HW2 Problem 1: HDFS (50%)

(a)

```
[root@quickstart cse427s]# hadoop fs -ls shakespeare
Found 4 items
-rw-r--r--   1 root supergroup    1784616 2018-09-07 20:43 shakespeare/comedies
-rw-r--r--   1 root supergroup    1479035 2018-09-07 20:43 shakespeare/histories
-rw-r--r--   1 root supergroup     268140 2018-09-07 20:43 shakespeare/poems
-rw-r--r--   1 root supergroup    1752440 2018-09-07 20:43 shakespeare/tragedies
```

```
[cloudera@quickstart ~]$ hadoop fs -cat shakespeare/poems | head -n 16



        SONNETS



TO THE ONLY BEGETTER OF
THESE INSUING SONNETS
MR. W. H. ALL HAPPINESS
AND THAT ETERNITY
PROMISED BY
OUR EVER-LIVING POET WISHETH
THE WELL-WISHING
ADVENTURER IN
SETTING FORTH
cat: Unable to write to output stream.
[cloudera@quickstart ~]$
```

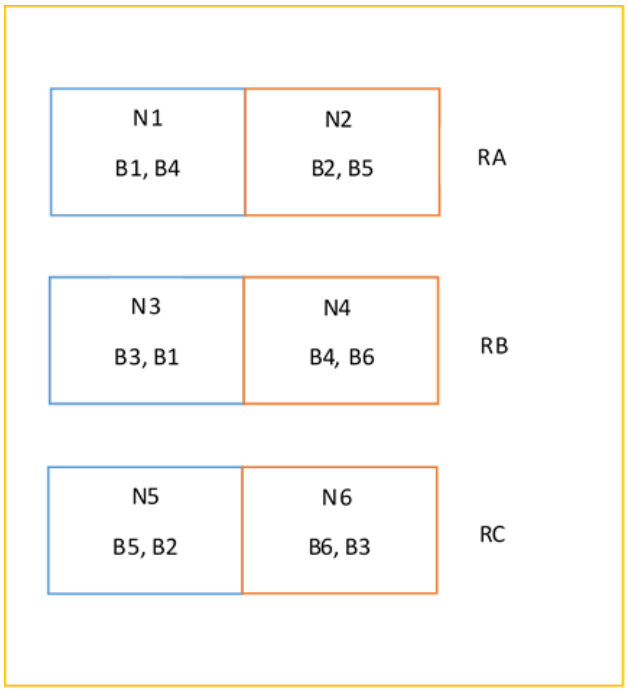
(b) 1) If the replication factor is large, it means that all the datanodes will be stored in different racks for many times. So the storage capacity of different data reduce and the storage efficiency will be very low.

2) Namenode stores meta-data about files and blocks. If the replication factor is large, namenode need to store multiple dictionaries for each file or block. So it can lead to memory insufficiency. Namenode is important to the system so we need to back up the namenode or run a secondary namenode. Both ways require plenty of CPU or large memory when the replication factor is large.

(c) Benefit: large blocks can minimize the cost of seeks. If the block is large enough, the time it takes to transfer the data from the disk can be significantly longer than the time to seek to the start of the block. Thus, transferring a large file made of multiple blocks operates at the disk transfer rate.

Disadvantage: Map tasks in MapReduce normally operate on one block at a time. If you have too few tasks, your job will run slower than they could otherwise. The data processing in each block is sequential. So if blocks are too large, the processing velocity inside blocks will be very slow.

(d)



Meta-data stored on the master node:
Data file: B1, B2, B3, B4, B5, B6.
B1: N1, N3
B2: N2, N5
B3: N3, N6
B4: N4, N1
B5: N5, N2
B6: N6, N4

(e)

1) HDFS does not work well for Low-latency data access. HDFS is optimized for delivering a high throughput of data, and this may be at the expense of latency.
2) HDFS does not work well for lots of small files. Because the name node holds filesystem metadata in memory, the limit to the number of files in a filesystem is governed by the amount of memory on the name node. As a rule of thumb, each file, directory, and block takes about 150 bytes.

(f) Command: hadoop fs -get /user/bar.txt baz.txt

The client visit namenode first and get file-block dictionary and block-location dictionary. Then collect blocks from datanodes according to those dictionaries. Copy the file bar.txt in the user's directory in HDFS /user to the local disk, named as baz.txt

HW2 Problem 2: MapReduce I (25%)

Mapper input: (15, 21, 24, 30, 49)

Mapper output: (3, 15) (5, 15) (3, 21) (7, 21) (2, 24) (3, 24) (2, 30) (3, 30) (5, 30) (7, 49)

Reducer input: (2, [24, 30]) (3, [15, 21, 24, 30]) (5, [15, 30]) (7, [21, 49])

Reducer output: (2, 54) (3, 90) (5, 45) (7, 70)

HW2 Problem 3: MapReduce II (25%)

(a) Mapper output: (gif, 1200) (html, 900) (gif, 1900) (jpg, 4000) (html. 1100)
Reducer input: (gif, [1200, 1900]) (html, [900, 1100]) (jpg, 4000)
Reducer output: (gif, 1550) (html, 1000) (jpg, 4000)

(b) We can use text data or string type to represent the keys and integer or double to represent values.