基于动态数据结构—池的打印任务调度

Dispatching of Printing Tasks Based on a Novel Dynamic Data Structure-Pool

张徐亮[1] 陈细木[2] 李诗雨[1]

(电子科技大学信息中心 成都610054)[1] (西南师范大学计算机科学系 重庆400715)[2]

Abstract In this paper, we analyze the dispatch problem of printing tasks, discuss the properties of a novel dynamic data structure--pool, and accordingly present a new method for task dispatch of printers based on pool data structure. The experimental results are satisfying.

Keywords Dynamic data structure, Pool, Dispatch, Cycle queue

一般而言,打印任务调度或管理多采用队列,实现先来先服务的打印服务。随着计算机网络不断的深入应用,共享打印机、网络打印机不断涌现出来。这些打印机势必面对繁多的打印任务,这些打印任务中有的需要及时打印出来,这样就要打破先来先服务的队列原则。因此,本文引入动态数据结构—池来实现打印队列任务的调度。在以下的叙述中,所有算法均采用SPARKS语言来进行描述。

1 基于循环队列的打印机任务管理

常用的打印机打印任务都采用的是循环队列(图1),实现先来先服务。设打印队列为Print_Queue,打印任务 $Task_1$ 当前正在打印,在打印队列Print_Queue中,等待打印的任务有 $Task_2$, $Task_3$, …, $Task_n$。当打印机完成当前任务 $Task_1$ 时,打印机发送空闲信息给操作系统,操作系统检查Print_Queue队列是否有等待打印的任务。如有则执行队列的删除操作Print_Queue_DeleteTask,将 $Task_2$ 送打印机。

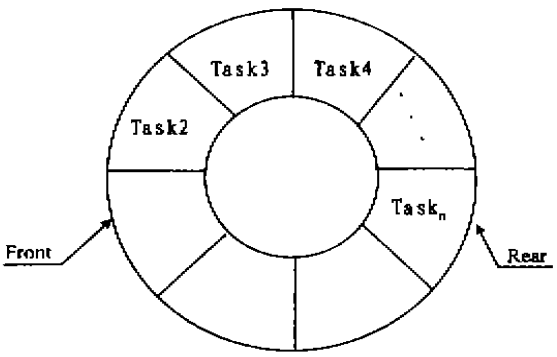

图1 循环打印队列

```
Procedure Print_Queue_Delete Task(Print_Queue, MAX,
    front, rear, task)
    //删除队头打印任务,并保存到变量 task 中//
    if front=rear then call QUEUE_EMPTY
    else
    [ front←(front+1)mod MAX
      task←Print_Queue(front)
    ]
End
```

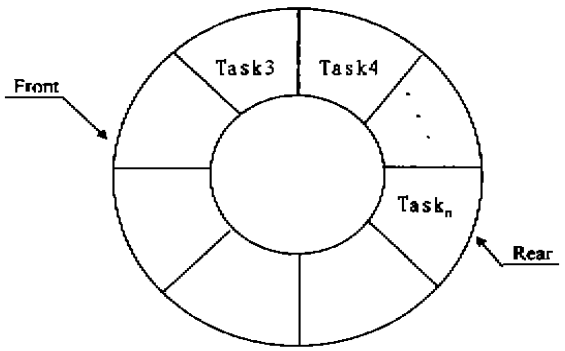

图2 删除 $Task_2$ 之后的循环队列

此时,打印队列中任务如下:

当有新的打印任务时,执行Print_Queue_AddTask操作,添加任务到打印队列Print_Queue中。

```
Procedure Print_Queue_AddTask (Print_Queue, MAX,
    rear, NewTask)
    //添加新任务 NewTask 到循环队列 Print_Queue 中//
    rear←(rear+1)mod MAX
    if rear=front then call QUEUE_FULL
    else
        Print_Queue(rear)←NewTask
End
```

2 基于一维池的打印机任务管理

打印任务一般都是先来先服务,如果打印任务有优先级的要求,就需要调整打印任务的顺序,此时只有通过打印管理程序修改队列中等待打印的任务的顺序来实现,使用起来不是很方便。基于这样的原因,作者

引入一维池对打印任务进行管理，打印任务池中的任务都带有优先级。

2.1 一维池的基本概念

定义1 池是n≥0个数据元素 $a_1,a_2,\cdots,a_n$ 的有序集合，池中元素以元素的某一属性为标准保持着有序状态。因此，元素在池中的位置会不断地发生变化。这种位置随时间不断变化的结构，称为动态数据结构。

具有 n 个数据元素的一维池是一个数据结构，其定义是：

Pool=(D,R)

其中：$D=\{a_i \mid a_i \in \text{data object}, 1\leqslant i\leqslant n, n\geqslant 0\}$；$R=\{N\}$；$N=\{r(a_{i-1},a_i) \mid a_{i-1},a_i\in D, 2\leqslant i\leqslant n\}$。

n 是一维池的深度，也就是一维池中元素的个数，当 n=0时，称该一维池为空池。

R 是元素之间的关系的集合，r 表示相邻元素之间的关系，可以用函数表示。r 多由用户来定义，典型的函数有：$Wa_i-Wa_{i-1}\geqslant\varepsilon$ 或者 $Wa_{i-1}-Wa_i\geqslant\varepsilon$，其中 $\varepsilon\geqslant 0$，W 则表示对应数据元素的重量。

由于池始终保持有序的特性，所以一维池实际上是一个有序的存储结构。如果和堆栈相比较的话，一维池可以说是一个有序的栈。为了保持池的有序状态，需有一个秩序维持程序。这个秩序维持程序也可以说是池的属性。秩序维持程序可以有两种机制，一种是周期机制，也就是每隔一定时间，池就进行一次秩序的调整；另一种是触发机制，此时，秩序维持程序在下列两个条件满足时执行：

1)当有新的元素加入时。此时要给这个新元素安排适当的位置；

2)元素的重量可以动态地改变。当元素的重量动态变化时，此时要建立新的秩序。

这两种情况都可以触发秩序维持程序，使池中元素能够保持秩序。

2.2 用一维池对打印任务进行管理

设打印任务池为 Print_Pool，Task1正在打印，打印任务池中还有很多其它等待打印的任务，如 Task2，Task3，…，Taskn 等。

当打印机完成当前打印任务时，操作系统检查打印池中是否有等待打印的作业，如有，则执行池的删除操作 Print_Pool_Delete Task，将池顶的任务送到打印机。

```
Procedure Print_Pool_DeleteTask(Print_Pool,task)
    if top≠0 then
    [
        task←Print_Pool(top)
        top←top-1
    ]
    else
        call POOL_EMPTY
End
```

当有新的打印任务时，执行 Print_Pool_AddTask 操作将新的打印任务添加到打印任务池中。

```
Procedure Print_Pool_AddTask(Print_Pool,NewTask)
    if top=m-1 then
    [
    call POOL_FULL
    return
    ]
    top←top+1
    call Print_Pool_Order(Print_Pool)
End
```

打印任务池中的任务的重量可以动态地变化，可以按最长等待时间、最短作业、平均等待时间等原则来对任务的优先级进行调整，也可以由应用程序或用户来指定。Print_Pool_Order(Print_Pool)将使一维任务池中的任务保持有序。

3 基于二维池的打印服务器

如果打印服务器可以连接多台打印机，就要对打印作业和打印机同时进行管理。本文用二维池来实现打印任务的管理。

3.1 二维池的概念

定义2 二维池是 m×n≥0个数据元素 $a_1,a_2,\cdots,a_{m\times n}$ 的有序集合，池中元素以元素的某一属性为标准保持着有序状态。

具有 m×n 个数据元素的二维池是一个数据结构：

Pool2=(D,R)

其中：$D=\{a_{ij} \mid a_{ij}\in \text{data object}, 1\leqslant i\leqslant m, 1\leqslant j\leqslant n, n\geqslant 0\}$；$R=\{N\}$；$N=\{r(a_{i-1,j-1},a_{i,j-1}(a_{i-1,j}),a_{i-1,j-1},a_{i-1,j}),a_{i-1,j})\in D, 2\leqslant i\leqslant m, 2\leqslant j\leqslant n\}$。

n 是二维池的深度，m 为二维池的宽度或跨度，m×n 也就是二维池中元素的个数。当 m×n=0时，称该二维池为空池。

R 是元素之间的关系的集合，r 表示相邻元素之间的关系，可以用函数表示。r 多由用户来定义，典型的函数有：

$Wa_{i-1,j}-Wa_{i-1,j-1}\geqslant\varepsilon$，且

$Wa_{i,j-1}-Wa_{i-1,j-1}\geqslant\varepsilon$

或者

$Wa_{i-1,j-1}-Wa_{i-1,j}\geqslant\varepsilon$，且

$Wa_{i-1,j-1}-Wa_{i,j-1}\geqslant\varepsilon$

其中 $2\leqslant i\leqslant m, 2\leqslant j\leqslant n, m,n\geqslant 0, \varepsilon\geqslant 0$。

3.2 二维打印服务器池

设打印服务器连接了 m 台打印机，则二维池为 Print_Pool2[1:m][1:n]。相当于是由 m 个一维池组成的，每个一维池对应某一台打印机。池中的任务由于其重量的关系会发生位置的变化，同时为了保证各打印机之间任务的均衡，也会对打印作业的位置进行调

整。任务的重量主要由打印作业的优先级组成，而任务的均衡则主要取决于作业的长度和当前各打印机剩余作业长度。池中元素重量从池顶到池底呈降序排列，作业的优先级由应用程序或系统程序进行设置；每一维中作业的长度以及当前打印机的剩余长度之和要尽量均衡。设 L_i 为第 i 台打印机对应池中作业长度以及打印机当前剩余作业长度之和，且设 $L_{i+1} \leqslant L_i$，其中 $1 \leqslant i \leqslant m-1$，则当 $\sum_{i=1}^{m-1}(L_{i+1}-L_i)$ 最小时让各打印机任务均衡。

设某台打印服务器共连接了三台打印机，构造二维池 print_pool2(1:3)(1:5)来对打印任务进行管理。设三台打印机分别为 p1、p2、p3，当前未完成的作业长度如表1所示。设池中现在有五个打印作业，t1，t2，t3，t4，t5，长度和优先级如表1所示。

表1　打印机当前未完作业的长度

	p1	p2	p3
未完成作业长度	7k	5k	0k

表2　作业的长度及优先级

	t1	t2	t3	t4	t5
作业长度	10k	12k	8k	30k	100k
作业优先级	9	8	10	6	2

此时，池中作业可能的分布如图3所示。

t1	t3	
t2	t4	t5

图3　池中作业可能的分布

首先考察一下二维打印池的秩序的维持。由于要考虑打印机任务的均衡，所以二维打印池的秩序维持程序与文[1]中二维池的秩序维持程序不同。当二维打印池中的任务按优先排好序之后，则进行任务的均衡调整。鉴于二维打印池的特殊性，本文认为处于打印池中同一层的优先级属于同一级别，可以进行位置的调整。如图3，t1和 t3可以交换，t2和 t4也可以交换等。打印池任务均衡就是通过层间任务的位置交换来实现的。为简单起见，可以让相邻两层间的任务按长度相互逆序排列，这样可以减小时间复杂度。

从图3可以得到 L1＝29k，L2＝33k，L3＝100k，从而 $\sum_{i=1}^{m-1}(L_{i+1}-L_i)=71k$。

二维打印池的秩序维持程序如下：

```
Procedure Print_Pool2_order(Print_pool2)
  if modified=false then return
  for temp_top←1 to number(Print_pool2)
    call DeletePool2(Print_Pool2,item)
    temp_pool(temp_top)←item
  end
  call PoolCycleOrder(temp_pool)
  k←1
  while k≤temp_top do
    Print_pool2((k mod (m+1)),⌊k/m+1⌋)←temp_pool
                (k)
    k←k+1
  end
  modified←false
  top←⌈temp_top/m⌉
  for k←1 to top do
    d←m
    while d>1 do
      d←⌊d/2⌋
      repeat
          flag←0
          for i←1 to n-d do
            j←i+d
        if k mod 2=1 then
              if print_pool2(i,k)>print_pool2(1,k)then
              [
                  temp←print_pool2(i,k)
                  print_pool2(i,k)←print_pool2(j,k)
                  print_pool2(j,k)←temp
                  flag←1
              ]
        end
      until flag=0
    end
  end
end
```

其中任务的均衡用的是 SHELL 排序。

以下我们考虑一下从池中取任务及向池中添加任务的算法。

当其中第 i 台打印机完成当前任务时，操作系统将检查与它相对应的一维池中有无任务，有则将任务分配给它。

```
Procedure Print_Pool2_Delete Task(Print_Pool2,i,task)
    if top[i]≠ 0 then
    [
       task[i]←Point_Pool2[i][top[i]]
       top[i]←top[i]-1
    ]
    else
    [
       call POOL_EMPTY(i)
       call Print_Pool2_Order(Print_Pool2)
    ]
End
```

其中 top 为一维数组，即 top(1:n)，用以指示二维池中每个一维池的池顶；

task 也是一维数组，即 task(1:n)，用以指示第 i 台打印机当前的打印任务；当有新任务加入时，执行 Print_Pool2_AddTask 将新任务添加到二维池中。

```
Procedure Print_Pool2_AddTask(Print_Pool2,NewTask)
    pool_top←0
    for i←1 to m do
       pool_top←pool_top+top[i]
    end
    if pool_top>=n*m then
    [
       call POOL_FULL
       return
```

```
    ]
    j←⌊pool-top/m⌋+1
    i←pool-top mode m+1
    Pool2(i)(j)←New Task
    If top(i)<n then
        Top(i)←top(i)+1
    modified←true
End
```

事实上循环队列和一维池也都可以实现多台打印机打印的调度，但由于打印机为一台以上，所以此时要用一个队列来对打印机的任务申请进行管理，也就是要对打印机进行排队管理；另外也不能使打印机任务达到均衡，这会使得一些打印机不停地工作，而一些打印机则经常闭置，显然这对打印机资源有效利用不利。

4 基于循环队列和基于池的打印任务调度的比较

队列简单易行，但不支持优先级；新的打印任务只能加入到队尾。如果想调整打印任务的顺序，要手动进行修改。池比队列要复杂一些，但它的最大特性是支持优先级，新的打印任务自动加入到适当的位置。

由于循环队列没有循环语句，所以队列的时间复杂度为0；而池的结构的维护由其秩序维持程序进行，其中主要是快速排序，而快速排序的时间复杂度为O(nlog2n)。

两种结构的空间复杂度是一样的。

参考文献

1 Ford W，Topp W 著. 数据结构 C++语言描述. 第3版. 刘卫东、沈官林译. 清华大学出版社，1998. 11

2 Stallings W. Operation Systems Internals and Design Principles. 3rd ed. 清华大学出版社，1998. 6

3 Shaffer D A 著. 数据结构与算法分析. 张铭，刘晓丹译. 电子工业出版社

（上接第123页）

序描述的低层次抽象到高层次抽象进行，其许多的描述恢复可复用性技术是半自动的。如果我们把自动程序的识别的研究结果按真实程序的比例放大，那么将来的工具就有可能控制一定规模的再工程过程。

面向对象的程序更新研究与可复用的再工程是相关的。在文[3]中介绍了一种在C语言描述的程序中识别对象的方法，候选类的选择是基于类型定义的分析；其次，那些带有一个给定类型参数或一个给定类型的值的过程、可被看成是候选的方法。在文[6]中，则描述了过程化的程序通过渐增的再工程方式建立一个面向对象的体系结构。

结束语 许多研究者和实践者都认为软件的可复用技术将提高软件的研究效率，并从根本上改变软件的升级，从而促进软件产业的变革。通过软件复用，在应用系统开发中可以充分地利用已有的开发成果，消除许多重复劳动，从而提高软件生产率和软件质量。软件复用中的一些问题是与现有系统密切相关的，而软件的再工程正是解决这些问题的主要技术手段。

可复用性的再工程技术有三方面的益处：第一，原有系统的优点将被保存下来；第二，设计和代码通过系统给定的应用域能达到一致，易于将来的维护；第三，可复用框架使得再工程和其它新系统的开发更容易。

未来的工作将更集中于评估这种技术在实用的再工程项目中的运用，以及高层次的再工程各阶段的自动化处理。这些技术问题的解决，将涉及到用CASE工具联接领域建模和设计抽象，以及程序中概念识别的自动化。本文中描述的这个具有可复用特性的代表性框架，也可引入形式化规格说明方式加以改进，当然这需要基于其它许多先进的研究领域，包括领域建模、可复用技术及软件规格说明中的代码自动生成技术等的进展。

参考文献

1 Jarzabek S. Software reengineering for reusability. IEEE Software，1993(3)：100～106

2 Arnold S. Software Reengineering. IEEE Computer Society Press，1993

3 *Chikofsky E，et al.* Reverse Engineering and Design Recovery：A Taxonomy. IEEE Software，1990(1)：13～18

4 Devanbu P B，et al. LaSSIE：A Knowledge-Based Software Information System. CACM，1991，34(5)：34～49

5 Rich C，Wills L. Recognizing Program's Design：A Graph-Parsing Approach IEEE Software，1991(1)：82～89

6 Jackobson S，et al. Re-engineering of old system to an object-oriented architecture. In：Proc. OOPSLA'91，1991. 340～350

7 郭耀，等 再工程——概念及框架. 计算机科学，1999，5(26)：78～83

8 蔡希尧，陈平. 面向对象技术. 西安电子科技大学出版社，1993