CS-580K/480K Advanced Topics in Cloud Computing

16. Big Data Programming Models and Frameworks
Where are we?

Cloud Platforms

Cloud Applications

New databases technologies (e.g., Key-value store, and Object store)

New Programming Models/Framework (e.g., serverless, microservices, Hadoop, and Spark)
Example – Word Count

• Counting the number of occurrences of each word in a large collection of documents, e.g.:

A wonderful serenity has taken possession of my entire soul, like these sweet mornings of spring which I enjoy with my whole heart. I am alone, and feel the charm of existence in this spot, which was created for the bliss of souls like mine. I am so happy, my dear friend, so absorbed in the exquisite sense of mere tranquil existence, that I neglect my talents. I should be incapable of drawing a single stroke at the present moment; and yet I feel that I never was a greater artist than now. When, while the lovely valley teems with vapour around me, and the meridian sun strikes the upper surface of the impenetrable foliage of my trees, and but a few stray gleams steal into the inner sanctuary, I throw myself down among the tall grass by the trickling stream; and, as I lie close to the earth, a thousand unknown plants are noticed by me: when I hear the buzz of the little world among the stalks, and grow familiar with the countless indescribable forms of the insects and flies, then I feel the presence of the Almighty, who formed us in his own image, and the breath of that universal love which bears and sustains us, as it floats around us in an eternity of bliss; and then, my friend, when darkness overspreads my eyes, and heaven and earth seem to dwell in my soul and absorb its power, like the form of a beloved mistress, then I often think with longing, Oh, would I could describe these conceptions, could impress upon paper all that is living so full and warm within me, that it might be the mirror of my soul, as my soul is the mirror of the infinite God! O my friend -- but it is too much for my strength -- I sink under the weight of the splendour of these visions! A wonderful serenity has taken possession of my entire soul, like these sweet mornings of spring which I enjoy with my whole heart. I am alone, and feel the charm of existence in this spot, which was created for the bliss of souls like mine. I am so happy, my dear friend, so absorbed in the exquisite sense of mere tranquil existence, that I neglect my talents. I should be incapable of drawing a single stroke at the present moment; and yet I feel that I never was a greater artist than now. When, while the lovely valley teems with vapour around me, and the meridian sun strikes the upper surface of the impenetrable foliage of my trees, and but a few stray gleams steal into the inner sanctuary, I throw myself down among the tall grass by the trickling stream; and, as I lie close to the earth, a thousand unknown plants are noticed by me: when I hear the buzz of the little world among the stalks, and grow familiar with the
A Straightforward Solution

How to make it run in parallel?

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>The</td>
<td>10</td>
</tr>
<tr>
<td>And</td>
<td>992</td>
</tr>
<tr>
<td>Word</td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
A Straightforward Solution

Any problems?
Data locality? Fault Resilience?

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>The</td>
<td>10</td>
</tr>
<tr>
<td>And</td>
<td>992</td>
</tr>
<tr>
<td>Word</td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
An Alternative – Word Count

• Split the “scanner” process into two parts:

Function1(String key, String value):

  // key: document name
  // value: document contents
  for each word w in value:
    EmitIntermediate(w, "1");

Function1 emits each word plus an associated count of occurrences (just ‘1’ in this simple example).
An Alternative – Word Count

• Split the “scanner” process into two parts:

Function2(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));
In Summary – word count

Input
- the quick brown fox
- the fox ate the mouse
- how now brown cow

Map
- Map
- Map
- Map

Shuffle & Sort
- the, 1
- brown, 1
- fox, 1
- how, 1
- now, 1
- brown, 1
- the, 1
- fox, 1
- the, 1

Reduce
- the, 1
- brown, 1
- fox, 1
- how, 1
- now, 1
- brown, 1
- the, 1
- fox, 1
- the, 1

Output
- brown, 2
- fox, 2
- how, 1
- now, 1
- the, 3
- ate, 1
- cow, 1
- mouse, 1
- quick, 1
Other Examples

• **Distributed Grep**: The map function emits a line if it matches a supplied pattern. The reduce function is an identity function that just copies the supplied intermediate data to the output.

• **Count of URL Access Frequency**: The map function processes logs of web page requests and outputs URL. The reduce function adds together all values for the same URL and emits a URL, total count pair.

• ...
What is Apache Hadoop?

- Open source software framework designed for storage and processing of large scale data on clusters of commodity hardware
- Created by Doug Cutting and Mike Carafella in 2005 and released by Yahoo in 2008 and later managed and maintained by Apache Software Foundation.
Who Uses Hadoop?
Motivations for Hadoop

- What were the limitations of earlier large-scale computing?
- What requirements should an alternative approach have?
- How does Hadoop address those requirements?
How much data?

- Facebook
  - 500 TB per day
- Yahoo
  - Over 170 PB
- eBay
  - Over 6 PB

- Getting the data to the processors becomes the bottleneck
Distributed Systems

- Allow developers to use multiple machines for tasks to achieve high performance.
- However, programming on distributed systems is much more complex
  - Synchronizing data exchanges (MPI)
  - Managing a finite bandwidth
  - Controlling computation timing is complicated
Distributed Systems: Failures

“You know you have a distributed system when the crash of a computer you’ve never heard of stops you from getting any work done.” –Leslie Lamport

- Distributed systems must be designed with the expectation of failure.

- Detection of faults and quick, automatic recovery from them is a core architectural goal of distributed systems.
Distributed Systems: Data Storage

• Typically divided into Data Nodes and Compute Nodes
  • At compute time, data is copied to the Compute Nodes
  • Fine for relatively small amount of data
  • Very expensive for large amount of data

• Modern systems deal with far more data than was gathering in the past where separation of data and computation limits scalability
Requirements for Hadoop

- Easy to program
- Support partial failure
- Must be scalable
Core Hadoop Concepts

- Data is spread among the machines in advance
  - Perform computation where the data is already stored as often as possible
- Nodes should communicate as little as possible
  - A “shared nothing” architecture
- Applications are written in a high-level programming language
  - No network programming or temporal dependency


The Hadoop Ecosystem

- **Hadoop Common**
  - Contains Libraries and other modules

- **HDFS**
  - Hadoop Distributed File System

- **Hadoop YARN**
  - Yet Another Resource Negotiator

- **Hadoop MapReduce**
  - A programming model for large scale data processing
Hadoop Distributed File System
Overview

- Responsible for storing large data on the cluster, especially for low-cost commodity hardware
- HDFS works best with a smaller number of large files
  - Optimized for streaming reads of large files and not random reads
  - Files in HDFS are “write-once”
- Data files are split into blocks and distributed across the nodes in the cluster
- Each block is replicated multiple times
Architecture of HDFS

- A single **Namenode** and multiple **Datanodes**
- HDFS exposes a file system namespace
- The NameNode executes file system namespace operations (e.g., open, close, and rename files and directories).
- A file is split into one or more blocks and these blocks are stored in a set of DataNodes.
- The DataNodes, one per node, manage storage attached to the nodes and serve client requested reads and writes
- TCP/IP communication protocol
Data Replication and Fault Tolerance

- All blocks in a file except the last block are the same size.
- The blocks of a file are replicated for fault tolerance.
- The block size and replication factor are configurable per file.
- Serving reads from the nearest node.
- NameNode periodically receives a Heartbeat and a Blockreport from each of the DataNodes.
- Re-replication is triggered as needed.
  - An unavailable DataNode
  - A replica corrupted
  - A hard disk failure

```
Namenode (Filename, numReplicas, block-ids, ...)
/users/sameerp/data/part-0, r:2, {1,3}, ...
/users/sameerp/data/part-1, r:3, {2,4,5}, ...
```

```
Datanodes

1 2
3 4
5
2 5
3 5
4
```
MapReduce
MapReduce Overview

▪ A programming model for distributing computation across multiple nodes

▪ Consists of two functions
  ▪ Map-function
  ▪ Reduce-function

▪ Three stages
  ▪ The map stage
  ▪ The shuffle stage
  ▪ The reduce stage
The Mapper

- Reads data as key/value pairs
- Outputs zero or more key/value pairs

map: (K1,V1) ? list(K2,V2)
Shuffle and Sort

- Output from the mapper is sorted by key

- All values with the same key are guaranteed to go to the same machine
The Reducer

- Gets a list of all values associated with a key as input
  - Starts after all maps finish
  - Reducers operating on different keys can be executed simultaneously

- The reducer outputs zero or more final key/value pairs
  - Usually just one output per input key

reduce: (K2,list(V2)) ? list(K3,V3)
Revisit “Word Count”

Input
- the quick brown fox
- the fox ate the mouse
- how now brown cow

Map
- the, 1
- brown, 1
- fox, 1
- the, 1
- fox, 1
- the, 1
- how, 1
- now, 1
- brown, 1
- ate, 1
- mouse, 1
- cow, 1
- quick, 1

Shuffle & Sort

Reduce
- brown, 2
- fox, 2
- how, 1
- now, 1
- the, 3
- ate, 1
- cow, 1
- mouse, 1
- quick, 1

Output
Architecture of Hadoop

- Hadoop has a Master-Slave architecture for data storage and distributed data processing using MapReduce and HDFS methods.
- The Jobtracker is a coordinator
- Tasktracker runs actual code of job on the data blocks of input file.
- Self-healing: if the one map node fails, rerun the map task on another healthy node
Hadoop Tools

- Hive
  - Hadoop processing with SQL
- Pig
  - Hadoop processing with scripting
- Cascading
  - Pipe and Filter processing model
- HBase
  - Database model built on top of Hadoop
- Flume
  - Designed for large scale data movement
Shortcomings of Hadoop

- Forces your data processing into Map and Reduce
  - Other workflows missing include join, filter, flatMap, groupByKey, union, intersection, ...
- Read and write to Disk before and after Map and Reduce (stateless machine)
  - Not efficient for iterative tasks, i.e. Machine Learning
- Only Java natively supported
  - Support for other languages needed
- Processing Overhead
  - Hadoop cannot do in-memory calculations hence it incurs processing overhead.
One Solution is Apache Spark

- A new general framework, which solves many of the shortcomings of MapReduce
- It is capable of leveraging the Hadoop ecosystem, e.g. HDFS, YARN, HBase, ...
- Has many other workflows
  - ~30 efficient distributed operations
- In-memory caching of data (for iterative, graph, and machine learning algorithms, etc.)
- Native Scala, Java, Python, and R support
- Supports interactive shells for exploratory data analysis
- Spark API is extremely simple to use
Spark Uses Memory instead of Disk

Hadoop: Use Disk for Data Sharing

Spark: In-Memory Data Sharing
## Sort competition

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Size</td>
<td>102.5 TB</td>
<td>100 TB</td>
</tr>
<tr>
<td>Elapsed Time</td>
<td>72 mins</td>
<td>23 mins</td>
</tr>
<tr>
<td># Nodes</td>
<td>2100</td>
<td>206</td>
</tr>
<tr>
<td># Cores</td>
<td>50400 physical</td>
<td>6592 virtualized</td>
</tr>
<tr>
<td>Cluster disk throughput</td>
<td>3150 GB/s (est.)</td>
<td>618 GB/s</td>
</tr>
<tr>
<td>Network</td>
<td>dedicated data center, 10Gbps</td>
<td>virtualized (EC2) 10Gbps network</td>
</tr>
<tr>
<td>Sort rate</td>
<td>1.42 TB/min</td>
<td>4.27 TB/min</td>
</tr>
<tr>
<td>Sort rate/node</td>
<td>0.67 GB/min</td>
<td>20.7 GB/min</td>
</tr>
</tbody>
</table>

Spark, 3x faster with 1/10 the nodes

Sort benchmark, Daytona Gray: sort of 100 TB of data (1 trillion records)

Conclusion

- Hadoop (HDFS, MapReduce)
  - Provides an easy solution for processing of Big Data
  - Brings a paradigm shift in programming distributed system.
- Spark
  - Has extended MapReduce for in memory computations
  - for streaming, interactive, iterative and machine learning tasks
- Changing the World
  - Made data processing cheaper and more efficient and scalable
  - Is the foundation of many other tools and software
Sources

- Distributed Software Engineering David A. Wheeler George Mason University
- Apache Hadoop and Spark: Introduction and Use Cases for Data Analysis Afzal Godil Information Access Division, ITL, NIST
- Spark Training [http://spark-summit.org/2014/training](http://spark-summit.org/2014/training)