Abstract

MapReduce primary purpose is to facilitate data parallel processing. It is currently the method implemented by Google’s search engine and Amazon consumer base entity to cultivate their large product and informational requests. As the needs of a growing and evolving civilization changes their information supplements to balance its complexities, of those dataset processes that must adapt with efficiencies in computational methods and application development. We demonstrated such efficiency obtained from parallel processing by implementing a Map Reduce word count algorithm. By increasing the number of processes for a job we got a chance to see basic parallel operational function. While increasing the byte volume gives us insight into the functionality of Hadoop distributive file system and how it handles streaming data. The various integrated functions work on the data streams and generates different output information based on the query request. Simple functionalities were part of this experiment and showed only surface results. The Hadoop can utilize many different algorithms to accommodate the needs of the system. We employed a basic application that demonstrate fundamental mechanisms involved in controlling primitive data types. There are deeper concepts involving analysis of performance improvement using programming methodologies. These programming designs can impact physical components on the commodity computers that make up the clusters. It is our purpose at this point to evaluate the Hadoop framework central functionality.

Introduction

Hadoop is comprised of several subprojects that complete a system of functions that solve an array of internet problems. For our purposes we will focus on two essential elements namely the Hadoop Distributed File System (HDFS) and the MapReduce sequence processing mechanisms. The input data is transparent to the various framework operations. These two functions constitute the Hadoop infrastructure that supports a big parallel architecture to clarify specific information requirements from expansive volumes of connected and disconnected content matter. We will explore and evaluate some of it parts to gain deeper insights into its operation to revisit and do more extensive experimental in the future.

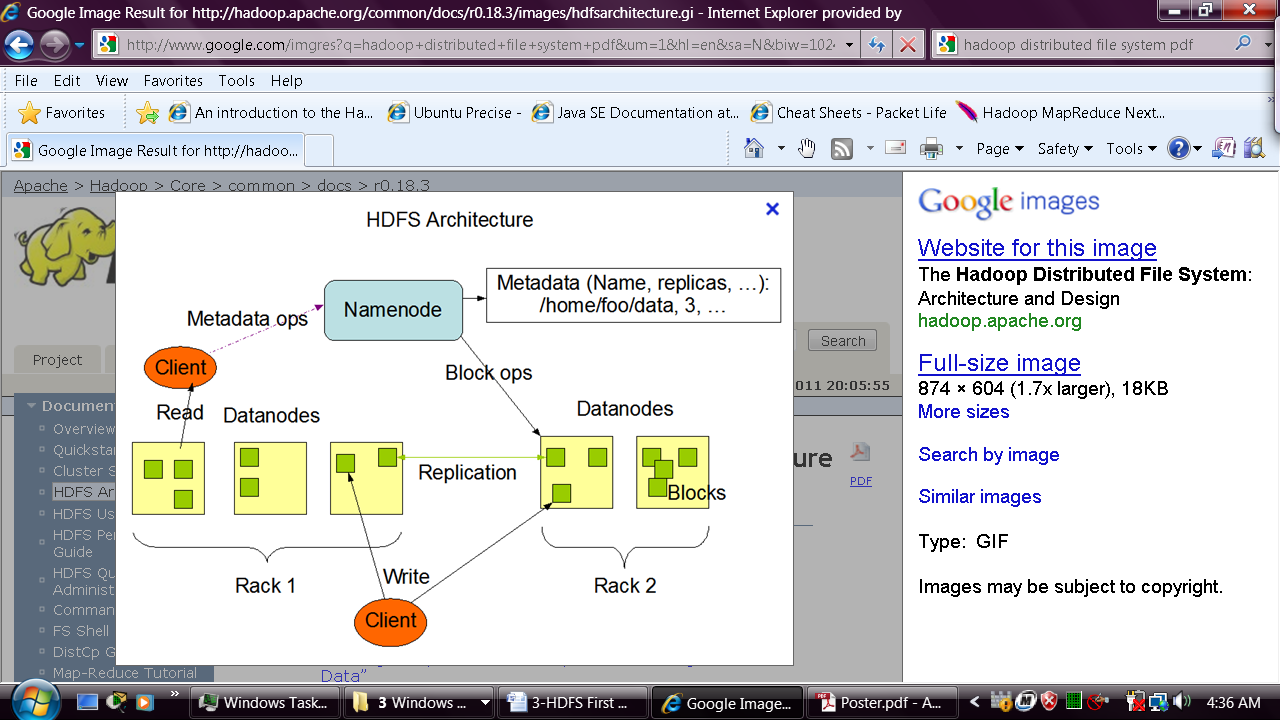
Hadoop Environment

The Hadoop can virtually be scaled to just about any size to begin investigating the various boundaries of its capabilities. The primary object that drives the frames clusters are NameNodes. The NameNode are central locations for information concerning the file system deployment within the Hadoop environment. An environment can have one or two NameNodes, configured to provide lower limits of redundancy between NameNodes. These NameNode are contacted by clients of the Hadoop Distributed File System (HDFS) to locate information within the file system and provide updates for data they have added, moved, manipulated, or deleted. The DataNode object is made up of a majority of servers contained in a Hadoop environment. Common Hadoop environments will have more than one DataNode object, and can number into the hundreds based on capacity and performance needs. The DataNode serves two functions, one it retains a segment of data in the HDFS and acts as a compute platform for running jobs, some of which will utilize the local data within the HDFS. The EdgeNode function accesses the point for external applications, tools, and users that need to utilize the Hadoop environment. While the EdgeNode rest between the Hadoop cluster and the rack network to provide access control, policy enforcement, logging, and gateway services to the Hadoop environment. A typical Hadoop environment will have a minimum of one EdgeNode and more based on performance needs [1].

Hadoop Distribute File System

The HDFS is a part of the Apache Hadoop overall project scheme. The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware.[2] Some basic functions are to provide distributed, highly fault-tolerant file system designed to run on low-cost commodity hardware. HDFS also offers high-throughput access to application data and is manageable by algorithms on large data sets. Hadoop is ideal for storing large amounts of data, like terabytes and petabytes, and uses HDFS as its storage system. HDFS allows the developer to connect nodes (commodity computers or non manufactured devices) contained within clusters over which data files are distributed. Experimenters can then access and store those data files as a single seamless file system. Access to data files is controlled in a streaming manner, meaning that applications or commands are executed directly using the MapReduce processing model. HDFS have characteristics common with other distributed file schemes. The difference is one noticeable aspect is HDFS's write-once-read-many model that relaxes concurrency control requirements, and simplifies data coherency. HDFS has a master/slave architecture. An HDFS cluster consists of a single NameNode, a master server that manages the file system namespace and regulates access to files by clients [3].

HDFS Architecture



Another unique feature of HDFS is the perspective of viewing locates processing logic near the data rather than moving the data to the application area. HDFS forces restrict data writing to one writer at a time. Bytes are always added to the end of a stream, while byte streams guarantee that information is stored in an ordered written manor.

HDFS a few of its most notable goals are as follows:

* Fault tolerance via fault detection while applying quick automated recovery protocols.
* Data is accessed through MapReduce streaming methods.
* It is encapsulated with in a robust and simple coherency model.
* Processing logic close to the data, rather than the data close to the processing logic.
* Portability across heterogeneous commodity hardware and operating systems.
* Scalability to reliably store and process large amounts of data.
* Distribute data and logic to process it in parallel on nodes where data is located.
* Reliability by automatically maintaining multiple copies of data and automatically redeploying processing logic in the event of failures.

Cluster Effect

HDFS provides interfaces for applications to migrate them closer to where the data is located [4]. It redeeming features are as follows:

**•** Data Coherency  
 – Write-once-read-many access model  
 – Client can only append to existing files

* Files are broken up into blocks– Each block replicated on multiple DataNodes
* Intelligent Client   
  – Client can find location of blocks  
  – Client accesses data directly from DataNode
* Types of Metadata   
  **–** List of files   
  – List of Blocks for each file   
  – List of DataNodes for each block   
  – File attributes, e.g creation time, replication factor
* A Transaction Log   
  **–** Records file creations, file deletions. Etc

The advantages of HDFS [3]

1. HDFS store large amount of information.
2. HDFS is simple and robust coherency model.
3. That is it should store data reliably.
4. HDFS is scalable and fast access to this information and it also possible to serve s large number of clients by simply adding more machines to the cluster.
5. HDFS should integrate well with Hadoop MapReduce, allowing data to be read and computed upon locally when possible.
6. HDFS provide streaming read performance.
7. Data will be written to the HDFS once and then read several times.
8. The overhead of cashing is helps the data should simply be re-read from HDFS source.
9. Fault tolerance by detecting faults and applying quick, automatic recovery.
10. Processing logic close to the data, rather than the data close to the processing logic Portability across heterogeneous commodity hardware and operating systems.
11. Economy by distributing data and processing across clusters of commodity personal computers.
12. Efficiency by distributing data and logic to process it in parallel on nodes where data is located.
13. Reliability by automatically maintaining multiple copies of data and automatically redeploying processing logic in the event of failures.

## The Disadvantage of HDFS [3] In distributed file system, it is limited in its power. The files in an NFS volume all reside on a single machine. This will create some problems

1. It does not gives any reliability guarantees if that machine goes down. Eg: – By replacing the files to other machine.
2. All the clients must go to this machine to retrieve their data. This can overload the server if a large number of clients must be handled.
3. Clients need to copy the data to their local machines before they can operate on it.

In distributed file system, it is limited in its power. The files in an NFS volume all reside on a single machine. This will create some problems

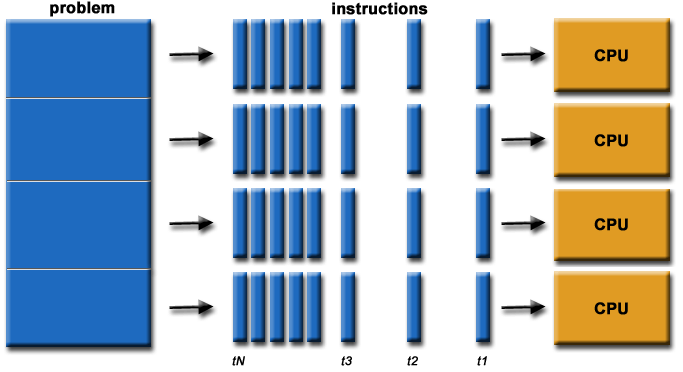
1. It does not gives any reliability guarantees if that machine goes down. Eg: – By replacing the files to other machine.
2. All the clients must go to this machine to retrieve their data. This can overload the server if a large number of clients must be handled.
3. Clients need to copy the data to their local machines before they can operate on it.

In addition, the user can use a web browser to browse HDFS files. One can access HDFS in different ways. HDFS provides a native Java™ application programming interface (API) and a native C-language wrapper for the Java API. HDFS provides high throughput access to application data and applications with large data sets.

Parallel Processing

Aparallel computer schema is a system of processing functions which work together to solve large problems as quickly as possible. This is achieved by using physical part that all commodity machines have in common. Those metrics are cpu, memory, disk, networks and the Internet [5]. At the heart of this structure lays the program driving the outer functionalities. Those elements consist of the following attributes:

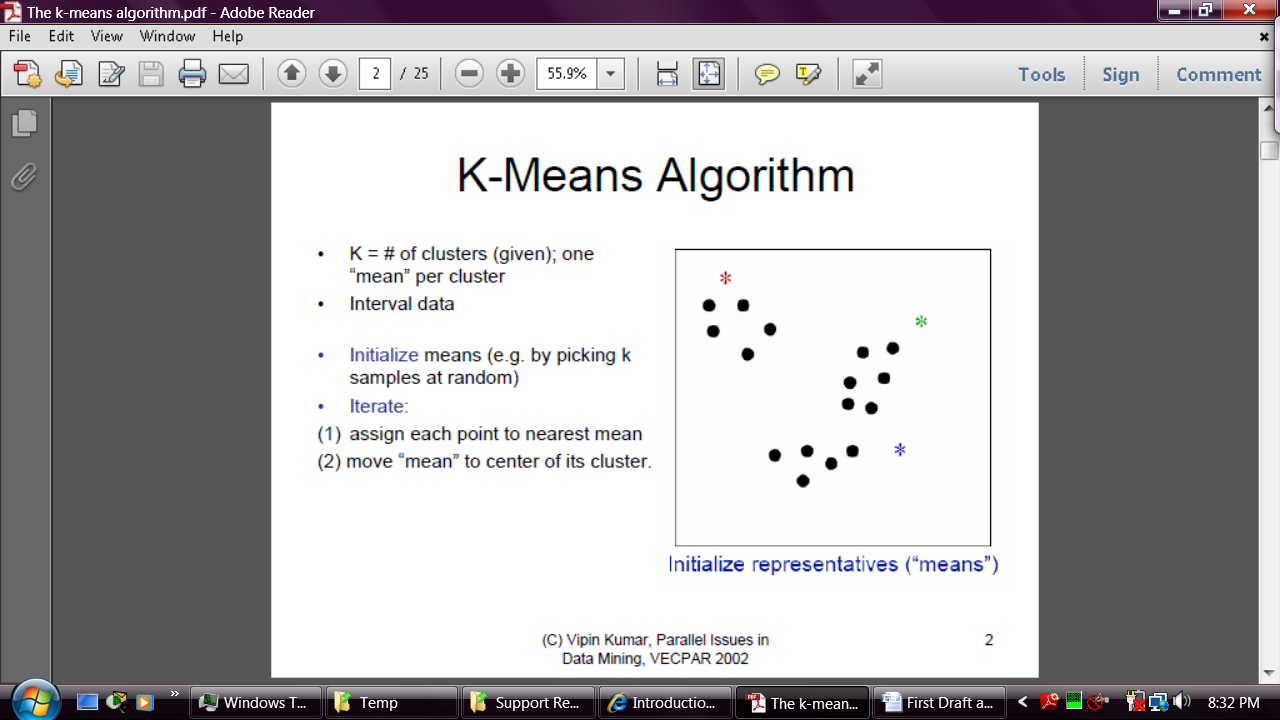
* + To be run on a single computer having a single Central Processing Unit (CPU);
  + A problem is broken into a discrete series of instructions.
  + Instructions are executed one after another.
  + Only one instruction may execute at any moment in time.



Two of the common fields that make use of parallel computing are science and the branch of engineering. In the pass parallel computing has been thought of as a high end facilitation of computing. It is a good method for modeling complex real world problems. Some important characteristic of parallel computing is that it provides concurrency which meansit can perform only one task at a time. Synchronization coordinates parallel tasks in real time. This function is often implemented by establishing a synchronization point within an application where a task may not proceed further until another task(s) reaches the same or logically equivalent point. Still many computing resources can reduce the overall processing time over a one processor. Synchronization usually causes a parallel application's clock to execution time faster. A few advantages are global address space provided for user-friendly programming memory and data sharing between tasks is faster and uniform because the proximity of memory to CPUs. A primary disadvantages is the lack of scalability between memory and CPUs. By using detailed documents to cite from, we were able to derive at deeper understanding of the intricacies of those methodologies implemented and allow us insights to examine how these techniques work.

k – mean Algorithms

The purpose of k – mean is to compute a single cluster to its interval data. Initialize the means by selecting k samples through random processing [6]. While looping give every point in proximity to closes mean. And position the “mean” to the center of its cluster [7].



The downside of k-means it has trouble with cluster of different sizes, densities, non-globular shapes, and empty cluster. These aspects need to be resolved by fixing them within the k-mean methodology or find another approach that can handle k-mean weakness. Perhaps the Model based k-means “means” are probabilistic models (unified framework Zhong & Ghosh, JMLR 03) is a step in a better direction.

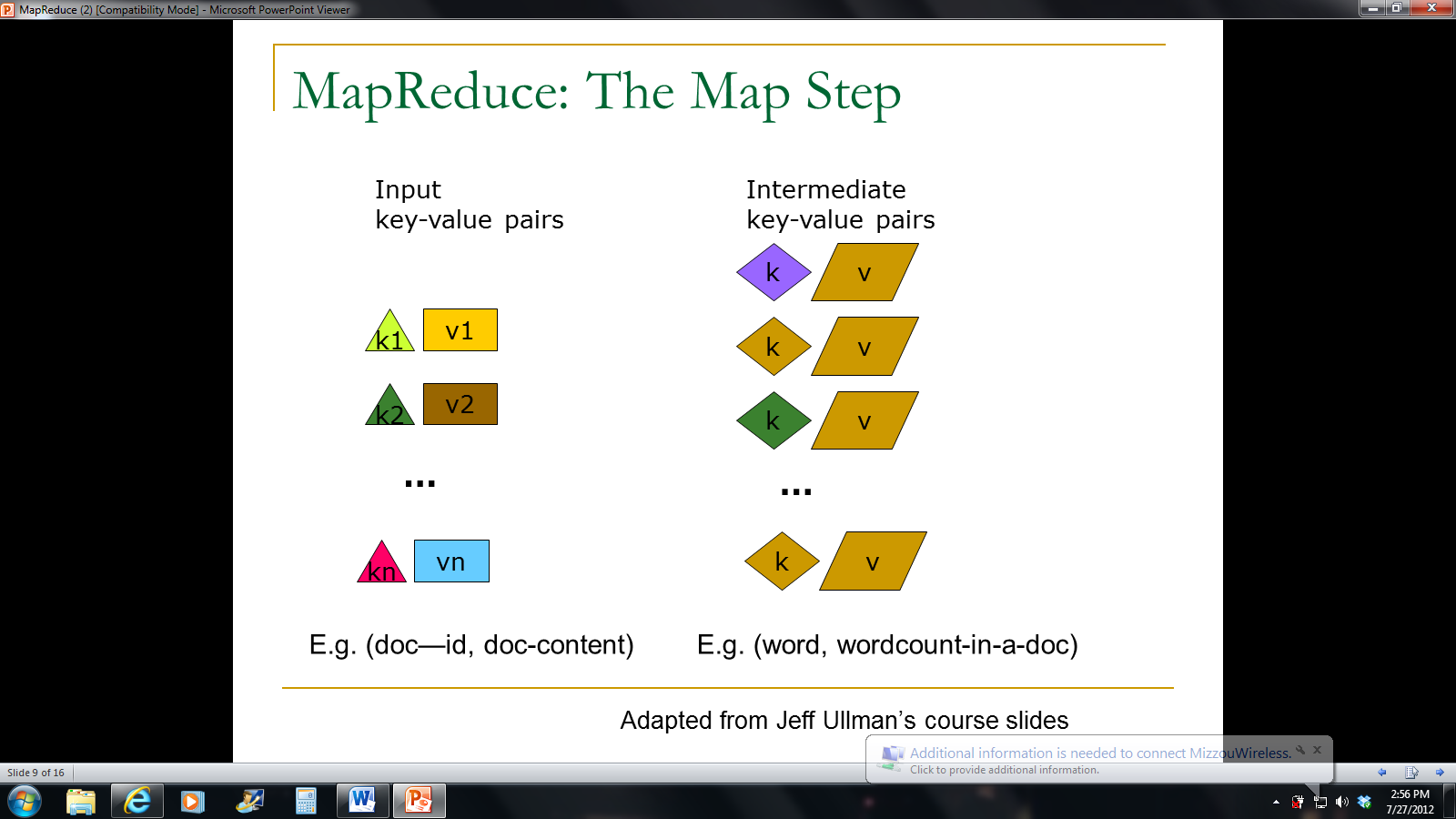
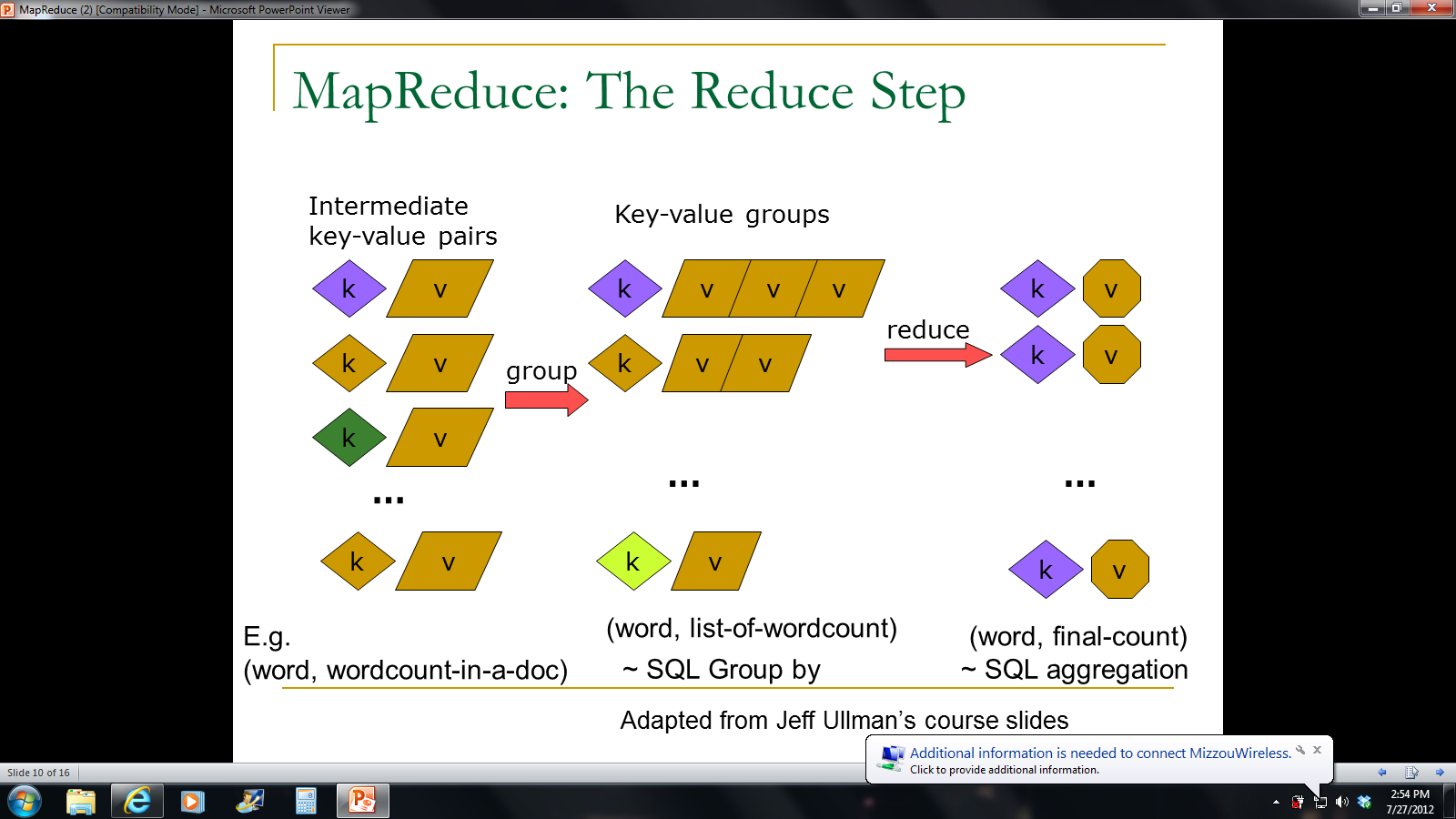
Map and Reduce

Google’s MapReduce programming model [8] serves as an example for processing large data sets in a enormous parallel fashion. They provided the first rigorous description of the methodology including its advancement as Google’s domain-specific language Sawzall. The model is rudimentary and efficient in supporting parallelism. The MapReduce programming model is clearly summarized in the following quote [8]: “The computation takes a set of input key/value pairs, and produces a set of output key/value pairs. The user of the MapReduce library expresses the computation as two functions: map and reduce. Map, written by the user, takes an input pair and produces a set of intermediate key/value pairs. The MapReduce library groups together all intermediate values associated with the same intermediate key I and passes them to the reduce function. The reduce function, also written by the user, accepts an intermediate key I and a set of values for that key. It merges together these values to form a possibly smaller set of values. Typically just zero or one output value is produced per reduce invocation. The intermediate values are supplied to the user’s reduce function via an iterator. This allows us to handle lists of values that are too large to fit in memory [8].” Google’s MapReduce framework engages parallel applications, distributed, data-intensive, by deconstructing large job into pieces via the map and reduce tasking processes. While the massive dataset is downsized into smaller partitions, such that each task processes a different partition in parallel. There are performances problems in the distributed MapReduce system that can be hard to pin point as well as localize to specific node or a set of nodes. On the other hand, the structure of large number of nodes performing similar tasks allows the opportunity for observing the system from multiple viewpoints. Data mining algorithms are programs that work on specific attributes of the primitive data type being investigated.

The map and reduce functions in Hadoop MapReduce consist of the following format [8]:

map: (K1, V1) → list(K2, V2)

reduce: (K2, list(V2)) → list(K3, V3)

Word Count

The word count application illustrated how power this layer of processing is even on a simple string text consisting of only on a few words. The run job activates core functionalities of the framework that consist of fifteen counters. The file system counters is made up four buffer types. The file read is all the data input of a given session and it keep tracks that information on the native file structure. Where the Hadoop Distribute File System (HDFS) accumulates additional read bytes in its counter. The file bytes written express the decomposition of unnecessary data request. Whereas the HDFS counter has an even small count of bytes then that of the naive output file. The map – reduce framework counter are as follows: Reduce input groups, combine output records, Map input records, Reduce shuffle bytes, Reduce output records, spilled records, map output bytes, Map input bytes, combine input records, Map output records, and Reduce input records[9]. The order is important because that is the way the input data is handed off from the beginning to the end. Map input records is the amount of input records consumed by all the maps in the job. It is accumulates each record that is read from the “Record Reader” and then passed to the map’s map() method via the framework [10]. The reduce shuffle function exposes many tuning parameters for memory management, which can assist in pointing out weak bit performance. Map input bytes of the uncompressed input is processed by the maps in the job. In the word count experiment showed that even though Java and other programming languages are implemented there are to carry out specific request attributes. Whereas the map – reduce retains control over how input is manipulated into parts.



Performance

There are performances problems in the distributed MapReduce system that can be hard to pin point as well as localize to specific node or a set of nodes. On the other hand, the structure of large number of nodes performing similar tasks allows the opportunity for observing the system from multiple viewpoints. Data mining algorithms are programs that work on specific attributes of the primitive data type being investigated. Performance data mining in a standalone-node simulation is useful as a learning tool. It presents an opportunity to view the functionality of the Hadoop environment characteristics. Performance improvement or tweaking the system is key to adjusting the structure to fit the data requirements in which the prototype system is being developed. A way of reducing communication overhead in a single cluster is to understanding payloads of data streaming being manipulated for patterns for meaningful information decisions. The possible wealth of lesser structured data resources such as weblogs, social media, email, and sensors, metadata can provide rich pools of facts. Those insights can be useful components for intelligence mining for a variety of knowledge base applications.

Verification

All the tasks are simulated within a single thread

Results are verified correctly

Future Work



Run the task with multi-threads or multi-servers to compare performance

Apply Map-Reduce to more complicated data-mining algorithms

Conclusion

The Hadoop is a very complex system of tasks that work together to achieve a fundamental purpose which is to handle expanding volumes of streaming data structures. As various contributors keep discovering more effective methodologies and applications that address real time problems scalabilities will be maintain

References

[1] Dell **Hadoop White Paper Series**  
 [http://i.dell.com/sites/content/business/solutions/whitepapers/en/Documents/hadoop- introduction.pdf](http://i.dell.com/sites/content/business/solutions/whitepapers/en/Documents/hadoop-%20introduction.pdf)

[2] Author Dhruba Borthakur, Hadoop Distributed File System Version Control System  
 Copyright © 2007 [The Apache Software Foundation.](http://www.apache.org/licenses/)  
 Web site: <http://hadoop.apache.org/hdfs/version_control.html>  
[3] Author Dhruba Borthakur, HDFS ArchitectureWeb site: <http://hadoop.apache.org/common/docs/r0.20.0/hdfs_design.html>

## [4] Hadoop Distributed File System (HDFS) Web site <http://www.j2eebrain.com/java-J2ee-hadoop-distributed-file-system-hdfs.html>

[5] Author: Alaise Barney, Lawrence Livermore National Laboratory  
 Web site <https://computing.llnl.gov/tutorals/parallel.com/>  
[6] Inderjit S. Dhillon, Yuqiang Guan, Brian Kulis: Kernel kmeans: spectral clustering and  
 normalized cuts. KDD 2004: 551-556

[7] Databases", Proc. 4 th International Conf. on Knowledge Discovery and Data Mining  
 (KDD-98). AAAI Press, Aug. 1998

[8] J. Dean, S. Ghemawat, MapReduce: Simplified Data Processing on Large Clusters, in:  
 OSDI’04, 6th Symposium on Operating Systems Design and Implementation, Sponsored by  
 USENIX, in cooperation with ACM SIGOPS, 2004, pp. 137–150.

[9] Author Dhruba Borthakur, Hadoop Distributed File System Version Control System  
 Copyright © 2007 [The Apache Software Foundation.](http://www.apache.org/licenses/)  
 Web site: <http://hadoop.apache.org/hdfs/version_control.html>  
  
[10] Author Dhruba Borthakur, HDFS ArchitectureWeb site: <http://hadoop.apache.org/common/docs/r0.20.0/hdfs_design.html>

## [11] Hadoop Distributed File System (HDFS) Web site <http://www.j2eebrain.com/java-J2ee-hadoop-distributed-file-system-hdfs.html>