So you want to build the worlds largest Jenkins cluster?

Stephen Connolly

@connolly_s

So you want to build the world's biggest Jenkins Cluster

Stephen Connolly

#JenkinsWorld

About me

Life

- ❤️❤️❤️ My wife and son
- ❤️❤️ Running (Marathon x 2)

About me

Life

- ❤️❤️❤️ My wife and son
- ❤️❤️ Running (Marathon x 2)
- 💔 My L5/S1 right now…

About me

Life

- ❤️❤️❤️ My wife and son
- ❤️❤️ Running (Marathon x 2)

Work

- IT -> Chemistry -> IT

Jenkins & OSS

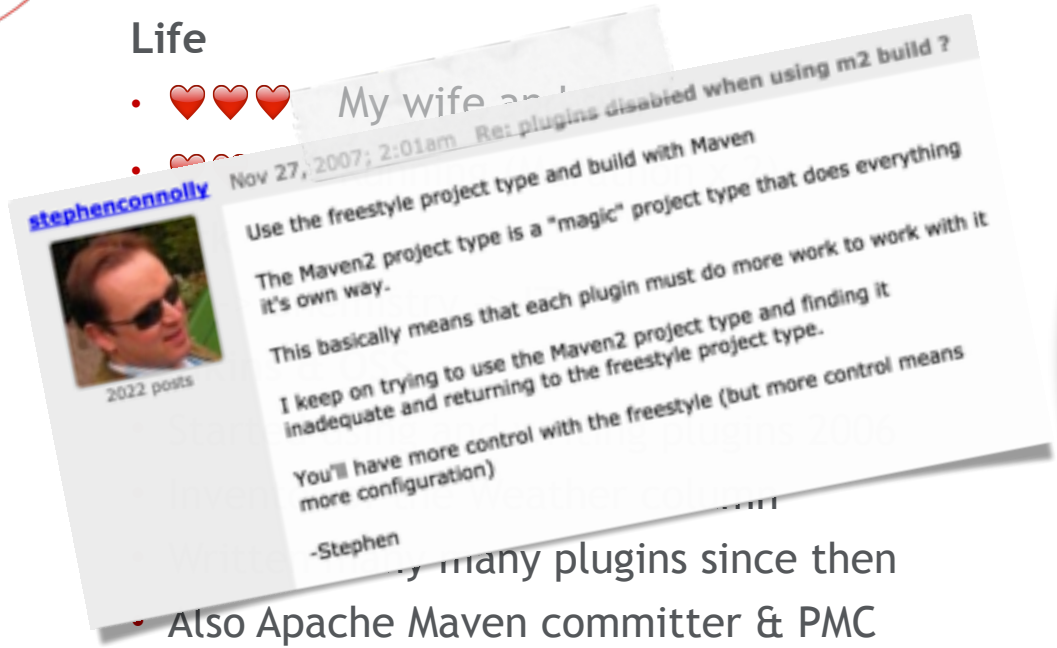
- Started using and writing plugins 2006
- Inventor of the Weather column
- Written many many plugins since then
- Also Apache Maven committer & PMC

About me

Jenkins World 2016

Life

- ❤❤❤ My wife and
- [illegible]



- [illegible] many plugins since then
- Also Apache Maven committer & PMC

Fighting against the evil Maven Job type since 2007…

Agenda

Real world

Theory

Experiments

Applied

Before we start…

Before we start…

world’s biggest Jenkins cluster

Before we start…

world's biggest Jenkins cluster

What does this mean?

Before we start…

world's biggest Jenkins cluster

or this?

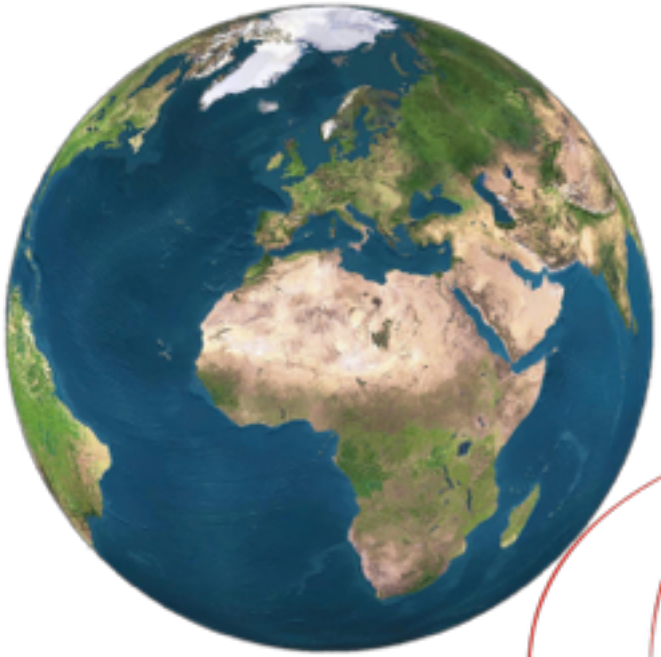
Biggest…

- If we want the physically largest…

#JenkinsWorld

Biggest…

- If we want the physically largest, this is easy with AWS:
  - Master in EU (Ireland)
  - Agent in US East
  - Agent in US West
  - Agent in São Paulo
  - Agent in Asia Pacific (Tokyo)
  - Agent in Asia Pacific (Sydney)

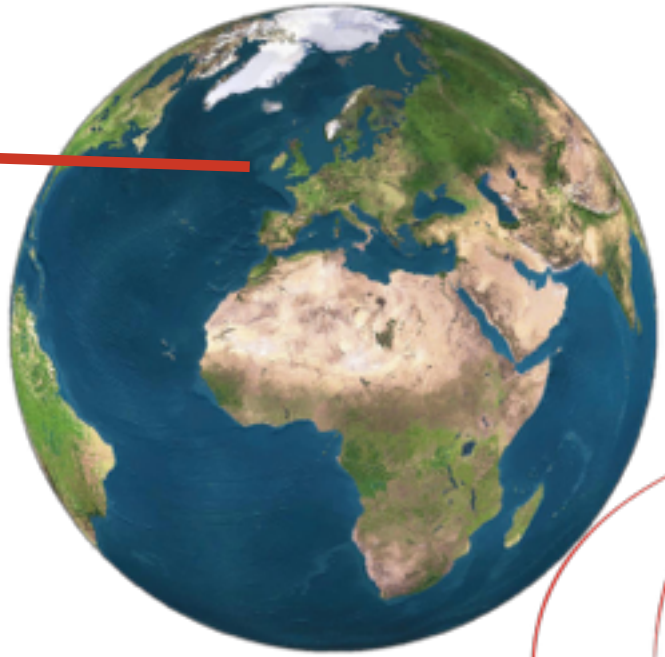


Smallest Bounding Sphere containing all points

Biggest…

- If we want the physically largest, this is easy with AWS:
  - Master in EU (Ireland)
  - Agent in US East
  - Agent in US West
  - Agent in São Paulo
  - Agent in Asia Pacific (Tokyo)
  - Agent in Asia Pacific (Sydney)

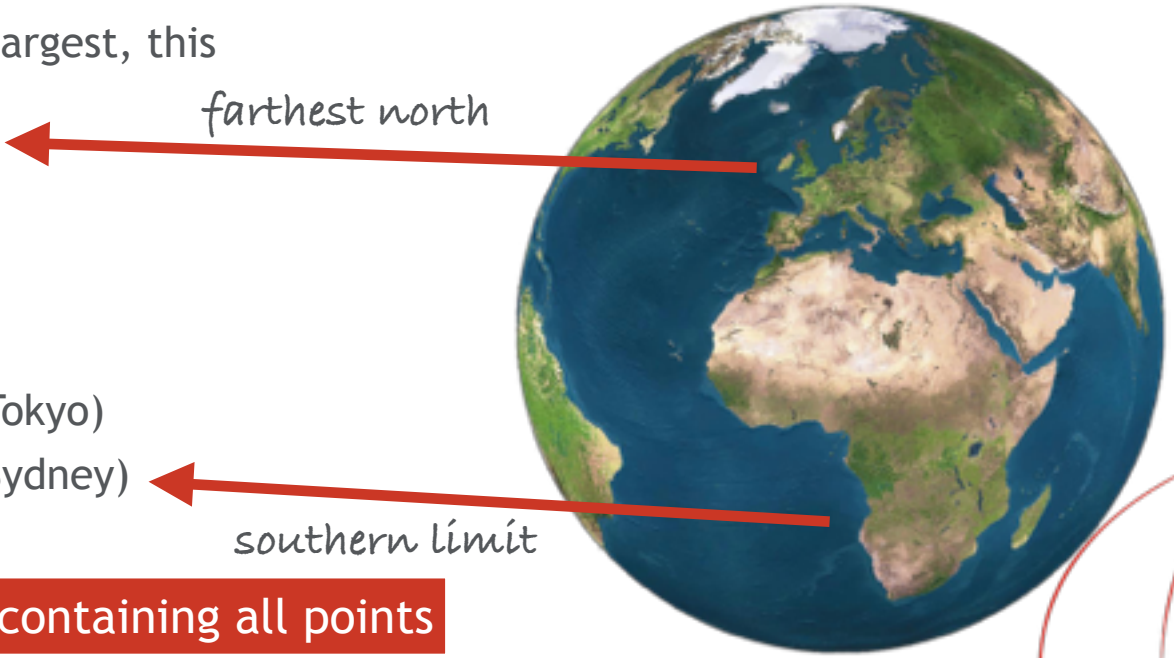
farthest north



Smallest Bounding Sphere containing all points

Biggest…

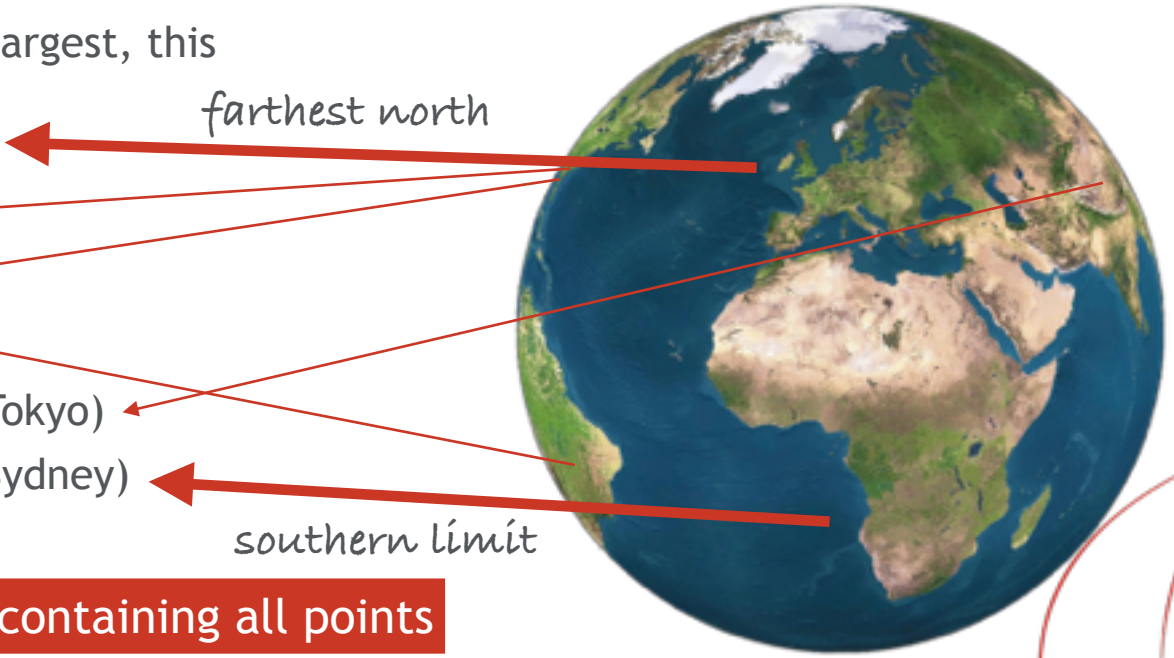
- If we want the physically largest, this is easy with AWS:
  - Master in EU (Ireland)
  - Agent in US East
  - Agent in US West
  - Agent in São Paulo
  - Agent in Asia Pacific (Tokyo)
  - Agent in Asia Pacific (Sydney)



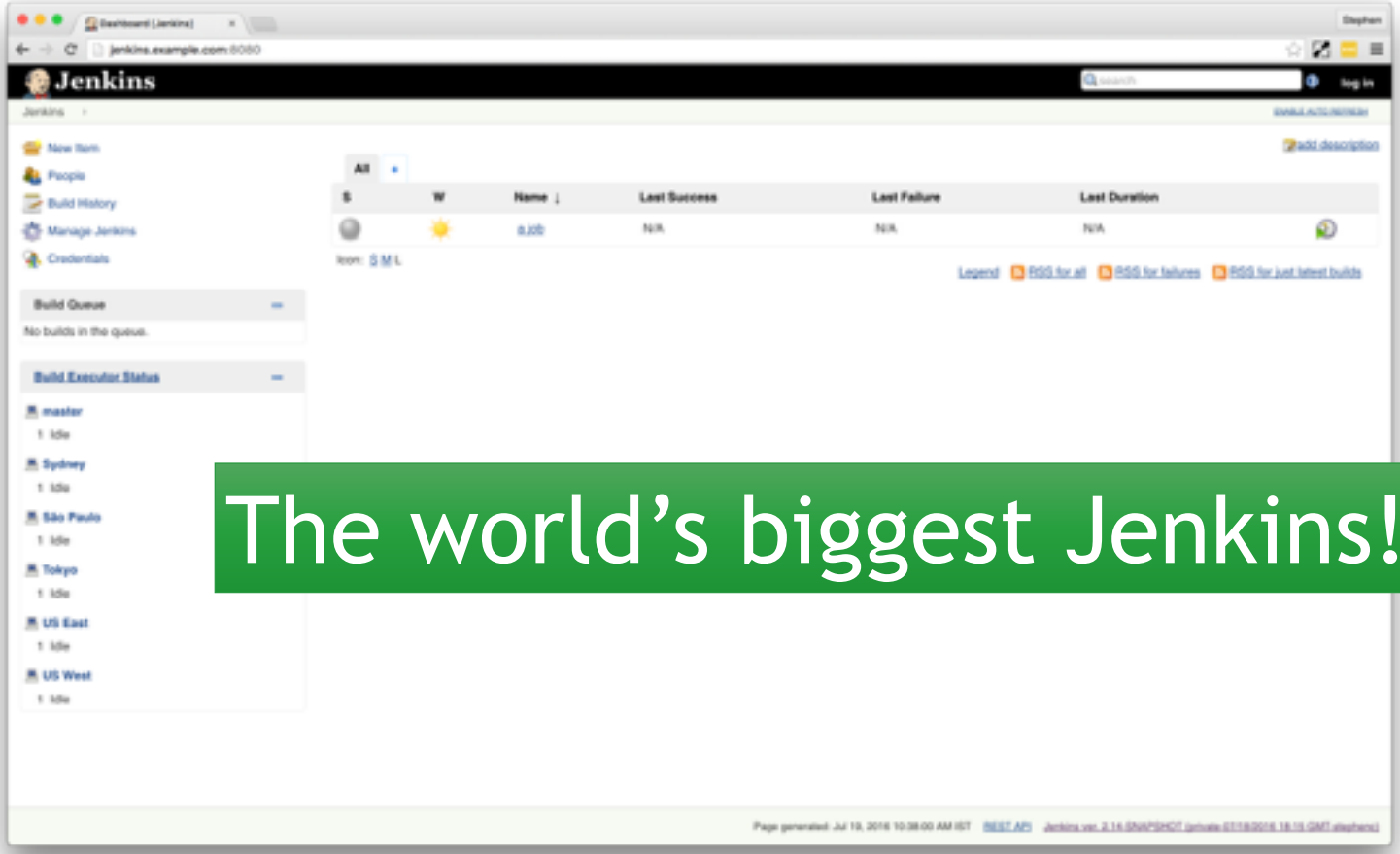
Smallest Bounding Sphere containing all points

Biggest…

- If we want the physically largest, this is easy with AWS:
  - Master in EU (Ireland)
  - Agent in US East
  - Agent in US West
  - Agent in São Paulo
  - Agent in Asia Pacific (Tokyo)
  - Agent in Asia Pacific (Sydney)



Smallest Bounding Sphere containing all points



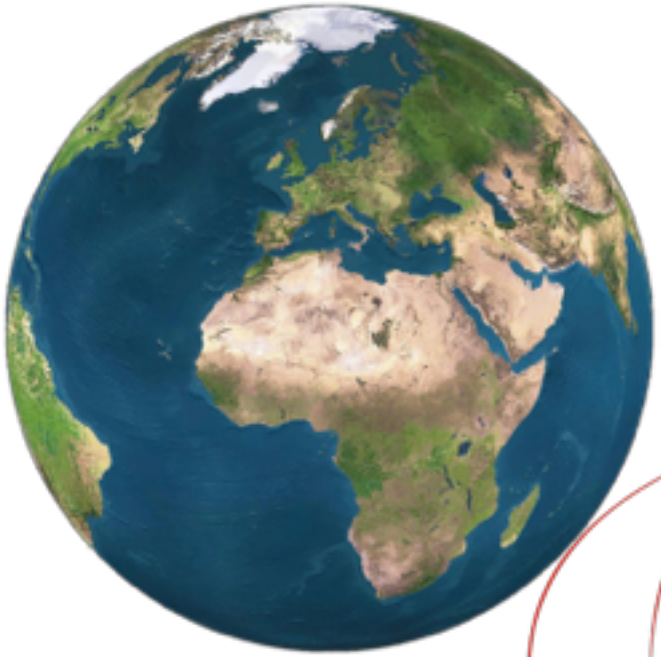
The world's biggest Jenkins!!!

Jenkins World 2016

#JenkinsWorld

Biggest…

- If we want the physically largest, this is easy with AWS:
  - Master in EU (Ireland)
  - Agent in US East
  - Agent in US West
  - Agent in São Paulo
  - Agent in Asia Pacific (Tokyo)
  - Agent in Asia Pacific (Sydney)
- If you want to beat me…
  add some height to get even bigger
  - A server in Leadville, CO?



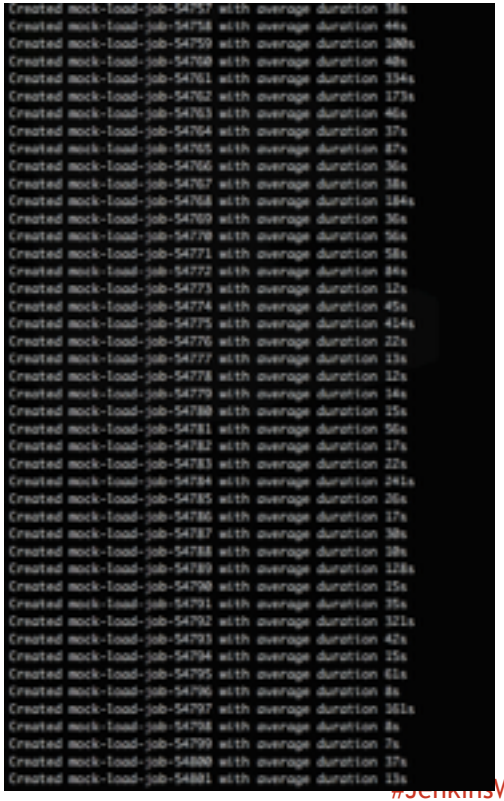
Biggest…

- If we want the most number of jobs…

Biggest…

- If we want the most number of jobs…
- Install Mock Load Builder plugin

```
java -jar jenkins-cli.jar \
 create-mock-load-jobs 200000
```

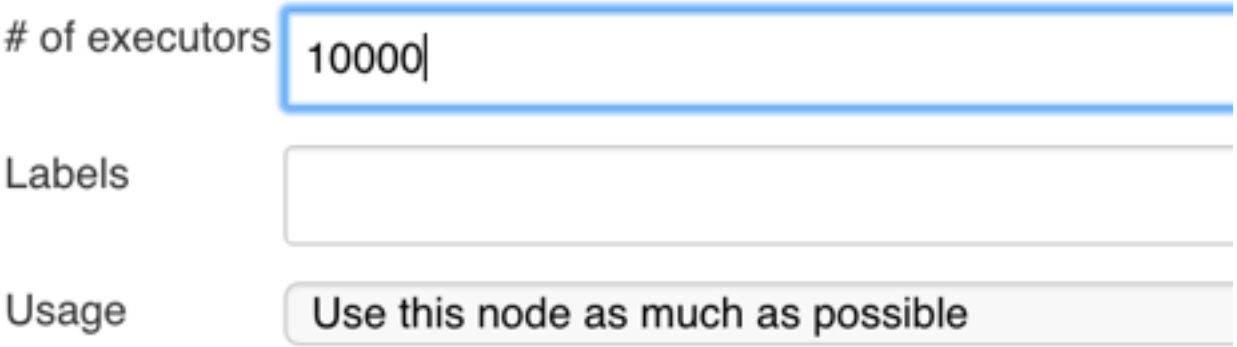


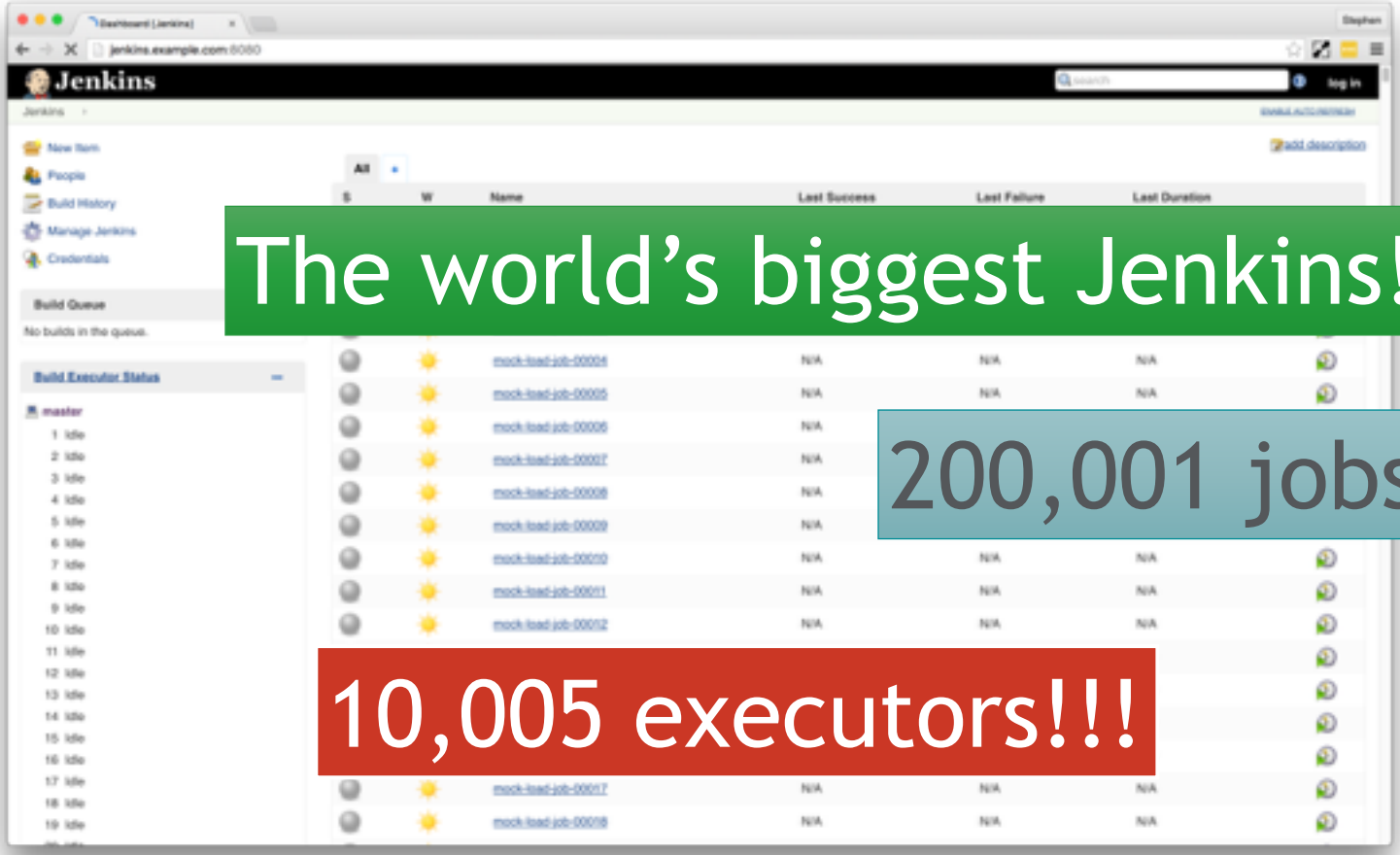

Jenkins World 2016
The world's biggest Jenkins!!!
200,001 jobs!!!
Created mock-load-job-199997 with average duration 30s
Created mock-load-job-199998 with average duration 13s
Created mock-load-job-199999 with average duration 43s
Created mock-load-job-200000 with average duration 339s
Overall average duration: 59s
Expected executor multiplier: 0.9950699166666667 x (number of builds
Current ideal max build rate: 6.0
#JenkinsWorld

I may need another of these

Biggest…

- If we want the most number of executors…





Jenkins World 2016

#JenkinsWorld


I asked for the biggest Jenkins cluster
And Stephen keeps using the wrong definitions of biggest

Cluster…

- Maybe I just run lots of Jenkins instances in parallel…

Jenkins World 2016

Please don’t make me pose for any more memes

We need some rules…

Rules

- Biggest means:

number of concurrent builds averaged over a 1h / 24h period

- Each node must be on-line for at least 90% of the 1h / 24h period
- Each executor must have an average utilisation of at least 90% for the 1h / 24h period

Rules

To be a Jenkins Cluster:

- any job in the cluster must be technically able to:
  - trigger any other job in the cluster
  - copy artifacts from any other job in the cluster
- users can navigate across the cluster

Note: *Access permissions are allowed to restrict this for real clusters*

Rules

- Jobs can be real jobs or mock load builder jobs but must:
  - output a console log (averaging at least 30 lines per minute)
  - produce JUnit style test results and archive them
  - produce build artifacts and archive them

- 20% of jobs must trigger other jobs in the cluster
- 20% of jobs must copy archived artifacts from another job in the cluster
- Job types ideally should representative of real world frequencies.

  In 2016 that means:
  - 75% freestyle, 15% maven, 5% matrix, 5% pipeline

Ok, where were we…

Agenda

Real world

Theory

Experiments

Applied


Jenkins phone home!

Usage Statistics in Jenkins

Encrypt

Decrypt

Anonymise

Statistics

#JenkinsWorld

https://wiki.jenkins-ci.org/display/JENKINS/Usage+Statistics

Usage Statistics in Jenkins

- Enabled by default, can opt out either:
  - by UI; or
  - by `-Dhudson.model.UsageStatistics.disabled=true`
- Runs once a day
- Sends encrypted payload to Jenkins OSS server via user's browser
- **All data is anonymised**

```
public boolean isDue() {
    // user opted out. no data collection.
    if(!Jenkins.getInstance().isUsageStatisticsCollected() || DISABLED)     return false;
```

Usage Statistics in Jenkins

- Records some basic information:
  - Servlet container
  - Jenkins version
  - Each defined node
    - JVM vendor and version
    - Number of executors
    - OS
- **Does not provide information on how many nodes were on-line**

```
JSONObject o = new JSONObject();
o.put("stat",1);
o.put("install", j.getLegacyInstanceId());
o.put("servletContainer", j.servletContext.getServerInfo());
o.put("version", Jenkins.VERSION);

List<JSONObject> nodes = new ArrayList<JSONObject>();
for( Computer c : j.getComputers() ) {
    JSONObject  n = new JSONObject();
    if(c.getNode()==j) {
        n.put("master",true);
        n.put("jvm-vendor", System.getProperty("java.vm.vendor"));
        n.put("jvm-name", System.getProperty("java.vm.name"));
        n.put("jvm-version", System.getProperty("java.version"));
    }
    n.put("executors",c.getNumExecutors());
    DescriptorImpl descriptor = j.getDescriptorByType(DescriptorImpl.class);
    n.put("os", descriptor.get(c));
    nodes.add(n);
}
o.put("nodes",nodes);
```

Usage Statistics in Jenkins

- As of 1st May 2016 there are 332 installations reporting at least 100 nodes

Usage Statistics in Jenkins

- As of 1st May 2016 there are 332 installations reporting at least 100 nodes
- The largest had 1,476 nodes and 2,941 executors

Usage Statistics in Jenkins

- As of 1st May 2016 there are 332 installations reporting at least 100 nodes
- The largest had 1,476 nodes and 2,941 executors
- By 1st July 2016 that instance had grown to 2,113 nodes and 4,215 executors
- By 1st September 2016 that instance had been replaced by another instance with 6,794 executors

Usage Statistics in Jenkins

- As of 1st May 2016 there are 332 installations reporting at least 100 nodes
- The largest had 1,476 nodes and 2,941 executors
- By 1st July 2016 that instance had grown to 2,113 nodes and 4,215 executors
- By 1st September 2016 that instance had been replaced by another instance with 6,794 nodes and 6,794 executors

6,794 nodes
6,794 executors

Usage Statistics in Jenkins

- As of 1st May 2016 there are 332 installations reporting at least 100 nodes
- The largest had 1,476 nodes and 2,941 executors
- By 1st July 2016 that instance had grown to 2,113 nodes and 4,215 executors
- By 1st September 2016 that instance had been replaced by another instance with 6,793 nodes and 6,794 executors

6,794 nodes
6,794 executors

Does not tell us how many on-line

Customer Support bundles

CloudBees support bundles

- At CloudBees, our support team regularly ask for support bundles
- Bundles are generated by the OSS Support Core plugin
- We anonymise some of the information and use for deeper analysis

CloudBees

Health warning

- Most of these bundles are from systems where customers had a requirement for support

Health warning

- Most of these bundles are from systems where customers had a requirement for support
  - We ask for support bundles routinely, so some are from customers who just had questions

Health warning

- Most of these bundles are from systems where customers had a requirement for support
  - We ask for support bundles routinely, so some are from customers who just had questions
- Expect short uptime

"It's not working, let's try kicking it up the arse before we call for support…"

Health warning

- Most of these bundles are from systems where customers had a requirement for support
  - We ask for support bundles routinely, so some are from customers who just had questions
- Expect short uptime
- Expect low usage

> “It’s not working, let’s not push it too hard until we get it fixed again…”

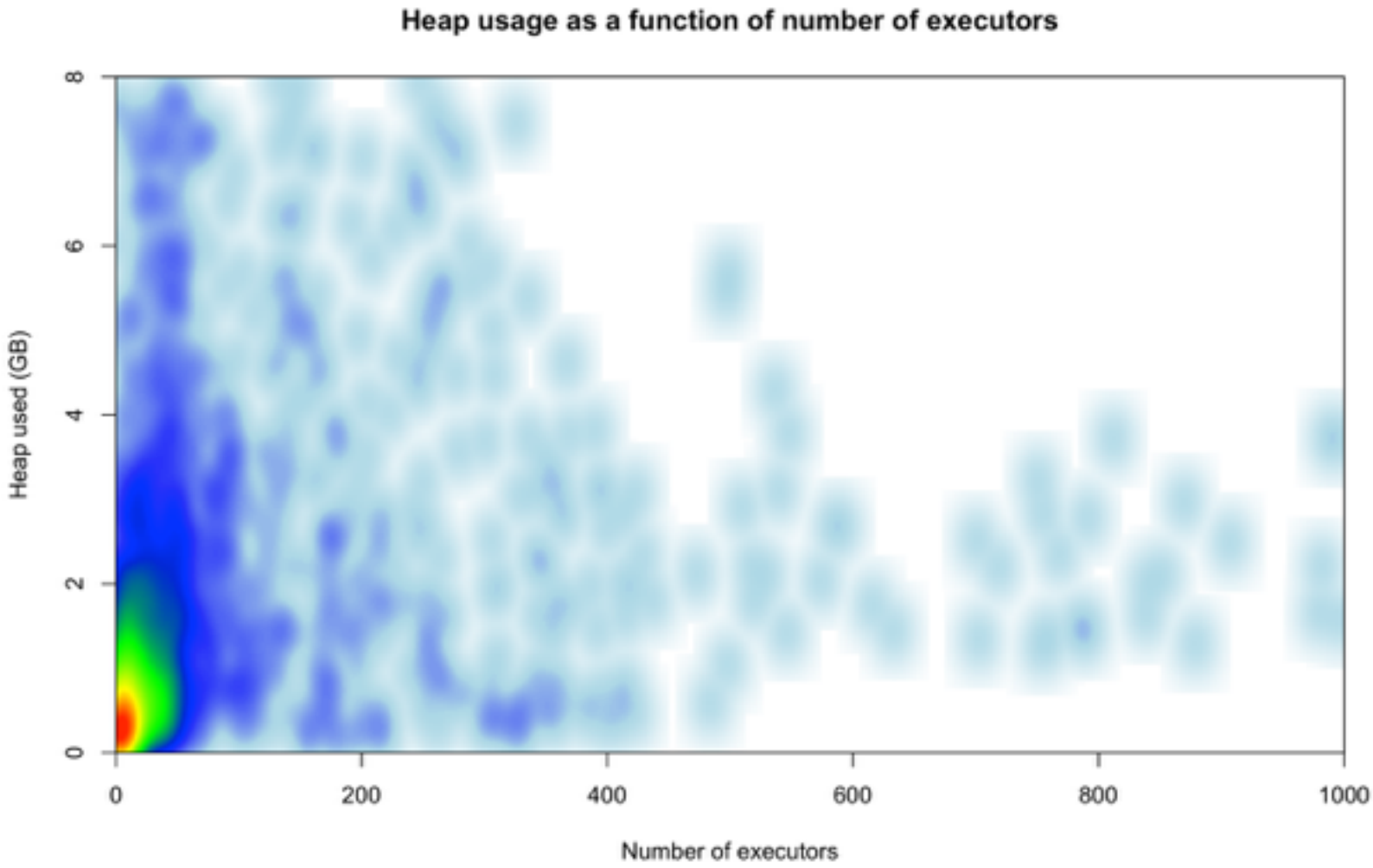
Health warning

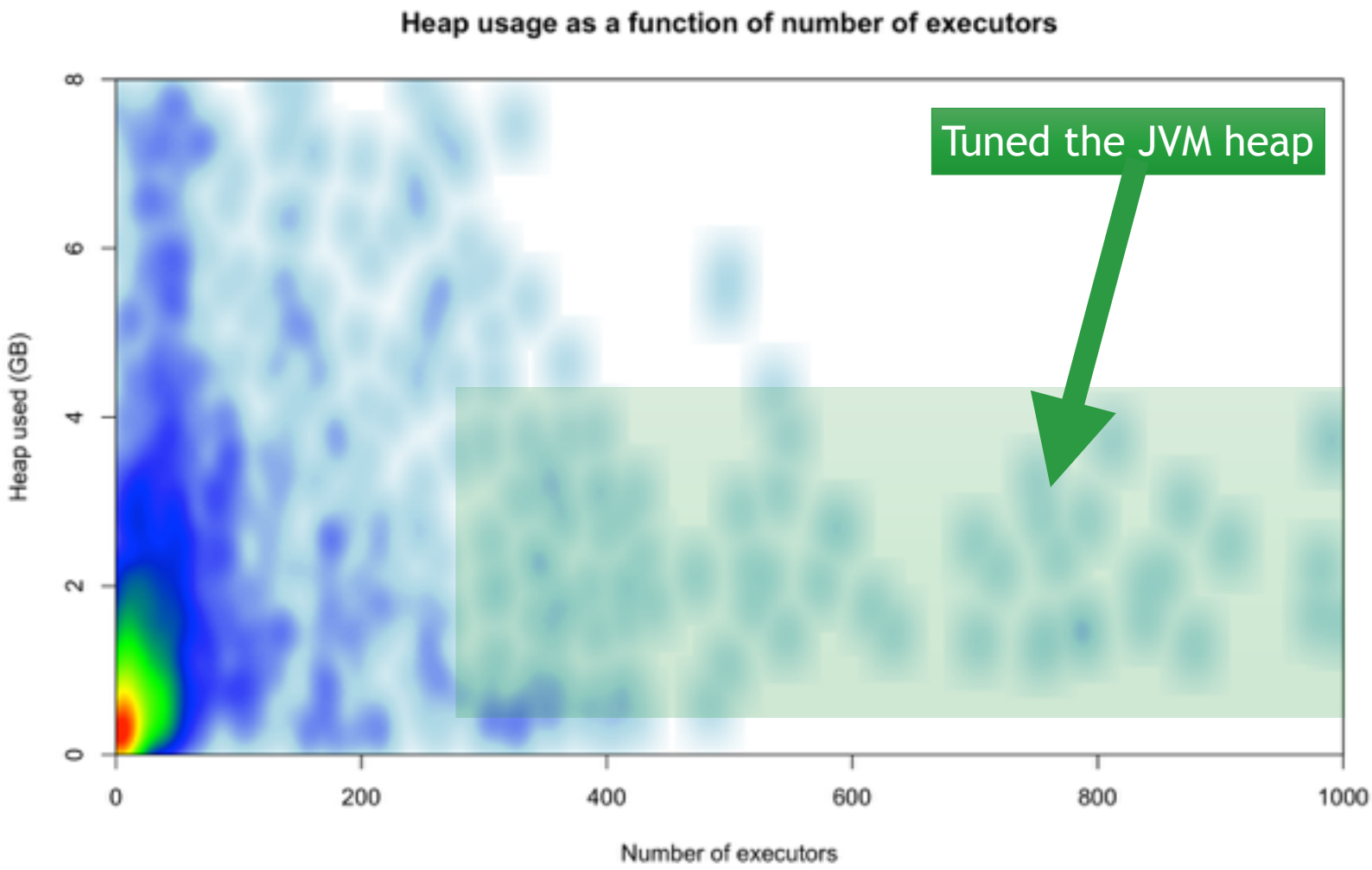
- Most of these bundles are from systems where customers had a requirement for support
  - We ask for support bundles routinely, so some are from customers who just had questions
- Expect short uptime
- Expect low usage
- Customers can censor how much of the bundle to send us

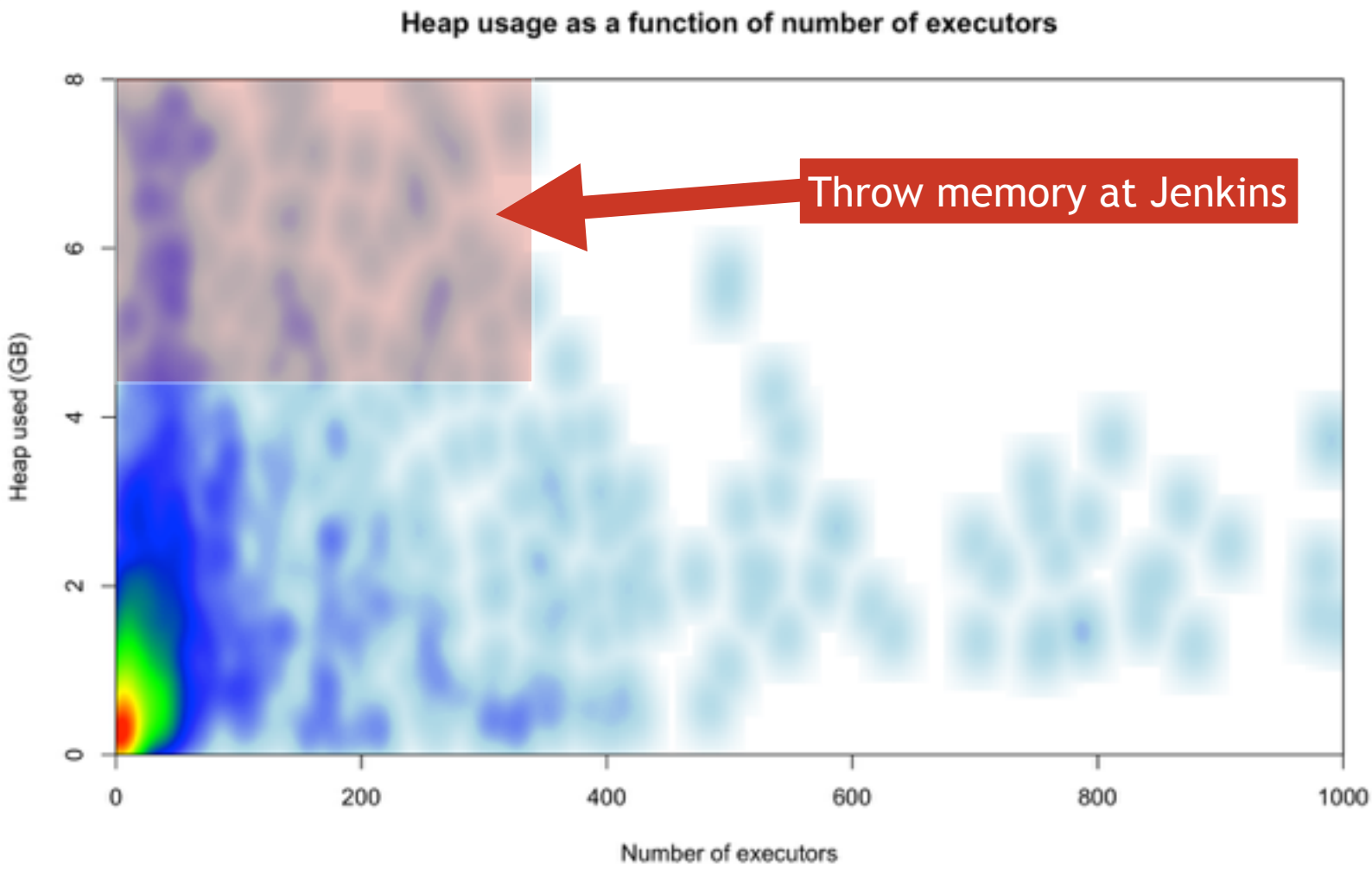
The bits we are interested in have been provided by >98% of our customers

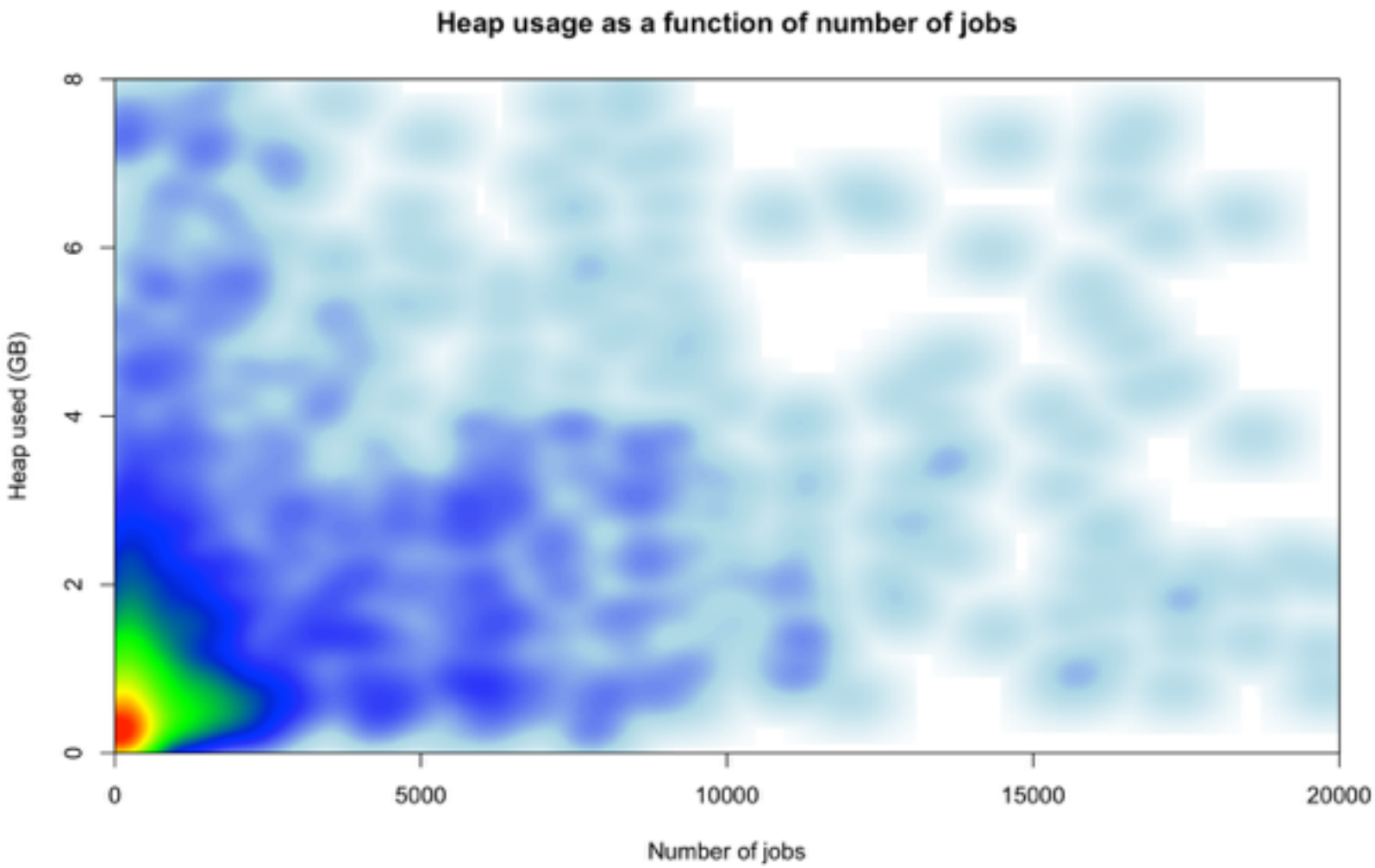
Positives

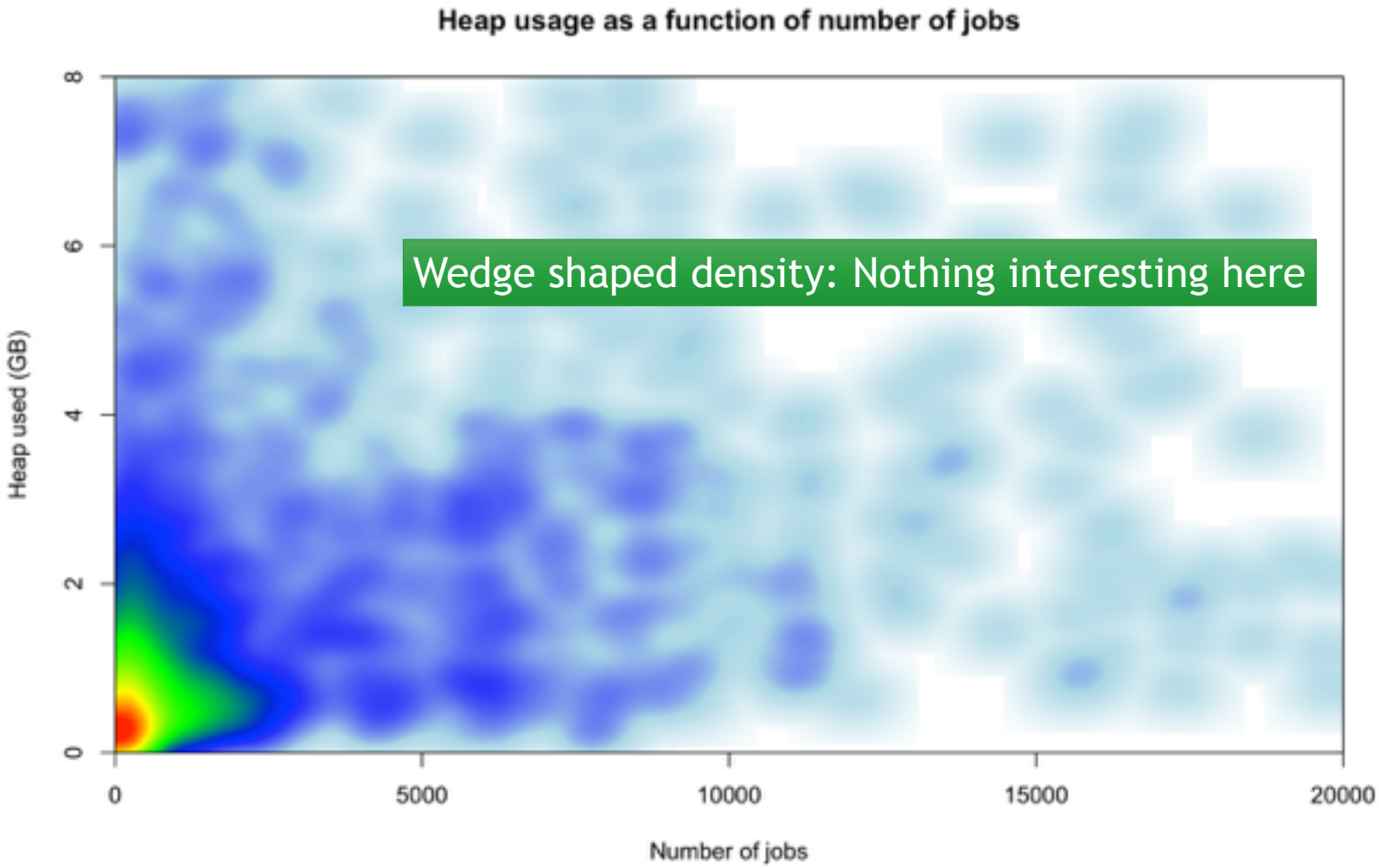
- Includes
  - # of nodes on-line
  - JVM tuning arguments

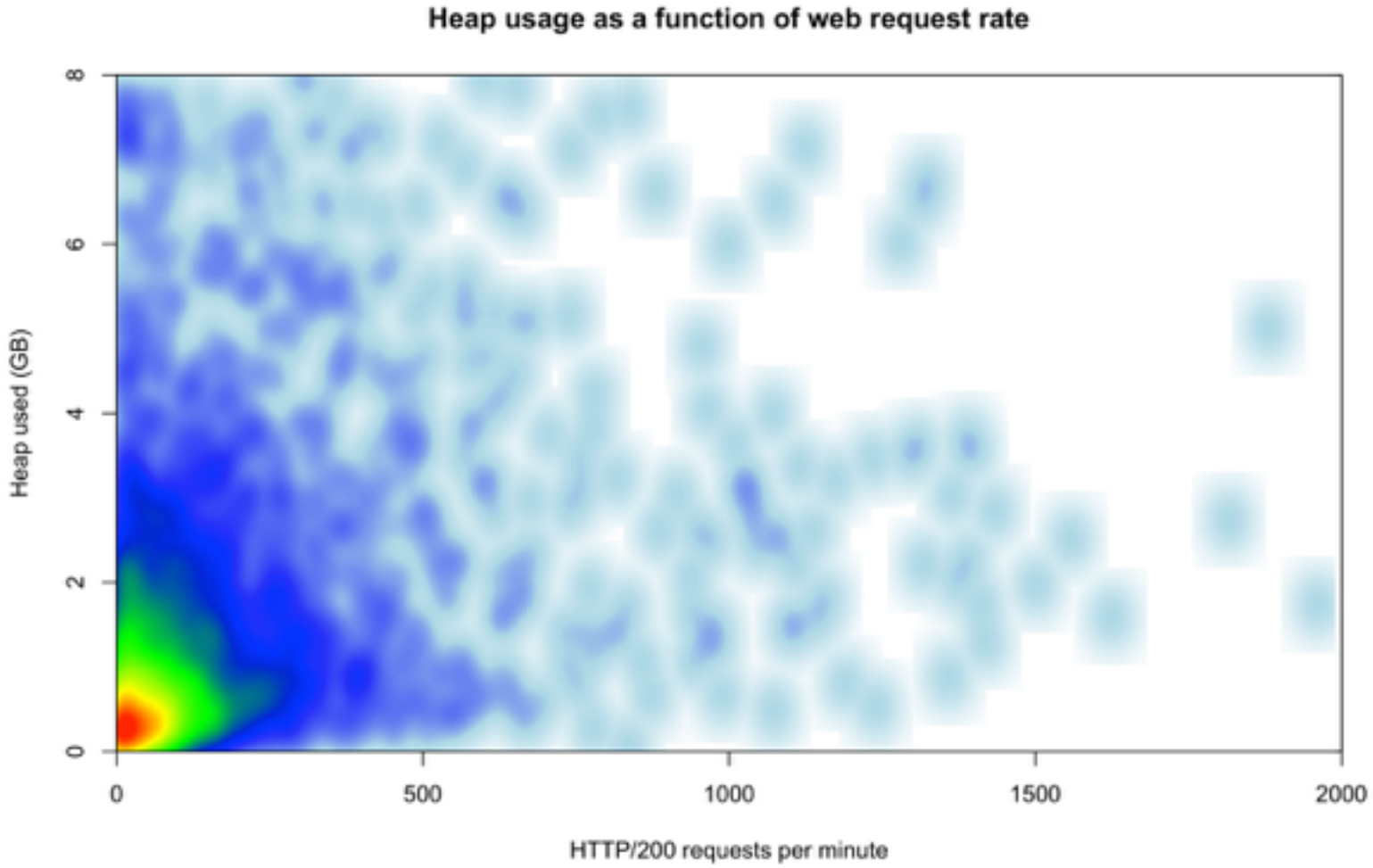

Heap usage as a function of number of executors
Heap used (GB)
0
2
4
6
8
0
200
400
600
800
1000
Number of executors

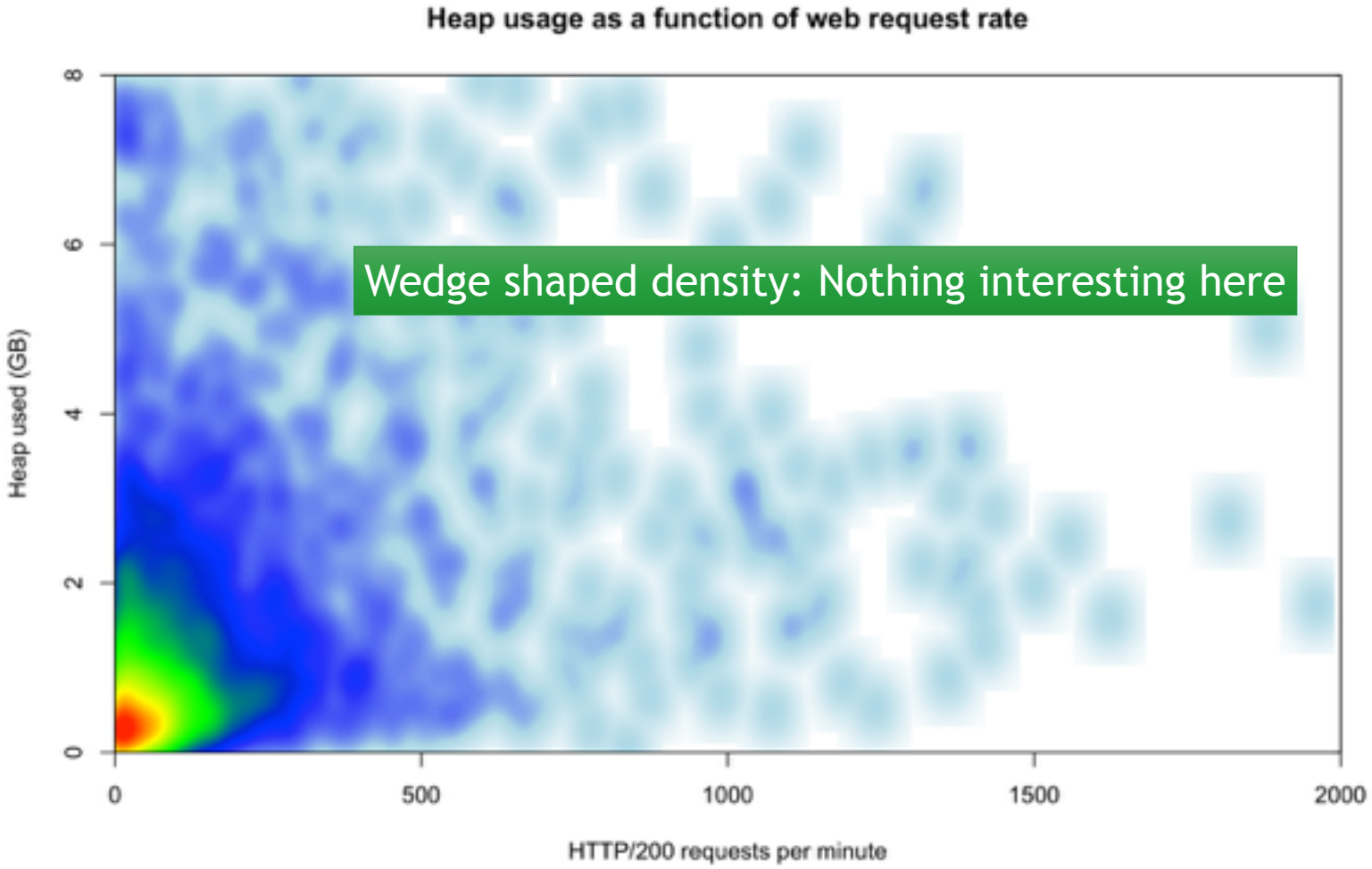

Heap usage as a function of number of executors
Tuned the JVM heap
Heap used (GB)
0
2
4
6
8
0
200
400
600
800
1000
Number of executors


Heap usage as a function of number of executors
Throw memory at Jenkins
Heap used (GB)
0
2
4
6
8
0
200
400
600
800
1000
Number of executors


Heap usage as a function of number of jobs
Heap used (GB)
0
2
4
6
8
0
5000
10000
15000
20000
Number of jobs


Heap usage as a function of number of jobs
Heap used (GB)
0
2
4
6
8
0
5000
10000
15000
20000
Number of jobs
Wedge shaped density: Nothing interesting here


Heap usage as a function of web request rate
Heap used (GB)
0
2
4
6
8
0
500
1000
1500
2000
HTTP/200 requests per minute


Heap usage as a function of web request rate
Heap used (GB)
0
2
4
6
8
0
500
1000
1500
2000
HTTP/200 requests per minute
Wedge shaped density: Nothing interesting here

Largest

- Support bundle taken after 40h of uptime
  - 1,104 nodes configured
  - 698 nodes on-line
  - 7,135 executors configured
  - 4,828 executors on-line
  - 5,784 jobs (92% freestyle)

- Master: Java 8

```
-Xms4096m -Xmx4096m
-XX:NewSize=2048m -XX:MaxNewSize=2048m
-XX:ParallelGCThreads=4 -XX:ConcGCThreads=4
-Dhudson.slaves.ChannelPinger.pingInterval=-1
```

- Agents running Java 7/8:
  - Majority SSH:

```
-Xms128m -Xmx512m
-Dhudson.remoting.Launcher.pingTimeoutSec=600
-Dhudson.remoting.Launcher.pingIntervalSec=1200
```

  - 5% SSH:

```
-Xms128m -Xmx512m
-Dhudson.remoting.Launcher.pingIntervalSec=-1
```

  - 1% JNLP (mostly Windows)

Root cause of support request:
NFS timeouts causing agent disconnects

Second largest

- Support bundle taken after 18 minutes uptime
  - 326 nodes configured
  - 313 nodes online
  - 1,382 executors online
  - 1,399 executors configured
  - 7,724 jobs

- Master: Java 7

  `-Xms4096m -Xmx4096m`

  `-XX:NewSize=200m -XX:MaxNewSize=200m`
- Agents running Java 7/8:
  - 70% SSH:

    `Default JVM Options`
  - 30% JNLP:

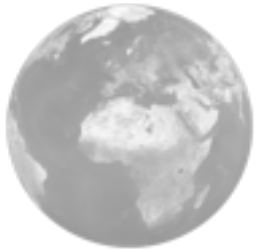
    `Default JVM Options`

Root cause of support request:
Build history widget rendering blocked due to synchronisation bug in customer's own custom plugin

Summary (real world)

- Probably going to be ok with 4GB heap to start
  - Grow from there to meet targets for jobs / builds / concurrent users
- Ping threads initiated by the master may cause issues
- Tuning the Agent JVMs is probably not a priority issue
- Getting 2,000+ nodes per Jenkins instance is possible
  - Keeping them all on-line concurrently may be a separate issue 😉

Agenda



Real world

Theory

Experiments

Applied

Why scale Jenkins?

- A single instance can handle 100k+ jobs
  - Put it on a big box
  - Organise jobs with folders
  - Use SSD for storage
  - Schedule restarts for weekends

200 executors

"~~640K~~ ^ ought to be enough for anybody"

—Bill Gates

Joel on Software

http://goo.gl/mEcQpB

Human Task Switches Considered Harmful

by Joel Spolsky

Monday, February 12, 2001

When you're managing a team of programmers, one of the first things you have to learn to get right is task allocation. That's just a five-dollar word for *giving people things to do*. It's known colloquially as "file

Joel on Software

http://goo.gl/mEcQpB

Human Task Switches Considered Harmful

by Joel Spolsky

Monday, February 12, 2001

When you're managing a team of programmers, one of
you have to learn to get right is task allocation. That's j
word for *giving people things to do*. It's known colloqu

CODING HORROR

programming and human factors

http://goo.gl/6VSjaz

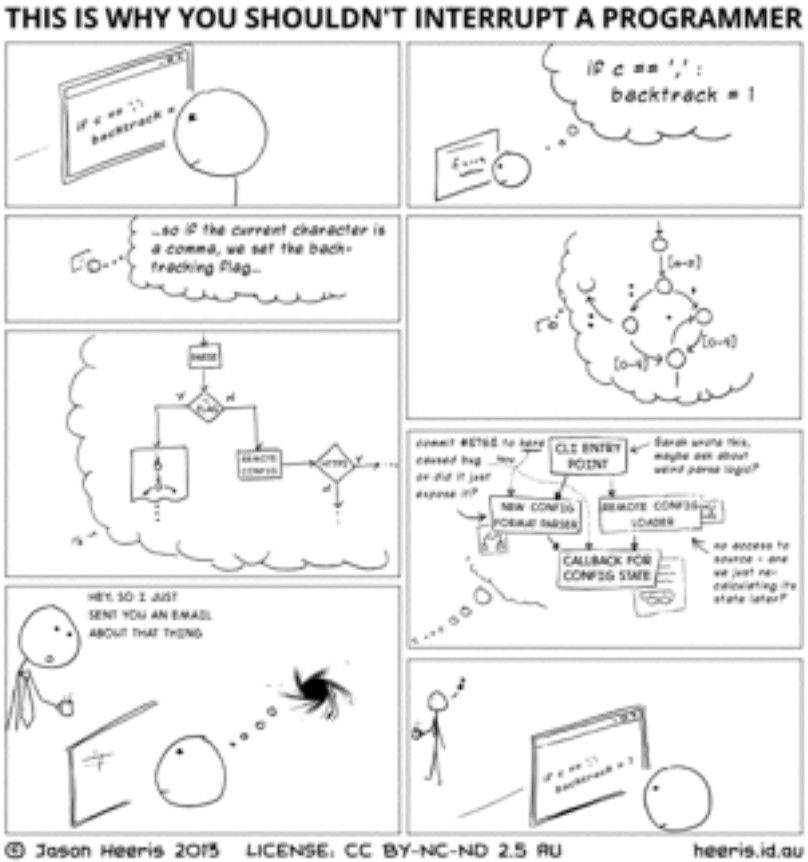
27 Sep 2006

The Multitasking Myth

In Quality Software Management: Systems Thinking, Gerald Weinberg proposed a rule of thumb to calculate the waste caused by project switching:


Joel on Software
http://goo.gl/mEcQpB
Human Task Switches Considered Harmful
by Joel Spolsky
Monday, February 12, 2001
When you're managing a team of programmers, one of
CODING HORROR
programming and human factors
http://goo.gl/6VSjaz
27 Sep 2006
The Multitasking Myth
ems Thinking, Gerald Weinberg proposed a rule of
y project switching:
Working Time Available Per Project
Loss to Context Switching
20
http://goo.gl/ljLjNZ
Susan Weinschenk Ph.D.
Brain Wise
The True Cost Of Multi-Tasking
You could be losing up to 40% of your productivity

How much does a context switch cost a developer



Jenkins World 2016

#JenkinsWorld

How much does a context switch cost a developer

Estimates vary…

How much does a context switch cost a developer

Estimates vary…

“60 minutes”

How much does a context switch cost a developer

Estimates vary…

“30 minutes”

“60 minutes”

How much does a context switch cost a developer

Estimates vary…

“30 minutes”

“45 minutes”

“60 minutes”

How much does a context switch cost a developer

Estimates vary…

“30 minutes”

“45 minutes”

“60 minutes”

My random sampling of the web and personal experience puts it at ~45 minutes

You can pay twice for delayed builds…

- Developer commits code
- Developer starts new task
- Developer gets failed build notification
- **Pay context switch to return to previous task**
- Fix build
- **Pay context switch to return to interrupted task**
- Continue interrupted task

How?

Little’s law

- Little's Law tells us that the average number of customers in the store L, is the effective arrival rate λ, times the average time that a customer spends in the store W

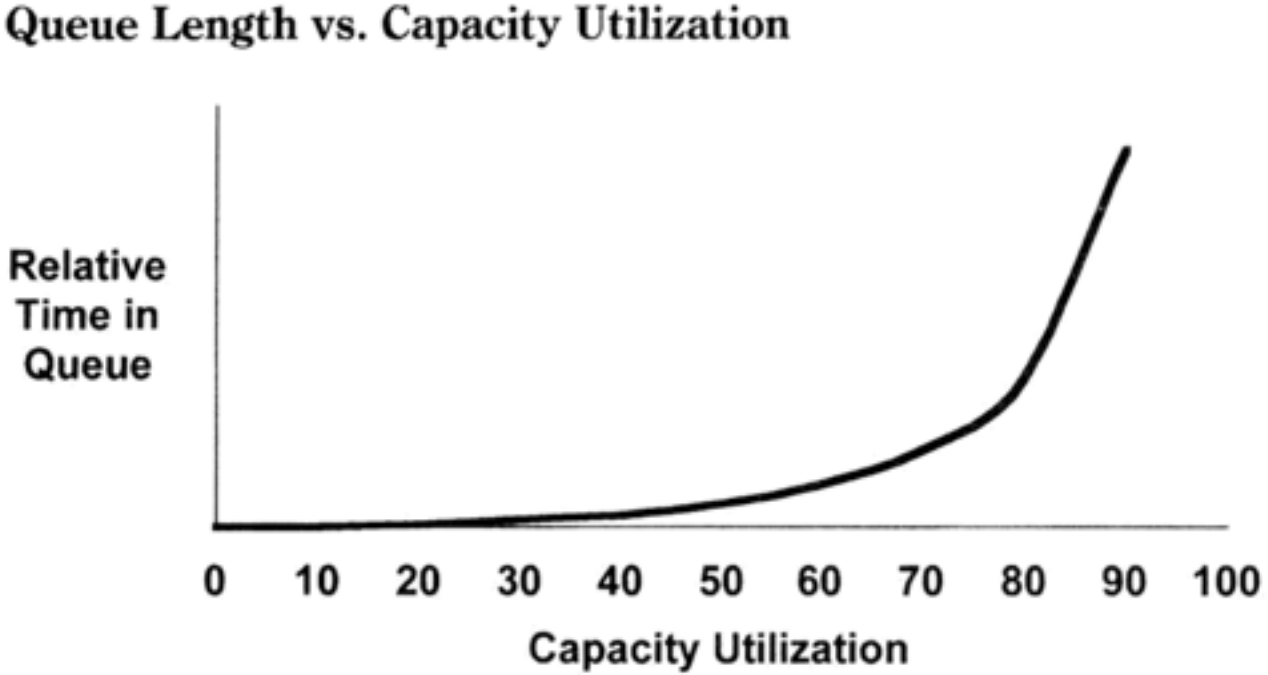
$$L = \lambda \times W$$

- Little's Law tells us that the average number of builds in the queue L, is the effective arrival rate λ, times the sum of the average time that a job spends waiting in the queue Q and the average time that a job spends building B

$$L = \lambda \times (Q + B)$$

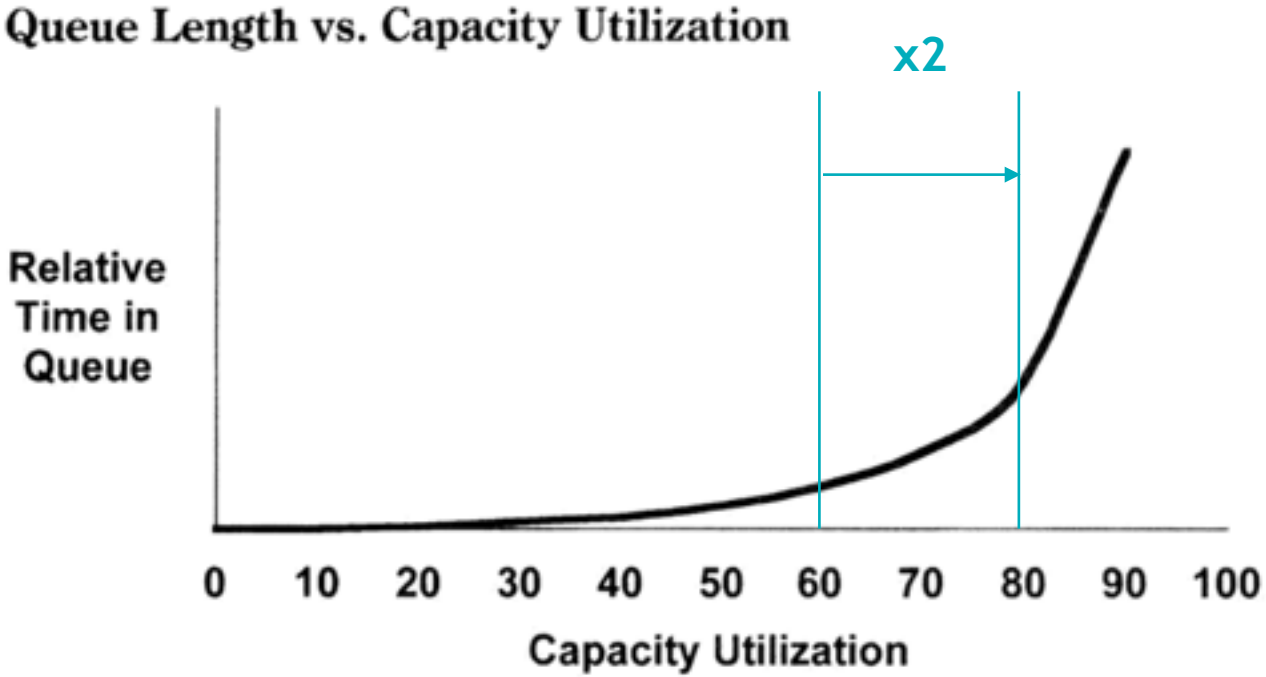
As a Jenkins administrator we cannot change B but we can minimise Q

Time in Queue



http://goo.gl/8MAosu

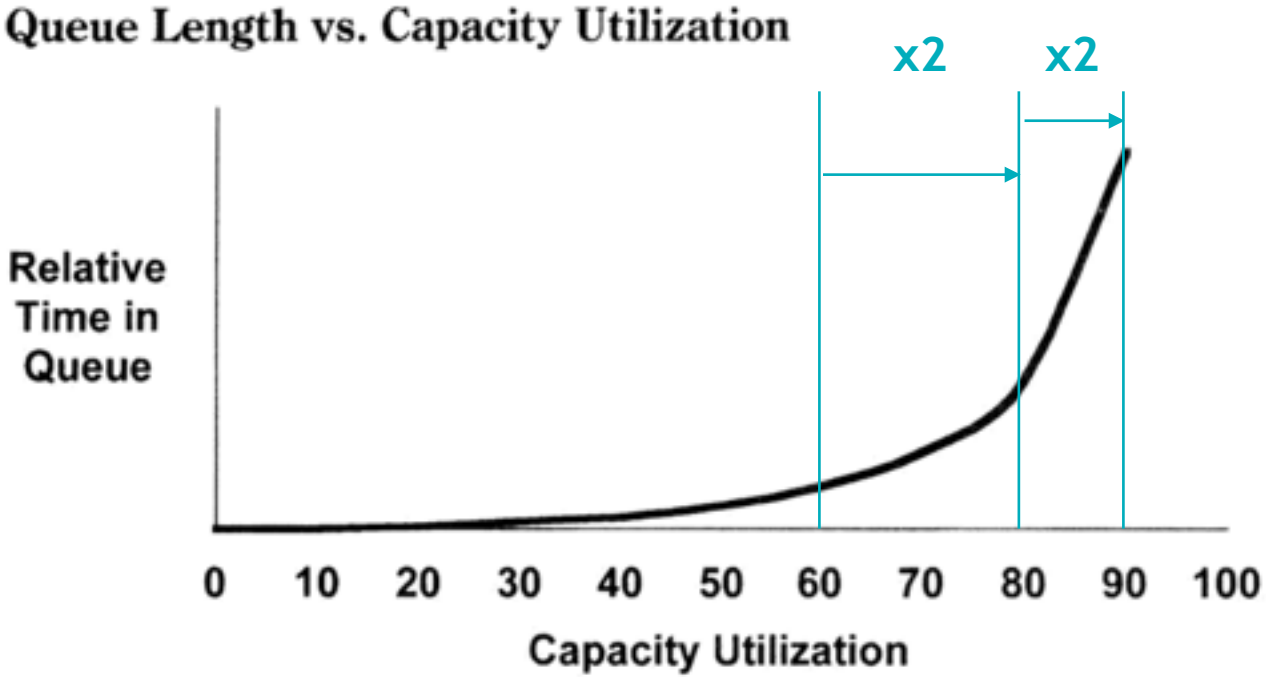
Time in Queue



Note: Assumes M/M/1/∞ Queue

http://goo.gl/8MAosu

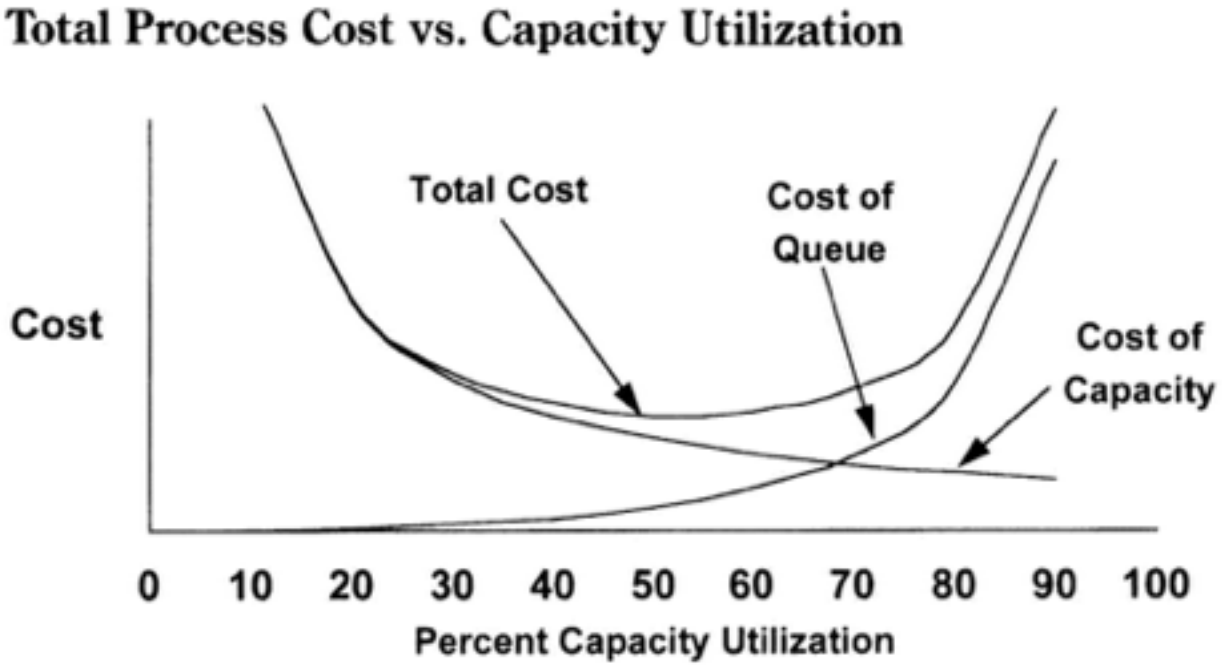
Time in Queue



Note: Assumes M/M/1/∞ Queue

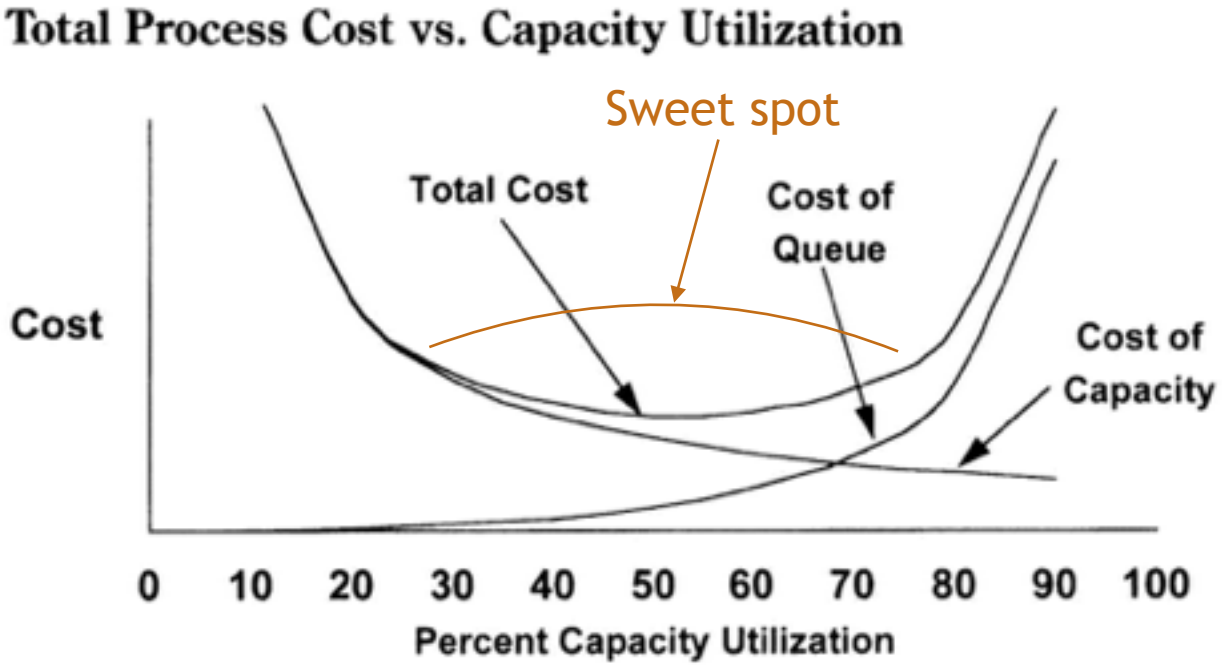
http://goo.gl/8MAosu

Time in Queue



http://goo.gl/8MAosu

Time in Queue



http://goo.gl/8MAosu

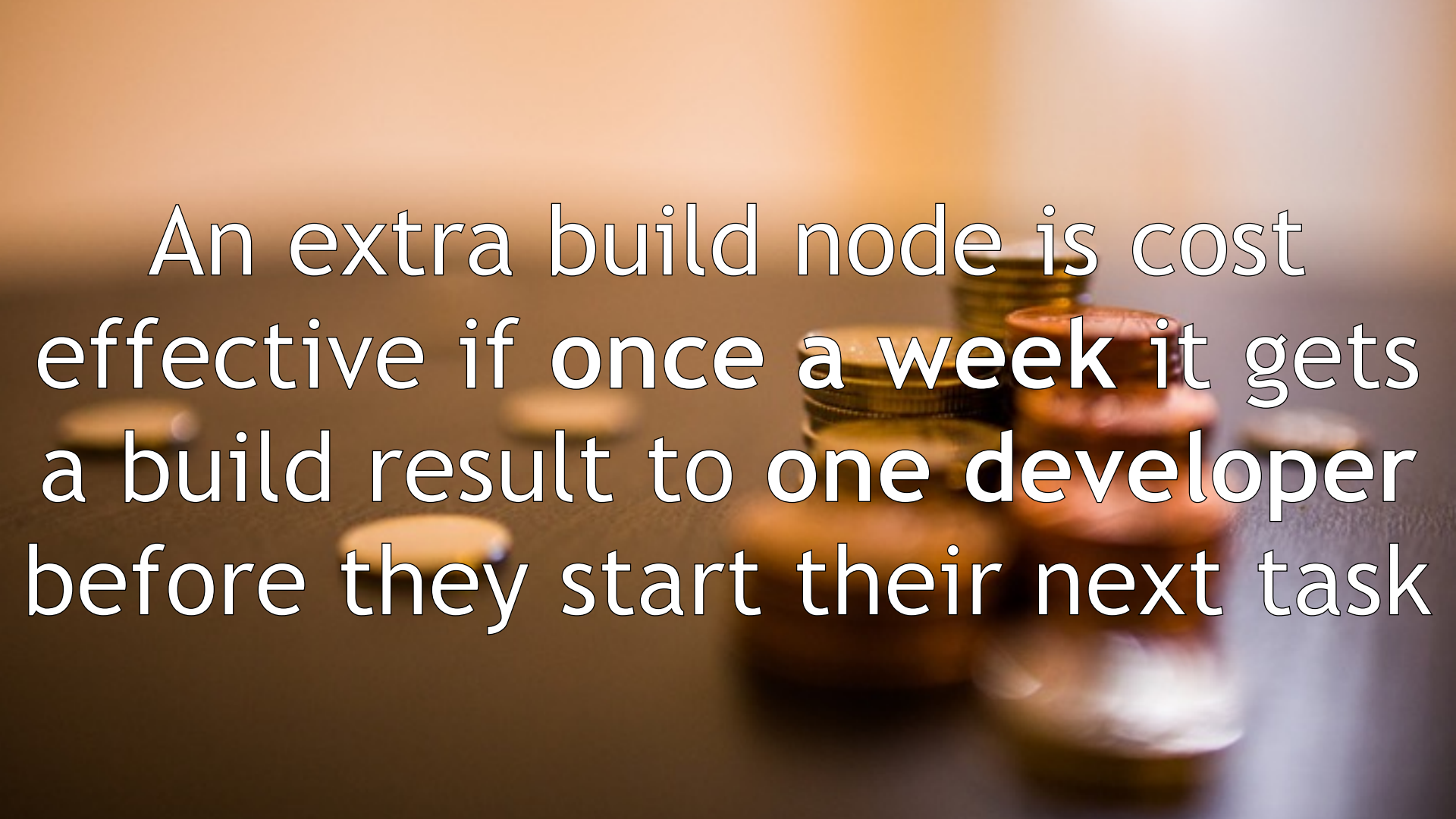
Traditional process optimisation for sweet spot

Time in Queue

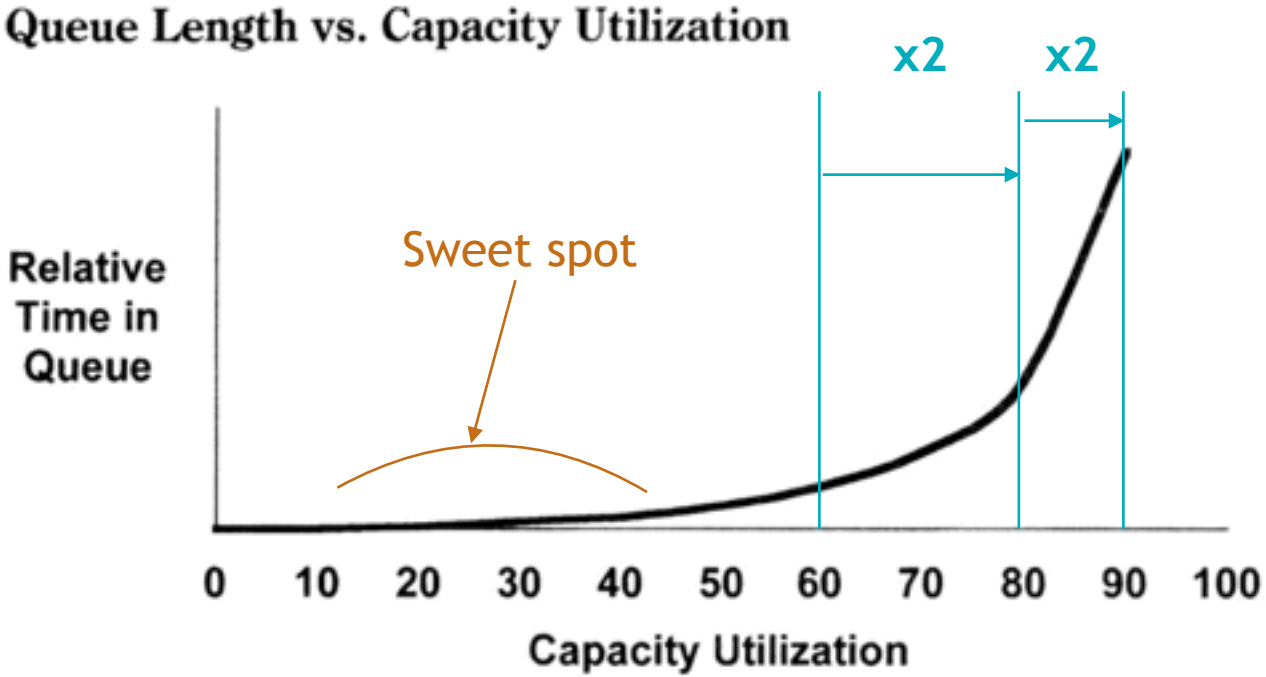
- What is the cost in capacity for a build node?
  - Includes the cost of the server resources.
  - Correct for the developer time saved by faster response.

- For cloud resources:
  - 1 high spec machine is less than $4k/year
  - Average Senior Developer in CA is $100k/year
  - Only need to save 4% of one senior developers time and the resource cloud resource has paid for itself

An extra build node is cost effective if it saves one developer 2 hours/week


An extra build node is cost effective if **once a week** it gets a build result to **one developer** before they start their next task

Time in Queue

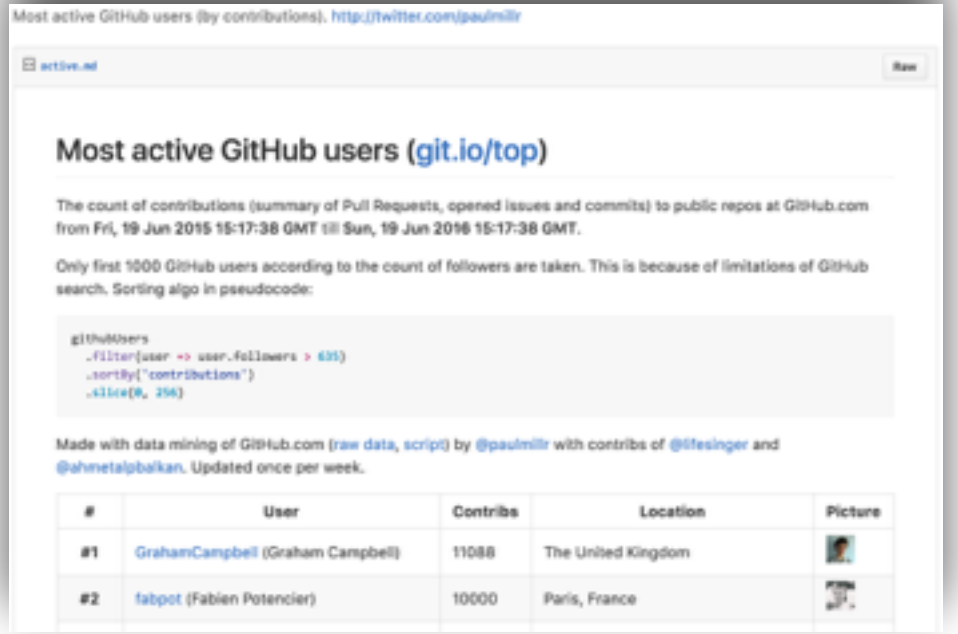


http://goo.gl/8MAosu

Build server process optimisation for sweet spot

Where are these builds coming from?

How often do developers commit?



http://git.io/top

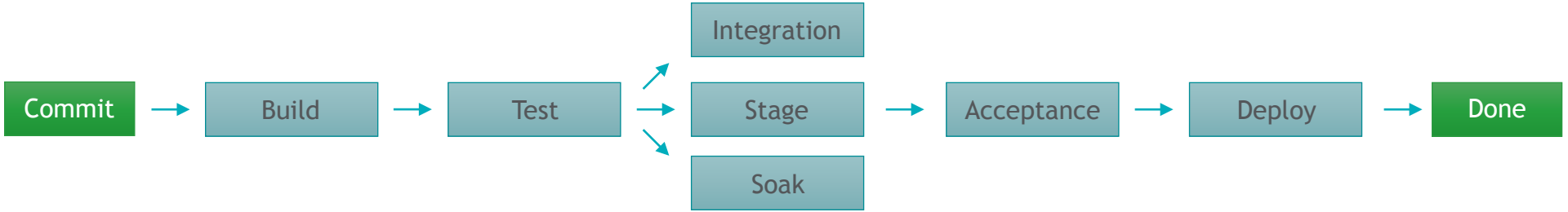
The most active developers:
- 10,000 commits per year

Assume 250 working days per year:
- 40 commits per day

5 commits per developer per hour

How long is your build chain?

or



?

5 commits per developer per hour

How big should my Jenkins be?

- Target 30-50% utilisation to maximise developer productivity
- Estimate developer activity that triggers builds
  - The upper limit is probably 5 commits per hour

Most developers are not Graham Campbell ->

- Estimate number of builds triggered by a commit
- Estimate build duration
- Apply Little's law => Number of executors

How big should my Jenkins be?

For **average developers**

How big should my Jenkins be?

For **average developers**

(not all Graham Campbell)

How big should my Jenkins be?

For **average developers**

Basic **CI** usage

How big should my Jenkins be?

For **average developers**

Basic **CI** usage

- Build and test only
- No deployment
- No DevOps build chains
- No CD

How big should my Jenkins be?

For **average developers**

With real world **test mix**

Basic **CI** usage

How big should my Jenkins be?

For **average developers**

Basic **CI** usage

With real world **test mix**

- Some unit tests
- Integration tests that use a DB
- Tests that require complex set-up and tear down
- Browser based tests

How big should my Jenkins be?

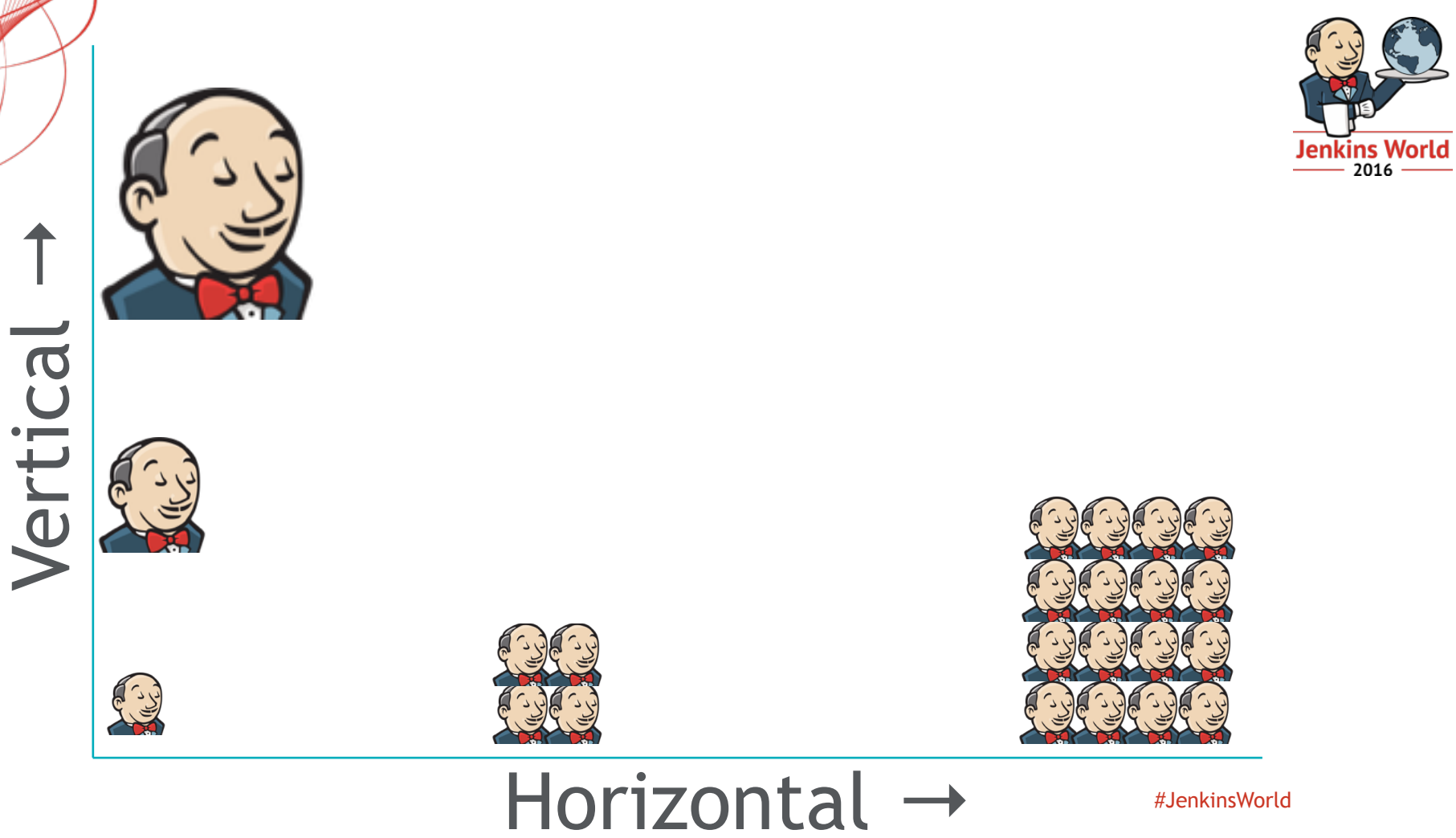
For **average developers**

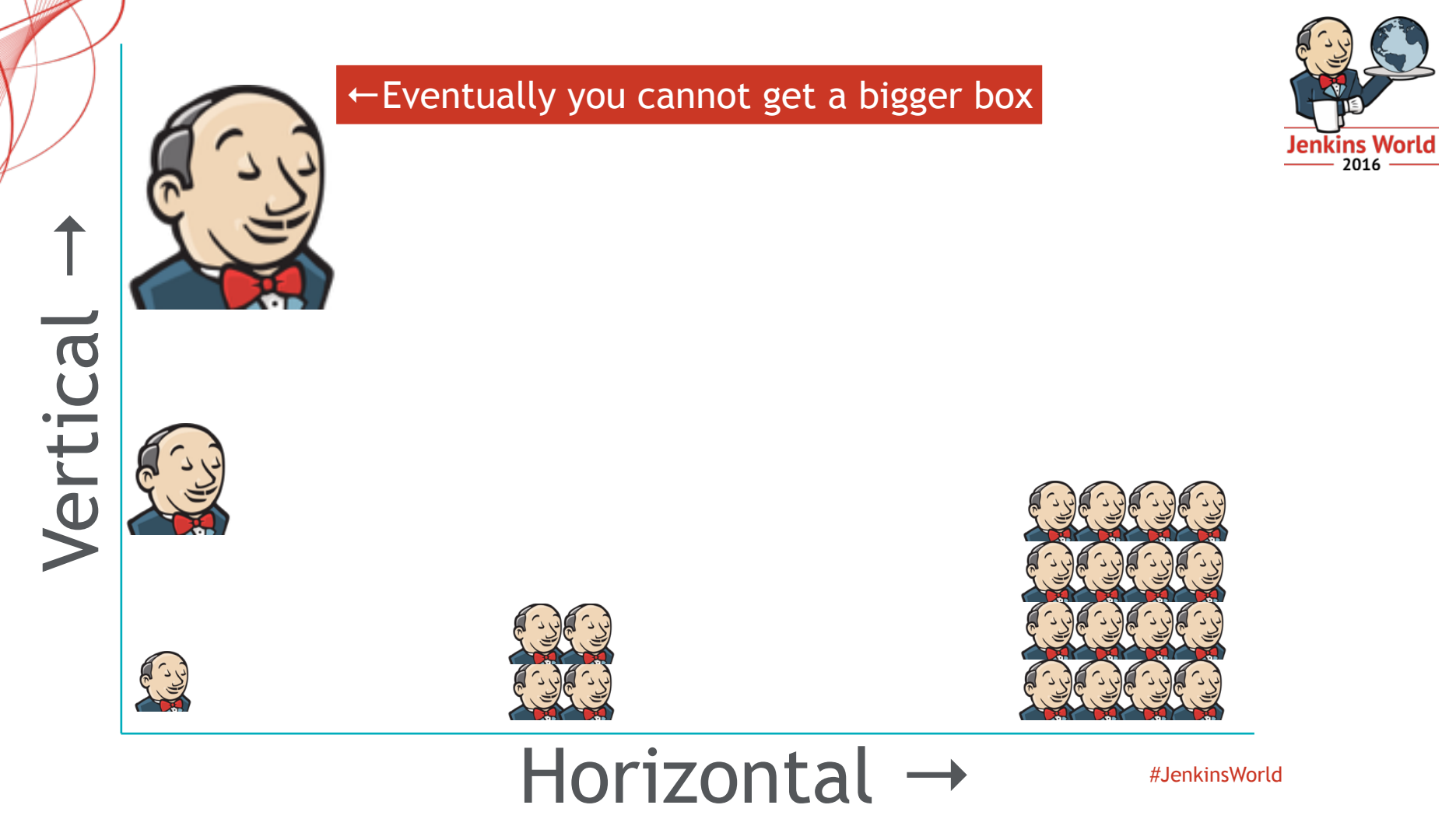
With real world **test mix**

Basic **CI** usage

At least 1 Jenkins master per 200 developers
(to maximise developer productivity)

"The Interlace, Singapore" by Mike Cartmell http://goo.gl/zDzGLs CC BY 2.0


Vertical →
Horizontal →

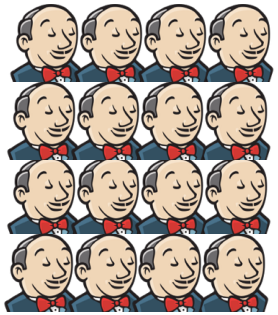


Jenkins World 2016

#JenkinsWorld

←Eventually you cannot get a bigger box

Have to use horizontal scaling anyway
↓



Vertical →

Horizontal →

Jenkins World 2016

#JenkinsWorld

Other approaches to scaling…

Other approaches to scaling…

- Microservices

Other approaches to scaling…

- **Microservices**
  - Split a single application into a suite of small services
  - Running in separate processes
  - Independently deployable
  - Scale each service according to requirements

Other approaches to scaling…

- **Microservices**
  - Split a single application into a suite of small services
  - Running in separate processes
  - Independently deployable
  - Scale each service according to requirements
- **Serverless**

Other approaches to scaling…

- **Microservices**
  - Split a single application into a suite of small services
  - Running in separate processes
  - Independently deployable
  - Scale each service according to requirements
- **Serverless**
  - Instead of supporting a specific application, clusters of servers provide a generic execution environment for any number of applications.
  - Run in stateless compute containers that are *event-triggered*, *ephemeral*, and fully managed by a 3rd party


What if I told you…


What if I told you…
…Jenkins jobs are
serverless microservices?

“Jenkins is cron on steroids”

— Lindsay Holmwood

http://goo.gl/0FjInB

Microservices: Real Architectural Patterns
- Camille Fournier

Cron Jobs as Microservices

… When it becomes very easy to make anything a microservice, everything becomes a microservice, including things we would traditionally run as cron jobs.

http://goo.gl/Nqj4io

Microservices: Real Architectural Patterns
- Camille Fournier

Cron Jobs as Microservices

… When it becomes very easy to make anything a microservice, everything becomes a microservice, including things we would traditionally run as *cron jobs*.

http://goo.gl/Nqj4io

Microservices

In short, the microservice architectural style is an approach to developing a single application as a **suite of small services**, each **running in its own process** and communicating with lightweight mechanisms, often an HTTP resource API. These services are **built around business capabilities** and **independently deployable** by fully automated deployment machinery. There is a **bare minimum of centralized management** of these services, which may be written in different programming languages and use different data storage technologies.

-- James Lewis and Martin Fowler

Microservice definition

- Small service - fine grained to perform a single function
- Runs in own process
- Built around business capability
- Independently deployable
- Bare minimum of centralised management

Jenkins Jobs are Microservices

- Small service ✅ => it just checks out and builds code
- Runs in own process ✅ => each job runs on its own executor
- Built around business capability ✅ => each job is targeted for a specific project
- Independently deployable ✅ => we can reconfigure individual jobs any time
- Bare minimum of centralised management ✅ => job sprawl is a real problem 😉

Pros and cons of Microservices

- Pros
  - Independence of services
  - Focus on business capabilities of apps
  - Simplicity of adding new features
  - Fault tolerance

- Cons
  - Implicit interfaces
  - Operational overhead
  - Require DevOps skills
  - Operational complexity

Pros and cons of Microservices

- Pros
  - Independence of services
  - Focus on business capabilities of apps
  - Simplicity of adding new features
  - Fault tolerance

- Cons
  - Implicit interfaces
  - Operational overhead
  - Require DevOps skills
  - Operational complexity

Pros and cons of Microservices (re-imagined for Jenkins)

- Pros
  - Independence of **jobs**
  - Focus on business capabilities of **jobs**
  - Simplicity of adding new **jobs**
  - Fault tolerance

- Cons
  - Implicit interfaces
  - Operational overhead
  - Require **Jenkins Admin** skills
  - Operational complexity

A platform for serverless applications

- Instead of supporting a **specific application**, clusters of servers provide a **generic execution environment** for any number of applications.
- Run in stateless compute containers that are **event-triggered**, **ephemeral** (may only last for one invocation), and fully managed by the **serverless application platform**

Jenkins is a platform for serverless microservices

- Instead of supporting a **specific job**, clusters of build agents provide a **generic build environment** for any number of jobs.
- Run in stateless executors that are **event-triggered**, **ephemeral** (may only last for one invocation), and fully managed by **Jenkins**

Pros and cons of serverless

- Pros
  - Reduced operational cost
  - Easier operational management

- Cons
  - Vendor control
  - Multitenancy problems
  - Vendor lock-in
  - Security concerns
  - Repetition of logic

Pros and cons of serverless (re-imagined for Jenkins)

- Pros
  - Reduced operational cost
  - Easier operational management

} If we remove the special snowflakes

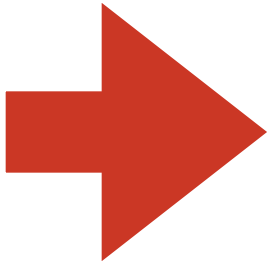
- Cons
  - ~~Vendor control~~
  - Multitenancy problems
  - ~~Vendor lock-in~~
  - Security concerns
  - Repetition of logic

How do we manage microservices?

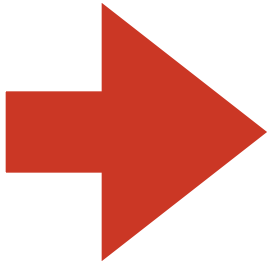
?

How do we manage microservices?

How do we manage microservices?



Building the Butler Army…



Jenkins

search

Jenkins › Chaos Butler

ENABLE AUTO REFRESH

- New Item
- People
- Build History
- Manage Jenkins
- Credentials
- Chaos Butler

Build Queue

No builds in the queue.

Build Executor Status

master
1 Idle
2 Idle

jnlp
1 Idle

Chaos Butler

The next victim is due to be selected in 0 ms

Recent victims

Victim	When
Jenkins	Wed Aug 31 17:01:06 IST 2016

Building the Butler Army

- Chaos Butler plugin => disconnect agents at random to test your processes ✅
- Chaos Steward plugin => restart masters at random
- Latency Butler plugin => keep builds in the queue for longer
- Conformity Butler plugin => disable jobs that do not adhere to best practices
- Doctor Butler plugin => disable jobs with failing health checks
- Janitor Butler plugin => disable unused jobs… for eventual removal
- Squeaky Wheel plugin => fails a build and disables job until ACK'd

Summary (theory)

- Target 30-50% capacity utilisation to maximise developer productivity
- Plan for at least 1 Jenkins master per 200 developers
  - Need more if your developers have an above average commit rate
- Eliminate special snowflakes
- Jobs are microservices
- Jenkins is a platform for serverless microservices

- Want Simian army for Jenkins?

Agenda

Real world

Theory

Experiments

Applied

Micro-experiments scaling Jenkins

- Memory usage
  - Jobs
  - Builds
  - Nodes

Micro-experiments scaling Jenkins

- Memory usage
  - Jobs
  - Builds
  - Nodes
- Remoting
  - Performance
  - Load

Micro-experiments scaling Jenkins

- Memory usage
  - Jobs
  - Builds
  - Nodes
- Remoting
  - Performance
  - Load
- Queue

Micro-experiments scaling Jenkins

- Memory usage
  - Jobs
  - Builds
  - Nodes
- Remoting
  - Performance
  - Load
- Queue
- Executor threads

Micro-experiments scaling Jenkins

- Memory usage
  - Jobs
  - Builds
  - Nodes
- Remoting
  - Performance
  - Load
- Queue
- Executor threads
- Archiving artifacts

Micro-experiments scaling Jenkins

- Memory usage
  - Jobs
  - Builds
  - Nodes
- Remoting
  - Performance
  - Load
- Queue
- Executor threads
- Archiving artifacts

} Measure memory consumption

Micro-experiments scaling Jenkins

- Memory usage
  - Jobs
  - Builds
  - Nodes

  } Measure memory consumption
- Remoting
  - Performance
  - Load

  } Measure scalability
- Queue
- Executor threads
- Archiving artifacts

Micro-experiments scaling Jenkins

- Memory usage
  - Jobs
  - Builds
  - Nodes

Measure memory consumption

- Remoting
  - Performance
  - Load

Measure scalability

Limit Experiment

- Queue
- Executor threads
- Archiving artifacts

Micro-experiments scaling Jenkins

- Memory usage
  - Jobs
  - Builds
  - Nodes

  } Measure memory consumption
- Remoting
  - Performance
  - Load

  } Measure scalability
- Queue

  Limit Experiment
- Executor threads
- Archiving artifacts

  } Measure

Memory usage

- We can use JAMM to measure memory usage
- JAMM walks the object graph to measure memory
- Need to use a custom implementation that understands Jenkins object graph
  - Ignores Jenkins singletons
  - Ignores parent references
  - Safely iterate run references
  - etc

https://github.com/jbellis/jamm

Demo 1

Memory usage (jobs/builds)

Name ↓	Memory usage
job	job: 81.27 KB 7 runs. Avg: 1.23 KB. Total 8.60 KB
job2	job: 75.30 KB 3 runs. Avg: 1.17 KB. Total 3.50 KB
job3	job: 7.55 KB N/A
matrix	job: 86.73 KB 4 runs. Avg: 1.66 KB. Total 6.65 KB
maven	job: 92.50 KB 6 runs. Avg: 31.84 KB. Total 191.05 KB
pipeline	job: 75.27 KB 4 runs. Avg: 4.80 KB. Total 19.21 KB

<— Freestyle

<— Empty builds

<— No builds

Memory usage

- Jobs with no builds are ~10kB
- Jobs with builds are ~100kB

- Builds start at ~2kB
- Maven builds start at ~30kB

- Nodes start at
  - ~210kB for JNLP
  - ~1.15MB for SSH (includes 1MB flight recording stream)

Remoting

- How much load does a remoting channel generate?
  - JNLP 2
  - JNLP 3
  - JNLP 4
  - SSH
- How many remoting channels can Jenkins handle?

Remoting loop-back harness

- Developed as part of CJOC's OperationsCenter2 remoting protocol test suite

- Single JVM for loopback or single server JVM and many client JVMs for client/server
- Sets up loop-back remoting connections
- Ramps up the number of connections
- Measures
  - average memory usage
  - JVM CPU load
  - JVM GC statistics

- Result gives the absolute upper limits for protocol on given system

Remoting loop-back harness


```
agent-load-test --help
 --client HOST:PORT      : Specify to run as a client only and connect to a
                           server on the specified HOST:PORT
 --clients CLIENTS       : The number of clients to simulate
 --collect SECONDS       : The number of seconds after all connections are
                           established to collect stats for before stopping
 --connect MILLIS        : The number of milliseconds to wait between client
                           starts
 --interval MILLISECONDS : The number of milliseconds each client waits before
                           sending a command
 --listen HOST:PORT      : Specify the hostname and port to listen on
 --protocol PROTOCOL     : The protocol to run the load test with
 --server                : Specify to run as a server only
 --size BYTES            : The number of bytes to pad the command with
 --stats FILE            : Filename to record stats to
 --warmup SECONDS        : The number of seconds after all connections are
                           established to warm up before resetting stats
```

Constraining heap to 2Gb

10 req/client/sec	# of clients	JVM load	Average heap usage
JNLP2	100	1.3	740 ± 140
JNLP3	100	1.3	800 ± 190
JNLP4 (TLS)	100	1.9	225 ± 50
JNLP4 (plaintext)	100	1.5	200 ± 40

Constraining heap to 512Mb

10 req/client/sec	# of clients	JVM load	Average heap usage
JNLP2	100	1.3	450 ± 30
JNLP3	100	1.3	512 ± 0
JNLP4 (TLS)	100	1.9	222 ± 40
JNLP4 (plaintext)	100	1.5	166 ± 45

Constraining heap to 128Mb

10 req/client/sec	# of clients	JVM load	Average heap usage
JNLP2	100	1.3	125 ± 1
JNLP3	100		
JNLP4 (TLS)	100	1.9	117 ± 4
JNLP4 (plaintext)	100	1.5	97 ± 2

Windows vs Everyone else…

10 req/client/sec	# of clients	JVM load	Average heap usage
JNLP2 *nix	100	1.3	740 ± 140
JNLP4 (plain) *nix	100	1.5	200 ± 40
JNLP2 win	100	0.41	540 ± 33
JNLP4 (plain) win	100	0.41	125 ± 33

Windows vs Everyone else…

10 req/client/sec	# of clients	JVM load	Average heap usage
JNLP2 *nix	100	1.3	740 ± 140
JNLP4 (plain) *nix	100	1.5	200 ± 40
JNLP2 win	100	0.41	540 ± 33
JNLP4 (plain) win	100	0.41	125 ± 33

15% difference

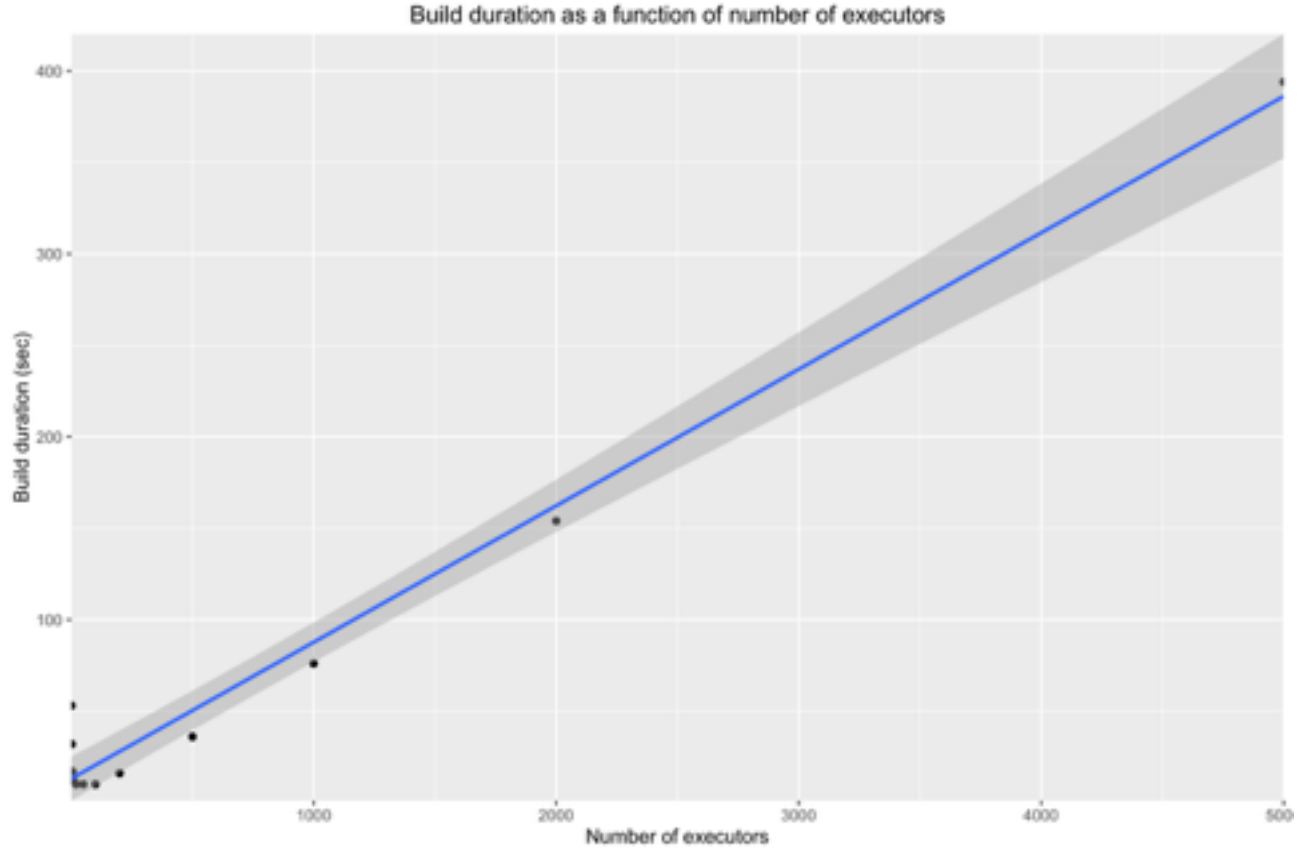
No difference

Queue

- How well does the queue scale?
- Put loads of no-op jobs in the queue and see how long it takes to complete as a function of number of executors
- Measure
  - Time it takes to enqueue all the jobs
  - Time it takes to assign all the jobs to build nodes

Demo 2

Queue



- Matrix build
- 1000 no-op combinations
- Three regions
- <20 executors
  - Executor availability
- 20-100 executors
  - Peak throughput
- >100 executors
  - Scheduling

Executor threads

- How much memory does an executor thread require?
- JAMM to the rescue again
- Idle executor ~ 1Kb
- Active executor ~ 50Kb
- Can be more depending on plugins in use

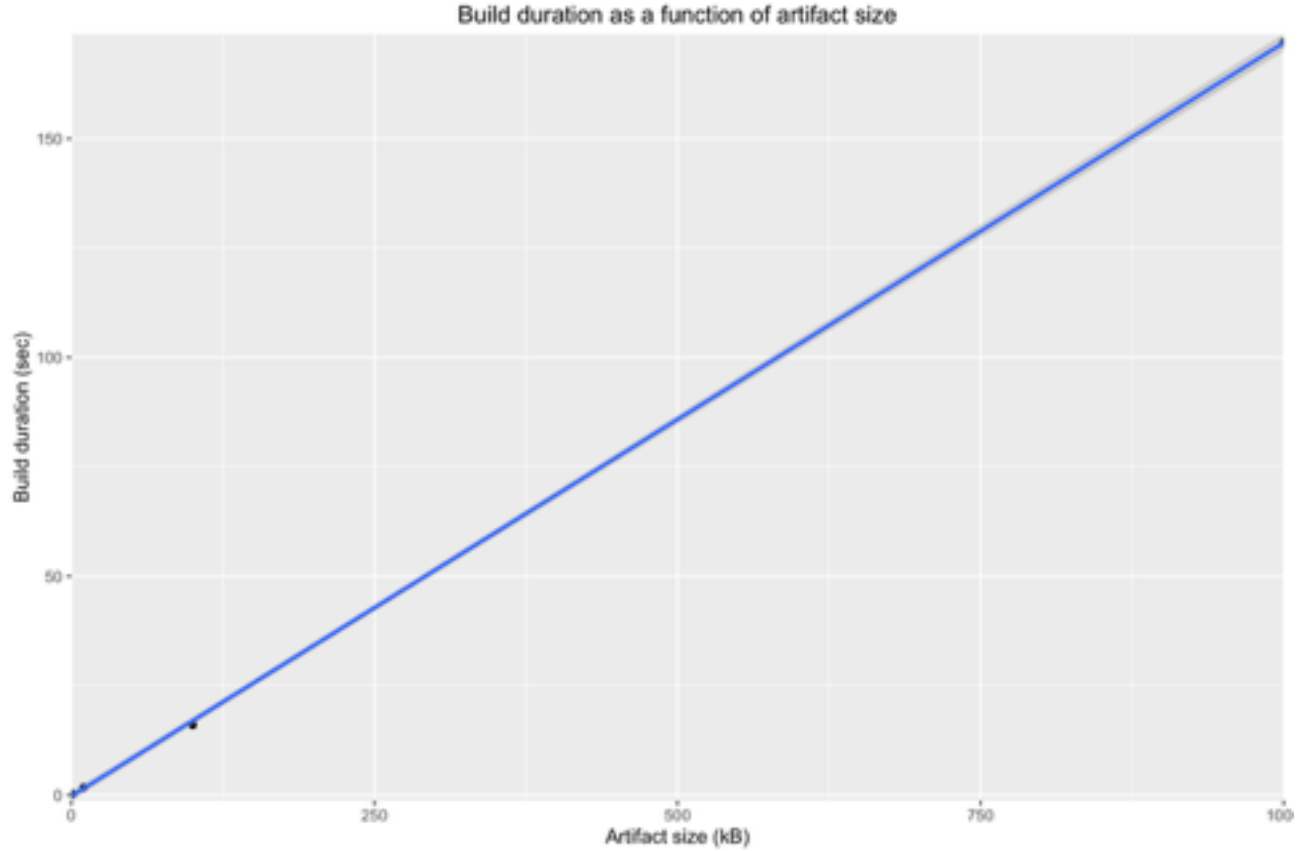
Archiving artifacts

- How much work does archiving artifacts put on the remoting layer?

- Use a custom agent that reports:
  - Number of commands sent and received
  - Number of bytes sent and received

- Run builds with different numbers and sizes of artifacts.

Vary size of artifact

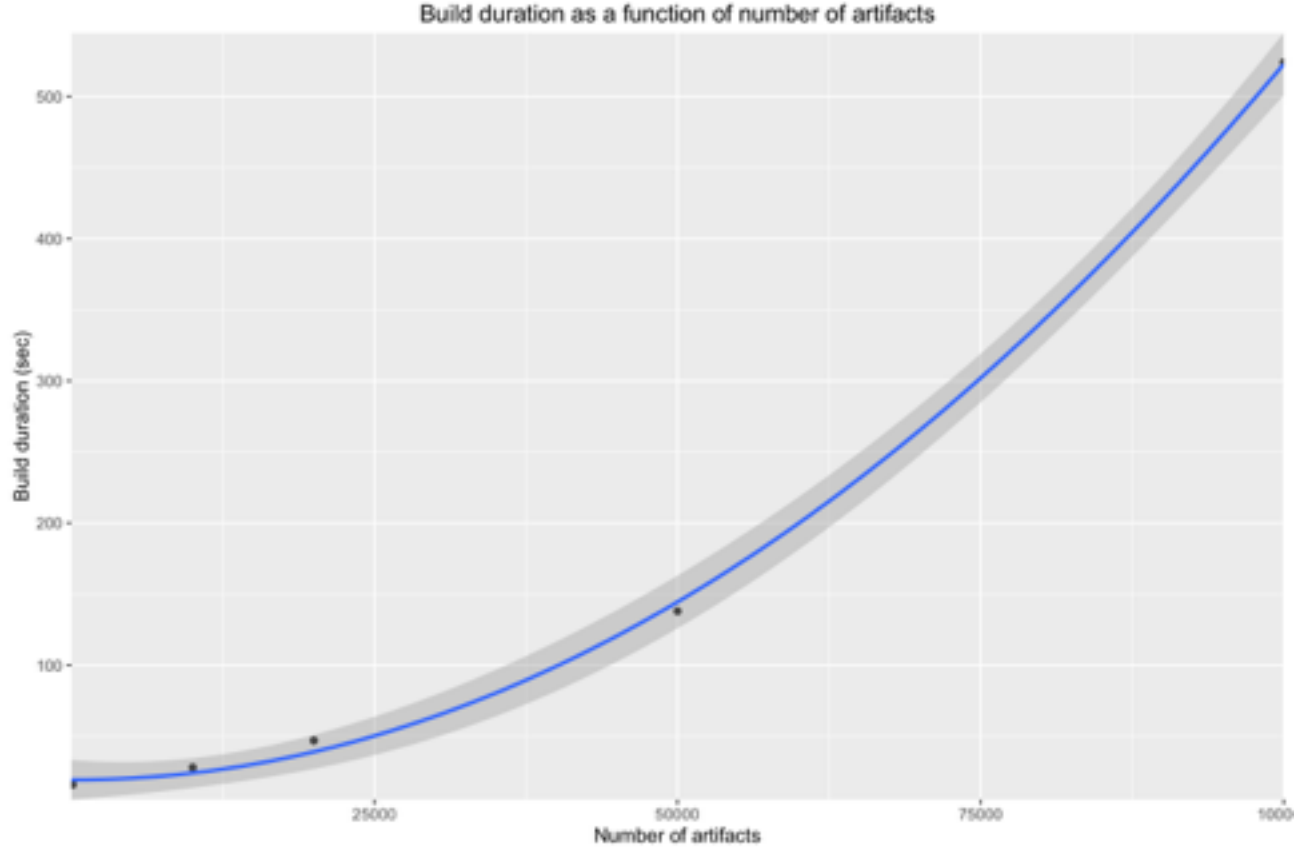


- Create various sizes of artifact
- Random bytes to ensure no compression effects

Duration = f(size)

Linear

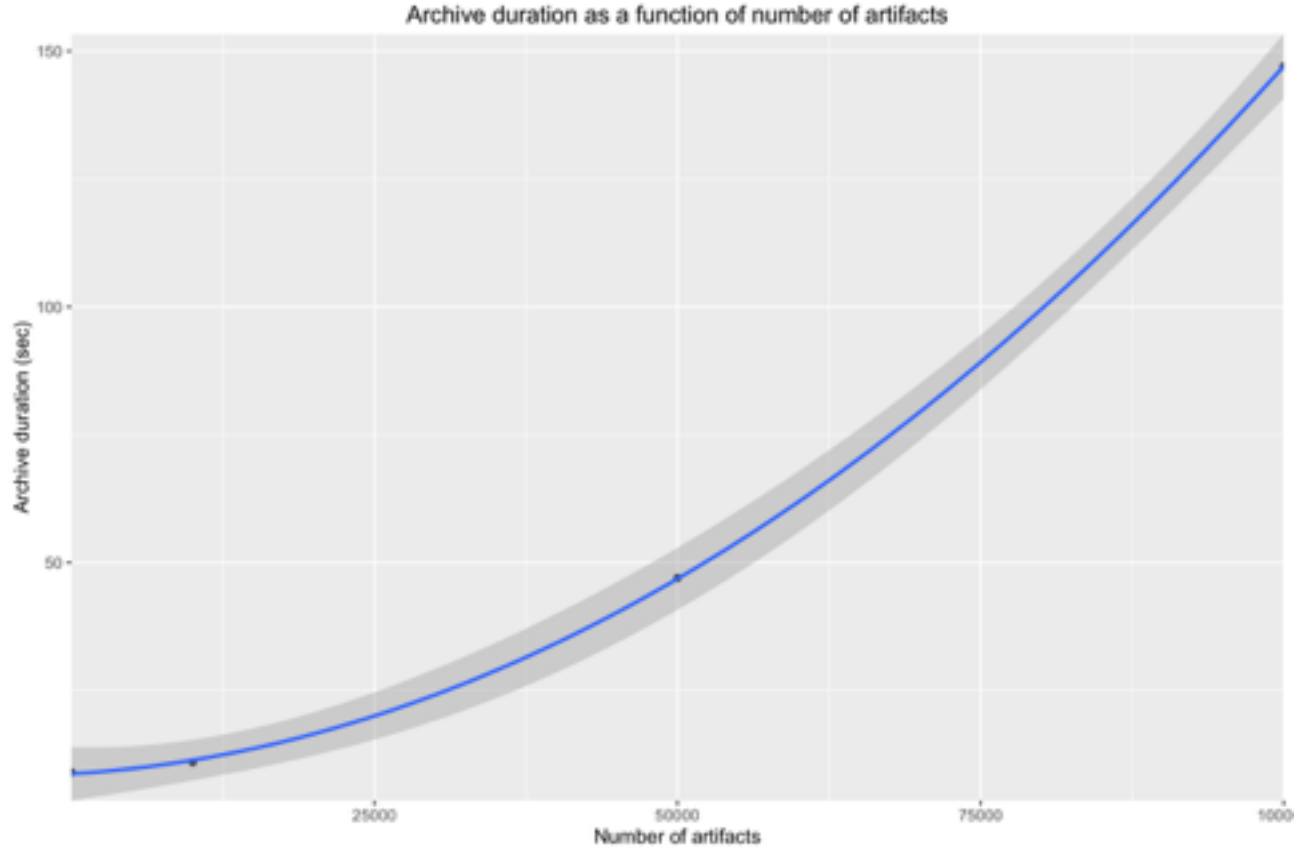
Vary number of artifacts (total size constant)



- Create 100m of artifacts to archive
- Random bytes to ensure no compression effects
- Vary number of files

Build Duration = $f(n^2)$

Vary number of artifacts (total size constant)



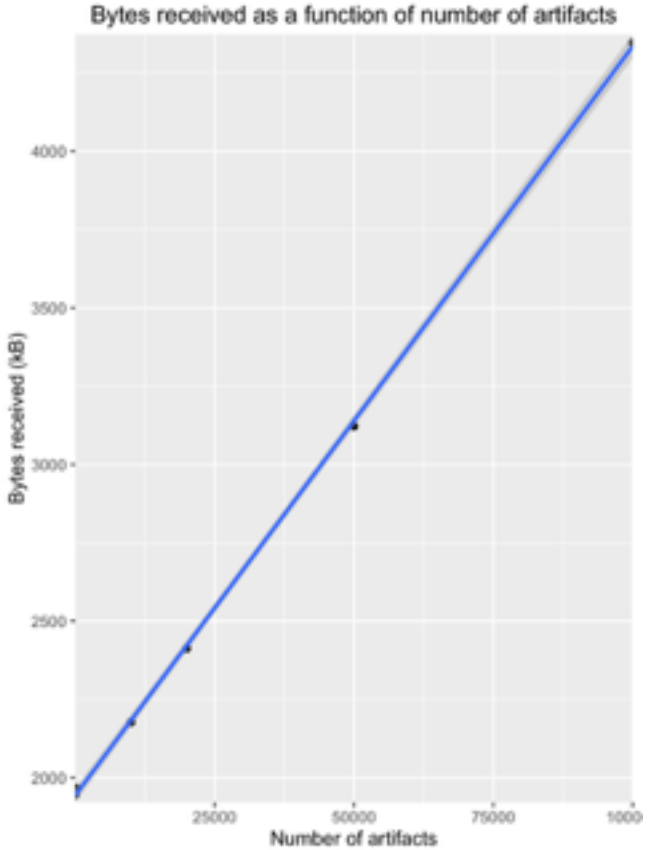
- Create 100m of artifacts to archive
- Random bytes to ensure no compression effects
- Vary number of files

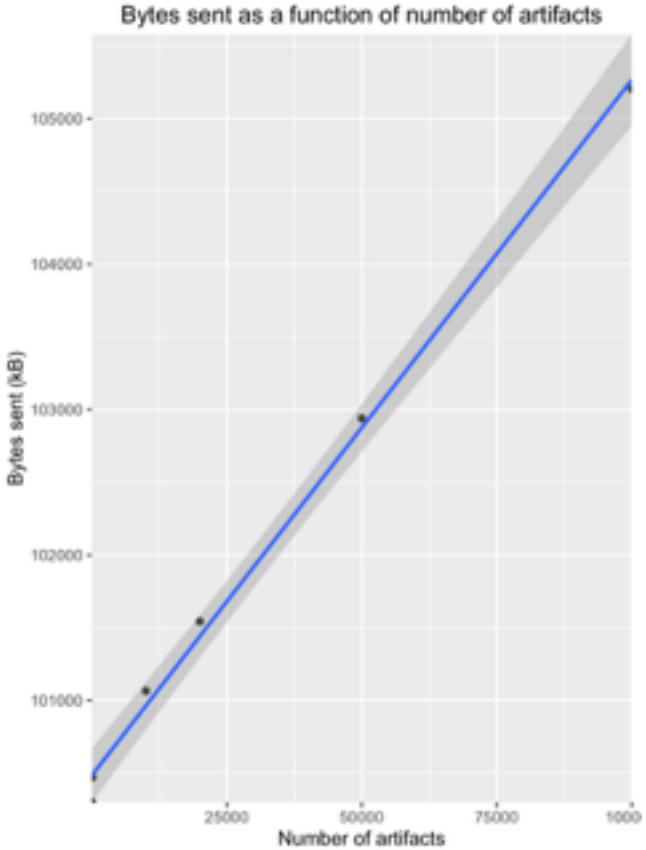
Build Duration = $f(n^2)$

Archive Duration = $f(n^2)$

$O(n^2)$

Vary number of artifacts (total size constant)





- Bytes received ≈ 24 n + v / 50
- Bytes sent ≈ 50 n + v
- 24 byte ACK for each file
- 150 byte ACK for each 8kB packet
- ~50 byte header for each file + file data itself

Summary (experiments)

- Memory usage
  - Jobs: starting from 100kB each
  - Builds: starting from 2kB each, 30kB for ~~Maven~~ Evil job type
  - Nodes: starting from 210kB per JNLP and 1200kB per SSH
- Remoting
  - Performance: JNLP4 => encryption + low memory use + more clients
  - Load: 4000 requests/sec ≈ ½ a 2.3GHz Intel Core i7 (2014)
- Queue: O(n) on number of executors (n > 100)
- Executor threads: idle from 1kB, active from 50kB
- Archiving artifacts: O(n) on size, $O(n^2)$ on number of files

Agenda

Real world

Theory

Experiments

Applied

Sizing the JVM heap…

- Ignoring Web UI requirements
- Absolute predicted base JVM heap average usage in MB:
  *(you **will** need more than this)*

$$512 + \frac{40a_{JNLP} + 240a_{SSH} + 2j(10+{}^{b}/_{j}) + 10c}{200}$$

a_{JNLP}	Number of JNLP agents	${}^{b}/_{j}$	Average builds per job
a_{SSH}	Number of SSH agents	c	Number of concurrent builds
j	Number of jobs		

Validating the JVM heap sizing guide…

- Support bundle taken after 40h of uptime
  - 1,104 nodes configured
  - 698 nodes on-line
  - 7,135 executors configured
  - 4,828 executors on-line
  - 5,784 jobs (92% freestyle)

- Master: Java 8

```
-Xms4096m -Xmx4096m
-XX:NewSize=2048m -XX:MaxNewSize=2048m
-XX:ParallelGCThreads=4 -XX:ConcGCThreads=4
-Dhudson.slaves.ChannelPinger.pingInterval=-1
```

- Agents running Java 7/8:
  - Majority SSH:

```
-Xms128m -Xmx512m
-Dhudson.remoting.Launcher.pingTimeoutSec=600
-Dhudson.remoting.Launcher.pingIntervalSec=1200
```

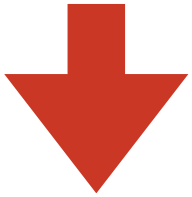
  - 5% SSH:

```
-Xms128m -Xmx512m
-Dhudson.remoting.Launcher.pingIntervalSec=-1
```

  - 1% JNLP (mostly Windows)

Validating the JVM heap sizing guide…

- Support bundle taken after 40h of uptime
  - 1,104 nodes configured
  - 698 nodes on-line
  - 7,135 executors configured
  - 4,828 executors on-line
  - 5,784 jobs (92% freestyle)



Average usage ≈ 2200MB

- Master: Java 8

```
-Xms4096m -Xmx4096m
-XX:NewSize=2048m -XX:MaxNewSize=2048m
-XX:ParallelGCThreads=4 -XX:ConcGCThreads=4
-Dhudson.slaves.ChannelPinger.pingInterval=-1
```

- Agents running Java 7/8:
  - Majority SSH:

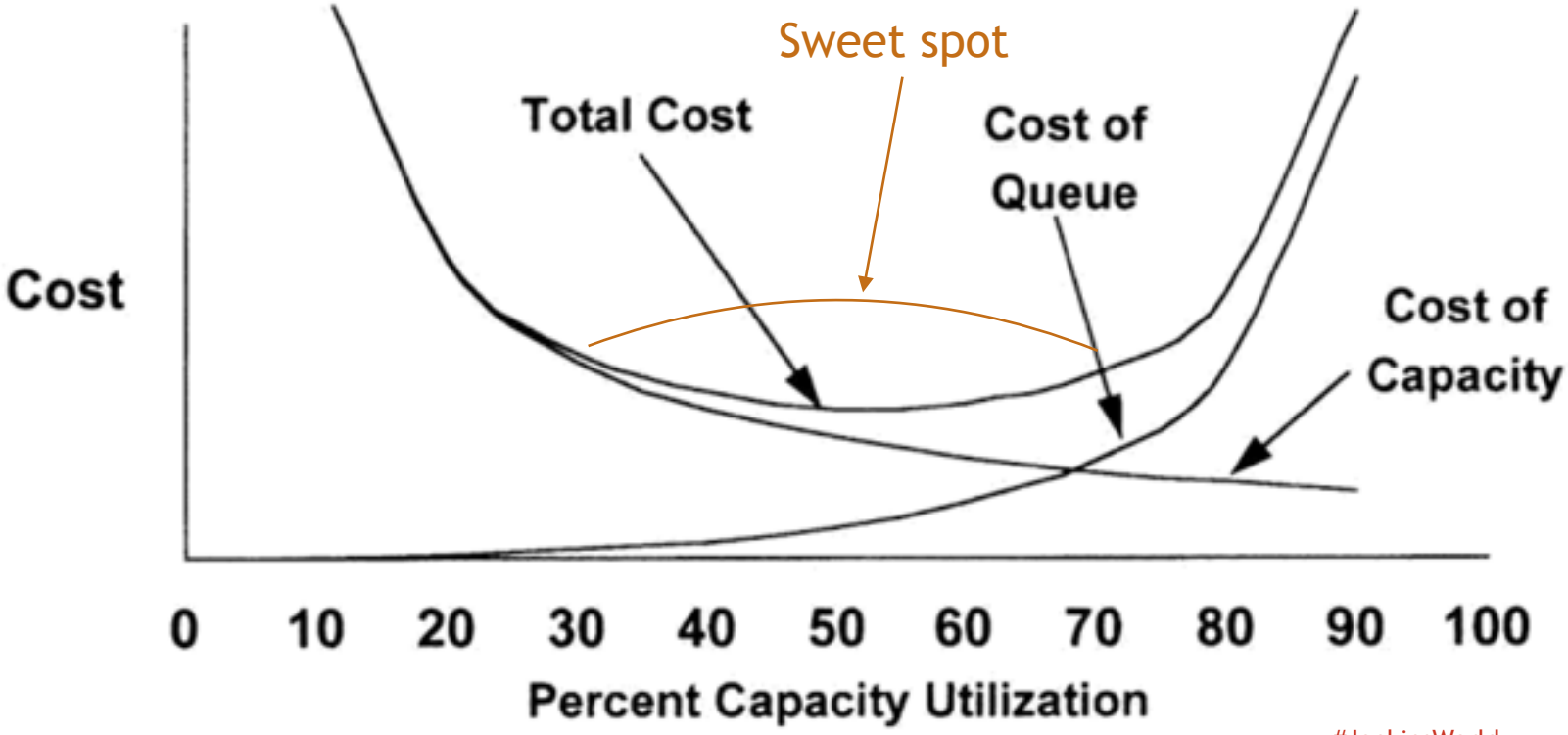
```
-Xms128m -Xmx512m
-Dhudson.remoting.Launcher.pingTimeoutSec=600
-Dhudson.remoting.Launcher.pingIntervalSec=1200
```

  - 5% SSH:

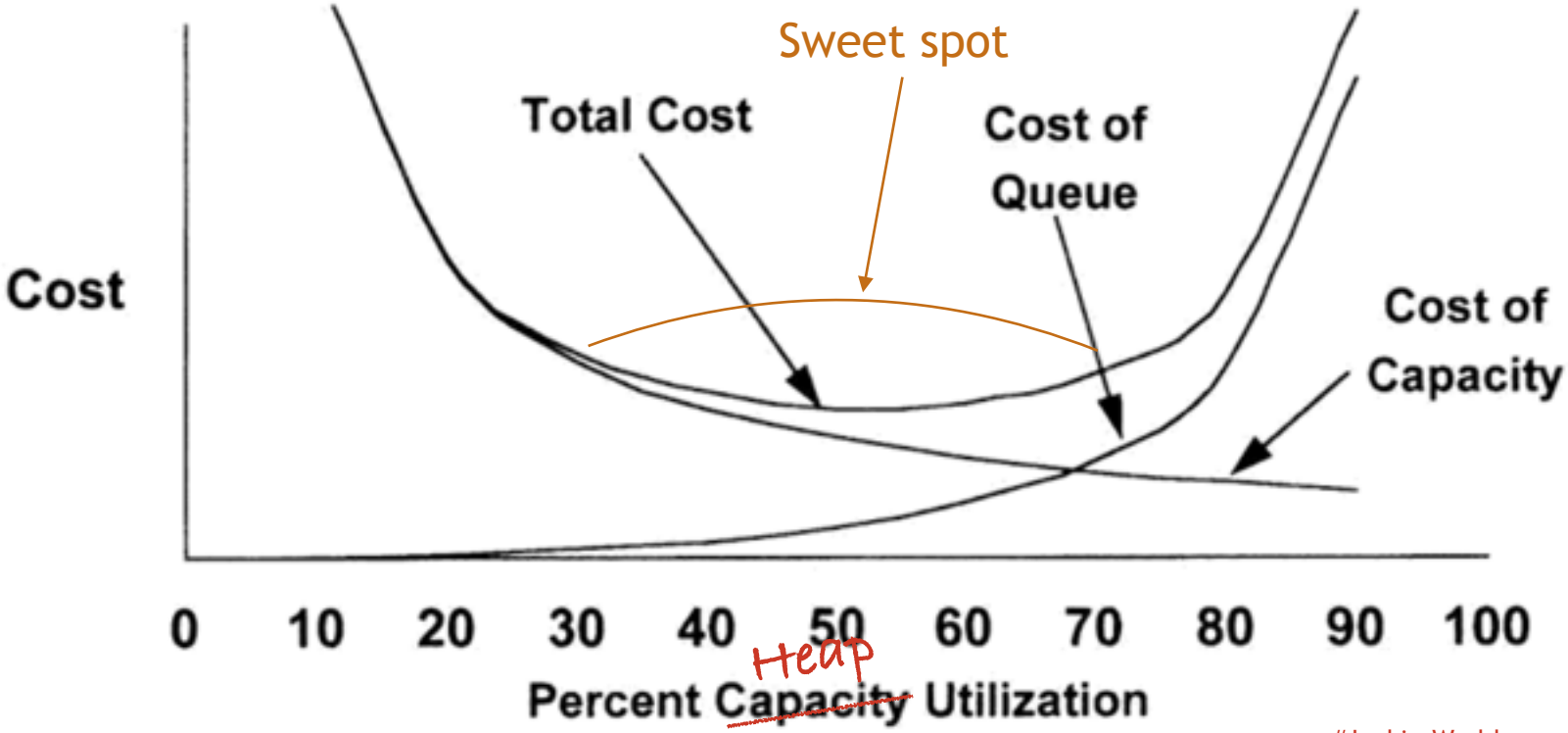
```
-Xms128m -Xmx512m
-Dhudson.remoting.Launcher.pingIntervalSec=-1
```

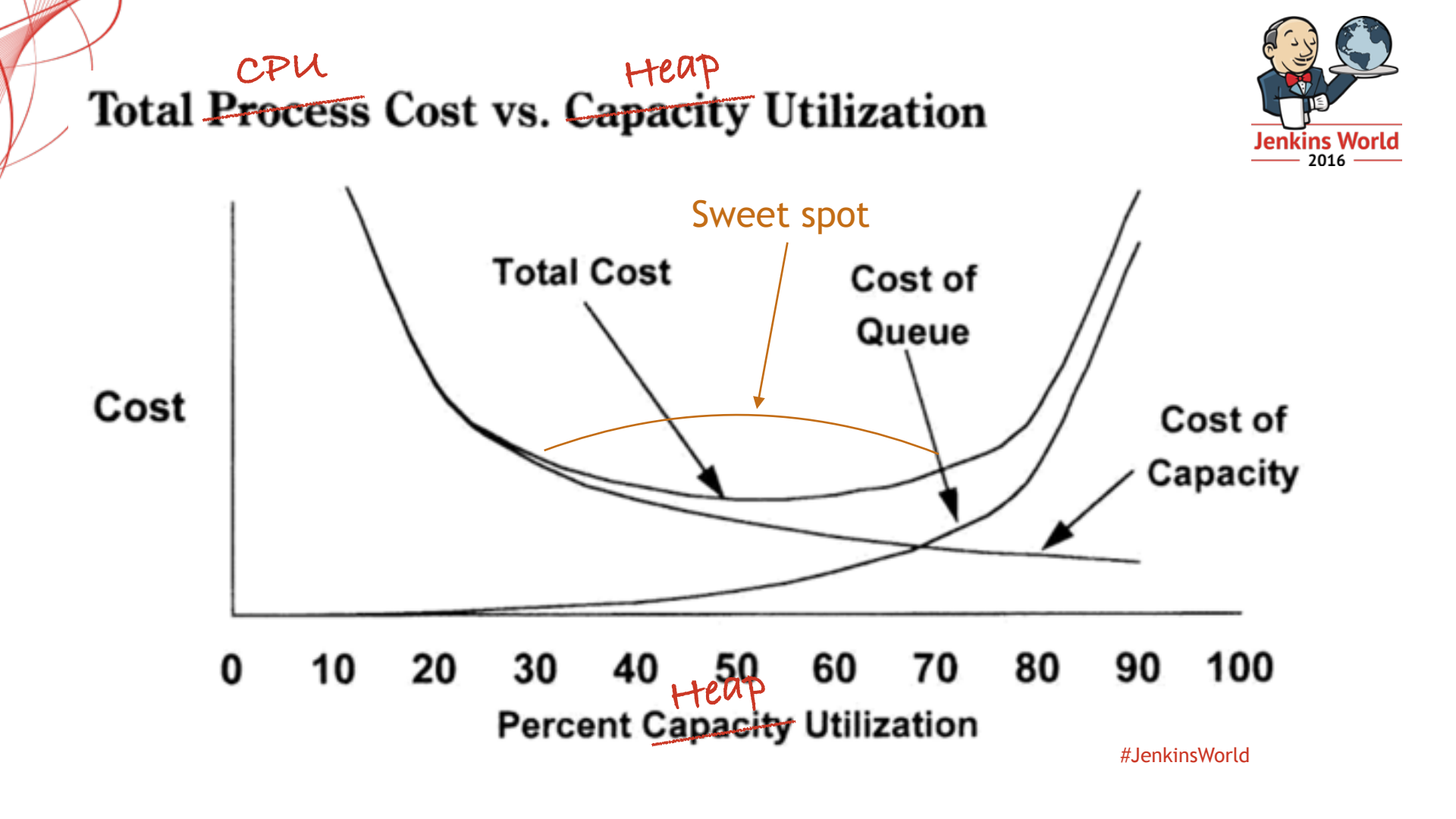
  - 1% JNLP (mostly Windows)

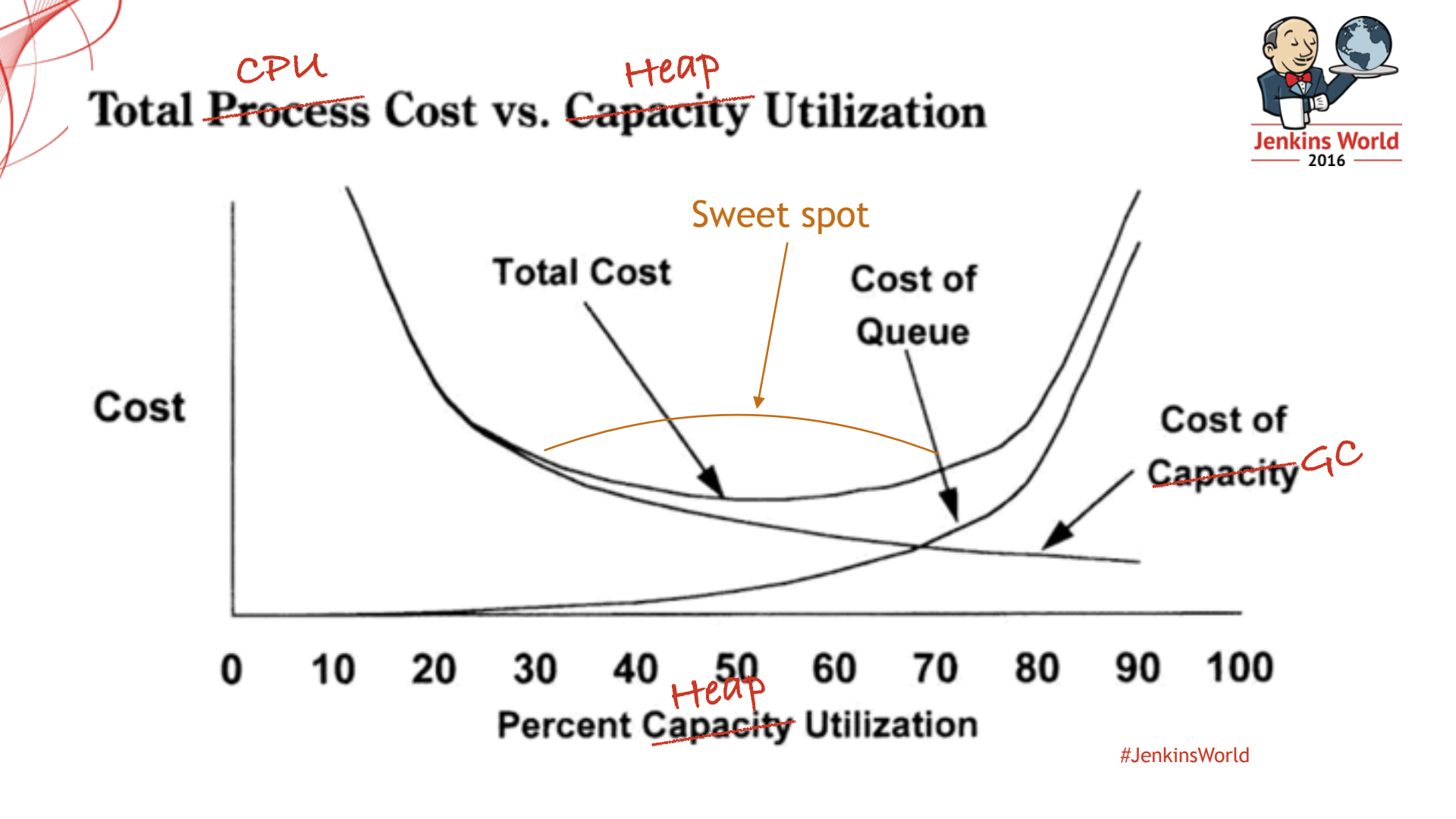
Total Process Cost vs. Capacity Utilization

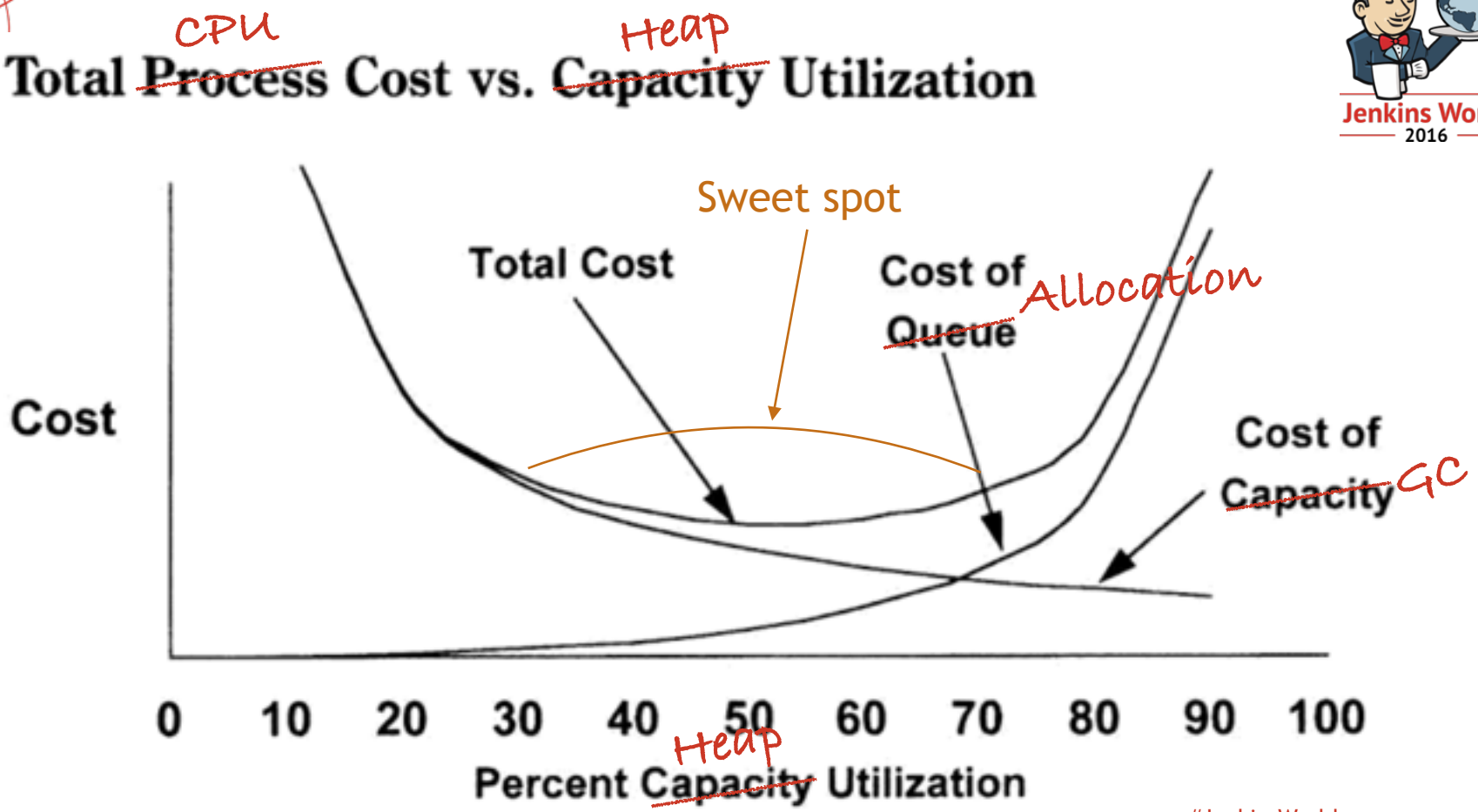


Total Process Cost vs. ~~Capacity~~ Heap Utilization




CPU
Heap
Total Process Cost vs. Capacity Utilization
Jenkins World
2016
Sweet spot
Total Cost
Cost of
Queue
Cost
Cost of
Capacity
0
10
20
30
40
50
60
70
80
90
100
Heap
Percent Capacity Utilization
#JenkinsWorld





Jenkins World 2016

#JenkinsWorld

Validating the JVM heap sizing guide…

- Support bundle taken after 40h of uptime
  - 1,104 nodes configured
  - 698 nodes on-line
  - 7,135 executors configured
  - 4,828 executors on-line
  - 5,784 jobs (92% freestyle)

```
-Xms4096m  -Xmx4096m
```

Average usage ≈ 2200MB

Instance sizing

- Aim for some reasonable maximums
  - 200 agents per master
  - 1000 jobs per master
  - average 15 builds per job
  - peak 100 concurrent builds

Instance sizing

- Aim for some reasonable maximums
  - 200 agents per master
  - 1000 jobs per master
  - average 15 builds per job
  - peak 100 concurrent builds

- Apply formula
  - 2.2Gb expected average memory

  - `-Xms4G –Xmx4G` should suffice

Instance sizing

- Aim for some reasonable maximums
  - 200 agents per master
  - 1000 jobs per master
  - average 15 builds per job
  - peak 100 concurrent builds
- Apply formula
  - 2.2Gb expected average memory
  - `-Xms4G –Xmx4G` should suffice

- ☑ **Give each team (or group of small teams) their own master**
- ☑ **Plan for growth up to ~200 agents**
- ☑ **Plan for growth up to ~1000 jobs**
- ☑ **Use inter-master solutions to connect teams**
- ☑ **4Gb is the starting point for JVM tuning**

Example


100,000 concurrent builds

Summary

- ☑ 4Gb JVM heap is your friend
  … by all means tune from there if evidence shows it is needed
- ☑ Can do 2,000 agents per master
  … should do 200 (less pain to administer)
- ☑ Agents sitting idle is a good thing
  … means developer productivity is maximised
- ☑ Your jobs are serverless microservices, Jenkins is their platform
  … manage your platform as a platform
  … manage your microservices as microservices

Summary

- ☑ 4Gb JVM heap is your friend
  … by all means tune from there if evidence shows it is needed
- ☑ Can do 2,000 agents per master
  … should do 200 (less pain to administer)
- ☑ Agents sitting idle is a good thing
  … means developer productivity is maximised
- ☑ Your jobs are serverless microservices, Jenkins is their platform
  … manage your platform as a platform
  … manage your microservices as microservices

- ☑ **Jenkins can scale**
  … 100,000 concurrent builds is just a question of having enough agents

Jenkins World

2016

#JenkinsWorld