

Background on Concurrency & Parallelism in Java (Part 1)

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt



Professor of Computer Science

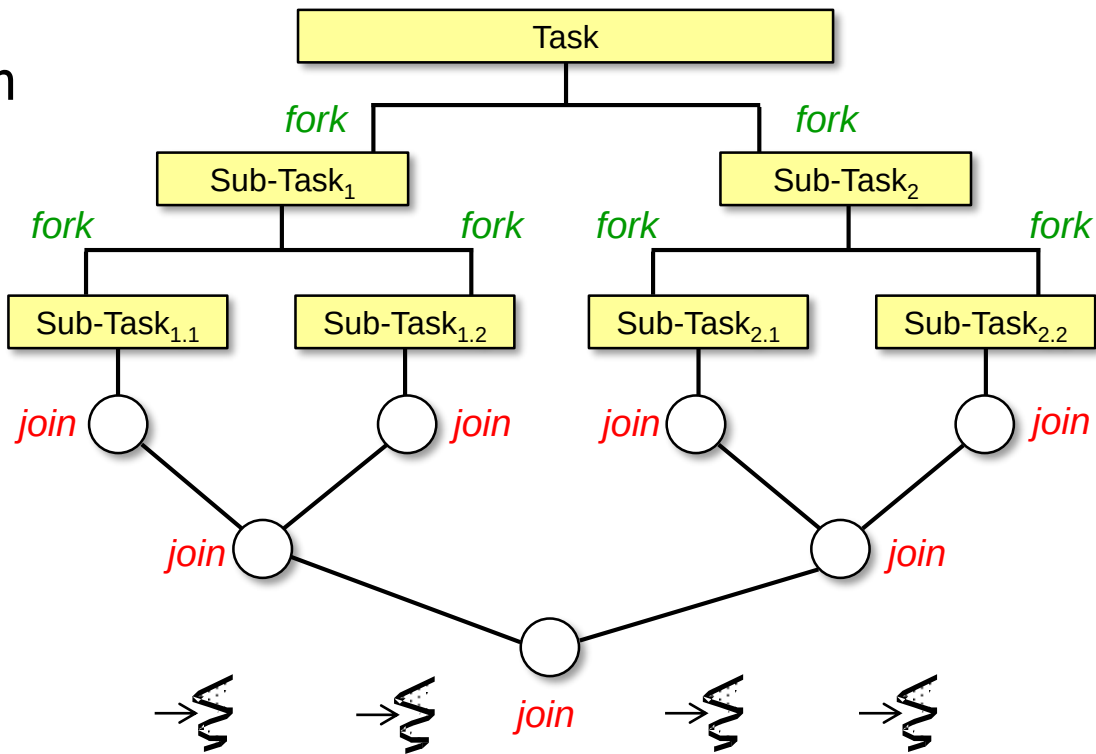
**Institute for Software
Integrated Systems**

**Vanderbilt University
Nashville, Tennessee, USA**



Learning Objectives in this Part of the Lesson

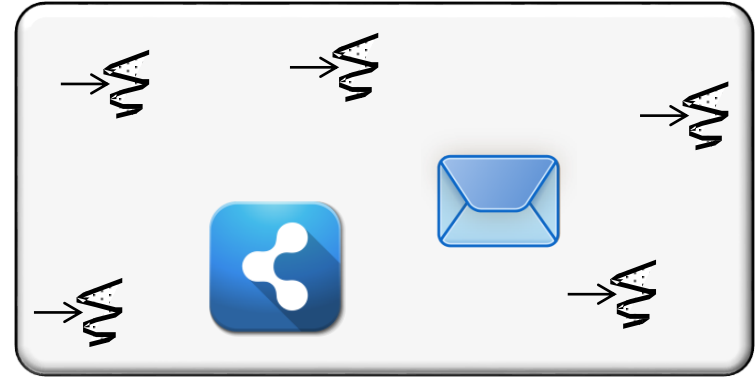
- Understand the meaning of the terms concurrency & parallelism



An Overview of Concurrency

An Overview of Concurrency

- Concurrency is a form of computing where threads can run simultaneously



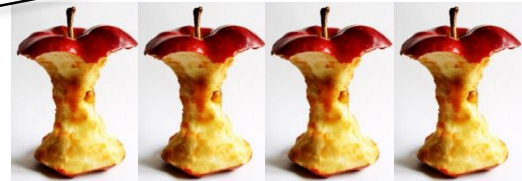
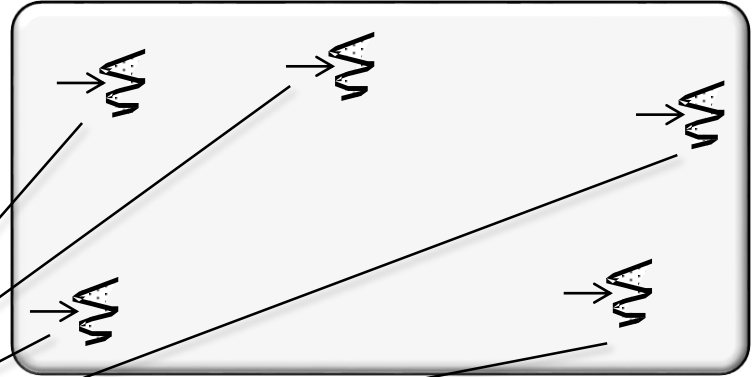
See [en.wikipedia.org/wiki/Concurrency \(computer science\)](https://en.wikipedia.org/wiki/Concurrency_(computer_science))

An Overview of Concurrency

- Concurrency is a form of computing where threads can run simultaneously

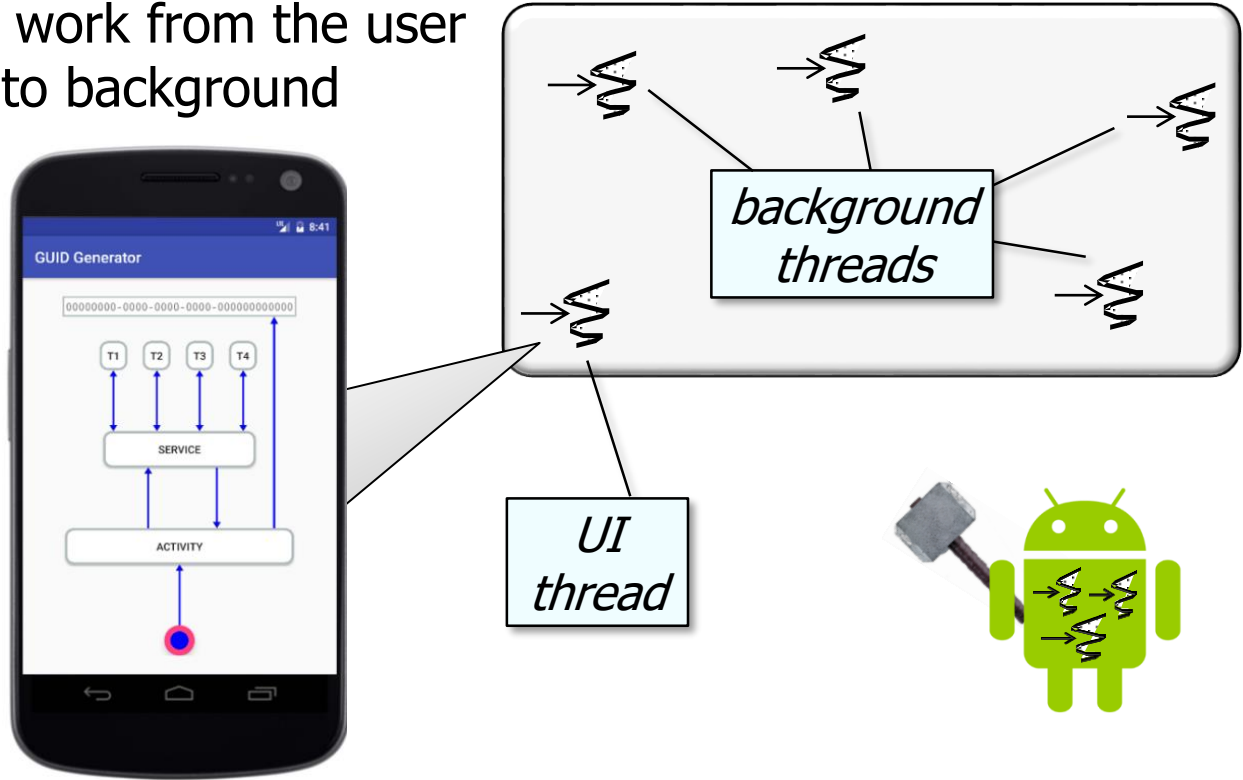
```
new Thread(() ->  
    someComputations());
```

*A Java threads are units of execution
for instruction streams that can run
concurrently on processor cores*



An Overview of Concurrency

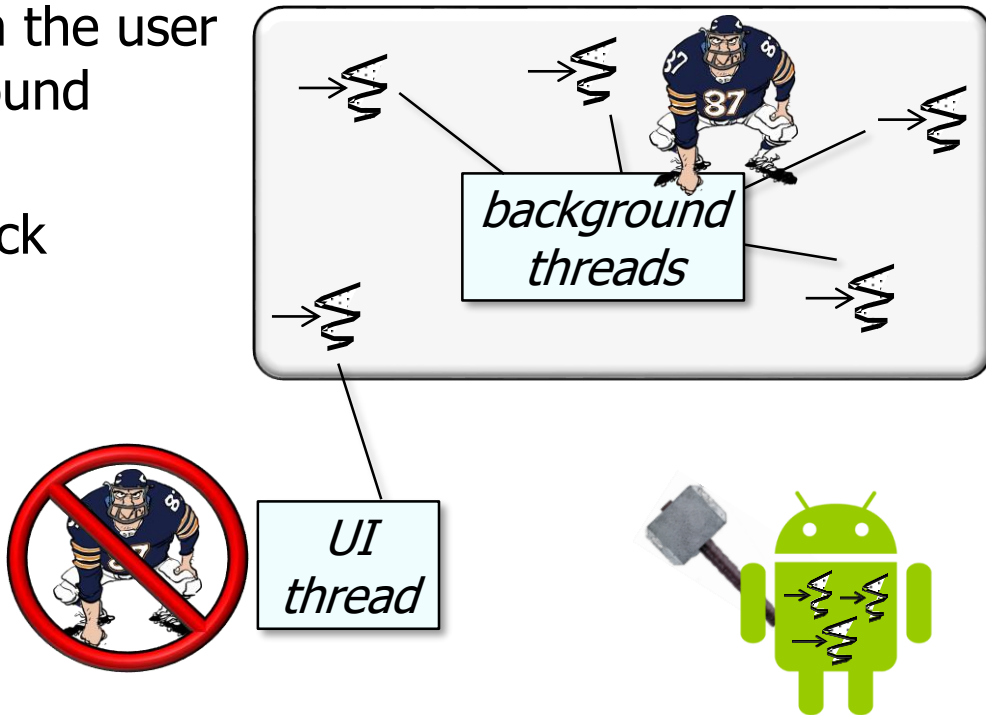
- Concurrency is a form of computing where threads can run simultaneously
- Often used to offload work from the user interface (UI) thread to background thread(s)



See developer.android.com/topic/performance/threads.html

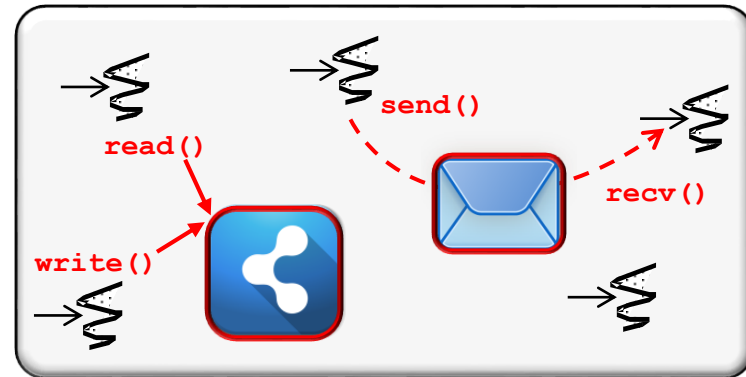
An Overview of Concurrency

- Concurrency is a form of computing where threads can run simultaneously
 - Often used to offload work from the user interface (UI) thread to background thread(s), e.g.
 - Background thread(s) can block
 - The UI thread does not block



An Overview of Concurrency

- Concurrent Java threads interact via shared objects and/or message passing

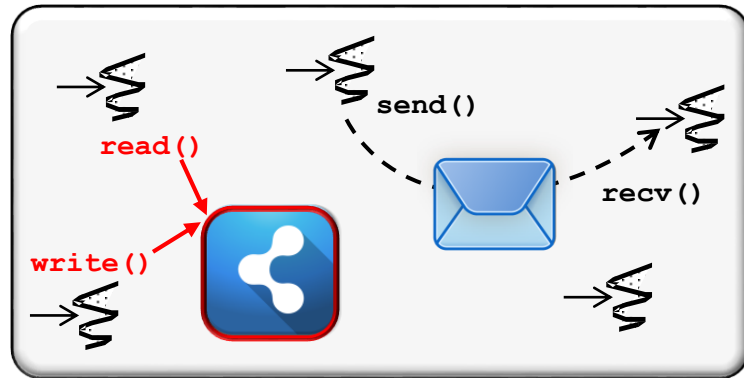
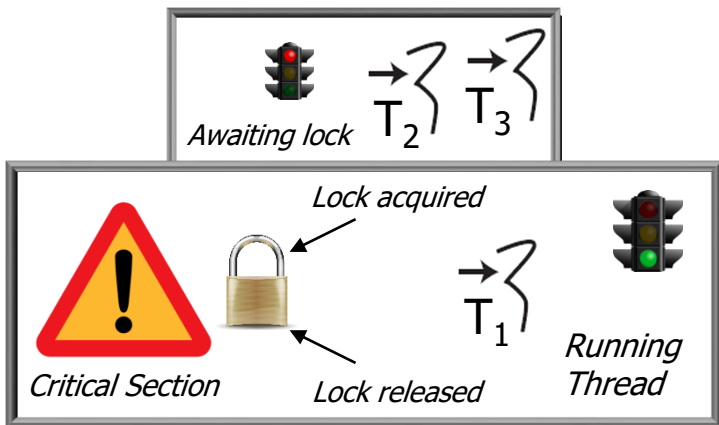


An Overview of Concurrency

- Concurrent Java threads interact via shared objects and/or message passing

- Shared objects**

- Synchronize concurrent operations on objects so object state remains coherent after each operation

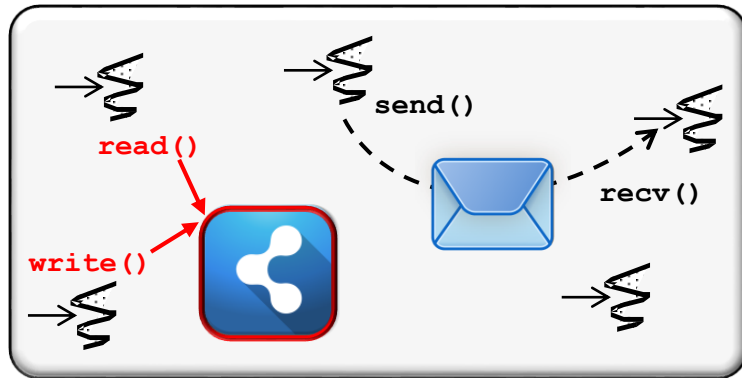


An Overview of Concurrency

- Concurrent Java threads interact via shared objects and/or message passing

- **Shared objects**

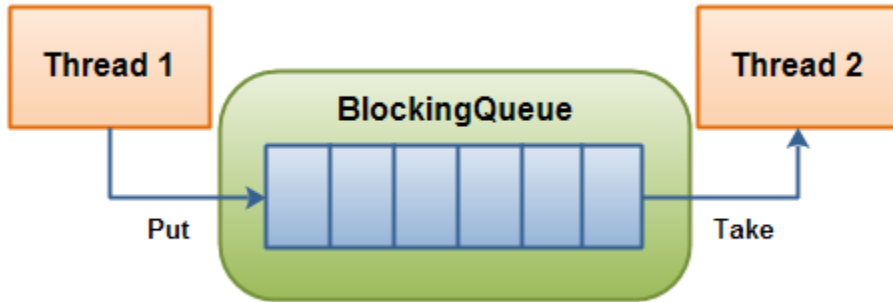
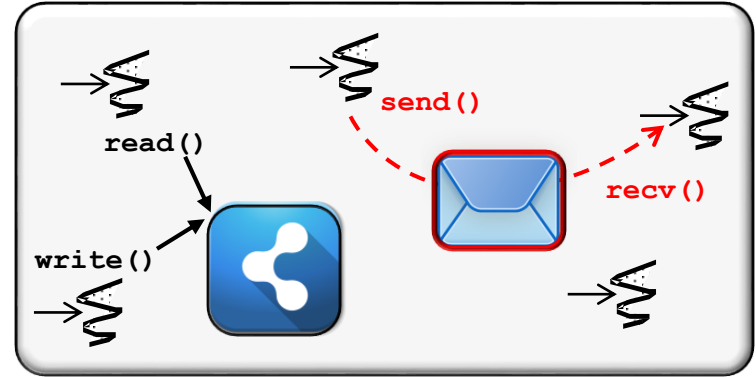
- Synchronize concurrent operations on objects so object state remains coherent after each operation
- Examples of Java synchronizers:
 - Synchronized statements/methods
 - Reentrant locks & intrinsic locks
 - Atomic operations
 - Semaphores & condition objects
 - “Compare-and-swap” (CAS) operations in `sun.misc.unsafe`



See dzone.com/articles/the-java-synchronizers

An Overview of Concurrency

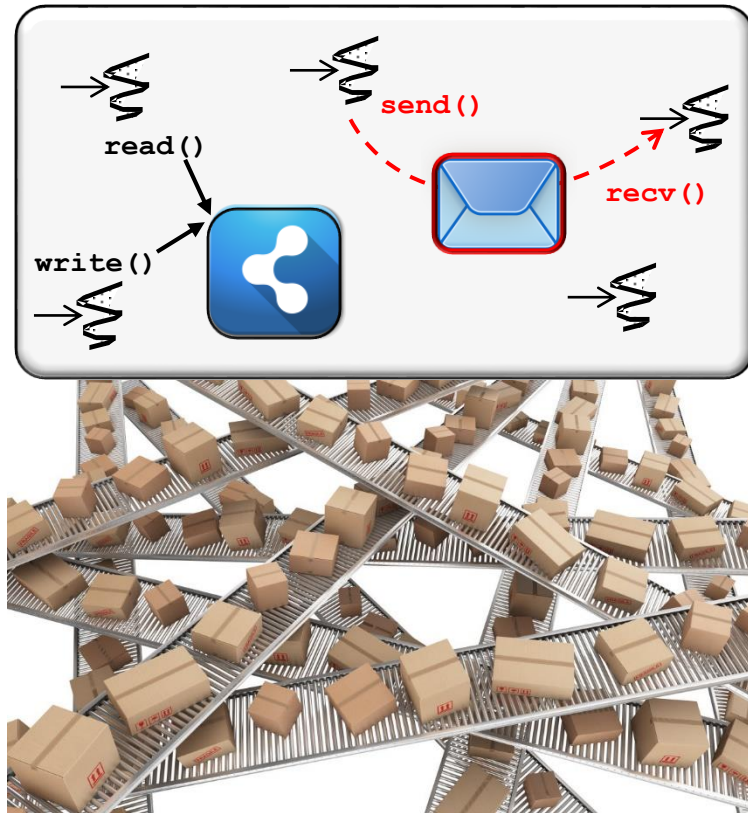
- Concurrent Java threads interact via shared objects and/or message passing
 - Shared objects
 - Message passing**
 - Send message(s) from producer thread(s) to consumer thread(s) via a thread-safe queue



See en.wikipedia.org/wiki/Message_passing

An Overview of Concurrency

- Concurrent Java threads interact via shared objects and/or message passing
 - **Shared objects**
 - **Message passing**
 - Send message(s) from producer thread(s) to consumer thread(s) via a thread-safe queue
 - Examples of Java thread-safe queues
 - Array & linked blocking queues
 - Priority blocking queue
 - Synchronous queue
 - Concurrent linked queue



See docs.oracle.com/javase/tutorial/collections/implementations/queue.html

An Overview of Concurrency

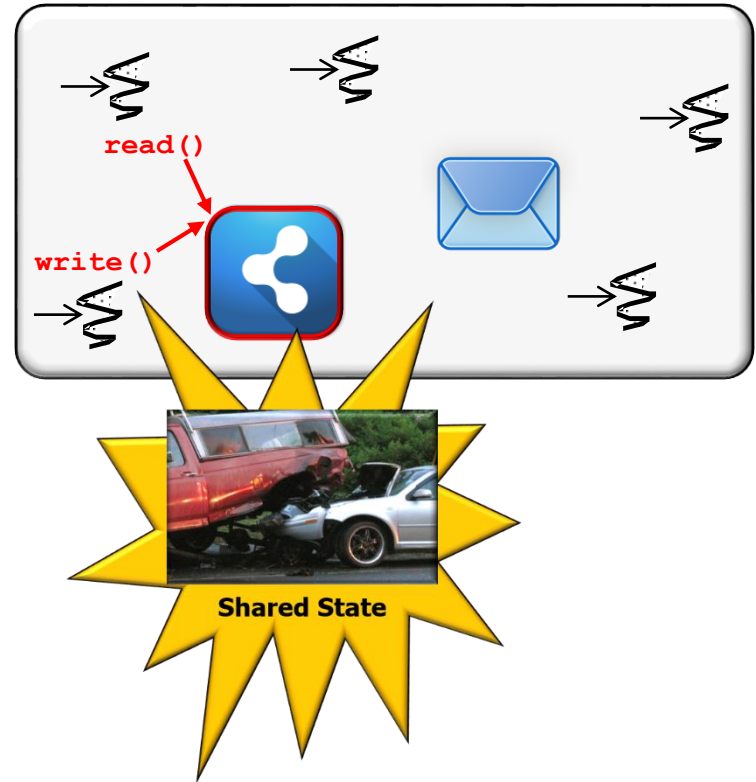
- Key goals of using shared objects and/or message passing are to share resources safely/efficiently & avoid hazards



See en.wikipedia.org/wiki/Thread_safety

An Overview of Concurrency

- Key goals of using shared objects and/or message passing are to share resources safely/efficiently & avoid hazards, e.g.
- Race conditions
 - Race conditions occur when a program depends upon the sequence or timing of threads for it to operate properly

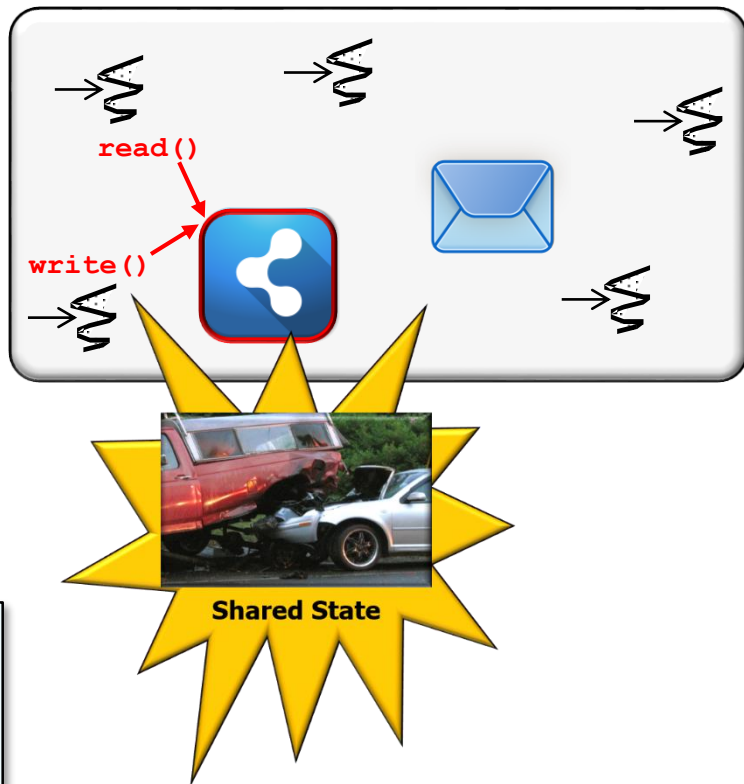


See en.wikipedia.org/wiki/Race_condition#Software

An Overview of Concurrency

- Key goals of using shared objects and/or message passing are to share resources safely/efficiently & avoid hazards, e.g.
- Race conditions
 - Race conditions occur when a program depends upon the sequence or timing of threads for it to operate properly

This test program induces race conditions due to lack of synchronization between producer & consumer threads accessing a bounded queue



An Overview of Concurrency

- Key goals of using shared objects and/or message passing are to share resources safely/efficiently & avoid hazards, e.g.
 - Race conditions
 - Memory inconsistencies
 - These errors occur when different threads have inconsistent views of what should be the same data



See jeremymanson.blogspot.com/2007/08/atomicity-visibility-and-ordering.html

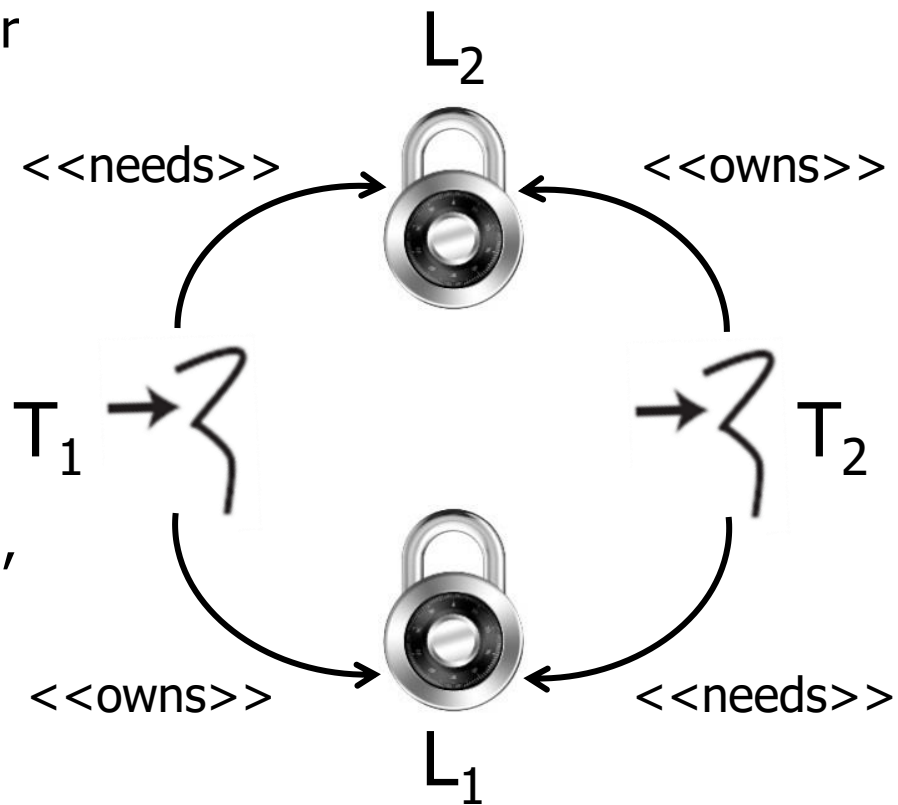
An Overview of Concurrency

- Key goals of using shared objects and/or message passing are to share resources safely/efficiently & avoid hazards, e.g.

- Race conditions
- Memory inconsistencies

- Deadlocks

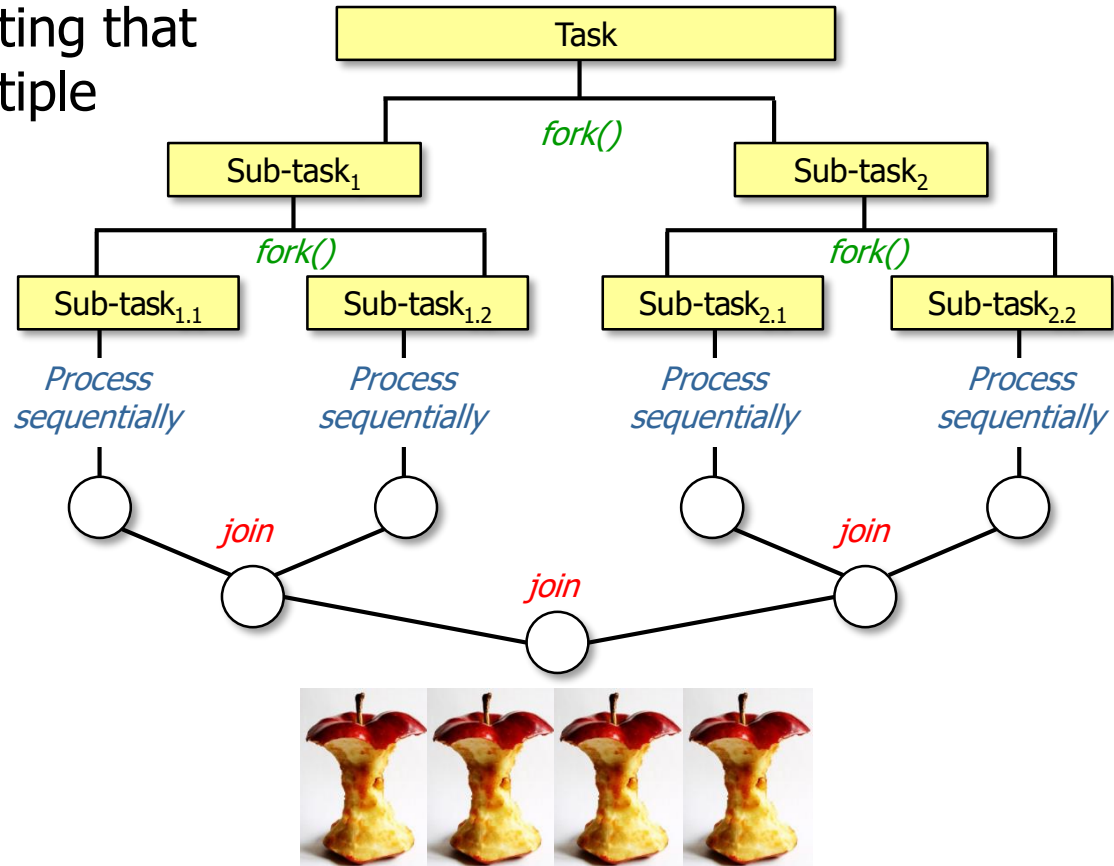
- Occur when 2+ competing threads are waiting for the other(s) to finish, & thus none ever do



An Overview of Parallelism

An Overview of Parallelism

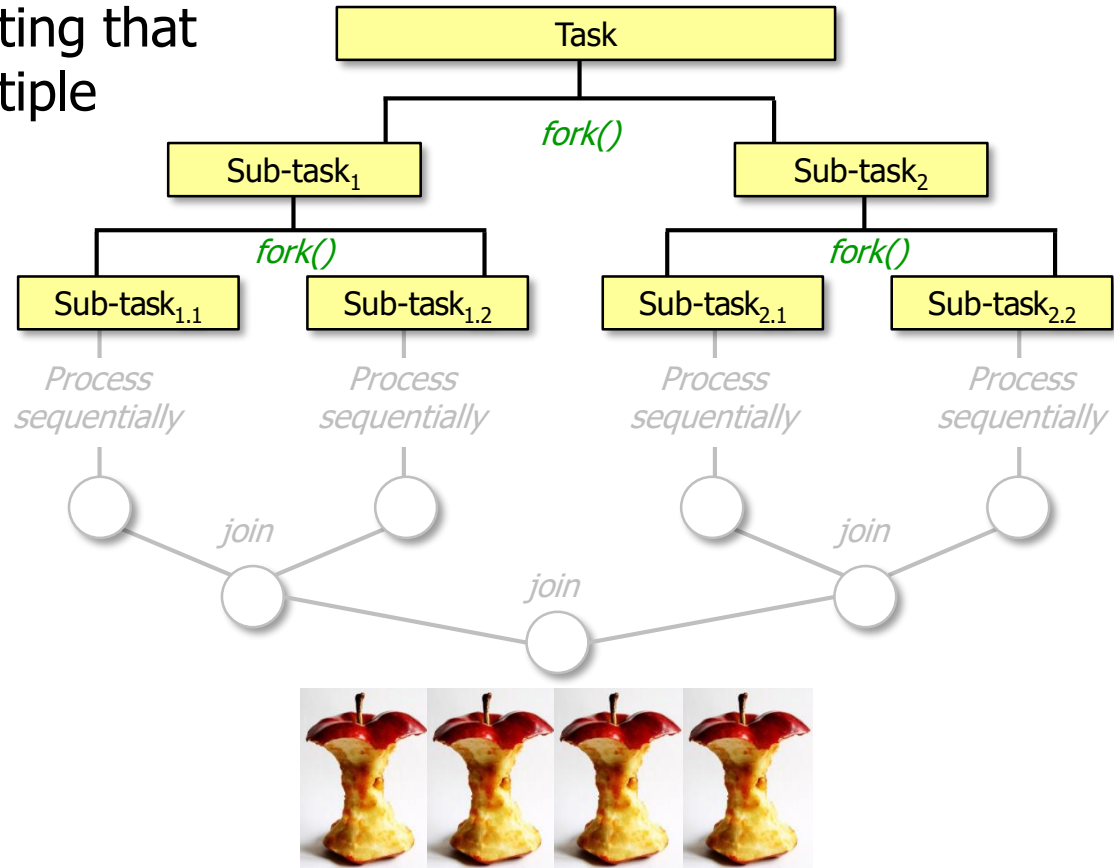
- Parallelism is a form of computing that performs several steps on multiple processor cores



See en.wikipedia.org/wiki/Parallel_computing

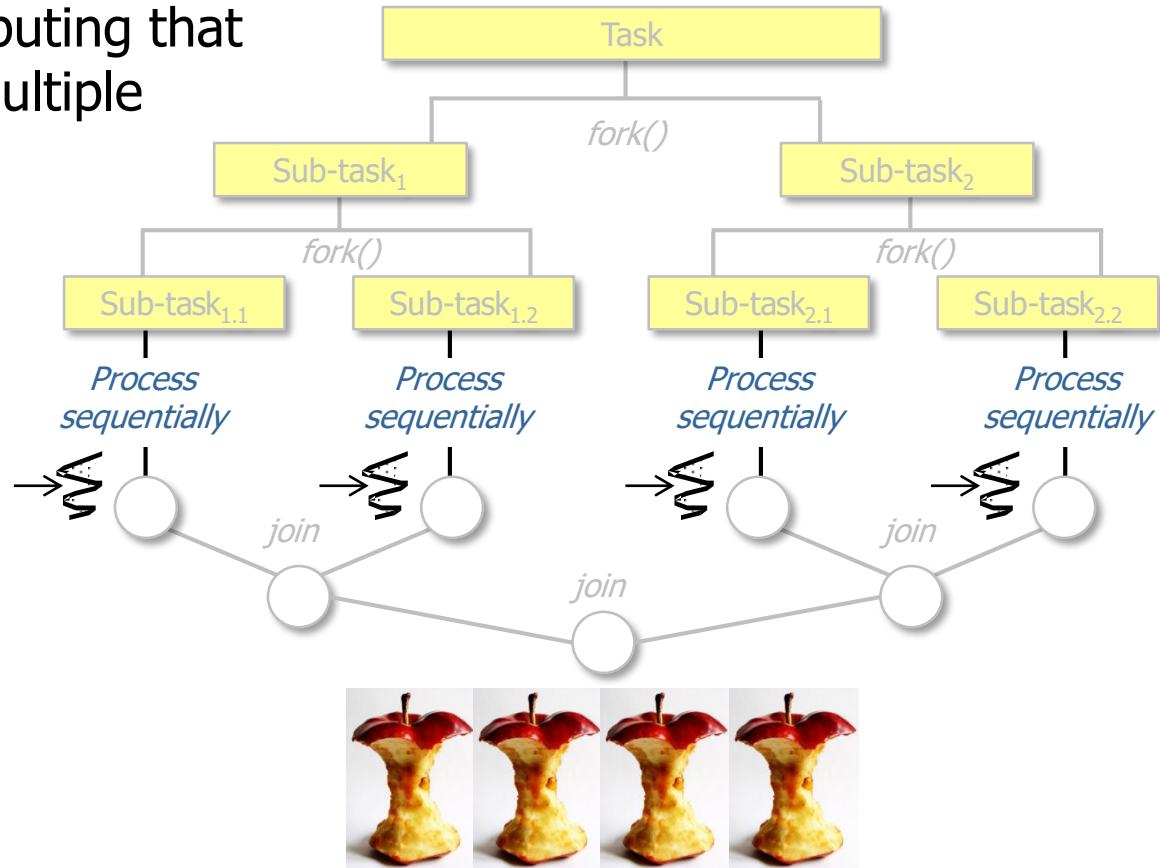
An Overview of Parallelism

- Parallelism is a form of computing that performs several steps on multiple processor cores, i.e.
 - Split – partition a task into sub-tasks



An Overview of Parallelism

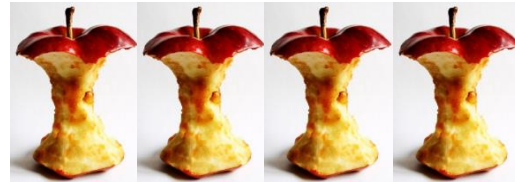
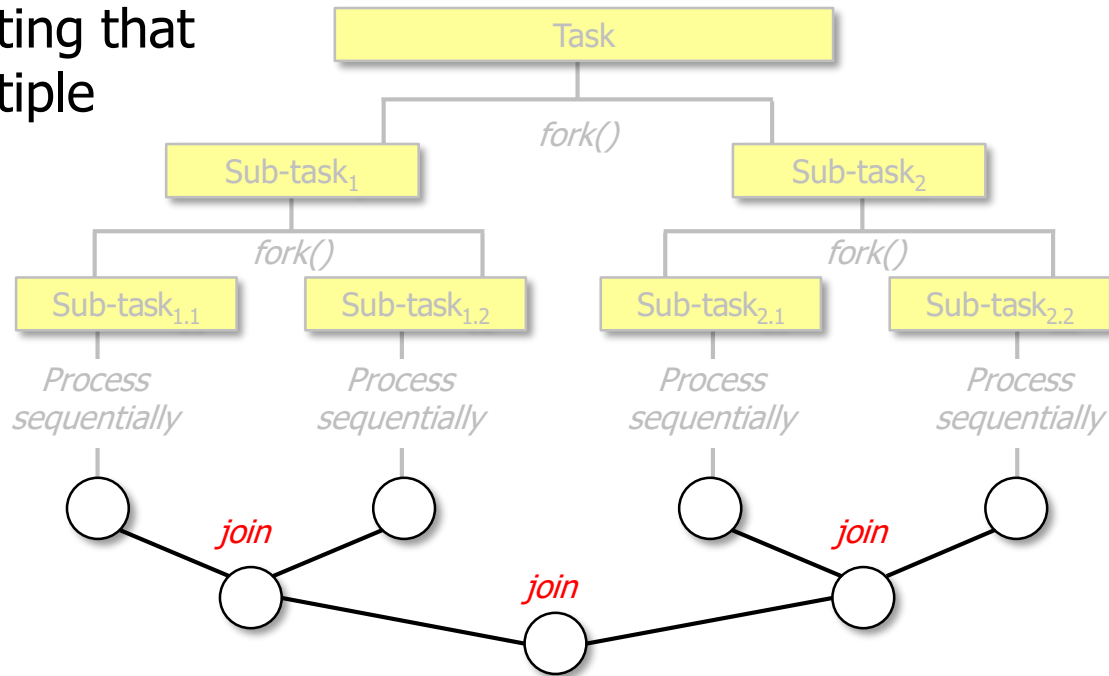
- Parallelism is a form of computing that performs several steps on multiple processor cores, i.e.
 - Split – partition a task into sub-tasks
 - Apply – Run independent sub-tasks in parallel



An Overview of Parallelism

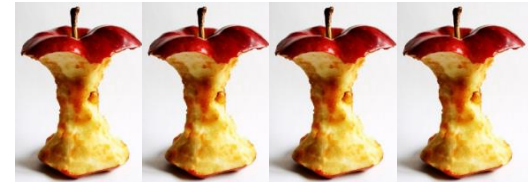
- Parallelism is a form of computing that performs several steps on multiple processor cores, i.e.

- Split – partition a task into sub-tasks
- Apply – Run independent sub-tasks in parallel
- Combine – Merge the sub-results from sub-tasks into one final result



An Overview of Parallelism

- A key goal of parallelism is to *efficiently* partition tasks into sub-tasks & combine results



An Overview of Parallelism

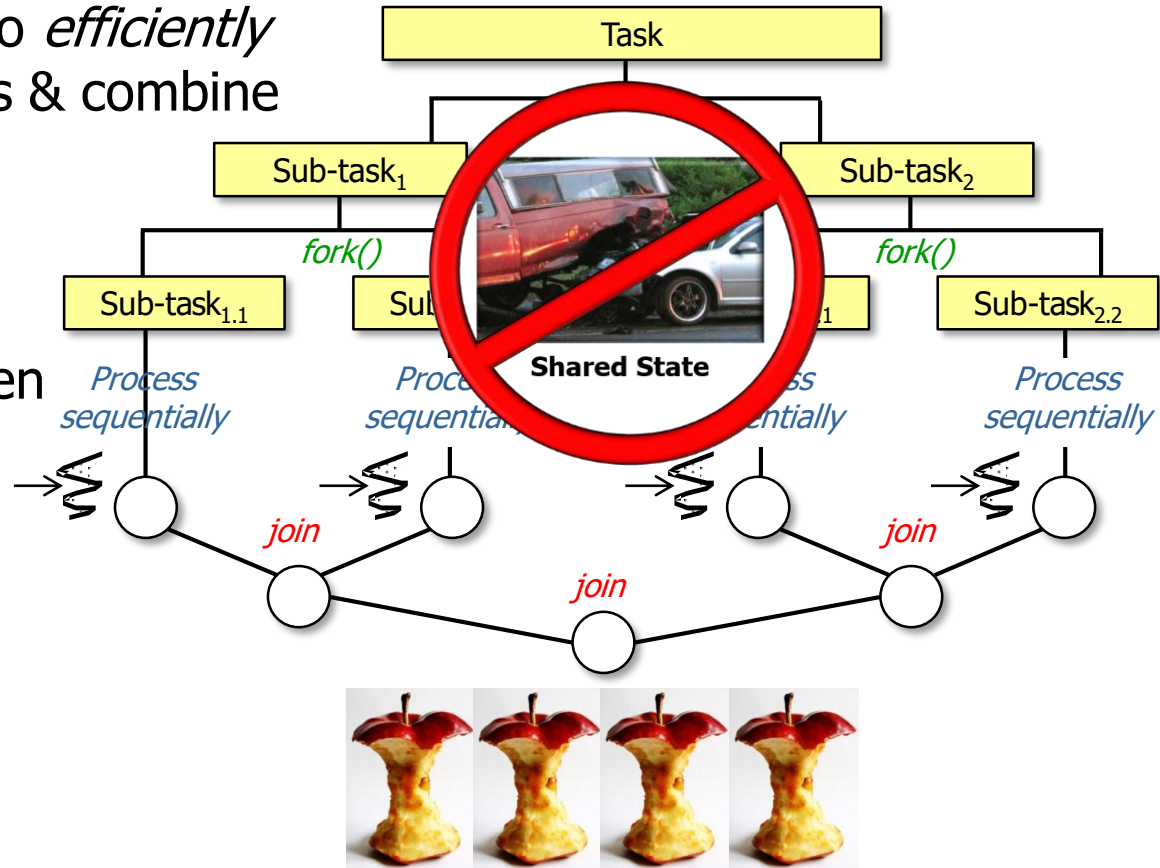
- A key goal of parallelism is to *efficiently* partition tasks into sub-tasks & combine results
- Parallelism thus focuses on optimizing performance
 - e.g., throughput, scalability, & latency



See www.ibm.com/developerworks/library/j-java-streams-4-brian-goetz

An Overview of Parallelism

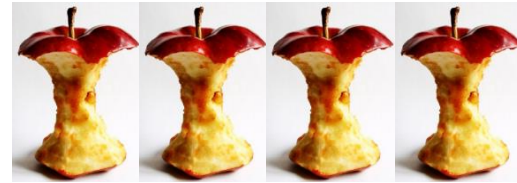
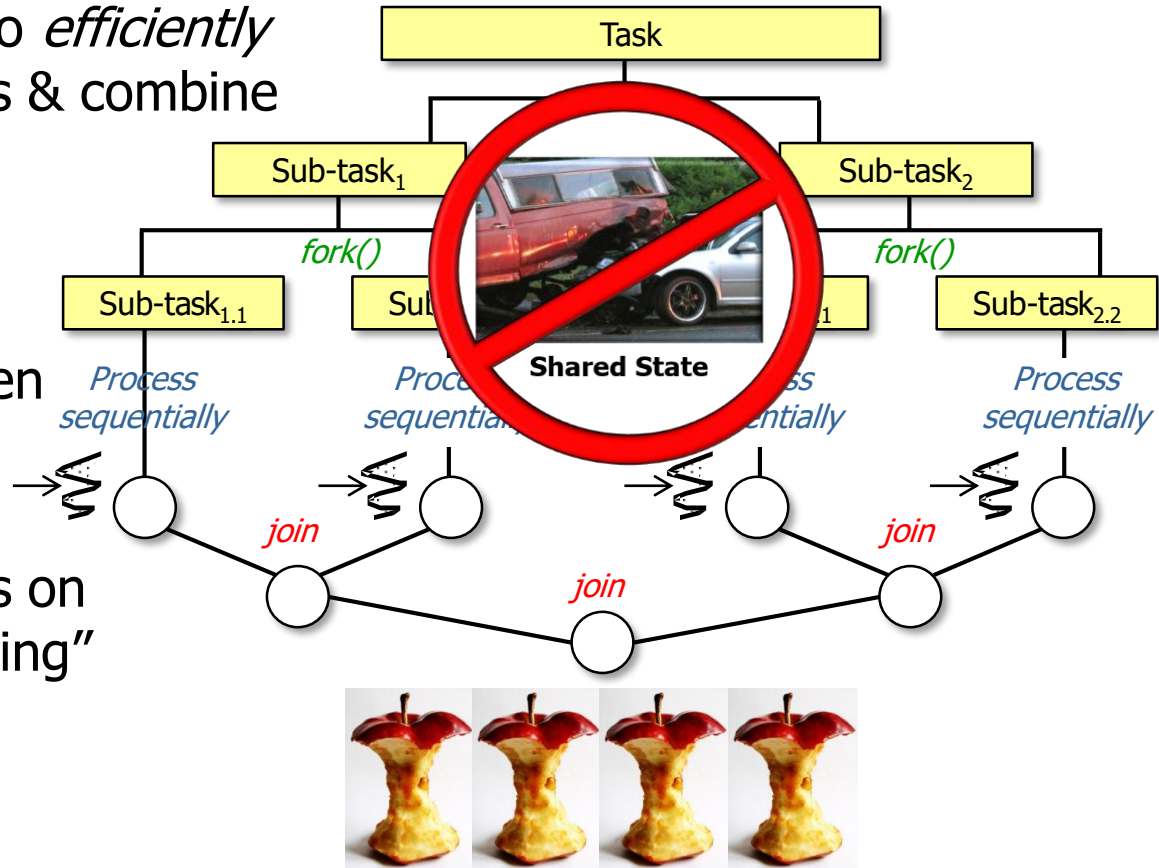
- A key goal of parallelism is to *efficiently* partition tasks into sub-tasks & combine results
- Parallelism thus focuses on optimizing performance
- Parallelism works best when threads share no mutable state & don't block



An Overview of Parallelism

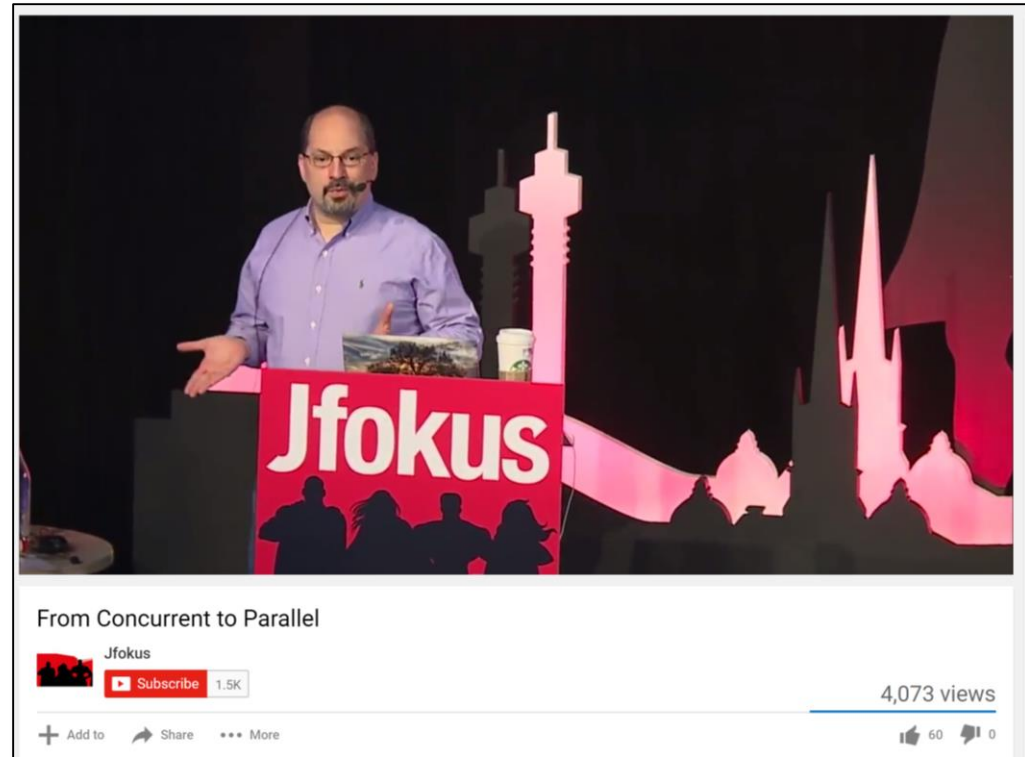
- A key goal of parallelism is to *efficiently* partition tasks into sub-tasks & combine results

- Parallelism thus focuses on optimizing performance
- Parallelism works best when threads share no mutable state & don't block
 - Hence Java 8's emphasis on "fork-join" & "work-stealing"



An Overview of Parallelism

- Brian Goetz has an excellent talk about the evolution of Java from concurrent to parallel computing

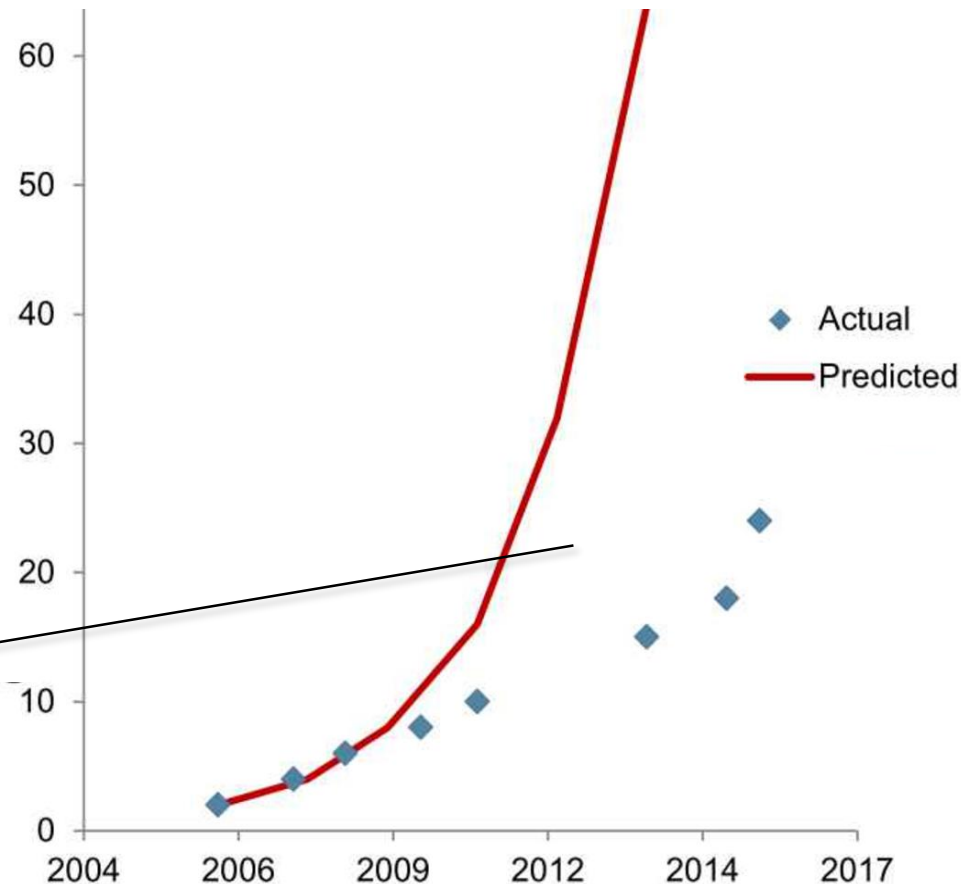


See www.youtube.com/watch?v=NsDE7E8sIdQ

An Overview of Parallelism

- Brian Goetz has an excellent talk about the evolution of Java from concurrent to parallel computing

His talk emphasizes that Java 8 combines functional programming with fine-grained data parallelism to leverage many-core processors



See www.infoq.com/presentations/parallel-java-se-8

End of Background on Java Concurrency & Parallelism (Part 1)