	Computação Paralela UFSCar - Sorocaba Bacharelado em Ciência da Computação Profª. Yeda Regina Venturini Profª. Tiemi Christine Sakata
Projeto final da disciplina de Computação Paralela (2º sem. 2013)	

Grupo

Guilherme Baldo
Rafael Rodrigues Machado
Rodrigo Barbiere

1. Descrição do Problema

Neste trabalho o grupo tem o objetivo de paralelizar uma aplicação que se trata de um cenário muito comum no dia-a-dia na computação, que é a compressão de imagens. A compressão de imagens é utilizada para transformar uma representação pura de imagem pixel-a-pixel (como por exemplo BitMap) em uma representação simplificada que elimina informação redundante, mas que representa a mesma imagem de outra forma.

Hoje existem diferentes algoritmos de compressão de imagens, alguns até ocasionam em perda de qualidade de imagem na compressão. Iremos utilizar técnica a compressão RLE (Run-Length Encoding), que é uma técnica de compressão do tipo Lossless (sem perda) que pode ser empregada em qualquer tipo de arquivo, consistindo em representar valores repetidos consecutivos como um inteiro, representando a contagem de repetições, seguido pelo caracter que está sendo repetido.

Exemplo: entrada:
"WWWWWWWWWWWWBWWWWWWWWWWWWBBBWWWWWWWWWWWWWWWWWWWWWWBWWWWWWWWWWWWWW", saída:
"12W1B12W3B24W1B14W".

O emprego de RLE é mais apropriado para imagens pois há muitos pixels vizinhos com a mesma cor, principalmente em imagens com número muito alto de pixels.

2. Arquitetura do computador onde os testes foram realizados

Para os experimento utilizados, o seguinte ambiente foi utilizado:

Processador	Intel core I7E
Cache	8 MB
Velocidade	3.7 GHz
Memória RAM	3 GB
SO	Linux Mint 15 (VM)

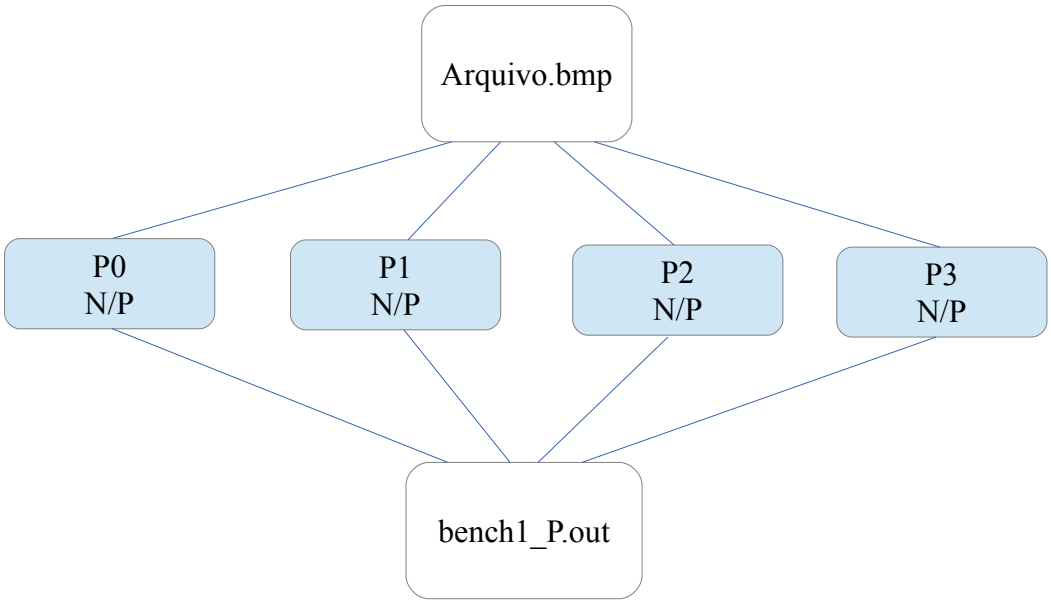
O ambiente utilizado foivirtualizado utilizando Virtual Box, rodando sobre um SO host Windows 7.

2. Divisão de dados

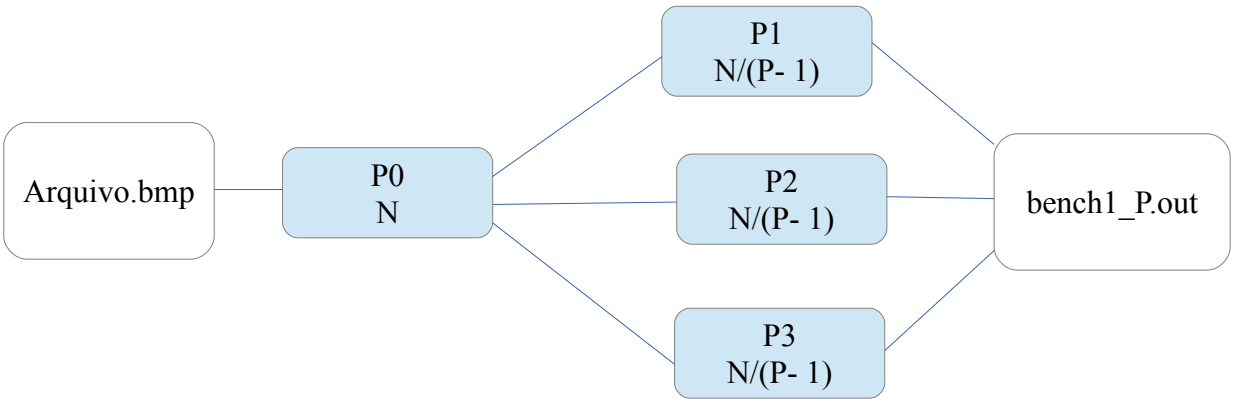
Durante o desenvolvimento foram abordadas duas formas de divisão devido a limitações de desenvolvimento.

Considerando 4 processos paralelos:

Para MPI, sendo N o número de linhas na imagem e P o número de threads ou processos executando a operação, teremos:

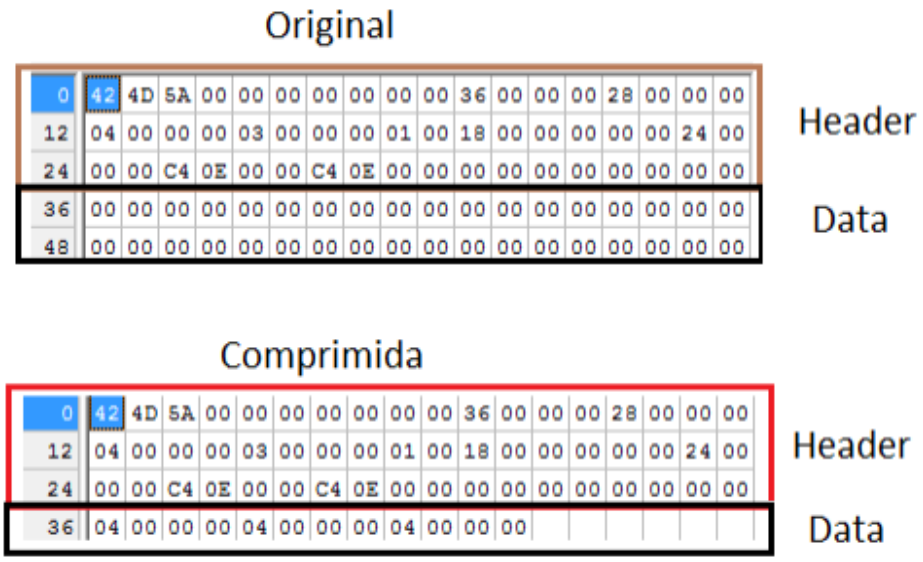


Para Pthreds, sendo N o número de linhas na imagem e P o número de threads ou processos, a abordagem foi atribuir a thread principal a tarefa de ler a imagem para a memória e assim as outras threads podem processar a imagem como sendo uma região de memória compartilhada, conforme a figura abaixo:



Essa abordagem foi necessária devido as funções fread e fwrite não serem threadsafe.

Abaixo segue o resultado de uma compactação de uma imagem com tamanho 4px (largura) por 3px (altura):



Bits de uma imagem após a compressão

Note que o header é mantido nos dois arquivos. Isso é necessário para que seja possível descompactar a imagem corretamente.

3. Tempos das Implementações

WallClock (s)

	Serial	**Paralelo**			
		2	4	8	16
MPI	0.625091	0.316044	0.162407	0.125497	0.169687
Pthreads	0,628354	0,317390	0,164818	0,137371	0,138063

Consumo de memória (Megabytes)

	Serial	**Paralelo**			
		2	4	8	16
MPI	585	582	588	576	576
Pthreads	585	585	585	585	585

Benchmarks

Benchmark 1

Speedup

	Serial	Paralelo			
		2	4	8	16
MPI	1,00000	1,97786	3,84892	4,98092	3,68379
Pthreads	1,00000	1,97975	3,81241	4,57414	4,55121

Eficiência

	Serial	Paralelo			
		2	4	8	16
MPI	1,00000	0,98893	0,96223	0,62262	0,23024
Pthreads	1,00000	0,98988	0,95310	0,57177	0,28445

Análise Parcial

Neste primeiro teste primeiro teste, foi utilizada uma imagem com tamanho igual a 206 MB. Analisando os resultados referentes a MPI, nota-se que conforme o número de processos aumenta, o speedup também aumenta. Isso ocorre devido ao overhead da comunicação entre os processos. O mesmo ocorre com a utilização do pthreads, apesar de a comunicação não existir.. Isso ocorre devido ao escalonamento das threads.

Benchmark 2

Speedup

	Serial	**Paralelo**			
		2	4	8	16
MPI	1,00000	1,92355	3,67283	4,86560	3,57434
Pthreads	1,00000	1,92128	3,65686	4,47248	4,49961

Eficiência

	Serial	**Paralelo**			
		2	4	8	16
MPI	1,00000	0,96177	0,91821	0,60820	0,22340
Pthreads	1,00000	0,96064	0,91421	0,55906	0,28123

Análise Parcial

Neste primeiro teste primeiro teste, foi utilizada uma imagem com tamanho igual a 65 MB.
Assim como na analise anterior, os resultados referentes a MPI apresentam um aumento no speedup, tambem causando devido a comunicação entre os processos. O mesmo ocorre com a utilização do pthreads, apesar de a comunicação não existir..
Acredita-se que isso ocorra na utilização do pthreads, devido ao fato de a imagem ser carregada or completo em memória. Isso causaria a ocorrência de vários cache misses, no momento em que as threads realizam a compressão das cadeias de bytes.

Análise Final

Após verificação dos dados coletados, é possível notar que ambas tecnologias apresentam um desenpenho muito próximo.

Análise da Escalabilidade

MPI

Como foi verificado anteriormente, a eficiência da implementação utilizando MPI diminui conforme aumenta o número de processos. Caindo drasticamente, cerca de três vezes menor, no momento em que o número de processos é maior do que o número de núcleos (8 núcleos no caso do ambiente utilizado para os testes).

Pthreads

O mesmo comportamento observado na utilização do MPI ocorre com a utilização do pthreads. No qual a eficiência diminui de forma brusca no momento em que o numero de threads ultrapassa o número de núcleos.

Ferramentas Utilizadas

Para a coleta dastas informações foi utilizado o commando ps disponível no sistema operacional utilizado.