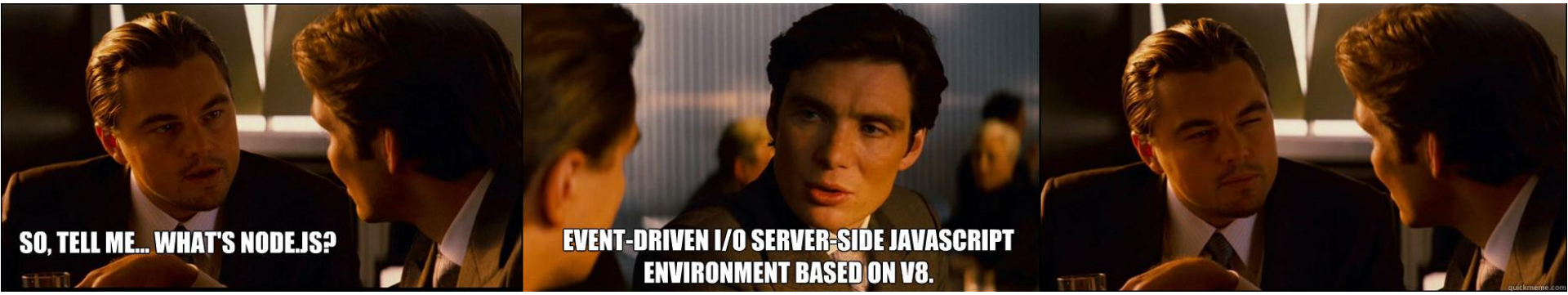
Node.js concurrency

Giacomo Fornari

University of Padua,
Sistemi concorrenti e distribuiti,
2016 - 2017

What is Node.js?



Two theses

1. waiting an I/O operation is a waste of time
2. thread-per-connection is memory-expensive

The cost of I/O

“Non-blocking”	L1-cache	3 cycles
	L2-cache	14 cycles
	RAM	250 cycles
“Blocking”	Disk	41 000 000 cycles
	Network	240 000 000 cycles

Ryan Dahl: Introduction to Node.js, https://www.youtube.com/watch?v=M-sc73Y-zQA

Features

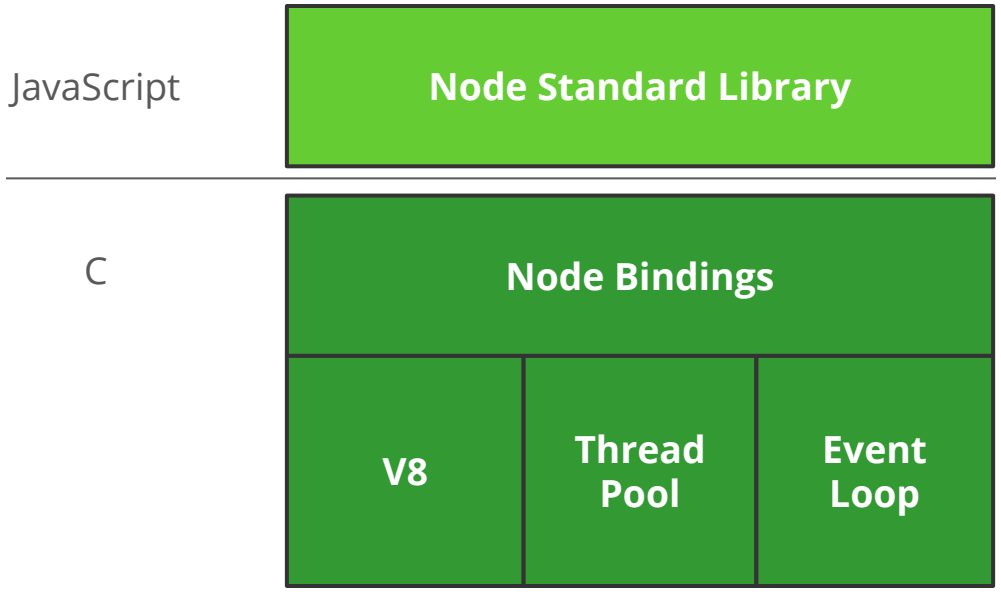
- Single-thread
- Highly concurrent
- No parallelism
- Event-driven

Programmer's point of view

"everything runs in parallel except your code" [1]

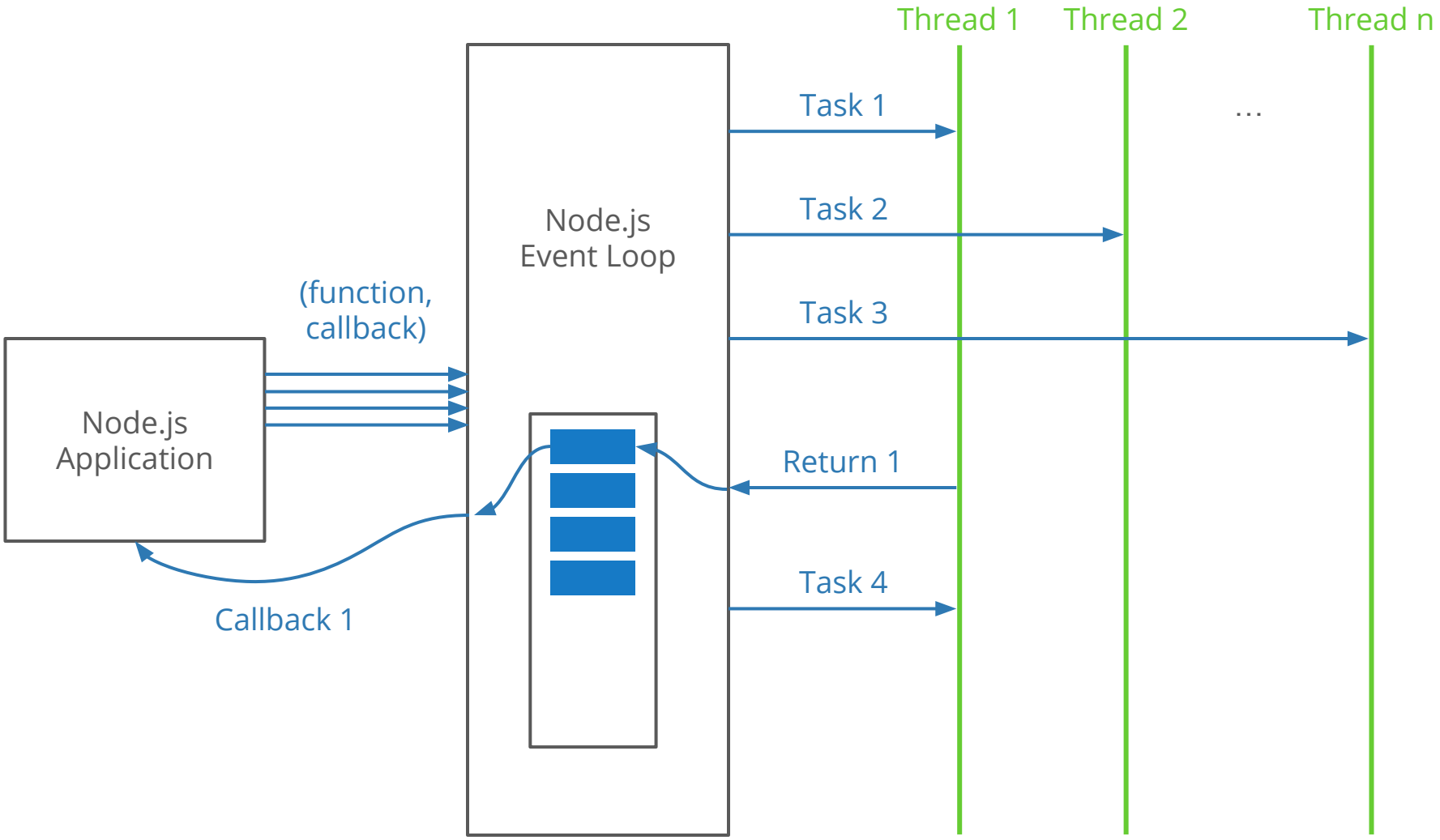
1. *http://blog.mixu.net/2011/02/01/understanding-the-node-js-event-loop/*

Architecture



Event Loop

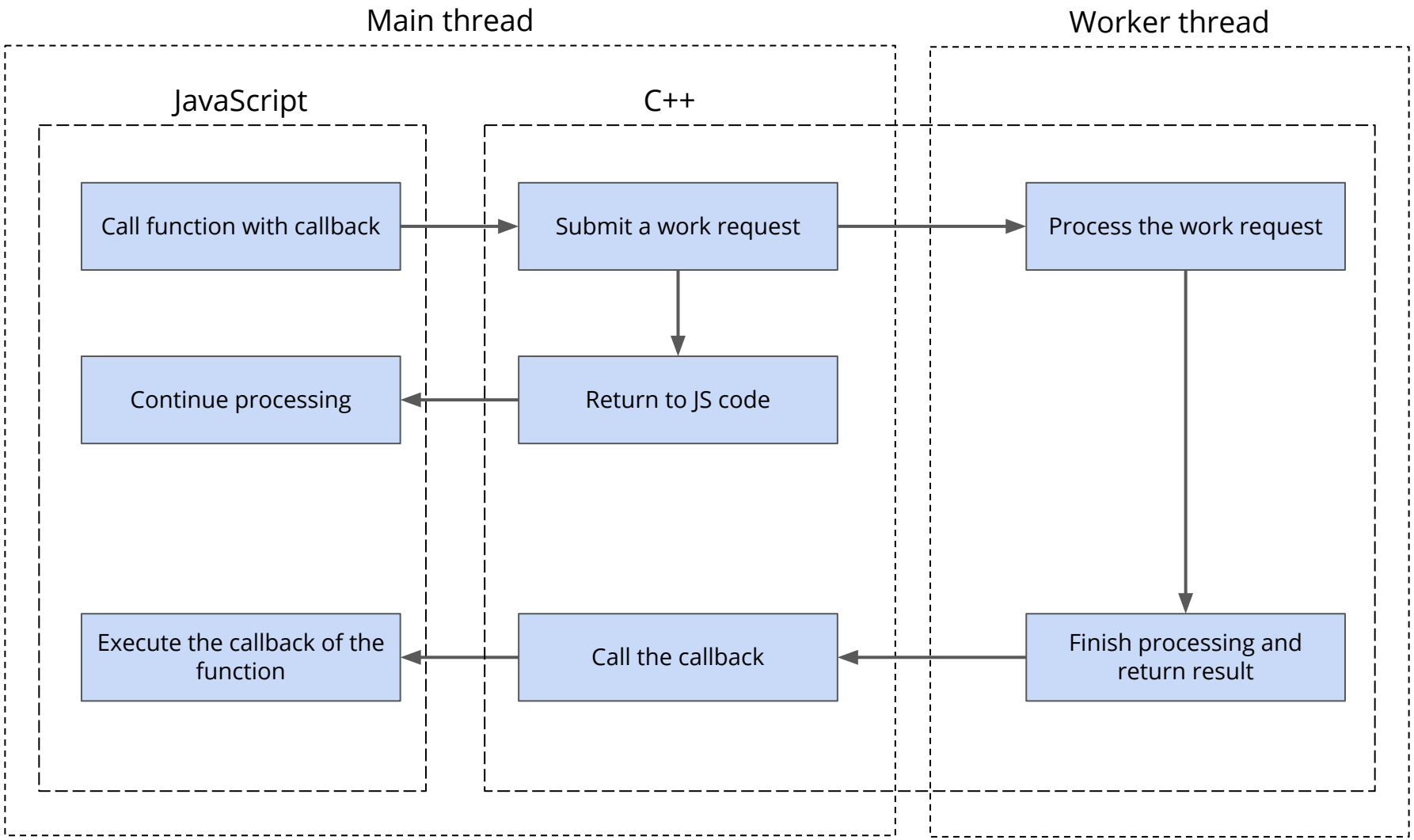
- Has a sequential queue of functions linked with events
- Runs on single-thread
  - it has only one stack
- The functions are defined by callbacks
  - short execution time
  - they can yield events and asynchronous operations
- Implicit usage



Event Loop diagram

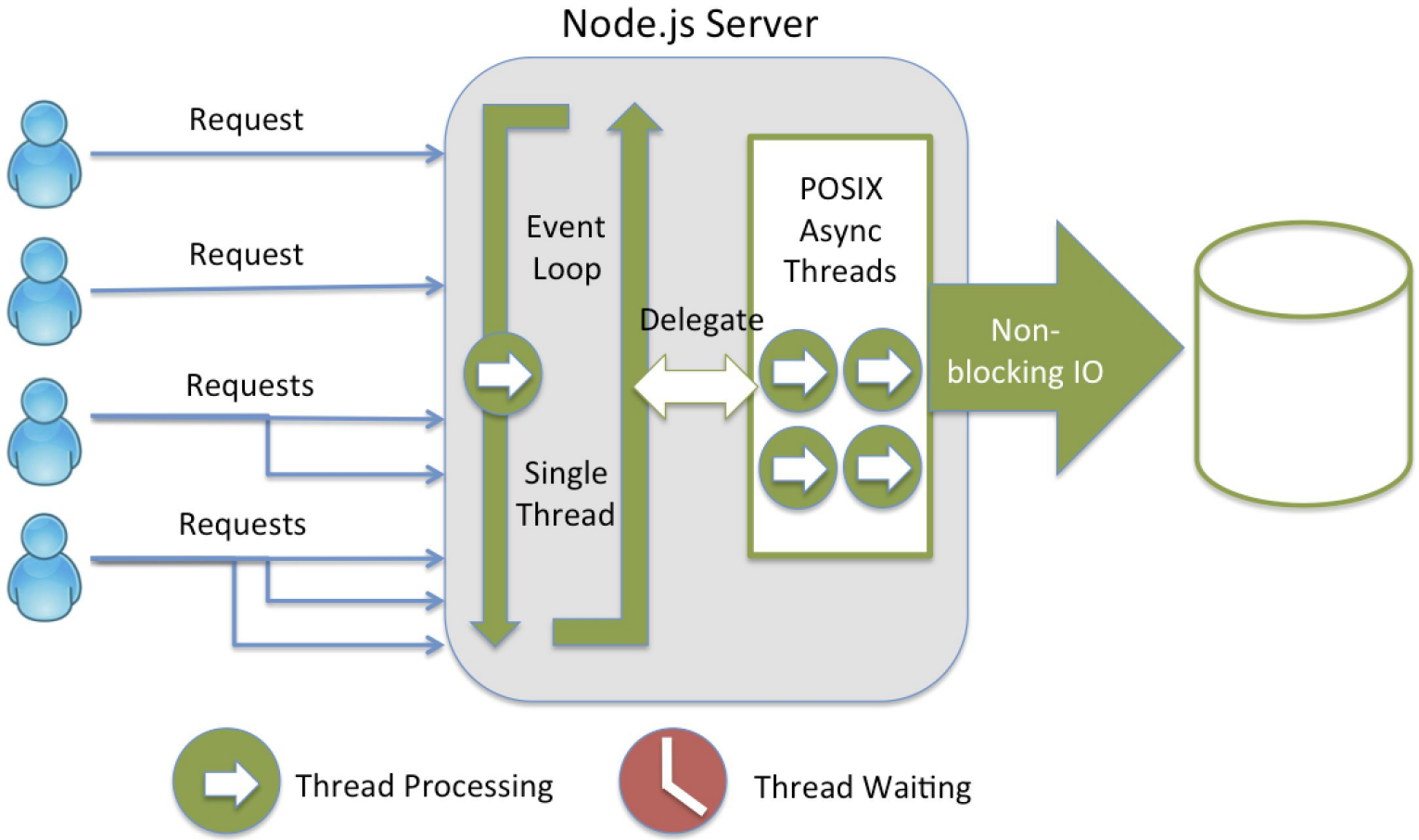
Thread Pool

- The strategy to achieve asynchronous I/O is not always a thread pool
  - Yes file-system operation
  - No network operations
- It has a fixed number of 4 thread
  - It is possible to set the number of thread before it is requested and created
- It has an internal queue of
  - function to execute
  - data for the function
  - function to call with the result



Thread delegation

Event Loop and Thread Pool diagram



https://strongloop.com/strongblog/node-js-is-faster-than-java/

return vs. callback

```
function add(a, b) {
  return a + b;
}
```

```
function add(a, b, cb) {
  cb(a + b);
}
```

sync op vs. async op

```
const dinner = ['pasta', 'meat', 'tiramisù'];
dinner.forEach((food) => console.log(food));
```

```
pasta
meat
tiramisù
```

```
const fs = require('fs');

dinner.forEach((food) => {
  fs.readdir('.', () => {
    console.log(food);
  });
});

console.log('prosecco');
```

```
prosecco
meat
tiramisù
pasta
```

Conclusions

- Concurrency reasoning easy
- No concurrent access of the state
- No locks
- No deadlock
- Still possible
  - starvation
  - race condition

References

- http://blog.mixu.net/2011/02/01/understanding-the-node-js-event-loop/
- http://www.journaldev.com/7462/node-js-architecture-single-threaded-event-loop
- http://neilk.net/blog/2013/04/30/why-you-should-use-nodejs-for-CPU-bound-tasks/
- https://oparu.uni-ulm.de/xmlui/bitstream/handle/123456789/2450/vts_8082_11772.pdf, ch. 5.5
- https://strongloop.com/strongblog/node-js-is-faster-than-java/
- http://www.slideshare.net/RonPerlmuter/nodejs-meetup-at-palo-alto-networks-tel-aviv
- http://www.slideshare.net/ganeshiyer7/nodejs-event-driven-concurrency-for-web-applications
- http://stackoverflow.com/questions/10680601/nodejs-event-loop
- http://stackoverflow.com/questions/22644328/when-is-the-thread-pool-used
- https://www.youtube.com/watch?v=M-sc73Y-zQA