CERN

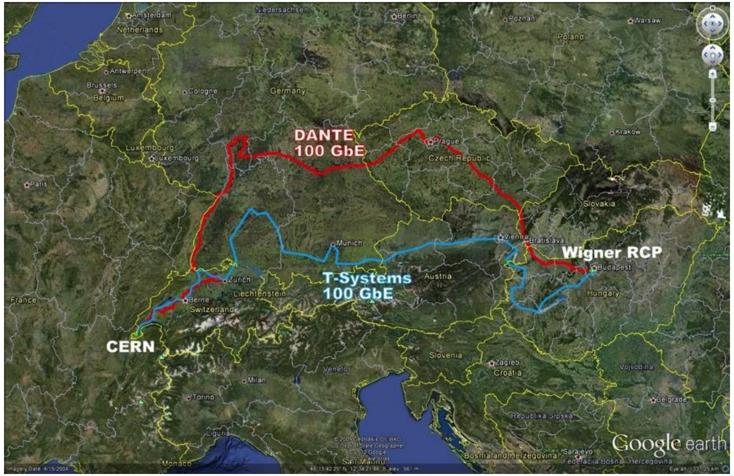
Scaling the CERN OpenStack cloud

Stefano Zilli

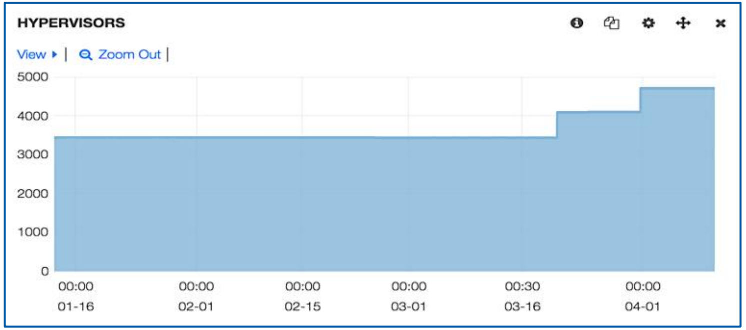
On behalf of CERN Cloud Infrastructure Team

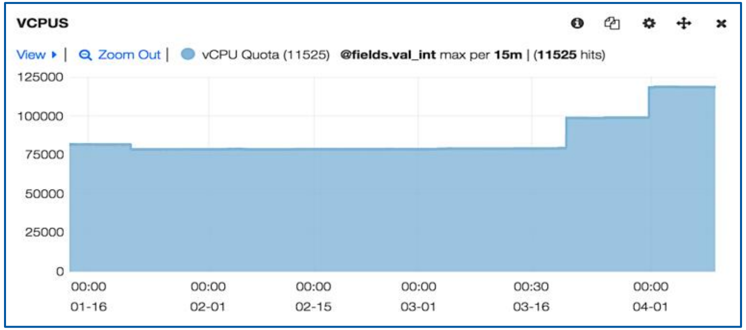
CERN Private Cloud - Numbers

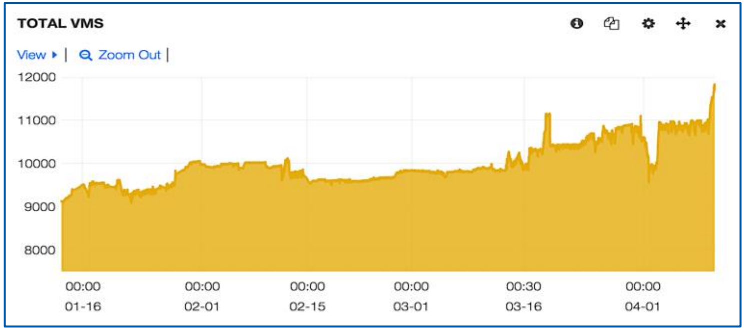
- Based on OpenStack Juno
- Spans between 2 datacentres
- 4700 hypervisors
  - 120000 cores
- 11000 VMs
- 1500 users
- 1800 projects

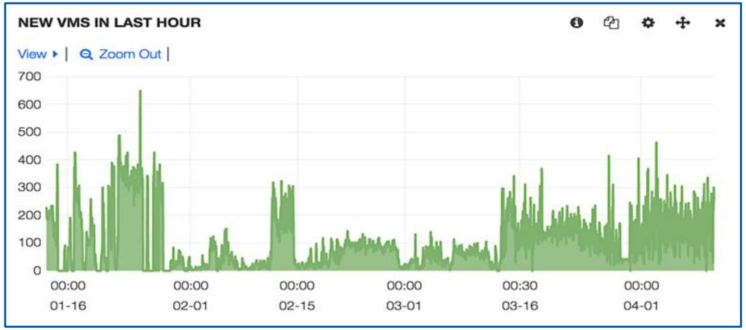


CERN Private Cloud - Numbers





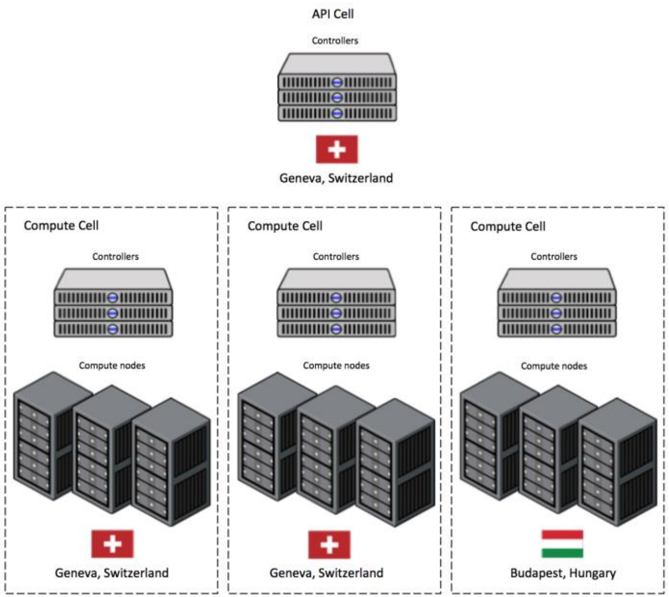




Nova Cells

Nova Cells

- What are cells?
  - Groups of hosts configured as a tree
  - Top level cell runs API services
  - Child cells run compute nodes
  - Every cell has its own database and RabbitMQ



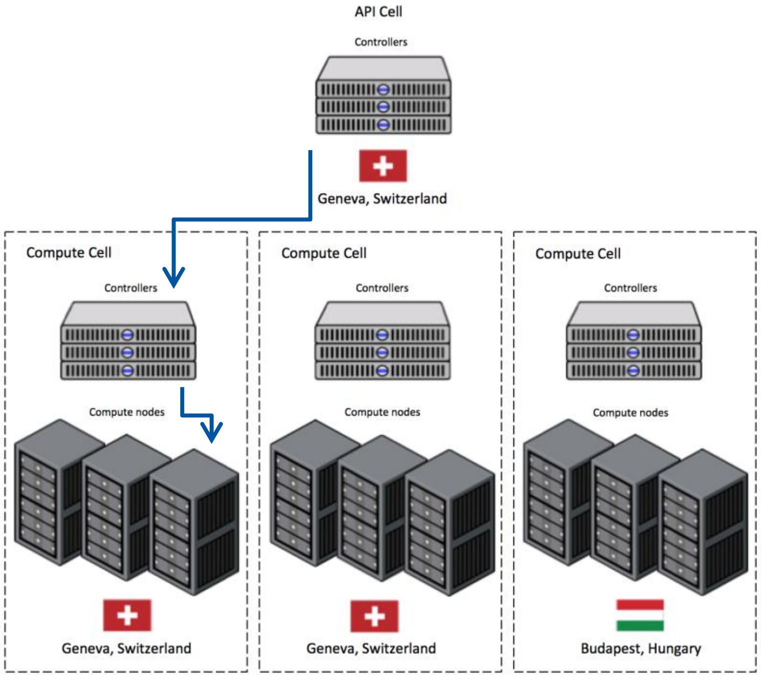
Nova Cells

- Why cells?
  - Expand our Cloud adding new cells
  - Single entry point for all our resources
  - Cloud architecture hidden to the user
  - Used by big companies (Rackspace, eBay/Paypal, GoDaddy, Walmart)
  - It’s becoming the default configuration

Scheduler

Scheduler

- Two schedulers
  - Cell scheduler
    - Top cell level
    - Decides the cell
  - Node scheduler
    - Compute cells
    - Decides the hypervisor



Cell Scheduler

- Possible to associate projects to cells
- Chooses the most appropriate cell depending on free resources
- By default not datacentre aware
  - Improved to make it datacentre aware
  - Cells mapped to a datacentre
  - Possible to specify a datacentre

Node Scheduler

- Time to schedule VMs increases with the number of nodes
  - Performance benefits with small cells (~200 nodes)
- Schedulers update the database only when the target hypervisor is decided
  - With many simultaneous requests and multiple scheduler failures due to race conditions

Controllers

Controllers

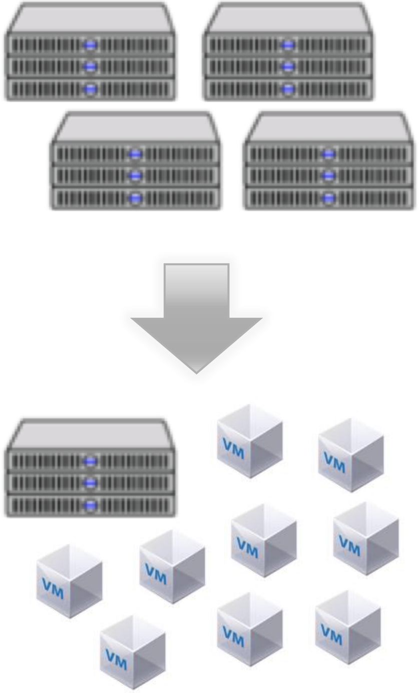
- What is a controller?
  - Node that runs the OpenStack services to manage the Cloud (scheduling, manage volumes, manage images, …)
- Two type of controllers
  - Top cell controllers
  - Child cell controllers

Top Cell Controllers

- Host API services
    - Entry point for the users
    - Need to be HA
- Problems
    - Several services share the same host
        - Different scaling needs
        - Different update paths
    - Physical machines with many idle resources

Top Cell Controllers

- Moved the majority of OpenStack services to virtual machines
    - Small and medium size VMs
    - Freed some previously idle resources
- Services are separated
    - Scaled our based on the service need
    - Services updated independently
- Subset of services still on physical machines in case of cold reboot scenario



Child Cell Controllers

- Nova services to manage compute nodes
- Initial idea
  - HA deployment
    - Multiple message brokers
    - Multiple cell schedulers
  - Problems
    - Race conditions on some nova components
    - Resources barely used

Child Cell Controllers

- One physical node per cell
  - Nova services
  - RabbitMQ message broker
- Move to a distributed cell architecture
  - Availability zones span between different cells
  - If a cell goes down new VMs are spawned on other cells

Metering

Metering

- Ceilometer as metering service
  - Collect usage information about cloud resources
  - Notifications and active polling
- Bottlenecks observed
  - High amount of requests to Nova, Keystone and Glance APIs due to active polling
  - Stress on backend database due to large amount of inserts

Metering

- Parallel API deployment for metering
  - No impact on user experience
  - No risk for the rest of the infrastructure
- Tried different backends
  - Not satisfying experience with MongoDB
  - Currently using HBase
  - Still some problems due to Ceilometer data structure

Conclusions

- OpenStack scales and meets our needs for LHC Run 2
- Working with upstream for new features and bug fixes
- Cells deployment allows us to scale out easily adding new groups of nodes
- Further work on ceilometer and nova cells to work smoothly at our scale

Questions?

Docs: http://cern.ch/go/9drV

Blog: http://cern.ch/go/9ktl

CERN

www.cern.ch

Other Slides

Workflows

- Increased number of hypervisors, projects and users
- Automatic tasks on Rundeck
  - New projects creation
  - Quota updates
  - Notifications in case of interventions
  - Preconfigured tasks to be run by the sysadmins

Project management

- Delegation to project owners
  - Manage tenant users
  - Allow operator access to tenants
- Hierarchical multitenancy (future)
  - Allocate resources to child tenant

Puppet configuration

- Puppet managed
  - Separated hostgroups per service
  - Same services in all cells share puppet code
  - Specific configuration defined in hiera