Maintaining performance in distributed systems

Alexander Reelsen
@spinscale
alexander.reelsen@elasticsearch.com

Agenda

- Distributed Systems
- Elasticsearch
- Performance aspects

Hardware & Operating System

JVM & GC

Libraries & Application

About...

Me

Software Engineer at Elasticsearch

Interested in all things search & scale

Search Meetup Munich

http://www.meetup.com/Search-Meetup-Munich/events/218856224/

Elasticsearch

Founded in 2012

Products: Elasticsearch, Logstash, Kibana, elasticsearch for Apache Hadoop, Marvel, Shield

Professional Services: Support subscriptions, trainings

Distributed systems

Distributed systems

- Fallacies of distributed computing
  1. The network is reliable.
  2. Latency is zero.
  3. Bandwidth is infinite.
  4. The network is secure.
  5. Topology doesn't change.
  6. There is one administrator.
  7. Transport cost is zero.
  8. The network is homogeneous.

by Peter Deutsch

https://en.wikipedia.org/wiki/Fallacies_of_distributed_computing

Distributed systems expectations

- Redundancy
- Resiliency
- Recovery
- Scalability
- Availability
- ...

Example...

- Cope with node outage

  Maintenance, network split, power loss, garbage collection

Example: Outages

- Cope with node outage

  Maintenance, network split, power loss, garbage collection

- Still operational
- No data loss
- CRUD works

Example: Recovery

- Nodes come back

  Maintenance, network failure ends...

- Self healing
- Shift data back
- Higher load

Example: Recovery

- Nodes come back

  Maintenance, network failure ends...

Example: Recovery

- Nodes come back

  Maintenance, network failure ends...

Example: Data distribution

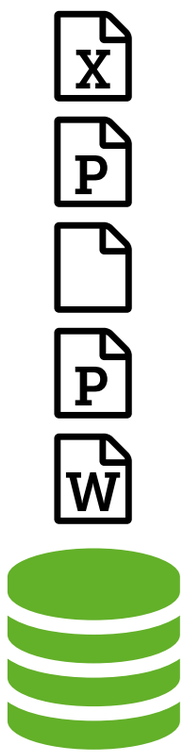
- Scalability

  Writes vs. reads

Example: Data distribution

- Scalability

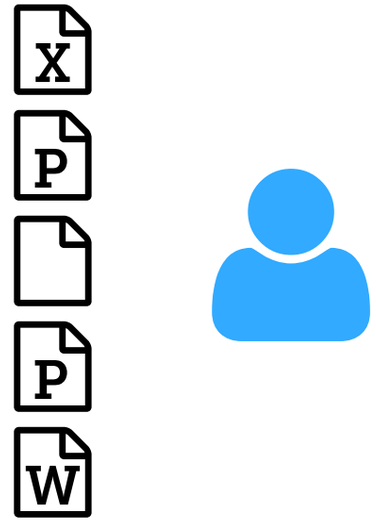
  Writes vs. reads



Example: Data distribution

- Scalability

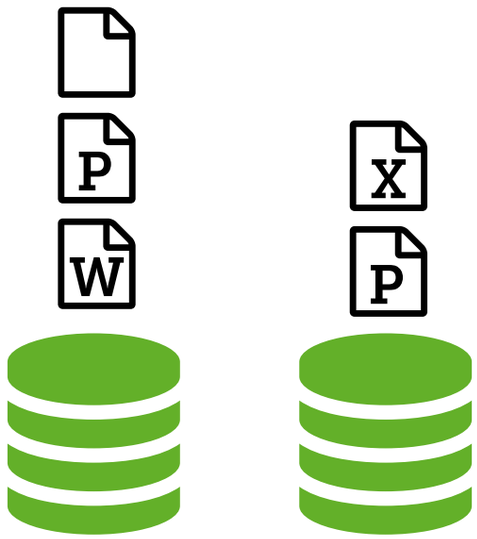
  Writes vs. reads



Example: Data distribution

- Scalability

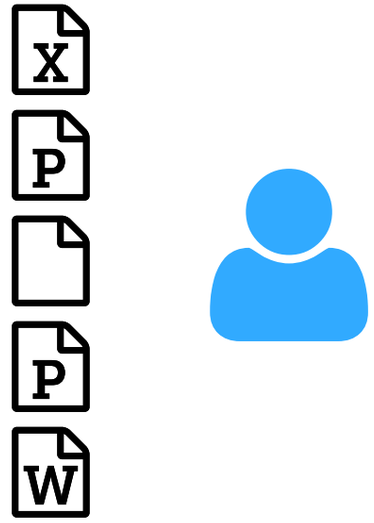
  Writes vs. reads



Example: Data distribution

- Scalability

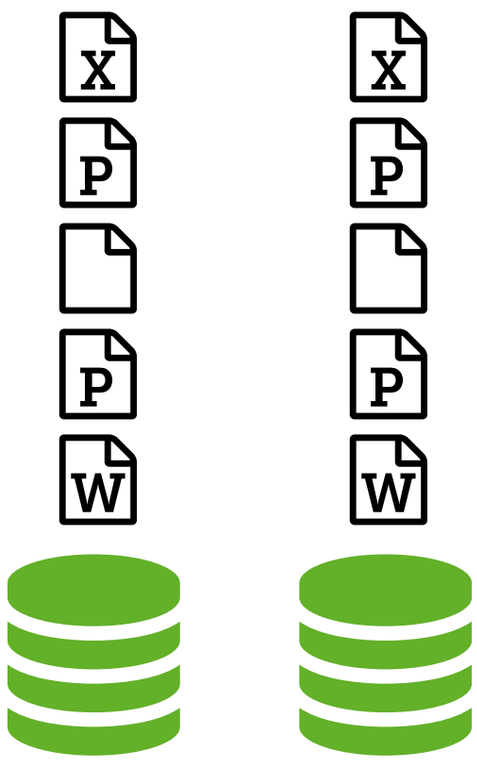
  Writes vs. reads



Example: Data distribution

- Scalability

  Writes vs. reads

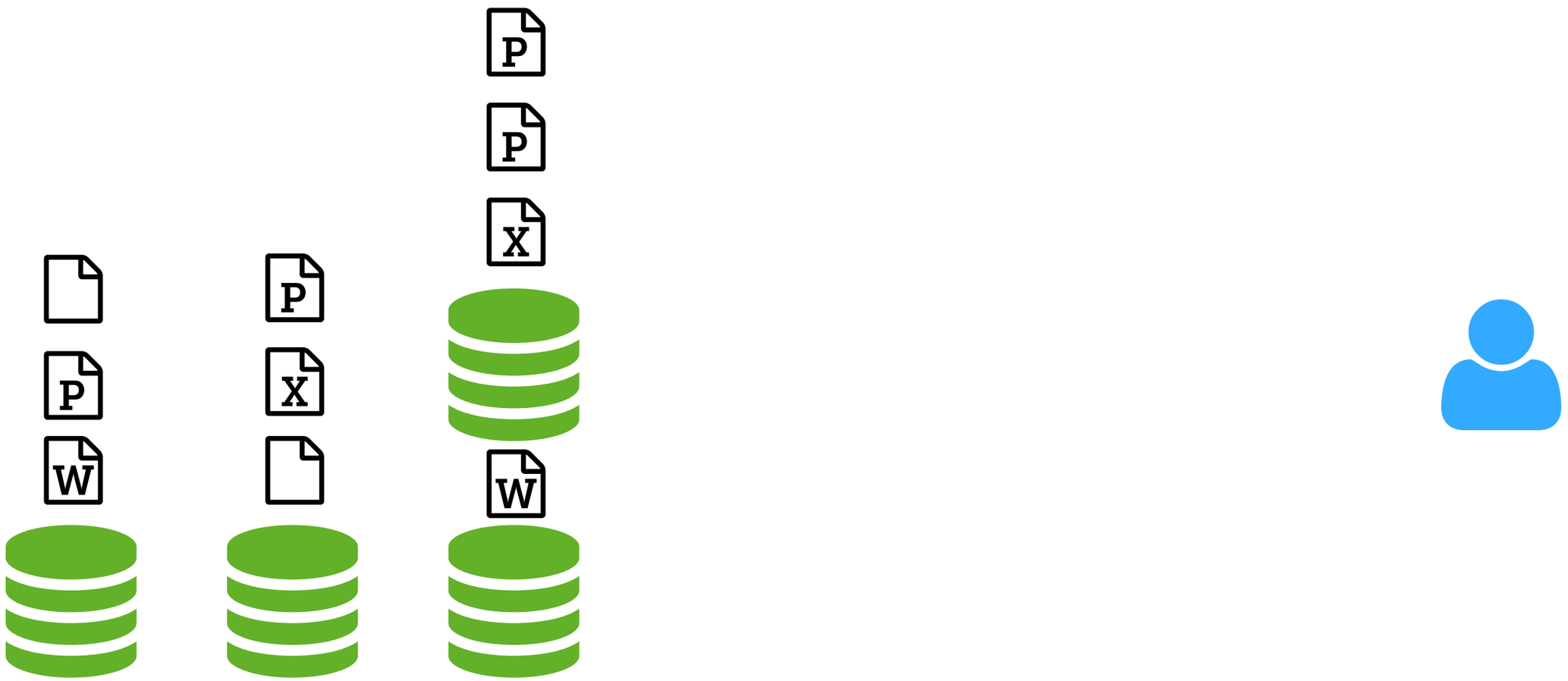


Example: Data distribution

- Read & write scalability

Example: Data distribution

- Read & write scalability



But... performance?

- ...is affected by this

  More Coordination, more distant

  Different boundaries compared to single process applications

  hard to predict/test

- on many different layers

- Application & Libraries
- Runtime environment (JVM)
- OS
- Hardware
- Network

Elasticsearch

Introduction

What is Elasticsearch?

- HTTP & JSON

What is Elasticsearch?

- HTTP & JSON
- Schema-less

What is Elasticsearch?

- HTTP & JSON
- Schema-less
- distributed

What is Elasticsearch?

- HTTP & JSON
- Schema-less
- distributed
- document-oriented

What is Elasticsearch?

- HTTP & JSON
- Schema-less
- distributed
- document-oriented
- near-realtime

What is Elasticsearch?

- HTTP & JSON
- Schema-less
- distributed
- document-oriented
- near-realtime
- search

What is Elasticsearch?

- HTTP & JSON
- Schema-less
- distributed
- document-oriented
- near-realtime
- search
- analytics

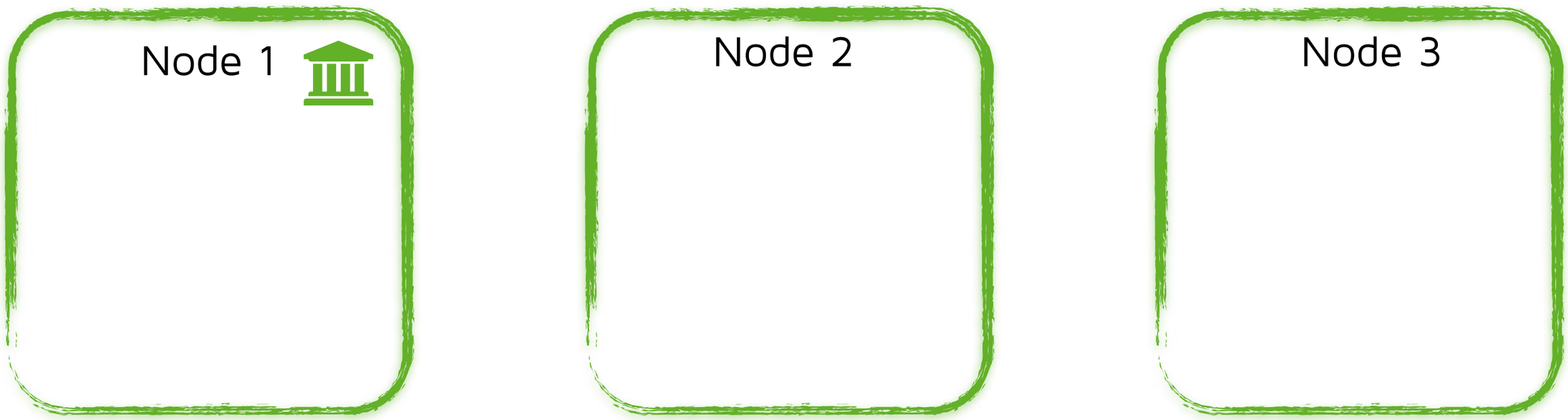
Master

- Cluster always has one master
- Reelection on node failure

Node joins

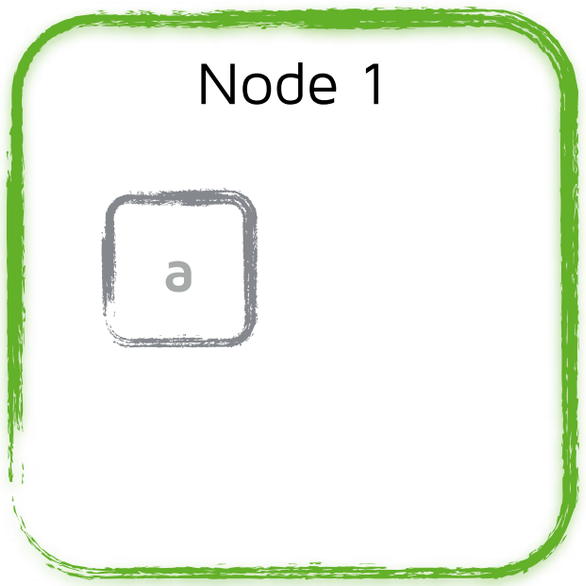
- Node 2 & Node 3 ping around

Node joins



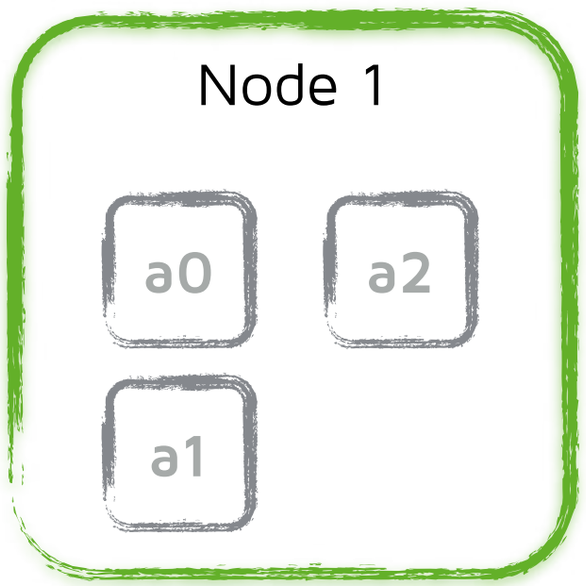
- Node 2 & Node 3 join cluster

Data distribution



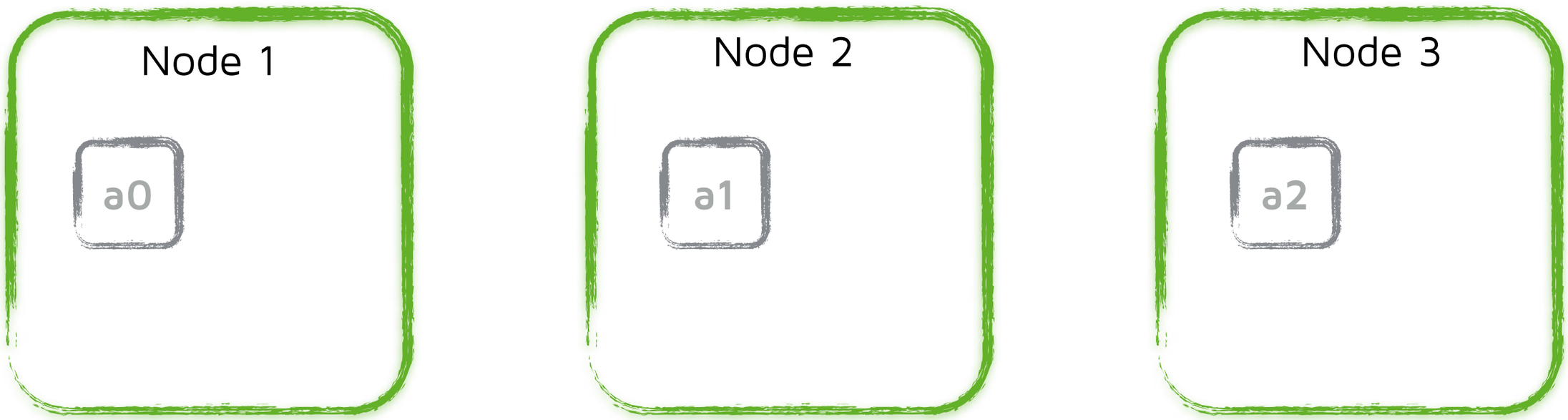
- Index: Collection of documents

Data distribution



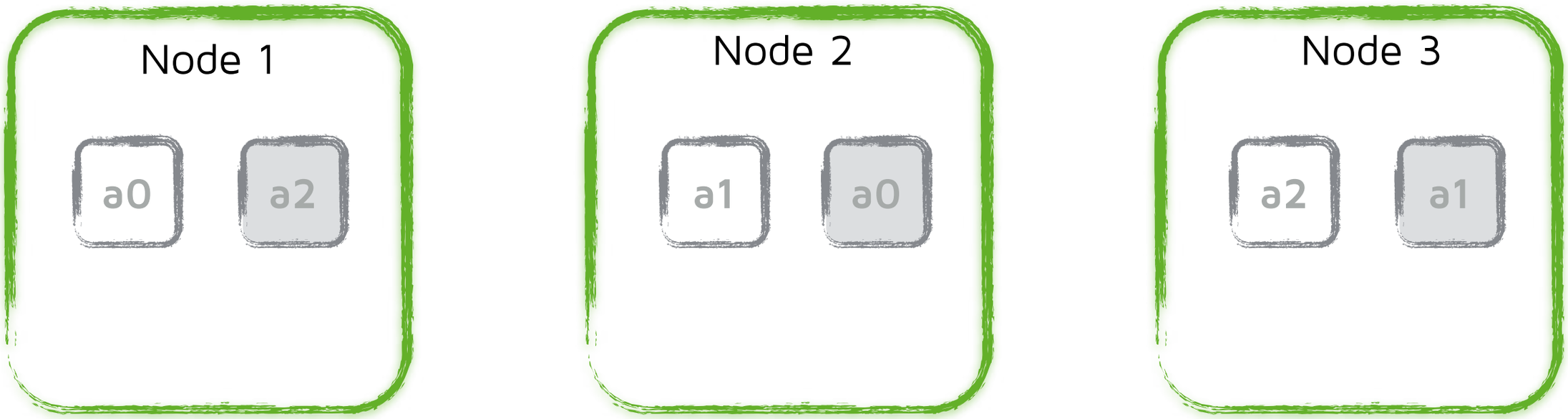
- Shards: Units of scale

Data distribution



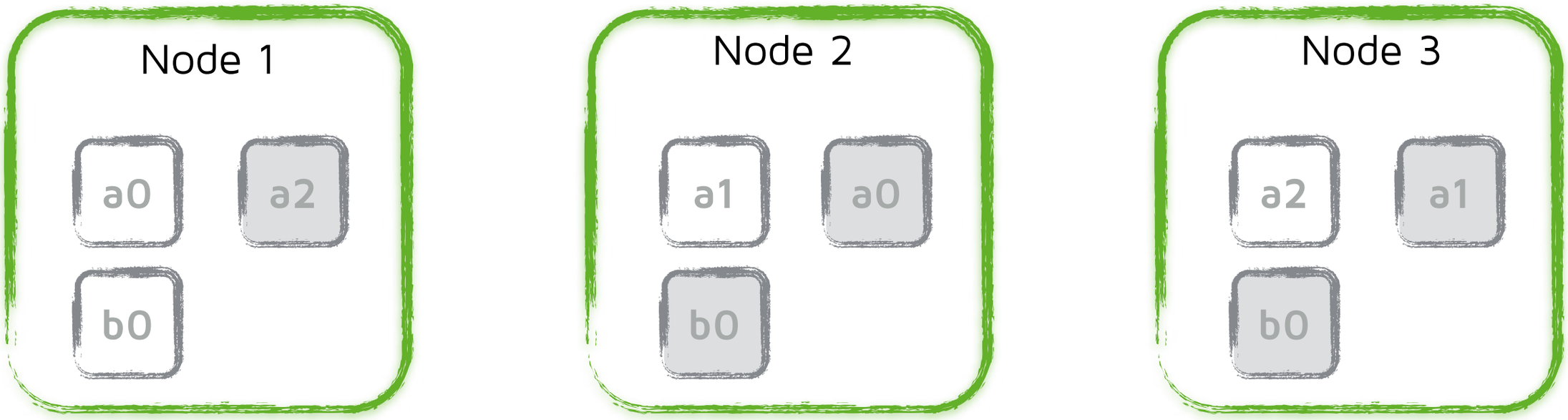
- Shards: Units of scale

Data distribution



- Primary shards
- Replica shards

Data distribution



- Different scaling strategies per index

Elasticsearch can easily max out...

- CPU

  Indexing, searching, highlighting

- I/O

  Indexing, searching, merging

- Memory

  Aggregations, indices

- Network

  Relocation, snapshot & restore

But... performance?

- Competing resources
- Resizing is out of our control
- Requires thorough testing & configuration

Hardware & operating system

Quiz

- What is locked memory?
- What is the best scheduler for SSDs?
- Is TRIM supported on every FS?
- What is mechanical sympathy?

Hardware

- Bigger is better? It depends...
- CPU: # cores, more parallel threads
- Main memory: No limit
- Disk: SAN vs. local, SSD vs. spindle
- Bare metal vs. virtualization

  https://speakerdeck.com/elasticsearch/life-after-ec2

SSDs are awesome

- TRIM
- Write amplification
- Garbage collection
- Coding for SSDs

  http://codecapsule.com/2014/02/12/coding-for-ssds-part-1-introduction-and-table-of-contents/

Operating systems

- File system descriptors, file system cache
- Memlocked memory

  `bootstrap.mlockall: true`
- NUMA

  http://engineering.linkedin.com/performance/optimizing-linux-memory-management-low-latency-high-throughput-databases

  http://queue.acm.org/detail.cfm?id=2513149
- Don't swap out if you need performance!
- OOM killer: Just dont...

JVM

Quiz

- When does the JIT compiler kick in?
- Are client/server JVMs different?
- What's the default thread stack size?
- Is there a memory based thread limit?

JVM tricks

- Less than 32 GB of heap, allowing to use compressed pointers
- Serialize everything yourself

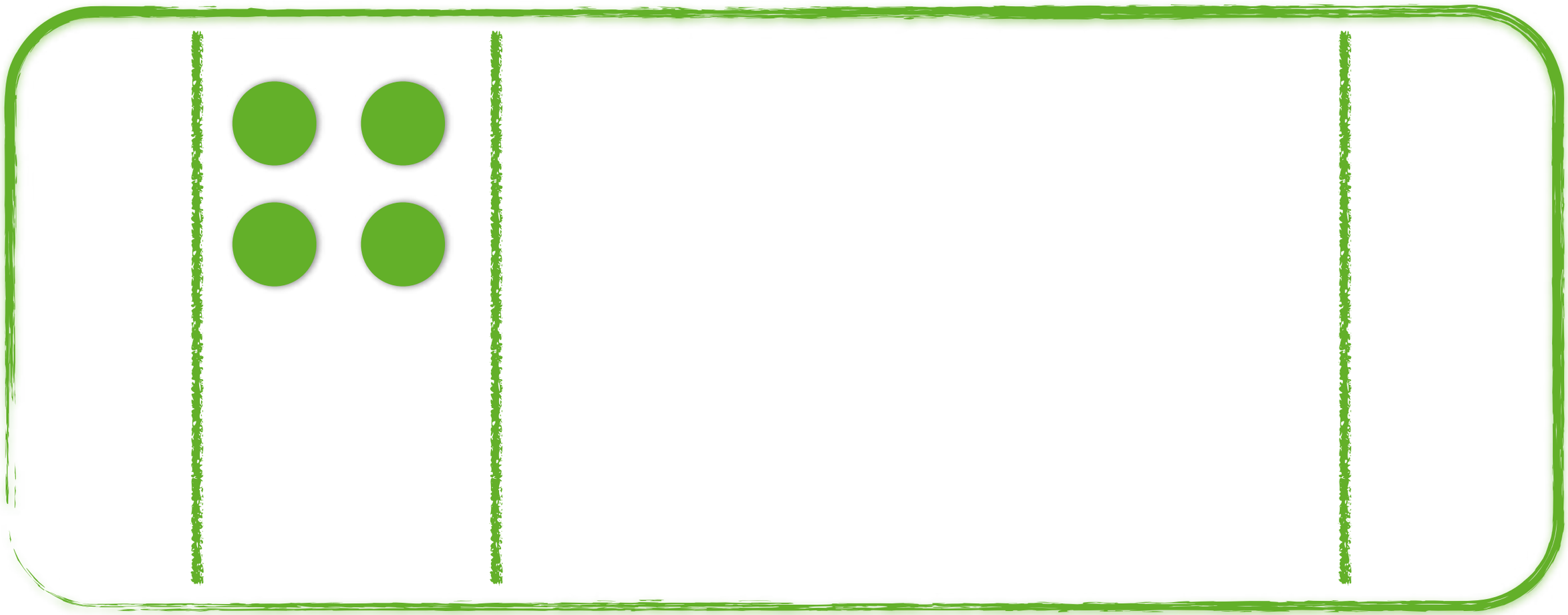
  JVM versions tend to be incompatible
- use server vm, allocate all memory on startup
- reduce thread stack size

  http://rdiyewar-tech.blogspot.de/2013/02/outofmemoryerror-because-of-default.html

JVM Threads

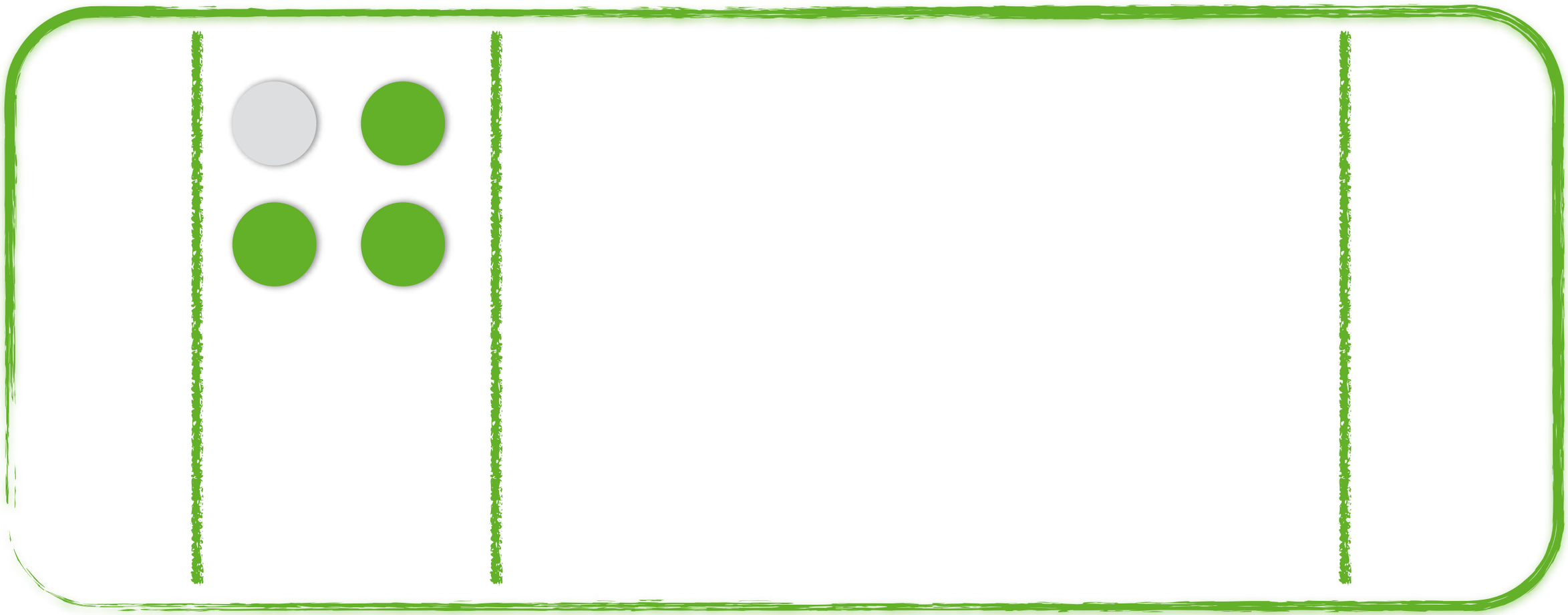
- JVM is good at managing threads, but not several thousands of them
- Single thread pool does not fit all
- Solution: Dedicated thread pools, based on the amount of available CPUs and their task complexity

JVM Garbage collection



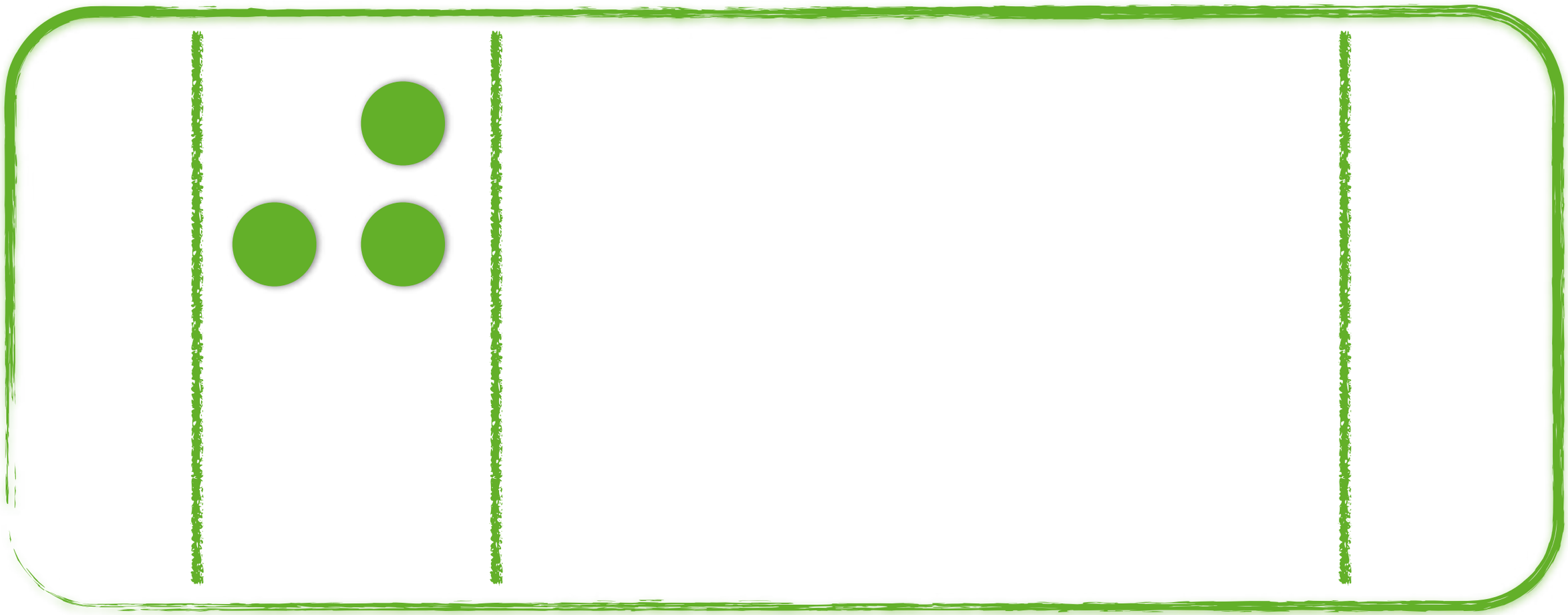
direct young old perm

JVM Garbage collection



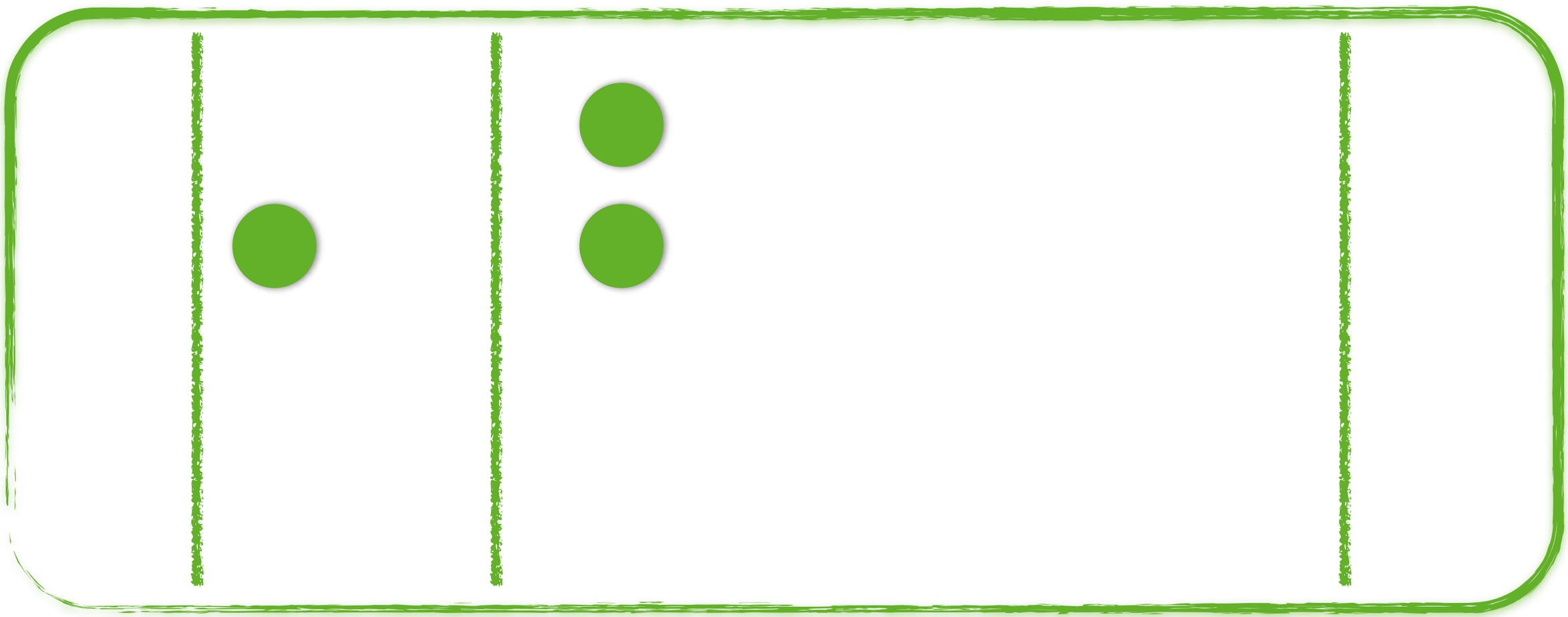
direct young old perm

JVM Garbage collection



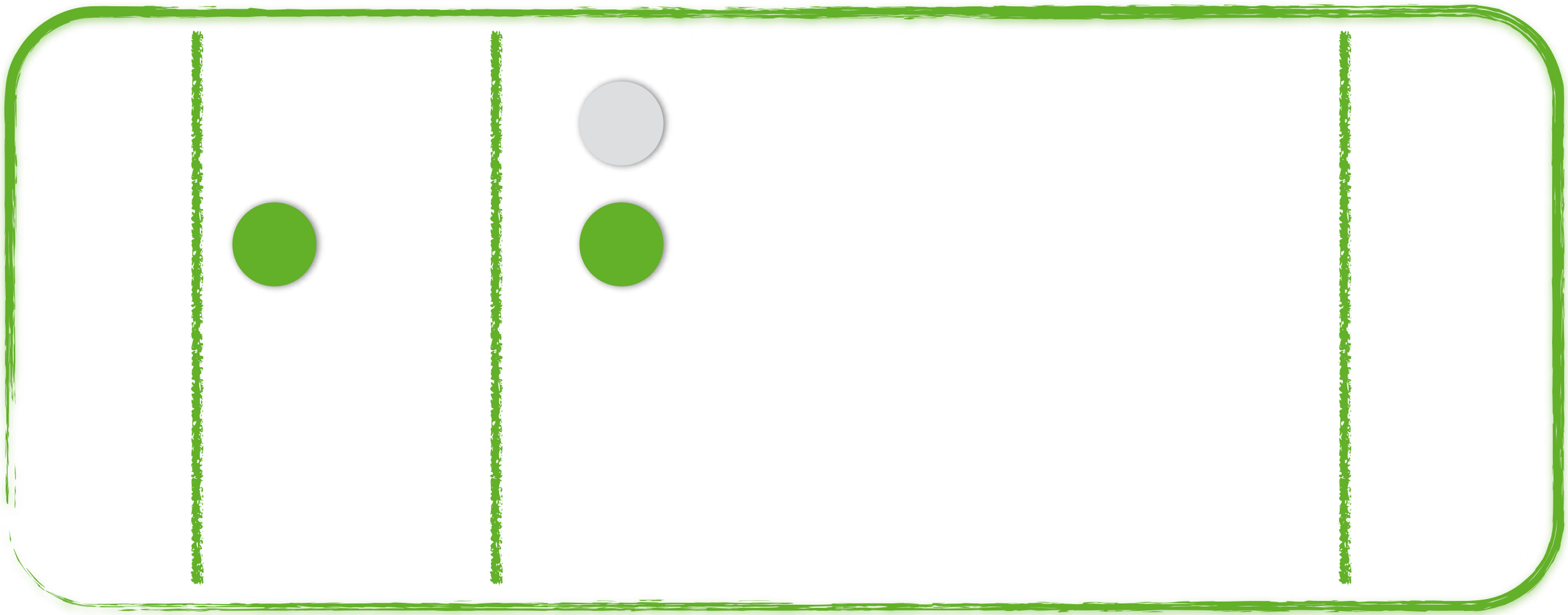
direct young old perm

JVM Garbage collection



direct young old perm

JVM Garbage collection



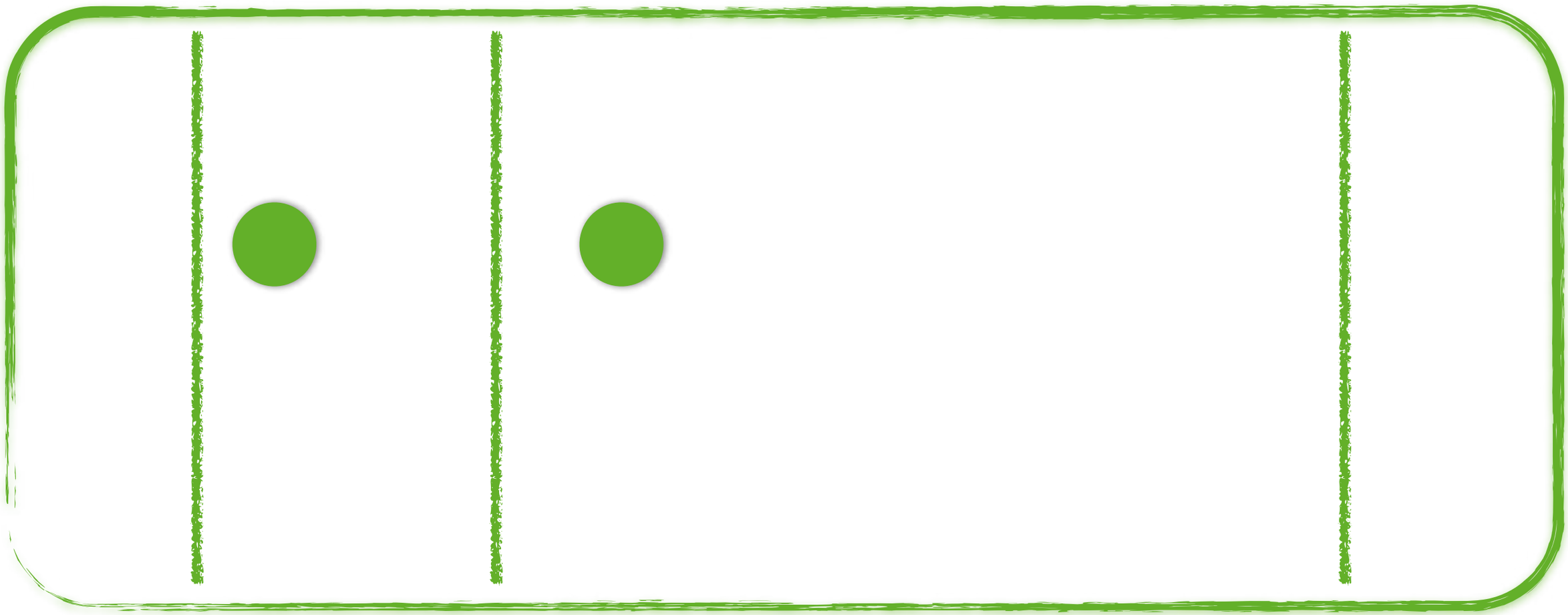
direct young old perm

JVM Garbage collection

stop the world

direct young old perm

JVM Garbage collection



direct young old perm

Improving garbage collection

- Create less objects, reuse structures
- Stream data in to avoid object creation

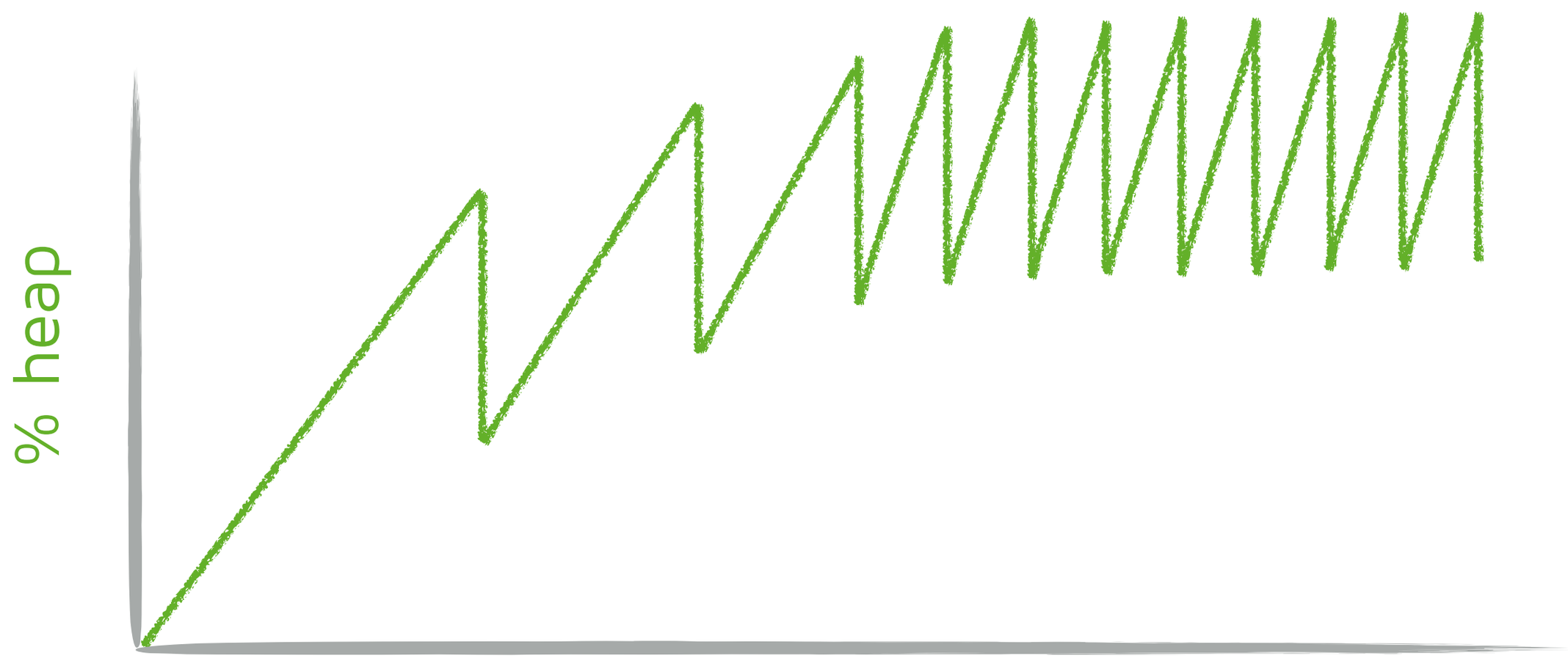
  reduces young gen promoting pressure
- `-XX:CMSInitiatingOccupancyFraction=75`
- Elasticsearch:

  Long GCs can result in nodes dropping out of the cluster and master reelections and data shifting (often happens due to GC pressure)

Garbage collectors

- Serial, Parallel, ParallelOld
- CMS - Concurrent mark-and-sweep
- G1
- Pauseless GC (Shenandoah, Azul)
- Going off-heap
- Using `java.misc.Unsafe` & handle memory allocation yourself

GC spiral of death



Libraries

Guice

- dependency injection container
- allows to create infrastructure for plugins
- singletons can be created eager (on startup)

HPPC

- First, Guava is awesome

  But can create a lot of objects due to immutability concept

- Meet High Performance Primitive Collections

  http://labs.carrotsearch.com/hppc.html

- Mapping updates (used `ImmutableOpenMap`)

- Now uses `ObjectObjectOpenHashMap`

  1000 properties: 0.2 seconds (was 5.1)

  2000 properties: 1.2 seconds (was 25)

  5000 properties: 4.2 seconds (was 231)

  10000 properties: 83.8 seconds (never finished before)

Lucene

- Writes are append-only (segments are immutable)
  - Allows the file system cache to kick in for huge segments
  - Lock-free read access
- Rate limiting on write
  - Saves IO and CPU
- Packed* classes, ordinals

Lucene

- Piggyback on Lucene segment lifecycle
  - Filter caching per segment
  - Field data caching per segment
- FSTs
  - Blazing fast in-memory structures, allow thousands of qps
  - Allow for complex searches like prefix/fuzzy searches or intersections

Sigar

- Awesome monitoring API
- Great helping library for getting all kinds of stats
- Output can vary on operating systems

jackson

- Stable and fast streaming JSON parser
- Supports YAML, SMILE, CBOR
- Other implementations

  https://github.com/RichardHightower/boon/wiki

Elasticsearch

Going async

- Enforces event driven architecture
- Support for non-blocking model
- Enforce loose coupling
- Prefers push over pull
- Callback based concurrency
- Helps to avoid contention on resources / threads

Reduce memory footprint

- Page-based cache recycling (old gen!)
- Reusing netty buffers
- Fielddata
- Probalistic data structures

  Bloom filters, T-Digest, HyperLogLog++

Node & network communication

- Maintaining different channels with different priorities

  IMMEDIATE, URGENT, HIGH, NORMAL, LOW, LANGUID

- Binary protocol

- TCP connections are held open

Prevent OOM

- Fielddata is number one OOM reason
- Circuit breaker

  per request & fielddata
- Doc-value based field data

Conducting performance tests

Performance test requirements

- Good Data & queries

  real life data

- Similar environment

  Virtualization, bare-metal, AWS, number of nodes

- Long running tests

  Avoid hitting the wrong caches and missing the right ones

- Rate limit the right things/things right

- Create your own benchmark numbers

Summary

Summary

- Know your full stack, it is invaluable
  - Hardware, OS, Environment, Language, Protocols, Libraries
- Monitor all the things
  - Prevent educated guesses
- Do not trust other people's numbers!
  - Fake your own!

Resources

Resources

- http://www.elasticsearch.org/blog/white-paper-testing-automation-for-distributed-applications/
- http://www.elasticsearch.org/blog/elasticsearch-testing-qa-increasing-coverage-randomizing-test-runs/
- http://www.elasticsearch.org/blog/performance-considerations-elasticsearch-indexing/
- http://www.elasticsearch.org/blog/resiliency-elasticsearch/
- http://www.elasticsearch.org/blog/averages-can-dangerous-use-percentile/
- http://www.elasticsearch.org/blog/count-elasticsearch/
- http://www.elasticsearch.org/guide/en/elasticsearch/resiliency/current/
- https://www.youtube.com/watch?v=U1C5m8b0qg0 (Akka Cluster)

Resources

- http://jprante.github.io/2012/11/28/Elasticsearch-Java-Virtual-Machine-settings-explained.html
- https://plumbr.eu/blog/what-garbage-collector-are-you-using
- https://plumbr.eu/blog/g1-vs-cms-vs-parallel-gc
- http://www.slideshare.net/aragozin/garbage-collection-in-jvm
- https://github.com/aragozin/jvm-tools
- https://github.com/brettwooldridge/HikariCP/wiki/Down-the-Rabbit-Hole
- http://www.artima.com/underthehood/flowP.html
- http://en.wikipedia.org/wiki/Switch_case#Compilation
- http://www.elasticsearch.org/blog/disk-based-field-data-a-k-a-doc-values/
- http://static.googleusercontent.com/media/research.google.com/fr//pubs/archive/40671.pdf

Resources

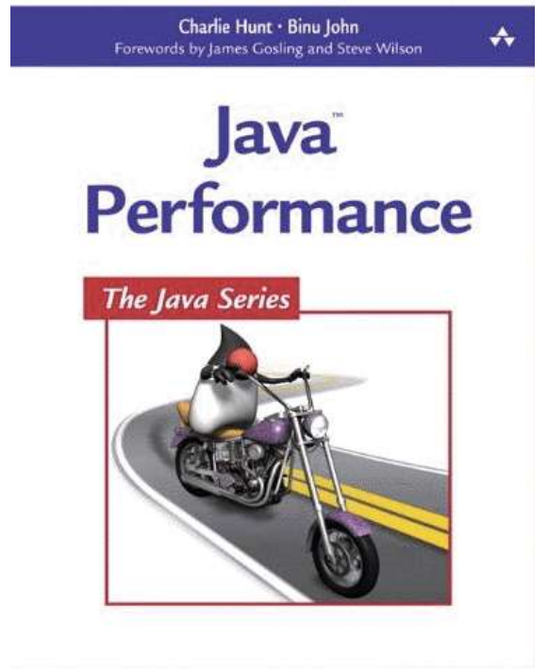
- https://www.youtube.com/watch?v=0b3sR32m0nU (How not to measure latency by Gil Tene)
- http://highscalability.com/blog/2012/3/12/google-taming-the-long-latency-tail-when-more-machines-equal.html
- http://www.ibm.com/developerworks/library/j-benchmark1/index.html
- http://www.ibm.com/developerworks/library/j-benchmark2/index.html
- https://www.youtube.com/watch?v=XmImGiVuJno (Benchmarking - You're doing it wrong by Aysylu Greenberg)

Resources

Java Performance

by Charlie Hunt

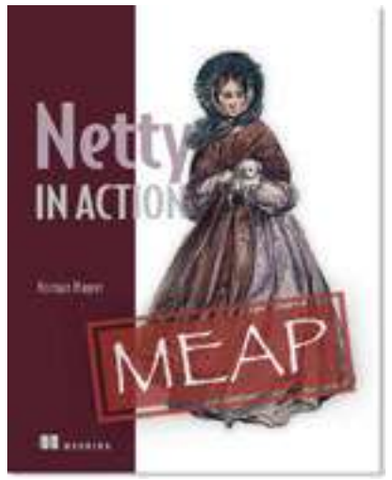
http://www.amazon.de/Java-Performance-Charlie-Hunt-ebook/dp/B005R4NELQ



Netty in Action

by Norman Maurer

http://www.amazon.de/Netty-Action-Norman-Maurer/dp/1617291471

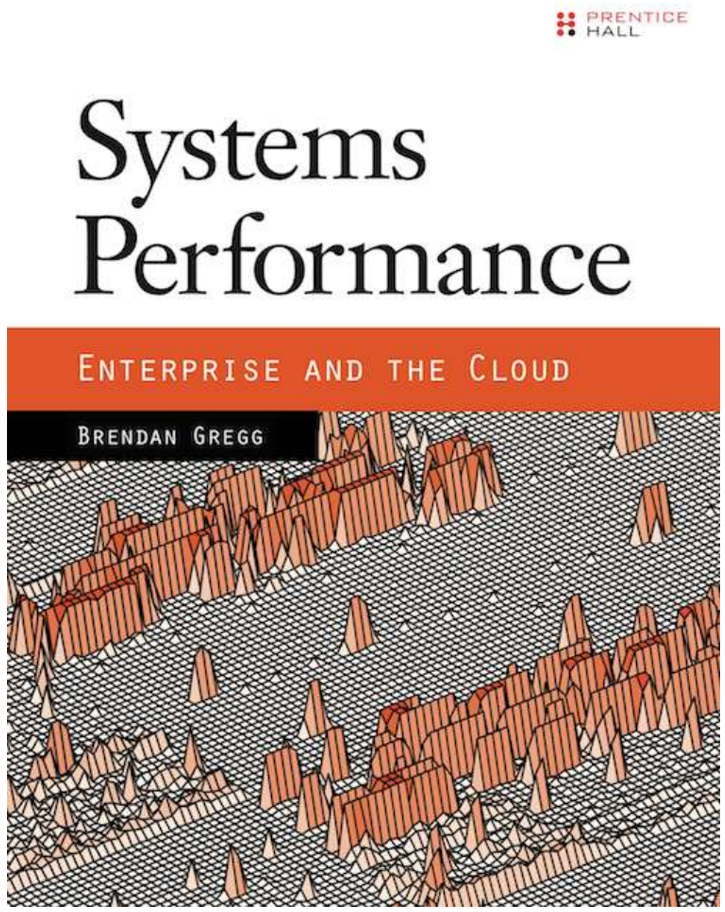


Resources

- Systems Performance - Enterprise and the cloud

  by Brendan Gregg

  http://www.amazon.de/Systems-Performance-Enterprise-Brendan-Gregg-ebook/dp/B00FLYU9T2

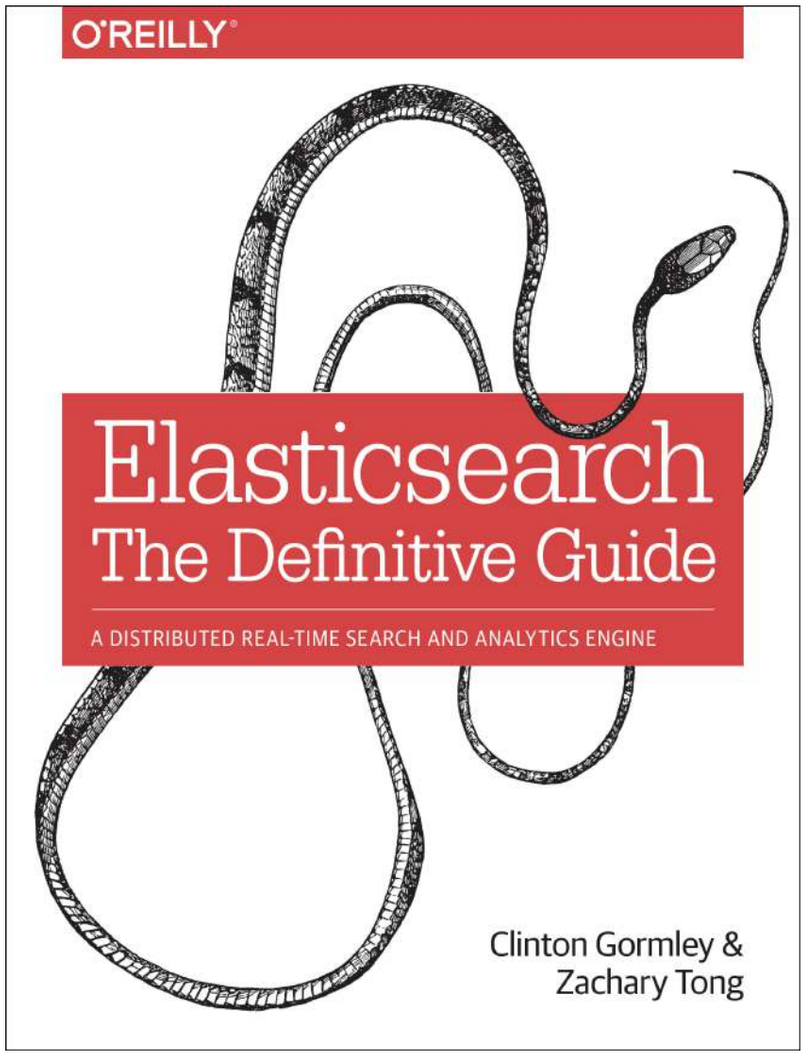


Resources

- Elasticsearch – The Definitive Guide

  by Clinton Gormley & Zachary Tong

  http://www.oreilly.de/catalog/9781449358549/



Thanks for listening!

We're hiring!
http://elasticsearch.com/jobs

We're helping!
http://elasticsearch.com/support
http://elasticsearch.com/training

Alexander Reelsen
@spinscale
alexander.reelsen@elasticsearch.com