Ano Lectivo 2015/2016

Concorrência e Paralelismo

Projeto 1

Docente: João Lourenço

Alunos: Nuno Mendonça nº 41623

David Gago nº 41710

Introduction

O objectivo descrito no enunciado é paralelizar um pequeno programa fornecido pelo docente da cadeira. Este programa gera uma imagem através de um algoritmo e um dos grandes desafios é melhorar a performance. Para atingir este objectivo, a estratégia geralmente utilizada é o uso de duas funções da biblioteca do Cilk Plus denominadas por cilk_spawn e cilk_sync, no ciclo que gera a imagem, subdividindo então ciclo gerando várias partes da imagem em separado mas ao mesmo tempo.

Cilk_Spawn gera uma tarefa para ser corrida em paralelo e Cilk_Sync é a função que sincroniza todas as tarefas corridas em paralelos.

O resultado esperado será, com o aumento do número de processadores disponíveis para a execução do programa, que a sua execução seja mais rápida. Em teoria, se o programa demorar 5 segundos a executar com um processador disponível, qualquer teste realizado um número de processadores maior que um tem de ser abaixo de 5s. Também qualquer teste com um número de processadores maior disponíveis que outro teste realizado deve apresentar um tempo de execução mais curto.

Approach

A aproximação geral para a realização deste problema, consistiu inicialmente em adicionar como parâmetro de entrada o número de tarefas que podiam ser realizadas simultaneamente.

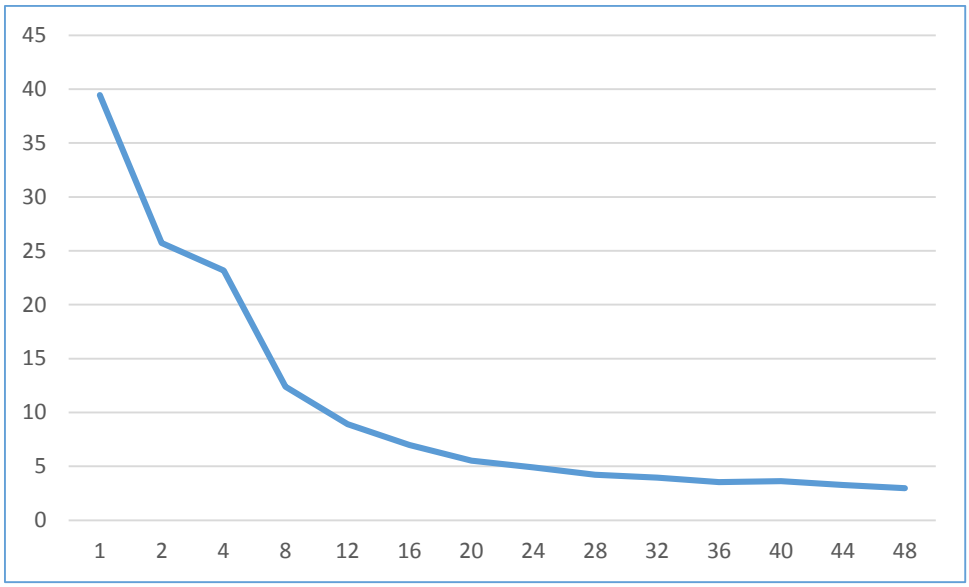
O passo seguinte foi colocar o ciclo do preenchimento da imagem dentro de uma função à parte, onde é possível definir o ponto X inicial e o comprimento a ser preenchido. Sendo assim é possível com o parâmetro de entrada do número de tarefas realizar varias iterações deste ciclo, fazendo assim o preenchimento de várias imagens que em conjunto resultam na imagem final que é o resultado pretendido.

Experimentou-se também separar o ciclo de preenchimento da matriz verticalmente mas isto não trouxe melhorias na performance do programa.

Evaluation

Os resultados que obtemos com os testes que realizamos foram os esperados. Para conferir se a nossa estratégia de melhorar o tempo de execução do programa funcionava correctamente, executamos varias iterações do programa, aumentando o número de processadores disponíveis para a sua execução.

De seguida é apresentado o gráfico com os resultados, em que o eixo X representa a variação do número de processadores disponíveis e o eixo Y representa o tempo que o programa demorou a executar. Em todos os testes executados o tamanho da imagem foi de 8192x8192.



A partir dos resultados obtidos, fomos capazes de concluir que a alteração realizada no código fazia com que o aumento do número de processadores diminuísse o tempo de execução. Este era o resultado esperado, pois com o aumento de processadores disponíveis, mais tarefas podem ser executadas no mesmo espaço de tempo. No limite, se tivéssemos um número infinito de recursos, o tempo de execução da parte paralelizada tenderia para 0s.

Conclusions

Resumindo, os resultados obtidos foram os esperados. Este curto exercício permitiu entender num sentido mais prático a diferença entre parte serializável e parte paralelizável num programa a funcionar. Embora a teoria seja importante, a parte pratica e "ver em acção" os conceitos leccionados é sempre a melhor e mais desafiante maneira de aprender os mesmos.