基于服务的两级协同分布式调度策略

张小松，刘 丹，李毅超

(电子科技大学计算机科学与工程学院 成都 610054)

【摘要】提出一种全局及局部两级协同调度算法。全局调度以具体服务状态和系统状态为决策依据选择服务节点，提高调度精度；局部调度在全局调度基础上，调整服务质量和服务状态以确保零拒绝率，并回馈服务状态以提高全局调度的正确性。理论分析和仿真结果表明该算法在保持同等效率前提下，获得比传统调度算法更高的可靠性。

关 键 词 分布式调度; 可靠性; 服务状态; 系统状态

中图分类号 TP316.4 文献标识码 A

A Two-Level Cooperation Distributed Scheduling Algorithm Based on Services

ZHANG Xiao-song，LIU Dan，LI Yi-chao

(School of Computer Science and Engineering, UEST of China Chengdu 610054)

Abstract A global and local two-level cooperation scheduling (TCS) algorithm is presented. In order to improve the scheduling precision, the global scheduling considers both the service states and system states to choose a suitable site to serve; Based on the global scheduling, the local scheduling adjusts the service quality and service states to guarantee no rejection of client request, and feeds back the service states to improve the correctness of global scheduling. The theoretical and simulated results show that, with the similar efficiency, the proposed algorithm owns higher reliability compared with the traditional distributed scheduling algorithms.

Key words distributed scheduling; reliability; server states; system states

分布式并行技术的核心在于设计良好的分布式并行调度算法。传统调度算法大多基于负荷平衡的调度算法[1~4]，这类算法假设系统节点仅具有两状态，如果在节点出现既非完全正常也非完全失效的不稳定状态时会引起误调度。本文提出一种综合考虑系统状态和服务状态的两级协同调度(Two-level Cooperation Scheduling, TCS)算法来保证系统的高可靠性，可以有效避免这种情况的发生。

1 状态模型及调度算法

1.1 系统状态及服务状态

节点状态由系统状态和服务状态组成。

定义 1 节点S_i，其系统状态定义为G_i。$G_i = \{T_s, T_w, T_e\}$。T_s表示S_i为轻载或适载状态；T_w 表示S_i为重载状态；T_e表示S_i上的管理器处于异常状态。当$G_i = T_s$，则S_i还有充足的空闲资源如CPU、RAM等；当$G_i = T_w$, 则

收稿日期：2004 – 05 – 17

基金项目：四川省科技攻关资助项目(02GG006)

作者简介：张小松(1968 –)，男，硕士，讲师，主要从事计算机网络、计算机信息安全、嵌入式应用方面的研究；刘 丹(1968 –)，男，博士，讲师，主要从事计算机网络方面的研究；李毅超(1969 –)，男，副教授，主要从事计算机网络、计算机信息安全方面的研究.

S_i上的系统资源已消耗殆尽，S_i不应接受更多任务；当$G_i = T_e$，则应重启S_i上的管理器或重启系统。

定义 2　节点S_i上服务器P_i的状态定义为$P_{i(Si)} = \{A_s, A_d, A_t, A_z\}$。$A_s$表示$P_i$为轻载或适载状态；$A_d$ 表示P_i为降级服务态；A_t表示P_i为异常状态；A_z表示P_i为僵死状态。

当$P_{i(Si)} = A_s$，则P_i的功能和性能都正常；当$P_{i(Si)} = A_d$，表示P_i负荷过大，其功能和性能均下降，此时P_i以降级性能的方式继续已接受的任务，一个新到达的任务不应再分配给P_i。P_i的负荷过重并不意味着系统负荷过重，即此时系统仍可能是$G_i = T_s$的情况。

当$P_{i(Si)} = A_t$，则P_i进程出现了某些异常情况，如因$G_i = T_w$，系统资源过于紧张导致P_i运行需要的资源紧张，或P_i的一个或多个线程因同步互斥等原因而处于睡眠状态等。从用户的角度看，此时P_i已不响应任何命令但并未僵死，其已接受的服务将在暂停一会后继续得到服务，但不能再将新任务分配给P_i服务，否则$P_{i(Si)}$可能由A_t迁移到A_z。

当$P_{i(Si)} = A_z$，P_i已不能进行任何服务而必须重启，其已接受的任务应迁移到有相同服务的其他节点。

$P_{i(Si)}$和G_i有很大相关性。如，当$G_i = T_s$时，$P_{i(Si)} = A_s$的概率很大。但当$G_i = T_e$ 时，并不意味着$P_{i(Si)} = A_z$。TCS算法同时考虑G_i和$P_{i(Si)}$来作为全局调度的基本信息，避免误调度，提高系统可靠性。

定义 3　定义分布式系统的事件集合 $\Sigma = \{PE_w, PE_l, E_r, GE_w, GE_l\}$。$PE_w$ 表示P_i由适载变为重载，PE_l表示P_i由重载变为适载，E_r表示P_i运行中出现错误，GE_w表示G_i由T_s变为T_w，GE_l表示G_i由T_w变为T_s。

1.2　状态迁移

图1所示为P_i的状态迁移图。P_i的状态迁移函数定义为：

1) $d\{A_s, PE_w\} = A_d$；2) $d\{A_d, PE_l\} = A_s$；3) $d\{A_s, GE_w\} = A_t$；4) $d\{A_t, GE_l\} = A_s$；5) $d\{A_d, GE_w\} = A_t$；6) $d\{A_t, GE_l\} = A_d$；7) $d\{A_s, E_r\} = d\{A_t, E_r\} = d\{A_d, E_r\} = A_z$。

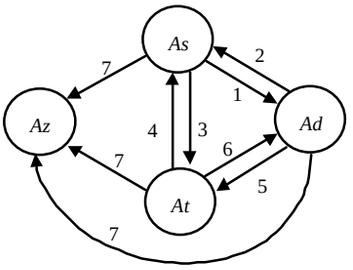

图1　Pi的状态迁移

1.3　状态拾取

调度模块包括协调器和监视器两个相互独立并相互监视的模块，当其中一个模块异常时，将由另一个模块重启动它。协调器的作用是获取并维护系统状态。监视器包含多个探测器，它负责获取并维护服务状态。对不同的服务进程需定制各自的监视器及探测器。探测器检测服务状态时，首先直接通过操作系统接口获得服务器进程的状态，如运行态、就绪态、阻塞态等，然后发送仿真客户请求给P_i以测试其性能参数。不同服务的服务参数不同，因此需对不同服务定制探测器。若探测器测试P_i的性能参数超出预定的正常范围，则P_i应采取降级服务的措施以避免服务失效，即$P_{i(Si)} = A_d$。

1.4　调度算法

调度算法在协调器中实现，协调器由算法模块和状态迁移模块组成。当一个请求到达时，算法模块根据当时系统和服务状态选中一个适合的服务节点。状态迁移模块负责周期性拾取系统状态和服务状态。

不失一般性，考虑一个新到的请求服务X，S_i表示节点i，P_i表示运行于S_i上且提供服务X的进程。

算法模块为：1) 获取具有X服务的节点集Ω_x；2) 若$\Omega_x = \Phi$，则返回失败；3) 从Ω_x中获取提供服务X且服务状态为A_s的节点集Ω_{x1}；4) 若$\Omega_{x1} \neq \Phi$，则从中选取系统负荷最轻的节点S_i ；(1) 若$G_i = T_s$，返回成功；(2) 若$G_i = T_e$，执行下一步；5) 从Ω_x中获取提供服务X且服务状态为A_d的节点集Ω_{x2}；6) 若$\Omega_{x2} \neq \Phi$，则从中选取系统负荷最轻的节点S_i ，(1) 若$G_i = T_s$，返回成功，(2) 若$G_i = T_e$，返回失败。

状态迁移模块该模块通过监视器周期性地获取系统状态和服务状态并实现状态迁移：1) 当监视器检测到服务X的服务质量(Quality of Service, QoS)超出正常范围，置$P_{i(Si)} = A_d$，返回；2) 当监视器检测到服务X的QoS在正常范围，置$P_{i(Si)} = A_s$，返回；3) 当S_i系统状态G_i变为T_w，置$P_{i(Si)} = A_t$，返回；4) 当S_i系统状态G_i变为T_s，(1) 若服务X的QoS超出正常范围，则设置提供服务X的进程P_i 的状态$P_{i(Si)} = A_d$，返回，(2) 若服务X的QoS在正常范围，则设置提供服务X的进程P_i 的状态$P_{i(Si)} = A_s$，返回；5) 当S_i系统状态G_i变为T_e，将S_i服务的所有任务迁移到其他节点，重启S_i，结束。

2 实例研究

将算法思想具体应用于分布式并行视频点播(Video on Demand, VOD)服务器系统中，应用的测试平台如图2所示。该测试平台由并行调度器[5]、并行VOD服务器和客户仿真器构成。系统中每个节点运行一个协调器、一个监视器和一个Darwin Streaming Server服务器。

并行调度器采用1.3节的算法。S_i上的协调器周期性地获取其系统状态，同时S_i上的监视器周期性地获取运行于其上的Darwin Streaming Server服务器状态。Darwin Streaming Server的状态可通过系统调用获得，而Darwin Streaming Server的服务质量按下面方法获取。

Darwin Streaming Server的响应时间Δt定义为：探测器在t_1时刻发送一个模拟请求给Darwin Streaming Server，在t_2时刻获得视频数据流响应，则$\Delta t=|t_2-t_1|$。设正常响应时间范围$\Delta t_n \in [T_l, T_u]$，对$S_i$，当$\Delta t_i \notin [T_l, T_u]$，表示服务质量下降。而由于分布式系统状态的不精确性，可能导致少量的误调度到这类节点，按照传统调度算法，此时会拒绝新任务导致任务实效。本文算法中，置Darwin Streaming Server的状态为A_d。同时，在A_d状态下重新定义正常响应时间范围$\Delta t_{nd} \in [T_{ld}, T_{ud}]$。当一个新的请求被分配到$S_i$，服务是否正常将按照新的响应时间范围判决。此时，为每个请求任务所付出的资源开销都比正常情况减小，尽可能以降级服务的形式满足被分配来的任务，而不导致任务失效。

在图2平台上进行了8组实验，服务器为同构型，节点数从1～8。每组实验都预先给各服务器增加一定的负荷，同时发起一组VOD请求。将本文算法和传统的算法进行比较测试，并对失效服务进行记录，统计结果如图3所示，图中表明本文算法具有更高的可靠性。

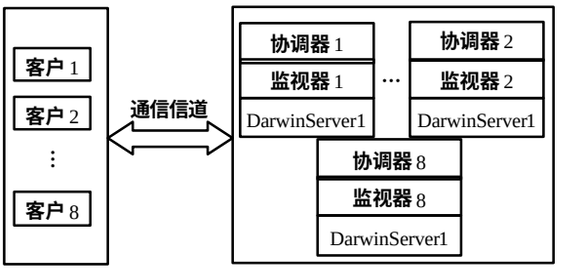

图2 VOD测试平台

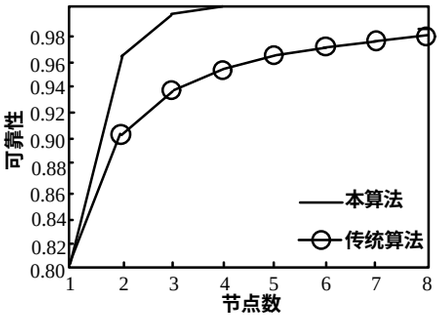

图3 可靠性分析

3 性能分析

分布式系统可靠性[5]：$R_{N1}=1-(1-p)^N$。式中，N为系统中的节点数，p为一个节点正常工作的概率。它的成立条件为当一个节点上服务器运行异常时，系统可以准确及时地获得该信息并不将新请求分配给故障节点服务。传统算法以管理器状态为判决依据，会产生一些误调度导致系统可靠性降低。这时其可靠性定义为：

定义 4 未知状态当一个服务器状态不为A_s 或 A_d，且系统管理进程仍然认为该服务器处于可服务状态，则定义该服务器处于未知状态。

就传统算法而言，在一个N节点的分布式系统中。假设服务进程X运行于N个节点上，P_x代表X在一个节点上为非未知状态的概率。则按平均概率计算，一个请求被分配到一个未知态服务器上的概率为$(1-P_x)/N$。此时，请求将会失效，因此可靠性为$R_{N2}=1-(1/N)(1-P_x)$。本算法考虑了服务进程的具体信息，避免了将请求误调度到未知态服务上，其获得的系统可靠性为R_{N1}，而获得的系统可靠性为R_{N2}。

4 结 论

本文给出了一个高可靠性的调度算法。它采用了两级调度机制，并以系统状态和服务状态为调度依据。该算法在VOD系统的服务性能测试中，表现出良好的可用性。算法可应用于各类分布式系统中，尤其适应于现今宽带网络服务器群高可靠、高吞吐量的要求。

(下转第232页)

表3 网络延迟和软件开销的模拟结果

数据量/KB	网络延迟/μs·B^{-1}	软件开销/μs·B^{-1}
64	1.526	0.419
128	1.488	0.382
256	1.526	0.363
512	1.479	0.363
1 024	1.469	0.358
2 048	1.373	0.375

另外，随着传输数据量的增加，每字节所需的软件开销便逐渐减小，所以在松散网络结构中大粒度是获取高效率的必要手段。

4 结束语

通过提供的LogP简化模型具体环境中的参数模拟，针对单机和多机的模拟结果，分析了其网络延迟和软件开销，得到其经验公式。因此可以确定LogP参数在该环境中的简化是正确而有效的。因此简化的LogP模型可以有效而正确地简化实际并行算法的设计与分析。

参 考 文 献

[1] 莫则尧, 李晓梅. 工作站网络环境下的并行计算[J]. 计算机学报, 1997, 20(6): 34 -37

[2]Matthew I, Agarwal A. LoPC: modeling contention in parallel algorithms[C]. ACM 0-89791-906-8/97/0006. Portland, Oregon, United States, 1997. 56-69

[3] Keeton K, Patterson D A, Anderson T E. LogP Quantified: the case for low-overhead local area networks [C]. In Hot Interconnects III, San Francisco, California, United States, 1995. 82-84

[4] Alexandrov A. Ionescu M F. LogGP: incorporating long messages into the LogP model for parallel computation [C]. Journal of Parallel and Distributed Computing 44, Ottawa, Canada, 1997. 36-39

[5] 孙家昶, 张林波. 网络并行计算与分布式编程环境[M]. 北京: 科学出版社, 1996

编 辑 熊思亮

(上接第212页)

参 考 文 献

[1] Kunz T. The influence of different workload descriptions on a heuristic load balancing scheme[J]. IEEE Transactions on Software Engineering, 1991, 17(7): 725 -730

[2] Pai V S, Aron M, Banga G, et al. Locality-aware request distribution in cluster-based network servers[C]. In Proceedings of the Eighth International Conference on Architectural Support for Programming Languages and Operating Systems(ASPLOS-VIII), San Jose, California, 1998

[3] Aron M, Druschel P, Waenepoel W Z. A mechanism for resource management in cluster-based network servers[C]. In Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, Santa Clara, USA, 2000

[4] Castro M, Dwyer M, Rumsewicz. M Load balancing and control for distributed world wide web servers[C]. In Proceedings of the 1999 International Conference on Control Applications, Kohala Coast, Hawaii, 1999

[5] Liu Xin-song. The performance research of the distributed parallel server system with distributed parallel I/O interface[J]. ACTA Electronica Sinica, 2002, 30(12): 1 808-1 811

编 辑 漆 蓉