

页面置换竞争性实验2021

页面置换竞争性实验2021

- 一、实验介绍
- 二、评测样例
 - 1、样例说明
 - 2、基准代码
- 三、实验要求
- 四、提交步骤
 - 1、提交代码
 - 2、查看当前排名的方法
- 五、测试集更新和征集说明

一、实验介绍

操作系统理论课介绍了页面置换的相关知识。我们来回顾一下：在地址映射过程中，若在页面中发现所要访问的页面不在内存中，则产生缺页中断。当发生缺页中断时，如果操作系统内存中没有空闲页面，则操作系统必须在内存选择一个页面将其移出内存，以便为即将调入的页面让出空间。而用来选择淘汰哪一页的规则叫做页面置换算法。

我们在lab4中也实现了处理缺页中断的过程。但是，lab4中的页面置换算法是由硬件系统实现的，并不需要我们干预。下面我们将模拟一个简单的页面置换场景，对同学们的算法进行竞速排名。

假设操作系统允许我们使用的物理页框数为4个，访问的虚拟页面依次为，Page1, Page7, Page8, Page9, Page10, Page8，那么当访问到Page10的时候就必须将当前一个物理页框中的内容换出，然后将Page10换入。所以，先产生缺页中断，然后进行页面置换。在本页面置换实验模拟中，我们将考虑上述两部分的代价：**总代价 = 中断的代价 + 单页面置换代价 * 置换页面数。**

二、评测样例

1、样例说明

以物理页框为4的情况为例。

测评数据：

```
1 4095
2 4095
3 8192
4 16384
5 32768
6 4097
7 4095
8 32768
9 16384
10 8192
```

初始物理页框 `physic_memery={-1,-1,-1,-1}` (-1仅用于样例说明, 在实际实验中为方便起见, 初始页框初始化为 0, 并且**不会调用需要0号页面的虚拟地址**)

1. 输入第一个数据(由评测机完成)4095, 评测程序调用 `pageReplace(physic_memery, 4095)` 函数, 物理页框修改为 `physic_memery={0,-1,-1,-1}`, 本次代价=中断代价+置换代价=1+2=3;
2. 输入第二个数据4095, 调用 `pageReplace(physic_memery, 4095)` 函数, 物理页框不变, 本次代价=0+0;
3. 输入地址8192, `physic_memery={0,2,-1,-1}`, `cost+=(1+2*1)`
4. 输入地址16384, `physic_memery={0,2,4,-1}`, `cost+=3`
5. 输入地址32768, `physic_memery={0,2,4,8}`, `cost+=3`
6. 输入地址4097, `physic_memery={1,2,4,8}`, `cost+=3`
7. 输入地址4095, `physic_memery={1,0,4,8}`, `cost+=3`
8. 输入地址32768, `physic_memery={1,0,4,8}`, `cost+=0`
9. 输入地址16384, `physic_memery={1,0,4,8}`, `cost+=0`

Cost = 18

假设在上述例子中可以实现某种预调度

1. 输入4095, `physic_memery={0,2,4,8}`, `cost+=(1+2*4)`
2. 输入4095, `physic_memery={0,2,4,8}`, `cost+=0`
3. 输入地址8192, `physic_memery={0,2,4,8}`, `cost+=0`
4. 输入地址16384, `physic_memery={0,2,4,8}`, `cost+=0`
5. 输入地址32768, `physic_memery={0,2,4,8}`, `cost+=0`
6. 输入地址4097, `physic_memery={1,2,4,8}`, `cost+=3`
7. 输入地址4095, `physic_memery={1,0,4,8}`, `cost+=3`
8. 输入地址32768, `physic_memery={1,0,4,8}`, `cost+=0`
9. 输入地址16384, `physic_memery={1,0,4,8}`, `cost+=0`

Cost = 15

2、基准代码

```

1 // pageReplace.cpp
2 #include "pageReplace.h" // 测评需求，请务必包含该头文件
3 #define MAX_PHY_PAGE 8 // 这里只使用了8个物理页框
4 #define MAX_PAGE 12
5 #define get_Page(x) (x>>MAX_PAGE)
6 void pageReplace(long * physic_memery, long nwAdd) {
7     int flag = 0;
8     static int clock = 0;
9     for (int i = 0; i < MAX_PHY_PAGE; i++) {
10         if ((nwAdd >> MAX_PAGE) == physic_memery[i]) {
11             return;
12         }
13     }
14     for (int i = 0; i < MAX_PHY_PAGE; i++) {
15         if (physic_memery[i] == 0) { //初始物理页框内容全部为0
16             physic_memery[i] = get_Page(nwAdd);
17             flag = 1;
18             break;
19         }
20     }
21     if (flag == 0)
22         physic_memery[(clock++) % MAX_PHY_PAGE] =
get_Page(nwAdd);
23 }

```

三、实验要求

- 物理页框数量为64（页框编号 0~63），要求初始页表为空，页面大小为4KB。
- 实现页面置换算法，函数命名为 `void pageReplace (long* physic_memery, long nwAdd)`（不需要实现 main 函数），该函数需要接收物理页框指针 `physic_memery`（初始物理页框为空）。其中物理页框 `physic_memery` 为一个数组，`physic_memery[p] = v` 表示当前物理页框 `physic_memery` 的第 `p` 个页框对应虚拟页 `v`。`nwAdd` 表示当前程序访问的虚拟地址（按字节编址），执行页面置换之后，该虚拟地址所在的虚拟页必须在物理页框中有对应。在实际实验中，`nwAdd < INT_MAX`。
- 要求学生实现的函数被调用时接收传入的虚拟地址，并且立即执行分配，修改 `physic_memery` 数组，评测会检测当前使用的虚拟地址是否在物理页框中有对应。
- 内存空间限制128M。`pageReplace` 函数总执行时间限制为300s（共有约166万条使用地址，所以会调用约166万次该函数）。测试地址由真实程序使用地址和模拟生成的程序使用地址组成。

- 评分会综合程序运行时间、程序使用内存、总置换代价给出。程序运行时间的权重为0.2，程序使用内存的权重为0.1，页面置换代价的权重为0.7。中断的代价为1，单页面置换代价为2。打分公式如下：

$$Score = \frac{t - t_{max}}{t_{min} - t_{max}} * 0.2 + \frac{m - m_{max}}{m_{min} - m_{max}} * 0.1 + \frac{c - c_{max}}{c_{min} - c_{max}} * 0.7 \quad (1)$$

其中 t 、 m 和 c 分别表示你的运行时间、使用内存和置换代价； t_{max} 和 t_{min} 分别表示当前所有学生提交中最长的和最短的运行时间， m_{max} 和 m_{min} 分别表示当前所有学生提交中最大和最小的使用内存， c_{max} 和 c_{min} 分别表示当前所有学生提交中最大的和最小的置换代价。

值得说明的是，为了评测的公平起见， t_{max} 、 m_{max} 与 c_{max} 在评分时将进行以下修正：

$$\begin{aligned} t_{max} &= \min(t_{max}, 3 * t_{standard}) \\ m_{max} &= \min(m_{max}, 12.6MB) \\ c_{max} &= \min(c_{max}, c_{standard}) \end{aligned} \quad (2)$$

其中 $t_{standard}$ 表示上述基准代码的运行时间， $c_{standard}$ 表示上述基准代码的置换代价。

- 提交次数不作限制，返回的评测结果仅供参考，且服务器仅**保留最近一次测评结果**。
- 竞速期间每天的02:00-05:00时间段会关闭评测，服务器将拉取同学最近一次提交逐个进行评测，并在08:00将结果公布，此结果同样作为参考。**需要注意的是，在此处参与评测的提交务必不要含有getScore文件（详见提交步骤-查看当前排名的方法）。**
- 参与竞争实验的同学需要**完成一份竞争实验的相关实验报告**，发送至助教邮箱18373494@buaa.edu.cn，实验报告中要求至少包括：**算法设计思想、算法实现技巧、以及竞争实验过程中的优化与改进、本地测试情况**等内容。推荐用markdown书写。
- 页面置换竞争性实验属于一项挑战性任务，可作为申优的条件之一。**原则上，以页面置换竞争性实验申优的同学，在竞争排名中不得低于前30%。**
- 页面置换竞争性实验的**开始评测时间为2021年5月7日20:00，截止时间为2021年5月10日20:00。**

四、提交步骤

1、提交代码

1. 创建并切换到 racing-2021 分支，可以用 `git checkout -b racing-2021`
2. 清空当前目录，`git rm -rf .`

3. 在当前目录下创建 `pageReplace.cpp` 文件。此文件必须 `#include "pageReplace.h"`，注意**双引号不要改成尖括号**！然后在 `pageReplace.cpp` 中实现 `void pageReplace (long* physic_memery, long nwAdd)` 函数。评测机只识别该文件，其他任何文件对评测不起作用。
4. `git add --all, git commit -m "blabla", git push origin racing-2021:racing-2021`。
5. 提交评测后首先会返回编译信息。**评测时间较长，约1-2分钟，请大家耐心等待。**
6. 编译环境：`g++ x86_64_linux_gnu, std=c++11`。

评测记录示例：（测评分数100在本次测评中是无意义的）

```
remote: *****
remote:
remote:          BUAA OSLAB AUTOTEST SYSTEM
remote:          Copyright (c) BUAA 2015-2019
remote: *****
remote:
remote: [ You are changing the branch: refs/heads/racing-2021. ]
remote:
remote: Autotest: Begin at Mon Apr 26 16:04:15 CST 2021
remote:
remote: warning: remote HEAD refers to nonexistent ref, unable to checkout.
remote:
remote: Switched to a new branch 'racing-2021'
remote: Branch racing-2021 set up to track remote branch racing-2021 from origin.
remote: [ found your pageReplace.cpp ]
remote: [ compile successfully ]
remote: [ your results are as follows ]
remote: your id is 19060000
remote: your cost is 486960
remote: your time is 0.807359
remote: your mem is 12.382812
remote: your score is 1.000000
remote: your rank is 1 of 1
remote: [ You got 100 (of 100) this time. Mon Apr 26 16:05:32 CST 2021 ]
```

2、查看当前排名的方法

1. 在 `pageReplace.cpp` 同路径下创建 `getScore` 文件(文件内容可有可无)。
2. `git add --all, git commit -m "blabla", git push origin racing-2021:racing-2021`，提交后若检测到 `getScore` 文件，则不触发编译，只返回当前你的运行时间、代价和排名。
3. 正常提交评测时，请将 `getScore` 文件删除再提交。

显示排名示例：

```
remote: *****
remote:
remote:          BUAA OSLAB AUTOTEST SYSTEM
remote:          Copyright (c) BUAA 2015-2019
remote: *****
remote:
remote: [ You are changing the branch: refs/heads/racing-2021. ]
remote:
remote: Autotest: Begin at Mon Apr 26 16:06:45 CST 2021
remote:
remote: warning: remote HEAD refers to nonexistent ref, unable to checkout.
remote:
remote: Switched to a new branch 'racing-2021'
remote: Branch racing-2021 set up to track remote branch racing-2021 from origin.
remote: [ found your getScore ]
remote: [ your nearest results are as follows ]
remote: your id is 19060000
remote: your cost is 486960
remote: your time is 0.807359
remote: your mem is 12.382812
remote: your score is 1.000000
remote: your rank is 1 of 1
remote: [ You got 100 (of 100) this time. Mon Apr 26 16:06:46 CST 2021 ]
```

五、测试集更新和征集说明

为防止出现过拟合的现象，鼓励同学们自己创造新的测试集并且进行测试。使用一个程序来体现内存的实际使用情况，并将内存调用情况输出到文件中（也就是编写一个数据生成器）。例如这个代码[matrix_mult.cpp](#)就是一个典型的例子，它输出矩阵乘法中内存调用的过程。

本实验鼓励同学们“分享”自己的测试用例，将数据生成器**源程序（请务必保证编译通过，运行正确）**发送到助教邮箱18373665@buaa.edu.cn，如果该样例程序比较有代表性，则可能增加到现有的测评数据集中。被选中评测数据的同学可酌情加分。

数据生成器输出的要求：

- 输出到文件 `case-学号.txt` 中，不要输出到标准输出
- 源程序用 `C++` 编写，命名为 `generate-学号.cpp`
- 输出文件的格式为：一行一个虚拟地址，最后一行为-1，且虚拟地址必须在 `[4096, 2147483647]` 之间，如下：

1	6355260
2	6355256
3	4184
4	2147483647
5	1926817
6	19260817
7	6487608
8	6443496
9	19260817
10	6355256
11	6355260
12	19260817
13	-1

- 同时附上一个README，说明该数据生成器生成的数据的特点（**单纯的随机数生成器将被拒绝**）

同时为了公平起见，防止部分同学“投机取巧”，我们也将竞速排名结束之后，对模拟生成的随机数据更换随机种子重新生成，并以**更换之后的数据对所有同学的代码进行多次重测**，取运行时间和代价的平均值参与最终排名。