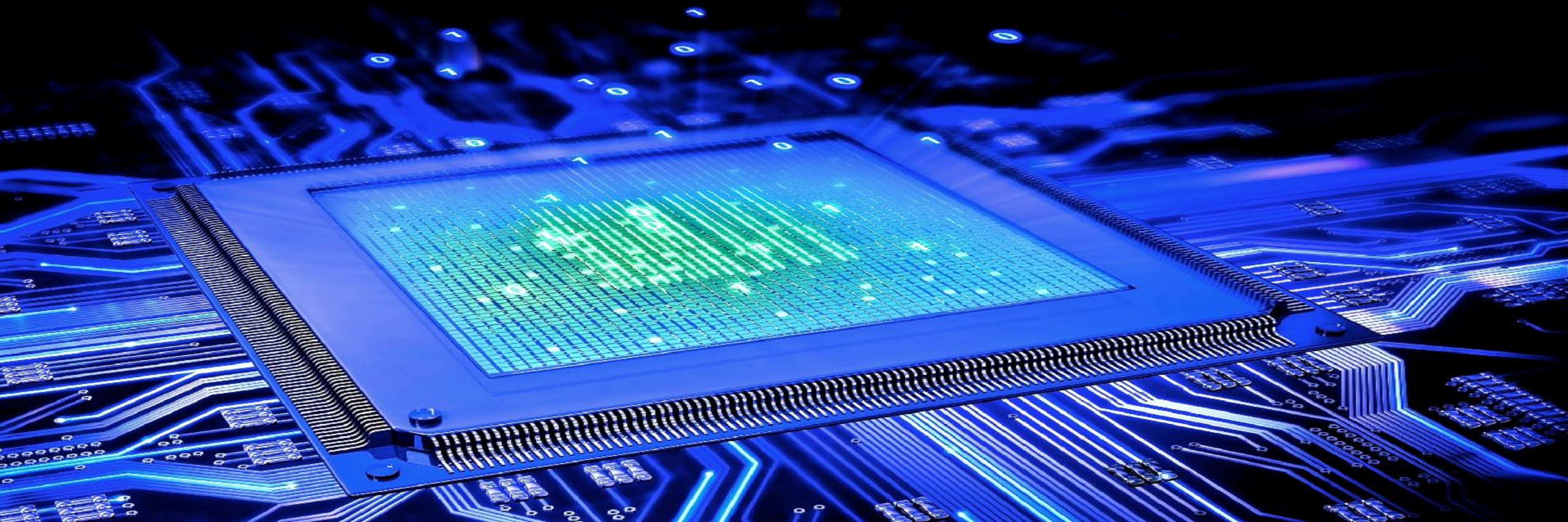


«Mechanical Sympathy» Distilled

a TechTalk by Michael Pellaton on 2017-11-29



Kurs «Performance Testing & Tuning Java Applications»

by Martin Thompson

Destillat aus 3 Kurstagen

🖥 Moderne Hardware

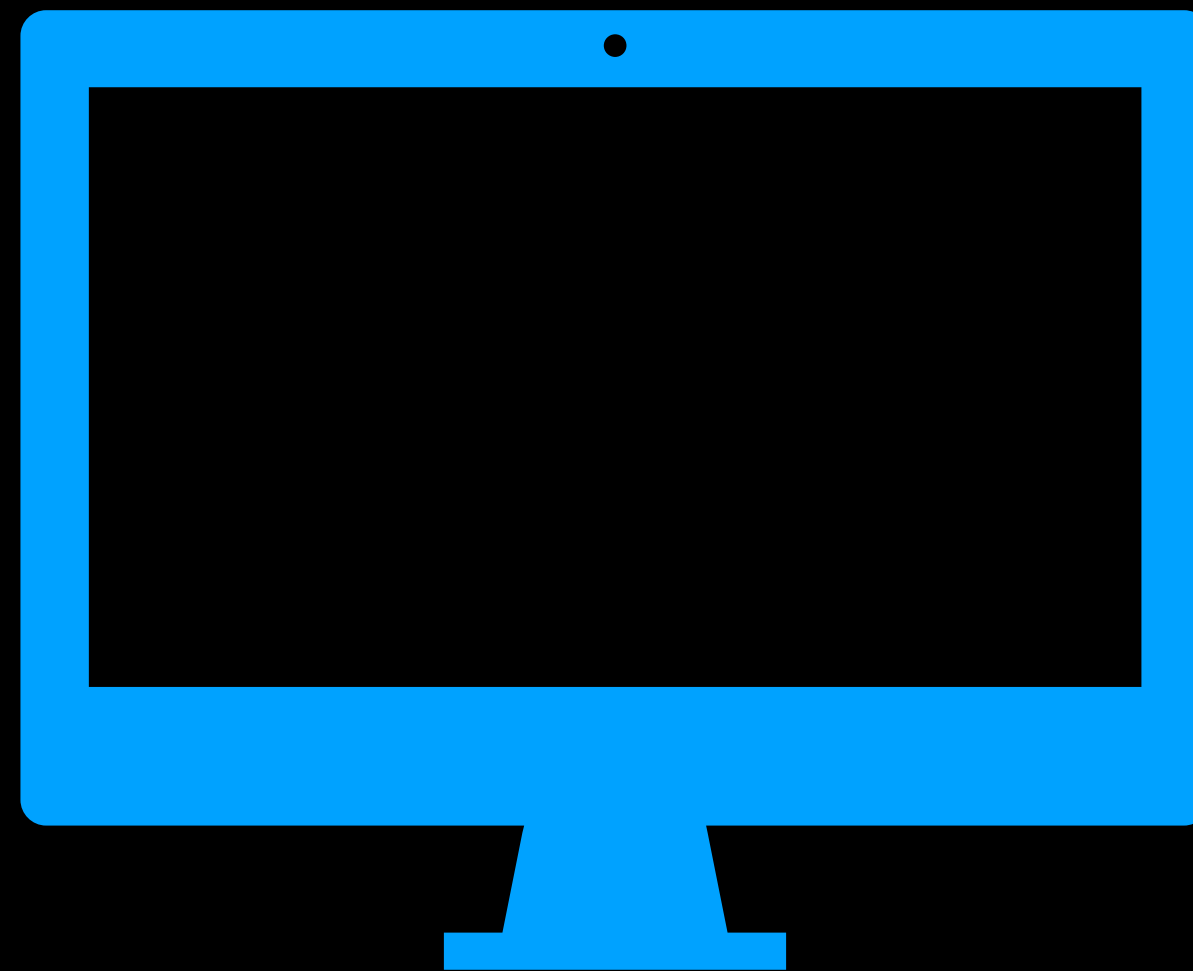
📈 Latenzen

🔍 Beispiel mit JMH

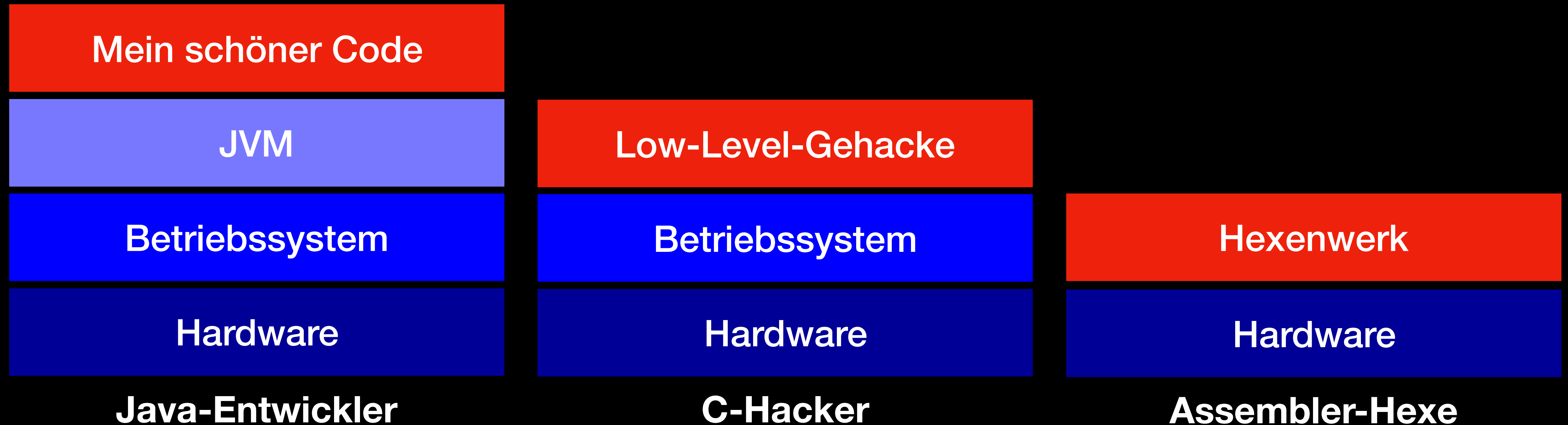
💡 Erkenntnis



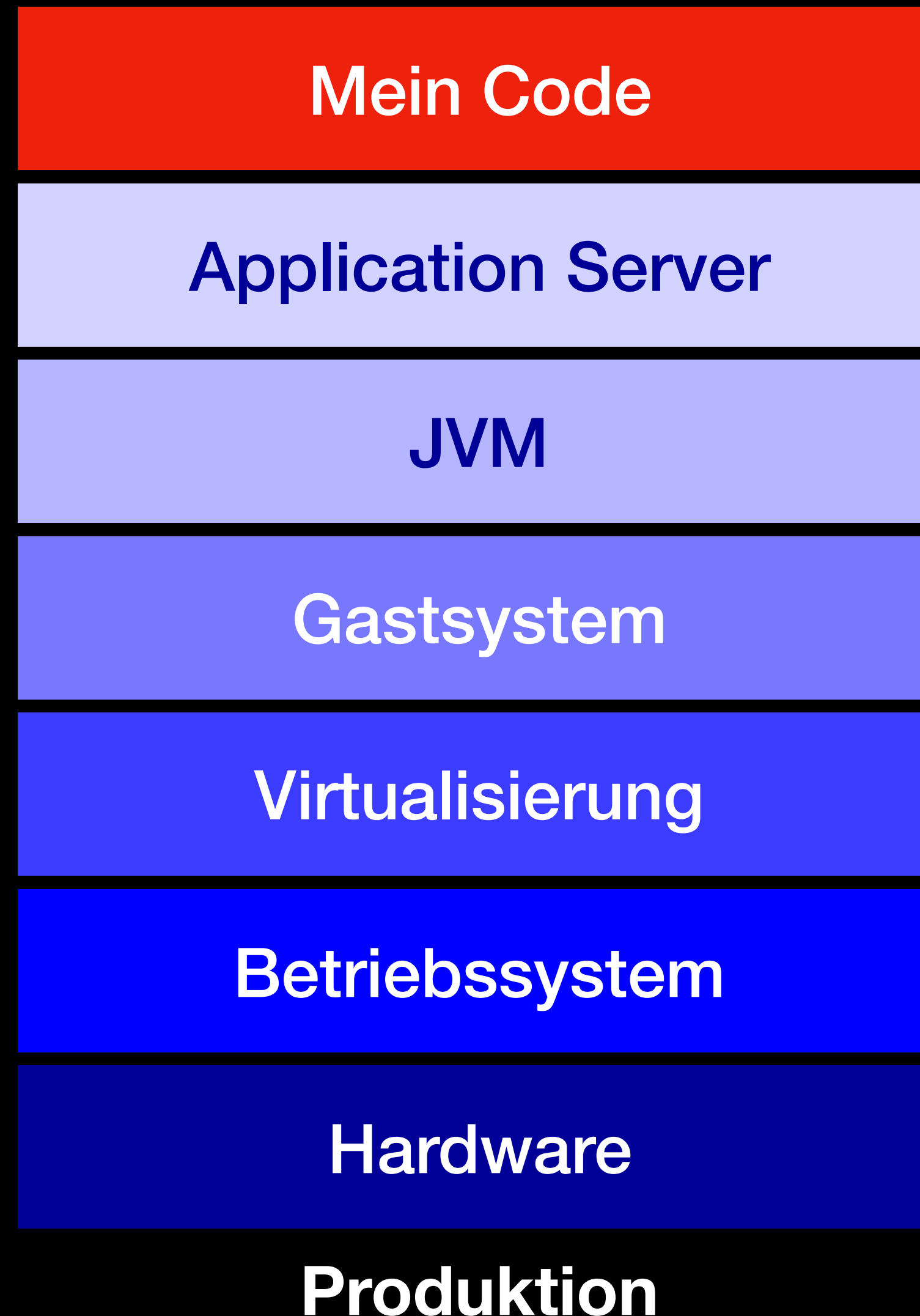
Moderne Hardware



Duke Javas Sicht der Welt

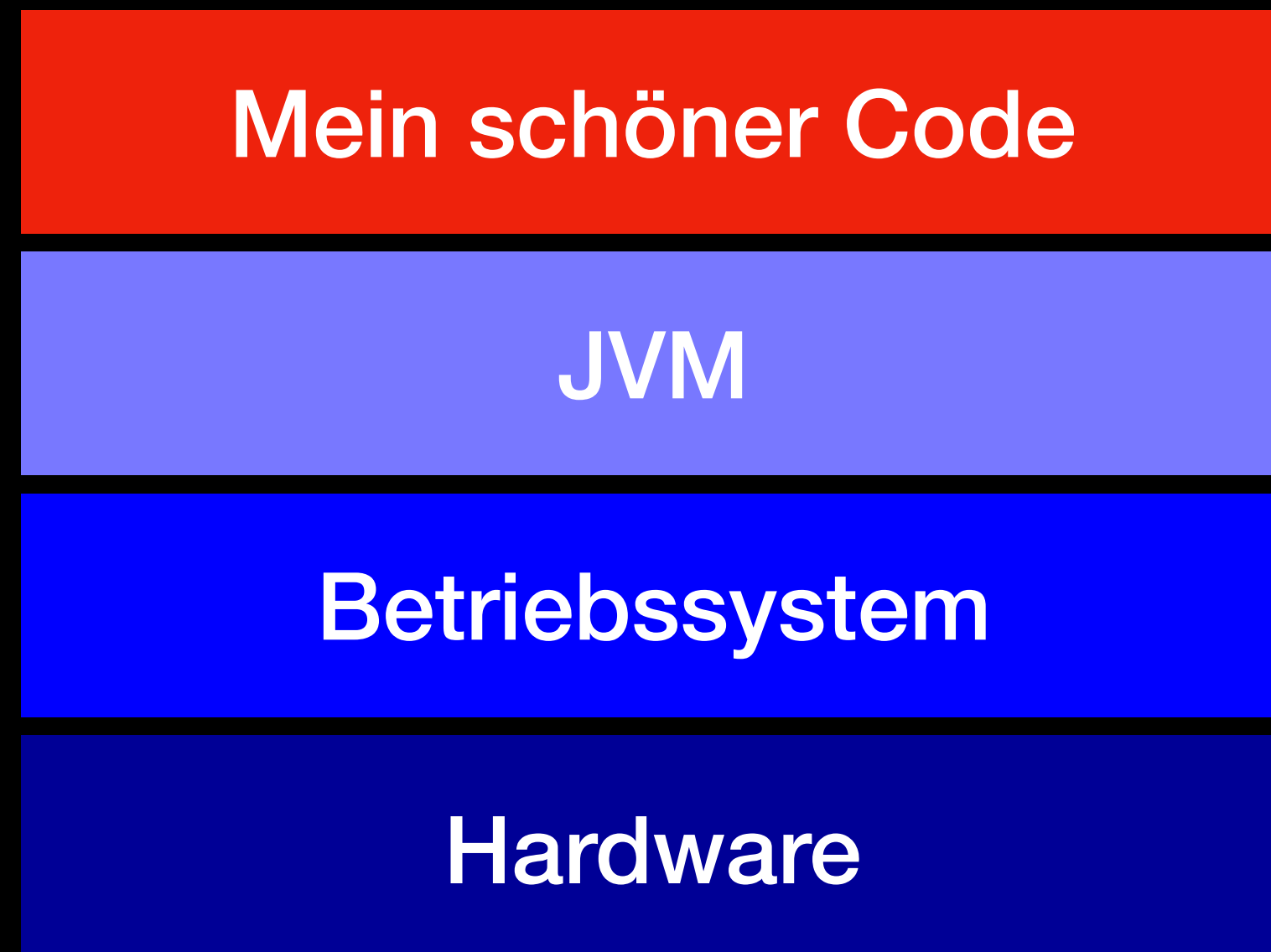


Duke Javas Sicht der Welt

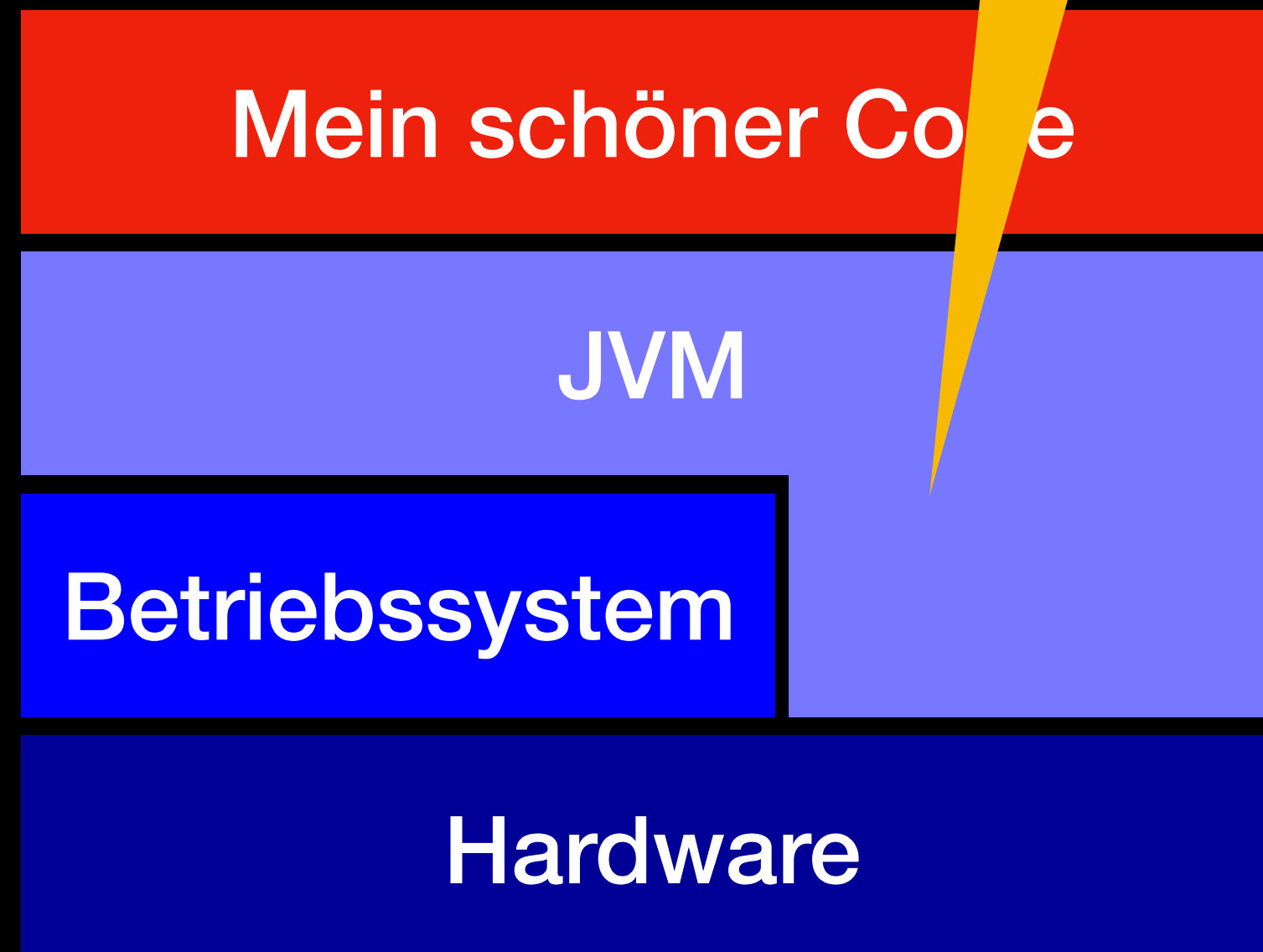


Duke Javas Sicht der Welt

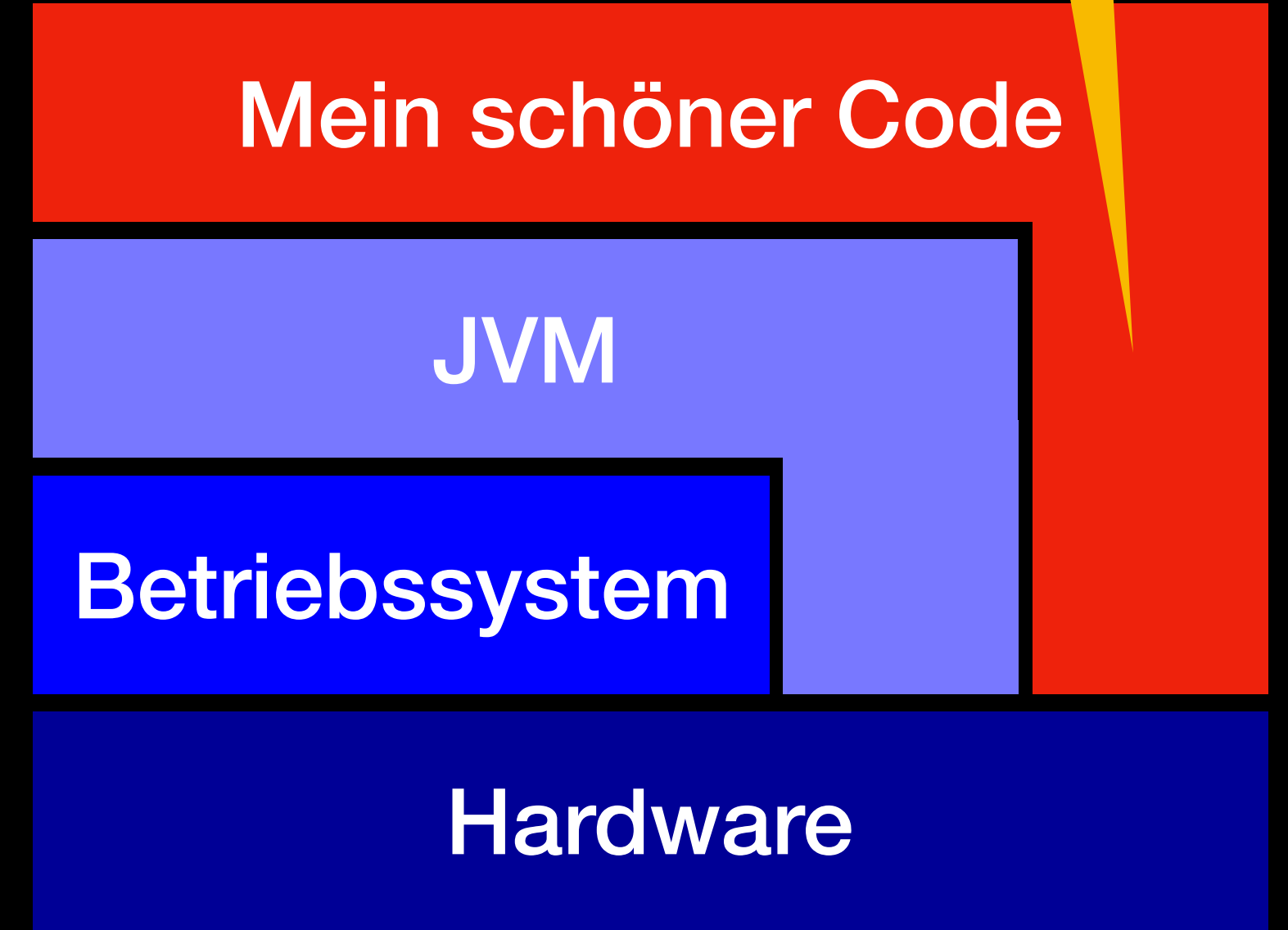
Stimmt das?



Nicht
alle Calls
durch OS

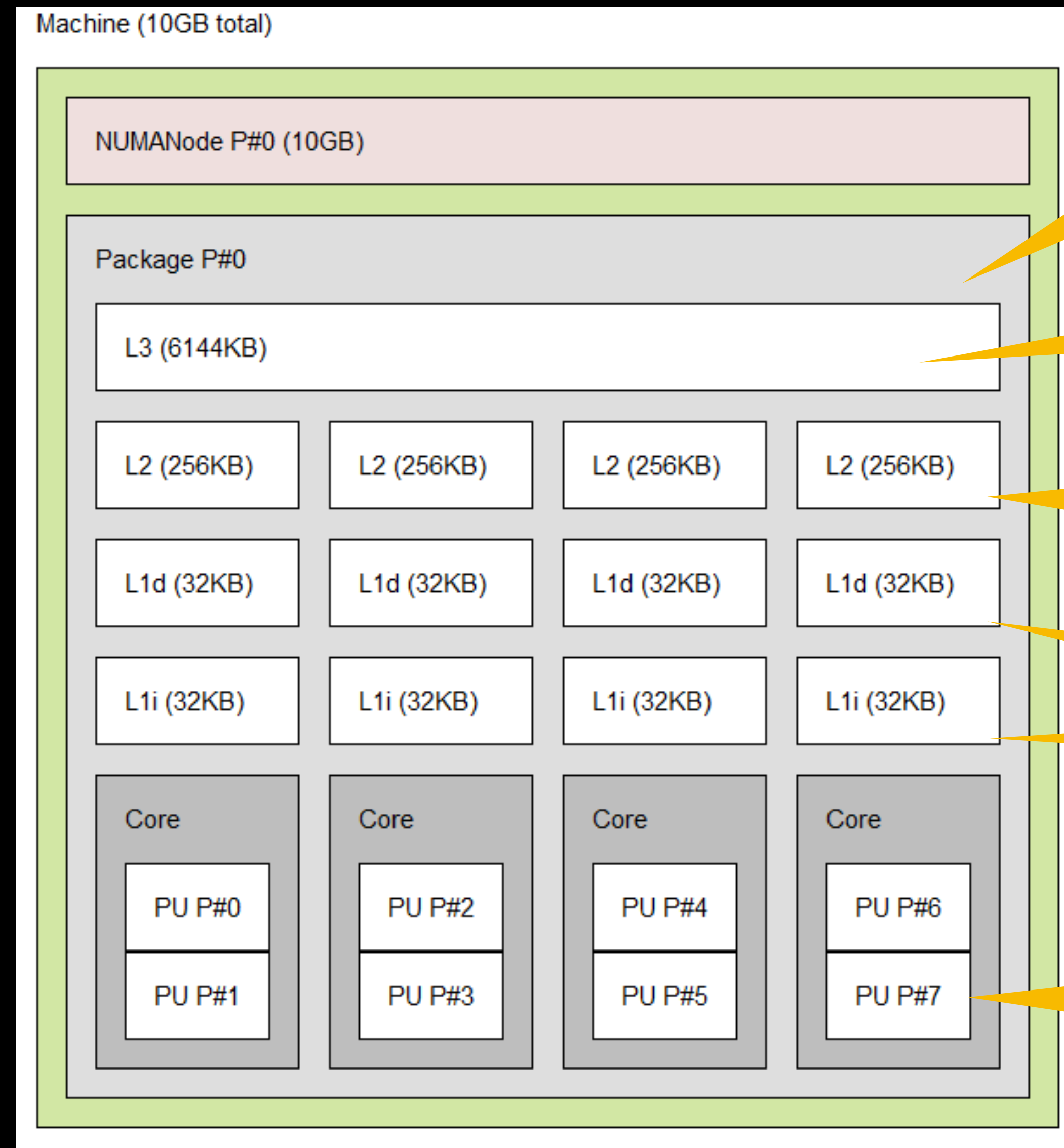


JIT



Rechnerarchitektur

hwloc-ls
auf meinem
Notebook



1 Socket

Gemeinsame
6MB L3

Pro Core
256KB L2

Pro Core
32KB L1i+d

4 Cores à
2 Hyper-
Threads

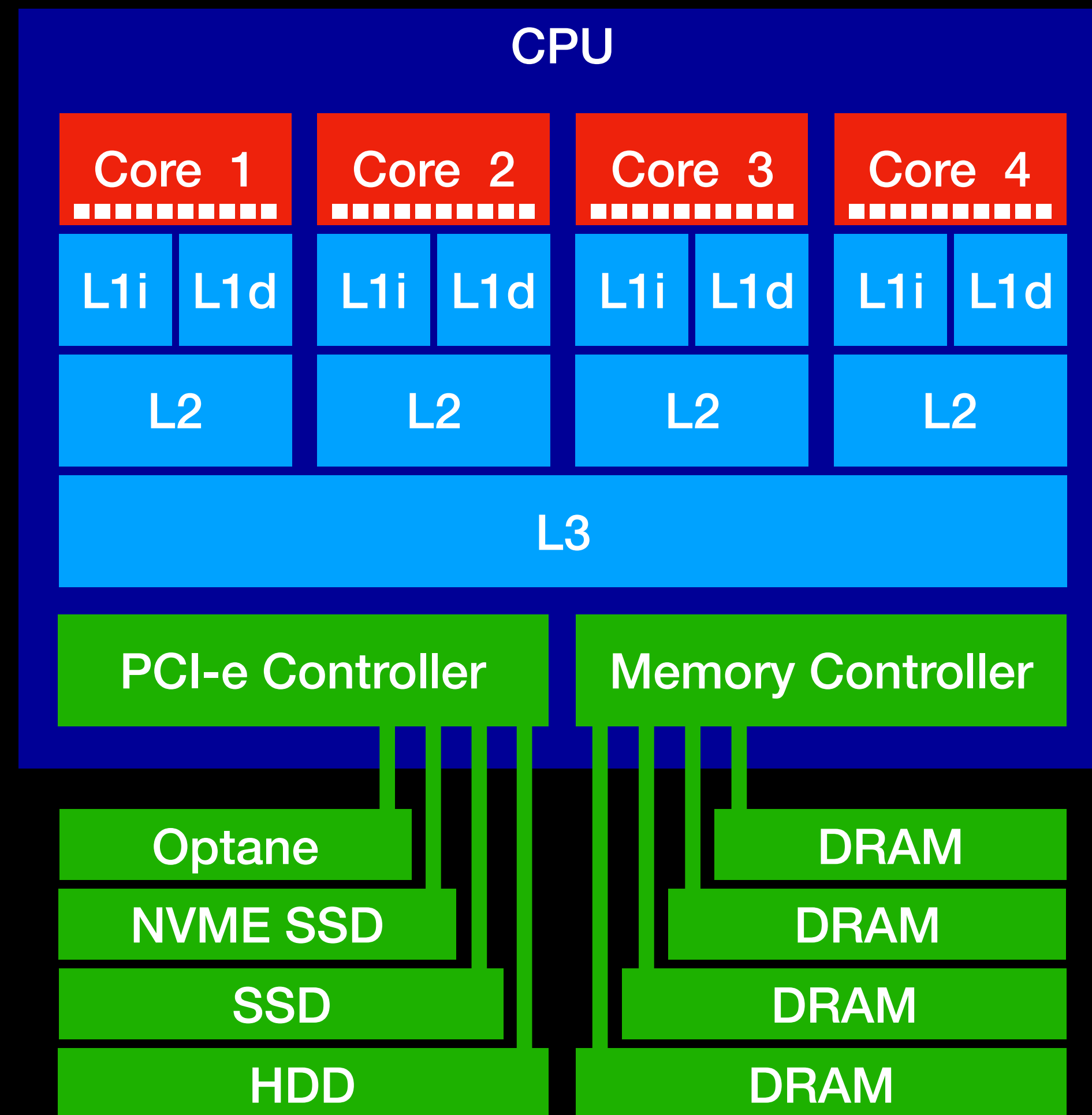
Rechnerarchitektur

Dual Xeon Server

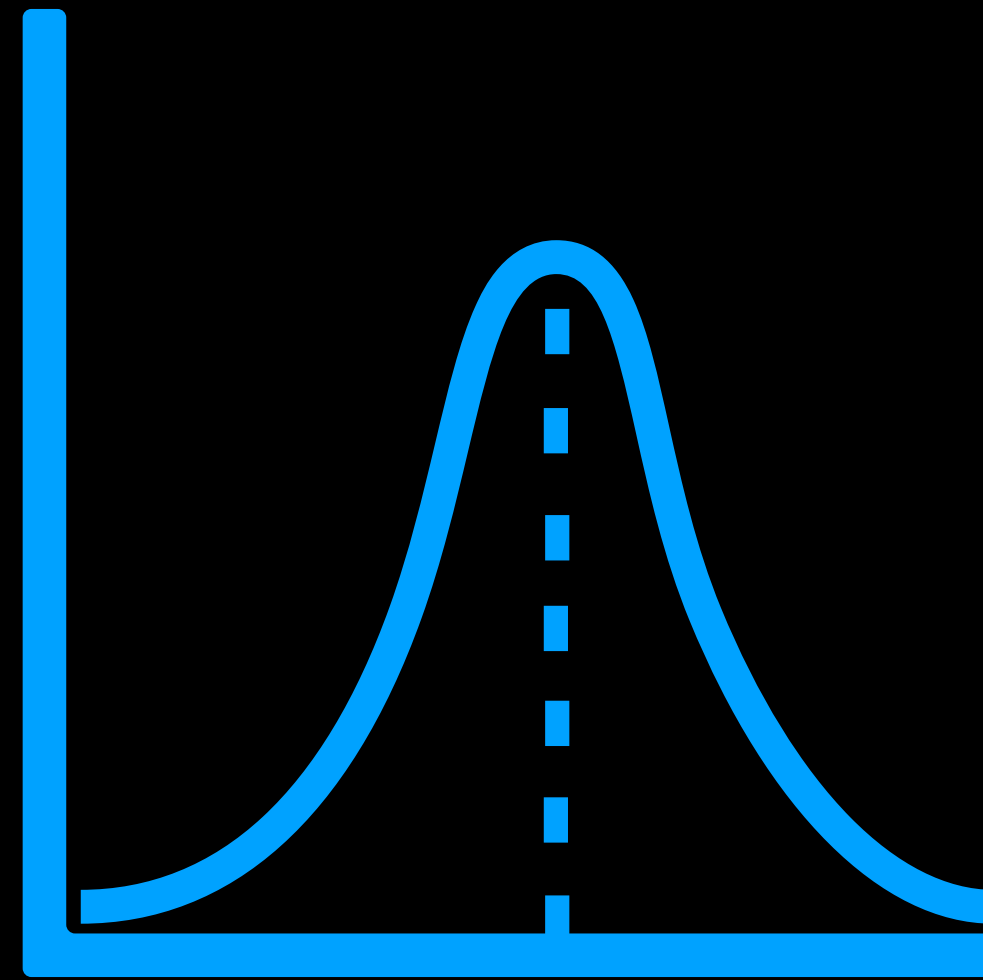


NIC und Systemdisk

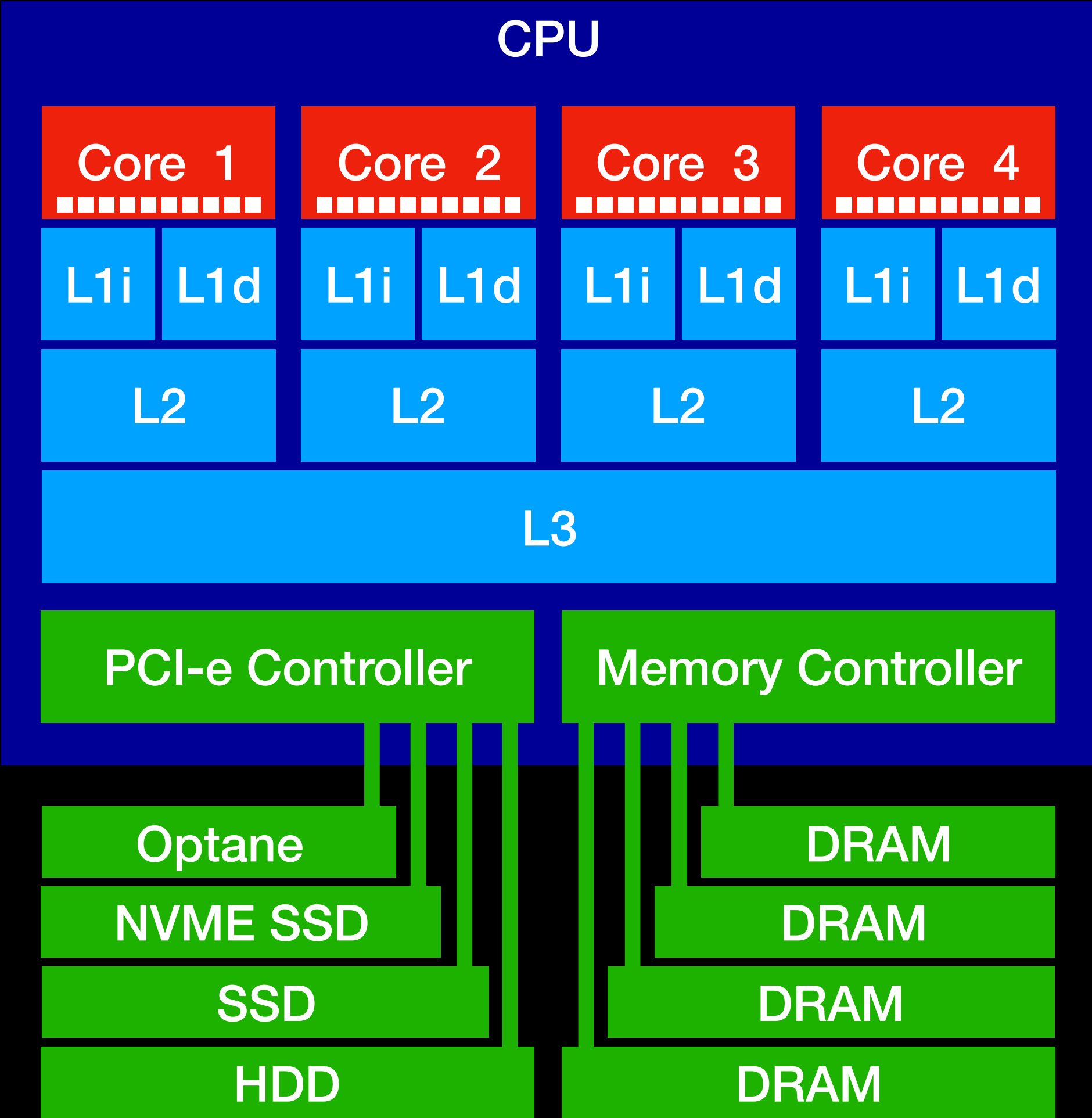
Rechnerarchitektur



Latenzen



Latenzen



~0.4ns (1 cycle) **1s**

~0.9ns (2 cycles) **2s**

~2.8ns (6 cycles) **7s**

~28ns (60 cycles) **1min**

~100ns **4min**

7h
17h
1.5-4d
1-9M

<10μs
~25μs
50-150μs
1-10ms

Internet Roundtrip SFO-NYC-SFO
Internet Roundtrip SFO-HKG-SFO

5a
11a

Beispiel mit JMH

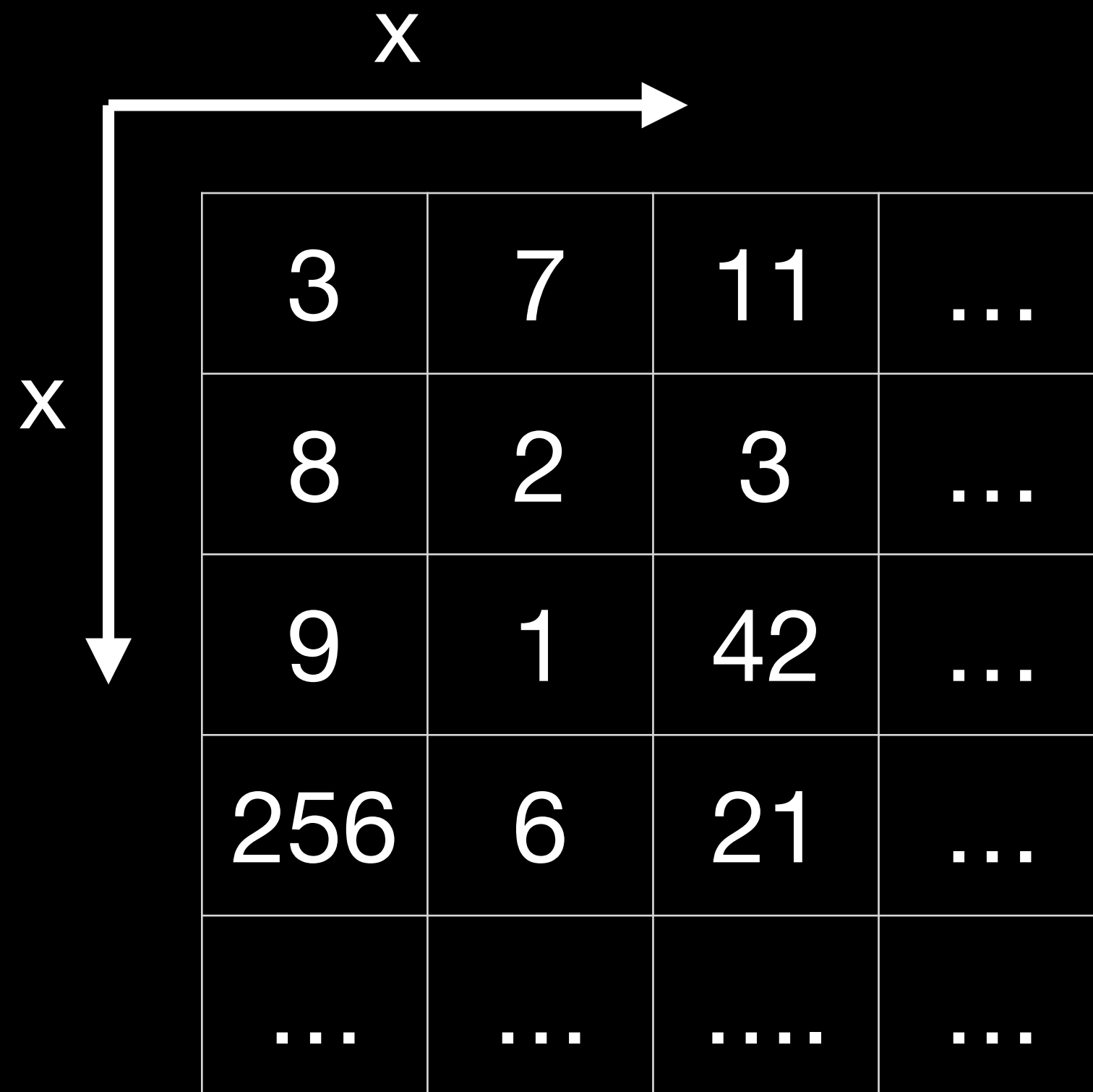


JMH

«JMH is a Java harness for building, running, and analysing nano/micro/milli/macro benchmarks written in Java and other languages targetting the JVM.»

- Löst die «Probleme» bei Performancetests in Managed Runtimes
 - JIT (Warmup)
 - Unerwünschte Optimierungen: Dead code Elimination & Co
- Sehr nah am JDK
- Von den Leuten, die auch die JVM bauen
- Annotationsbasiert (sehr ähnlich einem JUnit-Test)
- Codegenerierung während Compilation mittels AnnotationProcessor

Problemstellung



Gegeben

- quadratische Matrix mit $1k \times 1k$ ganzzahligen Werten
- Suche in kritischem Pfad der Applikation

Gesucht

- Maximalwert

Lösungsansätze

- `LinkedList<LinkedList<Integer>>: x, y`
- `LinkedList<LinkedList<Integer>>: y, x`
- `ArrayList<ArrayList<Integer>>: x, y`
- `ArrayList<ArrayList<Integer>>: y, x`
- `int[][]: x, y`
- `int[][]: y, x`

Setup JMH Project

```
$ mvn archetype:generate \  
    -DinteractiveMode=false \  
    -DarchetypeGroupId=org.openjdk.jmh \  
    -DarchetypeArtifactId=jmh-java-benchmark-archetype \  
    -DgroupId=org.sample \  
    -DartifactId=test \  
    -Dversion=1.0
```

Code

@Fork(1)

```
public class MatrixTraversalBenchmark {
```

```
    private int[][] matrixArray;
```

```
    private ArrayList<ArrayList<Integer>> matrixArrayList;
```

```
    private LinkedList<LinkedList<Integer>> matrixLinkedList;
```

@Setup(Invocation)

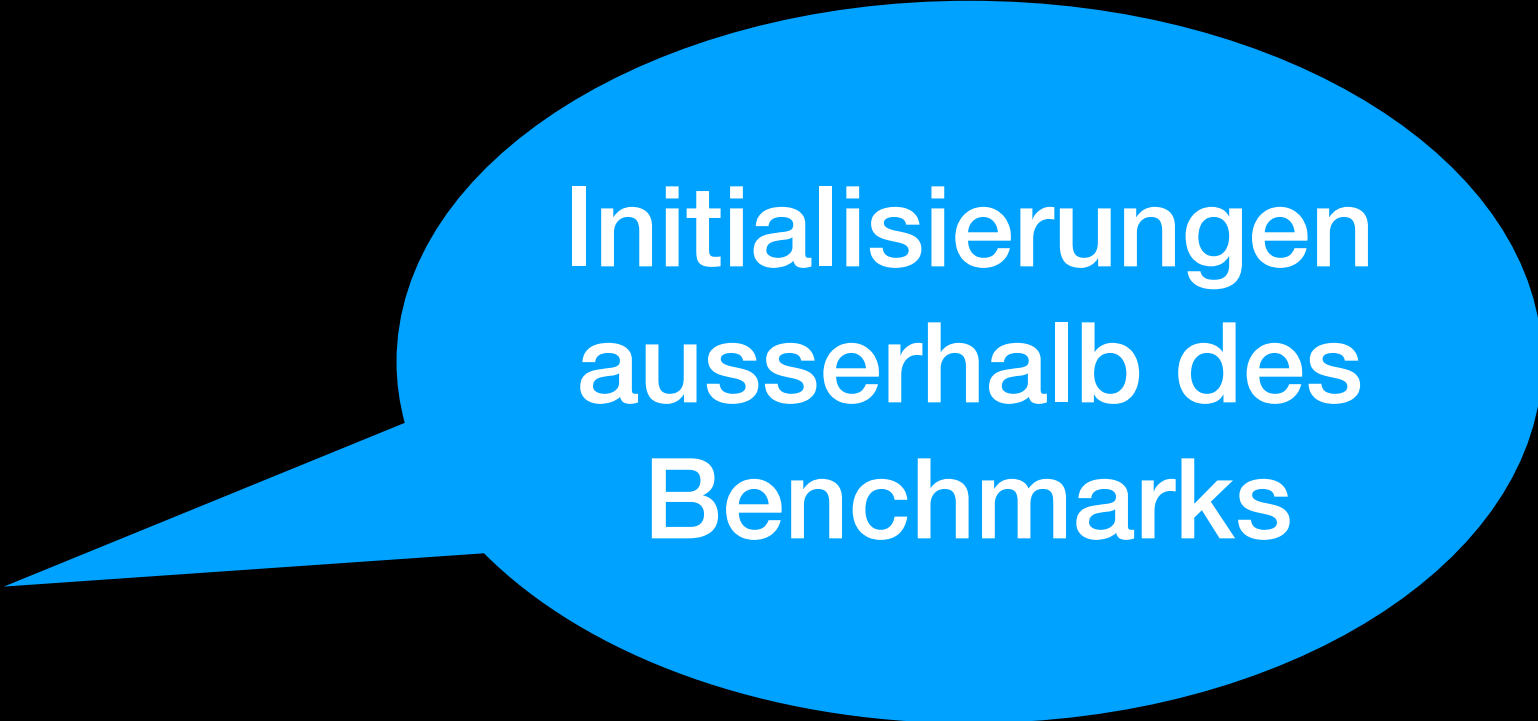
```
    public void setup() {
```

```
        matrixArray = randomMatrixArray();
```

```
        matrixArrayList = randomMatrixArrayList();
```

```
        matrixLinkedList = randomMatrixLinkedList();
```

```
    }
```



Initialisierungen
ausserhalb des
Benchmarks

Code

```
@Benchmark
@BenchmarkMode(Throughput)
@Warmup(iterations = JMH_WARMUP_ITERATIONS, time = JMH_WARMUP_TIME)
@Measurement(iterations = JMH_MEASUREMENT_ITERATIONS,
    time = JMH_MEASUREMENT_TIME)
public int findMaxValueInArrayXthenY() {
    int result = Integer.MIN_VALUE;
    for (int x = 0; x < MATRIX_SIZE; x++) {
        for (int y = 0; y < MATRIX_SIZE; y++) {
            result = result > matrixArray[x][y] ? result : matrixArray[x][y];
        }
    }
    return result;
}
```

Wert zurückgeben! JMH stellt sicher,
dass der Wert «gebraucht» wird.

Compile & Run JMH



```
$ mvn clean package
$ java -jar target/benchmark.jar
...
```

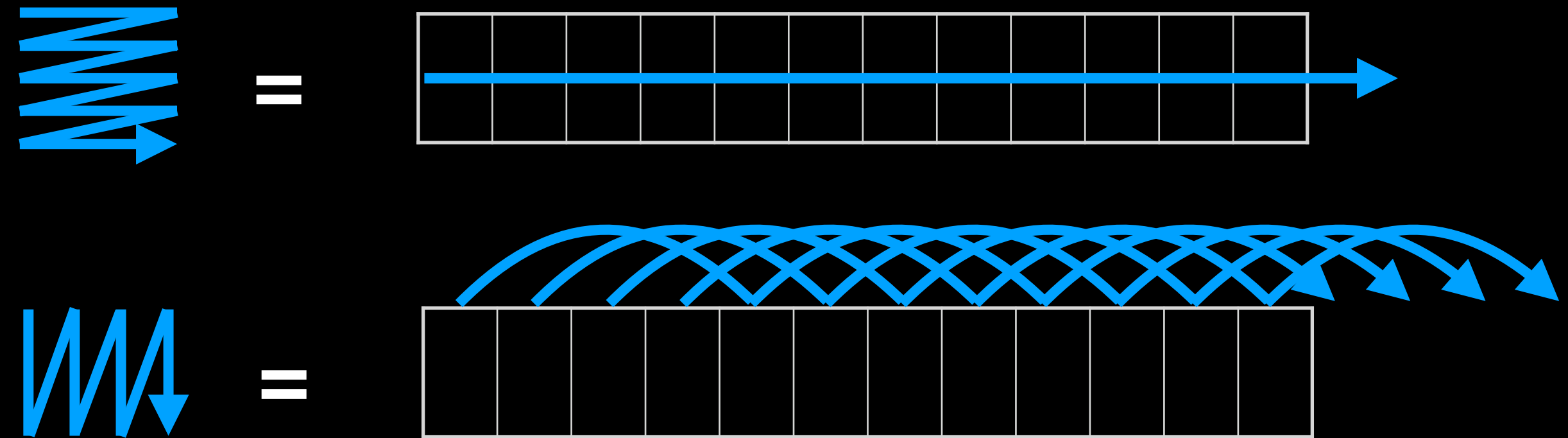
```
# Run complete. Total time: 00:20:12
```

Benchmark	Mode	Cnt	Score	Error	Units
MTB.findMaxValueInArrayXthenY	thrpt	5	3365.809	± 508.158	ops/s
MTB.findMaxValueInArrayListXthenY	thrpt	5	622.822	± 42.106	ops/s
MTB.findMaxValueInArrayYthenX	thrpt	5	561.045	± 52.901	ops/s
MTB.findMaxValueInArrayListYthenX	thrpt	5	141.397	± 17.826	ops/s
MTB.findMaxValueInLinkedListXthenY	thrpt	5	1.152	± 0.017	ops/s
MTB.findMaxValueInLinkedListYthenX	thrpt	5	0.825	± 0.089	ops/s

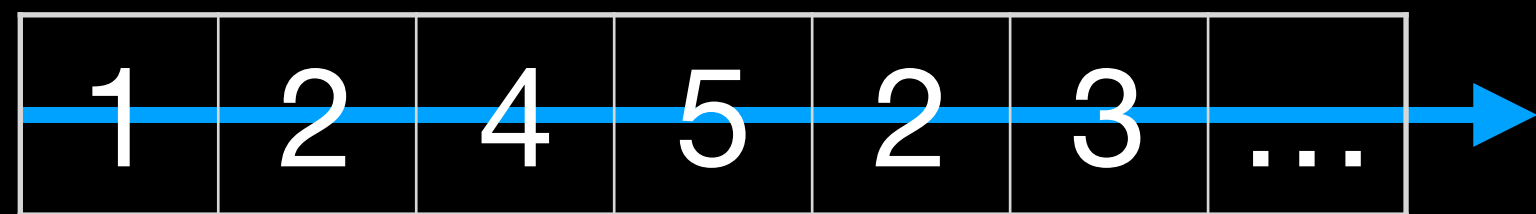
Interpretation

3	7	11	...
8	2	3	...
9	1	42	...
256	6	21	...
...

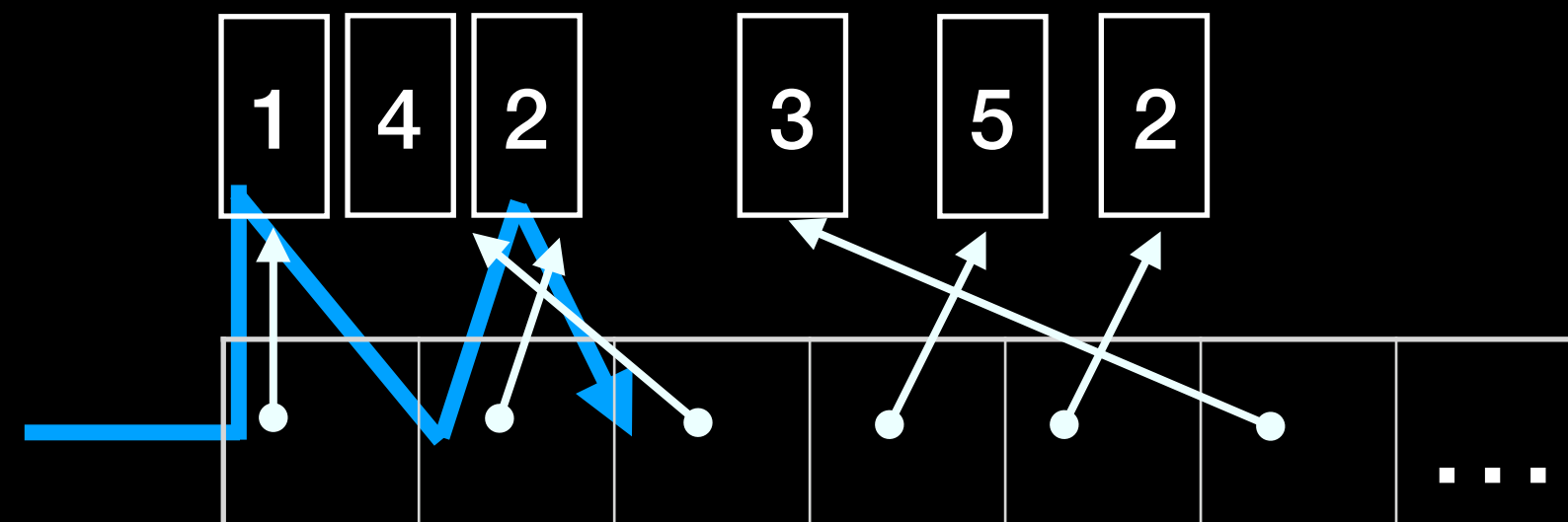
Speicher-Zugriffsmuster



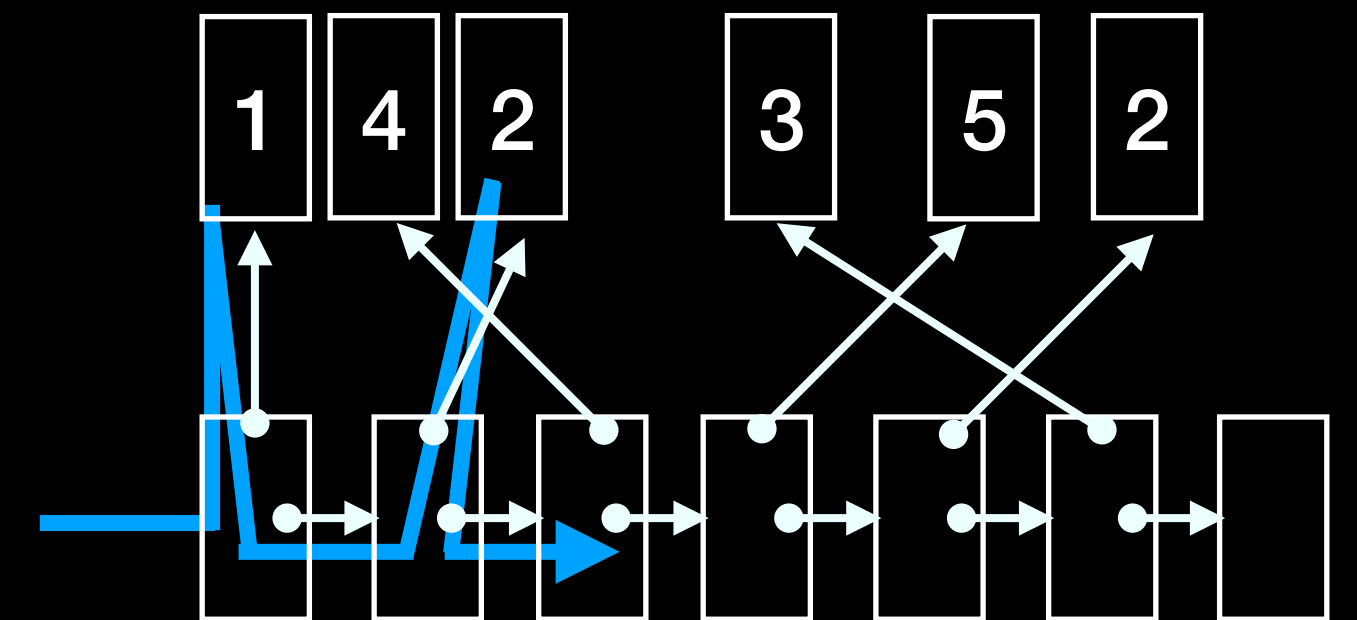
Array



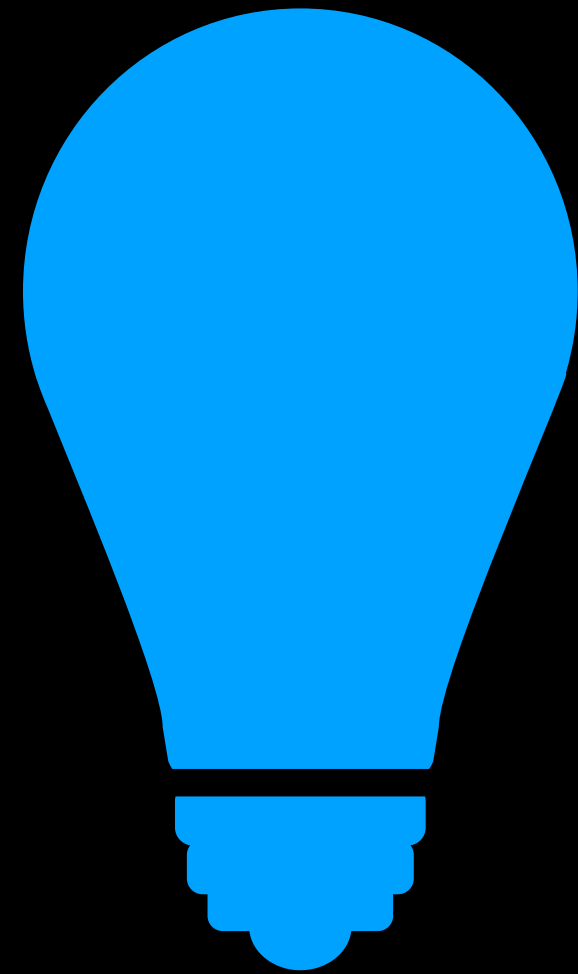
ArrayList



LinkedList



Erkenntnis



**Es ist von Vorteil zu
verstehen, was man tut!**

Links



- [GitHub: techtalk-mechanicalsympathy-example](#)
- [Mechanical Sympathy Blog](#)
- [Mechanical Sympathy Google Group](#)
- [Latency numbers every programmer should know](#)
- [Computer Latency at a Human Scale](#)
- [What Every Programmer Should Know About Memory](#)
- [OpenJDK: JMH](#)
- [Portable Hardware Locality \(hwloc\)](#)