Configuration Management at the RACF

WILLIAM STRECKER-KELLOGG
JASON A. SMITH
BROOKHAVEN NATIONAL LABORATORY

Current PuppetMaster

- 1 Dell PowerEdge R610, dual socket 6 core Intel Xeon X5660 @ 2.80GHz CPU, 96GB DDR3 1333 MHz RAM and 6 300GB 15k SAS6 hard drives in a hardware raid 10 (PERC H700)
- Runs 64bit RHEL6.6 hosting our central Puppet Master service using Puppet Server 1.0.2, Puppet 3.7.4 & PuppetDB 2.2.2 with a local PostgreSQL 9.3.6 backend
- Also serves our Puppet catalog git repo over authenticated https with custom hooks and production approval system, GLPI inventory service with a local MySQL 5.1.73 backend
- Old Puppet-dashboard still running, but will disable soon
  - Already replaced with Foreman 1.7.2 running on another VM

Puppet Server

- New server stack to replace Apache/Passenger
- Written in Clojure (JVM)
- Uses Jetty for HTTP(s) and JRuby v1.9
- True multithreading in JRuby allows faster and more efficient parallel compilation
  - MRI Ruby has Global Interpreter Lock
    - Multiple runs are in different processes
  - Future (possibly) environment-specific threads
    - Would allow custom modules to differ per-environment
- Default in Puppet Enterprise 3.7

Puppet Server

- Pros
  - Easier to install & configure compared to Apache/Passenger
  - Faster than old server—twice as fast at compilation
  - Active development ongoing
- Cons
  - Very slow startup (30s) even on modern hardware
  - Full restart needed after revoking or removing old client certs
  - Little to no SSL error-logging at levels above debug
  - Needs a lot of memory—2Gb default fails under heavy load
    - Increased to 20Gb and it works fine with > 1Hz clients
  - No Dashboard for monitoring server performance metrics
    - Disappointing considering puppetdb has one—graphite monitoring for enterprise

Puppet Server Performance

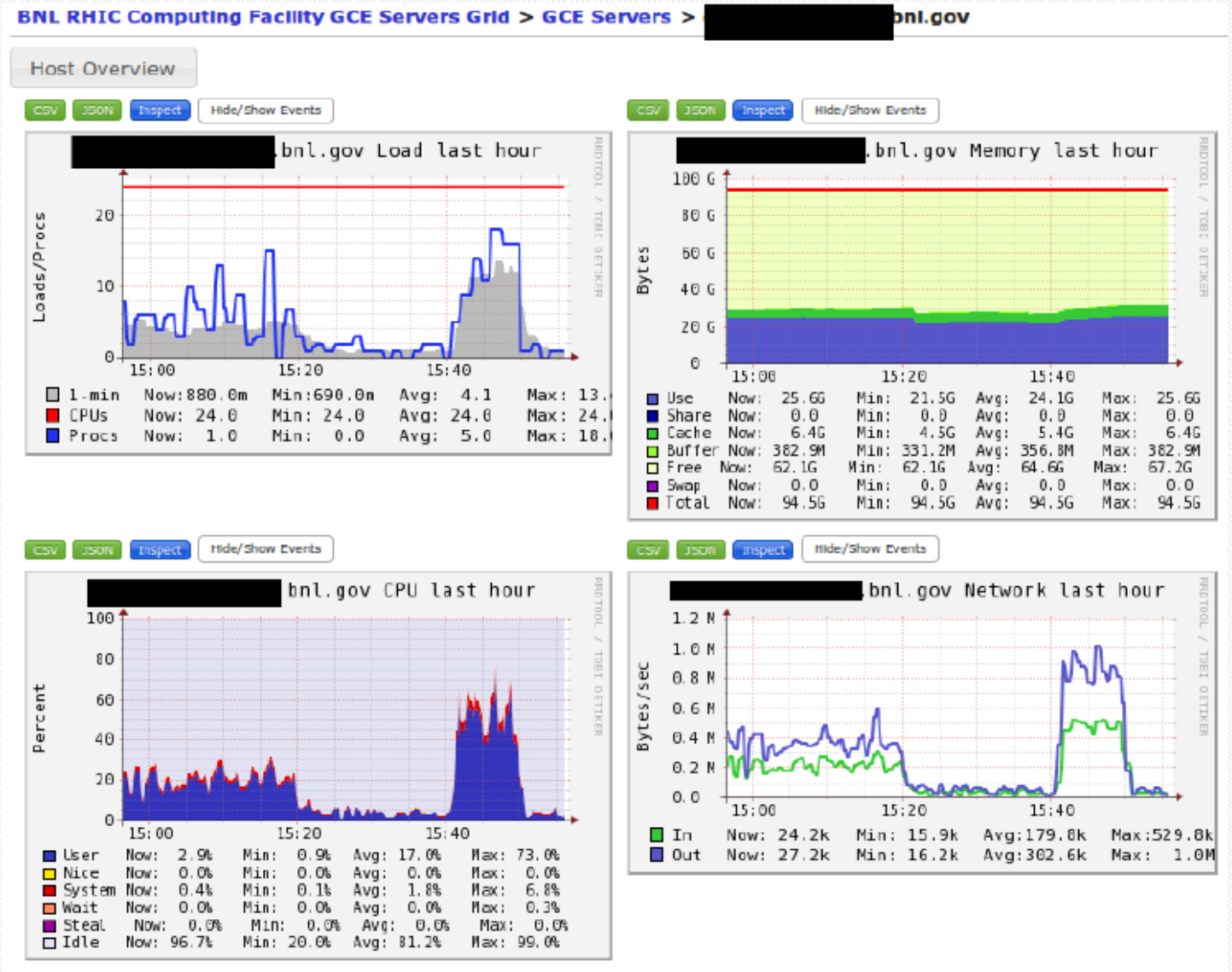
- Catalog compilation went from an average of 1.97 sec → 1.00 sec
  - Scraping catalog compilation from logs with some awk magic
- Ran tests of a mass puppet-run across our infrastructure, with a rate of 2Hz
  - Load was under 50% on our 2 year old server
    - Over half was from the old dashboard
  - Extrapolating this means we could handle 8Hz with linear scaling (>20,000 clients every hour!)

Load Tests

Previous Configuration

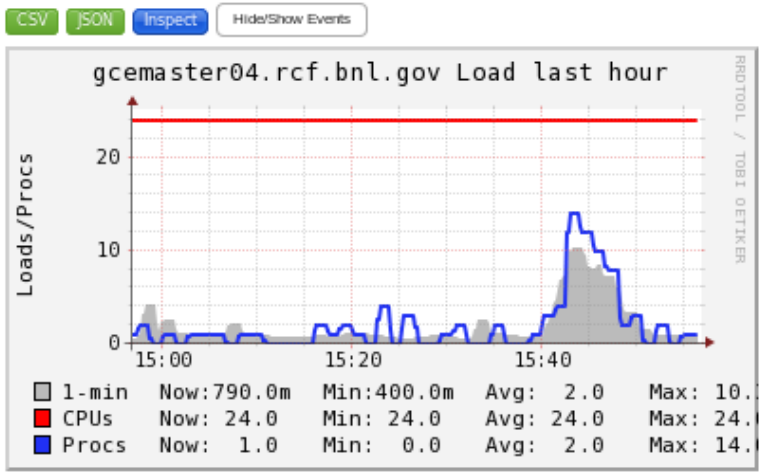
From James Pryor's slides from Lincoln

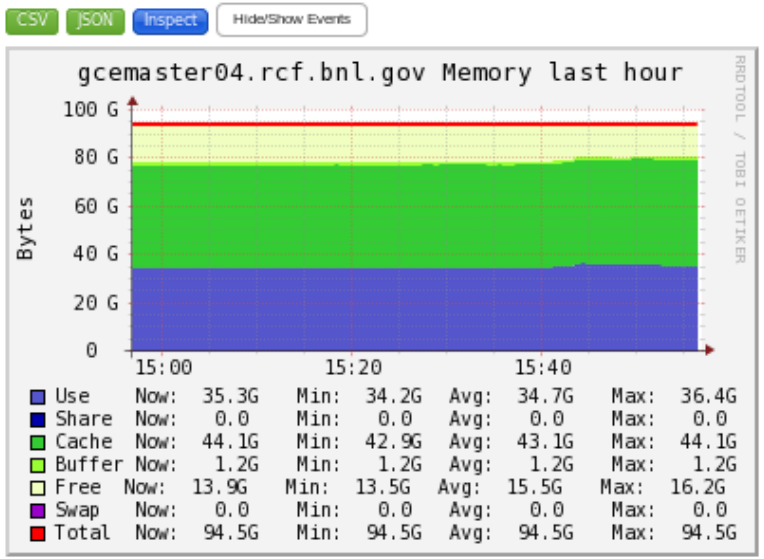
Load test of just over 1Hz

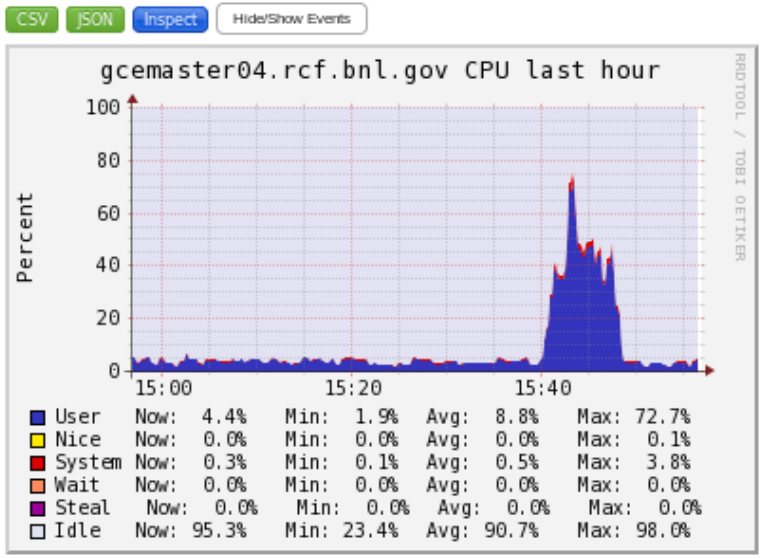


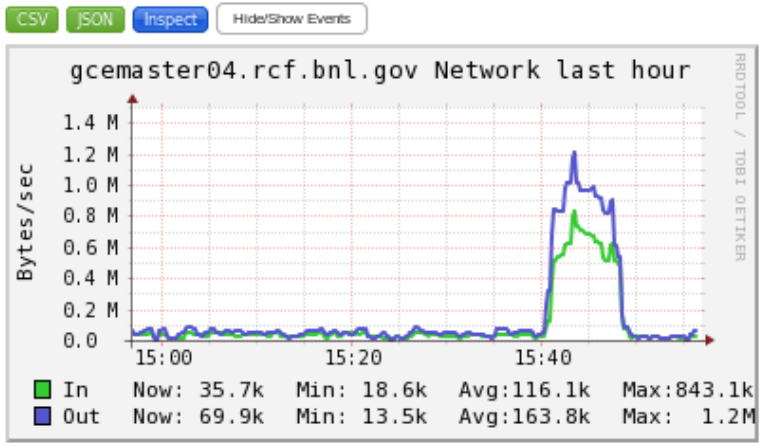
Load Tests

Ganglia During the 2Hz Test of new puppet server









Load Tests

Top output

Notice the puppet-d* user procs are for the outdated dashboard

- Generates at least half the load

```
top - 15:44:36 up 21 days,  3:44,  4 users,  load average: 8.94, 5.72, 2.85
Tasks: 557 total,  12 running, 545 sleeping,   0 stopped,   0 zombie
Cpu(s): 47.5%us,  2.4%sy,  0.0%ni, 50.0%id,  0.0%wa,  0.0%hi,  0.1%si,  0.0%st
Mem:  99051892k total, 84696516k used, 14355376k free,  1218532k buffers
Swap:  8388604k total,        0k used,  8388604k free, 45332908k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
12919 puppet    20   0 33.3g  21g  13m S 524.7 22.4 479:25.04 java
 5801 puppet-d  20   0  168m  72m 1948 R 98.2  0.1  73:48.69 ruby
 5784 puppet-d  20   0  168m  72m 1948 R 88.2  0.1  75:43.21 ruby
 5763 puppet-d  20   0  168m  72m 1948 S 83.6  0.1  76:28.03 ruby
 6019 puppet-d  20   0  168m  72m 1948 R 73.6  0.1  74:42.03 ruby
 7428 puppetdb  20   0 20.9g 2.9g 6576 S 67.3  3.0   3870:35 java
18810 postgres  20   0 24.4g 546m 538m S 44.4  0.6   0:03.78 postmaster
 5453 puppet-d  20   0  168m  72m 1964 R 30.5  0.1  75:58.38 ruby
17682 apache    20   0  346m  42m 5232 S 26.5  0.0   1:07.68 httpd
 5992 puppet-d  20   0  168m  72m 1948 R 16.3  0.1  73:56.12 ruby
 5832 puppet-d  20   0  168m  72m 1948 R 13.9  0.1  74:21.67 ruby
13883 postgres  20   0 24.7g 1.8g 1.5g R 13.6  1.9  18:40.53 postmaster
 5754 puppet-d  20   0  167m  71m 1948 R 12.9  0.1  71:29.10 ruby
11172 puppet    20   0 99616  22m 3568 R 12.3  0.0   0:03.65 ruby
 5284 mysql     20   0 5153m 1.8g  13m S  9.6  1.9 212:53.52 mysqld
30297 postgres  20   0 24.8g 1.9g 1.4g S  6.0  2.0  12:45.84 postmaster
18811 postgres  20   0 24.4g 540m 534m S  4.6  0.6   0:03.74 postmaster
19261 postgres  20   0 24.7g 1.5g 1.2g S  4.3  1.6   5:38.97 postmaster
13882 postgres  20   0 24.7g 1.8g 1.5g R  2.7  1.9  18:36.02 postmaster
19337 puppet    20   0 51512 7704 2116 R  2.7  0.0   0:00.08 puppet_ext_node
 3410 postgres  20   0 24.4g 563m 549m S  2.3  0.6   0:04.57 postmaster
17423 puppet-d  20   0  173m  73m 2316 S  2.3  0.1   2:00.85 ruby
20085 postgres  20   0 24.7g 1.9g 1.5g S  2.3  2.0  18:45.03 postmaster
20093 postgres  20   0 24.7g 1.9g 1.5g S  2.0  2.0  18:47.88 postmaster
 2568 postgres  20   0 24.6g 1.4g 1.1g S  0.7  1.4   4:17.99 postmaster
 7520 nscd      20   0  911m 1584 1040 S  0.7  0.0  26:43.87 nscd
19076 root      20   0 13304 1540  872 R  0.7  0.0   0:13.53 top
   11 root      RT   0     0    0    0 S  0.3  0.0   0:03.11 migration/2
   21 root      20   0     0    0    0 S  0.3  0.0   0:08.83 ksoftirqd/4
   57 root      20   0     0    0    0 S  0.3  0.0   0:09.99 ksoftirqd/13
```

Puppet Server Miscellanea

- Puppet server uses JRuby 1.9, so custom ruby modules may need updating if previously using OS's ruby 1.8 (RHEL 6)
- We see a 0.2% compilation failure rate due to DNS query errors
  - Org/jruby/ext/socket/RubyUDBSocket.java:160: bind: name or service not known
  - May be a bug in JVM (exception in socket.bind)
  - Still better than Passenger/Rack where we would get a 1% failure rate
- In production for 2 months with no problems

Puppet Testing

- Many types of testing exist (unit, acceptance, & integration)
- RSpec unit tests ensure a module's logic does what you think it should do
  - Can feel a bit redundant when writing for most puppet modules
- Serverspec uses RSpec for acceptance testing
  - Runs code on a host to see what changes are made
- Beaker is an acceptance testing framework that does VM provisioning

Jenkins CI

- Continuous Integration tool used to manage building & testing of software
  - Configurable job-scheduling and management system
  - Uses SCM commit hooks to compile and run test for many software projects
- We test pending production changes by running the full catalog on a pool of VMs managed by Jenkins
- Catches inter-module and system-wide issues that *Spec cannot

Jenkins Process

- Test pool of several VMs that represent a cross-section of our infrastructure
- Only successful tests allow a production merge
- Already managed production pushes with a git hook that traps changesets in a pending branch and forces a manual review
  - System has per-module ACLs for approvers
  - Mandated a required 5-min timeout for approving your own changes
    - Still can't prevent human error!

Jenkins Process

- Hooks and approval scripts modified to kick off a test agent run on the pending branch which must succeed on all VMs

- Push initiates 4 jobs in Jenkins
  1. Sync branch to Foreman as env (import:puppet_classes rake task).
  2. Assign VMs in Jenkins pool to test env (Foreman REST API)
  3. Run puppet-agent on each VM (with optional 2nd idempotent test)
  4. Reset VMs to pre-test state (RHEV REST & shell API)

Using Jenkins

- Final merge only allowed if Jenkins tests passed
- If multiple changesets are pending and one is merged into prod, it is merged into the other pending branches and re-tested
- Staff has to populate RHEV pool of VMs with a representative set of important services and hosts
- Future—tie in with monitoring to validate state of post-run pool of test hosts
  - E.g. Nagios hooks could test that the managed services themselves are still up and operating as expected

MCollective

- Server orchestration & parallel job execution framework
- Currently in testing
  - Using ActiveMQ broker
  - Installed and configured by puppet to leverage Puppet's certificates
- Plan to use same server as puppet-master and will integrate it into Foreman for push/kick runs
  - Will a single broker scale to several thousand hosts?
  - Parallelism is poor, can't control global concurrency of running jobs!?

Code Sharing

- Helped another Physics group set up their own puppet-master using ours as a template
- Collaborated with BNL's general IT dept and helped them set up a similar puppet-master for desktop management and other department's servers
- Using git-subtree to share a common set of modules
  - Can import/export a common subdirectory containing shared Puppet modules into a user's git Puppet reop
  - Uses standard git workflows of push updates & pull changes, test & merge, etc…

Future Parser

- Issues with bare strings
  - Those containing a dash are considered to be subtraction
  - Case-statement clauses need to be quoted
- Standard library needed update
- Strange, non-intuitive errors caused by:
  - $type as a class-parameter with undef as default value
  - Calling a parameterized class with undef as a value to override a default non-undef value
  - Overriding a variable with a default value explicitly set to undef
  - A module with a require parameter in the <||> operator overriding a non-undef default
  - Errors involving parameters being ‘nil’ in template evaluation
    - Parent-scope lookup in templates that used to work no longer does

Catalog Aging

- RACF started using puppet heavily in early 2011
- Puppet is rapidly evolving, was even faster back then
  - No Hiera, no Forge, far fewer tests, evolving “best practices”
- Had to develop many modules from scratch where versions later added to the forge became standard (“Puppet-supported”)
- Catalog is littered with cruft
  - Lots of if/else, case, etc…
  - Data, data everywhere in the manifests

Fixing Catalog Cruft

- Question of institutional priorities
  - Why fix it if it works?
  - Technical debt can creep up
- How to manage creeping cruft if puppet is still so rapidly evolving
  - At some point around 3 or 4 years ago they were discussing dropping parameterized classes
    - Still can't override a parameterized class included resource-style in a subclass
  - If we move to doing everything 'X' way, what if X changes next year?

External Node Classifier Cruft

- Have system of hierarchical ENCs
  - Each group doesn't want other groups to be able to change their node config
    - Early-on, made heavy use of ENC rather than roles & profiles
  - Like a hiera nodename regex matching level
  - At the time dashboard and GLPI didn't have ACLs
- Would like to migrate to Foreman for everything
  - Major undertaking—not at all trivial to set up ACLs on a per-node basis rather than a per-code-subtree basis

Planned Developments

- Start using Jenkins CI in production
  - Need to configure test pool of VMs
- Put MCollective in production
- Setup Icinga monitoring and configure automated monitoring with exported resources
- Migrate Cobbler KS templates to Foreman to use it for provisioning
- Refactor our puppet modules to follow more standard practices, like using Roles & Profiles, and Hiera
- Testing “Future Parser” in 3.x, will be default in 4.0

The End

THANK YOU!

QUESTIONS? COMMENTS?

WILLSK@BNL.GOV, SMITHJ4@BNL.GOV