So you want to build the worlds largest Jenkins cluster?

Stephen Connolly
So you want to build the world’s biggest Jenkins Cluster

Stephen Connolly
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

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

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

Real world

Theory

Experiments

Applied

#JenkinsWorld
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...
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 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 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
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!!!
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?
• If we want the most number of jobs...
• If we want the most number of jobs...
• Install Mock Load Builder plugin

```java
java -jar jenkins-cli.jar \create-mock-load-jobs 200000
```
The world’s biggest Jenkins!!!

200,001 jobs!!!
I may need another of these
If we want the most number of executors...

# of executors
10000

Labels

Usage
Use this node as much as possible
The world’s biggest Jenkins!!!

10,005 executors!!!

200,001 jobs!!!
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...
A big Jenkins “cluster”!!!
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

#JenkinsWorld
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

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

```java
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
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
• 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 usage as a function of number of executors

Tuned the JVM heap
Heap usage as a function of number of executors

Throw memory at Jenkins
Heap usage as a function of number of jobs
Heap usage as a function of number of jobs

Wedge shaped density: Nothing interesting here
Heap usage as a function of web request rate
Wedge shaped density: Nothing interesting here
• 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

#JenkinsWorld
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

“640k ^ ought to be enough for anybody”

– Bill Gates
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"
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 as project management, but the reality is it's much more subtle than that. Project management is about understanding how people work, and how to allocate their time and effort in a way that maximizes productivity.

The Multitasking Myth

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

- Working Time Available Per Project
- Loss to Context Switching

http://goo.gl/mEcQpB

http://goo.gl/6VSjaz
Human Task Switches Considered Harmful
by Joel Spolsky

Monday, February 12, 2001
When you’re managing a team of programmers, one of the most important things is to try to keep them working on one thing at a time. This is not just about keeping an eye on the work in progress, but about making sure that the work that’s being done is relevant.

Coding Horror
programming and human factors

The Multitasking Myth

27 Sep 2006

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
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”

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

“60 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
Get the build result to the developers as fast as possible before they context switch to the next task

by Mark Sadowski http://goo.gl/Ior3rY CC BY-SA 2.0
How?
Little’s law

• 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 $Q$
Queue Length vs. Capacity Utilization

Note: Assumes M/M/1/∞ Queue

http://goo.gl/8MAosu
Queue Length vs. Capacity Utilization

Note: Assumes M/M/1/∞ Queue

http://goo.gl/8MAosu
Time in Queue

Queue Length vs. Capacity Utilization

Note: Assumes M/M/1/∞ Queue

http://goo.gl/8MAosu
Time in Queue

http://goo.gl/8MAosu
Time in Queue

Traditional process optimisation for sweet spot

http://goo.gl/8MAosu
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

Queue Length vs. Capacity Utilization

Relative Time in Queue

Capacity Utilization

Sweet spot

Build server process optimisation for sweet spot

http://goo.gl/8MAosu
Where are these builds coming from?
How often do developers commit?

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

http://git.io/top
How long is your build chain?

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
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

Basic CI usage

With real world test mix

At least 1 Jenkins master per 200 developers

(to maximise developer productivity)
Horizontal

Vertical

“The Interlace, Singapore” by Mike Cartmell http://goo.gl/zDzGLs CC BY 2.0
Vertical ➝ Horizontal ➝
Eventually you cannot get a bigger box
Eventually you cannot get a bigger box

Have to use horizontal scaling anyway
Eventually you cannot get a bigger box

This is Old Hat

Vertical

Horizontal
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
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
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
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

#JenkinsWorld
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

#JenkinsWorld
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

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

If we remove the special snowflakes
How do we manage microservices?
How do we manage microservices?

Netflix

SABIAN ARMY
How do we manage microservices?

Netflix
Building the Butler Army...
Chaos Butler

The next victim is due to be selected in 0 ms

Recent victims

<table>
<thead>
<tr>
<th>Victim</th>
<th>When</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jenkins</td>
<td>Wed Aug 31 17:01:06 IST 2016</td>
</tr>
</tbody>
</table>

Build Queue

No builds in the queue.

Build Executor Status

- master
  - 1 Idle
  - 2 Idle
- jnlp
  - 1 Idle
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
- Remoting
  - Performance
  - Load
- Queue
- Executor threads
- Archiving artifacts

Measure memory consumption

Measure scalability
Micro-experiments scaling Jenkins

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

Measure memory consumption

Measure scalability

Limit Experiment
Micro-experiments scaling Jenkins

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

Measure memory consumption

Measure scalability

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)

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

← 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
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

<table>
<thead>
<tr>
<th>10 req/client/sec</th>
<th># of clients</th>
<th>JVM load</th>
<th>Average heap usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>JNLP2</td>
<td>100</td>
<td>1.3</td>
<td>740 ± 140</td>
</tr>
<tr>
<td>JNLP3</td>
<td>100</td>
<td>1.3</td>
<td>800 ± 190</td>
</tr>
<tr>
<td>JNLP4 (TLS)</td>
<td>100</td>
<td>1.9</td>
<td>225 ± 50</td>
</tr>
<tr>
<td>JNLP4 (plaintext)</td>
<td>100</td>
<td>1.5</td>
<td>200 ± 40</td>
</tr>
</tbody>
</table>
Constraining heap to 512Mb

<table>
<thead>
<tr>
<th>10 req/client/sec</th>
<th># of clients</th>
<th>JVM load</th>
<th>Average heap usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>JNLP2</td>
<td>100</td>
<td>1.3</td>
<td>450 ± 30</td>
</tr>
<tr>
<td>JNLP3</td>
<td>100</td>
<td>1.3</td>
<td>512 ± 0</td>
</tr>
<tr>
<td>JNLP4 (TLS)</td>
<td>100</td>
<td>1.9</td>
<td>222 ± 40</td>
</tr>
<tr>
<td>JNLP4 (plaintext)</td>
<td>100</td>
<td>1.5</td>
<td>166 ± 45</td>
</tr>
</tbody>
</table>
### Constraining heap to 128Mb

<table>
<thead>
<tr>
<th>10 req/client/sec</th>
<th># of clients</th>
<th>JVM load</th>
<th>Average heap usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>JNLP2</td>
<td>100</td>
<td>1.3</td>
<td>125 ± 1</td>
</tr>
<tr>
<td>JNLP3</td>
<td>100</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>JNLP4 (TLS)</td>
<td>100</td>
<td>1.9</td>
<td>117 ± 4</td>
</tr>
<tr>
<td>JNLP4 (plaintext)</td>
<td>100</td>
<td>1.5</td>
<td>97 ± 2</td>
</tr>
</tbody>
</table>
### Windows vs Everyone else...

<table>
<thead>
<tr>
<th>10 req/client/sec</th>
<th># of clients</th>
<th>JVM load</th>
<th>Average heap usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>JNLP2 *nix</td>
<td>100</td>
<td>1.3</td>
<td>740 ± 140</td>
</tr>
<tr>
<td>JNLP4 (plain) *nix</td>
<td>100</td>
<td>1.5</td>
<td>200 ± 40</td>
</tr>
<tr>
<td>JNLP2 win</td>
<td>100</td>
<td>0.41</td>
<td>540 ± 33</td>
</tr>
<tr>
<td>JNLP4 (plain) win</td>
<td>100</td>
<td>0.41</td>
<td>125 ± 33</td>
</tr>
</tbody>
</table>
## Windows vs Everyone else…

<table>
<thead>
<tr>
<th></th>
<th># of clients</th>
<th>JVM load</th>
<th>Average heap usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>JNLP2 *nix</td>
<td>100</td>
<td>1.3</td>
<td>740 ± 140</td>
</tr>
<tr>
<td>JNLP4 (plain) *nix</td>
<td>100</td>
<td>1.5</td>
<td>200 ± 40</td>
</tr>
<tr>
<td>JNLP2 win</td>
<td>100</td>
<td>0.41</td>
<td>540 ± 33</td>
</tr>
<tr>
<td>JNLP4 (plain) win</td>
<td>100</td>
<td>0.41</td>
<td>125 ± 33</td>
</tr>
</tbody>
</table>

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
- 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

\[ \text{Duration} = f(\text{size}) \]

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 = \( f(n^2) \)

\( \mathcal{O}(n^2) \)
Vary number of artifacts (total size constant)

- Bytes received \( \approx 24n + \frac{v}{50} \)
- Bytes sent \( \approx 50n + v \)
- 24 byte ACK for each file
- 150 byte ACK for each 8kB packet
- \( \sim 50 \) 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 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)

\[
512 + \frac{40a_{JNLP} + 240a_{SSH} + 2j(10 + \frac{b}{j}) + 10c}{200}
\]

\(a_{JNLP}\) Number of JNLP agents
\(a_{SSH}\) Number of SSH agents
\(j\) Number of jobs
\(b/j\) Average builds per job
\(c\) Number of concurrent builds

#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)

- 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)

#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)

- 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)

Average usage ≈ 2200MB

#JenkinsWorld
Total Process Cost vs. Capacity Utilization

Cost vs. Percent Capacity Utilization

- Total Cost
- Cost of Queue
- Cost of Capacity

Sweet spot
Total Process Cost vs. Capacity Utilization

Cost

Percent Capacity Utilization

Total Cost

Cost of Queue

Cost of Capacity

Sweet spot

#JenkinsWorld
Total Process Cost vs. Capacity Utilization

- CPU
- Heap

Sweet spot

Cost vs. Percent Capacity Utilization

Cost of Queue

Cost of Capacity

#JenkinsWorld
Total Process Cost vs. Capacity Utilization

- CPU
- Heap
- GC

Sweet spot

#JenkinsWorld
Total Process Cost vs. Capacity Utilization

- CPU
- Heap
- GC

Sweet spot
Allocation

Cost of Capacity
Cost of Queue
Total Cost

Percent Capacity Utilization

0 10 20 30 40 50 60 70 80 90 100

#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)

Average usage ≈ 2200MB

-Xms4096m -Xmx4096m
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
  - \texttt{-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

#JenkinsWorld
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
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