Project Report
Evaluating Page Replacement Algorithms

Tamilmani Manoharan (ID: 110385545)

1 Background Work

Finding good page replacement algorithms according to dynamic access patterns is critical. The kernel should have flexible option to dynamically switch between page replacement algorithms without reloading kernel. I have started off this project by reading about different page replacement algorithms given in references section. Among the algorithms I read, LRUK was chosen after deciding up with professors as it would be the first and simple step to start off with. Inorder to implement LRUK in base 4.4.1 kernel one should have good knowledge about existing implementation of page repalcement algorithms. So I have spent few days in gaining insight about the code flow of default LRU algorithm.

2 Implementation

Once I got better understanging of page replacement algorithm implemenation, I tried to implement LRUK in existing code. The main idea is that wherever reference bit is set , I have to add code which should calculate page heat value based on difference between 2nd previous reference time and the current time. The page heat value will be calculated as follows

diff=2nd Last RefTime−currentTime
pageHeat = 2000000/diff

2000000 is a factor that is multiplied to avoid floating point values.

In default algorithm , the page eviction is done based on reference bit. For LRUK, I have implemented in such a way that the pages should be evicted only if page heat is less than threshold value. This threshold parameter is configurable and it can modified using sysctl variable "kernel.lruk". So the pages which are above the threshold level will be put back to active list incase of LRUK algorithm.

The switch between algorithms can be done using sysctl varaible "kernel.sample".

Most of the modifications are done under mm directory. The sysctl part has been added in fs/nfs. PageHeat , Page Reference Time fields are added in struct page. I have defined page heat calculate function in mm_inline.h

The major problem I faced during testing was stablity. Whenver I run LRUK algorithm ,it will run without issues for one or two times but after that I got "NMI watchdog: BUG: soft lockup - CPU0 stuck for 22s!" and the system hangs. Later I debugged this issue by commenting out and then found that reference bit should also be set whenever I'm calculating page heat. The page eviction is done based on page heat value and reference bit will not be taken into account for LRUK algorithm. After this change, I found the system to be stable.

3 Experimental Results

3.1 Test Setup

The first step is that I have to install 4.4.1 in physical machine and boot with 4.4.1. We(Myslef and Ravi) have faced many difficulties during this stage. Booting with 4.4.1 got stuck while loading from initramdisk. Later we debugged this issue along with the help from Tao. We build 4.4.1 kernel with oldconfig(3.13) and then we are able to boot with 4.4.1 in physical machine. Inorder to put pressure on page cache and memory, I have limited RAM size to 2GB by specifying in grub file.

3.2 Test Code

For this, I have written different scenarios and finally finalized upon one scenario. I have created a big array of 1.5GB in size and a small array of 512MB in size. The smaller array is accessed more than the bigger array. Then using rusage linux API, I tried to find the page faults incurred by both LRU and LRUK algorithms. Upon hearing advice from Professor Mike, I wrote a python test script which could simulate different trials for LRU and LRUK algorithm and collect page faults for each trial.

3.3 Results

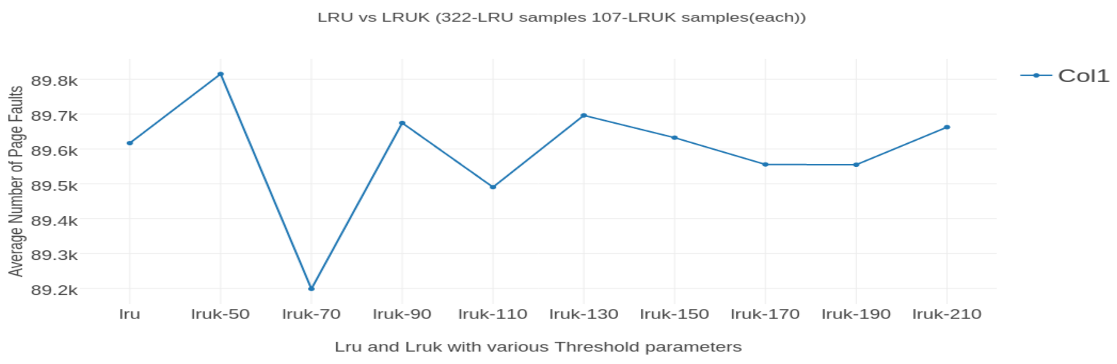

Figure 1: LRU vs LRUK Results

I have collected 322 LRU samples and 107 samples of LRUK for each threshold value. Then I took average of page faults for each category. The above figure shows that LRUK-70 performs better than other cases. So depending upon scenario, one algorithm may perform better than other. In this case, LRUK-70 performs better because the test program is developed in favour of LRUK. Also we could see that performance of LRUK depends on threshold parameter. The tabular format of results are given below:

LRU	89616
LRUK-50	89815
LRUK-70	89198
LRUK-90	89674
LRUK-110	89490
LRUK-130	89696
LRUK-150	89632
LRUK-170	89555
LRUK-190	89554
LRUK-210	89662

3.4 Changes

I have submitted by kernel changes, test code ,test script in github.
The github link is https://github.com/tamilmani1989/advproject

4 References

1. https://www.researchgate.net/publication/47862865_Towards_Self-Tuning_Memory_Management_for_Data_Servers

2. http://www.csd.uoc.gr/~hy460/pdf/p297-o_neil.pdf

3. http://www.openu.ac.il/home/wiseman/2os/lru/lru-k.pdf