hadoop=framework

->storage+caculation (distributed)

# distributed storage(PB level)

just data,not os,stored in multiple computers

e.g. BaiduCloud 360Cloud

in Hadoop,HDFS(Hadoop Distributed File System)

# distributed caculation

a task -> subtasks one task by many computers

target:reduce compute time

tranditional compute:drive -> memory IO cost

~:parallel IO

in Hadoop:MR(Map Reduce)

# version

Apache (official)

Cloudera(commercial,introduced some patch)

HortonWork(integrated apache)

# module

common:serial,rpg communication,

HDFS:storage

yarn:resource management platform

mapreduce:caculation

# derived framework

Ambari : Management Monitor

Avro : Serial

Cassandra : no single fault

Chukwa : Data collection (from all nodes to different destination-db,console)

HBase : Big Table,Structed data warehouse

Hive : Data aggregate & query instantly

Mahout : Machine learning & data mining library

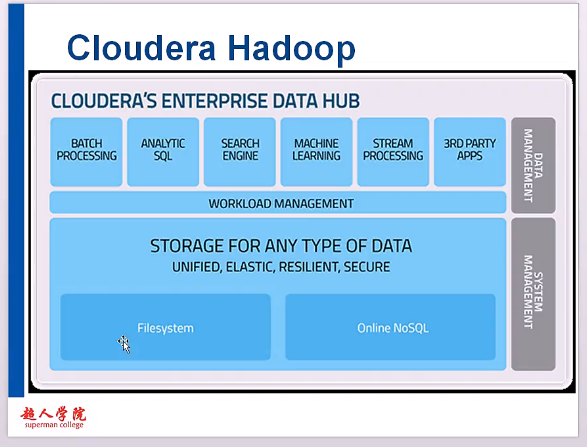
Pig : superior data stream language

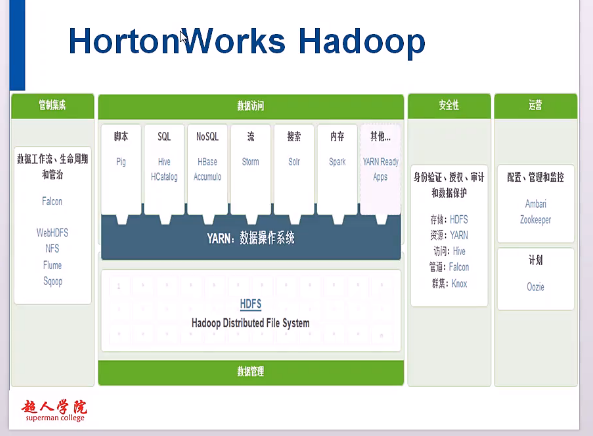
Spark : (In Memory) Common caculation engine

Tez : Common caculation stream framework

Zookeeper : a framework for coporation

# 3rd party-Hadoop







# Hadoop Components



compare:

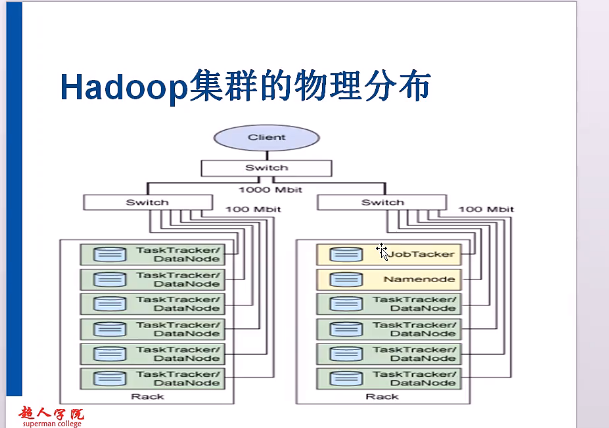
Map-Reduce:Batch processing

Storm:Streaming data process

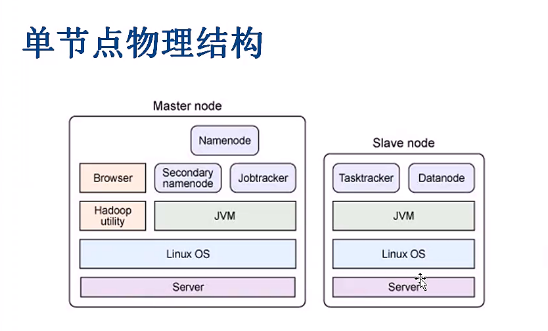
Spark:Caculate in memory

Distrubuted App:(Storm,Spark,HDFS...)

1. Master-Slave Structure
   1. HDFS(NameNode(M),Datanode(S))
   2. YARN(ResourceManager(M),Nodemanager(S))
   3. MR(JobTracker(M),TaskTracker(S))
   4. Storm(Nimbus(M),Supervisor(S))

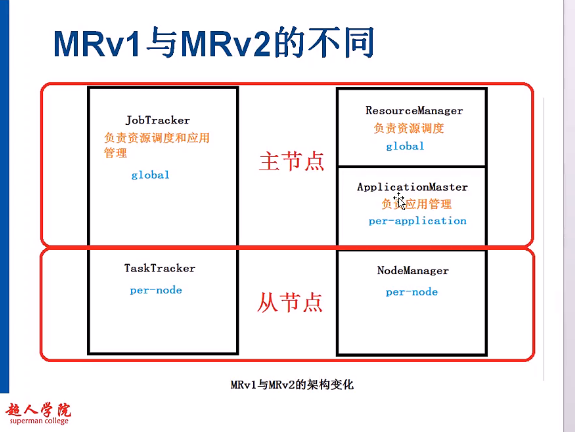


代码一样 进程不一样



Hadoop1 jdk1.6

Hadoop2 jdk1.7



左边列是v1 右边v2

ApplicationMaster 是YARN对外接口

## Hadoop单机配置(非分布式)

Hadoop 默认模式为非分布式模式，无需进行其他配置即可运行。非分布式即单 Java 进程，方便进行调试。

cd /usr/local/hadoop

mkdir ./input

cp ./etc/hadoop/\*.xml ./input # 将配置文件作为输入文件

./bin/hadoop jar ./share/hadoop/mapreduce/hadoop-mapreduce-examples-\*.jar grep ./input ./output 'dfs[a-z.]+'

cat ./output/\* # 查看运行结果

## Hadoop伪分布式配置

**core-site.xml**

<configuration>

<property>

<name>hadoop.tmp.dir</name>

<value>file:/usr/local/hadoop/tmp</value>

<description>Abase for other temporary directories.</description>

</property>

<property>

<name>fs.defaultFS</name>

<value>hdfs://localhost:9000</value>

</property>

</configuration>

**hdfs-site.xml**：

<configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>

<name>dfs.namenode.name.dir</name>

<value>file:/usr/local/hadoop/tmp/dfs/name</value>

</property>

<property>

<name>dfs.datanode.data.dir</name>

<value>file:/usr/local/hadoop/tmp/dfs/data</value>

</property>

</configuration>

**Hadoop配置文件说明**

Hadoop 的运行方式是由配置文件决定的（运行 Hadoop 时会读取配置文件），因此如果需要从伪分布式模式切换回非分布式模式，需要删除 core-site.xml 中的配置项。

此外，伪分布式虽然只需要配置 fs.defaultFS 和 dfs.replication 就可以运行（官方教程如此），不过若没有配置 hadoop.tmp.dir 参数，则默认使用的临时目录为 /tmp/hadoo-hadoop，而这个目录在重启时有可能被系统清理掉，导致必须重新执行 format 才行。所以我们进行了设置，同时也指定 dfs.namenode.name.dir 和 dfs.datanode.data.dir，否则在接下来的步骤中可能会出错。

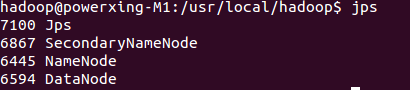
配置完成后，执行 NameNode 的格式化:

1. ./bin/hdfs namenode -format

接着开启 NameNode 和 DataNode 守护进程。

1. ./sbin/start-dfs.sh

启动完成后，可以通过命令 jps 来判断是否成功启动



成功启动后，可以访问 Web 界面 [http://localhost:50070](http://localhost:50070/) 查看 NameNode 和 Datanode 信息，还可以在线查看 HDFS 中的文件。

vm

<http://192.168.208.130:50070>

## 运行Hadoop伪分布式实例

在 HDFS 中创建用户目录：

1. ./bin/hdfs dfs -mkdir -p /user/hadoop

将 ./etc/hadoop 中的 xml 文件作为输入文件复制到分布式文件系统中

此在命令中就可以使用相对路径如 input，其对应的绝对路径就是 /user/hadoop/input:

1. ./bin/hdfs dfs -mkdir input
2. ./bin/hdfs dfs -put ./etc/hadoop/\*.xml input

Shell 命令

复制完成后，可以通过如下命令查看文件列表：

1. ./bin/hdfs dfs -ls input

Shell 命令

伪分布式运行 MapReduce 作业的方式跟单机模式相同，区别在于伪分布式读取的是HDFS中的文件（可以将单机步骤中创建的本地 input 文件夹，输出结果 output 文件夹都删掉来验证这一点）。

1. ./bin/hadoop jar ./share/hadoop/mapreduce/hadoop-mapreduce-examples-\*.jar grep input output 'dfs[a-z.]+'

Shell 命令

查看运行结果的命令（查看的是位于 HDFS 中的输出结果）：

1. ./bin/hdfs dfs -cat output/\*

Shell 命令

我们也可以将运行结果取回到本地：

1. rm -r ./output *# 先删除本地的 output 文件夹（如果存在）*
2. ./bin/hdfs dfs -get output ./output *# 将 HDFS 上的 output 文件夹拷贝到本机*
3. cat ./output/\*

Shell 命令

执行如下命令删除 output 文件夹:

1. ./bin/hdfs dfs -rm -r output *# 删除 output 文件夹*

Shell 命令

若要关闭 Hadoop，则运行

1. ./sbin/stop-dfs.sh

## 启动YARN

（伪分布式不启动 YARN 也可以，一般不会影响程序执行）

上述通过 ./sbin/start-dfs.sh 启动 Hadoop，仅仅是启动了 MapReduce 环境，我们可以启动 YARN ，让 YARN 来负责资源管理与任务调度。

首先修改配置文件 **mapred-site.xml**，这边需要先进行重命名：

1. mv ./etc/hadoop/mapred-site.xml.template ./etc/hadoop/mapred-site.xml

Shell 命令

然后再进行编辑，同样使用 gedit 编辑会比较方便些 gedit ./etc/hadoop/mapred-site.xml ： 不用时需要改回来

<configuration>

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

</configuration>

XML

接着修改配置文件 **yarn-site.xml**：

<configuration>

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

</configuration>

XML

然后就可以启动 YARN 了（需要先执行过 ./sbin/start-dfs.sh）：

1. ./sbin/start-yarn.sh *# 启动YARN*
2. ./sbin/mr-jobhistory-daemon.sh start historyserver *# 开启历史服务器，才能在Web中查看任务运行情况*

Shell 命令

开启后通过 jps 查看，可以看到多了 NodeManager 和 ResourceManager 两个后台进程，如下图所示。

开启后通过 jps 查看，可以看到多了 NodeManager 和 ResourceManager 两个后台进程，如下图所示。

启动 YARN 之后，运行实例的方法还是一样的，仅仅是资源管理方式、任务调度不同。观察日志信息可以发现，不启用 YARN 时，是 “mapred.LocalJobRunner” 在跑任务，启用 YARN 之后，是 “mapred.YARNRunner” 在跑任务。启动 YARN 有个好处是可以通过 Web 界面查看任务的运行情况：<http://localhost:8088/cluster>，如下图所示。

vm

<http://192.168.208.130:8088/cluster>