基于 Hadoop MapReduce 模型的应用研究

谢桂兰，罗省贤

（成都理工大学 信息工程学院，四川 成都 610059）

摘 要：MapReduce 是一种简化并行计算的分布式编程模型，是 Google 的一项重要技术，通常被用于数据密集型的分布式并行计算。探讨了来自 Apache 开源的分布式计算平台 Hadoop 的核心设计 MapReduce 编程模型，并通过算法实验分析和研究了 MapReduce 模型的工作方式和应用方法。

关键词：分布式并行计算；Hadoop；编程模型；MapReduce

中图分类号：TP391 **文献标识码**：A

Study on application of MapReduce model based on Hadoop

XIE Gui Lan, LUO Sheng Xian

(College of Information Engineering, Chengdu University of Technology, Chengdu 610059, China)

Abstract: MapReduce is a simplified programming model of distributed parallel computing. It is an important technology of Google, and is commonly used for data-intensive distributed parallel computing. This paper discusses MapReduce programming model, which is the core design of the open-source distributed computing platform named Hadoop from Apache organization, and analyzes and studies the working methods and application of MapReduce model through the algorithm experiments.

Key words: distributed parallel computing; Hadoop; programming model; MapReduce

云计算（Cloud Computing）是一种新近提出的技术概念，Google 是云计算的最早倡导者。“云计算”这一名词的出现，立刻在产业界和学术界引起了轰动。IBM、Google、Sun、MS、百度、Yahoo、亚马逊甚至瑞星、奇虎这些公司都开始进行“云计算”的部署工作。而 Hadoop 分布式计算平台是云计算的一个开源实现。互联网行业巨擘 Google 引以自豪的三大核心技术 GFS、Mapreduce 和 Bigtable 使其在激烈的行业竞争中占尽先机。Hadoop 正是借鉴了许多 GFS 和 MapReduce 的设计思想发展而来的，从某种意义上也可以将 Hadoop 理解为 GFS 和 Google MapReduce 的开源实现。

Hadoop 是一个可以更容易支持开发和并行处理大规模数据的分布式计算平台，其主要优点是：可扩展（Scalable）、低成本（Economical）、高效（Efficient）、可靠（Reliable）。它完全使用 Java 开发的开源软件，因而可以广泛运行在多种软硬件平台上。本文分析和研究了基于 Hadoop 的 MapReduce 编程模型的思想、计算流程和应用 MapReduce 模型实现分布式并行算法的方法，并设计了一套分布式并行随机数发生器算法验证用 MapReduce 解决并行计算问题的适用性和计算效率。

1 Hadoop 框架的工作机制

Hadoop 为应用程序透明地提供了一组稳定可靠的接口。Hadoop 框架的主要组成部分是 Hadoop 分布式文件系统（HDFS）和 MapReduce 的实现。HDFS 采用 Master/Slave 架构，一个 HDFS 集群由一个 NameNode 节点和一组 DataNode 节点组成。NameNode 是一个中心服务器，负责管理文件系统的名字空间（NameSpace）以及客户端对文件的访问。在集群系统中，一般在一个节点上运行一个 DataNode，负责管理它所在节点上的数据存储，并负责处理文件系统客户端的读写请求，在 NameNode 的统一调度下进行数据块的创建、删除和复制。Hadoop 还实现了 Google 的 MapReduce 分布式计算模型，MapReduce 把应用程序的总任务分割成许多子任务，每个子任务可以在任何集群节点（DataNode 节点，通常也作为计算节点）上并行处理。HDFS 创建了多份数据块（data blocks）的副本（Replicas），以保证各个子任务节点计算的可靠性（Reliability）。由于采用了分布式文件系统和 MapReduce 模型，因此 Hadoop 框架具有高容错性及对数据读写的高吞吐率，能自动处理失败节点。图 1 是 Hadoop 集群系统架构的示意图。

如图 1 所示，在 Hadoop 集群架构中，Client 是需要获取分布式文件系统文件的应用程序，并且会有一台 Master 主要负责 NameNode 的工作以及 JobTracker 的工作。JobTracker 在 Hadoop 集群中负责控制 MapReduce 应

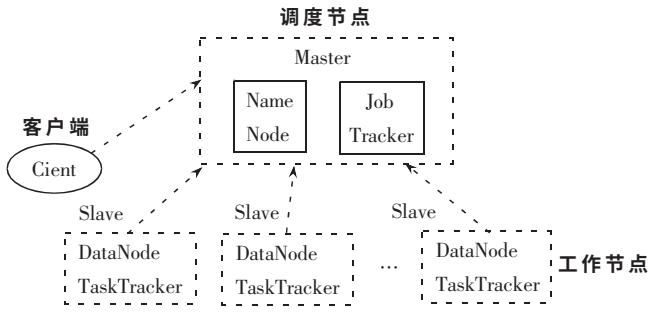

图 1 Hadoop 集群系统架构示意图

用程序的启动、跟踪和调度各个 Slave 任务的执行。每一台 Slave 通常具有 DataNode 的功能并负责 TaskTracker 的工作。TaskTracker 根据应用程序的要求负责管理本地数据的处理和结果数据的收集任务，并将状态和完成信息报告给 JobTracker。

2 MapReduce 分布式并行计算编程模型

MapReduce(映射-归并算法)模型是由 Google 公司的 Jeffrey Dean 和 Sanjay Ghemawat 提出的高阶并行函数的抽象模式，据报道每天在 Google 的集群上有上千个 MapReduce job 在执行。MapReduce 借用了函数式编程的思想，通过把海量数据集的常见操作抽象为 Map 和 Reduce 两种集合操作，大大简化了程序员编写分布式并行计算程序的难度。

在 MapReduce 计算模型中，有两个关键过程：映射过程 Map 和聚集过程 Reduce。因此需要用户提供两个关键函数，映射(Map)函数和聚集(Reduce)函数，这两个函数对一组输入的键值对(key/value)进行计算，得出另一组输出键值对[1]，即有：

$$\text{Map}:(k1,v1)\rightarrow \text{list}(k2,v2)$$

$$\text{Reduce}:(k2,\ \text{list}(v2))\rightarrow \text{list}(k3,v3)$$

用户定义的映射函数 Map 接收一组输入键值对 $(k1,v1)$，经过处理产生一组中间的 $(k2,v2)$ 键值对。MapReduce 函数库聚合所有相同的中间键 $k2$ 的相应值，产生关于 $k2$ 键的值集合 $\text{list}(v2)$，并且发送给由用户提供的归并函数 Reduce，Reduce 再进一步处理、合并该中间键的值集合，最后形成一个相对较小的键值对集合 $\text{list}(k3,v3)$。

运行于 Hadoop 平台下的 MapReduce 应用程序由一个 Mapper 类、Reducer 类和一个创建 JobConf 的驱动函数组成，需要时还可以包括一个 Combiner 类，这个类实际上也是 Reducer 的一种实现。图 2 是 Hadoop 的 MapReduce 任务基本计算流程以及关键函数间的关系。

从图中可以看出 MapReduce 任务的基本计算流程如下：首先对输入数据进行划分，如图 2 中 split0~split n 是划分后的数据块，这些数据块由 RecordReader 处理生成<k,v>键值对，然后进入 Map 操作。

Map 操作产生的中间结果被 Partitioner 类以指定的方式区分地写到输出文件里。可以根据需要为 Mapper 指定 Combiner 类对 Mapper 输出的<k,v>中间键值对进行合并，然后再输出到文件中。

Map 任务结束后进入 Reduce 过程。每个 Reduce 任务包括组合(shuffle)、排序(sort)和聚集数据(reduce)三个阶段。MapReduce 框架根据中间结果中的键 key，将多个 Mapper 产生的同一个 key 的中间结果通过 HTTP 协议传给处理这个 key 的 Reducer 类。在组合和排序阶段，将来自不同 Mapper 具有相同 key 值的<key,value>对合并到一起，最后将通过组合和排序后得到的<key, (list of values)>送到 Reducer 类的 reduce 方法中处理，将得到的结果写入由 Hadoop 的分布式文件系统 HDFS 管理的输出文件中。

3 并行随机数发生器算法的 MapReduce 实现

3.1 算法实现流程

大量的应用问题(如产生白噪声、游戏程序、随机算法等)需要产生随机数。一种产生(0,1)区间上伪随机数的串行算法如下：

$$x_i=(157\times x_{i-1})\ \bmod\ 323\,63;$$

$$y_i=(157\times x_{i-1})\ \bmod\ 317\,27;$$

$$z_i=(157\times x_{i-1})\ \bmod\ 316\,57;$$

$$s=(x_i+y_i+z_{i-3})\ \bmod\ 323\,62$$

$$r=(s+1)/323\,63$$

式中 r 为最终计算出的随机数，初值取为 $x_0=y_0=z_0=1$。为了快速同时产生(0,1)区间上均匀分布的大量随机数，本文基于上述算法利用 Hadoop 的 MapReduce 模型实现并行计算随机数，基本实现流程如图 3 所示。

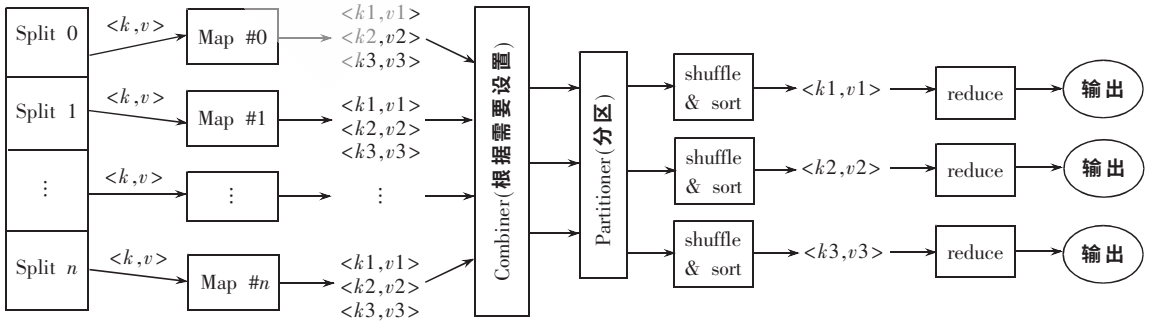

图 2 MapReduce 任务计算流程示意图

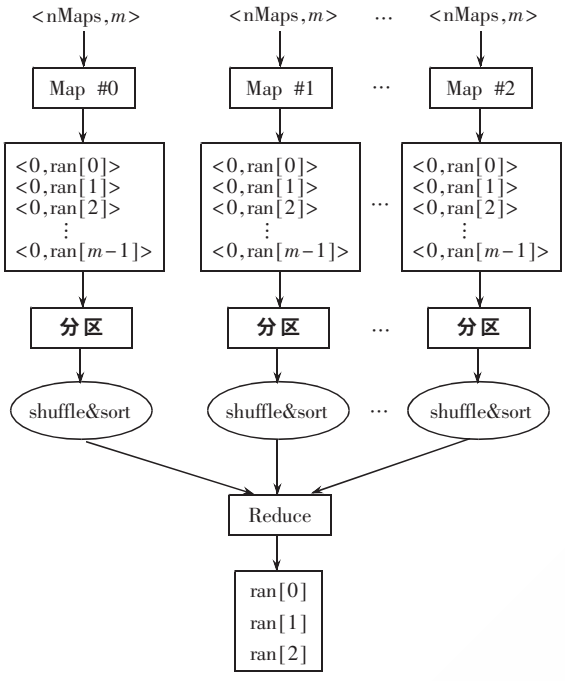

图 3 基于 MapReduce 模型并行计算随机数的实现流程

计算流程基本步骤如下：

(1) 由 MapReduce 框架为每个映射任务(Map task)分配计算负载，即确定每个映射任务要生成的随机数个数(m)，m 由下列公式计算得出：

m=nRandom(总随机数个数)/nMaps(映射任务数)

并将该任务信息分发到集群中的各个计算节点；

(2) 集群中的各计算节点执行 Map 任务，根据负载分配计算得出 m 个随机数并存入一个数组中，产生一组中间结果；

(3) MapReduce 框架进行分区操作，该过程由 Partitioner 类使用 Hash 函数按 key(或者一个 key 子集)进行分区操作，分区的数目等同于一个作业的 Reduce 任务的数目。即由 Partitioner 控制将中间过程的 key 发送给 m 个 Reduce 任务中的哪一个来进行 Reduce 操作[3]。本例中只需要设定一个 Reduce 任务；

(4) 各计算节点启动 Reduce 任务，将有相同 key 的随机数值进行聚集操作，并将最终结果写入文件进行输出。

3.2 Map 过程

为了能采用并行计算来产生随机数，根据上述的串行算法，首先要获得 x、y、z 的值，分别用变量 i_x、i_y、i_z 来存储。每个 Map 任务计算得出一组随机数并存入数组中，然后由 Reduce 任务收集这几组随机数后写入文件，最后将结果输出。

由图 3 可知，Map 个数和每个 Map 任务要生成的随机数个数 m 被处理成<key, value>即(nMaps,m)的形式传给 Map 任务，经过 Map 任务计算处理后生成键值对(0,ran[0])，(0,ran[1])，(0,ran[3])，……，(0,ran[m-2])，(0,ran[m-1])，并将该结果传给 Reduce 任务处理。Map 算法伪码描述如下：

```
public void map(int nMaps,int val)
{
    int nMaps=key.get();          //获取指定的 Map 个数
    int m=val.get();              //获取分配的任务数
    float ran[]=new float[m];
        for(int j=0; j<MyMapNo; j++){
        计算各 Map 任务的第一个随机数的 xi、yi、zi，
        存入变量 ix、iy、iz
        }
    for(int i=1; i<nMaps; i++){
        计算各 Map 任务的相邻两个随机数的xi、yi、zi 的
        增量
    }
    k=(ix+iy+iz)%M4;
    ran[0]=(float) ((k+1.0)/M1);
    emit(0,ran[0]);               //输出 ran[0]的值
    for(int i=1; i<m; i++){
    计算各 Map 任务的 xi、yi、zi，存入变量 ix、iy、iz
        k=(ix+iy+iz-3)%M4;
        ran[i]=(float) ((k+1.0)/M1);
        emit(0,ran[i]); }      //输出 ran[1]到 ran[m-1]的值
}
```

由于 Map 函数生成的结果是随机数，在中间过程中不需要合并操作，因此这里没有必要设置 Combiner 进行合并处理。

3.3 Reduce 过程

Reduce 过程只需要简单地对 Map 操作结果进行收集，并以<key, value>键值对的形式写入结果文件，然后在主函数中对结果进行打印输出。Reduce 过程的主要代码如下：

```
public void reduce(int  key,Iterator values)
{
    ……
    SequenceFile.Writer writer = SequenceFile.createWrit-
    er(fileSys, conf, outFile,
    IntWritable.class,FloatWritable.class,CompressionType.
        NONE);
    while (values.hasNext()) {
                        //收集 Map 传递过来的随机数值
     float result= values.next().get();
                            //将随机数值写入结果文件
    writer.append(new IntWritable(0),new FloatWritable
        (result)); }
    writer.close();
    }
```

4 算法实例

4.1 实验环境

本文使用 4 台微机搭建了算法实验平台，建立了一

套 Hadoop 集群系统进行分布式并行计算,各计算节点所用操作系统均为 Linux CentOS5.3,内核版本为 2.6.18,使用的 Hadoop 版本为 0.18.3,JDK 版本为 jdk1.6.0_16。

4.2 实验结果分析

本实验所用的节点均为同等配置,各节点名分别为 master、slave01、slave02、slave03,master 作为 NameNode 和 Job Tracker也参与了计算,所以它也作为 DataNode 和 TaskTracker,而 slave01~slave03 既作为 DataNode 也作为 TaskTracker。在搭建的 Hadoop 集群系统上运行了本文开发的并行随机数发生器,实验结果数据如表 1 所示。

表 1 随机数并行计算效果分析表

任务数 / 规模/m	计算时间/s		加速比		并行效率/%	
	P=2	P=4	P=2	P=4	P=2	P=4
1 000	61.063	53.986	1.11	1.26	55.7	31.5
100 000	72.793	47.489	1.18	1.80	59.0	45.0
10 000 000	303.021	135.982	1.57	3.50	78.5	87.5

经过算法实验和分析表明:

(1)当计算数据规模较大时,处理器数由 2 增加到 4 可达到 87.5%的并行效率,说明 Hadoop 在数据规模和处理器数选择适当时,可取得良好的并行计算效率。但由于 Hadoop 框架的开销较大,且当通信网速度较低时,小规模数据的并行计算在 Hadoop 平台下不能获得较高的并行计算效率;

(2)MapReduce 模型适用于解决具有高度内在并行性的数据密集型并行计算问题;

(3)对于适于采用 MapReduce 模型的并行计算问题,用户只需分解出单个节点应完成的计算单元,按传统串行算法写出计算函数和数据收集函数,在 Hadoop 平台下就可方便地进行并行计算。Hadoop 平台无需用户掌握复杂的消息传递机制的并行程序设计方法,显著简化了并行计算的软件开发难度,有助于普及并行计算。

本文分析了基于 Hadoop 的 MapReduce 分布式并行计算编程模型,并采用该 MapReduce 模型成功实现了利用并行计算产生随机数的分布式并行随机数发生器。本文实验实现了 Hadoop 在由低端机组成的集群上进行各种分布式计算,展示了 Hadoop 的强大功能。现在 Hadoop 已经被广泛地应用于海量数据分析、互联网服务、科学计算、企业级应用等。在"云计算"时代即将来临之际,有理由相信,Hadoop 的前景会更光明,它必然会在云计算这个舞台上扮演更为重要的角色。

参考文献

[1] WHITE T. Hadoop, the definitive guide[M]. O'Reilly Media, Inc.2009.

[2] DEAN J, GHEMAWAT S. MapReduce: simplified data processing on large clusters. [C]// Proc of the 6th Symposium on Operating Systems Design and Implementation. San Francisco: Google Inc, 2004.

[3] Hadoop 官方文档: http://hadoop.apache.org/common/docs/r0.18.2/cn/mapred_tutorial.html, 2008.

(收稿日期:2010-01-20)

作者简介:

谢桂兰,女,1985 年生,硕士研究生,主要研究方向:分布式系统与网络并行计算技术。

罗省贤,女,1954 年生,教授,主要研究方向:高性能计算领域中的网络并行计算,网格计算信息及信息处理。