Cloud Computing Applications and Paradigms

ACKNOWLEDGEMENTS

- This presentation has been made from various sources with minimum modifications from the presenter.
- The presenter is grateful to the authors of those various sources.
- The presenter acknowledge the efforts of those authors and thank them wholeheartedly.

Cloud applications

- Ideal applications for cloud computing:
  - Web services.
  - Resource Intensive
  - Transaction-based service.
- Applications unlikely to perform well on a cloud:
  - Applications with a complex workflow and multiple dependencies, as is often the case in high-performance computing.
  - Applications which require intensive communication among concurrent instances.
  - When the workload cannot be arbitrarily partitioned

Challenges for cloud application development

- Performance isolation –
  - nearly impossible to reach in a real system, especially when the system is heavily loaded.
- Reliability –
  - major concern; server failures expected when a large number of servers cooperate for the computations.
- Cloud infrastructure exhibits latency and bandwidth fluctuations which affect the application performance.
- Performance considerations limit the amount of *data logging*; the ability to identify the source of unexpected results and errors is helped by frequent logging.

Existing Applications

- Categories of existing applications
  - Processing pipelines.
    - are data-intensive and sometimes compute-intensive applications
  - Batch processing systems.
    - also cover a broad spectrum of data-intensive applications in enterprise computing
  - Web applications.

Processing Pipelines

- Indexing large datasets created by web crawler engines.
- Data mining - searching large collections of records to locate items of interests.
- Image processing .
  - Image conversion, e.g., enlarge an image or create thumbnails.
  - Compress or encrypt images.
- Video transcoding from one video format to another, e.g., from AVI to MPEG.
- Document processing.
  - Convert large collections of documents from one format to another, e.g., from Word to PDF.
  - Encrypt documents.
  - Use Optical Character Recognition to produce digital images of documents.

Batch Processing

- Generation of daily, weekly, monthly, and annual activity reports for retail, manufacturing, other economical sectors.
- Processing, aggregation, and summaries of daily transactions for financial institutions, insurance companies, and healthcare organizations.
- Processing billing and payroll records.
- Management of the software development, e.g., nightly updates of software repositories.
- Automatic testing and verification of software and hardware systems.

Web Access

- Sites for online commerce.
- Sites with a periodic or temporary presence.
  - Conferences or other events.
  - Active during a particular season (e.g., the Holidays Season) or income tax reporting.
- Sites for promotional activities.
- Sites that ``sleep'' during the night and auto-scale during the day.

Architectural styles for cloud applications

- Based on the client-server paradigm.
- Stateless servers –
  - view a client request as an independent transaction and respond to it;
  - the client is not required to first establish a connection to the server.
- Clients and servers communication using Remote Procedure Calls (RPCs).
- Simple Object Access Protocol (SOAP) - application protocol for web applications;
  - message format based on the XML.
  - Uses TCP or UDP transport protocols.
- Representational State Transfer (REST) - software architecture for distributed hypermedia systems.
  - Supports client communication with stateless servers,
  - it is platform independent, language independent,
  - supports data caching, and
  - can be used in the presence of firewalls.

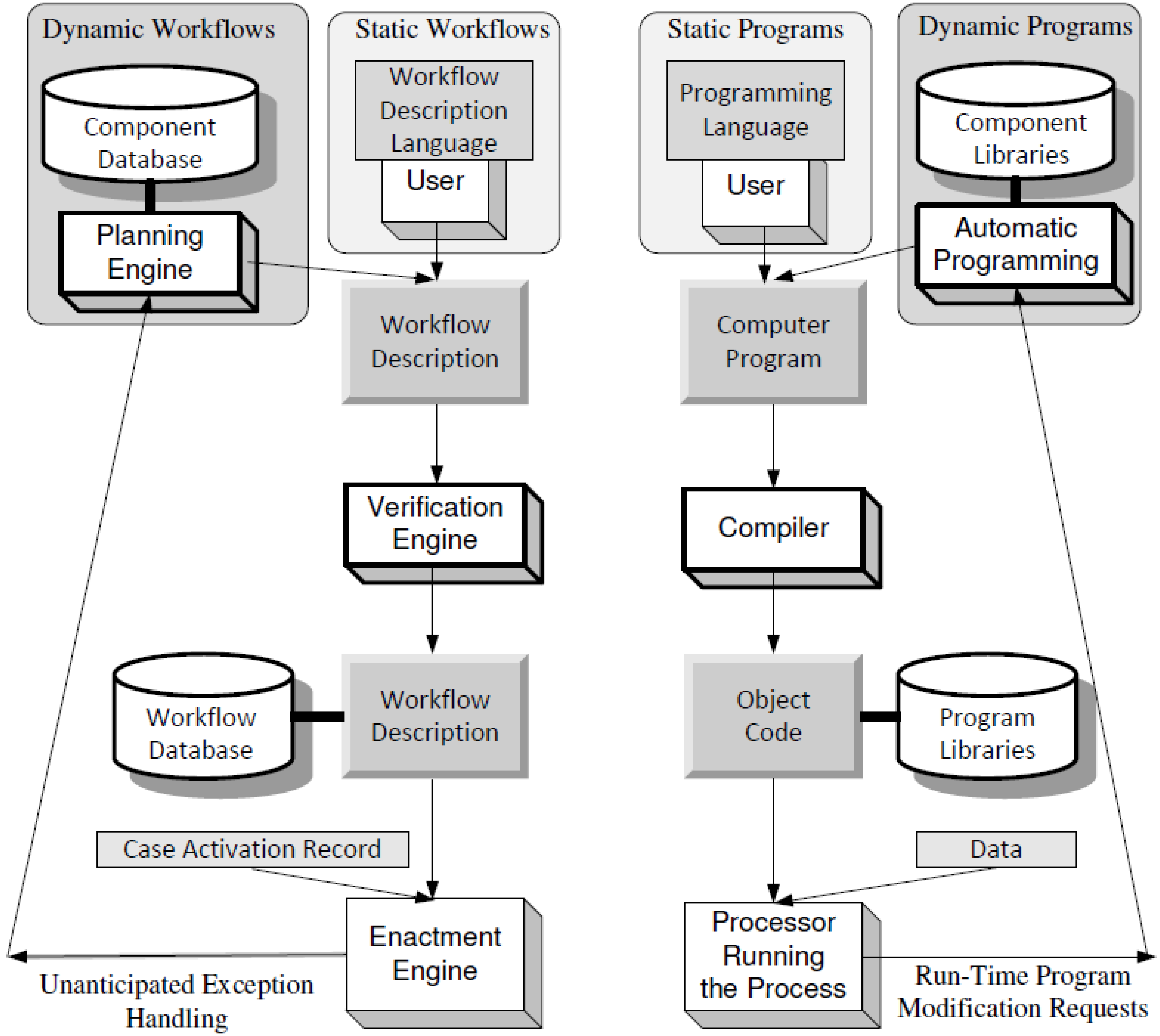
WORKFLOWS

Workflows

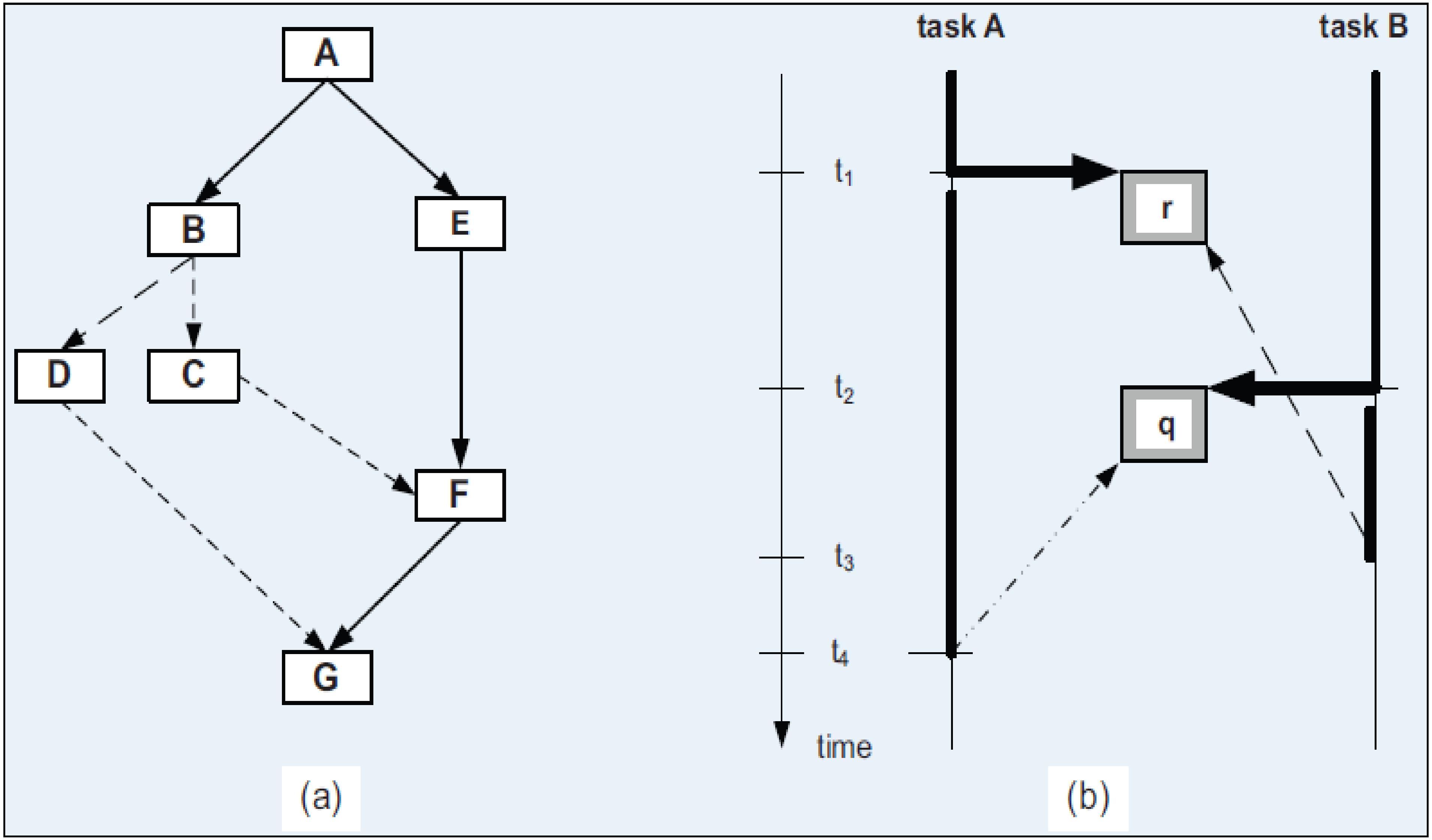
Description of a complex activity involving multiple interdependent tasks

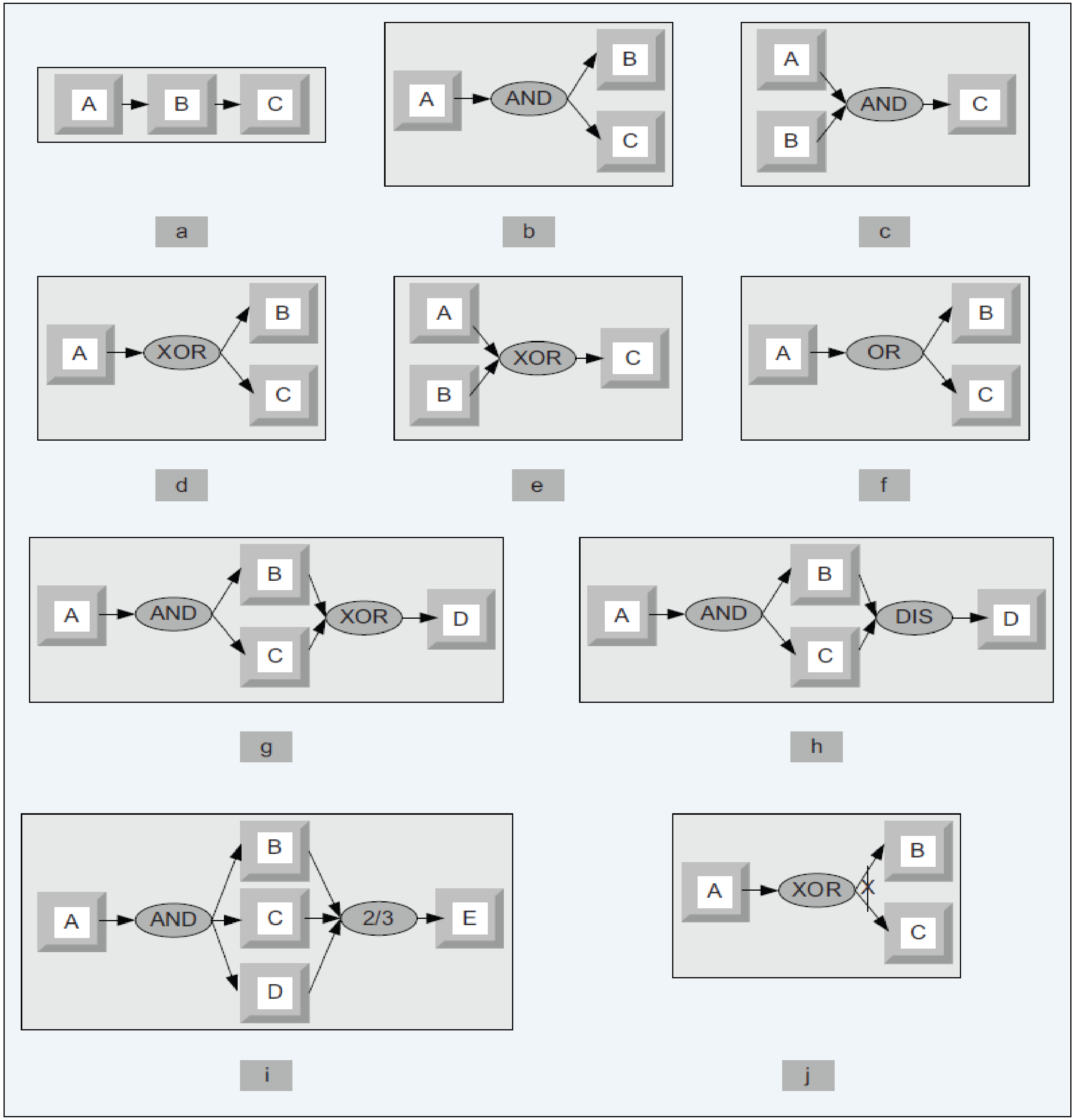
Workflows Concepts

- *Process description* - structure describing the tasks to be executed and the order of their execution. Resembles a flowchart.
- *Case* - an instance of a process description.
- *State of a case at time t* - defined in terms of tasks already completed at that time.
- *Events* - cause transitions between states.
- The *life cycle of a workflow* - creation, definition, verification, and enactment; similar to the life cycle of a traditional program (creation, compilation, and execution).


Dynamic Workflows
Component Database
Planning Engine
Static Workflows
Workflow Description Language
User
Workflow Description
Verification Engine
Workflow Database
Workflow Description
Case Activation Record
Enactment Engine
Unanticipated Exception Handling
Static Programs
Programming Language
User
Dynamic Programs
Component Libraries
Automatic Programming
Computer Program
Compiler
Object Code
Program Libraries
Data
Processor Running the Process
Run-Time Program Modification Requests

Workflow: Desirable Properties





Sequence

AND split

Synchronization

XOR split

XOR merge

OR split

Multiple Merge

Discriminator

N out of M join

Deferred Choice

ZOOKEEPER

Introduction to Apache ZooKeeper™

http://zookeeper.apache.org/

What is a Distributed System?

“A distributed system consists of multiple computers that communicate through a computer network and interact with each other to achieve a common goal.”

- Wikipedia

Fallacies of Distributed Computing

- The network is reliable.
- Latency is zero.
- Bandwidth is infinite.
- The network is secure.
- Topology doesn't change.
- There is one administrator.
- Transport cost is zero.
- The network is homogeneous.

Reference: *http://en.wikipedia.org/wiki/Fallacies_of_Distributed_Computing*

Coordination in a distributed system

- *Coordination*: An act that multiple nodes must perform together.
- Examples:
  - Group membership
  - Locking
  - Publisher/Subscriber
  - Leader Election
  - Synchronization
- Getting node coordination correct is very hard!

Copyright by Shamus O'Reilly via Flickr

Introducing ZooKeeper

“ ZooKeeper allows distributed processes to coordinate with each other through a shared hierarchical name space of data registers. ”

- ZooKeeper Wiki

ZooKeeper is much more than a distributed lock server!

What is ZooKeeper?

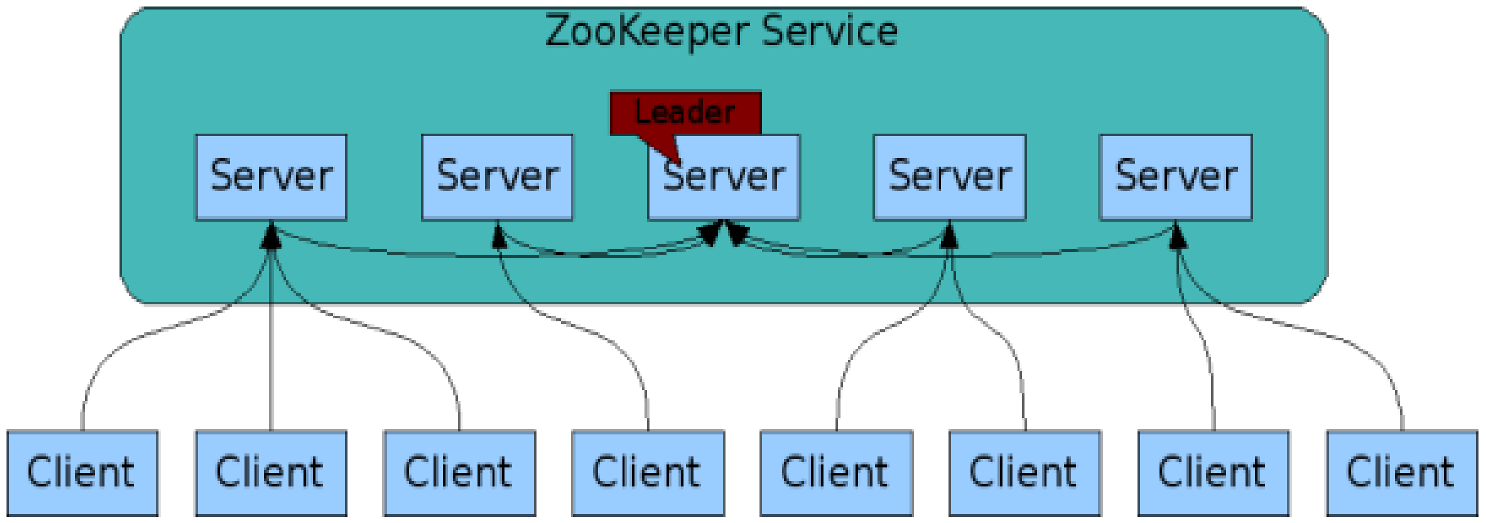
- An open source, high-performance coordination service for distributed applications.
- Exposes common services in simple interface:
  - naming
  - configuration management
  - locks & synchronization
  - group services

  … developers don't have to write them from scratch
- Build your own on it for specific needs.

ZooKeeper Use Cases

- Configuration Management
  - Cluster member nodes bootstrapping configuration from a centralized source in unattended way
  - Easier, simpler deployment/provisioning
- Distributed Cluster Management
  - Node join / leave
  - Node statuses in real time
- Naming service – e.g. DNS
- Distributed synchronization - locks, barriers, queues
- Leader election in a distributed system.
- Centralized and highly reliable (simple) data registry

The ZooKeeper Service



- ZooKeeper Service is replicated over a set of machines
- All machines store a copy of the data (in memory)
- A leader is elected on service startup
- Clients only connect to a single ZooKeeper server & maintains a TCP connection.
- Client can read from any Zookeeper server, writes go through the leader & needs majority consensus.

Image: https://cwiki.apache.org/confluence/display/ZOOKEEPER/ProjectDescription

The ZooKeeper Data Model

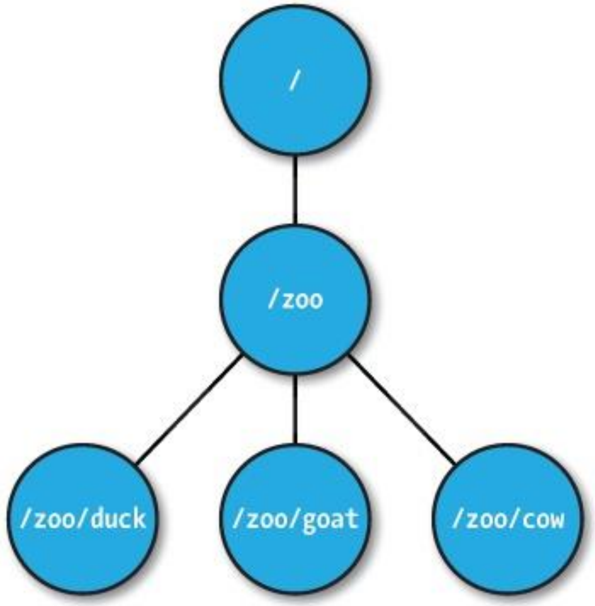
- ZooKeeper has a hierarchal name space.
- Each node in the namespace is called as a *ZNode.*
- Every ZNode has data (given as byte[]) and can optionally have children.

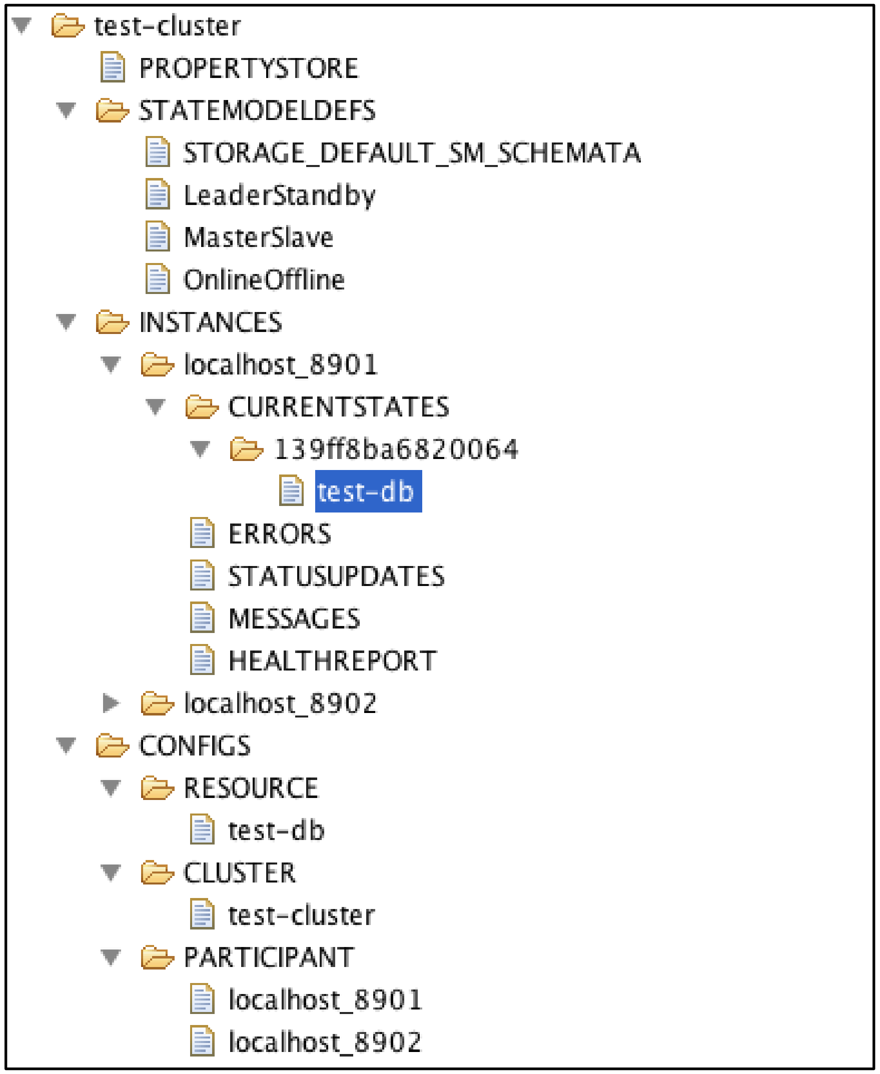
```
parent : "foo"
    |-- child1 : "bar"
    |-- child2 : "spam"
    `-- child3 : "eggs"
              `-- grandchild1 : "42"
```

- ZNode paths:
  - canonical, absolute, slash-separated
  - no relative references.
  - names can have Unicode characters

ZNodes

- Maintain a stat structure with version numbers for data changes, ACL changes and timestamps.
- Version numbers increases with changes
- Data is read and written in its entirety





```
test-cluster
    PROPERTYSTORE
    STATEMODELDEFS
        STORAGE_DEFAULT_SM_SCHEMATA
        LeaderStandby
        MasterSlave
        OnlineOffline
    INSTANCES
        localhost_8901
            CURRENTSTATES
                139ff8ba6820064
                    test-db
            ERRORS
            STATUSUPDATES
            MESSAGES
            HEALTHREPORT
        localhost_8902
    CONFIGS
        RESOURCE
            test-db
        CLUSTER
            test-cluster
        PARTICIPANT
            localhost_8901
            localhost_8902
```

Image: http://helix.incubator.apache.org/Architecture.html

ZNode Types

- Persistent Nodes
  - exists till explicitly deleted
- Ephemeral Nodes
  - exists as long as the session is active
  - can’t have children
- Sequence Nodes (Unique Naming)
  - append a monotonically increasing counter to the end of path
  - applies to both persistent & ephemeral nodes

ZNode Operations

Operation	Type
create	Write
delete	Write
exists	Read
getChildren	Read
getData	Read
setData	Write
getACL	Read
setACL	Write
sync	Read

ZNodes are the main entity that a programmer access.

ZNode Watches

- Clients can set watches on znodes:
  - NodeChildrenChanged
  - NodeCreated
  - NodeDataChanged
  - NodeDeleted
- Changes to a znode trigger the watch and ZooKeeper sends the client a notification.
- Watches are one time triggers.
- Watches are always ordered.
- Client sees watched event before new znode data.
- Client should handle cases of latency between getting the event and sending a new request to get a watch.

API Synchronicity

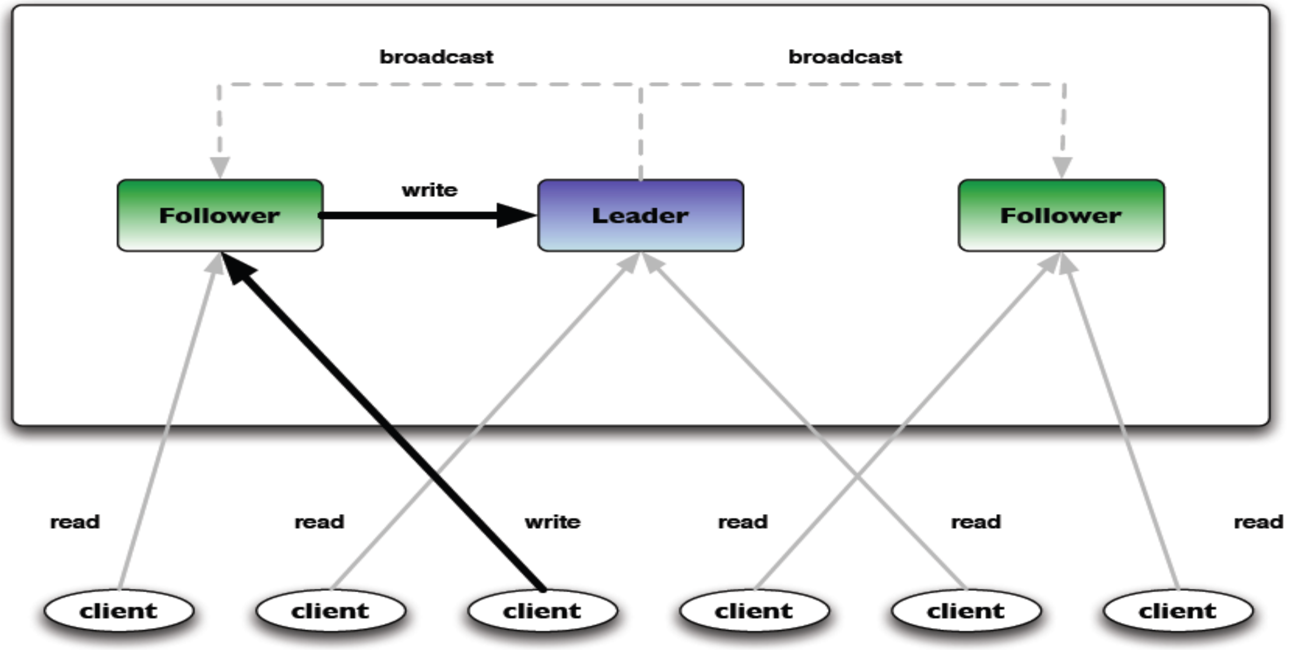
- API methods are sync as well as async
- Sync:

```
exists(“/test-cluster/CONFIGS", null);
```

- Async:

```
exists("/test-cluster/CONFIGS", null, new StatCallback() {
    @Override
    public processResult(int rc, String path, Object ctx, Stat stat)
    {
                //process result when called back later
    }
    }, null);
```

ZNode Reads & Writes



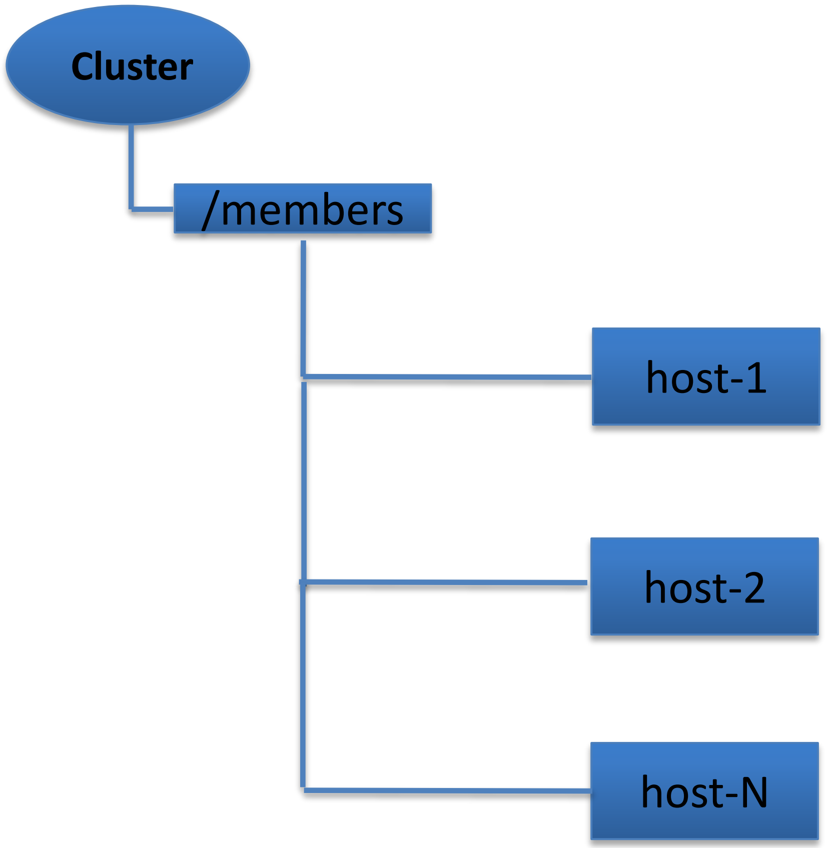
- Read requests are processed locally at the ZooKeeper server to which the client is currently connected
- Write requests are forwarded to the leader and go through majority consensus before a response is generated.

Image: http://www.slideshare.net/scottleber/apache-zookeeper

Consistency Guarantees

- **Sequential Consistency**: Updates are applied in order
- **Atomicity**: Updates either succeed or fail
- **Single System Image**: A client sees the same view of the service regardless of the ZK server it connects to.
- **Reliability**: Updates persists once applied, till overwritten by some clients.
- **Timeliness**: The clients' view of the system is guaranteed to be up-to-date within a certain time bound. (Eventual Consistency)

Recipe #1: Cluster Management



Each Client Host i, i:=1 .. N

1. Watch on /members
2. Create /members/host-${i} as ephemeral nodes
3. Node Join/Leave generates alert
4. Keep updating /members/host-${i} periodically for node status changes
   (load, memory, CPU etc.)

Recipe #2: Leader Election

1. A znode, say "/svc/election-path"
2. All participants of the election process create an ephemeral-sequential node on the same election path.
3. ***The node with the smallest sequence number is the leader.***
4. Each "follower" node listens to the node with the next lower seq. number
5. Upon leader removal go to

   election-path and find a new leader,

   or become the leader if it has the lowest sequence number.
6. Upon session expiration check the election state and go to election if needed

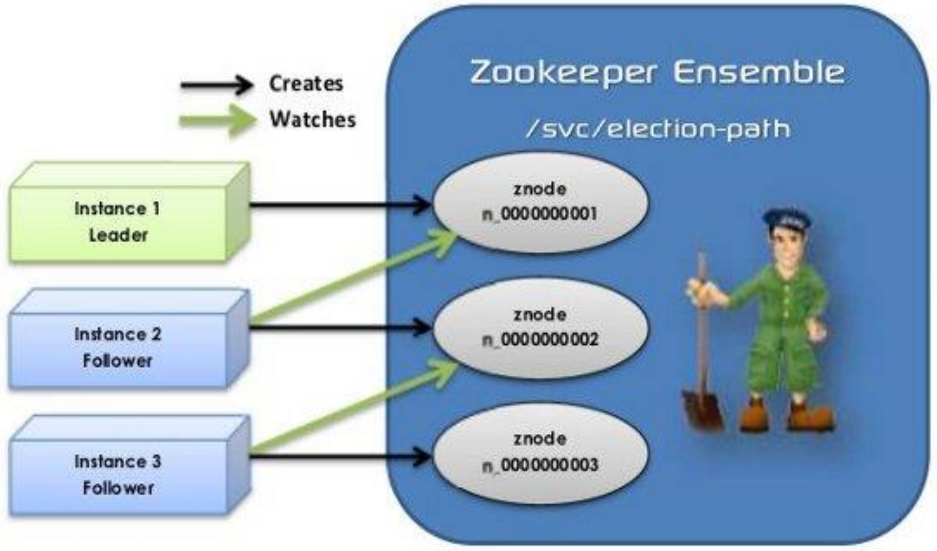


Image: http://techblog.outbrain.com/2011/07/leader-election-with-zookeeper/

Recipe #3: Distributed Exclusive Lock

Assuming there are N clients trying to acquire a lock

- Clients creates an ephemeral, sequential znode under the path /Cluster/_locknode_
- Clients requests a list of children for the lock znode (i.e. _locknode_)
- ***The client with the least ID according to natural ordering will hold the lock.***
- Other clients sets watches on the znode with id immediately preceding its own id
- Periodically checks for the lock in case of notification.
- The client wishing to release a lock deletes the node, which triggering the next client in line to acquire the lock.

```
ZK
|---Cluster
   +---config
   +---memberships
   +---_locknode_
      +---host1-3278451
      +---host2-3278452
      +---host3-3278453
      +--- …
      \---hostN-3278XXX
```

Language Bindings

- ZooKeeper ships client libraries in:
  - Java
  - C
  - Perl
  - Python
- Community contributed client bindings available for Scala, C#, Node.js, Ruby, Erlang, Go, Haskell

  https://cwiki.apache.org/ZOOKEEPER/zkclientbindings.html

A few points to remember

- Watches are one time triggers
  - Continuous watching on znodes requires reset of watches after every events / triggers
- Too many watches on a single znode creates the "*herd effect*" - causing bursts of traffic and limiting scalability
- If a znode changes multiple times between getting the event and setting the watch again, carefully handle it!
- Keep session time-outs long enough to handle long garbage-collection pauses in applications.
- Set Java max heap size correctly to *avoid swapping.*
- Dedicated disk for ZooKeeper transaction log

Who uses ZooKeeper?

Companies:

- Yahoo!
- Zynga
- Rackspace
- LinkedIn
- Netflix
- *and many more…*

Projects in FOSS:

- Apache Map/Reduce (Yarn)
- Apache HBase
- Apache Solr
- Neo4j
- Katta
- *and many more…*

Reference: https://cwiki.apache.org/confluence/display/ZOOKEEPER/PoweredBy

ZooKeeper In Action @Twitter

- Used within Twitter for service discovery
- How?
  - Services register themselves in ZooKeeper
  - Clients query the production cluster for service “A” in data center “XYZ”
  - An up-to-date host list for each service is maintained
  - Whenever new capacity is added the client will automatically be aware
  - Also, enables load balancing across all servers

Reference: http://engineering.twitter.com/

HADOOP AND MAPREDUCE

BIG DATA

How big is BIG?

Name (SI)	***Abbr.***	***Usage (Decimal)***	***Number of Bytes (Decimal)***	***Usage (Binary)***
1 bit	b			
1 byte	B			2^{0}
1 kilobyte	KB or kB	10^{3}	1,000	2^{10}
1 megabyte	MB	10^{6}	1,000,000	2^{20}
1 gigabyte	GB	10^{9}	1,000,000,000	2^{30}
1 terabyte	TB	10^{12}	1,000,000,000,000	2^{40}
1 petabyte	PB	10^{15}	1,000,000,000,000,000	2^{50}
1 exabyte	EB	10^{18}	1,000,000,000,000,000,000	2^{60}
1 zettabyte	ZB	10^{21}	1,000,000,000,000,000,000,000	2^{70}
1 yottabyte	YB	10^{24}	1,000,000,000,000,000,000,000,000	2^{80}

[1] http://en.wikipedia.org/wiki/Yottabyte

How big is BIG?

Name (SI)	*Abbr.*	*Usage (Decimal)*	*Number of Bytes (Decimal)*	*Usage (Binary)*
1 megabyte	MB	10^{6}	1,000,000	2^{20}
1 gigabyte	GB	10^{9}	1,000,000,000	2^{30}
1 terabyte	TB	10^{12}	1,000,000,000,000	2^{40}
1 petabyte	PB	10^{15}	1,000,000,000,000,000	2^{50}
1 exabyte	EB	10^{18}	1,000,000,000,000,000,000	2^{60}
1 zettabyte	ZB	10^{21}	1,000,000,000,000,000,000,000	2^{70}
1 yottabyte	YB	10^{24}	1,000,000,000,000,000,000,000,000	2^{80}

2006 - World's hard drives estimated at: ~**160 exabytes (EB)**[1]

2009 - Internet estimated to contain: ~**500 exabytes (EB)**[1]

2012 - *Not yet achieved one zettabyte (ZB) of information.*[1]

[1] http://en.wikipedia.org/wiki/Yottabyte

How big is BIG?

"Big" is really a red herring

- Oil companies, telecommunications companies, and other data-centric industries have had huge datasets for a long time.
- Datacenter energy example
- GPS ground stations example

As storage capacity continues to expand,
today's "big" is
tomorrow's "medium" and
next week's "small."

"Big data" is when the size of the data itself becomes part of the problem.

- At some point, traditional techniques for working with data run out of steam.

Reference: http://radar.oreilly.com/2010/06/what-is-data-science.html

How big is BIG?

- Size of the ‘digital universe’ in…
  2006 – 0.18 ZB
  2011 – 1.8 ZB ← 10-fold growth in five years!
- Examples from…
  New York Stock Exchange – 1 TB of **new** data **per day**
  Facebook – 10 billion photos, about 1 PB
  Large Hadron Collider near Geneva – will produce 15 PB per year
- Not just corporations, individuals too!
  Photos Photo Tags Spreadsheets Tweets
  Blogs Sensor Data YouTube Videos Word Documents
  PowerPoints etc.

Reference: Hadoop: The Definitive Guide, by Tom White

How big to BIG?

- Add to individuals and corporations another contributor: Machines !
- Today, machines generate data:
  - Machine operation logs, generated by
    - Monitoring agents installed in (Physical Machines – PMs), Virtual Machines (VMs)
    - Raw monitoring data are collected every second/minute/hour, at various granularity
  - Usage logs
    - Retail transactions - think of Amazon, EBay, PayPal - globally!
    - Consumer historic data (again, summarized/rolled-up data)
    - Computer and network performance for SLAs (Service Level Agreements)

Reference: Hadoop: The Definitive Guide, by Tom White

How big is BIG?

	1990	2011
Drive Capacity	1.4 GB	* ~1000 = 1 TB
Transfer Speed	4.4 MB/s	* ~25 =100 MB/s
Whole Drive Read Time	5 minutes	2.5 hours
Whole Drive Write Time	*Even Slower*	

Reference: Hadoop: The Definitive Guide, by Tom White

Issues with BIG Data

How can we process these vasts quantities of data?

How can we do it in (near) realtime?

How can we access the data in time to inform business decisions?

- *When to advertise goods and services?*
- *Which ones to advertise?*
- *To which customer/potential buyer?*
- *How frequently?*

HADOOP

Timeline

- Dec 2004: Dean/Ghemawat (Google) MapReduce paper
- 2005: Doug Cutting and Mike Cafarella (Yahoo) create Hadoop, at first only to extend Nutch (the name is derived from Doug’s son’s toy elephant)
- 2006: Yahoo runs Hadoop on 5-20 nodes

Timeline

- March 2008: Cloudera founded
- July 2008: Hadoop wins TeraByte sort benchmark (1st time a Java program won this competition)
- April 2009: Amazon introduce “Elastic MapReduce” as a service on S3/EC2
- June 2011: Hortonworks founded

Timeline

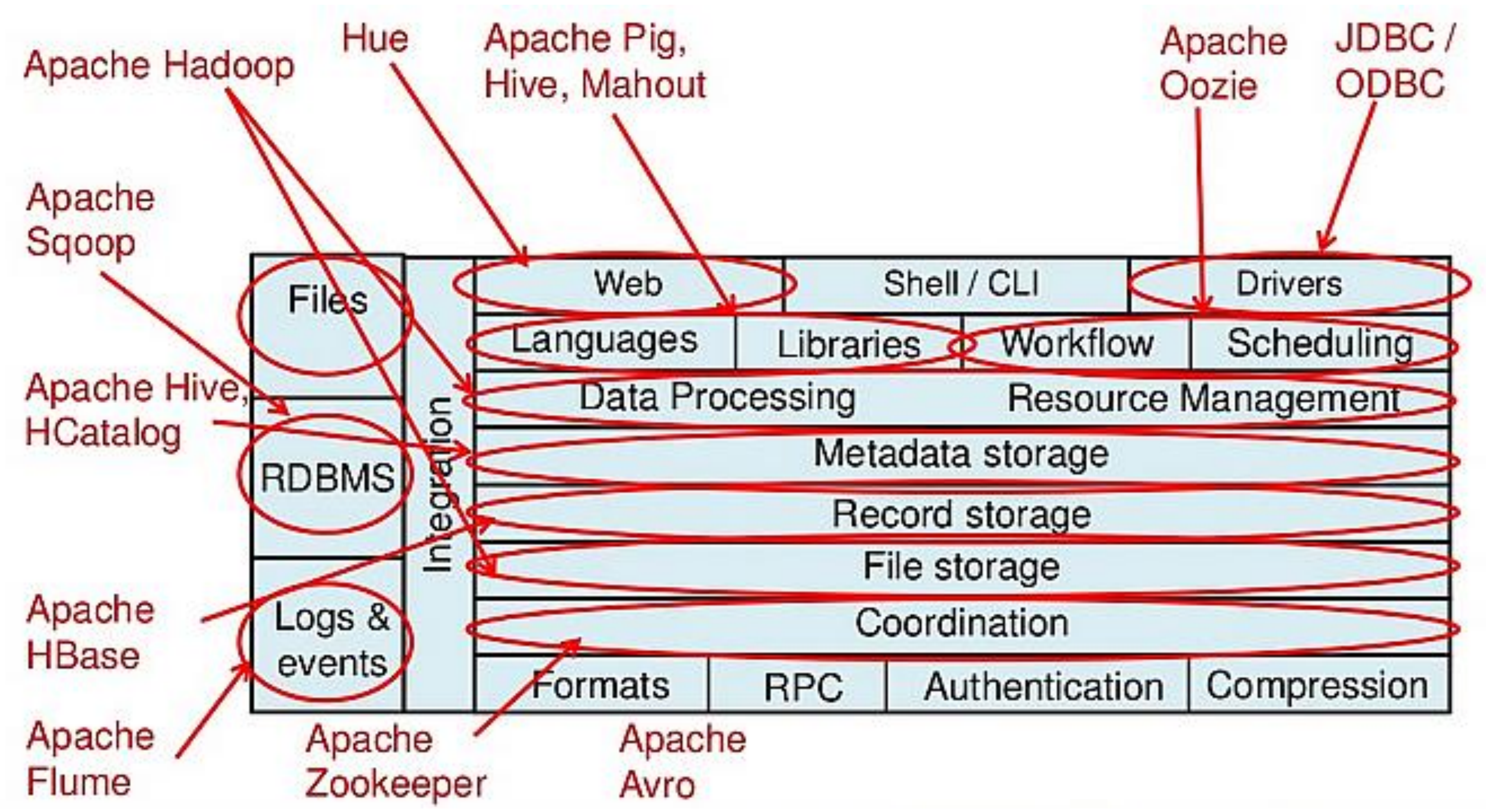
- 27 dec 2011: Apache Hadoop release 1.0.0

- June 2012: Facebook claim “biggest Hadoop cluster”,
  totalling more than 100 PetaBytes in HDFS

- 2013: Yahoo runs Hadoop on 42,000 nodes, computing about 500,000 MapReduce jobs per day

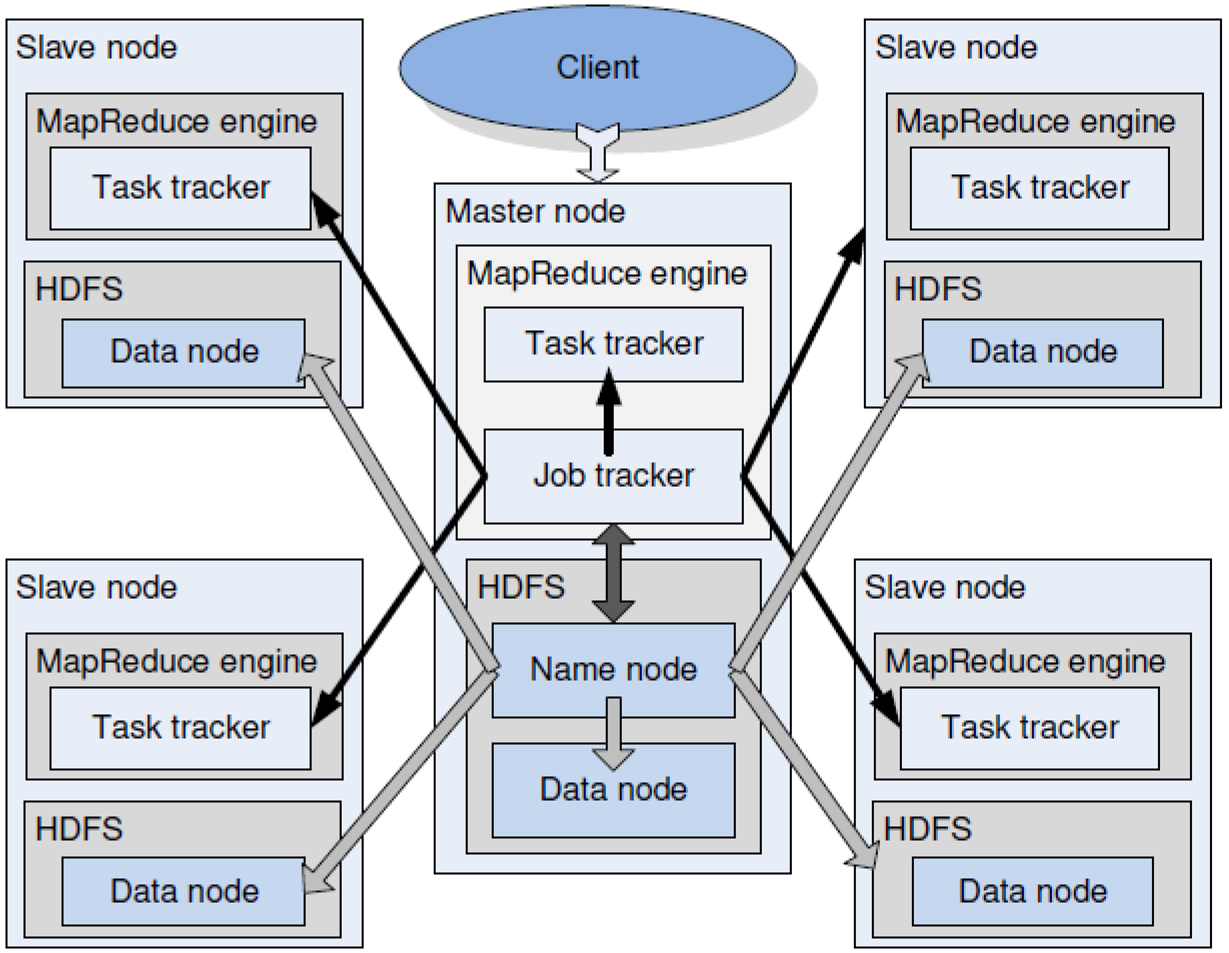
- 15 oct 2013: Apache Hadoop release 2.2.0 (YARN)

The wider Hadoop Ecosystem

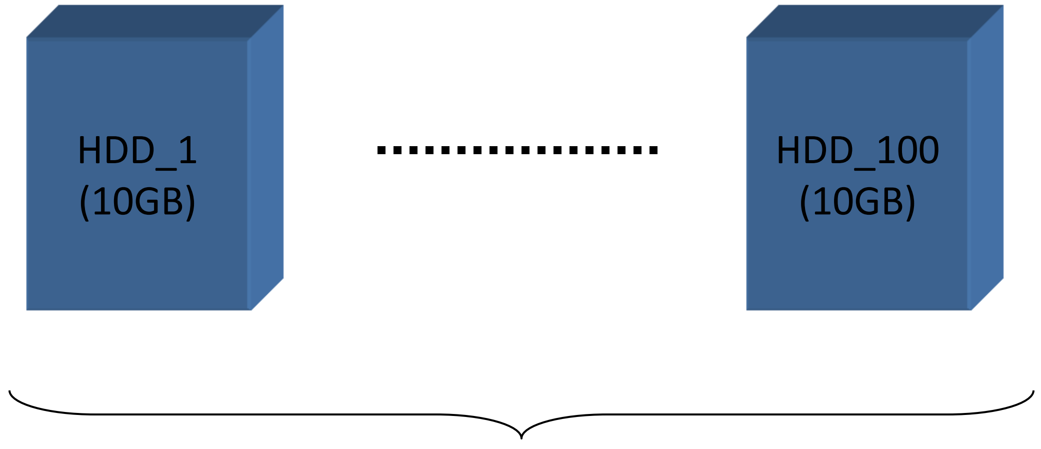


http://www.slideshare.net/cloudera/the-hadoop-stack-then-now-and-in-the-future-eli-collins-charles-zedlewski-cloudera

Apache Hadoop

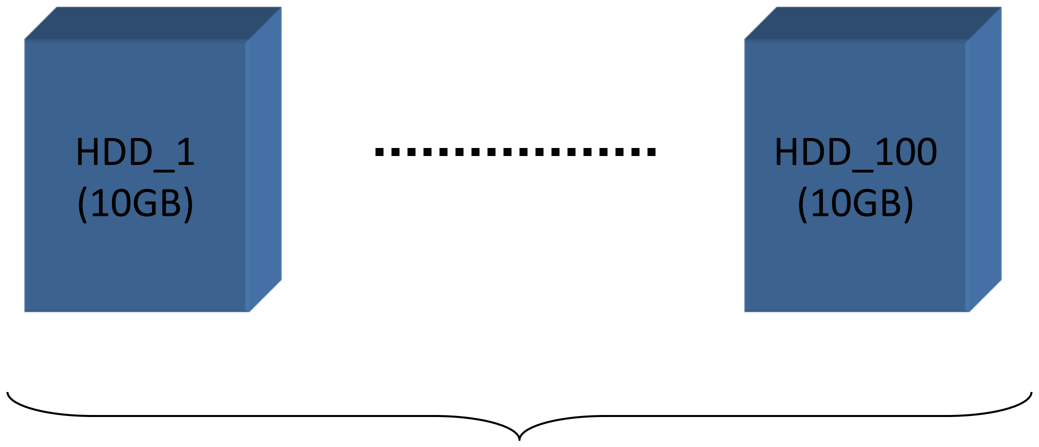


Hadoop Use-Case



1 TB of Data

Hadoop Use-Case

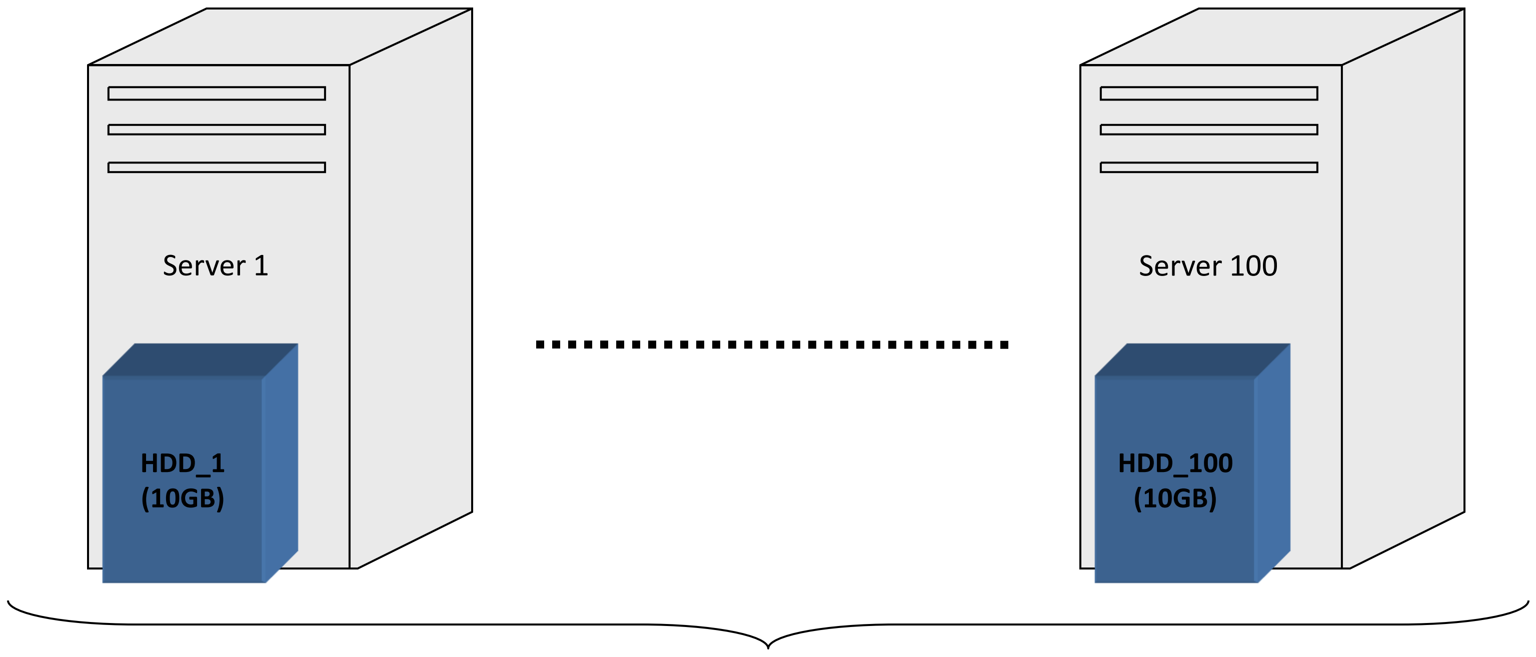


1 TB of Data in 100 HDDs, but how many computers should there be?

When N=1, reading 1 TB requires 2.5 <u>HOURS</u>.

What should N be in order to give us appreciable speed-up on reads?

Hadoop Use-Case



Given: 10 GB per drive = 10,000,000,000 bytes per drive

100, 10 GB drives = 1 TB = 1,000,000,000,000 bytes

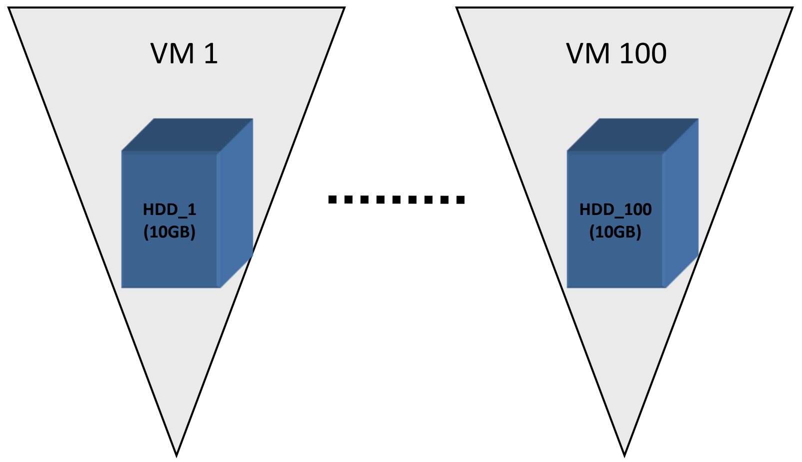
Read rate is 100 MB/second

Full 1 TB of data can be read in 100 seconds :

10 GB / 100 MB per second = 10,000,000,000 / 100,000,000 = 100 seconds to read one drive.
We read all 100 drives in parallel, and the computers can process the data read in parallel.

This is the architecture in which Hadoop shines because not only is the data read in parallel, it is processed in parallel as well!

Introduction to Hadoop and Big Data



Using Virtual Machines (VMs): managed by a cloud to host our Hadoop infrastructure and to run our MapReduce jobs.

VMs provide us with elastic resources in terms of

- Compute power : Number of VMs, Number of CPUs per VM
- Storage size
- Memory size

What is Hadoop?

- Distributed computing part of Nutch
    - Developed by Yahoo engineers, Nutch was a web search engine
- Provides tools that abstract into building blocks:
    - data storage, data analysis, machine coordination
- Storage system provided by Hadoop Distributed File System (HDFS)
    - Developed by Google (GFS later renamed to HDFS)
    - Protects against data loss from hardware failure
- Analysis provided by MapReduce
    - Developed by Google - Facilitates computation across multiple data sources
- The term Hadoop often refers to the full Hadoop ecosystem of tools
    - MapReduce, HDFS, Hive, Hbase, ZooKeeper, etc.

What is MapReduce?

- Developed by Google (2004)

- Provides data analysis capability of Hadoop

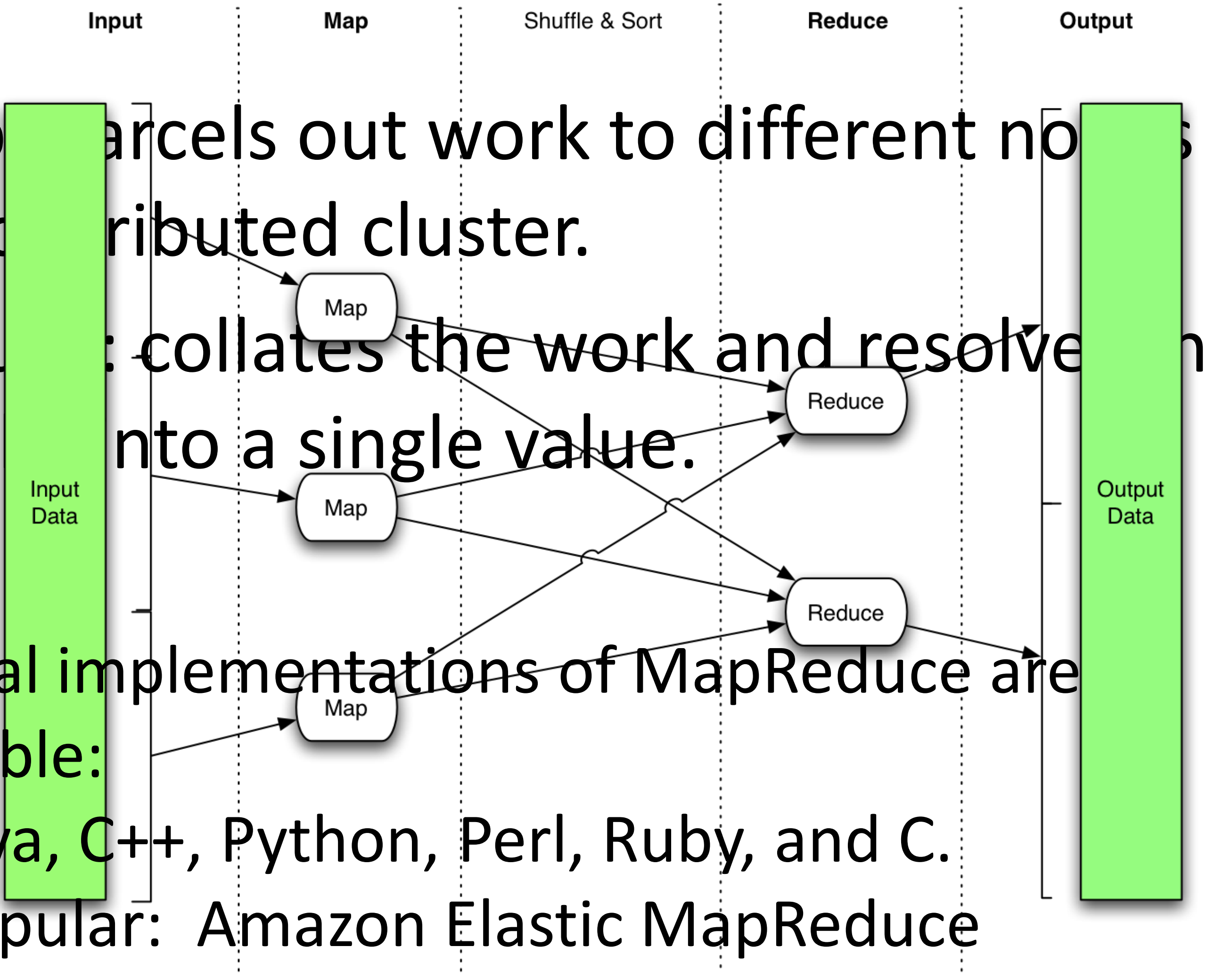
- Facilitates computation across multiple data sources

- Leverages *<u>data locality</u>* for performance
    - Data is co-located with compute node
    - Network bandwidth is at a premium
    - Avoid copying data around

- Has built-in reliability

– Google uses it for indexing Web pages

MapReduce



- Map arcels out work to different no s in the d ributed cluster.
- Redu : collates the work and resolve he resu nto a single value.

Several implementations of MapReduce are available:

Java, C++, Python, Perl, Ruby, and C.

Popular: Amazon Elastic MapReduce

What is MapReduce?

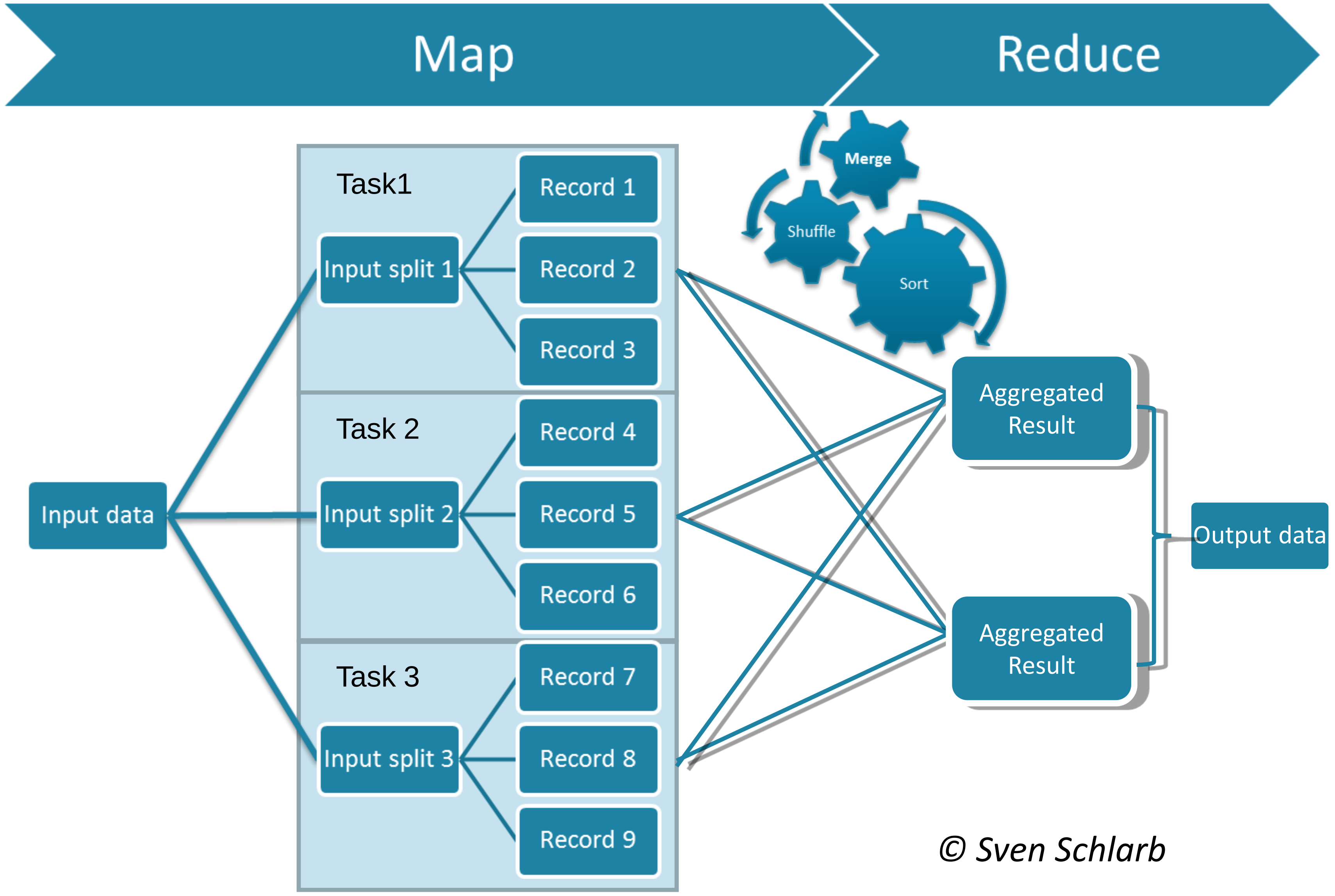
Map phase

- Map key-value input pairs to key-value output pairs
- Specify any processing to be performed on the key-value pairs
- Good time to perform data conditioning
    - drop any bad records
    - filter out bad or erroneous looking values
- Sorts and groups key-value pairs by key before sending the output to Reduce

Reduce phase

- Map key-value input pairs to key-value output pairs, just as in Map phase
- Key-value input pairs must match key-value output pairs from map function
- Specify processing to perform on data collected from distributed map nodes

MapReduce in a nutshell



© Sven Schlarb

MapReduce

- Batch query processor
- Assumes that the entire dataset is processed
- Suitable for write once/read many (worm) applications
- Not good if datasets need to get updated
  - Relational database is appropriate for this, not MapReduce
- Works well on unstructured data
  - Datasets with no particular internal structure
  - E.g. log files, other plaintext files
- Interprets data at processing time
  - Schema-free
  - Data not normalized because relies on operations to be local

Reference: Hadoop: The Definitive Guide, by Tom White

What is HDFS?

HDFS = Hadoop Distributed File System

- Storage system part of Hadoop
  - Developed by Google (2004)
  - Originally named Google File System (GFS)
  - Later renamed to HDFS
  - Protects against data loss from hardware failure

HDFS

- Stores very large files (100s of MB, GB, TB)
- Provides high performance streaming data access
- Uses off-the-shelf, non-custom, hardware
- Continues working without noticeable interruption if failure
- Stores replicas of blocks to facilitate recovery from hardware errors

Reference: Hadoop: The Definitive Guide, by Tom White

HDFS

- Two Types of Nodes

1. Namenodes
   - Masters: Manage filesystem namespace, Maintain filesystem tree
   - Maintain metadata for all files and directories in the tree
   - Maintain list of datanodes on which all blocks of a given file are located
   - Resilient to failure
     - Without namenode, cannot use the filesystem
2. Datanodes
   - Slaves to namenodes
   - Store and retrieve blocks when told
   - Reports to namenode periodically with lists of blocks they are storing
   - Compute checksums over blocks
   - Report checksum errors to namenodes

Reference: Hadoop: The Definitive Guide, by Tom White

HDFS is not good for

- Applications requiring low-latency access to data
- Lots of small files
  - Lots of small files mean lots of metadata to hold in the namenode’s memory
- Multiple writers
- Updates to offsets within the file

Reference: Hadoop: The Definitive Guide, by Tom White

Science and Engg, Legacy, Social computing etc.

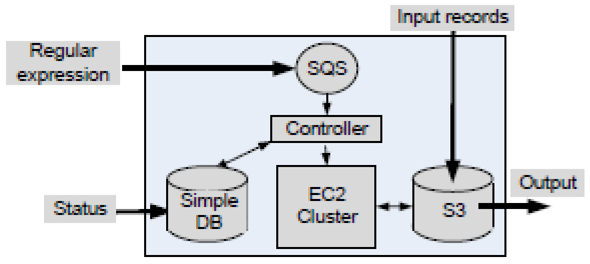
USE-CASES AND APPLICATION DOMAINS

Case Study: GrepTheWeb

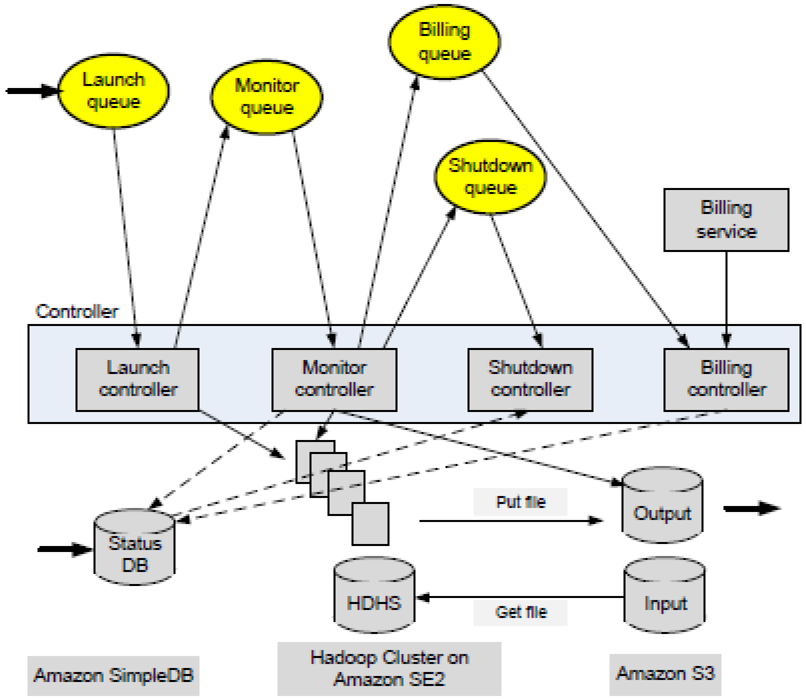
- The application illustrates the means to
  - create an on-demand infrastructure.
  - run it on a massively distributed system in a manner that allows it to run in parallel and scale up and down, based on the number of users and the problem size.
- GrepTheWeb
  - Performs a search of a very large set of records to identify records that satisfy a regular expression.
  - It is analogous to the Unix *grep* command.
  - The source is a collection of document URLs produced by the Alexa Web Search, a software system that crawls the web every night.
  - Uses message passing to trigger the activities of multiple controller threads which launch the application, initiate processing, shutdown the system, and create billing records.

(a) The simplified workflow showing the inputs:
- the regular expression.
- the input records generated by the web crawler.
- the user commands to report the current status and to terminate the processing.

(b) The detailed workflow. The system is based on message passing between several queues; four controller threads periodically poll their associated input queues, retrieve messages, and carry out the required actions



(a)



(b)

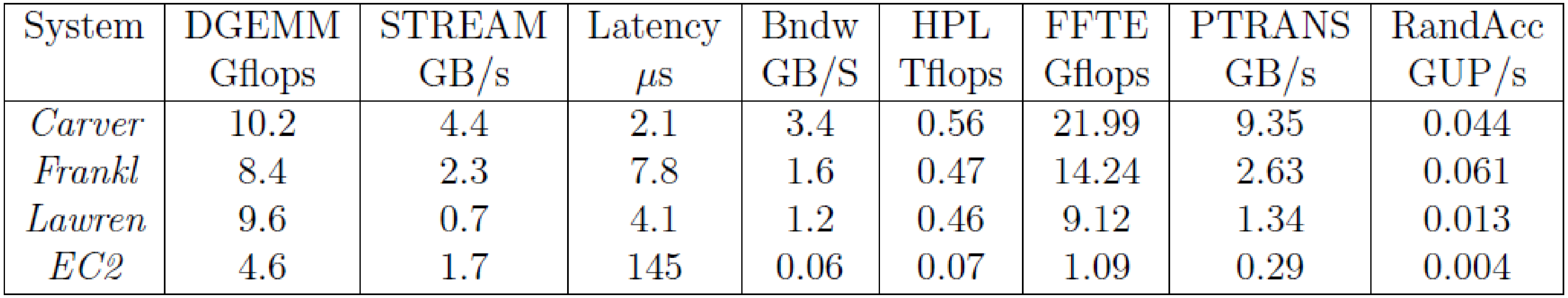
Clouds for science and engineering

- The generic problems in virtually all areas of science are:
  - Collection of experimental data.
  - Management of very large volumes of data.
  - Building and execution of models.
  - Integration of data and literature.
  - Documentation of the experiments.
  - Sharing the data with others; data preservation for a long periods of time.
- All these activities require “big” data storage and systems capable to deliver abundant computing cycles.
- Computing clouds are able to provide such resources and support collaborative environments.

Online data discovery

- Phases of data discovery in large scientific data sets:
  - recognition of the information problem.
  - generation of search queries using one or more search engines.
  - evaluation of the search results.
  - evaluation of the web documents.
  - comparing information from different sources.
- Large scientific data sets:
  - Biomedical and genomic data from the National Center for Biotechnology Information (NCBI).
  - astrophysics data from NASA.
  - atmospheric data from the National Oceanic and Atmospheric Administration (NOAA) and the National Center for Atmospheric Research (NCAR).

High performance computing on a cloud



System	DGEMM Gflops	STREAM GB/s	Latency μs	Bndw GB/S	HPL Tflops	FFTE Gflops	PTRANS GB/s	RandAcc GUP/s
Carver	10.2	4.4	2.1	3.4	0.56	21.99	9.35	0.044
Frankl	8.4	2.3	7.8	1.6	0.47	14.24	2.63	0.061
Lawren	9.6	0.7	4.1	1.2	0.46	9.12	1.34	0.013
EC2	4.6	1.7	145	0.06	0.07	1.09	0.29	0.004

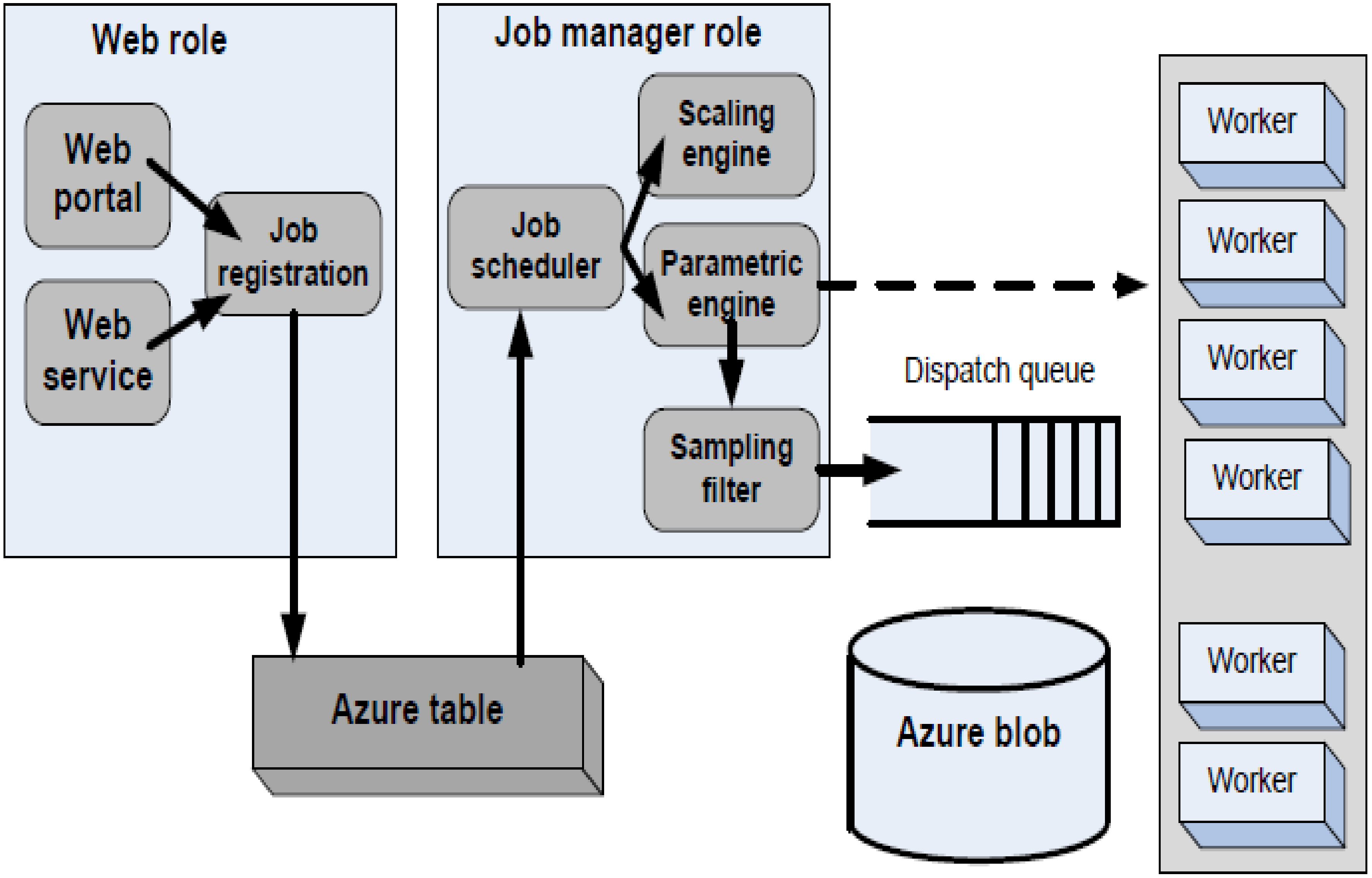
600 codes.

- Conclusion – communication-intensive applications are affected by the increased latency and lower bandwidth of the cloud. The low latency and high bandwidth of the interconnection network of a supercomputer cannot be matched by a cloud.

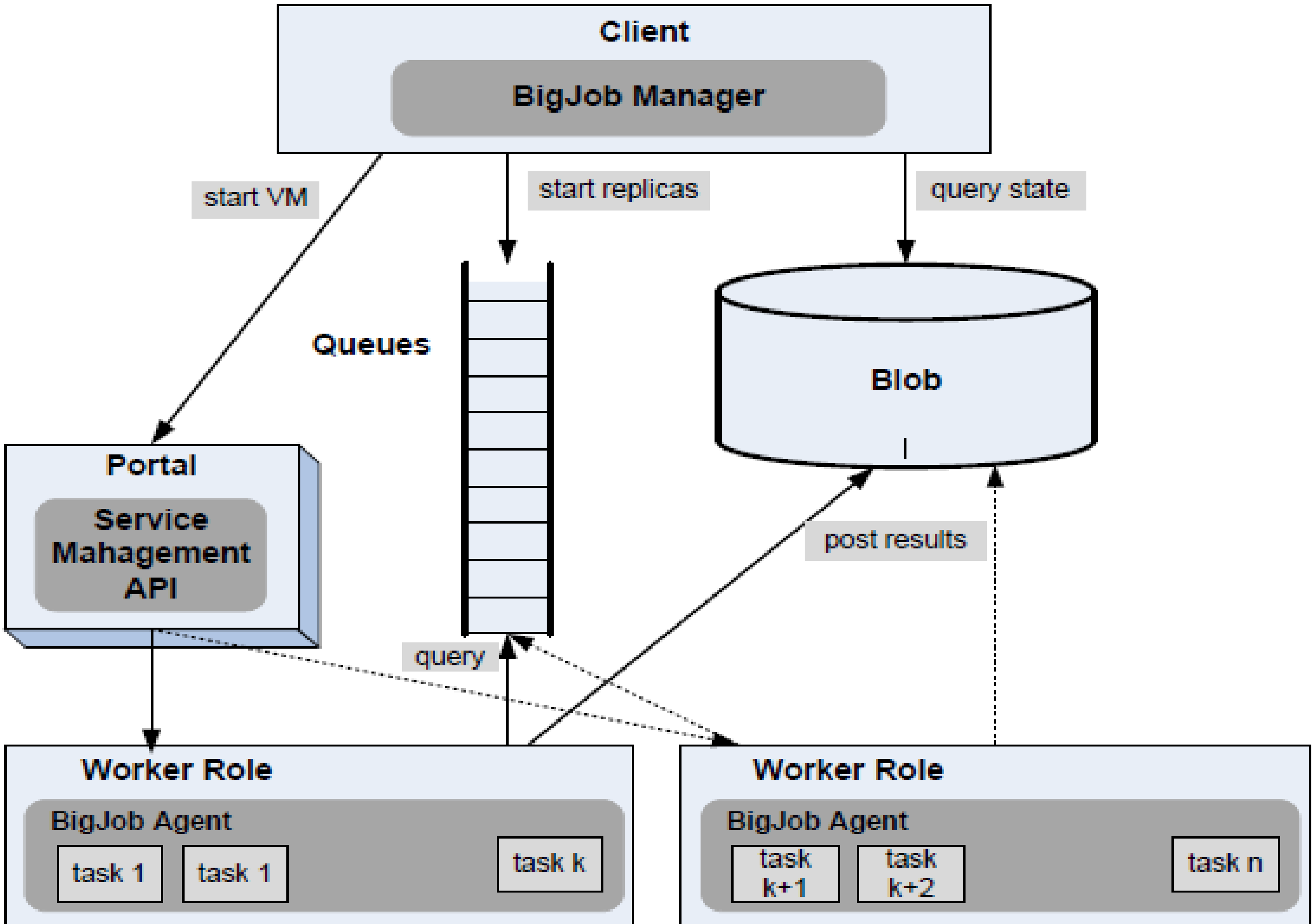
Legacy applications on the cloud

- Is it feasible to run legacy applications on a cloud?
- Cirrus - a general platform for executing legacy Windows applications on the cloud.
  - A Cirrus job –
    - a prologue,
      - sets up the running environment
    - commands, and
      - sequences of shell scripts
    - parameters.

Cirrus


Web role
Web portal
Web service
Job registration
Job manager role
Scaling engine
Job scheduler
Parametric engine
Sampling filter
Dispatch queue
Worker
Worker
Worker
Worker
Worker
Worker
Azure table
Azure blob

Execution of loosely-coupled workloads using the Azure platform



Legacy applications on the cloud

- BLAST
  - a biology code which finds regions of local similarity between sequences
  - used to infer functional and evolutionary relationships between sequences and identify members of gene families
- Azure-BLAST

Social computing and digital content

- Networks allowing researchers to share data and provide a virtual environment supporting remote execution of workflows are domain specific:
  - MyExperiment for biology.
  - nanoHub for nanoscience.
- Volunteer computing - a large population of users donate resources such as CPU cycles and storage space for a specific project:
  - Mersenne Prime Search
  - SETI@Home,
  - Folding@home,
  - Storage@Home
  - PlanetLab
- Berkeley Open Infrastructure for Network Computing (BOINC)
  - middleware for a distributed infrastructure suitable for different applications.

THANKS