CSC469
ASSIGNMENT 1

DANIEL BLOEMENDAL
#997678936

Contents

Date: October 9, 2013.

1. Tracking process activity

1.1. **Hypothesis.** The goal of the experiment was to investigate the activity of a single ready process running on a Linux 3.2 system. In particular we were interested in discovering how long a process is active on the system before it is disrupted by a timer interrupt. Furthermore, we wanted to know how long it takes for the operating system to service the timer interrupt before returning control to the single ready process.

Our expectation was that the timer interrupt would be very inexpensive to service. Since no context switch is occurring, the operating system can simply reschedule the ready process and continue its operation. Therefore we expected the process would receive almost all of the CPU time with short consistent interruptions determined by the frequency of the timer interrupt.

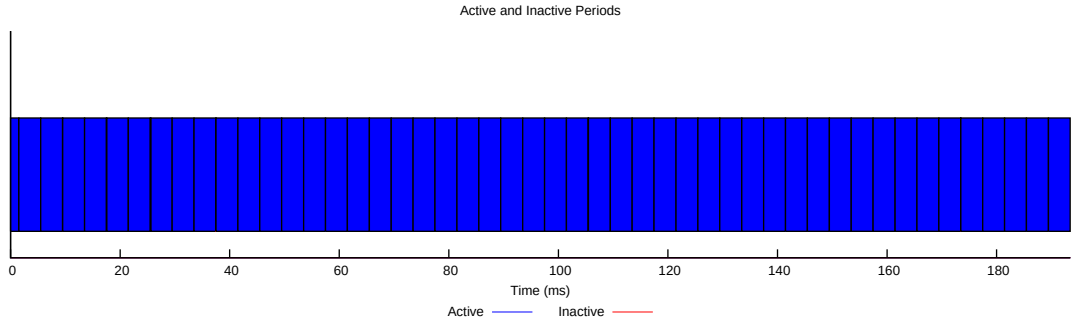
1.2. **Hardware.** The experiment was run on a CDF lab computer with the specifications listed in figure 1.

Figure 1. b2220-05 hardware

Host	b2220-05.cdf.toronto.edu
CPU	Intel® Core™ CPU G630 2.70GHz clock 3MB cache 2 cores
Memory	8GB physical 1GB swap
Kernel	Linux 3.2.0 x86_64

1.3. **Data.** The plot in figure 4 is based on 50 gathered samples of active & inactive periods. We considered a process to be inactive if at least 2500 CPU cycles occurred outside of the process. In other words the CPU cycle threshold for the experiment was 2500.

Figure 2. Timer interrupts



The following is a table with the first 15 samples of the experiment

FIGURE 3. Samples

Sample	Active (ms)	Inactive (ms)
1	1.502190	0.011074
2	3.987884	0.003625
3	3.995221	0.019725
4	3.979117	0.004095
5	3.994876	0.011533
6	0.058799	0.007539
7	3.920979	0.002301
8	3.999042	0.138075
9	3.858219	0.005810
10	3.995646	0.003678
11	3.995211	0.001495
12	3.997288	0.005308
13	3.993599	0.002890
14	3.996001	0.001489
15	3.997299	0.003389

1.4. **Observations.** From the data we can safely assume that the time slice our single process is receiving before the timer interrupt fires is 4 milliseconds long. As for the time it takes to service a timer interrupt, it seems to vary. Perhaps the operating system is performing maintenance operations during the interrupt. However, if we exclude some of the outliers like sample 8 in **figure 3** we come up with approximately 2.4 microseconds for a timer interrupt using all 50 samples.

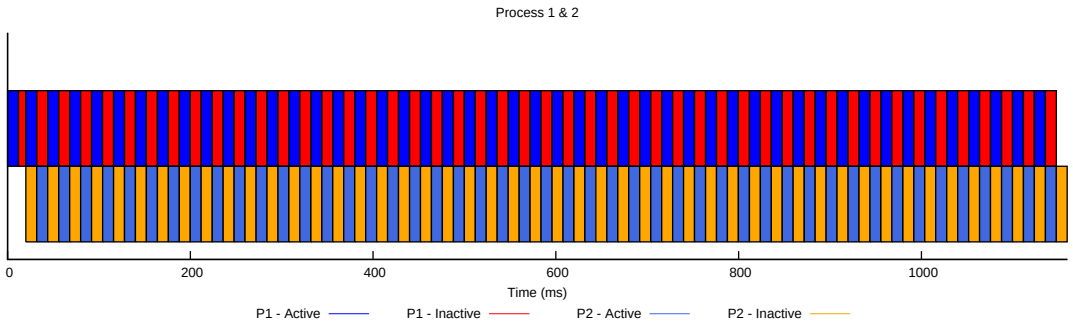
It should be noted that it seems like one other non-timer interrupt occurred during data collection. Referring to sample 6 in **figure 3**, we see that our process was interrupted shortly after the active period began. Since the data was collected over an SSH session, this may be an interrupt related to network I/O. However, it could certainly be some other device.

1.5. **Conclusions.** It appears that our hypothesis is correct. The active process received most of the CPU time during the experiment and timer interrupts were not too expensive, taking only 2.4 microseconds to service on average.

2. Context switching

In order to measure context switching time, two instances of the instrumentation tool were spawned. After obtaining the active & inactive periods from both running processes we come up with the following interleaved time slices.

Figure 4. Context switching



Context switch time is then computed by looking for the difference in cycles between the end of an active period in **process 1** and the beginning of an active period directly after in **process 2**. The following is a table with the first 15 results.

Figure 5. Context switching time

Time (ms)	Duration (ms)
19.980906	0.001890
43.974582	0.001836
67.968357	0.001767
91.962007	0.001769
115.955662	0.001778
139.949312	0.001768
163.943115	0.001784
187.934263	0.003382
211.930346	0.001780
235.924042	0.001732
259.917750	0.001759
283.909039	0.001810
307.905116	0.001737
331.896405	0.001830
355.892555	0.001700

The context switch seems to last around 1.8 microseconds on average, and in some cases lasts a little more than 3 microseconds. Finally it should be noted that the processes received a time slice of 12 ms. That's three timer interrupts, 4 ms each, worth of time.

E-mail address: d.bloemendal@utoronto.ca