T4: Simulador de Propagação de Vírus com OpenMP

Alexandre Moreira Medina

ELC139 - 2020a

Solução 1

Paralelizar a região onde é executada os n experimentos, definindo como privada o objeto que manipula a população. Para garantir que nenhum resultado seja alterado devido a condição de corrida, na variável *percent_infected* foi utilizado exclusão mútua.

```
#pragma omp parallel for schedule(dynamic) private(population)
        for (int it = 0; it < n_trials; it++) {
            population = new Population(population_size);
            population->propagateUntilOut(population->centralPerson(), prob_spread[ip], rand);
#pragma omp critical
            {
                percent_infected[ip] += population->getPercentInfected();
            }
        }
```

Solução 2

Paralelizar o loop de probabilidades, utilizando um vetor de População, para que cada thread manipule uma população separada uma da outra, fazendo com que o uso de exclusão mútua não seja necessária em nenhuma parte do loop.

```
#pragma omp parallel for schedule(dynamic)
      for (int ip = 0; ip < n_probs; ip++) {
         int current_thread = omp_get_thread_num();

         prob_spread[ip] = prob_min + (double) ip * prob_step;
         percent_infected[ip] = 0.0;
         rand.setSeed(base_seed+ip); // nova seqüência de números aleatórios

         // executa vários experimentos para esta probabilidade
         for (int it = 0; it < n_trials; it++) {
            population[current_thread]->propagateUntilOut(population[current_thread]->centralPerson(), prob_spread[ip], rand);
            percent_infected[ip] += population[current_thread]->getPercentInfected();
         }

         // calcula média dos percentuais de árvores queimadas
         percent_infected[ip] /= n_trials;

         // mostra resultado para esta probabilidade
         printf("%lf, %lf\n", prob_spread[ip], percent_infected[ip]);
      }
```

Especificações

Notebook Lenovo Ideapad 330 81FES00300

Intel Core i5-8250U

8GB Ram

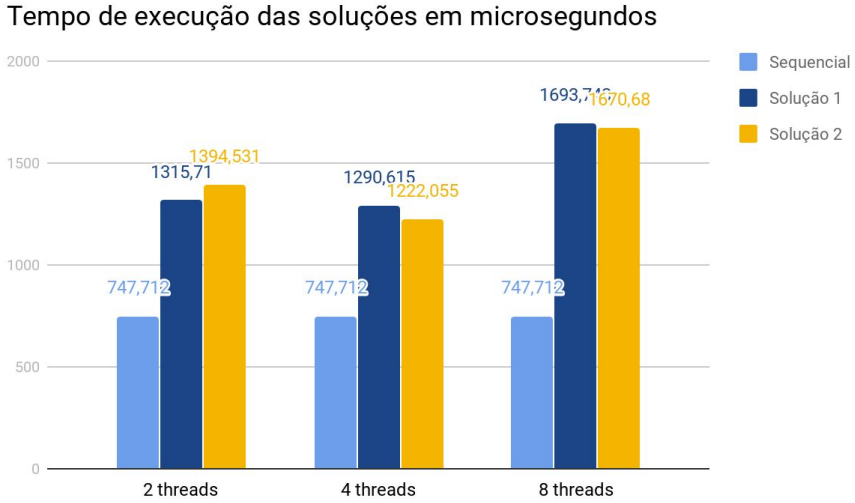
Ubuntu 19.04

População pequena - n = 10

	2 threads	Speedup
Sequencial	747,712	1
Solução 1	1315,71	1335,2
Solução 2	1394,531	0,53

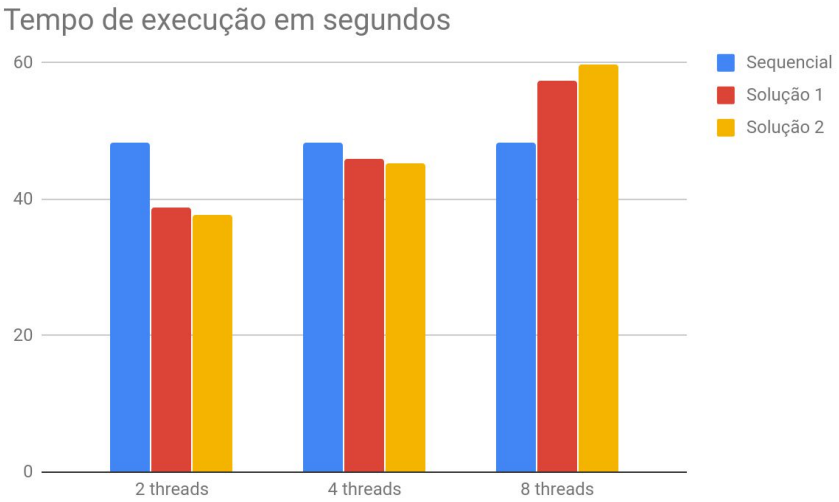
	2 threads	Speedup
Sequencial	747,712	1
Solução 1	1290,615	0,57
Solução 2	1222,055	0,61

	2 threads	Speedup
Sequencial	747,712	1
Solução 1	1693,743	0,441
Solução 2	1670,68	0,447



População média - n = 15

	2 threads	Speedup
Sequencial	48,290504	1
Solução 1	38,752963	1,24
Solução 2	37,794081	1,27
	4 threads	Speedup
Sequencial	48,290504	1
Solução 1	45,964727	1,05
Solução 2	45,246575	1,06
	8 threads	Speedup
Sequencial	48,290504	1
Solução 1	57,424346	0,84
Solução 2	59,817438	0,8



População grande - n = 100

	2 threads	Speedup
Sequencial	17483,6083	1
Solução 1	12713,8241	1,37
Solução 2	11731,2074	1,49

	4 threads	Speedup
Sequencial	17483,6083	1
Solução 1	11399,341	1,53
Solução 2	11286,5763	1,54

	8 threads	Speedup
Sequencial	17483,6083	1
Solução 1	12941,2841	1,35
Solução 2	12820,495	1,36