

Solution – Exercise 01

Hadoop, HDFS, YARN, MapReduce Example



Solution

Prerequisites:

- install Ubuntu 18.04
- Install Java JDK 1.8.0
- Create Hadoop user
- Install and Setup SSH (*public/private key authentication, authorized_keys, ...*)
- Install and Configure Hadoop 3.1.3 (*pseudo-distributed mode*)
- Start HDFS and YARN
- Clone Git Repo:

```
git clone https://github.com/marcelmittelstaedt/BigData.git
```

Solution

Exercise 2:

1. Copy sample file from GIT repo to **HDFS** user directory:

```
hadoop fs -put BigData/exercises/winter_semester_2019-2020/01_hadoop/sample_data/Faust_1.txt /user/hadoop/Faust_1.txt
```

2. Use and run default MapReduce Jar (*hadoop-mapreduce-examples-3.1.2.jar*) to calculate **wordcount** for text file „*Faust_1.txt*“.

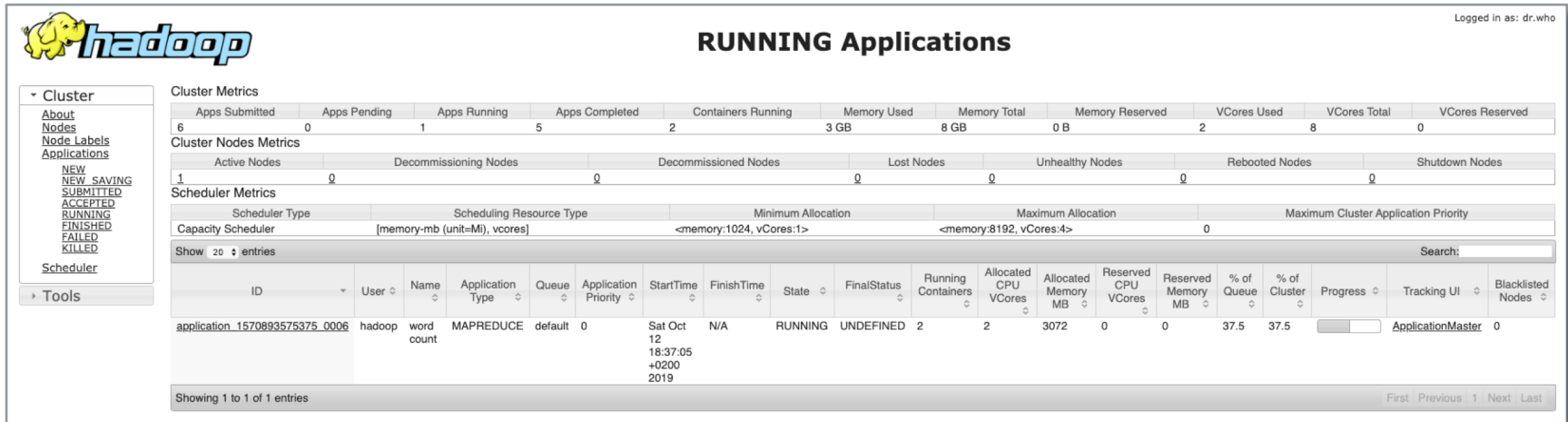
```
hadoop jar hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.1.2.jar wordcount /user/hadoop/Faust_1.txt /user/hadoop/Faust_1_Output
```

```
[...]  
2019-10-12 16:37:06,033 INFO mapreduce.Job: Running job: job_1570893575375_0006  
2019-10-12 16:37:12,157 INFO mapreduce.Job: Job job_1570893575375_0006 running in uber mode : false  
2019-10-12 16:37:12,158 INFO mapreduce.Job: map 0% reduce 0%  
2019-10-12 16:37:17,236 INFO mapreduce.Job: map 100% reduce 0%  
2019-10-12 16:37:22,279 INFO mapreduce.Job: map 100% reduce 100%  
2019-10-12 16:37:23,295 INFO mapreduce.Job: Job job_1570893575375_0006 completed successfully  
[...]
```

Solution

Exercise 2:

3. Take a look at Ressource Manager for Job Execution
(<http://XXX.XXX.XXX.XXX:8088/cluster/apps/RUNNING>):



The screenshot displays the Hadoop YARN Resource Manager web interface. The top left features the Hadoop logo and a navigation menu with links like 'About', 'Nodes', 'Node Labels', 'Applications', and 'Scheduler'. The main title is 'RUNNING Applications'. Below this, there are several summary tables: 'Cluster Metrics' showing 6 apps submitted, 1 pending, 1 running, and 5 completed; 'Cluster Nodes Metrics' showing 1 active node; and 'Scheduler Metrics' showing capacity and scheduling details. The main section is a table of running applications, with one entry visible: 'application_1570893575375_0006' by user 'hadoop', running a 'word count' MAPREDUCE job. The bottom of the interface shows pagination controls and a search bar.

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved
6	0	1	5	2	3 GB	8 GB	0 B	2	8	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Shutdown Nodes
1	0	0	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Cluster Application Priority
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>	<memory:8192, vCores:4>	0

Show 20 entries

ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU Vcores	Allocated Memory MB	Reserved CPU Vcores	Reserved Memory MB	% of Queue	% of Cluster	Progress	Tracking UI	Blacklisted Nodes
application_1570893575375_0006	hadoop	word count	MAPREDUCE	default	0	Sat Oct 12 18:37:05 +0200 2019	N/A	RUNNING	UNDEFINED	2	2	3072	0	0	37.5	37.5	<div></div>	ApplicationMaster	0

Showing 1 to 1 of 1 entries

Solution

Exercise 2:

4. a) Copy MapReduce output file back to ubuntu local filesystem (using bash):

```
hadoop fs -get /user/hadoop/Faust_1_Output/part-r-00000 Faust_1_Output.csv
```

```
shuf -n 10 Faust_1_Output.csv
```

```
Phantasie,          1
unwanden. 1
winden, 1
Offenbarung,       2
Undene! 1
Winternächte       1
derweil 1
wiederholten       1
tun 3
Gestalten.         1
```

Solution

Exercise 2:

4. **b)** Copy MapReduce output file back to ubuntu local filesystem (using Web Filebrowser):

The screenshot shows the Hadoop Web File Browser interface. The top navigation bar includes links for Hadoop, Overview, Datanodes, Datanode Volume Failures, Snapshot, Startup Progress, and Utilities. The main heading is "Browse Directory". Below it, the path "/user/hadoop/Faust_1_Output" is entered in the search bar. The "Go!" button is visible. Below the search bar, there are icons for file operations and a search input field. The table displays the following entries:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hadoop	supergroup	0 B	Oct 12 18:37	1	128 MB	._SUCCESS
-rw-r--r--	hadoop	supergroup	98.49 KB	Oct 12 18:37	1	128 MB	part-r-00000

Showing 1 to 2 of 2 entries. Navigation buttons: Previous, 1, Next.

In the foreground, a terminal window titled "part-r-00000" is open, displaying the contents of the file. The output is a list of words and their frequencies:

```
786 tilet 1
787 Eimer 1
788 Ein 97
789 Eine 2
790 Einen 4
791 Eimer 1
792 Einerlei. 1
793 Einfalt, 1
794 Einige 1
795 Einklang 1
796 Einmal 2
797 Eins 3
798 Eins! 1
799 Eins, 2
800 Einsamkeit 1
```



Solution

Exercise 3:

1. Copy sample file from GIT repo to **HDFS** user directory:

```
hadoop fs -put BigData/exercises/winter_semester_2019-2020/01_hadoop/sample_data/Faust_1.txt /user/hadoop/Faust_1.txt
```

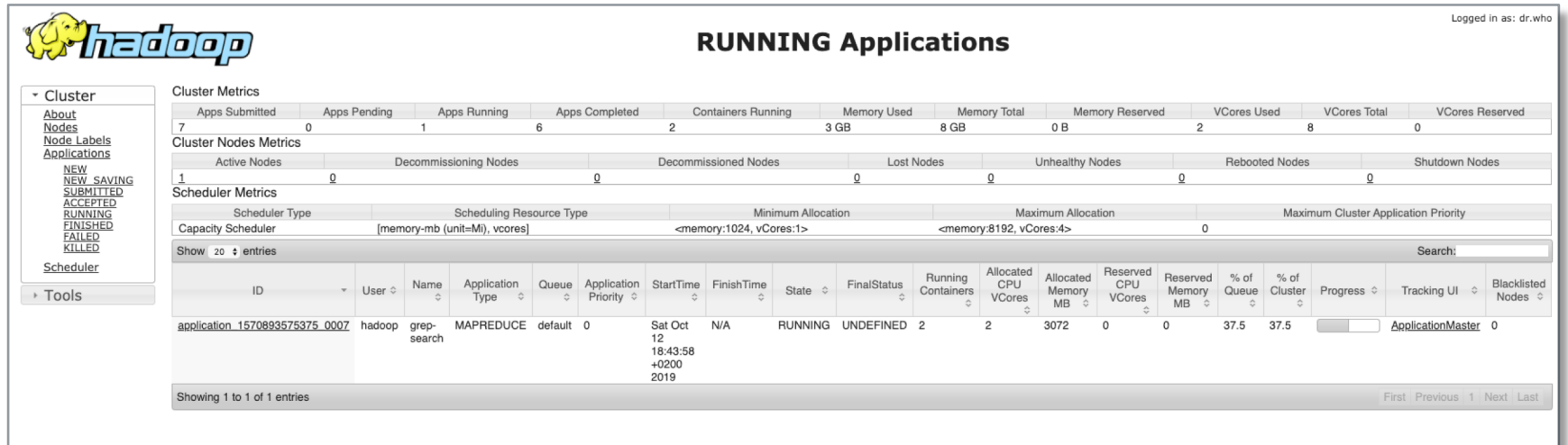
2. Use and run default MapReduce Jar (*hadoop-mapreduce-examples-3.1.2.jar*) to **grep** for string „Faust“ in text file „*Faust_1.txt*“ and count appearances of string.

```
hadoop jar hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.1.1.jar grep /user/hadoop/Faust_1.txt /user/hadoop/Faust_1_Count_Output 'Faust'
```

```
[...]  
2019-10-12 16:44:16,517 INFO mapreduce.Job: Running job: job_1570893575375_0008  
2019-10-12 16:44:27,637 INFO mapreduce.Job: Job job_1570893575375_0008 running in uber mode : false  
2019-10-12 16:44:27,638 INFO mapreduce.Job: map 0% reduce 0%  
2019-10-12 16:44:31,678 INFO mapreduce.Job: map 100% reduce 0%  
2019-10-12 16:44:36,717 INFO mapreduce.Job: map 100% reduce 100%  
2019-10-12 16:44:37,735 INFO mapreduce.Job: Job job_1570893575375_0008 completed successfully  
[...]
```

Exercise 3:

3. Take a look at Ressource Manager for Job Execution
(<http://XXX.XXX.XXX.XXX:8088/cluster/apps/RUNNING>):



hadoop

Cluster

- About
- Nodes
- Node Labels
- Applications
- NEW
- NEW SAVING
- SUBMITTED
- ACCEPTED
- RUNNING
- FINISHED
- FAILED
- KILLED
- Scheduler

Tools

RUNNING Applications

Logged in as: dr.who

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved
7	0	1	6	2	3 GB	8 GB	0 B	2	8	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Shutdown Nodes
1	0	0	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Cluster Application Priority
Capacity Scheduler	[memory-mb (unit=M), vcores]	<memory:1024, vCores:1>	<memory:8192, vCores:4>	0

Show 20 entries

ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Reserved CPU VCores	Reserved Memory MB	% of Queue	% of Cluster	Progress	Tracking UI	Blacklisted Nodes
application_1570893575375_0007	hadoop	grep-search	MAPREDUCE	default	0	Sat Oct 12 18:43:58 +0200 2019	N/A	RUNNING	UNDEFINED	2	2	3072	0	0	37.5	37.5	<div></div>	ApplicationMaster	0

Showing 1 of 1 entries

First Previous 1 Next Last

Solution

Exercise 2:

4. a) Copy MapReduce output file back to ubuntu local filesystem (using bash):

```
hadoop fs -get /user/hadoop/Faust_1_Count_Output/part-r-00000 Faust_1_Count_Output.csv
```

```
cat Faust_1_Count_Output.csv
```

```
50          Faust
```

Solution

Exercise 2:

4. b) Copy MapReduce output file back to ubuntu local filesystem (using Web Filebrowser):

The screenshot displays the Hadoop Web File Browser interface. The top navigation bar includes links for Hadoop, Overview, Datanodes, Datanode Volume Failures, Snapshot, Startup Progress, and Utilities. The main heading is "Browse Directory". Below this, a search bar contains the path "/user/hadoop/Faust_1_Count_Output" with a "Go!" button and icons for file operations. A "Show 25 entries" dropdown and a "Search:" input field are also present. The main content area shows a table of files with columns for Permission, Owner, Group, Size, Last Modified, Replication, Block Size, and Name. Two files are listed: "_SUCCESS" (0 B) and "part-r-00000" (9 B). A terminal window is overlaid on the bottom left, showing the command "part-r-00000" and its output "1 50 Faust" and "2".

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hadoop	supergroup	0 B	Oct 12 18:48	1	128 MB	_SUCCESS
-rw-r--r--	hadoop	supergroup	9 B	Oct 12 18:48	1	128 MB	part-r-00000

MapReduce Examples within *hadoop-mapreduce-examples-3.1.1.jar*:

aggregatewordcount:	An Aggregate based mapreduce program that counts the words in the input files.
aggregatewordhist:	An Aggregate based mapreduce program that computes the histogram of the words in the input files.
bbp:	A mapreduce program that uses Bailey-Borwein-Plouffe to compute exact digits of Pi.
dbcount:	An example job that counts the pageview logs stored in a database.
distbbp:	A mapreduce program that uses a BBP-type formula to compute exact bits of Pi.
grep:	A mapreduce program that counts the matches of a regex in the input.
join:	A job that performs a join over sorted, equally partitioned datasets.
multifilewc:	A job that counts words from several files.
pentomino:	A mapreduce tile laying program to find solutions to pentomino problems.
pi:	A mapreduce program that estimates Pi using a quasi-Monte Carlo method.
randomtextwriter:	A mapreduce program that writes 10 GB of random textual data per node.
randomwriter:	A mapreduce program that writes 10 GB of random data per node.
secondarysort:	An example defining a secondary sort to the reduce phase.
sort:	A mapreduce program that sorts the data written by the random writer.
sudoku:	A sudoku solver.
teragen:	Generate data for the terasort.
terasort:	Run the terasort.
teravalidate:	Checking results of terasort.
wordcount:	A mapreduce program that counts the words in the input files.
wordmean:	A mapreduce program that counts the average length of the words in the input files.
wordmedian:	A mapreduce program that counts the median length of the words in the input files.
wordstandarddeviation:	A mapreduce program that counts the standard deviation of the length of the words in the input files.