基于回填算法的时间感知的最小区域任务调度算法

袁佳欣 陈建新 肖 俊 吴道亮

(南京邮电大学无线宽带通信与传感网络技术教育部重点实验室 南京 210003)

摘 要 在云计算中,任务调度算法的好坏直接影响着云计算系统的性能,因此,一个优秀的云计算调度任务算法不仅能减少云计算数据中心的压力,更快、更好地处理用户大数据量的请求,而且还能使用户获得更好的用户体验。现有回填算法因考虑的指标过于单一,回填性能不佳,导致最终完成时间较长、任务时延较大的问题。为了解决这些问题,提出了基于回填算法的MRA算法;在此基础上,结合任务申请的处理器核心数与任务预计执行时长的关系,对等待任务进行回填作业。在进行回填作业的同时考虑了虚拟机的负载分布,实现了一定的负载均衡。实验结果表明,在任务最大完成时间、任务队列等待时延和虚拟机负载分布上,MRA算法均表现出优异的性能。

关键词 云计算,任务调度,Cloudsim,Infrastructure as a Service,QoS

中图法分类号 TP393 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.08.018

Time-aware Minimum Area Task Scheduling Algorithm Based on Backfilling Algorithm

YUAN Jia-xin CHEN Jian-xin XIAO Jun WU Dao-liang

(Key Lab of Broadband Wireless Communication & Sensor Network Technology, Ministry of Education, Nanjing University of Posts & Telecommunications, Nanjing 210003, China)

Abstract In the cloud computing, the task scheduling algorithm directly affects the performance of cloud computing system, so a good cloud computing scheduling task algorithm can not only reduce the pressure of cloud computing data center, deal with user's large amount of data requests faster and better, but also allow users to obtain better user experience. The existing backfilling algorithm considers single index, and its backfilling performance is poor, resulting in longer final completion time and longer task delay. In order to get rid of these limitations, an MRA algorithm based on backfilling algorithm was proposed. On this basis, the backfilling operation was performed on the basis of the relationship between the number of processor cores for task applications and the task execution time. In the backfilling operation, the virtual machine load distribution was also considered to achieve a certain load balancing. Experimental results show that the MRA algorithm has excellent performance in the maximum task completion time, task queue wait delay and load distribution of virtual machine.

Keywords Cloud computing, Task scheduling, Cloudsim, Infrastructure as a service, QoS

1 引言

近年来,随着信息技术的进步与发展,人们对计算机的计算性能、传输、存储的要求越来越高,一些传统的计算模式已经不能满足用户的需求。因此,云计算在该背景下应运而生。云计算作为下一代数据中心,提供了一种新型的可以按用户需求[1]来提供服务的模式。

通常,云计算有3种服务模式:IaaS,PaaS和SaaS,可以实现动态且灵活的应用供应。在云数据中心规模越来越大,用户数量日益增多的情况下,尽可能地提高处理用户提交的云任务的能力,减小用户等待时间,较好地保证用户的服务质量(QoS)[2]成为云计算领域研究的重要课题。越来越多的学者研究此课题[3],因此本文研究的基于云的任务调度算法具有较强的现实意义。

本文第2节介绍了该课题的相关研究;第3节对一些现有算法存在的问题进行了描述,并指出了缺陷;第4节提出了调度架构和MRA算法;第5节对设计的调度算法进行实验与仿真,以验证结果;最后总结全文。

2 相关工作

云计算的任务调度是指在有限时间内将大批量的任务分配到有限的资源上[4],一个好的任务调度算法是至关重要的。

到稿日期:2017-06-19 返修日期:2017-08-13 本文受南京邮电大学国自孵化基金(NY217021),中兴通讯P2P流媒体技术研究项目(2016外14)资助。

袁佳欣(1992-),男,硕士生,主要研究方向为云计算资源调度,E-mail:yuanjiaxin1992@163.com;**陈建新**(1973-),男,博士,副教授,硕士生导师,主要研究方向为可穿戴计算,E-mail:chenjx@njupt.edu.cn(通信作者);**肖 俊**(1987-),男,硕士生,主要研究方向为P2P流媒体技术研究;**吴道亮**(1991-),男,硕士生,主要研究方向为P2P流媒体技术研究。

此前，很多研究者都在研究基于云计算的调度算法，但其追求的最终目标不尽相同。有的学者提出任务调度算法来提高资源的利用率；有的学者追求最大完成时间，减少平均时延；还有一部分学者追求数据中心的最小能耗，以及面向经济模型的调度策略。不同的目标驱动产生了许多不同的调度算法。

Li 等[5]基于贪心算法对任务按照 QoS 分类，通过价值函数进行任务的调度，从而减少了提交的任务时间并增强了用户体验。Liu 等[6]提出了使用两个效用函数的实时在线调度算法，加快了任务的处理速度，并对超过 deadline 的任务进行惩戒。Gersovitz 等[7]提出了一种基于凸优化的算法，并制定了一套 SLA 策略，对超出特定上限的用户请求进行惩戒，最终降低了能耗，并且尽可能地满足了用户的需求。Patel 等[8]在多准则和多指标决策模型的基础上提出了一种改进的基于优先级的任务调度算法，最终缩短了任务的最大完成时间。Beghdad 等[9]使用了一种基于 Min-Min 算法并均衡处理所有资源的启发式调度算法，最终缩短了任务执行的最大完成时间，同时提高了资源的利用率。Suresh 等[10]在元调度器上使用了一种平衡螺旋法来优化传统的回填算法，提升了回填作业的效率，最终缩短了任务的最大完成时间。Vratt 等[11]提出了一种“P-Backfill”算法，按照优先级和出现频率将到来的任务分为 4 类进行回填调度，缩短了任务执行时间及平均等待时延。Liu 等[12]基于回填算法，通过实时计算任务的价值权重函数来分配任务至空闲虚拟机或者申请新虚拟机，减小了服务器的资源开销。

3　问题描述

上文的研究成果中，一些启发式算法存在收敛速度慢的缺点[12]，而基于回调算法的优化也因为考虑的指标过于单一而存在一定的缺陷。如文献[10]中，IBA 算法在使用 Balanced Spiral 方法对等待任务进行分类排序处理时，只考虑了提交任务所申请的处理器核心，按照处理器核心数将任务排列成“V”字形，然后对新提交的任务进行回填作业。初始化的任务队列为{Job1，Job2，Job3，Job4，Job5，Job6，Job7，Job8}，其中，Job1 和 Job2 已经提交至虚拟机，它们申请的 PE (Processing Element)数量和执行时间如表 1 所列。

表 1　任务参数

Table 1　Task parameters

Job id	Job1	Job2	Job3	Job4	Job5	Job6	Job7	Job8
PE	9	5	8	2	5	9	3	3
Time	3	4	3	3	3	3	2	4

传统回填算法的时空图如图 1(a)所示。传统回填的做法是优先将等待任务队列中申请 *PE* 数量较少的任务回填至前面运行，图 1(a)中回填的任务为 Job4，Job7，Job8，这种情况下最终消耗的时间单位为 18。而图 1(b)中采用的是 IBA 算法，使用了 BS 方法对任务重排序，该方法是先将 Job1－Job8 按照申请的 *PE* 大小进行升序排序，此时任务顺序为{Job4，Job7，Job8，Job2，Job5，Job3，Job1，Job6}，接着使用 *Left* 和 *Right* 两个集合进行分类。因为 Job6 最大，将其放在最后，把 Job1 划入 *R* 集合，Job3 划入 *L* 集合，按照当前 *L* 集合和 *R* 集合的大小来决定剩余的 Job5，Job2，Job8，Job7，Job4 的放置位置。若 L_sum＜＝R_sum，则将当前任务放入 L 集合；反之，放入 R 集合。经过 BS 方法处理后，任务队列的顺序为{Job3，Job5，Job8，Job4，Job7，Job2，Job1，Job6}。又因为 Job1，Job2 已经提交且正在运行，而 Job3 又没有足够的空间回填，所以 IBA 算法实际进行回填的任务为 Job5，Job8，Job4。这种情况下，最终消耗的时间单位为 16，确实相比传统的回填算法有所提升。

但我们发现，传统回填算法和 IBA 算法都仅考虑了任务申请 *PE* 数量这一个维度。且上文列举的情境中，每个独立任务的耗时相差不大，如果将 Job4 和 Job8 的执行时间单位都增加到 10，通过图 1(c)可以发现 IBA 算法明显造成了虚拟机中相当大的空闲饥饿，任务的占空比也随之增大。这种情况下，任务最终消耗的时间单位为 22。而此时，对于传统回填算法，若仍按图 1(a)中的调度顺序，其最终消耗的时间单位会增加到 24。因此，任务在回填调度时只考虑任务申请 *PE* 数量这个单一指标是有一定缺陷的。

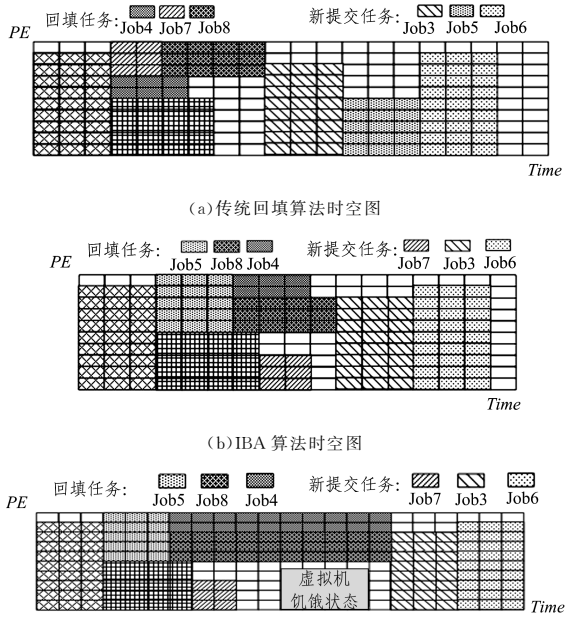

图 1　传统回填算法和 IBA 算法

Fig. 1　Traditional backfilling algorithm and IBA algorithm

此外，传统回填算法和现有的改进型回填算法旨在使任务的最终完成时间最小，并未考虑不同虚拟机上的负载均衡。FCFS，LJF 同样追求缩短任务的最大完成时间，减小平均时延。因此，不考虑负载均衡将很容易导致数据中心的主机负载过大而瘫痪，从而影响批量云任务的处理速度。

4　算法设计与架构

4.1　理论模型

由第 3 节可知，IBA 算法的不足在于仅考虑了任务申请的 *PE* 数量而未考虑每个独立任务的执行时间，从而造成了很大的任务饥饿，最终导致任务的执行时间变慢，平均时延

变大。而独立任务 i 的执行时间 $T(i)$ 如式(1)所示：

$$T(i)=\frac{MI_i}{PE_i * MIPS_j} \tag{1}$$

其中，MI_i 为任务 i 的长度，PE_i 为任务 i 申请的处理器数量，$MIPS_j$ 为虚拟机 j 的处理能力。$MIPS_j$ 的处理能力是不变的。因此，定义 T_{ratio} 为时间系数，可以得到式(2)：

$$\begin{cases} T(i)=T_{ratio} * \dfrac{1}{MIPS_j} \\ T_{ratio}=MI_i/PE_i \end{cases} \tag{2}$$

因此，在虚拟机选定的情况下，独立任务的执行时间完全取决于 T_{ratio}。第 3 节表 1 中调度的任务的 T_{ratio} 相差不大，一旦每个独立任务的 T_{ratio} 值的大小差异增大时，传统的回填算法和 IBA 算法就体现出劣势，造成虚拟机存在很大的饥饿。将申请的 PE 数量较小的任务称为窄任务，反之称为宽任务；将 T_{ratio} 较大的任务称为长任务，反之称为短任务。

图 1 中的时空图可以抽象为一个二维坐标系。横轴代表时间，整个虚拟机的 PE 数量在横轴为 T 的积分可以抽象为时空图中的面积。由此，可以得到式(3)：

$$S_{vm(j)}=\int PE\_sum_j \mathrm{d}T_{\mathrm{total}} \tag{3}$$

其中，$S_{vm(j)}$ 表示第 j 台 VM 的时空图面积，PE_sum_j 表示该 VM 的 PE 总量，T_{total} 是任务的最终完成时间。根据图 1，一旦 T_{total} 相差较大，如果将窄长任务放在最后执行会造成很大的占空比；若扫描前面的任务，针对窄长任务进行有效回填会提高虚拟机的资源利用率，减少饥饿，同时减小时空图的总面积。

由此，本文定义了式(4)：

$$BFI=\frac{PE_i}{T_{ratio}}=\frac{PE_i^2}{MI_i} \tag{4}$$

其中，BFI(Backfill Index)表示回填指数，我们会优先将 BFI 较小的任务进行回填。因此，在后续批量任务选定相应的虚拟机后，任务回填的优先级顺序如式(5)所示：

$$BFS=ASC\{BFI\}=ASC\{\frac{PE_i^2}{MI_i}\} \tag{5}$$

其中，BFS(Backfill Sequence)为回填顺序。根据 BFS 对选定虚拟机后的批量任务进行回填，考虑任务的执行时间因素来优先回填窄长任务，可以明显减少任务的饥饿程度，回填效果如图 2 所示。最终任务耗时为 19，相对图 1(c)中的效果有了显著提升。

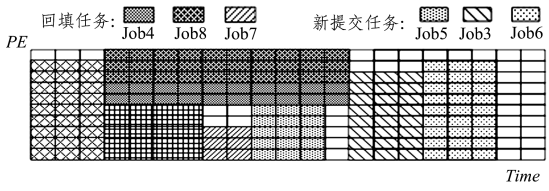

图 2 MRA 算法中的回填

Fig. 2 Backfilling in MRA algorithm

4.2 算法架构和具体步骤

针对 4.1 节的描述，提出了 MRA(Minimum Region Algorithm)算法，该算法旨在将虚拟机的 PE 数量和任务执行时间这两个维度维系为一个二维坐标系，对分配到虚拟机的窄长任务尽可能地进行回填以使整个虚拟机运行周期内饥饿空闲的时间降到最低，并且缩短任务的最大完成时间。整个过程抽象为尽可能使二维时空图所占用的面积最小。

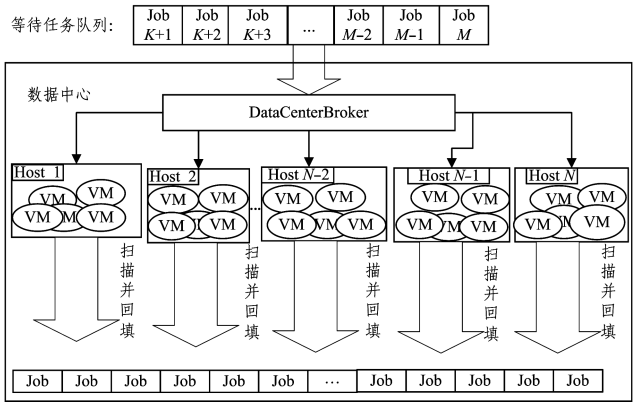

图 3 MRA 算法架构图

Fig. 3 Architecture of MRA algorithm

整个算法分为 4 个步骤：

1)将等待任务队列 Job$K$+1－Job$M$(Job1－Job$K$ 已经在虚拟机中运行)首先按照 MI 的值进行降序排列。

2)在数据中心将所有虚拟机按照 MIPS(Million Instructions Per Second)进行降序排列并建立索引信息。

3)数据中心代理将任务按照 1)的顺序尽可能提交到 MIPS 值最大的虚拟机上，在绑定的同时考虑虚拟机的负载。在数据中心存储每个虚拟机的任务运行总时间、每台虚拟机上运行的任务数量，并用一个负载权重来标记虚拟机的负载。首先将第一个任务绑定在虚拟机序列中 MIPS 最大的虚拟机上并更新负载权重，然后从第二个任务开始遍历任务序列。针对每一个任务，遍历所有虚拟机，若当前虚拟机未分配任务，则判断当前任务分配给该虚拟机是否最优，即任务预计运行时长是否最短；若当前虚拟机已分配过任务，则寻找分配到某台最优的虚拟机来完成分配并更新负载权重；若在虚拟机上分配当前任务且预计的负载加上虚拟机累积的负载相同时，寻找运行任务数量最少的虚拟机来完成初始绑定，这样可以防止所有任务都集中分配在性能最好的虚拟机上，从而保障最大完成时间最小的条件下无论虚拟机的性能好坏，每台机器的任务运行的总时长尽可能均衡，以此达到负载均衡。

4)对提交到当前虚拟机中正在运行的任务 Job1－Job$K$ 进行回填扫描。若有可回填的空间，将提交后的任务按照式(3)中 BFS 的顺序进行回填。

MRA 算法的伪代码如算法 1 所示。第 1 行对应于步骤 1)，第 2 行对应于步骤 2)，第 3－18 行对应于步骤 3)，第 17－21 行对应于步骤 4)。其中第 5－15 行考虑了负载均衡。

算法 1 MRA 算法

Input：taskList，vmList

Output：scheduling result

1. Descending sort tasks in waiting task queue by MI

2. Descending sort vm by MIPS and create the index of vmList

3. Scheduling the first task $task_1$ to the fast vm vm_1

4. for i=2→N do
5. for j=2→M do
6. if the task of vm_j=null then
7.　　Calculate the execute time on vm_j
8. if the execute time is shortest then
9. mapping $task_i$ to this vm and update load value
10. if the task of vm_j≠null then
11.　　Calculate the execute time of $task_i$ run on vm_j
12.　　mapping to the vm_j that execute faster and lowest load value
13. else if currentVmLoad equals the load of vm_j then
14.　　Chose the vm_j which the running task least and update load value
15. end for
16. end for
17. for j=1→M do
18.　Scanner the vm_j
19. if vm_j has extra space then
20.　Backfill the task from task queue on vm_j according the BFS policy
21. end for

5　实验仿真和结果

5.1　实验环境介绍

为了通过实验验证 MRA 的调度情况，本文采用 Cloudsim 3.0.3 仿真平台对 MRA 算法进行仿真。Cloudsim 是澳大利亚墨尔本大学开源的一套科学有效的云平台仿真工具，我们在 Cloudsim 中实现了提出的调度器，在 DataCenterBroker 中加入了 MRA 任务调度算法，并将其与 FCFS，LJF 和 IBA[10] 算法进行了性能对比。

5.2　实验指标参数

实验的初始化参数如表 2 所列。

表 2　实验初始化参数

Table 2　Experimental initialization parameters

Type	Parameters	Value
Datacenter	Number of Datacenters	1
	Number of Hosts	10
	Number of Pes per Host	16
	Performance of per Host	2000MIPS
	RAM&Storage	4096&100000
	BandWidth	1000
Virtual Machine	Total Number of VMs	20
	Numbers of Pes per Vm	8
	Perfomance of per VM	1000～2000MIPS
	RAM&Storage	2048&10000
Cloud Task	Type of Manager	SpaceShared
	Number of Pes per Task	1～8
	Length of per Task	10000～100000

我们建立了一个数据中心，其中包含 10 台主机，每台主机包括 16 个核心，每个核心的处理能力为 2000MIPS。虚拟机有 20 台，每台 8 个核心，每个核心的处理能力在 1000～2000MIPS 的范围内随机产生。任务在虚拟机上的策略采取 SpaceShare 模式。我们分别初始化了 500，1000，1500，2000，2500，3000 个任务，每个任务申请的核心数在 1～8 之间随机产生。任务的长度 MI 在 10000～100000 之间随机取值。

5.3　性能对比分析

实验中初始化 500～3000 个任务，对这些任务分别使用 4 种算法来进行调度。实验对比了 FCFS，LJF 和 IBA[10] 算法的 4 个指标，分别是最大完成时间、平均完成时间、平均时延和负载分布。具体结果如下。

5.3.1　最大完成时间

任务的最大完成时间的计算式如式(6)所示：

$$\min Makespan=\mathrm{Max}\{StartTime_i+\frac{MI_i}{PE_j*MIPS_j}\}$$
$$i\in[1,M],j\in[1,N] \quad (6)$$
$$\text{s.t. } \sum PE_i(t)\leqslant PE_j$$

其中，M 表示有任务的数量，N 表示虚拟机的总数。$StartTime_i$ 表示第 i 个任务的开始执行时间，PE 和 $MIPS_j$ 分别表示第 j 台虚拟机处理器的核心数量和处理器的 MIPS 值，$PE_i(t)$表示 t 时刻的任务申请 PE 的数量。

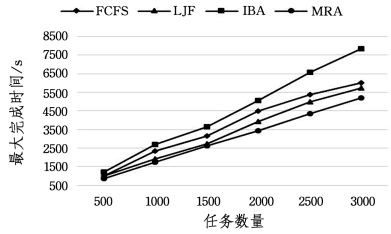

图 4　最大完成时间

Fig. 4　Makespan

从图 4 中可以看出，MRA 在不同任务数量下的最大完成时间均为最短，LJF 次之，而 IBA 算法中由于每个独立任务执行时间的差异较大，若仍按照 BS 方式处理会造成任务饥饿的加大，从而降低资源利用率。因此，任务的最大完成时间最长。

5.3.2　平均完成时间

任务的平均完成时间指所有任务完成时间的算数平均值，反映所有任务的平均完成时长，如式(7)所示：

$$AvgFinishTime=\frac{1}{M}\sum_{i=1}^{M}FinishTime_i \quad (7)$$

其中，M 表示任务的总个数。$FinishTime_i$ 是任务 i 的完成时刻。

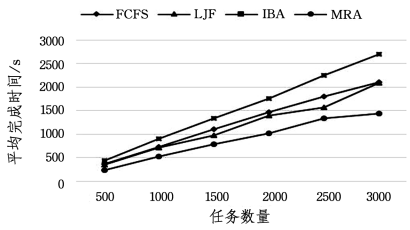

图 5　平均完成时间

Fig. 5　Average completion time

由图 5 可以看出，因为 MRA 的回填效率最高，所以其任务的平均完成时间仍然是最短的。整体情况与最大完成时间基本一致。

5.3.3　平均时延

在按需付费的云计算模式中，QoS 的保障非常有必要，其

中很关键的一个指标就是任务的平均时延。如果数据中心调度批量任务的时延过大，将影响使用者的体验。任务的平均时延的计算式如式(8)所示：

$$AvgDelay=\frac{1}{M}\sum_{i=1}^{M}ExecStartTime_i-SubmissionTime_j \tag{8}$$

如图 6 所示，IBA 虽然完成了平衡螺旋回填任务，但是只考虑了 PE 这一个维度，没有考虑任务执行时间的影响，因此造成了资源的占空，导致其平均时延最大。LJF 仅优先调度长任务，没有考虑 PE 的占用量，会造成大量的短任务长时间等待，所以其整体平均时延也较大。由于任务是随机生成的，因此 FCFS 的不确定因素较大，整体不是最优。MRA 算法由于综合考虑了 PE 的限制和任务的执行时间两个维度的关系来进行回填作业，不仅能回填长任务，而且还能回填短任务，减少了任务等待时间，降低了任务的饥饿程度。因此，其平均时延优于其他算法。

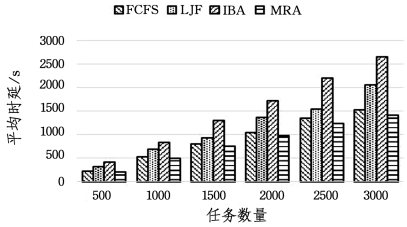

图 6 平均时延

Fig. 6 Average delay

5.3.4 负载分布

在云数据中心，主机或者虚拟机的负载均衡非常重要，其会影响整个数据中心运行的稳定性，如果负载得不到均衡，严重时将造成虚拟机或者主机的宕机。本文的 MRA 算法在进行回填调度作业时充分考虑了虚拟机的负载，在虚拟机处理相同负荷的情况下，将当前任务优先分配给运行任务较少的虚拟机，从而有效地保障各个虚拟机的性能不被破坏。从图 7 可以明显看出，在 20 台性能不同的虚拟机中，MRA 的负载最均衡。

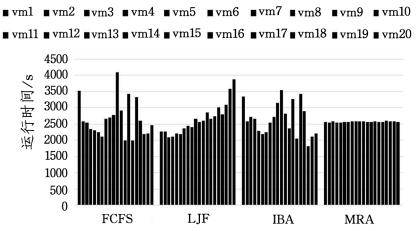

图 7 负载分布

Fig. 7 Load distribution

结束语 本文提出了一种基于回填作业的时间感知的任务调度算法，在虚拟机 PE 的空间限制的约束条件下，考虑任务的执行时间的影响，结合任务申请的 PE 数量与独立任务执行时间之间的关系对处于等待的批量任务进行作业回填。回填的同时对任务与虚拟机之间的绑定考虑了负载均衡，使得整个数据中心更加稳定。本文的实验结果表明，在任务最大完成时间、任务平均完成时间、平均时延和负载分布这 4 个指标中，MRA 的性能是最优异的。

参 考 文 献

[1] ELHADY G F, TAWFEEK M A. A comparative study into swarm intelligence algorithms for dynamic tasks scheduling in cloud computing[C]// IEEE Seventh International Conference on Intelligent Computing and Information Systems. IEEE, 2015: 362-369.

[2] MITTAL S, KATAL A. An optimized task scheduling algorithm in cloud computing[C]// IEEE Sixth International Conference on Advanced Computing. IEEE, 2016: 197-202.

[3] NOROOZOLIAEE M, HAMDAOUI B, GUIZANI M, et al. Online multi-resource scheduling for minimum task completion time in cloud servers[C]// Computer Communications Workshops. IEEE, 2014: 375-379.

[4] WADHONKAR A, THENG D. A survey on different scheduling algorithms in cloud computing[C]// International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics. IEEE, 2016: 665-669.

[5] LI J, FENG L, FANG S. An Greedy-Based Job Scheduling Algorithm in Cloud Computing[J]. Journal of Software, 2014, 9(4): 921-925.

[6] LIU S, QUAN G, REN S. On-Line Scheduling of Real-Time Services for Cloud Computing[C]// World Congress on Services. IEEE Computer Society. 2010: 459-464.

[7] GERSOVITZ M. SLA-based Optimization of Power and Migration Cost in Cloud Computing[C]// IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. IEEE, 2012: 172-179.

[8] PATEL S J, BHOI U R. Improved Priority Based Job Scheduling Algorithm in Cloud Computing Using Iterative Method[C]// International Conference on Advances in Computing & Communications. 2014: 199-202.

[9] BEGHDADBEY K, BENHAMMADI F, BENAISSA R. Balancing heuristic for independent task scheduling in cloud computing[C]// International Symposium on Programming and Systems. IEEE, 2015: 1-6.

[10] SURESH A, VIJAYAKARTHICK P. Improving scheduling of backfill algorithms using balanced spiral method for cloud metascheduler[C]// 2011 International Conference on Recent Trends in Information Technology (ICRTIT). IEEE, 2011: 624-627.

[11] VRATT SINGH L S, AHMED J, KHAN A. An Algorithm to Optimize the Traditional Backfill Algorithm Using Priority of Jobs for Task Scheduling Problems in Cloud Computing[J]. International Journal of Computer Science & Information Technology, 2014, 5(2): 1671-1674.

[12] LIU S, REN K, DENG K, et al. A task backfill based scientific workflow scheduling strategy on cloud platform[C]// Sixth International Conference on Information Science and Technology. 2016: 105-110.