Estudo de Caso em Escalonamento para Agregados de Computadores

Anderson M. Gomes*, Rafael P. Gonçalves†, Vinícius C. Marques‡, Maurício L. Pilla, Adenauer C. Yamin

Escola de Informática – UCPEL
Rua Félix da Cunha, 412 – Pelotas, RS
Fone (53) 3284–8000
amg1127@gmail.com, rafaelrpg@gmail.com, vini@sirius.inf.br, pilla@ucpel.tche.br, adenauer@ucpel.tche.br

1 Introdução

Uma questão importante para a programação paralela é como as tarefas são escalonadas para execução, maximizando o uso de recursos disponíveis e minimizando o tempo de execução. Dada a falta de alternativas para o escalonamento automático de tarefas em programas MPI, atualmente os programadores tornam-se responsáveis pela eficiência da execução de seus programas, necessitando conhecer os recursos disponíveis e dividindo da melhor forma possível as computações entre os nós de processamento disponíveis.

Neste trabalho, exploramos duas distribuições da carga entre os nós de processamento para uma aplicação de geração de fractais de Mandelbrot. Na primeira, a carga (pontos do fractal) é distribuída entre os nós igualmente. Na segunda, os pontos são divididos por linhas e distribuídos aos nós de processamento conforme os mesmos terminam as suas tarefas anteriores. Nossos experimentos confirmam que a segunda forma é mais adequada para fractais, dada a diferença de tempo de processamento para diferentes áreas, mesmo se considerando o custo adicional de comunicação. Na Seção 2, MPI é brevemente apresentado. Após, a aplicação desenvolvida é discutida na Seção 3. Na Seção 4, os resultados obtidos são apresentados. Finalmente, conclusões e futuros trabalhos são discutidos na Seção 5.

2 Message Passing Interface

O padrão *de-facto* para programação paralela em ambientes distribuídos, como *clusters*, é a *Message Passing Interface* (MPI) [MPI 2004, CAV 2005]. Esse padrão determina um conjunto de facilidades para disparo e comunicação entre processos, estejam esses situados em um mesmo nó de processamento ou em nós conectados por uma rede. Com isso, as peculiaridades e diferenças de *hardware* tornam-se virtualmente transparentes para o programador. Assim, a grande vantagem de MPI sobre especificações proprietárias desenvolvidas para supercomputadores é a sua portabilidade: como todas as

*Bolsista BIC/FAPERGS
†Bolsista BIC/UCPEL
‡Bolsista PROIC – Ensino Médio

implementações de MPI devem seguir a norma, um programa que utilize apenas funções providas pela especificação original podem ser facilmente compiladas e executadas em diferentes máquinas paralelas.

3 Gerador de Fractais

A aplicação para geração e visualização de Fractais de Mandelbrot [MAN 82] trabalha com o formato de imagem *Portable Network Graphics* (PNG) para a criação e a visualização dos Fractais de Mandelbrot. Ela é dividida em três módulos: *(i)* interface de linha de comando; *(ii)* interface de servidor; e *(iii)* visualizador, os quais serão discutidos a seguir.

3.1 Interface de linha de comando

A *Interface de Linha de Comando* consiste em um programa que recebe via argumentos em linha de comando todos os parâmetros necessários para a criação de um Fractal de Mandelbrot. A imagem é enviada para o dispositivo de saída padrão, exceto no caso em que o parâmetro '-o' é fornecido.

A geração da imagem é auxiliada pelo uso da biblioteca GD[1],versão 2.0. O programa não possui suporte a qualquer forma de processamento paralelo; o motivo para o seu desenvolvimento foi o de elaborar comparações de desempenho.

3.2 Interface de servidor

A *Interface de Servidor* é um programa que recebe, via um soquete TCP, os parâmetros necessários para a geração do fractal. A saída é enviada por esse mesmo soquete no momento em que um cliente estabelece uma conexão ao soquete que o programa escuta e lhe passa todos os parâmetros necessários para a geração. A geração da imagem também é feita com a biblioteca GD.

O programa foi projetado para ser executado em um ambiente MPI: em um primeiro instante, a área da imagem que deve ser gerada é dividida entre os nós de processamento e, depois que cada nó produz a sua porção, uma junção é feita e a imagem resultante é enviada ao cliente. A etapa de construção da imagem, neste caso, é feita simultaneamente pelos nós de processamento.

3.3 Visualizador

O *Visualizador* foi desenvolvido com a biblioteca QT[2], versão 4.0, e obtém fractais através de uma chamada à interface de linha de comando ou através de uma conexão a um host que esteja executando a interface de servidor.

O visualizador é interativo e possui diversos recursos para a manipulação de Fractais de Mandelbrot em tempo real: permite aumento e redução do zoom em uma área

[1]Disponível em `www.boutell.com/gd/`
[2]Disponível em `www.trolltech.com`

específica de um fractal, mostra na tela o tempo gasto para a geração da imagem, permite salvar o fractal atual em um arquivo e dispõe de uma ferramenta para editar a paleta de cores que é utilizada na criação dos fractais.

A Figura 1 mostra uma execução do Visualizador com um fractal já calculado. A maior parte da tela é destinada a mostrar o fractal propriamente dito, enquanto que as funções de recarregar o fractal, gerar um zoom na seleção, reduzir zoom, salvar imagem e configurar o visualizador podem ser acessadas por botões no canto esquerdo inferior.

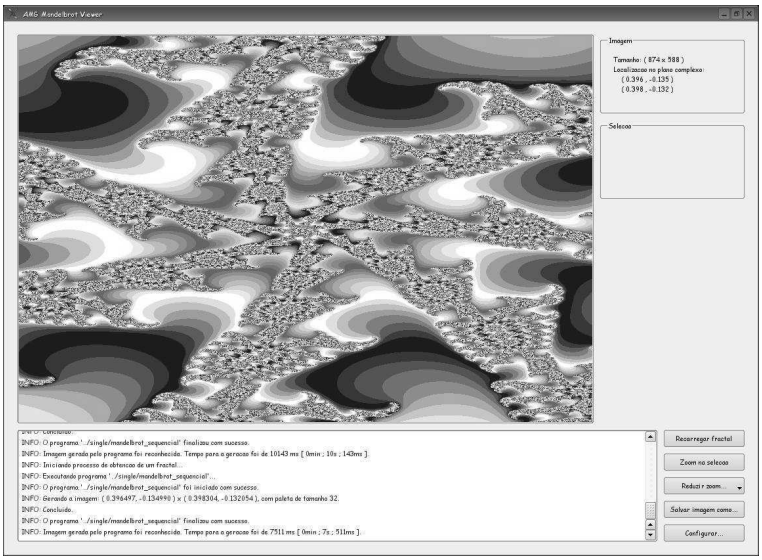

Figura 1: Visualizador de fractais em execução

4 Análise de Desempenho

As medidas de desempenho a seguir foram desenvolvidas no *cluster* H3P da UCPEL, formado por 12 nodos de processamento (dos quais 11 utilizados) com processadores Athlon 2600+ e 192 MB de memória, interligados por uma rede Fast Ethernet. O fractal calculado mede 1024 por 768 pixels com no máximo 1000 iterações por ponto. O tempo foi calculado com ênfase na parte de processamento, sendo que a construção do arquivo PNG não foi contabilizada dentro do tempo de execução.

A Figura 4 apresenta os *speedups* obtidos com as duas possíveis políticas de escalonamento (blocos ou linhas). A linha com o *speedup* ótimo serve apenas como um indicativo do máximo que poderia ser esperado da aplicação. O eixo horizontal apresenta o número de nós de processamento, enquanto que o eixo vertical apresenta o *speedup* sobre a aplicação seqüencial. Usando o escalonamento por blocos (dividindo as áreas entre os escravos igualmente), o *speedup* resultante fica bem abaixo do ideal. Isso deve-se às características do fractal, onde algumas regiões são bastante intensivas em termos de recursos de processamento, enquanto outras são calculadas rapidamente. Assim, enquanto que alguns escravos terminam rapidamente seus cálculos, os escravos que pegaram blocos complexos demoram mais a terminar. Como a computação só é considerada terminada quando todos os escravos terminam, o tempo de execução aumenta e o *speedup* diminui.

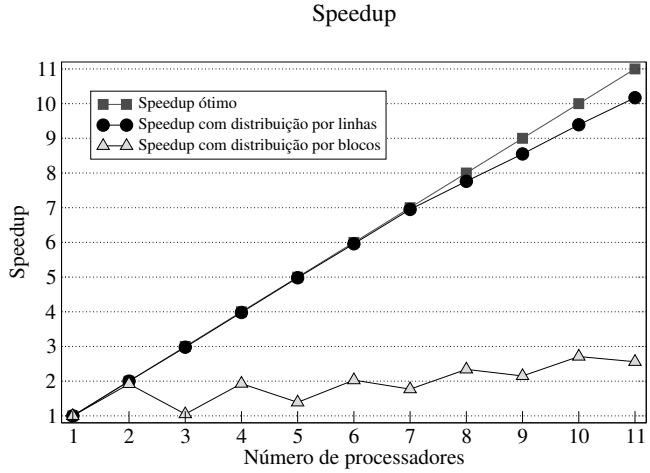

Figura 2: Speedups obtidos

Dividindo o fractal a ser calculado em linhas distribui mais uniformemente a carga entre os escravos. Quando um escravo termina uma linha, ele recebe a próxima linha a ser calculada. Assim, escravos com linhas mais rapidamente calculadas executam sob mais linhas, diminuindo a diferença de tempo de execução entre os escravos e melhorando o desempenho. O *speedup* para esse caso chegou bem próximo do ideal mesmo com grande número de escravos, com eficiência de 0,92 e *speedup* de 10,17 para 11 escravos

5 Conclusões

Neste artigo, mostramos o efeito de duas estratégias de escalonamento para o cálculo em paralelo de fractais de Mandelbrot em agregados de computadores, com uma aplicação incluindo visualizador.

Como trabalho futuro, pretendemos modificar a aplicação para permitir o uso de uma infra-estrutura de grade como o Globus e a norma MPI–2 para dinamicamente disparar e escalonar novos escravos.

Referências

[CAV 2005] CAVALHEIRO, G. G. H. **Princípios de programação concorrente**. 5^a Escola Regional de Alto Desempenho, Canoas, 2005.

[MAN 82] MANDELBROT, B. B. **The fractal geometry of nature**. New York: W. E. Freeman and Company, 1982.

[MPI 2004] MPI FORUM. **Message Passing Interface forum**. Disponível em <http://www.mpi-forum.org>. Acesso em: Set. 2004.