NICTA

The L4 Microkernel
Research to Mass Deployment and Back

Gernot Heiser

John Lions Professor of Operating Systems, University of New South Wales

Leader, Trustworthy Embedded Systems, NICTA

CTO and Founder, Open Kernel Labs

Australian Government
Department of Communications, Information Technology and the Arts
Australian Research Council

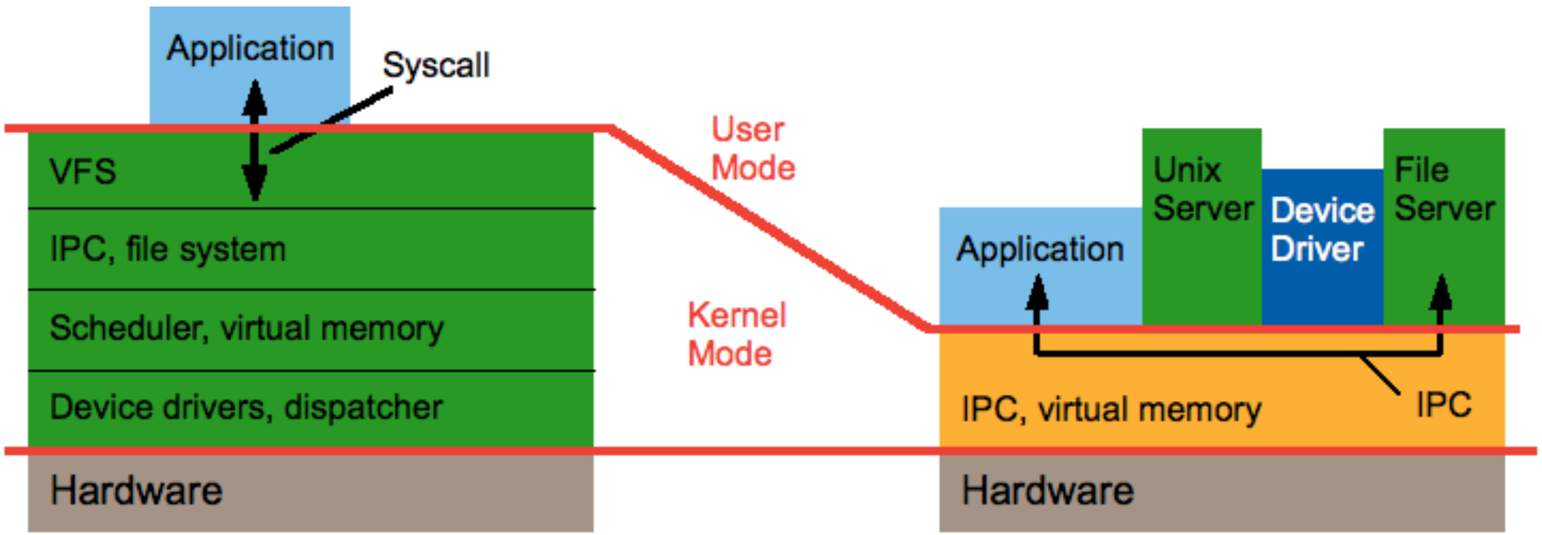
NICTA Members
ANU THE AUSTRALIAN NATIONAL UNIVERSITY
UNSW THE UNIVERSITY OF NEW SOUTH WALES
Department of State and Regional Development

Victoria The Place To Be
THE UNIVERSITY OF MELBOURNE
The University of Sydney
Queensland Government
Griffith UNIVERSITY
QUT Queensland University of Technology
The University Of Queensland AUSTRALIA

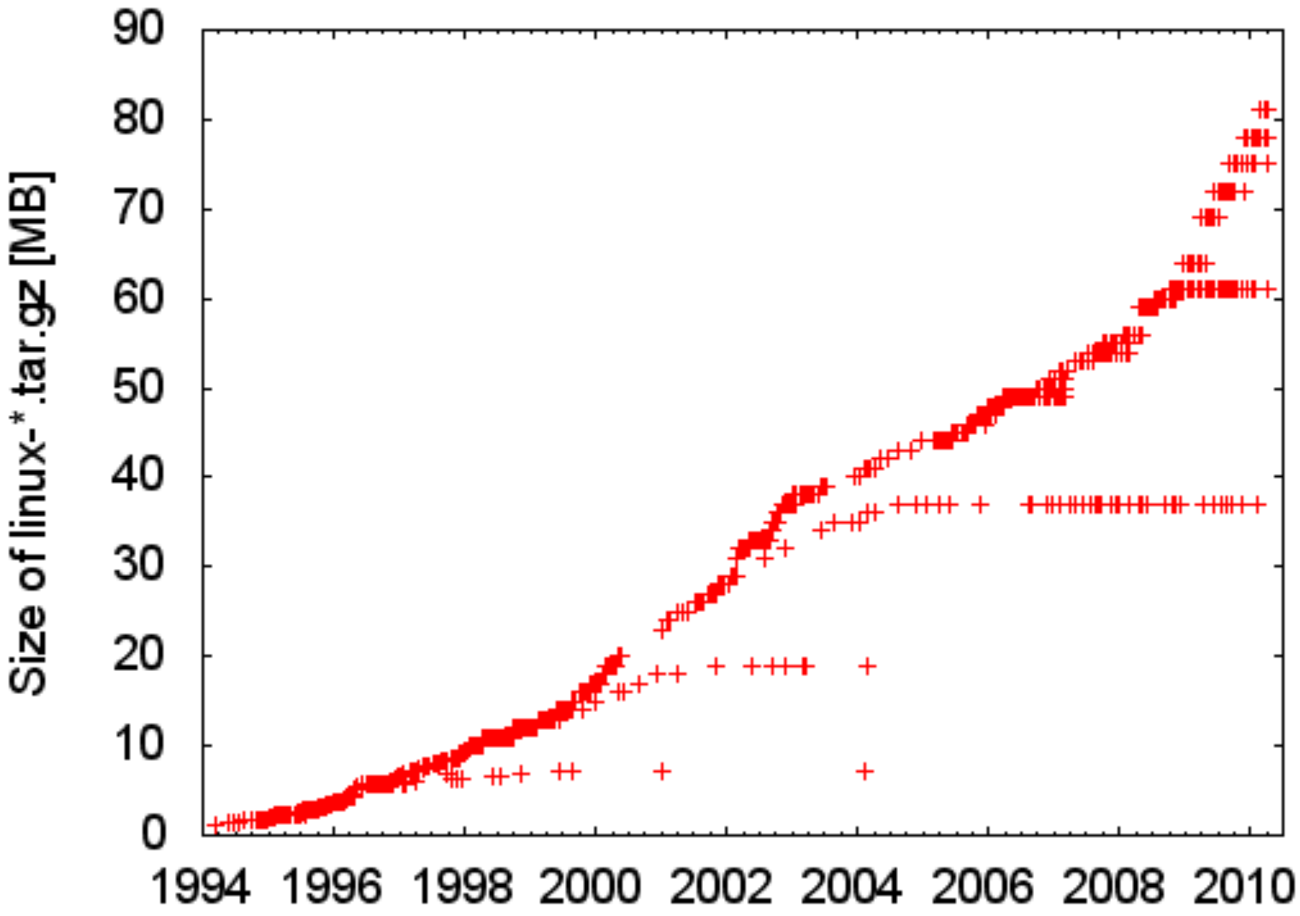
NICTA Partners

Microkernels — A Bit of History

- Originally proposed by Brinch Hansen [CACM ’70]
- Popularized in 1980’s (Mach, Chorus, etc)
- Idea: simplify kernel, increase robustness, flexibility…



Compare Linux



Microkernel Promises

- Combat kernel complexity, increase robustness, maintainability
  - dramatic reduction in amount of privileged code
  - modularity with hardware-enforced interfaces
  - normal resource management applicable to OS services
- Flexibility, adaptability, extensibility
  - policies defined at user level, subject to change
  - additional services provided by adding servers
- Hardware abstraction
  - hardware-dependent part of system is small, easy to optimise
- Security, safety
  - internal protection boundaries

Reality Check!
slow, inflexible
100µs IPC

Enter L4

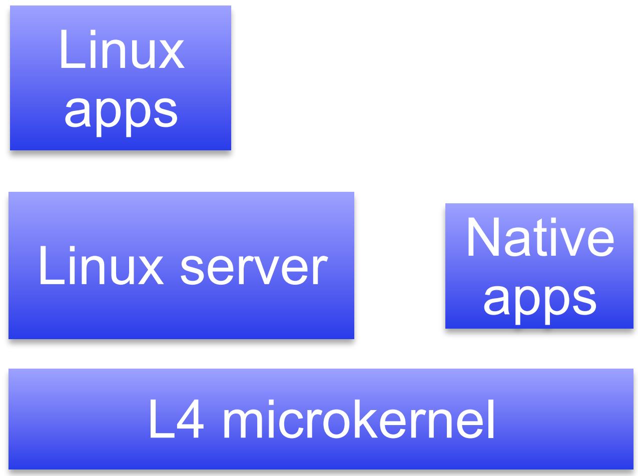
- Dramatically improved performance [Liedtke, SOSP '93, '95]
- Size:
  - L4 15kLOC assembler
    Mach: 90kLOC C
  - L4 small cache footprint ⇒ CPU limited
    Mach large cache footprint ⇒ memory limited
- API: minimal mechanisms
  - Threads, address spaces, IPC: minimal wrappers around hardware
- Lots of implementation tricks

Micro-kernel	CPU@MHz	IPC Cost [cycles]
Mach	i486@50	5750
Amoeba	68020@15	6000
Spin	21064@133	6783
L4	i486@50	250

Virtualization

L^{4}Linux [Härtig et al., SOSP'97]

- 5–10% overhead on macro-BMs
- 6–7% overhead on kernel compile

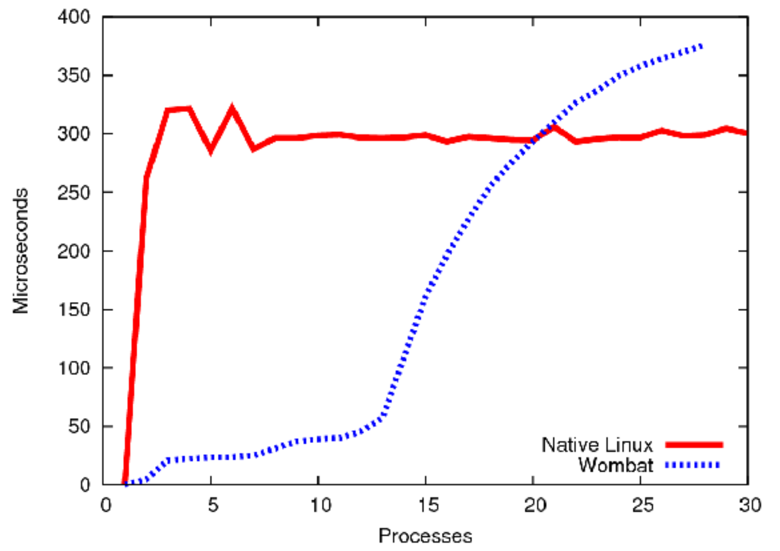


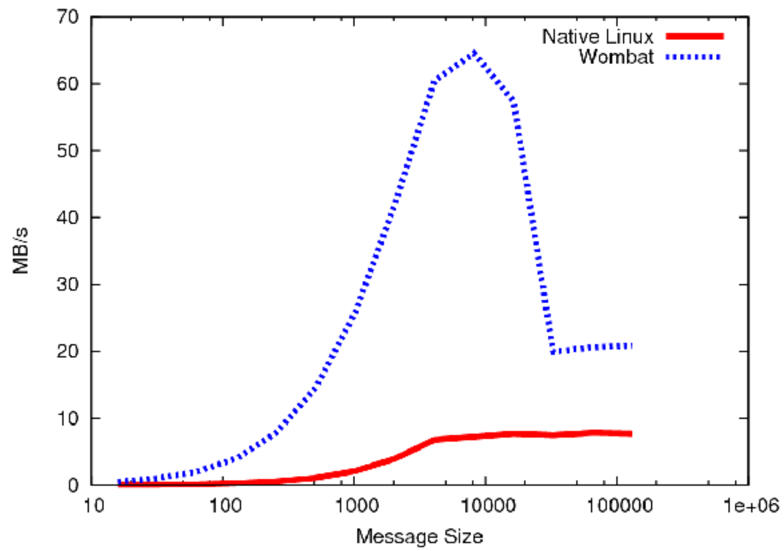
MkLinux (Linux on Mach):

- 27% overhead on kernel compile
- 17% overhead with Linux in kernel

NICTA Research: Focus on Embedded

- L4 implementations on embedded processors
  - ARM, MIPS
- Wombat: portable virtualized Linux for embedded systems
  - Outperforms native Linux on ARMv4/v5 thanks to fast context-switching tricks
- Basis for real-world deployments





Large-Scale Commercial Deployment

NICTA

Toshiba W47T
2006

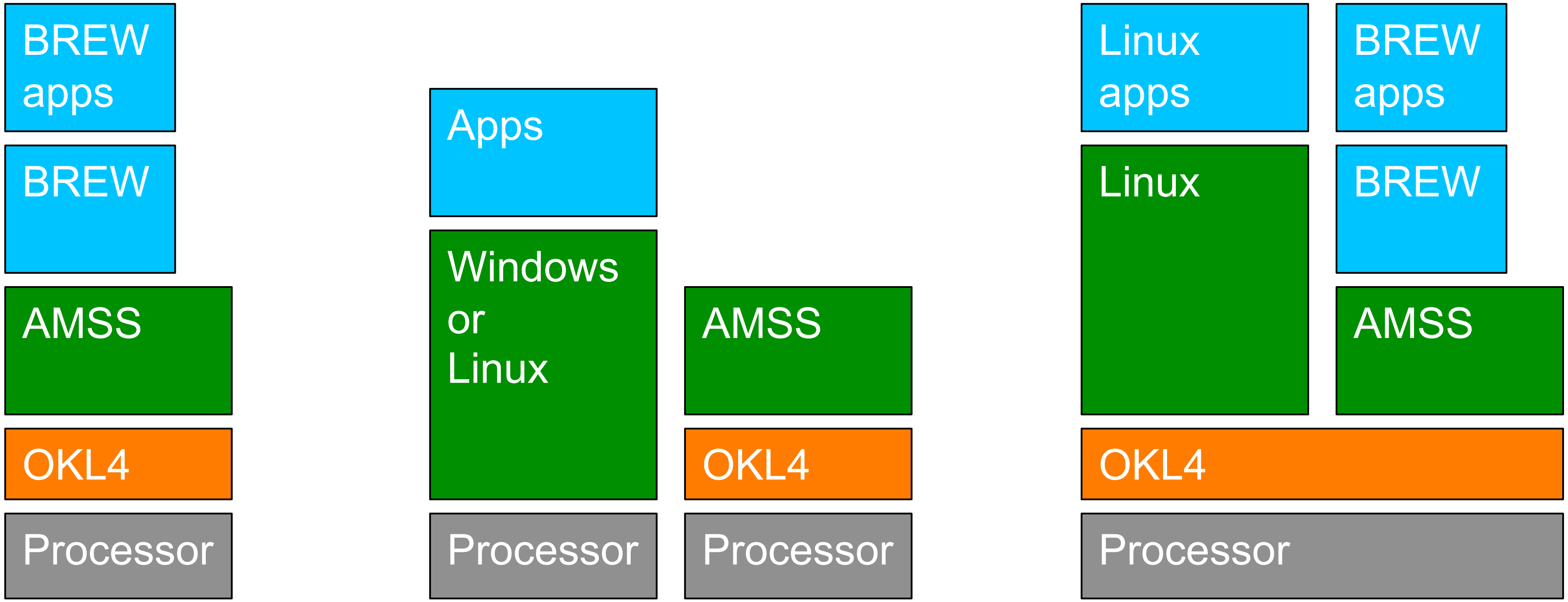
HTC TyTN II
2007

HTC Dream (G1)
2008

Motorola Evoke
2009

More than 700 million OKL4-based devices shipped to date!

System Architecture



What Have We Learned?

NICTA

Liedtke's microkernel design principles [CACM '96]

- Minimality
- Well-written
- Appropriate abstractions
- Unportable
- Synchronous (blocking) IPC
- Rich IPC message structure
- Fast thread access
  - Thread IDs as unique identifiers
  - Virtual TCB array
  - Per-thread kernel stack (process-oriented kernel)

What Have We Learned?

- Process-orientation wastes RAM
  - Replaced by single-stack (event-driven) approach
- Virtual TCB array wastes VAS, TLB entries
  - …without performance benefits on modern hardware
- Capabilities are better than thread UIDs
  - Provide uniform resource control model & avoid covert channels
- Also: IPC timeouts are useless
  - Replaced by block/poll bit
- Virtualization is essential
  - Re-think kernel abstractions

A Fork in the Road

NICTA

Research (NICTA)

- seL4 kernel
- Aim: extreme trustworthiness
- Formal verification
- API experiments

Commercialisation (OK Labs)

- OKL4 Microvisor
- Aim: virtualization platform for mobile systems
- Ease of deployment
- Match to commercial realities

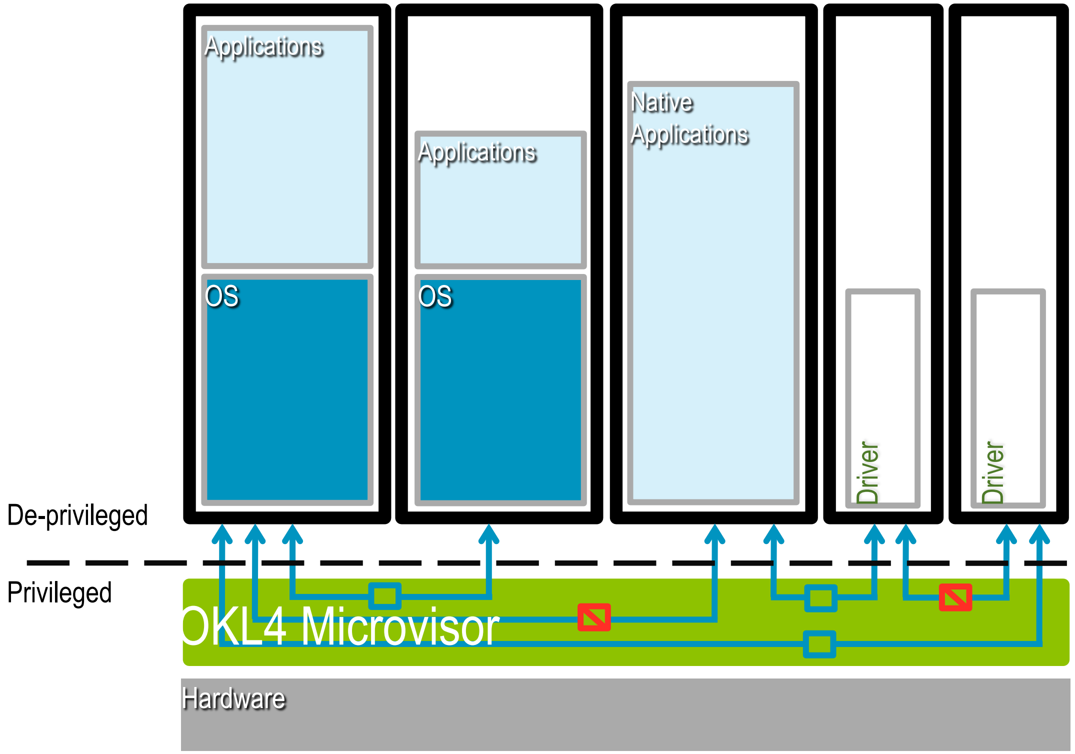
Concurrent development — how do results compare?

The OKL4 Microvisor

API optimised for low-overhead virtualization

- Eliminated:
  - recursive address spaces
  - Synchronous IPC
  - Kernel-scheduled threads
- API closely models hardware:
  - vCPU, vMMU, vIRQ + “channels” (FIFOs)
- Capabilities for resource control

OKL4 Capabilities



Control over communication channels

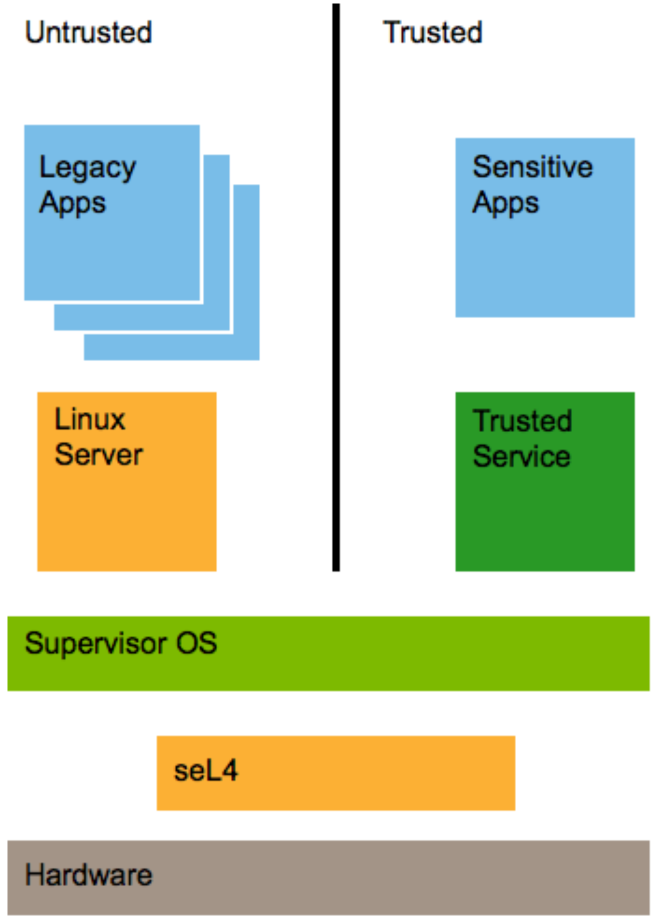
OKL4 Virtualization Performance

Benchmark	Native [μs]	Virtualized [μs]	Overhead
Null syscall	0.6	0.96	60 %
Read	1.14	1.31	15 %
Stat	4.73	5.05	7 %
Open/close	9.12	8.23	-10 %
Select(10)	2.62	2.98	14 %
Signal install	1.77	2.05	16 %
Signal handler	6.81	5.83	-14 %
Fork	1106	1190	8 %
Fork+execve	4710	4933	5 %
System	7583	7796	3 %

- On Beagle board (ARM Cortex A8 @ 500 MHz)
- Macro-benchmark overhead: < 1%

The seL4 Microkernel: Goals

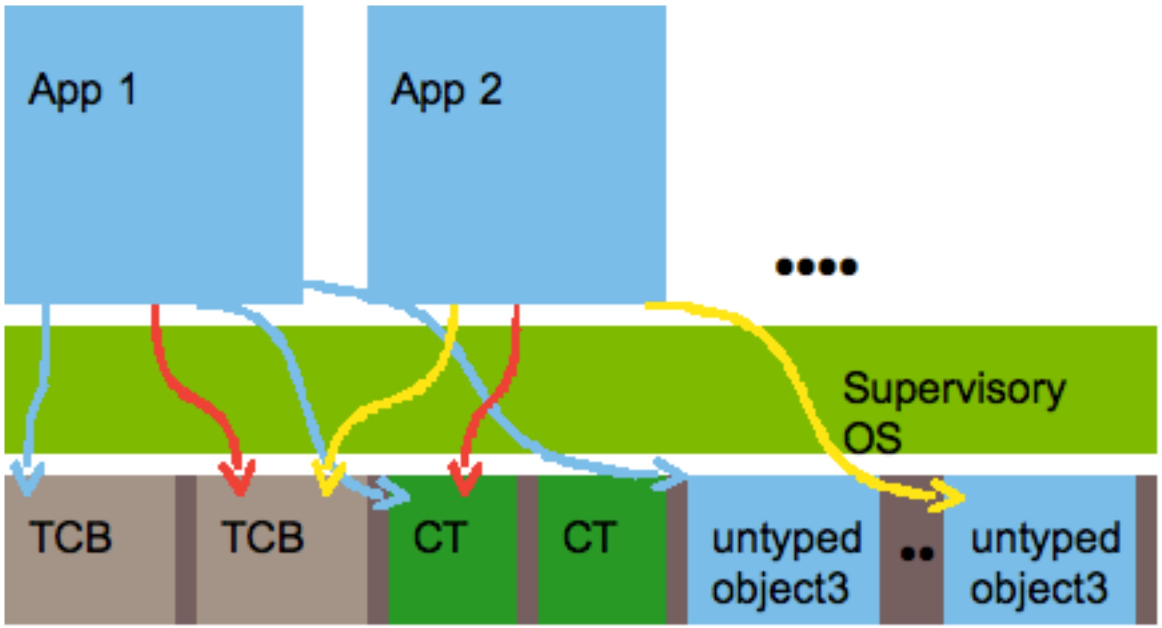
- General-purpose
- Formal verification
  - Functional correctness
  - Security/safety properties
- High performance
  - < 15 cy slower IPC than other L4



seL4 Novelty: Kernel Resource Management

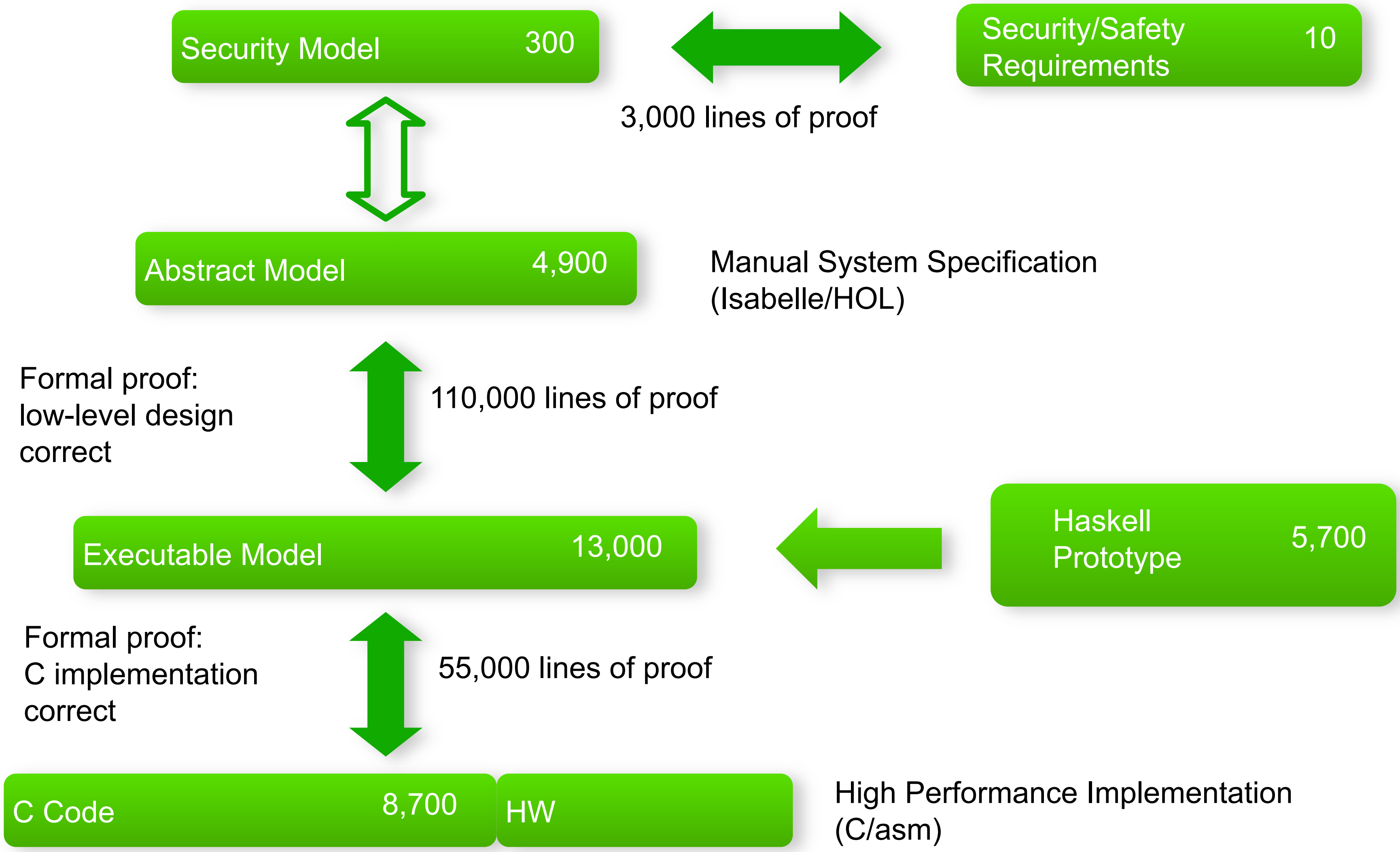
NICTA

- No kernel heap: all memory left after boot is handed to userland
  - Resource manager can delegate to subsystems
- Operations requiring memory explicitly provide memory to kernel
- Result: strong isolation of subsystems
  - Operate within delegated resources
  - No interference

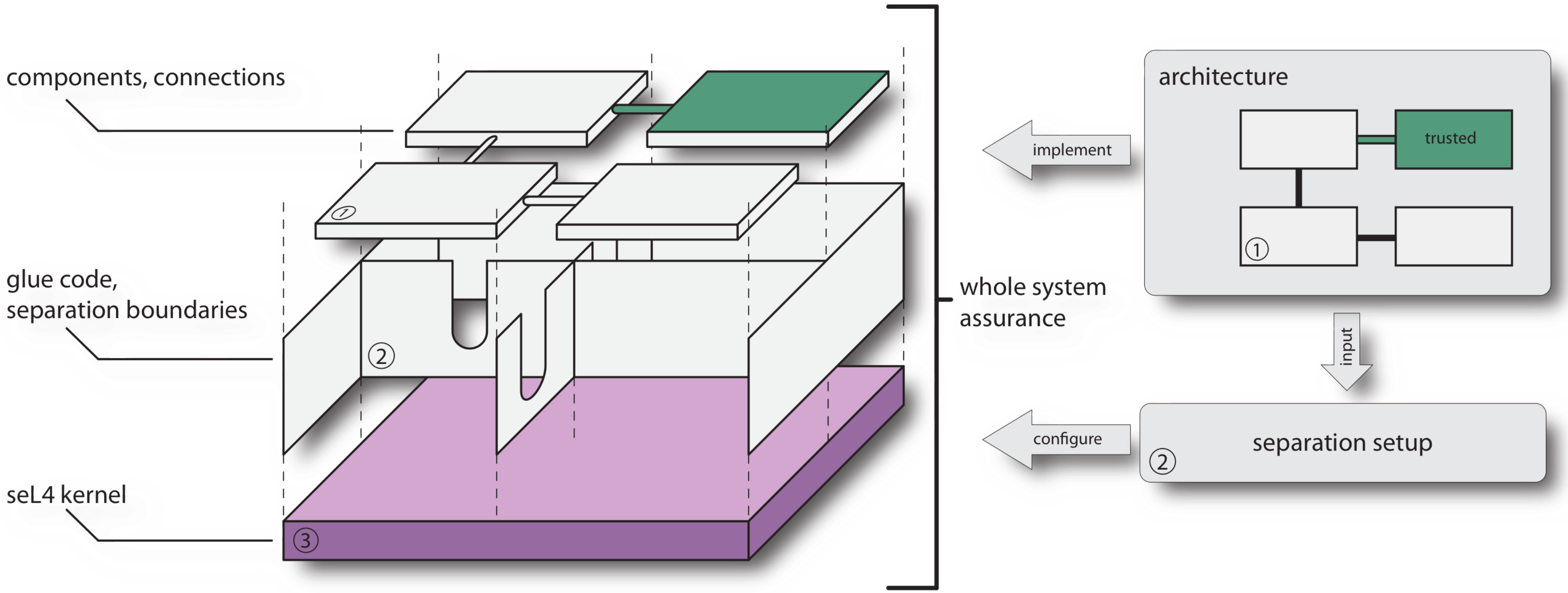


Formal Verification of seL4 Microkernel

NICTA



In Progress: Whole-System Guarantees



Liedtke's Design Rules 15 Years Later

Liedtke	seL4	OKL4 Microvisor
Minimality	Yes	Yes
Well written	Yes	Yes
Appropriate abstractions	Yes, but abstractions are quite different	
• thread	• thread	• virtual CPU
• address space	• address space	• virtual MMU
• synchonous IPC	• sync IPC + async notify	• virtual IRQ (async)
Unportable (asm)	No, almost no asm	No, almost no asm
Rich msg structure	No	No
Unique thread IDs	No, has capabilities	No, has capabilities
Virtual TCB array	No	No
Per-thread kernel stack	No, event kernel	No, event kernel

Conclusions

NICTA

- L4 microkernels are now “mainstream”
  - One of the most widely-deployed protected OS kernels ever
  - Most technically-advanced microkernels
- Commercial experience has had significant impact
  - Simplified API (timeouts, message structure)
  - Need for asynchronous communication primitives
  - Capabilities are suitable for the “real world”
- Best API is still an open question
- Microkernels are finally delivering on old promises
  - Small TCBs for safety, security, reliability
  - Performance is no longer an issue (for L4 kernels at least)