Transformada da distância - Otimização através de soluções híbridas de paralelismo

Afonso, João - *A71874,* Fernandes, Gabriel - *A71492*

***Resumo*—Depois de estudos prévios sobre paralelismo, usando como caso de estudo o algoritmo da transformada da distância, decidimos construir uma implementação híbrida usando as duas ferramentas (MPI e OpenMP) de paradigmas distintos de computação paralela. Visto que em MPI o *bottleneck* está na comunicação, decidimos investir algum tempo em arranjar uma maneira de o reduzir. Desta necessidade, surgiu-nos a ideia de implementar um algoritmo de compressão já existente de modo a diminuir o impacto dos tempos de comunicação. Como vamos explicar em seguida, esta modificação permitiu-nos chegar a conclusões interessantes, bem como espremer ainda mais da performance que já tínhamos alcançado anteriormente.**

Palavras-Chave*—*Transformada da Distância, Optimização, Paralelismo, OpenMP, MPI.

I. Caso de Estudo

O caso de estudo é a analise da performance de uma implementação paralela de um algoritmo dado, neste caso é a transformada da distância.

De uma maneira geral, este algoritmo é aplicado em áreas como a de processamento de imagem.

A sua função é transformar uma imagem binária que contenha um objeto (a branco) inserido num fundo (a preto) numa imagem *greyscale* (apenas com tons monocromáticos) em que cada pixel vá sendo gradualmente mais escuro conforme se encontre mais perto da fronteira.

Ora, para o algoritmo implementado, a sua execução consiste em percorrer a imagem coluna-a-coluna duas vezes e linha-a-linha duas vezes, conforme ilustrado na figura 1.

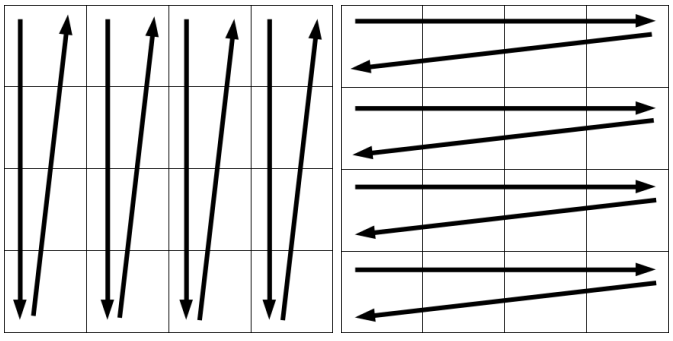


Figura 1. Quatro varrimentos da matriz

Devido a esta independência de dados, a execução de cada fase pode ser paralelizada, sendo este o ponto de partida e o foco principal do nosso estudo.

A. Análise assimptótica do algoritmo

O algoritmo foi implementado de forma a ter uma complexidade linear, de acordo com o trabalho desenvolvido por Meijster [3], representado no seguinte excerto de pseudo-código:

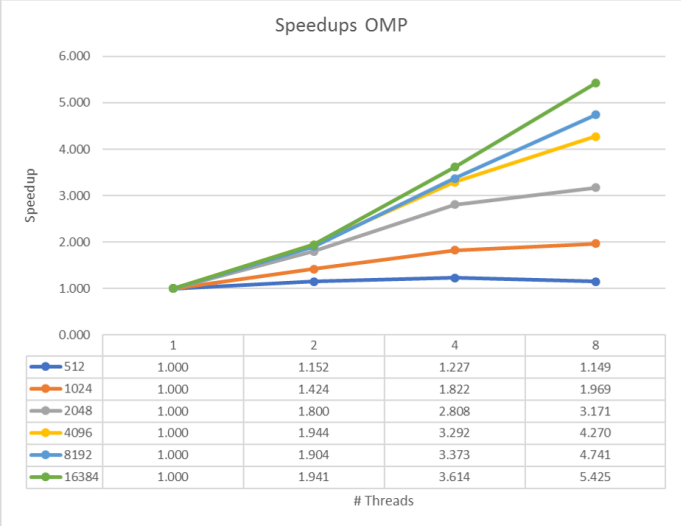
```
(* Varrimento vertical *)
para todo x pertencente a [0..m-1] faz
    (* primeiro varrimento *)
    se b[x,0] então
        g[x,0] := 0
    senão
        g[x,0] := infinito
    para y := 1 até n-1 faz
        se b[x,y] então
            g[x,y] := 0
        senão
            g[x,y] := 1 + g[x,y-1]
    (* segundo varrimento*)
    para y := n-2 até 0 faz
        se g[x,y+1] < g[x,y] então
            g[x,y] := (1 + g[x,y+1]

(* Varrimento horizontal *)
para todo y pertencente a [0..n-1] faz
    q := 0; s[0] := 0; t[0] := 0
    (* terceiro varrimento *)
    para u := 1 até m-1 faz
        enquanto
            q>=0 e f(t[q],s[q])>f(t[q],u) faz
                q := q-1
        se q<0 então
            q := 0; s[0] := u
        senão
            w := 1 + Sep (s[q],u)
            se w<m então
                q:=q+1; s[q]:=u; t[q]:=w
    (* quarto varrimento *)
    para u := m-1 até 0 faz
        dt[u,y] := f(u,s[q])
        se u=t[q] então
            q := q-1
```

Assim, são executadas duas fases, uma que processa a imagem verticalmente e outra horizontalmente, sendo que cada uma destas fases processa a imagem nos dois sentidos possíveis. Retira-se então que a complexidade do algoritmo é $4 \times N$, sendo N o número de pixeis da imagem.

II. Otimização das versões anteriores

Todas as versões anteriores estavam já bem exploradas no que diz respeito ao algoritmo em si (computação da primeira e segunda fases), não nos tendo sido possível efetuar nenhuma melhoria a esse respeito.

As versões sequencial e *OpenMP* utilizavam uma leitura e escrita ineficientes das imagens em disco, que não permitia uma execução em tempo aceitável dos *datasets* de maiores dimensões, tendo sido portanto otimizadas. Os *speedups* conseguidos recorrendo ao paradigma de memória partilhada encontram-se representados na Figura 2.



	1	2	4	8
512	1.000	1.152	1.227	1.149
1024	1.000	1.424	1.822	1.969
2048	1.000	1.800	2.808	3.171
4096	1.000	1.944	3.292	4.270
8192	1.000	1.904	3.373	4.741
16384	1.000	1.941	3.614	5.425

Figura 2. Speedups OpenMP

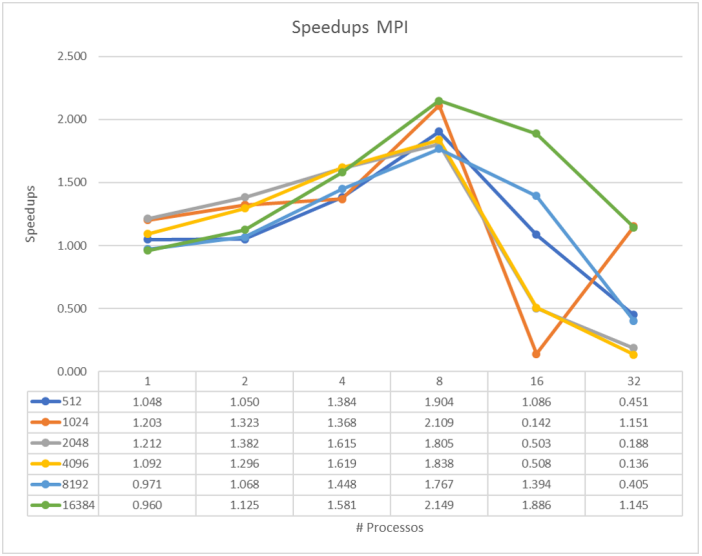
No que diz respeito à versão MPI, conseguimos encontrar um padrão que nos permitiu alterar a operação *MPI_Alltoallv* pela operação *MPI_Alltoall* na implementação que já se tinha revelado ser a mais rápida, conseguido assim reduções significativa no tempo de comunicação, principalmente entre máquinas distintas. No entanto supomos que essas discrepâncias se devem principalmente a uma elevada utilização da ligação *ethernet* aquando da execução dos testes anteriores.

III. Solução Hibrida

Perante os resultados obtidos para os paradigmas de memória partilhada e distribuída, uma vez que a versão em memória partilhada (OpenMP) tem *speedups* relativamente superiores à versão *MPI*, decidimos dar preferência à primeira na composição de uma solução onde estas funcionassem conjuntamente. Porém, não queríamos eliminar a possibilidade de computar imagens de tamanho elevado (superior à capacidade da memória *RAM*), sem ter de estar consecutivamente a aceder a disco.

A. Distribuição de carga entre OpenMP e MPI

Uma vez que a paralelização entre máquinas apenas era suportada pelo *MPI*, e tendo já presente a maior performance



	1	2	4	8	16	32
512	1.048	1.050	1.384	1.904	1.086	0.451
1024	1.203	1.323	1.368	2.109	0.142	1.151
2048	1.212	1.382	1.615	1.805	0.503	0.188
4096	1.092	1.296	1.619	1.838	0.508	0.136
8192	0.971	1.068	1.448	1.767	1.394	0.405
16384	0.960	1.125	1.581	2.149	1.886	1.145

Figura 3. Speedups MPI

conseguida pelo *OpenMP*, decidimos implementar uma arquitetura em que cada máquina executasse apenas um processo, contendo o número máximo possível de *threads* associadas.

Esta versão consistia apenas na fusão das implementações já efetuadas, como tal procurámos encontrar uma forma de a otimizar.

Deste modo, visto que o processo de comunicação ainda representava o maior *bottleneck* do nosso algoritmo, refletimos ainda mais na sua otimização, ocorrendo-nos a ideia de comprimir a imagem antes do envio, descomprimindo-a à chegada.

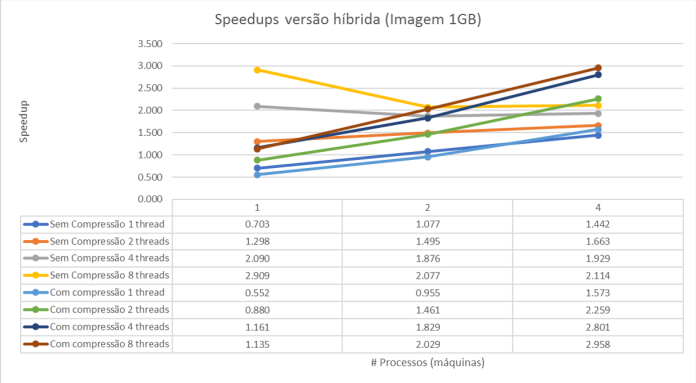
Ao explorar melhor esta ideia, de forma a tirar partido da morfologia da imagem após a primeira fase de execução (maioritariamente preta, com linhas horizontais em tons de cinza), decidimos implementar o algoritmo *LZW*, ou Lempel-Ziv-Welch, utilizado no formato *GIF* e no comando *compress* dos sistemas *unix*.

Para efetuar a compressão da matriz, o algoritmo começa por criar um dicionário contendo os possíveis valores para cada elemento da matriz, passo que ignorámos por poder ser feita uma conversão direta, isto é, $dicionario[i] = i$. De seguida é percorrida a matriz, as sequencias encontradas neste processo vão sendo acrescentadas ao dicionário e substituídas na matriz pelo valor inteiro correspondente à sua posição no dicionário. Este processo permite que o dicionário seja reconstruido em processo de descompressão, não tendo portanto de ser enviado em conjunto com a imagem comprimida.

A implementação eficiente do algoritmo revelou-se bastante mais desafiadora do que a própria transformada da distância, uma vez o algoritmo implica que seja encontrado o melhor balanceamento entre o tempo de compressão e a redução ao tamanho da imagem, com consequente redução no tempo de envio.

Assim, e uma vez que a exploração do algoritmo não está diretamente relacionada com o enunciado do projeto, limitámo-nos a implementar aquilo que podemos referir-nos como uma

versão *naive* e a otimizar alguns processos iterativos de forma a garantir que o algoritmo corre em tempo linear. Os resultados das versões com e sem compressão encontram-se expressos na Figura 4.



	1	2	4
Sem Compressão 1 thread	0.703	1.077	1.442
Sem Compressão 2 threads	1.298	1.495	1.663
Sem Compressão 4 threads	2.090	1.876	1.929
Sem Compressão 8 threads	2.909	2.077	2.114
Com compressão 1 thread	0.552	0.955	1.573
Com compressão 2 threads	0.880	1.461	2.259
Com compressão 4 threads	1.161	1.829	2.801
Com compressão 8 threads	1.135	2.029	2.958

Figura 4. Speedups das versões híbridas

Apesar dos tempos de execução serem bastante dependentes das configurações que definirmos para o algoritmo de compressão (como o tamanho máximo do dicionário), bem como da morfologia das imagens, conseguimos, para um maior número de máquinas, tempos de comunicação inferiores aos obtidos pela versão sem compressão. Uma vez que a otimização do algoritmo ainda pode ser bastante explorada, e que permite que a transformada da distância escale melhor para imagens maiores, bem como par um maior número de máquinas, consideramos ter encontrado um bom ponto de partida para futuras otimizações.

IV. Análise dos resultados

A. Tempo de Comunicação versus Tempo de Computação

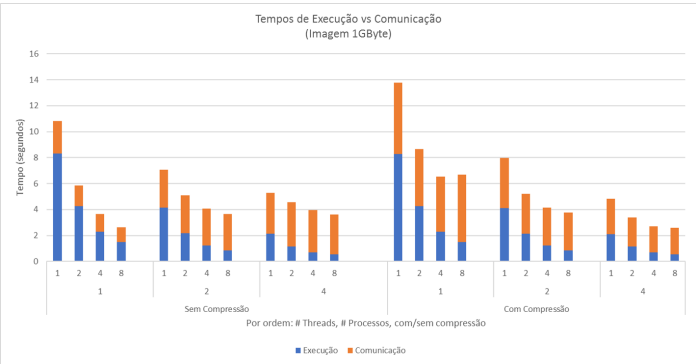


Figura 5. Comparação entre tempo de computação e tempo de comunicação para várias versões híbridas

Relativamente à eficiência do algoritmo, podemos observar que o tempo de comunicação aumenta para a versão sem compressão da implementação híbrida, conforme se aumentam o número de máquinas. Contudo, podemos reparar que este diminui com uma quantidade maior de máquinas para a versão com compressão. Podemos portanto concluir que a compressão é rentável quando temos comunicação entre várias máquinas, verificando-se o oposto para a versão sem compressão.

B. Speedups

Analisando os *speedups* das várias versões híbridas (com ou sem compressão), OpenMP e MPI podemos ver que: a versão MPI não escala de acordo com o número de máquinas disponíveis (devido à comunicação); a versão híbrida sem compressão, também não escala, embora seja melhor que a versão MPI devido a utilização do OpenMP dentro de cada máquina; a versão híbrida com compressão escala bastante bem e a versão OpenMP tem *speedup* melhor que o resto das outras versões executando apenas numa máquina.

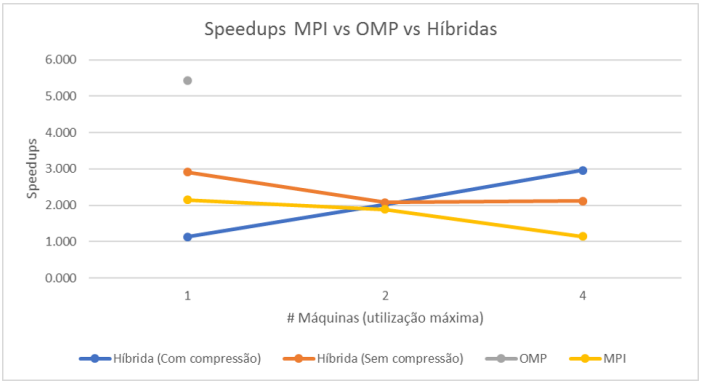


Figura 6. Speedups (todas as versões)

Para a implementação do algoritmo são utilizadas pelo menos duas instâncias da imagem em memória RAM. Como tal, uma vez que as máquinas utilizadas dispõem apenas de 8GB, uma imagem com 32K de largura, correspondente a 1 Giga de valores inteiros, ou 4GB de memória, acrescida pelos recursos utilizados pelo sistema operativo e outras variáveis do algoritmo, são suficientes para inutilizar a versão *OpenMP*, tornando-se a solução híbrida na mais eficiente de todas.

V. Variáveis de Teste

Com a finalidade de reduzir erros no processo de medição, estas foram efetuadas recorrendo à mediana dos valores de três testes. As várias variáveis de teste, bem como os motivos da sua escolha são esclarecidas em seguida.

A. Máquinas utilizadas

Primeiramente o nosso objetivo era utlizar as máquinas 652 do SeARCH [2], contudo não foi possível, portanto usamos as máquinas 321, equipadas com o Intel Xeon E5420 [1].

B. Imagens de input

As imagens de input encontram-se no formato *pbm*, no qual apenas estão representados os caracteres 0 (branco) ou 1 (preto). Todas as imagens que usamos nas nossa medições

têm a mesma morfologia, e dimensões que são potências de 2, desde 512×512 até 32768×32768.

A Figura 7 representa a morfologia da imagem utilizada, bem como o resultado produzido pelo programa.

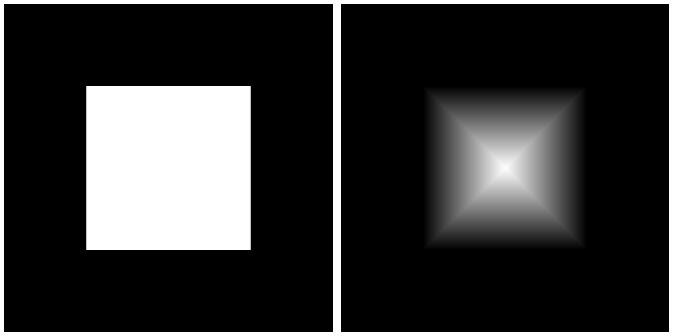


Figura 7. Imagem de input e output correspondente

C. *Número de processos / threads*

Dadas as especificações do *hardware* das máquinas utilizadas, não temos *hyper-threading* nos CPUs que compõem as máquinas 321. Cada nodo tem 2 CPUs cada um tendo 8 cores físicos. Para os nossos testes usamos 1, 2 e 4 máquinas 321, tendo respetivamente um total de 8, 16 e 32 cores.

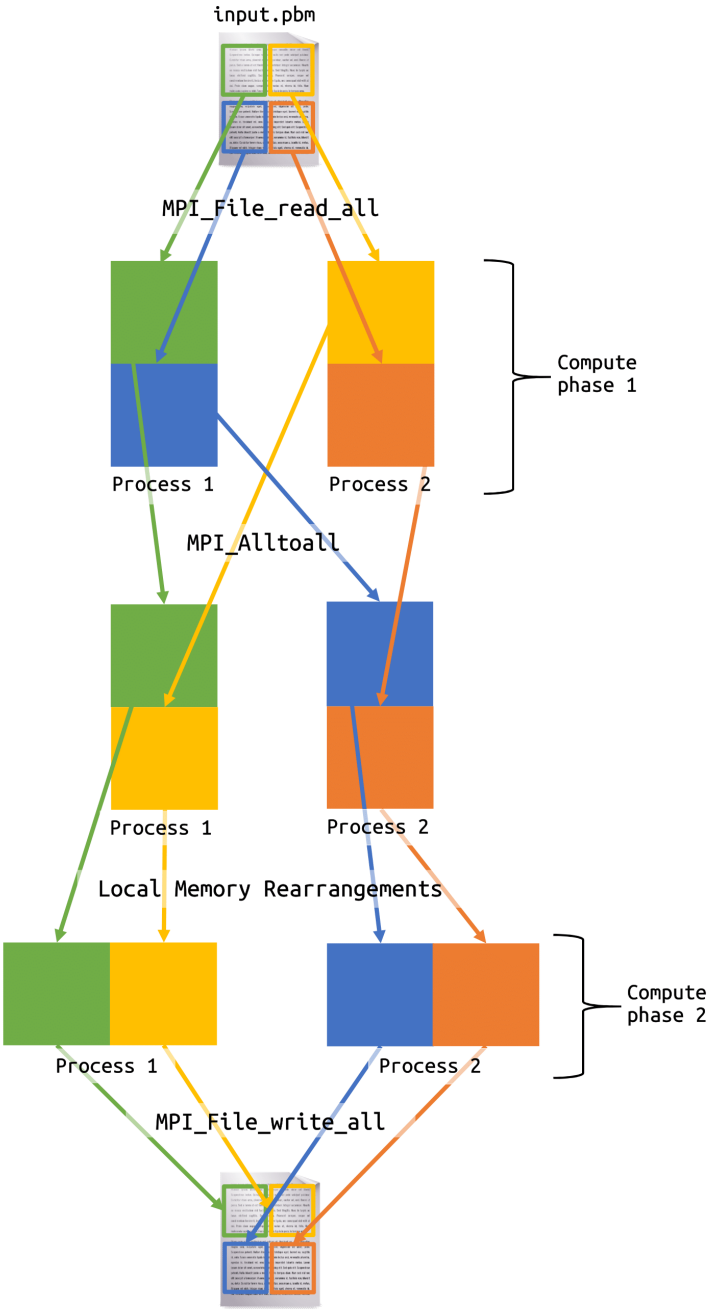
REFERÊNCIAS

[1] Intel® Xeon® Processor E5420. http://ark.intel.com/products/33927/Intel-Xeon-Processor-E5420-12M-Cache-$2_50-GHz-1333-MHz-FSB$.

[2] Search-ON2: Revitalization of HPC infrastructure of UMinho, (NORTE-07-0162-FEDER-000086), co-funded by the North Portugal Regional Operational Programme (ON.2-O Novo Norte), under the National Strategic Reference Framework (NSRF), through the European Regional Development Fund (ERDF).

[3] A. Meijster, J. B. T. M. Roerdink, and W. H. Hesselink. A general algorithm for computing distance transforms in linear time, 2000.

APÊNDICE A
DIAGRAMA ILUSTRATIVO DA IMPLEMENTAÇÃO FINAL DO ALGORITMO



APÊNDICE B
DATASETS

A. *OpenMP*

IMG SIDE	# THREADS	READ	P1	P2	WRITE	TOTAL	ALGORITHM
512	1	0.010568	0.002479	0.006933	0.006761	0.026741	0.009412
	2	0.010732	0.00439	0.003777	0.006254	0.025153	0.008167
	4	0.010083	0.005746	0.001923	0.006626	0.024378	0.007669
	8	0.010412	0.007225	0.000963	0.005838	0.024438	0.008188
1024	1	0.018029	0.009049	0.02765	0.018266	0.072994	0.036699
	2	0.017875	0.011411	0.014367	0.019114	0.062767	0.025778
	4	0.018047	0.012806	0.007334	0.019002	0.057189	0.02014
	8	0.018262	0.014969	0.00367	0.017921	0.054822	0.018639
2048	1	0.043015	0.035065	0.110743	0.063946	0.252769	0.145808
	2	0.043245	0.028681	0.052315	0.065068	0.189309	0.080996
	4	0.043464	0.026422	0.025508	0.065283	0.160677	0.05193
	8	0.043497	0.033562	0.012426	0.060726	0.150211	0.045988
4096	1	0.141694	0.137564	0.390611	0.228521	0.89839	0.528175
	2	0.142025	0.086793	0.18491	0.233054	0.646782	0.271703
	4	0.142925	0.067339	0.093087	0.230983	0.534334	0.160426
	8	0.122825	0.077017	0.046685	0.231022	0.477549	0.123702
8192	1	0.455223	0.438085	1.452191	0.902897	3.248396	1.890276
	2	0.456022	0.261488	0.731099	0.909144	2.357753	0.992587
	4	0.459488	0.195533	0.364824	0.907715	1.92756	0.560357
	8	0.46133	0.2161	0.182645	0.907465	1.76754	0.398745
16384	1	2.592416	1.787245	5.81679	3.635643	13.832094	7.604035
	2	2.00082	1.003596	2.914308	3.625421	9.544145	3.917904
	4	2.21372	0.652471	1.451754	3.621094	7.939039	2.104225
	8	2.112646	0.674529	0.727161	3.626507	7.140843	1.40169

B. MPI

IMG SIDE	# PROC	READ	P1	COMM	P2	WRITE	TOTAL	ALGORITHM
512	1	0.010548	0.000749	0.001644	0.006585	0.007163	0.026689	0.008978
	2	0.02115	0.000306	0.005727	0.002932	0.006481	0.036596	0.008965
	4	0.040045	0.000155	0.005345	0.001303	0.212182	0.25903	0.006803
	8	0.023877	0.000098	0.004196	0.000649	0.252282	0.281102	0.004943
	16	0.135209	0.00005	0.008288	0.000326	0.477765	0.621638	0.008664
	32	0.407693	0.000028	0.020669	0.000163	0.347409	0.775962	0.02086
1024	1	0.019376	0.003067	0.006119	0.021319	0.018918	0.068799	0.030505
	2	0.026405	0.001351	0.015796	0.010592	0.015217	0.069361	0.027739
	4	0.055773	0.000769	0.019509	0.006544	0.028105	0.1107	0.026822
	8	0.040147	0.000391	0.013627	0.003386	0.469113	0.526664	0.017404
	16	0.338023	0.000228	0.25739	0.00131	0.481732	1.078683	0.258928
	32	0.419795	0.0001	0.031125	0.000655	0.540485	0.99216	0.03188
2048	1	0.042142	0.012836	0.021898	0.085553	0.062661	0.22509	0.120287
	2	0.042284	0.006017	0.056751	0.042769	0.041333	0.189154	0.105537
	4	0.11486	0.002505	0.066923	0.020881	0.094138	0.299307	0.090309
	8	0.538223	0.001203	0.069194	0.010393	0.308953	0.927966	0.08079
	16	0.347403	0.000623	0.283987	0.005227	0.557074	1.194314	0.289837
	32	0.639065	0.000393	0.773589	0.002615	0.870628	2.28629	0.776597
4096	1	0.126414	0.052086	0.085999	0.345451	0.233162	0.843112	0.483536
	2	0.104691	0.026086	0.207469	0.173881	0.191449	0.703576	0.407436
	4	0.334709	0.012455	0.230024	0.083675	0.44125	1.102113	0.326154
	8	0.746251	0.006801	0.23854	0.042039	0.489964	1.523595	0.28738
	16	0.619975	0.002752	1.015692	0.020961	0.702888	2.362268	1.039405
	32	0.945248	0.001586	3.869439	0.01046	0.985025	5.811758	3.881485
8192	1	0.467679	0.213853	0.343302	1.39034	0.911391	3.326565	1.947495
	2	0.339274	0.105649	0.966942	0.69749	0.745395	2.85475	1.770081
	4	1.174661	0.050995	0.918603	0.335872	1.24658	3.726711	1.30547
	8	1.521185	0.031759	0.870295	0.167676	1.353153	3.944068	1.06973
	16	1.573609	0.028129	1.243734	0.084234	1.445843	4.375549	1.356097
	32	1.704406	0.027543	4.601975	0.04221	2.093415	8.469549	4.671728
16384	1	1.813302	0.963199	1.363288	5.595002	3.633647	13.368438	7.921489
	2	1.277886	0.428197	3.530744	2.797699	2.407387	10.441913	6.75664
	4	4.51721	0.205915	3.256764	1.348464	4.962464	14.290817	4.811143
	8	4.160283	0.131622	2.728262	0.678218	4.845827	12.544212	3.538102
	16	4.404341	0.11653	3.576186	0.338476	4.688221	13.123754	4.031192
	32	4.472146	0.117819	6.356776	0.168894	4.862788	15.978423	6.643489

C. Híbrida (com compressão)

IMG SIDE	#PROC	#THREADS	READ	P1	COMUNICATE	P2	WRITE	TOTAL	ALGORITHM
512	1	1	0.010666	0.002482	0.004791	0.007857	0.007	0.032796	0.01513
		2	0.010507	0.004316	0.004919	0.003985	0.006498	0.030225	0.01322
		4	0.010673	0.005328	0.005414	0.002011	0.006339	0.029765	0.012753
		8	0.010708	0.00695	0.004979	0.000984	0.006245	0.029866	0.012913
	2	1	0.010881	0.001511	0.004688	0.00396	0.007158	0.028198	0.010159
		2	0.0106	0.003529	0.004745	0.002029	0.006541	0.027444	0.010303
		4	0.010623	0.005219	0.004988	0.001045	0.006314	0.028189	0.011252
		8	0.010736	0.006832	0.004592	0.000549	0.006437	0.029146	0.011973
	4	1	0.020194	0.001105	0.004236	0.002142	0.00997	0.037647	0.007483
		2	0.020394	0.00317	0.003769	0.001244	0.009956	0.038533	0.008183
		4	0.020071	0.005067	0.004132	0.000841	0.010033	0.040144	0.01004
		8	0.020393	0.006592	0.004097	0.000576	0.009932	0.04159	0.011265
1024	1	1	0.019366	0.008953	0.018834	0.031444	0.020025	0.098622	0.059231
		2	0.019042	0.010961	0.019391	0.01539	0.01871	0.083494	0.045742
		4	0.019283	0.012833	0.017898	0.007799	0.018321	0.076134	0.03853
		8	0.019345	0.015003	0.017835	0.003766	0.017369	0.073318	0.036604
	2	1	0.017724	0.004952	0.017729	0.015507	0.017303	0.073215	0.038188
		2	0.01752	0.008483	0.016018	0.007674	0.016405	0.0661	0.032175
		4	0.016127	0.011384	0.015142	0.003996	0.01543	0.062079	0.030522
		8	0.016241	0.01423	0.014466	0.001954	0.016755	0.063646	0.03065
	4	1	0.03557	0.003192	0.251153	0.006434	0.027737	0.324086	0.260779
		2	0.035535	0.007155	0.014387	0.004131	0.025685	0.086893	0.025673
		4	0.036504	0.010436	0.012744	0.00228	0.025621	0.087585	0.02546
		8	0.03602	0.013614	0.01213	0.001214	0.02507	0.088048	0.026958
2048	1	1	0.038432	0.027853	0.069779	0.099674	0.061974	0.297712	0.197306
		2	0.037878	0.025249	0.071979	0.051058	0.060293	0.246457	0.148286
		4	0.038496	0.027608	0.069799	0.028847	0.059813	0.224563	0.126254
		8	0.038479	0.033007	0.07022	0.014004	0.059852	0.215562	0.117231
	2	1	0.030138	0.014962	0.066345	0.049987	0.052001	0.213433	0.131294
		2	0.030154	0.019983	0.055674	0.026732	0.050705	0.183248	0.102389
		4	0.030602	0.024776	0.053515	0.015455	0.050878	0.175226	0.093746
		8	0.030711	0.030109	0.055464	0.007483	0.050796	0.174563	0.093056
	4	1	0.097202	0.010232	0.053602	0.031637	0.086774	0.279447	0.095471
		2	0.095883	0.017003	0.042393	0.015506	0.085693	0.256478	0.074902
		4	0.096666	0.022905	0.040051	0.00798	0.083926	0.251528	0.070936
		8	0.097307	0.029149	0.039031	0.003983	0.085673	0.255143	0.072163
4096	1	1	0.123086	0.109714	0.279829	0.40397	0.232884	1.149483	0.793513
		2	0.121978	0.076474	0.28223	0.203571	0.230338	0.914591	0.562275
		4	0.123078	0.067612	0.270545	0.102049	0.229437	0.792721	0.440206
		8	0.122863	0.07795	0.270504	0.05134	0.229025	0.751682	0.399794
	2	1	0.092546	0.056493	0.250886	0.202229	0.189812	0.791966	0.509608
		2	0.091002	0.05513	0.197479	0.102188	0.188894	0.634693	0.354797
		4	0.09156	0.053781	0.195629	0.054642	0.187658	0.58327	0.304052
		8	0.091452	0.064371	0.196734	0.026317	0.187635	0.566509	0.287422
	4	1	0.322347	0.037161	0.196143	0.125107	0.3233	1.004058	0.358411
		2	0.316667	0.037042	0.154448	0.052588	0.311986	0.872731	0.244078
		4	0.317323	0.048852	0.142743	0.029765	0.31182	0.850503	0.22136
		8	0.317819	0.061016	0.151099	0.01444	0.322285	0.866659	0.226555
8192	1	1	0.454704	0.430148	1.120797	1.620834	0.908772	4.535255	3.171779
		2	0.455459	0.255727	1.122116	0.809227	0.902533	3.545062	2.18707
		4	0.45938	0.204205	1.068095	0.405895	0.899351	3.036926	1.678195
		8	0.459367	0.215452	1.068082	0.203275	0.909017	2.855193	1.486809
	2	1	0.324906	0.220945	0.965332	0.812308	0.74263	3.066121	1.998585
		2	0.323815	0.151425	0.776264	0.406912	0.738537	2.396953	1.334601
		4	0.322271	0.135465	0.738204	0.206679	0.736207	2.138826	1.080348
		8	0.322966	0.155425	0.746601	0.102042	0.736265	2.063299	1.004068
	4	1	1.156854	0.114957	0.703208	0.414025	1.22475	3.613794	1.23219
		2	1.157812	0.098032	0.582346	0.21046	1.214574	3.263224	0.890838
		4	1.158833	0.104218	0.554329	0.104471	1.204426	3.126277	0.763018
		8	1.158867	0.133974	0.537325	0.052503	1.207771	3.09044	0.723802
16384	1	1	1.811814	1.789076	5.478682	6.496299	3.618557	19.194428	13.764057
		2	1.782329	1.015747	4.379145	3.249098	3.598293	14.024612	8.64399
		4	1.790964	0.648606	4.277364	1.621424	3.585421	11.923779	6.547394
		8	1.794058	0.672736	5.214484	0.81202	3.621043	12.114341	6.69924
	2	1	1.267154	0.865761	3.839111	3.257798	2.945754	12.175578	7.96267
		2	1.256227	0.52338	3.052556	1.627701	2.937552	9.397416	5.203637
		4	1.26205	0.394105	2.949591	0.812964	2.944075	8.362785	4.15666
		8	1.260905	0.4273	2.912701	0.406765	2.936222	7.943893	3.746766
	4	1	4.508241	0.448449	2.720273	1.664537	4.876965	14.218465	4.833259
		2	4.694622	0.302175	2.232006	0.832497	4.853853	12.915153	3.366678
		4	4.505095	0.265093	2.034744	0.414633	4.838833	12.058398	2.71447
		8	4.501273	0.315303	2.047258	0.207973	4.836688	11.908495	2.570534
32768	1	1							
		2			Dataset maior que RAM da máquina				
		4							
		8							
	2	1	16.632971	3.579348	17.917342	12.989346	11.791124	62.910131	34.486036
		2	4.974491	2.042136	13.281669	6.51124	11.778733	38.588269	21.835045
		4	4.948364	1.313934	12.04837	3.244448	11.751579	33.306695	16.606752
		8	4.950403	1.362566	11.82318	1.624967	11.726861	31.487977	14.810713
	4	1	18.124667	1.743486	10.828475	6.655541	17.326468	54.678637	19.227502
		2	18.048543	1.049234	8.852342	3.321083	17.469127	48.740329	13.222659
		4	18.205786	0.785087	8.071162	1.753306	17.403438	46.218779	10.609555
		8	17.958457	0.894975	8.239141	0.829614	17.331521	45.253708	9.96373

D. Híbrida (sem compressão)

IMG SIDE	# PROC	#THREADS	READ	P1	COMUNICATE	P2	WRITE	TOTAL	ALGORITHM
512	1	1	0.010694	0.002473	0.00101	0.00786	0.006974	0.029011	0.011343
		2	0.010217	0.004333	0.001197	0.004063	0.006397	0.026207	0.009593
		4	0.010116	0.005531	0.001261	0.002053	0.006389	0.02535	0.008845
		8	0.010483	0.007127	0.001205	0.001038	0.006046	0.025899	0.00937
	2	1	0.010756	0.00151	0.005365	0.003966	0.006844	0.028441	0.010841
		2	0.01063	0.003347	0.005598	0.00206	0.006527	0.028162	0.011005
		4	0.010796	0.005103	0.006239	0.001119	0.0067	0.029957	0.012461
		8	0.01053	0.00678	0.005043	0.000521	0.006418	0.029292	0.012344
	4	1	0.020336	0.001216	0.005827	0.002213	0.010605	0.040197	0.009256
		2	0.020277	0.003118	0.006284	0.001185	0.010527	0.041391	0.010587
		4	0.020214	0.00511	0.006491	0.000647	0.010215	0.042677	0.012248
		8	0.020323	0.006743	0.006135	0.000518	0.010064	0.043783	0.013396
1024	1	1	0.019165	0.008896	0.004896	0.031231	0.019587	0.083775	0.045023
		2	0.019083	0.010932	0.004144	0.01568	0.018668	0.068507	0.030756
		4	0.019139	0.012943	0.003981	0.008003	0.018428	0.062494	0.024927
		8	0.019008	0.015073	0.003773	0.003859	0.017387	0.0591	0.022705
	2	1	0.017067	0.004956	0.016263	0.015593	0.016537	0.070416	0.036812
		2	0.01742	0.008684	0.016518	0.007866	0.016111	0.066599	0.033068
		4	0.017475	0.011375	0.017069	0.00413	0.015951	0.066	0.032574
		8	0.017514	0.014404	0.016399	0.001984	0.016927	0.067228	0.032787
	4	1	0.037025	0.003233	0.021199	0.007949	0.028476	0.097882	0.032381
		2	0.037088	0.007094	0.021631	0.004286	0.027361	0.09746	0.033011
		4	0.037502	0.01069	0.023378	0.002354	0.026986	0.10091	0.036422
		8	0.036799	0.013316	0.019921	0.001253	0.026219	0.097508	0.03449
2048	1	1	0.044183	0.028144	0.020351	0.101093	0.061475	0.255246	0.149588
		2	0.038114	0.025181	0.01991	0.051025	0.059925	0.194155	0.096116
		4	0.038113	0.028446	0.016516	0.027654	0.059379	0.170108	0.072616
		8	0.038136	0.032802	0.016469	0.013955	0.061648	0.16301	0.063226
	2	1	0.029846	0.015055	0.059478	0.050063	0.052564	0.207006	0.124596
		2	0.029934	0.020384	0.054658	0.029472	0.05104	0.185488	0.104514
		4	0.029949	0.024679	0.057622	0.015629	0.050621	0.1785	0.09793
		8	0.030304	0.029831	0.053509	0.007588	0.05028	0.171512	0.090928
	4	1	0.096974	0.010223	0.069239	0.031468	0.084073	0.291977	0.11093
		2	0.095613	0.016931	0.06397	0.015768	0.083825	0.276107	0.096669
		4	0.095945	0.023098	0.061726	0.008138	0.081895	0.270802	0.092962
		8	0.097275	0.02781	0.065479	0.004256	0.081788	0.276608	0.097545
4096	1	1	0.122201	0.109566	0.081002	0.404152	0.229667	0.946588	0.59472
		2	0.121548	0.076581	0.080491	0.203578	0.230411	0.712609	0.36065
		4	0.122796	0.069935	0.067509	0.107217	0.234942	0.602399	0.244661
		8	0.123049	0.076968	0.067545	0.051302	0.231564	0.550428	0.195815
	2	1	0.090167	0.05612	0.192899	0.202198	0.188724	0.730108	0.451217
		2	0.09058	0.055079	0.194864	0.101997	0.187602	0.630122	0.35194
		4	0.090901	0.054416	0.183929	0.054124	0.188713	0.572083	0.292469
		8	0.091353	0.066418	0.189276	0.026138	0.187096	0.560281	0.281832
	4	1	0.316481	0.030564	0.314889	0.105564	0.439151	1.206649	0.451017
		2	0.31904	0.039144	0.280417	0.058135	0.318716	1.015452	0.377696
		4	0.317033	0.048723	0.266935	0.028452	0.314415	0.975558	0.34411
		8	0.318004	0.061097	0.29033	0.014507	0.314229	0.998167	0.365934
8192	1	1	0.458745	0.431063	0.324398	1.620544	0.9059	3.74065	2.376005
		2	0.45549	0.260561	0.326018	0.813082	0.900621	2.755772	1.399661
		4	0.459671	0.193714	0.27413	0.405417	0.895117	2.228049	0.873261
		8	0.459681	0.211425	0.273237	0.202946	0.895709	2.042998	0.687608
	2	1	0.323823	0.221709	0.743833	0.810769	0.741303	2.841437	1.776311
		2	0.320982	0.171385	0.742679	0.406896	0.740112	2.382054	1.32096
		4	0.32307	0.13473	0.713869	0.20322	0.735905	2.110794	1.051819
		8	0.323187	0.15904	0.716322	0.101934	0.736132	2.036615	0.977296
	4	1	1.15823	0.11508	1.016513	0.414061	1.209078	3.912962	1.545654
		2	1.156217	0.096261	0.978137	0.207761	1.205683	3.644059	1.282159
		4	1.158507	0.103239	0.993866	0.104211	1.19882	3.558643	1.201316
		8	1.160074	0.137036	0.946238	0.05252	1.207221	3.503089	1.135794
16384	1	1	1.954628	1.792281	2.517014	6.506146	3.619246	16.389315	10.815441
		2	1.794593	1.014771	1.58791	3.256127	3.597462	11.250863	5.858808
		4	1.788595	0.657634	1.354939	1.625593	3.588507	9.015268	3.638166
		8	1.780213	0.671385	1.130978	0.811745	3.567967	7.962288	2.614108
	2	1	1.259233	0.865412	2.932987	3.259736	2.945262	11.26263	7.058135
		2	1.255372	0.524294	2.932497	1.631053	2.94193	9.285146	5.087844
		4	1.260592	0.406303	2.831797	0.814926	2.928982	8.2426	4.053026
		8	1.260454	0.438638	2.816245	0.406781	2.928474	7.850592	3.661664
	4	1	4.502845	0.448797	3.158365	1.665876	4.874191	14.650074	5.273038
		2	4.50355	0.302526	3.437592	0.833002	4.842348	13.919018	4.57312
		4	4.501924	0.263882	3.262786	0.414339	4.8324	13.275331	3.941007
		8	4.513226	0.323087	3.065523	0.207972	4.833251	12.943059	3.596582
32768	1	1							
		2			Dataset maior que RAM da máquina				
		4							
		8							
	2	1	13.68641	3.581971	16.872294	13.035313	11.797118	58.973106	33.489578
		2	5.132838	2.036014	17.749048	6.556558	11.783615	43.258073	26.34162
		4	5.007148	1.326467	14.260224	3.249336	11.796424	35.639599	18.836027
		8	5.10168	1.357804	14.993377	1.626655	11.764538	34.844054	17.977836
	4	1	18.093953	1.743574	12.029096	6.677492	17.389584	55.933699	20.450162
		2	18.023075	1.049694	11.7391	3.321728	17.291258	51.424855	16.110522
		4	17.971103	0.79264	11.964694	1.65745	17.336144	49.722031	14.414784
		8	17.9686	0.902047	12.500246	0.829527	17.344495	49.544915	14.23182

APÊNDICE C
MAPEAMENTO DE PROCESSOS

A. Por core

```
[a71874@compute-321-1 ~]$ mpirun -np 4 --map-by core --report-bindings executable
[compute-321-1.local:11414] MCW rank 0 bound to socket 0[core 0[hwt 0]]: [B/./.][././.]
[compute-321-1.local:11414] MCW rank 1 bound to socket 0[core 1[hwt 0]]: [./B/.][././.]
[compute-321-1.local:11414] MCW rank 2 bound to socket 0[core 2[hwt 0]]: [././B][././.]
[compute-321-1.local:11414] MCW rank 3 bound to socket 0[core 3[hwt 0]]: [././.][B/./.]

[a71874@compute-321-1 ~]$ mpirun -np 8 --map-by core --report-bindings executable
[compute-321-1.local:11055] MCW rank 0 bound to socket 0[core 0[hwt 0]]: [B/./.][././.]
[compute-321-1.local:11055] MCW rank 1 bound to socket 0[core 1[hwt 0]]: [./B/.][././.]
[compute-321-1.local:11055] MCW rank 2 bound to socket 0[core 2[hwt 0]]: [././B][././.]
[compute-321-1.local:11055] MCW rank 3 bound to socket 0[core 3[hwt 0]]: [././.][B/./.]
[compute-321-1.local:11055] MCW rank 4 bound to socket 1[core 4[hwt 0]]: [././.][B/./.]
[compute-321-1.local:11055] MCW rank 5 bound to socket 1[core 5[hwt 0]]: [././.][./B/.]
[compute-321-1.local:11055] MCW rank 6 bound to socket 1[core 6[hwt 0]]: [././.][././B]
[compute-321-1.local:11055] MCW rank 7 bound to socket 1[core 7[hwt 0]]: [././.][././B]

[a71874@compute-321-1 ~]$ mpirun -np 16 --map-by core --report-bindings executable
[compute-321-1.local:11081] MCW rank 0 bound to socket 0[core 0[hwt 0]]: [B/./.][././.]
[compute-321-1.local:11081] MCW rank 1 bound to socket 0[core 1[hwt 0]]: [./B/.][././.]
[compute-321-1.local:11081] MCW rank 2 bound to socket 0[core 2[hwt 0]]: [././B][././.]
[compute-321-1.local:11081] MCW rank 3 bound to socket 0[core 3[hwt 0]]: [././.][B/./.]
[compute-321-1.local:11081] MCW rank 4 bound to socket 1[core 4[hwt 0]]: [././.][B/./.]
[compute-321-1.local:11081] MCW rank 5 bound to socket 1[core 5[hwt 0]]: [././.][./B/.]
[compute-321-1.local:11081] MCW rank 6 bound to socket 1[core 6[hwt 0]]: [././.][././B]
[compute-321-1.local:11081] MCW rank 7 bound to socket 1[core 7[hwt 0]]: [././.][././B]
[compute-321-6.local:19798] MCW rank 8 bound to socket 0[core 0[hwt 0]]: [B/./.][././.]
[compute-321-6.local:19798] MCW rank 9 bound to socket 0[core 1[hwt 0]]: [./B/.][././.]
[compute-321-6.local:19798] MCW rank 10 bound to socket 0[core 2[hwt 0]]: [././B][././.]
[compute-321-6.local:19798] MCW rank 11 bound to socket 0[core 3[hwt 0]]: [././.][B/./.]
[compute-321-6.local:19798] MCW rank 12 bound to socket 1[core 4[hwt 0]]: [././.][B/./.]
[compute-321-6.local:19798] MCW rank 13 bound to socket 1[core 5[hwt 0]]: [././.][./B/.]
[compute-321-6.local:19798] MCW rank 14 bound to socket 1[core 6[hwt 0]]: [././.][././B]
[compute-321-6.local:19798] MCW rank 15 bound to socket 1[core 7[hwt 0]]: [././.][././B]
```

B. Por socket

```
[a71874@compute-321-1 ~]$ mpirun -np 4 --map-by socket --report-bindings executable
[compute-321-1.local:11440] MCW rank 0 bound to socket 0[core 0[hwt 0]]: [B/./.][././.]
[compute-321-1.local:11440] MCW rank 1 bound to socket 1[core 4[hwt 0]]: [././.][B/./.]
[compute-321-1.local:11440] MCW rank 2 bound to socket 0[core 1[hwt 0]]: [./B/.][././.]
[compute-321-1.local:11440] MCW rank 3 bound to socket 1[core 5[hwt 0]]: [././.][./B/.]

[a71874@compute-321-1 ~]$ mpirun -np 8 --map-by socket --report-bindings executable
[compute-321-1.local:11150] MCW rank 0 bound to socket 0[core 0[hwt 0]]: [B/./.][././.]
[compute-321-1.local:11150] MCW rank 1 bound to socket 1[core 4[hwt 0]]: [././.][B/./.]
[compute-321-1.local:11150] MCW rank 2 bound to socket 0[core 1[hwt 0]]: [./B/.][././.]
[compute-321-1.local:11150] MCW rank 3 bound to socket 1[core 5[hwt 0]]: [././.][./B/.]
[compute-321-1.local:11150] MCW rank 4 bound to socket 0[core 2[hwt 0]]: [././B][././.]
[compute-321-1.local:11150] MCW rank 5 bound to socket 1[core 6[hwt 0]]: [././.][././B]
[compute-321-1.local:11150] MCW rank 6 bound to socket 0[core 3[hwt 0]]: [././.][B/./.]
[compute-321-1.local:11150] MCW rank 7 bound to socket 1[core 7[hwt 0]]: [././.][././B]
```

C. Por nodo

```
[a71874@compute-321-1 ~]$ mpirun -np 4 --map-by node --report-bindings executable
[compute-321-1.local:11429] MCW rank 0 bound to socket 0[core 0[hwt 0]]: [B/./.][././.]
[compute-321-1.local:11429] MCW rank 2 bound to socket 0[core 1[hwt 0]]: [./B/.][././.]
[compute-321-6.local:19961] MCW rank 1 bound to socket 0[core 0[hwt 0]]: [B/./.][././.]
[compute-321-6.local:19961] MCW rank 3 bound to socket 0[core 1[hwt 0]]: [./B/.][././.]
```

```
[a71874@compute-321-1 ~]$ mpirun -np 8 --map-by node --report-bindings executable
[compute-321-1.local:11326] MCW rank 0 bound to socket 0[core 0[hwt 0]]: [B/./.][././.]
[compute-321-1.local:11326] MCW rank 2 bound to socket 0[core 1[hwt 0]]: [./B/.][././.]
[compute-321-1.local:11326] MCW rank 4 bound to socket 0[core 2[hwt 0]]: [./.B/.][././.]
[compute-321-1.local:11326] MCW rank 6 bound to socket 0[core 3[hwt 0]]: [././.B][././.]
[compute-321-6.local:19903] MCW rank 1 bound to socket 0[core 0[hwt 0]]: [B/./.][././.]
[compute-321-6.local:19903] MCW rank 3 bound to socket 0[core 1[hwt 0]]: [./B/.][././.]
[compute-321-6.local:19903] MCW rank 5 bound to socket 0[core 2[hwt 0]]: [./.B/.][././.]
[compute-321-6.local:19903] MCW rank 7 bound to socket 0[core 3[hwt 0]]: [././.B][././.]
```

APÊNDICE D
TABELA DE ESPECIFICAÇÕES DOS CPUS INSTALADOS NAS MÁQUINAS USADAS

A. *Intel Xeon Processor E5420 (presente nas máquinas 321)*

General information

Type	CPU / Microprocessor
Market segment	Server
Family	Intel Xeon 5400
Model number	**E5420**
CPU part numbers	• **EU80574KJ060N** is an OEM/tray microprocessor • **AT80574KJ060N** is an OEM/tray microprocessor in Halide free package • **BX80574E5420A** is a boxed microprocessor • **BX80574E5420P** is a boxed microprocessor
Frequency	**2500 MHz**
Bus speed	**1333 MHz**
Clock multiplier	**7.5**
Package	771-land Flip-Chip Land Grid Array (FC-LGA) 1.48" x 1.48" (3.75 cm x 3.75 cm)
Socket	**Socket 771 / LGA771**
Introduction date	Nov 12, 2007
Price at introduction	$316

S-spec numbers

Part number	ES/QS processors		Production processors	
	Q9DB	QFUP	SLANV	SLBBL
AT80574KJ060N		+		+
BX80574E5420A			+	+
BX80574E5420P			+	+
EU80574KJ060N	+		+	

Architecture / Microarchitecture

Microarchitecture	Core
Platform	Bensley Bensley-VS Cranberry Lake Stoakley
Processor core	Harpertown
Core steppings	C0 (SLANV) E0 (QFUP, SLBBL)
CPUIDs	10676 (SLANV) 1067A (QFUP, SLBBL)
Manufacturing process	0.045 micron 820 million transistors
Die size	214mm^2
Data width	**64 bit**
The number of CPU cores	**4**
The number of threads	**4**
Floating Point Unit	Integrated
Level 1 cache size	4 x 32 KB 8-way set associative instruction caches 4 x 32 KB 8-way set associative write-back data caches

Level 2 cache size	**2 x 6 MB 24-way set associative shared caches**
Multiprocessing	Up to 2 processors
Features	• MMX instructions • SSE / Streaming SIMD Extensions • SSE2 / Streaming SIMD Extensions 2 • SSE3 / Streaming SIMD Extensions 3 • SSSE3 / Supplemental Streaming SIMD Extensions 3 • SSE4.1 / Streaming SIMD Extensions 4.1 • EM64T / Extended Memory 64 technology / Intel 64 • NX / XD / Execute disable bit • VT-x / Virtualization technology
Low power features	• HALT state • Extended HALT state • Stop Grant state • Enhanced SpeedStep technology

Integrated peripherals / components

Integrated graphics	None

Electrical / Thermal parameters

V core	0.85V - 1.35V
Minimum/Maximum operating temperature	5°C - 67°C
Maximum power dissipation	133.25 Watt 108.5 Watt (sustained)
Thermal Design Power	**80 Watt**

Notes on Intel Xeon E5420

- Bus frequency is 333 MHz. Because the processor uses Quad Data Rate bus the effective bus speed is 1333 MHz
- Part AT80574KJ060N is manufactured in halogen-free package
- Part BX80574E5420A includes 3U+ active / 1U passive thermal solution
- Part BX80574E5420P includes 2U passive thermal solution