Hadoop集群搭建

Hadoop Cluster

Hadoop

Apache Hadoop

The Apache™ Hadoop® project develops open-source software for reliable, scalable, distributed computing.

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

Learn more » Download » Getting started »

Hadoop

- Hadoop是一个用java编写的分布式计算和存储框架
- 使我们可以从单一服务器扩展到数千台机器同时参与存储与计算

Hadoop

- Hadoop主要模块
- Hadoop Common：是其他Hadoop模块所需的Java库和实用程序。
- HDFS（Hadoop Distributed File System）：分布式文件系统
- YARN：资源管理和作业调度框架
- MapReduce：基于Yarn的分布式计算框架

HDFS

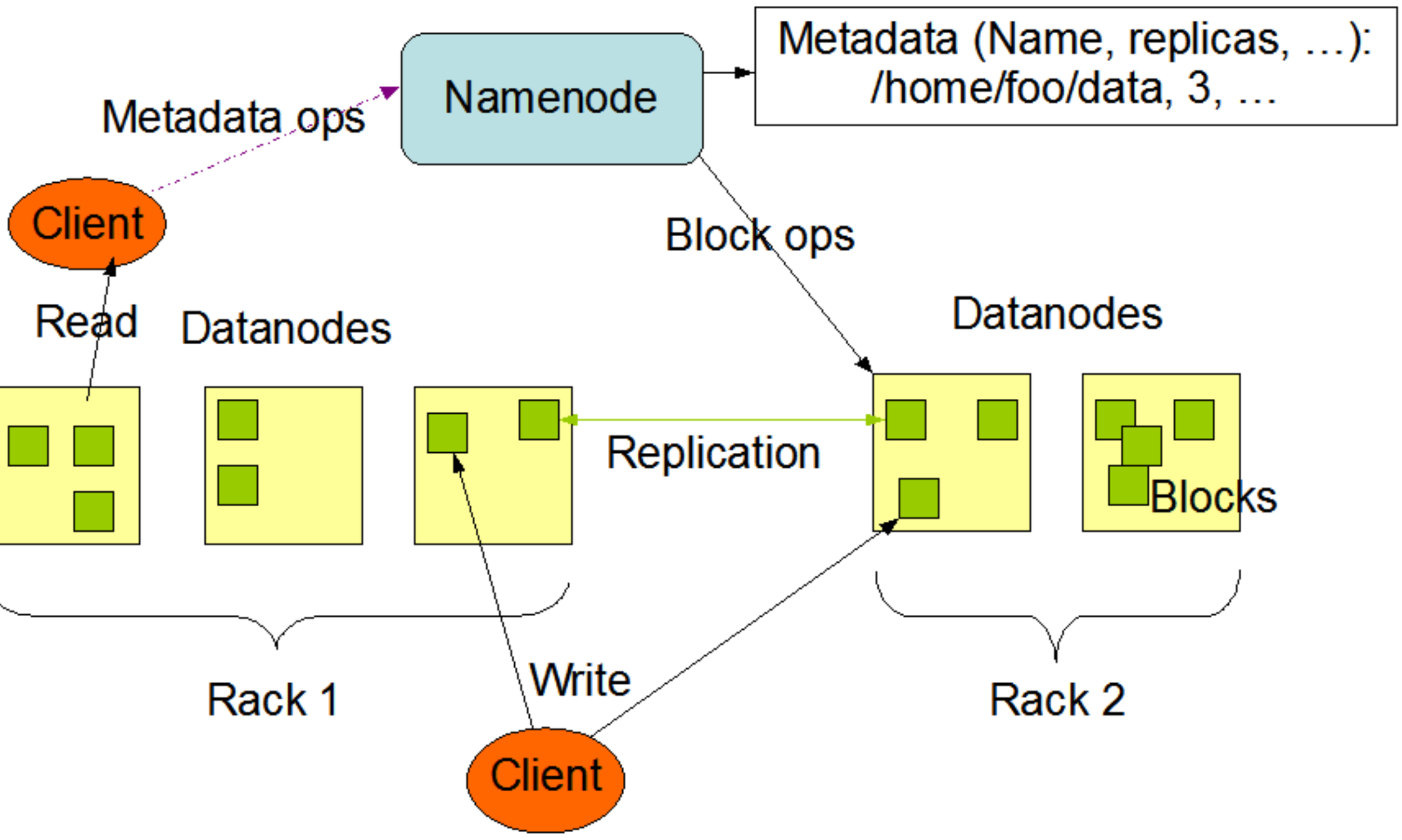
- HDFS是Hadoop Distributed File System 的缩写
- HDFS是一个可以运行在普通商用硬件设备上的分布式文件系统
    - 不需要运行在非常高可靠的硬件上
- 支持超大文件
    - 一般来说支持TB或者PB级的文件
- 高度容错性
    - 集群中数据会进行备份处理
- 适合一次读写，不适合多次读写操作
- 简单易用、不需要了解hadoop细节

HDFS架构

- HDFS是一个master/slave的架构。
- 一个HDFS的集群包含了1个NameNode, NameNode用来管理文件系统的命名空间和从客户端访问文件的规则
- 一个HDFS的集群包含了多个DataNodes，通常一个节点有一个DataNode。用来管理节点上的数据存储。
- HDFS一个一个文件系统的形式暴露给用户，使用户可以将数据以文件的形式将数据存储到HDFS中。
- 在HDFS内部，一个文件会被划分到一个或者多个blocks中，这些blocks存储在DataNodes中。
- NameNode会执行文件系统中的一些命令，open、close、rename文件或者目录。同样也会决定blocks与DataNodes的映射关系。
- DataNodes负责为来自客户端的读写请求提供服务。同样也负责执行来自于NameNode的block的创建、删除和重复(replication)的指令

HDFS架构

HDFS Architecture



HDFS架构

- 名字节点和数据节点是被设计运行在一般的商业机器上的软件。
- 这些机器一般是Linux操作系统。
- HDFS是使用Java语言编写的，任何可以运行Java的机器都可以运行名字节点和数据节点的程序。
- 正因为Java有着广泛的可移植性，HDFS也可以部署到各种各样的机器上。
- 一个典型的部署架构是：
- 　NameNode运行在集群的一台机器上
- 　DataNode运行在集群的其他机器上（实际中，很少有多个DataNode运行在同一个机器上）

Data Replication

- HDFS是运行在一个多机器的大集群中的可靠的文件系统。它将文件转换成一个block的序列，来存储。为了HDFS的高可用性，HDFS会对每个block进行复制多份存储。

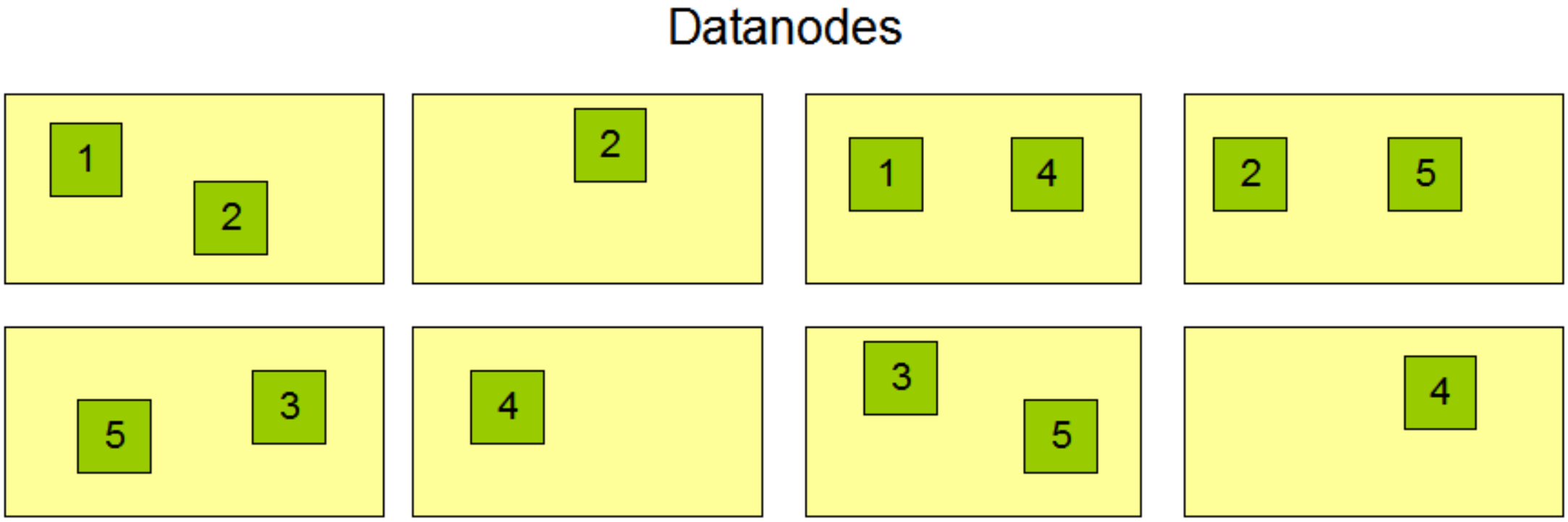
- Block的大小和备份的数目，可以在配置文件中设定。

- 关于block的复制，全部由NameNode节点进行决策。NodeName会周期的从DataNode节点接受一个心跳信号和Block的报告。来监控DataNode是否正常工作，其中Block的报告，包括DataNode中的所有block

Data Replication

Block Replication

Namenode (Filename, numReplicas, block-ids, …)
/users/sameerp/data/part-0, r:2, {1,3}, …
/users/sameerp/data/part-1, r:3, {2,4,5}, …



NameNode

- NameNode主要是用来保存HDFS的元数据信息，比如命名空间信息，块信息等。当它运行的时候，这些信息是存在内存中的。但是这些信息也可以持久化到磁盘上。

- fsimage： 保存的是上个检查点的HDFS的元信息, 它是在NameNode启动时对整个文件系统的快照

- edits： 保存的是从上个检查点开始发生的HDFS元信息状态改变信息，它是在NameNode启动后，对文件系统的改动序列

NameNode

```
(base) ec2-user@ip-172-31-45-49 current $ ll
total 2080
-rw-rw-r-- 1 ec2-user ec2-user      42 Jun 17 08:33 edits_0000000000000000001-0000000000000000002
-rw-rw-r-- 1 ec2-user ec2-user 1048576 Jun 17 09:06 edits_0000000000000000003-0000000000000000011
-rw-rw-r-- 1 ec2-user ec2-user      97 Jun 18 01:41 edits_0000000000000000012-0000000000000000014
-rw-rw-r-- 1 ec2-user ec2-user 1048576 Jun 18 01:41 edits_inprogress_0000000000000000015
-rw-rw-r-- 1 ec2-user ec2-user     635 Jun 18 00:51 fsimage_0000000000000000011
-rw-rw-r-- 1 ec2-user ec2-user      62 Jun 18 00:51 fsimage_0000000000000000011.md5
-rw-rw-r-- 1 ec2-user ec2-user     635 Jun 18 01:41 fsimage_0000000000000000014
-rw-rw-r-- 1 ec2-user ec2-user      62 Jun 18 01:41 fsimage_0000000000000000014.md5
-rw-rw-r-- 1 ec2-user ec2-user       3 Jun 18 01:41 seen_txid
-rw-rw-r-- 1 ec2-user ec2-user     216 Jun 18 00:51 VERSION
```

NameNode

- 只有在NameNode重启时，edits才会合并到fsimage文件中，从而得到一个文件系统的最新快照。但是在生产环境中NameNode是很少重启的，这也意味着当NameNode运行了很长时间后，edits文件会变得很大。

- 会引发下面3个问题：
  - edits文件会变的很大，怎么去管理这个文件是一个挑战。
  - NameNode的重启会花费很长时间，因为有很多改动(edits)要合并到fsimage文件上。
  - 如果NameNode挂掉了，那我们就丢失了很多改动因为此时的fsimage文件非常旧。（可能不会很多，很为大部分保存到edits中了，当然也hadoop也支持高可用的架构）
- 需要减小edits文件的大小和获得一个最新的fsimage文件，这样也会减小在NameNode上的压力

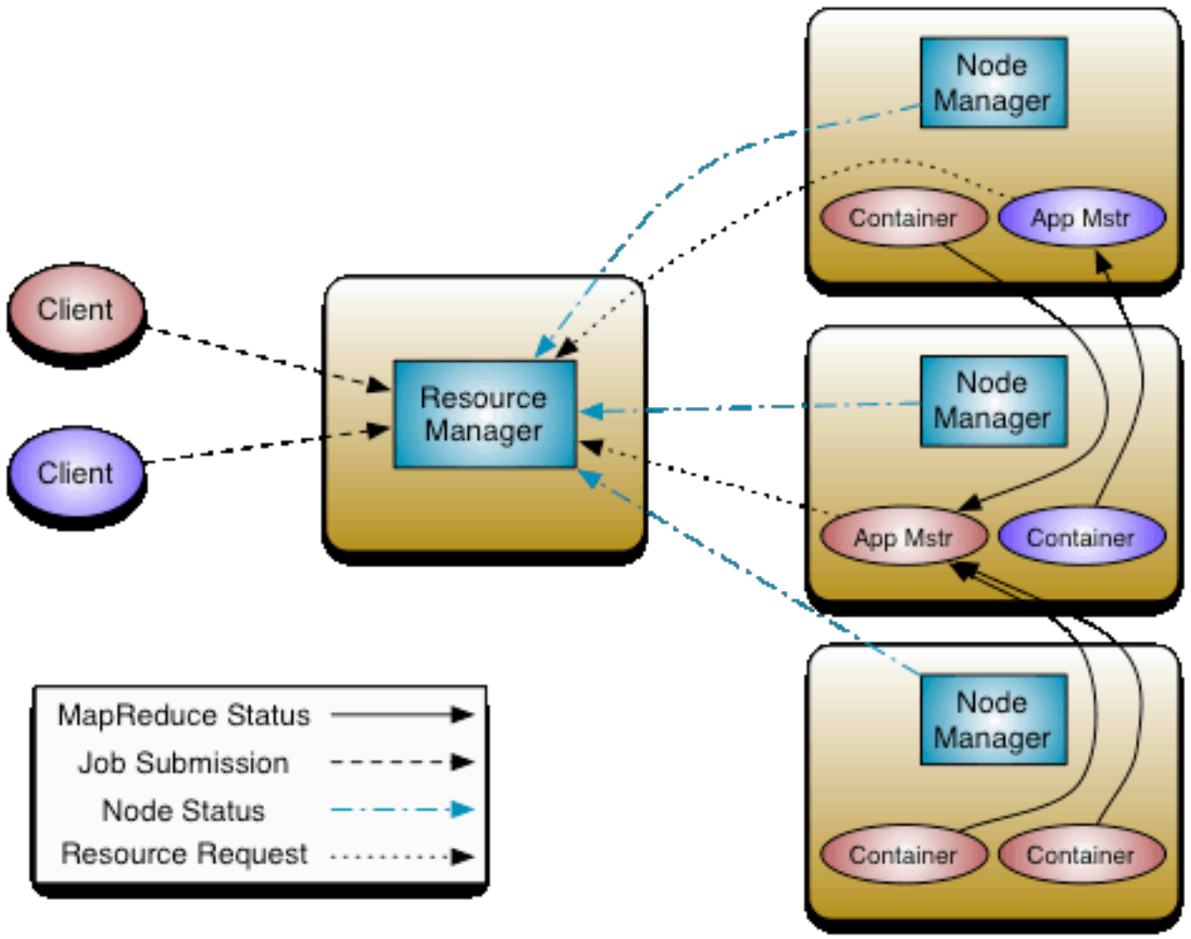
Secondary NameNode

- Secondary NameNode就是来帮助解决上述问题的，它的职责是合并NameNode的edit logs到fsimage文件中。Secondary NameNode 的工作流程
    - 首先，它定时到NameNode去获取edits，并更新到Secondary NameNode自己的fsimage, 一旦它有了新的fsimage文件，它将其拷贝回NameNode中。
        - NameNode在下次重启时会使用这个新的fsimage文件，从而减少重启的时间。
- Secondary NameNode的整个目的是在HDFS中提供一个检查点。它只是NameNode的一个助手节点。这也是它在社区内被认为是检查点节点的原因。
- Secondary NameNode所做是在文件系统中设置一个检查点来帮助NameNode更好的工作。
- 它不是要取代掉NameNode也不是NameNode的备份。

YARN

- YARN是一个资源管理、任务调度的框架，主要包含三大模块：
  - ResourceManager（RM）
  - NodeManager（NM）
  - ApplicationMaster（AM）
- ResourceManager负责所有资源的监控、分配和管理；
- ApplicationMaster负责每一个具体应用程序的调度和协调；
- NodeManager负责每一个节点的维护。
- 对于所有的applications，RM拥有绝对的控制权和对资源的分配权。而每个AM则会和RM协商资源，同时和NodeManager通信来执行和监控task。

YARN


Client
Client
Resource Manager
Node Manager
Container
App Mstr
Node Manager
App Mstr
Container
Node Manager
Container
Container
MapReduce Status
Job Submission
Node Status
Resource Request

ResourceManager

- ResourceManager负责整个集群的资源管理和分配，是一个全局的资源管理系统。

- NodeManager以心跳的方式向ResourceManager汇报资源使用情况（目前主要是CPU和内存的使用情况）。RM只接受NM的资源回报信息，对于具体的资源处理则交给NM自己处理。

NodeManager

- NodeManager是每个节点上的资源和任务管理器，它是管理这台机器的代理，负责该节点程序的运行，以及该节点资源的管理和监控。YARN集群每个节点都运行一个NodeManager。

- NodeManager定时向ResourceManager汇报本节点资源（CPU、内存）的使用情况和Container的运行状态。NodeManager接收并处理来自ApplicationMaster的Container启动、停止等各种请求。

ApplicationMaster

- 用户提交的每个应用程序均包含一个ApplicationMaster，它可以运行在ResourceManager以外的机器上。

- 负责与RM调度器协商以获取资源（用Container表示）。

- 将得到的任务进一步分配给内部的任务(资源的二次分配)。

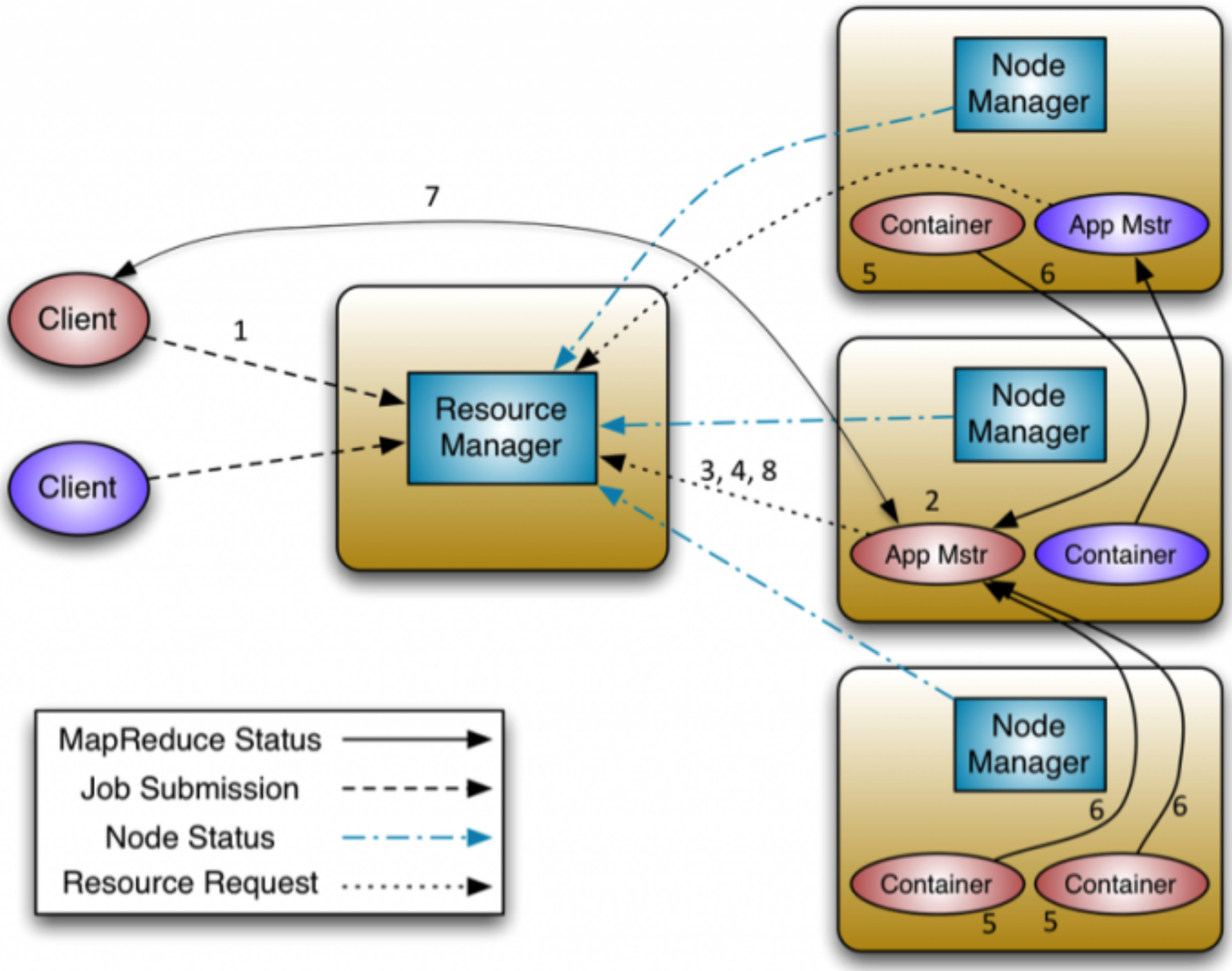
- 与NM通信以启动/停止任务。

- 监控所有任务运行状态，并在任务运行失败时重新为任务申请资源以重启任务。

执行流程

- client向RM提交应用程序，其中包括启动该应用的ApplicationMaster的必须信息，例如ApplicationMaster程序、启动ApplicationMaster的命令、用户程序等。
- ResourceManager启动一个container用于运行ApplicationMaster。
- 启动中的ApplicationMaster向ResourceManager注册自己，启动成功后与RM保持心跳。
- ApplicationMaster向ResourceManager发送请求，申请相应数目的container。
- ResourceManager返回ApplicationMaster的申请的containers信息。申请成功的container，由ApplicationMaster进行初始化。container的启动信息初始化后，AM与对应的NodeManager通信，要求NM启动container。AM与NM保持心跳，从而对NM上运行的任务进行监控和管理。
- container运行期间，ApplicationMaster对container进行监控。container通过RPC协议向对应的AM汇报自己的进度和状态等信息。
- 应用运行期间，client直接与AM通信获取应用的状态、进度更新等信息。
- 应用运行结束后，ApplicationMaster向ResourceManager注销自己，并允许属于它的container被收回。

执行流程


Node Manager
Container
App Mstr
5
6
7
Client
1
Client
Resource Manager
3, 4, 8
Node Manager
2
App Mstr
Container
Node Manager
6
6
Container
Container
5
5
MapReduce Status
Job Submission
Node Status
Resource Request

Hadoop集群搭建

- 下载hadoop-3.2.0.tar.gz
- sh build_hadoop.sh

```
#!/bin/sh

HADOOP_PATH=/home/ec2-user/hadoop_2
if [ ! -d ${HADOOP_PATH} ]; then
    mkdir -p ${HADOOP_PATH}
fi

cd ${HADOOP_PATH}
# Donwload Apache Hadoop 3.2.0
wget http://mirror.bit.edu.cn/apache/hadoop/common/hadoop-3.2.0/hadoop-3.2.0.tar.gz
tar xvf hadoop-3.2.0.tar.gz
~
```

Hadoop集群搭建

- 修改core-site.xml

- PATH：hadoop-3.2.0/etc/hadoop

```
<configuration>
  <property>
     <name>fs.default.name</name>
     <value>hdfs://ip-172-31-45-49:9000/</value>
  </property>
  <property>
    <name>hadoop.tmp.dir</name>
    <value> file:/home/ec2-user/hadoop/tmp</value>
  </property>
</configuration>
~
```

Hadoop集群搭建

- 修改core-site.xml
- fs.defaultFS：
  - 文件系统的名字
    - fs.default.name　Deprecated. Use (fs.defaultFS) property instead
- hadoop.tmp.dir:
  - 系统中基础的临时目录（默认在/tmp下面），系统重启后会删除tmp，要配置到一个可以持久化的目录

Hadoop集群搭建

- 修改hdfs-site.xml
- PATH：hadoop-3.2.0/etc/hadoop

```
<configuration>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>file:/home/ec2-user/hadoop/dfs/name</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>file:/home/ec2-user/hadoop/dfs/data</value>
  </property>
  <property>
    <name>dfs.replication</name>
    <value>3</value>
  </property>
  <property>
     <name>dfs.webhdfs.enabled</name>
     <value>true</value>
  </property>
  <property>
    <name>dfs.namenode.checkpoint.dir</name>
    <value>file:/home/ec2-user/hadoop/dfs/namesecondary</value>
  </property>
</configuration>
```

Hadoop集群搭建

- dfs.namenode.name.dir
  - 名字节点的路径
- dfs.datanode.data.dir
  - 数据节点的路径
- dfs.replication
  - 重复的份数
- dfs.namenode.checkpoint.dir
  - secondaryname节点的位置

Hadoop集群搭建

- 修改yarn-site.xml
- PATH：hadoop-3.2.0/etc/hadoop

```
<configuration>
  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>ip-172-31-37-59</value>
  </property>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
</configuration>
~
```

Hadoop集群搭建

- yarn.resourcemanager.hostname:
  - 指定Resourcemanager 运行在哪个节点上。
- yarn.nodemanager.aux-services:
  - 配置了 yarn 的默认混洗方式，选择为 mapreduce 的默认混洗算法。

Hadoop集群搭建

- 修改map-site.xml
- PATH：hadoop-3.2.0/etc/hadoop

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
</configuration>
~
```

Hadoop集群搭建

- 修改mapred-site.xml
- mapreduce.framework.name：
  - mapreduce依赖的框架

Hadoop集群搭建

- 修改workers
- PATH：hadoop-3.2.0/etc/hadoop

```
ip-172-31-45-49
ip-172-31-37-59
~
~
```

Hadoop集群搭建

- 修改hadoop-env.sh
- PATH：hadoop-3.2.0/etc/hadoop

```
# The java implementation to use. By default, this environment
# variable is REQUIRED on ALL platforms except OS X!
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.201.b09-0.43.amzn1.x86_64
```

Hadoop集群搭建

- 修改~/.bashrc
- 修改后 source ~/.bashrc

```
# HADOOP
export HADOOP_HOME=/home/ec2-user/hadoop/hadoop-3.2.0
export PATH=${HADOOP_HOME}/bin:$HADOOP_HOME/sbin:${PATH}
```

Hadoop集群搭建

- 格式化名字节点

```
(base) ec2-user@ip-172-31-45-49 hadoop (add-pyspark-scirpts-0601) $ hdfs namenode -format
2019-06-17 08:29:44,480 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = ip-172-31-45-49.cn-northwest-1.compute.internal/172.31.45.49
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 3.2.0
STARTUP_MSG:   classpath = /home/ec2-user/DataScience/hadoop/hadoop-3.2.0/etc/hadoop:/home/ec2-user/DataScience/hadoop/hadoop-3.2.0/share/hadoop/common/lib/kerby-asn1-1.0.1.jar:/home/ec2-user/DataScience/hadoop/hadoop-3.2.0/share/hadoop/common/lib/httpclient-4.5.2.jar:/home/ec2-user/DataScience/hadoop/hadoop-3.2.0/share/hadoop/common/lib/jsp-api-2.1.jar:/home/ec2-user/DataScience/hadoop/hadoop-3.2.0/share/hadoop/common/lib
2019-06-17 08:29:45,281 INFO common.Storage: Storage directory /home/ec2-user/DataScience/hadoop/dfs/name has been successfully formatted.
2019-06-17 08:29:45,290 INFO namenode.FSImageFormatProtobuf: Saving image file /home/ec2-user/DataScience/hadoop/dfs/name/current/fsimage.ckpt_0000000000000000000 using no compression
2019-06-17 08:29:45,382 INFO namenode.FSImageFormatProtobuf: Image file /home/ec2-user/DataScience/hadoop/dfs/name/current/fsimage.ckpt_0000000000000000000 of size 403 bytes saved in 0 seconds .
2019-06-17 08:29:45,391 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
2019-06-17 08:29:45,396 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at ip-172-31-45-49.cn-northwest-1.compute.internal/172.31.45.49
************************************************************/
(base) ec2-user@ip-172-31-45-49 hadoop (add-pyspark-scirpts-0601) $ ls dfs/name/current/
fsimage_0000000000000000000  fsimage_0000000000000000000.md5  seen_txid  VERSION
```

Hadoop集群搭建

- 启动hadoop集群

```
(base) ec2-user@ip-172-31-45-49 hadoop $ jps
4624 Master
4713 Worker
11404 Jps
(base) ec2-user@ip-172-31-45-49 hadoop $ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as ec2-user in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [ip-172-31-45-49]
Starting datanodes
Starting secondary namenodes [ip-172-31-45-49]
Starting resourcemanager
Starting nodemanagers
(base) ec2-user@ip-172-31-45-49 hadoop $ jps
4624 Master
12469 Jps
11751 DataNode
4713 Worker
11946 SecondaryNameNode
11595 NameNode
12301 NodeManager
```

Hadoop集群搭建

- 停止hadoop集群

```
(base) ec2-user@ip-172-31-45-49 hadoop $ jps
10656 Jps
4624 Master
9763 SecondaryNameNode
9414 NameNode
4713 Worker
9550 DataNode
(base) ec2-user@ip-172-31-45-49 hadoop $ stop-all.sh
WARNING: Stopping all Apache Hadoop daemons as ec2-user in 10 seconds.
WARNING: Use CTRL-C to abort.
Stopping namenodes on [ip-172-31-45-49]
Stopping datanodes
Stopping secondary namenodes [ip-172-31-45-49]
Stopping nodemanagers
Stopping resourcemanager
(base) ec2-user@ip-172-31-45-49 hadoop $ jps
4624 Master
4713 Worker
11404 Jps
```

Hadoop集群搭建

- HDFW webUI
- NameNoe:9870

Run Spark on HDFS

- 将Hadoop的配置文件导入到spark-env.sh

```
export SPARK_MASTER_IP=172.31.45.49
export SPARK_MASTER_WEBUI_PORT=8888
export HADOOP_CONF_DIR=${HADOOP_HOME}
```

```
python als_train.py spark://172.31.45.49:7077 'als training' hdfs://ip-172-31-45-49:9000/data/ml-100k/u.data 5 10
```

Hadoop常用命令

- Hadoop Common: Hadoop 的一个common包，支持各个功能
- Hadoop Distributed File System (HDFS™)：高吞吐量的分布式文件系统
- Hadoop YARN：一个job管理和集群资源调度的框架
- Hadoop MapReduce：基于yarn的分布式计算框架

Hadoop常用命令

- # 查看hdfs根目录
- hdfs dfs -ls /
- # 查看hdfs其它目录(/data)
- hdfs dfs -ls /data
- # 查看目录占用磁盘空间
- hdfs dfs -du /data
- # 将字节自动转化成单位
- hdfs dfs -du -h /data
- # 查看根目录磁盘空间
- hdfs dfs -df /

Hadoop常用命令

- # 将client 节点中中的tmp_file文件 put 到hdfs的/文件夹中
- hdfs dfs -put tmp_file /data
- # 如果文件存在，可以使用-f参数进行覆盖
- hdfs dfs -put -f data /data
- # 获取hdfs中的/data/ml-100k/u.data文件，到客户端的当前目录
- hdfs dfs -get /data/ml-100k/u.data
- # 获取hdfs中的文件，get到指定目录中
- hdfs dfs -get /data/ml-100k/u.data ./temp/
- # 将hdfs中/data/ml-100k/u.data 复制到 /user/zhang/test目录
- hdfs dfs -cp /user/zhang/NOTICE.txt /user/zhang/test

Hadoop常用命令

- # 创建一个temp_folder目录
- hdfs dfs -mkdir /temp_folder
- # 删除/tmp_file文件
- hdfs dfs -rm /tmp_file
- # 删除/temp_folder目录
- hdfs dfs -rm -r /user/zhang/abc
- # 删除/temp_folde目录
- hdfs dfs rmdir /temp_folde
- # 查看文件
- hdfs dfs -cat /tmp_file
- hdfs dfs -tail /tmp_file
- hdfs dfs -mv /tmp_file /tmp_file_2