Gregor:

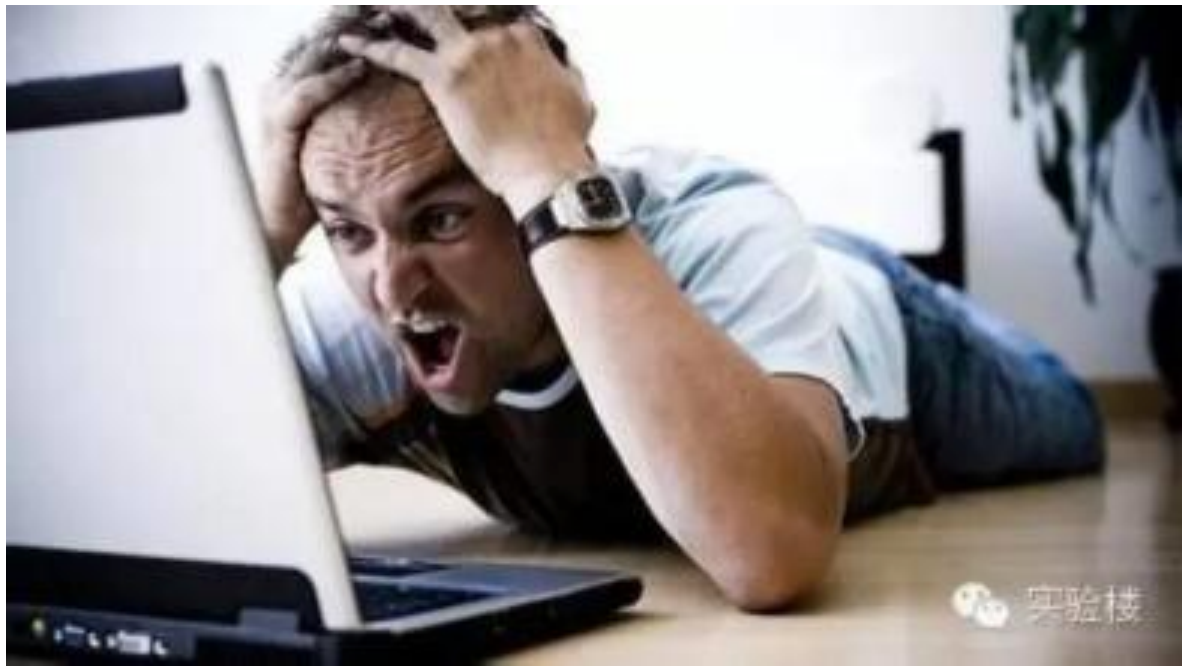
A Worker-Pool based Multithread Programming Framework

MSCS

Zhan Chen (zhanc1@andrew)
Lehao Sun (lehaos@andrew)

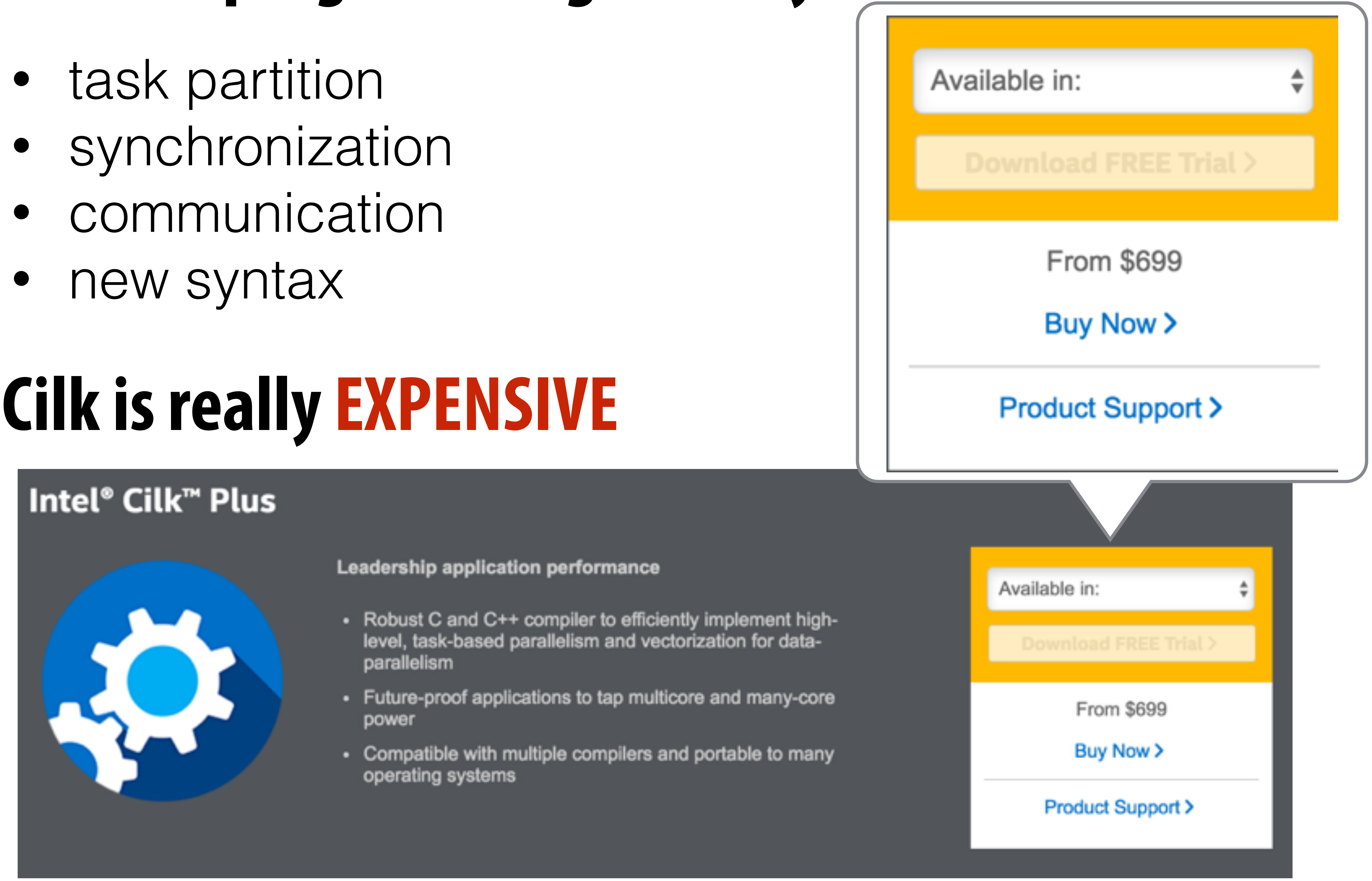
Why is Gregor?

- **Parallel programming is really HARD**
  - task partition
  - synchronization
  - communication
  - new syntax



Why is Gregor?

- **Parallel programming is really HARD**
  - task partition
  - synchronization
  - communication
  - new syntax
- **Cilk is really EXPENSIVE**



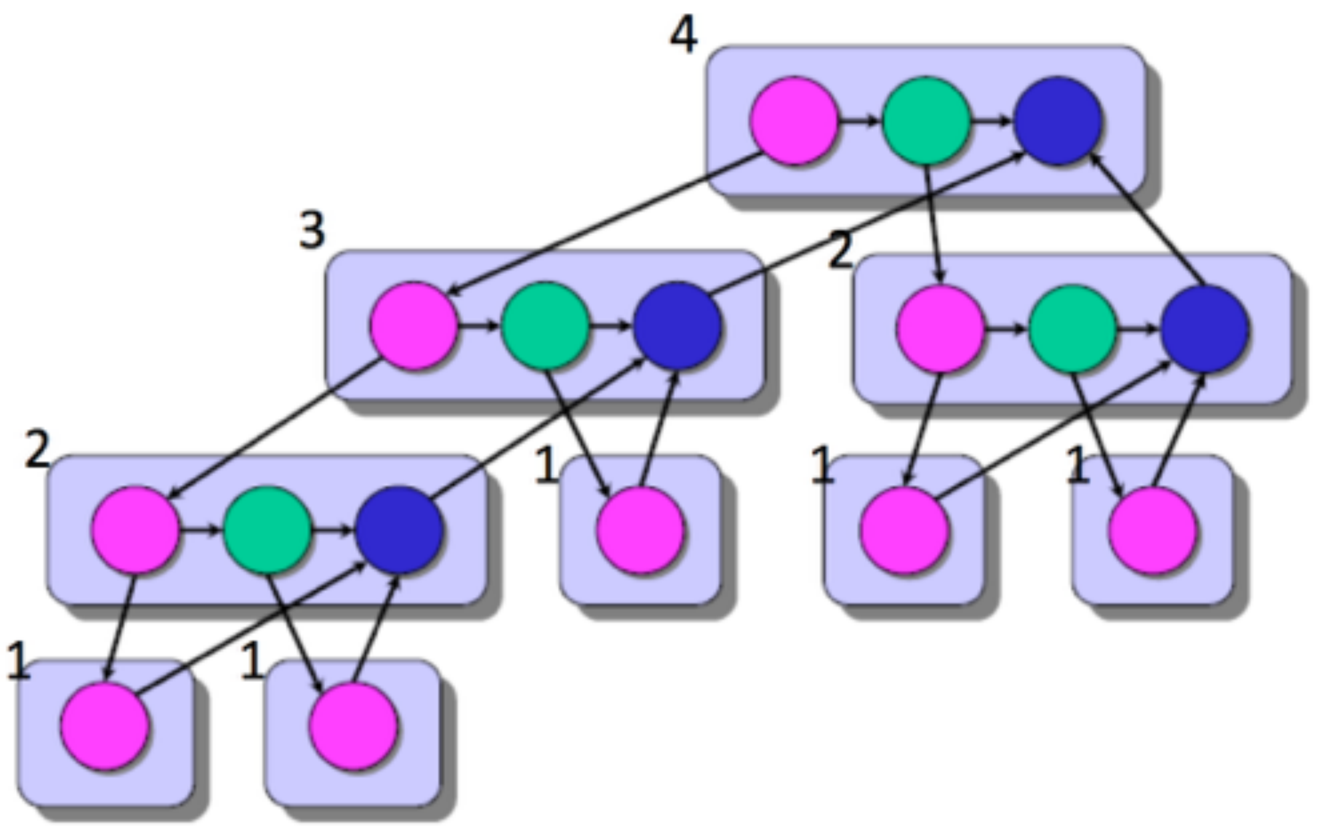
Intel® Cilk™ Plus

Leadership application performance

- Robust C and C++ compiler to efficiently implement high-level, task-based parallelism and vectorization for data-parallelism
- Future-proof applications to tap multicore and many-core power
- Compatible with multiple compilers and portable to many operating systems

What do we offer?

- spawn
- sync
- high performance
- good scalability
- ......



An example:
Programming with Gregor

Code example

```
int fib(int n) {
  if (n < 2)
        return (n);
  else {
        int x, y;
        spawn(INT, &x, fib, 1, INT, n - 1);
        spawn(INT, &y, fib, 1, INT, n - 2);

        __gregor_sync();
      return (x + y);
  }
}
```

SPAWN

```
spawn(INT, &x, fib, 1, INT, n-1);
```

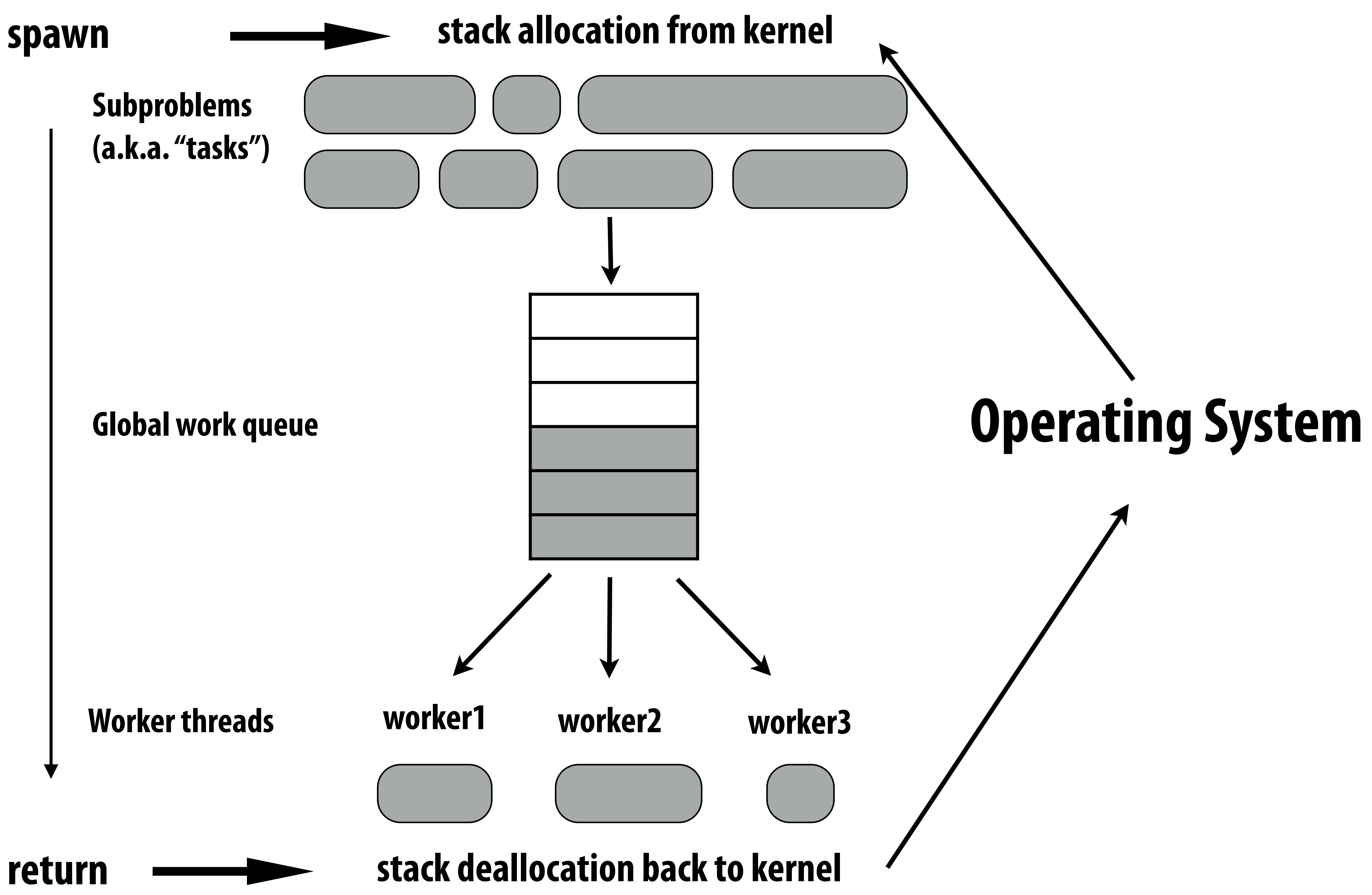
Semantics: invoke jobs which can run in parallel with the current one

sync

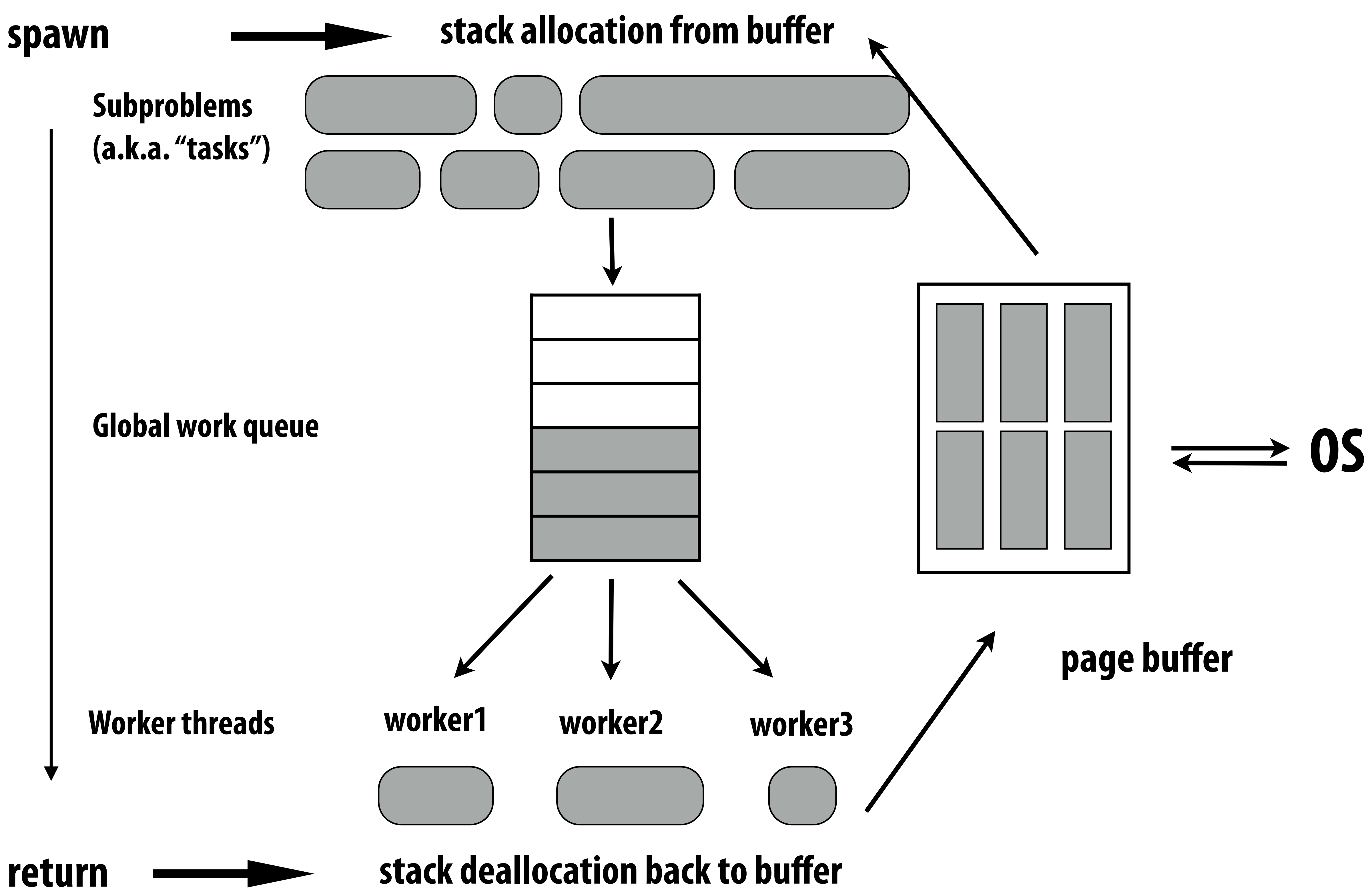
```
__gregor_sync();
```

Semantics: the control flow will not resume until all the spawned jobs return

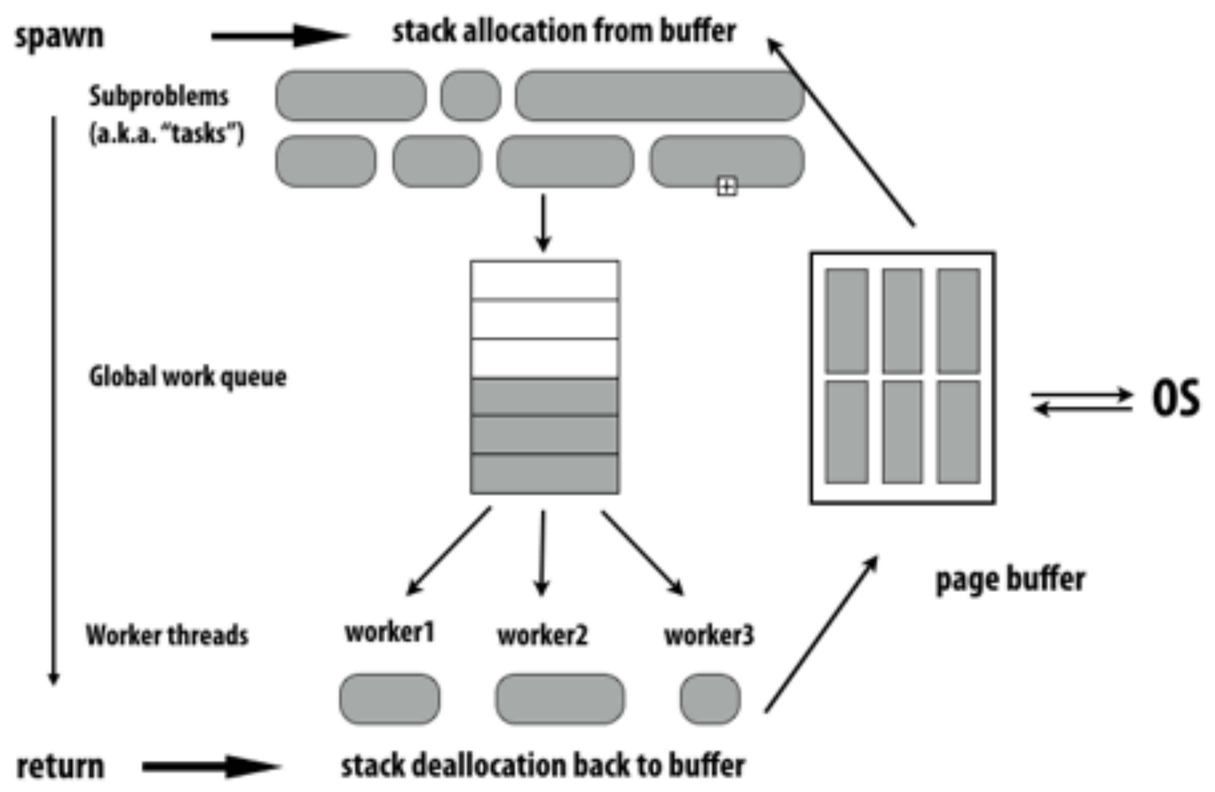
Initial Version

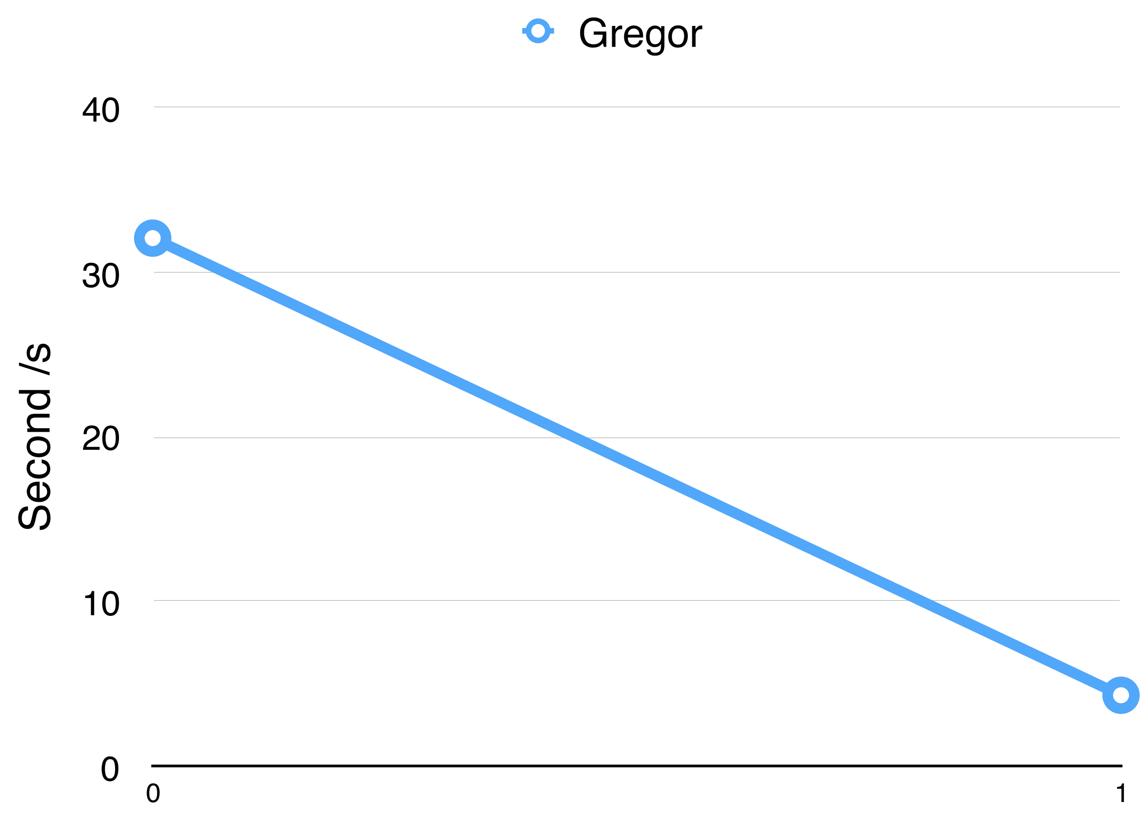


Optimized With Memory Management



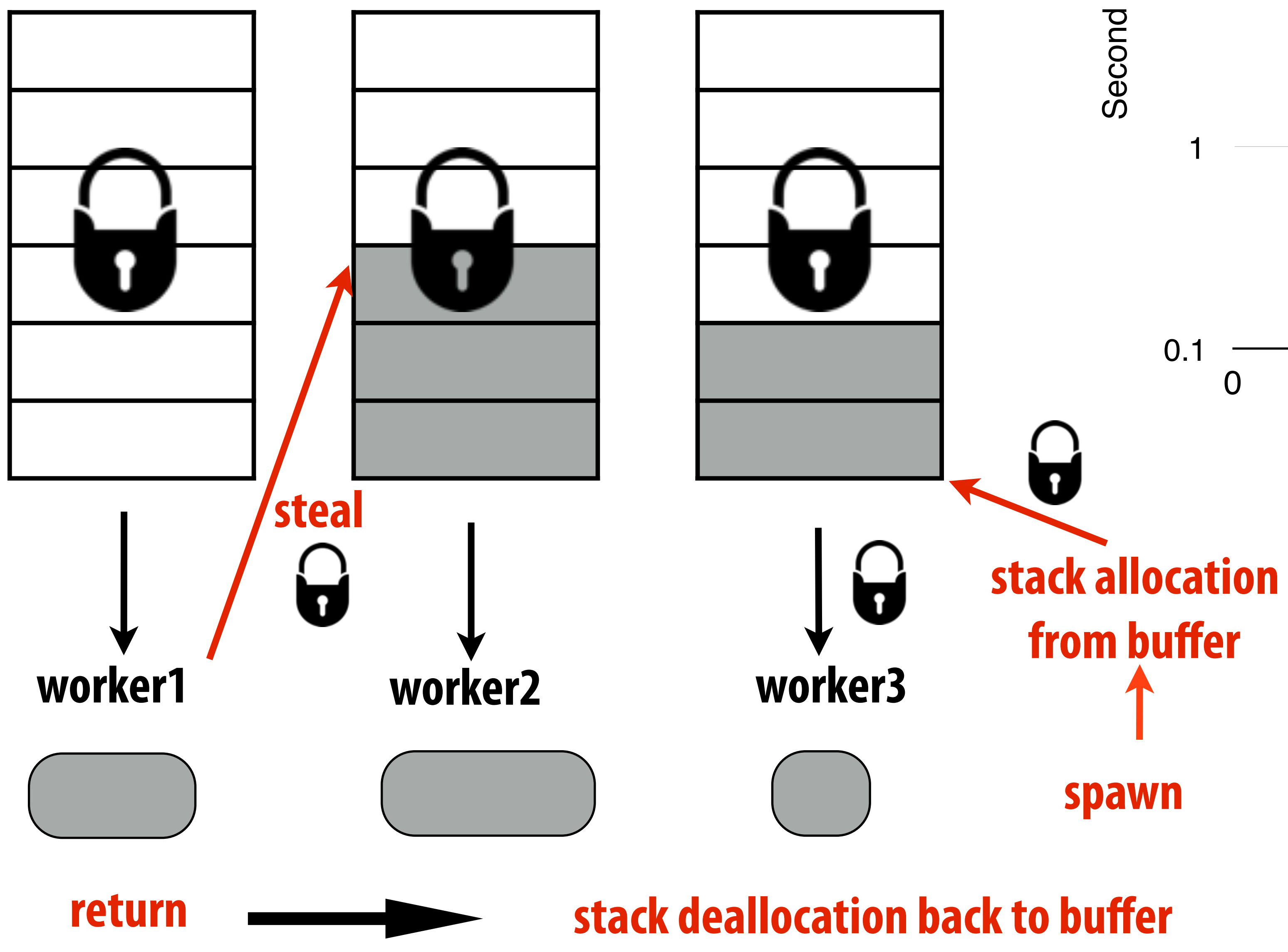
Optimized With Memory Management


spawn
stack allocation from buffer
Subproblems (a.k.a. "tasks")
Global work queue
OS
page buffer
Worker threads
worker1
worker2
worker3
return
stack deallocation back to buffer

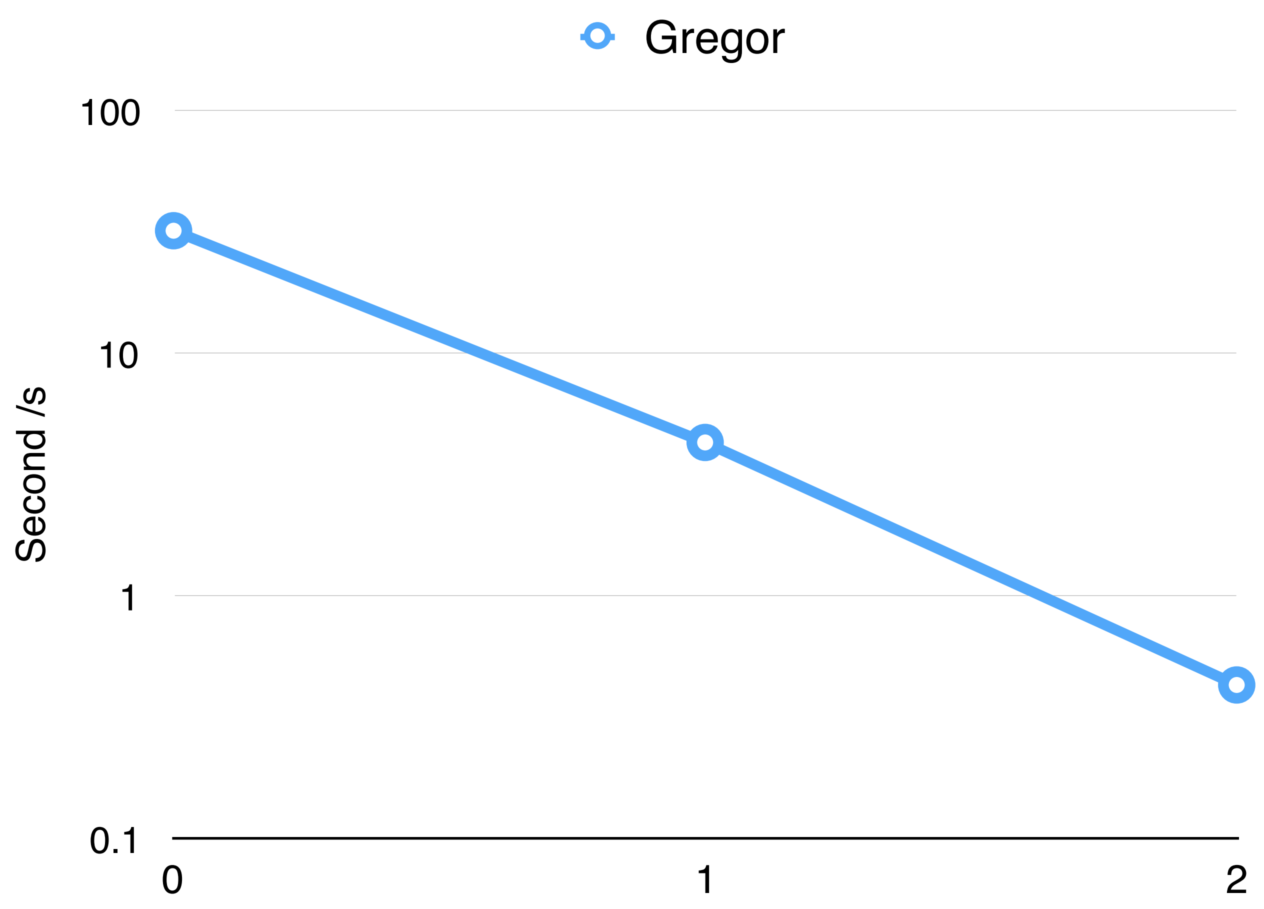


- Setting: latedays machine
- Application: Fib(30)

Optimized With Job Stealing

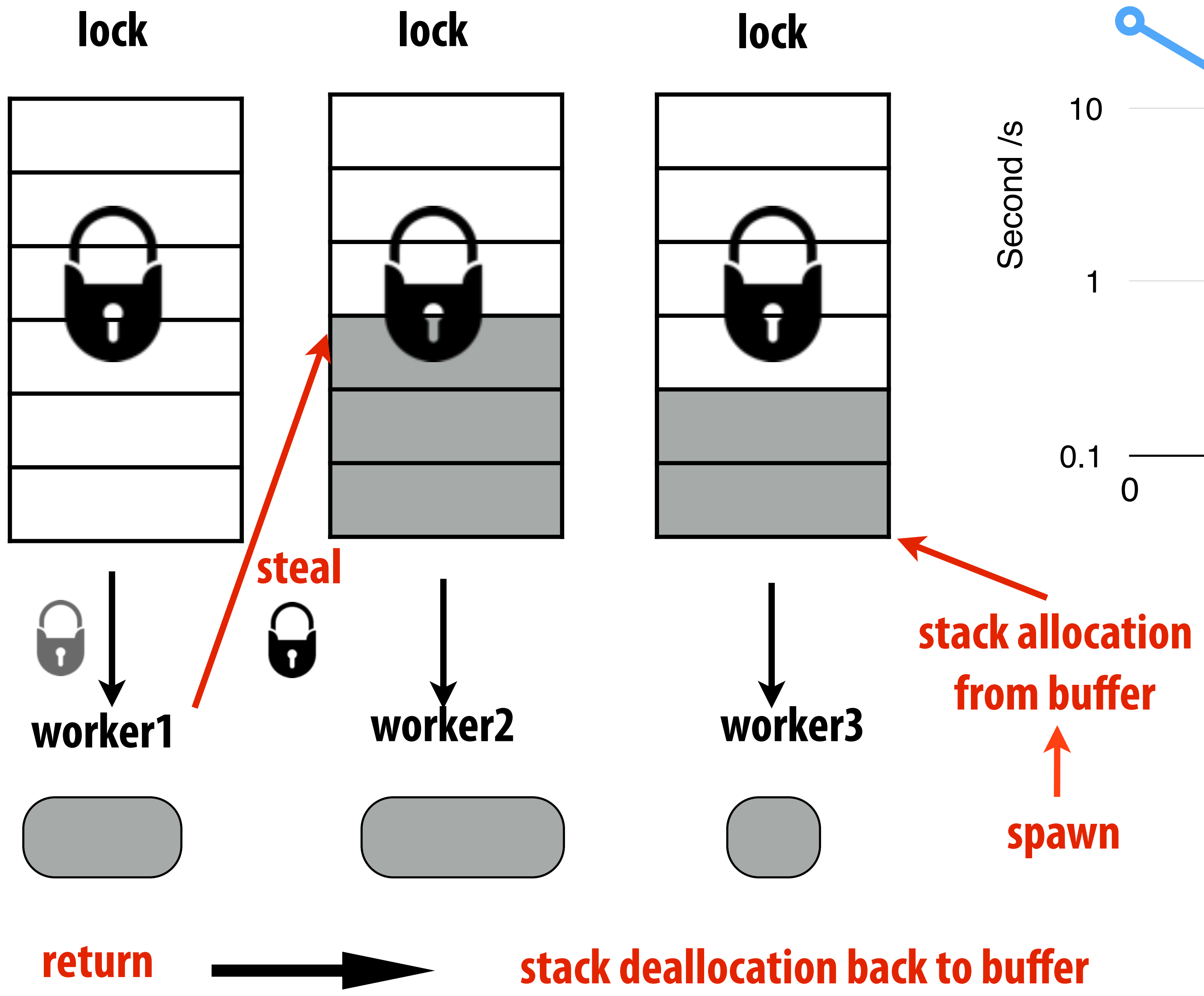


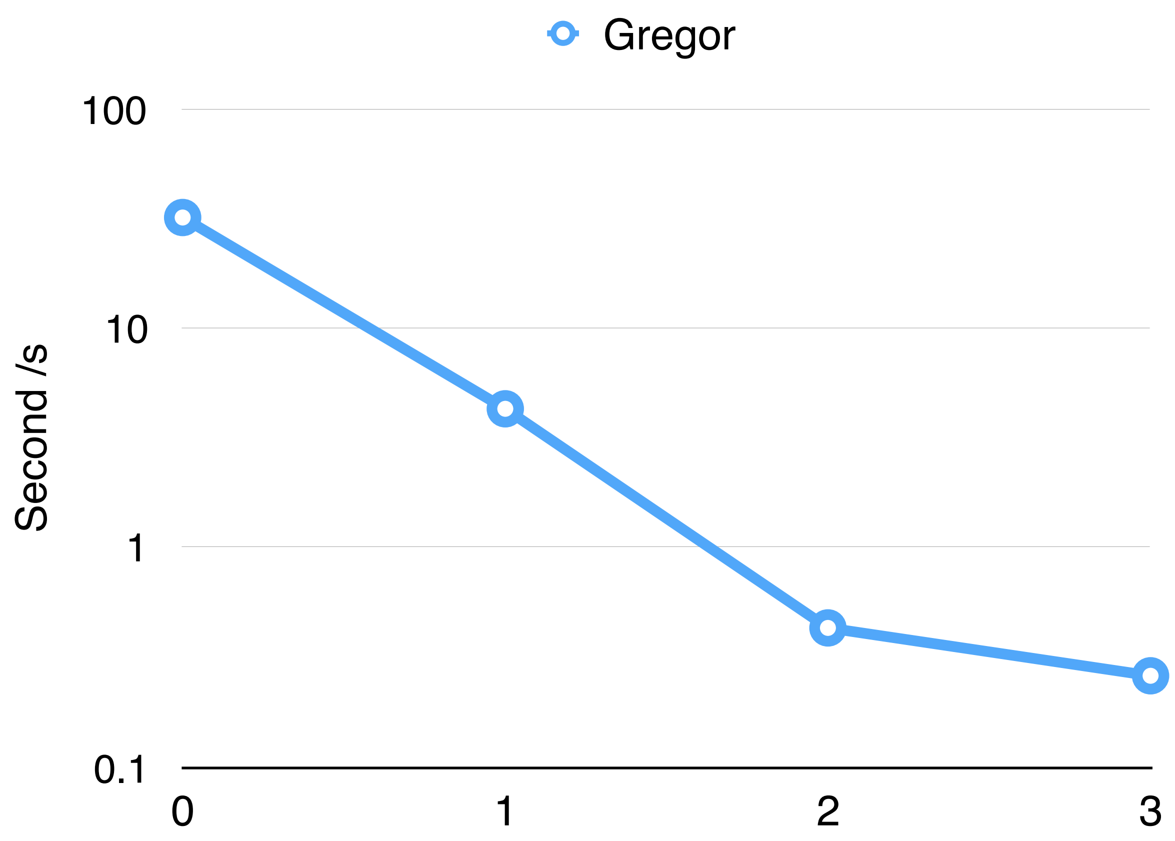
steal
stack allocation from buffer
spawn
worker1
worker2
worker3
return
stack deallocation back to buffer



- Setting: latedays machine
- Application: Fib(30)

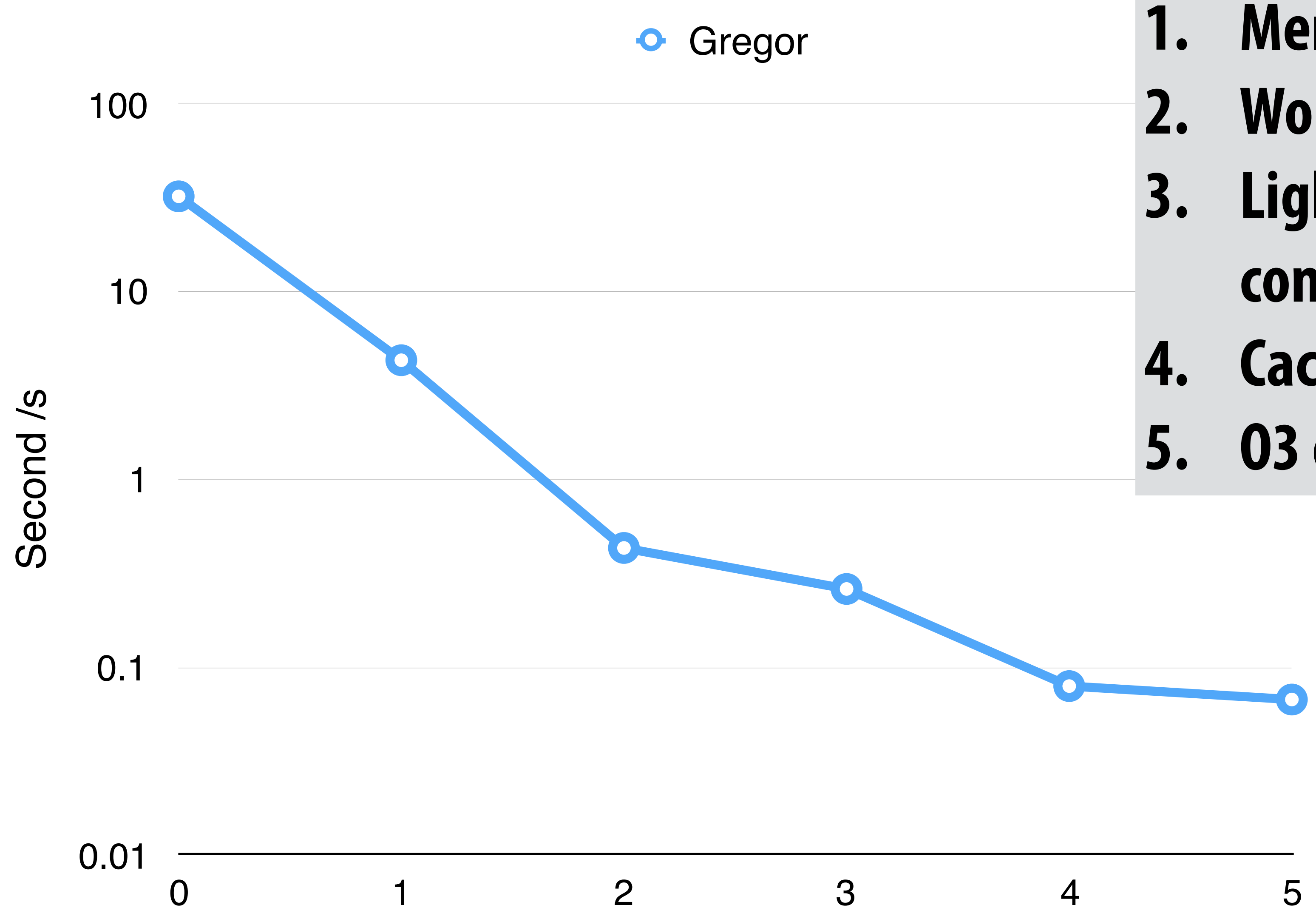
Optimized With Lightweight Synchronization and Context Switch Saving


lock
lock
lock
steal
stack allocation from buffer
worker1
worker2
worker3
spawn
return
stack deallocation back to buffer



- Setting: latedays machine
- Application: Fib(30)

Further Optimization: Cache Alignment and 03

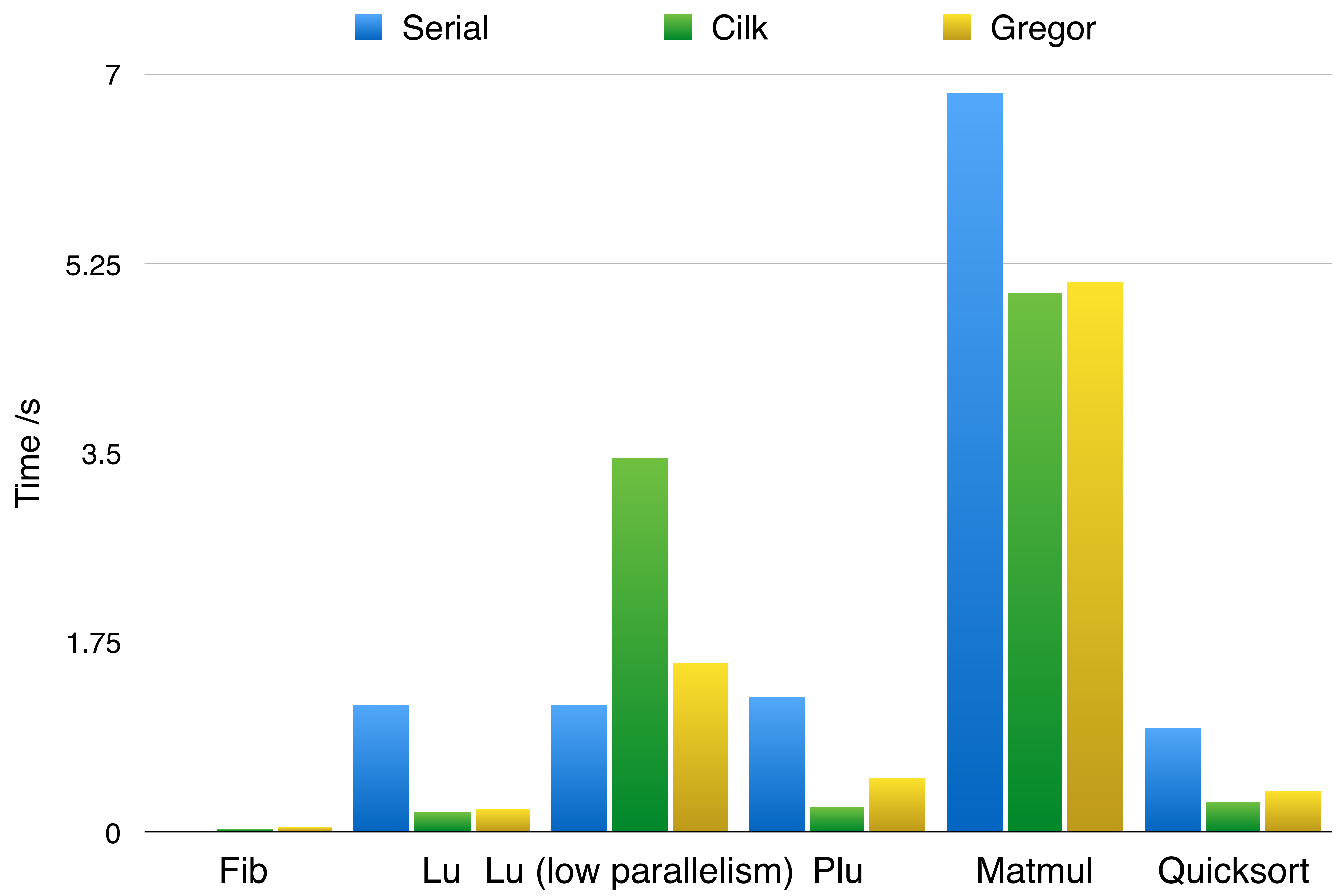


Overview

1. **Memory Management (buffer)**
2. **Work Stealing with giant lock**
3. **Lightweight Synchronization reduce context saving overhead**
4. **Cache Alignment**
5. **03 optimization**

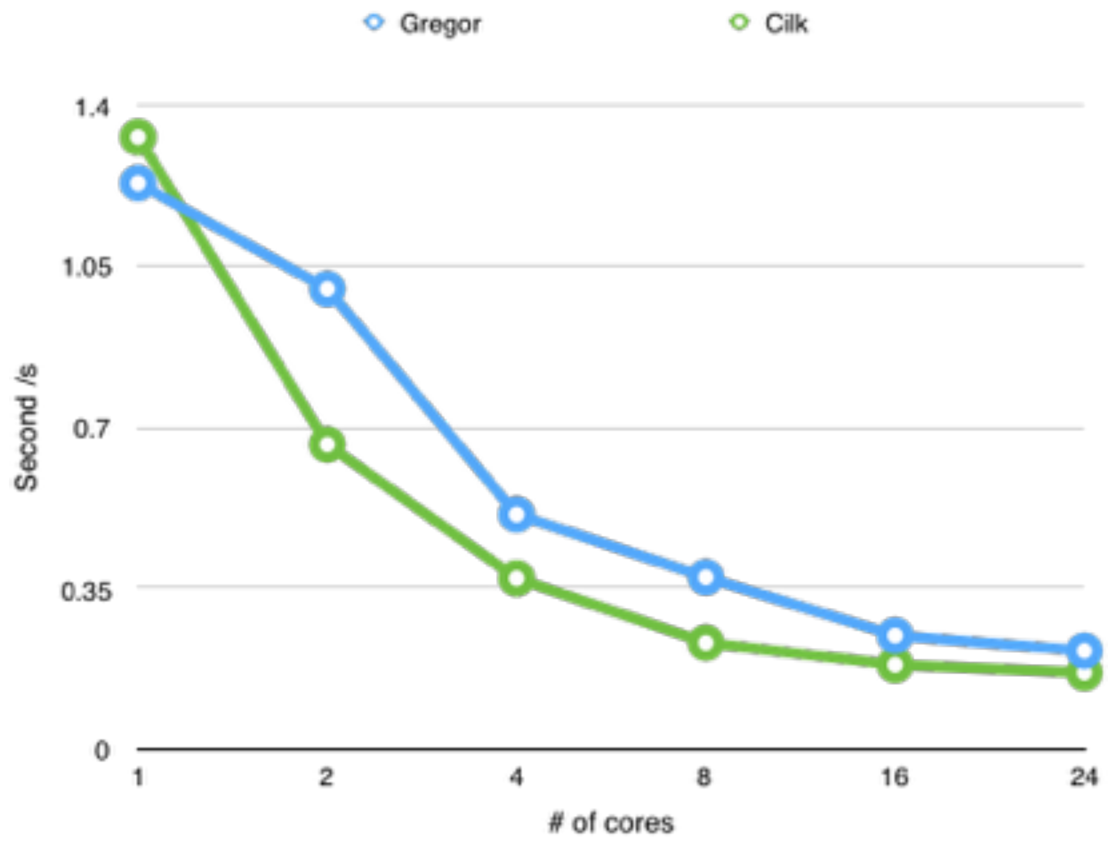
- Setting: latedays machine
- Application: Fib(30)

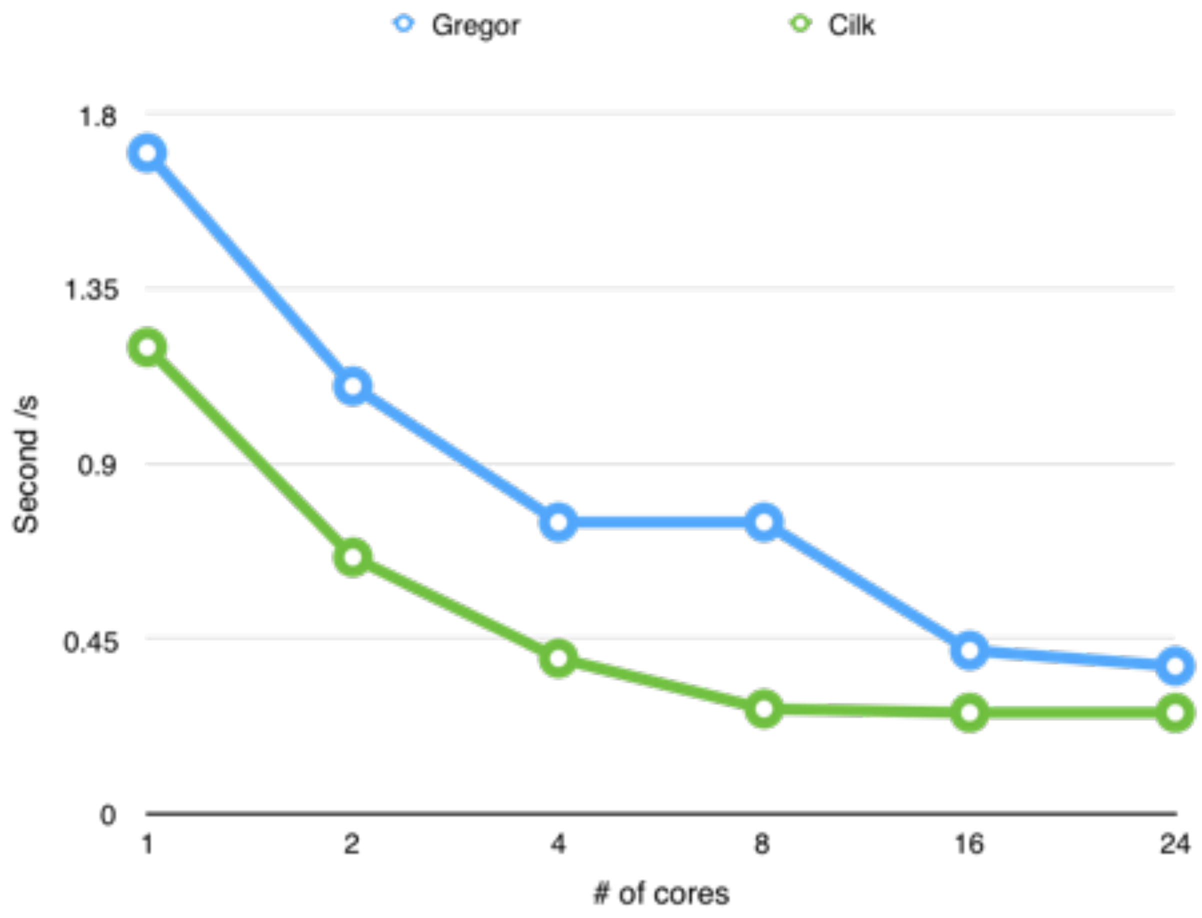
Results



Setting: latedays machine

Scalability





Lu

LU-decomposition (without pivoting) of a dense n x n matrix. The default number of n is 1024.

Quicksort

Sort an out-of-order array in O(nlogn). The default number of n is 10240000.

Setting: latedays machine

Thanks for listening

Zhan Chen (zhanc1@andrew)
Lehao Sun (lehaos@andrew)