Homework week 6

Chapter 18

Q1. How should page-table size change as the address space grows?

As address space grows, the page-table size grows proportionally.

Q1.1 As the page size grows? Why not use big pages in general?

Big page table would cause internal fragments, not every chunk of memory needs that large space.

Q2. What happens as you increase the percentage of pages that are allocated in each address space?

With more virtual address space used, more pages become valid and activated.

Q3. Which of these parameter combinations are unrealistic? Why?

All three work properly in the given condition, however, the first two I would say unrealistic.

- P 8 -a 32 -p 1024 -v -s 1: both the page size and the virtual address space are too small.

- P 8k -a 32k -p 1m -v -s 2: only 4 pages allowed in this case, perhaps not realistic.

- P 1m -a 256m -p 512m -v -s 3: the page size might be suitable for those machines with 16G RAM or 8G RAM.

Q4. Can you find the limits of where the program doesn’t work anymore? For example, what happens if

the address space size is bigger than physical memory?

For example, let the page size be greater than the virtual address space, it doesn’t work anymore, or let the page size be greater than the physical address space.

Chapter 19

Q1. How precise is such a timer? How long does an operation have to take in order for you to time it precisely? (this will help determine how many times, in a loop, you’ll have to repeat a page access in order to time it successfully)

The timer I used is clock_gettime(CLOCK_MONOTONIC,). Its precision is in the range of nanoseconds. I chose to run it 10,000 times, for the purpose of making the results stable and reliable.

Q5. How can you ensure the compiler does not remove the main loop above from your TLB size estimator?

Compile the program with the option “-o0”.

Q6. How can you do that? (hint: look up“pinning a thread” on Google for some clues) What will happen if you don’t do this, and the code moves from one CPU to the other?

If it doesn’t do this, every time it moves to another CPU, there will be a TLB miss, and it will take longer to access the memory.

Q7. Will this affect your code and its timing? What can you do to counterbalance these potential costs?

I think it will. However, I have tested both, and I don’t see the expected difference in the performance. To counterbalance these potential costs, I have initialized the variables after declaration.