

基于 MapReduce 框架的航班串编制算法

张 康, 喻 瑛, 王伟杰

(上海大学 机电工程与自动化学院, 上海 200072)

摘 要:为解决小规模航班串编制问题,提出一种简单的非分布式算法,并在单机运行平台进行测试。然而,随着民航企业的迅速发展,航班数量不断增加,非分布式的航班串编制算法已经无法满足实际生产需求。为解决大规模航班串编制问题,提出另外两种基于 MapReduce 框架的分布式航班串编制算法。第一种算法将简单的非分布式算法扩展到 MapReduce 框架,解决大规模航班串编制问题;第二种算法在第一种算法的基础上进一步改进,优化 Map 和 Reduce 的处理流程,删除第一种算法中的迭代过程,充分发挥 MapReduce 框架的批处理优势。搭建 Hadoop 平台进行验证,实验结果表明,提出的两种分布式算法中,第二种算法即改进后的分布式算法,较之简单的非分布式算法和第一种分布式算法,能够有效提高大规模航班串编制效率。

关键词: MapReduce 框架; Hadoop 平台; 航班串编制; 大数据

中图分类号: TP305

文献标识码: A

文章编号: 1673-629X(2017)03-0142-05

doi:10.3969/j.issn.1673-629X.2017.03.029

Flight String Compilation Algorithm Based on MapReduce Frame

ZHANG Kang, YU Ying, WANG Wei-jie

(Institute of Electromechanical Engineering and Automation, Shanghai University, Shanghai 200072, China)

Abstract: A simple algorithm without distribution is proposed to solve the small scale flight string compilation problem and tested on stand-alone operation platform. However, with the rapid development of civil aviation enterprises and the rising number of flights, the simple algorithm has been unable to meet the requirement of practical production. Two new distributed flight string compilation algorithms based on MapReduce frame are proposed. The first one is extended from the simple algorithm to solve the large scale flight string compilation problem. And the second is made further improvements on the basis of the former where the processes of Map and Reduce are simplified and the iteration is deleted. A Hadoop platform is constructed to verify these algorithms. Results shows that compared to the simple algorithm and the first distributed algorithm, the second improved algorithm could effectively improve the efficiency of compiling flight string with large scale flights.

Key words: MapReduce framework; Hadoop platform; flight string compilation; big data

0 引 言

飞机排班是民航企业制定生产计划的一项基本内容,排班人员根据航班计划和飞机维修计划以及商务部提供的航班连线信息、航站信息、飞机本身信息等,给出飞机调度决策,为每个运营航班指定一架具体执行的飞机。2010 年以来,随着中国经济的快速发展,在市场需求旺盛和人民币持续升值的双重利好下,中国民航经历了新一轮的发展高峰。单就 2015/2016 年冬春航季,根据中国民航排定的航班计划,在国内航班方面,国内 39 家航空公司每周共安排的航班数量高达

54 956 班次,比 2014/2015 年冬春航季增长 6.2%。

航班串编制^[1-2]是飞机排班的核心工作。随着航班规模逐年扩大,民航企业传统上使用的非分布式航班串编制的方法已经无法满足实际生产需求,从而需要一种更高效的计算机技术处理大规模航班串编制问题。近年来,Hadoop 分布式计算平台在处理大规模数据问题上表现出了显著优势,吸引了产业界、政府和学术界的广泛关注。

Hadoop^[3-5]是 Apache 软件基金会旗下的一个开源分布式计算平台,已成为大数据分析的主流框架。

收稿日期:2016-04-16

修回日期:2016-08-10

网络出版时间:2017-02-17

基金项目:上海市 2015 年度“科技创新行动计划”高新技术领域项目(15511109700)

作者简介:张 康(1991-),男,硕士研究生,研究方向为项目调度;喻 瑛,副教授,通讯作者,研究方向为不确定理论及其应用、项目优化调度、可靠性研究。

网络出版地址: <http://www.cnki.net/kcms/detail/61.1450.TP.20170217.1623.020.html>

针对 Hadoop 的研究与应用的相关工作正在积极开展。文献[6]探讨了蚁群算法的几种并行方式与适用场景以及结合 MapReduce 编程框架的可行性。文献[7]利用 MapReduce 并行编程模式,提出了一种基于 Hadoop 平台的高效、可扩展并行挖掘算法,节省了因大数据挖掘过程中产生的大量数据通信、中间数据以及执行大量交集操作而产生的时间耗费。文献[8]给出了基于 MapReduce 的计算等价类的数据约简算法与朴素贝叶斯分类算法,实现了气象数据挖掘研究。文献[9]利用 Hadoop 平台对医保数据进行挖掘,利用 MapReduce 编程方法实现并行挖掘,基于 Hadoop 平台的医保数据挖掘。文献[10]基于列存储数据库 Hbase 的存储模型,提出了一种 MapReduce 框架下的分布式关联规则挖掘算法(MCM-Apriori 算法),并进一步将改进 MCM-Apriori 算法应用于基于 Hadoop 云平台的网上图书销售系统。

文中研究了基于 MapReduce 框架的航班串编制问题,提出一种简单非分布式算法,并与两种分布式算法进行对比。搭建一主两从 Hadoop 平台,分别进行不同规模的航班串编制,比较三种算法的性能。

1 相关理论

1.1 航班串编制数学模型

作如下定义(以下所有定义的变量均考虑的是一个排班周期内,规定一天为一个排班周期):

U 为航班集合, u_i 为 U 中第 i 个航班;

Q^0 为航班串集合;

Q_{u_i} 为以航班 u_i 为起始航班的航班串, $Q_{u_i} \in Q^0$;

$X_{u_j}^{Q_{u_i}}$ 表示若航班 u_j 在航班串 Q_{u_i} 中,为 1,否则为 0;

q_k 为航班串中的第 k 个航班;

ta_{q_k} 为航班 q_k 的到港时间;

td_{q_k} 为航班 q_k 的离港时间;

aa_{q_k} 为航班 q_k 的到港机场;

da_{q_k} 为航班 q_k 的离港机场;

t_0 为完成一次过站作业所需的最短时间。

航班串编制数学模型可以描述为:

$$\min(|Q^0|) \quad (1)$$

s. t.

$$\sum_{Q_{u_i} \in Q^0} X_{u_j}^{Q_{u_i}} = 1, \forall u_j \in U \quad (2)$$

$$ta_{q_k} + t_0 \leq td_{q_{k+1}} \quad (3)$$

$$aa_{q_k} = da_{q_{k+1}} \quad (4)$$

其中, $|Q^0|$ 表示航班串数量,目标函数为使生成的航班串数量最少,从而减少飞机的使用架数,降低成本。式(2)约束各航班只能存在于一条航班串中;式(3)为航班串中相邻航班间的最短过站时间约束;式

(4)为相邻航班间的航站衔接约束。

1.2 Hadoop 平台

Hadoop 基于 Java 语言开发,运行在 Linux 操作系统之上,核心是 HDFS 和 MapReduce。其中 HDFS 负责的是大规模数据的存储,MapReduce 负责数据的并行计算。Hadoop 基于主从式架构,通过 Namenode、Datanode、Jobtracker 和 Tasktracker 管理,主要特性是移动计算,而不是移动数据。计算前,Namenode 分析程序需要的数据存储在集群中的哪些 Datanode 节点;Jobtracker 将 MapReduce 计算任务分配给这些节点上的 Tasktracker;Tasktracker 启动 Map 程序,开启计算任务;经过 Combiner、Shuffle 等过程,在 Reduce 阶段生成计算结果^[11]。

1.3 MapReduce 框架

MapReduce^[11-12] 分布式编程模型允许用户在不了解分布式系统底层细节的情况下开发并行应用程序。作为一种海量数据处理的并行编程模型,MapReduce 将分布式编程分为 Map(映射)和 Reduce(归约)两个阶段,Map 函数负责分块数据的处理,Reduce 函数负责对分块数据处理的中间结果进行归约。

MapReduce 框架运行应用程序的过程如下:

(1)输入文件被分成固定大小(默认为 64 MB,用户可以调整的 M 个分片(split))。Master 节点会尽量将任务分配到离输入分片较近的节点上执行,以减少网络通信量。

(2)在 Map 阶段,MapReduce 模型以一组 $(key_1, value_1)$ 作为 Map 函数的数据输入,经过映射,聚合所有具有相同的 key 值的中间结果,产生一组中间结果 $list(key_2, value_2)$,中间结果存储在本地磁盘上。

(3)在 Reduce 阶段,所有 Map 中输出的数据经过分区、混洗、排序后都传入到 Reduce 函数,函数把具有相同 key 值的中间结果进行合并产生结果 $(key_2, list(value_2))$,最终生成输出文件。

2 航班串编制算法

文中提出的三种航班串编制算法如下:

算法 1:不使用 MapReduce 框架的简单算法。

算法 2:基于 MapReduce 框架对算法 1 进行扩展。

算法 3:在算法 2 的基础上进行改进。

使用的原始航班数据中包括航班号、离港机场、到港机场、离港时间和到港时间等航班信息。为方便起见,使用表 1 中的形式。其中,离港时间“0725”,表示该航班的离港时间为 7 点 25 分。

航班衔接需要考虑两个约束关系。约束 1 为机场衔接约束,即同一航班串中的前一航班的到港机场必须是下一航班的离港机场。约束 2 为过站时间约束,

即同一航班串中的后一航班的离港时间与前一航班的到港时间之差必须大于等于飞机完成一次过站操作的时间。

表 1 原始航班数据

航班号	离港机场	离港时间	到港机场	到港时间
101	1	0725	4	1255
...
116	1	1810	3	2110

文中某个航班的后续航班指与该航班同时满足约束 1 与约束 2,即该航班组成航班串的航班;后续航班集合为该航班所有后续航班的集合。

定义 n 为 U 中的航班数量, u_i 为 U 中的第 i 个航班; Q_{u_i} 为以航班 u_i 为起始航班的航班串; F_{u_i} 为航班 u_i 的后续航班集合;子函数 followFlightSet 生成后续航班集合;子函数 generate 生成航班串。

2.1 算法 1

算法 1 的航班串编制模型,输入文件为表 1 所示的原始航班数据。首先,航班集合 U 中的航班,均要从航班集合中选择与该航班同时满足约束 1 和约束 2 的航班,生成该航班的后续航班集合。然后,对 U 中的航班依次调用递归函数 generate。generate 中,依次判断航班集合中的航班是否属于该航班的后续航班集合;如果是,则将搜索到的航班加入以该航班为起始航班的航班串中,同时,将搜索到的航班标记为当前航班,继续调用递归函数 generate;直至搜索完航班集合中的航班。

算法 1 的具体步骤如下:

- (1) $i = 1$;
- (2)调用子函数 followFlightSet (u_i , U),生成 u_i 的后续航班集合 F_{u_i} ;
- (3) $i = i + 1$;
- (4)如果 $i \leq n + 1$,转到(2);
- (5) $i = 1$;
- (6) $q = u_i$, $j = 1$, $U' = U$;
- (7)调用子函数 generate (Q_{u_i} , q , j , U'),生成以航班 u_i 为起始航班的航班串 Q_{u_i} ;
- (8) $i = i + 1$;
- (9)如果 $i \leq n + 1$,转到(6);
- (10)输出航班串集合;
- (11)算法结束。

2.1.1 followFlightSet 函数的伪代码

输入参数:航班 u_i 、航班集合 U ;
输出参数: u_i 的后续航班集合 F_{u_i} 、followFlightSet (u_i , U)。
begin
 $F_{u_i} = \varnothing$;
 $j = 1$;

while $j \leq n$ do //搜索 U 中所有的航班 u_j
if u_j 与 u_i 同时满足约束 1 和约束 2 then //如果 u_j 是 u_i 的后续航班
 $F_{u_i} = F_{u_i} \cup \{u_j\}$; //将 u_j 加入 F_{u_i} 中
end if
 $j = j + 1$;
end while
return F_{u_i} ;
End

使用子函数 followFlightSet 对表 1 所示的原始航班数据进行处理,生成如表 2 所示的数据形式,即生成各个航班的后续航班集合。

表 2 预处理后的航班数据

航班	后续航班
101	:111:126:137:146:147
...	...
116	:106:128:131:134:142:144

2.1.2 generate 函数的伪代码

输入参数: q 为待处理航班串 Q_{u_i} 中的当前尾航班; j 为 U 中航班编号;航班集合 U' , $U' = U$;
输出参数: u_i 为起始航班的航班串 Q_{u_i} ;generate (Q_{u_i} , q , j , U')
begin
while $j \leq n$ do //搜索 U' 中所有的航班 u_j
if $u_j \in F_q$ then //如果航班 u_j 是 F_q 中的航班
 $Q_{u_i} = Q_{u_i} \cup \{u_j\}$; //将航班 u_j 加入航班串集合 Q_{u_i} 中
 $U = U / \{u_j\}$; //集合 U 去掉航班 u_j
 $q = u_j$; //更新航班串 Q_{u_i} 中的当前尾航班
 $j = 1$;
 $U' = U$;
generate (Q_{u_i} , q , j , U'); //继续调用 generate 函数
else //如果航班 u_j 不是 Q_{u_i} 中的航班,继续判断下一个航班
 $j = j + 1$;
end if
end while
end

2.2 算法 2

算法 2 使用 MapReduce 框架对算法 1 进行扩展。将算法 1 的步骤进行拆分,算法 1 中生成各个航班的后续航班集合的步骤,在算法 2 中的预处理中实现;算法 1 中生成航班串的步骤,在算法 2 的 Reduce 函数中实现。

(1) Main 函数。

算法 2 的 Main 函数包含四个步骤。第一步,对原始航班数据进行预处理,生成各航班的后续航班集合;第二步,分配 Map 函数;第三步,分配 Reduce 函数;第四步,输出航班串。

(2) Map 函数。

Map 函数,输入格式为<LongWritable,Text>,其中值为文本文件中的一行航班数据(以回车符作为行结束标记);键为该行的首字符相对于文本文件的首地址的偏移量。Map 函数的输出格式为<IntWritable,Text>,键为对每行文本进行拆分后的航班,值为该航班的后续航班集合。

(3) Reduce 函数。

Reduce 函数,输入数据为 MapReduce 框架对 Map 阶段输出的中间结果进行分区、混洗、排序后的数据,即键相同的值的集合。Reduce 函数的输入格式为<IntWritable,Text>,其中键为航班,值为该航班的后续航班集合。以当前航班、航班的后续航班集合、航班集合 U 等为输入数据调用 generate 递归函数,生成当前航班为起始航班的航班串。Reduce 函数的输出格式为<Text,Text>,键为当前航班,值为组成的航班串。

具体步骤如下:

(1) $i = 1$;

(2) $q = u_i, j = 1, U' = U$;

(3)调用子函数 generate(Q_{u_i}, q, j, U'),生成以航班 u_i 为起始航班的航班串 Q_{u_i} ;

(4) $i = i + 1$;

(5)如果 $i \leq n + 1$,则转到(2),否则,进入下一步;

(6)Reduce 函数输出航班串集合;

(7)结束。

2.3 算法3

算法3在算法2的基础上进一步改进,去除算法2中的递归等操作,简化 Reduce 函数,降低运算时间。

(1) Main 函数。

Main 函数中,首先对原始航班数据进行预处理,初始 flag = 0。然后,判断标记 flag,如果 flag == 1,则需要继续分配 Map 和 Reduce 函数;否则,不需要分配新的 Map 和 Reduce 函数。具体步骤如图1所示。

(2) Map 函数。

Map 函数,输入格式为<LongWritable,Text>,其中值存储文本文件中的一行航班数据(以回车符作为行结束标记);键为该行的首字符相对于文本文件的首地址的偏移量。Map 函数的输出格式为<Text,Text>,键为对每行文本进行拆分后的航班,值为以该航班为起始航班的航班串。

(3) Reduce 函数。

Reduce 函数,输入数据为 MapReduce 框架对 Map 阶段输出的中间结果进行分区、混洗、排序后的数据,即键相同的值的集合。Reduce 函数的输入格式为<Text,Text>,其中键为当前航班,值为以该航班为起始航班的航班串。Reduce 函数再次从航班集合 U 中,搜索当前航

班的后续航班,添加到航班串中组成新的航班串。Reduce 函数的输出格式为<Text,Text>,键为航班,值为航班串。具体步骤见图1。

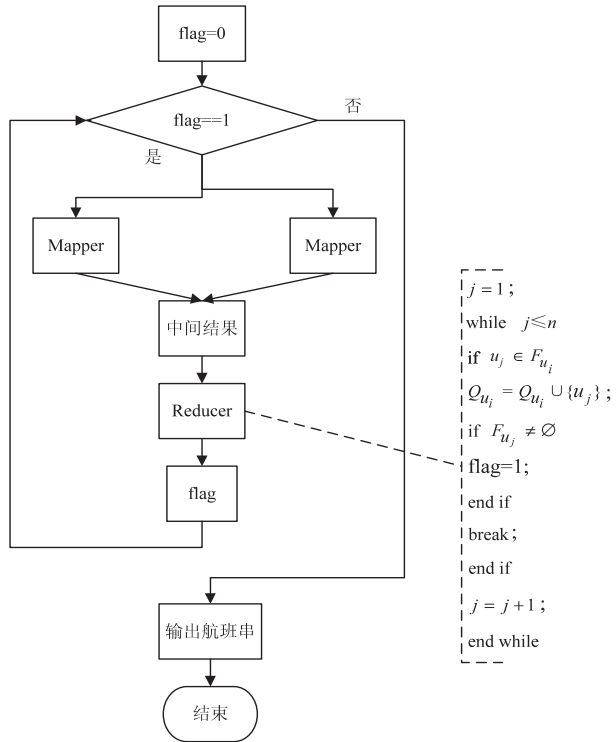


图1 算法3的流程图

3 算例分析

文中搭建的 Hadoop 集群,主节点 (Namenode) 命名为 master,从节点 (Datanode) 分别命名为 node₁ 和 node₂,如图2所示。三台计算机内存均为 2G,使用的 Linux 版本为 Ubuntu-14.04.2 desktop,Java 版本为 jdk-6u45-linux-x64,Hadoop 版本为 hadoop-2.6.0。

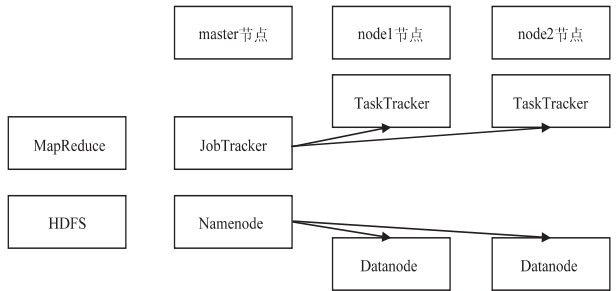


图2 一主两从 Hadoop 架构

分别实现三种航班串编制算法,测试相同航班数量下的运行时间。然后,逐步扩大航班规模,测试三种算法的运行时间。实验结果如表3所示。

从表3可知,算法1只适用于处理航班数量低于3000时的航班串编制问题,当航班规模为4000时,编制航班串的运算时间大于9个小时,是实际生产中无法接受的。算法2中,由于 Reduce 函数逻辑复杂,当航班数量为4000时,消耗的时间为8530844ms,虽

然少于算法 1,但是远高于算法 3。当航班数量小于 1 000时,算法 3 的运算时间大于算法 1 和算法 2;当航班数量大于 2 000 时,算法 3 的运算时间小于算法 1 和算法 2;尤其当航班数量大于 4 000 时,算法 3 的运算时间远小于算法 1 和算法 2。由此说明算法 3 在航班数量较大时,运算时间少,运行效率高。

表 3 不同算法的运行时间 ms

航班规模	算法 1	算法 2	算法 3
100	45	19 918	59 233
1 000	1 390	54 050	64 405
2 000	95 074	548 818	65 937
3 000	500 569	4 451 587	67 472
4 000	33 977 496	8 530 844	69 370
5 000	---	---	79 299
6 000	---	---	110 696
8 000	---	---	150 163
10 000	---	---	200 164
20 000	---	---	411 863

4 结束语

文中提出了三种航班串编制算法。算法 1 为不使用分布式框架的简单算法,编制小规模航班串的效率高于算法 2 和算法 3,但是不适用于编制大规模航班串。算法 2 在算法 1 的基础上进行扩展,使用分布式 MapReduce 框架,将算法 1 的步骤拆分,利用了其处理大数据的特点,编制大规模航班串的效率高于算法 1。算法 3 对算法 2 进行改进,去除了递归过程,简化了 Map 和 Reduce 阶段的流程,编制大规模航班串的效率远高于算法 1 和算法 2。算法 3 充分发挥了 MapReduce 框架批处理和大数据处理的优势,做到了扬长避短,提高了大规模航班串的编制效率,为民航企业进行

航班串编制提供了一种切实可行的方案。

参考文献:

[1] 李耀华,谭娜,郝贵和. 飞机排班航班串编制模型及算法研究[J]. 系统仿真学报,2008,20(3):612-615.

[2] 付维方,张伟刚,孙春林. 航班排班中航班串生成与筛选问题的算法与实现[J]. 中国民航学院学报,2006,24(5):4-6.

[3] White T. Hadoop 权威指南[M]. 周敏奇,王晓玲,金澈清,等,译. 第 2 版. 北京:清华大学出版社,2011.

[4] Ranger C, Raghuraman R, Penmetsa A, et al. Evaluating MapReduce for multi-core and multiprocessor systems[C]//High performance computer architecture. [s. l.]:[s. n.], 2007:13-24.

[5] Dean J, Ghemawat S. Mapreduce; a flexible data processing tool[J]. Communications of the ACM,2010,53(1):72-77.

[6] 王诏远,李天瑞,易修文. 基于 MapReduce 的蚁群优化算法实现方法[J]. 计算机科学,2014,41(7):261-265.

[7] 吕婉琪,钟诚,唐印浒,等. Hadoop 分布式架构下大数据集的并行挖掘[J]. 计算机技术与发展,2014,24(1):22-25.

[8] 张晨阳,马志强,刘利民,等. Hadoop 下基于粗糙集与贝叶斯的气象数据挖掘研究[J]. 计算机应用与软件,2015,32(4):72-76.

[9] 梁瑜. 基于 Hadoop 平台的医保数据挖掘[D]. 沈阳:东北大学,2012.

[10] 郭健,任永功. 云计算环境下的关联挖掘在图书销售中的研究[J]. 计算机应用与软件,2014,31(11):50-53.

[11] Dean J, Ghemawat S. MapReduce; simplified data processing on large clusters[J]. Communications of the ACM,2008,51(1):107-113.

[12] Dean J, Ghemawat S. MapReduce; simplified data processing on large clusters[C]//Proceedings of the 6th conference on symposium on operating systems design & implementation. Berkeley, CA, USA:USENIX Association,2004.