redhat.

OpenStack Telemetry and the 10,000 Instances

To infinity and beyond

Julien Danjou
Alex Krzos
9 May 2017

redhat.

OpenStack Telemetry and the ~~10,000~~ 5000 Instances

At least they tried!

Julien Danjou
Alex Krzos
9 May 2017

Introductions

Julien Danjou

Principal Software Engineer @ Red Hat

jdanjou@redhat.com

IRC: jd_

Alex Krzos

Senior Performance Engineer @ Red Hat

akrzos@redhat.com

IRC: akrzos

Agenda

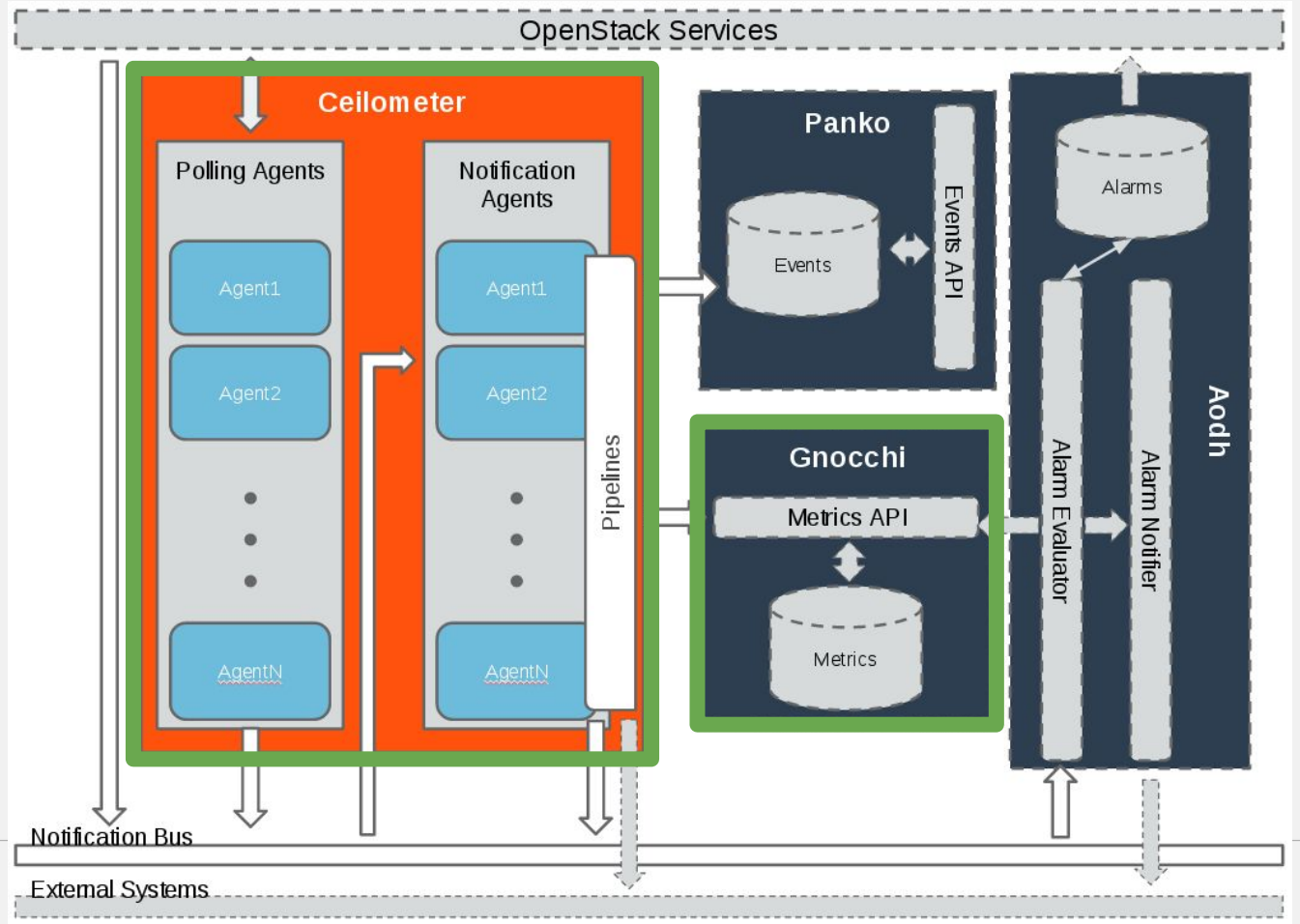
- What is OpenStack Telemetry?
- Telemetry Architecture
- Scale & Performance Testing
  - Workloads
  - Hardware
  - Results
  - Tuning
- Development influence
- Conclusion
- Q&A

OpenStack Telemetry

- **Ceilometer**
  - Polling data and transforming to samples
  - Store data in Gnocchi
- **Aodh**
  - Alarm evaluation engine
  - Evaluate threshold from Gnocchi
- **Panko**
  - CRUD OpenStack events
  - Fed by Ceilometer

- ***Gnocchi***
  - *Store metrics and resources index*
  - *Left Telemetry in March 2017*

Telemetry Architecture



What was actually tested for performance

Scale & Performance Testing

Goal: Scale to 10,000 instances and if not, find bottleneck(s) preventing scaling of OpenStack Telemetry's Gnocchi with Ceph Storage driver. Characterize overall performance of Gnocchi with Ceph Storage.

Workloads

Boot Persisting Instances

- Tiny Instances 500/1000 at a time, then quiesce for designated period (30m or 1hr)

Boot Persisting Instances with Network

- Tiny Instances with a NIC

Measure Gnocchi API Responsiveness

- Metric Create/Delete
- Resource Create/Delete
- Get Measures

Hardware

3 Controllers

- 2 x E5-2683 v3 - 28 Cores / 56 Threads
- 128GiB Memory
- 2 x 1TB 7.2K SATA in Raid 1

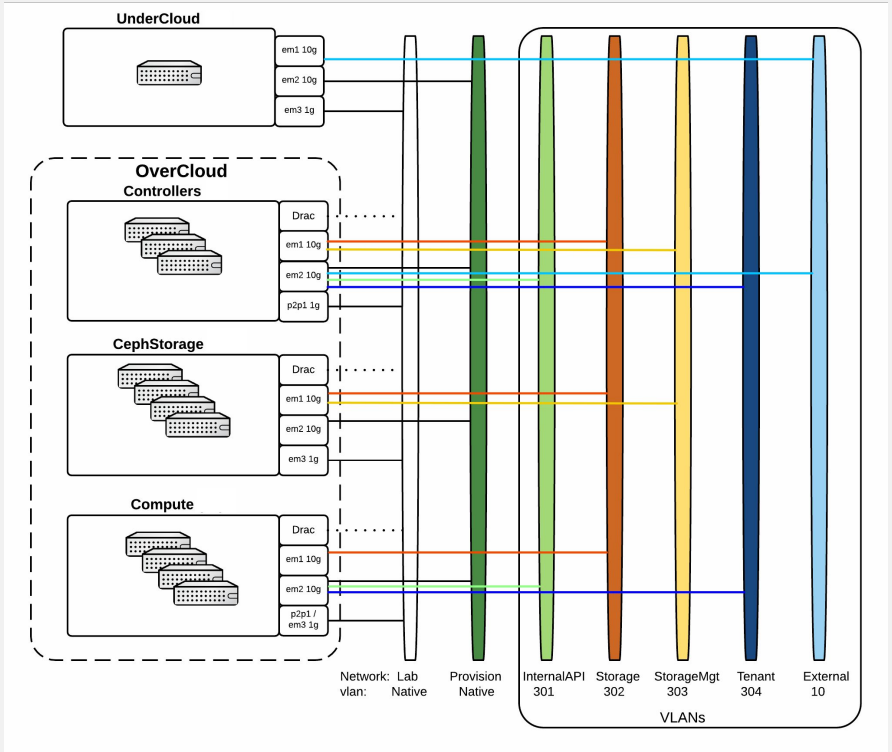
12 Ceph Storage Nodes

- 2 x E5-2650 v3 - 20 Cores / 40 Threads
- 128GiB Memory
- 18 x 500GB 7.2K SAS (2 - Raid 1 - OS, 16 OSDs), 1 NVMe Journal

31 Compute Nodes

- 2 x E5-2620 v2 - 12 Cores / 24 Threads
- 128GiB / 64 GiB Memory
- 2 x 1TB 7.2K SATA in Raid 1

Network Topology



10,000 Instance Test

Workload

- 500 instances every 1hr

Gnocchi

- metricd workers per Controller = 128
- metric_processing_delay = 15

Ceilometer

- Pipeline publish to Gnocchi
- Ceilometer-Collector disabled
- Rabbit_qos_prefetch_count = 512
- Low archival-policy
- Polling Interval 1200s

Ceph

- replica=1 for metrics pool

MariaDB

- max_connections=8192

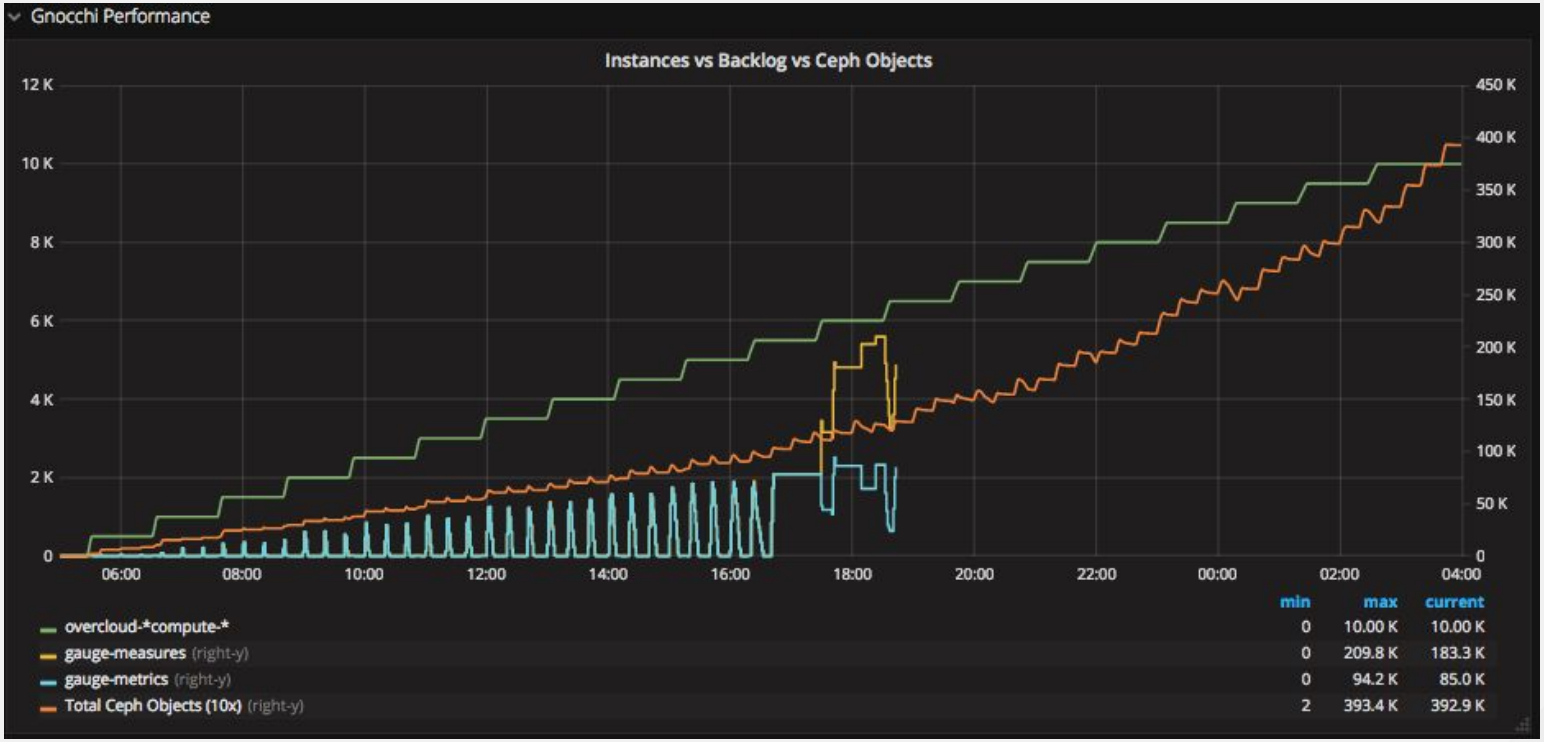
Nova

- NumInstances Filter
- Max_instances_per_host = 350
- Ram_weight_multiplier = 0

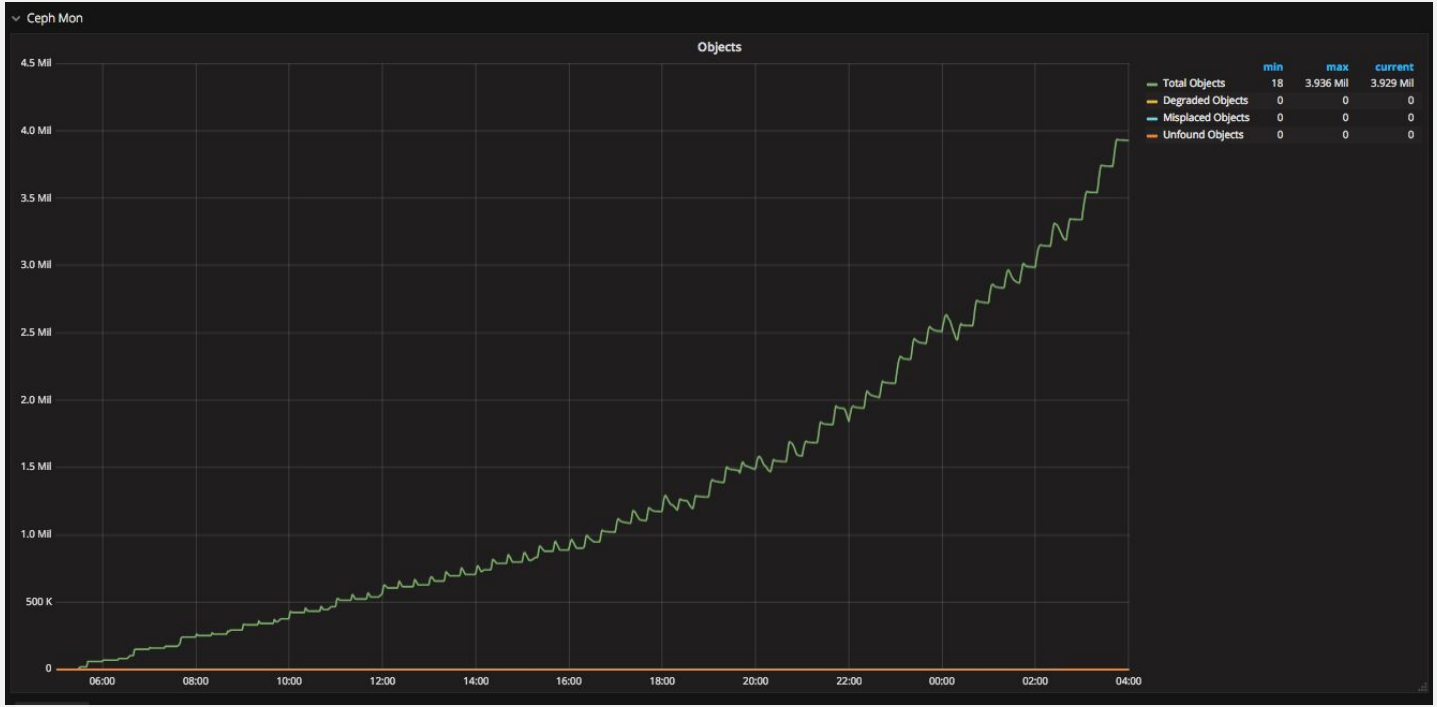
Patches

- max_parallel_requests in Ceilometer
- Batch Ceph omap object update in Gnocchi API

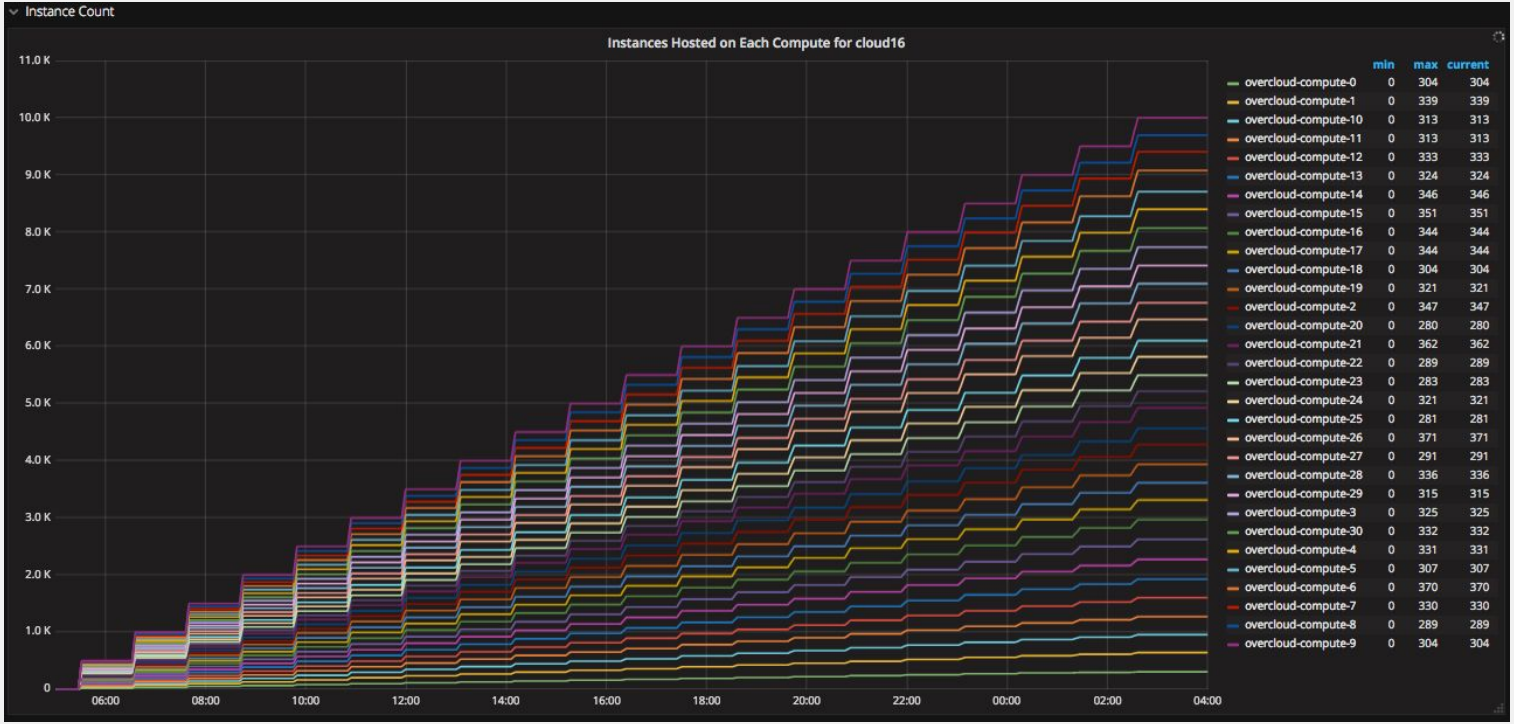
Results - 10k Test Gnocchi Performance



Results - 10k Test Ceph Objects



Results - 10k Test Instance Distribution



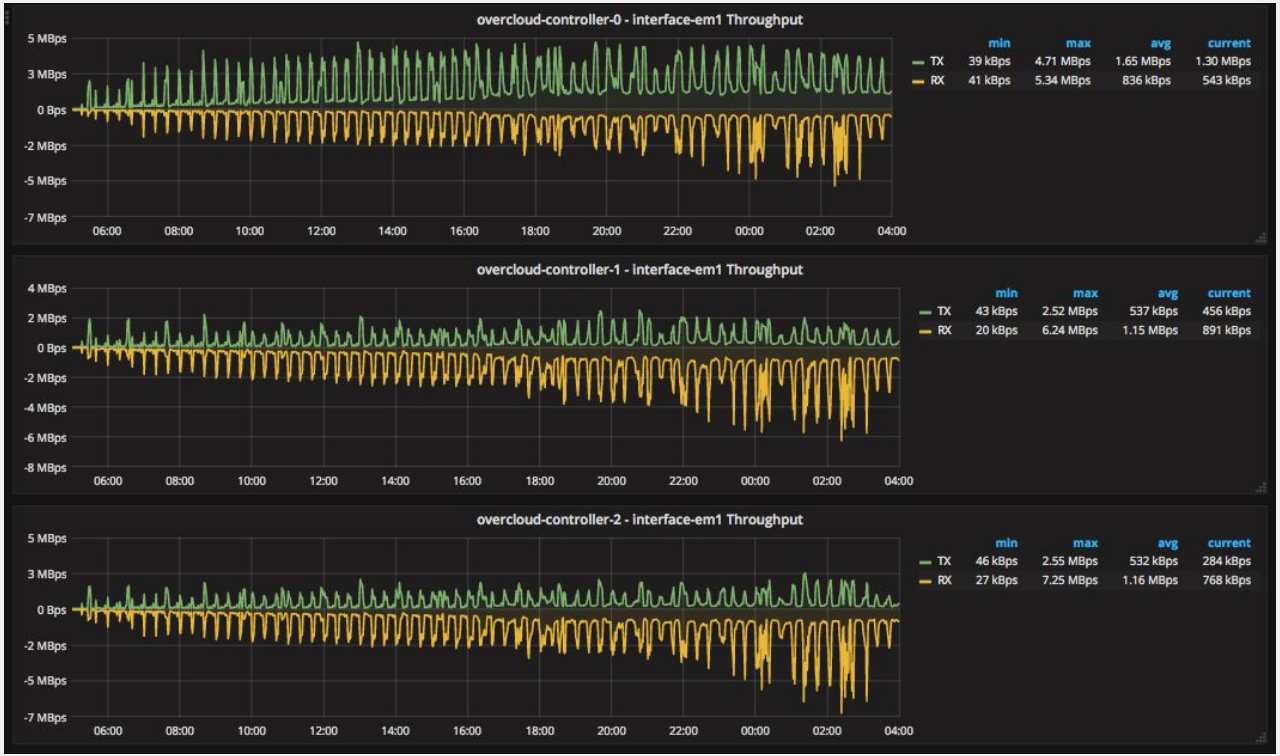
Results - 10k Test CPU on Controllers

Results - 10k Test Memory on All Hosts

Results - 10k Test Disks on Controllers

Results - 10k Test Disks on CephStorage

Results - 10k Test Network Controllers Em1



Results - 10k Test Network Controllers Em2

API Responsiveness Test

Workload

- 500 instances with Network every 30 minutes

Gnocchi

- metricd workers per Controller = 128
- metric_processing_delay = 30

Ceilometer

- Pipeline publish to Gnocchi
- Ceilometer-Collector disabled
- Rabbit_qos_prefetch_count = 512
- Low archival-policy
- Polling Interval 600s

Ceph

- replica=3 for metrics pool (default)

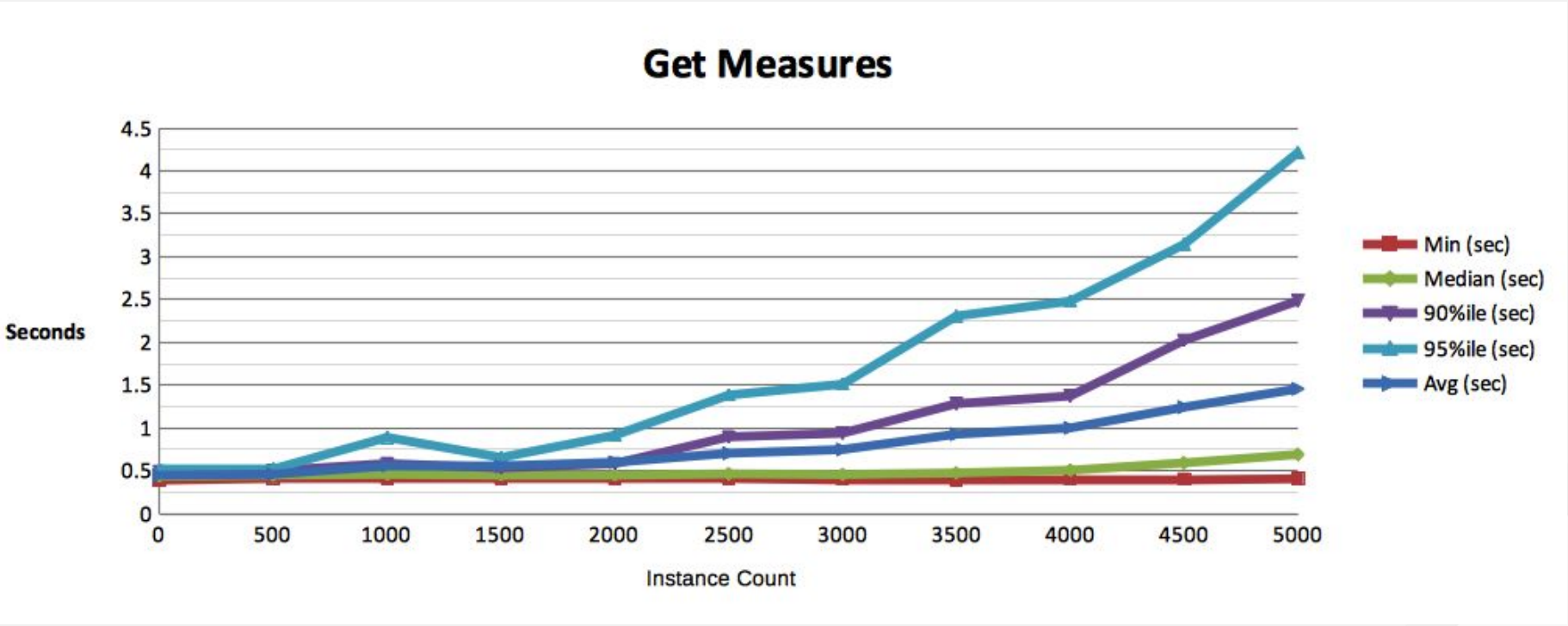
MariaDB

- max_connections=8192

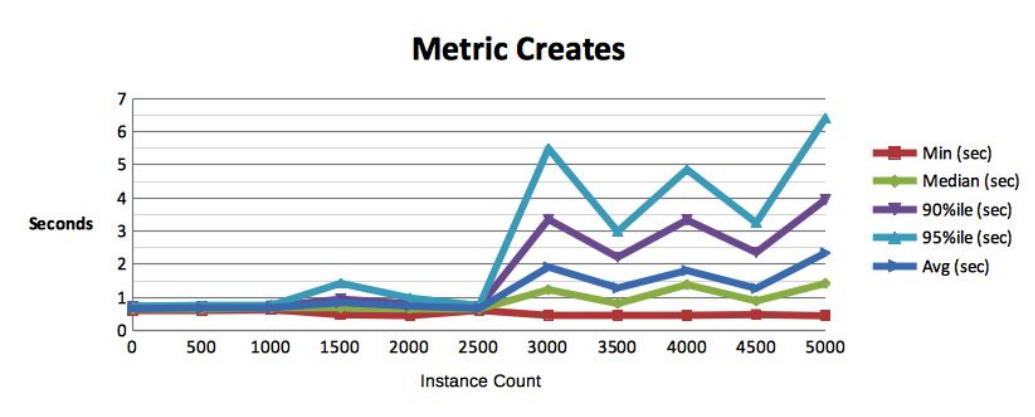
Nova

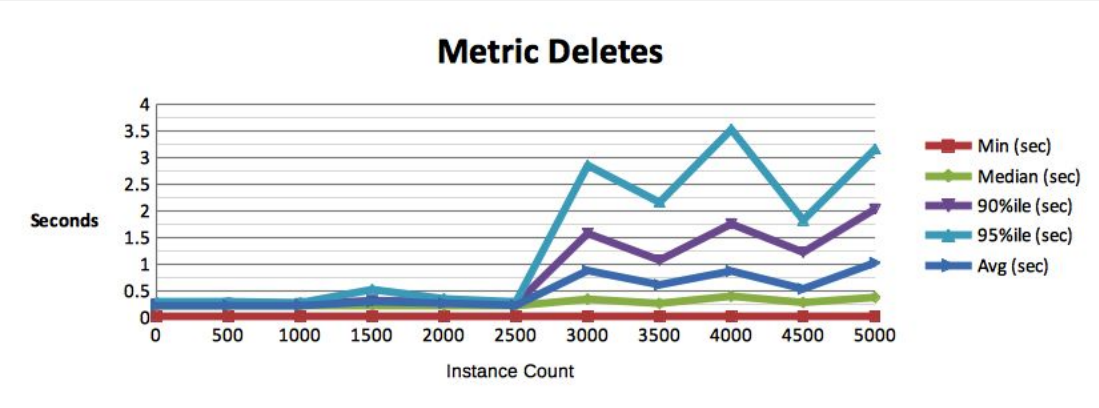
- NumInstances Filter
- Max_instances_per_host = 350
- Ram_weight_multiplier = 0

Results - API Get Measures



Results - API Create/Delete Metrics



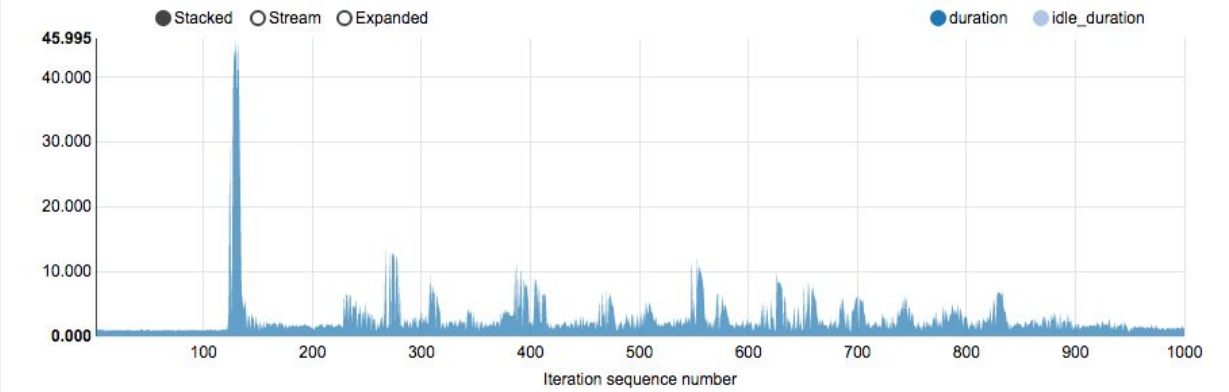


Results - API Create/Delete Metrics - Cont

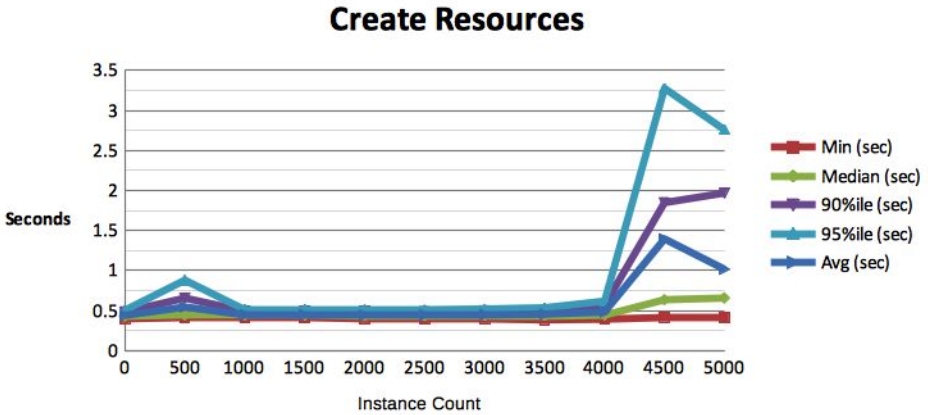
“Bad Timing” - Collision with Polling Interval

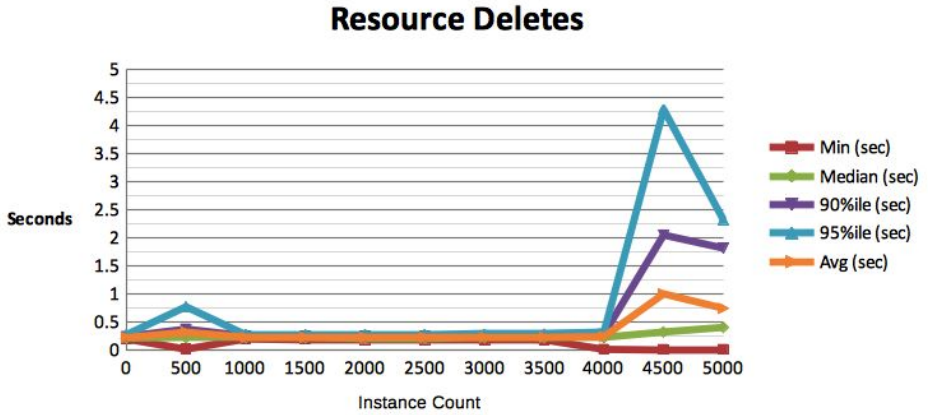
Total durations

Action	Min (sec)	Median (sec)	90%ile (sec)	95%ile (sec)	Max (sec)	Avg (sec)	Success	Count
gnocchi.metric_create	0.466	1.235	3.354	5.496	43.262	1.917	100.0%	1000
gnocchi.metric_delete	0.019	0.342	1.575	2.852	39.817	0.88	100.0%	1000
total	**0.68**	**1.811**	**5.392**	**6.911**	**45.995**	**2.808**	**100.0%**	**1000**



Results - API Create/Delete Resources





Tuning - Gnocchi

Gnocchi

- metricd workers - More workers = Capacity but costs memory
- metricd metric_processing_delay - Reduced Delay = Greater Capacity at CPU/IO Expense

MariaDB

- max_connections - indexer is in Mariadb

Haproxy

- check maxconn default in haproxy

Tuning - Ceilometer

Ceilometer

- Publish direct to gnocchi - “notifier://” -> “gnocchi://” in pipeline.yaml
- Disable Ceilometer-collector
- Set rabbit_qos_prefetch_count
- Default archive-policy - less definitions are less IO intensive
- **Understand what your desired goal is with Telemetry Data**

name	definition	aggregation_methods
bool	- points: 31536000, granularity: 0:00:01, timespan: 365 days, 0:00:00	last
high	- points: 3600, granularity: 0:00:01, timespan: 1:00:00	std, count, min, max, sum, mean
	- points: 10080, granularity: 0:01:00, timespan: 7 days, 0:00:00	
	- points: 8760, granularity: 1:00:00, timespan: 365 days, 0:00:00	
low	- points: 8640, granularity: 0:05:00, timespan: 30 days, 0:00:00	std, count, min, max, sum, mean
medium	- points: 10080, granularity: 0:01:00, timespan: 7 days, 0:00:00	std, count, min, max, sum, mean
	- points: 8760, granularity: 1:00:00, timespan: 365 days, 0:00:00	

Tuning - Httpd

HTTPD - Prefork MPM

- MaxRequestWorkers (MaxClients) / ServerLimit - Maximum Apache slots handling requests
- StartServers - Child Server Processes on Startup
- MinSpareServers / MaxSpareServers - Min/Max Idle Child Processes
- MaxConnectionsPerChild (MaxRequestsPerChild)
- Gnocchi WSGI API - Processes/Threads
    - More Processes = More Capacity for measures/metrics or to process requests for Gnocchi Data
- Careful planning values with multiple services hosted in same HTTPD instance

Issues - Gnocchi/Ceilometer

Gnocchi

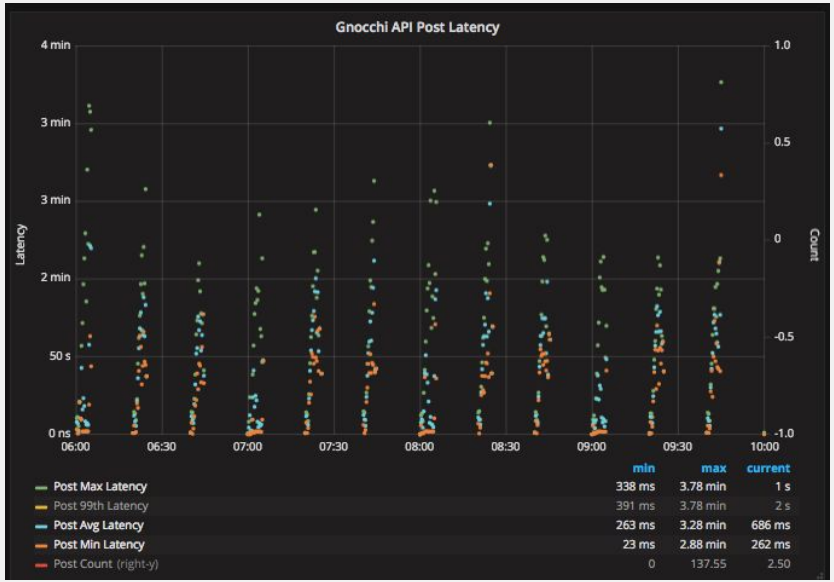
- Single Ceph Object for Backlog
- Many Small Ceph Objects
- Gnocchi API Slow posting new measures
- HTTPD prefork thrashing
- Gnocchi can lose block to work on
- Connection pool full
- Backlog status slow to retrieve

Ceilometer

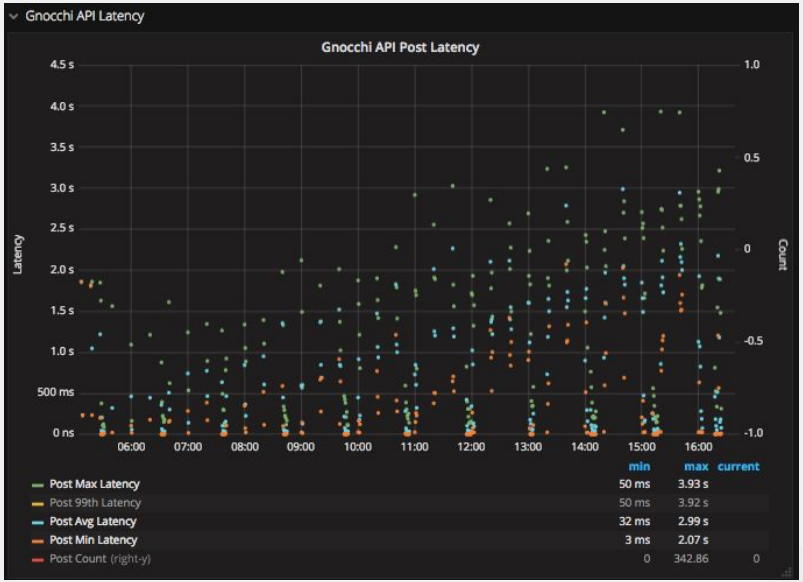
- Rabbitmq prefetching too many messages

Issues - Gnocchi Slow API POST

Threaded

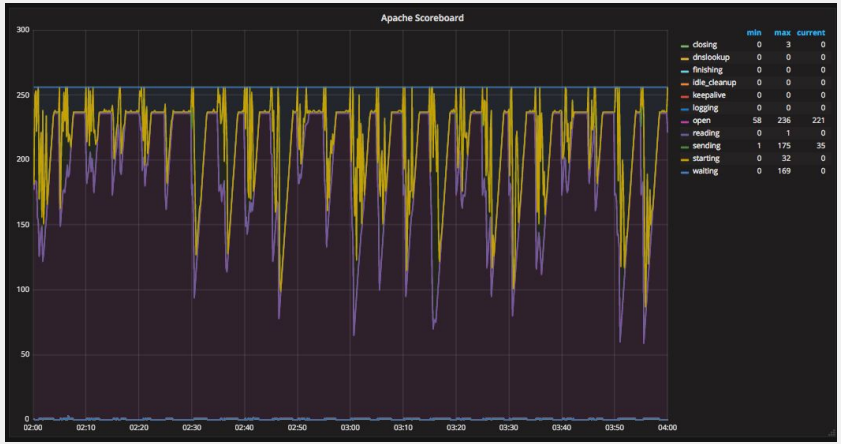


Batch

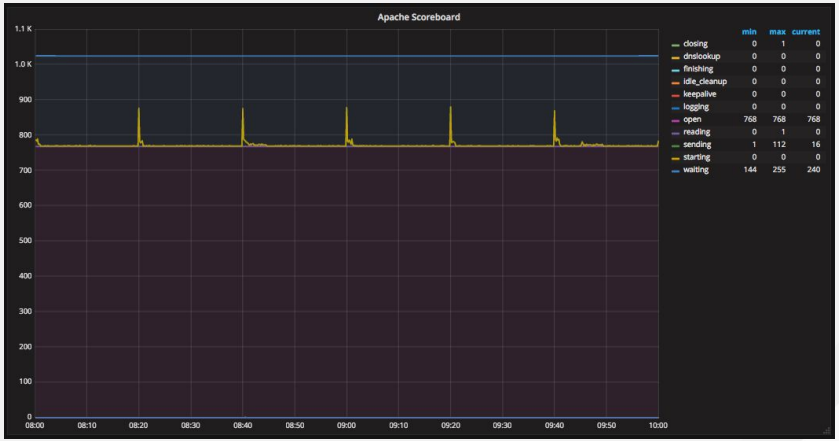


Issues - Gnocchi API (HTTPD) Thrashing

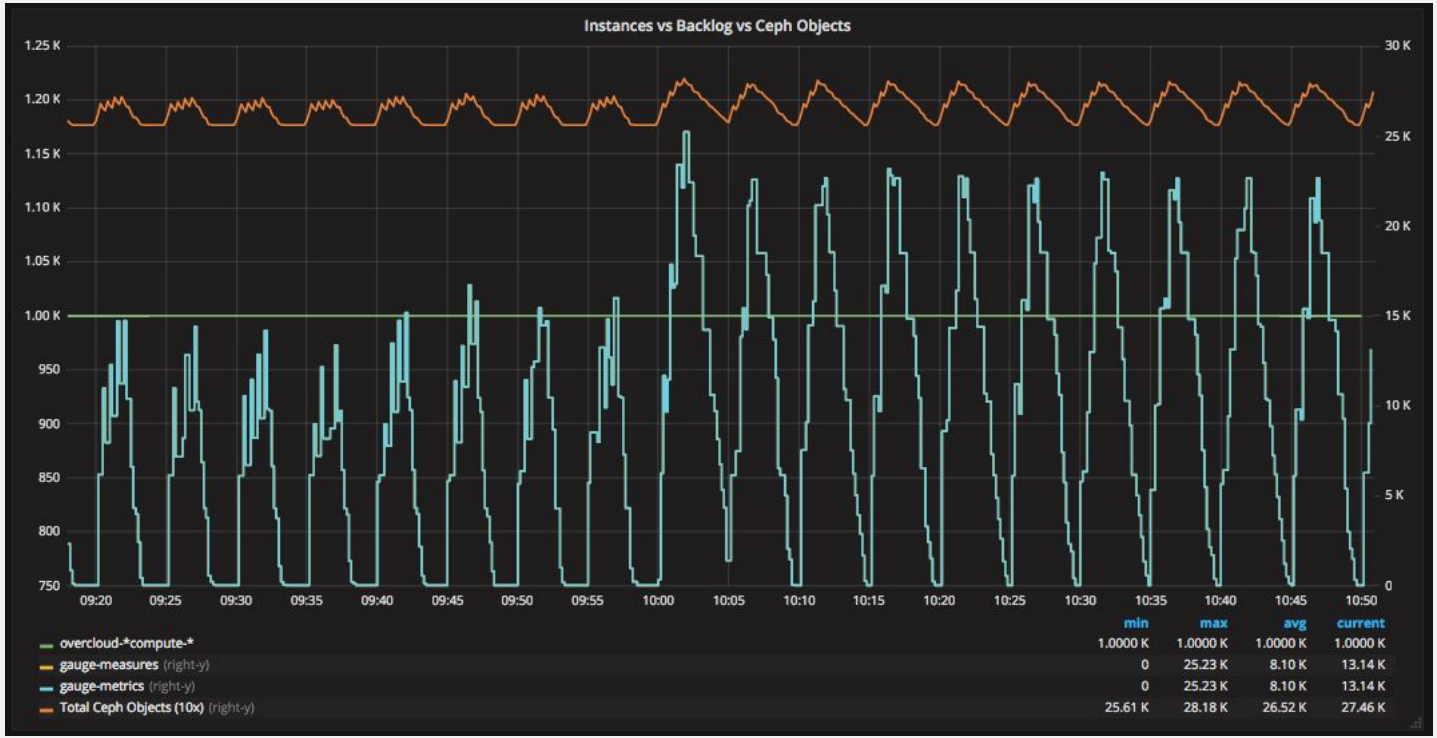
Threaded API
MinSpareServers 8
MaxClients/ServerLimit 256



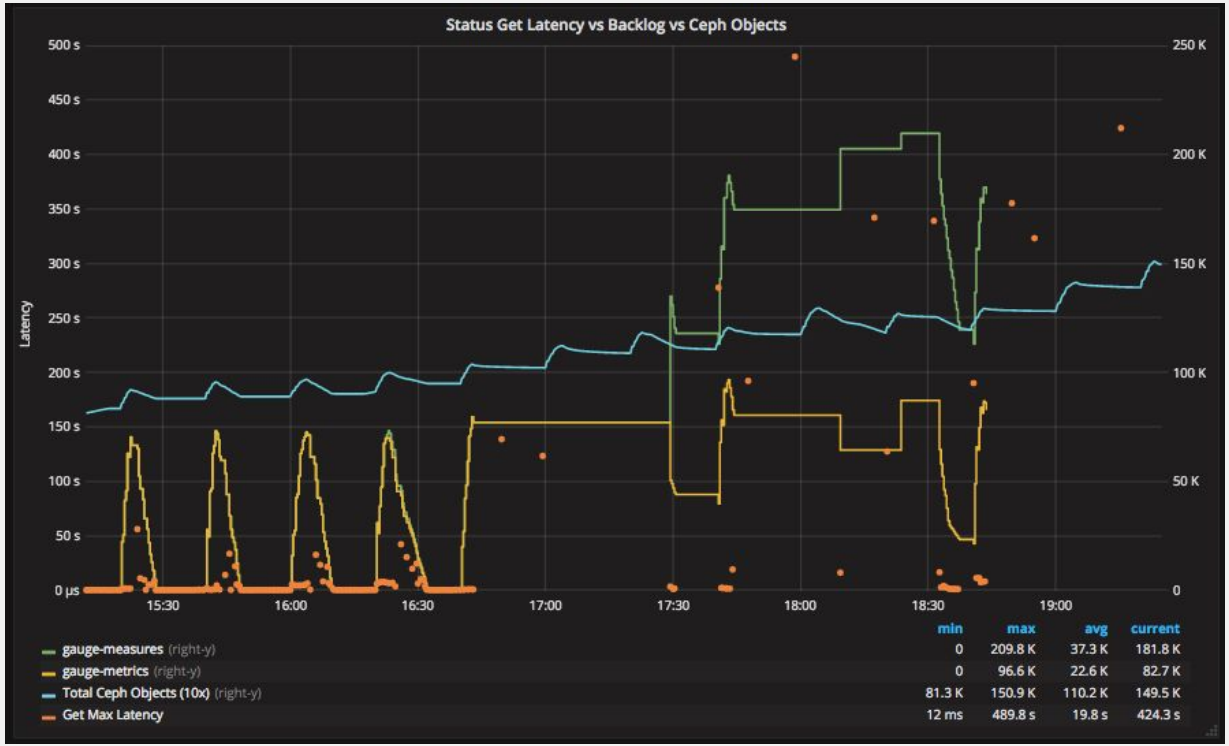
Batch API
MinSpareServers 256
MaxClients/ServerLimit 1024



Issues - Gnocchi Lost Block to work on

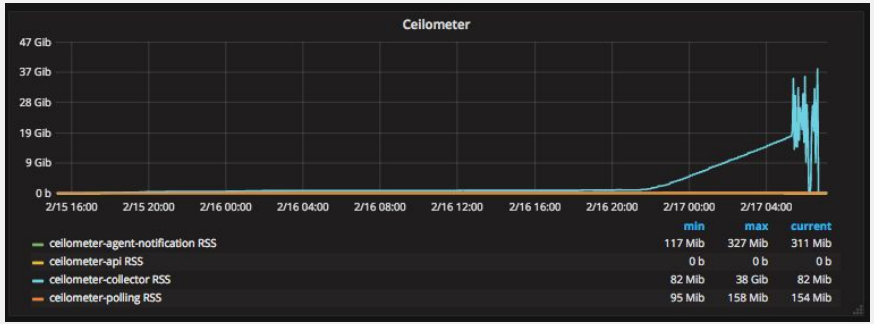


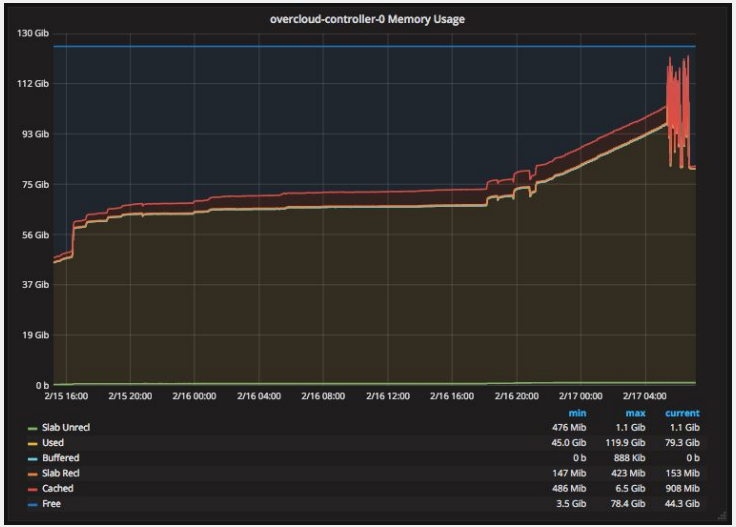
Issues - Gnocchi Slow Status API



Issues - Ceilometer Unlimited Prefetch

Set rabbit_qos_prefetch_count or make friends with the Linux OOM





Issues - Other

Nova

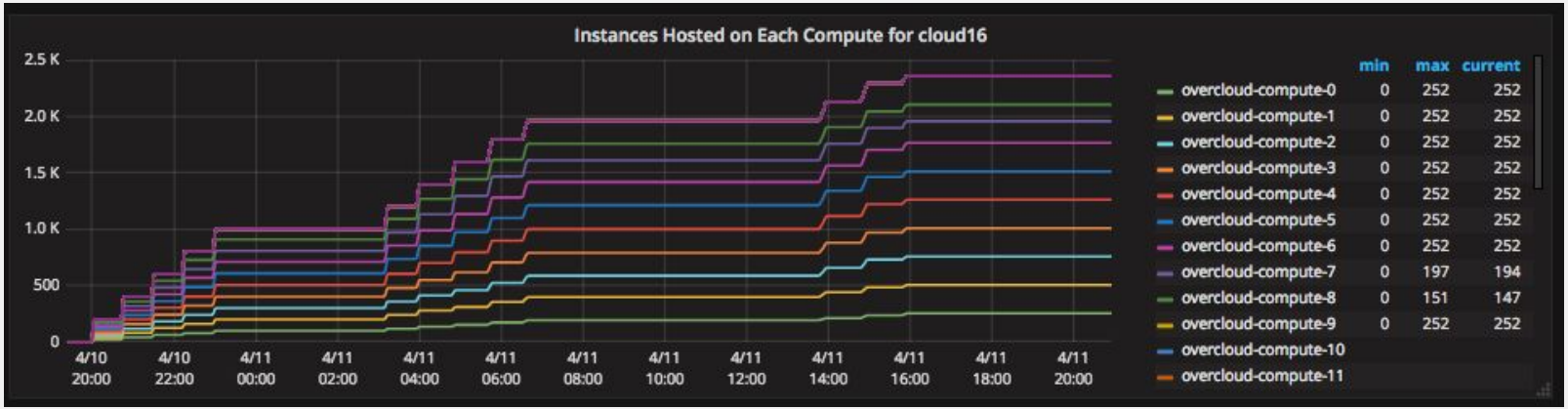
- virtlogd max open files
- Difficult to distribute small instances evenly
- Was able to schedule > max_instances_per_host
- Overhead memory for tiny instances

Hardware

- Uneven memory on some nodes (128GiB vs 64GiB)
- SMIs due to Power Control settings in BIOS
- Potentially a Slow Disk in the Ceph Cluster

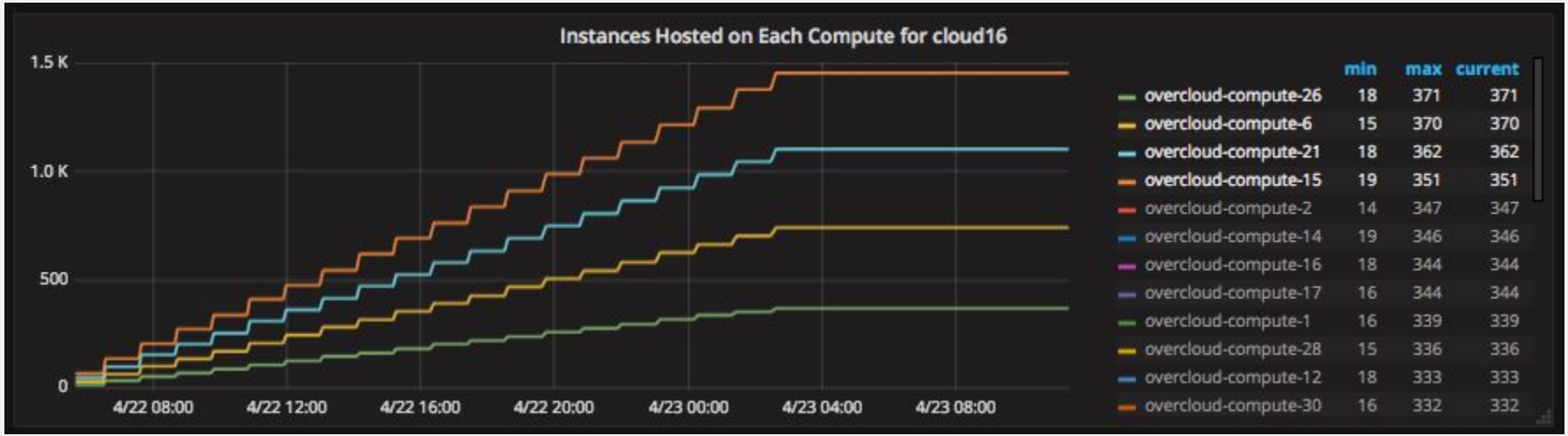
Issues - Instance Distribution (virtlogd)

Limits to 252 Instances on each Compute



Issues - Instance Distribution

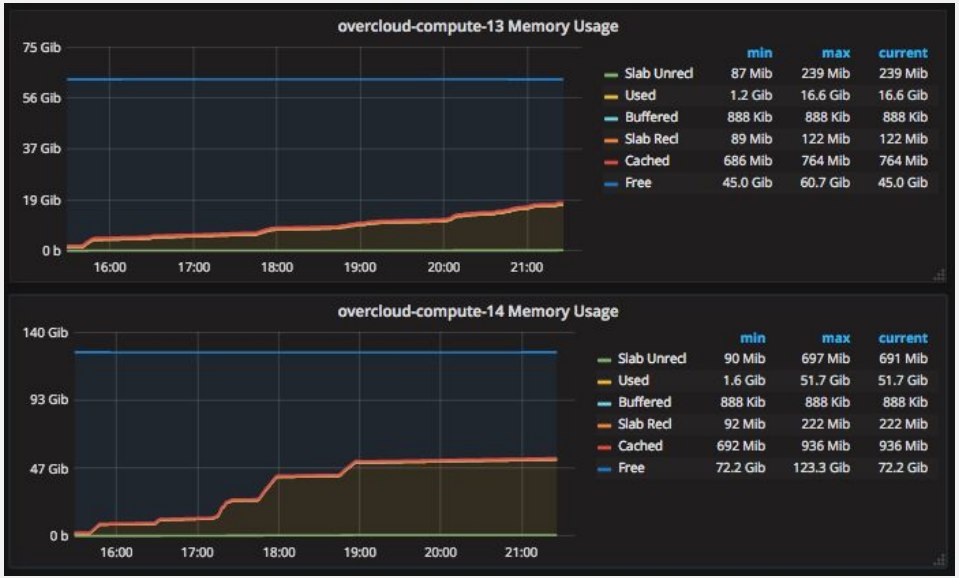
Max_instances_per_host was set to 350

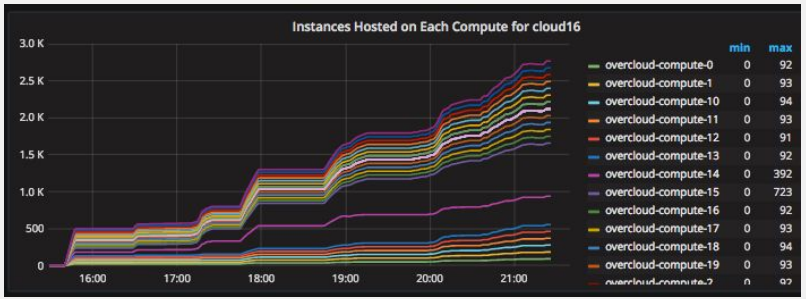


Issues - Uneven Memory

One Compute has 128GiB vs 64GiB of Memory

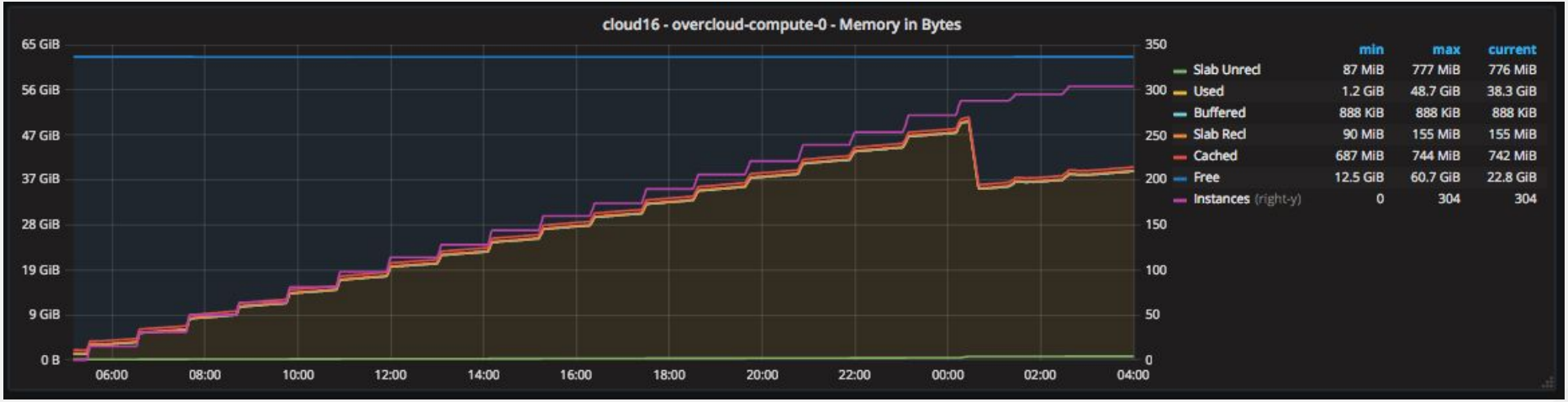
Set ram_weight_multiplier to 0 to remove “high-memory preference”





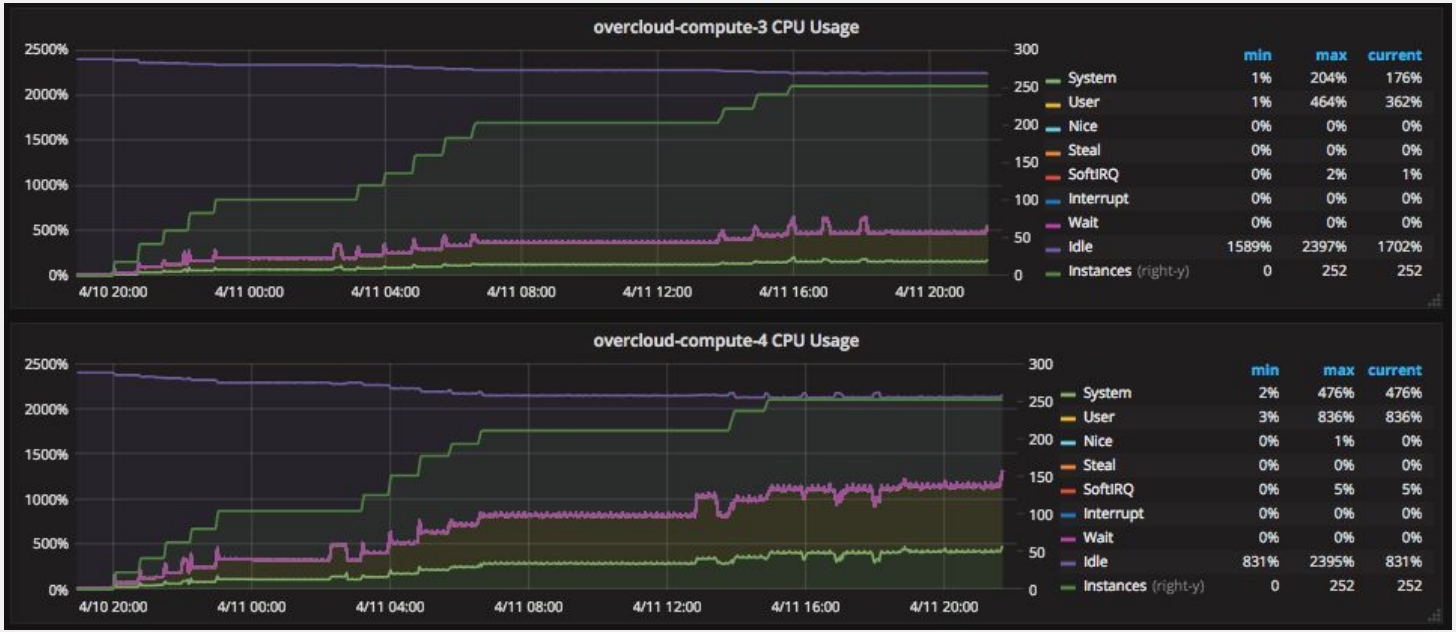
Issues - Overhead memory for tiny instances

Used Flavor m1.xtiny - 1 vCPU, 64MiB Memory, 1G Disk



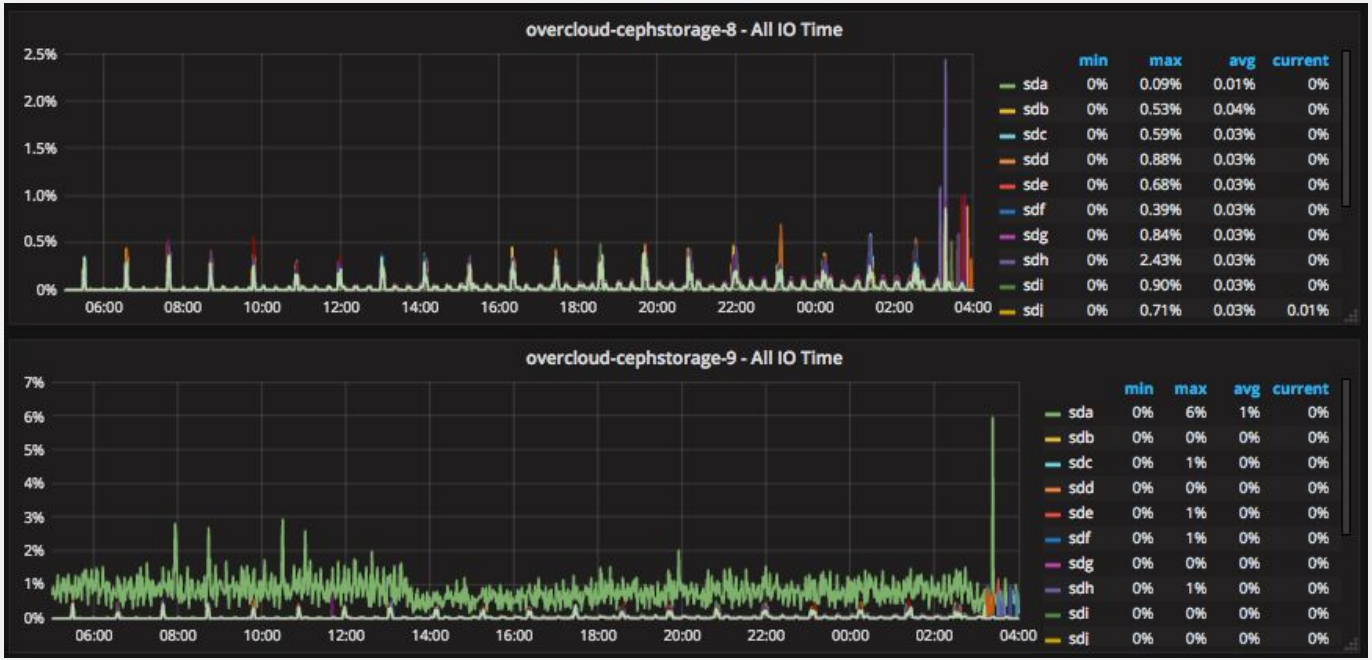
Issues - SMIs using more CPU

Overcloud-compute-4 has 480 SMIs every 10s resulting in higher CPU util, Set “OS Control” in your BIOS power settings…



Issues - Slow Disk in Ceph

Consistent Greater Disk IO % Time utilized on one Ceph Node's OS Disk



Future Gnocchi Performance and Scale Testing

Investigate Metricd processing responsiveness/timings
Investigate Ceph tuning and Ceph BlueStore
Isolating ingestion of new measures and retrieval APIs
Contribute benchmarks into OpenStack Rally

Development influence

How it changed Telemetry roadmap

Gnocchi 4 will include new features based on those feedbacks!

- API batches Ceph measures writes (merged)
- Use multiple Ceph Objects for Backlog (reviewing)
- Speed up backlog status retrieval (TBD)

Ceilometer will simplify the architecture

- Deprecation of the collector in Pike, disabled by default
- Removal of the collector in Queens

Conclusion

Why you should do the same at home

- Make performance teams and developers work hand-in-hand to make sure:
  - The software is understood and tested correctly
  - You got quality feedbacks from testers
    - And sometimes patches!
  - Developers focus their effort on the right places
    - Early optimization is the root of all evil
- The OpenStack Telemetry stack scales to up 5k nodes easily
  - We'll reiterate and we'll try to reach 10k
  - It's not clear that the rest of OpenStack scales that fare anyway

Q&A

redhat.

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos