Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2333 - Sistemas Operativos y Redes - 2/2018

Tarea 2: Múltiples Procesos

Nombre: Manuel Becker
Número Alumno: 13622986

1. ¿Qué ventajas y desventajas podría tener la implementación de esta tarea con `threads` en vez de el uso de procesos separados?

Implementar la tarea con threads tendría por un lado la ventaja de poder acceder a la memoria del proceso principal sin tener ningún problema con dicho acceso. En el caso de la implementación con procesos diferentes, al usar fork(), los procesos hijos copian la memoria y la hacen parte de su propio programa, pero luego de realizar los cambios no refleja su resultado en la memoria del padre. Por este motivo se decidió usar vfork() que permite dicho comportamiento. La posibilidad de realizar cambios de memoria principal con los threads permite por un lado, realizar partes del proceso principal simultáneamente, para luego reflejar dicho resultado en el proceso principal que generó el thread. El tema es que esta situación es propensa a errores, dado que se debe tener especial cuidado cuando los procesos generan threads si la memoria fue o no cambiada ya por el thread o si aún queda por ejecutarse. El acceso a la memoria compartida puede entonces generar errores, lo que puede significar una desventaja del uso de threads por sobre procesos separados.

Por otra parte, los procesos son más complejos de trabajar que los thread. El overhead necesario para crear un proceso es mucho mayor, ya que todas las variables se copian desde el padre al proceso hijo, lo que representa mayor complejidad del uso de procesos, por sobre lo que son los thread. Dado que el **thread si comparte variables** entre el proceso principal y el thread, no se requiere el mismo nivel de overhead y se tiene menor complejidad.

2. Ejecute su programa de tal manera que haya 1, 2, 4, 8 y 16 simulaciones ejecutando simultáneamente. Para cada ejecución mida su tiempo de ejecución usando `time`. Explique el comportamiento de los tiempos `real`, `user`, y `sys`.

Los resultados de la ejecución de las diferentes simulaciones se pueden observar en la tabla. Para todas las simulaciones se utilizó un tiempo de ejecución máximo de 20. Se usó para medir el tiempo de ejecución time.

Las diferencias entre User time y System time es que el primero corresponde al tiempo que el proceso pasa en CPU realizando sus tareas del código en espacio usuario, mientras que el segundo es el tiempo que pasa en CPU por estar en espacio kernel. El total representa la suma de ambos tiempos ya descritos.

En el caso de las diferentes cantidades de procesos que se están simulando, se obtuvieron resultados que no hacen mucho sentido. El tiempo de ejecución no debería mantenerse constante en el tiempo real para prácticamente todas las simulaciones. En teoría, los tiempos siempre deberían subir, ya que se están procesando un mayor número de procesos.

El comportamiento que se observa en los resultados que se muestran a continuación, representan para mí implementación de la tarea que las simulaciones pueden estar ejecutándose más rápido porque existen más procesos por los vfork() utilizados, que acaparan más el uso de la CPU.

Finalmente, a partir de los resultados obtenidos, no se observa realmente lo que se espera de la implementación de los procesos separados, ya que el tiempo real se mantiene mayoritariamente constante para mayor cantidades de procesos.

Numero de procesos	1	2	4	8	16
Real	0.028	0.027	0.022	0.022	0.022
User	0.004	0.004	0.004	0.004	0.004
System	0.004	0.004	0	0.004	0