

Apache Hadoop简介



目录

- ❖ 1.Hadoop 简介
- ❖ 2.Hadoop 起源
- ❖ 3.Hadoop 原理
- ❖ 4.Hadoop 部署
- ❖ 5.Hadoop 配置
- ❖ 6.Hadoop 监控
- ❖ 7.Hadoop 应用
- ❖ 8.Hadoop 展望

1.Hadoop 简介

1.Hadoop 简介

❖ Hadoop目前是Apache旗下的顶级项目之一，是Google在2004年提出的“MapReduce”分布式计算框架的一个Java实现。

“MapReduce”是一种简化的分布式编程模型，让程序自动分布到一个由普通机器组成的超大集群上并发执行。MapReduce的run-time系统会解决输入数据的分布细节，跨越机器集群的程序执行调度，处理机器的失效，并且管理机器之间的通讯请求。这样的模型允许程序员可以不需要有什么并发处理或者分布式系统的经验，就可以处理超大的分布式系统的资源。

2.Hadoop 起源

2.Hadoop 起源

- ❖ Google提出的“MapReduce”分布式计算框架，主要分为以下几个部分：



2.Hadoop 起源

❖ 分布式大规模数据处理MapReduce

在Google数据中心会有大规模数据需要处理，比如被网络爬虫（Web Crawler）抓取的大量网页等。由于这些数据很多都是PB级别，导致处理工作不得不尽可能的并行化，而Google为了解决这个问题，引入了 MapReduce这个编程模型，MapReduce是源自函数式语言，主要通过"Map（映射）"和"Reduce（化简）"这两个步骤来并行处理大规模的数据集。

Map会先对由很多独立元素组成的逻辑列表中的每一个元素进行指定的操作，且原始列表不会被更改，会创建多个新的列表来保存Map的处理结果。也就意味着，Map操作是高度并行的。当Map工作完成之后，系统会先对新生成的多个列表进行清理（Shuffle）和排序，之后会这些新创建的列表进行Reduce操作，也就是对一个列表中的元素根据Key值进行适当的合并。

2.Hadoop 起源

❖ 分布式基础设施 GFS:

由于搜索引擎需要处理海量的数据，所以Google的两位创始人Larry Page和Sergey Brin在创业初期设计一套名为"BigFiles"的文件系统，而GFS（全称为"Google File System"）这套分布式文件系统则是"BigFiles"的延续。

2.Hadoop 起源

❖ 分布式数据库BigTable

基于GFS提供分布式数据存储

❖ 协调系统Chubby

负责集群中各个节点的状态协调等

❖ 开发语言Sawzall

提供了可简易操作的开发语言

2.Hadoop 起源

- ❖ Hadoop的最初版本(现在称为HDFS和MapReduce)由Doug Cutting和Mike Cafarella于2004年开始实施。
- ❖ 2006年1月Doug Cutting加入Yahoo, 该项目也随之进入Yahoo。
- ❖ 2006年2月Apache Hadoop项目正式启动以支持MapReduce和HDFS的独立发展。同时雅虎的网格计算团队采用Hadoop。
- ❖ 至今Yahoo仍为Hadoop的主要贡献者。

3.Hadoop 原理

3.Hadoop 原理

❖ Hadoop与Google MapReduce的对应关系

Google calls it:	Hadoop equivalent:
MapReduce	Hadoop MapReduce
GFS	HDFS
Sawzall	Hive, Pig
BigTable	HBase
Chubby	ZooKeeper

3.Hadoop 原理

❖ Hadoop core

Hadoop的核心子项目，提供了一个分布式文件系统(HDFS)和支持MapReduce的分布式计算。

❖ HBase

建立在Hadoop内核之上，提供可靠的，可扩展的分布式数据库。

❖ ZooKeeper

一个高效的，可扩展的协调系统。分布式应用可以使用ZooKeeper来存储和协调关键共享状态。

❖ PIG

建立于Hadoop内核之上，是一种支持并行计算运行框架的高级数据流语言。

3.Hadoop 原理

- ❖ Hadoop最主要的就是Core，它又分为HDFS和MapReduce两个部分，前者提供分布式数据存储，后者提供任务的分发和归拢。其他组件都是围绕着这两个核心进行工作。

3.Hadoop 原理

❖HDFS的设计初衷:

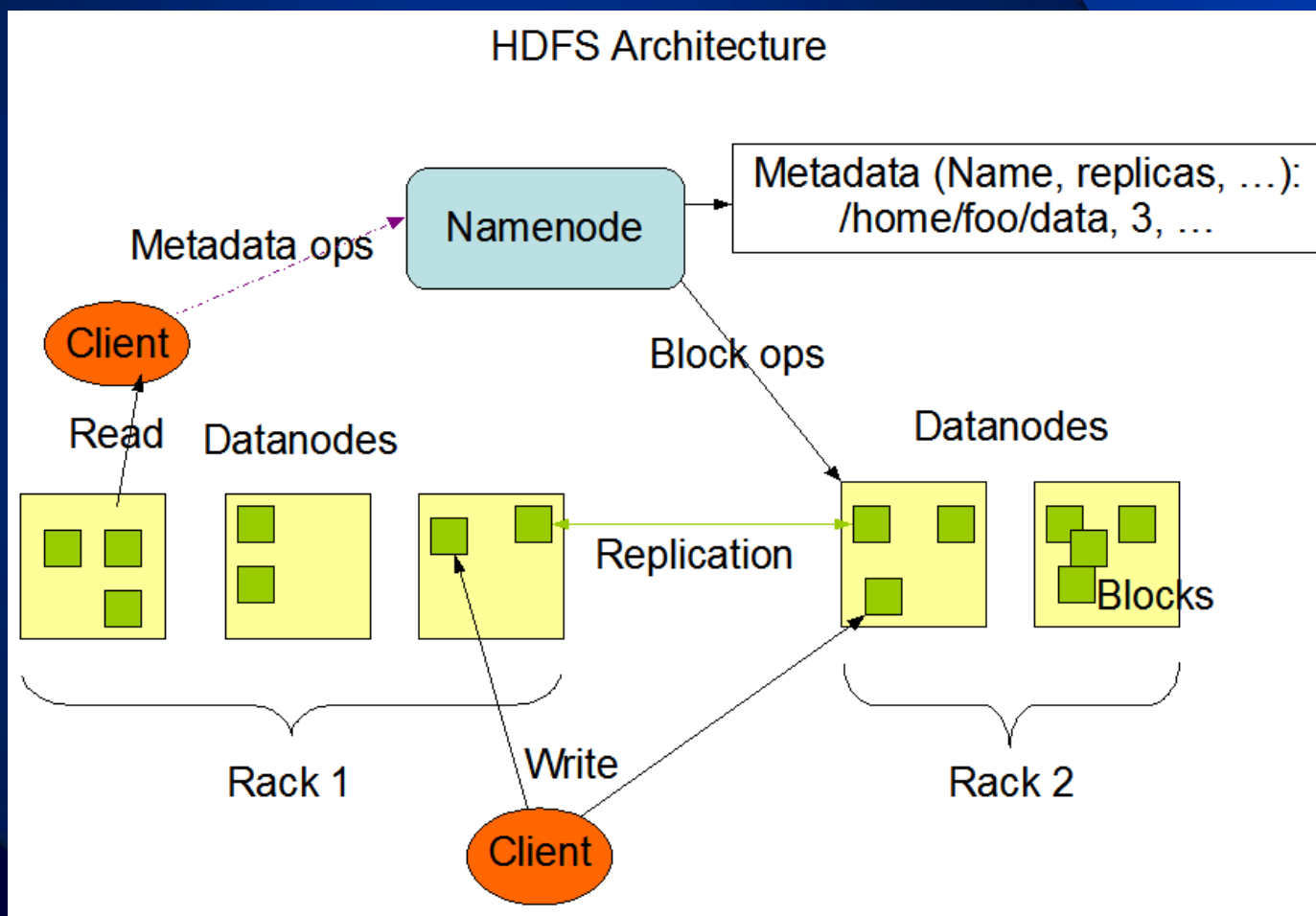
一次写入多次读取
不支持文件并发写入
不支持文件修改

❖HDFS的适用范围:

存储并管理PB级数据
处理非结构化数据
注重数据处理的吞吐量，对延时不敏感
应用模式为一次写入多次读取存取模式

3.Hadoop 原理

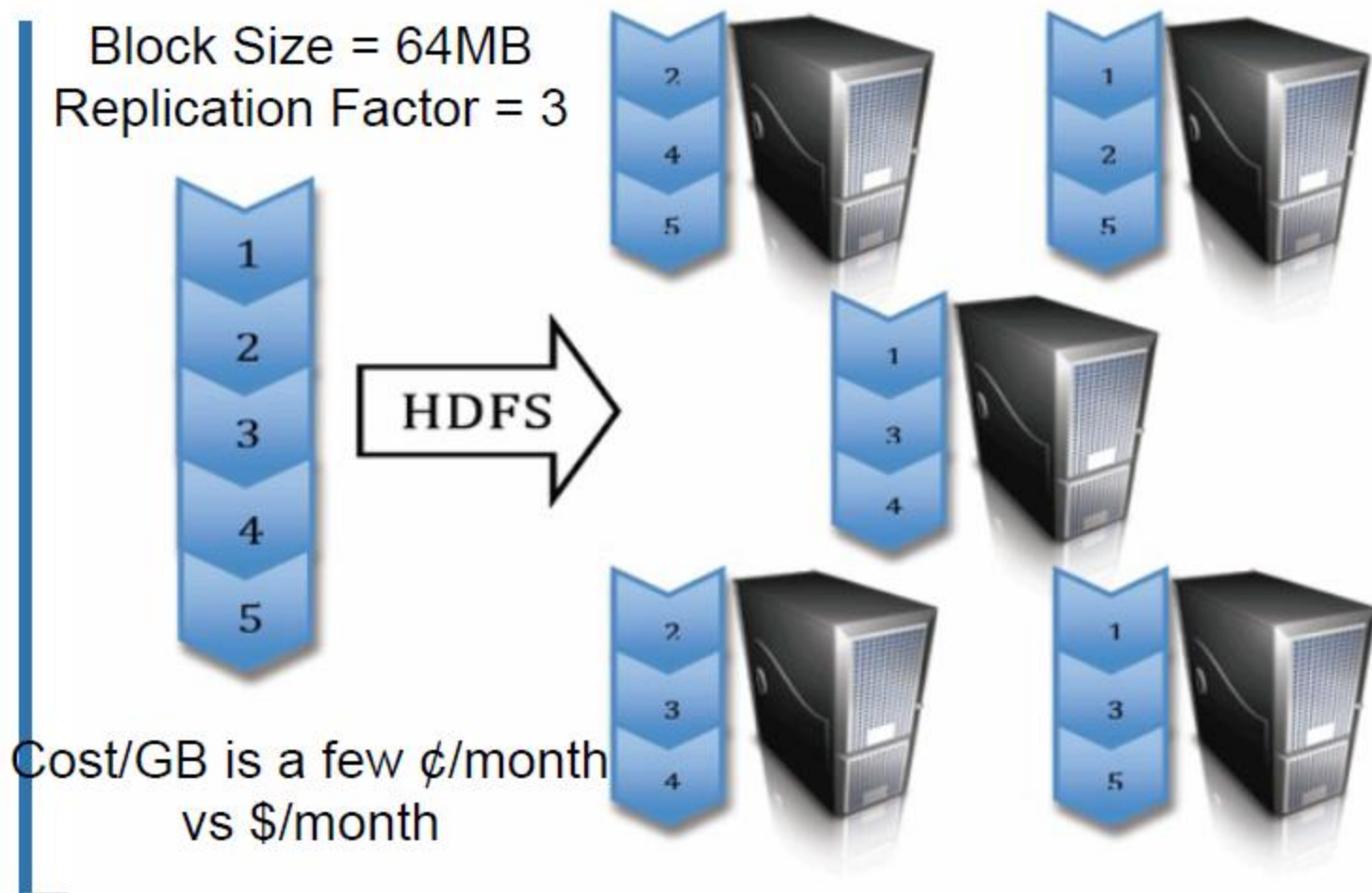
❖HDFS工作原理



3.Hadoop 原理

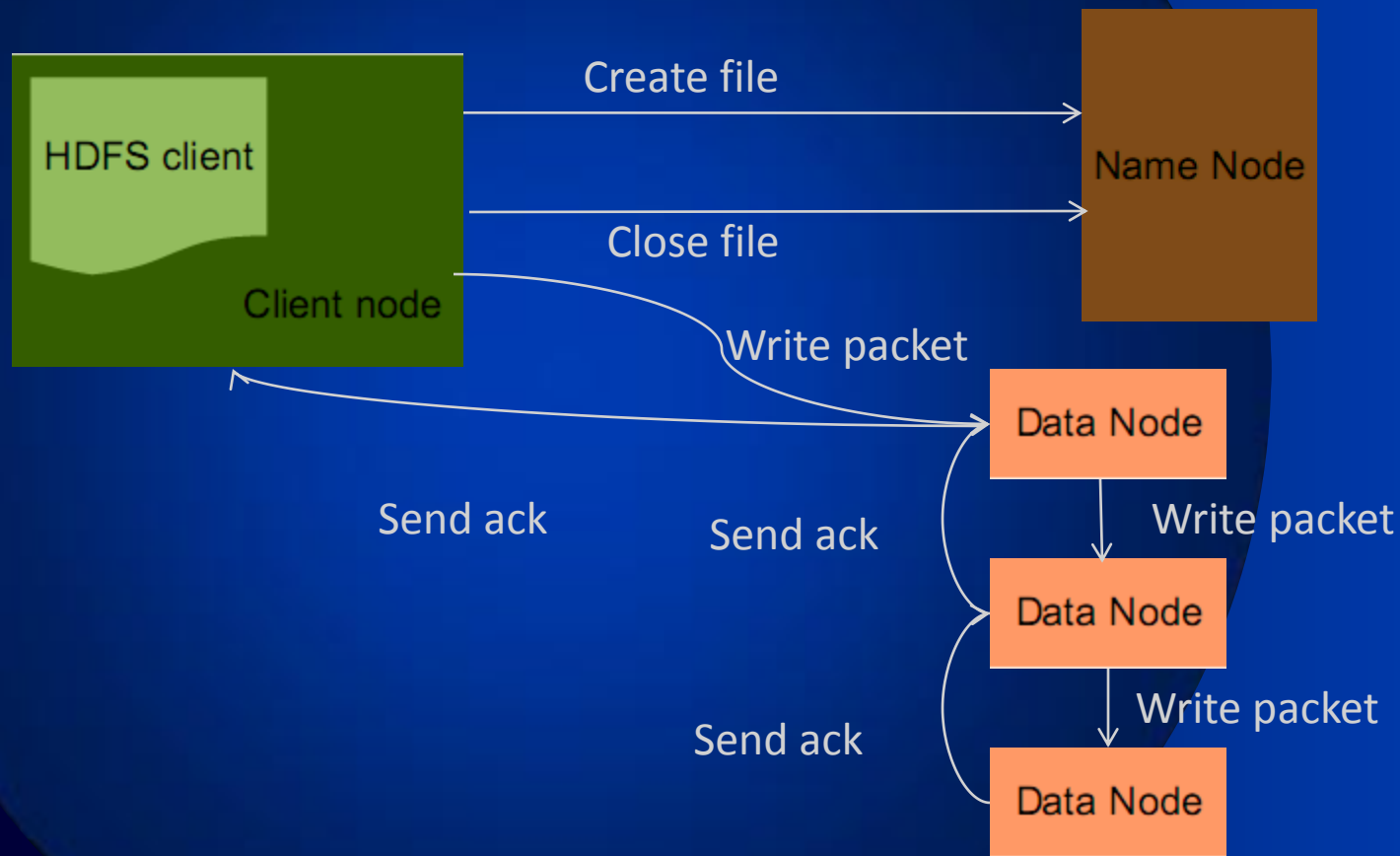
HDFS: Hadoop Distributed File System

Block Size = 64MB
Replication Factor = 3



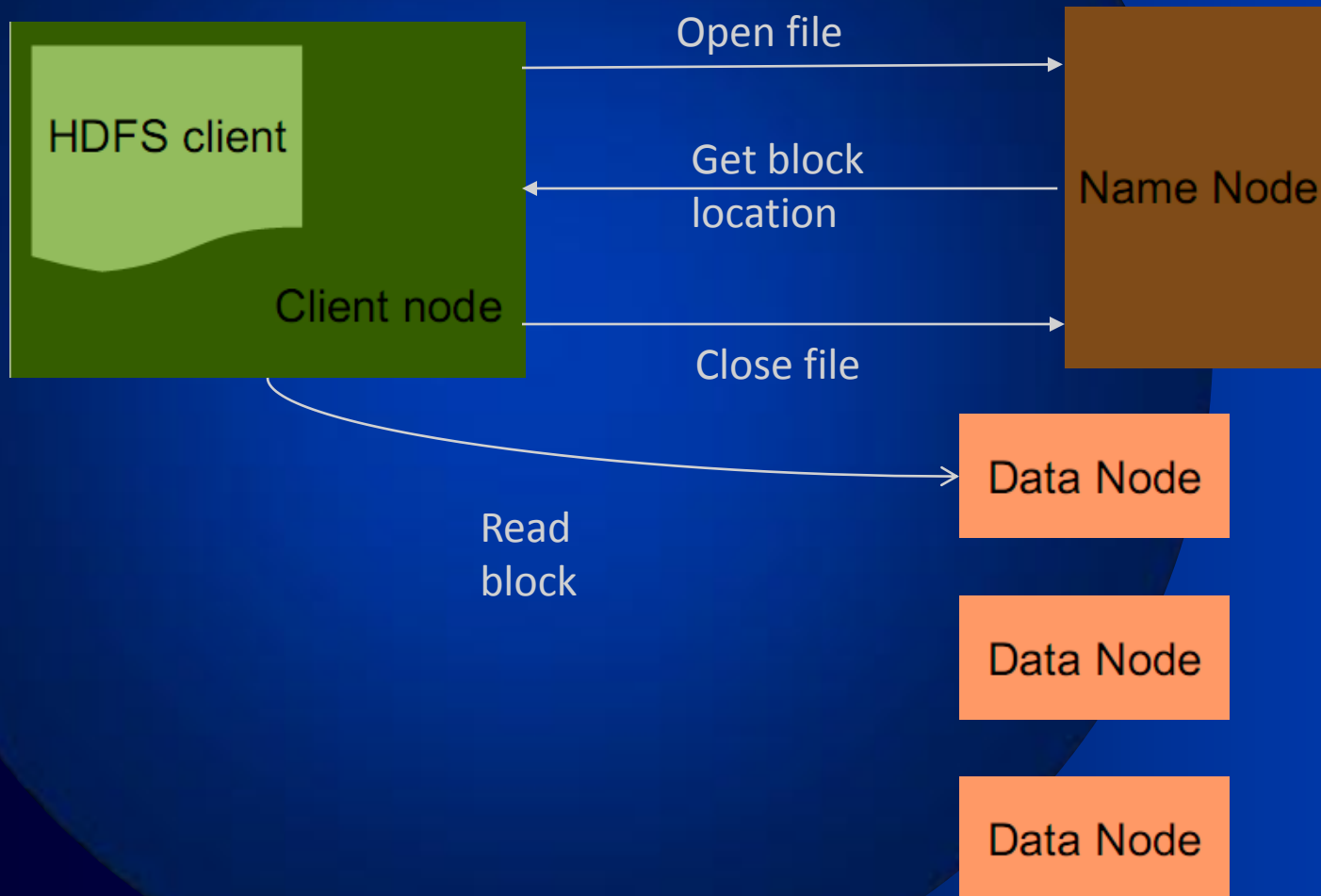
3.Hadoop 原理

❖HDFS如何写文件:



3.Hadoop 原理

❖HDFS如何读文件



3.Hadoop 原理

❖HDFS上传文件流程:

当客户端上传数据时，先向NameNode请求创建文件，然后NameNode会返回存储的DataNode节点和位置信息。

随后客户端先通过管道将文件传到本地硬盘，每凑满指定大小（默认是64M），再一起上传到一个DataNode，DataNode收到文件后会返回确认信息，并以4K为单位传到下一DataNode。

该文件上传方式被称为“流水式”。

3.Hadoop 原理

❖ MapReduce:

MapReduce来源于函数式编程语言。它由两个动词Map和Reduce组成，“Map（展开）”就是将一个任务分解成为多个任务，“Reduce”就是将分解后多任务处理的结果汇总起来，得出最后的分析结果。

不论是现实社会，还是在程序设计中，一项工作往往可以被拆分成多个任务，任务之间的关系可以分为两种：一种是不相关的任务，可以并行执行；另一种是任务之间有相互的依赖，先后顺序不能够颠倒，这类任务是无法并行处理的。

3.Hadoop 原理

❖ MapReduce理念:

处理海量数据 (>1TB)

上百/上千CPU实现并行处理

简单地实现以上目的

分而治之 (Divide and Conquer)

3.Hadoop 原理

❖MapReduce特性:

自动实现分布式并行计算

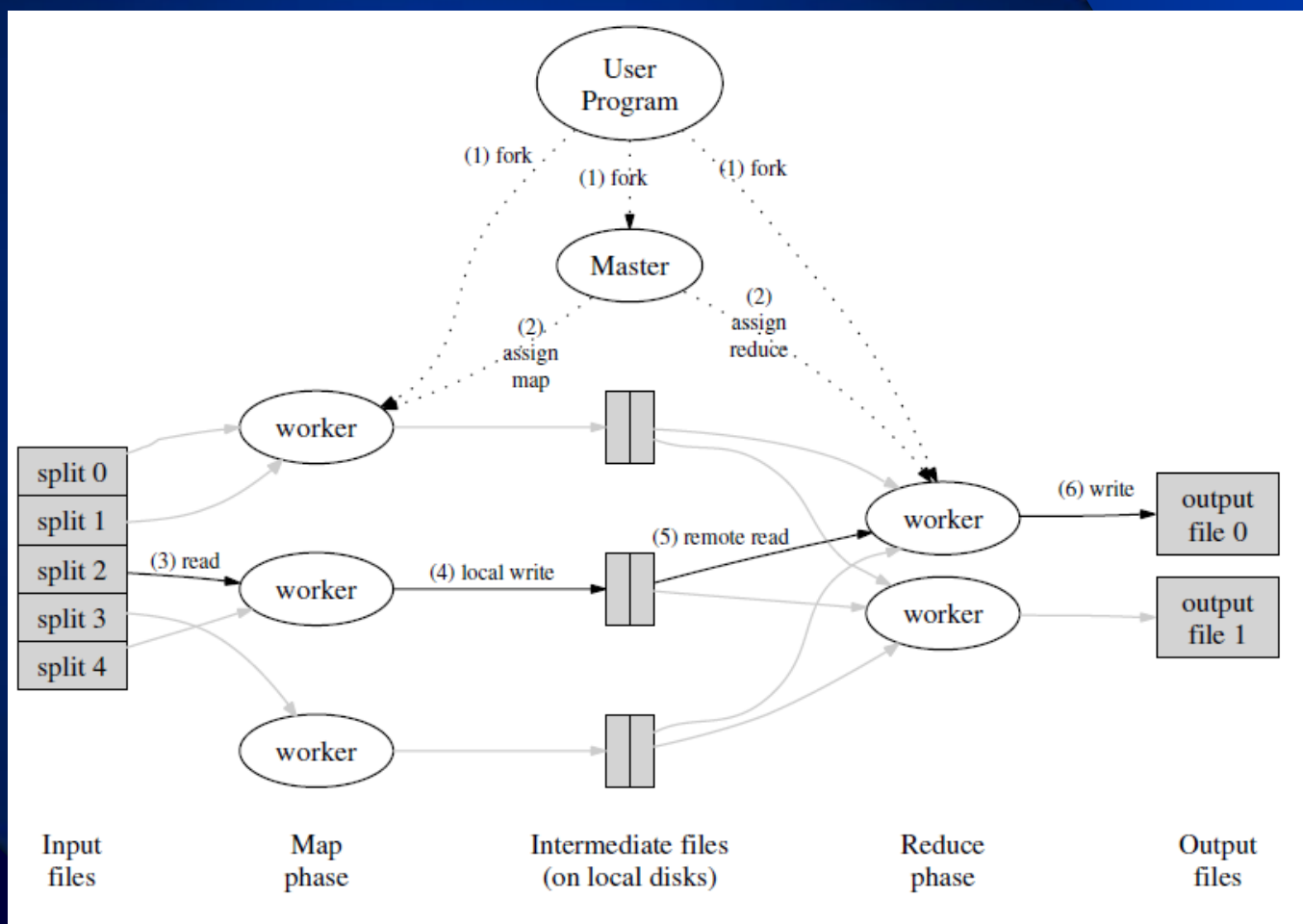
容错

提供状态监控工具

模型抽象简洁，程序员易用

3.Hadoop 原理

❖ MapReduce架构:



3.Hadoop 原理

❖ MapReduce示例:

示例: WordCount

■ 源数据

—Page 1:

- the weather is good

—Page 2:

- today is good

—Page 3:

- good weather is good

3.Hadoop 原理

❖ MapReduce 示例:

map 输出

- Worker 1:

- (the 1), (weather 1), (is 1), (good 1).

- Worker 2:

- (today 1), (is 1), (good 1).

- Worker 3:

- (good 1), (weather 1), (is 1), (good 1).

3.Hadoop 原理

❖ MapReduce 示例:

reduce 的输入

- Worker 1:
 - (the 1)
- Worker 2:
 - (is 1), (is 1), (is 1)
- Worker 3:
 - (weather 1), (weather 1)
- Worker 4:
 - (today 1)
- Worker 5:
 - (good 1), (good 1), (good 1), (good 1)

3.Hadoop 原理

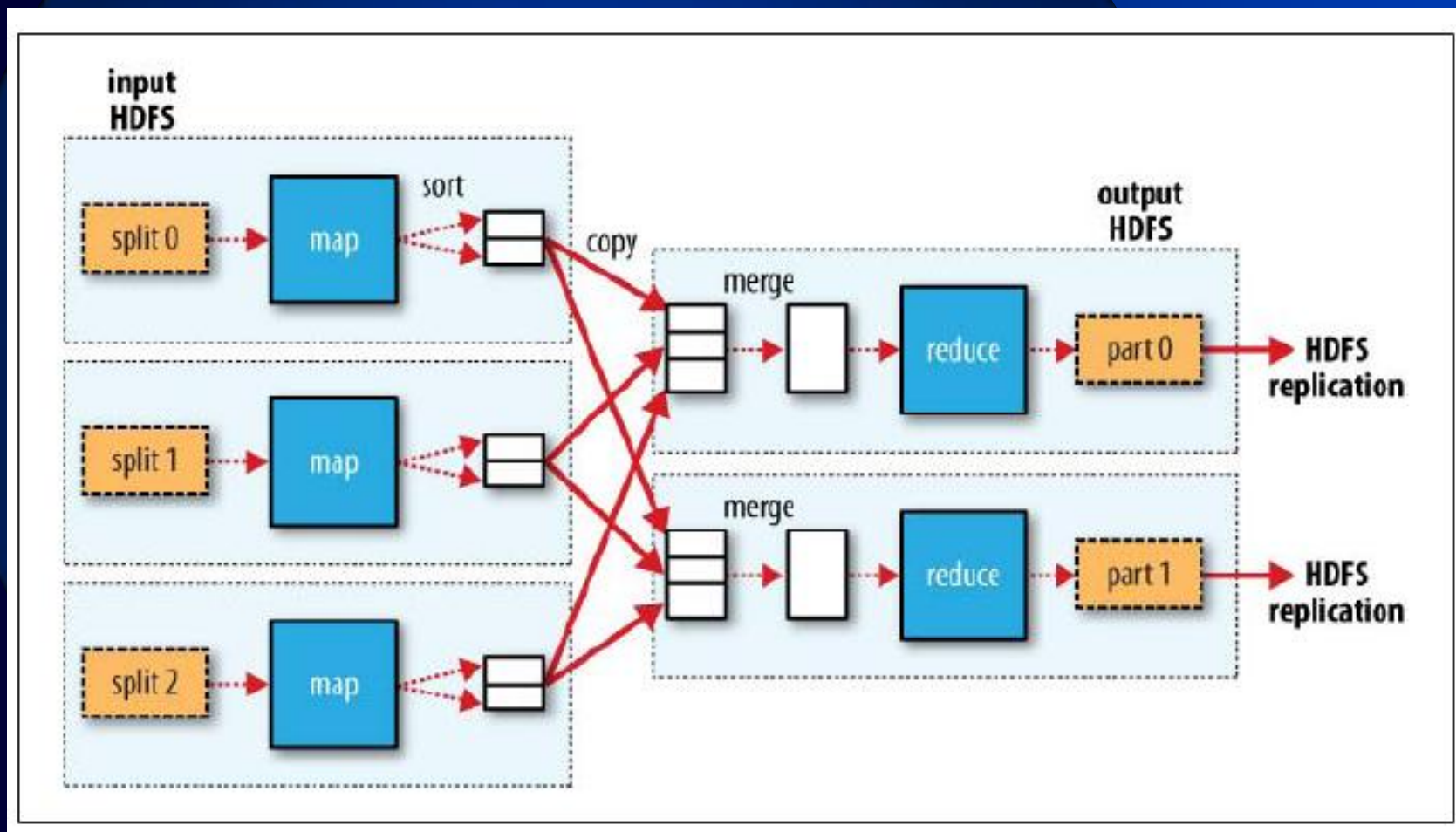
❖ MapReduce示例:

reduce输出

- Worker 1:
 - (the 1)
- Worker 2:
 - (is 3)
- Worker 3:
 - (weather 2)
- Worker 4:
 - (today 1)
- Worker 5:
 - (good 4)

3.Hadoop 原理

❖多Reduce处理示意图:



3.Hadoop 原理

❖ **Reduce**处理数据时，为了提高速度，并不是把结果直接写入硬盘，而是先在内存中进行缓存和排序，才会一起写入硬盘。

缓冲区的默认大小是100M，当数据量达到设定的阈值（默认0.8）后，会开始向硬盘导出数据，导出过程中数据将继续写入缓存区。如果导出数据比计算数据慢，则当缓存用满时，**Map**进程将被阻塞。

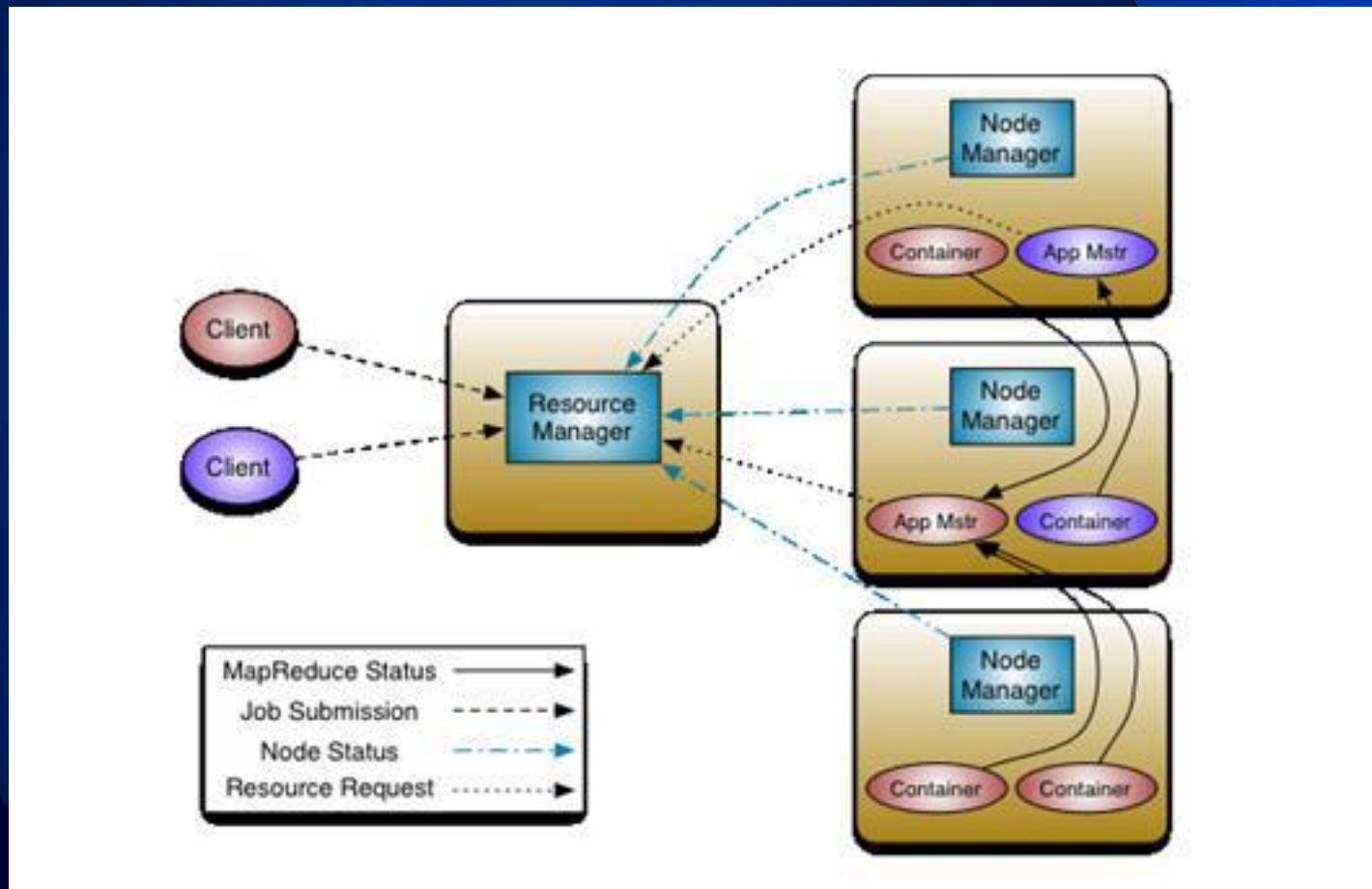
3.Hadoop 原理

❖Yarn的出现

随着Hadoop使用的规模和负荷逐渐增长，旧有的MapReduce架构弊端逐渐显现：JobTracker单点故障、JobTracker资源占用、任务分配逻辑过于简单、源码class功能复杂、升级维护麻烦等，Hadoop推出了新的分布式计算框架MapReduceV2，又名Yarn。

3.Hadoop 原理

❖Yarn的架构图:



3.Hadoop 原理

❖ Yarn的理念:

将资源管理（ResourceManager）和任务调度监控（NodeManager）分开。每个任务都有各自的ApplicationMaster负责资源调度和协调。

ResourceManager负责基于应用对资源池进行总体调度，会监控ApplicationMaster的存活状态并重启失败进程，但不对任务进程进行监控和跟踪，也不负责重启失败的任务进程。

NodeManager是每一台机器框架的代理，是执行应用程序的容器，监控应用程序的资源使用情况并且向调度器汇报。

ApplicationMaster负责向调度器索要适当的资源容器，运行任务，跟踪应用程序的状态和监控它们的进程，处理任务的失败原因。

3.Hadoop 原理

❖新的 Yarn 架构，各组件间的任务分工更明确，既尽量保留了老版本的 API 使程序只需做尽量小的改变，又实现了新功能。尤其是对集群资源的调配，比老版本的粒度小得多，可以更精细的控制任务地分配。

在新的 Yarn 中，ApplicationMaster 是一个可变更的部分，用户可以对不同的编程模型写自己的 AppMst，让更多类型的编程模型能够跑在 Hadoop 集群中。

4.Hadoop 部署

4.Hadoop 部署

❖ Hadoop的官网是：

<http://hadoop.apache.org/>

我们可以从以下链接下载到最新版可执行文件：

<http://hadoop.apache.org/releases.html>

但是官网只提供了x32平台的可执行文件，如果需要在x64平台上运行，需要自行下载源代码进行编译。

4.Hadoop 部署

- ❖ Hadoop编译需要Maven支持，编译后会生成和官网下载的可执行tar包结构一致的目标文件，直接解压即可使用。
- ❖ Hadoop运行需要JAVA环境支持，系统中需要部署JAVA运行环境。
- ❖ Hadoop运行需要集群各主机间建立SSH信任关系，请先给运行Hadoop的用户建立无密码互访。
- ❖ Hadoop可以通过IP或HOST通讯，如果使用HOST方式，请配置/etc/hosts文件或部署DNS环境。

4.Hadoop 部署

- ❖ JAVA运行环境和SSH互信配置完成后，只要把Hadoop的压缩包解压到运行目录（以下以“{\$HADOOP_HOME}”表示），在用户的配置文件中将{\$HADOOP_HOME}\bin、{\$HADOOP_HOME}\sbin目录加入PATH变量，确认下这两个目录中的几个可执行文件是否有运行权限，就完成了Hadoop的基本部署。

5.Hadoop 配置

5.Hadoop 配置

- ❖ Hadoop运行分三种方式：单机、伪集群、集群。
- ❖ Hadoop部署好后，默认就是单机模式。只有在有任务的时候才会启动Hadoop进程，并且只有一个进程，该进程将完成所有计算任务。
- ❖ 必须要修改配置文件才能启动伪集群和集群模式。这两种模式将按正常方式启动所有进程，也是常用的方式。

5.Hadoop 配置

- ❖ Hadoop的配置文件在Hadoop安装目录的etc/hadoop下，核心是core-site、hdfs-site、mapred-site、yarn-site四个xml文件。
core-site.xml包含Hadoop的全局配置。
hdfs-site.xml包含HDFS部分的具体配置。
mapred-site.xml包含MapReduce部分的具体配置。
yarn-site.xml包含Yarn部分的具体配置。

5.Hadoop 配置

❖ core-site.xml:

需要修改配置项有两项

fs.defaultFS: 设定集群名称

hadoop.tmp.dir: 设定临时文件目录

❖ mapred-site.xml:

需要修改的配置项也有两项

mapreduce.framework.name: 设定MapReduce方式

mapreduce.jobtracker.address: 设定Job Tracker的服务地址

❖ yarn-site.xml:

需要修改的配置有一项

yarn.resourcemanager.hostname: 设定RM主机的主机名

5.Hadoop 配置

❖ hdfs-site.xml:

dfs.ha.namenodes.集群名: NN主机列表

dfs.namenode.rpc-address.集群名.主机名:
NN主机的服务监听端口

dfs.namenode.http-address.集群名.主机名:
NN主机的WEB-UI端口

dfs.namenode.shared.edits.dir: 多NN之间共享存储的位置

dfs.journalnode.edits.dir: JN保存edits文件的位置

5.Hadoop 配置

❖ 以上配置中，有几项是关于地址、端口或主机名称的配置项。

如果要启动伪集群模式，只要把这些配置为本机回环地址（**localhost**）即可。

如果要启动集群模式，只要把这些配置为集群中对应的主机名即可。

6.Hadoop 监控

6.Hadoop 监控

❖ Hadoop启动后会在各节点启动WEB-UI，管理员可以通过浏览器访问指定端口来查看集群或各节点的基本信息。

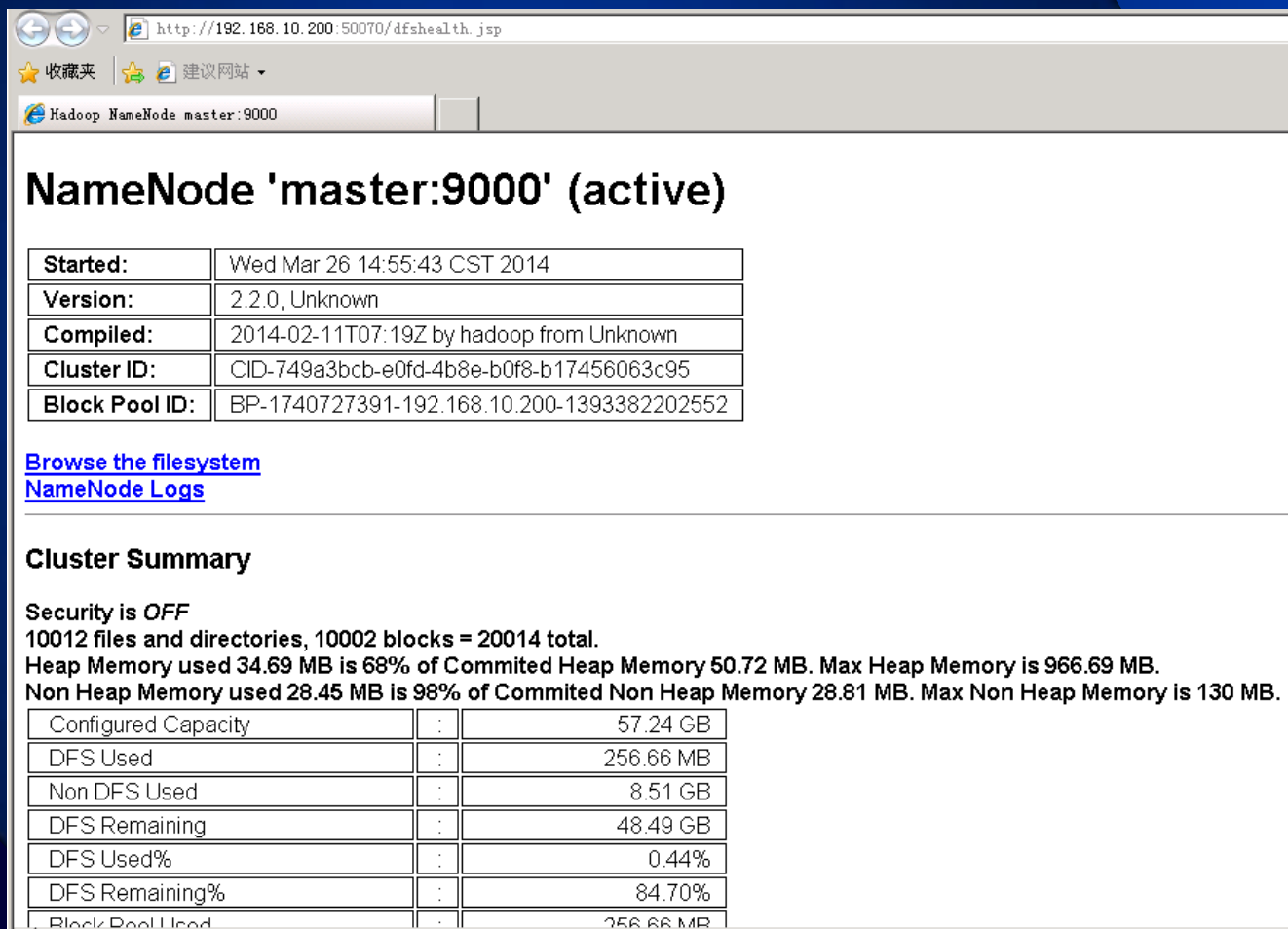
NameNode的默认监听端口是50070，

DataNode的默认监听端口是50075，

Yarn资源管理器的默认监听端口是8088。

6.Hadoop 监控

❖ NameNode的WEB-UI页面:



http://192.168.10.200:50070/dfshealth.jsp

收藏夹 | 建议网站

Hadoop NameNode master:9000

NameNode 'master:9000' (active)

Started:	Wed Mar 26 14:55:43 CST 2014
Version:	2.2.0, Unknown
Compiled:	2014-02-11T07:19Z by hadoop from Unknown
Cluster ID:	CID-749a3bcb-e0fd-4b8e-b0f8-b17456063c95
Block Pool ID:	BP-1740727391-192.168.10.200-1393382202552

[Browse the filesystem](#)
[NameNode Logs](#)

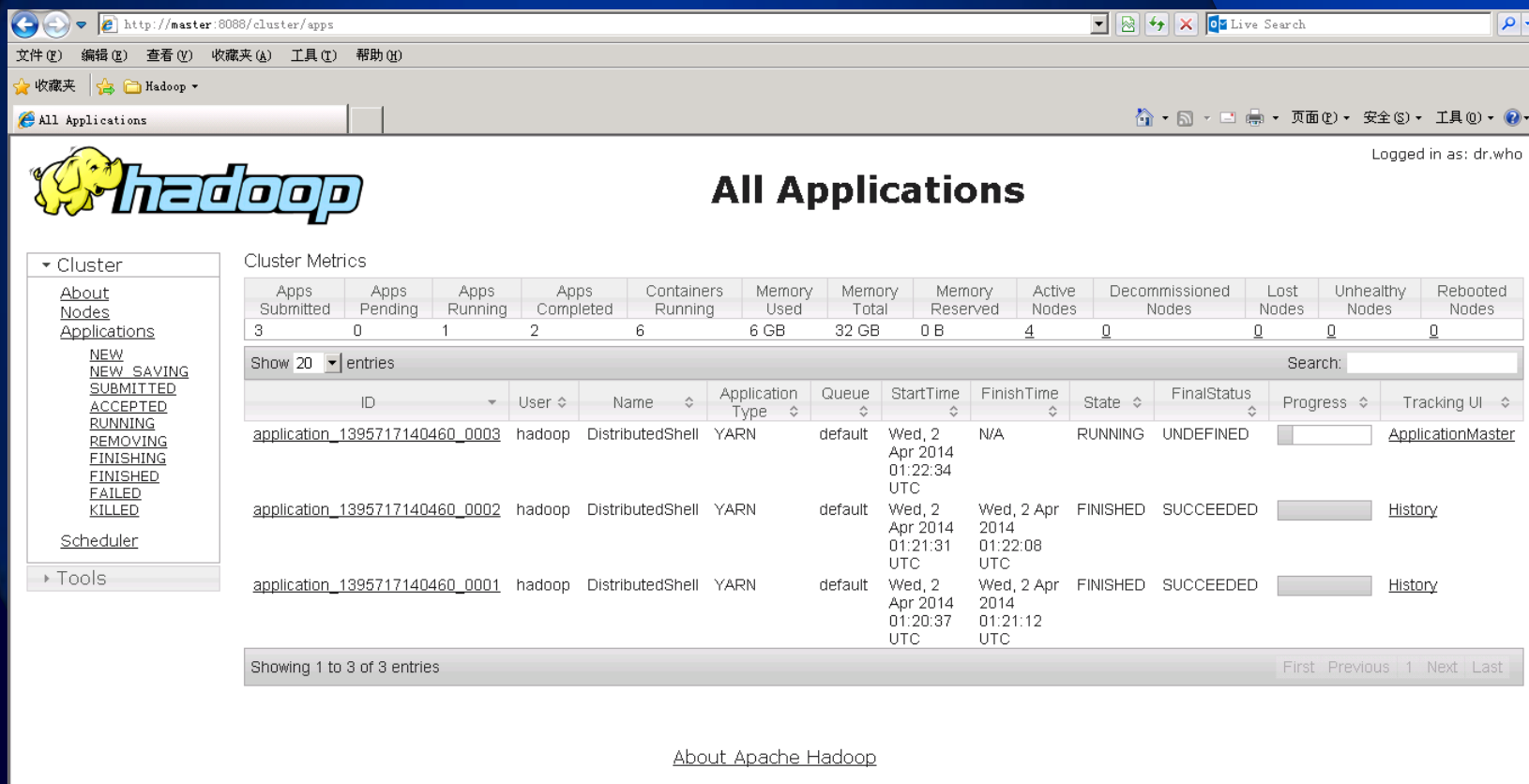
Cluster Summary

Security is OFF
10012 files and directories, 10002 blocks = 20014 total.
Heap Memory used 34.69 MB is 68% of Committed Heap Memory 50.72 MB. Max Heap Memory is 966.69 MB.
Non Heap Memory used 28.45 MB is 98% of Committed Non Heap Memory 28.81 MB. Max Non Heap Memory is 130 MB.

Configured Capacity	:	57.24 GB
DFS Used	:	256.66 MB
Non DFS Used	:	8.51 GB
DFS Remaining	:	48.49 GB
DFS Used%	:	0.44%
DFS Remaining%	:	84.70%
Block Pool Used	:	256.66 MB

6.Hadoop 监控

❖ Yarn资源管理器的WEB-UI页面:



The screenshot displays the Hadoop Yarn Resource Manager Web UI. The top navigation bar shows the URL `http://master:8088/cluster/apps` and the user is logged in as `dr.who`. The main content area is titled "All Applications" and features a table of applications. The table has columns for ID, User, Name, Application Type, Queue, StartTime, FinishTime, State, FinalStatus, Progress, and Tracking UI. The table lists three applications: `application_1395717140460_0003` (Running), `application_1395717140460_0002` (Finished), and `application_1395717140460_0001` (Finished). The sidebar on the left contains links for Cluster, About, Nodes, Applications, and Tools. The bottom of the page includes a link to "About Apache Hadoop".

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
3	0	1	2	6	6 GB	32 GB	0 B	4	0	0	0	0

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI
application_1395717140460_0003	hadoop	DistributedShell	YARN	default	Wed, 2 Apr 2014 01:22:34 UTC	N/A	RUNNING	UNDEFINED	<div></div>	ApplicationMaster
application_1395717140460_0002	hadoop	DistributedShell	YARN	default	Wed, 2 Apr 2014 01:21:31 UTC	Wed, 2 Apr 2014 01:22:08 UTC	FINISHED	SUCCEEDED	<div></div>	History
application_1395717140460_0001	hadoop	DistributedShell	YARN	default	Wed, 2 Apr 2014 01:20:37 UTC	Wed, 2 Apr 2014 01:21:12 UTC	FINISHED	SUCCEEDED	<div></div>	History

Showing 1 to 3 of 3 entries

[First](#) [Previous](#) [1](#) [Next](#) [Last](#)

[About Apache Hadoop](#)

6.Hadoop 监控

❖现在用于监控Hadoop集群状态的常用软件是Ganglia和Chukwa，其他还有Cacti、Zabbix等。报警软件使用最多的则是老牌开源监控软件Nagios。

下面将介绍使用Ganglia监控Hadoop集群的方法。

6.Hadoop 监控

- ❖ Ganglia是一个很强大的常用开源监控软件，Hadoop默认就对他提供了很好的支持。Hadoop的监控配置文件是Hadoop安装目录的etc/hadoop下的hadoop-metrics.properties文件。

6.Hadoop 监控

❖ 打开文件就可以看到内置了四个Ganglia的配置项dfs、mapred、rpc、ugi，我们只需要把之前的注释去掉就能打开，其他就需要自己编写了。

```
# Configuration of the "dfs" context for ganglia
# Pick one: Ganglia 3.0 (former) or Ganglia 3.1 (latter)
# dfs.class=org.apache.hadoop.metrics.ganglia.GangliaContext
# dfs.class=org.apache.hadoop.metrics.ganglia.GangliaContext31
# dfs.period=10
# dfs.servers=localhost:8649
```

```
# Configuration of the "rpc" context for ganglia
# rpc.class=org.apache.hadoop.metrics.ganglia.GangliaContext
# rpc.class=org.apache.hadoop.metrics.ganglia.GangliaContext31
# rpc.period=10
# rpc.servers=localhost:8649
```

```
# Configuration of the "mapred" context for ganglia
# Pick one: Ganglia 3.0 (former) or Ganglia 3.1 (latter)
# mapred.class=org.apache.hadoop.metrics.ganglia.GangliaContext
# mapred.class=org.apache.hadoop.metrics.ganglia.GangliaContext31
# mapred.period=10
# mapred.servers=localhost:8649
```

```
# Configuration of the "ugi" context for ganglia
# ugi.class=org.apache.hadoop.metrics.ganglia.GangliaContext
# ugi.class=org.apache.hadoop.metrics.ganglia.GangliaContext31
# ugi.period=10
# ugi.servers=localhost:8649
```


6.Hadoop 监控

❖ Ganglia的监控界面:



7.Hadoop 应用

7.Hadoop 应用

❖ MapReduce的应用:

并行计算处理引擎

【1】日志分析

【2】排序

【3】搜索

【4】广告计算，广告优化、分析，点击流分析，链接分析

【5】搜索关键字进行内容分类

【6】搜索引擎，创建索引

【7】word 计数，统计值计算，统计数据，过滤，分析，查询

【8】垃圾数据分析

【9】数据分析

【10】机器学习

【11】数据挖掘

【12】大规模图像转换

7.Hadoop 应用

❖正在使用Hadoop的企业:

云服务提供商: Amazon、Yahoo

购物平台: eBay、淘宝

搜索引擎: Google、百度

SNS社交网络: LinkedIn、Facebook

科技企业: IBM

等等.....

7.Hadoop 应用

❖ Amazon、Yahoo:

Yahoo作为Hadoop的推动者功不可没。目前Yahoo运营着世界最大的Hadoop集群为学术机构和大公司提供服务。同时Hadoop还用于垃圾邮件过滤系统。

亚马逊是目前世界上最大的云计算服务提供商，他的Amazon Elastic Compute Cloud（亚马逊弹性计算云，简称EC2）正在为世界上数以万计的各种规模公司提供着计算及存储服务。

7.Hadoop 应用

❖ 淘宝、阿里巴巴:

这个可能是大家接触最多的Hadoop应用实例，经常可以在各个页面看到根据你之前浏览、购买的记录而向你推荐的商品。



7.Hadoop 应用

❖ 京东:

京东和淘宝类似，他通过分析点击流，统计出了浏览当前商品的最终用户购买了什么，向你提供类似条件下其他人的选择。

浏览了该商品的用户最终购买



华硕 (ASUS) B85M-E 主板(Intel B85/LGA 1150)
¥549.00



技嘉 (GIGABYTE) B85M-D3H 主板 (Intel B85/LGA 1150)
¥579.00

浏览了该商品的用户最终购买



九州风神 (DEEPCOOL) 玄冰400 多平台 CPU散热器 12025发光风扇 四热管 可调速
¥99.90



九州风神 (DEEPCOOL) 冰凌MINI 旗舰版 多平台CPU散热器 适用于AMD AM2/AM2+AM3 INTEL LGA775/1156
¥39.90

7.Hadoop 应用

❖ eBay、Paypal:

eBay是世界最大的综合电子商务平台，Paypal则是世界最大的电子支付平台。他们都掌握着庞大的用户消费数据，二者结合后更是从销售-购物-支付全覆盖。

目前eBay每天要使用Hadoop处理点击流、图像等8T~10T数据，生成搜索索引、商品推荐、分析报告等，同时部署了Storm进行相关数据流的实时处理。

7.Hadoop 应用

❖ 纽约时报使用Hadoop和EC2在36个小时内将4TB的TIFF图像（包括 40.5万个大TIFF图像、3百多万万个的SGML文章和40多万万个XML文件）转换为80万个适合在Web上使用的PNG图像。

该实例充分展现了Hadoop强大的的分布式计算能力。

7.Hadoop 应用

❖Hadoop使用的其他范例：

奇虎360：使用Hadoop存储软件管家中软件，使用CDN技术将用户请求引到最近的Hadoop集群并进行下载。

百度：存储、分析日志、数据挖掘和机器学习（主要是推荐系统）

广告类公司：存储日志，通过协调过滤算法为客户推荐广告

华为：云计算平台

8.Hadoop 展望

8.Hadoop 展望

❖ Hadoop有一些局限性，比如在向HDFS写数据有延时，在做Map-Reduce的时候比较慢，不适应数据快速处理等。

其中有些问题可以通过调整操作系统和Hadoop配置优化，但是更多是由于架构本身的特征决定的。

随着Hadoop的发展，他本身也在弥补自身的不足：更新HDFS存储策略、Yarn替代之前的MapReduce、加快信息处理速度、加入数据流处理技术等。

8.Hadoop 展望

❖独立操作系统

Hadoop目前依赖于底层操作系统支持，连HDFS都是建立在系统本地文件系统之上。虽然这样降低了系统复杂度，但是难免对集群性能造成影响。

未来Hadoop可能会发展为一个独立的操作系统，直接构建于硬件之上，带来更快的速度和更高的稳定性。