第 42 卷 第 6A 期
2015 年 6 月

计 算 机 科 学
Computer Science

Vol. 42 No. 6A
June 2015

云计算中基于序贯博弈的任务调度策略

刘美林 王 勇 李 凯 刘鹏飞 任兴田 杨建红

(北京工业大学计算机学院 北京 100124)

摘 要 随着互联网应用的普及,云计算逐渐成为企业、学者等各界研究的热点。云计算是继分布式计算、并行计算、网格计算之后的一种新兴计算模式。在对云计算系统的研究中,任务调度是其研究的重点。在总结了云计算的研究现状之后,提出了一种基于序贯博弈的任务调度策略,在任务的响应时间上有较好的优化。

关键词 云计算,序贯博弈,纳什均衡,任务调度

中图法分类号 TP312 文献标识码 A

Task Scheduling Strategy in Cloud Computing Based on Sequential Game

LIU Mei-lin WANG Yong LI Kai LIU Peng-fei REN Xing-tian YANG Jian-hong

(School of Computer Science, Beijing University of Technology, Beijing 100124, China)

Abstract With the popularity of Internet applications, cloud computing has become the hot spots in all sectors of business and academics. Cloud computing is a new computing model after distributed computing, parallel computing and grid computing. In the study of cloud computing, task allocation is the focus of the research. In this paper, after summarizing the research status of cloud computing, we proposed a task scheduling strategy based on sequential game theory, which has a better optimization in resonse time of the task.

Keywords Cloud computing, Sequential game, Nash equilibrium, Task schedule

1 引言

近年来,云计算的研究成为计算机科学领域的一个研究热点[1]。云计算并不是一种全新的概念,它是分布式计算、并行计算和网格计算的进一步发展,是一种新兴的计算模式。它是基于互联网的计算,能够向各种互联网应用提供硬件服务、基础架构服务、平台服务、软件服务、存储服务的系统[2]。任务调度是云计算的关键技术之一,其虚拟化技术使得它的任务调度不同于以往的并行分布式计算。

传统分布式系统的任务调度策略可以分为静态的和动态的[3]。在静态算法中,任务调度策略在系统初始化时便确定下来,并在系统运行期间保持不变。例如,在文献[4]中,Kim 和 Kameda 提出了一种负载均衡的任务调度算法,该算法通过调整各个计算节点的负载量来优化任务的平均响应时间。静态算法的一个缺点就是假设系统中计算资源和通信网络的各因素参数是可提前预知的,并保持不变[5]。相反地,在动态任务调度算法[6,7]中,是根据系统运行时的动态信息来决定任务调度方案的。显然,动态调度算法会有较好的优化效果,但静态调度算法更易于实现。

目前为止,许多学者已经对云计算环境下的任务调度作了研究,并取得了一定的成果。这些算法大致可以分为基于时间的、基于 QoS 的和基于经济效益的 3 大类。已有的基于时间优化的任务调度算法有 Min-Min、Max-Min 算法[8,9]、遗传算法[10]等。研究最广泛的是基于 QoS 的任务调度算法。文献[11-13]都是基于 QoS 的任务调度算法。文献[14-16]提出了一些基于经济效益优化的任务调度算法

在这些调度算法中,很少有应用到博弈论解决问题的。因此,我们使用序贯理性假设,把任务调度问题建模为一个多阶段序贯博弈,把调度器的响应时间作为博弈问题的优化目标,基于序贯博弈给出了任务调度策略。

2 任务调度系统模型

根据云计算系统的特点,以下两点假设是合理的:

1)调度器对任务进行分解以后得到任务分片,假定每个计算节点都具备任务分片的执行能力,因为在一个云计算系统中,节点的配置是可控的。

2)目前,在云计算系统中,任务一次运行的代价通常较大(如执行时间较长),且用户提出的 QoS 要求因素很多(如可靠性、费用、执行时间等)。本模型中主要考虑任务的执行时间。由于云计算系统覆盖的地理范围可能较大,任务分片在网络上的传输时间较长,因此可以忽略调度器的内部处理代价(如进行任务分片所需要的时间),假设任务分片的执行代价和任务分片在网络上的传输代价是任务执行代价的关键所在。

根据以上的假设,任务调度的系统模型如图 1 所示。

在该系统模型中,假定有 l 个用户、n 个调度器、m 个执行任务的计算节点,并且第 i 个调度器可将用户的请求分解为 m 个任务分片——活动分片。

刘美林(1989—),女,硕士,主要研究方向为网格计算、云计算;王 勇(1974—),男,副教授,硕士生导师,主要研究方向为可信计算、服务组合、网格计算、云计算;李 凯(1988—),男,硕士,主要研究方向为网格计算、云计算。

．用户：产生对调度器的请求（任务），各个用户之间彼此独立地产生任务，用户 k 产生任务的平均速率为 β_k，且服从泊松分布；用户产生的任务被调度器发送到计算节点上执行。

．调度器：从用户接受任务，把任务分解为活动分片，然后把活动分片分配给云底层的计算节点执行，基于前面的假设 2），对任务进行分解的时间忽略不计。

．任务分片：根据计算节点的数量，调度器 i 将用户的请求分解为 m 个任务分片，a_{ij} 为调度器 i 的第 j 个任务分片，满足以下约束：

$$a_{ij} \geqslant 0 \text{ 且 } \sum_{j=1}^{m} a_{ij} = 1 \tag{1}$$

．计算节点：任务的执行者，基于前面的假设 1），云计算系统中的计算节点具备一般性任务的执行能力。任务分片在计算节点 j 上的平均执行速率为 u_j，执行时间可以服从任意分布，每一个计算节点 j 可以被看作一个具有一般重试时间和服务器崩溃的 M/G/1 排队系统[17,18]。

设 λ_i 为调度器 i 发出任务的平均速率，则满足式（2）和式（3）两个限制条件。式（2）的含义是到达所有调度器的任务的平均速率的加和应该小于所有计算节点对任务的平均执行速率的加和。式（3）的含义是到达计算节点 j 的活动分片的速率的加和应该小于 j 的任务执行速率。式（2）和式（3）是显然的。

$$\sum_{i=1}^{n} \lambda_i < \sum_{j=1}^{m} u_j \tag{2}$$

$$\sum_{i=1}^{n} \lambda_i a_{ij} < u_j \tag{3}$$

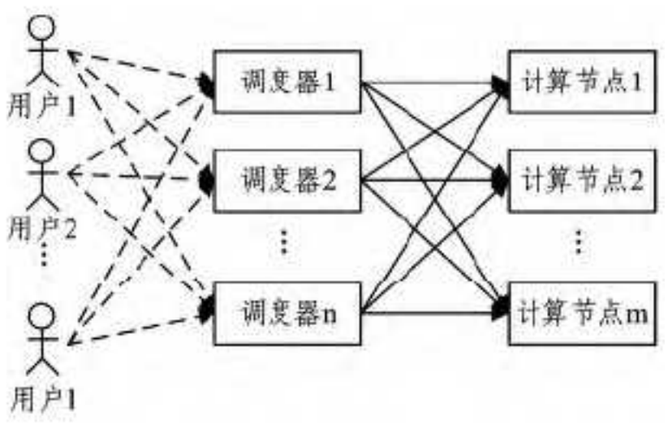

图 1　任务调度的系统模型

3　任务调度中的纳什均衡

云计算系统中各个调度器共享计算节点，彼此独立，总是期望其执行任务的响应时间最短。

调度器 i 的任务分解策略为 $a_i = \{a_{i1}, \cdots, a_{ij}, \cdots, a_{im}\}$，在博弈中，每个调度器期望自己对一次任务执行的时间最短，以一次任务执行的响应时间作为博弈的目标优化函数。一次任务执行的响应时间由各个活动分片的处理时间决定，要确定任务执行的响应时间，首先需要考察每个活动分片的处理时间。

活动分片的处理时间由两部分组成：活动分片 a_{ij} 从调度器 i 到计算节点 j 的传输时间和在计算节点 j 的排队时间和执行时间。

假定所有任务的平均数据长度为 b 位，e_{ij} 为调度器 i 到计算节点 j 之间线路传输延迟，而 c_{ij} 为调度器 i 到计算节点 j 之间线路的传输速率，则活动分片 a_{ij} 的传输时间如式（4）所示。

$$L_{ij} = e_{ij} \times a_{ij} + \frac{b \times a_{ij}}{c_{ij}} \tag{4}$$

计算节点可认为是一个 M/G/1 排队系统，活动分片 a_{ij} 在计算节点 j 上的平均处理时间如式（5）所示。

$$F_{ij} = \overline{h_j} \times a_{ij} + \frac{a_{ij} \times \overline{h_j^2} \times \sum_{i=1}^{n} \lambda_i a_{ij}}{1 - \overline{h_j} \times \sum_{i=1}^{n} \lambda_i a_{ij}} \tag{5}$$

其中，$\overline{h_j} = \frac{1}{u_j}$ 为计算节点 j 任务执行时间的均值，$\overline{h_j^2}$ 为执行时间的方差，λ_i 为任务到达调度器 i 的平均速率。

活动分片 a_{ij} 的总处理时间为在计算节点 j 上的处理时间和在线路上的传输时间的加和，及活动分片 a_{ij} 的总处理时间如式（6）所示。

$$\begin{aligned} F_{ij} + L_{ij} &= \overline{h_j} \times a_{ij} + \frac{a_{ij} \times \overline{h_j^2} \times \sum_{i=1}^{n} \lambda_i a_{ij}}{1 - \overline{h_j} \times \sum_{i=1}^{n} \lambda_i a_{ij}} + \\ & e_{ij} \times a_{ij} + \frac{b \times a_{ij}}{c_{ij}} \end{aligned} \tag{6}$$

定义 $u_{ji} = \frac{1}{\overline{h_j}} - \sum_{\substack{k=1 \\ k \neq i}}^{n} \lambda_k a_{kj}$ 为计算节点 j 为调度器 i 提供的计算能力，代入式（6）可得式（7）。

$$\begin{aligned} F_{ij} + L_{ij} &= \overline{h_j} \times a_{ij} + \frac{a_{ij} \times \overline{h_j^2} \times (\lambda_i a_{ij} + \frac{1}{\overline{h_j}} - u_{ji})}{\overline{h_j} \times (u_{ji} - \lambda_i a_{ij})} + \\ & e_{ij} \times a_{ij} + \frac{b \times a_{ij}}{c_{ij}} \end{aligned} \tag{7}$$

根据以上的推导可知，各个调度器的平均响应时间可由式（8）表示，即需要优化的目标函数，我们的优化目标是使调度器的平均响应时间最短。

$$\begin{aligned} T_i &= \sum_{j=1}^{m} (F_{ij} + L_{ij}) \\ &= \sum_{j=1}^{m} [\overline{h_j} \times a_{ij} + \frac{a_{ij} \times \overline{h_j^2} \times (\lambda_i a_{ij} + \frac{1}{\overline{h_j}} - u_{ji})}{\overline{h_j} \times (u_{ji} - \lambda_i a_{ij})} + e_{ij} \times a_{ij} + \\ & \frac{b \times a_{ij}}{c_{ij}}] \end{aligned} \tag{8}$$

因为式（8）对于每个策略分量 a_{ij} 具有正的一阶、二阶偏导数，其中，

$$\frac{\partial T_i}{\partial a_{ij}} = \overline{h_j} + \frac{\overline{h_j^2} u_{ji}}{\overline{h_j}^2 (u_{ji} - \lambda_i a_{ij})^2} - \frac{\overline{h_j^2}}{\overline{h_j}} + e_{ij} + \frac{b}{c_{ij}} > 0$$

$$\frac{\partial^2 T_i}{\partial a_{ij}^2} = \frac{2\lambda_i \overline{h_j^2} u_{ji}}{\overline{h_j}^2 (u_{ji} - \lambda_i a_{ij})^3} > 0$$

所以，式（8）是连续、递增的凸函数，存在唯一的纳什均衡解使得式（8）最小。

上述优化问题符合 Kuhn-Tucker 条件，取拉格朗日函数如式（9）所示。

$$\begin{aligned} L = \sum_{j=1}^{m} [\overline{h_j} \times a_{ij} + \frac{a_{ij} \times \overline{h_j^2} \times (\lambda_i a_{ij} + \frac{1}{\overline{h_j}} - u_{ji})}{\overline{h_j} \times (u_{ji} - \lambda_i a_{ij})} + e_{ij} \times a_{ij} + \\ \frac{b \times a_{ij}}{c_{ij}} - \alpha \times a_{ij}] + \alpha \end{aligned} \tag{9}$$

令 $\frac{\partial L}{\partial a_{ij}} = 0$ 可得式（10）。

$$\overline{h_j} + \frac{\overline{h_j^2} u_{ji}}{\overline{h_j}^2 (u_{ji} - \lambda_i a_{ij})^2} - \frac{\overline{h_j^2}}{\overline{h_j}} + e_{ij} + \frac{b}{c_{ij}} - \alpha = 0 \tag{10}$$

对式（10）求解，可得优化解 a_{ij} 的求解公式如式（11）所示，满足约束（1）、（2）、（3）。

$$a_{ij}=\frac{u_{ji}}{\lambda_i}-\frac{\sqrt{\overline{h_j^2}u_{ji}}}{\lambda_i\overline{h_j}\sqrt{\alpha+\frac{\overline{h_j^2}}{\overline{\overline{h_j}}}-\overline{h_j}-e_{ij}-\frac{b}{c_{ij}}}} \tag{11}$$

其中，α 的取值由式(11)和约束(1)得到的式(12)决定。

$$\sum_{j=1}^{m}\left(\frac{u_{ji}}{\lambda_i}-\frac{\sqrt{\overline{h_j^2}u_{ji}}}{\lambda_i\overline{h_j}\sqrt{\alpha+\frac{\overline{h_j^2}}{\overline{\overline{h_j}}}-\overline{h_j}-e_{ij}-\frac{b}{c_{ij}}}}\right)=1 \tag{12}$$

由式(11)可得，当不等式(13)满足时，$a_{ij}<0$，此时令 $a_{ij}=0$。

$$\alpha<\frac{\overline{h_j^2}}{u_{ji}\overline{\overline{h_j^2}}}-\frac{\overline{h_j^2}}{\overline{\overline{h_j}}}+\overline{h_j}+e_{ij}+\frac{b}{c_{ij}} \tag{13}$$

考虑到结果的紧致性，可得纳什均衡解的求解算法如下：

1）令 $\tau_{ij}=\frac{\overline{h_j^2}}{u_{ji}\overline{\overline{h_j^2}}}-\frac{\overline{h_j^2}}{\overline{\overline{h_j}}}+\overline{h_j}+e_{ij}+\frac{b}{c_{ij}}$，并对计算节点按照 τ_{ij} 从小到大的顺序排序 $\tau_{i1}<\tau_{i2}<\cdots<\tau_{im}$；

2）如果 $j>d_i$，则 $a_{ij}=0$，其中 $d_i(1\leqslant d_i\leqslant m)$ 为满足不等式(15)的最大整数；

3）否则，按照式(11)求解 a_{ij}，其中 $1\leqslant j\leqslant d_i$，$\alpha$ 的取值由式(14)决定，d_i 为满足不等式(15)的最大整数；

$$\sum_{j=1}^{d_i}\left(\frac{u_{ji}}{\lambda_i}-\frac{\sqrt{\overline{h_j^2}u_{ji}}}{\lambda_i\overline{h_j}\sqrt{\alpha+\frac{\overline{h_j^2}}{\overline{\overline{h_j}}}-\overline{h_j}-e_{ij}-\frac{b}{c_{ij}}}}\right)=1 \tag{14}$$

$$\sum_{j=1}^{d_i}\left(\frac{u_{ji}}{\lambda_i}-\frac{\sqrt{\overline{h_j^2}u_{ji}}}{\lambda_i\overline{h_j}\sqrt{\alpha+\frac{\overline{h_j^2}}{\overline{\overline{h_j}}}-\overline{h_j}-e_{ij}-\frac{b}{c_{ij}}}}\right)\leqslant 1 \tag{15}$$

4）返回 1），直至计算出各参与者的最优策略。

4 基于序贯博弈的任务调度策略

由式(11)可以看到，活动分片 a_{ij} 的求解公式中包含了计算节点 j 为调度器 i 提供的计算能力 u_{ji}，其值可以根据计算节点 j 执行理事的均值来求取，不同的 u_{ji} 值可以产生不同的均衡解 a_{ij}。本文提出了基于序贯博弈来预测 u_{ji} 的任务调度策略，使用了序贯理性假设，即博弈的参与者总在每一个时间点最优化自己的策略。

4.1 任务调度的序贯博弈模型

在序贯博弈中，博弈参与者根据上一次的博弈结果，预测下一次博弈中的 u_{ji} 的值，不断调整自己的任务分片策略，最终达到一种均衡的状态。令 G 为序贯博弈的阶段博弈，$G(n)$ 为第 n 阶段的博弈，此阶段的活动分片记为 $a_{ij}^{G(n)}$，此阶段 α 的取值记为 $\alpha^{G(n)}$，而 u_{ji} 的值记为 $u_{ji}^{G(n)}$。序贯博弈模型的数据流如图 2 所示。

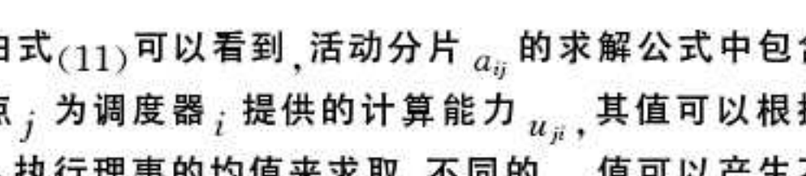

图 2 序贯博弈模型的数据流图

由式(11)可得序贯博弈中 $a_{ij}^{G(n)}$ 的取值由式(16)所示，其中 $n\geqslant 1$。

$$a_{ij}^{G(n)}=\frac{u_{ji}^{G(n-1)}}{\lambda_i}-\frac{\sqrt{\overline{h_j^2}u_{ji}^{G(n-1)}}}{\lambda_i\overline{h_j}\sqrt{\alpha^{G(n-1)}+\frac{\overline{h_j^2}}{\overline{\overline{h_j}}}-\overline{h_j}-e_{ij}-\frac{b}{c_{ij}}}} \tag{16}$$

其中，$u_{ji}^{G(n-1)}=\frac{1}{\overline{\overline{h_j}}}-\sum_{\substack{k=1\\k\neq i}}^{n}\lambda_k a_{kj}^{G(n-1)}$，$\alpha^{G(n-1)}$ 的取值由式(17)决定。

$$\sum_{j=1}^{d_i}\left[\frac{u_{ji}^{G(n-1)}}{\lambda_i}-\frac{\sqrt{\overline{h_j^2}u_{ji}^{G(n-1)}}}{\lambda_i\overline{h_j}\sqrt{\alpha^{G(n-1)}+\frac{\overline{h_j^2}}{\overline{\overline{h_j}}}-\overline{h_j}-e_{ij}-\frac{b}{c_{ij}}}}\right]=1 \tag{17}$$

其中，d_i 的取值为满足不等式(18)的最大整数。

$$\sum_{j=1}^{d_i}\left[\frac{u_{ji}^{G(n-1)}}{\lambda_i}-\frac{\sqrt{\overline{h_j^2}u_{ji}^{G(n-1)}}}{\lambda_i\overline{h_j}\sqrt{\alpha^{G(n-1)}+\frac{\overline{h_j^2}}{\overline{\overline{h_j}}}-\overline{h_j}-e_{ij}-\frac{b}{c_{ij}}}}\right]\leqslant 1 \tag{18}$$

令 $\tau_{ij}=\frac{\overline{h_j^2}}{u_{ji}^{G(n-1)}\overline{h_j}^2}-\frac{\overline{h_j^2}}{\overline{\overline{h_j}}}+\overline{h_j}+e_{ij}+\frac{b}{c_{ij}}$，并对计算节点按照 τ_{ij} 从小到达的顺序排序 $\tau_{i1}<\tau_{i2}<\cdots<\tau_{im}$。

4.2 基于序贯博弈的任务调度算法

在求得各个参与者(调度器)最优调度策略的基础上，得到任务调度算法的流程如图 3 所示。

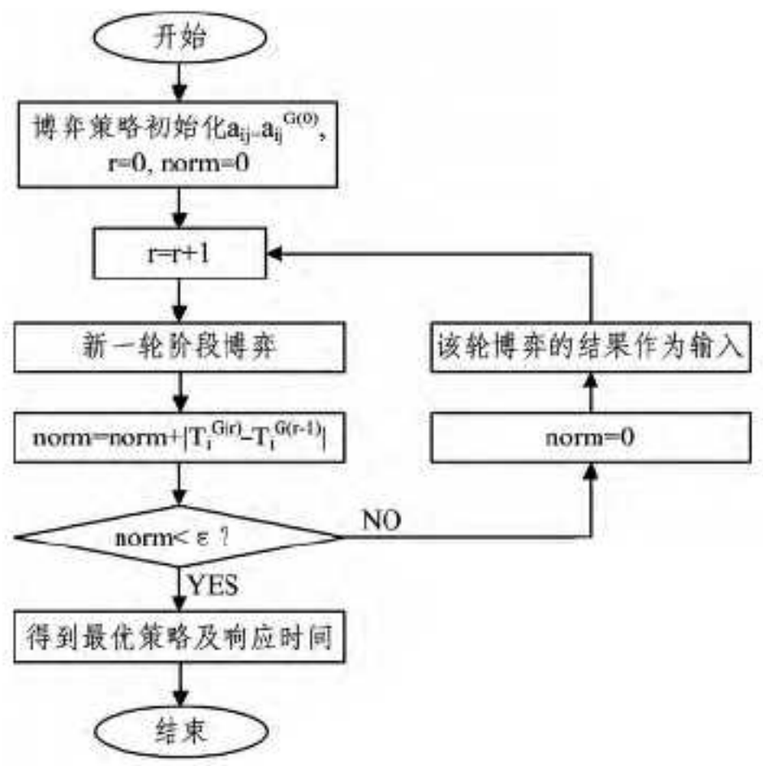

图 3 任务调度算法流程

1）初始化，使得每个参与者的初始策略为 $a_i^{G(0)}=\{a_{i1}^{G(0)},\cdots,a_{ij}^{G(0)},\cdots,a_{im}^{G(0)}\}=\{\frac{1}{m},\cdots,\frac{1}{m},\cdots,\frac{1}{m}\}$，同时定义博弈阶段计数器 r；

2）由式(16)求得参与者 i 的策略 $a_i^{G(r)}$，并由式(8)求得响应时间 $T_i^{G(r)}$，同时求取 $norm=norm+|T_i^{G(r)}-T_i^{G(r-1)}|$，将 $norm$ 传递给下一个参与者 $i+1$；

3）重复 2），直至计算出所有参与者的策略；

4）判断 $norm$ 是否满足预定的误差要求，如满足，结束算法执行，得到各参与者的最优策略及最优策略下的响应时间；如不满足要求，$norm=0$，转到 2）继续执行。

5 实验及其实验结果分析

5.1 实验设计

为方便比较实验结果，标记 4.2 节的算法为 GSA 算法，引用文献[19]中提到的均衡任务调度算法(标记为 BSA)作为实验结果的比较对象。BSA 算法按照式(19)对任务进行分解。

$$a_{ij}=\frac{u_{ji}}{\sum_{j=1}^{m}u_{ji}} \tag{19}$$

应用 4.2 节的最优解的求解方法，我们分别做了均衡状态下各参与者的响应时间、算法的公平性两个实验。

5.2 实验结果分析

1)各参与者的响应时间实验

假定在该云计算系统中有8个计算节点,7个调度器。其中,每个计算节点的任务平均处理速率为u_j。在该实验中,假设每个计算节点的任务平均处理速率服从指数分布。每个调度器的相对任务到达速率为ϕ_i,实际的任务到达速率λ_i按照式(20)计算。其中ρ为负载系数。

$$\lambda_i=\phi_i\cdot\rho\cdot\sum_{j=1}^{m}u_j \tag{20}$$

假定任务的平均长度为$b=1\text{Mbit}$,连接线路的平均带宽为$c_{ij}=100\text{Kbps}$,连接线路的平均延迟为$e_{ij}=0.5\text{s}$,系统的负载系数$\rho=0.5$,调度器的相对任务到达速率如表1所列。

表1 调度器的相对任务到达速率

调度器	1	2	3—4	5	6—7
相对任务到达速率(任务/s)	0.0035	0.002	0.001	0.0006	0.0005

第一组实验为计算节点中有部分节点的计算能力较强的实验,设系统中计算节点的任务平均处理速率如表2所列。

表2 计算节点的任务平均处理速率(部分节点计算能力较强)

计算节点	1	2	3	4
任务平均处理速率(任务/s)	0.28	0.22	0.19	0.23
计算节点	5	6	7	8
任务平均处理速率(任务/s)	0.33	0.26	0.22	0.19

在上述初始条件下,应用GSA算法求得调度策略后,代入式(8)可以求得采用GSA算法的各个参与者的响应时间;通过BSA算法求得调度策略后,同样代入式(8)可以求得采用BSA算法的各个参与者的响应时间。两种算法下各个参与者的响应时间如表3所列。

表3 部分节点计算能力较强时两种算法的响应时间

时间/调度器 / 算法	1	2	3	4	5	6	7
GSA	13.61	13.61	13.61	13.61	13.61	13.61	13.61
BSA	14.68	14.68	14.68	14.68	14.68	14.68	14.68

两种算法中各个参与者的响应时间对比图如图4所示。从图中可以看出,采用GSA算法的情况下各个参与者的响应时间要小于采用BSA算法的情况下各个参与者的响应时间,说明在云计算系统中,部分节点的计算能力较强时,GSA算法的效率要优于BSA算法。

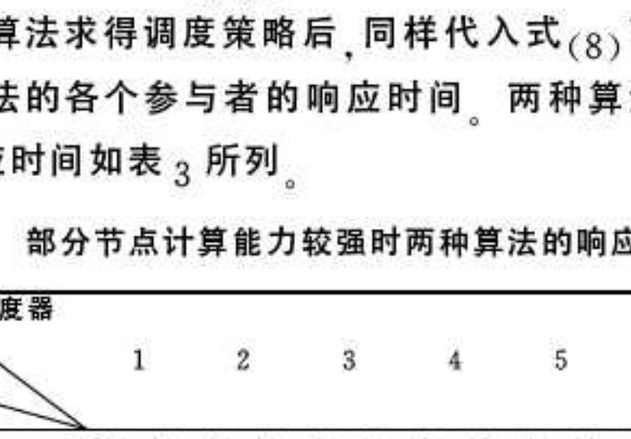

图4 各个参与者的响应时间(部分节点计算能力较强)

第二组实验为计算节点的计算能力均衡的实验,设系统中各计算节点的任务平均处理速率如表4所列。其余各参数同第一组实验中的数据。

表4 计算节点的任务平均处理速率(节点的计算能力均衡)

计算节点	1	2	3	4	5	6	7	8
任务平均处理速率(任务/s)	0.25	0.26	0.27	0.25	0.24	0.26	0.28	0.24

应用GSA算法求得调度策略后,代入式(8)求得各参与者的响应时间;通过BSA算法求得调度策略后,同样代入式(8)求得各参与者的响应时间。两种算法下,各参与者的响应时间如表5所列。

表5 节点计算能力均衡时两种算法的响应时间

时间/调度器 / 算法	1	2	3	4	5	6	7
GSA	10.54	10.54	10.54	10.54	10.54	10.54	10.54
BSA	10.82	10.82	10.82	10.82	10.82	10.82	10.82

两种算法中各个参与者的响应时间对比如图5所示。从图中可以看出,采用GSA算法的情况下各个参与者的响应时间要小于采用BSA算法的情况下各个参与者的响应时间,说明在云计算系统中,各计算节点的计算能力均衡时,GSA算法的效率要优于BSA算法。

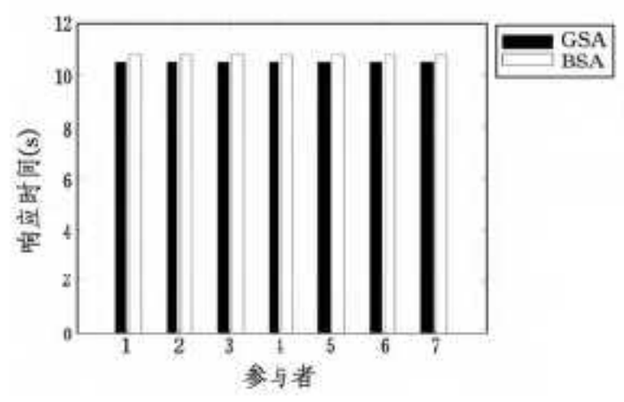

图5 各个参与者的响应时间(节点计算能力均衡)

2)算法公平性实验

算法的公平性主要考察采用某个算法的情况下,各个参与者的响应时间的差异,如果差异较小,则说明算法的公平性较好;反之,则说明算法的公平性较差。算法的公平性是考察算法优劣的另一个重要方面。

采用式(21)所示的公平性索引作为衡量算法公平性的指标。

$$FI=\frac{(\sum_{i=1}^{n}T_i)^2}{n\sum_{i=1}^{n}T_i^2} \tag{21}$$

以各参与者的响应时间实验中的第一组实验(部分计算节点的计算能力较强)数据作为本实验的实验数据,将GSA算法和BSA算法下各参与者的响应时间分别代入式(21)计算,可得如下结果:

$FI_{GSA}=1$

$FI_{BSA}=1$

从上面的结果可以看出,GSA算法和BSA算法的公平性索引值均为1,说明两种算法均具有较好的公平性。即在采用两种算法的情况下,各个参与者获得的机会是公平的。

结束语 本文通过对云计算系统中的主要研究问题——任务调度问题进行研究,将博弈论应用于任务调度的研究中,分析了调度器作为博弈参与者时,在它们之间存在的纳什均衡,最终提出了基于序贯博弈的任务调度策略。最后,我们通过各个参与者的响应时间实验和算法公平性实验证明了本文

(下转第358页)

理办法,对通信资源的权责界面进行了划分,对通信设备的标识方法及标识规范进行了定义。同时,需要合适的 IT 技术作为管理办法的技术支撑。因此,我们设计了电力通信标识管理系统。该系统对通信设备采用 RFID 电子标签作为资源管理手段,在每个通信设备上关联一个 RFID 电子标签。利用手持式 RFID 读写设备,结合网络通信手段,将现场物理设备所关联的数据同后台的电力通信标识管理系统进行实时同步。利用该 RFID 中间件,可以满足多种不同类型的终端接入。同时,该中间件满足后台的管理系统的无缝接入。

通过对该中间件的接入,满足了重庆市电力公司在通信设备电子化管理上的需求。

结束语　本文全面地分析了 RFID 系统的结构和工作原理,在此基础上,结合实际应用中能耗、数据类型繁杂、存储等问题设计了 RFID 中间件三层结构,并分析了各个层面的主要功能和操作要点。同时针对存储、数据访问中进行策略设计,显著提升系统性能。并结合重庆市电力公司通信标识管理系统的具体需求予以应用,证明该中间件的设计能够解决电力通信标识管理的实际需要,该方案切实可行,能够有效解决电力通信标识管理的问题。

参 考 文 献

[1] 张捍东,朱林. 物联网中的 RFID 技术及物联网的构建[J]. 计算机技术与发展,2011,21(5):56-59

[2] 李如年. 基于 RFID 技术的物联网研究[J]. 中国电子科学研究院学报,2009,4(6):594-597

[3] 沈苏彬,范曲立,宗平,等. 物联网的体系结构与相关技术研究[J]. 南京邮电大学学报:自然科学版,2009,29(6):1-11

[4] 陈海明,崔莉,谢开斌,等. 物联网体系结构与实现方法的比较研究[J]. 计算机学报,2013,36(1):168-188

[5] 周永彬,冯登国. RFID 安全协议的设计与分析[J]. 计算机学报,2006,29(4):581-589

[6] 孙红,厉彦刚,陈世平,等. RFID 中间件数据处理研究[J]. 上海理工大学学报,2014,(3):234-238

[7] 丁振华,李锦涛,冯波,等. RFID 中间件研究进展[J]. 计算机工程,2006,32(21):9-11

[8] 蒋邵岗,谭杰. RFID 中间件数据处理与过滤方法的研究[J]. 计算机应用,2008,28(10):2613-26

(上接第 344 页)

提出的基于序贯博弈的任务调度策略(GSA 算法)具有较好的实验效果。

参 考 文 献

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

[2] 张建勋,古志民,郑超. 云计算研究进展综述[J]. 计算机应用研究,2010,27(2):429-433

[3] Fujimo N, HagiharaK. A comprison among grid scheduling algorithms for independent coarse-grained tasks [C]// Proceeding of the 2004 Symposium on Applications and the Internet Workshops. Washington, USA: IEEE Computer Society Press, 2004: 674-680

[4] Kim C, Kameda H. An Algorithm for Optimal Static Load Balancing in Distributed Computer Systems[J]. IEEE Transactions on computers, 1992, 41(3): 381-384

[5] Subrata R, Zomaya A Y, Landfeldt B. Game-Theoretic Approach for Load Balancing in Computational Grids[J]. IEEE Transaction on Parallel and Distributed Systems, 2008, 19(1): 62-76

[6] Lu K, Subrata R, Zomaya A Y. An Efficient Load Balancing Algorithm for Heterogeneous Grid Systems Considering Desirability of Grid Sites[C]// Proc. 25th IEEE Int'l Performance Computing and Comm. Conf. (IPCCC '06). 2006

[7] Lu K, Subrata R, Aomaya A Y. Towards Decentalized Load Balancing in a Computational Grid Environment[C]// Proc. First Int'l Conf. Grid and Pervasive Computing (GPC '06). 2006

[8] Liu G, Li J, Xu J. An Improved Min-min Algorithm in Cloud Computing[C]// Proc. of the 2012 International Conference of Modern Computer Science and Applications. 2013, 191: 47-52

[9] Parsa S, Entezari-Maleki R. RASA: A New Grid Task Scheduling Algorithm [J]. International Journal of Digital Content Technology and its Applications, 2009, 3: 91-99

[10] Li J, Peng J. Task Scheduling Algrithm Based on Improved Genetic Algorithm in Cloud Computing Environment[J]. Journal of Computer Applications, 2011, 31(1): 184-186

[11] He X, Sun X, Laszewski G. A QoS Guided Min-Min Heuristic for Grid Task Scheduling[J]. Journal of East China Normal University (Natural Science), 2010, 1: 127-134

[12] Chanhan S S, Joshi R. A Heuristic for QoS Based Independent Task Scheduling in Grid Environment[C]// Proc. of the International Conference on Industrial and Information System. 2010: 102-106

[13] Xu M, Cui L, Wang H, et al. A Multiple QoS Constrained Scheduling Strategy of Multiple Workflows for Cloud Computing[C]// Proc. of the 2009 IEEE Intenational Symposium on Parallel and Distributed Processing with Application. 2009: 629-634

[14] Deelman E, Singh G, Livny M, et al. The Cost of Doing Science on the Cloud: the Montage Example[C]// Proc. of the ACM/IEEE Conference on Supercomputing Piscataway. IEEE Press, 2001: 1-12

[15] Assuncao M, Costanzo A, Buyya R. Evaluating the Cost-benefit of Using Cloud Computing to Extend the Capacity of Clusters [C]// Proc. of the 18th ACM Intenational Symposium on High Performance Distributed Computing. Ney York: ACM Press, 2009: 141-150

[16] Ge X, Chen H, Du B, et al. Scheduling Strategy Research Based on Cloud Computing for Cluster Expansion[J]. Application Research on Computers, 2011, 28(3): 995-997

[17] Wang Jin-ting. Reliability analysis of M/G/1 queues with general retial times and server breakdowns[J]. Progress in Natural Science, 2006, 16(5): 464-473

[18] 孙荣桓,李建. 排队论基础[M]. 北京:科学出版社,2002

[19] Chow Y C, Kohler W H. Models for Dynamic Load Balancing in a Heterogeneous Multiple Processor System[J]. IEEE Transaction on Computers, 1979, 28: 354-361