Chronicle SOFTWARE

Microservices: Single digit microseconds latency

Dmitry Pisklov

26 OCT 2019

Outline

Chronicle SOFTWARE

- Microservice application: architecture example
- What is latency and how can we measure it
- Reducing the latency

Intro: Who we are

Chronicle SOFTWARE

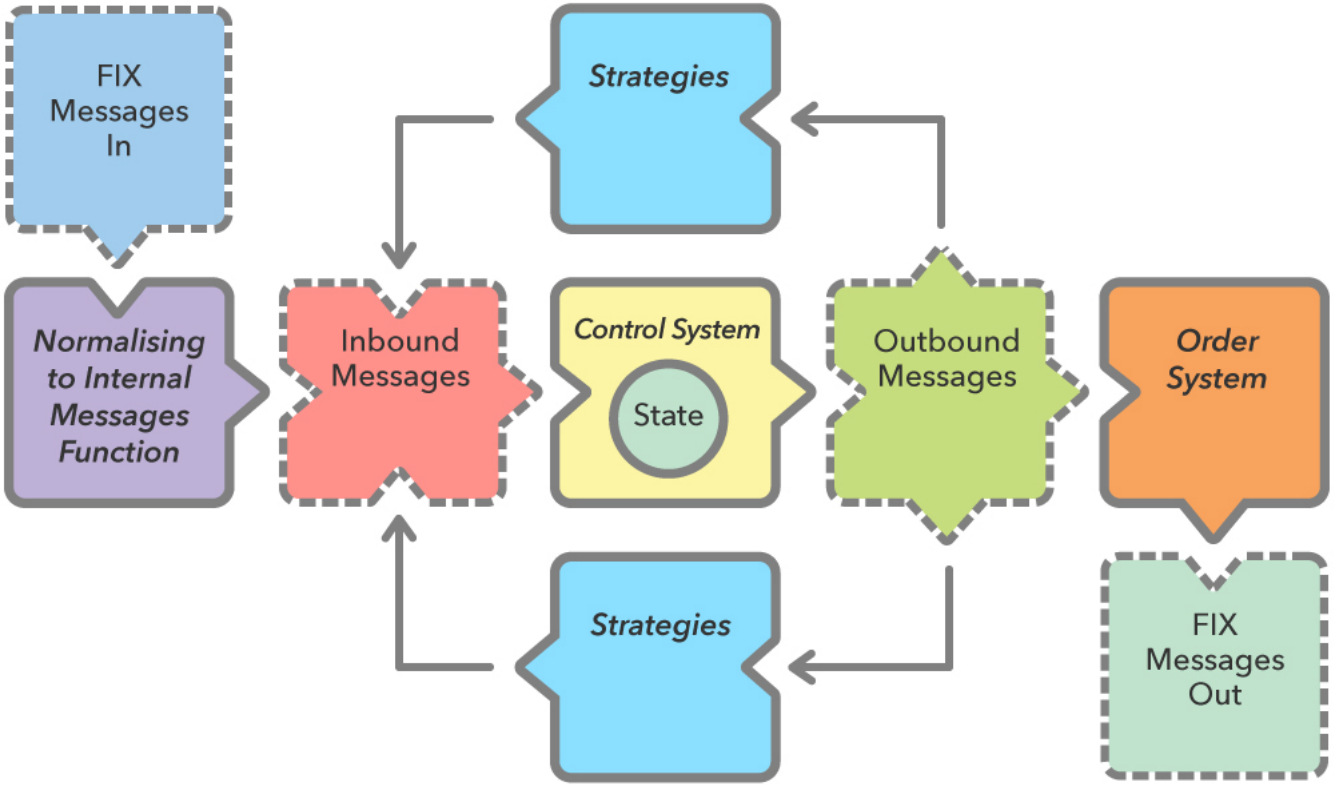
- Dmitry Pisklov
  - Developer @ Chronicle Software
- Chronicle Software founder – Peter Lawrey
  - Java Champion
  - Most answers for Java and JVM on StackOverflow.com

Chronicle SOFTWARE

0. Microservices: Architecture Example

0.0. Micro-services Architecture Example

Chronicle SOFTWARE

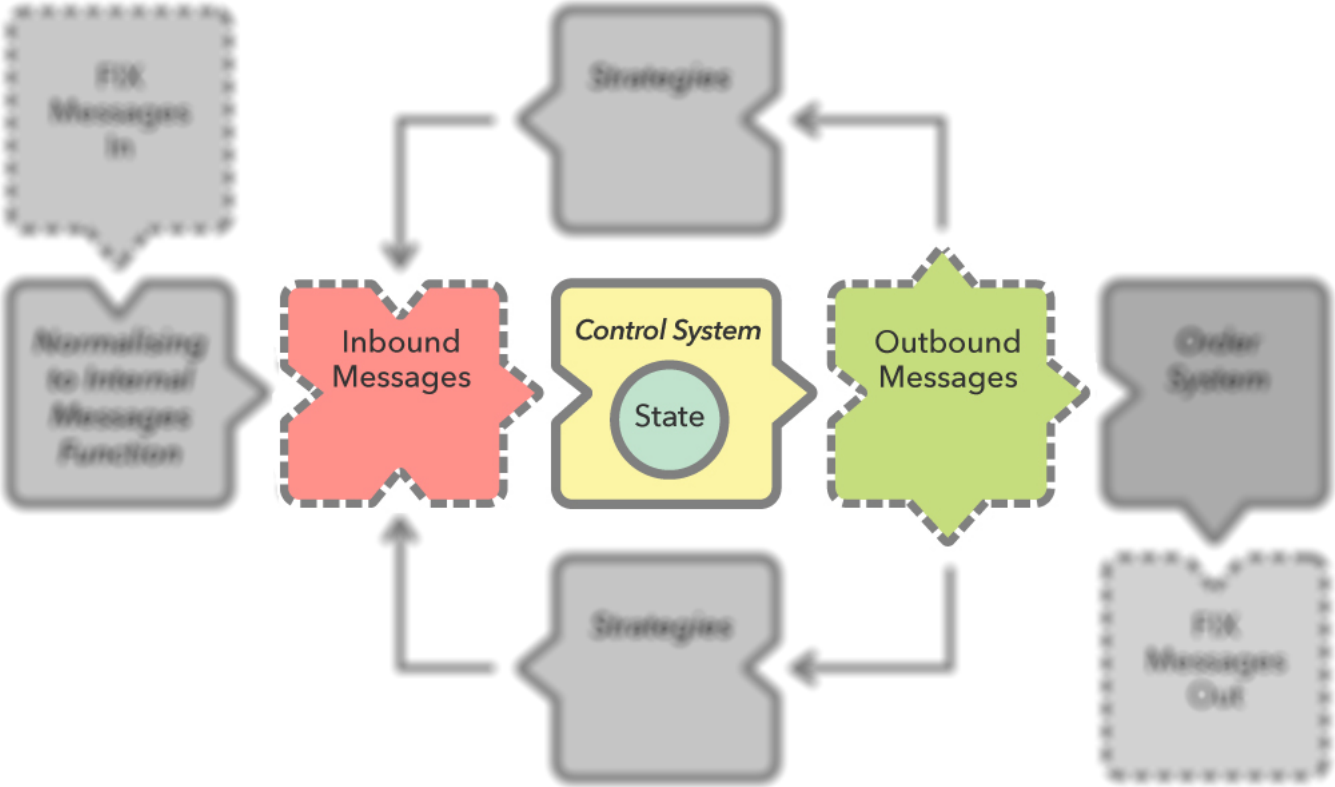


0.1. Micro-services Architecture Example

Chronicle SOFTWARE

0.2. Micro-services Architecture Example

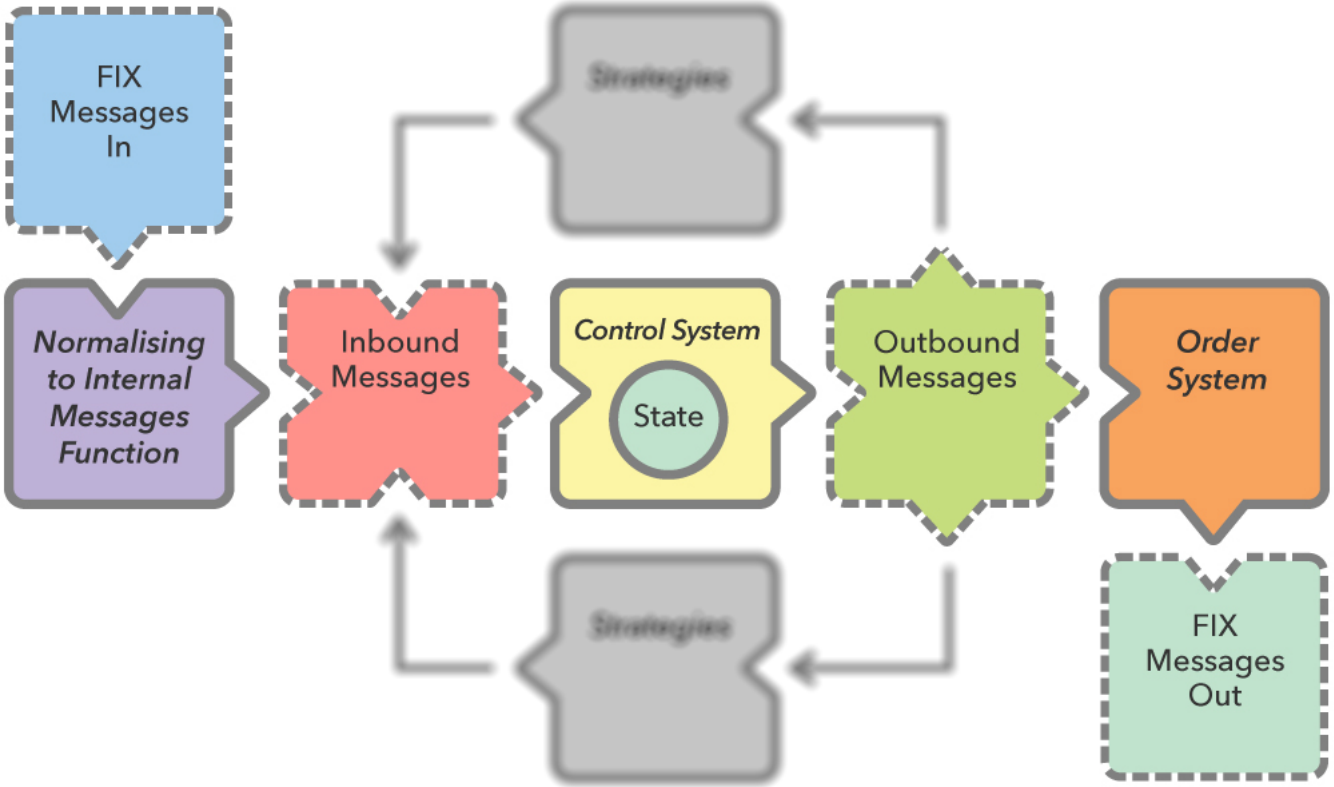
Chronicle SOFTWARE



0.3. Micro-services Architecture Example

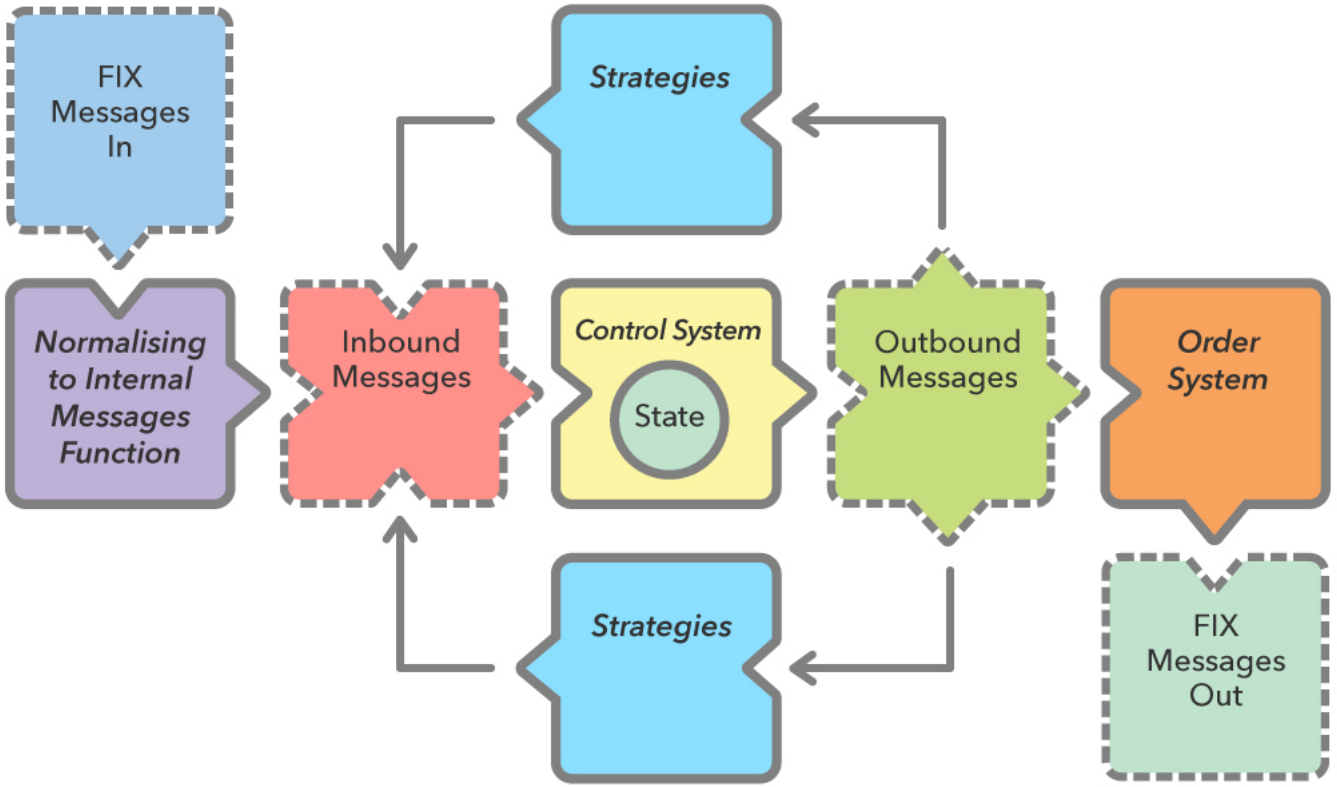
0.4. Micro-services Architecture Example

Chronicle SOFTWARE

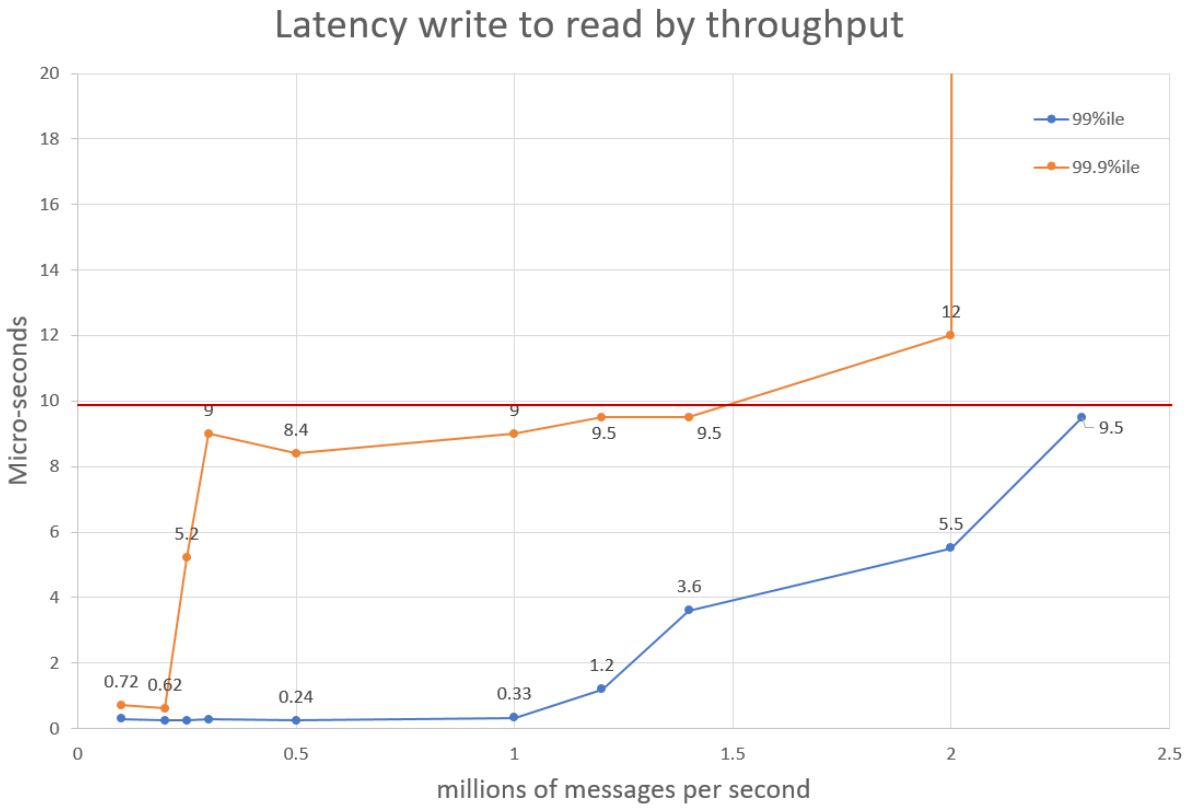


0.5. Micro-services Architecture Example

Chronicle SOFTWARE



0.6. How fast is fast? Example measured



Chronicle SOFTWARE

1. Latency: what is it and how to measure latency?

1.0. Measuring latency

Chronicle SOFTWARE

- What is Latency?
- Latency in Microservices

1.0. Measuring latency

Chronicle SOFTWARE

- What is Latency?
- Latency in Microservices
  - Service response time
    - Marshaling time
    - Computation time

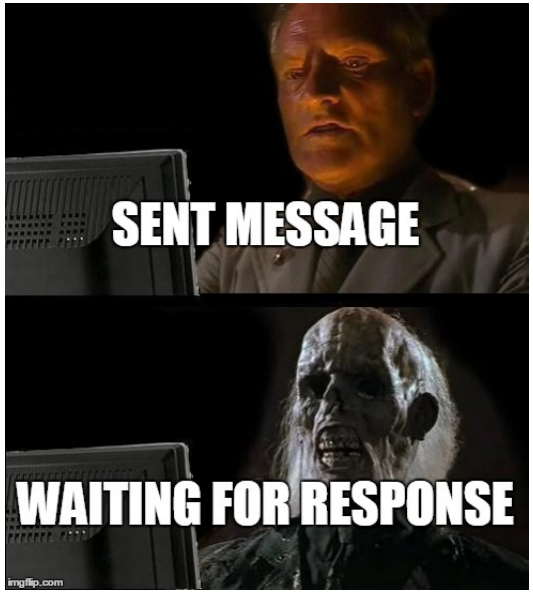
1.0. Measuring latency

Chronicle SOFTWARE

- What is Latency?
- Latency in Microservices
  - Service response time
    - Marshaling time
    - Computation time
  - IPC latency

Chronicle SOFTWARE

- Measuring latency – how?



Chronicle SOFTWARE

- JMH is for Micro-benchmarks
- JLBH – Java Latency Benchmark Harness
- Documentation and examples
  - GitHub - https://github.com/OpenHFT/Chronicle-Core#jlbh

1.2. JMH vs JLBH: why we use JLBH

Chronicle SOFTWARE

- Code running in context – full stack
- Variable throughput
  - For each throughput we can estimate max tail latency and provide SLAs
- Accounts for coordinated omission
- Various sampling points in the code

1.3. Measuring latency – JLBH example

Chronicle SOFTWARE

```
End to End: (1,000,000)                          50/90 99/99.9 99.99/99.999 - worst was 7.8 /
--------------------------------------------------------------------------------------------
-------------------------------- BENCHMARK RESULTS (RUN 5) ----------------------------------
Run time: 10.825s
Correcting for co-ordinated:false
Target throughput:100000/s = 1 message every 10us
End to End: (1,000,000)                          50/90 99/99.9 99.99/99.999 - worst was 7.5 /
--------------------------------------------------------------------------------------------
--------------------------------- SUMMARY (end to end)-----------------------------------------
Percentile   run1         run2         run3         run4         run5      % Variation
50:             7.99         7.53         7.22         7.83         7.49         5.32
90:          6047.74         8.62        10.05        10.20        93.15        86.73
99:         22994.94       747.78      2485.25      7362.56      8646.66        87.57
99.7:       31825.92      1956.35     44613.63     36126.72     14348.29        93.56
99.9:       41402.37      3245.06     64307.20     56115.20     18063.36        92.62
worst:      51265.54      8884.22    194576.38     77299.71     24698.88        93.30
--------------------------------------------------------------------------------------------
```

1.3. Measuring latency – JLBH example

Chronicle SOFTWARE

```
End to End: (1,000,000)                          50/90 99/99.9 99.99/99.999 - worst was 7.8 /
--------------------------------------------------------------------------------------------
-------------------------------- BENCHMARK RESULTS (RUN 5) ----------------------------------
Run time: 10.825s
Correcting for co-ordinated:false
Target throughput:100000/s = 1 message every 10us
End to End: (1,000,000)                          50/90 99/99.9 99.99/99.999 - worst was 7.5 /
--------------------------------------------------------------------------------------------
--------------------------------- SUMMARY (end to end)-----------------------------------------
Percentile   run1         run2         run3         run4         run5      % Variation
50:             7.99         7.53         7.22         7.83         7.49         5.32
90:          6047.74         8.62        10.05        10.20        93.15        86.73
99:         22994.94       747.78      2485.25      7362.56      8646.66        87.57
99.7:       31825.92      1956.35     44613.63     36126.72     14348.29        93.56
99.9:       41402.37      3245.06     64307.20     56115.20     18063.36        92.62
worst:      51265.54      8884.22    194576.38     77299.71     24698.88        93.30
--------------------------------------------------------------------------------------------
```

Chronicle SOFTWARE

- Community asks – we do!

Chronicle SOFTWARE

2. Fighting latency in Your Software

2.0. Use less code to achieve perfection

Chronicle SOFTWARE

“Perfection is achieved, not when there is nothing more to add, but when there is nothing left to take away.”

– Antoine de Saint-Exupéry

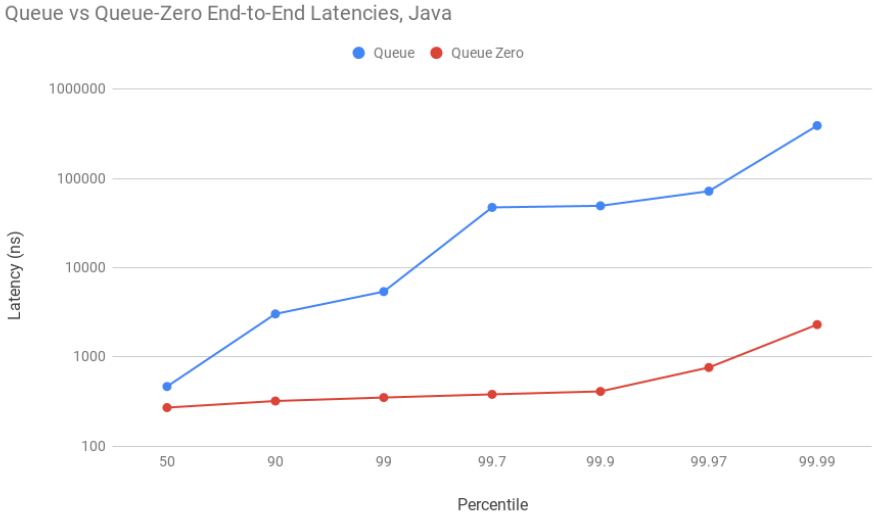
Chronicle SOFTWARE

- JVM only inlines smaller methods
  - `-XX:InlineSmallCode=`*`size`*
  - `-XX:MaxInlineSize=`*`size`*
- Large methods are not JIT-compiled
  - `-XX:-DontCompileHugeMethods`
  - `-XX:HugeMethodLimit=`*`size`*

2.2. Use specializations – with care

Chronicle SOFTWARE

- Specialized code is faster
  - Less checks, less conditions, less data written
  - More limitations



2.3. Generate optimized code

Chronicle SOFTWARE

- Primitive specializations
- Generated efficient marshaling/unmarshaling code
- Koloboke collections by Roman Leventov – only generates actually used methods

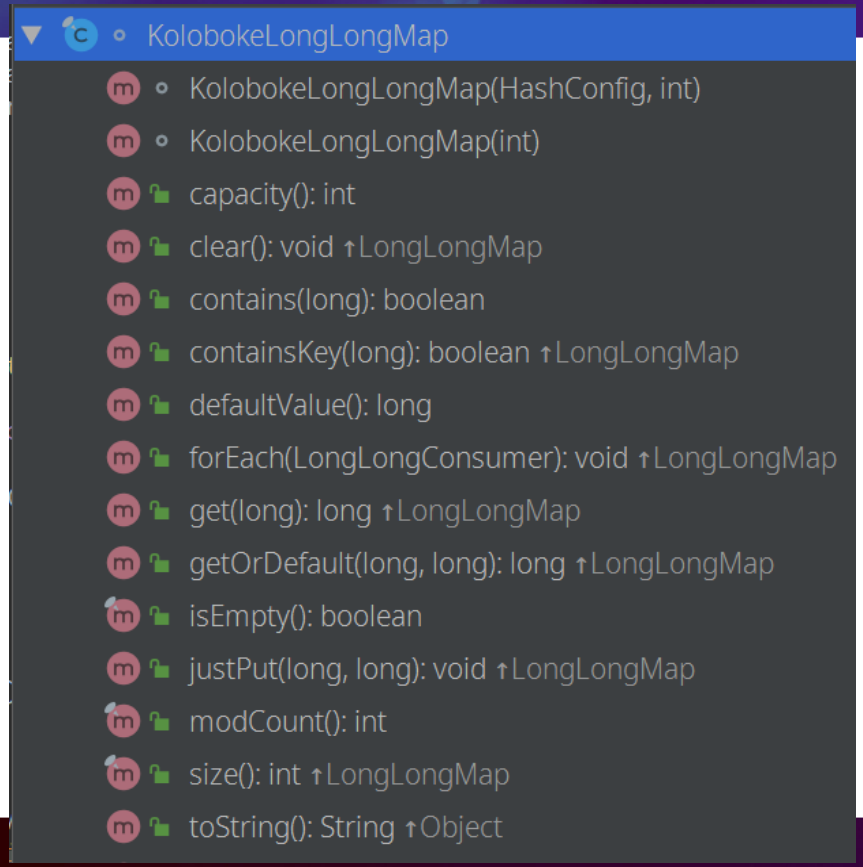
2.4.0. Specialization example – Koloboke

Chronicle SOFTWARE

```
@KolobokeMap
public abstract class LongLongMap {

    public abstract void justPut(long key, long value);

    public abstract long get(long key);

    public abstract int size();

    public abstract boolean containsKey(long key);

    public abstract void clear();

    public abstract void forEach(LongLongConsume var1);

}
```

2.4.1. Specialization example – Koloboke

Chronicle SOFTWARE



2.5. Multithreading? Forget it!

Chronicle SOFTWARE

- Threads are evil (for microservices)
  - Single-threaded application with event loop
    - Even faster on dedicated CPU

2.6. Multithreading? Forget it!

Chronicle SOFTWARE

- Threads are evil (for microservices)
  - Single-threaded application with event loop
    - Even faster on dedicated CPU
  - Shared memory & CAS for synchronization – only when needed, avoid sharing data

Chronicle SOFTWARE

- Threads are evil (for microservices)
  - Single-threaded application with event loop
    - Even faster on dedicated CPU
  - Shared memory & CAS for synchronization – only when needed, avoid sharing data
  - Memory barriers – use minimally required!
    - StoreLoad (volatile)
    - StoreStore (ordered a.k.a. lazySet)

2.8. Memory barriers benchmarked

Chronicle SOFTWARE

- How fast are different barriers?

```
public final void lazySet(long newValue) {
    unsafe.putOrderedLong( o: this, valueOffset, newValue);
}
```

Chronicle SOFTWARE

- How fast are different barriers?

```
public final void lazySet(long newValue) {
    unsafe.putOrderedLong( o: this, valueOffset, newValue);
}
```

Benchmark	Mode	Samples	Score	Score error	Units
lazySetLong	avgt	5	17.630	0.650	ns/op
volatileSetLong	avgt	5	23.009	0.794	ns/op

2.9. Memory barriers usage example

Chronicle SOFTWARE

```
public WireOut marshallable(@NotNull WriteMarshallable object) {
    long position = bytes.writePosition();
    bytes.writeInt(0);

    object.writeMarshallable( wire: RawWire.this);

    int length = Maths.toInt32( x: bytes.writePosition() - position - 4);
    bytes.writeOrderedInt(position, length);
    return RawWire.this;
}
```

2.10. Java – what means my name to you?

Chronicle SOFTWARE

- Know your language
  - How efficient JDK data structures are?

2.11. Java SDK efficiency

Chronicle SOFTWARE

- Know your language
  - How efficient JDK data structures are? HashMap#put:

```
if ((p = tab[i = (n - 1) & hash]) == null)
    tab[i] = newNode(hash, key, value, next: null);
```

```
Node<K,V> newNode(int hash, K key, V value, Node<K,V> next) {
    return new Node<>(hash, key, value, next);
}
```

2.12. YAGNI

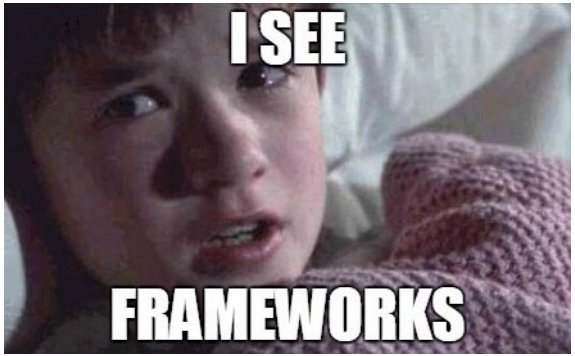
Chronicle SOFTWARE

- Know your language (Doug bless Java!)
- YAGNI – don’t write code for what you don’t use

2.13. To framework or not to framework

Chronicle SOFTWARE

- Know your language (Doug bless Java!)
- YAGNI – don’t write code for what you don’t use
- If it can be done without a 3rd party library/framework – do it!



2.14. Safety first?

Chronicle SOFTWARE

- Know your language (Doug bless Java!)
- YAGNI – don’t write code for what you don’t use
- If it can be done without a 3rd party library/framework – do it!
- Don’t be afraid to be Unsafe – it’s not scary! (provided you know what you are doing…)

2.15.0. Cutting off safety nets – example

Chronicle SOFTWARE

```
public int read(ByteBuffer buf) throws IOException {

    if (buf == null)
        throw new NullPointerException();

    synchronized (readLock) {
        if (!ensureReadOpen())
            return -1;
        int n = 0;
        try {
            begin();

            synchronized (stateLock) {
                if (!isOpen()) {
                    return 0;
                }
                readerThread = NativeThread.current();
            }

            for (;;) {
                n = IOUtil.read(fd, buf, -1, nd);
                if ((n == IOStatus.INTERRUPTED) && isOpen()) {
                    continue;
                }
                return IOStatus.normalize(n);
            }

        } finally {
            readerCleanup();
            end(n > 0 || (n == IOStatus.UNAVAILABLE));
            synchronized (stateLock) {
                if ((n <= 0) && (!isInputOpen))
                    return IOStatus.EOF;
            }
            assert IOStatus.check(n);

        }
    }
}
```

2.15.1. Cutting off safety nets – example

Chronicle SOFTWARE

```
synchronized (readLock) {
    if (!ensureReadOpen())
        return -1;
    int n = 0;
    try {
        begin();

        synchronized (stateLock) {
            if (!isOpen()) {
                return 0;
            }
            readerThread = NativeThread.current();
        }
```

2.15.2. Cutting off safety nets – example

Chronicle SOFTWARE

```
} finally {
    readerCleanup();
    end(n > 0 || (n == IOStatus.UNAVAILABLE));
    synchronized (stateLock) {
        if ((n <= 0) && (!isInputOpen))
            return IOStatus.EOF;
    }
```

2.15.3. Cutting off safety nets – example

Chronicle SOFTWARE

```
for (;;) {
    n = IOUtil.read(fd, buf, -1, nd);
    if ((n == IOStatus.INTERRUPTED) && isOpen()) {
        continue;
    }
    return IOStatus.normalize(n);
}
```

```
static int read(FileDescriptor fd, ByteBuffer dst, long position,
                NativeDispatcher nd)
        throws IOException
{
    if (dst.isReadOnly())
        throw new IllegalArgumentException("Read-only buffer");
    if (dst instanceof DirectBuffer)
        return readIntoNativeBuffer(fd, dst, position, nd);
```

2.15.4. Cutting off safety nets – example

Chronicle SOFTWARE

```
Class<?> fdi = Class.forName("sun.nio.ch.FileDispatcherImpl");
Method read0 = Jvm.getMethod(fdi, name: "read0", FileDescriptor.class, long.class, int.class);
READ0_MH = MethodHandles.lookup().unreflect(read0);
```

2.15.4. Cutting off safety nets – example

Chronicle SOFTWARE

```
Class<?> fdi = Class.forName("sun.nio.ch.FileDispatcherImpl");
Method read0 = Jvm.getMethod(fdi, name: "read0", FileDescriptor.class, long.class, int.class);
READ0_MH = MethodHandles.lookup().unreflect(read0);
```

```
public int read(ByteBuffer buf) throws IOException {
    if (buf == null)
        throw new NullPointerException();

    if (isBlocking() || !isOpen() || !(buf instanceof DirectBuffer))
        return super.read(buf);
    return read0(buf);
}
```

2.15.5. Cutting off safety nets – example

Chronicle SOFTWARE

```
int read0(ByteBuffer buf) throws IOException {
    final long address = ((DirectBuffer) buf).address() + buf.position();
    int n = OS.read0(fd, address, buf.remaining());
    if ((n == IOStatus.INTERRUPTED) && socketChannel.isOpen()) {
        return 0;
    }
    int ret = IOStatus.normalize(n);
    if (ret > 0)
        buf.position(buf.position() + ret);
    else if (ret < 0)
        open = false;
    return ret;
}
```

2.15.6. Cutting off safety nets – example

Chronicle SOFTWARE

- Numbers? I haz sum 4 u!
  - 50%-tile: 6.8 → 5.7 µs
  - 90%-tile: 8.2 → 7.1 µs
  - Consistently 1.1µs less
    - YMMV

2.16. Waiting for something – eagerly!

Chronicle SOFTWARE

- Use busy loops when waiting on condition and non-blocking operations
  - wait/notify or sleep are slower, and also stalling CPU
  - `while (condition) Thread.yield();`
  - `while (condition);`
    - The lowest latency
    - Avoids CPU slowdown

2.17. Busy wait vs sleep example

Chronicle SOFTWARE

- BUSY100

  50/90 97/99 99.7/99.9 99.97/99.99 - worst
  0.095/0.11 0.11/0.16 0.36/0.65 0.65/0.65-0.65

```
BUSY100 {
    public void disturb() { busyWait( nanos: 1.0E8D); }
},
```

- PAUSE1

  50/90 97/99 99.7/99.9 99.97/99.99 - worst
  0.26/0.34 0.59/0.66 0.71/0.75 12/13 - 16

```
PAUSE1 {
    public void disturb() { Jvm.pause( millis: 1L); }
},
```

Chronicle SOFTWARE

- Strings
  - Strings are immutable (and expensive)
  - Use StringBuilder Luke (and you can share it!)
  - Chronicle Bytes – can do much more, heap or off heap

2.19. Bytes code examples

Chronicle SOFTWARE

```
Bytes b = Bytes.from("Hello World");
try {
    b.readSkip(6);
    assertTrue(StringUtils.isEqual( S: "World", b));
} finally {
    b.release();
}
```

2.20. Bytes code examples

Chronicle SOFTWARE

```
Bytes b = Bytes.from("Hello World");
try {
    b.readSkip(6);
    assertTrue(StringUtils.isEqual( S: "World", b));
} finally {
    b.release();
}
```

```
b.append("Hello World");
b.move( from: 3,  to: 1,  length: 3);
assertEquals( expected: "Hlo o World", b.toString());
b.move( from: 3,  to: 5,  length: 3);
assertEquals( expected: "Hlo o o rld", b.toString());
```

Chronicle SOFTWARE

- Garbage Collection
  - Even minor collections are slow for us (several ms)
  - Avoid garbage at all cost

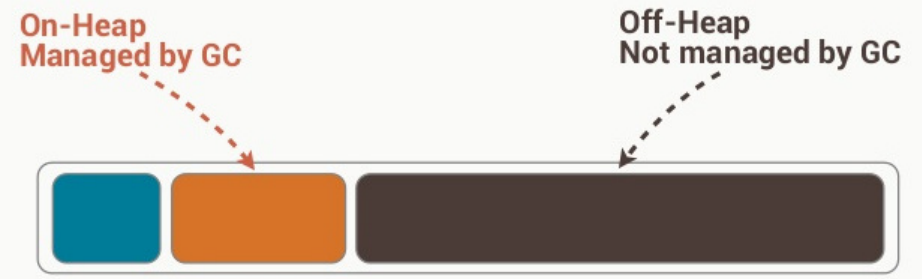
Chronicle SOFTWARE

- Garbage Collection
  - Even minor collections are slow for us (several ms)
  - Avoid garbage at all cost
    - Reuse objects (especially when marshalling)

```
private final TransactionsRequest tr = new TransactionRequest();

public receive(WireIn wire) {
    tr.reset();
    wire.read().marshallable(tr);
    process(tr);
}
```

Chronicle SOFTWARE

- Garbage Collection
  - Even minor collections are slow for us (several ms)
  - Avoid garbage at all cost
    - Reuse objects (especially when marshalling)
    - Pool objects if you can’t reuse single object
    - Most of all – use off-heap memory



2.23. Object pooling example

Chronicle SOFTWARE

```
public class StringBuilderPool {
    private final ThreadLocal<StringBuilder> sbtl = withInitial(
            () -> new StringBuilder( capacity: 128));

    @ForceInline
    public StringBuilder acquireStringBuilder() {
        StringBuilder sb = sbtl.get();
        sb.setLength(0);
        return sb;
    }
}
```

Chronicle SOFTWARE

3. Fighting latency: Inter-process communication

3.0. Memory vs Network IO vs Disk IO

Chronicle SOFTWARE

- Writing/reading to/from memory is the fastest option
  - Right after CPU caches...
  - Remember about “mechanical sympathy”

3.0. Memory vs Network IO vs Disk IO

Chronicle SOFTWARE

- Writing/reading to/from memory is the fastest option
  - Right after CPU caches...
  - Remember about “mechanical sympathy”
- Disk IO is slower than DC-local network IO
  - UDP is faster than TCP

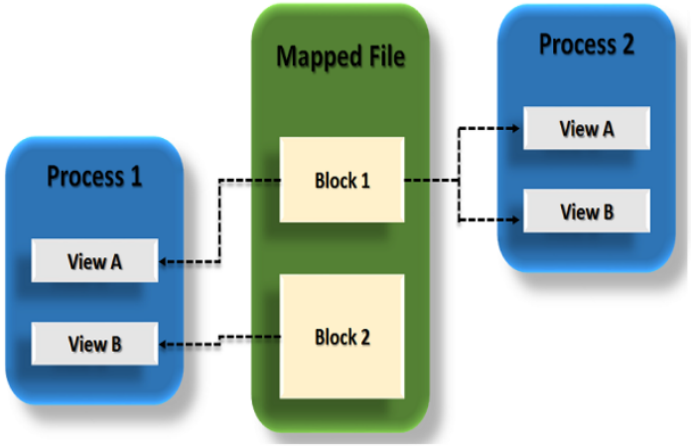
3.1. Writing to memory + writing to disk

Chronicle SOFTWARE

- What if we can write to main memory while OS writes to (local) disk for us?

Chronicle SOFTWARE

- What if we can write to main memory while OS writes to (local) disk for us?
- Welcome to memory-mapped files



Chronicle SOFTWARE

- What if we can write to main memory while OS writes to (local) disk for us?
- Welcome to memory-mapped files
  - Memory-mapping is shared between processes – effectively providing shared memory IPC

Chronicle SOFTWARE

- Chronicle Queue – uses off-heap memory to map files
  - **4µs roundtrip** on consumer-grade (a.k.a. desktop) box for 1024 bytes-long message

Chronicle SOFTWARE

- [Un]Marshaling can be the biggest contribution to latency
  - Choose (and benchmark) your tools
    - SBE (Agrona)
    - Chronicle Wire
    - Protobuf
    - FlatBuffers etc...

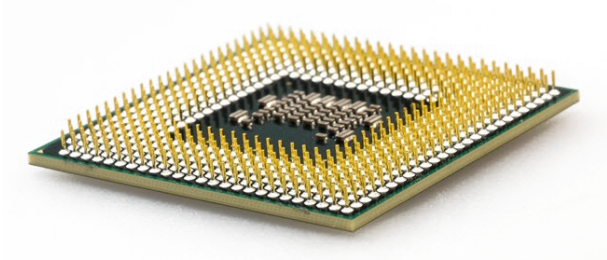
Chronicle SOFTWARE

- [Un]Marshaling can be the biggest contribution to latency
  - Choose (and benchmark) your tools
    - SBE (Agrona)
    - Chronicle Wire
    - Protobuf
    - FlatBuffers etc...

4. Fighting latency: Environment (OS & hardware)

Chronicle SOFTWARE

- Intel C-states
  - They will kill your latency!
  - `intel_idle.max_cstate=0 processor.max_cstate=0 idle=poll`



Chronicle SOFTWARE

- Intel C-states
  - They will kill your latency!
- Turbo boost
  - Check your BIOS
  - Depends on thermal envelope
    - Careful – AVX throttling!

4.2. CPU: governors

Chronicle SOFTWARE

- Intel C-states
  - They will kill your latency!
- Turbo boost
- CPU caches
- Linux CPU governors

```
# cpupower frequency-set -g performance
```

Chronicle SOFTWARE

- Intel C-states
  - They will kill your latency!
- Turbo boost
- CPU caches
- Linux CPU governors
- Cool CPU is – surprisingly – slow

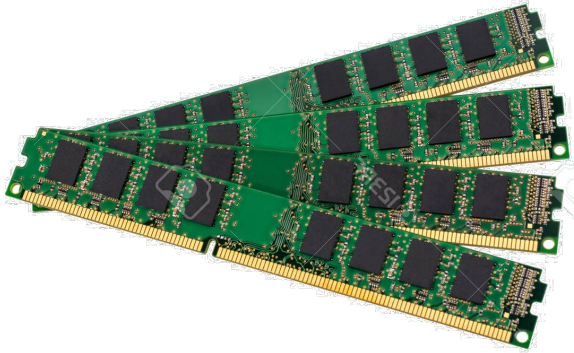
Chronicle SOFTWARE

- Isolate OS threads
  - `isolcpus`
- Isolate IRQs
- Threads CPU affinity

For more – come to the discussion zone!

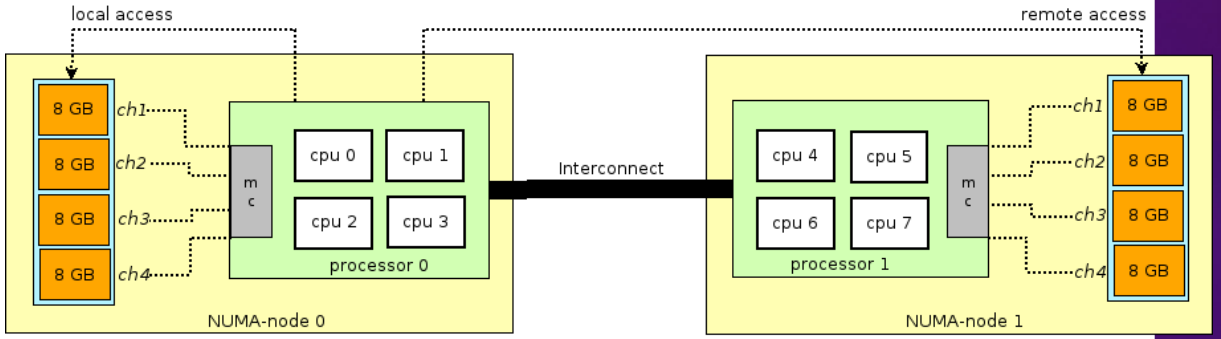
Chronicle SOFTWARE

- Swap kills your performance
  - `sysctl -w vm.swappiness=0`



Chronicle SOFTWARE

- Swap kills your performance
  - `sysctl -w vm.swappiness=0`
- NUMA
  - `numactl`
  - Disable node interleaving



4.7. Memory management: THP

Chronicle SOFTWARE

- Swap kills your performance
  - `sysctl -w vm.swappiness=0`
- NUMA
  - `numactl`
  - Disable node interleaving
- Transparent Huge Pages are bad
  - `transparent_hugepage=never`

4.8. Disk: IO scheduler

Chronicle SOFTWARE

- SSD disks only
  - Disable IO scheduler
    - `elevator=noop`

Chronicle SOFTWARE

- SSD disks only
  - Disable IO scheduler
- File system matters
  - ext4 is faster than xfs
    - `barrier=0`
    - `noatime`

Chronicle SOFTWARE

- Kernel operations are expensive
  - Kernel bypass is faster
    - Solarflare networks

Chronicle SOFTWARE

- Kernel operations are expensive
  - Kernel bypass is faster
    - Solarflare networks
  - Memory-mapped writes are user-space
    - OS later flushes data to disk asynchronously

Chronicle SOFTWARE

- System-specific tools
  - RedHat/CentOS: tuned
    - `tuned-adm profile latency-performance`

Chronicle SOFTWARE

- System-specific tools
  - RedHat/CentOS: tuned
    - `tuned-adm profile latency-performance`
- Optimizing network IO
  - Kernel TCP buffers
    - `sysctl -w net.core.rmem_max=2097152`
    - `sysctl -w net.core.wmem_max=2097152`

Summary

Summary & take away

Chronicle SOFTWARE

- Microservices – not necessarily slow thing in the cloud
- JLBH rulez – use it!
- Your environment can be friend – or foe, your choice
  - If you do the homework

Chronicle SOFTWARE

Thank you for your attention!

Q & A

Chronicle SOFTWARE

Linkedin: “Chronicle Performance Engineers”

Blog: http://vanilla-java.github.io/

Blog: http://blog.pisklov.me

http://chronicle.software

https://github.com/OpenHFT/