Introduction to Cloud Data Center and Network Issues

Agenda

- **Cloud Computing / Data Center Basic Background**
- Enabling Technology
- Infrastructure as a Service
  A Cloud DC System Example
- Networking Issues in Cloud DC

Brand New Technology ??

- Not exactly, for large scale computing in the past: utility mainframe, grid computing, super computer
- Past demand: scientific computing, large scale engineering (finance, construction, aerospace)
- New demand: search, e-commerce, content streaming, application/web hosting, IT outsourcing, mobile/remote apps, big data processing…
- Difference: aggregated individual small demand, highly volatile and dynamic, not all profitable

Cloud Data Center

	Traditional Data Center	Cloud Data Center
Servers	Co-located Dependent Failure	Integrated Fault- Tolerant
Resources	Partitioned Performance Interrelated	Unified Performance Isolated
Management	Separated Manual	Centralized Full Control With Automation
Scheduling	Plan Ahead Overprovisioning	Flexible Scalable
Renting	Per Physical Machines	Per Logical Usage
Application / Services	Fixes on Designated Servers	Runs and Moves across All VMs

Cloud Computing:

End Device / Client

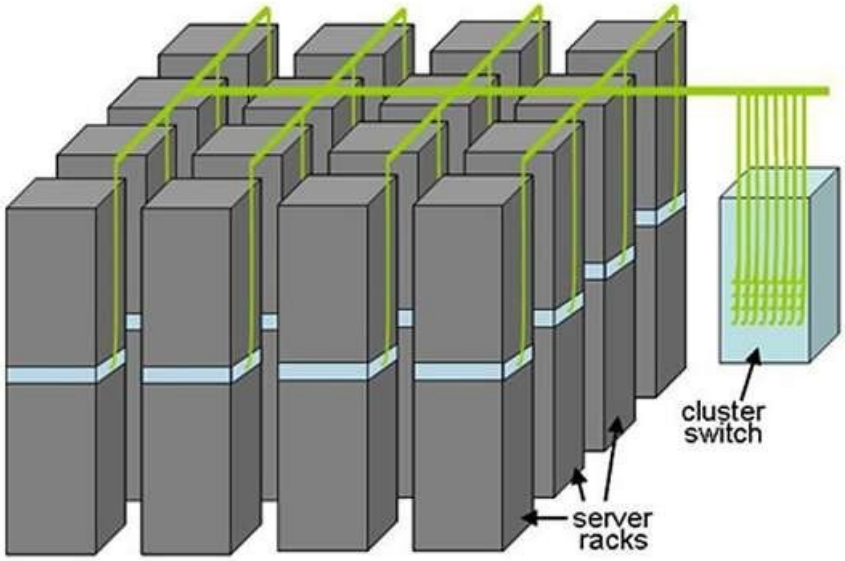
Common App.

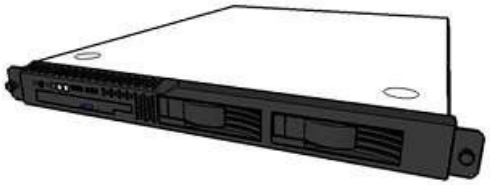
Platform Cloud Data Center

Cloud DC Requirements:

- On-Demand Self-Service
- Resource Pooling
- Rapid Elasticity
- Measured Usage
- Broad Network Access

Network Dependent





What's on Amazon?

Dropbox, Instagram, Netflix, Pinterest
Foursquare, Quora, Twitter, Yelp
Nasdaq, New York Times….
…and a lot more

FIGURE 1.1: Typical elements in warehouse-scale systems: 1U server (left), 7′ rack with Ethernet switch (middle), and diagram of a small cluster with a cluster-level Ethernet switch/router (right).

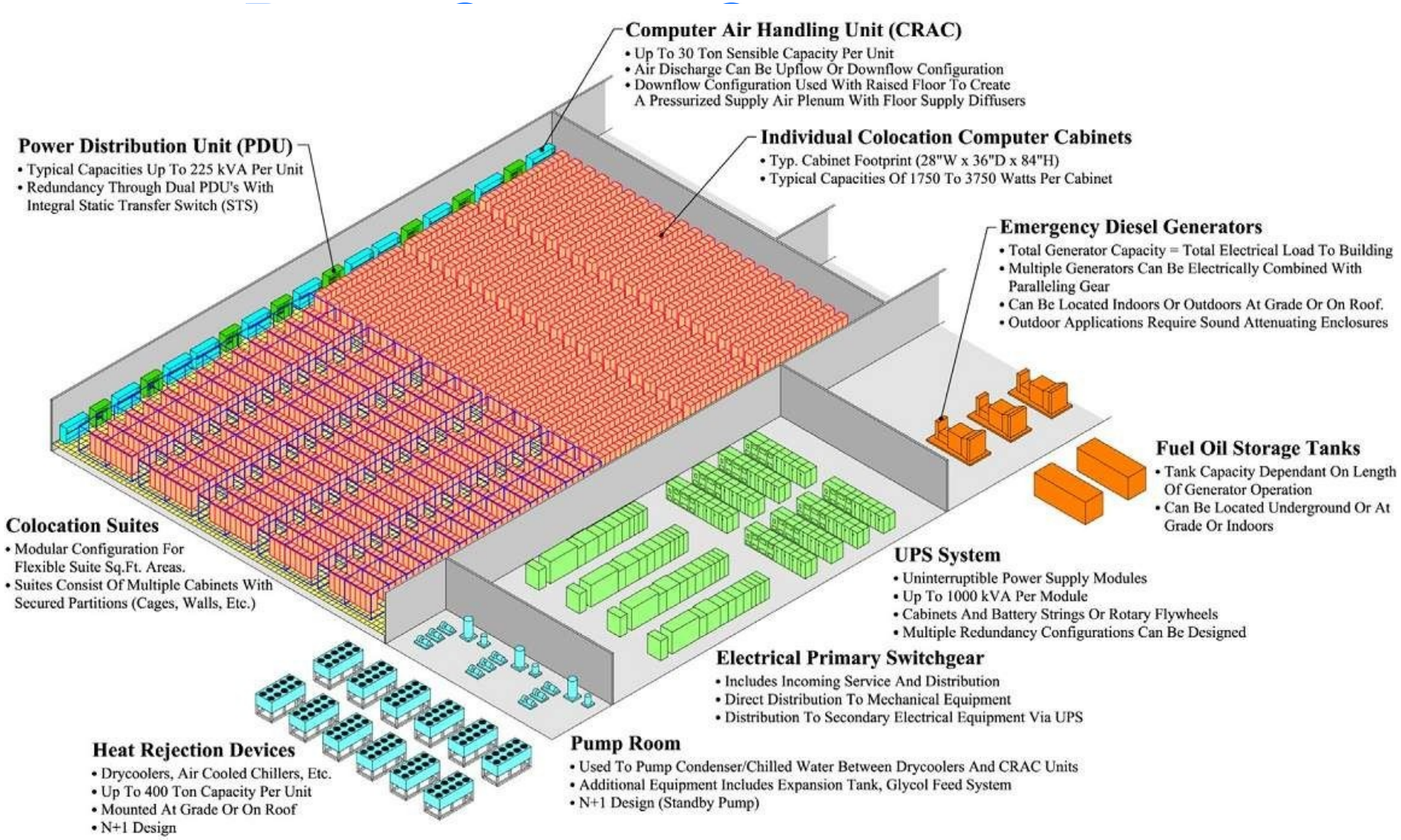


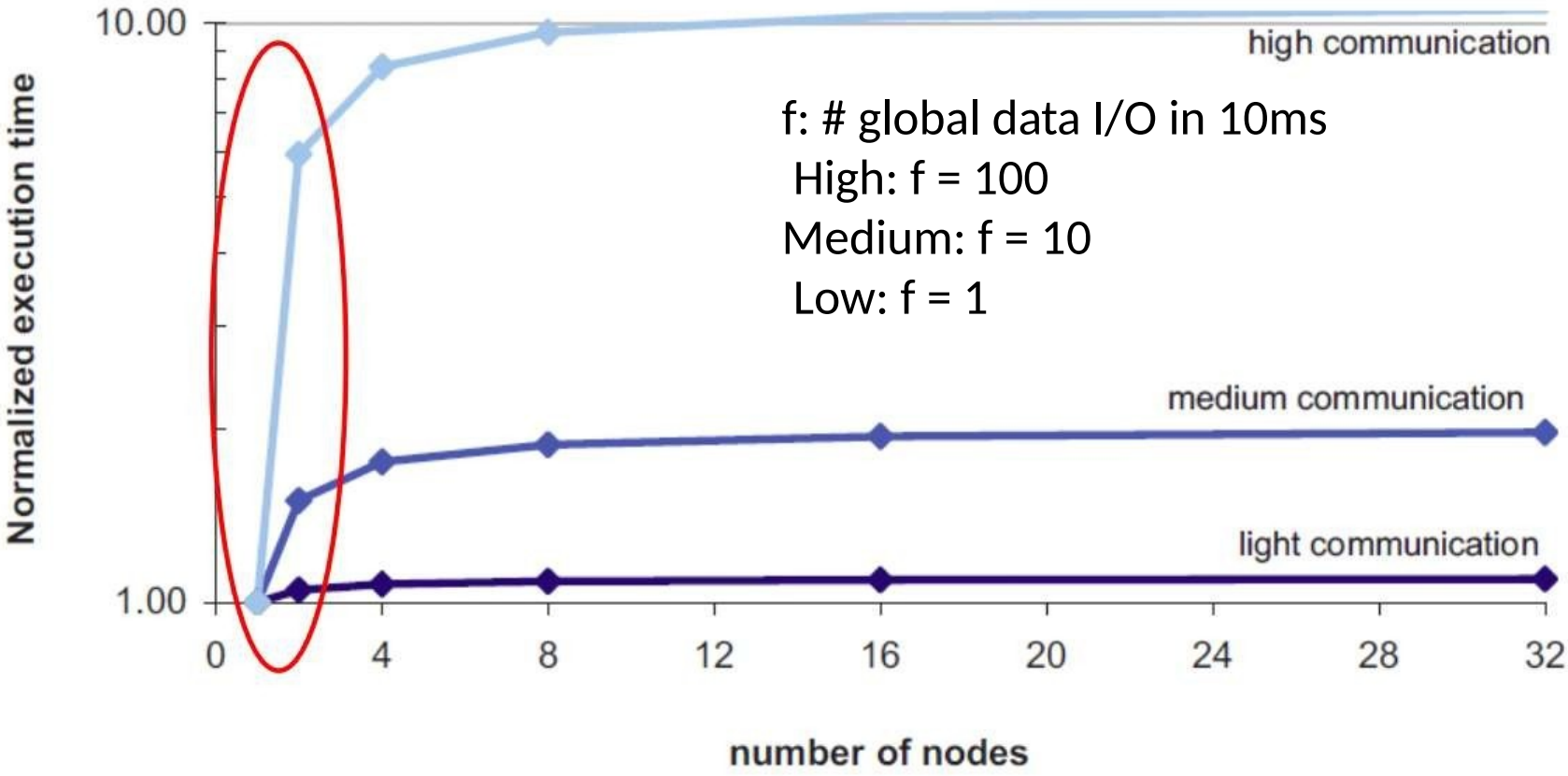
FIGURE 4.1: The main components of a typical datacenter (image courtesy of DLB Associates [23]).

Clusters of Commodities

- Current cloud DC achieves high performance using commodity servers and switches
  →no specialized solution for supercomputing
- Supercomputing still exists, an example is Symmetric Multi-Processing server
  →128-core on shared RAM-like memory
- Compare to 32 4-core LAN connected server
  - Accessing global data, SMP: 100ns, LAN: 100us
- Computing penalty from delayed LAN-access
- Performance gain when clusters grow large

Execution time vs. # of nodes

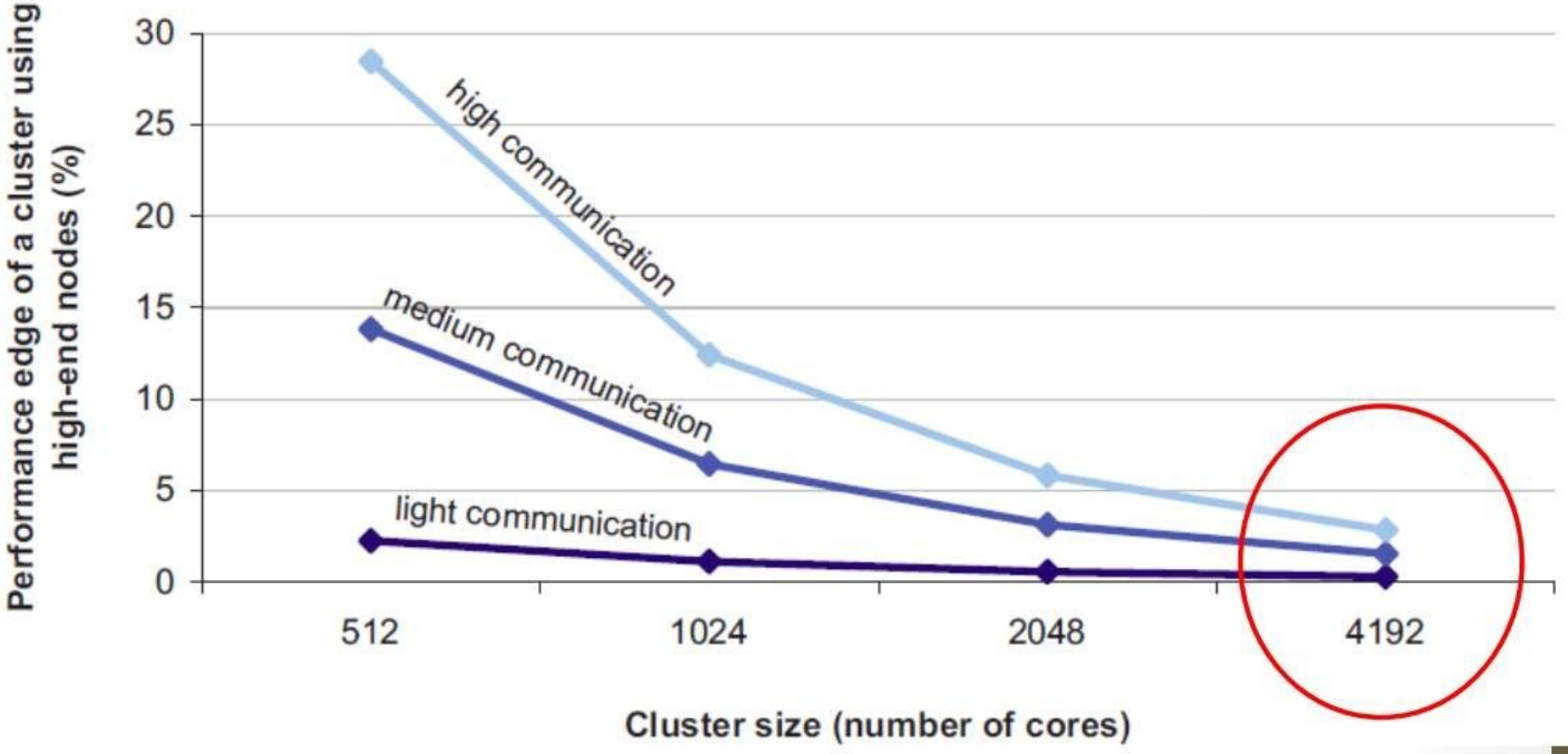
This is not a comparison of server-cluster and single high-end server.



f: # global data I/O in 10ms
High: f = 100
Medium: f = 10
Low: f = 1

Penalty most obvious from 1 → 2 nodes
Under light communication, little increase in execution time

Performance advantage vs. cluster size



- Compare the performance of building a cluster using 128-core SMP vs. 4-core SMP [Using 128-core SMP has performance advantage, but by how much.]
- More performance gain under high-communication scenario,
- As cluster size grows, performance advantage is smaller.
- With much higher cost for 128-core and only 5% performance advantage ➔ worth?

Agenda

- Cloud Computing / Data Center Basic Background
- **Enabling Technology**
- Infrastructure as a Service A Cloud DC System Example
- Networking Issues in Cloud DC

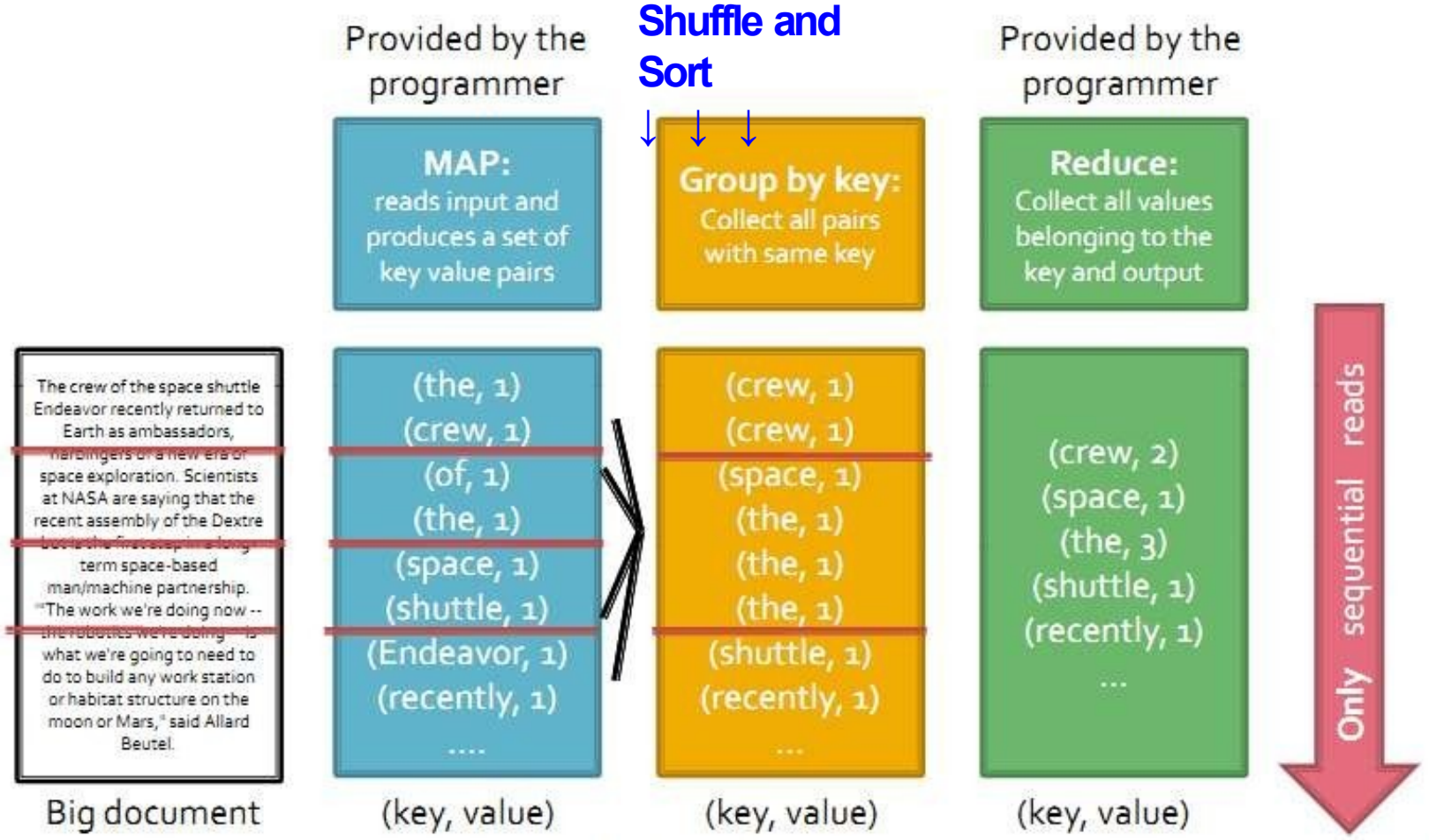
A DC-wide System

- Has software systems consisting of:
  - Distributed system, logical clocks, coordination and locks, remote procedural call…etc
  - Distributed file system
  - (We do not go deeper into above components)
  - Parallel computation: MapReduce, Hadoop
- Virtualized Infrastructure:
  - Computing: Virtual Machine / Hypervisor
  - Storage: Virtualized / distributed storage
  - Network: Network virtualization…the next step?

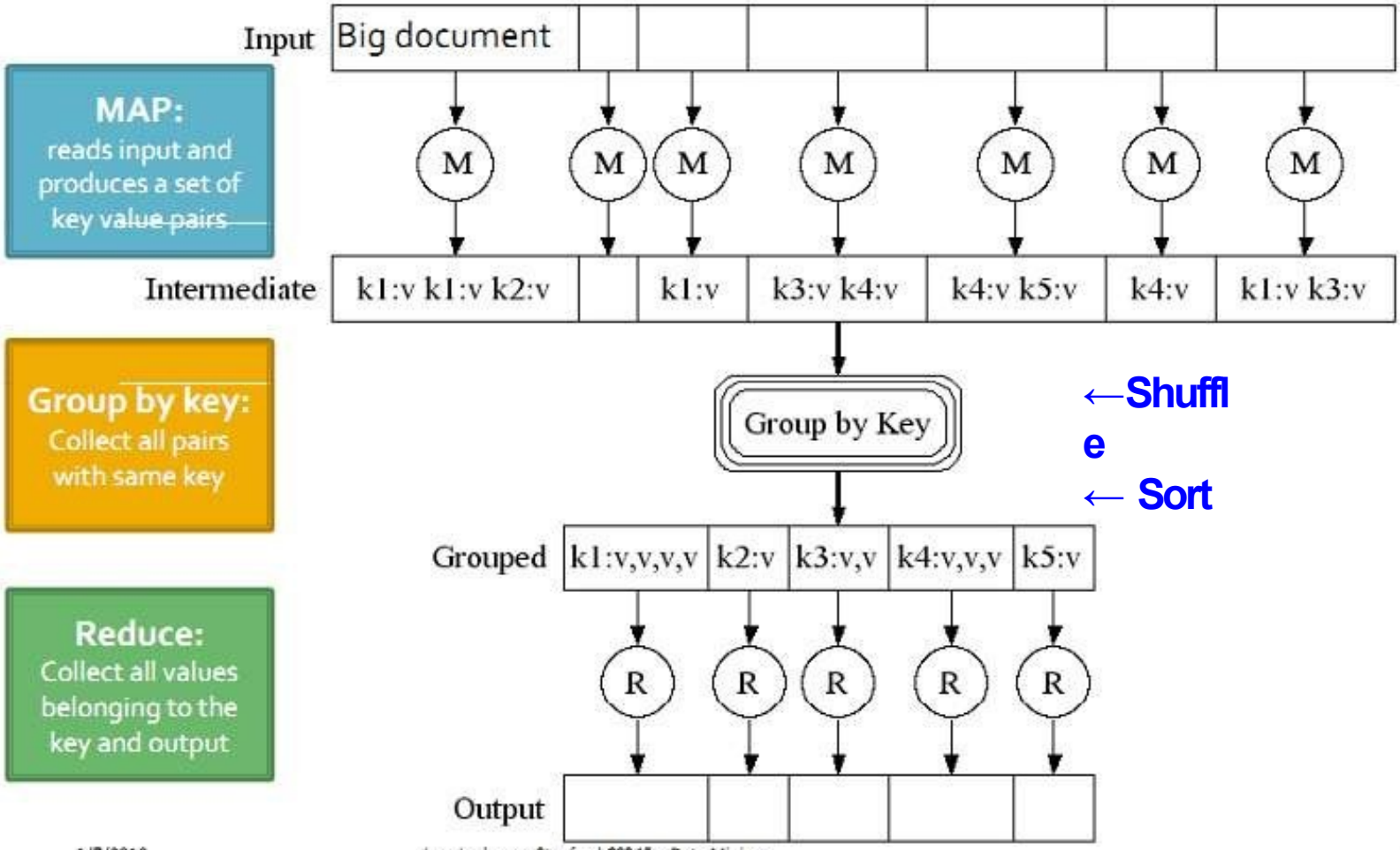
MapReduce

- 100 TB datasets
  - Scanning on 1 node – 23 days
  - On 1000 nodes – 33 minutes
- Single machine performance does not matter
  - Just add more… but HOW to use so many clusters ?
  - How to make distributed programming simple and elegant ?
- Sounds great, but what about MTBF?
  - MTBF = Mean Time Between Failures
  - 1 node – once per 3 years
  - 1000 nodes – 1 node per 1 day
- MapReduce refer to both:
  - Programming framework
  - Fault-tolerant runtime system

Map-Reduce: Word counting



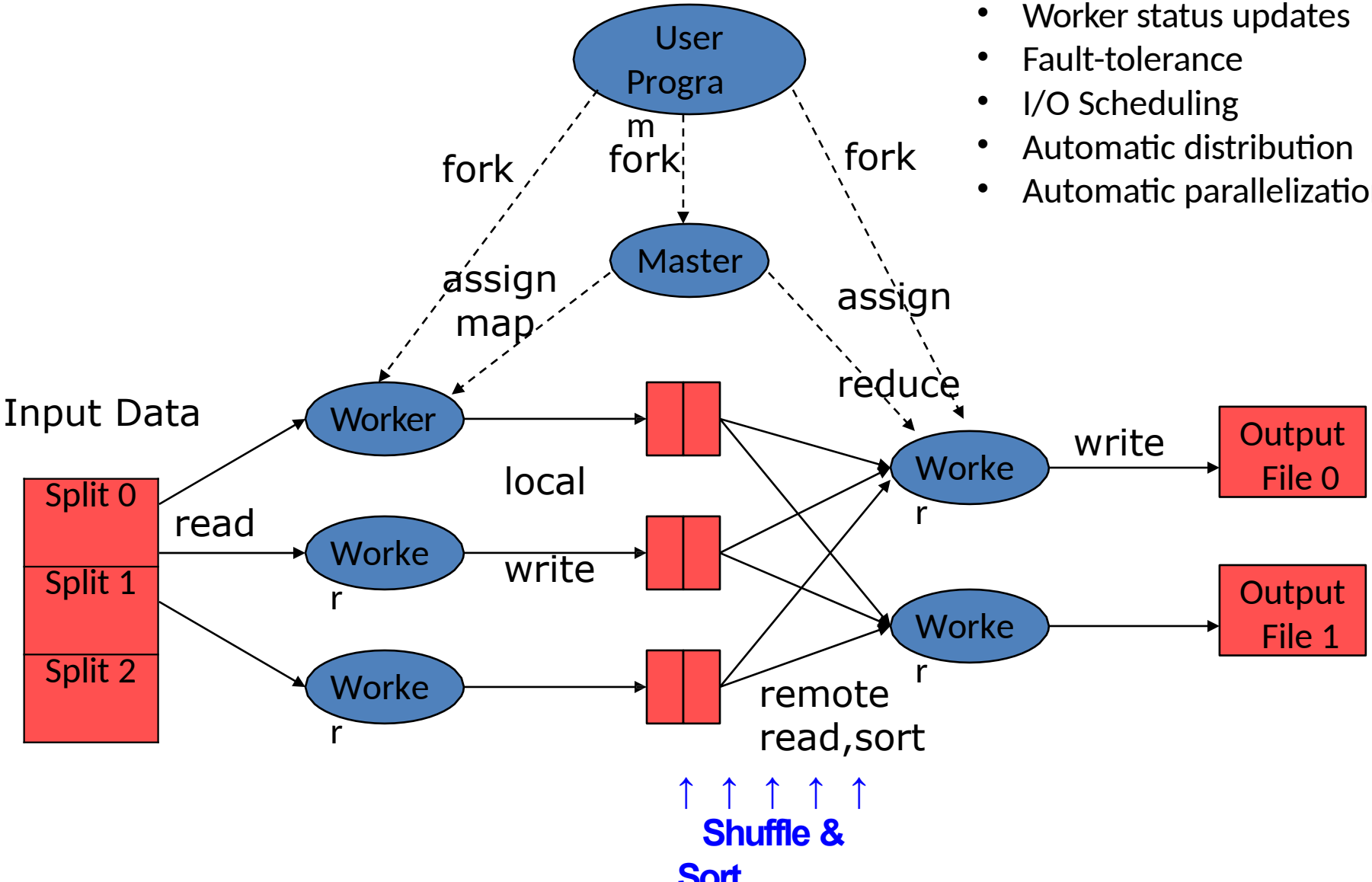
Map-Reduce: A diagram



Distributed Execution Overview

Master also deals with:

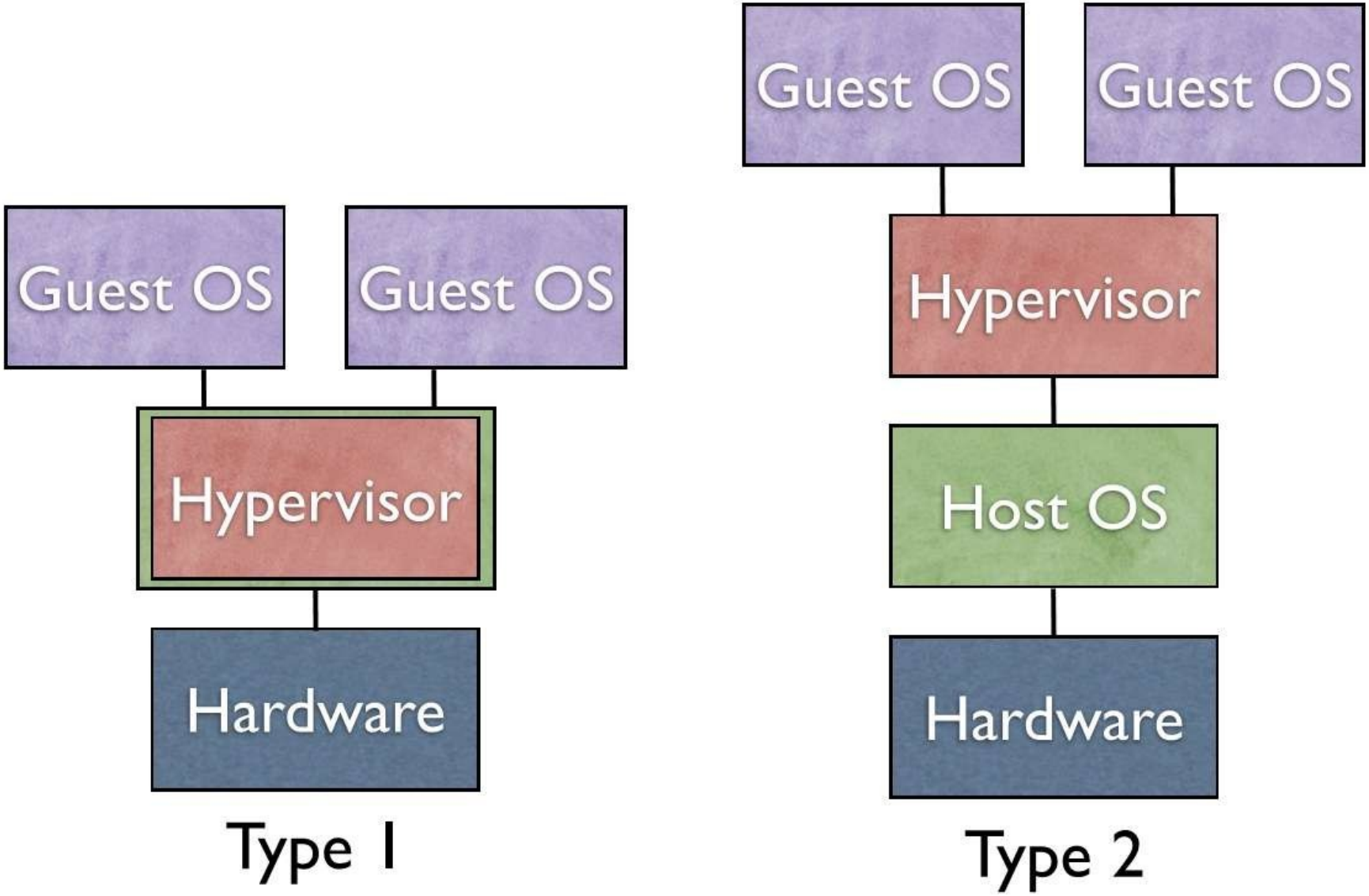
- Worker status updates
- Fault-tolerance
- I/O Scheduling
- Automatic distribution
- Automatic parallelization



VM and Hypervisor

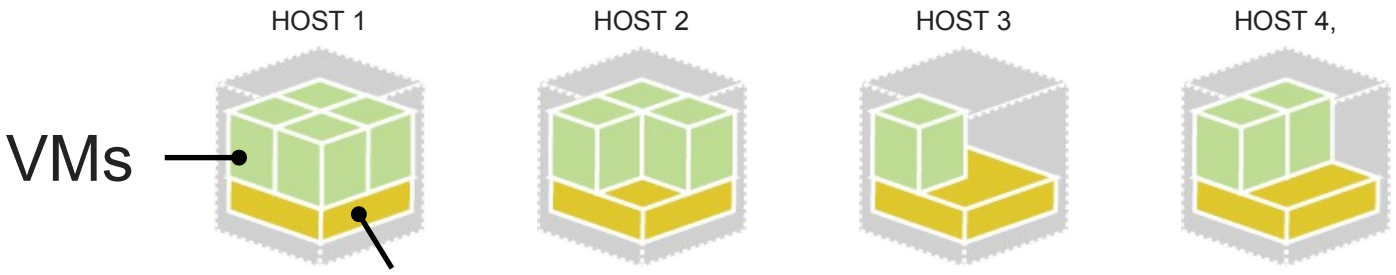
- Virtual Machine: A software package, sometimes using hardware acceleration, that allows an isolated guest operating system to run within a host operating system.
- Stateless: Once shut down, all HW states disappear.
- Hypervisor: A software platform that is responsible for creating, running, and destroying multiple virtual machines.
- Type I and Type II hypervisor

Type 1 vs. Type 2 Hypervisor



Concept of Virtualization

- Decoupling HW/SW by abstraction & layering
- Using, demanding,
  but not owning or configuring
- Resource pool: flexible to
  slice, resize, combine, and distribute
- A degree of automation by software



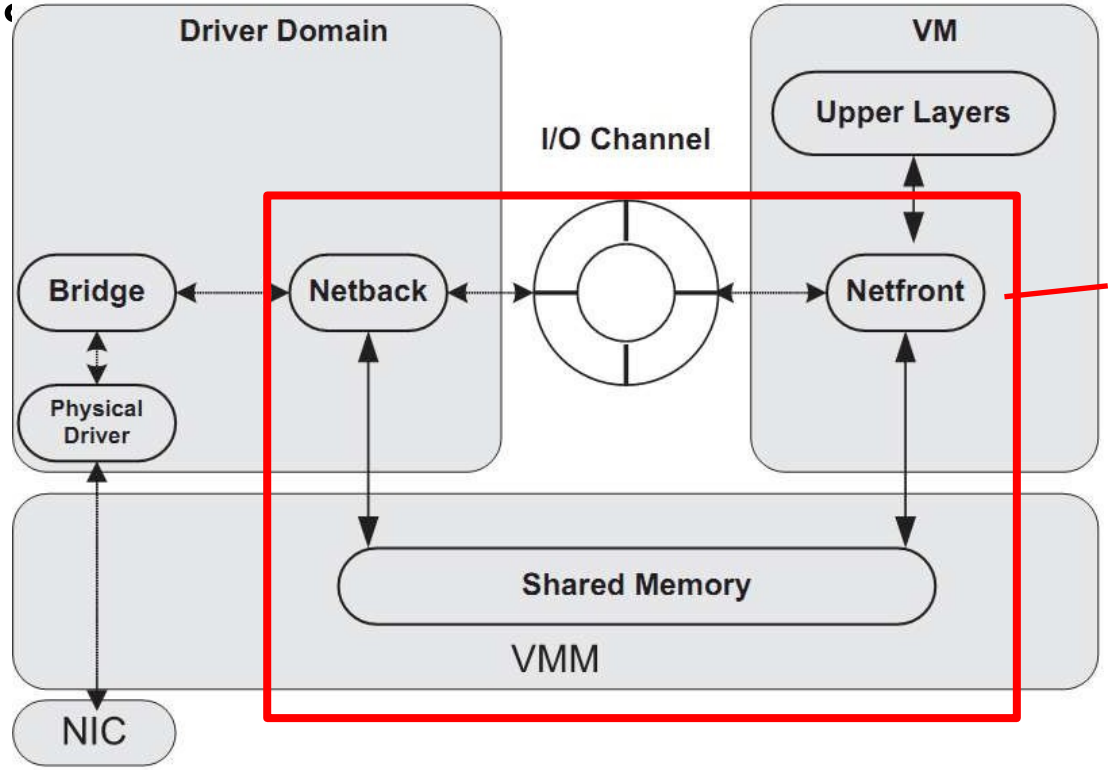
Hypervisor:
Turns 1 server into many “virtual machines” (instances or VMs)
(VMWare ESX, Citrix XEN Server, KVM, Etc.)

Concept of Virtualization

- **Hypervisor**: abstraction for HW/SW
- For SW: Abstraction and automation of physical resources
  - Pause, erase, create, and monitor
  - Charge services per usage units
- For HW: Generalized interaction with SW
  - Access control
  - Multiplex and demultiplex
- Ultimate hypervisor control from operator

I/O Virtualization Model

- Protect I/O access, multiplex / demultiplex traffic
- Deliver PKTs among VMs in shared memory
- Performance bottleneck: Overhead when communicating between driver domain and VMs
- VM scheduling and long queue→delay/throughput variance



Bottleneck:
CPU/RAM I/O lag
VM Scheduling
I/O Buffer Queue

Agenda

- Cloud Computing / Data Center Basic Background
- Enabling Technology
- **Infrastructure as a Service**
  A Cloud DC System Example
- Networking Issues in Cloud DC

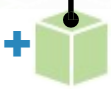
OpenStack Status

- OpenStack
  - Founded by NASA and Rackspace in 2010
  - Today 183 companies and 3386 people
  - Was only 125 and 1500 in fall, 2011.
  - Growing fast now, latest release Essex, Apr. 5th
- Aligned release cycle with Ubuntu, Apr. / Oct.
- Aim to be the “Linux” in cloud computing sys.
- Open-source v.s. Amazon and vmware
- Start-ups are happening around OpenStack
- Still lacks big use cases and implementation

Questi ons arise as the environment grows...

"VM s ab

- How do you make your apps cloud aware?
- How do you empower employees to self-service?
- Where should you provision new VMs?
- How do you keep track of it all?



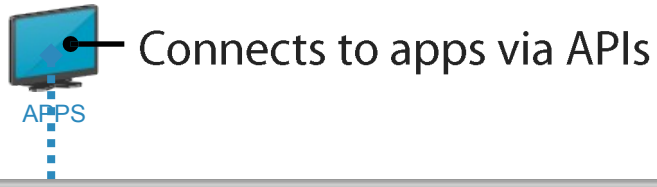
A Cloud Management Layer Is Missing

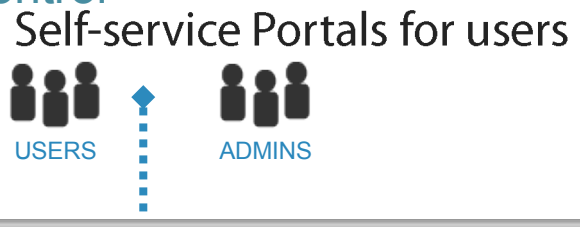
1. Virtualization
2. Cloud Data Center
3. Cloud Federation

Automation & Efficiency →

Solution: OpenStack, The Cloud Operating System

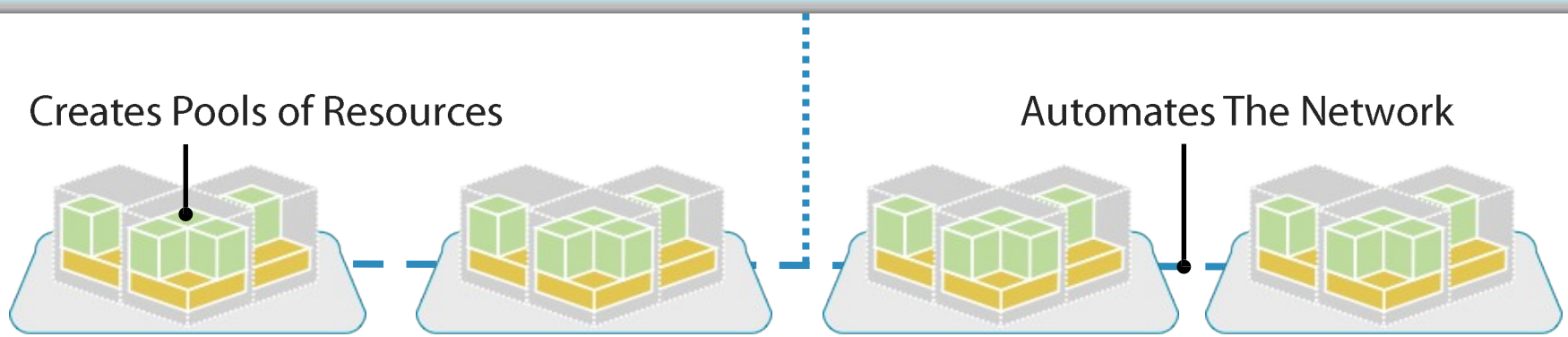
A new management layer that adds automation and control





openstack
CLOUD SOFTWARE

CLOUD OPERATING SYSTEM



1. Server Virtualization
2. Cloud Data Center
3. Cloud Federation

Automation & Efficiency →

A common platform is here.

OpenStack is open source software powering public and private clouds.

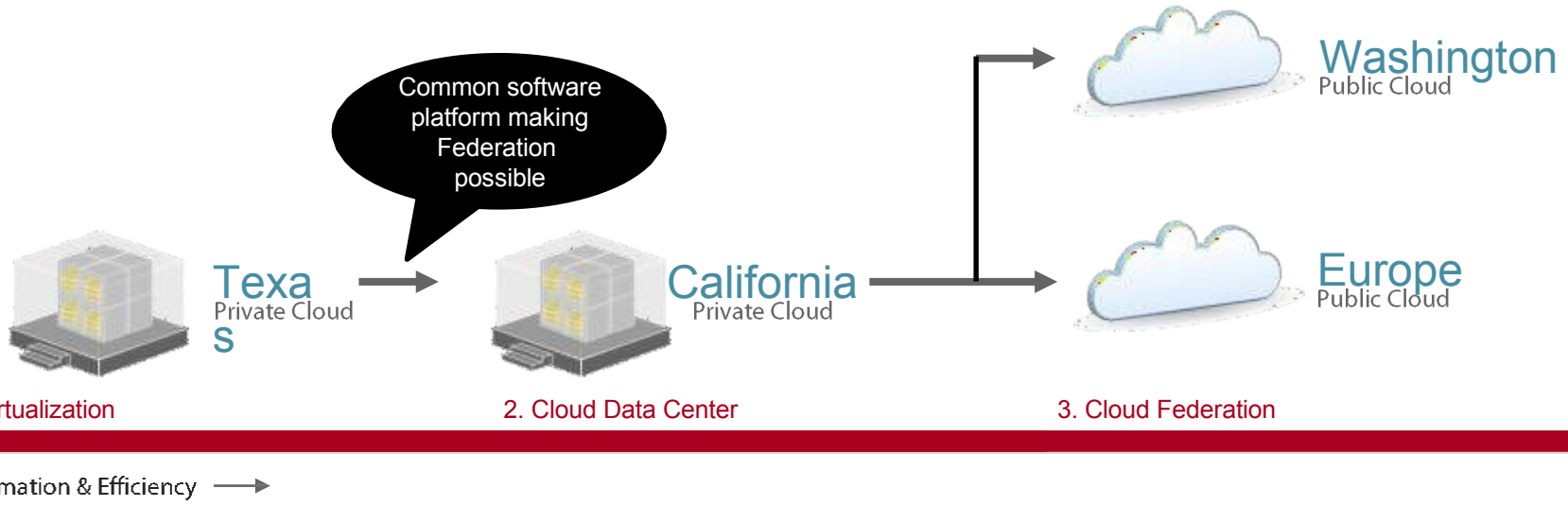
Private Cloud:
Run OpenStack software in your own corporate data centers

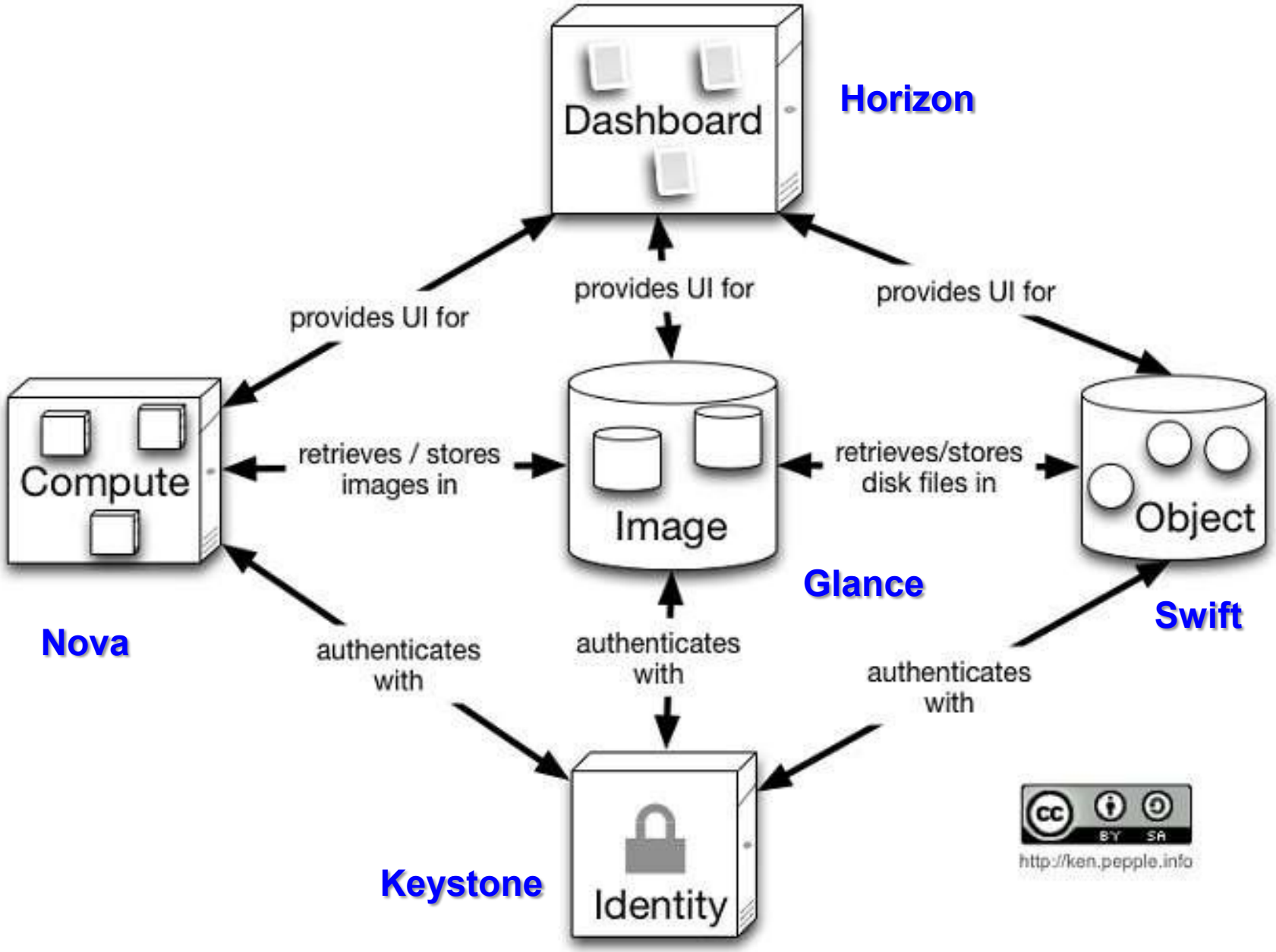
openstack™
CLOUD SOFTWARE

Public Cloud:
OpenStack powers some of the worlds largest public cloud deployments.

OpenStack enables cloud federation

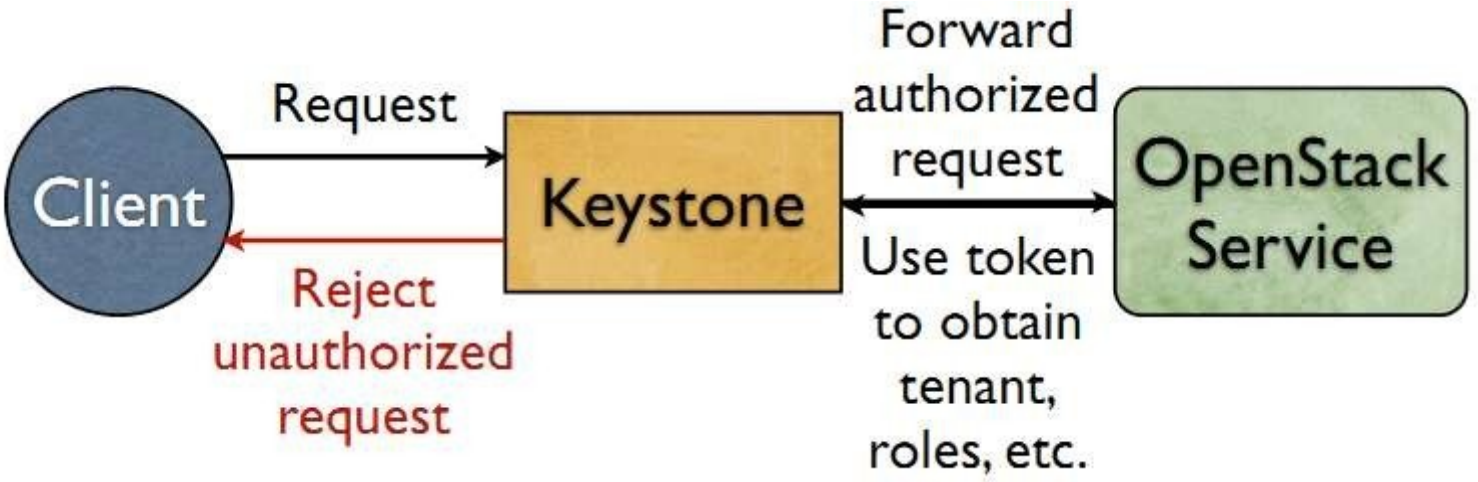
Connecting clouds to create global resource pools



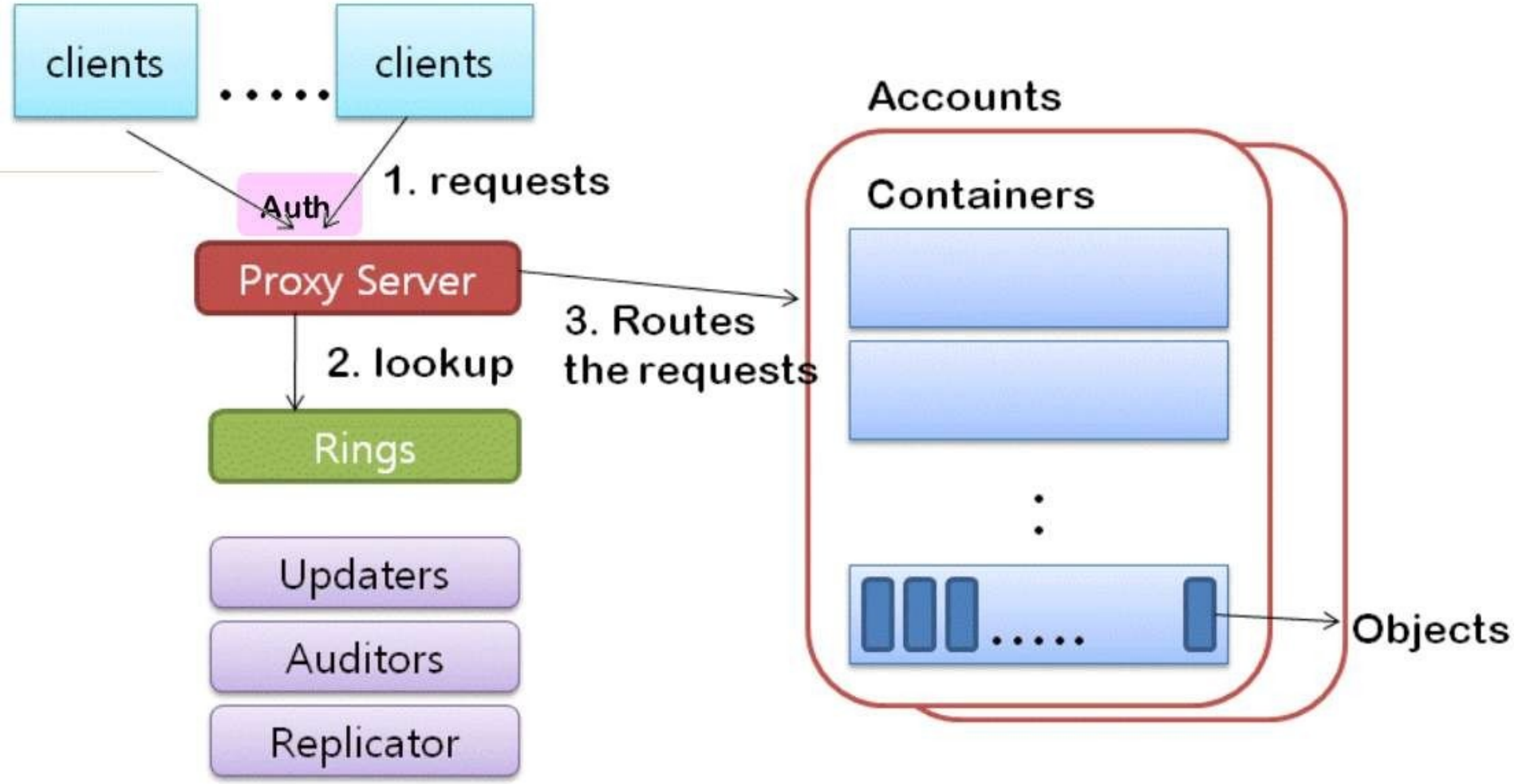

Dashboard
Horizon
provides UI for
provides UI for
provides UI for
Compute
retrieves / stores images in
Image
retrieves/stores disk files in
Object
Glance
Nova
Swift
authenticates with
authenticates with
authenticates with
Keystone
Identity
http://ken.pepple.info

Keystone Main Functions

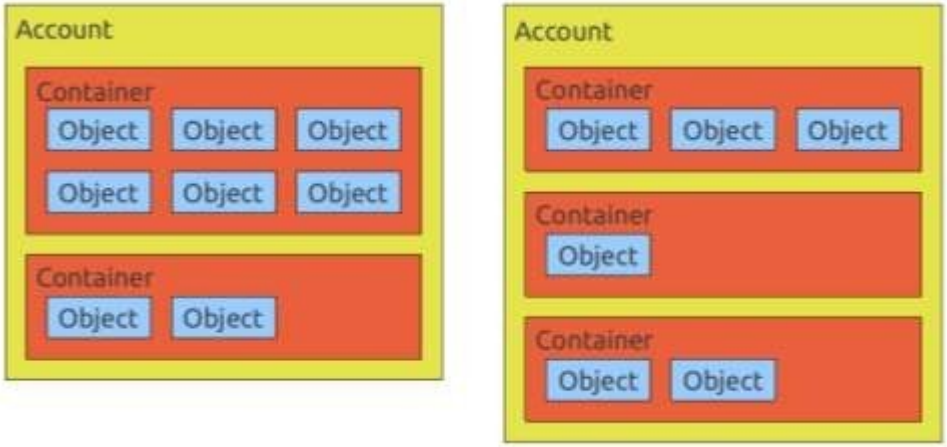
- Provides 4 primary services:
  - Identity: User information authentication
  - Token: After logged in, replace account-password
  - Service catalog: Service units registered
  - Policies: Enforces different user levels
- Can be backed by different databases.



Swift Main Components



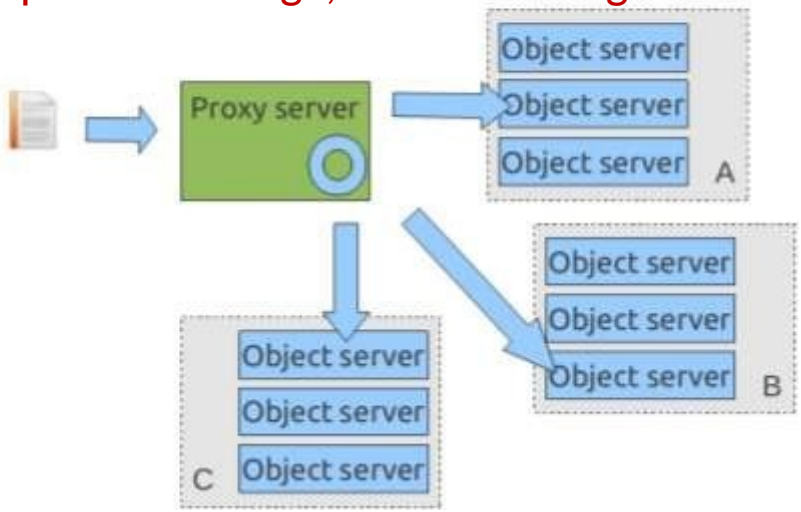
Objects, containers, accounts



↑ Logical view

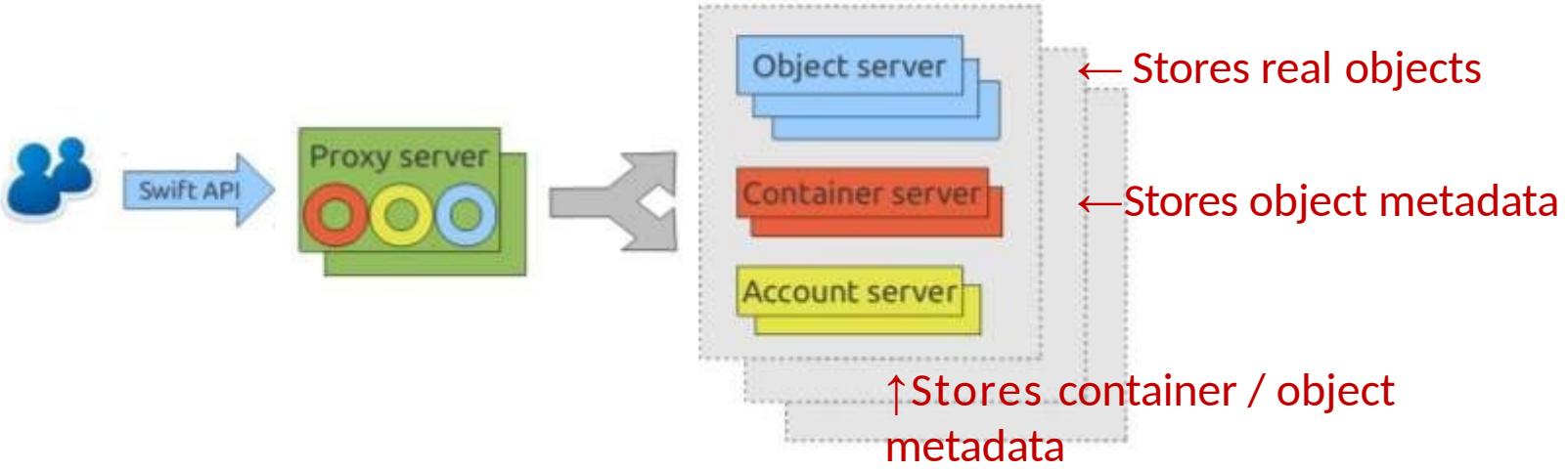
Swift request

Duplicated storage, load balancing



Swift main components

↓Physical arrangement



← Stores real objects

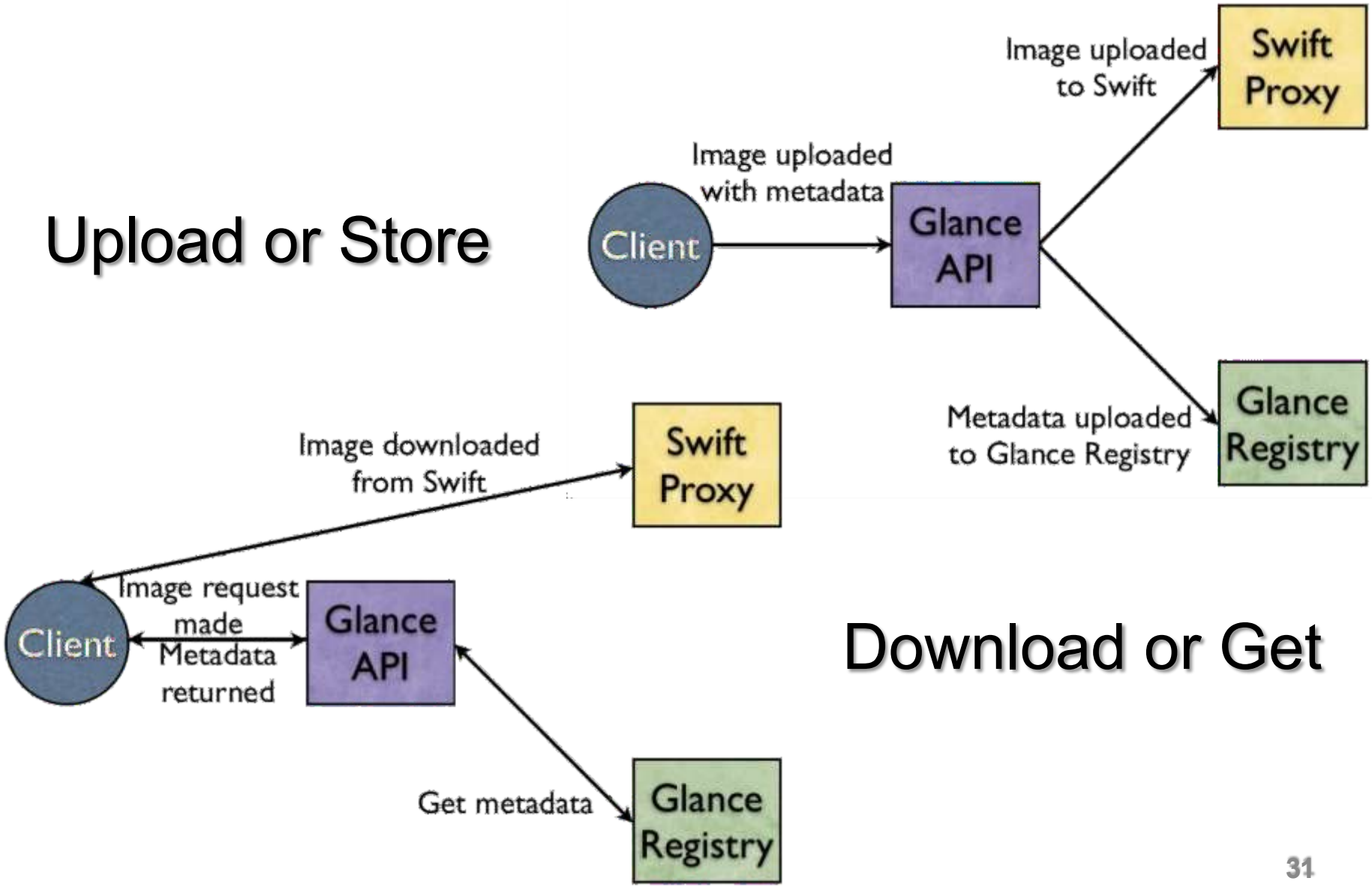
←Stores object metadata

↑Stores container / object metadata

Glance

- Image storage and indexing.
- Keeps a database of metadata associated with an image, discover, register, and retrieve.
- Built on top of Swift, images store in Swift
- Two servers:
  - Glance-api: public interface for uploading and managing images.
  - Glance-registry: private interface to metadata database
- Support multiple image formats

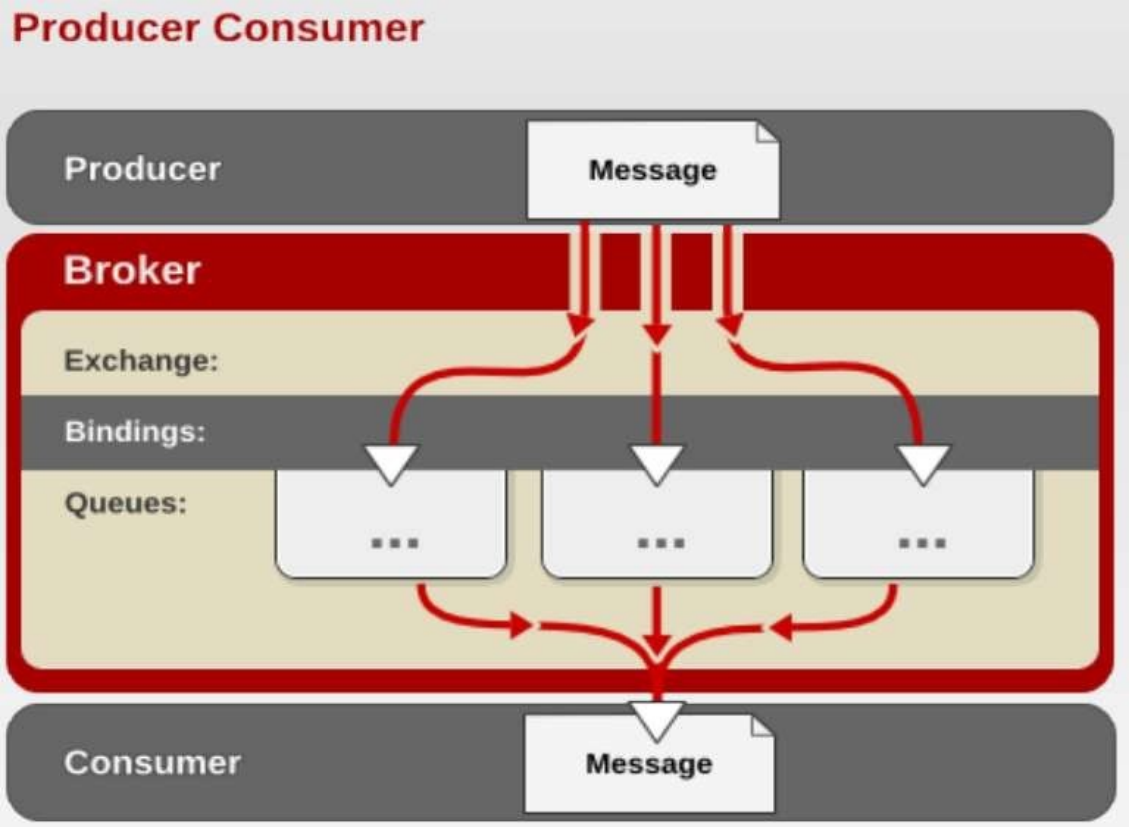
Glance Process



Nova

- Major components:
  - API: public facing interface
  - Message Queue: Broker to handle interactions between services, currently based on RabbitMQ
  - Scheduler: coordinates all services, determines placement of new resources requested
  - Compute Worker: hosts VMs, controls hypervisor and VMs when receives cmds on Msg Queue
  - Volume: manages permanent storage

Messaging (RabbitMQ)

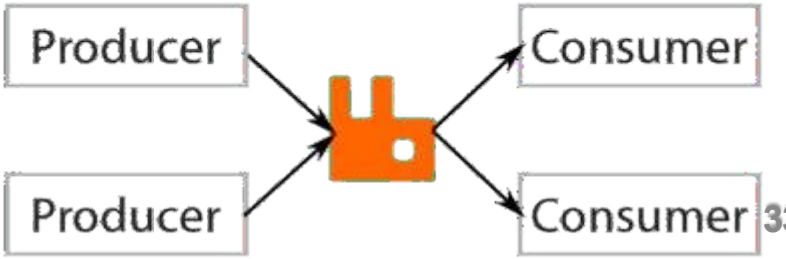


- A vhost contains
  - exchanges, queues and bindings
  - authentication & permissions
- Users may be registered with multiple vhosts

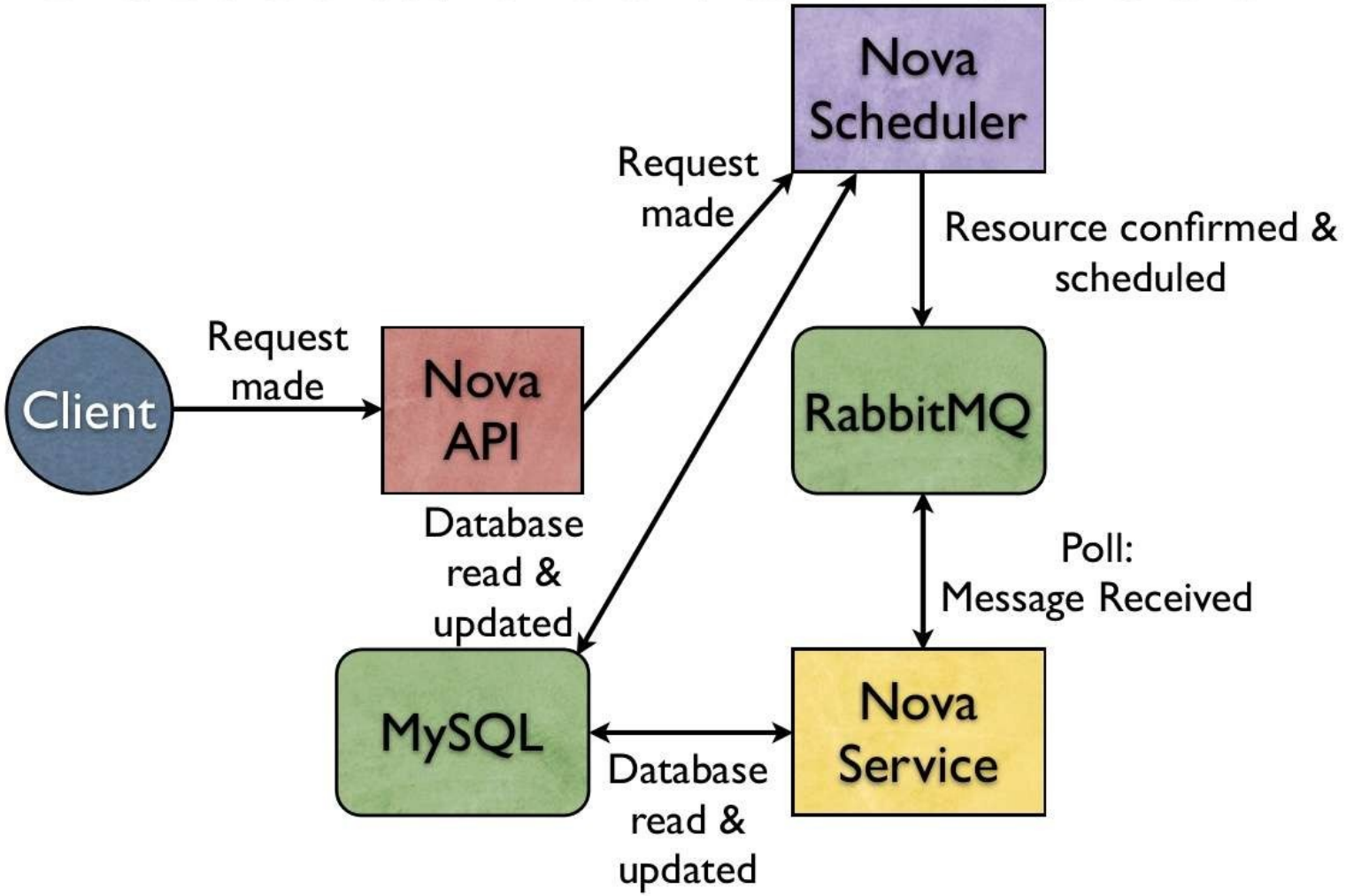
Decoupling

E.g. website passing orders to a credit-card charging engine

Work aggregation, distribution and Decoupling

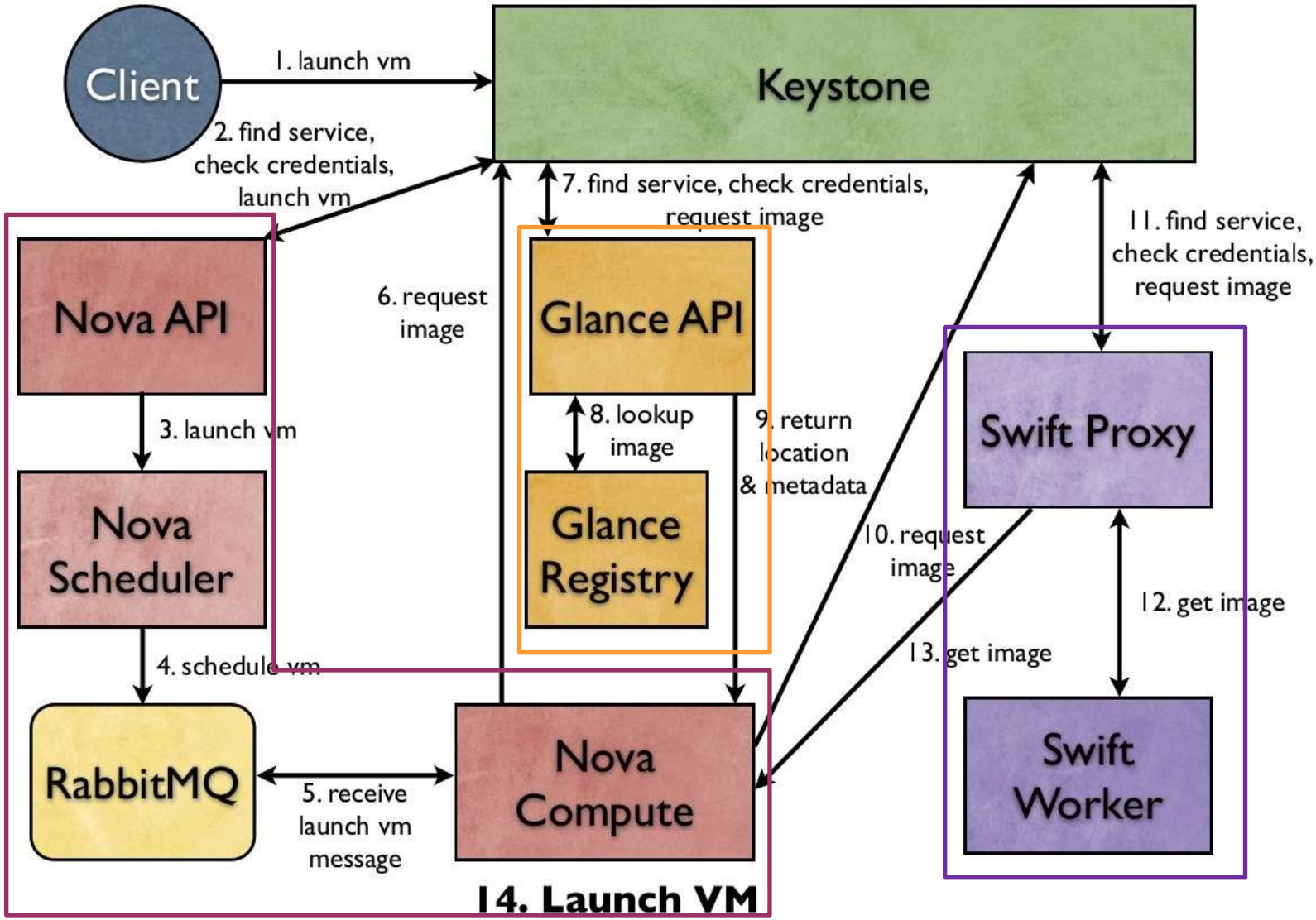


General Nova Process

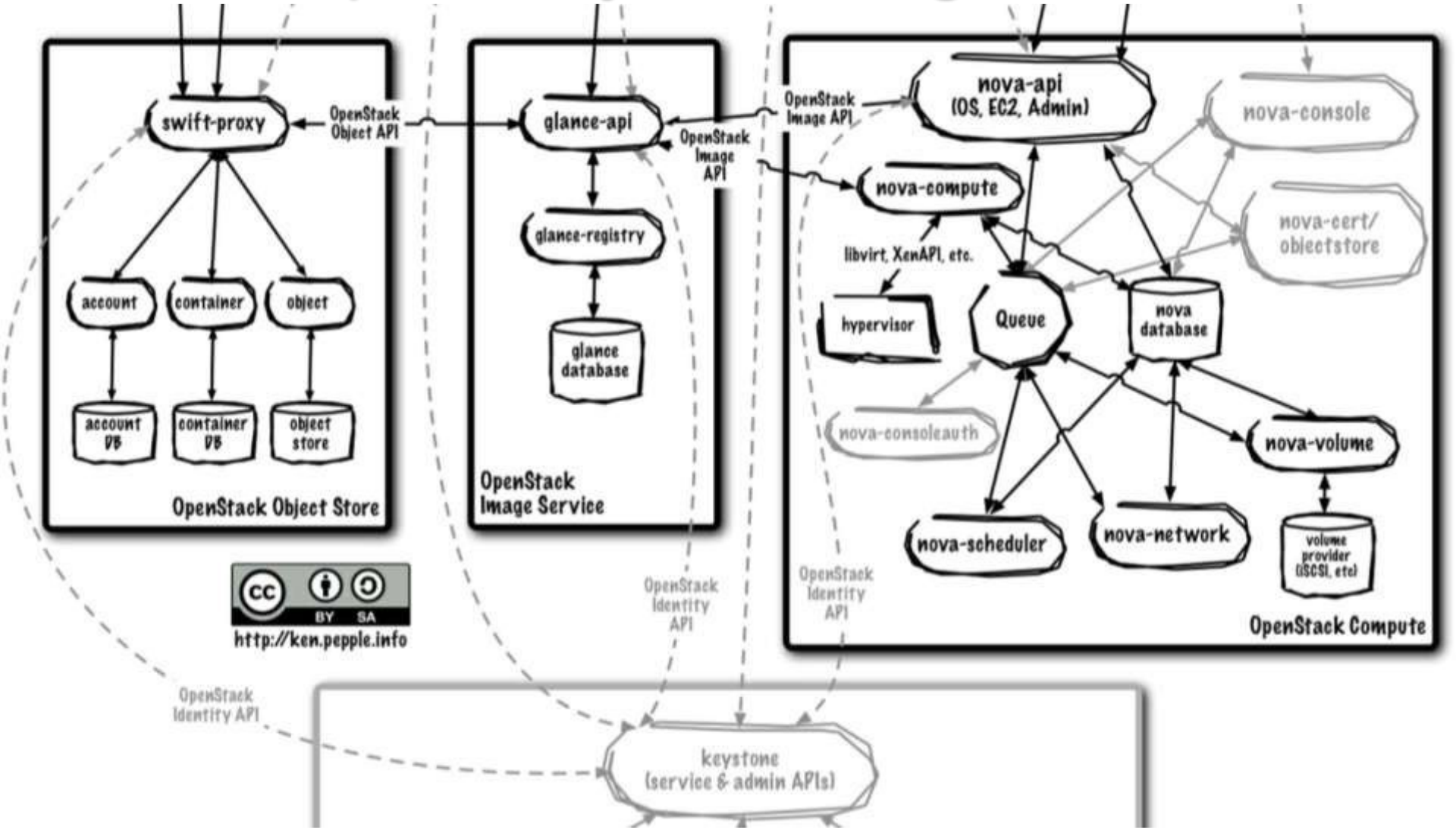


Connections to RabbitMQ and MySQL happen regularly to monitor status of system.

Launching a VM



Complete System Logical View



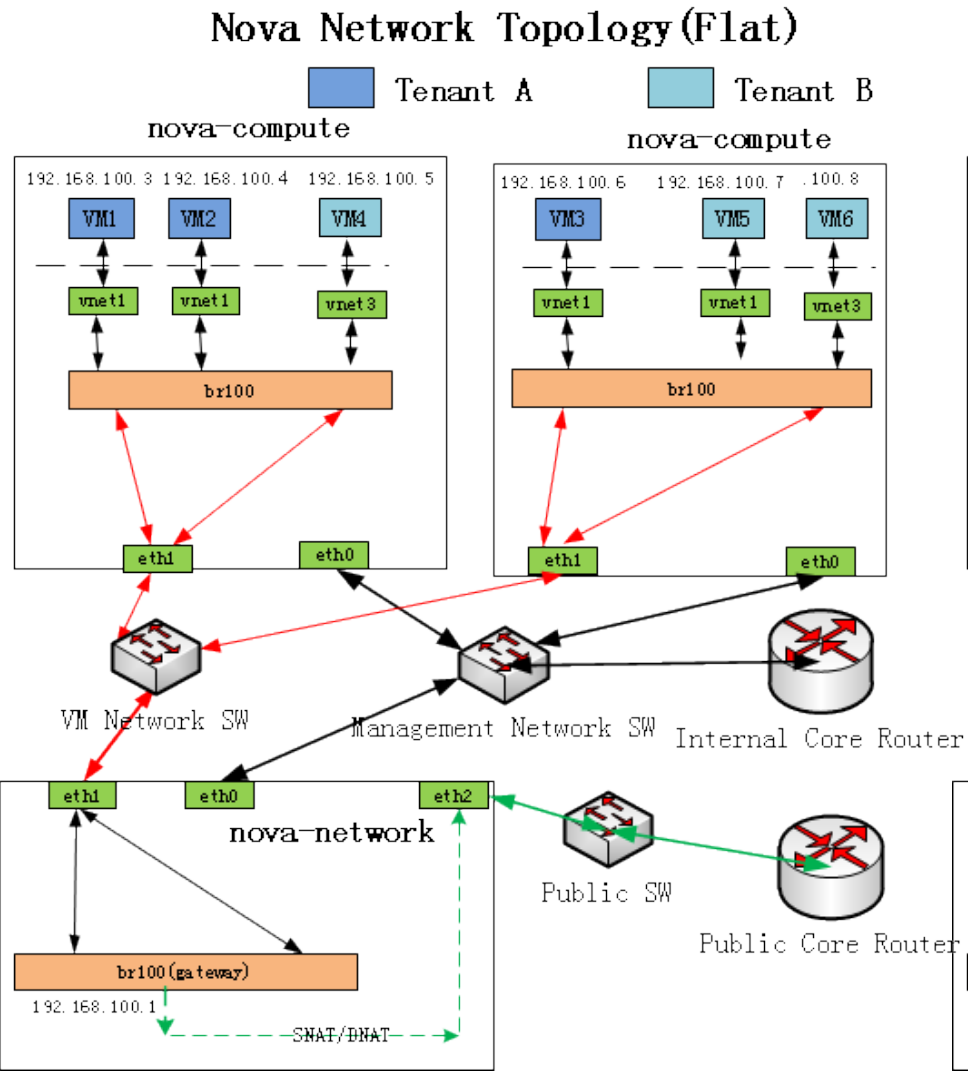
Agenda

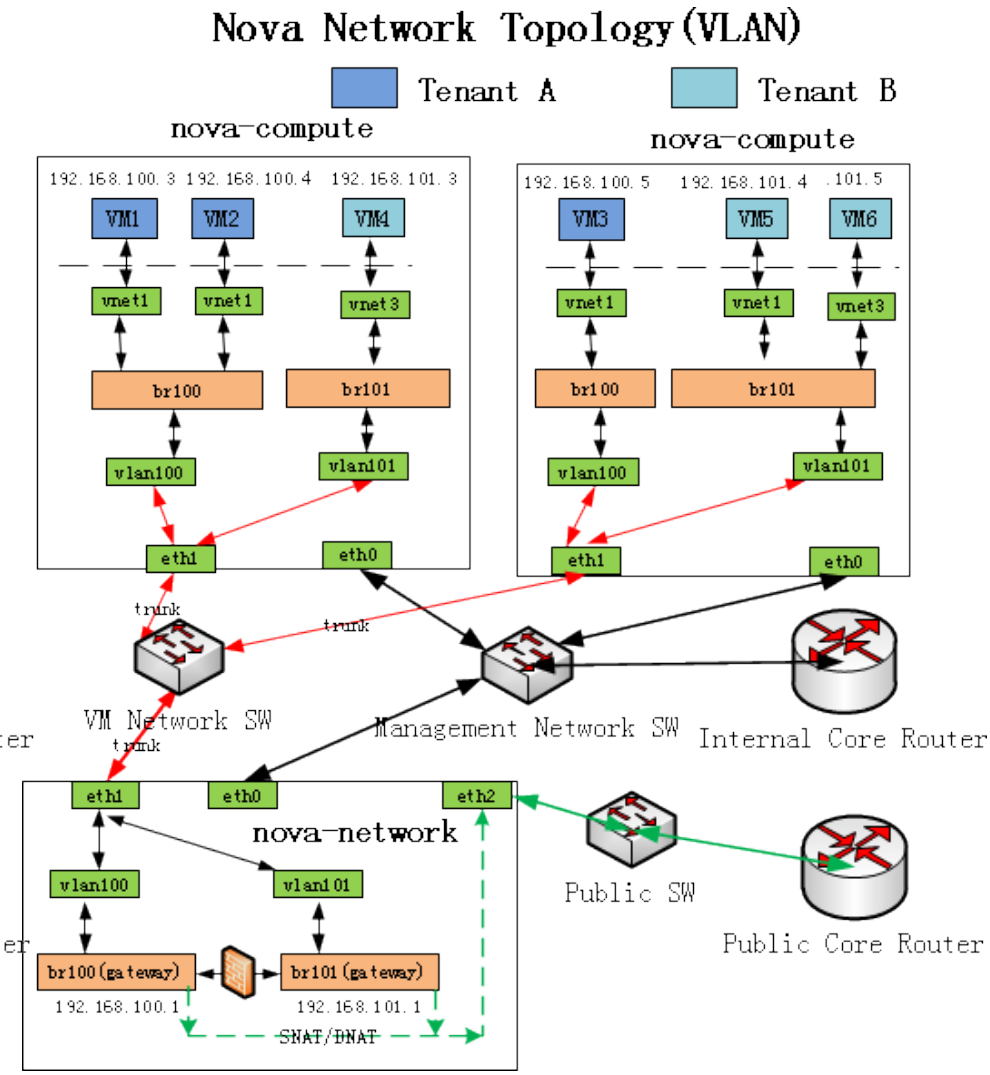
- Cloud Computing / Data Center Basic Background
- Enabling Technology
- Infrastructure as a Service A Cloud DC System Example
- **Networking Issues in Cloud DC**

Primitive OpenStack Network

- Each VM network owned by one network host
  - Simply a Linux running Nova-network daemon
- Nova Network node is the only gateway
- Flat Network Manager:
  - Linux networking bridge forms a subnet
  - All instances attached same bridge
  - Manually configure server, controller, and IP
- Flat DHCP Network Manager:
  - Add DHCP server along same bridge
- Only gateway, per-cluster, fragmentation

OpenStack Network



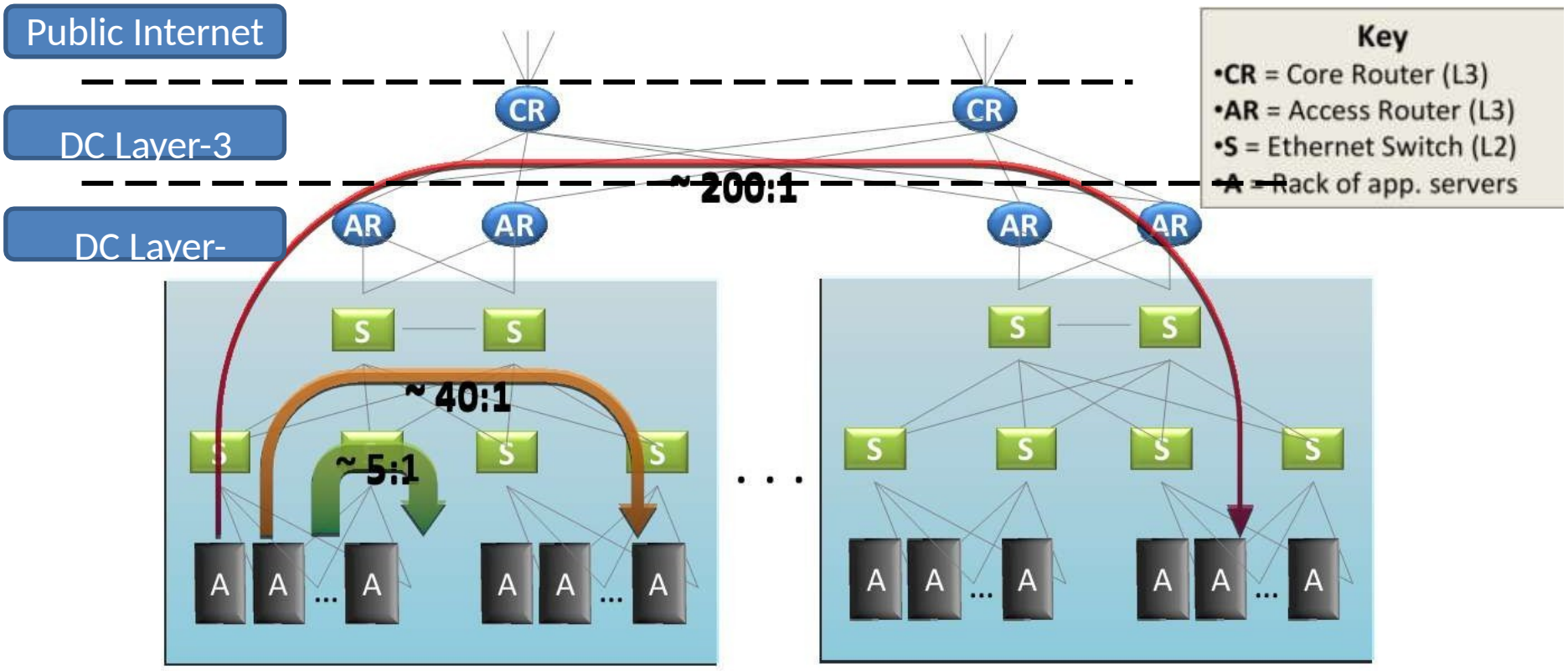


Linux server running Nova-network daemon.

The only gateway of all NICs bridged into the

VMs bridged in to a raw Ethernet device

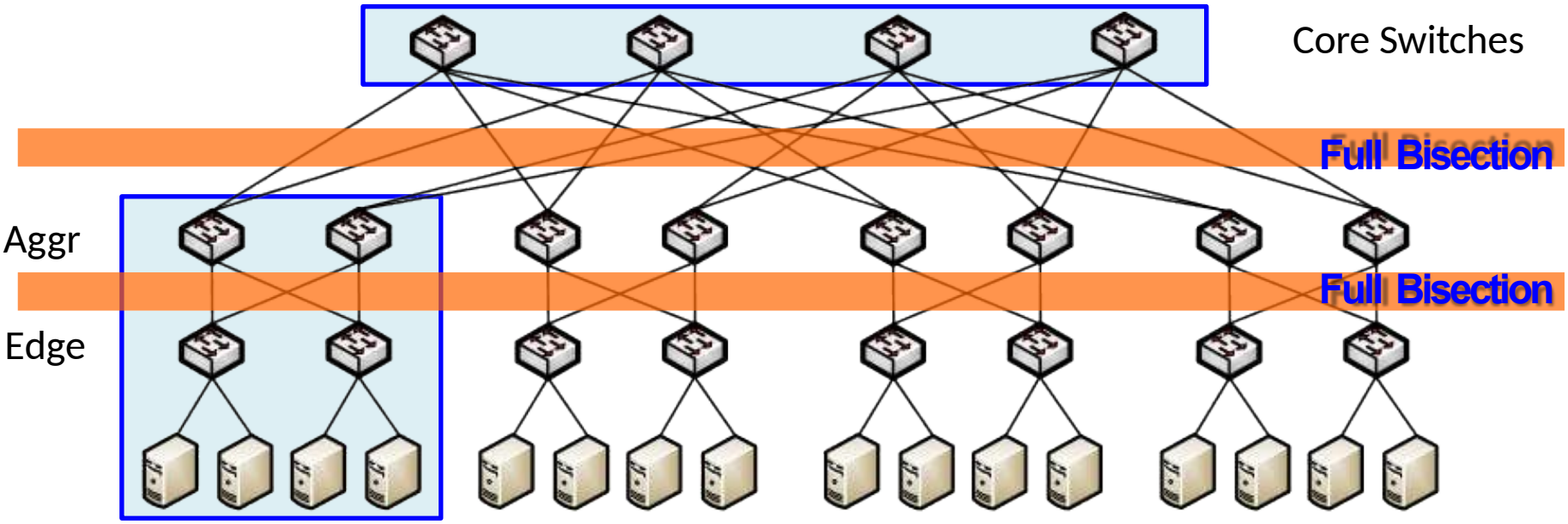
Conventional DCN Topology



- Oversubscription
- Fragmentation of resources:
  Network limits cross-DC communication
- Hinders applications' scalability
- Only reachability isolation
  Dependent performance bottleneck

- Scale-up proprietary design – expensive
- Inflexible addressing, static routing
- Inflexible network configuration
  Protocol baked / embedded on chips

A New DCN Topology

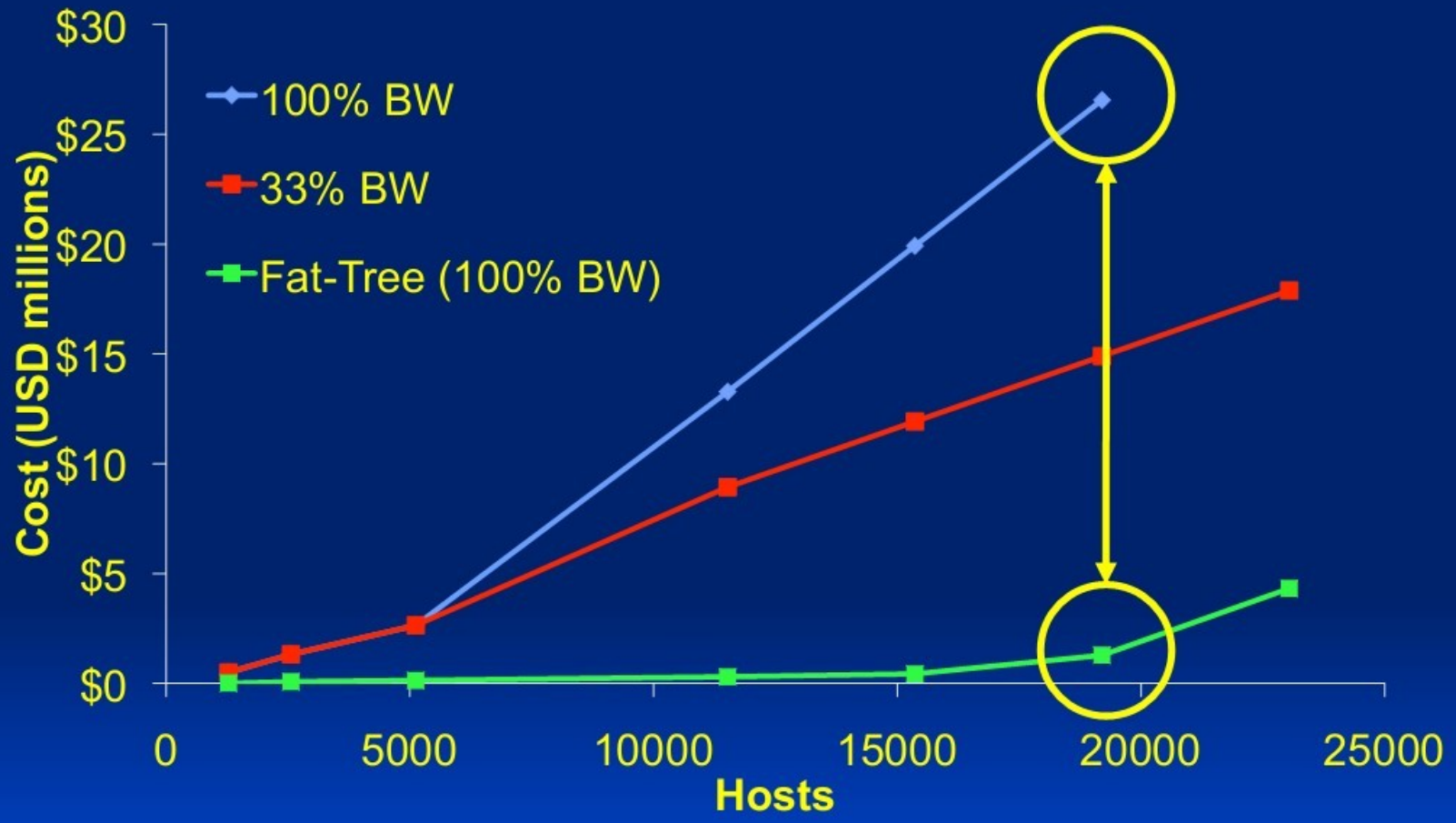


k = 4

- k pod with ($k^2/4$ hosts, k switches)per pod
- $(k/2)^2$ core switches, $(k/2)^2$ paths for S-D
- $(5k^2/4)$ k-port switches, $k^3/4$ hosts
- 48-port: 27,648 hosts, 2,880 switches
- Full bisection BW at each level
- **Modular scale-out** cheap design

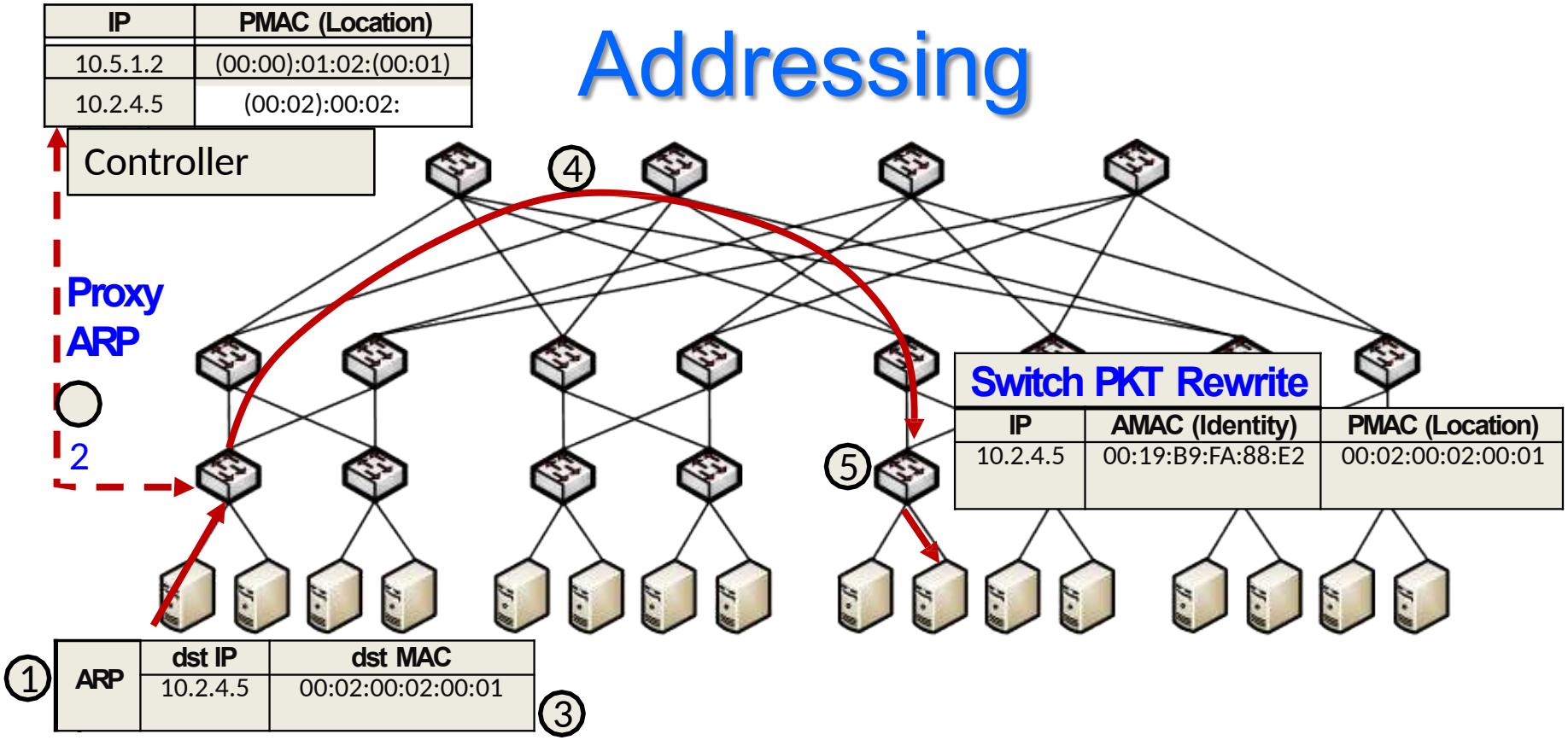
- Cabling explosion, copper trans. range
- Existing addressing/routing/forwarding do not work well on fat-tree / clos
- Scalability issue with millions of end hosts
- Configuration of millions of parts

Cost of Data Center Networks



- Factor of 10+ price difference between traditional approach and proposed architecture

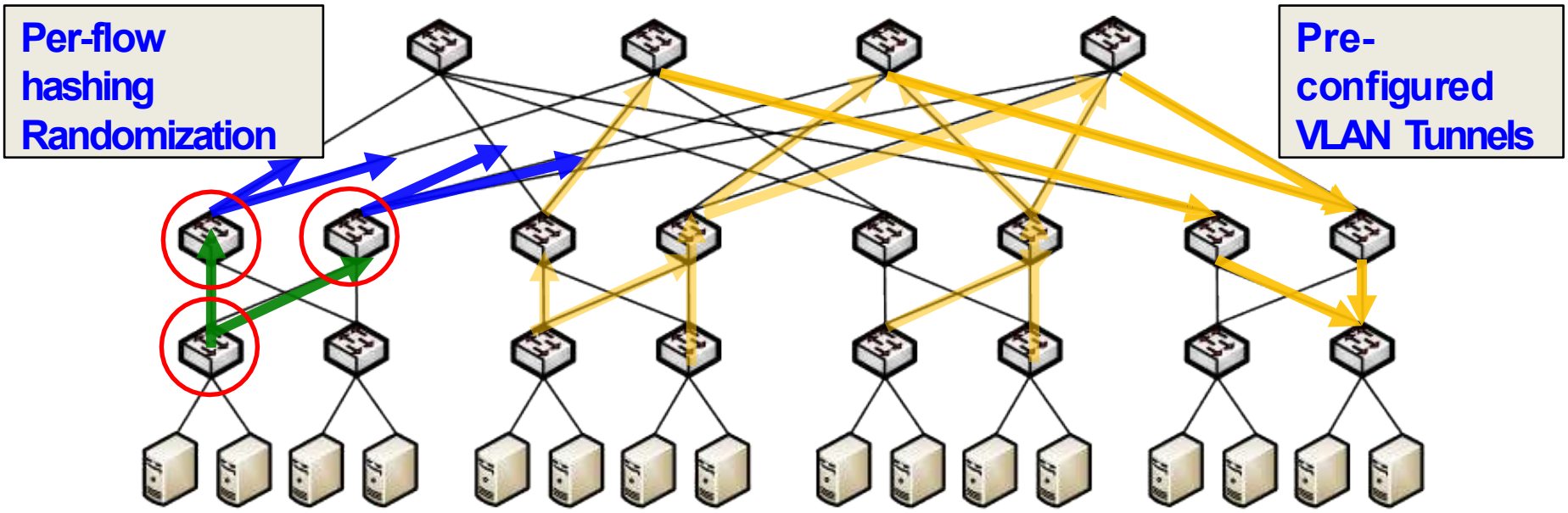
Addressing



- Switches: 32~64 K flow entries, 640 KB
- Assume 10,000k VMs on 500k servers
- **Identity-based**: 10,000k flat entries,
  100 MB huge, flexible, per-VM/APP
  VM migration, continuous connection
- **Location-based**: 1k hierarchical entries
  10 KB easy storage, fixed, per-server
  Easy forwarding, no extra reconfiguration

- AMAC: Identity, maintained at switches
- PMAC: (pod,position,port,vmid)
  IP→ PMAC, mapped at controller
- Routing: Static VLAN or ECMP-hashing
  (To be presented later)
- Consistency / efficiency / fault-tolerant?
  Solve by (controller, SW, host) diff. roles
- Implemented: server- / switch- centric

Load Balancing / Multipathing



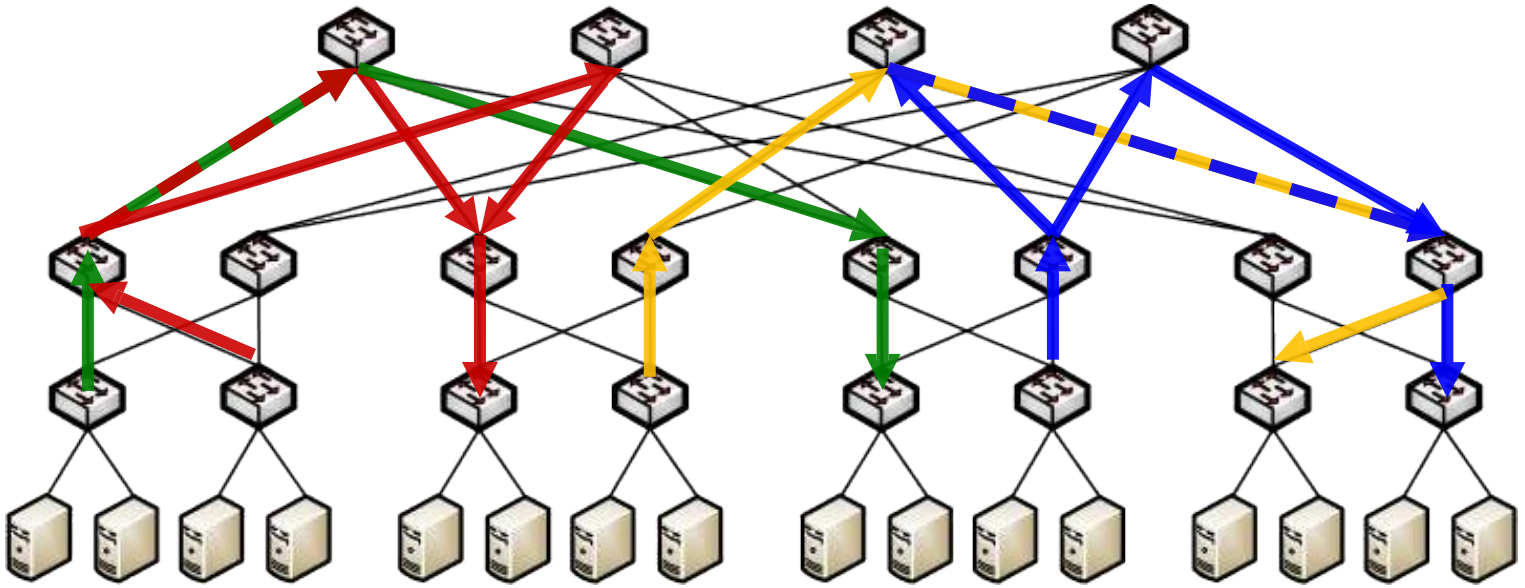
End hosts "transparent": Sends traffic to networks as usual, without seeing detail
OpenFlow: Controller talks to (HW/SW switches, kernel agents), manipulates entries

- [illegible]
- Network delay / PKT loss → Performance ↓
- Still, only commodity hardware
- Aggregated individual small demand
  → Traffic extremely volatile / unpredictable
- Traffic matrix: dynamic, evolving, not steady
- User: Don't know infrastructure, topology
- Operator: Don't know application, traffic

[illegible] capacity!

- VLAN: multiple preconfigured tunnels
  →Topological dependent
- Multipath-TCP: modified transport mech.
  →Distributes and shifts load among paths
- ECMP/VLB: Randomization, header hash
  →Only randomized upward paths
  →Only for symmetric traffic

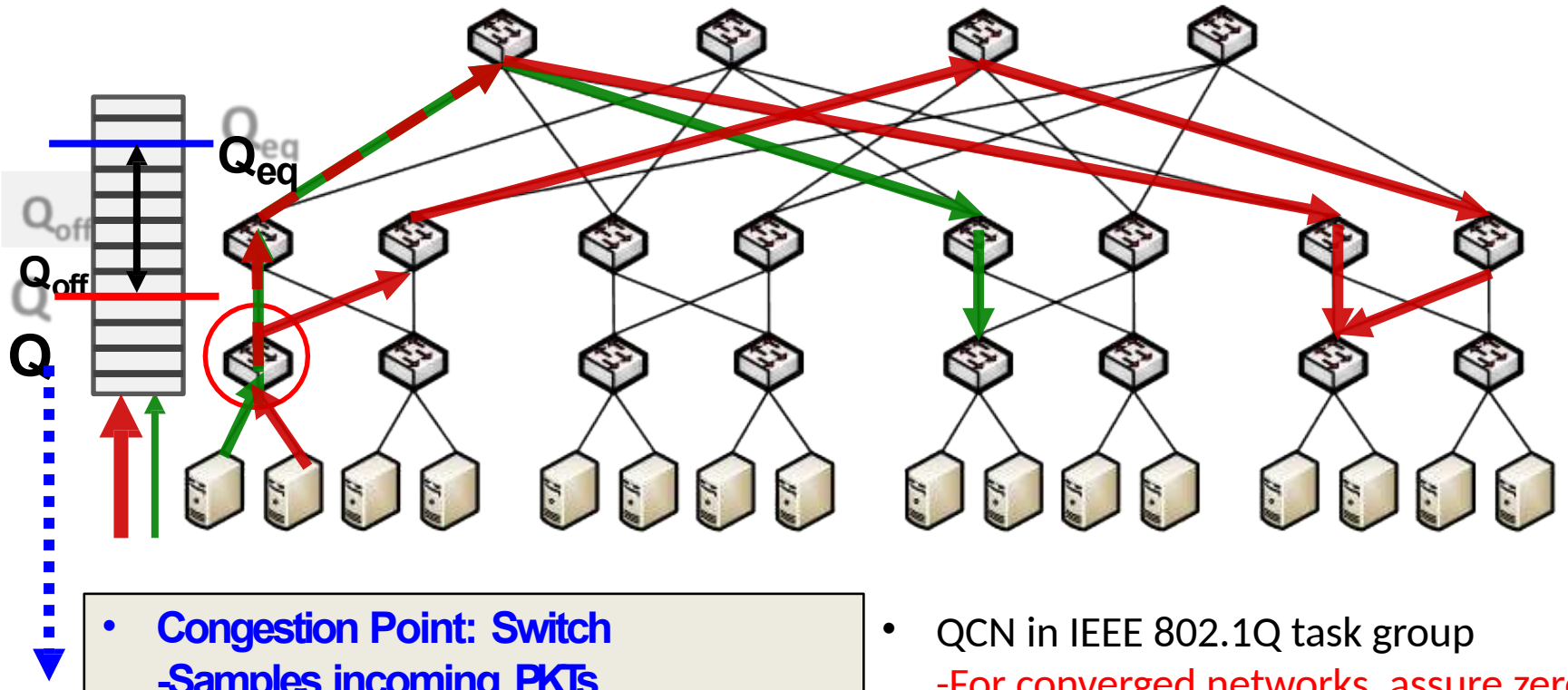
Flow Scheduling



- ECMP-hashing → per-flow static path
- Long-live elephant flows may collide
- Some links full, others under-utilized

- Flow-to-core mappings, re-allocat flows
- What time granularity? Fast enough?
- Controller computation? Scalable?

Reactive Reroute

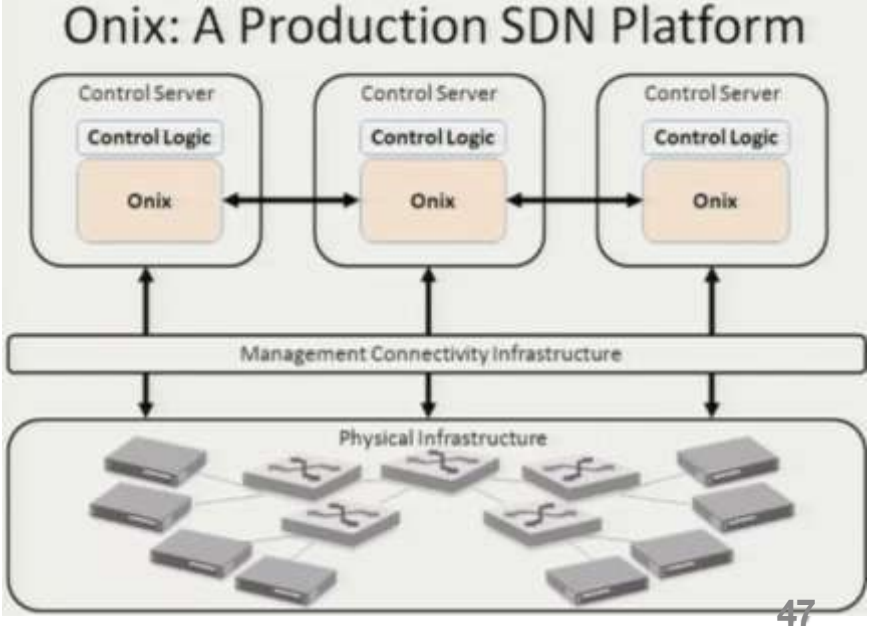


- **Congestion Point: Switch**
  - **-Samples incoming PKTs**
  - **-Monitor and maintain queue level**
  - **-Send feedback msg to src**
  - **-Feedback msg according to Q-len**
  - **-Choose to re-hash elephant flows**
- **Reaction Point: Source Rate Limiter**
  - **-Decrease rate according to feedback**
  - **-Increase rate by counter / timer**

- QCN in IEEE 802.1Q task group
  - -For converged networks, assure zero drop
  - -Like TCP AIMD but on L2, w/ diff purpose
  - -CP directly reacts, not end-to-end
  - -Can be utilized for reactive reroute
- May differentiate FB msg
  - -Decrease more for lower classes (QoS)
  - -Decrease more for larger flows (Fairness)
- Large flows are suppressed →High delay

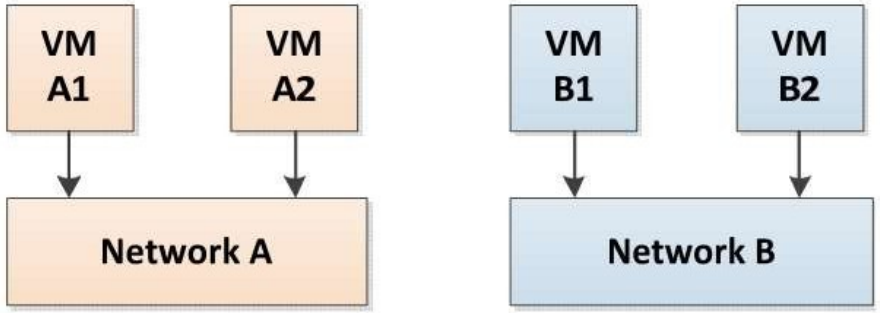
Controller

- DCN relies on controller for many functions:
  - Address mapping / mgmt / registration / reuse
  - Traffic load scheduling / balancing
  - Route computation, switch entries configuration
  - Logical network view ↔ physical construction
- An example: Onix
  - Distributed system
  - Maintain, exchange & distribute net states
    - Hard static: SQL DB
    - Soft dynamic: DHT
  - Asynchronous but eventually consistent

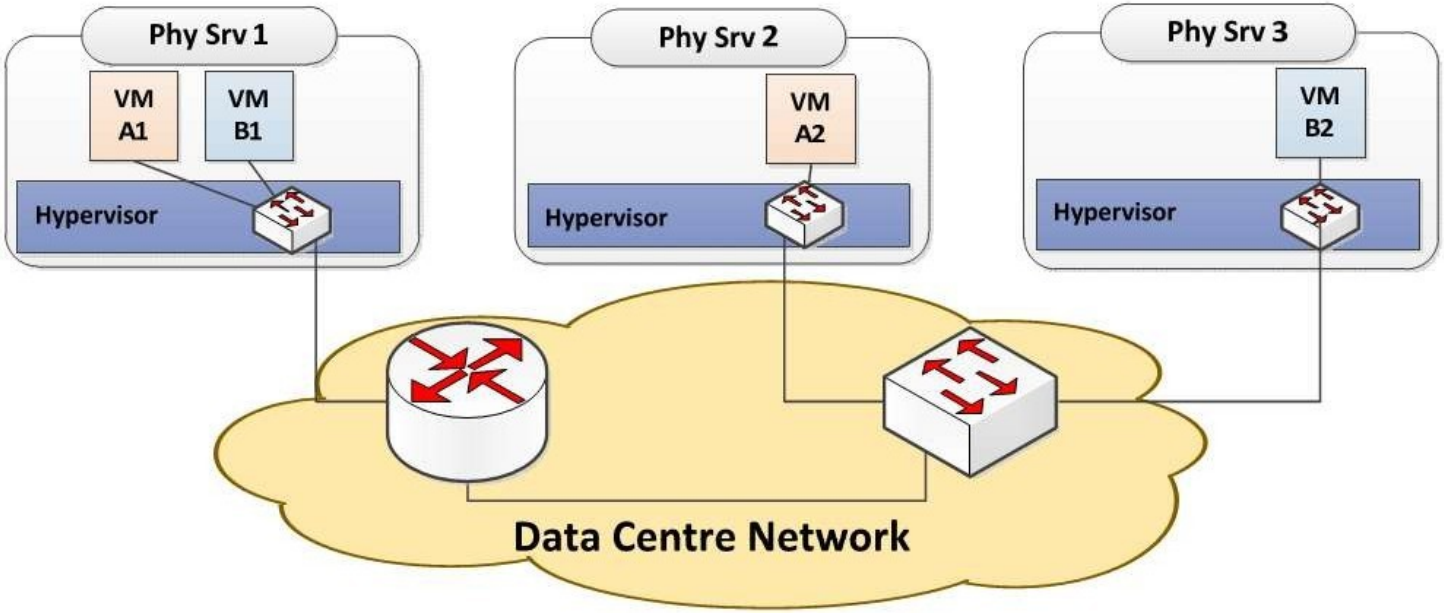


Tenant View vs Provider View

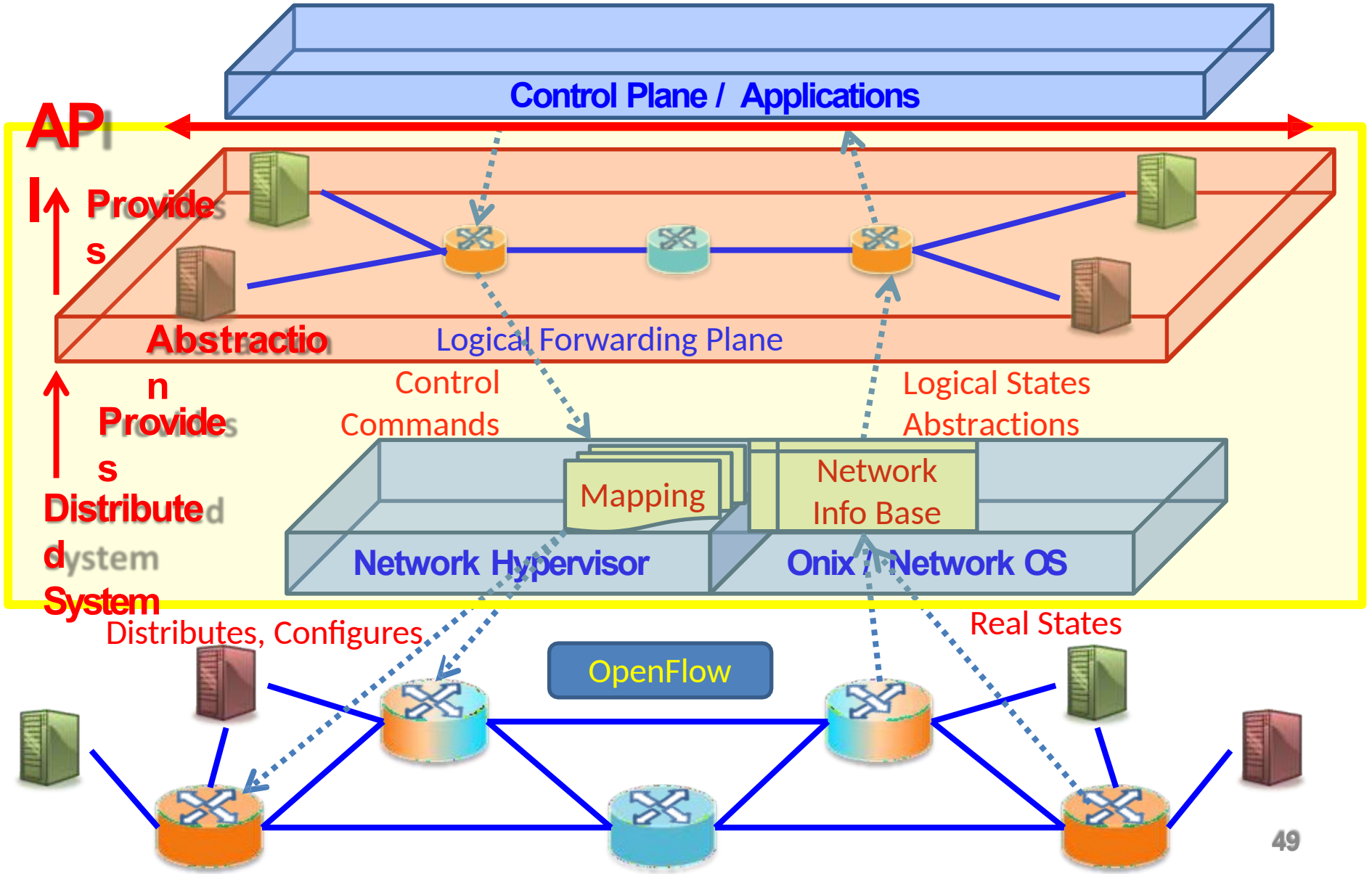
Tenant View



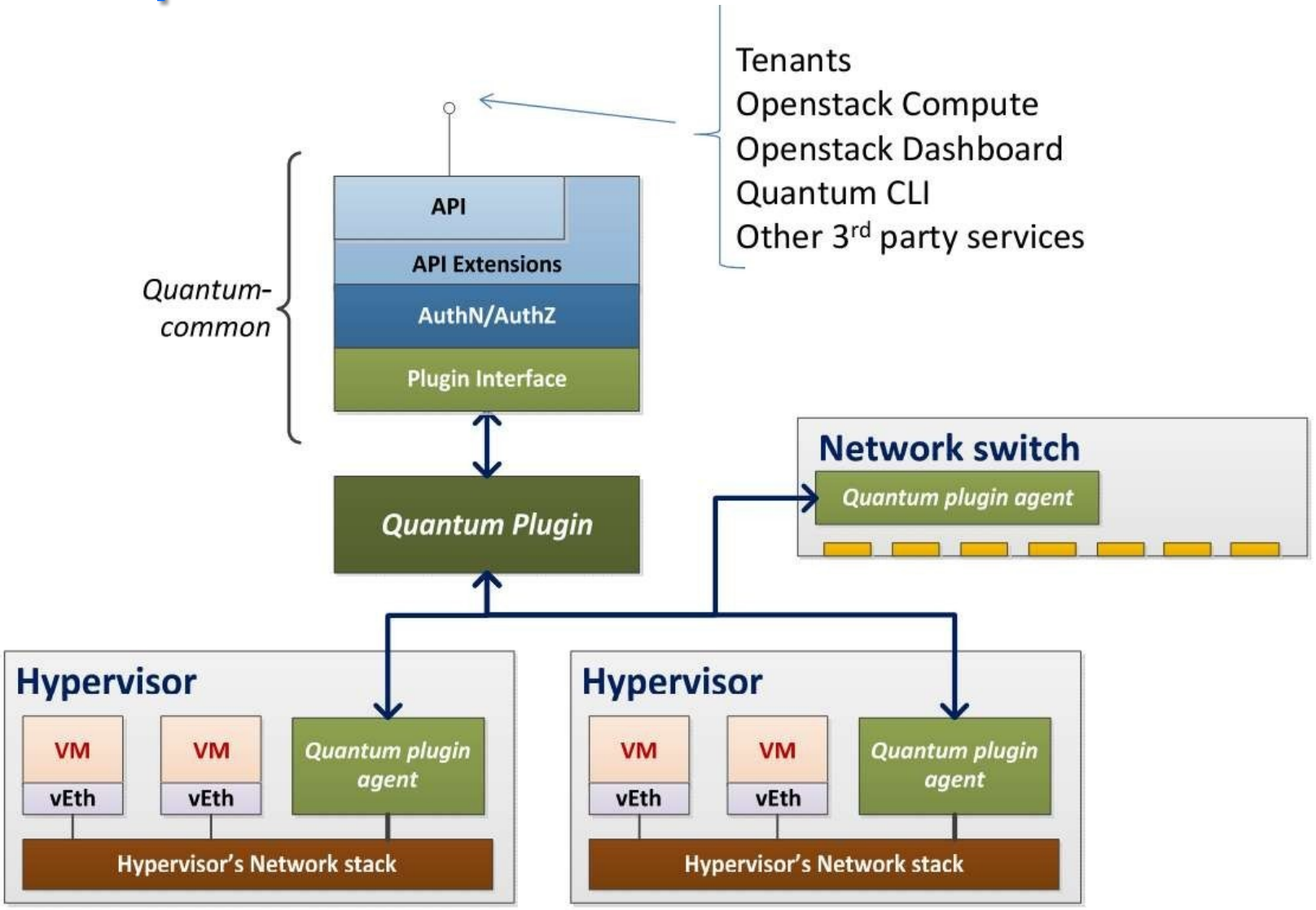
Provider View



Onix Functions



OpenStack Quantum Service

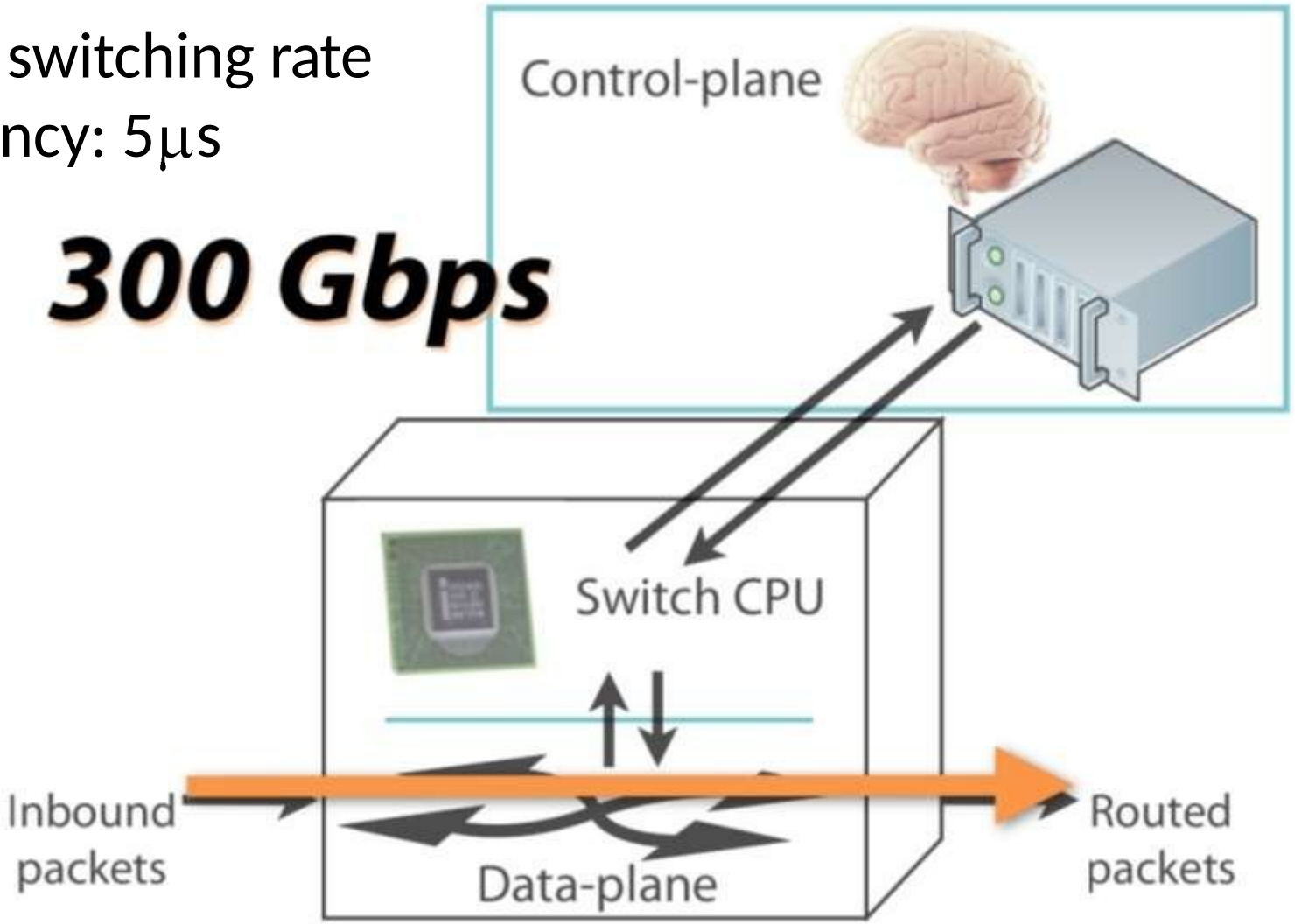


XenServer: Domain 0

Kernel-based VM: Linux Kernel

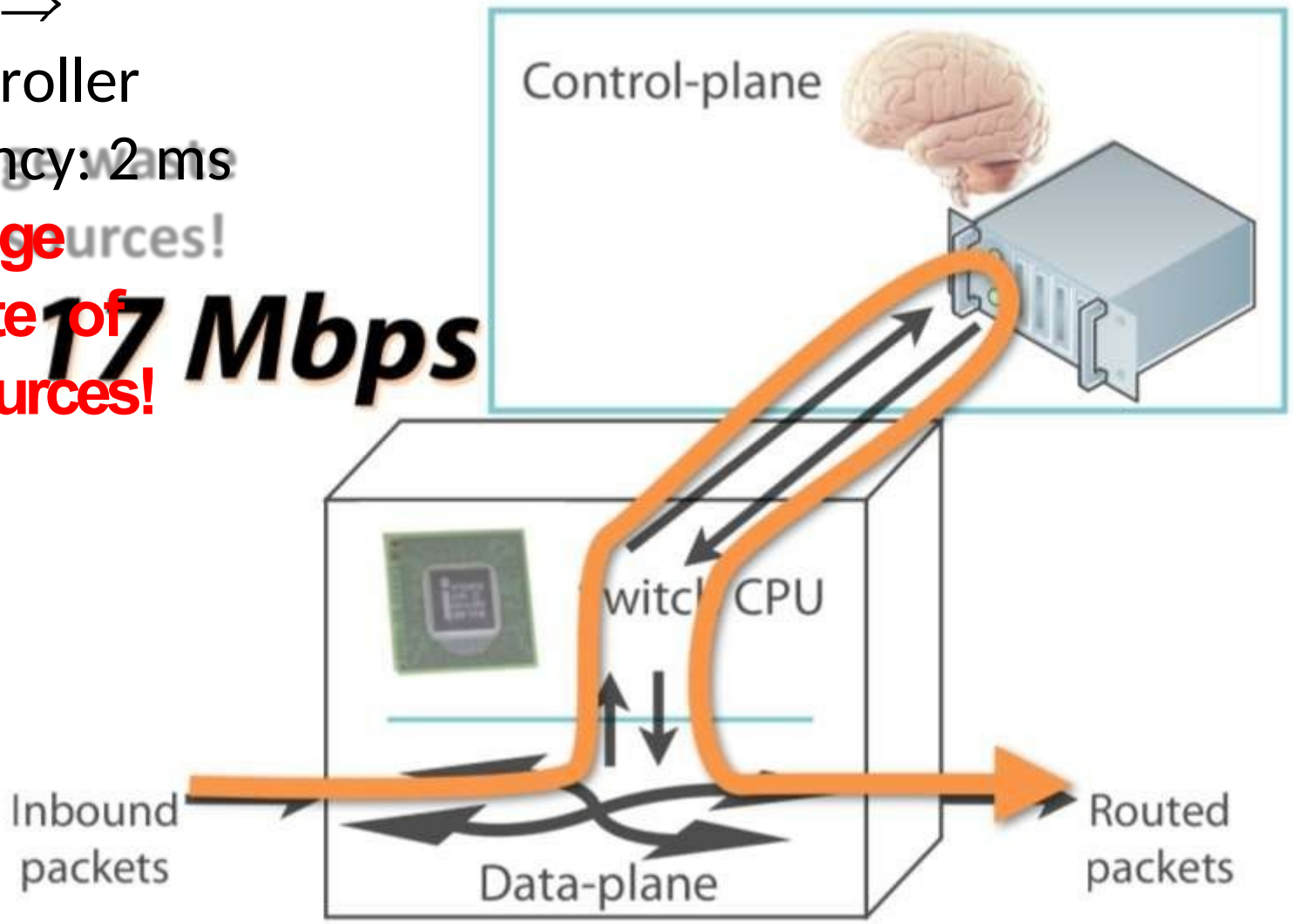
Always Call for Controller?

ASIC switching rate
Latency: 5μs



Always Call for Controller?

CPU ⇒
Controller
Latency: 2ms
A huge waste of resources!



Conclusion

- Concept of cloud computing is not brand new
  - But with new usage, demand, and economy
  - Aggregated individual small demands
  - Thus pressures traditional data centers
  - Clusters of commodities for performance and economy of scale
- Data Center Network challenges
  - Carry tons of apps, tenants, and compute tasks
  - Network delay / loss = service bottleneck
  - Still no consistent sys / traffic / analysis model

Questions?

Reference

- YA-YUNN SU, “Topics in Cloud Computing”, NTU CSIE 7324
- Luiz André Barroso and Urs Hölzle, “The Datacenter as a Computer - An Introduction to the Design of Warehouse-Scale Machines”, Google Inc.
- 吳柏均 , 郭嘉偉 , “MapReduce: Simplified Data Processing on Large Clusters”, CSIE 7324 in class presentation slides.
- Stanford, “Data Mining”, CS345A, http://www.stanford.edu/class/cs345a/slides/02-mapreduce.pdf
- Dr. Allen D. Malony, CIS 607: Seminar in Cloud Computing, Spring 2012, U. Oregon http://prodigal.nic.uoregon.edu/~hoge/cis607/
- Manel Bourguiba, Kamel Haddadou, Guy Pujolle, “Packet aggregation based network i/o virtualization for cloud computing”, Computer Communication 35, 2012
- Eric Keller, Jen Roxford, “The „Platform as a Service“ Model for Networking”, in Proc. INM WREN , 2010
- Martin Casado, Teemu Koponen, Rajiv Ramanathan, Scott Shenker, “Virtualizing the Network Forwarding Plane”, in Proc. PRESTO (November 2010)
- Guohui Wang T. S. Eugene Ng, “The Impact of Virtualization on Network Performance of Amazon EC2 Data Center”, INFOCOMM 2010
- OpenStack Documentation http://docs.openstack.org/

Reference

- Bret Piatt, OpenStack Overview, OpenStack Tutorial http://salsahpc.indiana.edu/CloudCom2010/slides/PDF/tutorials/OpenStackTutorialIEEEClo udCom.pdf http://www.omg.org/news/meetings/tc/ca-10/special-events/pdf/5-3_Piatt.pdf
- Vishvananda Ishaya, Networking in Nova http://unchainyourbrain.com/openstack/13-networking-in-nova
- Jaesuk Ahn, OpenStack, XenSummit Asia http://www.slideshare.net/ckpeter/openstack-at-xen-summit-asia http://www.slideshare.net/xen_com_mgr/2-xs-asia11kahnopenstack
- Salvatore Orlando, Quantum: Virtual Networks for Openstack http://qconlondon.com/dl/qcon-london-2012/slides/SalvatoreOrlando_QuantumVirtualNetworksForOpenStackClouds.pdf
- Dan Wendlandt, Openstack Quantum: Virtual Networks for OpenStack http://www.ovirt.org/wp-content/uploads/2011/11/Quantum_Ovirt_discussion.pdf
- David A. Maltz, “Data Center Challenges: Building Networks for Agility, Senior Researcher, Microsoft”, Invited Talk, 3rd Workshop on I/O Virtualization, 2011 http://static.usenix.org/event/wiov11/tech/slides/maltz.pdf
- Amin Vahdat, “PortLand: Scaling Data Center Networks to 100,000 Ports and Beyond”, Stanford EE Computer Systems Colloquium, 2009 http://www.stanford.edu/class/ee380/Abstracts/091118-DataCenterSwitch.pdf

Reference

- Mohammad Al-Fares , Alexander Loukissas , Amin Vahdat, “A scalable, commodity data center network architecture”, ACM SIGCOMM 2008
- Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, Sudipta Sengupta, “VL2: a scalable and flexible data center network”, ACM SIGCOMM 2009
- Radhika Niranjan Mysore, Andreas Pamboris, Nathan Farrington, Nelson Huang, Pardis Miri, Sivasankar Radhakrishnan, Vikram Subramanya, Amin Vahdat, “PortLand: a scalable fault-tolerant layer 2 data center network fabric”, ACM SIGCOMM 2009
- Jayaram Mudigonda, Praveen Yalagandula, Mohammad Al-Fares, Jeffrey C. Mogul, “SPAIN: COTS data-center Ethernet for multipathing over arbitrary topologies”, USENIX NSDI 2010
- Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, Jonathan Turner, “OpenFlow: enabling innovation in campus networks”, ACM SIGCOMM 2008
- Mohammad Al-Fares, Sivasankar Radhakrishnan, Barath Raghavan, Nelson Huang, Amin Vahdat, “Hedera: dynamic flow scheduling for data center networks”, USENIX NSDI 2010
- M. Alizadeh, B. Atikoglu, A. Kabbani, A. Lakshmikantha, R. Pan, B. Prabhakar, and M. Seaman, “Data center transport mechanisms: Congestion control theory and IEEE standardization,” Communication, Control, and Computing, 2008 46th Annual Allerton Conference on

Reference

- A. Kabbani, M. Alizadeh, M. Yasuda, R. Pan, and B. Prabhakar. “AF-QCN: Approximate fairness with quantized congestion notification for multitenanted data centers”, In High Performance Interconnects (HOTI), 2010, IEEE 18th Annual Symposium on
- Adrian S.-W. Tam, Kang Xi H., Jonathan Chao , “Leveraging Performance of Multiroot Data Center Networks by Reactive Reroute”, 2010 18th IEEE Symposium on High Performance Interconnects
- Daniel Crisan, Mitch Gusat, Cyriel Minkenberg, “Comparative Evaluation of CEE-based Switch Adaptive Routing”, 2nd Workshop on Data Center - Converged and Virtual Ethernet Switching (DC CAVES), 2010
- Teemu Koponen et al., “Onix: A distributed control platform for large-scale production networks”, OSDI, Oct, 2010
- Andrew R. Curtis (University of Waterloo); Jeffrey C. Mogul, Jean Tourrilhes, Praveen Yalagandula, Puneet Sharma, Sujata Banerjee (HP Labs), SIGCOMM 2011

Backup Slides

Symmetric Multi-Processing (SMP): Several CPUs on shared RAM-like memory

What advantages do SMP offer

1. A global view: offered by shared memory
2. Low communication latency: accessing global data
   - SMP: ~100ns
   - 1 Gbps LAN: ~100us (100, 000 ns)

- How does this affect application performance?

# of global access per 1ms of work

$$Execution\ time = 1ms + f * [100ns * \frac{1}{\#\ of\ nodes} + 100ms * \left(1 - \frac{1}{\#\ of\ nodes}\right)]$$

Local computation

Time to access global data

This is 0 for one node

↑Data distributed evenly among nodes

Compare the execution time as number of nodes increases

Typical datacenter CRAC