研究生国家奖学金申请审批表

<table>
<tr><td rowspan="6">基本情况</td><td>姓名</td><td>何马均</td><td>性别</td><td>男</td><td>出生年月</td><td>1993.08</td></tr>
<tr><td>政治面貌</td><td>团员</td><td>民族</td><td>汉</td><td>入学时间</td><td>2016.09</td></tr>
<tr><td>基层单位</td><td>电子科技大学</td><td>专业</td><td>软件工程</td><td>攻读学位</td><td>硕士</td></tr>
<tr><td>学制</td><td>3</td><td>学习阶段</td><td>☑硕士
□博士</td><td>学号</td><td>201621220146</td></tr>
<tr><td>身份证号</td><td colspan="5"></td></tr>
<tr><td>申请理由</td><td colspan="6">过去的这一年里，我以优秀研究生的标准要求着自己。使自己在各个方面都得到提升。
科研上，我从兴趣出发，根据自身研究方向上的要求，有针对性地阅读相关文献，并尝试撰写各类科研文档。
项目中，我努力踏实，不抛弃不放弃，遇到困难不退缩，积极尝试各种解决方案，以使项目更为完美。
工作之余，我也比较注重个人内在素质的提升，广泛阅读，开拓视野，尤为钟情于古文学作品。
生活上，我主动组织并参与实验室的各类活动，努力培养与维护同学之间来之不易的友谊。总之，这一年，对我来说，是不平凡的一年。

申请人签名：何马均

2018 年 09 月 21 日</td></tr>
</table>

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

On minimizing total energy consumption in the scheduling of virtual machine reservations

Wenhong Tian [a,b,*], Majun He [a], Wenxia Guo [a], Wenqiang Huang [a], Xiaoyu Shi [b], Mingsheng Shang [b], Adel Nadjaran Toosi [c], Rajkumar Buyya [c]

[a] School of Information and Software Engineering, University of Electronic Science and Technology of China (UESTC), China
[b] Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, China
[c] CLOUDS Lab., Dept. of Information and Computing Systems, The University of Melbourne, Australia

ARTICLE INFO

Keywords:
Energy efficiency
Cloud Data centers
Resource scheduling
Virtual machine reservation

ABSTRACT

This paper considers the energy efficient scheduling of virtual machine (VM) reservations in a Cloud Data center. Concentrating on CPU-intensive applications, the objective is to schedule all reservations non-preemptively, subjecting to constraints of physical machine (PM) capacities and running time interval spans, such that the total energy consumption of all PMs is minimized (called MinTEC for abbreviation). The MinTEC problem is NP-complete in general. The best known results for this problem is a 5-approximation algorithm for special instances using First-Fit-Decreasing algorithm and 3-approximation algorithm for general offline parallel machine scheduling with unit demand. By combining the features of optimality and workload in interval spans, we propose a method to find the optimal solution with the minimum number of job migrations, and a 2-approximation algorithm called LLIF for general cases. We then show how our algorithms are applied to minimize the total energy consumption in a Cloud Data center. Our theoretical results are validated by intensive simulation using trace-driven and synthetically generated data.

1. Introduction

Cloud computing has evolved from various recent advancements in virtualization, Grid computing, Web computing, utility computing and other related technologies. It offers three level of services, namely Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). In this paper, we concentrate on CPU-intensive computing at IaaS level in Cloud Data centers. Cloud computing providers (such as Amazon) offer virtual machine reservation services with specified computing units. For reservation services, customers request certain units of computing resources in advance to use for a period of time in the future, so providers can have enough time to do scheduling. The resources in this paper include:

1. Physical Machines (PMs): physical computing devices which can host multiple virtual machines; each PM can be a composition of CPU, memory, hard drives, network cards, and etc.
2. Virtual Machine (VMs): virtual computing platforms on PMs using virtualization software; each VM has a number of virtual CPUs, memory, storage, network cards, and related components.

The architecture and process of VM reservation scheduler are provided in Fig. 1, referring to Amazon EC2 (Amazon EC2). As noted in the diagram, the major processes of resource scheduling are:

1. User reservation requesting: the user initiates a reservation through the Internet (such as a Cloud service provider's Web portal);
2. Scheduling management: Scheduler Center makes decisions based on the user's identity (such as geographic location, etc.) and the operational characteristics of the request (quantity and quality requirements). The request is submitted to a data center, then the data center management program submits it to the Scheduler Center, finally the Scheduler Center allocates the request based on scheduling algorithms;
3. Feedback: Scheduling algorithms provide available resources to the user;
4. Executing scheduling: Scheduling results (such as deploying steps) are sent to the next stage;

* Corresponding author. School of Information and Software Engineering, University of Electronic Science and Technology of China (UESTC), China; Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, China.
E-mail addresses: tianwenhong@cigit.ac.cn, tian_wenhong@uestc.edu.cn (W. Tian), adel.nadjaran@unimelb.edu.au (A.N. Toosi), rbuyya@unimelb.edu.au (R. Buyya).

https://doi.org/10.1016/j.jnca.2018.03.033
Received 16 December 2017; Received in revised form 27 March 2018; Accepted 29 March 2018
Available online 7 April 2018
1084-8045/

On Minimizing the Makespan Of A Set of Offline MapReduce Jobs

Majun He, Wenxia Guo,Houwen Huang,Bo He,Jin Wang,Wenhong Tian
University of Electronic Science and Technology of China (UESTC)

***Abstract*—There are quite a few algorithms on minimizing the makespan of a set of offline MapReduce jobs. However, these algorithms are heuristic or suboptimal. The best-known algorithm for minimizing the makespan is 3-approximation by applying Johnson model. In this paper, we evaluate three algorithms in terms of approximation, computational complexity, stability and optimal configuration of the underline cluster for the first time.**

***Keywords*-MapReduce, Offline jobs, The total completion time, The Makespan, Approximation**

I. Introduction

Originally, MapReduce was designed for processing large batch jobs periodically in a FIFO (First-In-First-Out) fashion. There are quite a few algorithms on minimizing the makespan of a set of offline MapReduce jobs. However, these algorithms are heuristic or suboptimal.

Notice that the scenario we discussed applies to multi-tenant Hadoop clusters where Map and Reduce slots can be assigned to competing jobs. In virtualized environments (e.g., Amazon AWS), there is an alternative choice to (optimally or not) scheduling jobs on Map/Reduce slots; namely, one can assign a separate virtual Hadoop cluster to each job and leave it to the node-to-resource mapping of the underlying layer to handle job concurrency. In this paper, we focus on minimizing the makespan of a set of MapReduce jobs and compare three algorithms in terms of approximation, computational complexity and stability, theoretically.

II. Problem Formulation

We consider the following problem same as in [1] [2]. Let J= $\{J_1, J_2, \ldots, J_n\}$ be a set of n MapReduce jobs with no data dependencies among them. We consider the offline case in which all jobs are available at time 0. These jobs can be executed in any order. A MapReduce job J_i consists of two stages, a map stage M and reduce stage R. Each stage consists of a number of tasks. We consider MapReduce as two non-overlapped stages, i.e., map and reduce stage respectively. The job execution time is considered as the sum of the complementary, non-overlapping map and reduce stage execution times, same as in [1]. Let c_i denote the completion time of J_i (i.e., the time when J_i's reduce tasks are all finished). The makespan of the job set $\{J_1, J_2, \ldots, J_n\}$ is defined as C_{max} =$max_{i\in[n]}(c_i - t_0)$, where t_0 is the start-time of the first job. We aim to determine an execution order of the jobs $J_i \in J$ such that the makespan of all jobs is minimized.

Let S^M and S^R denote the set of map slots and reduce slots configured by MapReduce manager (i.e., S=$S^M \cup S^R$), so that the number of map slots and reduce slots are $|S^M|$ and $|S^R|$, correspondingly. Let ϕ denote the job execution order for a set of MapReduce jobs. We denote $|J_i^M|$ and $|J_i^R|$ as the number of tasks in J_i's map stage and reduce stage, respectively. Let $t_{i,j}^M$ and $t_{i,j}^R$ denote the execution time of J_i's j-th map task and j-th reduce task, respectively. Let T_i^M and T_i^R denote the expected execution time of J_i's map and reduce stage respectively. J_i requests $S_i^M \times S_i^R$ MapReduce slots and has Map and Reduce stage durations (T_i^M, T_i^R) respectively, the operator '$\times$' is for convenient representation of slots in both Map and Reduce stage but does not mean any operation. Let us set the actually allocated MapReduce slots for job J_i as $|A_i^M| \times |A_i^R|$, the max available MapReduce slots in the Hadoop cluster is $|S_i^M| \times |S_i^R|$.

III. Main Results

In this section, we firstly introduce UAAS (Utilizing All Available Slots) algorithm for offline MapReduce job scheduling; then we compare algorithm with two best known algorithms (BalancedPools [1] and MK_JR [2]) with regard to minimizing the makespan of a set of offline MapReduce Jobs.

The Pseudocode of the UAAS (Utilizing All Available Slots) algorithm is given in Algorithm III.1. Basically, UAAS has inputs given by the total number of MapReduce slots for a Hadoop cluster, estimated Map and Reduce durations of all jobs. UAAS applies two lists for jobs' durations. It firstly finds the shortest duration among all durations; if the job is the first Map type job, places the item at the first place, or place it at the last place if the item is Reduce type job. If the job is not the first Map type job, places the item right next to the previous job, i.e., in non-decreasing order of Map durations; if the job is not the first Reduce type job, places the item left next to the previous

A Comparative Study of Data Skew in Hadoop

Majun He, Guozhong Li, Chaojie Huang, Yufei Ye
School of Information and Software Engineering
University of Electronic Science and Technology of China
Chengdu, China

Wenhong Tian
School of Information and Software Engineering
University of Electronic Science and Technology of China
Chengdu, China

ABSTRACT
MapReduce which has been a well-known programming model processes numerous raw data in large scale clusters. However, great challenges have been brought to MapReduce programming model while routinely handling the big data. To mitigate the process time of the clusters through minimizing the makespan is one of the key challenges. For now, (data) skew is partly responsible for that and there are some methods presented by research teams from different perspectives. In order to fully understand and utilize the state-of-the-art of data skew problem, in this paper, we compare six algorithms: Hadoop default (speculative execution), SkewReduce, SkewTune, iShuffle, LEEN and LIBRA. They are compared in terms of architecture and main features, core algorithms, performance metrics and evaluation methods. Finally, a few challenging problems as future research trends are summarized.

CCS Concepts
• Theory of computation→Approximation algorithms analysis.

Keywords
Hadoop; MapReduce; (Data) Skew; Load balancing; Comparative study.

1. INTRODUCTION
MapReduce is a powerful and economical tool for processing massively parallel data[1]. The computational load is imbalanced among map tasks or reduce tasks, referred to map-skew and reduce-skew respectively. Longer job execution times and lower cluster throughput are brought about by all kinds of skew (a.k.a load imbalance).

During both the map phase and reduce phase, the data skew can happen [2]. Some subsets of input data are harder to process than others cause map skew. A method for specific application which splits large, expensive records into some smaller ones has been provided by Lin [3]. In contrast, reduce skew is much more challenging. The requirement that all tuples sharing the same key are dispatched to the same reducer should be satisfied. Many real world applications exhibit large amounts of data skew, including scientific applications [4], [5], distributed database operations, searching engine applications and some simple applications.

Figure 1 shows the timeline of reduce tasks from 1 PB Terasort running on the Hadoop by Yahoo! benchmark test[1]. We can find that a lot of reduce tasks take longer time to run. Table 1 from three product Hadoop clusters [6] shows the number of jobs that have stragglers in the map phase, reduce phase, and both.

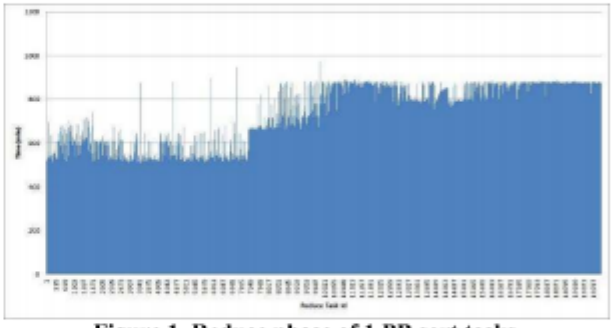

Figure 1. Reduce phase of 1 PB sort tasks.

To maximize the performance, all tasks of jobs need to be finished around the same time ideally. When some tasks take much longer time to complete, that is called a straggler and can postpone the makespan of the job. Hadoop default speculative execution (*default* for abbreviation) is an useful method where the slow tasks run on an alternative machine with the hope that it can complete there faster. An observation found by Google is that Hadoop speculative execution can reduce 44 percent time of job execution [1]. Unfortunately, when the cause of skew is an uneven data distribution across partitions or an uneven processing time across partitions, speculative execution is not helpful. This is because that re-executing the same task on the same input data but on a different machine would lead to the same slow execution time.

In the following sections, we will compare other five algorithms with regard to data skew handling. The rest of the paper is organized as follows. Section 2 provides the architecture of all algorithms. Main features are discussed in the section 3. Section 4 describes and analyzes the core algorithm in six methods. In section 5, we discuss the performance metrics and environment of each algorithm. Finally conclusions are given in section 6.

[1] http://sortbenchmark.org/.

ICNCC 2017, December 8–10, 2017, Kunming, China

ACM ISBN 978-1-4503-5366-3/17/12…$15.00
DOI: https://doi.org/10.1145/3171592.3171610

中华人民共和国国家知识产权局

610000

四川省成都市成华区建设北路二段四号电子科技大学
田文洪(13408064175)

发文日：

2018年02月28日

申请号或专利号：**201810003621.3**　　发文序号：**2018022400213840**

申请人或专利权人：田文洪 曾柯铭 吝博强 何马均

发明创造名称：一种基于卷积神经网络的高精度自动识别驾驶员不安全行为的方法与装置

发明专利申请初步审查合格通知书

上述专利申请，经初步审查，符合专利法实施细则第44条的规定。
根据专利法第34条的规定，专利申请自申请日起满十八个月即行公布。

初步审查合格的上述发明专利申请是以：
2018年2月10日提交的说明书摘要；
2018年1月3日提交的权利要求书；
2018年1月3日提交的说明书；
2018年1月3日提交的说明书附图
为基础的。

提示：

1.发明专利申请人可以自申请日起3年内提交实质审查请求书、缴纳实质审查费，申请人期满未提交实质审查请求书或者期满未缴纳或未缴足实质审查费的，该申请被视为撤回。

2. 专利费用可以通过网上缴费、邮局或银行汇款缴纳，也可以到国家知识产权局面缴。

网上缴费：电子申请注册用户可登陆 http://cponline.sipo.gov.cn，并按照相关要求使用网上缴费系统缴纳。

邮局汇款，收款人姓名：国家知识产权局专利局收费处，商户客户号：110000860。

银行汇款，开户银行：中信银行北京知春路支行，户名：中华人民共和国国家知识产权局专利局，账号：7111710182600166032。

汇款时应当准确写明申请号、费用名称（或简称）及分项金额。未写明申请号和费用名称（或简称）的视为未办理缴费手续。

了解更多详细信息及要求，请登陆 http://www.sipo.gov.cn 查询。

审 查 员：史博颖　　审查部门：专利审查协作北京中心初步审查部

联系电话：010-53962169

中华人民共和国国家版权局

计算机软件著作权登记证书

证书号：软著登字第2485254号

软件名称：基于深度学习的驾驶员违规行为识别系统
[简称：DeepDrive]
3.0

著作权人：田文洪

开发完成日期：2017年07月30日

首次发表日期：2017年07月30日

权利取得方式：原始取得

权利范围：全部权利

登记号：2018SR156159

根据《计算机软件保护条例》和《计算机软件著作权登记办法》的规定，经中国版权保护中心审核，对以上事项予以登记。

中华人民共和国国家版权局
计算机软件著作权登记专用章

No. 02376599

2018年03月09日