Deployment and Usage of MCollective in Production.

S. Dietrich, J. Engels, S. Sternberger
HEPiX Spring 2015 Workshop
Oxford University (UK), 2015-03-26

HELMHOLTZ ASSOCIATION

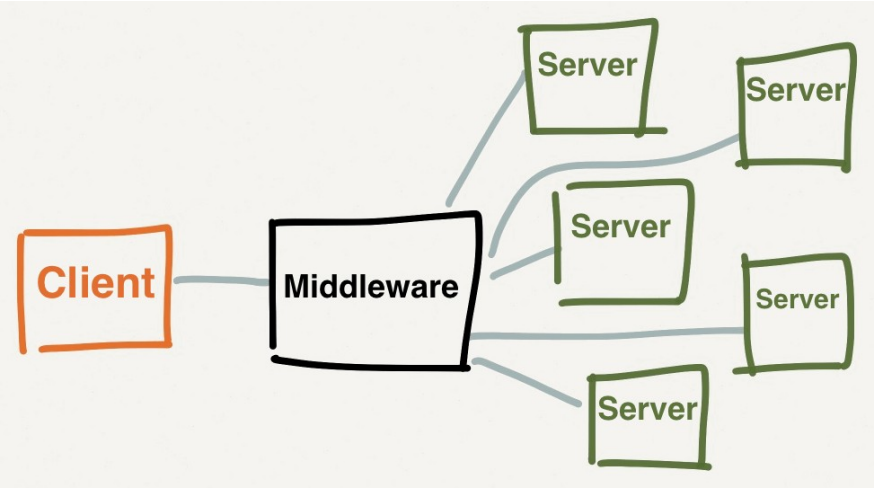
DESY

Outline

- What is MCollective?
- Use cases
- Current Infrastructure
- Middleware Setup
- Security and Performance Problems
- Current Status and Open Items

What is MCollective?

- "*The Marionette Collective, also known as MCollective, is a framework for building server orchestration or parallel job execution systems.*"
  Excerpt from Puppet Labs MCollective documentation
- Based on a messaging system with a middleware
- Publish/subscribe message pattern
- Interact with nodes through MCollective
  - Orchestrate your servers in parallel
  - Start/Stop services, execute jobs or simple queries for monitoring
- Good integration with Puppet
  - Filter nodes by Puppet classes or facts
- Framework
  - Not everything is covered by default



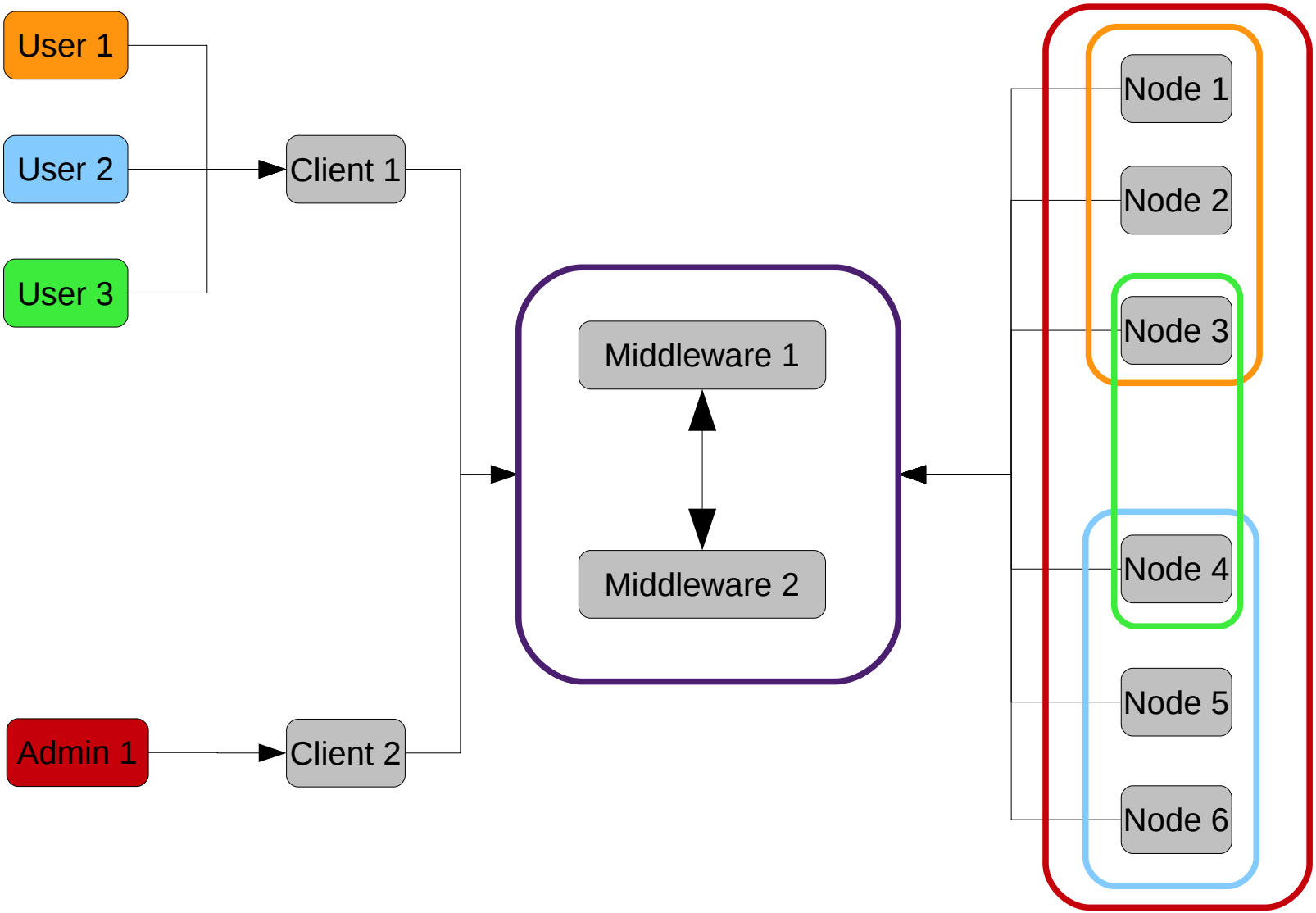
Use cases

- Multiple groups at DESY have interest in MCollective
  - Not limited to IT, but also MCS (control system) and FS-EC (experiment control)
- Orchestrating Puppet agent runs
  - Granular control of Puppet agents
- Query the infrastructure
  - Let`s identify all nodes, which are vulnerable to GHOST...
- Package updates
  - …and patch them against GHOST!
  - We do not use Puppet for software updates
- Other system maintenance tasks
  - The standard pssh use case

Infrastructure Overview

- ~ 2000 nodes managed by Puppet
  - Multiple distributions
  - Scientific Linux 6, EL 7, Ubuntu 12.04, Ubuntu 14.04 and Debian 7
- Puppet 3.7.4
- PuppetDB 2.2.2
- Foreman 1.5.3
- MCollective 2.8.0

Logical Overview



Middleware Setup

- ActiveMQ 5.9.1
- 2 server (2-way 12 core, 64GB RAM)
- High availability using “ring topology”
- Setup using Forge modules from Puppet Labs
  - puppetlabs-activemq (v0.2.0)
  - puppetlabs-java (v1.0.1)
  - puppetlabs-java_ks (v1.2.3)
  - puppetlabs-mcollective (v1.1.6)
    - Local modifications were necessary

Memory Settings

- ActiveMQ memory settings require some guessing
- "*As your deployment gets bigger, you may need to increase the <memaryUsage> and <tempUsage> elements in the <systemUsage> element. Unfortunately, we lack a lot of solid data for what to actually set these to. Most users leave the defaults for memory and temp until they have problems, then double the defaults and see whether their problems go away. This isn't perfectly effecient, but anecdotally it appears to work.*"
  - Excerpt from MCollective Puppet Labs documentation
- The following settings work for us
  - <memoryUsage limit="256 mb"/>
  - <storeUsage limit="1 gb"/>
  - <tempUsage limit="1 gb"/>

Memory Settings

- JAVA Memory settings (activemq-wrapper.conf)
- Initial Java Heap Size (in MB)
  - wrapper.java.initmemory=4096
- Maximum Java Heap Size (in MB)
  - wrapper.java.maxmemory=32768
- Java Additional Parameters
  - wrapper.java.additional.4=-Dorg.apache.activemq.UseDedicatedTaskRunner=false
  - Otherwise ActiveMQ will run out of connections/memory

Security

- MCollective supports multiple security plugins
- We use SSH key security plugin
  - https://github.com/puppetlabs/mcollective-sshkey-security
  - SSH keys already deployed with Puppet
- Connection to middleware secured with SSL
  - CA verified TLS
  - No certificates issued per user
  - Shared certificate for groups of MCollective clients
  - Subject to change...
- Subcollectives
  - Allow further division for security and traffic reasons
  - Not yet in production...

Performance Problems

- Major slowdown (10x) after setting up SSH key plugin
- Most nodes did not respond anymore
  - Ran into the default timeout of 2s
  - Increasing timeout to 60s helped
  - Something must be wrong...
- Severe bug in SSH key security plugin
  - Complete ssh_known_hosts parsed for each SSH key check
  - Currently ~5300 hosts in our known hosts file...
  - Already fixed by other contributor on GitHub
- Changed handling of facts
  - Now using YAML facts instead of always calling facter
- Persistence turned off

Current Status

- Performance and stability is now good enough
- Running MCollective in production
- Following use cases are currently covered
  - Steering Puppet agent runs
  - Querying the infrastructure
  - Small pssh tasks (e.g. package updates, services management)

```
% mco package -F lsbmajdistrelease=14.04 status openafs-modules-dkms
 - [ ============================================================>  ] 142 / 145

             exfldb01n0: openafs-modules-dkms-purged.
                  fai05: openafs-modules-dkms-1.6.7-1ubuntu1.
         it-cups-public: openafs-modules-dkms-1.6.7-1ubuntu1.
                   idp1: openafs-modules-dkms-1.6.7-1ubuntu1.
                   <Additional Nodes>
Summary of Ensure:

   1.6.7-1ubuntu1 = 112
           purged = 30

Finished processing 142 / 145 hosts in 182002.52 ms

No response from:
    atlas28
```

Open Items

- Improve Monitoring
  - JMX monitoring with Jolokia for ActiveMQ
  - Load-balancing between brokers – restart MCollective on nodes?
  - Stale puppet agents (e.g. by unresponsive NFS mounts)
- Configuration management
  - Do we really need the puppetlabs-mcollective module?
- Subcollectives
  - Currently in testing phase...
- Execute puppet resources with Mcollective?
- STOMP + NIO for better performance?
- Security
  - Alternative to current security plugin ?
  - More granular settings? Read-only for some users?

Questions?