

## Benchmarking Elasticsearch with Rally

Daniel Mitterdorfer @dmitterd

# "Elasticsearch is just a search engine, isn't it?"









0 34 "-" "Mogilla/5.0 ( 2.0.0.3 (Ubuntu-feisty)" HTTP/1.1" 200 11514 "-" #61201 Firefox/2.0.0.3 (Ub

TTP/1.1" 500 617 "http:/ t) Gecko/20061201 Fir

> "Hozilla/5.0 htu-feisty)" "Mozilla/5.1 htu-feisty) 1" 200 382 (ko/200612

> > W: Linux

# How do you evaluate performance for all these use-cases?



#### **During Development**





#### **During Development**

0	Oracle Java Mission Control					
2016040						
	C Allocations					
	E Exerts B Operative Interval: 9 min 51 a (art)			<ul> <li>Synchroniae Sele</li> </ul>		
5	ومعاصداته ويخاله استجهامها والتزانا أبدارا الأبادا تتنتكا ومعملتهم وأورودها وسيعقب	which the build	محبب والملالح ومحبوه	فيسادان		
General	47/1680415.8M			40/168140		
E.						
1	General Allecation Ir New TLAB Allecation Outside TLABs					
Memory	Allocation by Case Allocation by Thread Allocation Profile					
<b>ED</b>	Allecation Profile					
2	Stack Trace	Average Object Size	Total TLAB size	Pressource		
	* % java.lang.Thread.runib	13.75 40	899.26.00	78,034		
Code	* % jansuti.concurrent.TveedPoolSeculorSillorker.run()	13.76 kB	899.25.08	78,2375		
-	* * java.util.concurrent.ThreadPoolExecutor.rumWorker(ThreadPoolExecutorEWorker)	13.76 kB	899.25.08	78,025		
	* * ang alasti saansh samman uli sansumeri Alastrasi Burnalia nun()	17.17.60	815.01 GB	21.615		
1.0	* > org-elasticsearch.common.util.concurrent.ThreadContextEContextEContextEcontextE	17,57.48	815.91.08	21.61%		
	Y % org.elasticsearch/transport/Transport/Service/S4.doBum)	17.19 48	815.25 GB	71.556		
<b>Weeds</b>	* & org.diabfictearch.transport.RecurstRiandierRegistry.process/Recsived/TransportRecurst, Transport.	17.59 kB	815.25.08	71.55%		
1772	* % org elasticasarch.action.aupport.replication.franaportReplicationActionShrimaryOperationfhanaportHand	17.26 kB	815.19 GB	\$1.55%		
18.	* sing electrosechuction support replication.TransportReplicationActionBhilmaryOperationTransportHa	17.26 kB	815.19.08	3rt.66%		
-	* b organizationsanch common of Loonourset Abstract/furnable.run()	17.26 48	813.19.08	21.55%		
	* % org-elasticsearch-action-support-replication.TransportReplicationAction8hr/marp/hase.dofunO	17.26 kB	815.19.08	21.54K		
10	Y % orgatizationarchaetion.aupport.replication.TransportReplicationActionStrimaryPhase.auxout	17.26 48	815.19 GB	21.555		
070	* % org.statticsearch.action.buik.TransportShareBuikAction.shareBperationOnthinary(MetaBa	17.27 kB	814.75.00	21.51%		
	Y Sing elasticsearch action built fransportShardBuiltAction ahardDpentionOnPrimary(Met	17.27.48	814.75 GB	P1.515		
100	* * org-elasticssarch.action.bulk.TransportShardBulk.Action.handlehendMetaDeta, BulkS	17.27 kB	814.46.00	3-1.484		
	* % organiseficieserch.action.bulk?transportChardbulkAction.index(MetaDeta, BulkCha	17.27 kB	814.46 GB	21.45%		
PASSAGE.	* % org.easticsearch.action.buik.TransportShareBuikAction.shardhoexOperationL	17.28 kB	813.96.08	31.64%		
1	Y % org alatficeanth action.index.Transportindex.Action.assoulatindexTeepantO	17.28 18	813.06 GB	71.64%		
Sec. 2.	Y % organisatioseanch.index.shard.index.dhard.index	23.98 kB	593.46.08	62.09%		
1	It is use also index also also also also also also also also	23.06 16	593.45 GB	53.09%		
	In the organization of the second section index. Transport index Rotion, prepare index Operat	69 bytes	219.89.00	19.50%		
Columba.	> org.stasticsearch.action.aupport.replication.TransportReplicationStrine	24 bytes	225.66 MB	0.03%		
	> > org-alasticsearch action support, replication. TransportReplicationAction processAfter	162 bytes	69.45 MB	0.01%		
	> segularitiesenth action advin indices fuch TransportDrantTrahAction shardOperationOn.	13.13 HD	3/70.3/7 MB	0.03%		



#### Nightly benchmarks



Sizing benchmarks for specific scenarios\*





10 \*) numbers on axis intentionally stripped to avoid interpretation out of context

Performance measurement / tuning at customer site





## How we measure





#### **Rally** You know ... for benchmarking Elasticsearch

### https://github.com/elastic/rally



#### 10.000 feet view of Rally





### Demo



### 7 Deadly Benchmark Sins



#### Sin #1: Not paying attention to system setup Hardware

- Bare-metal
- SSDs
- Server-class CPU
- Single socket, multi socket?
- Enough memory head-room for FS cache



#### Sin #1: Not paying attention to system setup

**Operating System and JVM** 

- Linux, Windows
- No Swap
- Check network configuration
- Think about file system, LVM, etc.
- I/O scheduler: cfq, noop, deadline
- CPU governor: powersave, performance
- JVM version



#### Sin #1: Not paying attention to system setup Benchmark Setup

- Beware of unwanted caching effects (FS cache, ...)
- Benchmark driver and ES on separate machines
- One node per machine (or adjust JVM parameters (GC threads))
- Low-latency, high-throughput network between benchmark driver and ES
- No traffic on this network



#### Sin #2: No warmup

Awake before your first coffee? Elasticsearch isn't either.

- JIT compiler needs to run first
- Creation of long-living data structures
- FS cache for Lucene segments (memory-mapped IO)
- Benchmark driver needs to reach stable state too



#### Warmup Behaviour: C2 Compilation Events/s





#### Warmup Behaviour: Benchmark Driver Throughput





#### Sin #3: No bottleneck analysis

Are you really benchmarking what you think you're benchmarking?

- Benchmark driver
- System setup: analysis of system background noise (jhiccup)
- Network



#### **First Driver Stress Tests**

Contention all over the place





#### Sin #4: The divine benchmarking script

"After all, it produces numbers with 6 decimal places!"

- Not paying attention how metrics are gathered
  - System.currentTimeMillis() VS. System.nanoTime()
- Not checking measurement overhead
- No return code checks: the fast 404
- Blind trust in tools: No cross-verification



#### **Cross-Validation of Metrics**

Metric	Rally	Flight Recorder	GC log
Young Gen GC	79,416 ms	89,003 ms(?)	80,853 ms
Old Gen GC	23,964 ms	156,630 ms(?)	23,989 ms



#### Sin #5: Denying Statistics

#### Run-to-run variance

- How is run-to-run variance distributed?
- Multiple trial runs and t-test



#### **Run-to-run Variance Verification**

Lat Indexing Throughput Distribution





#### Sin #5: Denying Statistics

#### Latency Measurement

- The meaningless mean: Half of the responses are worse than the mean
- Cannot calculate 99.99th percentile from 10 samples
- Don't average percentiles
- Latency distribution is multi-modal







Service Time





Service Time

```
while (!isDone()) {
    long start = System.nanoTime();
    // block until the request has finished
    send(createRequest());
    long end = System.nanoTime();
    long serviceTime = end - start;
}
```



Latency

queuing effects 3 User issues request Request processing Response arrived



Latency: Include wait time

```
// generator thread
while (!isDoneGenerating()) {
    long start = System.nanoTime();
    queue.put(createRequest(), start);
}
```

```
// request issuing thread
while (!isDoneSending()) {
    request, start = queue.take();
    send(request);
    long end = System.nanoTime();
    long latency = end - start;
}
```



Throughput: System at t = n seconds





Throughput: System at t = n + 1 second



😪 elastic





**Resulting Throughput** 

# 5 ops/s



Latency ... at which throughput?





40 Source: http://robharrop.github.io/maths/performance/2016/02/20/service-latency-and-utilisation.html

Latency at a defined throughput

```
// generator thread
while (!isDoneGenerating()) {
    long start = System.nanoTime();
    queue.put(createRequest(), start);
    Thread.sleep(waitTime(targetThroughput));
}
```

```
// request issuing thread
while (!isDoneSending()) {
    request, start = queue.take();
    send(request);
    long end = System.nanoTime();
    long latency = end - start;
}
```



#### Sin #7: Treat Performance as One-Dimensional Vary inputs

- Bulk size
- Query parameters
- Document structure



#### Sin #7: Treat Performance as One-Dimensional Vary execution order

- Run queries in different order: Avoid caching effects
- Interfere operations: How does indexing behave with concurrent queries?



# Sin #7: Treat Performance as One-Dimensional

#### And more

- Hardware
- OS
- JDK
- •



# Summary







# Performance is easy, all you need to know is everything

Sergey Kuksenko, Oracle Performance Engineer

## **Questions?**



#### Slides

https://bit.ly/rally-muc



#### **Further Resources**

- "How not to measure latency": <u>http://www.youtube.com/watch?</u>
   <u>v=lJ8ydluPFeU</u>
- "Benchmarking Blunders and Things That Go Bump in the Night": <u>http://arxiv.org/pdf/cs/0404043v1.pdf</u>



#### **Image Sources**

- "Searching" by "Skyblueseed": <u>https://www.flickr.com/photos/skyblueseed/</u> <u>5792500545/</u> (CC BY-NC-ND 2.0)
- "Curiosity Mastcam L sol 673" by "2di7 & titanio44": <u>https://www.flickr.com/photos/lunexit/14570422596/</u> (CC BY-NC-ND 2.0)
- "80's style Hacker Picture" by "Brian Klug": <u>https://www.flickr.com/photos/</u> <u>brianklug/6870005158/</u> (CC BY-NC 2.0)
- "Works Mini Cooper S DJB 93B" by "Andrew Basterfield": <u>https://</u> www.flickr.com/photos/andrewbasterfield/4759364589/ (CC BY-SA 2.0)



#### **Image Sources**

- "photo" by "Odi Kosmatos": <u>https://www.flickr.com/photos/kosmatos/</u> <u>8162850619/</u> (CC BY 2.0)
- "gags9999": <u>https://www.flickr.com/photos/gags9999/14124313715/</u> (CC BY 2.0)
- "Bachelor Students Chemistry Lab" by "NTNU": <u>https://www.flickr.com/photos/92416586@N05/12188423293/</u> (CC BY 2.0)
- "42" by "Lisa Risager": <u>https://www.flickr.com/photos/risager/5067595483/</u> (CC BY-SA 2.0)

