第42卷 第6A期
2015年6月

计 算 机 科 学
Computer Science

Vol. 42 No. 6A
June 2015

云环境下基于可靠性的均衡任务调度算法研究

王 勇 刘美林 李 凯 任兴田 许荣强
(北京工业大学计算机学院 北京 100124)

摘 要 云计算作为一种新兴的具有商业特性的计算模式,已经受到了广泛的关注。云计算中的关键问题——任务调度问题也成为了社会各界研究的热点。主要以云计算系统中的可靠性需求为优化目标,运用博弈论工具,将云计算的任务调度系统建模为一个合作博弈模型。合作博弈的参与者为计算节点,效用函数为计算节点在稳定状态下的提供能力,博弈策略为任务在计算节点上的速率分配策略。系统中的各计算节点相互合作,选择自己的博弈策略,以期使系统在稳定状态下的提供能力最大。将计算节点看作具有一般重试时间和服务器崩溃的M/G/1排队系统,根据M/G/1排队论,分析了计算节点在稳定状态的提供能力,并根据合作博弈理论知识,证明了纳什讨价还价解的存在性,从而给出了最优博弈策略的求解算法;在此基础上,给出了基于可靠性的均衡任务调度算法。

关键词 云计算,合作博弈,任务调度,讨价还价解,可靠性

Reliability-based Job Scheduling Algorithm in Cloud Computing

WANG Yong LIU Mei-lin LI Kai REN Xing-tian XU Rong-qiang
(College of Computer Science and Technology, Beijing University of Technology, Beijing 100124, China)

Abstract As an emerging computing model with commercial properties, cloud computing has been widely concerned. In the research of cloud computing, task scheduling is a key issue. In this paper, we considered the requirements of reliability as the main target to optimize. Using game theory, we modeled the task scheduling system as a cooperative game. In the cooperative game model, computing nodes are participants, the ability that computing nodes can provide under steady state is the utility function and the strategy in the game is the task rate allocation strategy on computing nodes. To make the system provide max ability under steady state, computing nodes in the system cooperate with each other, choosing their own game strategy. We regarded the computing nodes as M/G/1 queues in this paper, and according to the M/G/1 queuing theory we analyzed the ability that computing nodes can provide under steady state. We proved that Nash bargaining solution exists. On this basis, we proposed the balancing task scheduling algrithm based on reliability.

Keywords Cloud computing, Cooperative game, Task scheduling, Bargaining solution, Reliability

1 引言

“计算”是人类社会活动中一个不可或缺的解决问题的方法和工具。但是,随着人们生活水平的提高和科技的不断进步,计算的规模呈几何级数增长,计算的形式也发生了很大的改变。伴随着互联网进入Web 2.0时代,云计算也应运而生。但是云计算并不是一个全新的概念,而是由分布式计算、并行计算、网格计算和一些其他的混合技术发展而来的。云计算通过虚拟技术将各种IT资源进行整合,并通过互联网提供给用户使用。这些IT资源非常丰富,包括应用程序、存储容量、计算能力、编程工具、协作工具和通信服务等。

任务调度问题是云计算研究中的关键问题之一,但由于云计算具有其独特的商业特性,因此,云计算的任务调度问题要比传统的分布式计算的任务调度问题更加复杂。目前,大多数对于云计算的任务调度研究都是基于Hadoop平台的。Hadoop平台本身集成了FIFO(First Input First Output)调度算法、计算能力调度算法和公平调度算法3种算法,这些算法只能处理一些概念简单、数量级不大的任务。面对日益复杂的云计算任务调度的需要,这些简单的任务调度算法已经不能满足用户的要求,因此,许多学者开始从多方面入手,研究云计算的任务调度问题。

文献[1-6]都是基于改进的遗传算法、蚁群算法等的研究成果,这些成果在优化任务调度的完成时间、提高任务调度效率方面都取得了比较好的效果。另外,在国外的研究成果中,Intel公司[7]研究了虚拟资源间的竞争关系,对虚拟机间由于竞争关系所带来的性能下降进行了建模,并以此为基础提出了预测虚拟资源竞争的模型。Freeman等人[8]提出将VM的管理集成到批调度器中,采用两级调度的方法向用户提供尽可能多的服务。Buyya等人[9,10]提出了面向市场的云计算架构,并给出了基于市场的资源管理策略。文献[11,12]利用改进的Min-Min算法,根据用户的QoS需求对系统中的任务调度进行了优化。文献[13]提出了一种云环境下多作业流多QoS约束的调度策略,对不同的用户考虑不同的QoS。文献[14,15]中提出的任务调度算法综合考虑了多个优化目标,如任务完成时间、资源利用率、系统的吞吐量、能量消耗等。

以上提到的云计算任务调度的研究工作,采用了不同的

王 勇 男,博士,副教授,主要研究方向为服务计算、网格计算、可信计算,E-mail:wangy@bjut.edu.cn。

研究思路，利用了不同的数学工具，取得了较好的研究成果。不同于已有的研究工作，本文以云计算系统中计算节点的可靠性（稳定状态的提供能力）为效用函数，运用博弈论为分析工具，以任务在计算节点上的速率分配策略为博弈策略，分析了合作博弈讨价还价解的存在性，并在此基础上给出了基于可靠性的均衡任务调度算法。

2 云计算任务调度系统模型

根据云计算的特点，以下2点假设是合理的：

1）目前，在云计算系统中，任务运行一次的代价通常较大（如执行时间较长），此外，云计算覆盖的地理范围可能较大，任务在网络上的传输时间较长，所以可以忽略调度器的内部处理代价（如进行任务分片时的可靠性），假设任务在计算节点上的执行代价是任务执行代价的关键所在。

2）调度器对任务进行分解以后得到任务分片，假定云计算系统中的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，计算节点的配置是可控的。

根据以上假设，给出如图1所示的任务调度系统的模型。

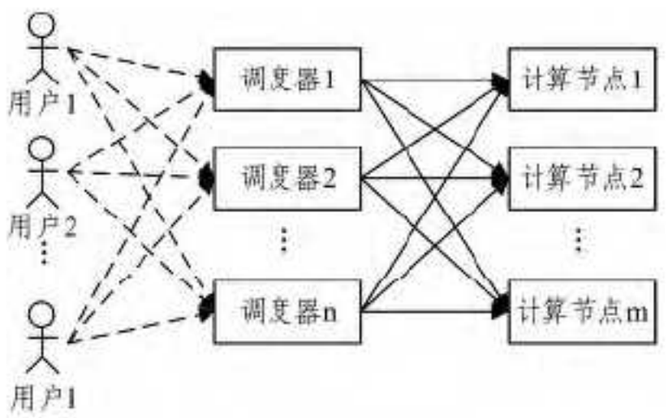

图1 任务调度系统的模型

假定在图1所示的任务调度系统中，有l个用户、n个调度器、m个执行任务的计算节点。

·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，把任务分配给系统中的计算节点执行。基于前面的假设1），对任务进行分解的时间忽略不计。

计算节点：任务的执行者，基于前面的假设2），计算节点具备一般性任务的执行能力。任务在计算节点j上的平均执行速率为u_j，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[16,17]。

设λ_i为任务到达调度器i的平均速率，则满足限制条件式(1)。式(1)的含义是所有调度器发出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。

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

设ϕ_j为计算节点j的平均任务到达速率，则满足限制条件(2)和(3)。式(2)的含义是计算节点j的平均任务达到速率应该小于其任务平均执行速率。而式(3)的含义是所有调度器发出任务的平均速率的加和应该等于所有计算节点的任务的平均到达速率的加和。

$$0\leqslant\phi_j<u_j \tag{2}$$

$$\sum_{i=1}^{n}\lambda_i=\sum_{j=1}^{m}\phi_j \tag{3}$$

3 基于可靠性的任务调度的合作博弈模型

3.1 合作博弈理论简介

本节简单介绍合作博弈的主要定义和结论，关于合作博弈的进一步指示请参见文献[18,19]。

定义1 一个合作博弈由以下部分组成：

1）m个博弈参与者；

2）一个非空、紧致的凸集$X\subseteq\mathrm{R}^m$，为m个参与者的可能的博弈策略的集合；

3）每个参与者$j(j=1,2,\cdots,m)$的效用函数$f_j(x)(x\in X)$是从X到R^m的函数，可以取最大值或最小值；

4）对于每个参与者$j(j=1,2,\cdots,m)$，如果效用函数$f_j(x)(x\in X)$以最小值为最佳效用，则$f_j(x)$的最小值记为u_j^0，则$u^0=(u_1^0,u_2^0,\cdots,u_m^0)$称为初始协商点。

定义2 在初始值u^0的基础上，一个合作博弈的可达效用集合记为$G:G=\{(U,u^0)\mid U\subset\mathrm{R}^m$是非空的且有上限的凸函数，$U_0=\{u\in U\mid u\geqslant u^0\}$是非空的$\}$。

博弈问题的均衡解可能不是唯一的，下面给出Pareto最优解的定义。

定义3 设$x\in X$且$f(x)=(f_1(x),f_2(x),\cdots,f_m(x))$，$f(x)\in U$，效用函数取最大值为最优解，如果对于每个$x'\in X$，均存在$f_j(x')\leqslant f_j(x),j=1,2,\cdots,m$，则称$x$是Pareto最优的。

定义4 如果映射$S:G\to\mathrm{R}$满足如下条件，则称其为纳什讨价还价点：

1）$S(U,u^0)\in U_0$；

2）$S(U,u^0)$是Pareto最优的；

3）线性公理：如果$\phi:\mathrm{R}^m\to\mathrm{R}^m,\phi_j(u_j)=a_ju_j+b_j,j=1,2,\cdots,m,a_j>0,b_j>0$，则$S(\phi(U),\phi(u^0))=\phi(S(U,u^0))$；

不相关公理：如果$V\subset U$，并且$S(U,u^0)\in V$，则$S(U,u^0)=S(V,u^0)$；

4）对称公理：S是对称的含义是两个参与者具有相同的初始值u_j^0和相同的效用函数$f_j(x)$，则两个参与者具有相同的结果。

5）如果$u^*=S(U,u^0)$为一纳什讨价还价点，则$f^{-1}(u^*)$为纳什讨价还价解的集合（定义5）。

6）下面的定理为纳什讨价还价点的性质。

定理1 设效用函数$f_j(x)$紧致且有上限，$x\in X$为一非空的凸闭集，$X_0=\{x\in X\mid f(x)\geqslant u^0\}$为能够获得的性能严格优于初始性能的参与者的集合，如果对于每一个$f_j(x),j\in J$，在X_0上是一一对应的，则存在一个唯一的纳什讨价还价解，且满足$f_j(x)>u_j^0,j\in J$，并且是对下面优化问题的求解：

$$\max_x\prod_{j\in J}(f_j(x)-u_j^0),x\in X_0 \tag{4}$$

式(4)的优化问题等同于下面的优化问题：

$$\max_x\sum_{j\in J}\ln(f_j(x)-u_j^0),x\in X_0 \tag{5}$$

需要注意的是不属于集合J的参与者不在上述优化问题考虑的范围内。

3.2 博弈分析

把云计算中的任务调度问题建模为一个合作博弈问题，m个计算节点作为博弈的参与者，计算节点的任务到达速率$\varphi=\{\phi_1,\phi_2,\cdots,\phi_j,\cdots,\phi_m\}$为博弈策略。

由文献[16]可知，云计算系统中的计算节点作为具有一般重试时间和服务器崩溃的 M/G/1 排队系统，其稳定状态的提供能力的计算公式如式(6)所示。

$$A_j=1-\phi_j\beta_{1j}(1+u_j'\gamma_j) \tag{6}$$

其中，ϕ_j 为计算节点 j 的平均任务到达速率，β_{1j} 为计算节点 j 任务服务时间的均值，u_j' 为计算节点 j 忙时失败的平均速率，γ_j 为计算节点 j 的重试时间的均值，A_j 满足约束如式(7)所示。

$$0<A_j\leqslant 1 \tag{7}$$

取计算节点 j 的稳定状态提供能力的倒数 D_j 为合作博弈的效用函数，如式(8)所示。

$$D_j=\frac{1}{1-\phi_j\beta_{1j}(1+u_j'\gamma_j)} \tag{8}$$

每一个计算节点 j 具有一个最大的 D_j^0，如式(9)所示。

$$D_j^0=\frac{1}{1-\phi_j^{\max}\beta_{1j}(1+u_j'\gamma_j)} \tag{9}$$

其中，$\varphi_j^{\max}$ 为计算节点 j 在保持服务质量的前提下，能够接受的最大的任务到达速率，且满足约束式(10)和式(11)。

$$0\leqslant\phi_j^{\max}<u_j \tag{10}$$

$$\phi_j<\phi_j^{\max} \tag{11}$$

我们注意到 D_j 是凸的且有界的，由 3.1 节合作博弈的定义与性质可知纳什讨价还价解可由式(12)定义的优化问题求解得到，且满足约束式(1)—式(3)、式(6)、式(7)、式(10)、式(11)。

$$\max_{\phi} D=\sum_{j=1}^{m}\ln(D_j^0-D_j)$$
$$=\sum_{j=1}^{m}\ln\left[\frac{1}{1-\phi_j^{\max}\beta_{1j}(1+u_j'\gamma_j)}-\frac{1}{1-\phi_j\beta_{1j}(1+u'_j\gamma_j)}\right] \tag{12}$$

通过 $\frac{\partial\max D}{\partial\phi_j}\leqslant 0$，$\frac{\partial^2\max D}{\partial\phi_j^2}\leqslant 0$ 可知式(12)为一个凹函数，且约束式(1)—式(3)、式(6)、式(7)、式(10)、式(11)均为线性约束，因此，满足一阶 Kuhn-Tucker 条件。取拉格朗日函数如式(13)所示。

$$L=\sum_{j=1}^{m}\ln\left[\frac{1}{1-\phi_j^{\max}\beta_{1j}(1+u_j'\gamma_j)}-\frac{1}{1-\phi_j\beta_{1j}(1+u_j'\gamma_j)}\right]+$$
$$\alpha\left(\sum_{j=1}^{m}\phi_j-\sum_{i=1}^{n}\lambda_i\right) \tag{13}$$

令 $\frac{\partial L}{\partial\phi_j}=0$ 可得式(14)：

$$\frac{1-\phi_j^{\max}B_j}{B_j\phi_j^2-(1+\phi_j^{\max}B_j)\phi_j+\phi_j^{\max}}-\alpha=0 \tag{14}$$

其中，$B_j=\beta_{1j}(1+u_j'\gamma_j)$。

对式(14)进行求解，可得式(15)：

$$\phi_j=\frac{1+\phi_j^{\max}B_j}{2B_j}-\frac{\sqrt{(1+\phi_j^{\max}B_j)^2-4B_j\left(\phi_j^{\max}-\frac{1-\phi_j^{\max}B_j}{\alpha}\right)}}{2B_j} \tag{15}$$

将式(15)代入式(3)可得式(16)，α 的取值由式(16)决定。

$$\sum_{j=1}^{m}\left\{\frac{1+\phi_j^{\max}B_j}{2B_j}-\frac{\sqrt{(1+\phi_j^{\max}B_j)^2-4B_j\left(\phi_j^{\max}-\frac{1-\phi_j^{\max}B_j}{\alpha}\right)}}{2B_j}\right\}$$
$$=\sum_{i=1}^{n}\lambda_i \tag{16}$$

由式(15)可知，当式(17)成立时，$\varphi_j<0$，此时令 $\phi_j=0$。

$$\alpha<\frac{1}{\phi_j^{\max}}-\beta_{1j}(1+u_j'\gamma_j) \tag{17}$$

考虑到结果的紧致性，可得纳什讨价还价解的求解算法如下：

1) 令 $\theta_j=\frac{1}{\phi_j^{\max}}-\beta_{1j}(1+u_j'\gamma_j)$，并对计算节点按照 θ_j 从小到大的顺序进行排序 $\theta_1<\theta_2<\cdots<\theta_j<\theta_m$，排序的结果保存到变量 $index$ 中。

2) 对 d 从 m 到 1，依次执行步骤 3)—5)：

3) 利用式(18)求取 α：

$$\sum_{j=1}^{d}\left\{\frac{1+\phi_{index(j)}^{\max}B_{index(j)}}{2B_{index(j)}}-\right.$$
$$\left.\frac{\sqrt{(1+\phi_{index(j)}^{\max}B_{index(j)})^2-4B_{index(j)}\left(\phi_{index(j)}^{\max}-\frac{1-\phi_{index(j)}^{\max}B_{index(j)}}{\alpha}\right)}}{2B_{index(j)}}\right\}$$
$$=\sum_{i=1}^{n}\lambda_i \tag{18}$$

4) 将 α 代入式(19)求取 j 从 1 到 d 的 $\phi_{index(j)}$，其余计算节点的平均任务到达速率赋值为 0，其中 $d\leqslant m$。

$$\phi_{index(j)}=\frac{1+\phi_{index(j)}^{\max}B_{index(j)}}{2B_{index(j)}}-$$
$$\frac{\sqrt{(1+\varphi_{index(j)}^{\max}B_{index(j)})^2-4B_{index(j)}\left(\phi_{index(j)}^{\max}-\frac{1-\phi_{index(j)}^{\max}B_{index(j)}}{\alpha}\right)}}{2B_{index(j)}} \tag{19}$$

5) 若在步骤 4)中求出的 ϕ_j 中存在 $\phi_j<0$，则 d 减 1，返回到步骤 3)继续执行，否则结束循环，得到各参与者的最优策略。

3.3 均衡的任务调度算法

在 3.2 节求出各参与者的最优策略的基础上，可以给出任务调度算法如下：

1) 系统参数初始化。设调度器 i 发出任务的平均速率为 $\lambda_i(0)$，$i=1,2,\cdots,n$，“0”符号表示稳定状态，下同。计算节点 j 的任务平均任务处理速率为 $u_j(0)$，$j=1,2,\cdots,m$。

2) 设定 $\phi_j^{\max}$ 的初始值为 $\phi_j^{\max}(0)=\frac{u_j(0)}{3}$，$\phi_j^{\max}$ 为计算节点 j 能够接受的最大任务到达速率，计算节点 j 的平均任务到达速率初始值为 $\phi_j(0)=\phi_j^{\max}(0)$，$u_j'(0)=\frac{u_j(0)}{10}$，$\gamma_j(0)=\frac{5}{u_j(0)}$，目标函数优化值 D 的误差精度 $\varepsilon(0)\leqslant 10^{-6}$；

3) 利用公式 $D=\sum_{j=1}^{m}\frac{1}{1-\phi_j\beta_{1j}(1+u_j'\gamma_j)}$ 计算目标函数优化值 $D(0)=latterD$；

4) 若 $D(0)=latterD$ 相对于 D 的优化目标值的误差 $\varepsilon(0)$ 满足 $\varepsilon(0)\leqslant 10^{-6}$，则求得任务调度的最优分配策略；否则，执行以下步骤：

4.1) 令 $formerD=latterD$，$formerD$ 用于暂存上一次调度方案下的目标函数值 $latterD$；

4.2) 按式 $\theta_j=\frac{1}{\phi_j^{\max}}-\beta_{1j}(1+u'_j\gamma_j)$ 计算系统中计算节点 j 在处于最大的任务到达速率 $\phi_j^{\max}$ 时允许提供的单位任务计算能力 θ_j，并对计算节点按照 θ_j 从小到大的顺序进行排序

$\theta_1<\theta_2<\cdots<\theta_j<\theta_m$，排序的结果保存到变量 $index$ 中。

4.3) 对于 d 从 m 到 1，依次执行以下步骤：

4.3.1) 利用式(18)求取 α；

4.3.2) 将 α 代入式(19)求取 j 从 1 到 d 的 $\phi_{index(j)}$，其余计算节点的平均任务到达速率赋值为 0；

4.3.3) 在上步求出的 ϕ_j 中，若存在 $\phi_j<0$，则 d 减 1，返回到步骤 4.3.1) 继续执行；否则结束循环，得到各参与者的最优策略 $\varphi=\{\phi_1,\phi_2,\cdots,\phi_j,\cdots,\phi_m\}$。

4.4) 将求得的 $\varphi=\{\phi_1,\varphi_2,\cdots,\phi_j,\cdots,\phi_m\}$ 代入公式 $D=\sum_{j=1}^{m}\frac{1}{1-\phi_j\beta_{1j}(1+u_j{}'\gamma_j)}$ 求取新的目标函数值 $latterD$，$\varepsilon=|latterD-formerD|$，若 $\varepsilon\leqslant10^{-6}$，则得到最优任务分配策略，算法结束；否则返回到步骤 4.1) 继续执行。

4 实验设计与结果分析

为了证明本文中提出的算法具有较好的优化效果，我们选取了已有的基于非合作博弈的任务调度算法[20]和均衡任务调度算法[21]进行比较。为了实验结果比较的方便，标记基于非合作博弈的任务调度算法为 NSA 算法，标记均衡任务调度算法为 BSA 算法，标记本文提出的基于可靠性的算法为 RSA 算法。

设定 ω_i 为调度器的相对任务到达速率，则调度器的实际任务到达速率 λ_i 按照式(20)计算。

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

其中，ρ 为云计算系统的负载系数。在本节中，我们分别从以下 5 个方面进行了实验设计。实验的详细数据和实验结果分析如下所示。

4.1 均衡状态下的目标函数值

在该实验中，假定云计算系统的负载系数 ρ 为 0.6，调度器的数目为 10，计算节点的数目为 15，各调度器的相对任务到达速率 ω_i 如表 1 所列。

表 1 调度器相对任务到达速率

调度器	1	2	3-5	6	7	8	9,10
相对任务到达速率(任务/s)	0.0035	0.002	0.001	0.0006	0.0005	0.0002	0.0001

由云计算系统的特点可知，计算节点的计算能力可能是均衡的，也可能出现部分节点计算能力较强的情况。因此，在该实验中我们分别进行了两组实验。

第一组实验是系统的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平均任务处理速率(计算能力)u_j 如表 2 所列。

表 2 计算节点的平均任务处理速率(部分节点计算能力较强)

计算节点	1-7	8-10	11	12	13	14	15
平均任务处理速率(任务/s)	0.02	0.033	0.0231	0.02511	0.0153	0.023	0.025

在上述初始条件下，我们可以分别求得 3 种算法下各计算节点的稳定状态的提供能力，如表 3 所列。其对比图如图 2 所示。

表 3 计算节点稳定状态的提供能力(部分节点计算能力较强)

计算节点	RSA	NSA	BSA
1	1.0000	1.0000	1.0091
2	1.0000	1.0000	1.0091
3	1.0000	1.0000	1.0091
4	1.0000	1.0000	1.0091
5	1.0000	1.0000	1.0091
6	1.0000	1.0000	1.0091
7	1.0000	1.0000	1.0091
8	1.0248	1.0329	1.0091
9	1.0248	1.0329	1.0091
10	1.0248	1.0329	1.0091
11	1.0000	1.0000	1.0091
12	1.0000	1.0000	1.0091
13	1.0000	1.0000	1.0091
14	1.0000	1.0000	1.0091
15	1.0000	1.0000	1.0091
$\sum D_j$	15.0744	15.0978	15.1365

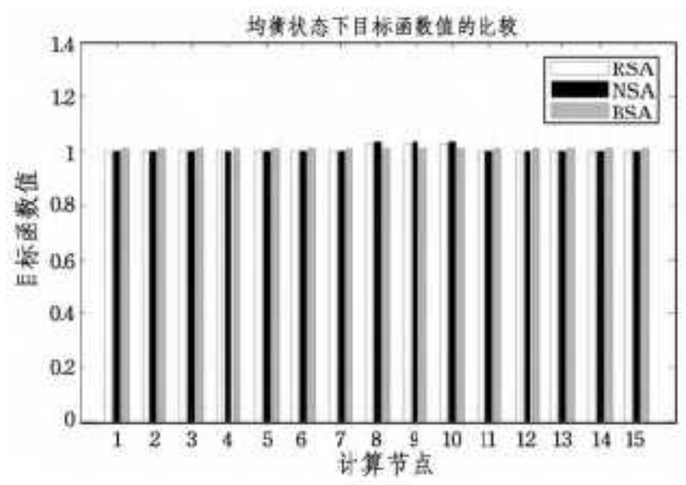

图 2 均衡状态下目标函数值的比较(部分节点计算能力较强)

从表 3 和图 2 中可以看出，对于 RSA 算法和 NSA 算法，计算能力强的节点会从调度器接收到较多的任务，而在 BSA 算法下，所有计算节点提供的计算能力是一样的。从表 3 的最后一行还可以看出，所有计算节点稳定状态的提供能力的加和中，RSA 算法的结果最小，说明 3 种算法中，RSA 算法能使系统提供更高的计算能力，具有更好的优化效果。

第二组实验是系统中计算节点的计算能力均衡的实验。设计算节点的平均任务处理速率(计算能力)u_j 如表 4 所列。其他实验数据同第一组中的数据。

表 4 计算节点的平均任务处理速率(节点计算能力均衡)

计算节点	1	2	3,4	5	6,7	8,9	10	11,12	13,14	15
平均任务处理速率(任务/s)	0.031	0.03	0.029	0.0312	0.03	0.032	0.033	0.029	0.03	0.031

在上述初始条件下我们可以分别求得 3 种算法下各计算节点的稳定状态的提供能力，如表 5 所列。其对比图如图 3 所示。

表 5 计算节点稳定状态的提供能力(节点计算能力均衡)

计算节点	RSA	NSA	BSA
1	1.0127	1.0107	1.0091
2	1.0018	1.0000	1.0091
3	1.0000	1.0000	1.0091
4	1.0000	1.0000	1.0091
5	1.0149	1.0139	1.0091
6	1.0018	1.0000	1.0091
7	1.0018	1.0000	1.0091
8	1.0234	1.0268	1.0091
9	1.0234	1.0268	1.0091
10	1.0337	1.0427	1.0091
11	1.0000	1.0000	1.0091
12	1.0000	1.0000	1.0091
13	1.0018	1.0000	1.0091
14	1.0018	1.0000	1.0091
15	1.0127	1.0107	1.0091
$\sum D_j$	15.1298	15.1316	15.1365

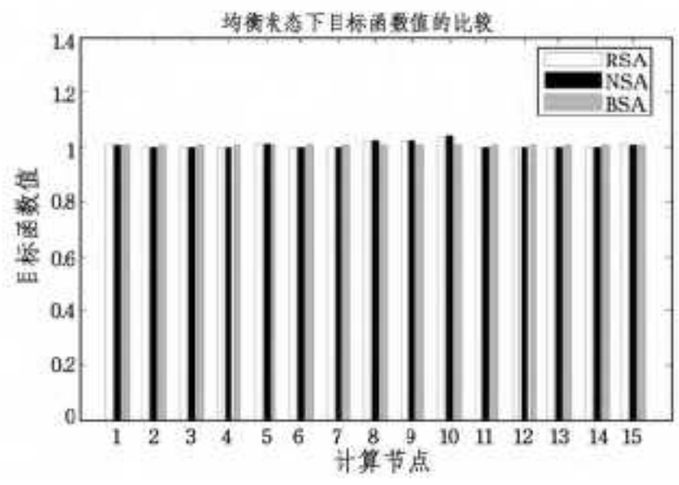

图 3　均衡状态下目标函数值的比较(节点计算能力均衡)

从表 5 的最后一行可以看出,所有计算节点稳定状态的提供能力的加和中,RSA 算法的结果最小,说明 3 种算法中,RSA 算法能使系统提供更高的计算能力,具有更好的优化效果。

综合以上两组实验,可以看出:无论云计算系统中计算节点的计算能力是否均衡,本文提出的 RSA 算法都能使系统提供更高的计算能力,较 NSA 算法和 BSA 算法具有更好的优化效果。

4.2　系统负载的影响

在该实验中,我们假设系统的负载系数 ρ 由 0.1 依次增加到 0.9,每次增加 0.1。其余实验数据同实验 4.1 的第一组实验中的数据。

在以上初始条件下,我们可以分别用 3 种算法计算出不同系统负载系数下的目标函数值,如图 4 所示。从图 4 中可以看出,随着系统负载系数的增大,3 种算法下的目标函数值都逐渐增大,并且,RSA 算法下的目标函数值一直是 3 种算法中最小的,说明 RSA 算法能使系统提供更高的计算能力。随着系统负载系数的增大,RSA 算法较 NSA 算法和 BSA 算法的优势也逐渐变得明显。

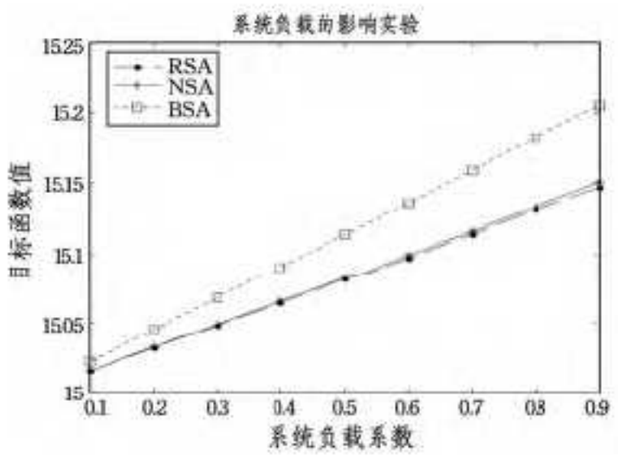

图 4　系统负载的影响

4.3　系统规模的影响

在云计算的任务调度系统中,影响云计算系统对外提供计算能力的因素包括调度器和计算节点,因此,系统规模的变化包括调度器数目的变化和计算节点数目的变化。本节中我们分别做了以下两组实验。

第一组实验是调度器数目变化的影响实验。假设调度器的数目从 5 变化到 20,依次增加 1 个调度器。系统的负载系数设为 0.6,计算节点的数目设为 15,各计算节点的平均任务处理速率(计算能力)u_j 如表 6 所列。20 个调度器的相对任务到达速率 ω_i 如表 7 所列。

表 6　计算节点的平均任务处理速率

计算节点	1-3	4-7	8-10	11	12	13	14	15
平均任务处理速率(任务/s)	0.01	0.02	0.033	0.028	0.03	0.028	0.024	0.03

表 7　调度器的相对任务到达速率

调度器	1	2	3-5	6	7	8	9,10	11	12
相对任务到达速率(任务/s)	0.0035	0.002	0.001	0.0006	0.0005	0.0002	0.0001	0.003	0.002

调度器	13	14	15	16	17	18	19	20
相对任务到达速率(任务/s)	0.0035	0.001	0.0045	0.0003	0.001	0.0035	0.00063	0.0029

在上述初始条件下,分别用 3 种算法求得调度器数目变化时系统中计算节点的任务到达速率,然后得到反映系统稳定状态下的提供能力的目标函数值,各算法对应的目标函数值如表 8 所列。其对比图如图 5 所示。

表 8　系统规模的影响——调度器数目变化

调度器数目	RSA	NSA	BSA
5	15.0821	15.0831	15.1156
6	15.0881	15.0892	15.1239
7	15.0930	15.0942	15.1307
8	15.0950	15.0963	15.1335
9	15.0960	15.0973	15.1349
10	15.0962	15.0983	15.1362
11	15.1266	15.1291	15.1776
12	15.1470	15.1499	15.2053
13	15.1830	15.1869	15.2540
14	15.1934	15.1975	15.2680
15	15.2387	15.2460	15.3312
16	15.2419	15.2493	15.3354
17	15.2525	15.2602	15.3495
18	15.2897	15.2988	15.3991
19	15.2965	15.3058	15.4081
20	15.3275	15.3382	15.4495

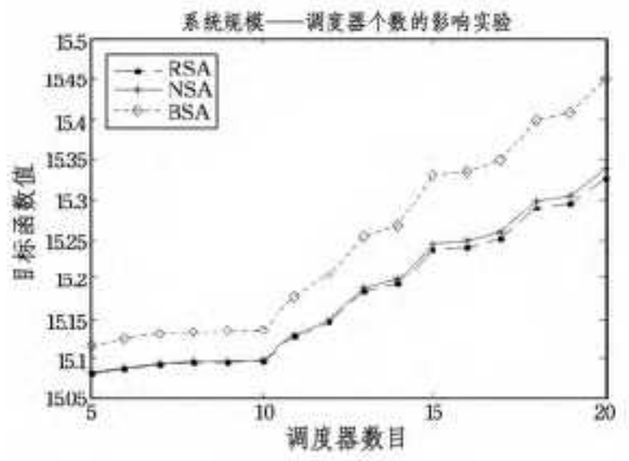

图 5　系统规模的影响——调度器数目变化

在表 8 中,从纵向来看,无论采用 RSA 算法、NSA 算法还是 BSA 算法,当调度器的数目增加导致系统的任务随之增加时,系统的目标函数值都会随之增大;从横向来看,RSA 算法对应的目标函数值无论在调度器的数目为多少时,都是 3 种算法中目标函数值最小的,即 RSA 算法能够使系统提供更高的计算能力,较 NSA 算法和 BSA 算法有较好的优化效果。从图 5 中可以看出,随着调度器数目的增加,RSA 算法较其它两种算法的优势也逐渐增大。

第二组实验是计算节点数目变化的影响实验。假设调度器的数目为 15,系统的负载系数为 $\rho=0.6$,计算节点的数目从 15 变化到 20,依次增加 1 个。其中,各调度器的相对任务到达速率 ω_i 如表 9 所列,20 个计算节点的平均任务处理速率(计算能力)u_j 如表 10 所列。

表 9　调度器的相对任务达到速率

调度器	1	2	3-5	6	7	8
相对任务到达速率（任务/s）	0.0035	0.002	0.001	0.0006	0.0005	0.0002
调度器	9,10	11	12	13	14	15
相对任务到达速率（任务/s）	0.0001	0.003	0.002	0.0035	0.001	0.0045

表 10　计算节点的平均任务处理速率

计算节点	1-3	4-7	8-10	11	12	13	14
平均任务处理速率（任务/s）	0.01	0.02	0.033	0.028	0.03	0.028	0.024
计算节点	15	16	17	18	19	20	
平均任务处理速率（任务/s）	0.03	0.028	0.033	0.025	0.019	0.021	

在上述初始条件下，分别用 3 种算法得到反映系统稳定状态下的提供能力的目标函数值，各算法对应的目标函数值如表 11 所列。其对比图如图 6 所示。

表 11　系统规模的影响——计算节点数目变化

计算节点数目	RSA	NSA	BSA
15	15.2387	15.2460	15.3312
16	16.2583	16.2671	16.3532
17	17.2782	17.2870	17.3753
18	18.2964	18.3056	18.3974
19	19.3102	19.3198	19.4195
20	20.3255	20.3356	20.4415

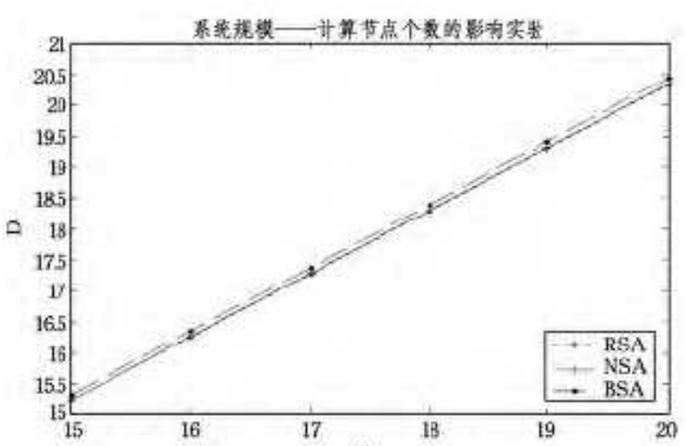

图 6　系统规模的影响——计算节点数目变化

虽然在图 6 中 RSA 算法和 NSA 算法的目标函数值的差别不是很明显，但是在表 11 中，我们可以看出无论计算节点的个数为多少时，RSA 算法的目标函数值在 3 种算法中都是最小的，即说明 RSA 算法能让系统提供更高的计算能力。较 NSA 算法和 BSA 算法，RSA 算法具有更好的优化效果。

通过以上两组实验结果的对比，说明无论系统规模怎样变化，本文提出的 RSA 算法总能使系统提供更高的计算能力，相对于 NSA 算法和 BSA 算法，RSA 算法总是有更好的优化效果。

4.4　公平性实验

算法的公平性可以用来衡量用户使用系统中资源的机会是否相同。我们用式(21)所示的公平性索引来衡量一个算法的公平性。公平性索引值越趋近于 1，说明一个算法的公平性越好。

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

在该实验中，我们采用与实验 4.1 节中第一组实验相同的初始数据。将其实验结果代入式(21)，可得到如下结果：

$$FI_{RSA}=0.9999, FI_{NSA}=0.9998, FI_{BSA}=1$$

通过以上结果我们可以看出，3 种算法都具有较好的公平性，其中 BSA 算法的公平性最好，NSA 算法的公平性稍差一些，本文提出的 RSA 算法的公平性略微好于 NSA 算法。

4.5　算法的收敛性实验

算法的收敛性也是衡量算法好坏的一个重要指标。算法的收敛速度直接影响到系统完成任务调度的速度，从而影响系统返回给用户计算结果的速度。当本文提出的算法中，影响算法收敛速度的主要因素是计算任务在计算节点上的速率分配策略的速度。当计算任务在计算节点上的速率分配策略且我们认为当两次计算结果的变化范围 ε 满足 $\varepsilon\leqslant 10^{-6}$ 时，计算结果便趋于稳定，即得到前面所说的纳什讨价还价解。

在该实验中，我们首先测试了实验(1)中第一组实验时 ε 的收敛情况，结果如图 7 所示。从图中可以看出，经过 3 次迭代就可以得到最终的分配策略，收敛速度非常快。

接下来，我们分别测试了系统负载变化时算法的收敛性、调度器数目变化时算法的收敛性和计算节点数目变化时算法的收敛性，其结果分别如图 8—图 10 所示。

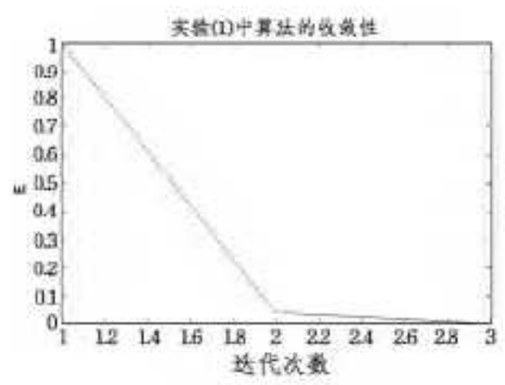

图 7　实验一中算法的收敛性

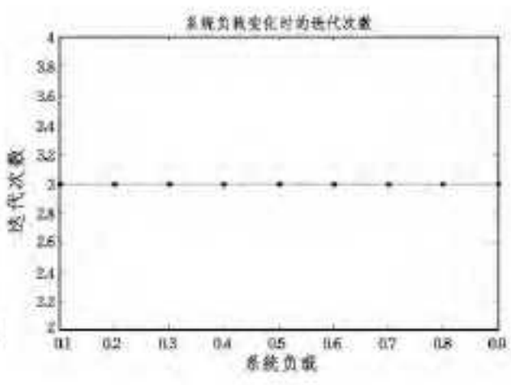

图 8　系统高负载变化时算法的收敛性

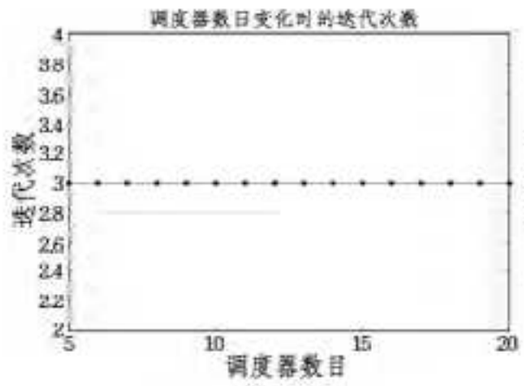

图 9　调度器数目变化时算法的收敛性

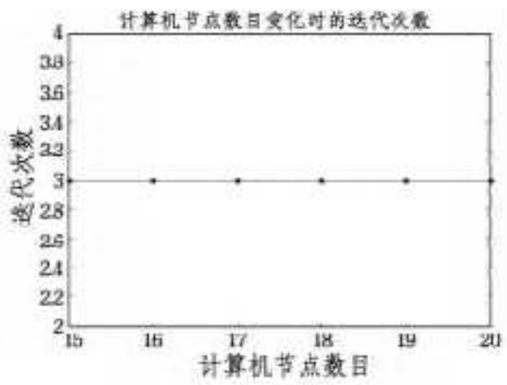

图 10　计算节点数目变化时算法的收敛性

从以上各实验结果中可以看出，在多种实验条件下，本文提出的 RSA 算法都是经过 3 次迭代就可以得到最终的任务分配策略，说明 RSA 算法的收敛速度很快，而且比较稳定。

结束语　本文将合作博弈理论应用于云计算的任务调度问题的研究中，建立了云计算调度问题中的合作博弈模型，将系统中的计算节点看作博弈的参与者，将计算节点的稳定状态的提供能力看作效用函数，将任务在计算节点上的速率分配策略看作博弈策略。通过理论分析，最后提出了基于可靠性的均衡任务调度算法。本文的研究为云计算任务调度问题的研究提供了新的方法和思路。在今后的研究工作中，我们会逐渐完善本文中提出的算法，使其优化效果更好。

参 考 文 献

[1] 朱宗斌，杜中军. 基于改进 GA 的云计算任务调度算法[J]. 计算机工程与应用，2011，12(9)

[2] 李建锋，彭舰. 云计算环境下基于改进遗传算法的任务调度算法[J]. 计算机应用，2011，31(1)

[3] 张春艳，刘清林，孟珂. 基于蚁群优化算法的计算任务分配[J]. 计算机应用，2012，32(5)：1418-1420

[4] 刘永，王新华，邢长明，等. 云计算环境下基于蚁群优化算法的资源调度策略[J]. 计算机技术发展，2011，21(9)

[5] 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

[6] 王永贵,韩瑞莲.基于改进蚁群算法的云环境任务调度研究[J].计算机测量与控制,2011,19(5):1203-1211

[7] Iyer R, Illikkal R, Tickoo O, et al. VM3: Measuring, Modeling and Managing VM Shared Resources[J]. Computer Networks, 2009,53(17):2873-2887

[8] Freeman T, Keahey K. Flying Low: Simple Leases with Workspace Pilot[C]// Proc of the 14th International Euro-Par Conference on Parallel Processing. Berlin: Springer-Verlag, 2008: 499-509

[9] Buyya R, Yeoa C S, Venugopal S. Market-Oriented Cloud Computing: vision, hype and relaity for delivering IT services as computing utilities[C]// Proc of the 10th IEEE International Conference on High Performance Computing and Communications. 2008:1-12

[10] Buyya R, Yeoa C S, Venugopal S, et al. Cloud Computing and Emerging IT Platforms: vision, hype and relity for delivering computing as the 5th utility[J]. Future Generation Computer Systems, 2009, 25(8):599-616

[11] He Xiao-shan, Sum Xian-he, vol Laseewski G. QoS Guided Min-Min Heuristic for Grid Task Scheduling[J]. Journal of Computer Science and Technology, 2003, 18(4):442-451

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

[13] Xu Meng, Cui Li-zhen, Wang Hai-yang, et al. A Multiple QoS Constrained Scheduling Strategy of Multiple Workflows for Cloud Computing[C]// Proc of IEEE International Symposium on Parallel and Distributed Processing with Applications. 2009: 629-634

[14] Prodan R, Wieczorek M, Fard H M. Double Auction-Based Scheduling of Scientific Applications in Distributed Grid and Cloud Environments[J]. Journal Grid Computing, 2011, 9(4):531-548

[15] Mezmaz M, Melab N, Kessaci Y, et al. A Parallel Bi-objective Hybrid Metaheuristic for Energy-Aware Scheduling for Cloud Computing Systems[J]. Parallel Distributed Computing, 2011, 71(1):1497-1508

[16] Wang Jin-ting. Reliability Analysis of M/G/1 Queues with General Retrial Times and Server Breakdowns[J]. Progress in Natural Science, 2006, 16(5):464-473

[17] 孙荣桓,李建.排队论基础[M].北京:科学出版社,2002

[18] Nash J. The Bargaining Problem [J]. Econometrica, 1950, 18 (2):155-162

[19] Muthoo A. Bargaining Theory with Applications [M]. Cambridge, UK: Cambridge University Press, 1999

[20] 王勇,李凯,刘美林.基于可靠性和非合作博弈的计算网格任务调度方法[P].中国发明专利,2012

[21] 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

(上接第 310 页)

结束语 在异步的基于占空比的无线传感器网络 MAC 协议设计中,提升协议性能的一个关键问题就是如何协调发送方与接收方之间的唤醒调度,使之能够更好地进行数据收发之间的衔接与配合,从而降低传输时延,提升系统性能。本文分析了传统协议中接收方主导唤醒模式对传输时延的影响,提出一个发送方主导的唤醒策略,并以此为基础设计 RISD-MAC 协议。RISD-MAC 协议使接收方能够根据发送方的唤醒调度自适应调整自己的唤醒时间,以匹配发送方的数据发送,从而降低传输时延,提高传输效率。仿真结果表明,同 RI-MAC 协议相比,RISD-MAC 协议能够在保证分组投递率的情况下,减少传输时延,降低占空比。

参 考 文 献

[1] Hu Fei, Cao Xiao-jun. Wireless Sensor Networks: Principles and Practice [B]. Auerbach Publications, 2010

[2] Akyildiz I F, Vuran M C. Wireless Sensor Networks [B]. John Wiley & Sons Ltd., 2010

[3] Pei Huang, Li Xiao, Soltani S, et al. The Evolution of MAC Protocols in Wireless Sensor Networks: A Survey [J]. IEEE Communications Surveys & Tutorials, 2013, 15 (1):101-120

[4] 胥楚贵,邓晓衡,刘持标,等. AGQ-MAC:无线传感器网络中基于 Grid Quorum 的异步低占空比 MAC 协议[J].小型微型计算机系统,2013,34(10):2383-2387

[5] Wei Ye, Heidemann J, Estrin D. An Energy-Efficient MAC Protocol for Wireless Sensor Networks [C]// Proceedings of the IEEE INFOCOM. 2002

[6] Wei Ye, Silva F, Heidemann J. Ultra-Low Duty Cycle MAC with Scheduled Channel Polling [C]// Proceedings of the ACM SenSys. 2006

[7] Sun Yan-jun, Du Shu, Gurewitz O, et al. DW-MAC: A Low Latency, Energy Efficient Demand-Wakeup MAC Protocol for Wireless Sensor Networks [C]// Proceedings of the ACM MOBIHOC. 2008

[8] Tang Hong-wei, Cao Jian-nong, Liu Xue-feng, et al. SR-MAC: A Low Latency MAC Protocol for Multi-Packet Transmissions in Wireless Sensor Networks [J]. Journal of Computer Science and Technology, 2013, 28 (2): 329-342

[9] Tang Hong-wei, Sun Cai-xia, Liu Yong-peng, et al. Low-Latency Asynchronous Duty-Cycle MAC Protocol for Burst Traffic in Wireless Sensor Networks [C]// Proceedings of the IEEE IWCMC. Sardinia, 2013

[10] Polastre J, Hill J, Culler D. Versatile Low Power Media Access for Wireless Sensor Networks [C]// Proceedings of the ACM SenSys. 2004

[11] Buettner M, Yee G V, Anderson E, et al. XMAC: A Short Preamble MAC Protocol for Duty-Cycled Wireless Sensor Networks [C]// Proceedings of the ACM SenSys. 2006

[12] El-Hoiydi A, Decotignie J-D. WiseMAC: An Ultra Low Power MAC Protocol for Multi-hop Wireless Sensor Networks [C]// Proceedings of the ALGOSENSORS. 2004

[13] Sun Yan-jun, Gurewitz O, Johnson D B. RI-MAC: A Receiver Initiated Asynchronous Duty Cycle MAC Protocol for Dynamic Traffic Loads in Wireless Sensor Networks [C]// Proceedings of the ACM SenSys. 2008

[14] Tang Lei, Sun Yan-jun, Gurewitz O, et al. PW-MAC: An Energy-Efficient Predictive-Wakeup MAC Protocol for Wireless Sensor Networks [C]// Proceedings of the IEEE INFOCOM. 2011

[15] Yick J, Mukherjee B, Ghosal D. Wireless Sensor Network Survey [J]. Elsevier Computer Networks, 2008, 52(12):2292-2330

[16] 张棋飞,刘威,孙宝林,等.基于冲突分类模型的冲突解析算法[J].软件学报,2010,21(3):548-563

[17] Doudou M, Djenouri D, Badache N. Survey on Latency Issues of Asynchronous MAC Protocols in Delay-Sensitive Wireless Sensor Networks [J]. IEEE Communications Surveys & Tutorials, 2013, 15(2):528-550