Thread-Level Speculation with Kernel Support

Clemens Hammacher, Kevin Streit,
Andreas Zeller and Sebastian Hack

SAARLAND UNIVERSITY
GRADUATE SCHOOL OF COMPUTER SCIENCE

International Conference on Compiler Construction
March 17th, 2016, Barcelona

SAARLAND UNIVERSITY
GRADUATE SCHOOL OF COMPUTER SCIENCE

Thread-Level Speculation (TLS)

- *unsafe / optimistic* parallel execution
- data or control dependences might exist
- runtime system checks for violations
- rollback & re-execute in case of conflict

SAARLAND UNIVERSITY
GRADUATE SCHOOL OF COMPUTER SCIENCE

Thread-Level Speculation (TLS)

- *unsafe / c*
- data or c
- runtime s
- rollback &

```
for (auto &task : tasks[today]) {
    task.compute();
    if (!task.success()) {
      fatalError(“task failed: ” + task.id());
    }
}
```

```
for (auto id : taskIds[today]) {
    tasks[id].compute();
}
```

Thread-Level Speculation (TLS)

- *unsafe / c*
- data or c
- runtime s
- rollback &

```
for (auto &task : tasks[today]) {
    task.compute();
    if (!task.success()) {
      fatalError(“task failed: ” + task.id());
    }
}
```

```
for (auto id : taskIds[today]) {
    tasks[id].compute();
}
```

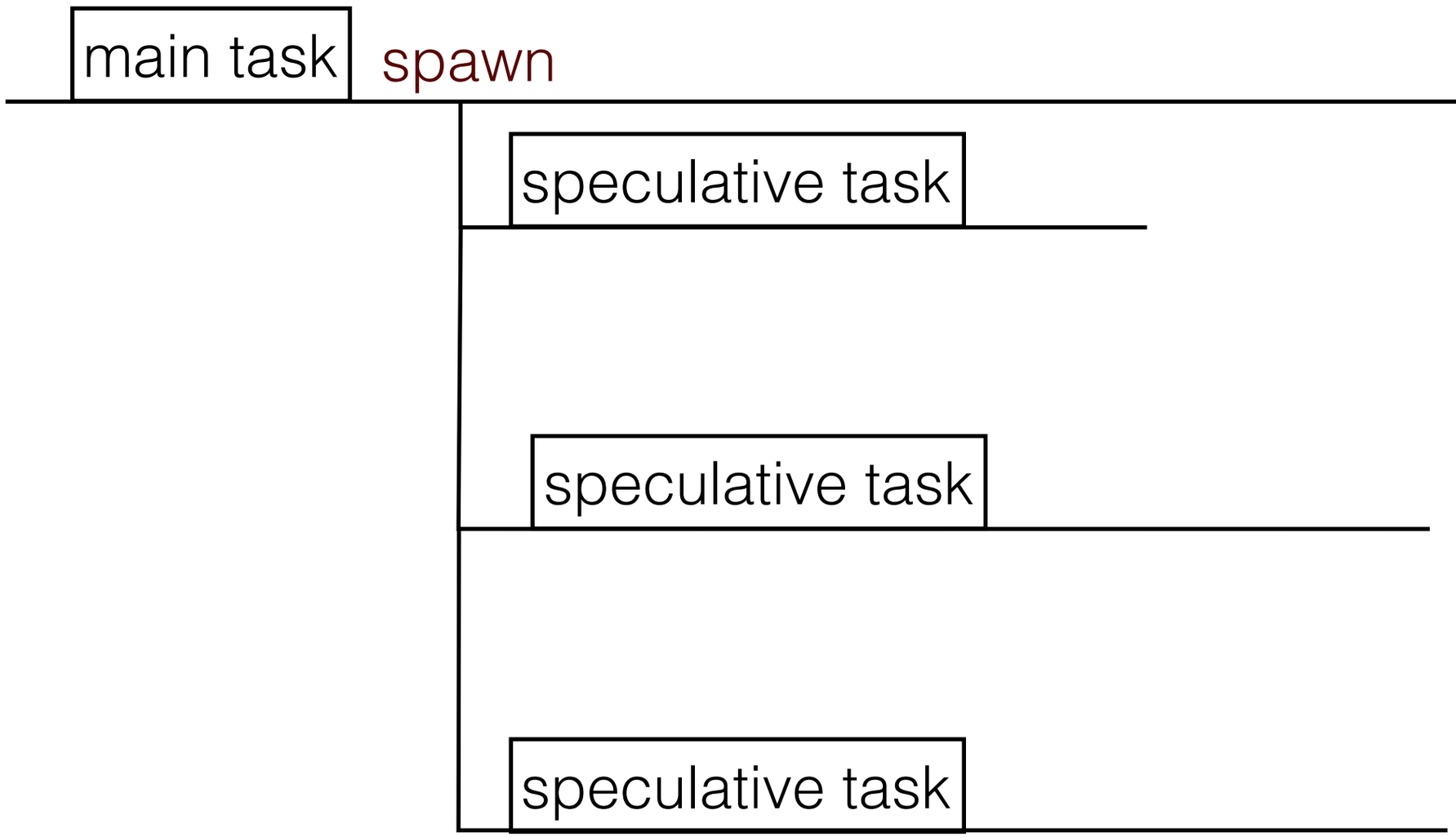
→ manual / guided parallelization

→ automatic parallelization

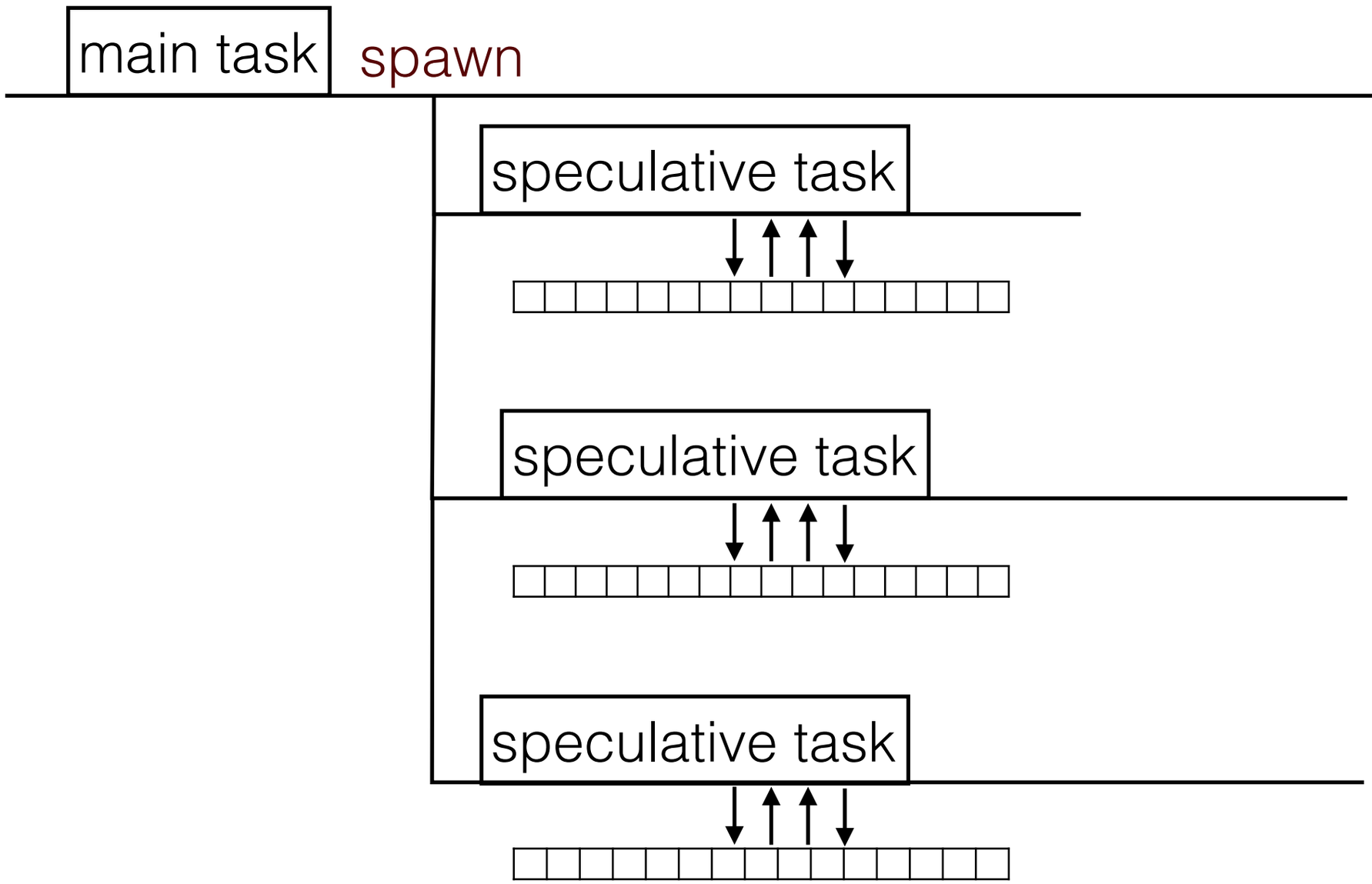
Execution in TLS

main task

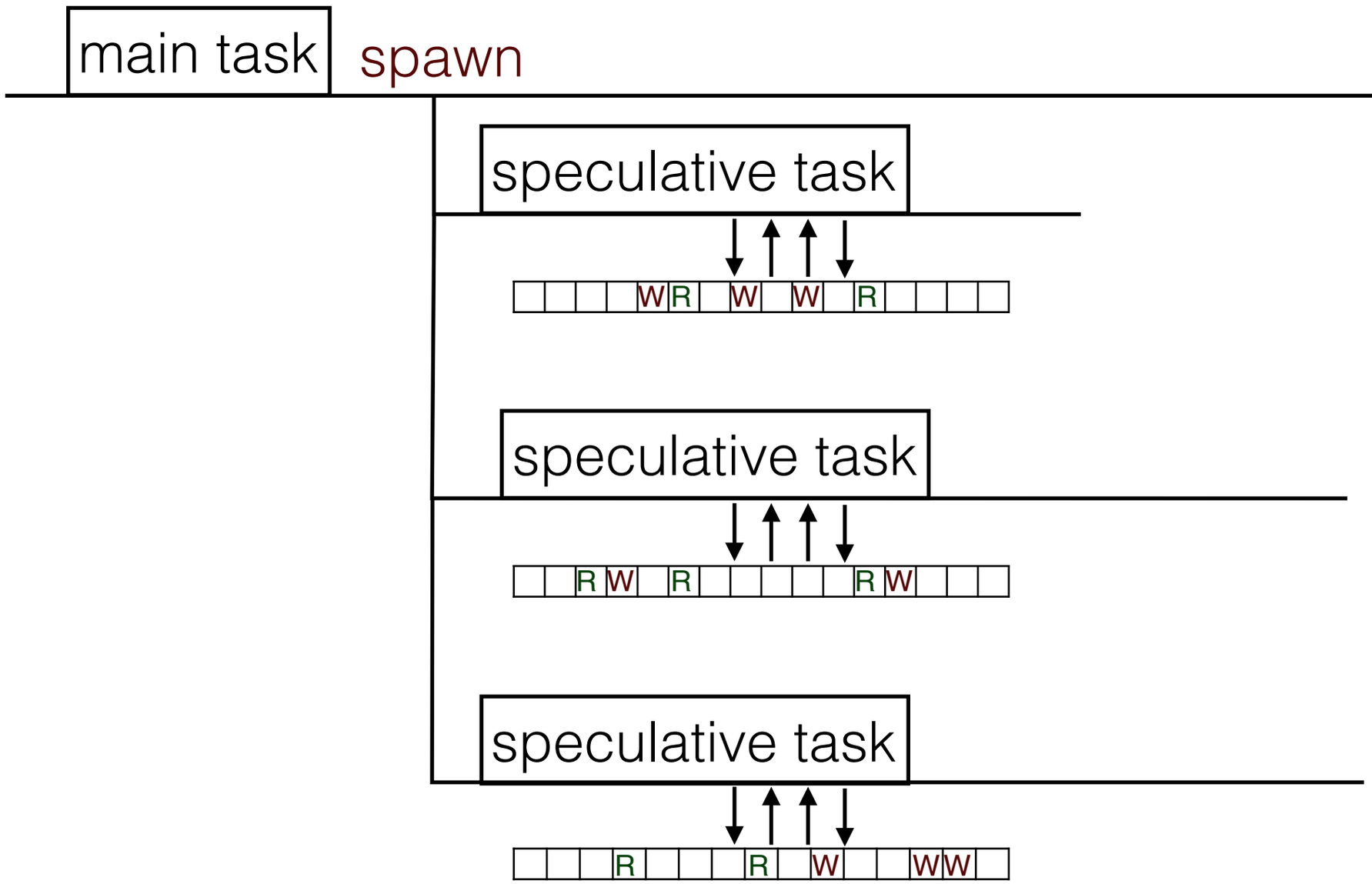
Execution in TLS



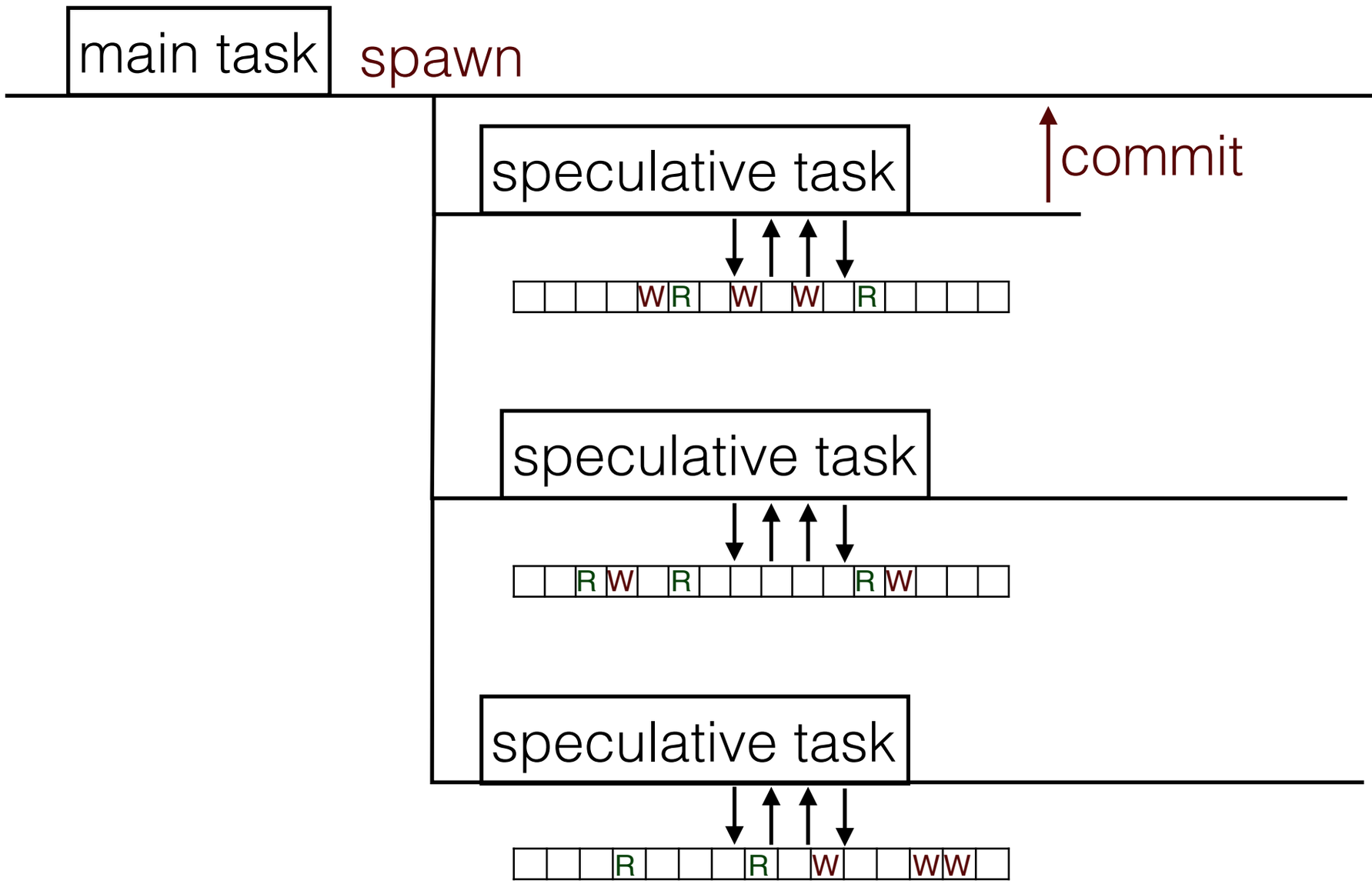
Execution in TLS



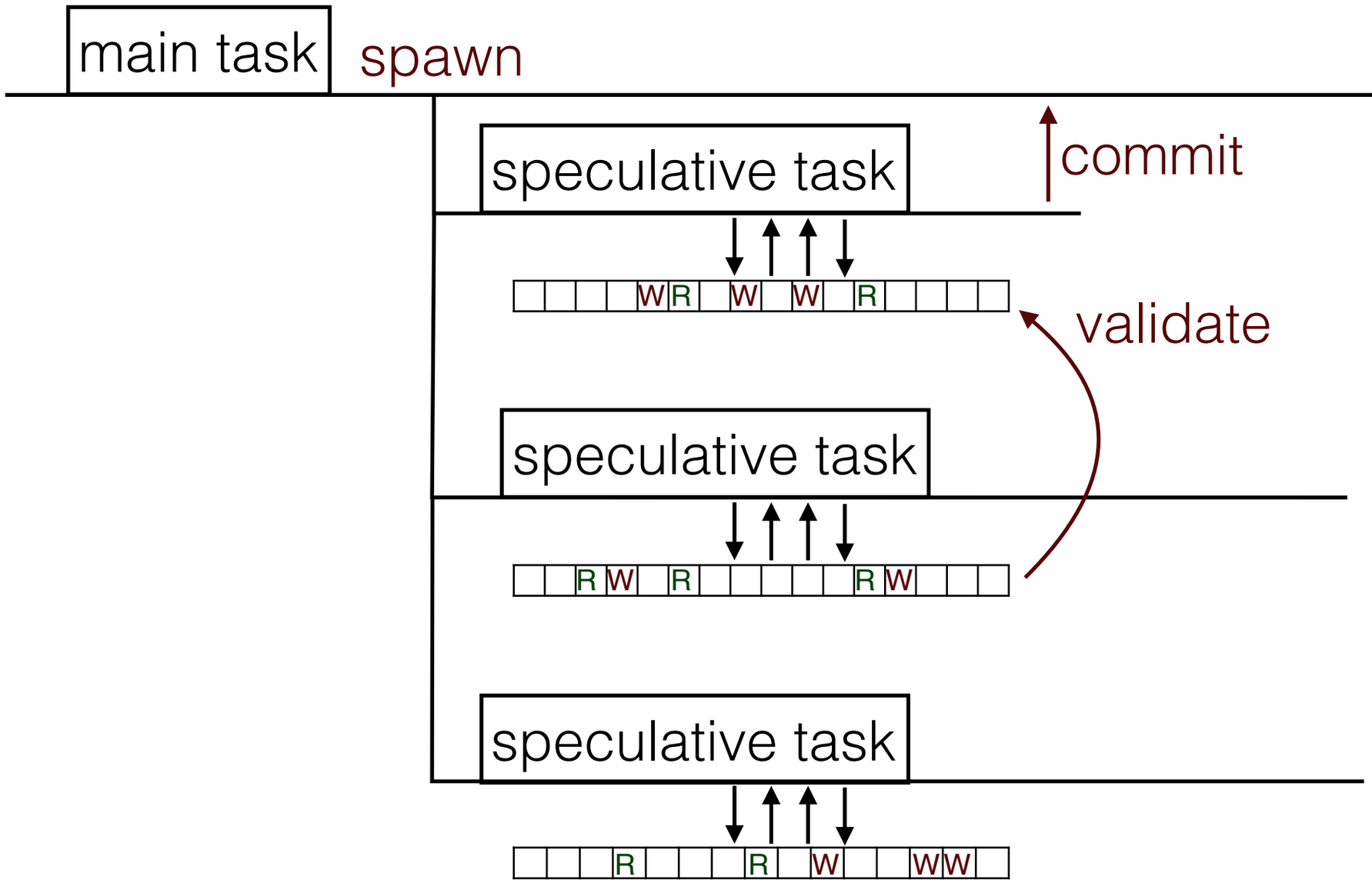
Execution in TLS



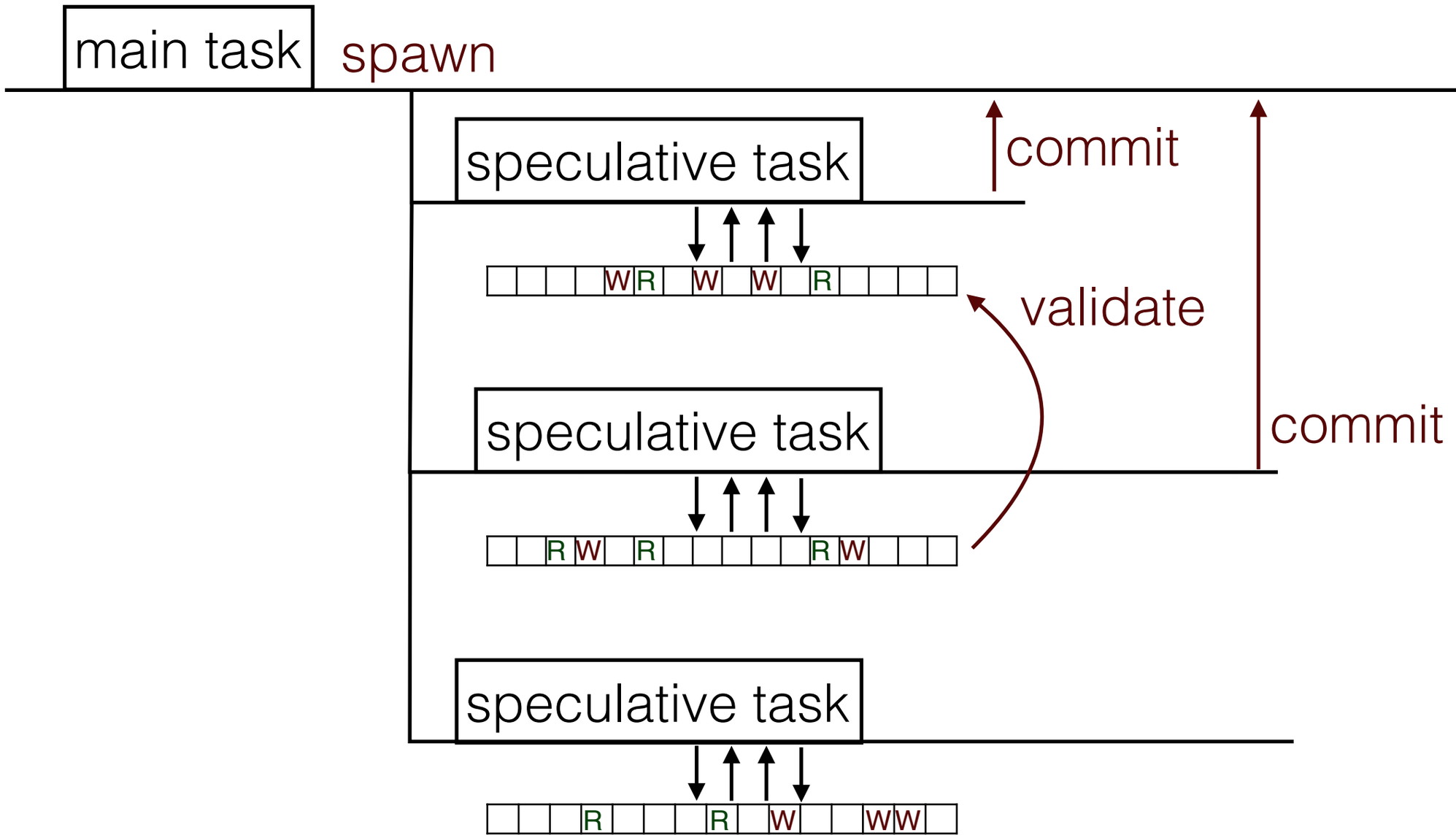
Execution in TLS



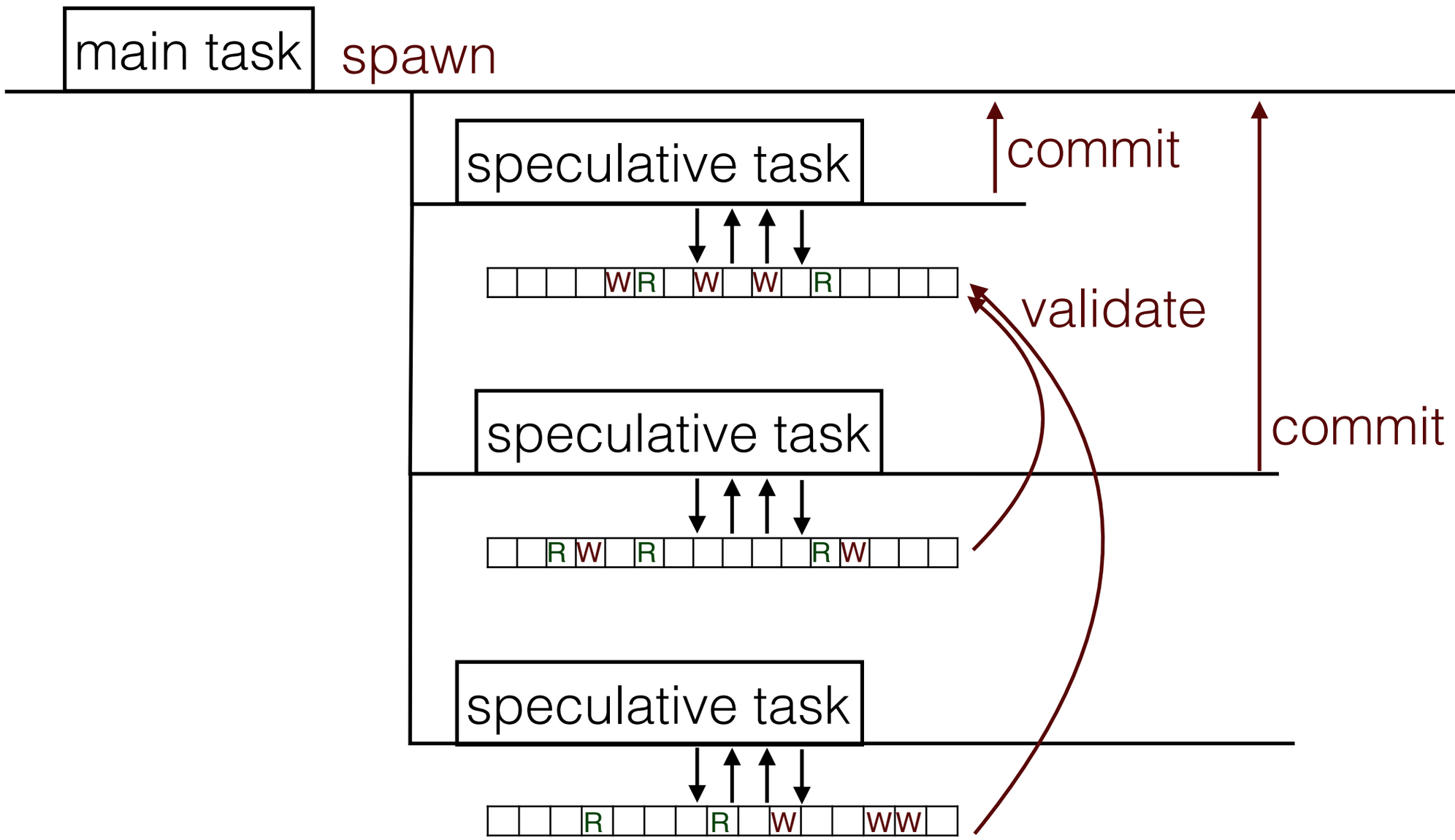
Execution in TLS



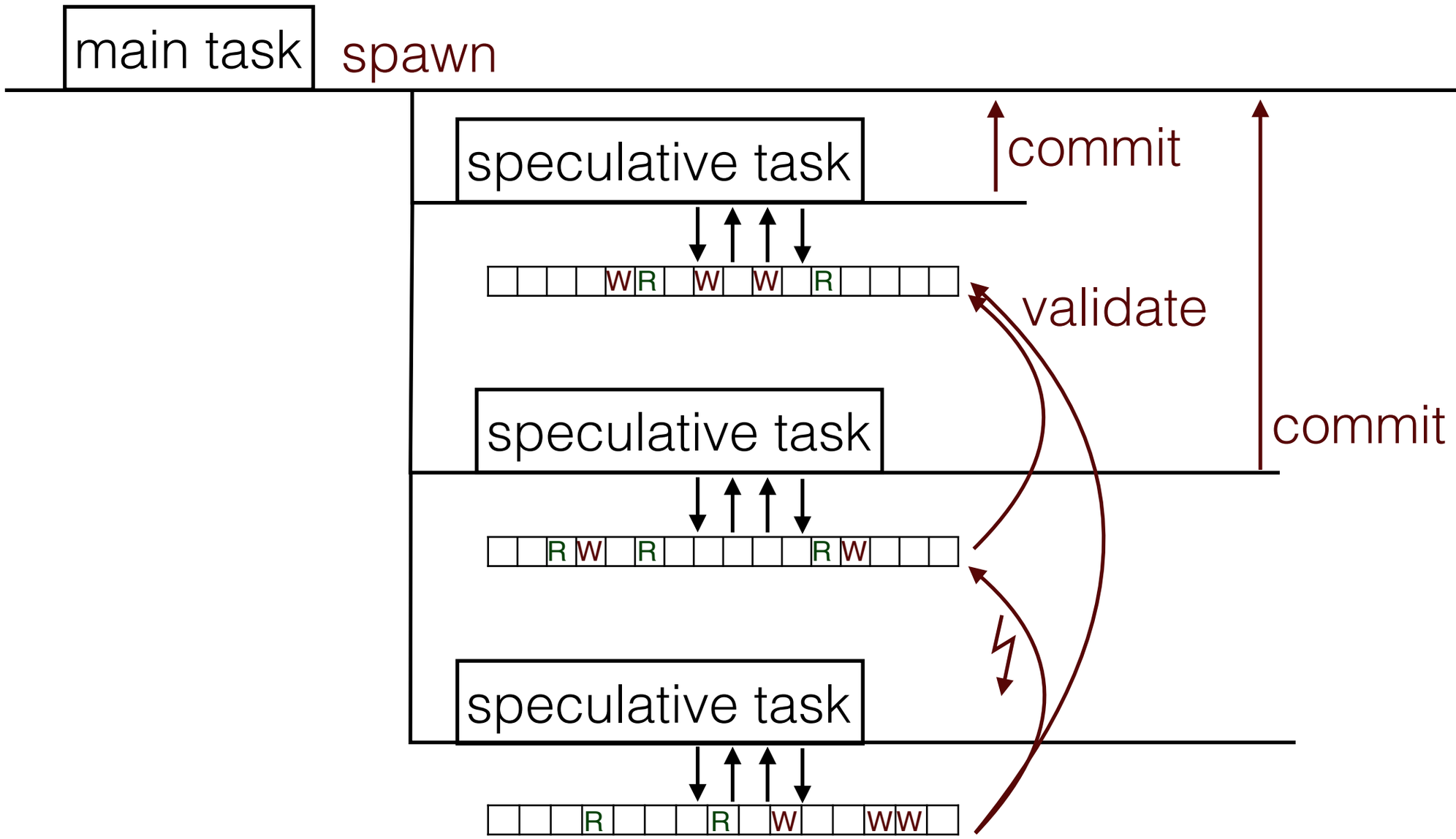
Execution in TLS



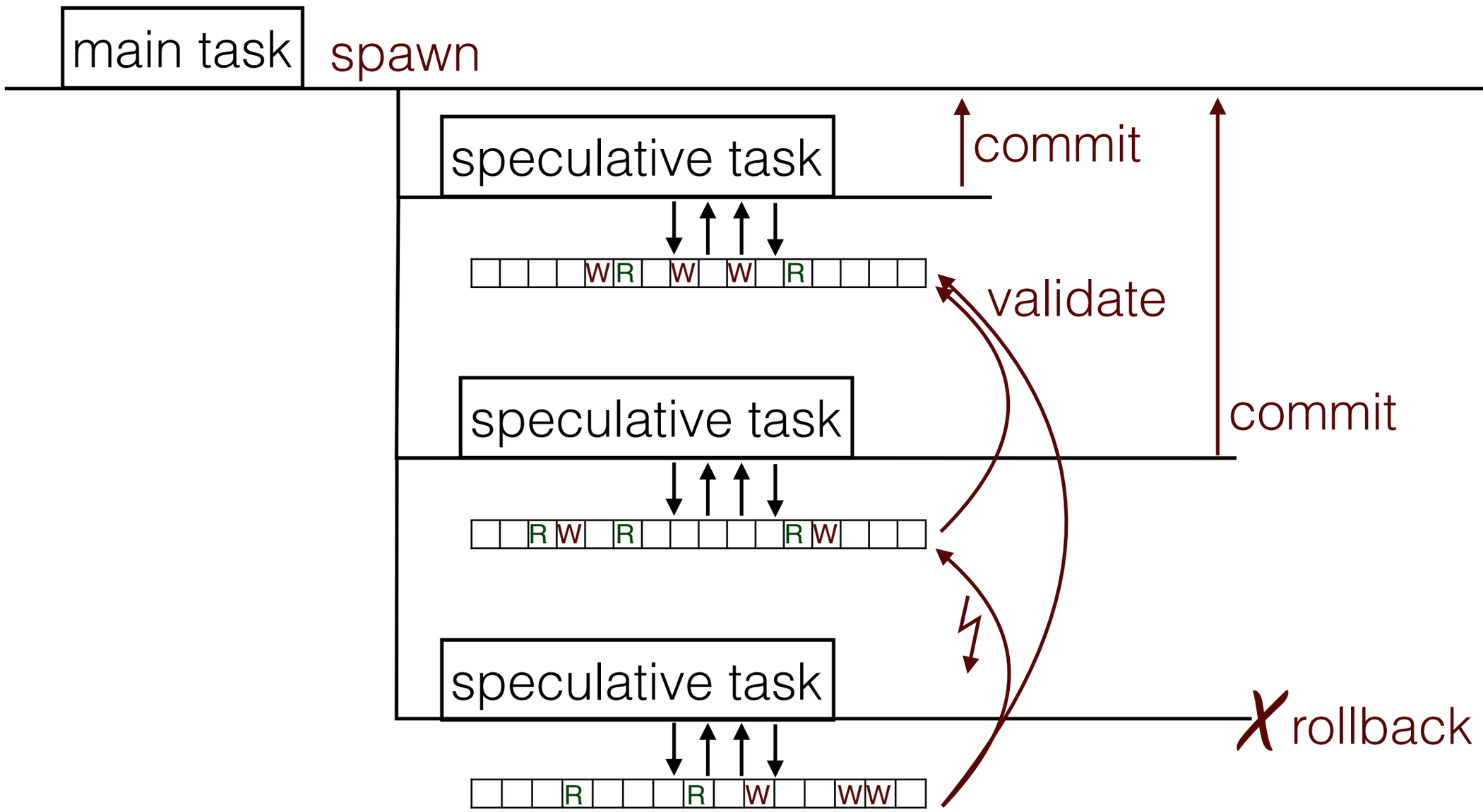
Execution in TLS



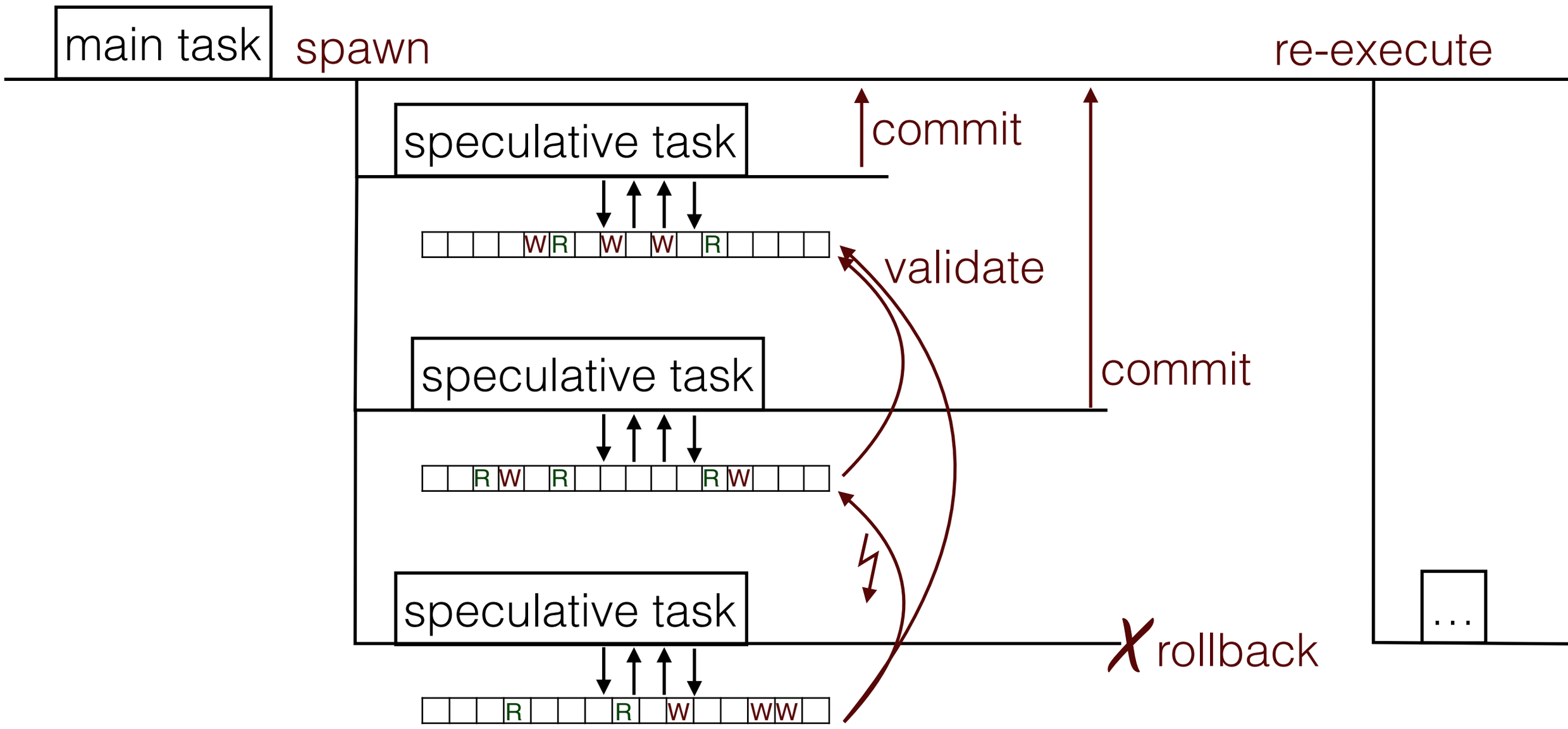
Execution in TLS



Execution in TLS



Execution in TLS



TLS Implementations

S-TLS VM-TLS

- Thread-Level Speculation in pure software
- similar to software transactional memory (STM)
- *explicit* tracking of memory accesses
- code *instrumentation*
- *multithreaded* in same address space
- first described in 1993 by Herlihy and Moss

TLS Implementations

S-TLS

VM-TLS

performance on STM benchmarks
(lots of small transactions)

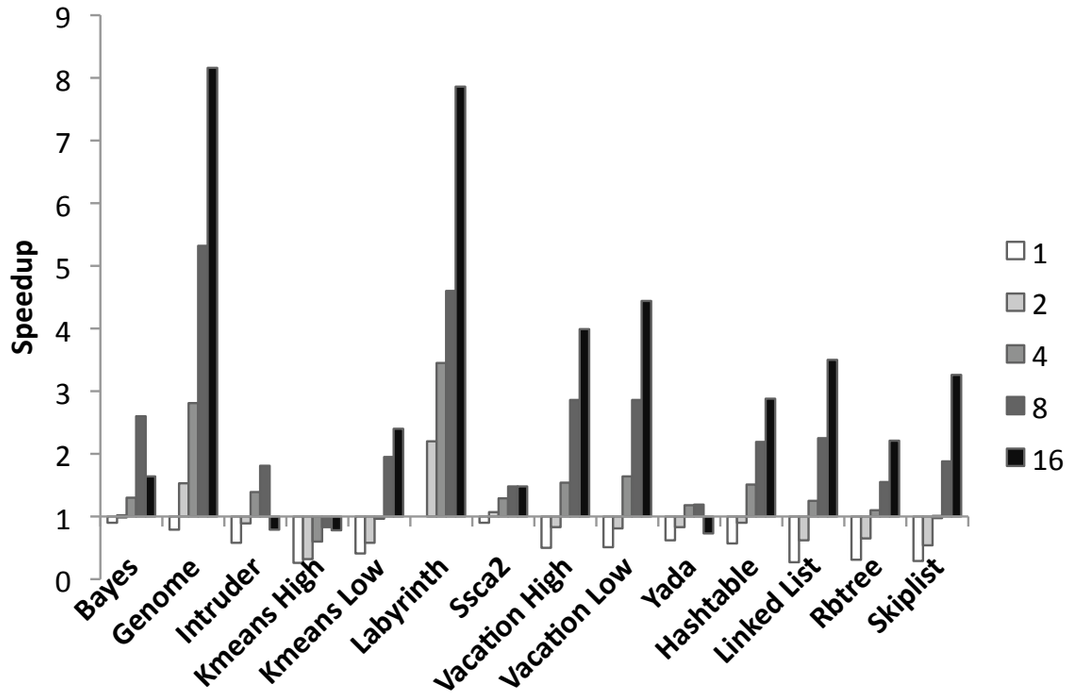
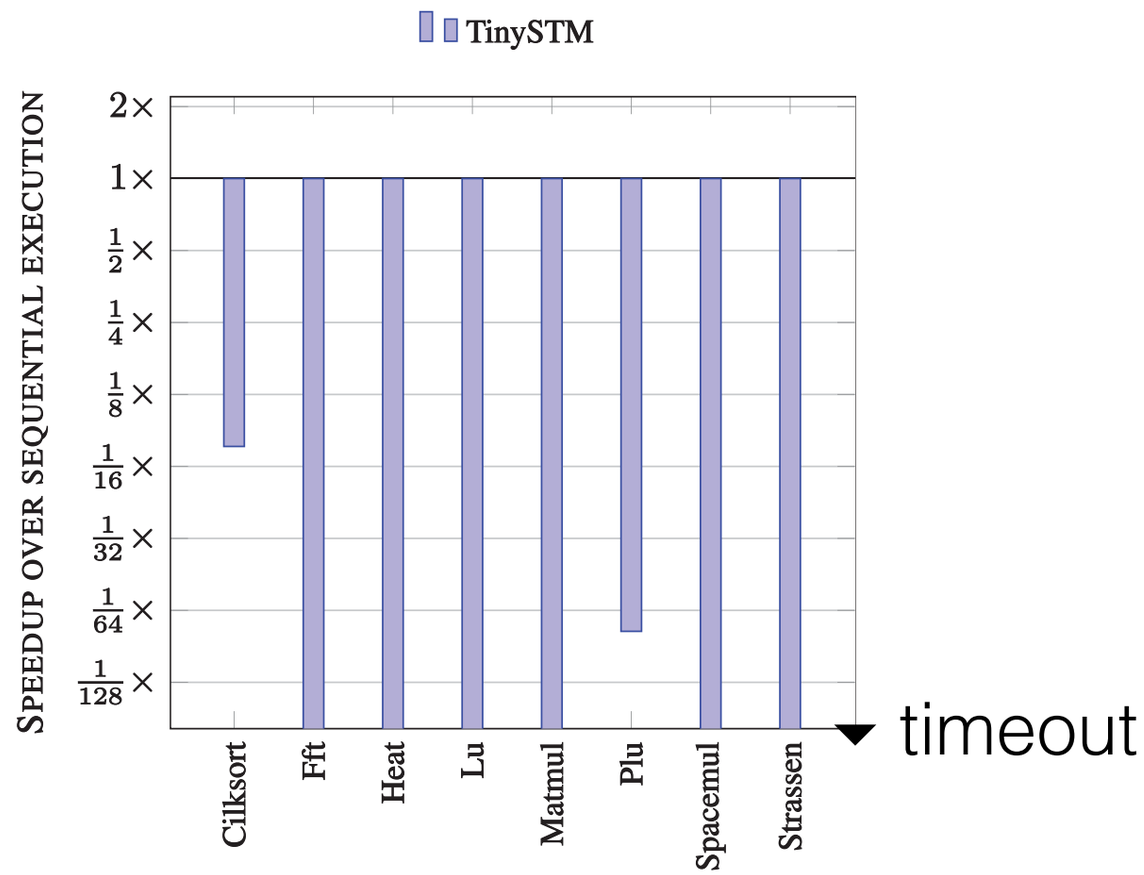


Figure 11. TinySTM performance with 16 core x86
source: Dragojevic et al., 2011, *Communications of the ACM*.

performance on TLS benchmarks
(few large transactions)



TLS Implementations

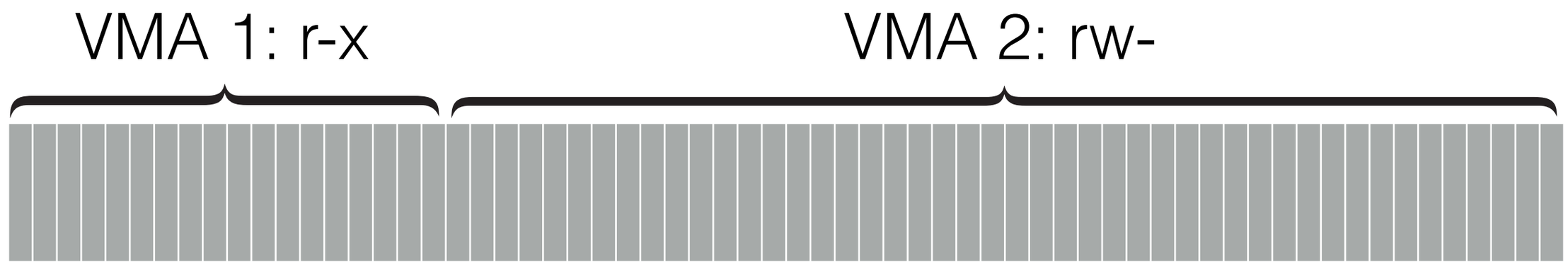
S-TLS

VM-TLS

- Virtual-memory based approaches
  - *implicit* tracking by protecting memory pages
  - segfault handler logs access & unprotects the page
  - can execute *arbitrary code*, even from compiled libraries
  - *multi-process execution* for separate address spaces
  - introduces in 2001 by Papadimitriou and Mowry

Memory Protection
In Virtual-Memory Based TLS

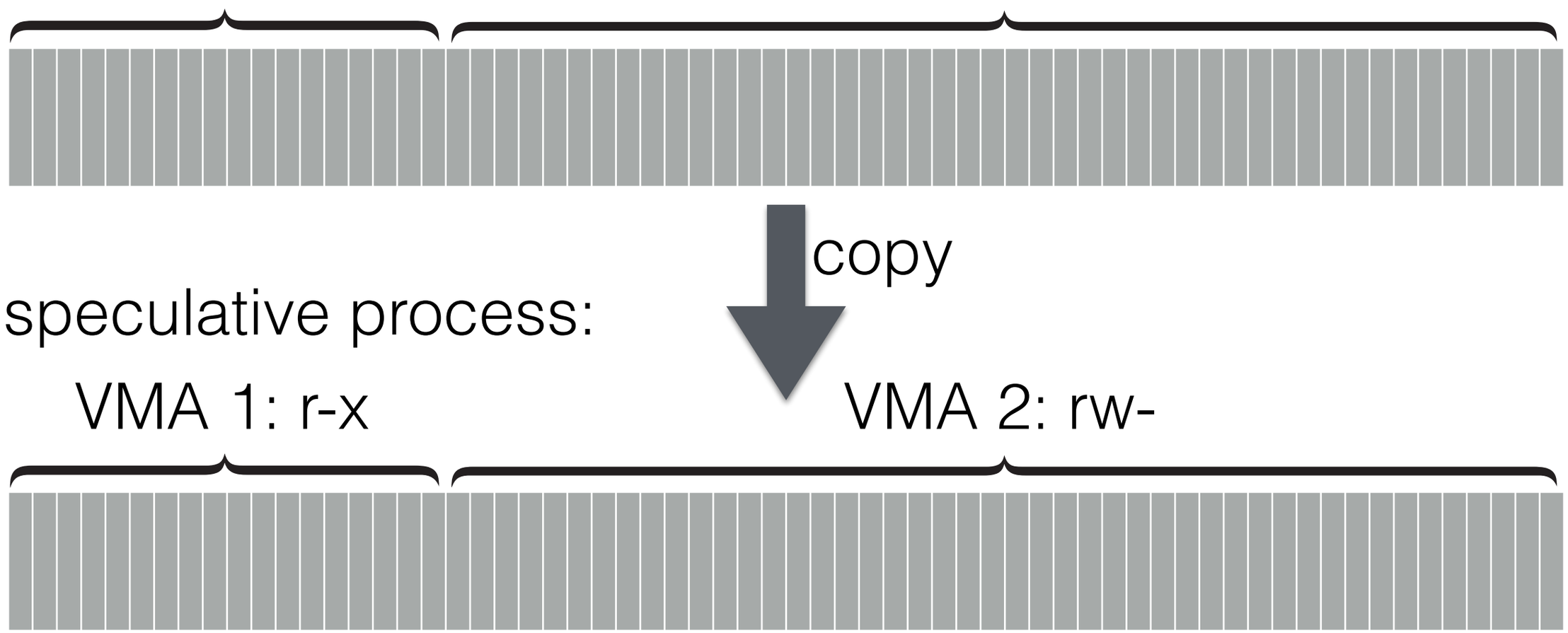
main process:



SAARLAND UNIVERSITY
GRADUATE SCHOOL OF COMPUTER SCIENCE

Memory Protection
In Virtual-Memory Based TLS

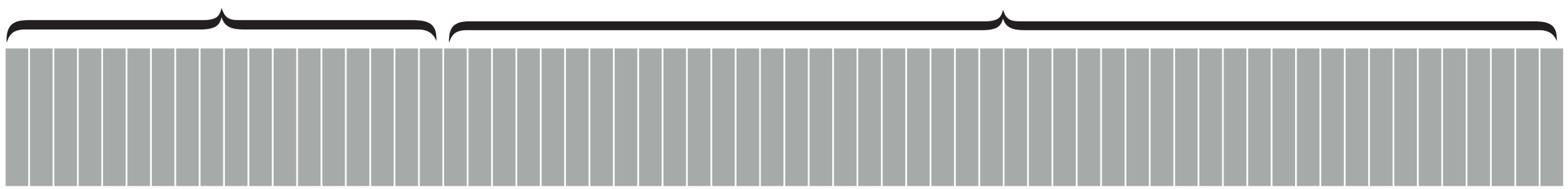
main process:



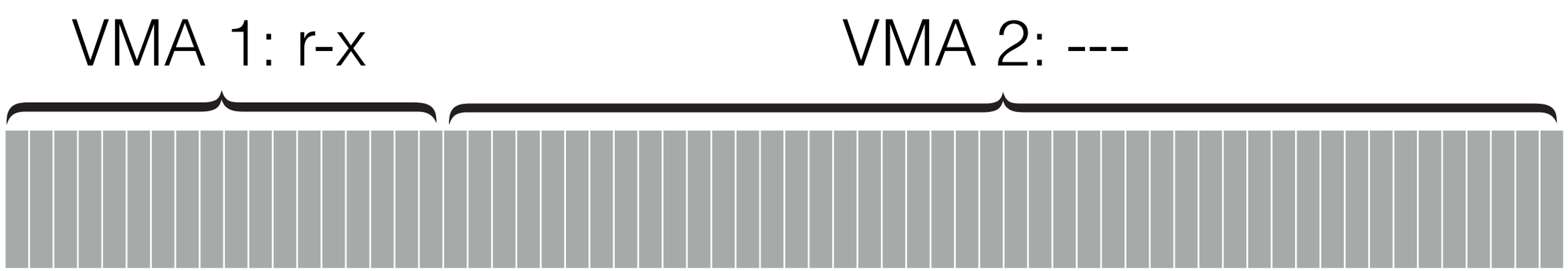
SAARLAND UNIVERSITY
GRADUATE SCHOOL OF COMPUTER SCIENCE

Memory Protection
In Virtual-Memory Based TLS

main process:



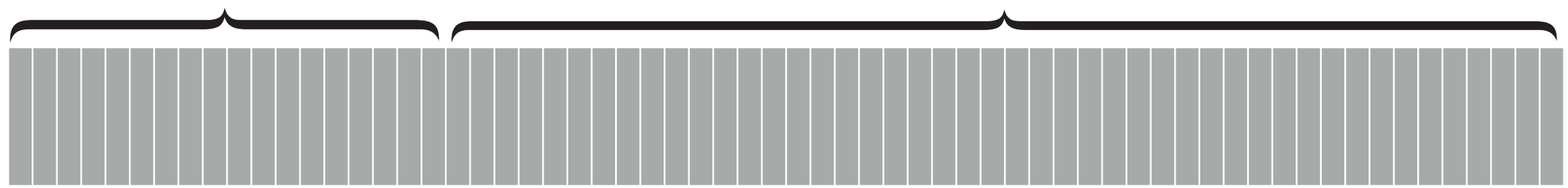
speculative process:



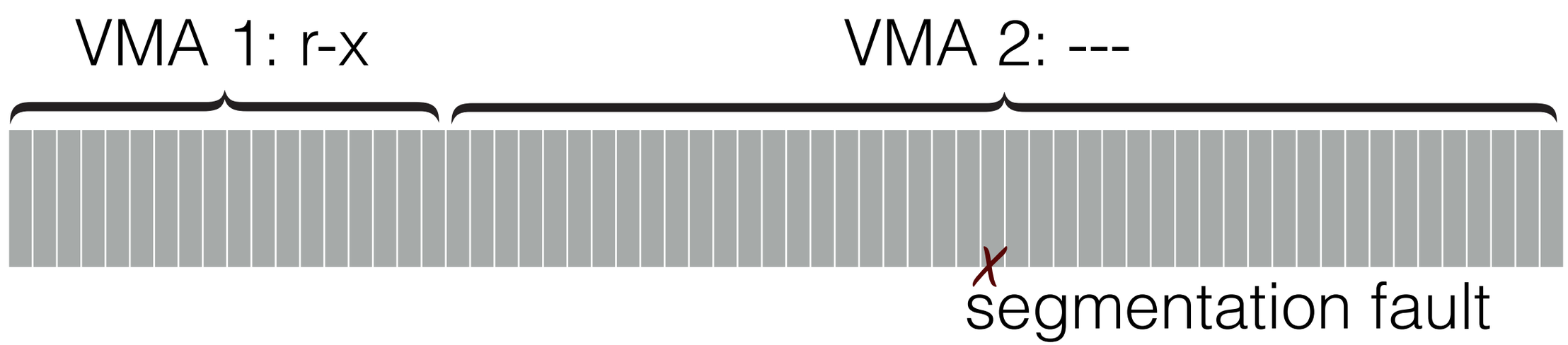
SAARLAND UNIVERSITY
GRADUATE SCHOOL OF COMPUTER SCIENCE

Memory Protection
In Virtual-Memory Based TLS

main process:



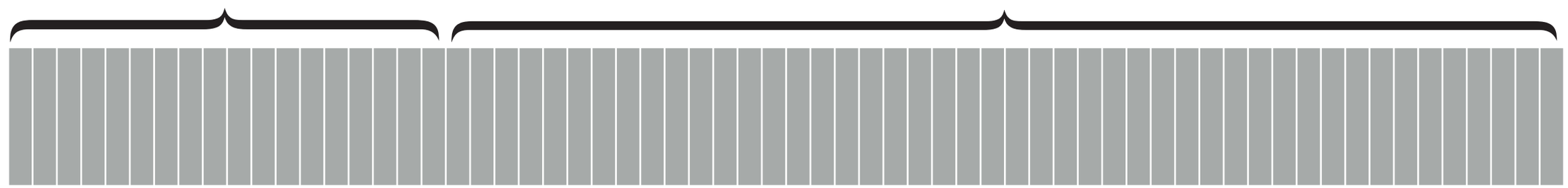
speculative process:



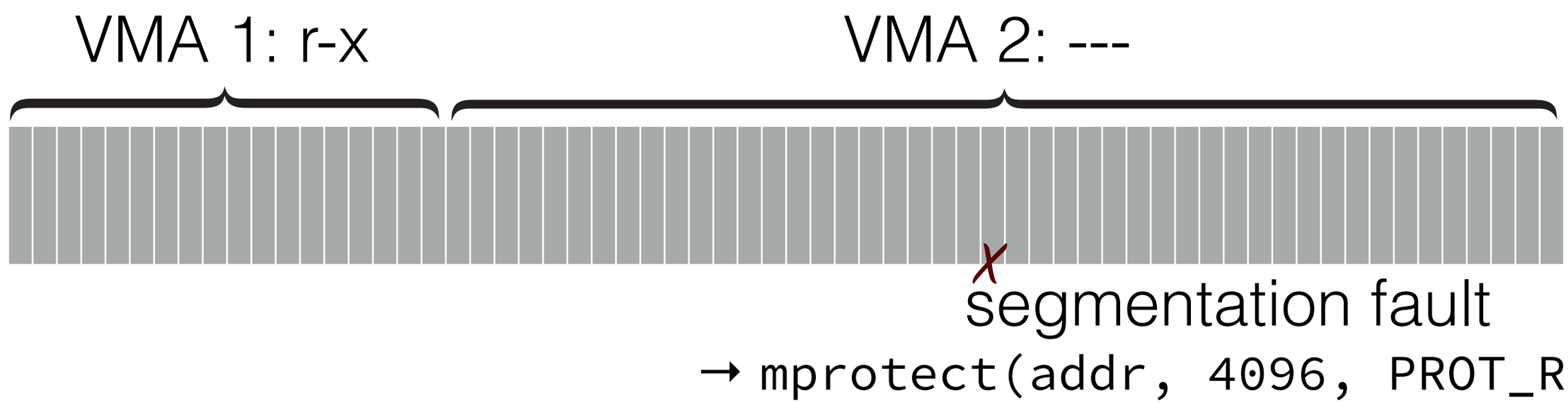
SAARLAND UNIVERSITY
GRADUATE SCHOOL OF COMPUTER SCIENCE

Memory Protection
In Virtual-Memory Based TLS

main process:



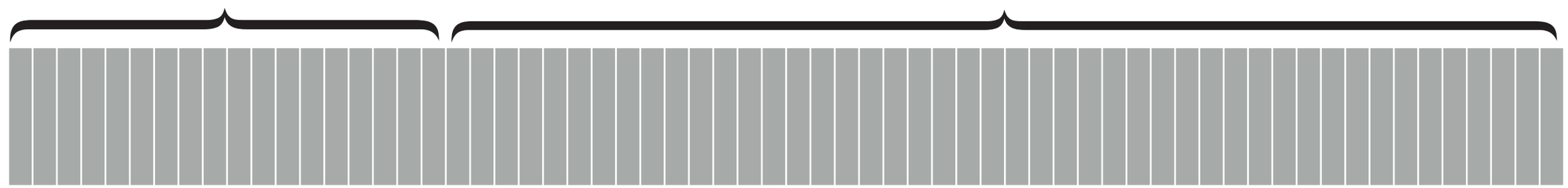
speculative process:



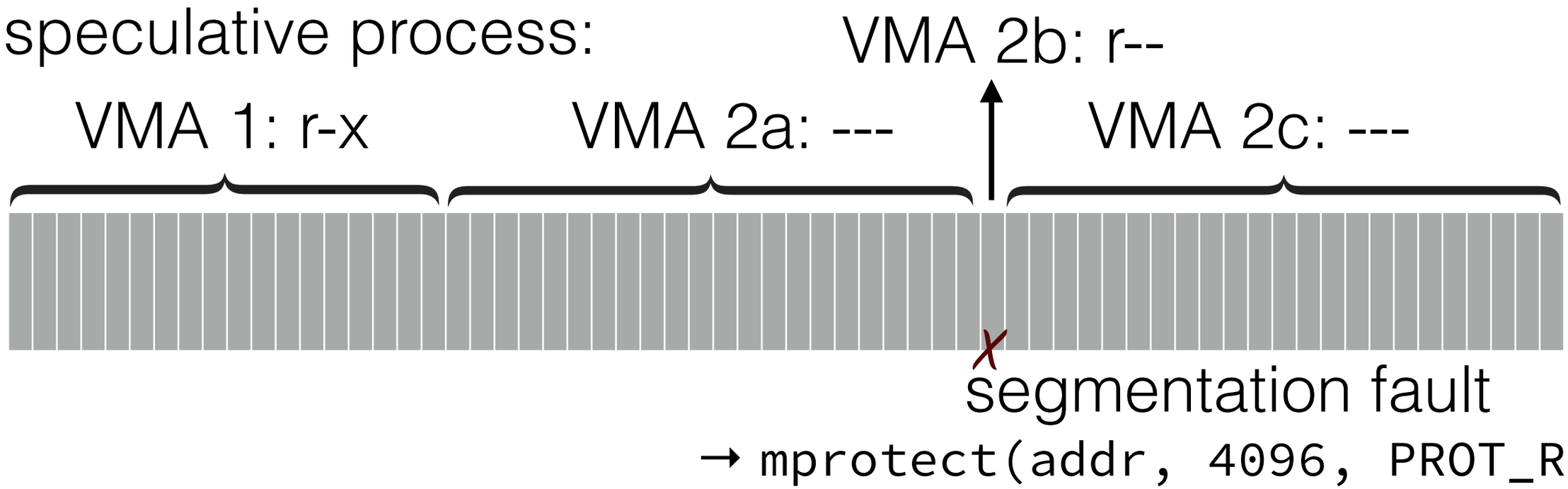
SAARLAND UNIVERSITY
GRADUATE SCHOOL OF COMPUTER SCIENCE

Memory Protection
In Virtual-Memory Based TLS

main process:

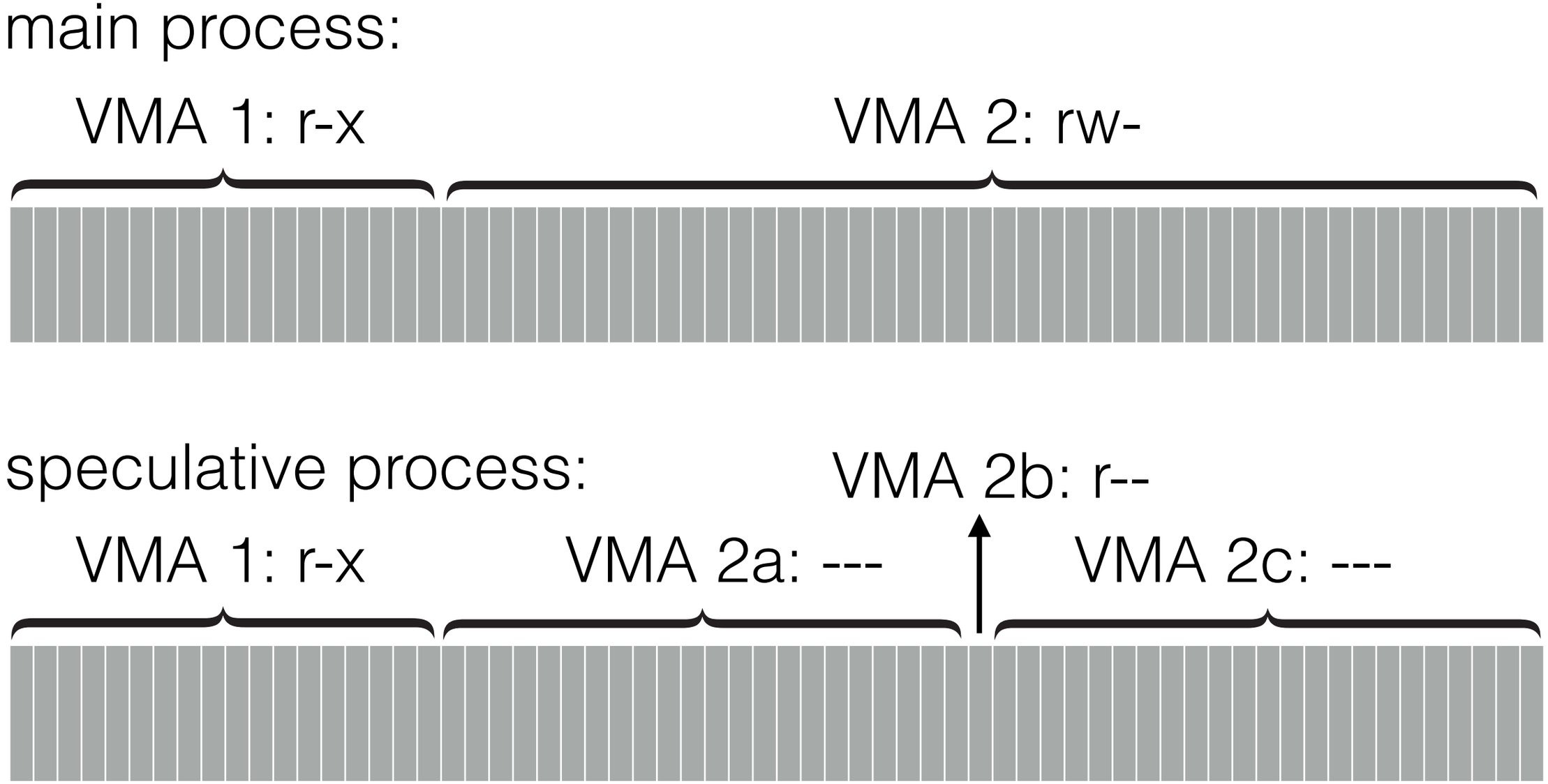


speculative process:



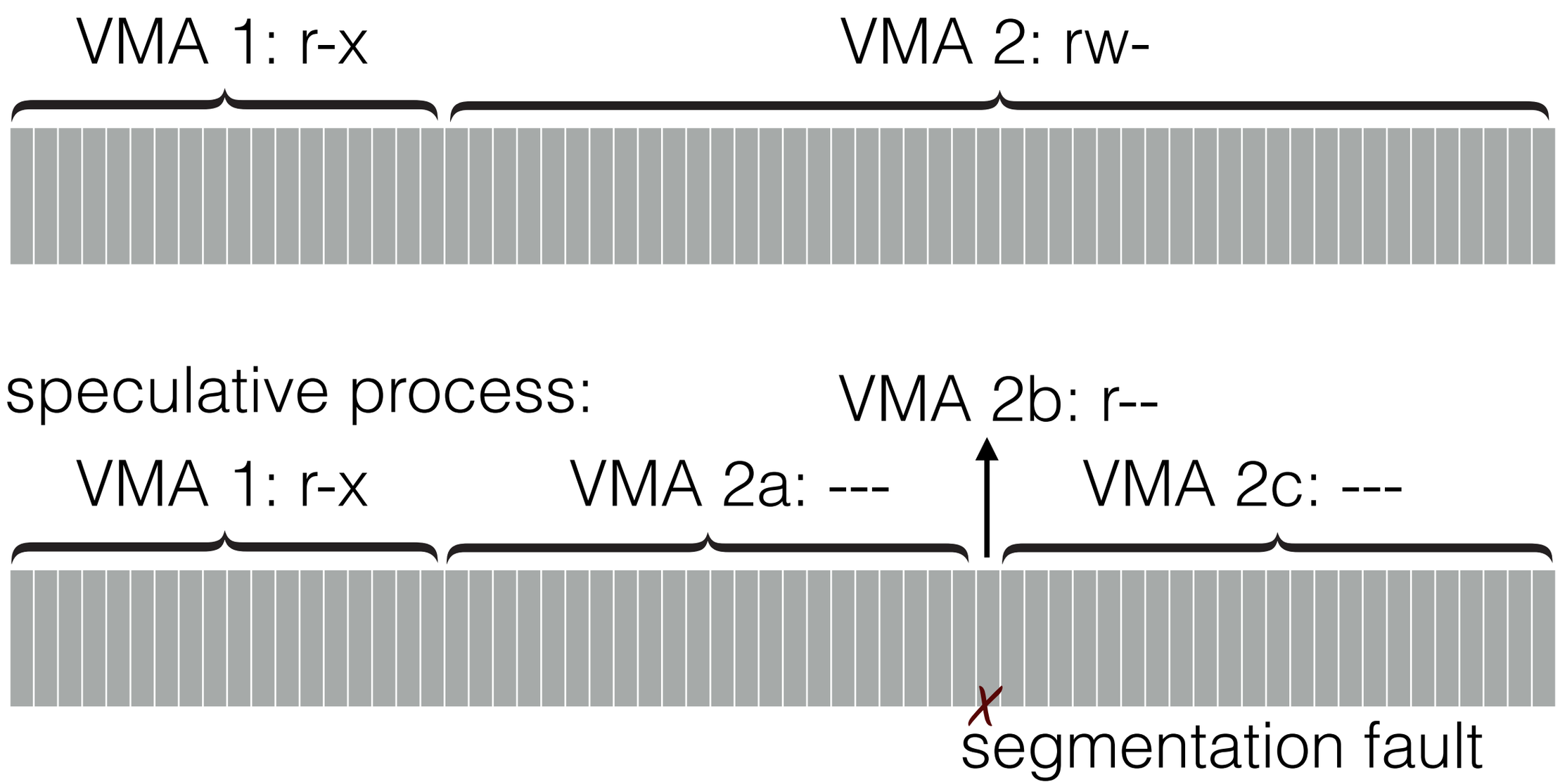
→ `mprotect(addr, 4096, PROT_READ)`

Memory Protection
In Virtual-Memory Based TLS



SAARLAND UNIVERSITY
GRADUATE SCHOOL OF COMPUTER SCIENCE

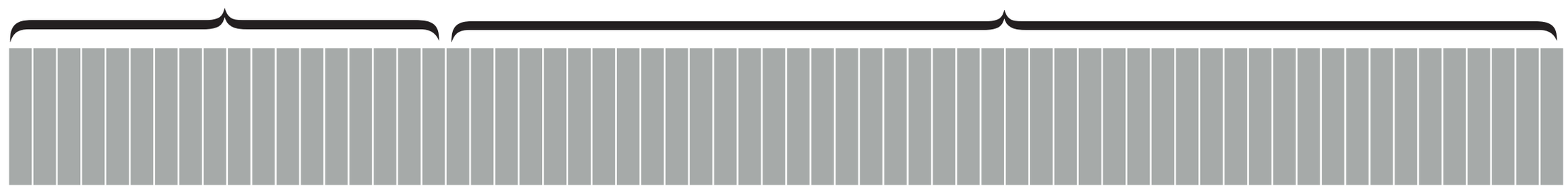
Memory Protection
In Virtual-Memory Based TLS



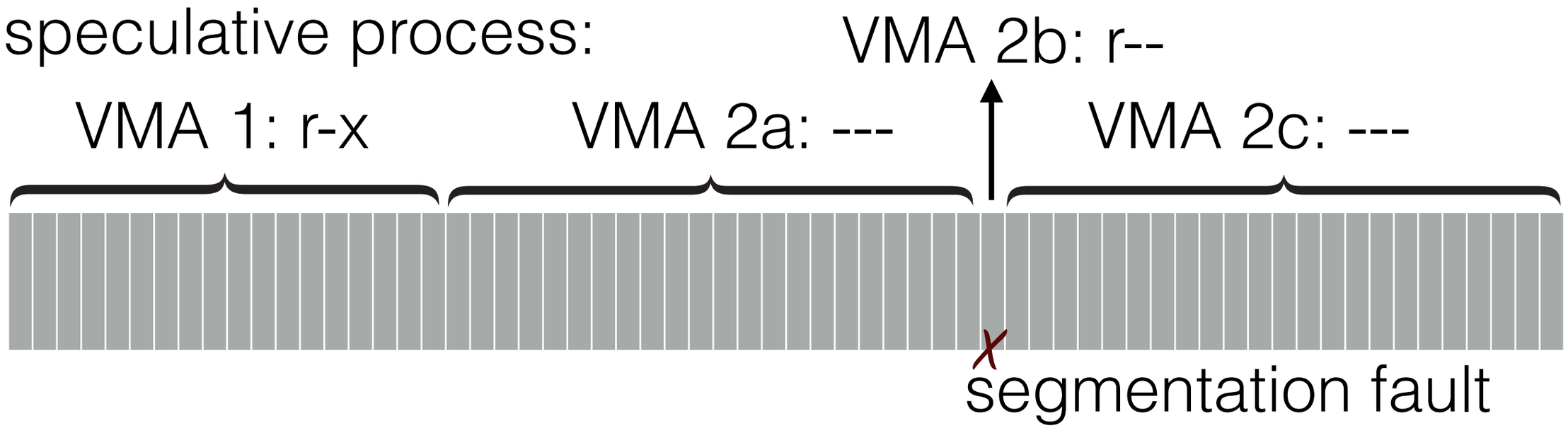
SAARLAND UNIVERSITY
GRADUATE SCHOOL OF COMPUTER SCIENCE

Memory Protection
In Virtual-Memory Based TLS

main process:



speculative process:

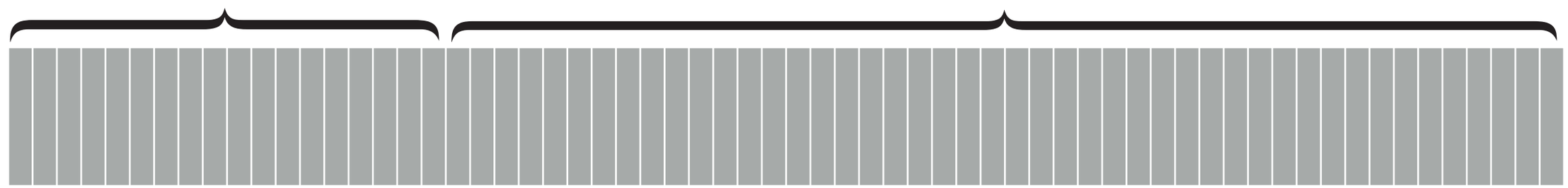


→ `mprotect(addr, 4096, PROT_READ|PROT_WRITE)`

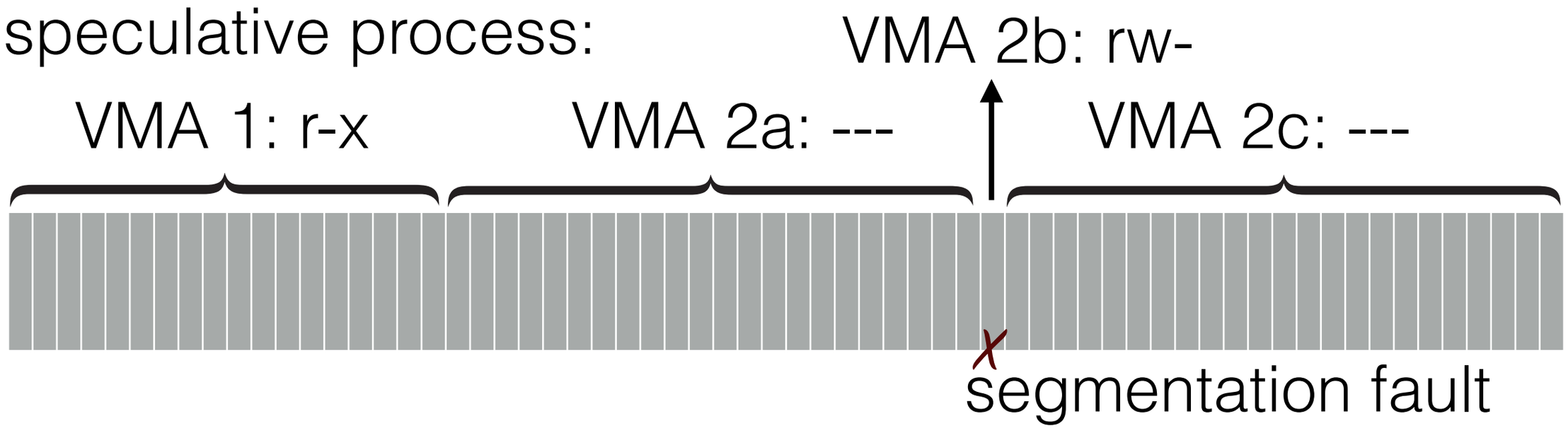
SAARLAND UNIVERSITY
GRADUATE SCHOOL OF COMPUTER SCIENCE

Memory Protection
In Virtual-Memory Based TLS

main process:



speculative process:

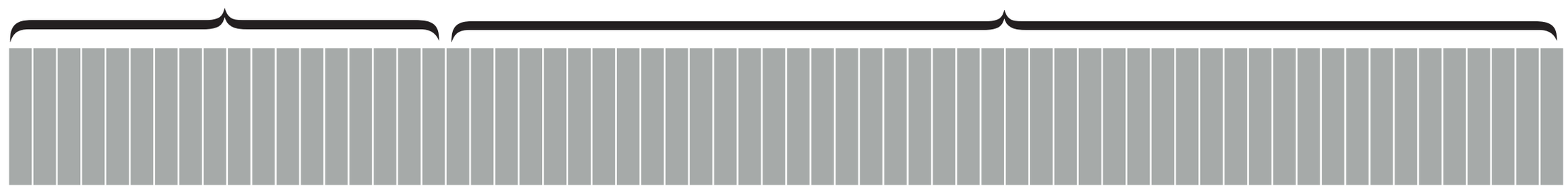


→ `mprotect(addr, 4096, PROT_READ|PROT_WRITE)`

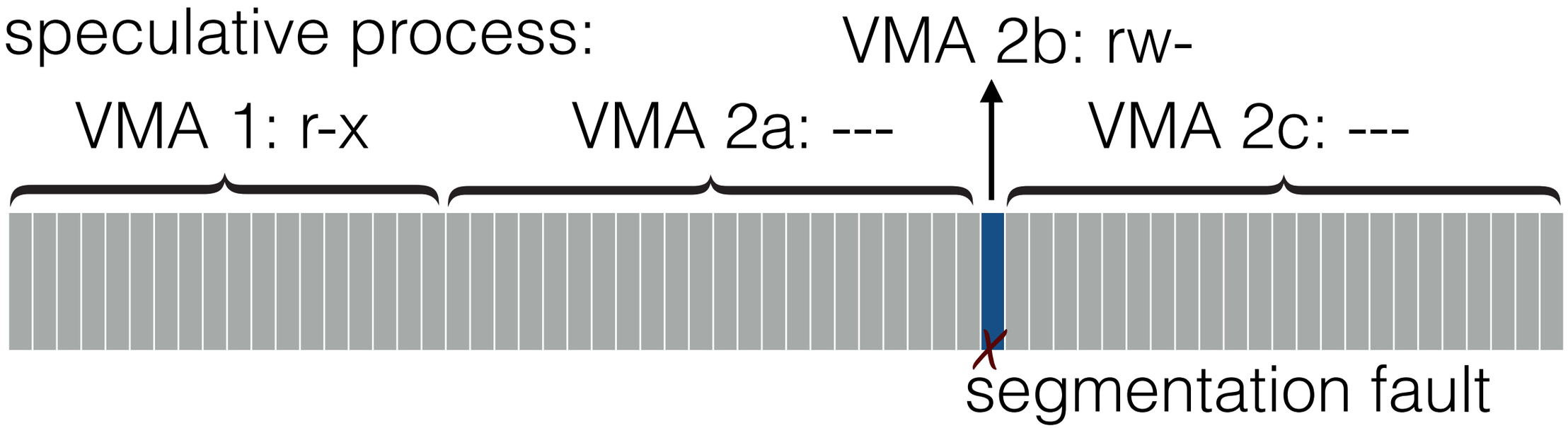
SAARLAND UNIVERSITY
GRADUATE SCHOOL OF COMPUTER SCIENCE

Memory Protection
In Virtual-Memory Based TLS

main process:

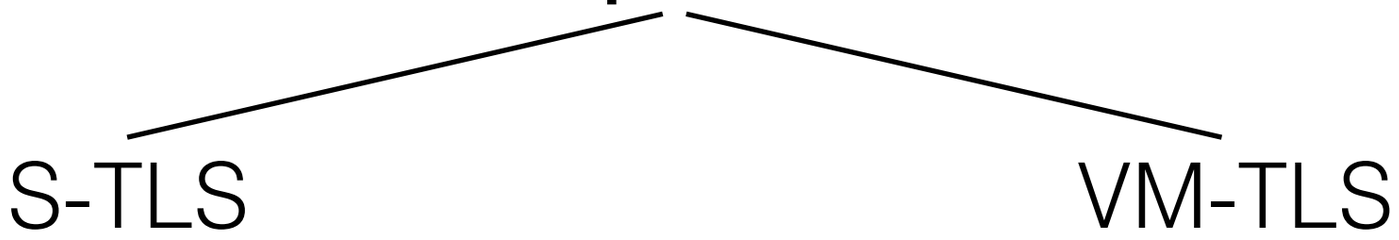


speculative process:



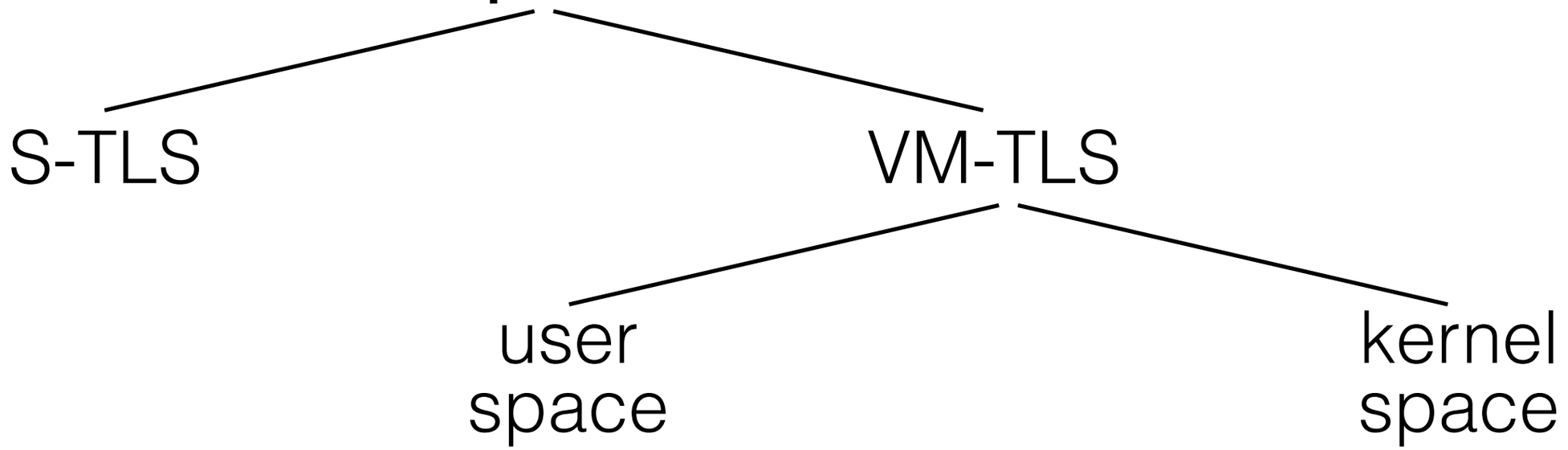
→ `mprotect(addr, 4096, PROT_READ|PROT_WRITE)`

TLS Implementations



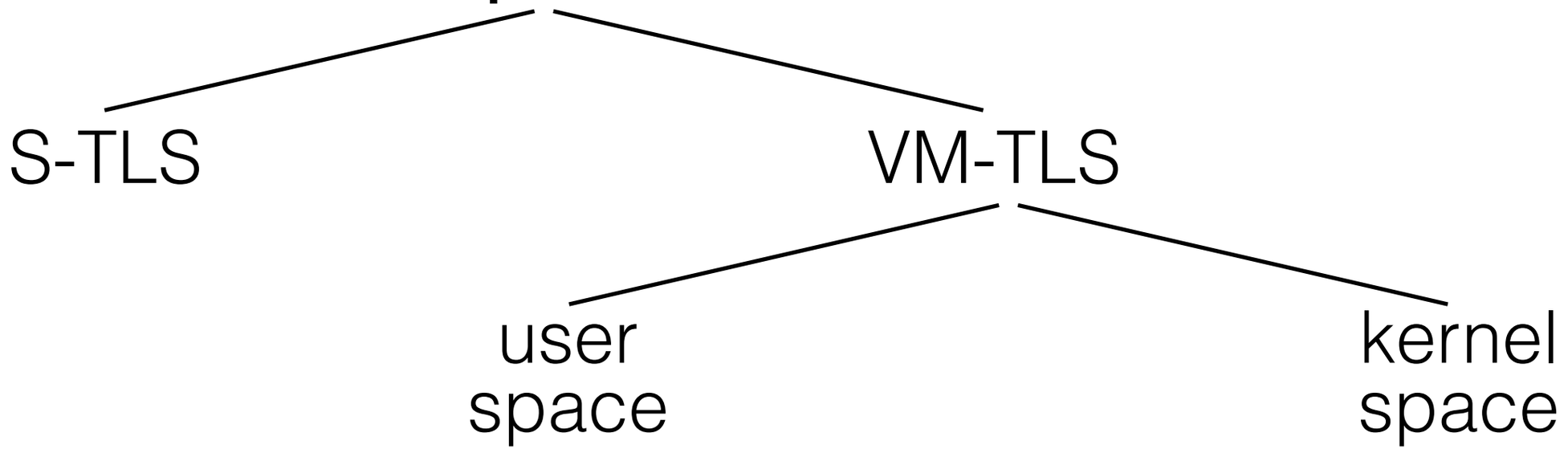
	user space library	instrumentation	setup cost	execution overhead	commit cost	granularity	I/O isolation
S-TLS	✓	✓	$	$$$ (per access)	$$$	✓ (arbitrary)	—
VM-TLS	✓	—	$$$	$$ (per page)	$$	✗ (memory page)	—

TLS Implementations



	user space library	instrumentation	setup cost	execution overhead	commit cost	granularity	I/O isolation
S-TLS	✓	✓	$	$$$ (per access)	$$$	✓ (arbitrary)	—
VM-TLS	✓	—	$$$	$$ (per page)	$$	✗ (memory page)	—

TLS Implementations



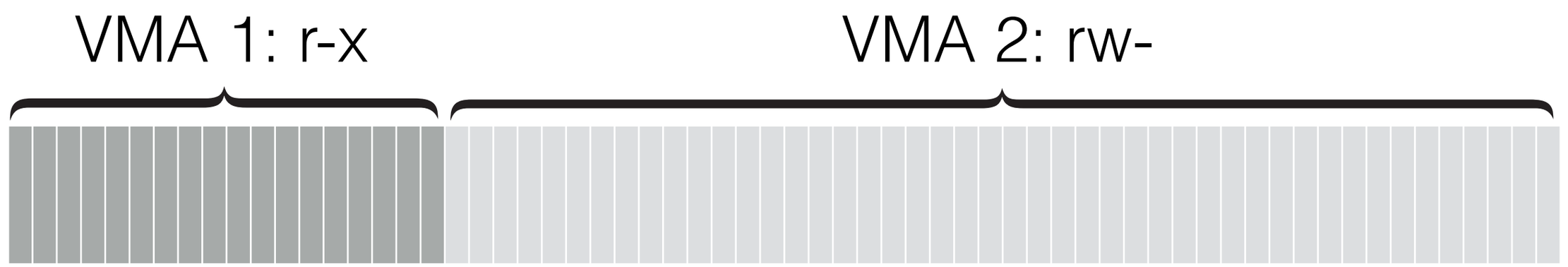
	user space library	instrumentation	setup cost	execution overhead	commit cost	granularity	I/O isolation
S-TLS	✓	✓	$	$$$ (per access)	$$$	✓ (arbitrary)	—
U-TLS	✓	—	$$$	$$ (per page)	$$	✗ (memory page)	—
K-TLS	—	—	$$	$ (per page)	$	✗ (memory page)	✓

SAARLAND UNIVERSITY
GRADUATE SCHOOL OF COMPUTER SCIENCE

Memory Protection
In Kernel Space

main process:

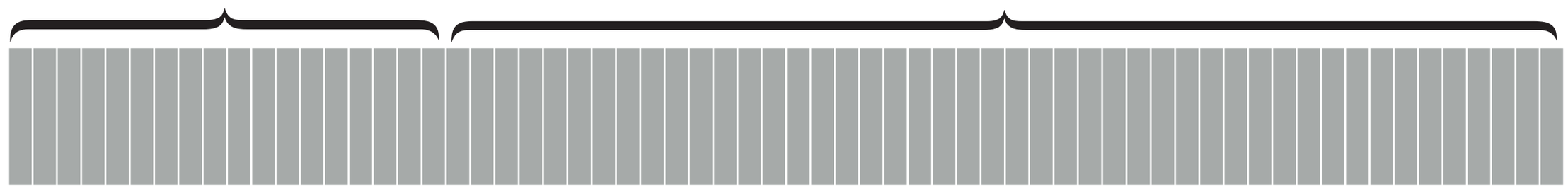
speculative process:



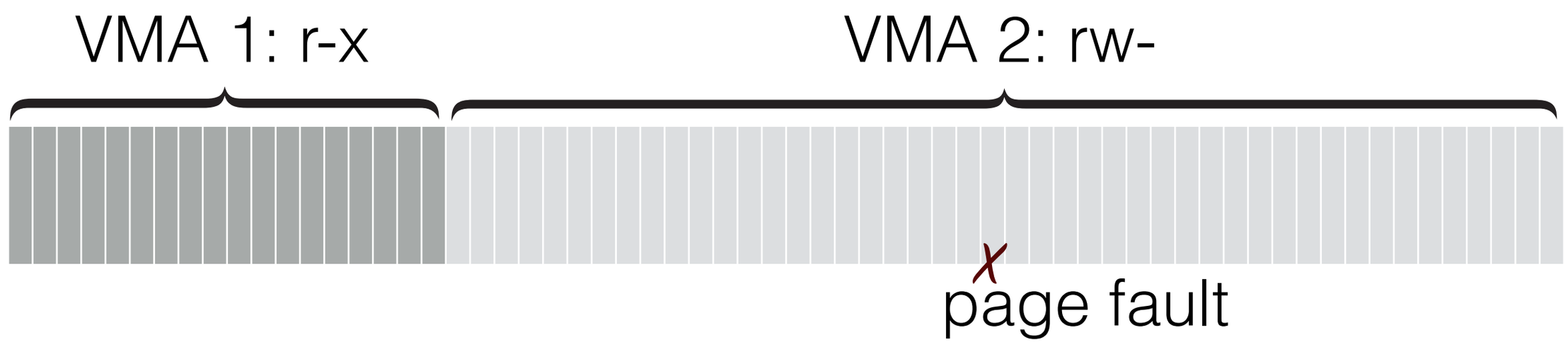
SAARLAND UNIVERSITY
GRADUATE SCHOOL OF COMPUTER SCIENCE

Memory Protection
In Kernel Space

main process:



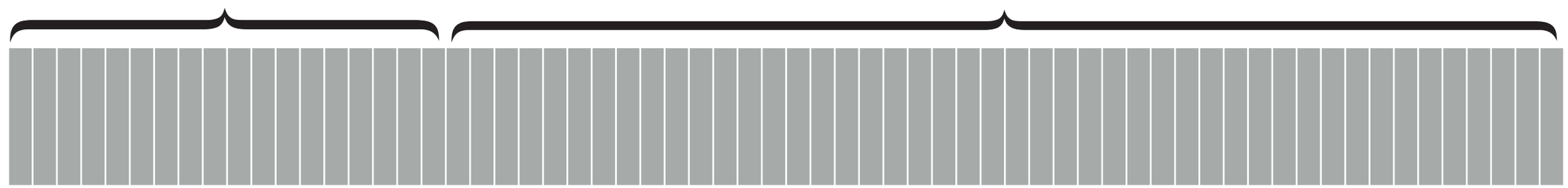
speculative process:



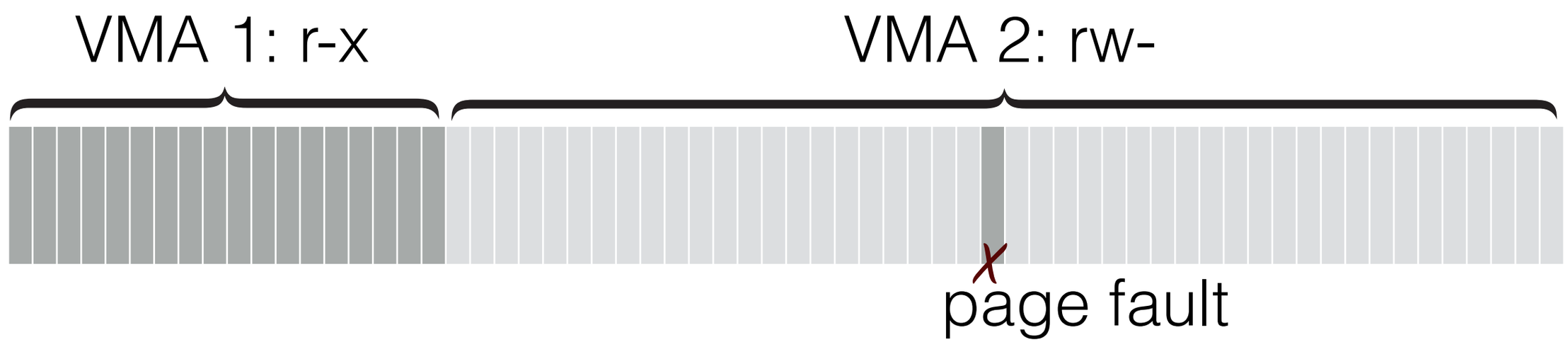
SAARLAND UNIVERSITY
GRADUATE SCHOOL OF COMPUTER SCIENCE

Memory Protection
In Kernel Space

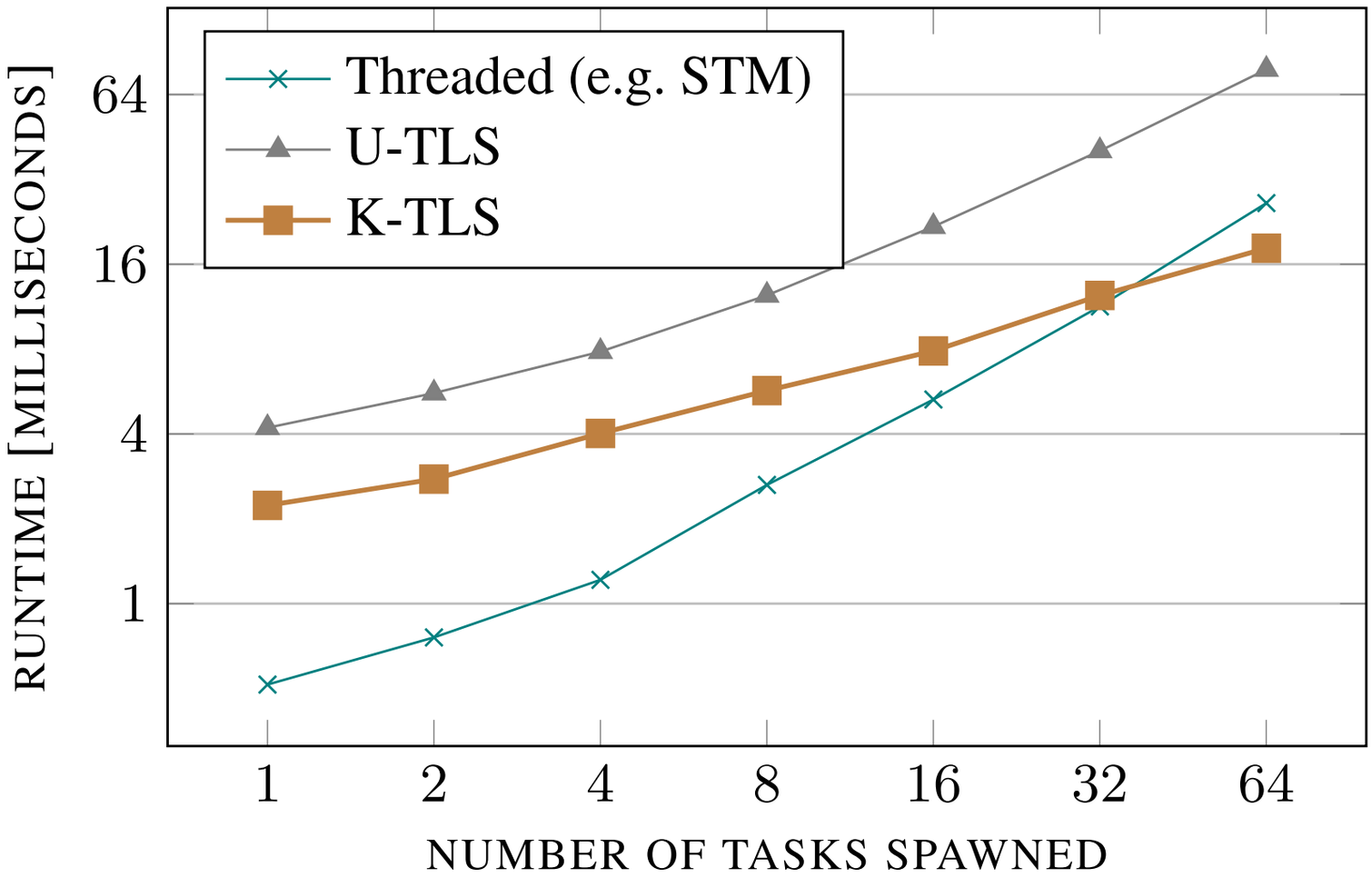
main process:



speculative process:

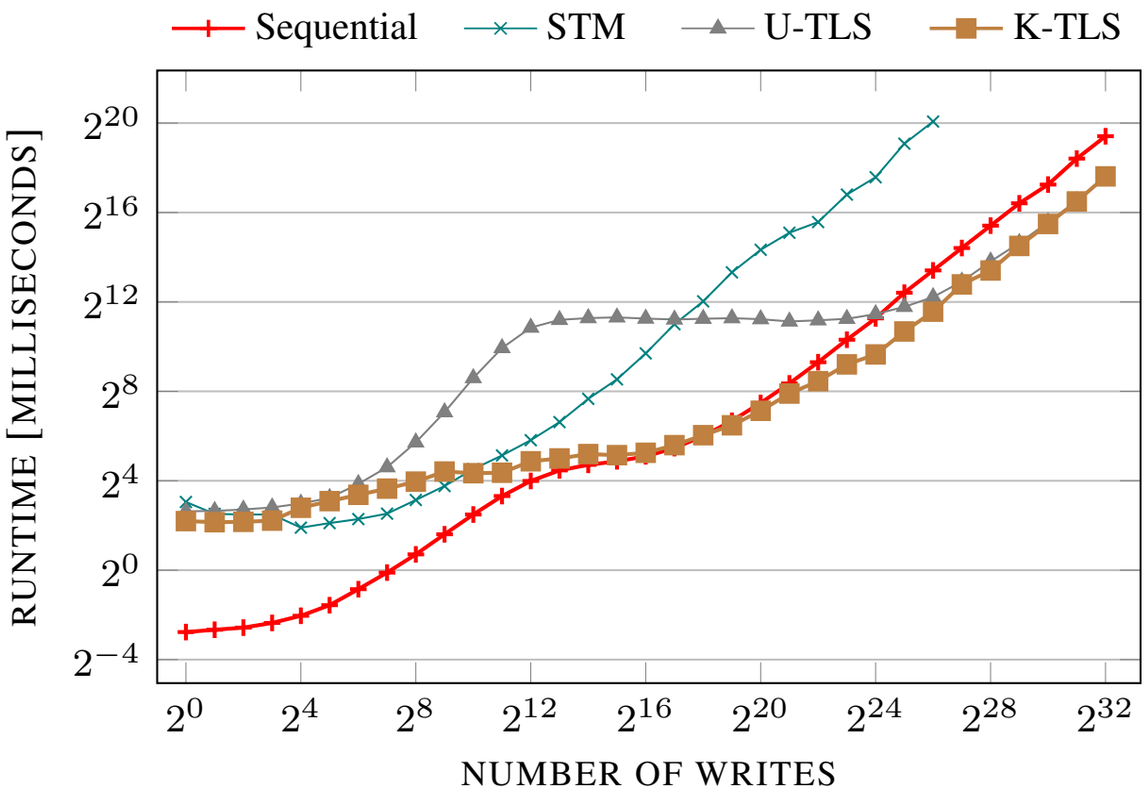


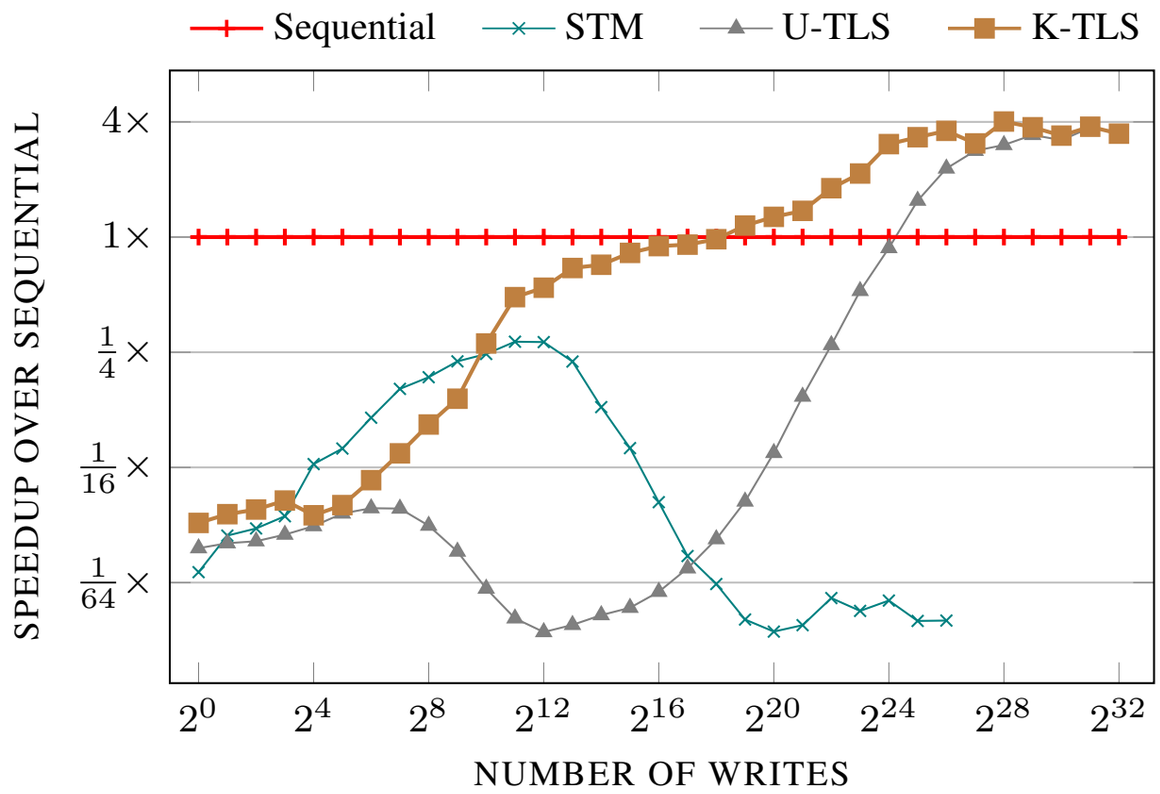
Evaluation
Spawn Time



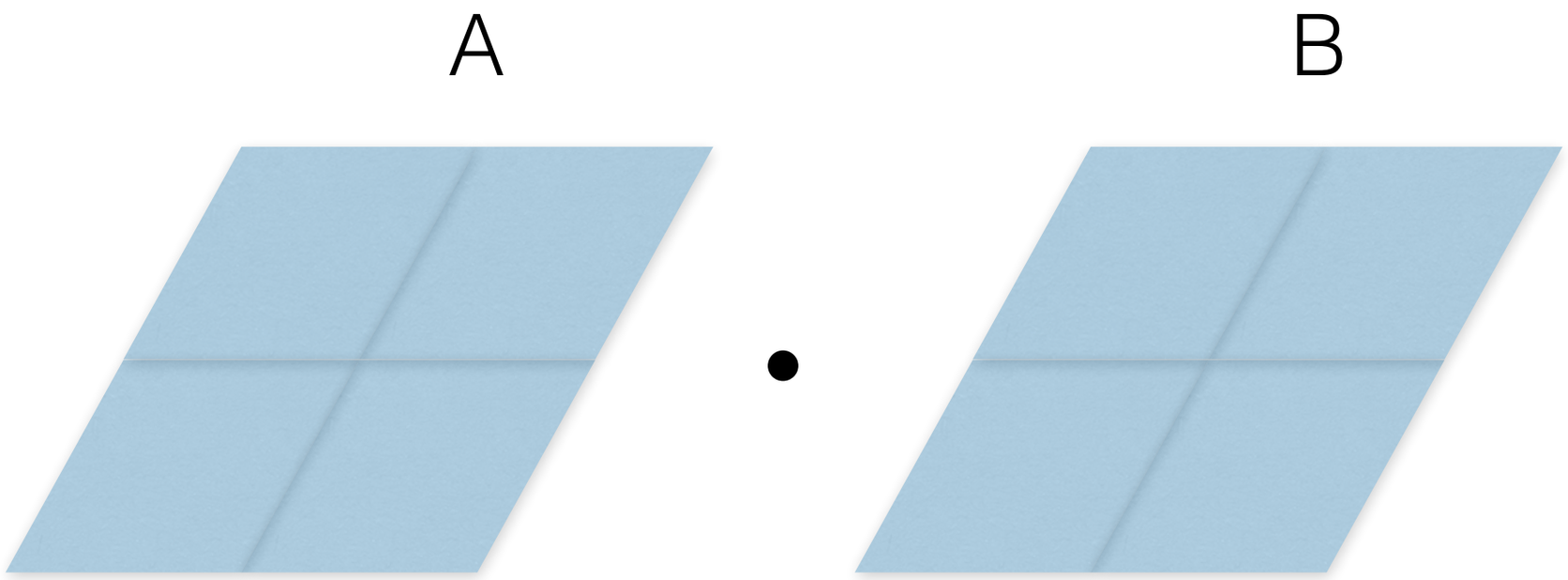
SAARLAND UNIVERSITY
GRADUATE SCHOOL OF COMPUTER SCIENCE

Evaluation
Random Memory Writes

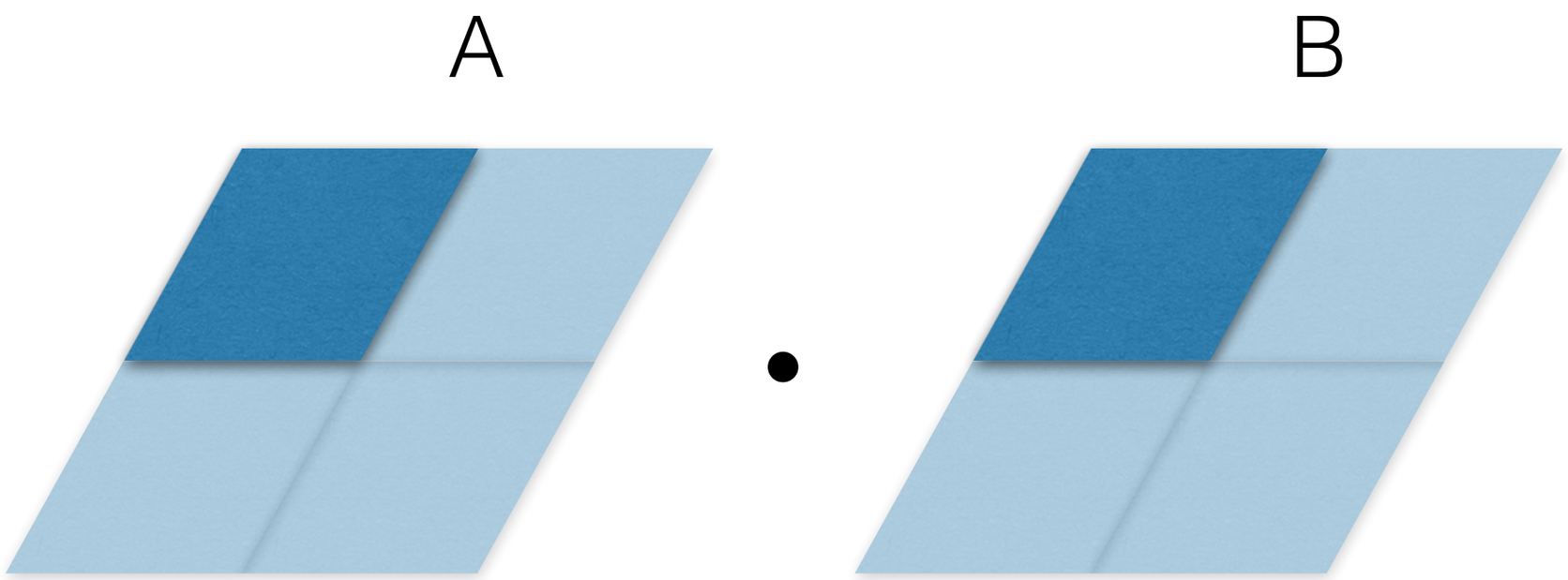




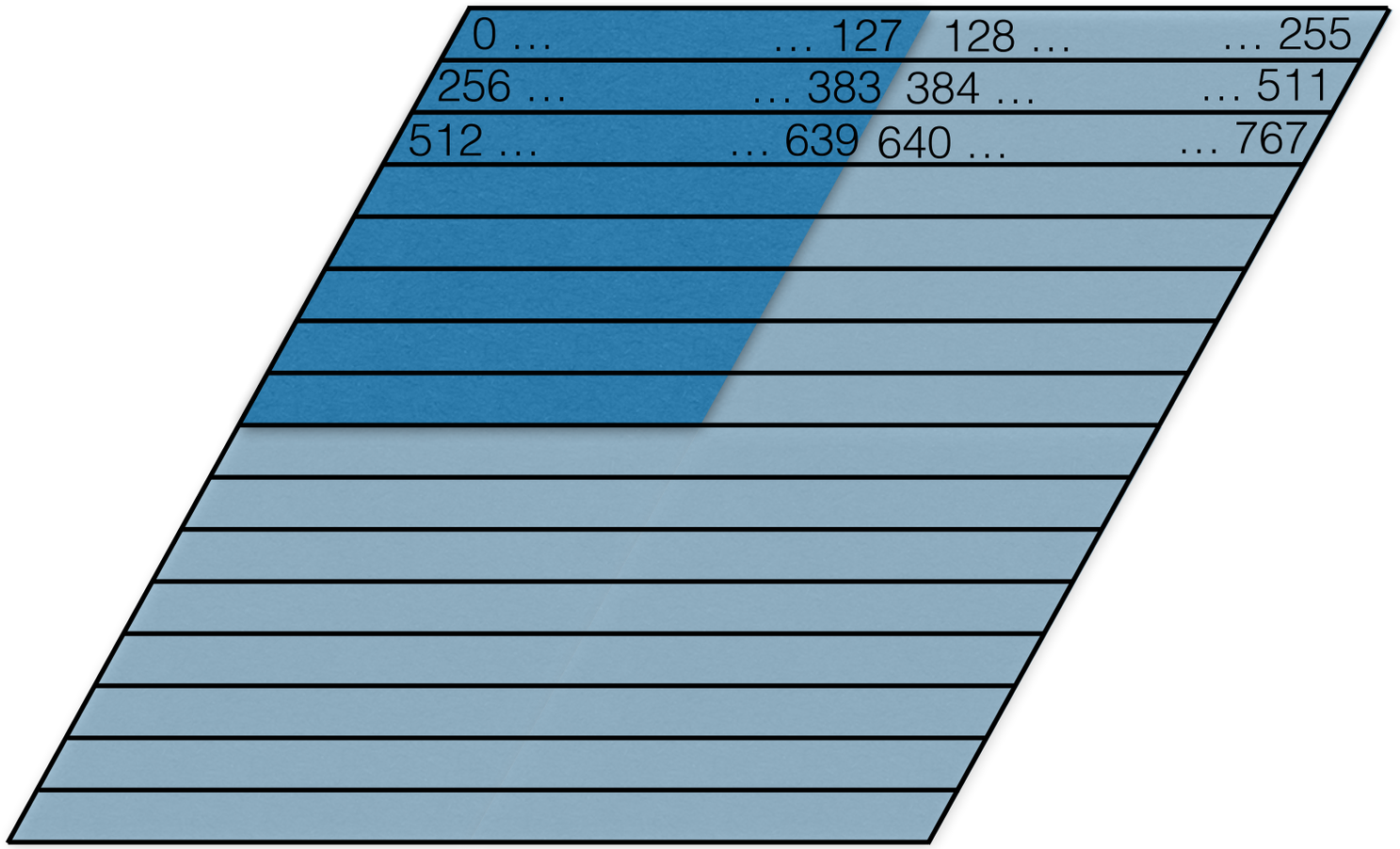
Evaluation
Strassen Matrix Multiplication



Evaluation
Strassen Matrix Multiplication

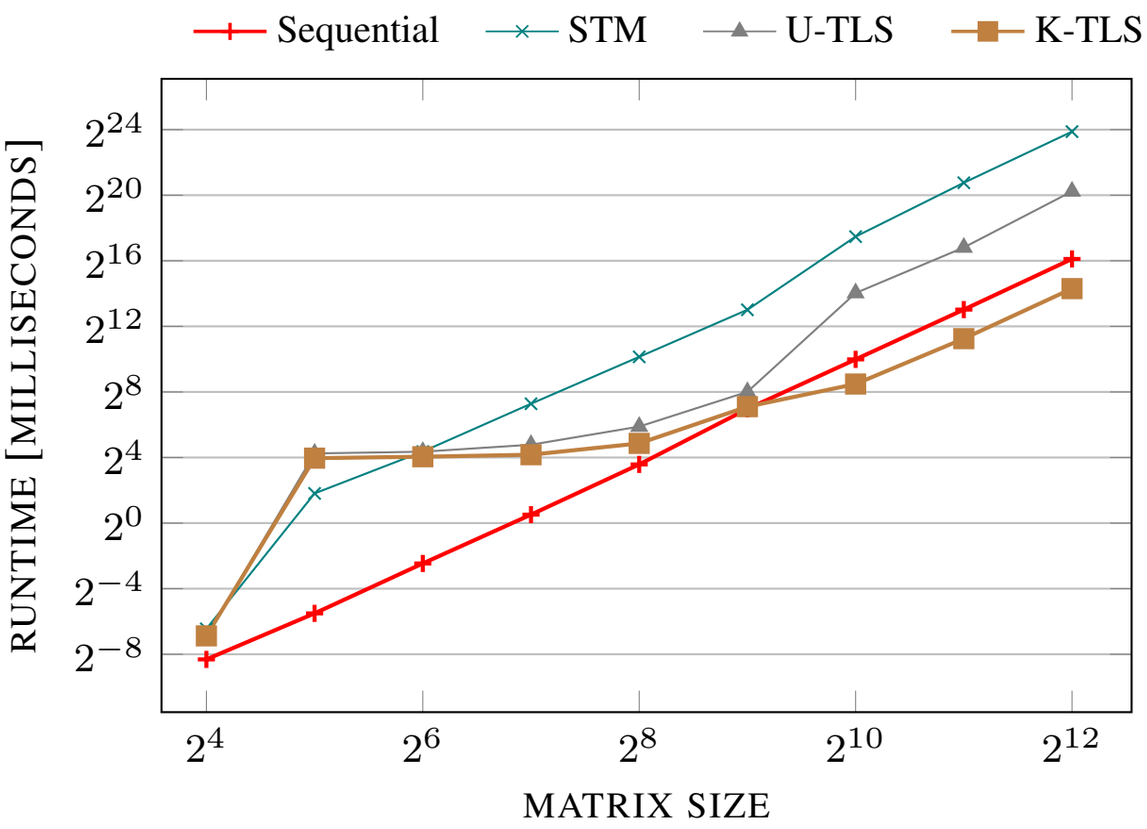


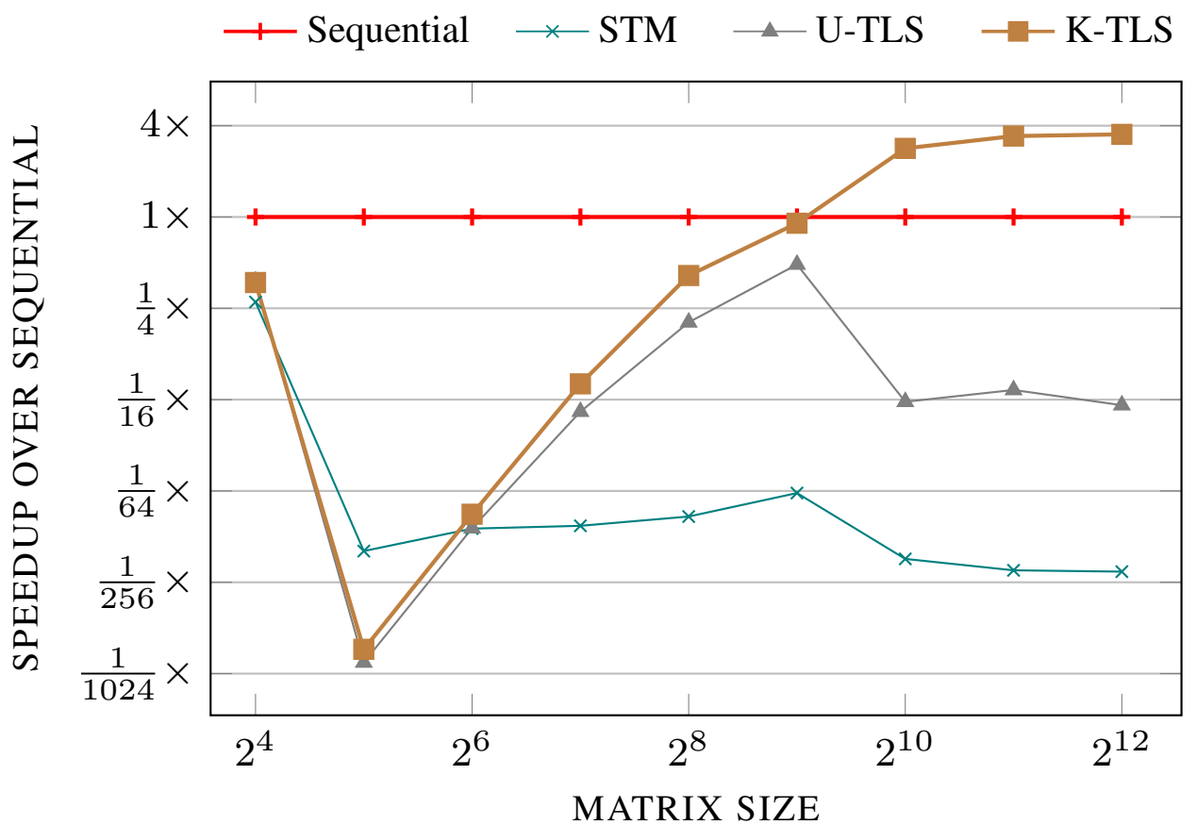
Evaluation
Strassen Matrix Multiplication



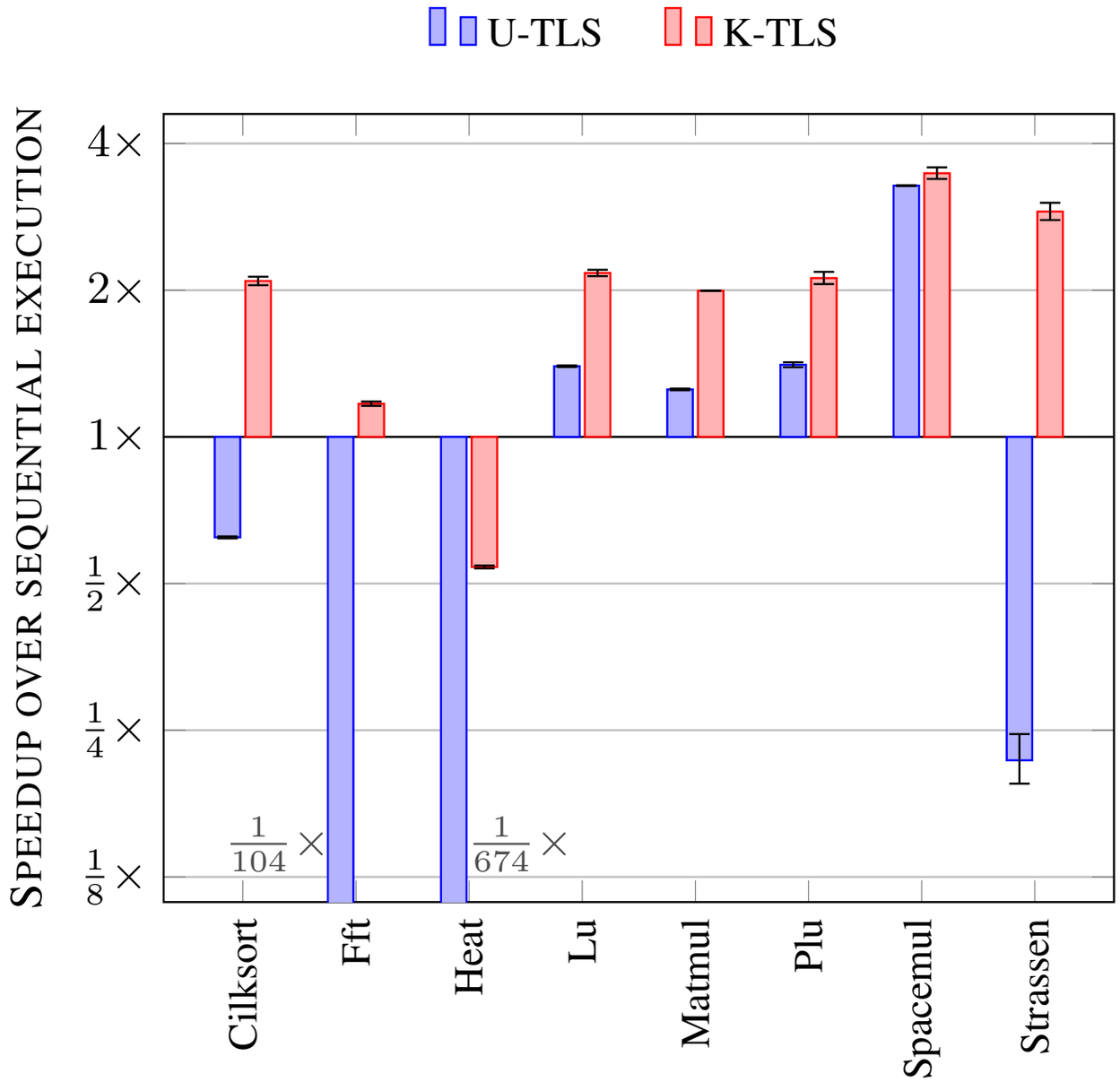
SAARLAND UNIVERSITY
GRADUATE SCHOOL OF COMPUTER SCIENCE

Evaluation
Strassen Matrix Multiplication





Evaluation
Cilk Suite



SAARLAND UNIVERSITY
GRADUATE SCHOOL OF COMPUTER SCIENCE

Summary

- K-TLS: Thread-Level Speculation with Kernel Support
  - superior to software-only approaches like STM
  - superior to virtual-memory based method in user space
  - full isolation even for system calls
  - no instrumentation → easy to integrate
  - open source: https://github.com/hammacher/k-tls