

گزارش تمرین سوم درس رایانش ابری (آشنایی عملیاتی با Hadoop)

محمدجواد زندیه (۹۸۳۱۰۳۲)

گام اول:

۱. با استفاده از دستور docker-compose up ابتدا image های لازم را pull کرده و سپس container ها را می‌سازیم.

```
~/Desktop/files/university-courses/cloud-computing/home-works/hw3/HM3 | base | 12:06:29
docker-compose up

~/Desktop/files/university-courses/cloud-computing/home-works/hw3/HM3 | base | 11:59:54
docker ps -a
CONTAINER ID   IMAGE                                     COMMAND                  CREATED        STATUS              PORTS
de4d472d22ad   arminzolfagharid/hadoop-nodemanager:v2-hadoop3.2.1   "/entrypoint.sh /run..." 12 minutes ago Up 5 minutes (healthy) 8042/tcp
07e37c7e042f   arminzolfagharid/hadoop-datanode:v2-hadoop3.2.1       "/entrypoint.sh /run..." 12 minutes ago Up 5 minutes (unhealthy) 9864/tcp
3dce95bfec7b   arminzolfagharid/hadoop-namenode:v2-hadoop3.2.1       "/entrypoint.sh /run..." 12 minutes ago Up 5 minutes (healthy) 0.0.0.0:9000->9000/tcp, 0.0.0.0:9870->9870/tcp
aa04938685b2   arminzolfagharid/hadoop-resourcemanager:v2-hadoop3.2.1 "/entrypoint.sh /run..." 12 minutes ago Up 10 seconds (health: starting) 8088/tcp
e064327de584   arminzolfagharid/hadoop-historyserver:v2-hadoop3.2.1   "/entrypoint.sh /run..." 12 minutes ago Up 5 minutes (healthy) 8188/tcp
```

۲. وظیفه هریک از container ها در hadoop

کانتینر NameNode:

- جز اصلی HDFS است که به عنوان سرور مرکزی ابر داده عمل میکند و مدیریت namespace مربوط به file system و نگاشت بلاک اطلاعات در کلاستر Hadoop را انجام میدهد.
- **مدیریت Namespace:** فضای نام file system را مدیریت می کند که شامل دایرکتوری ها، فایل ها و ویژگی های آنهاست. همچنین ساختار سلسله مراتبی file system را حفظ می کند و مکان و سازماندهی بلوک های داده را در فایل ها پیگیری می کند.
- **ذخیره سازی متادیتا:** ساختار دایرکتوری، مجوزهای فایل، لیست های کنترل دسترسی و مکان های بلوک را ذخیره می کند. این کار برای عملکرد مناسب HDFS بسیار مهم است و برای دسترسی سریع در حافظه ذخیره می شود.
- **نگاشت بلاک ها:** نگاشت بین فایل ها و بلوک های داده ای که آن فایل ها را تشکیل می دهند را حفظ می کند. ردیابی بلوک های داده مرتبط با هر فایل و DataNodes هایی که این بلوک ها در آن ذخیره می شوند را پیگیری می کند. این اطلاعات به کلاینت و DataNode ها اجازه می دهد تا به طور موثر به داده های مورد نیاز دسترسی پیدا کرده و آنها را بازیابی کنند.
- **تعامل با کلاینت:** به عنوان نقطه تماس اولیه برای کلاینت هایی است که به HDFS دسترسی دارند. کلاینت ها با NameNode ارتباط برقرار می کنند تا عملیات مختلفی مانند ایجاد فایل، حذف، تغییر نام و ناوبری سیستم فایل را انجام دهند.
- **کپی کردن بلاک ها و بازیابی دیتا:** مسئول مدیریت تکرار بلوک در HDFS است. سلامت و در دسترس بودن DataNode ها را در کلاستر نظارت می کند و اطمینان حاصل می کند که تعداد مورد نیاز کپی داده برای هر بلوک حفظ می شود. اگر یک DataNode از کار بیفتد یا یک نسخه از بین برود، NameNode فرایند تکرار یا بازیابی را برای حفظ یکپارچگی داده ها و تحمل خطا آغاز می کند.

کانتینر DataNode:

- یکی از اجزای HDFS است که بر روی هر یک از Node ها در کلاستر Hadoop اجرا میشود و وظیفه ذخیره و مدیریت منابع روی دیسک محلی را برعهده دارد.
- **ذخیره سازی داده ها:** داده ها را در سیستم دیسک محلی Node ذخیره می کند. بلوک های داده را از کلاینت یا دیگر DataNode ها دریافت می کند و آنها را روی دیسک محلی خود می نویسد. همچنین بلوک های داده را در صورت درخواست بازیابی می کند و آنها را به کلاینت ها یا دیگر DataNode ها ارائه می دهد.
- **رونوشت داده ها (data replication):** در فرآیند replication در HDFS شرکت می کند. درخواست های تکرار را از NameNode (سرور مرکزی در HDFS) دریافت می کند و کپی های اضافی از بلوک های داده را با انتقال آنها به DataNode دیگر ایجاد می کند. این تکرار دوام داده ها و تحمل خطا را در صورت خرابی Node تضمین می کند.
- **مدیریت سلامت:** به صورت دوره ای Heartbeat را به NameNode ارسال می کند تا در دسترس بودن و وضعیت سلامت آن را نشان دهد.
- **گزارش های مربوط به بلاک:** به طور منظم گزارش های بلوکی را به NameNode ارسال می کند و اطلاعاتی در مورد بلوک های داده ای که در حال حاضر ذخیره می کند ارائه می دهد. این به NameNode کمک می کند تا تصویری به روز از توزیع داده ها در سراسر کلاستر حفظ کند و در مورد تخصیص و کپی کردن بلوک تصمیم گیری کند.
- **ریکاوری بلاک:** در صورت خرابی یا در دسترس نبودن یک DataNode آنگاه NameNode از گزارش های بلوک سایر DataNode ها برای شناسایی بلوک های گمشده یا کم تکرار شده استفاده می کند. سپس با ایجاد کپی های جدید از بلوک های گمشده در DataNode های مختلف، فرآیند بازیابی را هماهنگ می کند.
- **یکپارچگی داده:** یکپارچگی بلوک های داده ای را که دریافت یا بازیابی می کند تأیید می کند. checksum را برای بلوک ها محاسبه می کند و آنها را با checksum ذخیره شده توسط NameNode مقایسه می کند. اگر عدم تطابق وجود داشته باشد، نشان دهنده خرابی داده ها است و DataNode این را به NameNode گزارش می دهد.

کانتینر HistoryServer:

- هیستوری مربوط به کارهای تکمیل شده MapReduce و سایر برنامه های Hadoop را ذخیره و نمایش می کند. این به کلاینت ها اجازه می دهد تا به جزئیات کارهای قبلی از جمله گزارش ها، آمار و تشخیص خطاها دسترسی داشته باشند و آن ها را تجزیه و تحلیل کنند.
- **ذخیره سازی هیستوری کارها:** زمانی که یک برنامه MapReduce تکمیل میشود، اطلاعات آن از جمله مسیرهای ورودی و خروجی، وضعیت کار انجام شده و پیگیری ها در آن ذخیره میشوند.
- **بازیابی اطلاعات کارهای انجام شده:** رابطی را برای کاربران فراهم میکند که بتوانند اطاعات کارهای انجام شده را مشاهده کنند که در مواقع عیبیابی بسیار مناسب هستند.

- دارای **web-based UI**: برای مشاهده اطاعات کارها یک رابط کاربری مبتنی بر web دارد که تعامل را راحت تر میکند.
- **حفظ و پاکسازی کارها**: برای مدیریت بهتر حافظه، میتوان مدت و تعداد کارهای تکمیل شده را برای ذخیره در حافظه مشخص کرد.
- **امنیت و کنترل دسترسی**: برای حفاظت از هیستوری کارهای انجام شده، با زیرساخت‌های امنیتی Hadoop در ارتباط است تا بتواند دسترسی‌های غیرمجاز را کنترل کند.

کانتینر ResourceManager:

- جز اصلی YARN است که وظیفه مدیریت و تخصیص منابع به برنامه‌هایی که در کلاستر Hadoop درحال اجرا هستند را دارد.
- **تخصیص منابع**: منابع موجود در کلاستر را مدیریت می کند و آنها را به برنامه های مختلف اختصاص می دهد. این برنامه منابع (مانند CPU، حافظه و دیسک) موجود در هر Node را ردیابی می کند و تصمیم می گیرد که چگونه این منابع را بین برنامه ها بر اساس نیاز آنها توزیع کند.
- **برنامه ریزی برنامه ها**: زمان بندی برنامه های کاربردی در کلاستر را مدیریت می کند. درخواست های برنامه را دریافت می کند و بر اساس منابع موجود و قوانین زمان بندی تصمیم میگیرد که چه زمانی و کجا آنها را اجرا کند. تخصیص عادلانه و کارآمد منابع به برنامه ها را با در نظر گرفتن عواملی مانند اولویت، تقاضای منابع و محدودیت های تعریف شده توسط کاربر تضمین می کند.
- **مدیریت کانتینر**: کانتینرها را مدیریت میکند. از طرف برنامه ها کانتینرهایی ایجاد می کند، منابع را به آنها اختصاص می دهد و پیشرفت و وضعیت آنها را ردیابی می کند. بر سلامت کانتینرها نظارت می کند و در صورت خرابی یا محدودیت منابع اقدام می کند.
- **مانیتورینگ**: اطلاعاتی در مورد وضعیت و پیشرفت برنامه ها و کانتینرهای در حال اجرا در کلاستر جمع آوری می کند. این یک نمای زمان واقعی از استفاده از منابع کلاستر، ظرفیت موجود و وضعیت برنامه های در حال اجرا را حفظ می کند. این اطلاعات برای نظارت، گزارش و تصمیم گیری آگاهانه در مورد تخصیص منابع و زمان بندی استفاده می شود.
- **تحمل خطا (fault tolerance)**: تحمل خطا را در کلاستر با شناسایی و مدیریت خرابی ها تضمین می کند. با NodeManagers و NameNode هماهنگ می شود تا خرابی های Node را مدیریت کند، کانتینرها را مجدداً تخصیص دهد و پایداری کلاستر را حفظ کند.
- **امنیت**: سیاست های امنیتی و محدودیت های منابع تعریف شده برای کلاستر را اعمال می کند. مجوزهای دسترسی کاربر را تأیید می کند، سهمیه منابع را اعمال می کند و اطمینان می دهد که برنامه ها به قوانین مشخص شده پایبند هستند. این یک محیط امن و کنترل شده برای اجرای برنامه های کاربردی در کلاستر Hadoop فراهم می کند.

کانتینر NodeManager:

- یکی از اجزای YARN است که بر روی هر یک از Node ها در کلاستر Hadoop اجرا میشود تا مدیریت منابع و زمانبندی کارها در آن Node را مدیریت کند.
- **مدیریت منابع:** تخصیص منابع سیستم مانند CPU، حافظه و دیسک را در هر Node از کلاستر Hadoop مدیریت می کند. منابع موجود در Node را ردیابی می کند و این اطلاعات را به ResourceManager گزارش می دهد.
- **مدیریت کانتینر:** مسئولیت مدیریت کانتینرها را بر عهده دارد. درخواست های تخصیص کانتینر را از ResourceManager دریافت میکند و کانتینرها را روی Node های مربوطه راه اندازی می کند.
- **مانیتورینگ تسکها:** بر اجرای تسکها درون کانتینرها نظارت می کند. استفاده از منابع و وضعیت تسکهای در حال اجرا بر روی Node را ردیابی می کند و این اطلاعات را در اختیار ResourceManager قرار می دهد. این به ResourceManager اجازه می دهد تا تصمیمات آگاهانه ای در مورد تخصیص منابع و زمان بندی اتخاذ کند.
- **بهروزرسانی های Heartbeat و Status:** به صورت دوره ای Heartbeat را به ResourceManager ارسال می کند تا یک بهروزرسانی در مورد سلامت و در دسترس بودن Node ارائه دهد.
- **امنیت و جداسازی کانتینر:** با اجرای ایزوله کانتینر در امنیت موثر است. این تضمین می کند که تسکهای در حال اجرا در کانتینرها از یکدیگر جدا شده و دسترسی محدودی به منابع سیستم دارند.

۳. بررسی صحت نقش هریک از کانتینرها با استفاده از دستور jps: مشاهده میشود که process هر کانتینر به درستی در حال اجرا است.

```

# ~/Desktop/files/university-courses/cloud-computing/home-works/hw3/HW3
# docker exec -it nodemanager sh
# jps
1389 Jps
911 NodeManager
# exit

# ~/Desktop/files/university-courses/cloud-computing/home-works/hw3/HW3
# docker exec -it datanode sh
# jps
886 DataNode
1190 Jps
# exit

# ~/Desktop/files/university-courses/cloud-computing/home-works/hw3/HW3
# docker exec -it namenode sh
# jps
1320 Jps
874 NameNode
# exit

# ~/Desktop/files/university-courses/cloud-computing/home-works/hw3/HW3
# docker exec -it resourcemanager sh
# jps
1110 Jps
856 ResourceManager
# exit

# ~/Desktop/files/university-courses/cloud-computing/home-works/hw3/HW3
# docker exec -it historyserver sh
# jps
929 ApplicationHistoryServer
1337 Jps
# exit

```

۴. مشاهده NameNode به وسیله WebUI در localhost:9870 و file system آن:

The screenshot shows the 'Overview' page for the NameNode 'namenode:9000' (active). The page has a green navigation bar with links: Hadoop, Overview, Datanodes, Datanode Volume Failures, Snapshot, Startup Progress, and Utilities. The main content area is dark grey and contains a table with the following information:

Started:	Mon Jun 05 11:54:19 +0330 2023
Version:	3.2.1, rb3cbbb467e22ea829b3808f4b7b01d07e0bf3842
Compiled:	Tue Sep 10 20:26:00 +0430 2019 by rohitsharmaks from branch-3.2.1
Cluster ID:	CID-adeb84cd-ba3e-4da5-9eac-e300b3254a93
Block Pool ID:	BP-384747812-172.18.0.4-1685953064138

Below the table is a 'Summary' section with the following text:

Security is off.

Safe mode is ON. The reported blocks 0 needs additional 4 blocks to reach the threshold 0.9990 of total blocks 5. The minimum number of live datanodes is not required. Safe mode will be turned off automatically once the thresholds have been reached.

12 files and directories, 5 blocks (5 replicated blocks, 0 erasure coded block groups) = 17 total filesystem object(s).

Heap Memory used 117.21 MB of 165.5 MB Heap Memory. Max Heap Memory is 875 MB.

Non Heap Memory used 54.16 MB of 55.69 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.

At the bottom, there is a table for capacity:

Configured Capacity:	0 B
Configured Remote Capacity:	0 B

The screenshot shows the 'Browse Directory' page for the NameNode 'namenode:9000' (active). The page has a green navigation bar with links: Hadoop, Overview, Datanodes, Datanode Volume Failures, Snapshot, Startup Progress, and Utilities. The main content area is dark grey and contains a search bar with the text '/' and a 'Go!' button. Below the search bar is a table with the following information:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	root	supergroup	0 B	Jun 05 11:49	0	0 B	rmstate

Below the table is a 'Showing 1 to 1 of 1 entries' message and a pagination bar with 'Previous', '1', and 'Next' buttons.

بررسی صحت گام اول با استفاده از فایل جدید:

بدلیل مشکلاتی که docker-compose فایل گام اول داشت، از docker-compose موجود در این ریپازیتوری برای ادامه کار استفاده شده

است (<https://github.com/wxw-matt/docker-hadoop>).

```

$ docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
49743fe9ca67   wxwmatt/hadoop-namenode:2.1.1-hadoop3.3.1-java8   "/entrypoint.sh /run..." 21 minutes ago Up 21 minutes (healthy)   0.0.0.0:9000->9000/tcp, 0.0.0.0:9870->9870/tcp
22556dbe2f63   wxwmatt/hadoop-datanode:2.1.1-hadoop3.3.1-java8   "/entrypoint.sh /run..." 21 minutes ago Up 21 minutes (healthy)   0.0.0.0:9864->9864/tcp
69f752841477   wxwmatt/hadoop-nodemanager:2.1.1-hadoop3.3.1-java8 "/entrypoint.sh /run..." 21 minutes ago Up 21 minutes (healthy)   0.0.0.0:8042->8042/tcp
2d25f774f443   postgres:latest                                "docker-entrypoint.s..." 21 minutes ago Up 21 minutes             0.0.0.0:5432->5432/tcp
5b039ad5835c   wxwmatt/hadoop-historyserver:2.1.1-hadoop3.3.1-java8 "/entrypoint.sh /run..." 21 minutes ago Up 21 minutes (healthy)   0.0.0.0:8188->8188/tcp
73a9947fc58f   wxwmatt/hadoop-resourcemanager:2.1.1-hadoop3.3.1-java8 "/entrypoint.sh /run..." 21 minutes ago Up 21 minutes (healthy)   0.0.0.0:8088->8088/tcp

```

```

$ docker exec -it namenode sh
# jps
406 NameNode
1467 Jps
# exit

$ docker exec -it datanode sh
# jps
817 Jps
357 DataNode
# exit

$ docker exec -it resourcemanager sh
# jps
972 Jps
351 ResourceManager
# exit

$ docker exec -it nodemanager sh
# jps
355 NodeManager
820 Jps
# exit

$ docker exec -it historyserver sh
# jps
787 Jps
362 ApplicationHistoryServer
# exit

```

localhost:9870/explorer.html#/

Courses
Webmail
Translate
WhatsApp
Telegram Web
Instagram
Feed | LinkedIn
Overleaf
GitHub
GitLab
Nima
Discord
GNN
ZTM: PyTorch
DeepLearning.AI

Hadoop
Overview
Datanodes
Datanode Volume Failures
Snapshot
Startup Progress
Utilities

Browse Directory

/
Go!

Show 25 entries
Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	root	supergroup	0 B	Jun 06 10:38	0	0 B	rmstate

localhost:9870/dfshealth.html#tab-overview

LmsHome
Courses
Webmail
Translate
WhatsApp
Telegram Web
Instagram
Feed | LinkedIn
Overleaf
GitHub
GitLab
Nima
Discord
GNN
ZTM: PyTorch
DeepLearning.AI

Hadoop
Overview
Datanodes
Datanode Volume Failures
Snapshot
Startup Progress
Utilities

Overview 'namenode:9000' (✓active)

Started:	Tue Jun 06 10:38:14 +0330 2023
Version:	3.3.1, ra3b9c37a397ad4188041dd80621bdeefc46885f2
Compiled:	Tue Jun 15 15:21:00 +0430 2021 by ubuntu from (HEAD detached at release-3.3.1-RC3)
Cluster ID:	CID-fc3d03c0-4071-4c2e-a021-5792fc7b280a
Block Pool ID:	BP-237335111-172.18.0.4-1686035293244

Namenode: http://localhost:9870

Datanode: http://localhost:9864

Resourcemanager: http://localhost:8088

Nodemanager: http://localhost:8042

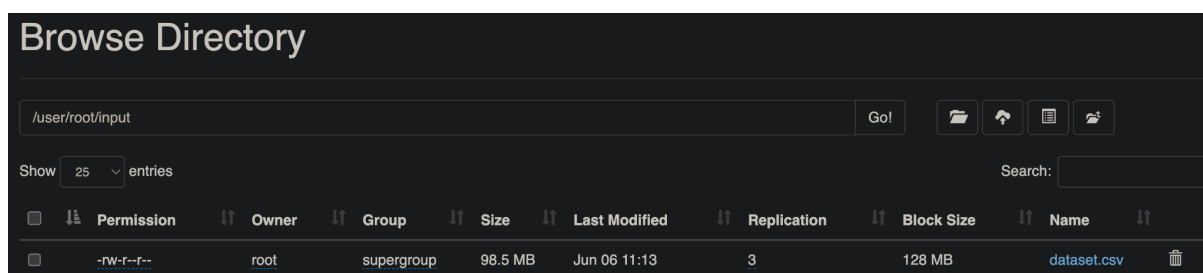
Historyserver: http://localhost:8188

گام دوم و سوم:

۱. ابتدا باید فایل dataset.csv را در مسیر /docker-hadoop/jobs/data/ قرار می‌دهیم (قرار دادن فایل در shared volume کانتینر (NameNode).

۲. برای ساخت دایرکتوری و قراردادن دیتاست بر روی HDFS از دستورات زیر استفاده می‌کنیم:

```
~/De/f/u/cloud-computing/home-works/hw3/9831832_hw3/docker-hadoop 1 x | base 11:11:56
./hdfs dfs -mkdir -p /user/root/input
~/De/f/u/cloud-computing/home-works/hw3/9831832_hw3/docker-hadoop ✓ | base 11:12:21
./hdfs dfs -copyFromLocal -f /app/data/dataset.csv /user/root/input
~/De/f/u/cloud-computing/home-works/hw3/9831832_hw3/docker-hadoop ✓ | base 11:13:10
./hdfs dfs -ls /user/root/input
Found 1 items
-rw-r--r-- 3 root supergroup 103280184 2023-06-06 07:43 /user/root/input/dataset.csv
```



برای آنکه بتوانیم برنامه‌های MapReduce خود را به زبان پایتون بنویسیم، باید روی تمام کانتینرها (بجز postgres) پایتون را نصب کنیم. در نتیجه دو دستور زیر را در تمام کانتینرها اجرا می‌کنیم:

apt-get update & apt-get install python3 & apt-get install python3-pip

```
# python3 --version
Python 3.7.3
# pip3 --version
pip 18.1 from /usr/lib/python3/dist-packages/pip (python 3.7)
```

۳. اجرای برنامه‌های map reduce بر روی hadoop:

برای اجرای job ها بر روی هادوپ، فایل‌های مربوط به mapper و reducer را برای هر سه بخش در shared volume قرار می‌دهیم (/docker-hadoop/jobs/jars/). سپس در کانتینر namenode دستور اجرای job به صورت زیر اجرا می‌کنیم:

```
hadoop jar /opt/hadoop-3.3.1/share/hadoop/tools/lib/hadoop-streaming-3.3.1.jar \
  -file /app/jars/mapper.py \
  -mapper "python3 mapper.py" \
  -file /app/jars/reducer.py \
  -reducer "python3 reducer.py" \
  -input /user/root/input/dataset.csv \
  -output /user/root/output/
```

دقت شود که محل output که مشخص کرده‌ایم باید خالی باشد در غیر این صورت تسک هادوپ fail می‌شود.

۴. خروجی برنامه‌های map reduce:

برنامه اول:

```
# hdfs dfs -cat /user/root/output/part1/part-00000
Both Candidate 535871 158881 28456 20053 14495
Donald Trump 362583 120214 25877 19339 16291
Joe Biden 566176 221989 20740 18632 11760
```

برنامه دوم:

```
# hdfs dfs -cat /user/root/output/part2/part-00000
california 0.3681267474370923 0.27539608574091334 0.35647716682199443 2146
florida 0.3539990534784666 0.3610979649787033 0.2849029815428301 2113
new york 0.3799675587996756 0.2871046228710462 0.3329278183292782 2466
texas 0.3590935502614759 0.35095874491574663 0.28994770482277743 1721
```

برنامه سوم:

```
# hdfs dfs -cat /user/root/output/part3/part-00000
california 0.3682403433476395 0.28412017167381975 0.34763948497854075 2330
new york 0.38531603643389456 0.2740822522771184 0.34060171128898703 3623
```

کامندهای لازم برای اجرای کارها در my-command.sh قرار دارند.

همانطور که مشخص است، نتایج در مسیر /user/root/output ذخیره شده اند:

Browse Directory

/user/root/output Go! 📁 📄 📂 📅

Show 25 entries Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	<input type="checkbox"/>
<input type="checkbox"/>	drwxr-xr-x	root	supergroup	0 B	Jun 10 16:45	0	0 B	part1	<input type="checkbox"/>
<input type="checkbox"/>	drwxr-xr-x	root	supergroup	0 B	Jun 10 16:50	0	0 B	part2	<input type="checkbox"/>
<input type="checkbox"/>	drwxr-xr-x	root	supergroup	0 B	Jun 10 16:51	0	0 B	part3	<input type="checkbox"/>

Browse Directory





/user/root/output/part1 Go! 📁 📄 📂 📅

Show 25 entries Search:



<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	<input type="checkbox"/>
<input type="checkbox"/>	-rw-r--r--	root	supergroup	0 B	Jun 10 16:45	3	128 MB	SUCCESS	<input type="checkbox"/>
<input type="checkbox"/>	-rw-r--r--	root	supergroup	137 B	Jun 10 16:45	3	128 MB	part-00000	<input type="checkbox"/>

Showing 1 to 2 of 2 entries Previous 1 Next

Browse Directory





/user/root/output/part2 Go!    

Show 25 entries Search:



<input type="checkbox"/>	↓ Permission	↑ Owner	↑ Group	↓ Size	↓ Last Modified	↑ Replication	↓ Block Size	↑ Name	↓
<input type="checkbox"/>	-rw-r--r--	root	supergroup	0 B	Jun 10 16:50	3	128 MB	_SUCCESS	
<input type="checkbox"/>	-rw-r--r--	root	supergroup	290 B	Jun 10 16:50	3	128 MB	part-00000	

Showing 1 to 2 of 2 entries Previous 1 Next

Browse Directory





/user/root/output/part3 Go!    

Show 25 entries Search:


<input type="checkbox"/>	↓ Permission	↑ Owner	↑ Group	↓ Size	↓ Last Modified	↑ Replication	↓ Block Size	↑ Name	↓
<input type="checkbox"/>	-rw-r--r--	root	supergroup	0 B	Jun 10 16:51	3	128 MB	_SUCCESS	
<input type="checkbox"/>	-rw-r--r--	root	supergroup	150 B	Jun 10 16:51	3	128 MB	part-00000	

Showing 1 to 2 of 2 entries Previous 1 Next

Browse Directory

/user/root/input Go!    

Show 25 entries Search:

<input type="checkbox"/>	↓ Permission	↑ Owner	↑ Group	↓ Size	↓ Last Modified	↑ Replication	↓ Block Size	↑ Name	↓
<input type="checkbox"/>	-rw-r--r--	root	supergroup	98.5 MB	Jun 06 11:13	3	128 MB	dataset.csv	

Showing 1 to 1 of 1 entries Previous 1 Next

دریافت نتایج خروجی:

با استفاده از دستوری زیر نتایج را به داخل namenode و فضای shared منتقل میکنیم

```
# hdfs dfs -get /user/root/output /app/res
# tree /app/res/output
/app/res/output
├── part1
│   ├── _SUCCESS
│   └── part-00000
├── part2
│   ├── _SUCCESS
│   └── part-00000
└── part3
    ├── _SUCCESS
    └── part-00000

3 directories, 6 files
```