Paging: Faster Translations (TLBs)

Annalise Tarhan

February 12, 2021

1 For timing, you'll need to use a timer like gettimeofday(). How precise is such a timer? How long does an operation have to take in order for you to time it precisely?

gettimeofday() uses the system's clock time and is measured in microseconds. Any operation that is faster than a microsecond might not register at all, and those that are only a few microseconds long won't be measured very precisely. The solution is to run the operation many, many times, and then divide by the number of trials.

2 Write a program tlb.c that can roughly measure the cost of accessing each page. Inputs to the program should be: the number of pages to touch and the number of trials.

See tlb.c

3 Now write a script to run this program, while varying the number of pages accessed from 1 up to a few thousand, perhaps incrementing by a factor of two per iteration. How many trials are needed to get reliable measurements?

See tlb_driver.sh. The script is written in two parts because the program segfaults if the number of pages is 2048 or greater. Since the number of pages was

incremented by a factor of two each iteration, that gave me no data for numbers of pages greater than 1024. Hence the second part of the script.

I started getting consistent results with 64 pages and when the number of trials was around 32 or higher. The typical result was 11 nanoseconds per page access. That number increased significantly when the number of pages exceeded 1024, typically settling on 19 nanoseconds per page access.

4 Next, graph the results, making a graph that looks similar to the one above. Visualization usually makes the data easier to digest; why do you think that is?

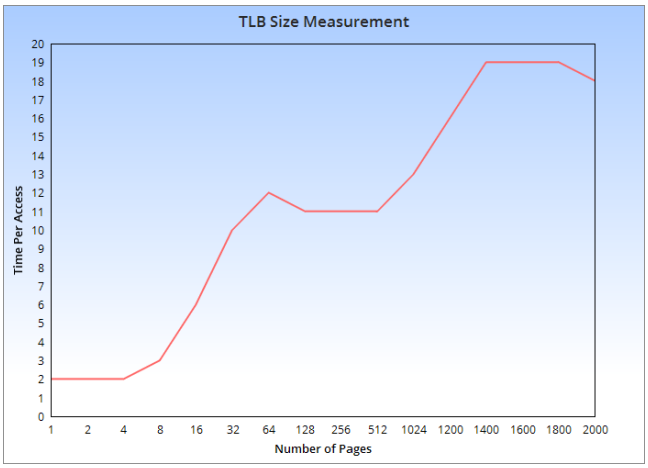

Figure 1:

The point of data is to indicate relationships and patterns. Raw numbers alone force us to try to imagine those patterns, while graphics make them obvious.

5 One thing to watch out for is compiler optimization. Compilers do all sorts of clever things, including removing loops which increment values that no other part of the program subsequently uses. How can you ensure the compiler does not remove the main loop above from your TLB size estimator?

Change the optimization setting to none using the -O0 flag.

6 Another thing to watch out for is the fact that most systems today ship with multiple CPUs, and each CPU, of course, has its own TLB hierarchy. To really get good measurements, you have to run your code on just one CPU, instead of letting the scheduler bounce it from one CPU to the next. How can you do that? What will happen if you don't do this, and the code moves from one CPU to another?

On Linux, use the pthread_setaffinity_np function. If the code isn't pinned to a CPU, the context switch would increase the measured time.

7 Another issue that might arise relates to initialization. If you don't initialize the array before accessing it, the first time you access it will be very expensive, due to initial access costs such as demand zeroing. Will this affect your code and its timing? What can you do to counterbalance these potential costs?

Yes, it will make the first iteration, with the small number of pages and small number of trials seem much slower than it should. To avoid that problem, initialize each element of the array before starting the timer.