SIMULADOR DE PROPAGAÇÃO DE VÍRUS COM OPENMP

Gabriel Gomes Pereira

SOFTWARE E HARDWARE UTILIZADO

SOFTWARE

Subsistema windows para linux – Ubuntu 18.04 LTS

HARDWARE

Processador: Ryzen 5 1400 4 núcleos 3.2 a 3.4 GHZ

Memória RAM: 8GB

Estratégias utilizadas

Solução 1

Foi paralelizado o trecho de código responsável por testar a simulação repetidas vezes. Foi utilizado omp for com scheduling estático.

Solução 2

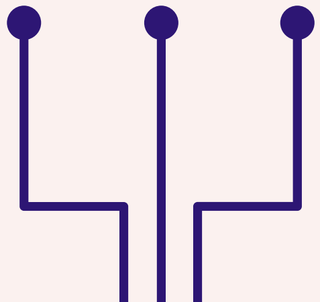
Foi paralelizado o trecho de código responsável pelo cálculo do percentual de infectados utilizando novamente omp for com scheduling estático

SOLUÇÃO 1 – PARELIZAR OS TESTES DA SIMULAÇÃO

Cada thread processa uma porção aproximadamente igual de experimentos.

SOLUÇÃO 2 – PARELIZAR O ACESSO A MATRIZ PARA O CÁLCULO DO PERCENTUAL

Cada thread percorre uma porção aproximadamente igual da matriz que representa a população para obter o número de infectados.



Problema

População: 15 x 15
Experimentos: 2000
Número de probabilidades: 101

Obs: Os tempos de execução estão em microssegundos

Resultados solução 1

THREADS	TEMPO	SPEEDUP	EFICIÊNCIA
1	8418556		
2	7392327	1,1388	0,5694
4	16700543	0,5041	0,1260
8	18154080	0,4637	0,057

Resultados solução 2

THREADS	TEMPO	SPEEDUP	EFICIÊNCIA
1	8558744		
2	11000976	0,778	0,389
4	12820199	0,6676	0,1669
8	18472845	0,4633	0,0579

Problema

População: 30 x 30
Experimentos: 2000
Número de probabilidades: 101

Resultados solução 1

THREADS	TEMPO	SPEEDUP	EFICIÊNCIA
1	54128310		
2	36032838	1,5022	0,7511
4	68972348	0,785	0,1962
8	77806757	0,7	0,0875

Resultados solução 2

THREADS	TEMPO	SPEEDUP	EFICIÊNCIA
1	54320301		
2	60904615	0,9	0,45
4	69763119	0,78	0,195
8	93533697	0,5807	0,072

Problema

População: 60 x 60
Experimentos: 2000
Número de probabilidades: 101

Resultados solução 1

THREADS	TEMPO	SPEEDUP	EFICIÊNCIA
1	352280929		
2	215167147	1,6372	0,8186
4	290601457	1,2122	0,3025
8	336704619	1,0463	0,1307

Resultados solução 2

THREADS	TEMPO	SPEEDUP	EFICIÊNCIA
1	358022514		
2	384651142	0,9308	0,4654
4	412506255	0,87	0,0417
8	469732709	0,7622	0,09527

CONCLUSÕES

SOLUÇÃO 1

Na maioria dos casos só se obteve desempenho utilizando 2 threads. Nos demais casos o desempenho decaiu em relação a execução sequencial.

SOLUÇÃO 2

Foi a pior das soluções onde não se obteve ganho de desempenho em relação a execução sequencial em nenhum dos casos.