LHCb's Puppet 3.5 to Puppet 4.9 migration

Three weeks long hackathon

H. Mohamed[1], **F. Sborzacchi**[2], **T. Colombo**[1], **L. Brarda**[1], **N. Neufeld**[1]
R. Schwemmer[1], **M. Daoudi**[1]

1. CERN, Geneva, Switzerland
2. INFN, Rome, Italy

hristo.mohamed@cern.ch

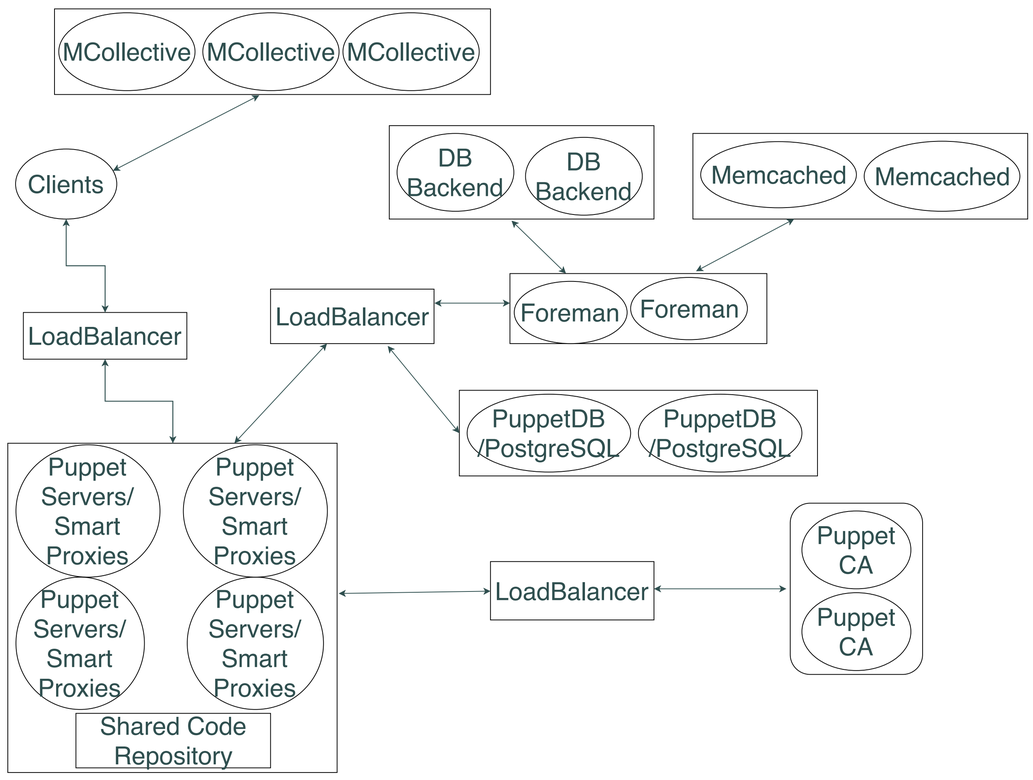
Objectives

- Migrate our entire code base from Puppet 3.5 to Puppet 4.9
- Put high availability on every component of our Puppet infrastructure
- Scale our Puppet infrastructure to be able to initiate a site-wide run in predictable time
- Scale out MCollective installation
- Do all of the above in the time frame of a month!

Introduction

Up until September 2017 LHCb Online was running on Puppet 3.5 a Master/Client non-redundant architecture. As a result, we had problem with outages, both planned and unplanned, as well as with scalability issues (unable to predict amount of time needed to complete site-wide run). What's more our code base was very large, aging and starting to lack compatibility with newer modules. Further more, Puppet 5.0 was stable and around the corner!

New Infrastructure[1]



Updating the Code base

For the redo of our code base, we used the following principles

- **Always** use modules from the forge(if available), never write your own
- (All) Specific data for our infrastructure goes to Hiera
- Write **every** module like an API
- Reuse profiles whenever possible
- Start using CI/CD pipeline

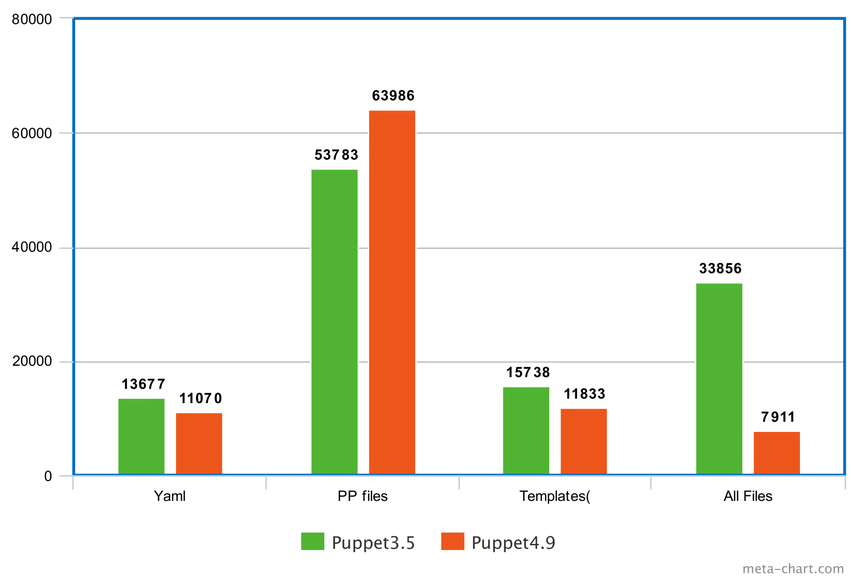


Figure 1: Code reduction in lines of code

Performance tunning and other considerations

Below are some performance tuning we used in our infrastructure

1. Test for yourself if deadline/cfq/noop performs better in your environment
2. Enable G1GC garbage collector with 4GB+ Heaps
3. Monitor PuppetDB performance drop and regularly initiate a vacuum[2]
4. For Foreman running behind Passenger ensure maximum number of instances from startup (process spawning is expensive)[3]
5. If using Passenger with Apache use MPM and a large number of workers (due to blocking I/O)[3]

Performance tunning of Puppet Masters - JRuby

1. Puppet Server uses JRuby instances. Default value are between 1 and 4 (num-cpus - 1)[4] . We see that num-cpus is better (CFQ scheduler)

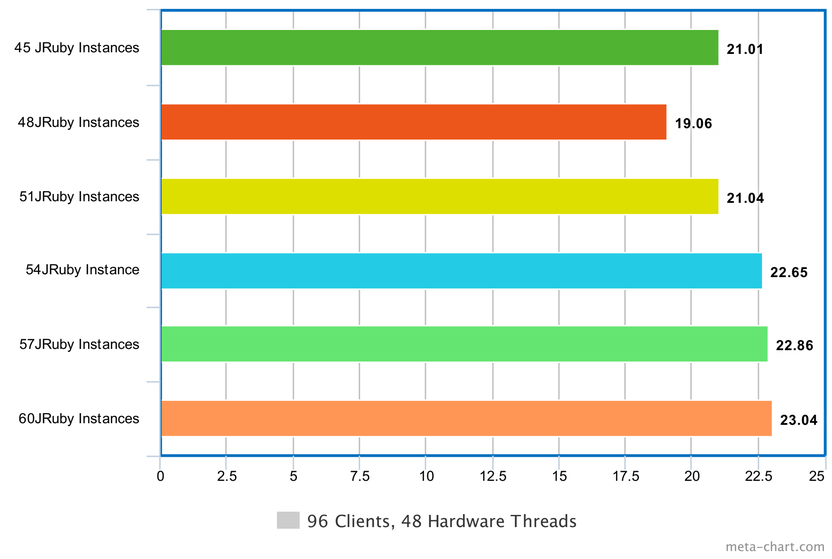


Figure 2: Performance split over 3 Puppet Masters, 16 hardware threads each

2. Minimum 512 MB of memory per JRuby with light catalogs / small number of environments
3. Ensure your system has adequate entropy

Results

1. We now have over 2500 hosts running our highly available Puppet
2. We can tolerate failure in every component
3. On average our infrastructure needs 17-20 seconds catalog compilation time
4. Full run of Puppet on our entire infrastructure takes less than 30 minutes!

Conclusion

Puppet's initial release was in 2005. Since then it has come a long way to become a complete configuration management solution. Nowadays for a successful Puppet Open Source deployment, multiple components are needed as well as following software engineering principles.

References

[1] Christopher Pisano. Journey to high availability. 2016.

[2] Robert Treat Greg Smith and Christopher Browne. Tuning your postgresql server. *PostgreSQL wiki*, 2018.

[3] Phussion Passanger. Phussion passanger documentation. 2018.

[4] PuppetLabs. Puppetlabs documentation. 2018.

[5] Júlio Brettas. Poster template - escola de modelos de regresso. 2018.

Additional materials used

Additional information and resources used:

- Pro Puppet
  Turnbull, James, McCune, Jeffrey
- http://www.brendangregg.com/linuxperf.html
  Brendan Gregg