Map reduce

KELOMPOK 8
SISTEM BASIS DATA

Link github

https://github.com/Chokode/HadoopWordCount_K2

Anggota

ELDISJA HADASA
2106640133

HANDANESWARI PRAMUDHYTA IMANDA
2106731346

SYAUQI AULIYA MUHAMMAD
2106707201

MUHAMMAD AQIL MUZAKKY
2106731604

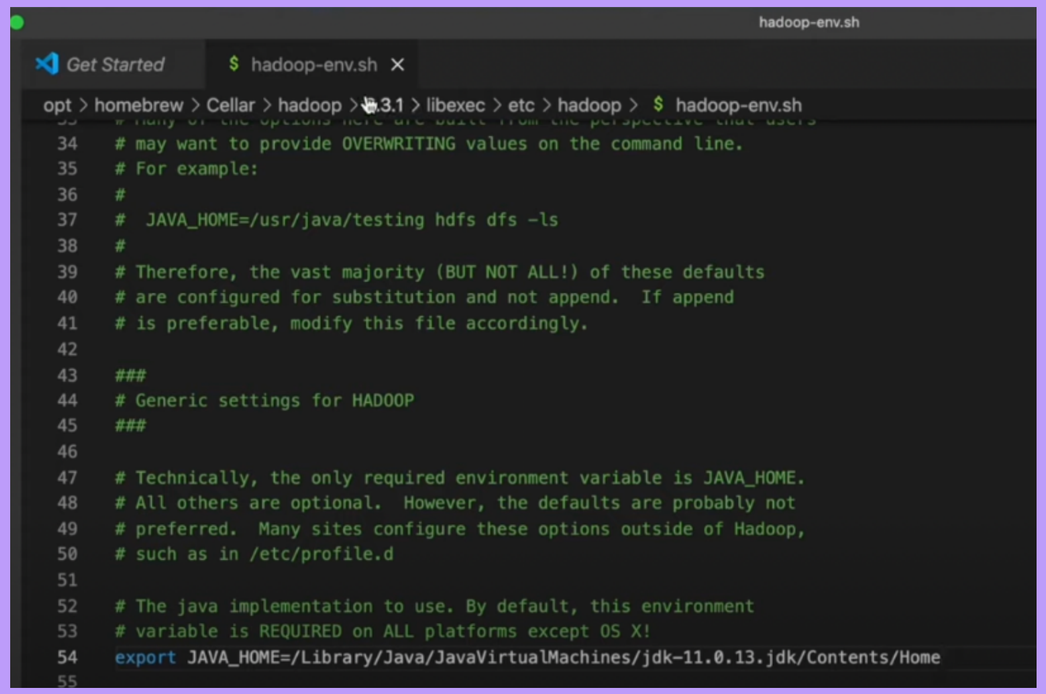
Instalasi Hadoop

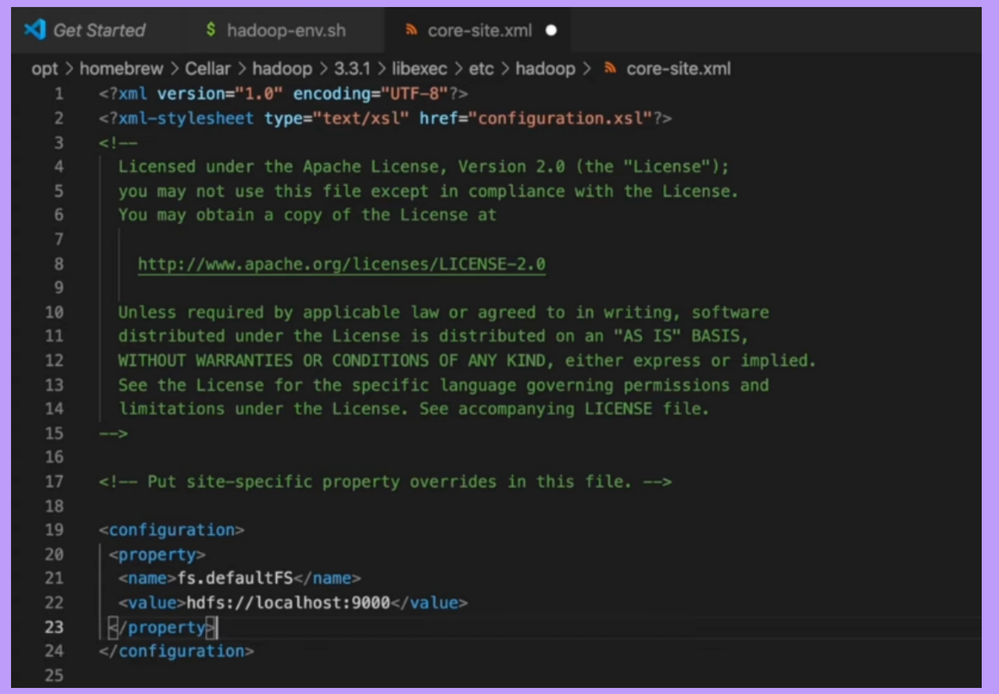
```
➜  ~ brew install hadoop
==> Downloading https://ghcr.io/v2/homebrew/core/hadoop/manifests/3.3.1
Already downloaded: /Users/arjuncodes/Library/Caches/Homebrew/downloads/
5d6f36248fca3674bcabe543982c55932cfdae628969660977994dbf--hadoop-3.3.1.b
nifest.json
==> Downloading https://ghcr.io/v2/homebrew/core/hadoop/blobs/sha256:913
Already downloaded: /Users/arjuncodes/Library/Caches/Homebrew/downloads/
900d93f2b8b7e9e7cc24d540f96c907ccddfdaad2271ff06e834781c--hadoop--3.3.1.
nterey.bottle.tar.gz
==> Pouring hadoop--3.3.1.arm64_monterey.bottle.tar.gz
🍺  /opt/homebrew/Cellar/hadoop/3.3.1: 22,487 files, 1GB
==> Running `brew cleanup hadoop`...
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.
Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man brew`).
➜  ~
```

```
Last login: Sat Jan 29 12:45:13 on ttys000
➜  ~ /usr/libexec/java_home
/Library/Java/JavaVirtualMachines/jdk-11.0.13.jdk/Contents/Home
➜  ~
```

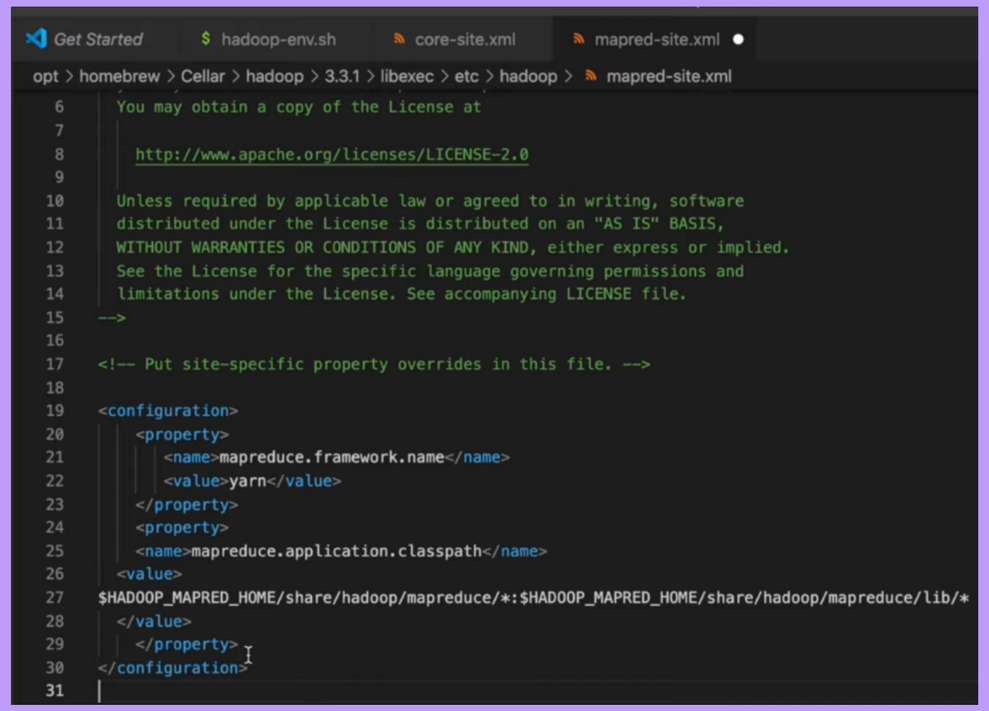
```
==> Running `brew cleanup hadoop`...
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.
Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man brew`).
➜  ~ cd /opt/homebrew/Cellar/hadoop/3.3.1/libexec/etc/hadoop
➜  hadoop git:(stable) ls
capacity-scheduler.xml          kms-log4j.properties
configuration.xsl               kms-site.xml
container-executor.cfg          log4j.properties
core-site.xml                   mapred-env.sh
hadoop-env.sh                   mapred-queues.xml.template
hadoop-metrics2.properties      mapred-site.xml
hadoop-policy.xml               shellprofile.d
hadoop-user-functions.sh.example ssl-client.xml.example
hdfs-rbf-site.xml               ssl-server.xml.example
hdfs-site.xml                   user_ec_policies.xml.template
httpfs-env.sh                   workers
httpfs-log4j.properties         yarn-env.sh
httpfs-site.xml                 yarn-site.xml
kms-acls.xml                    yarnservice-log4j.properties
kms-env.sh
➜  hadoop git:(stable)
```

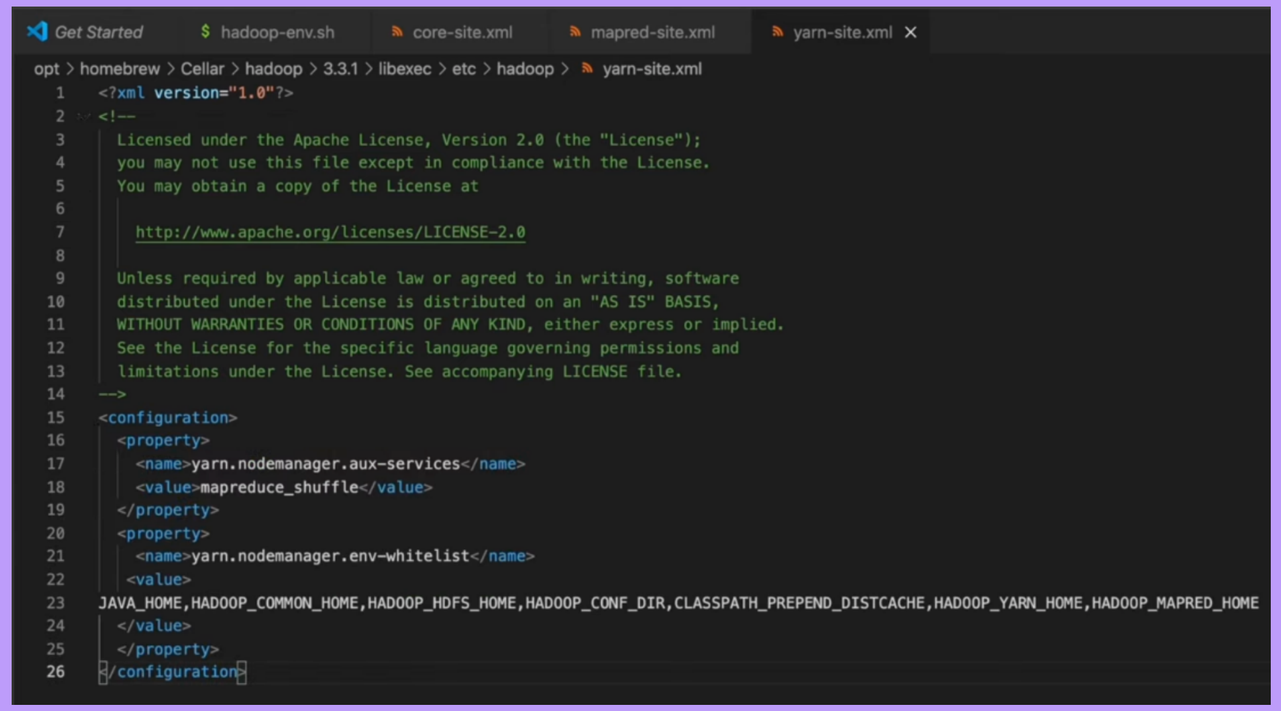
Instalasi Hadoop





Instalasi Hadoop





Instalasi Hadoop

```
➜  hadoop git:(stable) hadoop namenode -format
WARNING: Use of this script to execute namenode is deprecated.
WARNING: Attempting to execute replacement "hdfs namenode" instead.

WARNING: /opt/homebrew/Cellar/hadoop/3.3.1/libexec/logs does not exist. Creating
.
```

```
➜  ~ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as arjuncodes in 10 secon
ds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
```

```
Starting nodemanagers
➜  ~ jps
16513 NodeManager
16229 SecondaryNameNode
16581 Jps
15158 ResourceManager
15995 NameNode
16095 DataNode
➜  ~
```

```
localhost: arjuncodes@localhost: Permission denied (publ
interactive).
➜  ~ ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
```

```
interactive).
➜  ~ ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
Generating public/private rsa key pair.
Your identification has been saved in /Users/arjuncodes/.ssh/id_rsa
Your public key has been saved in /Users/arjuncodes/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:BARWRLu4h6V66GWhjoVEE13DP/jcZ02SG+GBzyHZAcA arjuncodes@Arjuns-MacBook-Pro
.local
The key's randomart image is:
+---[RSA 3072]----+
| .. =OB...=..    |
|  .o ..E + =     |
| o    + . = =    |
|. .  o =   B .   |
| .  o = S   *    |
|.. . * o . + .   |
|. o.* .   o      |
| +.+..           |
|..+.             |
+----[SHA256]-----+
➜  ~ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
➜  ~ start-all
```


```
interactive).
➜  ~ ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
Generating public/private rsa key pair.
Your identification has been saved in /Users/arjuncodes/.ssh/id_rsa
Your public key has been saved in /Users/arjuncodes/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:BARWRLu4h6V66GWhjoVEE13DP/jcZ02SG+GBzyHZAcA arjuncodes@Arjuns-MacBook-Pro
.local
The key's randomart image is:
+---[RSA 3072]----+
| .. =OB...=..    |
|  .o ..E + =     |
| o    + . = =    |
|. .  o =   B .   |
| .  o = S   *    |
|.. . * o . + .   |
|. o.* .   o      |
| +.+..           |
|..+.             |
+----[SHA256]-----+
➜  ~ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
➜  ~ start-all
```

Map reduce

- MapReduce adalah paradigma pemrograman yang digunakan untuk memproses data besar dengan efisien.
- Terdiri dari dua langkah utama, yaitu "map" dan "reduce".
- Pada langkah "map", data input diproses menjadi pasangan kunci-nilai yang lebih kecil.
- Langkah "reduce" menggabungkan pasangan kunci-nilai yang sama berdasarkan kunci yang sama.
- Proses MapReduce dilakukan di atas klaster komputer yang terdiri dari beberapa mesin, di mana setiap mesin berkontribusi dalam pemrosesan data.
- Kelebihan MapReduce adalah kemampuannya untuk memproses data dengan skala besar secara efisien dan paralel.
- Ini juga menawarkan toleransi kesalahan yang baik. MapReduce telah banyak digunakan dalam industri untuk pengolahan data besar, terutama dalam konteks Big Data.

File WordCount

Percobaan ini menggunakan 6 dataset dengan ukuran sebagai berikut.

- File 1 : 1 MB
- File 2 : 10 MB
- File 3 : 100 MB
- File 4 : 1000 MB
- File 5 : 5000 MB
- File 6 : 10000 MB

HADOOP

1 MB

Thu Jun 8 00:14:48 +0700 2023	Thu Jun 8 00:14:49 +0700 2023	Thu Jun 8 00:15:02 +0700 2023

TIME : 14 SECONDS..

10 MB

Thu Jun 8 00:02:43 +0700 2023	Thu Jun 8 00:02:44 +0700 2023	Thu Jun 8 00:03:08 +0700 2023

TIME : 25 SECONDS..

100 MB

Thu Jun 8 00:49:42 +0700 2023	Thu Jun 8 00:49:42 +0700 2023	Thu Jun 8 00:50:08 +0700 2023

TIME : 26 SECONDS..

1000 MB

Thu Jun 8 00:53:02 +0700 2023	Thu Jun 8 00:53:02 +0700 2023	Thu Jun 8 00:54:11 +0700 2023

TIME : 69 SECONDS..

5000 MB

Thu Jun 8 00:53:02 +0700 2023	Thu Jun 8 00:53:02 +0700 2023	Thu Jun 8 00:54:11 +0700 2023

TIME : 296 SECONDS..

10 000 MB

Thu Jun 8 01:05:45 +0700 2023	Thu Jun 8 01:05:46 +0700 2023	Thu Jun 8 01:19:36 +0700 2023

TIME : 831 SECONDS.

JAVA

1 MB

```
(base) syauqimuhammad@Chos-Mac WordCounter % java FileCounter
Number of lines: 1441
Number of words: 186121
Number of characters: 1119528
Running time: 198 milliseconds
```

TIME : 198 MILISECONDS..
= 0.198 SECONDS

10 MB

```
(base) syauqimuhammad@Chos-Mac WordCounter % java FileCounter
Number of lines: 14401
Number of words: 1861201
Number of characters: 11195280
Running time: 567 milliseconds
```

TIME : 567 MILISECONDS
= 0.567 SECONDS

100 MB

```
(base) syauqimuhammad@Chos-Mac WordCounter % java FileCounter
Number of lines: 129601
Number of words: 16750801
Number of characters: 100757520
Running time: 1389 milliseconds
```

TIME : 1389 MILISECONDS..
= 1.389 SECONDS

1000 MB

```
(base) syauqimuhammad@Chos-Mac WordCounter % java FileCounter
Number of lines: 1296001
Number of words: 167508001
Number of characters: 1007575200
Running time: 9770 milliseconds
```

TIME : 9770 MILISECONDS.
= 9.77 SECONDS

5000 MB

```
unt_b434cedc\bin' 'WordCounter'
Number of lines: 159327905
Number of words: 993626753
Number of characters: 548601016
Running time: 120157 milliseconds
```

TIME :120157 MILISECONDS
= 120 SECONDS

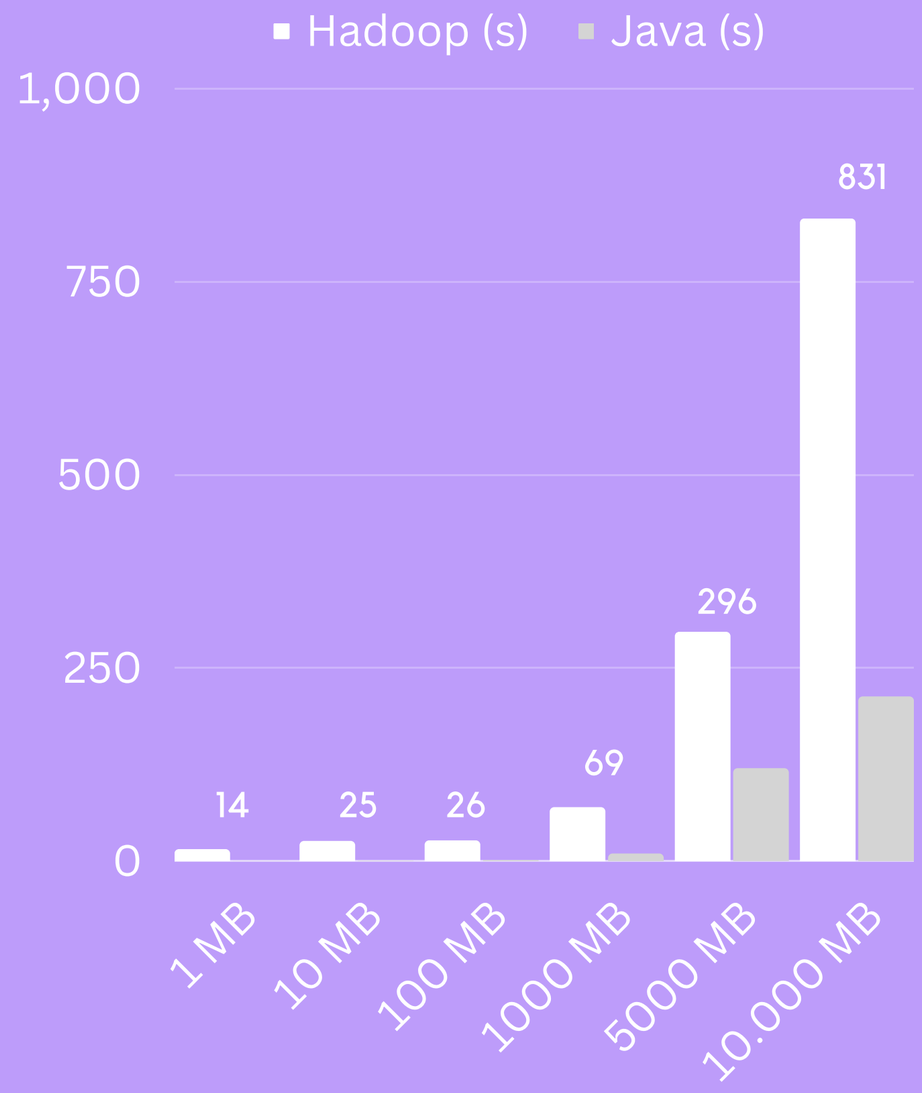
10 000 MB

```
f13b70c953a8e6d\redhat.java\jdt_ws\wordcount_b434cedc\bin' 'WordCounter
mber of lines: 318655810
mber of words: 1987253506
mber of characters: 1097202032
nning time: 213646 milliseconds
```

TIME : 213646 MILISECONDS
= 213 SECONDS

RUNNING TIME

Data (MB)	Time (s)	
	Hadoop	Java
1	14	0.198
10	25	0.567
100	26	1.389
1000	69	9.77
5000	296	120
10000	831	213



Analisa

Dari hasil yang didapatkan dapat dilihat peforma runtime Java jauh lebih cepat daripada Hadoop. Hal ini dikarenakan hadoop selalu menginisialisasi cluster, penjadwalan tugas, dan overhead komunikasi antar node. Sehingga, terjadi delay yang signifikan pada runtime relatif terhadap ukuran data yang sebenarnya. Mekanisme penyimpanan dan pembagian hadoop yang mengharuskan data dipartisi dan didistribusikan di antara beberapa node juga menyebabkan overhead dan memperlambat runtime program.

Berbeda dengan hadoop, Java memiliki pendekatan sekuensial dalam eksekusi kode. Pemrosesan sekuensial adalah pemrosesan data secara berurutan, tanpa overhead tambahan yang terkait dengan pemrosesan data terdistribusi. Oleh karena itu, wordcount pada kasus ini lebih cepat menggunakan Java karena file-file yang digunakan ukurannya relatif kecil untuk diproses dengan Hadoop.

Kesimpulan

Pada kasus kali ini, wordcount lebih cepat dilakukan oleh Java daripada Hadoop. Hal ini disebabkan oleh overhead konfigurasi pada Hadoop yang disebabkan ukuran file dianggap relatif kecil oleh Hadoop. Sehingga, overhead untuk inisialisasi dan penjadwalan task MapReduce pada Hadoop lebih terasa serta membuat peforma Hadoop lebih lambat dibandingkan dengan Java yang menggunakan pendekatan sekuen.