

## بخش اول:

```

→ hind git:(main) sh start-hadoop.sh
WARN[0000] /home/nda/cc_projects/p3/hind/docker-compose.yml: 'version' is obsolete
[+] Running 26/2
  ✓ datanode 3 layers [██████]    0B/0B      Pulled      570.4s
  ✓ namenode 21 layers [██████████]   0B/0B      Pulled      570.4s
[+] Running 2/3
  ⚡ Network hind_default     Created           1.2s
  ✓ Container namenode     Started          0.7s
  ✓ Container hind-datanode-1 Started          0.7s
Namenode: http://127.0.0.1:50070
Datanode: http://127.0.0.1:50075
→ hind git:(main) docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
e122c12c9fdd      docker.arvancloud.ir/bde2020/hadoop-datanode:1.1.0-hadoop2.8-java8   "/entrypoint.sh sh ..."   43 minutes ago   Up 43 minutes (healthy)
          0.0.0.0:50075->50075/tcp, ::::50075->50075/tcp  hind-datanode-1
c64c820cf2cd      docker.arvancloud.ir/bde2020/hadoop-namenode:1.1.0-hadoop2.8-java8   "/entrypoint.sh sh ..."   43 minutes ago   Up 43 minutes (healthy)
          0.0.0.0:50070->50070/tcp, ::::50070->50070/tcp  namenode
706c319e7177      bitnami/kafka:latest           "/opt/bitnami/script..."   8 days ago       Up Less than a second (h
          health: starting  0.0.0.0:9092->9092/tcp, ::::9092->9092/tcp  kafka
→ hind git:(main)

```

کانتینرها:

## :namenode

نقش: مدیریت کننده مرکزی داده‌ها در Hadoop

وظیفه:

ذخیره و مدیریت متادادا (metadata) مربوط به فایل‌های ذخیرشده در سیستم

ردیابی مکان دقیق بلاک‌های داده که در دیتابانودها (DataNodes) ذخیره شده‌اند

پاسخ به درخواست‌های خواندن/نوشتن از کلاینت‌ها

آدرس دسترسی: رابط وب آن در آدرس <http://127.0.0.1:50070>

## :datanode

نقش: ذخیره‌سازی داده‌ها در سیستم HDFS

وظیفه:

ذخیره و مدیریت بلاک‌های واقعی داده

گزارش دادن به namenode درباره وضعیت سلامت و مکان بلاک‌ها

پاسخ به درخواست‌های خواندن و نوشتمن که از طرف کلاینت یا namenode ارسال می‌شوند

آدرس دسترسی: رابط وب آن در آدرس <http://127.0.0.1:50075>

## :kafka

نقش: پیاده‌سازی صف پیام (Message Queue).

وظیفه:

ارسال، دریافت و ذخیره پیام‌ها برای ارتباطات بین سیستم‌ها

عمل کردن به عنوان یک سیستم پخش داده‌های بلادرنگ که می‌تواند برای پردازش داده‌ها و جریان‌های بلادرنگ در محیط

Hadoop استفاده شود

این سرویس معمولاً برای انتقال داده‌های بزرگ و عملیات بلادرنگ بین کلاینت‌ها یا سیستم‌های مختلف (مانند Flink یا Spark)

موردن استفاده قرار می‌گیرد

وضعیت فعلی آن در حال تلاش برای شروع است (Health: Starting)



## Overview 'namenode:8020' (active)

<b>Started:</b>	Tue Dec 03 11:44:55 +0330 2024
<b>Version:</b>	2.8.0, r91f2b7a13d1e97be65db92ddabc627cc29ac0009
<b>Compiled:</b>	Fri Mar 17 07:42:00 +0330 2017 by jdu from branch-2.8.0
<b>Cluster ID:</b>	CID-e60d416a-d04a-47d4-8683-486c0570c735
<b>Block Pool ID:</b>	BP-198958037-172.18.0.2-1733213694256

## Summary

Security is off.

Safemode is off.

1 files and directories, 0 blocks = 1 total filesystem object(s).

Heap Memory used 111.45 MB of 218.5 MB Heap Memory. Max Heap Memory is 889 MB.

Non Heap Memory used 45.71 MB of 46.5 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>

<b>Configured Capacity:</b>	95.56 GB
<b>DFS Used:</b>	28 KB (0%)
<b>Non DFS Used:</b>	82.46 GB

DFS Health Overview	
Safemode is off.	
1 files and directories, 0 blocks = 1 total filesystem object(s).	
Heap Memory used 111.45 MB of 218.5 MB Heap Memory. Max Heap Memory is 889 MB.	
Non Heap Memory used 45.71 MB of 46.5 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.	
Configured Capacity:	
DFS Used:	95.56 GB
Non DFS Used:	28 KB (0%)
DFS Remaining:	82.46 GB
Block Pool Used:	8.2 GB (8.58%)
DataNodes usages% (Min/Median/Max/stdDev):	28 KB (0%)
Live Nodes	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	1 (Decommissioned: 0)
Dead Nodes	0 (Decommissioned: 0)
Decommissioning Nodes	0
Total Datanode Volume Failures	0 (0 B)
Number of Under-Replicated Blocks	0
Number of Blocks Pending Deletion	0
Block Deletion Start Time	Tue Dec 03 11:44:55 +0330 2024
Last Checkpoint Time	Tue Dec 03 11:44:54 +0330 2024

localhost:50070/dfsh										You can paste the image from the clipboard.	
<b>Block Deletion Start Time</b>										Tue Dec 03 11:44:55 +0330 2024	
<b>Last Checkpoint Time</b>										Tue Dec 03 11:44:54 +0330 2024	

## NameNode Journal Status

Current transaction ID: 1

Journal Manager	State
FileJournalManager(root=/hadoop/dfs/name)	EditLogFileOutputStream(/hadoop/dfs/name/current/edits_inprogress_00000000000000000001)

## NameNode Storage

Storage Directory	Type	State
/hadoop/dfs/name	IMAGE_AND_EDITS	Active

## DFS Storage Types

Storage Type	Configured Capacity	Capacity Used	Capacity Remaining	Block Pool Used	Nodes In Service
DISK	95.56 GB	28 KB (0%)	8.2 GB (8.58%)	28 KB	1

Hadoop, 2017.

localhost:50075/datanode.html											
Hadoop Overview Utilities ▾											

## DataNode on localhost:50010

<b>Cluster ID:</b>	CID-e60d416a-d04a-47d4-8683-486c0570c735
<b>Version:</b>	2.8.0

## Block Pools

Namenode Address	Block Pool ID	Actor State	Last Heartbeat	Last Block Report
namenode:8020	BP-198958037-172.18.0.2-1733213694256	RUNNING	2s	an hour

## Volume Information

Directory	Capacity Used	Capacity Left	Capacity Reserved	Reserved Space for Replicas	Blocks
/hadoop/dfs/data/current	28 KB	8.19 GB	0 B	0 B	0

Hadoop, 2017.

## بخش دوم:

```
⚡ mapper.py > ...
1  #!/usr/bin/env python
2  import sys
3  import re
4
5  #pattern to match key-value pairs
6  pattern = r'(\w+),(.+)' 
7
8  for line in sys.stdin:
9      line = line.strip()
10     match = re.search(pattern, line)
11     if match:
12         key = match.group(1) #document id
13         value = match.group(2) #content
14         words = value.split()
15         for word in words:
16             print("{}\t{}".format(word.lower(), key))
17
```

```
⚡ reducer.py > ...
1  #!/usr/bin/env python
2  import sys
3  from collections import defaultdict
4
5  #dict to store the documents where each word is
6  word_docs = defaultdict(set)
7
8  #processing input from the mapper
9  for line in sys.stdin:
10     line = line.strip()
11     word, doc = line.split('\t', 1)
12     word_docs[word].add(doc)
13
14 #output each word with its set of docs
15 for word, docs in sorted(word_docs.items()):
16     print("{} : {}".format(word, ', '.join(sorted(docs))))
```

```
→ hind git:(main) ✘ docker cp ./reducer.py c64c820cf2cd:.
Successfully copied 2.05kB to c64c820cf2cd:.
→ hind git:(main) ✘ docker cp ./mapper.py c64c820cf2cd:.
Successfully copied 2.05kB to c64c820cf2cd:.
→ hind git:(main) ✘ docker exec -it c64 bash
root@c64c820cf2cd:/# ls
bin  dev  entrypoint.sh  hadoop  home  lib  mapper.py  mnt  proc  root  run.sh  srv  tmp  var
boot  dh-python_2.20170125~bp08+1_all.deb  etc  hadoop-data  init.sh  lib64  media  opt  reducer.py  run  sbin  sys  usr
root@c64c820cf2cd:/#
```

```
root@c64c820cf2cd:~/input# cat input1.txt
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.
```

```
SUCCESSFULLY COPIED 2.05KB TO c04c820cf2cd:.
→ hind git:(main) ✘ docker exec -it c64 bash
root@c64c820cf2cd:/# hadoop jar /opt/hadoop-2.8.0/share/hadoop/tools/lib/hadoop-streaming-2.8.0.jar -files mapper.py,reducer.py -mapper mapper.py -red
ucer reducer.py -input /input.txt -output output
24/12/03 10:19:50 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.session-id
24/12/03 10:19:50 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessionId=
24/12/03 10:19:50 INFO jvm.JvmMetrics: Cannot initialize JVM Metrics with processName=JobTracker, sessionId= - already initialized
24/12/03 10:19:51 INFO mapred.FileInputFormat: Total input files to process : 1
24/12/03 10:19:51 INFO mapreduce.JobSubmitter: number of splits:1
24/12/03 10:19:51 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local1779845027_0001
24/12/03 10:19:51 INFO mapred.LocalDistributedCacheManager: Localized file:/mapper.py as file:/tmp/hadoop-root/mapred/local/1733221191314/mapper.py
24/12/03 10:19:51 INFO mapred.LocalDistributedCacheManager: Localized file:/reducer.py as file:/tmp/hadoop-root/mapred/local/1733221191315/reducer.py
24/12/03 10:19:51 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
24/12/03 10:19:51 INFO mapred.LocalJobRunner: OutputCommitter set in config null
24/12/03 10:19:51 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCommitter
24/12/03 10:19:51 INFO mapreduce.Job: Running job: job_local1779845027_0001
24/12/03 10:19:51 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
24/12/03 10:19:51 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup fa
ilures: false
24/12/03 10:19:51 INFO mapred.LocalJobRunner: Waiting for map tasks
24/12/03 10:19:51 INFO mapred.LocalJobRunner: Starting task: attempt_local1779845027_0001_m_000000_0
24/12/03 10:19:51 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
24/12/03 10:19:51 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup fa
```

```
24/12/03 10:19:52 INFO mapreduce.Job: map 100% reduce 100%
24/12/03 10:19:52 INFO mapreduce.Job: Job job_local1779845027_0001 completed successfully
24/12/03 10:19:52 INFO mapreduce.Job: Counters: 35
  File System Counters
    FILE: Number of bytes read=277136
    FILE: Number of bytes written=942930
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=3950
    HDFS: Number of bytes written=3291
    HDFS: Number of read operations=13
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=4
  Map-Reduce Framework
    Map input records=25
    Map output records=231
    Map output bytes=3140
    Map output materialized bytes=3608
    Input split bytes=83
    Combine input records=0
    Combine output records=0
    Reduce input groups=140
    Reduce shuffle bytes=3608
    Reduce input records=231
    Reduce output records=140
    Spilled Records=462
    Shuffled Maps =1
    Failed Shuffles=0
    Merged Map outputs=1
    GC time elapsed (ms)=10
    Total committed heap usage (bytes)=710410240
  Shuffle Errors
    BAD_ID=0
    CONNECTION=0
    IO_ERROR=0
    WRONG_LENGTH=0
    WRONG_MAP=0
    WRONG_REDUCE=0
  File Input Format Counters
    Bytes Read=1975
  File Output Format Counters
    Bytes Written=3291
24/12/03 10:19:52 INFO streaming.StreamJob: Output directory: output
root@c64c820cf2cd:/#
```

The screenshot shows the Hadoop File Explorer interface at the URL `localhost:50070/explorer.html#/`. The top navigation bar includes links for Hadoop, Overview, Datanodes, Datanode Volume Failures, Snapshot, Startup Progress, and Utilities.

## Browse Directory

Browse Directory							
Path		File Details				Actions	
Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	root	supergroup	1.93 KB	Dec 03 13:19	3	128 MB	input1.txt
drwxr-xr-x	root	supergroup	0 B	Dec 03 13:24	0	0 B	user

Show 25 entries Search:  Go! File icon

Showing 1 to 2 of 2 entries Previous 1 Next

Hadoop, 2017.

## Browse Directory

Browse Directory							
Path		File Details				Actions	
Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	root	supergroup	0 B	Dec 03 13:49	0	0 B	output

Show 25 entries Search:  Go! File icon

Showing 1 to 1 of 1 entries Previous 1 Next

Hadoop, 2017.

## Browse Directory

Browse Directory							
Path		File Details				Actions	
Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	root	supergroup	0 B	Dec 03 13:49	3	128 MB	_SUCCESS
-rw-r--r--	root	supergroup	3.21 KB	Dec 03 13:49	3	128 MB	part-00000

Show 25 entries Search:  Go! File icon

Showing 1 to 2 of 2 entries Previous 1 Next

## Output (part-00000) :

```
1 5g : {doc16}
2 accessibility : {doc9}
3 advanced : {doc25}
4 advancements : {doc22}
5 advances : {doc10}
6 agricultural : {doc10}
7 analysis : {doc20, doc21, doc3}
8 analytics : {doc1, doc11, doc18, doc8}
9 and : {doc1, doc10, doc12, doc13, doc16, doc19, doc2, doc20, doc21, doc23, doc25, doc4, doc5, doc6, doc7, doc8, doc9}
10 are : {doc14, doc21}
11 artificial : {doc2, doc22}
12 automates : {doc12}
13 automation : {doc15}
14 autonomous : {doc2}
15 behavior : {doc8}
16 benefits : {doc23}
17 big : {doc8}
18 blockchain : {doc7}
19 business : {doc21}
20 change : {doc3}
21 climate : {doc3}
22 cloud : {doc13}
23 collection : {doc12}
24 combat : {doc3}
25 complex : {doc25, doc4}
26 computing : {doc13, doc25, doc4}
27 connectivity : {doc16}
28 consumer : {doc8}
29 critical : {doc14}
30 cryptocurrency : {doc7}
31 cybersecurity : {doc14}
32 data : {doc1, doc10, doc11, doc12, doc13, doc14, doc15, doc16, doc17, doc18, doc19, doc2, doc20, doc21, doc22, doc23, doc24, doc25, doc3, doc4, doc5, doc6, doc7, doc8, doc9}
33 decisions : {doc24}
34 digital : {doc14}
35 driven : {doc15, doc24, doc6}
36 drives : {doc2}
37 e-commerce : {doc20}
38 education : {doc18}
39 educational : {doc6}
40 energy : {doc24}
41 engineering : {doc10}
42 enhances : {doc16, doc4, doc7}
43 environmental : {doc17}
```

توضیحات بخش دوم:

کد :mapper.py

در این کد هدف استخراج کلمات از محتواهای داکها و چاپ آنها به همراه شناسه داک است. مراحل:

الگوی استخراج شناسه داک و محتوا:

از یک الگوی (pattern) برای استخراج دو قسمت از ورودی استفاده می‌شود. این الگو به صورت (+W)(+W) است:  
(+W) شناسه داک (که شامل حروف و اعداد است) را استخراج می‌کند  
(+) محتوای داک را که ممکن است شامل هر نوع داده‌ای باشد، استخراج می‌کند

تقسیم‌بندی محتوا به کلمات:

پس از استخراج محتوا، آن را به کلمات تقسیم می‌کنیم (value.split())  
این کلمات به صورت حروف کوچک (با استفاده از word.lower()) تبدیل می‌شوند

خروجی: برای هر کلمه در داک، شناسه داک به همراه کلمه چاپ می‌شود. فرمت خروجی به صورت word\tdoc\_id است که در آن ۱۴ یک تب است

کد :reducer.py

در این کد داده‌هایی که از mapper ارسال شده‌اند پردازش می‌شود تا کلمات و داکاتی که آن‌ها در آن‌ها ظاهر شده‌اند، مشخص شوند

ساخت دیکشنری برای ذخیره داکات هر کلمه:

یک دیکشنری به نام word\_docs ساخته می‌شود که در آن برای هر کلمه یک مجموعه (set) از شناسه‌های داکات قرار می‌گیرد  
این دیکشنری به گونه‌ای تنظیم شده است که برای هر کلمه، داکات تکراری را ذخیره نکند (چون از set استفاده می‌کنیم)

### پردازش ورودی:

ورودی‌های دریافتی از mapper به صورت خط به خط پردازش می‌شود. در هر خط، کلمه و شناسه داک جدا می‌شود.  
برای هر کلمه، شناسه داک به مجموعه‌ای که در آن کلمه ذخیره شده اضافه می‌شود

خروجی: پس از پردازش همه داده‌ها، کلمات به ترتیب حروف الفبا مرتب می‌شوند.  
برای هر کلمه، داکاتی که آن کلمه در آن‌ها ظاهر شده است چاپ می‌شود. فرمت خروجی به صورت `{word : {doc_ids}}` است.

### توضیح خروجی:

`5g : {doc16}`: کلمه 5g فقط در داک doc16 ظاهر شده است.  
`doc8, doc1, doc11, doc18`: کلمه analytics در داکات `{doc1, doc11, doc18, doc8}` ظاهر شده است.

`and : {doc1, doc10, doc12, doc13, doc16, doc19, doc2, doc20, doc21, doc23, doc25, doc1, doc10, doc12}`: کلمه and در داکات مختلف از جمله `{doc4, doc5, doc6, doc7, doc8, doc9}` وغیره ظاهر شده است.

این خروجی نشان می‌دهد که هر کلمه در کدام داکها ظاهر شده است. این نوع پردازش برای ایجاد شاخص‌های معکوس (inverted index) مفید است که در موتورهای جستجو برای جستجوی سریع کلمات در داکات مختلف استفاده می‌شود

### بخش سوم:

```
 mapper2.py > ...
 1  #!/usr/bin/env python
 2  import sys
 3  import re
 4
 5  #extracting document id and content
 6  pattern = r'(\w+),(.+)'
 7
 8  for line in sys.stdin:
 9      line = line.strip()
10      match = re.search(pattern, line)
11      if match:
12          doc_id = match.group(1) #extracting doc id
13          content = match.group(2) #extracting content of the document
14          words = content.lower().split() #converting to lowercase and split into words
15          for word in words:
16              print("{}\t{}".format("{}", "{}".format(word, doc_id), 1))
```

```

reducer2.py > ...
1  #!/usr/bin/env python
2  import sys
3  from collections import defaultdict
4
5  # dict to store word frequencies for each document
6  doc_word_counts = defaultdict(lambda: defaultdict(int))
7
8  # from mapper
9  for line in sys.stdin:
10     line = line.strip()
11     key, count = line.split('\t')
12     word, doc_id = key.split(',')
13     doc_word_counts[doc_id][word] += int(count)
14
15 # top k words per document
16 K = 3
17 for doc_id, word_counts in doc_word_counts.items():
18     # sorting words by frequency in desc, then alphabetically
19     top_words = sorted(word_counts.items(), key=lambda x: (-x[1], x[0]))[:K]
20     for word, count in top_words:
21         print("{} , {} , {}".format(doc_id, word, count))

```

```

→ hind git:(main) ✘ docker cp ./input2.txt c64c820cf2cd:.
Successfully copied 3.58kB to c64c820cf2cd:.
→ hind git:(main) ✘ docker cp ./mapper2.py c64c820cf2cd:.
Successfully copied 2.05kB to c64c820cf2cd:.
→ hind git:(main) ✘ docker cp ./reducer2.py c64c820cf2cd:.
Successfully copied 2.56kB to c64c820cf2cd:.

```

```

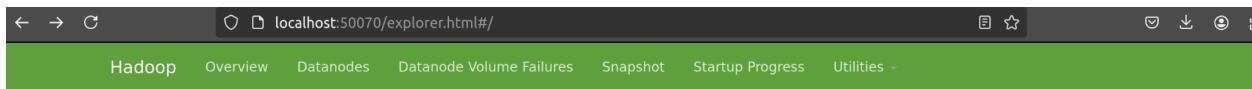
→ hind git:(main) ✘ docker exec -it c64 bash
root@c64c820cf2cd:~/# hadoop jar /opt/hadoop/2.8.0/share/hadoop/tools/lib/hadoop-streaming-2.8.0.jar -files mapper2.py,reducer2.py -mapper mapper2.py -reducer reducer2.py -input /input2.txt -output /outputtwo
24/12/03 14:48:30 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.session-id
24/12/03 14:48:30 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessionId=
24/12/03 14:48:30 INFO jvm.JvmMetrics: Cannot initialize JVM Metrics with processName=JobTracker, sessionId= - already initialized
24/12/03 14:48:30 INFO mapred.FileInputFormat: Total input files to process : 1
24/12/03 14:48:30 INFO mapreduce.JobSubmitter: number of splits:1
24/12/03 14:48:30 INFO mapreduce.Job: Submitting tokens for job: job_local1869274592_0001
24/12/03 14:48:30 INFO mapred.LocalDistributedCacheManager: Localized file:/mapper2.py as file:/tmp/hadoop-root/mapred/local/1733237310761/mapper2.py
24/12/03 14:48:30 INFO mapred.LocalDistributedCacheManager: Localized file:/reducer2.py as file:/tmp/hadoop-root/mapred/local/1733237310762/reducer2.py
24/12/03 14:48:30 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
24/12/03 14:48:30 INFO mapred.LocalJobRunner: OutputCommitter set in config null
24/12/03 14:48:30 INFO mapreduce.Job: Running job: job_local1869274592_0001
24/12/03 14:48:30 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCommitter
24/12/03 14:48:30 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
24/12/03 14:48:30 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
24/12/03 14:48:30 INFO mapred.LocalJobRunner: Waiting for map tasks
24/12/03 14:48:30 INFO mapred.LocalJobRunner: Starting task: attempt_local1869274592_0001_m_000000_0
24/12/03 14:48:30 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
24/12/03 14:48:30 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
24/12/03 14:48:30 INFO mapred.Task: Using ResourceCalculatorProcessTree : []
24/12/03 14:48:30 INFO mapred.MapTask: Processing split: hdfs://namenode:8020/input2.txt:0+1541
24/12/03 14:48:30 INFO mapred.MapTask: numReduceTasks: 1
24/12/03 14:48:31 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
24/12/03 14:48:31 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
24/12/03 14:48:31 INFO mapred.MapTask: soft limit at 83886080
24/12/03 14:48:31 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
24/12/03 14:48:31 INFO mapred.MapTask: kvstart = 26214396; length = 6553600
24/12/03 14:48:31 INFO mapred.MapTask: Map output collector class = org.apache.hadoop.mapred.MapTask$MapOutputBuffer
24/12/03 14:48:31 INFO streaming.PipeMapRed: PipeMapRed exec [./mapper2.py]
24/12/03 14:48:31 INFO Configuration.deprecation: mapred.work.output.dir is deprecated. Instead, use mapreduce.task.output.dir
24/12/03 14:48:31 INFO Configuration.deprecation: map.input.start is deprecated. Instead, use mapreduce.map.input.start
24/12/03 14:48:31 INFO Configuration.deprecation: mapred.task.is.map is deprecated. Instead, use mapreduce.task.ismap
24/12/03 14:48:31 INFO Configuration.deprecation: mapred.task.id is deprecated. Instead, use mapreduce.task.attempt.id

```

```

24/12/03 14:48:31 INFO mapreduce.Job: map 100% reduce 100%
24/12/03 14:48:31 INFO mapreduce.Job: Job job_local1869274592_0001 completed successfully
24/12/03 14:48:31 INFO mapreduce.Job: Counters: 35
    File System Counters
        FILE: Number of bytes read=276878
        FILE: Number of bytes written=942325
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=3082
        HDFS: Number of bytes written=478
        HDFS: Number of read operations=13
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=4
    Map-Reduce Framework
        Map input records=10
        Map output records=192
        Map output bytes=2855
        Map output materialized bytes=3245
        Input split bytes=83
        Combine input records=0
        Combine output records=0
        Reduce input groups=164
        Reduce shuffle bytes=3245
        Reduce input records=192
        Reduce output records=30
        Spilled Records=384
        Shuffled Maps =1
        Failed Shuffles=0
        Merged Map outputs=1
        GC time elapsed (ns)=4
        Total committed heap usage (bytes)=709361664
    Shuffle Errors
        BAD_ID=0
        CONNECTION=0
        IO_ERROR=0
        WRONG_LENGTH=0
        WRONG_MAP=0
        WRONG_REDUCE=0
    File Input Format Counters
        Bytes Read=1541
    File Output Format Counters
        Bytes Written=478
24/12/03 14:48:31 INFO streaming.StreamJob: Output directory: /outputtwo

```



## Browse Directory

/									<input type="button" value="Go!"/>	<input type="button" value="New"/>
Show <input type="text" value="25"/> entries		Search: <input type="text"/>								
Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name			
-rw-r--r--	root	supergroup	1.93 KB	Dec 03 13:19	3	128 MB	input1.txt	<input type="button" value="Edit"/>		
-rw-r--r--	root	supergroup	1.5 KB	Dec 03 17:28	3	128 MB	input2.txt	<input type="button" value="Edit"/>		
drwxr-xr-x	root	supergroup	0 B	Dec 03 18:18	0	0 B	outputtwo	<input type="button" value="Edit"/>		
drwxr-xr-x	root	supergroup	0 B	Dec 03 13:24	0	0 B	user	<input type="button" value="Edit"/>		

Showing 1 to 4 of 4 entries

Hadoop, 2017.

localhost:50070/explorer.html#/outputtwo

Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities

## Browse Directory

/outputtwo

Show 25 entries Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	root	supergroup	0 B	Dec 03 18:18	3	128 MB	_SUCCESS
-rw-r--r--	root	supergroup	478 B	Dec 03 18:18	3	128 MB	part-00000

Showing 1 to 2 of 2 entries

Previous 1 Next

Hadoop, 2017.

### Output (part-00000) :

```

1 doc8,and,1
2 doc8,architectural,1
3 doc8,designs,1
4 doc9,analytics,2
5 doc9,data,2
6 doc9,and,1
7 doc2,is,3
8 doc2,sustainability,3
9 doc2,environmental,1
10 doc3,artificial,2
11 doc3,automation,2
12 doc3,intelligence,2
13 doc1,in,3
14 doc1,innovation,3
15 doc1,technology,3
16 doc6,digital,2
17 doc6,education,2
18 doc6,and,1
19 doc7,healthcare,2
20 doc7,in,2
21 doc7,and,1
22 doc4,blockchain,2
23 doc4,security,2
24 doc4,and,1
25 doc5,energy,5
26 doc5,and,2
27 doc5,a,1
28 doc10,and,2
29 doc10,a,1
30 doc10,accountability,1

```

توضیحات بخش سوم:  
کد :mapper2.py

این کد برای استخراج اطلاعات از ورودی و ارسال آن به کد reducer نوشته شده است. مراحل:

استخراج شناسه داک (document ID) و محتوا:  
از یک الگو (pattern) استفاده می‌شود که شناسه داک و محتوای آن را استخراج کند. pattern به صورت `(.+)(\w+)` است که:  
`(\w+)` شناسه داک (که شامل حروف و اعداد است) را استخراج می‌کند  
`(.+)` محتوای داک را که ممکن است شامل هر گونه داده‌ای باشد، استخراج می‌کند

پردازش محتوا:  
متن محتوا به حروف کوچک تبدیل می‌شود (`content.lower()`). سپس متن به کلمات مختلف تقسیم می‌شود (`split()`)

خروجی برای هر کلمه:  
برای هر کلمه در محتوا، یک مقدار 1 برای آن کلمه و داک مربوطه چاپ می‌شود  
فرمت خروجی به این صورت است: `1 word,doc_id`. این اطلاعات به `reducer` فرستاده می‌شود

:`reducer2.py`

در این کد اطلاعات ارسال شده از `mapper` پردازش می‌شود تا کلمات پرکاربردترین در هر داک به دست آید

ساخت دیکشنری برای شمارش کلمات:  
یک دیکشنری دو سطحی (`doc_word_counts`) ایجاد می‌شود:  
سطح اول، شناسه داک (`doc_id`) را ذخیره می‌کند  
سطح دوم، کلمات (`word`) و تعداد تکرار آنها در داک مربوطه را ذخیره می‌کند

جمع‌آوری داده‌ها از `Mapper`:  
داده‌هایی که از `mapper` ارسال شده است، دریافت می‌شود. داده‌ها از هم تفکیک شده و برای هر کلمه در هر داک، تعداد تکرار آن ثبت می‌شود

انتخاب K کلمه پرکاربردتر در هر داک:  
کلمات به ترتیب تعداد تکرار آنها (در اولویت) و به ترتیب الفبایی (در صورت نساوی تعداد تکرار) مرتب می‌شوند. سپس تنها K کلمه اول انتخاب می‌شود (که در اینجا  $K=3$  است)

خروجی:  
برای هر `doc` سه کلمه پرکاربرد با تعداد تکرار آن‌ها چاپ می‌شود. فرمت خروجی به صورت `doc_id,word,count` است.

توضیح خروجی:  
مثال:  
در داک `doc8`, کلمه `and` یک بار ظاهر شده است  
در داک `doc5`, کلمه `energy` پنج بار ظاهر شده است  
در داک `doc1`, کلمه `in` سه بار ظاهر شده است