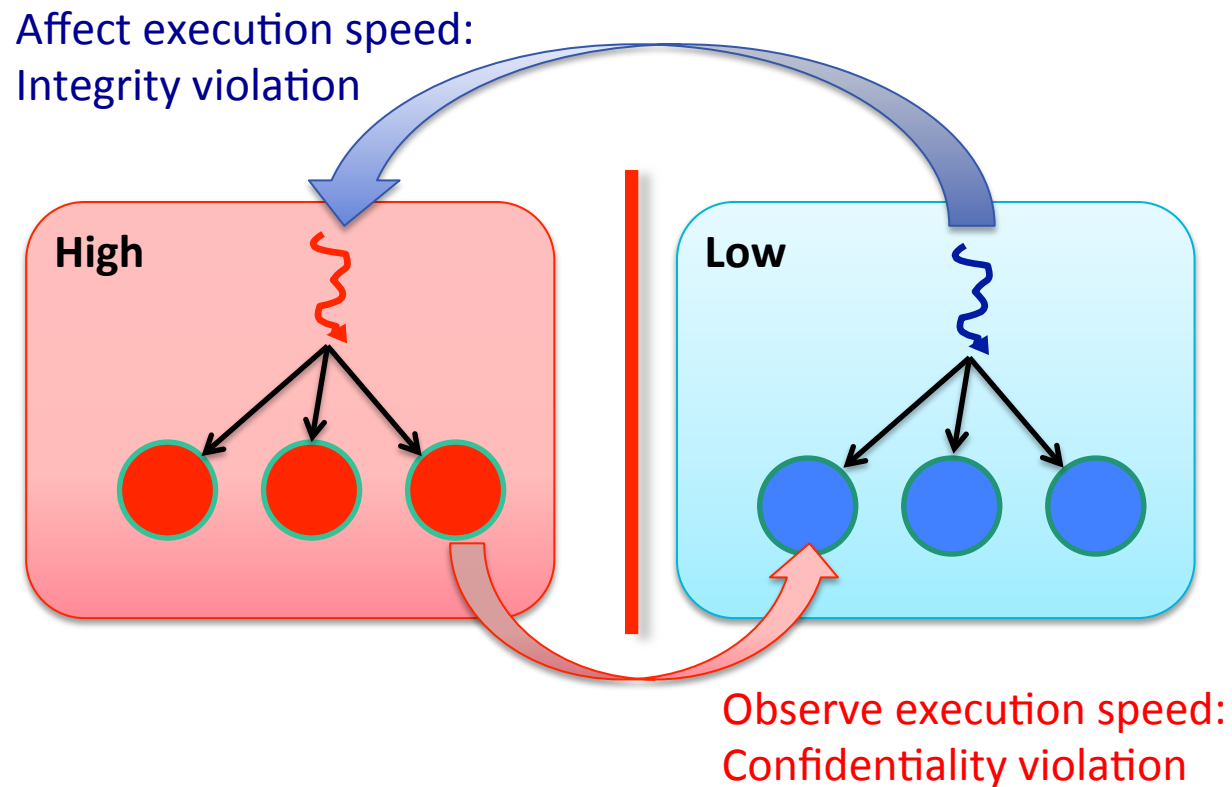
WiP: Temporal Isolation Guarantees

Safety: Timeliness

- Execution interference

Security: Confidentiality

- Leakage via timing channels



Using seL4: DARPA HACMS Program



HACMS: High-Assurance Cyber-Military Systems

- Goal: create technology for the construction of high-assurance cyber-physical systems
 - functionally correct
 - satisfying appropriate safety and security properties
- Specific project aims:
 - Protect autonomous systems from cyber attacks
 - Demonstrate deployment in real-world systems
 - **Open-source all non-vehicle-specific code**

HACMS: 3 Teams



Air Team – “SMACCM”



Land Team



Red
Team

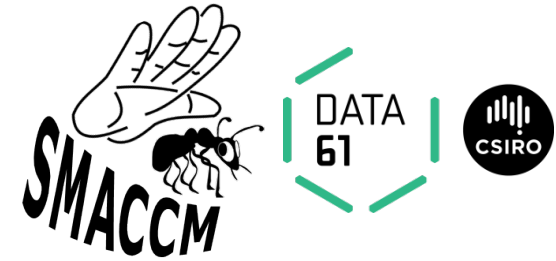
Image courtesy of chanpipat at FreeDigitalPhotos.net

HACMS: 3 Phases



- Phase 1: August '12 to January '14
 - Simplified high-assurance system
- Phase 2: February '14 to July '15
 - Adding real-world complexity
 - Full-system demo
- Phase 3: August'15 to January'17
 - Transition to real-world military vehicle
 - Boeing Unmanned Little Bird helicopter
 - Autonomous US Army trucks
 - Possibly research drone as “minimal viable product”

Secure, Mathematically-Assured Composition of Control Models



SMACCM Objectives:

- Provable vehicle safety
- “Red Team” must not be able to divert vehicle
- No sacrificing performance

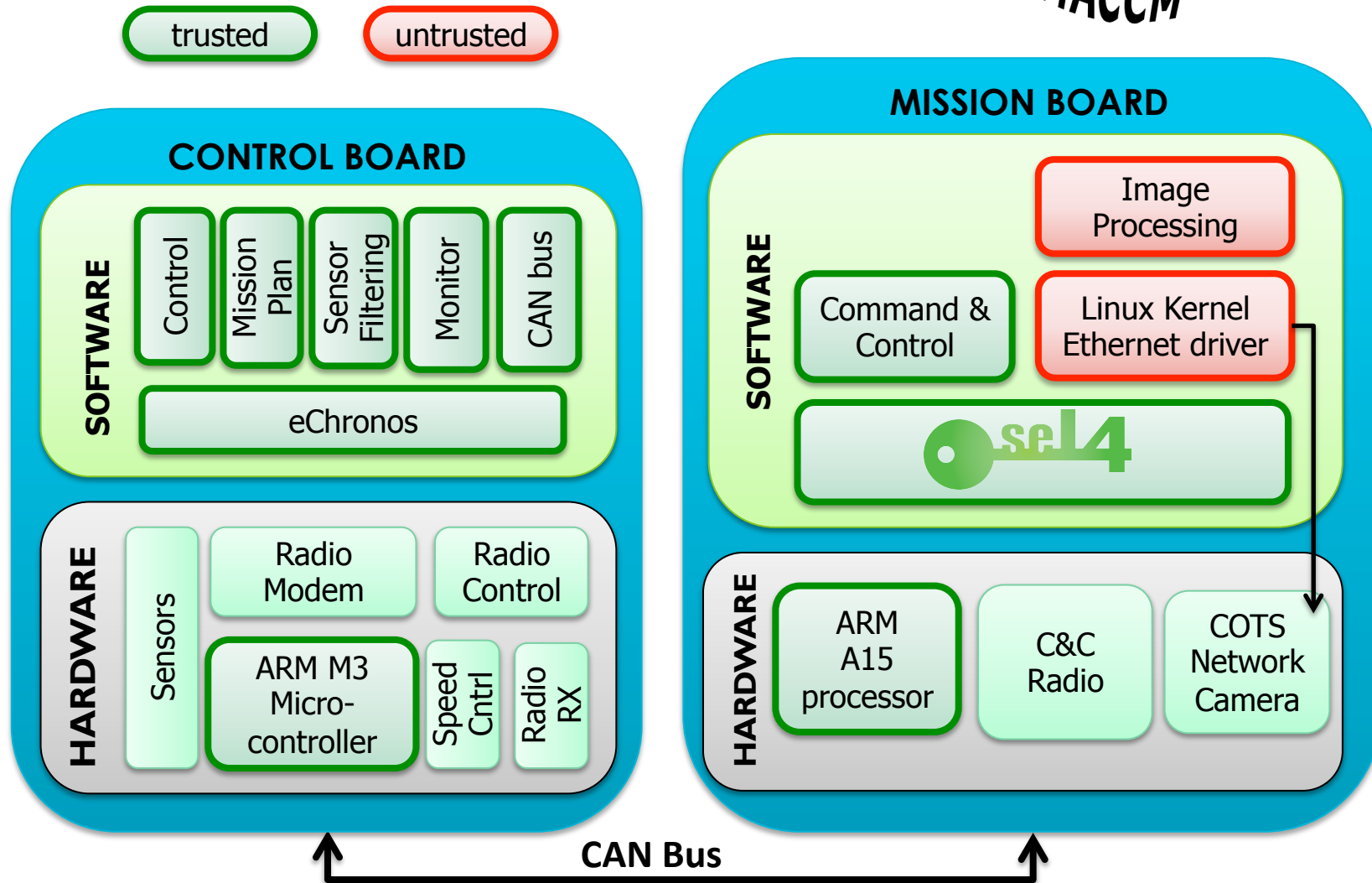
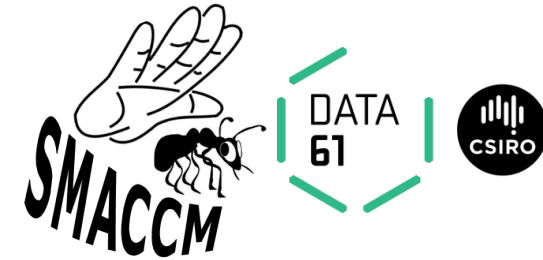
Unmanned Little Bird
Deployment Vehicle



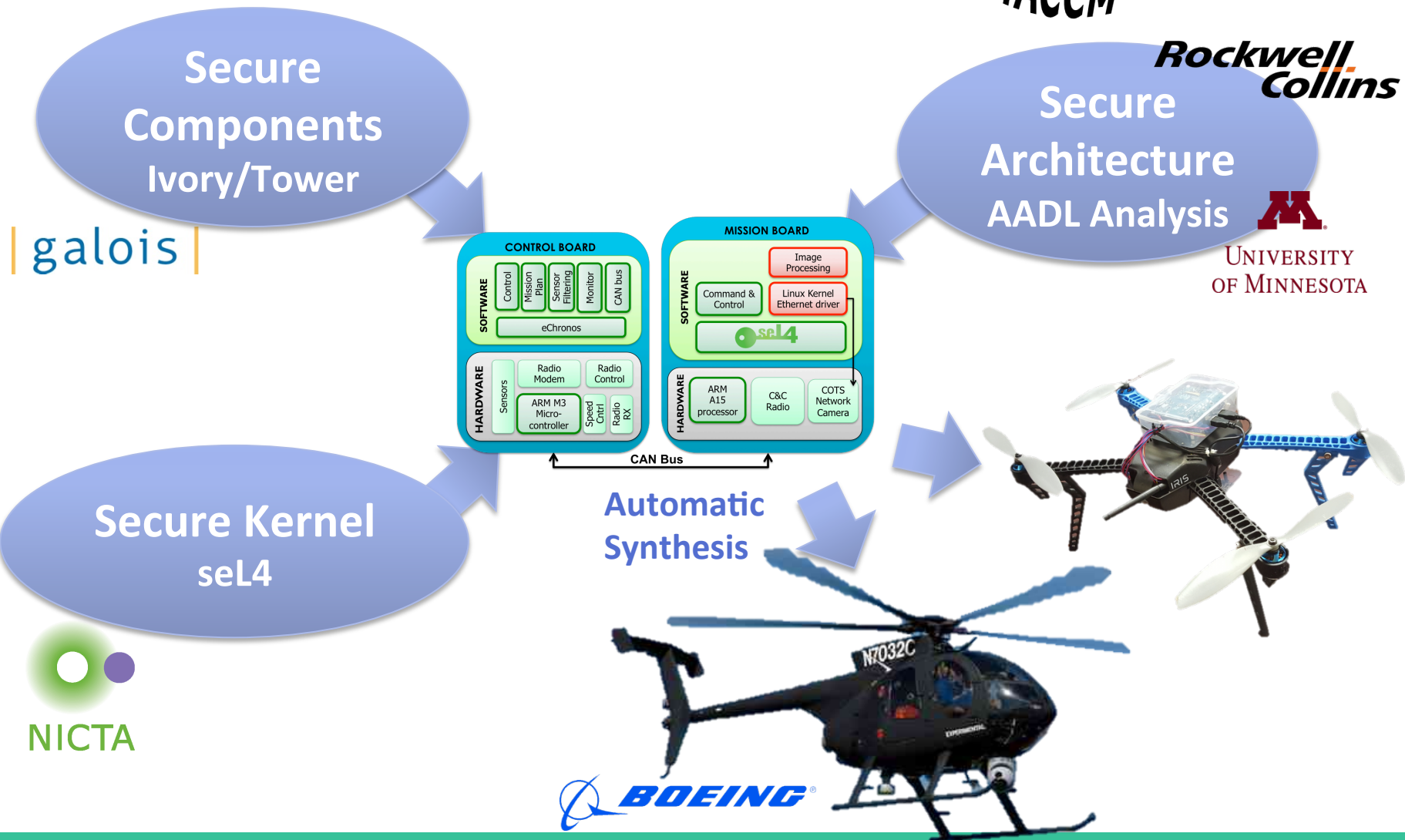
SMACCMcopter
Research Vehicle



SMACCMcopter Architecture



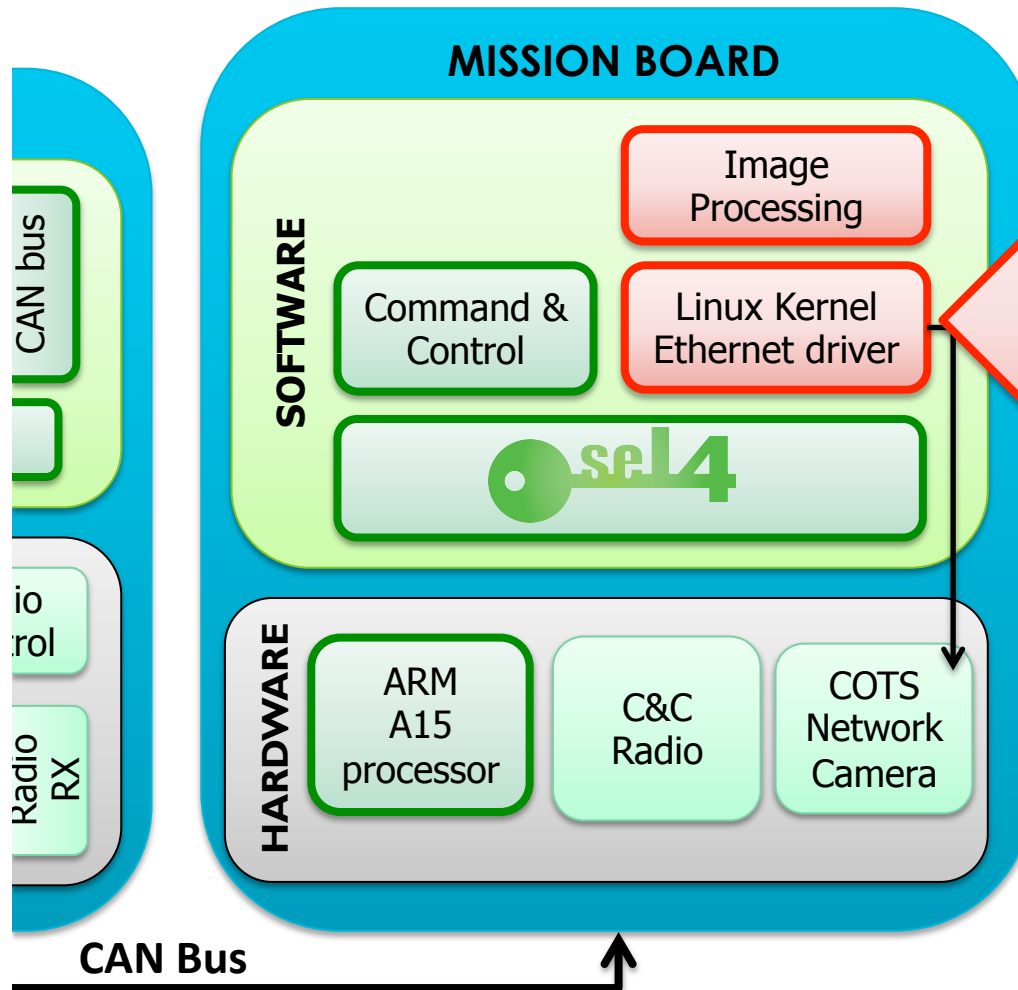
SMACCM Building Blocks



Phase 2 Security Evaluation



Image courtesy of chanpipat at FreeDigitalPhotos.net



Red Team unable to compromise rest of system (white-box attack)

"World's most highly assured drone" [DARPA]

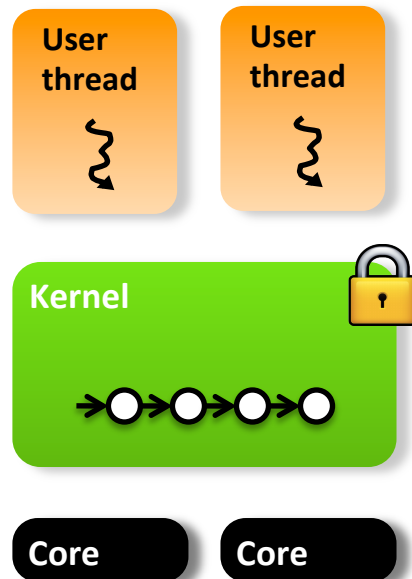
DATA
61



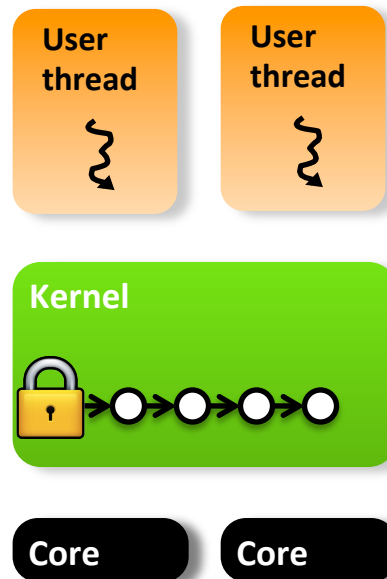
Dealing with Multicore

Approaches for Multicore Kernels

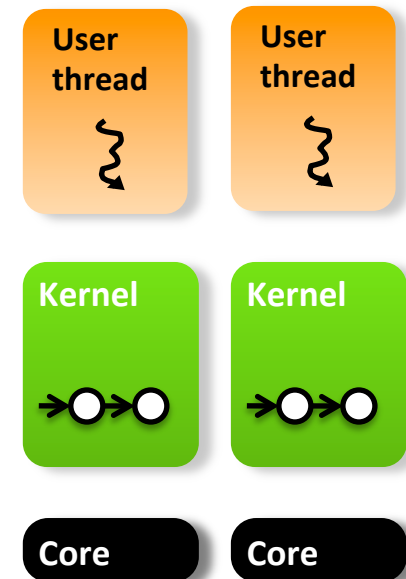
SMP big lock



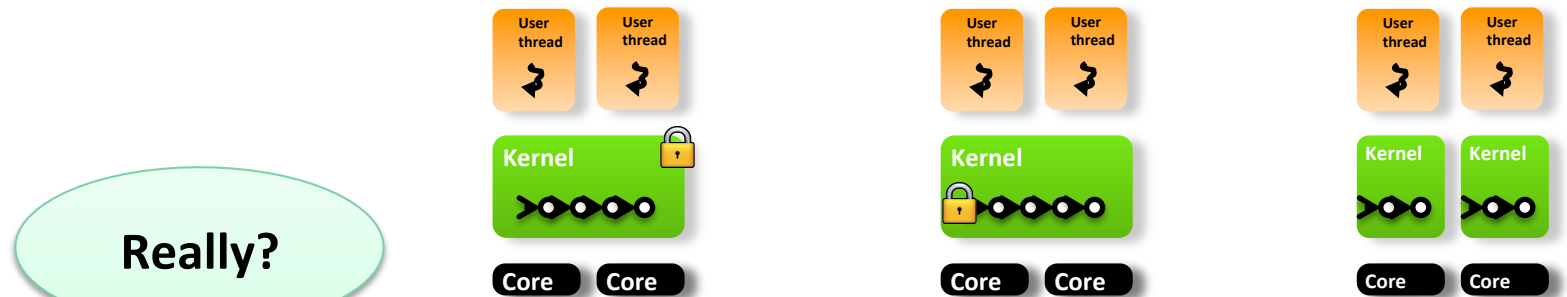
SMP fine-grained locks



Multikernel no locks



Multicore Kernel Trade-Offs



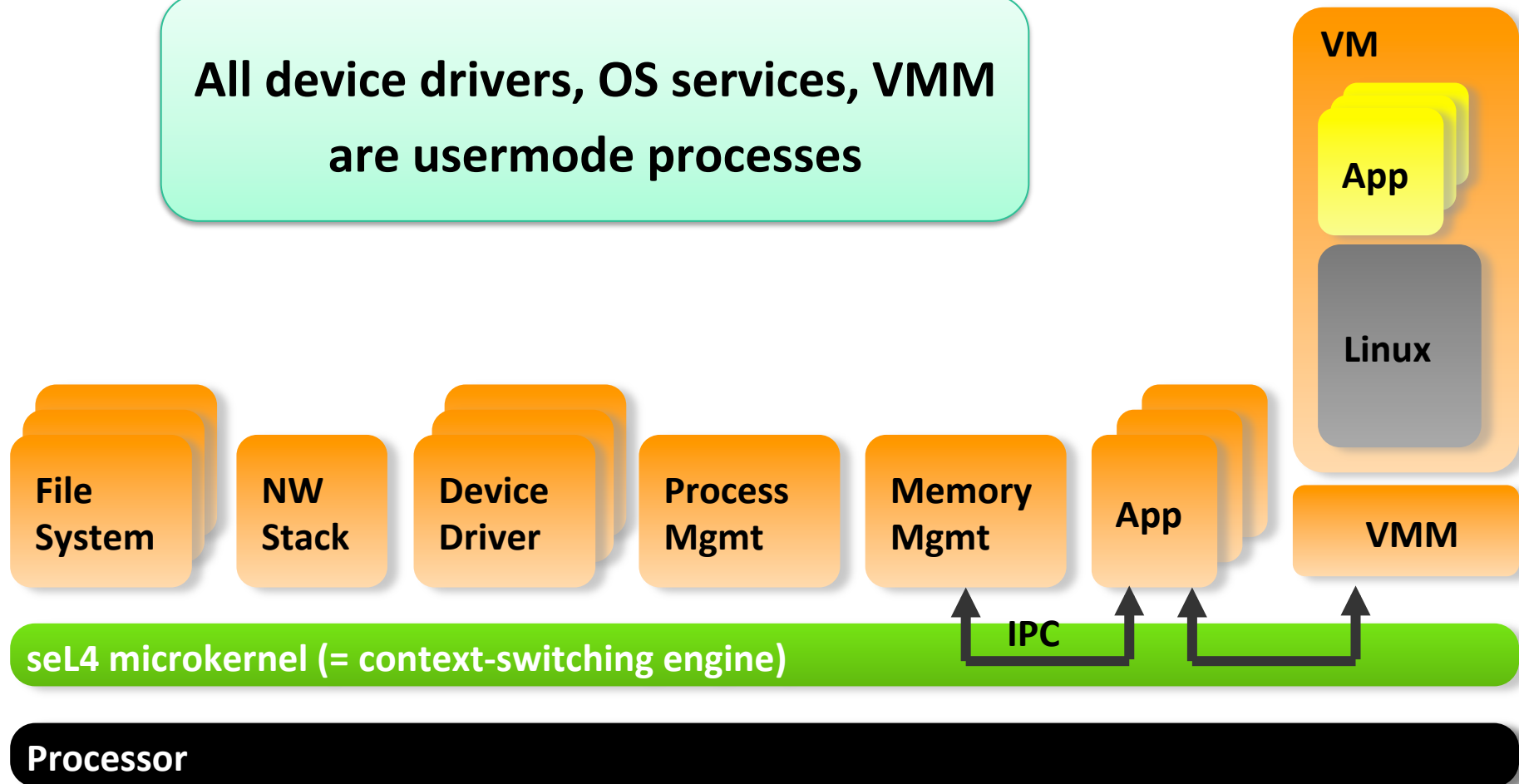
Really?

Property	Big Lock	Fine-grained Locking	Multikernel
Data structures	shared	shared	distributed
Scalability	poor	good	excellent
Concurrency in kernel	zero	high	zero
Kernel complexity	low	high	low
Resource management	centralised	centralised	distributed

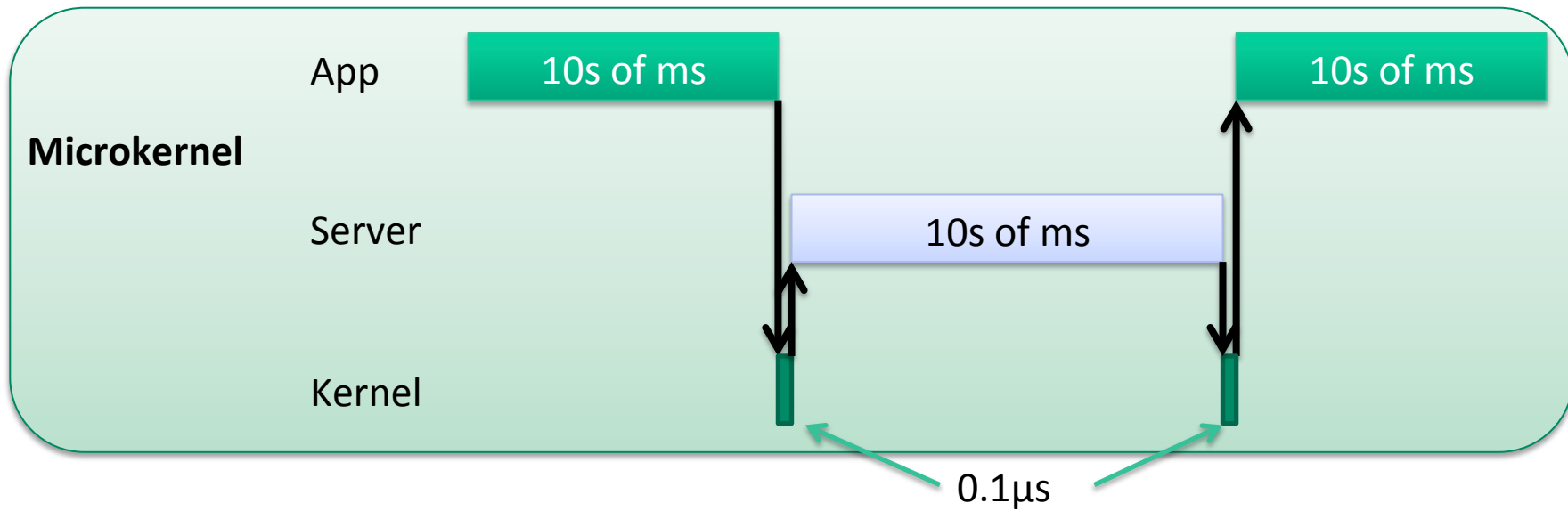
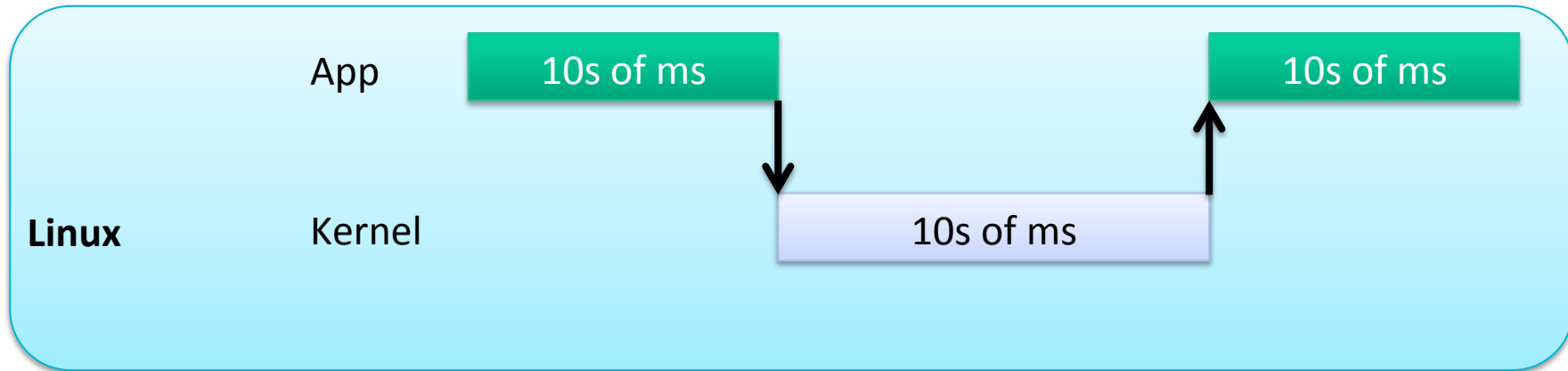
Remember: Microkernel \neq Operating System!



All device drivers, OS services, VMM
are usermode processes



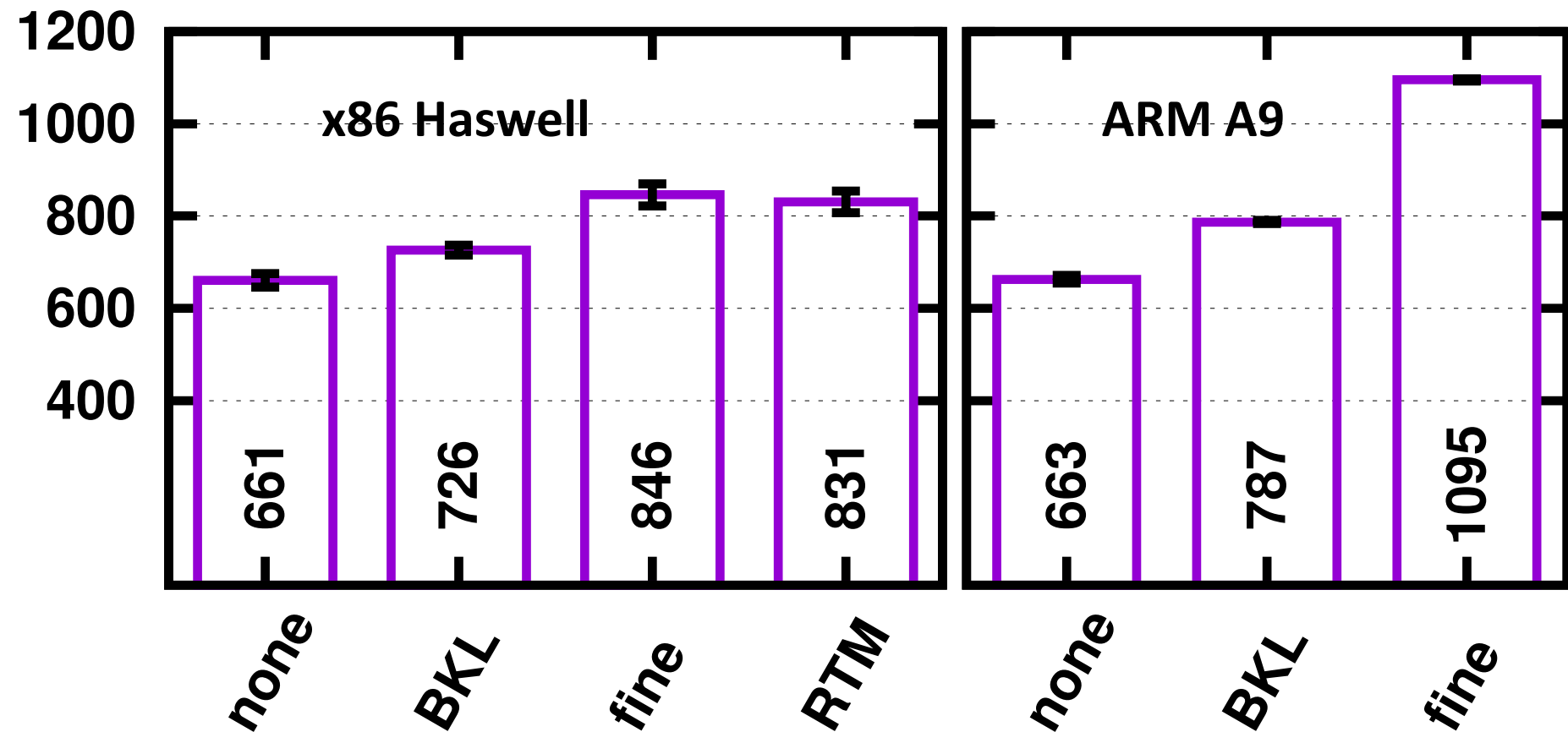
Microkernel vs Linux Execution



Cost of Locking: Round-Trip Intra-Core IPC



Cycles

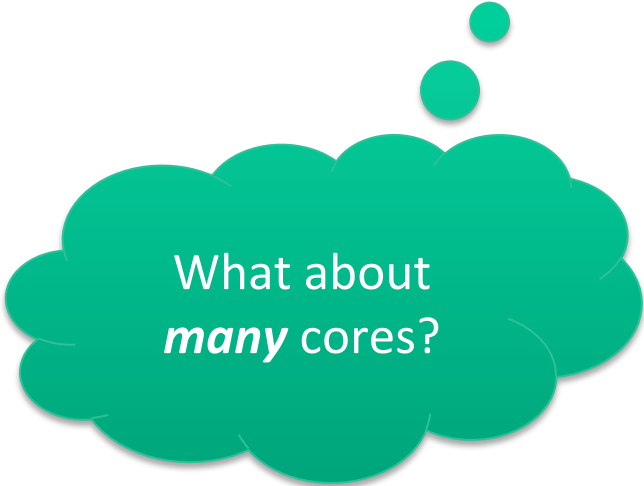


Microkernel Multicore Design



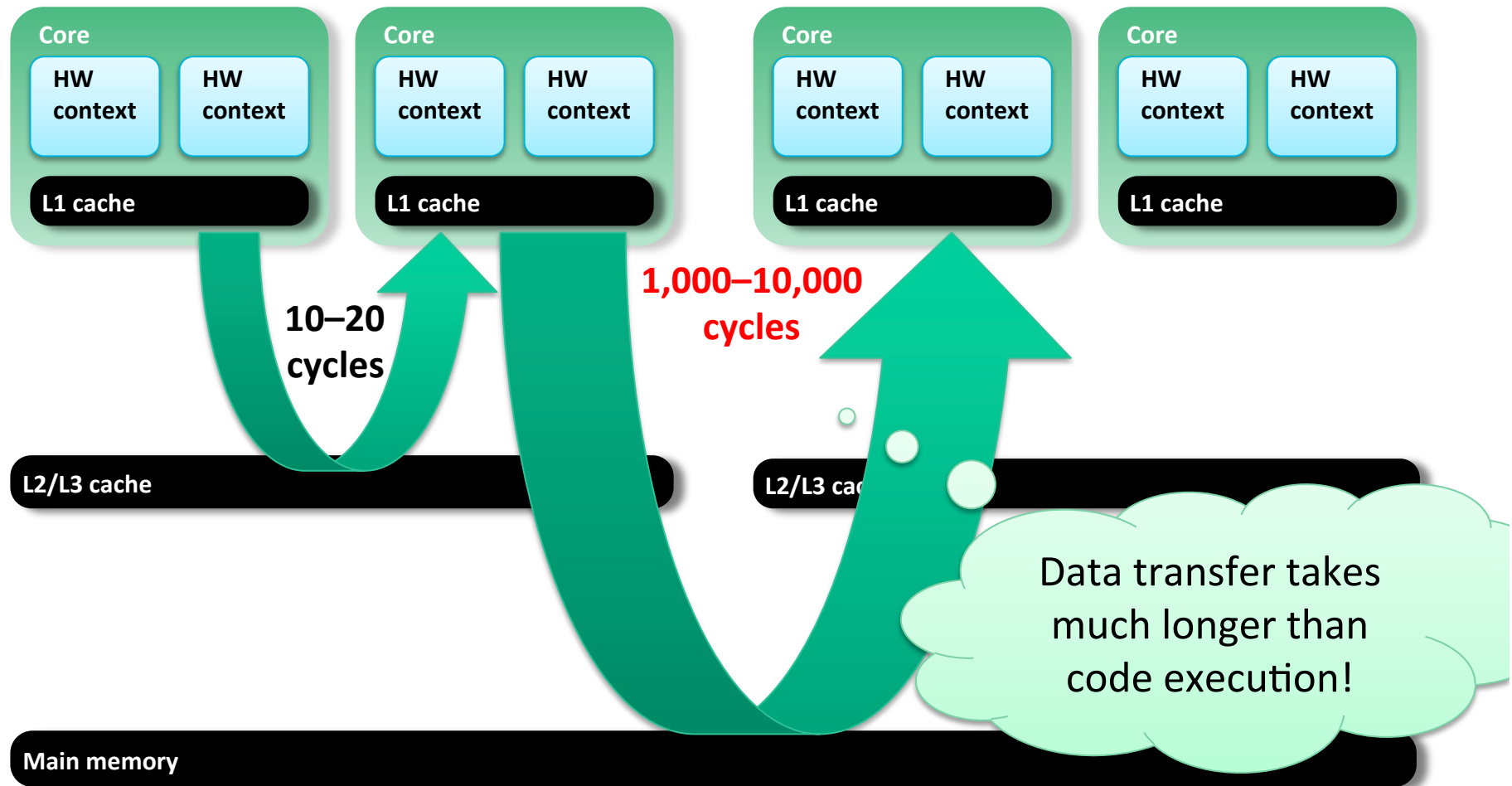
Assertion 1: Minimise locks, not locked code

- Amount of locked code is small anyway, 100–200 instructions
- Corresponds to fine- to medium-grained locks in Linux
- Cost of locks is within an OoM of kernel execution time
- Kernel times are short \Rightarrow contention is low



What about
many cores?

Cache Line Migration Latencies



Microkernel Multicore Design



Assertion 1: Minimise locks, not locked code

- Amount of locked code is small anyway, 100–200 instructions
- Corresponds to medium-grained locks in Linux
- Cost of locks is within an OoM of kernel execution time
- Kernel times are short \Rightarrow contention is low

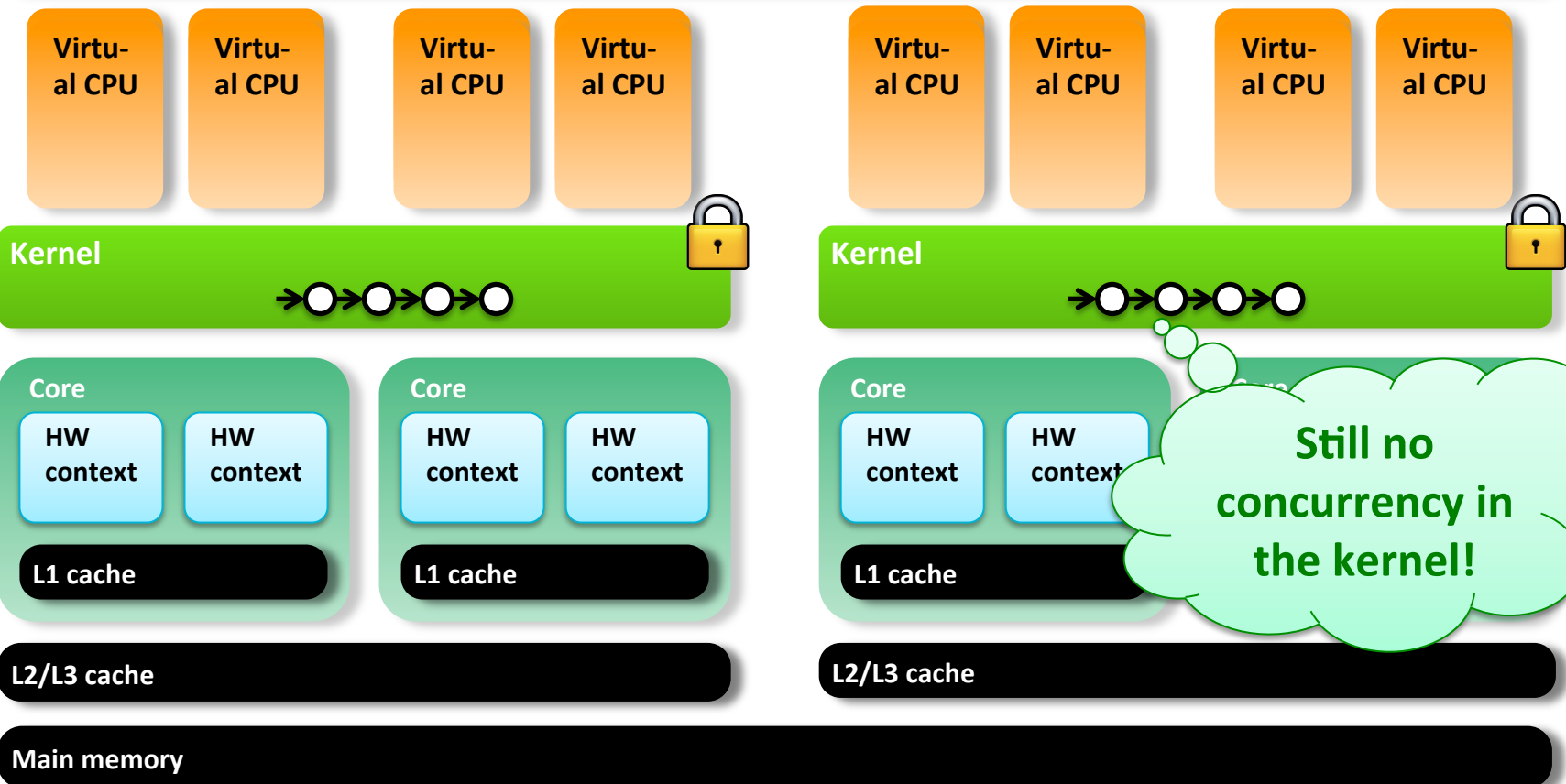
Assertion 2: Don't share mikrokernel data without shared cache

- Migrating only a few cache lines takes longer than rest of syscall

seL4 Multicore Design: Clustered Multikernel



SMP Linux



Microkernel Multicore Design



Assertion 1: Minimise locks, not locked code

- Amount of locked code is small anyway, 100–200 instructions
- Corresponds to medium-grained locks in Linux
- Cost of locks is within an OoM of kernel execution time
- Kernel times are short \Rightarrow contention is low

Assertion 2: Don't share mikrokernel data without shared cache

- Migrating only a few cache lines takes longer than rest of syscall

Assertion 3: Big lock will perform for closely-coupled cores

- Shared caches presently have moderate core counts
- Big lock in a *well-designed* microkernel will scale there



Thank you

Gernot Heiser | gernot.heiser@data61.csiro.au | @GernotHeiser
<http://microkernel.dude.wordpress.com>

February 2016

<https://trustworthy.systems/>

