18648 Lab2 Writeup

Mengwen He, Tarun Ala, Hongbao Zhang

October 16, 2016

1 What is the difference between concurrent execution and parallel execution?

Concurrency is the property of two different tasks to run independently of each other and run at approximately the same time. This could also be achieved using a time sharing mechanism.

Parallelism is the mechanism of achieving concurrency where tasks are run at the same time. This can be achieved using multiple processors.

2 Question 2

Imagine Dexter needs to instrument his kernel to collect some data points. The points are generated within the kernel at some memory address and need to be visualized in a userspace application. Suppose each point appears periodically at one memory address in the kernel, for example corresponding to a memory-mapped register of a device. If a point is not read before the next one is ready, then the point is lost.

Suppose that on Dexters platform:

- the computation overhead of reading the point from the initial memory address is negligible
- it takes $100\mu s$ to complete a round-trip between user-space and kernel space
- it takes 10ns to copy one data point value from kernel memory to user memory.

2.1 As a trained syscall writer, Dexter creates a syscall to retrieve the one data value and call the syscall in a tight while loop from a userspace application. Suppose the data points are generated at a rate of 1,000 points per second. What fraction of points, if any, are lost due to the overhead delay?

A point is created every 1ms. The round trip takes 100us=0.1ms. As this time is much less than the total time required to create the point and as it takes only 10ns to copy the point from kernel to userspace. Hence all the points are read given the 1000 per seconds sampling rate.

2.2 Dexter is not satisfied with a slow sampling rate, so from now on, suppose the data points are generated at a rate of 100,000 points per second. Assuming the same implementation approach, what fraction of points, if any, are lost due to the overhead delay under the faster sampling rate?

Now, the 100000 points per second sampling rate creates a point every 10us. The round trip takes 100us. Assuming that the copying from the kernel to userspace memory occurs during the round trip, the time is covered in the overhead of 0.1ms. As the execution takes 100us, the loop would pick up only 10 out of every 100points. Hence the loss of points is 9/10.

2.3 Dexter abandons the naive approach and changes his implementation to amortize the delay of kernel-user crossing over many points by buffering the points in kernel memory. For a buffer size of 1000 points, what fraction of points, if any, are lost due to the overhead delay? (Assume Dexter reads the points once the buffer is full)

Assuming the sampling rate of 100000 points per second would create a point every 10us. Hence for the buffer to get filled, it would take 1000*10us=10ms. The read time for the buffer would be 1000*10ns=10us. Hence 1 point would be lost due to the buffer being full and being read during this time. Hence the total number of points loss is 1/1001.

2.4 Dexter is not satisfied with losing any points at all. How can he improve his implementation to achieve this?

A solution for the given problem would be implementing hardware interrupts where the points could be stored in the buffer size of 1000 and read after the buffer gets half full.