北京邮电大学软件学院

2020-2021 学年第二学期实验报告

课程名称： 大数据原理与技术

项目名称： 实验 1：安装单机 Hadoop 系统

项目完成人：

姓名： 马瑞遥 **学号：** 2018211915

指导教师： 孙鹏飞

日　　期： 2021 年 3 月 22 日

一、 实验目的

1) 掌握在自己本地电脑上正确安装和运行伪分布式 Hadoop 系统的方法。
2) 掌握运行 Hadoop 系统自带的 WordCount 可执行程序文件的方法,并产生输出结果。
3) 掌握程序运行后在 Hadoop Web 作业状态查看界面上作业运行状态的方法。
4) 掌握 Hadoop 的相关命令，掌握管理 HDFS 文件系统的方法。

二、 实验内容

1) 每人在自己本地电脑上正确安装和运行伪分布式 Hadoop 系统。
2) 安装完成后,自己寻找一组英文网页数据,在本机上运行 Hadoop 系统自带的 WordCount 可执行程序文件,并产生输出结果。
3) 实验结果提交：要求书写一个实验报告，其中包括：
   a) 系统安装运行的情况
   b) 实验数据说明（下载的什么网页数据，多少个 HTML 或 text 文件)
   c) 程序运行后在 Hadoop Web 作业状态查看界面上的作业运行状态屏幕拷贝
   d) 实验输出结果开头部分的屏幕拷贝
   e) 实验体会

三、 实验环境

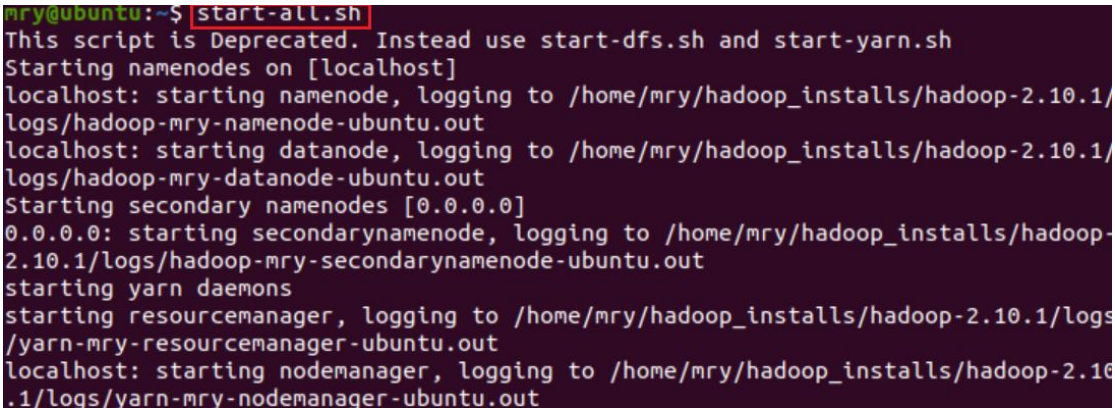
1) 一台 hp 计算机（HP Spectre x360 Convertible 13-ap0xxx）
2) VMware Workstation Pro 软件
3) Linux 系统主机（Ubuntu 20.04.2 虚拟机）
4) OpenSSH_8.2p1

5） Java 1.8.0_281

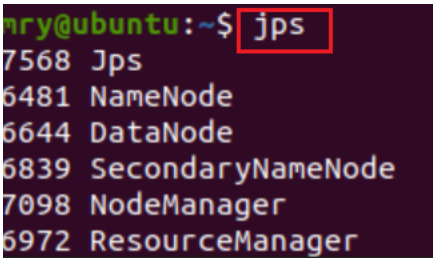
四、 实验结果

4.1 系统安装运行的情况

1） 在 Linux 命令行中输入命令：start-all.sh，启动 Hadoop。



2） 之后，输入命令：jps，以检查 Hadoop 是否成功启动。



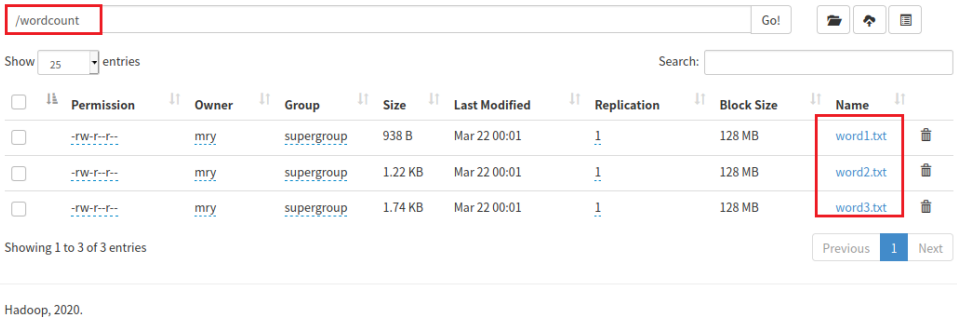
由上图可知，NameNode，DataNode，SecondaryNameNode，NodeManager，ResourceManager 都已成功启动，即 Hadoop 已成功启动。

至此，已成功在本地电脑（Ubuntu 20.04.2 虚拟机）上正确安装和运行伪分布式 Hadoop 系统。

4.2 实验数据说明

1） 本次实验的实验数据是三个.txt 形式的英文文本文件，这三个文本文件来自于三篇英文论文的摘要。分别将这三个文本文件命名为 word1，

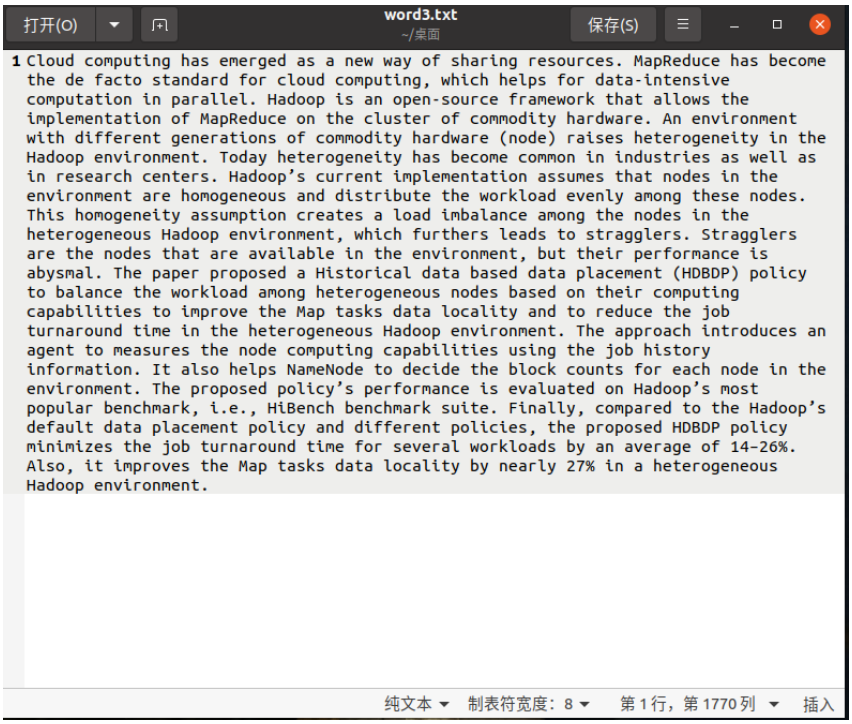
word2，word3，并使用 Hadoop 的 put 命令（如命令，hadoop fs -put /home/mry/Desktop/word1.txt /wordcount，前面为本地文件路径，后面为 HDFS 中的存放路径）将其存放在 HDFS 的 wordcount 目录下。



2) word1 中的内容是 Analysis of hadoop MapReduce scheduling in heterogeneous environment 这篇文章的摘要，其文件内容展示如下所示。

3) word2 中的内容是 Investigating Automatic Parameter Tuning for SQL-on-Hadoop Systems 这篇文章的摘要，其文件内容展示如下所示。

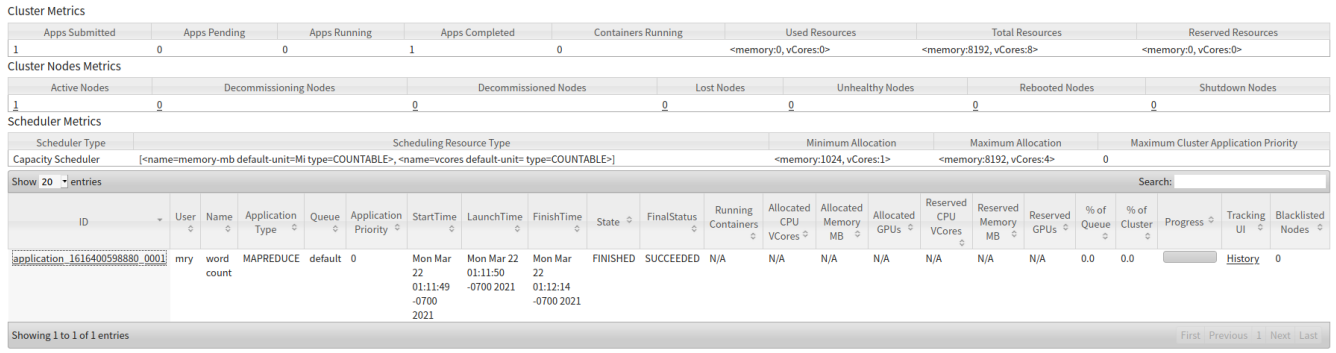
4) word3 中的内容是 Historical data based approach for straggler avoidance in a heterogeneous Hadoop cluster 这篇文章的摘要，其文件内容展示如下。



上述三个英文文本文件的相关链接及完整内容会在附录中给出。

4.3 程序运行后在Hadoop Web作业状态查看界面上的作业运行状态

运行 Hadoop 中自带的 WordCount 程序，对之前提交在 HDFS 中的 wordcount 目录进行词频计数，并将运行结果指定存放在 HDFS 中的 outputwc 目录下（如命令：hadoop jar hadoop-mapreduce-examples-2.10.1.jar wordcount /wordcount /outputwc，分别是可执行程序文件本地路径，调用方法名，HDFS 中的待处理数据路径，实验输出结果路径）。运行结束后，在 localhost:8088 中，查看其作业运行状态，发现有一个已完成的作业，其信息如下所示。



Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Used Resources	Total Resources	Reserved Resources
1	0	0	1	0	<memory:0, vCores:0>	<memory:8192, vCores:8>	<memory:0, vCores: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

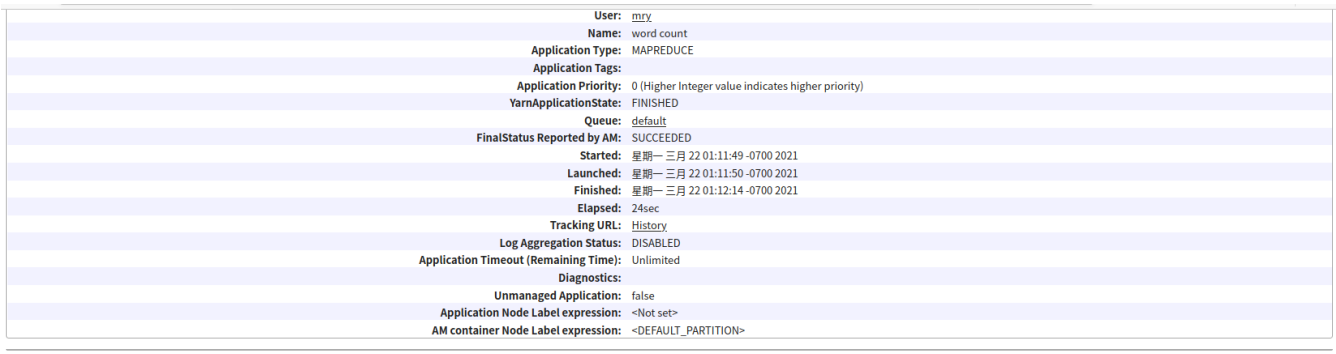
Scheduler Metrics

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

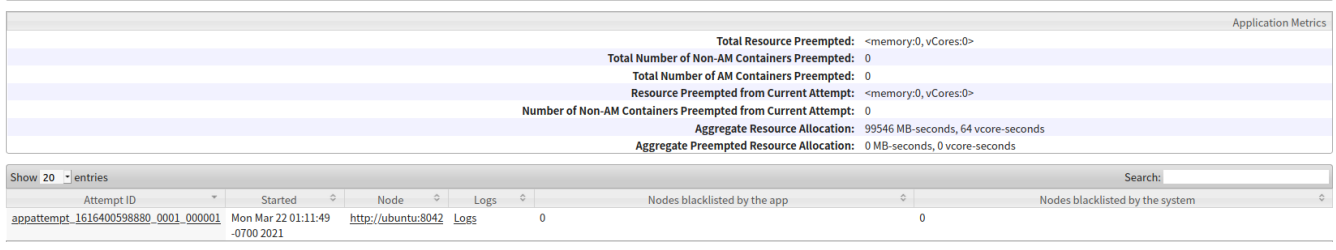
Show 20 entries Search:

ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Allocated GPUs	Reserved CPU VCores	Reserved Memory MB	Reserved GPUs	% of Queue	% of Cluster	Progress	Tracking UI	Blacklisted Nodes
application_1616400598880_0001	mry	word count	MAPREDUCE	default	0	Mon Mar 22 01:11:49 -0700 2021	Mon Mar 22 01:11:50 -0700 2021	Mon Mar 22 01:12:14 -0700 2021	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0	0.0		History	0

Showing 1 to 1 of 1 entries First Previous 1 Next Last



User:	mry
Name:	word count
Application Type:	MAPREDUCE
Application Tags:	
Application Priority:	0 (Higher Integer value indicates higher priority)
YarnApplicationState:	FINISHED
Queue:	default
FinalStatus Reported by AM:	SUCCEEDED
Started:	星期一 三月 22 01:11:49 -0700 2021
Launched:	星期一 三月 22 01:11:50 -0700 2021
Finished:	星期一 三月 22 01:12:14 -0700 2021
Elapsed:	24sec
Tracking URL:	History
Log Aggregation Status:	DISABLED
Application Timeout (Remaining Time):	Unlimited
Diagnostics:	
Unmanaged Application:	false
Application Node Label expression:	<Not set>
AM container Node Label expression:	<DEFAULT_PARTITION>



Application Metrics

Total Resource Preempted:	<memory:0, vCores:0>
Total Number of Non-AM Containers Preempted:	0
Total Number of AM Containers Preempted:	0
Resource Preempted from Current Attempt:	<memory:0, vCores:0>
Number of Non-AM Containers Preempted from Current Attempt:	0
Aggregate Resource Allocation:	99546 MB-seconds, 64 vcore-seconds
Aggregate Preempted Resource Allocation:	0 MB-seconds, 0 vcore-seconds

Show 20 entries Search:

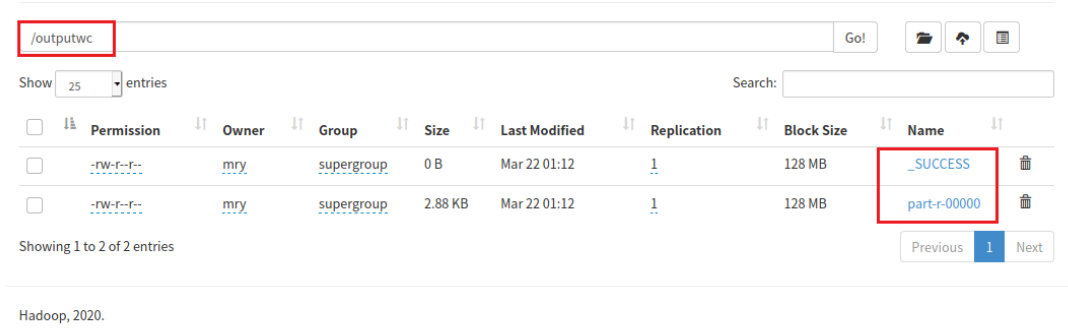
Attempt ID	Started	Node	Logs	Nodes blacklisted by the app	Nodes blacklisted by the system
appattempt_1616400598880_0001_000001	Mon Mar 22 01:11:49 -0700 2021	http://ubuntu:8042	Logs	0	0

4.4 实验输出结果

1) WordCount 程序运行过程中，Linux 命令行中的实验输出结果如下所示。

```
21/03/22 01:12:15 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=4847
                FILE: Number of bytes written=843975
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=4284
                HDFS: Number of bytes written=2946
                HDFS: Number of read operations=12
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=3
                Launched reduce tasks=1
                Data-local map tasks=3
                Total time spent by all maps in occupied slots (ms)=25682
                Total time spent by all reduces in occupied slots (ms)=3455
                Total time spent by all map tasks (ms)=25682
                Total time spent by all reduce tasks (ms)=3455
                Total vcore-milliseconds taken by all map tasks=25682
                Total vcore-milliseconds taken by all reduce tasks=3455
                Total megabyte-milliseconds taken by all map tasks=26298368
                Total megabyte-milliseconds taken by all reduce tasks=3537920
        Map-Reduce Framework
                Map input records=3
                Map output records=585
                Map output bytes=6308
                Map output materialized bytes=4859
                Input split bytes=318
                Combine input records=585
                Combine output records=359
                Reduce input groups=297
                Reduce shuffle bytes=4859
                Reduce input records=359
                Reduce output records=297
                Spilled Records=718
                Shuffled Maps =3
                Failed Shuffles=0
                Merged Map outputs=3
```

2) 程序运行结束后，在 localhost:50070 中，查看 outputwc 目录下的实验输出结果，发现有_SUCCESS 文件和 part-r-00000 文件，表目程序运行成功，其输出结果存放在 part-r-00000 文件中。



3) part-r-00000 文件中的输出结果如下（完整文件内容将在附录中给出）。

```
 1 (HDBDP)        1
 2 (node)  1
 3 -       1
 4 14-26%.        1
 5 27%     1
 6 A       1
 7 After   1
 8 Also,   1
 9 An      1
10 Apache  2
11 Cloud   1
12 Dryad,  1
13 Finally,      1
14 Furthermore,  1
15 Google  1
16 HDBDP   1
17 Hadoop  9
18 Hadoop,       1
19 Hadoop-focused  1
20 Hadoop.       2
21 Hadoop’s      3
22 HiBench       1
23 Historical    1
24 Hive    4
25 Hive.   1
26 In      1
27 It      1
28 Map     2
29 MapReduce     6
30 Microsoft     1
31 NameNode      1
32 Numerous      1
33 Over    1
34 Proper  1
35 SQL-on-Hadoop   2
36 Stragglers      1
```

五、 附录

5.1 实验体会

通过本次实验，我明白了如何在 Linux 系统上安装和运行伪分布式 Hadoop 系统，掌握了如何运行 Hadoop 系统自带的可执行程序文件（WordCount）的方法，并能够在 localhost:8088 中查看作业运行状态，在 localhost:50070 中管理 HDFS 的分布式文件及相应的实验输出结果。同时，我知道了 Hadoop 中的相关命令操作（start-all.sh，jps 等），并掌握了上传文件到 HDFS 及运行 Hadoop 可执行程序文件的方法。

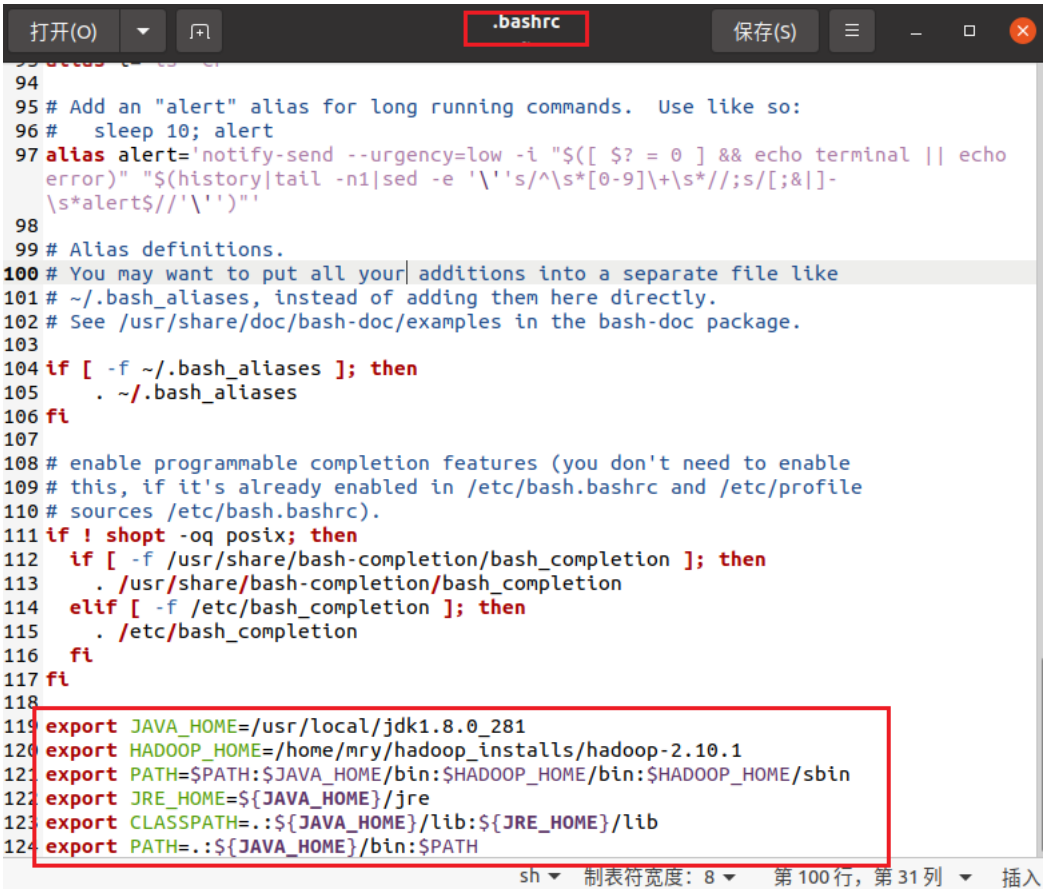
5.2 调试心得

5.2.1 bash_profile 文件不是当前 Linux 的环境配置文件

在进行 Hadoop 环境变量的配置，并按照 ppt 中所说打开环境配置文件 bash_profile 时，发现打开了一个空文件，即 bash_profile 文件不存在，不是当前 Linux 系统的环境配置文件。

查阅相关资料后发现，本电脑中的虚拟机版本是 Ubuntu 20.04.2，其环境变量的相关配置不在 bash_profile 文件中，而在 bashrc 文件中。成功找到并打开 bashrc 文件后（gedit ~/.bashrc），便可进行相关的环境变量配置。

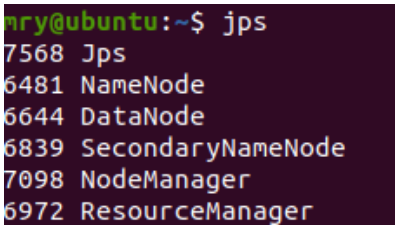
```
mry@ubuntu:~$ gedit ~/.bashrc
```



5.2.2 输入 start-all.sh 命令后，启动的进程与 ppt 上的不完全一致

在输入 start-all.sh 命令以启动 Hadoop 时，发现自己虚拟机中启动的 Hadoop 的进程是：NameNode，DataNode，SecondaryNameNode，NodeManager，

ResourceManager;而 ppt 中所展示的 Hadoop 的启动进程是:NameNode,DataNode, SecondaryNameNode, TaskTracker, JobTracker。二者不完全一致。



4706 JobTracker
4582 SecondaryNameNode
4278 NameNode
4413 DataNode
4853 TaskTracker
4889 Jps

查阅资料后发现，Hadoop 启动进程的差异是由版本号（Yarn 的有无等）所导致的。在 Hadoop 一代版本中，还未引入 Yarn 框架，启动时会有 TaskTracker 和 JobTracker 这两个进程，即 ppt 中所展示的；而在 Hadoop 二代版本中，已从 MapReduce 中分离出了 Yarn 框架负责资源管理等，启动时便不会有 TaskTracker 和 JobTracker 这两个进程，取而代之的是 NodeManager 和 ResourceManager 这两个进程，即我自己虚拟机所跑出的结果。

5.2.3 运行 WordCount 程序时卡住不动，无法继续运行

在运行 Hadoop 自带的 WordCount 程序时，发现程序运行到一半时卡住不动了，无法继续执行。

上网搜集了资料后发现，原因是运行程序时，当剩余硬盘容量不足设定阈值时，会判定该节点坏了，由于是伪分布式，所以任务会被无限搁置。解决方法是在 yarn-site.xml 中添加配置信息如下：

```
<property>
        <name>yarn.nodemanager.disk-health-checker.max-disk-utilization-per-disk-percentage</name>
        <value>95.0</value>
</property>
```

这之后，WordCount 程序便可顺利运行。

5.3 实验数据

5.3.1 word1.txt

1) 链接

https://doi.org/10.1016/j.asej.2020.06.009

2) 完整内容

Over the last decade, several advancements have happened in distributed and parallel computing. A lot of data is generated daily from various sources, and this speedy data proliferation led to the development of many more frameworks that are efficient to handle such huge data e.g. - Microsoft Dryad, Apache Hadoop, etc. Apache Hadoop is an open-source application of Google MapReduce and is getting a lot of attention from various researchers. Proper scheduling of jobs needs to be done for better performance. Numerous efforts have been done in the development of existing MapReduce schedulers and in developing new optimized techniques or algorithms. This paper focuses on the Hadoop MapReduce framework, its shortcomings, various issues we face while scheduling jobs to nodes and algorithms proposed by various researchers. Furthermore, we then classify these algorithms on various quality measures that affect MapReduce performance.

5.3.2 word2.txt

1) 链接

https://doi.org/10.1016/j.bdr.2021.100204

2) 完整内容

SQL-on-Hadoop engines such as Hive provide a declarative interface for processing large-scale data over computing frameworks such as Hadoop. The underlying frameworks contain a large number of configuration parameters that can significantly impact performance, but which are hard to tune. The problem of automatic parameter tuning has become a lively research area and several sophisticated tuning advisors have been proposed for Hadoop. In this paper, we conduct an experimental study to explore the impact of Hadoop parameter tuning on Hive. We reveal that the performance of Hive queries does not necessarily improve when using Hadoop-focused tuning advisors out-of-the-box, at least when following the current approach of applying the same tuning setup uniformly for evaluating the entire query. After extending the Hive query processing engine, we propose an alternative tuning approach and experimentally show how current Hadoop tuning advisors can now provide good and robust performance for Hive queries, as well as improved cluster resource utilization. We share our observations with the community and hope to create an awareness for this problem as well as to initiate new research on automatic parameter tuning for SQL-on-Hadoop systems.

5.3.3 word3.txt

1) 链接

https://doi.org/10.1007/s12652-020-02699-0

2) 完整内容

Cloud computing has emerged as a new way of sharing resources. MapReduce has become the de

facto standard for cloud computing, which helps for data-intensive computation in parallel. Hadoop is an open-source framework that allows the implementation of MapReduce on the cluster of commodity hardware. An environment with different generations of commodity hardware (node) raises heterogeneity in the Hadoop environment. Today heterogeneity has become common in industries as well as in research centers. Hadoop's current implementation assumes that nodes in the environment are homogeneous and distribute the workload evenly among these nodes. This homogeneity assumption creates a load imbalance among the nodes in the heterogeneous Hadoop environment, which furthers leads to stragglers. Stragglers are the nodes that are available in the environment, but their performance is abysmal. The paper proposed a Historical data based data placement (HDBDP) policy to balance the workload among heterogeneous nodes based on their computing capabilities to improve the Map tasks data locality and to reduce the job turnaround time in the heterogeneous Hadoop environment. The approach introduces an agent to measures the node computing capabilities using the job history information. It also helps NameNode to decide the block counts for each node in the environment. The proposed policy's performance is evaluated on Hadoop's most popular benchmark, i.e., HiBench benchmark suite. Finally, compared to the Hadoop's default data placement policy and different policies, the proposed HDBDP policy minimizes the job turnaround time for several workloads by an average of 14–26%. Also, it improves the Map tasks data locality by nearly 27% in a heterogeneous Hadoop environment.

5.4 实验输出结果（part-r-00000 文件内容）

```
(HDBDP)	1
(node)	1
-	1
14–26%.	1
27%	1
A	1
After	1
Also,	1
An	1
Apache	2
Cloud	1
Dryad,	1
Finally,	1
Furthermore,	1
Google	1
HDBDP	1
Hadoop	9
Hadoop,	1
Hadoop-focused	1
Hadoop.	2
Hadoop's	3
HiBench	1
```

Historical 1
Hive 4
Hive. 1
In 1
It 1
Map 2
MapReduce 6
Microsoft 1
NameNode 1
Numerous 1
Over 1
Proper 1
SQL-on-Hadoop 2
Stragglers 1
The 5
This 2
Today 1
We 2
a 8
abysmal. 1
advancements 1
advisors 3
affect 1
agent 1
algorithms 2
algorithms. 1
allows 1
also 1
alternative 1
among 3
an 7
and 12
application 1
applying 1
approach 3
are 5
area 1
as 9
assumes 1
assumption 1
at 1
attention 1
automatic 2
available 1

average 1
awareness 1
balance 1
based 2
be 1
become 3
been 2
benchmark 1
benchmark, 1
better 1
block 1
but 2
by 3
can 2
capabilities 2
centers. 1
classify 1
cloud 1
cluster 2
commodity 2
common 1
community 1
compared 1
computation 1
computing 4
computing, 1
computing. 1
conduct 1
configuration 1
contain 1
counts 1
create 1
creates 1
current 3
daily 1
data 9
data-intensive 1
de 1
decade, 1
decide 1
declarative 1
default 1
developing 1
development 2

different 2
distribute 1
distributed 1
does 1
done 2
e.g. 1
each 1
efficient 1
efforts 1
emerged 1
engine, 1
engines 1
entire 1
environment 2
environment, 2
environment. 4
etc. 1
evaluated 1
evaluating 1
evenly 1
existing 1
experimental 1
experimentally 1
explore 1
extending 1
face 1
facto 1
focuses 1
following 1
for 11
framework 1
framework, 1
frameworks 3
from 2
furthers 1
generated 1
generations 1
getting 1
good 1
handle 1
happened 1
hard 1
hardware 1
hardware. 1

has 4
have 3
helps 2
heterogeneity 2
heterogeneous 4
history 1
homogeneity 1
homogeneous 1
hope 1
how 1
huge 1
i.e., 1
imbalance 1
impact 2
implementation 2
improve 2
improved 1
improves 1
in 13
industries 1
information. 1
initiate 1
interface 1
introduces 1
is 6
issues 1
it 1
its 1
job 3
jobs 2
large 1
large-scale 1
last 1
leads 1
least 1
led 1
lively 1
load 1
locality 2
lot 2
many 1
measures 2
minimizes 1
more 1

most 1
nearly 1
necessarily 1
needs 1
new 3
node 2
nodes 5
nodes. 1
not 1
now 1
number 1
observations 1
of 16
on 7
open-source 2
optimized 1
or 1
our 1
out-of-the-box, 1
over 1
paper 2
paper, 1
parallel 1
parallel. 1
parameter 3
parameters 1
performance 4
performance, 1
performance. 2
placement 2
policies, 1
policy 3
policy’s 1
popular 1
problem 2
processing 2
proliferation 1
propose 1
proposed 5
provide 2
quality 1
queries 1
queries, 1
query 1

query. 1
raises 1
reduce 1
research 3
researchers. 2
resource 1
resources. 1
reveal 1
robust 1
same 1
schedulers 1
scheduling 2
setup 1
several 3
share 1
sharing 1
shortcomings, 1
show 1
significantly 1
sophisticated 1
sources, 1
speedy 1
standard 1
stragglers. 1
study 1
such 3
suite. 1
systems. 1
tasks 2
techniques 1
that 7
the 33
their 2
then 1
these 2
this 3
time 2
to 15
tune. 1
tuning 8
turnaround 2
underlying 1
uniformly 1
using 2

utilization. 1
various 5
way 1
we 4
well 3
when 2
which 3
while 1
with 2
workload 2
workloads 1