Department of Computer Science
University of Cyprus

EPL646 – Advanced Topics in Databases

Lecture 14

Big Data Management IV:

Big-data Infrastructures (Background, IO, From NFS to HFDS)

Chapter 14-15: Abideboul et. Al.

Demetris Zeinalipour

http://www.cs.ucy.ac.cy/~dzeina/courses/epl646

Lecture Outline

- Introduction to Cloud Computing
  - Typical Datacenters, Cloud Stack and Buzzwords, Public / Private Clouds, Utility Computing, Killer Apps, Economic Model
- Distributed System Basics
  - I/O Performance
  - Replication Strategies
  - The Hadoop Project (Core, HDFS, Map-Reduce, HBase, HIVE)
  - The Hadoop Distributed File System (HDFS)
  - HDFS vs. NFS (Network File System)
  - HDFS Example Deployments (Yahoo, Facebook)

Cloud Computing

- Different definitions for "Cloud Computing" exist
  - http://tech.slashdot.org/article.pl?sid=08/07/17/2117221
- Common ground of many definitions
  - processing power, storage and software are **commodities** that are readily available from large infrastructure

    αγαθά
  - **service-based view**: "everything as a service (*aaS)", where only "Software as a Service (SaaS)" has a precise and agreed-upon definition
  - utility computing: **pay-as-you-go** model

Google's Datacenter in Oregon

Microsoft Azure in Chicago

Cloud Computing (Datacenters)

Example: data centers

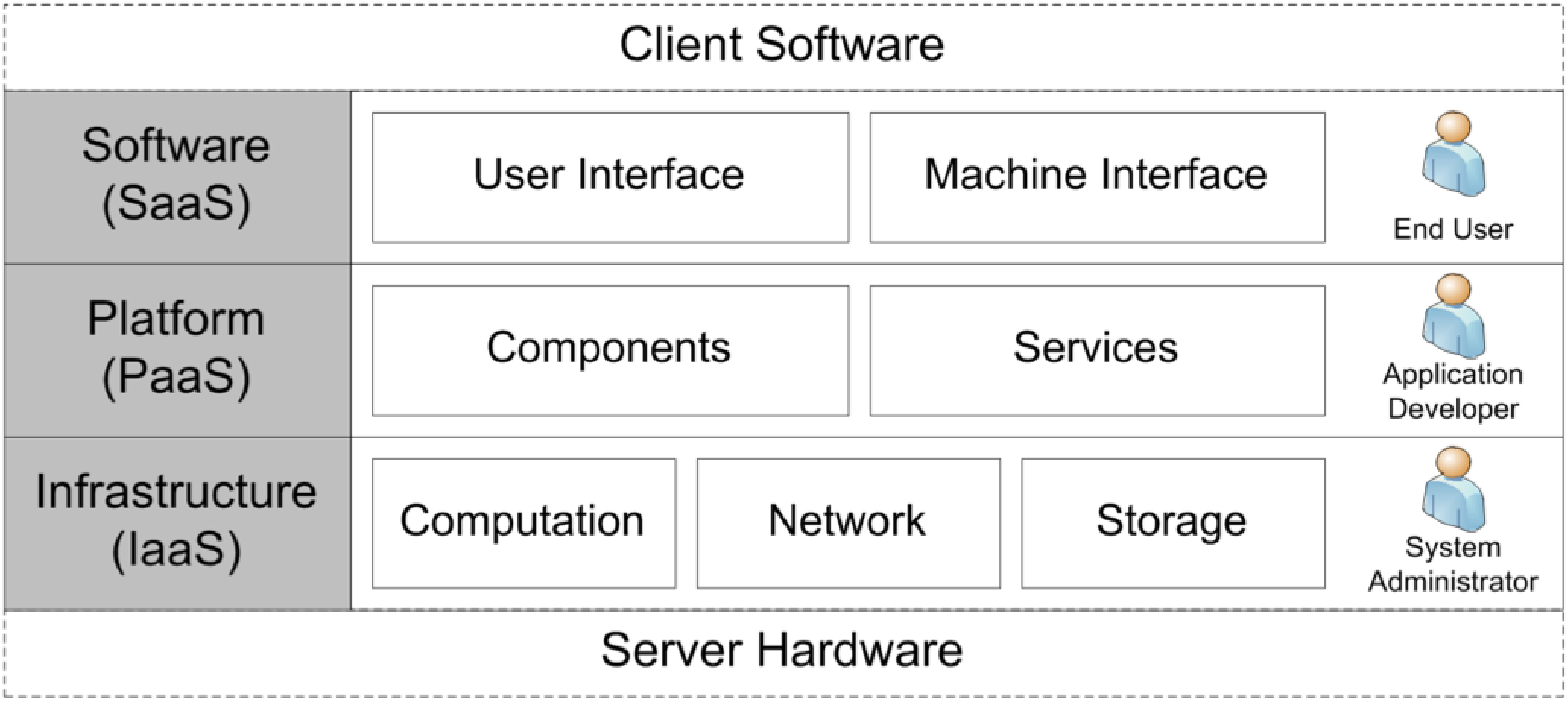
Datacenter

Typical setting of a Google data center.

1. ≈ 40 servers per rack;
2. ≈ 150 racks per data center (cluster);
3. ≈ 6,000 servers per data center;
4. how many clusters? Google's secret, and constantly evolving . . .

Rough estimate: 150-200 data centers? 1,000,000 servers?

Cloud Computing (Cloud Stack)



Source: Wikipedia (http://www.wikipedia.org)

SaaS Examples: Google Apps, Quickbooks Online and Salesforce.com.

PaaS Examples: Amazon Elastic Beanstalk, Heroku, EngineYard, Google App Engine, and Microsoft Azure.

IaaS Examples: Amazon CloudFormation (and underlying services such as Amazon EC2), Rackspace Cloud, Google Compute Engine, and RightScale.

Cloud Computing (Public vs. Private)

- Term **cloud computing** usually refers to both
  - **SaaS**: applications delivered over the Internet as services
  - **The Cloud**: data center hardware and systems software
- Public clouds
  - available in a **pay-as-you-go** manner to the public
  - service being sold is **utility computing**
  - Amazon Web Service, Microsoft Azure, Google AppEngine
- Private clouds
  - internal data centers of businesses or organizations
  - normally not included under **cloud computing**

Based on: "Above the Clouds: A Berkeley View of Cloud Computing", RAD Lab, UC Berkeley

Cloud Computing
(Utility Computing – Υπ. Ωφελείας)

NewSQL-as-a-Service
To Amazon RDS* (Relational Database Service)

Pay by the hour your DB Instance runs.

US – N. Virginia | US – N. California | EU – Ireland | APAC – Singapore

DB Instance Class	Price Per Hour
Small DB Instance	$0.11
Large DB Instance	$0.44
Extra Large DB Instance	$0.88
Double Extra Large DB Instance	$1.55
Quadruple Extra Large DB Instance	$3.10

963$ / year

27,165 $ / year

*(*essentially MySQL running on Amazon EC2 – Elastic Computing Cloud)*

DB Instance Classes

Amazon RDS currently supports five DB Instance Classes:

- Small DB Instance: 1.7 GB memory, 1 ECU (1 virtual core with 1 ECU), 64-bit platform, Moderate I/O Capacity
- Large DB Instance: 7.5 GB memory, 4 ECUs (2 virtual cores with 2 ECUs each), 64-bit platform, High I/O Capacity
- Extra Large DB Instance: 15 GB of memory, 8 ECUs (4 virtual cores with 2 ECUs each), 64-bit platform, High I/O Capacity
- Double Extra Large DB Instance: 34 GB of memory, 13 ECUs (4 virtual cores with 3,25 ECUs each), 64-bit platform, High I/O Capacity
- Quadruple Extra Large DB Instance: 68 GB of memory, 26 ECUs (8 virtual cores with 3.25 ECUs each), 64-bit platform, High I/O Capacity

For each DB Instance class, RDS provides you with the ability to select from 5GB to 1TB of associated storage capacity.
One ECU provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor.

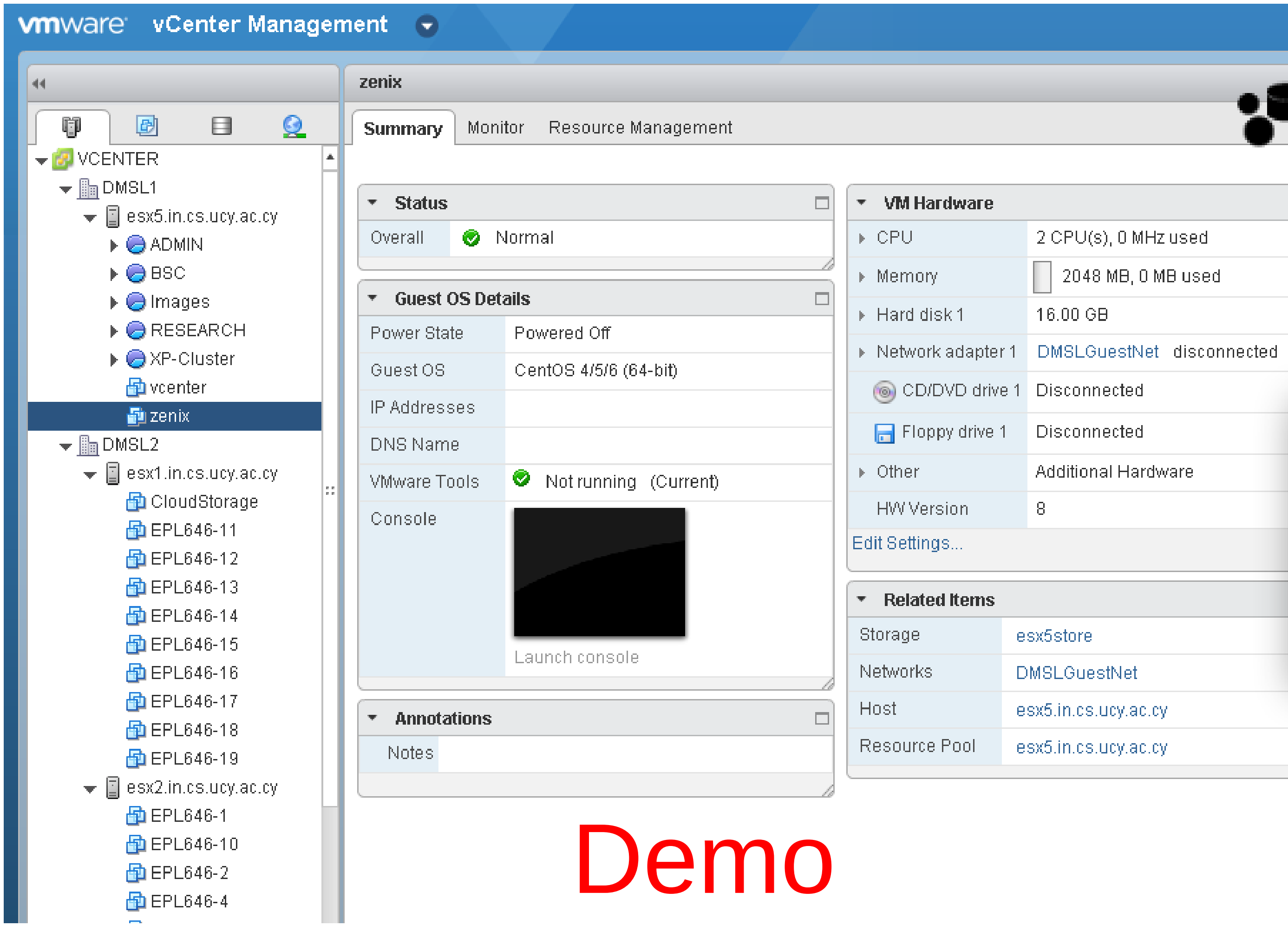
Cloud Computing
(Utility Computing – Υπ. Ωφελείας)

- Illusion of infinite computing resources
  - available on demand
  - no need for users to plan ahead for provisioning
- No up-front cost or commitment by users
  - companies can start small
  - increase resources only when there is an increase in need
- Pay for use on short-term basis as needed
  - processors by the hour and storage by the day
  - release them as needed, reward conservation

Cloud Computing
(Virtualization - Τεχνολογία εικονικών συστημάτων)

- Virtual resources abstract from physical resources
  - hardware platform, software, memory, storage, network
  - fine-granular, lightweight, flexible and dynamic
- Relevance to cloud computing
  - centralize and ease administrative tasks
  - improve scalability and work loads
  - increase stability and fault-tolerance
  - provide standardized, homogenous computing platform through hardware virtualization, i.e. **virtual machines**

Our IaaS Private Cloud



Demo

Cloud Computing (Economic Model)

Resource	Cost in Medium Data Center	Cost in Very Large Data Center	Ratio
Network	$95/Mbps/month	$13/Mbps/month	7.1x
Storage	$2.20/GB/month	$0.40/GB/month	5.7x
Administration	≈140 servers/admin	>1000 servers/admin	7.1x

Source: James Hamilton (http://perspectives.mvdirona.com)

- Cloud computing is 5-7x cheaper than traditional in-house computing
- Added benefits
  - utilize off-peak capacity (Amazon)
  - sell .NET tools (Microsoft)
  - reuse existing infrastructure (Google)

Cloud Computing (Killer Apps: OLTP/OLAP)

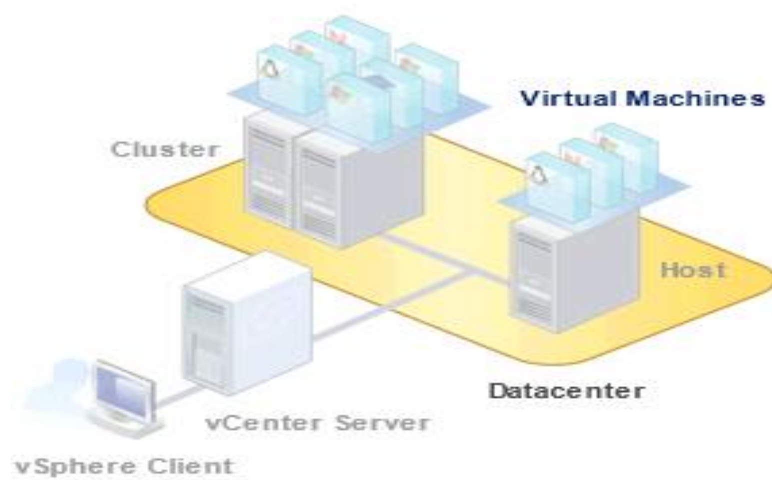
- Data management applications are potential candidates for deployment in the cloud
  - **industry:** enterprise database system have significant up-front cost that includes both hardware and software costs
  - **academia:** manage, process and share mass-produced data in the cloud
- Many "Cloud Killer Apps" are in fact data-intensive
  - Batch Processing as with map/reduce
  - Online Transaction Processing (OLTP) as in automated business applications
  - Offline Analytical Processing (OLAP) as in data mining or machine learning

Cloud Computing (Killer Apps: eScience)

- Old model
  - “Query the world”
  - data acquisition coupled to a specific hypothesis
- New model
  - “Download the world”
  - data acquired en masse, in support of many hypotheses
- E-science examples
  - astronomy: high-resolution, high-frequency sky surveys, …
  - oceanography: high-resolution models, cheap sensors, satellites, …
  - biology: lab automation, high-throughput sequencing, …

Distributed Systems Basics (I/O Performance)

*Each Disk: 2TB 7,200 rpm **6Gb SAS (Serial Attach. SCSI)** 3.5" HDD*
In RAID-5 configuration
[10Gbps FCoE also available]



Distributed Systems Basics (I/O Performance)

Performance

Type	Latency	Bandwidth (throughput)
Disk	$\approx 5 \times 10^{-3}$s (5 millisec.);	At best 100 MB/s
LAN	$\approx 1 - 2 \times 10^{-3}$s (1-2 millisec.);	$\approx$ 1GB/s (single rack); $\approx$ 100MB/s (switched);
Internet	Highly variable. Typ. 10-100 ms.;	Highly variable. Typ. a few MB/s.;

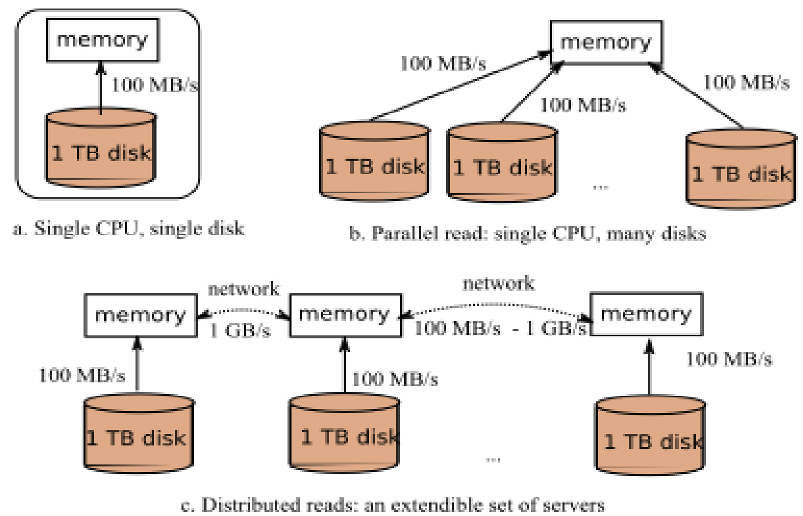
Bottom line (1): it is approx. one order of magnitude faster to exchange main memory data between 2 machines in a data center, that to read on the disk.

Bottom line (2): exchanging through the Internet is slow and unreliable with respect to LANs.

Distributed Systems Basics (I/O Performance)

Distribution, why?

Sequential access. It takes 166 minutes (more than 2 hours and a half) to read a 1 TB disk.
Parallel access. With 100 disks, assuming that the disks work in parallel and sequentially: about 1mn 30s.
Distributed access. With 100 computers, each disposing of its own local disk: each CPU processes its own dataset. **(Similar to Parallel but more scalable)**



Scalability

The latter solution is *scalable*, by adding new computing resources.

Distributed Systems Basics (I/O Performance)

What you should remember: performance of data-centric distr. systems

1. disk transfer rate is a bottleneck for large scale data management; parallelization and distribution of the data on many machines is a means to eliminate this bottleneck;
2. *write once, read many*: a distributed storage system is appropriate for large files that are written once and then repeatedly scanned;
3. *data locality*: bandwidth is a scarce resource, and program should be "pushed" near the data they must access to.

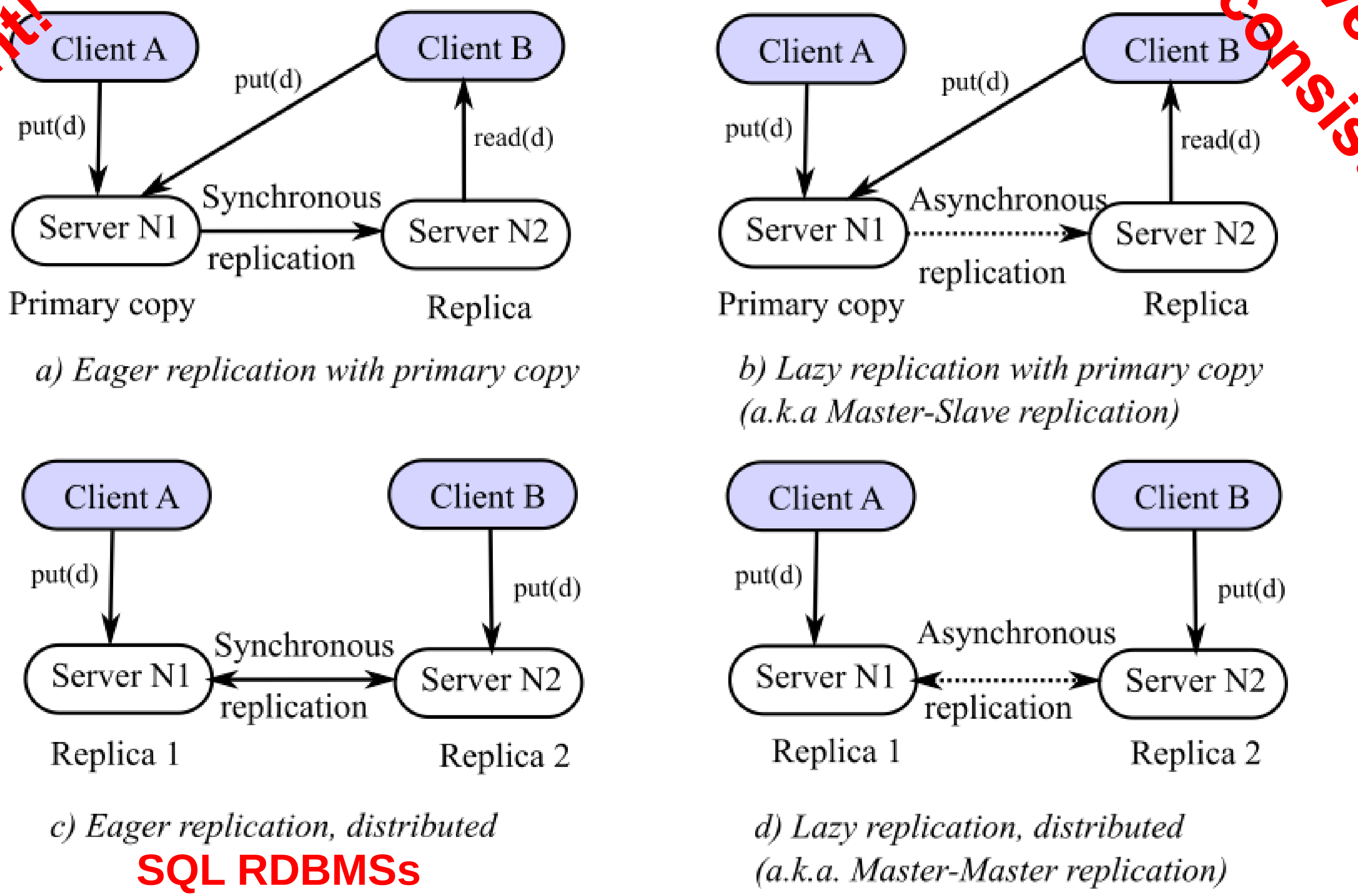
Distributed Systems Basics (Replication Strategies)

Some illustrative scenarios

Strongly consistent!

NOSQL DBMSs

Eventually consistent!



SQL RDBMSs

Distributed Systems Basics (Replication Strategies)

Consistency management in distr. systems

Consistency: essentially, ensures that the system faithfully reflects the actions of a user.

- Strong consistency (ACID properties) – requires a (slow) synchronous replication, and possibly heavy locking mechanisms.
- Weak consistency – accept to serve some requests with outdated data.
- Eventual consistency – same as before, but the system is guaranteed to converge towards a consistent state based on the last version.

In a system that is not eventually consistent, conflicts occur and the application must take care of data reconciliation: given the two conflicting copies, determine the new current one. (Recall conflict resolution in CouchDB)

Standard RDBMS favor consistency over availability – one of the reasons (?) of the 'NoSQL' trend.

Terminology (MR => HADOOP => HBASE)

- ***Map-Reduce:*** *a* ***programming model for processing*** *large data sets.*
  - *Invented by Google! "MapReduce: Simplified Data Processing on Large Clusters, Jeffrey Dean and Sanjay Ghemawat, OSDI'04: Sixth Symposium on Operating System Design and Implementation,San Francisco, CA, December, 2004."*
  - *Can be implemented in any language (recall javascript Map-Reduce we used in the context of CouchDB).*
- ***Hadoop:*** *Apache's open-source* ***software framework*** *that supports* ***data-intensive distributed applications***
  - *Derived from Google's MapReduce + Google File System (GFS) papers.*
  - *Enables applications to work with thousands of computation-independent computers and petabytes of data.*
  - Download: http://hadoop.apache.org/

Terminology
(MR => HADOOP => HBASE)

- ***Hadoop Project Modules:***
  - ***Hadoop Common:*** *The common utilities that support the other Hadoop modules.*
  - ***Hadoop Distributed File System (HDFS™):*** *A distributed file system that provides high-throughput access to application data.*
  - ***Hadoop YARN (Yet Another Resource Negotiator):*** *A framework for job scheduling and cluster resource management.*
  - ***Hadoop MapReduce (MapReduce v2.0):*** *A YARN-based system for parallel processing of large data sets.* ***(Next Lectures)***
- ***Other Hadoop-related projects at Apache include:***
  - *Ambari: Dashboard management system for Hadoop.*
  - *Avro™: A data serialization system.*
  - *Cassandra™: A scalable multi-master database with no single points of failure.*
  - *Chukwa™: A data collection system for managing large distributed systems.*
  - ***HBase™ (Hadoop Database): A scalable, distributed database that supports structured data storage for large tables. (Next Lectures)***
  - *Hive™: A data warehouse infrastructure that provides data summarization and ad hoc querying.*
  - *Mahout™: A Scalable machine learning and data mining library.*
  - ***Pig™: A high-level data-flow language and execution framework for parallel computation. (Next Lectures)***
  - *ZooKeeper™: A high-performance coordination service for distributed applications.*

Large-Scale File Systems (GFS => HDFS)

History and development of GFS

Problem: if nodes can fail, how can we store data persistently?
Answer: Distributed File System (global file namespace)

Google File System, a paper published in 2003 by Google Labs at OSDI.

The Google File System, Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, 19th ACM Symposium on Operating Systems Principles, Lake George, NY, October, 2003.

Explains the design and architecture of a distributed system apt at serving very large data files; internally used by Google for storing documents collected from the Web.

Open Source versions have been developed at once: Hadoop File System (HDFS), and Kosmos File System (KFS).

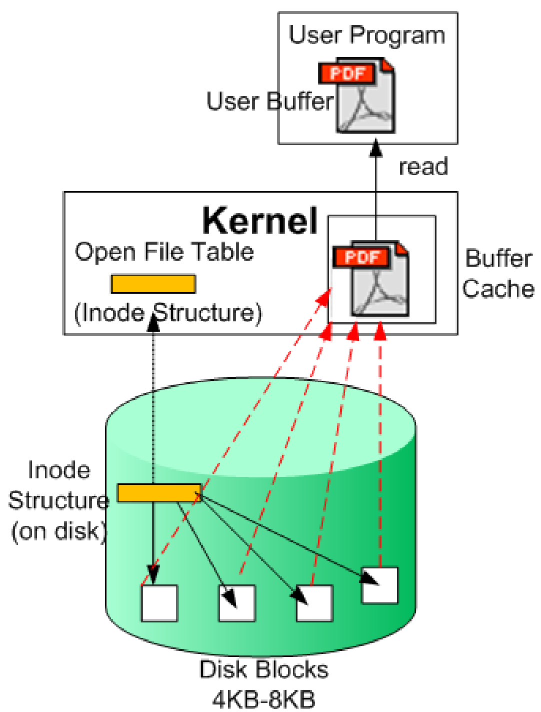
Network File Systems (NFS=>GFS=>HDFS)

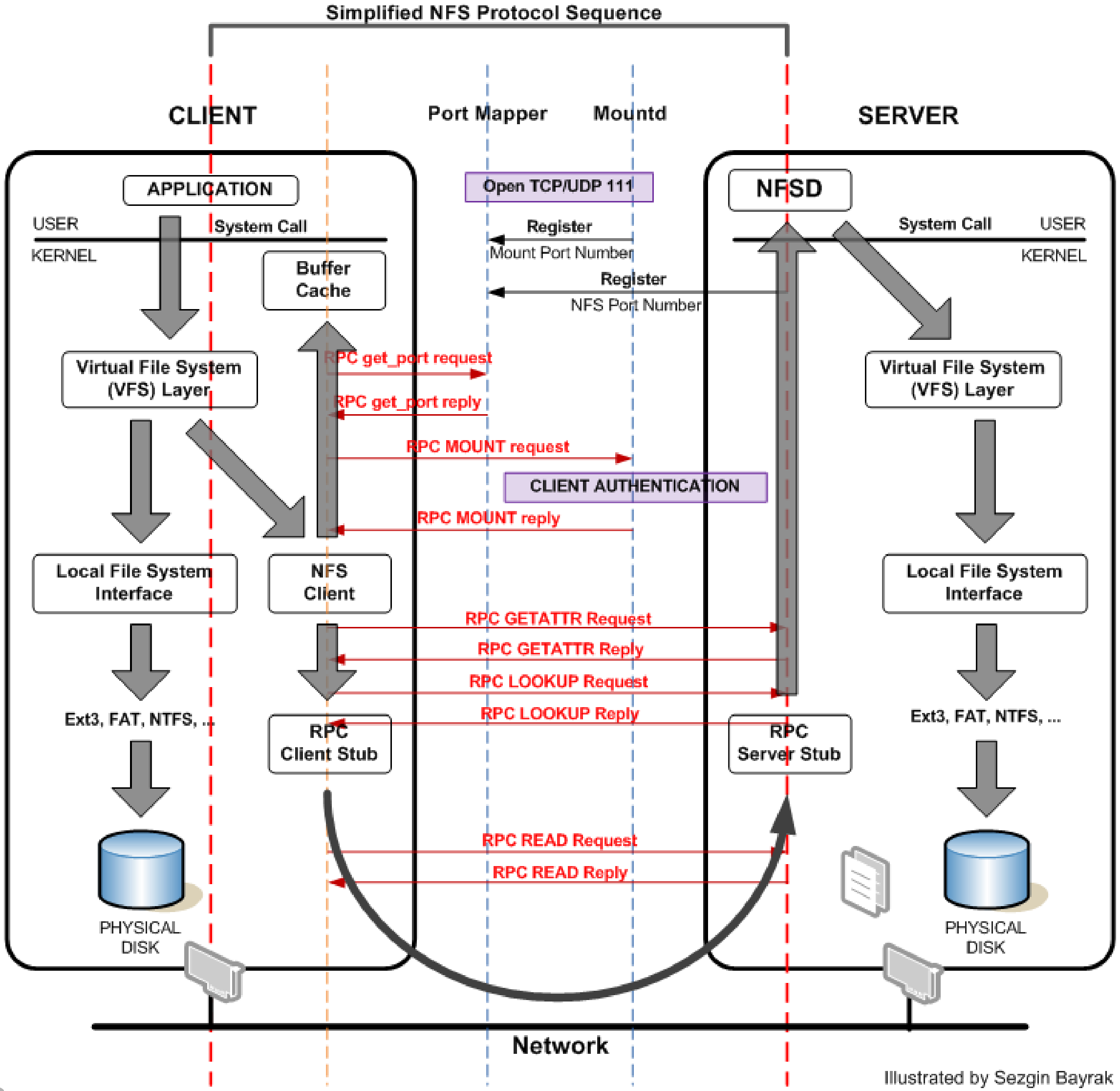
UNIX NFS (Network File System):
nfsd (deamon) mounts remote folders to a UNIX host (/etc/fstab).

```
1)dzeina@evterpi> df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/VGSystem-LVroot
                       2031440    950588    975996  50% /
/dev/mapper/VGSystem-LVtmp
                       1015704     34104    929172   4% /tmp
/dev/mapper/VGSystem-LVvar
                       3301112   2002692   1128044  64% /var
/dev/mapper/VGSystem-LVopt
                        507748    112975    368559  24% /opt
/dev/mapper/VGSystem-LVusr
                       4570784   2777372   1557492  65% /usr
/dev/mapper/VGSystem-LVusrLocal
                        507748     65526    416008  14% /usr/local
/dev/sda1               101086     40747     55120  43% /boot
tmpfs                  2023380         0   2023380   0% /dev/shm
/dev/mapper/VGData-LVdata
                      58120148  43379612  11744012  79% /sys-data
csfs4.cs.ucy.ac.cy:/vol/home/research
                     576716800 508312448  68404352  89% /home/research
csfs1.cs.ucy.ac.cy:/home/faculty
                     629145600 524337472 104808128  84% /home/faculty
csfs5.cs.ucy.ac.cy:/home/projects
                      51606528  20917760  28067328  43% /home/projects
```

Network File Systems (NFS=>GFS=>HDFS)

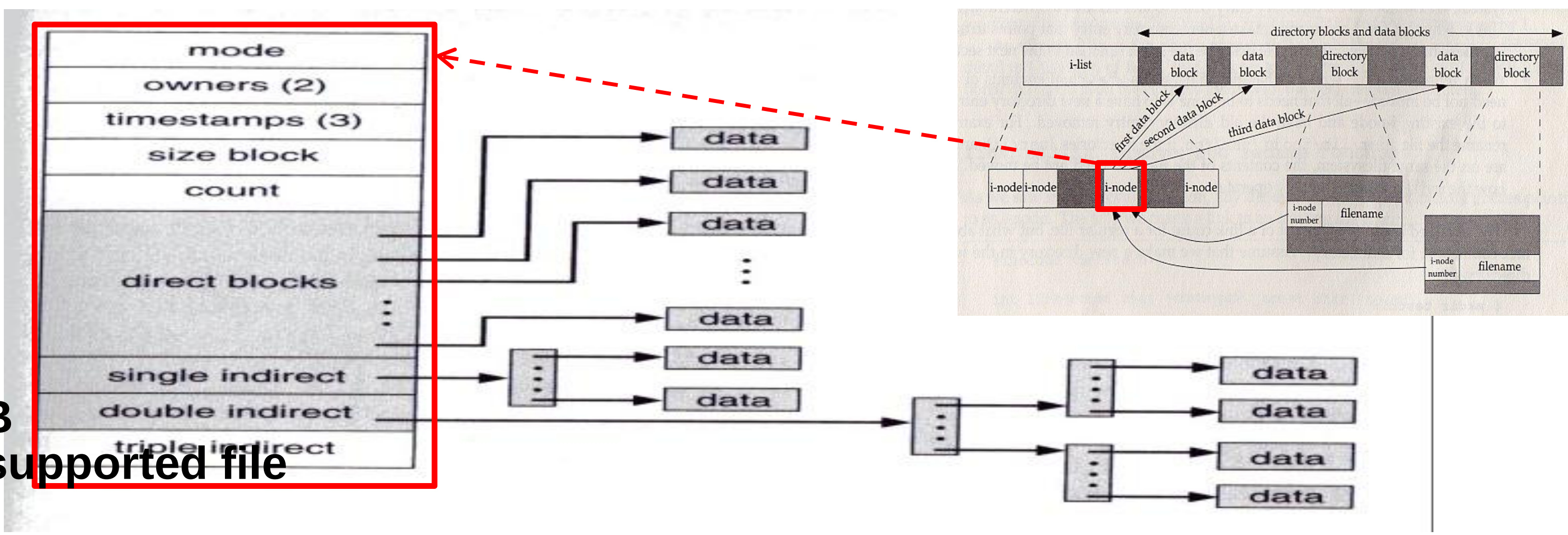
NFS uses a Client/Server Architecture that is a single point of failure by default.





iNode Δομές στο UNIX (Επανάληψη)

- **Τι γίνεται εάν ένα αρχείο έχει πολλά blocks**;
- Υπάρχει **αρκετός χώρος** για να αποθηκευτούν **όλα τα i-nodes** των blocks που συσχετίζονται με το αρχείο;
- Το Υποσύστημα Αρχείων χρησιμοποιεί ένα **ιεραρχικό σχήμα** το οποίο αποτελείται από δένδρα δεικτών βάθους 0 (direct, αυτό το οποίο είδαμε ήδη), 1 (single), 2 (double), και 3 (triple)



2KB block
<24KB
<1MB
<512MB
< max supported file size

Large-Scale File Systems (Hadoop File System - HDFS)

The problem

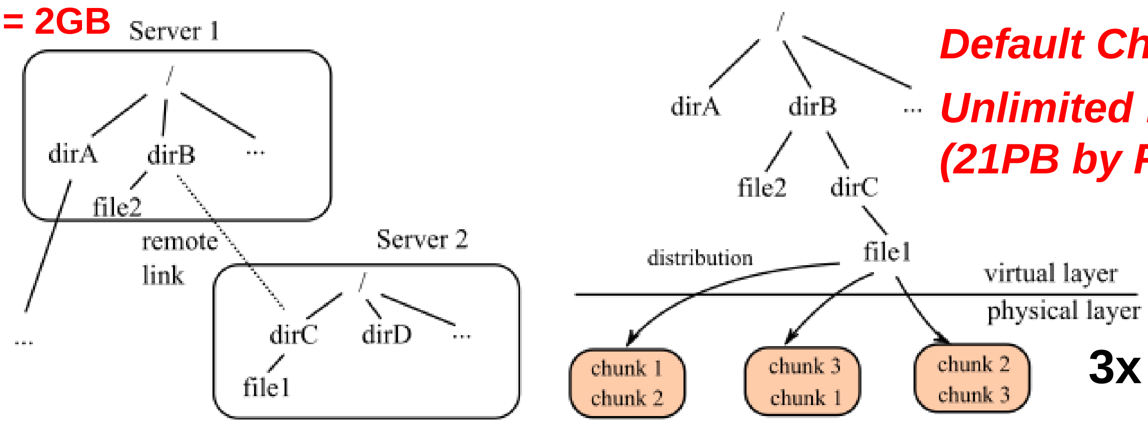
Why do we need a distributed file system in the first place?

Fact: standard NFS (left part) does not meet scalability requirements (what if *file1* gets really big?).

Chuck (Block) Size: 4KB-32KB
NFS File Size Limit = 2GB

Default Chunk Size: 64 MB

Unlimited File Size (21PB by Facebook)



A traditional network file system

A large scale distributed file system

3x replicas per chunk

Right part: GFS/HDFS storage, based on (i) a virtual file namespace, and (ii) partitioning of files in “chunks”.

Large-Scale File Systems (Hadoop File System - HDFS)

YAHOO!

2010

	Target	Deployed
Capacity	10PB	14PB
Nodes	10,000	4000
Clients	100,000	15,000
Files	100,000,000	60,000,000

TABLE 1: TARGETS FOR HDFS VS. ACTUALLY DEPLOYED VALUES AS OF 2009

facebook

2010

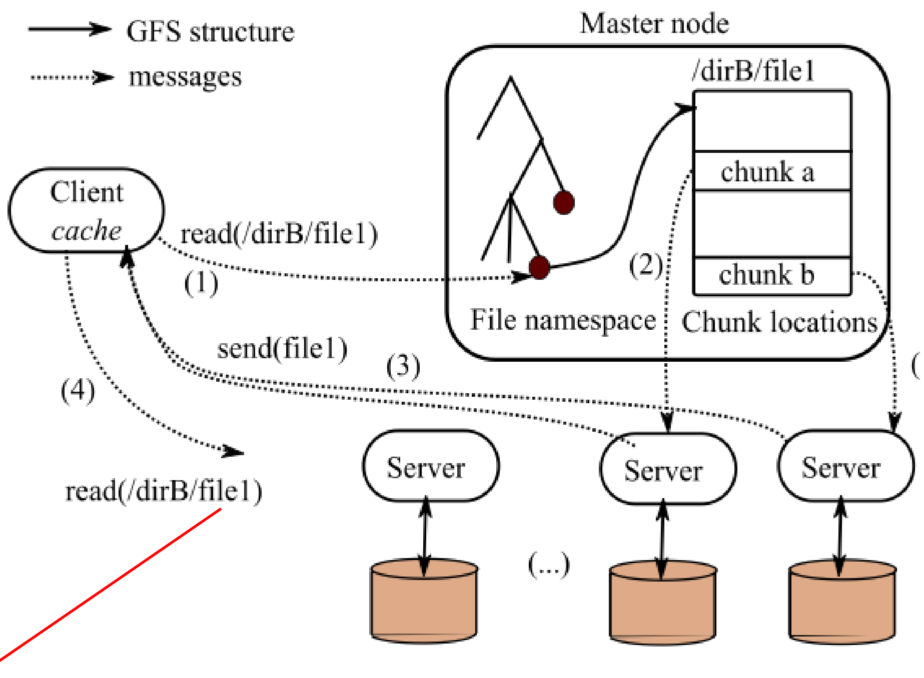
- 21 PB of storage in a single HDFS cluster
- 2000 machines
- 12 TB per machine (a few machines have 24 TB each)
- 1200 machines with 8 cores each + 800 machines with 16 cores each
- 32 GB of RAM per machine
- 15 map-reduce tasks per machine

HDFS scalability: the limits to growth
http://static.usenix.org/publications/login/2010-04/openpdfs/shvachko.pdf

Large-Scale File Systems (Hadoop File System - HDFS)

Architecture

A Master node performs administrative tasks, while servers store "chunks" and send them to Client nodes.



The Client maintains a cache with chunks locations, and directly communicates with servers.

Namespace lookup are fast (1 Master enough!) [1GB Metadata = 1PB Data]

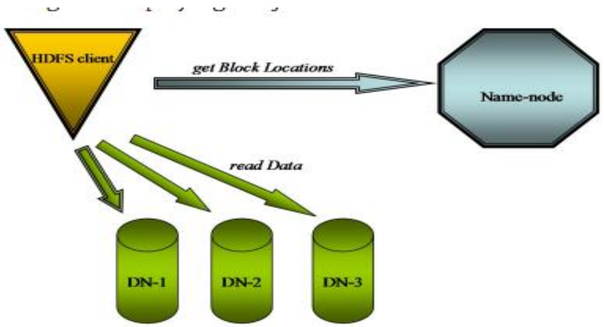
In **NFS Metadata + Transfers going** through same server => **Not Scalable**

HDFS designed for **unreliable hardware** (2-3 failures / 1000 nodes / day)

New Hardware: 3x more unreliable!!!

Large-Scale File Systems (Hadoop File System - HDFS)

Technical details



- The architecture works best for very large files (e.g., several Gigabytes), divided in large (64-128 MBs) chunks.
  $\Rightarrow$ this limits the metadata information served by the Master.
- Each server implements recovery and replication techniques (default: 3 replicas).
- (Availability) The Master sends heartbeat messages to servers, and initiates a replacement when a failure occurs.
- (Scalability) The Master is a potential single point of failure; its protection relies on distributed recovery techniques for all changes that affect the file namespace.

User Interface

- API
  - Java API
  - C language wrapper (*libhdfs*) for the Java API is also avaiable
- POSIX like command
  - hadoop dfs -mkdir /foodir
  - hadoop dfs -cat /foodir/myfile.txt
  - hadoop dfs -rm /foodir myfile.txt
- HDFS Admin
  - bin/hadoop dfsadmin –safemode
  - bin/hadoop dfsadmin –report
  - bin/hadoop dfsadmin -refreshNodes
- Web Interface
  - Ex: http://localhost:50070

NameNode '172.16.203.136:8020'

Started:	Sun May 17 11:52:41 CST 2009
Version:	0.20.0, r763504
Compiled:	Thu Apr 9 05:18:40 UTC 2009 by ndaley
Upgrades:	There are no upgrades in progress.

Browse the filesystem
Namenode Logs

Cluster Summary

55 files and directories, 52 blocks = 107 total. Heap Size is 4.94 MB / 198.5 MB (2%)

Configured Capacity	:	13.32 GB
DFS Used	:	881.47 KB
Non DFS Used	:	6.88 GB
DFS Remaining	:	6.44 GB
DFS Used%	:	0.01 %
DFS Remaining%	:	48.35 %
Live Nodes	:	2
Dead Nodes	:	0

Live Datanodes : 2

Node	Last Contact	Admin State	Configured Capacity (GB)	Used (GB)	Non DFS Used (GB)	Remaining (GB)	Used (%)	Used (%)
172.16.203.132	2	In Service	6.66	0	3.06	3.6	0.01	
172.16.203.133	2	In Service	6.66	0	3.82	2.84	0.01	

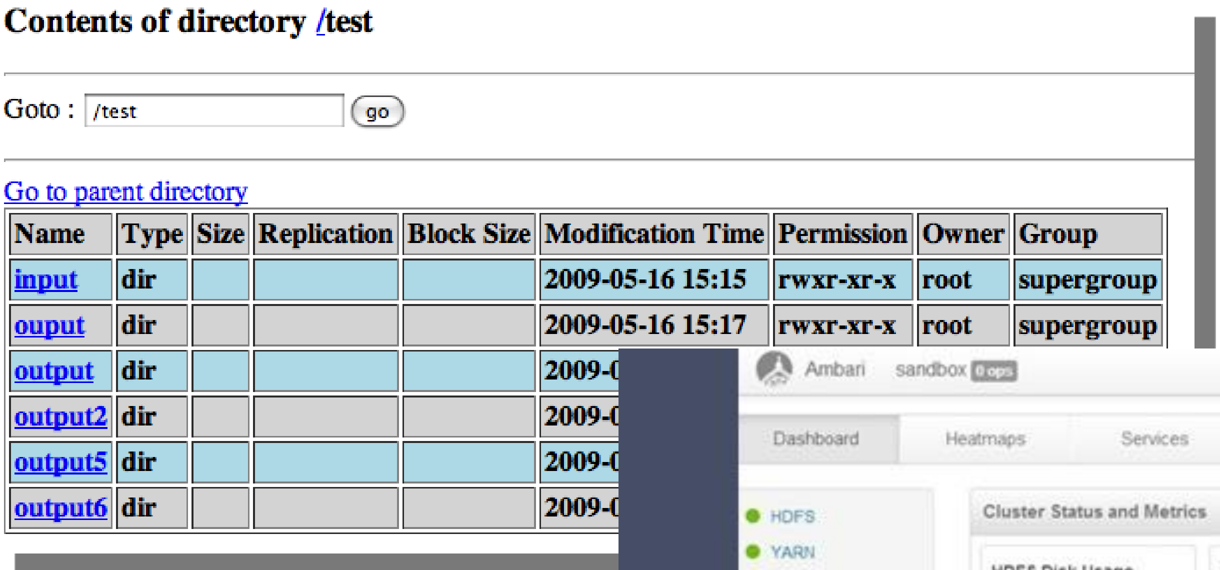
NameNode Storage:

Storage Directory	Type	State
/tmp/namenode	IMAGE_AND_EDITS	Active

Web Interface

(http://localhost:50070)

Browse the file system

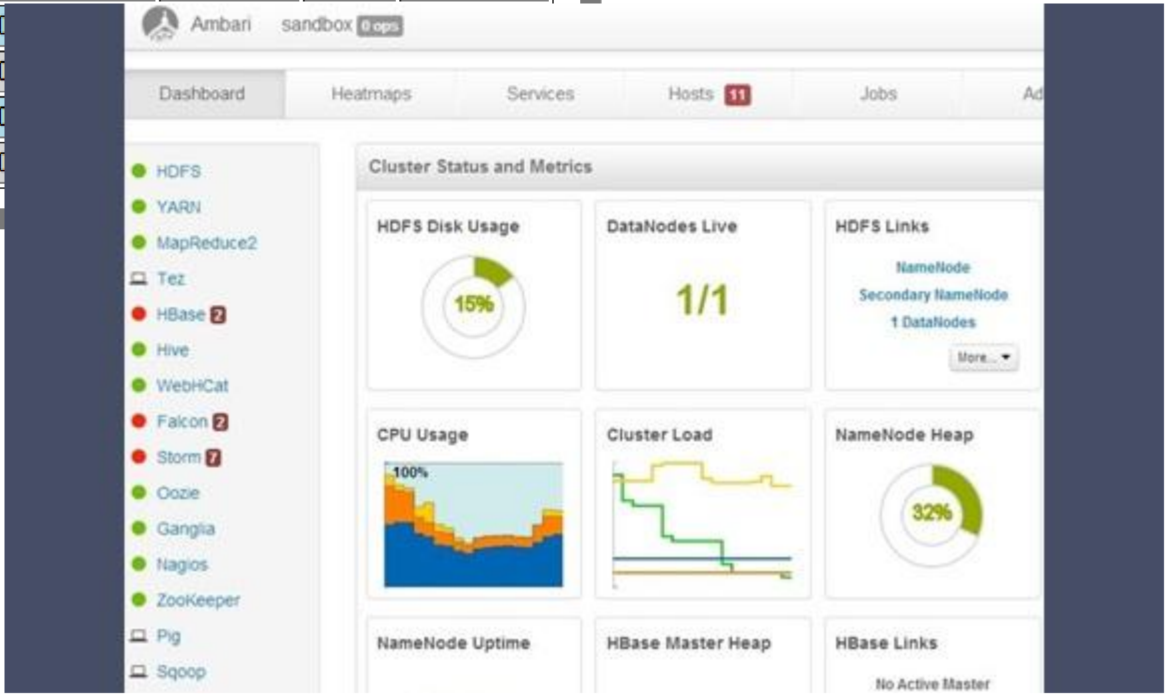


Contents of directory /test

Goto : /test go

Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
input	dir				2009-05-16 15:15	rwxr-xr-x	root	supergroup
ouput	dir				2009-05-16 15:17	rwxr-xr-x	root	supergroup
output	dir				2009-0			
output2	dir				2009-0			
output5	dir				2009-0			
output6	dir				2009-0			



POSIX Like command

```
Usage: hadoop fs [generic options]
        [-appendToFile <localsrc> ... <dst>]
        [-cat [-ignoreCrc] <src> ...]
        [-checksum <src> ...]
        [-chgrp [-R] GROUP PATH...]
        [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
        [-chown [-R] [OWNER][:[GROUP]] PATH...]
        [-copyFromLocal [-f] [-p] [-l] <localsrc> ... <dst>]
        [-copyToLocal [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
        [-count [-q] [-h] <path> ...]
        [-cp [-f] [-p | -p[topax]] <src> ... <dst>]
        [-createSnapshot <snapshotDir> [<snapshotName>]]
        [-deleteSnapshot <snapshotDir> <snapshotName>]
        [-df [-h] [<path> ...]]
        [-du [-s] [-h] <path> ...]
        [-expunge]
        [-find <path> ... <expression> ...]
        [-get [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
        [-getfacl [-R] <path>]
        [-getfattr [-R] {-n name | -d} [-e en] <path>]
        [-getmerge [-nl] <src> <localdst>]
        [-help [cmd ...]]
        [-ls [-d] [-h] [-R] [<path> ...]]
        [-mkdir [-p] <path> ...]
        [-moveFromLocal <localsrc> ... <dst>]
        [-moveToLocal <src> <localdst>]
        [-mv <src> ... <dst>]
        [-put [-f] [-p] [-l] <localsrc> ... <dst>]
```

```
[root@172 bin]# ./hadoop dfs -mkdir /test/input10
[root@172 bin]# ./hadoop dfs -put ../conf/* /test/input10
[root@172 bin]# ./hadoop dfs -ls /test/input10/
Found 13 items
-rw-r--r--   3 root supergroup       6275 2009-05-17 12:21 /test/input10/capacity-scheduler.xml
-rw-r--r--   3 root supergroup        535 2009-05-17 12:21 /test/input10/configuration.xsl
-rw-r--r--   3 root supergroup        270 2009-05-17 12:21 /test/input10/core-site.xml
-rw-r--r--   3 root supergroup       2296 2009-05-17 12:21 /test/input10/hadoop-env.sh
-rw-r--r--   3 root supergroup       1245 2009-05-17 12:21 /test/input10/hadoop-metrics.properties
-rw-r--r--   3 root supergroup       4190 2009-05-17 12:21 /test/input10/hadoop-policy.xml
-rw-r--r--   3 root supergroup        259 2009-05-17 12:21 /test/input10/hdfs-site.xml
-rw-r--r--   3 root supergroup       2815 2009-05-17 12:21 /test/input10/log4j.properties
-rw-r--r--   3 root supergroup        272 2009-05-17 12:21 /test/input10/mapred-site.xml
-rw-r--r--   3 root supergroup         10 2009-05-17 12:21 /test/input10/masters
-rw-r--r--   3 root supergroup         30 2009-05-17 12:21 /test/input10/slaves
-rw-r--r--   3 root supergroup       1243 2009-05-17 12:21 /test/input10/ssl-client.xml.example
-rw-r--r--   3 root supergroup       1195 2009-05-17 12:21 /test/input10/ssl-server.xml.example
[root@172 bin]# ./hadoop dfs -tail /test/input10/masters
localhost
[root@172 bin]#
```

Java API

- Latest API
  - http://hadoop.apache.org/core/docs/current/api/

```
URI uri = new URI("hdfs://namenode/");
FileSystem fs = FileSystem.get(uri, new Configuration());
Path file = new Path("answer");

DataOutputStream out = fs.create(file);
out.writeInt(42);
out.close();

DataInputStream in = fs.open(file);
System.out.println(in.readInt());
in.close();

fs.delete(file);
```