Universidade Federal de São Carlos

Centro de Ciências e de Tecnologia

Departamento de Computação

Programação Paralela e Distribuída

Exercício Programa 2 - EP 2

Professor Hermes Senger

Nome: Jhon Wislin Ribeiro Citron
RA: 776852

2º Semestre de 2023/2

1 - Especificações da Máquina

O job a ser executado (job.sh) foi submetido no nó de entrada do cluster, com as configurações de processador descritas abaixo por meio do comando ***lscpu***.

```
Arquitetura:                    x86_64
Modo(s) operacional da CPU:     32-bit, 64-bit
Ordem dos bytes:                Little Endian
CPU(s):                         4
Lista de CPU(s) on-line:        0-3
Thread(s) per núcleo:           1
Núcleo(s) por soquete:          4
Soquete(s):                     1
Nó(s) de NUMA:                  1
ID de fornecedor:               GenuineIntel
Família da CPU:                 6
Modelo:                         63
Nome do modelo:                 Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz
Step:                           0
CPU MHz:                        2297.339
BogoMIPS:                       4594.67
Fabricante do hipervisor:       VMware
Tipo de virtualização:          completo
cache de L1d:                   32K
cache de L1i:                   32K
cache de L2:                    256K
cache de L3:                    25600K
CPU(s) de nó0 NUMA:             0-3
Opções:                         fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov
pat pse36 clflush dts mmx fxsr sse sse2 ss ht syscall nx pdpe1gb rdtscp lm constant_tsc
arch_perfmon pebs bts nopl xtopology tsc_reliable nonstop_tsc cpuid pni pclmulqdq ssse3
fma cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c
rdrand hypervisor lahf_lm abm cpuid_fault invpcid_single pti ssbd ibrs ibpb stibp fsgsbase
tsc_adjust bmi1 avx2 smep bmi2 invpcid xsaveopt arat md_clear flush_l1d
arch_capabilities
```

Nó de entrada

Para a execução do problema em paralelo, foi utilizado o cluster OpenHPC da UFSCar, onde, ele foi submetido e executado no nó C21, pertencente a fila fast do cluster. As configurações do nó se encontram a seguir.

- Nome do Nó: C21
- Arquitetura do Processador: x86_64
- Número de Núcleos por Soquete: 8
- Número Total de Núcleos: 128
- Memória Total: 257500 MB

- Número de Soquetes: 8
- Número de Placas-Mãe: 1
- Número de Threads por Núcleo: 2 (cada núcleo do processador pode lidar com 2 threads simultaneamente, indicando suporte a tecnologia Hyper-Threading)
- Partições Disponíveis: fast (o nó está associado à partição "fast" para execução de trabalhos)

Este nó possui uma arquitetura x86_64 com 128 núcleos no total, distribuídos em 8 soquetes, com cada soquete possuindo 8 núcleos, e suporta 2 threads por núcleo. Ele está executando o sistema operacional Linux, versão 4.18.0-425.13.1.el8_7.x86_64. Atualmente, este nó se encontra disponível para execução de tarefas na partição "fast".

2 - Resultados

Com a execução e obtenção dos tempos de execução, foram feitos alguns cálculos para o Speedup e Eficiência em relação à variação de threads e utilizando um grid de 2500 por 2500, e um grid de 5000 por 5000.

Programa Sequencial (Grid 2500x2500)		
Threads	**Tempo**	**Speedup**
1	280,6	1

Tabela 1 - Resultado Sequencial com Grid de 2500 x 2500

Programa com Pthreads (Grid 2500 x 2500)			
Threads	**Tempo**	**Speedup**	**Eficiencia**
1	284,7	0,986	98,57%
2	183,4	1,530	76,49%
5	68,4	4,101	82,02%
10	48,7	5,768	57,68%
20	35,9	7,826	39,13%
40	48,2	5,821	14,55%

Tabela 2 - Resultado Pthreads com Grid de 2500 x 2500

Programa Sequencial (Grid 5000x5000)		
Threads	**Tempo**	**Speedup**
1	1123,1	1

Tabela 3 - Resultado Sequencial com Grid de 5000 x 5000

Programa com Pthreads (Grid 5000 x 5000)			
Threads	**Tempo**	**Speedup**	**Eficiencia**
1	1035,9	1,084	108,42%
2	560,5	2,004	100,19%
5	215,0	5,224	104,47%
10	134,0	8,380	83,80%
20	126,0	8,913	44,57%
40	152,5	7,364	18,41%

Tabela 4 - Resultado Pthreads com Grid de 5000 x 5000

3 - Gráficos do Speedup

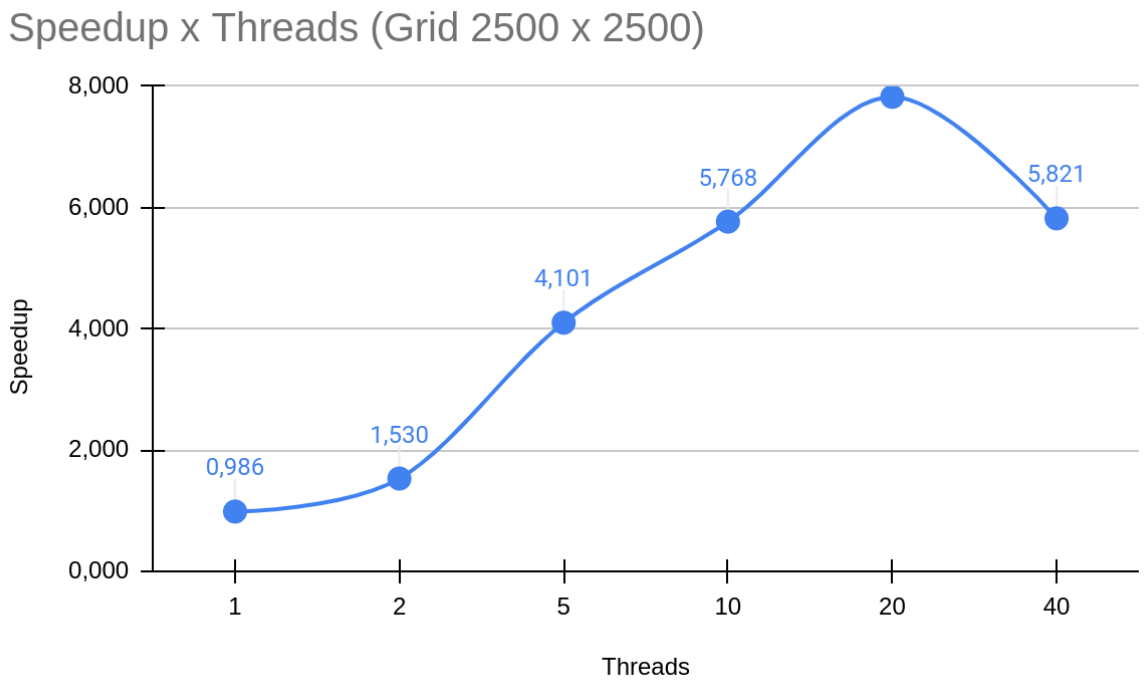

Gráfico 1 - Speedup pelo Número de Threads com Grid de 2500 x 2500

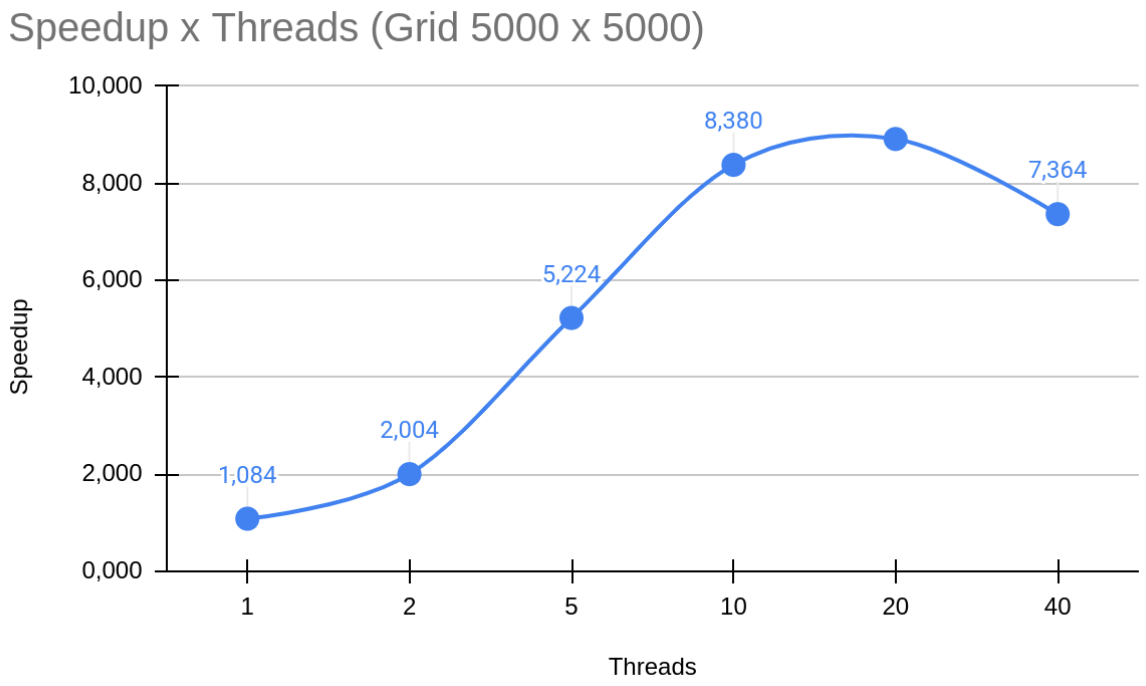

Gráfico 2 - Speedup pelo Número de Threads com Grid de 5000 x 5000

4 - Análise dos Resultados

A estratégia de paralelismo adotada consistiu em dividir o cálculo de Jacobi entre as threads. Cada thread executou a função ***jacobi_iteration***, realizando o cálculo em sua parte designada do grid e determinando um **erro_local**, que foi comparado com um **erro_global** por uma função ***max***. É crucial destacar a inclusão de um mutex para evitar condições de corrida, sendo o mutex um mecanismo de sincronização utilizado para controlar o acesso das threads a uma região crítica. Além disso, a atualização do grid foi encapsulada na função ***atualiza_grid***, atribuindo a cada thread a responsabilidade de atualizar uma parte específica do grid.

Com isso, para observar o comportamento do programa conforme o tamanho do grid fosse alterado, foram obtidos resultados dos tempos de execução correspondentes aos grids de 2500 por 2500 e 5000 por 5000.

Grid 2500 x 2500: Analisando os tempos de execução obtidos, com o aumento no número de threads de 1 até 40, foi possível observar que o tempo de execução diminuiu, atingindo o mínimo de 35,9 segundos com 20 threads. No entanto, com 40 threads, o tempo de execução aumentou para 48,2 segundos, resultando em um speedup menor. Isso sugere um overhead associado ao número excessivo de threads. Em relação à eficiência obtida, pode ser considerado que ela se manteve alta de 1 a 5 threads, estando em uma faixa de 75% a 100%. Posteriormente, a eficiência diminui, atingindo seu valor mínimo em 40 threads, com 14,55% de eficiência.

Grid 5000 x 5000: Analisando os tempos de execução obtidos, com o aumento no número de threads de 1 até 40, foi possível observar que o tempo de execução diminuiu, atingindo o mínimo de 126 segundos com 20 threads. No entanto, com 40 threads, o tempo de execução aumentou para 152,5 segundos, resultando em um speedup menor. Isso sugere um overhead associado ao número excessivo de threads. Em relação à eficiência obtida, pode ser considerado que ela se manteve alta de 1 a 10 threads, estando em uma faixa de 80% a 110%. Posteriormente, a eficiência diminui, atingindo seu valor mínimo em 40 threads, com 18,41% de eficiência.

Em resumo, a estratégia de paralelismo adotada proporcionou grandes melhorias nos tempos de execução. Contudo, ao ampliar o número de threads, observou-se uma diminuição na eficiência, indicando um ponto crítico de paralelismo. Vale ser ressaltado que a eficiência se manteve alta até 5 threads para o grid de 2500 por 2500, e o mesmo para o grid de 5000 por 5000, se mantendo alta até 10 threads. Além disso, vale ser ressaltado que com 40 threads ocorreu uma queda no speedup em ambos os casos, indicando overheads e limitações de escalabilidade.