Assignment 2: Measuring the Size and Cost of Accessing a TLB Report

Student: Chengbin Hu

This assignment require us to set a timer to measure the costing for different size page table. Because at first the OS will take a look at TLB for the fastest accessing, by increase the size of page numbers we access, we could estimate the TLB size and (if have) the second cache size.

So the basic process I would use the tlb.c to accept 2 parameters, one for the page numbers we want to measure, the other is the loops we want to repeat those page numbers.

At the beginning, I will set the program to run at one single CPU.

Then first use malloc to form a larget array contains pagenumber*pagesize/sizeof(int). This is an array use to loop for page number entries. For example if page size is 4kb, then one jump is 1024. if we only measure one page table entry, the a[1024] could just fill a page. if 2 page entry, we will generate a length [2048]. and so on. Then at last use the loops as the assignment suggested to perform page table accessing.

By timing those process we could calculate the average accessing time for a single accessing of page table entry = total/ (loops* page numbers).

Question1.

I use gettimeofday() as my timer. From the man gettimeofday page, the accuracy is micro seconds. I think make sure the run time at one or several seconds is long enough to get accuracy accessing time of TLB.(round 1ns) so I make the total accessing times(page number*trail) be 10^9) which could see it in run.c files.

There is another timer clock_gettime() take struct timespec* This is more accuracy to ns. we can read more from man clock_gettime

Question2.

The tlb.c has the code to accept two arguments number of pages to touch and the number of trails. output is average accessing time / perpage accessing

Question3.

By run the ./run I could get the results. From my estimation total time about one second is accuracy enough for micro second timer. And it is not a good idea for run 10s second for each measurement.(too boring). First I set accessing times(page number*trail) be 10^9. so 1,000,000,000 trails for 1 pagenum. 500,000,000 trails for 2 and so on until 8.

And I found there is a jump at 8-16. so from 16 I reduce the total page numbers accessing to 10^8. That is 6,250,000 for 16 3,125,000 for 16 and so on until 2048.

I run the Program on c4lab20 first. and repeat several times and get similar result:

```
<1>, <1.958270ns>

<2>, <1.296857ns>

<4>, <1.285698ns>

<8>, <1.331906ns>

<16>, <7.318920ns>

<32>, <7.815410ns>

<64>, <8.183170ns>

<128>, <8.571680ns>

<256>, <8.556610ns>

<512>, <8.544791ns>

<1024>, <8.588252ns>

<2048>, <14.263437ns>
```

then run it on c4lab19 repeat several times seems lab19 has larger 2nd cache because there is no 2nd jump on average time:

```
<1>, <1.781157ns>

<2>, <1.142369ns>

<4>, <1.155072ns>
```

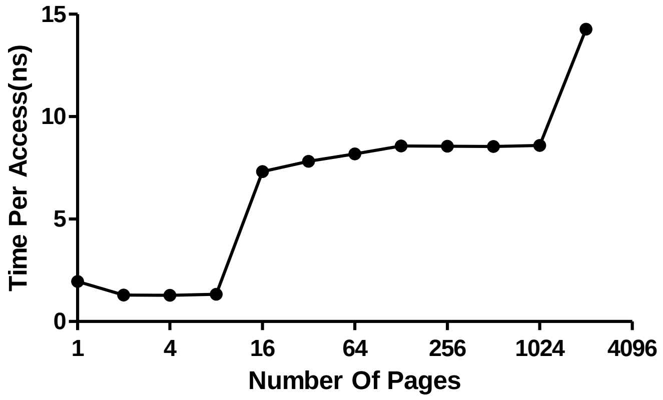
```
<8>, <1.194813ns>

<16>, <6.926810ns>

<32>, <7.115680ns>

<64>, <7.791880ns>

<128>, <7.843570ns>

<256>, <7.750590ns>

<512>, <7.838540ns>

<1024>, <7.708030ns>

<2048>, <8.218051ns>
```

then run it on c4lab18 repeat several times there is no 2nd jump on average time too:

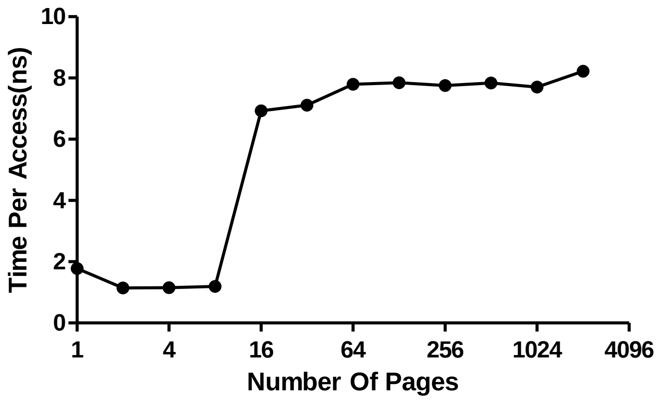
```
<1>, <1.975989ns>

<2>, <1.279945ns>

<4>, <1.285944ns>

<8>, <1.332021ns>

<16>, <7.672760ns>

<32>, <8.073230ns>

<64>, <8.806360ns>

<128>, <8.920570ns>

<256>, <8.664090ns>

<512>, <8.835597ns>

<1024>, <8.634242ns>

<2048>, <9.082563ns>
```

Question 4. graph.

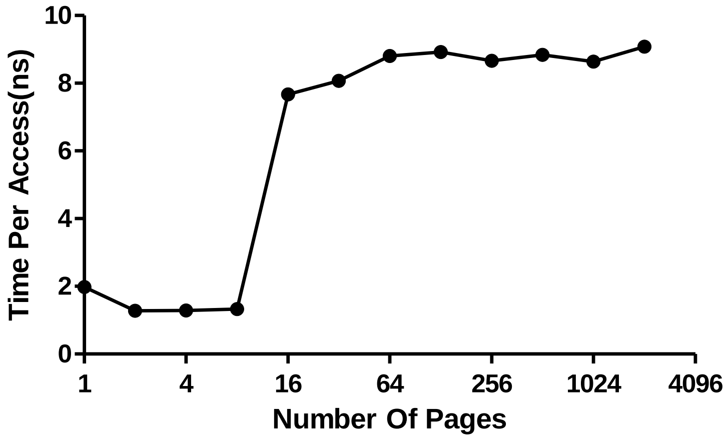
TLB Size Measurement(c4lab20)



TLB Size Measurement(c4lab19)



TLB Size Measurement(c4lab18)



Question 5.

To prevent compiler does not remove the main loop. I add increment+=1 after the end of the timer. This will notice the compiler that I will use the increment again after the loop so the loop will be run to the end.

Question 6.

If we do not set the CPU affinity to one CPU, the different size of TLB size of different CPU will affect the accessing time. typicall will make the single access time faster because there will be more page entry on the TLB but this efficiency is hard to predict because it should consider the CPU scheduling policy.

Finally, for the accuracy of the test, it is important to remember that the in order to do an accessing of pagenum entry. the CPU has also do some work, like increase the increment, change array pointer. Those step also take time but hard to measure separately. The other problem is the CPU use to do context switch many times, the context switch should be considered.