UNIVERSITY OF SAN FRANCISCO
CHANGE THE WORLD FROM HERE

Concurrency

Sami Rollins

Concurrency

- **What is concurrency?**

Concurrency

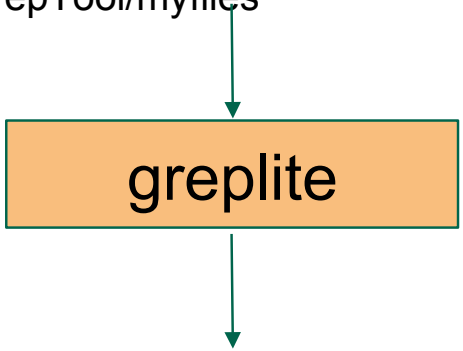
- **What is concurrency?**
  - Executing multiple tasks simultaneously
    - Processing multiple HTTP requests at the same time

- **Why concurrency?**
  - Improve performance
    - Take advantage of multiple cores
    - Do useful computation when blocked on i/o
  - Allow background activities

GrepLite

searchterm: computing

directory: /Users/srollins/GrepTool/myfiles



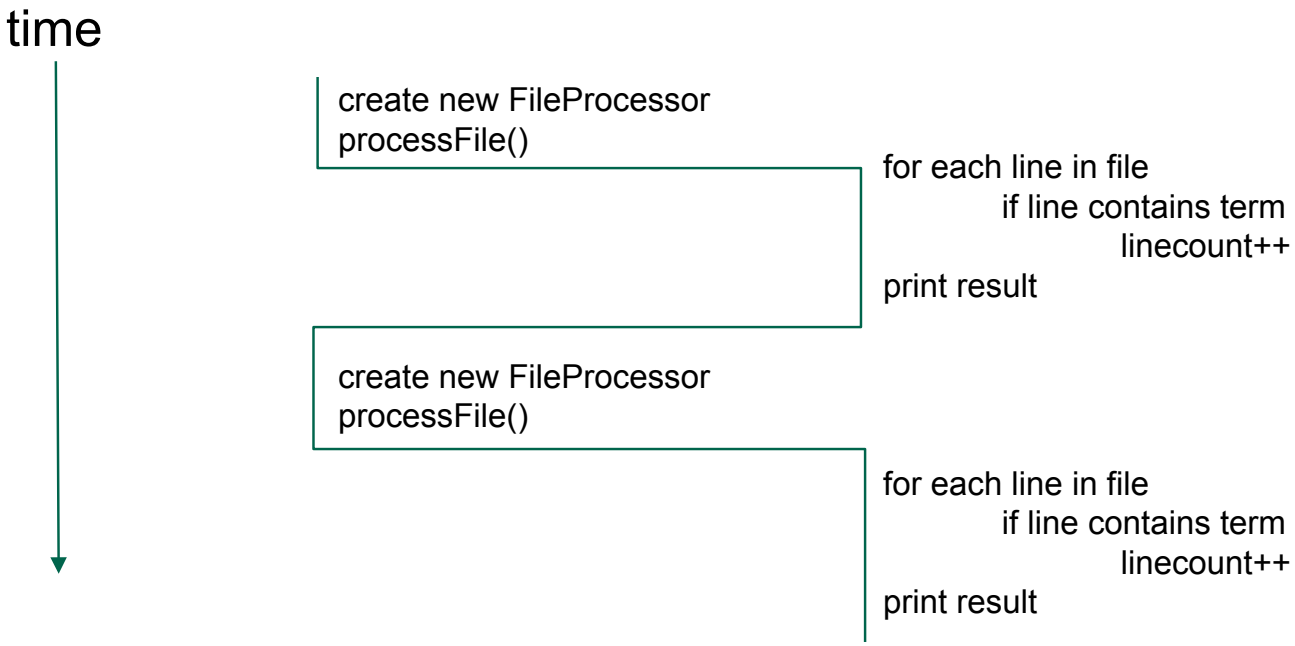
file 1: <count>

file 2: <count>

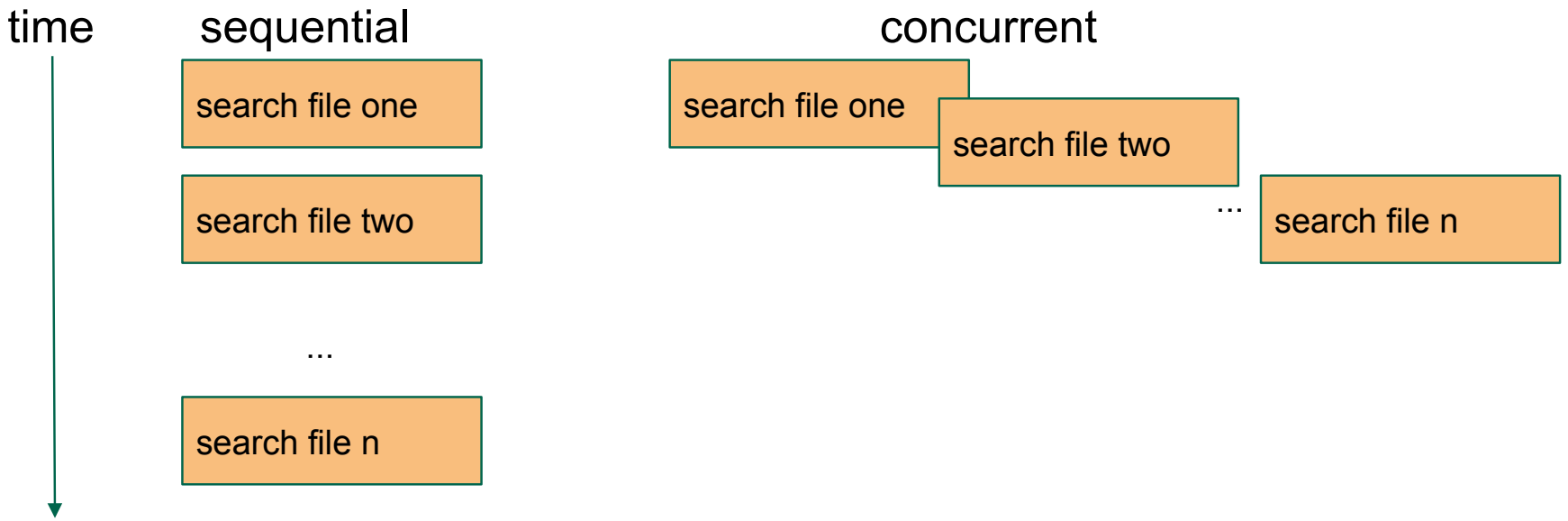
…

file n: <count>

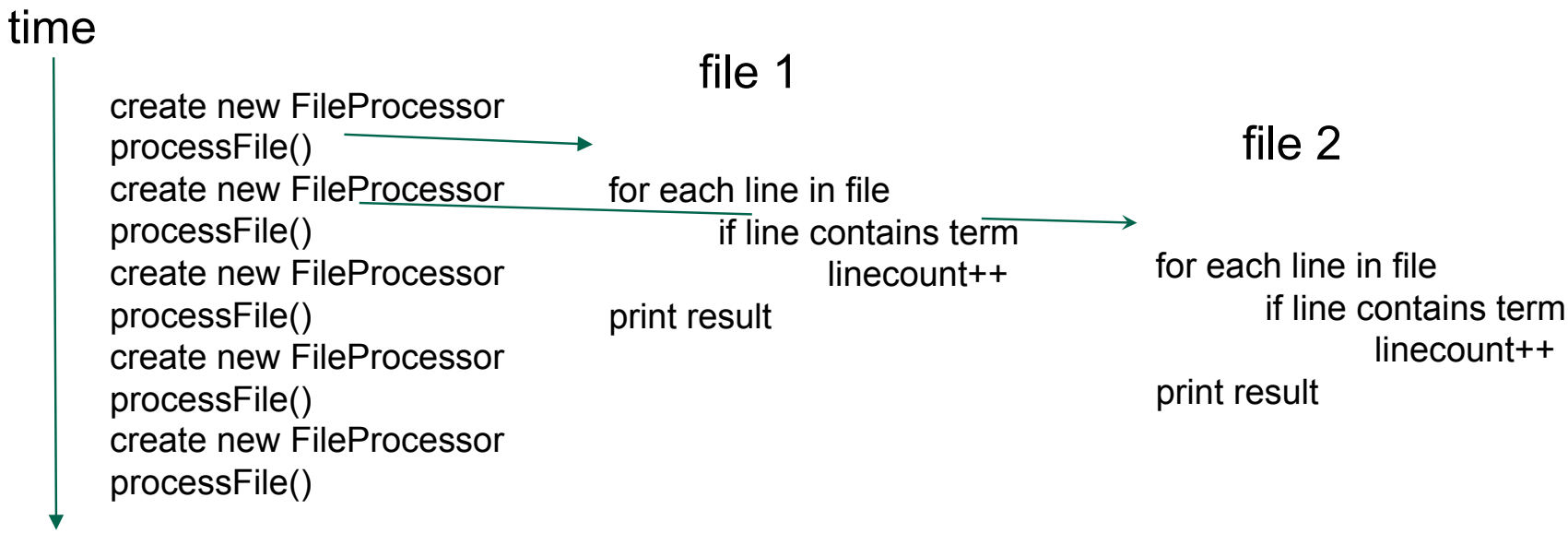
GrepLite Algorithm - Sequential



Sequential versus Concurrent

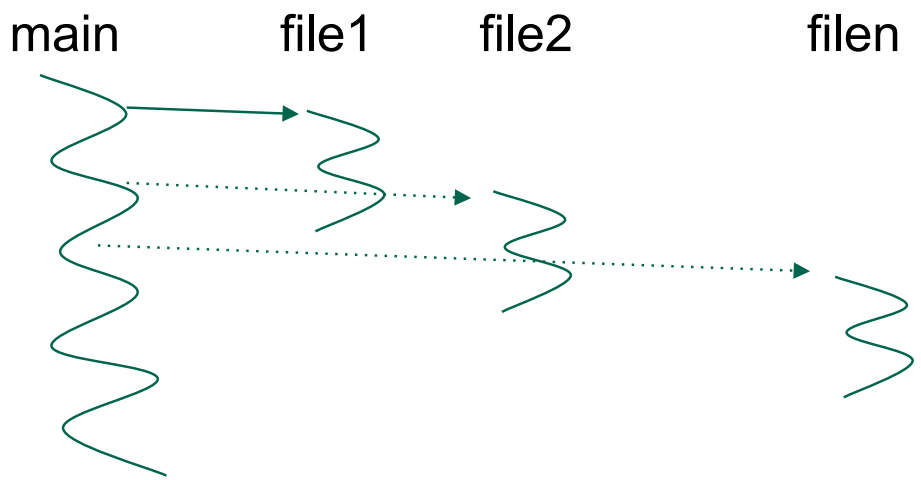


GrepLite Algorithm - Concurrent



Threads

- **Programming abstraction that allow *lightweight processes* within an application**



Challenges

- **Order of execution**
  - Race conditions
- **Inter-thread communication**
- **Shared data and thread safety**
- **Overhead**

Threads and sharing

Atomic operations

Suppose two threads operate on the same data

a initially has the value of 4

Thread 1:	Thread 2:
`a = a + 1;`	`a = a * 2;`

What are the possible resulting values for a?

10 (4 + 1 = 5, 5 * 2 = 10)

9 (4 * 2 = 8, 8 + 1 = 9)

how about 5 or 8?

Threads and sharing

a = a + 1

1. read value of a
2. set a to a + 1

a = a * 2

3. read value of a
4. set a to a * 2

Option 1: 1, 2, 3, 4 -- a will be10
Option 2: 3, 4, 1, 2 -- a will be 9
Option 3: 1, 3, 2, 4 -- a will be 8
Option 4: 1, 3, 4, 2 -- a will be 5
BUT, options 3 and 4 should never occur

Atomic operations

atomic operations execute without interference from other operations

request a *lock* before an operation, release it when the operation is complete

`synchronized` is a Java keyword that acquires a lock on an object
`synchronized` may be used for methods or blocks of statements

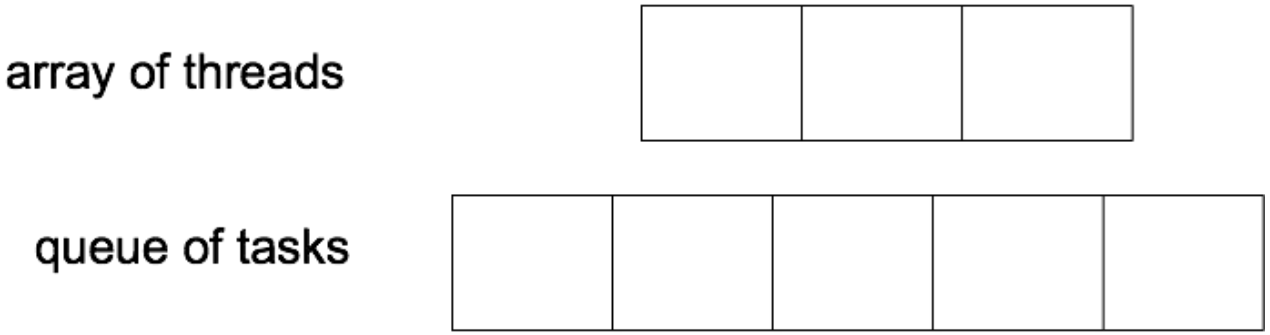
synchronized methods may not interleave

Thread pool

There is an overhead associated with creating and destroying threads

Each operation requires a new thread

A solution is to create a *pool* of a fixed number of threads. As tasks arrive, a thread from the pool is used to execute the task.



Communication

Threads *wait* for new work

When new work is inserted, the queue must *notify* the threads that new work is available

`wait` and `notify` are Java keywords.

`wait` can be called on any object. The thread that calls `wait` suspends execution until it is *woken up* with a `notify` message called on the same object.

In most cases, `notifyAll` is used instead of `notify`. `notifyAll` will wake up every thread that has called `wait` on the given object.