From Zero to OpenStack

Richard Hartmann,
RichiH@{freenode,OFTC,IRCnet},
rh@globalways.net

Globalways AG

August 20, 2011

Outline

Outline

Who am I?

- Project & Network Operations Manager at Globalways AG
- Passionate about FLOSS
- freenode staff

What is Globalways?

- ISP & ICTP
- German-wide DSL network
- Own DC in Stuttgart
- Total of six POPs in Frankfurt, Stuttgart & Reutlingen
- Connected via redundant dark fibers, wavelengths & leased lines
- Several high-profile government and industry clients
- Almost all services virtualized with Debian on Xen
- Always looking for new people...

What is the cloud?

- Common buzz-word
- On-demand services
- Dynamic and elastic; assumption of unlimited resources
- Pay-as-you-go
- Different layers of magic
  - Physical infrastructure
  - Software
  - Business processes
  - End-user interaction
- Everything-as-a-Service

Problems with existing solutions

- Expensive once you reach certain thresholds of usage
- Vendor lock-in
- You can not deploy a cloud solution used by a third-party, yourself
- Thus, no possibility to mix your own cloud with identical third-party cloud seamlessly
- No private cloud unless you are a huge customer (government, etc)

Problems with existing solutions, contd.

- No true QoS in the back-bone
- No guarantee for minimum IOPS
- No on-site, in-person verification of compliance (locked cages, access control, etc)
- You may be breaking European/German simply by using any cloud service provided by a US company or any of its subsidiaries, without being aware of it... (PATRIOT Act)

How does OpenStack solve this?

- It’s FLOSS
- Open collaboration, everyone is welcome
- Your data stays yours; just export here and import there
- You can create your own cloud, use a third-party, or a mix thereof
- Picking from a wide array of service providers allows you to choose based on the features you need, e.g. compliance, QoS, etc.
- European providers are not subject to the PATRIOT Act!

How does OpenStack solve this?

- Modular design; pick and mix the OpenStack projects you want & need
- Heavy use of sharding
- Highly asynchronous design
- Eventual consistency is considered acceptable
- Emphasis on reliability and error-avoidance
- Blueprints, unit tests & continuous integration

Design goals of all OpenStack projects

- Highly modularized: Add new functionality easily and quickly
- High availability: Scale to high workloads without failing
- Fault tolerance: Isolate faults automatically to minimize effects and cascades
- Recoverable: Diagnose, debug, & rectify faults quickly
- Transparent & public governance
- Open standards: Community-driven, RESTful API

History of OpenStack

- 2010-03: Rackspace decides to open source Rackspace Cloud
- 2010-05: NASA opens Nebula to the public
- 2010-07: Formal launch of the project & first design summit in Austin
- 2010-10: Release of Austin
- 2011-02: Release of Bexar
- 2011-04: Release of Cactus
- 2011-08: Over 100 developers working on OpenStack
- Young, aggressive, efficient project

Projects within OpenStack

- Core
  - Glance: Manage virtual machine images
  - Nova: Manage virtualization solutions, e.g. Xen, KVM, UML, LXC, VMWare...
  - Swift: Object storage
- Incubation
  - Keystone: Authentication frame-work
  - Dashboard: web UI
- More to come...

Swift

- Object storage
- No inherent limit on object sizes
- Supports striped transfers to increase speed
- Built-in redundancy
- Compability layer to emulate Amazon S3
- Scales extremely well

Layout of Swift components

- Auth nodes (obvious...)
- Proxy nodes
  - Central gateway for access control, etc
  - Handles most faults within Swift
  - Does not cache data, forwards only
  - Optional rate-limiting capability
- Storage nodes
  - Store accounts, containers and objects
  - Main part of Swift
- Ring
  - Keeps everything together
  - Distributed hash; determines storage locations
  - Partitioned to keep memory foot-print small

Layout of Swift components

- Replicators
  - DB replicator
    - Hash-based synchronization
    - Resynchronisation via trivial high-water mark
    - Simply looks at DB uid, DB-local id, timestamp and hash
    - Initial seed to new machines via plain rsync!
  - Storage replicator
    - Hash-based, as well
    - Synchronization based on rsync
    - Use of partitioned rings keeps directory indices small and thus rsync fast

Layout of Swift components

- Account Reaper
  - Asynchronous
  - Allows for lazy deletion!
- Auditor
  - Asynchronous
  - Trawls the complete storage network
  - Quarantines and replaces bad objects automagically
  - Absolute must; failed disks are easy, bit rot is hard!

Initial deployment

- Simply install Ubuntu LTS 10.04 and add the OpenStack repository
- Debian packages coming along, as well
- Dell's Crowbar is looking very promising to automate most of deployment; based on Opscode's Chef Server
- URLs with detailed instructions at the end of this presentation :)

Deployment best practices

- Five or more fully separate zones
  - Physical location
  - Power and backup power
  - Network connectivity
  - WAN interconnects
  - You can start with one machine per zone; with proper planning, you can simply add machines to the zones to scale as you grow
- Three or more copies of every object, at max one per zone
- Even if you lose a whole zone, you can replicate from two zones to two others, essentially cutting replication time in half

Deployment best practices, contd.

- Separate management network from data network
- VLANs are OK, two physical interfaces preferable
- Never use RAID; failures should be known to Swift ASAP!
- Contain maintenance, upgrades etc to one zone at a time
- Quick replacement/repair? Just do it. Might take longer? Take disk/machine out of order so Swift can start to replicate in the background
- Swift provides good fault detection & handling; still monitor on top of that, yourself!
    - OSSEC to aggregate & parse log files
    - Zabbix, Nagios, Ganglia, etc. for everything you can think of (e.g. load, smart status, sensors, disk space, port status...)

Our deployment

- Still in the testing phase, but looking very good
- Five zones across three data centers
- 1 Gbit/s switching within zones
- Zones interconnected via MPLS over redundant 10 Gbit/s lines
- Switching & MPLS ensure line-rate speeds with minimal overhead

Is it worth it?

- Short answer: Yes
- Yet, OpenStack is still somewhat intimidating
- The next release, Diablo, is scheduled for September and considered ready for general consumption
- Working with a well-designed product is fun
- You know the people who build and use OpenStack care about your data

More resources

- `http://www.openstack.org`
- `http://wiki.openstack.org`
- `http://glance.openstack.org`
- `http://nova.openstack.org`
- `http://swift.openstack.org`
- `http://tinyurl.com/openstack-releases`
- `http://tinyurl.com/openstack-admin-cactus`
- `https://github.com/dellcloudedge/crowbar`
- `http://www.opscode.com`

Thanks!

- I hope you had fun listening to this talk
- Feel free to contact me with questions