

MapReduce 并行编程架构模型研究

江务学¹, 张 璟², 王志明¹

(1 东莞职业技术学院, 广东 东莞 523808; 2 西安理工大学 计算机科学与工程学院, 陕西 西安 710048)

摘 要: 针对高质量高效率的 MapReduce 应用程序的开发, 分析了基于 Hadoop MapReduce 模型的工作机制, 从开发类库级阐述了 MapReduce 并行工作流程, 提出了一个具有通用性的 MapReduce 开发框架原型, 对开源的 MapReduce 模型的改进作了一些展望.

关键词: MapReduce 模型; 并行编程; 心跳程序

中图分类号: TP311

文献标识码: A

文章编号: 1000-7180(2011)06-0168-03

Study on Parallel Programming Framework Model Based on MapReduce

JIANG Wu-xue¹, ZHANG Jing², WANG Zhi-ming¹

(1 Dongguan Polytechnic, Dongguan 523808, China;

2 School of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China)

Abstract: For the development of high quality MapReduce application program, this paper analyses the working mechanism based on Hadoop MapReduce, describes the MapReduce paralleling workflow from the class library level, proposes a general framework prototype for development of MapReduce application, makes some prospect of improvement of open source MapReduce model.

Key words: MapReduce model; parallel programming; heart beat

1 引言

云计算软件系统的关键技术主要有云计算文件系统、计算模型和数据管理三大方面^[1], Google 云计算平台的关键技术分别是 GFS、MapReduce 和 BigTable, Hadoop 是与 Google 云平台关键技术的开源实现, 其中, Hadoop MapReduce 也是一种典型的并行开源云计算模型.

MapReduce 计算模型借鉴了早年的一种应用于人工智能领域的 Lisp 语言的设计思想^[2], 该语言利用把处理并发、容错、数据分布等的细节透明化的思想, 设计了在 map 操作和 reduce 操作, Map 操作对输入记录的每个逻辑块进行处理, 产生一组中间的键值对集, Reduce 操作对具有相同键的键值对进行数据合并.

对 MapReduce 计算模型的并行研究有基于面向过程语言环境级、基于面向对象的多线程级和基于 MPI 环境级的开发. 开发原理复杂, 开发过程冗余并且效率低. 为了能提供高质量高效率的大规模数据处理的并行计算, 本文对 Hadoop MapReduce 实现原理进行 Java 类库级的分析, 并提出了一个具有通用性的 MapReduce 开发框架原型, 以提高云计算程序的开发效率和云计算平台并行计算资源的利用率.

2 MapReduce 模型

MapReduce 是一种粗粒度的并行编程模型, 是谷歌公司为了在廉价的计算机集群上处理以 PB 数量级计算的大数据集而开发的新技术^[3]. MapReduce 提供了比多线程编程更上一层的接口, 屏蔽了

收稿日期: 2010-08-13; **修回日期:** 2010-10-26

基金项目: 国家高技术研究发展计划“八六三”(2007AA010305)

早期并行计算尤其是分布式处理的许多细节问题,降低了开发者的入门门槛,减轻了开发者的负担,提高了该模型的开发效率和对分布式资源的利用率. MapReduce 实现过程包含映射和规约两个主要过程,其间通过键值对来实现其间的过程信息传递.

Hadoop MapReduce 是运行于 HDFS 之上的一种并行运算模型, HDFS 属于主从式的文件系统, 一个主节点称为 NameNode, 其他属于从节点称为 DataNode. MapReduce 将 NameData 节点 JobTracker 上的作业 (Job) 分解成多个 Map 任务 (Task), 并且映射到各个 DataNode 节点的 TaskTracker 上. 一个 Map 任务可以执行在集群中的任何一个计算机节点上. 多个 Map 任务可以并行的执行在集群中的多个节点上. Map 任务负责转换输入记录成为键值对. 所有 Map 任务的输出会被重新组合成多个排序的集合, 其中的每一个集合会被派发给一个单独的 Reduce 任务^[4]. Reduce 任务对集合中排序的键值和关联在键值的多个数据值进行处理, Reduce 任务在集群中并行运行. 一个 MapReduce 任务的执行过程以及数据输入输出的类型可以表示为: $(input) <k1, v1> \rightarrow map \rightarrow <k2, v2> \rightarrow combine \rightarrow <k2, v2> \rightarrow reduce \rightarrow <k3, v3> (output)$.

3 类库开发级的 MapReduce

MapReduce 是一个在大量的通用目的的计算机上执行分布式计算任务的健壮的框架. MapReduce 实现一个作业调度包括以下过程: 创建 Job、初始化 Job、执行 task, 其间有 task 与 job 的不断通信过程.

(1) 创建 Job 过程

首先通过 JobConf 类创建一个 Job 实例并对其进行配置, 当 JobClient. runJob(job) 开始运行 Job 并生成一个 JobClient 实例后, 根据 JobConf 中所配置的具体 InputFormat 类将 Job 分割成若干个数据子集, 每一个数据集对应一个 Map 任务, 并将每个数据块的路径、偏移量和块大小等信息打包在 JobFile. jar 中. JobClient 通过其方法 submitJob() 向 JobTracker 提交作业, JobTracker 再按照 JobFile. jar 中的相关信息来调度执行 Job 并将一个记录了 Job 状态信息的 JobStatus 对象回传给 JobClient 实例, 同时, JobClient 创建一个 RunningJob 对象, 该对象能定期监控并打印 JobStatus 对象中的状态信息并把监控信息.

(2) 初始化 Job 过程

完成创建 Job 后, 便进入初始化 Job 的过程. JobTracker 接收到 Job 后, 会创建一个 JobInProgress 对象, 该对象根据 Job. jar 的位置、Map 任务和 Reduce 任务所要处理的数据等参数对任务进行管理和调度. 所有被提交的 Job 对象保存在 map 的一个成员变量 jobs 中, list 的成员变量 jobsByPriority 对已提交的 Job 对象进行动态的优先级管理, JobTracker 则通过 JobInitThread() 方法对 jobs 最高优先级的作业进行调度, 执行 initTasks() 方法后才对作业进行实质性的初始化.

(3) 初始化 Task 过程

JobInProgress 对象创建 TaskInProgress 对象, 该对象分别再创建 Map 的监控对象和 Reduce 的监控对象, 其中 Map 监控对象的个数由 JobClient. readSplitFile() 中的分割块的数量决定, Reduce 监控对象的个数则是由 JobConfig 类的设置决定的, 如果不指定, 则默认为一个 Reduce 监控对象, 之后 JobInProgress 创建 JobStatus 对象用来记录 job 的执行状态和进度等信息. JobTracker 启动任务时通过每一个 TaskInProgress 来 launchTask, 这时会把 Task 对象 (即 MapTask 和 ReduceTask) 序列化写入相应的 TaskTracker 服务中, 并根据配置启动各个 map task 和 reduce task.

(4) Task 与 Job 通信过程

启动 Job 和 Task 并分工执行后, JobTracker 负责初始化和调度 Job, TaskTracker 负责任务级的管理和执行, 过程中, TaskTracker 会循环调用 run() 方法与 JobTracker 连接, 一旦连接成功后, TaskTracker 每隔一定时间 (默认 10 s) 调用 heartBeat() 方法来与 JobTracker 建立通信: 先通过 heartBeat() 程序将本节点上所执行任务状态信息发送给 JobTracker; 再用 transmitHeartBeat() 方法获取 JobTracker 由于心跳程序的作用而发送来的的应答; 最后用 HeartbeatResponse. getActions() 方法获取 JobTracker 传送过来的指令集合, 该集合以数组的形式存放, 其内容包含 JobTracker 需要 TaskTracker 执行的新任务的和终止正在执行的任务等信息. 当然, TaskTracker 在向 JobTracker 发送心跳程序之前, 要监控该节点当前执行的任务数和本地磁盘的使用情况, 以便用来判断是否满足接受新任务的条件, 再根据条件来决定将心跳程序中的 askForNewTask 项置为 false 还是 true, 以控制是否接受 JobTracker 的新任务.

3 Hadoop MapReduce 并行编程框架

(1) Hadoop Map/Reduce 并行软件设计框架

通常,一个 hadoop MapReduce 程序至少包含 Mapper、Reducer 和 main() 主程序三大部分,其架构模式如下:

```
Class MapReduce{
    Class Mapper ...{
        .....
    }
    Class Reducer ...{
        .....
    }
    Main(){
        .....
        JobConf conf=new JobConf("MR. class");
        .....}
}
```

(2) mapper 类的设计

```
public class MapExam extends
MapReduceBase implements Mapper<常规系列参数 1
> //通过继承 MapReduceMap 类和应用 Mapper 接口来构造 MapExam 类
{ ..... public void map(常规系列参数 1) throws IO-
Exception //重写 map 方法
{ type key; type value; ..... { ..... output. collect
(key, new IntWritable(value)); }
}
}
```

(3) Reducer 类的设计

```
public class ReduceExam extends
MapReduceBase implements Reducer<常规系列参数 2
> //构造 ReduceExam 类
{ public void reduce(常规系列参数 2) throws IOExcep-
tion //重写 reduce 方法
{..... output. collect(key, new IntWritable(value));
}
}
```

(4) 主函数的设计

```
public class MainClass {
    public static void main(String[] args) throws IOExcep-
tion {
        .....
    }
    JobConf jobconf = new JobConf(getConf(),"作业名
称");
    jobconf. setJobName("Jobname");
```

```
FileInputFormat. addInputPath(jobconf, new Path(主
函数的第一个参数));
```

```
FileOutputFormat. setOutputPath(jobconf, new Path
(主函数的第二个参数));
```

```
.....
```

```
jobconf. setMapperClass(MainClassMapper. class);
jobconf. setReducerClass(MainClassReducer. class);
jobconf. setOutputKeyClass(Text. class);
jobconf. setOutputValueClass(IntWritable. class);
JobClient. runJob(conf);
}
}
```

4 结束语

MapReduce 编程模型是云计算软件系统的核心技术之一,它从复杂的实现细节中提炼出简单的业务处理模型,提供了一系列功能强大而又方便使用的接口,通过这些接口可以自发的并行和分布执行大规模的计算. MapReduce 是一种粗粒度的并行计算模型,究其执行内核,其并行机制的实现仍然类似于 Java 本身的多线程机制^[5]. Map-Reduce 不仅仅是一种编程模型,同时也是一种高效的任务调度模型. Map-Reduce 这种编程模型并不仅适用于云计算,在多核和多处理器、cell processor、P2P 以及异构机群上也具有良好的性能.

MapReduce 作为一种较为流行的云计算编程模型,应用前景广阔. 但其仍然存在着作业(任务)调度算法比较简单、消息通信机制比较单一、计算和存储分工不是很明确等问题^[6],为了提高 MapReduce 的计算效率,对 Map-Reduce 任务调度机制、与 HDFS 通信的数据存储策略、数据输入格式与切分和监控信息通信等方面做一些改进也是将来对 MapReduce 研究的热点问题.

参考文献:

- [1] 陈康,郑伟民. 云计算:系统实例与研究现状[J]. 软件学报,2009,20(5):1338-1348.
- [2] 王鹏. 云计算的关键技术与应用实例[M]. 北京:人民邮电出版社,2010.
- [3] 陈全,邓倩妮. 云计算及其关键技术[J]. 计算机应用,2009,29(9):2562-2566.
- [4] Zhang Liangjie, Zhou Qun. CCOA: cloud computing open-architecture[C]// IEEE International Conference on Web Services. Los Angeles, CA: Press IEEE Computer Society, 2009:608-612.

(下转第 175 页)

$$(5) \begin{bmatrix} \lambda^{j+1} \\ \gamma^{j+1} \end{bmatrix} = \begin{bmatrix} \lambda^j \\ \gamma^j \end{bmatrix} + \alpha_j p_j$$

$$(6) \delta_{j+1} = \begin{bmatrix} \lambda^{j+1} \\ \gamma^{j+1} \end{bmatrix} - \begin{bmatrix} \lambda^j \\ \gamma^j \end{bmatrix} = \alpha_j p_j, \text{ 如果 } \|\delta_{j+1}\|$$

$$\leq \epsilon \text{ 则停机, } \begin{bmatrix} \omega \\ \gamma \end{bmatrix} = \begin{bmatrix} \lambda^{j+1} \bar{\omega} \\ \gamma^{j+1} \end{bmatrix};$$

(7) 否则,

$$g_{j+1} = \nabla f(\lambda^{j+1}, \gamma^{j+1}),$$

$$\Delta g_j = g_{j+1} - g_j$$

(8) 利用 DFP 公式求 H_{j+1}

$$H_{j+1} = H_j + \frac{\delta_{j+1} \delta_{j+1}^T}{\delta_{j+1}^T \Delta g_{j+1}} - \frac{H_j \Delta g_{j+1} \Delta g_{j+1}^T H_j}{\Delta g_{j+1}^T H_j \Delta g_{j+1}};$$

$$(9) p_{j+1} = -H_{j+1} g_{j+1}, j = j + 1;$$

(10) 返回第(4)步;

(11) 应用所得到的最优解 $\begin{bmatrix} \omega \\ \gamma \end{bmatrix}$ 进行分类。

而对于非线性问题,也就是求解以下无约束条件的数学规划问题,再用 DFP-PSVM 算法求解。

$$\text{Min } \frac{\nu}{2} \|e - D(K(A, A')D\mu - e\gamma)\|^2 + \frac{1}{2} \left\| \begin{bmatrix} \mu \\ \gamma \end{bmatrix} \right\|^2 \quad (35)$$

同样采用 MIAS 库中的乳腺肿瘤 X 线图像,其中良性样本 85 个,而恶性样本 15 个。用 DFP-PSVM 方法的分类准确率提高到 89%,其中 81 个良性样本和 8 恶性样本被正确分类。实验证明,在小样本情况下,对分布极不平衡的数据,本文提出的 DFP-PSVM 算法能减弱多样本类对少样本类的影响,从而具有较高的分类能力。

5 结束语

本文采用基于可变形模型的梯度矢量流方法分割乳腺 X 光片的肿瘤区域,提取形状和纹理特征作为分类算法的特征输入,从而构造了一个基于支持向量机的医学图像分类器。

支持向量机利用接近边界的少数向量来构造一个最优分类面。然而当两类中的样本数量差别悬殊时,PSVM 算法则会过度拟合样本量大的那一类,而对样本量很小的那一类的错分率相当高。为解决此问题,本文提出了一种改进的支持向量机算法——DFP-PSVM 算法。提高了分类能力。

参考文献:

- [1] 边肇祺,张学工. 模式识别[M]. 北京:清华大学出版社,2000.
- [2] 李向军,庄彬,孟志青. 基于 V-SVM 的时态周期模式发现[J]. 微电子学与计算机,2009,26(11):169-176.
- [3] 汪健,张虹. MSSFD——一种递进式人脸检测算法[J]. 微电子学与计算机,2007,24(12):195-198.
- [4] Nakagawa T, Hara T, Fujita H, et al. Automated contour extraction of mammographic mass shadow using an improved active contour mode[C]// Proceedings of the 15th International Congress Series. New York: Computer Assisted Radiology and Surgery, 2004(1268):882-885.
- [5] Xu C, Prince J L. Global optimality of gradient vector flow[C]// Proc of 34th Annual Conference on Information Sciences and Systems (CISS'00), Princeton University, Ciss, 2000:624-629.
- [6] Rangayyan R M, Nema M, El-faramawy, onsy A. alim, measures of acutance and shape for classification of breast tumors[J]. IEEE Transactions on Medical Imaging, 1997, 16(6):799-810.
- [7] Fung G, Mangasarian O L. Proximal support vector machine classifiers[C]// Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining. NY, USA: ACM, 2001:77-86.

作者简介:

华翔女, (1979-), 博士, 讲师. 研究方向为图像提取、边缘检测。

(上接第 170 页)

- [5] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[C]// Proc 6th Symp on Operating System Design and Implementation, New York, ACM Press, 2004:137-150.
- [6] 杨文志. 云计算技术指南—应用、平台与架构[M]. 北京:化学工业出版社, 2010.

作者简介:

江学男, (1976-), 讲师. 研究方向为分布式系统和 Web 技术。

张璟男, (1952-), 博士, 教授, 博导. 研究方向分布式计算和计算机网络。

王志明男, (1971-), 教授. 研究方向为网络技术。