|  |
| --- |
| به‌نام خداوند بخشنده و مهربان  **تمرین سوم:** نگاشت-کاهش **استاد:** محمد‌علی نعمت‌بخش  **درس:** پایگاه‌داده پیشرفته دستیار: امیر سرتیپی  **نام و نام‌خانوادگی:** سید عمید اسدالهی مجد  **شماره دانشجویی:** 4003614004 |

* لینک گیت‌هاب: <https://github.com/amidmajd/hadoop-wordCount-exercise>

## مقدمه

در این تمرین برای راه‌اندازی Hadoop با استفاده از داکر انجام شد و کد‌های mapper و reducer با استفاده از زبان پایتون نوشته شده است.

## راهنمای اجرا

ابتدا محفضه‌های داکر را با اجرای دستور docker-compose up -d درون پوشه hadoop-cluster-docker که شامل فایل docker-compose.ymlاست، راه‌‌اندازی می‌کنیم و سپس دستورات ذکر شده در فایل README.md که در پوشه اصلی تمرین قرار دارد را، درون پوشه اصلی تمرین، به ترتیب اجرا می‌نماییم.

## راه‌اندازی هدوپ (Hadoop)

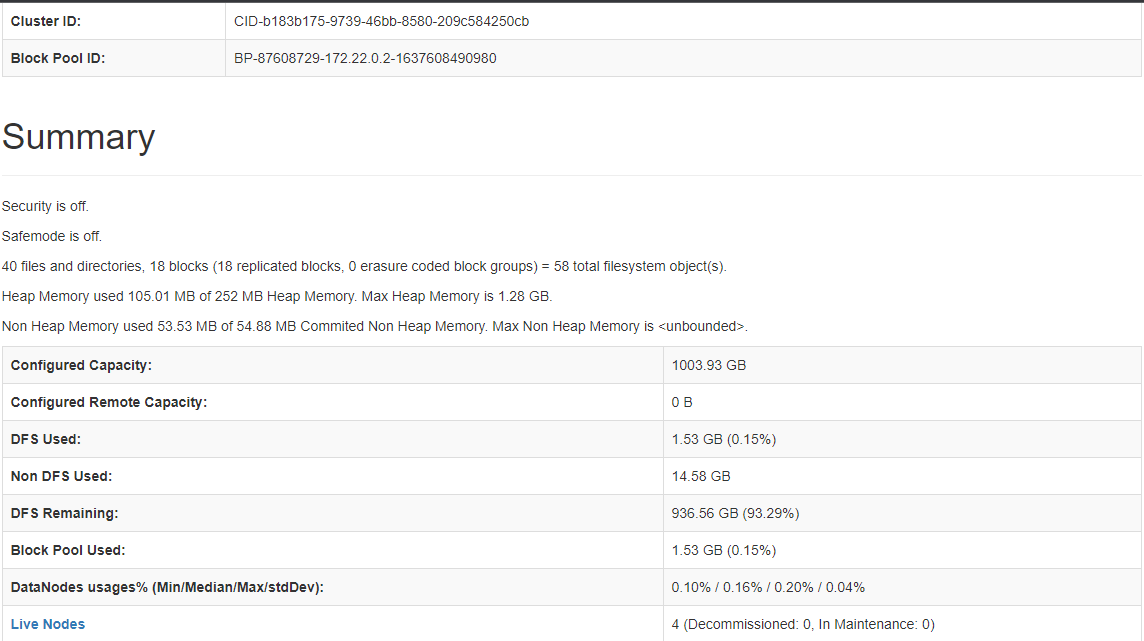
ابتدا داکر را روی سیستم نصب می‌نماییم که از وبسایت داکر برای ویندوز قابل دانلود و نصب است. سپس با ساختن فایل docker-compose.yml که تمام محفضه‌ها (Container) را آدرس‌دهی می‌کند و می‌سازد کار را ادامه می‌دهیم. با اجرای دستور docker-compose up ابتدا با استفاده از Dockerfile های تعریف شده در پوشه‌های مربوط به هر محفضه استفاده docker image مربوطه را ساخته، سپس آن‌ها را با تنظیمات موجود در فایل docker-compose.yml اجرا می‌نماید و محفضه‌های مورد نیاز را می‌سازد.

هر یک از بخش‌های Hadoop یک محفضه مخصوص به خود با پورت جداگانه خود در داکر دارند که عبارتند از:

* Name Node : 9870
* Data Node : 9864
* Node Manager : 9864
* Resource Manager : 8088
* History Server : 8188

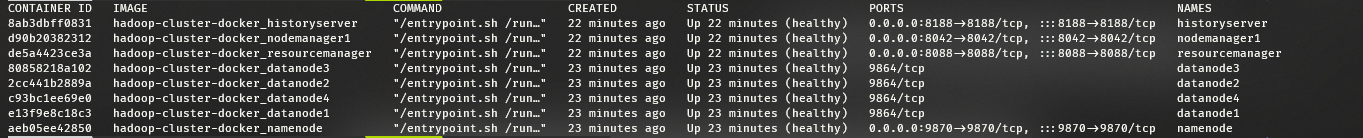
فایل docker-compose.yml طوری تنظیم شده است که ۴ نود داده (data node) داشته باشیم.

آدرس دسترسی به هریک از این محفضه‌ها در فایل README.md که در پوشه مربوط به داکر  
(Hadoop-cluster-docker) قرار دارد، نیز ذکر شده است. رابط کاربری تحت وب هریک از این محفضه‌ها نیز قابل دستری است، به طول مثال با رفتن به آدرس<http://localhost:9870> می‌توان به NameNode دسترسی پیدا کرد. (شکل۱)



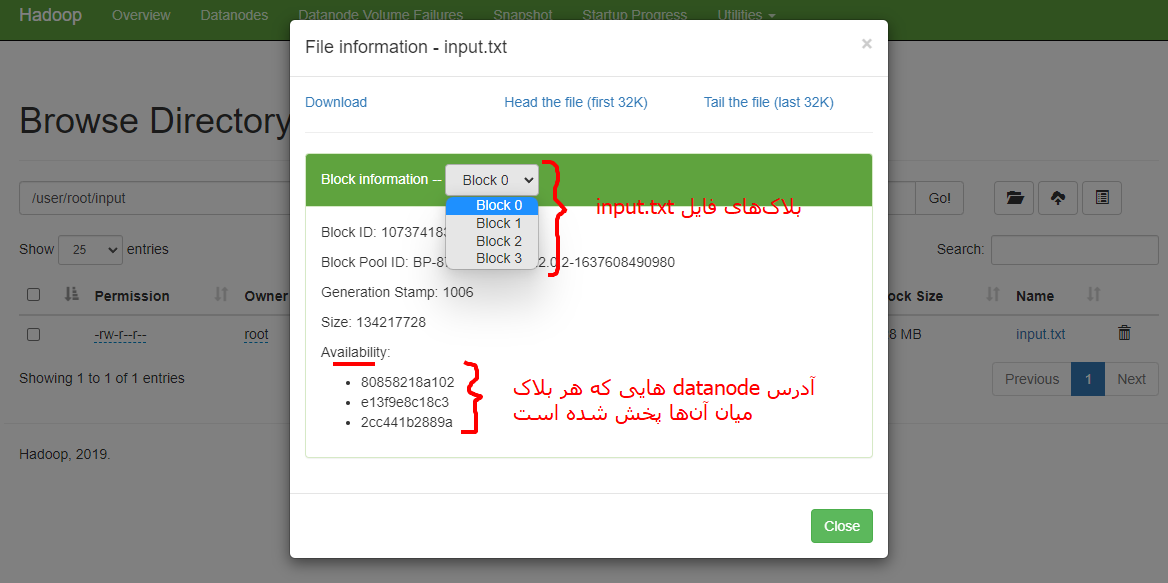
شکل 1: بخش Overview از رابط کاربری تحت وب NameNode

با استفاده از دستور docker exec -it *CONTAINER\_NAME* /bin/bash نیز می‌توان به command line هر یک از این محفضه‌ها دسترسی پیدا کرد. همچنین با استفاده از دستور docker ps می‌توان از لیست محفظه‌ها و تنظیمات مهم آن‌ها را مشاهده کرد و از درستی اجرای هریک از محفضه‌‌ها اطمینان حاصل نمود. (شکل۲)



شکل 2: خروجی دستور docker ps

یکی از بخش‌های کاربردی قابل دسترسی تحت وب NameNode بخش browse the file system است که از منوی utilites قابل دسترسی می‌باشد و می‌توان تمام فایل‌های موجود در HDFS را مشاهده نمود. به طور مثال فایل ورودی که حجمی در حدود ۵۰۰ مگابایت دارد در ۴ بلاک (هر بلاک حداکثر ۱۲۸ مگابایت) ذخیره شده است. این ۴ بلاک میان datanode ها پخش شده‌اند که در شکل ۳ قابل مشاهده است.



شکل 3: مشاهده فایل‌های موجود در HDFS و بلاک‌ها و پخش‌شدگی بلاک‌ها میان datanode ها

## بارگذاری و یا دریافت ورودی و خروجی در HDFS

در شکل ۴ برای بارگذاری کد‌های mapper و reducer موجود در پوشه src و همچنین فایل متنی ورودی بر روی HDFS، ابتدا آ‌ن‌ها را با استفاده از دستور docker cp از سیستم اصلی خودمان به پوشه home محفضه NameNode منتقل می‌کنیم.

سپس با استفاده از دستور docker exec -it namenode /bin/bash به command line محفضه NameNode وارد می‌شویم. حال با استفاده از دستور hdfs dfs یک دایرکتوی در HDFS ایجاد کرده و فایل متنی ورودی را به آن بر روی HDFS منتقل می‌نماییم. (hdfs dfs -put)

برای دریافت خروجی از HDFS از دستور hdfs dfs -get استفاده می‌کنیم و فایل متنی خروجی را به پوشه home محفضه NameNode منتقل می‌کنیم و در انتها با استفاده از دستور docker cp آن را به سیستم اصلی خودمان انتقال می‌دهیم.



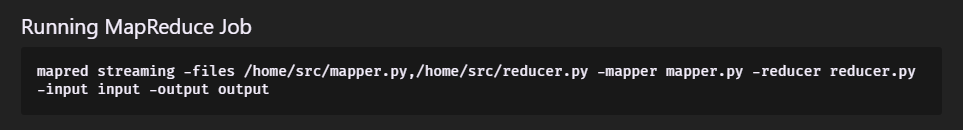
شکل 4: بارگذاری و یا دریافت ورودی و خروجی در HDFS اجرا شده بر روی داکر

## دستور Mapred در Hadoop

این دستور که در NameNode باید اجرا شود چند پارامتر به عنوان ورودی می‌گیرید و عملیات کاهش-نگاشت را با استفاده از mapper و reducer داده شده بر روی فایل متنی داده شده انجام داده و خروجی را در یک فایل متنی در آدرس داده‌شده ذخیره می‌کند. این دستور که در واقع یک فایل اجرایی جاوا می‌باشد تنها در صورتی قابلیت اجرای فایل‌های پایتون را دارد که پایتون در هریک از محفظه‌های داکر مربوط به Hadoop (یا بر روی سیستم اصلی که Hadoop در آن نصب شده است) نصب شده باشد.

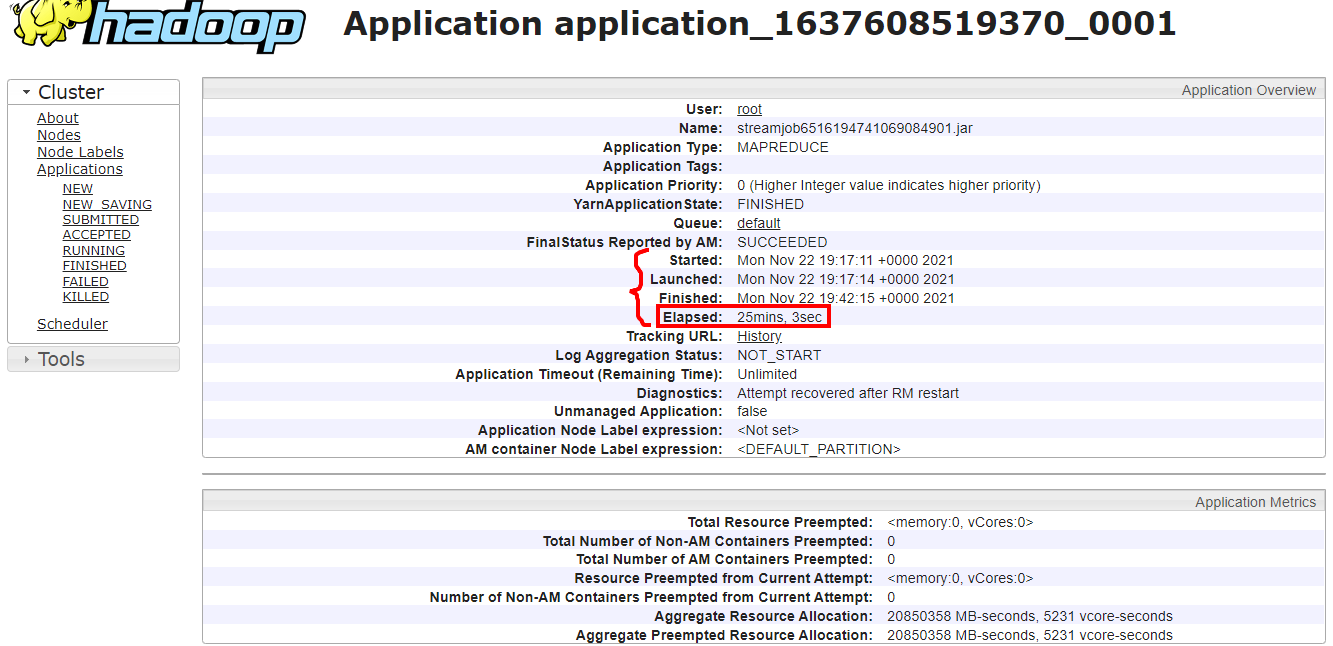
فایل ورودی متنی باید حتما بر روی HDFS قرار داشته باشد و همچنین خروجی نیز بر روی HDFS ذخیره می‌شود.

دستور اجرا در شکل ۵ قابل مشاهده است. در این دستور فایل‌های mapper.py و reducer.py به عنوان نگاشت‌هنده و کاهش‌دهنده داده شده و همچنین آدرس پوشه input موجود در HDFS که حاوی فایل‌های متنی ورودی است و آدرس پوشه خروجی روی HDFS نیز ذکر شده است.



شکل 5: دستور اجرای عملیات نگاشت-کاهش

زمان اجرای کار، هم در command line پس از اتمام دستور اجرای کار، و هم در رابط کاربری تحت وب محفضه ResourceManager در آدرس <http://localhost:8088> قابل مشاهده است. در شکل ۶ مشاهده می‌شود که زمان اجرای دستور شکل ۵ برابر با ۲۵ دقیقه و ۳ ثانیه می‌باشد.



شکل6: اطلاعات job اجرا شده در رابط کاربری تحت وب Resource Manager

## عملیات map و reduce

دو فایل در پوشه src موجود است که مربوط به این عملیات می‌باشند. فایل ورودی متنی توسط Hadoop از طریق ورودی استاندارد (Standard Input) به هر بخش از عمیلات داده می‌شود و همچنین خروجی هر بخش نیز از خروجی استاندارد (Standard Output) توسط Hadoop دریافت می‌شود.

در کد برنامه از تولید‌کننده‌های (Generators)پایتون استفاده شد که به‌جای اینکه ابتدا تمام عناصر را دریافت کنند و سپس یک لیست یا آرایه را برگردانند، به محض آماده شدن هر عنصر آن را برمی‌گردانند. در واقع هر عنصر در حافظه قرار نمیگیرد تا وقتی که بخواهیم آن ‌را استفاده کنیم. در واقع اینگونه فقط زمان دسترسی به هر شئ (Object) آن شئ از محل اصلی (در اینجا ورودی استاندارد) دریافت شده و در حافظه قرار می‌گیرد. یعنی yield به جای return فقط اشاره‌گری به آن تکه از محل اصلی (در اینجا ورودی استاندارد) است. اینگونه می‌توان در مصرف حافظه بسیار صرفه‌جویی کرد و نیاز نیست تمام داده‌ها در حافظه بارگذاری شوند. همچنین در زمان اجرا نیز تاثیر بسزایی دارد زیرا نیاز نیست منتظر باشیم حلقه کامل انجام شود و سپس خروجی برگردانده شود و سپس توسط حلقه دیگری عملیات مورد نظر روی داده‌ها انجام شود.

### Mapper.py

در این ماژول و تابع اصلی آن، خط به خط متن را از ورودی استاندارد با استفاده از تابع read\_input می‌خوانیم و در هر خط روی هر کلمه تابع clean\_word را اجرا می‌نماییم. اگر تابع clean\_word کلمه‌ای را برگرداند آن‌را به همراه عدد یک در خروجی استاندارد چاپ می‌کنیم. یعنی این کلمه یک بار یافت شد. میان هر کلمه و تعداد آن با یک فاصله tab جدا شده است.

در تابع **read\_input** از Generator ‌ها استفاده شد و برای خواندن خط به خط از ورودی استاندارد، به محض دریافت هر خط از ورودی استاندارد اشاره‌گری به آن خط را باز می‌گرداند و منتظر نمی‌ماند تا تمام خط‌ها را در یک آرایه برگرداند. در واقع خروجی این تابع یک Iterable است.

تابع **clean\_word** وظیفه تمیز‌کردن هر کلمه را دارد. ابتدا بررسی می‌کند و اگر کاراکتر نامربوطی (مانند نقطه، ویرگول، پرانتز و یا علامت نقل قول) در آن کلمه بود آن را با استفاده از تابع کمکی remove\_chars از کلمه حذف می‌کند و کلمه را به حالت حروف کوچک (lowercase) تبدیل می‌کند. سپس بررسی می‌کند که آیا این کلمه یک کلمه توقفی یا حرف اضافه است یا خیر، در صورتی که کلمه یک کلمه توقفی یا حرف اضافه نبود خود کلمه و در غیر اینصورت پوچ (Null) برمی‌گرداند.

### Reducer.py

در هدوپ به صورت خودکار یک مرحله میانی تحت عنوان shuffle & sort روی خروجی نهایی مرحله map انجام می‌شود و سپس به عنوان ورودی به مرحله reduce داده می‌شود. در این ماژول و در تابع اصلی آن، خط به خط متن از این ورودی ذکر شده توسط تابع read\_mapper\_output دریافت می‌شود. هر خط در واقع یک کلمه و تعداد آن (۱) است که با یک فاصله tab جدا شده‌اند. این کلمات به صورت الفبایی توسط shuffle & sort درونی Hadoop مرتب شده‌اند. با استفاده از کتابخانه collections که یکی از کتابخانه‌های همراه پایتون است (جزؤ standard lib) و شئ Counter (در واقع یک دیکشنری از اشیاء و تعداد آن‌ها است و تعدای متد کاربردی نیز ارائه می‌کند) که توسط این کتابخانه ارائه می‌شود، تعداد کل هر کلمه محاسبه و ذخیره می‌شود. در انتها کلمات به صورت مرتب شده نزولی بر اساس تعدادشان در خروجی استاندارد چاپ می‌شوند. (۱۰ کلمه برتر درواقع ۱۰ کلمه اول خروجی reducerهستند)

در تابع **read\_mapper\_output** از Generator ‌ها استفاده شد و دقیقا مشابه تابع read\_input از ماژول mapper عمل می‌کند. تنها تفاوت این تابع این است که خط دریافتی را با استفاده از فاصله tab به دو بخش کلمه و تعداش تقسیم می‌کند و بر‌می‌گرداند.

## ۱۵ کلمه برتر به همراه تعداد تکرار قابل مشاهده در فایل خروجی output.txt

