UFPEL

Relatório de Sistemas Operacionais

Desempenho de algoritmos recursivos com Pthreads

Alejandro da Silva Pereira

Ciência da Computação

Guilherme da Silva Xavier

Engenharia de Computação

Pelotas, 27 de Maio de 2021.

UFPEL

1. Introdução

Este relatório tem como objetivo discutir o desempenho de um programa para diferentes tipos de exemplos, com diferentes números de processadores virtuais. Este programa foi construído como um suporte de execução implementando o modelo n x m de threads para uma linguagem concorrente baseada na criação de tarefas. O suporte foi implementado na forma de biblioteca, usando a linguagem de programação C++ e a biblioteca de threads Pthreads.

Nessa biblioteca, cada um dos processadores virtuais executa um loop que retira uma tarefa da lista de tarefas prontas e a executa, passando o parâmetro de entrada. Caso essa lista de tarefas prontas esteja vazia, o pv permanece em estado de espera aguardando que haja alguma tarefa para ser consumida. Ao finalizar a execução, é guardado o ponteiro do dado retornado na estrutura da tarefa e armazenada na lista de resultados. No momento da sincronização, podem acontecer três situações: a tarefa está pronta para executar, já foi executada e está na lista de tarefas terminadas, ou está executando. Nesse último caso, a primitiva aguarda até que a tarefa esteja concluída.

Os exemplos recursivos implementados foram: o cálculo de Fibonacci, fatorial, soma e potenciação.

2. Resultados

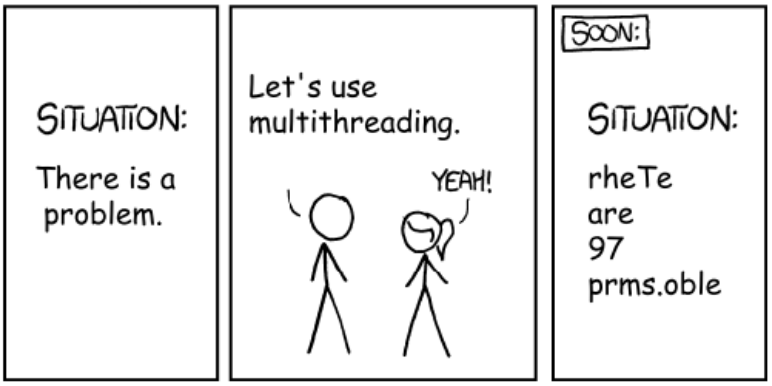
Tempo de Execução dos Exemplos (ms)						
	Número de Processadores Virtuais					
Exemplos	2	5	7	10	20	Entradas
Fibonacci	3009	3012	3006	3003	3010	n = 20
Fatorial	3012	3015	3008	3011	3005	n = 12
Soma	3010	3005	3010	3010	3011	n = 100
Potência	3011	3015	3015	3011	3018	n = 9 e = 9

Pelotas, 27 de Maio de 2021.

3. Conclusão

Observando os resultados, podemos concluir que nem sempre a execução de um programa ocorre de forma mais rápida usando multithreading. Acreditamos que um dos motivos para isso ocorrer, se dá pelo fato de que na implementação da nossa biblioteca, os threads fazem muitos acessos a recursos compartilhados, e acabam ficando ociosos em seções críticas, resultando em menor paralelismo. Programas concorrentes desempenham mal devido ao overhead de criar muitas threads e forçá-las a esperar pelo mutex. Além disso, a nossa escolha de implementar a primitiva sync, de forma que quando vai sincronizar uma tarefa que já está em execução por outro processador virtual, acaba entrando em estado de sleep e volta novamente a procurar nos recursos compartilhados, podendo assim, também ter um impacto nesse resultado negativo.

Apesar disso, conseguimos perceber que obteve-se um grande ganho de processamento em relação a implementações sequenciais, onde conseguimos realizar diversos cálculos, de forma eficiente, até mesmo para tarefas que possuem complexidade alta. Alguns exemplos com entradas grandes, como o de fibonacci, não poderiam ser executados em tempo razoável.



Fonte: https://xkcd.com/

Pelotas, 27 de Maio de 2021.