
Table of Contents

简介	1.1
1. HDFS	1.2
1.1 HDFS Report分析	1.2.1
2. MapReduce	1.3
3. Yarn	1.4
4. Hadoop	1.5
4.1 GP-Hadoop-Elk的原理区别	1.5.1
5. Spark	1.6
5.1 跨集群访问HBase	1.6.1
5.2 test	1.6.2
结束	1.7

Introduction

- 简介
- 1. Hadoop
 - 1.1 HDFS
 - 1.1.1 HDFS Report分析
 - 1.2 MapReduce
 - 1.3 Yarn
 - 1.4 其他
 - 1.4.1 HDFS Report分析
- 2. Spark
 - 2.1 SparkStreaming
 - 2.1.1 跨集群访问HBase
 - 2.2 SparkSql
- 结束

1. HDFS

1.1 HDFS Report分析方法 -- hdfs dfsadmin

hdfs dfsadmin -report

```
[root@CNSZ431014 ~]# hdfs dfsadmin -report
Configured Capacity: 74426056298496 (67.69 TB)
Present Capacity: 70124136352340 (63.78 TB)
DFS Remaining: 33715992505225 (30.66 TB)
DFS Used: 36408143847115 (33.11 TB)
DFS Used%: 51.92%
Under replicated blocks: 29904
Blocks with corrupt replicas: 0
Missing blocks: 0
Missing blocks (with replication factor 1): 0

-----
Live datanodes (6):

Name: 10.14.18.207:25009 (CNSZ431018)
Hostname: CNSZ431018
Rack: /default/rack0
Decommission Status : Normal
Configured Capacity: 17720489594880 (16.12 TB)
DFS Used: 7213844099821 (6.56 TB)
Non DFS Used: 1017260168087 (947.40 GB)
DFS Remaining: 9489385326972 (8.63 TB)
DFS Used%: 40.71%
DFS Remaining%: 53.55%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 18
Last contact: Mon Dec 11 10:59:04 CST 2017

Name: 10.14.18.208:25009 (CNSZ431019)
Hostname: CNSZ431019
Rack: /default/rack0
Decommission Status : Normal
Configured Capacity: 17720489594880 (16.12 TB)
DFS Used: 7133852738211 (6.49 TB)
Non DFS Used: 1016951088951 (947.11 GB)
DFS Remaining: 9569685767718 (8.70 TB)
DFS Used%: 40.26%
DFS Remaining%: 54.00%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
```

```
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 16
Last contact: Mon Dec 11 10:59:04 CST 2017
```

```
Name: 10.14.18.206:25009 (CNSZ431017)
Hostname: CNSZ431017
Rack: /default/rack0
Decommission Status : Normal
Configured Capacity: 17720489594880 (16.12 TB)
DFS Used: 6728401972418 (6.12 TB)
Non DFS Used: 1018018554490 (948.10 GB)
DFS Remaining: 9974069067972 (9.07 TB)
DFS Used%: 37.97%
DFS Remaining%: 56.29%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 19
Last contact: Mon Dec 11 10:59:04 CST 2017
```

```
Name: 10.14.18.203:25009 (CNSZ431014)
Hostname: CNSZ431014
Rack: /default/rack0
Decommission Status : Normal
Configured Capacity: 7088195837952 (6.45 TB)
DFS Used: 5117967231685 (4.65 TB)
Non DFS Used: 416702126051 (388.08 GB)
DFS Remaining: 1553526480216 (1.41 TB)
DFS Used%: 72.20%
DFS Remaining%: 21.92%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 23
Last contact: Mon Dec 11 10:59:04 CST 2017
```

```
Name: 10.14.18.204:25009 (CNSZ431015)
Hostname: CNSZ431015
Rack: /default/rack0
Decommission Status : Normal
Configured Capacity: 7088195837952 (6.45 TB)
DFS Used: 4596038181902 (4.18 TB)
Non DFS Used: 417036831431 (388.40 GB)
DFS Remaining: 2075120824619 (1.89 TB)
DFS Used%: 64.84%
```

```

DFS Remaining%: 29.28%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 22
Last contact: Mon Dec 11 10:59:03 CST 2017

```

```

Name: 10.14.18.205:25009 (CNSZ431016)
Hostname: CNSZ431016
Rack: /default/rack0
Decommission Status : Normal
Configured Capacity: 7088195837952 (6.45 TB)
DFS Used: 5618039623078 (5.11 TB)
Non DFS Used: 415951177146 (387.38 GB)
DFS Remaining: 1054205037728 (981.80 GB)
DFS Used%: 79.26%
DFS Remaining%: 14.87%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 28
Last contact: Mon Dec 11 10:59:04 CST 2017

```

hdfs dfsadmin -safemode 安全模式

NameNode在启动的时候首先进入安全模式，如果datanode丢失的block达到一定的比例（由hdfs-site.xml文件中dfs.safemode.threshold.pct决定，默认0.999f），则系统会一直处于安全模式状态即只读状态；否则没有其他情况影响，一般情况下，系统会自动离开安全模式。

dfs.safemode.threshold.pct 表示HDFS启动的时候，如果DataNode上报的 block个数0.999倍才可以离开安全模式，否则一直是这种只读状态。如果设为1则hdfs永远是处于SafeMode。

通常两种情况可以离开这处安全模式：

1、修改 dfs.safemode.threshold.pct为一个比较小的值，缺省值是0.999 2、hadoop dfsadmin -safemode leave 命令强制离开

用户可以使用命令行（hdfs dfsadmin -safemode value）做如下的操作：

```

# hdfs dfsadmin -safemode get    ## 返回安全模式是否开启的信息，返回 Safe mode is OFF/OPEN
# hdfs dfsadmin -safemode enter  ## 进入安全模式
# hdfs dfsadmin -safemode leave  ## 强制 NameNode 离开安全模式
# hdfs dfsadmin -safemode wait   ## 等待，一直到安全模式结束

```


2. MapReduce

3. Yarn

4. Hadoop

GP、Hadoop、Elk的原理区别

存储模型

hadoop是hdfs，扩展是通过元数据来做的，中心节点用来存元数据，在加入新的节点的时候，只需要修改元数据就可以了，所以hdfs的扩展能力是受到管理元数据那台机器的性能限制的。

mpp通常采用的是没有中心节点的存储模型，比如hash，你每次增加节点的时候，都需要rehash，这样当规模到了几百台的时候，扩展能力就下来了

内存管理方式

mpp内存管理比较精细，他主要的想法是在每个机器上放个数据库，传统数据库的内存管理比较复杂，主要是内外存交互的东西，这样的架构决定了mpp在小数据量的时候，延迟可以做的比较小，但是在大数据量的时候，吞吐量做不上去。

hive的内存管理非常粗放，他后来就是mapreduce的job，mr的job是没有太多精细的内存管理的，他就是拼命地scan，完了顶多就是个spill，这样的架构导致throughput很大，但是latency很高，当你集群规模很大的时候，你一般会追求很大的throughput

当数据量很大的时候，如果你用mpp那种传统的内存管理的话，大批量的计算反而会慢，而且更加占资源。

事务

hive不支持传统意义上的那种高并发的事务

一旦你要上分布式事务，基本上你的可扩展性就上不去了

failover机制，

hive的failover就是mr的failover，job挂掉了重新换机器跑就完了，但是mpp如果采用传统架构的话，他的计算是要attach到数据节点上去的，如果你规模上去，那么fail的可能性就上去了，这样如果你每次计算都有台机器挂了，你一挂，别人就要等你，而不是换台机器继续跑，那么这个也限制了可扩展性，当然，如果mpp在底层用了统一的存储，完了计算也可以到处转移，再想个办法把中间状态记录下来，也可以扩展（这个实际上就是sparksql）

扩展性

MPP DB 还是基于原 DB 扩展而来，DB 里面天然追求一致性（Consistency），必然带来分区容错性较差。集群规模变得太大，业务数据太多时，MPP DB 的元数据管理就完全是一个灾难。元数据巨大无比，一旦出错很难恢复，动不动导致毁库。

所以 MPP DB 要在扩展性上有质的提示，要对元数据，以及数据存储有架构上的突破，降低对一致性的要求，这样扩展性才能提升，否则的话很难相信一个 MPP DB 数据库是可以容易扩展的

并发

一个查询系统，设计出来就是提供人用的，所以能支持的同时并发越高越好。MPP DB 核心原理是一个大的查询通过分析为一个个子查询，分布到底层的执行，最后再合并结果，说白了就是通过多线程并发来暴力 SCAN 来实现高速。这种暴力SCAN的方法，对单个查询来说，动用了整个系统的能力，单个查询比较快，但同时带来用力过猛的问题，整个系统能支持的并发必然不高，从目前实际使用的经验来说，也就支持50~100的并发能力。

hadoop 和 mpp 的本质区别是：就是什么时候解决 data locality 的问题 hadoop 的思路是每次计算的时候解决，mpp的思路是加载的时候解决。

从查询引擎看，由于数据库支持索引，查询性能应该优于HADOOP。但是对于PB级别的数据，无法给所有维度的查询建立索引，主要靠全表扫描。因此对于复杂查询，MPP并不比HADOOP特别是现在的SPARK方案体现出优势，而且架不住Hadoop集群机器多。

5. Spark

5.1 安全集群模式下，Spark跨集群连接HBase

背景：本端集群的Spark组件需要操作对端集群的HBase组件。Spark程序运行模式为yarn-cluster模式。

前置条件：两个集群已经互信。集群互信的基本条件是，本端集群和对端集群的版本必须完全一致，并且都为安全模式。

跨集群连接HBase的步骤如下：

1. 修改spark客户端的conf目录下的**spark-defaults.conf**，确保

```
spark.hbase.obtainToken.enabled = true
spark.inputFormat.cache.enabled = false
```

这是为了开启Spark on HBase特性，安全模式下必须开启此特性，Spark中才能认证HBase，否则会报出GSSEException：

```
javax.security.sasl.SaslException: GSS initiate failed [Caused by GSSEException: No
valid credentials provided (Mechanism level: Failed to find any Kerberos tgt)]
```

通过实际投产情况，发现跨集群连接HBase的情况下，必须是通过修改**spark-defaults.conf**文件来开启Spark on HBase才有效果；如果不修改这个文件，用别的手段去修改参数（比如程序代码中直接指定配置项的值），虽然Spark的Environment信息中显示为开启，但是实际还是无法连接，仍然报出GSSEException。

2. 继续修改**spark-defaults.conf**文件，将**spark.yarn.cluster.driver.extraClassPath**参数值，加入\$PWD作为**classpath**的一部分，并保证\$PWD在首位。例如，修改前为：

```
spark.yarn.cluster.driver.extraClassPath = /opt/huawei/Bigdata/FusionInsight/spark/
cfg/:/opt/huawei/Bigdata/FusionInsight/spark/spark/lib/*
```

修改后为：

```
spark.yarn.cluster.driver.extraClassPath = $PWD:/opt/huawei/Bigdata/FusionInsight/s
park/cfg/:/opt/huawei/Bigdata/FusionInsight/spark/spark/lib/*
```

如果不加入这一步，跨集群情况下，当前FusionInsight版本会存在无法刷新HBase Token导致token过期的问题。因为yarn内部启动container的shell脚本中，定义的classpath的顺序会导致刷新token时，优先读取节点上的配置文件，而不是Spark客户端提交的，导致刷新HBase Token时连接的zookeeper服务不是HBase所在集群的，而变成本集群的了，最终导致无法刷新HBase Token而连接过期。

3. 将spark客户端的conf目录下的**hbase-site.xml**，替换为对端集群的**hbase-site.xml**文件。

4. 在**spark-submit**提交时，通过--files参数将对端集群**hbase-site.xml**文件发送到每个**Executor**

```
$SPARK_HOME/bin/spark-submit \
--master yarn \
--deploy-mode cluster \
--driver-memory 2G \
--num-executors 5 \
--principal logc \
--keytab /logcAPP/SparkApp/SparkStreamingApp/conf/logc.keytab \
--jars $SPARK_HOME/lib/streamingClient/kafka-clients-0.8.2.1.jar,$SPARK_HOME/lib/streamingClient/kafka_2.10-0.8.2.1.jar,$SPARK_HOME/lib/streamingClient/spark-streaming-kafka_2.10-1.5.1.jar,/logcAPP/SparkApp/SparkStreamingApp/lib/commons-dbcop2-2.1.1.jar,/logcAPP/SparkApp/SparkStreamingApp/lib/commons-pool2-2.4.2.jar,/logcAPP/SparkApp/SparkStreamingApp/lib/ojdbc7.jar,/logcAPP/SparkApp/SparkStreamingApp/lib/fastjson-1.2.31.jar \
--files /logcAPP/SparkApp/SparkStreamingApp/conf/dbconfig.properties,/logcAPP/SparkApp/SparkStreamingApp/conf/BehaviorTraceInfo.properties,/logcAPP/SparkApp/SparkStreamingApp/conf/hbase-site.xml \
--name BehaviorTraceInfo_New \
--class com.berchina.iec.spark.streaming.BehaviorTraceInfo_New \
/logcAPP/SparkApp/SparkStreamingApp/BehaviorTraceInfo.jar cluster 300
```

5. 程序代码中，在创建**HBase**连接时，将--files参数发送过来的对端**hbase-site.xml**文件作为连接配置，再创建连接：

```

public class HBaseTableFactory {
    private static Configuration conf = null;
    private static Connection conn = null;

    private Table hTable;

    public Table getTable() {
        return hTable;
    }

    static {
        init();
    }

    private static void init() {
        conf = HBaseConfiguration.create();

        conf.addResource(new Path(System.getProperty("user.dir") + File.separator + "h
base-site.xml"));
        System.err.println("配置读取测试：" + conf.get("hbase.zookeeper.quorum"));

        try {
            conn = ConnectionFactory.createConnection(conf);
        } catch (IOException e) {
            System.err.println("ConnectionFactory.createConnection错误：" + e.getMessage());
        }
    }

    public HBaseTableFactory(String tableName) {
        try {
            hTable = conn.getTable(TableName.valueOf(tableName));
        } catch (IOException e) {
            System.err.println("conn.getTable(" + tableName + ")错误：" + e.getMessage());
        }
    }
}

```

注意，yarn-cluster模式下，无需在代码中编写安全认证相关代码。但是如果是长时间运行的Spark Streaming应用，会存在认证过期问题，需要指定principal和keytab参数，使得yarn能去定期认证。关于这一点，在后面的小节中会详细介绍。

5.2 test

结束

结束