

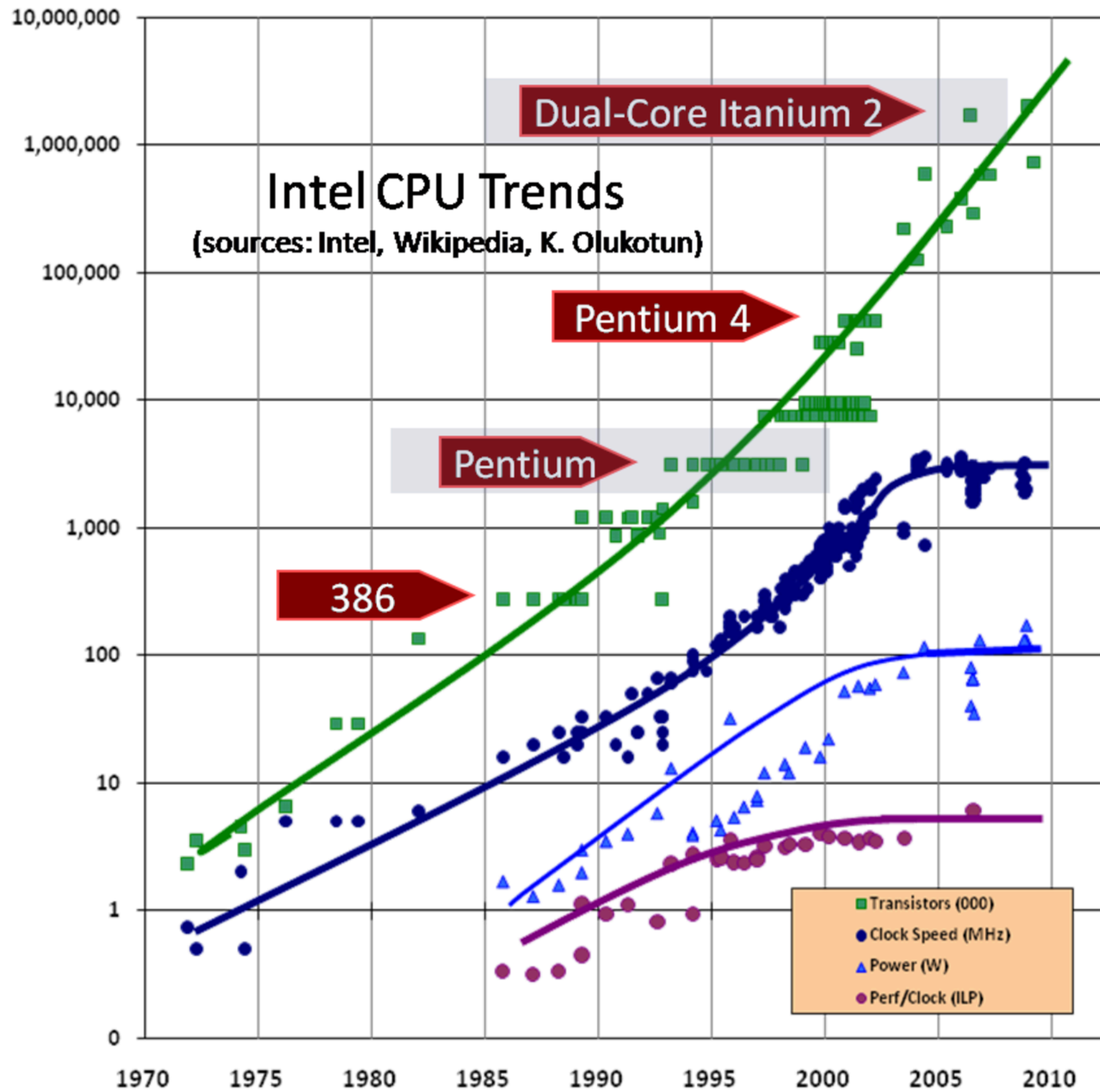
Concurrent programming

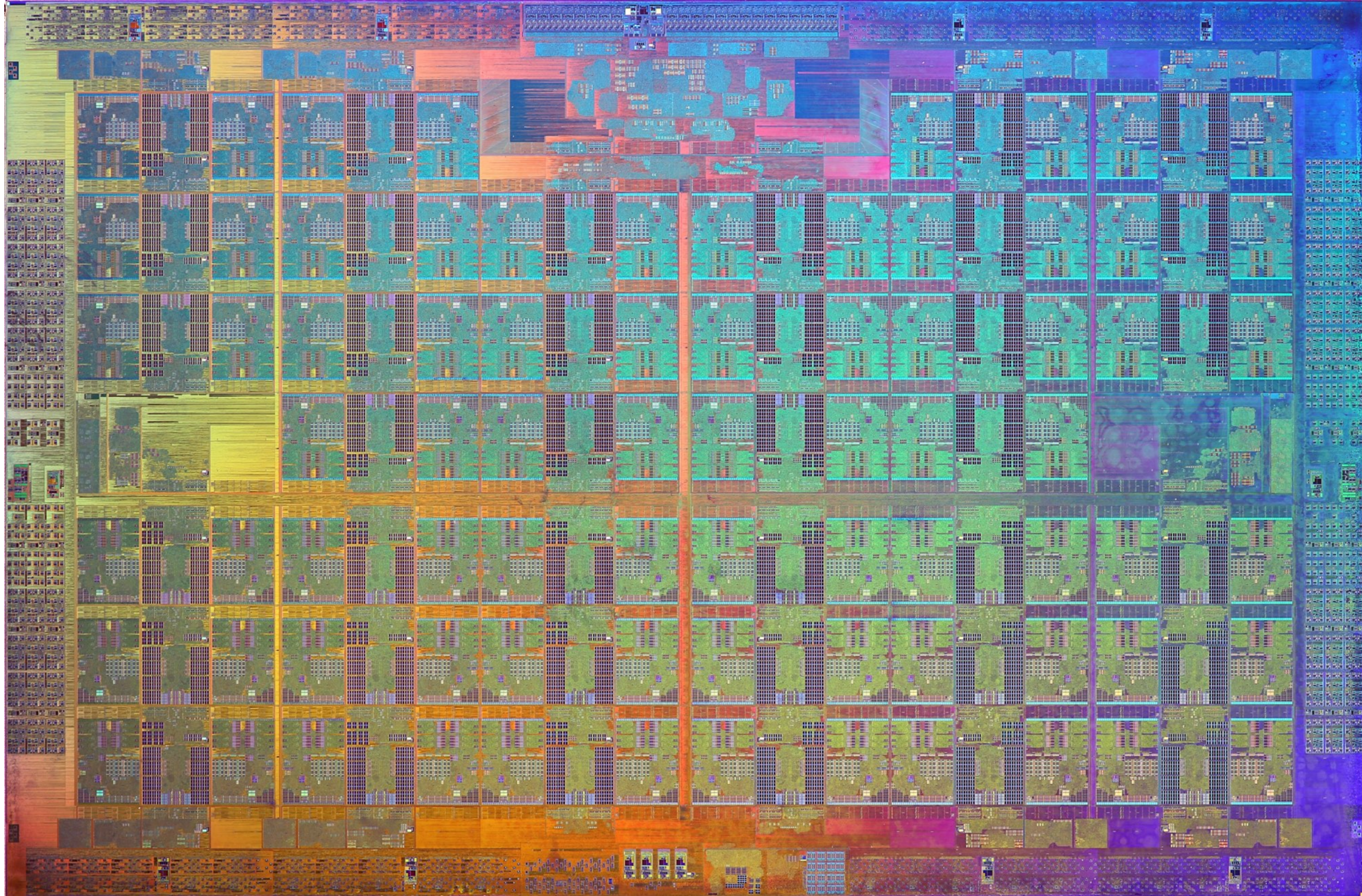
Niklas Gustavsson

ngn@spotify.com

@protocol7







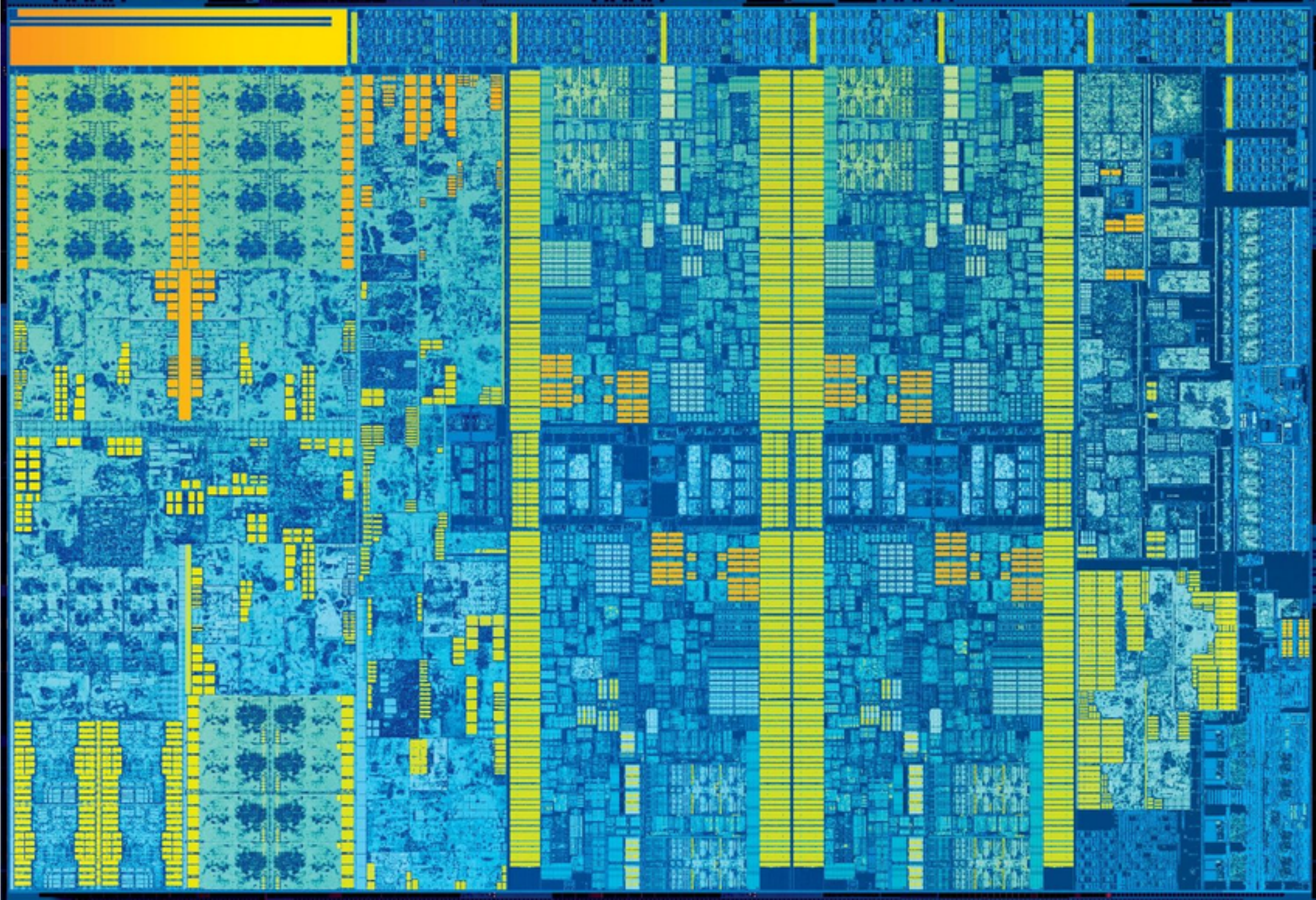


天河

天河

天河二号

天河二号



Cache latency

- L1: 1ns
- L2: 3ns
- L3: 15-20ns
- RAM: 65ns

Counting things

```
public class NaiveCounter {  
    private long count = 0;  
  
    public void increment() {  
        count++;  
    }  
  
    public long count() {  
        return count;  
    }  
}
```


volatile

```
public class VolatileCounter {  
    private volatile long count = 0;  
  
    public void increment() {  
        count++;  
    }  
  
    public long count() {  
        return count;  
    }  
}
```

synchronized

```
public class SynchronizedCounter {  
    private long count = 0;  
  
    public synchronized void increment() {  
        count++;  
    }  
  
    public synchronized long count() {  
        return count;  
    }  
}
```

ReadWriteLock

CAS

```
java.util.concurrent.atomic
```

AtomicLong


```
public class AtomicCounter {  
    private final AtomicLong count = new AtomicLong();  
  
    public void increment() {  
        count.incrementAndGet();  
    }  
  
    public long count() {  
        return count.longValue();  
    }  
}
```

```
public final long incrementAndGet() {  
    for (;;) {  
        long current = get();  
        long next = current + 1;  
        if (compareAndSet(current, next))  
            return next;  
    }  
}
```

LongAdder

```
public void add(long x) {
    Cell[] as; long b, v; int m; Cell a;
    if ((as = cells) != null || !casBase(b = base, b + x)) {
        boolean uncontended = true;
        if (as == null || (m = as.length - 1) < 0 ||
            (a = as[getProbe() & m]) == null ||
            !(uncontended = a.cas(v = a.value, v + x)))
            longAccumulate(x, null, uncontended);
    }
}
```

```
public class LongAdderCounter {  
    private final LongAdder count = new LongAdder();  
  
    public void increment() {  
        count.increment();  
    }  
  
    public long count() {  
        return count.sum();  
    }  
}
```

Cache lines

Aside: Lists

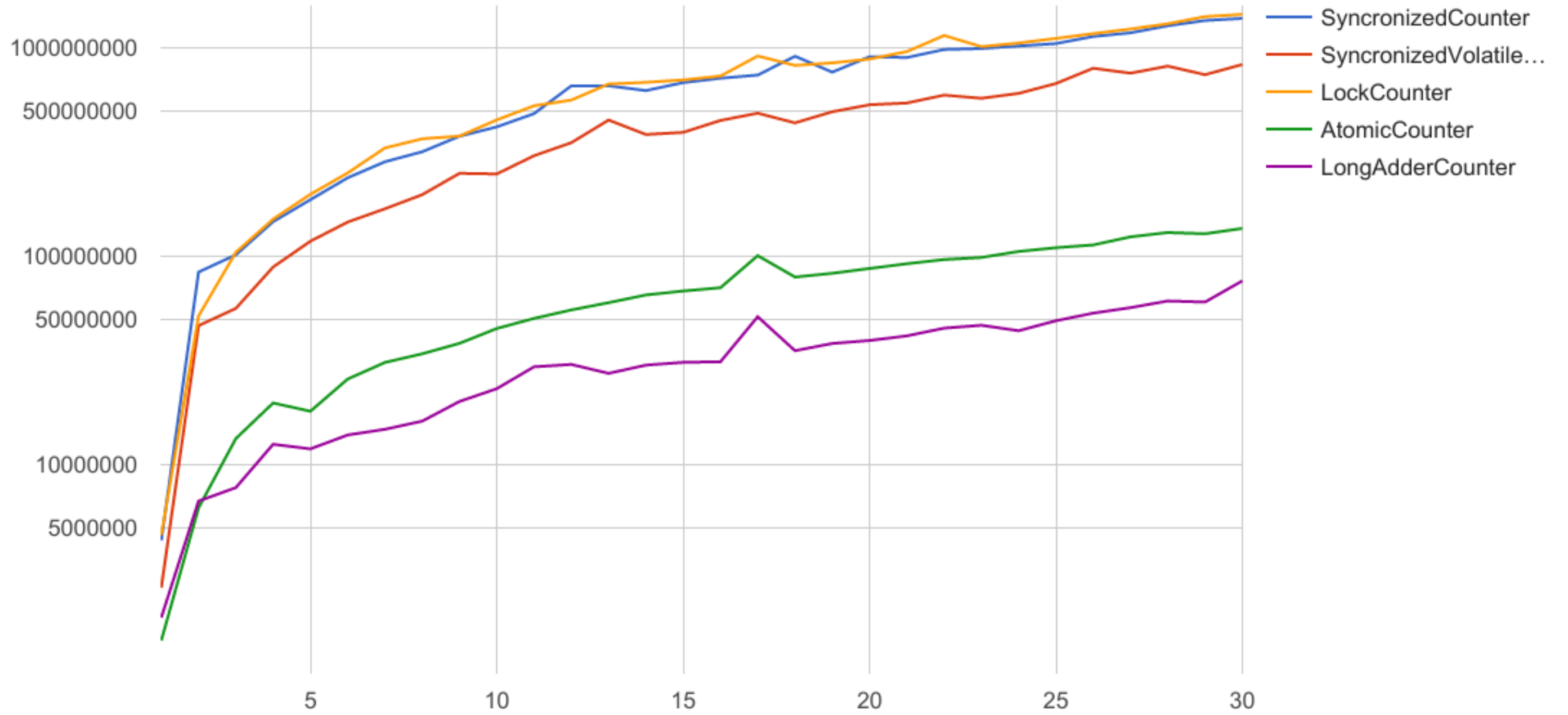
JEP 169: Value Objects

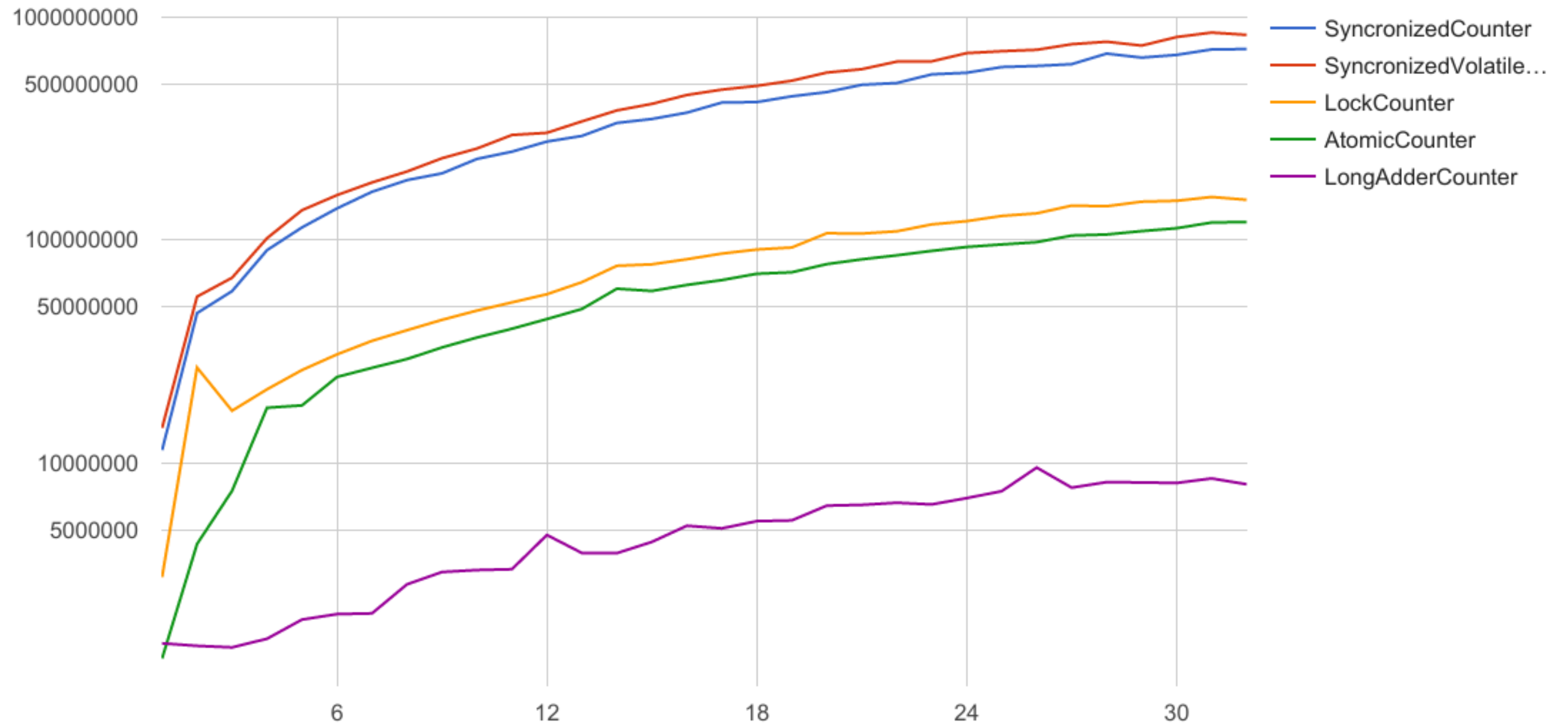
False sharing

Padding

```
volatile long p0, p1, p2, p3, p4, p5, p6;  
volatile long value;  
volatile long q0, q1, q2, q3, q4, q5, q6;
```

```
@sun.misc.Contended static final class Cell {  
    volatile long value;  
    ...  
}
```

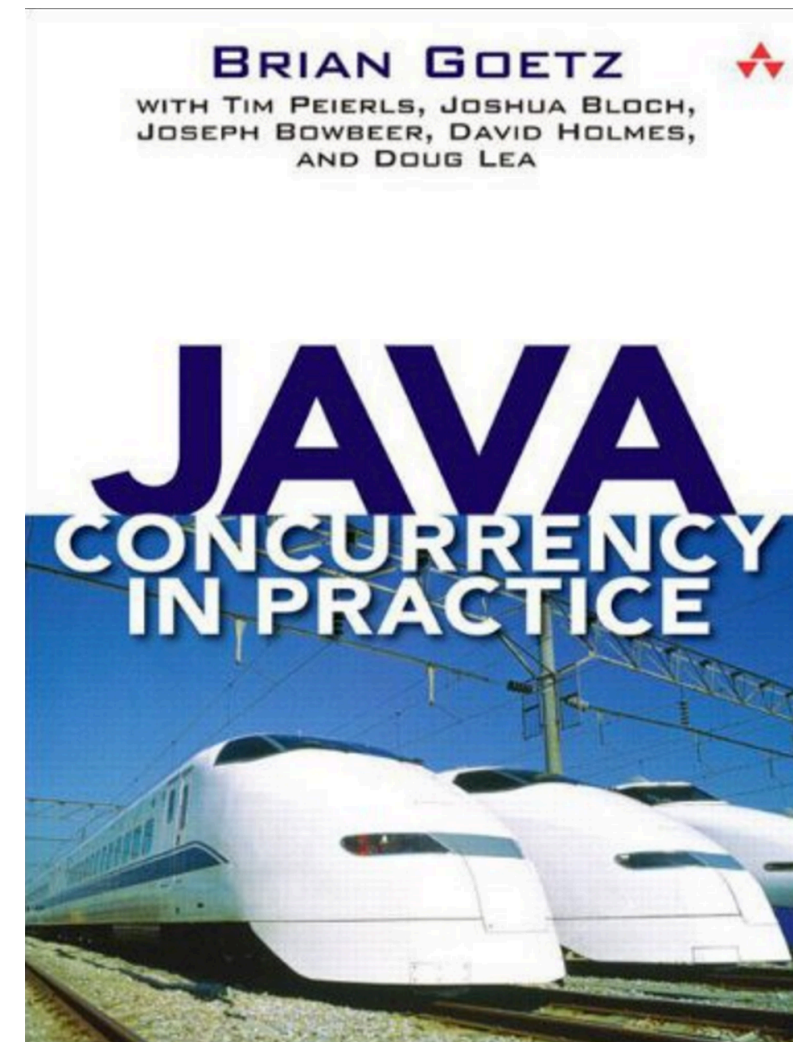




Java memory model

Further reading

- `java.util.concurrent` JavaDocs
- Java Concurrency in Practice
- Anything by Doug Lea
 - Including source code
- Anything by Martin Thompson,
 - @mjpt777
- Netty



Questions?

ngn@spotify.com

[@protocol7](#)

