基于 QoS 分类的任务调度算法研究

赵科伟,洪 龙,周宁宁
(南京邮电大学 计算机学院,江苏 南京 210000)

摘 要:为了提高云服务供应商提供服务的质量和用户对云服务的满意度,文中提出了一种基于 QoS 分类的任务调度算法。该算法针对的是独立任务的调度,也就是任务之间没有依赖关系,因此该方法可以利用模糊聚类算法对任务集进行分类,然后采用传统的分段 Min-Min 算法进行任务的分配。分段 Min-Min 算法相比 Min-Min 算法是以更小粒度来分配资源,因此能提高任务和资源之间的匹配程度,在此基础上针对某些节点负载过重的情况采取优化方法,这样能进一步降低完成时间,同时取得了一定程度的负载均衡。实验结果表明,提出的改进方法既能满足用户的 QoS 需求,又能取得较短的完成时间。

关键词:云计算;任务调度;Min-Min;模糊聚类;负载均衡
中图分类号:TP301.6 **文献标识码**:A **文章编号**:1673-629X(2016)07-0065-05
doi:10.3969/j.issn.1673-629X.2016.07.014

Research on Task Scheduling Algorithm Based on QoS Classification

ZHAO Ke-wei, HONG Long, ZHOU Ning-ning
(School of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210000, China)

Abstract: In order to improve the quality of cloud service providers and the satisfaction of users about cloud services, a task scheduling algorithm based on QoS classification is put forward. This algorithm is suited with independent tasks. Firstly, fuzzy clustering algorithm is used to classify task set. Then the traditional segmented Min-Min algorithm is applied for task allocation. Segmented Min-Min algorithm is more granular for resource allocation compared with the Min-Min algorithm, so it can improve the matching degree between task and resource. Only in this way can further reduce the completion time, and achieve a certain load balancing. The experimental results show that the proposed method can not only meet the needs of the user's QoS, but also obtain shorter completion time.

Key words: cloud computing; task scheduling; Min-Min; fuzzy clustering; load balance

1 概 述

云计算是一个用时付费的模型,用户根据这个模型按需访问网络提供的共享的、可配置的计算机资源池(如网络、服务器、存储、应用和服务)[1]。云计算的承诺是为用户提供按需服务,因此特别注重 QoS,而任务调度策略的优劣直接影响着服务质量,毫无疑问这成为研究云计算的一个重要组成部分[2-3]。

为了解决任务到资源节点的映射问题,国内外许多专家学者做了大量的工作。由于调度问题是 NP 完全问题,所以许多启发式算法被提出。这些映射启发式算法大致分为两类:最小化任务完成时间的算法和人工智能的调度算法。最小化任务完成时间算法主要有最小完成时间(MCT)算法[4]、最小执行时间(MET)算法、最小最小(Min-Min)算法、最大最小(Max-Min)算法[5],这些算法都是静态的调度算法。对 Min-Min[6] 算法来说,它的主要思想是,假设云计算环境由 n 个任务 $T=\{T_1,T_2,\cdots,T_n\}$ 和 m 个虚拟机 $R=\{R_1,R_2,\cdots,R_m\}$ 组成,当集合 T 不空时,一直执行如下步骤直到集合 T 为空:

(1)计算每个任务在每台虚拟机上的完成时间,找出每个任务对应的最小完成时间,把此最小完成时间放到一个数组 completeTime[n]中;

(2)从 completeTime[n]中选择最小的完成时间对应的任务 T_i ($1 \leqslant i \leqslant n$,其中 i 为正整数)进行调度;

(3)从集合 T 中删除前面已经调度过的任务;

收稿日期:2015-09-07 **修回日期**:2015-12-10 **网络出版时间**:2016-05-25
基金项目:国家自然科学基金资助项目(61170322,61373065,61302157);软件开发环境国家重点实验室开放课题(SKLSDE-2011KF-04)
作者简介:赵科伟(1988-),男,硕士生,研究方向为云计算与大数据处理;洪 龙,教授,博士,硕士生导师,研究方向为数理逻辑与分布式系统;周宁宁,博士,副教授,通讯作者,研究方向为图像分析与处理。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20160525.1706.026.html

(4)重复步骤 1 ~ 3,直到集合 T 为空。

可以看出 Min-Min 算法的不足之处:

(1)忽略了任务的 QoS 需求,不能满足任务的多样化需求;

(2)总是优先调度短任务,而且总是把任务调度到运行速度最快的虚拟机上面。

针对这些缺点,Max-Min 算法从最小完成时间的数组中选择最大的完成时间的任务进行调度。针对 Min-Min 算法容易造成负载不均衡的问题,Min-Min-Opt 算法将负载重的虚拟机上的任务调度到负载轻的虚拟机上来平衡负载。Segmented Min-Min[7] 首先将每个任务在所有虚拟机上运行时间的平均值作为任务的关键字,其次将关键字按照降序排列,并且将关键字划分为 N 段,最后针对每一段分别调用 Min-Min 算法,能很好地提高算法的运行效率。而人工智能调度算法主要有遗传算法[8]、蚁群算法[9]、粒子群算法[10]等。

以上这些算法在将任务与资源进行匹配时,大多考虑的是计算型任务,没有考虑任务的其他属性。传统的聚类算法把每个对象严格地划分到各个类别当中,然而实际生活中有许多对象没有严格的属性,模糊聚类主要针对这些对象进行聚类。文中采用的模糊聚类算法是模糊 C 均值算法[11-12],根据任务对时间、带宽、存储的需求将任务分为 3 类,然后针对不同任务采用不同的调度算法。

2 调度模型

2.1 任务模型

用户指请求云计算资源的实体。第 i 个用户 u_i 采用三元组形式表示为($u_{i-\mathrm{Id}}$, $u_{i-\mathrm{name}}$, $u_{i-\mathrm{task}}$)。其中, $u_{i-\mathrm{Id}}$ 作为该用户的唯一标识, $u_{i-\mathrm{name}}$ 表示用户的名字, $u_{i-\mathrm{task}}$ 表示该用户提交的任务集合。如此,请求云计算资源的用户集合为 $U = \{u_1, u_2, \cdots, u_n\}$ 。设任务集合中的任务是相互独立的,则 $u_{i-\mathrm{task}}$ 可以描述为 $u_{i-\mathrm{task}} = \{t_{i-1}, t_{i-2}, \cdots, t_{i-m}\}$, t_{i-j} 代表第 i 个用户的第 j 个子任务,而 t_{i-j} 又可表示为($t_{i-j-\mathrm{length}}$, $t_{i-j-\mathrm{time}}$, $t_{i-j-\mathrm{bw}}$, $t_{i-j-\mathrm{store}}$)。其中, $t_{i-j-\mathrm{length}}$ 代表第 i 个用户的第 j 个子任务的长度, $t_{i-j-\mathrm{time}}$ 代表第 i 个用户的第 j 个子任务对时间的需求, $t_{i-j-\mathrm{bw}}$ 代表第 i 个用户的第 j 个子任务对带宽的需求, $t_{i-j-\mathrm{store}}$ 代表第 i 个用户的第 j 个子任务对存储的需求。假设每个用户最多有 m 个子任务,然后将所有用户的所有任务按顺序编号,因此 $T = \{t_1, t_2, \cdots, t_{mn}\}$ 代表需要调度的所有子任务的集合。

2.2 资源模型

云系统中的资源由资源池中虚拟机表示,第 j 个 vm_j 采用三元组形式表示为($\mathrm{vm}_{j-\mathrm{Id}}$, $\mathrm{vm}_{j-\mathrm{type}}$, $\mathrm{vm}_{j-\mathrm{perform}}$)。其中, $\mathrm{vm}_{j-\mathrm{Id}}$ 作为虚拟机的唯一标识, $\mathrm{vm}_{j-\mathrm{type}}$ 表示虚拟机的类型, $\mathrm{vm}_{j-\mathrm{perform}}$ 表示虚拟机的性能。如此,云计算资源集合为 $V = \{\mathrm{vm}_1, \mathrm{vm}_2, \cdots, \mathrm{vm}_k\}$ 。虚拟机的性能又可以采用四元组形式表示为($\mathrm{vm}_{j-\mathrm{perform-comp}}$, $\mathrm{vm}_{j-\mathrm{perform-price}}$, $\mathrm{vm}_{j-\mathrm{perform-fault}}$, $\mathrm{vm}_{j-\mathrm{perform-ram}}$)。其中, $\mathrm{vm}_{j-\mathrm{perform-comp}}$ 表示第 j 台虚拟机的计算能力, $\mathrm{vm}_{j-\mathrm{perform-price}}$ 表示第 j 台虚拟机的价格, $\mathrm{vm}_{j-\mathrm{perform-fault}}$ 表示第 j 台虚拟机的故障率, $\mathrm{vm}_{j-\mathrm{perform-ram}}$ 表示第 j 台虚拟机的内存容量。

2.3 任务分配模型

传统调度算法主要包括在线模式调度算法、批处理模式调度算法、启发式调度算法。在线模式中,一个任务一旦到达调度系统就立马分配虚拟机。批处理模式中,任务到达时没有被立即调度,而是放入一个预调度的集合中,这个集合中的任务叫元任务,这些元任务将在一个预定时间进行统一调度,在统一调度之前如何找到一种分配策略来缩短任务完成的总时间是调度问题的关键。因此元任务的分配由元任务到虚拟机的映射表示。例如,图 1 就表示元任务到虚拟机的一种分配策略。

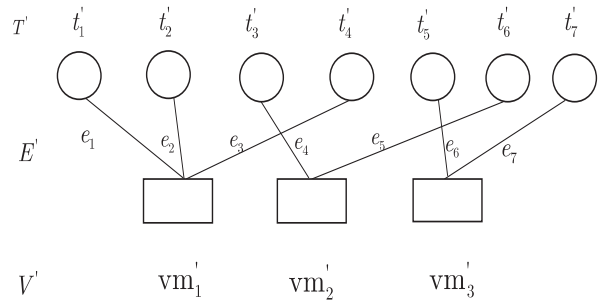

图 1 元任务到虚拟机的放置示意图

图中,元任务集合 T 由七个元任务组成,即 $T' = \{t_1', t_2', t_3', t_4', t_5', t_6', t_7'\}$;虚拟机集合 V' 由三个虚拟机组成,即 $V' = \{\mathrm{vm}_1', \mathrm{vm}_2', \mathrm{vm}_3'\}$;边的集合 E' 由 7 个边组成,即 $E' = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$ 。边代表一个元任务到一个虚拟机的映射,例如边 e_1 代表元任务 t_1' 分配给虚拟机 vm_1',因此元任务分配模型可以用二部图 $G = (T' \times V', E')$ 表示。其中, $r = |T'|, s = |V'|$, T' 是所有元任务的集合, V' 是所有虚拟机的集合, $E' \subseteq T' \times V'$, E' 是 T' 和 V' 之间的边集,一条边 $e(t,v)$ 代表元任务 $t \in T'$ 被调度到虚拟机 $v \in V'$ 上。分配策略是一个函数 $f: T' \to V'$,表示分配一个元任务 t 给虚拟机 $f(t)$,假如对每一个元任务 $t \in T'$, $f(t)$ 都有定义,那么这种分配是一种全局分配。在一种全局分配 α 中,一些虚拟机分配了一些元任务,假定元任务在虚拟机上是顺序执行的,因此一台虚拟机的负载表示这台虚拟机完成上面的所有元任务的总的时间。

假设在初始时刻三台虚拟机的负载如表 1 所示。

表 1　虚拟机初始负载

虚拟机	初始负载
vm_1'	6.2
vm_2'	3.5
vm_3'	1.2

各个元任务在所分配的虚拟机上的执行时间如表 2 所示。

表 2　任务执行时间

元任务	虚拟机	执行时间
t_1'	vm_1'	0.1
t_2'	vm_2'	0.2
t_3'	vm_3'	0.3
t_4'	vm_4'	0.4
t_5'	vm_5'	0.5
t_6'	vm_6'	0.6
t_7'	vm_7'	0.7

那么在一种分配策略下,虚拟机的负载等于虚拟机的初始负载 L_v^{init} 加上该虚拟机执行完其上的所有任务的时间 $L_v^{\text{task}}(\alpha)$,即虚拟机的负载 $L_v(\alpha) = L_v^{\text{init}} + L_v^{\text{task}}(\alpha)$。其中,$L_v^{\text{init}}$ 代表虚拟机 v 在开始时刻的初始负载,$L_v^{\text{task}}(\alpha)$ 代表虚拟机 v 执行完其上的所有元任务的总时间。因此在这种分配策略下,每个虚拟机都对应一个负载,在 α 下,所有元任务的总的完成时间 $\text{makespan}_\alpha = \max L_v(\alpha)$,其中 $v \in V$。由前面的例子可以算出虚拟机 vm_1 上面的负载为 6.1+0.1+0.2+0.4=6.8,虚拟机 vm_2 上面的负载为 3.5+0.3+0.6=4.4,虚拟机 vm_3 上的负载为 1.2+0.5+0.7=2.4。因此在 α 分配下面的虚拟机的完成时间为 6.8。不同的分配方式会得到不同的完成时间,基于上面这些表示方式,任务调度的目标就是在统一调度之前找到一种全局分配使元任务的总完成时间 makespan_α 最小,然后按照这种调度策略分配任务到实际的虚拟机上面执行。

3　任务模糊聚类

模糊 C 均值聚类(FCM)是由 Dumm 第一个提出的,并且由 Bezdek 将其发展和推广。FCM 可以描述为最小化指标函数的优化迭代算法。设集合由 $X = \{x_1, x_2, \cdots, x_n\}$ 是样本集合,把 X 分为 c 类,进一步设定 c 个聚类中心的集合 $V = \{v_1, v_2, \cdots, v_c\}$。

FCM 的目标函数如下:

$$J_{\text{FCM}}(U,V) = \sum_{i=1}^{n}\sum_{j=1}^{c} u_{ij}^m \parallel x_i - x_j \parallel^2 \qquad (1)$$

式中,c、n 是聚类的个数和样本数目;u_{ij} 表示对象 j 相对于集合 i 的隶属度,$\sum_{i=1}^{c} u_{ij} = 1$,$u_{ij}$ [0,1];U 为样本空间 X 的一个 C 划分;m 为加权指数。

文中使用 FCM 算法将用户需求模糊划分为 3 个类别,主要步骤如下:

(1)以任务的需求向量 t 建立初始化样本矩阵。

(2)将原始数据进行标准化和极差标准化处理,将数据压缩到[0,1]之间。

(3)对模糊伪划分矩阵 $\boldsymbol{U}$ 进行初始化。

(4)计算簇质心,并更新模糊伪划分。

(5)如果聚类的目标函数收敛,则算法停止。否则,返回步骤(4)。

由于文中是根据任务对时间需求、带宽需求、存储需求来建模的,经过模糊 C 均值聚类之后,得到每个任务属于时间型任务、带宽型任务、存储型任务的百分比,再根据一定的阈值将任务分配到三种类型的任务集合 S_j 中。表 3 表示任务根据模糊 C 均值聚类得到的每个任务相对应于各个类别的隶属程度,选取其中的 5 个任务,其中类别 1 代表时间型任务,类别 2 代表带宽型任务,类别 3 代表存储型任务。

表 3　类别比例　　%

类别	t_1	t_2	t_3	t_4	t_5
1	2.724 916 5	2.177 143 3	1.654 619 2	2.028 207 3	89.838 57
2	7.782 916 5	6.433 247	5.070 003 5	96.798 004	7.604 486 5
3	89.492 165	91.389 6	93.275 375	1.173 795 3	2.556 951

根据任务相对应于各个类别的隶属程度,文中选定一个阈值范围 $[a,b]$,如果任务的最大占比小于 a,那么将任务分配给类别 1,如果任务的最大占比大于 b,那么将任务分配给类别 3,如果任务的最大占比在阈值 $[a,b]$ 之间,那么将任务分配给类别 2。因此任务按照模糊聚类分为三类,第一类为偏好时间的任务,第二类为偏好带宽的任务,第三类为偏好存储的任务。

4　资源分配

针对上面划分的三个任务类别,对于不同的任务需求采取不同的调度算法,这样能更好地满足用户的个性化需求。对偏好完成时间的任务来说,为其分配较高计算能力的虚拟机能够降低任务的整体完成时间。Min-Min 算法是一种简单的算法,它运行时间相对较短,能取得相对好的性能,然而 Min-Min 算法总是优先调度短任务,造成虚拟机的利用率较低。文中提出的改进的分段 Min-Min 算法主要思想是尽量将长任务分配到处理能力强的虚拟机上,短任务分配到处理能力弱的虚拟机上,这样长任务花费的时间相对会减少。如果将长任务分配到处理能力弱的虚拟机上,那么长任务执行时间会相对拉高整体任务的完成时间。但是如果长任务和短任务的比例不均匀,容易

造成各个虚拟机上面负载的极度不平衡,因此在前面分配的基础上,将负载较重的虚拟机上的任务重新分配到负载较轻的虚拟机上不但能降低任务的总完成时间,而且能平衡各个虚拟机上面的负载。改进的分段 Min-Min 算法分为两个阶段,第一个阶段采用分段 Min-Min,步骤如下:

(1)计算每个任务的关键字:计算每个任务在每台虚拟机上的执行时间,对每个任务对应的所有执行时间求平均值。

(2)将任务按照关键字降序排列。

(3)将所有任务划分为 N 段。

(4)将虚拟机按照处理能力降序排列,也划分为 N 段。

(5)将任务 N 段顺序映射到虚拟机的 N 段上。

(6)每个映射采用 Min-Min 调度算法。

第二阶段将重负载虚拟机的任务分配给轻负载虚拟机。

改进算法将第一阶段取得的 makespan 的虚拟机定义为重负载的虚拟机。

1)将重负载虚拟机上的任务按升序排列。

2)假如将任务集中的第一个任务(也就是最小的任务)分配给其他虚拟机,更新该负载虚拟机及其他虚拟机的完成时间。

(1)如果其他虚拟机的最大完成时间小于 makespan,那么将该任务分配给完成时间最小的虚拟机;

(2)如果其他虚拟机的最大完成时间大于 makespan,那么选择任务集中的其他任务。

若步骤(1)、(2)中有任务分配出去,则更新 makespan。

3)重复步骤 1)、2),直到重负载虚拟机的任务不再分配出去,则算法结束。

对偏好带宽的任务来说,采取轮转调度的方法。该算法的最大优点在于简单易行,但由于任务对于宽带的实际需求不同,采用加权轮转调度算法,这样能根据不同任务的需求分配合适的带宽资源。对偏好存储的任务来说,为每个任务分配一定的优先级,然后按照优先级别高低依次调度各个任务。

5 QOSFCM 调度算法流程图

QOSFCM 算法流程图如图 2 所示。

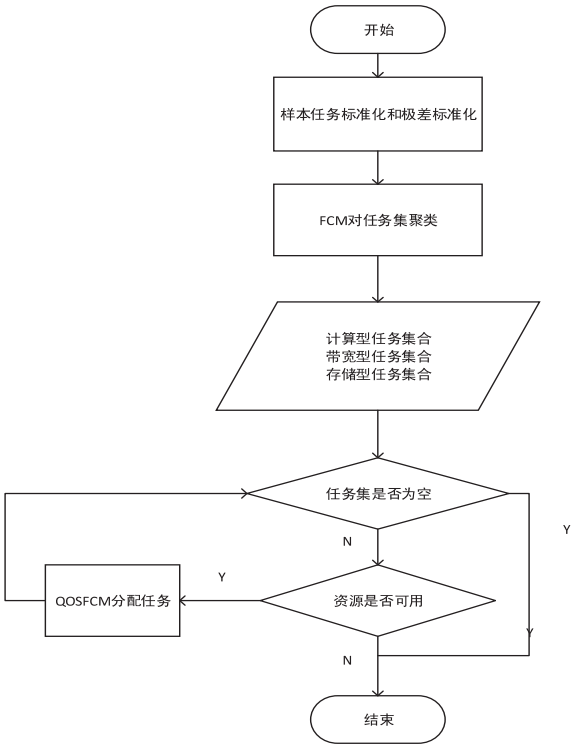

图 2 QOSFCM 算法流程图

6 实验结果与分析

CloudSim[13] 是澳大利亚墨尔本大学 Rajkumar Buyya 博士领导开发的云计算仿真平台,对不同应用和服务模型的调度策略进行性能测试。

伯格模型[14]将社会个体的正义性定义为一个评价函数 JEF = θln(AR/JR)。其中,AR 表示实际分配到的社会财富;JR 表示期望分配到的社会财富;θ 表示均衡性常数。当函数值为 0,认为是正义;函数值大于 0,表示行为者的期待分配小于实际分配,认为非正义;函数值小于 0,表示行为者的期待分配多于实际分配,也认为是非正义的。文中用户满意度采用伯格模型来描述。

用户对云服务的需求为:

$$t_{AP} = \sqrt{\frac{t_1 (t_{Cal})^2 + t_2 (t_{Com})^2 + t_3 (t_{Stor})^2}{t_1 + t_2 + t_3}} \quad (2)$$

其中,t_{AP} 代表用户对云服务的整体性能需求;t_{Cal},t_{Com},t_{Stor} 分别代表用户对云服务的计算能力、通信能力、存储能力的需求;t_1,t_2,t_3 分别代表用户对云服务的计算能力、通信能力、存储能力的需求的经验系数。

图 3 是文中提出的算法相对于经典算法用户满意度的实验图。

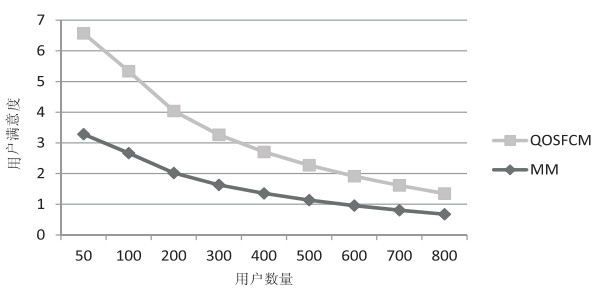

图 3 不同数量的用户满意度

由于 Min-Min 算法只考虑了任务整体的完成时间,没有考虑用户任务的其他性能需求,文中提出的算法很明显优于 Min-Min 算法。

针对用户期望在较短时间内提供服务的特点,在 Segment-Min-Min 的基础上,对任务调度的结果进行二次优化调度,这样在整体上降低了任务的完成时间。

比较 Min-Min(MM)、Min-MinOPT(MMOPT)、Min-MinSegment(MMSEG)、QOSFCM 四种算法,如图 4 ~ 6 所示。前三种传统算法只考虑了用户的总体执行时间,而没有考虑用户对于计算性能、带宽、存储的差异化需求,因而不能很好地提高用户的满意度。QOSFCM 针对用户的不同需求分配相应的虚拟机,不但提高了资源利用率,而且提高了用户满意度,因而此算法有一定程度的优越性。针对前三种算法要么优先考虑短任务,要么优先考虑长任务的特点,文中采用长任务分配高计算能力的虚拟机,短任务分配低计算能力的虚拟机,再在此基础上采取优化的方法对任务调度进行二次调度,能很好地提高全体任务的总的完成时间。实验证明 QOSFCM 在完成时间方面有一定的优越性。

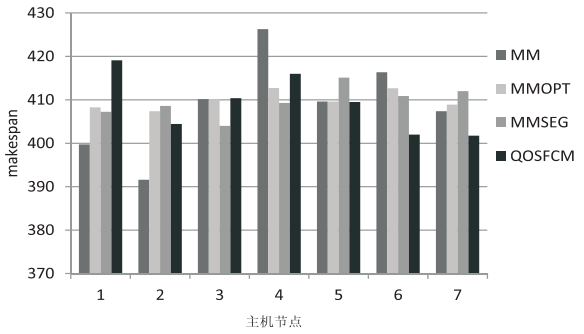

图 4　200 个任务分配到七台虚拟机节点

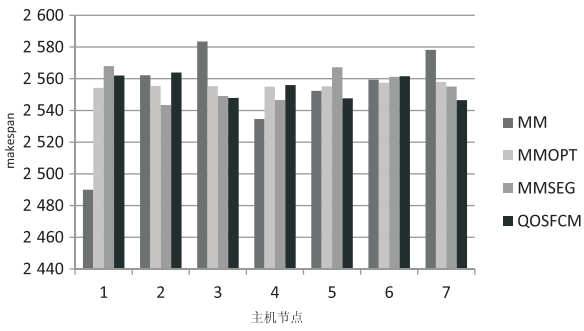

图 5　500 个任务分配到七台虚拟机节点

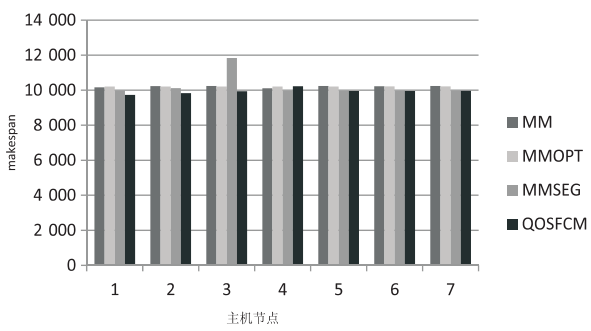

图 6　1 000 个任务分配到七台虚拟机节点

7　结束语

文中提出了一种采用基于模糊聚类的任务调度算法。算法针对不同用户的不同需求采用相应的调度算法,比较实验结果说明,文中提出的 QOSFCM 算法相比传统算法有一定程度的改进。

参考文献:

[1] Buyya R, Broberg J, Goscinski A. 云计算原理与范式[M]. 李红军, 李冬梅, 张秀程, 等, 译. 北京: 机械工业出版社, 2013:1-10.

[2] Armbrust M, Fox A, Griffith R, et al. A view of cloud computing[J]. Communications of the ACM, 2010, 53(4):50-58.

[3] 刘　鹏. 云计算[M]. 北京: 电子工业出版社, 2011:3-9.

[4] Freund R F, Gherrity M, Ambrosius S, et al. Scheduling resources in multi-user, heterogeneous, computing environments with SmartNet[C]//Proc of 7th IEEE heterogeneous computing workshop. [s. l.]: IEEE, 1998:184-199.

[5] Braun T D, Siegal H J, Beck N, et al. A comparison study of static mapping heuristics for a class of meta-tasks on heterogeneous computing systems[C]//Proceedings of eighth heterogeneous computing workshop. [s. l.]: IEEE, 1999:15-29.

[6] Anousha S, Ahmadi M. A new heuristic algorithm for improving total completion time in grid computing[J]. Lecture Notes in Electrical Engineering, 2014, 308:17-26.

[7] Meraji S, Salehnamadi M R. A batch mode scheduling algorithm for grid computing[J]. Journal of Basic and Applied Scientific Research, 2013, 3(4):173-181.

[8] 熊聪聪, 冯　龙, 陈丽仙, 等. 云计算中基于遗传算法的任务调度算法研究[J]. 华中科技大学学报: 自然科学版, 2012(S1):1-4.

[9] 华夏渝, 郑　骏, 胡文心. 基于云计算环境的蚁群优化计算资源分配算法[J]. 华东师范大学学报: 自然科学版, 2010(1):127-134.

[10] 封良良. 云计算环境下基于改进粒子群的任务调度算法[D]. 乌鲁木齐: 新疆大学, 2013.

[11] 李文娟, 张启飞, 平玲娣, 等. 基于模糊聚类的云任务调度算法[J]. 通信学报, 2012, 33(3):146-154.

[12] Miyamoto S, Ichihashi H, Honda K. Algorithms for fuzzy clustering—methods in c - means clustering with applications [M]. Berlin: Springer-Verlag, 2008.

[13] Buyya R, Murshed M. GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing[J]. Concurrency and Computation: Practice and Experience, 2002, 14(13-15):1175-1220.

[14] 赵春燕. 云环境下作业调度算法研究与实现[D]. 北京: 北京交通大学, 2009.