

文章编号:1001-9081(2010)09-2530-05

基于资源预测的网格任务调度模型

程宏兵

(江苏城市职业学院 信息工程系,南京 210017)

(2005tafe@163.com)

摘要:跨越虚拟组织中多个域(或集群)的网格任务调度由于资源的不确定性(如动态性和异构性)而成为网格应用中亟待解决的问题。提出了一种有效的基于资源预测的网格任务调度模型——RPTS,该模型利用加权最小二乘方法进行参数估计的自回归滑动平均(ARMA)预测方法对网格环境下的主机负载进行预测。利用上述资源预测结果和一类数据并行性网格任务的建模结果,对它们进行预处理、匹配并调度执行。RPTS充分考虑了网格环境下资源的动态性和异构性,为解决网格环境下任务调度问题提供了一种较好的方法。与其他一些网格任务调度方法进行了一系列的仿真实验,结果表明RPTS模型具有任务执行时间最短和稳定性较好的特点。

关键词:网格计算;任务调度;资源预测;自回归滑动平均模型

中图分类号: TP315; TP393.07 **文献标志码:** A

Task scheduling model based on resource prediction for grid computing

CHENG Hong-bing

(Department of Information Engineering, College of Jiangsu City Vocation, Nanjing Jiangsu 210017, China)

Abstract: Resources in grid computation environments are heterogeneous and dynamic, and tasks in grid computation environments are executed by computers from different domains or clusters of virtual organization synergistically; the static task scheduling is not fit for tasks execution in grid computing environments. In the paper, a task scheduling model based on the results of resources prediction was proposed. Firstly, a method of weighted least square estimation was given to construct Autoregressive Moving Average (ARMA) model, which would be applied in CPU load prediction of grid computer. After modeling a kind of data parallel grid tasks, the task scheduling model based on the results of resources prediction was presented. Finally the simulations on the proposed model and some other models were designed and accomplished. The simulation results demonstrate that the presented model can run both significantly faster and more stable than other models.

Key words: grid computing; task scheduling; resource prediction; Autoregressive Moving Average (ARMA) model

0 引言

网络是一个集成的计算与资源环境,是未来信息社会的网络基础设施。其目标是要把分布在不同地方的各种资源,如计算资源、存储资源等联合起来,形成一个虚拟的、统一的、强大的计算环境^[1],从而使目前分散的网络资源统一成能协同工作的“网络计算机”,以解决人类面临的诸多挑战性难题。在网格计算研究方面,任务调度是目前的研究热点,是网格应用中亟待解决的一个关键问题,将直接影响网格应用的实现。网格环境中任务调度遇到的一个困难就是各种资源是动态和异构的。参与计算的每个处理器的速度是不一样的、网络性能也是动态变化的。用常规静态的任务调度方法不能满足网络异构和动态的特性。

对于网格任务调度,国外学者提出了不少方法,如J. M. Schopf等人分析了随机调度算法(Stochastic Scheduling)^[2],Yang, L. Y等人研究了保守调度任务方法^[3],以及其他一些主机负载均衡的调度方法。国内对网格任务调度算法也进行了积极的研究,并已经有了一些研究成果,金海等人开发了一种容错计算网格任务调度的随机Petri网模型,并给出了网格任务分派策略和计算站点内的任务选择策略^[4]。Jian Zhang等人提出了一种动态调度算法^[5];查礼等人讨论了一种数据

和计算密集混合元任务的调度算法(Transfer Computation Ratio,TCR)^[6];G. Yang等人研究了把整个任务分成几个子任务的块Broyden算法^[7],并将其应用于非线性方程组的并行求解。

鉴于以上一些研究成果的启发,考虑到网格环境下的资源动态和异构的特性,本文提出了一种基于资源预测的网格任务调度模型——RPTS(tasks scheduling model based on resource prediction),该模型首先对网格资源进行预测,并对一类数据并行性网格任务进行建模,然后根据资源预测的结果对网格任务进行调度并执行。目前,国外一些研究人员对网络环境中的主机负载预测的可行性进行了一些探讨,考虑到主机负载的变化具有自相似性、长期依赖性,采用比较符合上述特性的线性时间序列模型^[8]对主机负载进行预测。其中,对线性时间序列模型的可行性研究主要从它们的预测计算成本、预测平均值、预测平均方差以及预测结果属于某一置信区间的概率等方面来进行,分析结果表明,线性时间序列模型中的自回归滑动平均(Autoregressive Moving Average, ARMA)模型与自回归(Autoregressive, AR)模型相似,具有预测性能稳定、计算成本小(占用CPU时间和网络性能)的特点,并且模型初始化时,可以使用决定数量的时间来适配且主机负载预测的未来时间比较固定,同时ARMA模型对AR模型预测主

收稿日期:2010-03-22;修回日期:2010-05-18。

基金项目:江苏省高校自然科学基金资助项目(09KJB510021);江苏省教育科学“十一五”规划课题(JSJC2009-Q08);江苏城市职业学院立项课题(09CZL008);江苏省大学生创新计划项目(2007-369)。

作者简介:程宏兵(1979-),男,江西九江人,博士,主要研究方向:网格计算、信息安全、计算机网络。

机负载时没有充分考虑的一些特性(如自相似性、长期依赖性)进行了考虑。MA模型由于需要非决定数量的时间来适配,这一点对系统设计者来说是难以接受的,并且该模型使用Powell最小化过程来减小未来预测时间段内CPU负载的预测平均方差,导致问题复杂化。对于ARIMA,ARFIMA等模型,相关研究^[8]表明,其在实际应用中产生的参数过多而导致计算成本过大,并且它们在预测时的错误分布极不正常。LAST模型对未来主机的负载预测统一为对主机历史负载衡量时的最后的值,此种模型实际应用价值不大。对于MEAN模型,其将未来的所有负载都预测成平均值,但实际情况并非如此,因此MEAN的实用性也不大。

1 网格环境资源预测建模及任务建模

网格环境中包括各种具体的资源,如CPU、网络通信资源、存储器、程序、数据等,其中CPU和网络通信资源是网格环境中决定性的资源,是本文预测的主要对象,下面首先将对其分别进行讨论,接着对一类数据并行性网格任务进行建模。

1.1 预测模型的求解及应用

本文使用加权最小二乘方法进行参数估计的ARMA模型对网格环境下主机负载进行预测,下面具体给出用加权最小二乘估计求解该预测模型的过程。

1.1.1 模型求解

描述平稳随机过程的自回归滑动平均模型ARMA, (p, q) 阶ARMA模型数学表达式为:

$$\Theta_p(B)Y_t = W_q(B)e_t$$

其中: $\Theta_p(B) = 1 - \alpha_1 B - \alpha_2 B^2 - \dots - \alpha_p B^p$, $W_q(B) = 1 - w_1 B - w_2 B^2 - \dots - w_q B^q$, e_t 是均值为零方差不为零的白噪声, $\alpha_i (i = 1, 2, \dots, p)$ 和 $w_j (j = 1, 2, \dots, q)$ 是模型的待定系数。

最小二乘方法是线性回归模型参数估计最常用的方法之一,其数学描述如下:对于下面的回归模型:

$$Y^{(p)} = \beta_1 X_1^{(p)} + \beta_2 X_2^{(p)} + \dots + \beta_N X_N^{(p)}, p = 1, 2, \dots, M$$

模型的残差:

$$e^{(p)} = Y^{(p)} - (\beta_1 X_1^{(p)} + \beta_2 X_2^{(p)} + \dots + \beta_N X_N^{(p)});$$

$$p = 1, 2, \dots, M$$

其中: $Y^{(p)}$ 为预测变量, $X_1^{(p)} \dots X_N^{(p)}$ 为影响预测的因变量;

β_1, \dots, β_N 为回归系数; N 为模型阶次, M 为样本集的数目。

$$\text{设 } \beta = [\beta_1 \dots \beta_N], X = \begin{bmatrix} X_1^{(1)} & \dots & X_N^{(1)} \\ \vdots & & \vdots \\ X_1^{(M)} & \dots & X_N^{(M)} \end{bmatrix}, Y = [Y^{(1)} \dots$$

$$Y^{(M)}]^T, E = [e^{(1)} \dots e^{(M)}]^T$$

则按如下目标函数参数优化:

$$\frac{1}{2} E^T E = \text{Min}$$

得到最小二乘估计的解:

$$\beta = (X^T X)^{-1} X^T Y$$

如果在样本集中存在某个样本异常,则通常的最小二乘方法求不出估计结果,这里采用加权最小二乘方法,将目标函数变为:

$$\frac{1}{2} (H^{-1} E)^T (H^{-1} E) = \text{Min}$$

其中 $H = \text{Cov}(E)$, 这样求得估计值: $\beta = (X^T H^{-1} X)^{-1} (X^T H^{-1} Y)$ 。

1.1.2 网格主机负载预测

应用以上方法得到主机负载预测模型ARMA:

$$(1 - \gamma_1 B^1 - \gamma_2 B^2 - \gamma_3 B^3 - \gamma_4 B^4 - \gamma_5 B^5 - \gamma_6 B^6 - \gamma_7 B^7 - \gamma_8 B^8 - \gamma_9 B^9 - \gamma_{10} B^{10} - \gamma_{11} B^{12} - \gamma_{13} B^{13} - \gamma_{14} B^{14} - \gamma_{15} B^{15} - \gamma_{16} B^{16}) \Delta_T Y(t) = e_t$$

其中: $Y(t)$ 为第 t 时段的网格中某主机负载; $\gamma_1, \dots, \gamma_{16}$ 为自回归系数; e_t 为随机干扰, T 为时段数 (T 为 1 s 或 2 s 或 5 s 或 10 s)。

在主机负载预测模型的适用性仿真实验中,在网格环境下,随机对一台某域的AMD Athlon 1600 MHz机器进行负载预测,本文采用Dinda开发的工具Load Trace Playback Tool^[9]来产生机器上的背景工作负载,考虑对两种情况下的主机负载进行预测,一种是对主机负载变化相对比较平滑的情况进行预测,实验数据见表1;另一种是对主机负载变化相对很大的情况进行预测,实验数据见表2(设两种情况下第一次预测时间分别为 t 和 τ ,实际负载记为 L ,预测负载记为 L_p ,预测误差记为 $(L_p - L)/L \times 100\%$)。

表1 ARMA模型预测负载变化比较平滑的主机

预测时间/s	CPU 负载实际值/%	基于最小二乘法		基于加权最小二乘法	
		CPU 预测值/%	CPU 预测偏差/%	CPU 预测值/%	CPU 预测偏差/%
t	55.881	57.102	2.185	56.896	1.816
$t + 15$	70.420	69.153	-1.799	69.860	-0.795
$t + 35$	83.932	85.415	1.767	85.002	1.275
$t + 50$	59.452	60.706	2.109	60.328	1.473
$t + 65$	47.024	45.331	-3.600	46.114	-1.935
$t + 75$	66.738	68.017	1.916	67.452	1.070
$t + 90$	37.825	39.021	3.162	38.694	2.297

表2 ARMA模型预测负载变化相对很大的主机

预测时间/s	CPU 负载实际值/%	基于最小二乘法		基于加权最小二乘法	
		CPU 预测值/%	CPU 预测偏差/%	CPU 预测值/%	CPU 预测偏差/%
τ	26.479	28.150	6.311	27.368	3.357
$\tau + 10$	75.490	72.308	-4.215	73.106	-3.158
$\tau + 25$	40.662	43.017	5.791	42.715	5.049
$\tau + 45$	80.201	77.404	-3.487	77.893	-2.878
$\tau + 60$	30.549	32.633	6.822	32.052	4.920
$\tau + 70$	70.837	73.724	4.076	73.240	3.392
$\tau + 90$	25.428	23.996	-5.632	24.374	-4.145

从表1、2的预测结果可以看出,本文提出的 ARMA 模型在网格主机的负载预测中具有很好的效果,尤其是采用加权最小二乘进行参数估计既提高了预测的准确性,又使模型的鲁棒性得到提高。

1.2 网格任务建模

在文中,采用一种基于时间平衡的网格任务作为模型的仿真实验用例,下面具体对这类网格任务进行建模。

对于一类数据并行性网格任务,把整个任务分成几个子任务,并按照一定的规则把它分配到各个处理器执行。此时,整个任务的执行时间等于各个子任务中最后完成的那个子任务的时间,最理想的情况是使得各个参与工作的 CPU 负载均衡,即每个子任务的处理时间都一样,表示如下:

$$E_i(T_i) = E_j(T_j); i \neq j \quad (1)$$

$$\sum T_i = T_{\text{total}}$$

其中 T_i 表示分配给处理器 i 的待处理子任务,设子任务 T_i 中存在 D_i 数据需要进行通信,可知 $D_i \subseteq T_i$, T_{total} 表示总共待处理的子任务, $E_i(T_i)$ 表示处理器 i 处理子任务 T_i 所花的时间,该时间包括处理器 i 初始化子任务、进行计算以及数据 D_i 进行通信的时间。

对于上述任务,最好的执行情况是,假设有 n 个处理器参与计算,每个处理器处理单个数据任务的时间为 $Comp_1, Comp_2, \dots, Comp_n$, 则 n 个处理器同时处理单个数据任务的时间为 $t_u = 1 / (\sum_{j=1}^n (1/Comp_j))$, 处理完整个数据任务 T_{total} 时间为 $t_u \times T_{\text{total}} = T_{\text{total}} / (\sum_{j=1}^n (1/Comp_j))$, 此时,执行时间最短。

考虑到还有其他任务竞争 CPU 资源,实际的任务执行时间可以表示为 $t_r = T_{\text{total}} / (\sum_{j=1}^n (1/Comp_j)) \times slowdown(load_{\text{CPU}})$, 其中, $slowdown(load_{\text{CPU}})$ 表示其他任务竞争 CPU 时对任务执行时间的影响。

同样,考虑到其他数据传输对网络带宽的竞争,使用 $slowdown(cap_{\text{NETW}})$ 表示其他数据传输竞争带宽时对数据传

输时间的影响。

基于上述讨论,本文对处理器有任务竞争时的任务执行进行建模,模型的好坏直接关系到调度算法的好坏。对于式(1)中 $E_i(T_i)$, 采用一种基于文献[10]改进的模型 $E_i(T_i)$, 具体如下:

$$E_i(T_i) = start_up + (T_i \times Comp_i + D_i \times Comm_i \times slowdown(cap_{\text{NETW}})) \times slowdown(load_{\text{CPU}}) \quad (2)$$

在模型 $E_i(T_i)$ 中, $start_up$ 表示初始化的时间, $Comp_i$ 表示处理单个数据的时间, $Comm_i$ 表示单位数据进行通信的时间, $slowdown(load_{\text{CPU}})$ 表示其他任务竞争 CPU 导致的对执行时间的影响; $slowdown(cap_{\text{NETW}})$ 表示其他数据传输竞争带宽时对数据传输时间的影响。本文中,用主机负载的预测值表示 $slowdown(load_{\text{CPU}})$ 中的 $load_{\text{CPU}}$; 用文献[10]中网络性能预测值四元组 (T_1, T_2, m, t) 的 t/m 表示 $slowdown(cap_{\text{NETW}})$ 中的 cap_{NETW} 。

2 RPTS 模型及其他常用任务调度方法

基于上面的研究结果本文已经获得了一些数据参数。如预测到的主机负载、预测得到的网络性能四元组以及网格任务建模后的计算和数据通信需求等一些数据参数。同时,由式(1)考虑到,一个网格任务是由一些计算密集型的子任务或数据密集型的子任务组成。本文设计并提出一种网格任务调度模型 RPTS(图1)。其基本思想是:利用 ARMA 负载预测方法(CPU_L predictor)对虚拟组织中的各域主机进行负载预测,通过四元组^[10]网络性能预测方法(Network_P predictor)对虚拟组织的网络性能进行测试,把上述最终预测结果输入到调度中心,使用调度算法对任务进行调度,考虑使用 CPU 负载及其变化较小、计算能力较大的节点对计算密集型的子任务进行调度;使用网络性能较好的节点对数据密集型子任务进行调度,然后,解出式(1),得出分配给各个处理器的子任务 T_i , 最后把子任务分配给虚拟组织中相应的主机并执行。

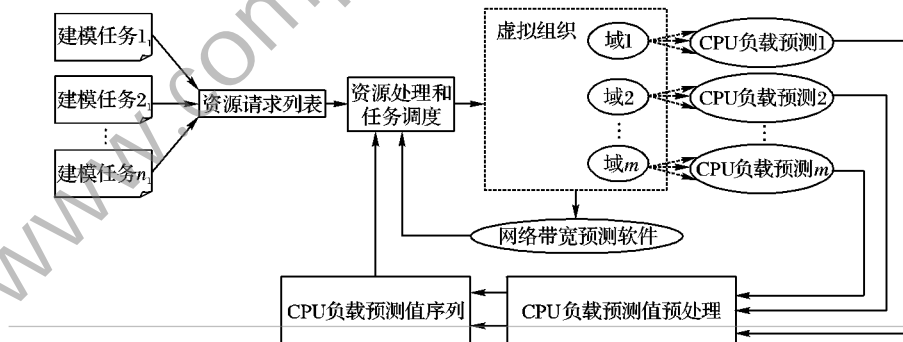


图1 基于资源预测的网格任务调度模型

2.1 RPTS 模型任务调度算法

下面首先给出 RPTS 模型中任务调度算法的一些基本概念,然后给出调度算法的描述。

设网格环境的某一虚拟组织中提供服务的不同域的数目为 m , 每个域中的主机数为 $j(j \geq 1, \text{对不同的域 } j \text{ 可以不同})$ 主机预测结果记为 $D_{i-cpu_j}(i = 1, \dots, m, j)$ (注: D_{i-cpu_j} 表示域 i 中的 j 主机的处理能力,本文中主机的处理能力表示成单位时间的指令执行数和其负载的函数形式,并与单位时间指令执行数成正比,与负载成反比,设随机预测的网络性能预测结果数目为 n , 即四元组的数目为 n , 性能预测结果记为 $netb_k =$

$(Dx_Tr, Dy_Ts, m, t)(k = 1, \dots, n)$ (注: $netb_k = (Dx_Tr, Dy_Ts, m, t)$, 其中 Dx, Dy 表示主机所在的域, $x, y \in [1 \dots m]$; T_r, T_s 表示某域中的主机, $r, s \in [1 \dots j]$, 当 $x = y$ 时, $netb_k$ 表示同一个域的两台主机终端的网络性能预测结果)。仿真实验中设网格任务以先来先服务(FCFS)的方式得到调度,并使网格任务建模结果存放在 RRL(Recourse Request List)中。

算法 RPTS 模型任务调度算法。

/* 预处理 */

1) 设置 CPU_Set = NULL, NETB_Set = NULL。其中: CPU_Set 表示主机负载预测结果集, NETB_Set 表示网络性能

预测结果集。

2) 在某一限定的时间内(注:在仿真实验中,采取对所有参与服务的主机负载进行预测,但是在具体网格应用中应是对虚拟组织中的局部主机进行负载预测,故应设置一时间限制)预测各域的主机负载,把结果转换成主机处理能力的表示形式,归入 CPU_Set 中,并对结果升序排序;随机地在各域之间进行网络性能预测,以四元组的形式表示预测结果,归入 NETB_Set 中,并对结果以域 D 为主序, m/t 为辅序进行升序排序,即先对预测结果按域排序,然后再将相同域的预测结果按照 m/t 进行升序排列。

3) 对 CPU 和网络带宽预测值进行预处理。

a) 设置长度为 n 的线性表并初始化为空,并设置表头、表尾指针;

b) 从 CPU_Set 中选择计算能力最强的主机置于表首,并使表头指针前移一位;

c) 从 NETB_Set 中查找具有表首主机为一终端的网络性能预测结果:

若存在,从结果中选择与表首主机具有最佳性能的主机,并置于表尾,并使表尾指针后移一位;否则,执行 d);

d) 重复 a) ~ b), 直到所有主机负载预测结果加入表中。

4) 读取任务等待队列,建模任务并归入 RRL (Resource Request List) 中,对于 RRL 中建模的网格任务,根据式(2)解出式(1),得到分解的各子任务 T_i (注:任务划分的子任务数目不超过预测到的主机数,在仿真实验中,主机数为虚拟组织中的主机数)的计算资源需求和通信资源需求信息,并同时计算密集型子任务按照计算资源需求进行排序;对数据密集型子任务按照通信资源需求进行排序。

/* 匹配 (Mapping) */

5) Read (RRL)。

6) 对于计算密集型子任务的资源需求序列和 3) 中的主机负载预测结果表从表首以升序的方式依次匹配;对于数据密集型子任务通信资源需求序列和 3) 中的主机负载预测结果表从表尾以降序的方式依次匹配。

/* 调度、执行 */

7) 按照 6) 中资源—子任务匹配结果进行调度、执行。

8) 以某一时间间隔重复 1) ~ 7)。该算法的主要思想是使预测资源与子任务进行有效匹配,时间主要花在主机负载的预测、网络性能预测以及对预测结果进行的预处理方面。对于资源预测,ARMA 模型建立之后,其用于实际预测的时间相对可以忽略不计。而网络性能的测试在本模型中与主机负载预测并行执行,测试是在虚拟组织的终端之间随机进行,对其服务要求是“尽力而为”。对于预处理阶段,选择合适的排序算法其时间复杂度可达 $O(n \lg n)$,对于匹配阶段其时间复杂度为 $O(n)$ 。

2.2 其他任务调度方案

1) 利用预测到的 CPU 负载值进行调度,而不考虑负载预测值的改进情况,采用随机调度方法^[2]进行任务调度,称这种模型为单步随机调度方法,记为 OSSS (One Step Stochastic Scheduling)。

2) 利用主机负载预测的历史信息 $D_{i_cpu_1}$, $D_{i_cpu_2}$, ..., $D_{i_cpu_k}$ 的平均值,并采用本文模型中的任务调度算法,称这种算法为预测平均值间隔调度算法 (Predicted Mean Interval Value, PMIS)。

3) 利用主机负载预测的历史信息 $D_{i_cpu_1}$, $D_{i_cpu_2}$, ...,

$D_{i_cpu_k}$ 的平均值,并用变化量 σ_i 对平均值进行改进,使用保守调度方法^[3]进行任务调度,称这种算法为历史平均保守调度算法 (History Mean Conservative Scheduling, HMCS)。

4) 利用主机负载的历史信息 Y_1, Y_2, \dots, Y_i 的平均值并使用其方差对之进行改进,并采用动态方法^[5]进行调度,称这种算法为历史动态调度算法 (History Dynamic Scheduling, HDS)。

5) 利用主机负载的历史信息 Y_1, Y_2, \dots, Y_i 的平均值进行调度,并采用并行调度方法^[11]进行调度,称这种算法为历史平均调度算法 (History Mean Scheduling, HMS)。

3 仿真实验

3.1 实验方法

在本节,为了检验模型的有效性,构造合适的网格服务环境^[12],参考文献[13~15]的实验方法及网格任务实现方式,在 Redhat Linux 9.0 环境下进行了一系列仿真实验,采用 Globus3.0^[16]中的工具包,并利用基于 Globus3.0 安全认证机制进行不同域间的操作。通过编程,对所提出的模型同其他常用的一些任务调度模型进行了比较,在此,称本文的任务调度模型为资源预测任务调度模型 (Tasks Scheduling Model Based on Resource Prediction, RPTS)。其中,使用 1 台 P4 3.0 GHz, 512 MB DDR 机器进行资源的预测、预处理及网格任务的调度。在 7 台机器,分为 3 个不同的域间进行了任务调度算法的比较实验,采用 Dinda 开发的工具 Load Trace Playback Tool 来产生各个机器上的背景工作负载,来模拟对 CPU 资源的竞争。考虑了四种情况下的背景 workflow: 高平均值高标准差 (High Mean Value, High Standard Error, HMVHSE) 的工作流、高平均值低标准差 (High Mean Value, Low Standard Error, HMLVSE) 的工作流、低平均值高标准差 (Low Mean Value, High Standard Error, LMVHSE) 的工作流和低平均值低标准差 (Low Mean Value, Low Standard Error, LMLVSE) 的工作流。

3.2 实验结果

下面给出仿真实验的结果,图 2(a)~(d) 是在 4 种情况背景 workflow 下各调度模型重复执行相同任务流 20 次的实验数据曲线。

表 3 对 4 种情况下的实验数据进行了统计。其中 SC (Simulation Condition) 表示仿真条件;SD (Simulation data) 表示仿真数据;SO (Simulation object) 表示仿真对象。

3.3 结果分析

从图 2 可以看出,本文提出的基于资源预测的任务调度模型 RPTS 都是较优的,实验数据曲线相对平滑,任务执行时间基本最短。从表 3 的具体数量来看,RPTS 模型算法的平均时间要比 HMCS 要少 8.46% ~ 15.32%;比 HDS 要少 9.01% ~ 16.33%;比 HMS 要少 9.86% ~ 29.29%;比 OSSS 要少 14.01% ~ 33.19%;比 PMIS 要少 14.92% ~ 17.81%。对于标准差,在高负载的情况下,RPTS 模型算法的标准差明显小于其他算法,例如,比 HMCS 要少 14.09% ~ 136.65%,比 HDS 要少 144.46% ~ 204.89%,比 HMS 要少 52.10% ~ 139.94%,比 PMIS 要少 123.01% ~ 416.63%。但是在负载较小时,这种优势不太明显。对于每种模型算法的标准差与平均值的比值,RPTS 是 2.32% ~ 3.98%,HMCS 是 2.37% ~ 6.11%,HDS 是 2.01% ~ 8.69%,HMS 是 0.09% ~ 7.88%,OSSS 是 0.97% ~ 2.90%,PMIS 是 2.66% ~ 12.58%,可见,RPTS 模型算法有较好的稳定性。

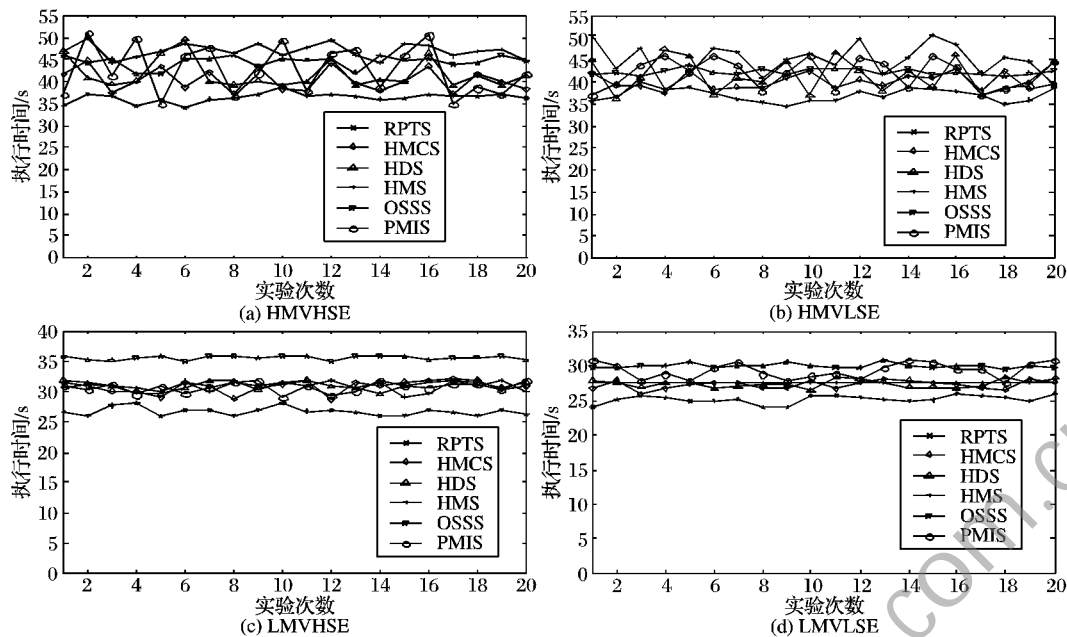


图 2 4 种情况背景 workflow 下的实验数据曲线

表 3 4 种情况背景 workflow 下的实验数据统计

SC	SD	SO					
		RPTS	HMCS	HDS	HMS	OSSS	PMIS
HMOVHSE	Minimum cost/s	33.973 658	37.125 856	39.101 252	44.258 562	41.625 038	35.034 851
	Maximum cost/s	38.724 323	44.896 562	49.520 306	49.885 623	45.996 585	50.968 562
	Mean cost/s	36.315 061	40.369 178	41.699 869	46.950 514	44.480 192	42.784 413
	Mean standard error	1.042 160	2.466 319	3.177 484	1.585 173	1.289 167	5.384 123
HMOVLE	Minimum cost/s	34.403 261	37.524 103	37.020 589	38.363 035	41.030 910	37.102 362
	Maximum cost/s	39.969 433	43.070 126	47.465 008	50.668 523	43.852 039	46.125 203
	Mean cost/s	37.081 423	40.108 933	41.492 775	44.943 296	42.277 070	42.004 455
	Mean standard error	1.474 424	1.682 217	3.604 318	3.537 739	0.669 577	3.288 180
LMOVHSE	Minimum cost/s	25.885 623	28.723 584	29.710 209	29.250 211	35.023 445	29.185 212
	Maximum cost/s	28.024 324	32.025 850	31.983 253	31.978 020	35.998 708	32.679 244
	Mean cost/s	26.736 161	30.832 467	31.103 270	30.913 642	35.609 882	30.725 483
	Mean standard error	0.631 727	0.997 546	0.729 145	0.777 425	0.342 828	0.817 099
LMOVLE	Minimum cost/s	24.012 341	26.058 963	26.501 232	27.620 121	29.496 831	27.654 706
	Maximum cost/s	26.032 526	28.158 576	28.345 237	27.697 801	30.776 234	30.896 087
	Mean cost/s	25.183 422	27.312 832	27.451 682	27.665 515	29.998 552	29.378 156
	Mean standard error	0.585 569	0.647 120	0.551 759	0.024 997	0.299 991	1.077 084

4 结语

本文提出了一种基于资源预测的网络任务调度模型 RPTS,该模型先对网络资源进行有效预测;同时对一类数据并行性网络任务执行进行资源需求建模,然后根据资源预测结果和任务执行资源需求建模结果进行资源的预处理、匹配、调度并执行。最后与其他一些常见调度算法进行了一系列的仿真实验比较,结果显示了 RPTS 模型具有任务执行时间较短和稳定性较好的特点,尤其是当 CPU 负载高度变化的时候,RPTS 模型显示了很强的适应性。下一步的工作是:对 CPU 负载和网络带宽的概率统计特性进行更加精确的数学描述,同时把网络计算环境中其他一些资源考虑进来,从而使 RPTS 模型更加精确、全面;另外,对模型的适应性作进一步的研究,特别是对多域情况下的域及域中主机的自由加入和退出等实际网络应用中可能出现的问题进行深入的研究。

参考文献:

[1] FOSTER I, KESSELMAN C. The grid: Blueprint for a new computing infrastructure [M]. 2nd ed. San Francisco, California: Morgan

Kaufmann Publishers, 2005.
[2] SCHOPF J M, BERMAN F. Stochastic scheduling [C]// Proceedings of the 1999 ACM Conference on Supercomputing. New York: ACM Press, 1999: 48-58.
[3] YANG L Y, SCHOPF J M, FOSTER I. Conservative scheduling: Using predicted variance to improve scheduling decisions in dynamic environments [C]// Proceedings of the 2005 ACM Conference on Supercomputing. New York: ACM Press, 2005: 454-461.
[4] 金海,陈刚,赵美平.容错计算网络作业调度模型的研究[J].计算机研究与发展,2004,41(8):1382-1388.
[5] ZHANG J, LU X D. A dynamic job scheduling algorithm for computational grid [C]// Proceedings of the 2nd International Workshop on Grid and Cooperative Computing. Shanghai, China: [s. n.], 2005: 40-47.
[6] 查礼,徐志伟,林国璋,等.数据和计算密集混合元任务的网格调度算法[J].计算机工程与设计,2003,24(10):1-4.
[7] 杨庚,王绍棣,沈金龙.基于曙光并行机的超大规模非线性方程组并行算法研究[J].计算机学报,2002,25(4):397-402.

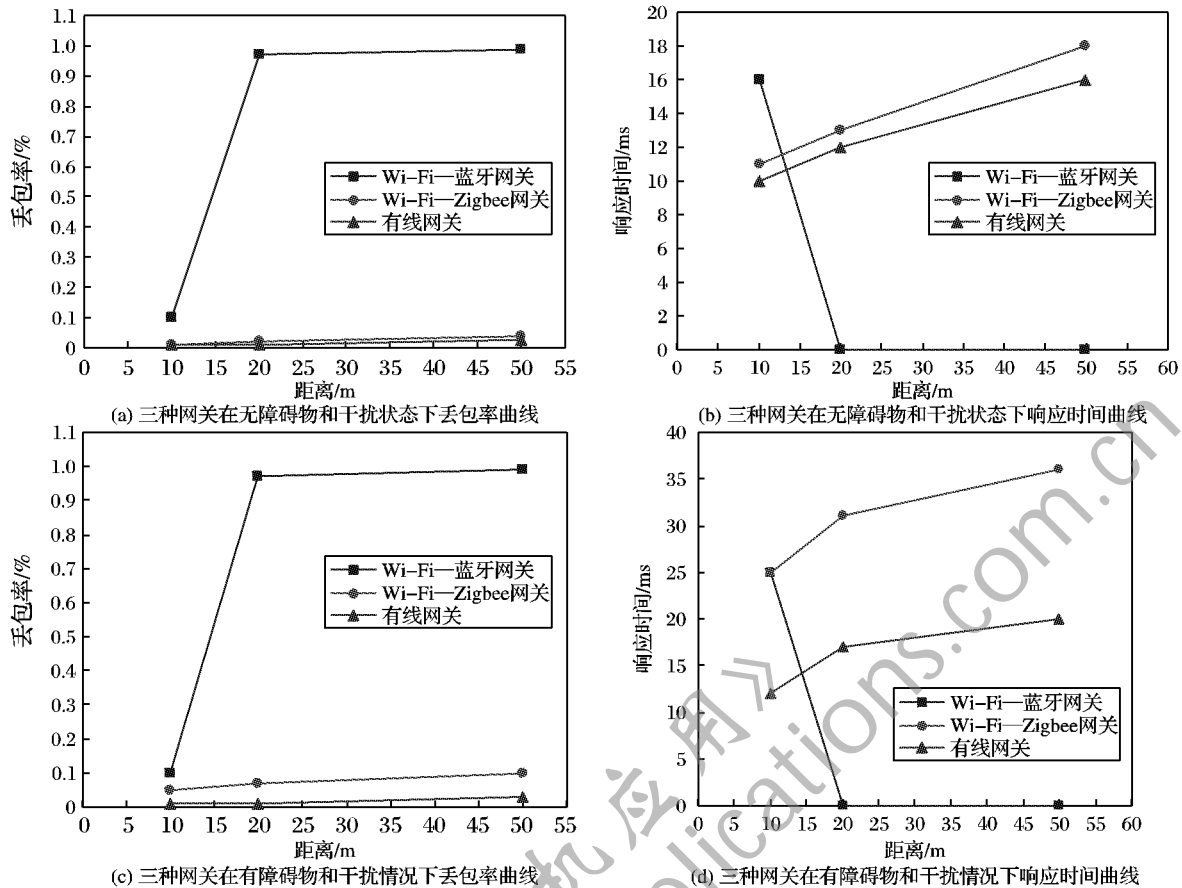


图8 三种网关在不同环境下丢包率与响应时间的比较

参考文献:

- [1] Samsung Corporation. S3C2440A 32-bit RISC microprocessor user's manual [EB/OL]. [2009-10-12]. <http://www.samsung.com>.
- [2] Jennic Corporation. Preliminary data sheet-JN5139-1v3 [EB/OL]. [2010-01-11]. <http://www.jennic.com>.
- [3] 彭建, 崔更申, 陈约林. ZigBee 无线网络协议层在 Linux 系统中的构建[J]. 传感器与微系统, 2007, 26(8): 69-71.
- [4] 胡彧, 杨琳. 基于自定义家电管理协议的家庭 Web 网关实现[J]. 计算机工程, 2009, 35(5): 271-274.
- [5] 黄丽莹. 骨干无线网状网的排队延迟性能研究[J]. 计算机应用, 2009, 29(8): 2179-2182.
- [6] 王殊, 阎毓杰, 胡富平, 等. 无线传感器网络的理论及应用[M]. 北京: 北京航空航天大学出版社, 2007.
- [7] KIM S H, KANG J S, PARK H S, *et al.* UPNP-ZigBee Internetworking architecture mirroring a multi-hop ZigBee network topology [J]. IEEE Transactions on Consumer Electronics, 2009, 55(3): 1286-1294.
- [8] 徐勇军. 低速无线个域网实验教程[M]. 北京: 北京理工大学出版社, 2008.
- [9] YU M C, SHIN D, SHIN D K, *et al.* Fundamentals and design of smart home middleware [C]// CSO 2009: International Joint Conference on Computational Sciences and Optimization. Washington, DC: IEEE Press, 2009: 647-650.
- [10] DARIANIAN M, MICHAEL M P. Smart home mobile RFID-based Internet-of-things systems and services [C]// ICACTE 2008: International Conference on Advanced Computer Theory and Engineering. Washington, DC: IEEE Press, 2008: 116-120.
- [8] DINDA P A, O'HALLARON, HOST D R. Host load prediction using linear models [J]. Cluster Computing, 2000, 3(4): 132-145.
- [9] DINDA P. Playload: A load trace playback tool [EB/OL]. [2009-12-28]. <http://www.cs.northwestern.edu/~pdinda/LoadTraces/>.
- [10] RIPEANU M, IAMNITCHI A, FOSTER I. Performance predictions for a numerical relativity package in grid environments [J]. International Journal of High Performance Computing Applications, 2007, 15(4): 375-387.
- [11] KUMAR S, DAS S K, BISWAS R. Graph partitioning for parallel applications in heterogeneous grid environments [C]// International Parallel and Distributed Processing Symposium. Florida, USA: [s. n.], 2004: 787-792.
- [12] 胡春明, 怀进鹏, 孙海龙. 基于 Web 服务的网格体系结构及其支撑环境研究[J]. 软件学报, 2004, 15(7): 1064-1073.
- [13] YANG G, RONG CH-M, DAI Y P. A distributed honeypots for grid security [C]// Proceedings of the 2nd International Workshop on Grid and Cooperative Computing. Shanghai: [s. n.], 2003: 1083-1087.
- [14] RANGANATHAN K, FOSTER I. Simulation studies of computation and data scheduling algorithms for data grids [J]. Journal of Grid Computing, 2003, 1(1): 53-62.
- [15] IVERSON M A, OZGUNER F, FOLLEN G J. Run-time statistical estimation of task execution times for heterogeneous distributed computing [C]// Proceedings of the 5th IEEE International Symposium on High Performance Distributed Computing. Washington, DC: IEEE Computer Society, 1996: 132-139.
- [16] Globus Alliance. Globus toolkit [EB/OL]. [2009-12-28]. <http://www.globus.org/toolkit/>.

(上接第 2534 页)