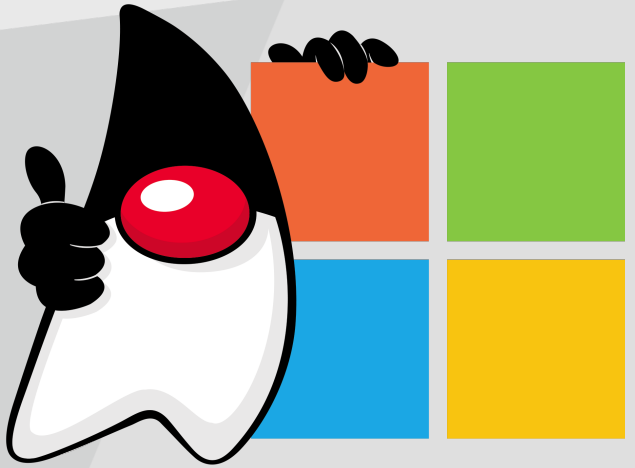
The Diabolical Developer

Microsoft



JVM Ergonomics for Containers and Kubernetes

Understand impact of resource constraints in the JVM

Martijn Verburg – Principal SWE Group Manager - @karianna

With many thanks to The Diabolical PM Bruno Borges!
Extra Guidance from Monica Beckwith, Kirk Pepperdine & Ben Evans

Microsoft Developer Division
Java Engineering Group (JEG), May 2022

Agenda

JVM inside Containers and on Kubernetes: what you must know!

- **Context**
  - Java At Microsoft, Hardware Resources and Cloud Compute
- **JVM Ergonomics**
  - Understand the default values of the JVM
  - How the amount of memory and CPU impacts selection of Garbage Collector
- **JVM Garbage Collectors**
  - Recommendations for better starting points in Cloud native applications
  - How to tune GCs
- **Java on Kubernetes**
  - Recommended starting points, Topologies
- **Conclusion**

Java at Microsoft

Java is widely used across Microsoft

2,000,000+ JVMs in production*

LinkedIn
2000+ Java microservices in production, Java 11+

Azure
Azure internal systems and infrastructure, Big Data etc.

Yammer
Back-end implemented in Java

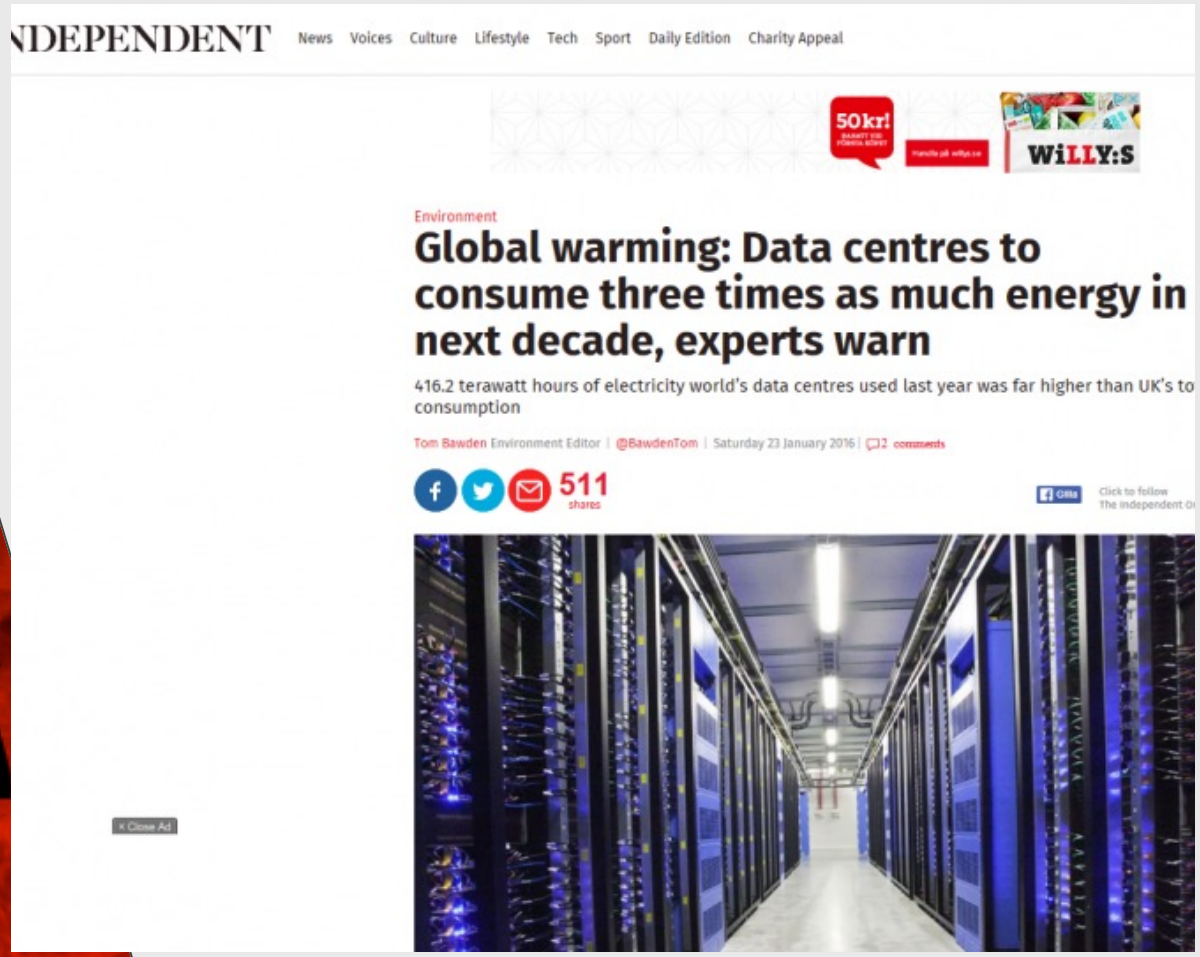
Minecraft
Thousands of servers built in Java and millions of players on the very popular Java Edition (Java 17+)

Android
50+ apps published by Microsoft in Google Play Store

Bing and MSN
Infrastructure with Java-based big data services

*Internal usage; does not include customer workloads, not all in containers (yet)

Cloud Compute and Climate Change



INDEPENDENT News Voices Culture Lifestyle Tech Sport Daily Edition Charity Appeal

50kr! WiLLY:S

Environment

Global warming: Data centres to consume three times as much energy in next decade, experts warn

416.2 terawatt hours of electricity world's data centres used last year was far higher than UK's to consumption

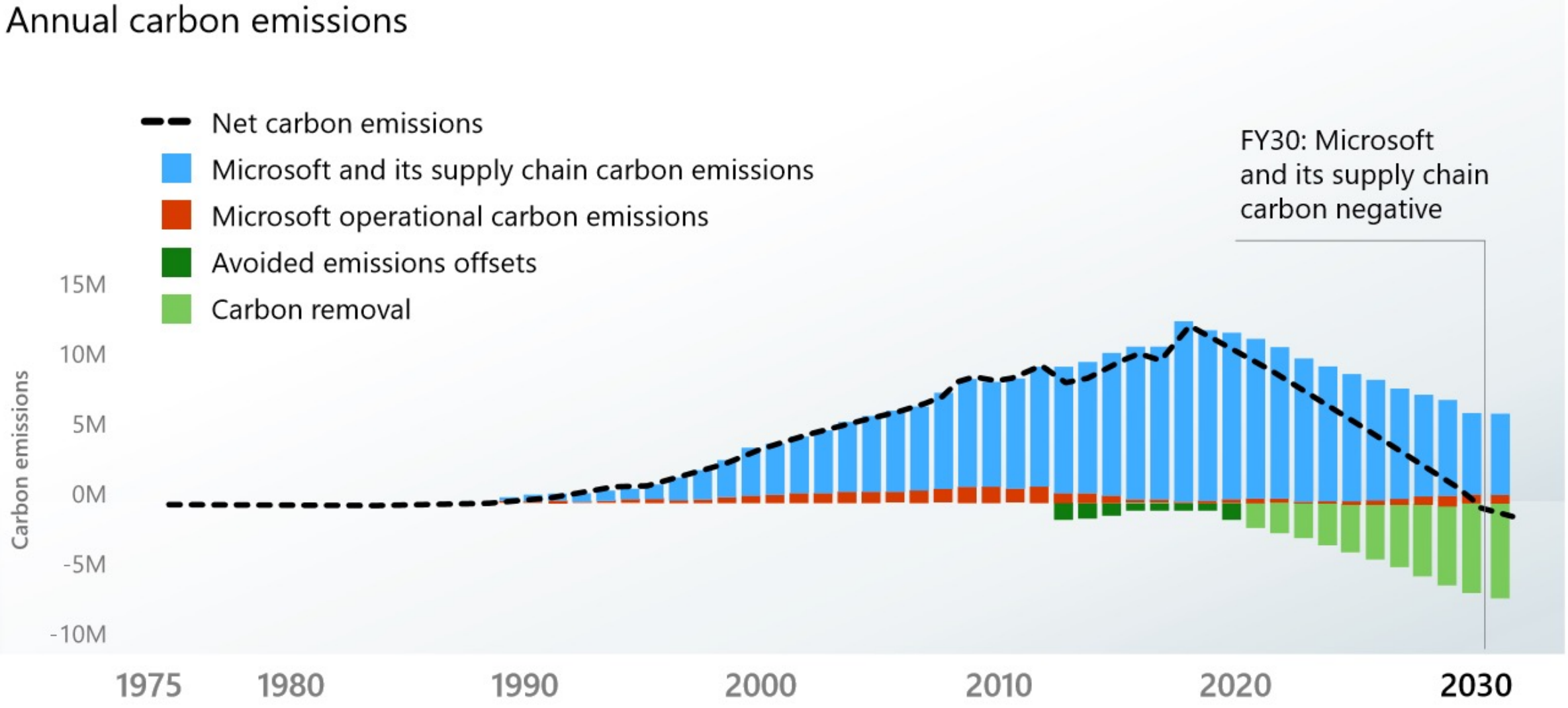
Tom Bawden Environment Editor | @BawdenTom | Saturday 23 January 2016 | 2 comments

511 shares

Click to follow The Independent

× Close Ad

Microsoft's pathway to carbon negative by 2030



Hardware Resources and Cloud Compute


Microsoft
Azure
Microsoft
Azure
Microsoft
Azure


Microsoft
Azure

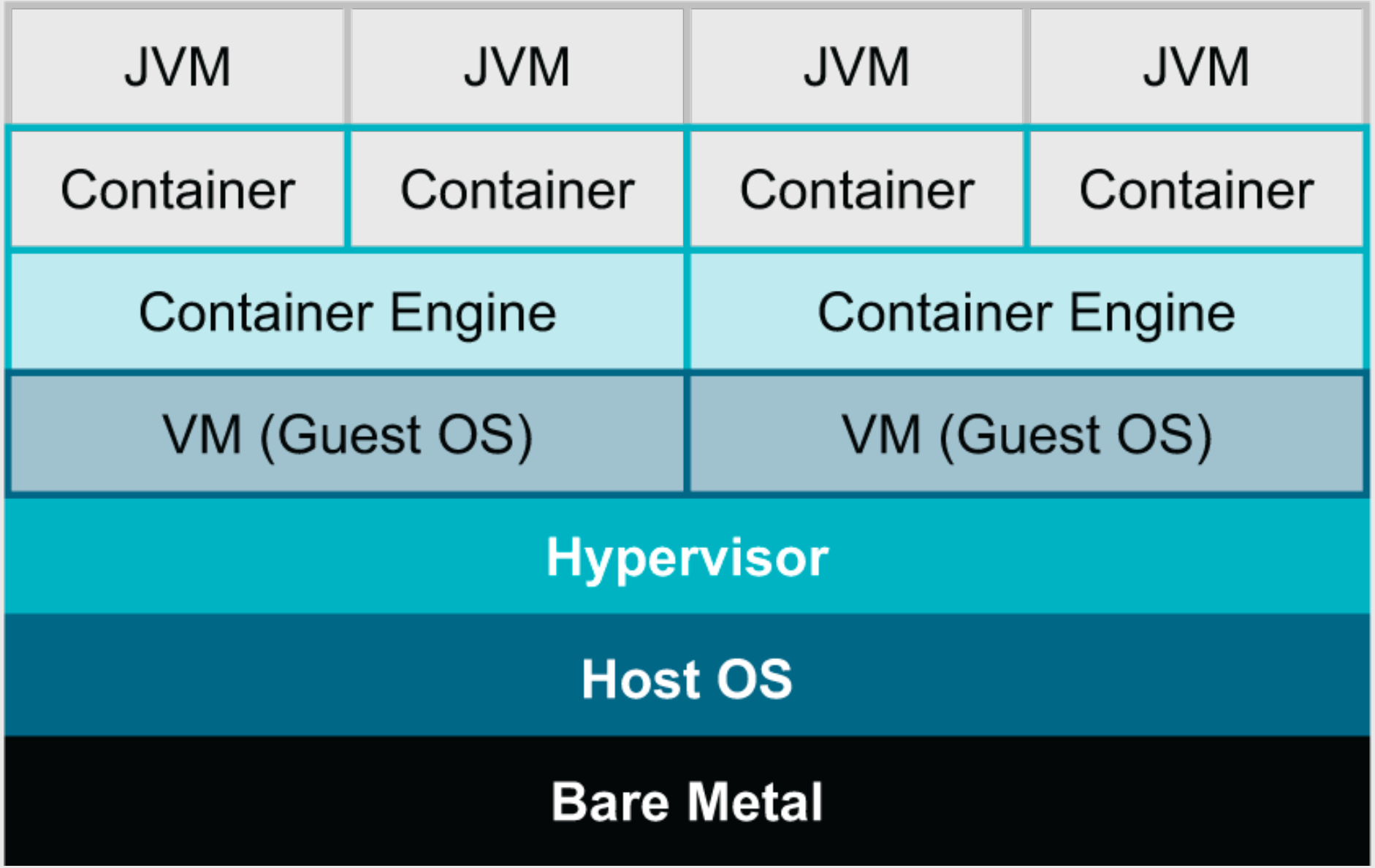

Microsoft
Azure


Microsoft
Azure


Microsoft
Azure

Data Centres in the Cloud Age

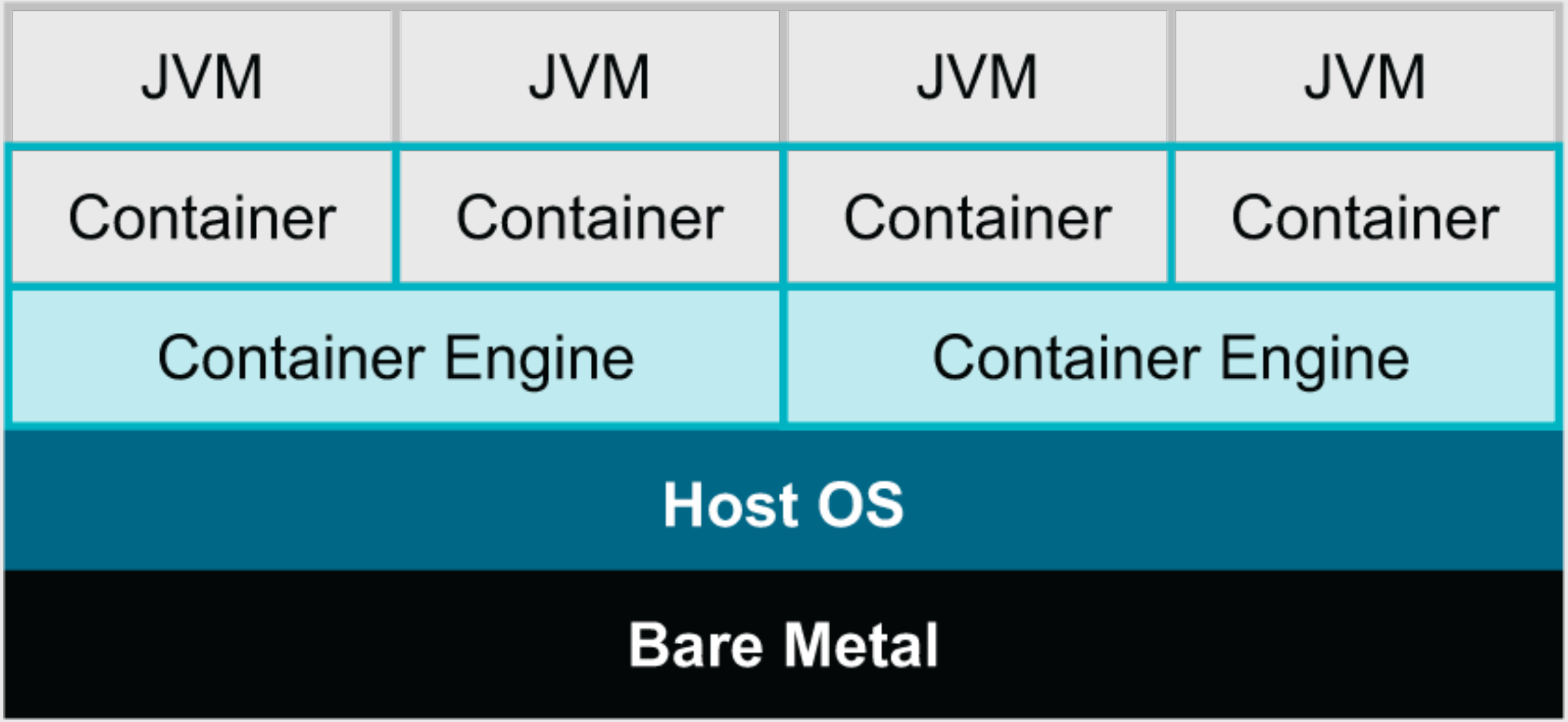
- Each host **has limited resources**
- Metal as a Service (MaaS) is rare
- Infrastructure as a Service (IaaS)
  - Typically, VMs and/or containers
- Understand the SKU you're on!


JVM
JVM
JVM
JVM
Container
Container
Container
Container
Container Engine
Container Engine
VM (Guest OS)
VM (Guest OS)
Hypervisor
Host OS
Bare Metal

Containers via Host O/S's and VMs

This is how we started with containers – note each layer takes some resource

- **Bare Metal Host (& Host O/S)**
  - Only has so much CPU, RAM, HDD and Network I/O capacity!
  - Often Linux, sometimes this is replaced by a Type-1 Hypervisor
- **Hypervisors**
  - Enables creation and maintenance of VMs, uses a small amount of resource to do so.
  - Type-1 Runs on bare metal, Type-2 runs on a host O/S
- **Virtual Machines (VMs)**
  - This is the IaaS unit SKU you usually get on cloud.
- **Containers**
  - Hello Docker (for most people) and K8s to orchestrate


JVM
JVM
JVM
JVM
Container
Container
Container
Container
Container Engine
Container Engine
Host OS
Bare Metal

Containers via Container Engines

A quick reminder

- **Bare Metal Host (& Host O/S)**
  - Only has so much CPU, RAM, HDD and Network I/O capacity!
- **Container Engines**
  - Replaces Host O/S's and Hypervisors in most cases. Serves up containers only.
- **Containers**
  - Hello Docker (for most people) and K8s to orchestrate

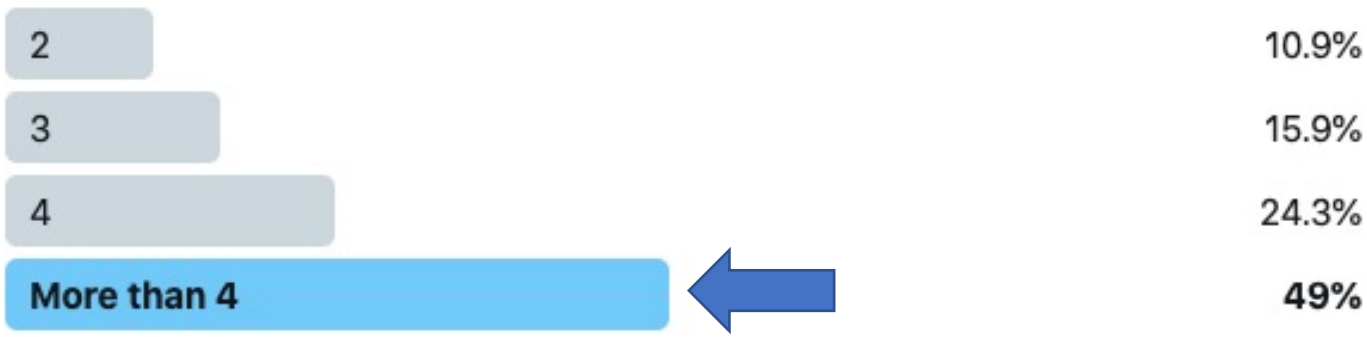
Calculate what you need with headroom

Seriously, 64GB of RAM will not give you 16x8GB VMs that work, stop it.

JVM Ergonomics

Question 1 of 5: How many Garbage Collectors are available on an usual vanilla @OpenJDK 11+ distribution?

#OpenJDK



239 votes · Final results

Question 2 of 5: Do you expect the JVM to choose the best Garbage Collector based on what ergonomics?

129 votes · Final results

Question 3 of 5: How many Garbage Collectors do you think the JVM may consider when evaluating ergonomics?

123 votes · Final results

Question 4 of 5: Do you trust JVM Ergonomics to pick the best GC for you?

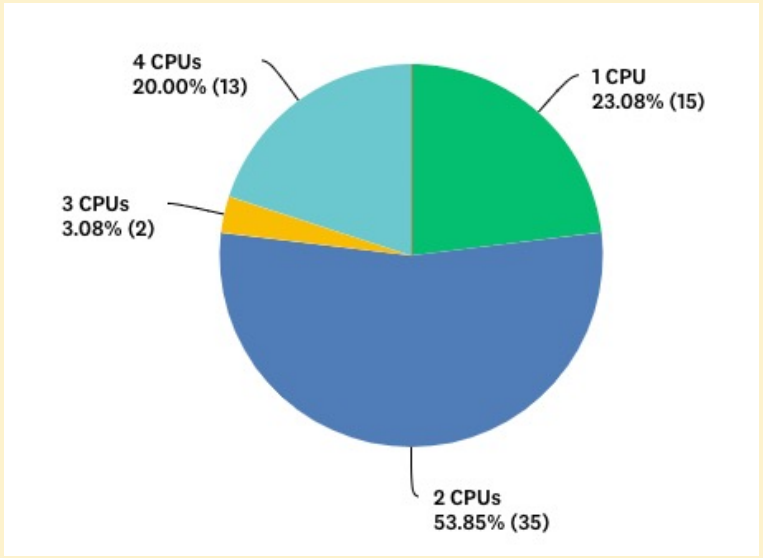
135 votes · Final results

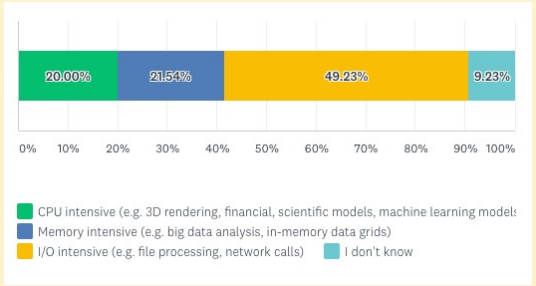
Question 5 of 5: Would you like better JVM Ergonomics?

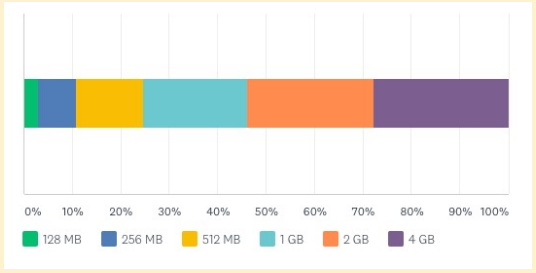
128 votes · Final results

Survey Summary (150 ppl)

65% of ppl



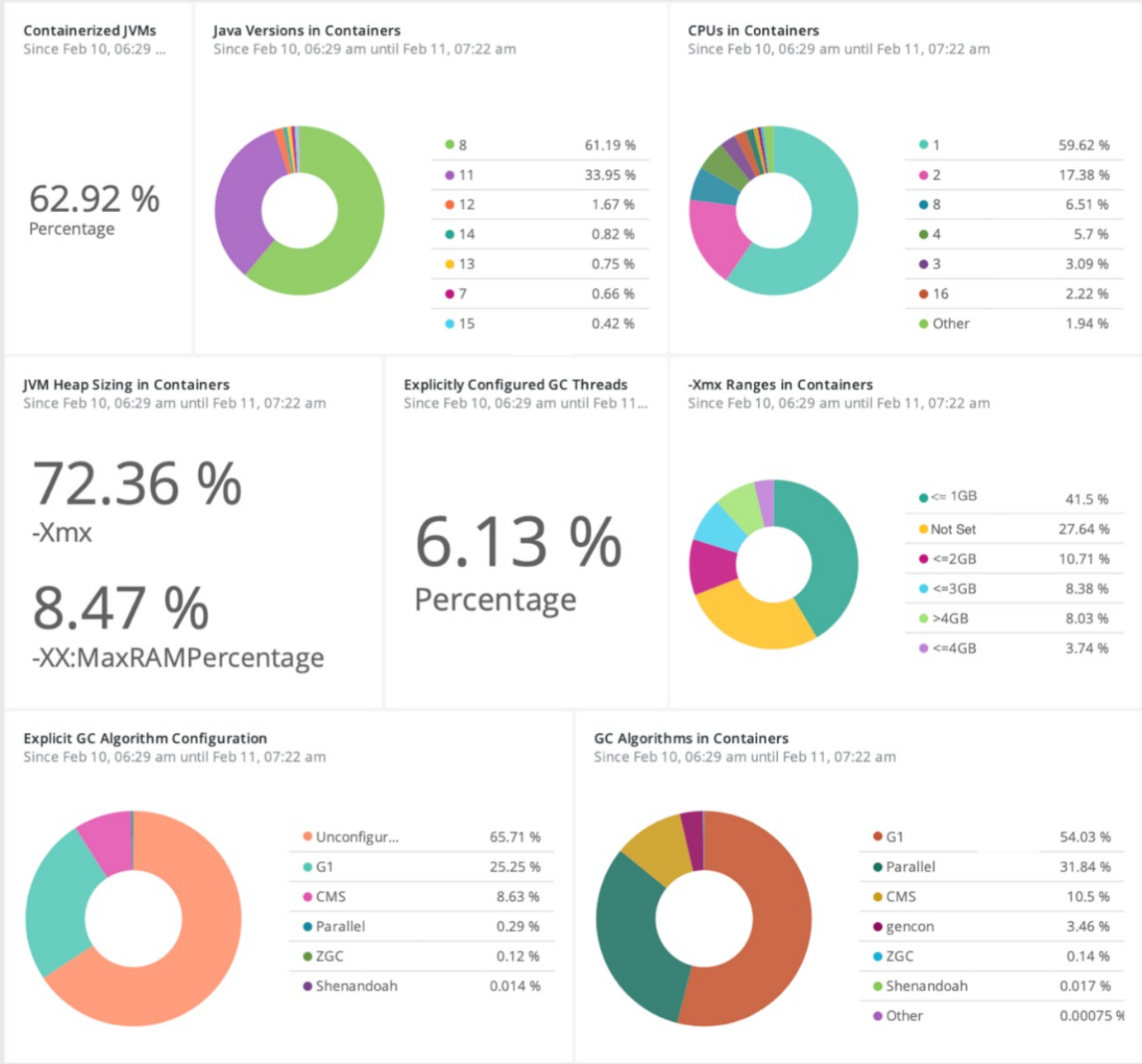




- Most devs are deploying JVM workloads in containers with:
  - Up to 4 CPUs (65%)
  - Up to 4 GB RAM (65%)
  - I/O intensive (50%)
- Overall
  - Up to 2 GB (48%)
  - Up to 3 CPUs (50%)

JVM Ergonomics

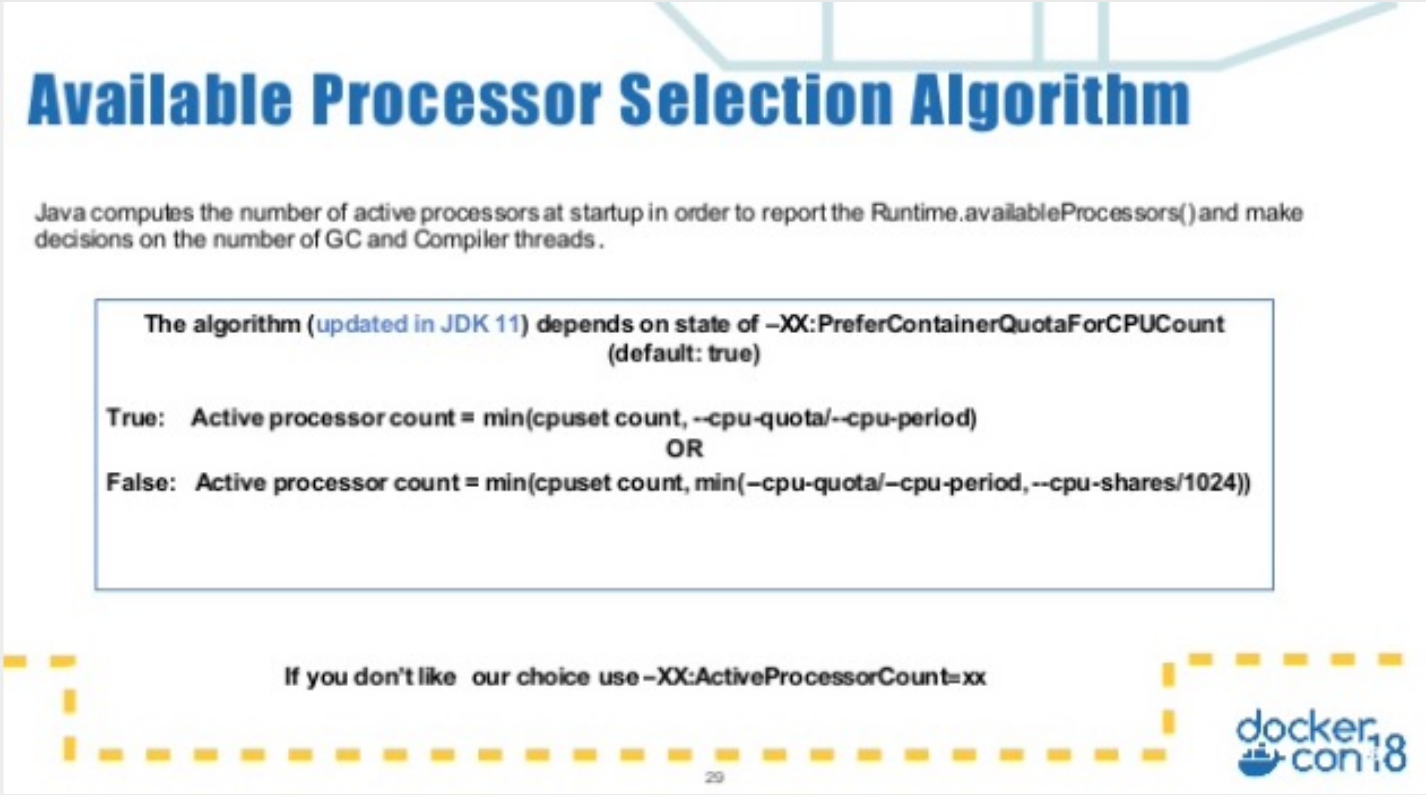
- **New Relic (Azure Partner)**
  - 10+ Million of prod JVMs analysed
  - Majority with 1 CPU
  - Majority with 1GB or less RAM
  - Majority with GC not configured
- **Typical ‘fixes’ to Perf issues:**
  - Increase heap size
  - More replicas
  - Migration to another stack
- **Ultimately, increased COGS**



JVM Ergonomics

Default settings when no GC is specified.

- **HotSpot JVM / OpenJDK**
  - Java 11 or later
    - SerialGC or G1GC
  - Java 8
    - SerialGC or ParallelGC
  - **Default GC**
    - **Serial GC if 1791MB or less** memory available.
  - *Otherwise, G1GC.*



JVM Ergonomics Demo

```java
public class App {

  public static void main(String args[]) {
      var procs = Runtime.getRuntime().availableProcessors();
      System.out.println("Active Processors: " + procs);
  }

}
```

```sh
#!/bin/sh
docker run \
  -v $HOME:/app \
  --cpus 2 \
  --memory 1500m \
  -ti --rm \
  mcr.microsoft.com/openjdk/jdk:17-ubuntu \
    java -XX:+PrintFlagsFinal /app/App.java \
    | grep -e "Use.*GC" -e "Active"
```

```
     int ActiveProcessorCount                      = -1                                  {product} {default}
    bool UseAdaptiveSizeDecayMajorGCCost           = true                                {product} {default}
    bool UseAdaptiveSizePolicyWithSystemGC         = false                               {product} {default}
    bool UseDynamicNumberOfGCThreads               = true                                {product} {default}
    bool UseG1GC                                   = false                               {product} {default}
    bool UseGCOverheadLimit                        = true                                {product} {default}
    bool UseMaximumCompactionOnSystemGC            = true                                {product} {default}
    bool UseParallelGC                             = false                               {product} {default}
    bool UseSerialGC                               = true                                {product} {ergonomic}
    bool UseShenandoahGC                           = false                               {product} {default}
    bool UseZGC                                    = false                               {product} {default}
Active Processors: 2
```

JVM Garbage Collectors

Garbage Collectors

Recommendations

	Serial	Parallel	G1	Z	Shenandoah
Number of cores	1	2+	2+	2+	2+
Multi-threaded	No	Yes	Yes	Yes	Yes
Java Heap size	<4GBytes	<4Gbytes	>4GBytes	>4GBytes	>4GBytes
Pause	Yes	Yes	Yes	Yes (<1ms)	Yes (<10ms)
Overhead	Minimal	Minimal	Moderate	Moderate+	Moderate++
Tail-latency Effect	High	High	High	Low	Moderate
JDK version	All	All	JDK 8+	JDK 17+	JDK 11+
Best for	**Single core, small heaps**	**Multi-core small heaps.** **Batch jobs, with any heap size.**	**Responsive in medium to large heaps (request-response/DB interactions)**	**responsive in medium to large heaps (request-response/DB interactions)**	**responsive in medium to large heaps (request-response/DB interactions)**

What to know

- **The JVM Heap**
  - Contiguous block of memory
  - Entire space is reserved
  - Only some space is allocated
  - Broken up into different areas or regions
- **Object Creation / Removal**
  - Objects are created by application (mutator) threads
  - Objects are removed or relocated by Garbage Collection
- **Poorly tuned GC leads to**
  - High pause times
  - High % of time spent pausing
  - Starvation of threads
  - OutOfMemoryError (OOME)
- **Tuning GC is worth**
  - Performance gains lead to Cost savings
- **Setting Heap size is not enough**
  - Understanding the workload is key
  - Select appropriate Garbage Collector
  - Enough CPUs
  - Performance requirements and SLAs

Heap Size Configuration

- **Default Ergonomics (Heap)**
  - Inside containers is **1/4** available memory.
  - Outside containers is **1/64** available memory.
- **Recommended starting point**
  - **Servers**
    - Set to whatever the application needs
  - **Containers**
    - Set to whatever the application needs but 75% of container memory limit
      - You can go higher, the larger your heap.
- **Manually configure Heap**
  - **-Xmx**
    - Set value in MB: 256m
    - Set value in GB: 2g
    - Great for well-sized workloads
  - **-XX:MaxRAMPercentage**
    - Set value in percentage: 75
    - Great for workloads to be scaled along container memory limits

"[GC] Tuning is basically trying to optimize this [object] moving to 'move as little as possible, as late as possible so not disturb the flow.'"

Monica Beckwith
Principal Software Engineer
Microsoft Java Engineering Group

[Watch Monica's Tuning and Optimizing Java Garbage Collection (infoq.com)]

JVM Ergonomics and GCs – Summary

Java 11+ - OpenJDK HotSpot Ergonomics will use, by default, either SerialGC or G1GC

- G1GC only when 2+ available processors and 1792+ MB available memory – regardless of heap size.
- SerialGC otherwise.

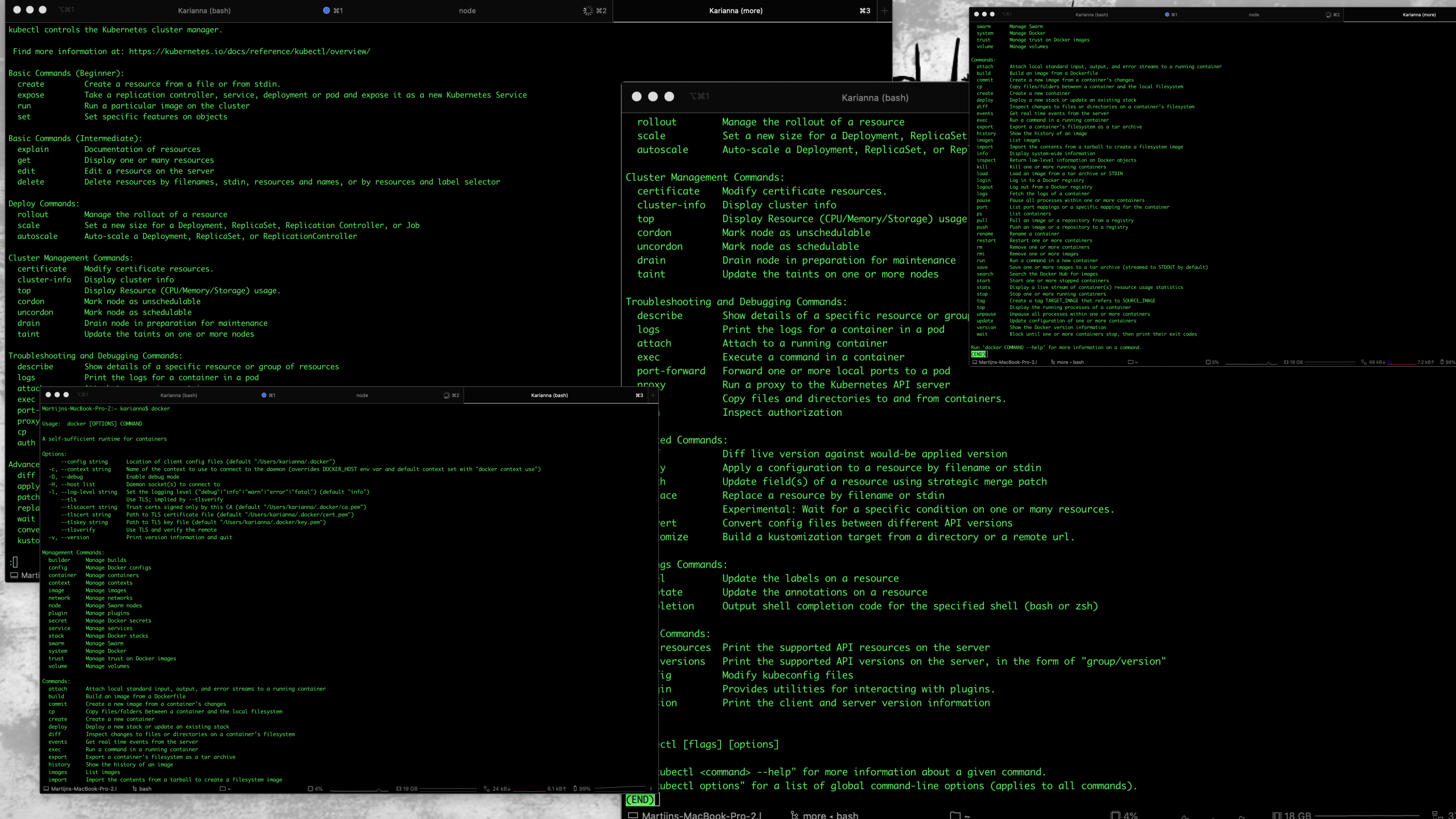
ParallelGC in general outperforms G1GC for smaller heaps

- Up to 4GB, ParallelGC performs better as a throughput GC.
- Between 2-4GB, ParallelGC may still perform better for throughput, but G1GC could be considered.
- ParallelGC still triggers Stop the World (StW), impacting in latency on tail performance.

Heap size not being properly dimensioned for containers by Ergonomics

- Default ergonomics will allocate 1/4 of available memory when inside containers, and 1/64 if not in container.
- Make sure a heap size is defined, either with -Xmx or with -XX:MaxRAMPercentage. Allocate at least 75%.

Java on Kubernetes


Karianna (bash)
node
Karianna (more)
kubectl controls the Kubernetes cluster manager.
Find more information at: https://kubernetes.io/docs/reference/kubectl/overview/
Basic Commands (Beginner):
create Create a resource from a file or from stdin.
expose Take a replication controller, service, deployment or pod and expose it as a new Kubernetes Service
run Run a particular image on the cluster
set Set specific features on objects
Basic Commands (Intermediate):
explain Documentation of resources
get Display one or many resources
edit Edit a resource on the server
delete Delete resources by filenames, stdin, resources and names, or by resources and label selector
Deploy Commands:
rollout Manage the rollout of a resource
scale Set a new size for a Deployment, ReplicaSet, Replication Controller, or Job
autoscale Auto-scale a Deployment, ReplicaSet, or ReplicationController
Cluster Management Commands:
certificate Modify certificate resources.
cluster-info Display cluster info
top Display Resource (CPU/Memory/Storage) usage.
cordon Mark node as unschedulable
uncordon Mark node as schedulable
drain Drain node in preparation for maintenance
taint Update the taints on one or more nodes
Troubleshooting and Debugging Commands:
describe Show details of a specific resource or group of resources
logs Print the logs for a container in a pod
Karianna (bash)
rollout Manage the rollout of a resource
scale Set a new size for a Deployment, ReplicaSet
autoscale Auto-scale a Deployment, ReplicaSet, or Rep
Cluster Management Commands:
certificate Modify certificate resources.
cluster-info Display cluster info
top Display Resource (CPU/Memory/Storage) usage
cordon Mark node as unschedulable
uncordon Mark node as schedulable
drain Drain node in preparation for maintenance
taint Update the taints on one or more nodes
Troubleshooting and Debugging Commands:
describe Show details of a specific resource or grou
logs Print the logs for a container in a pod
attach Attach to a running container
exec Execute a command in a container
port-forward Forward one or more local ports to a pod
proxy Run a proxy to the Kubernetes API server
Copy files and directories to and from containers.
Inspect authorization
ed Commands:
Diff live version against would-be applied version
y Apply a configuration to a resource by filename or stdin
h Update field(s) of a resource using strategic merge patch
ace Replace a resource by filename or stdin
Experimental: Wait for a specific condition on one or many resources.
ert Convert config files between different API versions
omize Build a kustomization target from a directory or a remote url.
gs Commands:
l Update the labels on a resource
tate Update the annotations on a resource
letion Output shell completion code for the specified shell (bash or zsh)
Commands:
resources Print the supported API resources on the server
versions Print the supported API versions on the server, in the form of "group/version"
ig Modify kubeconfig files
in Provides utilities for interacting with plugins.
ion Print the client and server version information
ctl [flags] [options]
ubectl <command> --help" for more information about a given command.
ubectl options" for a list of global command-line options (applies to all commands).
(END)
Martijns-MacBook-Pro-2.l
more ◂ bash
Martijns-MacBook-Pro-2:~ karianna$ docker
Usage: docker [OPTIONS] COMMAND
A self-sufficient runtime for containers
Options:
--config string Location of client config files (default "/Users/karianna/.docker")
-c, --context string Name of the context to use to connect to the daemon (overrides DOCKER_HOST env var and default context set with "docker context use")
-D, --debug Enable debug mode
-H, --host list Daemon socket(s) to connect to
-l, --log-level string Set the logging level ("debug"|"info"|"warn"|"error"|"fatal") (default "info")
--tls Use TLS; implied by --tlsverify
--tlscacert string Trust certs signed only by this CA (default "/Users/karianna/.docker/ca.pem")
--tlscert string Path to TLS certificate file (default "/Users/karianna/.docker/cert.pem")
--tlskey string Path to TLS key file (default "/Users/karianna/.docker/key.pem")
--tlsverify Use TLS and verify the remote
-v, --version Print version information and quit
Management Commands:
builder Manage builds
config Manage Docker configs
container Manage containers
context Manage contexts
image Manage images
network Manage networks
node Manage Swarm nodes
plugin Manage plugins
secret Manage Docker secrets
service Manage services
stack Manage Docker stacks
swarm Manage Swarm
system Manage Docker
trust Manage trust on Docker images
volume Manage volumes
Commands:
attach Attach local standard input, output, and error streams to a running container
build Build an image from a Dockerfile
commit Create a new image from a container's changes
cp Copy files/folders between a container and the local filesystem
create Create a new container
deploy Deploy a new stack or update an existing stack
diff Inspect changes to files or directories on a container's filesystem
events Get real time events from the server
exec Run a command in a running container
export Export a container's filesystem as a tar archive
history Show the history of an image
images List images
import Import the contents from a tarball to create a filesystem image
Martijns-MacBook-Pro-2.l
bash
swarm Manage Swarm
system Manage Docker
trust Manage trust on Docker images
volume Manage volumes
Commands:
attach Attach local standard input, output, and error streams to a running container
build Build an image from a Dockerfile
commit Create a new image from a container's changes
cp Copy files/folders between a container and the local filesystem
create Create a new container
deploy Deploy a new stack or update an existing stack
diff Inspect changes to files or directories on a container's filesystem
events Get real time events from the server
exec Run a command in a running container
export Export a container's filesystem as a tar archive
history Show the history of an image
images List images
import Import the contents from a tarball to create a filesystem image
info Display system-wide information
inspect Return low-level information on Docker objects
kill Kill one or more running containers
load Load an image from a tar archive or STDIN
login Log in to a Docker registry
logout Log out from a Docker registry
logs Fetch the logs of a container
pause Pause all processes within one or more containers
port List port mappings or a specific mapping for the container
ps List containers
pull Pull an image or a repository from a registry
push Push an image or a repository to a registry
rename Rename a container
restart Restart one or more containers
rm Remove one or more containers
rmi Remove one or more images
run Run a command in a new container
save Save one or more images to a tar archive (streamed to STDOUT by default)
search Search the Docker Hub for images
start Start one or more stopped containers
stats Display a live stream of container(s) resource usage statistics
stop Stop one or more running containers
tag Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
top Display the running processes of a container
unpause Unpause all processes within one or more containers
update Update configuration of one or more containers
version Show the Docker version information
wait Block until one or more containers stop, then print their exit codes
Run 'docker COMMAND --help' for more information on a command.
(END)
Martijns-MacBook-Pro-2.l
more ◂ bash

The passionate, functional, micro-serviced approach

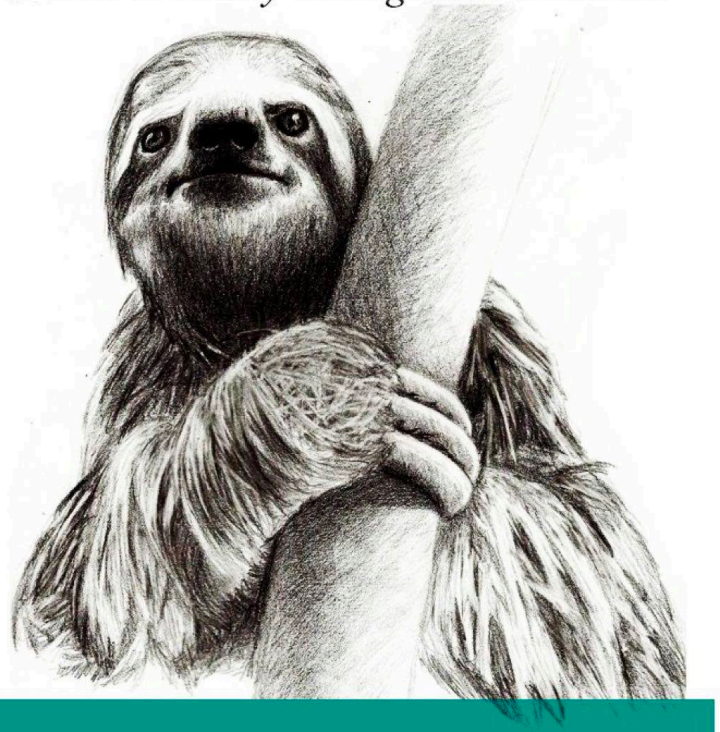
Expert

Resumé Driven Development

O RLY?

@ThePracticalDev

Cutting corners to meet arbitrary management deadlines



Essential

Copying and Pasting from Stack Overflow

O'REILLY®

The Practical Developer
@ThePracticalDev

Kubernetes CPU Throttling

How it impacts the JVM

- **CPU requests on Kubernetes are for CPU time**
  - "1000m" does *NOT* mean a single *vCPU*, or *core*.
  - "1000m" means the application can consume a full CPU cycle per period.
  - "1000m" allows an application with multiple threads to run in parallel.
    - When all threads combined consume "1000m" in CPU time, the application is throttled.

      Example
      - Thread A spends 400m; Thread B spends 500m. Thread C spends 100m.
      - App now must wait 500m for the next cycle.

- **Java applications are, in general, multi-threaded**
  - Concurrent GCs will have their own threads.
  - Web apps and REST/gRPC microservices will have their own threads.
  - Database Connection Pools will have their own threads.

CPU Throttling

How the JVM is throttled on Kubernetes

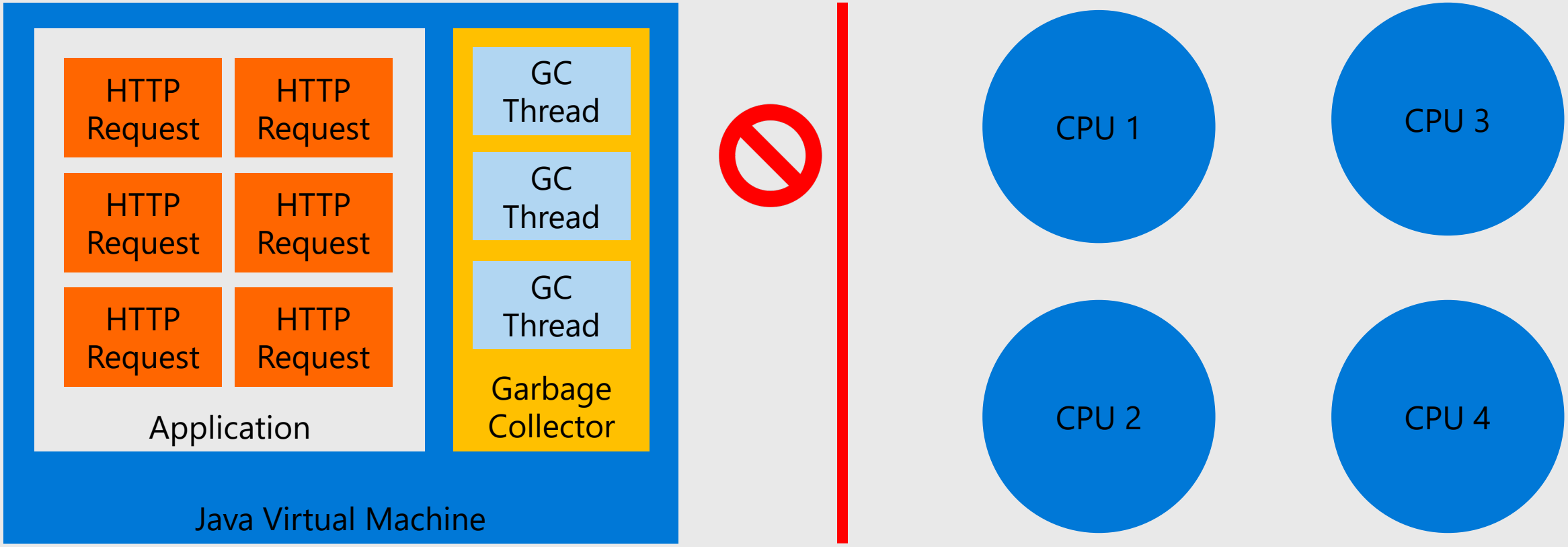
CPU Limit: 1000m

Remaining CFS Period: 100ms

- **Each request:** 200m
- **Remaining CPU time:** 200m
- **GC Work (total):** 200m
- **Remaining CPU time:** 0m

80ms

60ms



Application throttled for 60ms

JVM on Kubernetes

- **JVM Available Processors**
  - Up to 1000m: 1 proc
  - 1001-2000m: 2 procs
  - 2001-3000m: 3 procs
  - …

- **Trick the JVM**
  - Limit may be 1000m, but you may still tell the JVM it can use 2 or more processors!
  - Use this flag:
    -XX:ActiveProcessorCount

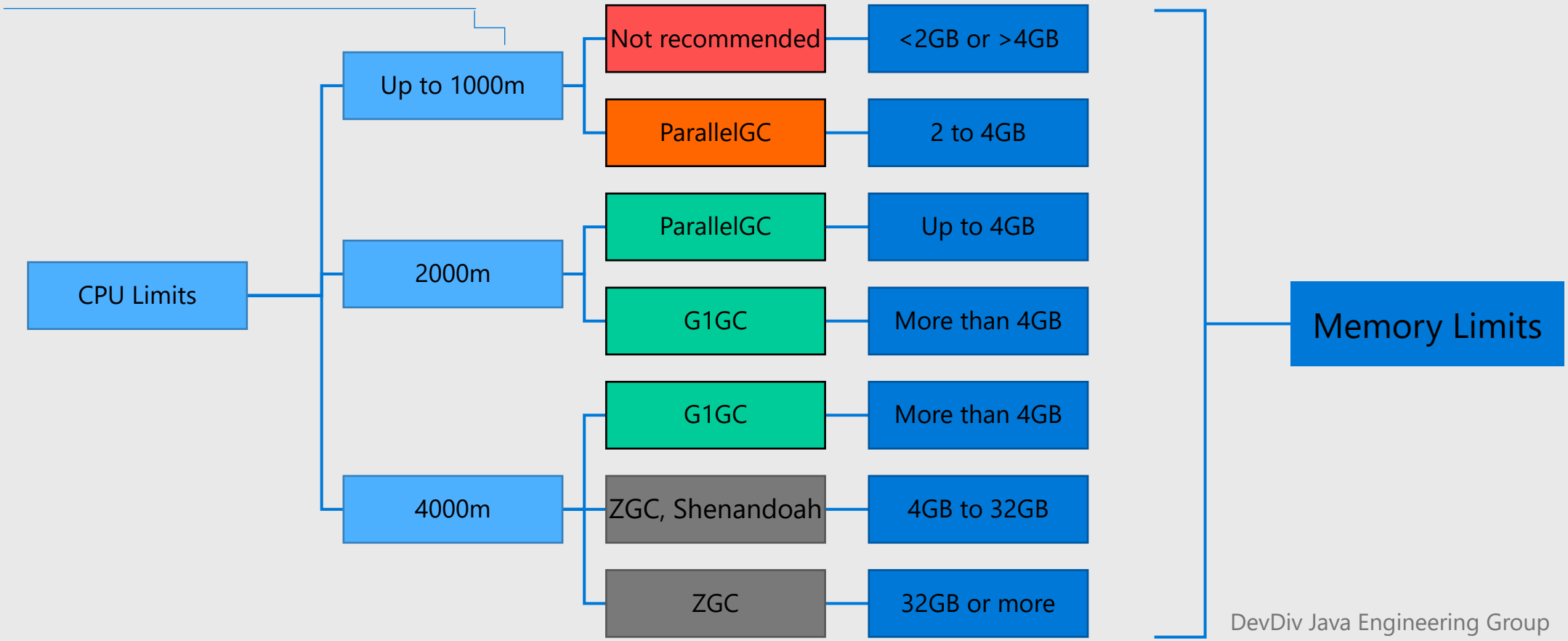
Kubernetes: Better Starting Points

Recommendations to follow instead of JVM Ergonomics

With 1000m or less, set:

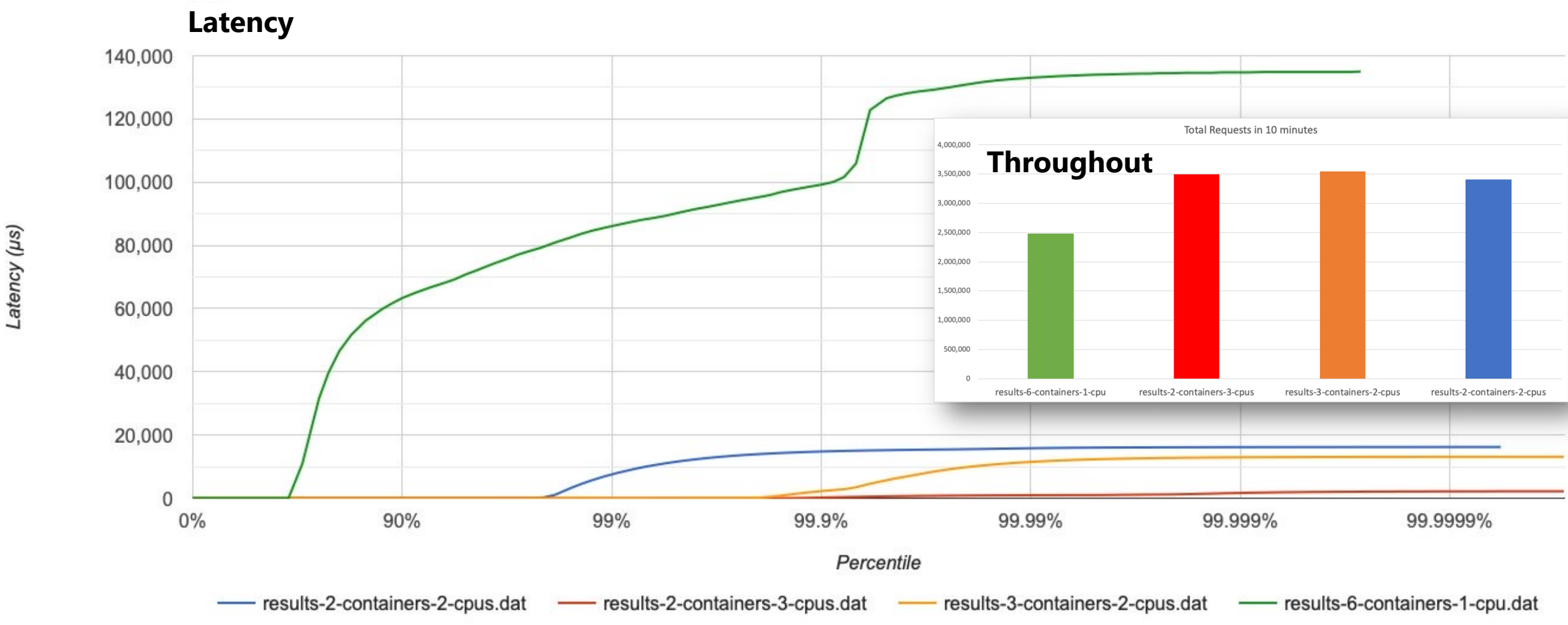
`--XX:ActiveProcessorCount=2`

For small JVM Heap set to 75%
then increase % as Heap increases



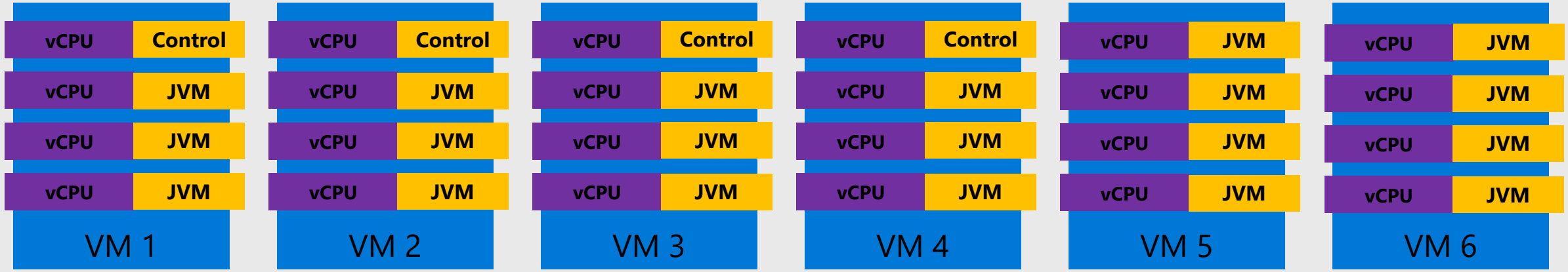
Benchmark

Latency: lower is better. Throughput: higher is better.



Azure Kubernetes Cluster

Short but wide – 6 x 4 = 24 vCPUs



- **D4 v3 VM** $0.192/hour
  - **4 vCPU**
  - **16 GB**
- **JVM**
  - 1 vCPU
  - 2 GB RAM

- **Total Resources Consumed**
  - 18 JVMs replicas
  - 18 vCPUs
  - 36 GB of RAM (of 96)

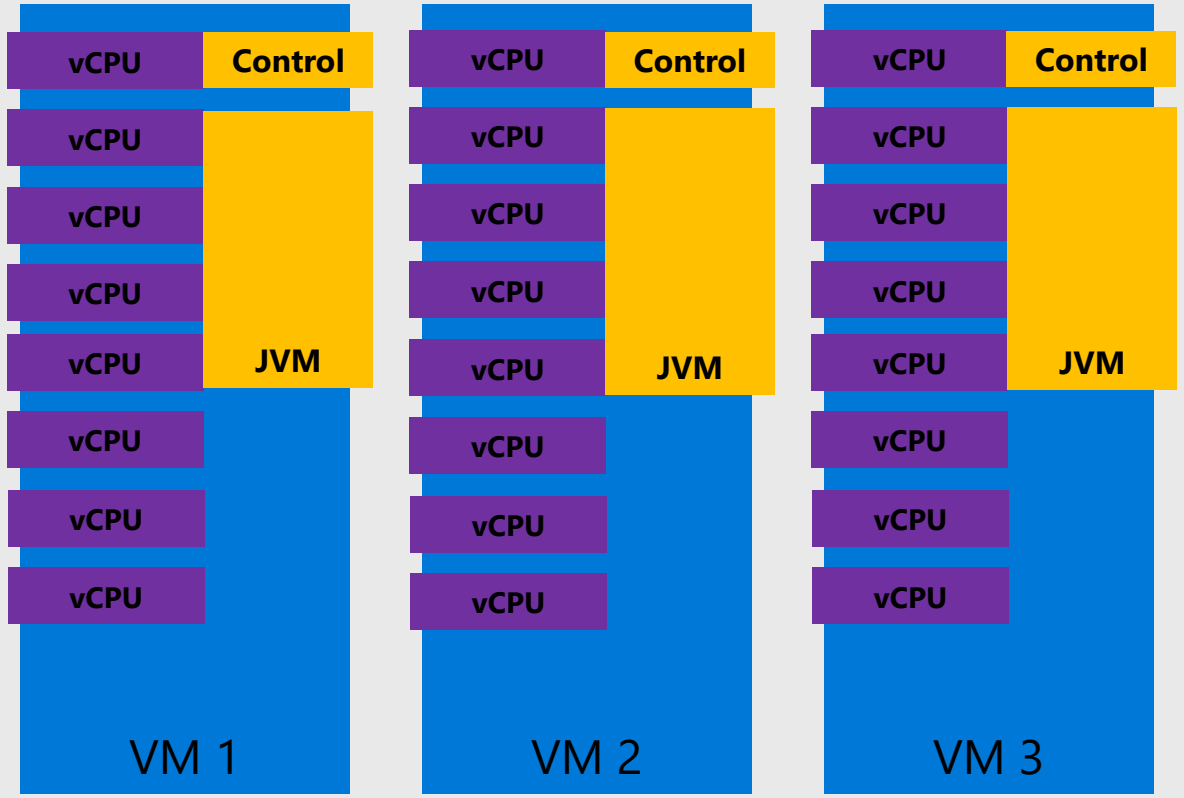
Estimate: $840.96

- Garbage Collector selected by Ergonomics:
  - **Serial GC**

- Concurrent/Parallel GCs won't be effective
- Constant CPU Throttling on each JVM
- Constant Stop-the-World by GC
- High latency, low throughput

Azure Kubernetes Cluster

Tall but narrow – 3 x 8 = 24 vCPUs



- **D8 v3 VM** $0.384/hour
  - 8 vCPUs
  - 32 GB
- **JVM**
  - 8 GB RAM
  - 4 vCPUs
- **Total Resources Consumed**
  - 12 vCPUs
  - 24 GB of RAM (of 96)
- **Garbage Collector** *(recommended)*:
  - G1GC
- **Benefits**
  - CPU Throttling unlikely
  - Lower latency, higher throughput

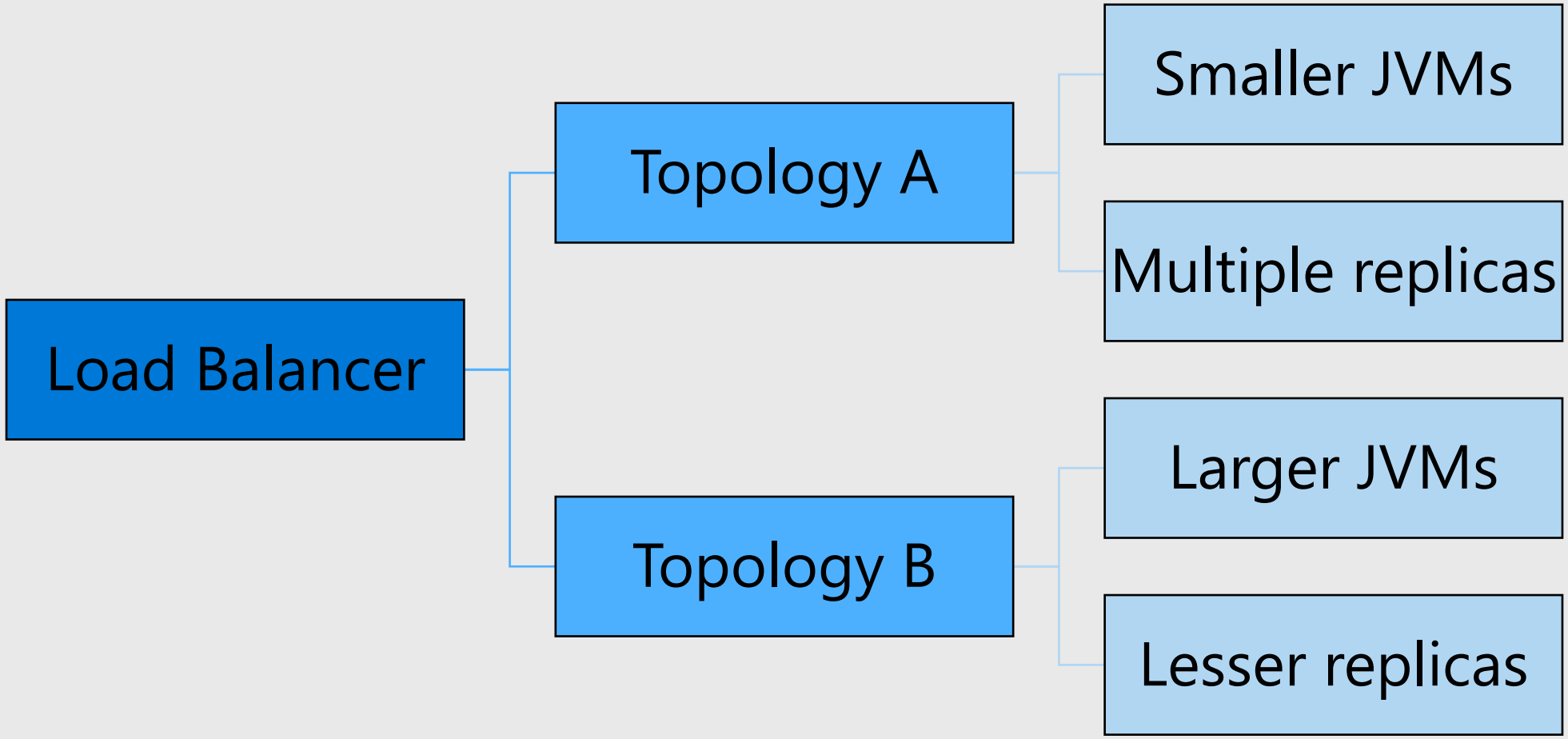
Estimate: $840.96 (same cost)

Savings:

- 9 vCPUs on standby
- 72 GB of RAM on standby

A/B Routing Multiple Topologies

Monitor the topologies for resource consumption, latency, and throughput.



Steps to Address Perf Issues

Optimize runtime for the workload

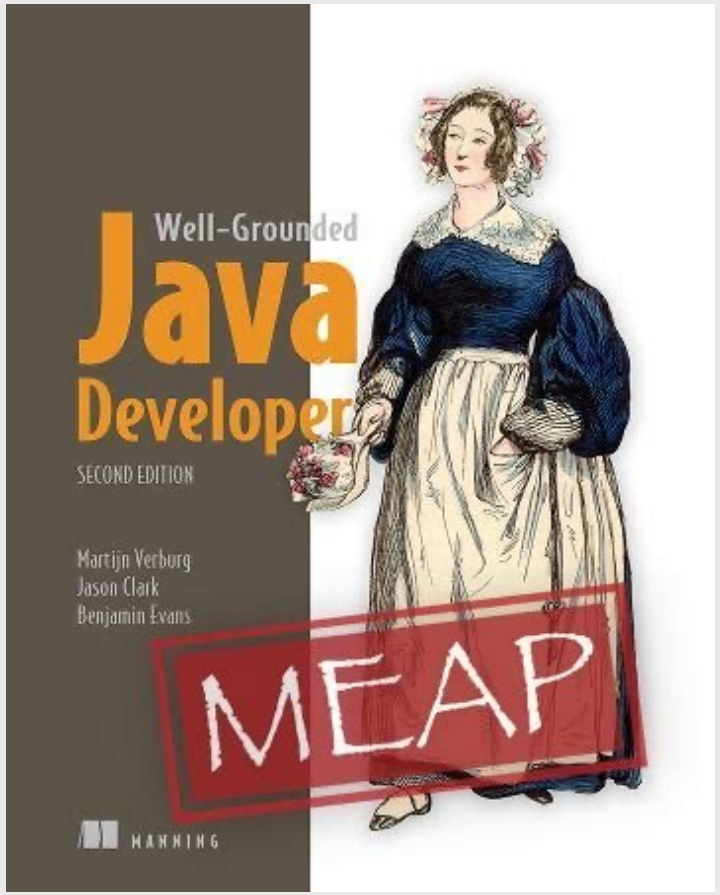
- **Understand Your Tech Stack**
  - Understand how the runtime responds to workloads
  - Understand JVM Ergonomics
  - Understand JVM Garbage Collectors
- **Observe and Analyze**
  - Monitor with Azure App Insights and other APM solutions
  - Analyze JVM data with JDK Flight Recorder (JFR) and Microsoft JFR Streaming
  - Analyze Garbage Collection logs with GC analyzers and Microsoft GCToolKit
- **Reorganize existing resources**
  - Consume the same amount of resources
  - Increase the performance
  - Maintain or reduce the cost

Conclusion

Java on Kubernetes scaling

- **Different workloads may need different topologies**
  - Scaling out with more replicas is not a silver bullet for performance increase
- **Give more resources to JVMs in the beginning**
  - Lesser replicas, more CPU/memory
- **Start with Parallel GC for smaller heaps**
  - Avoid JVM default ergonomics
  - Ensure you know which GC is being used
- **Increase performance by understanding bottlenecks**
  - Analyse JFR data
  - Analyse GC logs
- **Scale out, and up, as needed**

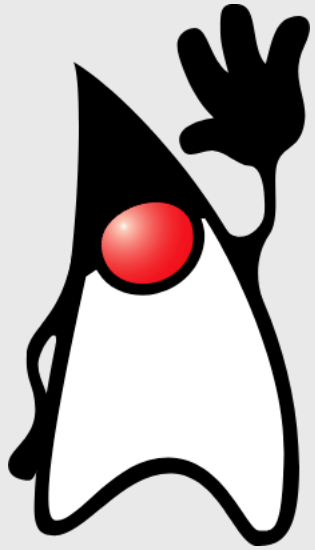
Learn more in Depth!



https://www.manning.com/books/the-well-grounded-java-developer-second-edition

https://docs.microsoft.com/en-us/azure/developer/java/containers/overview

The End



@javaatmicrosoft

https://docs.microsoft.com/java

Microsoft Developer Division

Java Engineering Group (JEG)