Broadcast implementado em MPI

Luis Felipe Risch e Paulo Mateus Luza Alves

Departamento de Informática – Universidade Federal do Paraná (UFPR)

pmla20@inf.ufpr.br (GRR20203945) e lfr20@inf.ufpr.br (GRR20203940)

Implementação do algoritmo

O algoritmo é uma implementação do algoritmo de broadcast utilizando o modelo de mensagens MPI (Message Passing Interface). O objetivo do algoritmo é permitir que um processo raíz envie uma mensagem para todos os outros processos em um grupo de comunicação.

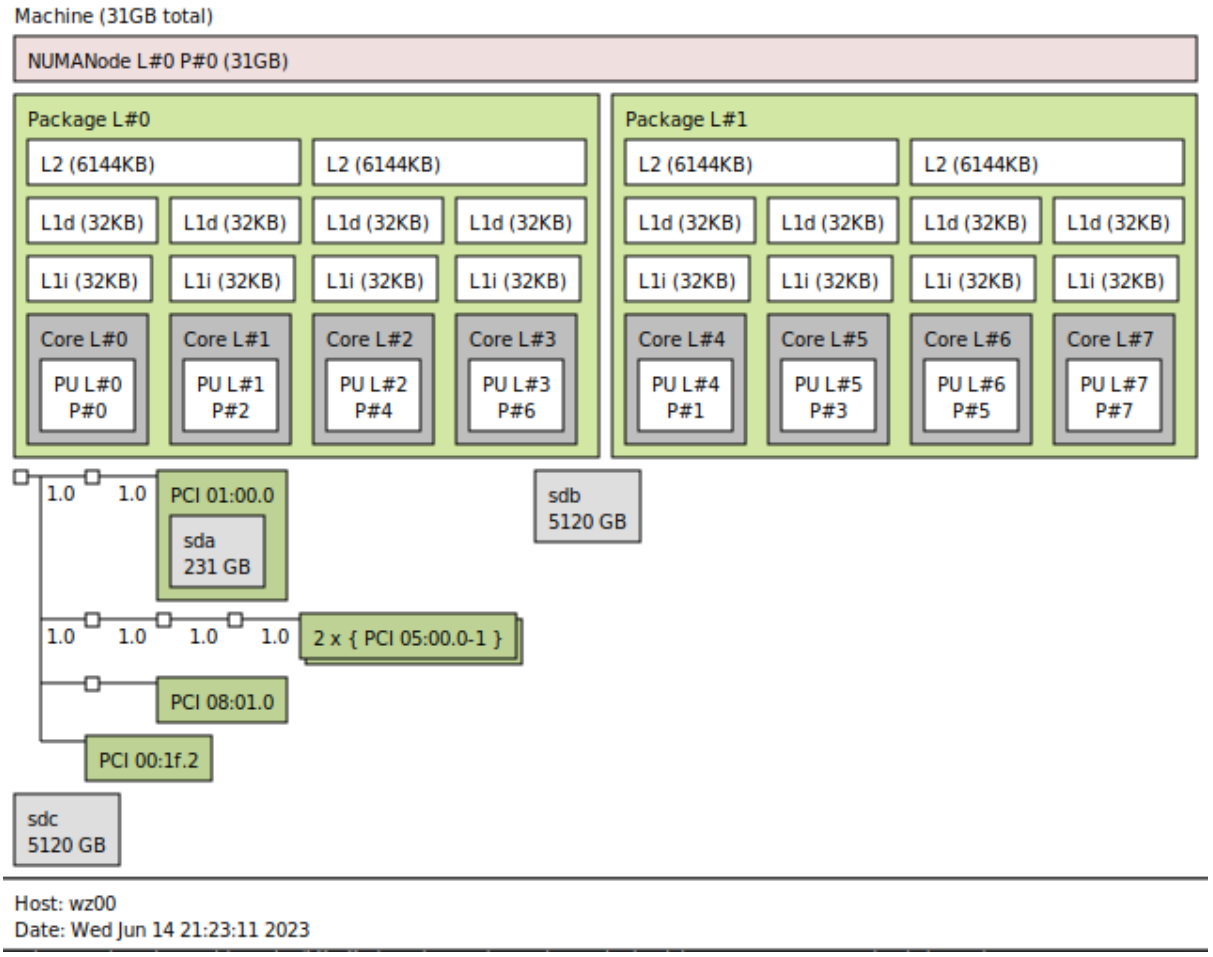
Para executar tal tarefa, o algoritmo utiliza de duas ferramentas, sendo elas a passagem das informações em um arquitetura de árvore não binária e o *split* da mensagem em duas metades, que ao fim são "trocadas" para completar seu conteúdo. Desta forma, cada nodo recebe sua metade de um nodo anterior, ou seja, com menor rank, e transmite essa mesma mensagem para um nodo posterior, ou seja, de maior rank. Por fim, essas metades são trocadas entre os nodos de paridade distintas para completar o conteúdo.

O algoritmo construído funciona da seguinte forma:

1. Verifica se o número de nodos é menor ou igual a um. Se isso for verdade, não há necessidade de continuar a execução do algoritmo por dois motivos possíveis:
    a. Se houver apenas um nodo, não é necessário fazer nada, pois a mensagem já está contida em todos os nodos do grupo. Portanto, não há necessidade de troca de mensagem;
    b. Se não houver nenhum nodo, também não há mensagem para ser trocada. Portanto, não há ação a ser realizada;
2. Inicializa variáveis, incluindo flags para determinar se o processo é ímpar (*is_odd*), inteiros (*change_size e loop_size*) e ponteiros de inteiro longo (*internal_buffer, loop_buffer e change_buffer*);
3. Divide o *buffer* principal em dois *buffers*(*loop_buffer e change_buffer*) que vão apontar para o início ou para metade do buffer principal respectivamente, e calculamos os tamanhos destes respectivos *buffers*, dependendo da paridade do nodo em questão;
4. Após todos os cálculos e tamanhos definidos, o algoritmo inicia o processo de troca de mensagem, iniciando com apenas o nodo raiz enviando sua metade e os demais ficando em aguardo do recebimento;
5. Em seguida, inicia-se um loop de envio das metades, onde a cada loop, os nodos que enviam suas mensagens são os quais o id é inferior a variável do loop *np*, a qual dobra a cada iteração. Vale ressaltar que esta troca de mensagem é feita entre nodos de mesma paridade, assim, todos os ímpares possuem a metade direita da mensagem e os pares a metade esquerda;
6. Por fim, inicia-se o processo de troca de metades, ou seja, os nodos pares buscam a metade faltante nos nodos ímpares e vice e versa.

Descrição do processador usado

Segue a imagem do processador obtido com o programa *lstopo.*



Para complementar a imagem acima, segue as informações obtidas ao rodar o programa *lscpu*:

Hostname: wz00;

Processador: Intel(R) Xeon(R) CPU E5462 @ 2.80GHz

CPU MHz: 2793,018;

L1d cache: 256 KiB;

L1i cache: 256 KiB;

L2 cache: 24 MiB;

flags: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ht tm pbe syscall nx lm constant_tsc arch_perfmon pebs bts rep_good nopl cpuid aperfmperf pni dtes 64 monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr pdcm dca sse4_1 lahf_lm pti tpr_shadow vnmi flexpriority vpid dtherm

Descrição dos experimentos

Para realizar os experimentos solicitados pela especificação do trabalho, seguimos os seguintes passos:

1. Agendamos 10 jobs nos hosts de processamento, de forma exclusiva, para rodar o programa myBroadcast_rb com a versão de broadcast desenvolvida pelos autores, usando 8000 mensagens de tamanho 4 Kb. Aguardamos a conclusão das execuções e extraímos as principais informações (tempo, latência e vazão) para preencher uma planilha de dados.
2. Agendamos mais 10 jobs nos hosts de processamento, de forma exclusiva, para rodar o programa myBroadcast_rb com a versão de broadcast desenvolvida pelos autores, porém utilizando 8000 mensagens de tamanho 16 Kb. Novamente, esperamos que os resultados fossem computados e extraímos as principais informações para a planilha de dados.
3. Em seguida, agendamos 10 jobs nos hosts de processamento, de forma exclusiva, para rodar o programa myBroadcast_rb com a versão de broadcast do MPI (Message Passing Interface) para 8000 mensagens de tamanho 4 Kb. Aguardamos os resultados e adicionamos as principais informações na planilha de dados.
4. Finalmente, agendamos mais 10 jobs nos hosts de processamento, de forma exclusiva, para rodar o programa myBroadcast_rb com a versão de broadcast do MPI para 8000 mensagens de tamanho 16 Kb. Após a conclusão dos experimentos, extraímos as principais informações e as adicionamos à planilha de dados.

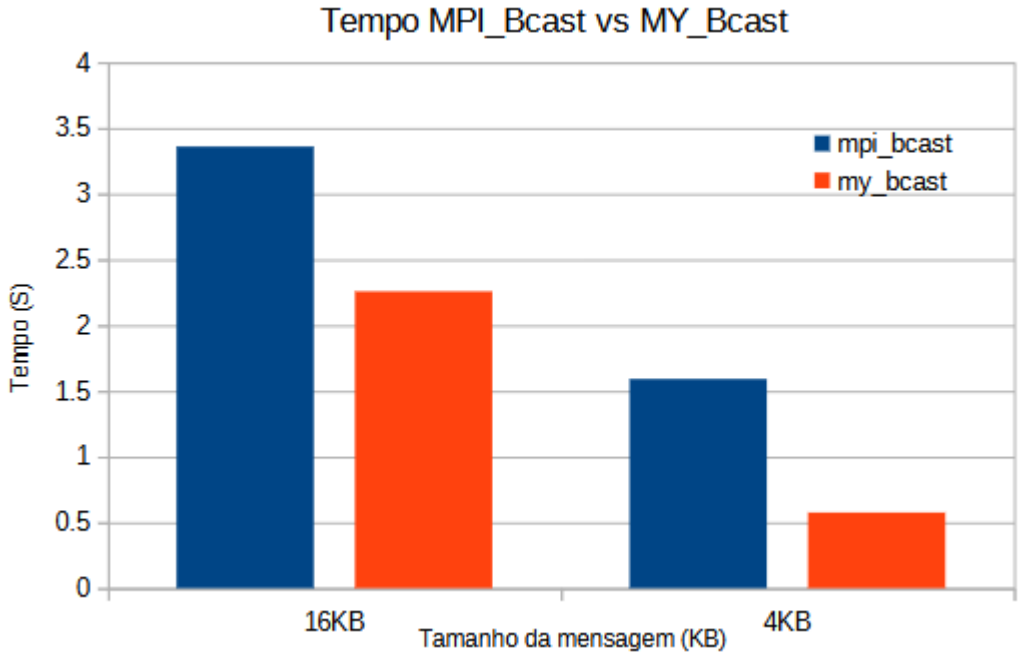
Um detalhe importante é que para todos os experimentos usamos 8 nodos e 1 processo por nodo.

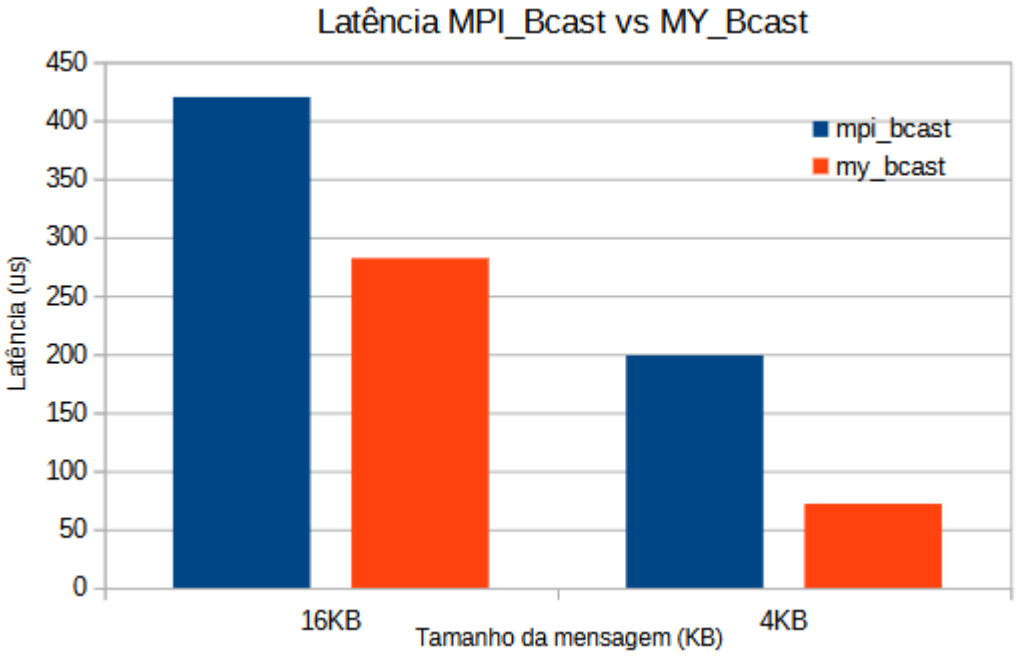
Após preenchermos a planilha com todas as informações necessárias, calculamos as respectivas médias e geramos gráficos para comparar os resultados obtidos pelo broadcast implementado pelos autores com os do próprio MPI, os quais serão apresentados nas próximas seções.

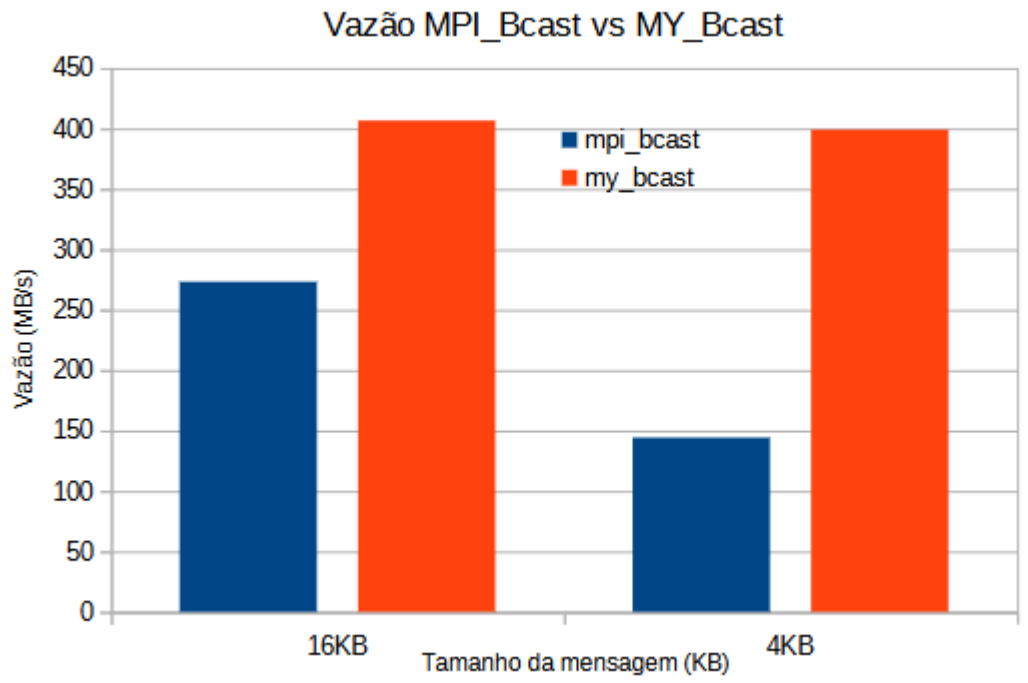
Tabela de dados - Sumarizando a vazão, aceleração e latência

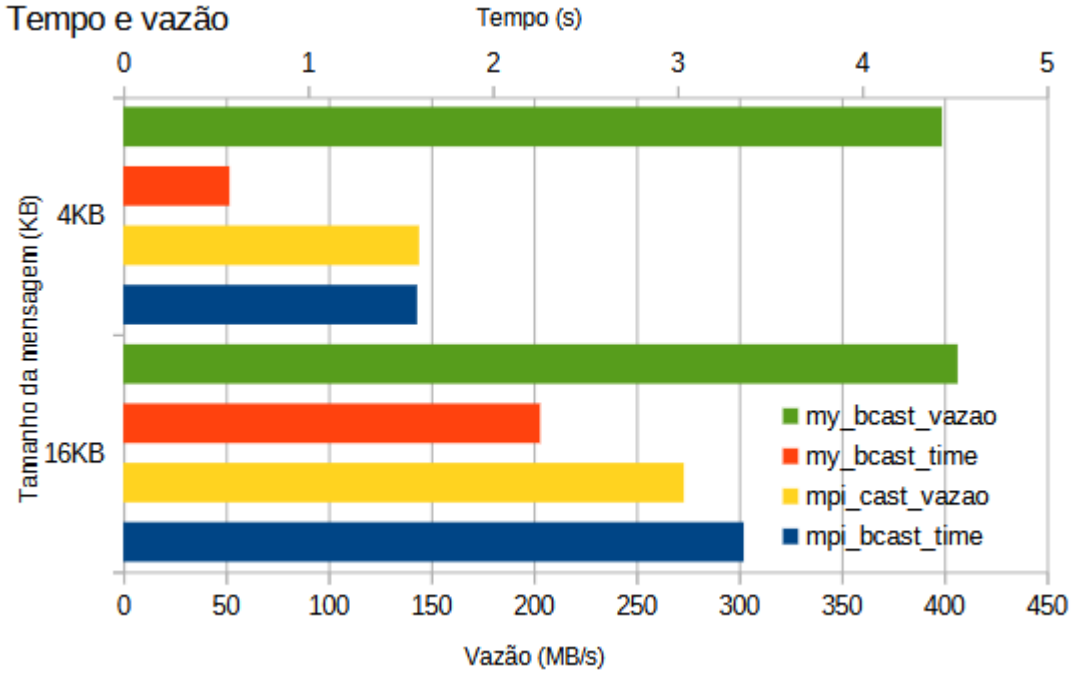
Valores Médios				
	my_bcast_16k	**my_bcast_4k**	**mpi_bcast_16k**	**mpi_bcast_4k**
tempo (s)	2.257	0.575	3.359	1.591
vazão (MB/S)	406.55	398.919	273.137	144.142
latência (us)	282.105	71.885	419.904	198.919

Gráfico da vazão, aceleração e latência






Vazão MPI_Bcast vs MY_Bcast
450
400
350
300
250
200
150
100
50
0
Vazão (MB/s)
mpi_bcast
my_bcast
16KB
4KB
Tamanho da mensagem (KB)


Tempo e vazão
Tempo (s)
0
1
2
3
4
5
Tamanho da mensagem (KB)
4KB
16KB
my_bcast_vazao
my_bcast_time
mpi_cast_vazao
mpi_bcast_time
0
50
100
150
200
250
300
350
400
450
Vazão (MB/s)