MONASH University
Information Technology

FIT3142 Distributed Computing

Topic 5: Grid, Clouds, Distributed Storage / Hadoop / Spark

Dr Carlo Kopp
Clayton School of Information Technology
Monash University

Why Study Grids, Clouds, and Distributed Storage?

- Study Grids, Clouds, and Distributed Storage schemes are now becoming more commonly used, therefore must be understood as each has unique middleware and features;
- Foundation knowledge: Because these distributed systems are now widely used, understanding them is essential;
- Foundation knowledge: These schemes all employ unique abstractions and have unique behaviour;
- Practical skills: When coding distributed applications you will likely have to run the code on a grid, cloud or distributed storage system, so you need to understand their design;
- Practical skills: You may have to benchmark an application on a grid, cloud or distributed storage system, so understanding their middleware features and performance matters;

References/Reading (I)

1. Ch.1; Ch.3; Ch.4; Coulouris, Dollimore, Kindberg and Blair, *Distributed Systems: Concepts and Design*, Edition 5, © Addison-Wesley, 2012.
2. Pawel Plaszczak and Richard Wellner, Jr.; *Grid Computing*, Elsevier, 2005.
3. http://www.globus.org/
4. http://www.csse.monash.edu.au/~carlo/SYSTEMS/IPC-Introduction-0595.htm
5. http://www.csse.monash.edu.au/~carlo/SYSTEMS/RPC-Intro-0796.htm
6. http://www.csse.monash.edu.au/~carlo/WALNUT/msc.thesis.ck.pdf
7. http://www.javacoffeebreak.com/articles/rmi_corba/index.html
8. http://java.sun.com/javase/technologies/core/corba/index.jsp
9. http://www.cs.utexas.edu/~wcook/Drafts/2006/WSvsDO.pdf
10. http://www.eg.bucknell.edu/~cs379/DistributedSystems/rmi_tut.html

References/Reading (II)

11. *HDFS Architecture Guide*, Apache Hadoop docs, URI: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html
12. Robert Chansler et al, *The Hadoop Distributed File System*, The Architecture of Open Source Applications, URI: http://www.aosabook.org/en/hdfs.html
13. *MapReduce Tutorial*, Apache Hadoop docs, URI: https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html
14. *What is the Hadoop Distributed File System (HDFS)?* IBM Hadoop tutorial, URI: http://www-01.ibm.com/software/data/infosphere/hadoop/hdfs/
15. M. Tim Jones, Consultant Engineer, *Spark, an alternative for fast data analytics*, IBM Bluemix series, URI: http://www.ibm.com/developerworks/library/os-spark/
16. *Apache Mesos distributed systems kernel*, URI: http://mesos.apache.org/
17. *Apache Mesos Architecture*, URI: http://mesos.apache.org/documentation/latest/mesos-architecture/

Distributed Computing Schemes

- Numerous schemes in wide use today, many up to decades old by design:

1. *Client / Server* – socket applications, file serving, web serving;
2. *Parallel Processing with homogenous workloads* – clusters running MPI, Grids running MPI or other parallel applications;
3. *Parallel Processing with heterogeneous workloads* – clusters for web serving, Grids with some mixed applications; Clouds with mixed workloads;

- Specific applications produce specific workloads, and the type of environment should be matched to the workload;
- *Homogenous* = processes mostly running same application;
- *Heterogeneous* = processes may run any applications;
- *There is no "panacea" since applications behave differently!*

Parallel Processing…

- Parallel applications led to parallel computing:

1. *Coarse-grained systems* – SMP (Symmetrical Multi Processing);
2. *Medium-grained systems* – Clusters;
3. *Fine-grained systems* – MPP (Massively Parallel Processing);

- The intention behind all of these schemes is to divide up large computational tasks across large numbers of processors;
- Performance improvements intended to be proportional to the number of processors, but limited by Amdahl's Law;
- These schemes were well established by the 1990s but with the exception of SMP, usage was mostly limited to specialised applications, mainly in scientific computing.

Parallel Processing Limitations

- **Problems with MPP:**

1. **Highly application specific;**
2. **Limited to "embarrassingly" parallel methods;**
3. **Expensive to administer, specialist skills required;**
4. **Costly due to the use of custom hardware;**

- **SMP is heavily dependent on the internal bus design used to interconnect the processors:**

1. **Difficult to scale-up due to bus limitations;**
2. **Low parallelism yields resulting from bus size constraints;**
3. **Provides transparent access to parallelism;**
4. **Costly due to the use of custom hardware for busses and processor modules;**
5. **Widely used in period server hosts (Sun, SGI, DEC, IBM);**

Clustered Computing

- **Cluster computing provides a cheaper alternative with free software components:**

1. **Parallel Virtual Machine (PVM) software;**
2. **Message Passing Interface (MPI) software;**
3. **Parametric Computing Toolsets (Nimrod/Enfuzion);**

- **Network Area Storage (NAS)**

1. **Network File System (NFS) – established client–server code;**
2. **Andrew File System (AFS) with greater fault-tolerance and resource utilisation;**

- **Distributed Storage Schemes**

1. **Hadoop**
2. **Spark**

- ***Clusters limited in size by building volume, cooling and power, and importantly, middleware limitations.***

Grids vs. Clouds

- The limitations of clusters led developers to the idea of "clustering clusters" to aggregate many more cores than could be run in a single cluster – this led to *Grids*;
- Most *Grids* have been built by joining multiple clusters, using common middleware, usualy *Globus / OpenGrid*;
- *Grids* have been mostly used in scientific computing, and usually run with homogenous workloads;
- *Clouds* evolved from large clusters used for web searching and retail applications, where the workloads were inherently heterogeneous – the ability to run large numbers of often different applications on shared resources was the priority;
- *Cloud* middleware is usually run on a cluster, but may also aggregate many clusters, as is done in a Grid;
- *Clouds* often run with VMs – "platform agnostic" model.

GRIDS

Grid Nomenclature: What is a/the "grid"?

- **The use of upper and lower cases**
  - *grid* for enterprise-wide grids
  - The *Grid* for the global grid, when that happens
  - Capital 'G' for *Grid computing* and *Grid technologies*
- **The linking of computers for sharing processing load forms a basic computer grid (named after the electrical power generation "grid model").**

Grid Definition

- Contemporary computer grids however extend this functionality by providing seamless (web-like) access to a variety of networked resources, including:

1. large data stores and information repositories
2. expensive instruments
3. high-speed links
4. sensors networks
5. multimedia services

- Grids are now well established in HPPC/HPC applications, mainly in scientific computing.

Grid Features

- Grid services can easily be discovered;
- Individuals and organisation would use these services in quite different ways:

1. Science, Remote-access: A scientist may be able to access data and devise an experiment requiring the use of a powerful particle accelerator with a common web browser;
2. Business, On-demand Computing: A business may shield itself from market fluctuations though a pay-by-the-use grid access;
3. Entertainment, Resource-sharing: A group of like-minded individuals may pool their computers to solve a problem or to entertain – gaming, file distribution;

- Grids have yet to penetrate commercial applications in any number at this time – competition with commercial "clouds".

A CERN Grid Definition

- **Explanation from CERN – the European Organisation for Nuclear Research**
  - Interconnecting computing capacity across multiple sites
  - Everyone get access to the capacity
  - Access when the users need it

Main Grid Application Areas

- **Grid Computing**
  - Virtualisation
  - On-demand provisioning of service (resource)
  - On-demand sharing of service (resource)
- **The utility computing model**
  - Ubiquitous digital market of services
- **Non-proprietary and interoperable frameworks are needed for these services.**

Grid Beginnings

- **Internet piloted by ARPANET, 1960s**
- **WWW concept floated in CERN, 1989**
- **Weather prediction in early 20th Century led to cell-based computations**
  - Large number of equations needed to be solved in record time
- **Finite element simulations, protein folding, financial portfolio design, earthquake engineering**
- **Era of supercomputers, 1970s – 1980s**

Grid Beginnings

- **Parallel computers, SMP and MPP, 1990s**
- **Embarrassingly parallel applications**
  - Image processing
  - Cryptography
  - Parametric computing
- **Difficult to parallelise applications**
  - Tightly coupled problems, equations, reservations, scheduling, modelling
  - Solved with Domain Decomposition
    - > Implicit or Explicit domain decomposition

Grid Security

- **Grid Security Infrastructure**
  - Authentication of users or services
  - Authorisation of communications
  - Management of user credentials and maintaining group membership information
- **Virtual Private Networks**
  - Secure connections over Internet
  - Virtualisation
  - Scalability

Globus (Grid) vs Java Distributed Applications

•Globus does not hide the architecture of its components •It can provide better utlisation of its underlying infrastructure •The applications require infrastructure level awareness and customisations, hence costlier • Better access to message passing libraries	•Java provides a virtual machine overlay which is standard over all of its participating components •Can not guarantee making use of all features of the underlying infrastructure •Provides fast and easy path to distributed application development

Grid Resources

- **Grids provide more than just CPU as the resource**
- **The resources may include**
  - Disk
  - Memory
  - Special architecture
  - User attached devices
  - Services

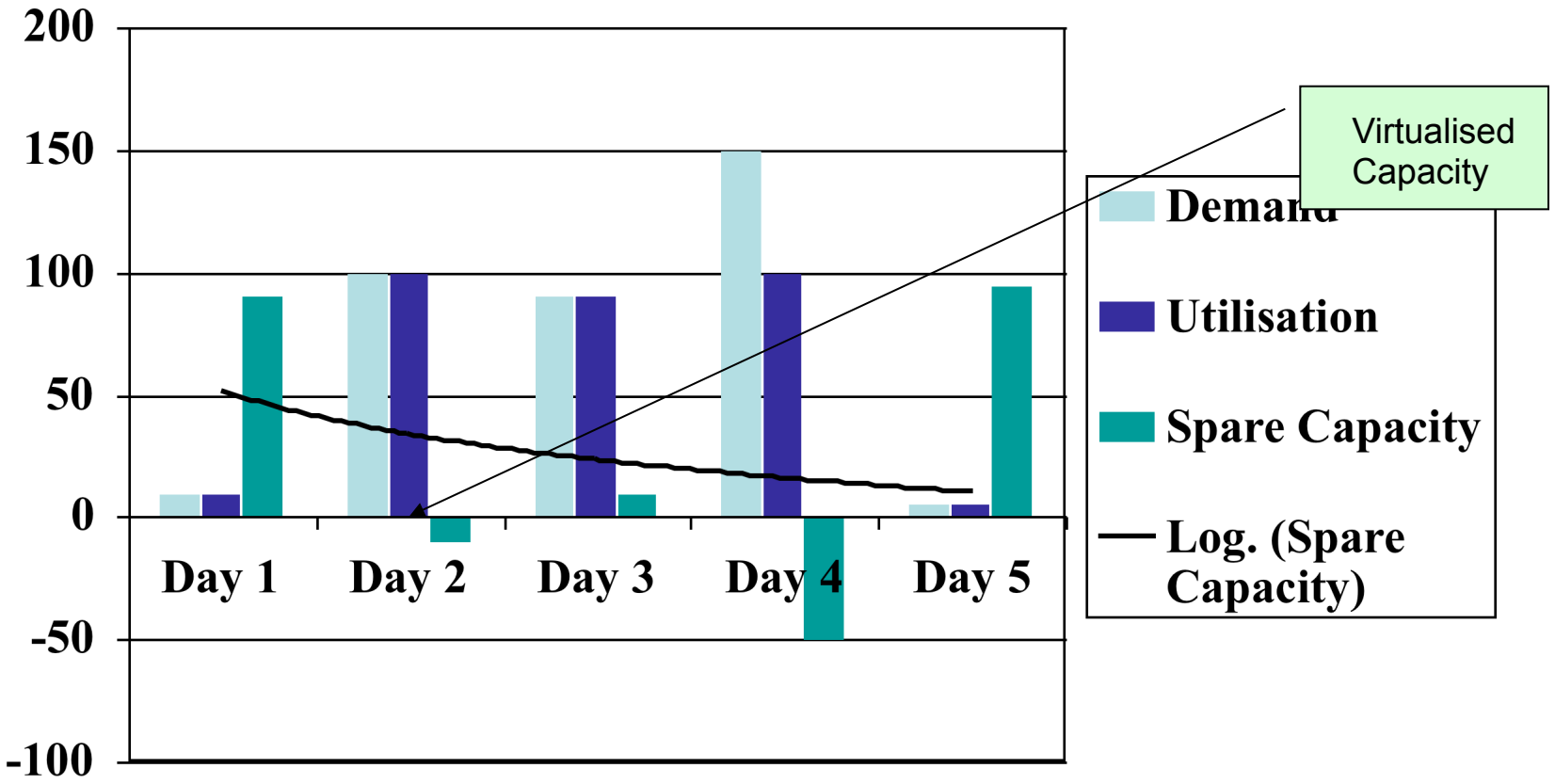
Grid Resource Characteristics

- **Platform - CPU architecture and OS**
  - Some applications are only available on certain platforms, notably windows
- **QoS – real-time applications**
  - Access-grid for real-time collaboration
- **Production ready infrastructure**
  - Highly reliable, scalable, and economical e.g. SAN, HSM for storage
- **Specialist applications**
  - Climate change research, genomic studies, financial forecasting
- **Communications, using open and interoperable protocols**
  - TCP/IP, SSH, OpenSSL, SOAP, RMI

Virtualisation

- Difficult to predict the demand for computing power;
- Demand can fluctuate over time – e.g. web servers;
- Hardware capital investment is static and entails recurring maintenance cost;
- Virtualisation allows making use of unutilised capacities within the organisation;
- It also allows bringing resources from outside within the utilisation framework;

Example: Grid Resource Virtualisation



CLOUDS

Cloud Computing (GSA)

- ***On-demand self-service*** **-- A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service's provider.**
- ***Broad network access*** **-- Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, laptops, and PDAs).**
- ***Resource pooling*** **-- The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand.**

Cloud Computing (GSA)

Continued ….

- ***Resource pooling*** **-- There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter). Examples of resources include storage, processing, memory, network bandwidth, and virtual machines.**
- ***Rapid elasticity*** **-- Capabilities can be rapidly and elastically provisioned, in some cases automatically, to quickly scale out and rapidly released to quickly scale in. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time.**

Cloud Computing (GSA)

- ***Measured Service*** **-- Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported providing transparency for both the provider and consumer of the utilized service.**
- **Many of the functional definitions used for "cloud computing" directly overlap or encompass "grid computing";**
- **As a result we can describe "cloud computing" as a "grid-like" but less disciplined distributed computing model, where there is no rigid constraint on interface standardisation, permitting proprietary interfaces.**

Widely Used Cloud Middleware

- Amazon EC2
- Apache Cloudstack
- OpenStack
- Eucalyptus

Apache CloudStack / Open Source Cloud

- **“The Apache CloudStack platform enables service providers to set up an on-demand, elastic cloud computing service. It enables a utility computing service by allowing service providers to offer self-service virtual machine instances, storage volumes, and networking configurations over the Internet.”**
- **“The Apache CloudStack platform enables enterprises to set up a private cloud for use by their own employees. The current generation of virtualization infrastructure targets enterprise IT departments who manage virtual machines the same way they would manage physical machines. The Apache CloudStack platform, on the other hand, enables self-service of virtual machines by users outside of the IT department.”**

OpenStack / Open Source Cloud

- "Massively scalable cloud operating system";
- Initiated by NASA and Rackspace Hosting;
- "global collaboration of developers and cloud computing technologists producing the ubiquitous open source cloud computing platform for public and private clouds."
- "The project aims to deliver solutions for all types of clouds by being simple to implement, massively scalable, and feature rich."
- 'The technology consists of a series of interrelated projects delivering various components for a cloud infrastructure solution."

Elastic Utility Computing Architecture / Open Source

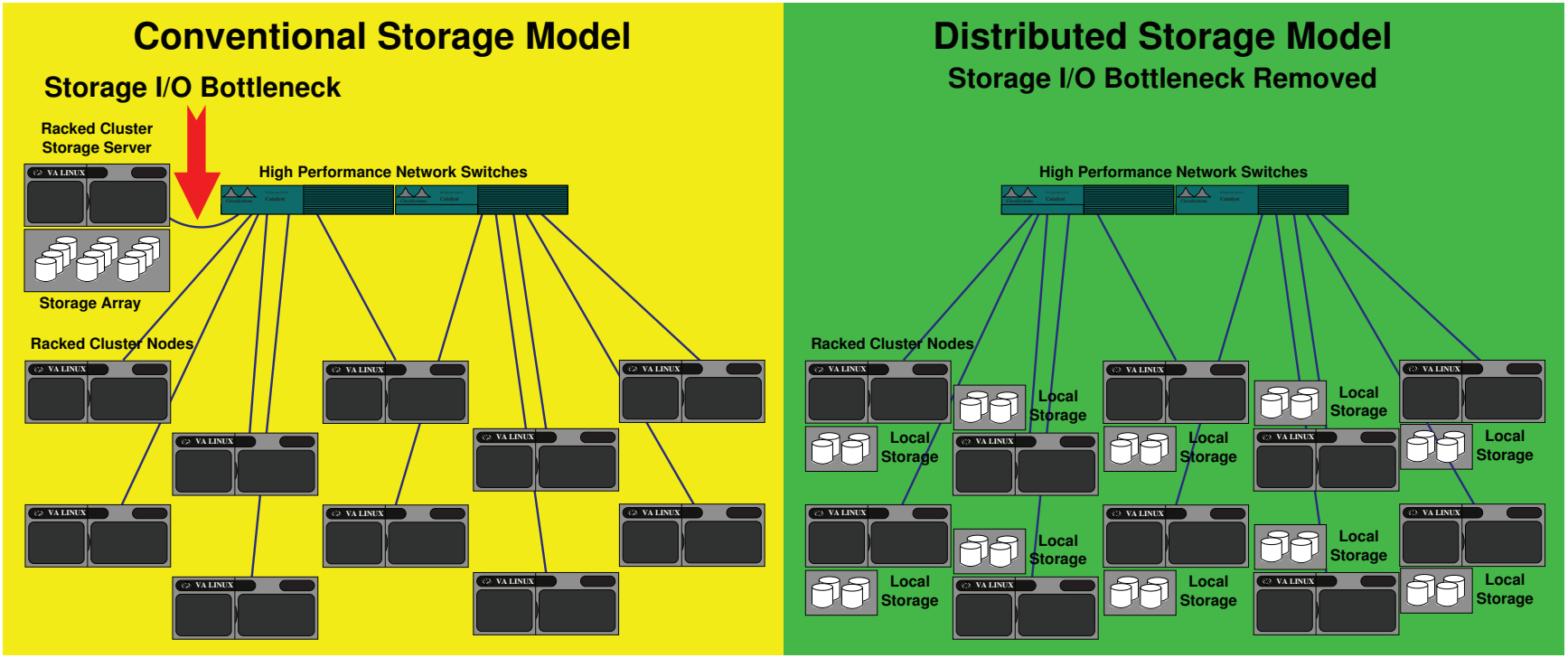
- **“The Eucalyptus Cloud platform is open source software for building AWS-compatible private and hybrid clouds. It pools together existing virtualized infrastructure to create cloud resources for compute, network and storage.”**
- **“Besides being compatible with the de facto standard AWS APIs and tools, the Eucalyptus Cloud is modular, distributed, and scalable and supports multiple hypervisors.”**
- **“Eucalyptus Systems provides IT organizations in enterprises and technology businesses with the leading open source software for building AWS-compatible private and hybrid clouds. Eucalyptus is uniquely suited for enterprise clouds with production-ready software that supports industry-standard AWS APIs, including EC2, S3, EBS, and IAM.”**

DISTRIBUTED STORAGE

Distributed vs. Conventional Storage Schemes

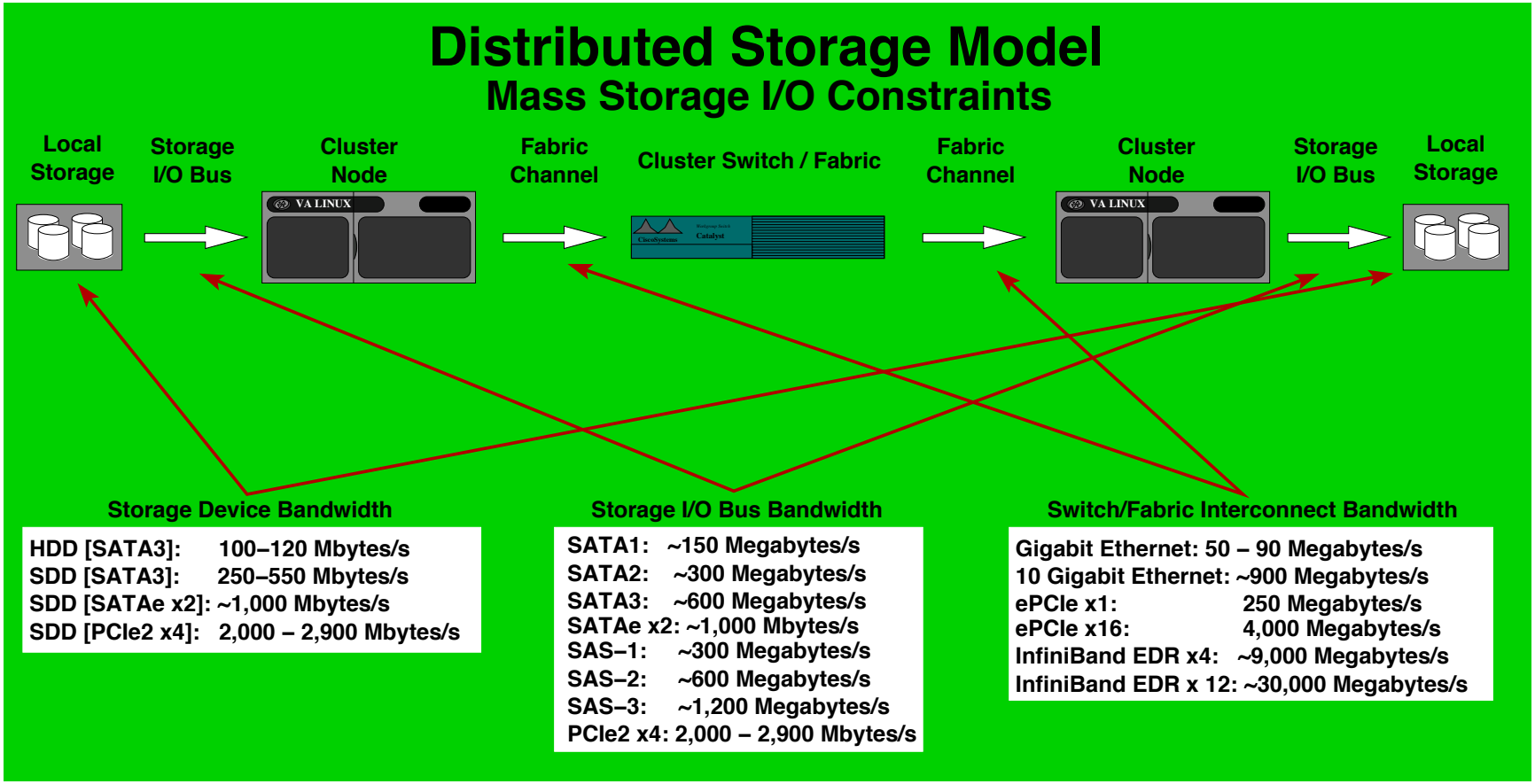
- Traditional mass storage schemes involve the use of local disk storage on machines, which may be individual disk drives, or RAID (Redundant Array of Independent Disks) storage arrays;
- Where the dataset is large, and significant disk bandwidth is required for applications, the local interface to the disk or disk storage array will become a performance bottleneck;
- The idea of distributed storage is to spread a large dataset across a very large number of host machines, so that the disk I/O traffic is also spread, across a large number of disk interfaces, thus removing the interface bottleneck of a large storage device or array;
- The *Hadoop Distributed File System* is the most widely used distributed storage scheme, but not the only one;

Conventional vs. Distributed Storage Models



- **In the distributed model, the I/O traffic to/from disks is spread across multiple nodes, and multiple storage I/O channels, to avoid any single bottleneck, and provide redundancy.**

Distributed Storage Performance

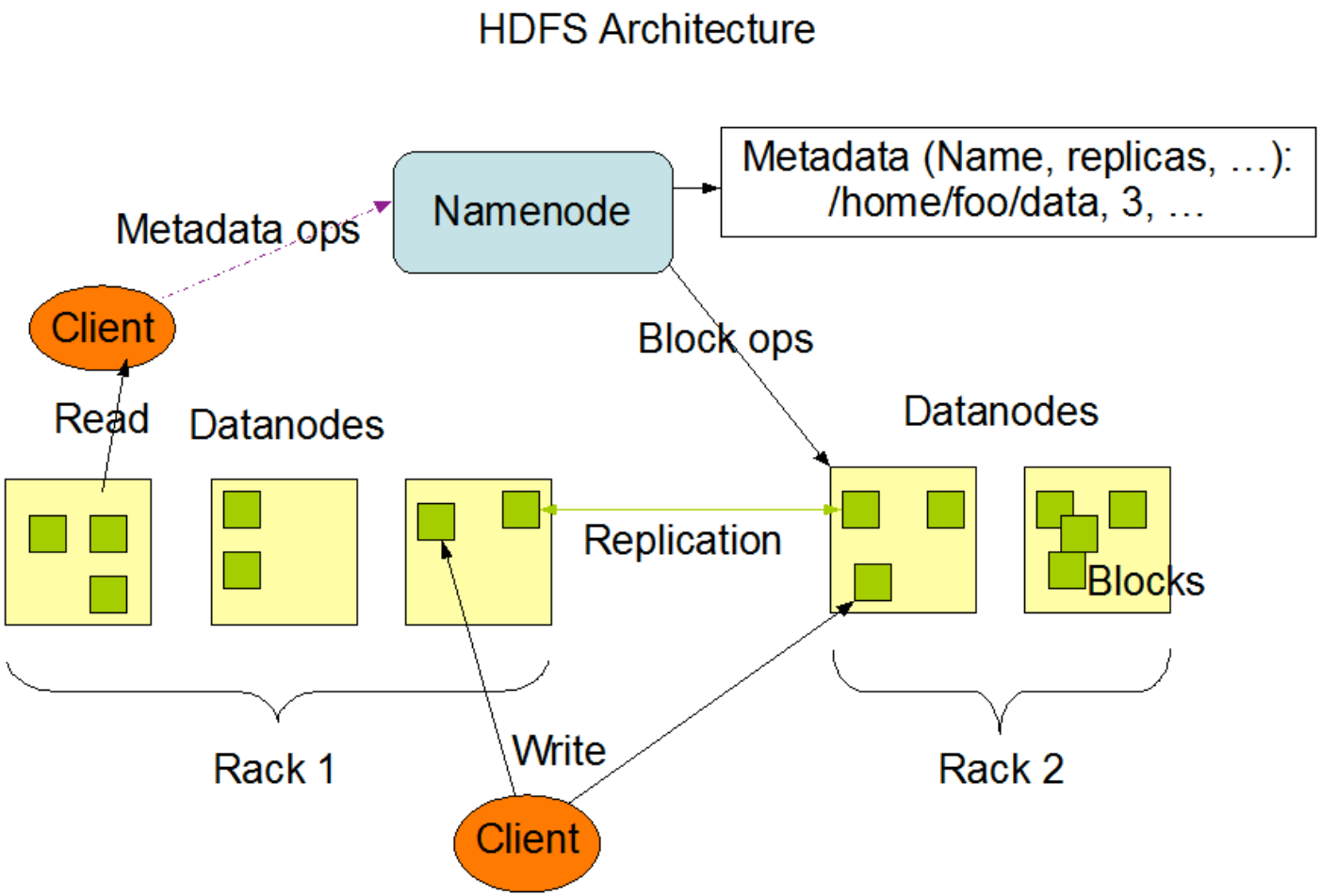


Above are best case bounds, as queueing effects reduce bandwidth

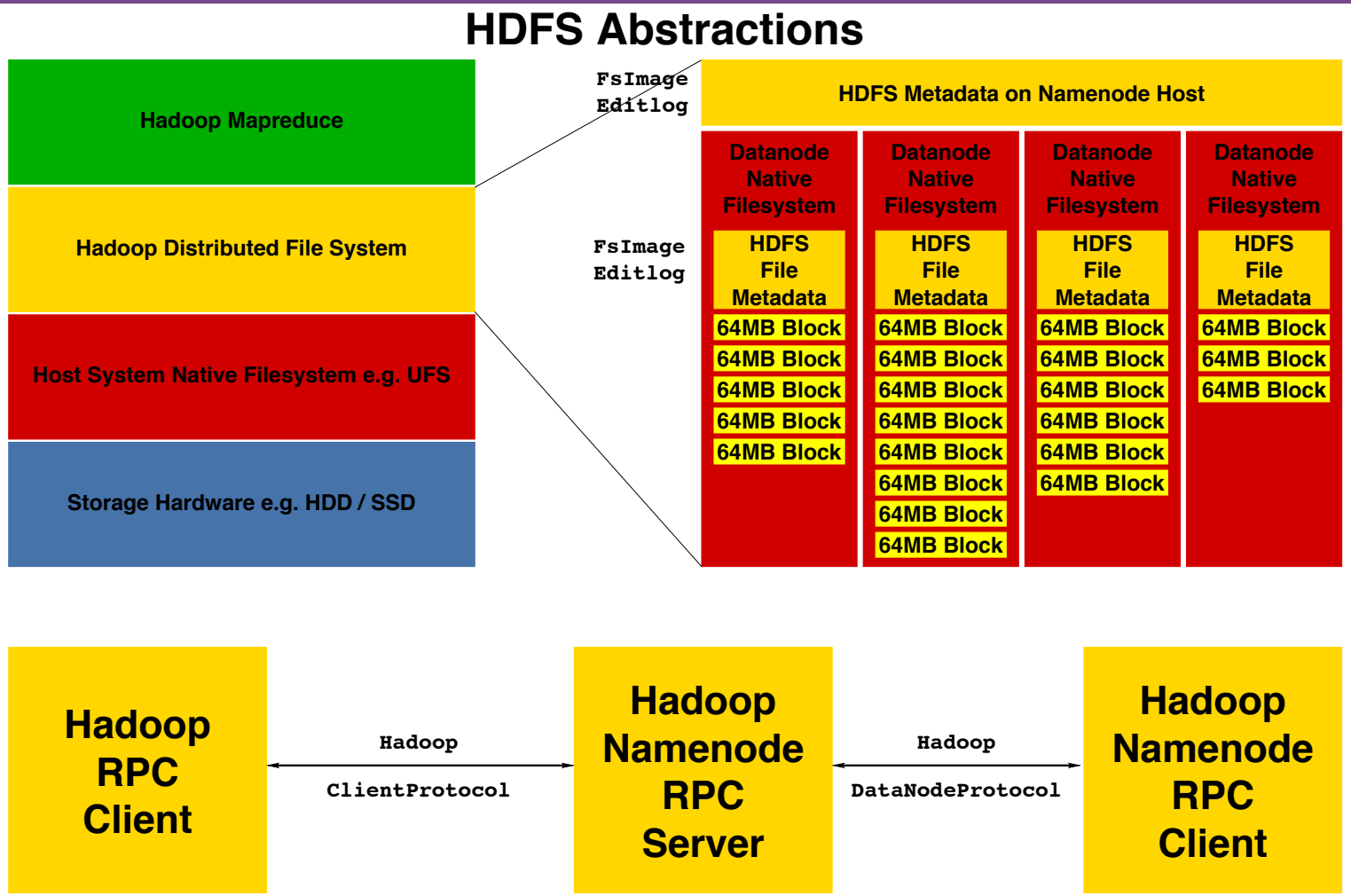
Distributed Storage Schemes - *Hadoop*

- ***Hadoop*** **is the most widely used distributed storage scheme, employing the Hadoop Distributed File System (HDFS);**
- **In a *Hadoop* system, a single central `Namenode` host acts as an RPC server for a group of `Datanode` hosts, each of which stores a portion of the filesystem, with two copies of each 'data block' put on different `Datanode` hosts for redundancy, and to spread I/O traffic load and improve performance;**
- **The `Namenode` host and `Datanode` hosts all maintain copies of the `FsImage` and `EditLog` files, which provide block maps using `i-nodes`, and a *transaction journal* for reliability;**
- **The main drawback of *Hadoop* in 'big data' applications is that all accesses occur via the `Namenode` host, which can suffer overload costing performance; While Hadoop distributes I/O load, compute load is not distributed!**

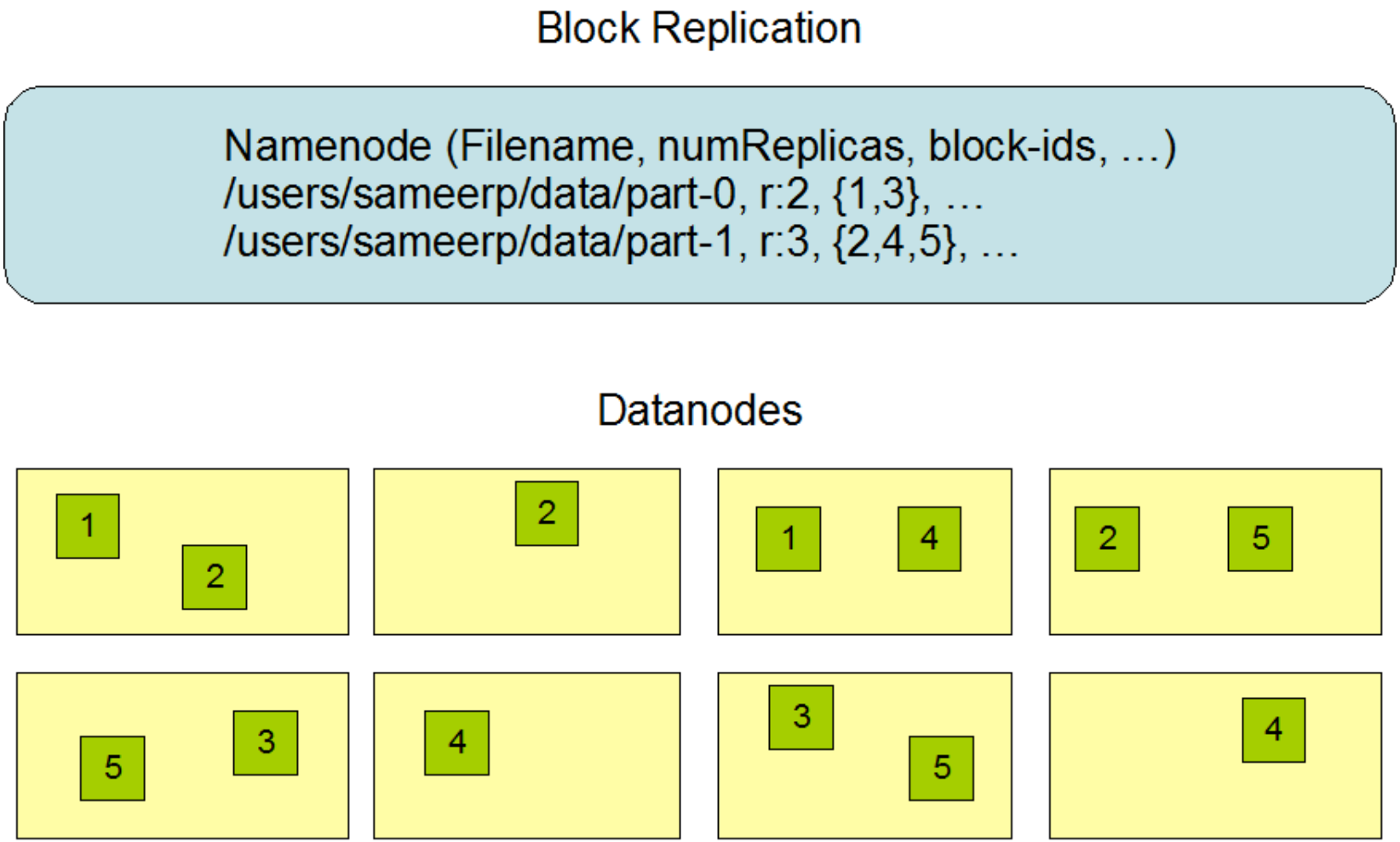
Hadoop HDFS Architecture



Hadoop HDFS Abstractions



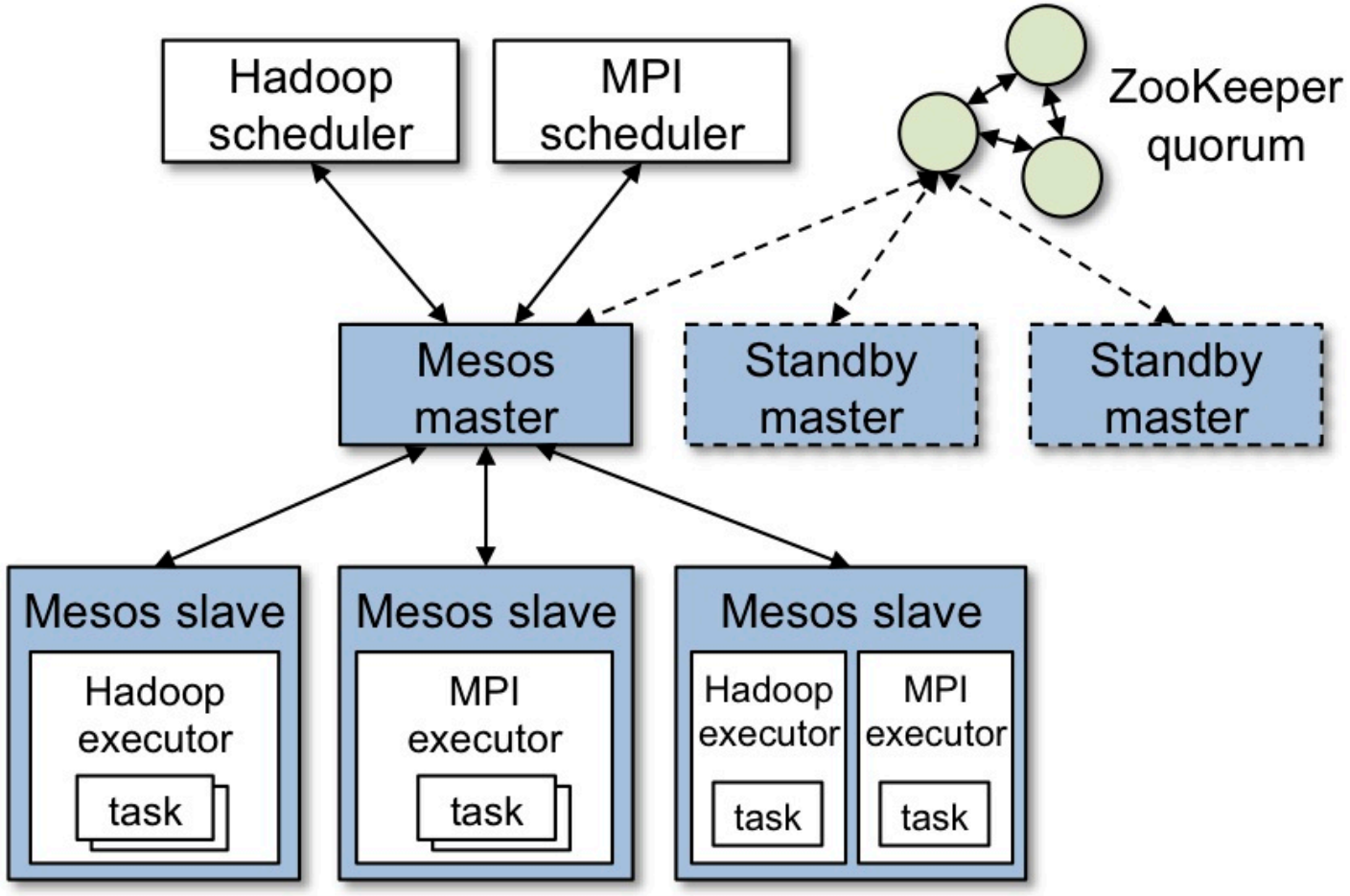
Hadoop HDFS Architecture



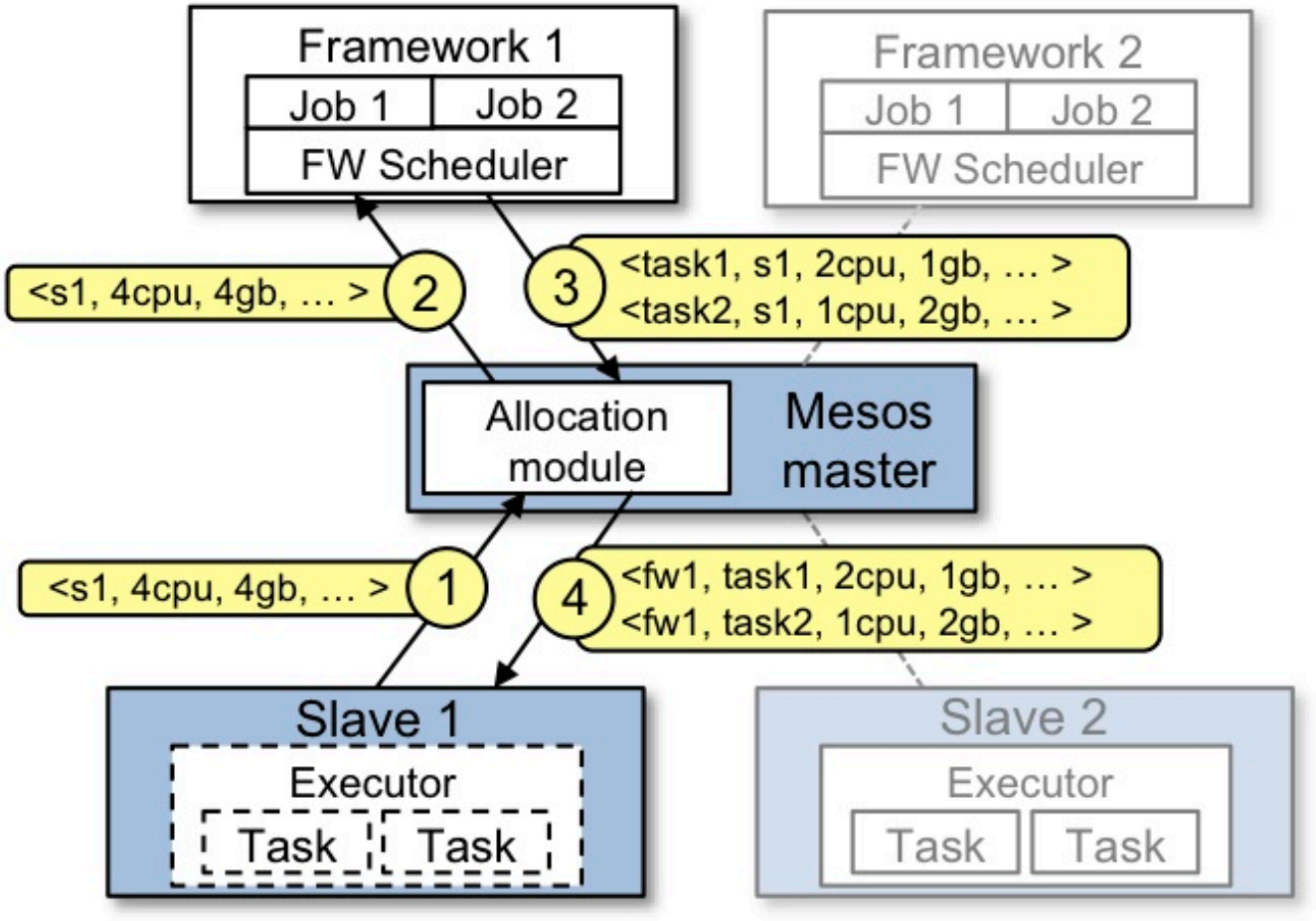
Distributed Storage Schemes - *Spark*

- An increasingly popular alternative to *Hadoop* is the *Spark* clustering environment, developed by the University of California, Berkeley, developed for 'big data' analytics;
- While *Hadoop* is unable to distribute computing load, *Spark* was developed specifically to do so;
- *Spark* was designed to cache datasets in the memory of nodes across the cluster, to permit each node to operate in parallel on the dataset, improving performance by exploiting the high bandwidth to main memory, and the local computing power in the node;
- A common practice is to run both *Spark* and *Hadoop, and MPI,* on a single cluster, using the *Mesos* distributed kernel to manage both systems;

Mesos Architecture



Mesos Architecture Example



Summary

- There are a wide range of distributed computing schemes now in use, and the number is growing over time;
- Traditional parallel clustering environments can be extended into Grids, comprising large numbers of clusters;
- Clouds are now the preferred solution in commercial computing, intended to provide elastic sharing of compute and storage resources;
- The advent of ‘big data’ and the usefulness of ‘big data analytics’ has seen the development of Hadoop, and more recent schemes as Spark;

READING / BACKGROUND [N/E]

Distributed Computing Schemes

- Extending Topic 1 discussion to explore most widely used new distributed computing schemes, especially grids;
- Technologies leading to large scale distributed computing models;
- The ideas underpinning clusters, grids, clouds and web services;
- Distributed computing technologies - computer hardware, networking, storage;
- Taxonomy - application and portals, application environments, middleware, fabric and connectivity;
- Grid Classifications [Definitions];
- Evolutionary path to Grid and Web applications;
- This is important foundation knowledge for later materials.

Batch Processing…

- Central computers used *batch jobs* concept
- Batch jobs are scheduled according to their resource needs:

1. Short queue for low CPU low resource jobs
2. Long queue for massive jobs

- SMP brought the concept of job scheduling over groups of CPUs;
- Batch processing is well suited to repeated runs using the same application;
- Problems can arise in mixed workloads with determining best scheduling strategies.

Batch Processing …

- Batch job processing then moved to distributed environments
- Load Sharing Facility (LSF), a sophisticated batch queue manager with distributed systems support
- Condor and Condor-G, supports service disruptions.
- Unicore, is much more than a queue manager, a vertically integrated grid infrastructure.
- Globus Resource Allocation Manager (Grids).

Data Storage

- **Network Area Storage (NAS):**

1. **Logically distributed storage;**
2. **Protocol based access to a central storage infrastructure, NFS, AFS;**
3. **Easy to implement;**
4. **Low cost but performance limited by network;**

- **Storage Area Network (SAN):**

1. **Physically distributed storage;**
2. **Requires special networking infrastructure;**
3. **Higher in *initial* cost;**
4. **Highly-scalable and flexible;**

Storage Management

- Hierarchical Storage Management (HSM)

1. SAMfs, Sun Microsystems, open tar based
2. Data Management Facility (DMF), widely used
3. CASTOR, from CERN

- Meta data – data describing other data:

1. Meta Data Service (MDS)
2. Storage Resource Broker (SRB)

- Metadata management is becoming a major area of interest in archiving and search applications.

HPPC Infrastructure

- High Performance Parallel Computing
- High Throughput Networks:

1. Ethernet (10 Megabit/s)
2. Fast Ethernet(100 Megabit/s)
3. Gigabit(1000 Megabit/s)
4. 10 Gigabit etc

- Low-latency networks: Myrinet, 4-10 times lower latency than standard TCP/IP over Gigabit;
- Network infrastructure performance can be critical in any HPPC or HPC system designs – throughput and latency can have a large performance impact.

Long Distance High Throughput Data Transfers

- HPPC/HPC applications forced the development of new protocols capable of performing large parallel transfers quickly;
- GridFTP: provides software level data streaming for higher throughput;
- Fast TCP: provides better utilisation of the network by TCP parameter tuning;
- SCTP: provides data streaming at the network level for better network bandwidth utilisation and quality of service;
- Insufficient network performance can cripple HPPC/HPC applications.

Distributed Computing Evolution (Refer Topic 1)

- BSD Sockets: Early FTP, SMTP, and NNTP Services
- Remote Procedure Call (RPC)
- Windows COM/DCOM
- CORBA for supporting OO concepts
- Java Remote Method Invocation (RMI)
- Web Services, for a uniform higher-level communications standard

Technology Trends (1)

- Initially, distributed applications used sockets to connect two processes on different hosts; the programmer had to construct everything else.
- RPC was intended to remove much of the burden to the programmer by providing a mechanism for remote calling of procedures, hiding complexity.
- CORBA / OLE/DCOM extended the model from procedures to complete objects, including data and procedural components.
- Java RMI conceptually like CORBA / DCOM but platform specific.
- Web Services modelled on RPC approach.

Technology Trends (2)

- *Language Dependencies:* the "preferred" language of the period has influenced the design of distributed computing models.
- Sockets and RPC designed around C language.
- CORBA / OLE / DCOM designed around C++ language.
- Java RMI designed around Java and W3
- Web Services designed around XML, W3 and Java.
- *Level of Abstraction:* details of network progressively hidden away from programmers as technology has evolved.