Relatório
Trabalho 2 de Programação Paralela

GRR20190427 - João Lucas Cordeiro

1 Introdução

Este é um relatório do segundo trabalho de Programação Paralela, onde implementamos uma nova versão da soma de prefixos, parecida com a que já tínhamos, que roda com várias threads. O algoritmo desenvolvido será detalhado, assim como o resultado dos experimentos usando o código.

2 O Algoritmo

O algoritmo começa criando as threads com a função *PartialSum*, que trabalharão em pedaços do vetor de entrada. Nessa primera parte, somamos todos os números em cada um desses pedaços, pegamos essa soma e salvamos no vetor que guarda as somas parciais, o *partialSum*. Uma barreira faz as threads esperarem todas terminarem para continuar.

Então, fazemos uma soma de prefixos no *partialSum*, para usarmos no próximo passo.

Agora, criamos as threads com a função *FinalSum* que trabalharão nas mesmas partes das threads anteriores, dividindo o *InputVector* da mesma forma. Em cada thread, na primeira posição que aquela thread está lidando, é colocado o valor daquela posição do *InputVector* somado com a soma parcial da divisão anterior. Então, até o fim, é colocado o valor daquela posição do *InputVector* somado com a posição anterior do *OutputVector*. Uma barreira faz as threads esperarem todas terminarem para continuar.

Assim, fazemos a soma de prefixos do vetor de entrada e guardamos no vetor de saída.

3 Descrição do Processador

Os resultados citados neste relatório foram obtidos na máquina cpu1 do DINF. Usando o comando *lscpu* temos estas informações do processador:

Arquitetura: x86_64
Modo(s) operacional da CPU: 32-bit, 64-bit
Ordem dos bytes: Little Endian
Tamanhos de endereço: 48 bits physical, 48 bits virtual
CPU(s): 32
Lista de CPU(s) on-line: 0-31
Thread(s) per núcleo: 1
Núcleo(s) por soquete: 8
Soquete(s): 4
Nó(s) de NUMA: 1
ID de fornecedor: AuthenticAMD
Família da CPU: 23
Modelo: 1
Nome do modelo: AMD EPYC 7401 24-Core Processor
Step: 2
CPU MHz: 1999.999
BogoMIPS: 3999.99
Virtualização: AMD-V
Fabricante do hipervisor: KVM
Tipo de virtualização: completo
cache de L1d: 2 MiB
cache de L1i: 2 MiB
cache de L2: 16 MiB
cache de L3: 64 MiB
CPU(s) de nó0 NUMA: 0-31
Vulnerability Itlb multihit: Not affected
Vulnerability L1tf: Not affected
Vulnerability Mds: Not affected
Vulnerability Meltdown: Not affected
Vulnerability Mmio stale data: Not affected
Vulnerability Retbleed: Mitigation; untrained return thunk; SMT disabled
Vulnerability Spec store bypass: Mitigation; Speculative Store Bypass disabled via prctl
Vulnerability Spectre v1: Mitigation; usercopy/swapgs barriers and __user pointer sanitization
Vulnerability Spectre v2: Mitigation; Retpolines, IBPB conditional, STIBP disabled, RSB filling, PBRSB-eIBRS Not affected
Vulnerability Srbds: Not affected
Vulnerability Tsx async abort: Not affected
Opções:

fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ht syscall nx mmxext fxsr_opt pdpe1gb rdtscp lm rep_good no pl cpuid extd_apicid tsc_known_freq pni pclmulqdq ssse3 fma cx16 sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand hypervisor la hf_lm svm cr8_legacy abm sse4a misalignsse 3dnowprefetch osvw perfctr_core ssbd ibpb vmmcall fsgsbase tsc_adjust bmi1 avx2 smep bmi2 rdseed adx sm ap clflushopt sha_ni xsaveopt xsavec xgetbv1 xsaves clzero xsaveerptr virt_ssbd arat npt nrip_save arch_capabilities

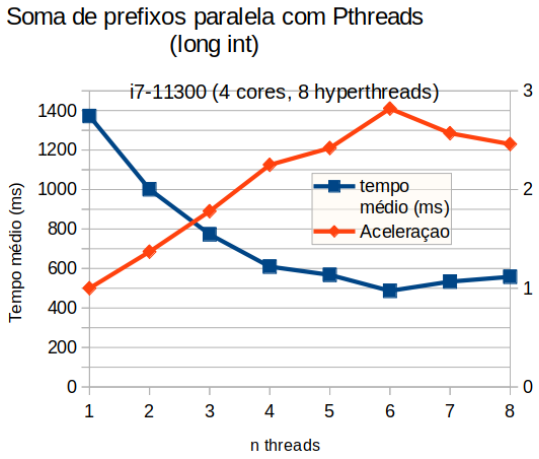
4 Resultados

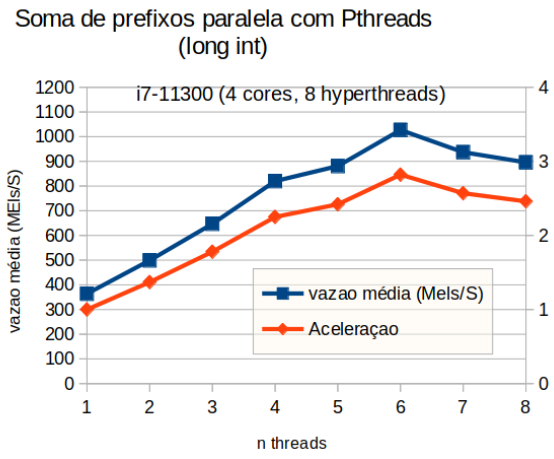
Os experimentos são rodados usando os scripts que o professor disponibilizou para a realização do trabalho. Aqui, mostramos os resultados de uma execução do experimento:

Threads	Tempo Med. v1	Tempo Med. v2	Aceleração
1	1372.14	1906.38	0.72
2	1001.16	1046.77	0.96
3	772.88	767.38	1.01
4	609.40	664.78	0.92
5	567.76	530.22	1.07
6	486.66	533.77	0.91
7	533.53	544.55	0.98
8	557.90	480.14	1.16

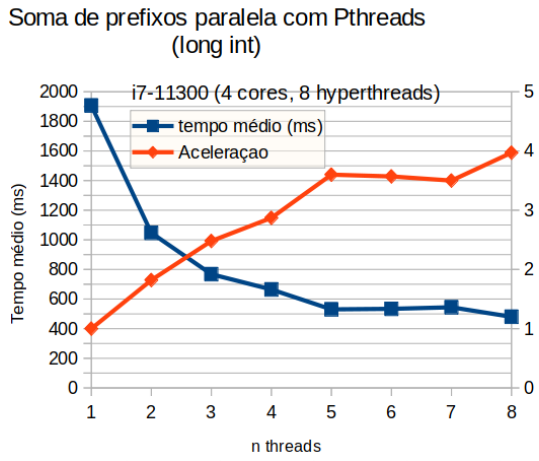
E aqui, tamos os gráficos gerados pela planilha do professor:

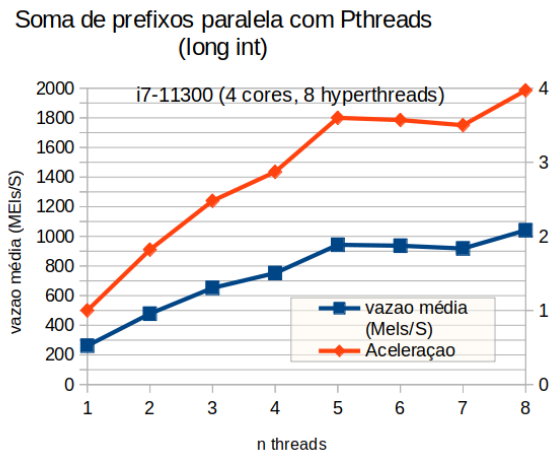
Gráficos para a v1:





Gráficos para a v2:





Gráficos comparando a v1 e a v2: