一种支持任务合并的交换网络实时调度策略

刘君瑞[1] 陈颖图[2] 樊晓桠[1]
(西北工业大学计算机学院 西安 710072)[1] (中航一集团第六三一研究所一室 西安 710068)[2]

摘 要 通过分析现有网络通信和实时系统的调度算法,在实时调度算法LSF(Least Start First)的基础上,提出支持任务合并的交换式网络实时调度策略TC-LSF(Tasks Combining-Least Start First)来保证任务在网络通信中的实时性。该算法使用任务合并策略对多个通信任务进行合并,从而节省相同网络寻径增加的网络开销,使网络的通信效率得到极大提高。给出了算法的实施细节和C语言程序片段,并对算法的性能进行了分析。

关键词 实时调度策略,LSF,任务合并,元任务,超任务
中图法分类号 TP393 **文献标识码** A

Real-time Scheduling Algorithm TC-LSF Used for the Switch Network

LIU Jun-rui[1] CHEN Ying-tu[2] FAN Xiao-ya[1]
(College of Computer Science, Northwestern Polytechnical University, Xi'an 710072, China)[1]
(The 1st Department, Aeronautical Computing Technique Research Institute, Xi'an 710068, China)[2]

Abstract By analyzing the existing scheduling algorithm in the network and the real-time systems, the author put up a real-time scheduling algorithm TC-LSF(Tasks Combining-Least Start First) used for the switch network based on the LSF algorithm, to ensure the real-time restrictions of those tasks in the real-time network. The algorithm used the tasks-combining strategy to combine multiple communications tasks, omitted the same routing and reduced the routing overhead. So, the network communication efficiency was greatly improved. This paper gave the implementation details and C fragments of the algorithm, and the performance of the algorithm was also analyzed.

Keywords Real-time scheduling algorithm, LSF, Tasks combining, Primary task, Super task

1 引言

实时调度算法是实时系统研究的一个关键技术,它保证在系统现有资源条件下各实时任务的处理操作满足其时间确定性要求。实时调度策略有静态调度和动态调度两大类,较有代表性的算法有EDF和LSF等。而许多实时任务在网络通信中也需满足实时性要求,因此实时通信调度算法就成为实时通信网络的一个核心问题。本文在实时系统调度算法LSF(Least Start First)的基础上提出一种适用于实时通信交换网络的调度策略TC-LSF(Tasks Combining-Least Start First),以保证任务在网络通信中的实时性。此算法通过合并某些实时通信任务,使多个通信任务共享一个通信任务的网络寻址过程,从而节省多个被合并的通信任务的网络寻径开销,使网络的通信效率大大提高。

2 常见的网络调度算法

网络调度算法的性能指标主要有延时、公平性、算法复杂度、吞吐量、各个业务流隔离度以及带宽利用率等,分为两类:分类优先级调度和基于帧的调度。分类优先级调度算法维持一个虚拟时间的全局变量,任务按照虚拟变量控制的时间戳递增顺序被调度。代表性的算法有:WFQ,WF2Q,VC等,这类算法具有好的公平性和较低的时延,且各类业务互不影响,但由于需要计算虚拟时间及时间戳,使得算法复杂度高。基于帧的调度算法主要是轮询式,调度器以轮询方式服务所有非空队列,算法复杂度低,但延时和公平性较差。根据不同的分配带宽原则有WRR,DRR和ERR等轮询算法。

最近几年的研究主要集中在通过负载均衡、输入输出队列调度、服务量预测以及加权平均等方法来提高调度算法的性能,但是大部分算法都是通过某种策略合理安排通信任务的执行顺序来提高网络的通信效率。本文在实时系统调度算法LSF(Least Start First)的基础上,提出使用通信任务合并的方法减少任务通信过程相同路径的重复寻径开销,源地址和目的地址相同的通信任务共享一次网络寻径成果,从而节省了大量的网络寻径时间,使网络的通信效率大大提高。作者把这种算法命名为TC-LSF(Tasks Combining-Least Start First)。

3 支持任务合并的实时调度策略TC-LSF

3.1 任务合并思想介绍

首先,作者建立实时通信任务的模型。假设实时网络系统中通信任务的调度因子包括通信源端口S_ID,通信的目的端口D_ID,任务的最迟开始时间Start_time(使用绝对时间来

到稿日期:2010-02-26 返修日期:2010-05-05 本文受浙江湖州市2008年科技创新项目"万亿次高性能计算集群的研究与设计"资助。
刘君瑞(1978—),女,博士生,讲师,主要研究方向为高性能网络、集群计算,E-mail:liu.junrui@nwpu.edu.cn;陈颖图(1978—),男,高级工程师,主要研究方向为智能信息处理技术、集群计算;樊晓桠(1962—),男,教授,博士生导师,主要研究方向为微处理器系统结构、集成电路设计。

描述，即任务最迟开始时刻距离当前时刻的时间差），任务的通信时间 Comm_time（任务的通信时间只包含任务本身数据通信的时间，不包含网络寻径的时间），那么可以使用四元组 $T(S\_ID, D\_ID, Start\_time, Comm\_time)$ 来描述实时通信系统的所有通信任务。由系统用户发出的初级实时通信任务作者称之为元任务 PT（Primary Task）。对于按照任务合并规则对多个任务合并后组成的一个新的通信任务称之为超任务 ST（Super-Task），同样对于超任务 ST 依然可以用这个四元组来描述：$ST(S\_ID, D\_ID, Start\_time, Comm\_time)$。在通信任务的输入调度队列中，元任务与超任务一起参与系统调度。

3.2 任务合并的先决条件

通信任务合并的目标就是：在不破坏整个通信系统中所有任务的通信实时性要求下尽可能合并一些能够合并的任务，通过降低通信任务的网络寻径开销来提高系统的通信效率。因此实时通信系统中任务的合并条件可以归纳如下：

（1）进行合并的任务具有相同的源端口和目的端口；

（2）任务合并后形成的超任务执行不破坏源端口向当前目的端口的所有通信任务的实时性要求；

（3）任务合并后形成的超任务执行不破坏源端口向其他目的端口的所有通信任务的实时性要求；

（4）任务合并后形成的超任务执行不破坏其他源端口向当前目的端口的所有通信任务的实时性要求。

下面举例说明任务合并的先决条件。

假设有一个 n 端口的交换机，对于每一个端口作者建立两个通信任务队列分别为：以当前端口为源端口的输入任务队列 InTasks 和以当前端口为目的端口的输出任务队列 OutTasks，并且 InTasks 和 OutTasks 中的任务已经按照 LSF 算法进行调度排序。再假设完成一次网络寻径的平均时间开销是 route_time。对于 InTasks 队列中的任意两个任务 $Tm(S\_ID, D\_ID, Start\_time, Comm\_time)$ 和 $Tn(S\_ID, D\_ID, Start\_time, Comm\_time)$ 进行合并时需要满足：

（1）$Tm.S\_ID = Tn.S\_ID$ and $Tm.D\_ID = Tn.D\_ID$；

（2）$Tm.Start\_time + Tm.Comm\_time + route\_time <= Tn.Start\_time$；

（3）$2 * route\_time <= \text{Min}\{Tk.Start\_time \mid Tk.S\_ID = Tm.S\_ID, 1<k<N\}$；

（4）$Tl.Start\_time + Tl.comm\_time + route\_time <= \text{Min}\{Tm.start\_time, Tn.Start\_time\}$

and

$\text{Min}\{Tm.start\_time, Tn.Start\_time\} + Tm.comm\_time + Tn.comm\_time + route\_time <= T(l+1).Start\_time$

式中 $Tl, T(l+1)$ 为当前目的端口的输出队列 OutTasks 里相邻的两个实时任务。

3.3 任务合并的实施方法

两个或多个任务满足合并要求就可以执行合并，组成一个超任务，任务合并成为一个超任务时需要更改 4 个元素中的通信时间，超任务的通信时间是所有子任务的通信时间和。如任务 $Tm(S\_ID, D\_ID, Start\_time, Comm\_time)$ 和 $Tn(S\_ID, D\_ID, Start\_time, Comm\_time)$ 合并后产生一个超任务 ST，可以描述如下 $ST(S\_ID, D\_ID, \min\{Tm.Start\_time, Tn.start\_time\}, Tm.Comm\_time + Tn.Comm.\_time)$，合并后的超任务如果和其他元任务再次满足合并要求，可以再次进行合并形成一个更大的超任务。

3.4 算法实现

下面作者使用 C 语言给出算法的实现过程。首先，定义一个结构体来描述通信任务。

```
Struct Task{
  Int S_ID;//通信任务的源端口信息
  Int D_ID;//通信任务的目的端口信息
  Int Start_time;//通信任务的最迟开始时间
  Int Comm_time;//通信任务的预计通信时间
  Struct Task * next;//指向下一个通信任务的指针
}
```

对于 n 端口的交换机通过创建一个二维数组来描述交换机中所有端口的通信任务输入和输出队列。

Struct Task Taskqueue[n][2];

其中元素 Taskqueue[k][0]是交换机端口 k 的通信任务输入队列，元素 Taskqueue[k][1]是交换机端口 k 的通信任务输出队列，假设网络寻径的平均时间为 route_time，下面用 C 语言描述交换机端口 k 输入队列中两个通信任务合并过程的程序片段。

```
Struct Task * In_T1, * In_T2, * Out_T1, * P;
Struct Task * in_head, * out_head;
In_head= &Taskqueue[k][0];//取交换机端口 K 的输入通信任务队列的头
out_head= &Taskqueue[k][1];//取交换机端口 K 的输出通信任务队列的头
/ * 从交换机端口 K 的输入任务队列中取出两个任务 IN_T1,IN_T2,判断能否进行合并 * /
IN_T1= In_head→next;
do{
  IN_T2= IN_T1→next;
/ * 判断两个任务的源端口和目的端口是否相同 * /
If(IN_T1. S_ID= IN_T2. S_ID) and(IN_T1. D_ID= IN_T2. D_ID){
/ * 判断任务合并时会不会影响被合并任务的实时性 * /
  If((IN_T1. Start_time+IN_T1. Comm_time+route_time)<=IN_T2. Start_time) or((IN_T2. Start_time+IN_T2. Comm_time+route_time)<=IN_T1. Start_time) {
/ * 判断任务合并后会不会影响同一个源端口发往其他目的端口的通信任务的实时性 * /
P=IN_T2;
While(route_time * 2<=P. Start_time) {
  P=P→next;
  If P→next==NULL beak;
}
If P→next==NULL{
  ST_time=IN_T1. Start_time+IN_T1. comm_time+IN_T2. Comm_time;
  Out_T1=out_head. next;
/ * 判断任务合并后会不会影响其他源端口发往同一个目的端口的通信任务的实时性 * /
FOR(i=1;I<=n;i++){
  If (Out_T1. Start_time+Out_T1. Comm_time+route_time<= IN_T1. start_time) and(ST_time+route_time <=Out_T1→next. Start_time) {
  / * 超任务的通信时间为被合并任务的通信时间之和 * /
```

```
IN_T1. Comm_time =IN_T1. Comm_time+IN_T2. Comm_time
/ * 超任务的最迟开始时间取两个通信任务中最迟开始时间的较小值 * /
If (IN_T1. Start_time>IN_T2. Start_time) N_T1. Start_time=IN_T2. Start_time;
/ * 删除被合并的任务 In_T2 * /
IN_T1→next=IN_T2→next;
Free(IN_T2);
} Else {
Out_T1=Out_T1→next;
If Out_T1==NULL break;
}
}
}
}
/ * 继续获取输入队列中下一组通信任务进行任务合并 * /
In_T1= In_T1->next;
} while(In_T1<>NULL)
```

3.5 实时调度策略 TC-LSF 在交换机级联网络中的应用

当多个交换机级联使用时，可以把任务的通信过程分成多个阶段，每个阶段完成一段寻址，那么任务合并策略就能够在每个通信阶段使用。只要在当前通信阶段通信任务的源地址和目的地址相同，就能够参考任务合并策略进行有条件的任务合并工作，这样能够减少任务的通信过程中某一个通信阶段的寻址与建立通信链路的网络开销，也能够在很大程度上提高网络的通信效率。

4 算法性能分析与比较

对于任务合并策略实时通信调度算法 TC-LSF 的性能，作者分别在单个交换机的网络环境中和交换机级联的环境中进行分析。

4.1 单交换机网络环境

假设交换机在对应端口建立通路的时间是 ECR_time，撤销通路的时间是 RCR_time，如果通信任务队列中有两个任务符合任务合并条件并进行了合并，那么网络就省去一次建立通路与撤销通路的时间，即省去了 ECR_time+RCR_time，如果有 n 个任务进行了合并，形成一个超任务，那么就省去 $n-1$ 次通路建立与撤销的时间，即省去了 $(n-1)*$(ECR_time+RCR_time)。

4.2 交换机级联网络环境

假设交换机在对应端口建立一级通路的时间是 ECR_time，撤销一级通路的时间是 RCR_time，如果两级交换机级联，通信任务的源端口和目的端口相同并进行了任务合并，那么省去的工作就是两级交换机分别建立通路和撤销通路，即省去了 $2*$(ECR_time+RCR_time)，如果这种情况下有 n 个任务满足合并条件并进行了合并，那么系统中以超任务的形式进行通信，将节省两级交换机的 $(n-1)$ 次通路建立与通路撤销的工作，即节省 $2(n-1)*$(ECR_time+RCR_time)。

如果有 m 级交换机级联，通信任务的源端口和目的端口相同，则进行任务合并，那么就省去了 m 级交换机分别建立通路和撤销通路的工作，即省去了 $m*$(ECR_time+RCR_time)，如果这种情况下有 n 个任务满足合并条件并进行了合并，那么系统中以超任务的形式进行通信，将节省两级交换机的 $(n-1)$ 次通路建立与通路撤销的工作，即节省 $m(n-1)*$(ECR_time+RCR_time)。

如果任务的源端口和目的端口不一样，只是在通信中间阶段某一个或几个步骤寻径工作一样，那么就节省了相应寻径的通信开销，比如其中有 k 个步骤寻径一样，作者对任务的通信过程进行分解之后，在这些通信阶段按照任务合并策略把任务进行合并，就能够省去 k 级建立通路和撤销通路的工作，即节省 $k*$(ECR_time+RCR_time)，如果我们合并了 n 个任务，就能够省去 n 次 k 级的通路建立与通路撤销的工作，即节省 $k(n-1)*$(ECR_time+RCR_time)。

在整个通信系统中，能够使用 TC-LSF 算法进行合并的通信任务或通信子寻径过程越多，通信效率提高的效果就越明显，节省的通信开销就越可观。

结束语 支持任务合并的实时通信调度策略 TC-LSF 提出使用任务合并的方法来降低网络寻径的开销。对于由同一个源端口向同一个目的端口发出的实时通信任务，如果能够把多个任务合并成一个超级任务，并且还能够不影响其他所有任务的通信实时性限制，那么多个通信任务的网络寻径过程就可以只做一次，从而节省大量的重复寻径带来的网络开销，网络的通信效率将到极大提高。

参 考 文 献

[1] Wu Jun, Luo Junzhou. Randomized scheduling algorithm for input-queued switches[J]. Journal of Southeast University, 2005, 3: 6-10

[2] Dong Yu-guo, Wang Sheng-rong, Guo Yun-fei, et al. Exploring Load Balancing of a Parallel Switch with input Queues[J]. Journal of Software, 2007, 2: 229-235

[3] 李峭，张晓林，熊华钢. 基于样本路径分析的交换式以太网实时通信调度方法[J]. 航空学报，2005，9：575-580

[4] 周卫华，朱新宇，武穆清，等. 一种公平输入排队调度算法[J]. 电子与信息学报，2005，3：341-345

[5] 綦朝晖，孙济州，李小图. 基于队列的模糊拥塞控制算法[J]. 软件学报，2005，16(2)：286-294

[6] 王玥，蔡皖东，段琪. 一种自适应动态负载均衡算法[J]. 计算机工程与应用，2006(21)：121-123

[7] 李勇，罗军舟，吴俊. 一种交叉点小缓存 CICQ 交换机高性能调度算法[J]. 计算机研究与发展，2006，43(12)：2033-2040

[8] 王景存，谢馨艾，王沁，等. 基于输入队列的调度算法及其稳定性证明[J]. 计算机工程，2007，11：130-134

[9] 李季，曾化鑫，郭子荣. 基于时槽预定的加权公平调度策略[J]. 软件学报，2007，10：2605-2612

[10] 徐刚，丁泉龙. 基于预测的最长队列优先调度算法[J]. 计算机工程，2008，1：7-9

[11] 尹德斌，谢剑英. 一种新的加权公平队列调度算法[J]. 计算机工程，2008，2：28-31

[12] 张福阳，熊庆旭. 一种简单的 VOQ 交换机时延确保分组调度算法[J]. 北京航空航天大学学报，2008，11：1323-1326

[13] 曾媛，龚文斌，刘会杰，等. 适合星载交换机的调度算法[J]. 计算机工程，2009，2：158-161

[14] 曾媛，龚文斌，刘会杰，等. 低轨卫星交换机的建模与仿真[J]. 系统仿真学报，2009，3：1518-1561