

1BRC



WITH @ROYVANRIJN

**PROCESSING A BILLION ROWS
IN UNDER TWO SECONDS**

A large, brilliant firework explodes in the night sky, creating a massive, dense cloud of golden sparks and light. Below, a street scene is visible with several cars parked along the side and buildings in the background. The scene is illuminated by streetlights and the firework's glow. A dark blue triangular graphic with white symbols is overlaid on the right side of the image.

01/01/2024



"The One Billion Row Challenge"

How fast can YOU aggregate 1B rows using modern [#Java](#)? Grab your threads, flex your SIMD, and kick off 2024 true coder style by joining this friendly little competition. Submissions accepted until Jan 31.

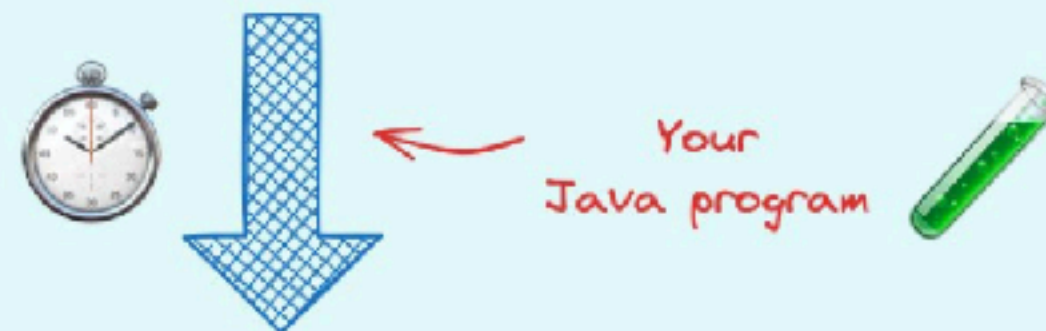
morling.dev/blog/one-billi...

1 🐝🚗 The One Billion Row Challenge

1	Hamburg;12.0
2	Bulawayo;8.9
3	Palembang;28.8
4	St. John's;15.2
5	Cracow;12.6
...	...
1,000,000,000	Helsinki;-2.7

measurements.txt

← 1B temperature values



{ ..., Bulawayo=3.6/18.9/34.0, ..., Helsinki=-21.0/5.9/24.3, ... }

↗ Determine min/mean/max value per station

@gunnarmorling

The challenge:

Parse a file with 1.000.000.000 rows

Print the results

Using plain 'vanilla' Java.



Input:

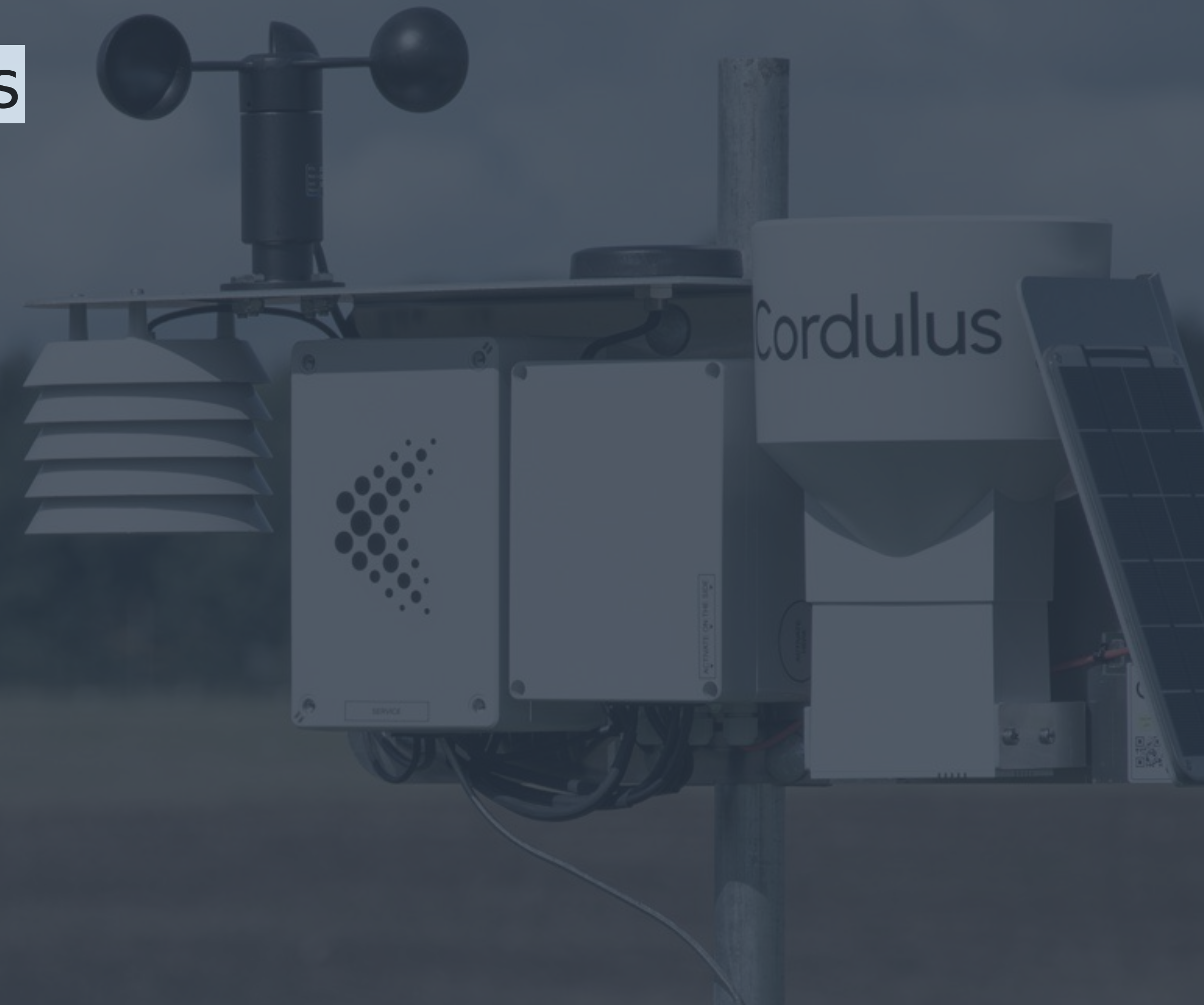
```
<string: station name>;<double: measurement>/n
```

Station name: 1-100 characters

Temperature: -99.9 to 99.9

Example:

Hamburg;	12.0
Bulawayo;	8.9
Palembang;	38.8
St. John's;	15.2
Cracow;	12.6
Bridgetown;	26.9
Istanbul;	6.2
Roseau;	34.4
Conakry;	31.2



Output:

For each station print: **min, mean, max**

Example:

```
{Abha=-23.0/18.0/59.2, Abidjan=-16.2/26.0/67.3,  
Abéché=-10.0/29.4/69.0, Accra=-10.1/26.4/66.4,  
Addis Ababa=-23.7/16.0/67.0,  
Adelaide=-27.8/17.3/58.5, ...}
```

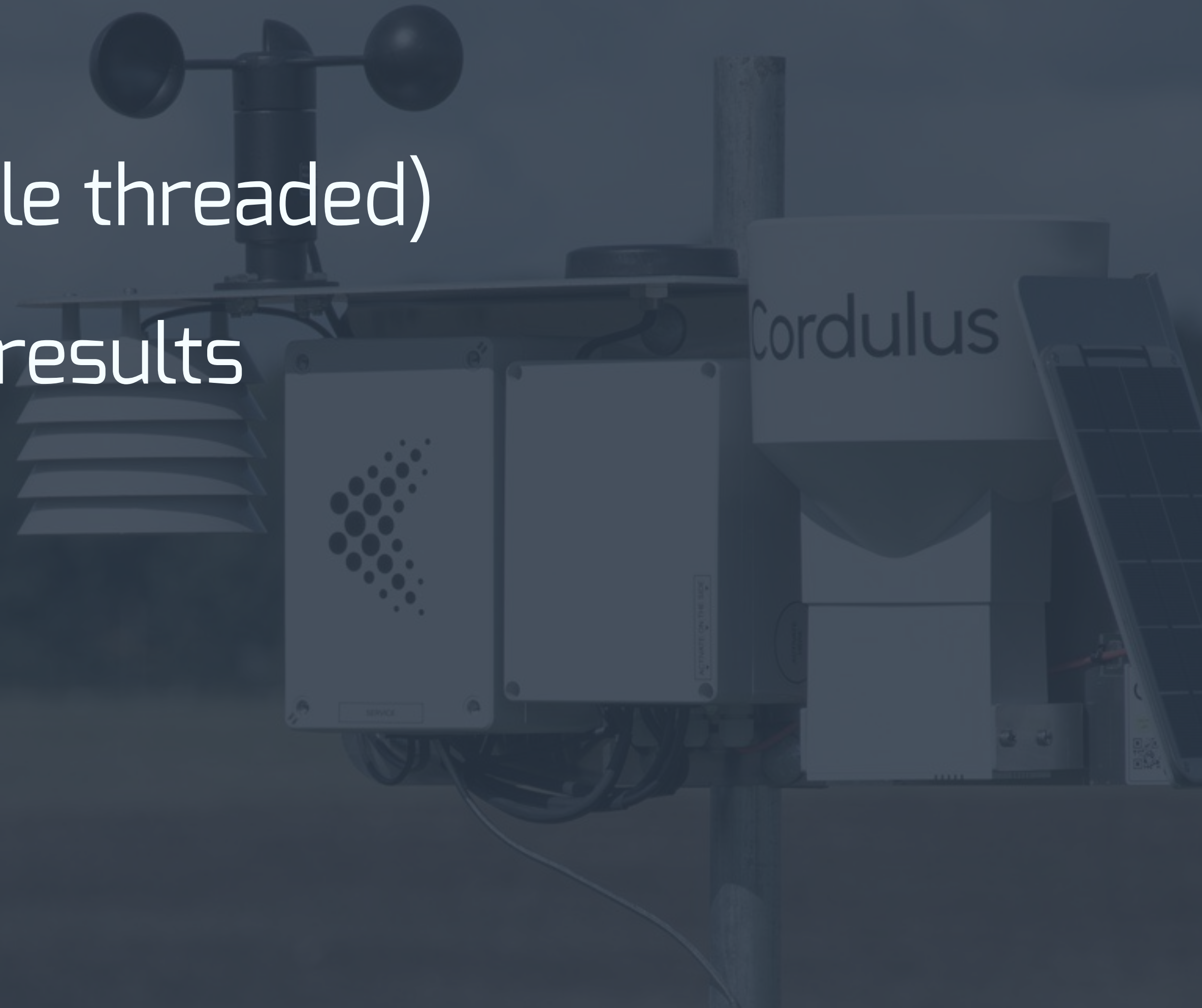
Baseline implementation:

Description:

Uses Files.lines to read (single threaded)

Uses streams to collect the results

Runtime: 04:49.679




```
public static void main(String[] args) throws IOException {
    Collector<Measurement, MeasurementAggregator, ResultRow> collector = Collector.of(
        MeasurementAggregator::new,
        (a, m) -> {
            a.min = Math.min(a.min, m.value);
            a.max = Math.max(a.max, m.value);
            a.sum += m.value;
            a.count++;
        },
        (agg1, agg2) -> {
            var res = new MeasurementAggregator();
            res.min = Math.min(agg1.min, agg2.min);
            res.max = Math.max(agg1.max, agg2.max);
            res.sum = agg1.sum + agg2.sum;
            res.count = agg1.count + agg2.count;

            return res;
        },
        agg -> {
            new ResultRow(agg.min, mean: (Math.round(agg.sum * 10.0) / 10.0) / agg.count, agg.max);
        });

    Map<String, ResultRow> measurements = new TreeMap<>(Files.lines(Paths.get(FILE)) Stream<String>
        .map(l -> new Measurement(l.split(regex: ";"))) Stream<Measurement>
        .collect(groupingBy(m -> m.station(), collector)));

    System.out.println(measurements);
}
```

```
public static void main(String[] args) throws IOException {
    Collector<Measurement, MeasurementAggregator, ResultRow> collector = Collector.of(
        MeasurementAggregator::new,
        (a, m) -> {
            a.min = Math.min(a.min, m.value);
            a.max = Math.max(a.max, m.value);
            a.sum += m.value;
            a.count++;
        },
        (agg1, agg2) -> {
            var res = new MeasurementAggregator();
            res.min = Math.min(agg1.min, agg2.min);
            res.max = Math.max(agg1.max, agg2.max);
            res.sum = agg1.sum + agg2.sum;
            res.count = agg1.count + agg2.count;

            return res;
        },
        agg -> {
            new ResultRow(agg.min, mean: (Math.round(agg.sum * 10.0) / 10.0) / agg.count, agg.max);
        });

    Map<String, ResultRow> measurements = new TreeMap<>(Files.lines(Paths.get(FILE)) Stream<String>
        .map(l -> new Measurement(l.split(regex: ";"))) Stream<Measurement>
        .collect(groupingBy(m -> m.station(), collector)));

    System.out.println(measurements);
}
```

```
public static void main(String[] args) throws IOException {
    Collector<Measurement, MeasurementAggregator, ResultRow> collector = Collector.of(
        MeasurementAggregator::new,
        (a, m) -> {
            a.min = Math.min(a.min, m.value);
            a.max = Math.max(a.max, m.value);
            a.sum += m.value;
            a.count++;
        },
        (agg1, agg2) -> {
            var res = new MeasurementAggregator();
            res.min = Math.min(agg1.min, agg2.min);
            res.max = Math.max(agg1.max, agg2.max);
            res.sum = agg1.sum + agg2.sum;
            res.count = agg1.count + agg2.count;

            return res;
        },
        agg -> {
            new ResultRow(agg.min, mean: (Math.round(agg.sum * 10.0) / 10.0) / agg.count, agg.max);
        });

    Map<String, ResultRow> measurements = new TreeMap<>(Files.lines(Paths.get(FILE)) Stream<String>
        .map(l -> new Measurement(l.split(regex: ";")) Stream<Measurement>
        .collect(groupingBy(m -> m.station(), collector)));

    System.out.println(measurements);
}
```

```
public static void main(String[] args) throws IOException {  
    Collector<Measurement, MeasurementAggregator, ResultRow> collector = Collector.of(  
        MeasurementAggregator::new,  
        (a, m) -> {  
            a.min = Math.min(a.min, m.value);  
            a.max = Math.max(a.max, m.value);  
            a.sum += m.value;  
            a.count++;  
        },  
        (agg1, agg2) -> {  
            var res = new MeasurementAggregator();  
            res.min = Math.min(agg1.min, agg2.min);  
            res.max = Math.max(agg1.max, agg2.max);  
            res.sum = agg1.sum + agg2.sum;  
            res.count = agg1.count + agg2.count;  
  
            return res;  
        },  
        agg -> {  
            new ResultRow(agg.min, mean: (Math.round(agg.sum * 10.0) / 10.0) / agg.count, agg.max);  
        });  
  
    Map<String, ResultRow> measurements = new TreeMap<>(Files.lines(Paths.get(FILE)) Stream<String>  
        .map(l -> new Measurement(l.split(regex: ";")) Stream<Measurement>  
        .collect(groupingBy(m -> m.station(), collector)));  
  
    System.out.println(measurements);  
}
```

```
public static void main(String[] args) throws IOException {
    Collector<Measurement, MeasurementAggregator, ResultRow> collector = Collector.of(
        MeasurementAggregator::new,
        (a, m) -> {
            a.min = Math.min(a.min, m.value);
            a.max = Math.max(a.max, m.value);
            a.sum += m.value;
            a.count++;
        },
        (agg1, agg2) -> {
            var res = new MeasurementAggregator();
            res.min = Math.min(agg1.min, agg2.min);
            res.max = Math.max(agg1.max, agg2.max);
            res.sum = agg1.sum + agg2.sum;
            res.count = agg1.count + agg2.count;

            return res;
        },
        agg -> {
            new ResultRow(agg.min, mean: (Math.round(agg.sum * 10.0) / 10.0) / agg.count, agg.max);
        });

    Map<String, ResultRow> measurements = new TreeMap<>(Files.lines(Paths.get(FILE)) Stream<String>
        .map(l -> new Measurement(l.split(regex: ";"))) Stream<Measurement>
        .collect(groupingBy(m -> m.station(), collector)));

    System.out.println(measurements);
}
```

01 Jan 2024:

Description:

Initial commit (first!)

Uses Files.lines in parallel

Uses ConcurrentHashMap

Runtime: ~~04:49.679~~ > 02:08.845



Watch, learn, adopt, experiment.

1brc / src / main / java / dev / morling / onebrc /



martin2038 and **martin.cong** martin2038: first submission (#665) ✓

Name	Last commit message
..	
CalculateAverage_0xshivamagarwal.java	Add 0xshivamagarwal Implementation (#508)
CalculateAverage_3j5a.java	CalculateAverage_3j5a off-the-shelf Java components + ArraysSupport (#...
CalculateAverage_AbstractKamen.java	CalculateAverage_AbstractKamen
CalculateAverage_C5H12O5.java	Processing byte array backwards (#504)
CalculateAverage_EduardoSaverin.java	EduardoSaverin (#689)
CalculateAverage_JaimePolidura.java	Added improvments on string copying, string comparation & calculation...
CalculateAverage_JamalMulla.java	Second attempt with various improvements (#510)
CalculateAverage_JesseVanRooy.java	CalculateAverage_JesseVanRooy (Submission 1) (#335)
CalculateAverage_Judekeyser.java	Justin's implementation (#666)
CalculateAverage_JurenIvan.java	My first submission (#697)
CalculateAverage_Kidlike.java	submission for kidlike (#294)
CalculateAverage_MahmoudFawzyKhalil.java	MahmoudFawzyKhalil's implementation (#438)
CalculateAverage_MeanderingProgrammer.java	First attempt from MeanderingProgrammer
CalculateAverage_PanagiotisDrakatos.java	My Probably last attempt to optimize performance (#693)
CalculateAverage_PawelAdamski.java	Paweł Adamski - 1brc submission (#629)
CalculateAverage_SamuelYvon.java	Graal Native for SamuelYvon (#332)

```
Map<String, Measurement> resultMap = Files.lines(Path.of(FILE)).parallel() Stream<String>
    .map(record -> {
        // Map to <String,double>
        int pivot = record.indexOf(";");
        String key = record.substring(0, pivot);
        double measured = Double.parseDouble(record.substring(beginIndex: pivot + 1));
        return new AbstractMap.SimpleEntry<>(key, measured);
    }) Stream<SimpleEntry<...>>
    .collect(Collectors.toConcurrentMap(
        // Combine/reduce:
        AbstractMap.SimpleEntry::getKey,
        entry -> new Measurement(entry.getValue()),
        Measurement::combineWith));

System.out.print("{");
System.out.print(
    resultMap.entrySet().stream().sorted(Map.Entry.comparingByKey()).map(Object::toString).collect(Collectors.joining(delimiter: ", ")));
System.out.println("}");
```

```
Map<String, Measurement> resultMap = Files.lines(Path.of(FILE)).parallel() Stream<String>
    .map(record -> {
        // Map to <String,double>
        int pivot = record.indexOf(";");
        String key = record.substring(0, pivot);
        double measured = Double.parseDouble(record.substring(beginIndex: pivot + 1));
        return new AbstractMap.SimpleEntry<>(key, measured);
    }) Stream<SimpleEntry<...>>
    .collect(Collectors.toConcurrentMap(
        // Combine/reduce:
        AbstractMap.SimpleEntry::getKey,
        entry -> new Measurement(entry.getValue()),
        Measurement::combineWith));

System.out.print("{");
System.out.print(
    resultMap.entrySet().stream().sorted(Map.Entry.comparingByKey()).map(Object::toString).collect(Collectors.joining(delimiter: ", ")));
System.out.println("}");
```

```
Map<String, Measurement> resultMap = Files.lines(Path.of(FILE)).parallel() Stream<String>
    .map(record -> {
        // Map to <String,double>
        int pivot = record.indexOf(";");
        String key = record.substring(0, pivot);
        double measured = Double.parseDouble(record.substring(beginIndex: pivot + 1));
        return new AbstractMap.SimpleEntry<>(key, measured);
    }) Stream<SimpleEntry<...>>
    .collect(Collectors.toConcurrentMap(
        // Combine/reduce:
        AbstractMap.SimpleEntry::getKey,
        entry -> new Measurement(entry.getValue()),
        Measurement::combineWith));

System.out.print("{");
System.out.print(
    resultMap.entrySet().stream().sorted(Map.Entry.comparingByKey()).map(Object::toString).collect(Collectors.joining(delimiter: ", ")));
System.out.println("}");
```

```
Map<String, Measurement> resultMap = Files.lines(Path.of(FILE)).parallel() Stream<String>
    .map(record -> {
        // Map to <String,double>
        int pivot = record.indexOf(";");
        String key = record.substring(0, pivot);
        double measured = Double.parseDouble(record.substring(beginIndex: pivot + 1));
        return new AbstractMap.SimpleEntry<>(key, measured);
    }) Stream<SimpleEntry<>>
    .collect(Collectors.toConcurrentMap(
        // Combine/reduce:
        AbstractMap.SimpleEntry::getKey,
        entry -> new Measurement(entry.getValue()),
        Measurement::combineWith));

System.out.print("{");
System.out.print(
    resultMap.entrySet().stream().sorted(Map.Entry.comparingByKey()).map(Object::toString).collect(Collectors.joining(delimiter: ", ")));
System.out.println("}");
```

```
Map<String, Measurement> resultMap = Files.lines(Path.of(FILE)).parallel() Stream<String>
    .map(record -> {
        // Map to <String,double>
        int pivot = record.indexOf(";");
        String key = record.substring(0, pivot);
        double measured = Double.parseDouble(record.substring(beginIndex: pivot + 1));
        return new AbstractMap.SimpleEntry<>(key, measured);
    }) Stream<SimpleEntry<...>>
    .collect(Collectors.toConcurrentMap(
        // Combine/reduce:
        AbstractMap.SimpleEntry::getKey,
        entry -> new Measurement(entry.getValue()),
        Measurement::combineWith));
```

```
System.out.print("{");
```

```
System.out.print(
    resultMap.entrySet().stream().sorted(Map.Entry.comparingByKey()).map(Object::toString).collect(Collectors.joining(delimiter: ", ")));
System.out.println("}");
```

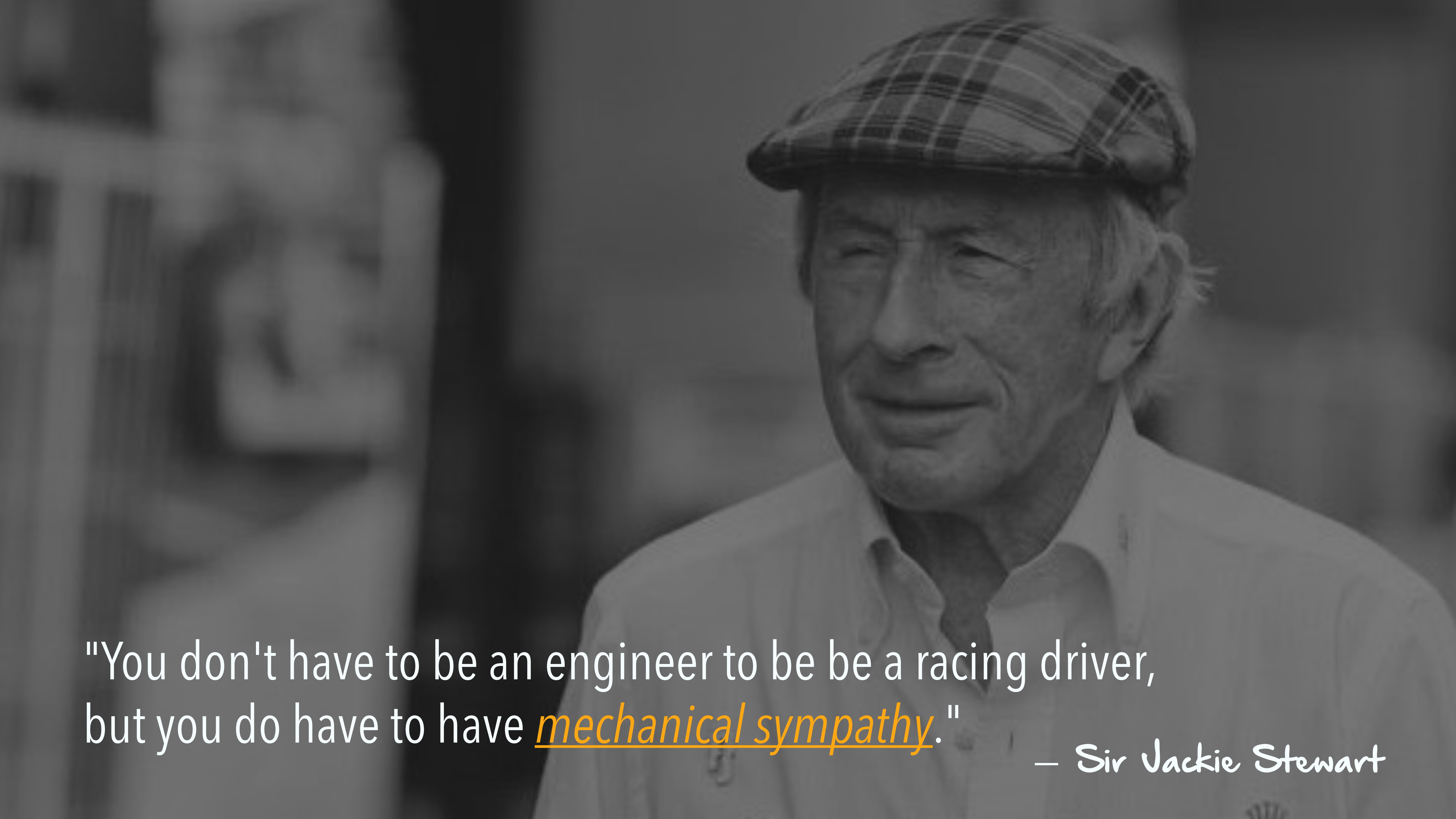
03 Jan 2024

Description:

A **lot** has happened. →

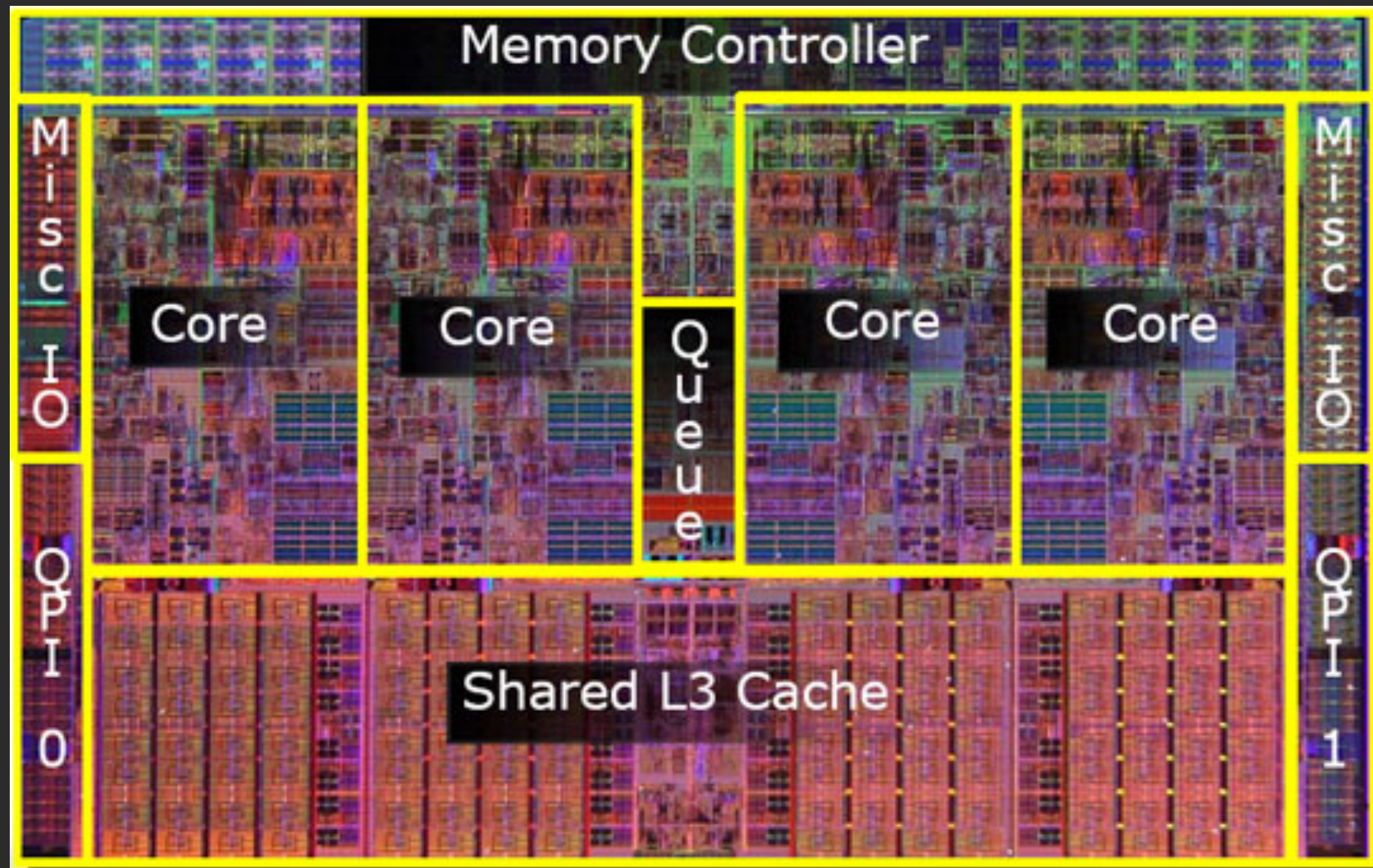
```
/**
 * Changelog:
 *
 * Initial submission:          62000 ms
 * Chunked reader:              16000 ms
 * Optimized parser:            13000 ms
 * Branchless methods:          11000 ms
 * Adding memory mapped files:  6500 ms (based on bjhara's submission)
 * Skipping string creation:     4700 ms
 * Custom hashmap...             4200 ms
 * Added SWAR token checks:     3900 ms
 * Skipped String creation:      3500 ms (idea from kgonia)
 * Improved String skip:         3250 ms
 * Segmenting files:             3150 ms (based on spullara's code)
 * Not using SWAR for EOL:      2850 ms
 *
 * Best performing JVM on MacBook M2 Pro: 21.0.1-graal
 * `sdk use java 21.0.1-graal`
 *
 */
```

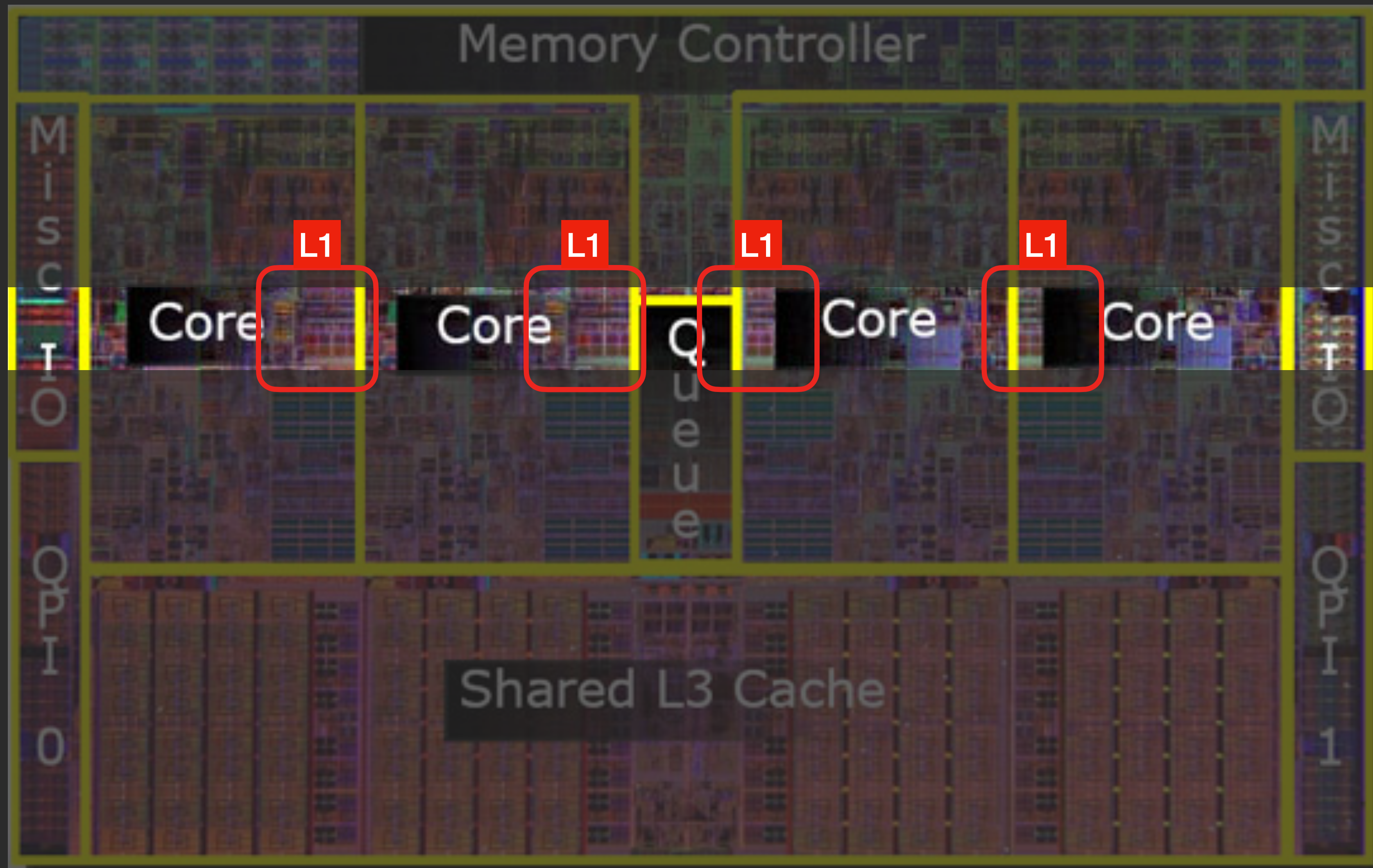
Runtime: ~~02:08.845~~ > 00:23.366

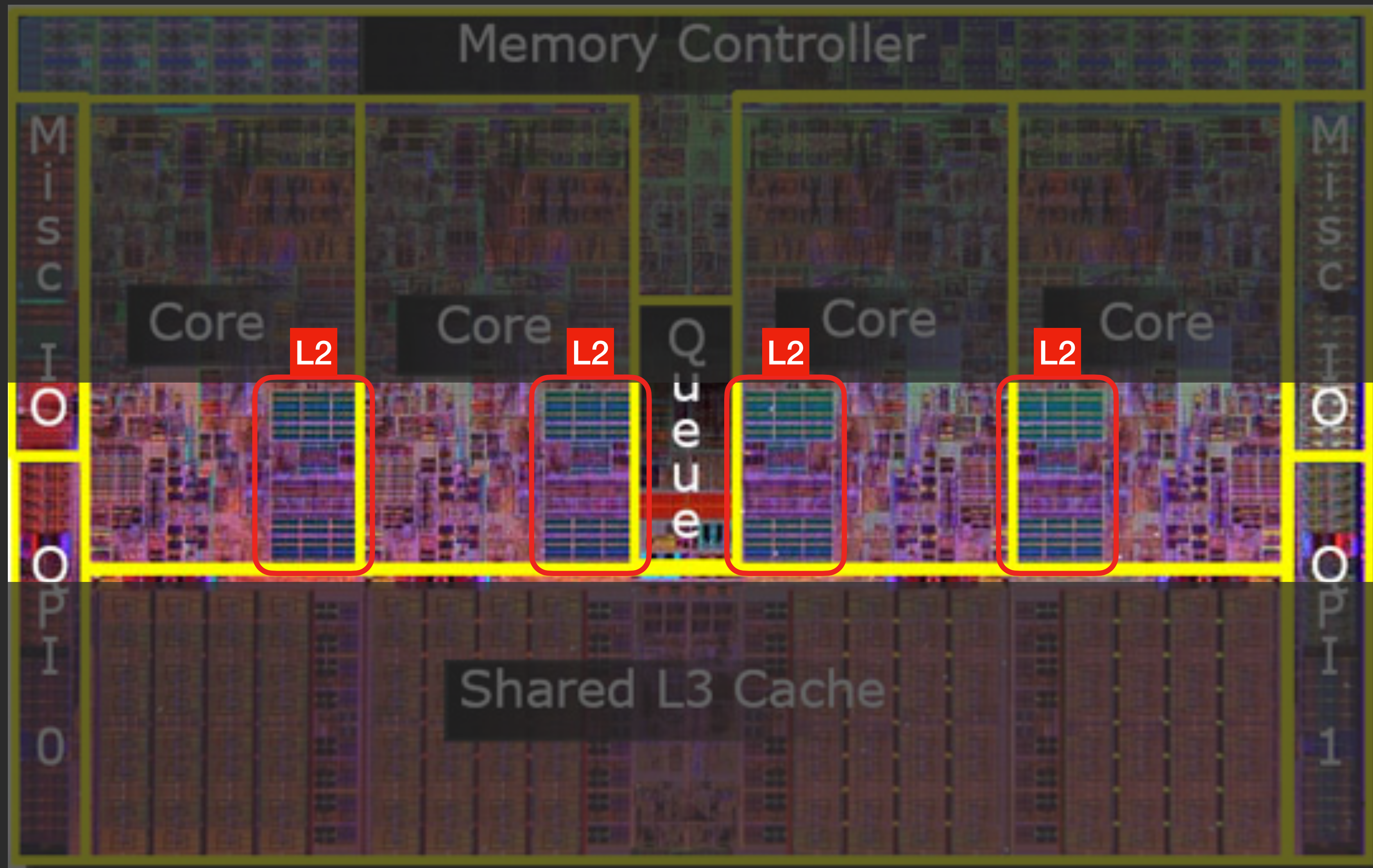


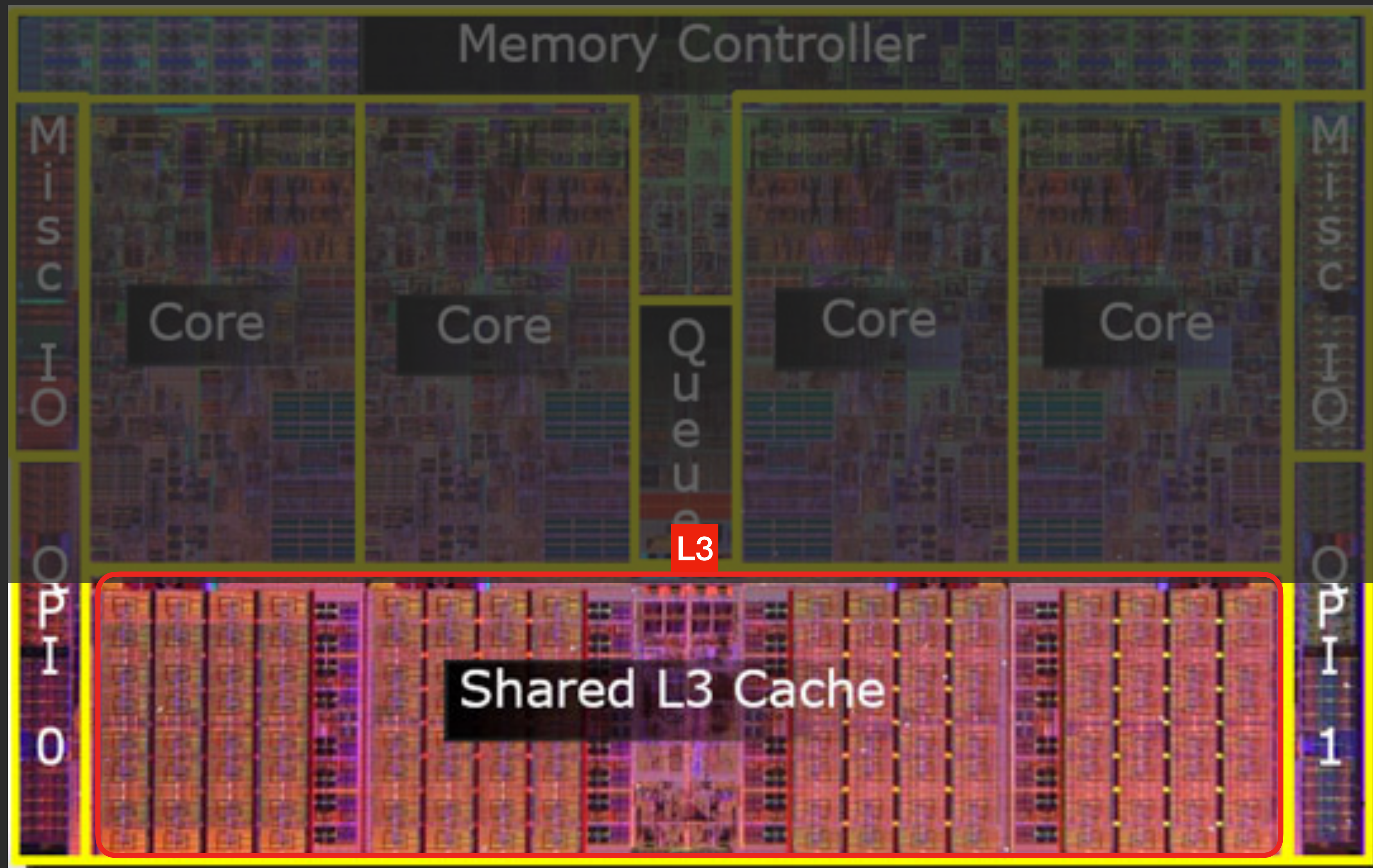
"You don't have to be an engineer to be a racing driver,
but you do have to have mechanical sympathy."

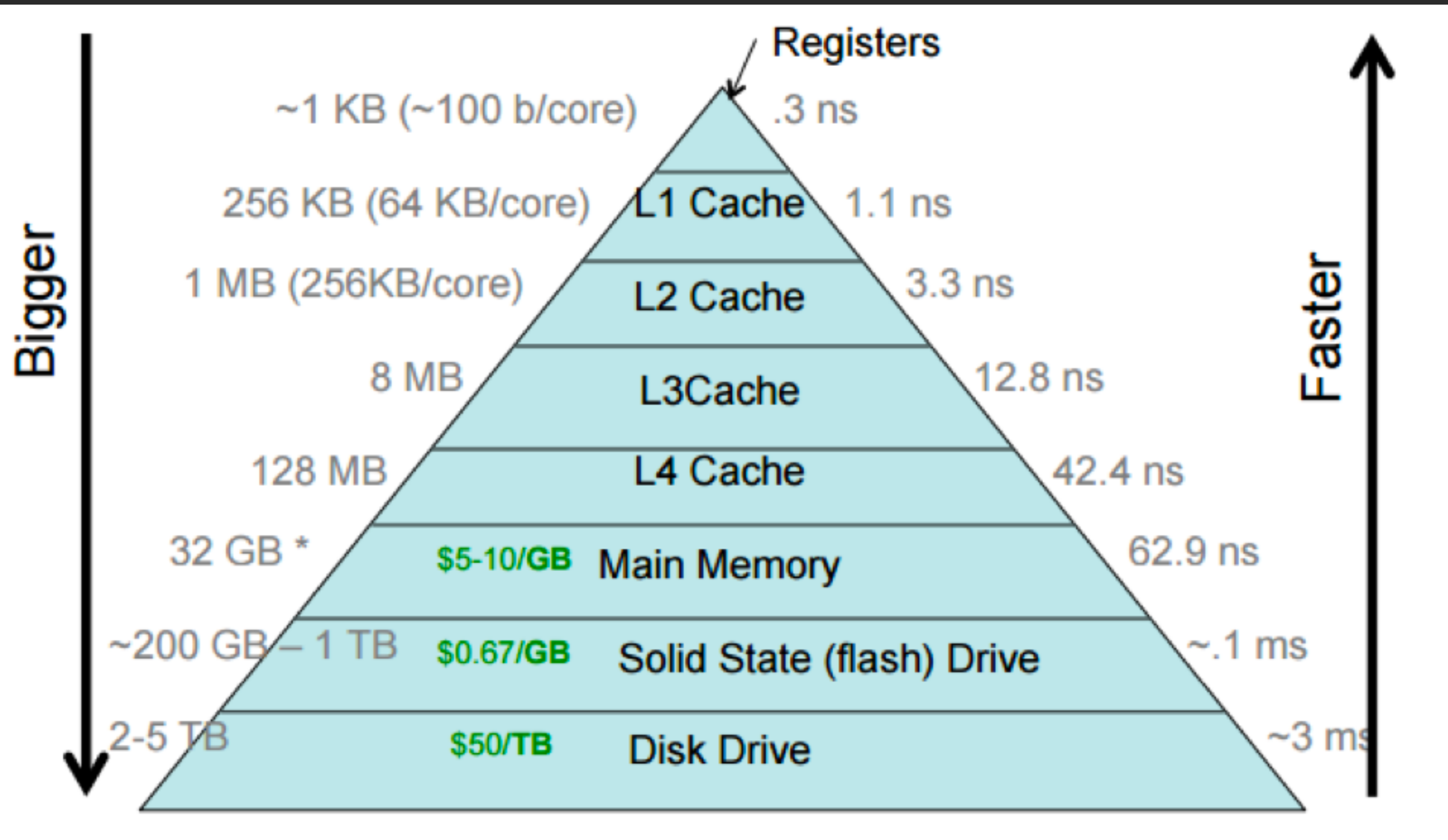
— Sir Jackie Stewart











Temperature as integer

6 usages new *

```
static final class Measurement {
```

7 usages

```
int min, max, count;
```

4 usages

```
long sum;
```

new *

```
public Measurement() {
```

```
    this.min = 10000;
```

```
    this.max = -10000;
```

```
}
```

no usages new *

```
public Measurement updateWith(int measurement) {
```

```
    min = min(min, measurement);
```

```
    max = max(max, measurement);
```

```
    sum += measurement;
```

```
    count++;
```

```
    return this;
```

```
}
```

1 usage new *

```
public Measurement updateWith(Measurement measurement) {
```

```
    min = min(min, measurement.min);
```

```
    max = max(max, measurement.max);
```

```
    sum += measurement.sum;
```

```
    count += measurement.count;
```

3 usages new *

```
record FileSegment(long start, long end) {  
}
```

*/** Using this way to segment the file is much prettier, from spullara */*

1 usage new *

```
private static List<FileSegment> getFileSegments(File file) throws IOException {  
    final int numberOfSegments = Runtime.getRuntime().availableProcessors();  
    final long fileSize = file.length();  
    final long segmentSize = fileSize / numberOfSegments;  
    final List<FileSegment> segments = new ArrayList<>();  
    try (RandomAccessFile randomAccessFile = new RandomAccessFile(file, mode: "r")) {  
        for (int i = 0; i < numberOfSegments; i++) {  
            long segStart = i * segmentSize;  
            long segEnd = (i == numberOfSegments - 1) ? fileSize : segStart + segmentSize;  
            segStart = findSegment(i, skipSegment: 0, randomAccessFile, segStart, segEnd);  
            segEnd = findSegment(i, skipSegment: numberOfSegments - 1, randomAccessFile, segEnd, fileSize);  
  
            segments.add(new FileSegment(segStart, segEnd));  
        }  
    }  
    return segments;  
}
```

3 usages new *

```
record FileSegment(long start, long end) {  
}
```

*/** Using this way to segment the file is much prettier, from spullara */*

1 usage new *

```
private static List<FileSegment> getFileSegments(File file) throws IOException {  
    final int numberOfSegments = Runtime.getRuntime().availableProcessors();  
    final long fileSize = file.length();  
    final long segmentSize = fileSize / numberOfSegments;  
    final List<FileSegment> segments = new ArrayList<>();  
    try (RandomAccessFile randomAccessFile = new RandomAccessFile(file, mode: "r")) {  
        for (int i = 0; i < numberOfSegments; i++) {  
            long segStart = i * segmentSize;  
            long segEnd = (i == numberOfSegments - 1) ? fileSize : segStart + segmentSize;  
            segStart = findSegment(i, skipSegment: 0, randomAccessFile, segStart, segEnd);  
            segEnd = findSegment(i, skipSegment: numberOfSegments - 1, randomAccessFile, segEnd, fileSize);  
  
            segments.add(new FileSegment(segStart, segEnd));  
        }  
    }  
    return segments;  
}
```

3 usages new *

```
record FileSegment(long start, long end) {  
}
```

*/** Using this way to segment the file is much prettier, from spullara */*

1 usage new *

```
private static List<FileSegment> getFileSegments(File file) throws IOException {  
    final int numberOfSegments = Runtime.getRuntime().availableProcessors();  
    final long fileSize = file.length();  
    final long segmentSize = fileSize / numberOfSegments;  
    final List<FileSegment> segments = new ArrayList<>();  
    try (RandomAccessFile randomAccessFile = new RandomAccessFile(file, mode: "r")) {  
        for (int i = 0; i < numberOfSegments; i++) {  
            long segStart = i * segmentSize;  
            long segEnd = (i == numberOfSegments - 1) ? fileSize : segStart + segmentSize;  
            segStart = findSegment(i, skipSegment: 0, randomAccessFile, segStart, segEnd);  
            segEnd = findSegment(i, skipSegment: numberOfSegments - 1, randomAccessFile, segEnd, fileSize);  
  
            segments.add(new FileSegment(segStart, segEnd));  
        }  
    }  
    return segments;  
}
```

Memory mapped files

```
var results = getFileSegments(new File(FILE)).stream().map(segment -> {

    long segmentEnd = segment.end();
    try (var fileChannel = (FileChannel) Files.newByteChannel(Path.of(FILE), StandardOpenOption.READ)) {
        var bb = fileChannel.map(FileChannel.MapMode.READ_ONLY, segment.start(), size: segmentEnd - segment.start());

        int startPointer;
        int limit = bb.limit();
        while ((startPointer = bb.position()) < limit) {

            /// omitted.... parse entry
        }
        return null; // entries
    }
    catch (IOException e) {
        throw new RuntimeException(e);
    }
}).parallel().flatMap(v -> v.values.stream())
    .collect(Collectors.toMap(e -> new String(e.key), BitTwiddledMap.Entry::measurement, (m1, m2) -> m1.updateWith(m2), TreeMap::new));

// Seems to perform better than actually using a TreeMap:
System.out.println(results);
```

Memory mapped files

```
var results = getFileSegments(new File(FILE)).stream().map(segment -> {  
  
    long segmentEnd = segment.end();  
    try (var fileChannel = (FileChannel) Files.newByteChannel(Path.of(FILE), StandardOpenOption.READ)) {  
        var bb = fileChannel.map(FileChannel.MapMode.READ_ONLY, segment.start(), size: segmentEnd - segment.start());  
  
        int startPointer;  
        int limit = bb.limit();  
        while ((startPointer = bb.position()) < limit) {  
  
            /// omitted.... parse entry  
        }  
        return null; // entries  
    }  
    catch (IOException e) {  
        throw new RuntimeException(e);  
    }  
}).parallel().flatMap(v -> v.values.stream())  
    .collect(Collectors.toMap(e -> new String(e.key), BitTwiddledMap.Entry::measurement, (m1, m2) -> m1.updateWith(m2), TreeMap::new));  
  
// Seems to perform better than actually using a TreeMap:  
System.out.println(results);
```

UNSAFE



Unsafe allows direct memory access (like in C)

Java wants to move away from Unsafe

Alternatives are really good (ByteBuffer etc) but...

GETTING UNSAFE



49 usages

```
private static final Unsafe UNSAFE = initUnsafe();
```

1 usage new *

```
private static Unsafe initUnsafe() {  
    try {  
        final Field theUnsafe = Unsafe.class.getDeclaredField(name: "theUnsafe");  
        theUnsafe.setAccessible(true);  
        return (Unsafe) theUnsafe.get(Unsafe.class);  
    }  
    catch (NoSuchFieldException | IllegalAccessException e) {  
        throw new RuntimeException(e);  
    }  
}
```

USING UNSAFE



```
final long mapAddress = fileChannel.map(FileChannel.MapMode.READ_ONLY, offset: 0, fileSize, Arena.global()).address();

// Read the first byte:
UNSAFE.getBytes(mapAddress);
```

SWAR

Instead of processing each input character separately...

Why not do 8 at a time?

SWAR = SIMD As A Register

SIMD = Single Instruction/Multiple Data

SWAR

1) We read 8 bytes as a 64 bit long.

```
Tokio;-18.3\nAmsterdam;31.2...
```

SWAR

1) We read 8 bytes as a 64 bit long.

```
Tokio;-18.3\nAmsterdam;31.2...
```

```
1-;oikoT > 0x312D3B6F696B6F54
```

SWAR

1) We read 8 bytes as a 64 bit long.

```
Tokio;-18.3\nAmsterdam;31.2...
```

```
1-;oikoT > 0x312D3B6F696B6F54
```

2) Make a mask for the separator we're looking for:

```
;;;;; > 0x3B3B3B3B3B3B3B3B
```

SWAR

`1-;oikoT` > `0x312D3B6F696B6F54`
`;;;;;;;` > `0x3B3B3B3B3B3B3B3B`

```
1 usage new *
private static int firstAnyPattern(long word, long pattern) {
    final long match = word ^ pattern;
    long mask = match - 0x0101010101010101L;
    mask &= ~match;
    mask &= 0x8080808080808080L;
    return Long.numberOfTrailingZeros(mask) >>> 3;
}
```

SWAR

0x312D3B6F696B6F54
0x3B3B3B3B3B3B3B3B

^

```
00110001 00101101 00111011 01101111 01101001 01101011 01101111 01010100
00111011 00111011 00111011 00111011 00111011 00111011 00111011 00111011
00001010 00010110 00000000 01010100 01010010 01010000 01010100 01101111
```

1 usage new *

```
private static int firstAnyPattern(long word, long pattern) {
    final long match = word ^ pattern;
    long mask = match - 0x0101010101010101L;
    mask &= ~match;
    mask &= 0x8080808080808080L;
    return Long.numberOfTrailingZeros(mask) >>> 3;
}
```

SWAR

match	=	00001010	00010110	00000000	01010100	01010010	01010000	01010100	01101111
0x0101010101010101L		00000001	00000001	00000001	00000001	00000001	00000001	00000001	00000001

1 usage new *

```
private static int firstAnyPattern(long word, long pattern) {  
    final long match = word ^ pattern;  
    long mask = match - 0x0101010101010101L;  
    mask &= ~match;  
    mask &= 0x8080808080808080L;  
    return Long.numberOfTrailingZeros(mask) >>> 3;  
}
```

SWAR

	match	-	00001010	00010110	00000000	01010100	01010010	01010000	01010100	01101111
0x0101010101010101L			00000001	00000001	00000001	00000001	00000001	00000001	00000001	00000001
	mask		00001001	00010101	11111111	01010010	01010001	01001111	01010011	01101110

1 usage new *

```
private static int firstAnyPattern(long word, long pattern) {  
    final long match = word ^ pattern;  
    long mask = match - 0x0101010101010101L;  
    mask &= ~match;  
    mask &= 0x8080808080808080L;  
    return Long.numberOfTrailingZeros(mask) >>> 3;  
}
```

SWAR

mask	00001001 00010101 11111111 01010010 01010001 01001111 01010011 01101110
~match	11110101 11101001 11111111 10101011 10101101 10101111 10101011 10010000

1 usage new *

```
private static int firstAnyPattern(long word, long pattern) {  
    final long match = word ^ pattern;  
    long mask = match - 0x0101010101010101L;  
    mask &= ~match;  
    mask &= 0x8080808080808080L;  
    return Long.numberOfTrailingZeros(mask) >>> 3;  
}
```

SWAR

mask	00001001 00010101 11111111 01010010 01010001 01001111 01010011 01101110
~match	11110101 11101001 11111111 10101011 10101101 10101111 10101011 10010000
	00000001 00000001 11111111 00000010 00000001 00001111 00000011 00000000

1 usage new *

```
private static int firstAnyPattern(long word, long pattern) {  
    final long match = word ^ pattern;  
    long mask = match - 0x0101010101010101L;  
    mask &= ~match;  
    mask &= 0x8080808080808080L;  
    return Long.numberOfTrailingZeros(mask) >>> 3;  
}
```

SWAR

mask	00000001 00000001 11111111 00000010 00000001 00001111 00000011 00000000
0x8080808080808080	10000000 10000000 10000000 10000000 10000000 10000000 10000000 10000000

1 usage new *

```
private static int firstAnyPattern(long word, long pattern) {  
    final long match = word ^ pattern;  
    long mask = match - 0x0101010101010101L;  
    mask &= ~match;  
    mask &= 0x8080808080808080L;  
    return Long.numberOfTrailingZeros(mask) >>> 3;  
}
```

SWAR

mask	00000001 00000001 11111111 00000010 00000001 00001111 00000011 00000000
0x8080808080808080	10000000 10000000 10000000 10000000 10000000 10000000 10000000 10000000
	00000000 00000000 <u>10000000</u> 00000000 00000000 00000000 00000000 00000000

1 usage new *

```
private static int firstAnyPattern(long word, long pattern) {  
    final long match = word ^ pattern;  
    long mask = match - 0x0101010101010101L;  
    mask &= ~match;  
    mask &= 0x8080808080808080L;  
    return Long.numberOfTrailingZeros(mask) >>> 3;  
}
```

SWAR

00000000 00000000 10000000 00000000 00000000 00000000 00000000 00000000

= 47

$47 \gg 3 = 47/8 = 5$

Tokio;-18.3\n

1 usage new *

```
private static int firstAnyPattern(long word, long pattern) {  
    final long match = word ^ pattern;  
    long mask = match - 0x0101010101010101L;  
    mask &= ~match;  
    mask &= 0x8080808080808080L;  
    return Long.numberOfTrailingZeros(mask) >>> 3;  
}
```

Stringless...

Instead of allocating, storing and comparing a **String**...

We can check **8** bytes of a **String** using:

```
return UNSAFE.getLong(address1) != UNSAFE.getLong(address2);
```



Branchless programming

CPU's have big pipelines of instructions

Your code needs to fill this pipeline, fast.

Which instruction comes 'next' after an IF-statement?

Branches are often bottlenecks.

ABS, WITH BRANCHES

```
int absoluteValue(int x) {  
    if (x < 0) {  
        return -x;  
    } else {  
        return x;  
    }  
}
```

ABS, BRANCHLESS

```
int absoluteValueBranchless(int x) {  
    int mask = x >> 31; // Right shift by 31 bits to fill with sign bit  
    return (x + mask) ^ mask;  
}
```

Parse the temperature

1 usage new *

```
private static int branchlessParseInt(final byte[] input, int start, int length) {  
    // 0 if positive, 1 if negative  
    final int negative = ~(input[start] >> 4) & 1;  
    // 0 if nr length is 3, 1 if length is 4  
    final int has4 = ((length - negative) >> 2) & 1;  
  
    final int digit1 = input[start + negative] - '0';  
    final int digit2 = input[start + negative + has4];  
    final int digit3 = input[start + negative + has4 + 2];  
  
    return (-negative ^ (has4 * (digit1 * 100) + digit2 * 10 + digit3 - 528) - negative); // 528 == ('0' * 10 + '0')  
}
```

Parse the temperature

```
1 usage new *
```

```
private static int branchlessParseInt(final byte[] input, int start, int length) {
```

```
    // 0 if positive, 1 if negative
```

```
    final int negative = ~(input[start] >> 4) & 1;
```

```
    // 0 if nr length is 3, 1 if length is 4
```

```
    final int has4 = ((length - negative) >> 2) & 1;
```

```
    final int digit1 = input[start + negative] - '0';
```

```
    final int digit2 = input[start + negative + has4];
```

```
    final int digit3 = input[start + negative + has4 + 2];
```

```
    return (-negative ^ (has4 * (digit1 * 100) + digit2 * 10 + digit3 - 528) - negative); // 528 == ('0' * 10 + '0')
```

```
}
```

-	=	101101
.	=	101110
0	=	110000
1	=	110001
2	=	110010
3	=	110011
4	=	110100
5	=	110101
6	=	110110
7	=	110111
8	=	111000
9	=	111001

Parse the temperature

1 usage new *

```
private static int branchlessParseInt(final byte[] input, int start, int length) {  
    // 0 if positive, 1 if negative  
    final int negative = ~(input[start] >> 4) & 1;  
    // 0 if nr length is 3, 1 if length is 4  
    final int has4 = ((length - negative) >> 2) & 1;  
  
    final int digit1 = input[start + negative] - '0';  
    final int digit2 = input[start + negative + has4];  
    final int digit3 = input[start + negative + has4 + 2];  
  
    return (-negative ^ (has4 * (digit1 * 100) + digit2 * 10 + digit3 - 528) - negative); // 528 == ('0' * 10 + '0')  
}
```

Parse the temperature

1 usage new *

```
private static int branchlessParseInt(final byte[] input, int start, int length) {  
    // 0 if positive, 1 if negative  
    final int negative = ~(input[start] >> 4) & 1;  
    // 0 if nr length is 3, 1 if length is 4  
    final int has4 = ((length - negative) >> 2) & 1;  
  
    final int digit1 = input[start + negative] - '0';  
    final int digit2 = input[start + negative + has4];  
    final int digit3 = input[start + negative + has4 + 2];  
  
    return (-negative ^ (has4 * (digit1 * 100) + digit2 * 10 + digit3 - 528) - negative); // 528 == ('0' * 10 + '0')  
}
```

Parse the temperature

1 usage new *

```
private static int branchlessParseInt(final byte[] input, int start, int length) {  
    // 0 if positive, 1 if negative  
    final int negative = ~(input[start] >> 4) & 1;  
    // 0 if nr length is 3, 1 if length is 4  
    final int has4 = ((length - negative) >> 2) & 1;  
  
    final int digit1 = input[start + negative] - '0';  
    final int digit2 = input[start + negative + has4];  
    final int digit3 = input[start + negative + has4 + 2];  
  
    return (-negative ^ (has4 * (digit1 * 100) + digit2 * 10 + digit3 - 528) - negative); // 528 == ('0' * 10 + '0')  
}
```

Parse the temperature



USER **MERYKITTY** ENTERED THE CHAT

1 usage

```
private static final long DOT_BITS = 0x10101000;
```

1 usage

```
private static final long MAGIC_MULTIPLIER = (100 * 0x10000000 + 10 * 0x10000 + 1);
```

// Awesome idea of [merykitty](#):

2 usages  Roy van Rijn

```
private int readTemperature() {
```

```
    // This is the number part: X.X, -X.X, XX.x or -XX.X
```

```
    final long numberBytes = UNSAFE.getLong(ptr);
```

```
    final long invNumberBytes = ~numberBytes;
```

```
    final int dotPosition = Long.numberOfTrailingZeros(invNumberBytes & DOT_BITS);
```

```
    // Calculates the sign
```

```
    final long signed = (invNumberBytes << 59) >> 63;
```

```
    final int min28 = (dotPosition ^ 0b11100);
```

```
    final long minusFilter = ~(signed & 0xFF);
```

```
    // Use the pre-calculated decimal position to adjust the values
```

```
    final long digits = ((numberBytes & minusFilter) << min28) & 0x0F000F0F00L;
```

```
    // Update the pointer here, bit awkward, but we have all the data
```

```
    ptr += (dotPosition >> 3) + 3;
```

```
    // Multiply by a magic (100 * 0x10000000 + 10 * 0x10000 + 1), to get the result
```

```
    final long absValue = ((digits * MAGIC_MULTIPLIER) >>> 32) & 0x3FF;
```

```
    // And perform abs()
```

```
    return (int) ((absValue + signed) ^ signed); // non-patented method of doing the same trick
```

```
}
```

1 usage

```
private static final long DOT_BITS = 0x10101000;
```

1 usage

```
private static final long MAGIC_MULTIPLIER = (100 * 0x1000000 + 10 * 0x10000 + 1);
```

```
// Awesome idea of merykitty:
```

2 usages  Roy van Rijn

```
private int readTemperature() {
```

```
    // This is the number part: X.X, X.X, XX.X or -XX.X
```

```
    final long numberBytes = UNSAFE.getLong(ptr);
```

```
    final long invNumberBytes = ~numberBytes;
```

```
    final int dotPosition = Long.numberOfTrailingZeros(invNumberBytes & DOT_BITS);
```

```
    // Calculates the sign
```

```
    final long signed = (invNumberBytes << 59) >> 63;
```

```
    final int min28 = (dotPosition ^ 0b11100);
```

```
    final long minusFilter = ~(signed & 0xFF);
```

```
    // Use the pre-calculated decimal position to adjust the values
```

```
    final long digits = ((numberBytes & minusFilter) << min28) & 0x0F000F0F00L;
```

```
    // Update the pointer here, bit awkward, but we have all the data
```

```
    ptr += (dotPosition >> 3) + 3;
```

```
    // Multiply by a magic (100 * 0x1000000 + 10 * 0x10000 + 1), to get the result
```

```
    final long absValue = ((digits * MAGIC_MULTIPLIER) >>> 32) & 0x3FF;
```

```
    // And perform abs()
```

```
    return (int) ((absValue + signed) ^ signed); // non-patented method of doing the same trick
```

```
}
```

```

1 usage
private static final long DOT_BITS = 0x10101000;
1 usage
private static final long MAGIC_MULTIPLIER = (100 * 0x1000000 + 10 * 0x10000 + 1);

// Awesome idea of merykitty:
2 usages  👤 Roy van Rijn
private int readTemperature() {
    // This is the number part: X.X, -X.X, XX.x or -XX.X
    final long numberBytes = UNSAFE.getLong(ptr);
    final long invNumberBytes = ~numberBytes;

    final int dotPosition = Long.numberOfTrailingZeros(invNumberBytes & DOT_BITS);

    // Calculates the sign
    final long signed = (invNumberBytes << 59) >> 63;
    final int min28 = (dotPosition ^ 0b11100);
    final long minusFilter = ~(signed & 0xFF);
    // Use the pre-calculated decimal position to adjust the values
    final long digits = ((numberBytes & minusFilter) << min28) & 0x0F000F0F00L;

    // Update the pointer here, bit awkward, but we have all the data
    ptr += (dotPosition >> 3) + 3;

    // Multiply by a magic (100 * 0x1000000 + 10 * 0x10000 + 1), to get the result
    final long absValue = ((digits * MAGIC_MULTIPLIER) >>> 32) & 0x3FF;
    // And perform abs()
    return (int) ((absValue + signed) ^ signed); // non-patented method of doing the same trick
}

```

$$-18.3/nTe > eT/n3.81-$$

-	=	101101
.	=	101110
0	=	110000
1	=	110001
2	=	110010
3	=	110011
4	=	110100
5	=	110101
6	=	110110
7	=	110111
8	=	111000
9	=	111001

```

1 usage
private static final long DOT_BITS = 0x10101000;
1 usage
private static final long MAGIC_MULTIPLIER = (100 * 0x1000000 + 10 * 0x10000 + 1);

// Awesome idea of merykitty:
2 usages  Roy van Rijn
private int readTemperature() {
    // This is the number part: X.X, -X.X, XX.x or -XX.X
    final long numberBytes = UNSAFE.getLong(ptr);
    final long invNumberBytes = ~numberBytes;

    final int dotPosition = Long.numberOfTrailingZeros(invNumberBytes & DOT_BITS);

    // calculates the sign
    final long signed = (invNumberBytes << 59) >> 63;
    final int min28 = (dotPosition ^ 0b11100);
    final long minusFilter = ~(signed & 0xFF);
    // Use the pre-calculated decimal position to adjust the values
    final long digits = ((numberBytes & minusFilter) << min28) & 0x0F000F0F00L;

    // Update the pointer here, bit awkward, but we have all the data
    ptr += (dotPosition >> 3) + 3;

    // Multiply by a magic (100 * 0x1000000 + 10 * 0x10000 + 1), to get the result
    final long absValue = ((digits * MAGIC_MULTIPLIER) >>> 32) & 0x3FF;
    // And perform abs()
    return (int) ((absValue + signed) ^ signed); // non-patented method of doing the same trick
}

```

-	=	101101
.	=	101110
0	=	110000
1	=	110001
2	=	110010
3	=	110011
4	=	110100
5	=	110101
6	=	110110
7	=	110111
8	=	111000
9	=	111001

```

1 usage
private static final long DOT_BITS = 0x10101000;
1 usage
private static final long MAGIC_MULTIPLIER = (100 * 0x1000000 + 10 * 0x10000 + 1);

// Awesome idea of merykitty:
2 usages  👤 Roy van Rijn
private int readTemperature() {
    // This is the number part: X.X, -X.X, XX.x or -XX.X
    final long numberBytes = UNSAFE.getLong(ptr);
    final long invNumberBytes = ~numberBytes;

    final int dotPosition = Long.numberOfTrailingZeros(invNumberBytes & DOT_BITS);

    // Calculates the sign
    final long signed = (invNumberBytes << 59) >> 63;
    final int min28 = (dotPosition ^ 0b11100);
    final long minusFilter = ~(signed & 0xFF);
    // Use the pre-calculated decimal position to adjust the values
    final long digits = ((numberBytes & minusFilter) << min28) & 0x0F000F0F00L;

    // Update the pointer here, bit awkward, but we have all the data
    ptr += (dotPosition >> 3) + 3;

    // Multiply by a magic (100 * 0x1000000 + 10 * 0x10000 + 1), to get the result
    final long absValue = ((digits * MAGIC_MULTIPLIER) >>> 32) & 0x3FF;
    // And perform abs()
    return (int) ((absValue + signed) ^ signed); // non-patented method of doing the same trick
}

```

-	=	101101
.	=	101110
0	=	110000
1	=	110001
2	=	110010
3	=	110011
4	=	110100
5	=	110101
6	=	110110
7	=	110111
8	=	111000
9	=	111001

...0X1010

<<59 X1010...00000000

>>63 X 0 or -1 (1111111111111111111111...)

1 usage

```
private static final long DOT_BITS = 0x10101000;
```

1 usage

```
private static final long MAGIC_MULTIPLIER = (100 * 0x1000000 + 10 * 0x10000 + 1);
```

// Awesome idea of [merykitty](#):

2 usages  Roy van Rijn

```
private int readTemperature() {
```

```
    // This is the number part: X.X, -X.X, XX.x or -XX.X
```

```
    final long numberBytes = UNSAFE.getLong(ptr);
```

```
    final long invNumberBytes = ~numberBytes;
```

```
    final int dotPosition = Long.numberOfTrailingZeros(invNumberBytes & DOT_BITS);
```

```
    // Calculates the sign
```

```
    final long signed = (invNumberBytes << 59) >> 63;
```

$$12 \wedge 28 = 16$$

```
    final int min28 = (dotPosition ^ 0b11100);
```

$$20 \wedge 28 = 8$$

```
    final long minusFilter = ~(signed & 0xFF);
```

$$28 \wedge 28 = 0$$

```
    // Use the pre-calculated decimal position to adjust the values
```

```
    final long digits = ((numberBytes & minusFilter) << min28) & 0x0F000F0F00L;
```

```
    // Update the pointer here, bit awkward, but we have all the data
```

```
    ptr += (dotPosition >> 3) + 3;
```

```
    // Multiply by a magic (100 * 0x1000000 + 10 * 0x10000 + 1), to get the result
```

```
    final long absValue = ((digits * MAGIC_MULTIPLIER) >>> 32) & 0x3FF;
```

```
    // And perform abs()
```

```
    return (int) ((absValue + signed) ^ signed); // non-patented method of doing the same trick
```

```
}
```

1 usage

```
private static final long DOT_BITS = 0x10101000;
```

1 usage

```
private static final long MAGIC_MULTIPLIER = (100 * 0x10000000 + 10 * 0x10000 + 1);
```

// Awesome idea of [merykitty](#):

2 usages  Roy van Rijn

```
private int readTemperature() {
```

```
    // This is the number part: X.X, -X.X, XX.x or -XX.X
```

```
    final long numberBytes = UNSAFE.getLong(ptr);
```

```
    final long invNumberBytes = ~numberBytes;
```

```
    final int dotPosition = Long.numberOfTrailingZeros(invNumberBytes & DOT_BITS);
```

```
    // Calculates the sign
```

```
    final long signed = (invNumberBytes << 59) >> 63;
```

```
    final int min28 = (dotPosition ^ 0b11100);
```

$0 \wedge 0xFF = 0$

```
    final long minusFilter = ~(signed & 0xFF);
```

$-1 \wedge 0xFF = 0xFF$ (first byte)

```
    // Use the pre-calculated decimal position to adjust the values
```

```
    final long digits = ((numberBytes & minusFilter) << min28) & 0x0F000F0F00L;
```

```
    // Update the pointer here, bit awkward, but we have all the data
```

```
    ptr += (dotPosition >> 3) + 3;
```

```
    // Multiply by a magic (100 * 0x10000000 + 10 * 0x10000 + 1), to get the result
```

```
    final long absValue = ((digits * MAGIC_MULTIPLIER) >>> 32) & 0x3FF;
```

```
    // And perform abs()
```

```
    return (int) ((absValue + signed) ^ signed); // non-patented method of doing the same trick
```

```
}
```

```

1 usage
private static final long DOT_BITS = 0x10101000;
1 usage
private static final long MAGIC_MULTIPLIER = (100 * 0x1000000 + 10 * 0x10000 + 1);

// Awesome idea of merykitty:
2 usages  👤 Roy van Rijn
private int readTemperature() {
    // This is the number part: X.X, -X.X, XX.x or -XX.X
    final long numberBytes = UNSAFE.getLong(ptr);
    final long invNumberBytes = ~numberBytes;

    final int dotPosition = Long.numberOfTrailingZeros(invNumberBytes & DOT_BITS);

    // Calculates the sign
    final long signed = (invNumberBytes << 59) >> 63;
    final int min28 = (dotPosition ^ 0b11100);
    final long minusFilter = ~(signed & 0xFF);

    // Use the pre-calculated decimal position to adjust the values
    final long digits = ((numberBytes & minusFilter) << min28) & 0x0F000F0F00L;

    // Update the pointer to the next byte, which is the decimal point
    ptr += (dotPosition >> 3) + 3;

    // Multiply by a magic (100 * 0x1000000 + 10 * 0x10000 + 1), to get the result
    final long absValue = ((digits * MAGIC_MULTIPLIER) >>> 32) & 0x3FF;
    // And perform abs()
    return (int) ((absValue + signed) ^ signed); // non-patented method of doing the same trick
}

```

-	=	101101
.	=	101110
0	=	110000
1	=	110001
2	=	110010
3	=	110011
4	=	110100
5	=	110101
6	=	110110
7	=	110111
8	=	111000
9	=	111001

extract just the digits we need, with - and .

1 usage

```
private static final long DOT_BITS = 0x10101000;
```

1 usage

```
private static final long MAGIC_MULTIPLIER = (100 * 0x10000000 + 10 * 0x10000 + 1);
```

```
// Awesome idea of merykitty:
```

2 usages  Roy van Rijn

```
private int readTemperature() {
```

```
    // This is the number part: X.X, -X.X, XX.x or -XX.X
```

```
    final long numberBytes = UNSAFE.getLong(ptr);
```

```
    final long invNumberBytes = ~numberBytes;
```

digits is now in the form:

```
    final int dotPosition = Long.numberOfTrailingZeros(invNumberBytes & DOT_BITS);
```

0xUU00TTHH00

```
    // Calculates the sign
```

where:

```
long signed = (invNumberBytes << 59) >> 63;
```

UU = units

```
min28 = (dotPosition ^ 0b11100);
```

TT = tens

```
minusFilter = ~(signed & 0xFF);
```

HH = hundreds

```
    // Use the calculated decimal position to adjust the values
```

```
    final long digits = ((numberBytes & minusFilter) << min28) & 0x0F000F0F00L;
```

So in the case -18.3, we want -183, so the number in hex is currently:

```
    // Update the pointer here, bit awkward, but we have all the data
```

0x0300080100

```
    ptr += (dotPosition >> 3) + 3;
```

```
    // Multiply by a magic (100 * 0x10000000 + 10 * 0x10000 + 1), to get the result
```

```
    final long absValue = ((digits * MAGIC_MULTIPLIER) >>> 32) & 0x3FF;
```

```
    // And perform abs()
```

```
    return (int) ((absValue + signed) ^ signed); // non-patented method of doing the same trick
```

```
}
```

1 usage

```
private static final long DOT_BITS = 0x10101000;
```

1 usage

```
private static final long MAGIC_MULTIPLIER = (100 * 0x1000000 + 10 * 0x10000 + 1);
```

```
// Awesome idea of merykitty:
```

2 usages  Roy van Rijn

```
private int readTemperature() {
```

```
    // This is the number part: X.X, -X.X, XX.x or -XX.X
```

```
    final long numberBytes = UNSAFE.getLong(ptr);
```

```
    final long invNumberBytes = ~numberBytes;
```

```
    digits = 0x0300080100
```

```
    final int dotPosition = Long.numberOfTrailingZeros(invNumberBytes & DOT_BITS);
```

if we shift this number we can align UU, TT and HH:

```
    // Calculates the sign
```

```
    0x0000000300080100 = digits << (invNumberBytes < 0 ? 32 : 0);
```

```
    0x0003000801000000 = digits << (dotPosition < 32 ? dotPosition : 0);
```

```
    0x0300080100000000 = digits << (dotPosition < 32 ? dotPosition : 0);
```

```
    // Use the pre-calculated decimal position to adjust the values
```

so...

```
final long digits = ((numberBytes & minusFilter) << min28) & 0x0F000F0F00L;
```

digits * 0x1010001L is all of them combined

```
    ptr += (dotPosition >> 3) + 3;
```

```
    // Multiply by a magic (100 * 0x1000000 + 10 * 0x10000 + 1), to get the result
```

```
    final long absValue = ((digits * MAGIC_MULTIPLIER) >>> 32) & 0x3FF;
```

```
    // And perform abs()
```

```
    return (int) ((absValue + signed) ^ signed); // non-patented method of doing the same trick
```

```
}
```

1 usage

```
private static final long DOT_BITS = 0x10101000;
```

1 usage

```
private static final long MAGIC_MULTIPLIER = (100 * 0x1000000 + 10 * 0x10000 + 1);
```

// Awesome idea of [merykitty](#):

2 usages  Roy van Rijn

```
private int readTemperature() {
```

```
    // This is the number part: X.X, -X.X, XX.x or -XX.X
```

```
    final long numberBytes = UNSAFE.getLong(ptr);
```

```
    final long invNumberBytes = ~numberBytes;
```

```
    final int dotPosition = Long.numberOfTrailingZeros(invNumberBytes & DOT_BITS);
```

0x0000000300080100 = digits * 0x1L

0x0003000801000000 = digits * 0x1000L

0x0300080100000000 = digits * 0x100000L

```
    final long minusFilter = ~(signed & 0xFF);
```

we can combine this into a single multiplication!

```
    final long digits = ((numberBytes & minusFilter) << min28) & 0x0F000F0F00L;
```

0x0303080C01080100 = digits * 0x1010001L

```
    // Update the pointer here, bit awkward, but we have all the data
```

```
    ptr = (ptr + dotPosition > 3) + 4;
```

```
    // Multiply by a magic (100 * 0x1000000 + 10 * 0x10000 + 1), to get the result
```

```
    final long absValue = ((digits * MAGIC_MULTIPLIER) >>> 32) & 0x3FF;
```

```
    // And perform abs()
```

```
    return (int) ((absValue + signed) ^ signed); // non-patented method of doing the same trick
```

```
}
```

1 usage

```
private static final long DOT_BITS = 0x10101000;
```

1 usage

```
private static final long MAGIC_MULTIPLIER = (100 * 0x1000000 + 10 * 0x10000 + 1);
```

// Awesome idea of [merykitty](#):

2 usages  Roy van Rijn

```
private int readTemperature() {
```

```
    // This is the number part: X.X, -X.X, XX.x or -XX.X
```

```
    final long numberBytes = UNSAFE.getLong(ptr);
```

```
    final long invNumberBytes = ~numberBytes;
```

```
    final int dotPosition = Long.numberOfTrailingZeros(invNumberBytes & DOT_BITS);
```

what if we multiply the numbers, TT times 10 and HH times 100

```
    // Calculates the sign
```

```
    0x0000000300080100 = digits * 0x1L < 63;
```

```
    0x0001E00500A00000 = digits * 0x1000L * 10
```

```
    0x12C0320640000000 = digits * 0x100000L * 100
```

```
    // Use the pre-calculated decimal position to adjust the values
```

as a single multiplication:

```
    final long digits = (numberBytes & minusFilter) << min28) & 0x0F000F0F00L;
```

```
    0x2C2120B70A080100 = digits * MAGIC_MULTIPLIER
```

```
    // Update the pointer here, less awkward, but we have all the data
```

```
    ptr += 0xB7 = 183 >> 3) + 3;
```

```
    // Multiply by a magic (100 * 0x1000000 + 10 * 0x10000 + 1), to get the result
```

```
    final long absValue = ((digits * MAGIC_MULTIPLIER) >>> 32) & 0x3FF;
```

```
    // And perform abs()
```

```
    return (int) ((absValue + signed) ^ signed); // non-patented method of doing the same trick
```

```
}
```

1 usage

```
private static final long DOT_BITS = 0x10101000;
```

1 usage

```
private static final long MAGIC_MULTIPLIER = (100 * 0x1000000 + 10 * 0x10000 + 1);
```

// Awesome idea of [merykitty](#):

2 usages  Roy van Rijn

```
private int readTemperature() {
```

```
    // This is the number part: X.X, -X.X, XX.x or -XX.X
```

```
    final long numberBytes = UNSAFE.getLong(ptr);
```

```
    final long invNumberBytes = ~numberBytes;
```

```
    final int dotPosition = Long.numberOfTrailingZeros(invNumberBytes & DOT_BITS);
```

```
    // Calculates the sign
```

```
    final long signed = (invNumberBytes << 59) >> 63;
```

```
    final int min28 = (dotPosition ^ 0b11100);
```

```
    final long minusFilter = ~(signed & 0xFF);
```

```
    // Use the pre-calculated decimal position to adjust the values
```

```
    final long digits = ((numberBytes & minusFilter) << min28) & 0x0F000F0F00L;
```

```
    // Update the pointer here, bit awkward, but we have all the data
```

```
    ptr += (dotPosition >> 3) + 3;
```

```
    // Multiply by a magic (100 * 0x1000000 + 10 * 0x10000 + 1), to get the result
```

```
    final long absValue = ((digits * MAGIC_MULTIPLIER) >>> 32) & 0x3FF;
```

```
    // And perform abs()
```

```
    return (int) ((absValue + signed) ^ signed); // non-patented method of doing the same trick
```

```
}
```

Keeping track...

We now read the file (MMAP) and:

- Find and read city names (SWAR)
- Parse temperatures (branchless)
- Next we need to **store** them.

HashMap

max size: 8

0



1



2



3



4



5



6



7



HashMap

max size: 8

“London”
-183

0

1

2

3

4

5

6

7

HashMap

max size: 8

“London”
-183

HASHCODE:
328756875

0

1

2

3

4

5

6

7

HashMap

max size: 8

“London”
-183

HASHCODE:

328756875

$\%8 = 3$

0

1

2

3

4

5

6

7

HashMap

max size: 8

“London”
-183

HASHCODE:

328756875

$\%8 = 3$

0

1

2

3

4

5

6

7

“London”

-183

HashMap

max size: 8

“London”
201

HASHCODE:

328756875

$\%8 = 3$

0

1

2

3

4

5

6

7

“London”

HashMap

max size: 8

“London”
201

HASHCODE:

328756875

$\%8 = 3$

0

1

2

3

4

5

6

7

“London”

-183, 201

HashMap

max size: 8

“Rotterdam”
167

HASHCODE:

773734787

$\%8 = 3$

0

1

2

3

4

5

6

7

“London”

HashMap

max size: 8

“Rotterdam”
167

HASHCODE:

773734787

$\%8 = 3$

0

1

2

3

4

5

6

7

“London”



HashMap

max size: 8

0

1

2

3

4

5

6

7

“London”

“Rotterdam”



HashMap

0

1

2

3

4

5

6

7

“London”

“Rotterdam”

forward probing



Map implementation

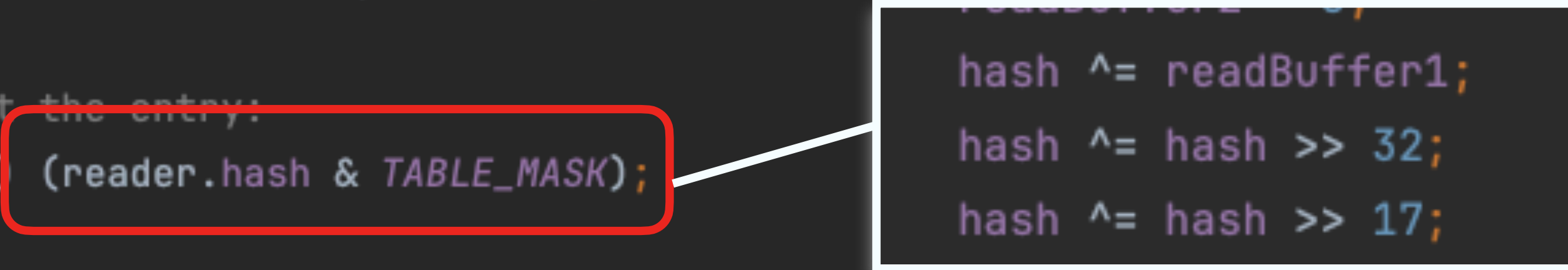
```
int temperature = reader.readTemperature();

// Find or insert the entry:
int index = (int) (reader.hash & TABLE_MASK);
while (true) {
    entry = table[index];
    if (entry == null) {
        int length = reader.entryLength;
        byte[] entryBytes = (length < PREMADE_MAX_SIZE && entryCount < PREMADE_ENTRIES) ?
            preConstructedEntries[entryCount++]
            : new byte[ENTRY_BASESIZE_WHITESPACE + length]; // with enough room
        table[index] = fillEntry(entryBytes, reader.entryStart, length, temperature, reader.readBuffer1, reader.readBuffer2);
        break;
    }
    else if (reader.matches(entry)) {
        updateEntry(entry, temperature);
        break;
    }
    else {
        // Move to the next index
        index = (index + 1) & TABLE_MASK;
    }
}
```

Map implementation

```
int temperature = reader.readTemperature();

// Find or insert the entry:
int index = (int) (reader.hash & TABLE_MASK);
while (true) {
    entry = table[index];
    if (entry == null) {
        int length = reader.entryLength;
        byte[] entryBytes = (length < PREMADE_MAX_SIZE && entryCount < PREMADE_ENTRIES) ?
            preConstructedEntries[entryCount++]
            : new byte[ENTRY_BASESIZE_WHITESPACE + length]; // with enough room
        table[index] = fillEntry(entryBytes, reader.entryStart, length, temperature, reader.readBuffer1, reader.readBuffer2);
        break;
    }
    else if (reader.matches(entry)) {
        updateEntry(entry, temperature);
        break;
    }
    else {
        // Move to the next index
        index = (index + 1) & TABLE_MASK;
    }
}
}
```



```
hash ^= readBuffer1;
hash ^= hash >> 32;
hash ^= hash >> 17;
```

Map implementation

```
int temperature = reader.readTemperature();

// Find or insert the entry:
int index = (int) (reader.hash & TABLE_MASK);
while (true) {
    entry = table[index];
    if (entry == null) {
        int length = reader.entryLength;
        byte[] entryBytes = (length < PREMADE_MAX_SIZE && entryCount < PREMADE_ENTRIES) ?
            preConstructedEntries[entryCount++]
            : new byte[ENTRY_BASESIZE_WHITESPACE + length]; // with enough room
        table[index] = fillEntry(entryBytes, reader.entryStart, length, temperature, reader.readBuffer1, reader.readBuffer2);
        break;
    }
    else if (reader.matches(entry)) {
        updateEntry(entry, temperature);
        break;
    }
    else {
        // Move to the next index
        index = (index + 1) & TABLE_MASK;
    }
}
}
```

Map implementation

```
int temperature = reader.readTemperature();

// Find or insert the entry:
int index = (int) (reader.hash & TABLE_MASK);
while (true) {
    entry = table[index];
    if (entry == null) {
        int length = reader.entryLength;
        byte[] entryBytes = (length < PREMADE_MAX_SIZE && entryCount < PREMADE_ENTRIES) ?
            preConstructedEntries[entryCount++]
            : new byte[ENTRY_BASESIZE_WHITESPACE + length]; // with enough room
        table[index] = fillEntry(entryBytes, reader.entryStart, length, temperature, reader.readBuffer1, reader.readBuffer2);
        break;
    }
    else if (reader.matches(entry)) {
        updateEntry(entry, temperature);
        break;
    }
    else {
        // Move to the next index
        index = (index + 1) & TABLE_MASK;
    }
}
}
```

Map implementation

```
int temperature = reader.readTemperature();

// Find or insert the entry:
int index = (int) (reader.hash & TABLE_MASK);
while (true) {
    entry = table[index];
    if (entry == null) {
        int length = reader.entryLength;
        byte[] entryBytes = (length < PREMADE_MAX_SIZE && entryCount < PREMADE_ENTRIES) ?
            preConstructedEntries[entryCount++]
            : new byte[ENTRY_BASESIZE_WHITESPACE + length]; // with enough room
        table[index] = fillEntry(entryBytes, reader.entryStart, length, temperature, reader.readBuffer1, reader.readBuffer2);
        break;
    }
    else if (reader.match(entry)) {
        updateEntry(entry, reader.readBuffer1, reader.readBuffer2);
        break;
    }
    else {
        // Move to the next index
        index = (index + 1) & TABLE_MASK;
    }
}

}
```

1 usage Roy van Rijn

```
private static byte[] fillEntry(final byte[] entry, final long fromAddress, final int entryLength) {
    UNSAFE.putLong(entry, ENTRY_SUM, temp);
    UNSAFE.putInt(entry, ENTRY_MIN, temp);
    UNSAFE.putInt(entry, ENTRY_MAX, temp);
    UNSAFE.putInt(entry, ENTRY_COUNT, x: 1);
    UNSAFE.putByte(entry, ENTRY_LENGTH, (byte) entryLength);
    UNSAFE.copyMemory(srcBase: null, fromAddress, entry, ENTRY_NAME, bytes: entryLength - 16);
    UNSAFE.putLong(entry, offset: ENTRY_NAME + entryLength - 16, readBuffer1);
}
```

flyweight-pattern

Map implementation

```
int temperature = reader.readTemperature();

// Find or insert the entry:
int index = (int) (reader.hash & TABLE_MASK);
while (true) {
    entry = table[index];
    if (entry == null) {
        int length = reader.entryLength;
        byte[] entryBytes = (length < PREMADE_MAX_SIZE && entryCount < PREMADE_ENTRIES) ?
            preConstructedEntries[entryCount++]
            : new byte[ENTRY_BASESIZE_WHITESPACE + length]; // with enough room
        table[index] = fillEntry(entryBytes, reader.entryStart, length, temperature, reader.readBuffer1, reader.readBuffer2);
        break;
    }
    else if (reader.matches(entry)) {
        updateEntry(entry, temperature);
        break;
    }
    else {
        // Move to the next index
        index = (index + 1) & TABLE_MASK;
    }
}
}
```

Map implementation

```
int temperature = reader.readTemperature();

// Find or insert the entry:
int index = (int) (reader.hash & TABLE_MASK);
while (true) {
    entry = table[index];
    if (entry == null) {
        int length = reader.entryLength;
        byte[] entryBytes = (length < PREMADE
            preConstructedEntries[entryCount]
            : new byte[ENTRY_BASESIZE_WHEN_NEEDED]);
        table[index] = fillEntry(entryBytes, length);
        break;
    }
    else if (reader.matches(entry)) {
        updateEntry(entry, temperature);
        break;
    }
    else {
        // Move to the next index
        index = (index + 1) & TABLE_MASK;
    }
}
}
```

```
public static void updateEntry(final byte[] entry, final int temp) {

    int entryMin = UNSAFE.getInt(entry, ENTRY_MIN);
    int entryMax = UNSAFE.getInt(entry, ENTRY_MAX);
    long entrySum = UNSAFE.getLong(entry, ENTRY_SUM) + temp;
    int entryCount = UNSAFE.getInt(entry, ENTRY_COUNT) + 1;

    if (temp < entryMin) {
        UNSAFE.putInt(entry, ENTRY_MIN, temp);
    }
    else if (temp > entryMax) {
        UNSAFE.putInt(entry, ENTRY_MAX, temp);
    }
    UNSAFE.putInt(entry, ENTRY_COUNT, entryCount);
    UNSAFE.putLong(entry, ENTRY_SUM, entrySum);
}
```

Map implementation

```
int temperature = reader.readTemperature();

// Find or insert the entry:
int index = (int) (reader.hash & TABLE_MASK);
while (true) {
    entry = table[index];
    if (entry == null) {
        int length = reader.entryLength;
        byte[] entryBytes = (length < PREMADE_MAX_SIZE && entryCount < PREMADE_ENTRIES) ?
            preConstructedEntries[entryCount++]
            : new byte[ENTRY_BASESIZE_WHITESPACE + length]; // with enough room
        table[index] = fillEntry(entryBytes, reader.entryStart, length, temperature, reader.readBuffer1, reader.readBuffer2);
        break;
    }
    else if (reader.matches(entry)) {
        updateEntry(entry, temperature);
        break;
    }
    else {
        // Move to the next index
        index = (index + 1) & TABLE_MASK;
    }
}
```



Thomas Wuerthinger

@thomaswue

OK, I need to stop now, this is getting ridiculous 🤡. Kernel unmapping already the primary bottleneck... [#1BRC](#) [@gunnarmorling](#)

```
thomas@...  
eb=-39.6/10.7/61.7, Zanzibar City=-23.4/  
26.0/76.7, Zürich=-37.9/9.3/58.7, Ürümqi  
=-49.2/7.4/54.7, İzmir=-36.1/17.9/71.6}  
  
real    0m0.621s  
user    0m13.935s  
sys     0m0.444s
```



Thomas Wuerthinger

@thomaswue

...

Turns out the easiest way to work around the unmapping issue is to spawn a process fork that does the work and let the OS clean up the fork later. Now I only need to convince [@gunnarmorling](#) that this is legit 😊
[#1brc](#). It is just a few lines of beautiful Java Process API code 🙌.

```
ma... 🔍 ☰ —
0.0, Zanzibar City=-
ich=-39.4/9.3/58.0,
8.0, Izmir=-33.6/17
0.484s
0.005s
0.002s
to:~/dev/1brc$
```

```
nWorker() throws IOException {
    info = ProcessHandle.current().info();
    workerCommand = new ArrayList<>();
    send(workerCommand::add);
    present(args -> workerCommand.addAl
    "-worker");
    (workerCommand) ProcessBuilder
    put(ProcessBuilder.Redirect.PIPE)
    cess
    eam() InputStream
    System.out);
```

// Idea of [thomaswue](#), don't wait for slow unmap:

1 usage  Roy van Rijn

```
private static void spawnWorker() throws IOException {
    ProcessHandle.Info info = ProcessHandle.current().info();
    ArrayList<String> workerCommand = new ArrayList<>();
    info.command().ifPresent(workerCommand::add);
    info.arguments().ifPresent(args -> workerCommand.addAll(Arrays.asList(args)));
    workerCommand.add("--worker");
    new ProcessBuilder().command(workerCommand).inheritIO().redirectOutput(ProcessBuilder.Redirect.PIPE)
        .start().getInputStream().transferTo(System.out);
}
```

 Roy van Rijn

```
public static void main(String[] args) throws Exception {

    if (args.length == 0 || !("--worker".equals(args[0]))) {
        spawnWorker();
        return;
    }
}
```

MAIN

// Idea of [thomaswue](#), don't wait for slow unmap:

1 usage  Roy van Rijn

```
private static void spawnWorker() throws IOException {  
    ProcessHandle.Info info = ProcessHandle.current().info();  
    ArrayList<String> workerCommand = new ArrayList<>();  
    info.command().ifPresent(workerCommand::add);  
    info.arguments().ifPresent(args -> workerCommand.addAll(Arrays.asList(args)));  
    workerCommand.add("--worker");  
    new ProcessBuilder().command(workerCommand).inheritIO().redirectOutput(ProcessBuilder.Redirect.PIPE)  
        .start().getInputStream().transferTo(System.out);  
}
```

 Roy van Rijn

```
public static void main(String[] args) throws Exception {  
  
    if (args.length == 0 || !("--worker".equals(args[0]))) {  
        spawnWorker();  
        return;  
    }  
}
```

MAIN

// Idea of [thomaswue](#), don't wait for slow unmap:

1 usage  Roy van Rijn

```
private static void spawnWorker() throws IOException {  
    ProcessHandle.Info info = ProcessHandle.current().info();  
    ArrayList<String> workerCommand = new ArrayList<>();  
    info.command().ifPresent(workerCommand::add);  
    info.arguments().ifPresent(args -> workerCommand.addAll(Arrays.asList(args)));  
    workerCommand.add("--worker");  
    new ProcessBuilder().command(workerCommand).inheritIO().redirectOutput(ProcessBuilder.Redirect.PIPE)  
        .start().getInputStream().transferTo(System.out);  
}
```

 Roy van Rijn

```
public static void main(String[] args) throws Exception {  
  
    if (args.length == 0 || !("--worker".equals(args[0]))) {  
        spawnWorker();  
        return;  
    }  
}
```

MAIN

--WORKER

```
// Idea of thomaswue, don't wait for slow unmap:
```

```
1 usage  👤 Roy van Rijn
```

```
private static void spawnWorker() throws IOException {  
    ProcessHandle.Info info = ProcessHandle.current().info();  
    ArrayList<String> workerCommand = new ArrayList<>();  
    info.command().ifPresent(workerCommand::add);  
    info.arguments().ifPresent(args -> workerCommand.addAll(Arrays.asList(args)));  
    workerCommand.add("--worker");  
    new ProcessBuilder().command(workerCommand).inheritIO().redirectOutput(ProcessBuilder.Redirect.PIPE)  
        .start().getInputStream().transferTo(System.out);  
}
```

```
👤 Roy van Rijn
```

```
public static void main(String[] args) throws Exception {  
  
    if (args.length == 0 || !("--worker".equals(args[0]))) {  
        spawnWorker();  
        return;  
    }  
}
```

MAIN

--WORKER

```
// Idea of thomaswue, don't wait for slow unmap:
```

```
1 usage  👤 Roy van Rijn
```

```
private static void spawnWorker() throws IOException {
    ProcessHandle.Info info = ProcessHandle.current().info();
    ArrayList<String> workerCommand = new ArrayList<>();
    info.command().ifPresent(workerCommand::add);
    info.arguments().ifPresent(args -> workerCommand.addAll(Arrays.asList(args)));
    workerCommand.add("--worker");
    new ProcessBuilder().command(workerCommand).inheritIO().redirectOutput(ProcessBuilder.Redirect.PIPE)
        .start().getInputStream().transferTo(System.out);
}
```

```
👤 Roy van Rijn
```

```
public static void main(String[] args) throws Exception {

    if (args.length == 0 || !("--worker".equals(args[0]))) {
        spawnWorker();
        return;
    }
}
```



MAIN



--WORKER

```
// Idea of thomaswue, don't wait for slow unmap:
```

```
1 usage  👤 Roy van Rijn
```

```
private static void spawnWorker() throws IOException {  
    ProcessHandle.Info info = ProcessHandle.current().info();  
    ArrayList<String> workerCommand = new ArrayList<>();  
    info.command().ifPresent(workerCommand::add);  
    info.arguments().ifPresent(args -> workerCommand.addAll(Arrays.asList(args)));  
    workerCommand.add("--worker");  
    new ProcessBuilder().command(workerCommand).inheritIO().redirectOutput(ProcessBuilder.Redirect.PIPE)  
        .start().getInputStream().transferTo(System.out);  
}
```

```
👤 Roy van Rijn
```

```
public static void main(String[] args) throws Exception {  
  
    if (args.length == 0 || !("--worker".equals(args[0]))) {  
        spawnWorker();  
        return;  
    }  
}
```



MAIN



--WORKER

UNMAPPING...

```
// Idea of thomaswue, don't wait for slow unmap:
```

```
1 usage  👤 Roy van Rijn
```

```
private static void spawnWorker() throws IOException {
    ProcessHandle.Info info = ProcessHandle.current().info();
    ArrayList<String> workerCommand = new ArrayList<>();
    info.command().ifPresent(workerCommand::add);
    info.arguments().ifPresent(args -> workerCommand.addAll(Arrays.asList(args)));
    workerCommand.add("--worker");
    new ProcessBuilder().command(workerCommand).inheritIO().redirectOutput(ProcessBuilder.Redirect.PIPE)
        .start().getInputStream().transferTo(System.out);
}
```

```
👤 Roy van Rijn
```

```
public static void main(String[] args) throws Exception {

    if (args.length == 0 || !("--worker".equals(args[0]))) {
        spawnWorker();
        return;
    }
}
```

Which JVM?

Tried different versions/vendors/suppliers

Native compilation (Graal) is fastest

Graal (native-image)

```
source "$HOME/.sdkman/bin/sdkman-init.sh"
sdk use java 21.0.2-graal 1>&2

# ./mvnw clean verify removes target/ and will re-trigger native image creation.
if [ ! -f target/CalculateAverage_royvanrijn_image ]; then

    JAVA_OPTS="--enable-preview"

    NATIVE_IMAGE_OPTS="-H:+UnlockExperimentalVMOptions --initialize-at-build-time=dev.morling.onebrc.CalculateAverage_royvanrijn"
    NATIVE_IMAGE_OPTS="${NATIVE_IMAGE_OPTS} --gc=epsilon -O3 -march=native -R:MaxHeapSize=515m -H:-GenLoopSafepoints"
    NATIVE_IMAGE_OPTS="${NATIVE_IMAGE_OPTS} -H:InlineAllBonus=10 -H:-ParseRuntimeOptions ${JAVA_OPTS}"

    native-image $NATIVE_IMAGE_OPTS\
        -cp target/average-1.0.0-SNAPSHOT.jar\
        -o target/CalculateAverage_royvanrijn_image\
        dev.morling.onebrc.CalculateAverage_royvanrijn
fi
```

Graal (native-image)

```
source "$HOME/.sdkman/bin/sdkman-init.sh"
sdk use java 21.0.2-graal 1>&2

# ./mvnw clean verify removes target/ and will re-trigger native image creation.
if [ ! -f target/CalculateAverage_royvanrijn_image ]; then

    JAVA_OPTS="--enable-preview"

    NATIVE_IMAGE_OPTS="-H:+UnlockExperimentalVMOptions --initialize-at-build-time=dev.morling.onebrc.CalculateAverage_royvanrijn"
    NATIVE_IMAGE_OPTS="${NATIVE_IMAGE_OPTS} --gc=epsilon -O3 -march=native -R:MaxHeapSize=515m -H:-GenLoopSafepoints"
    NATIVE_IMAGE_OPTS="${NATIVE_IMAGE_OPTS} -H:InlineAllBonus=10 -H:-ParseRuntimeOptions ${JAVA_OPTS}"

    native-image $NATIVE_IMAGE_OPTS\
        -cp target/average-1.0.0-SNAPSHOT.jar\
        -o target/CalculateAverage_royvanrijn_image\
        dev.morling.onebrc.CalculateAverage_royvanrijn
fi
```

Graal (native-image)

```
source "$HOME/.sdkman/bin/sdkman-init.sh"
sdk use java 21.0.2-graal 1>&2

# ./mvnw clean verify removes target/ and will re-trigger native image creation.
if [ ! -f target/CalculateAverage_royvanrijn_image ]; then

    JAVA_OPTS="--enable-preview"

    NATIVE_IMAGE_OPTS="-H:+UnlockExperimentalVMOptions --initialize-at-build-time=dev.morling.onebrc.CalculateAverage_royvanrijn"
    NATIVE_IMAGE_OPTS="${NATIVE_IMAGE_OPTS} --gc=epsilon -O3 -march=native -R:MaxHeapSize=515m -H:-GenLoopSafepoints"
    NATIVE_IMAGE_OPTS="${NATIVE_IMAGE_OPTS} -H:InlineAllBonus=10 -H:-ParseRuntimeOptions ${JAVA_OPTS}"

    native-image $NATIVE_IMAGE_OPTS\
        -cp target/average-1.0.0-SNAPSHOT.jar\
        -o target/CalculateAverage_royvanrijn_image\
        dev.morling.onebrc.CalculateAverage_royvanrijn
fi
```

Graal (na

```
source "$HOME/.sdkman/bin
```

```
sdk use java 21.0.2-graal
```

```
# ./mvnw clean verify rem
```

```
if [ ! -f target/Calculat
```

```
JAVA_OPTS="--enable-p
```

```
NATIVE_IMAGE_OPTS="-H
```

```
NATIVE_IMAGE_OPTS="$@
```

```
NATIVE_IMAGE_OPTS="$@
```

```
native-image $NATIVE
```

```
-cp target/average-
```

```
-o target/Calculate
```

```
dev.morling.onebroc.
```

```
fi
```

```
Using java version 21.0.2-graal in this shell.
=====
GraalVM Native Image: Generating 'CalculateAverage_royvanrijn_image' (executable)...
=====
[1/8] Initializing... (8,6s @ 0,15GB)
Java version: 21.0.2+13-LTS, vendor version: Oracle GraalVM 21.0.2+13.1
Graal compiler: optimization level: 3, target machine: native, PGO: off
C compiler: cc (apple, arm64, 15.0.0)
Garbage collector: Epsilon GC (max heap size: 515,00MB)
1 user-specific feature(s):
- com.oracle.svm.thirdparty.gson.GsonFeature
-----
1 experimental option(s) unlocked:
- '-H:-ParseRuntimeOptions' (origin(s): command line)
-----
Build resources:
- 24,18GB of memory (75,6% of 32,00GB system memory, determined at start)
- 12 thread(s) (100,0% of 12 available processor(s), determined at start)
[2/8] Performing analysis... [*****] (5,0s @ 0,28GB)
3,299 reachable types (72,1% of about 4,578 (total))
3,888 reachable fields (49,8% of about 7,805 (total))
16,870 reachable methods (46,1% of about 36,555 (total))
1,063 types, 1, 112 fields, and 687 methods registered for reflection
58 types, 1, 63 fields, and 52 methods registered for JNI access
4 native libraries: -framework Foundation, dl, pthread, z
[3/8] Building universe... (0,8s @ 0,33GB)
[4/8] Parsing methods... [*] (0,8s @ 0,30GB)
[5/8] Inlining methods... [****] (0,3s @ 0,31GB)
[6/8] Compiling methods... [*****] (16,5s @ 0,44GB)
[7/8] Layouting methods... [*] (1,3s @ 0,49GB)
[8/8] Creating image... [*] (1,5s @ 0,40GB)
8,82MB (54,30%) for code area: 7,934 compilation units
6,98MB (43,01%) for image heap: 105,041 objects and 53 resources
448,19kB ( 2,70%) for other data
16,24MB in total
-----
Generating file for employee rotterdam/Patrick_Salij with company details
-----
Top 10 origins of code area:
6,82MB java.base
1,39MB svm.jar (Native Image)
223,50kB java.logging
153,40kB com.oracle.svm.svm_enterprise
42,68kB jdk.proxy3
42,45kB jdk.proxy1
33,57kB average-1.0.0-SNAPSHOT.jar
32,89kB jdk.internal.vm.ci
18,38kB jdk.internal.vm.compiler
17,98kB org.graalvm.collections
16,60kB for 3 more packages
Top 10 object types in image heap:
2,23MB byte[] for code metadata
1,29MB byte[] for java.lang.String
768,96kB java.lang.String
561,09kB java.lang.Class
282,76kB byte[] for general heap data
229,66kB java.util.HashMap$Node
170,84kB byte[] for reflection metadata
154,64kB com.oracle.svm.core.hub.DynamicHubCompanion
154,34kB byte[] for embedded resources
129,86kB char[]
1,07MB for 860 more object types
-----
Use '-H:+BuildReport' to create a report with more details.
-----
Security report:
```

31 Jan 2024

Results

These are the results from running all entries into the challenge on eight cores of a [Hetzner AX161](#) dedicated server (32 core AMD EPYC™ 7502P (Zen2), 128 GB RAM).

#	Result (m:s.ms)	Implementation	JDK	Submitter	Notes	Certificates
1	00:01.535	link	21.0.2-graal	Thomas Wuerthinger , Quan Anh Mai , Alfonso² Peterssen	GraalVM native binary, uses Unsafe	Certificate
2	00:01.587	link	21.0.2-graal	Artsiom Korzun	GraalVM native binary, uses Unsafe	Certificate
3	00:01.608	link	21.0.2-graal	Jaromir Hamala	GraalVM native binary, uses Unsafe	Certificate
	00:01.880	link	21.0.1-open	Serkan ÖZAL	uses Unsafe	Certificate
	00:01.921	link	21.0.2-graal	Van Phu DO	GraalVM native binary, uses Unsafe	Certificate
	00:02.018	link	21.0.2-graal	Stephen Von Worley	GraalVM native binary, uses Unsafe	Certificate
	00:02.157	link	21.0.2-graal	Roy van Rijn	GraalVM native binary, uses Unsafe	Certificate

Runtime:

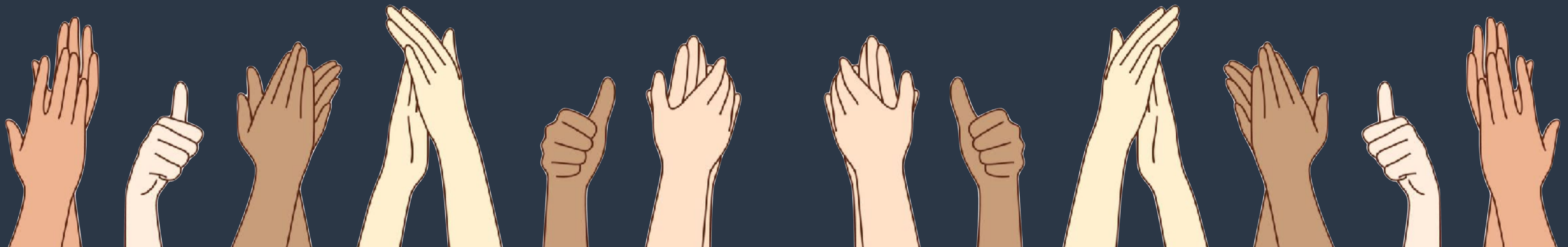
~~04:49.679~~

00:02.157





Gunnar Morling
@gunnarmorling





Questions?