به نام ایزد هستی بخش

فرزین فرهادی – ۱۴۰۲۵۲۱۱۰۶

گزارش تمرین دوم تحلیل سیستم ها و داده های حجیم

دانشگاه ارومیه – نیمسال ۱-۱۴۰۳

طی صحبت‌های انجام شده در کلاس و ضبط شده در ویدئو، قرار است که PostHistory مربوط به فایل Stack Over Flow Dump را پردازش کرده و الگوریتم Word Count را روی آن پیاده سازی نمایید.

برای این تمرین Pre-Processing نیاز نیست. همچنین خود MapReduce برنامه Word Count را بصورت پیش فرض دارد.

در نهایت خروجی برنامه را بدست آورده و یک گزارش کامل علمی نیز بنویسید.

بخش‌های تمرین:

۱. آماده سازی و نصب MapReduce

۲. آماده کردن فایل جهت انتقال به HDFS

۳. راه اندازی Word Count بر روی فایل بزرگ تکه تکه شده (مثلا هر تکه ۱۰ گیگ)

۴. گزارش کامل کامل علمی

این تمرین در مقیاس توزیع کوچک و قطعه بندی شده در گروه دو نفره انجام شده است. اعضای گروه: فرزین فرهادی و سامان محمدزاده.

در روند انجام تمرین با مشکلاتی رو به رو شدیم که در حین حل آن مشکلات راه حل های گوناگونی امتحان کردیم، از جمله ویرایش hadoop-env.sh، اطمینان از ارتباط غیرتعمالی و ... اما بعدا دیدیم که این تغییرات راه حل مشکل ما نبوده. از این رو در این فایل این عملیات ذکر نخواهند شد.

بدیهی است که برای نسخه های مختلف Hadoop روش های های جایگزین بیشتری برای انجام تنظیم های اولیه وجود دارد، مانند تنظیم hdfs-site.xml و ...که ام تنها ازیکی از آنها استفاده کردیم. اما بهتر است خاطرنشان کنیم که روش های جایگزین بیشتری وجود دارد.

مراحل شامل:

- **پوشه ها و فایل ها**
- **یادآوری چیستی شمارش واژگان حجیم با MapReduce.**
- **ماهیت داده**
- **آماده سازی محیط**
- **آماده سازی داده**
- **پیشنیازها**
- **تعریف و برقراری ارتباط میان گره ها**
- **تعریف Masters و Workers**
- **Hadoop Distributed File System Configurations**
- **تنظیم محل ذخیره سازی گره ها**
- **شروع سرویس**
- **اعمال MapReduce**
- **نتایج**

پوشه ها و فایل ها

- بخاطر حجم بالای ورودی و خروجی پروژه آنها را ضمیمه نکردم.

Raw Dataset Head	شامل فایل head ۱ مگابایتی دیتاست خام
Codes	کد های اصلی استفاده شده در پروژه
Output Heads	شامل فایل های head ۱ مگابایتی خروجی های های نهایی دیتاست
WordClouds	تصاویر ابر-واژگان توکن های معنادار بدون Stopwords.

یادآوری چیستی شمارش واژگان حجیم با MapReduce.

شمارش واژگان داده های متنی بسیار حجیم (در حد بالای ۱۰۰) گیگابایت شاید در وهله نخست ساده به نظر بیاید اما پیچیدگی زمانی نمایی دارد. از این رو باید الگوریتمی بهینه و توزیعی برای حل هرچه سریعتر این مشکل داشته باشیم.

در اینگونه مواقع از الگوریتم MapReduce توزیع شده روی گره (سیستم) های متفاوت استفاده می کنیم. به طوری که داده در گره های مختلف به بلاک های مختلف تقسیم بندی می شود تا مناسب CPU و RAM باشد و با الکوریتم های نهفته از پیش تعیین شده جداول Map گوناگونی برای هر بلاک ایجاد می شود. سپس Mapها را با هم تجمیع و کاهش می دهد (Reduce). برای نمونه:

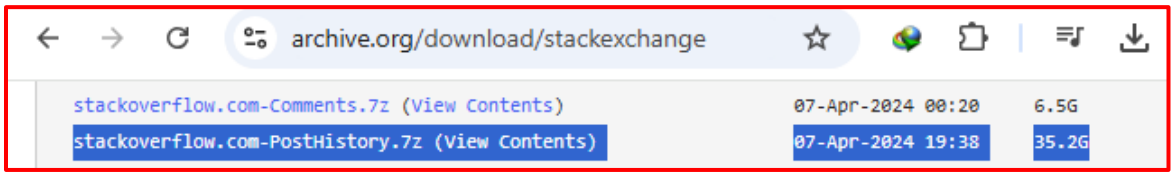
Block 1 Map		Block 2 Map		Block 3 Map		Reduce	
ایران	۵	شهر	۷	سرزمین	۱۵	ایران	۸
مادر	۱۲	ملت	۷	ایران	۱	مادر	۱۲
سرزمین	۱	ایران	۲	خاک	۸	سرزمین	۱۶
کشور	۲			آب	۲	کشور	۲
				آینده	۱	شهر	۷
				نهاد	۲	ملت	۷
						خاک	۸
						آب	۲
						آینده	۱
						نهاد	۲

شایان ذکر است که چون هیچ PreProccessای روی فایل انجام نمی دهیم، عناصری مانند "here" و "here." دو توکن مجزا ایجاد می کنند.

ماهیت داده

- طبق خواست تمرین سراغ دانلود StackOverFlow-PostHisory می رویم که یک فایل فشرده حجیمی است که وبسایت Arhive.Org فابل دسترس است:

- https://archive.org/download/stackexchange/stackoverflow.com-PostHistory.7z



- پس از استخراج فایل فشرده بالا به یک فایل xml حجیم تر (حدود ۱۷۰ گیگابایت) می رسیم.

- برای بیشتر آشنا شدن با ساختار داده از ۱ مگابایت نخست فایل Head گرفتیم:

- فایل هدف ما متن موجود در کلید Text است که پس از استخراج این متون آنها را به الگوریتم MapReduce برای شمارش خواهیم داد.

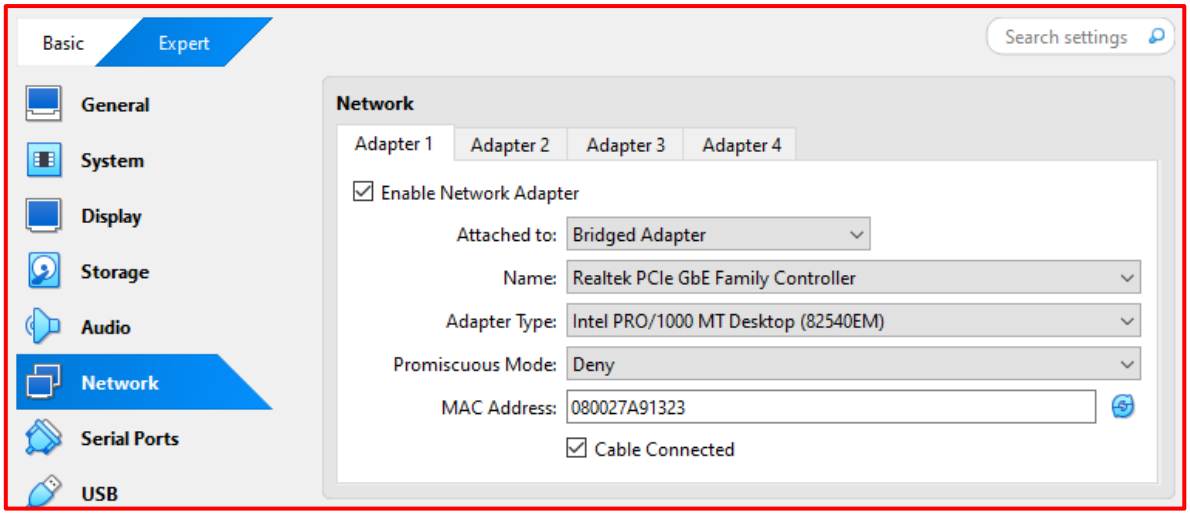
- در بخش بعدی به نحوه برخورد با داده برای استخراج متن خواهیم پرداخت.

آماده سازی محیط

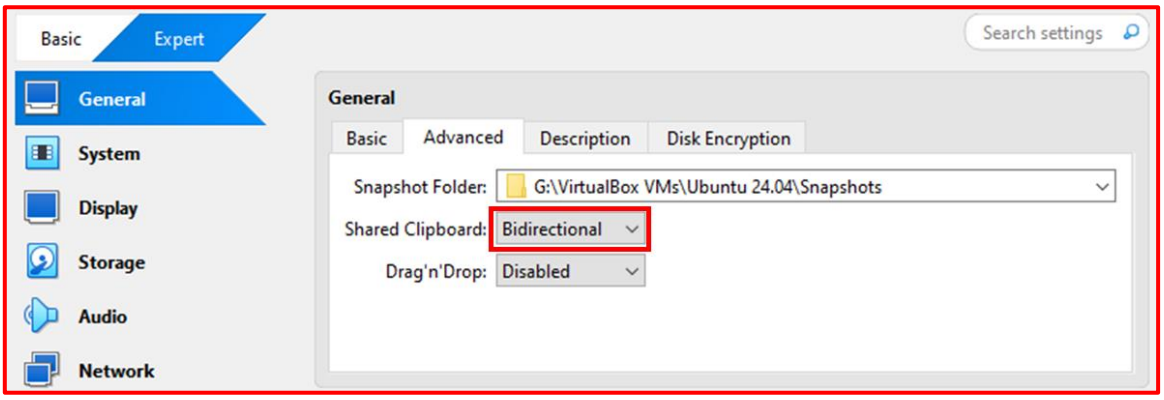
- **محیط اجرا و گره‌ها:** در هر عملیات یا سیستم توزیع شده ما یک مستر (NameNode) و گره های برده (دیتانود) داریم که عملیات به صورت توزیعی روی برده ها با مدیریت مستر انجام می شود. اگر تمرین فوق را در قالب گروه گسترده انجام دهیم کل عملیات در زمان بسیار کمتر می توانست انجام پذیرد؛ چون دیتانود (گره های Slave) زیادی موجود می شد. اما ما صرفا فقط از یک سیستم دارای اوبونتو و یک ماشین مجازی اوبونتو استفاده کردیم.
  - سیستم عامل اوبونتو، رم ۱۶ گیگابایت، حافظه ۹۰ گیگابایت (گره Master و Data)
  - سیستم عامل مجازی اوبونتو (VirtualBox)، رم ۴ گیگابایت، حافظه ۲۵ گیگابایت (گره Data)

- **ارتباط:** ارتباط میان دو رایانه با Access Point مودم بی‌سیم انجام شده است.

- برای پل زدن اینترنت محیط ویندوز به اوبونتوی مجازی گزینه Bridged Network را از بخش Settings برای اینترنت مودم فعال کردیم.



- برای راحتی ارتباط کلیپ‌بورد کی‌بورد های محیط اوبونتوی مجازی و محیط ویندوز گزینه کلیپ‌بورد مشترک را از بخش Settings ماشین مجازی فعال کردیم:



آماده سازی داده

- دیدیم که داده مدنظر ما بسیار حجیم است، حدود ۱۷۰ گیگابایت.
- اما بدیهی است که که پس از استخراج کلید Text از فایل xml حجم فایل تا حدی کاهش می یابد. پس از انجام این پیش‌پردازش به حجم ۱۰۰ گیگابایت رسیدیم.
  - بخاطر کمبود حجم، نمی توانیم تمامی داده را یکجا پردازش کنیم پس باید متن هایی که می خواهیم به MapReduce دهیم را از پیش قطعه (Chunk) بندی کنیم.
  - برای قطعه بندی داده نخست باید آن را بخوانیم، روش خوانش را نخست خواستیم از روی ID پیش ببریم که کارساز نبود، چون هم آیدی ها مرتب نبودند و هم عملیات خوانش بسیار زیادی می خواست که طبیعتا زمانبر خواهد بود.

- سعی کردیم از روی شمارش خط داده را قطعه بندی کنیم. به طوری که تعداد میزان حجم هر chunk را مشخص کرده و نسبت به اینکه در کدام خط شروع و پایان می یابد آنها را بازه بندی کردیم. گزارش بسیار مختصری هم از ساخت هر chunk و میزان زمان صرف شده برای ساخت گرفتیم. این گزارشها در پوشه Chunks قابل دسترس است.
- نسبت به شرایطی که در روز های مختلف برای پردازش داشتیم chunkها با اندازه های مختلف ساختیم:

۱.	10 * 2 = 20 GB
۲.	10 * 2 = 20 GB
۳.	1 * 10 = 10 GB
۴.	1 * 10 = 10 GB
۵.	1 * 10 = 10 GB
۶.	1 * 10 = 10 GB
۷.	(2 * 8) + (1 * 3.5) = 19.5 GB
جمع	= 99.5 GB ~ 100 GB

- از جمله مشکلات عدم امکان فراهم سازی فضای مورد نیاز برای پردازش یکجای این داده:
  - کمبود گره
  - کمبود مجموع فضای خالی گره ها. در مجموع ما ۲۵+۹۰ گیگابایت فضای خالی در گره‌ها داریم که با در نظر گرفتن فایل های نصب سیستم عامل بسیار کمتر نیز می شود.
  - فایل مازاد تولیدی زیاد در **هنگام** پردازش:
    - میزان فضایی که خود دیتاست اشغال می کند.
    - میزان فضایی که Replicant های دیتاست اشغال می کند.
    - میزان فضایی tmp و cache که هنگام پردازش ایجاد می شود.
    - میزان فضایی که فایل خروجی برای ذخیره شدن نیاز دارد.

پیشنیازها

دقت شود که تمامی تنظمیاتی که از این بخش به بعد ذکر می شود باید عینا روی تمامی گره ها اجرا شده و با هم هماهنگ باشند.

بروزرسانی سیستم

- آپدیت سیستم عامل لینوکس:

```
sudo apt update
```

SSH

- نصب SSH (Secure Shell)، پروتکلی برای برقراریارتباط امن میان گره های متصل به شبکه مشترک و رد و بدل کردن داده:

```
sudo apt install openssh-server
```

- پس از نصب SSH آن وضعیت آن را چک کنید که فعال باشد:

```
farzin-vm@farzin-vm-VirtualBox:~$ sudo systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/usr/lib/systemd/system/ssh.service; enabled; preset: enabled)
     Active: active (running) since Sun 2024-12-08 00:29:09 +0330; 1min 51s ago
TriggeredBy: ● ssh.socket
       Docs: man:sshd(8)
             man:sshd_config(5)
    Process: 1361 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
   Main PID: 1375 (sshd)
      Tasks: 1 (limit: 4614)
     Memory: 2.1M (peak: 2.3M)
        CPU: 22ms
     CGroup: /system.slice/ssh.service
             └─1375 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Dec 08 00:29:09 farzin-vm-VirtualBox systemd[1]: Starting ssh.service - OpenBSD Secure Shell server...
Dec 08 00:29:09 farzin-vm-VirtualBox sshd[1375]: Server listening on :: port 22.
Dec 08 00:29:09 farzin-vm-VirtualBox systemd[1]: Started ssh.service - OpenBSD Secure Shell server.
farzin-vm@farzin-vm-VirtualBox:~$
```

- اگر SSH فعال نبود، آنرا از طریق دستور زیر فعال کنید:

```
sudo systemctl start ssh
```

جاوا

- برای راه اندازی Apache Hadoop (در مورد بعدی به آن می پردازیم) نیاز به داشتن نسخه ای از جاوا داریم، برای نصب:

```
sudo apt install openjdk-11-jdk
```

`java -version` (برای اطمینان از نصب)

Apache Hadoop

- چارچوبی است که امکان پردازش توزیع شده مجموعه داده های بزرگ را در میان گره های کامپیوتر با استفاده از مدل های برنامه نویسی ساده فراهم می کند.

- ما نسخه 3.3.6 آنرا از طریق ترمینال (به صورت دستی نیز می توان دانلود کرد) دریافت، فایل را استخراج کرده و به مسیر ~/hadoop منتقلش می کنیم.

```
wget https://downloads.apache.org/hadoop/common/hadoop-3.3.6/hadoop-3.3.6.tar.gz
tar -xvzf hadoop-3.3.6.tar.gz
mv hadoop-3.3.6 ~/hadoop
```

Environment Variables

- دقت داشته باشید که تمامی تنظیماتی که برای این بخش ذک
- Hadoop برای انجام هماهنگ عملیات خود نیاز به تعریف متغیر های محیطی دارد که باید کاربر آنها را تعریف کند. از جمله آنها مسیر های Hadoop و جاوا هستند که قبلا نصب کردیم.
- این متغیر های محیطی را داخل فایل مخفی bashrc تعریف می کنیم که در واقع شل یونیکس بوده و Configuration و ارجاعات اولیه سیستم عامل هنگام راه اندازی را شامل می شود.
- اقدام به ویرایش مسیر ~/.bashrc می کنیم:

```
sudo ~/.bashrc
```

- مسیر های مربوط به hadoop و جاوا را به آخر این فایل اضافه می کنیم. از این رو دیگر به مسیر استخراج شده hadoop با ارجاع $HADOOP_HOME دسترسی خواهیم داشت:

```
export HADOOP_HOME=/home/farzin/hadoop
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
```

تنها تفاوت در سیستم ها **متن سبز رنگی** است که باید نام کاربر در سیستم باشد. هر چند که بعداً به این نتیجه خواهیم رسید که مجبور هستیم که تمامی دستگاه ها باید نام این مسیر را نام کاربری Master قرار دهند.

- از آنجایی که bashrc یک فایل script است، پس ازهربار ویرایش برای اعمال تغییرات آن باید دستور زیر را اعمال کنیم:

```
source ~/.bashrc
```

- برای اطمینان از اعمال تغییرات بالا می توان متغیر های سراسری تنظیم شده را echo کرد:

```
farzin-vm@farzin-vm-VirtualBox: ~
farzin-vm@farzin-vm-VirtualBox:~$ echo $HADOOP_HOME
/home/farzin/hadoop
farzin-vm@farzin-vm-VirtualBox:~$ echo $HADOOP_CONF_DIR
/home/farzin/hadoop/etc/hadoop
```

- **مشکلات احتمالی:** در صورت به مشکل برخوردن در مسیر های دسترسی در آینده (فعلا اجباری در این اعمال ای نتغییرات نیست):
  - اعمال متغیر های محلی روی مسیر ~/.profile و سپس source کردن آن.
  - اعمال متغیر های محلی روی $HADOOP_HOME/etc/hadoop/hadoop-env.sh.

تعریف و برقراری ارتباط میان گره ها

- بدیهی است که دو سیستم در حالت عادی با هم ارتباطی ندارند. بیش از هر چیز این دو سیستم اگر میخواهند دچار اتلاف حجم اینترنت خارج از LAN نشوند بهتر است به صورت سیمی یا یک Access Point (نقطه دسترسی) به هم متصل شوند.
- این ارتباط از طریق SSH برقرار می شود که در مراحل پیشین در موردش بحث کردیم. این ارتباط همانطور که از نامش مشخص است باید شناخته شده و امن باشد. از این رو نخست باید هویتی برای سیستم ها در شبکه تعریف کرده و کلید هایی برای ارتباط امن و بدون رمز تعریف کنیم.

نامگذاری گره ها

- همانطور که گفتیم ما در این تمرین از دو گره استفاده کردیم. شبکه مشترک گره ها به صورت بی‌سیم نسبت به موقعیت از Access Point مودم یا هات‌اسپات گوشی استفاده کردیم.
- می دانیم که اینگونه ارتباطات معمولا دارای نشانی شبکه 192.168.1.0 و Subnet Mask 255.255.255.0 هستند. که باعث می شود آدرس آی‌پی دستگاه ها یا همان میزبان‌ها (Hosts) 192.168.1.x باشند.
- برای دریافت آدرس آی‌پی در دستگاه مورد نظر دستور ifconfig را وارد کردیم.

```
farzin@farzin-Aspire-A715-74G: ~
farzin@farzin-Aspire-A715-74G:~$ ifconfig
wlp8s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.6  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::18e4:4892:31d3:388e  prefixlen 64  scopeid 0x20<link>
        ether 4c:1d:96:b9:93:44  txqueuelen 1000  (Ethernet)
        RX packets 8070  bytes 2680205 (2.6 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8623  bytes 5117234 (5.1 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Master

```
farzin-vm@farzin-vm-VirtualBox: ~
farzin-vm@farzin-vm-VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.9  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::a00:27ff:fea9:1323  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:a9:13:23  txqueuelen 1000  (Ethernet)
        RX packets 1874  bytes 128923 (128.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3883  bytes 337743 (337.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Slave

- برای راحتی در بخاطر سپردن نام میزبان‌ها و ارجاع نسبتا زیاد به نام دستگاه ها بهتر است پس از افزودن آی‌پی میزبان ها به سیستم آنها را نامگذاری نماییم. بدین ترتیب در صورت تغییر آدرس ها تنها کافیست آی‌پی ها در فایل ارجاعی را تغییر دهیم. برای نمونه برای اشاره به یک کاربر در سیستمی دیگر:

```
username@hostname or username@ip-address
```

- برای تنظیم میزبان ها در هر دو سیستم از طریق دستور زیر آدرس و نام هایی که خودمان به دلخواه برگزیدیم را وارد می کنیم.

```
sudo nano /etc/hosts
```

```
  GNU nano 7.2                    /etc/hosts
127.0.0.1 localhost
127.0.1.1 farzin-vm-VirtualBox

192.168.1.6 master-UU
192.168.1.9 slave-UU
```

برقراری ارتباط

- همانطور که گفتیم گره ها پس از اینکه به یک شبکه مشترک وصل شدند باید باید بتوانند ارتباط کرده و به هم دیگر دسترسی بدون پسورد داشته باشند. هر چند که این دسترسی را می توان میان همه گره ها درست کرد اما تنها دسترسی داشتن گره Master به گره [های] Slave کافی است.
- برای این ارتباط یک کلید عمومی در گره Master تعریف کرده و آن را به گره های Slave کپی می کنیم.
- دستور ساخت کلید عمومی. کلید از نوع RSA و ۴۰۹۶ بیتی است.

```
ssh-keygen -t rsa -b 4096
```

- کپی کردن کلید عمومی به فهرست کلید های Slave:

```
ssh-copy-id -i $HOME/.ssh/id_rsa.pub farzin-vm@slave-UU
```

در صورت مشاهده پیام زیر، پاسخ yes را وارد کنید.

```
Are you sure you want to continue connecting
(yes/no/[fingerprint])?
```

- برای تست امکان برقراری ارتباط. دستور زیر را از Master برای پیوند به ترمینال Slave استفاده می کنیم. اگر ارتباط بدون درخواست پسورد برقرار شد یعنی همه چیز درست پیش می رود.

```
farzin@farzin-Aspire-A715-74G: ~
farzin-vm@farzin-vm-VirtualBox:~$ ssh farzin@master-uu
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-49-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

96 updates can be applied immediately.
8 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet

Last login: Wed Dec 11 21:58:24 2024 from 192.168.1.9
farzin@farzin-Aspire-A715-74G:~$
```

دسترسی به ترمینال Slave بدون درخواست پسورد

- هنگام انجام عملیات به اینکه در کدام سیستم هستیم نیز باید دقت کنیم. برای خارج شدن از ترمینال دستگاهی دیگر کافیست دستور exit را وارد کنیم.

تعریف Masters و Workers

- در محل های مشخص مسیر Hadoop فایل هایی وجود دارند برای تعریف Masterها و Slaveها.
- با ویرایش مسیر زیر نام میزبان Master را وارد می کنیم. در صورت وجود localhost آنرا پاک کنید.

```
sudo nano $HADOOP_HOME/etc/hadoop/masters
```

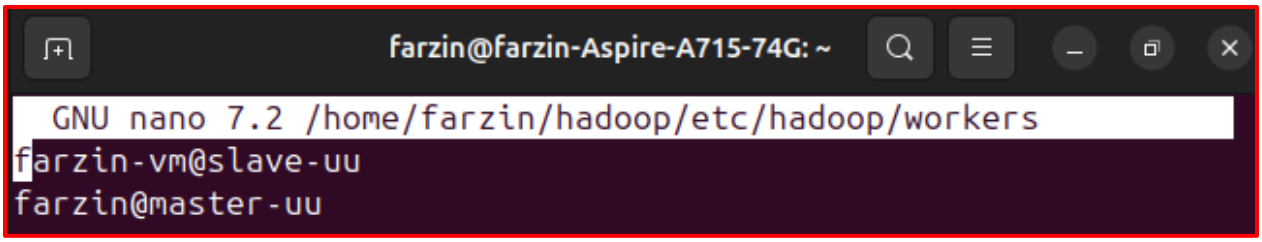
```
farzin@farzin-Aspire-A715-74G: ~
  GNU nano 7.2 /home/farzin/hadoop/etc/hadoop/masters
master-UU
```

- برای اشاره سیستم های کارگر معمولا از اصطلاح «برده» یا Slave استفاده می کنیم. در Hadoop نیز دو فایل Workers و Slaves برای تعریف این گره ها وجود دارد اما فایل

Slaves دیگر منقضی شده است و کارکرد ندارد. از این رو گره های کارگر را تنها در فایل Workers تعریف می کنیم. در صورت وجود localhost آنرا پاک کنید.

ما از master-UU هم به عنوان مسترنود و هم به عنوان دیتانود استفاده می کنیم تا عملیات سریع تر پیش برود.

```
sudo nano $HADOOP_HOME/etc/hadoop/workers
```



دقت کنید که گره های کارگر باید همراه با نام کاربری شان تعریف شوند وگرنه خطای عدم هماهنگی کلید عمومی می گیریم چون Hadoop نمی تواند به تنهایی کاربری که کلید عمویم مشترک داریم را چک کند.

Hadoop Distributed File System Configurations

- در این بخش اقدام به ویرایش فایل های پیکربندی هدوپ می کنیم تا گره ها و مسیرها را برای هدوپ نیز بشناسانیم. در این فایل ها پیکربندی های گوناگونی می توان انجام داد اما ما تنها به تنظیمات لازم برای انجام عملیات توزیع شده MapReduce می پردازیم. فقط کافیست تا کد های موجود در تصاویر را با کد های پیشفرض جایگزین کنید.
- در hdsf تقریبا تمامی کارها برای عملیات MapReduce اتوماتیک است. ما تنها پیکربندی های اولیه را انجام می دهیم.

core-site.xml

- این فایل شامل تنظیمات کلی و مرکزی خوشه hdfs می شود.
- ما دو چیز را در این فایل تعریف می کنیم:
  - مسیر tmp در گره ها برای ذخیره سازی داده های موقت تولید شده هنگام انجام عملیات MapReduce.
  - تعریف NameNode به عنوان مستر یا همان مدیر.
- دستور ویرایش این فایل:

```
sudo nano $HADOOP_HOME/etc/hadoop/core-site.xml
```

```
<configuration>
<property>
    <name>hadoop.tmp.dir</name>
    <value>/app/hadoop/tmp</value>
    <description>A base for other temporary directories.</description>
</property>

<property>
    <name>fs.default.name</name>
    <value>hdfs://master-UU:54310</value>
    <description>The name of the default file system.  A URI whose
    scheme and authority determine the FileSystem implementation.  The
    uri's scheme determines the config property (fs.SCHEME.impl) naming
    the FileSystem implementation class.  The uri's authority is used to
    determine the host, port, etc. for a filesystem.</description>
</property>
</configuration>
```

مسیر ذخیره فایل های موقت عملیات MapReduce که در مراحل بعدی بیشتر به آن خواهیم پرداخت.

تعریف NameNode با استفاده از نام میزبان و پورت پیشفرض آن.

mapred-site.xml

- این فایل شامل پارامترهایی برای برای تعریف نحوه اجرای و مدیریت کارهای MapReduce در سراسر خوشه هدوپ است.
- ما دو چیز را در این فایل تعریف می کنیم:
  - تنظیم نام میزبان و پورت پیشفرض JobTracker برای:
    - پذیرش و مدیریت MapReduce.
    - نظارت بر پیشرفت کار.
    - رسیدگی به failها.
  - چارچوب اجرایی MapReduce را براساس YARN تنظیم می کند که چارچوبی برای مدیریت منابع جدا از امور (job) تحت مدیریت مستر است.
- دستور ویرایش این فایل:

```
sudo nano $HADOOP_HOME/etc/hadoop/mapred-site.xml
```

```
<configuration>
<property>
    <name>mapred.job.tracker</name>
    <value>master-UU:54311</value>
    <description>The host and port that the MapReduce job tracker runs
    at.  If "local", then jobs are run in-process as a single map
    and reduce task.
    </description>
</property>

<property>
    <name>mapred.framework.name</name>
    <value>yarn</value>
</property>
</configuration>
```

تنظیم مستر به عنوان ردیاب امور

تعریف چارچوب MapReduce

yarn-site.xml

- این فایل برای پیکربندی چارچوب YARN است که در بخش پیشین به آن اشاره شد. وظیفه YARN مدیریت منابع و برنامه ریزی کارها (Jobs) در سرتاسر خوشه برای بالا بردن دسترسی (Availability) است.
- ما سه چیز را در این فایل تعریف می کنیم:
  - سرویس ردیاب یا همان پیگیر NameNodeها.
  - سرویس برنامه ریز و تنظیم کننده منابع.
  - سرویس ثبت درخواست و تعامل مستر[ها] با مدیر منابع.
- دستور ویرایش این فایل:

```
sudo nano $HADOOP_HOME/etc/hadoop/yarn-site.xml
```

```
<configuration>
<property>
    <name>yarn.resourcemanager.resource-tracker.address</name>
    <value>master-UU:8025</value>
</property>
<property>
    <name>yarn.resourcemanager.scheduler.address</name>
    <value>master-UU:8035</value>
</property>
<property>
    <name>yarn.resourcemanager.address</name>
    <value>master-UU:8050</value>
</property>
</configuration>
```

NodeManagers (عوامل YARN) از این سرویس برای ارسال پالس به ResourceManager استفاده می‌کنند. پالس بروزرسانی وضعیت، از جمله در دسترس و سالم بودن منابع.

مسئول تخصیص منابع به برنامه ها است. از سیاست های زمان‌بندی (مانند FIFO، ظرفیت یا Fair Scheduler) برای توزیع منابع بین taskها استفاده می‌کند.

مسترها از این نقطه پایانی برای ارسال درخواست‌های Job و تعامل با ResourceManager برای بروزرسانی وضعیت و هماهنگی استفاده می‌کنند.

hdfs-site.xml

- این فایل برای تعریف پارامتر های ذخیره سازی، Replicant، رفتار hdfs و ... است.

- ما در این فایل صرفا فقط تعداد Replicant (نمونه های عینی داده ها) تعریف می کنیم که معمولا با تعداد دیتانودها همسان است. این Replicantها برای تحمل خطا های احتمالی در طول عملیات مفید هستند.
- می توان تنظیماتی از جمله تعریف مسیر ذخیره سازی فایل های گره های مستر و برده را نیز در این فایل انجام داد اما نیاز نیست چون مسیر tmp که در core-site.xml تعریف کردیم نیز این کار را انجا می دهد که در مراحل بعدی به آن خواهیم پرداخت.
- دستور ویرایش این فایل:

```
sudo nano $HADOOP_HOME/etc/hadoop/hdfs-site.xml
```

```
<configuration>
<property>
    <name>dfs.replication</name>
    <value>2</value>
    <description>Default block replication.
        The actual number of replications can be specified whe>
        The default is used if replication is not specified in>
        </description>
</property>
</configuration>
```

تنظیم محل ذخیره سازی گره ها

- بخاطر داریم که مسیر app/hadoop/tmp/ را به عنوان محل ذخیره موقت گره ها اختصاص دادیم. اکنون کافیست که مسیر های جداگانه برای مسترها و برده ها ایجاد کرده و مالکیت کاربران گره ها را به آنها بدهیم.
- مراحل کار:
  - برای اطمینان از پاک شدن فایل های قدیمی در این مسیر، آن را یکبار کاملا پاک می کنیم:

```
sudo rm -rf /app/hadoop/tmp
```

  - ساخت مسیر های جداگانه برای گره ها:
    - NameNode:

```
sudo mkdir -pv /app/hadoop/tmp/hdfs/namenode
```

    - DataNode:

```
sudo mkdir -pv /app/hadoop/tmp/hdfs/datanode
```

- اعطای دسترسی بازگشتی کاربران (هم مستر هم برده) به این مسیرها و تمامی زیرمسیرهایشان:

```
sudo chown username:username -R /app/hadoop/tmp/
```

شروع سرویس

- تمامی تنظیمات اولیه انجام شد. حال سراغ شروع سرویس hdfs و yarn می رویم. پیش از هر چیز یکبار Filesystem خوشه هدوپ را فرمت کنیم کافیست:

```
hdfs namenode -format
```

- برای اطمینان از عدم تداخل هرگونه سرویس احتمالی در حال اجرای پیشین یکبار دستور زیر را وارد می کنیم.

```
stop-all.sh
```

- دستور شروع سرویس های hdfs و yarn در گره مستر:

```
start-dfs.sh && start-yarn.sh
```

- **خطای محتمل:**

- اگر با تنظیماتی که تا کنون انجام داده ایم پیش برویم در شروع سرویس ها به مشکل برخواهیم خورد. متاسفانه هنگام برخورد با این خطا اسکرین شاتی از ان نگرفتم و هر چقدر خواستم دوباره ایجادش کنم نتوانستم. (😁) از این رو خطا را صرفا توضیح می دهم.
- در صورت اختصاص دادن نام کاربری و نام میزبان ها در فایل Workers، هنگام ساخت عناصری مانند DataNode در سمت برده ها با خطای عدم دسترسی به مسیر زیر مواجه می شویم:

```
/home/master_username/hadoop/bin/hdfs
```

- مهم نیست که تمامی تنظیمات و ارجاعات در برده ها براساس نام کاربری خودش باشد اما هنگام ساخت DataNodeها سراغ مسیر نام کاربری مستر می رود.
- نخست رفتم سراغ محتویات فایل start-dfs.sh تا ببینم این مسیر در کجا فراخوانده شده. همانطور که در تصویر نیز می بینیم، متغیر سراسری HADOOP_HDFS_HOME مسیر هدوپ را تعیین می کند که در تمامی سیستم ها روی مسیر هدوپ مستر است که دلیل مواجه شدن با خطاست.

```
#----------------------------------------------------------
# datanodes (using default workers file)

echo "Starting datanodes"
hadoop_uservar_su hdfs datanode "${HADOOP_HDFS_HOME}/bin/hdfs" \
    --workers \
    --config "${HADOOP_CONF_DIR}" \
    --daemon start \
    datanode ${dataStartOpt}
(( HADOOP_JUMBO_RETCOUNTER=HADOOP_JUMBO_RETCOUNTER + $? ))
```

- **راه‌حل احتمالی:** سعی کردیم تا این متغیر سراسری را در فایل هایی مانند bashrc، profile و hadoop-enc.sh تعریف کنیم، اما کارساز نبود باز خطای یکسان را دریافت می کردیم. به گمانم ما هرچقدر پیکربندی های برده ها را تغییر دهیم باز در مستر سراغ ارجاعات می گردد.
- **راه‌حل استفاده شده:** برای حل این مشکل یک میانبر زدیم که صرفا به قول معروف «کارمان راه بیوفتد». مسیر ذکر شده با نام کاربری مستر را در سیستم برده ایجاد کردیم.
- دقت کنید که کاربر جدید ایجاد نکردیم، صرفا در مسیر home پوشه ای با نام کاربری مستر ایجاد کرده و پوشه hadoop را به آنجا منتقل کردیم.

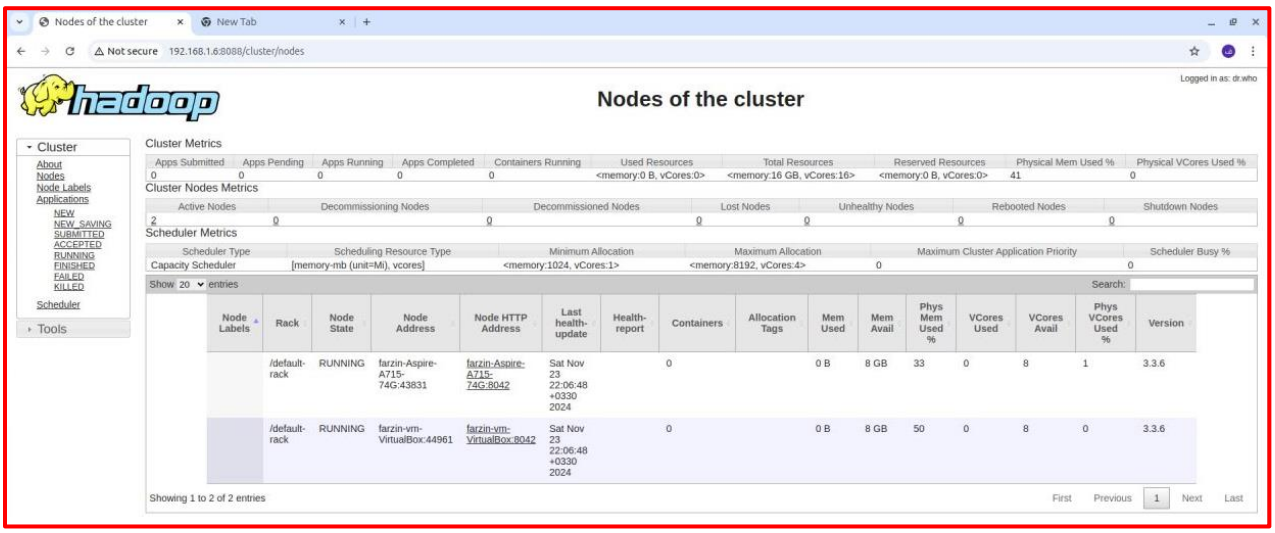
```
mkdir -p /home/farzin
```

- تمامی متغیر های سراسری موجود در فایل های bashrc، profile و hadoop-enc.sh را تغییر دادیم و مشکل حل شد. به طوری که پس از شروع کردن سرویس های HDFS و YARN با وارد کردن دستور jps (Java Process Status) می توان دیمون (Daemon) های در حال پردازش در هر دو سیستم را دید.

- میبینیم که تمامی سرویس با تمامی عناصرش بدون خطا در حال اجرا است. برای توقف سرویس ها می توان از دستور زیر استفاده کرد.

```
stop-dfs.sh && stop-yarn.sh
```

- می توان از طریق آدرس host-ip:8088 وضعیت گره ها، تاریخچه و گزارش های مربوط به آن ها را بررسی کرد. از آنجایی که ما کار خیلی سنگین و پیچیده ای انجام نمی دهیم اطلاعات چندانی از این بخش دریافت نمی کنیم. تنها گره هایمان را بررسی می کنیم که در حالت Running هستند.



اعمال MapReduce

- حال که سرویس در حال اجرا شدن است کافیست که توکن های دیتاست chunk بندی شده را با روش MapReduce شمارش کنیم.

پوشه های ورودی و خروجی

- نخست باید پوشه های ورودی که Chunkها را می گیرد و پوشه خروجی که نتایج MapReuduce در آنجا ذخیره می شود (به صورت پیش‌فرض input/ و output/ در مسیر hdfs) را بسازیم. بدون وجود این پوشه ها با خطا مواجه می شویم.
- این مورد را نیز بخاطر داشته باشیم که پس از هر مرحله اجرای MapReduce باید این دو پوشه تخلیه یا حذف شوند (در صورت حذف مسیر ورودی اتوماتیک ایجاد نمی شود باید خودمان ایجاد کنیم، اما مسیر خروجی به صورت اتوماتیک ایجاد می شود) وگرنه با انباشتگی حجم یا خطای عدم امکان OverWrite مواجه می شویم. پس باید پیش از انجام MapReduce روی هر Chunk دستورات زیر را که مخصوص محیط hdfs است اجرا کنیم:

```
hdfs dfs –rm -r /output
hdfs dfs –rm -r /input
hdfs dfs –mkdir –p /input
```

- در این پروژه فایل پیشفرض ورودی را text.txt و خروجی را part-00000 در نظر گرفتیم. نحوه منتقل کردن فایل ورودی به پوشه ورودی hdfs:

```
Hdfs dfs –put text.txt /input/text.txt
```

- پس از انتقال ورودی به مسیر input/ می توان با دستور زیر اطلاعات نحوه ذخیره سازی ورودی را گرفت.

```
hdfs fsck /input/text.txt –files –blocks -locations
```

```
farzin@farzin-Aspire-A715-74G: ~
/input/text.txt 106912050 bytes, replicated: replication=2, 1 block(s):  OK
0. BP-1111751746-127.0.1.1-1732430424733:blk_1073741830_1006 len=106912050 Live_repl=2  [Da
tanodeInfoWithStorage[192.168.1.9:9866,DS-5a6aec67-d66e-4ab0-a210-f68768d3a77c,DISK], Datan
odeInfoWithStorage[192.168.1.6:9866,DS-55fa30d4-2af2-411d-8813-8ef1bbc0051d,DISK]]


Status: HEALTHY
 Number of data-nodes:  2
 Number of racks:               1
 Total dirs:                    0
 Total symlinks:                0

Replicated Blocks:
 Total size:    106912050 B
 Total files:   1
 Total blocks (validated):      1 (avg. block size 106912050 B)
 Minimally replicated blocks:   1 (100.0 %)
 Over-replicated blocks:        0 (0.0 %)
 Under-replicated blocks:       0 (0.0 %)
 Mis-replicated blocks:         0 (0.0 %)
 Default replication factor:    2
 Average block replication:     2.0
 Missing blocks:                0
 Corrupt blocks:                0
 Missing replicas:              0 (0.0 %)
 Blocks queued for replication: 0

Erasure Coded Block Groups:
 Total size:    0 B
 Total files:   0
 Total block groups (validated):        0
 Minimally erasure-coded block groups:  0
 Over-erasure-coded block groups:       0
 Under-erasure-coded block groups:      0
 Unsatisfactory placement block groups: 0
 Average block group size:      0.0
 Missing block groups:          0
 Corrupt block groups:          0
 Missing internal blocks:       0
 Blocks queued for replication: 0
FSCK ended at Sun Nov 24 12:41:19 IRST 2024 in 16 milliseconds


The filesystem under path '/input/text.txt' is HEALTHY
```

- دقت کنید که هر بلاک از دیتاست ورودی ۱۲۸ مگابایت است پس برای تقسیم بندی داده به بلاک های متفاوت و پخش آن میان دیتانودها کل ورودی باید از ۱۲۸ مگابایت بیشتر باید. متاسفانه من اسکرین شات از دیتاست ۱۰۰ مگابایتی را فقط ذخیره کردم که فقط یک بلاک درست می کند. اما از طریق تعداد دیتانودها می توان درست پیش رفتن روند را دید.

اجرای MapReduce

- دستور اجرای MapReduce برمبنای هدوپ به شکل زیر است: (دقت کنید، اگر بخواهیم این کدهای زیر را به صورت یکجا در ترمینال اجرا کنیم باید خطوط را با \ از هم جدا کنیم.)

```
# Automatically find the Hadoop streaming jar
hadoop_streaming_jar = subprocess.check_output(
    "echo $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-*.jar",
    shell=True, text=True
).strip()

subprocess.run([
    "hadoop", "jar", hadoop_streaming_jar,        فایل jar اجرای جریانی از کارها
    "-mapper", "mapper.py",                       ارجاع به کد پایتون mapper
    "-reducer", "reducer.py",                     ارجاع به کد پایتون reducer
    "-input", input_path,                         مسیر ورودی
    "-output", output_path                        مسیر خروجی
], check=True)
```

- کد هایی ساده برای Map و Reduce کردن نوشتیم:
  - *mapper.py: نیازی نیست که هر واژه در یک خط باشد. ملاک جدا شدن با فاصله یا خط جدید است.*

```python
#!/usr/bin/env python3
import sys

current_word = None
current_count = 0
word = None

# Input comes from standard input (stdin)
for line in sys.stdin:
    # Parse the input we got from mapper.py
    line = line.strip()
    word, count = line.split("\t", 1)
    try:
        count = int(count)
    except ValueError:
        # Ignore lines where count is not a number
        continue

    # Aggregate counts for the same word
    if current_word == word:
        current_count += count
    else:
        if current_word:
            # Write the result to stdout
            print(f"{current_word}\t{current_count}")
        current_word = word
        current_count = count

# Output the last word if needed
if current_word == word:
    print(f"{current_word}\t{current_count}")
```

- *:reducer.py*

```python
#!/usr/bin/env python3
import sys

# Input comes from standard input (stdin)
for line in sys.stdin:
    # Remove leading and trailing whitespace
    line = line.strip()
    # Split the line into words
    words = line.split()
    # Emit each word with a count of 1
    for word in words:
        print(f"{word}\t1")
```

کد یکپارچه و گزارش‌گیری

- من کد یکپارچه ای برای عمل MapReduce نوشتم که تمامی مراحل بالا را همراه با پرینت و ذخیره یک Log مختصر انجام داده و خروجی شمارش و Log را به طور محلی ذخیره می کند. تنها کافیست فایل Chunk ورودی را داشته باشیم و کد مربوطه را اجرا کنیم فقط همین.
- تصویر این کد کمی طولانی‌ست پس در اینجا نمی گنجد. این کد با عنوان run_mapreduce_with_export.py ذخیره شده است که در ابتدای گزارش در بخش « پوشه ها و فایل ها» ذکر شده بود.
- با اعمال این کد روی تمامی Chunkها تعداد زیادی فایل از شمارش نهایی Chunkها به دست می آید.
- یک نمونه Log برای فایل 10GB را در تصویر زیر می توانید ببینید. البته نسبت به تنوع پراکندگی توکن ها برای هر 10GB فایل حدود ۴۰۰۰ تا ۵۵۰۰ ثانیه زمان صرف می شود. همچنین شایان ذکر است که خروجی برای هر 10GB فایل حدود 1GB است.

```
=== MapReduce Job Summary ===
Total Word Count: 1172500939
Total Processing Time: 4030.10 seconds
Results exported to: mapreduce_result.txt
```

Reduce نهایی

- همانطور که گفتیم هم اکنون ما تعداد زیادی فایل MapReduce از Chunk های گوناگون داریم. ما باید تمامی این فایل ها را تنها با هم Reduce کنیم. این کار را با کد Final_Reducer.py انجام می دهیم که تفاوت چندانی با کد run_mapreduce_with_export.py و بسیار شبیه هستند با این تفاوت که فقط کار Reduce روی تمامی فایل های متنی یک مسیر اعمال می کند.

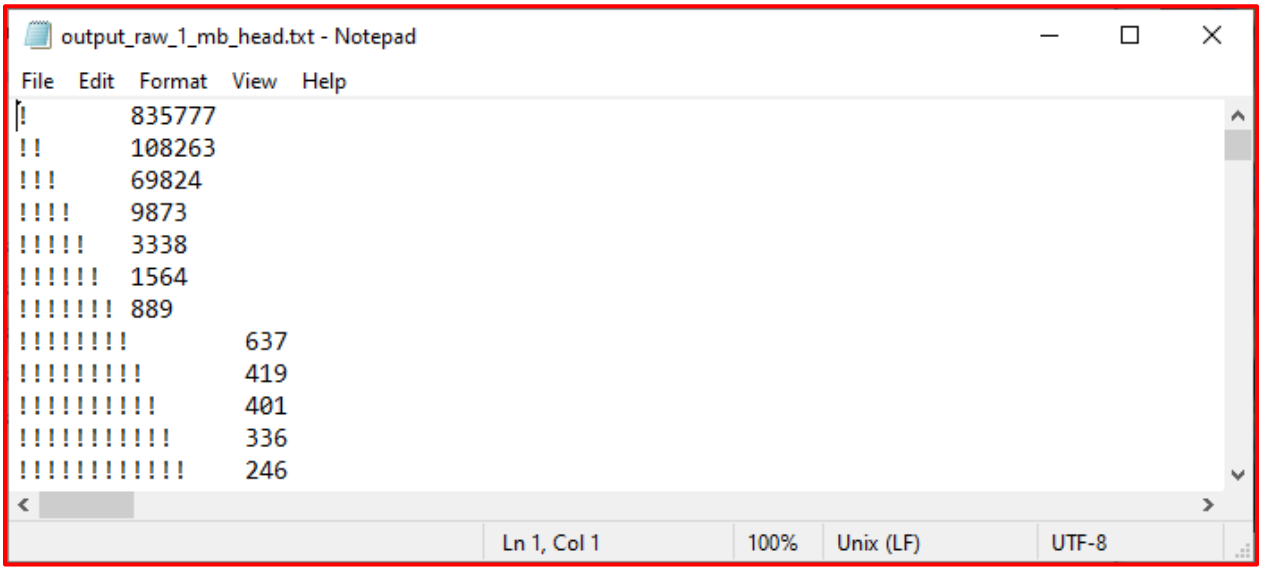
نتایج

- تمامی فایل های ذکر شده در این بخش با هماهنگی با استاد در اختیار ایشان قرار گرفته می شود.

خروجی خام

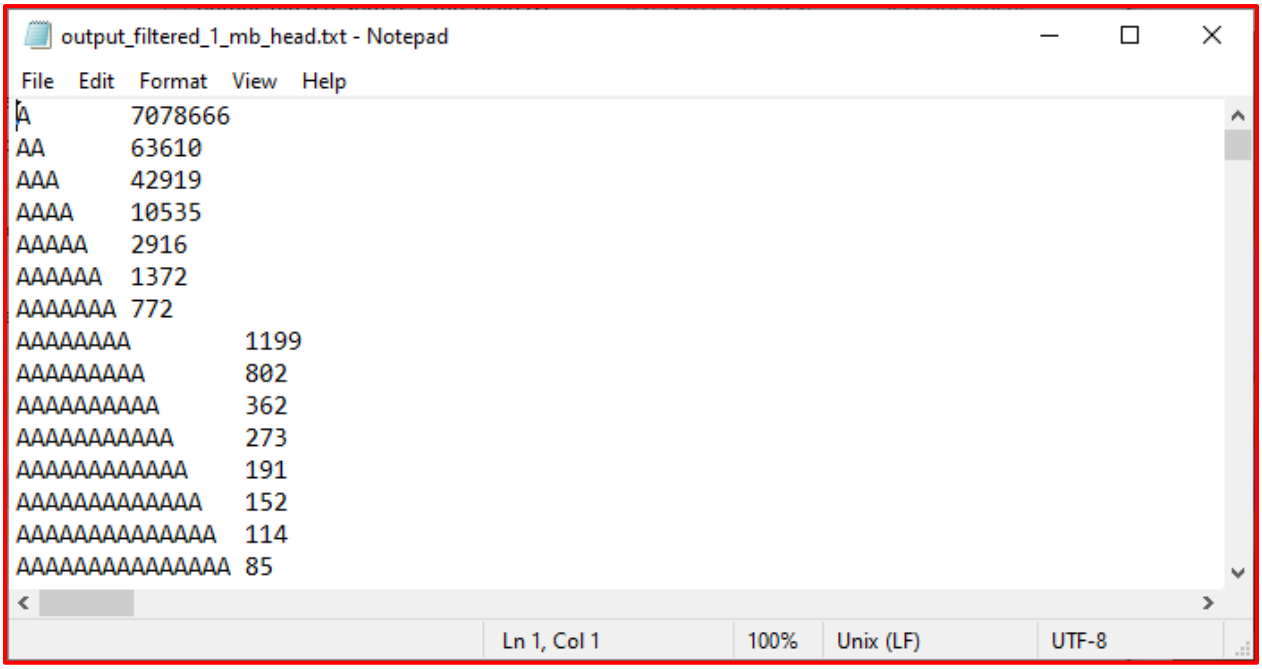
- فایل نهایی شمارش واژگان برمبنای MapReduce برای StackExchange-Posthistory با حجم خام حدود ۱۷۰ گیگابایت به بزرگی ۱۱ گیگابایت شد. این فایل به صورت خودکار فیلتر

شده براساس حروف الفبا است. اسکرین شاتی از فایل head ۱ مگابایتی این خروجی با عنوان output_raw_1_mb_head.txt:



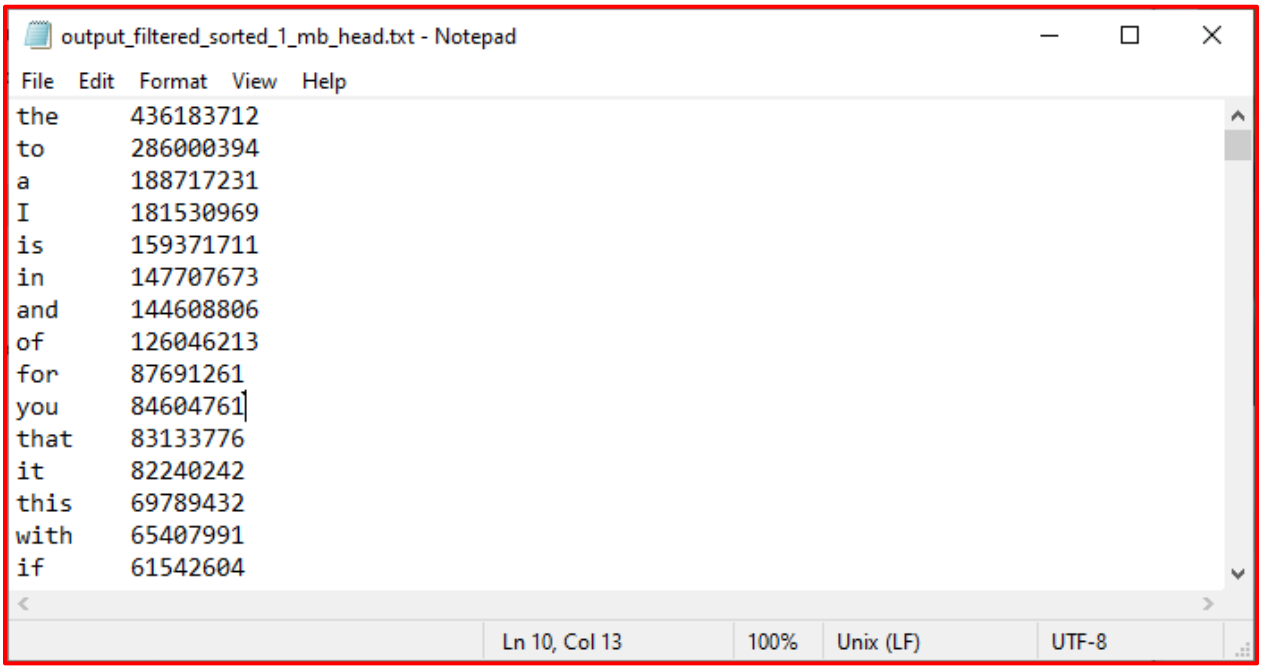
خروجی فیلتر شده

- این فایل را **فیلتر** کردیم، به شکلی که هر توکنی که شامل کاراکتری به غیر از ۲۶ کاراکتر زبان انگلیسی داشته باشد به کل آن خط حذف شود. طبیعتا حجم فایل بسیار کاهش یافت چون بسیاری از توکن ها سینتک های کد بودند. حجم فایل به حدود ۱۹۵ مگابایت کاهش یافت. اسکرین شاتی از فایل head ۱ مگابایتی این خروجی با عنوان output_filtered_1_mb_head.txt:



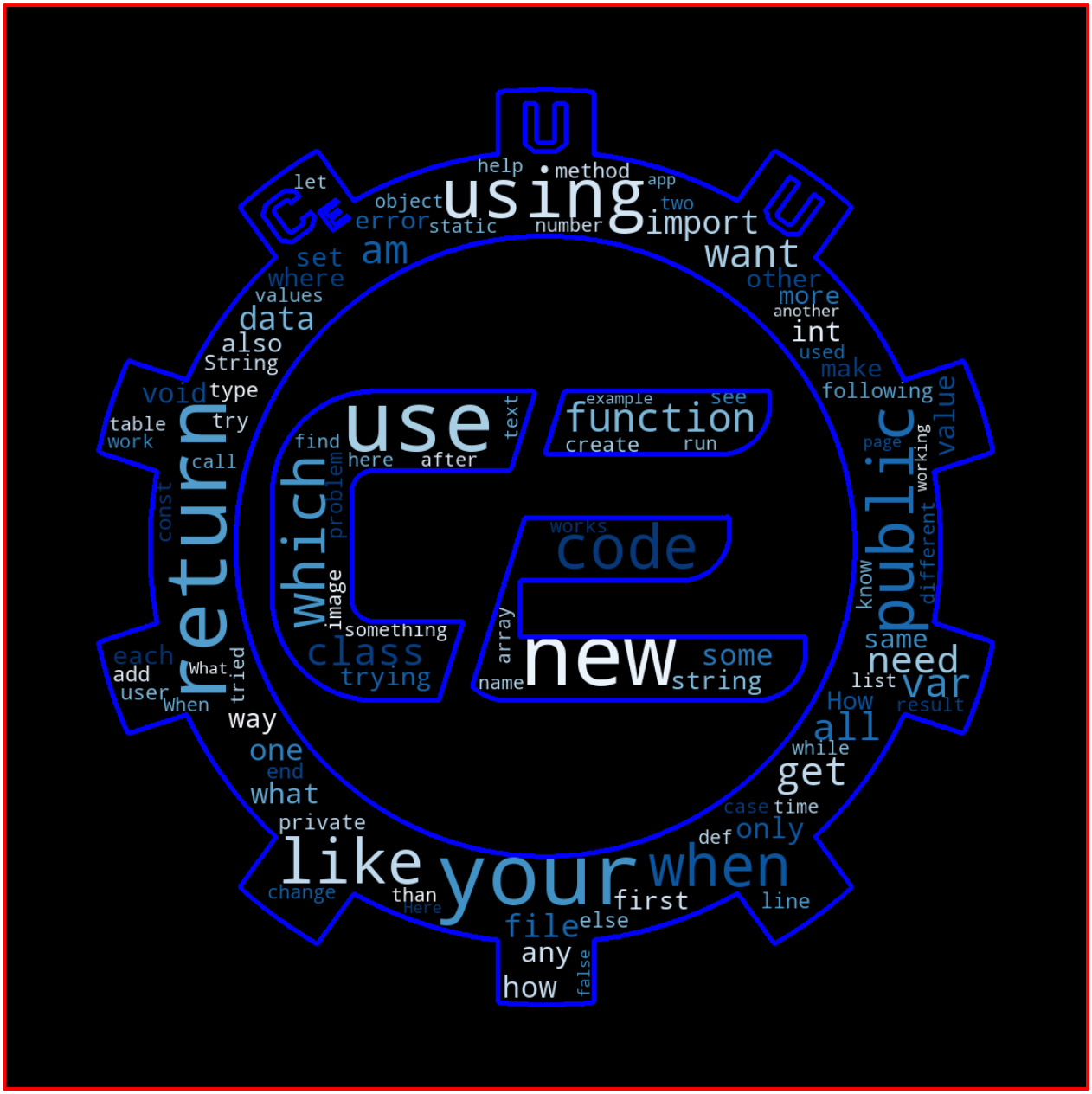
خروجی مرتب شده

- فایل فیلتر شده را براساس فراوانی توکن ها نیز **مرتب** کردیم. اسکرین شاتی از فایل head ۱ مگابایتی این خروجی با عنوان output_filtered_sorted_1_mb_head.txt:



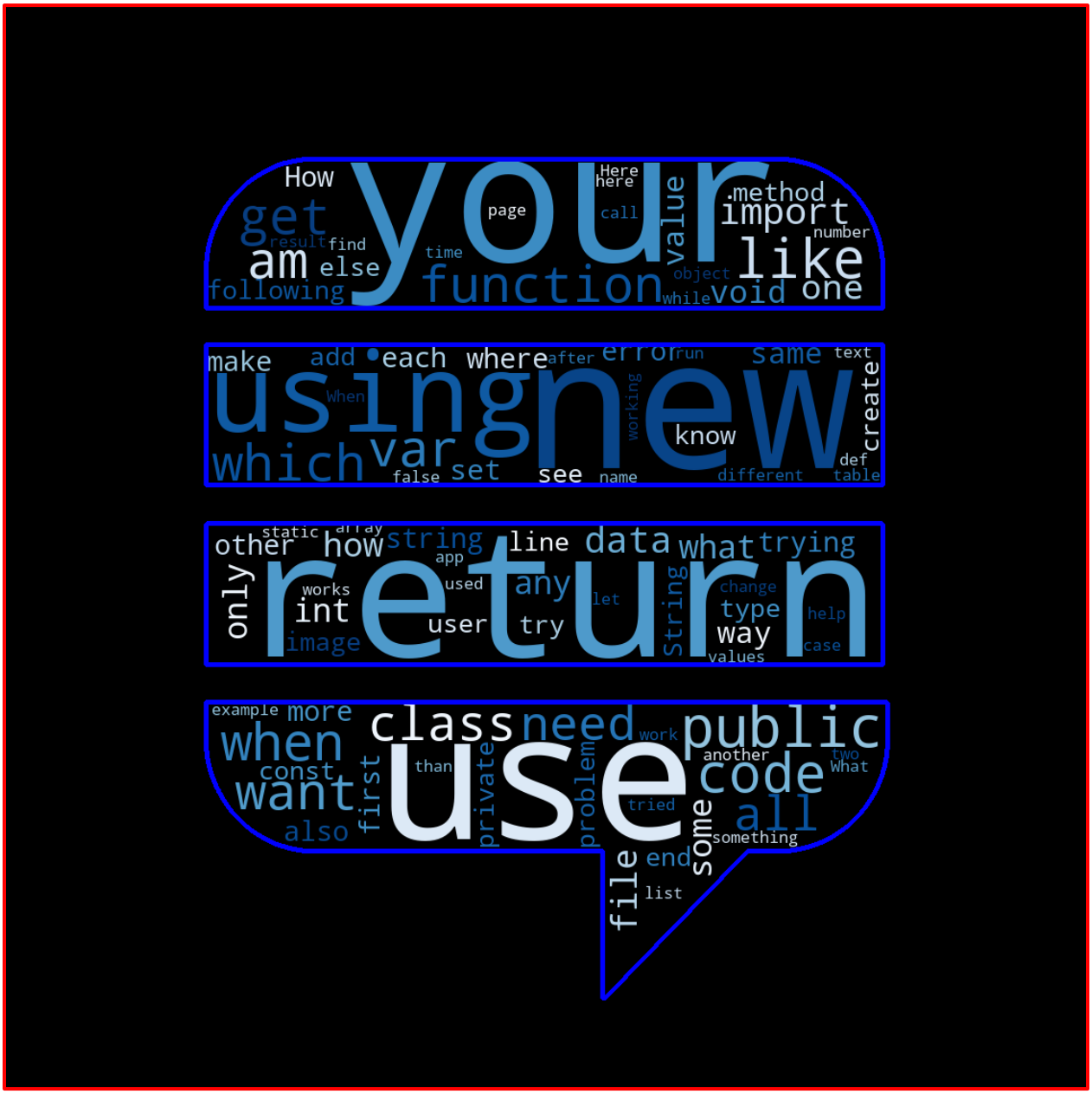
WordClouds

- در نهایت تمامی StopWords های زیر را از خروجی پاک کرده و براساس ۱۰۰ واژه نخست پرتکرار WordCloud هایی به شکل زیر ساختیم:



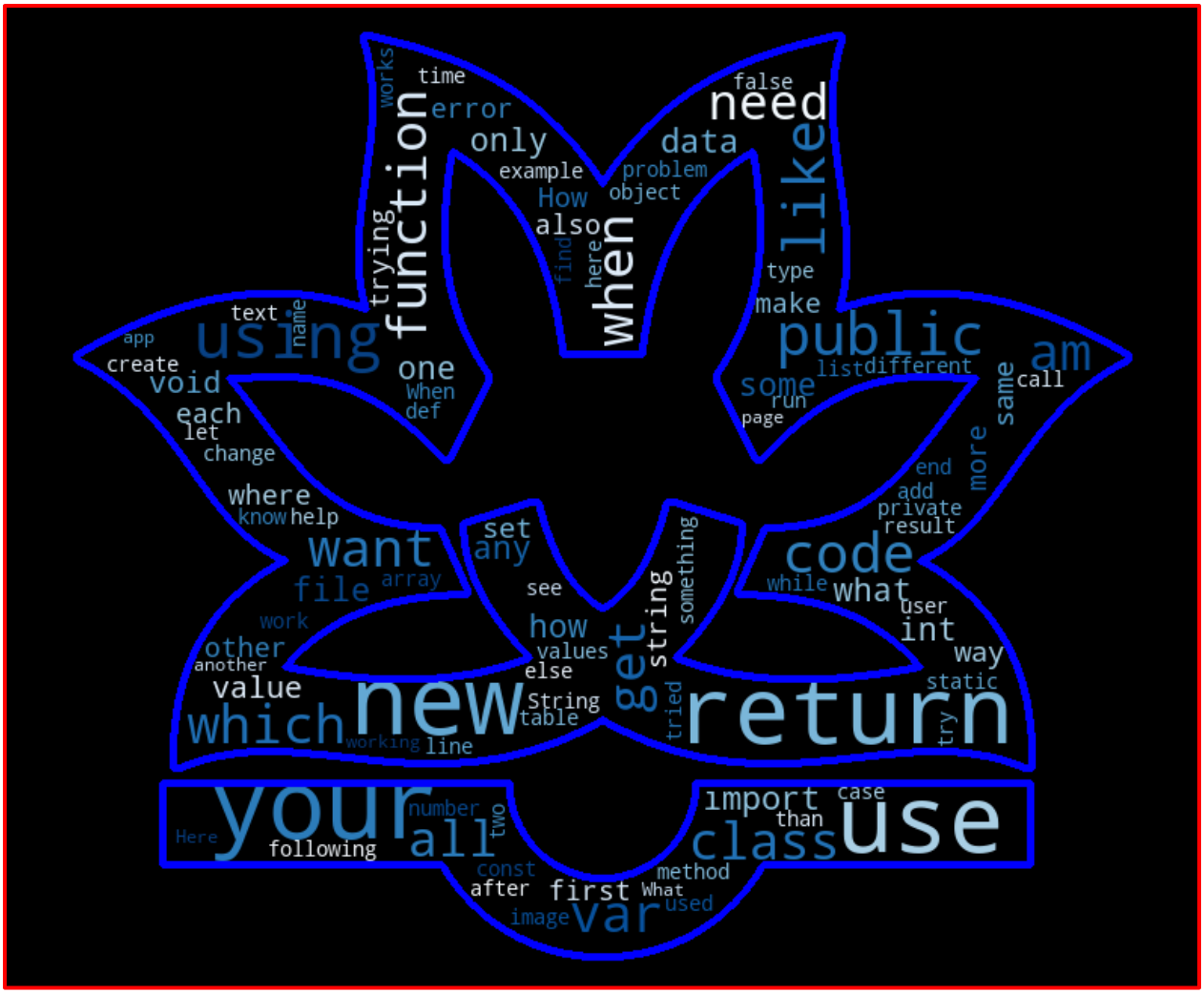
شکل ۱ لوگوی انجمن علمی گروه کامپیوتر دانشگاه ارومیه

شکل ۲ نقشه ایران



شکل ۳ لوگوی StackExchange

شکل ٤ نقشه دریاچه ارومیه



شکل ۵ لوگوی دانشگاه ارومیه