

INFS 3204/7204 Service-Oriented Architecture



A/Prof Heng Tao SHEN
ITEE, UQ
Semester 2, 2011

M9: Cloud Computing Advance

1

M9 Topics

- Core Technologies
 - Distributed File System
 - MapReduce
- Commercial Platforms
 - Windows Azure
 - Google AppEngine
 - Amazon EC2
- Open Source Platforms
 - Hadoop

2

Distributed File Systems

- Google File System (GFS)
 - A scalable distributed file system for **large distributed data-intensive** application
 - A master-worker paradigm
 - Implemented in Linux

3

GFS – Chunks & Chunkservers

- Files are divided into fixed size (64MB) chunks
- Chunkservers store chunks on local disks
- Each chunk is replicated on multiple (3 replica by default) chunkservers for reliability

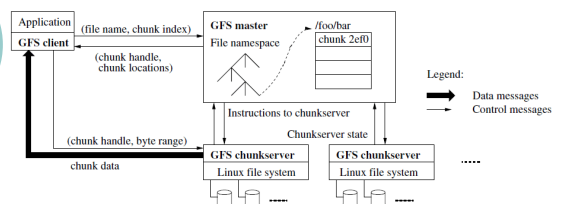
4

GFS - Master

- Maintains all file system metadata stored in memory for fast operations
- Periodically scans through its state at background for chunk garbage collection, re-replication in the presence of chunkserver failures and chunk migration for load-balancing
- Periodically communicates with chunkserver in HeartBeat messages to give instructions and collect its state

5

GFS – Architecture



A GFS cluster consists of

- a single master
- multiple chunkservers
- accessed by multiple clients

6

GFS – Communication Process

- Client sends the master a request containing the file name and chunk index
- Master replies with the corresponding chunk handle and locations of the replica
- Client caches this information using file name and chunk index as the key
- Client sends a request to the closest replica to corresponding Chunkservers
- Further reads on the same chunk require no more client-master interaction until the cached information expires or the file is reopened

7

MapReduce - Motivation

- Problem
 - Large-scale computing requires computations to be distributed across machines in order to finish in a reasonable amount of time
- Issues involved
 - Parallelisation
 - Data distribution
 - Failure-tolerance
 - Load balancing

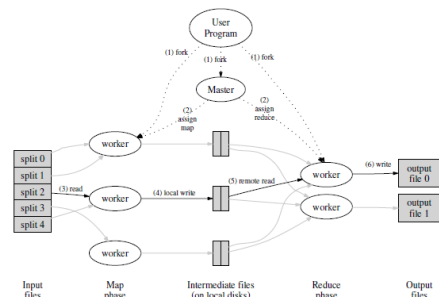
8

MapReduce – Programming Model

- Input: a set of input key/value pairs
- Output: a set of output key/value pairs
- Two main functions (written by users)
 - Map:
 - takes an input pair and produces a set of intermediate key/value pairs
 - E.g. `map(k1, v1) → list(k2, v2)`
 - Reduce:
 - takes an intermediate key and a set of values for the key and merges together these values to form a possibly smaller set of values
 - E.g. `reduce(k2, list(v2)) → list(v2)`

9

MapReduce – Flowchart



Dean, J., Ghemawat, S. MapReduce: Simplified Data Processing on Large Clusters. OSDI 2004.

10

MapReduce- Execution Steps

- Input files are split into **M** pieces, typically 16-64 MB per piece
- **Master** picks idle **workers** and assigns each one map/reduce task
- A **map** task **worker** reads the contents of the corresponding input split. It parses the key/value pairs out of the input split, passes them to the Map function and produces intermediate key/value pairs in buffer
- Periodically the buffered pairs are written to **local disk**, partitioned into **R** regions. The locations of these buffered pairs are passed to the master responsible for forwarding these locations to the reduce tasks
- When a reduce worker is notified by the master, it uses remote procedure calls to read the buffered data from the local disks of the map workers. When it reads all the intermediate data, it sorts it by the intermediate keys
- The reduce worker then passes the key and the corresponding set of intermediate values to the user's Reduce function. The output of the Reduce function is appended to a final output file for this reduce task
- When all map and reduce tasks have been completed, the output of MapReduce is **R** output files, one per reduce task

11

MapReduce – Word Count Example

- **Map**(string key, string value):


```

//key: document name
//value: document contents
for each word w in value:
    EmitIntermediate(w, "1")
            
```
- **Reduce**(string key, Iterator values) :


```

//key: a word
//values: a list of counts
Int result = 0;
For each v in values:
    result+= ParseInt(v);
Emit(AsString(result))
            
```

12

MapReduce – Inverted Index

- **Map**(string key, string value):
//key: document ID
//value: document contents
for each word w in value:
EmitIntermediate(<word, document_ID>)
- **Reduce**(string key, Iterator values) :
//key: a word
//values: a list of document_IDs
list(document_ID) ← Sort(values);
Emit(<word, list(document_ID)>)

13

MapReduce – Fault Tolerance

- Worker failure
 - Master pings every worker periodically
 - No response → the worker as failed
 - Map worker failure:
 - The completed task has to be re-executed by another worker, because the output is stored on the local disk, which will not be accessible due to the failure
 - Reduce worker failure:
 - No need to re-execute the completed task, because the output is stored in a global file system

14

MapReduce – Fault Tolerance

- Master failure
 - Can write periodic checkpoints for recovery
 - Or abort the MapReduce computation if it happens

15

Platforms- Windows Azure

- Commercially available in 2/2010
<http://www.microsoft.com/windowsazure/>
- The **Windows Azure Platform** is a Microsoft [cloud platform](#) used to build, host and scale web applications through Microsoft datacenters
- Windows Azure Platform is thus classified as [platform as a service](#) and forms part of Microsoft's [cloud computing](#) strategy, along with their [software as a service](#) offering, [Microsoft Online Services](#)

16

Windows Azure

- The platform consists of various on-demand services hosted in Microsoft data centers and commoditized through three product brands
 - [Windows Azure](#) (an operating system providing scalable computing and storage facilities)
 - [SQL Azure](#) (a cloud-based, scale-out version of [SQL Server](#))
 - Windows Azure [AppFabric](#) (a collection of services supporting applications both in the cloud and on premise)

17

SQL Azure

- Similar to an instance of SQL Server on your premises, SQL Azure Database exposes a tabular data stream (TDS) interface for Transact-SQL-based database access
- This allows your database applications to use SQL Azure Database in the same way that they use SQL Server
- Because SQL Azure Database is a service, administration in SQL Azure Database is slightly different

18

Windows Azure AppFabric

- **AppFabric** refers to two related Microsoft products
 - Windows Server AppFabric
 - a set of integrated technologies that make it easier to build, scale and manage Web and composite applications on IIS
 - Windows Azure AppFabric
 - helps developers connect apps and services between Windows Azure and on-premises deployments
 - provides a comprehensive cloud middleware platform for developing, deploying and managing applications on the Windows Azure platform

19

Platforms – Google AppEngine

- Google App Engine is “a system that exposes various pieces of Google’s scalable infrastructure so that you can write server-side applications on top of them”

- Reference: Google.com. Google App Engine Campfire One Transcript. <http://code.google.com/appengine/articles/cf1-text.html>, Accessed on 9 Feb, 2009

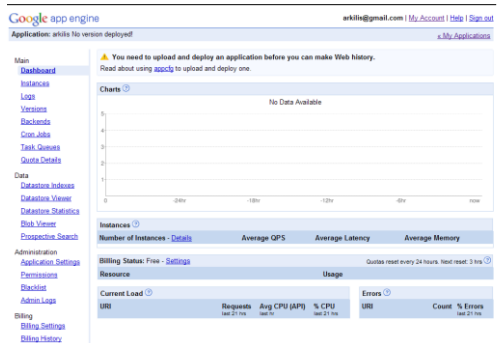
20

Google AppEngine

- Simply this is a platform which allows users to run and host their web applications on Google’s infrastructure
- These applications are easy to build, easy to maintain and easy to scale whenever traffic and data storage needed
- By using Google’s App Engine, there are no servers to maintain and no administrators needed
- The idea is user just to upload his application and it is ready to serve its own customers

21

Google AppEngine Administrator Console



Platforms - Amazon Elastic Compute Cloud (EC2)

- Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud
- It is designed to make web-scale computing easier for developers

23

Amazon Elastic Compute Cloud (EC2) (Paper Definition)

- Amazon’s Elastic Compute Cloud (EC2) is an ‘infrastructure as a service’ - a web service interface that allows customers to rent computers on which to run their computer applications. Using EC2, software developers do not need to have their own servers to run, test, or deploy their applications
- Furthermore, they are isolated from common failure scenarios and pay only for the capacity they actually use. EC2 is a part of a larger blanket of web services offered, Amazon Web Services (AWS)

24

Comparisons

- distinguished based on the cloud system software's level of abstraction and the level of management of the resources

25

Google AppEngine

- Application domain-specific platforms which are targeted exclusively at traditional Web applications, enforcing an application structure of clean separation between a stateless computation tier and a stateful storage tier
- Impressive automatic scaling and high-availability mechanisms, and the proprietary MegaStore data storage

26

EC2

- Much like physical hardware
- Users can control nearly the entire software stack, from the kernel upward
- This low level makes it inherently difficult for Amazon to offer automatic scalability and failover because the semantics associated with replication and other state management issues are highly application-dependent

27

Azure

- Intermediate between application frameworks like AppEngine and hardware virtual machines like EC2
- Written using the .NET libraries, and compiled to the Common Language Runtime, a language-independent managed environment
- Significantly more flexible than AppEngine's, but still constrains the user's choice of storage model and application structure

28

Hadoop

- <http://hadoop.apache.org/>
- Hadoop is support by Apache

29

Hadoop Ecosystem



30

Hadoop - Pig



- Pig is a platform for **analyzing large data sets** that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs
- The salient property of Pig programs is that their structure is **amenable to substantial parallelization**, which in turns enables them to handle very large data sets
- <http://pig.apache.org/>

31

Hadoop - Hive



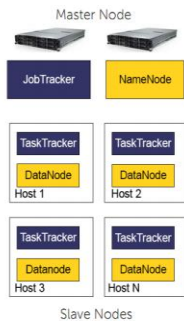
- Hive is a **data warehouse** system for Hadoop that facilitates easy data summarization, ad-hoc queries, and the analysis of large datasets stored in Hadoop compatible file systems
- Hive provides a mechanism to project structure onto this data and query the data using a SQL-like language called HiveQL. At the same time this language also allows traditional map/reduce programmers to plug in their custom mappers and reducers when it is inconvenient or inefficient to express this logic in HiveQL

32

HDFS & MapReduce



- **Hadoop Distributed File System**
 - A scalable, Fault tolerant, High performance distributed file system capable of running on Sun hardware
 - Hadoop cluster with 3 nodes minimum
 - Data divided into 64MB or 128MB blocks, each block replicated 3 times (default)
 - No 15k RPM disks or RAID required
 - **NameNode** holds filesystem metadata
 - Files are broken up and spread over the **DataNodes**
- **Hadoop Map Reduce**
 - Software framework for distributed computation
 - Input | Map() | Copy/Sort | Reduce() | Output
 - **JobTracker** schedules and manages jobs
 - **TaskTracker** executes individual map() and reduce() tasks on each cluster node



Slave Nodes

33

Hadoop Distributed File System

NameNode

- Manages file system NameSpace
 - Maps a file name to set of blocks
 - Maps a block to the DataNodes where it resides
- Cluster configuration management
- Replication engine for blocks
- Metadata management
 - Metadata are in main memory
 - List of files, list of blocks in each file
 - List of DataNode in each block
 - File attributes, replication factor...
- Transaction Log
 - Records for file creation, file deletion...

DataNode

- Block Server
 - Stores data in the local file system
 - Stores the metadata of a block
 - Serves data and metadata to clients
- Block Report
 - Periodically sends a report of all existing blocks to the NameNode
- Pipeline of Data
 - Forwards data to other specified DataNodes

34

MapReduce



Map Phase

- Raw data analyzed and converted to name/value pair

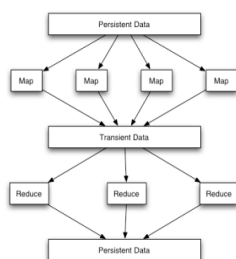
Shuffle Phase

- All name/value pairs are sorted and grouped by their keys

Reduce Phase

- All values associated with a key are processed for results

Input | Map() | Copy/Sort | Reduce() | Output



35

HBase



- Clone of Big Table (Google)
- Implemented in Java (Clients : Java, C++, Ruby...)
- Data is stored "Column-oriented"
- Distributed over many servers
- Tolerant of machine failure
- Layered over HDFS
- Strong consistency

- It's not a relational database (No joins)
- Sparse data – nulls are stored for free
- Semi-structured or unstructured data
- Data changes through time
- Versioned data
- Scalable – Goal of billions of rows x millions of columns

		Table			
Row	Timestamp	Animal	Size	Repair	Cost
Region	Enclosure1	12	Zebra	Medium	1000€
		11	Lion	Big	
	Enclosure2	13	Monkey	Small	1500€

(Table, Row_Key, Family, Column, Timestamp) = Cell (Value)

36

Companies that Use Hadoop

- Hadoop is a core part of the computing infrastructure for many web companies, such as
 - **Facebook**
 - Yahoo
 - LinkedIn
 - Twitter
 - ...

37

Facebook Messages

- The new version of Facebook Messages combines messages, email, chat & SMS into one real-time conversation
- Messages by nature are a write-dominated workload, and combining the different streams of messages only makes it more so

All your messages together



38

Scale of Data

- Now
 - Almost 300 TB / month
 - Compressed and replicated size

39

New Requirements of Messages

- High write throughput
 - Millions of messages /day
 - Billions of instant messages /day
 - Each message could be written several times

40

New Requirements of Messages

- Large tables
 - Mailbox would grow indefinitely
 - Since messages wouldn't be deleted unless done by the users
 - This means an ever-growing of list thread and messages per user
 - Majority of messages would not be read but must be available at all times and with low latency, so archiving would be difficult
 - Random write performance will typically decrease in a system like MySQL as the number of rows in the table increases

41

New Requirements of Messages

- Data migration
 - New messaging product has a new data model, so existing data have to be migrated
 - The ability to perform large scans, random access, and fast bulk imports would help to reduce the time spent migrating users to the new system

42

Why MySQL cannot Handle it?

- Problems of MySQL
 - Though proven to be very good random read performance, it typically suffers from low random write throughput
 - It is difficult to scale up MySQL clusters rapidly while maintaining good load balancing and high uptime
 - Administration of MySQL clusters requires a relatively high management overhead and they typically use more expensive hardware

43

Facebook Solution

- Hbase
 - Message metadata and indices
 - Search index
 - Small message bodies
- Haystack (photos store)
 - Attachments
 - Large messages

44

Why HBase

- High write throughput
 - High write-dominated workload
- Horizontal scalability
- Automatic failover
 - One machine goes down another must take over its place
- Benefits of HDFS

45

Facebook Solution Hadoop + HBase

- HBase
 - a bigtable-like (modeled after Google's Bigtable) structured storage for Hadoop HDFS filesystem
 - targeted at random read and write access of (fairly-) structured data
 - provides a highly consistent, high write-throughput key-value store
 - supports large tables, on the order of billions of rows and millions of columns
 - Facebook uses HBase in a real-time fashion to store and serve data for the Facebook messaging product
- Hadoop map-reduce is used extensively to migrate all the existing data from the existing databases into HBase and transform it into the desired format

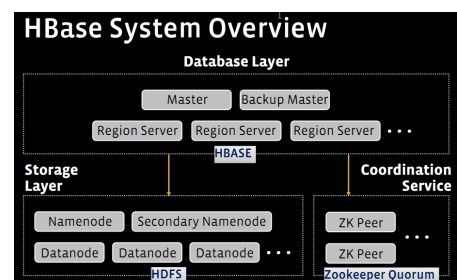
46

What HDFS can offer

- Fault tolerance
- Scalability
- Checksums
- Map Reduce
- HDFS already tested at Facebook
 - Hive warehouse – petabyte scale cluster

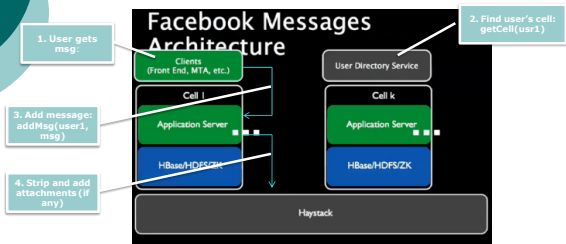
47

Facebook core technologies – HBase Overview



48

Facebook Messages architecture



49

Summary

- This week:
 - Core Technologies
 - Distributed File System
 - MapReduce
 - Platforms
 - Azure
 - Google AppEngine
 - EC2
 - Hadoop
- Next Lecture:
 - **Guest Lecture on Workflow**

50