USN | 1 | C | R | 2 | 0 | M | C | 0 | 7 | 3 |

20MCA352

Third Semester MCA Degree Examination, Feb./Mar. 2022
Big Data Analytics

Time: 3 hrs. Max. Marks: 100

Note: Answer any FIVE full questions, choosing ONE full question from each module.

Module-1

1 a. Define Big Data. **(02 Marks)**
b. Explain different characteristics of Big data Analytics. **(10 Marks)**
c. Write a note on missing values. **(08 Marks)**

OR

2 a. Discuss different types of data sources of big data analytics. **(10 Marks)**
b. Discuss the applications of big data analytics. **(10 Marks)**

Module-2

3 a. Briefly explain mobile intelligence and big data. **(10 Marks)**
b. Write a note on cloud and big data. **(10 Marks)**

OR

4 a. Explain working together of HDFS and map reduce. **(10 Marks)**
b. Define crowd sourcing. Explain different types of crowd sourcing. **(10 Marks)**

Module-3

5 a. Compare between Big data, RDBMS and grid computing. **(10 Marks)**
b. Write a note on: i) History of Hadoop ii) Volunteer computing **(10 Marks)**

OR

6 a. Illustrate the disabilities of implementing storage and analysis support for big data. **(10 Marks)**
b. Explain different Hadoop versions and releases. **(10 Marks)**

Module-4

7 a. Briefly explain the anatomy of file write. **(10 Marks)**
b. Explain different HDFS concepts in detail. **(10 Marks)**

OR

8 a. Discuss different file system openmions. **(10 Marks)**
b. Define file system interface. Explain different file system interfaces. **(10 Marks)**

Module-5

9 a. What is MapReduce? Sketch the neat diagram and explain the logical data flow in MapReduce. **(10 Marks)**
b. Write a short note on: i) MapReduce UI ii) Hadoop streaming **(10 Marks)**

OR

10 a. Write a java MapReduce code to find maximum temperature from the weather data set. **(10 Marks)**
b. Write a note on: i) Hadoop logs ii) Remote Debugging. **(10 Marks)**

* * * * *

Big Data Analytics

1. a. Define Big Data.

Big data refers to data that is so large, fast or complex that it's difficult or impossible to process using traditional methods. Big data also encompasses a wide variety of data types, including the following: structured data, such as transactions and financial records; unstructured data, such as text, documents and multimedia files; and. semi structured data, such as web server logs and streaming data from sensors.

b. Explain different characteristics of Big Data Analytics

Characteristics of Big Data:
1. Volume
2. Velocity
3. Variety
4. Value

Volume: The main characteristic that makes data "big" is the sheer volume. It makes no sense to focus on minimum storage units because the total amount of information is growing exponentially every year.
Variety: is one the most interesting developments in technology as more and more
Information is digitized. Traditional data types (structured data) include things on a bank statement like date, amount, and time.
Velocity is the frequency of incoming data that needs to be processed.
Value: Analysis add value to your business is measured.

c. Write a note on missing values.

In statistics, missing data, or missing values, occur when no data value is stored for the variable in an observation. Missing data are a common occurrence and can have a significant effect on the conclusions that can be drawn from the data.

2. a. Discuss different types of data source of big data analytics.

Types of Data Sources. Data can originate from a variety of different sources.
They are as follows:
Transactional data
Unstructured data
Qualitative/Export based data
Data poolers
Publicly available data

Transactional Data: Transactions are the first important source of data. Transactional data consist of structured, low level, detailed information capturing the key characteristics of a customer transaction (e.g., purchase, claim, cash transfer, credit card payment). This type of data is usually stored in massive online transaction processing (OLTP) relational databases. It can also be summarized over longer time horizons by aggregating it into averages, absolute/relative trends, maximum/minimum values, and so on.

Unstructured data: Embedded in text documents (e.g., emails, web pages, claim forms) or multimedia content can also be interesting to analyze. However, these sources typically require extensive pre-processing before they can be successfully included in an analytical exercise.

Qualitative/Expert based data: Another important source of data is qualitative, expert based data. An expert is a person with a substantial amount of subject matter expertise within a particular setting (e.g., credit portfolio manager, brand manager). The expertise stems from both common sense and business experience, and it is important to elicit expertise as much as possible before the analytics is run. This will steer the modelling in the right direction and allow you to interpret the analytical results from the right perspective. A popular example of applying expert based validation is checking the univariate signs of a regression model. For example, one would expect a priori that higher debt has an adverse.
Data poolers: Nowadays, data poolers are becoming more and more important in the industry. Popular examples are Dun & Bradstreet, Bureau Van Dijck, and Thomson Reuters. The core business of these companies is to gather

data in a particular setting (e.g., credit risk, marketing), build models with it, and sell the output of these models (e.g., scores), possibly together with the underlying raw data, to interested customers. A popular example of this in the United States is the FICO score, which is a credit score ranging between 300 and 850 that is provided by the three most important credit bureaus: Experian, Equifax, and TransUnion. Many financial institutions use these FICO scores either as their final internal model or as a benchmark against an internally developed credit scorecard.

Publicly available data: Finally, plenty of publicly available data can be included in the analytical exercise. A first important example is macroeconomic data about gross domestic product (GDP), inflation, unemployment, and so on. By including this type of data in an analytical model, it will become possible to see how the model varies with the state of the economy. This is especially relevant in a credit risk setting, where typically all models need to be thoroughly stress tested. In addition, social media data from Facebook, Twitter, and others can be an important source of information. However, one needs to be careful here and make sure that all data gathering respects both local and international privacy regulations.

b. Discuss the applications of big data analytics.

Analytics is everywhere and strongly embedded in our daily lives.

The relevance, importance and impact of analytics are now bigger than ever before and, given that more and more data are being collected and that there is strategic value in knowing what is hidden in data, analytics will continue to grow.

Physical mail box: a catalogue sent to us through mail most probably as a result of a response modeling analytical exercise that indicated, given my characteristics and previous purchase behaviour, we are likely to buy one or more products from it.

Behavioral Scoring Model: Checking account balance of the customer from the past 12 months and credit payments during that period, together with other kinds of information available to the bank, to predict whether a customer will default on the loan during the next year.

Social Media: As we logged on to my Facebook page, the social ads appearing there were based on analyzing all information (posts, pictures, my friends and their behaviour, etc.) available to Facebook. Twitter posts will be analyzed (possibly in real time) by social media analytics to understand both the subject tweets and the sentiment of them.

Marketing	Risk Management	Government	Web	Logistics	Other
Response modeling	Credit risk modeling	Tax avoidance	Web analytics	Demand forecasting	Text analytics
Net lift modeling	Market risk modeling	Social security fraud	Social media analytics	Supply chain analytics	Business process analytics
Retention modeling	Operational risk modeling	Money laundering	Multivariate testing		
Market basket analysis	Fraud detection	Terrorism detection			
Recommender systems					
Customer segmentation					

3. a. **Briefly explain mobile intelligence and big data**

Mobile Business Intelligence and Big Data Analytics on mobile devices is what some refer to as putting BI in your pocket. Mobile drives straight to the heart of simplicity and ease of use that has been a major barrier to BI adoption since day one.

Kerzner explains his view on this topic:
They have been working on Mobile BI for a while but the iPad was the inflection point where I think it started to become mainstream.
We have seen customers over the past decade who focused on the mobile space generally and mobile applications in particular.
One client in particular told me that he felt like he was pushing a boulder up a hill until he introduced mobility to enhance productivity.
Once the new smart phones and tablets arrived, his phone was ringing off the hook and he was trying to figure out which project to say yes to, because he couldn't say yes to everyone who suddenly wanted mobile analytics in the enterprise.

Ease of Mobile Application Deployment

Three elements that have impacted the viability of mobile BI:

1. Location—the GPS component and location . . . know where you are in time as well as the movement.
2. It's not just about pushing data; you can transact with your smart phone based on information you get.
3. Multimedia functionality allows the visualization pieces to really come into play.

Three challenges with mobile BI include:

1. Managing standards for rolling out these devices.
2. Managing security (always a big challenge).
3. Managing "bring your own device," where you have devices both owned by the company and devices owned by the individual, both contributing to productivity.

b. Write a note on cloud and big data.

The Cloud and Big Data

It is important to remember that for all kinds of reasons—technical, political, social, regulatory, and cultural—cloud computing has not been a successful business model that has been widely adopted for enterprises to store their Big Data assets. However, there are many who believe that some obvious industry verticals will soon realize that there is a huge ROI opportunity if they do embrace the cloud.
As per by Avinash Koushik, Google's digital marketing evangelist.

"There will be Big Data platforms that companies will build, especially for the core operational system of the world. Where we continue to have an explosive amount of data come in and because the data is so proprietary that building out an infrastructure in-house seems logical. I actually think it's going to go the cloud; it's just a matter of time! It's not value add enough add enough to collect, process and store data".

Market economics are demanding that capital-intensive infrastructure costs disappear and business challenges are forcing clients to consider newer models. At the crossroads of high capital costs and rapidly changing business needs is a sea change that is driving the need for a new, compelling value proposition that is being manifested in a cloud-deployment model.

With a cloud model, you pay on a subscription basis with no upfront capi- tal expense. You don't incur the typical 30 percent maintenance fees—and all the updates on the platform are automatically available. The traditional cost of value chains is being completely disintermediated by platforms—massively scalable platforms where the marginal cost to deliver an incremental product or service is zero.

The ability to build massively scalable platforms—platforms where you have the option to keep adding new products and services for zero additional cost—is giving rise to business models that weren't possible before. Mehta calls it "the next industrial revolution, where the raw material is data and data factories replace manufacturing factories." He pointed out a few guiding principles that his firm stands by:

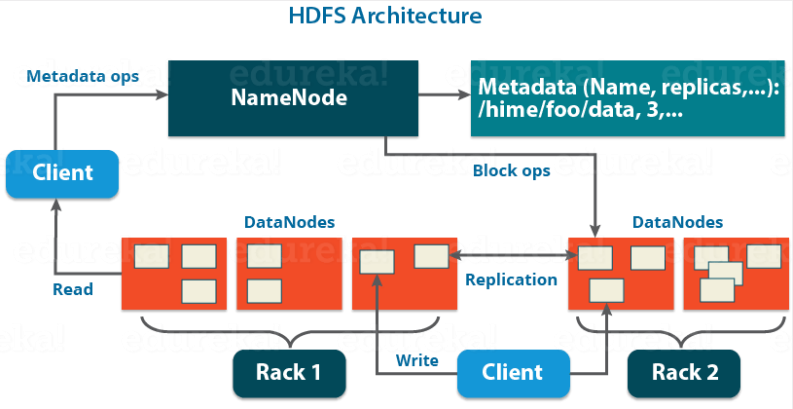
1. Stop saying “cloud.” It’s not about the fact that it is virtual, but the true value lies in delivering software, data, and/or analytics in an “as a service” model. Whether that is in a private hosted model or a publicly shared one does not matter. The delivery, pricing, and consumption model matters.
2. Acknowledge the business issues. There is no point to make light of matters around information privacy, security, access, and delivery. These issues are real, more often than not heavily regulated by multiple government agencies, and unless dealt with in a solution, will kill any platform sell.
3. Fix some core technical gaps. Everything from the ability to run analytics at scale in a virtual environment to ensuring information processing and analytics authenticity are issues that need solutions and have to be fixed.

4. a. Explain working together of HDFS and Map Reduce

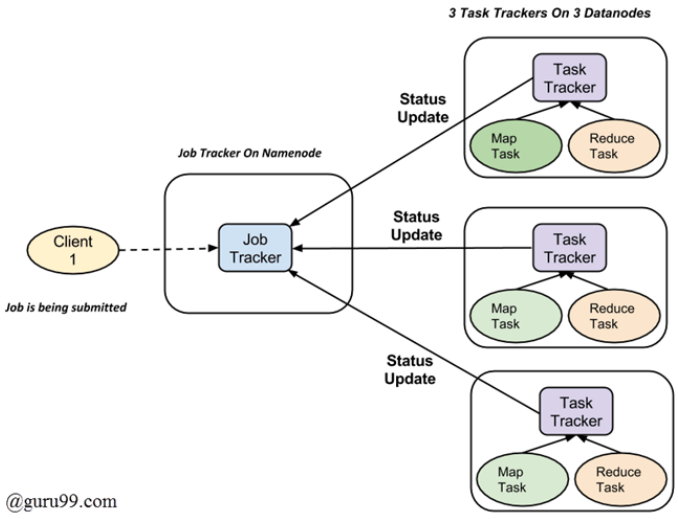
The two critical components of Hadoop are:

The Hadoop Distributed File System (HDFS)
MapReduce

The Hadoop Distributed File System (HDFS): HDFS is the storage system for a Hadoop cluster. When data lands in the cluster, HDFS breaks it into pieces and distributes those pieces among the different servers participating in the cluster. Each server stores just a small fragment of the complete data set, and each piece of data is replicated on more than one server.



MapReduce: Because Hadoop stores the entire dataset in small pieces across a collection of servers, analytical jobs can be distributed, in parallel, to each of the servers storing part of the data. Each server evaluates the question against its local fragment simultaneously and reports its results back for collation into a comprehensive answer. MapReduce is the agent that distributes the work and collects the results.



Both HDFS and MapReduce are designed to continue to work in the face of system failure. HDFS continually monitors the data stored on the cluster. If a server becomes unavailable, a disk drive fails, or data is damaged, whether due to hardware or software problems, HDFS automatically restores the data from one of the known good replicas stored elsewhere on the cluster. Likewise, when an analysis job is running, MapReduce monitors progress of each of the servers participating in the job. If one of them is slow in returning an answer or fails before completing its work, MapReduce automatically starts another instance of that task on another server that has a copy of the data. Because of the way that HDFS and MapReduce work, Hadoop provides scalable, reliable, and fault-tolerant services for data storage and analysis at very low cost.

b. **Define crowd sourcing. Explain different types of crowd sourcing.**

Crowdsourcing is a great way to capitalize on the resources that can build algorithms and predictive models

Kaggle: Kaggle describes itself as "an innovative solution for statistical/analytics outsourcing." That's a very formal way of saying that Kaggle manages competitions among the world's best data scientists. Here's how it works: Corporations, governments, and research laboratories are confronted with complex statistical challenges. They describe the problems to Kaggle and provide data sets. Kaggle converts the problems and the data into contests that are posted on its web site. The contests feature cash prizes ranging in value from $100 to $3 million. Kaggle's clients range in size from tiny start-ups to multinational corporations such as Ford Motor Company and government agencies such as NASA.

As per Anthony Goldbloom, Kaggle's founder and CEO: The idea is that someone comes to us with a problem, we put it up on our website, and then people from all over the world can compete to see who can produce the best solution."

Kaggle's approach is that it is truly a win-win scenario—contestants get access to real- world data (that has been carefully "anonymized" to eliminate privacy concerns) and prize sponsors reap the benefits of the contestants' creativity.Crowdsourcing is a disruptive business model whose roots are in technology but is extending beyond technology to other areas.

There are various types of crowd sourcing, such as crowd voting, crowd purchasing, wisdom of crowds, crowd funding, and contests.

Take for example:

- 99designs.com/, which does crowdsourcing of graphic design
- agentanything.com/, which posts "missions" where agents vie for to run errands
- 33needs.com/, which allows people to contribute to charitable programs that make a social impact

5. a. Compare between Big Data, RDBMS and Grid computing

Relational Database Management Systems

Why can't we use databases with lots of disks to do large-scale analysis? Why is Hadoop needed?

The answer to these questions comes from another trend in disk drives: seek time is improving more slowly than transfer rate. Seeking is the process of moving the disk's head to a particular place on the disk to read or write data. It characterizes the latency of a disk operation, whereas the transfer rate corresponds to a disk's bandwidth.

If the data access pattern is dominated by seeks, it will take longer to read or write large portions of the dataset than streaming through it, which operates at the transfer rate. On the other hand, for updating a small proportion of records in a database, a traditional B-Tree (the data structure used in relational databases, which is limited by the rate at which it can perform seeks) works well. For updating the majority of a database, a B-Tree is less efficient than MapReduce, which uses Sort/Merge to rebuild the database.

In many ways, MapReduce can be seen as a complement to a Relational Database Management System (RDBMS). (The differences between the two systems are shown in Table 1-1.) MapReduce is a good fit for problems that need to analyze the whole dataset in a batch fashion, particularly for ad hoc analysis. An RDBMS is good for point queries or updates, where the dataset has been indexed to deliver low-latency retrieval and update times of a relatively small amount of data. MapReduce suits applications where the data is written once and read many times, whereas a relational database is good for datasets that are continually updated.[5]

Table 1-1. RDBMS compared to MapReduce

	Traditional RDBMS	MapReduce
Data size	Gigabytes	Petabytes
Access	Interactive and batch	Batch
Updates	Read and write many times	Write once, read many times
Transactions	ACID	None

Grid Computing:

The High Performance Computing (HPC) and Grid Computing communities have been doing large-scale data processing for years, using such APIs as Message Passing Interface (MPI). Broadly, the approach in HPC is to distribute the work across a cluster of machines, which access a shared filesystem, hosted by a SAN. This works well for predominantly compute-intensive jobs, but becomes a problem when nodes need to access larger data volumes (hundreds of gigabytes, the point at which MapReduce really starts to shine), since the network bandwidth is the bottleneck and compute nodes become idle.

MapReduce tries to collocate the data with the compute node, so data access is fast since it is local. This feature, known as data locality, is at the heart of MapReduce and is the reason for its good performance. Recognizing that network bandwidth is the most precious resource in a data center environment (it is easy to saturate network links by copying data around), MapReduce implementations go to great lengths to conserve it by explicitly modelling network topology. Notice that this arrangement does not preclude high-CPU analyses in MapReduce

b. Write a note on i) history of Hadoop ii) Volunteer Computing

History of Hadoop:

- There are many Big Data technologies that have been making an impact on the new technology stacks for handling Big Data, but Apache Hadoop is one technology that has been the darling of Big Data talk. Hadoop is an open-source platform for storage and processing of diverse data types that enables data-driven enterprises to rapidly derive the complete value from all their data.
- The original creators of Hadoop are Doug Cutting and Mike Cafarella. They were building a model called "Nutch" with the goal of creating a large web index. They integrated the concepts from MapReduce and GFS into Nutch; then later these two components were pulled out to form the genesis of the Hadoop project.
- The name Hadoop itself comes from the Doug's son, he just made the word up for a yellow plush elephant toy that he has. Yahoo! Hired Doug and invested significant resources into growing the Hadoop project, initially to score and index the web for the purpose of Yahoo! Search.
- Hadoop gives organizations the flexibility to ask questions across their structured and unstructured data that were previously impossible to ask or solve: The scale and variety of data have permanently overwhelmed the ability to cost-effectively value using traditional platforms.

- The scalability and elasticity of free, open-source Hadoop running on standard hardware allow organizations to hold onto more data than ever before. Hadoop excels at supporting complex analyses across large collection of data.
- Hadoop handle variety of workloads, including search, log processing, recommendation system, data warehousing and video/image analysis. Today's explosion of data types and volumes means that Big Data equals big opportunities and Apache empowers organizations to work on the most modern scale out architectures.
- Apache Hadoop is an open source project administered by the Apache Software Foundation. The software was originally developed by the world's largest internet companies to capture and analyze the data that they generate.

Volunteer Computing:

Volunteer computing projects work by breaking the problem they are trying to solve into chunks called work units, which are sent to computers around the world to be analyzed. For example, a SETI@home work unit is about 0.35 MB of radio telescope data, and takes hours or days to analyze on a typical home computer. When the analysis is completed, the results are sent back to the server, and the client gets another work unit. As a precaution to combat cheating, each work unit is sent to three different machines and needs at least two results to agree to be accepted. Although SETI@home may be superficially similar to Map Reduce

6. a. Illustrate the disabilities of implementing storage and analysis support of big data

Data Storage and Analysis

The problem is simple: although the storage capacities of hard drives have increased massively over the years, access speeds—the rate at which data can be read from drives—have not kept up. One typical drive from 1990 could store 1,370 MB of data and had a transfer speed of 4.4 MB/s,[4] so you could read all the data from a full drive in around five minutes. Over 20 years later, 1-terabyte drives are the norm, but the transfer speed is around 100 MB/s, so it takes more than two and a half hours to read all the data off the disk.

This is a long time to read all data on a single drive—and writing is even slower. The obvious way to reduce the time is to read from multiple disks at once. Imagine if we had 100 drives, each holding one hundredth of the data. Working in parallel, we could read the data in under two minutes.

Using only one hundredth of a disk may seem wasteful. But we can store 100 datasets, each of which is 1 terabyte, and provide shared access to them. We can imagine that the users of such a system would be happy to share access in return for shorter analysis times, and statistically, that their analysis jobs would be likely to be spread over time, so they wouldn't interfere with each other too much.

There's more to being able to read and write data in parallel to or from multiple disks, though.

The first problem to solve is hardware failure: as soon as you start using many pieces of hardware, the chance that one will fail is fairly high. A common way of avoiding data loss is through replication: redundant copies of the data are kept by the system so that in the event of failure, there is another copy available. This is how RAID works, for instance, although Hadoop's filesystem, the Hadoop Distributed Filesystem (HDFS), takes a slightly different approach, as you shall see later.

The second problem is that most analysis tasks need to be able to combine the data in some way, and data read from one disk may need to be combined with data from any of the other 99 disks. Various distributed systems allow data to be combined from multiple sources, but doing this correctly is notoriously challenging. MapReduce provides a programming model that abstracts the problem from disk reads and writes, transforming it into a computation over sets of keys and values. We look at the details of this model in later chapters, but the important point for the present discussion is that there are two parts to the computation—the map and the reduce—and it's the interface between the two where the "mixing" occurs. Like HDFS, MapReduce has built-in reliability.

b. Explain different Hadoop Versions and releases.

Table 1-2. Features supported by Hadoop release series

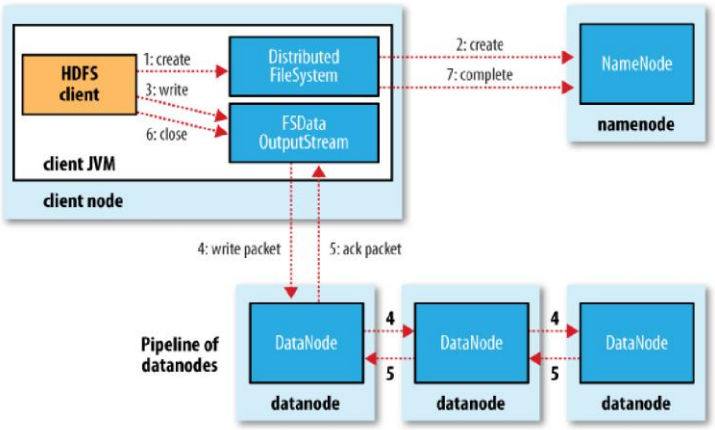
Feature	1.x	0.22	0.23
Secure authentication	Yes	No	Yes
Old configuration names	Yes	Deprecated	Deprecated
New configuration names	No	Yes	Yes
Old MapReduce API	Yes	Deprecated	Deprecated
New MapReduce API	Partial	Yes	Yes
MapReduce 1 runtime (Classic)	Yes	Yes	No
MapReduce 2 runtime (YARN)	No	No	Yes
HDFS federation	No	No	Yes
HDFS high-availability	No	No	Planned

7. a. Briefly explain the anatomy of file write.

The case we're going to consider is the case of creating a new file, writing data to it,
then closing the file. See Figure 3-4.The client creates the file by calling create() on DistributedFileSystem (step 1 in Figure 3-4). DistributedFileSystem makes an RPC call to the namenode to create a new file in the filesystem's namespace, with no blocks associated with it (step 2). The namenode performs various checks to make sure the file doesn't already exist, and that the client has the right permissions to create the file. If these checks pass, the namenode makes a record of the new file; otherwise, file creation fails and the client is thrown an IOException.

The DistributedFileSystem returns an FSDataOutputStream for the client to start writing data to. Just as in the read case, FSDataOutputStream wraps a DFSOutput Stream, which handles communication with the datanodes and namenode.

As the client writes data (step 3), DFSOutputStream splits it into packets, which it writes to an internal queue, called the data queue. The data queue is consumed by the Data Streamer, whose responsibility it is to ask the namenode to allocate new blocks by
picking a list of suitable datanodes to store the replicas. The list of datanodes forms a
pipeline—we'll assume the replication level is three, so there are three nodes in the
pipeline. The DataStreamer streams the packets to the first datanode in the pipeline,
which stores the packet and forwards it to the second datanode in the pipeline. Similarly,the second datanode stores the packet and forwards it to the third (and last)
datanode in the pipeline (step 4).



b. Explain different HDFS concept in detail.

HDFS concepts:

A disk has a block size, which is the minimum amount of data that it can read or write. Filesystems for a single disk build on this by dealing with data in blocks, which are an integral multiple of the disk block size. Filesystem blocks are typically a few kilobytes in size, while disk blocks are normally 512 bytes. This is generally transparent to the filesystem user who is simply reading or writing a file — of whatever length. However,there are tools to perform filesystem maintenance, such as df and fsck, that operate on the filesystem block level.
HDFS, too, has the concept of a block, but it is a much larger unit—64 MB by default.Like in a filesystem for a single disk, files in HDFS are broken into block-sized chunks,which are stored as independent units.

Name nodes and Data nodes:

An HDFS cluster has two types of node operating in a master-worker pattern: a namenode (the master) and a number of datanodes (workers). The namenode manages the filesystem namespace. It maintains the filesystem tree and the metadata for all the files and directories in the tree. This information is stored persistently on the local disk in the form of two files: the namespace image and the edit log. The namenode also knows the datanodes on which all the blocks for a given file are located, however, it doesnot store block locations persistently, since this information is reconstructed fromdatanodes when the system starts.

A client accesses the filesystem on behalf of the user by communicating with the namenode and datanodes. The client presents a POSIX-like filesystem interface, so the user code does not need to know about the namenode and datanode to function.Datanodes are the workhorses of the filesystem. They store and retrieve blocks when they are told to (by clients or the namenode), and they report back to the namenode periodically with lists of blocks that they are storing.

8. a. Discuss different file system operations.

The filesystem is ready to be used, and we can do all of the usual filesystem operations
% hadoop fs -mkdir books

Creates a directory called books on the Hadoop File system. In the above command hadoop

indicates it is a Hadoop command

Fs file system command

-mkdir command to execute on hdfs, ie creation of a directory Books

% hadoop fs -ls .

Found 2 items

drwxr-xr-x - tom supergroup 0 2009-04-02 22:41 /user/tom/books

-rw-r--r-- 1 tom supergroup 118 2009-04-02 22:29 /user/tom/quangle.txt

The information returned is very similar to the Unix command ls -l, with a few minor differences. The first column shows the file mode. The second column is the replication factor of the file (something a traditional Unix filesystem does not have). Remember we set the default replication factor in the site-wide configuration to be 1, which is why we see the same value here. The entry in this column is empty for directories since the concept of replication does not apply to them—directories are treated as metadata and stored by the namenode, not the datanodes. The third and fourth columns show the file owner and group. The fifth column is the size of the file in bytes, or zero for directories. The sixth and seventh columns are the last modified date and time. Finally, the eighth column is the absolute name of the file or directory.

%hadoop fs -help <<command>>

- to get detailed help on every command.

% hadoop fs -copyFromLocal input/docs/quangle.txt

hdfs://localhost/user/tom/quangle.txt

% hadoop fs -copyFromLocal input/docs/quangle.txt

/user/tom/quangle.txt

% hadoop fs -copyFromLocal input/docs/quangle.txt quangle.txt

- Start by copying a file from the local filesystem to HDFS

% Hadoop fs -copyToLocal hdfs://Buangle.txt c:/input/docs/Buangle.txt

- Start by copying a file from the HDFS to local filesystem

Hadoop FS changing the file permissions Command: chmod

hadoop fs -chmod [-R] <MODE[,MODE]...| OCTALMODE> URI [URI …]

- Change the permissions of files. With -R, make the change recursively through the directory structure. The user must be the owner of the file, or else a super-user.

% hadoop fs -chmod 744 /user/training/samplezero.txt

b. Define file system interface. Explain different file system interfaces.

File system	URI scheme	Java implementation	Description
Local	file	fs.LocalFileSystem	A filesystem for a locally connected disk with clientside checksums.
HDFS	hdfs	hdfs.DistributedFileSystem	Hadoop's distributed filesystem. HDFS is designed to work efficiently in conjunction with MapReduce.
HFTP	hftp	hdfs.HftpFileSystem	A filesystem providing read-only access to HDFS over HTTP.
HSFTP	hsftp	hdfs.HsftpFileSystem	A filesystem providing read-only access to HDFS over HTTPS.
WebHDFS	webhdfs	hdfs.web.WebHdfsFileSystem	A filesystem providing secure read-write access to HDFS over HTTP. WebHDFS is intended as a replacement for HFTP and HSFTP.
HAR	har	fs.HarFileSystem	A filesystem layered on another filesystem for archiving files. Hadoop Archives are typically used for archiving files in HDFS to

			reduce the namenode's memory usage.
KFS (Cloud-Store)	kfs	fs.kfs.KosmosFileSystem	CloudStore (formerly Kosmos filesystem) is a distributed filesystem like HDFS or Google's GFS, written in C++.
FTP	ftp	fs.ftp.FTPFileSystem.	A filesystem backed by an FTP server
S3 (native)	s3n	fs.s3native.NativeS3FileSystem	A filesystem backed by Amazon S3.
S3 (block based)	s3	fs.s3.S3FileSystem	A filesystem backed by Amazon S3, which stores files in blocks (much like HDFS) to overcome S3's 5 GB file size limit.
Distributed RAID	hdfs	hdfs.DistributedRaidFileSystem	A "RAID" version of HDFS designed for archival storage.
View	viewfs	viewfs.ViewFileSystem	A client-side mount table for other Hadoop filesystems.

9. a. **What is Map Reduce? Sketch the neat diagram and explain the logical data flow in mapreduce.**

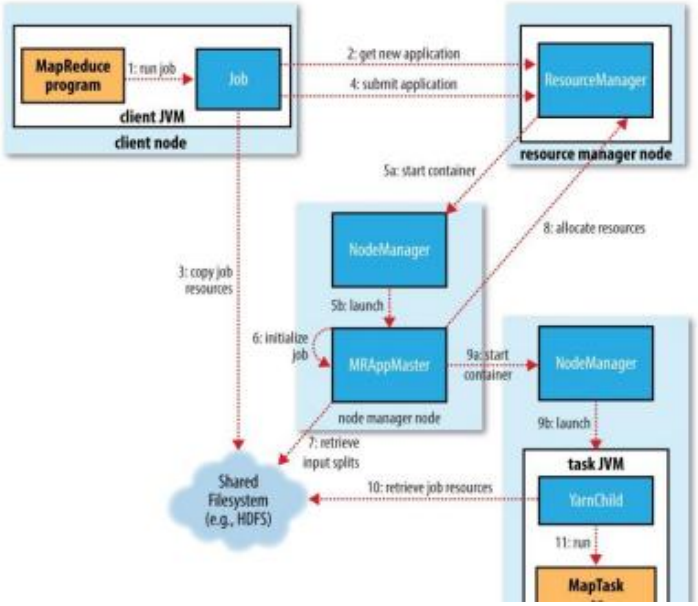
Map Reduce:

MapReduce is a programming model and an associated implementation for processing and generating big data sets with a parallel, distributed algorithm on a cluster. A MapReduce program is composed of a map procedure, which performs filtering and sorting, and a reduce method, which performs a summary operation.

- One map task is created for each split which then executes map function for each record in the split.
- It is always beneficial to have multiple splits, because time taken to process a split is small as compared to the time taken for processing of the whole input. When the splits are smaller, the processing is better load balanced since we are processing the splits in parallel.
- However, it is also not desirable to have splits too small in size. When splits are too small, the overload of managing the splits and map task creation begins to dominate the total job execution time.
- For most jobs, it is better to make split size equal to the size of an HDFS block (which is 64 MB, by default).
- Execution of map tasks results into writing output to a local disk on the respective node and not to HDFS.
- Reason for choosing local disk over HDFS is, to avoid replication which takes place in case of HDFS store operation.
- Map output is intermediate output which is processed by reduce tasks to produce the final output.
- Once the job is complete, the map output can be thrown away. So, storing it in HDFS with replication becomes overkill.
  In the event of node failure before the map output is consumed by the reduce task, Hadoop reruns the map task on another node and re-creates the map output.
- Reduce task don't work on the concept of data locality. Output of every map task is fed

to the reduce task. Map output is transferred to the machine where reduce task is running.

- On this machine the output is merged and then passed to the user defined reduce function.
- Unlike to the map output, reduce output is stored in HDFS (the first replica is stored on the local node and other replicas are stored on off-rack nodes). So, writing the reduce output



b. Write a short note on: i) MapReduceUI ii) Hadoop Streaming

The MapReduce Web UI

Hadoop comes with a web UI for viewing information about your jobs. It is useful for following a job's progress while it is running, as well as finding job statistics and logs after the job has completed. You can find the UI at *http://resource-manager-host:8088/*.

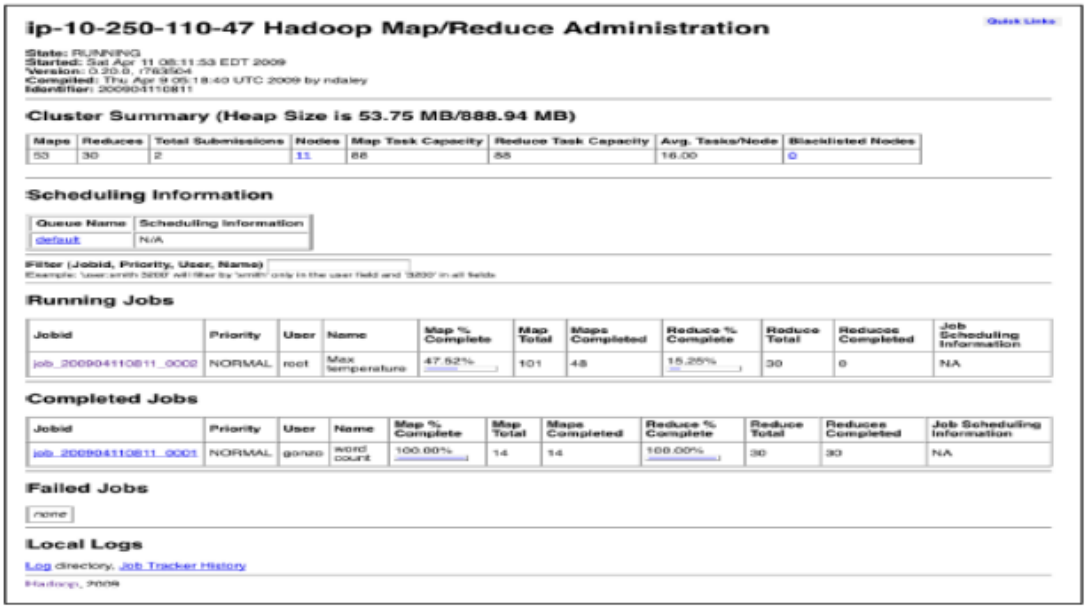
Retrieving the Results

Once the job is finished, there are various ways to retrieve the results. Each reducer produces one output file, so there are 30 part files named *part-r-00000* to *part-r-00029* in the *max-temp* directory.

As their names suggest, a good way to think of these "part" files is as parts of the *max-temp* "file."

If the output is large (which it isn't in this case), it is important to have multiple parts so that more than one reducer can work in parallel. Usually, if a file is in this partitioned form, it can still be used easily enough—as the input to another MapReduce job, for example. In some cases, you can exploit the structure of multiple partitions to do a map-side join, for example (see "Map-Side Joins" on page 269).

MapReduceUI:



Hadoop Streaming:

Hadoop provides an API to MapReduce that allows you to write your map and reduce functions in languages other than Java. Hadoop Streaming uses Unix standard streams as the interface between Hadoop and your program, so you can use any language that can read standard input and write to standard output to write your MapReduce program. Streaming is naturally suited for text processing (although, as of version 0.21.0, it can handle binary streams, too), and when used in text mode, it has a line-oriented view of data. Map input data is passed over standard input to your map function, which processes it line by line and writes lines to standard output. A map output key-value pair is written as a single tab-delimited line. Input to the reduce function is in the same format—a tab-separated key-value pair—passed over standard input. The reduce function reads lines from standard input, which the framework guarantees are sorted by key, and writes its results to standard output.

10. a. Write a Java Map Reduce code to find maximum temperature from the weather data set.

Example 2-3. Mapper for maximum temperature example

```
import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class MaxTemperatureMapper
  extends Mapper<LongWritable, Text, Text, IntWritable> {

  private static final int MISSING = 9999;

  @Override
  public void map(LongWritable key, Text value, Context context)
      throws IOException, InterruptedException {

    String line = value.toString();
    String year = line.substring(15, 19);
    int airTemperature;
    if (line.charAt(87) == '+') { // parseInt doesn't like leading plus signs
      airTemperature = Integer.parseInt(line.substring(88, 92));
    } else {
      airTemperature = Integer.parseInt(line.substring(87, 92));
    }
    String quality = line.substring(92, 93);
    if (airTemperature != MISSING && quality.matches("[01459]")) {
      context.write(new Text(year), new IntWritable(airTemperature));
    }
  }
}
```

Example 2-4. Reducer for maximum temperature example

```
import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class MaxTemperatureReducer
  extends Reducer<Text, IntWritable, Text, IntWritable> {

  @Override
  public void reduce(Text key, Iterable<IntWritable> values,
      Context context)
      throws IOException, InterruptedException {

    int maxValue = Integer.MIN_VALUE;
    for (IntWritable value : values) {
      maxValue = Math.max(maxValue, value.get());
    }
    context.write(key, new IntWritable(maxValue));
  }
}
```

Example 2-5. Application to find the maximum temperature in the weather dataset

```
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class MaxTemperature {

  public static void main(String[] args) throws Exception {
    if (args.length != 2) {
      System.err.println("Usage: MaxTemperature <input path> <output path>");
      System.exit(-1);
    }

    Job job = new Job();
    job.setJarByClass(MaxTemperature.class);
    job.setJobName("Max temperature");

    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    job.setMapperClass(MaxTemperatureMapper.class);
    job.setReducerClass(MaxTemperatureReducer.class);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

b. Write a note on: i) Hadoop logs ii) Remote Debugging

Hadoop Logs:

Logs	Primary audience	Description	Further information
System daemon logs	Administrators	Each Hadoop daemon produces a logfile (using log4j) and another file that combines standard out and error. Written in the directory defined by the HADOOP_LOG_DIR environment variable.	"System logfiles" on page 307 and "Logging" on page 349.
HDFS audit logs	Administrators	A log of all HDFS requests, turned off by default. Written to the namenode's log, although this is configurable.	"Audit Logging" on page 344.

Logs	Primary audience	Description	Further information
MapReduce job history logs	Users	A log of the events (such as task completion) that occur in the course of running a job. Saved centrally on the jobtracker, and in the job's output directory in a *_logs/history* subdirectory.	"Job History" on page 166.
MapReduce task logs	Users	Each tasktracker child process produces a logfile using log4j (called *syslog*), a file for data sent to standard out (*stdout*), and a file for standard error (*stderr*). Written in the *userlogs* subdirectory of the directory defined by the HADOOP_LOG_DIR environment variable.	This section.