Cloud Computing, Open Stack and Data-centers

Gwendal Simon

http://perso.telecom-bretagne.eu/gwendalsimon
@gwendal

Recommended Book

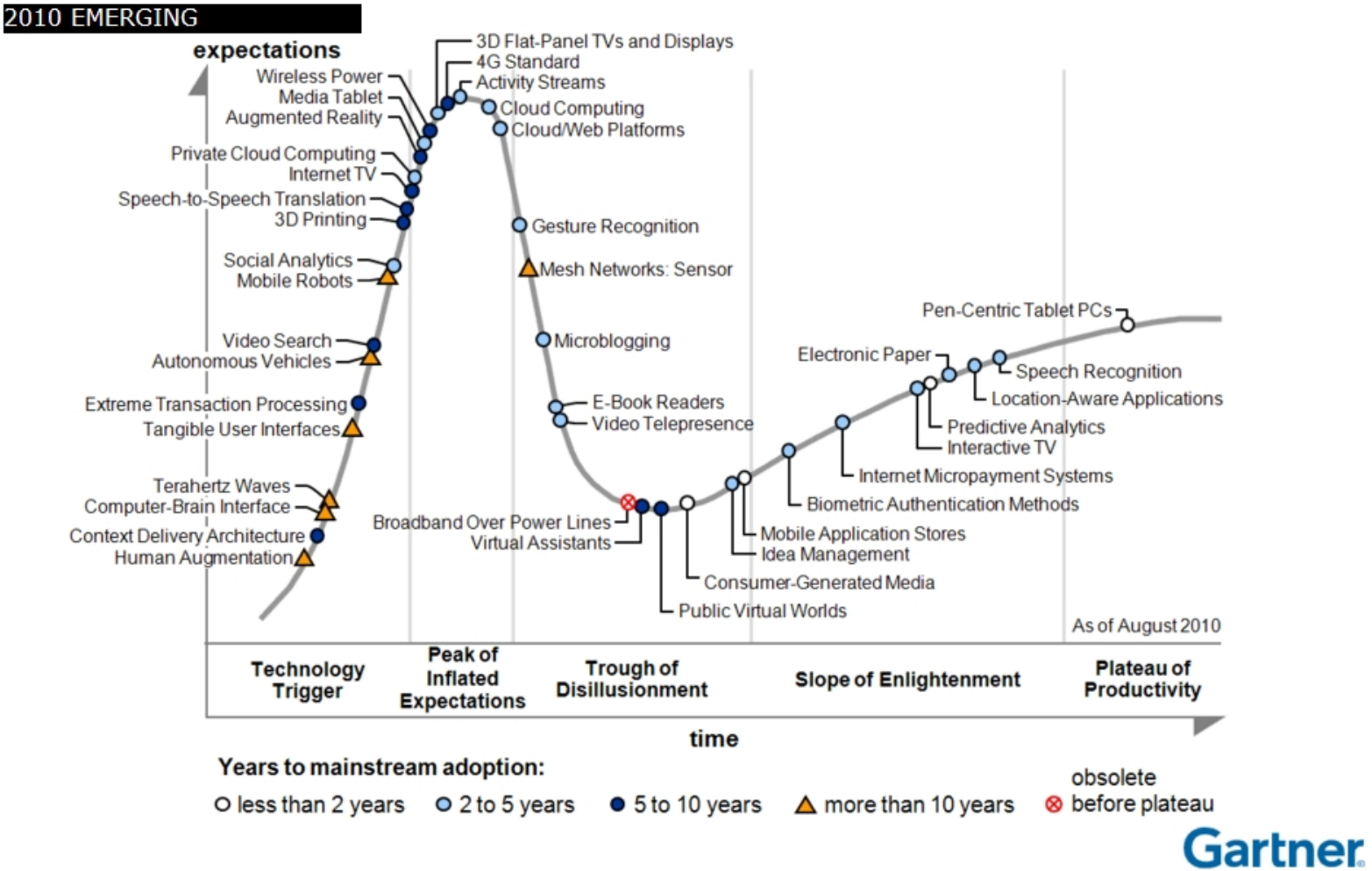
The Datacenter as a Computer : An Introduction to the Design of Warehouse-Scale Machines

2nd edition (2013)

Luiz Barroso, Jimmy Clidaras, and Urs Hölzle

Morgan & Claypool publishers

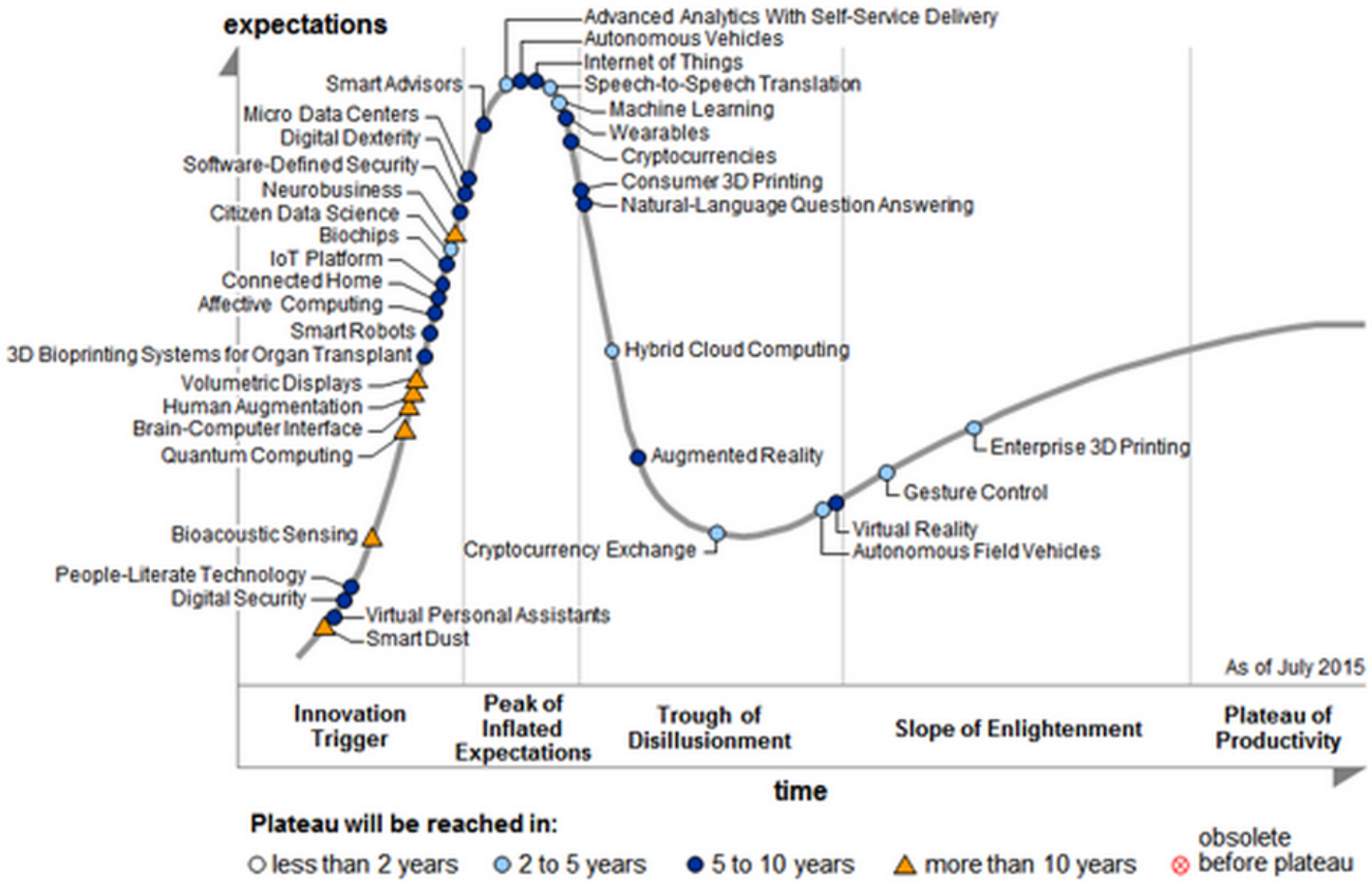
A hyped tech in 2010



Gartner

No longer hyped, but operational

Figure 1. Hype Cycle for Emerging Technologies, 2015



Definition in a nutshell

Cloud computing is a model for enabling convenient, *on-demand* network access to a *shared* pool of configurable computing resources (*e.g.*, networks, servers, storage, applications, and services) that can be rapidly *provisioned* and released with *minimal management effort* or service provider interaction

NIST : http ://csrc.nist.gov/groups/SNS/cloud-computing/index.html

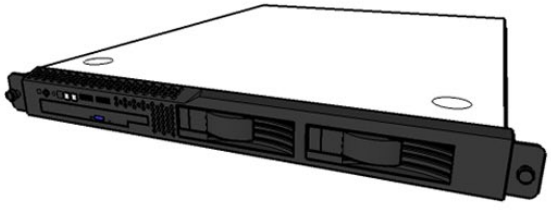
The ∗aaS paradigm

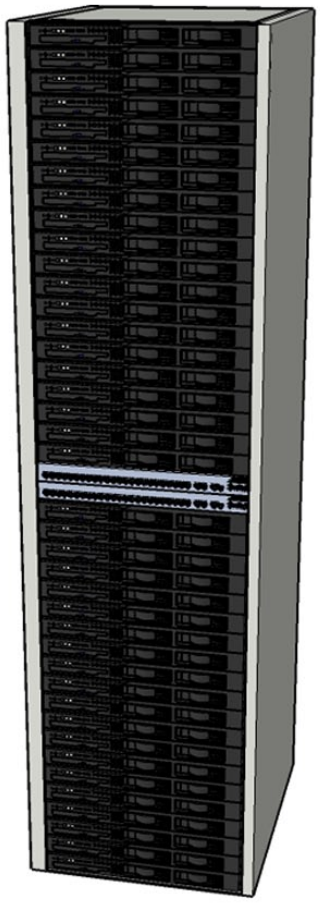
- SaaS : Software as a Service
  - applications for *end-users*
    - email, office suite, photos sharing, video storage
- PaaS : Platform as a Service
  - services for *web app developers*
    - workflow facilities and various basic services
- IaaS : Infrastructure as a Service
  - resources for *developers*
    - servers, network equipment, memory, CPU

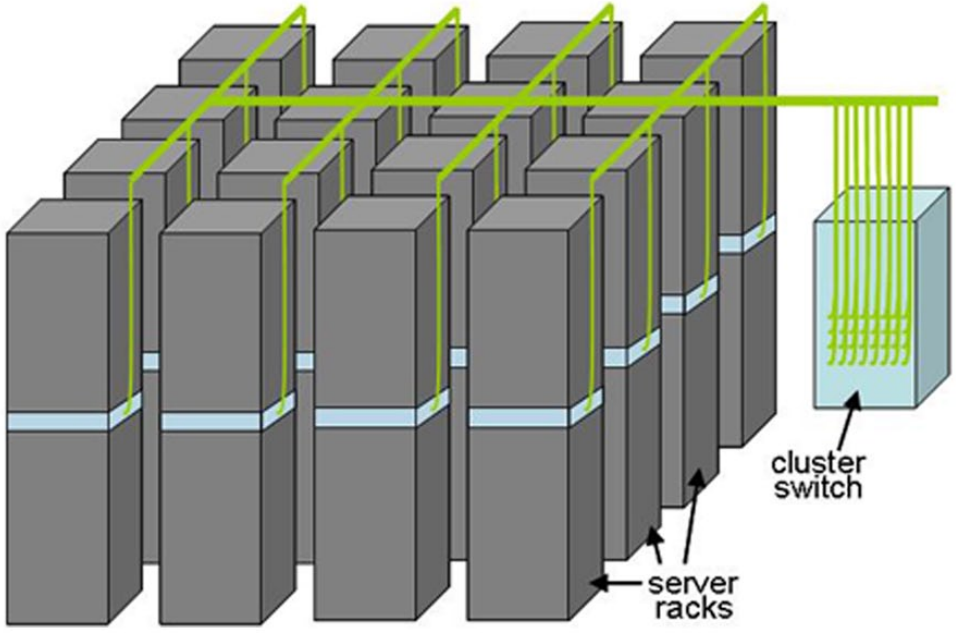
Avancement

1. Basic Elements of a Datacenter
2. Case Studies
3. Dealing with Concerns
4. Software Infrastructure
5. Conclusion

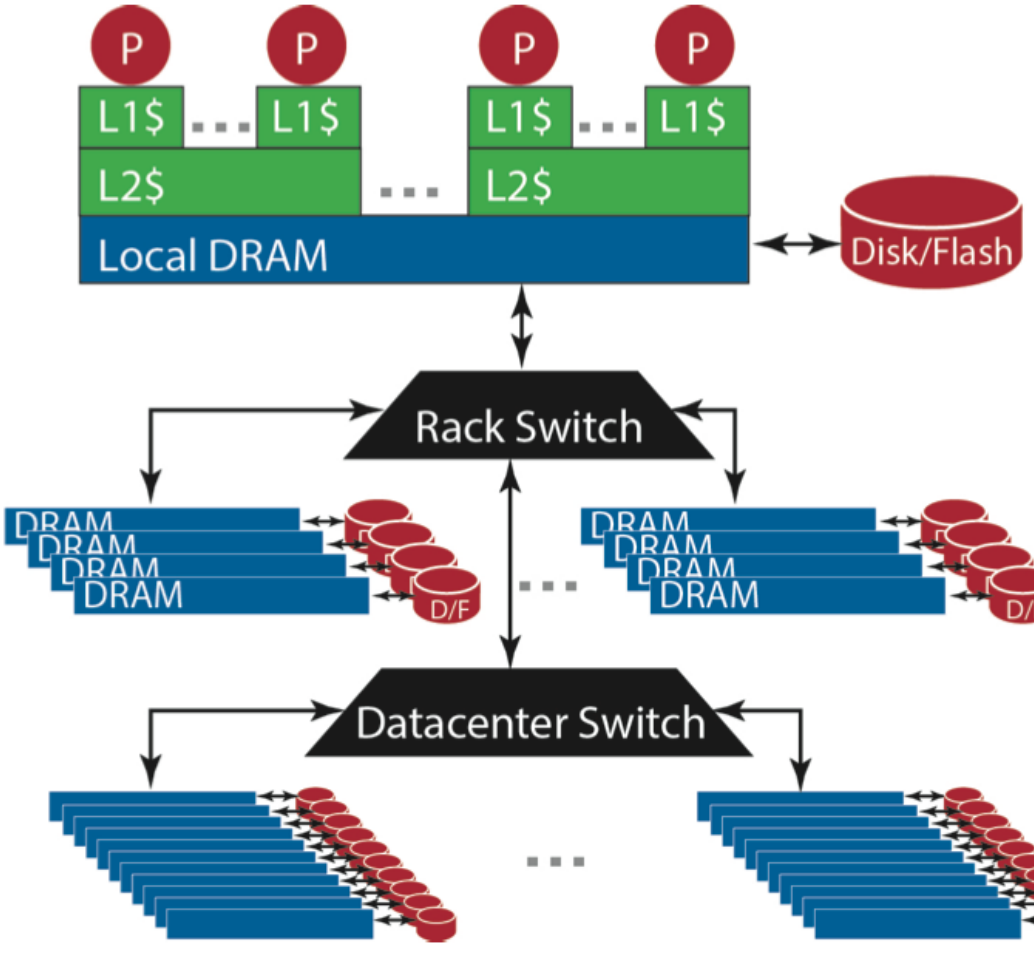
Architecture Basics







Storage and Access Time



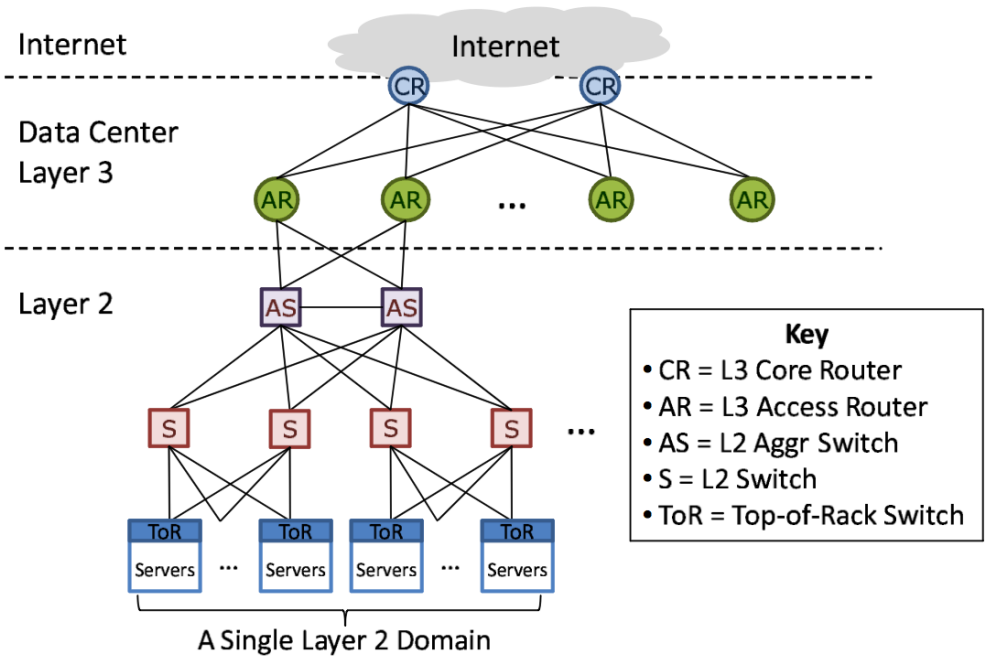
One Server
DRAM: 16 GB, 100 ns, 20 GB/s
Disk: 2T B, 10 ms, 200 MB/s
Flash: 128 GB, 100 us, 1 GB/s

Local Rack (80 servers)
DRAM: 1 TB, 300 us, 100 MB/s
Disk: 160 TB, 11 ms, 100 MB/s
Flash: 20 TB, 400 us, 100 MB/s

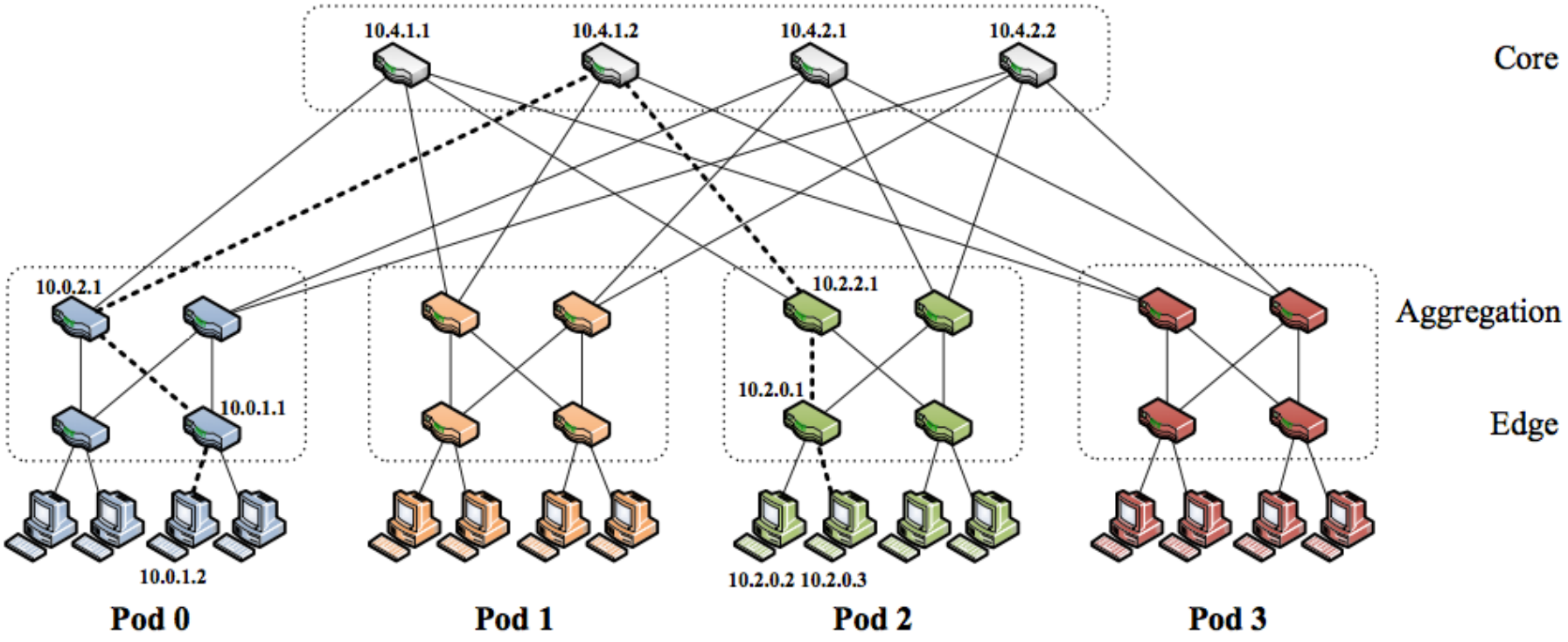
Cluster (30 racks)
DRAM: 30 TB, 500 us, 10 MB/s
Disk: 4.80 PB, 12 ms, 10 MB/s
Flash: 600 TB, 600 us, 10 MB/s

Oversubscription :

$$\frac{\text{worst-case communication between servers}}{\text{capacity of network topology}}$$



Network Fat-Tree Topologies



Typical Technologies and Prices (in 2013)

Server :

- computation : 24-32 cores at $200 per core
- network : $150 per 10 Gbps port NIC

10 Gbps switch :

- today : 48-64 ports at $450 per port
- trend : up to 150 ports at $100 per port

Memory :

Technology	Access time (ns)	Cost ($/MB)	Max. size
SRAM	4	27	$\approx$ 250 Mbits
RLDRAM	15	0.27	$\approx$ 2 Gbits
DRAM	55	0.016	$\approx$ 100 Gbits

Avancement

1 Basic Elements of a Datacenter

2 Case Studies

3 Dealing with Concerns

4 Software Infrastructure

5 Conclusion

Case Studies

- A Commercial Offer : Amazon
- Switching to the Cloud : Netflix

Amazon Web Services

- DynamoDB
- Simple Storage Service (S3)
- Elastic Compute Cloud (EC2)
- Elastic Load Balancing (ELB)
- Virtual Private Cloud (VPC)
- Glacier
- Elastic MapReduce (EMR)
- CloudFront

Case Studies

- A Commercial Offer : Amazon
- Switching to the Cloud : Netflix

See Slides

http ://www.slideshare.net/adrianco/netflix-global-cloud

Avancement

1 Basic Elements of a Datacenter

2 Case Studies

3 Dealing with Concerns

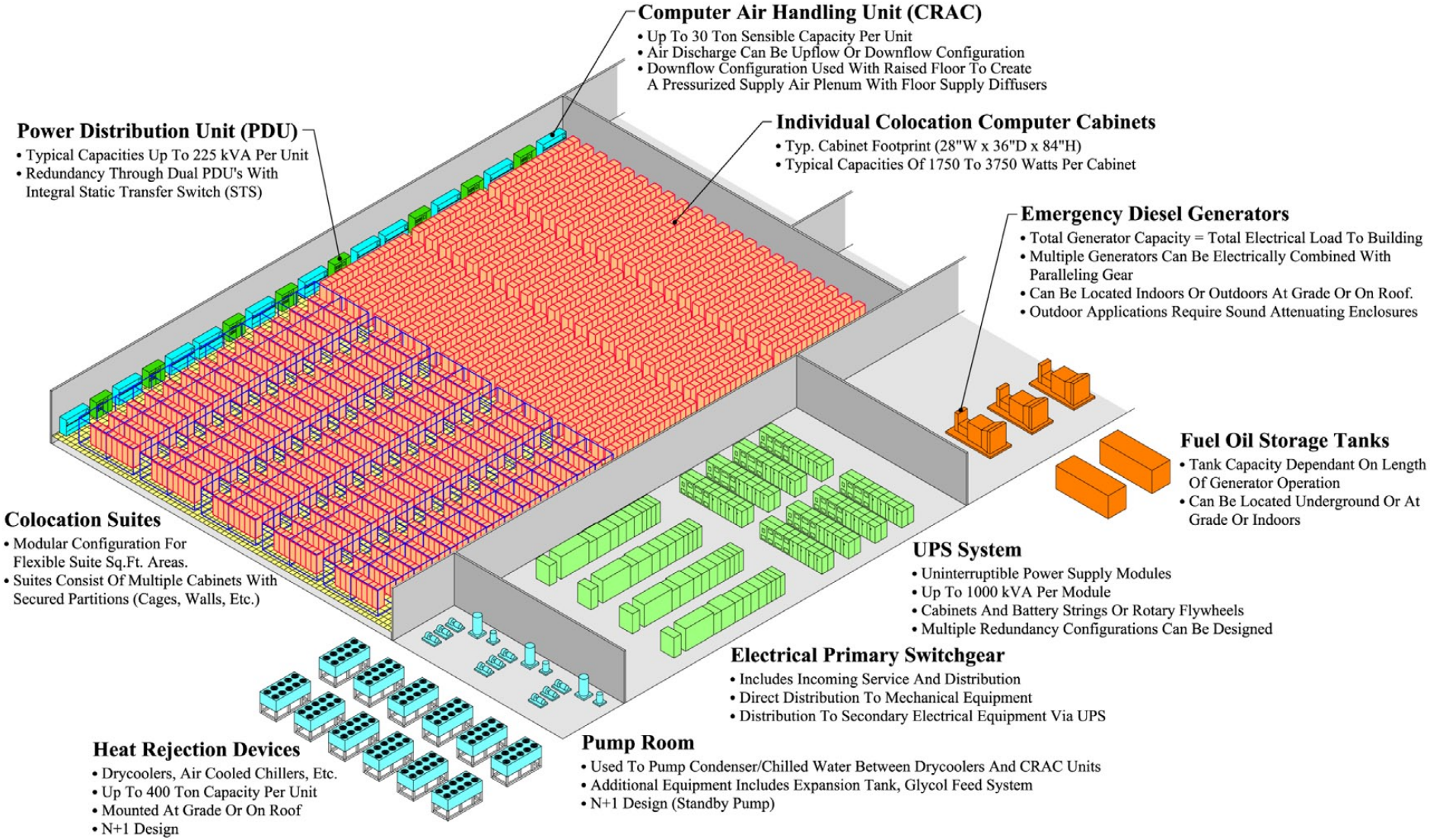
4 Software Infrastructure

5 Conclusion

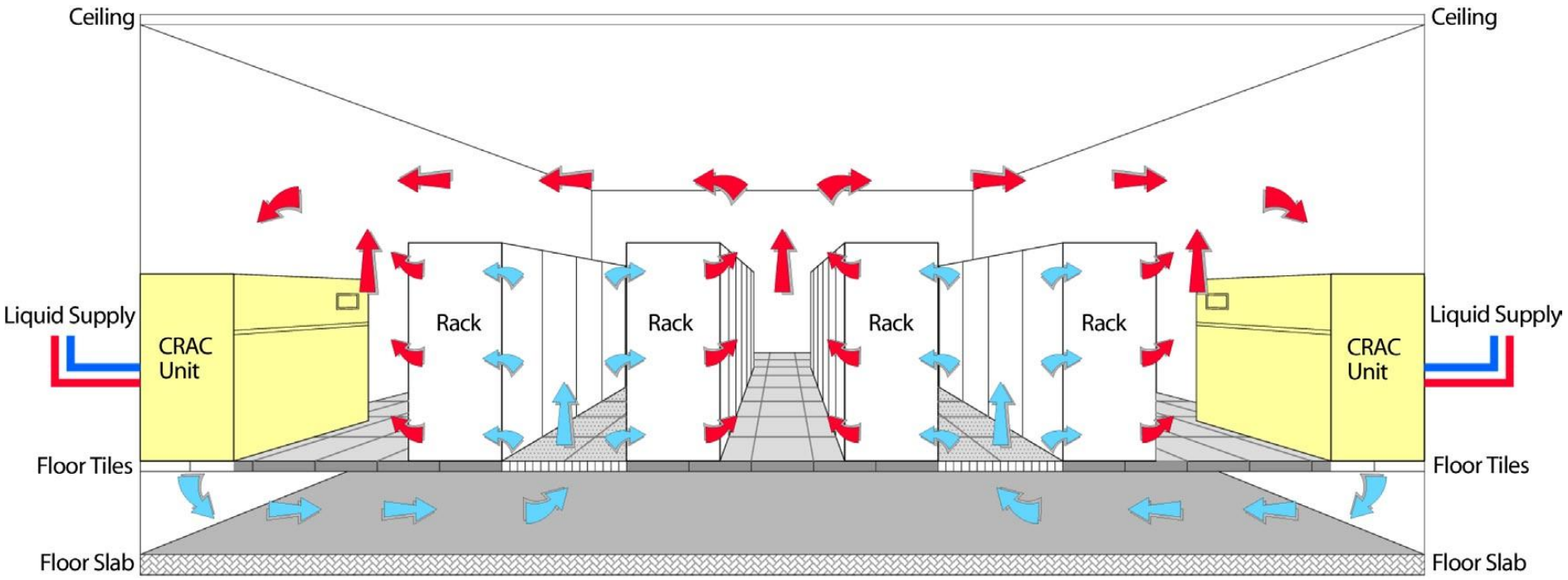
Dealing with Concerns

- Energy
- Failures

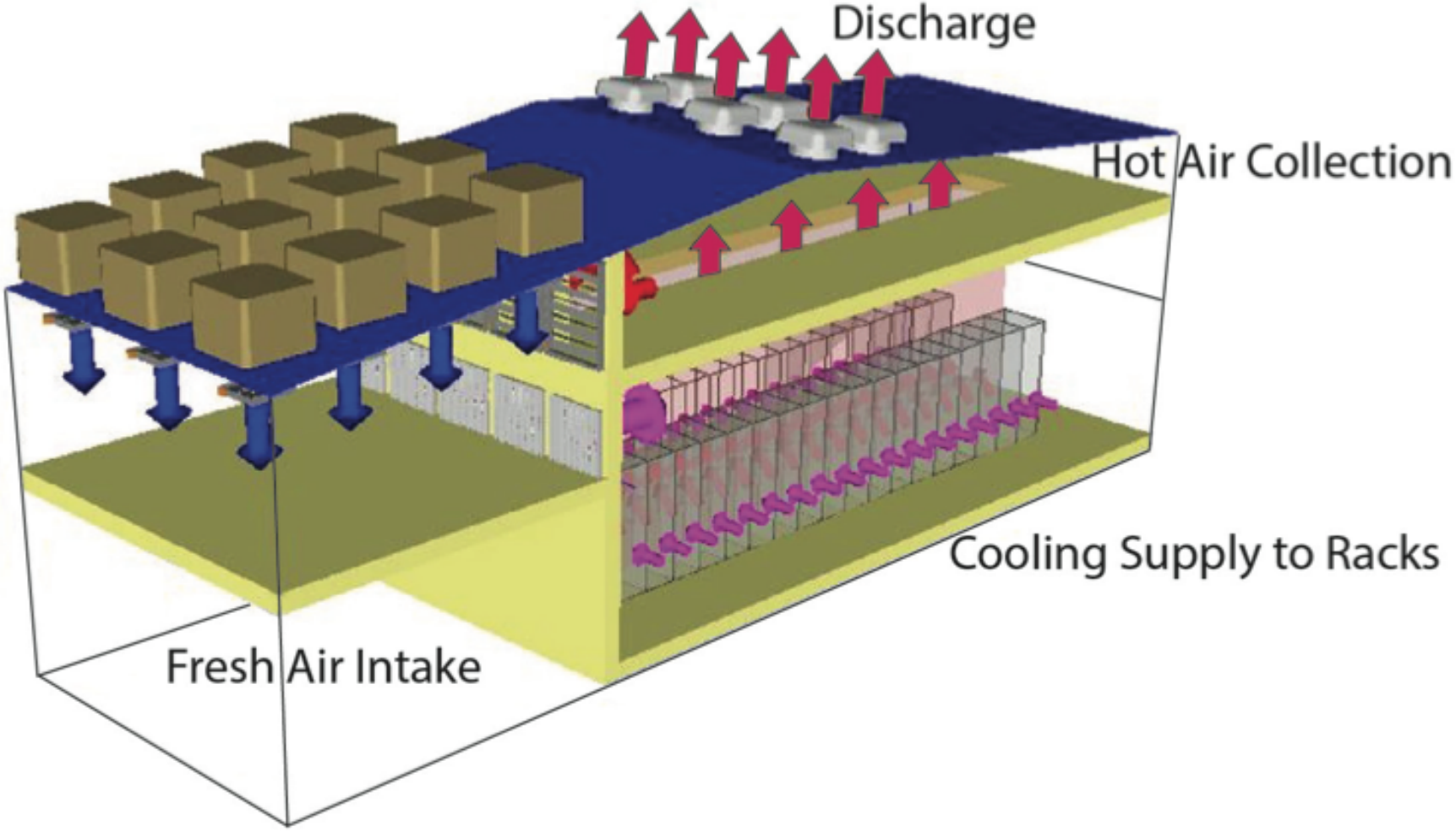
Main Electric Components



Cooling Challenge (closed loop)



Cooling Challenge (open loop)



Evaluating Energy Efficiency

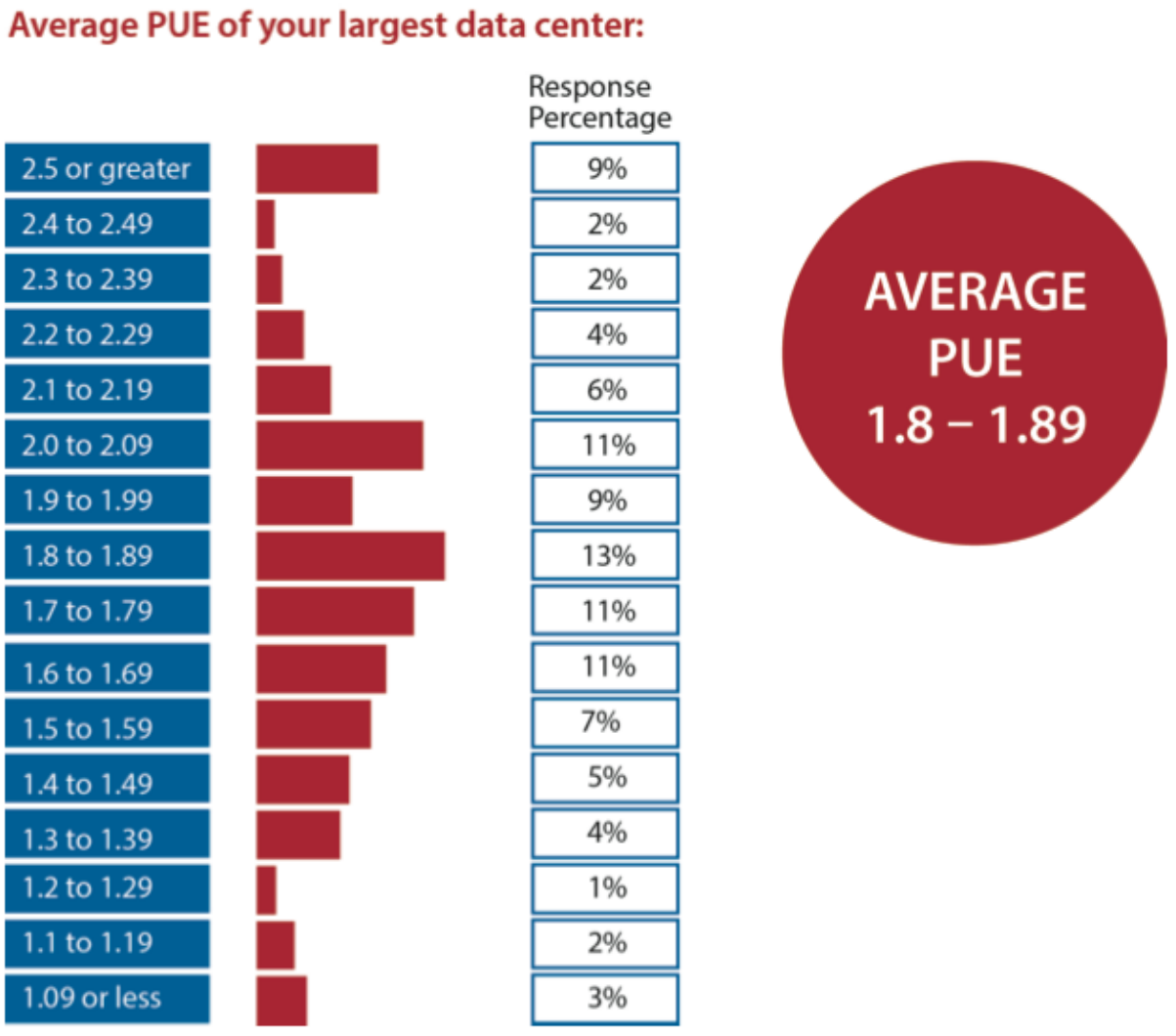
$$\textit{Power Usage Effectiveness} = \frac{\text{total energy}}{\text{energy in equipment}}$$

- first generation datacenter PUE ≥ 3.0
- toward PUE around 1.1

$$\textit{Server PUE} = \frac{\text{total server power}}{\text{critical component power}}$$

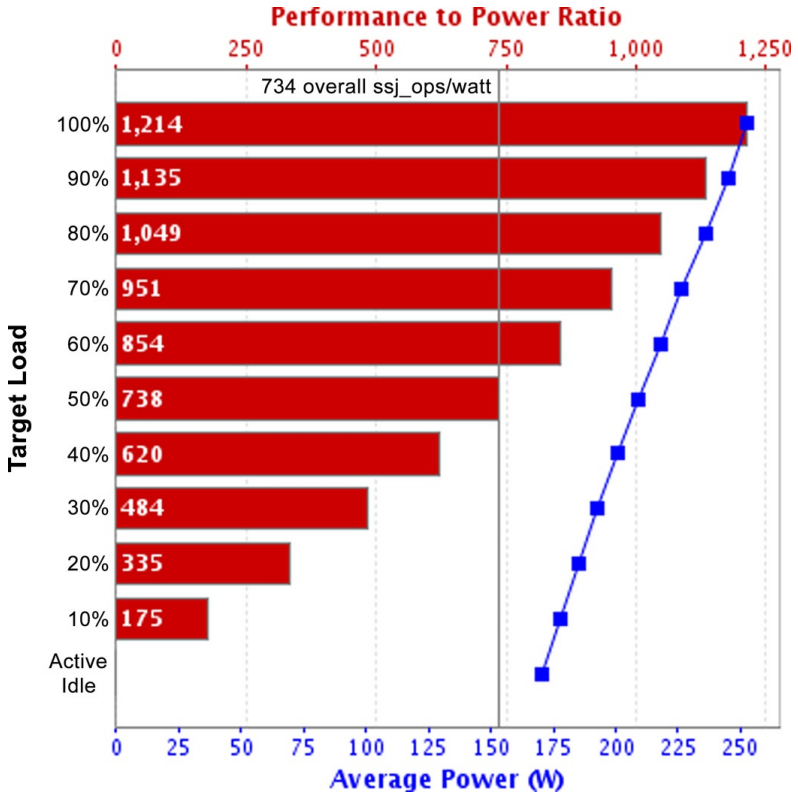
- basic SPUE is 1.7
- state-of-the-art servers reach 1.1

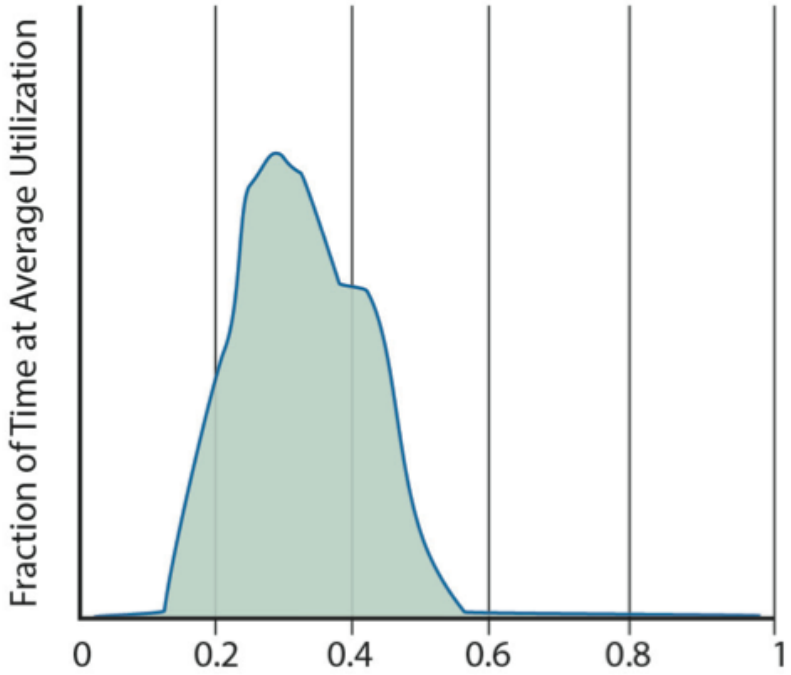
Current data-center PUE



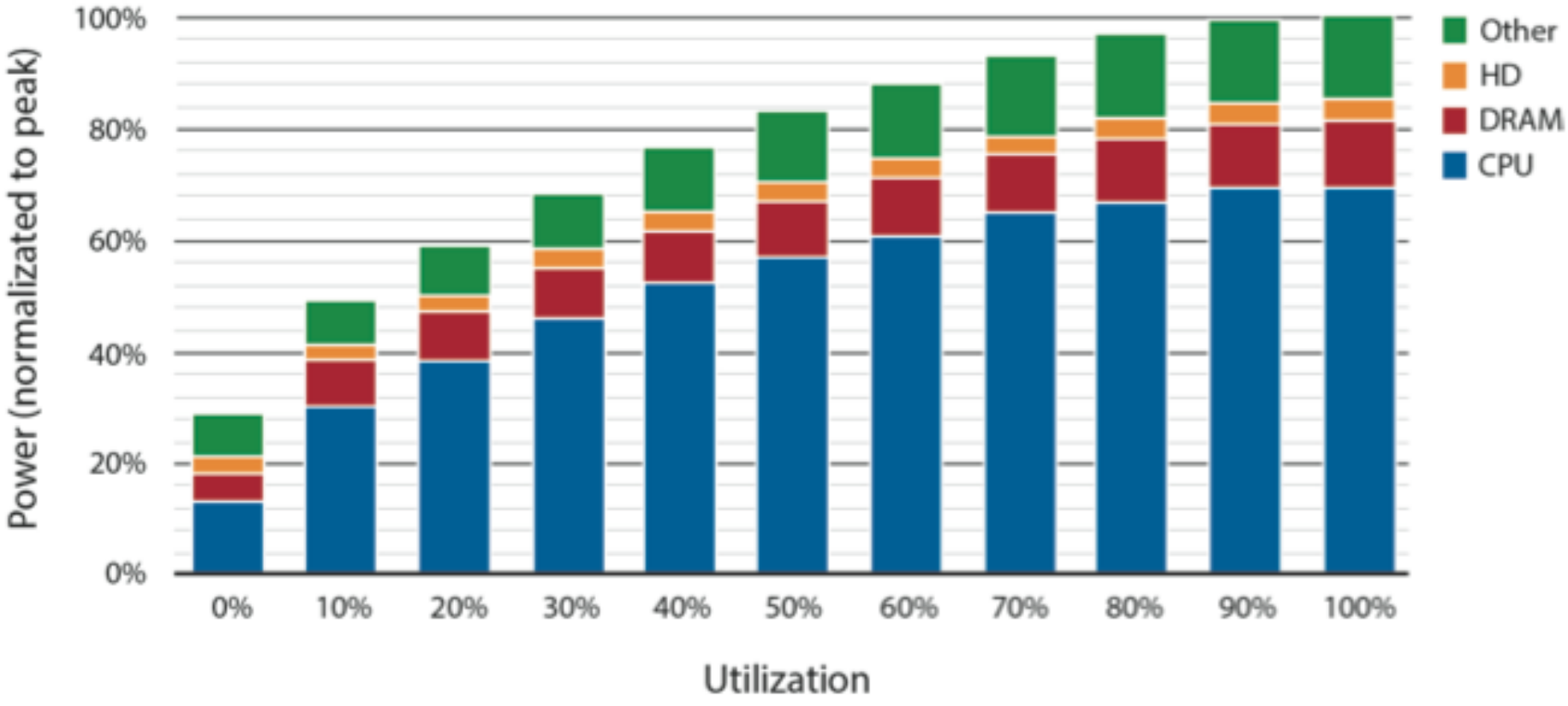
Computing Efficiency

Benchmarking cluster-level efficiency : on-going work
Benchmarking individual computer is easier





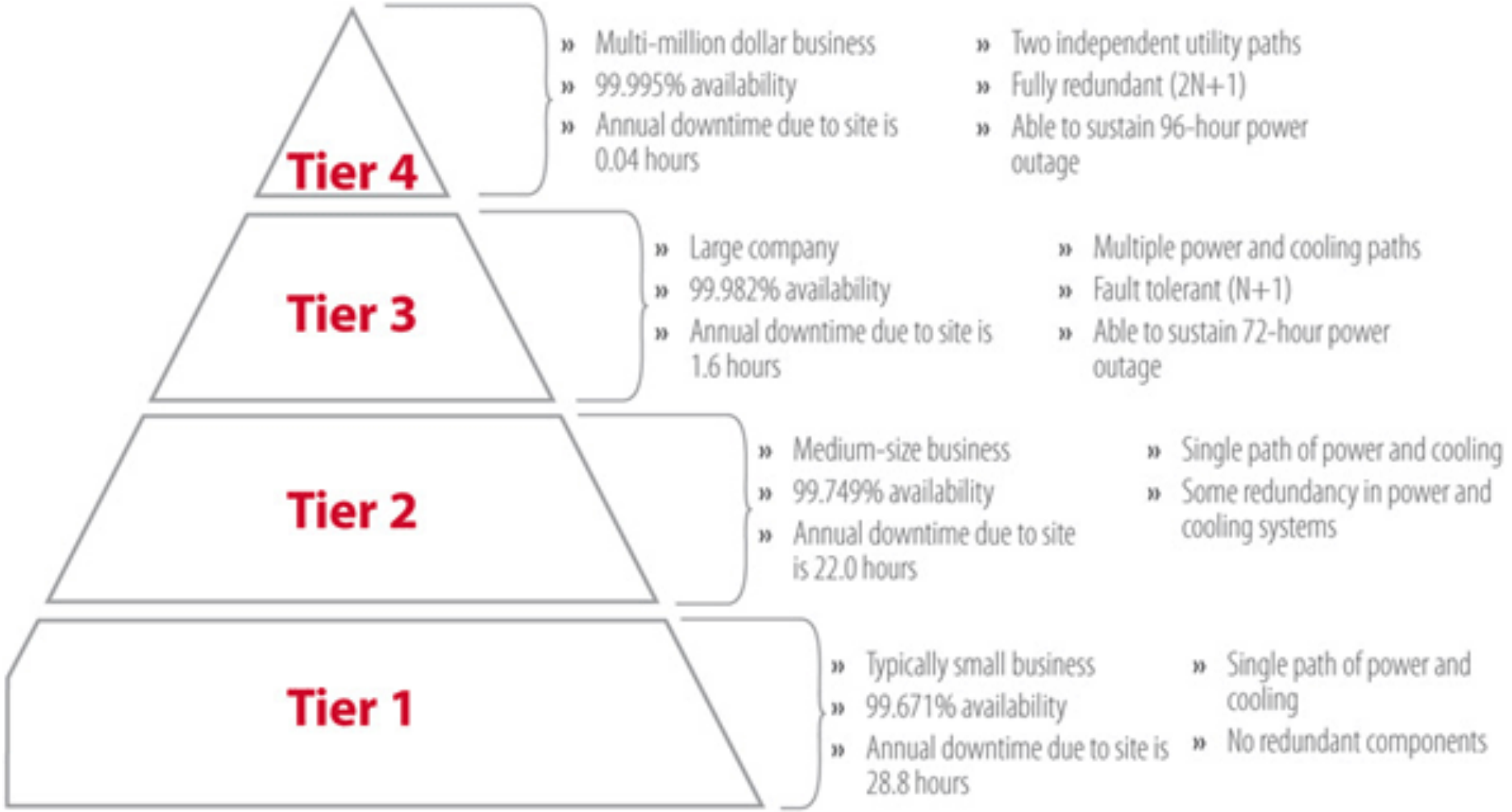
Toward Energy-Proportional Servers



Dealing with Concerns

- Energy
- Failures

Data-Center Tiers



Origin of Impacting Failures

Cause	% of events
software	33 %
configuration	28 %
human	13 %
network	12 %
hardware	11 %
other	3 %

hardware faults are masked by fault-tolerant software

Failures and Crash

Average machine availability is $\simeq$ 99.9%

- 95% of machines restart less than once a month
- 80% of restart events last less than 10 minutes

Software most frequent faults (in one year) :

- *DRAM soft-errors* : 1% experience uncorrectable err
- *disk soft-errors* : 3% of drives see corrupted sectors

Avancement

1 Basic Elements of a Datacenter

2 Case Studies

3 Dealing with Concerns

4 Software Infrastructure

5 Conclusion

Software Infrastructure

- Cluster-Level : MapReduce
- Open Stack

Motivation

Map/Reduce is a software framework for easily writing applications which ***process*** *vast amounts of data* (multi-terabyte data-sets) *in-parallel* on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner.

Functional Programming

Two fundamentals functions :

- **map** : apply a function to a list of elements
  - `map f [] = []`
    `| map f [x::xs] = (f x) :: (map f xs)`
  - *map square [1,2,5]* $\rightarrow$ *[1,4,25]*
- **reduce** : build a value from a function and a list
  - `reduce f a [] = a`
    `| reduce f a [x::xs] = reduce f (f x a) xs`
  - *reduce add 0 [1,3,6]* $\rightarrow$ *10*

MapReduce

Implementing two functions w.r.t data `(key,val)`

- **map** : smaller sub-problems distributed to nodes
  - `map (inKey, inVal)` $\rightarrow$ `list (outKey, v)`
  - produces *intermediate* values with an output key

- **reduce** : combines results of sub-problems
  - `reduce (outKey, list v)` $\rightarrow$ `outVal`
  - produces an output value from intermediate values

Example : Word Count

```
map(filename, content):
  for each w in content:
     emitInt(w, 1)

reduce(word, partCount):
  int result = 0
  for pc in partCount:
     result += pc
  emit(result)
```

map(file1, “hello me, goodbye me”)→
<hello,1> <me,1> <goodbye,1> <me,1>

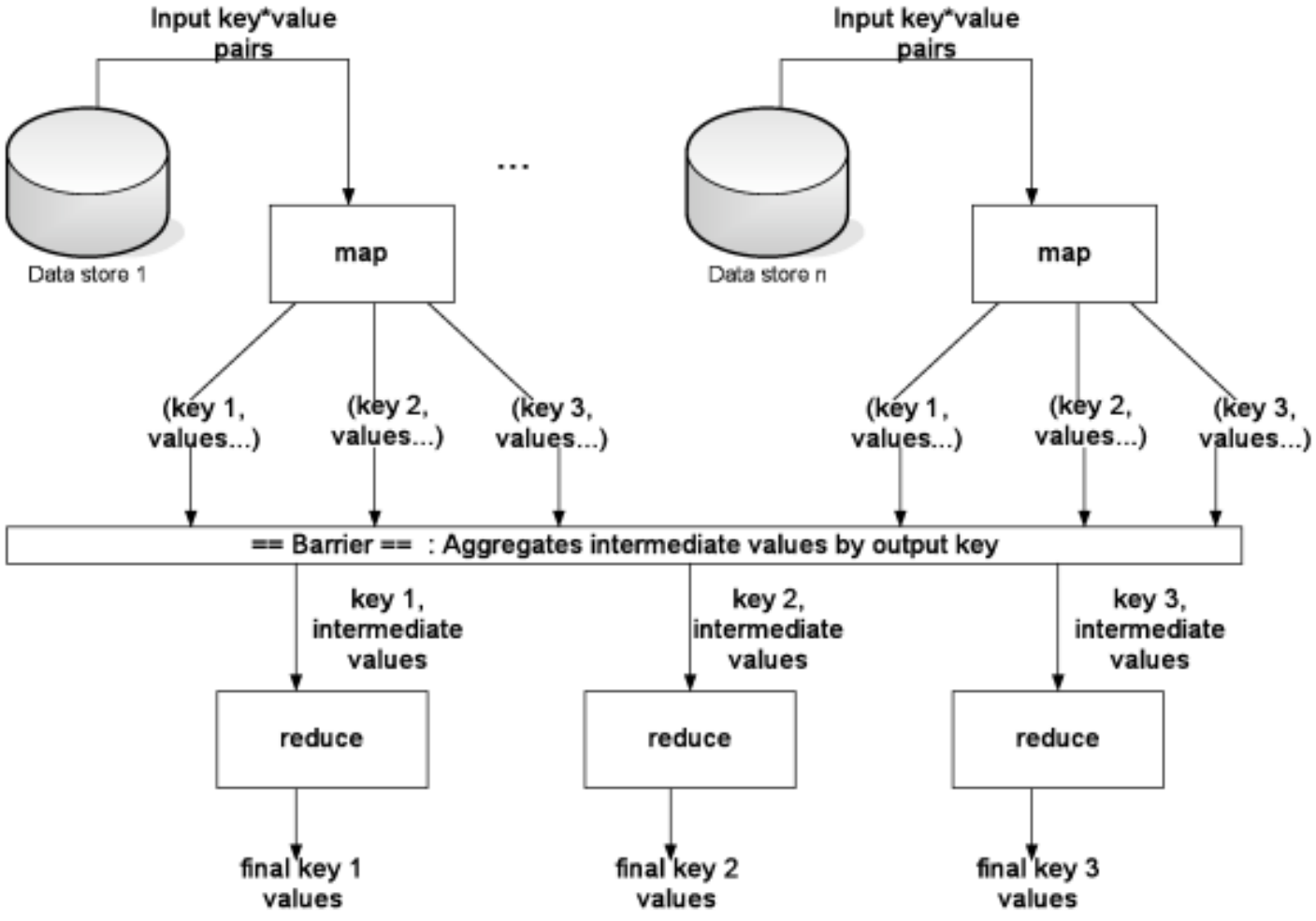
map(file2, “hello you, bye you”)→
<hello,1> <you,1> <bye,1> <you,1>

a given *key* is allocated to a given *server*

reduce(hello,<1,1>) → 2
...

<hello,2> <me,2> <you,2> <goodbye,1> <bye,1>

MapReduce Implemented



Software Infrastructure

- Cluster-Level : MapReduce
- Open Stack

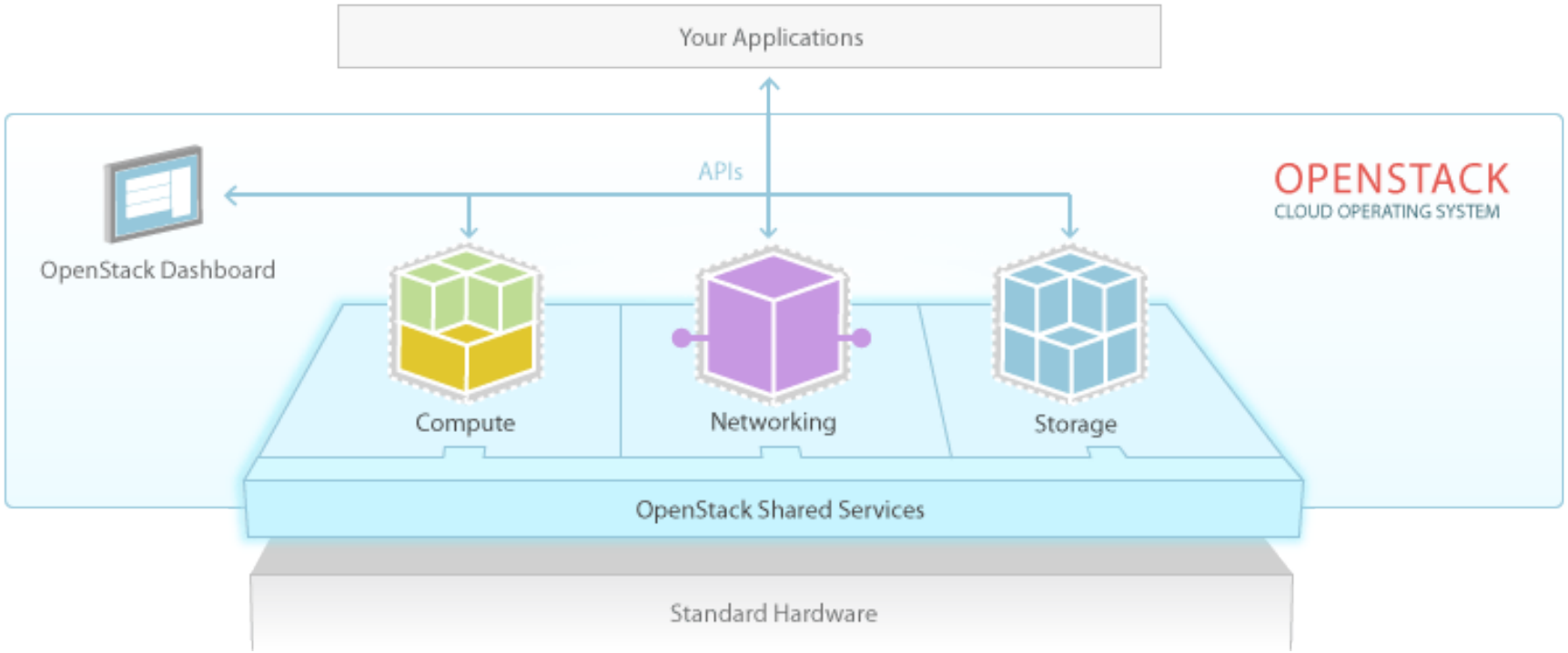
Main Idea

OpenStack is an open source IaaS initiative for creating and managing large groups of virtual private servers in a data center.

The goals of the OpenStack initiative are to

- support interoperability between cloud services
- build cloud services in own data centers

In a diagram

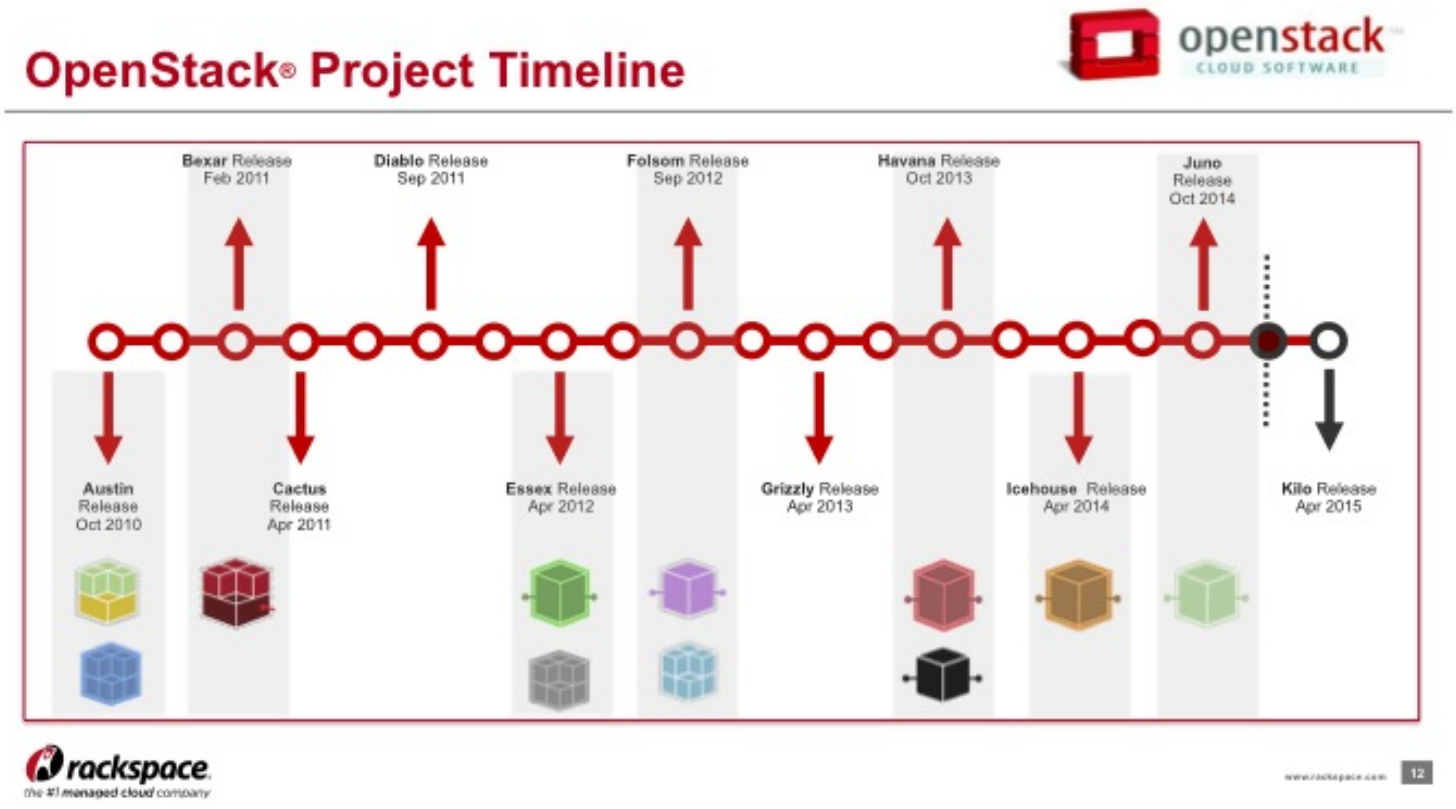

Your Applications
APIs
OPENSTACK
CLOUD OPERATING SYSTEM
OpenStack Dashboard
Compute
Networking
Storage
OpenStack Shared Services
Standard Hardware

History

- 2010 : Rackspace and NASA joint effort
- until 2012 : managed by Rackspace and 25 partners
- 2012 : non-profit foundation named OpenStack

Now 600 supporting companies :
https ://www.openstack.org/foundation/companies/

Releases



and then **Liberty** (2015.2) and **Mitaka** (2016.1)
next will be **Newton** (2016.2) and **Ocata** (2017.1)

Services

Core services :

Nova	manages lifecycle of compute instances
Neutron	enables network connectivity
Keystone	provides authentication
Swift	stores and retrieves any data *object*
Cinder	provides persistent *block* storage
Glance	stores and retrieves VM disk images

And various optional (also less mature) projects :

- Horizon, Heat, Magnum, Sahara, ...

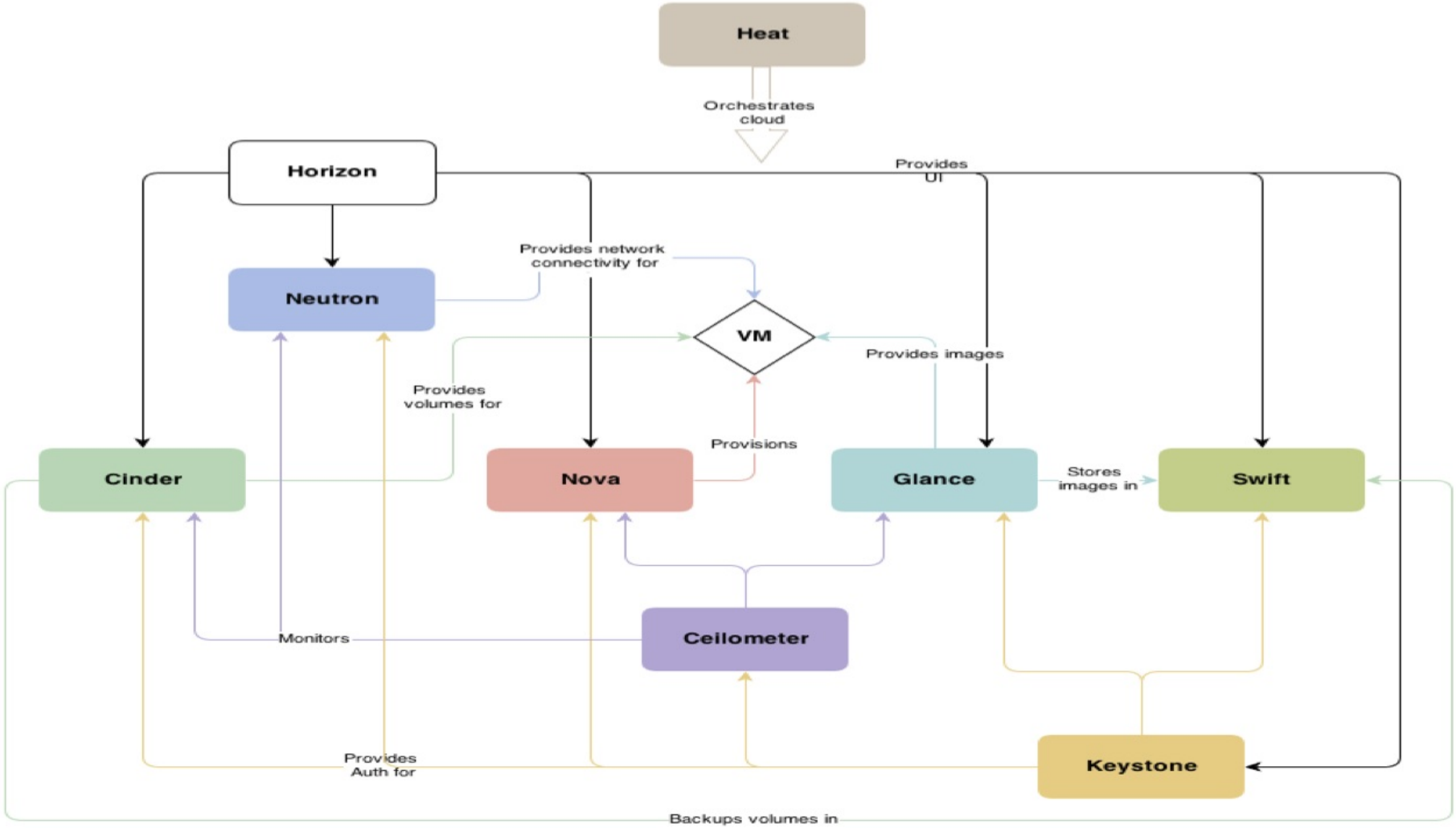
Community

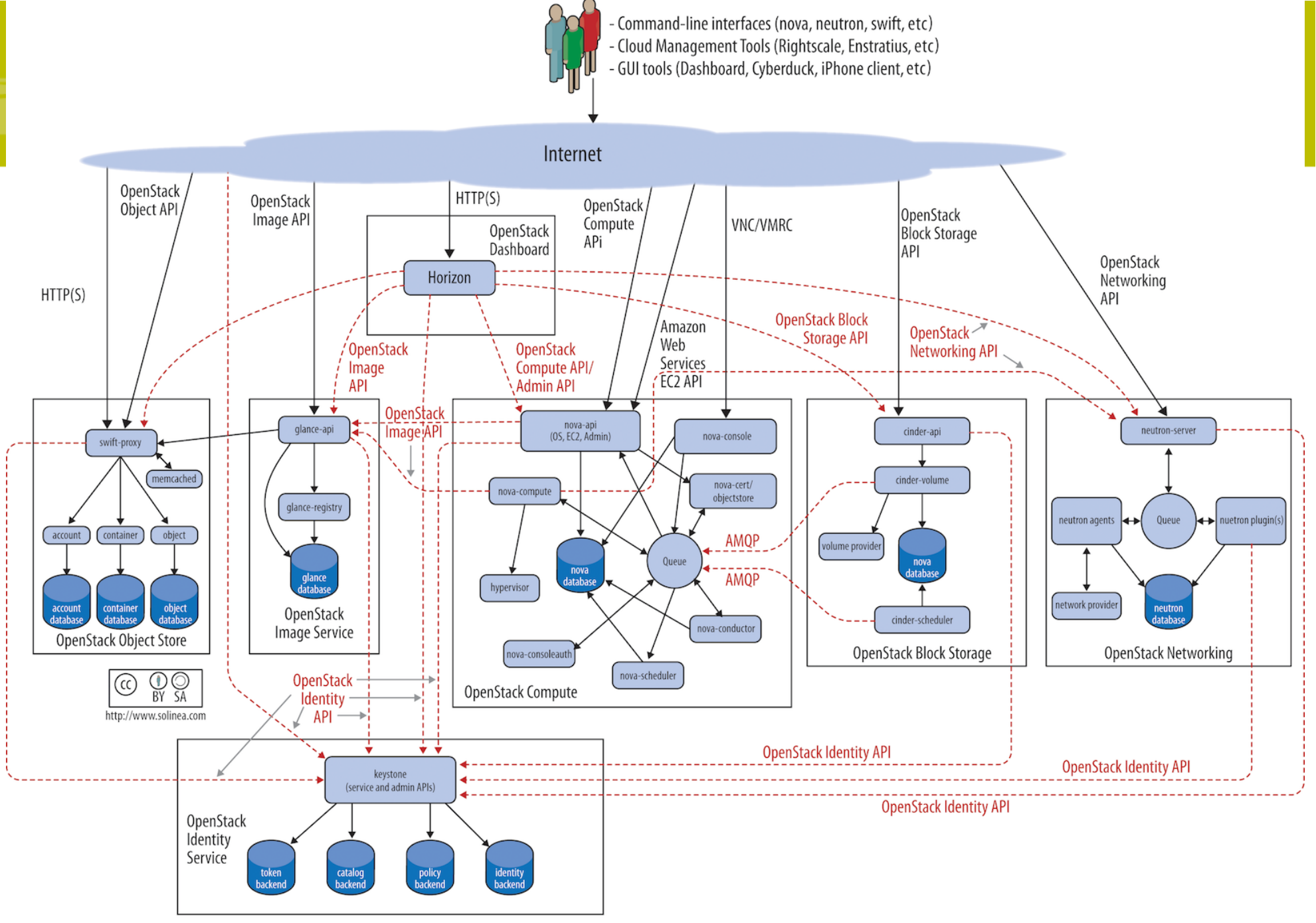
A huge community :

- 40k registered people from 179 countries
- 20M+ lines of code

How it works :
https ://www.openstack.org/software/roadmap/

A more detailed view of services




- Command-line interfaces (nova, neutron, swift, etc)
- Cloud Management Tools (Rightscale, Enstratius, etc)
- GUI tools (Dashboard, Cyberduck, iPhone client, etc)
Internet
OpenStack Object API
OpenStack Image API
HTTP(S)
OpenStack Compute APi
VNC/VMRC
OpenStack Block Storage API
OpenStack Networking API
HTTP(S)
OpenStack Dashboard
Horizon
OpenStack Image API
OpenStack Compute API/ Admin API
Amazon Web Services EC2 API
OpenStack Block Storage API
OpenStack Networking API
OpenStack Image API
swift-proxy
memcached
account
container
object
account database
container database
object database
OpenStack Object Store
glance-api
glance-registry
glance database
OpenStack Image Service
nova-api (OS, EC2, Admin)
nova-console
nova-compute
nova-cert/ objectstore
hypervisor
nova database
Queue
AMQP
AMQP
nova-conductor
nova-consoleauth
nova-scheduler
OpenStack Compute
cinder-api
cinder-volume
volume provider
nova database
cinder-scheduler
OpenStack Block Storage
neutron-server
neutron agents
Queue
nuetron plugin(s)
network provider
neutron database
OpenStack Networking
BY SA
http://www.solinea.com
OpenStack Identity API
keystone (service and admin APIs)
OpenStack Identity Service
token backend
catalog backend
policy backend
identity backend
OpenStack Identity API
OpenStack Identity API
OpenStack Identity API

Avancement

1 Basic Elements of a Datacenter

2 Case Studies

3 Dealing with Concerns

4 Software Infrastructure

5 Conclusion

A 20-sec conclusion

A new approach of delivering services

A lot of technologies, pushed to their limits