Fast Java Programs

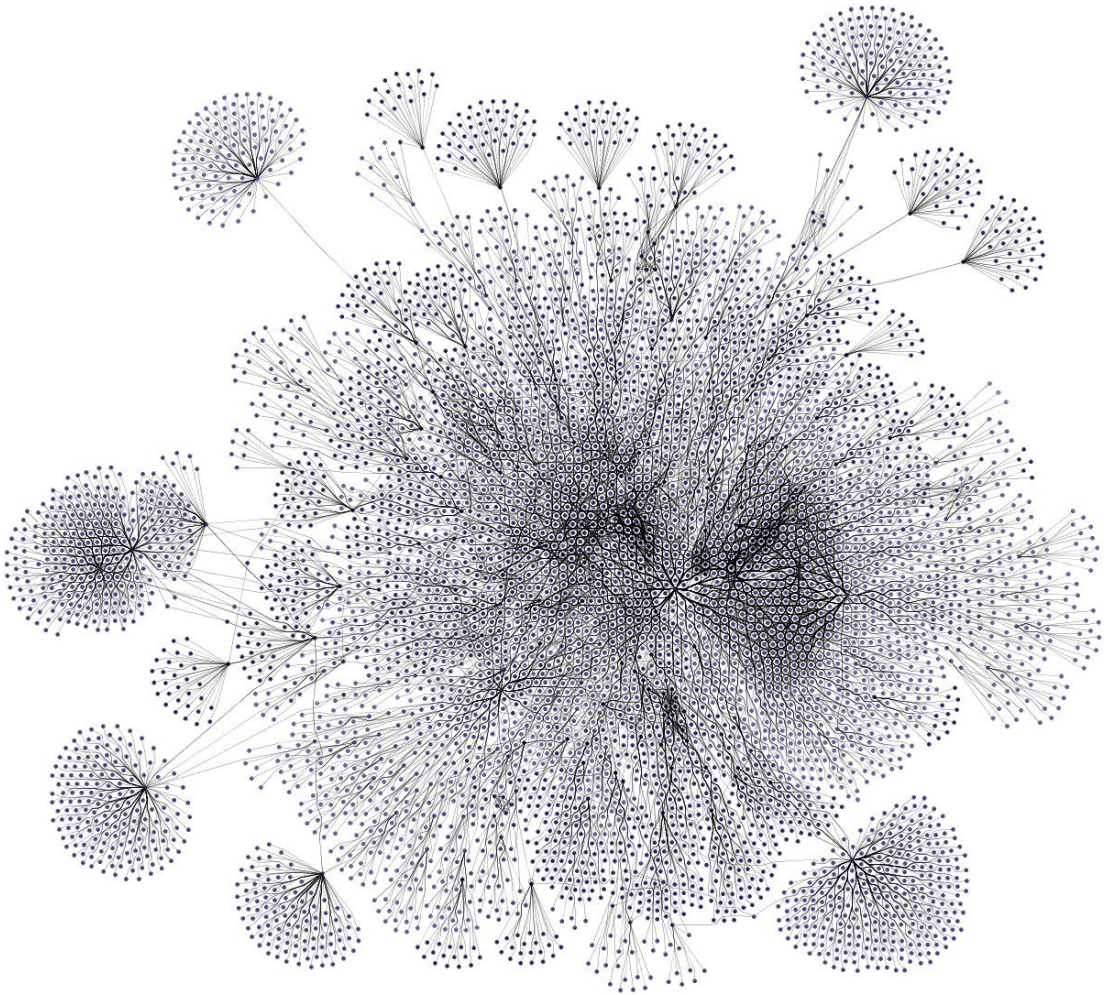
Jacob Hansson @jakewins

Outline

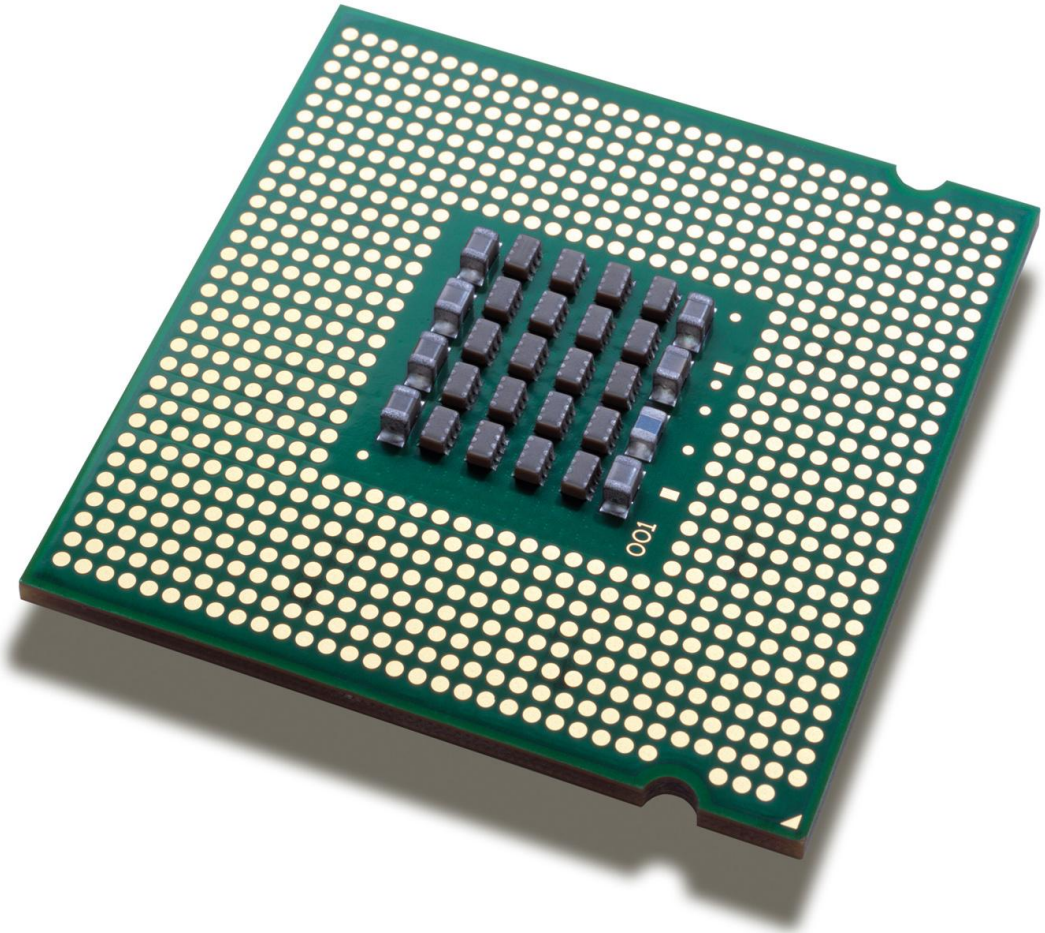
- Intro
- The Stack
- Measuring
- Architecture
- Patterns & APIs
- Q&A

Outline

- **Intro**
- The Stack
- Measuring
- Architecture
- Patterns & APIs
- Q&A



Jacob Hansson @jakewins



Jacob Hansson @jakewins

Jacob Hansson @jakewins

Why?

Premature optimization

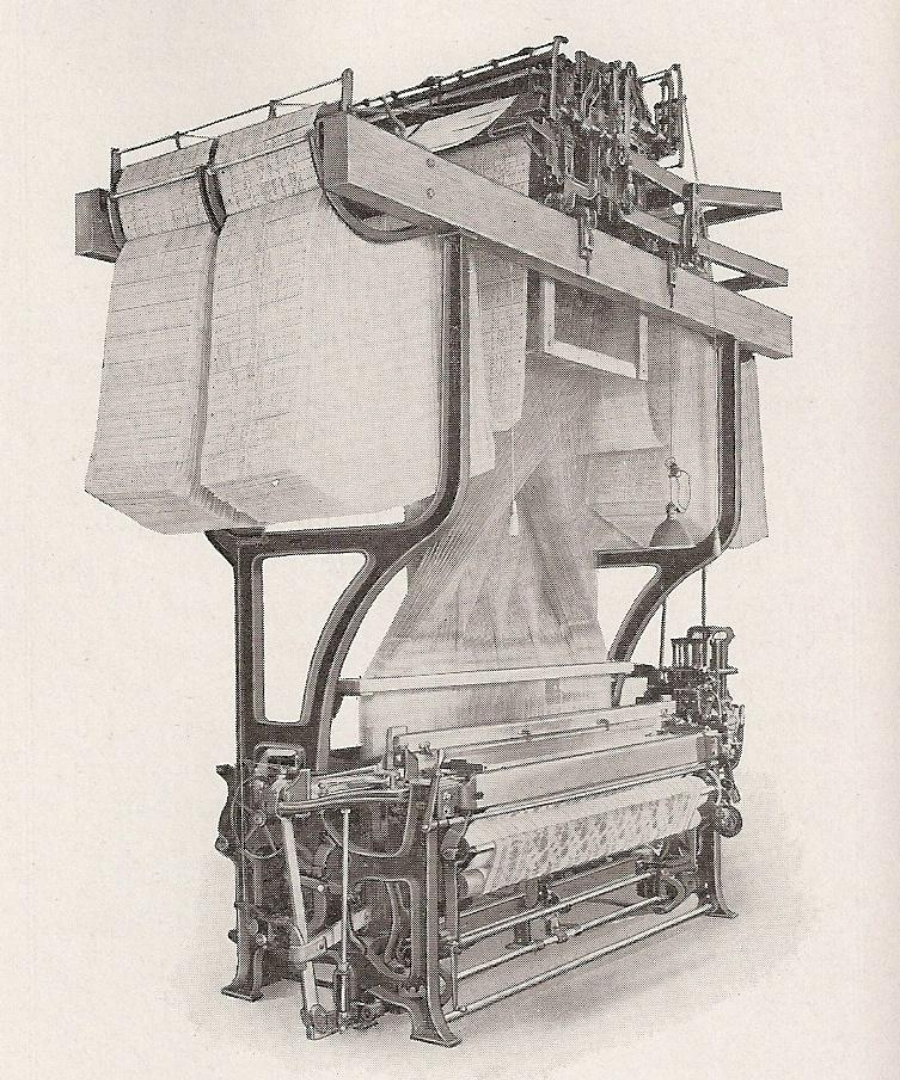
(is not a free ticket to write dumb code)

Key points

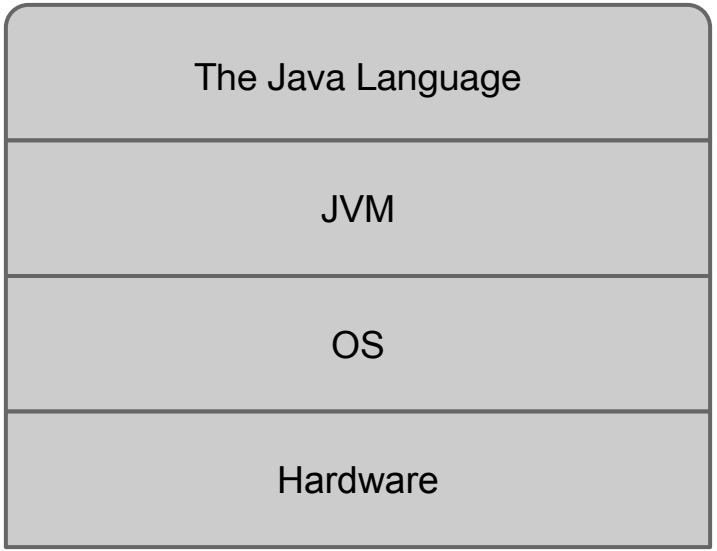
- Perform science
- Mechanical sympathy
- Performance comes from architecture
- Build an arsenal of known unknowns

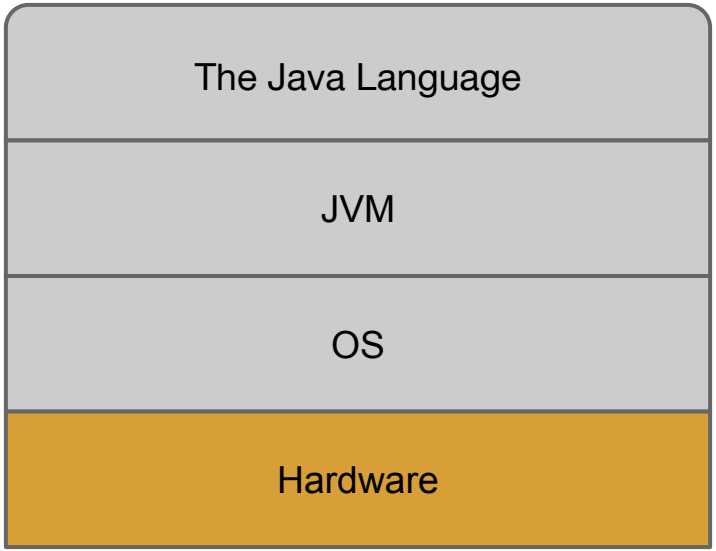
Outline

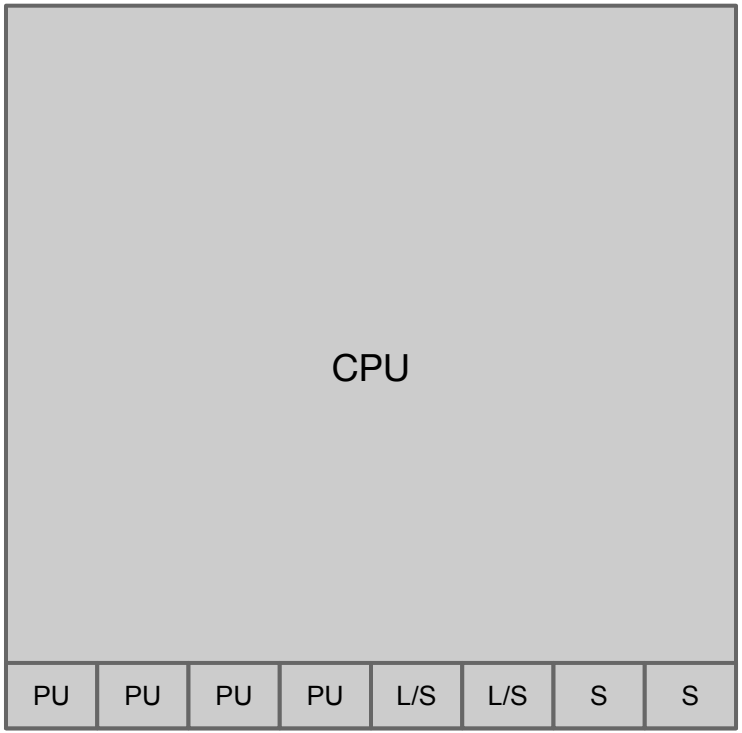
- Intro
- **The Stack**
- Measuring
- Architecture
- Patterns & APIs
- Q&A



Jacob Hansson @jakewins

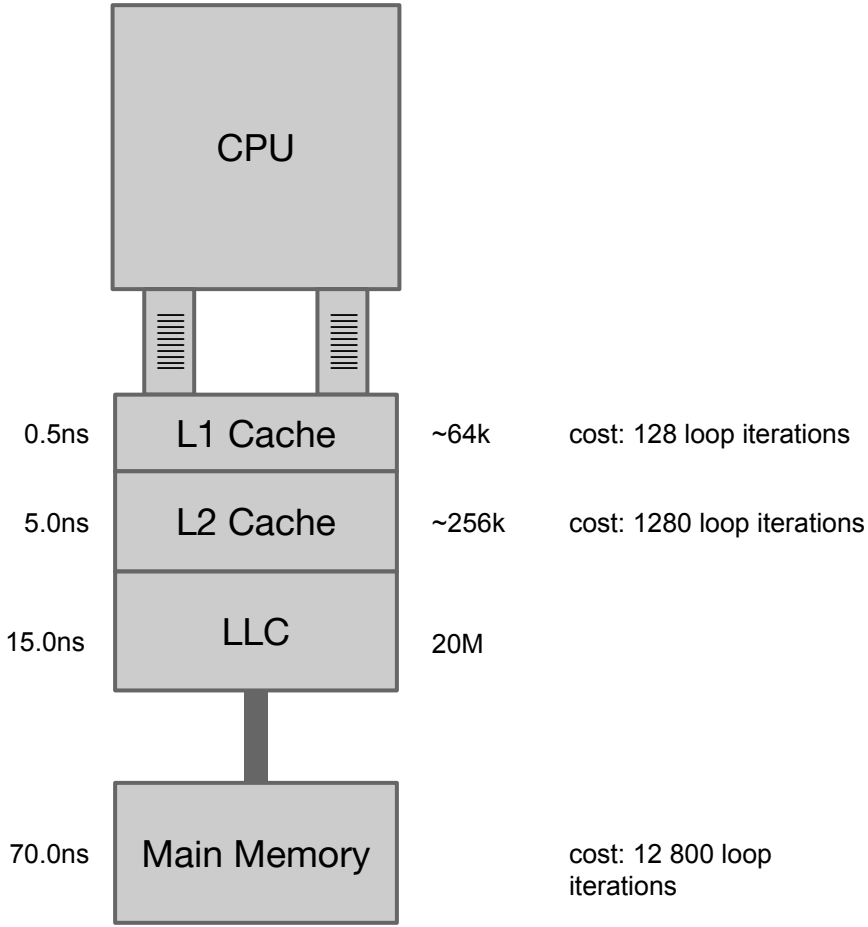

The Java Language
JVM
OS
Hardware

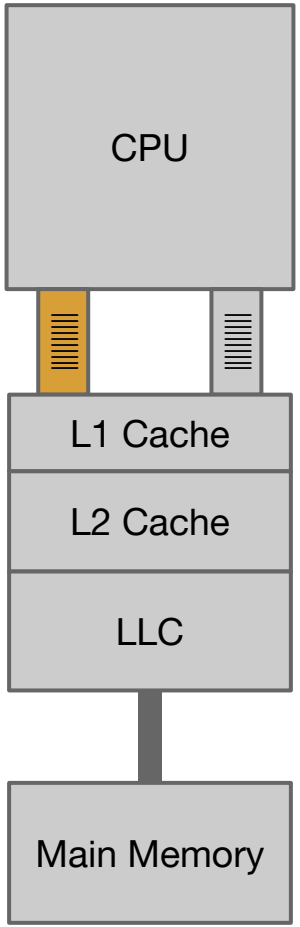

The Java Language
JVM
OS
Hardware



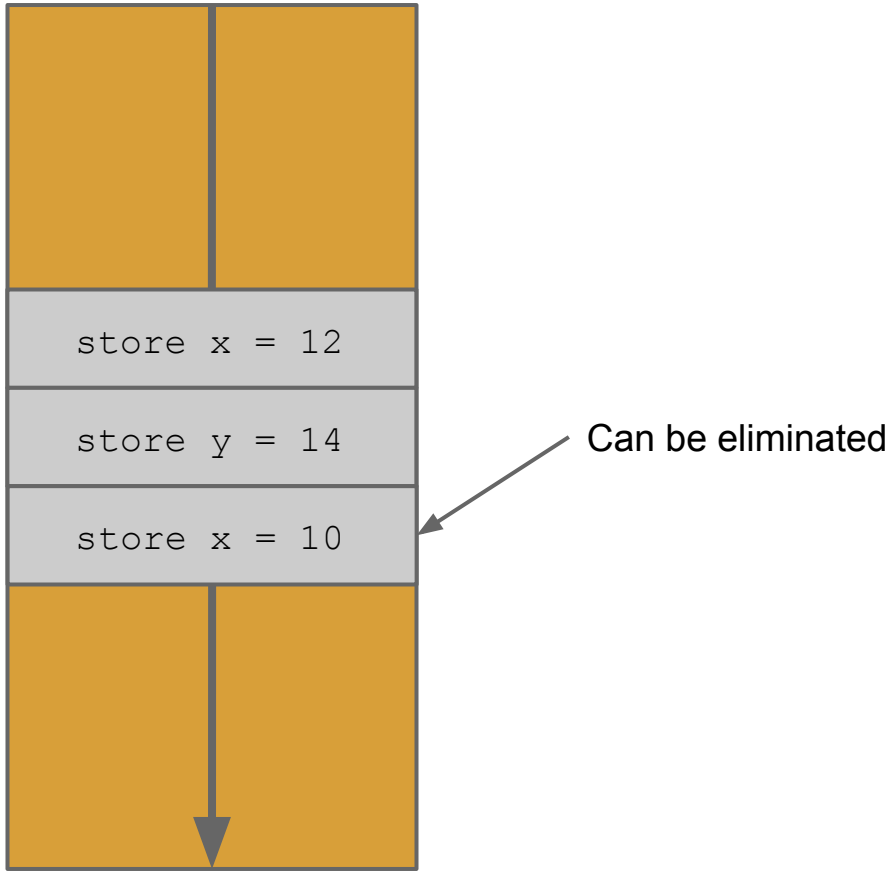
Jacob Hansson @jakewins

```
long sum = 0;
for( int i=0;
     i<vals.length;
     i++ )
{
  sum += vals[i];
}
```

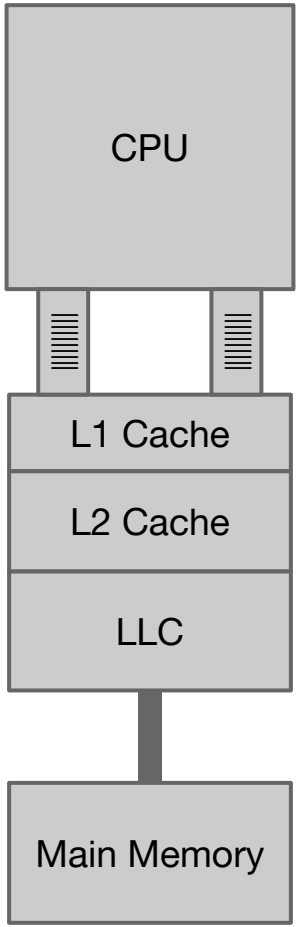




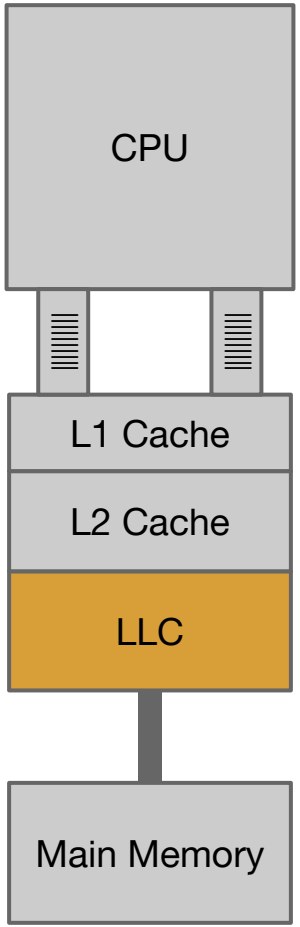
Jacob Hansson @jakewins



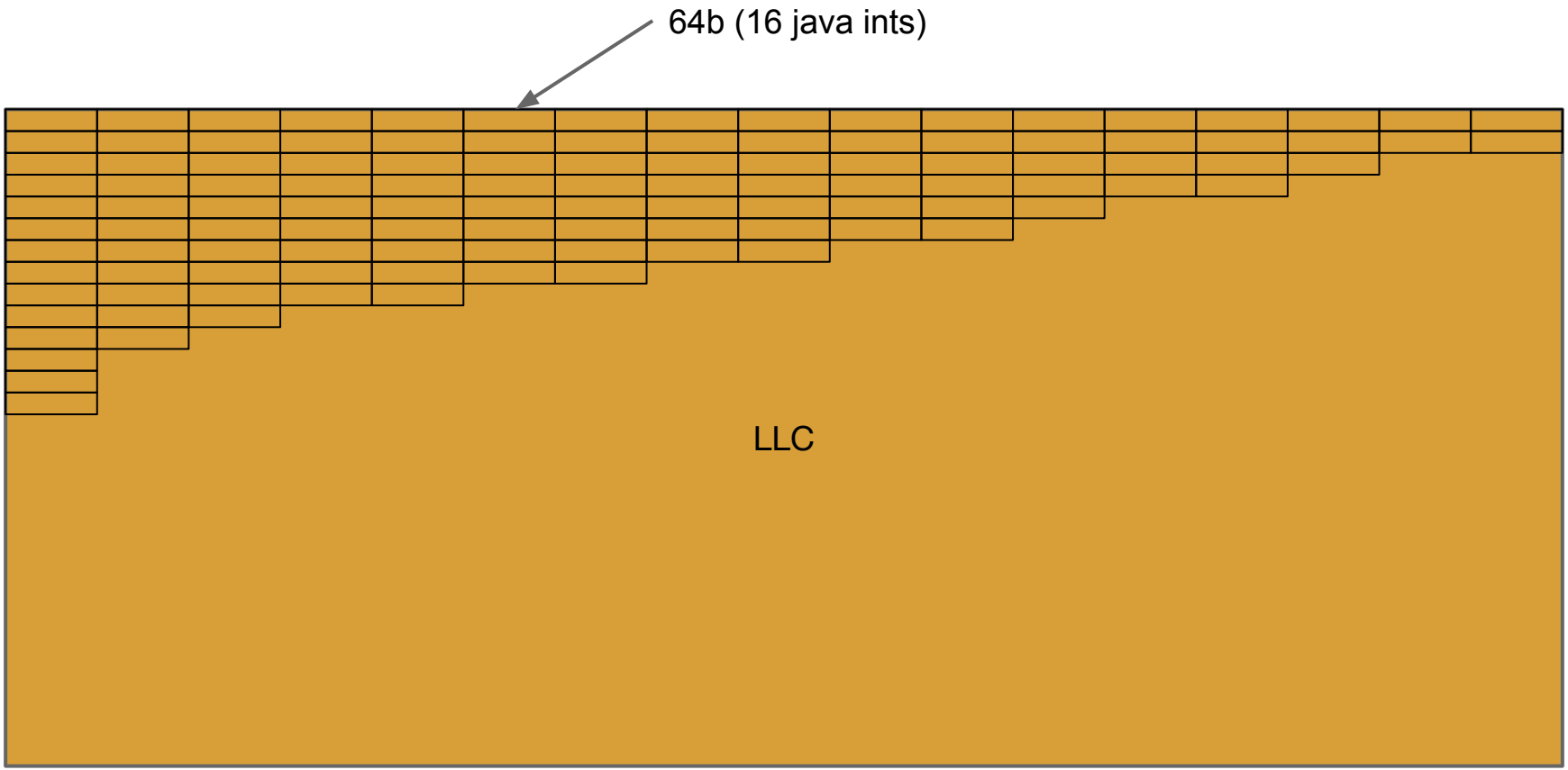
Jacob Hansson @jakewins



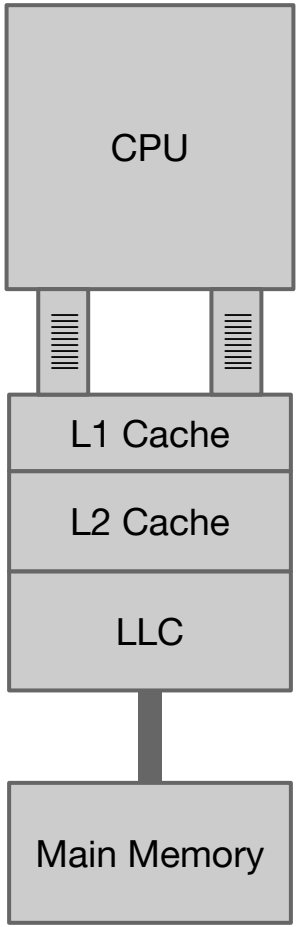
Jacob Hansson @jakewins



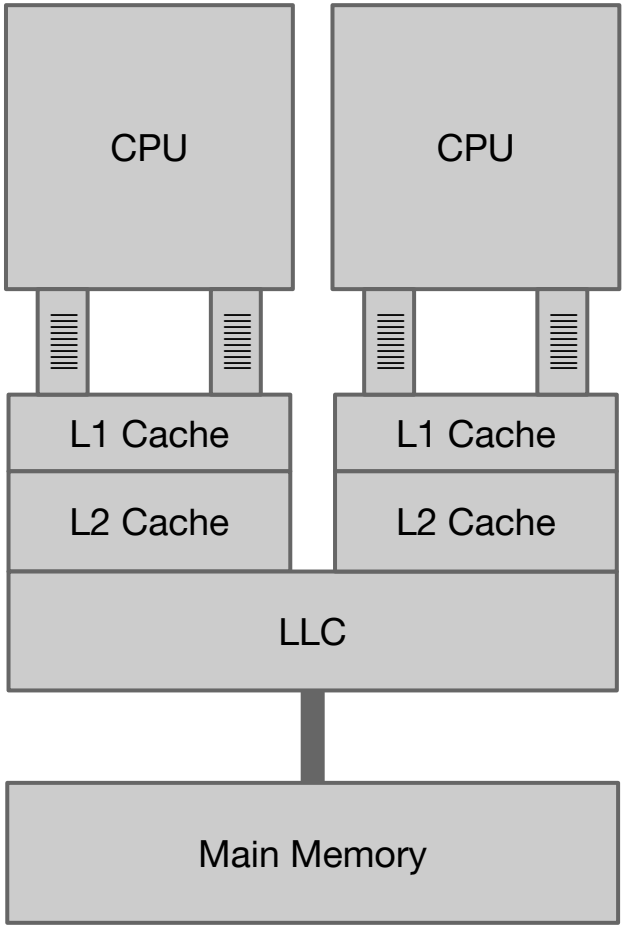
Jacob Hansson @jakewins



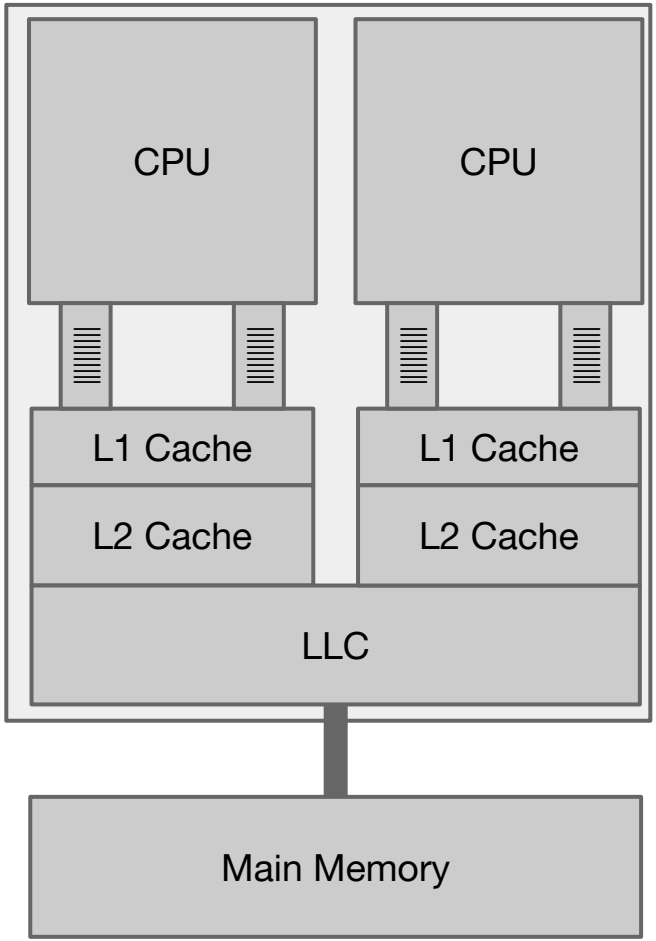
Jacob Hansson @jakewins



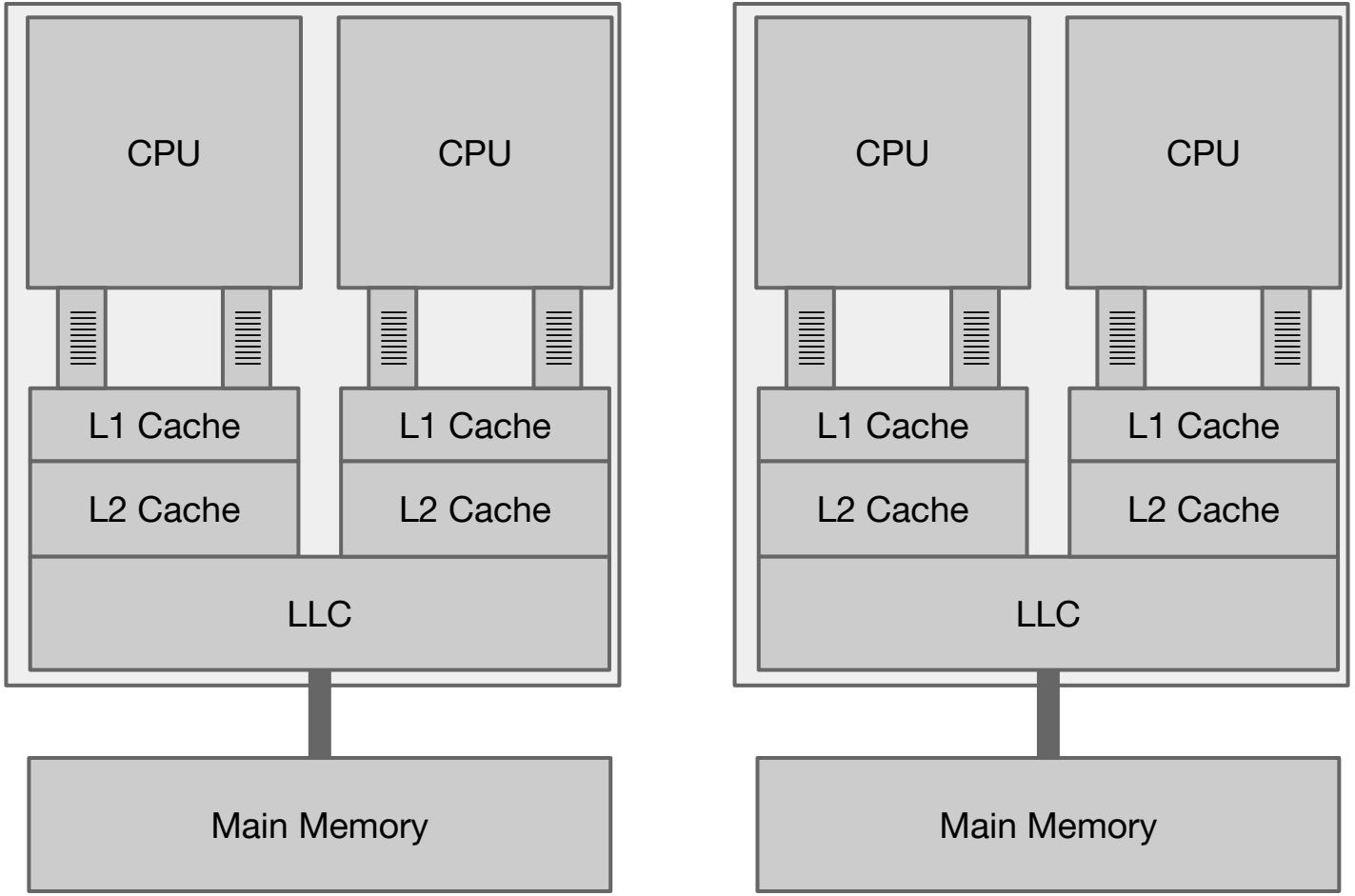
Jacob Hansson @jakewins



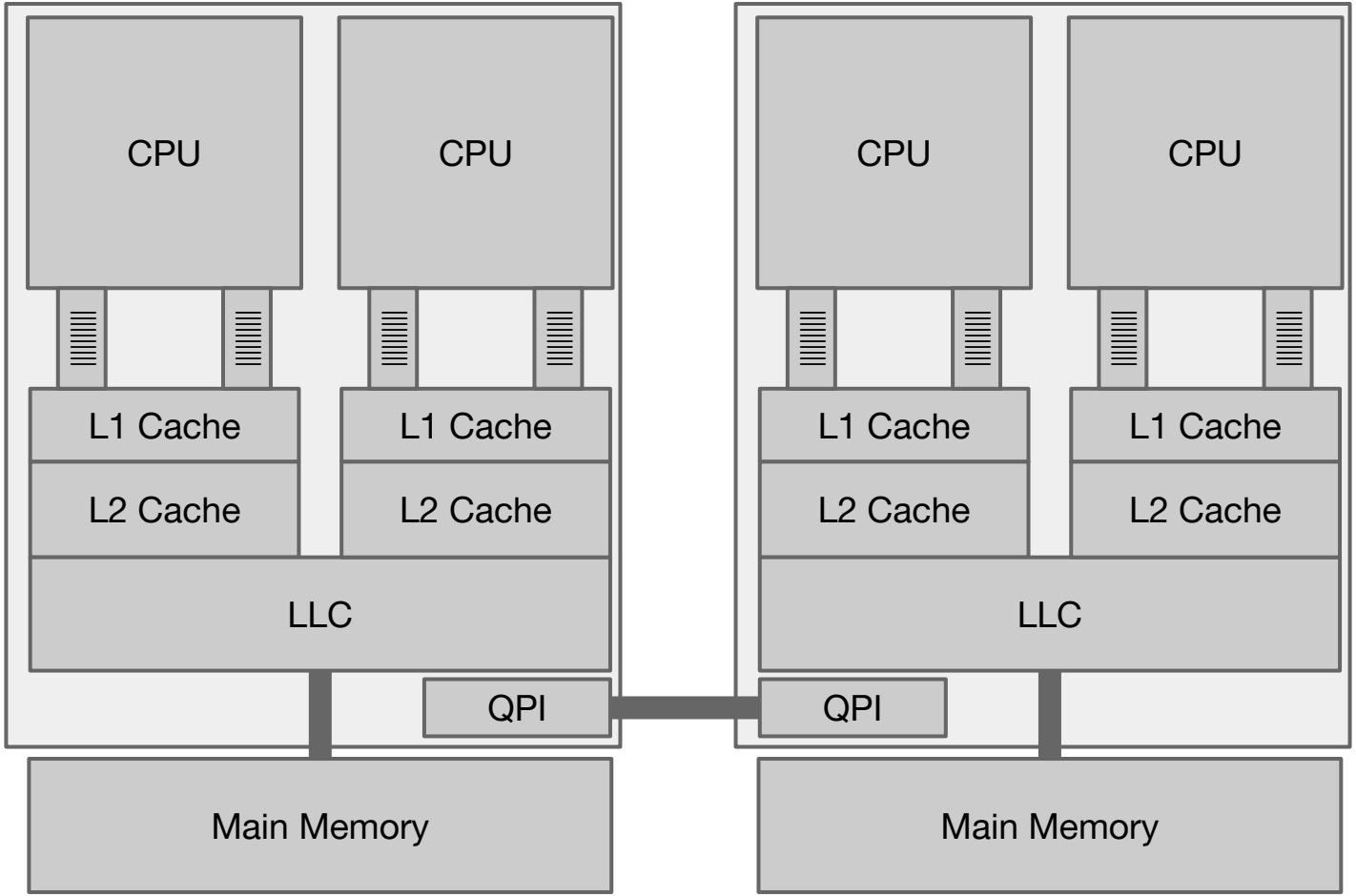
Jacob Hansson @jakewins



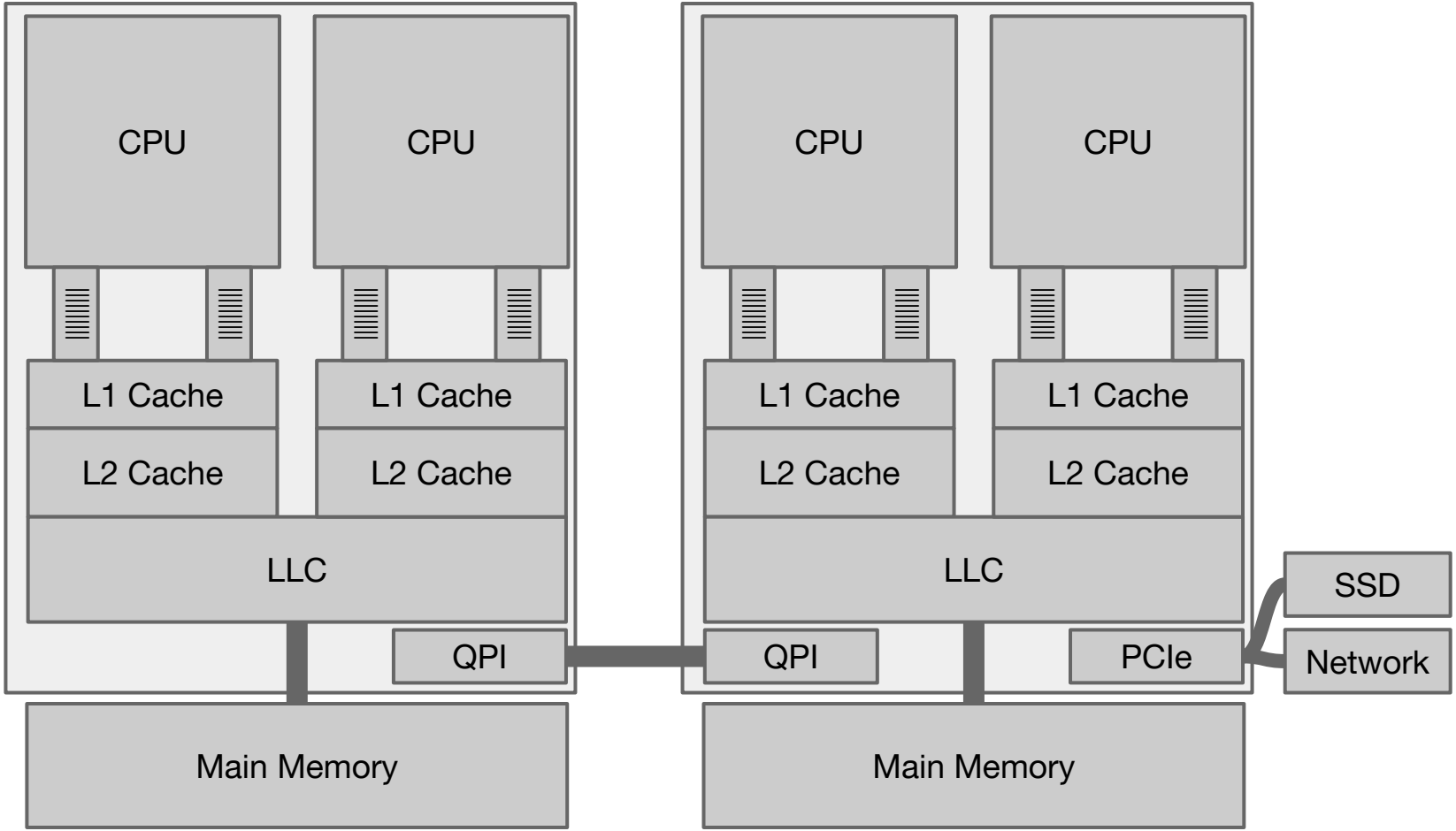
Jacob Hansson @jakewins



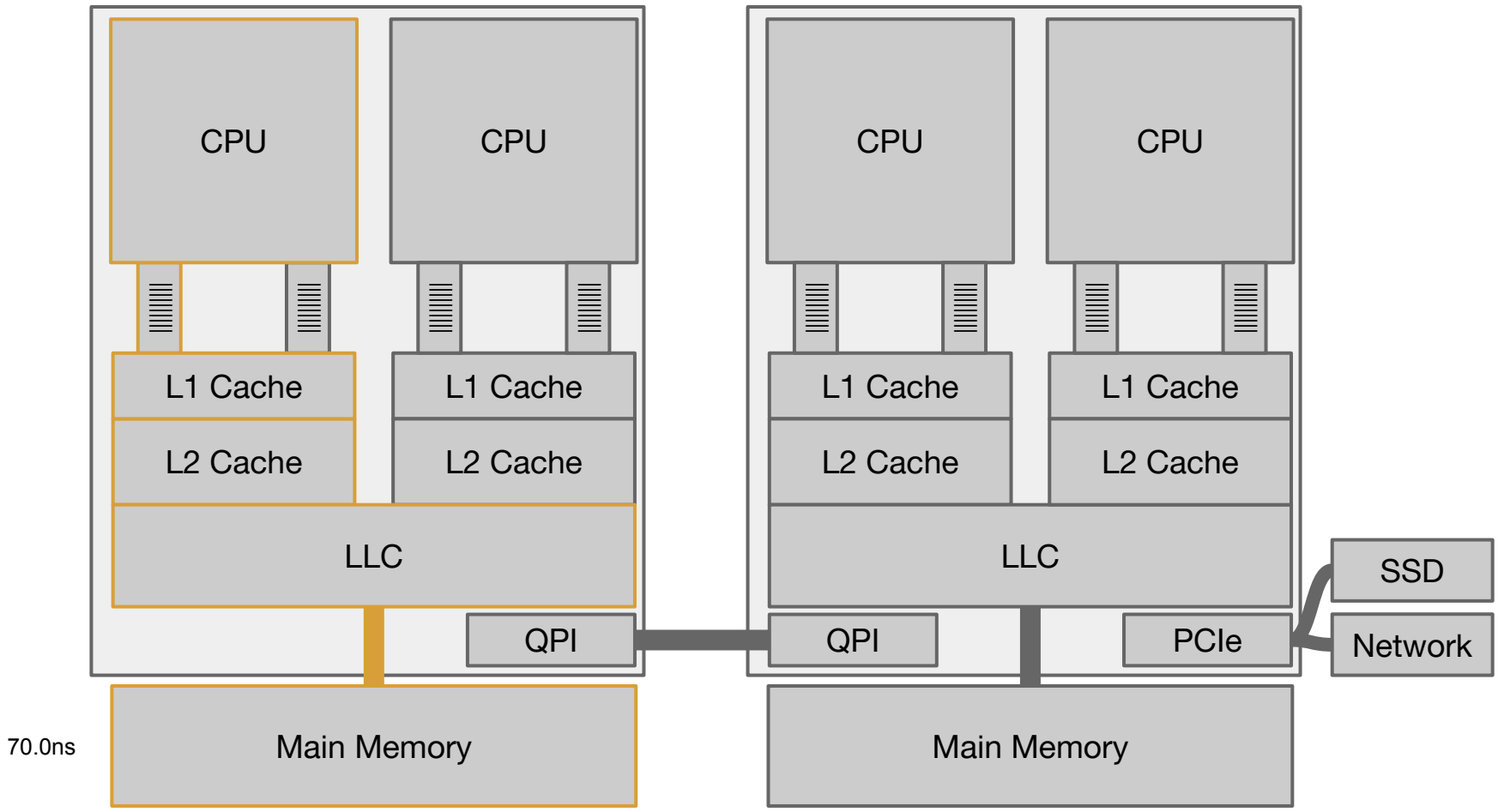
Jacob Hansson @jakewins



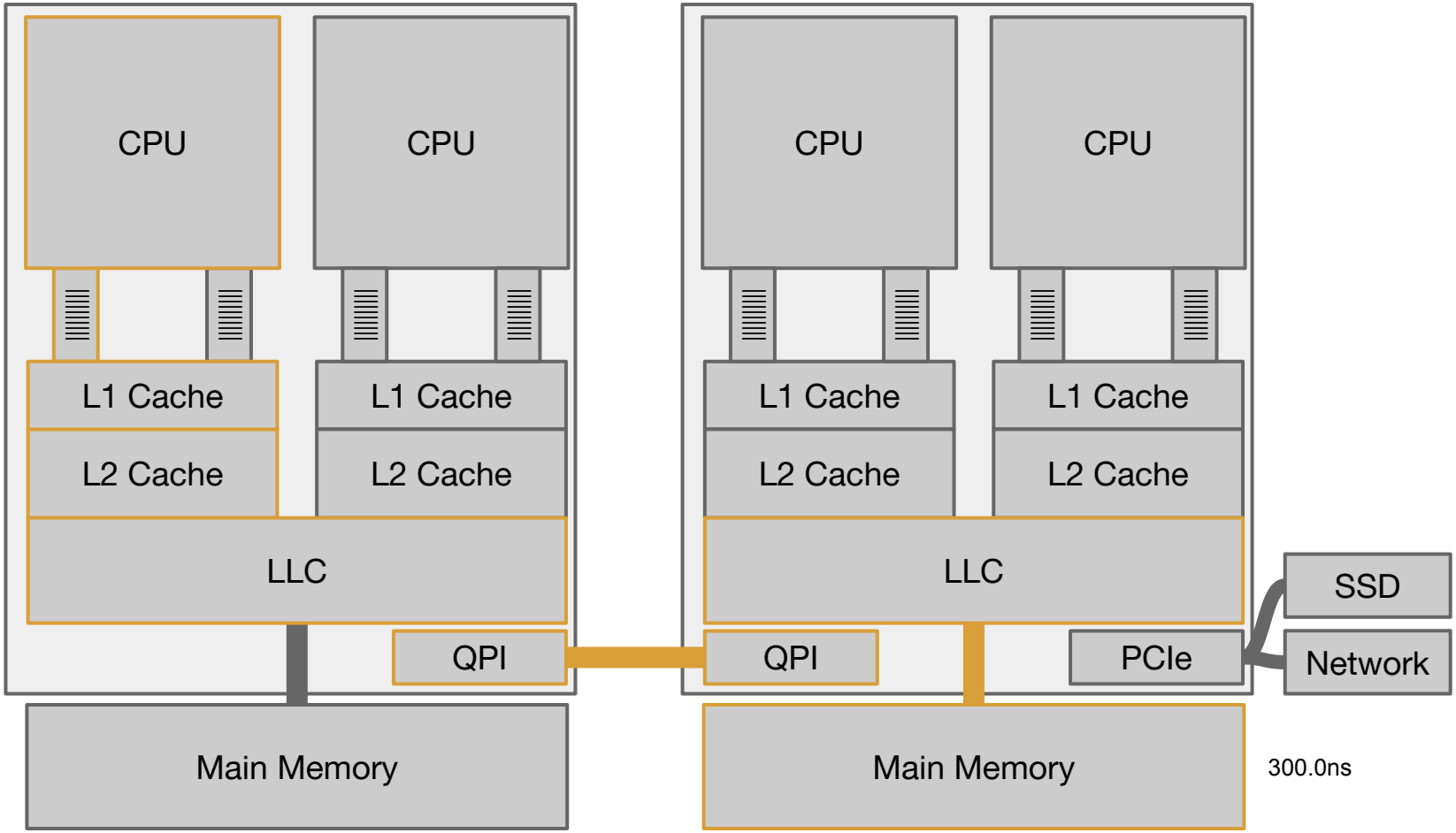
Jacob Hansson @jakewins



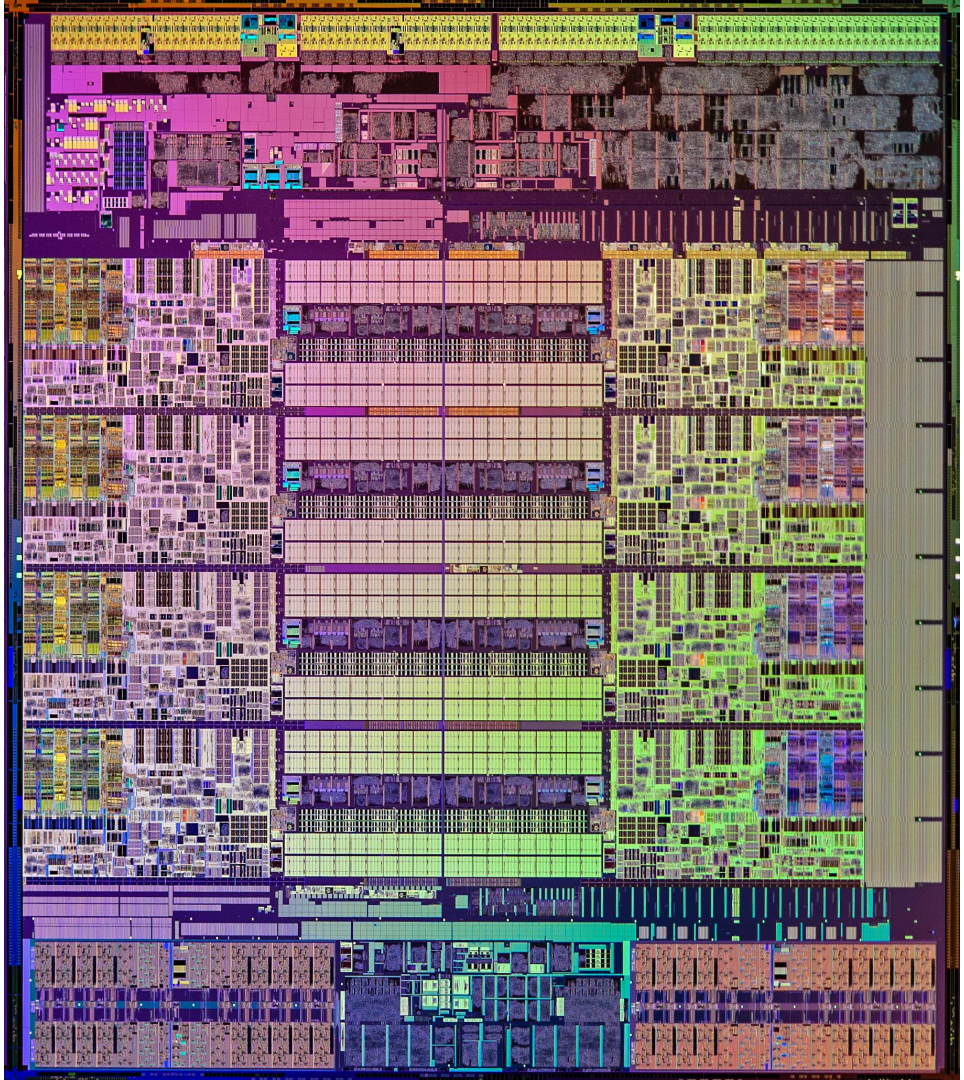
Jacob Hansson @jakewins



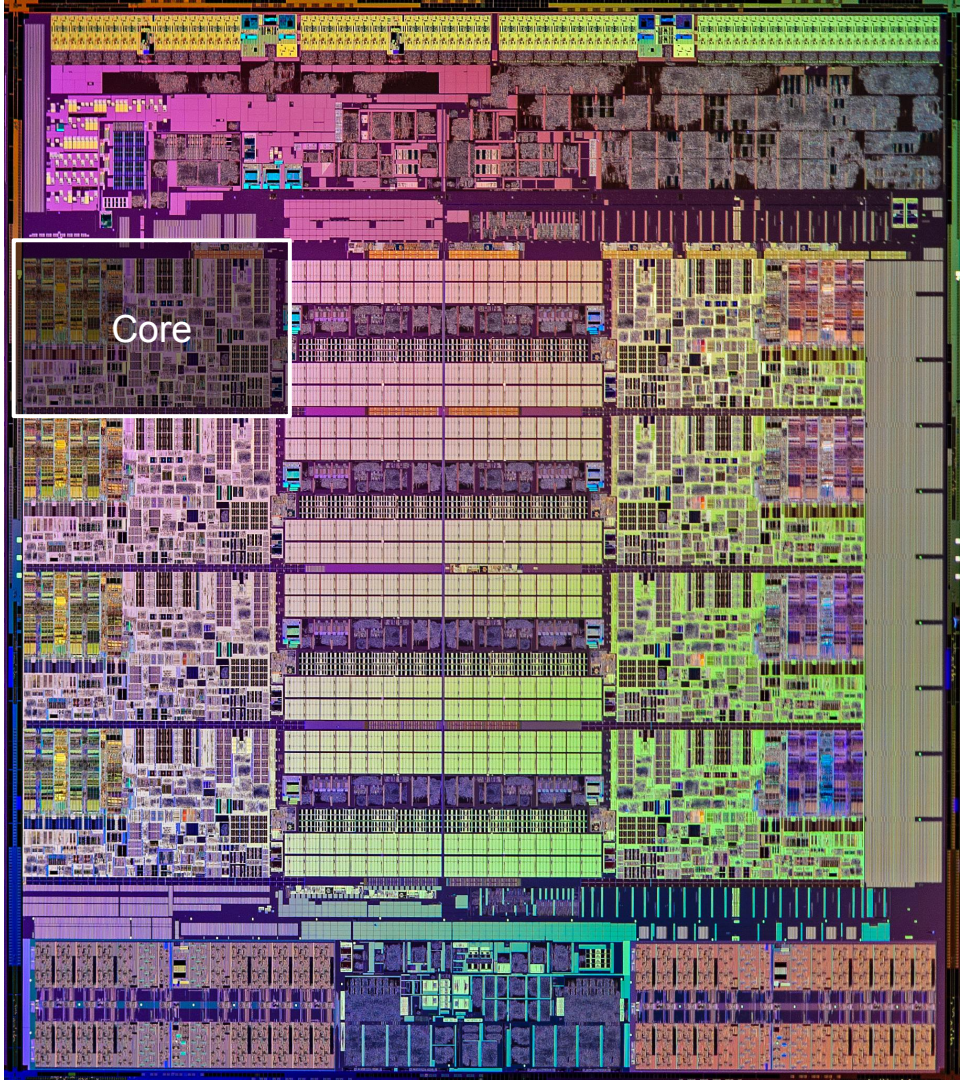
Jacob Hansson @jakewins



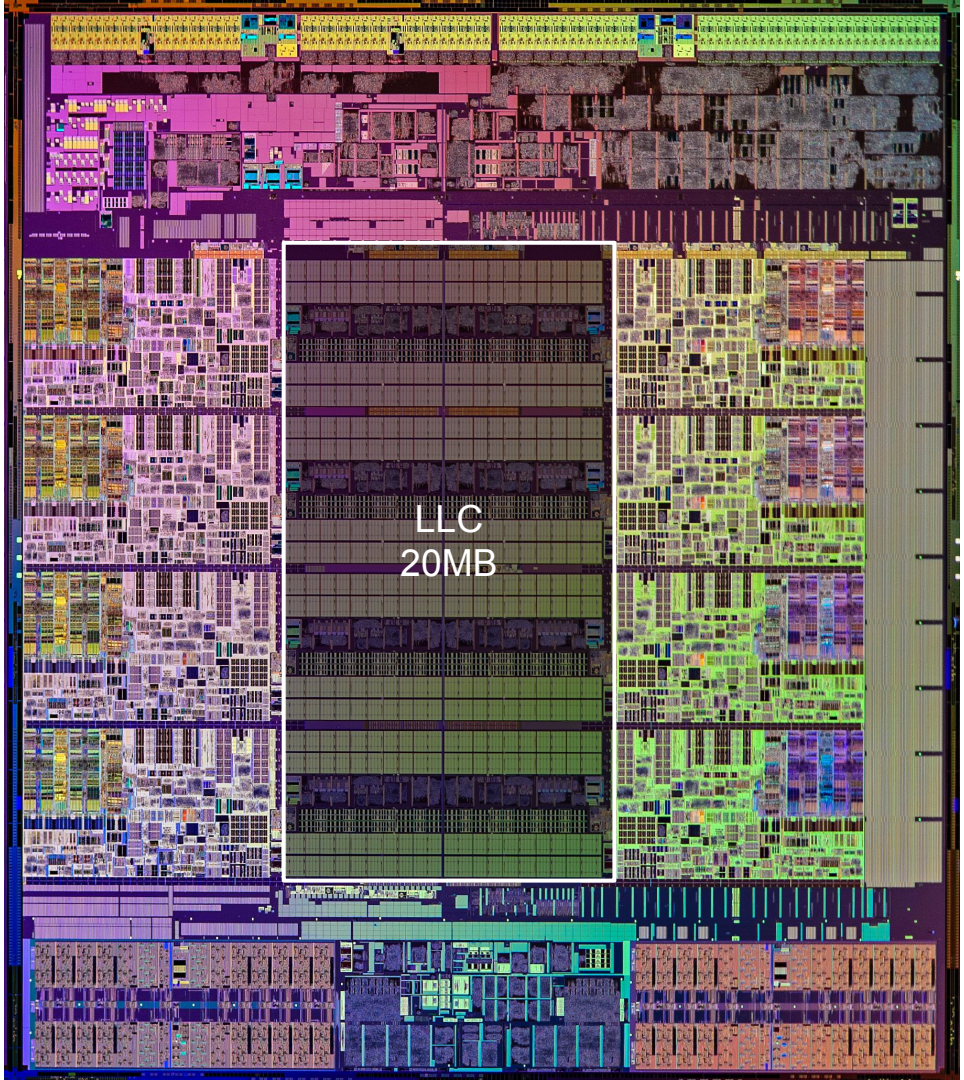
Jacob Hansson @jakewins



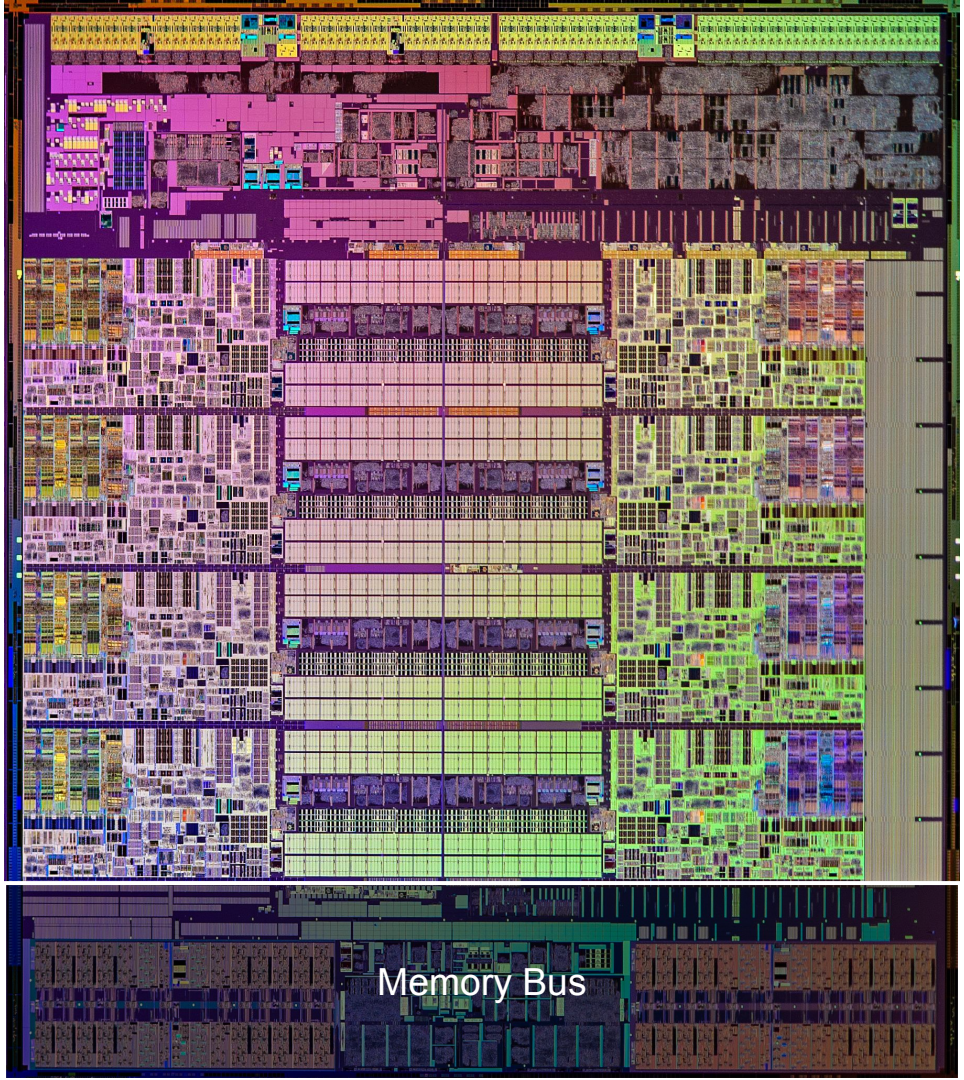
Jacob Hansson @jakewins



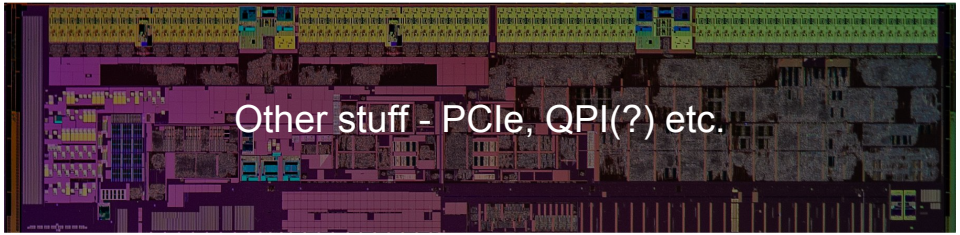
Jacob Hansson @jakewins

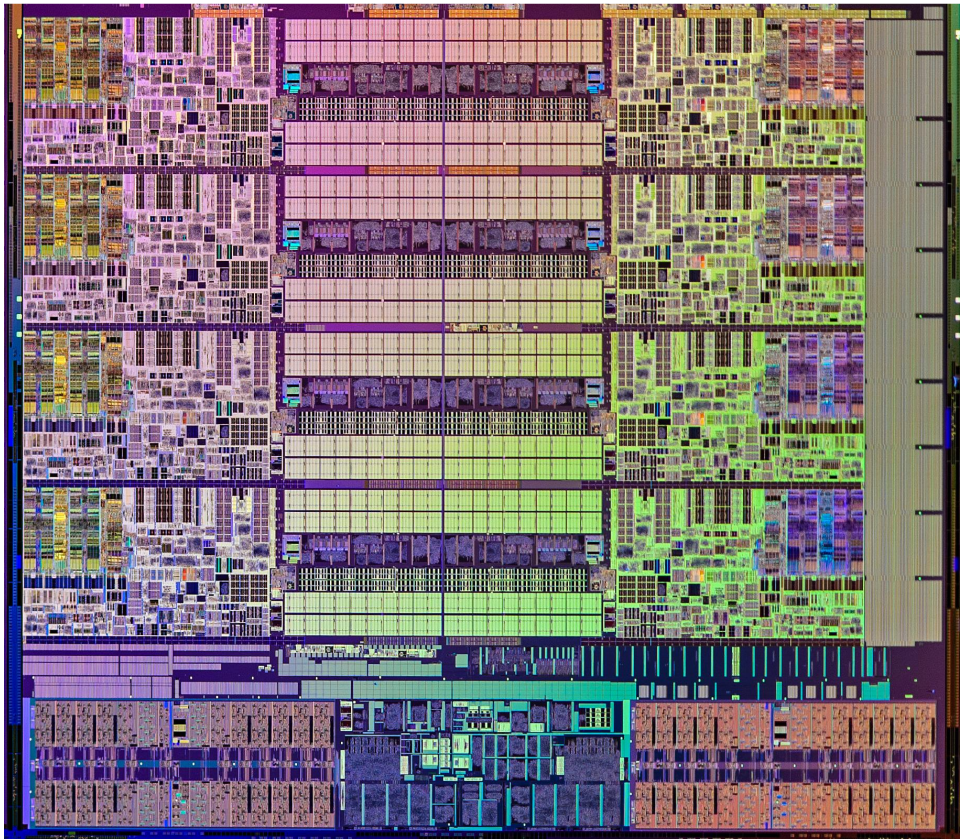


Jacob Hansson @jakewins

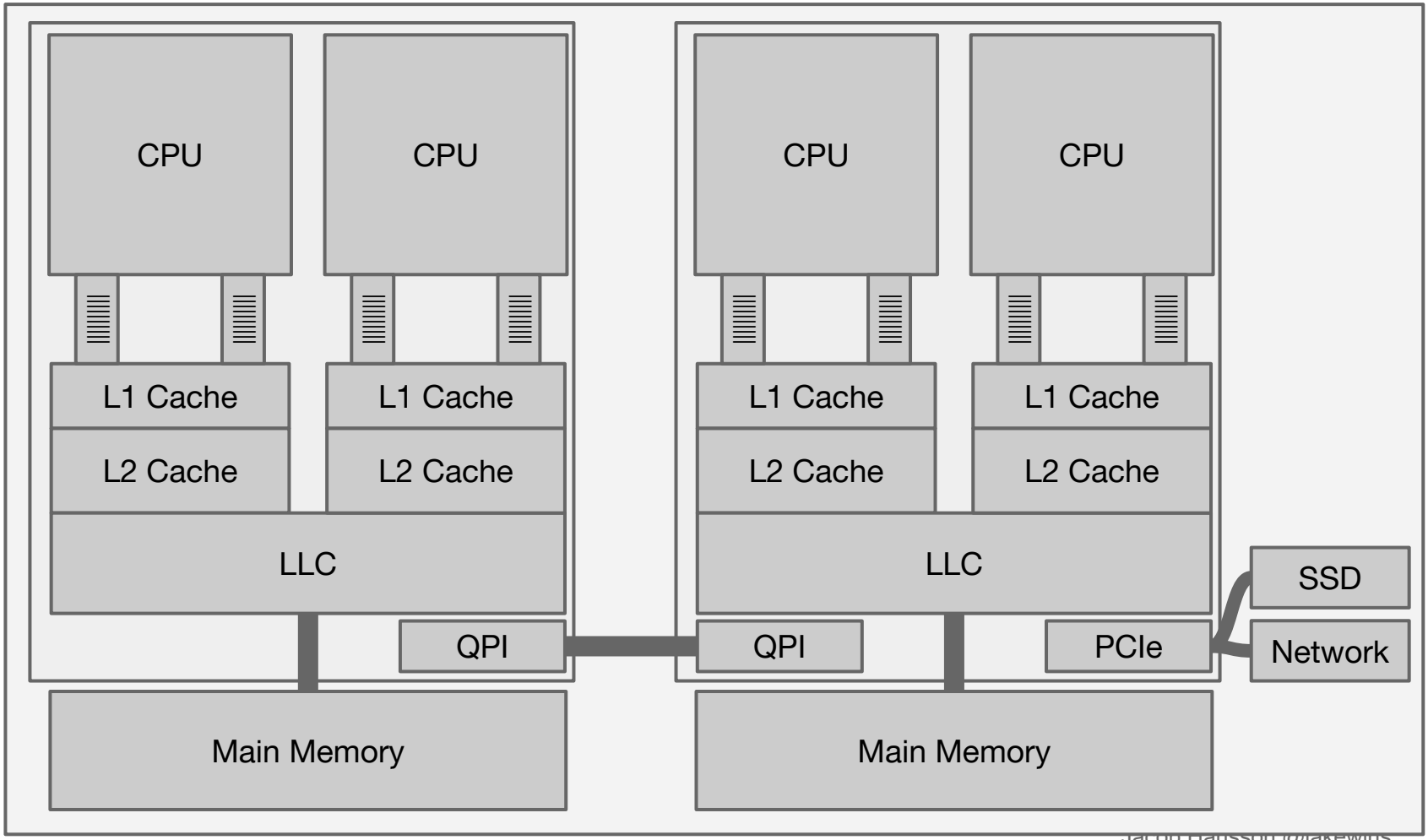


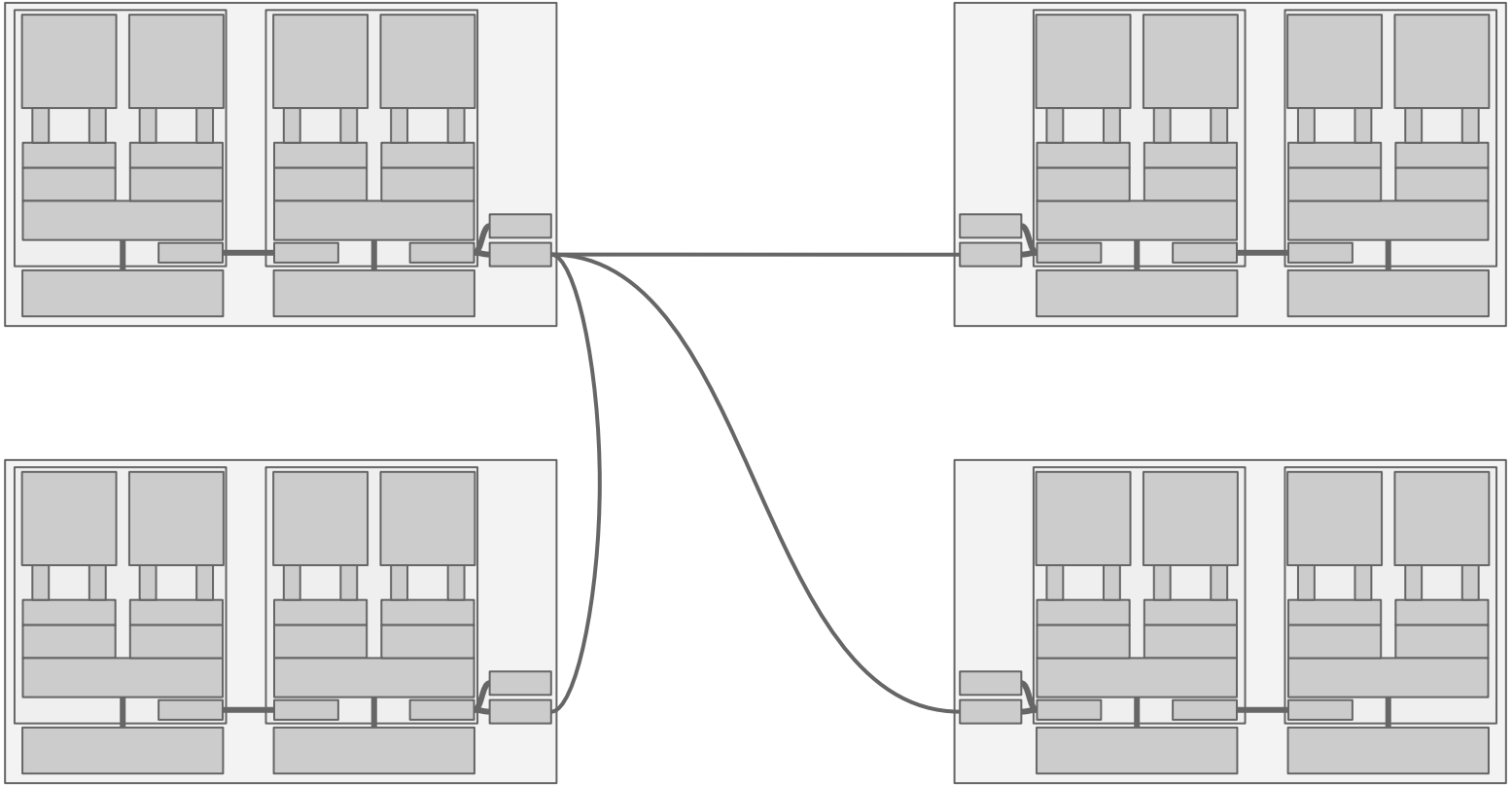
Jacob Hansson @jakewins



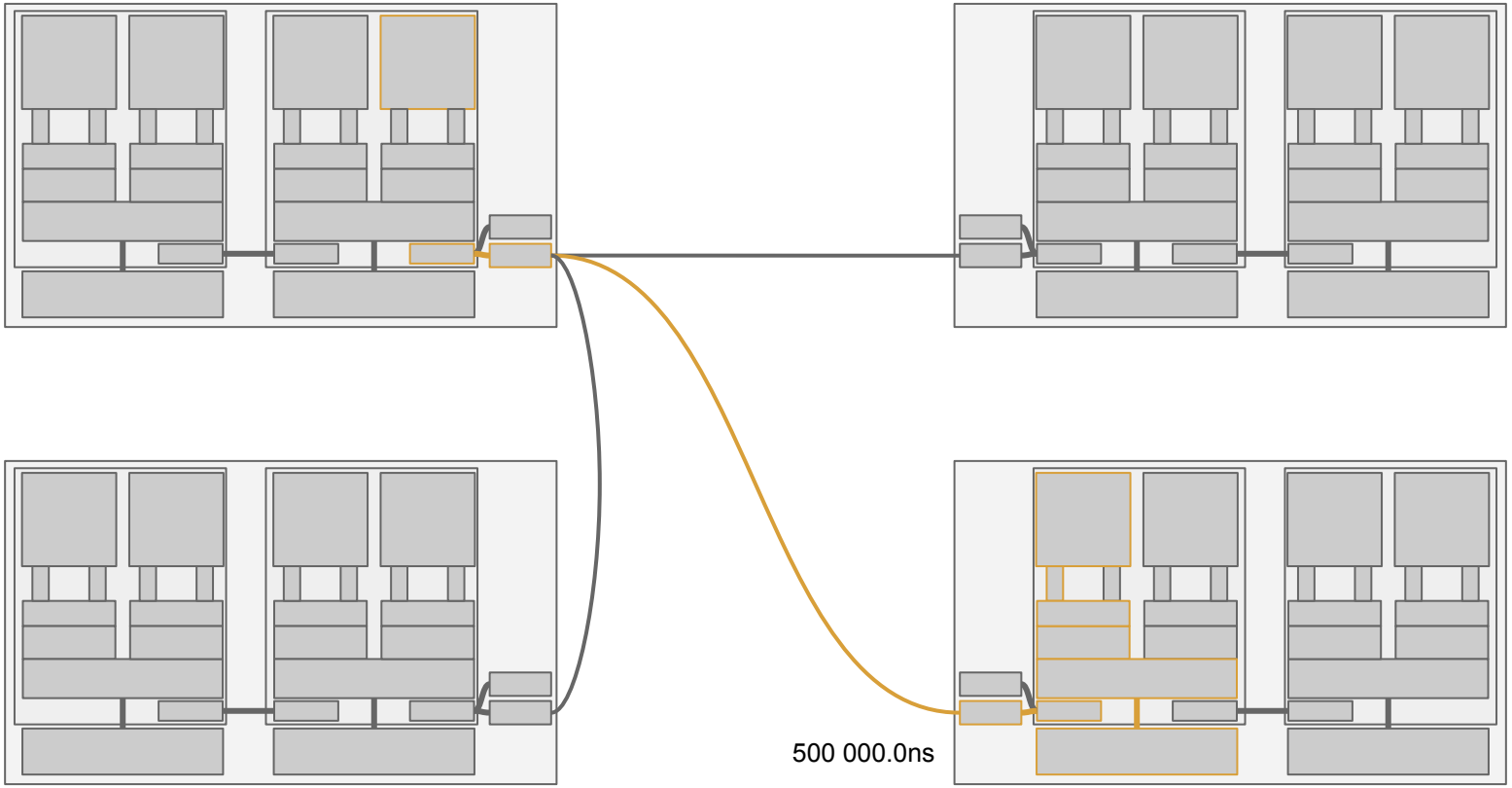


Jacob Hansson @jakewins

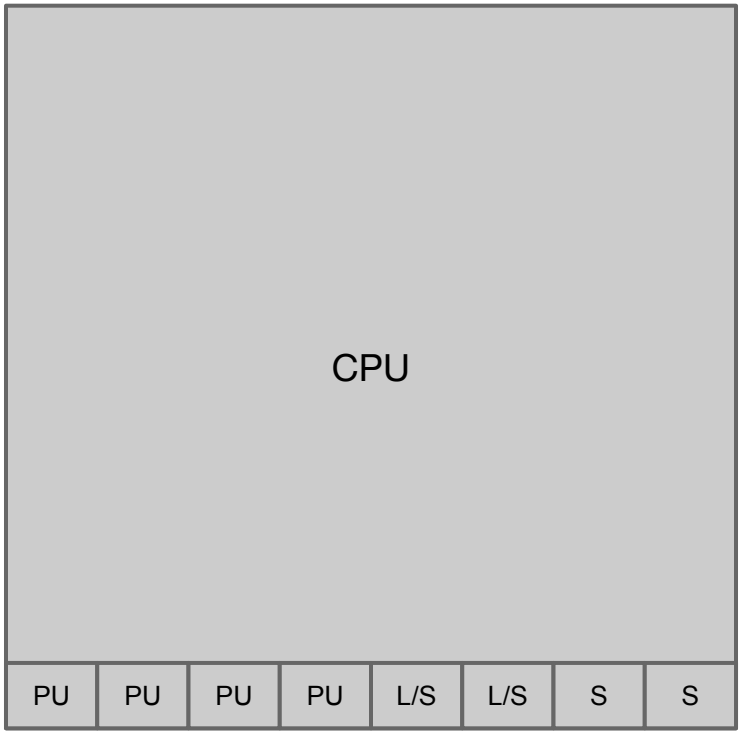

CPU
CPU
CPU
CPU
L1 Cache
L1 Cache
L1 Cache
L1 Cache
L2 Cache
L2 Cache
L2 Cache
L2 Cache
LLC
LLC
SSD
QPI
QPI
PCIe
Network
Main Memory
Main Memory



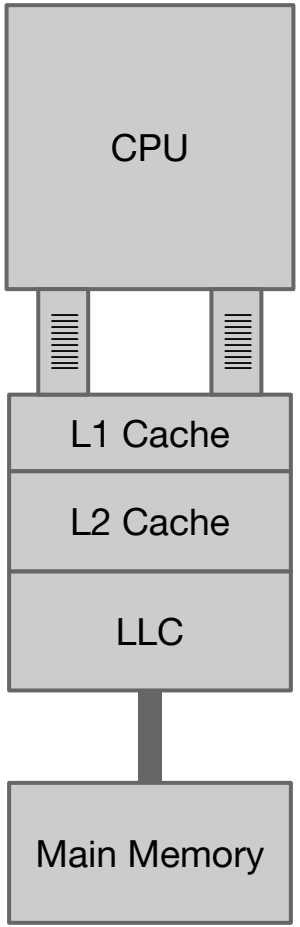
Jacob Hansson @jakewins



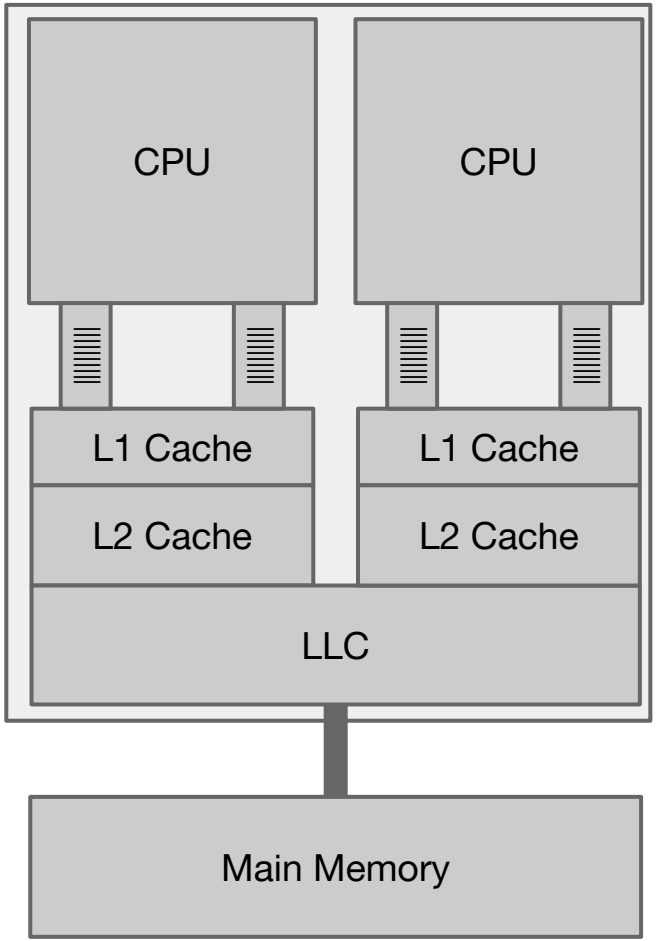
Jacob Hansson @jakewins



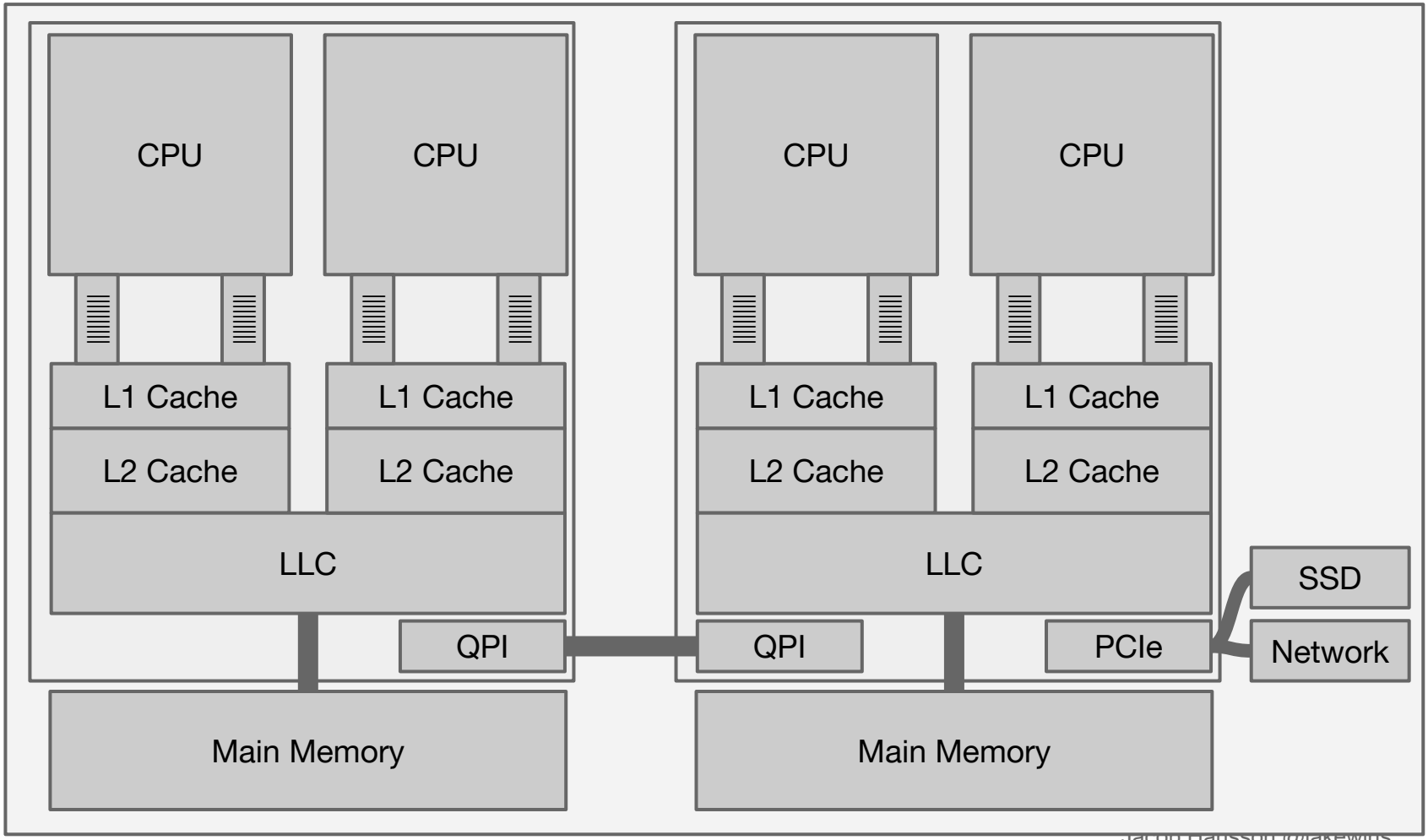
Jacob Hansson @jakewins

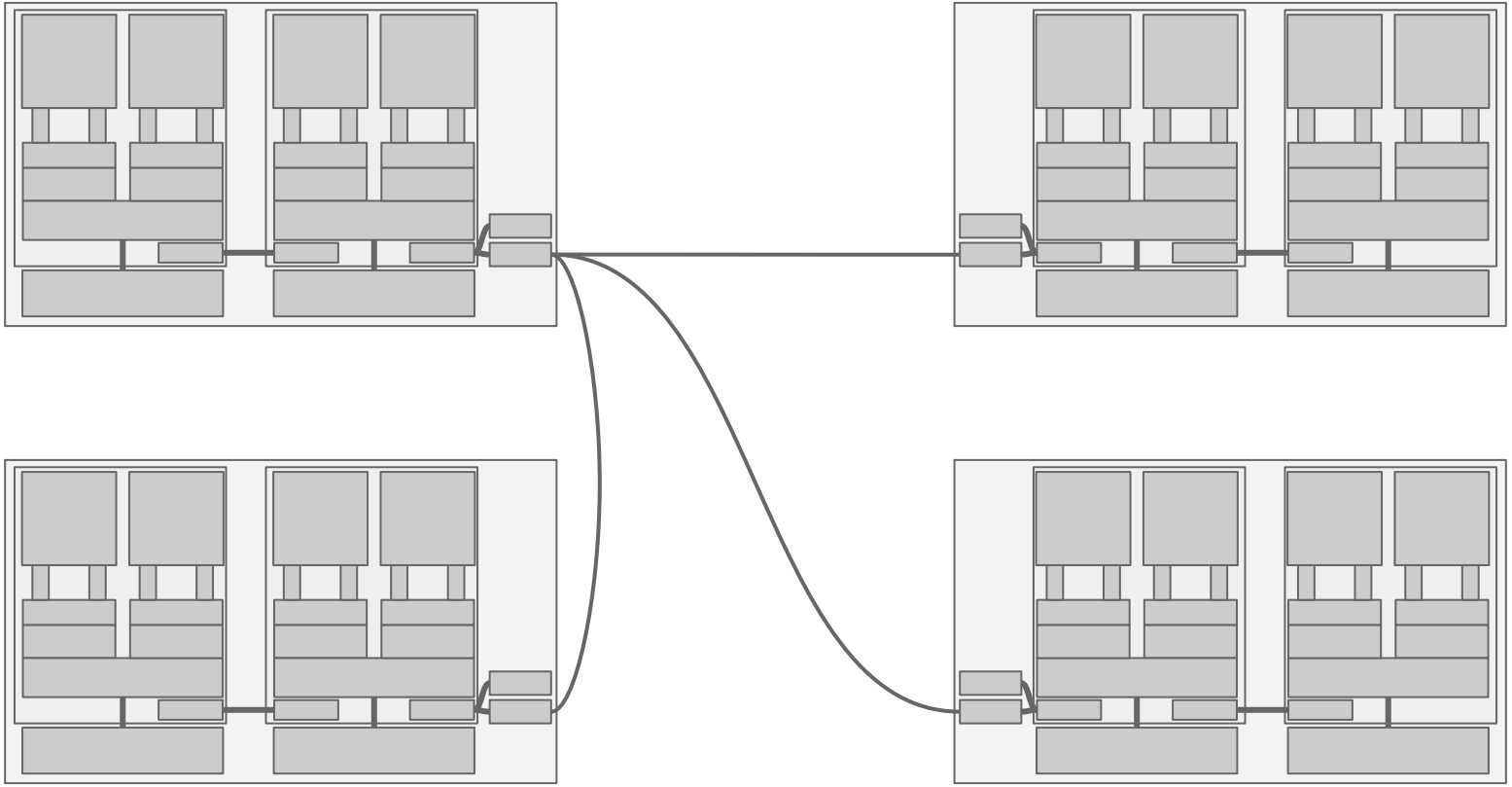


Jacob Hansson @jakewins



Jacob Hansson @jakewins

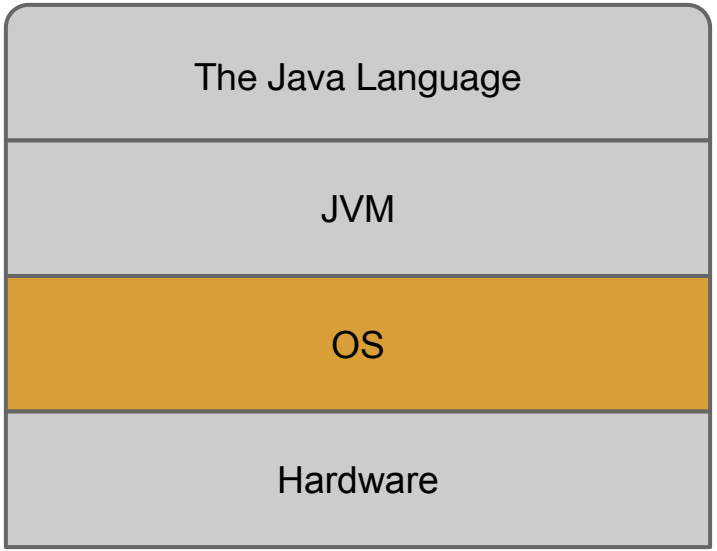

CPU
CPU
CPU
CPU
L1 Cache
L1 Cache
L1 Cache
L1 Cache
L2 Cache
L2 Cache
L2 Cache
L2 Cache
LLC
LLC
SSD
QPI
QPI
PCIe
Network
Main Memory
Main Memory



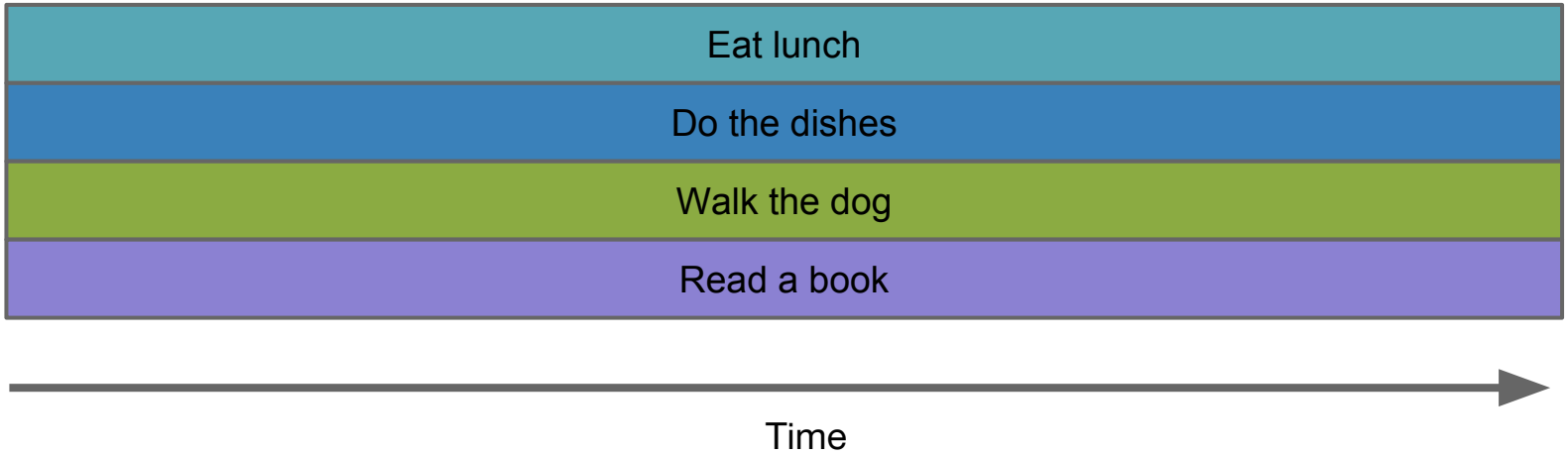
Jacob Hansson @jakewins

Key points

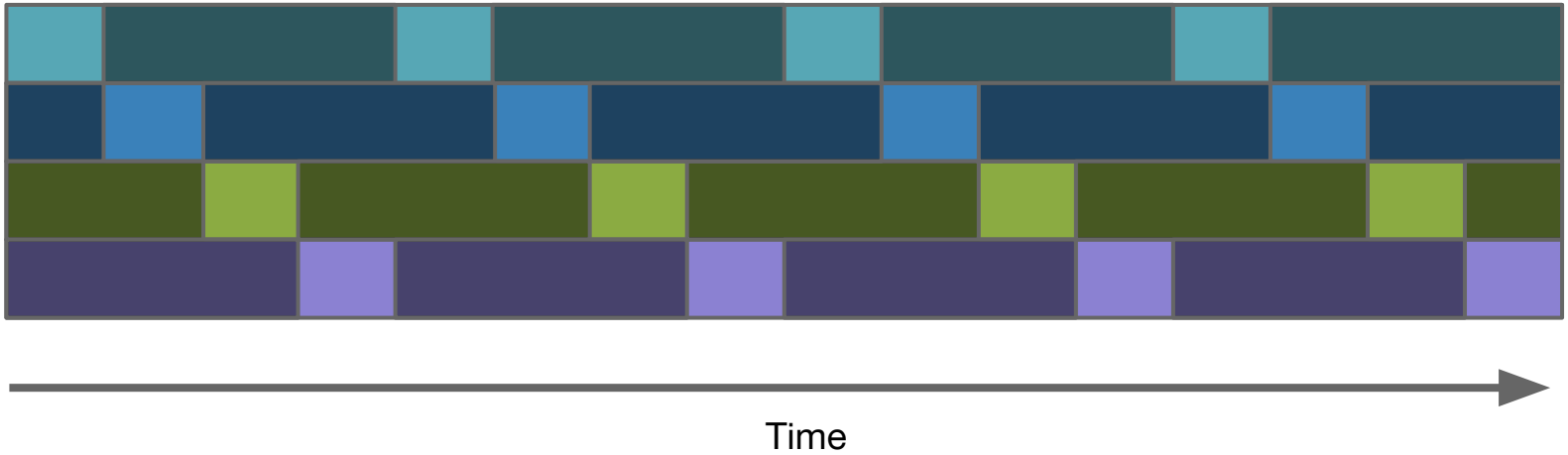
- Modern hardware is fractal
- Concurrent software is distributed system
- Memory is major choke point

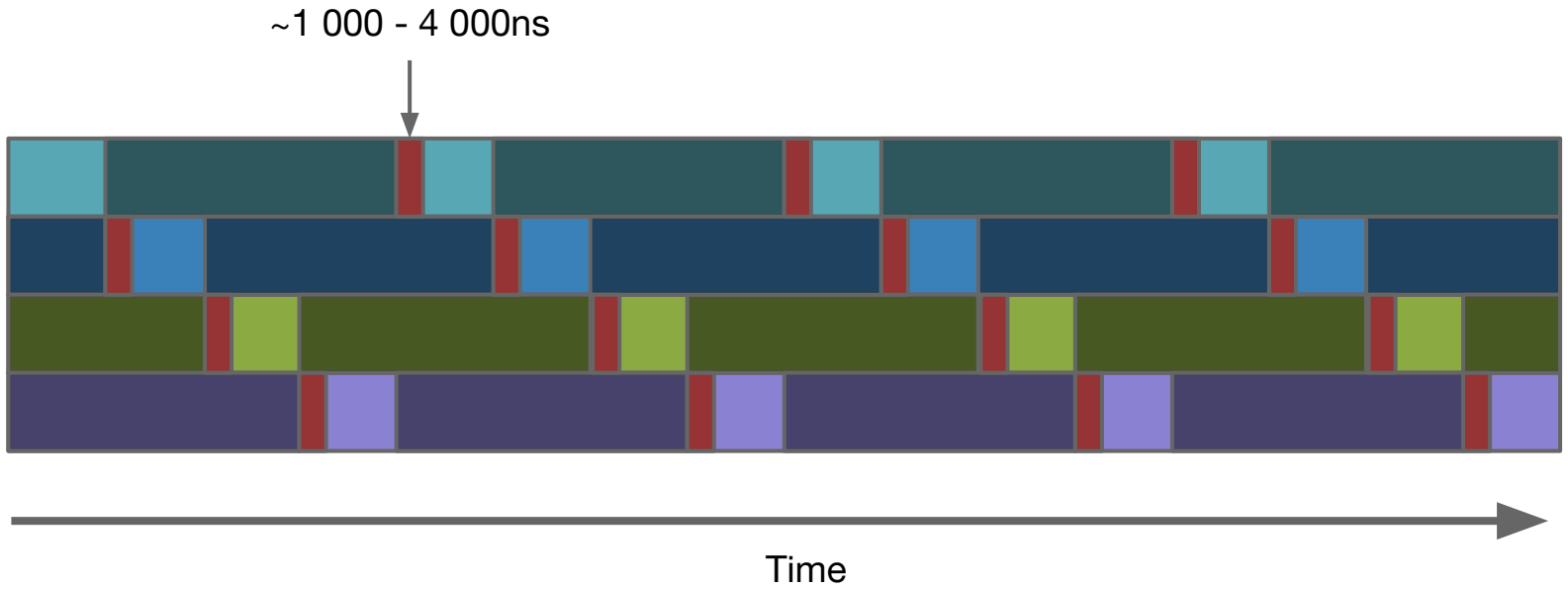

The Java Language
JVM
OS
Hardware

Concurrency!

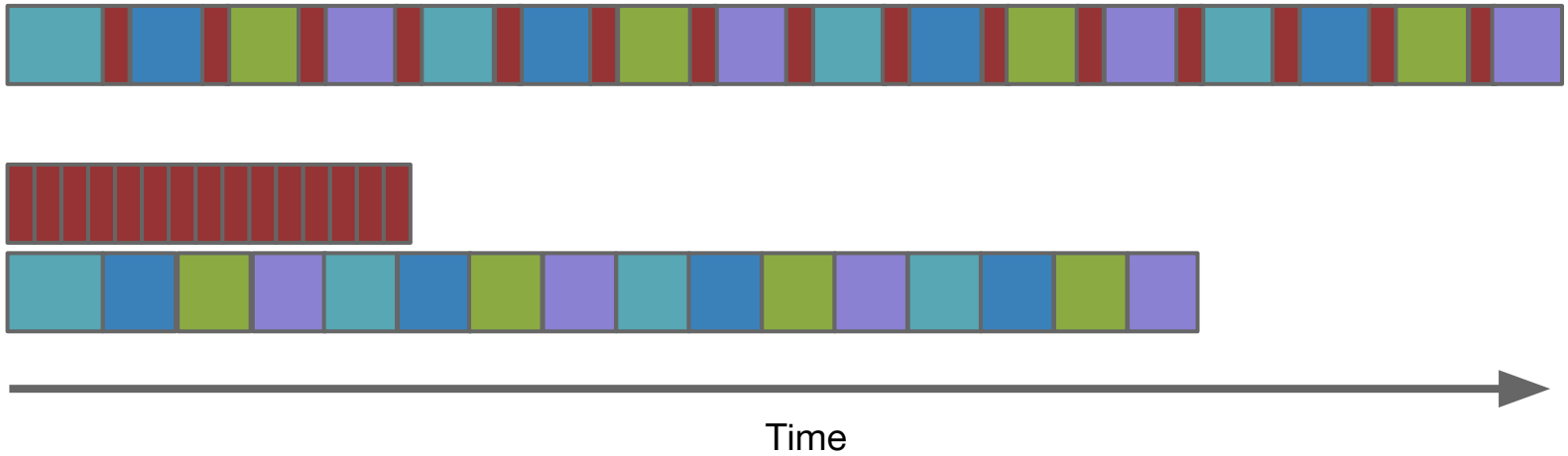


Concurrency!

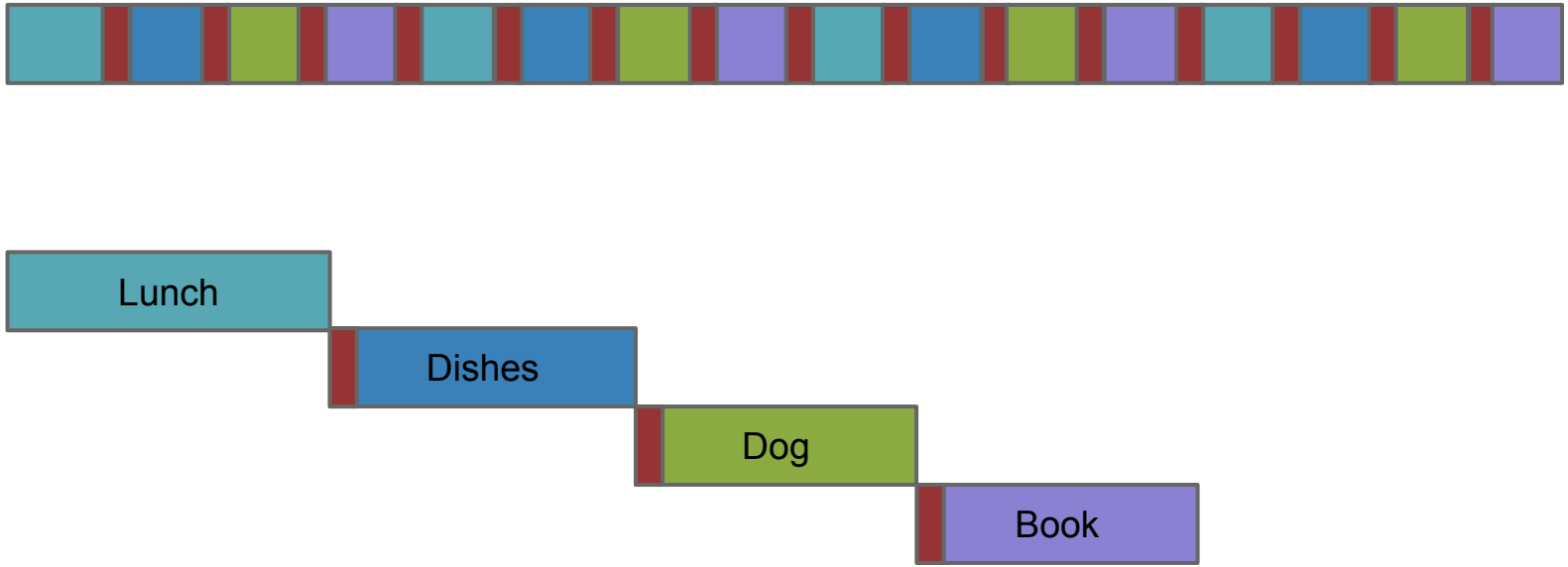




Jacob Hansson @jakewins



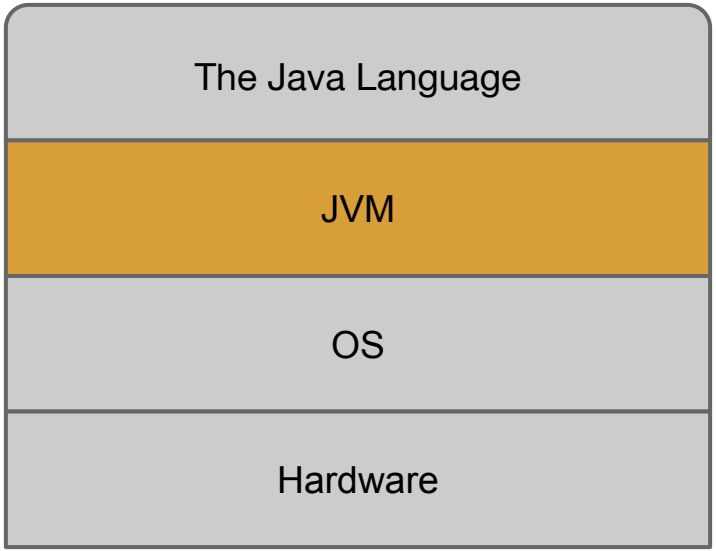
Jacob Hansson @jakewins



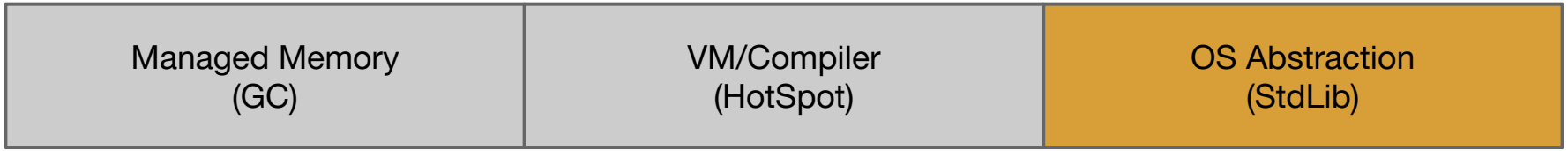
Jacob Hansson @jakewins

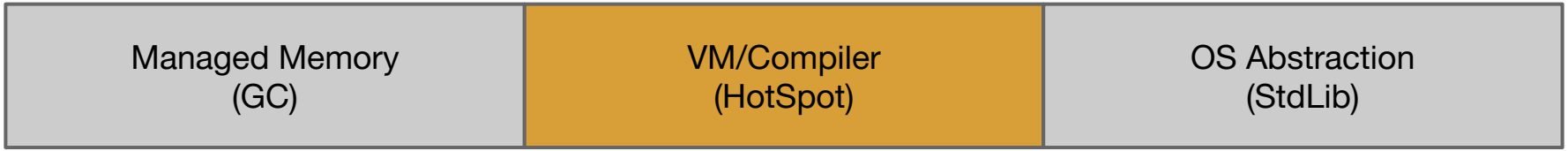
Key points

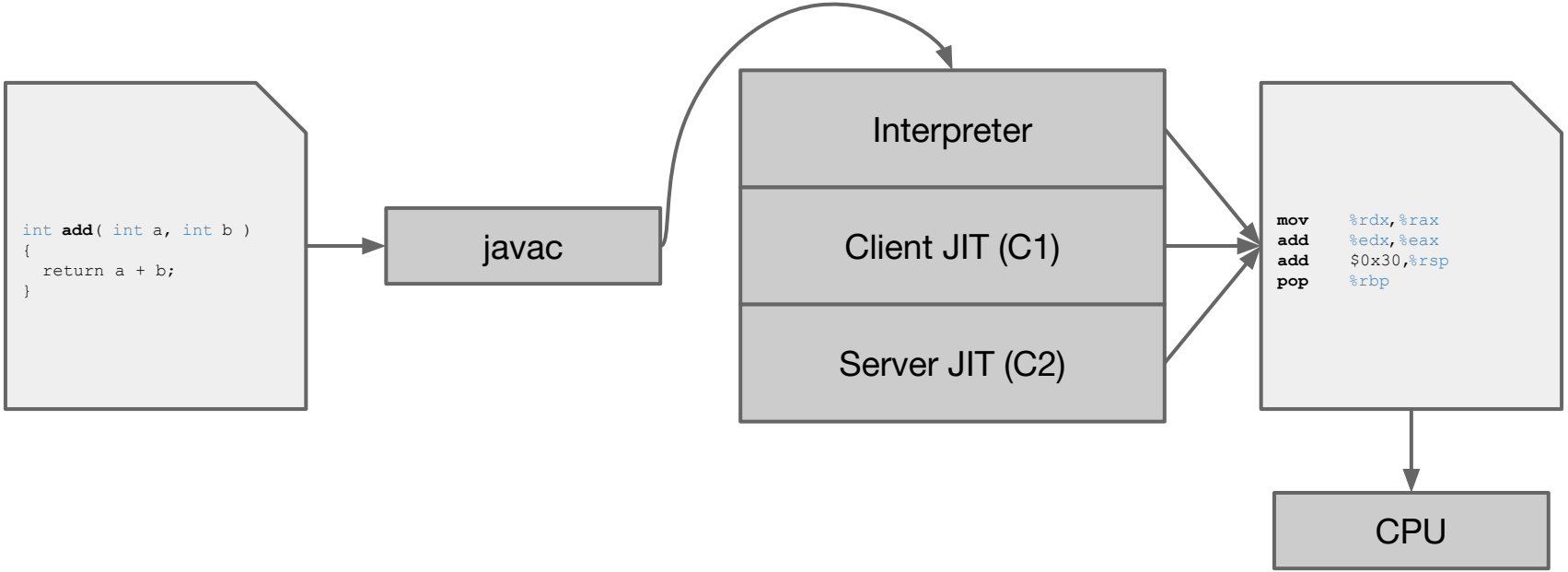
- Cost of mode switch is high
- Thread scheduling is key part of architecture
- Java locks can cause context switches
- Work in batches


The Java Language
JVM
OS
Hardware

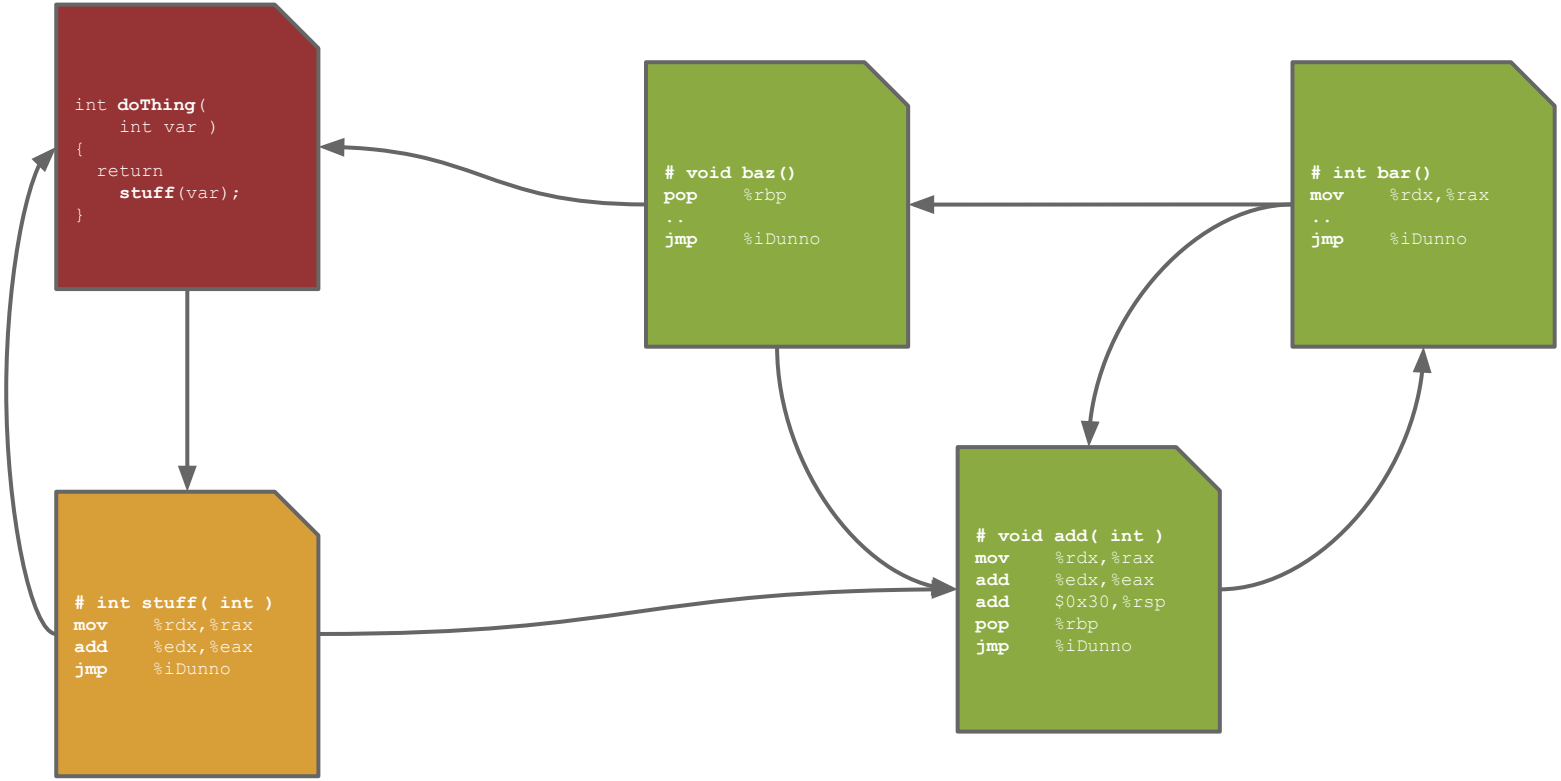
Managed Memory (GC)	VM/Compiler (HotSpot)	OS Abstraction (StdLib)


Managed Memory
(GC)
VM/Compiler
(HotSpot)
OS Abstraction
(StdLib)

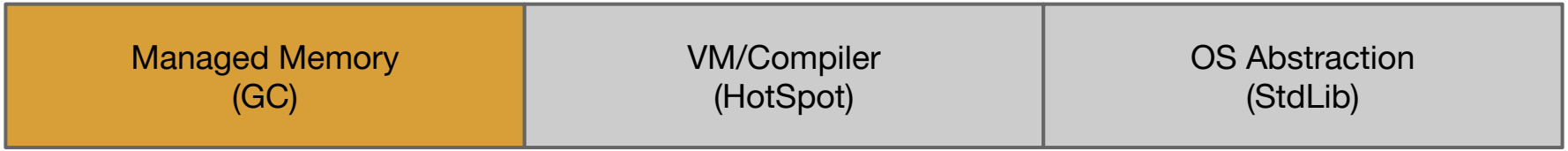

Managed Memory
(GC)
VM/Compiler
(HotSpot)
OS Abstraction
(StdLib)



Jacob Hansson @jakewins



Jacob Hansson @jakewins


Managed Memory
(GC)
VM/Compiler
(HotSpot)
OS Abstraction
(StdLib)

new Object()

Key points

- HotSpot is *fantastic*, help it help you
- *new* is not free, not even close
- the first time you meet a new JVM, turn on GC logging, and keep it on.

Outline

- Intro
- The Stack
- **Measuring**
- Architecture
- Patterns & APIs
- Q&A

Courtesy of Capital Pictures

IKEA TESTS

There are..

- Lies
- Damn lies
- Statistics
- Dynamically profiled code that rewrites itself while it executes

∞ ops / ns

```
public void bench() {
    Math.log(Math.PI);
}
```

∞ ops / ns

```
public double bench() {
    return Math.log(Math.PI);
}
```

45.038 ops / ns

```
public int bench(int reps) {
    int s = 0;
    for(int i = 0; i < reps; i++) {
        s += (x + y);
    }
    return s;
}
```

http://openjdk.java.net/projects/code-tools/jmh/

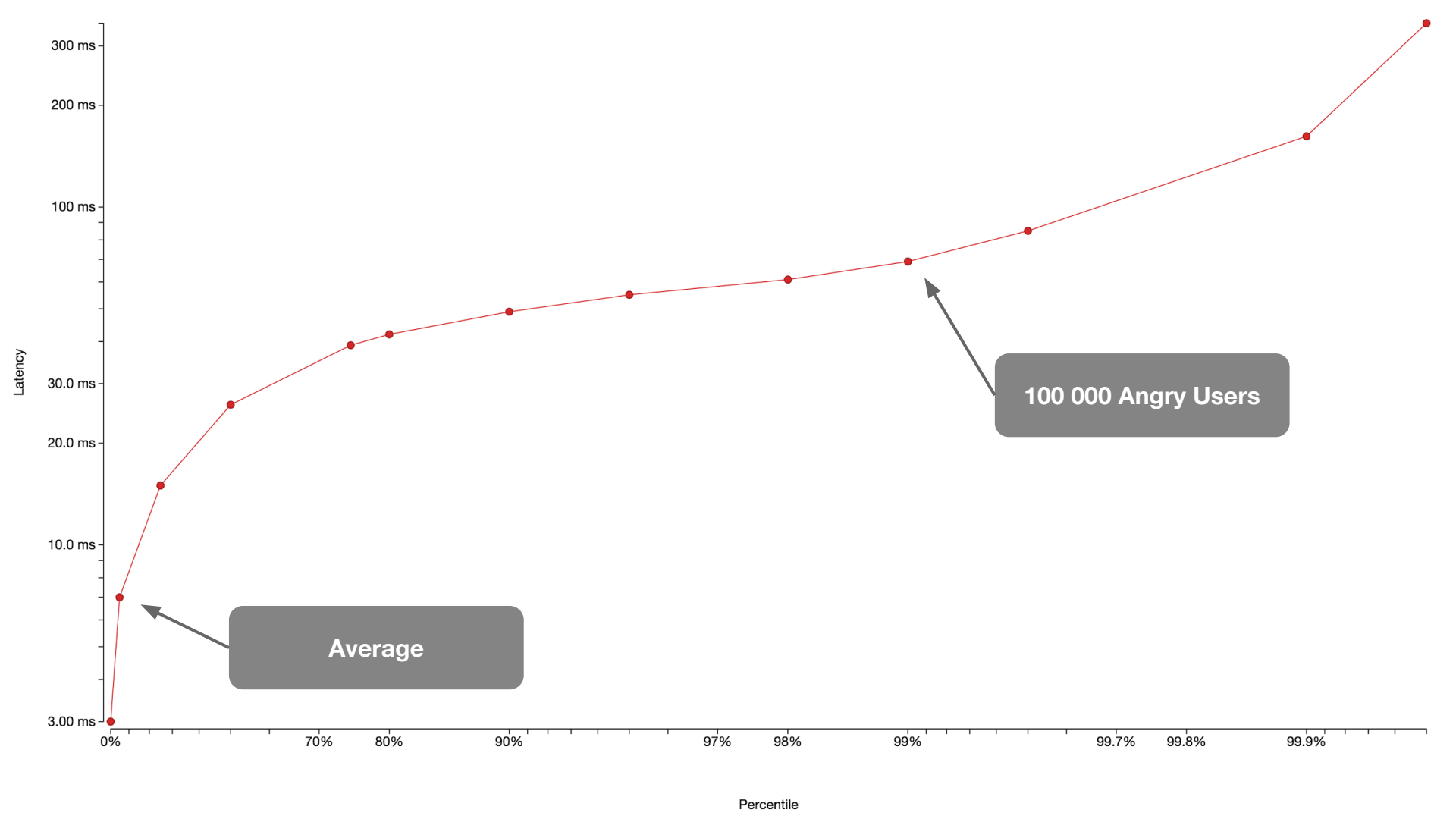
0.048 ops / ns

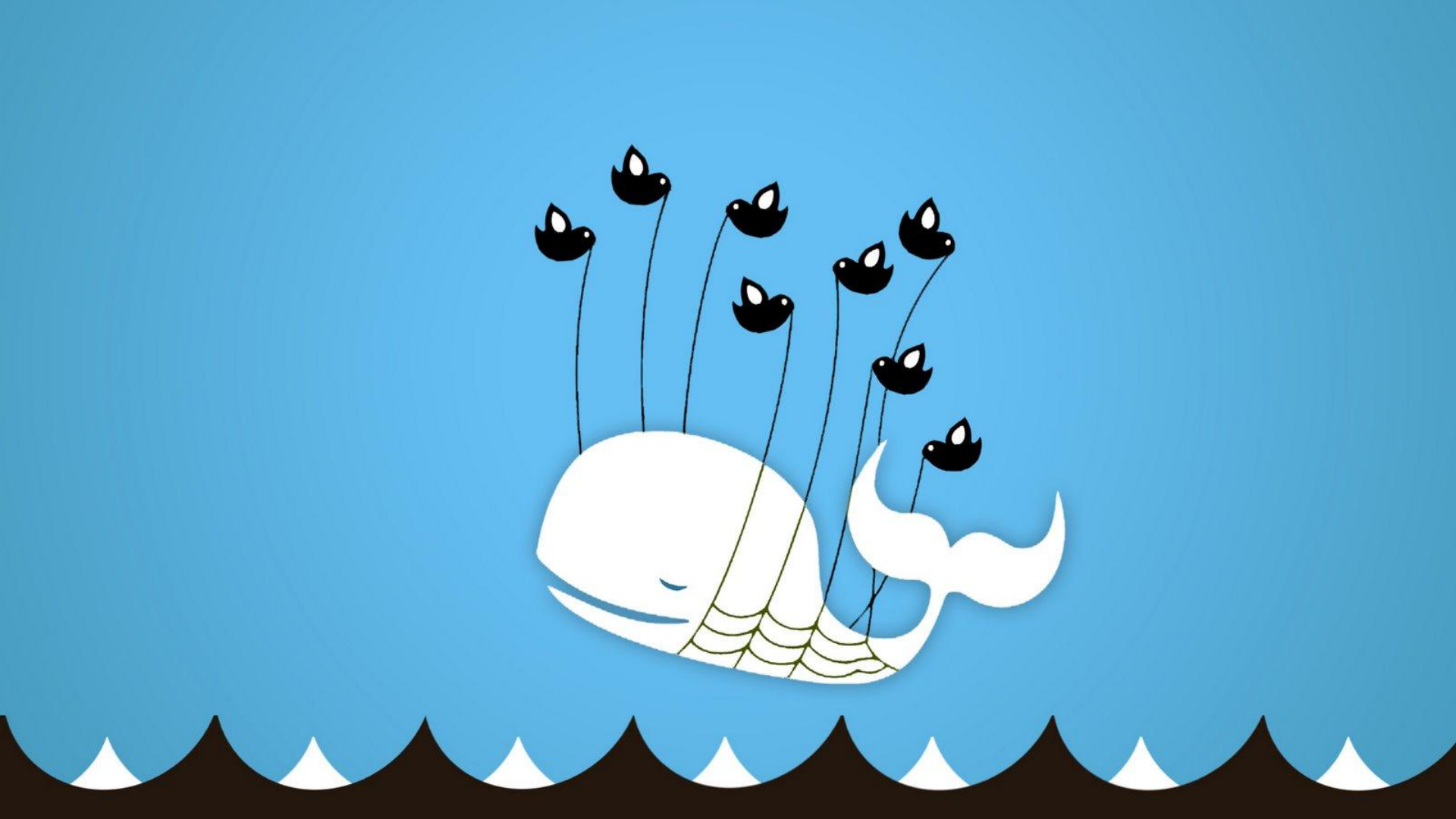
```
@State( Scope.Thread )
class MyBenchmark {
    private double pi = Math.PI;

    @Benchmark
    public double bench() {
        return Math.log(pi);
    }
}
```

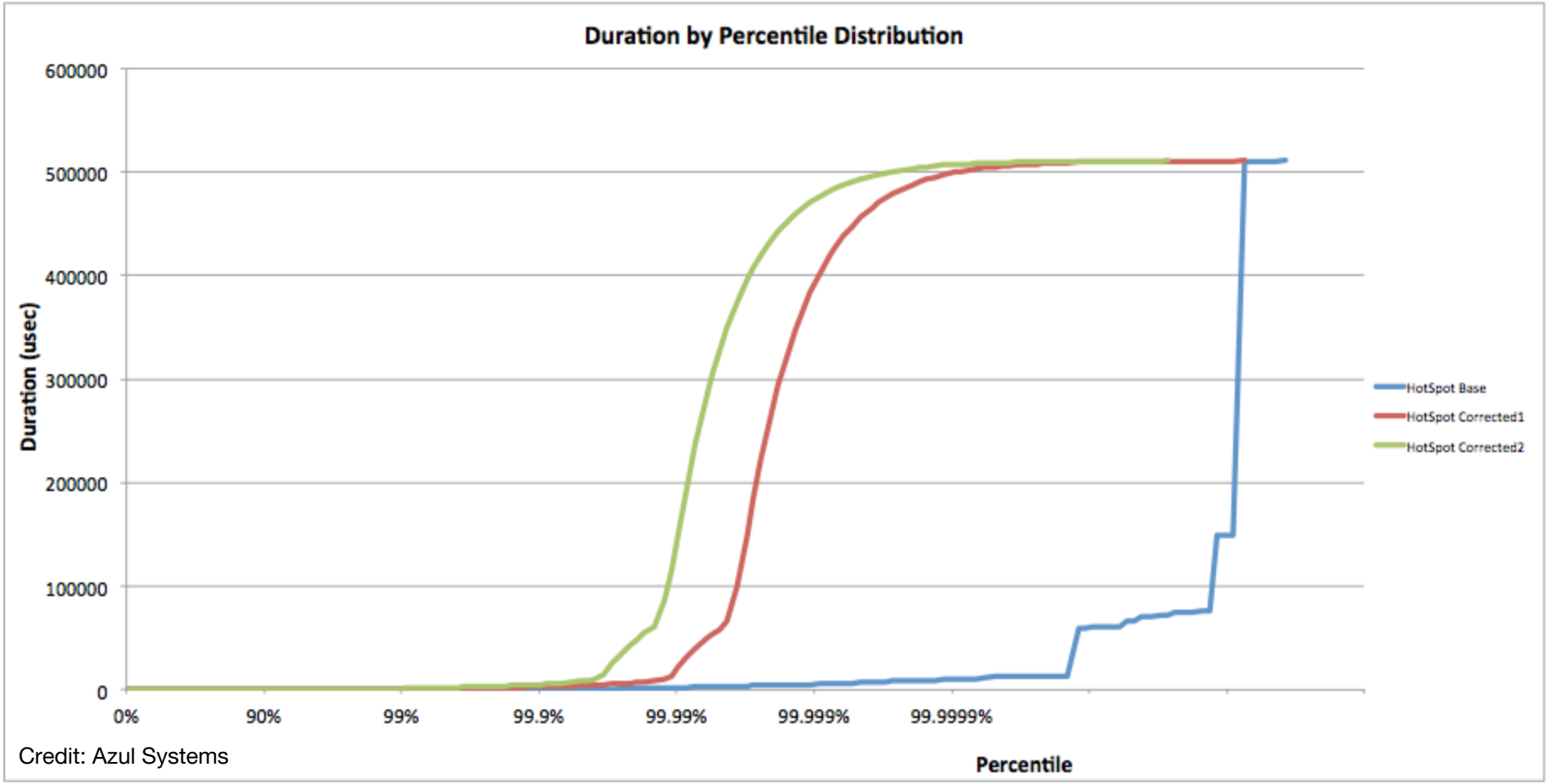
P191w

```
Build 4507
Device: Hard Drive
Short Test
Status:

No problems have been found with this system so far. Do you
want to run the remaining memory tests? This will take
about 30 minutes or more.
Do you want to continue? (Recommended)
Yes or No

PRESS ESC TO ABORT TESTING
```

FIXED ASSET-
DO NOT COUNT
INVENTORY
acer


300 ms
200 ms
100 ms
30.0 ms
20.0 ms
10.0 ms
3.00 ms
Latency
0%
70%
80%
90%
97%
98%
99%
99.7%
99.8%
99.9%
Percentile
100 000 Angry Users
Average



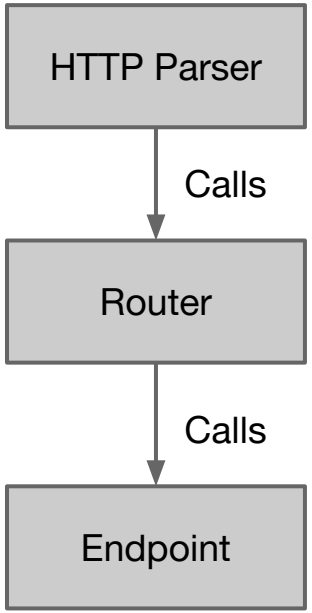


Jacob Hansson @jakewins

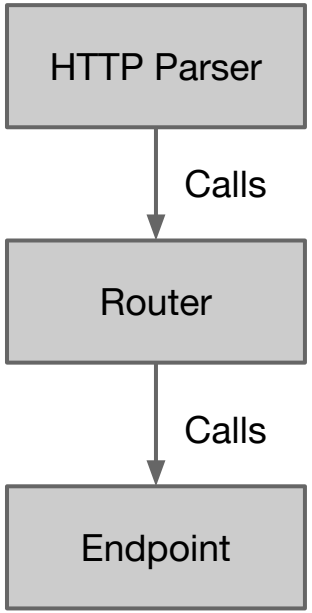
Courtesy of Capital Pictures

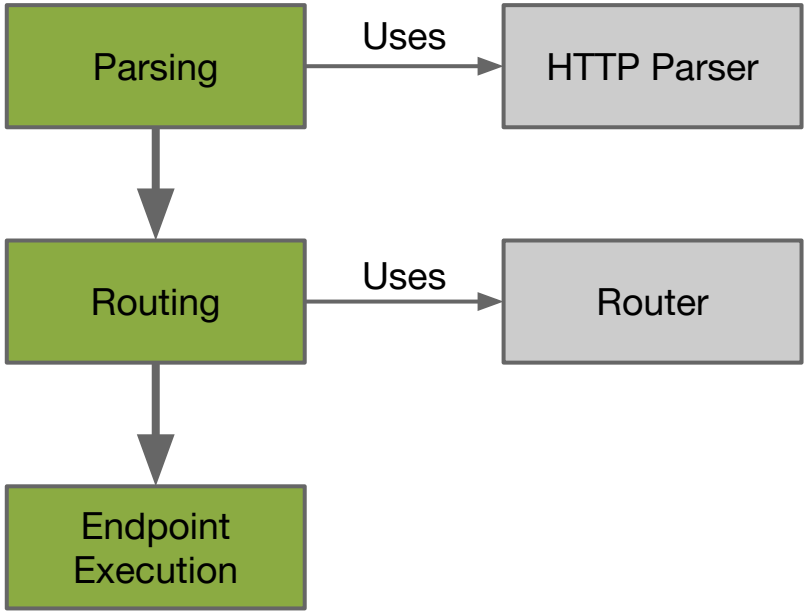
Outline

- Intro
- The Stack
- Measuring
- **Architecture**
- Patterns & APIs
- Q&A

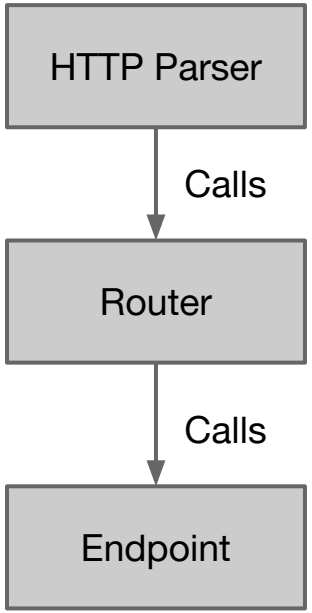


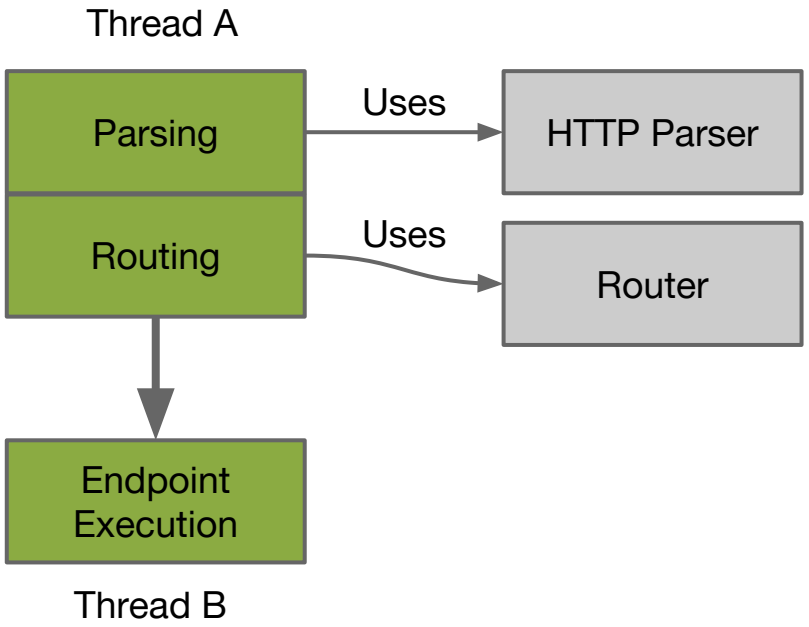
- Processors
- Services



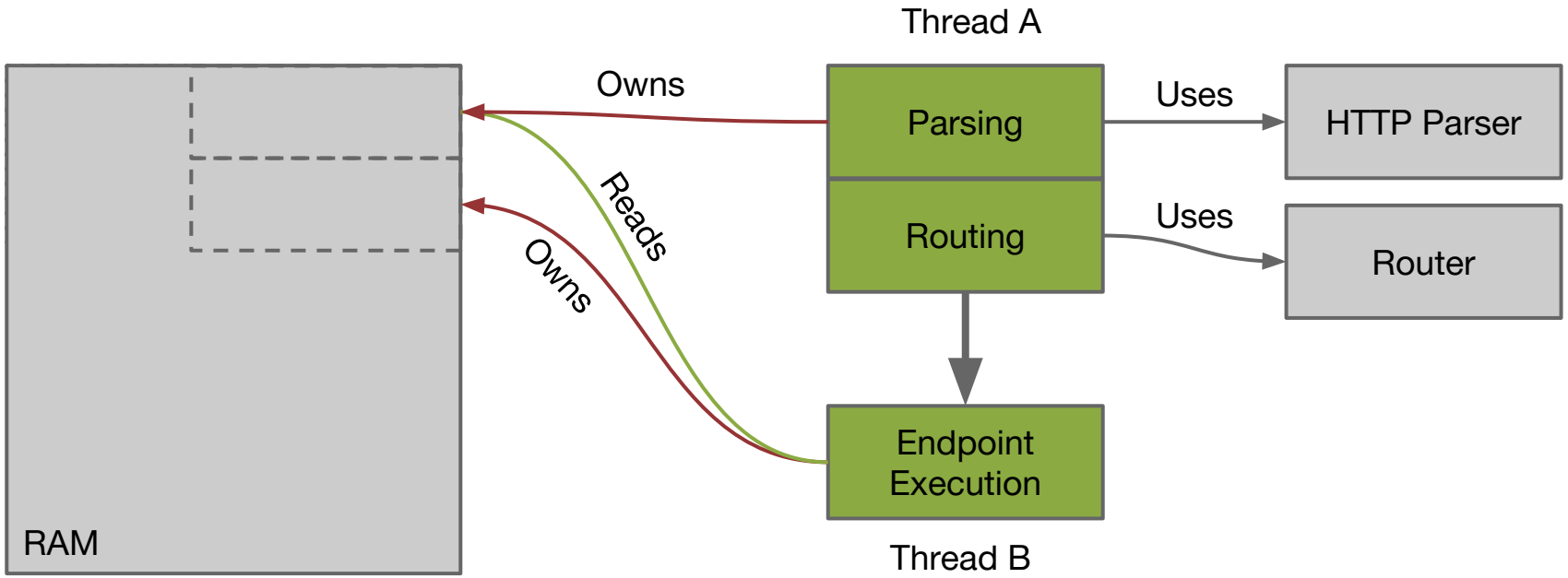


Jacob Hansson @jakewins





Jacob Hansson @jakewins



Jacob Hansson @jakewins

```
interface Locks
{
    void lock( long processId, long resourceId );
}
```

```
interface Locks
{
    void lock( long processId, long resourceId );
}
```

```
interface Locks {
    interface Client extends AutoCloseable {
        void lock( long resourceId );
    }

    Client newClient();
}
```

Key points

- Architecture > micro optimizations
- Design with processors and services
- Single-writer principle
- Make multi-client services explicit

Outline

- Intro
- The Stack
- Measuring
- Architecture
- **Patterns & APIs**
- Q&A

Patterns to help GC

```
// Iterator
interface MyEvents extends Iterator<Event> {
    boolean hasNext();
    Event next();
}
```

```
// Cursor
interface MyEvents extends Event {
    boolean next();
}
```

```
// Regular heap
new byte[..]

// "Off-heap"
ByteBuffer.allocateDirect(..);

// malloc
unsafe.allocateMemory(..);
// realloc
unsafe.reallocateMemory(..);
// free
unsafe.freeMemory(..);
```

Benefits of "raw" memory:

- No impact on GC*. Remember, GC time is (usually) relative to heap size.
- Significantly faster in some cases (1-100x, test!)
- Channels API pass direct BB straight through, no copying

*as long as you keep the memory around!

```
// Classic
thing.handle( new MyEvent( .. ) );
```

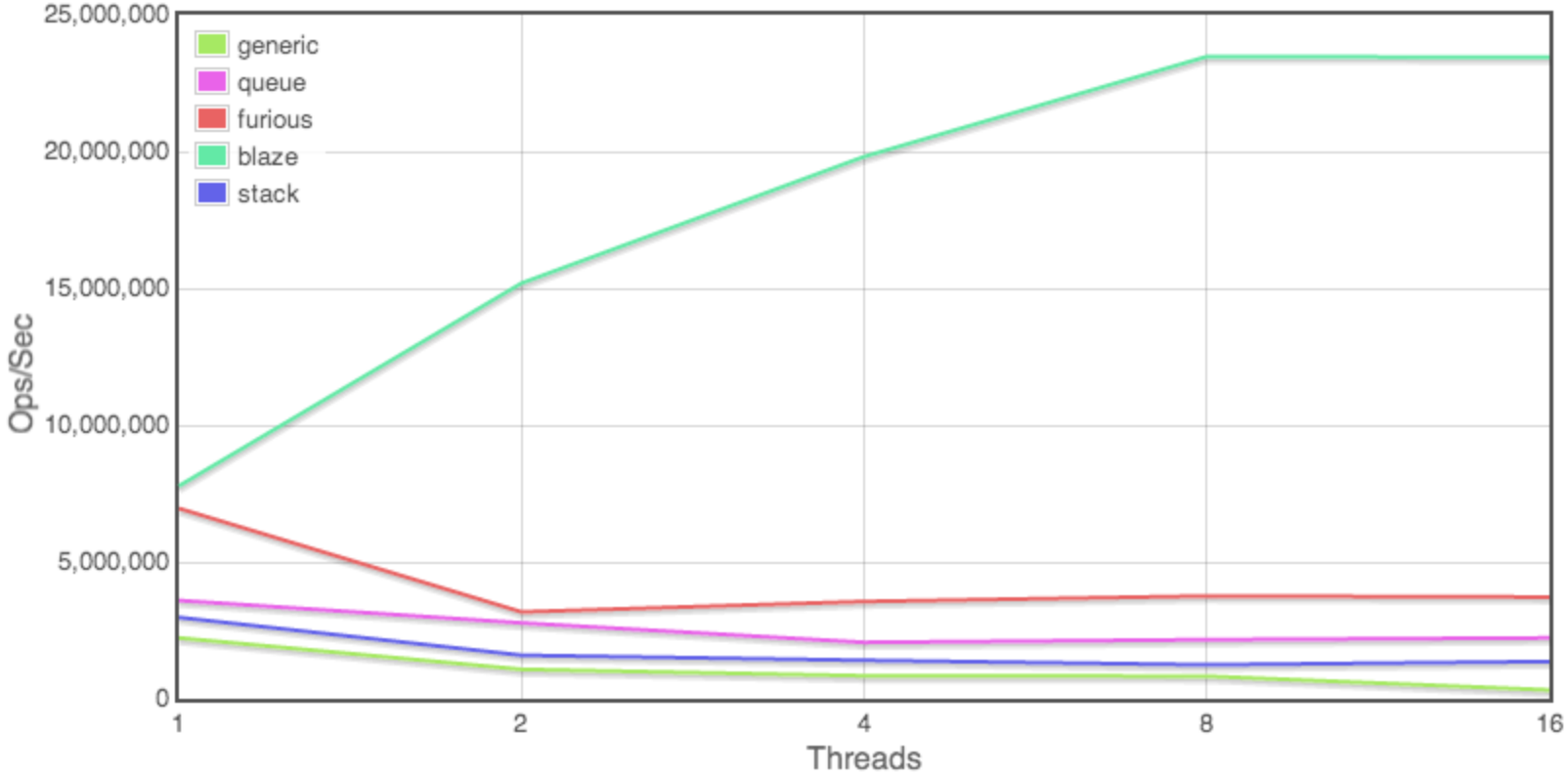
```
// Reset-pattern
thing.handle( event.reset( .. ) );
```

```
class Foo
{
    private final Thing thing = ..;

    public void sendEvent()
    {
       thing.handle( new Event( .. ) );
    }
}
```

```
class Foo
{
    private final Thing thing = ..;
    private final Event event = new Event();

    public void sendEvent()
    {
       thing.handle( event.reset( .. ) );
    }
}
```

Pooling: Now you have two problems! Or?

Throughput:



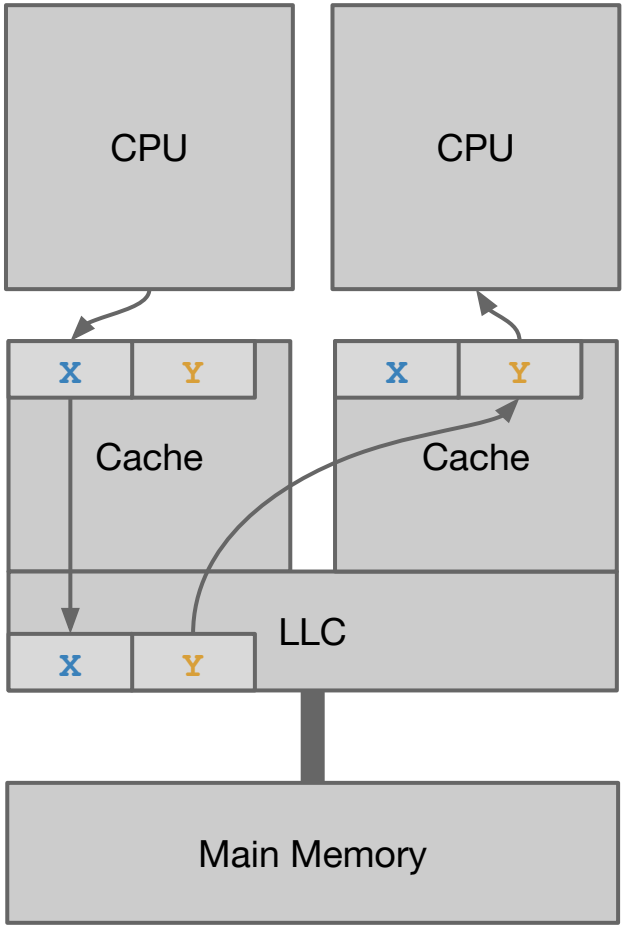
https://github.com/chrisvest/stormpot

```
Map<Integer, Object> map = ..;

int myKey = 12;     // Ends up on stack ( = ~free )
map.put( 12, .. ); // Creates Object( != free )
```

```
public void myHelpfulMethod( int ... vals )
{
   ..
}

myHelpfulMethod( 12 ); // Creates array
```

Other tips & tricks

```
int readByReaderThread;
int writtenByWriterThread;

// With writer thread running, reader performance:
0.104 reads/ns

// Without writer thread running
0.388 reads/ns (!)
```



Jacob Hansson @jakewins

Console | Experiment Summary | Tuning Navigator | Memory Hotspots | Advanced Profile Info

False-True-Sharing-(1)-2009-10-23-18-11-02 Granularity Cachelines

Cacheline Address / Offset / Thread / Function	MEM_UNCORE_RE..._CORE_L2_HITM	MEM_INST_RETIRED.STORES	Contributors
0x00498180	**118,426**	**649,611**	**Offsets: 4 Threads: 4**
Offset:0x20(32)	30,846	163,139	Threads: 1
Thread:00000fe0(0059)	30,846	163,139	Functions: 2
work	30,846	163,126	
initialize_res	0	13	
Offset:0x30(48)	29,973	162,848	Threads: 2
Thread:00000f70(0062)	29,938	162,785	Functions: 1
work	29,938	162,785	
Thread:00000fe0(0059)	35	63	Functions: 2
initialize_res	0	63	

```
// Java 8
@Contended
int writtenByWriterThread;

// Java 7
long p01, p02, p03, p04, p05, p06, p07, p08;
int writtenByWriterThread;
```

The power of

final

&

class

-XX:MinInliningThreshold
-XX:MaxInlineSize

Collection libraries

https://github.com/JCTools/JCTools
https://github.com/real-logic/Agrona
https://github.com/goldmansachs/gs-collections
http://fastutil.di.unimi.it/docs/
http://trove.starlight-systems.com/

Outline

- Intro
- The Stack
- Measuring
- Architecture
- Patterns & APIs
- **Q&A**