

分布式统计计算

6. MapReduce 和 Spark 框架

李舰

华东师范大学

2018-12-24

目 录

- 1 相关环境介绍
 - 云计算简介
 - 容器与虚拟化
 - Linux 基础
 - Docker 简介
- 2 Hadoop 和 MapReduce
- 3 Spark 简介

目 录

- 1 相关环境介绍
 - 云计算简介
 - 容器与虚拟化
 - Linux 基础
 - Docker 简介
- 2 Hadoop 和 MapReduce
- 3 Spark 简介

计算方式的变迁

● 大型机与超算

- 大型机大规模地应用到业界已经有大半个世纪的历史了，这个领域的巨头是 IBM 公司。大型机由于高安全性（目前尚无黑客可以攻击）、高可靠性（全年宕机时间不超过 5 分钟、可以在不重启的情况下运行 10 年）、强大的事务处理能力（一天可以处理百亿次事务），从一开始就成为对数据有高端需求的公司的标配。
- 超算（超级计算机），也被称为巨型机，主要用来进行非常复杂的大规模计算，通常用于科研和军方目的。

● 云计算的兴起

- 通过大量便宜的 PC 服务器实现并行计算，从上世纪 80 年代开始就一直有着广泛的应用。
- 2006 年，亚马逊公司将其弹性计算云服务命名为“云计算”，后来云计算这个词就成了这种远程并行框架的代称。
- 2004 年谷歌向公众发布了大规模数据处理框架 MapReduce，2006 年，Apache 启动了对 Hadoop 项目的独立支持，发展成通用的 MapReduce 实现，使得云计算流行起来。

云计算的特点

● 基于 PC 服务器

- 与大型机的方式不同，基于 X86 的 PC 服务器集群作为硬件服务器，形成了当今主流的云计算平台方案。
- 这种方式的好处是硬件便宜、维护容易、性能强大，缺点是比较费电、占空间大。

● 可扩容

- 如果能够在数据量比较小、服务器数目比较少的时候就开发和部署一套分布式的存储与分析框架，当未来数据量暴增的时候，就可以很简单地通过添加硬件的方式实现扩容，而无需像传统的方式那样进行漫长的备份、迁移、升级。
- 对初创公司来说，规模小的时候用比较少的机器搭建集群，一旦业务量激增，只需要买入新的硬件接入集群即可，成本的增长几乎是线性的，不会被大的系统商绑架。

● 数据的应用层次

- 数据的存储，将数据保存并管理起来，通常提供查询功能。
- 数据的处理，通常进行一些汇总、变换等简单操作。
- 数据的分析，通常需要进行复杂的分析和建模。

目 录

- 1 相关环境介绍
 - 云计算简介
 - 容器与虚拟化
 - Linux 基础
 - Docker 简介
- 2 Hadoop 和 MapReduce
- 3 Spark 简介

Docker 简介

● 容器与 hypervisor 虚拟化

- hypervisor 虚拟化通过中间层将一台或多台独立的机器虚拟运行于物理硬件上。
- 容器是直接运行在操作系统内核之上的用户空间。
- 容器拥有更高的资源使用效率，单个操作系统能够承载更多的容器。

● Docker 简介

- dotCloud 公司于 2010 年在旧金山成立，专注于 PaaS 平台。
- 2013 年，dotCloud 将其核心技术 Docker 开源，基于 Apache 2.0 协议。Docker 是一个能够把开发的应用程序自动部署到容器的引擎，很快风靡全球，随后 dotCloud 公司改名为 Docker Inc。
- 2014 年，Docker 把 PaaS 业务 dotCloud 出售给位于 cloudControl。
- 2016 年 2 月，CloudControl 在官方博客中宣告即将破产。

Windows 下安装 Docker

● 基于 Docker for Windows 安装

- Docker for Windows 是一个社区版的软件 (Community Edition)，由 Docker 官方发布，包含了所有依赖和运行环境。
- 在 <https://docs.docker.com/docker-for-windows/install/> 下载最新的安装版和参考文档。
- 目前只能安装在 Windows 10 64 位下的专业版、企业版和教育版中，要求系统支持虚拟化的功能，内存大于 4G。

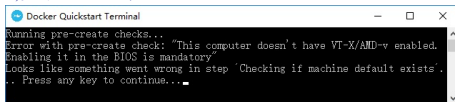
● 基于 Docker Toolbox 安装

- 如果计算时是 Win7、Win8 或者 Win10 的家庭版，需要通过 Docker Toolbox 来安装。
- 在 https://docs.docker.com/toolbox/toolbox_install_windows/ 下载最新的版本和参考文档。
- 需要同时安装一个 Oracle 的虚拟机，在模拟的 MINGW 环境中运行和使用 Docker。

Windows 下安装 Docker

● 安装 Docker Toolbox

- 下载 Docker Toolbox 后，默认安装即可，在桌面上会生成 3 个图标，双击“Docker Quickstart Terminal”会打开 Docker。
- 第一次启动时会配置运行环境，如果出现如下出错提示，说明需要启动虚拟化功能：



```
Docker Quickstart Terminal
Running pre-create checks...
Error with pre-create check: "This computer doesn't have VT-X/AMD-v enabled.
Enabling it in the BIOS is mandatory"
Looks like something went wrong in step 'Checking if machine default exists'.
.. Press any key to continue..._
```

- 重启计算机，在 BIOS 设置中开启虚拟化（virtualization）的功能即可。
 - 第一次启动时还会联网下载 boot2docker.iso，需要耐心等待。
- ## ● 设置和使用
- 如果默认安装，工作目录和启动路径为 **C:\Program Files\Docker Toolbox**。打开桌面的“Oracle VM VirtualBox”图标，可以设置虚拟机的存储路径。
 - 打开桌面的“Docker Quickstart Terminal”可以进入一个虚拟的 Linux 环境，在该环境下可以运行 Docker 的命令。

Linux 下安装 Docker

● Ubuntu 下安装

- 更新软件源然后自动安装：

```
sudo apt-get update  
sudo apt-get install docker.io
```

- 查看安装后的版本：

```
sudo docker version
```

- 安装后会自建一个 docker 组（如果没有可以手动建组），然后把当前用户加入该组，就可以省掉 **sudo**。

```
sudo groupadd docker  
sudo gpasswd -a jian docker
```

● 相关路径

- 默认安装于 **/usr/bin** 目录，可以直接调用命令，无需修改环境变量。
- 默认的资源路径位于 **/var/lib/docker**。
- 默认下载的镜像文件位于 **/var/lib/docker/graph**。

目 录

- 1 相关环境介绍
 - 云计算简介
 - 容器与虚拟化
 - Linux 基础
 - Docker 简介
- 2 Hadoop 和 MapReduce
- 3 Spark 简介

系统安装（以 Ubuntu 为例）

● 制作安装 U 盘

- 选择想要安装的版本（当前稳定版是 18.04），到官网（<http://www.ubuntu.com/download/alternative-downloads>）下载相应版本的 ISO 文件，版本中带有“LTS”标记的表示会长期提供支持，此后的例子都是基于“Ubuntu 14.04.4 Desktop (64-bit)”。
- 到 <https://unetbootin.github.io/> 下载刻录安装 U 盘的工具 unetbootin。
- 将一个 U 盘格式化 fat32 格式，然后打开 unetbootin，选择 Ubuntu 的 ISO 镜像文件，刻录到 U 盘。

● 安装系统

- 将计算机设置成 U 盘启动，通过安装 U 盘开始安装。
- 在分区的步骤选择“something else”手动分区，可以将 swap 分区设为 2G，其他全部设为 / 分区，选 ext3 格式和主分区。
- 根据提示设置好时区、键盘（美式键盘）、账号信息等。安装时建议选择中文版，这样会安装中文语言支持，装好后再将系统设置成英文。

常用设置

● 软件源

- 打开 Ubuntu Software Center (可以通过开始菜单搜索), 选择 “Edit -> Software Sources”, 在 “Download from” 中选择一个国内的服务器, 作为软件源服务器。
- 打开命令行 (可以通过开始菜单搜索 Terminal), 输入以下命令进行更新:

```
sudo apt-get update
```

● 安装必备软件

- 使用命令行安装中文输入法和 vim (文本编辑器):

```
sudo apt-get install ibus-googlepinyin  
sudo apt-get install vim
```

- 注意, 命令中的 **sudo** 表示允许普通用户使用超级权限 (root 用户), 有时候会提示输入密码。

基础操作

● 文件目录

- 登录后默认的工作路径是 `/home` 下以登录用户为名的文件夹，该文件夹也可以用“`~`”代替。
- 命令 `cd` 表示进入到某个路径，`ls` 表示列出当前路径下的所有文件和文件夹名，例如：

```
cd /home  
cd ~  
ls
```

● 文件操作

- 创建文件夹：`mkdir -m 777 test1`，其中 777 表示所有者有读取、写入、执行的权限。
- 删除文件：`rm file1`。
- 删除文件夹：`rm -rf test1`。

文件夹

基础操作

● 打开文本文件

- 使用 Ubuntu 桌面版自带的 gedit, 将会通过图形界面工具打开文本:

```
sudo gedit test1.txt
```

- 使用 vi 命令在命令行进行操作:

```
sudo vi test1.txt
```

● vi 操作

- Linux 系统中都会默认装有 vi, 如果额外安装了 vim, 将会高亮显示其中的文本。
- 使用 vi 命令打开文档后, 默认处于只读状态, 键入“i”会进入插入模式, 此时可以通过移动光标对文本进行修改。
- 按 Esc 键可以从插入模式进入命令行模式。
- 在命令行模式下键入“:”进入底行模式, 此时可以输入命令, 其中“q!”表示不存盘强制退出 vi, “wq”表示存盘退出。

目 录

- 1 相关环境介绍
 - 云计算简介
 - 容器与虚拟化
 - Linux 基础
 - Docker 简介
- 2 Hadoop 和 MapReduce
- 3 Spark 简介

基本信息

● 容器和镜像

- 容器 (Container) 可以认为是一个虚拟的操作系统环境，在逻辑上可以当作是一台独立运行的计算机。
- 将某个容器的某个状态保存起来，就是镜像 (Image)，通过镜像文件可以进入到相应的容器。

● 查看基本信息

- 查看所有容器、镜像数量以及 Docker 的基本配置：

```
sudo docker info
```

- 查看容器信息（不加参数 **-a** 则只输出当前运行的容器）：

```
sudo docker ps -a
```

- 查看镜像信息及显示某镜像历史：

```
sudo docker images  
sudo docker history 2cc48731dfff
```

创建容器

● 创建一个 Ubuntu 容器

- 在 Ubuntu 命令行执行如下命令：

```
sudo docker run -i -t ubuntu:14.04 /bin/bash
```

- 基于基础的 Ubuntu 镜像（标签为 14.04）创建，如果是第一次创建，会自动到 Docker Hub Registry 中下载。
 - 会随机产生容器的 ID（例如 96f3f2a403f9）和 NAMES（例如 hungry_turing）。
 - 新建容器后自动执行后面的代码 /bin/bash，即进入 Ubuntu 的 Bash shell 界面，exit 命令可以退出容器并回到宿主机命令行。
 - 再次运行该命令会新建另一个容器。
- ## ● docker run 的参数
- -i 表示开启 STDIN，用来保证标准输入。
 - -t 表示为容器分配一个伪 tty 终端，用来提供交互式 shell。
 - -d 表示将容器放在后台运行，否则结束操作即退出容器。
 - --name xxx 表示将容器命名为 xxx，省略则自动命名。

容器和镜像

容器操作

- 可以使用容器名或者 ID 来启动和进入某个容器：

```
sudo docker start test1  
sudo docker attach test1
```

镜像操作

- 命令 `pull` 可以下载官方的镜像，`search` 可以查找镜像。

```
sudo docker pull ubuntu  
sudo docker search xxx
```

删除容器和镜像

- 假设容器 ID 为 `5ef7cb5c3f10`，镜像 ID 为 `2cc48731dfff`，可以分别删除：

```
sudo docker rm 5ef7cb5c3f10  
sudo docker rmi 2cc48731dfff
```

交互式容器和守护式容器

● 交互式容器

- 创建一个名为 test1 的交互式容器：

```
sudo docker run --name test1 -i -t ubuntu:14.04 /bin/bash  
sudo docker attach test1
```

- 对于交互式容器，进入后用 `exit` 命令退出后即停止该容器。
- 使用 `attach` 命令可以再次进入该容器。

● 守护式容器

- 创建一个名为 test2 的守护式容器：

```
sudo docker run --name test2 -d ubuntu:14.04 /bin/sh  
-c "while true; do echo hello world; sleep 1; done"  
sudo docker attach test2  
sudo docker start test2  
sudo docker logs test2  
sudo docker stop test2
```

- 对于守护式容器，开启后会一直在后台运行。
- 进入该容器后使用 `Ctrl+C` 可以强制退出，此时会关闭该容器，要通过 `start` 命令再次开启。

镜像和注册库

- 注册库 (Registry)

- 注册库是 Docker 管理镜像的方式。
- Docker 公司提供了公共注册库平台：Docker Hub。
- 用户也可以建私有的注册库。

- Docker Hub

- 主页：<https://hub.docker.com/>。
- 用户可以注册自己的账号，例如：`jianl`。
- 在命令行可以使用 `docker login` 命令登录，将会直接把 Docker Hub 当作注册库。

- Docker Hub 结构

- 注册库中可以包含多个仓库 (Repository)，每个仓库可以包含多个标签 (Tag)。
- Docker Hub 包含顶层 (top-level) 仓库和用户 (user) 仓库。
- 顶层仓库由 Docker 公司维护，一个镜像标签的格式为 “`ubuntu:14.04`”。用户仓库由注册用户自己维护，一个镜像标签的格式为 “`jianl/image1:latest`”。注意，如果创建镜像时不指定标签，将会被自动命名为 `latest`。

创建镜像

● 使用 `docker commit`

- 进入一个容器，进行相关操作，通常是安装软件或者进行一些配置。
- 假设该容器 ID 是 `96f3f2a403f9`，我们想把此容器此时的状态建立一个镜像，名为 `jianl/image1:v1`，则运行：

```
sudo docker commit 96f3f2a403f9 jianl/image1:v1
```

- 此时会新建一个本地镜像，如果需要将该镜像推送到 Docker Hub（这是镜像中指定用户名的原因，用户名需要与 Docker Hub 账号一致，否则提示无权限），运行如下代码：

```
sudo docker push jianl/image1:v1
```

● 使用 `Dockerfile`

- 这是 Docker 中创建镜像的推荐方式，在一个名为 `Dockerfile` 的文件中写下操作的代码，然后使用 `docker build` 命令构建一个镜像。
- 该方式容易共享开发过程，而且可以指定更多的参数。

目 录

- ① 相关环境介绍
- ② Hadoop 和 MapReduce
 - 简介和安装
 - MapReduce 并行
 - RHadoop 简介
- ③ Spark 简介

目 录

- ① 相关环境介绍
- ② Hadoop 和 MapReduce
 - 简介和安装
 - MapReduce 并行
 - RHadoop 简介
- ③ Spark 简介

Hadoop 平台

● 简介

- Apache Hadoop 是一款支持数据密集型分布式应用并以 Apache 2.0 许可协议发布的开源软件框架。它支持在商品硬件构建的大型集群上运行的应用程序。Hadoop 是根据 Google 公司发表的 MapReduce 和 Google 文件系统的论文自行实现而成。

● 主要子项目

- Hadoop Common: 在 0.20 及以前的版本中, 包含 HDFS、MapReduce 和其他项目公共内容, 从 0.21 开始 HDFS 和 MapReduce 被分离为独立的子项目, 其余内容为 Hadoop Common。
- HDFS: Hadoop 分布式文件系统 (Distributed File System) — HDFS (Hadoop Distributed File System)。
- MapReduce: 并行计算框架, 0.20 前使用 `org.apache.hadoop.mapred` 旧接口, 0.20 版本开始引入 `org.apache.hadoop.mapreduce` 的新 API。

Docker 与 Hadoop

● 操作步骤

- ① 使用 Ubuntu 镜像新建一个容器。
- ② 在容器中安装好 Hadoop 的运行环境。
- ③ 将该容器存成一个名为 `jian1/hadoop:v0` 的镜像。
- ④ 使用该镜像生成新的容器，构建 Hadoop 集群。在本例子中，我们建立一个 Hadoop 集群，包含 1 个主节点 (Master) 和 2 个从节点 (Slave)。

● 新建容器

- 创建并进入容器：

```
sudo docker run -i -t ubuntu:14.04 /bin/bash
```

- 在该容器中安装 Java：

```
sudo apt-get install software-properties-common
sudo apt-get install python-software-properties
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get update
sudo apt-get install oracle-java7-installer
```

安装 Hadoop

● 下载后解压缩

- 使用命令 `wget` 下载 Hadoop 源码，直接解压即安装：

```
cd ~
mkdir soft/apache/hadoop
cd soft/apache/hadoop
wget http://mirrors.sonic.net/apache/hadoop/common/
    hadoop-2.6.0/hadoop-2.6.0.tar.gz
tar xvzf hadoop-2.6.0.tar.gz
```

● 配置环境变量

- 使用命令 `vi ~/.bashrc` 打开文件，在末尾添加如下信息：

```
export JAVA_HOME=/usr/lib/jvm/java-7-oracle
export HADOOP_HOME=/root/soft/apache/hadoop/
    hadoop-2.6.0
export HADOOP_CONFIG_HOME=$HADOOP_HOME/etc/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
```

配置 SSH

● 安装配置 SSH

- 安装 SSH，并生成访问密钥：

```
sudo apt-get install ssh  
cd ~/  
ssh-keygen -t rsa -P '' -f ~/.ssh/id_dsa  
cd .ssh  
cat id_dsa.pub >> authorized_keys
```

● 自动运行 SSH 服务

- 添加 `/var/run/sshd` 目录（否则启动 SSH 时会报错）：

```
sudo mkdir /var/run/sshd
```

- 在 `~/.bashrc` 文件的最后一行添加自动运行命令：

```
/usr/sbin/sshd
```

配置 Hadoop

- 创建相关目录

- 临时目录:

```
cd $HADOOP_HOME/  
mkdir tmp  
cd tmp/  
pwd /root/soft/apache/hadoop/hadoop-2.6.0/tmp
```

- NameNode 的存放目录:

```
cd $HADOOP_HOME/  
mkdir namenode  
cd namenode/  
pwd /root/soft/apache/hadoop/hadoop-2.6.0/namenode
```

- DataNode 的存放目录:

```
cd $HADOOP_HOME/  
mkdir datanode  
cd datanode/  
pwd /root/soft/apache/hadoop/hadoop-2.6.0/datanode
```

配置 Hadoop

- core-site.xml 配置

- 打开文本:

```
cd $HADOOP_CONFIG_HOME/  
sudo vi core-site.xml
```

- 在文本中修改配置

- 在“configuration”中添加如下信息:

```
<property>  
<name>hadoop.tmp.dir</name>  
<value>/root/soft/apache/hadoop/hadoop-  
2.6.0/tmp</value>  
<description>description</description>  
</property>  
<property>  
<name>fs.default.name</name>  
<value>hdfs://master:9000</value>  
<final>true</final>  
<description>description</description>  
</property>
```

配置 Hadoop

- **hdfs-site.xml 配置**

- Vi 打开 **hdfs-site.xml** 文件后在“configuration”中添加：

```
<property>
<name>dfs.replication</name>
<value>2</value>
<final>true</final>
<description>description</description>
</property>
<property>
<name>dfs.namenode.name.dir</name>
<value>/root/soft/apache/hadoop/hadoop-
2.6.0/namenode</value>
<final>true</final>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>/root/soft/apache/hadoop/hadoop-
2.6.0/datanode</value>
<final>true</final>
</property>
```

配置 Hadoop

- **mapred-site.xml 配置**

- 将 `mapred-site.xml.template` 模板文件复制成 `mapred-site.xml`, 然后打开文本:

```
cd $HADOOP_CONFIG_HOME/  
cp mapred-site.xml.template mapred-site.xml  
sudo vi core-site.xml
```

- **在文本中修改配置**

- 在“configuration”中添加如下信息:

```
<property>  
<name>mapred.job.tracker</name>  
<value>master:9001</value>  
<description>description</description>  
</property>
```

- 其中的端口号可以自行指定。

配置 Hadoop

- **hadoop-env.sh 配置**

- 打开文本:

```
sudo vi hadoop-env.sh
```

- 在打开的文本中修改 **JAVA_HOME** 环境变量:

```
export JAVA_HOME=/usr/lib/jvm/java-7-oracle
```

- **格式化 namenode**

- 第一次启动时要先进行格式化, 运行如下命令:

```
hadoop namenode -format
```

- **至此, Hadoop 容器安装完成, 将其存成镜像**

- 退出容器, 运行以下命令创建镜像:

```
sudo docker commit -m "hadoop install" 5203ef61f0f2  
jianl/hadoop:v0
```

启动 Hadoop

● 启动容器

- 分别新建并启动 master, slave1, slave2 这三个容器。

```
docker run -ti --name master -h master jianl/hadoop:v0
docker run -ti --name slave1 -h slave1 jianl/hadoop:v0
docker run -ti --name slave2 -h slave2 jianl/hadoop:v0
```

● 配置 hosts

- 每次启动容器后 IP 地址或自动分配，最简单的办法是在每个节点中配置 `/etc/hosts` 文件，添加 3 个节点的 IP 地址，例如：

```
172.17.0.2    master
172.17.0.3    slave1
172.17.0.4    slave2
```

- 在 master 节点的 `$HADOOP_CONFIG_HOME/slaves` 文件中添加以下内容：

```
slave1
slave2
```

启动 Hadoop

● 启动 Hadoop

- 启动前可以先测试 SSH 连接：

```
ssh slave1
```

- 在 master 节点中执行以下命令开始启动：

```
start-all.sh
```

● 测试安装

- 在每个节点运行 `jps` 命令。
- 在 master 节点运行以下命令，查看 DataNode 是否可以正常启动：

```
hdfs dfsadmin -report
```

- 通过 Web 页面查看 DataNode 和 NameNode 的状态：
<http://172.17.0.2:50070/>。

目 录

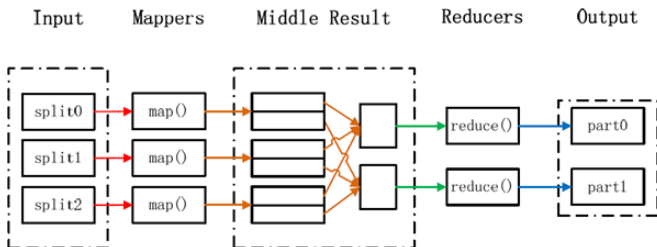
- ① 相关环境介绍
- ② Hadoop 和 MapReduce
 - 简介和安装
 - MapReduce 并行
 - RHadoop 简介
- ③ Spark 简介

MapReduce 简介

● 什么是 MapReduce

- Google 提出的一个软件架构，用于大规模数据集的并行运算，该专利于 2010 年 1 月获批。
- Map（映射）对列表的每一个元素进行指定的操作，得到一组中间值。
- Reduce（归约）把这些中间值通过一定的函数进行处理来获取最终的结果。

● MapReduce 流程示例



mapper.R 文件

● 代码说明

- 新建一个名为 `mapper.R` 的文本文件，将代码拷入。
- 该脚本实现“Map”的操作。

● 代码示例

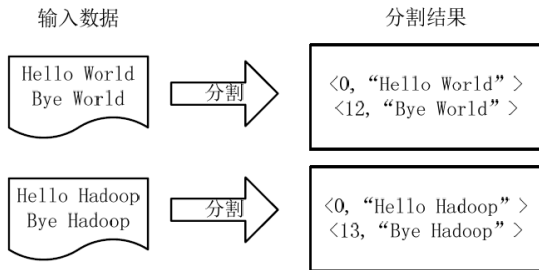
```
trimWhiteSpace <- function(line) {  
  gsub("(^ +)|( +$)", "", line)  
}  
splitIntoWords <- function(line) {  
  unlist(strsplit(line, "[[:space:]]+"))  
}  
  
con <- file("stdin", open = "r")  
while (length(line <- readLines(con, n = 1,  
  warn = FALSE)) > 0) {  
  line <- trimWhiteSpace(line)  
  words <- splitIntoWords(line)  
  for (w in words) cat(w, "\t1\n", sep="")  
}  
close(con)
```

Map 流程

● 拆分文件

- 将文件拆分成 splits, 由于测试用的文件较小, 所以每个文件为一个 split。
- 将文件按行分割形成 $\langle \text{key}, \text{value} \rangle$ 对, 由 MapReduce 框架自动完成。
- 其中偏移量 (即 key 值) 包括了回车所占的字符数 (Windows 和 Linux 环境会不同)。

● 流程示例

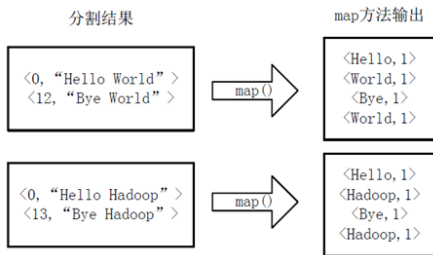


Map 流程

● Map 过程

- 将分割好的 `<key,value>` 对交给用户定义的 `map` 方法进行处理，生成新的 `<key,value>` 对。
- `mapper.R` 主要操作的就是这个过程。

● 流程示例

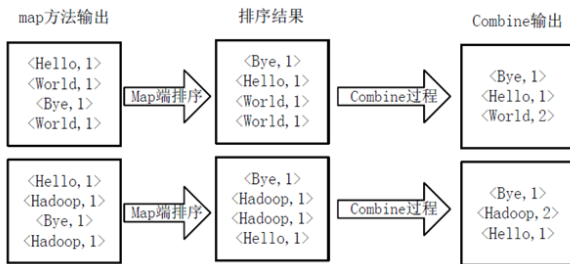


Map 流程

Combine 过程

- 得到 map 方法输出的 $\langle \text{key}, \text{value} \rangle$ 对后, Mapper 会将它们按照 key 值进行排序。
- 然后执行 Combine 过程, 将 key 至相同 value 值累加, 得到 Mapper 的最终输出结果。

流程示例



reducer.R

● 代码说明

- 新建一个名为 `reducer.R` 的文本文件，将代码拷入。
- 该脚本实现“Reduce”的操作。
- 每个 Reduce 任务会读入一个 Map 处理好的键值对，并将结果累加，最后汇总，得到最后的输出结果。

● 代码示例

```
trimWhiteSpace <- function(line) {  
  gsub("(^ +)|( +$)", "", line)  
}  
  
splitLine <- function(line) {  
  val <- unlist(strsplit(line, "\\t"))  
  list(word = val[1], count = as.integer(val[2]))  
}  
  
env <- new.env(hash = TRUE)
```

reducer.R

● 代码示例

```
con <- file("stdin", open = "r")
while (length(line <- readLines(con, n = 1,
  warn = FALSE)) > 0) {
  line <- trimWhiteSpace(line)
  split <- splitLine(line)
  word <- split$word
  count <- split$count
  if (exists(word, envir = env, inherits = FALSE)) {
    oldcount <- get(word, envir = env)
    assign(word, oldcount + count, envir = env)
  } else {
    assign(word, count, envir = env)
  }
}
close(con)

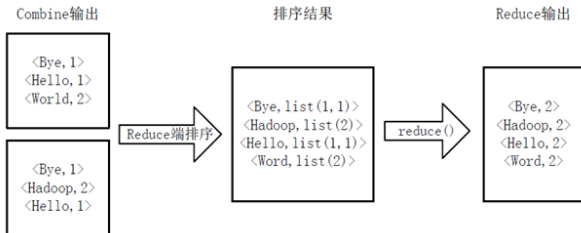
for (w in ls(env, all = TRUE)) {
  cat(w, "\t", get(w, envir = env), "\n", sep = "")
}
```

Ruduce 流程

● Reduce 过程

- Reducer 首先对从 Mapper 接收的数据进行排序。
- 再交由用户自定义的 reduce 方法进行处理，得到新的 $\langle \text{key}, \text{value} \rangle$ 对，并作为 WordCount 的输出结果。

● 流程示例



运行

● 复制文件

- 首先将本地文件复制到 DFS 文件系统。
- 以系统中自带的 README.txt 文件为例。

```
cd $HADOOP_HOME
bin/hadoop dfs -copyFromLocal '/root/soft/apache/
hadoop/hadoop-2.6.0/README.txt' /readme
bin/hadoop dfs -ls /
```

● 执行任务

- 通过 Streaming 框架处理 mapper.R 和 reducer.R 脚本，自动分配任务并运行。

```
bin/hadoop jar /root/soft/apache/hadoop/hadoop-2.6.0/
contrib/streaming/hadoop-streaming-1.0.4.jar \
-file /home/jl/mapper.R -mapper /home/jl/mapper.R \
-file /home/jl/reducer.R -reducer /home/jl/reducer.R \
-input /readme -output /RCount
```

- 将结果从 HDF 中复制出来：

```
bin/hadoop dfs -get /RCount/part-00000 /home/jl/out.txt
```

目 录

- ① 相关环境介绍
- ② Hadoop 和 MapReduce
 - 简介和安装
 - MapReduce 并行
 - RHadoop 简介
- ③ Spark 简介

什么是 RHadoop

- Streaming 和 RHadoop

没太大必要用
Rhadoop, 用Spark.

- Hadoop 基于 Java 编写, 但是提供了 Hadoop Streaming 框架, 让任何语言编写的 map/reduce 程序能够在 hadoop 集群上运行。
- map/reduce 程序只要遵循从标准输入 stdin 中读入, 写出到标准输出 stdout 即可, 不限语言, 当然也支持 R 脚本。
- RHadoop 由 Revolution 公司 (2015 年被微软收购) 开发和维护, 提供了一套能直接在 R 环境中操作 Hadoop 的接口包。

- RHadoop 的组成部分

- rhdfs 包用来连接 HDFS 的文件系统。
- rhbase 包用来连接 HBASE 分布式数据库。
- plyrmr 提供了一套类似 plyr 和 reshape2 包的数据处理方式, 基于 MapReduce 实现。
- rmr2 基于 MapReduce 进行统计分析。
- ravro 包用来读写 avro 形式的数据。

RHadoop 与 Docker 安装

● 安装 R

- 基于 `jianl/hadoop:v0` 新建一个容器，完成后保存成镜像。
- 在容器中安装 R，并运行：

```
R CMD javareconf
```

- 安装 `rJava`、`reshape2`、`Rcpp`、`iterators`、`iterators`、`digest`、`RJSONIO`、`functional` 等包。
- 修改 `~/.bashrc`，加入：

```
export HADOOP_CMD=/root/soft/apache/hadoop/  
hadoop-2.6.0/bin/hadoop  
export HADOOP_STREAMING=/root/soft/apache/hadoop/  
hadoop-2.6.0/share/hadoop/tools/lib/  
hadoop-streaming-2.6.0.jar
```

● 安装 RHadoop 相关包

- 到 <https://github.com/RevolutionAnalytics/RHadoop/wiki/Downloads> 下载 `rhdfs` 和 `rmr` 源码包。
- 使用 `R CMD INSTALL XXX.tar.gz` 命令进行安装。

RHadoop 示例

● 代码说明

- 先使用 `to.dfs` 函数将数据复制到 DFS 系统，然后通过 `mapreduce` 函数来实现 MapReduce 框架。

● 代码示例

```
library(rhdfs)
library(rmr2)
wordcount <- function(input, output = NULL){
  mapreduce(input = input, output = output,
    map = function(k, v){
      v2=unlist(strsplit(x=v,split="[:,space:]]+"))
      lapply(v2, function(w){keyval(w,1)})
    },
    reduce=function(k,v) keyval(k,sum(unlist(v)))
  )
}
lines <- c("hello world", "bye world")
lines_dfs <- to.dfs(lines)
wc1 <- wordcount(input=lines_dfs)
do.call(rbind, from.dfs(wc1))
```

目 录

- 1 相关环境介绍
- 2 Hadoop 和 MapReduce
- 3 Spark 简介
 - 简介和安装
 - Spark 和 R

目 录

- 1 相关环境介绍
- 2 Hadoop 和 MapReduce
- 3 Spark 简介
 - 简介和安装
 - Spark 和 R

什么是 Spark

● Spark 的兴起

- Hadoop 的 MapReduce 虽然方便，但是每个节点的计算都需要读写 HDFS 文件，如果算法需要频繁地存取数据，就会很低效。针对这个问题，加州大学伯克利分校 AMP 实验室于 2012 年开始开发 Spark（2009 年就有雏形了），这是一种类似于 MapReduce 的框架。
- Spark 拥有 MapReduce 所具有的优点。但是中间输出结果可以保存在内存中，从而不再需要读写 HDFS，因此 Spark 能更好地适用于数据挖掘与机器学习等需要迭代的 MapReduce 的算法。2014 年，Spark 成为 Apache 的顶级项目。

● Spark 的应用

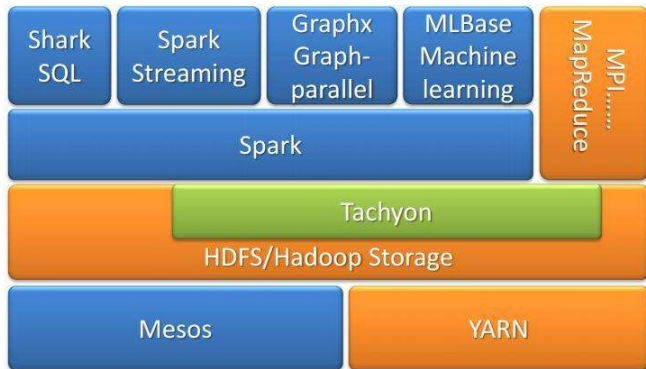
- Spark 是 MapReduce 的替代方案，同时可以兼容 HDFS、Hive 等分布式存储系统，能运行在 Hadoop 集群管理 Yarn 上，可融入 Hadoop 生态，在日常使用中通常与 Hadoop 互为补充，通常和 Hadoop 部署在同一个集群。
- Spark 基于 Scala 开发，支持 Java、Python 和 Spark SQL，也有 R 语言扩展包。

Spark 生态圈

● Spark 相关环境

- YARN 是 Hadoop 的资源管理器。Mesos 是 Apache 下的开源分布式资源管理框架，它被称为是分布式系统的内核。Tachyon 是 Apache 下一个高性能分布式存储系统。

● Spark 生态结构



Windows 下安装 Spark

● 下载和安装

- 进入 <http://spark.apache.org/downloads.html>, 选择“Pre-built”版本。

Download Apache Spark™

1. Choose a Spark release: 2.4.0 (Nov 02 2018) ▼

2. Choose a package type: Pre-built for Apache Hadoop 2.7 and later ▼

3. Download Spark: [spark-2.4.0-bin-hadoop2.7.tgz](#)

4. Verify this release using the [2.4.0 signatures and checksums](#) and [project release KEYS](#).

Note: Starting version 2.0, Spark is built with Scala 2.11 by default. Scala 2.10 users should download the Spark source package and build with Scala 2.10 support.

- 解压到某个文件夹即完成安装，例如
D:\Software\spark-2.4.0-bin-hadoop2.7。

● 设置

- 将 D:\Software\spark-2.4.0-bin-hadoop2.7\bin 加入 PATH 环境变量。
- 新建环境变量 SPARK_HOME，将其值设置为 D:\Software\spark-2.4.0-bin-hadoop2.7。

Linux 下安装 Spark

● 安装 Scala

- Spark 依赖 Scala 环境，要先安装：

```
wget https://downloads.lightbend.com/scala/2.11.7/  
scala-2.11.7.tgz  
tar -zxvf scala-2.11.7.tgz
```

- 安装后加入环境变量：

```
export SCALA_HOME=/usr/scala-2.11.7
```

● 安装 Spark

- 下载和安装：

```
wget https://d3kbcqa49mib13.cloudfront.net/  
spark-2.2.0-bin-hadoop2.7.tgz  
tar -zxvf spark-2.2.0-bin-hadoop2.7.tgz
```

- 加入环境变量：

```
export SPARK_HOME=/usr/local/spark-2.2.0
```

- 详见官网：<http://spark.apache.org/docs/latest/>。

目 录

- 1 相关环境介绍
- 2 Hadoop 和 MapReduce
- 3 Spark 简介
 - 简介和安装
 - Spark 和 R

SparkR 简介

● 简介

- SparkR 是一个 Spark 的官方接口 R 包，详细文档请参考 <http://spark.apache.org/docs/latest/sparkr.html>。
- SparkR 提供了 Spark 中弹性分布式数据集 (RDD) 的 API，用户可以在集群上通过 R shell 交互性的运行 job。在 R 中直接处理 SparkR DataFrames 即可。
- 使用 sparklyr 包，可以在 Spark 上实现 dplyr 包的数据操作方式。
- 调用 Spark 内置的 MLlib 库，可以实现很多机器学习的方法。

● 安装 SparkR

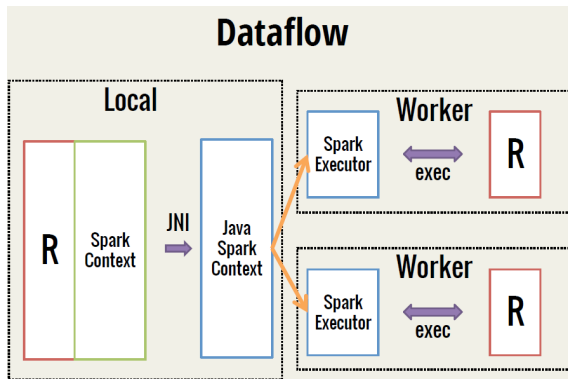
- Spark 默认安装了 R 环境，设置好环境变量后，在命令行运行 sparkR 即能启动支持 Spark 的 R。
- 将 D:\Software\spark-2.4.0-bin-hadoop2.7\R\lib 里的 SparkR 文件复制到 R 的 library 路径，即在 R 中完成 Spark 的接口包的安装。

SparkR 的运行机制

● 运行机制简介

- 通过 R 代码和 Spark 集群进行交互，每个节点可以调用 R 函数，可以通过 RDD 的机制实现并行计算，也可以在 R 中直接调用内置的并行函数。

● 运行机制示例



使用示例

● 简介

- 在 R 中加载 `SparkR` 包，然后使用 `sparkR.session` 函数，连接 Spark 集群，默认在本机建立集群。
- 可以使用 `as.DataFrame` 的函数将 R 中的数据对象转化成 `SparkDataFrame` 类型，也可以从文件或者 HIVE 中取数，通过该对象在 R 中进行操作。

● 代码示例

```
library(SparkR)
sparkR.session()
df <- as.DataFrame(faithful)
head(select(df, df$eruptions))
head(summarize(groupBy(df, df$waiting),
  count = n(df$waiting)))
head(agg(cube(df, "cyl", "disp", "gear"),
  avg(df$mpg)))
head(agg(rollup(df, "cyl", "disp", "gear"),
  avg(df$mpg)))
```

Thank you!

李舰 Email: jian.li@188.com