文章编号：1674－9057(2017)02－0360－06 doi:10.3969/j.issn.1674－9057.2017.02.019

基于优先排队论网络延迟云计算资源调度算法

崔建明[a]，刘佳祎[b]，杨呈永[a]

（桂林理工大学 a. 现代教育技术中心；b. 信息科学与工程学院，广西 桂林 541000）

摘 要：针对虚拟机在实际应用环境中，对于不同数据在不同情况下需要不同优先级输出的问题，采用运筹学优先制M/M/1排队模型，对虚拟机请求作出网络延迟分析并对传统的顺序输出方法加以改进。结合数据资源在云计算环境下的Map-Reduce模型，提出新的调度算法以及不同于传统算法的度量指标。经过CloudSim仿真软件进行模拟实验，结果表明，新的算法将网络延迟以及物理机和虚拟机的CPU、内存等资源综合考虑，在性能上要优于随机算法、转轮算法等传统算法，大大改善了负载均衡度，缩短任务调度总时间，并使总调度时间的负载效率得以提高。

关键词：云计算；网络延迟；优先制M/M/1排队模型；资源调度

中图分类号：TP311.5 **文献标志码**：A

0 引 言

随着计算机技术的发展，对计算机的计算能力要求也随之提高，传统的计算模式已无法及时、高效地处理用户提交的任务。在这种趋势下许多大中型企业逐渐用具有云服务资源池的企业云服务模式取代传统模式，例如，亚马逊具有将用户和开发者们所需的存储服务器、带宽和CPU等资源通过云服务模式进行服务的数据中心。国内外研究者通过对云计算展开相关的研究，逐渐形成3个不同层次的服务资源池：基础设施即服务(infrastructure as a service，IaaS)、平台即服务(platform as a service，PaaS)和软件即服务(software as a service，SaaS)，可以预见云服务资源在未来会以大规模化和商业化形式呈现出来[1－2]。不合理的调度方案会导致一部分资源上任务过多，另一部分资源上没有任务、出现空闲，进而造成资源负载的不均衡现象，最终无法在用户的预期时间内完成相应任务，再加上云服务是要付费的，这就使如何保持资源负载均衡高效、省时地完成用户任务成为云计算研究中的一大难题[3－4]。

起初常用的云计算资源调度方法主要有随机算法、轮转算法、散列算法等[5－6]：随机算法是把虚拟机的请求随机地分配到能提供相应资源的物理机上；轮转算法是一种将反映用户需求的请求轮流、有序地分配到提供相应资源物理机上的最为传统的负载均衡调度算法；散列算法是把虚拟机请求的任务在一个提前设置好的散列函数中映射到对应的物理器上。随后，国内外一些研究者提出智能算法，可以依据具体的实际问题建立数学模型，然后求解，这些算法具有自动适应能力和搜索能力，如遗传算法、粒子群算法、蚁群算法等[7－10]。

针对虚拟机的资源调度和相关优化问题，文献[11]针对虚拟机多个属性层次部署提出了一种调度策略的分析方法，其优点在于通过对虚拟机权重向量定量，并记录其使用的服务器资源，预测评估这些服务器，可以有效地解决服务器间负载的不均衡问题，传统的调度方法很难达到整体效益的最优化；文献[12]给出了一种动态调度策略，不仅具有及时性，还可以使运营商资源具有最高效益；Tian[13]引入了一种仅考虑当前分配时刻及多维度资源但未考虑生命周期的动态负载调度算法；

收稿日期：2016－12－09

基金项目：广西高等学校科研项目（YB2014149）

作者简介：崔建明（1962—），硕士，高级实验师，研究方向：云计算与数据挖掘，cjm@glut.edu.cn。

引文格式：崔建明，刘佳祎，杨呈永．基于优先排队论网络延迟云计算资源调度算法［J］．桂林理工大学学报，2017，37(2)：360－365.

Li[14]提出了一项云任务调度策略，该策略基于蚁群优化算法，能均衡系统的负载，更能最小化系统的调度时间。

云计算环境下资源调度是单一调度算法求解中具有局限性的一种 NP-Hard 难题。因此，本文提出了一种基于运筹学优先制 M/M/1 排队模型的网络延迟均衡调度实算法——NLBSA 算法，它将网络延迟、物理机和虚拟机的 CPU、内存等资源均加以考虑，可缩短虚拟机请求的调度时间。

1　云计算环境下的资源调度模型

Map-Reduce[15]是当前很多云计算系统都采用的，由 Google 推出的用于对超过 1 TB 的数据集进行并行运算的一种很受欢迎的云计算分布式编程模型，同时它也是一种任务调度模型。Map-Reduce 具有编程人员在不了解分布式并行编程的情况下，也可以轻松地将相应的程序部署到系统中的矢量编程语言的特性。图 1 描述了 Map-Reduce 调度模型的工作过程。

Map-Reduce 的工作步骤[16]：

（1）Map-Reduce 库将 User Program 划分为 N 个子任务（N 由用户自己定义，每个子任务在 16 ~ 64 MB），如图 1 所示分成了 split0 ~ split4。

（2）Master 为空闲的 Worker 分配 Map 任务或者 Reduce 任务。

（3）Map worker 从对应分片读取输入数据，并解析出相应的键值对（Key/Value），然后传递给 Map 函数，其中将产生的中间键值对分为 R 个区缓存在本地磁盘上。其位置最终会被传输给 Master，Master 负责将这些信息发送给 Reduce worker。

（4）当 Reduce worker 全部读取其负责的相应中间键值对后，由于分区较少，会使不同的键映射到同一个 Reduce 任务，因此需要根据同键值对聚集一起的原则对其进行排序。

（5）排序好的键值对由 Reduce worker 传递给 Reduce 函数，其产生的数据会自动添加到 R 个分区所对应的输出文件中。

（6）在 Map 和 Reduce 任务完成后，Map-Reduce 会把放在这 R 个分区的输出文件对应返回给其用户程序。

在云计算环境中，用户请求的任务相对较多而任务调度算法又过于简单，这使得在执行过多的任务时系统需要提前进行预测。因此合理分配资源，保质、保量完成用户的任务，显得尤为重要。

2　基于具有优先制 M/M/1 排队模型的网络延迟均衡调度

2.1　优先制 M/M/1 排队模型延迟分析

本文把运筹学中的优先制 M/M/1 模型运用到传统排队网络模型中，以减少虚拟机请求网络传输所用的时间，即网络延迟（network latency）。数据的达到流具有不同的 k 个优先级，按照采集数据节点跳数来决定任务节点处理数据的优先级。在所有采集的数据中，设第 1 跳数据为第 1 类分组流，具有最高优先级，它在后续所有任务节点传输的过程中最先发送；第 2 跳的数据为第 2 类分组流，排在第 1 类分组流后等待发送，拥有次高优先级；依次类推，伴随采集跳数的增加，后续节点需要处理的数据也将会越多。不同优先级的划分可以延迟前续节点采集的数据在后续过程的

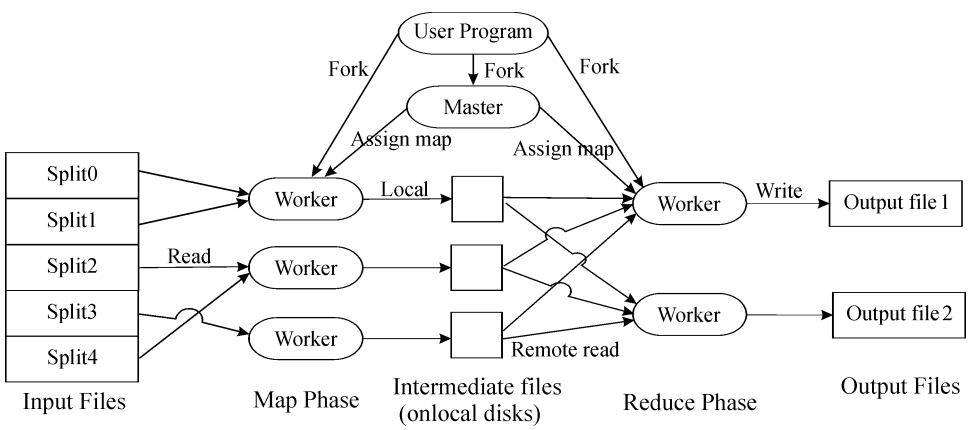


图 1　Map-Reduce 调度模型工作过程

Fig. 1　Working process of Map-Reduce scheduling model

传输效果。

2.1.1 传统的 M/M/1 排队网络模型延迟分析 在无优先级的 M/M/1 队列中，假定第 i 个数据的服务时间是到达间隔相互独立的 P_i。令 $P = \{P_1, P_2, \Lambda\}$，则平均服务时间为 $\overline{P} = \mathrm{E}\{P\} = 1/c$，服务时间的二阶矩为 $\overline{P^2} = \mathrm{E}\{P^2\}$。

设第 i 个数据抵达时，第 j 个数据在传输，X_i 为其剩余服务时间，有 R_i 个数据正在队列里等待。设第 k 个用户服务的时间为 P_k，可得用户 i 的等待时间 W_i 为 $X_i + R_i$ 个用户的服务时间，即

$$X_i + \sum_{k=i-R_i}^{i=1} P_k; \tag{1}$$

$$\overline{W_i} = \mathrm{E}\{X_i\} + \mathrm{E}\{\sum_{k=i-R_i}^{i=1} P_k\} = \mathrm{E}\{X_i\} + \overline{P} \times \mathrm{E}\{R_i\}。 \tag{2}$$

式中：R_i 和 P_k 均为随机变量；$\mathrm{E}\{R_i\}$ 表示平均排队长度，为 R_i 的均值；$\overline{P}$ 为 P_k 的均值。当 i 趋近于无穷时，稳态时队列平均延迟为 $\lim\limits_{i\to\infty}\overline{W_i}$，有

$$W = X + \overline{P}R_Q = X + \frac{1}{\omega}\lambda W = X + \rho W, \tag{3}$$

式中：$X = \lim\limits_{i\to\infty}\mathrm{E}\{X_i\}$，$R_Q = \mathrm{E}\{R_i\}$，$\rho = \lambda/\omega$。

整理上式，得

$$W = X/(1-\rho)。 \tag{4}$$

只需平均剩余服务时间 N 就能获得队列平均的等待时间 W。假定系统各态历经性均有稳态解，则得到 N 在时间 t 趋于无穷时，其值为

$$N = \lim_{t\to\infty}\frac{1}{t}\int_0^t n(t)\,\mathrm{d}t = \lim_{t\to\infty}\frac{1}{t}\sum_{i=1}^{m(t)}\frac{1}{2}P_i^2, \tag{5}$$

式中：$m(t)$ 代表在[0, t] 区间内已服务用户数，$n(t)$ 代表在时间数据流内剩余的传输时间。将上式中的极限与求和用参数 λ 代替，

$$X = \frac{1}{2}\lambda P_i^2, \tag{6}$$

再代入式(3)，可以得到下式

$$W = \frac{X}{1-\rho} = \frac{\lambda\overline{P^2}}{2(1-\rho)}。 \tag{7}$$

普通 M/M/1 队列的传输时间是负指数分布情况，可得 $\overline{P^2} = 2/\omega^2$，代入式(7)，得

$$W = \frac{\rho}{\omega(1-\rho)}。 \tag{8}$$

2.1.2 基于优先制 M/M/1 排队模型网络延迟分析 当系统中仅有一个任务节点，遇到拥有较高优先级的组数据到达时，除了具有同等级别的数据正在被传输需要排队等待，其余情况下，任务节点都将会中断正在传输的较低优先级的数据分组，并且将其重新排回队列中等待，同时任务节点将立即处理当前较高级的数据分组。

1）当最高优先权的数据抵达时，会默认任务节点中只存在第 1 类分流数据，而忽略其他低优先级数据。在传输过程中，每一级别数据的传输时间都是服从于负指数分布的，$1/\omega$ 为平均传输时间。记 $\overline{G_i}$、$\overline{W_i}$ 分别为第 i 级的数据在节点中的平均迟延及排队等待平均时间。结合式(8)及 $\rho_1 = \lambda_1/\omega$，$\lambda = \lambda_1$，$\mathrm{E}(\overline{P^2}) = 2/\omega^2$，可得

$$\overline{W_1} = \frac{\lambda_1}{\omega(\omega-\lambda_1)}; \tag{9}$$

$$\overline{G_1} = \overline{W_1} + \frac{1}{\omega} = \frac{1}{\omega-\lambda_1}。 \tag{10}$$

2）当第 2 级优先权数据分组达到时，由于存在着强占优先权，它们的传输不受第 1 优先级以外影响，可以认为任务节点中只有第 1、2 级优先权数据，记第 1 级和第 2 级数据在节点中传输时的平均时延为 $\overline{G_{1-2}}$，那么易知

$$(\lambda_1+\lambda_2)\overline{G_{1-2}} = \lambda_1\overline{G_1} + \lambda_2\overline{G_2}, \tag{11}$$

整理可得

$$\overline{G_2} = \left(1+\frac{\lambda_1}{\lambda_2}\right)\overline{G_{1-2}} - \frac{\lambda_1}{\lambda_2}\overline{G_1}。 \tag{12}$$

其中，求 $\overline{G_{1-2}}$ 的方法如下：当较高级别的数据抵达时，中断较低级别的数据传输，使其重新等待排队，根据负指数分布的性质，传输用时分布不受前置数据传输及传输耗时多少的影响，依旧为同一参数的负指数分布。

因此，按照达到率为 $\lambda = \lambda_1 + \lambda_2$ 时的 M/M/1 排队模型算出 $\overline{G_{1-2}}$，即

$$\overline{G_{1-2}} = \frac{1}{\omega-(\lambda_1+\lambda_2)}, \tag{13}$$

将其代入式(12)，得

$$\overline{G_2} = \left(1+\frac{\lambda_1}{\lambda_2}\right)\frac{1}{\omega-(\lambda_1+\lambda_2)} - \frac{\lambda_1}{\lambda_2}\frac{1}{\omega-\lambda_1}; \tag{14}$$

$$\overline{W_2} = \overline{G_2} - 1/\omega,。 \tag{15}$$

归纳可得

$$\overline{G_{1-q}} = \frac{1}{\omega - \sum\limits_{i=1}^{q}\lambda_i}; \tag{16}$$

$$\overline{G}_q = \frac{\sum_{i=1}^{q}\lambda_i}{\lambda_q}\overline{G} - \frac{\sum_{i=1}^{q}\lambda_i\overline{W}_i}{\lambda_q}; \tag{17}$$

$$\overline{W}_q = \overline{G}_q - 1/\omega。 \tag{18}$$

上式分别为 1 ~ q 级优先权的数据流平均延时、第 q 优先权的数据平均延时、第 q 级优先权的数据平均等待延时。其证明了基于优先制 M/M/1 的网络排队模型有效地缩短了网络延迟的时间。

2.2　负载均衡调度算法的度量指标

1）PM 资源：$PM_i(i, pCPU_i, pRam_i, pStore_i)$ 其中 i 是 PM 的编号，$pCPU_i$、$pRam_i$、$pStore_i$ 是同一台 PM 所提供的 CPU 值、内存值、存储数值。

2）VM 资源：$VM_j(j, vCPU_j, vRam_j, vStore_j)$ 其中 j 是 VM 的 ID 类型，$vCPU_j$、$vRam_j$、$vStore_j$ 是虚拟机 VM 所需的 CPU 值、内存值、存储数值。

3）时间跨度：假设把 0 ~ T 分为相同时间长度的时隙，那么 k 段时隙则将定义为 $\{(t_1 - t_0), (t_2 - t_1), \cdots, (t_k - t_{k-1})\}$。由此可知 T_n 表示的时间跨度是 $(t_n - t_{n-1})$。

4）某段时间内 PM_i 的平均 CPU 利用率

$$pCPU_i^U = \frac{\sum_{n=0}^{k}(pCPU_i^{T_n} \times T_n)}{\sum_{n=0}^{k} T_n}, \tag{19}$$

其中，$pCPU_i^{T_n}$ 是 $pCPU_i$ 在 T_n 时间段的平均利用率。按照这种方式 PM_i 的内存利用率 $pRam_i^U$ 和储存利用率 $pStore_i^U$ 以及 VM_j 的相关利用率均可以计算。

5）PM_i 的综合负载不均衡度 CLB_i：在统计学中，随机变量分布的离散程度更多的是用标准差来反映：

$$CLB_i = \sqrt{\frac{(Ave_i - CPU_U^A)^2}{3} + \frac{(Ave_i - Ram_U^A)^2}{3} + \frac{(Ave_i - Store_U^A)^2}{3}}。 \tag{20}$$

其中，Ave_i 是一台 PM 所提供的 CPU 值、内存值和存储数值 3 者的平均值，即

$$Ave_i = \frac{pCPU_u^A + pRam_u^A + pStore_u^A}{3}, \tag{21}$$

CPU_u^A、Ram_u^A、$Store_u^A$ 分别是 CPU、内存和存储 3 者的平均利用率。该指标受到文献[17] 中 DRS 负载均衡度量标准的启发。如果 $CLB_i \leqslant 1$，假设主机调度程序运行正常，那么所有 VM 将获得 PM 授予它们的权利；如果 $CLB_i > 1$，那么 PM 将被视为资源不能满足 VM 的权利。

6）capacity-makespan（CM）容量 - 完工总时间：设每个请求只需要一台物理机上的部分容量，则有

$$CM = c_j t_j, \tag{22}$$

其中：c_j 是 VM_j 的 CPU 请求；t_j 是请求 j 的处理时长。

7）新的 makespan（NM）调度总时间：考虑到虚拟机一般采用互联网进行通信，具有网络延时，故将传统调度总时间重新定义为容量 - 完工总时间与网络延时之和。记

$$NM_i = \sum_{i}^{q} c_i t_i + \frac{q}{\omega - \sum_{i=1}^{q}\lambda_i}。 \tag{23}$$

在分配 VM 请求到 PM 的过程中，用 $Q(i)$ 表示分配到 PM_i 的 VM 请求的集合，j 是 $Q(i)$ 中最大负载的虚拟机 ID，则有

$$NM_i = \max_{j \in Q(i)}\left(\sum_{j}^{q} c_j t_j + \frac{q}{\omega - \sum_{j=1}^{q}\lambda_j}\right)。 \tag{24}$$

因此，所有 PM 的 makespan 可用公式表示为

$$makespan = \max_i(NM_i), \tag{25}$$

8）makespan 的负载效率（$skew$）：传统的 $skew$ 用所有计算机平均负载中最小与最大值的商来表示，即

$$skew(M) = \frac{\min_i(Load_i^A)}{\max_i(Load_i^A)}, \tag{26}$$

由此，提出一个不同于传统的衡量 makepan 的指标，即

$$skew(NM) = \frac{\min\sum_{j \in Q(i)}\left(\sum_{j}^{q} c_j t_j + \frac{q}{\omega - \sum_{j=1}^{q}\lambda_j}\right)}{\max\sum_{j \in Q(i)}\left(\sum_{j}^{q} c_j t_j + \frac{q}{\omega - \sum_{j=1}^{q}\lambda_j}\right)}, \tag{27}$$

负载均衡效果的好坏取决于负载均衡效率的大小，这个值越大效果越好。

可以看出，makespan、skew 以及综合负载不均衡度指标是不同于传统衡量指标的主要地方。由于云数据中心具有固定处理时间间隔，提出了一种新型的网络延迟均衡调度算法——NLBSA 算法。

2.3　NLBSA 算法

首先找到具有平均 makespan 最低值的物理机，如果这台物理机的资源不足，则将虚拟机的请求分配到下一个（具有第二小）平均 makespan 最

低值的物理机上，并且提供足够多的资源保证所有请求都不会被拒绝。

具体实现过程的伪代码如下：

1）输入：衡量指标①PM 资源（资源由 i，$pCPU_i$，$pRam_i$，$pStore_i$ 等参数来定义）；

2）输入：衡量指标②VM 请求（请求由（j，$vCPU_j$，$vRam_j$，$vStore_j$ 等参数来定义）；

3）令 d = 0；

4）for 调度队列对头的每一个任务 k，do（for k = form 1 to n do）；

5）对于所有的物理机，根据式（25）找到一个具有最小调度总时间的物理机，标记为 PM-Min；

6）If 请求 k 仍可以共享 PM-Min 的容量，则

7）分配任务给它；

8）else；

9）找到下一个调度总时间最小的物理机；

10）d = d + 1；

11）分配该任务到 PM - d；

12）end if；

13）end for；

14）输出：将每个请求分配到一个 PM，同时分配请求的区间到 PM；

该算法使用了优先级队列数据结构，确保每个 PM 具有 1 个优先级（平均 makespan），每次算法都会取优先级最高的一个，即选取平均 makespan 最小的 PM。在优先级队列排队中 m 个元素排序时间为 $O(m)$，队列的插入和最小值查找需要 $O(long_2^w)$ 步。因此，NLBSA 算法的计算复杂度是 $O(jlong_2^u)$，其中 w 是 VM 请求个数，u 是所需要 PM 的个数。

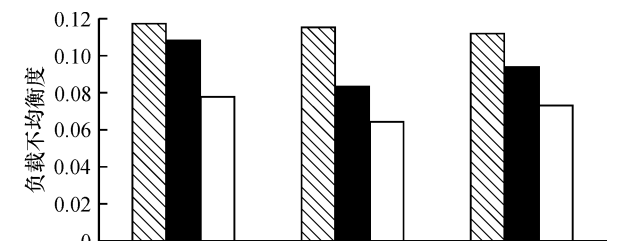
3 实验与结果分析

为了测试 NLBSA 算法在资源调度中的性能，采用 Lawrence Livermore National Laboratory（LLNL）的数据在 CPU 为 3.2 GHz，8 G 内存，windows 7 操作系统的电脑上用 CloudSim Toolkit 3.0 云计算仿真平台进行实验，并与随机算法（Random）和轮转算法（Round）这两种传统算法的模拟结果进行比较。本文在虚拟机数量为 100 和最大持续时间超过 15 个时隙的情况下，不均衡度、makespan、shew of makespan 指标的对比结果如图 2 ~ 4。

图 2 描述了各个算法在改变虚拟机最大持续时间中负载不均衡度的变化情况。对于 Random 算法来讲，无论最大持续时间超过 15 个时隙、30 个时隙，甚至 60 个时隙，它的负载均衡度虽然很稳定但都超过 0.1。对于 Round 算法，它的负载不均衡度不稳定，在超过 30 个时隙时它的负载不均衡度降低。而对于 NLBSA 算法，它的负载不均衡度相当稳定，并且不均衡度均不超过 0.1。

图 3 展示了 3 种算法在虚拟机最大持续时间改变的情况下对总调度时间的反应情况。在算法执行过程中，NLBSA 算法较传统算法有着更短的调度时间，该算法随着持续时间的增大其调度时间也能够平稳的增长，而 Random 算法和 Round 算法在执行过程中随着持续时间的增加完成时间在逐步不稳定增大。

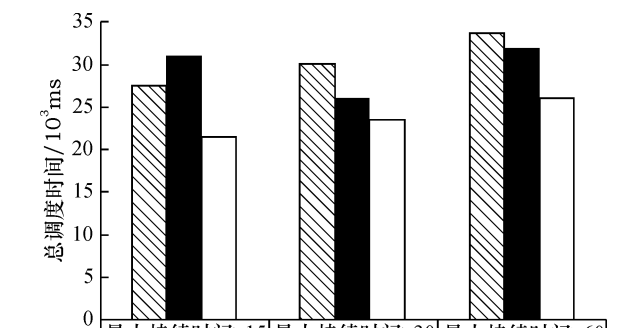
图4阐述了各种算法在不同最大持续时间内总调度时间的负载效率情况。根据本文中提出的新的度量指标可以看出，数值越大负载均衡效果越好。从图表中可以看出 Random 算法和 Round 算法的总调度时间的负载效率无论多长的持续时间均在 0.01 之内，而 NLBSA 算法则随着持续时间的增长，总调度时间的负载效率也随之平稳增加，这



	最大持续时间>15	最大持续时间>30	最大持续时间>60
Random	0.1173	0.115	0.112
Round	0.1082	0.0835	0.094
NLBSA	0.0778	0.0645	0.073

图 2 改变虚拟机最大持续时间下不均衡度对比

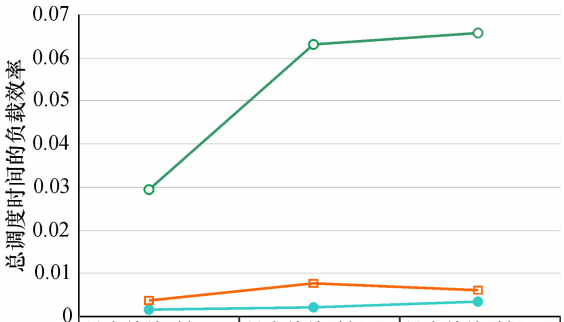
Fig. 2 Contrrast diagram of inbalance degree in changing the maximum duration of the virtual machine



	最大持续时间>15	最大持续时间>30	最大持续时间>60
Random	27 500	30 100	33 750
Round	31 000	26 050	32 000
NLBSA	21 500	23 500	26 050

图 3 改变虚拟机最大持续时间下 makespan 对比

Fig. 3 Contrast diagram of makespan changing in the maximum duration of virtual machine



	最大持续时间>15	最大持续时间>30	最大持续时间>60
Random	0.001 5	0.002 2	0.003 4
Round	0.003 6	0.007 8	0.006 2
NLBSA	0.029 5	0.063 1	0.065 7

图 4　改变虚拟机最大持续时间下 shew of makespan 对比

Fig. 4　Contrast of shew of makespan changing in maximum duration of virtual machine

说明该算法在负载均衡方面的优越性。

4　结　论

在分析当前云计算资源调度研究现状的基础上，提出一种基于优先制 M/M/1 排队模型的网络延迟均衡调度算法——NLBSA 算法。本文将排队论运用到网络中，缩短了网络延时，并且提出新的 makespan、shew 指标。实验模拟结果表明：NLBSA 算法将物理机、虚拟机、CPU、内存等因素综合考虑，用于解决虚拟机请求的调度问题和容量共享，并且在 makespan、shew of makespan 和不均衡度等指标上有较好的性能。

参考文献：

[1] 童晓渝，张云勇，房秉毅，等. 电信运营商实施云计算的策略建议 [J]. 信息通信技术，2012，6（1）：34－38.

[2] 袁文成，朱怡安，陆伟. 面向虚拟资源的云计算资源管理机制 [J]. 西北工业大学学报，2010，28（5）：704－708.

[3] 游新冬，徐向华，万键，等. 虚拟环境下基于市场机制的资源分配方法 [J]. 华中科技大学学报（自然科学版），2010，38（6）：48－51.

[4] Mell P，Grance T. The NIST definition of cloud computing [J]. Communications of the ACM，2010，53（6）：50－55.

[5] 田文洪，赵勇. 云计算资源调度管理 [M]. 北京：国防工业出版社，2011：80－100.

[6] 杨琴，周国华. 服务资源智能调度算法及其应用 [M]. 北京：科学出版社，2014：70－90.

[7] Dean J，Ghemawat S. Map-Reduce：simplified data processing on large clusters [J]. Communications of the ACM，2012，51（1）：107－113.

[8] 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），2010，11（1）：127－134.

[9] 廖周宇，谢晓兰，刘建明. 云计算环境下基于 SVM 的数据分类 [J]. 桂林理工大学学报，2013，33（4）：765－769.

[10] 朱泽民，张青. 基于多维 QoS 和云计算的资源负载均衡调度研究[J]. 计算机测量与控制，2013，21(1)：263－265.

[11] 庄威，桂小林，林建材，等. 云环境下基于多属性层次分析的虚拟机部署与调度策略 [J]. 西安交通大学学报，2013，47（2）：28－32.

[12] 尹红军，李京，宋浒，等. 云计算中运营商效益最优的资源分配机制 [J]. 华中科技大学学报（自然科学版），2011，39（S1）：51－55.

[13] Dabbagh M，Hamdaoui B，Guizani M，et al. Energy-efficient resource allocation and provisioning framework for cloud data centers [J]. IEEE Transactions on Network & Service Management，2015，12（3）：377－391.

[14] Li K，Xu G C，Zhao G Y，et al. Cloud task scheduling based on load balancing ant colony optimization [C] //Sixth Chinagrid Conference. IEEE，2011：3－9.

[15] Li J Y，Meng L K，Wang F Z，et al. A map-reduce-enabled SOLAP cube for large-scale remotely sensed data aggregation [J]. Computers and Geosciences，2014，70(9)：110－119.

[16] 董西成. Hadoop 技术内幕：深入解析 Map Reduce 架构设计与实现原理[M]. 北京：机械工业出版社，2013：26－40.

[17] Emeakaroha V C，Maurer M，Stern P，et al. Managing and optimizing bioinformatics workflows for data analysis in clouds [J]. Journal of Grid Computing，2013，11(3)：407－428.

Resource scheduling algorithm based on priority queuing and network delay in cloud computation

CUI Jian-ming[a]，LIU Jia-yi[b]，YANG Cheng-yong[b]

(a. Modern Educational Technology Center；b. College of Information Science and Engineering，Guilin University of Technology，Guilin 541004，China)

Abstract：In virtual machine application，as different data under different circumstances require different priority output，from the operation research system of M/M/1 queue model，network latency analysis to virtual machine request output method is improved and so the traditional order. The combinations of resources in cloud computing environment Map-Reduce model put forward a new scheduling algorithm，different from traditional metrics of the algorithm. After Cloudsim simulation software simulation experiment，the results prove that the new algorithm synthesized considers network latency as well as physical and virtual machine CPU，memory and other resources into account. Its performance is better than traditional algorithms，such as random algorithm，wheel algorithm. It greatly improves the balancing degree，speeds up the task scheduling time and the loading time.

Key words：cloud computing；network delay；priority queuing M/M/1 model；resource scheduling