CG2271 Real Time Operating Systems

Tutorial 11

1. Are the tasks you create in your term assignment more similar to processes on general operating systems or more similar to threads? Explain your answer. In addition, explain why real-time tasks are not considered to be threads.

   They are more similar to threads in that only MSW, registers and PC are saved, unlike processes on GOS where file handles, memory handles, environment etc are saved.

   However they are still not threads because a thread is a "process within a process", which these tasks are not.

2. Explain, with an example, how temporal and spatial localities ensure that in a short term only small portion of memory is accessed at any time. Explain why this is relevant memory hierarchies.

   Consider a program:

   ```
   int a[128];

   <statement 1>
   <statement 2>
   …
   <statement n>
   for(i=0; i<127; i++)
     a[i]=a[i]+a[i+1];
   ```

   This program exhibits spatial locality in that i) statements 1 to n and the loop itself would form instructions in memory that are accessed consecutively and ii) the array consists of consecutive memory locations that are accessed sequentially. It also exhibits temporal locality because i) the instructions within the loop would be accessed 127 times and ii) each array element is read twice and written to once. (e.g. read a[0], read a[1], write to a[0]. Read a[1], read a[2], write to a[1], etc).

   This is important to memory hierarchy because it means that in the short term only a small portion of memory is being accessed, and if this portion can be copied to a fast memory, we can save on memory access times.

3. Suppose we had two memory technologies T1 and T2 with the following characteristics:

Technology	Cost/MB	Access Speed
T1	$2	10ns
T2	$0.08	80ns

Suppose we were building a computer system with 1GB of memory. Complete the following table:

For simplicity, 1GB=1000MB instead of 1024MB. 256KB=25% of 1MB. Average memory access time for configuration 3 = 10+(0.02*80) because in a memory hierarchy, if there is a "cache miss", i.e. the data needed is not in the T1 memory, we need to copy from the T2 memory back to the T1 memory. This happens 2% of the time incurring an average time of 0.02*80 ns. The T1 memory is then read. So T1 memory is always accessed regardless of a hit or a miss.

Configuration	Amount of T1 memory	Amount of T2 memory	Cost	Average access time.
1	-	1GB	**$80**	**80ns**
2	1GB	-	**$2000**	**10ns**
3*	256KB	1GB	**$80.50**	**11.6ns**

* Configuration 3 is arranged as a cache-main memory hierarchy, and 98% of all memory accesses would be handled by the T1 memory.

Given your answers, explain why memory hierarchies make sense.

Memory hierarchy allows us to have an access time close to the expensive T1 memory (11.6ns vs 10ns), at a price that is very close to the cheaper T2 memory ($80.50 vs $80). This represents a 1.6% drop in performance against a 96% drop in price.

(Actually a 2% miss rate is already very high. This study shows a cache miss rate of 0.03% - i.e. 99.97% of all accesses can be found in the cache.

http://www.cs.wisc.edu/multifacet/misc/spec2000cache-data/)

4. Explain what address translation in virtual memories is, and why it is necessary.

Virtual memory is a scheme combining hardware (page-translation mechanisms, page fault detection) and software (page fault resolution, page replacement algorithms, on-disk page management) to trick the CPU into thinking that there is more main memory than is physically available, by using space on the hard-disk to simulate memory space. This allows a user to squeeze in more programs and data than is otherwise possible. Locality ensures that memory access times would be close to main memory (or cache memory) access times rather than hard-disk access times. Again this gives huge, cheap memory space at fast access times.

5. In this question we will consider a virtual memory system with 64 bytes per page, 10 bit virtual addresses and 9 bit physical addresses.

   a. What is the maximum size in bytes of the virtual memory? Physical memory?

      1024 bytes of VM, 512 bytes of physical memory.

   b. What is the maximum number of virtual and physical pages?

      Byte index is 6 bits (2^6=64), leaving 4 bits for VPN or 16 virtual pages, and 3 bits for PPN or 8 physical pages.

   c. Using the page table below, translate the following virtual addresses to physical addresses:

      0, 15, 576, 987, 1020

	V	PPN
0	1	2
1	1	3
2	0	(13,5,17)
3	0	(12,5,6)
4	1	4
5	1	7
6	0	(10,3,1)
7	0	(3,4,1)
8	0	(7,5,5)
9	1	6
10	0	(6,3,13)
11	0	(7,1,16)
12	1	5
13	0	(6,4,1)
14	0	(6,8,3)
15	1	1

Virtual Address	**Binary (page number/Byte Index)**	**Virtual Page Number**
0	**0000** 000000	0
15	**0000** 001111	0
576	**1001** 000000	9
987	**1111** 011011	15
1020	**1111** 111100	15

Using the table above we get the following translations:

Virtual Address	**VPN**	**PPN**	Binary (**PPN/**Byte Index)	**Physical Address**
0	0	2	**010** 000000	128
15	0	2	**010** 001111	143
576	9	6	**110** 000000	384
987	15	1	**001** 011011	91
1020	15	1	**001** 111100	124