Making Systems Trustworthy: The seL4 Microkernel

Security is no excuse for poor performance!

Gernot Heiser | gernot.heiser@data61.csiro.au | @GernotHeiser
October 2016

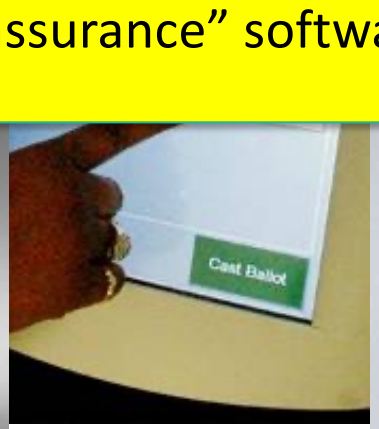
https://sel4.systems


Windows
An exception 06 has occured at 0028:C11B3ADC in VxD DiskTSD(03) +
00001660. This was called from 0028:C11B40C8 in VxD voltrack(04) +
00000000. It may be possible to continue normally.
* Press any key to attempt to continue.
* Press CTRL+ALT+RESET to restart your computer. You will
lose any unsaved information in all applications.
Press any key to continue

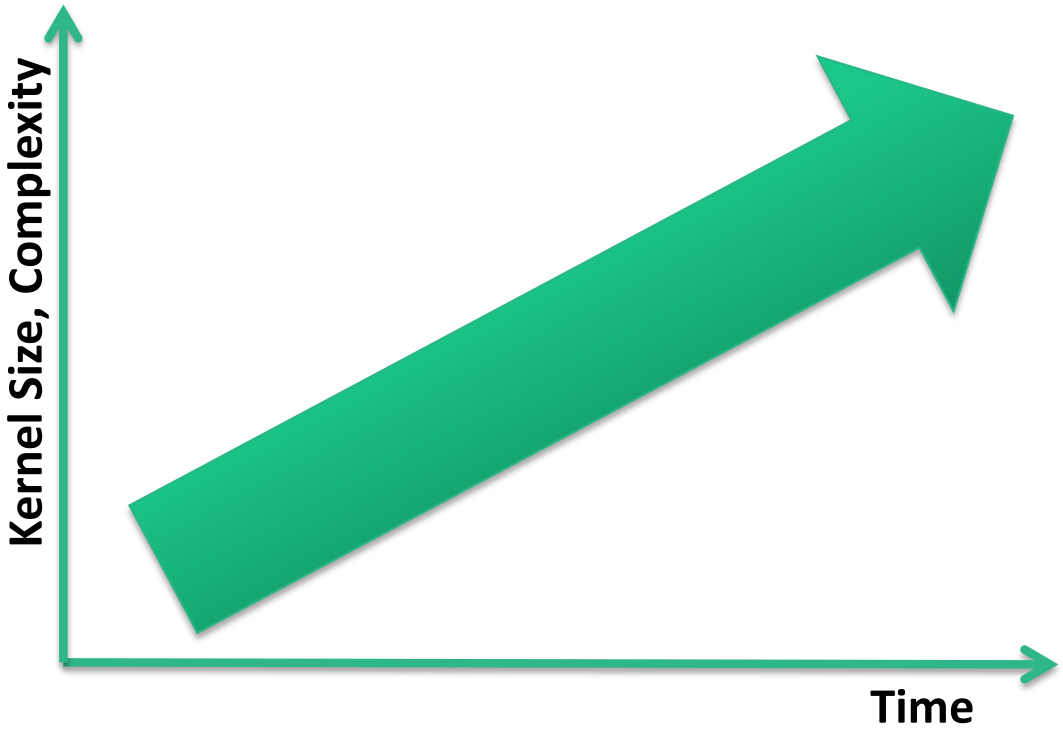
Present Systems are *NOT* Trustworthy!

Yet they are expensive:

- $1,000 per line of code for "high-assurance" software!



Trends in Commodity Operating Systems



Complexity Drivers

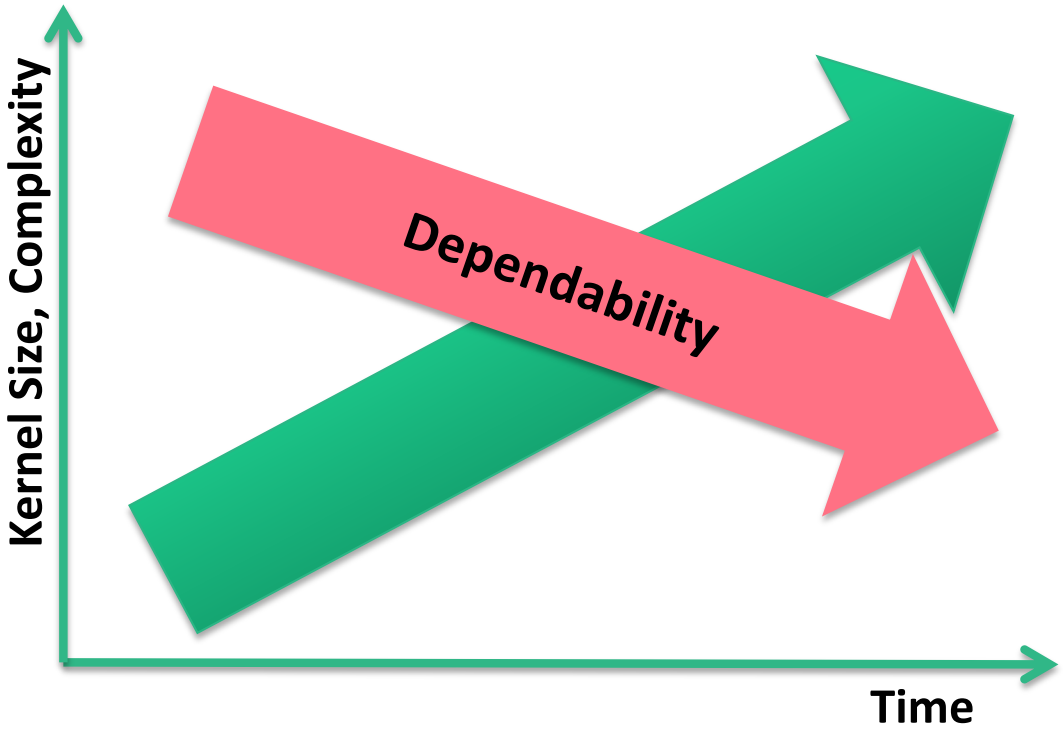
- New hardware
  - New device drivers / driver classes
  - New file systems
  - Multicore scalability
- New usage domains
  - Better power management
  - New network protocols
  - Better real-time behaviour
- New security challenges
  - New crypto libs, protocols
  - Improved access control
- Etc …

Complexity: Enemy of Dependability

- Typical defect density of industry-standard code: 2–5 bugs per kSLOC
  - Linux might be somewhat better: ≈ 1 bug/kSLOC
- 10–25% of kernel bugs are security vulnerabilities
  - Conservatively, this means 0.1 vulnerability / kSLOC
- Linux kernel is 10s of MSLOC ⇒ **thousands of vulnerabilities!**

SLOC:
source lines of code

Trends in Commodity Operating Systems



Complexity Drivers

- New hardware
  - New device drivers / driver classes
  - New file systems
  - Multicore scalability
- New usage domains
  - Better power management
  - New network protocols
  - Better real-time behaviour
- New security challenges
  - New crypto libs, protocols
  - Improved access control
- Etc …

Complexity: Enemy of Dependability

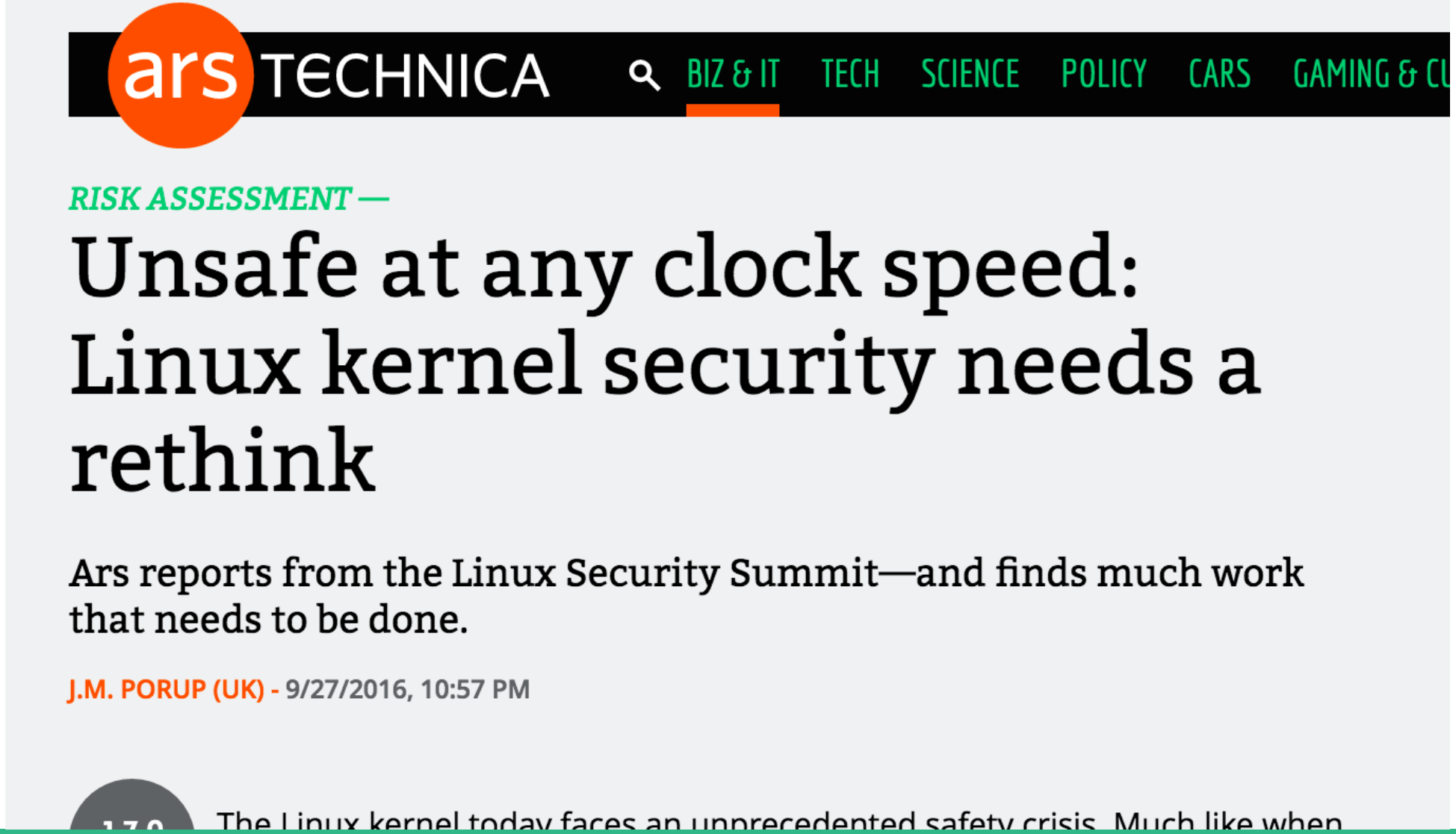
- Typical defect density of industry-standard code: 2–5 bugs per kSLOC
  - Linux might be somewhat better: ≈ 1 bug/kSLOC
- 10–25% of kernel bugs are security vulnerabilities
  - Conservatively, this means 0.1 vulnerability / kSLOC
- Linux kernel is 10s of MSLOC ⇒ thousands of vulnerabilities!

Core problem: new features increase kernel complexity ⇒ reduce dependability

- Impossible to assure security – too many bugs
- Impossible to assure safety – too complex to analyse timeliness

The monolithic OS model
Is fundamentally broken!

I'm not alone saying this…



ars TECHNICA BIZ & IT TECH SCIENCE POLICY CARS GAMING & CU

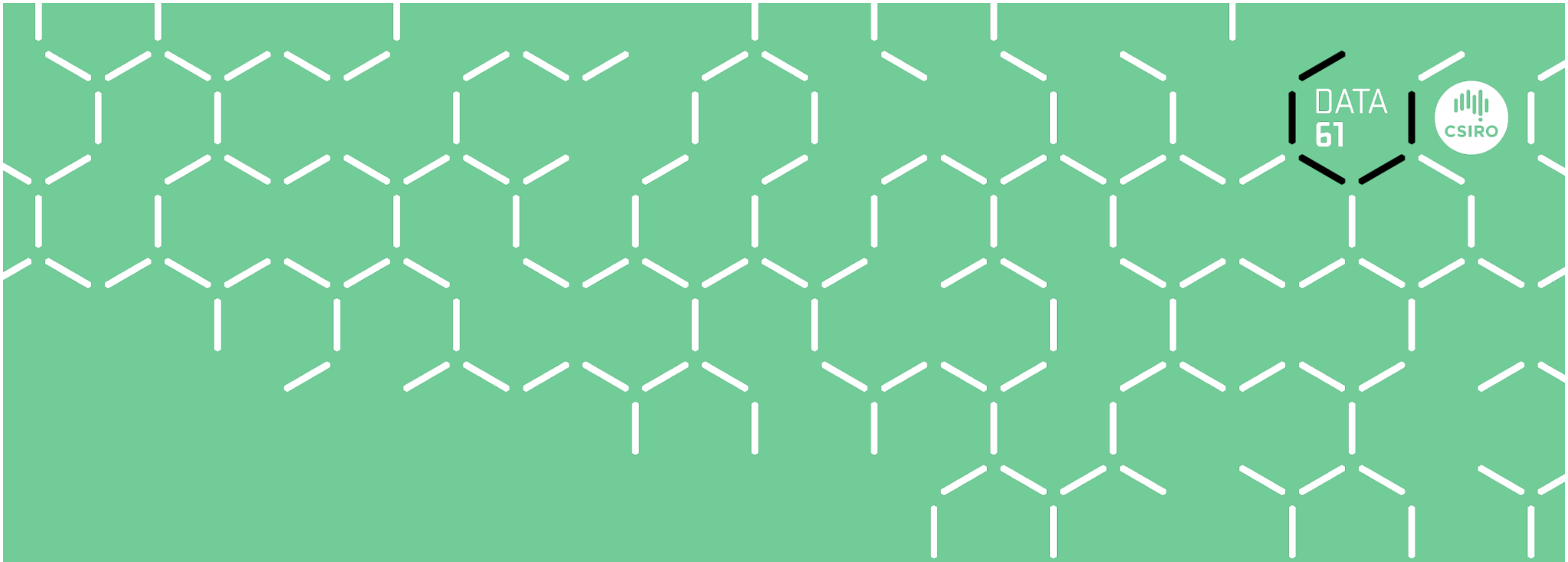
RISK ASSESSMENT —

Unsafe at any clock speed: Linux kernel security needs a rethink

Ars reports from the Linux Security Summit—and finds much work that needs to be done.

J.M. PORUP (UK) - 9/27/2016, 10:57 PM

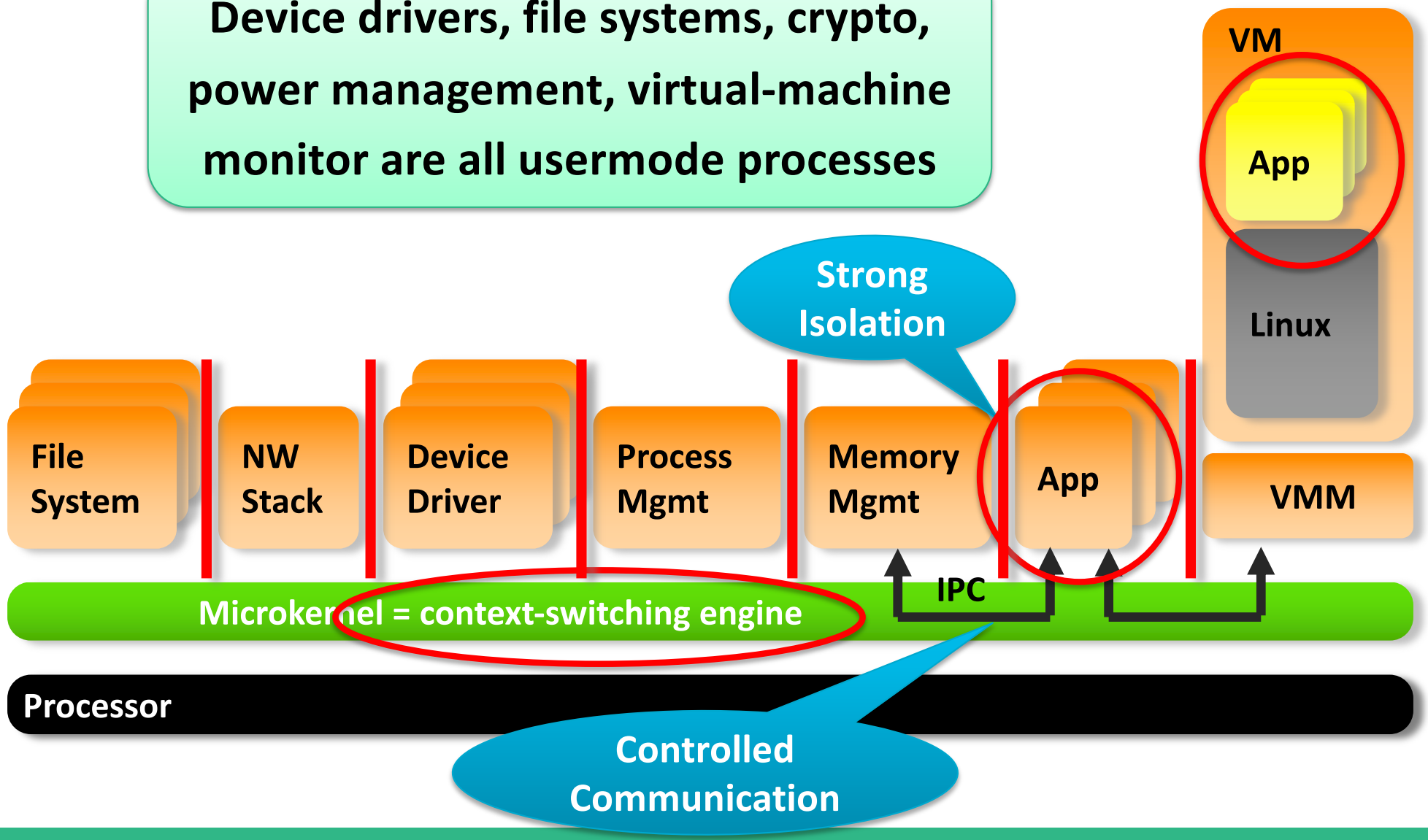
170 The Linux kernel today faces an unprecedented safety crisis. Much like when



Microkernels

Alternative: Microkernels

Device drivers, file systems, crypto, power management, virtual-machine monitor are all usermode processes



Monolithic Kernels vs Microkernels

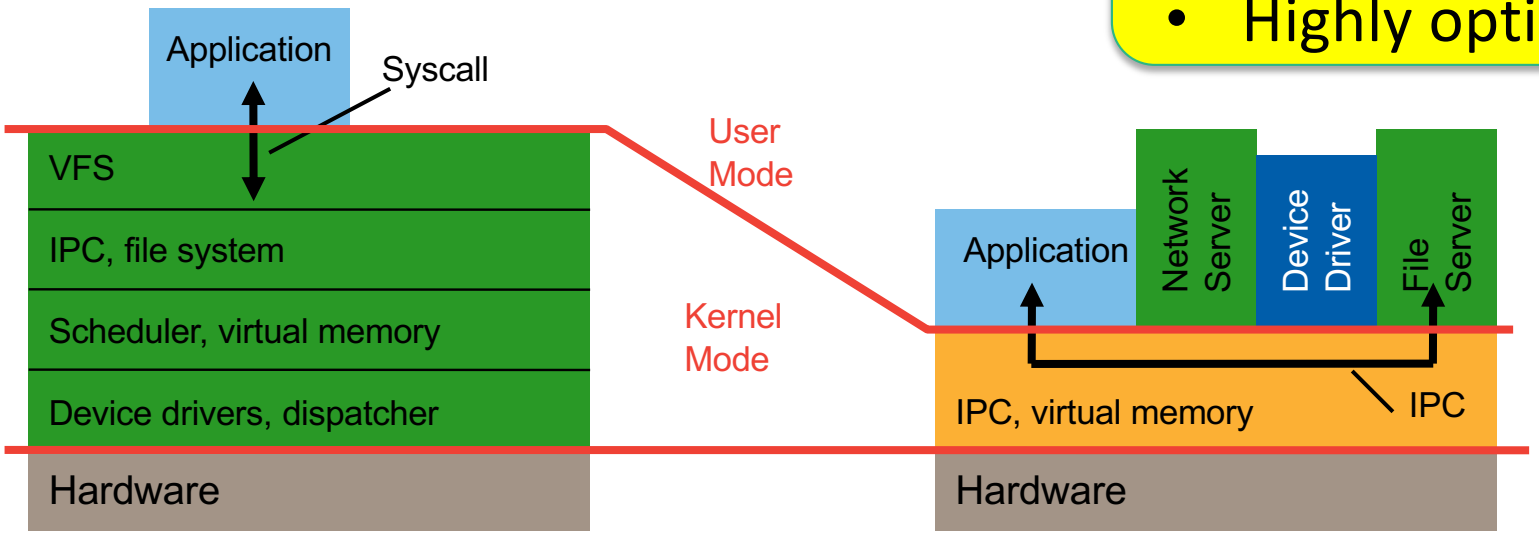
Monolithic OS

- New features add code to kernel
- New policies add code to kernel
- Kernel complexity grows

Microkernel OS

- New features add usermode code
- New policies replace usermode code
- Kernel complexity is stable

- Adaptable
- Dependable
- Highly optimised

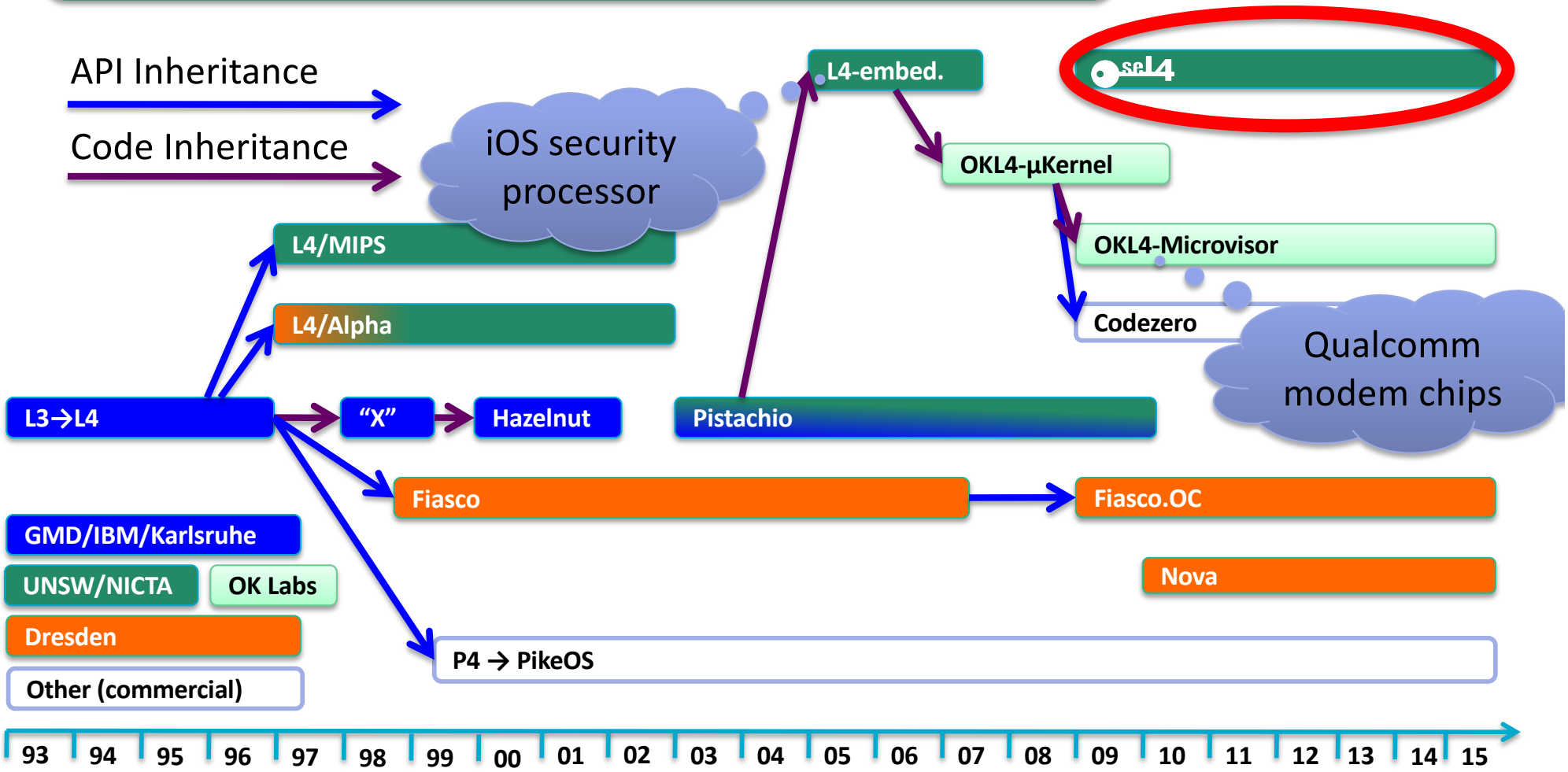


Case in Point: L4 Microkernels

			Size in kSLOC		
Name	**Architecture**	**Year**	**C/C++**	**asm**	**total**
Original	i486	1993	0	6.4	6.4
L4/Alpha	Alpha	1997	0	14.2	14.2
L4/MIPS	MIPS64	1997	6.0	4.5	10.5
Hazelnut	x86	1999	10.0	0.8	10.8
Pistachio	x86	2003	22.4	1.4	23.0
L4-embedded	ARMv5	2005	7.6	1.4	9.0
OKL4 3.0	ARMv6	2007	15.0	0.0	15.0
Fiasco.OC	x86	2008	36.2	1.1	37.6
seL4	ARMv6	2009	9.7	0.5	10.2

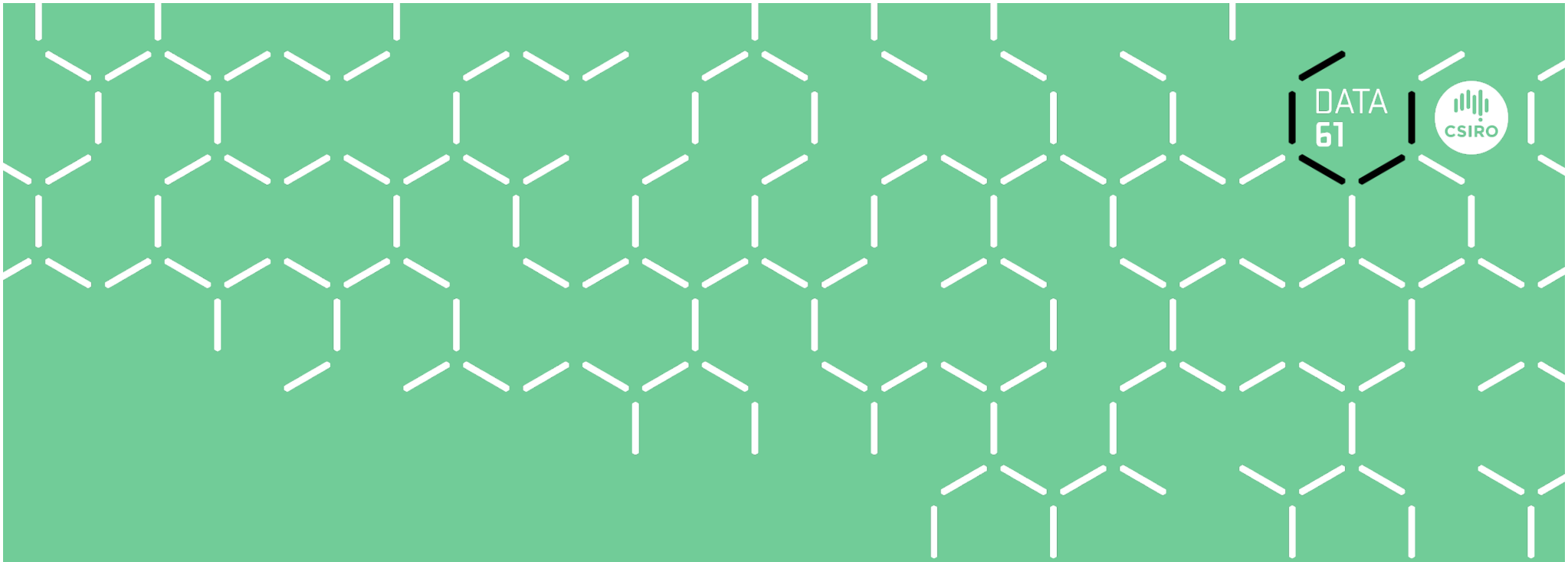
L4 Microkernel Family Tree

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



L4 IPC Performance over 20 Years

Name	Year	Processor	MHz	Cycles	µs
Original	1993	i486	50	250	5.00
Original	1997	Pentium	160	121	0.75
L4/MIPS	1997	R4700	100	86	0.86
L4/Alpha	1997	21064	433	45	0.10
Hazelnut	2002	Pentium 4	1,400	2,000	1.38
Pistachio	2005	Itanium	1,500	36	0.02
OKL4	2007	XScale 255	400	151	0.64
NOVA	2010	i7 Bloomfield (32-bit)	2,660	288	0.11
seL4	2013	i7 Haswell (32-bit)	3,400	301	0.09
seL4	2013	ARM11	532	188	0.35
seL4	2013	Cortex A9	1,000	316	0.32



The seL4 Microkernel

What is seL4?

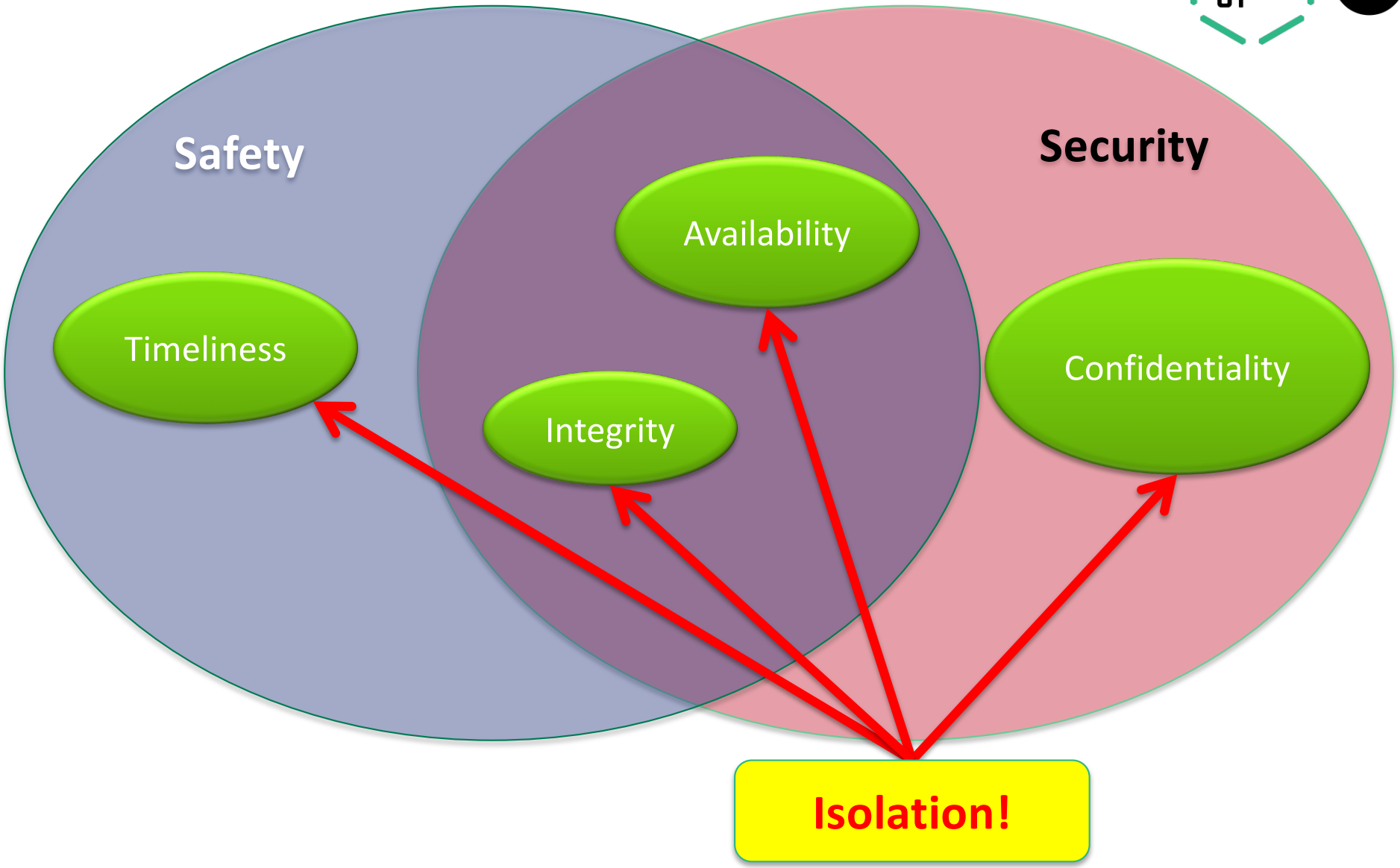
seL4: The world's only operating-system kernel with provable security enforcement

Open Source

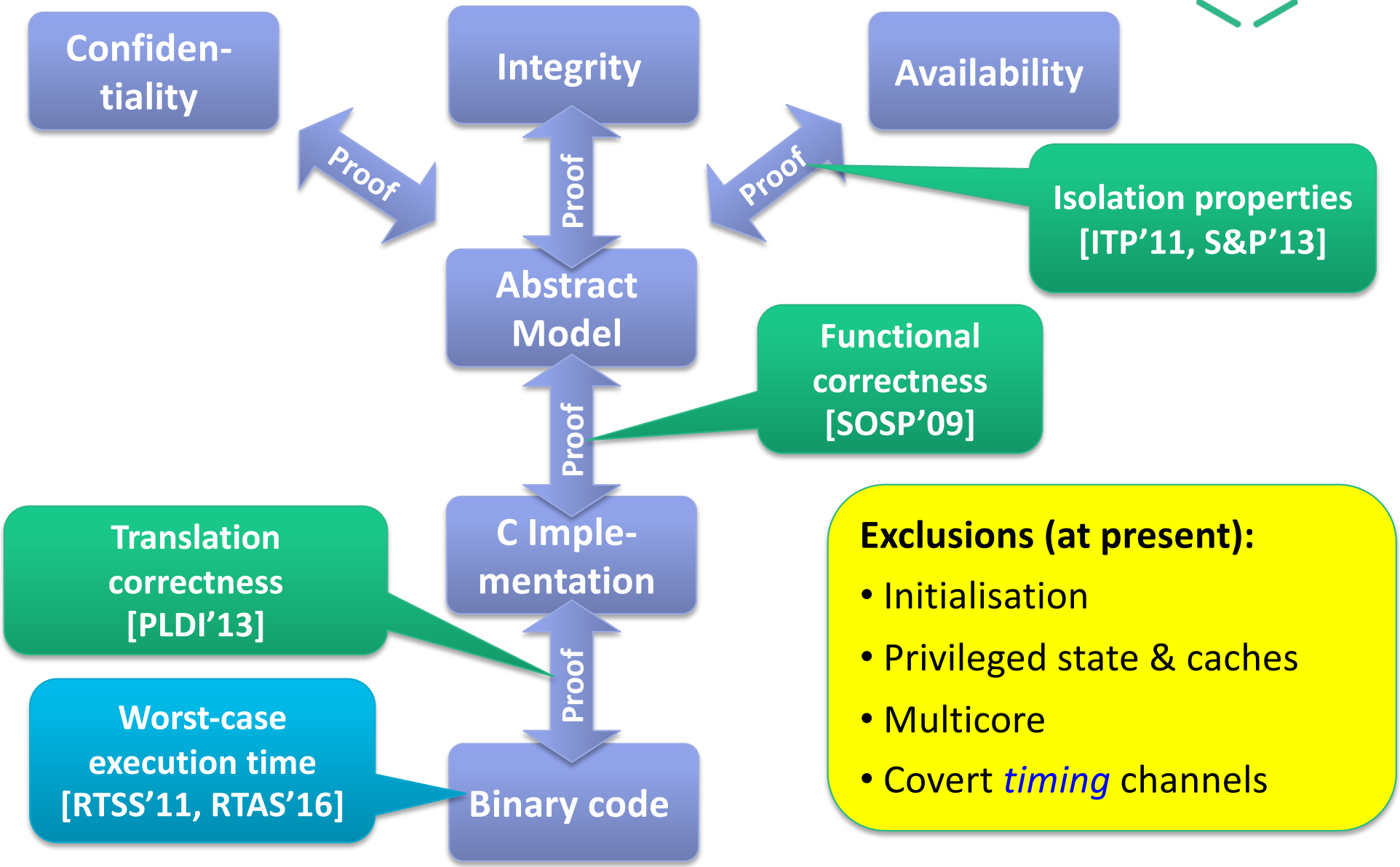
seL4: The world's only protected-mode OS with complete, sound timeliness analysis

seL4: The world's fastest mikrokernel

Security vs Safety



Provable Security and Safety



Exclusions (at present):

- Initialisation
- Privileged state & caches
- Multicore
- Covert *timing* channels

Capability-Based Access Control

Cap = Access Token:
Prima-facie evidence of privilege

Obj reference

Access rights

Object

Eg. thread, address space

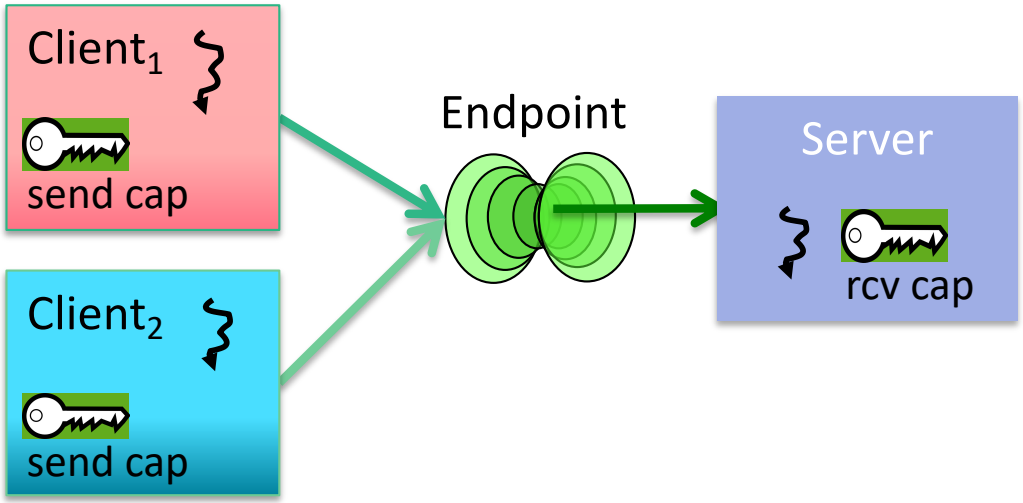
Eg. read, write, send, execute…

Capabilities provide
- Fine-grained access control
- Reasoning about information flow

Any system call is invoking a capability:

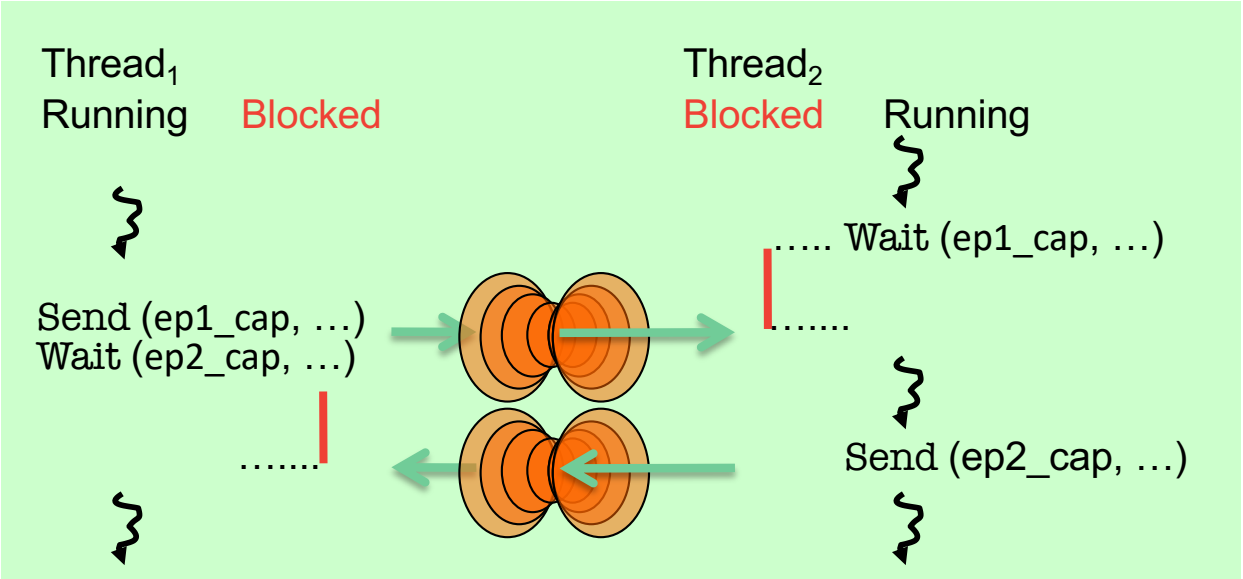
```
err = method( cap, args );
```

Example: IPC (Message Passing)



Only holder of a send capability to the server's endpoint can talk to the server!

IPC Is Synchronous (Blocking)

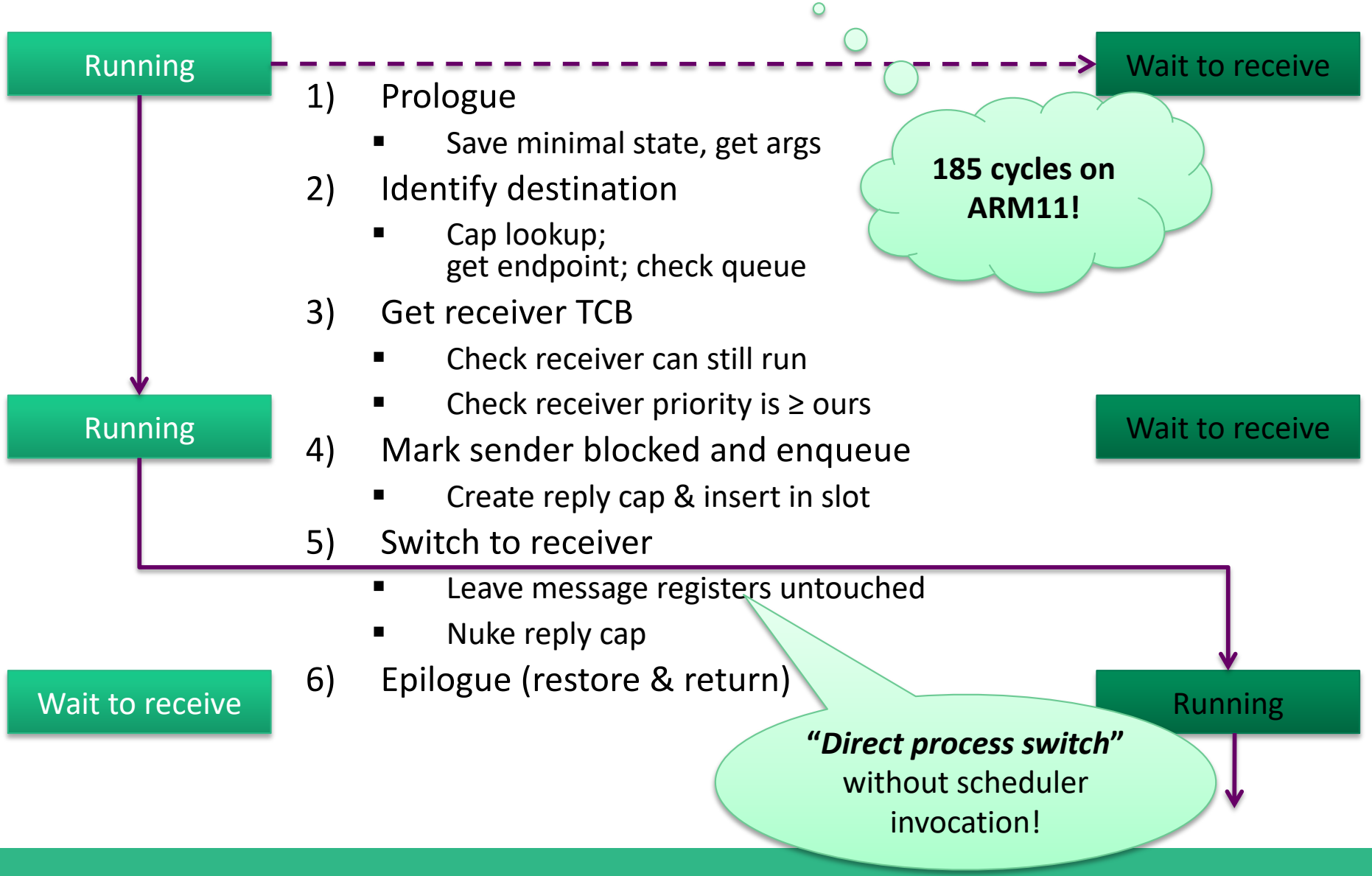


Threads must rendez-vous for message transfer

- One side blocks until the other is ready
- Implicit synchronisation

Leads to Simple Implementation

Simple send (e.g. as part of RPC-like “call”):



Fastpath Coding Tricks


```
slow =    cap_get_capType(en_c) != cap_endpoint_cap ||
          !cap_endpoint_cap_get_capCanSend(en_c);
if (slow)   enter_slow_path();
```

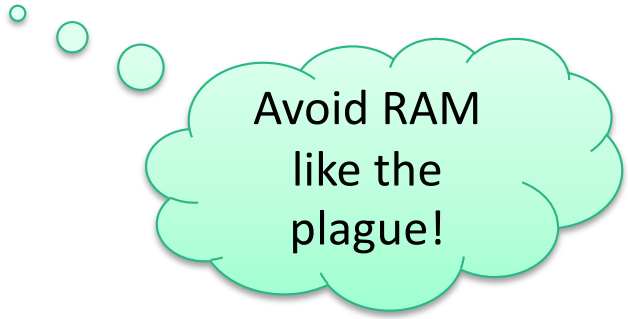
Common case: 0

Common case: 1

- Reduces branch-prediction footprint
- Avoids mispredicts, stalls & flushes
- Uses ARM instruction predication

Other implementation tricks

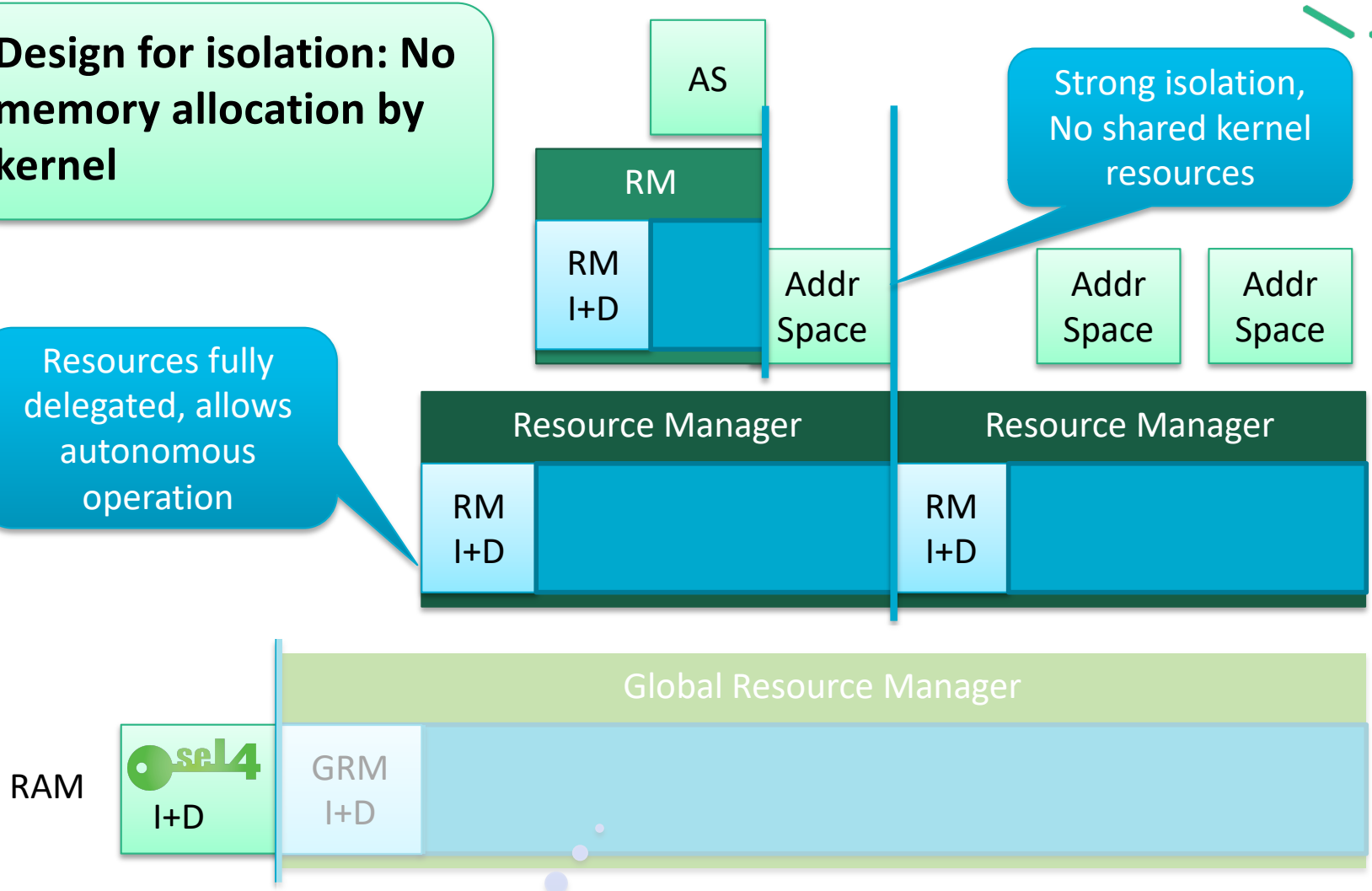
- Cache-friendly data structure layout, especially TCBs
  - data likely used together is on same cache line
  - helps best-case and worst-case performance



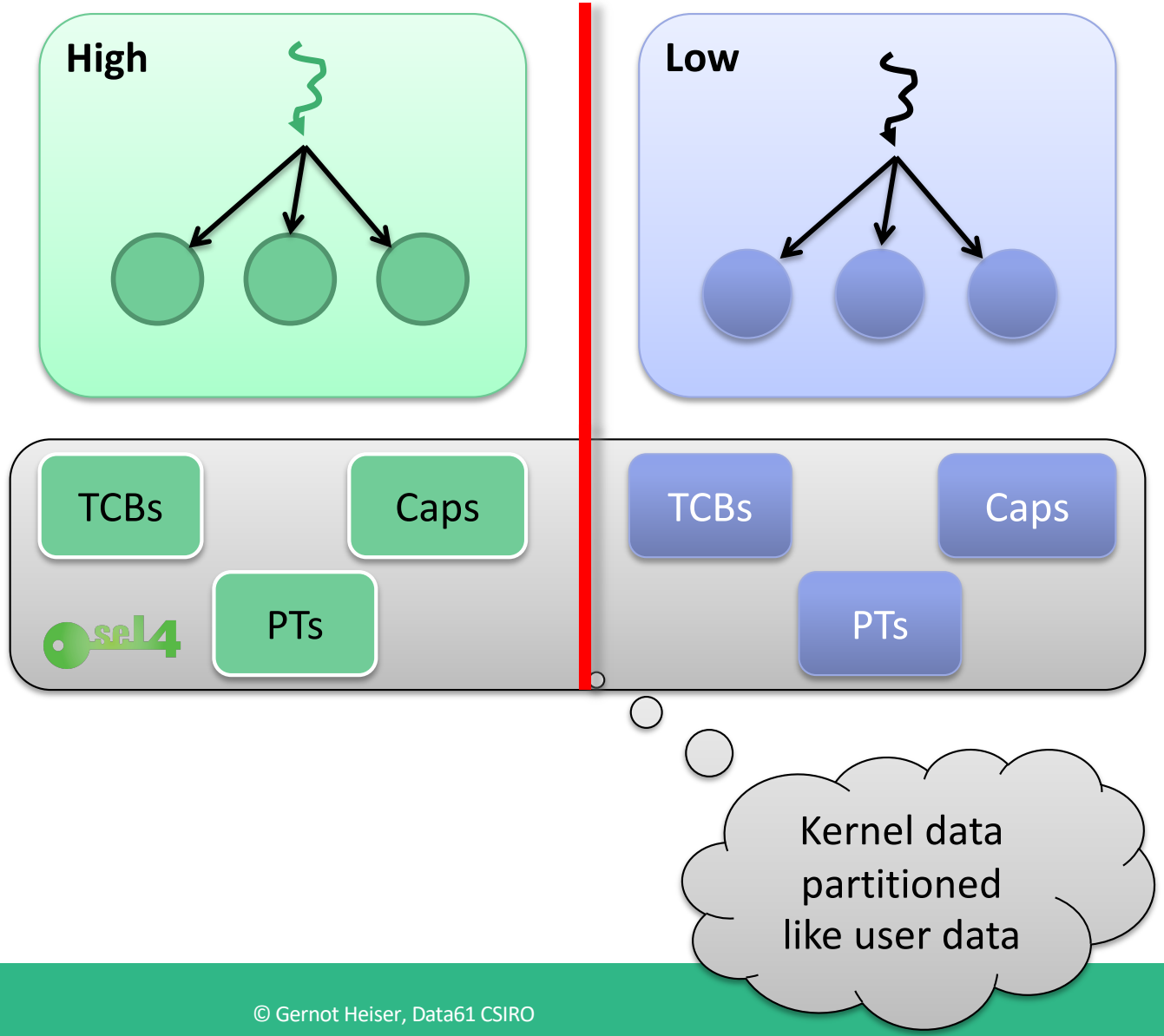
- Kernel mappings locked in TLB (using superpages)
  - helps worst-case performance
  - helps establish invariants: page table never walked when in kernel

What's Different to Other Microkernels?

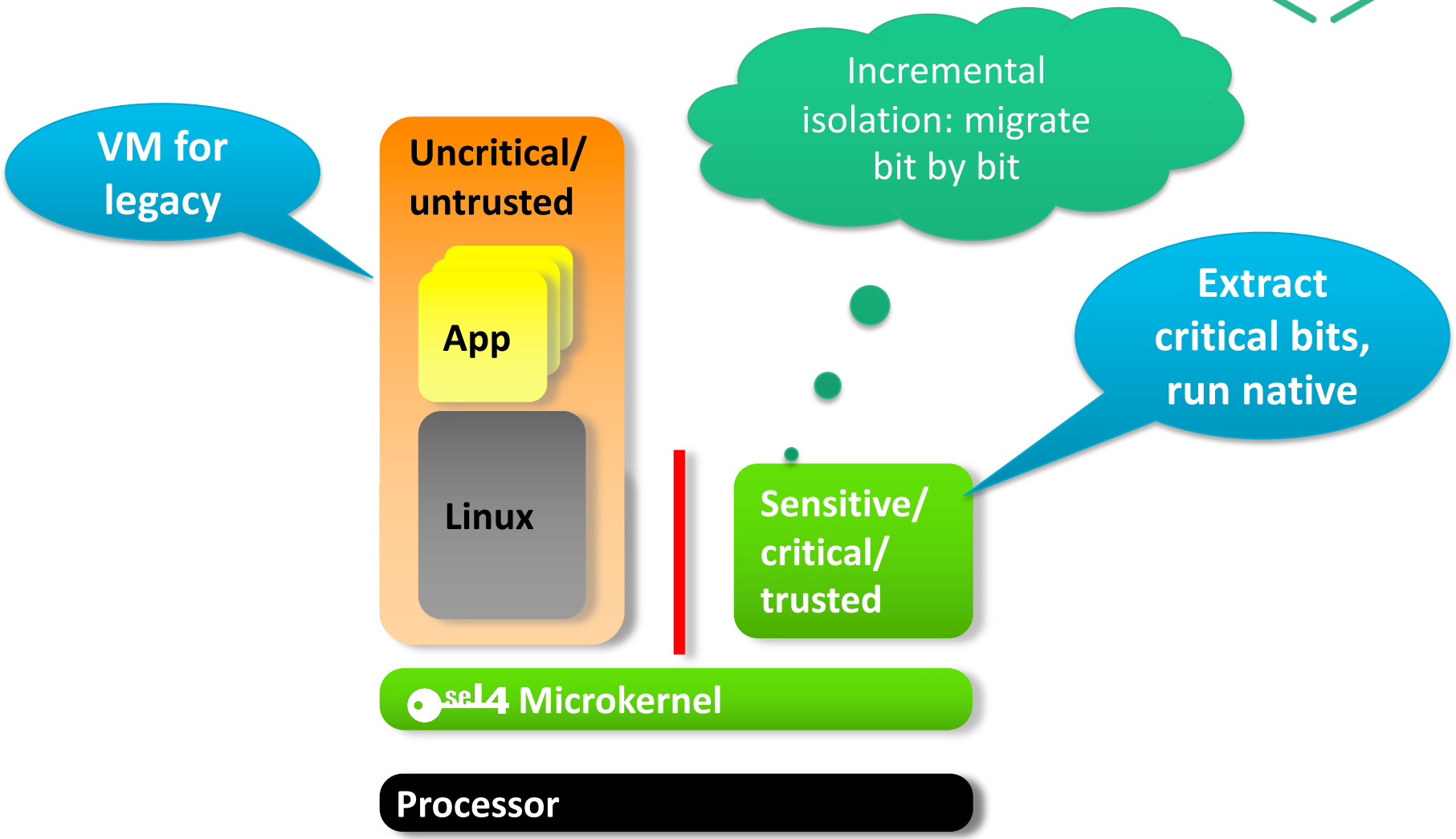
Design for isolation: No memory allocation by kernel



seL4 Isolation Goes Deep



Building Trustworthy Systems



Critical Systems: DARPA HACMS

Boeing Unmanned Little Bird

Retrofit existing system!

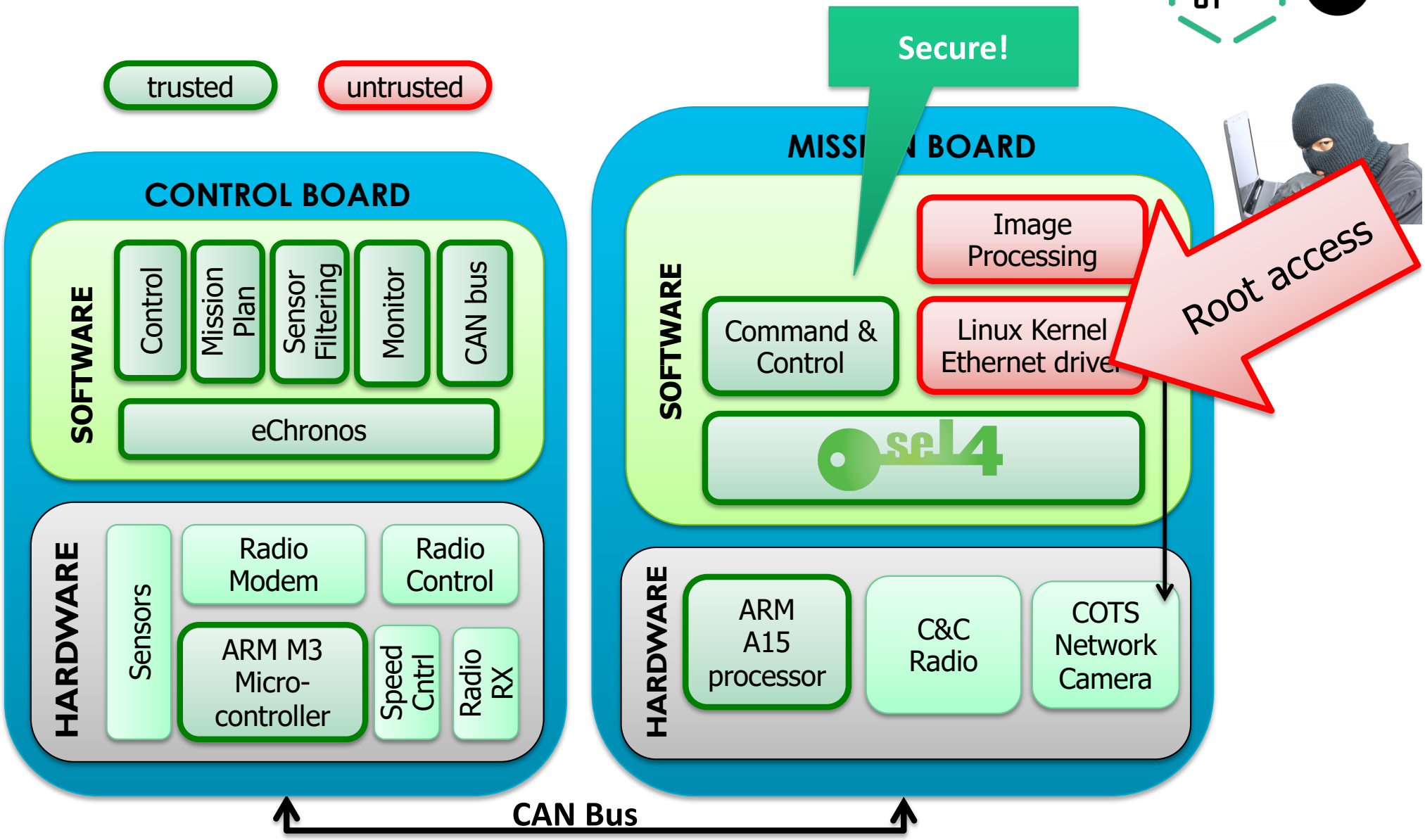
US Army Autonomous Trucks

SMACCMcopter Research Vehicle

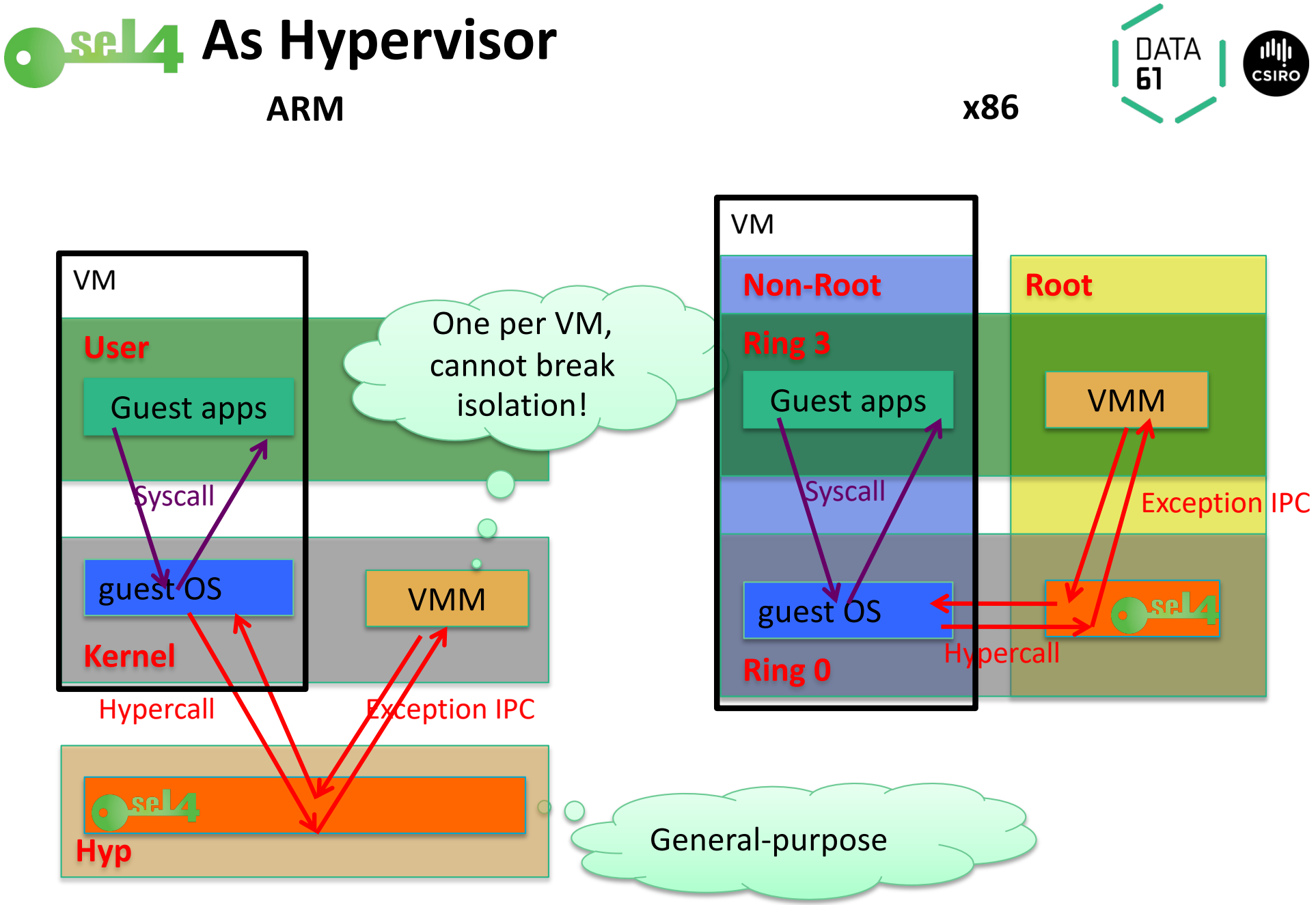
Develop technology

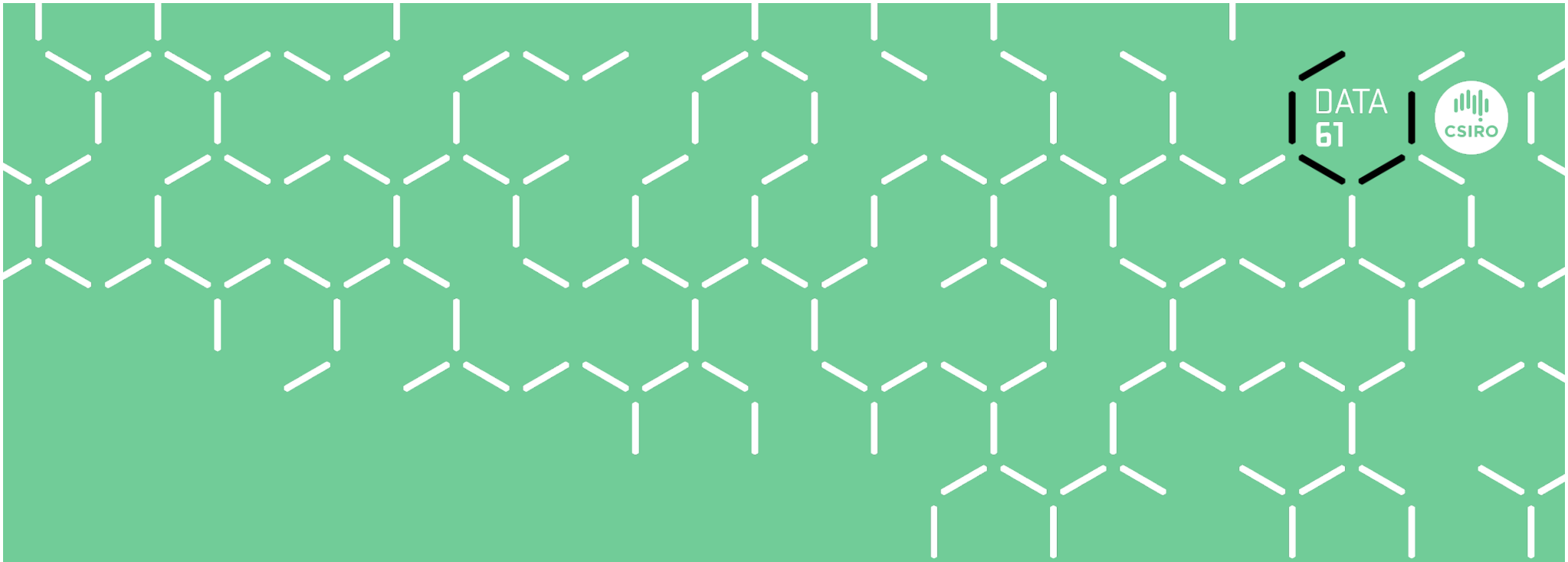
TARDEC GVRbot

SMACCMcopter Architecture



As Hypervisor





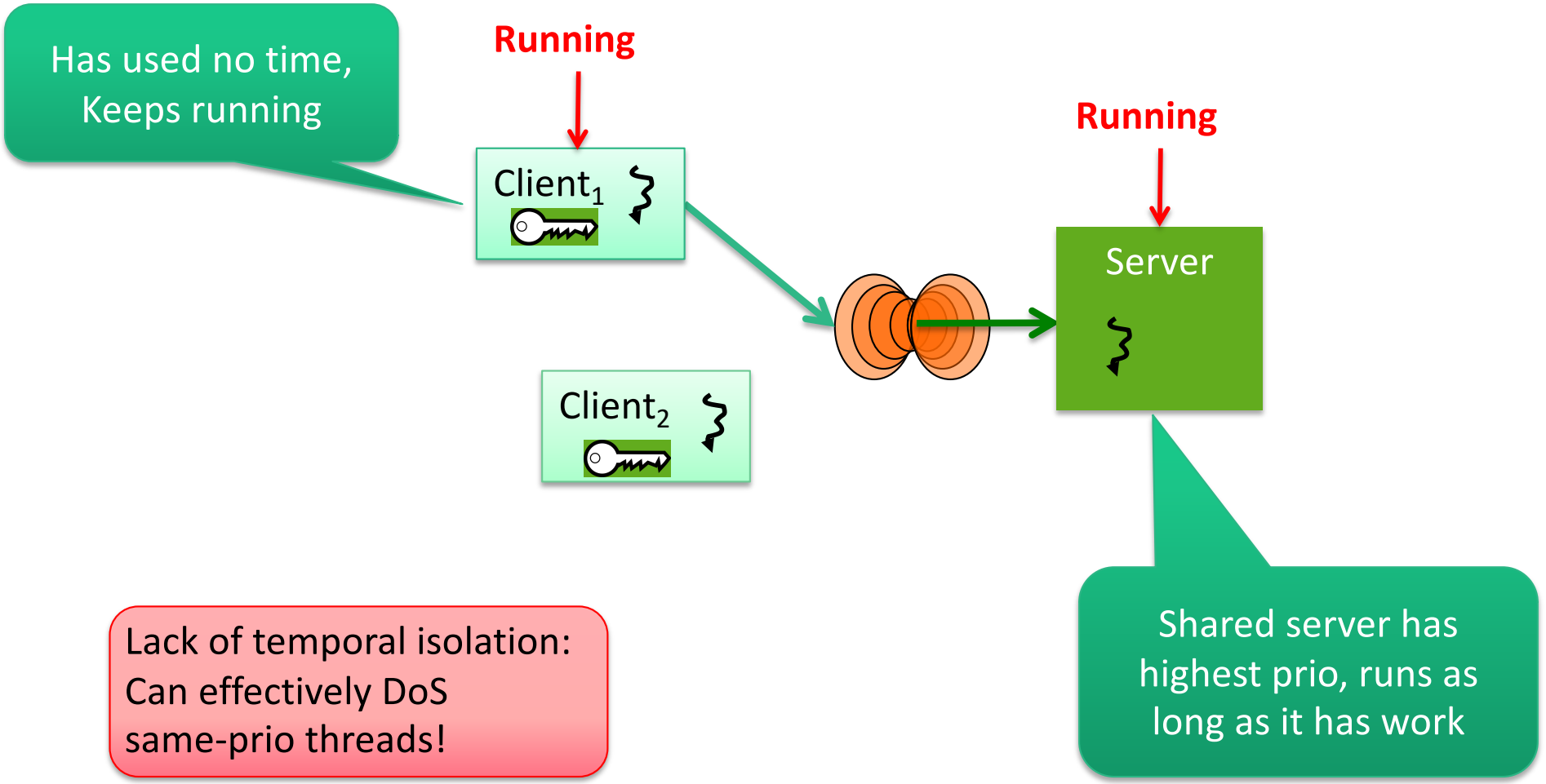
Requirements for an Operating System for the Future

Requirements for A Future-Proof OS

- Strong security, including protection against side channels
  - ✓ seL4 is only OS with mathematically proved isolation
  - ✓ Unique timing-channel defence mechanisms in development
- Hard real-time support, fast interrupt handling
  - ✓ seL4 is only protected-mode OS with complete and sound timing analysis
- Scalable multicore support
  - ✓ Clustered multikernel for low-overhead, scalable multicore support

Real-Time Issues

Problem With Servers As Threads



Separate Scheduling from Thread

Classical Thread Attributes:

- Priority
- Time slice

Not runnable if null

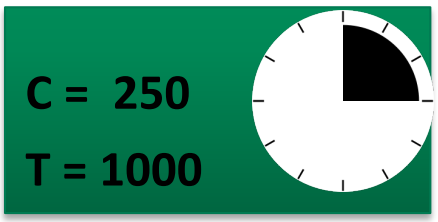
New Thread Attributes:

- Priority
- Scheduling context capability

Upper bound, not reservation!

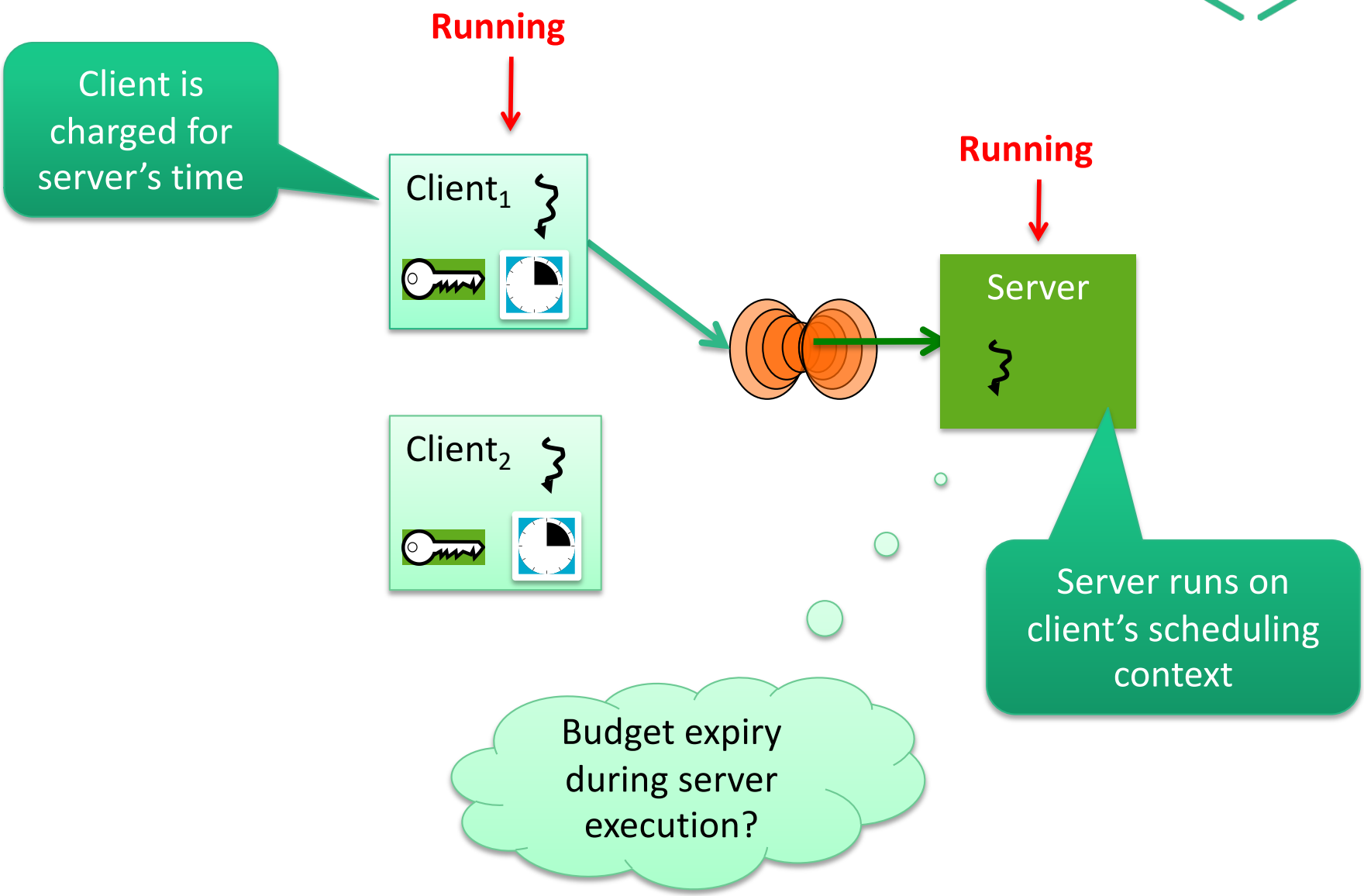
Scheduling context object
- T: period
- C: budget (≤ T)

Not yet in mainline!



SchedControl capability conveys right to assign budgets (i.e. perform admission control)

Shared Server w Scheduling Contexts

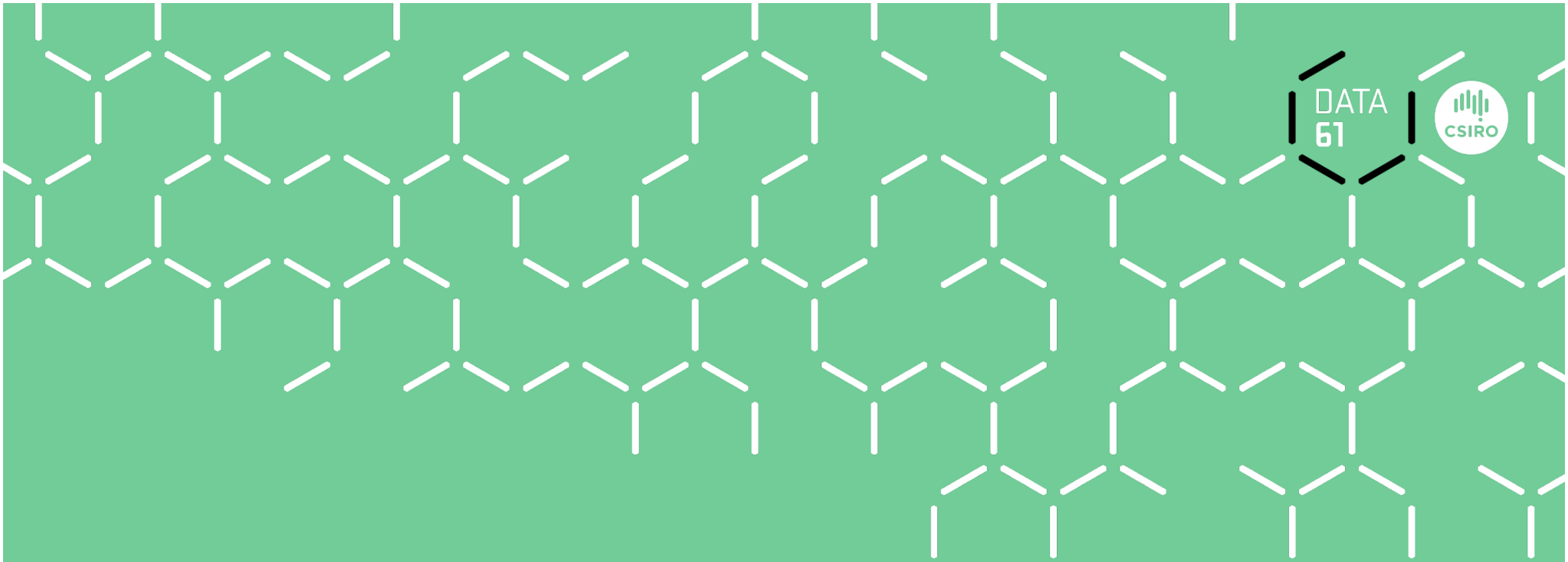


Budget Expiry Options

- Multi-threaded servers (COMPOSITE [Parmer '10])
  - Model allows this
  - Forcing all servers to be thread-safe is policy 😢

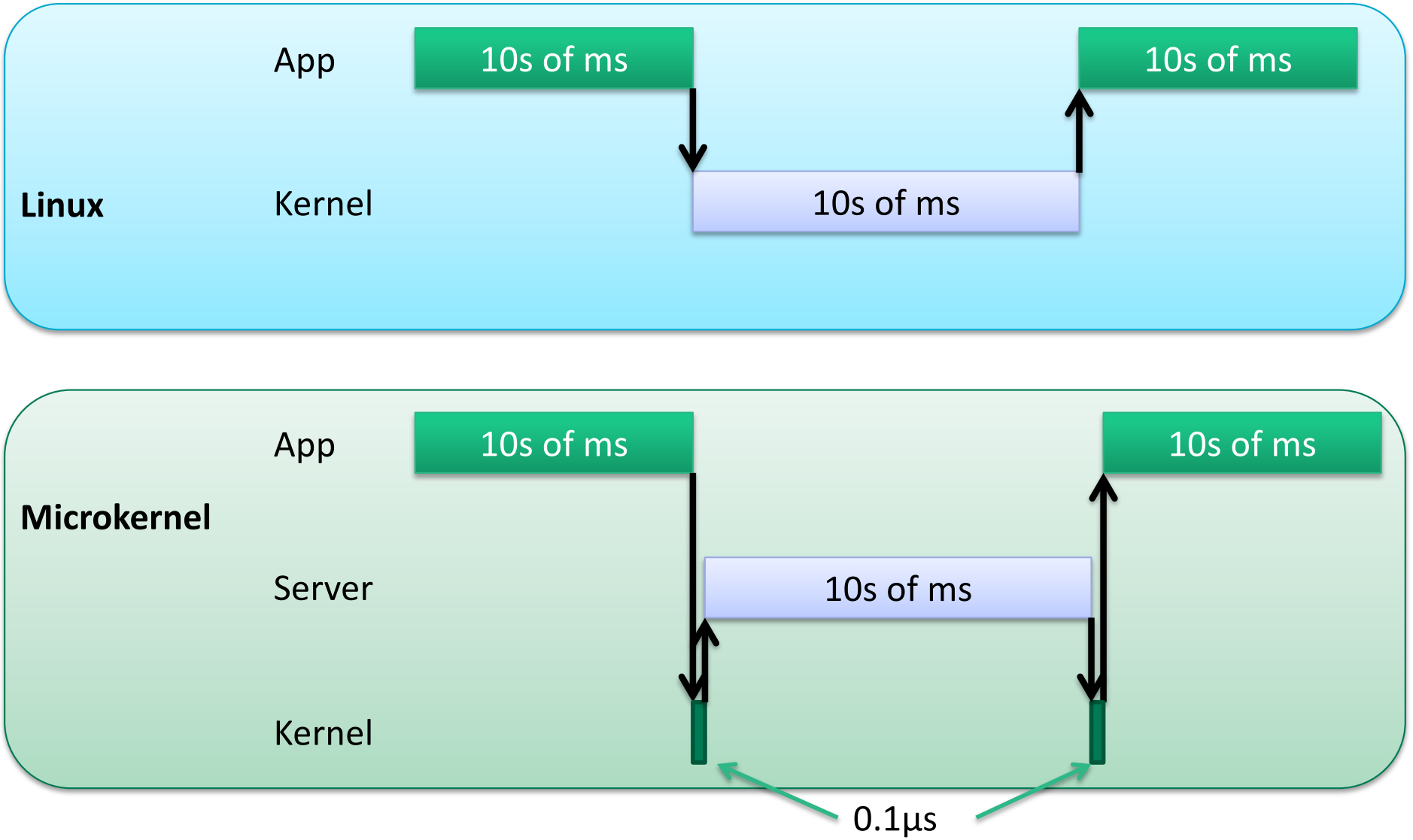
- Bandwidth inheritance with "helping" (Fiasco [Steinberg '10])
  - Ugly dependency chains 😢
  - Wrong thread charged for recovery cost 😢

- Use *timeout exceptions* to trigger one of several possible actions:
  - Provide emergency budget
  - Cancel operation & roll-back server
  - Change criticality
  - Implement priority inheritance (if you must…)

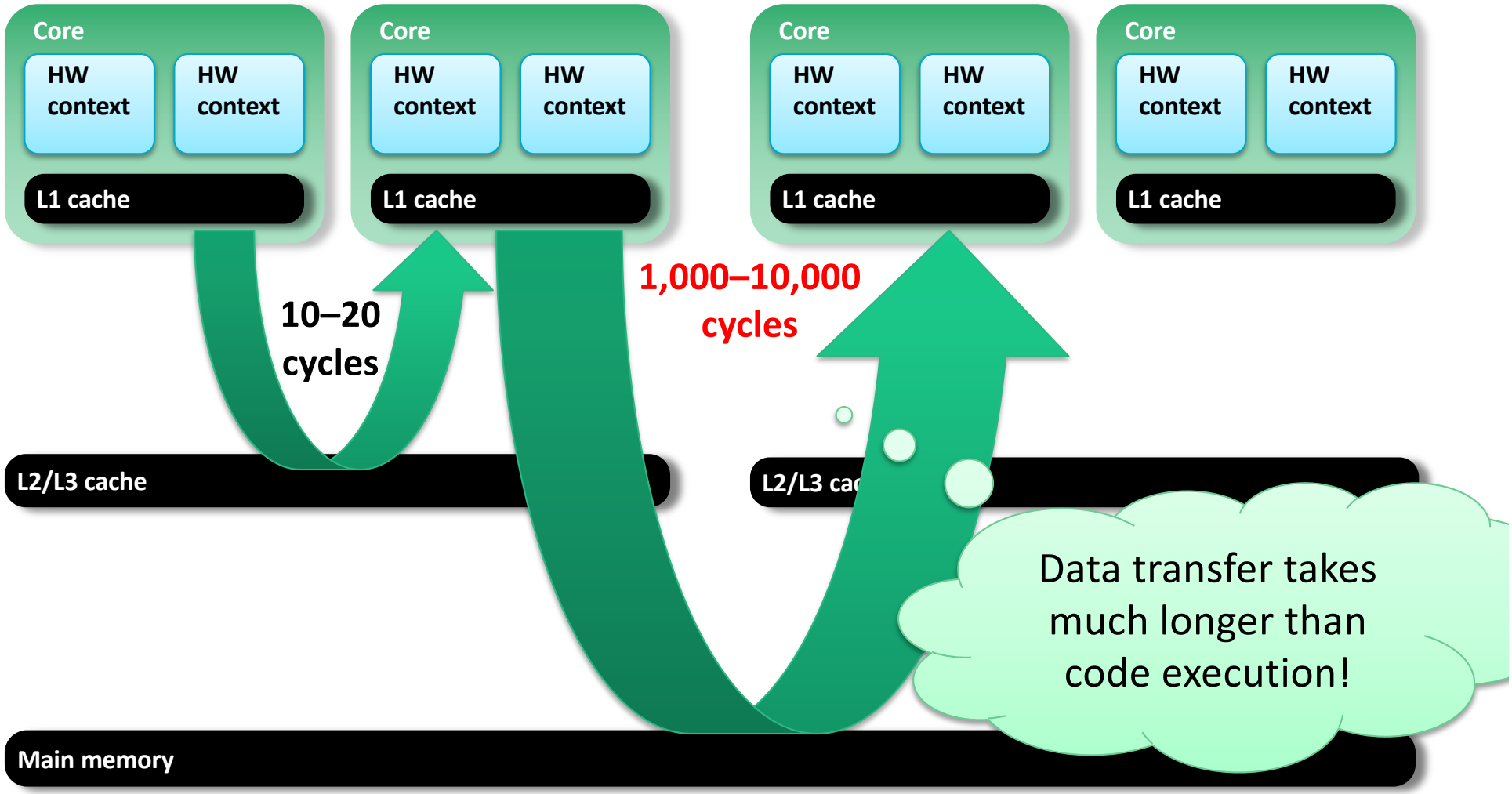


Multicore

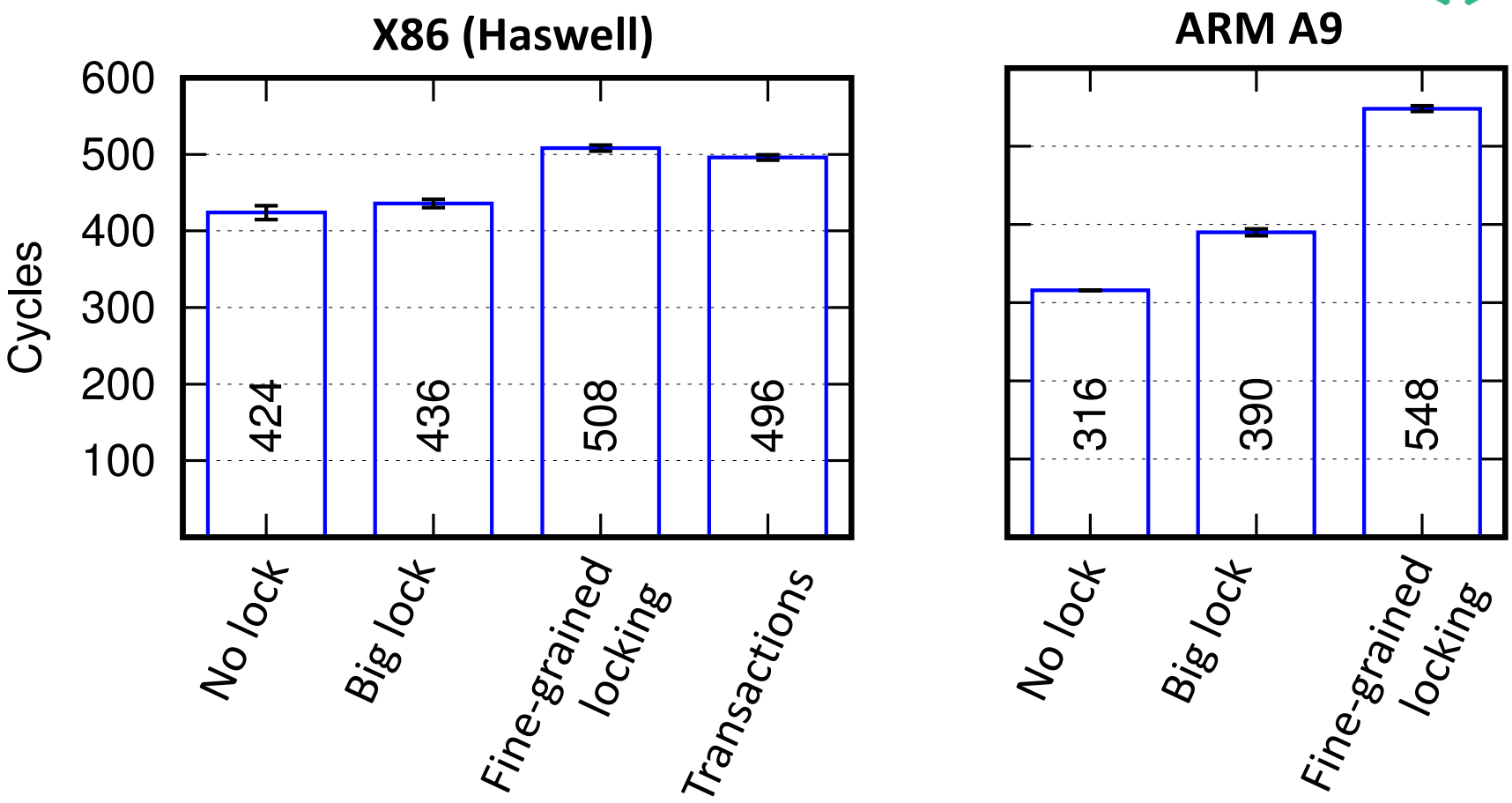
Microkernel vs Linux Execution



Cache Line Migration Latencies

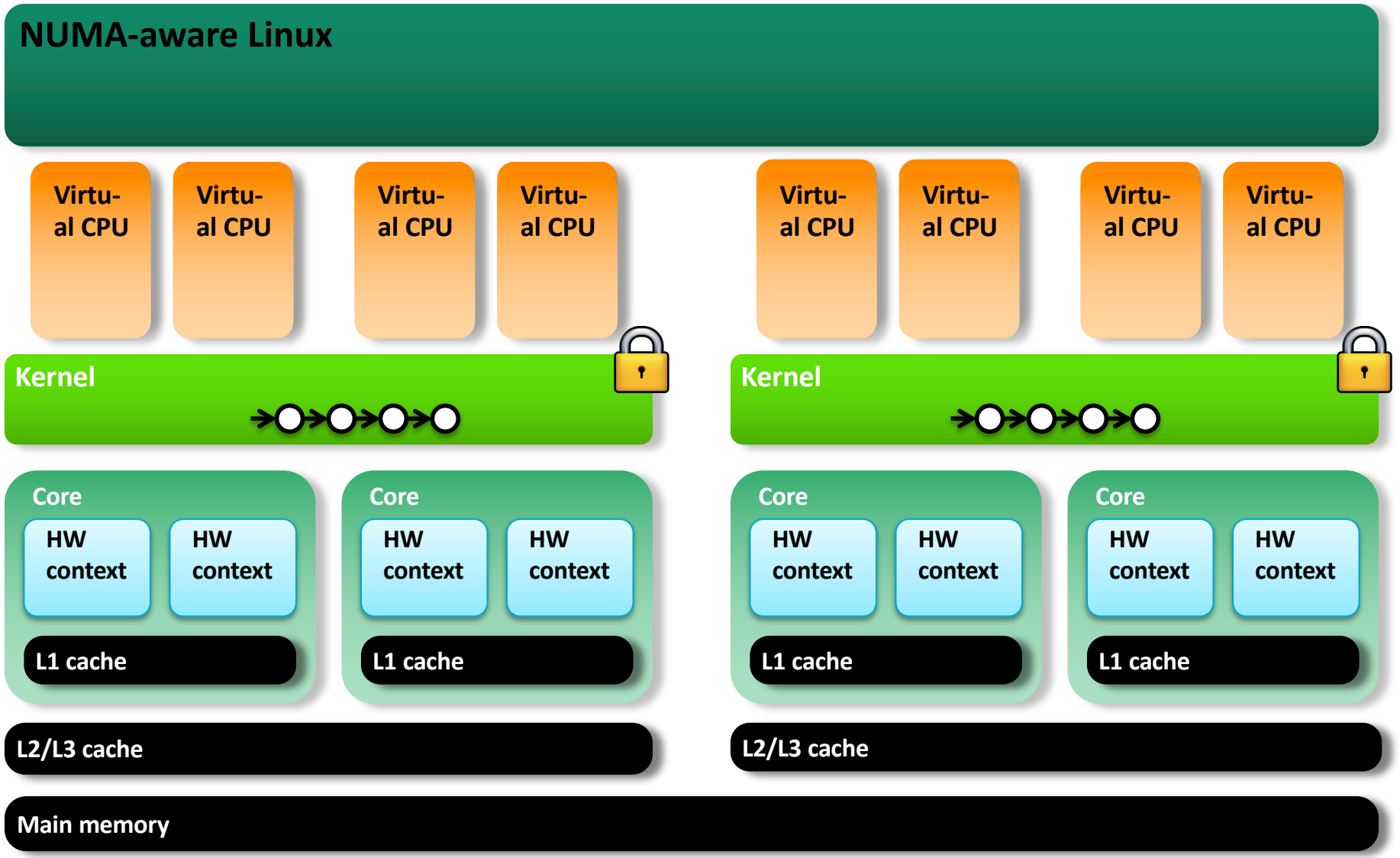


Cost of Locking

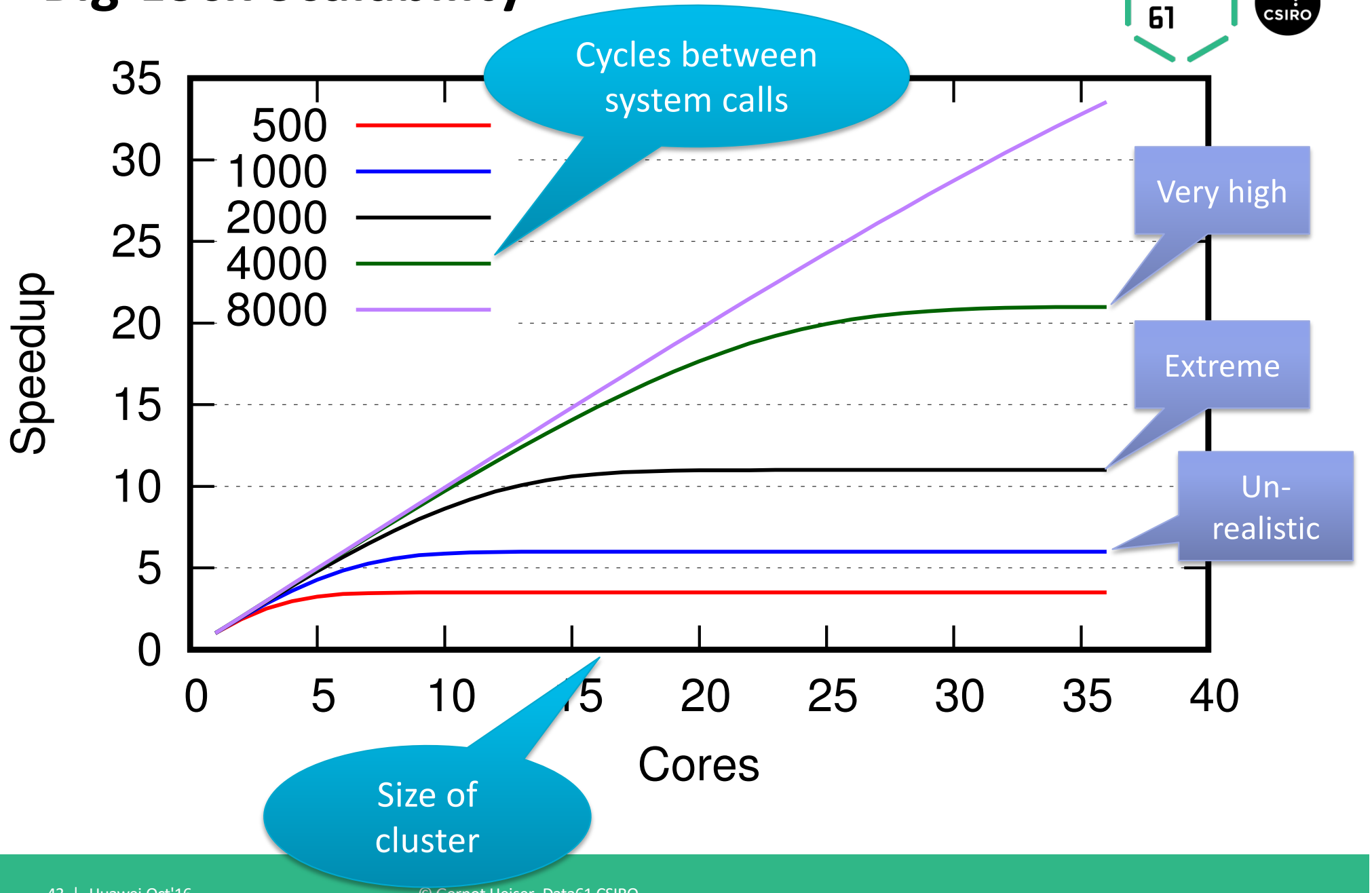


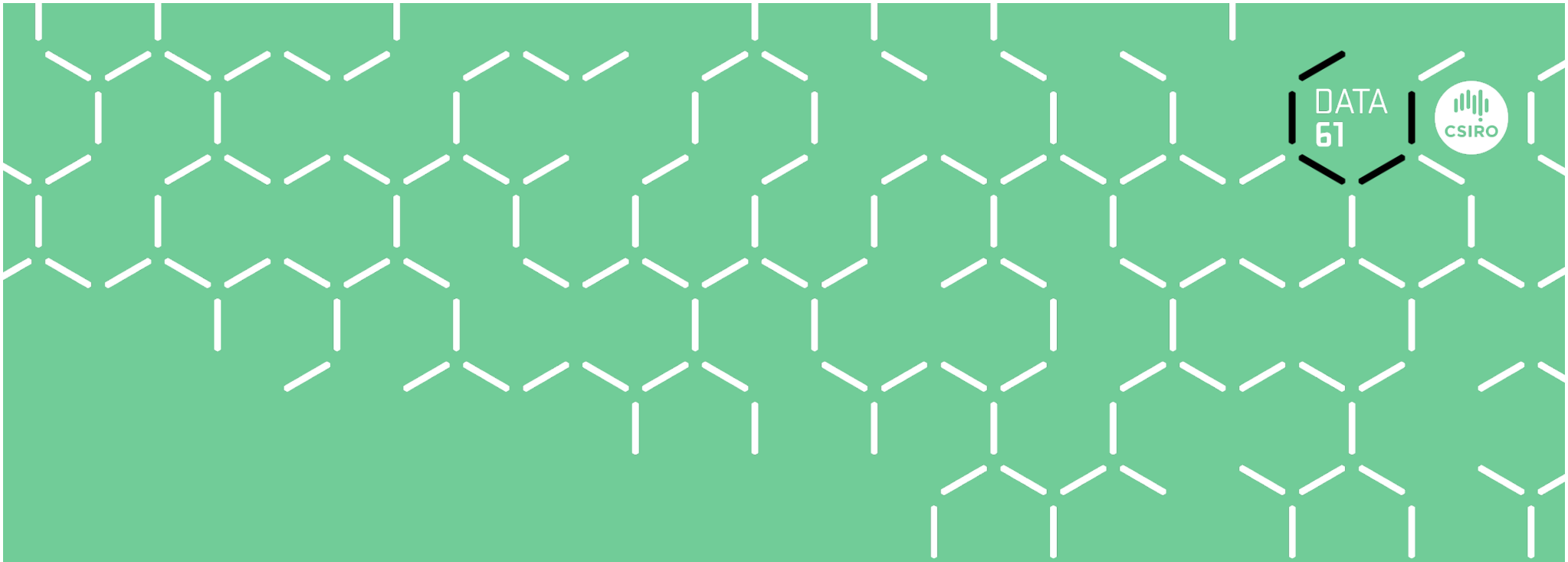
Locks have a cost –
significant in a fast microkernel!

seL4 Multicore Design: Clustered Multikernel



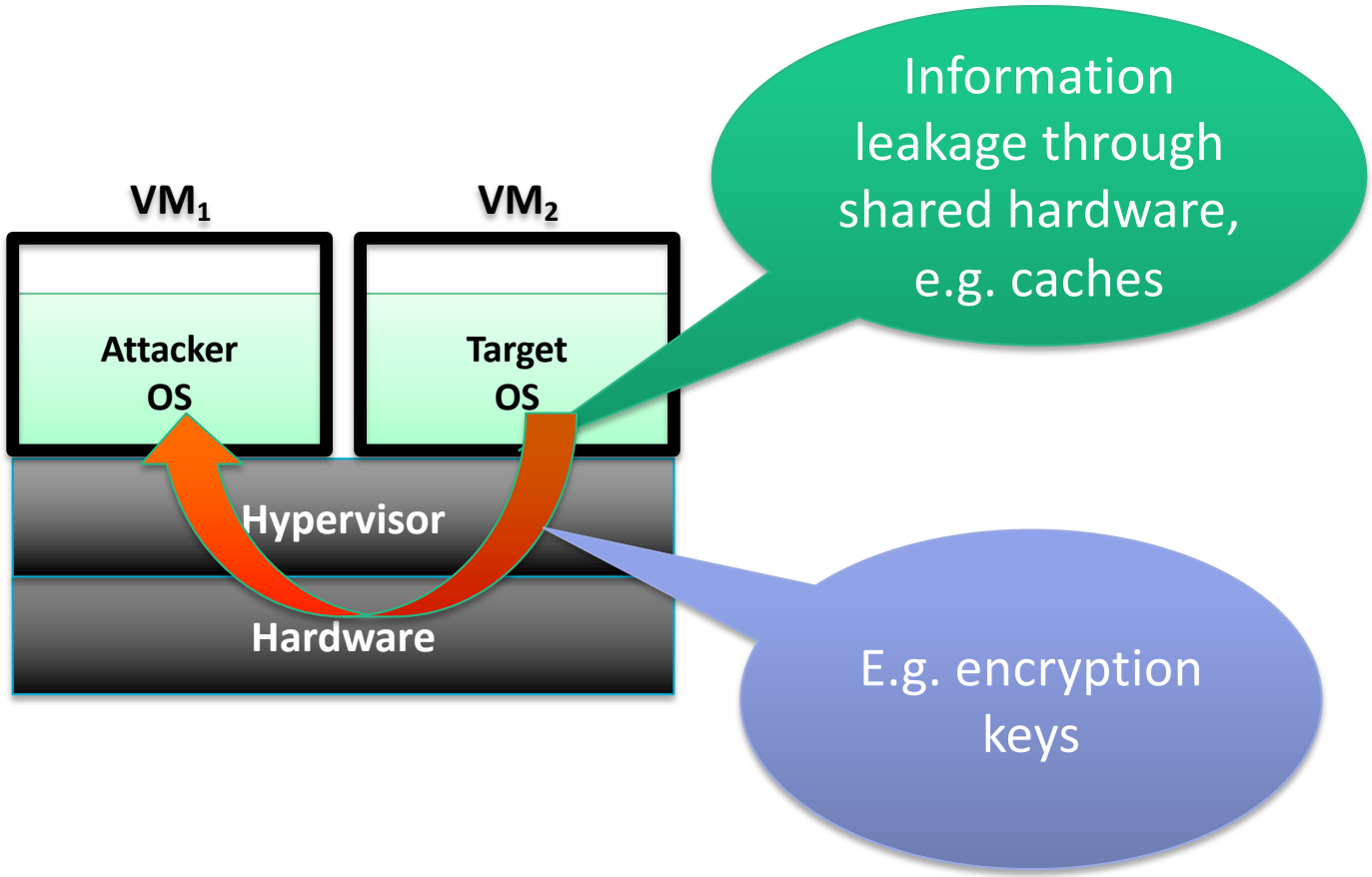
Big-Lock Scalability





Side Channels

Side Channel Attacks



Types of Side Channels

Storage Channels

- Use some shared state
- Could be inside the OS/hypervisor
  - Eg existence of a file
  - Eg accessibility of an object

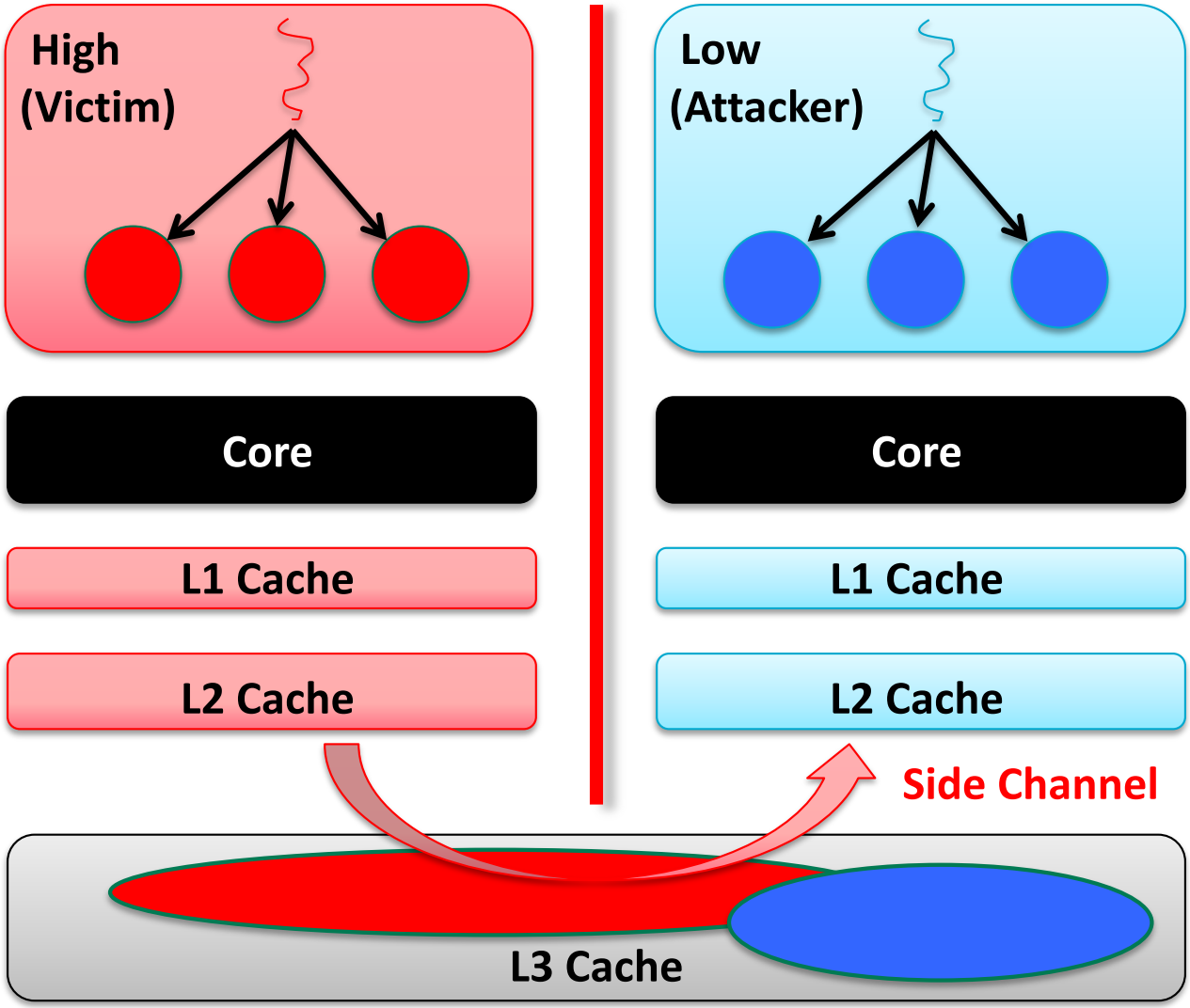
Timing Channels

- Observe timing of events
- Eg memory access latency
  - Senses victim's cache footprint

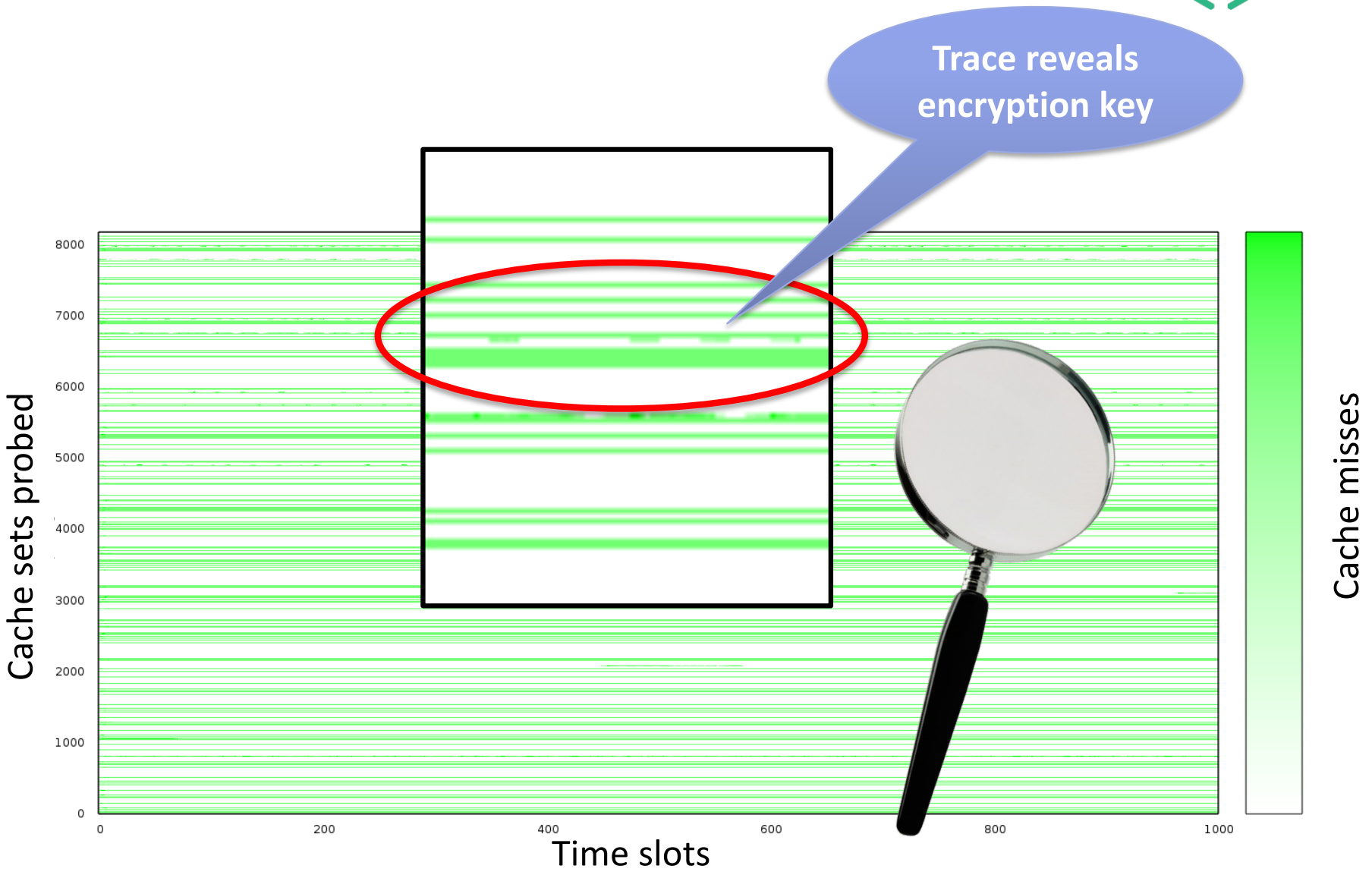
seL4: The world's only OS proved free of storage channels!

How about timing channels?

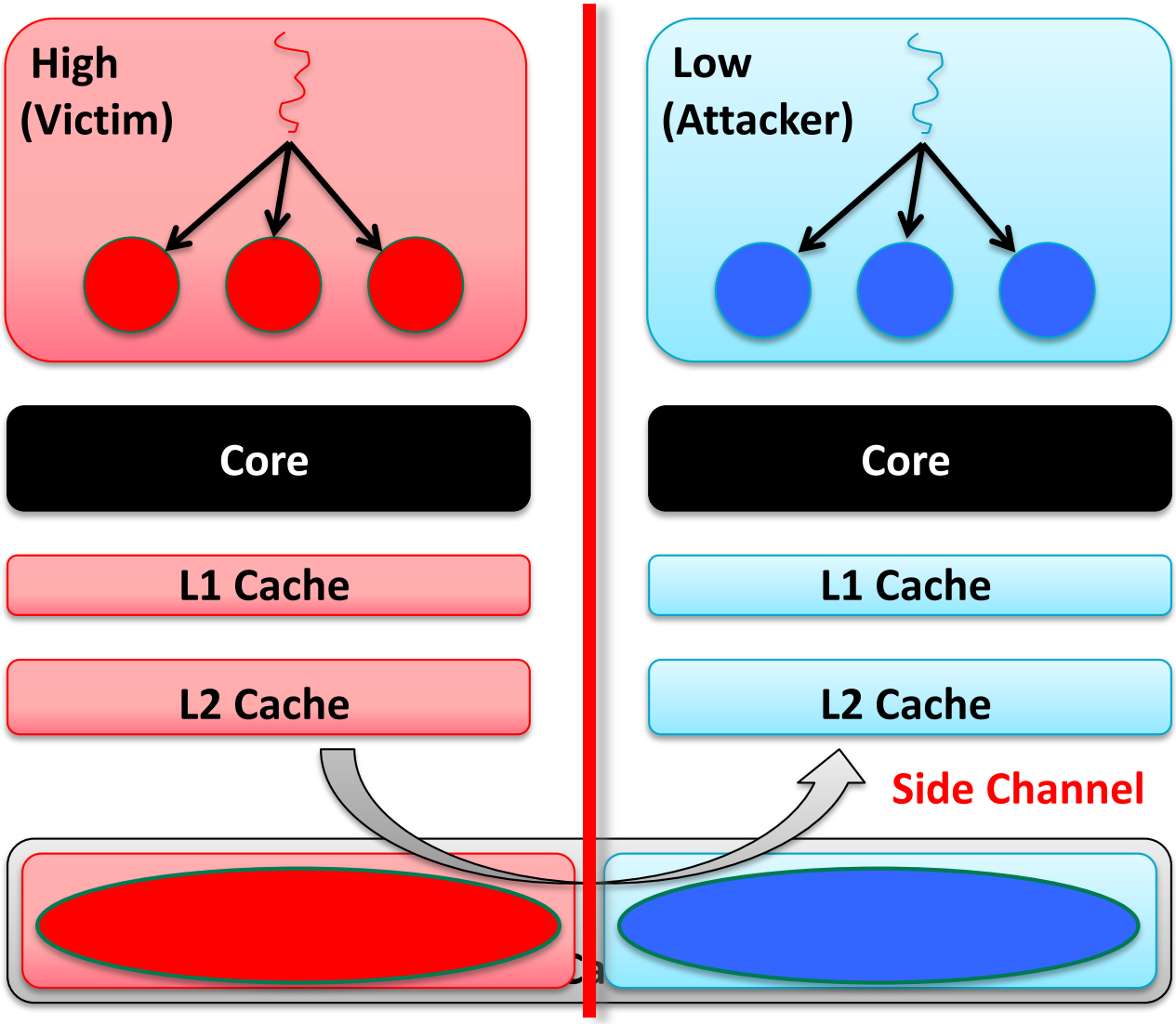
Timing Side-Channel Attack in Public Cloud



Analysing Memory Access Latency

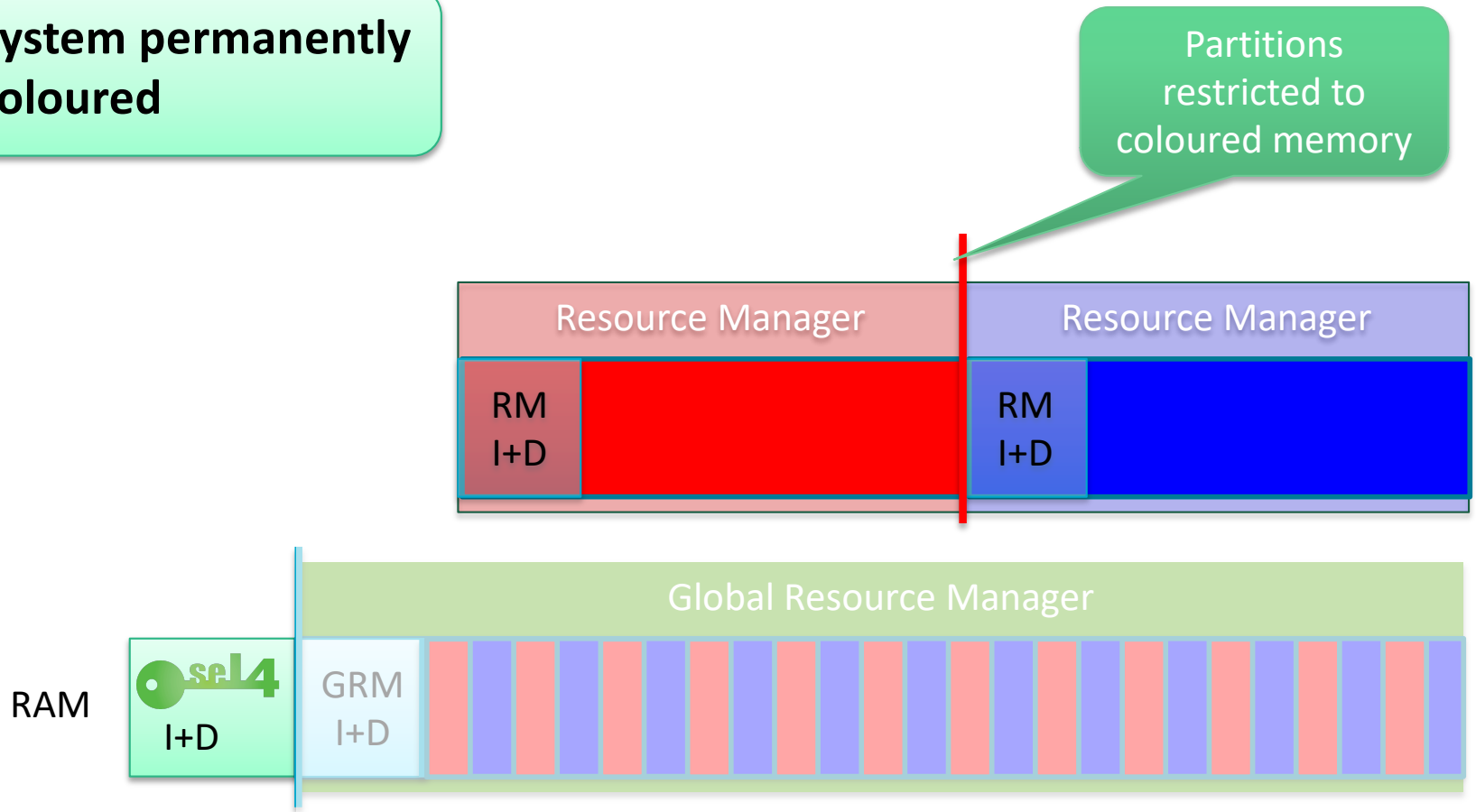


Mitigation: Partition Cache (Colouring)



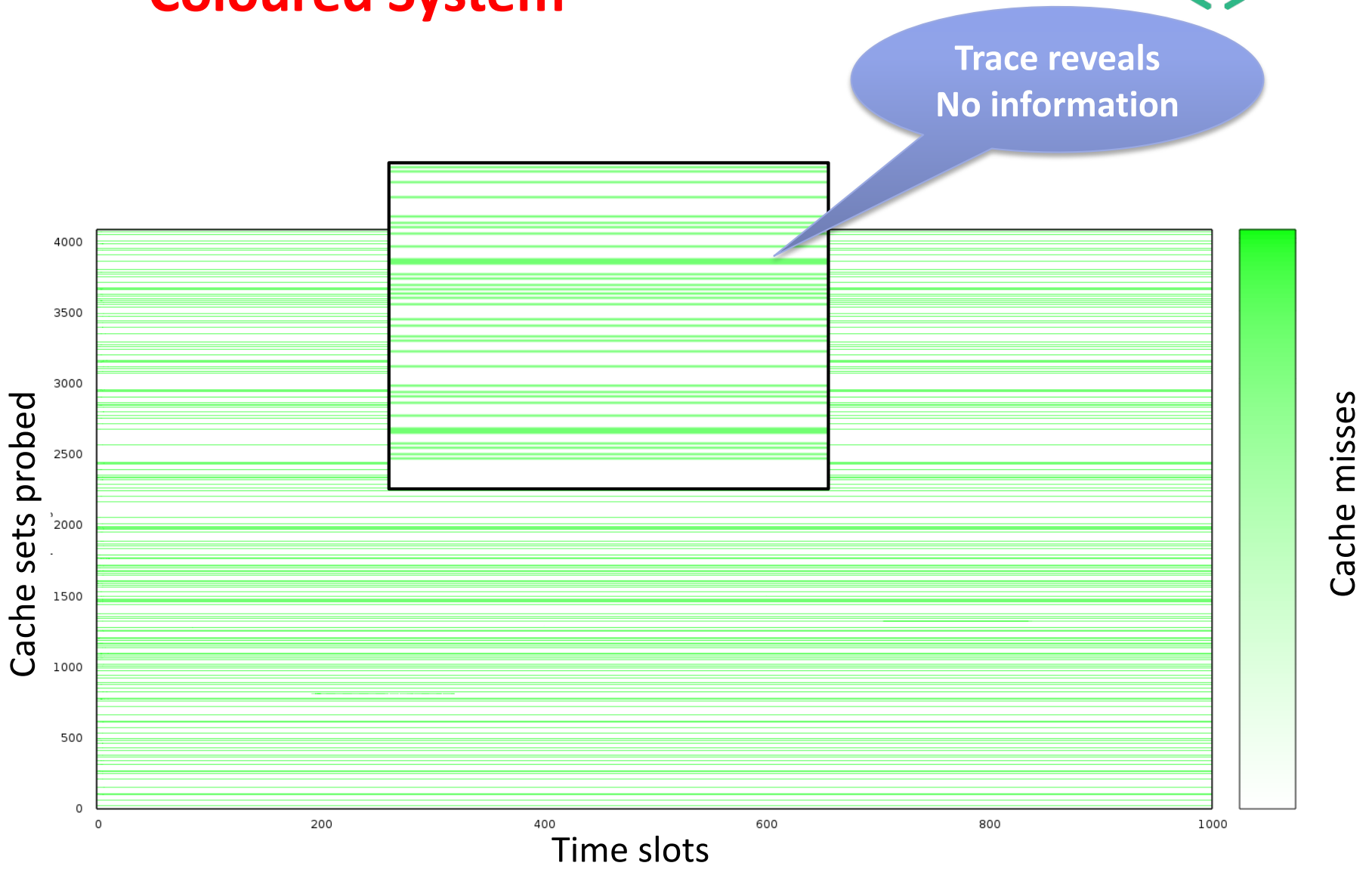
Colouring the System is Easy

System permanently coloured

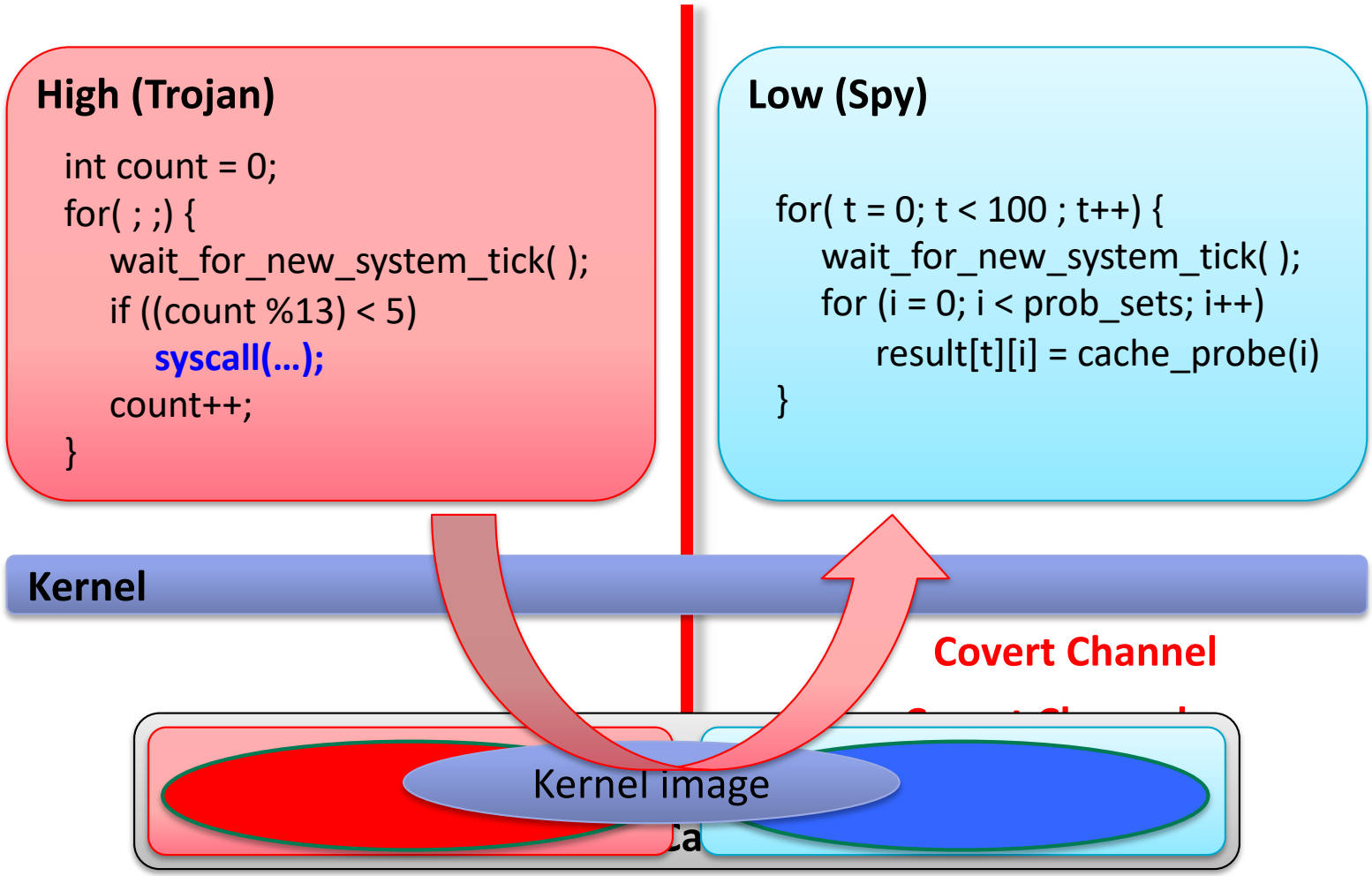


Analysing Memory Access Latency

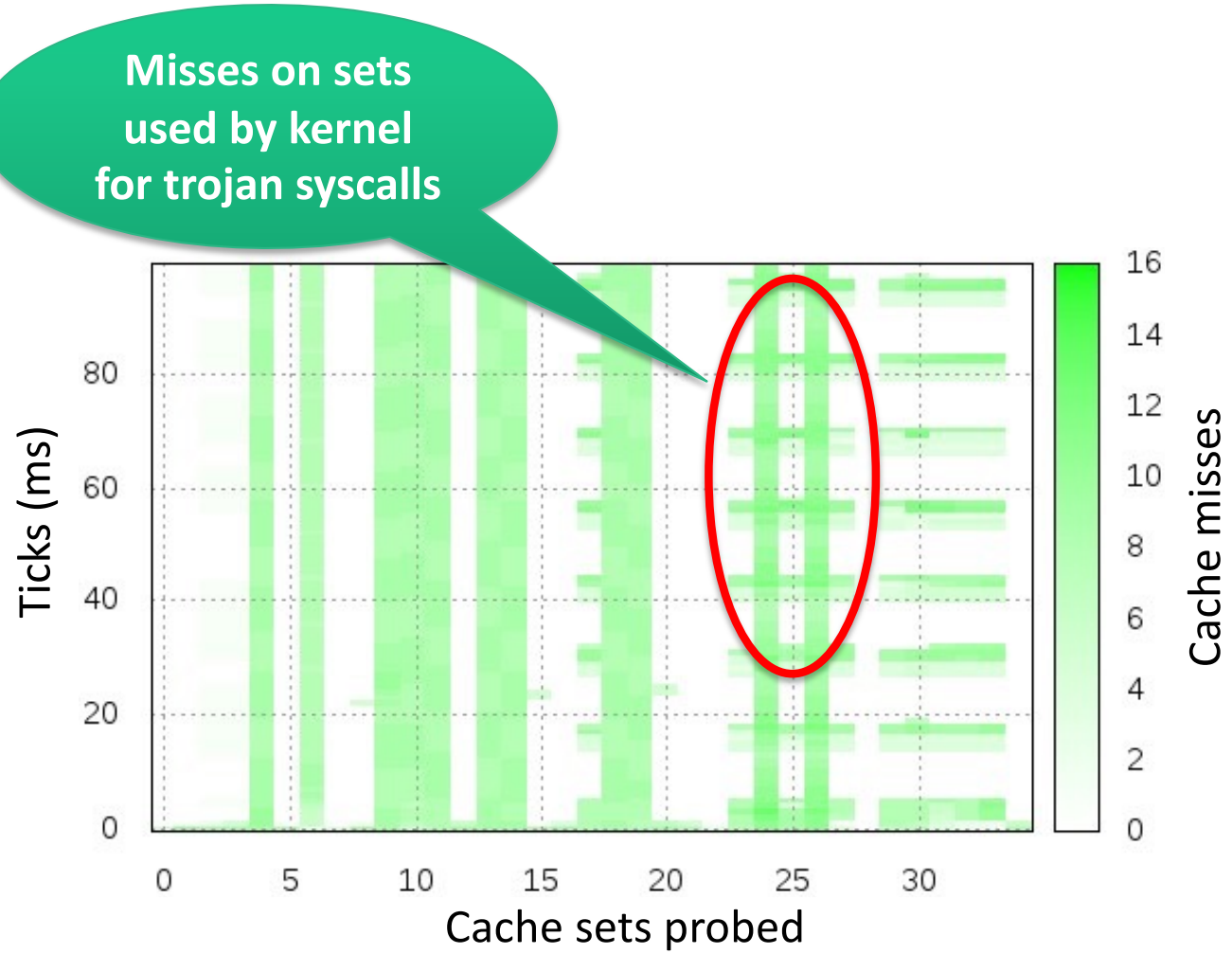
Coloured System



Timing Channel Through Kernel



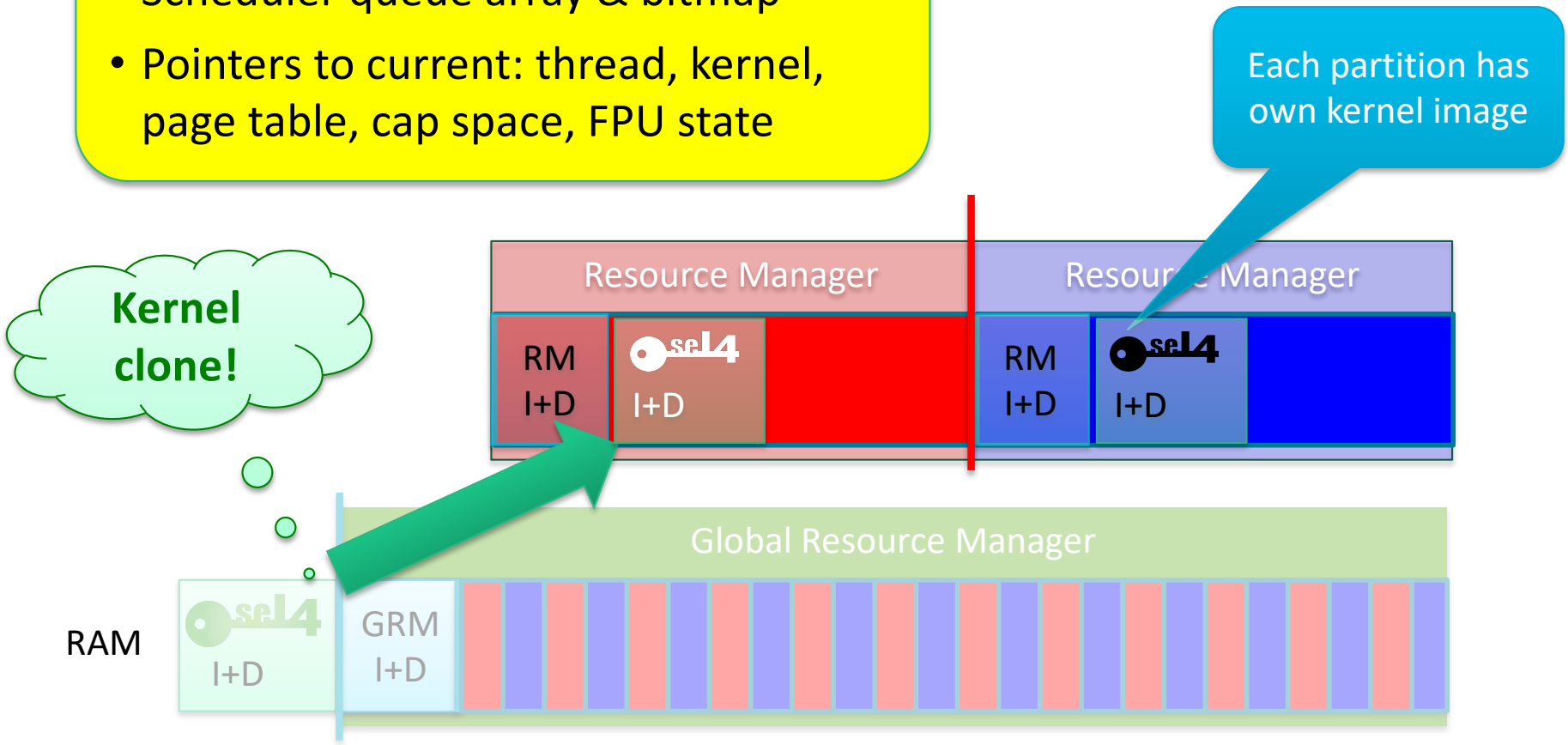
Cache Covert Channel Through Kernel
Spy observations



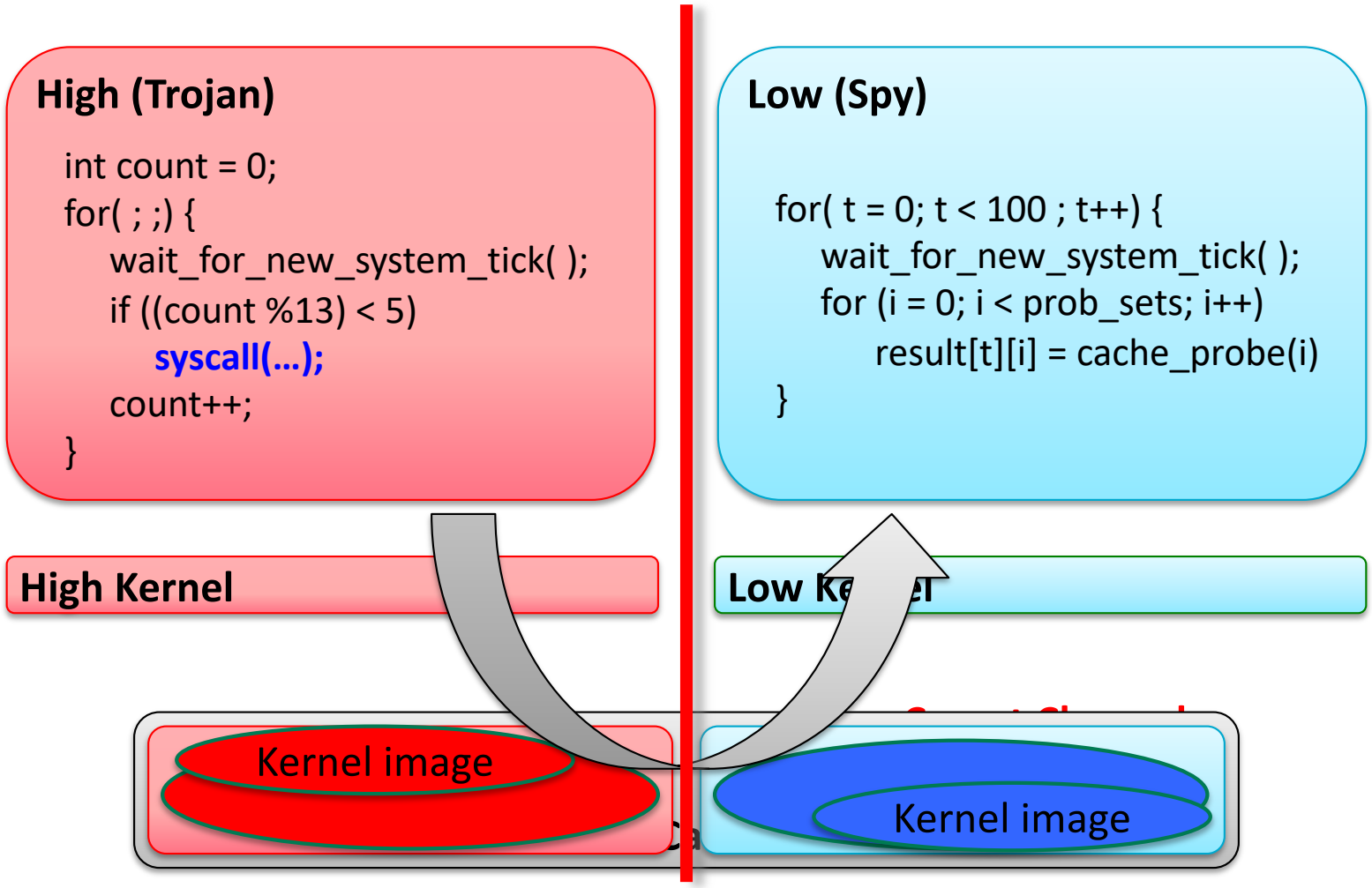
Colouring the Kernel

Only shared kernel data:

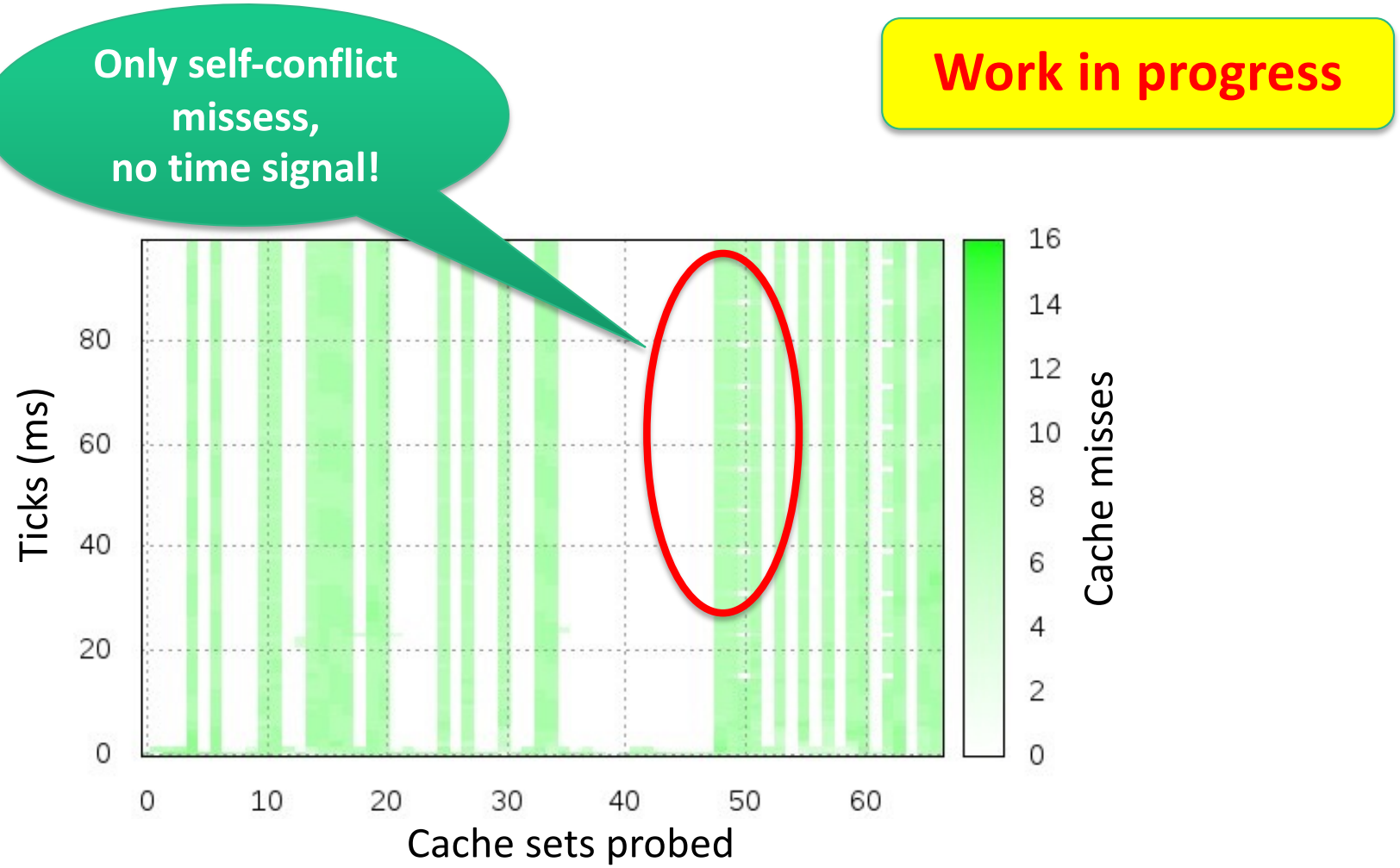
- Scheduler queue array & bitmap
- Pointers to current: thread, kernel, page table, cap space, FPU state

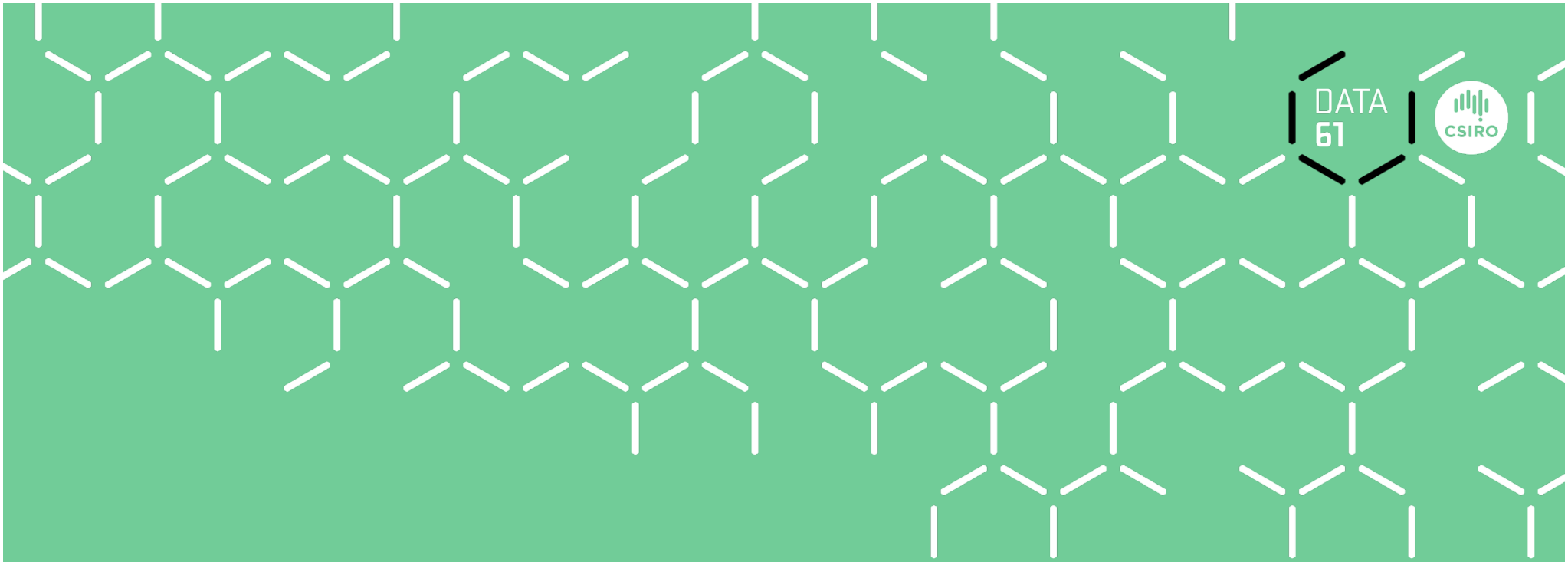


Timing Channel Through Kernel



Cache Covert Channel Through Kernel
Spy observations with coloured kernel





Status and Roadmap

seL4 Status

Unique to seL4!

Variant/Feature	ARM		x86	
	Implementation	Verification	Implementation	Verification
Bug-free source	✓	✓	✓	Q1/17
Bug-free binary	✓	✓	✓	TBD
Bug-free initialis.	✓	In progress	✓	TBD
Integrity	✓	✓	✓	TBD
Confidentiality	✓	✓	✓	TBD
Timeliness	✓	✓	✓	TBD
64-bit mode	Q2/17	TBD	✓	Q1/17
Multicore	Q4/16	TBD	Q4/16	In progress
Mixed-criticality RT	✓	Q2/18	✓	Q2/18
HW virtualisation	✓	Q1/17	Imminent	TBD
Multicore VMM	Q2/17	TBD	Q2/17	TBD

Thank you

Security is no excuse for poor performance!

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

August 2016

http://sel4.systems