CSCI446/946 Big Data Analytics

Week 12 Advanced Analytical Theory and Methods: MapReduce and Hadoop

School of Computing and Information Technology

University of Wollongong Australia

Advanced Analytical Theory and Methods: MapReduce and Hadoop

- Overview
- Analytics for Unstructured Data
  - Use Cases, MapReduce, Hadoop
- The Hadoop Ecosystem
  - Pig, Hive, Hbase, Mahout
- NoSQL
- Summary

THE APACHE® SOFTWARE FOUNDATION

hadoop

Overview

- The Apache Hadoop software library
  - A framework allows for the distributed processing of large datasets across clusters of computers using simple programming models
- Hadoop
  - How it stores data in a distributed system
  - How it implements a simple programming paradigm – MapReduce
  - How its tools help MapReduce programming

All the figures, tables and codes are from the book "Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data" unless indicated otherwise.

Analytics for Unstructured Data

- Unstructured data (text, image, video, etc.)
  - Data that has no inherent, consistent structure
- MapReduce paradigm offers the means to
  - break a large task into smaller tasks
  - run tasks in parallel
  - consolidate the outputs of the individual tasks into the final output
- Apache Hadoop implements MapReduce

Analytics for Unstructured Data

- Use Cases
  - IBM Waston won Jeopardy champions
  - Linkedin utilizes Hadoop for multiple tasks
    - Process daily production database transaction logs
    - Examine the users' activities (views and clicks)
    - Feed the extracted data back to the production systems
  - Yahoo!'s Hadoop applications involve
    - Search index creation, web page content optimization, web ad placement optimization, spam filters, …

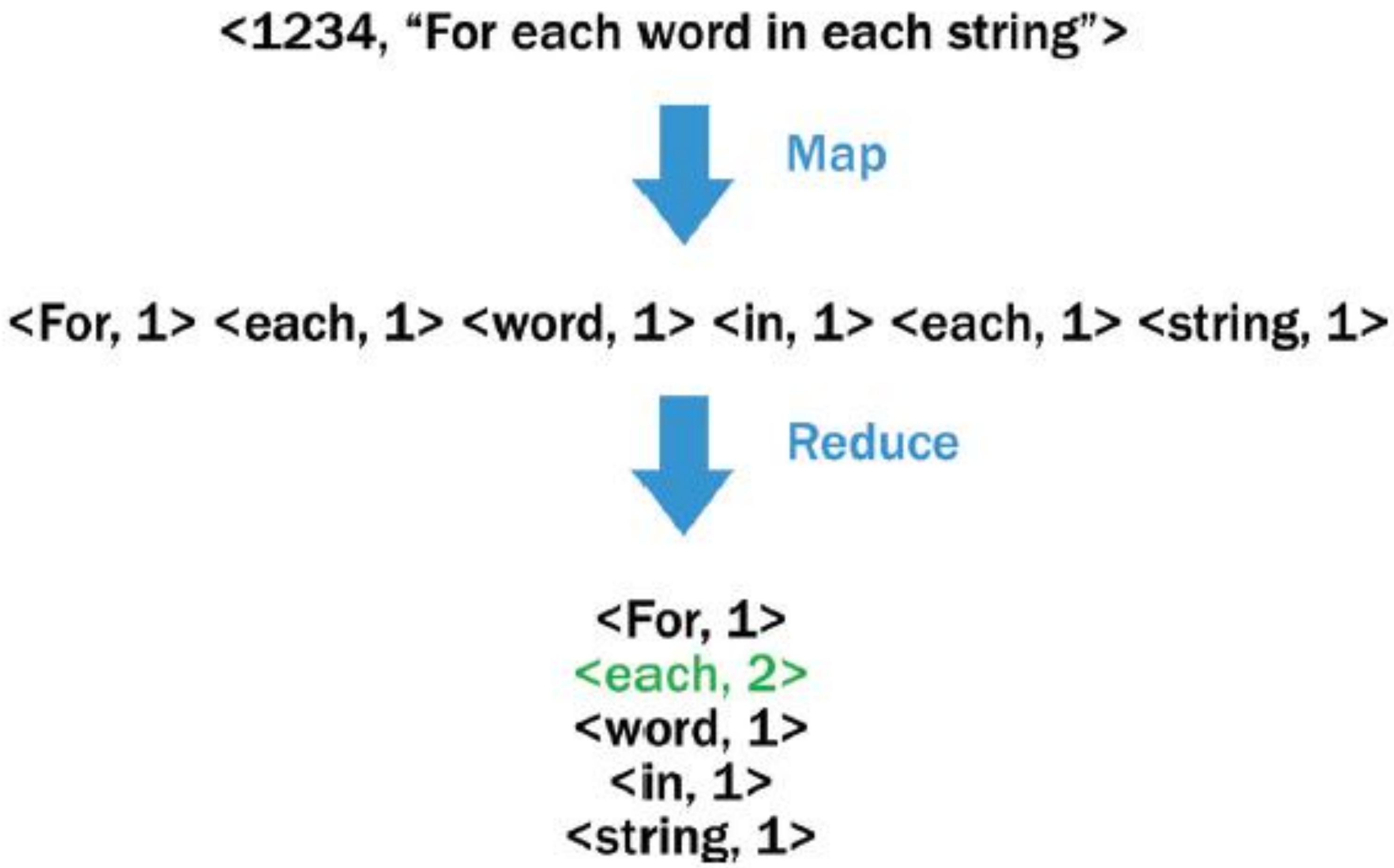
Analytics for Unstructured Data

- MapReduce consists of two basic parts
  - A map step and a reduce step
- Map step
  - Applies an operation to a piece of data
  - Provides some intermediate output
- Reduce step
  - Consolidates the above intermediate outputs
  - Provides the final output

Analytics for Unstructured Data

- Each step uses key/value pairs as input and output, denoted as `<key, value>`
- The pairs can take complex forms
  - For example, the key is a filename, and the value is the entire content of the file
- A simple illustration of MapReduce
  - The task is to count the number of times each word appears in a collection of documents

Analytics for Unstructured Data



<1234, “For each word in each string”>

Map

<For, 1> <each, 1> <word, 1> <in, 1> <each, 1> <string, 1>

Reduce

<For, 1>
<each, 2>
<word, 1>
<in, 1>
<string, 1>

- The map step parse the string into single words and emits a set of key/value pairs in the form <word, 1>;
- For each unique key, the reduce step sums the 1 values and outputs the <word, count> key/value pairs;

Analytics for Unstructured Data

- MapReduce
  - Has the advantage of being able to distribute the workload over a cluster of computers and run the tasks in parallel
  - The documents, or even their pieces, could be processed simultaneously during the map step
  - The processing of one portion of the input can be carried out independently of the others

Analytics for Unstructured Data

- MapReduce
  - A simple paradigm to understand, but not easy to implement, esp. in a distributed system
- Executing a MapReduce job requires
  - Management and coordination of activities of
    - Scheduling jobs, monitoring jobs
    - Spreading data, conducting the map step across
    - Collecting numerous intermediate outputs
    - Making the final output

Analytics for Unstructured Data

- Apache Hadoop
  - Handles these activities and more, and make most of them transparent to the users
  - An open source project managed and licenced by the Apache Software Foundation
- The development of MapReduce proceeds easily because
  - The MapReduce paradigm has already been implemented in Apache Hadoop

Analytics for Unstructured Data

- Apache Hadoop
  - Hadoop Distributed File System (HDFS)
    - How data is stored in a Hadoop environment
  - Structuring a MapReduce job in Hadoop
    - How a MapReduce job is run
  - Additional considerations
  - Developing a Hadoop MapReduce program

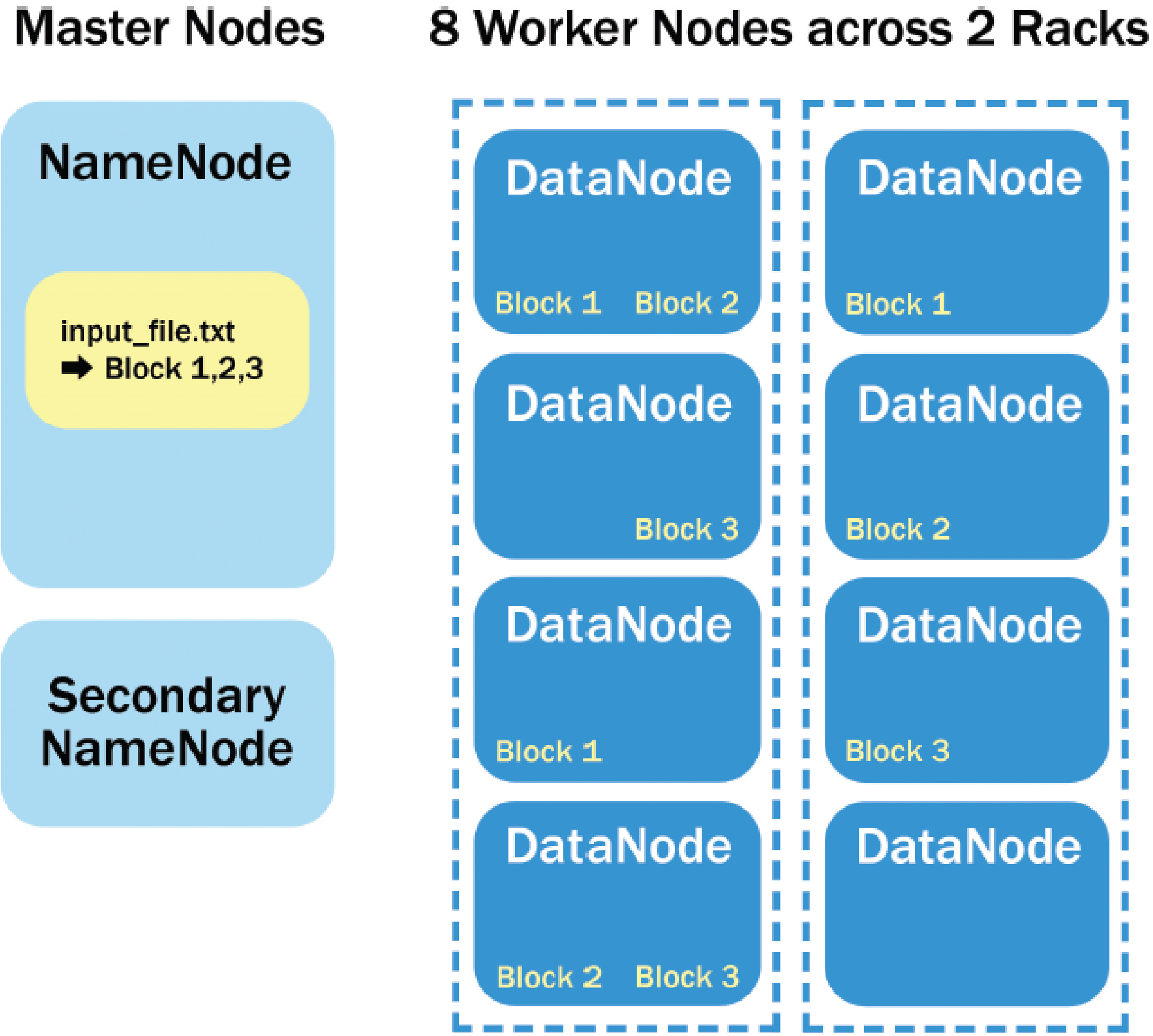
Analytics for Unstructured Data

- Hadoop Distributed File System (HDFS)
  - Provides the capability to distribute data across a cluster to use parallel processing of MapReduce
  - HDFS is not an alternative to common file systems, but depends on each disk drive's file system
  - HDFS breaks a file into blocks and stores the blocks across the clusters
    - The blocks of a file are stored on different machines
    - By default, creates three copies of each block across

Analytics for Unstructured Data

- HDFS uses three Java daemons (background processes)
  - NameNode determines and tracks where the blocks of a data file are stored. It runs on a single machine and resides in its memory
  - DataNode manages data stored on each machine
  - Secondary NameNode provides the capability to perform some of the NameNode tasks to reduce the load of NameNode
    - But it is not a backup or redundant NameNode

Analytics for Unstructured Data



Analytics for Unstructured Data

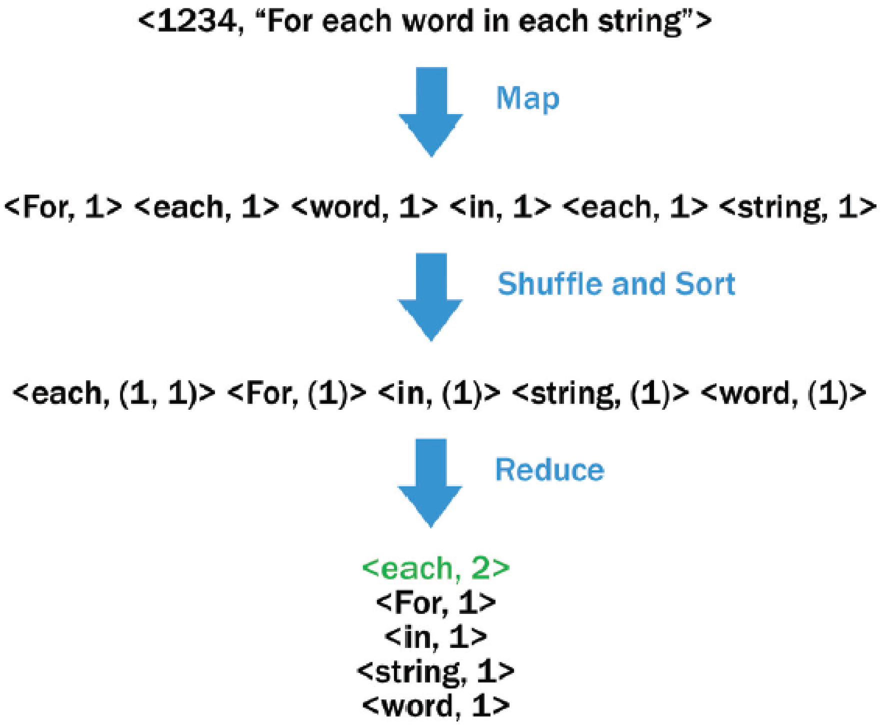
- A MapReduce program has three classes
  - The driver class provides details such as
    - Input and output file locations
    - Names of the mapper and reducer Java classes
    - Various job configuration options
  - The mapper class provides the logic to be processed on each data block corresponding to the specified input files
  - The reducer class provides the logic to process the values for each key to emit a key/value pair

Analytics for Unstructured Data

- Additional consideration in structuring a job
  - Combiner (“local reducer”)
    - It is a useful option to apply between the map task and the shuffle and sort
    - It applies the same logic used in the reducer on the output of each map task
  - Partitioner
    - It determines the reducers that receive keys and the corresponding list of values

Analytics for Unstructured Data

- Additional consideration in structuring a job



<1234, “For each word in each string”>

Map

<For, 1> <each, 1> <word, 1> <in, 1> <each, 1> <string, 1>

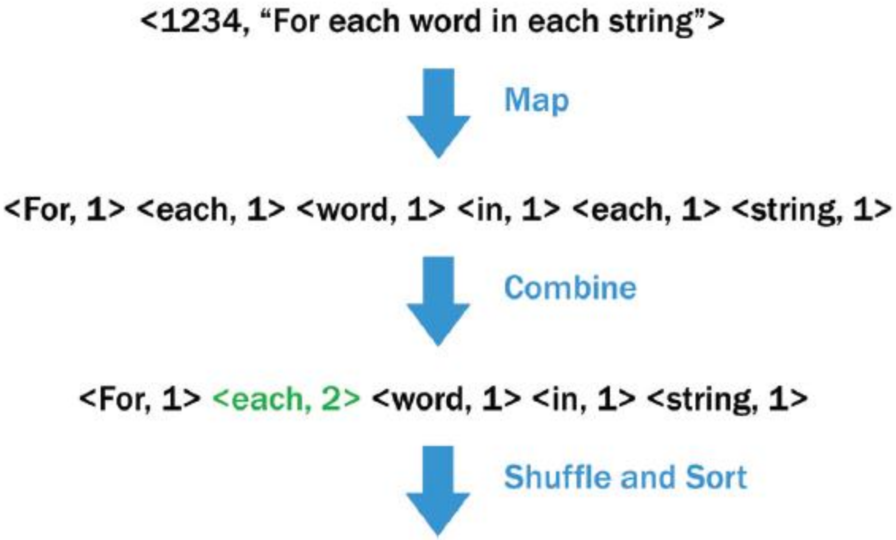
Shuffle and Sort

<each, (1, 1)> <For, (1)> <in, (1)> <string, (1)> <word, (1)>

Reduce

<each, 2>
<For, 1>
<in, 1>
<string, 1>
<word, 1>

Shuffle and sort



<1234, “For each word in each string”>

Map

<For, 1> <each, 1> <word, 1> <in, 1> <each, 1> <string, 1>

Combine

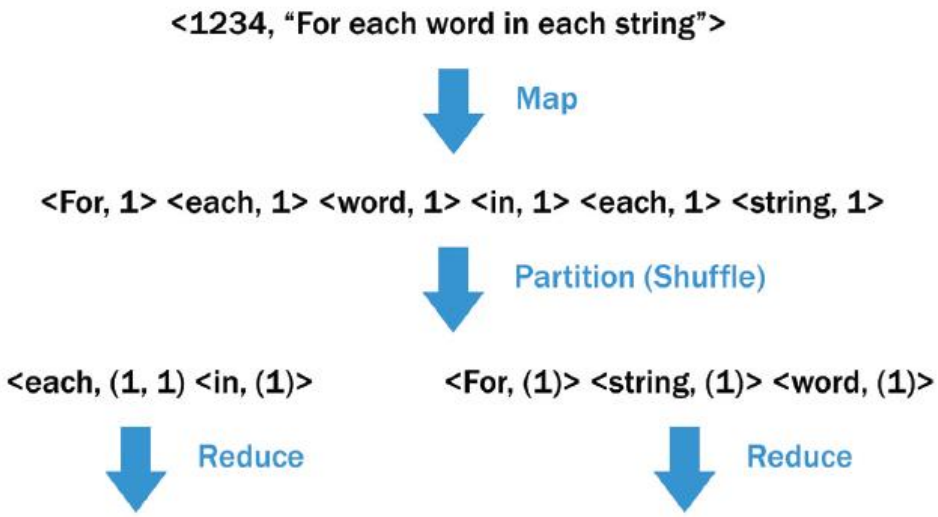
<For, 1> <each, 2> <word, 1> <in, 1> <string, 1>

Shuffle and Sort

Combiner

Analytics for Unstructured Data

- Additional consideration in structuring a job



<1234, "For each word in each string">

Map

<For, 1> <each, 1> <word, 1> <in, 1> <each, 1> <string, 1>

Partition (Shuffle)

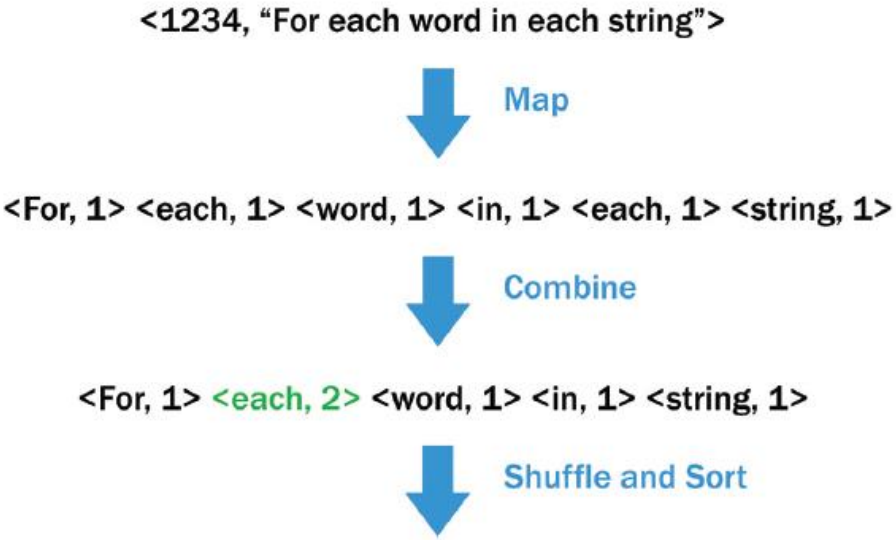
<each, (1, 1) <in, (1)>

<For, (1)> <string, (1)> <word, (1)>

Reduce

Reduce

Paritioner



<1234, "For each word in each string">

Map

<For, 1> <each, 1> <word, 1> <in, 1> <each, 1> <string, 1>

Combine

<For, 1> <each, 2> <word, 1> <in, 1> <string, 1>

Shuffle and Sort

Combiner

Analytics for Unstructured Data

- Developing and executing a program (via Java)
  - A typical MapReduce program consists of three Java files, one each for the driver code, map code, and reduce code
  - Java files can be written for the combiner and partitioner
  - Jave code is compiled as a Java Archive (JAR) file
  - The JAR file is executed against the specified HDFS input files

Analytics for Unstructured Data

- Three key challenges to a new developer
  - Defining the logic of the code to use the MapReduce paradigm
  - Learning the Apache Hadoop Java classes, methods, and interfaces
  - Implementing the driver, map, and reduce functionality in Java
- Some prior experience with Java will help

Analytics for Unstructured Data

- Three key challenges to a new developer
  - Defining the logic of the code to use the MapReduce paradigm
  - Learning the Apache Hadoop Java classes, methods, and interfaces
  - Implementing the driver, map, and reduce functionality in Java
- Some prior experience with Java will help

Analytics for Unstructured Data

- For non-Java developers
  - Hadoop Streaming API
    - `Hadoop-streaming.jar` file: accepts the HDFS paths for the input/output files and the paths for the files that implement the map and reduce functionality
    - There are some important considerations/limitations
  - Hadoop pipes
    - A mechanism uses compiled C++ code for the map and reduce functionality
    - An advantage of using C++ is the numerical libraries

Analytics for Unstructured Data

- To directly work with data in HDFS
  - Use the C API (libhdfs) or the Java API provided with Apache Hadoop
  - These APIs allow reads and writes to HDFS data files outside the typical MapReduce paradigm
  - Such approach may be helpful when
    - Debug a MapReduce job by examining the input data
    - Transform the HDFS prior to running a MapReduce job

Analytics for Unstructured Data

- MapReduce 2.0
  - Also called Yet Another Resource Negotiator (YARN)
- The two functionalities are separated
  - MapReduce functionality
  - Those to manage job running and associated responsibilities in a distributed environment
  - YARN makes it possible for utilizing the paradigms other than MapReduce in Hadoop

The Hadoop Ecosystem

- Hadoop’s proprietary and open source tools
  - Make Apache Hadoop easier to use
  - Provide additional functionality and features
- Hadoop-related Apache projects
  - Pig: A high-level data-flow programming language
  - Hive: Provide SQL-like access
  - Mahout: Provide analytical tools
  - Hbase: Provides real-time reads and writes

The Hadoop Ecosystem

- Apache Pig
  - Consists of a data flow language, Pig Latin, and an environment to execute the Pig code
  - Benefit is simplifying the tasks of developing and executing a MapReduce job
  - When Pig commands are executed, the running of a job at background is transparent to the user
  - Three main characteristics
    - Ease of programming, behind-the-scenes code optimization, and extensibility of capabilities

The Hadoop Ecosystem

- Apache Hive
  - Similar to Pig, Hive enables users to process data without explicitly writing MapReduce code
  - One key difference to Pig
    - Hive language (HiveQL) resembles Structured Query Language (SQL) rather than a script language
  - Hive may be a good tool to use if
    - Data easily fits into a table structure
    - Data is already in HDFS
    - Developers are comfortable with SQL

The Hadoop Ecosystem

- Apache HBase
  - Pig and Hive are intended for batch applications
  - Differently, Hbase provides real-time read and write access to large-scale datasets
  - Is built upon HDFS
  - Share the workload over a large number of nodes in a distributed cluster
  - Uses a key/value structure to store the contents of an HBase table

The Hadoop Ecosystem

- Apache Mahout
  - Tools such as R may suffer from performance issues with the large datasets in stored in Hadoop
  - Supports the application of analytical techniques within the Hadoop environment
  - Provides Java libraries to apply analytical techniques in a scalable manner to Big Data
  - Implemented algorithms
    - Classification, clustering, collaborative filtering

NoSQL

- NoSQL (Not only Structured Language)
  - A term used to describe those data stores that are applied to unstructured data
  - As the size of data grows, the solution can scale by adding machines to the distributed system
  - Four major categories of NoSQL tools
    - Key/value store, Document store, Column family store, Graph databases
  - The choice of data store is take-dependent

Summary

- The MapReduce paradigm and its implementation in Apache Hadoop
- The Hadoop Distributed File System (HDFS)
- Developing a Hadoop MapReduce program
- Apache projects in the Hadoop Ecosystem

Q&A

Images Courtesy of Google Image