Nothing is ever simple

The Tale of a Performance Improvement

Daniel Mitterdorfer, @dmitterd

2017-03-22

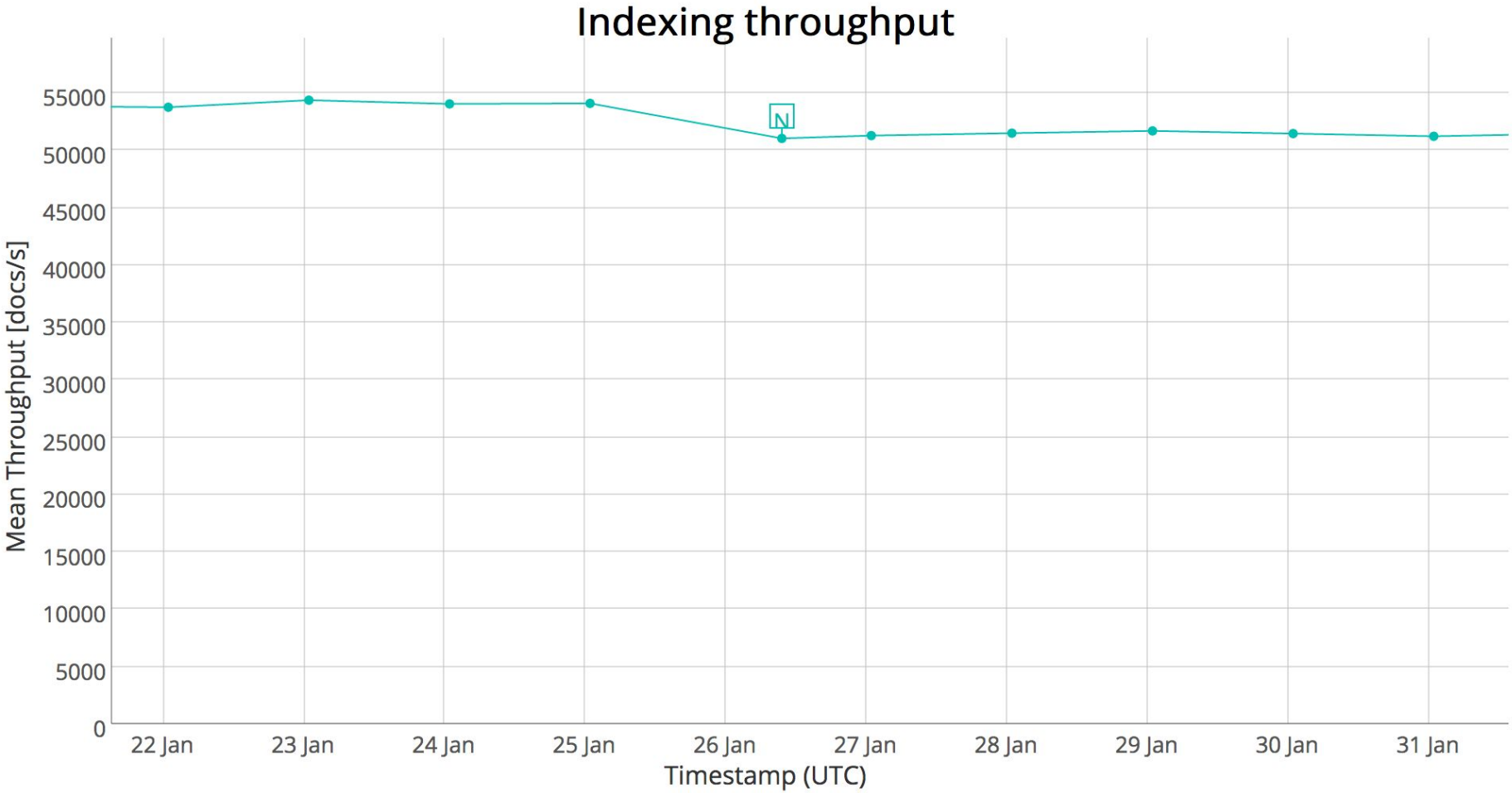
elastic

The Code Change

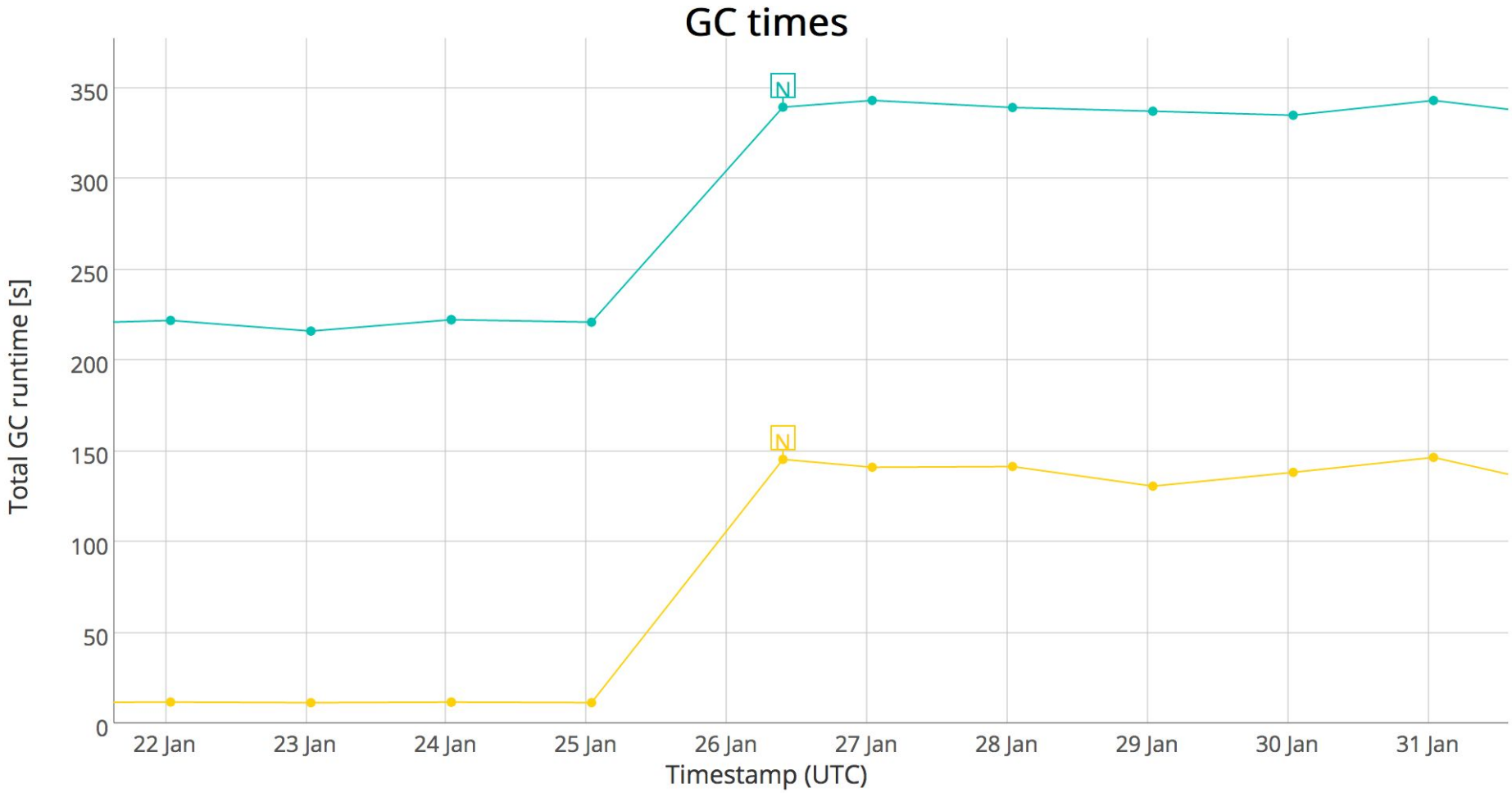
12 ...ansport-netty4/src/main/java/org/elasticsearch/http/netty4/Netty4HttpServerTransport.java View

```
@@ -143,17 +143,7 @@
143 143         Setting.byteSizeSetting("http.tcp.receive_buffer_size", NetworkService.TcpSettings.TCP_RECEIVE_BUFFER_SIZE,
144 144             Property.NodeScope, Property.Shared);
145 145     public static final Setting<ByteSizeValue> SETTING_HTTP_NETTY_RECEIVE_PREDICTOR_SIZE =
146     -       Setting.byteSizeSetting("transport.netty.receive_predictor_size",
147     -           settings -> {
148     -               long defaultReceiverPredictor = 512 * 1024;
149     -               if (JvmInfo.jvmInfo().getMem().getDirectMemoryMax().getBytes() > 0) {
150     -                   // we can guess a better default...
151     -                   long l = (long) ((0.3 * JvmInfo.jvmInfo().getMem().getDirectMemoryMax().getBytes()) / SETTING_HTTP_WORKER_COUNT.get
152     -                       (settings));
153     -                   defaultReceiverPredictor = Math.min(defaultReceiverPredictor, Math.max(l, 64 * 1024));
154     -               }
155     -               return new ByteSizeValue(defaultReceiverPredictor).toString();
156     -           }, Property.NodeScope);
    146 +       Setting.byteSizeSetting("http.netty.receive_predictor_size", new ByteSizeValue(64, ByteSizeUnit.KB), Property.NodeScope);
157 147     public static final Setting<ByteSizeValue> SETTING_HTTP_NETTY_RECEIVE_PREDICTOR_MIN =
158 148         byteSizeSetting("http.netty.receive_predictor_min", SETTING_HTTP_NETTY_RECEIVE_PREDICTOR_SIZE, Property.NodeScope);
159 149     public static final Setting<ByteSizeValue> SETTING_HTTP_NETTY_RECEIVE_PREDICTOR_MAX =
```


elastic


Indexing throughput
55000
50000
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
Mean Throughput [docs/s]
22 Jan
23 Jan
24 Jan
25 Jan
26 Jan
27 Jan
28 Jan
29 Jan
30 Jan
31 Jan
Timestamp (UTC)

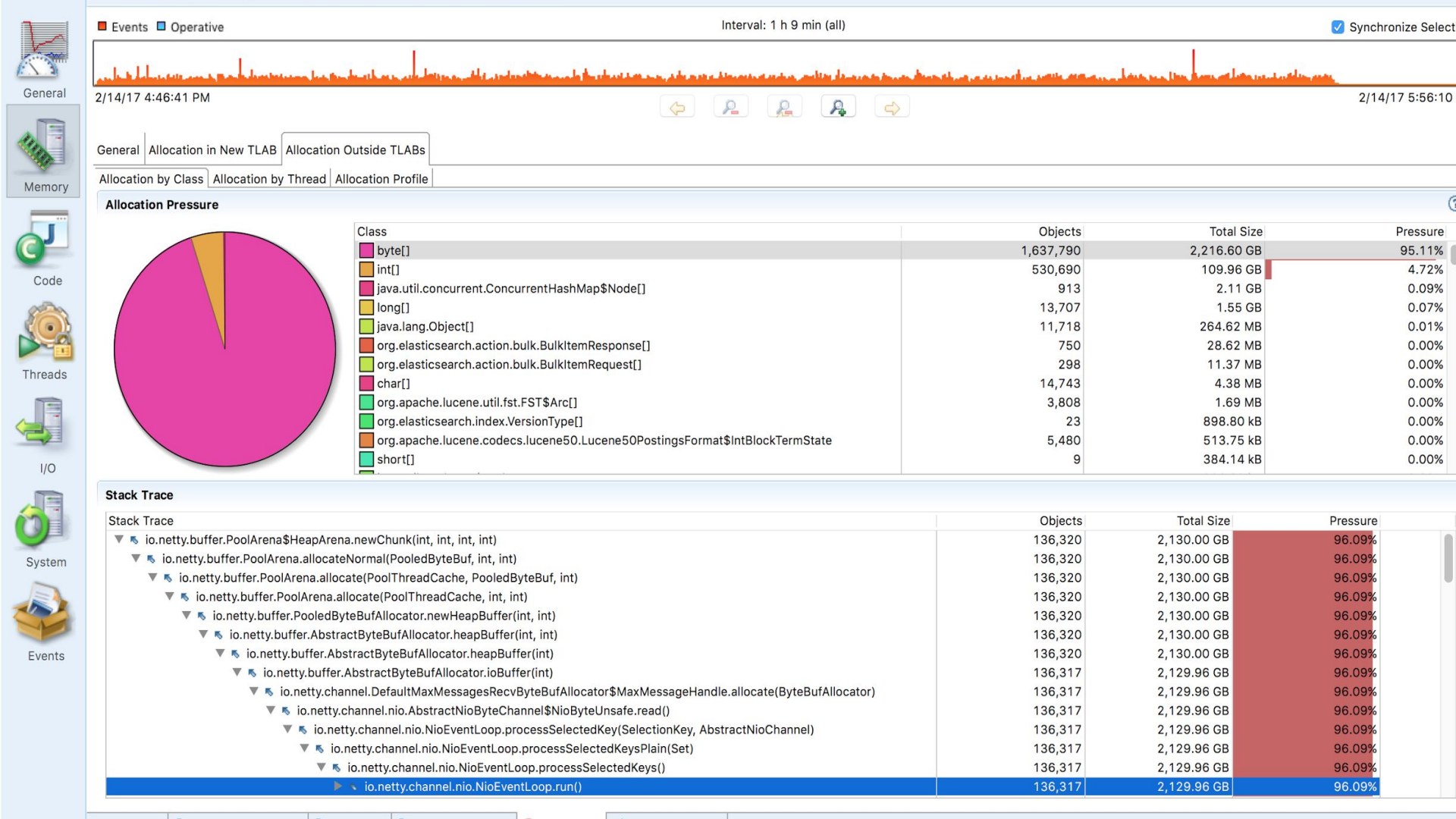
elastic


GC times
Total GC runtime [s]
350
300
250
200
150
100
50
0
N
N
22 Jan
23 Jan
24 Jan
25 Jan
26 Jan
27 Jan
28 Jan
29 Jan
30 Jan
31 Jan
Timestamp (UTC)

elastic

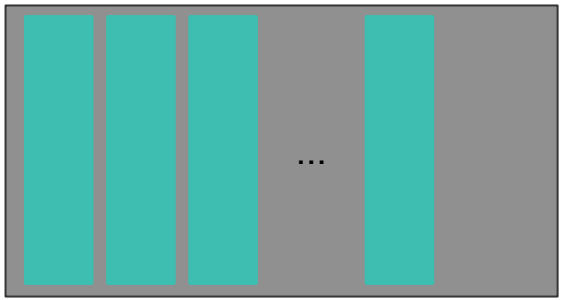
78 GB

2130 GB


Events
Operative
Interval: 1 h 9 min (all)
Synchronize Select
2/14/17 4:46:41 PM
2/14/17 5:56:10
General
Memory
Code
Threads
I/O
System
Events
General
Allocation in New TLAB
Allocation Outside TLABs
Allocation by Class
Allocation by Thread
Allocation Profile
Allocation Pressure
Class
Objects
Total Size
Pressure
byte[] 1,637,790 2,216.60 GB 95.11%
int[] 530,690 109.96 GB 4.72%
java.util.concurrent.ConcurrentHashMap$Node[] 913 2.11 GB 0.09%
long[] 13,707 1.55 GB 0.07%
java.lang.Object[] 11,718 264.62 MB 0.01%
org.elasticsearch.action.bulk.BulkItemResponse[] 750 28.62 MB 0.00%
org.elasticsearch.action.bulk.BulkItemRequest[] 298 11.37 MB 0.00%
char[] 14,743 4.38 MB 0.00%
org.apache.lucene.util.fst.FST$Arc[] 3,808 1.69 MB 0.00%
org.elasticsearch.index.VersionType[] 23 898.80 kB 0.00%
org.apache.lucene.codecs.lucene50.Lucene50PostingsFormat$IntBlockTermState 5,480 513.75 kB 0.00%
short[] 9 384.14 kB 0.00%
Stack Trace
Stack Trace
Objects
Total Size
Pressure
io.netty.buffer.PoolArena$HeapArena.newChunk(int, int, int, int) 136,320 2,130.00 GB 96.09%
io.netty.buffer.PoolArena.allocateNormal(PooledByteBuf, int, int) 136,320 2,130.00 GB 96.09%
io.netty.buffer.PoolArena.allocate(PoolThreadCache, PooledByteBuf, int) 136,320 2,130.00 GB 96.09%
io.netty.buffer.PoolArena.allocate(PoolThreadCache, int, int) 136,320 2,130.00 GB 96.09%
io.netty.buffer.PooledByteBufAllocator.newHeapBuffer(int, int) 136,320 2,130.00 GB 96.09%
io.netty.buffer.AbstractByteBufAllocator.heapBuffer(int, int) 136,320 2,130.00 GB 96.09%
io.netty.buffer.AbstractByteBufAllocator.heapBuffer(int) 136,320 2,130.00 GB 96.09%
io.netty.buffer.AbstractByteBufAllocator.ioBuffer(int) 136,317 2,129.96 GB 96.09%
io.netty.channel.DefaultMaxMessagesRecvByteBufAllocator$MaxMessageHandle.allocate(ByteBufAllocator) 136,317 2,129.96 GB 96.09%
io.netty.channel.nio.AbstractNioByteChannel$NioByteUnsafe.read() 136,317 2,129.96 GB 96.09%
io.netty.channel.nio.NioEventLoop.processSelectedKey(SelectionKey, AbstractNioChannel) 136,317 2,129.96 GB 96.09%
io.netty.channel.nio.NioEventLoop.processSelectedKeysPlain(Set) 136,317 2,129.96 GB 96.09%
io.netty.channel.nio.NioEventLoop.processSelectedKeys() 136,317 2,129.96 GB 96.09%
io.netty.channel.nio.NioEventLoop.run() 136,317 2,129.96 GB 96.09%

Buffers in Netty

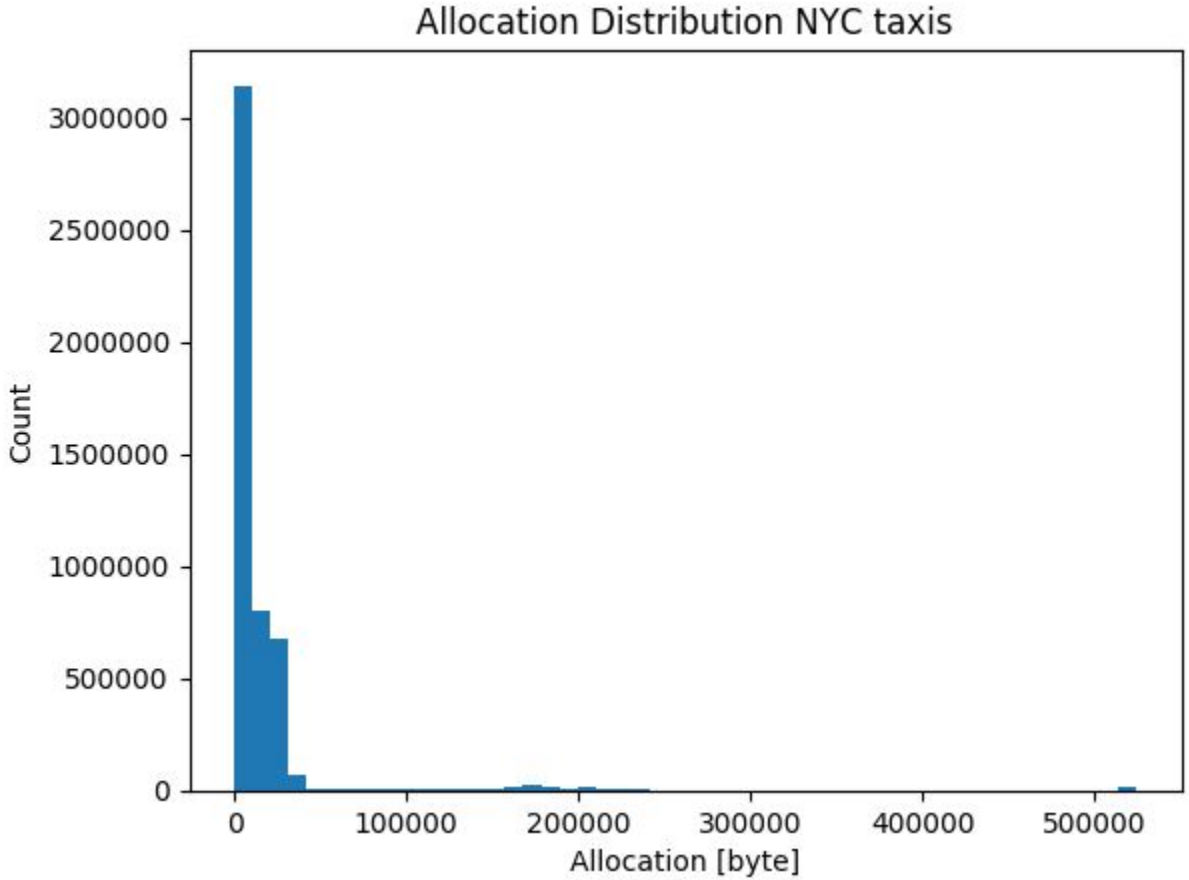
Many Buffers - One per read



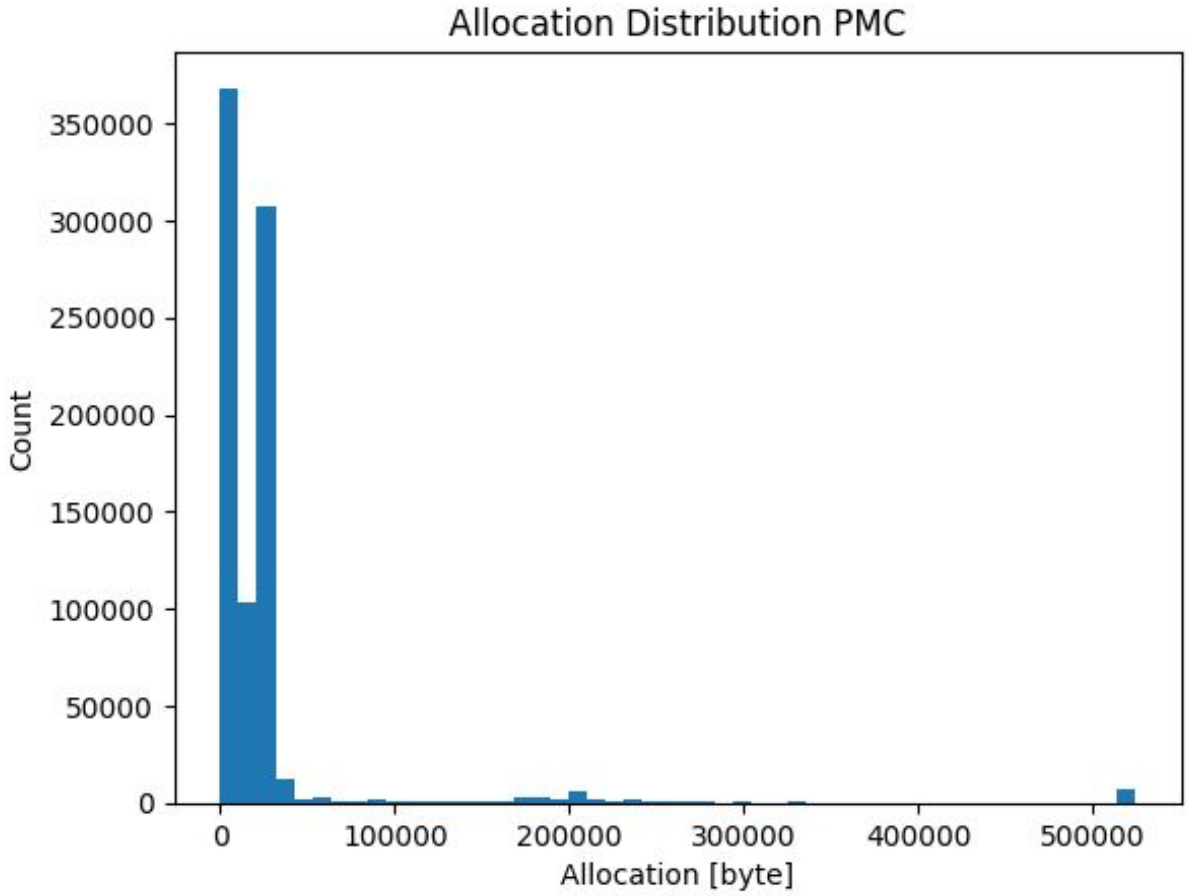
One HTTP Request

MTU

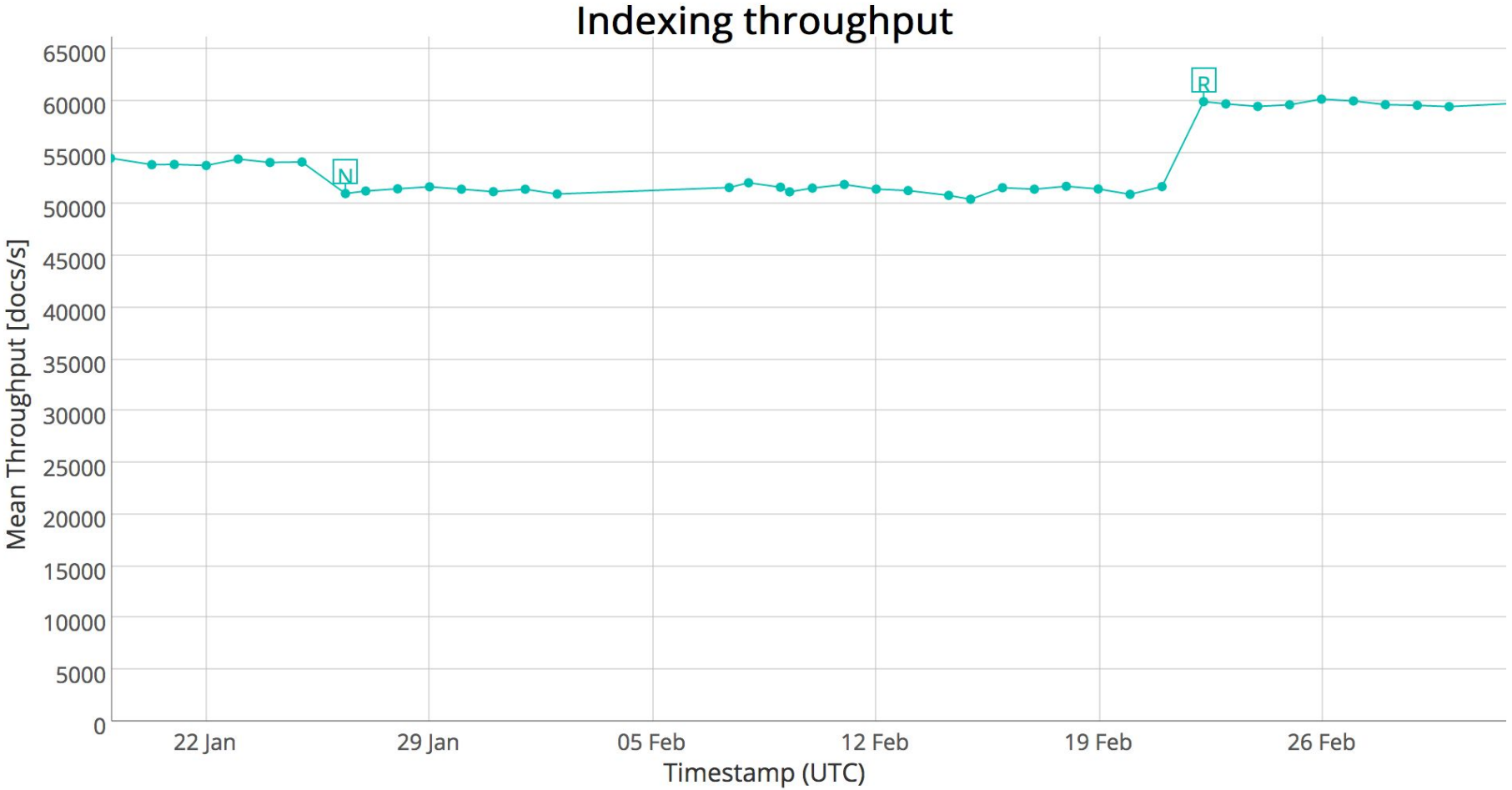
512kB


Allocation Distribution NYC taxis
3000000
2500000
2000000
1500000
1000000
500000
0
Count
0
100000
200000
300000
400000
500000
Allocation [byte]

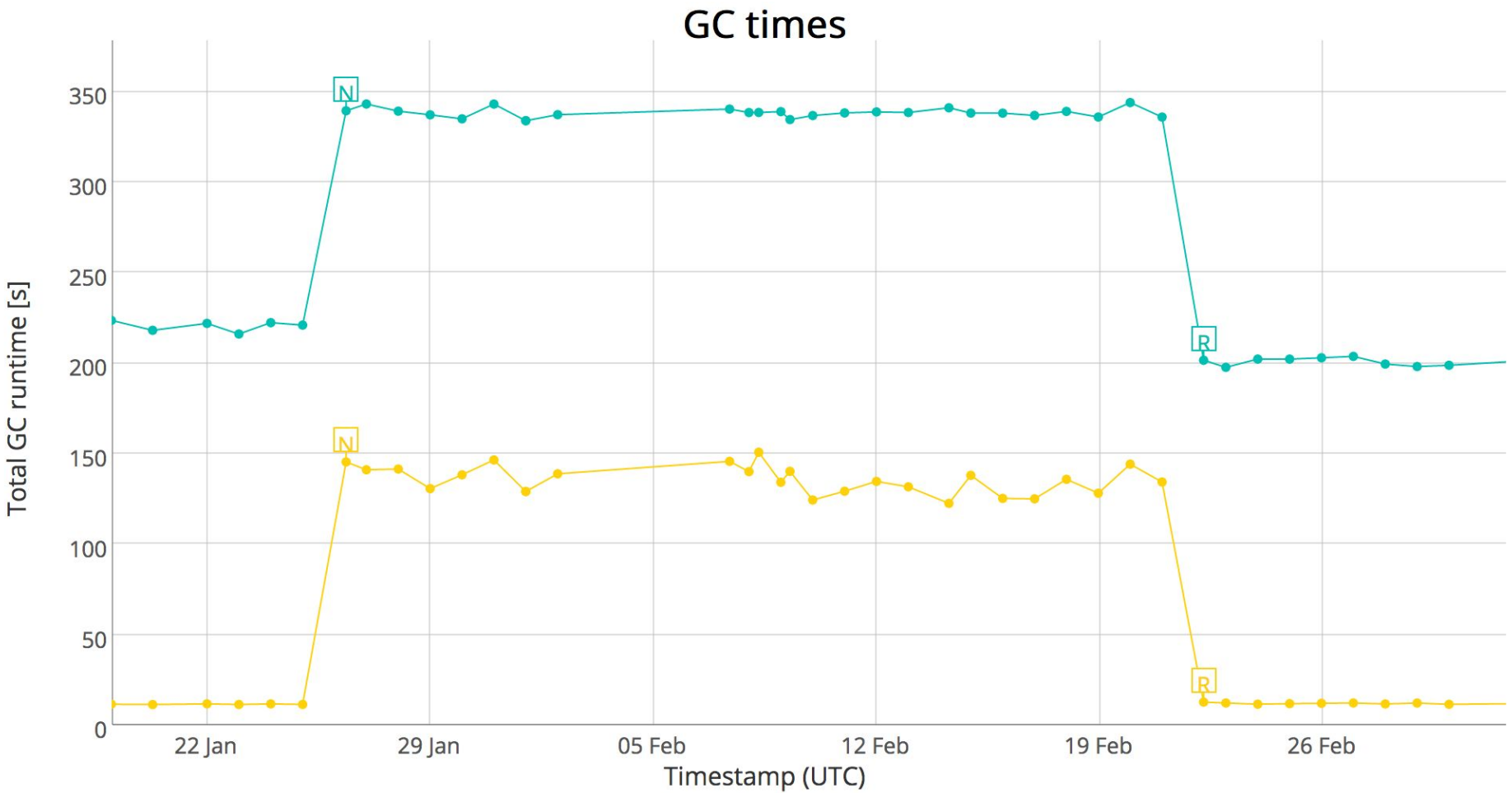
elastic


Allocation Distribution PMC
Count
350000
300000
250000
200000
150000
100000
50000
0
0
100000
200000
300000
400000
500000
Allocation [byte]

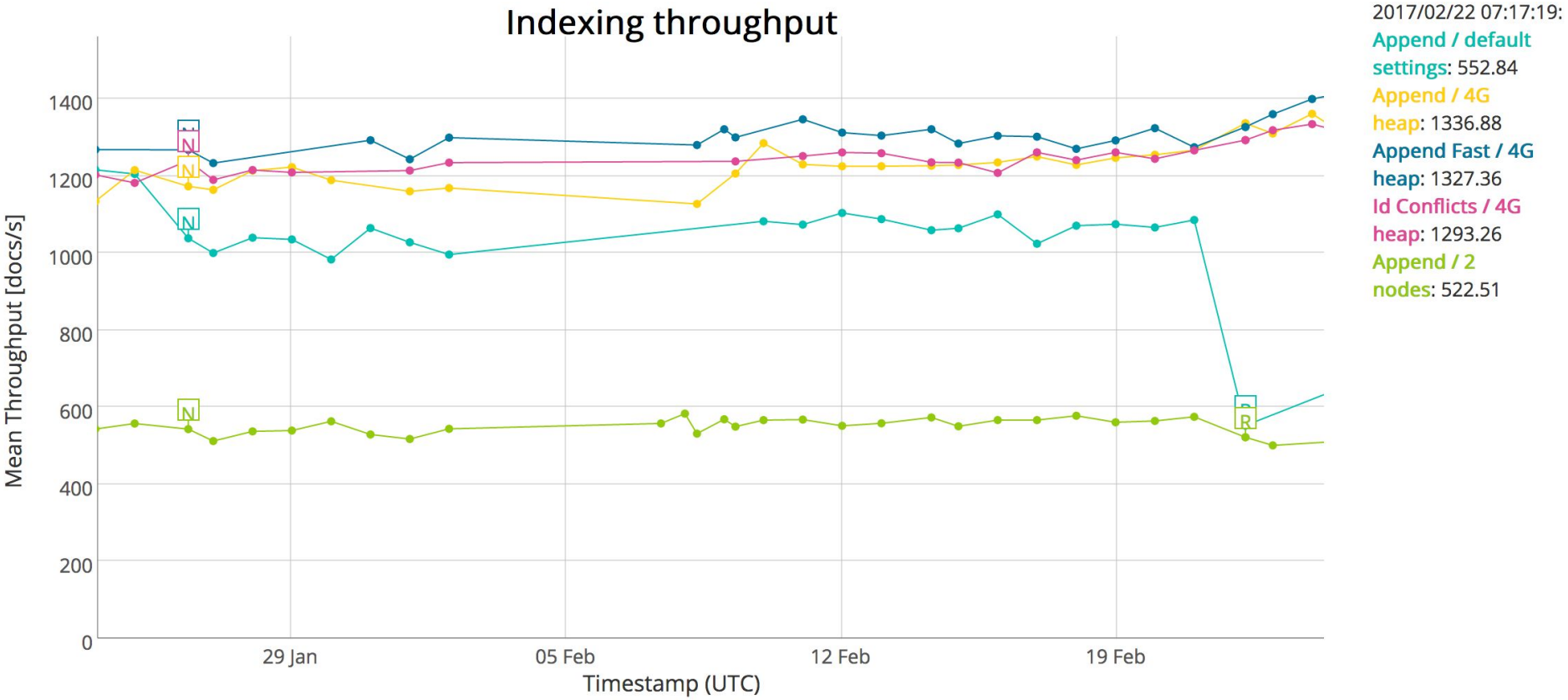
elastic


Indexing throughput
65000
60000
55000
50000
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
Mean Throughput [docs/s]
N
R
22 Jan
29 Jan
05 Feb
12 Feb
19 Feb
26 Feb
Timestamp (UTC)

elastic

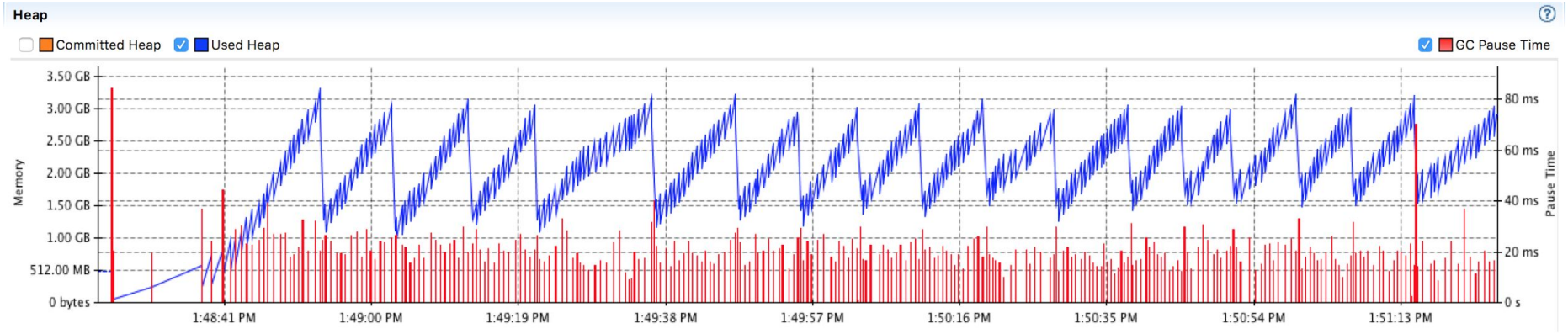

GC times
Total GC runtime [s]
350
300
250
200
150
100
50
0
N
N
R
R
22 Jan
29 Jan
05 Feb
12 Feb
19 Feb
26 Feb
Timestamp (UTC)

elastic


Indexing throughput
Mean Throughput [docs/s]
0
200
400
600
800
1000
1200
1400
29 Jan
05 Feb
12 Feb
19 Feb
Timestamp (UTC)
2017/02/22 07:17:19:
Append / default settings: 552.84
Append / 4G heap: 1336.88
Append Fast / 4G heap: 1327.36
Id Conflicts / 4G heap: 1293.26
Append / 2 nodes: 522.51

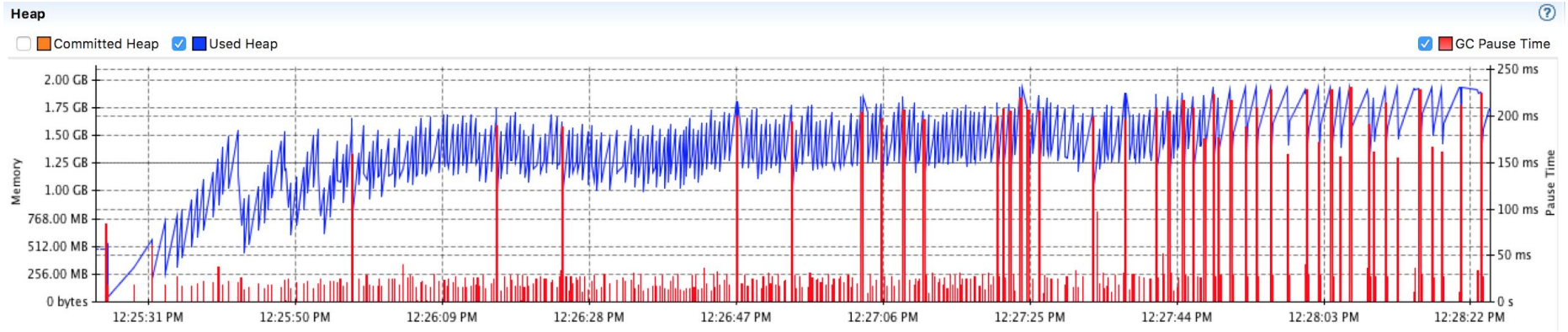
elastic

4G Heap

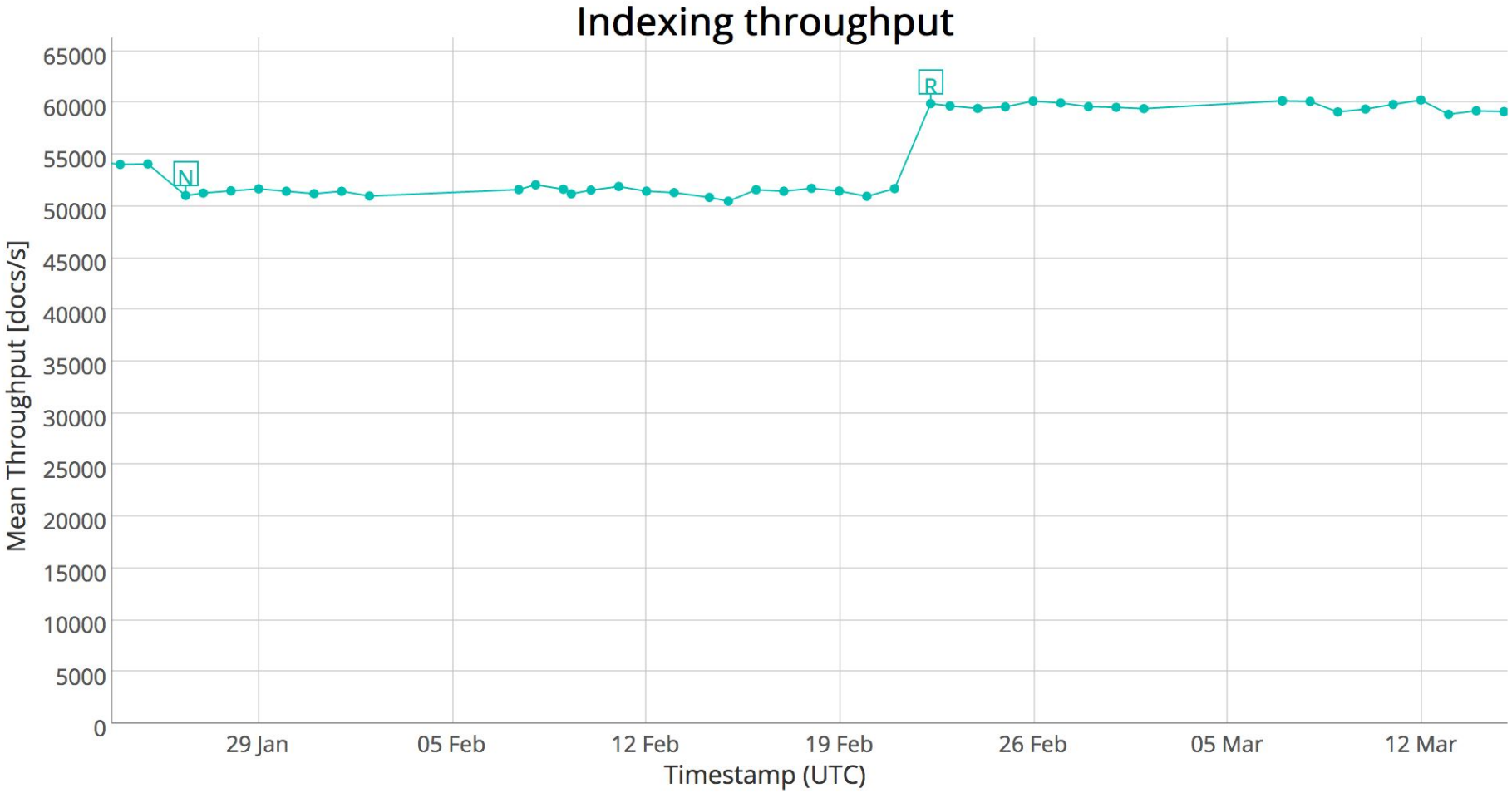


elastic

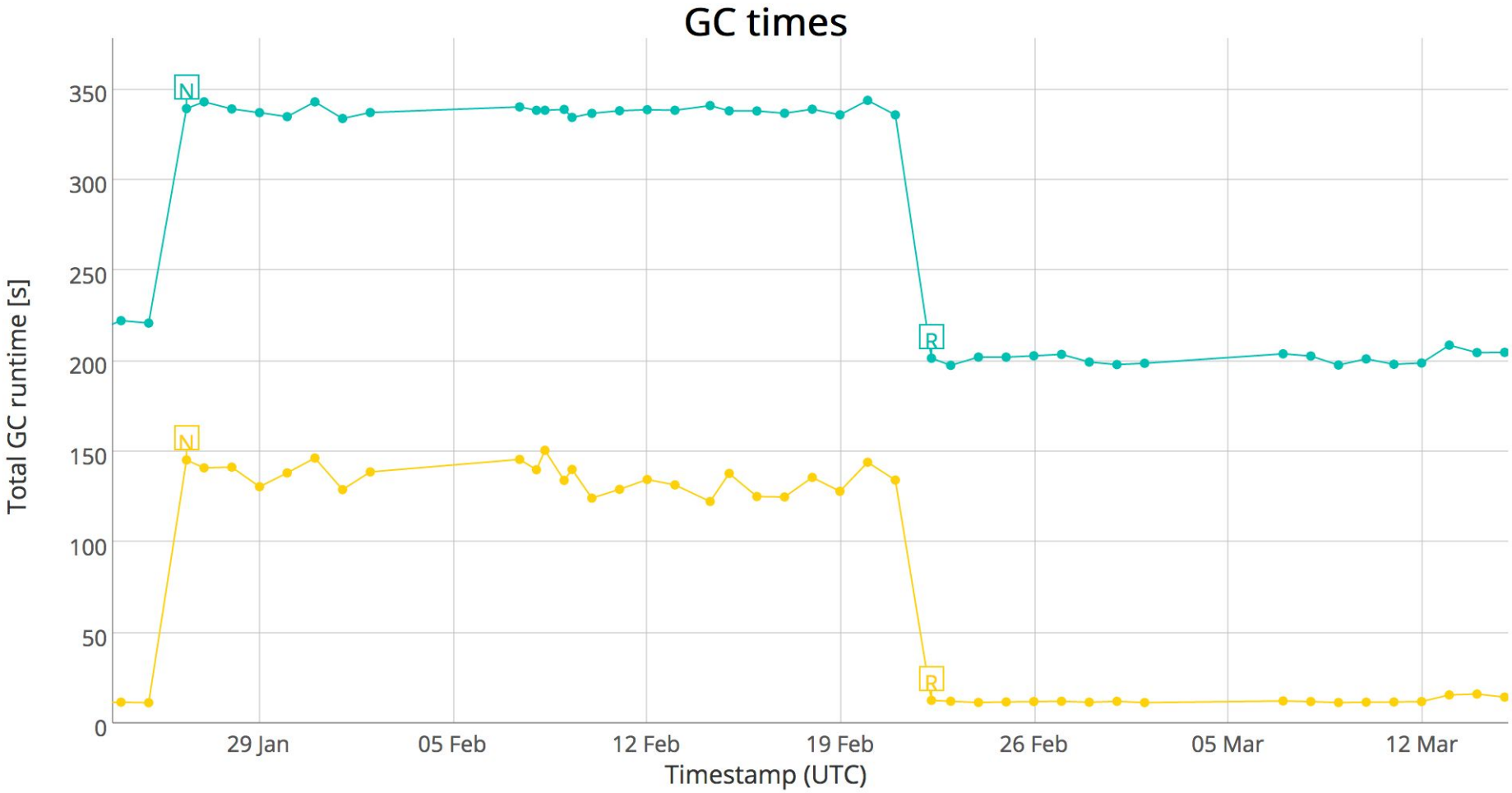
2G Heap



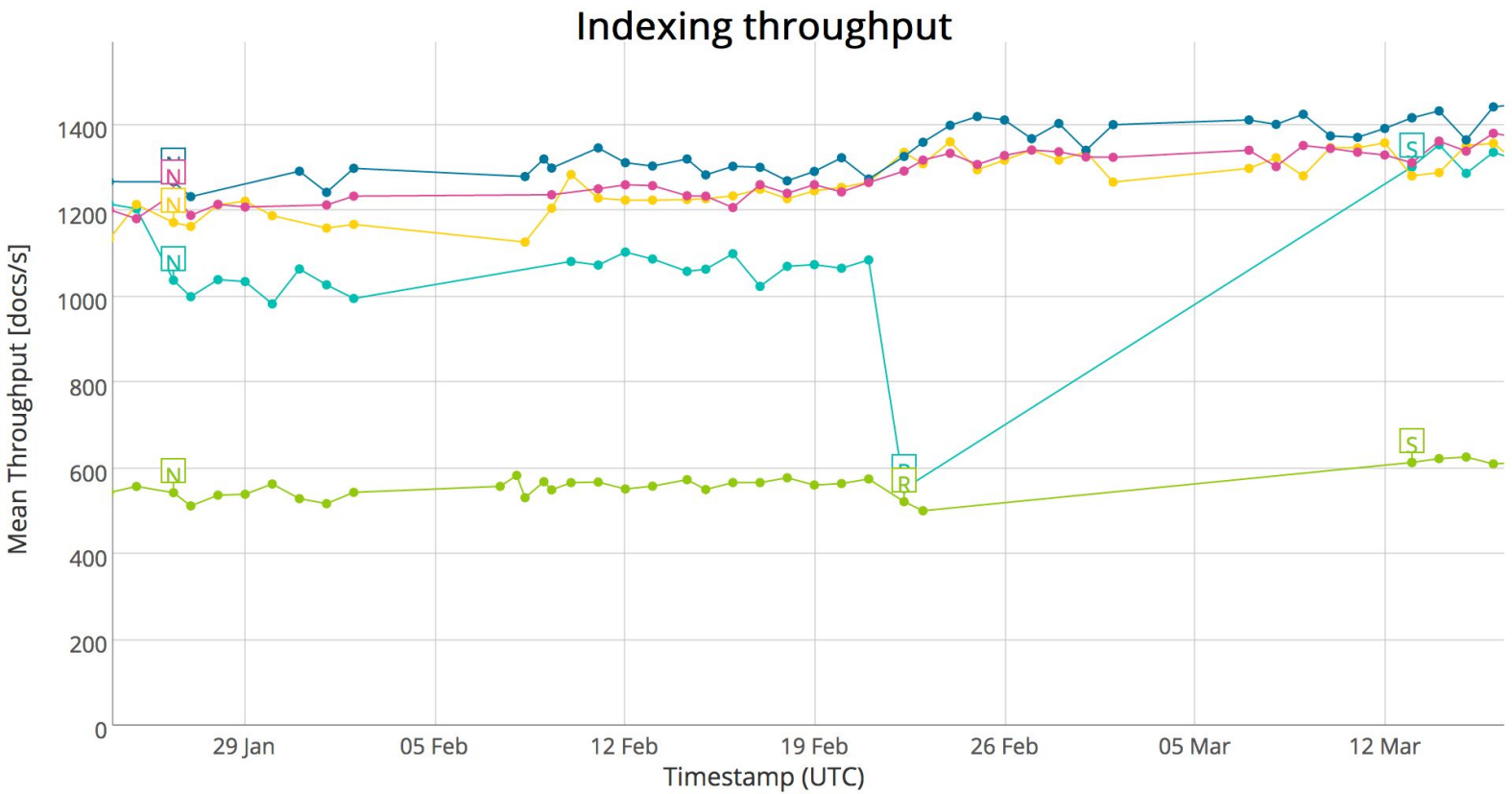
elastic


Indexing throughput
Mean Throughput [docs/s]
65000
60000
55000
50000
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
N
R
29 Jan
05 Feb
12 Feb
19 Feb
26 Feb
05 Mar
12 Mar
Timestamp (UTC)

elastic


GC times
Total GC runtime [s]
350
300
250
200
150
100
50
0
N
R
N
R
29 Jan
05 Feb
12 Feb
19 Feb
26 Feb
05 Mar
12 Mar
Timestamp (UTC)

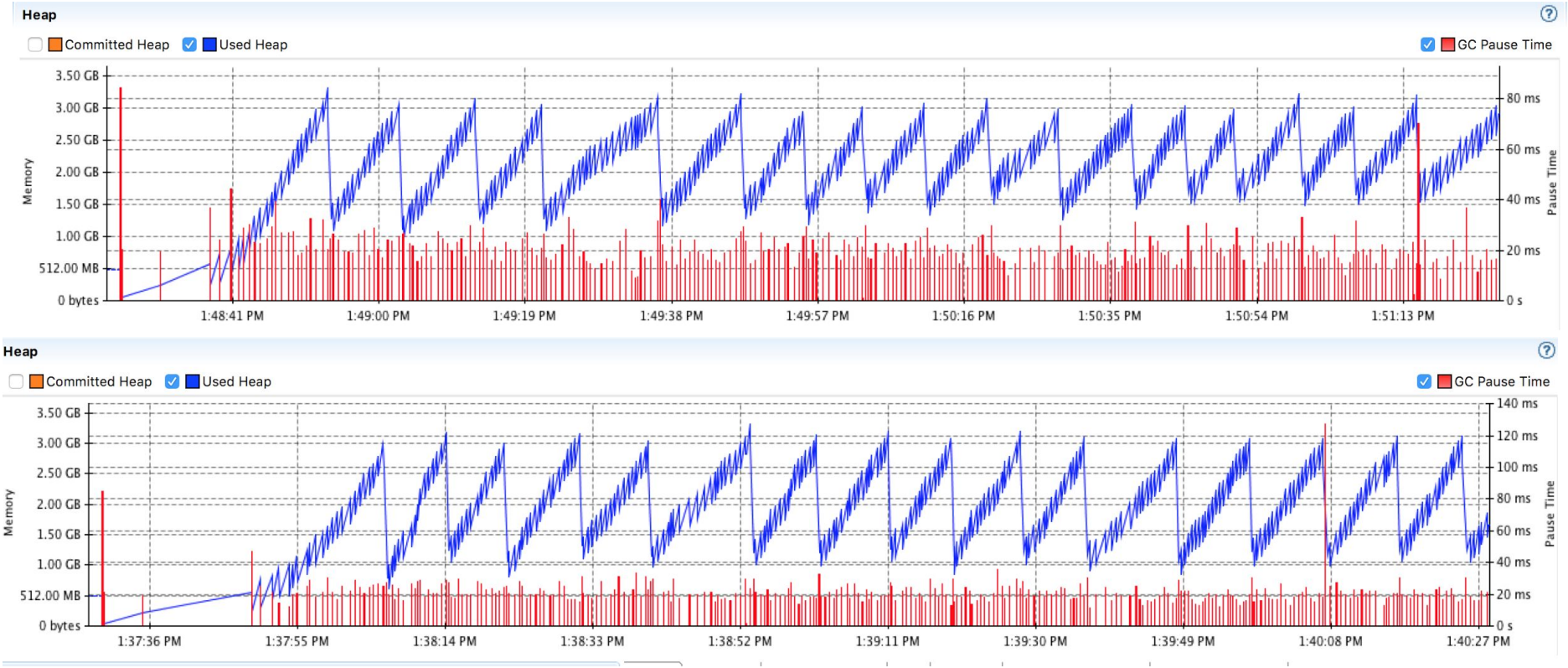
elastic



2017/03/13 00:00:51:
Append / default settings: 1303.08
Append / 4G heap: 1282.17
Append Fast / 4G heap: 1417.41
Id Conflicts / 4G heap: 1312.57
Append / 2 nodes: 613.96

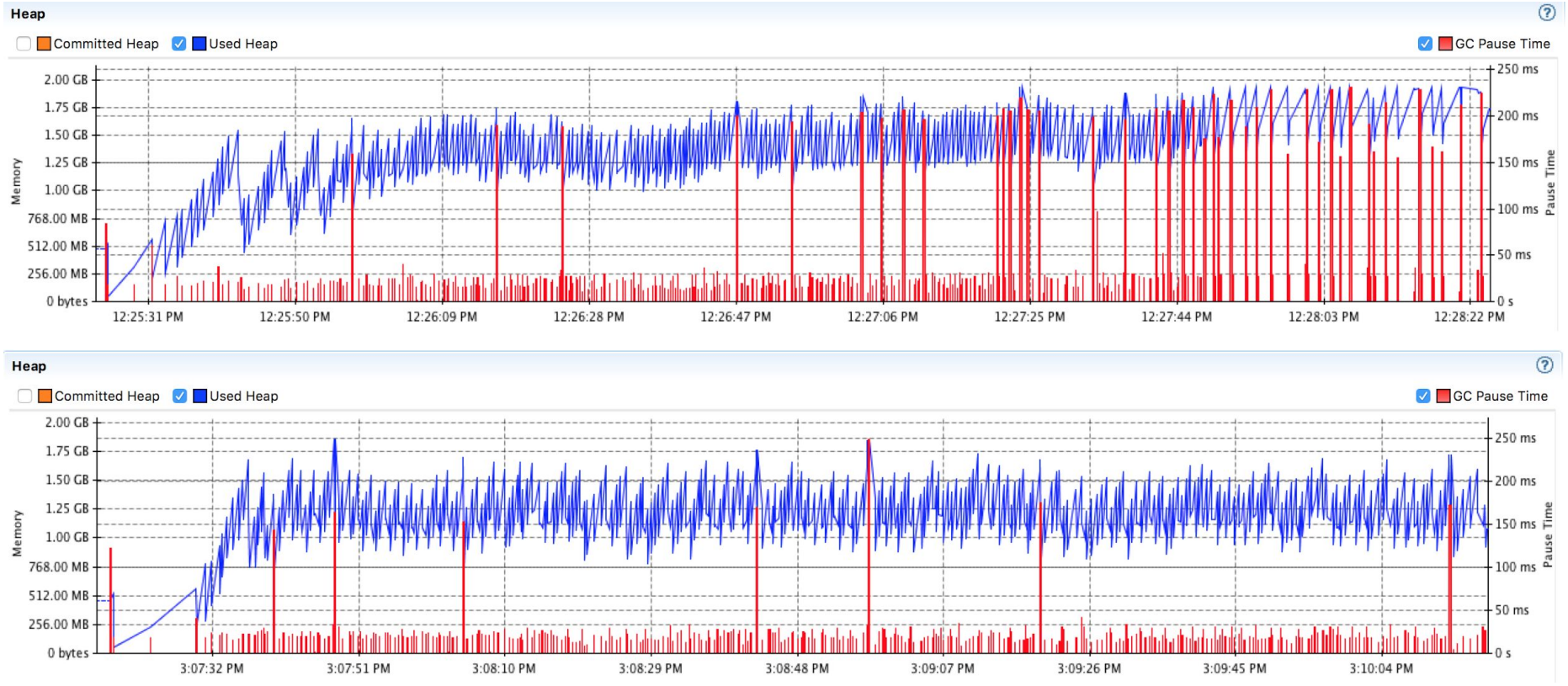
elastic

4G Heap with 32k (top) and 64k (bottom)



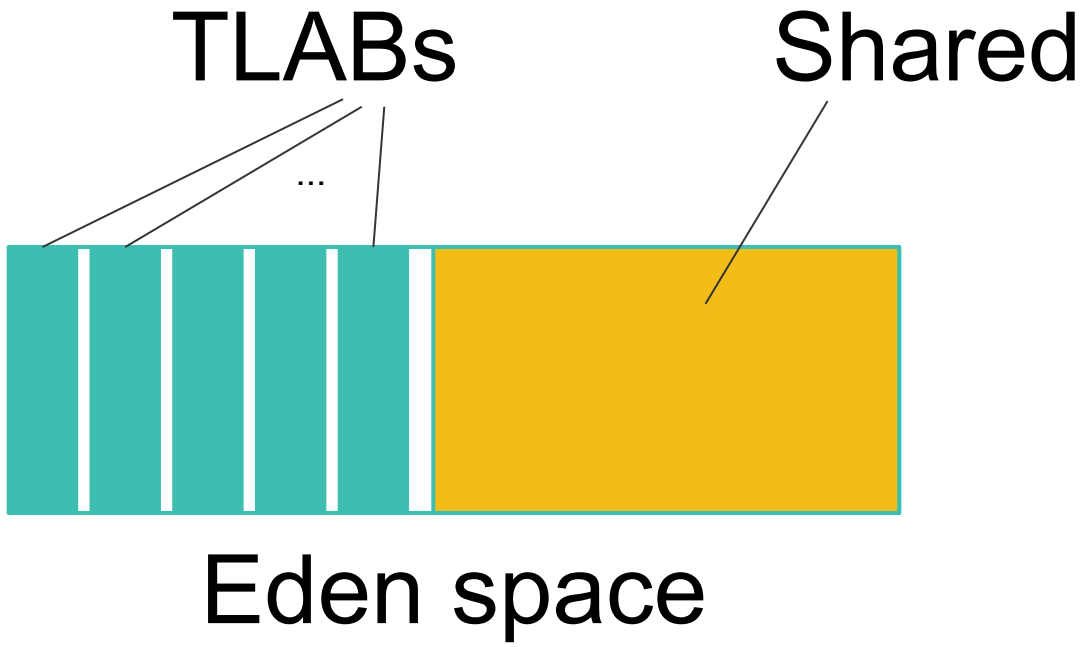
elastic

2G Heap with 32k (top) and 64k (bottom)



elastic

Thread Local Allocation Buffers (TLABs)



Allocation Numbers (2GB heap)

Receive predictor size [kB]	Number of TLABs	Allocations outside TLAB
32	238,169*	**36,108,455***
64	689,016	726,691

* until OOMEd

Ongoing investigation

https://github.com/elastic/elasticsearch/issues/23185

- Question: Why do fill up TLABs more quickly with smaller buffers?
- Hypothesis: We need to allocate more buffers but do not use them efficiently

Takeaways

Follow Up Talk

“Java Flight Recorder: The hidden arrow in your quiver”

May 10, Lightweight Java User Group

Image Credits

- Shuttle Cockpit by Robert Young (license: CC BY-NC 2.0)
- Down the rabbit hole! by Nullfy from nullfy.com! (license: CC BY-NC 2.0)