

# So you want to build the worlds largest Jenkins cluster?

Stephen Connolly





**Jenkins World**  
2016

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

Stephen Connolly

#JenkinsWorld

# About me



Jenkins World  
2016

## Life

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



#JenkinsWorld

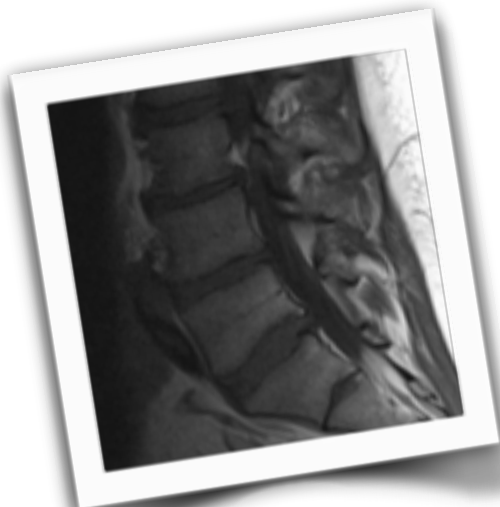
# About me



Jenkins World  
2016

## Life

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



#JenkinsWorld

# About me



Jenkins World  
2016

## 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



#JenkinsWorld

# About me



Jenkins World  
2016

## Life



My wife and I

Nov 27, 2007; 2:01am Re: plugins disabled when using m2 build ?

[stephenconnolly](#)



2022 posts

Use the freestyle project type and build with Maven  
The Maven2 project type is a "magic" project type that does everything  
it's own way.

This basically means that each plugin must do more work to work with it  
I keep on trying to use the Maven2 project type and finding it  
inadequate and returning to the freestyle project type.

You'll have more control with the freestyle (but more control means  
more configuration)

-Stephen

...many plugins since then

Also Apache Maven committer & PMC



Fighting against the evil Maven Job type since 2007...

# Agenda



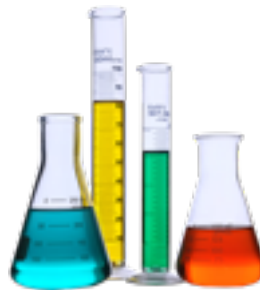
Jenkins World  
2016



Real world



Theory



Experiments



Applied

Before we start...



Jenkins World  
2016



#JenkinsWorld



Before we start...



Jenkins World  
2016

# world's biggest Jenkins cluster

#JenkinsWorld

Before we start...



Jenkins World  
2016

# world's biggest Jenkins cluster



*what does this mean?*

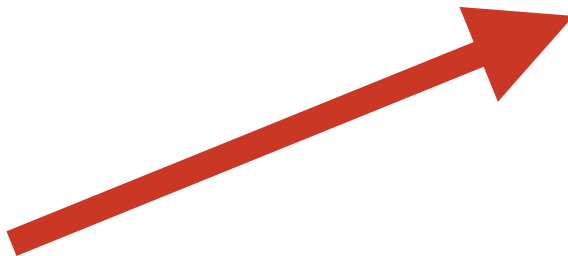
#JenkinsWorld

Before we start...



Jenkins World  
2016

# world's biggest Jenkins cluster



*or this?*

#JenkinsWorld

# Biggest...



**Jenkins World**  
2016

- If we want the physically largest...



- Region & Number of Availability Zones
- New Region Coming Soon

# Biggest...



Jenkins World  
2016

- 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...



Jenkins World  
2016

- 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...



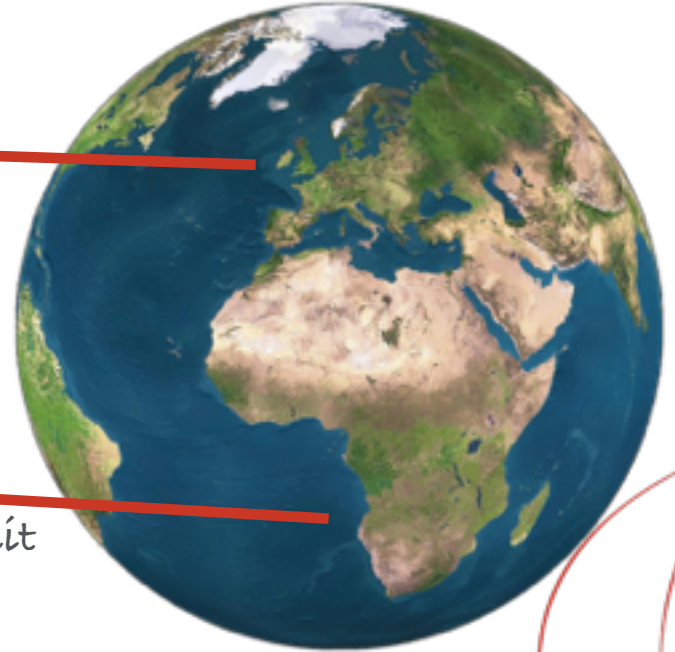
Jenkins World  
2016

- 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

southern limit



Smallest Bounding Sphere containing all points



# Biggest...



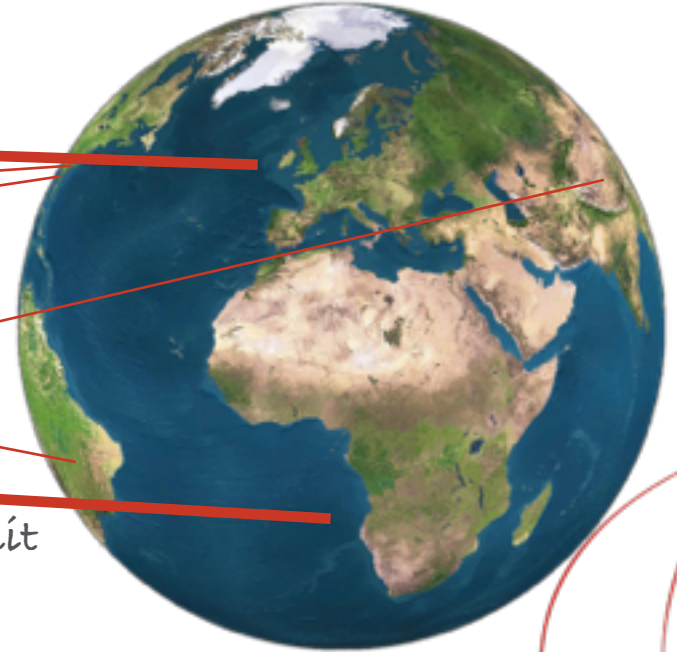
Jenkins World  
2016

- 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

southern limit



Smallest Bounding Sphere containing all points

A screenshot of the Jenkins web interface. The browser address bar shows 'jenkins.example.com:8080'. The Jenkins logo is in the top left. A sidebar on the left contains navigation links: 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'Credentials'. Below these are sections for 'Build Queue' (showing 'No builds in the queue.') and 'Build Executor Status' (listing nodes: 'master', 'Sydney', 'São Paulo', 'Tokyo', 'US East', and 'US West', each with '1 idle'). The main content area shows a table of build jobs with columns for 'Name', 'Last Success', 'Last Failure', and 'Last Duration'. A single job named 'a JOB' is listed with a sun icon. Below the table are links for 'RSS for all', 'RSS for failures', and 'RSS for just latest builds'. The footer of the page indicates it was generated on 'Jul 19, 2016 10:08:00 AM EDT' and provides links for 'REST API' and 'Jenkins ver. 2.16 SNAPSHOT (build:61182016.18.11) (msbhrnc)'.

	W	Name ↓	Last Success	Last Failure	Last Duration
🌞	☀️	a JOB	N/A	N/A	N/A

The world's biggest Jenkins!!!



# Biggest...



Jenkins World  
2016

- 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?



#JenkinsWorld

# Biggest...



**Jenkins World**  
2016

- If we want the most number of jobs...

# Biggest...



Jenkins World  
2016

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

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

```
Created mock-load-job-54751 with average duration 38s  
Created mock-load-job-54752 with average duration 44s  
Created mock-load-job-54753 with average duration 288s  
Created mock-load-job-54754 with average duration 48s  
Created mock-load-job-54755 with average duration 134s  
Created mock-load-job-54756 with average duration 173s  
Created mock-load-job-54757 with average duration 46s  
Created mock-load-job-54758 with average duration 37s  
Created mock-load-job-54759 with average duration 36s  
Created mock-load-job-54760 with average duration 38s  
Created mock-load-job-54761 with average duration 184s  
Created mock-load-job-54762 with average duration 36s  
Created mock-load-job-54763 with average duration 56s  
Created mock-load-job-54764 with average duration 58s  
Created mock-load-job-54765 with average duration 84s  
Created mock-load-job-54766 with average duration 12s  
Created mock-load-job-54767 with average duration 45s  
Created mock-load-job-54768 with average duration 414s  
Created mock-load-job-54769 with average duration 22s  
Created mock-load-job-54770 with average duration 13s  
Created mock-load-job-54771 with average duration 12s  
Created mock-load-job-54772 with average duration 14s  
Created mock-load-job-54773 with average duration 25s  
Created mock-load-job-54774 with average duration 56s  
Created mock-load-job-54775 with average duration 17s  
Created mock-load-job-54776 with average duration 22s  
Created mock-load-job-54777 with average duration 24s  
Created mock-load-job-54778 with average duration 25s  
Created mock-load-job-54779 with average duration 56s  
Created mock-load-job-54780 with average duration 17s  
Created mock-load-job-54781 with average duration 22s  
Created mock-load-job-54782 with average duration 24s  
Created mock-load-job-54783 with average duration 26s  
Created mock-load-job-54784 with average duration 17s  
Created mock-load-job-54785 with average duration 38s  
Created mock-load-job-54786 with average duration 38s  
Created mock-load-job-54787 with average duration 128s  
Created mock-load-job-54788 with average duration 25s  
Created mock-load-job-54789 with average duration 35s  
Created mock-load-job-54790 with average duration 121s  
Created mock-load-job-54791 with average duration 42s  
Created mock-load-job-54792 with average duration 25s  
Created mock-load-job-54793 with average duration 61s  
Created mock-load-job-54794 with average duration 8s  
Created mock-load-job-54795 with average duration 361s  
Created mock-load-job-54796 with average duration 8s  
Created mock-load-job-54797 with average duration 7s  
Created mock-load-job-54800 with average duration 17s  
Created mock-load-job-54801 with average duration 13s
```

jenkinsWorld



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...



Jenkins World  
2016

- If we want the most number of executors...

# of executors

Labels

Usage



Jenkins World  
2016

The screenshot shows the Jenkins dashboard with a table of jobs and a list of executors. The table has columns for 'S', 'W', 'Name', 'Last Success', 'Last Failure', and 'Last Duration'. The jobs listed are 'mock-load-job-00004' through 'mock-load-job-00012'. The executor list on the left shows 19 executors, all in an 'idle' state.

S	W	Name	Last Success	Last Failure	Last Duration
		mock-load-job-00004	N/A	N/A	N/A
		mock-load-job-00005	N/A	N/A	N/A
		mock-load-job-00006	N/A	N/A	N/A
		mock-load-job-00007	N/A	N/A	N/A
		mock-load-job-00008	N/A	N/A	N/A
		mock-load-job-00009	N/A	N/A	N/A
		mock-load-job-00010	N/A	N/A	N/A
		mock-load-job-00011	N/A	N/A	N/A
		mock-load-job-00012	N/A	N/A	N/A

Build Executor Status

- master
- 1 idle
- 2 idle
- 3 idle
- 4 idle
- 5 idle
- 6 idle
- 7 idle
- 8 idle
- 9 idle
- 10 idle
- 11 idle
- 12 idle
- 13 idle
- 14 idle
- 15 idle
- 16 idle
- 17 idle
- 18 idle
- 19 idle

The world's biggest Jenkins!!!

200,001 jobs!!!

10,005 executors!!!



I asked for the biggest Jenkins cluster

And Stephen keeps using the wrong definitions of biggest

# Cluster...



Jenkins World  
2016

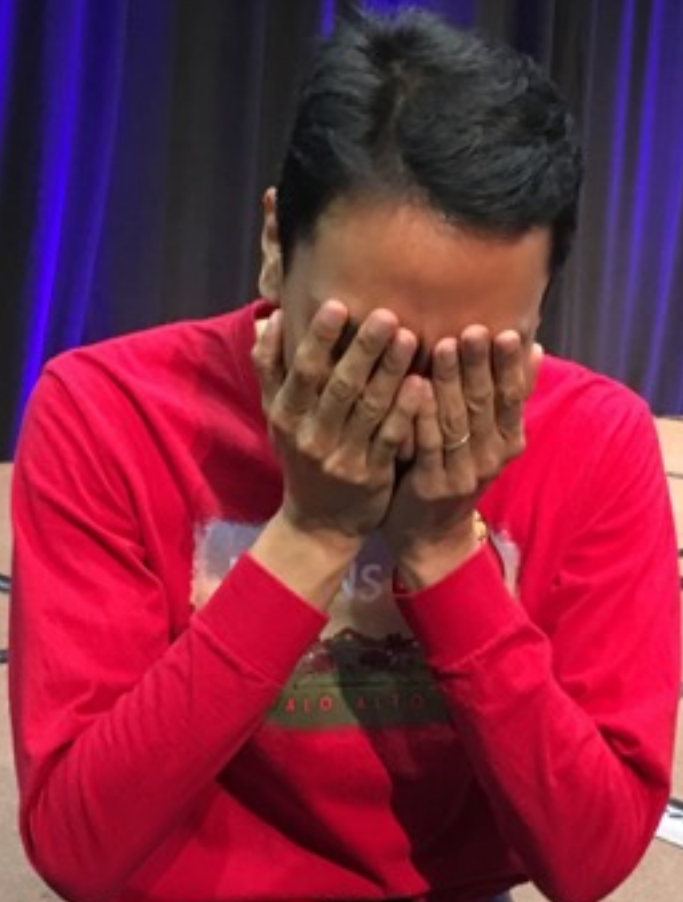
- Maybe I just run lots of Jenkins instances in parallel...



Jenkins World  
2016

A big Jenkins “cluster”!!!

Please don't make me pose for any more memes



We need some rules...



Jenkins World  
2016



#JenkinsWorld



- 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



Jenkins World  
2016

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




Jenkins World  
2016

- 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 be representative of real world frequencies.

In 2016 that means:

- 75% freestyle, 15% maven, 5% matrix, 5% pipeline



A wooden signpost stands in a grassy field. The sign is rectangular with a black border and a white background, featuring the word "Lost" in bold black letters and a black arrow pointing to the right. The background shows a misty, forested hillside under a grey sky.

Lost

Ok, where were we...

# Agenda



Jenkins World  
2016



Real world



Theory



Experiments



Applied

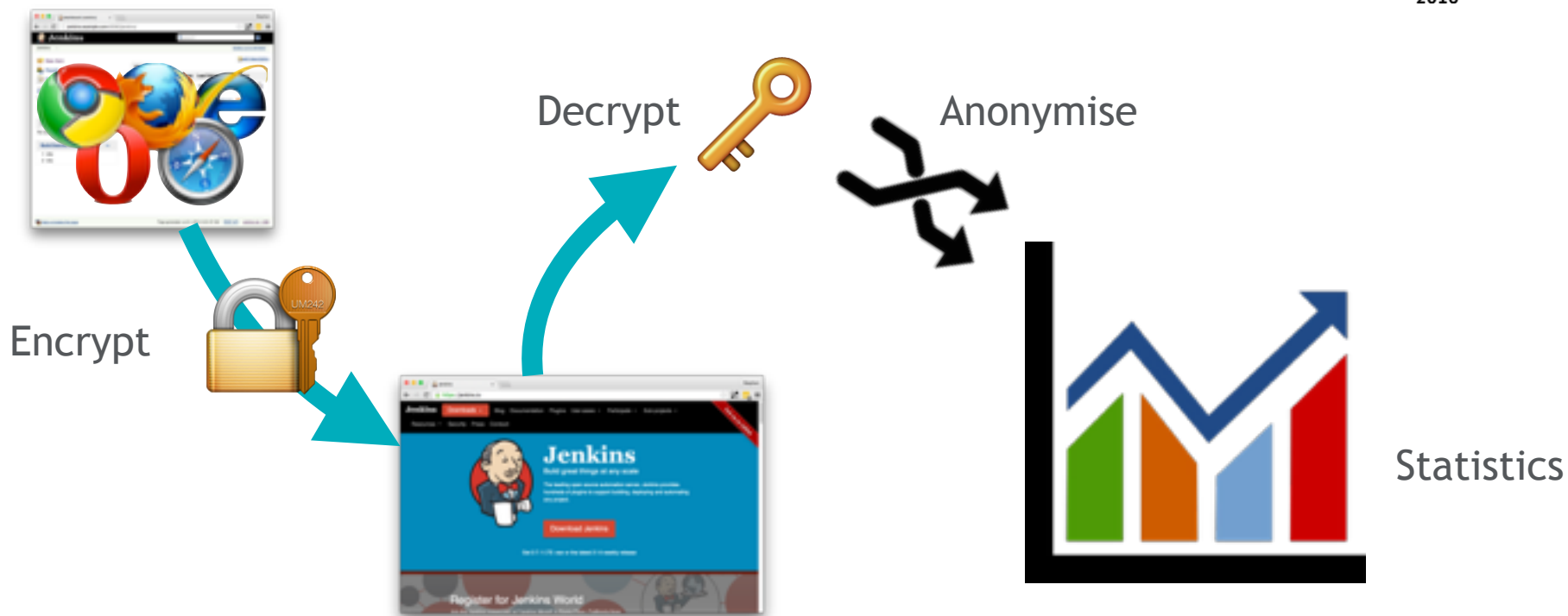


Jenkins phone home!



Jenkins World  
2016

# Usage Statistics in Jenkins



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

#JenkinsWorld



Jenkins World  
2016

# 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



Jenkins World  
2016

- 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



Jenkins World  
2016

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

# Usage Statistics in Jenkins



Jenkins World  
2016

- 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



Jenkins World  
2016

- 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



Jenkins World  
2016

## 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



Jenkins World  
2016

## 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



Jenkins World  
2016

- 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



#JenkinsWorld



# Health warning



Jenkins World  
2016

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



# Health warning



Jenkins World  
2016



- 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





- 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...”



- 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...”



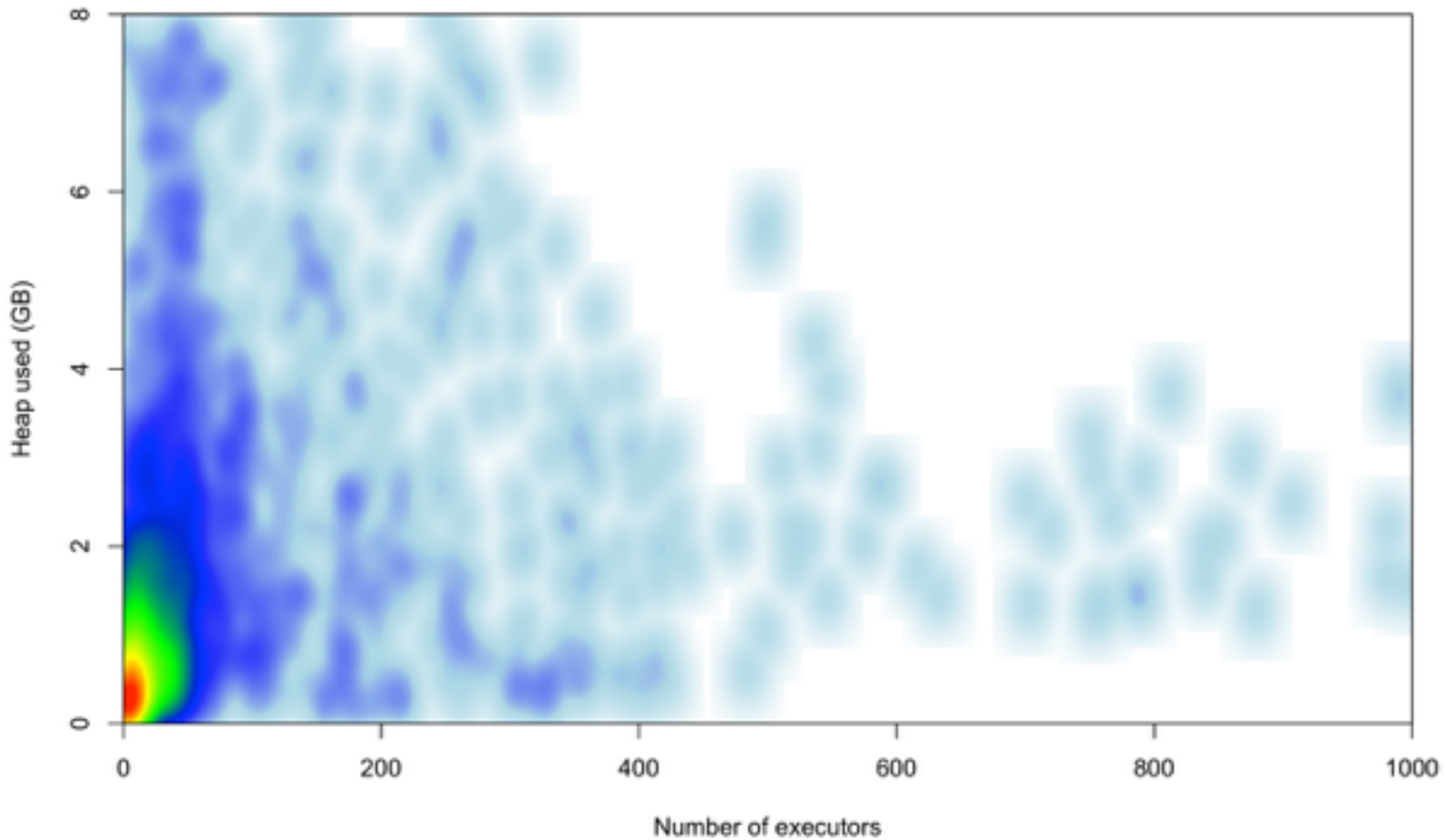
- Most of these bundles are from systems where customers had a requirement for support
  - We ask for support bundles routinely, so some are from customers who just had questions
- Expect short uptime
- Expect low usage
- Customers can censor how much of the bundle to send us

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



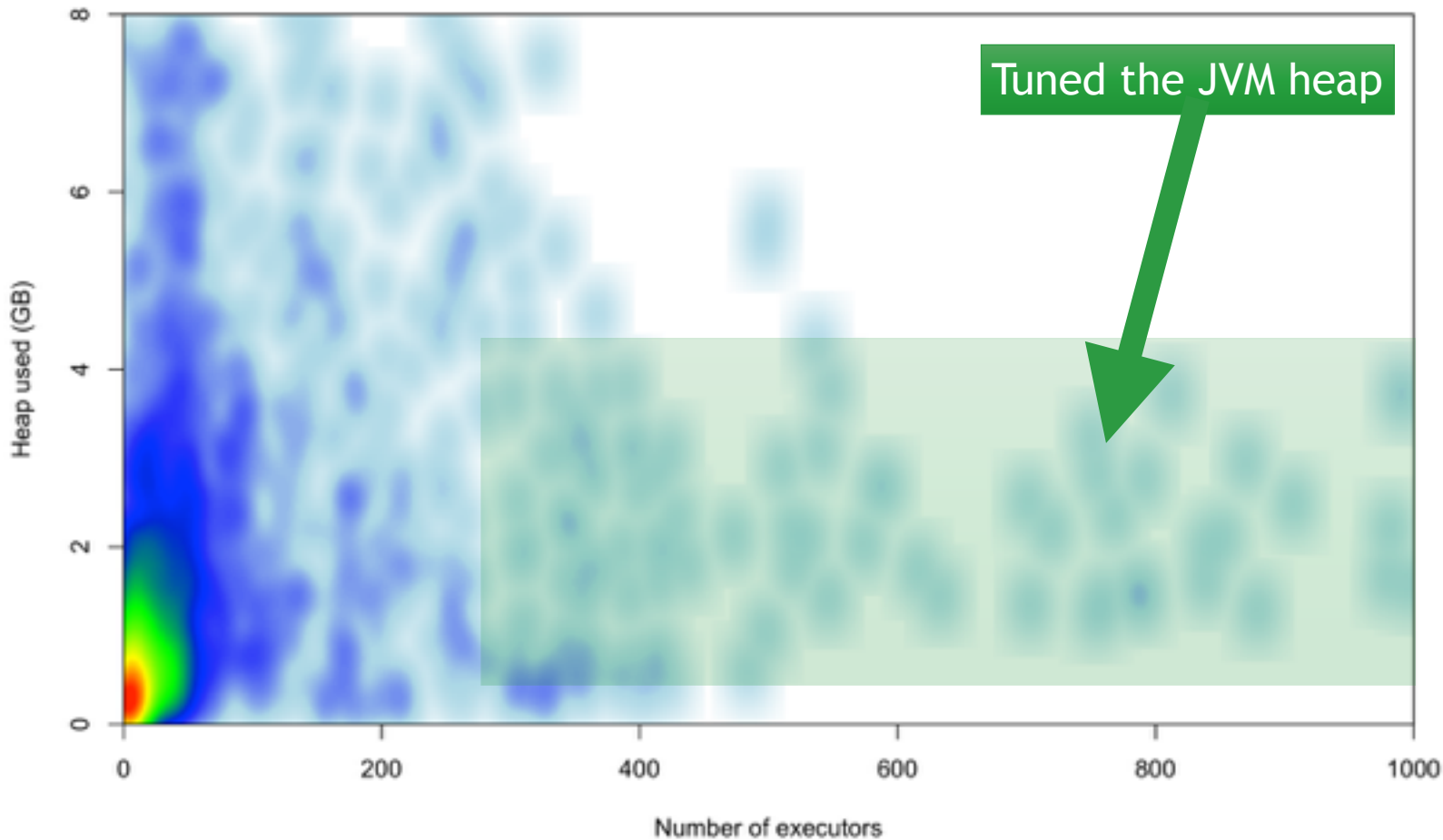
- Includes
  - # of nodes on-line
  - JVM tuning arguments

# Heap usage as a function of number of executors

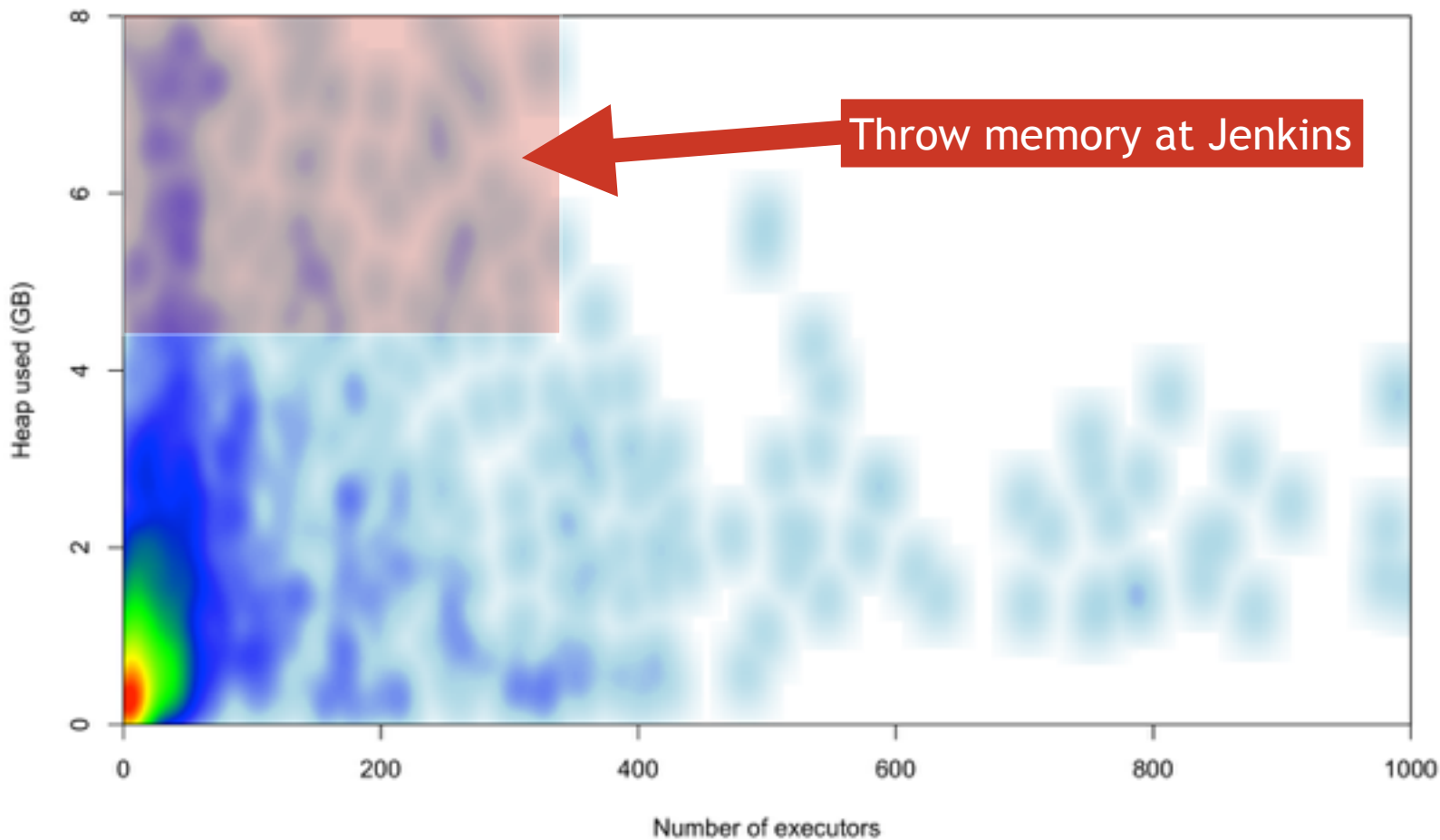




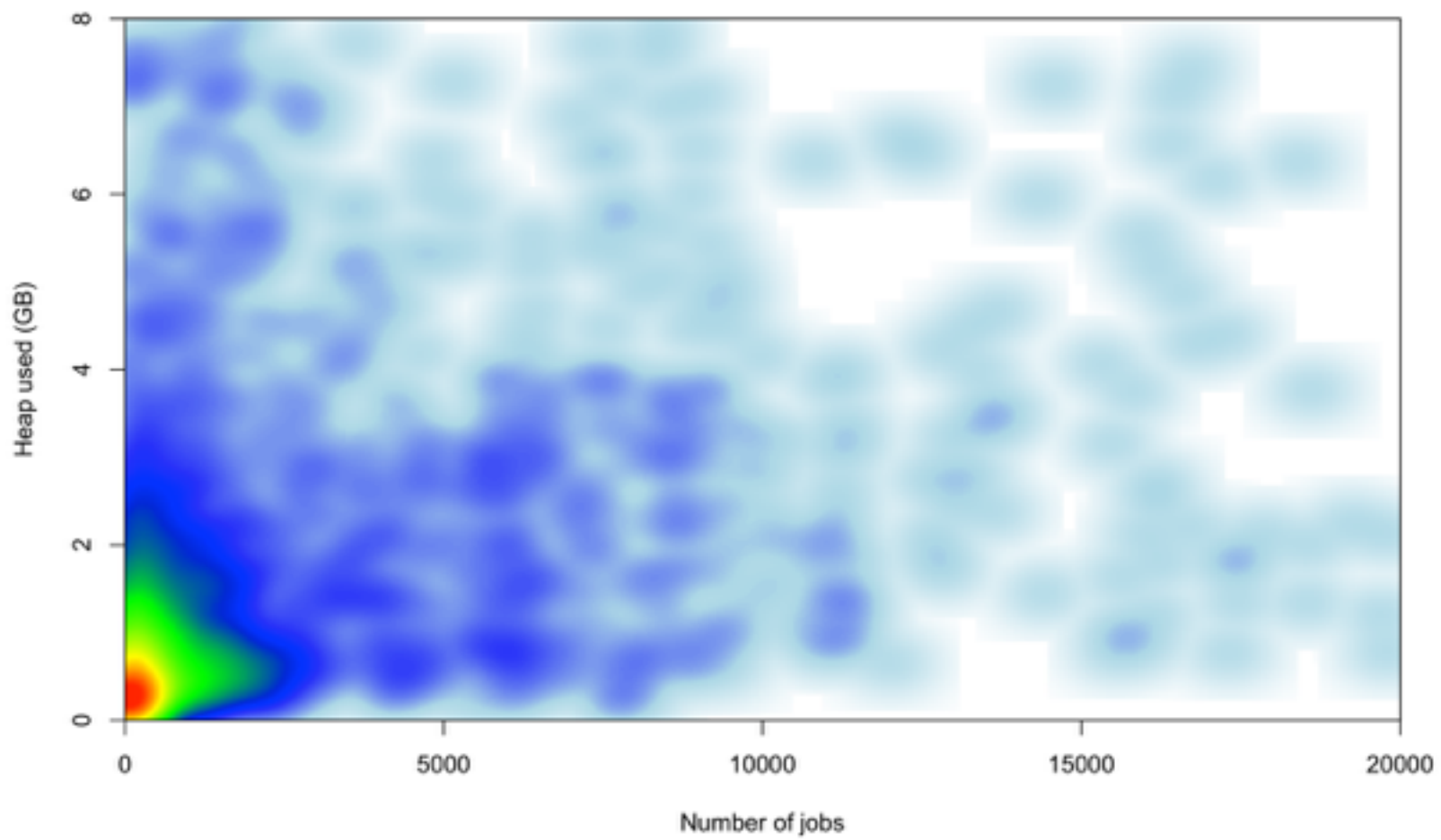
# Heap usage as a function of number of executors



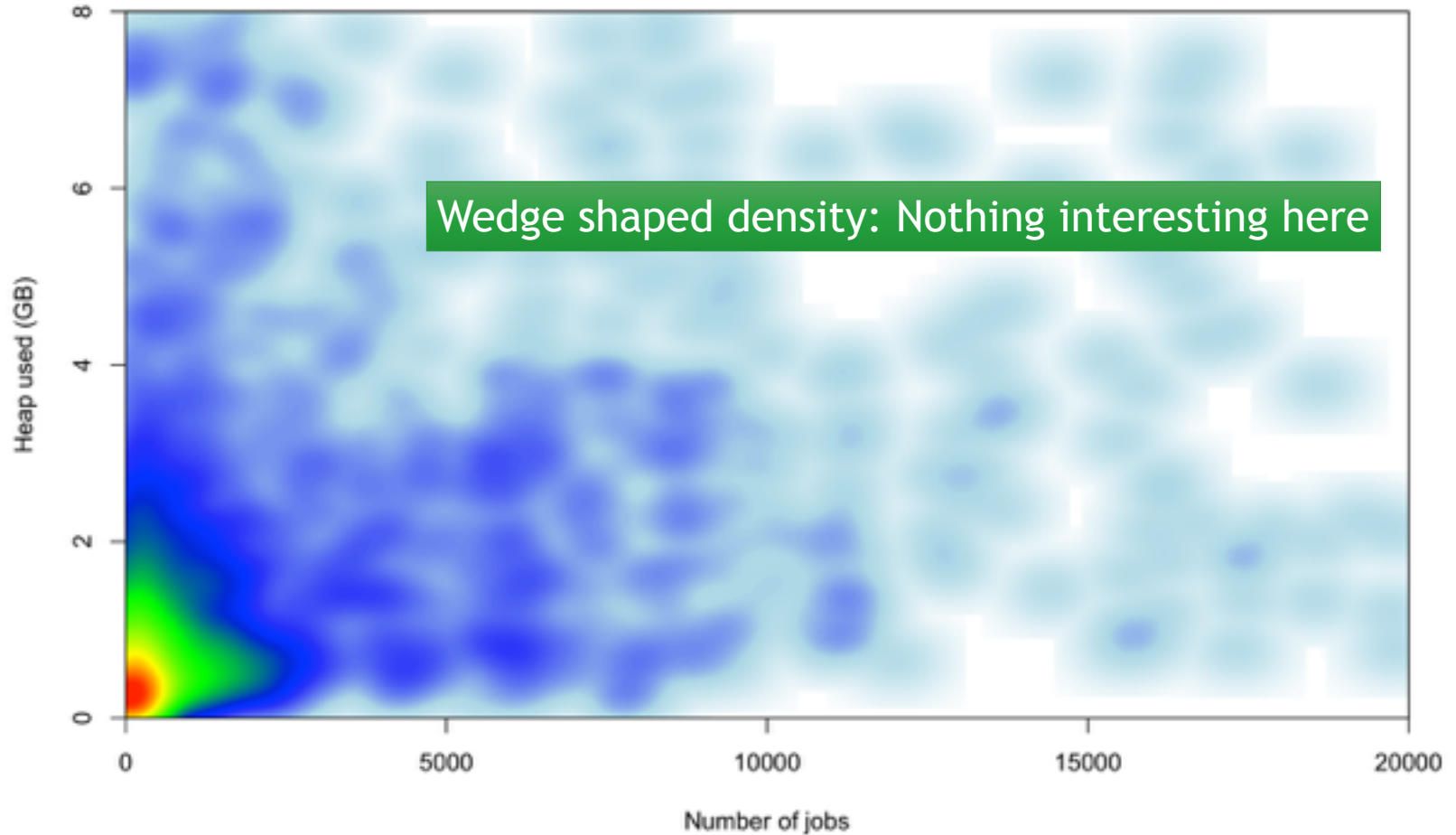
## Heap usage as a function of number of executors



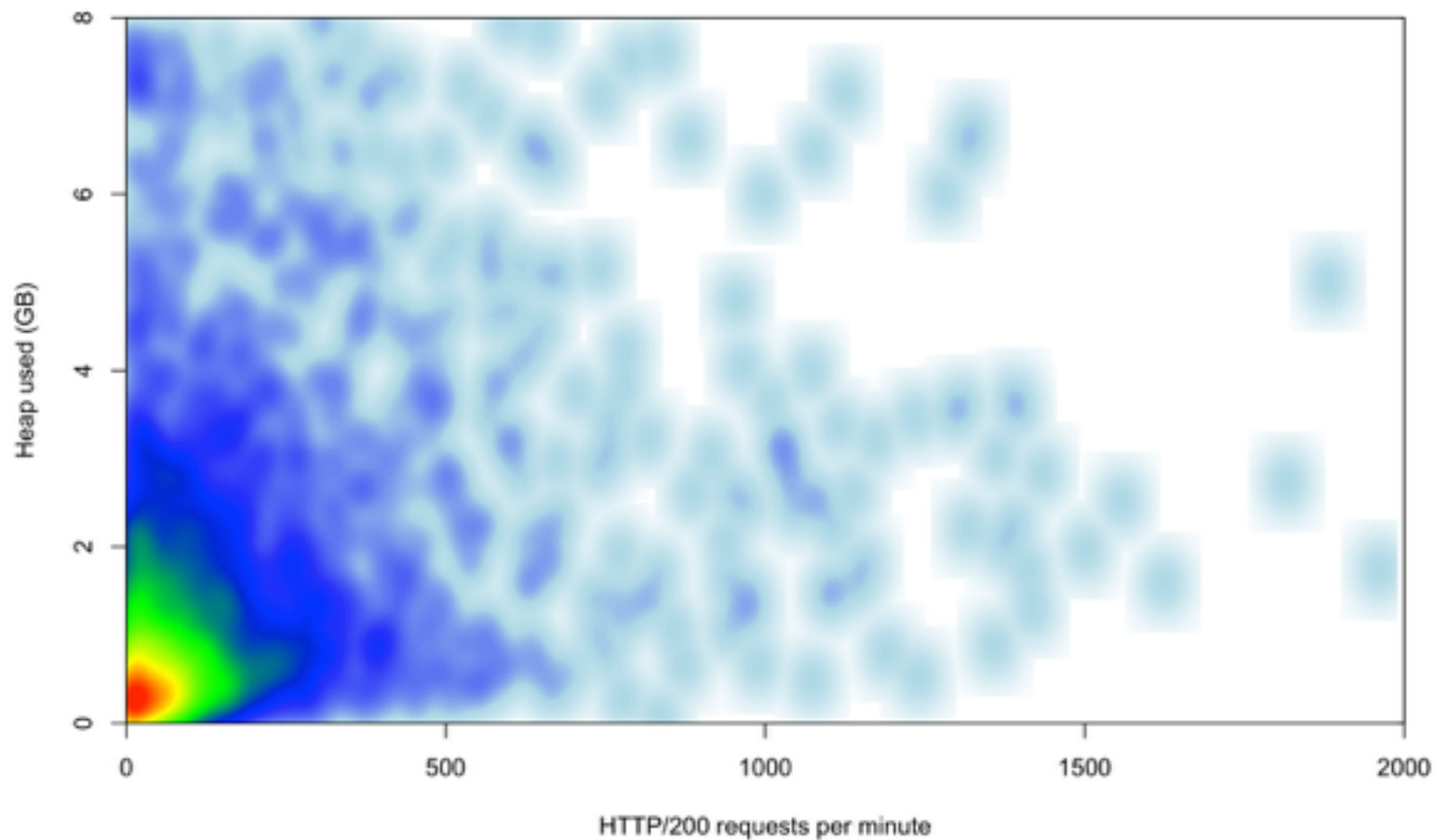
# Heap usage as a function of number of jobs



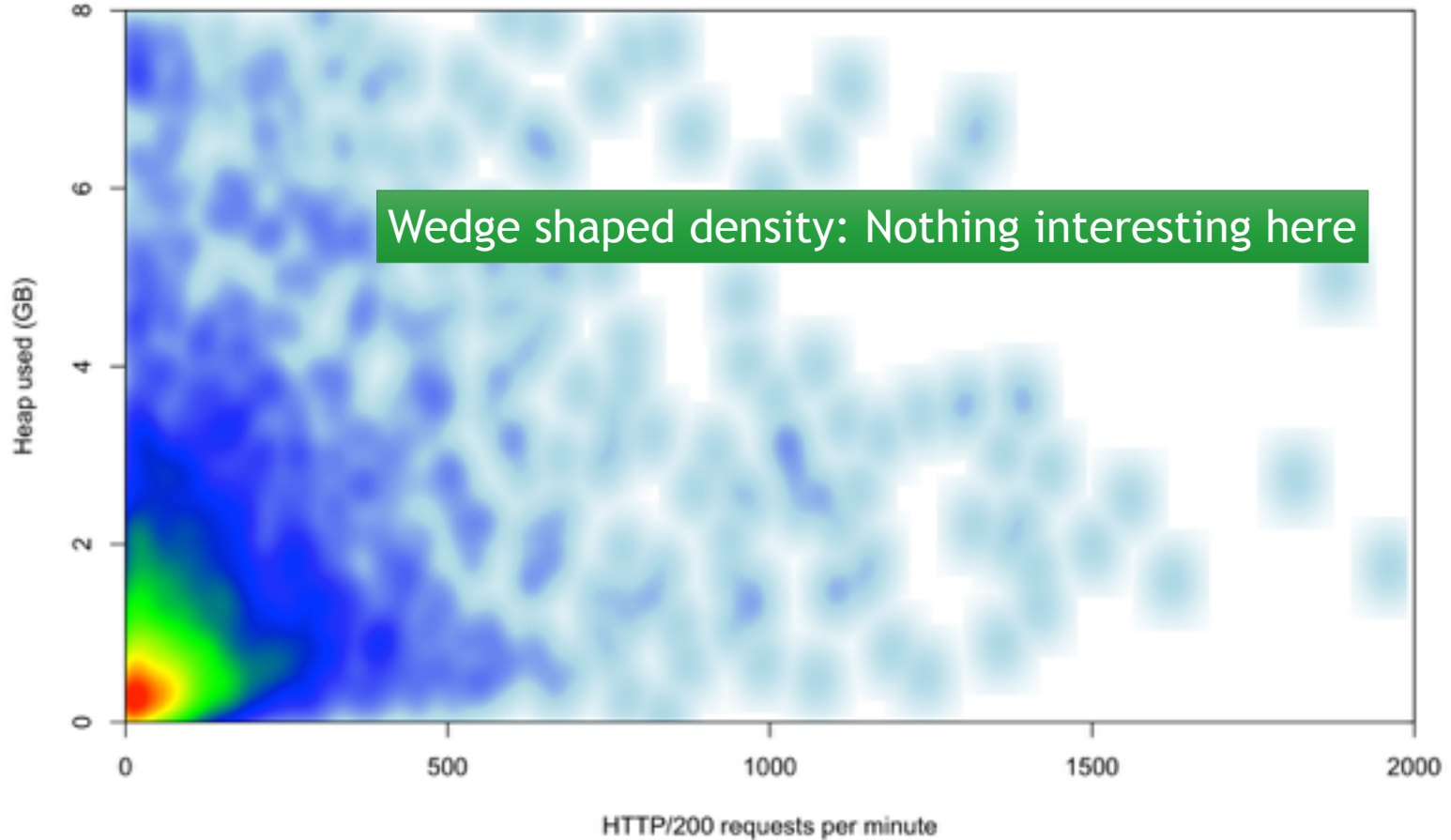
# Heap usage as a function of number of jobs



## Heap usage as a function of web request rate



## Heap usage as a function of web request rate



# Largest



Jenkins World  
2016

- 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

#JenkinsWorld

## Second largest



Jenkins World  
2016

- 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)



Jenkins World  
2016

- 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 🙄



#JenkinsWorld

# Agenda



Jenkins World  
2016



Real world



Theory



Experiments



Applied

# Why scale Jenkins?



Jenkins World  
2016

- 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



#JenkinsWorld

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 the things you have to learn to get right is task allocation. That's just a fancy word for *giving people things to do*. It's known colloquially



## CODING HORROR

programming and human factors

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 you

you  
wor



Susan Weinschenk Ph.D.

Brain Wise

<http://goo.gl/ljLjNZ>

# The True Cost Of Multi-Tasking

You could be losing up to 40% of your productivity



## CODING HORROR

programming and human factors

27 Sep 2006

# The Multitasking Myth

<http://goo.gl/6VSjaz>

ems Thinking, Gerald Weinberg proposed a rule of thumb for project switching:

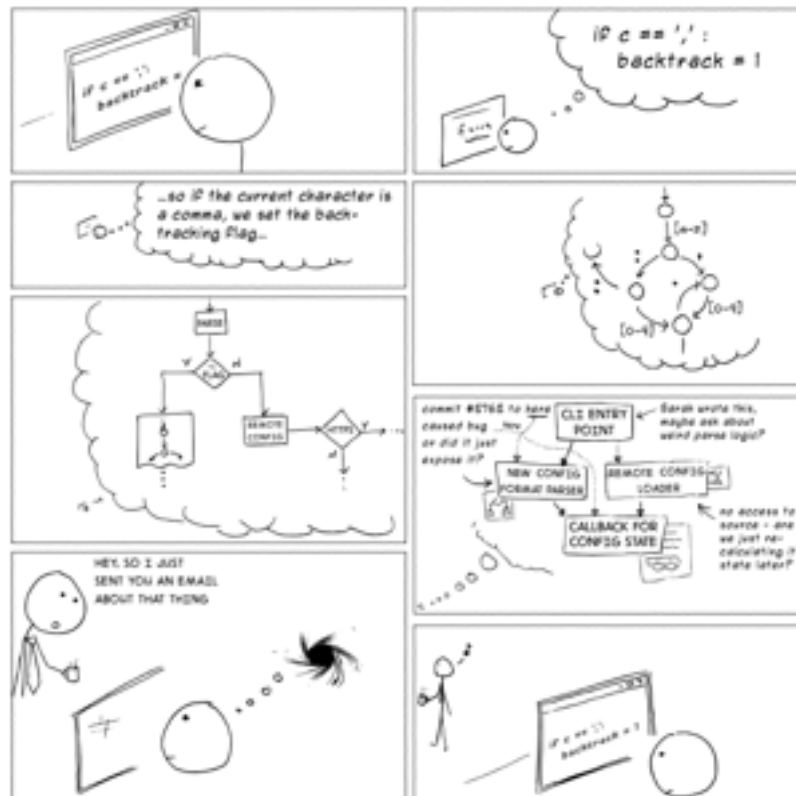
■ Working Time Available Per Project  
■ Loss to Context Switching





# How much does a context switch cost a developer

## THIS IS WHY YOU SHOULDN'T INTERRUPT A PROGRAMMER



# How much does a context switch cost a developer



Jenkins World  
2016

Estimates vary...



# How much does a context switch cost a developer



Jenkins World  
2016

Estimates vary...

“60 minutes”

# How much does a context switch cost a developer



Jenkins World  
2016

Estimates vary...

“30 minutes”

“60 minutes”

# How much does a context switch cost a developer



Jenkins World  
2016

Estimates vary...

“30 minutes”

“45 minutes”

“60 minutes”

# How much does a context switch cost a developer



Jenkins World  
2016

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...



Jenkins World  
2016

- 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

A male sprinter is captured in motion on a red running track. He is wearing a white tank top with red accents, red shorts, and bright green running shoes. He is holding a purple baton in his right hand. The background is blurred, showing a green field and a brick building. Overlaid on the image is white text with a black outline.

Get the build result to the  
developers as fast as possible  
before they context switch to  
the next task



# How?

#JenkinsWorld

# Little's law



Jenkins World  
2016

- Little's Law tells us that the average number of customers in the store  $L$ , is the effective arrival rate  $\lambda$ , 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  $\lambda$ , 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  $Q$

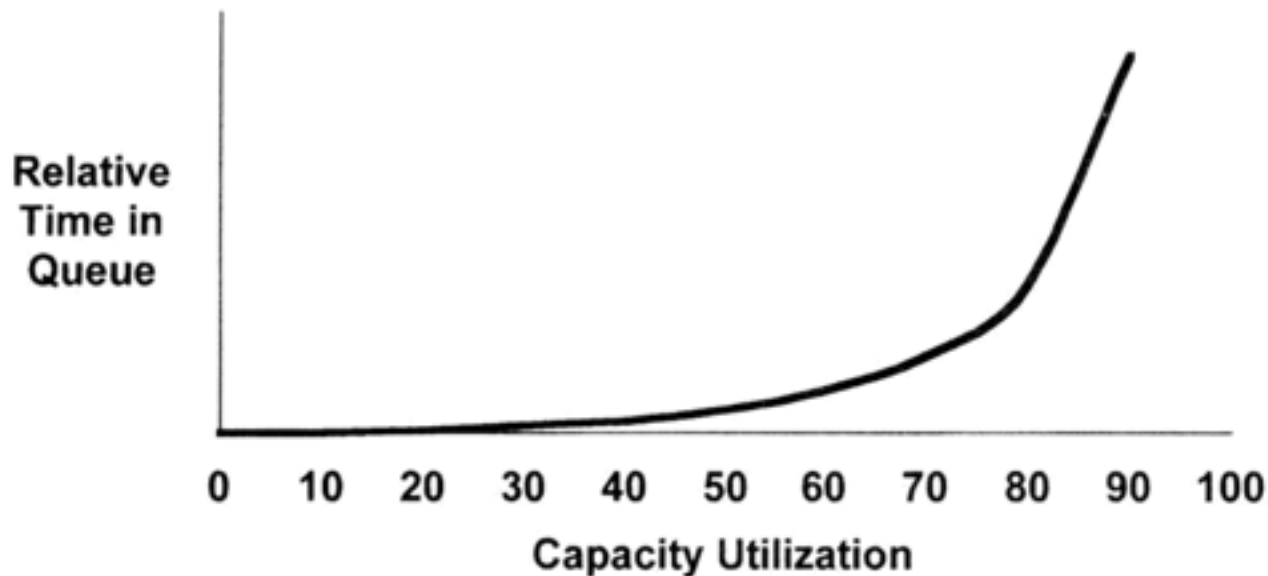


# Time in Queue



Jenkins World  
2016

## Queue Length vs. Capacity Utilization



<http://goo.gl/8MAosu>

*Note:* Assumes M/M/1/∞ Queue

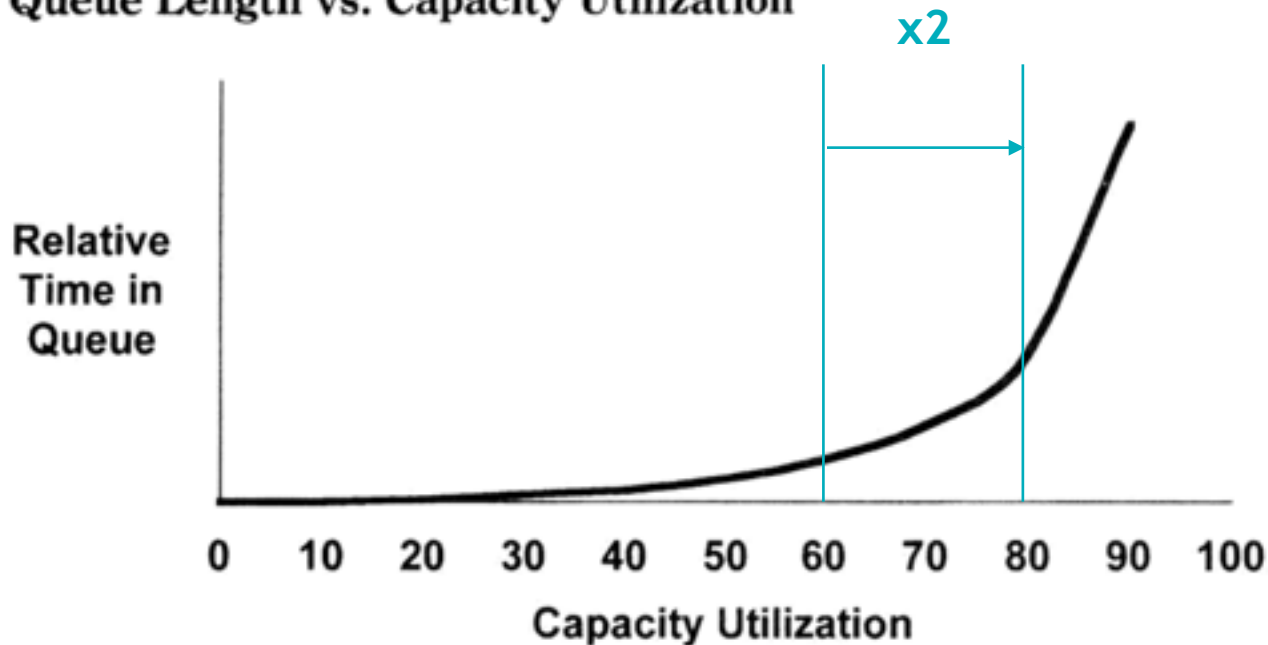
#JenkinsWorld

# Time in Queue



Jenkins World  
2016

## Queue Length vs. Capacity Utilization



<http://goo.gl/8MAosu>

*Note:* Assumes M/M/1/∞ Queue

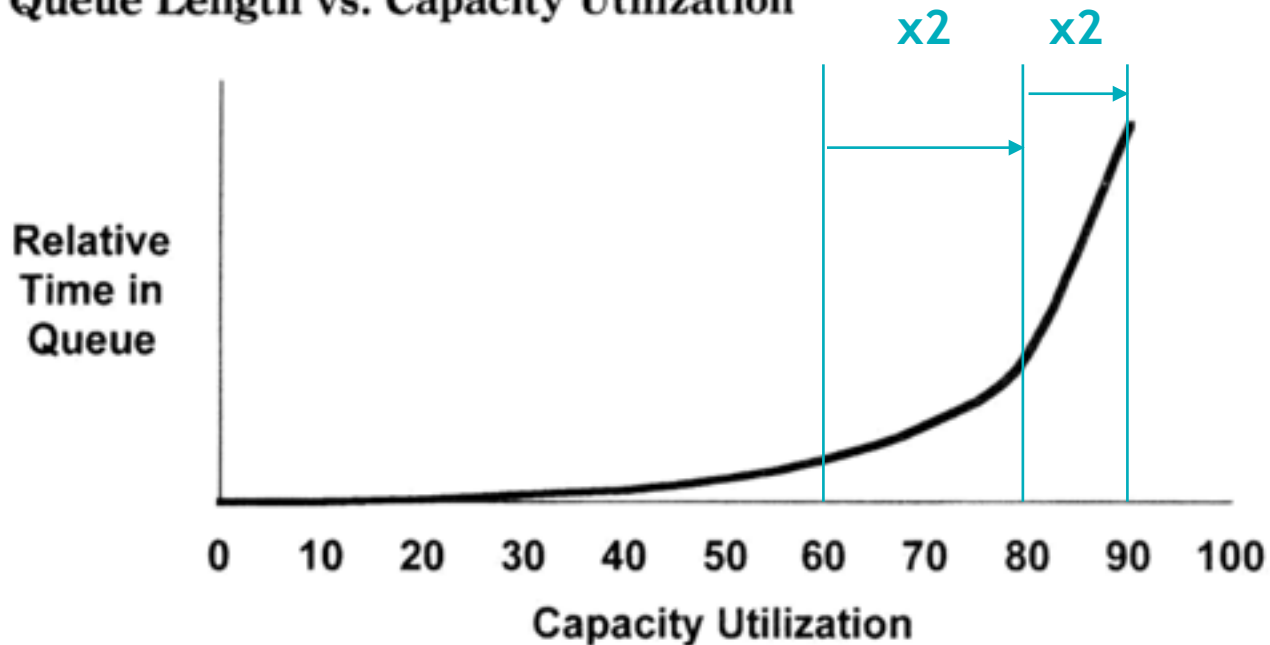
#JenkinsWorld

# Time in Queue



Jenkins World  
2016

## Queue Length vs. Capacity Utilization



<http://goo.gl/8MAosu>

*Note:* Assumes M/M/1/∞ Queue

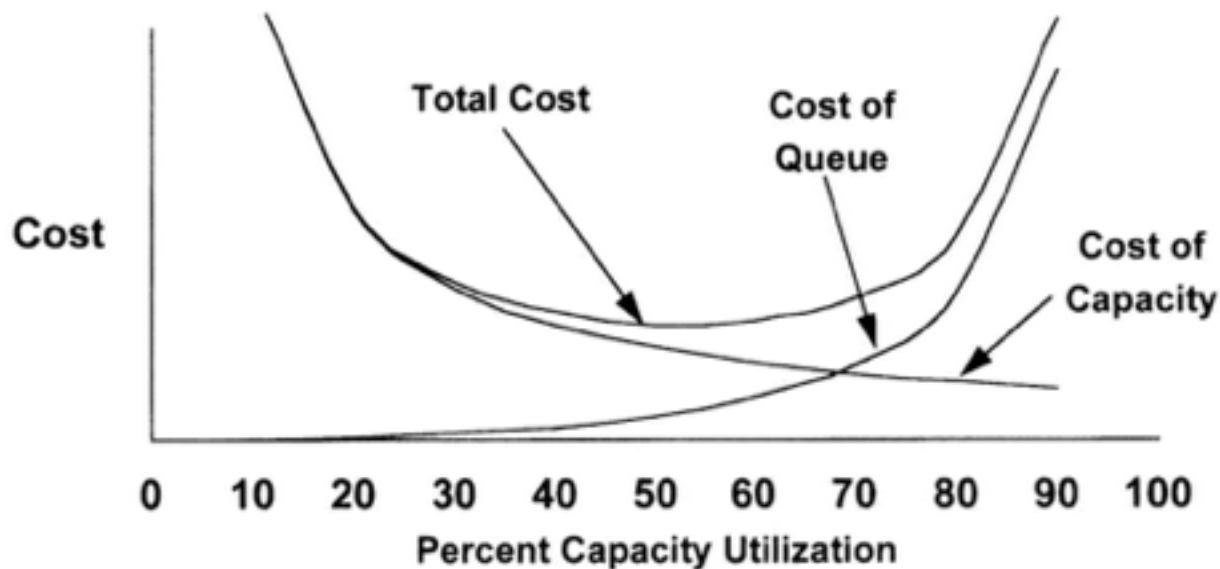
#JenkinsWorld

# Time in Queue



Jenkins World  
2016

## Total Process Cost vs. Capacity Utilization



<http://goo.gl/8MAosu>

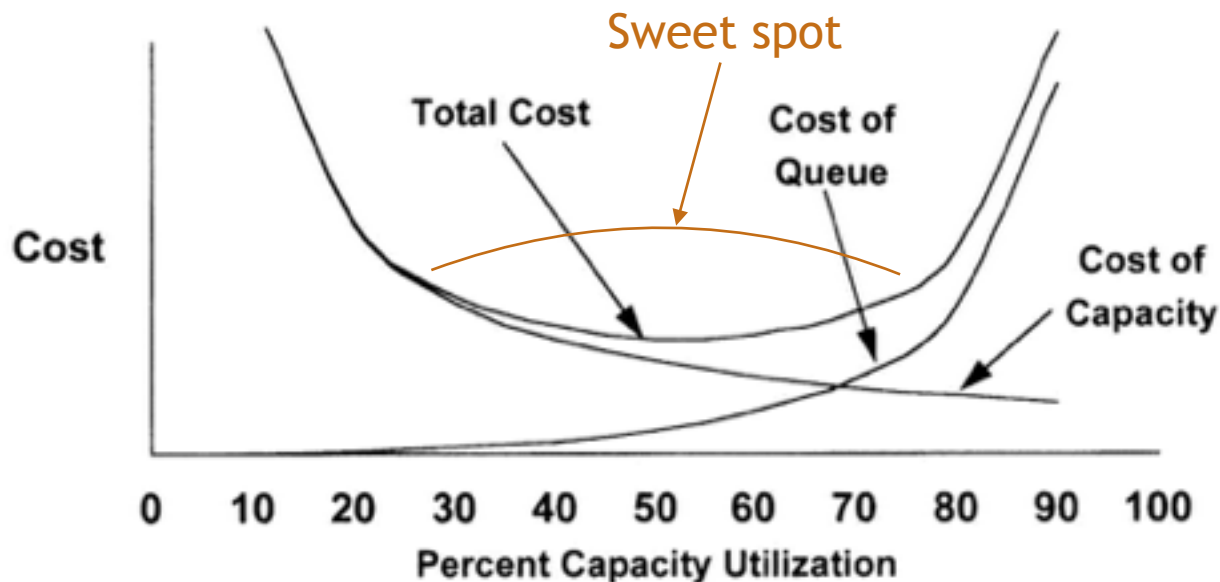
#JenkinsWorld

# Time in Queue



Jenkins World  
2016

## Total Process Cost vs. Capacity Utilization



<http://goo.gl/8MAosu>

Traditional process optimisation for sweet spot

#JenkinsWorld

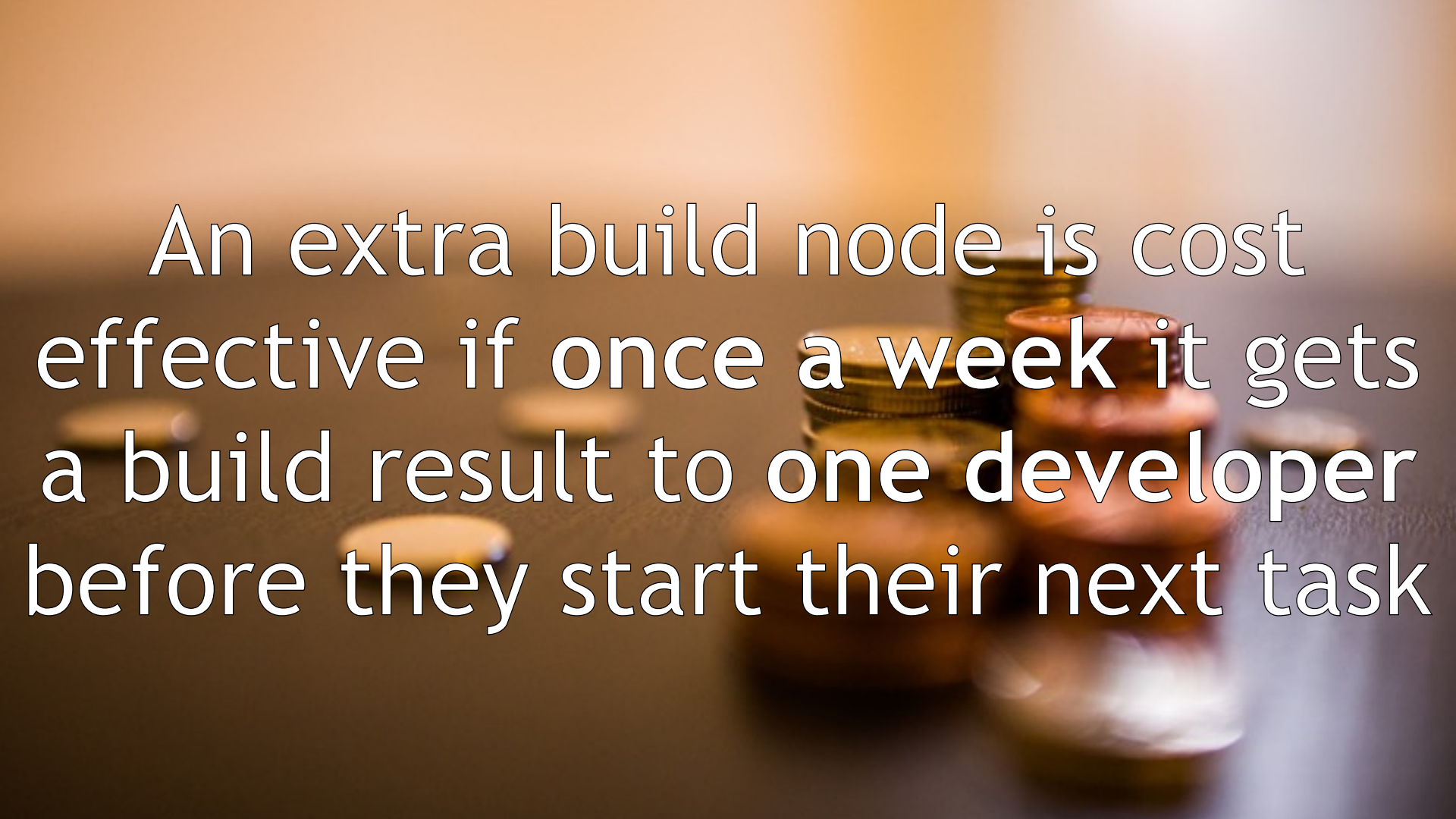
# Time in Queue



Jenkins World  
2016

- 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

A stack of several coins, including gold and copper ones, is placed on a wooden surface. The background is a warm, blurred orange-brown color. The text is overlaid on the image in a white, outlined font.

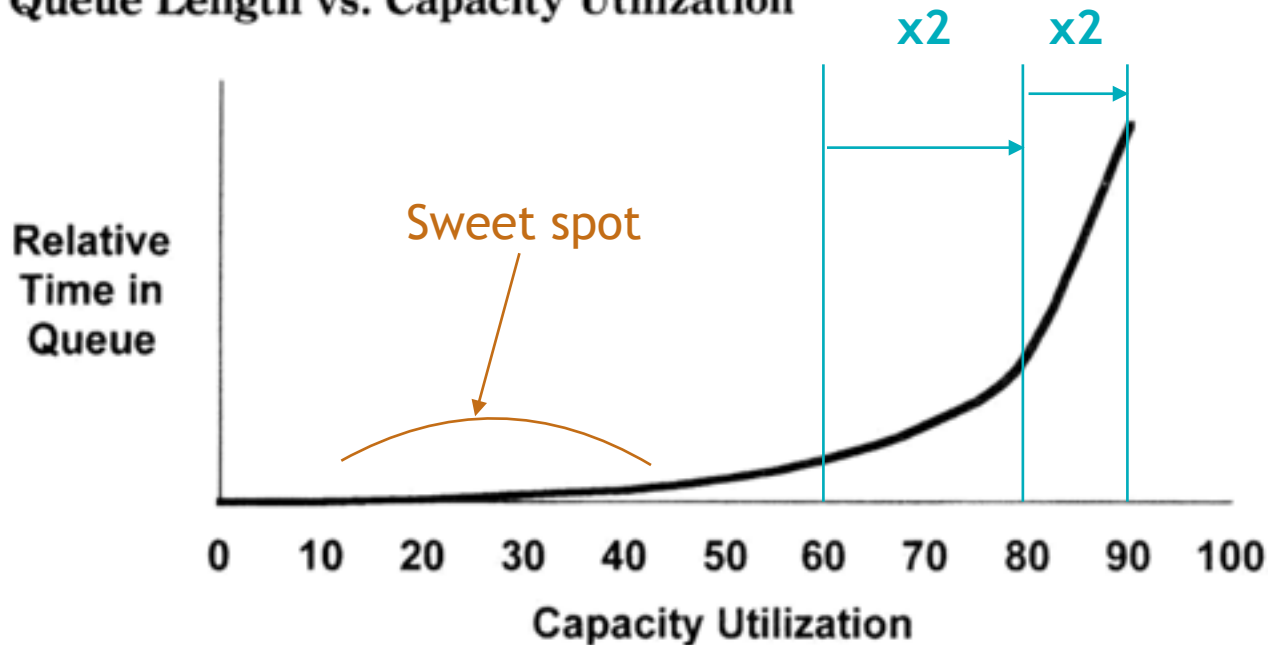
An extra build node is cost effective if once a week it gets a build result to one developer before they start their next task

# Time in Queue



Jenkins World  
2016

## Queue Length vs. Capacity Utilization



<http://goo.gl/8MAosu>

Build server process optimisation for sweet spot

#JenkinsWorld





# Where are these builds coming from?



Jenkins World  
2016

# How often do developers commit?

Most active GitHub users (by contributions). <http://twitter.com/paulmiller>

active-ad Run

## Most active GitHub users ([git.io/top](http://git.io/top))

The count of contributions (summary of Pull Requests, opened issues and commits) to public repos at GitHub.com from Fri, 19 Jun 2015 15:17:38 GMT till Sun, 19 Jun 2016 15:17:38 GMT.

Only first 1000 GitHub users according to the count of followers are taken. This is because of limitations of GitHub search. Sorting algo in pseudocode:

```
githubUsers
  .filter(user => user.followers > 635)
  .sortBy("contributions")
  .slice(0, 256)
```

Made with data mining of GitHub.com ([raw data](#), [script](#)) by @paulmiller with contribs of @ifesinger and @ahmetalpalkan. Updated once per week.

#	User	Contribs	Location	Picture
#1	<a href="#">GrahamCampbell</a> (Graham Campbell)	11088	The United Kingdom	
#2	<a href="#">fabpot</a> (Fabien Potencier)	10000	Paris, France	

<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**

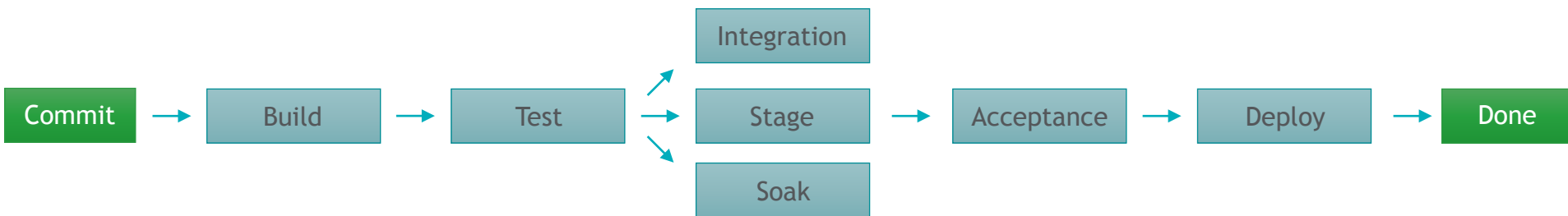
# How long is your build chain?



Jenkins World  
2016



or



?

5 commits per developer per hour

#JenkinsWorld

# 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?



Jenkins World  
2016

For average developers

# How big should my Jenkins be?



Jenkins World  
2016

## For average developers

(not all Graham Campbell)



# How big should my Jenkins be?



Jenkins World  
2016

For average developers

Basic CI usage

# How big should my Jenkins be?



Jenkins World  
2016

For average developers

## Basic CI usage

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



# How big should my Jenkins be?



Jenkins World  
2016

For average developers

With real world test mix

Basic CI usage

# How big should my Jenkins be?



Jenkins World  
2016

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?



Jenkins World  
2016

For average developers

With real world test mix

Basic CI usage

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

# Horizontal



Vertical

Vertical →



Horizontal →

← Eventually you cannot get a bigger box

Vertical →



Horizontal →

← Eventually you cannot get a bigger box



Have to use horizontal scaling anyway



Vertical →

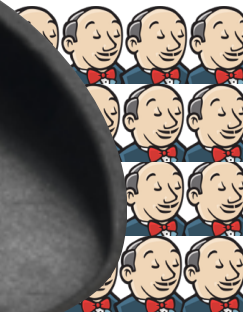
Horizontal →

← Eventually you cannot get a bigger box

Vertical →



anyway



Horizontal →



# Other approaches to scaling...



Jenkins World  
2016

?

#JenkinsWorld

# Other approaches to scaling...



Jenkins World  
2016

- **Microservices**

## Other approaches to scaling...



Jenkins World  
2016

- **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...



Jenkins World  
2016

- **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...



Jenkins World  
2016

- **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?**

A close-up photograph of a bodybuilder's muscular legs, showing the quadriceps and calves, with hands resting on a barbell. The lighting is dramatic, highlighting the muscle definition.

“Jenkins is cron on steroids”

– Lindsay Holmwood



<http://goo.gl/OFjlnB>





## *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



Jenkins World  
2016

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



Jenkins World  
2016

- 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



Jenkins World  
2016

- 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



Jenkins World  
2016

- 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



Jenkins World  
2016

- 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)



Jenkins World  
2016

- 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

#JenkinsWorld



# A platform for serverless applications



Jenkins World  
2016

- 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



Jenkins World  
2016

- 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



Jenkins World  
2016

- 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)



Jenkins World  
2016

- Pros
    - Reduced operational cost
    - Easier operational management
  - Cons
    - ~~Vendor control~~
    - Multitenancy problems
    - ~~Vendor lock-in~~
    - Security concerns
    - Repetition of logic
- } If we remove the special snowflakes

# How do we manage microservices?



Jenkins World  
2016

?

# How do we manage microservices?



Jenkins World  
2016

NETFLIX



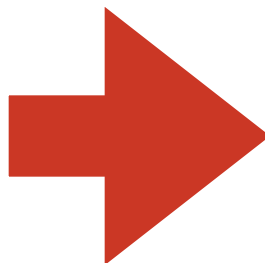
#JenkinsWorld

# How do we manage microservices?



Jenkins World  
2016

NETFLIX

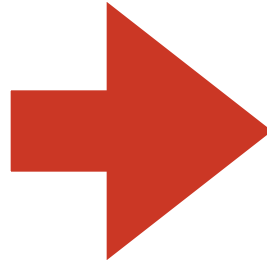


#JenkinsWorld

# Building the Butler Army...









Jenkins World  
2016



#JenkinsWorld



-  [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
 <a href="#">_Jenkins</a>	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)



Jenkins World  
2016

- 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?



#JenkinsWorld

# Agenda



Jenkins World  
2016



Real world



Theory



Experiments



Applied

# Micro-experiments scaling Jenkins



Jenkins World  
2016

- Memory usage
  - Jobs
  - Builds
  - Nodes

#JenkinsWorld

# Micro-experiments scaling Jenkins



Jenkins World  
2016

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

# Micro-experiments scaling Jenkins



Jenkins World  
2016

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

# Micro-experiments scaling Jenkins



Jenkins World  
2016

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



# Micro-experiments scaling Jenkins



Jenkins World  
2016

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

#JenkinsWorld

# Micro-experiments scaling Jenkins



Jenkins World  
2016

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



Measure memory consumption

# Micro-experiments scaling Jenkins



Jenkins World  
2016

- Memory usage

- Jobs
- Builds
- Nodes



Measure memory consumption

- Remoting

- Performance
- Load



Measure scalability

- Queue

- Executor threads
- Archiving artifacts

# Micro-experiments scaling Jenkins



Jenkins World  
2016

- Memory usage

- Jobs
- Builds
- Nodes



Measure memory consumption

- Remoting

- Performance
- Load



Measure scalability

- Queue

- Executor threads
- Archiving artifacts

Limit Experiment

# Micro-experiments scaling Jenkins



Jenkins World  
2016

- Memory usage

- Jobs
- Builds
- Nodes



Measure memory consumption

- Remoting

- Performance
- Load



Measure scalability

- Queue

- Executor threads
- Archiving artifacts



Limit Experiment

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)



Jenkins World  
2016

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

← Freestyle

← Empty builds

← No builds



# Memory usage



Jenkins World  
2016

- 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



Jenkins World  
2016

- 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



Jenkins World

2016

```
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



Jenkins World  
2016

<i>10 req/client/sec</i>	# 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





Jenkins World  
2016

<i>10 req/client/sec</i>	<i># of clients</i>	<i>JVM load</i>	<i>Average heap usage</i>
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



<i>10 req/client/sec</i>	# 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...



Jenkins World  
2016

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

#JenkinsWorld



# Windows vs Everyone else...



Jenkins World  
2016

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

Red arrows point from the JVM load values 1.3 and 1.5 to the '15% difference' box. Green arrows point from the JVM load values 0.41 and 0.41 to the 'No difference' box.

#JenkinsWorld

# Queue



Jenkins World  
2016

- 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



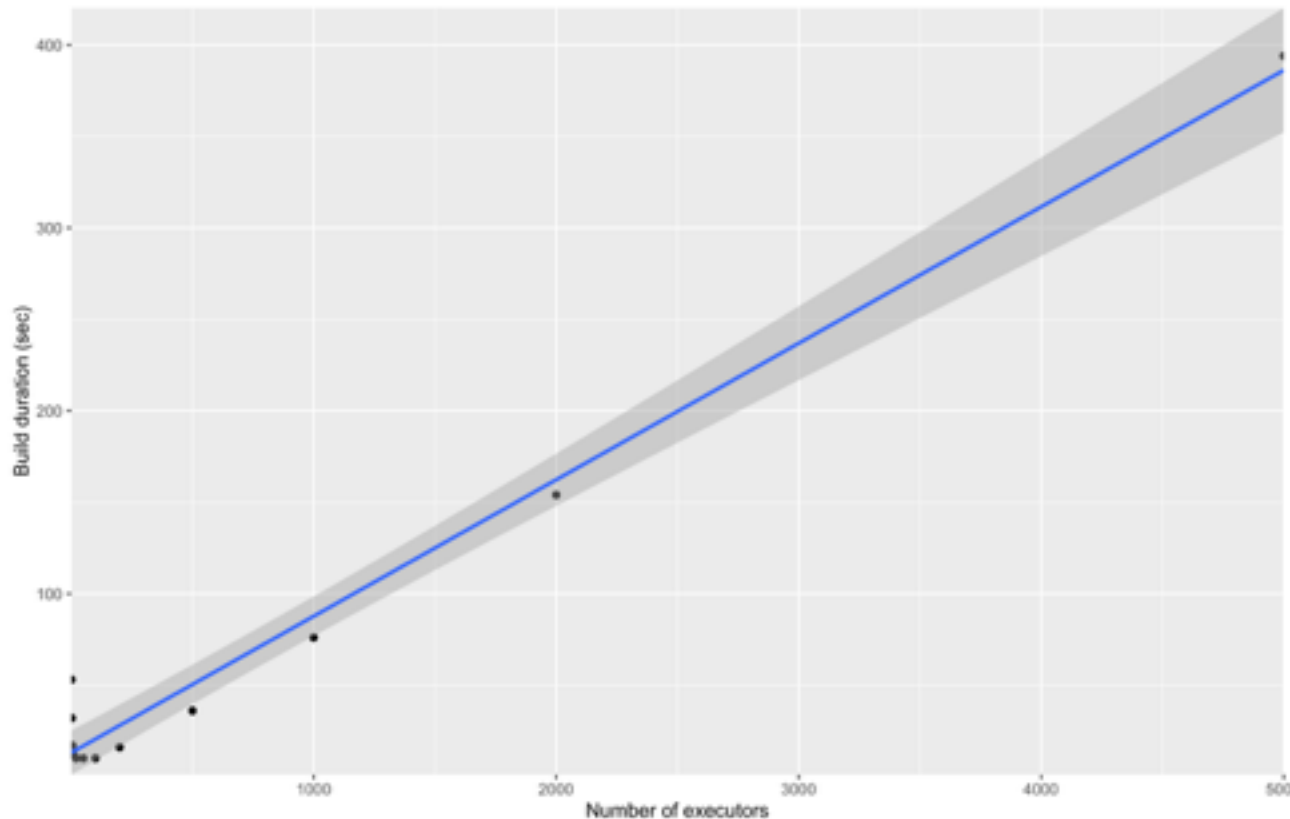
# Demo 2

# Queue



Jenkins World  
2016

Build duration as a function of number of executors



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

#JenkinsWorld

# Executor threads



Jenkins World  
2016

- 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



Jenkins World  
2016

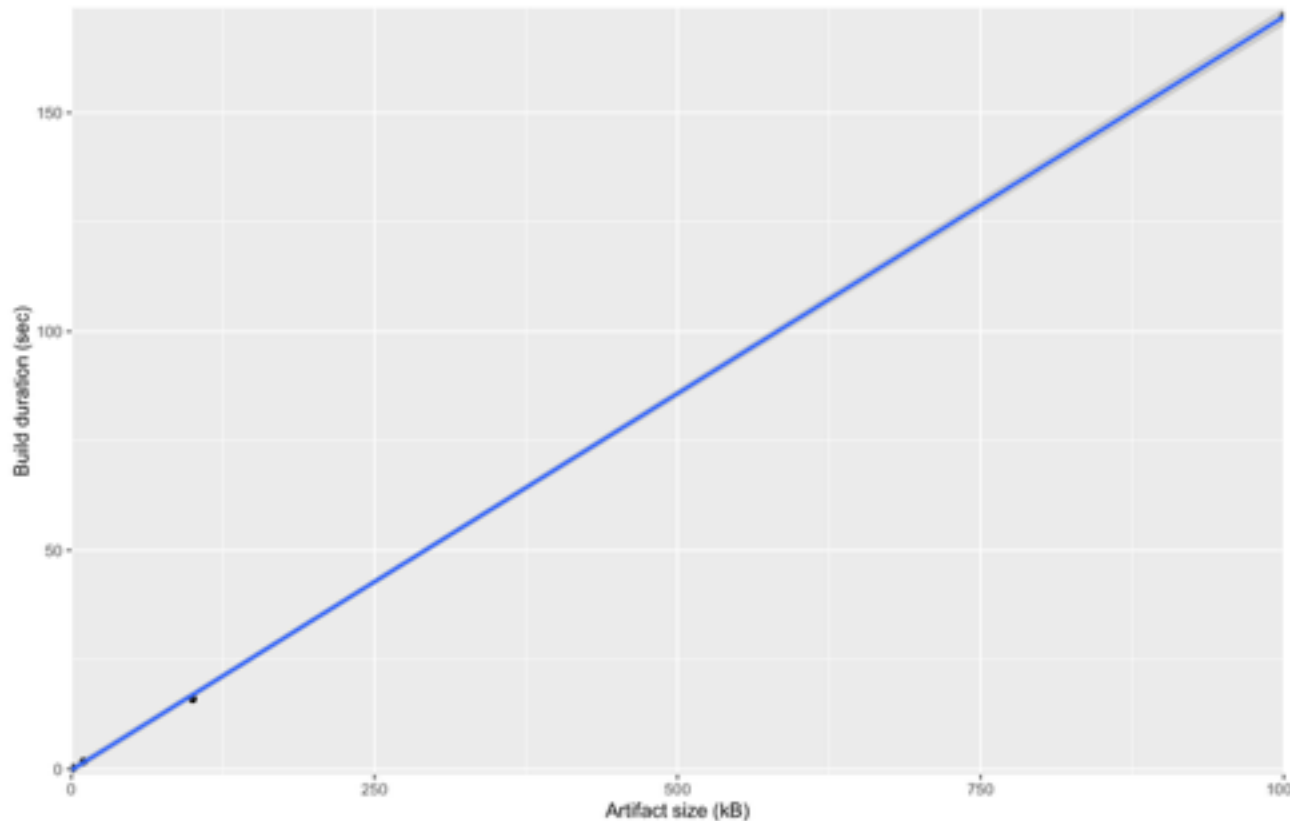
- How much work does archiving artifacts put on the remoting layer?
- Use a custom agent that reports:
  - Number of commands sent and received
  - Number of bytes sent and received
- Run builds with different numbers and sizes of artifacts.

# Vary size of artifact



Jenkins World  
2016

Build duration as a function of artifact size



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

Duration =  $f(\text{size})$

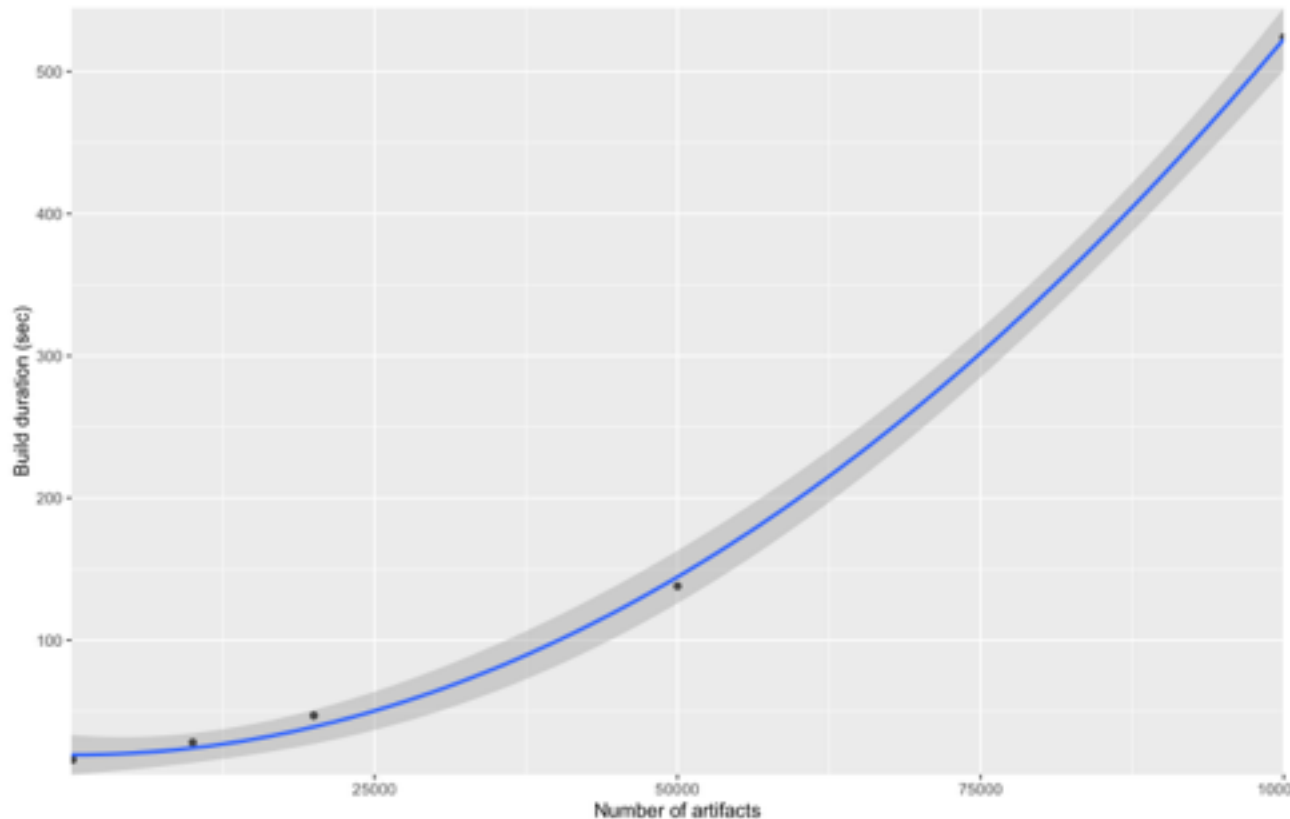
*Linear*

# Vary number of artifacts (total size constant)



Jenkins World  
2016

Build duration as a function of number of artifacts



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

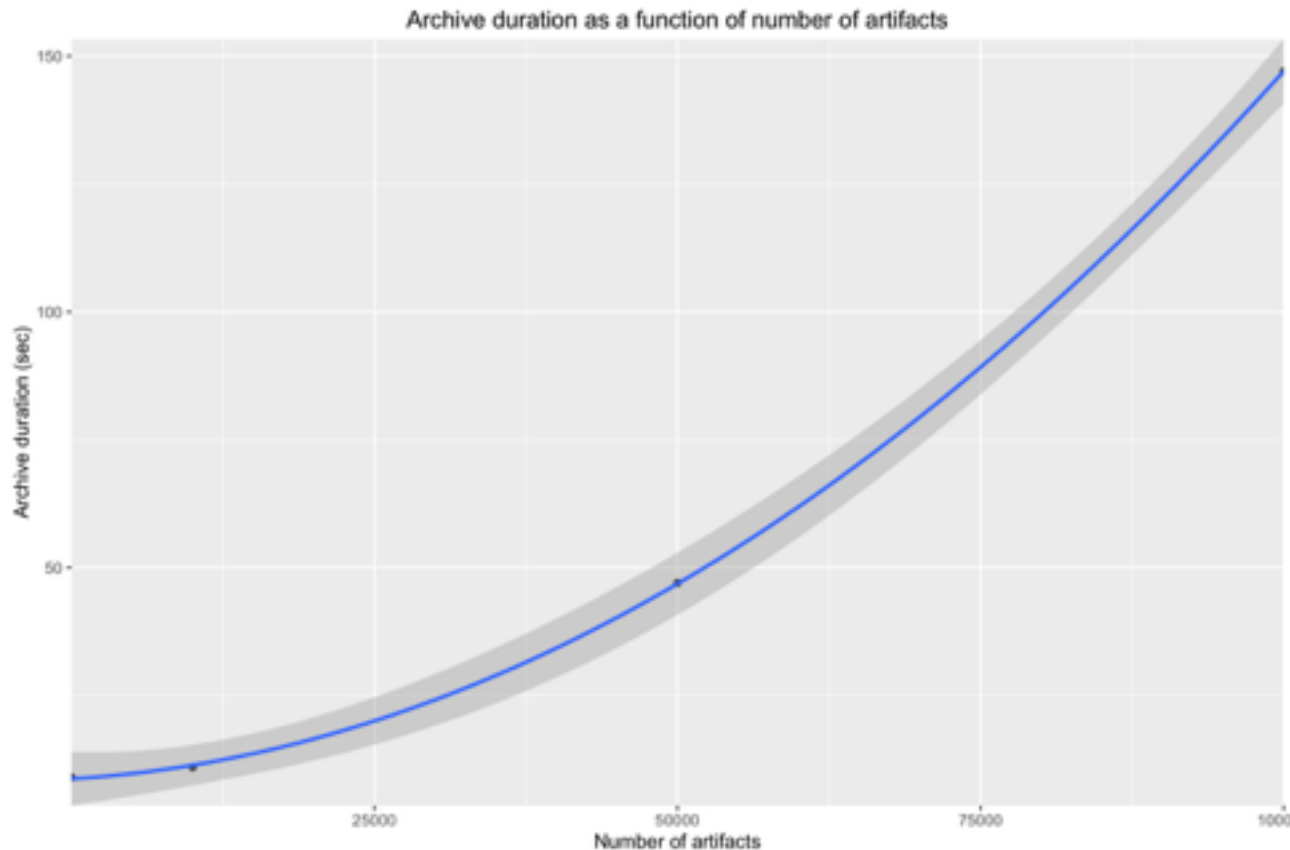
$$\text{Build Duration} = f(n^2)$$



# Vary number of artifacts (total size constant)



Jenkins World  
2016



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

Build Duration =  $f(n^2)$

Archive Duration =  $f(n^2)$

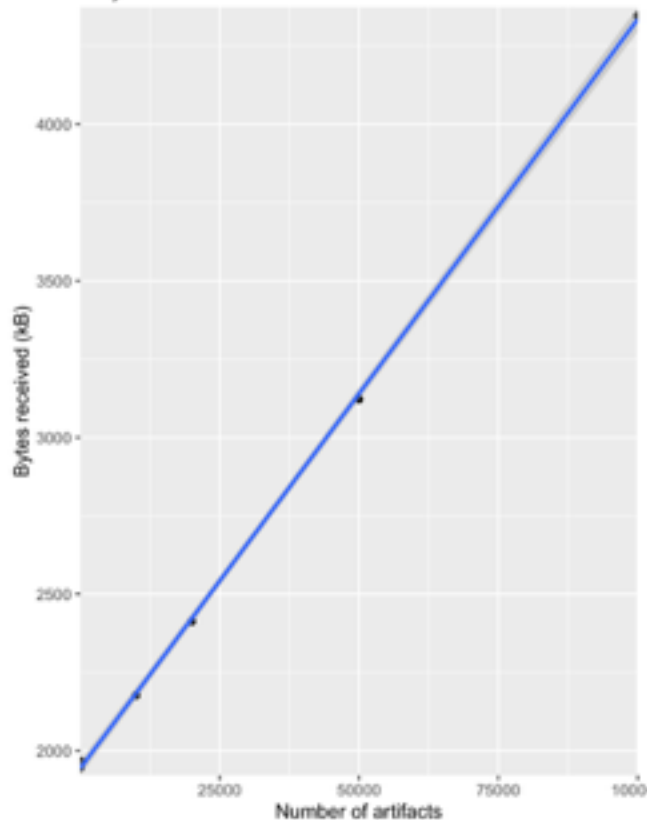
$O(n^2)$

#JenkinsWorld

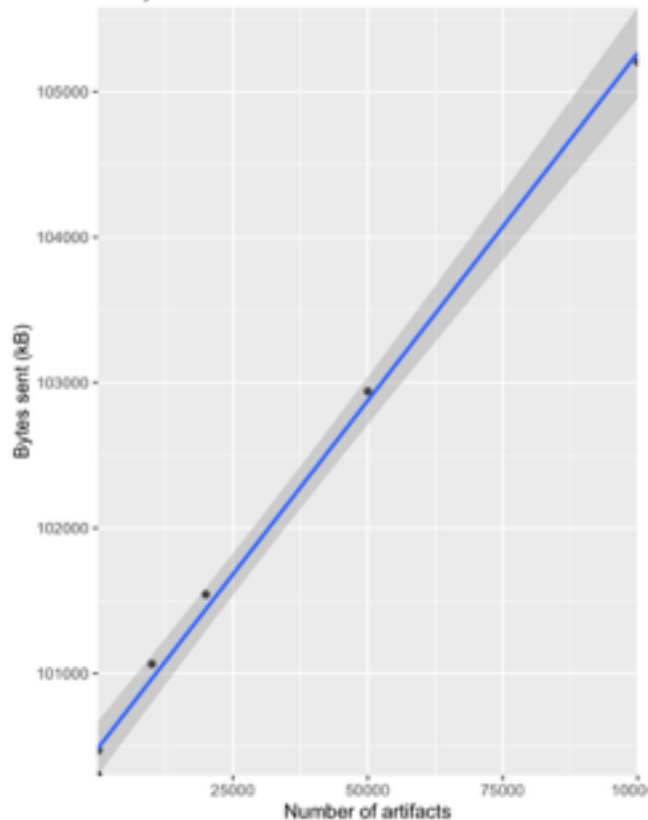
# Vary number of artifacts (total size constant)



Bytes received as a function of number of artifacts



Bytes sent as a function of number of artifacts



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

# Summary (experiments)



Jenkins World  
2016

- Memory usage
  - Jobs: starting from 100kB each
  - Builds: starting from 2kB each, 30kB for ~~Maven~~<sup>Evil</sup> 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  $\approx$   $\frac{1}{2}$  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



#JenkinsWorld

# Agenda



Jenkins World  
2016



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)

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

$a_{\text{JNLP}}$  Number of JNLP agents

$a_{\text{SSH}}$  Number of SSH agents

$j$  Number of jobs

$b/j$  Average builds per job

$C$  Number of concurrent builds

# Validating the JVM heap sizing guide...



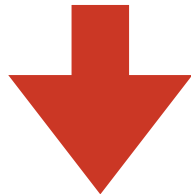
Jenkins World  
2016

- 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...

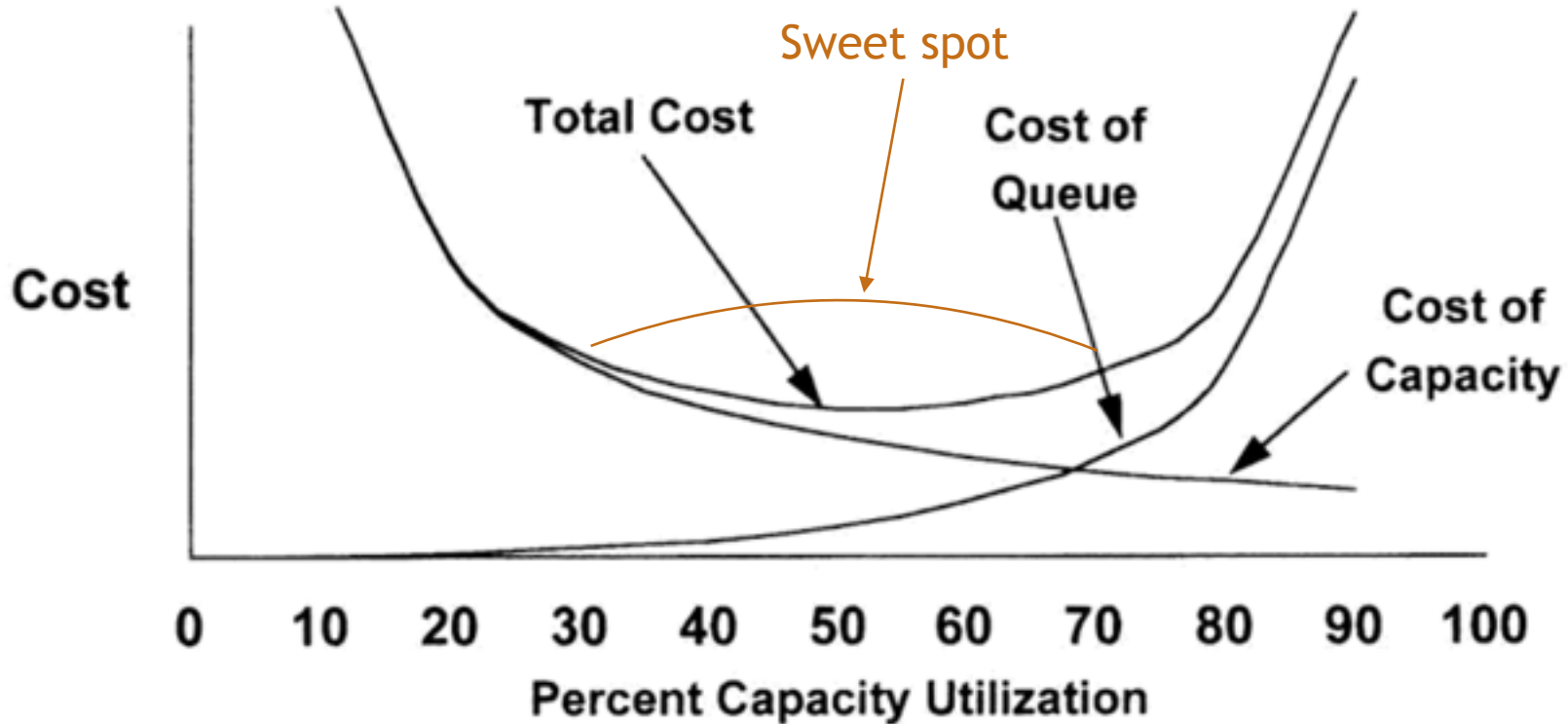
- Support bundle taken after 40h of uptime
  - 1,104 nodes configured
  - 698 nodes on-line
  - 7,135 executors configured
  - 4,828 executors on-line
  - 5,784 jobs (92% freestyle)



Average usage  $\approx$  2200MB

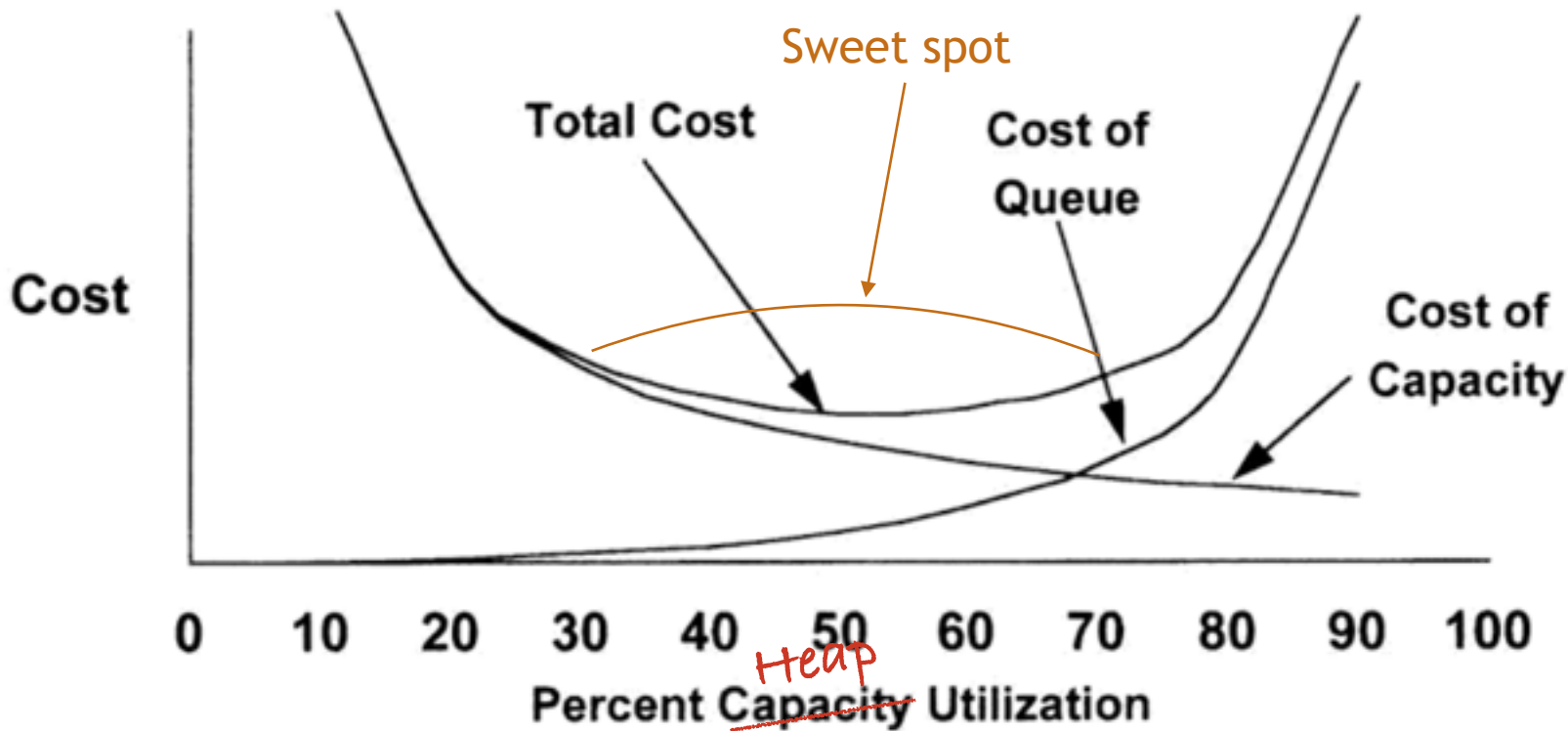
- Master: Java 8
  - Xms4096m -Xmx4096m
  - XX:NewSize=2048m -XX:MaxNewSize=2048m
  - XX:ParallelGCThreads=4 -XX:ConcGCThreads=4
  - Dhudson.slaves.ChannelPinger.pingInterval=-1
- Agents running Java 7/8:
  - Majority SSH:
    - Xms128m -Xmx512m
    - Dhudson.remoting.Launcher.pingTimeoutSec=600
    - Dhudson.remoting.Launcher.pingIntervalSec=1200
  - 5% SSH:
    - Xms128m -Xmx512m
    - Dhudson.remoting.Launcher.pingIntervalSec=-1
  - 1% JNLP (mostly Windows)

# Total Process Cost vs. Capacity Utilization

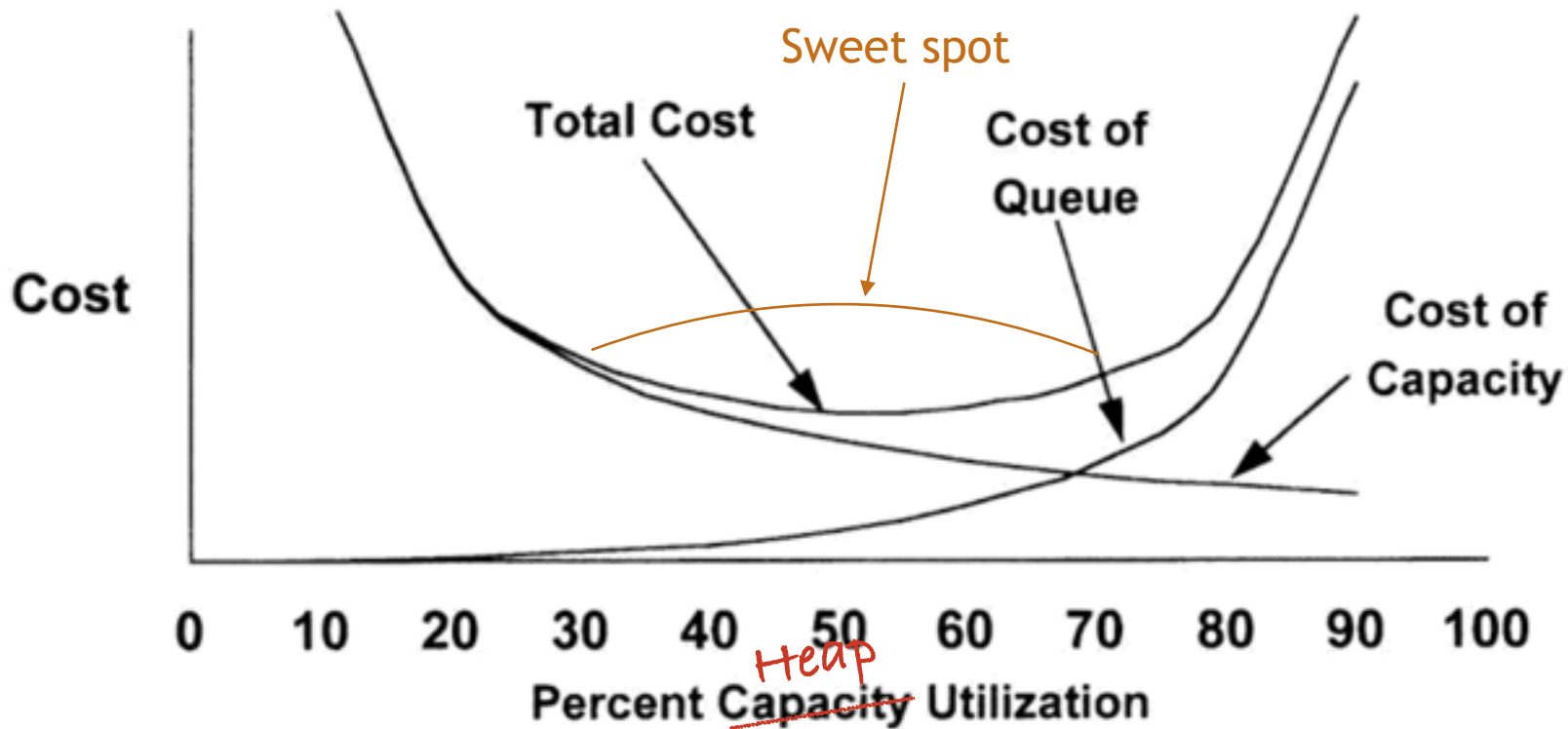




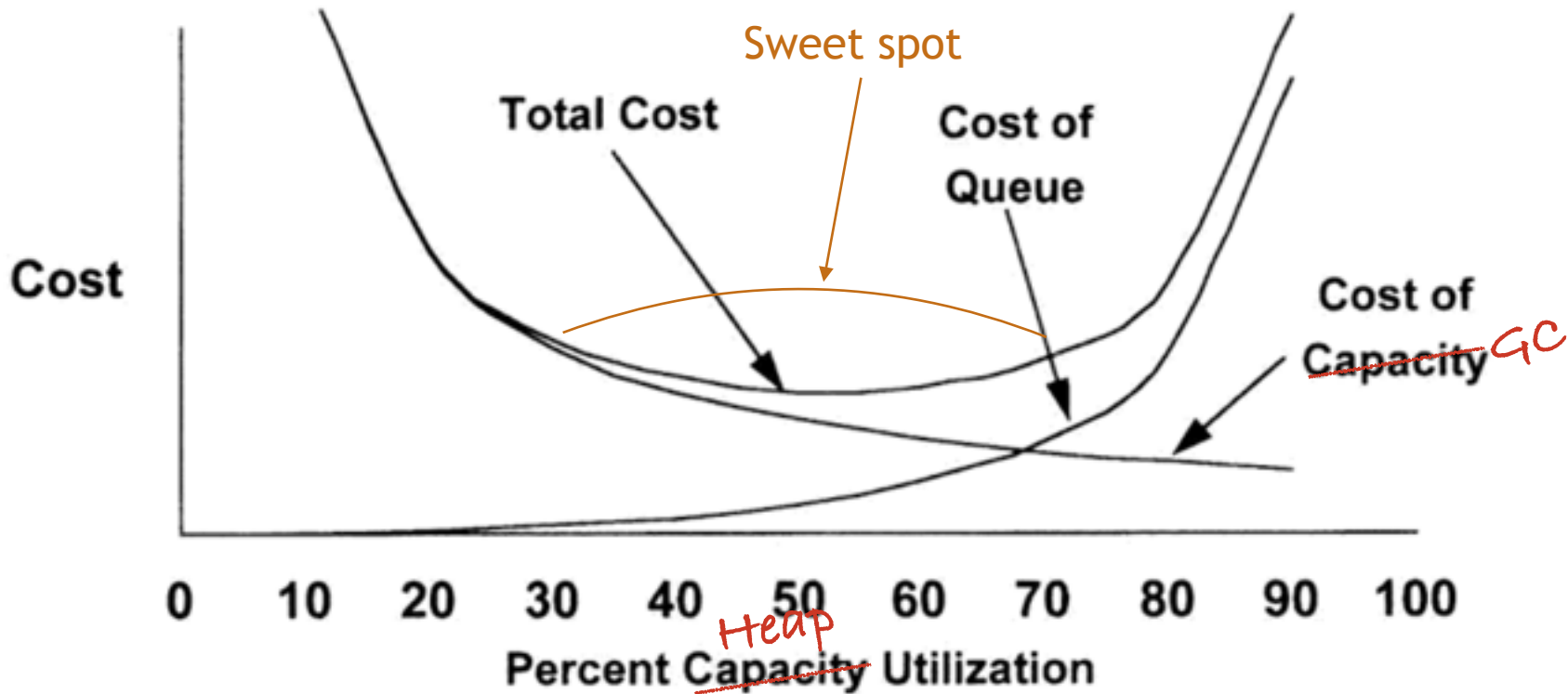
# Total Process Cost vs. ~~Capacity~~ Utilization



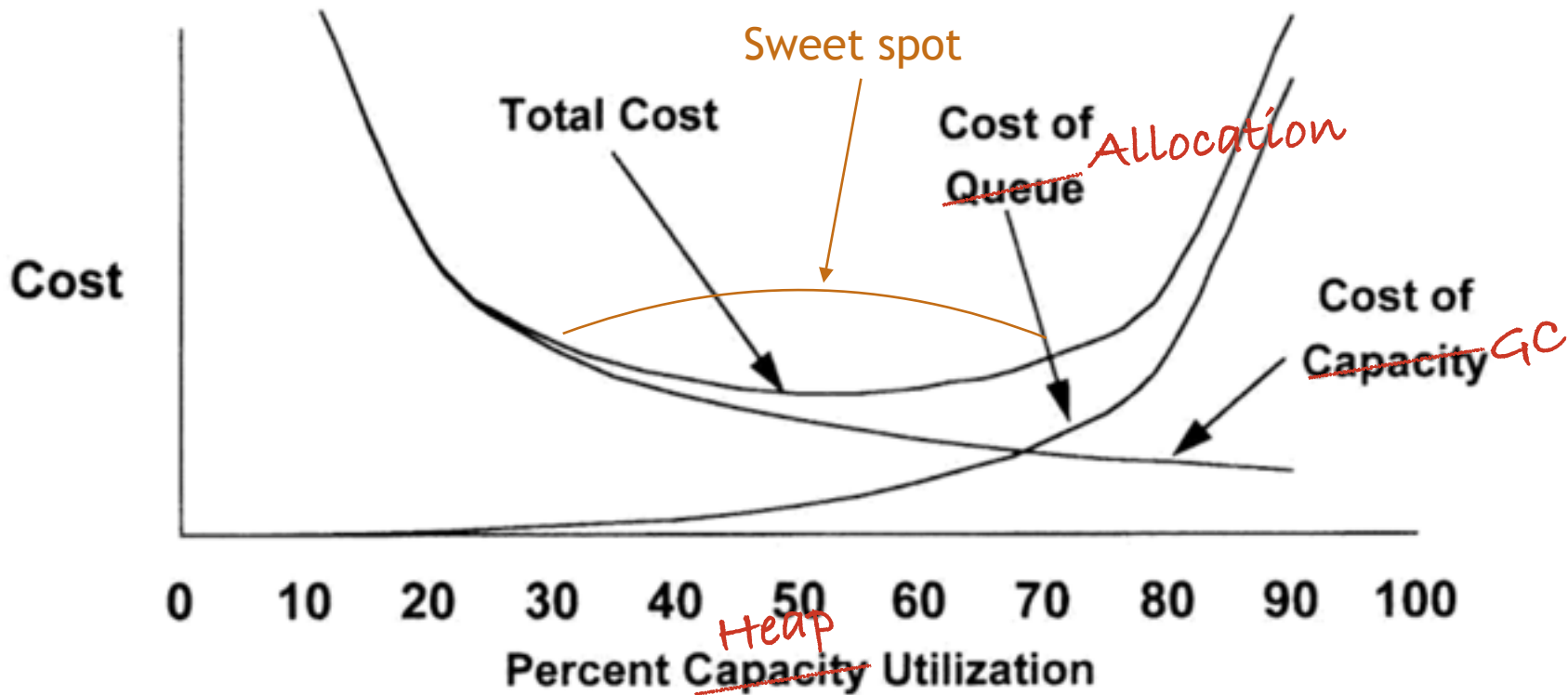
~~CPU~~ ~~Heap~~  
**Total Process Cost vs. Capacity Utilization**



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



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





Jenkins World  
2016

## 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  $\approx$  2200MB

```
-Xms4096m -Xmx4096m
```



#JenkinsWorld

# Instance sizing



Jenkins World  
2016

- 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



Jenkins World  
2016

- ☑ 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



Jenkins World  
2016

- ☑ 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