Relatório de Computação de Alto Desempenho

Ordenação em duas fases utilizando Sockets e Bag of Tasks

Aluno: Gabriel Henrique Martinez Saraiva

Professor: Dr. Aleardo Manacero Jr.

Introdução

A redução do custo de componentes de médio/alto desempenho para computação pessoal, incentiva a utilização de clusters e sistemas com vários núcleos na aceleração da execução de aplicações que demandam muito processamento.

Dessa forma esse trabalho visa aprender a programar para utilizando sockets para que seja possível explorar a capacidade computacional desses sistemas e avaliar o desempenho de um programa que faz ordenação de um vetor relativamente grande utilizando-se das arquiteturas mono e multicomputador. Foram comparadas as execuções com diferentes números de nós de processamento e diferentes números de núcleos utilizando diferentes cargas de trabalho visando analisar como a relação entre tamanho da tarefa e numero de nós interfere no speedup do paradigma utilizado.

Os resultados se mostraram positivos. No cluster, foi possível obter uma redução do tempo de execução em 6.9x com o uso dos 8 nós. Quanto a execução em um computador de 6 núcleos e 12 threads o ganho de velocidade foi na ordem de 7.4x.

Objetivos

O objetivo do trabalho é mostrar a variação do tempo de execução e, portanto o speedup, e analisar como essa métrica varia de acordo com o ambiente utilizado, número de elementos de processamento e carga de trabalho. Não é objetivo desse trabalho comparar o speedup entre a versão desenvolvida e uma ordenação com desempenho ótimo, mesmo que seja sequencial, pois os parâmetros fornecidos para o desenvolvimento do projeto tornam essa comparação desonesta.

Metodologia

Esse trabalho foi realizado utilizando um cluster heterogêneo com 8 nós (Intel Core i7 3770@3.40GHz, 16GB DDR3) e um computador com hardware de servidor (Intel Xeon E5-2420@1.90GHz, 32GB DDR3). Essa característica heterogênea do cluster afeta o speedup drasticamente.

A comunicação do sistema foi feita utilizando-se sockets do UNIX. E a medição de tempo foi feita utilizando bibliotecas de tempo real em C que fornecem uma resolução de tempo aceitável para esse trabalho.

Foram feitas amostras com a ordenação de vetores de 200, 400, 800 e 1600 mil elementos de forma paralela utilizando bag of tasks com 1, 2, 4 e 8 elementos de processamento. As cargas de trabalho para o bubble-sort foram divididas em 32 tarefas de tamanhos iguais.

Uma característica não explorada que afeta severamente o resultado obtido foi o não uso do paralelismo dentro dos elementos de processamento no cluster. Sendo que cada nó trabalha de forma s seqüencial. Isso se da devido à complexidade paralelizar os algoritmos de ordenação, que não era o objetivo do trabalho atual.

Para obter uma análise estatística que amenize a influência de elementos externos ao sistema cada teste foi realizado 5 vezes não consecutivas e a média dos tempos foi utilizada.

O speedup aqui é definido como o tempo de execução sequencial (em um nó) dividido pelo tempo de execução em paralelo (vários nós).

Resultados

Os resultados são exibidos para os testes feitos com o cluster (CLUSTER) e os testes feitos com o *Xeon* (ZEUS).

As métricas utilizadas foram:

- Tempo total de execução;
- Tempo de execução total da fase de Bubble-Sort (processamento + comunicação);
- Tempo de execução total da fase de Merge-Sort. Essa fase não possui comunicação, uma vez que é feita sequencialmente;

Com essas métricas é possível definir o speedup geral e analisar os gargalos do sistema.

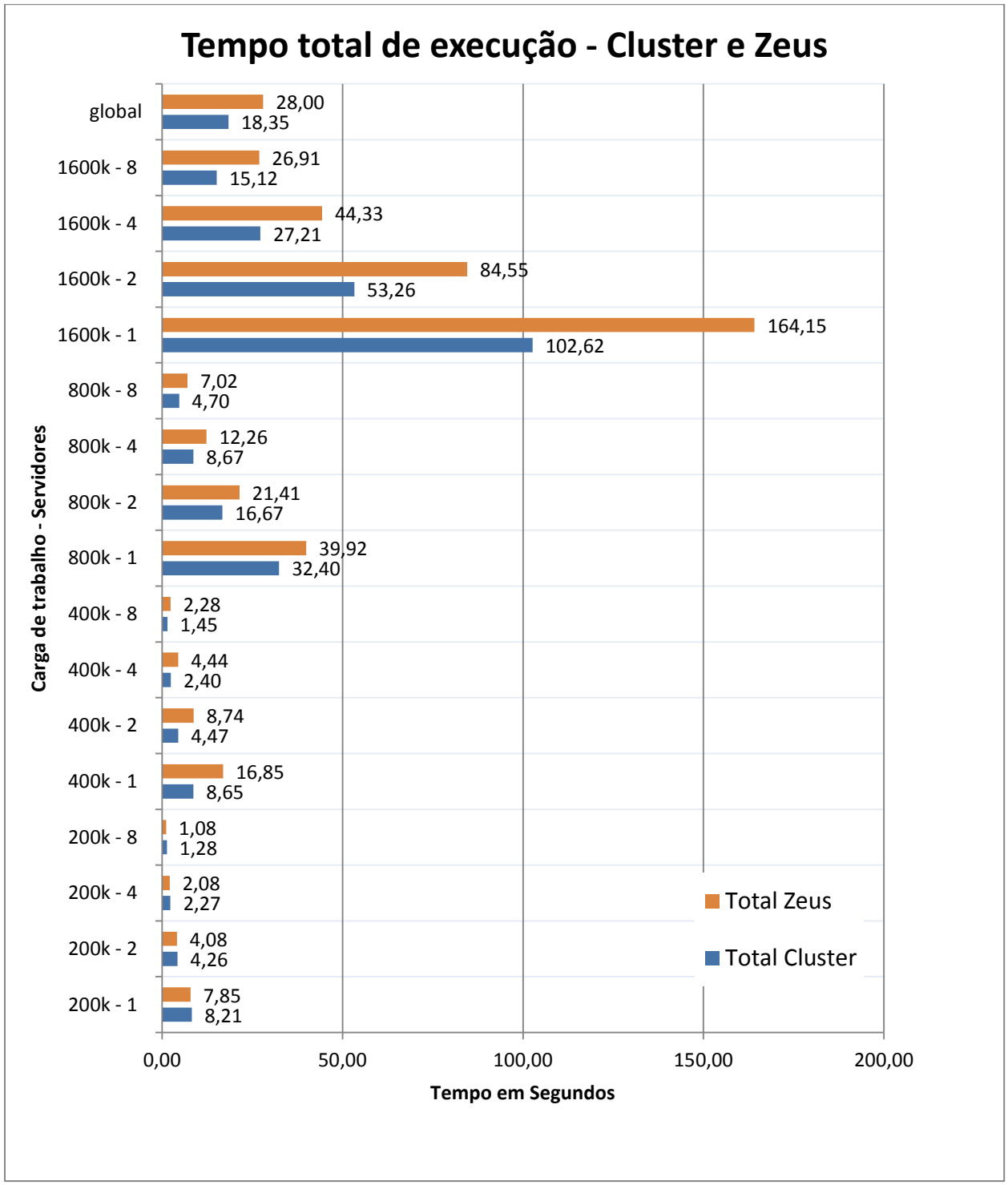

Gráfico 1 - Média dos tempos totais de execução de cada caso de teste

Como pode ser visto no gráfico 1, o maior speedup surge ao se utilizar 8 elementos de processamento (ep). Esse speedup chega a ser de até 6.9x para o Cluster e 7.4x no Zeus. A marca "global" é a media de todos os casos de execução, que mostra o cluster mais rápido que o Zeus.

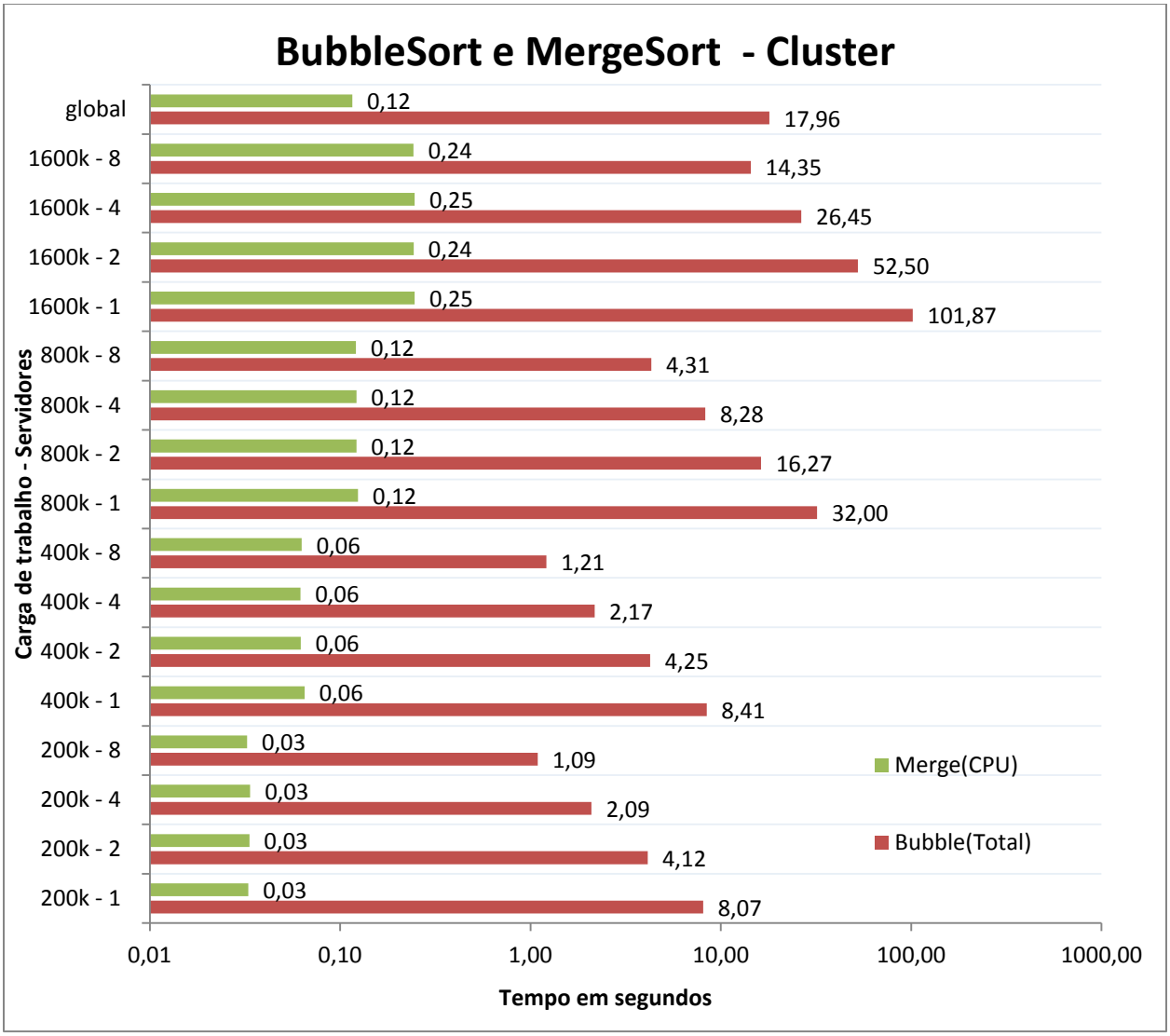

Gráfico 2 - Tempo de execução das ordenações do Cluster

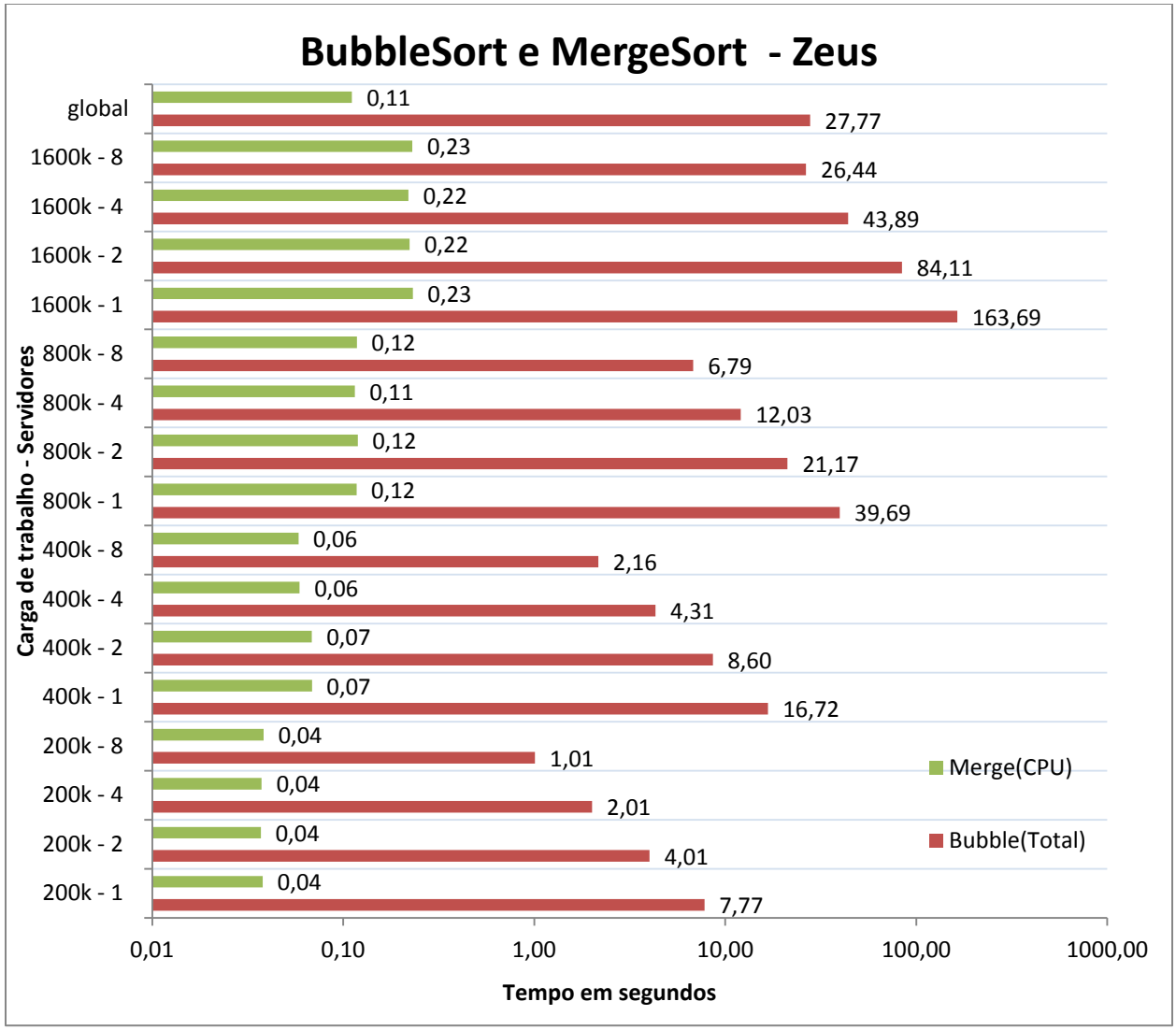

Gráfico 3 - Tempo de execução das ordenações - ZEUS

Nos gráficos 2 e 3 é possível perceber o que limita o ganho de desempenho é o bubble-sort que tem um tempo de execução muito grande comparado com o merge-sort e o resto das tarefas, o que era esperado. Na média global é possível notar que o Bubble-Sort consome quase que a totalidade do tempo de execução das ordenações.

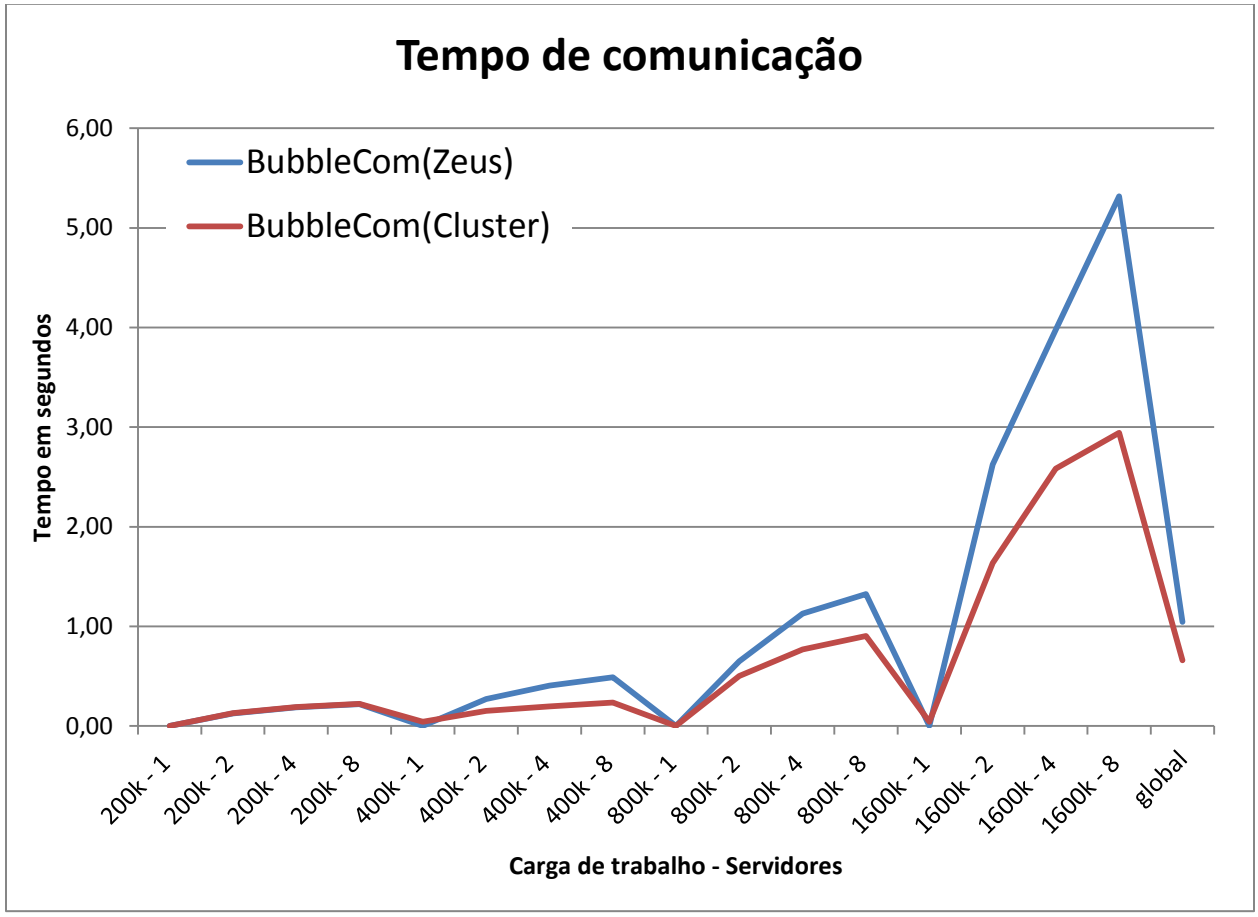

Gráfico 4 - Tempo de comunicação

O gráfico 4 mostra o tempo total de comunicação para o Bubble-Sort nos casos de teste. Nesse trabalho o merge-sort não possui tempo de comunicação pois é feito sequencialmente. Assim é possível notar que a comunicação aumenta conforme crescem o número de nós. A surpresa nesse gráfico veio do fato do Zeus consumir quase o dobro do tempo em comunicação do que o Cluster, mesmo utilizando a loopback para fazer sua comunicação, evitando assim o tempo de propagação do sinal pela rede (ainda que mínimo) e roteamento no switch que o cluster sofre. O motivo mais aparente para esse gargalo é o S.O. que tem que gerenciar toda a comunicação feita pelos sockets sozinho, enquanto que o o cluster que divide essa tarefa com os nós.

Tabela 1 - Listagem de Speedup em cada teste

Speedup		
Carga - Servidores	**Zeus**	**Cluster**
200k - 1	1.00	1.00
200k - 2	1.92	1.93
200k - 4	3.77	3.62
200k - 8	7.24	6.39
400k - 1	1.00	1.00
400k - 2	1.93	1.94
400k - 4	3.80	3.60
400k - 8	7.39	5.99
800k - 1	1.00	1.00
800k - 2	1.86	1.94
800k - 4	3.25	3.74
800k - 8	5.68	6.89
1600k - 1	1.00	1.00
1600k - 2	1.94	1.93
1600k - 4	3.70	3.77
1600k - 8	6.10	6.79
Máximo	7.39	6.89
Mínimo	1.86	1.93

Na tabela 1 é possível analisar os ganhos de velocidade na execução de cada grupo de testes. Nela os speedups estão calculados em relação ao grupo de mesma carga de trabalho. Chegando a apresentar o máximo para o Zeus de 7.39x em 400k, com 8 servidores e para o Cluster o ganho é de 6.89x em 800k com 8 servidores. Esses valores estão dentro do que era esperado.

Conclusões

Com o trabalho realizado foi possível aprender muito sobre sockets, Unix e HPC em geral. Com os resultados obtidos é possível notar que o desempenho obtido depende muito do nível de programação paralela utilizado, e da forma como a solução do problema é tratada. Esse caso mostrou que sockets são muito bons para sistemas multicomputadores, mas que em sistemas onde é possível utilizar memória compartilhada, o uso de threads ainda se mostra mais vantajoso devido ao menor custo para o S.O.

Embora os testes no Zeus apresentem um speedup maior com o aumento de paralelismo, o Cluster ainda sim apresentou tempos absolutos muito menores, mesmo executando apenas um processo por nó, deixando assim uma enorme margem de redução do tempo de execução a ser explorada em trabalhos futuros.