一种支持实时集群系统的双层调度机制*)

陈泽晖　常光辉　卜长清　陈蜀宇
（重庆大学　重庆 400044）

摘　要　提出一种集群环境下的实时调度机制，它通过前端负载机平衡调度、后端机区分实时和非实时任务队长差调度而提供实时优先服务，有效提高了以往前端机单点调度模式下的集群节点利用率和实时任务响应性。对比实验表明，在保证吞吐率的情况下，实时任务的响应时间明显减少，为分布式实时调度提供了一种可行的设计参考。

关键词　实时集群，双层调度，实时响应特性

Two-level Scheduling Mechanism Supporting Real-time Cluster System

CHEN Ze-hui　CHANG Guang-hui　BU Chang-qing　CHEN Shu-yu
(Chongqing University, Chongqing 400044, China)

Abstract　We propose a real-time scheduling mechanism in cluster system, which provides real-time priority services by combining load balancing scheduling in front-end and scheduling algorithm based on length difference between real-time and non-real-time task queue in back-end. The new scheduling mechanism increases the utilization of cluster nodes in case of scheduling only in front-end and enhances the response of real-time tasks. As is shown in the comparison experiments, the response time of real-time tasks is decreased observably under the condition of generating guaranteed throughput, the mechanism provides a viable reference for the designing of distributed real-time scheduling.

Keywords　Real-time cluster, Two-level scheduling, Real-time response characteristic

1　引言

高速网络系统和技术成熟、性价比高的商用处理器的出现，使得集群体系成为高性能计算和重大应用领域的强有力的竞争者[1]。集群结构是基于开放体系标准，采用规模化生产的通用商业产品，具有很好的性价比，且能根据用户要求对集群规模进行扩展，最充分地满足用户对计算能力的要求[2-4]。在传感器网络(Sensor Network)[5,6]这类环境中，中央服务器不仅需要支持大容量、高密度的荷载，更需要对具有不同时限要求的传感器事件提供即时响应，如监测战场敌方车辆移动的传感器信息就应得到最优先的服务，对这类重要信息的响应延迟将导致灾难性的后果。所以集群的实时调度成为一个有重要实用价值的问题。

现有的集群大部分是属于科学计算的高性能集群[7]，还有一种是目标为提高集群系统效率的单一负载平衡集群，如LVS[8]。这两种设计方法都没有提供带有优先级实时区分特性的任务服务机制。另外，在实时集群计算中，也有人提出了集中调度的方法[9]，这种方法是在集群的前端调度机上区分实时和非实时任务，将它们分别派发给后端相对应的各自独立的实时服务器池和非实时服务器池，其优点是实现相对简单，对实时任务的响应具有优良的保证，但它最大的缺点是在某种任务空闲时完全浪费了相对应服务器池的计算资源，导致这种设计方法的性价比降低，在实际应用中难以推广。除此之外，集中调度策略在前端机做所有的调度工作，包括负载平衡和实时区分，导致前端机易成为系统瓶颈，降低了系统可扩展性[10]。

2　双层调度机制的集群系统结构

双层调度策略将实时调度这一模块移到了后端机服务器之上。并且，后端机不再区分实时和非实时服务器池，每个服务器既服务实时任务同时也服务非实时任务。这样，后端机就承担了区分实时、非实时任务且保证实时任务优于非实时任务服务的任务。然而，前端机仍然具有调度功能，这种情况下，前端机的调度需要解决的任务是集群系统的负载平衡，也就是说能够将外部发来的任务均匀地送到后端每个服务器，实现整个集群系统后端节点的负载平衡。这就是双层调度的设计思想。系统结构如图1。

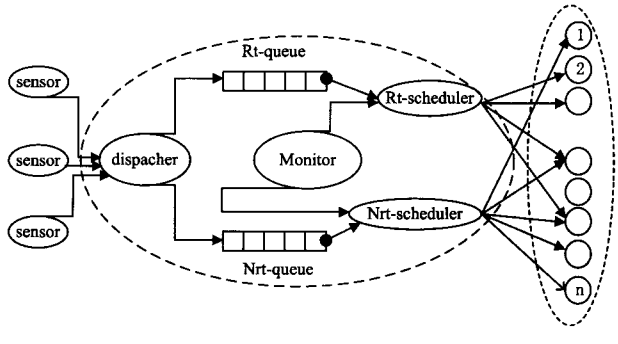

图1　双层实时调度集群系统结构

3　前端机组件的工作原理

由上一节所述，前端机的主要任务是把从外部接受到的任务负载平衡均匀派发到后端服务器。每次派发之前，可以

*)由教育部新世纪优秀人才支持计划(NCET-04-0843)资助。

先检查后端机的负载情况，然后将任务派发到挑选出来的一个负载最轻的后端服务器，这就实现了前端机的对系统的负载平衡调度。实际中，我们可以采用最小连接数算法：

(1)第一次先均匀地将队列中的任务作为试探任务发送到后端机，此后端机上的连接数加 1；

(2)当后端机有回复的时候，相应后端机上的连接数减 1，保存后端机连接数信息；

(3)找到最小连接数的后端机作为派发对象机；

(4)发送任务到派发对象机，转到(2)。

连接数的定义：连接数就是后端机上的实时与非实时任务的总数。由于在回复了任务之后，后端机上的任务数将出现变化，于是我们可以将任务发出去的时候在前端机对后端机的连接数进行计数，每发送一个任务就在相对应的后端机计数器加 1，每回收一个任务减 1。这样，始终在前端机都保持有后端机的实际连接数记录。借鉴集中调度策略的设计方法，在前端机存在两个线程：接受线程、转发线程。于是，调度器的转发线程就可以根据当前的记录选取有最短连接数的后端机作为派发对象。

在数据结构方面，需要一个映射数组来存储后端机连接数。这里，下标是有明确意义的，它代表了对应的后端机号。根据下标，也就是后端机号来存储连接数(图 2)。

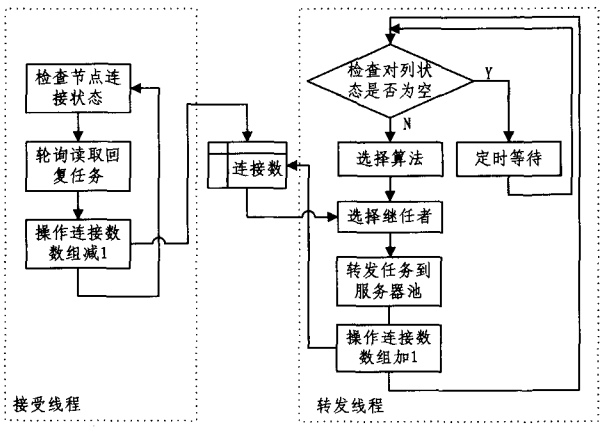

图 2　双层调度策略下前端机工作流程图

其中，连接数共享存储区域由两个线程采用互斥方法共同维护，以提供系统的负载情况信息，以便调度器可以根据当前的负载信息对任务多出调度。

4　后端服务器的设计

双层调度策略下的后端机设计相对要复杂些，在其上维持了两个任务队列 Nrt_queue，Rt_queue，这两个队列中的任务来自前端机。后端机上的调度器仍然要对其进行第二层调度，调度算法采用长度差值算法(qld)。该算法原理为：令 Nrt_queue 队列的长度为 nl，Rt_queue 队列的长度为 rl，当($nl-rl$)$<C$($0<C$ 为一个常数)时，指服务器中实时任务与非实时任务的差值已经超过一个设定好的阈值，则从实时任务队列中取一个任务，移交给服务线程对其处理；当系统满足($nl-rl$)$<C$ 条件时，将会一直服务实时任务，则必定某一个时刻($nl-rl$)$<C$ 将不被满足，此时就轮到非实时任务执行。特别地，当 $rl=0$ 时，表明系统中已经没有实时任务。那么也将会有一个时刻 $nl-0<C$，若按照前述的算法调度实时任务就会发生紊乱，因为系统中已经没有实时任务可处理。因此，每次调度前应检查 $rl=0$ 这个条件。之后检查 $nl=0$ 这个条件。若满足，则系统进入空等状态，直到有任务进入。

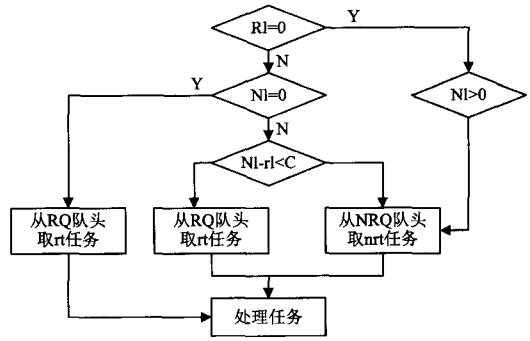

图 3　队列长度差值算法

由以上分析可知，在后端机服务器调度中实时任务获得了高优先权，所以满足软实时调度需求。图 3 为该算法流程图。

双层调度策略下的后端机服务器组件分为两部分：读取线程、服务线程。尽管在这种模式下后端机多加进了调度任务，但是我们仍然把它作为一个模块放在服务组件部分。现在描述一下后端服务器的工作内容：

(1)从网卡读取任务；

(2)从任务消息中析取区任务类型；

(3)将区分后的任务排入响应队列中；

(4)采用队长差算法调度任务；

(5)处理任务；

(6)回送任务到前端机。

事实上，集中调度策略下后端机只有一个队列，因为后端机不存在调度任务的工作，只需要将队列中任务取出服务即可。然而在双层调度模式下，后端服务器增加了一个队列，用以区分实时和非实时任务。前面从网卡读取任务都是一样的，在排队的时候却要先从任务消息中析取任务的类型。这是因为前端机往每一个后端机上转发的时候并不会区分实时非实时服务器，所以所有的后端机上面收到了两种任务类型。这两种任务类是嵌入在任务消息中，采用一个消息析取函数就可以得到它的任务类型。队长差调度算法前面已经做了详细的介绍。下面给出双层调度策略下后端机服务器的组件工作流程图(图 4)。

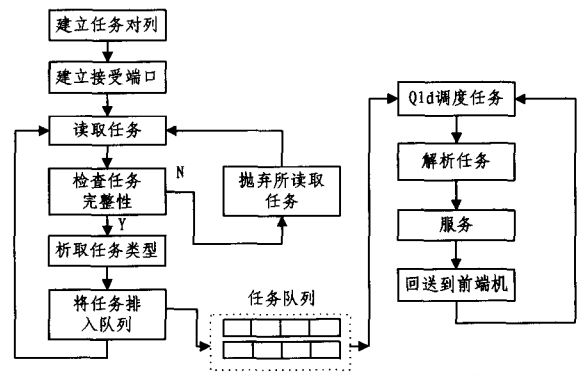

图 4　后端服务器工作流程图

图 4 详细描述了后端服务器的工作原理和实现过程，在此不再赘述。

(下转第 114 页)

[8] Jin X, Wong W-C, Chan S, et al. Serving dynamic groups in application-level multicast // High Performance Switching and Routing, 2005. HPSR. 2005 Workshop. May 2005: 432-436

[9] Yiu W, Wong K, Chan S, et al. Lateral error recovery for media streaming in application-level multicast. Multimedia, IEEE Transactions, April 2006, 8(2): 219-232

[10] Tian R X, Zhang Q, Xiang Z, et al. Robust and efficient path diversity in application-layer multicast for video streaming. Circuits and Systems for Video Technology. IEEE Transactions, 2005, 15(8): 961-972

[11] Xie X R. Computer Network. 4th edition. Beijing: Publishing House of Electronics Industry, 2005: 412-419

(上接第 105 页)

5 对比实验与分析

利用实验室前期开发的 Linux 平台下集群中间件 RTCS,在局域网环境中搭建了模拟实验平台,进行了对比验证实验。

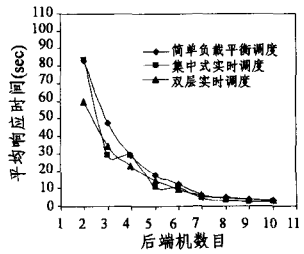

图 5 传输率在 1.54kbps 下三种调度的平均响应时间对比

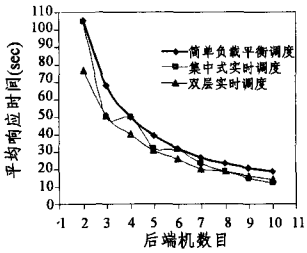

图 6 传输速率在 5.84kpbs 下三种调度的平均响应时间对比

实验的目的主要是考察双层调度机制与一般的简单负载平衡调度在时间响应上的优势。以及与现有集中式实时调度算法在资源利用率方面的比较。

表 1 响应时间下降百分比(与简单负载平衡调度比较)

后端机数	2	3	4	5	6	7	8	9	10
集中调度	0.72%	19.50%	0.25%	16.33%	0.47%	15.23%	17.96%	27.64%	34.24%
双层调度	22.21%	24.16%	21.23%	24.45%	18.65%	25.78%	21.67%	23.55%	26.78%

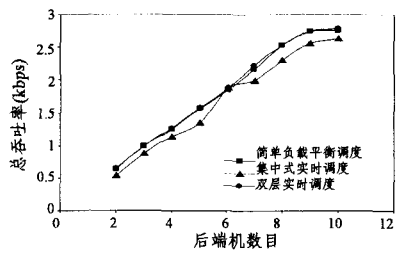

图 7 传输速率为 5.84kpbs 的三种调度策略吞吐率对比

由图 5,6 和表 1 可以看出,两种实时调度算法在时间响应方面明显优于负载平衡算法,其中集中式调度策略在小数目后端机情况下的波动比较大,呈现交替状态。这可能是验证程序在集中式调度模式下存在某一方面的不足导致了性能表现的不稳定。同时也表明,前端机全部承担调度任务,加大了前端机设计与实现的复杂性。然而,双层调度算法则相对来说表现平稳,说明双层调度策略分解了系统整体设计的复杂性。从表 1 中我们还看到,当节点数为 10 的时候,双层调度策略的实时响应性与集中调度相比较略有下降,证明了集中式调度算法的在实时保证方面的优越性。为了验证双层调度策略的整体优势,我们还进行了吞吐率测试。结果如图 7 和表 2。

表 2 总产出率变化百分比(与简单负载平衡算法比较)

后端机数	2	3	4	5	6	7	8	9	10
集中式	−0.34%	−5.57%	−5.42%	−6.22%	0.93%	−6.63%	−8.32%	−8.28%	−4.80%
分布式	0.43%	0.20%	2.02%	1.19%	0.98%	3.23%	0.32%	−0.01%	0.67%

由这组实验我们可以清晰地看出,集中式实时调度的吞吐率明显小于简单负载平衡调度,双层调度机制又显示出明显优势,总体平均比简单负载平衡调度提高了 1.003%的吞吐率。

由此可以得出:双层调度策略不但有效地缩短了系统响应时间,而且在系统资源利用率方面大大优于现有的集中式实时调度策略。这也说明了这种调度策略对于实时集群系统的设计具有重要的参考价值。

参 考 文 献

[1] Buyya R. High Performance Cluster Computing: Architectures and Systems, Volume I. New Jersey: Prentice Hall PTR, 1999

[2] Aron M, Druschel P, Zwaenepoel W. Cluster Reserves: A Mechanism for Resource Management in Cluster-based Network Servers // Proc. of 2000 ACM SIGMETRICS Intl. Conf. on Measurement and Modeling of ComputerSystems. Santa Clara, CA, June 2000

[3] Pradham P, Chiueh T. Implementation and Evaluation of a QoS-capable Cluster-based Router // Proc. of Super Computing 2000. Dallas, Nov. 2000

[4] Bianchini R, Carrera E V. Analytical and experimental evaluation of cluster-based network servers // World Wide Web Journal, 2000, 3(4)

[5] www. isi. edu/research. html # Sensor_Networks

[6] Martinez K, Hart J K, Ong R. Environmental Sensor Networks. IEEE Computer, 2004, 37(8): 50-56

[7] http://os. rdxx. com/Linux/2003-10/4/175020874_2. shtml

[8] Zhang W. Linux Virtual Server for Scalable Network Services // Ottawa Linux Symposium 2000. http://www. linuxvirtualserver. org/Documents. html

[9] Tang R, Simha R. A Delay Differentiation Approach to Real-time Scheduling on Cluster-based Multimedia Servers // ICT 2002. Beijing, June 2002

[10] Nimmagadda S, Harari I. Scalability Issues in Cluster Computing Operating Systems // Proc. of the First Workshop on Cluster-based Computing. Rhodes, Greece, 1999