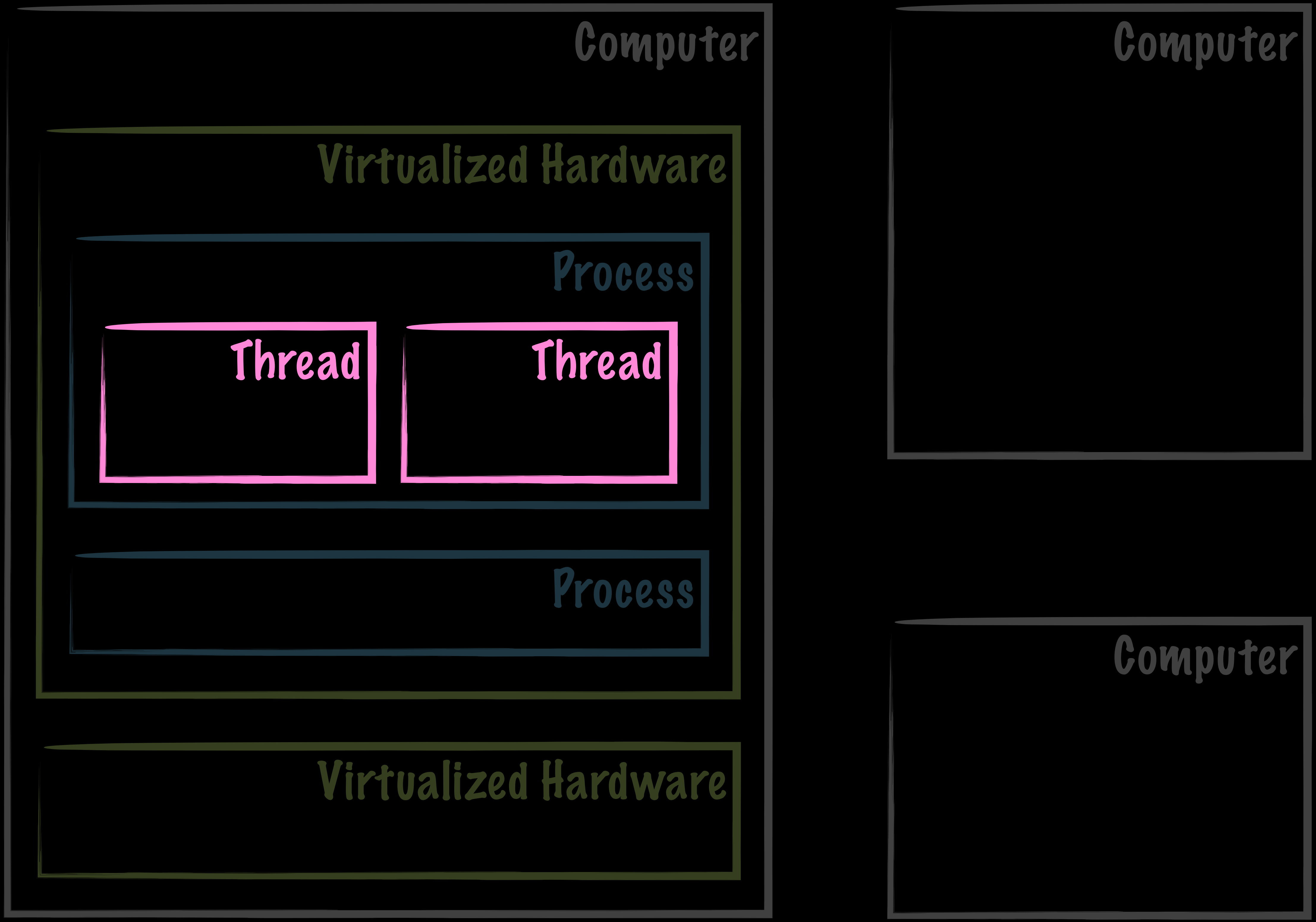
Programming with Threads

Benefits and Risks

A summary of the first two chapters of
Göetz, Brian, et al. Java concurrency in practice. Addison-Wesley, 2006.

How can we perform several computations concurrently?


Computer
Virtualized Hardware
Process
Thread
Thread
Process
Virtualized Hardware
Computer
Computer

Benefits

- Model concurrent tasks
  - Do this but also that
- Exploit multiple processors
  - Shorten execution time
- Handle asynchronous events
  - When waiting, do something else

Risks

- Thread-safety
  - Nothing bad should happen
- Liveness
  - Something good should happen
- Performance
  - It should happen quickly

Risks

Thread-safety – Description

”A class is thread-safe if it behaves correctly when accessed from multiple threads, […] with no additional synchronization […] on the part of the calling code.”

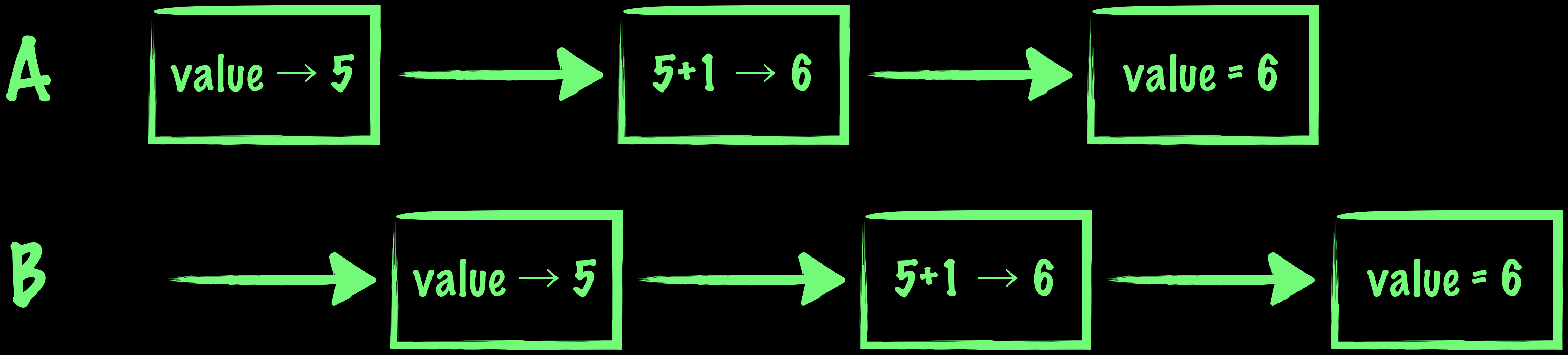
Risks

Thread-safety – Example

```
public class UnsafeSequence {
    private int value;

    public int getNext() {
        return value++;
    }
}
```

load
add
store



Thread-safety – Solution

Three ways to fix safety

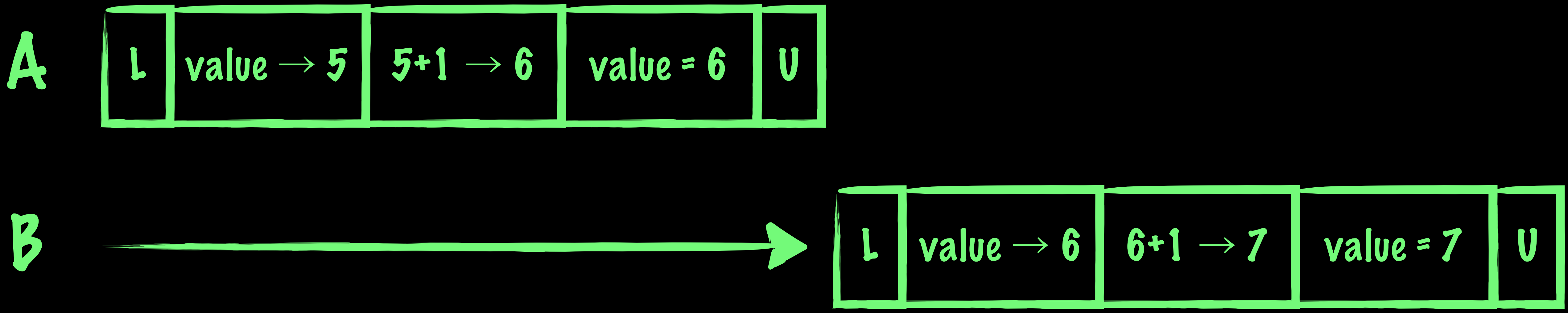
- Don't share state — need counter
- Make state immutable — need increment
- Use synchronization — okay

Risks

Thread-safety – Example

```
public class Sequence {
    private int value;

    public synchronized int getNext() {
        return value++;
    }
}
```



Risks

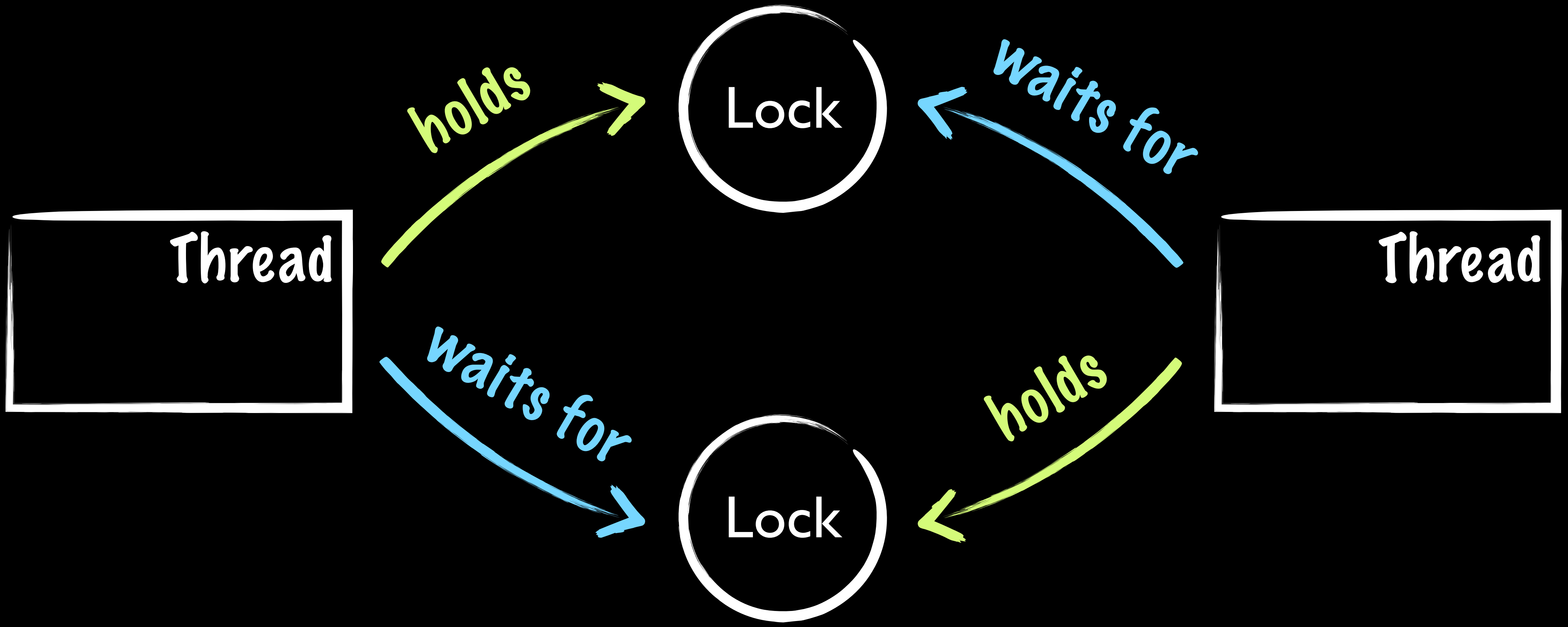
Liveness – Description

”A liveness failure occurs when an activity gets into a state such that it is permanently unable to make forward progress.”

Risks

Liveness – Example

Deadlock



Risks

Performance

Not only does synchronization make selected parts of the execution sequential, it also adds overhead when acquiring and releasing the locks.

Threads – Summary

Benefits

- Model concurrent tasks
- Exploit multiple processors
- Handle asynchronous events

Risks

- Safety
- Liveness
- Performance

This slide is intentionally left blank-ish.