TABLE OF CONTENTS

PART 1. READ THE FOLLOWING PAPER (ATTACHED), AND WRITE A SHORT SUMMARY/REPORT.

THE GOOGLE FILE SYSTEM

Google File System (GFS) is a scalable, distributed file system for large distributed, data-intensive applications. It provides fault tolerance when running on inexpensive, common hardware and provides high aggregate performance for a large number of clients. The challenges we encountered were component failures, huge files, new data additions, and increased application flexibility. So when designing, we made some assumptions: the system consists of many inexpensive commodity components that often fail; the system stores a small number of large files; the workload mainly includes large stream reads and small stream reads; the system must effectively Simultaneously, multiple clients attached to the same file implement well-defined semantics; high sustained bandwidth is more important than low latency.

A GFS cluster consists of a primary server and multiple block servers and is accessed by multiple clients, as shown in Figure 1.

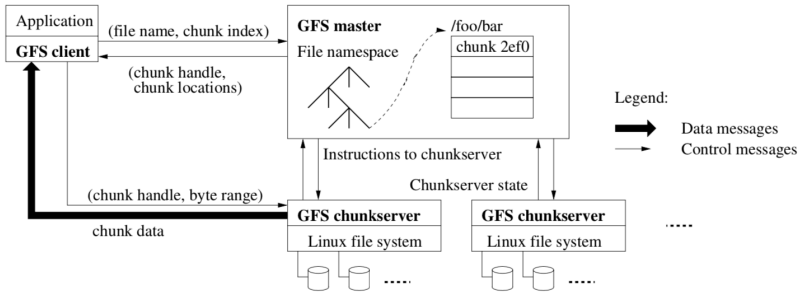


Figure 1: GFS Architecture

Files are divided into fixed-size blocks, each of which is copied to multiple block servers, with three copies being stored by default. The host maintains all file system metadata. This includes namespaces, access control information, mapping from file to block, and the current location of the block. The GFS client code linked to each application implements a file system API and communicates with the primary server and the block server to read or write data on behalf of the application. Instead of reading and writing file data through the primary server, the client asks which block servers the primary server should contact.

Block size is one of the key design parameters, you need to choose the right size, otherwise it will affect the speed and hot issues. All metadata is stored in the host's memory. GFS has a loose consistency model.

The host performs all namespace operations. In addition, it manages block copies throughout the system: it can make placement decisions, create new blocks and thus create replicas, and coordinate various system-wide activities to keep blocks fully replicated, balancing the load on all block servers and Recycle unused blocks.

A component failure can cause the system to become unavailable or, worse, corrupt the data. We have built-in tools in the system to diagnose problems in the unavoidable situation.

Fast recovery, block replication, primary replication, diagnostic logging. Optimized for large files, these files are usually attached (perhaps simultaneously) and then read (usually in sequential order), and both extend and relax the standard file system interface to improve the overall system.

The system can replicate critical data and fast auto-recovery capabilities to provide fault tolerance. Block replication allows us to tolerate chunkserver failures. In addition, we use checksums to detect data corruption at the disk or IDE subsystem level, given the number of disks in the system.

Provides high total throughput. We do this by separating the file system control through the host from the data transfer passed directly between the block server and the client. The size of large blocks and the leasing of blocks minimizes the impact of the main operation on common operations, which grants permissions to the master copy in the data mutation. This makes it a bottleneck to become a simple, centralized primary server. We believe that improvements to the network stack will remove the current limit on the write throughput seen by a single client.

MAPREDUCE: SIMPLIFIED DATA PROCESSING ON LARGE CLUSTERS

MapReduce is a programming model and associated implementation for processing and generating large data sets. The user specifies a mapping function that processes key/value pairs to generate a set of intermediate key/value pairs, and a reduction function that combines all intermediate values associated with the same intermediate key. The runtime system is responsible for partitioning the input data and is highly scalable.

The input data is large and must be distributed over hundreds or thousands of computers to complete the calculation in a reasonable amount of time. To cope with this complexity we use mapping, compute a set of key/value pairs, and then perform a reduction operation key on all values sharing the same value to properly combine the exported data.

Users of the MapReduce library represent calculations as two functions: Map and Reduce.

Map accepts an input pair and produces a set of intermediate key/value pairs. Combine all the intermediate values associated with the same intermediate key and pass them to the Reduce function.

The Reduce function accepts a set of values for the intermediate key and the key, which combines the values together to form a possibly smaller set of values. Typically, each Reduce call produces only zero or one output value. The intermediate value is provided to the user's reduce function through the iterator.

The input data is automatically divided into M splits. The Map is distributed on multiple computers and can be processed in parallel by different machines. The partition function divides the intermediate key space into R segments. One of the copies is very special - the master. The staff assigned the map task will resolve the key/value pairs from the input data and pass each pair to the user-defined Map function. The reduce worker iterates through the sorted intermediate data and passes the key and corresponding intermediate value set to the user's Reduce function for each unique intermediate key encountered. The output of the Reduce function will be appended to the final output file of this reduce partition.

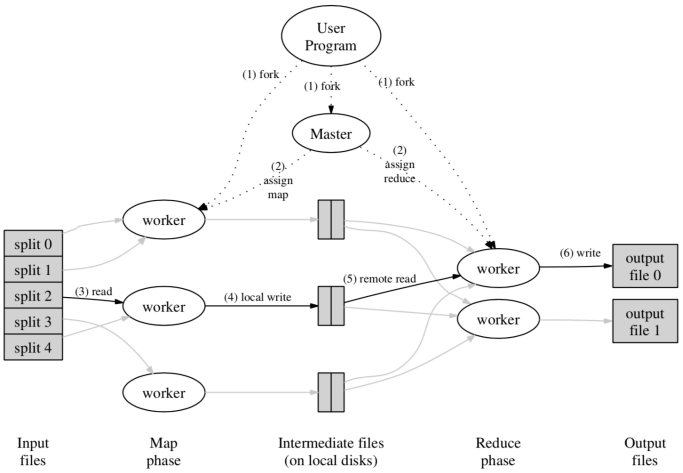


Figure 1: Execution overview

One of the common reasons for the total time spent on "MapReduce" operations is that the machine spends an unusually long time completing one of the last few maps or reducing the computational task. When the MapReduce operation is nearing completion, the primary server schedules backup execution of the remaining ongoing tasks. The task is marked as completed whenever the main execution or backup execution is completed. This greatly reduces the time to complete large MapReduce operations.

Users of the MapReduce library can provide special partitioning capabilities. In a given partition, intermediate key/value pairs are processed in increasing key order. The intermediate keys generated by each mapping task are repeated in large numbers, and the user-specified Reduce functions are interchangeable and associative. We allow the user to specify an optional Combiner function that will partially merge before sending data over the network.

The reader does not necessarily need to provide data read from the file. For example, defining a reader is easy, and it can read records from a database or a data structure that is mapped into memory.

In a similar manner, we support a set of output types for generating data in different formats, and user code can easily add support for new output types.

The MapReduce library provides support for reading in- put data in several different formats. For example, “text” mode input treats each line as a key/value pair: the key is the offset in the file and the value is the contents of the line. Another common supported format stores a sequence of key/value pairs sorted by key. Each input type implementation knows how to split itself into mean- ingful ranges for processing as separate map tasks (e.g. text mode’s range splitting ensures that range splits oc- cur only at line boundaries). Users can add support for a new input type by providing an implementation of a sim- ple *reader* interface, though most users just use one of a small number of predefined input types. A *reader* does not necessarily need to provide data read from a file.

MR can skip bad records. It can be executed locally, can get status information, can provide counters, can be sorted, for data mining, for machine learning and many other systems, can be extended to large machine clusters.

BIGTABLE: A DISTRIBUTED STORAGE SYSTEM FOR STRUCTURED DATA

Bigtable is a distributed storage system for managing structured data that is designed to scale to very large scales: petabytes of data in thousands of commercial servers. Bigtable achieves multiple goals: broad applicability, scalability, high performance and high availability. In many ways, Bigtable is similar to a database: it shares many implementation strategies with the database, achieving scalability and high performance, but Bigtable provides a different interface than such systems. Bigtable provides customers with a simple data model that supports dynamic control of data layout and formatting and allows customers to reason about the locality of the data represented in the underlying storage. Bigtable also treats data as uninterpreted strings. The Bigtable mode parameter allows the client to dynamically control whether data is provided from memory or from disk.

Bigtable is a sparse, distributed, and persistent multidimensional sort map. The map is indexed by row keys, column keys, and timestamps; each value in the map is an uninterpreted byte array. Bigtable uses the row keys to maintain data in lexicographic order. Clients can take advantage of attributes by selecting row keys so they can do good data access.

Column keys are divided into sets called column families, which form the basic unit of access control. All data stored in a column family is usually of the same type (we compress the data in the same column family together). You must create a column family before you can store data under any of the column keys in that family. Once you have created a family, you can use any of the column keys in the family. The column key is named using the following syntax: family:qualifier.

Each cell in Bigtable can contain multiple versions of the same data; these versions are indexed by timestamps.

The Bigtable API provides the ability to create and drop tables and column families. It also provides the ability to change cluster, table, and column family metadata. Bigtable can be used in conjunction with MapReduce. Bigtable uses the Distributed Google File System (GFS) [17] to store log and data files.

The lesson we learn is that large distributed systems are vulnerable to many types of failures, not just the standard network partitioning and failure-stopping faults assumed in many distributed protocols. We added a checksum to the RPC mechanism. We also deal with some of the problems by eliminating the assumptions made by one part of the system on the other.

It's important to delay adding new features until you know how to use the new features. When people ask for distributed transactions, the most important use is to maintain secondary indexes, and we plan to add a special mechanism to meet this need. The new mechanism will not be as common as distributed transactions, but will be more efficient and will also interact better with our optimistic cross-data center replication approach.

THE CHUBBY LOCK SERVICE FOR LOOSELY COUPLED DISTRIBUTED SYSTEMS

The Chubby Lock Service is designed to provide coarse-grained locks and reliable storage for loosely coupled distributed systems in a loosely coupled distributed system of medium-sized small computers connected by high-speed networks.

The coarse-grained lock has a much smaller load on the lock server. Moving a lock from a client to a client can require an expensive recovery process that allows a coarse-grained lock to handle lock server failures well, and this lock allows a modest number of less-available lock servers for many clients. Provide adequate services. Locking server failover does not introduce a new recovery path.

Chubby has two main components that communicate through RPC: the library to which the server and client applications are linked. The Chubby unit consists of a small group of servers called replication servers to reduce the likelihood of related failures. The replica uses a distributed consensus protocol to elect the primary server. The master copy must obtain voting rights from most copies and promise that these copies will not elect other master copies in a matter of seconds. If the master copy continues to win a majority vote, the copy will periodically renew the master copy lease.

A copy maintains a copy of a simple database, but only the primary copy initiates reads and writes to that database. All other copies only replicate updates from the primary server sent using the consensus protocol. Non-primary copies respond to such requests by returning to the primary identity. When the client finds the host, the client directs all requests to that host until it stops responding or indicates that it is no longer the host. Write requests are propagated to all replicas through a consensus protocol. Such requests are acknowledged when a write reaches most of the copies in the cell. The read request is only satisfied by the primary device; if the primary lease has not expired, this is safe because there may be no other primary lease. If one primary server fails, the other copies will run the election agreement when their primary server lease expires; otherwise, the other copies will run the election agreement. A new owner is usually selected in a few seconds. If the copy fails and cannot be recovered for hours, the simple replacement system will select a new calculation from the free pool. Then, replace the IP address of the failed copy with the IP address of the new copy. The current user periodically polls the DNS and eventually notices the change. It then updates the list of members of the cell in the cell database; the list is consistent across all members through the normal replication protocol. At the same time, the new copy gets the latest copy of the database from the combined combination of the backup and the active copy stored on the file server. Once the new copy processes the request that the current host is waiting to submit, the copy can vote in the election of the new host.

Nodes can be permanent or short-lived. Any node can be explicitly deleted, but if no client opens the temporary node, the temporary nodes will also be deleted (for directories, they are empty). Each node has various metadata, including three access control list (ACL) names that control the read, write, and change ACL names of the nodes. Unless overridden, the node inherits the ACL name of its parent directory when it is created. These ACL files consist of a simple list of subject names.

Chubby provides a way to introduce serial numbers into only those interactions that use locks. At any time, the lock holder can request a serial number to describe the state of the lock immediately after acquisition. Chubby provides a mechanism to reduce the risk of delaying or reordering requests for servers that do not support the sequencer. Customers can specify any lockout delay that prevents a faulty client from using the lock for long periods of time.

The Chubby client may subscribe to a series of events when creating a handle. These events are passed asynchronously to the client via an upstream call from the Chubby library. To reduce read traffic, the Chubby client caches file data and node metadata (including file missing) in a consistent write-through cache maintained in memory. When you want to change file data or metadata, the modifications are blocked and the primary server invalidates the data to each client that may have cached the data. Cache protocol: It invalidates the changed cached data and never updates it.

If the client's session remains valid, the client's handle, lock, and cached data will remain valid.

The client requests a new session the first time it contacts the owner of the Chubby unit. It explicitly ends the session when the session is terminated or when the session is idle. Each session has an associated lease-to-future interval during which the primary server guarantees that the session is not unilaterally terminated. The end of this

interval is called the session lease timeout. The host is free to advance the timeout to the future, but may not be able to move it backwards. The primary server leases the timeout in advance in three situations: when creating a session, when a primary failover occurs, and when responding to the client's KeepAlive RPC.

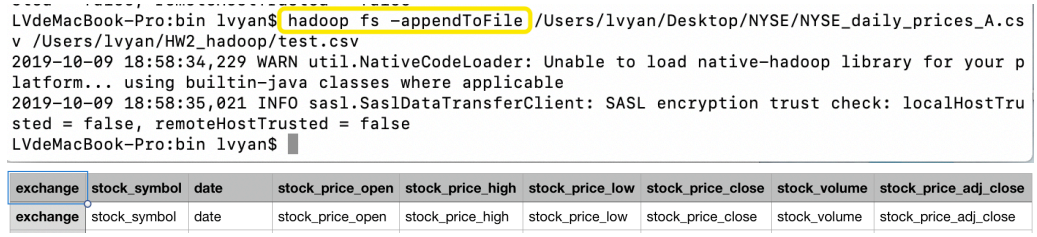
Every few hours, the master node of each Chubby unit writes a snapshot of its database to a GFS file server located in a different building. Using a separate building ensures that backups are not affected by building damage, and that backups do not introduce any circular dependencies into the system.

Chubby allows a set of files to be mirrored from one unit to another. The image is fast, because the file is small, and if you add, delete, or modify the file, the event mechanism immediately notifies the image. If the mirror cannot be accessed, it will remain unchanged until the connection is restored. The updated files are then determined by comparing their checksums.

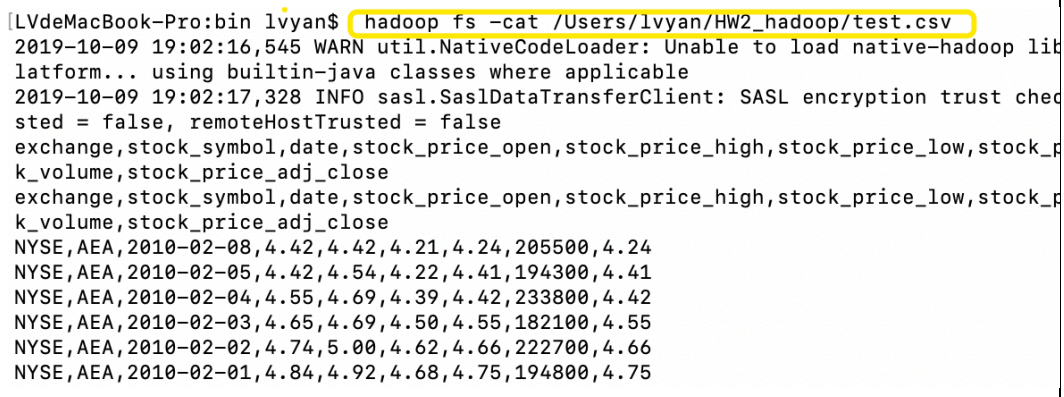
Chubby's expansion mechanisms include: Proxies and Partitioning

PART 2 – PROGRAMMING ASSIGNMENT

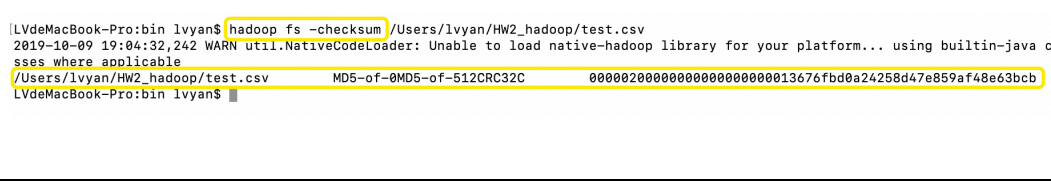
<table>
<tr><th>Command</th><th>Screenshot</th></tr>
<tr><td>appendToFile:

append NSYE_daily_price_A.csv after test.csv</td><td>


</td></tr>
<tr><td>Cat

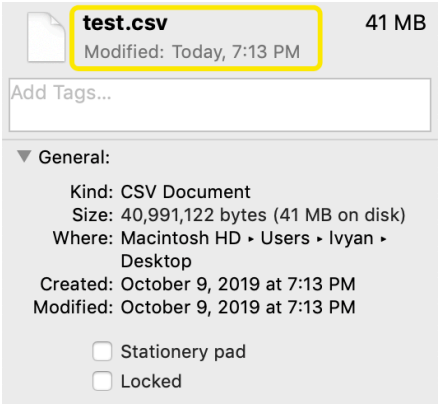
Used on the file test.csv</td><td>


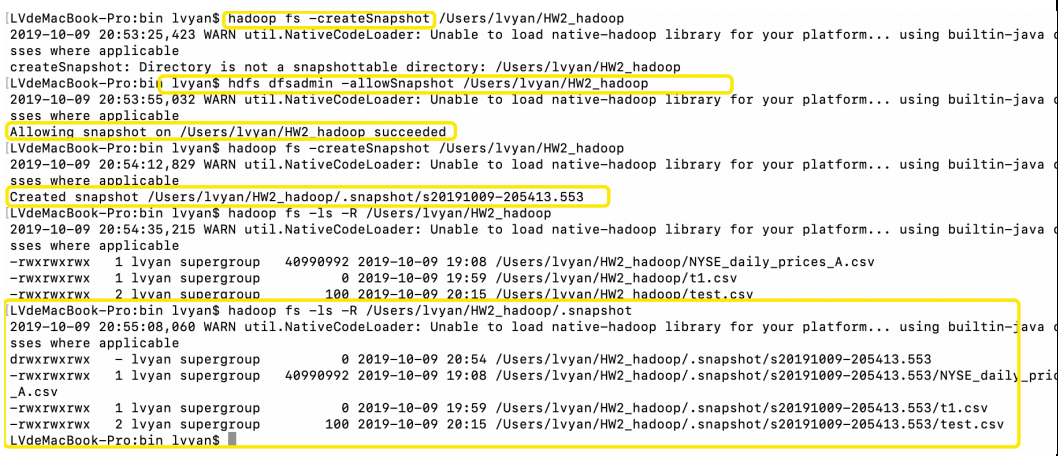
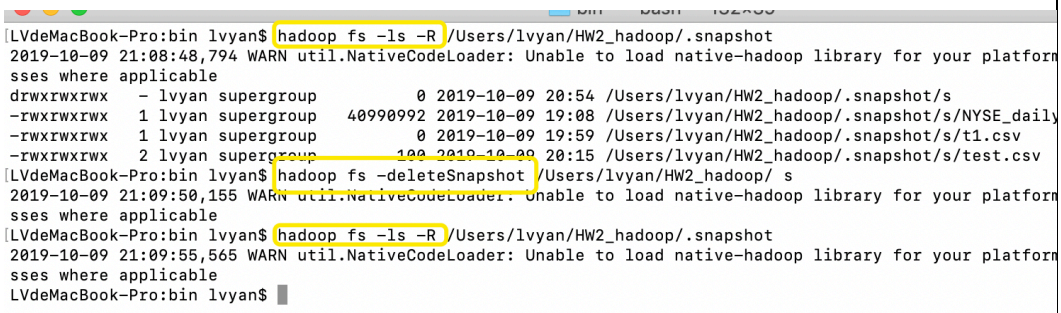
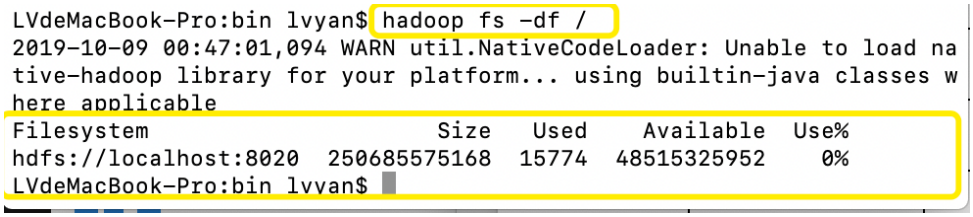
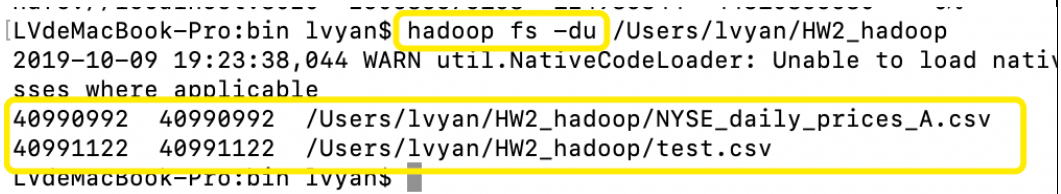
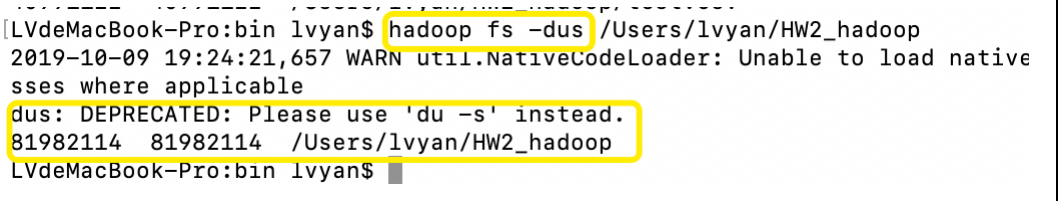
</td></tr>
<tr><td>Checksum

Check the sum of test.csv</td><td>


</td></tr>
</table>

Chgrp Change the group of NSYE_daily_price_A.csv to GROUP	Usage: hadoop fs [generic options] -chgrp [-R] GROUP PATH... LVdeMacBook-Pro:bin lvyan$ hadoop fs -chgrp GROUP /Users/lvyan/HW2_hadoop 2019-10-09 19:06:02,210 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable LVdeMacBook-Pro:bin lvyan$ LVdeMacBook-Pro:bin lvyan$ hadoop fs -ls /Users/lvyan/HW2_hadoop 2019-10-09 19:08:21,597 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable Found 2 items -rw-r--r-- 1 lvyan GROUP 40990992 2019-10-09 19:08 /Users/lvyan/HW2_hadoop/NYSE_daily_prices_A.csv -rw-r--r-- 1 lvyan supergroup 40991122 2019-10-09 18:58 /Users/lvyan/HW2_hadoop/test.csv
Chmod Change the permission to 777, which is shown as rwxrwxrwx	LVdeMacBook-Pro:bin lvyan$ ./hadoop fs -chmod -R 777 / 2019-10-09 00:52:19,152 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable LVdeMacBook-Pro:bin lvyan$ ./hadoop fs -ls / 2019-10-09 00:52:34,584 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable Found 2 items drwxrwxrwx - lvyan supergroup 0 2019-10-09 00:37 /Users drwxrwxrwx - lvyan supergroup 0 2019-10-03 15:42 /testdir LVdeMacBook-Pro:bin lvyan$
Chown Change the owner of file to 777	drwxrwxrwx - 777 supergroup 0 2019-10-09 00:37 /Users drwxrwxrwx - 777 supergroup 0 2019-10-03 15:42 /testdir LVdeMacBook-Pro:bin lvyan$ hadoop fs -chown -R lvyan / 2019-10-09 12:08:55,992 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable LVdeMacBook-Pro:bin lvyan$ hadoop fs -ls / 2019-10-09 12:08:59,666 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable Found 2 items drwxrwxrwx - lvyan supergroup 0 2019-10-09 00:37 /Users drwxrwxrwx - lvyan supergroup 0 2019-10-03 15:42 /testdir LVdeMacBook-Pro:bin lvyan$
copyFromLocal copy local file NSYE_daily_price_A.csv to hdfs direction named HW2_hadoop	LVdeMacBook-Pro:bin lvyan$ hadoop fs -copyFromLocal /Users/lvyan/Desktop/NYSE/NYSE_daily_prices_A.csv /Users/lvyan/HW2_hadoop 2019-10-09 19:08:06,969 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable 2019-10-09 19:08:07,783 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false LVdeMacBook-Pro:bin lvyan$ hadoop fs -ls /Users/lvyan/HW2_hadoop 2019-10-09 19:08:21,597 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable Found 2 items -rw-r--r-- 1 lvyan GROUP 40990992 2019-10-09 19:08 /Users/lvyan/HW2_hadoop/NYSE_daily_prices_A.csv -rw-r--r-- 1 lvyan supergroup 40991122 2019-10-09 18:58 /Users/lvyan/HW2_hadoop/test.csv LVdeMacBook-Pro:bin lvyan$

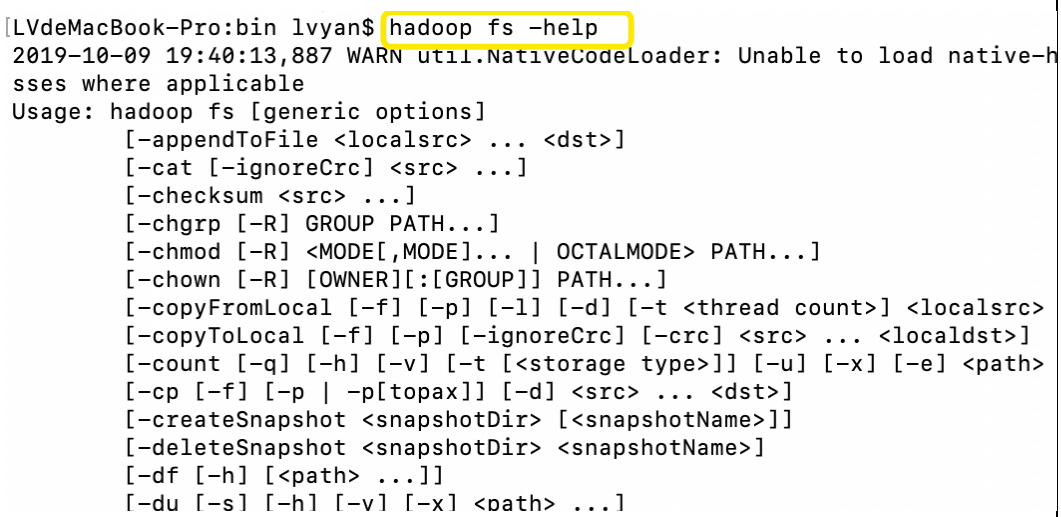
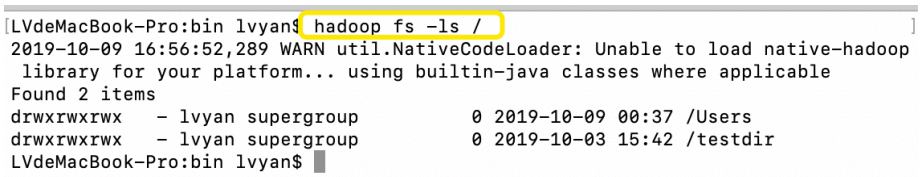
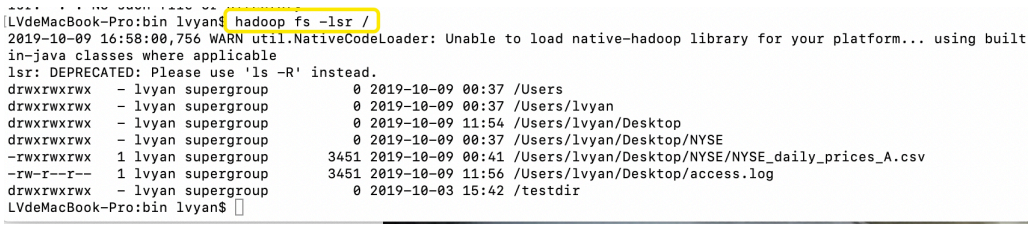
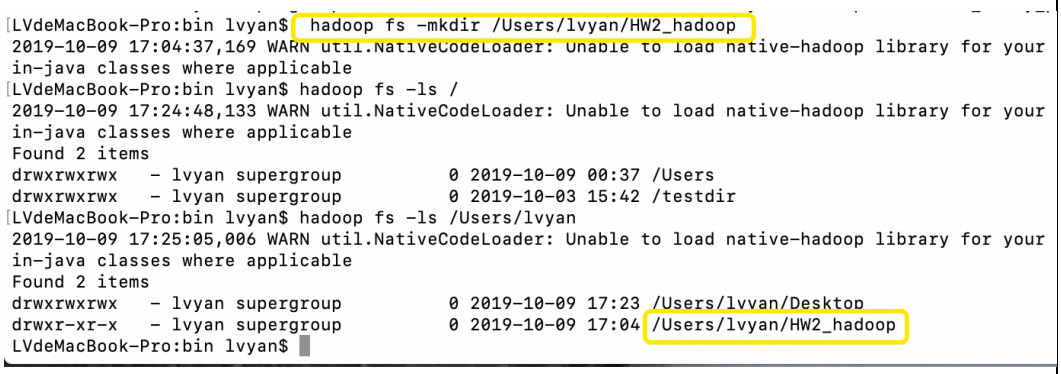
copyToLocal copy test.csv to lcoal file system	LVdeMacBook-Pro:bin lvyan$ hadoop fs -copyToLocal /Users/lvyan/HW2_hadoop/test.csv /Users/lvyan/Desktop 2019-10-09 19:13:17,214 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platfor sses where applicable 2019-10-09 19:13:18,023 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false LVdeMacBook-Pro:bin lvyan$ 
Count Count the numebr of directions and files	LVdeMacBook-Pro:bin lvyan$ hadoop fs -count / 2019-10-09 19:14:51,226 WARN util.NativeCodeLoader: Unable to load native-h sses where applicable 5 2 81982114 / LVdeMacBook-Pro:bin lvyan$ hadoop fs -count /Users/lvyan/HW2_hadoop 2019-10-09 19:15:07,759 WARN util.NativeCodeLoader: Unable to load native-h sses where applicable 1 2 81982114 /Users/lvyan/HW2_hadoop LVdeMacBook-Pro:bin lvyan$
Cp Copy a file named test.csv in HW2_hadoop to testdir	LVdeMacBook-Pro:bin lvyan$ hadoop fs -ls /testdir 2019-10-09 19:17:49,768 WARN util.NativeCodeLoader: Unable to load native-hadoop libr sses where applicable LVdeMacBook-Pro:bin lvyan$ hadoop fs -cp /Users/lvyan/HW2_hadoop/test.csv /testdir 2019-10-09 19:18:09,130 WARN util.NativeCodeLoader: Unable to load native-hadoop libr sses where applicable 2019-10-09 19:18:09,894 INFO sasl.SaslDataTransferClient: SASL encryption trust check false 2019-10-09 19:18:10,004 INFO sasl.SaslDataTransferClient: SASL encryption trust check false LVdeMacBook-Pro:bin lvyan$ hadoop fs -ls /testdir 2019-10-09 19:18:14,474 WARN util.NativeCodeLoader: Unable to load native-hadoop libr sses where applicable Found 1 items -rw-r--r-- 1 lvyan supergroup 40991122 2019-10-09 19:18 /testdir/test.csv LVdeMacBook-Pro:bin lvyan$

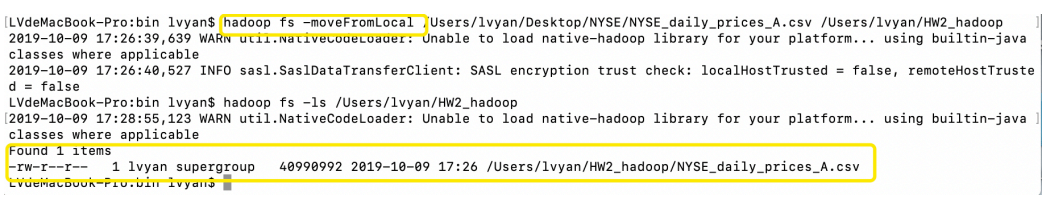
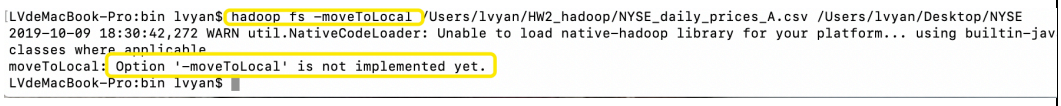
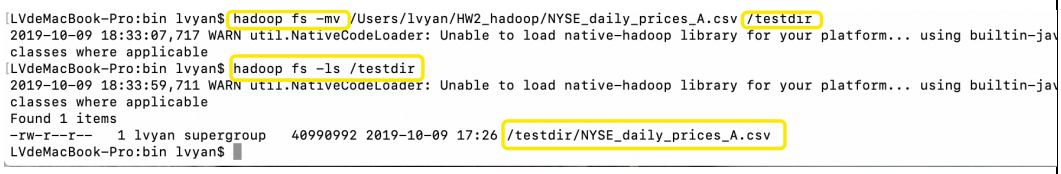
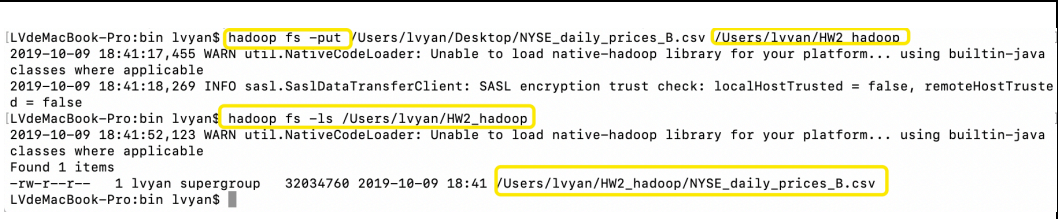
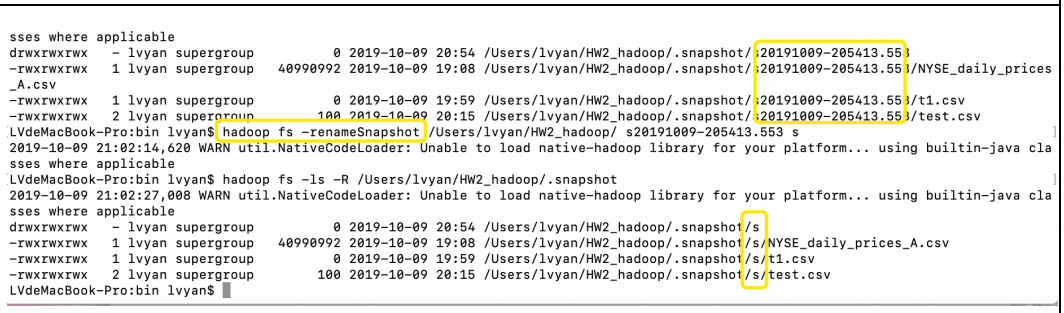
createSnapshot create a snapshot for files in HW2_hadoop	
deleteSnapshot delete snapshot in HW2_hadoop	
Df Show df in hdfs	
Du Show du in HW2_hadoop	
Dus Show dus in HW2_hadoop	

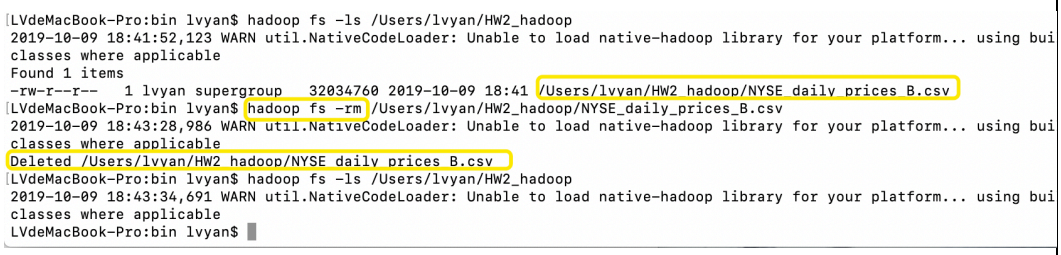
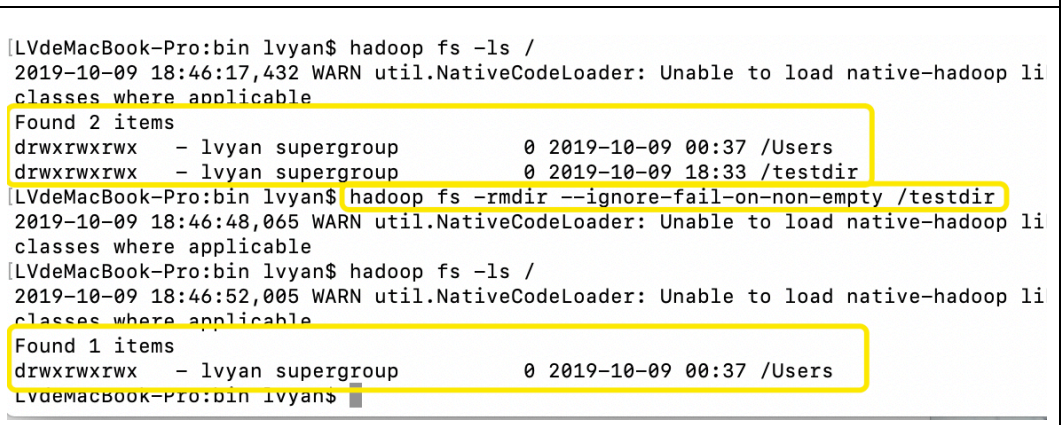
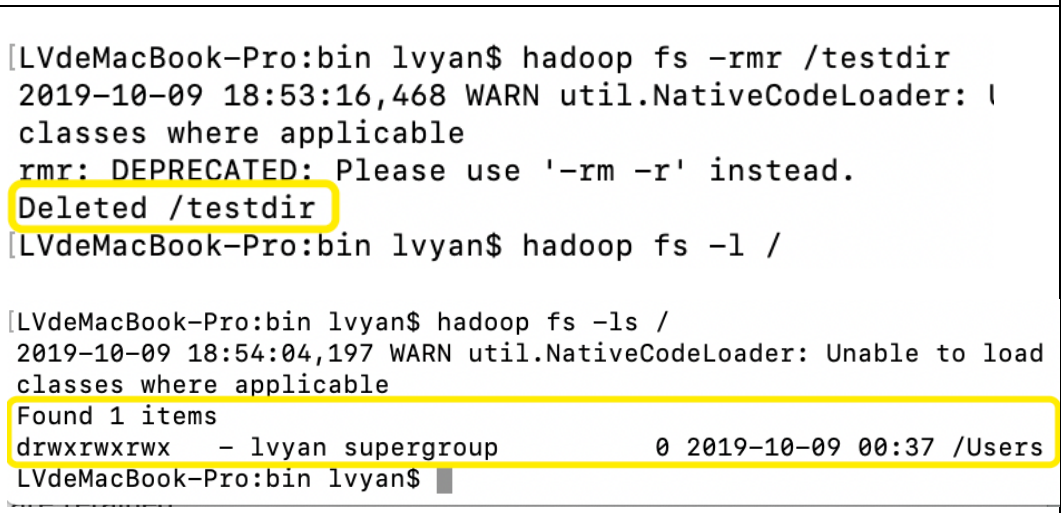
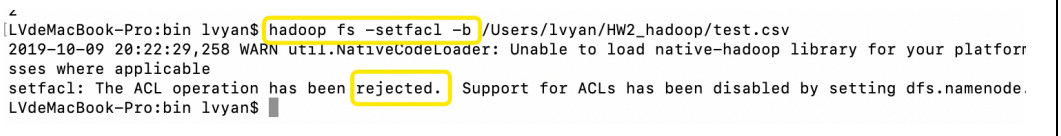
Expunge clean	LVdeMacBook-Pro:bin lvyan$ hadoop fs -expunge 2019-10-09 19:25:10,199 WARN util.NativeCodeLoa sses where applicable LVdeMacBook-Pro:bin lvyan$
Find Find files named test.csv	LVdeMacBook-Pro:bin lvyan$ hadoop fs -find / -iname test.csv -print 2019-10-09 19:28:59,138 WARN util.NativeCodeLoader: Unable to load native- sses where applicable /Users/lvyan/HW2_hadoop/test.csv /testdir/test.csv LVdeMacBook-Pro:bin lvyan$
Get Copy a file to local file system named test.csv	LVdeMacBook-Pro:bin lvyan$ hadoop fs -get /Users/lvyan/HW2_hadoop/test.csv /Users/lvyan/Desktop 2019-10-09 19:30:12,804 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your sses where applicable 2019-10-09 19:30:13,619 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTr false LVdeMacBook-Pro:bin lvyan$ test.csv 41 MB Modified: Today, 7:30 PM Add Tags... General: Kind: CSV Document Size: 40,991,122 bytes (41 MB on disk) Where: Macintosh HD ▸ Users ▸ lvyan ▸ Desktop Created: October 9, 2019 at 7:30 PM Modified: October 9, 2019 at 7:30 PM Stationery pad Locked
Getfacl	LVdeMacBook-Pro:bin lvyan$ hadoop fs -getfacl /Users/lvyan/HW2_hadoop/test. 2019-10-09 19:31:24,741 WARN util.NativeCodeLoader: Unable to load native-h sses where applicable # file: /Users/lvyan/HW2_hadoop/test.csv # owner: lvyan # group: supergroup user::rw- group::r-- other::r-- LVdeMacBook-Pro:bin lvyan$
getfattr	LVdeMacBook-Pro:bin lvyan$ hadoop fs -getfattr -d /Users/lvyan/HW2_hadoop/test.csv 2019-10-09 20:17:43,358 WARN util.NativeCodeLoader: Unable to load native-hadoop libra sses where applicable # file: /Users/lvyan/HW2_hadoop/test.csv LVdeMacBook-Pro:bin lvyan$

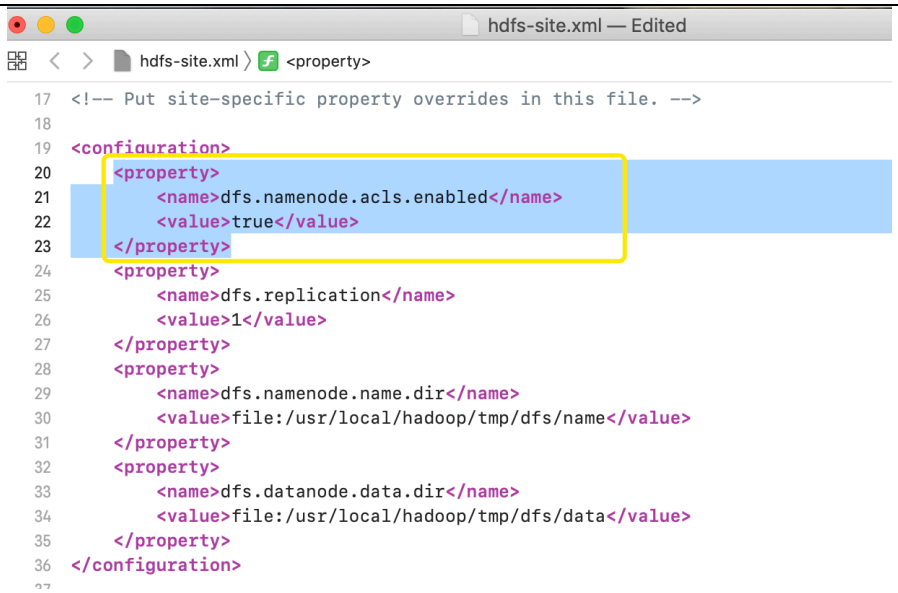
	LVdeMacBook-Pro:bin lvyan$ hadoop fs -getfattr -d /Users/lvyan/HW2_hadoop/test.csv
	2019-10-09 20:19:32,657 WARN util.NativeCodeLoader: Unable to load native-hadoop library
	sses where applicable
	# file: /Users/lvyan/HW2_hadoop/test.csv
	user.web="www.google.com"
	LVdeMacBook-Pro:bin lvyan$

Getmerge Merge the files in HW2_hadoop to a local file named localfile	LVdeMacBook-Pro:bin lvyan$ hadoop fs -getmerge /Users/lvyan/HW2_hadoop /Users/lvyan/Desktop/localfile 2019-10-09 19:37:55,620 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platfc sses where applicable 2019-10-09 19:37:56,392 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted false LVdeMacBook-Pro:bin lvyan$ localfile exchange,stock_symbol,date,stock_price_open,stock_price_high,stock_price_low,stock_price_c lose,stock_volume,stock_price_adj_close NYSE,AEA,2010-02-08,4.42,4.42,4.21,4.24,205500,4.24 NYSE,AEA,2010-02-05,4.42,4.54,4.22,4.41,194300,4.41 NYSE,AEA,2010-02-04,4.55,4.69,4.39,4.42,233800,4.42 NYSE,AEA,2010-02-03,4.65,4.69,4.50,4.55,182100,4.55 NYSE,AEA,2010-02-02,4.74,5.00,4.62,4.66,222700,4.66 NYSE,AEA,2010-02-01,4.84,4.92,4.68,4.75,194800,4.75 NYSE,AEA,2010-01-29,4.97,5.05,4.76,4.83,222900,4.83 NYSE,AEA,2010-01-28,5.12,5.22,4.81,4.98,283100,4.98 NYSE,AEA,2010-01-27,4.82,5.16,4.79,5.09,243500,5.09 NYSE,AEA,2010-01-26,5.18,5.18,4.81,4.84,554800,4.84 NYSE,AEA,2010-01-25,5.42,5.48,5.20,5.22,257300,5.22 NYSE,AEA,2010-01-22,5.52,5.59,5.31,5.37,260800,5.37 NYSE,AEA,2010-01-21,5.67,5.74,5.37,5.51,264300,5.51
Head The the head for a file named test.csv	LVdeMacBook-Pro:bin lvyan$ hadoop fs -head /Users/lvyan/HW2_hadoop/test.csv 2019-10-09 19:39:39,384 WARN util.NativeCodeLoader: Unable to load native-hadoc sses where applicable 2019-10-09 19:39:40,121 INFO sasl.SaslDataTransferClient: SASL encryption trust false exchange,stock_symbol,date,stock_price_open,stock_price_high,stock_price_low,st exchange,stock_symbol,date,stock_price_open,stock_price_high,stock_price_low,st NYSE,AEA,2010-02-08,4.42,4.42,4.21,4.24,205500,4.24 NYSE,AEA,2010-02-05,4.42,4.54,4.22,4.41,194300,4.41 NYSE,AEA,2010-02-04,4.55,4.69,4.39,4.42,233800,4.42 NYSE,AEA,2010-02-03,4.65,4.69,4.50,4.55,182100,4.55 NYSE,AEA,2010-02-02,4.74,5.00,4.62,4.66,222700,4.66 NYSE,AEA,2010-02-01,4.84,4.92,4.68,4.75,194800,4.75 NYSE,AEA,2010-01-29,4.97,5.05,4.76,4.83,222900,4.83 NYSE,AEA,2010-01-28,5.12,5.22,4.81,4.98,283100,4.98 NYSE,AEA,2010-01-27,4.82,5.16,4.79,5.09,243500,5.09 NYSE,AEA,2010-01-26,5.18,5.18,4.81,4.84,554800,4.84 NYSE,AEA,2010-01-25,5.42,5.48,5.20,5.22,257300,5.22 NYSE,AEA,2010-01-22,5.52,5.59,5.31,5.37,260800,5.37 NYSE,AEA,2010-01-21,5.67,5.74,5.37,5.51,264300,5.51 NYSE,AEA,2010-01-20,5.65,5.70,5.53,5.66,244600,5.66 NYSE,AEA,2010-01-19,5.54,5.70,5.54,5LVdeMacBook-Pro:bin lvyan$

Help	
Ls List the directions and files	
Lsr	
Mkdir Make a directory	

moveFromLocal move a local file named NSYE_daily_price_A.csv to hdfs direction named HW2_hadoop	
moveToLocal move NSYE_daily_price_A.csv to local file system	
Mv Move a file named NSYE_daily_price_A.csv in HW2_hadoop to another directory named testdir	
Put Upload a local file to hdfs	
renameSnapshot	

Rm Remove a file	
Rmdir Remove a directory	
Rmr	
setfacl	

	hdfs-site.xml — Edited hdfs-site.xml 〉 <property>  LVdeMacBook-Pro:bin lvyan$ hadoop fs -setfacl -b /Users/lvyan/HW2_hadoop/test.csv 2019-10-10 15:37:22,357 WARN util.NativeCodeLoader: Unable to load native-hadoop libra for your platform... using builtin-java classes where applicable LVdeMacBook-Pro:bin lvyan$

Setfattr	LVdeMacBook-Pro:bin lvyan$ hadoop fs -getfattr -d /Users/lvyan/HW2_hadoop/test.csv
	2019-10-09 20:17:43,358 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
	sses where applicable
	# file: /Users/lvyan/HW2_hadoop/test.csv
	LVdeMacBook-Pro:bin lvyan$ hadoop fs -setfattr -n user.web -v www.google.com /Users/lvy
	2019-10-09 20:19:27,808 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
	sses where applicable
	LVdeMacBook-Pro:bin lvyan$ hadoop fs -getfattr -d /Users/lvyan/HW2_hadoop/test.csv
	2019-10-09 20:19:32,657 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
	sses where applicable
	# file: /Users/lvyan/HW2_hadoop/test.csv
	user.web="www.google.com"
	LVdeMacBook-Pro:bin lvyan$

Setrep	LVdeMacBook-Pro:bin lvyan$ hadoop fs -stat %r /Users/lvyan/HW2_hadoop/test.csv 2019-10-09 20:20:47,390 WARN util.NativeCodeLoader: Unable to load native-hadoop sses where applicable 1 LVdeMacBook-Pro:bin lvyan$ hadoop fs -setrep 2 /Users/lvyan/HW2_hadoop/test.csv 2019-10-09 20:21:01,358 WARN util.NativeCodeLoader: Unable to load native-hadoop sses where applicable Replication 2 set: /Users/lvyan/HW2_hadoop/test.csv LVdeMacBook-Pro:bin lvyan$ hadoop fs -stat %r /Users/lvyan/HW2_hadoop/test.csv 2019-10-09 20:21:06,495 WARN util.NativeCodeLoader: Unable to load native-hadoop sses where applicable 2 LVdeMacBook-Pro:bin lvyan$

Stat	LVdeMacBook-Pro:bin lvyan$ hadoop fs -stat "type:%F perm:%a %u:%g size:%b mtime:%y atime:%x name:%n" / 2019-10-09 19:51:46,981 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platfo sses where applicable type:directory perm:777 lvyan:supergroup size:0 mtime:2019-10-09 23:16:52 atime:1970-01-01 00:00:00 name LVdeMacBook-Pro:bin lvyan$
Tail Get the tail of test.csv	LVdeMacBook-Pro:bin lvyan$ hadoop fs -tail /Users/lvyan/HW2_hadoop/test.csv 2019-10-09 19:52:52,048 WARN util.NativeCodeLoader: Unable to load native-hado sses where applicable 2019-10-09 19:52:52,761 INFO sasl.SaslDataTransferClient: SASL encryption trus false .10.451000.8.39 NYSE,AYI,2001-12-28,12.91,12.91,11.90,12.00,324500,8.32 NYSE,AYI,2001-12-27,12.70,12.95,12.70,12.90,374500,8.94 NYSE,AYI,2001-12-26,13.09,13.20,11.96,12.51,255900,8.67 NYSE,AYI,2001-12-24,12.35,12.95,12.10,12.89,112300,8.93 NYSE,AYI,2001-12-21,11.20,11.95,11.15,11.90,367700,8.25 NYSE,AYI,2001-12-20,12.00,12.00,11.35,11.51,535400,7.98 NYSE,AYI,2001-12-19,11.16,12.10,11.05,11.95,290400,8.28 NYSE,AYI,2001-12-18,10.90,11.50,10.70,11.50,685500,7.97 NYSE,AYI,2001-12-17,11.65,12.10,10.80,11.00,523600,7.62 NYSE,AYI,2001-12-14,11.90,12.08,11.76,11.98,146000,8.30 NYSE,AYI,2001-12-13,12.10,12.40,11.75,12.00,248600,8.32 NYSE,AYI,2001-12-12,12.48,12.85,11.95,12.00,226700,8.32 NYSE,AYI,2001-12-11,11.30,12.58,11.30,12.58,595400,8.72 NYSE,AYI,2001-12-10,13.10,13.10,11.20,11.20,348800,7.76 NYSE,AYI,2001-12-07,13.88,13.88,13.40,13.48,399500,9.34 NYSE,AYI,2001-12-06,13.95,14.20,13.70,13.98,291300,9.69 NYSE,AYI,2001-12-05,13.75,14.20,13.75,14.06,906200,9.74 NYSE,AYI,2001-12-04,13.70,13.84,13.60,13.75,780700,9.53 LVdeMacBook-Pro:bin lvyan$
Test	brary for your platform... using builtin-java classes where applicable LVdeMacBook-Pro:bin lvyan$ hadoop fs -test -e /Users/lvyan/HW2_hadoop/test.cs 2019-10-10 15:39:23,999 WARN util.NativeCodeLoader: Unable to load native-had brary for your platform... using builtin-java classes where applicable LVdeMacBook-Pro:bin lvyan$ hadoop fs -test -f /Users/lvyan/HW2_hadoop/test.cs 2019-10-10 15:39:40,133 WARN util.NativeCodeLoader: Unable to load native-had brary for your platform... using builtin-java classes where applicable LVdeMacBook-Pro:bin lvyan$ hadoop fs -test -z /Users/lvyan/HW2_hadoop/test.cs 2019-10-10 15:39:47,337 WARN util.NativeCodeLoader: Unable to load native-had brary for your platform... using builtin-java classes where applicable LVdeMacBook-Pro:bin lvyan$
Text Get the contetn of test.csv	LVdeMacBook-Pro:bin lvyan$ hadoop fs -text /Users/lvyan/HW2_hadoop/test.csv 2019-10-09 19:55:44,379 WARN util.NativeCodeLoader: Unable to load native-hadoo sses where applicable 2019-10-09 19:55:45,143 INFO sasl.SaslDataTransferClient: SASL encryption trust false 2019-10-09 19:55:45,226 INFO sasl.SaslDataTransferClient: SASL encryption trust false exchange,stock_symbol,date,stock_price_open,stock_price_high,stock_price_low,st exchange,stock_symbol,date,stock_price_open,stock_price_high,stock_price_low,st NYSE,AEA,2010-02-08,4.42,4.42,4.21,4.24,205500,4.24 NYSE,AEA,2010-02-05,4.42,4.54,4.22,4.41,194300,4.41 NYSE,AEA,2010-02-04,4.55,4.69,4.39,4.42,233800,4.42 NYSE,AEA,2010-02-03,4.65,4.69,4.50,4.55,182100,4.55 NYSE,AEA,2010-02-02,4.74,5.00,4.62,4.66,222700,4.66

Touch	sses where applicable LVdeMacBook-Pro:bin lvyan$ hadoop fs -touch /Users/lvyan/HW2_hadoop/test.csv 2019-10-09 21:17:36,493 WARN util.NativeCodeLoader: Unable to load native-hadoop library f sses where applicable LVdeMacBook-Pro:bin lvyan$
Touchz Created an empty file named t1.csv	-rw-r--r-- 1 lvyan GROUP 40990992 2019-10-09 19:08 /Users/lvyan/HW2_hadoop/NYSE_daily_prices_A.csv -rw-r--r-- 1 lvyan GROUP 0 2019-10-09 19:57 /Users/lvyan/HW2_hadoop/t.csv -rw-r--r-- 1 lvyan supergroup 40991122 2019-10-09 19:57 /Users/lvyan/HW2_hadoop/test.csv LVdeMacBook-Pro:bin lvyan$ hadoop fs -touchz /Users/lvyan/HW2_hadoop/t1.csv 2019-10-09 19:59:38,406 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... sses where applicable LVdeMacBook-Pro:bin lvyan$ hadoop fs -ls /Users/lvyan/HW2_hadoop 2019-10-09 19:59:42,809 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... sses where applicable Found 4 items -rw-r--r-- 1 lvyan GROUP 40990992 2019-10-09 19:08 /Users/lvyan/HW2_hadoop/NYSE_daily_prices_A.csv -rw-r--r-- 1 lvyan GROUP 0 2019-10-09 19:57 /Users/lvyan/HW2_hadoop/t.csv -rw-r--r-- 1 lvyan GROUP 0 2019-10-09 19:59 /Users/lvyan/HW2_hadoop/t1.csv -rw-r--r-- 1 lvyan supergroup 40991122 2019-10-09 19:57 /Users/lvyan/HW2_hadoop/test.csv LVdeMacBook-Pro:bin lvyan$
Truncate Truncate test.csv to size of 100	LVdeMacBook-Pro:bin lvyan$ hadoop fs -truncate 100 /Users/lvyan/HW2_hadoop/test.csv 2019-10-09 20:12:55,214 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cla sses where applicable Truncating /Users/lvyan/HW2_hadoop/test.csv to length: 100. Wait for block recovery to complete before further updating this file. LVdeMacBook-Pro:bin lvyan$ File information - test.csv Download Head the file (first 32K) Tail the file (last 32K) Block information -- Block 0 Block ID: 1073741831 Block Pool ID: BP-491383355-10.110.39.130-1570130386611 Generation Stamp: 1012 Size: 100 Availability: • 10.110.39.130
usage	LVdeMacBook-Pro:bin lvyan$ hadoop fs -usage test 2019-10-10 15:44:13,321 WARN util.NativeCodeLoader: Unable to load native-hadoo brary for your platform... using builtin-java classes where applicable Usage: hadoop fs [generic options] -test -[defswrz] <path> LVdeMacBook-Pro:bin lvyan$

BEFORE PROGRAMMING

In this homework, almost all the input file are csv file or text file, based on this, I defined all the Input Format as LongWritable in mapper class. That's because LongWritable support the raeding of csv file and txt file. Except the SequenceFileInputFormat.

When Mapper, line is split and generate (key, value) pair. According to different purpose, I split each line by blank or ",". As for value, if I need to count, it will be 1, if I want to find the max value, it will be the value of some attributes. So, the second attribute when I extend Mapper is IntWritable. For preparation for Reducer, the third and the fourth attribute when I extend Mapper is Text and IntWritable respectively.

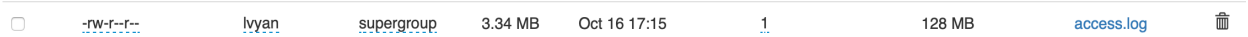
For Reducer, I just get the specific result for each key. If I need sum, I will add all the value for each key; If I need max value, I will get the max value for each key. So, the first and second attribute when I extends Reducer is Text and IntWritable respectively, which is match the output data format of Mapper, and the third and fourth attribute is Tetx and IntWritable respectively according the results I want to get.

But for Part_4, as I want to find the max price, which is a float data type. So I replace IntWritable with DoubleWritable.

In Driver class, I initiate configuration and job. Then, I set driver class, mapper class, reducer class, input format class, output format class, output key class, output value class, input path and output path for job.

PART 3 – PROGRAMMING ASSIGNMENT

Step 1: Upload access.log file to hdfs, and this file is under /logs directory.



Step 2: Implement Mapper class. According to the format of each line, I split line by blanks. The first token in each line is ip address which is key for Map Reduce. When one ip address access network once, the value will be marked as 1.

```
public class AccessMapper extends Mapper<LongWritable, Text, Text, IntWritable>{
    Text word = new Text();
    IntWritable one = new IntWritable(1);

    //number of times each IP accesses the website
    public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
        String line = value.toString();
        String[] tokens = line.split("\\s+");
        String ipAddress = tokens[0];
        word.set(ipAddress);
        context.write(word, one);
    }
}
```

Step 3 : Implement Reducer class. Get the sum of value for each key.

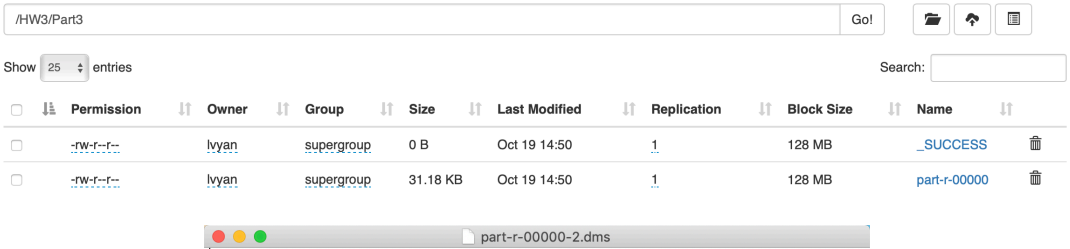
```
public class AccessReducer extends Reducer<Text, IntWritable, Text, IntWritable>{

    @Override
    protected void reduce(Text key, Iterable<IntWritable> values,
            Reducer<Text, IntWritable, Text, IntWritable>.Context context) throws IOException, InterruptedException {
        // TODO Auto-generated method stub
        int sum = 0;
        for(IntWritable val : values) {
            sum += val.get();
        }
        IntWritable count = new IntWritable(sum);
        context.write(key, count);
    }
}
```

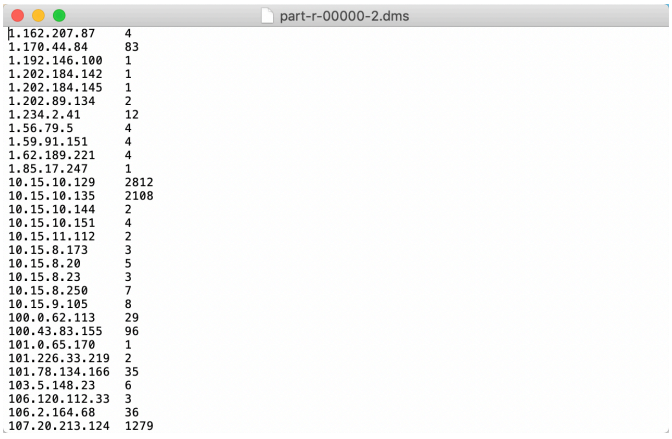
Step 4: Implement Driver class. I will get input path and output path from command line.

```java
public static void main( String[] args ){
      System.out.println( "Hello World!" );
    Configuration conf = new Configuration();

    //create a new job
    Job job;
    try {
        job = Job.getInstance(conf, "Access count Part3");

        //set the driver class
        //a class contains the name of main method
        job.setJarByClass(App.class);

        //set mapper and reducer classes
        job.setMapperClass(AccessMapper.class);
        job.setReducerClass(AccessReducer.class);

        //set inputformat and output format class
        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        //set the output key and value type
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        //set the input and output path
        //path will come from the command line
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ClassNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (InterruptedException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
```

Step 5: Run jar and get results. Input path is /logs/access.log , output path is /HW3/Part3. Both of these two files are stored in HDFS.

hadoop jar /Users/lvyan/Desktop/hw3.jar HW3.Part3.App /logs/access.log /HW3/Part3





PART 4 – PROGRAMMING ASSIGNMENT

PART 4.1

Step 1: Implement the Mapper class, it will map all the files in one directory. For the format of each file, we can know the first line is header, so I skip it. Because the input file is csv files, I split each line by ",". Our purpose is to find the max stock_price_high for each stock, so I set stock_symbol as a key and set stock_price_high as a value.

```
//Max price of stock_price_high for each stock
public class MaxMapper extends Mapper<LongWritable, Text, Text, DoubleWritable>{
    Text word = new Text();
    DoubleWritable max = new DoubleWritable();

    public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
        if(key.get() == 0) {
            return;
        }
        String line = value.toString();
        String[] tokens = line.split(",");
        word.set(tokens[1]);
        max.set(Double.parseDouble(tokens[4]));
        context.write(word, max);
    }
}
```

Step 2: Implement Reducer class, it will find the max stock_price_high for each stock_symbol.

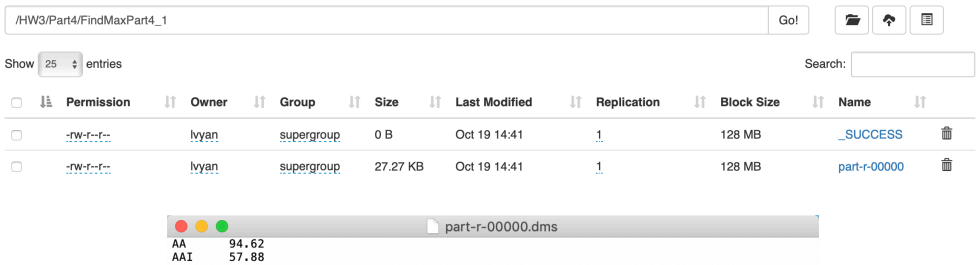
```
//Max price of stock_price_high for each stock
public class MaxReducer extends Reducer<Text, DoubleWritable, Text, DoubleWritable>{

    @Override
    protected void reduce(Text key, Iterable<DoubleWritable> values,
            Reducer<Text, DoubleWritable, Text, DoubleWritable>.Context context) throws IOException, InterruptedException {
        // TODO Auto-generated method stub
        double max = 0;
        for(DoubleWritable val : values) {
            max = Math.max(max, val.get());
        }
        DoubleWritable MAX = new DoubleWritable(max);
        context.write(key, MAX);
    }
}
```

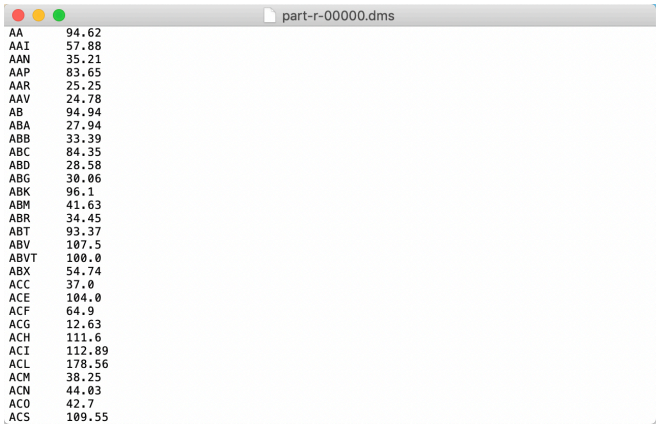
Step 3: Implement Driver class. It will get input path and output path from command line. I will choose a directory as input path; Hadoop can recognize and process all the file in this directory.

```java
public class Part4App {
    public static void main( String[] args ){
        long beginTime = System.currentTimeMillis();
        System.out.println( "BEGIN TIME: " +beginTime);

        //create the configuration, it will used for job configuration
        Configuration conf = new Configuration();

        //create a new job according to the configuraiton file
        try {
            Job job = Job.getInstance(conf, "Find the max stock price");

            //set driver class
            job.setJarByClass(Part4App.class);

            //set mapper and reducer
            job.setMapperClass(MaxMapper.class);
            job.setReducerClass(MaxReducer.class);

            //set inputFormat ans outputFormat class
            job.setInputFormatClass(TextInputFormat.class);
            job.setOutputFormatClass(TextOutputFormat.class);

            //set the output key ans value type
            job.setOutputKeyClass(Text.class);
            job.setOutputValueClass(DoubleWritable.class);

            //set the inout and output path
            FileInputFormat.addInputPath(job, new Path(args[0]));
            FileOutputFormat.setOutputPath(job, new Path(args[1]));

            long endTime = System.currentTimeMillis();
            System.out.println( "END TIME: " +endTime);

            long seconds = (endTime - beginTime) / 1000;
            System.out.println( "TOTAL TIME(secondes): " +seconds);

            try {
                System.exit(job.waitForCompletion(true) ? 0 : 1);
            } catch (ClassNotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        } catch (IOException e) {
```

Step 4 : Run jar file and get results. Input directory is /NYSE, output path is /HW3/Part4/FindMaxPart4_1.

hadoop jar /Users/lvyan/Desktop/hw3.jar HW3.Par4_1.Part4App /NYSE /HW3/Part4/FindMaxPart4_1





Total Running time: 1’27”

PART 4.2

Step 1: Merge all the files in this dictionary. Skip the first line, read content in this file line by line and write each line into context. An important thing is, I must add "\n" after each line.

```
public class PutMerge {

    public static void main(String[] args) throws IOException {

        Configuration conf = new Configuration();

        FileSystem hdfs = FileSystem.get(conf);
        FileSystem local = FileSystem.getLocal(conf);

        Path localPath = new Path(args[0]);
        Path hdfsPath = new Path(args[1]);

        FileStatus[] files = local.listStatus(localPath);
        FSDataOutputStream output = hdfs.create(hdfsPath);
        FSDataInputStream input = null;
        Reader reader = null;
        BufferedReader bufferedReader = null;
        Writer writer = new OutputStreamWriter(output);;
        BufferedWriter bufferedWriter = new BufferedWriter(writer);

        //read each file in the directory
        for(FileStatus file : files) {
            if(!file.getPath().getName().contains("NYSE_daily_prices_"))
                continue;
            input = local.open(file.getPath());
            reader = new InputStreamReader(input);
            bufferedReader = new BufferedReader(reader);
            String line = null;
            bufferedReader.readLine();
            while((line = bufferedReader.readLine()) != null) {
                byte[] bytes = line.getBytes();
                bufferedWriter.write(new String(bytes)+"\n");
            }
        }

        //close all the stream
        bufferedWriter.close();
        writer.close();
        output.close();
        bufferedReader.close();
        reader.close();
        input.close();
        hdfs.close();

    }
}
```

Step 2: implement Mapper class. Get the symbol_stock and stock_price_high in each line. Same as Part4.1.

```
//Max price of stock_price_high for each stock
public class MaxMapper extends Mapper<LongWritable, Text, Text, DoubleWritable>{
    Text word = new Text();
    DoubleWritable max = new DoubleWritable();

    public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
        String line = value.toString();
        String[] tokens = line.split(",");
        word.set(tokens[1]);
        max.set(Double.parseDouble(tokens[4]));
        context.write(word, max);
    }
}
```

Step 3: Implement Reducer class. Get the max value with the same key. Same as Part4.1.

```
//Max price of stock_price_high for each stock
public class MaxReducer extends Reducer<Text, DoubleWritable, Text, DoubleWritable>{

    @Override
    protected void reduce(Text key, Iterable<DoubleWritable> values,
            Reducer<Text, DoubleWritable, Text, DoubleWritable>.Context context) throws IOException, InterruptedException {
        // TODO Auto-generated method stub
        double max = 0;
        for(DoubleWritable val : values) {
            max = Math.max(max, val.get());
        }
        DoubleWritable MAX = new DoubleWritable(max);
        context.write(key, MAX);
    }

}
```

Step 4: Implement Driver class. Get input path and output path from command line. Same as Part4.1.

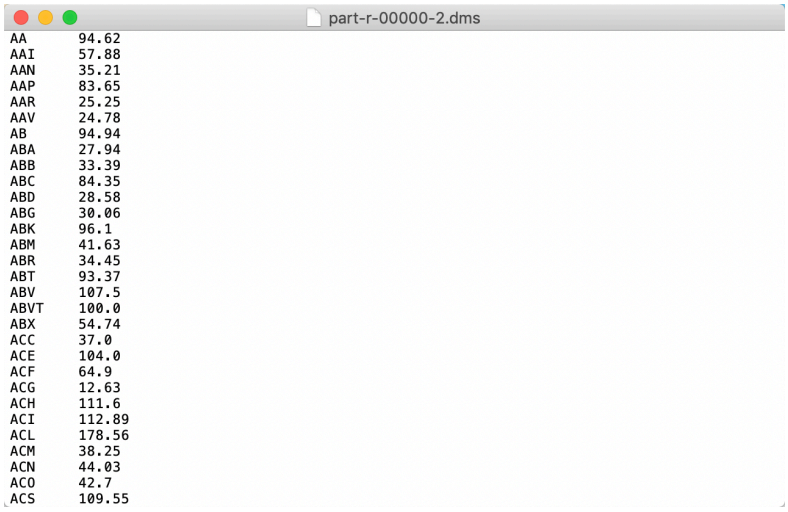
```java
public class Part4App {
    public static void main( String[] args ){
        long beginTime = System.currentTimeMillis();
        System.out.println( "BEGIN TIME: " +beginTime);

        //create the configuration, it will used for job configuration
        Configuration conf = new Configuration();

        //create a new job according to the configuraiton file
        try {
            Job job = Job.getInstance(conf, "Find the max stock price");

            //set driver class
            job.setJarByClass(Part4App.class);

            //set mapper and reducer
            job.setMapperClass(MaxMapper.class);
            job.setReducerClass(MaxReducer.class);

            //set inputFormat ans outputFormat class
            job.setInputFormatClass(TextInputFormat.class);
            job.setOutputFormatClass(TextOutputFormat.class);

            //set the output key ans value type
            job.setOutputKeyClass(Text.class);
            job.setOutputValueClass(DoubleWritable.class);

            //set the inout and output path
            FileInputFormat.addInputPath(job, new Path(args[0]));
            FileOutputFormat.setOutputPath(job, new Path(args[1]));

            long endTime = System.currentTimeMillis();
            System.out.println( "END TIME: " +endTime);

            long seconds = (endTime - beginTime) / 1000;
            System.out.println( "TOTAL TIME(secondes): " +seconds);

            try {
                System.exit(job.waitForCompletion(true) ? 0 : 1);
            } catch (ClassNotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
```

I get a singleFile:

Step 5: Run jar file and get results. Input path is /HW3/Part4/singleFile , output path is /HW3/Part4/FindMaxPart4_2.

Obviously, the results are the same as the previous part. Running time is much less than the previous one. Total Running time: 52'' Which is 33'' faster than the previous one.

hadoop jar /Users/lvyan/Desktop/hw3.jar HW3.Part4_2.Part4App /HW3/Part4/singleFile /HW3/Part4/FindMaxPart4_2



PART 5 – PROGRAMMING ASSIGNMENT

COMBINEFILEINPUTFORMAT

An abstract class, the getSplits(JobContext) of this class returns not List<FileSplit> but List<CombineFileSplit>. CombineFileSplit is constructed from files in the input path. The object cannot have files in different pools. Each CombineFileSplit may contain blocks of different files.

Step 1: Implement FileLineWritable which implements WritableComparable. This is a customized format. In this class, defined some basic operation of input file format. This project will read file line by line in the future.

```java
public class FileLineWritable implements WritableComparable<FileLineWritable>{

    public long offset;
    public String fileName;
    public void readFields(DataInput input) throws IOException {
        // TODO Auto-generated method stub
        this.offset = input.readLong();
        this.fileName = Text.readString(input);

    }

    public void write(DataOutput output) throws IOException {
        // TODO Auto-generated method stub
        output.writeLong(offset);
        Text.writeString(output, fileName);
    }

    public int compareTo(FileLineWritable b) {
        // TODO Auto-generated method stub
        int n = this.fileName.compareTo(b.fileName);
        if(n != 0)
            return n;
        return (int)Math.signum((double)(this.offset - b.offset));
    }
```

Step 2: Implement CombineRecordReader which extends RecordReader<FileLineWritable, Text>. And implement all the unimplemented methods. These are, getCurrentKey(), getCurrentValue(), getProgress(), nextKeyValue(). It is very similar to LineRecordReader.

```java
public class CombineRecordReader extends RecordReader<FileLineWritable, Text>{

    private LineRecordReader lrr = new LineRecordReader();
    private long startOffset;
    private long end;
    private long pos;
    private FileSystem fs;
    private Path path;
    private FileLineWritable key;
    private Text value;
    private FSDataInputStream input;
    private LineReader reader;

    public CombineRecordReader(CombineFileSplit split, TaskAttemptContext context, Integer index) throws IOException {
        this.path = split.getPath(index);
        Configuration conf = context.getConfiguration();
        fs = this.path.getFileSystem(conf);
        this.startOffset = split.getOffset(index);
        this.end = startOffset + split.getLength(index);
        input = fs.open(path);
        reader = new LineReader(input);
        this.pos = startOffset;
    }
```

Step 3: Implemengt CombineSmallfileInputForamt which extends CombineFileInputFormat<FileLineWritable, Text>. In this class, set the max split size and RecordReader which I implemented in the previous step. The most importance method is createRecordReader, it defined which RecordReader we will use in the next step.

```java
//CombineSmallfileInputFormat extends CombineFileInputFormat
//it creates subclass to support the specific input

//CombineSmallfileRecordReader is a customer RecordReader。
public class CombineSmallfileInputFormat extends CombineFileInputFormat<FileLineWritable, Text>{

    public CombineSmallfileInputFormat() {
        super();
        setMaxSplitSize(67108864); // 64MB
    }

    protected boolean isSplitable(JobContext context, Path file){
          return false;
    }

    @Override
    public RecordReader<FileLineWritable, Text> createRecordReader(InputSplit split, TaskAttemptContext context)
            throws IOException {
        // TODO Auto-generated method stub
        return new CombineFileRecordReader<FileLineWritable,Text> ((CombineFileSplit)split, context, CombineRecordReader.class);
    }

}
```

Step 4: Implement Mapper class. This program will split the line by blank, count the occurrence of each key.

```
public class CombineMapper extends Mapper<FileLineWritable, Text, Text, IntWritable>{
    private IntWritable one = new IntWritable(1);

    @Override
    protected void map(FileLineWritable key, Text value,
            Mapper<FileLineWritable, Text, Text, IntWritable>.Context context)
            throws IOException, InterruptedException {
        // TODO Auto-generated method stub
        String[] tokens = value.toString().split("\\s+");
        for(String token : tokens) {
            context.write(new Text(token), one);
        }

    }

}
```

Step 5: Implement Reducer class. Get the total occurrence of each word.

```
public class CombineReducer extends Reducer<Text, IntWritable, Text, IntWritable>{

    @Override
    protected void reduce(Text key, Iterable<IntWritable> values,
            Reducer<Text, IntWritable, Text, IntWritable>.Context context) throws IOException, InterruptedException {
        // TODO Auto-generated method stub
        int sum = 0;
        for(IntWritable value : values) {
            sum += value.get();
        }
        context.write(key, new IntWritable(sum));
    }

}
```

Step 6: Implement driver class. Defined the input format as CombineSmallfileInputFormat.

```
public class CombineApp {

    public static void main(String[] args) throws IOException, ClassNotFoundException, InterruptedException {
        Configuration conf = new Configuration();

        Job job = Job.getInstance(conf, "Combination File Input Format");

        //set driver class
        job.setJarByClass(CombineApp.class);

        //set mapper and reducer
        job.setMapperClass(CombineMapper.class);
        job.setReducerClass(CombineReducer.class);

        //add input path and output path
//      Path input = new Path("/Users/lvyan/Desktop/NYSE");
//      Path output = new Path("/Users/lvyan/Desktop/sequence");
//      FileInputFormat.addInputPath(job, input);
//      FileOutputFormat.setOutputPath(job,output);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        //set input and output format class
        job.setInputFormatClass(CombineSmallfileInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(IntWritable.class);

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

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

Step 7: Running this project and get result. The file I used was 2 ebooks.

```
19/10/19 15:54:22 INFO mapred.JobClient:  map 100% reduce 100%
19/10/19 15:54:22 INFO mapred.JobClient: Job complete: job_local828822811_0001
19/10/19 15:54:22 INFO mapred.JobClient: Counters: 17
19/10/19 15:54:22 INFO mapred.JobClient:   Map-Reduce Framework
19/10/19 15:54:22 INFO mapred.JobClient:     Spilled Records=434688
19/10/19 15:54:22 INFO mapred.JobClient:     Map output materialized bytes=2536011
19/10/19 15:54:22 INFO mapred.JobClient:     Reduce input records=217344
19/10/19 15:54:22 INFO mapred.JobClient:     Map input records=29907
19/10/19 15:54:22 INFO mapred.JobClient:     SPLIT_RAW_BYTES=255
19/10/19 15:54:22 INFO mapred.JobClient:     Map output bytes=2101317
19/10/19 15:54:22 INFO mapred.JobClient:     Reduce shuffle bytes=0
19/10/19 15:54:22 INFO mapred.JobClient:     Reduce input groups=12423
19/10/19 15:54:22 INFO mapred.JobClient:     Combine output records=0
19/10/19 15:54:22 INFO mapred.JobClient:     Reduce output records=12423
19/10/19 15:54:22 INFO mapred.JobClient:     Map output records=217344
19/10/19 15:54:22 INFO mapred.JobClient:     Combine input records=0
19/10/19 15:54:22 INFO mapred.JobClient:     Total committed heap usage (bytes)=458227712
19/10/19 15:54:22 INFO mapred.JobClient:   File Input Format Counters
19/10/19 15:54:22 INFO mapred.JobClient:     Bytes Read=0
19/10/19 15:54:22 INFO mapred.JobClient:   FileSystemCounters
19/10/19 15:54:22 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=5309083
19/10/19 15:54:22 INFO mapred.JobClient:     FILE_BYTES_READ=5067793
19/10/19 15:54:22 INFO mapred.JobClient:   File Output Format Counters
19/10/19 15:54:22 INFO mapred.JobClient:     Bytes Written=133115
```

Users > lvyan > Desktop > sequence > Part5_combine_ebooks > part-r-00000

```
    10566
"Defects,"  3
"Information    3
"Plain  6
"Project    15
"Right  3
#60463] 3
$5,000) 3
&   18
'AS-IS',    3
("the   3
($1 3
(801)   3
(_italics_).    3
(a) 3
(and    3
(any    3
(b) 3
(c) 3
(does   3
(if 3
(or 9
(trademark/copyright)   3
(www.gutenberg.org),    3
*   177
*** 12
*****   6
-   9
```

FIXEDLENGTHINPUTFORMAT

The input format used to read the records in the file is a fixed length. The content of the file does not have to be text or binary data. The user must set the length of the record by the parameter fixedlengthinputformat.record.length. The default value is 0. If the value of the parameter is less than or equal to 0, an IOException will be thrown. RecordReader is FixedLengthRecordReader.

Step 1: Implement Mapper class to count the frequency of each key. I use ebook1.txt and count the occurrence of each word. Once a word occurrence, will emit(word, 1) pair.

```java
public class FixLengthMapper extends Mapper<LongWritable, BytesWritable, Text, IntWritable>{
    private IntWritable one = new IntWritable(1);
    private Text word = new Text();
    @Override
    protected void map(LongWritable key, BytesWritable value,
            Mapper<LongWritable, BytesWritable, Text, IntWritable>.Context context)
            throws IOException, InterruptedException {
        // TODO Auto-generated method stub
        String[] tokens = value.toString().split("\\s+");
        for(String token : tokens) {
            word.set(token);
            context.write(word, one);
        }
    }
}
```

Step 2: Implement Reducer class to count the frequency of each key. Get the frequency of each word.

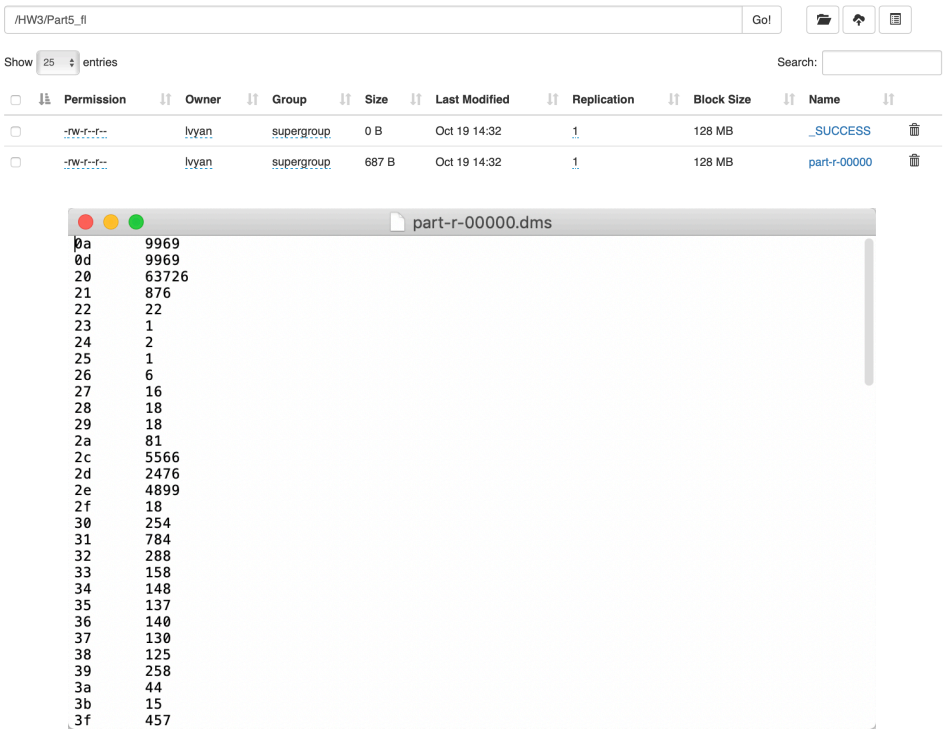
```java
public class FixLengthReducer extends Reducer<Text, IntWritable, Text, IntWritable>{
    @Override
    protected void reduce(Text key, Iterable<IntWritable> values,
            Reducer<Text, IntWritable, Text, IntWritable>.Context context) throws IOException, InterruptedException {
        // TODO Auto-generated method stub
        int sum = 0;
        for(IntWritable value : values) {
            sum += value.get();
        }
        context.write(key, new IntWritable(sum));
    }

}
```

Step 3: Implement Driver class

```java
public class FixLengthApp {
    public static void main(String[] args) throws IOException, ClassNotFoundException, InterruptedException {
        Configuration conf = new Configuration();
        conf.setInt(FixedLengthInputFormat.FIXED_RECORD_LENGTH, 1);

        Job job = Job.getInstance(conf, "fixed length input format for word count");

        //set driver class
        job.setJarByClass(FixLengthApp.class);

        //set mapper and reducer
        job.setMapperClass(FixLengthMapper.class);
        job.setReducerClass(FixLengthReducer.class);

        //set input and output format
        job.setInputFormatClass(FixedLengthInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        //set output key and value
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        //set input and output path
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Step 4: Run jar file and get results. The input path is ebooks/ebook1.txt, the output path is /HW3/Part5_fl.

hadoop jar /Users/lvyan/Desktop/hw3.jar HW3.Part5_FixedLengthInputFormat.FixLengthApp /ebooks/ebook1.txt /HW3/Part5_fl



KEYVALUETEXTINPUTFORMAT

InputFormat for plain text files, each line is divided into keys and values using a separator byte specified by the parameter mapreduce.input.keyvaluelinerecordreader.key.value.separator, which uses \t by default. If the separator does not exist, then the entire line will be used as the key and the value will be null. RecordReader is KeyValueLineRecordReader.

Step 1: Implement Mapper class to count the frequency of each key. I use ebook1.txt and count the occurrence of each word.

```
public class KeyValueMapper extends Mapper<Text, Text, Text, IntWritable>{

    IntWritable one = new IntWritable(1);
    Text word = new Text();
    @Override
    protected void map(Text key, Text value, Mapper<Text, Text, Text, IntWritable>.Context context)
            throws IOException, InterruptedException {
        // TODO Auto-generated method stub
        String[] tokens = value.toString().split("\\s+");
        for(String token : tokens) {
            word.set(token);
            context.write(word, one);
        }
    }

}
```

Step 2: Implement Reducer class to count the frequency of each key.

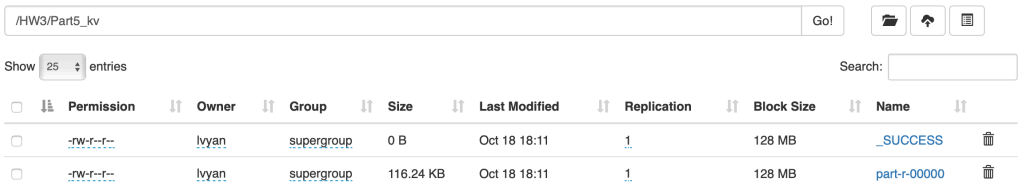
```java
public class KeyValueReducer extends Reducer<Text, IntWritable, Text, IntWritable>{

    @Override
    protected void reduce(Text key, Iterable<IntWritable> values,
            Reducer<Text, IntWritable, Text, IntWritable>.Context context) throws IOException, InterruptedException {
        // TODO Auto-generated method stub
        int sum = 0;
        for(IntWritable value : values) {
            sum += value.get();
        }
        context.write(key, new IntWritable(sum));
    }
}
```

Step 3: Implement Driver class.

```java
public class KeyValueApp {
    public static void main(String[] args) throws IOException, ClassNotFoundException, InterruptedException {
        Configuration conf = new Configuration();
        conf.set("mapreduce.input.keyvaluelinerecordreader.key.value.separator", " ");
        Job job = Job.getInstance(conf, "Key Value word count");

        job.setJarByClass(KeyValueApp.class);

        job.setInputFormatClass(KeyValueTextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        job.setMapperClass(KeyValueMapper.class);
        job.setReducerClass(KeyValueReducer.class);

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

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

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

    }
}
```

Step 4: Run jar file and get results. The input path is /ebooks/ebook1.txt and the output path is /HW3/Part5_kv.

hadoop jar /Users/lvyan/Desktop/hw3.jar HW3.Part5_KeyValueTextInputFormat.KeyValueApp /ebooks/ebook1.txt /HW3/Part5_kv



part-r-00000-2.dms

```
        3781
"Defects,"      1
"Information    1
"Plain  2
"Project        5
"Right  1
#60463] 1
$5,000) 1
&       6
'AS-IS',        1
("the   1
(801)   1
(_italics_).    1
(a)     1
(and    1
(any    1
(b)     1
(c)     1
(does   1
(if     1
(or     2
*       55
***     2
*****   1
-       3
1--Driven       1
1.      1
1.C     1
1.E     1
1.E.1   3
```

NLINEINPUTFORMAT

Use the N line in the input as an InputSplit, where N can be specified by the parameter mapreduce.input.lineinputformat.linespermap. The default is 1. RecordReader also uses LineRecordReader.

Step 1: Implement Mapper class to count the frequency of each key. I use ebook.txt and count the occurrence of each word.

```
public class NLMapper extends Mapper<LongWritable, Text, Text, IntWritable>{
    private IntWritable one = new IntWritable(1);
    private Text word = new Text();
    @Override
    protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text, IntWritable>.Context context)
            throws IOException, InterruptedException {
        // TODO Auto-generated method stub
        String[] tokens = key.toString().split("\\s+");
        for(String token : tokens) {
            word.set(token);
            context.write(word, one);
        }
    }
}
```

Step 2: Implement Reducer class to count the frequency of each key.

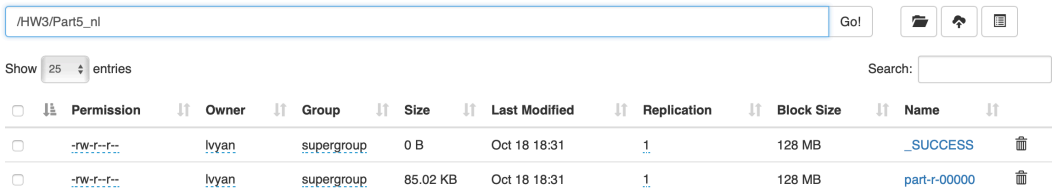
```
public class NLReducer extends Reducer<Text, IntWritable, Text, IntWritable>{

    @Override
    protected void reduce(Text key, Iterable<IntWritable> values,
            Reducer<Text, IntWritable, Text, IntWritable>.Context context) throws IOException, InterruptedException {
        // TODO Auto-generated method stub
        int sum = 0;
        for(IntWritable value : values) {
            sum += value.get();
        }

        context.write(key, new IntWritable(sum));
    }

}
```

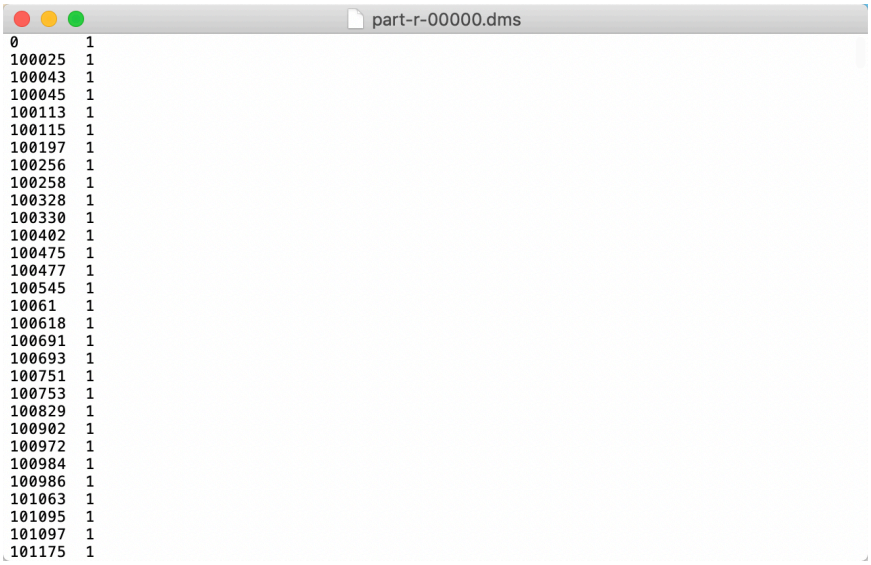
Step 3: Implement Driver class.

```java
public class NLApp {

    public static void main(String[] args) throws IOException, ClassNotFoundException, InterruptedException {

        Configuration conf = new Configuration();
        conf.setInt("mapreduce.input.lineinputformat.linespermap", 500);

        Job job = Job.getInstance(conf, "nl word count");

        job.setJarByClass(NLApp.class);

        job.setInputFormatClass(NLineInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        job.setMapperClass(NLMapper.class);
        job.setReducerClass(NLReducer.class);

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

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

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

    }
}
```

Step 4: Run jar file and get results. The input path is /ebooks/ebook1.txt and the output path is /HW3/Part5_nl.

hadoop jar /Users/lvyan/Desktop/hw3.jar HW3.Part5_NLineInputFormat.NLApp /ebooks/ebook1.txt /HW3/Part5_nl





SEQUENCEFILEINPUTFORMAT

The InputFormat for the sequence file, the method that gets InputSplit inherits from FileInputFormat, is not overridden, and its RecordReader is SequenceFileRecordReader. This class has three subclasses: SequenceFileAsBinaryInputFormat(read the key value from the binary format of the sequence file), SequenceFileAsTextInputFormat(convert the key values in the sequence file to a string form), and SequenceFileInputFilter(samples are taken from the sequence file and then processed by the MapReduce job, and the samples are determined by the filter class). The RecordReader of the three are: SequenceFileAsBinaryRecordReader, SequenceFileAsTextRecordReader, and FilterRecordReader.

Step 1: create a sequence file. In this file, I add 104 records. Then store this file to local file system.

```java
public class GenerateSequence {
    public static void main(String[] args) throws IOException {
        Configuration conf = new Configuration();
        Path path = new Path("/Users/lvyan/Desktop/sequence/seq.seq");
        FileSystem fs = FileSystem.getLocal(conf);
        SequenceFile.Writer writer = SequenceFile.createWriter(fs, conf, path, Text.class, IntWritable.class);

        for(int i = 0; i < 100; i++) {
            writer.append(new Text("" + i), new IntWritable(i));
        }

        writer.append(new Text("" + 99), new IntWritable(2));
        writer.append(new Text("" + 10), new IntWritable(2));
        writer.append(new Text("" + 33), new IntWritable(2));
        writer.append(new Text("" + 12), new IntWritable(2));

        writer.close();
    }
}
```

Step 2: Upload sequence file to hdfs.

```
LVdeMacBook-Pro:Desktop lvyan$ hadoop fs -copyFromLocal /Users/lvyan/Desktop/sequence/seq.seq /HW3
2019-10-18 19:30:25,752 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where app
licable
```

☐	-rw-r--r--	lvyan	supergroup	2.45 KB	Oct 18 19:30	1	128 MB	seq.seq

Step 3: Implement Mapper class to count the frequency of each key.

```java
public class SequenceMapper extends Mapper<Text, IntWritable, Text, IntWritable>{

    @Override
    protected void map(Text key, IntWritable value, Mapper<Text, IntWritable, Text, IntWritable>.Context context)
            throws IOException, InterruptedException {
        // TODO Auto-generated method stub
        int v = Integer.valueOf(value.toString());
        context.write(key,  new IntWritable(v));
    }
}
```

Step 4: Implement Reducer class to count the frequency of each key.

```java
public class SequenceReducer extends Reducer<Text, IntWritable, Text, IntWritable>{

    @Override
    protected void reduce(Text key, Iterable<IntWritable> values,
            Reducer<Text, IntWritable, Text, IntWritable>.Context context) throws IOException, InterruptedException {
        // TODO Auto-generated method stub
        int sum = 0;
        for(IntWritable value : values) {
            sum += value.get();
        }
        context.write(key, new IntWritable(sum));
    }
}
```

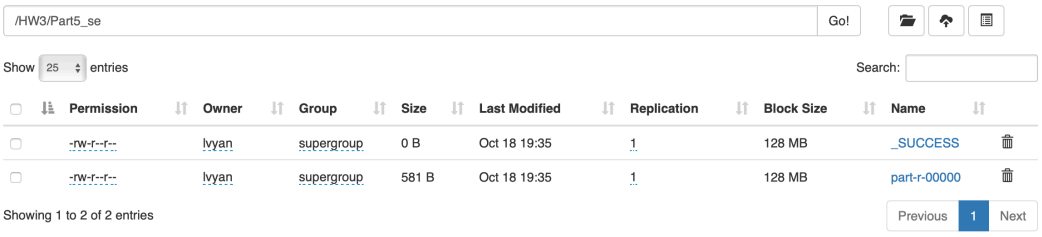
Step 5: Implement Driver class

```
public static void main(String[] args) throws IOException, ClassNotFoundException, InterruptedException {
    Configuration conf = new Configuration();

    Job job = Job.getInstance(conf, "sequence input format");

    job.setJarByClass(SequenceApp.class);

    job.setMapperClass(SequenceMapper.class);
    job.setReducerClass(SequenceReducer.class);

    job.setInputFormatClass(SequenceFileInputFormat.class);
    job.setOutputFormatClass(TextOutputFormat.class);

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

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

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

}
```

Step 6: Run jar file and get results. The input path is /HW3/seq.seq and the output path is /HW3/Part5_se.

hadoop jar /Users/lvyan/Desktop/hw3.jar HW3.Part5_SequenceFileInputFormat.SequenceApp /HW3/seq.seq /HW3/Part5_se

```
LVdeMacBook-Pro:Desktop lvyan$ hadoop jar /Users/lvyan/Desktop/Part5_se.jar HW3.Part5_SequenceFileInputFormat.SequenceApp /HW3/seq.seq /HW3/Part5_se
2019-10-18 19:35:01,073 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2019-10-18 19:35:01,772 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
2019-10-18 19:35:02,635 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute you
er to remedy this.
2019-10-18 19:35:02,663 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/lvyan/.staging/job_1571435001307_(
2019-10-18 19:35:02,805 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2019-10-18 19:35:03,273 INFO input.FileInputFormat: Total input files to process : 1
2019-10-18 19:35:03,290 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2019-10-18 19:35:03,301 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2019-10-18 19:35:03,307 INFO mapreduce.JobSubmitter: number of splits:1
2019-10-18 19:35:03,404 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2019-10-18 19:35:03,424 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1571435001307_0005
2019-10-18 19:35:03,425 INFO mapreduce.JobSubmitter: Executing with tokens: []
2019-10-18 19:35:03,574 INFO conf.Configuration: resource-types.xml not found
2019-10-18 19:35:03,574 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2019-10-18 19:35:03,632 INFO impl.YarnClientImpl: Submitted application application_1571435001307_0005
2019-10-18 19:35:03,663 INFO mapreduce.Job: The url to track the job: http://LVdeMacBook-Pro.local:8088/proxy/application_1571435001307_0005/
2019-10-18 19:35:03,664 INFO mapreduce.Job: Running job: job_1571435001307_0005
2019-10-18 19:35:12,786 INFO mapreduce.Job: Job job_1571435001307_0005 running in uber mode : false
2019-10-18 19:35:12,788 INFO mapreduce.Job:  map 0% reduce 0%
2019-10-18 19:35:16,854 INFO mapreduce.Job:  map 100% reduce 0%
2019-10-18 19:35:21,909 INFO mapreduce.Job:  map 100% reduce 100%
2019-10-18 19:35:21,916 INFO mapreduce.Job: Job job_1571435001307_0005 completed successfully
2019-10-18 19:35:22,026 INFO mapreduce.Job: Counters: 50
        File System Counters
                FILE: Number of bytes read=932
                FILE: Number of bytes written=452935
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
```



part-r-00000.dms

```
0	0
1	1
10	12
11	11
12	14
13	13
14	14
15	15
16	16
17	17
18	18
19	19
2	2
20	20
21	21
22	22
23	23
24	24
25	25
26	26
27	27
28	28
29	29
3	3
30	30
31	31
32	32
33	35
34	34
35	35
```

TEXTINPUTFORMAT

The InputFormat for plain text files is also the default InputFormat. The input file is broken down into lines, Enter or line feed as a mark for the end of the line, the key is the position of the line in the file, and the value is the line content. The InputFormat uses the LineRecordReader to read the contents of the InputSplit.

Step 1 : Create Mapper class. Count the occurrence of each word. Line is split by blank.

```
10 public class TextMapper extends Mapper<LongWritable, Text, Text, IntWritable>{
11
12     Text word = new Text();
13     IntWritable one = new IntWritable(1);
14
15     public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
16         String line = value.toString();
17         String[] tokens = line.split("\\s+");
18         for(String  token : tokens) {
19             word.set(token);
20             context.write(word , one);
21         }
22     }
23 }
```

Step 2: Implement Reducer class. Get the sum of each key.

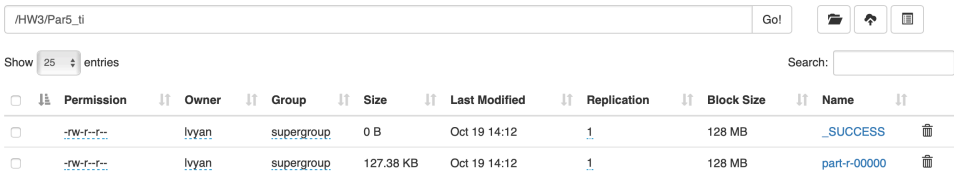
```
8
9  public class TextReducer extends Reducer<Text, IntWritable, Text, IntWritable>{
10     @Override
11     protected void reduce(Text key, Iterable<IntWritable> values,
12             Reducer<Text, IntWritable, Text, IntWritable>.Context context) throws IOException, InterruptedException {
13         // TODO Auto-generated method stub
14         int sum = 0;
15         for(IntWritable val : values) {
16             sum += val.get();
17         }
18         IntWritable count = new IntWritable(sum);
19         context.write(key, count);
20     }
21 }
```

Step 3: Implement dirver class.

```java
public class TextApp {
    public static void main( String[] args )
        {
//          System.out.println( "Hello World!" );
            Configuration conf = new Configuration();
            // Create a new Job
            try {
                Job job = Job.getInstance(conf, "Text Input Format");
                //set the driver class
                //a class contains the name of main method
                job.setJarByClass(TextApp.class);

                //set mappers and reducer classes
                job.setMapperClass(TextMapper.class);
                job.setReducerClass(TextReducer.class);

                //set inputformat and outputformat class
                job.setInputFormatClass(TextInputFormat.class);
                job.setOutputFormatClass(TextOutputFormat.class);

                //set the output key and value type
                job.setOutputKeyClass(Text.class);
                job.setOutputValueClass(IntWritable.class);

                //set the input and output path
                //path will come from the command line at the arg0
                FileInputFormat.addInputPath(job, new Path(args[0]));
                FileOutputFormat.setOutputPath(job, new Path(args[1]));

                System.exit(job.waitForCompletion(true) ? 0 : 1);
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
    catch (ClassNotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
}
```

Step 4: Run and get Results. The input path is /ebooks/ebook1.txt and the output path is /HW3/Par5_ti.

hadoop jar /Users/lvyan/Desktop/hw3.jar HW3.Part5_TextInputFormat.TextApp /ebooks/ebook1.txt /HW3/Par5_ti



part-r-00000.dms

```
        3522
"Defects,"      1
"Information    1
"Plain  2
"Project        5
"Right  1
#60463] 1
$5,000) 1
&       6
'AS-IS',        1
("the   1
($1     1
(801)   1
(_italics_).    1
(a)     1
(and    1
(any    1
(b)     1
(c)     1
(does   1
(if     1
(or     3
(trademark/copyright)   1
(www.gutenberg.org),    1
*       59
**      4
*****   2
-       3
1--Driven       1
1.      1
```

All the applications when I execute hadoop jar …jar ..App <input> <output>.

ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus
application_1571508605636_0010	lvyan	Access count Part3	MAPREDUCE	default	0	Sat Oct 19 14:49:51 -0400 2019	Sat Oct 19 14:49:51 -0400 2019	Sat Oct 19 14:50:10 -0400 2019	FINISHED	SUCCEEDED
application_1571508605636_0009	lvyan	Find the max stock price	MAPREDUCE	default	0	Sat Oct 19 14:45:23 -0400 2019	Sat Oct 19 14:45:23 -0400 2019	Sat Oct 19 14:45:59 -0400 2019	FINISHED	SUCCEEDED
application_1571508605636_0008	lvyan	Find the max stock price	MAPREDUCE	default	0	Sat Oct 19 14:40:32 -0400 2019	Sat Oct 19 14:40:33 -0400 2019	Sat Oct 19 14:41:52 -0400 2019	FINISHED	SUCCEEDED
application_1571508605636_0007	lvyan	fixed length input format for word count	MAPREDUCE	default	0	Sat Oct 19 14:32:07 -0400 2019	Sat Oct 19 14:32:07 -0400 2019	Sat Oct 19 14:32:24 -0400 2019	FINISHED	SUCCEEDED
application_1571508605636_0005	lvyan	Key Value word count	MAPREDUCE	default	0	Sat Oct 19 14:29:02 -0400 2019	Sat Oct 19 14:29:02 -0400 2019	Sat Oct 19 14:29:19 -0400 2019	FINISHED	SUCCEEDED
application_1571508605636_0004	lvyan	nl word count	MAPREDUCE	default	0	Sat Oct 19 14:25:42 -0400 2019	Sat Oct 19 14:25:42 -0400 2019	Sat Oct 19 14:26:32 -0400 2019	FINISHED	SUCCEEDED
application_1571508605636_0002	lvyan	sequence input format	MAPREDUCE	default	0	Sat Oct 19 14:20:07 -0400 2019	Sat Oct 19 14:20:08 -0400 2019	Sat Oct 19 14:20:25 -0400 2019	FINISHED	SUCCEEDED
application_1571508605636_0001	lvyan	Text Input Format	MAPREDUCE	default	0	Sat Oct 19 14:12:33 -0400 2019	Sat Oct 19 14:12:34 -0400 2019	Sat Oct 19 14:12:53 -0400 2019	FINISHED	SUCCEEDED