CSE 30341: Home Work Project 3

Assigned: Feb 26

Due: Mon, Mar 19, 10:40 AM

Late submissions will not be accepted

Group effort

Memory simulator:

In this project, we will write a memory simulator in order to understand the behavior of page replacement algorithms for actual programs. First we collect actual memory execution traces. We will then use them in our simulator for the following page replacement policies: FIFO, LRU and second-chance. For each of the policies, you will measure the maximum number of page frames that your program will require. Now, rerun the simulator to use ¼, ½, ¾, 1 times this number of frames and plot the number of page faults, number of disk writes and number of disk reads (somewhat similar to Figure 9.13). Next we describe each of these steps in detail

1. **Step 1: Collect memory traces for the simulator:**

   First we will collect the memory traces from an actual program. We will use the X86-64 version of PIN tool (http://rogue.colorado.edu/Pin/index.html). I've installed this version in /home/cse30341-sp07/OS/hwp3/pin/Bin/. This program will work on expsys-desktop1 through expsys-desktop6 (but not on the Itanium servers). The specific command to collect all memory traces of a program is

   /home/cse30341-sp07/OS/hwp03/pin/Bin/pin -t
   /home/cse30341-sp07/OS/hwp03/pin/SimpleExamples/pinatrace -values 0 -- <program>

   For example, you can collect the memory access trace for *ls* using

   /home/cse30341-sp07/OS/hwp03/pin/Bin/pin -t
   /home/cse30341-sp07/OS/hwp03/pin/SimpleExamples/pinatrace -values 0 -- /bin/ls

   The output is generated in a file called pinatrace.out. The file is of the format

   <program counter> R/W <dest data address> <size> <src data address>

   For this project, we will only look at the Read/Write bit and the dest data address. I've collected two such traces, called trace1.out and trace2.out; available in /home/cse30341-

sp07/OS/hwp03/traces/. You will run your simulator for these two tracefiles. You will also generate a third trace file by running pin through your own simulator. Note that if the traces from your own simulator turns out to be too big, choose any simpler program (perhaps a simpler version of hwp #2 that uses a smaller matrix).

2. **Step 2: Calculate the maximum number of data frames required:**

Next, calculate the maximum number of page frames required for executing each trace. Since we use a page frames of size 1024, a simple shell script to calculate this figure is:

```
awk '{addr=strtonum($3)/1024; printf "%8d\n", addr}' trace1.out | sort -u | wc -l
```

3. **Step 2: Write the memory simulator:**

Next, you will write the actual memory simulator. Your simulator can assume that the system will only run your program. You do not have to evict a page before your program first uses it. Your simulator will take the tracefile and the number of memory frames as inputs. The number of frames required by the trace was calculated in step 2. You will run your simulator to use ¼, ½, ¾ and 1 times this limit. Assume page frames are 1024 bytes large. Repeat your experiments for FIFO, LRU and second-chance page replacement policies and plot the results.

4. **Step 3: Write a report:**

The most important component of this project is a report which clearly explains your implementation detail and an analysis of the resulting graphs. It is not acceptable to just present your graphs without any explanation.

You may turn in your program and any raw data that you deem to be of interest in the AFS drop box. These files may be used to verify your graphs. However, for the most part, we will only grade your project based on the written report. Turn in a hard copy of the report. Feel free to discuss the output of your simulator (especially for the traces provided) with your colleagues.