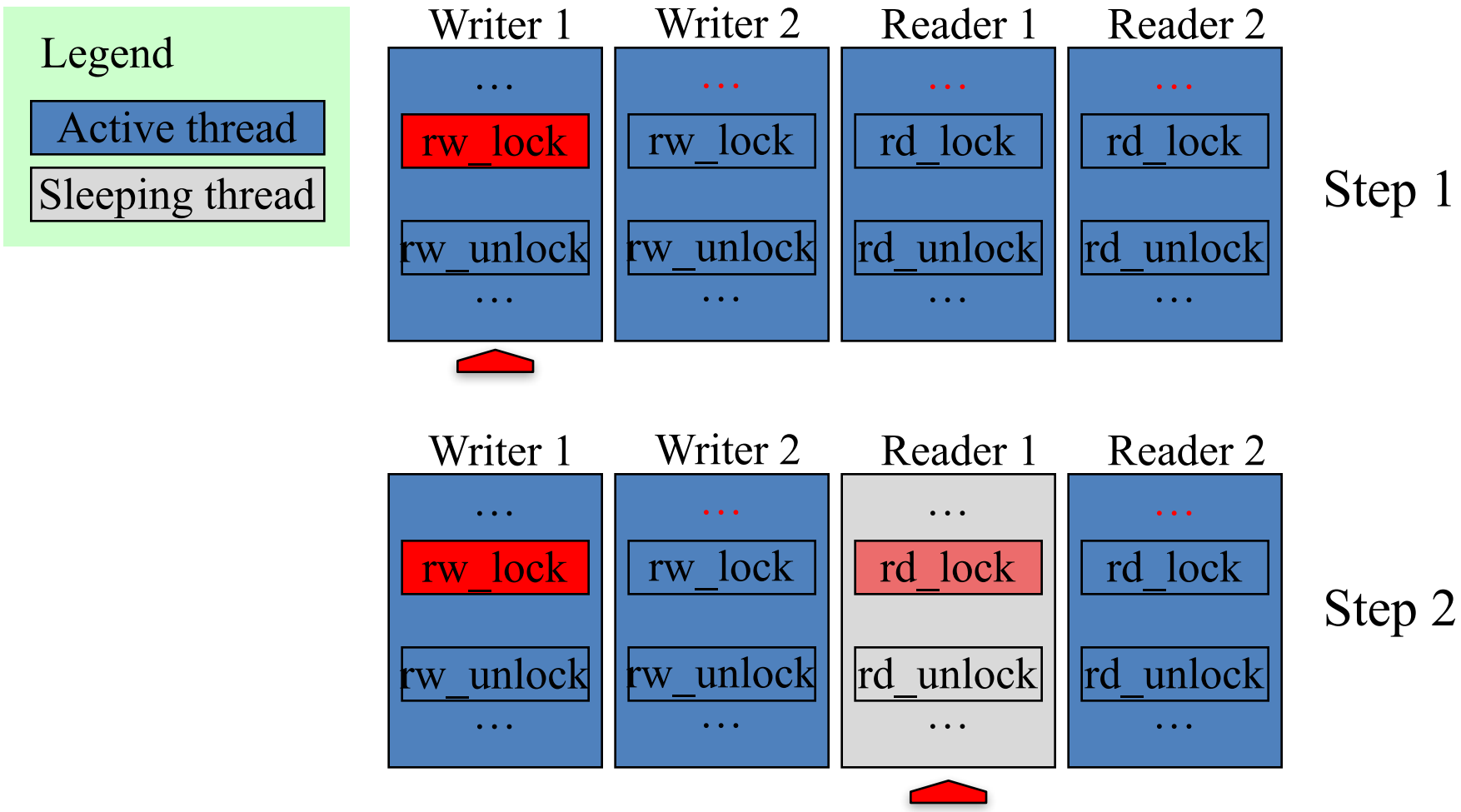
POSIX Threads: a first step toward parallel programming

George Bosilca
bosilca@icl.utk.edu

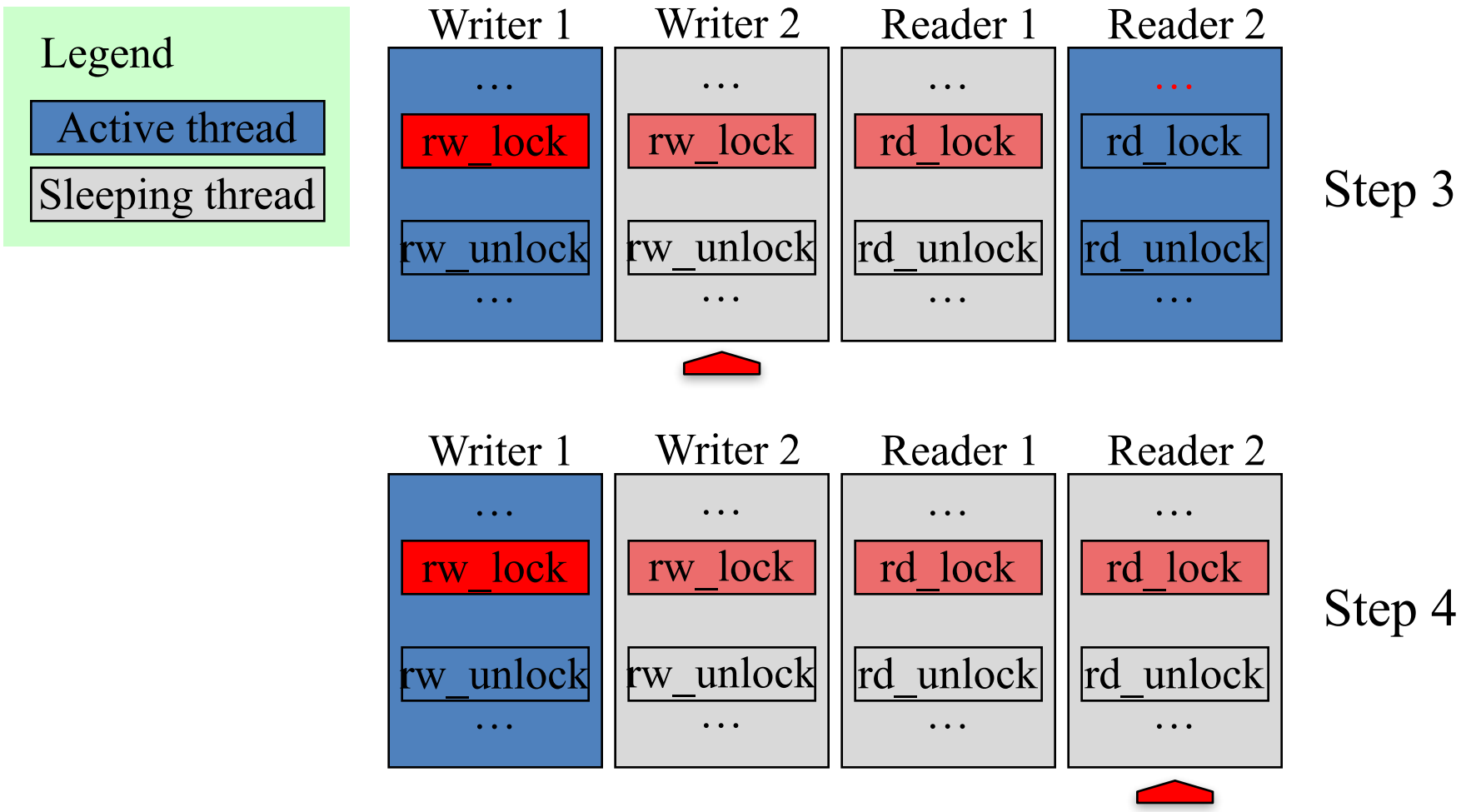
Shared/Exclusive Locks

- **R**ead**W**rite **Mut**ual **ex**clusion
- Extension used by the reader/writer model
- 4 states: write_lock, write_unlock, read_lock and read_unlock.
- multiple threads may hold a shared lock simultaneously, but only one thread may hold an exclusive lock.
- if one thread holds an exclusive lock, no threads may hold a shared lock.

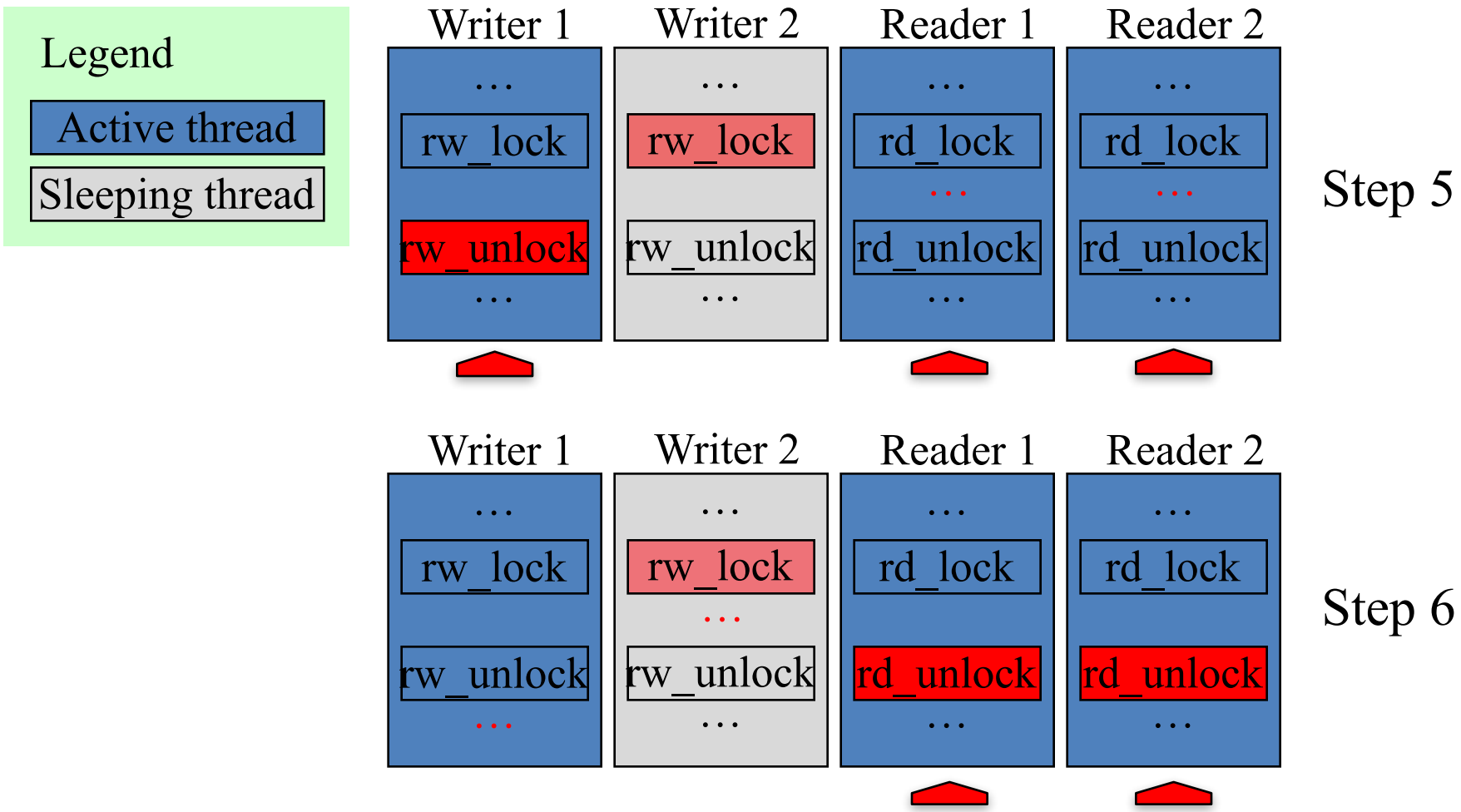
Shared/Exclusive Locks



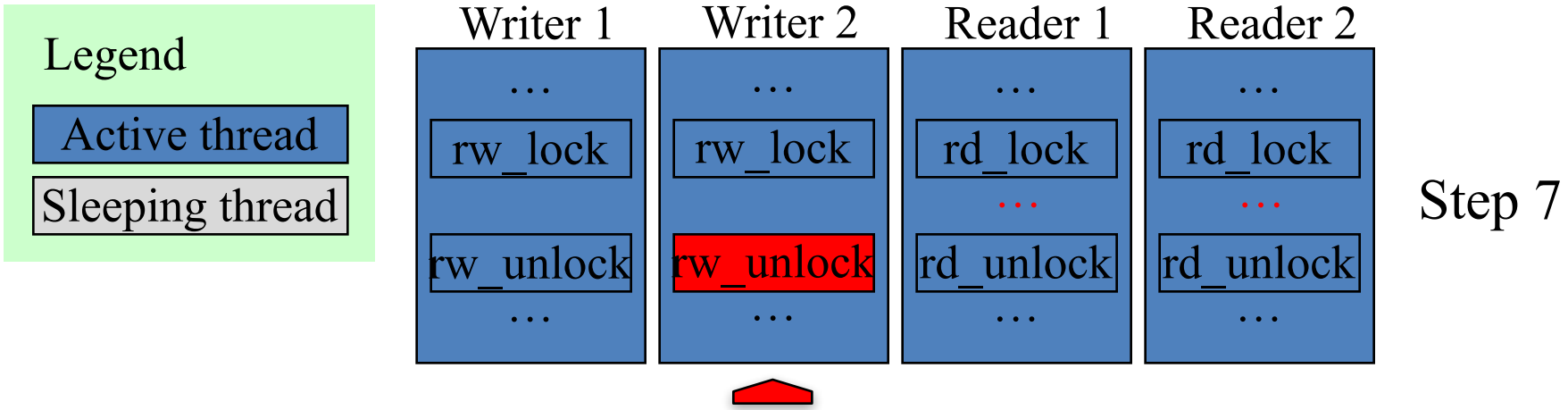
Shared/Exclusive Locks



Shared/Exclusive Locks

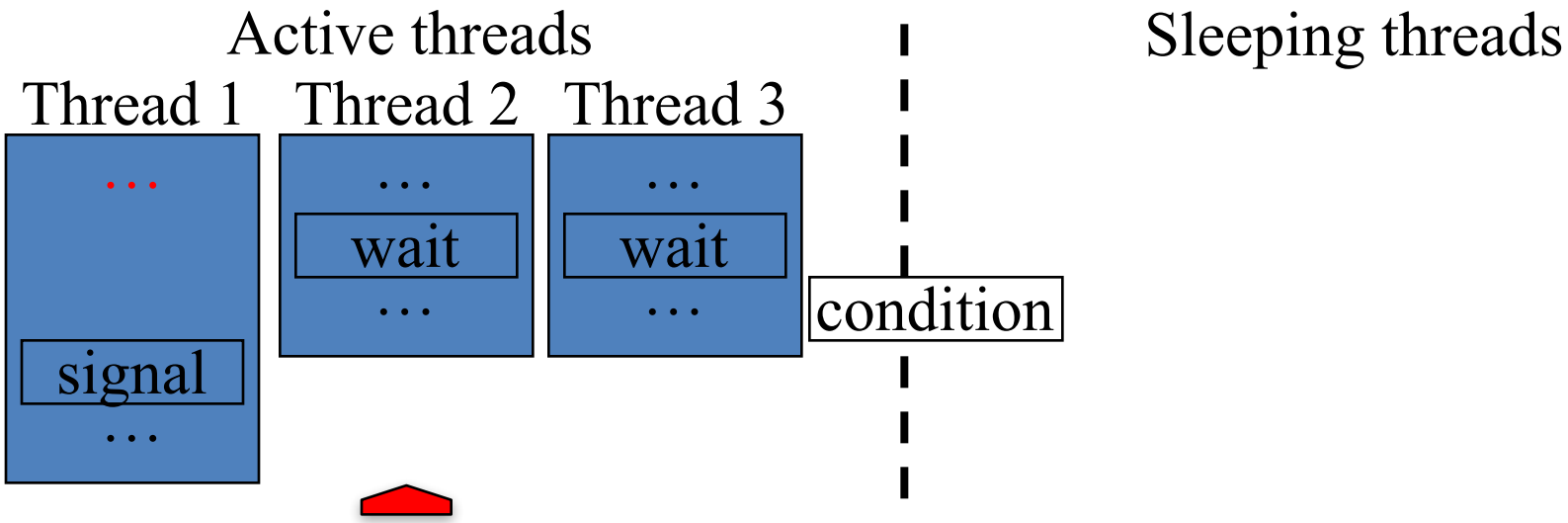


Shared/Exclusive Locks



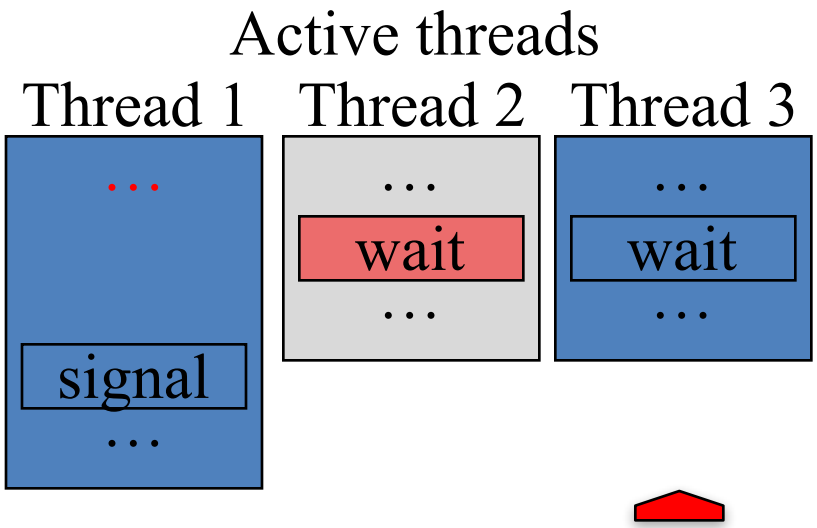
Condition Variable

- Block a thread while waiting for a condition
- Condition_wait / condition_signal
- Several thread can wait for the same condition, they all get the signal



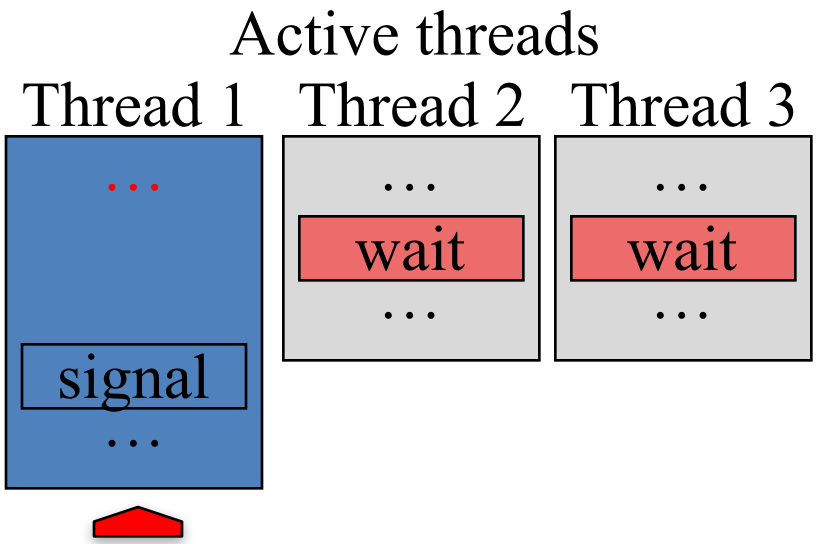
Condition Variable

- Block a thread while waiting for a condition
- Condition_wait / condition_signal
- Several thread can wait for the same condition, they all get the signal



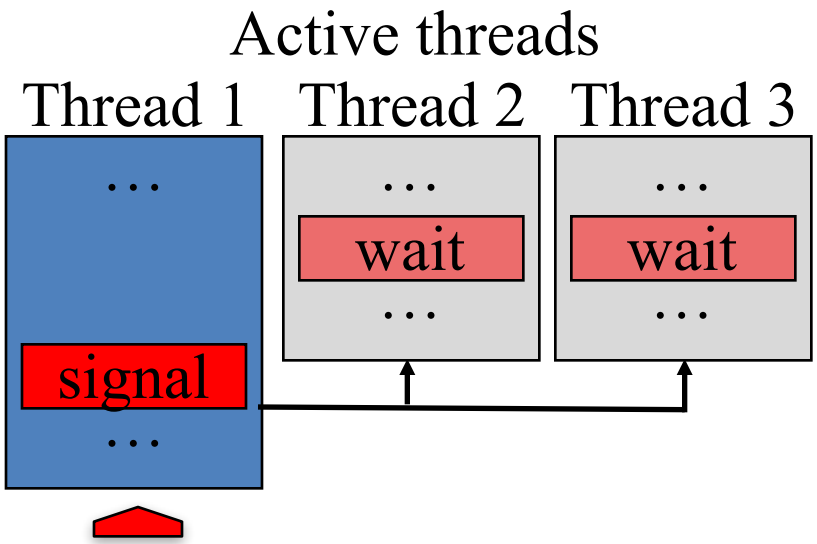
Condition Variable

- Block a thread while waiting for a condition
- Condition_wait / condition_signal
- Several thread can wait for the same condition, they all get the signal



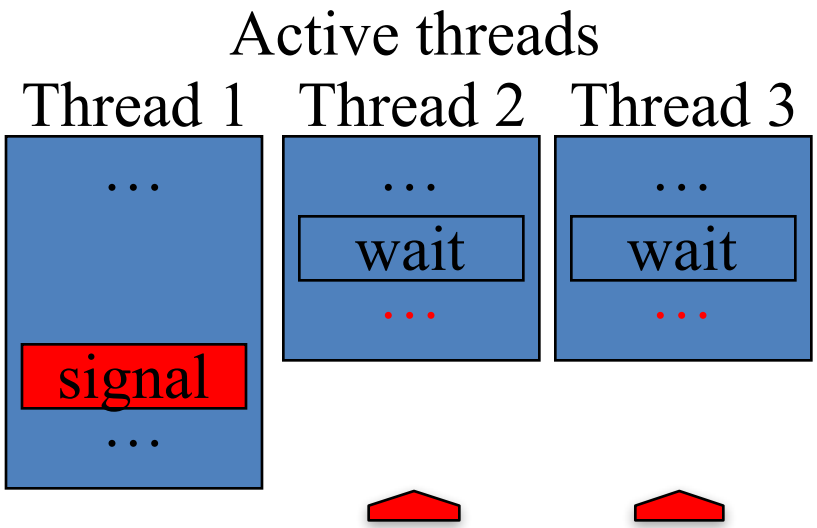
Condition Variable

- Block a thread while waiting for a condition
- Condition_wait / condition_signal
- Several thread can wait for the same condition, they all get the signal



Condition Variable

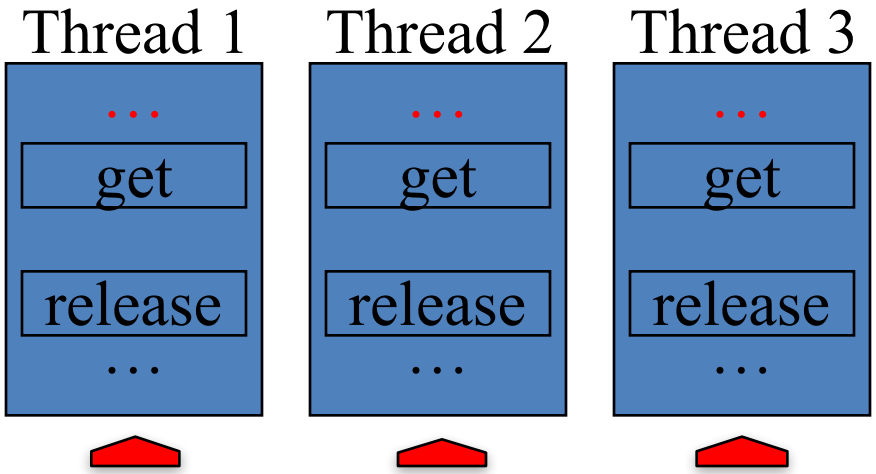
- Block a thread while waiting for a condition
- Condition_wait / condition_signal
- Several thread can wait for the same condition, they all get the signal



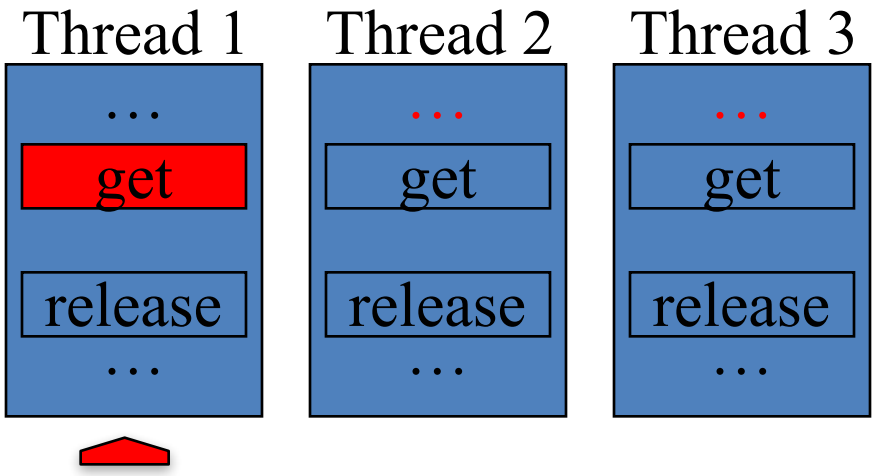
Semaphores

- simple counting mutexes
- The semaphore can be hold by as many threads as the initial value of the semaphore.
- When a thread get the semaphore it decrease the internal value by 1.
- When a thread release the semaphore it increase the internal value by 1.

Semaphores

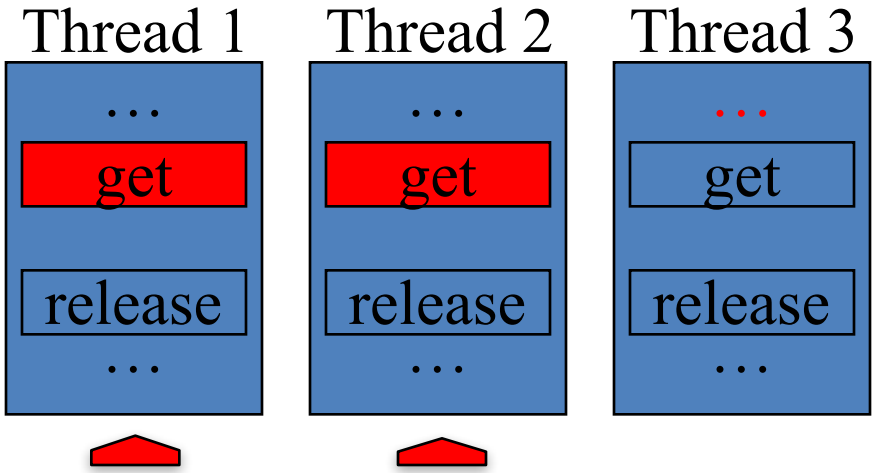


Semaphore (2)

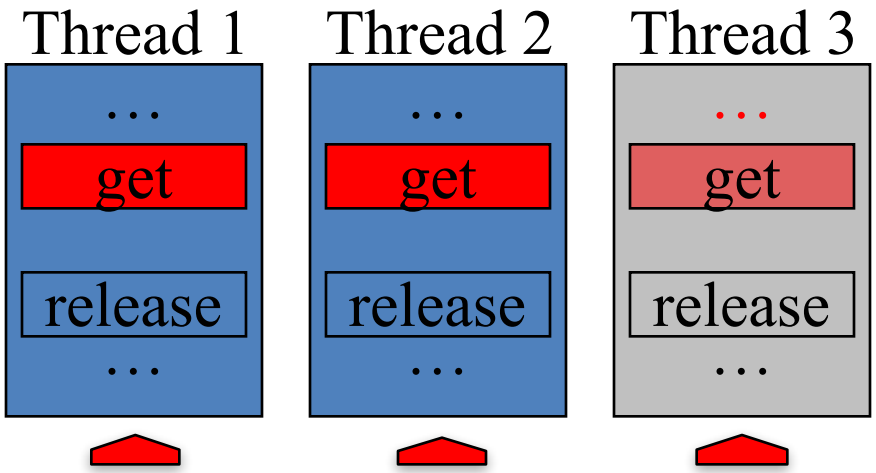


Semaphore (1)

Semaphores

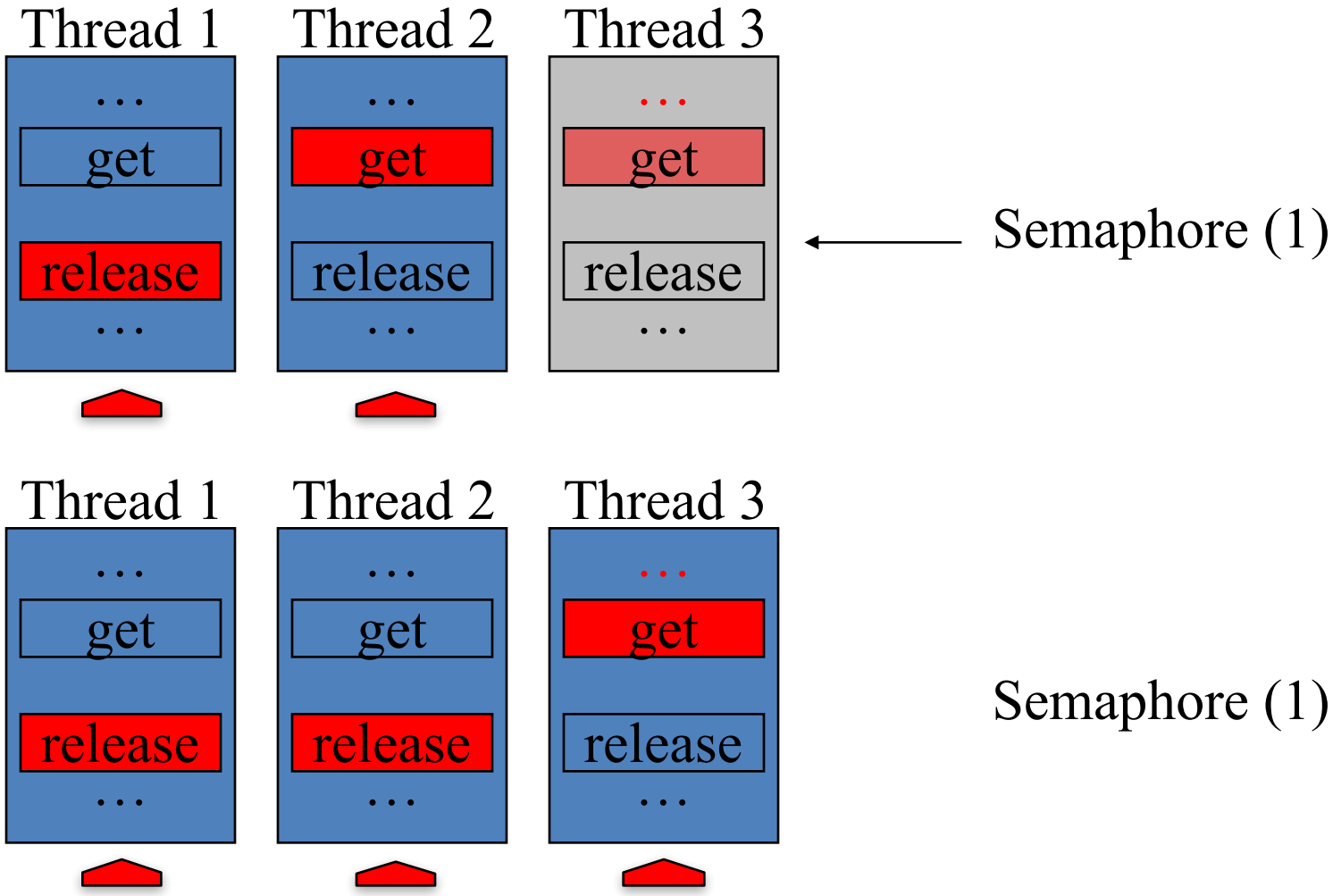


Semaphore (0)

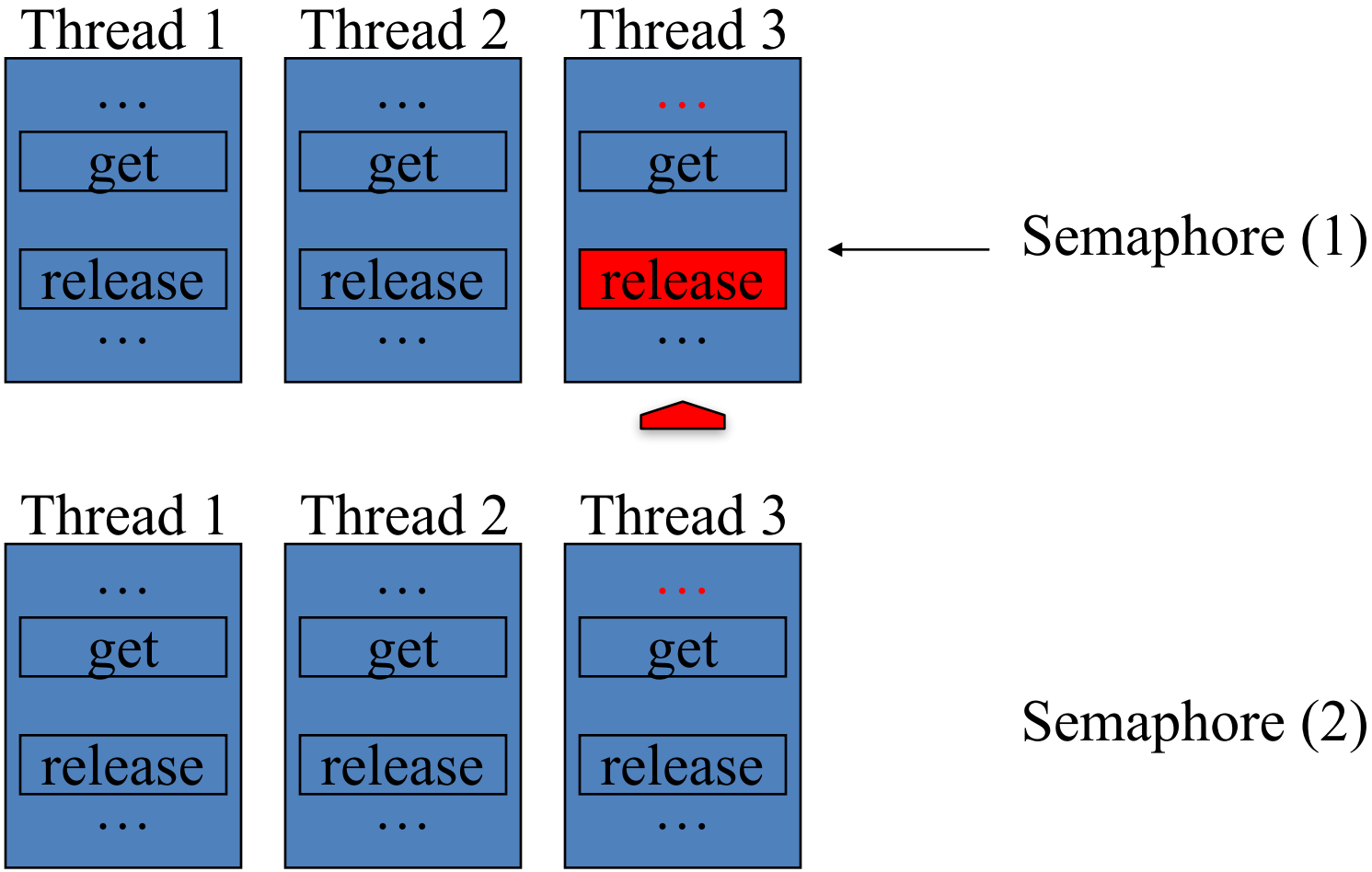


Semaphore (0)

Semaphores


Thread 1
Thread 2
Thread 3
…
get
release
…
Semaphore (1)
Thread 1
Thread 2
Thread 3
…
get
release
…
Semaphore (1)

Semaphores


Thread 1
Thread 2
Thread 3
…
get
release
…
Semaphore (1)
Thread 1
Thread 2
Thread 3
…
get
release
…
Semaphore (2)

Atomic instruction

- Is any operation that a CPU can perform such that all results will be made visible to each CPU at the same time and whose operation is safe from interference by other CPUs
  - TestAndSet
  - CompareAndSwap
  - DoubleCompareAndSwap
  - Atomic increment
  - Atomic decrement

Pthread API

Prefix	Use
pthread_	Thread management (create/destroy/cancel/join/exit)
pthread_attr_	Thread attributes
pthread_mutex_	Mutexes
pthread_mutexattr_	Mutexes attributes
pthread_cond_	Condition variables
pthread_condattr_	Condition attributes
pthread_key_	Thread-specific data key (TLS)
pthread_rwlock_	Read/write locks
pthread_barrier_	Synchronization barriers

Thread Management (create)

int pthread_create (pthread_t *restrict thread,	**[OUT] thread id**
const pthread_attr_t *restrict attr,	**[IN] attributes**
void *(*start_routine)(void *),	**[IN] thread function**
void *restrict arg**)**	**[IN] argument for thread function**
int pthread_exit (void *value_ptr**)**	**[OUT] Return to caller/joiner**
int pthread_cancel (pthread_t thread**)**	**[IN] thread to be cancelled**
int pthread_attr_init (pthread_attr_t *attr**)**	**[OUT] attributes to be initialized**
int pthread_attr_set*(pthread_attr_t *restrict attr, ***)**	**[IN] set attributed (state, stack)**
int pthread_attr_destroy (pthread_attr_t *attr**)**	**[IN] attributed to be destroyed**

Attributes: Detached or joinable state, Scheduling inheritance, Scheduling policy, Scheduling parameters, Scheduling contention scope, Stack size, Stack address, Stack guard (overflow) size

Questions:

- Once created what will be the status of the thread and how it will be scheduled by the OS ? (use sched_setscheduler)
- Where it will be run ? (use sched_setaffinity or HWLOC)