IBM CloudFirst Factory

OpenStack as a public and private cloud: deployment lessons learned

Nikolay Marin

Based on work done by Ryan DeJana, Andrea Greggo, Angel Tomala-Reyes, Esteban Arias Navarro, Dan Krook, Franz Friedrich Liebenger Portela, Lisa DeLuca, Pablo Barquero Garro, and Dima Rekesh

Agenda

- CloudFirst Factory overview: what is it?
- CloudFirst Factory: implementation
- Deploying OpenStack as a public and private cloud
- Custom OpenStack extensions
- Summary and next steps

CloudFirst Factory: what is it?

- A cloud lab aimed at promoting innovation
- Multiple environments: IBM internal and IBM external
  - http://cloudfirst.demos.ibm.com [external]
  - http://cloudalpha.demos.ibm.com [external]
  - http://cloudfirst.democentral.ibm.com [IBM internal]

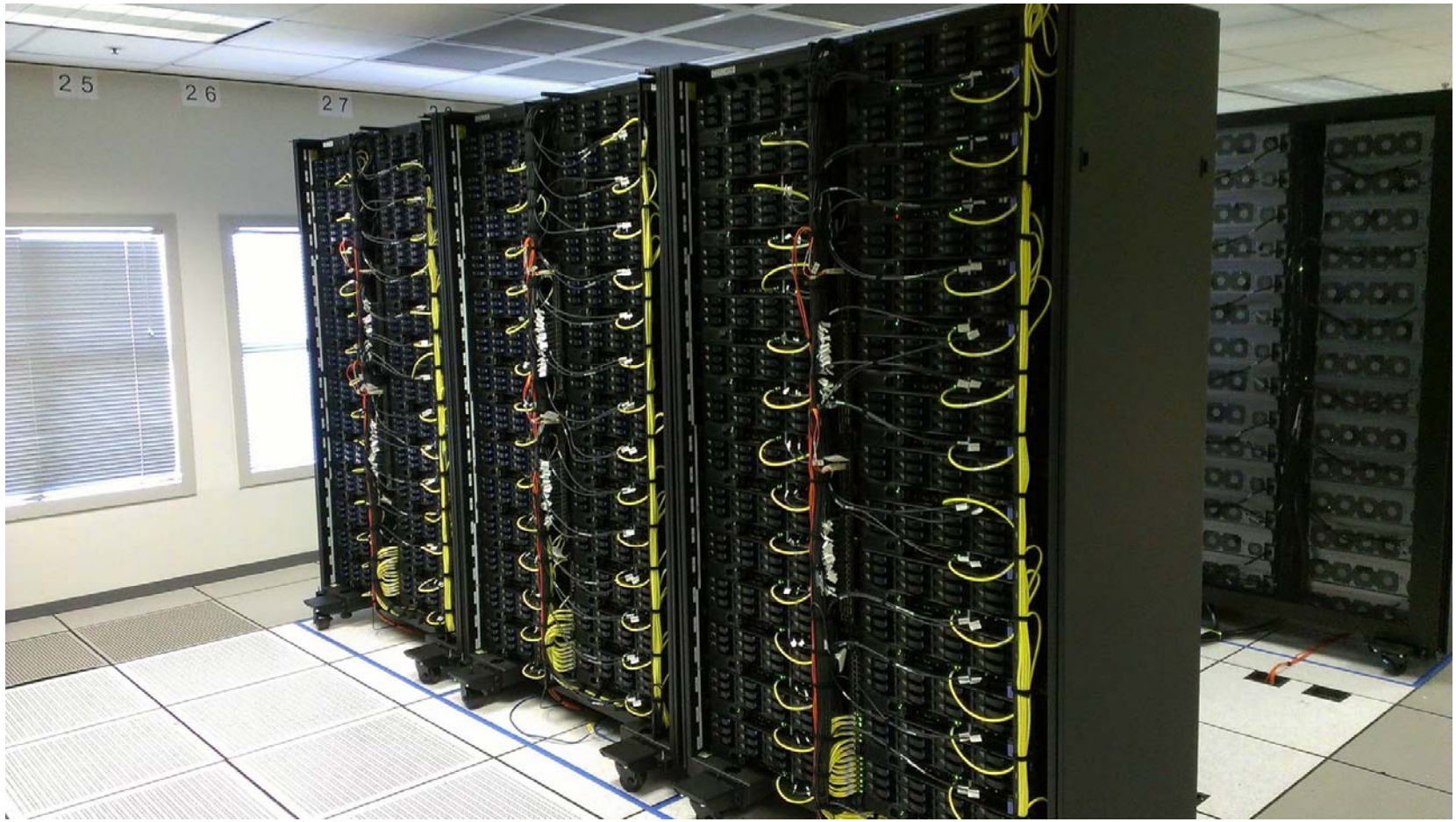
- Self-service interface
- Custom on-boarding process: users submit projects, if approved, a project is created for them and they are assigned as the admin

CloudFirst Factory: why OpenStack?

- Biggest Open Source Cloud Manager project
- Open APIs
- Works with Open hypervisors (e.g. KVM)
- Rapid pace of innovation
- Rich set of features
- Relatively simple
- Strategic to IBM

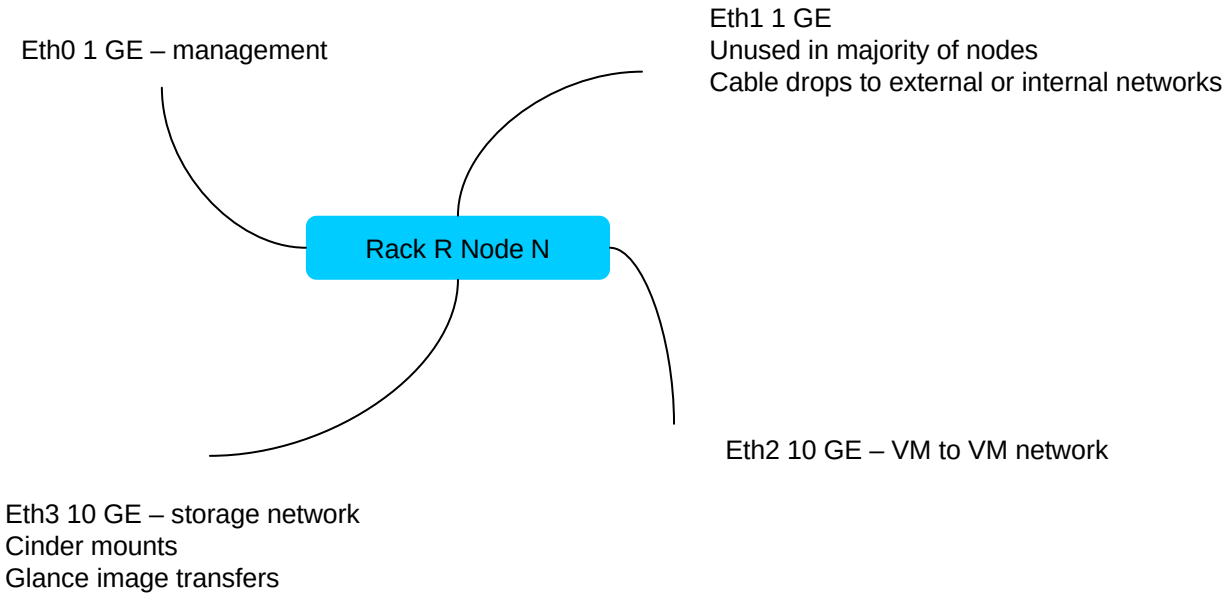
CloudFirst Factory Hardware

- 6 iDataPlex racks, 28 servers each
- Servers built for Big Data: lots of direct attached storage, dual 10 GE interconnects
- 1 GE network for management



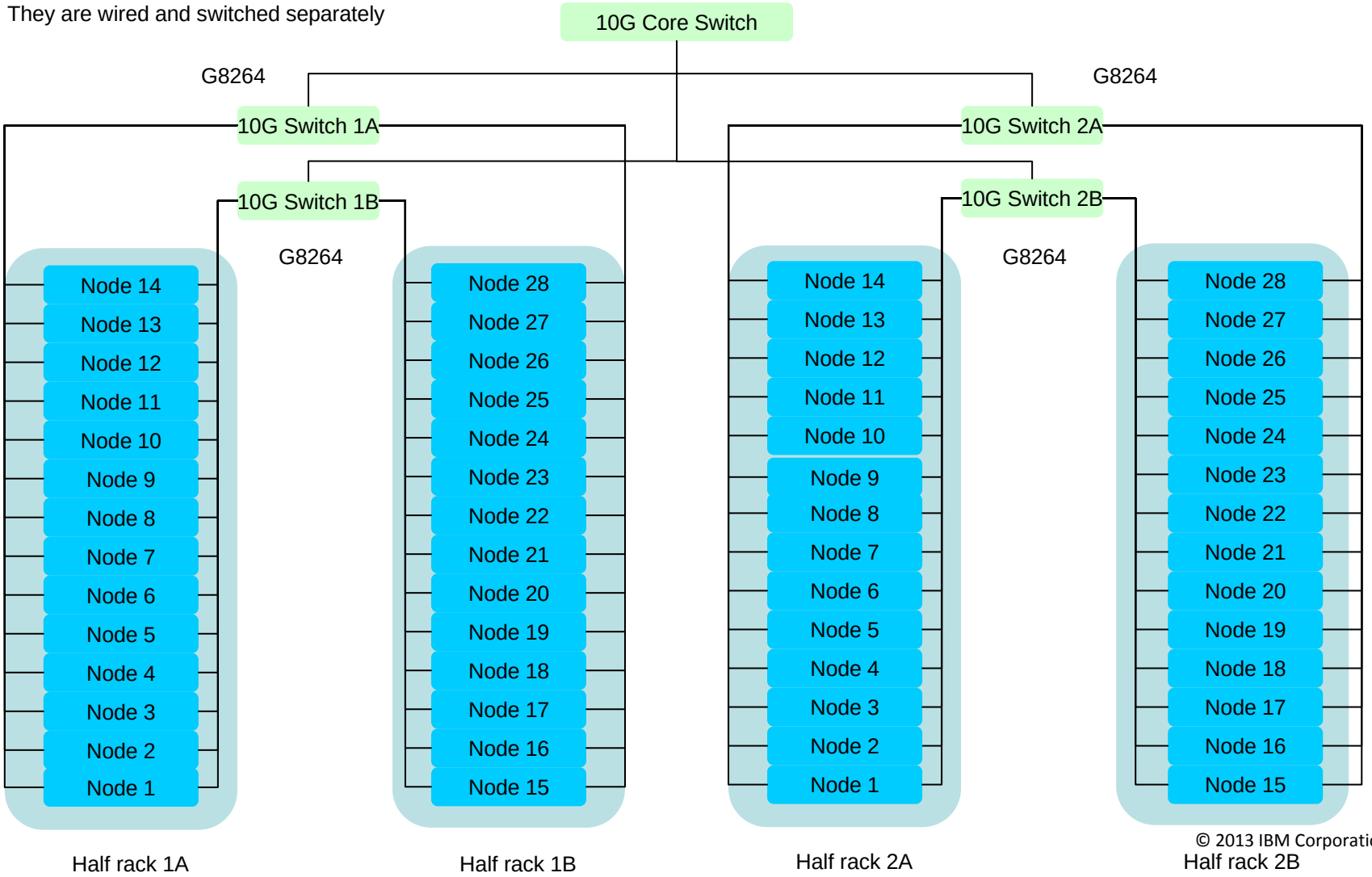
Cloud First Factory: A Compute node

- -SB- IBM System x iDataPlex dx360 M3 server
- -SB- Mellanox ConnectX-2 EN Dual-port SFP+ 10GbE PCI-E 2.0 Adapter
- 1x 1GE on-board adapter (management)
- 128 G RAM: 16 * 8GB (1x8GB, 2Rx4, 1.5V) PC3-10600 CL9 ECC DDR3 1333 MHz LP RDIMM
- Intel Xeon Processor X5670 6C 2.93GHz 12MB Cache 1333MHz 95w
- 12 * IBM 3TB 7.2K 6Gbps NL SAS 3.5" HS HDD. Each could get us ~130 MB/s on sequential writes
- IBM ServeRaid M1050 controller. Capped at ~600 MB/s total, but, crossflashed to LSI bios, scales to 1.8 GB/s



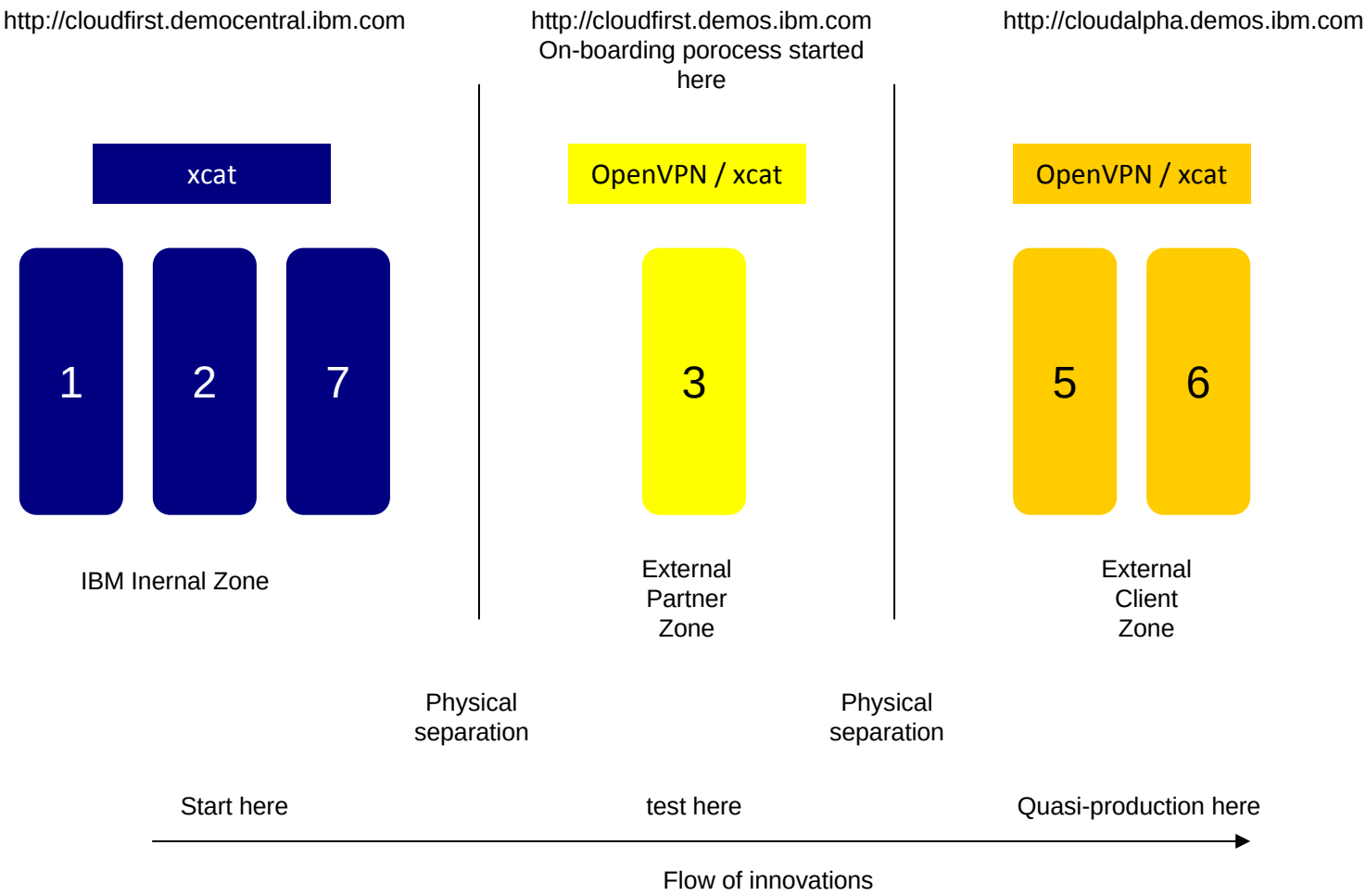
Cloud First Factory: 10 GE network

The 1 GE adapters are used primarily for management.
Each node has 2 * 10 GE adapters
They are wired and switched separately

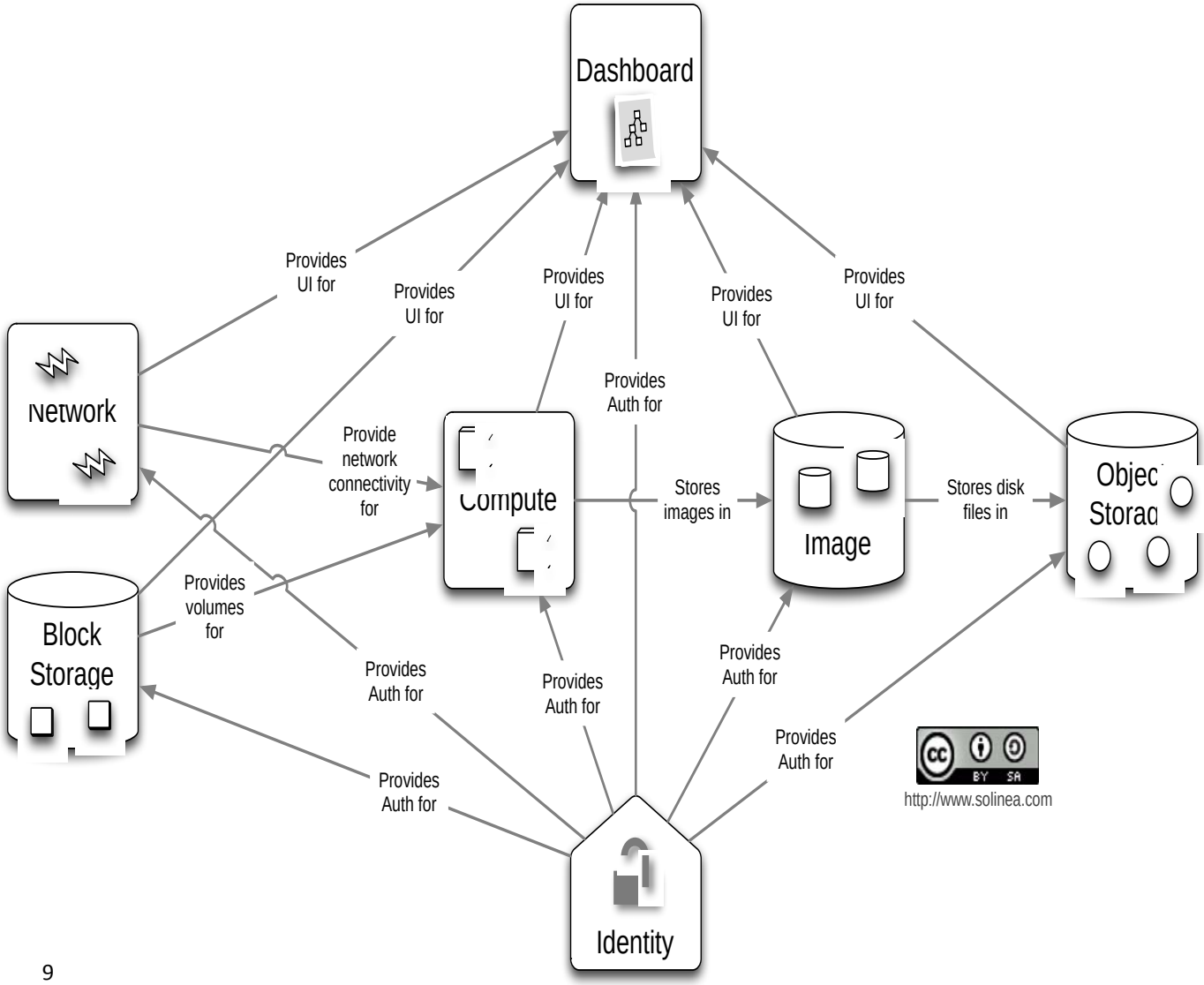


Cloud First Factory: Racks and Zones

- The equipment was split into three separate environment, each with a distinct purpose
- These were completely separate OpenStack installations



General OpenStack architecture



OpenStack Components:

Compute (Nova)

Block Storage (Cinder)

Network (Quantum)
Provision and manage virtual resources

Dashboard (Horizon)
Self-service portal

Image (Glance)
Catalog and manage server images

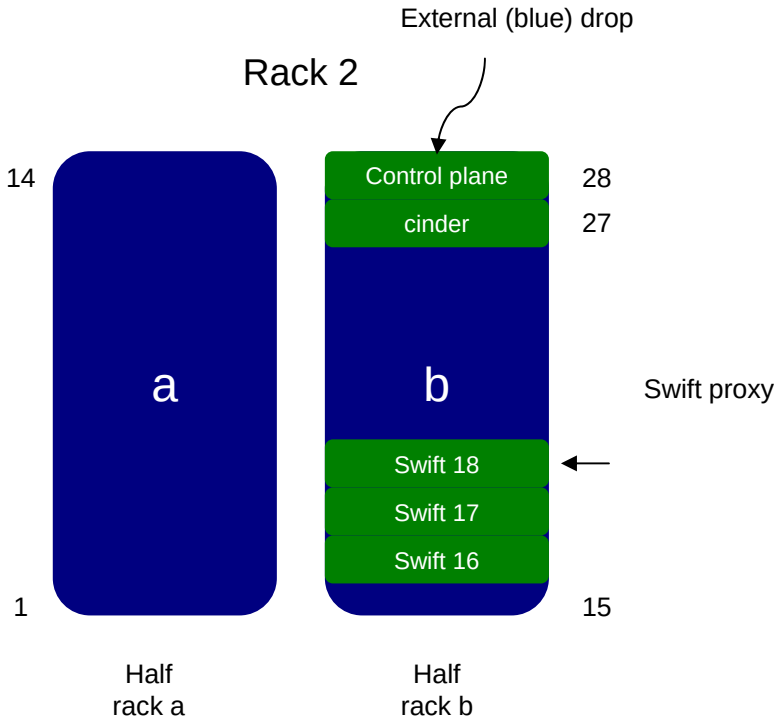
Identity (Keystone)
Unified authentication and authorization

Object Storage (Swift)
petabytes of secure, reliable object storage

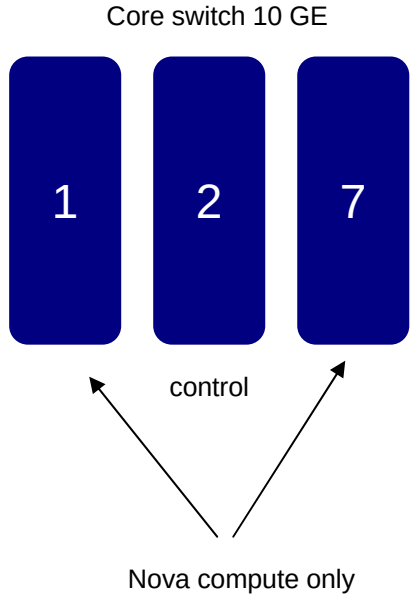
OpenStack on the IBM Intranet

This Zone was the first we set up and therefore, the simplest
Almost every component runs on bare metal, non-virtualized
We use xcat to set up OS (RHEL 6.4) on the nodes
Because this is IBM Internal network, security is not as important

http://cloudfirst.democentral.ibm.com



Rack 2 node 28

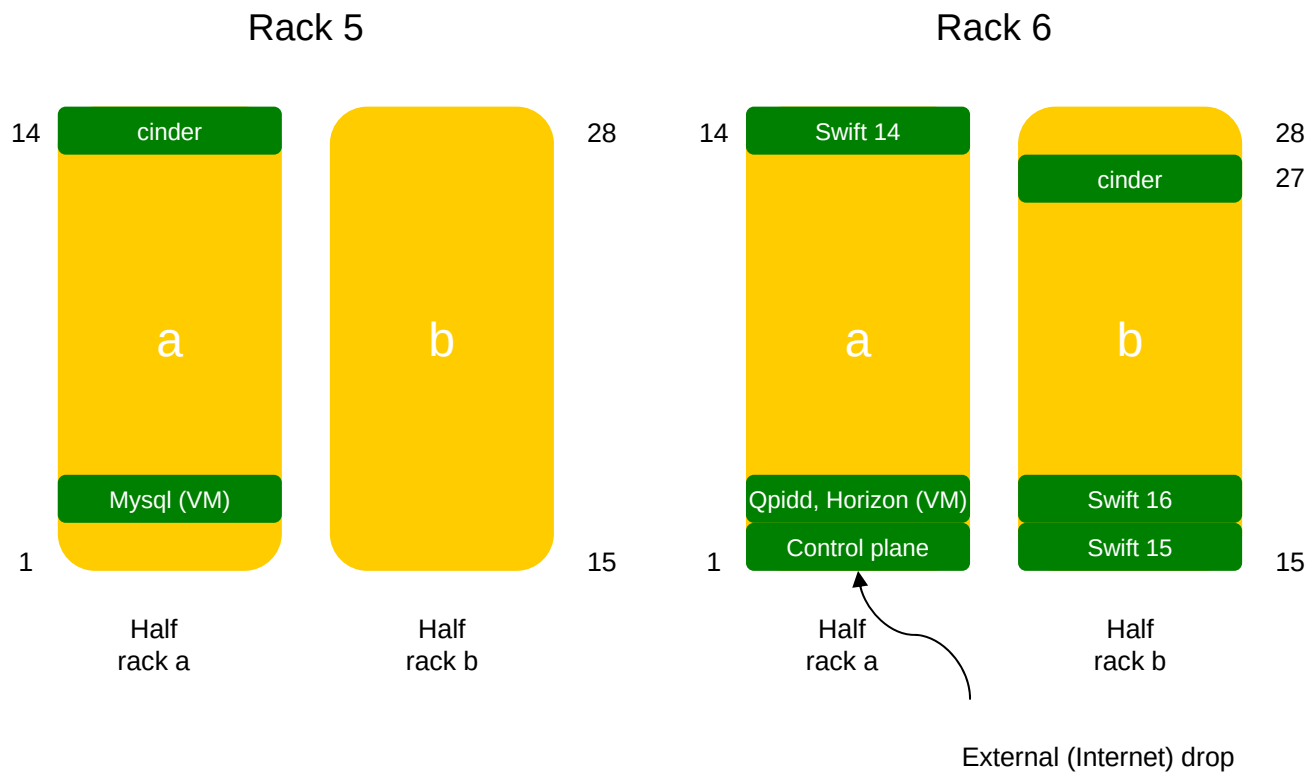


- mysql – the overall database
- qpidd (message processing)
- glance (images)
- nova network (gateway for VLANs, floating IPs)
- keystone (authentication / entitlement)
- horizon (user intefrace) – is currently running virtualized.

OpenStack in the Internet Zones

- The use of VMs is more widespread
- Still, not optimal – e.g. Swift needs to be in both racks
- Control node (rack 6 node 1) is the single point of failure
- Control node is also the major network bottleneck!

http://cloudalpha.demos.ibm.com



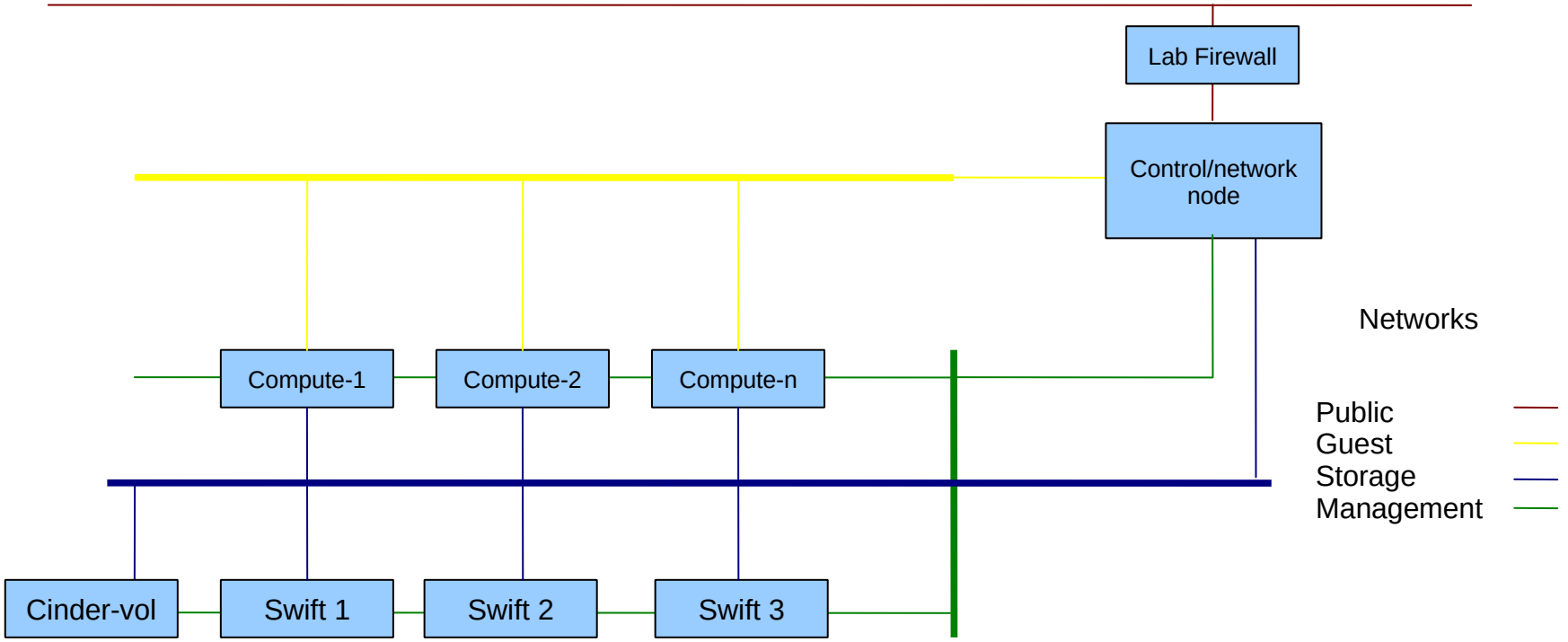
Being on the Internet: networking

Make a list of all entry points and reduce their number

- Each project is assigned a separate VLAN (nova networking)
- These VLANs are pre-created in the switches manually and are defined in the racktop switches as well as in the core switch
- Each VLAN starts with a restrictive policy (ports closed)
- Each VM gets assigned a private IP in that VLAN, which is not externally routable. Access to this VLAN from other private VLANs is controlled by the security policy
- The floating IPs are externally routable; access also controlled by the security policy
  - Limit the number of external IPs per project
- All external traffic routes through the network node
- The OpenVPN / jumpbox is used to get access to the hypervisor managed network

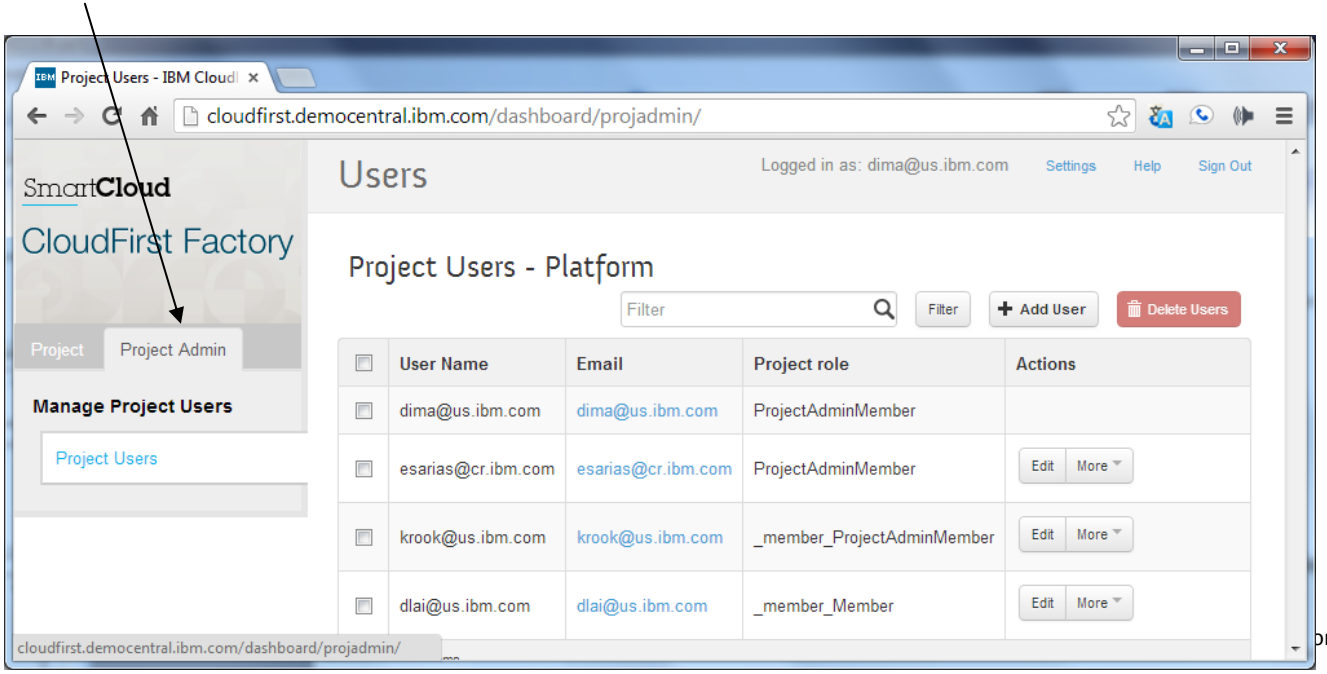
Running OpenStack on the Internet

Note that all traffic between VLANs flows through the control node
All traffic between nodes on floating IPs flows through the control node



User Authentication and Entitlement modifications

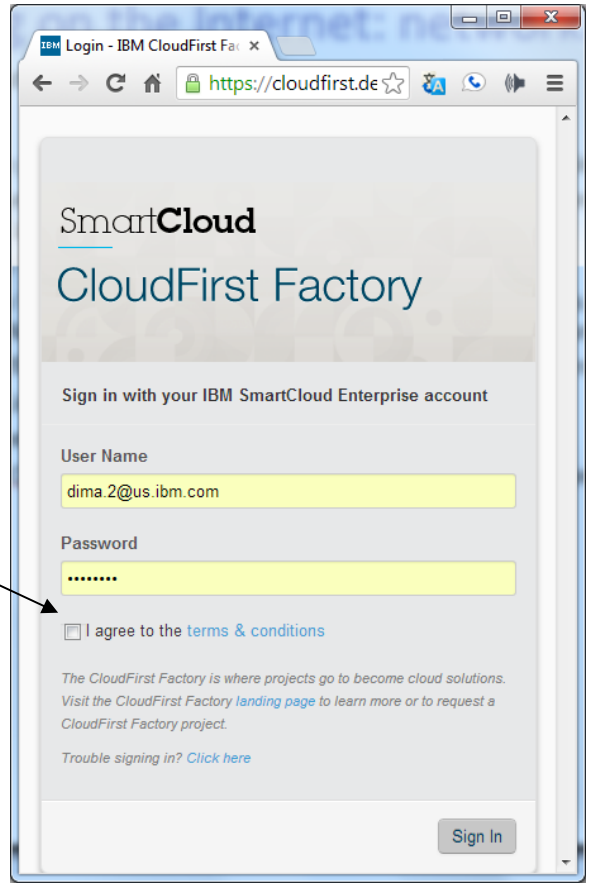
- Out of the box, Horizon assumes that the admin will create all IDs.
  - Not the way to go for a public cloud
- At IBM, we have two auth API: external (ibm.com) and internal (w3.ibm.com)
- We created keystone adapters for both systems
- Both adapters require that the userid exists in the OpenStack Zone
- We created a special role, Project Admin, who can manage users within their project
  - Project Admins create userids in OpenStack, then the users can log in and authenticate against w3.ibm.com or ibm.com, depending on the zone
- We created a custom tab for Horizon to expose this functionality



Being on the Internet - two portals

Make a list of all entry points and reduce their number

- The internal Horizon portal is unmodified and for admin only
- This portal is available only on the internal network (behind OpenVPN)
- The external, modified portal has no admin tab at all
- The external portal has a custom skin
  - Note the terms and conditions checkbox
- The external portal has the custom Project Admin Tab
- Image upload was disabled
- VNC console was removed in the yellow zone
- External portal runs over https (out of the box comes with http)
- Horizon was scanned for vulnerabilities and largely passed



Being on the Internet - API

- We implemented a secure reverse proxy for UI services (that support Horizon)
- We moved the admin Horizon UI into a VM
  - In the future, this could support load balancing and VM updates
- Implemented a reverse proxy for web services
  - By default, they run over a number of different ports and are not secure
  - Just one port to worry about
  - In the future: support for load balancing and VM updates

Being on the Internet - Images

- IBM has internal standards for images that are exposed to the Internet (ITCS 104)
- Userid / password policy; ssh keys preferred as the way to access
- Lock down the ports (iptables exposes 22 only by default), SELinux
- Semi- automated OS patch management
- All floating / public IPs are periodically scanned
- Disabled user-initiated image upload

The Custom On-boarding process

- IBM Forms Experience Builder used to create a custom workflow
- Single-Signon with ibm.com (or guest) allows you to launch it
- Once the form is filled out, email is generated to the admin. They go into the tool to approve requests, which creates the ID for the ProjectAdmin and associates the role with their ID, allowing them to manage users in the project
- The same instance of on-boarding application for all three zones
- Modification to keystone was made allowing ProjectAdmin to create users

Modularity tweaks

In general, moving OpenStack services into VMs is not a bad idea

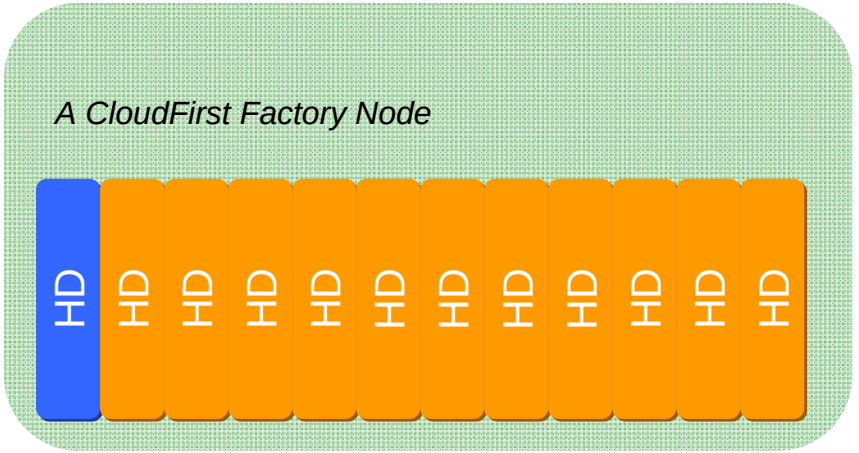
- Easier and cleaner to upgrade – no need to change the bare metal OS.
- Easier to move VMs between hosts and to back them up
- Eventually, resiliency – multiple services running in multiple VMs..
- But, is it fast enough? Is disk I/O fast enough?
- But , Is it too complex, to run inside a VM?
- But, what creates these VMs (devops aspects) ?

- OK: mysql in a VM
- OK: QPID to own VM
  - But, issues with performance, so moved back to bare metal
  - Ryan: briefly describe issue with the memory leak
- OK: Horizon in a VM
- Maybe: Glance in a VM
- Not OK: swift in a VM
- Not OK: cinder in a VM

Resiliency considerations

- Ability to recover from the loss of a single HD
  - OS: we have a pair of mirrored disks in RAID 1
  - But, what's so important? The OS can be reloaded easily. It's the question of how quickly..
  - Compute nodes: image cache could be lost
  - Compute nodes: VM disks; we are converting to RAID 10 for regular VMs
  - Swift: a bunch of JBODs
  - Glance: RAID 10
  - Cinder: RAID 10

- Ability to recover the loss of a single node
  - A lot of work to do to get here!

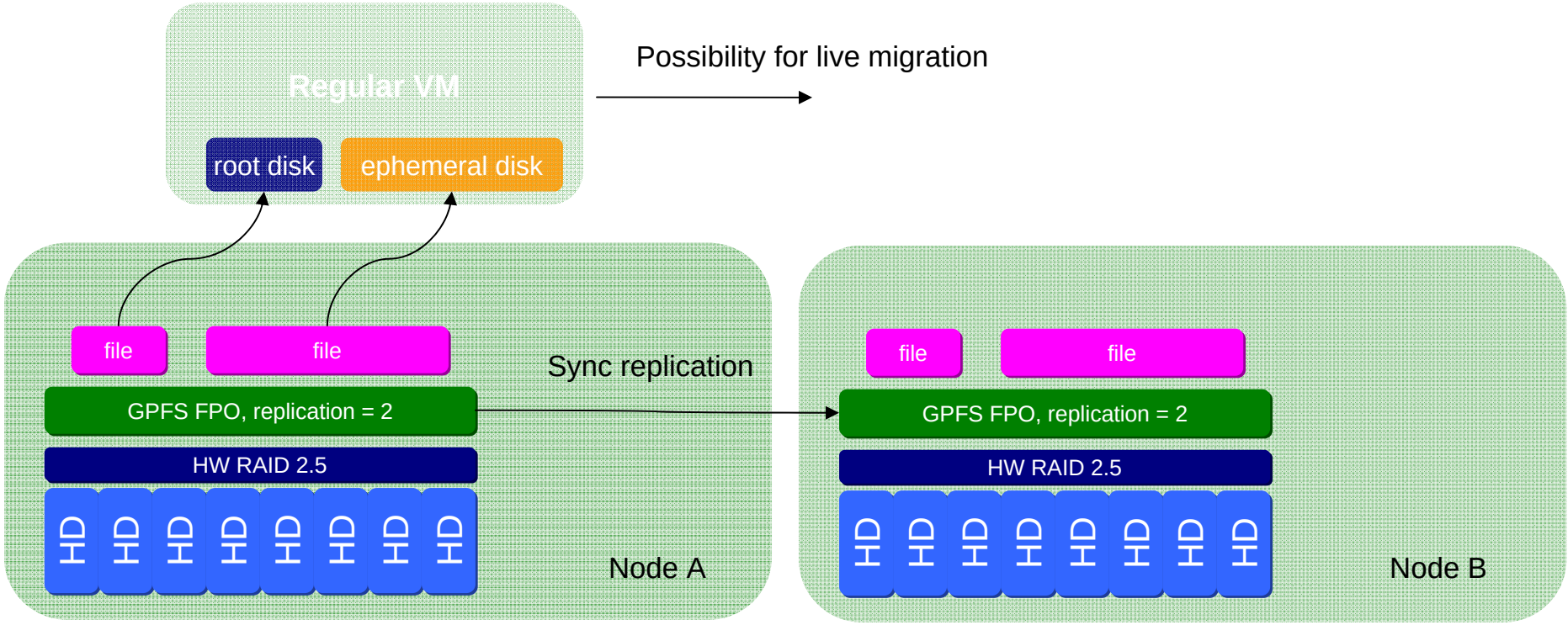


Supporting Big Data: what it means

- The Cloud must work well with direct attached storage
  - Central storage just does not scale; look at our topology!
  - Hadoop requires it
- Virtual Machines must control the noisy neighbor effect
  - Imagine N VMs hitting a RAID 0 array
- I/O overhead should be minimal
  - Putting the disks of your VMs into a file just is not fast enough
- Options:
  - RAID 0, dump VM disks into it
  - give each VM a separate disk
  - raw disk pass-through
  - PCI pass-through

Challenge: Regular VM I/O performance

Performance of the ephemeral disk: ~ 50% of bare metal or less, depending on network
Write traffic must cross network (replicate = 2)
Ephemeral storage is backed up
VM Placement: anywhere

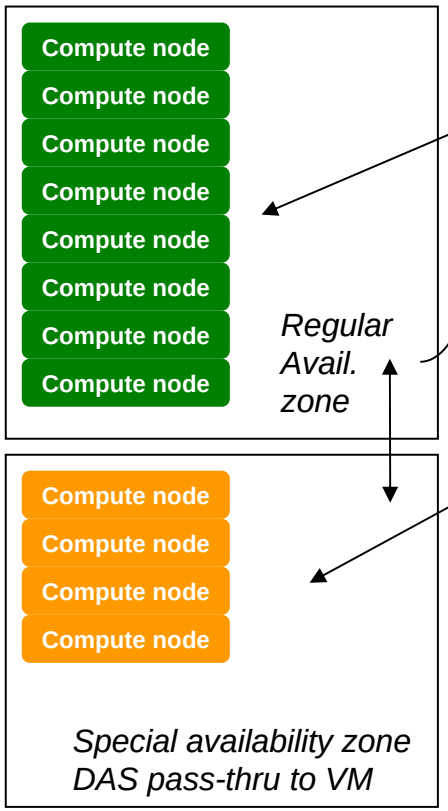


- *Redundant disks support the VM root disk and may provide redundancy via GPFS FPO*
- *Live migration of the VM possible to the second replicated pair*
- Moderate noisy neighbor problem
- Both Root and Ephemeral disks are replicated across network

Better support for High I/O VMs in the Cloud

Push complexity out of IAAS into user space

Approach A:
special HW / OpenStack Availability Zone
Special "Super Large" VM flavor or bare metal



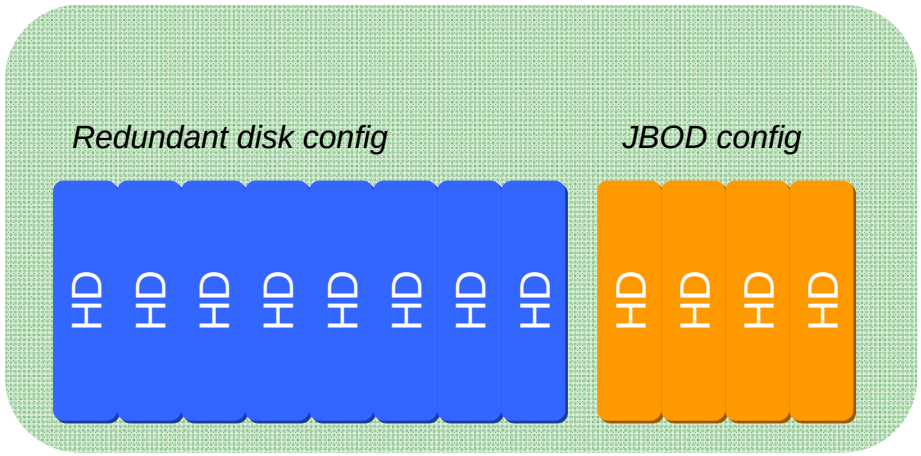
Regular VM flavors provision here

Zone boundaries can be dynamic

Superlarge VM flavor provisions only here (regular flavors do not)

The superlarge VM takes over the node and disks are passed to it raw for maximum performance
This could be done with bare metal provisioning as well

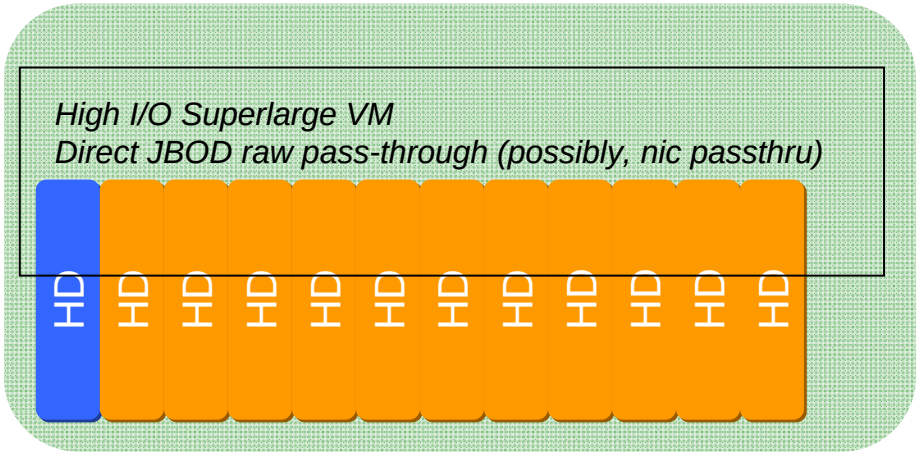
Approach B: all nodes the same.
Reserved direct attached hard drives for special "High I/O" VM flavors



Some fraction of DAS is reserved for special flavor VMs.
This fraction (hypervisor disk config) could also be dynamic
These VMs get one or more JBODs as direct pass through for maximum performance
The JBODs cost extra
This also eliminates noisy neighbor effects on disk

Approach A: Superlarge High I/O VM option

Simple.. Implemented in CloudFirst Factory
The VM takes over the entire physical node and grabs all its resources
Performance goal: > 90% of bare metal (100% if implemented via bare metal)
Placement: static OpenStack availability zone (simplest) or any node (with live migration)



- *Disk 0 reserved for the VM root disk and may provide redundancy via GPFS FPO*
- *Live migration of the VM root disk possible, but the data will not move.*
- *Snapshotting / image capture of the VM root disk only*

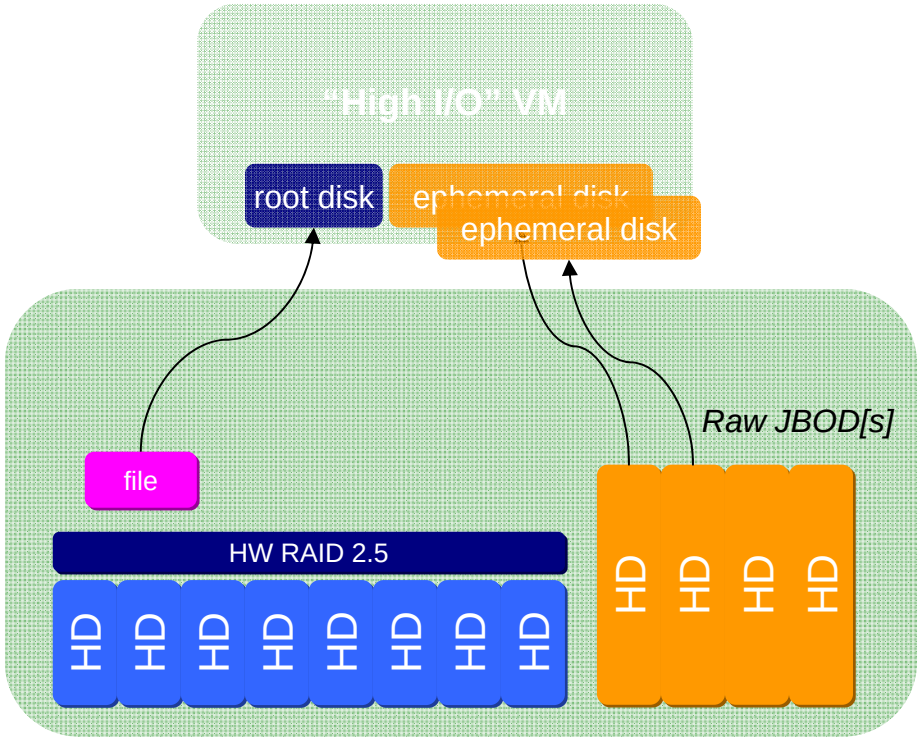
How to implement?

a. Special OpenStack availability zone. Boundaries managed by an async process
b. Tweak the placement algorithm to contain fragmentation (e.g. avoid provisioning into empty nodes) + async VM live migration consolidation
c. Live migrating away smaller VM at provision time

But.. Only one instance size.

Approach B: “High I/O” VM in Cloud FirstFactory

Relatively simple..
The VM has a small or modest size root disk and a large “ephemeral” disk
Performance of the ephemeral disk: > 90% of bare metal
Placement: anywhere



- *Redundant disks support the VM root disk and may provide redundancy via GPFS FPO*
- *JBODs are passed through directly to the ephemeral disk of the VM (one or more)*
- *Live migration of the VM would only involve root disk (data will not move)*

A spectrum of VM sizes: 1, 2, 3.. N disks each
Eliminates the noisy neighbor problem
Reduces system load on image capture
Provides high performance storage
Uses up the extra disk capacity of SCE
% of JBOD disks may vary..
For maximum impact, hypervisor disk config should be dynamic, allowing autonomic VM placement optimization

Direct attached disk passthru: implementation details

- Simple set of updates to nova.virt.libvirt.driver enables, with the following assumptions
  - Disks and CPUs are allocated in a 1:1 ratio - no CPU over commit allowed
  - Only soft reboot is supported as hard reboot rebuilds the libvirt.xml file
    - Support would require tracking which VM had which disk and in what order
  - Create process is single threaded on the compute node in order to easily determine disk usage.
  - Migration is neither supported nor prevented.
  - Disks are not erased up after VM deletion

Direct Attached Disk Passthru: Code Changes

- Add lock to method
  - **@lockutils.synchronized('directDiskLock','disklock')**
  - def spawn(self, context, instance, image_meta, injected_files,...
- Disk detection and allocation
  - Read the list of disks available
    - Add new method to the CONF object to return the list of supported disks
  - Loop over the list, determine if a disk is already in use
    - Leverage virtinst.VirtualDisk.path_in_use_by to determine if disk is already in use by a VM
  - Attach available disks to the block_device_mapping object using the local volume type
- Update reboot to support only soft reboots

Migration from Folsom to Grizzly

This was a major migration of a heavily used environment – three racks

Concerns about uptime, data integrity

- Created a test environment consisting of two nodes; rehearsed upgrade / downgrade there multiple times until we got it right
- Database schema changed, so it needed to be modified and data in it, preserved
- There were shutoff VMs in compute nodes which were in deleted state with incorrect hostname in MySQL. This prevented nova compute from starting (widespread).
- Some deleted / shutoff VMs were present in more than one node (incomplete previous attempts to migrate them?)
- Some VMs were active in different nodes from what MySQL thought
- We used IBM OpenStack Grizzly EE. It does not include Horizon or Swift; so we had to complement it, luckily with compatible versions
- Other than that, the upgrade was straightforward
- The VMs remained accessible throughout, no reboots.

In Summary

- OpenStack remains an emerging technology
- It is not mature yet
- Error handling not robust
- Understanding the flow of calls and messages is needed
- Large volume of message based rpc calls
- Logging is not optimal (either too much or too little)
- You must be willing to look at code
- Networking (nova-network) is complicated
  - Multiple bridges
  - IPTables configuration not straightforward

CloudFirst Factory

http://cloudfirst.demos.ibm.com

Nikolay Marin

Going Forward: issues to consider 1/2

- OpenStack is not VSphere
  - It uses commodity hardware, may fail. This probably should not change.
- Network availablity: the way to go is to have multiple network gateways (to avoid single points of failure)
  - This currently means nova-network (vs Neutron)
- each compute node should not be connected to external network (security...)
- each compute node only supports VMs on it; this means that if 1 node goes down, only those VMs will be unreachable
- Should we use a shared (clustered) file system?
  - Slower disk performance because of the network traffic involved in data sync
  - Reduces overall storage capacity
  - But,
    - Allows for live migration of VMs
    - May enable better availability of VMs
    - Balance may be, shared f/s between 2 or 4 nodes rather than global

Going Forward: issues to consider 2/2

- RAID
  - A good idea
  - Back cinder volumes by RAID 5/10
  - Back swift by RAID/5/10
  - No real need to back VMs (unless you have a specific requirement)

- Consider swfit for image (glance) storage
  - Scalable and replicated storage
  - Able to add addition glance VMs without needed a shared file system
  - Or just remind users that cloud instances may come and go...

- Use of Linux Containers as a form of virtualization
  - Low memory overhead
  - Virtually non-existent CPU overhead
  - Very well suited for High I/O
  - Much higher VM / physical node density
  - Much quicker to deploy