RIVERSIDE RESEARCH

Single Kernel Multicore seL4 (SKMS)

Leveraging single-core proofs for multicore seL4 via an asymmetric architecture

Jacob Saunders, **David Swasey**, **Scott Brookes**, Gordon Stewart

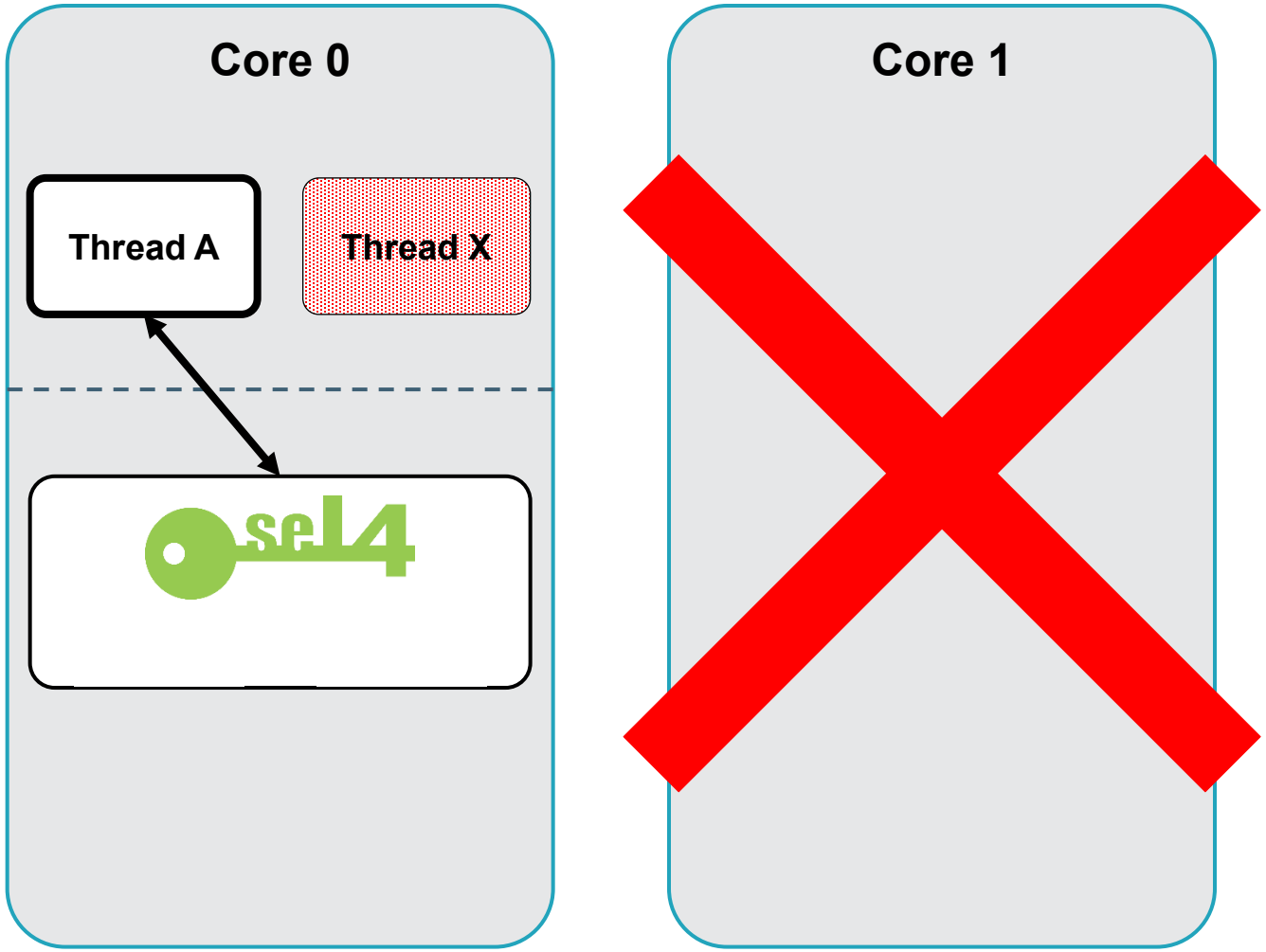
3 September 2026

Bottom line, up front

- We are systems and FM people, new to seL4
- Tried to summarize prior community efforts in formally verified multicore seL4
  - SMP vs Static Mutli-kernel
- We bring prior experience with an experimental asymmetrical kernel architecture
  - Contribution: We prototyped this architecture for seL4 (SKMS)
- We bring prior experience with SMP hypervisor verification
  - Contribution: We inspected the seL4 proof architecture and drafted a verification plan
- Goal: Community feedback on architecture & verification potential/interest

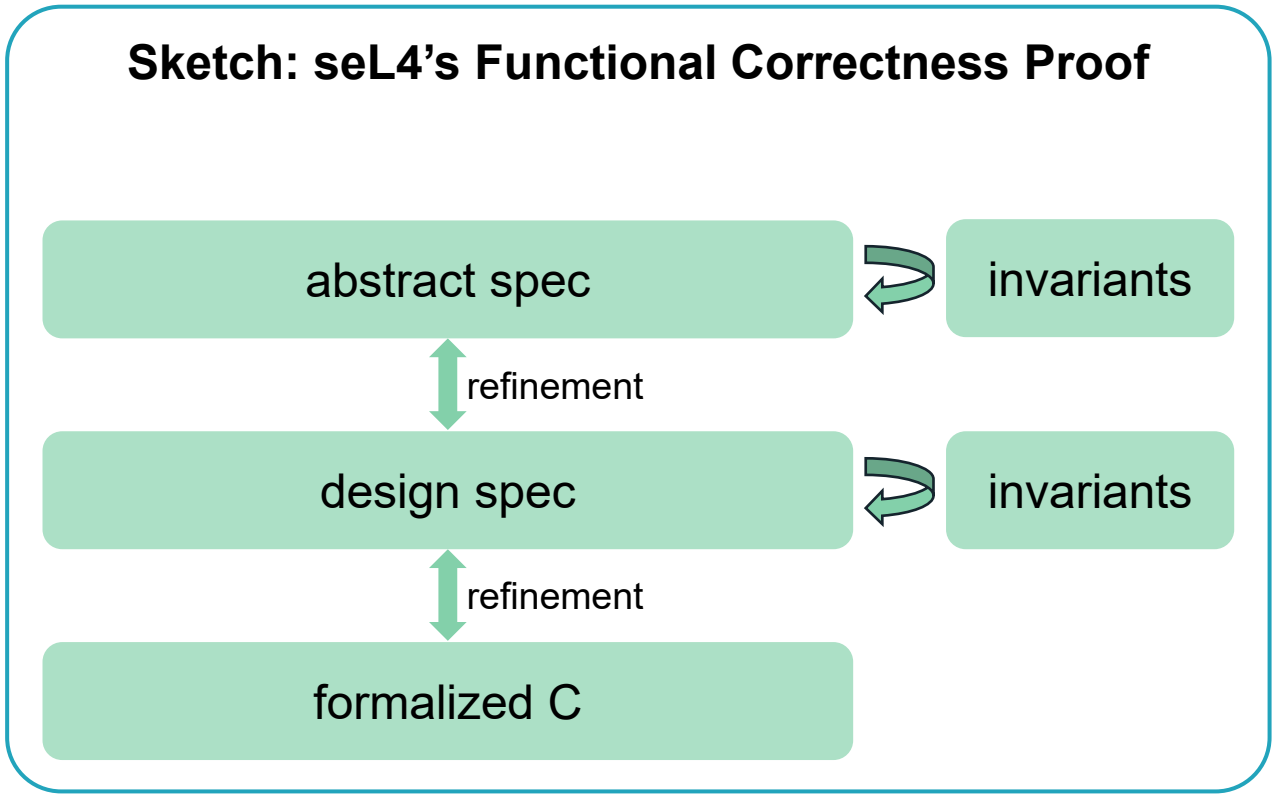
RIVERSIDE RESEARCH

Verified seL4 cannot (yet) use multiple cores



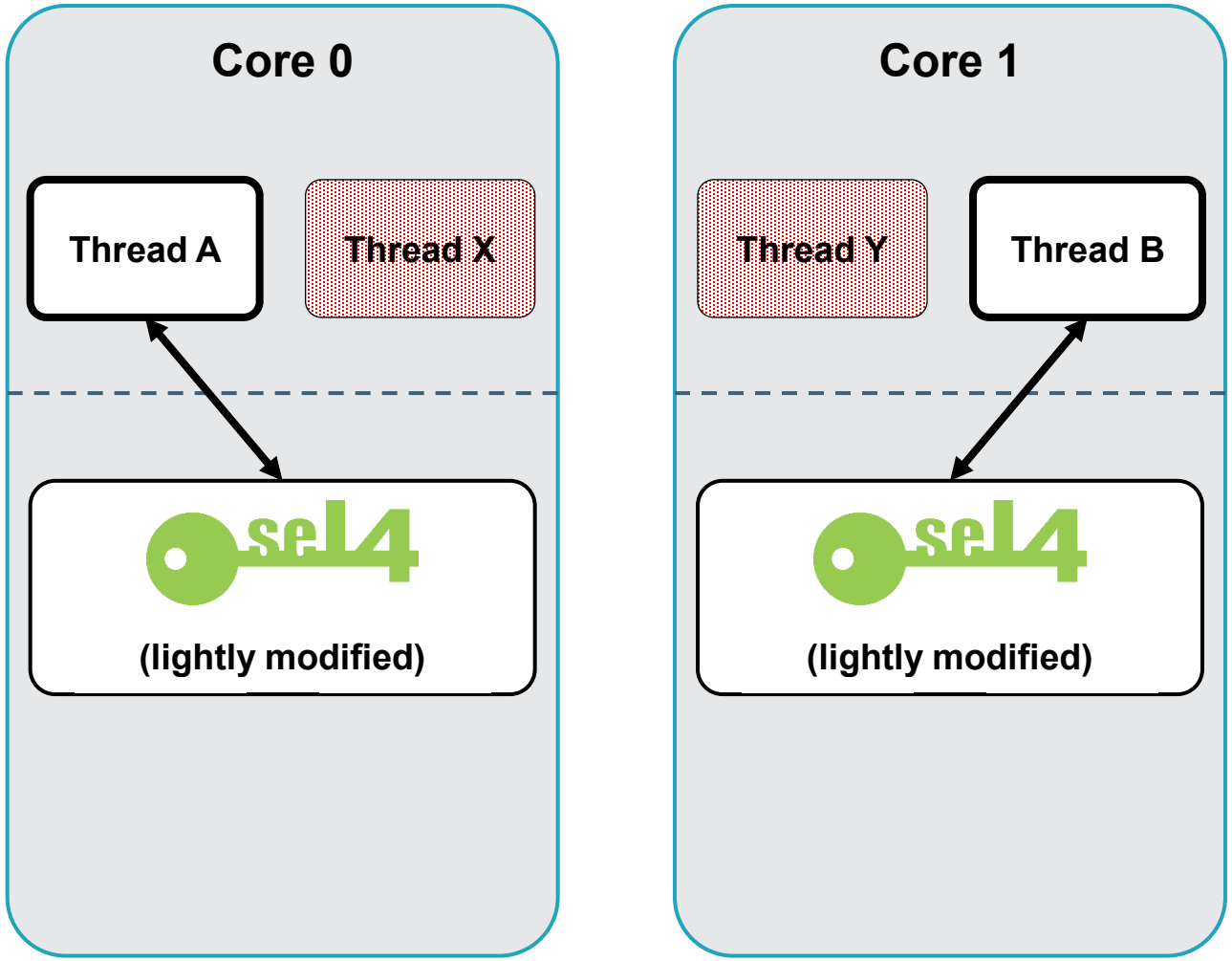
- seL4's verification offers…
  - Functional correctness of the kernel's high-level C code
  - Binary correctness, where supported
  - Integrity and availability of resources, mediated by capabilities
  - And more
- **… for a single core**

Why isn't there a multicore proof?



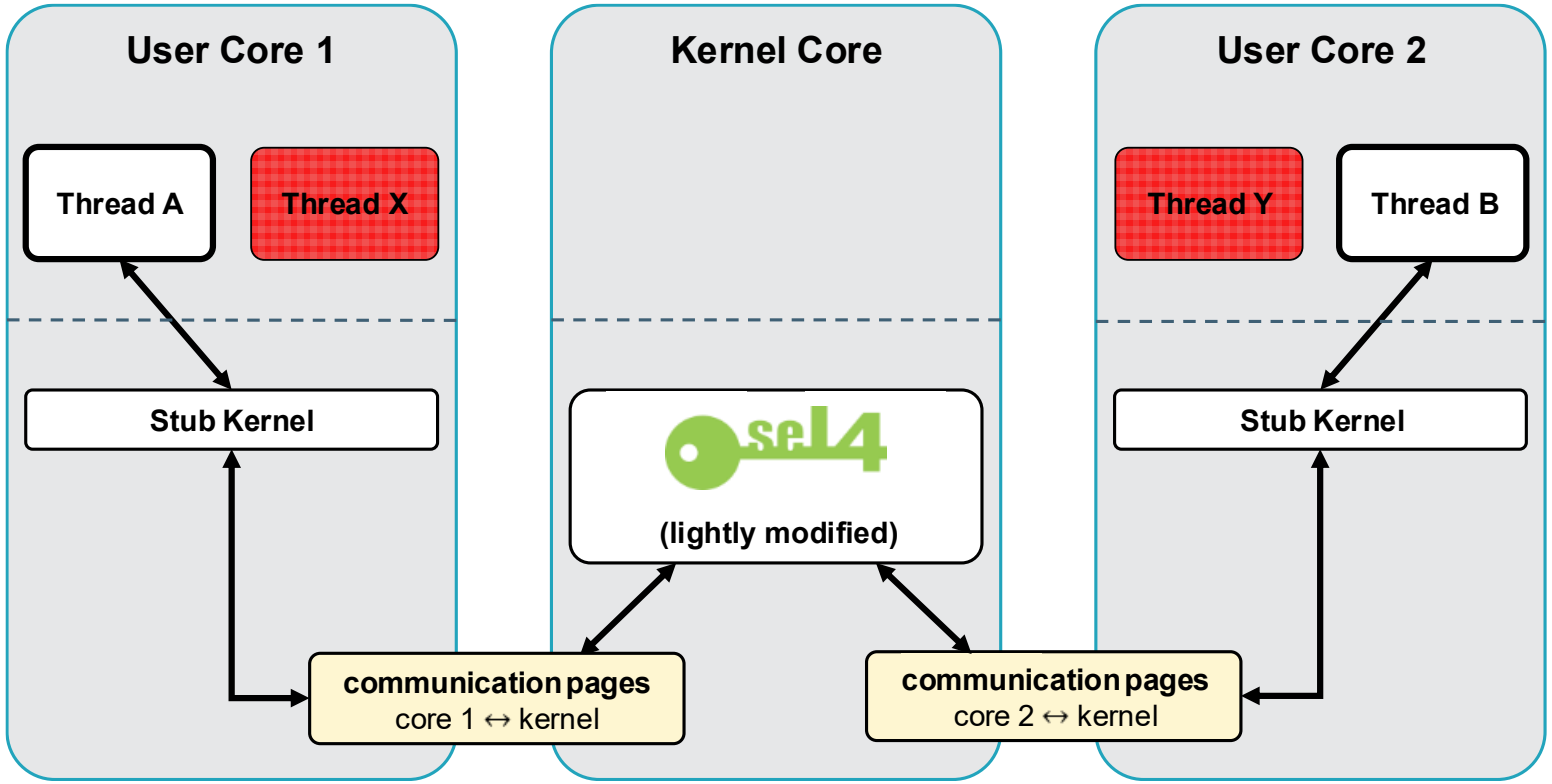
- The scale and complexity of seL4 proofs raise maintenance costs
- The seL4 functional correctness proofs are sequential, not parallel
  - Adding (more) concurrency here would introduce significant complications
  - Such complications could cascade through invariants, specs, and proofs
- **SMP for seL4 is a monumental task!**

Roadmap: Verified Static Multikernel seL4 (as we understand it)



- One seL4 instance per core
- Kernel state and memory statically partitioned
- No direct cross-core kernel interference
  - User threads may use shared memory to communicate across cores
- Promises to **reuse** much of the sequential proof
  - Builds assurance incrementally

Single Kernel Multicore seL4 (SKMS)



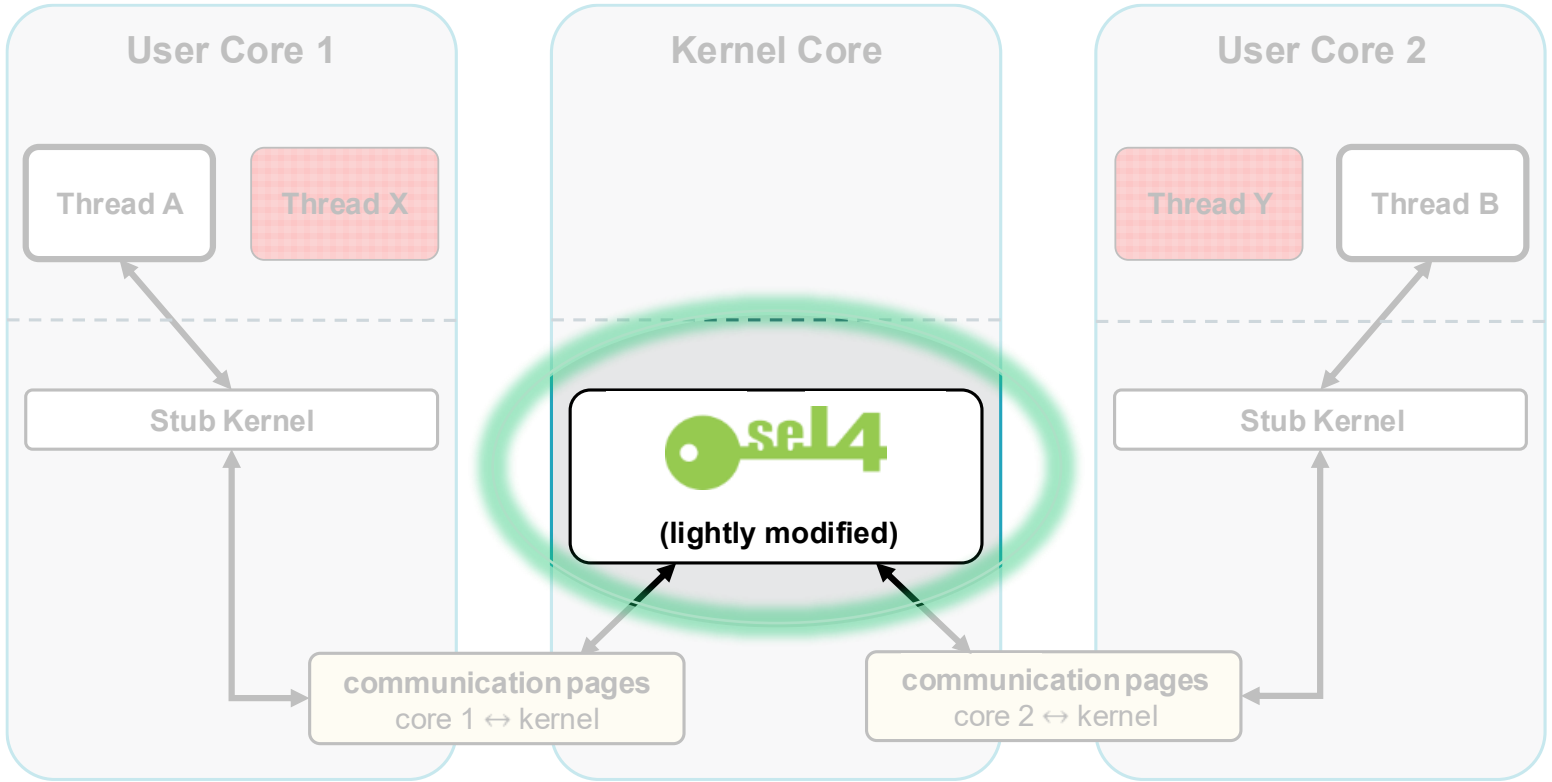
- The kernel core runs *the* seL4 instance
- User cores run user mode threads and a stub kernel
- seL4 interacts indirectly with user mode threads
  - Shared memory and IPIs enable stub kernel ↔ seL4 communication
  - Stub kernels manage user core hardware on behalf of seL4
- User cores map no kernel memory
  - Stub kernel could perhaps be untrusted (or omitted)

SKMS prototype immediately available (X86)

Rethinking operating system design: asymmetric multiprocessing for security and performance
Scott Brookes and Stephen Taylor, NSPW '16, doi:10.1145/3011883.3011886

RIVERSIDE RESEARCH

SKMS Verification Plan (sequential seL4 proof + concurrent comms)



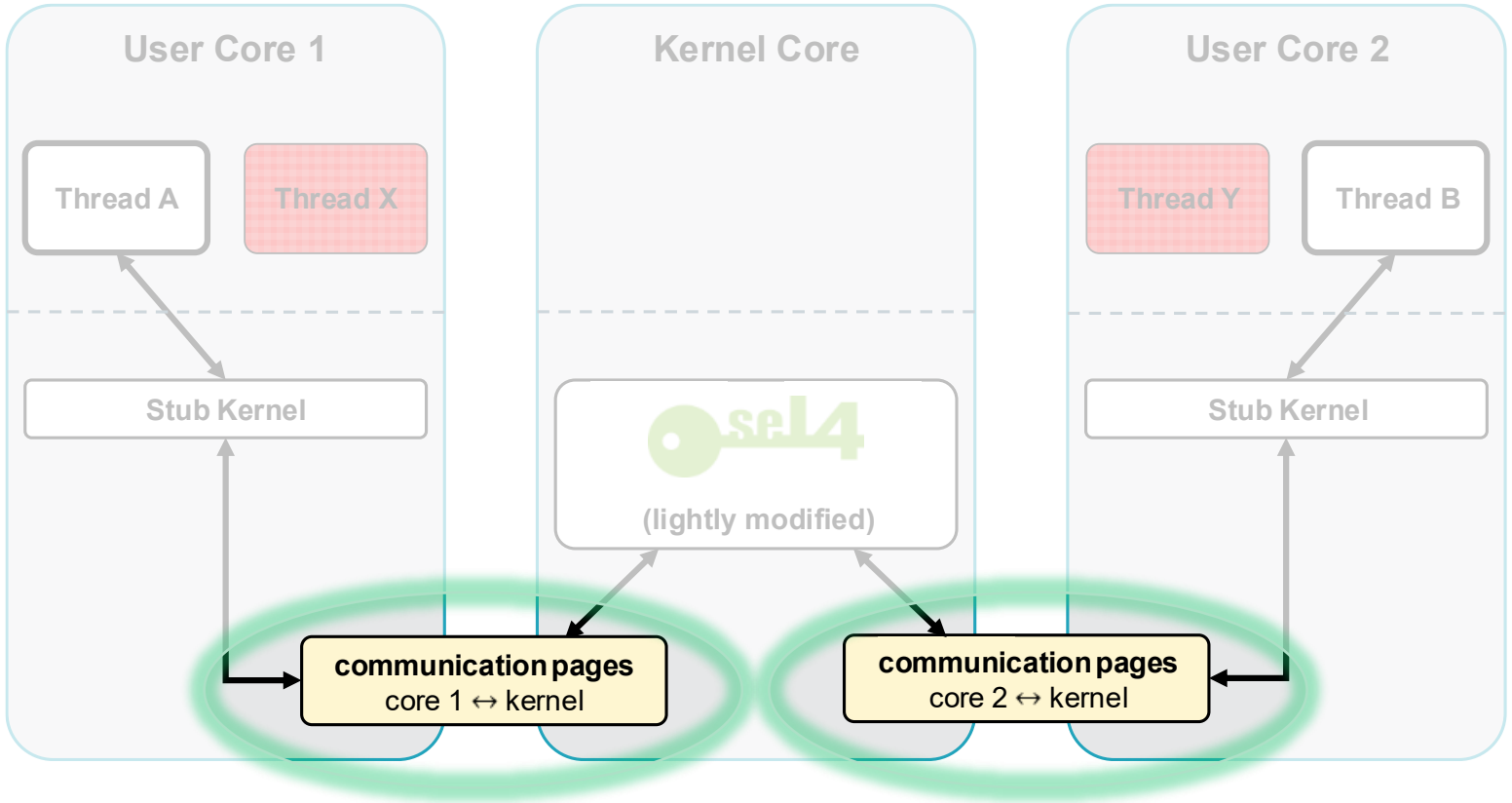
Sequential proof

› Verify a modified seL4 scheduler
  › Every user mode thread now bound to a user core
  › A large, systematic change to seL4's existing sequential proof

Concurrent proofs

RIVERSIDE RESEARCH

SKMS Verification Plan (sequential seL4 proof + concurrent comms)



Sequential proof

› Verify a modified seL4 scheduler
  › Every user mode thread now bound to a user core
  › A large, systematic change to seL4's existing sequential proof

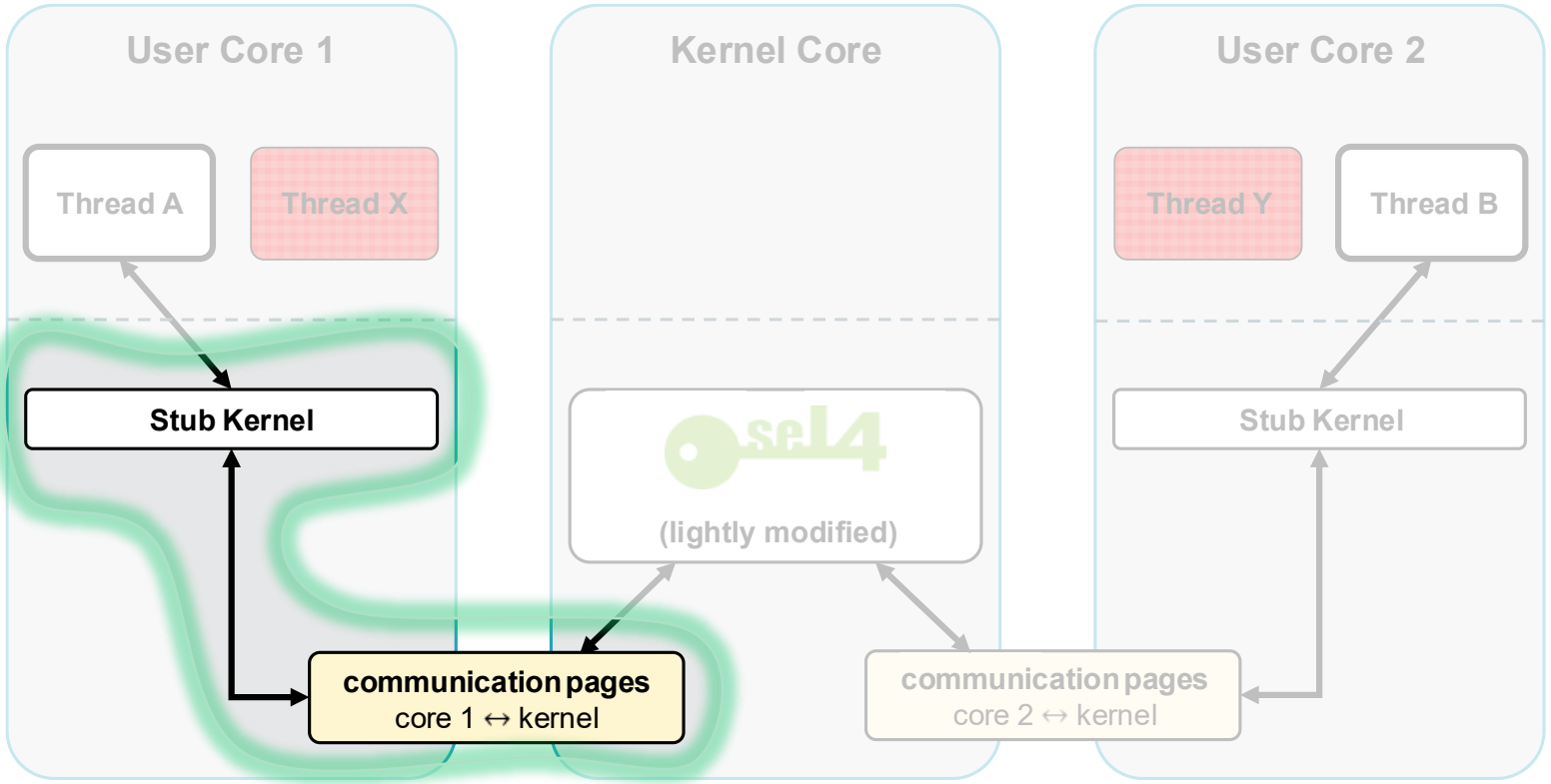
Concurrent proofs

› Verify the comms library

› Verify the stub kernel's use of the comms library

RIVERSIDE RESEARCH

SKMS Verification Plan (sequential seL4 proof + concurrent comms)



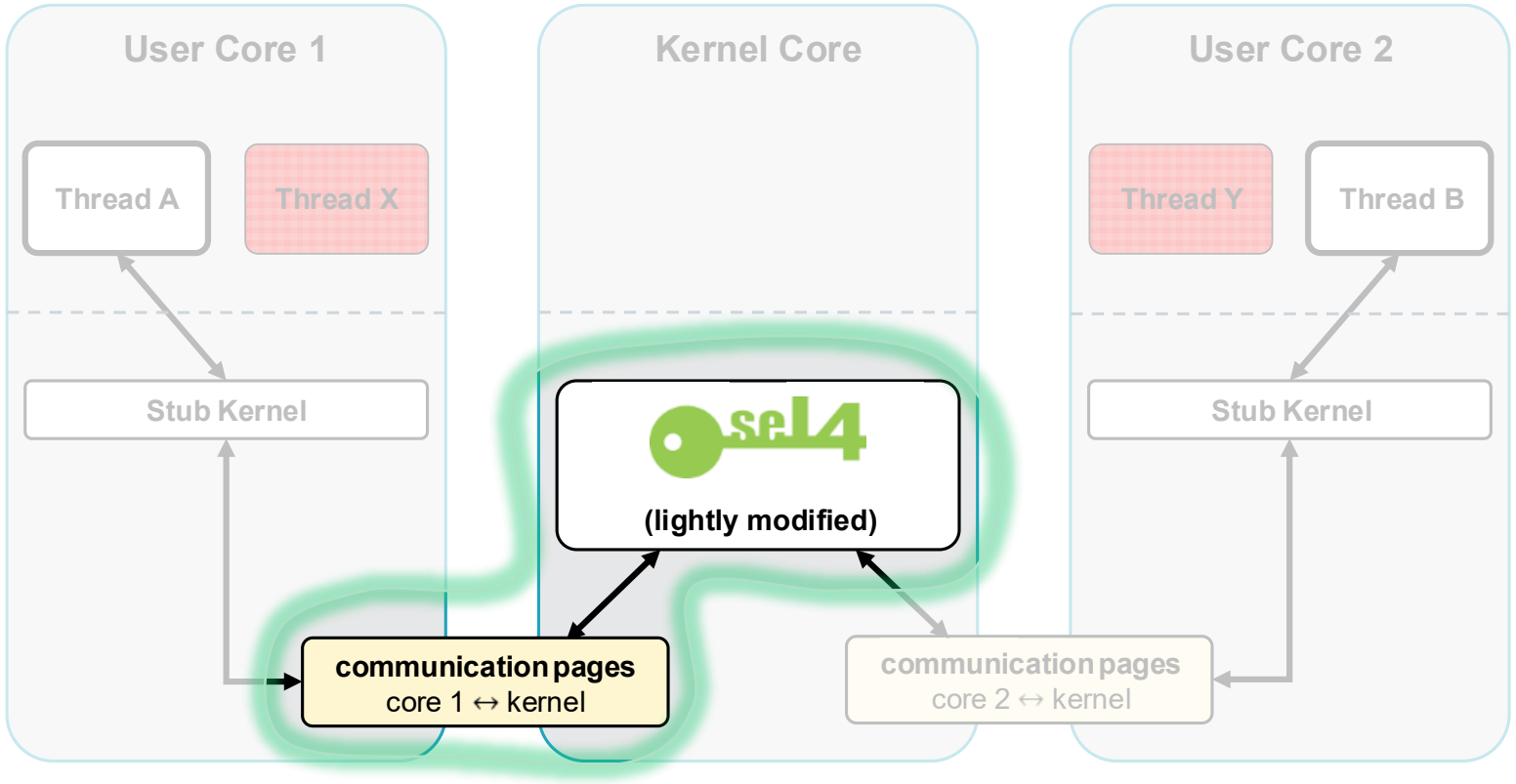
Sequential proof

- Verify a modified seL4 scheduler
  - Every user mode thread now bound to a user core
  - A large, systematic change to seL4's existing sequential proof

Concurrent proofs

- Verify the comms library
- Verify the stub kernel's use of the comms library
- Verify seL4's use of the comms library

SKMS Verification Plan (sequential seL4 proof + concurrent comms)



Sequential proof

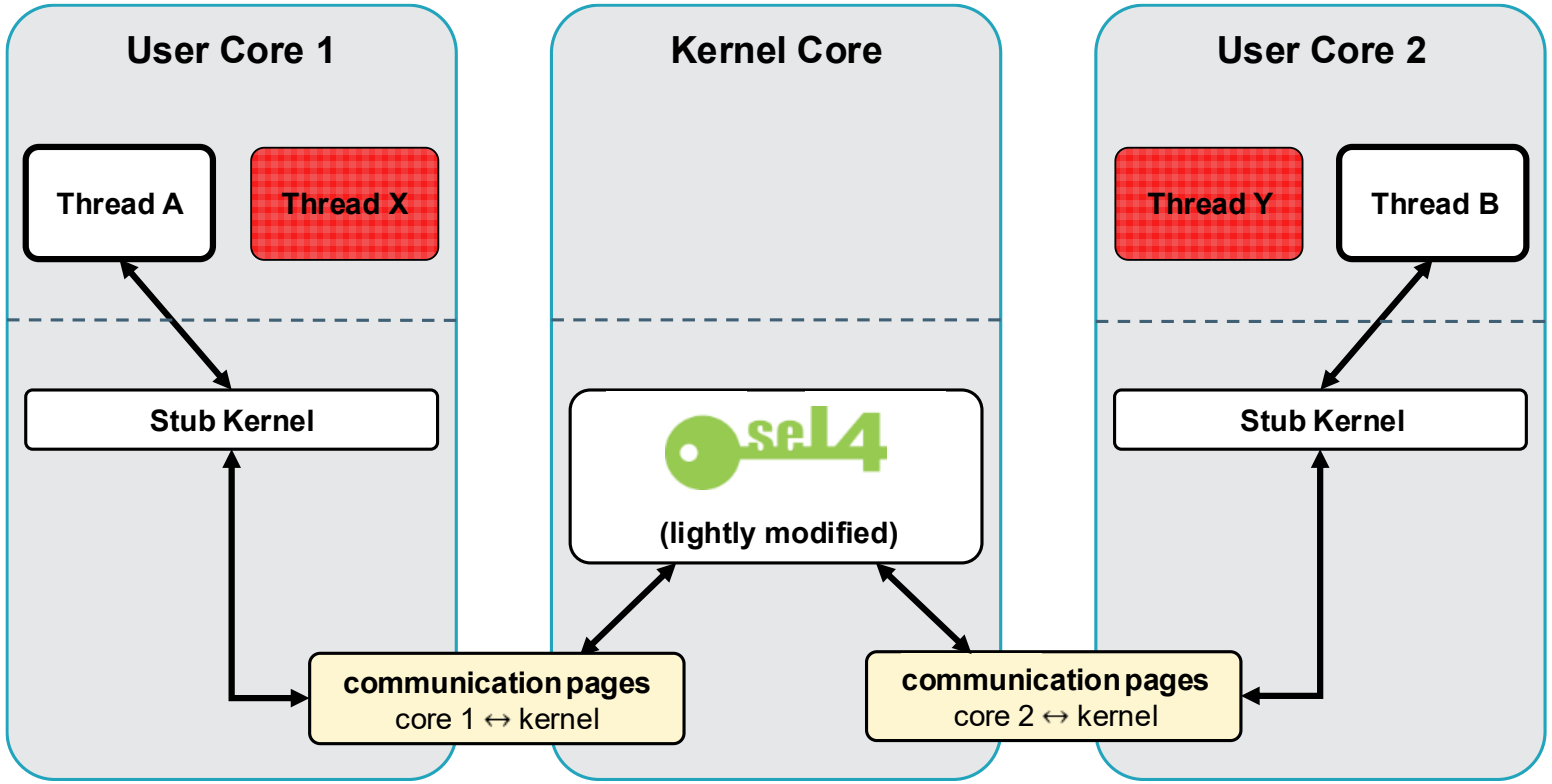
- Verify a modified seL4 scheduler
  - Every user mode thread now bound to a user core
  - A large, systematic change to seL4's existing sequential proof

Concurrent proofs

- Verify the comms library
- Verify the stub kernel's use of the comms library
- Verify seL4's use of the comms library

RIVERSIDE RESEARCH

Gaps in the SKMS Verification Plan



Inherited gaps

› Stub kernel entry/exit and inline assembler will remain unverified

› Critical assumptions about the bootstrap process

New gap

› Two separate proofs with no machine-checked evidence they compose in a meaningful way

  › The sequential proofs in Isabelle/HOL

  › The concurrent proofs in (say) Rocq and Iris

RIVERSIDE RESEARCH

RIVERSIDE RESEARCH

Questions? Comments? Feedback?

David Swasey pswasey@riversideresearch.org
Scott Brookes sbrookes@riversideresearch.org