

《云计算技术》说明文档

1951121 宁之恒 1952748郑启帆

1.架构说明

- 4台云服务器

配置 - 2台4核8G华为云，1台2核2G阿里云，1台2核4G腾讯云

操作系统 - CentOS

With Hadoop 3.3.1, Hive 3.1.3, Spark 2.3.0

组件分布如下

	A	B	C	D
1	Node01 华为云	Node02 华为云	Node03 阿里云	Node04 腾讯云
2	Namenode	Secondarynamenode		
3	Resourcemanager			
4	Datanode	Datanode	Datanode	Datanode
5	Hive			
6	Mysql			
7	spark	spark	spark	spark

1.1文件分块方法

在hdfs-site.xml文件中配置dfs.blocksize属性，将文件分成大小统一的block存储(64M),寻址时间=1/1000读取时间时性能较好，如果某个block小于64m，实际占用空间为该block实际大小。

1.2文件备份方法

在hdfs-site.xml文件中配置dfs.replication属性，每个block在系统中有3份完全相同的备份。

个队列可以配置一定的资源，每个队列中的job任务公平共享其所在队列的所有资源，队列中的job任务都是按照优先级分配资源，优先级越高分配的资源越多，但是为了确保公平每个job任务都会分配到资源。优先级是根据每个job任务的理想获取资源量减去实际获取资源量的差值决定的，差值越大优先级越高。

```
<property>
  <name>yarn.resourcemanager.scheduler.class</name>
  <value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FairScheduler</value>
</property>
<property>
  <name>yarn.scheduler.fair.user-as-default-queue</name>
  <value>>false</value>
</property>
"/opt/hadoop/etc/hadoop/yarn-site.xml" 71L, 2234C
```

Hadoop web UI

Hadoop

Overview

Datanodes

Datanode Volume Failures

Snapshot

Startup Progress

Utilities

Overview

'ecs-329d-0001:9000' (active)

Started:	Mon Dec 20 16:12:09 +0800 2021
Version:	3.3.1, ra3b9c37a397ad4188041dd80621bdeefc46885f2
Compiled:	Tue Jun 15 13:13:00 +0800 2021 by ubuntu from (HEAD detached at release-3.3.1-RC3)
Cluster ID:	CID-ce2d647a-523f-4c9a-94e1-918ced626dde
Block Pool ID:	BP-1424625122-192.168.0.166-1639730666975

Summary

Security is off.

Safemode is off.

367 files and directories, 276 blocks (276 replicated blocks, 0 erasure coded block groups) = 643 total filesystem object(s).

Heap Memory used 54.57 MB of 275.5 MB Heap Memory. Max Heap Memory is 1.69 GB.

Non Heap Memory used 77.69 MB of 79.75 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:	333.91 GB
Configured Remote Capacity:	0 B
DFS Used:	1.98 GB (0.59%)
Non DFS Used:	23.16 GB
DFS Remaining:	294.02 GB (88.05%)
Block Pool Used:	1.98 GB (0.59%)
DataNodes usages% (Min/Median/Max/stdDev):	0.48% / 0.61% / 0.67% / 0.07%
Live Nodes	4 (Decommissioned: 0, In Maintenance: 0)
Dead Nodes	0 (Decommissioned: 0, In Maintenance: 0)

NameNode Journal Status

Current transaction ID: 10138

Journal Manager	State
FileJournalManager(root=/opt/hadoop/hadoopdata/dfs/name)	EditLogFileOutputStream(/opt/hadoop/hadoopdata/dfs/name/current/edits_inprogress_00000000000000010138)

NameNode Storage

Storage Directory	Type	State
/opt/hadoop/hadoopdata/dfs/name	IMAGE_AND_EDITS	Active

DFS Storage Types

Storage Type	Configured Capacity	Capacity Used	Capacity Remaining	Block Pool Used	Nodes in Service
DISK	333.91 GB	1.98 GB (0.59%)	294.02 GB (88.05%)	1.98 GB	4

Hadoop, 2021.

Datanode Information

- ✓ In service

⬇ Down

🔄 Decommissioning

🛑 Decommissioned

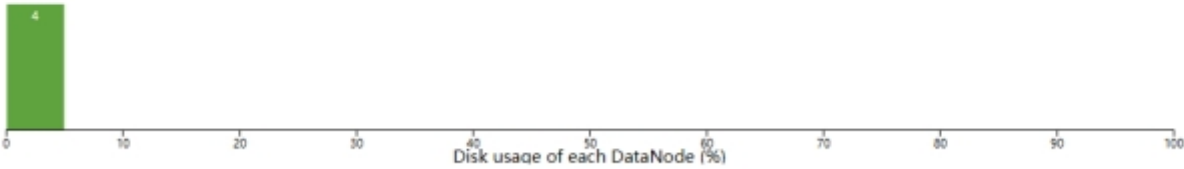
⛔ Decommissioned & dead

🔧 Entering Maintenance

🔧 In Maintenance

🔧 In Maintenance & dead

Datanode usage histogram



In operation

DataNode State: All

Show 25 entries

Search:

Node	Http Address	Last contact	Last Block Report	Used	Non DFS Used	Capacity	Blocks	Block pool used	Version
✓ /default-rack/node04:9866 (175.24.202.153:9866)	http://node04:9864	0s	305m	448.31 MB	5.54 GB	78.62 GB	172	448.31 MB (0.56%)	3.3.1
✓ /default-rack/ecs-329d-0001:9866 (192.168.0.166:9866)	http://ecs-329d-0001:9864	1s	110m	676.58 MB	8.39 GB	98.18 GB	272	676.58 MB (0.67%)	3.3.1
✓ /default-rack/node03:9866 (106.15.248.113:9866)	http://node03:9864	2s	325m	291.29 MB	3.95 GB	58.93 GB	197	291.29 MB (0.48%)	3.3.1
✓ /default-rack/ecs-329d-0002:9866 (124.71.199.129:9866)	http://ecs-329d-0002:9864	1s	142m	616.34 MB	5.28 GB	98.18 GB	167	616.34 MB (0.61%)	3.3.1

Showing 1 to 4 of 4 entries

Previous

1

Next

Yarn web UI



All Applications

- Cluster
- About
- Nodes
- Node Labels
- Applications
- NEW
- NEW SAVING
- SUBMITTED
- ACCEPTED
- SUBMITTING
- PENDING
- FAILED
- KILLED
- Scheduler

Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Used Resources	Total Resources	Base
0	0	0	0	0	<memory:0 B, vCores:0>	<memory:9.08 GB, vCores:7>	<memory:0 B,

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes
4	0	0	0	0

User Metrics for root

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Containers Pending	Containers Reserved	Memory Used	Memory Pending
0	0	0	0	0	0	0	0 B	0 B

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Fair Scheduler	[memory-mb (unit=MB), vcores]	<memory:512, vCores:1>	<memory:2560, vCores:2>

Show 20 entries

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU V-Cores	Allocated Memory MB	Allocated GPUs	Reserved CPU V-Cores
----	------	------	------------------	------------------	-------	----------------------	-----------	------------	------------	-------	-------------	--------------------	-----------------------	---------------------	----------------	----------------------

No data available in table

Showing 0 to 0 of 0 entries



All Applications

- Cluster
- About
- Nodes
- Node Labels
- Applications
- NEW
- NEW SAVING
- SUBMITTED
- ACCEPTED
- SUBMITTING
- PENDING
- FAILED
- KILLED
- Scheduler

Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Used Resources	Total Resources
2	0	1	1	2	<memory:4.50 GB, vCores:3>	<memory:9.08 GB, vCores:7>

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes
4	0	0	0

User Metrics for root

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Containers Pending	Containers Reserved	Memory Used
2	0	1	1	2	2	1	4.50 GB


Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Fair Scheduler	[memory-mb (unit=MB), vcores]	<memory:512, vCores:1>	<memory:2560, vCores:2>

Show 20 entries

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU V-Cores
application_1649144720978_0002	root	Hive on Spark sessionId = f21196d4- e936-40f3- 8b34- 8a9fbb2281e	SPARK		root.default	0	Wed Dec 22 11:50:01 +0800 2021	Wed Dec 22 11:50:02 +0800 2021	N/A	RUNNING	UNDEFINED	2	3

Spark web UI

 2.3.0

Jobs

Stages

Storage

Environment

Executors

Hive on Spark (sessionId = f2119... application UI)

Spark Jobs (7)

User: root
Total Uptime: 3.0 min
Scheduling Mode: FIFO
Active Jobs: 1

Event Timeline

Active Jobs (1)

Job ID (Job Group)	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
0 (queryId = root_38211222116857_591c171-67b1-4856-a2a0-4ccab53de904)	Reducer 3 Reducer 3	2021/12/22 11:51:57 (kill)	1.1 min	0/0	0/18

 2.3.0

Jobs

Stages

Storage

Environment

Executors

Hive on Spark (sessionId = f2119... application UI)

Stages for All Jobs

Active Stages: 1

Pending Stages: 2

Active Stages (1)

Stage ID	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
0	select count(*) count calling nbr from ... 100 (Stage-1) Map 1	2021/12/22 11:51:57 (kill)	1.7 min	0/1 (1 running)	55.7 MB			6.6 MB

Pending Stages (2)

Stage ID	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
2	Reducer 3	Unknown	Unknown	0/1				
1	Reducer 2	Unknown	Unknown	0/16				

```
Query Hive on Spark job[0] stages: [0, 1, 2]
Spark job[0] status = RUNNING
```

STAGES	ATTEMPT	STATUS	TOTAL	COMPLETED	RUNNING	PENDING	FAILED
Stage-0	0	RUNNING	1	0	1	0	0
Stage-1	0	PENDING	16	0	0	16	0
Stage-2	0	PENDING	1	0	0	1	0

STAGES: 00/03 [>>-----] 0% ELAPSED TIME: 51.27 s

```
█
```

2.系统功能展示

创建一个test.txt

```
[root@ecs-329d-0001 ~]# vim test.txt
[root@ecs-329d-0001 ~]# cat test.txt
fjdkfdsj
[root@ecs-329d-0001 ~]# █
```

上传到hdfs

```
[root@ecs-329d-0001 ~]# vim test.txt
[root@ecs-329d-0001 ~]# cat test.txt
fjdkfdsj
[root@ecs-329d-0001 ~]# hadoop fs -put test.txt /user/
[root@ecs-329d-0001 ~]# █
```

Browse Directory

/user

Go!

Show

25

entries

Search:

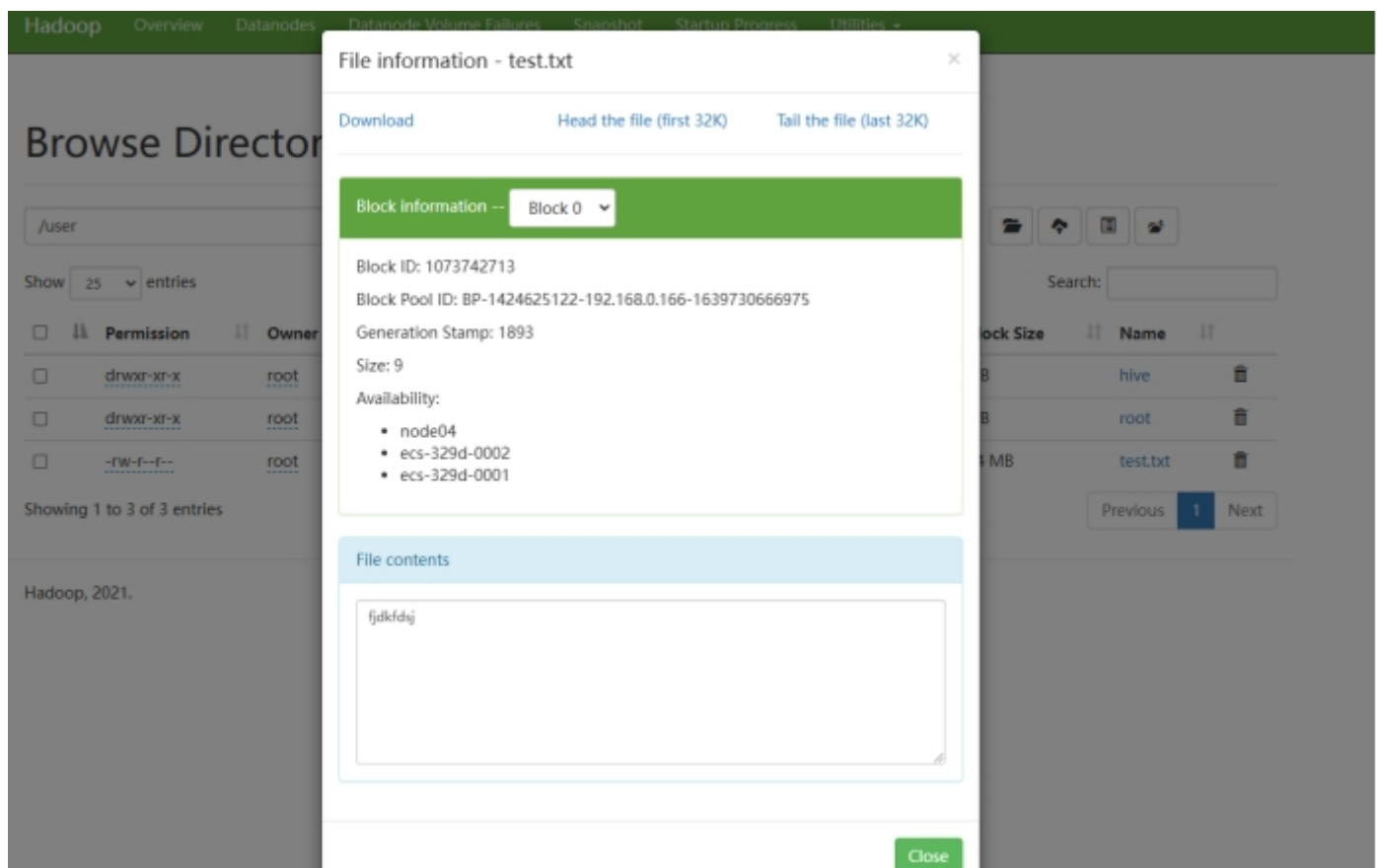
<input type="checkbox"/>	<div><div></div></div> Permission	<div><div></div></div> Owner	<div><div></div></div> Group	<div><div></div></div> Size	<div><div></div></div> Last Modified	<div><div></div></div> Replication	<div><div></div></div> Block Size	<div><div></div></div> Name	<div><div></div></div>
<input type="checkbox"/>	drwxr-xr-x	root	supergroup	0 B	Dec 17 17:51	0	0 B	hive	<div><div></div></div>
<input type="checkbox"/>	drwxr-xr-x	root	supergroup	0 B	Dec 17 16:55	0	0 B	root	<div><div></div></div>
<input type="checkbox"/>	-rw-r--r--	root	supergroup	9 B	Dec 22 15:20	3	64 MB	test.txt	<div><div></div></div>

Showing 1 to 3 of 3 entries

Previous

1

Next

[查看副本](#)

Node01

[illegible]

Node02

```

1 华为云01 2 华为云02 3 阿里云01 4 腾讯云04
Huawei 7 (Build 8060)
Copyright (c) 2020 Huawei Computer, Inc. All rights reserved.

Type 'help' to learn how to use Huawei prompt.
[Ctrl]-[B]

Connecting to 124.71.199.129:22...
Connection established.
To escape to local shell, press 'Ctrl+Alt+.'.

WARNING: The remote SSH server rejected X11 forwarding request.

Welcome to Huawei Cloud Service

Last failed login: Wed Dec 22 09:12:30 CST 2021 from 110.0.180.66 on ssh-notty
There were 3 failed login attempts since the last successful login.
Last login: Tue Dec 21 10:22:30 2021 from 111.107.14.24
[rest@ecs-329d-0002 ~]$ cd /opt/hadoop/hadoopdata/dfs/data/current/BF-1424625122-100.100.0.100-10307320600873/current/finalized/subdir5/subdir3/
[rest@ecs-329d-0002 subdir3]$ ls
blk_1073742604      blk_1073742611_1707.meta  blk_1073742632      blk_1073742640_1817.meta  blk_1073742666      blk_1073742688_1805.meta  blk_1073742703      blk_1073742704_1804.meta  blk_1073742713
[rest@ecs-329d-0002 subdir3]$ cat blk_1073742611
blk_1073742604_1700.meta  blk_1073742632_1800.meta  blk_1073742652      blk_1073742666_1843.meta  blk_1073742702      blk_1073742703_1803.meta  blk_1073742712      blk_1073742713_1803.meta
blk_1073742611          blk_1073742632_1707.meta  blk_1073742640      blk_1073742652_1842.meta  blk_1073742702_1802.meta  blk_1073742704
[rest@ecs-329d-0002 subdir3]$ cat blk_1073742713
[rest@ecs-329d-0002 subdir3]$

```

Node04


```
1 华为云01 2 华为云02 3 阿里云03 4 腾讯云04 +
Xshell 7 (Build 0090)
Copyright (c) 2020 Netsarang Computer, Inc. All rights reserved.

Type 'help' to learn how to use Xshell prompt.
[C:\~]$

Connecting to 175.24.202.153:22...
Connection established.
To escape to local shell, press 'Ctrl+Alt+J'.

[WARNING] The remote SSH server rejected X11 forwarding request.
Last failed login: Wed Dec 22 06:23:47 CST 2021 from 52.55.83.23 on ssh:netty
There were 133 failed login attempts since the last successful login.
Last login: Tue Dec 21 10:25:01 2021 from 111.187.14.24
[root@node04 ~]# cd /opt/hadoop/hadoopdata/dfs/data/current/BP-1424625122-192.168.0.166-1639730666975/current/finalized/subdir0/subdir3/
[root@node04 subdir3]# ls
blk_1073742653          blk_1073742665          blk_1073742667          blk_1073742702          blk_1073742712          blk_1073742713
blk_1073742653_1830.meta blk_1073742665_1842.meta blk_1073742667_1844.meta blk_1073742702_1882.meta blk_1073742712_1892.meta blk_1073742713_1893.meta
[root@node04 subdir3]# cat blk_1073742713
fjdkfdsj
[root@node04 subdir3]#
```

修改文件

```
[root@ecs-329d-0001 subdir3]# cat test.txt
sfgsfsgfsgkfkdskfkk
[root@ecs-329d-0001 subdir3]# hadoop fs -appendToFile test.txt /user/test.txt
[root@ecs-329d-0001 subdir3]# hadoop fs -cat /user/test.txt
fjdkfdsj
sfgsfsgfsgkfkdskfkk
[root@ecs-329d-0001 subdir3]#
```


Download

Head the file (first 32K)

Tail the file (last 32K)

Block information -- Block 0 ▾

Block ID: 1073742713

Block Pool ID: BP-1424625122-192.168.0.166-1639730666975

Generation Stamp: 1893

Size: 9

Availability:

- node04
- ecs-329d-0002
- ecs-329d-0001

File contents

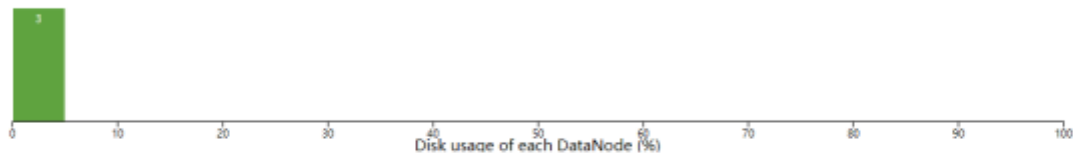
fjdkfdsj
sfgsfsgskfkdskk

Close

Datanode Information

 In service
 Down
 Decommissioning
 Decommissioned
 Decommissioned & dead
 Entering Maintenance
 In Maintenance
 In Maintenance & dead

Datanode usage histogram



In operation

DataNode State All

Show 25 entries

Search:

Node	Http Address	Last contact	Last Block Report	Used	Non DFS Used	Capacity	Blocks	Block pool used	Version
✓/default-rack/ecs-329d-0001:9866 (192.168.0.166:9866)	http://ecs-329d-0001:9866	0s	197m	716.12 MB	8.28 GB	98.18 GB <div><div></div></div>	276	716.12 MB (0.71%)	3.3.1
✓/default-rack/node03:9866 (106.15.248.113:9866)	http://node03:9866	0s	249m	681.24 MB	3.94 GB	58.93 GB <div><div></div></div>	264	681.24 MB (1.13%)	3.3.1
✓/default-rack/ecs-329d-0002:9866 (124.71.199.129:9866)	http://ecs-329d-0002:9866	0s	34m	717.69 MB	5.21 GB	98.18 GB <div><div></div></div>	270	717.69 MB (0.71%)	3.3.1

Showing 1 to 4 of 4 entries

Previous 1 Next

```

1 华为云CLI  2 华为云云OS  3 阿里云云OS  4 腾讯云云OS
Hadoop 7 (Build 0096)
Copyright (c) 2020 Hadoop Computer, Inc. All rights reserved.

Type 'help' to learn how to use Hadoop prompt.
[Ctrl+D]

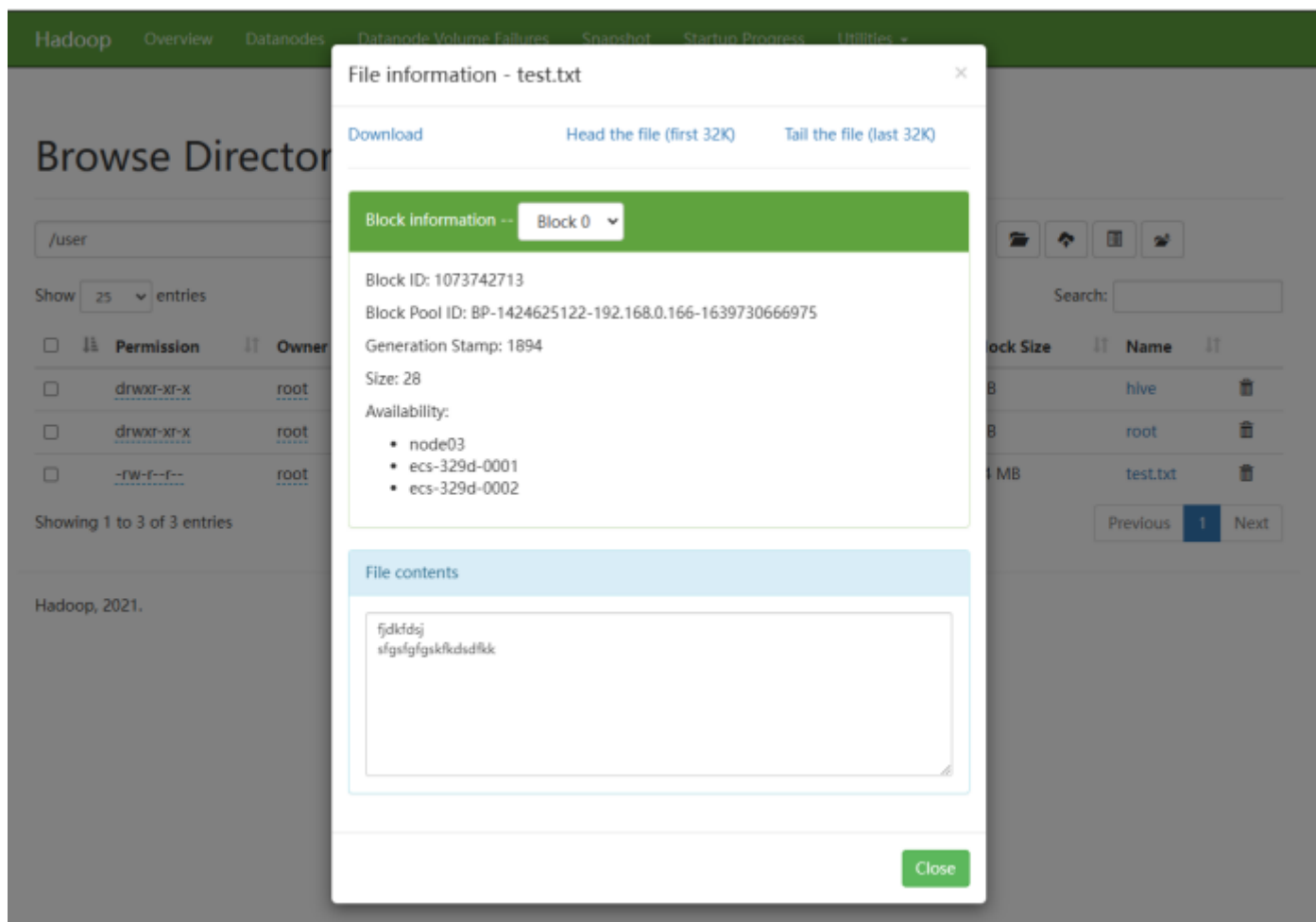
Connecting to 121.37.172.109:22...
Connection established.
To escape to local shell, press 'Ctrl+Alt+.'.

[Warning] The remote SSH server rejected X11 forwarding request.

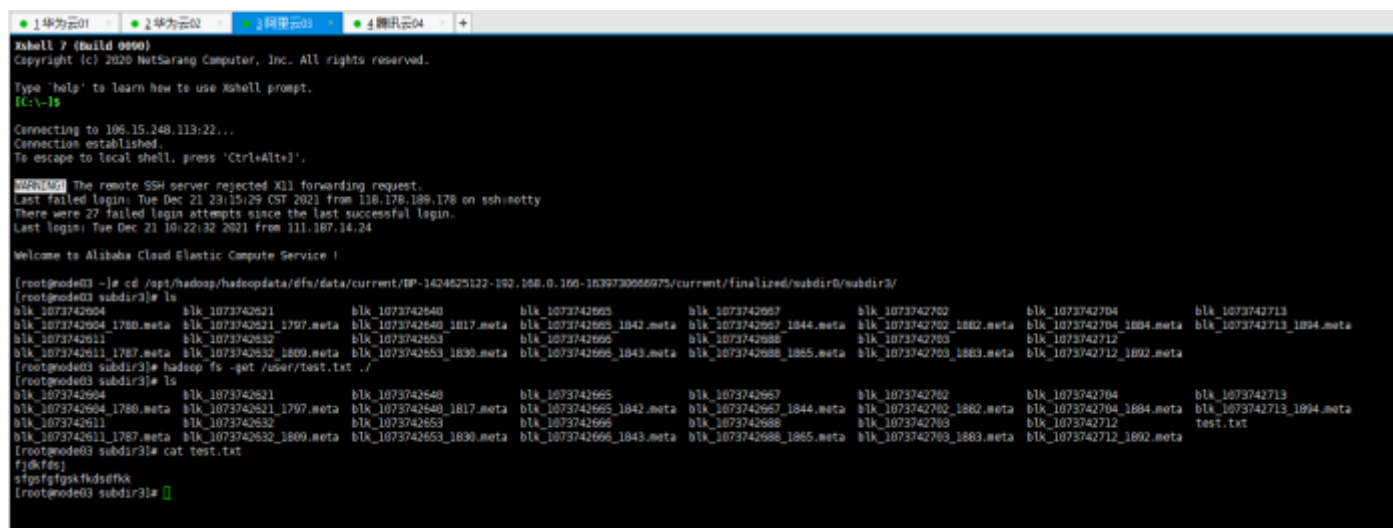
Welcome to Huawei Cloud Service

Last login: Wed Dec 22 11:45:19 2021
[root@ecs-3296-0001 ~]# ls
apache-hive-3.12.0-bin.tar.gz      hadoop-3.13.1.tar.gz      mysql-connector-java-8.0.27-tar.gz      scale-2.13.7.tgz      spark-2.3.0-bin-without-hadoop.tgz      spark-2.4.7.tgz      wget-log.1
apache-maven-3.8.4-bin.tar.gz     hbase                       rebase.out                          spark-2.3.0              spark-2.3.0-bin-hadoop3.2-scala2.13.tgz      spark-3.1.0-bin-hadoop3.2-scala2.13.tgz      wget-log.2
hadoop                             mysql80-community-release-el8-2.noarch.rpm  openjdk-8u11-b04-linux-s64-16_jan_2018.tar.gz  spark-2.3.0-bin-hadoop2.7.tgz      spark-2.4.7              vget-log
[root@ecs-3296-0001 ~]# vi test.txt
[root@ecs-3296-0001 ~]# cat test.txt
fjKfKfK
[root@ecs-3296-0001 ~]# hadoop fs -put test.txt /user/
[root@ecs-3296-0001 ~]# cd /opt/hadoop/hadoopdata/dfs/data/current/BP-1424035122-102_208.0.136-1636730966575/current/finalized/subdirb/subdir3/
[root@ecs-3296-0001 subdir3]# ls
blk_1073742604      blk_1073742621      blk_1073742640      blk_1073742655      blk_1073742667      blk_1073742702      blk_1073742764      blk_1073742713
blk_1073742604.1780.meta  blk_1073742621.1767.meta  blk_1073742640.1817.meta  blk_1073742655.1942.meta  blk_1073742667.1844.meta  blk_1073742702.1862.meta  blk_1073742764.1884.meta  blk_1073742713.1893.meta
blk_1073742611      blk_1073742627      blk_1073742643      blk_1073742658      blk_1073742669      blk_1073742707      blk_1073742730      blk_1073742712
blk_1073742611.1787.meta  blk_1073742627.1809.meta  blk_1073742643.1836.meta  blk_1073742658.1843.meta  blk_1073742669.1865.meta  blk_1073742707.1883.meta  blk_1073742730.1892.meta  test.txt
[root@ecs-3296-0001 subdir3]# cat blk_1073742713
fjKfKfK
[root@ecs-3296-0001 subdir3]# %C
[root@ecs-3296-0001 subdir3]# cat test.txt
vgsfgfsgkfhdfk
[root@ecs-3296-0001 subdir3]# hadoop fs -appendToFile test.txt /user/test.txt
[root@ecs-3296-0001 subdir3]# hadoop fs -cat /user/test.txt
fjKfKfK
vgsfgfsgkfhdfk
[root@ecs-3296-0001 subdir3]# hadoop fs -cat /user/test.txt
fjKfKfK
vgsfgfsgkfhdfk
[root@ecs-3296-0001 subdir3]#

```



可以看到系统自动保证副本数维持在设定值。



系统功能正常

系统优点

综合考虑各个节点的硬件性能，合理分配了各个节点所承担的任务，以此为依据分配组件，分散资源压力，避免某个或某几个节点压力过大。每个文件以64m为标准分block，据专家所说寻址时间 $=1/1000$ 读取时间时性能较佳，每个block有3份副本，可以容忍一台datanode宕机，任务调度使用Fair Scheduler，在多用户使用的场景下可以避免相互抢占资源，保证每个用户都能分配一定资源。

系统缺点

初期没有考虑到网络因素，云服务器选用了3个厂商，跨域通信比内网通信更缓慢，系统的性能没能完全发挥。只有一个namenode，如果namenode宕机则整个系统无法正常运作。

3.数据清洗

以下程序是把数据集的每一个属性放进hashset里面去重，并且求了主叫号码和被叫号码的交集，这对后来的建表提供了极大的帮助

Java

```
1 String iPath = "src/main/resources/data/data.txt";
2 BufferedReader reader = new BufferedReader(new FileReader(iPath));
3 String temp;
4 List<Set<String>> sets = new ArrayList<>();
5 for (int i = 0; i < 14; i++) {
6     sets.add(new HashSet<>());
7 }
8 Map<String, Integer> map = new HashMap<>();
9 while ((temp = reader.readLine()) != null) {
10     String[] split = temp.split("\\s+");
11     for (int k = 0; k < 14; k++) {
12         sets.get(k).add(split[k]);
13     }
14 }
```

```
dayId: 29
callingNbr: 450763
calledNbr: 1234497
callingOptr: 1
calledOptr: 3
callingCity: 1
calledCity: 327
callingRoamCity: 310
calledRoamCity: 330
startTime: 75618
endTime: 68938
rawDur: 4695
callType: 3
callingCell: 27546
主叫号码与被叫号码的交集数量为: 161688
```

通过对数据的去重和分析，发现存在大量的数据倾斜，比如主叫号码的全是电信的账号，被叫号码是主叫号码的两倍以上，数据集中的通话时间存在一定的不合理性，大约有百分之60的通话记录结束时间是00:00:00，所以我们需要根据开始时间（start_time）和通话时长（raw_dur）计算结束时间，但只有时分秒往往是不够的。根据问题要求，我们是需要计算出每个人在对应时间段的通话时长比例的，若出现如下情况

	A	B	C
1	startTime	endTime	rawDur
2	23:58:50	1:58:50	3600

不仅不能直接对时间相加减判断rawDur是否正确，而且对于根据时间段分片的操作也不能准确地进行。常见的数据库时间存储一般是timestamp或者date类型都要求是完整的包含年月日，所以需要 对数据集的属性进行扩充。这里选用java的Calendar对数据进行处理，先对开始时间进行处理，把诸如 00:00:00(HH:mm:ss)变成2012-02-02 00:00:00(yyyy-MM-dd HH:mm:ss)，然后再采用java的Calendar类对时间进行相加减，最后写入到新的文件里面保存为最终的数据。

Java

```
1 Calendar newTime = Calendar.getInstance();
2 //build.getStartTime()是已经加上年月日的开始时间
3 newTime.setTime(build.getStartTime());
4 int i = newTime.get(Calendar.MONTH) + 1;
5
6 String date = newTime.get(Calendar.DATE) > 9 ?String.valueOf(newTime.get(Calendar.DATE)): "0" + newTime.get(Calendar.DATE);
7
8 String begin = newTime.get(Calendar.YEAR) + "-" + "0" + i + "-" + date+ " " +
    newTime.get(Calendar.HOUR_OF_DAY) + ":" + newTime.get(Calendar.MINUTE) + ":"
    + newTime.get(Calendar.SECOND);
```

原文件(data.txt)

20120201	349723	y24373194378	1	2	0551	0551	0551	0551	08:39:50	08:42:54	184	1	373
20120201	338799	172001	1	1	0551	0551	0551	0551	00:01:40	00:04:43	183	1	318
20120201	1008566	1023148	1	1	0551	0551	0551	0551	00:10:41	00:10:57	16	1	749
20120201	349695	y21064962234	1	3	0551	0551	0551	0551	05:46:50	05:47:49	59	1	257
20120201	344115	4428205	1	1	0551	0551	0551	0551	08:08:38	08:09:06	28	1	2963
20120201	743622	y25212535979	1	2	0551	0551	0551	0551	08:27:45	08:28:17	32	1	22
20120201	744540	y26959267112	1	2	0551	0551	0551	0551	08:38:30	08:39:54	84	1	142
20120201	301056	y25432756021	1	2	0551	0551	0551	0551	06:36:58	06:37:15	17	1	687
20120201	306660	y22412491256	1	3	0551	0551	0551	0551	07:38:32	07:38:49	17	1	185
20120201	377980	y61259047620	1	2	0551	0551	0551	0551	08:03:31	08:05:03	92	1	39

清洗后的数据(newData.txt)

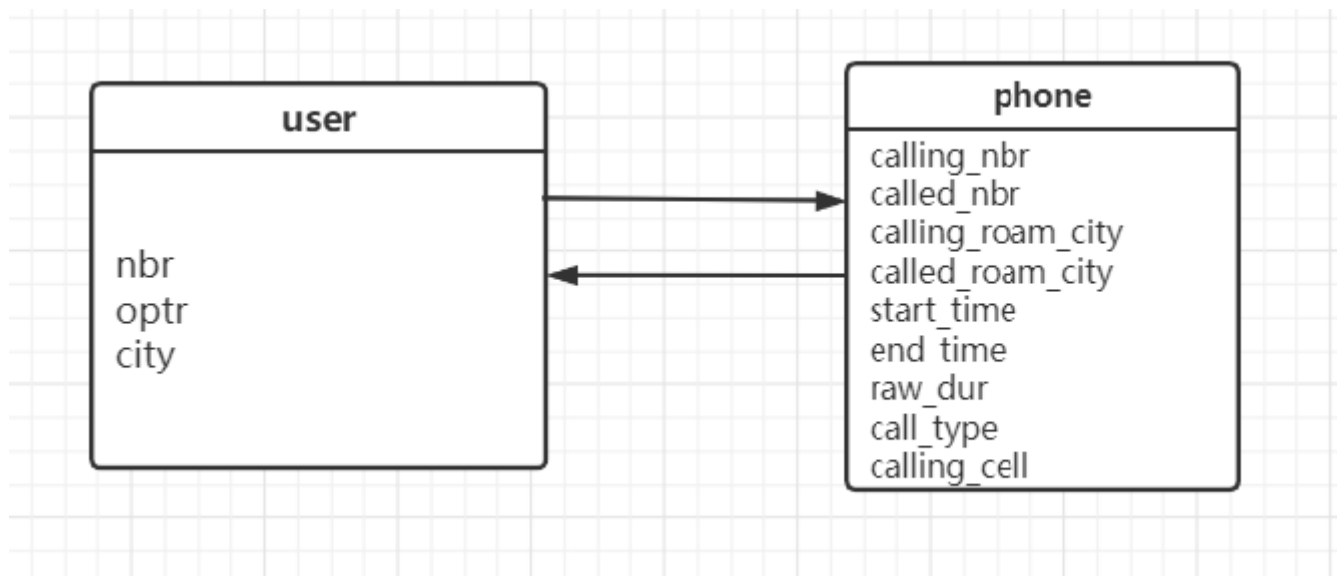
```
2012-02-01,349723,y24373194378,1,2,0551,0551,0551,0551,2012-02-01 8:39:50,2012-02-01 8:42:54,184,1,373
2012-02-01,338799,172001,1,1,0551,0551,0551,0551,2012-02-01 0:1:40,2012-02-01 0:4:43,183,1,318
2012-02-01,1008566,1023148,1,1,0551,0551,0551,0551,2012-02-01 0:10:41,2012-02-01 0:10:57,16,1,749
2012-02-01,349695,y21064962234,1,3,0551,0551,0551,0551,2012-02-01 5:46:50,2012-02-01 5:47:49,59,1,257
2012-02-01,344115,4428205,1,1,0551,0551,0551,0551,2012-02-01 8:8:38,2012-02-01 8:9:6,28,1,2963
2012-02-01,743622,y25212535979,1,2,0551,0551,0551,0551,2012-02-01 8:27:45,2012-02-01 8:28:17,32,1,22
2012-02-01,744540,y26959267112,1,2,0551,0551,0551,0551,2012-02-01 8:38:30,2012-02-01 8:39:54,84,1,142
2012-02-01,301056,y25432756021,1,2,0551,0551,0551,0551,2012-02-01 6:36:58,2012-02-01 6:37:15,17,1,687
2012-02-01,306660,y22412491256,1,3,0551,0551,0551,0551,2012-02-01 7:38:32,2012-02-01 7:38:49,17,1,185
2012-02-01,377980,y61259047620,1,2,0551,0551,0551,0551,2012-02-01 8:3:31,2012-02-01 8:5:3,92,1,39
```

4.HIVE数据库

4.1数据库设计

普通的关系型数据库，应设计如下，其中user是通话用户表，phone是通话记录表，是user与user多对多的联系集，但这样做有哪些不好呢，若是涉及连表操作，笛卡尔积会产生大量的中间结果，所以为了查询优化，允许对数据存储设计一定的冗余，通过分析业务需求，最后设计成一张大表，并对其query

Mysql



Hive

```
create table phone
(
    day_id            date      comment '日期',
    calling_nbr       varchar(255) comment '主叫号码',
    called_nbr        varchar(255) comment '被叫号码',
    calling_optr      varchar(1)   comment '主叫号码运营商',
    called_optr       varchar(1)   comment '被叫号码运营商',
    calling_city      varchar(255) comment '主叫号码归属地',
    called_city       varchar(255) comment '被叫号码归属地',
    calling_roam_city varchar(255) comment '主叫号码漫游城市',
    called_roam_city  varchar(255) comment '被叫号码漫游城市',
    start_time        timestamp   comment '通话开始时间',
    end_time          timestamp   comment '通话结束时间',
    raw_dur           int         comment '通话时长',
    call_type         varchar(1)   comment '通话类型',
    calling_cell      varchar(255) comment '主叫蜂窝号码'
)row format delimited
fields terminated by ",";
```

4.2数据库优化

存储优化

经查阅文献资料发现，Hive提供了多种文件存储的格式，常见的有TextFile，Parquet，ORC，Sequence，RC，AVRO等。其中Textfile文件存储格式未经压缩，会占用大量存储空间，并且查询速度较慢。ORC是一种列示存储格式，查询的时候不需要扫描全部的数据，而只需要读取每次查询涉及的列，所以存储格式的压缩比与查询速度表现都比较良好，所以我们选择以ORC文件格式存储数据，因此将上述以TextFile文件格式存储的数据库转换为ORC存储格式。

新建另一个另一张表phone的存储格式为ORC，通过insert overwrite table phone_orc select * from phone; 命令从Textflie文件格式的表中导入数据到ORC文件格式的表中。

以下是两种存储格式的数据库信息的对比。

TEXTFILE

Permission	↕↑	Owner	↕↑	Group	↕↑	Size
<u>-rW-r--r--</u>		<u>dr.who</u>		<u>supergroup</u>		195.14 MB

ORC

Permission	↕↑	Owner	↕↑	Group	↕↑	Size
<u>-rwxrwxrwx</u>		<u>root</u>		<u>supergroup</u>		34.27 MB

可以看到存储空间从195M变成了34M，降低了70%，优化效果十分可观。

查询举例：

查询主叫号码1015618的电话号码打电话的总次数

Java

```
1  select count(*) from phone where calling_nbr='1015618'
```

	A	B
1	TEXTFILE	ORC
2	3m 12s 726ms	2m 1s 498ms

MAPREDUCE优化

我们都知道hive底层默认使用的是MR引擎，无论是map还是reduce的数量都会极大地影响整个查询的速度，我们可以通过设置一些参数来调整

- 注意到hadoop默认分块大小是128M，对于190M这样的数据来说，若不修改分块大小，那么执行mapreduce的过程中，就会导致算力不平衡（128+62），进而导致运行的缓慢；但若一个任务中有很多小文件，如10M，20M等等，这样一个小文件也会被当作一个map任务来执行，而一个map任务启动和初始化的时间远远大于逻辑处理的时间，就会造成很大的资源浪费。所以我们应该合理设置分块的大小来使得资源合理分配。
- Reduce的个数对整个作业的运行性能有很大影响。如果Reduce设置的过大，那么将会产生很多小文件，对NameNode会产生一定的影响，而且整个作业的运行时间未必会减少；如果Reduce设置的过小，那么单个Reduce处理的数据将会加大，很可能会引起OOM异常。实验过程中发现部分任务中不管数据量多大，不管有没有调整reduce个数的参数，任务中一直都只有一个reduce任务。经过探究发现，分为以下三种情况
 - 没有group by的汇总，

SQL

```
1 select call_type,count(1) from phone where call_type = 1 group by pt
2
3 select count(1) from phone where call_type=1
```

以上这两种sql语句执行起来是不一样的

- 使用了Order by进行排序
- sql中包含笛卡尔积

在设置reduce个数的时候需要考虑这两个原则：使大数据量利用合适的reduce数；是单个reduce任务处理合适的数据量；

下面是我为了优化尝试修改的部分参数

	A	B
1	描述	参数设置
2	设置分块大小	set dfs.block.size=64M
3	设置每个Map最大输入大小	set mapred.max.split.size=67108864
4	开启并发执行	set hive.exec.parallel=true
5	设置并发线程数	set hive.exec.parallel.thread.number=16
6	设置map的任务数	set mapred.map.tasks=4
7	设置reduce的任务数	set mapred.reduce.tasks=4
8	设置每个reduce任务处理的数据量	set hive.exec.reducers.bytes.per.reducer=128M
9	设置每个任务最大的reduce数	set hive.exec.reducers.max=999

分区表优化

Hive在使用条件查询的时候会扫描整张表，消耗很多时间做没必要的工作。若大量的查询是基于某个字段的，那么我们可以对数据进行分区处理。分区表是hive提供的一个特性，利用partition关键字可以把现成的数据表分成多个分区（文件夹）存储分区将表的数据在物理上分成不同的文件夹，以便于在查询时可以精准指定所要读取的分区目录，从而降低读取的数据量。

对于通话类型这个字段，由于只有三种类型（市话，长途，漫游），所以我重建一张分区表partition_phone，通过命令insert overwrite table default.phone partition(call_type) select * from phone，在物理上就可以把hadoop中的文件分为三个子文件，

查询举例:

查询每个主叫用户，查询通话类型为市话的次数

SQL

```
1 select calling_nbr,count(1) times from phone where call_type=1 group by calling_nbr
```

	A	B
1	original	Partition
2	3m 41s 478ms	2 min 44s 704ms

索引优化

ORC文件格式内置多种轻量级索引，已经能提供很好的查询性能优化，为验证建立密集索引是否会提升查询性能，

执行语句：查询主叫号码运营商为移动或连通的的个数

SQL

```
1 Select count(*) from phone where calling_optr=2 and calling_optr=3
```

	A	B
1	无索引	有索引
2	2 min 18s 230ms	2min 20s 250ms

发现建立索引反而会降低查询速度，推测可能与ORC文件的压缩方式与存储方式有关

4.3云计算策略

problem 1

计算出用户的每日平均通话次数，并将结果以<主叫号码, 每日平均通话次数>的格式保存成txt或excel文件。

计算用户每日的平均通话次数，再计算平均时我分为两种情况解读，一种是除以总天数29天，所以最后的结果分为三列，

<号码，平均通话次数（只包含通话的天数），平均通话次数（30天）>

主叫号码

SQL

```
1 select calling_nbr ,
2 count(1)/count(distinct(day_id)) times,
3 count(1)/29 times2
4 from phone
5 group by calling_nbr order by times desc
```

被叫号码

SQL

```
1 select called_nbr ,
2 count(1)/count(distinct(day_id)) times,
3 count(1)/29 times2
4 from phone
5 group by called_nbr order by times desc
```

Problem 2

计算出不同通话类型（市话、长途、国际）下各个运营商（移动，联通，电信）的占比，并画出饼状图。

主叫号码

SQL

```
1 select calling_optr,call_type,count(1) times from phone
2 group by calling_optr,call_type
```

被叫号码

SQL

```
1 select called_optr,call_type,count(1) times from phone
2 group by called_optr,call_type
```

Problem 3

计算出用户在各个时间段（时间段的划分如表1所示）通话时长所占比例，并将结果以<主叫号码, 时间段1占比, ..., 时间段8占比>的格式保存成txt或excel文件。

计算预处理，先按照通话时间排序得到每个人的通话总时长

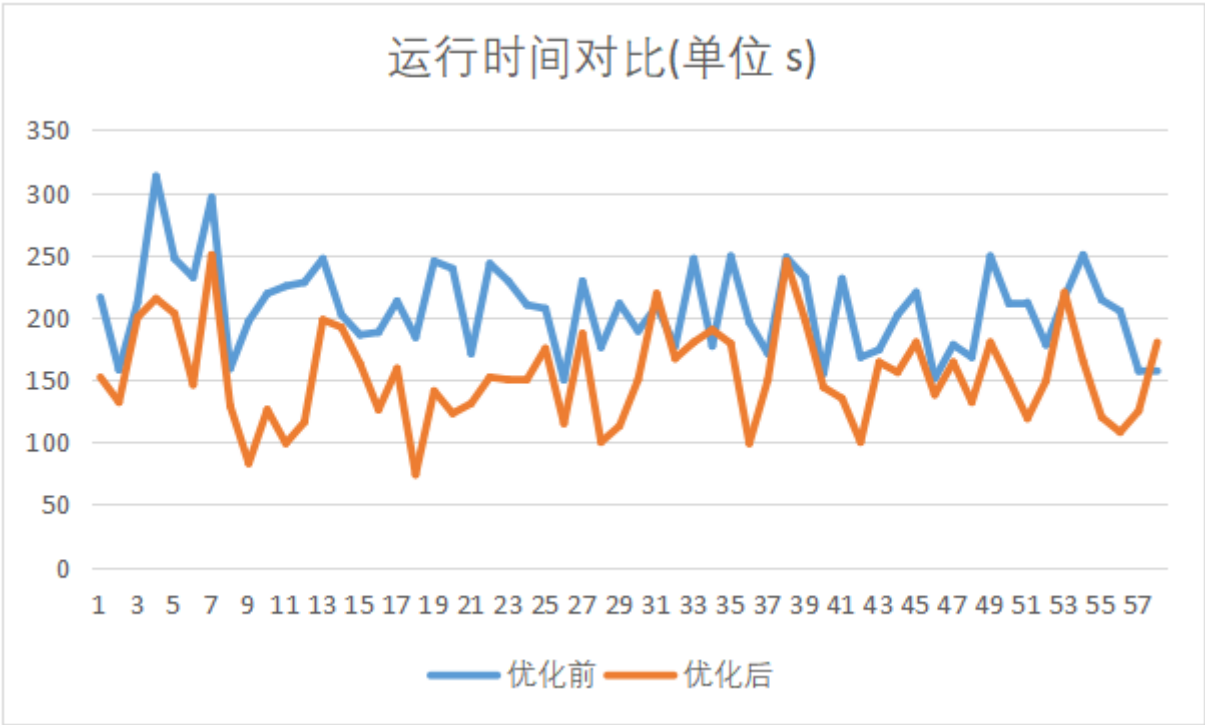
SQL

```
1 select calling_nbr,sum(raw_dur) as rawDur
2 from phone group by calling_nbr order by rawDur desc
```

然后通过自己写的java分块规则把每一段时间都分成八分，比如3:50~7:20会被分成[0,7800,4800,0,0,0,0,0]这样的一个序列，最后结合预处理计算出来的总时长，以及mybatis的函数sql调用，成功得到最后的结果

4.4优化数据展示

由于测试数据比较少，具有较大的偶然性，认真听取了答辩时老师所说的不足之处，我测试了大约20条sql语句，并编写脚本让它们随机执行60次，最终得到优化前后性能对比的执行时间折线图



通过图像可以看到，大多数的运行速度都比原来的快，只有少部分在优化后没有得到明显的提升，猜测可能是部分优化效果冲突，比如Mapreduce和数据分区优化方向不一致会导致整体流程变慢。

4.5Hive特点分析

Hive是Hadoop上的数据仓库工具，处理的是结构化数据。其定位是数据仓库，适用于实时性要求不高的场合。因为Hive会将大部分的SQL查询转换成MapReduce任务，而MapReduce具有高延迟的特性，其JVM的启动过程以及Map和Reduce的过程都会耗费一些时间，所以其具有一个查询时间的下限，无法很快的应对实时性查询。

Hive主要用来进行数据分析，对于分析统计海量数据的执行时间而言，Hive的执行延迟已经不能成为它的缺点，MapReduce的算法使得分布式节点的性能数量成为执行时间的主要限制因素。

Hive由于是搭建在分布式文件系统的基础上，所以具有很好的扩展性和容错性。