Universidade do Minho

Mestrado em Engenharia Informática

Engenharia dos Sistemas de Computação - 2014/2015

NAS Parallel Benchmarks (NPB)

24 de Março de 2015

Resumo

Este trabalho demonstra o resultado do Benchmark no nó compute-431-6 do SeARCH no Departamento de Informática da Universidade do Minho. Desta forma será possível analisar os recursos disponíveis com os testes intensivos que o NPB providencia.

Fábio Gomes pg27752

Índice

Introdução

O problema que nos foi apresentado consiste em analisar a performance de um sistema de forma a poder saber as capacidades e limites operacionais para as aplicações que irão ser desenvolvidas para aquela infraestrutura.

Foi proposto que os testes incidissem em diferentes classes com diferentes compiladores, como o *icc* e *gcc* (versão 4.4.6 e 4.9.0), número de processos (*MPI*) e threads (*OpenMP*) e ainda para as versões SERIAL, OpenMP e MPI para cada Benchmark.

Os dados que o NPB devolve não são suficientes e para tal foi necessário recorrer a ferramentas externas para obter informações extras ao longo do tempo de execução. Especificamente o derivado *dstat* para a alocação de memória, acessos ao disco e dados enviados/recebidos na rede; o *mpstat* para a utilização dos diferentes processadores.

Com estes dados recolhidos foi possível gerar gráficos para comparar as diferentes combinações de compiladores e número de processos e threads.

NAS Parallel Benchmarks (NPB)

São um pequeno conjunto de programas destinados a ajudar a avaliar o desempenho dos supercomputadores paralelos. Os *Benchmarks* são derivadas de aplicações da dinâmica de fluido computacional (CFD) e consistem em cinco *kernels* e três *pseudo-aplicações* (NPB 1). O pacote de Benchmark foi estendido para incluir novos pontos de referência para malha adaptativa não-estruturada, I/O, aplicações multi-zona, e grelhas computacionais paralelas. Os tamanhos dos problemas no NPB são predefinidos e indicados com diferentes classes (A,B,C,D,E,F ou S,W). Implementações de referência de NPB estão disponíveis em modelos de programação mais usadas como MPI e OpenMP (NPB 2 e NPB 3), sendo estas as versões que mais nos interessam.

Há diferentes especificações de Benchmark:

- 5 kernels
  - `IS - Integer Sort, random memory access`
  - `EP - Embarrassingly Parallel`
  - `CG - Conjugate Gradient, irregular memory access and communication`
  - `MG - Multi-Grid on a sequence of meshes, long- and short-distance communication, memory intensive`
  - `FT - discrete 3D fast Fourier Transform, all-to-all communication`
- 3 pseudo-aplicações
  - `BT - Block Tri-diagonal solver`
  - `SP - Scalar Penta-diagonal solver`
  - `LU - Lower-Upper Gauss-Seidel solver`

E classes de tamanho:

- S: small for quick test purposes
- W: workstation size (a 90's workstation; now likely too small)
- A, B, C: standard test problems; ~4X size increase going from one class to the next
- D, E, F: large test problems; ~16X size increase from each of the previous classes

Optei pela *EP*, *FT*, *IS* e *SP* como foco para os testes com classes A e B em cada.

Caracterização do Sistema

Para utilização deste projeto utilizei o nó 431-6 do SeARCH, recorrendo a jobs específicos para ele, com a seguinte especificação:

	Cluster Node
Manufacturer	intel
Model	X5650
Clock speed	2.67 GHz
Architecture	x86-64
#Cores	12
#Threads per Core	2
Total Threads	24

Compilação, Versões e Parâmetros

Para este trabalho tive que utilizar diferentes versões de compiladores e ferramentas, todas com -O.

gcc / mpicc / gfortran

4.4.6 e 4.9.0

Icc / ifort

13.0.1

Para cada kernel de Benchmark foi necessário usar estas 3 versões:

Serial

Versão base em que tipicamente é utilizado um processador.

OpenMP

Utiliza um número especificado de threads, neste caso 2, 8, 16 e 48.

MPI

Com o número providenciado de processos, 2, 4, 8 e 16 (no caso do SP 4, 9 e 16).

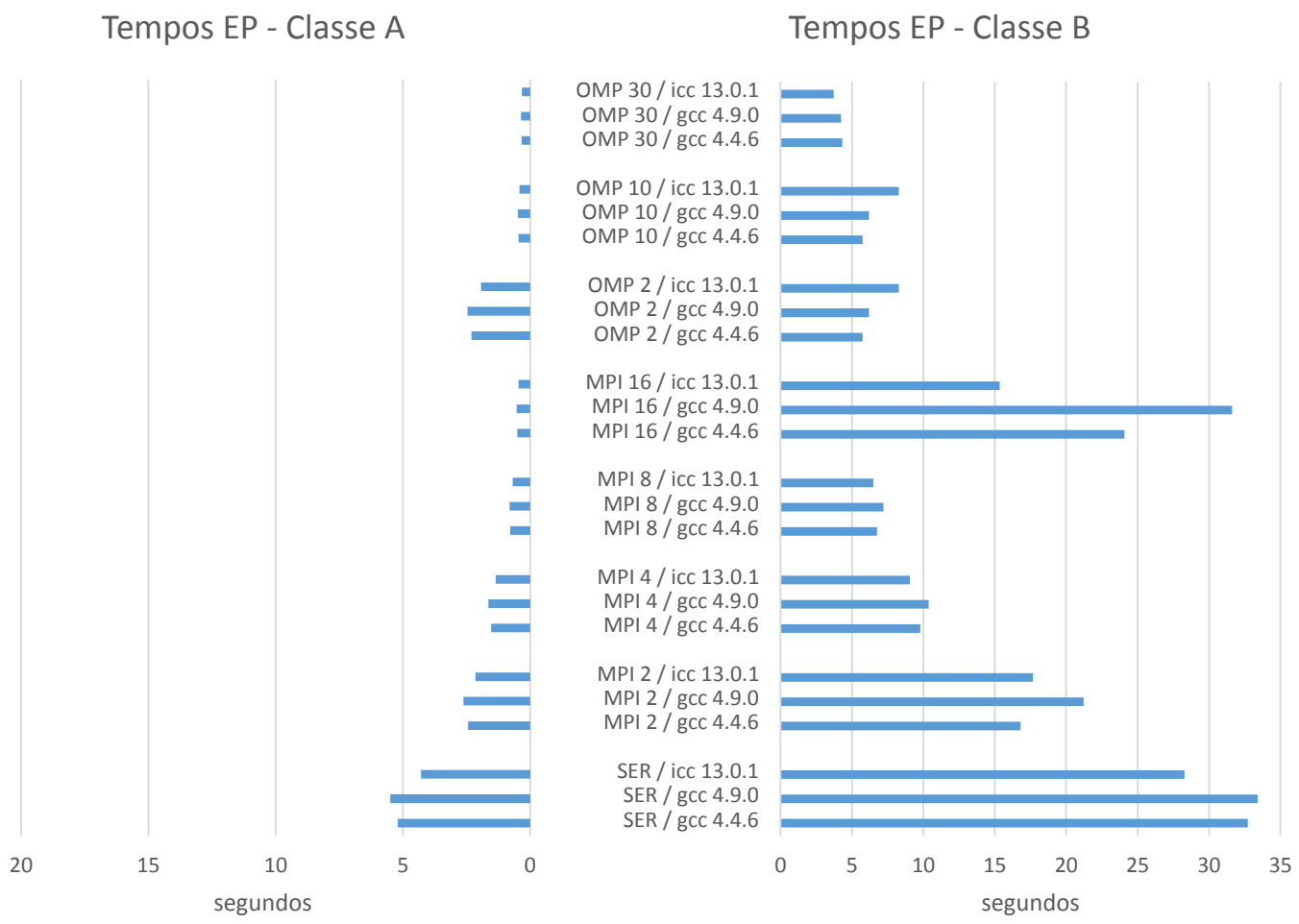
Benchmark EP

Há 2 classes, A e B, que estão representadas. A verde são marcados os tempos melhores entre o mesmo teste comparando com as 3 versões de compiladores e a vermelho o pior.

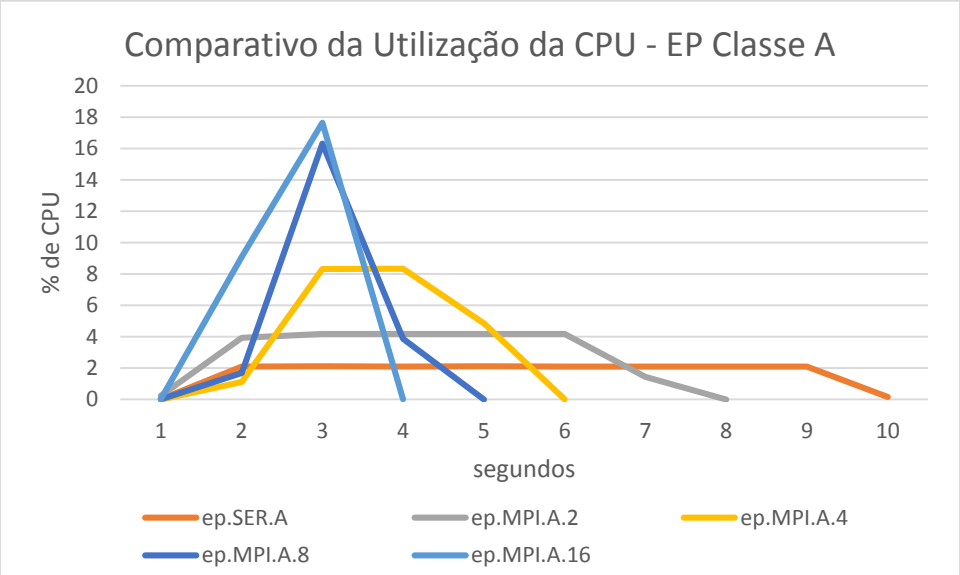
Análise da Duração

A seguinte tabela inclui os tempos de execução e os *Mop/s* (Milhões de Operações por segundo) para cada combinação de Teste.

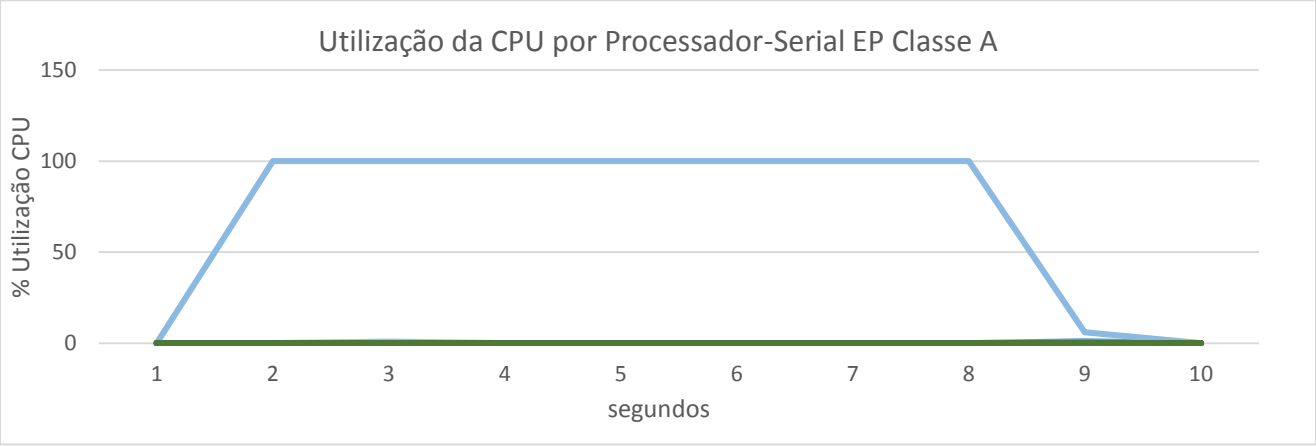
		EP															
		SER		MPI								OpenMP					
				2 Procs		4 Procs		8 Procs		16 Procs		2 Threads		10 Threads		30 Threads	
		s	Mop/s	s	Mop/s	s	Mop/s	s	Mop/s	s	Mop/s	s	Mop/s	s	Mop/s	s	Mop/s
gcc 4.4.6	**A**	19,4	27,7	8,59	62,48	5,36	100,2	2,27	236,6	1,15	467	10	53,45	1,82	194	0,78	690
gcc 4.9.0		20,3	26,5	8,88	60,4	4,47	120,1	2,44	220,25	1,21	442	9,84	54,57	2,01	266	0,81	659
icc 13.0.1		9,36	57,4	4,14	129,8	2,56	209,6	1,19	452,81	0,64	835	4,33	124,1	0,78	687	0,37	1461
gcc 4.4.6	**B**	77,2	27,8	34,3	62,55	17,3	124,5	9,29	231,24	4,59	468	31,8	67,55	7,04	305	2,95	728
gcc 4.9.0		79,3	27,1	34,9	61,46	22,5	95,26	9,8	219,19	5,01	428	35	61,39	6,87	312	3,55	604
icc 13.0.1		37,4	57,4	15,9	135,4	10,2	209,6	5,23	410,62	2,2	975	15,9	135,4	3,05	703	1,41	1520



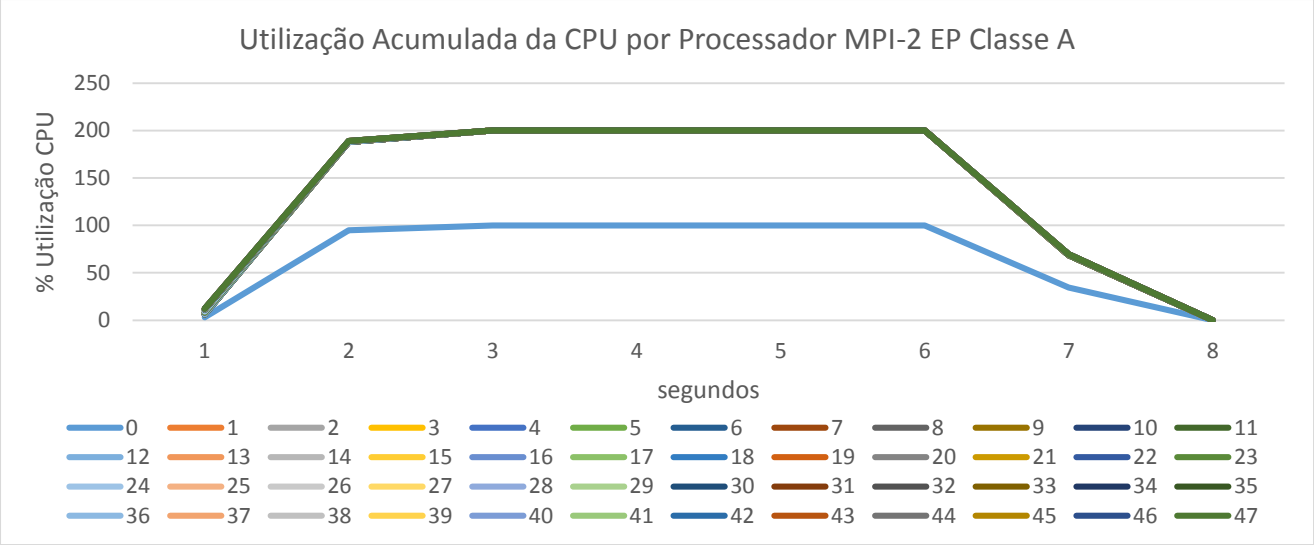
É possível distinguir que o `icc` foi o melhor compilador porque obteve 100 % dos melhores tempos. Sendo assim foi utilizado o `icc` como base dos testes seguintes.



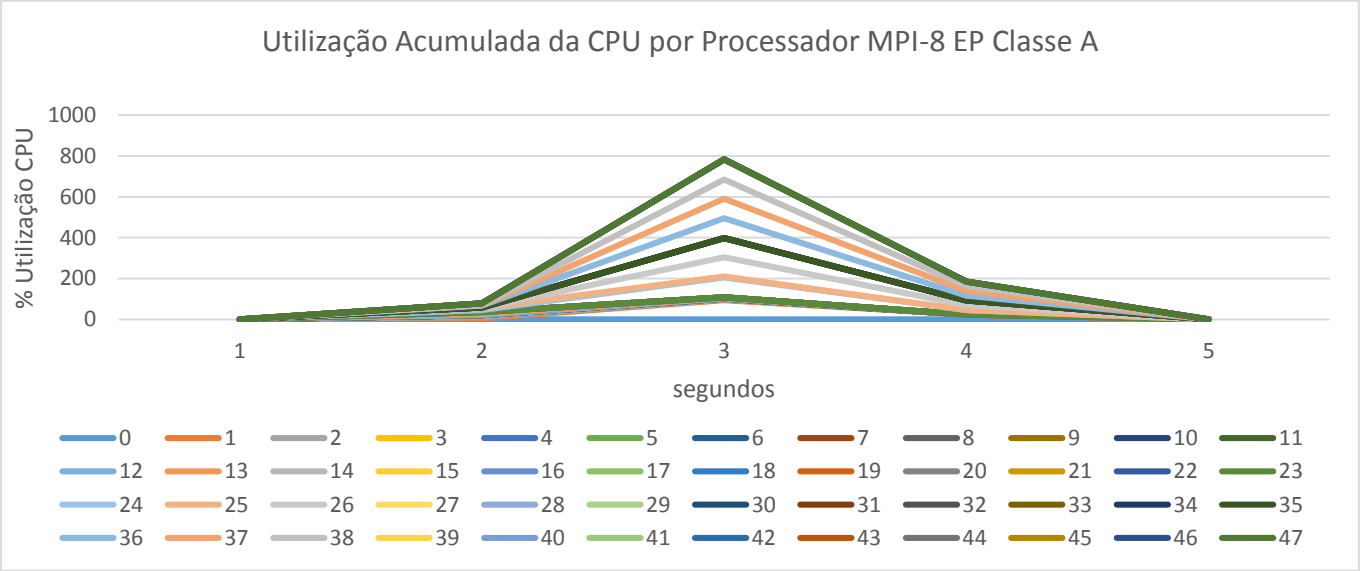
Este gráfico comparativo em que temos os 5 testes para o EP com a Classe A, não é muito interessante pois fica apenas claro quais são as combinações mais rápidas. Assim utilizando a soma acumulada, ou apenas individual, das percentagens de utilização por processador dá uma melhor perceção dos resultados. Visto o `mpstat` ter monitorizado durante os jobs 48 processadores o máximo de performance será `48*100% = 4800%`. Os gráficos seguintes são alguns exemplos.



Como a versão Serial só utiliza um processador a utilização individual será de 100.



Com 2 processos, a utilização acumulada é de 200 % como previsto.

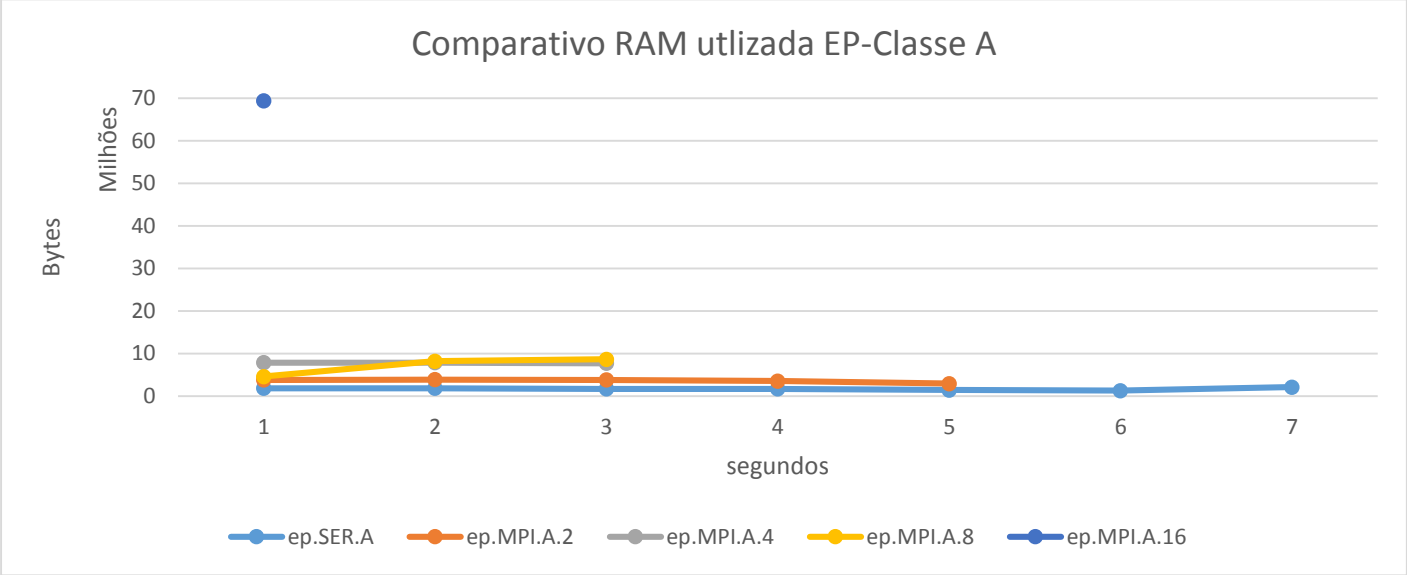


Com 8 processos a utilização bate nos 800 %.

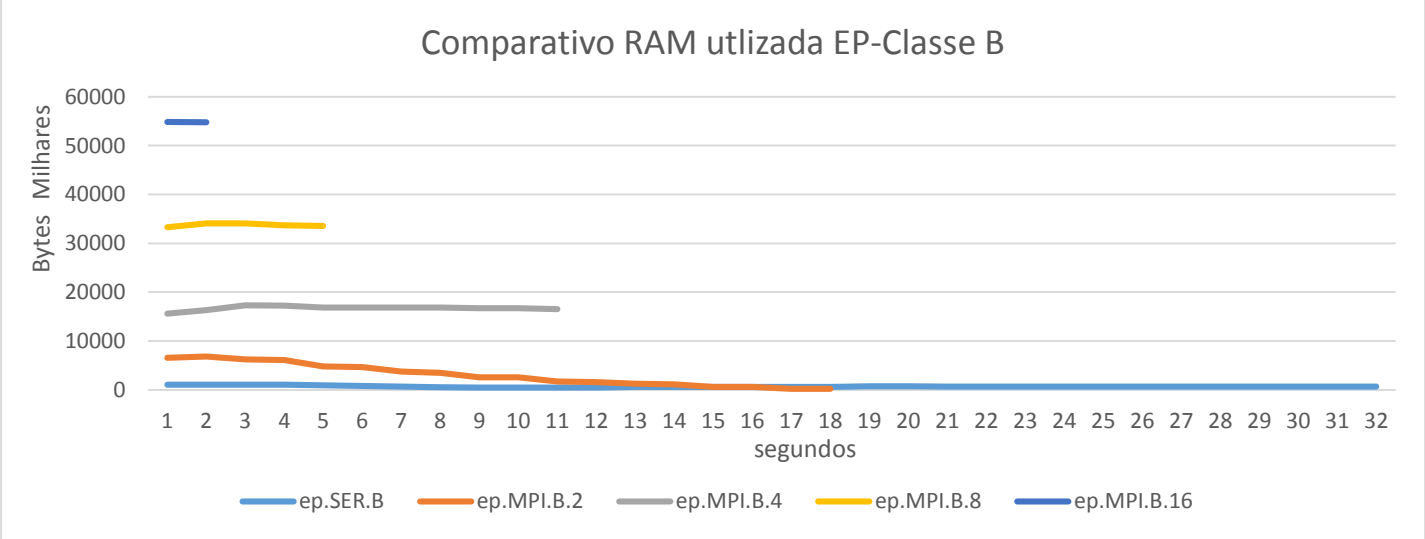
Análise da Memória

Os gráficos seguintes demonstram a alocação de Memória ao longo do tempo de execução.

Classe A:



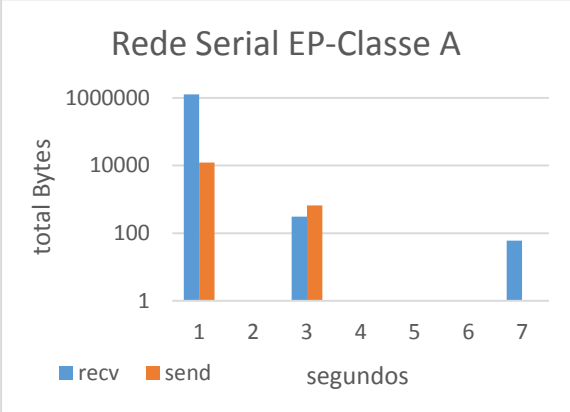
Classe B:

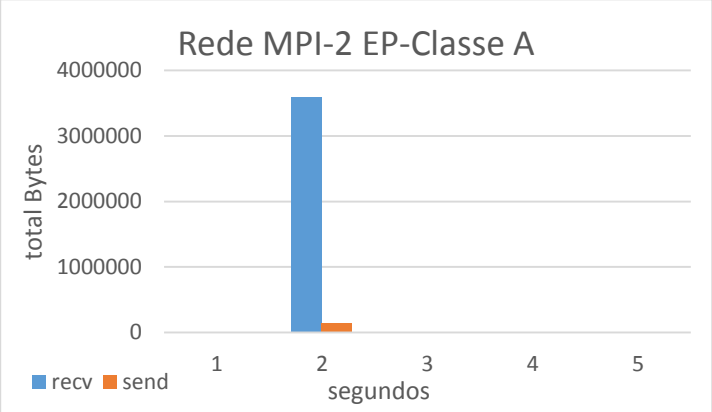


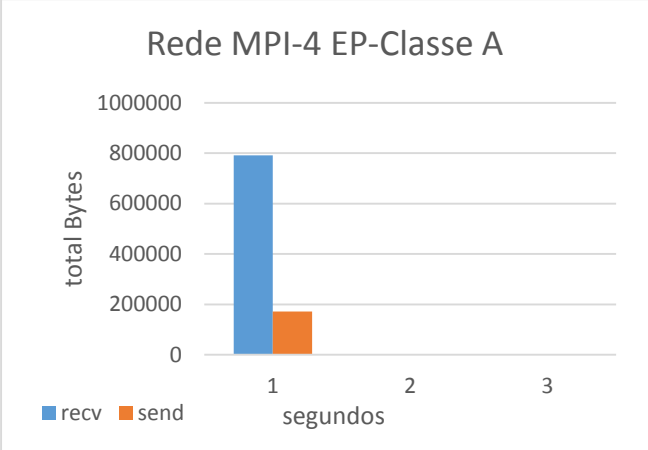
Como expectável a memória necessária para o funcionamento dos programas aumenta com o número de paralelismo. Cabendo a nós, programadores, decidir qual o melhor balanço entre duração e memória disponível.

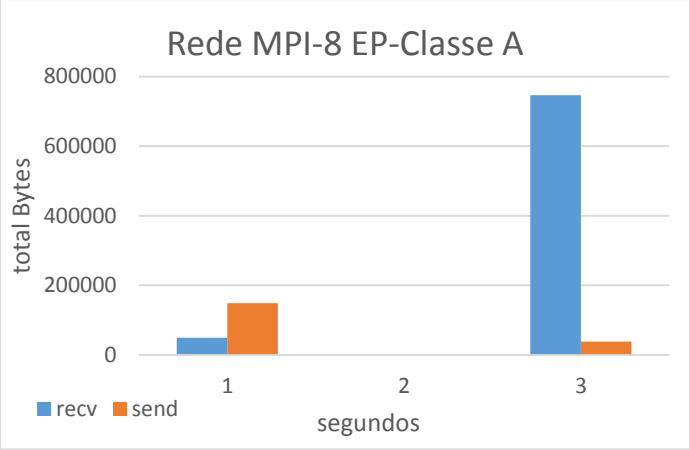
Análise da Rede

Os gráficos são muito diferentes e não é fácil fazer uma comparação entre eles pois também os valores ora são na ordem das centenas de milhar como passam para as centenas como estão a zero. De notar que o gráfico para o Serial da Classe A está com eixo dos Bytes com escala logarítmica de base 10.









Análise de Acessos ao Disco

Os testes que utilizei não são focados na interação com o Disco por isso não faz sentido realizar esta medição para qualquer dos testes seguintes.

Conclusão

O `icc` neste kernel foi o que obteve melhores resultados.

Benchmark FT

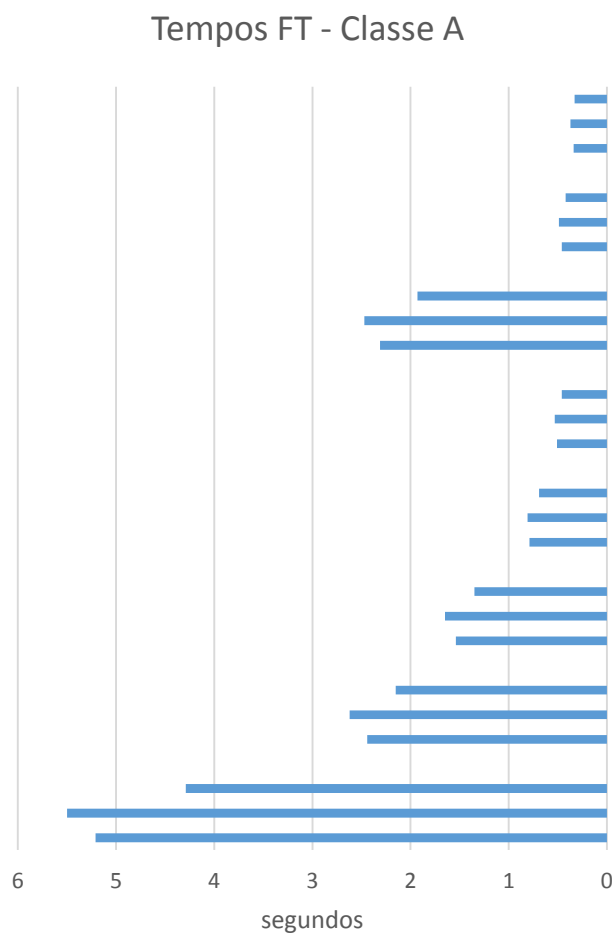
Há 2 classes, A e B, que estão representadas. A verde são marcados os tempos melhores entre o mesmo teste comparando com as 3 versões de compiladores e a vermelho o pior.

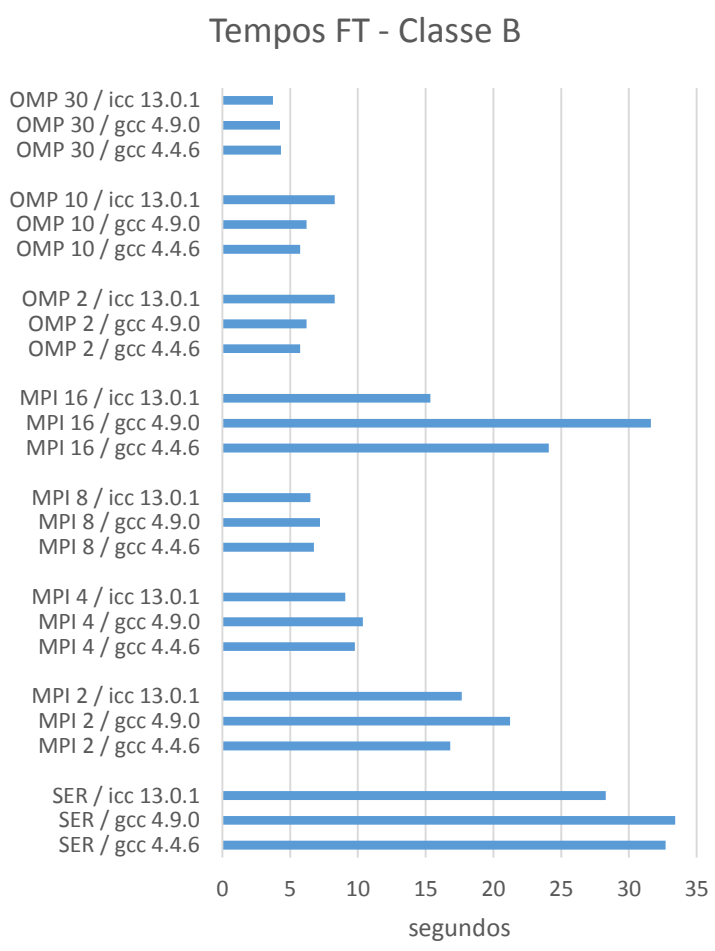
Análise da Duração

A seguinte tabela inclui os tempos de execução e os *Mop/s* (Milhões de Operações por segundo) para cada combinação de Teste.

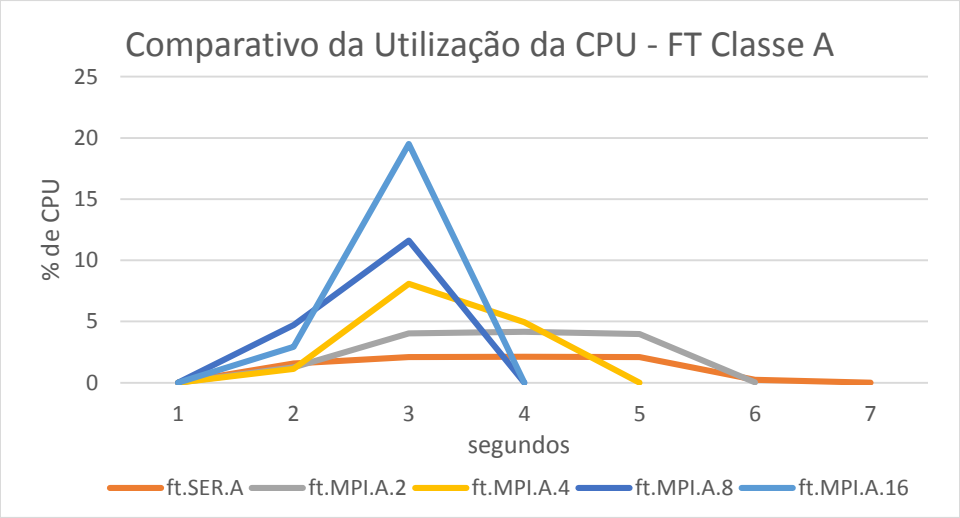
.

		FT															
		SER		MPI								OpenMP					
				2 Procs		4 Procs		8 Procs		16 Procs		2 Threads		10 Threads		30 Threads	
		s	Mop/s	s	Mop/s	s	Mop/s	s	Mop/s	s	Mop/s	s	Mop/s	s	Mop/s	s	Mop/s
gcc 4.4.6	**A**	5,21	1369	2,44	2920	1,54	4619	0,79	9019	0,51	14009	2,31	3086	0,46	15597	0,34	20806
gcc 4.9.0		5,5	1297	2,62	2724	1,65	4320	0,81	8819	0,53	13550	2,47	2888	0,49	14462	0,37	19245
icc 13.0.1		4,29	1663	2,15	3315	1,35	5301	0,69	10403	0,46	15378	1,93	3700	0,42	17144	0,33	21332
gcc 4.4.6	**B**	69,5	1324	32,7	2814	16,8	5476	9,78	9414	6,76	13609	24,1	3821	5,74	16042	4,33	21247
gcc 4.9.0		65,3	1410	33,4	2756	21,2	4337	10,4	8876	7,21	12769	31,6	2911	6,2	14851	4,24	21709
icc 13.0.1		60,3	1528	28,3	3254	17,7	5209	9,06	10161	6,5	14167	15,3	139	8,29	11109	3,72	24746

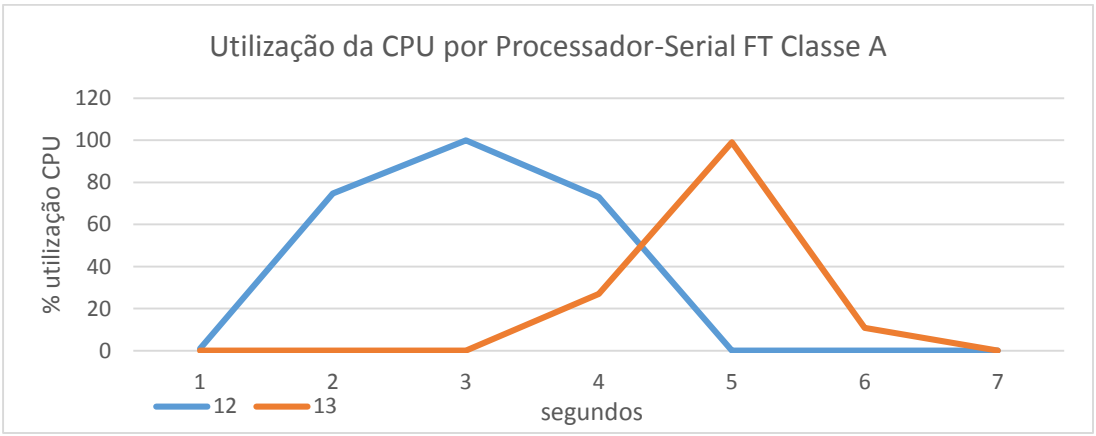




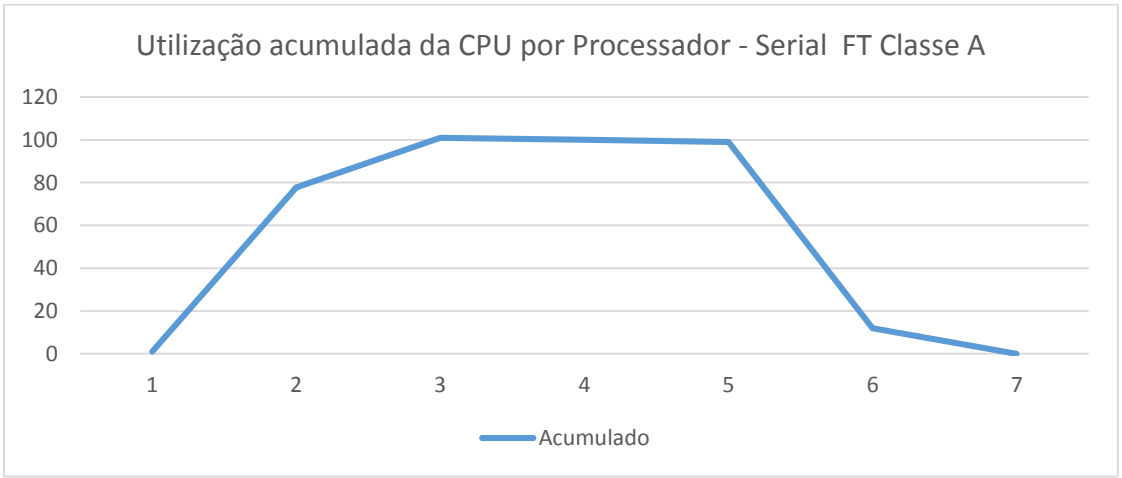
O compilador `icc` obteve uma maioria de melhores tempos, não sendo o melhor em apenas 2 ocasiões.



Este gráfico comparativo anterior apresenta os 5 testes para o FT com a Classe A, sendo possível reparar quais são as combinações mais rápidas. De seguida são apresentados os resultados em gráfico da soma acumulada, ou apenas individual, das percentagens de utilização por processador.



Aconteceu uma situação interessante, o processo trocou de processador durante a execução.

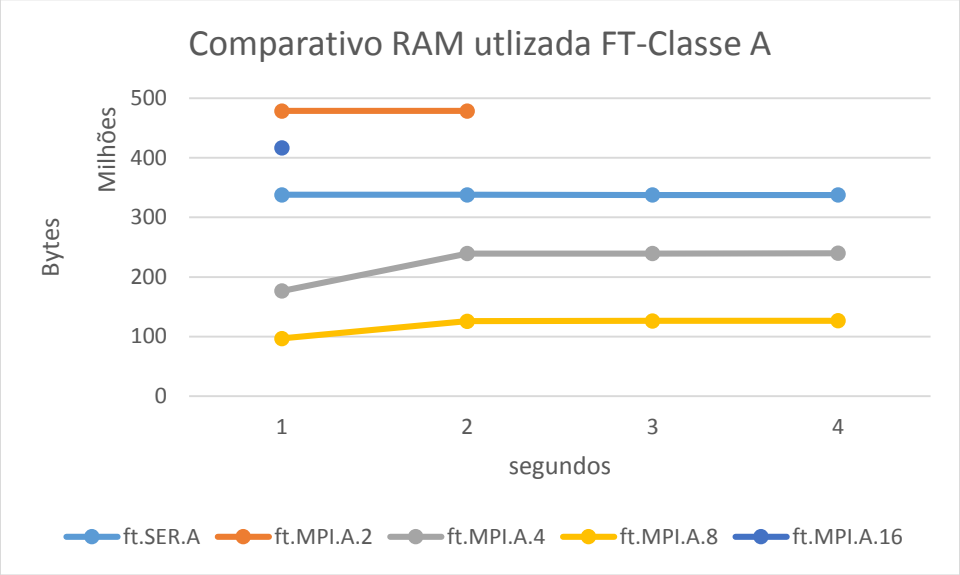


Mas o acumulado permaneceu no máximo dos 100 %.

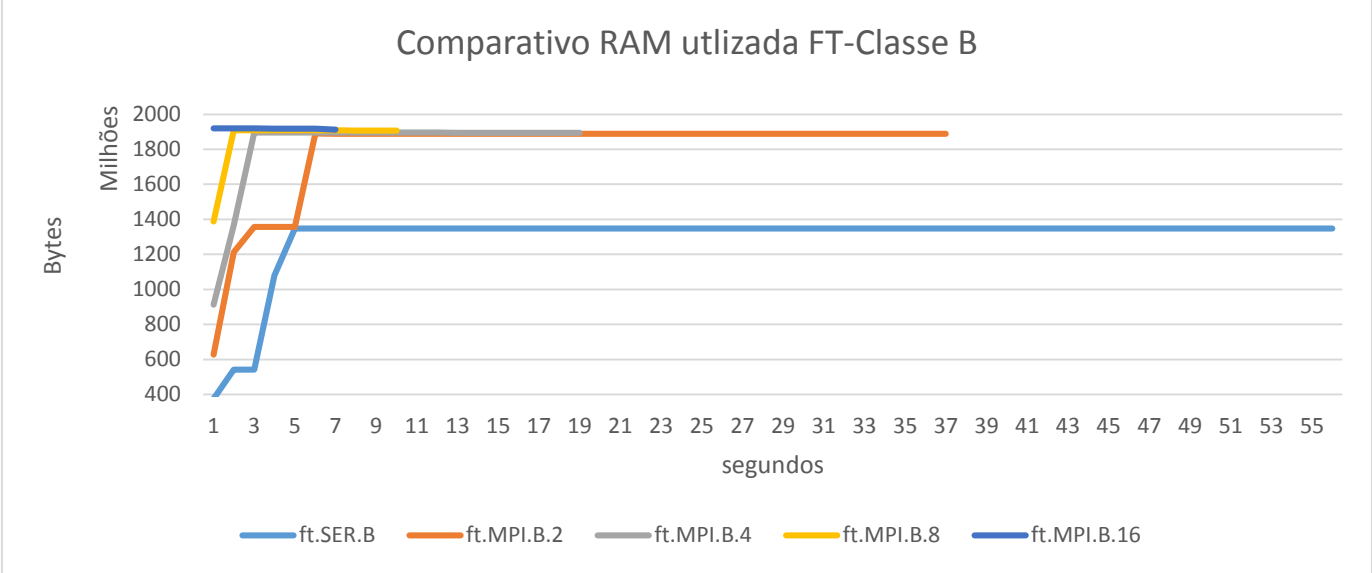
Análise da Memória

Os gráficos seguintes demonstram a alocação de Memória ao longo do tempo de execução.

Classe A:



Classe B:



A memória usando MPI com a Classe B ficou estável para os testes na ordem dos 1800 MBytes.

Conclusão

O `icc` neste kernel foi, de novo, o que obteve melhores resultados.

Benchmark IS

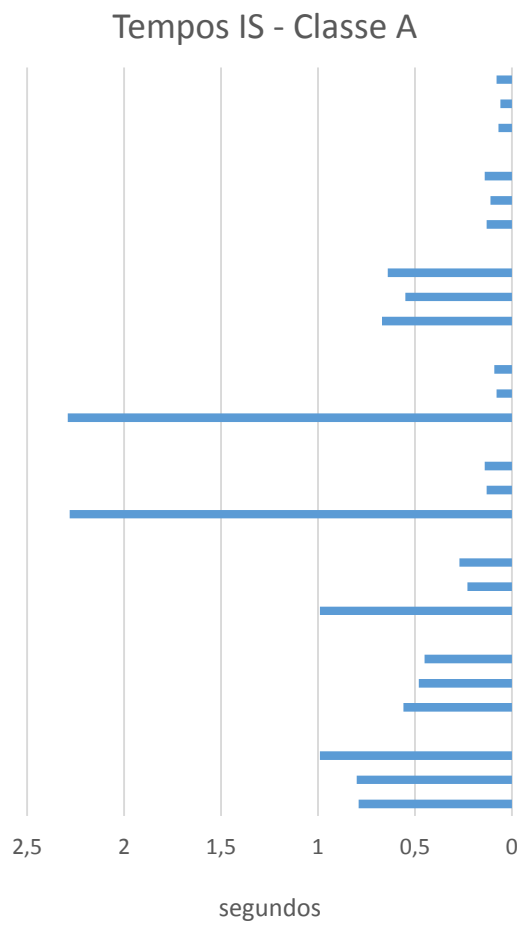
Há 2 classes, A e B, que estão representadas. A verde são marcados os tempos melhores entre o mesmo teste comparando com as 3 versões de compiladores e a vermelho o pior. Este é o teste mais curto.

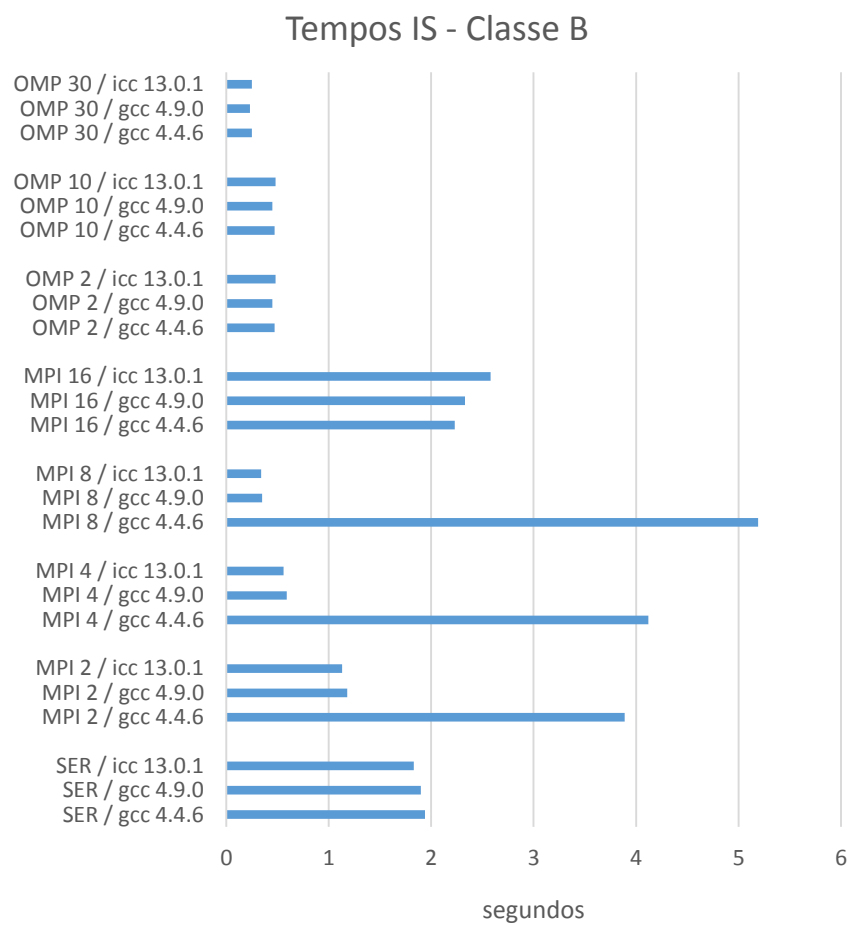
Análise da Duração

A seguinte tabela inclui os tempos de execução e os *Mop/s* (Milhões de Operações por segundo) para cada combinação de Teste.

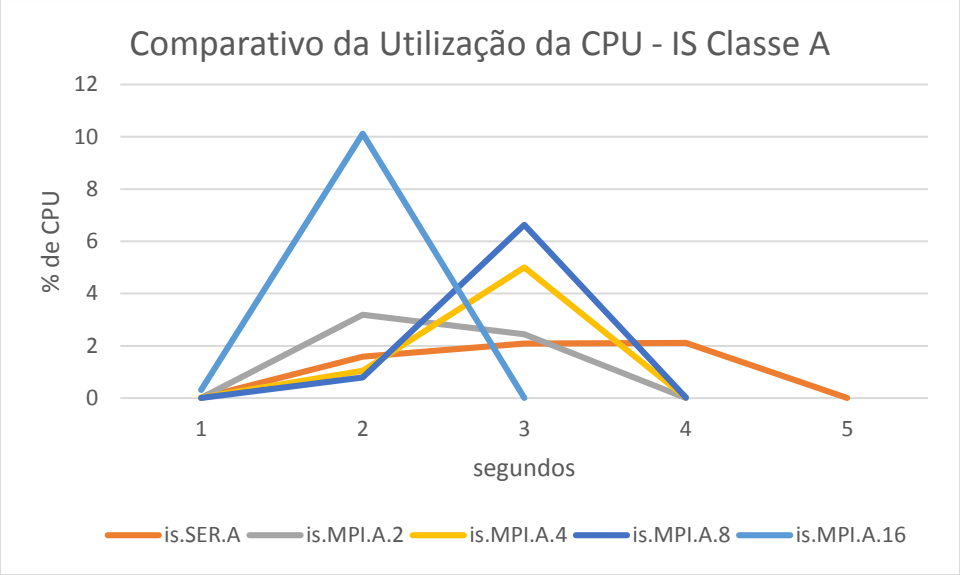
.

		IS															
		SER		MPI								OpenMP					
				2 Procs		4 Procs		8 Procs		16 Procs		2 Threads		10 Threads		30 Threads	
		s	Mop/s	s	Mop/s	s	Mop/s	s	Mop/s	s	Mop/s	s	Mop/s	s	Mop/s	s	Mop/s
gcc 4.4.6	**A**	0,79	107	0,56	148,8	0,99	84,32	2,28	37	2,29	37	0,67	125,9	0,13	660	0,07	1179
gcc 4.9.0		0,8	104	0,48	174,3	0,23	358,9	0,13	648	0,08	1078	0,55	151	0,11	770	0,06	1361
icc 13.0.1		0,99	84,3	0,45	185	0,27	315,5	0,14	590	0,09	998	0,64	131,7	0,14	616	0,08	1111
gcc 4.4.6	**B**	3,3	102	1,94	172,8	3,89	86,21	4,12	81	5,19	65	2,23	150,4	0,47	707	0,25	1341
gcc 4.9.0		3,29	102	1,9	177	1,18	284,1	0,59	570	0,35	955	2,33	143	0,45	743	0,23	1437
icc 13.0.1		4,23	79,4	1,83	183,6	1,13	296	0,56	595	0,34	995	2,58	130	0,48	705	0,25	1329

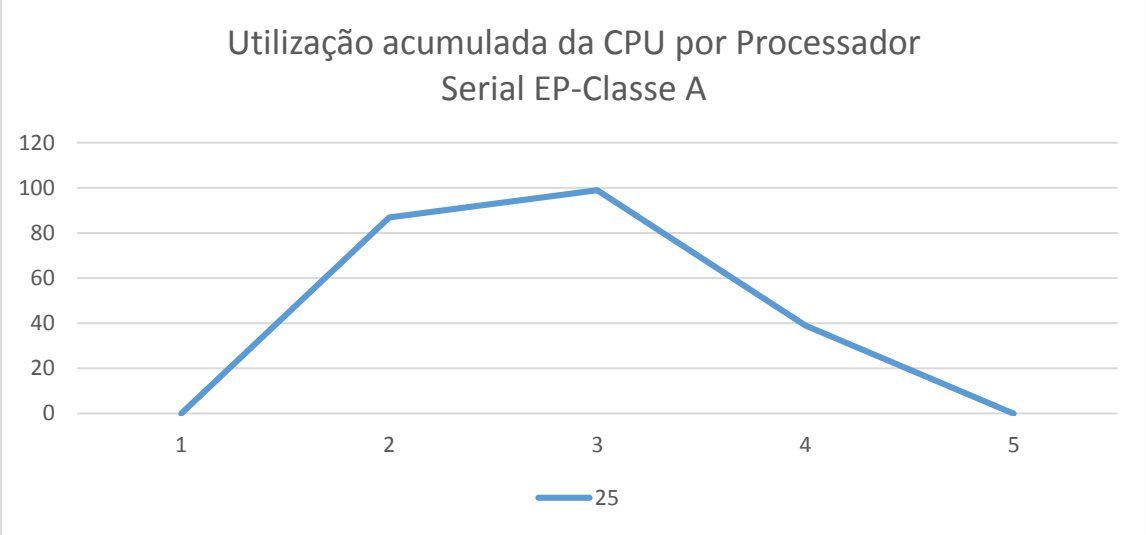




Neste teste o `gcc 4.9.0` foi o que obteve no geral os melhores resultados, batendo o `icc`.



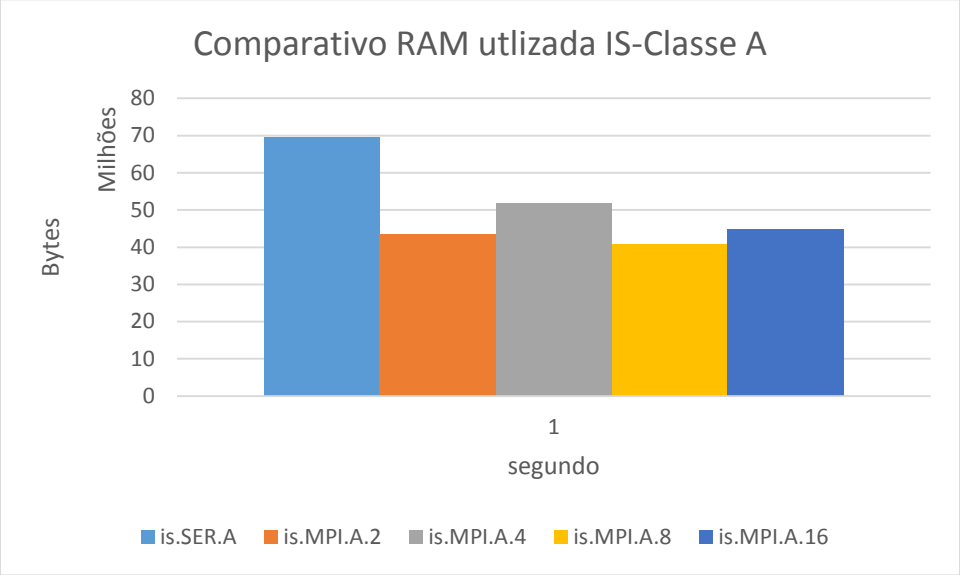
Este gráfico comparativo anterior apresenta os 5 testes para o IS com a Classe A, sendo possível reparar quais são as combinações mais rápidas.



Análise da Memória

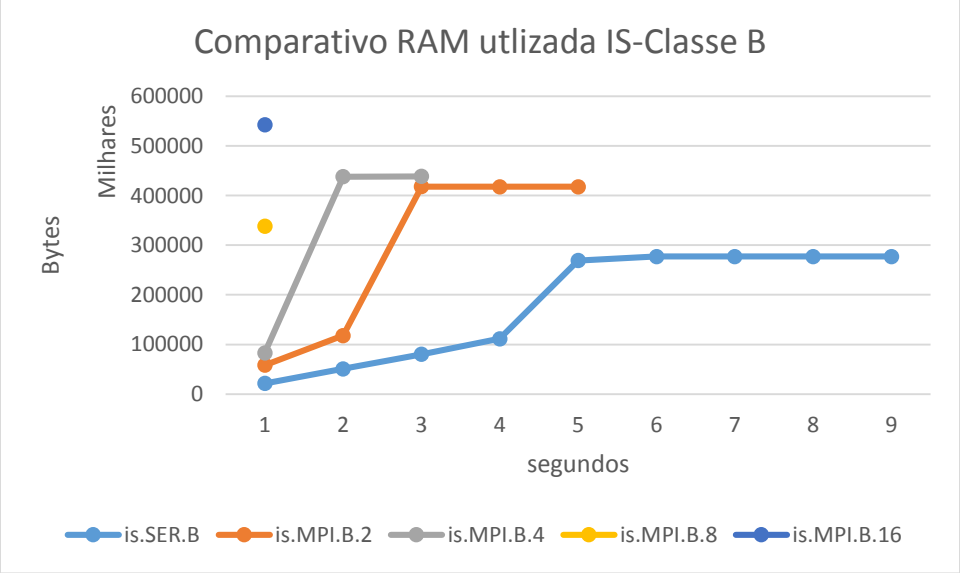
Os gráficos seguintes demonstram a alocação de Memória ao longo do tempo de execução.

Classe A:



Como estes testes normalmente nem 1 segundo duram, realizei vários seguidos e depois uma média.

Classe B:



Esta Classe já proporciona tempos maiores podendo assim gerar um gráfico temporal. Curiosamente com o aumento dos processos a memória utilizada diminuiu.

Conclusão

O `gcc 4.9.0` neste kernel foi o que obteve melhores resultados.

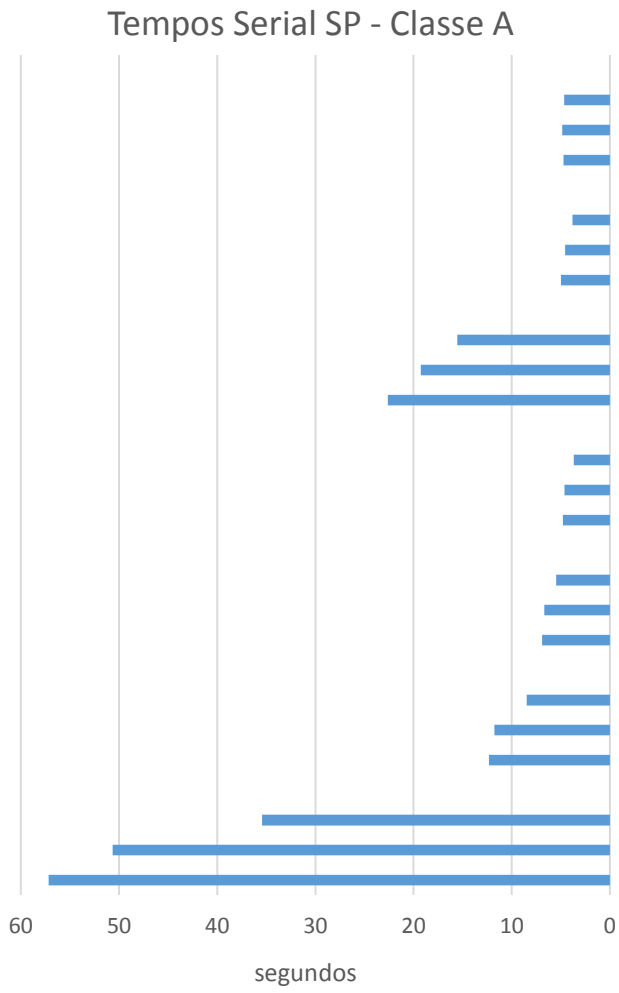
Benchmark SP

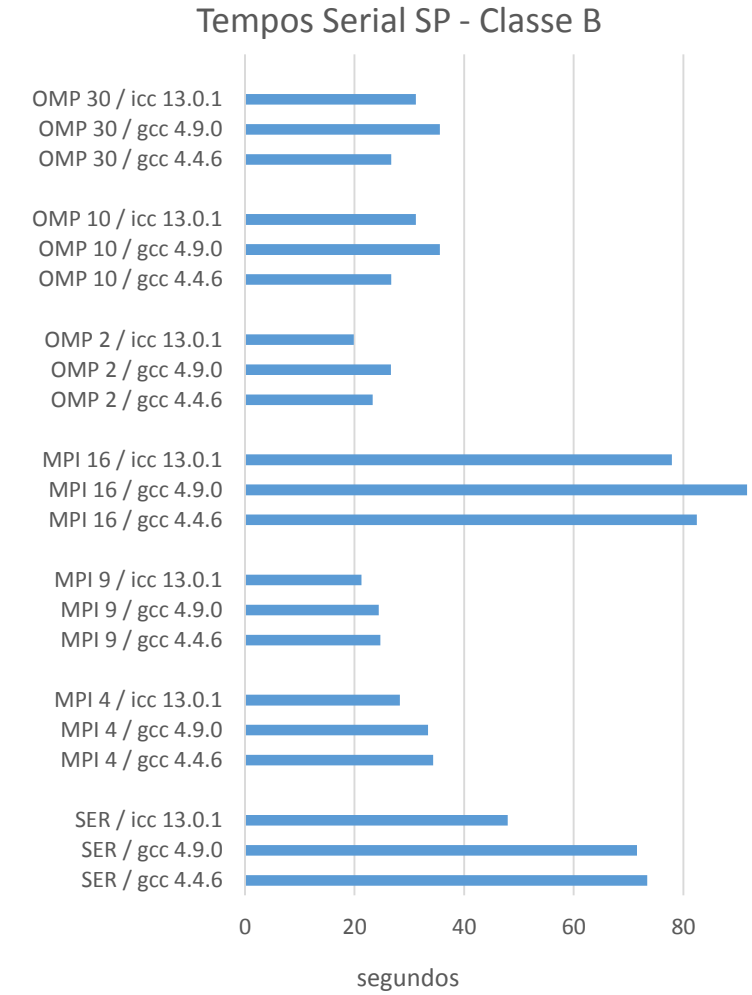
Há 2 classes, A e B, que estão representadas. A verde são marcados os tempos melhores entre o mesmo teste comparando com as 3 versões de compiladores e a vermelho o pior.

Análise da Duração

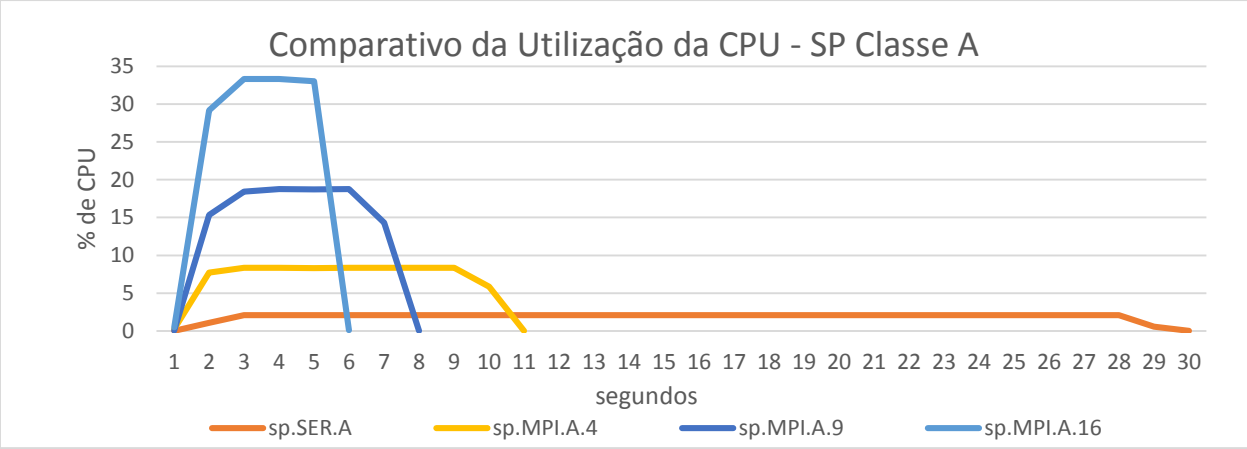
A seguinte tabela inclui os tempos de execução e os *Mop/s* (Milhões de Operações por segundo) para cada combinação de Teste.

		sp													
		SER		MPI						OpenMP					
				4 Procs		9 Procs		16 Procs		2 Threads		10 Threads		30 Threads	
		s	Mop/s	s	Mop/s	s	Mop/s	s	Mop/s	s	Mop/s	s	Mop/s	s	Mop/s
gcc 4.4.6	A	57,2	1486	12,3	6908	6,9	12313	4,78	17800	22,6	3756	4,99	17024	4,74	17918
gcc 4.9.0		50,7	1678	11,8	7223	6,69	12699	4,64	18333	19,3	4415	4,55	18678	4,85	17543
icc 13.0.1		35,5	2398	8,48	10022	5,48	1500	3,67	23179	15,5	5472	3,81	22289	4,66	18230
gcc 4.4.6	B	232	1532	73,4	4836	34,4	10330	24,7	14363	82,5	4304	23,3	15229	26,7	13299
gcc 4.9.0		232	1529	71,5	4963	33,4	10633	24,4	1456	91,7	3872	26,6	13325	35,6	9986
icc 13.0.1		162	2195	47,9	7408	28,3	12566	21,3	16703	78	4554	19,9	17862	31,2	11373

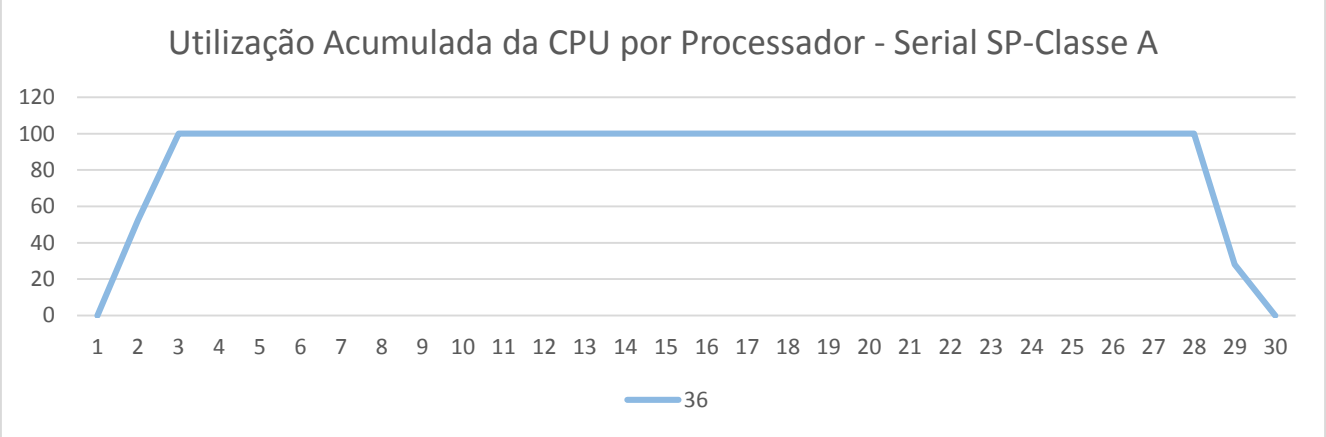




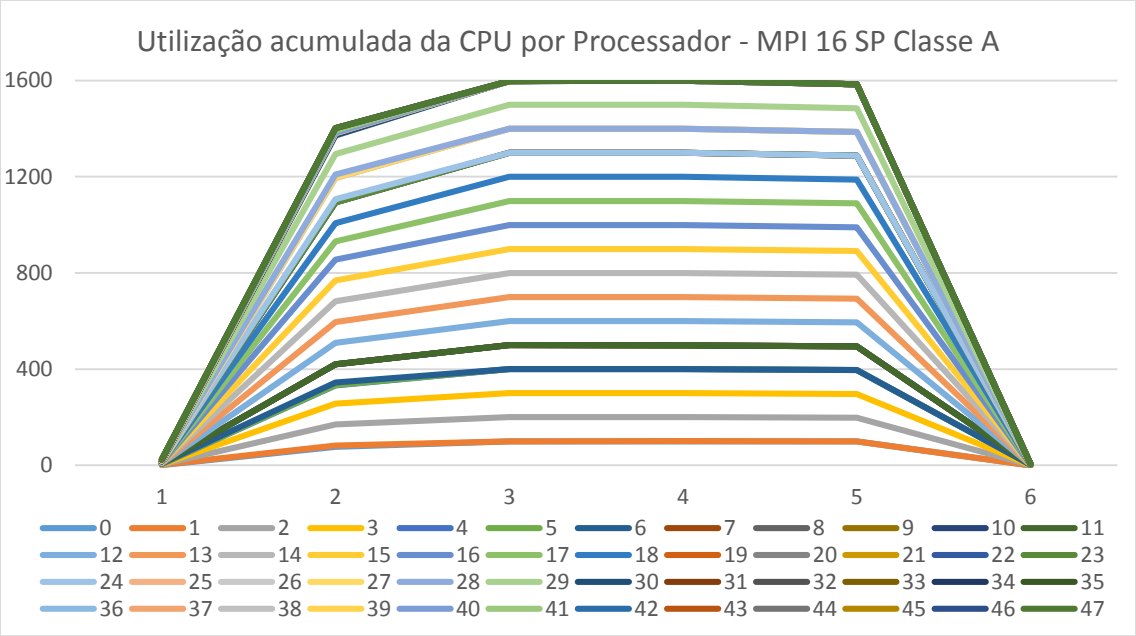
O `icc` para esta sequência de testes foi o compilador que obteve melhores resultados, falhando apenas um.



Com o aumento dos processos a utilização geral da CPU aumenta provocando tempos melhores.



O Processador 36 tomou conta do processo e a sua utilização foi de 100 %.

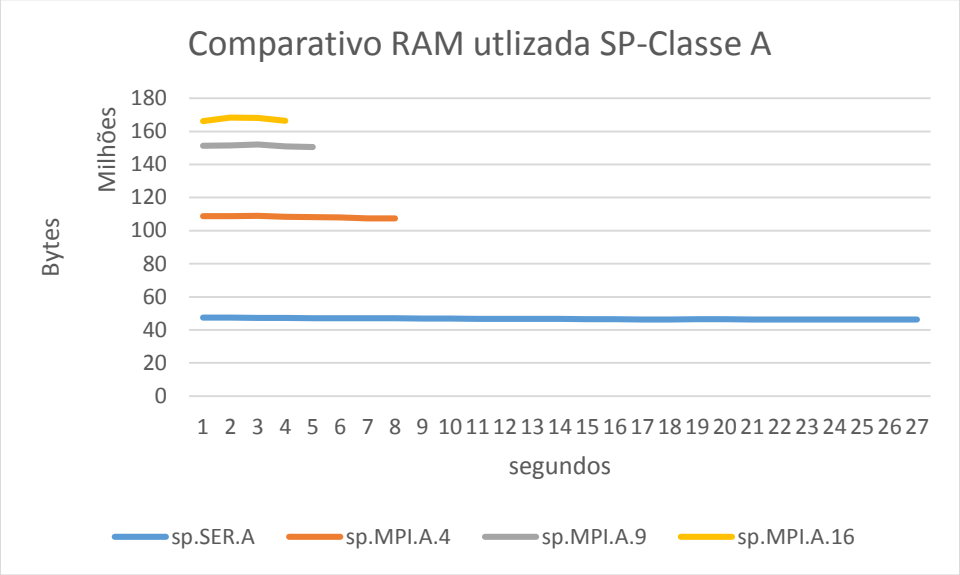


Com 16 processos para o MPI a soma de todas as contribuições chega ao previsto de 1600 %.

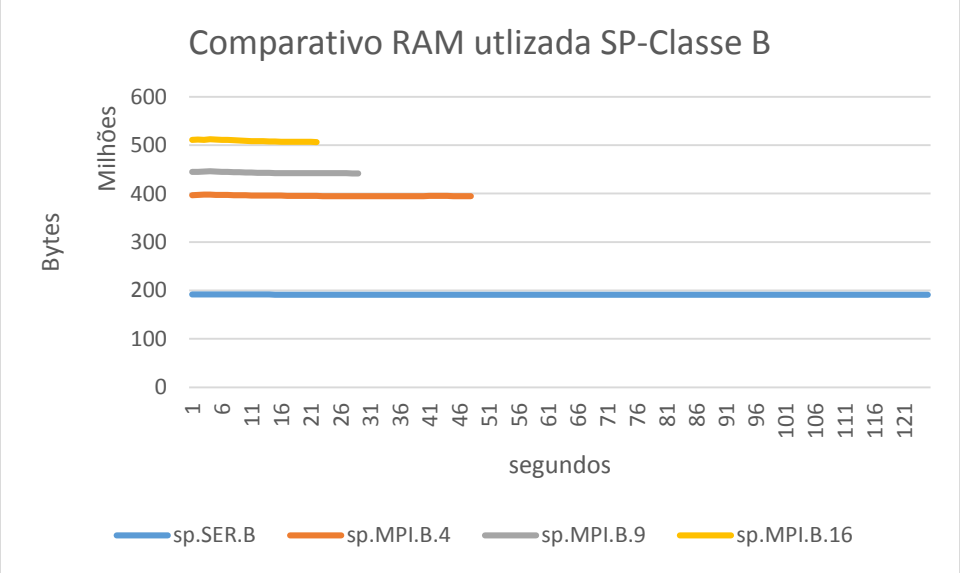
Análise da Memória

Os gráficos seguintes demonstram a alocação de Memória ao longo do tempo de execução.

Classe A:



Classe B:



Com o aumento de Processos o consumo de memória inevitavelmente também aumentou.

Conclusão

O `icc` neste kernel obteve os melhores resultados.

Scripts, Diretorias e Gráficos

Estes testes tiveram que ser realizados sucessivamente e sem a ajuda de scripts o trabalho seria muito penoso. Criei a seguinte estrutura de pastas para facilitar a organização:

```
.
|-- gcc
|   |-- 4.4.6
|   |   |-- MPI
|   |   |   |-- A
|   |   |   `-- B
|   |   |-- OMP
|   |   |   |-- A
|   |   |   `-- B
|   |   `-- SER
|   |       |-- A
|   |       `-- B
|   `-- 4.9.0
|       |-- MPI
|       |   |-- A
|       |   `-- B
|       |-- OMP
|       |   |-- A
|       |   `-- B
|       `-- SER
|           |-- A
|           `-- B
`-- icc
    |-- MPI
    |   |-- A
    |   `-- B
    |-- OMP
    |   |-- A
    |   `-- B
    `-- SER
        |-- A
        `-- B
```

Nas pastas A e B estão os executáveis respetivos.

De seguida fica um excerto do script que utilizei para fazer make aos programas e depois copiar para as pastas devidas, mostradas acima. Neste caso para MPI com `gcc` com a variável `v` a ser a versão (4.4.6 ou 4.9.0). Há o `make bundle` do NPB mas assim tenho a certeza e maior controlo do que acontece.

```
for th in 2 4 8 16
do
make is CLASS=A "NPROCS=$th"
make ep CLASS=A "NPROCS=$th"
make ft CLASS=A "NPROCS=$th"

make is CLASS=B "NPROCS=$th"
make ep CLASS=B "NPROCS=$th"
make ft CLASS=B "NPROCS=$th"
done

cp bin/*.A.* "../tests/gcc/$v/MPI/A/"
cp bin/*.B.* "../tests/gcc/$v/MPI/B/"
```

Para a medição de estatísticas foi necessário utilizar o `dstat` em background e o `mpstat` depois a colecionar os dados de cpu.

```
dstat -cdmrn --integer --output OMP_dstat.txt > /dev/null &
mpstat -P ALL 1 >> mps_OMP.txt
```

Com as medições iniciadas já se pode começar a chamar os programas. Incluí `sleep` para depois ser mais fácil analisar as medições pois estão espaçadas e em *idle*.

```
exec > "t_OMP.txt"
for th in 2 10 20 30
do
export OMP_NUM_THREADS=$th
echo  $th " ./ep.A.x"
./ep.A.x
sleep 3
echo  $th " ./ft.A.x"
./ft.A.x
sleep 3
echo  $th " ./is.A.x"
./is.A.x
sleep 3
echo  $th " ./sp.A.x"
./sp.A.x
sleep 3
done
sleep 6
```

Análise Final de Resultados e Conclusão

O trabalho necessário para realizar este projeto foi imenso, sendo talvez o que mais tempo precisou incluindo os de PCP do semestre passado. Tivemos que aplicar conhecimentos de Linux para mais rapidamente o realizarmos. Não consegui realizar tudo o que me propus, como realizar o teste em várias máquinas para fazer o comparar e dar sentido ao Benchmark e utilizar diferentes otimizações.

Os dados que deram origem a todo o trabalho estão divididos em 5 ficheiros *xlsx*: *grafico.xlsx* contém o resumo dos tempos obtidos, *A_cpu.xlsx* com a utilização por processador e *A_io.xlsx* com os dados de memória, disco e rede; e os respetivos para B, *B_cpu.xlsx* e *B_io.xlsx*. Devem ser consultados pois contêm dados individuais para cada teste. Apenas alguns deles foram colocados no Relatório.

Sempre utilizei o `top` para uma rápida análise da utilização dos processos no sistema mas com este projeto descobri novas ferramentas que serão muito úteis no futuro e que dão mais informações.

Com tudo concluído, é tempo de fazer as comparações das diferentes métricas e analisar os resultados. Nos 4 testes realizados o `icc` venceu três e o `gcc 4.9.0` outro. Analisando ainda os dados obtidos o `icc` foi o compilador que no geral melhores resultados conseguiu.