COSC1112/1114: Operating Systems Principles

Tutorial 03 (week 04)

1. **Provide two programming examples in which multithreading does not provide better performance than a single-threaded Solution.**

Answer:

- Any kind of sequential program is not a good candidate to be threaded. An example of this is a program that calculates an individual tax return.
- Any kind of program that solves problems that can be broken up into smaller sub-problems which can be solved independently (divide-and-conquer). An example is a "shell" program such as the C-shell or Korn shell. Such a program must closely monitor its own working space such as open files, environment variables, and current working directory.

2. **Is it possible to have concurrency but not parallelism? Explain.**

Answer:

Yes. Concurrency means that more than one process or thread is progressing at the same time. However, it does not imply that the processes are running simultaneously. The scheduling of tasks allows for concurrency, but parallelism is supported only on systems with more than one processing core.

3. **Using Amdahl's Law, calculate the speedup gain of an application that has a 60 precents parallel component for**
   (a) two processing cores
   (b) four processing cores.

Answer:

Two processing cores = 1.43 speedup; four processing cores = 1.82 speedup.

4. **Which of the following components of program state are shared across threads in a multithreaded process?**
   - **Register values**
   - **Heap memory**
   - **Global variables**
   - **Stack memory**

Answer:

The threads of a multithreaded process share heap memory and global variables. Each thread has its separate set of register values and a separate stack.

5. **Linux does not distinguish between processes and threads. Instead, Linux treats both in the same way, allowing a task to be more akin to a process or a thread depending on the set of flags passed to the clone() system call. However, many operating systems— such as Windows or Solaris—treat**

processes and threads differently. Typically, such systems use a notion wherein the data structure for a process contains pointers to the separate threads belonging to the process. Contrast these two approaches for modelling processes and threads within the kernel.

Answer:

On one hand, in systems where processes and threads are considered as similar entities, some of the operating system code could be simplified. A scheduler, for instance, can consider the different processes and threads on an equal footing without requiring special code to examine the threads associated with a process during every scheduling step. On the other hand, this uniformity could make it harder to impose process-wide resource constraints in a direct manner. Instead, some extra complexity is required to identify which threads correspond to which process and perform the relevant accounting tasks.

6. **The program shown below uses the Pthreads API. What would be the output from the program at LINE C and LINE P?**

```
#include <stdio.h>
#include <unistd . h>
#include <stdlib.h>
#include <pthread . h>

int value = 0;

void* runner ( void* param) {
        value = 5;
        pthread_exit ( 0 ) ;
}

int main ( ) {
        int pid ;
        pthread_t t id ;
        pthread_attr_t at t r ;

        pid = fork ( ) ;
        i f ( pid == 0) {
                pthread_attr_ini t (&at t r ) ;
                pthread_create(&tid ,&attr , runner ,NULL) ;
                pthread_join (tid ,NULL) ;
                print f ( "CHILD: v = %d" , value ) ; /* LINE C */
        } else if (pid >0) {
                wait (NULL) ;
                printf ( "PARENT: v = %d" , value ) ; /* LINE P */
        }
        return EXIT_SUCCESS;
}
```

Answer:

Output at LINE C is 5 as the thread executes in the same context as the child process. Output at LINE P is 0.

7. **Google's Chrome browser is designed to open each new website in a separate process. Would the same benefits have been achieved if instead Chrome had been designed to open each new website in a separate thread? Explain.**

Answer:

No. The primary reason for opening each website in a separate process is that if a web application in one website crashes, the other browser processes are unaffected. Because multiple threads all belong to the same process, any thread that crashes would affect the entire process.

8. **Consider a multiprocessor system and a multithreaded program written using the many-to-many threading model. Let the number of user-level threads in the program be more than the number of processors in the system. Discuss the performance implications of the following scenarios.**
   - **The number of kernel threads allocated to the program is less than the number of processors.**
   - **The number of kernel threads allocated to the program is equal to the number of processors.**
   - **The number of kernel threads allocated to the program is greater than the number of processors but less than the number of user-level threads.**

Answer:

When the number of kernel threads is less than the number of processors, then some of the processors would remain idle since the scheduler maps only kernel threads to processors and not user-level threads to processors. When the number of kernel threads is exactly equal to the number of processors, then it is possible that all of the processors might be utilized simultaneously. However, when a kernel-thread blocks inside the kernel (due to a page fault or while invoking system calls), the corresponding processor would remain idle. When there are more kernel threads than processors, a blocked kernel thread could be swapped out in favor of another kernel thread that is ready to execute, thereby increasing the utilization of the multiprocessor system.