T5: Aplicação do Método de Monte Carlo em OpenMP

Felipe Marin

Solução OpenMP 1

- Paraleliza o laço de probabilidades
- Cada thread recebe um ‘pedaço’ dos valores de probabilidade de maneira dinâmica.

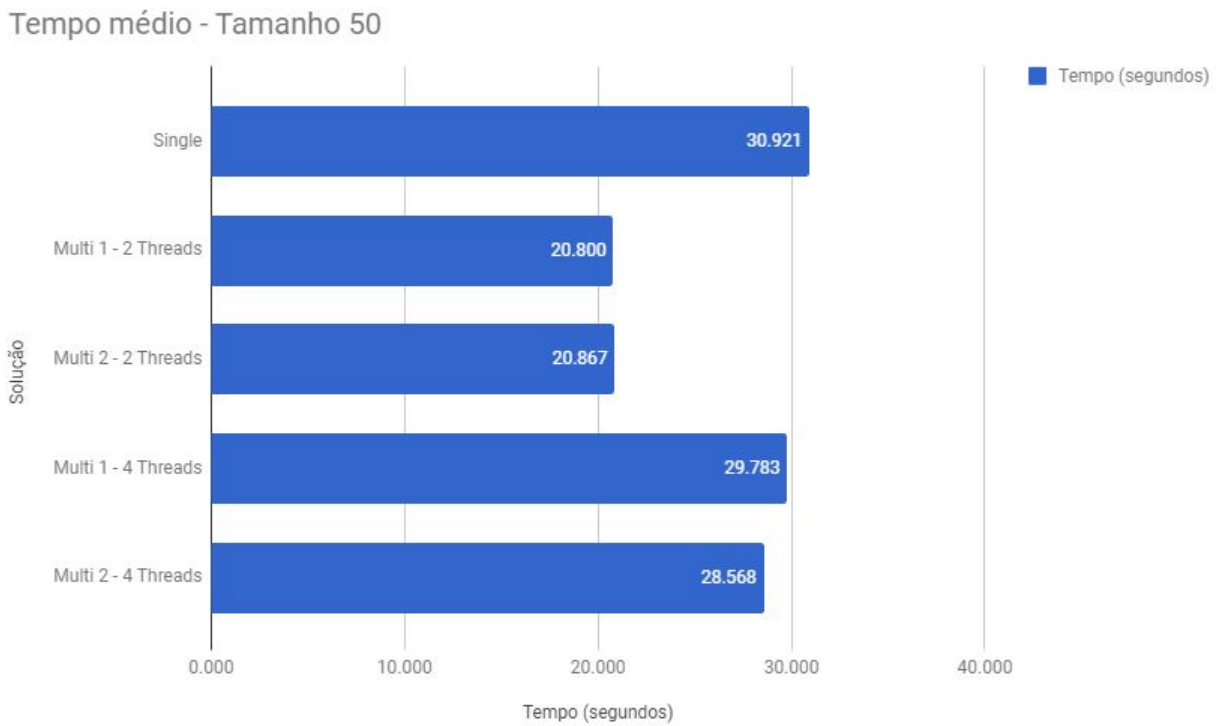
```
//paraleliza o laço de probabilidades
#pragma omp parallel shared(prob_spread, percent_burned) private(forest, ip, it)
{
forest = new Forest(forest_size);
// para cada probabilidade, calcula o percentual de árvores queimadas
#pragma omp for schedule(dynamic, chunk)
for (ip = 0; ip < n_probs; ip++) {
   prob_spread[ip] = prob_min + (double) ip * prob_step;
   percent_burned[ip] = 0.0;
   rand.setSeed(base_seed+ip); // nova sequência de números aleatórios
   // executa vários experimentos
   for (it = 0; it < n_trials; it++) {
      // queima floresta até o fogo apagar
      forest->burnUntilOut(forest->centralTree(), prob_spread[ip], rand);
      percent_burned[ip] += forest->getPercentBurned();
   }
   // calcula m�dia dos percentuais de árvores queimadas
   percent_burned[ip] /= n_trials;
   // mostra resultado para esta probabilidade
   printf("%lf, %lf\n", prob_spread[ip], percent_burned[ip]);
}
}
```

Solução OpenMP 2

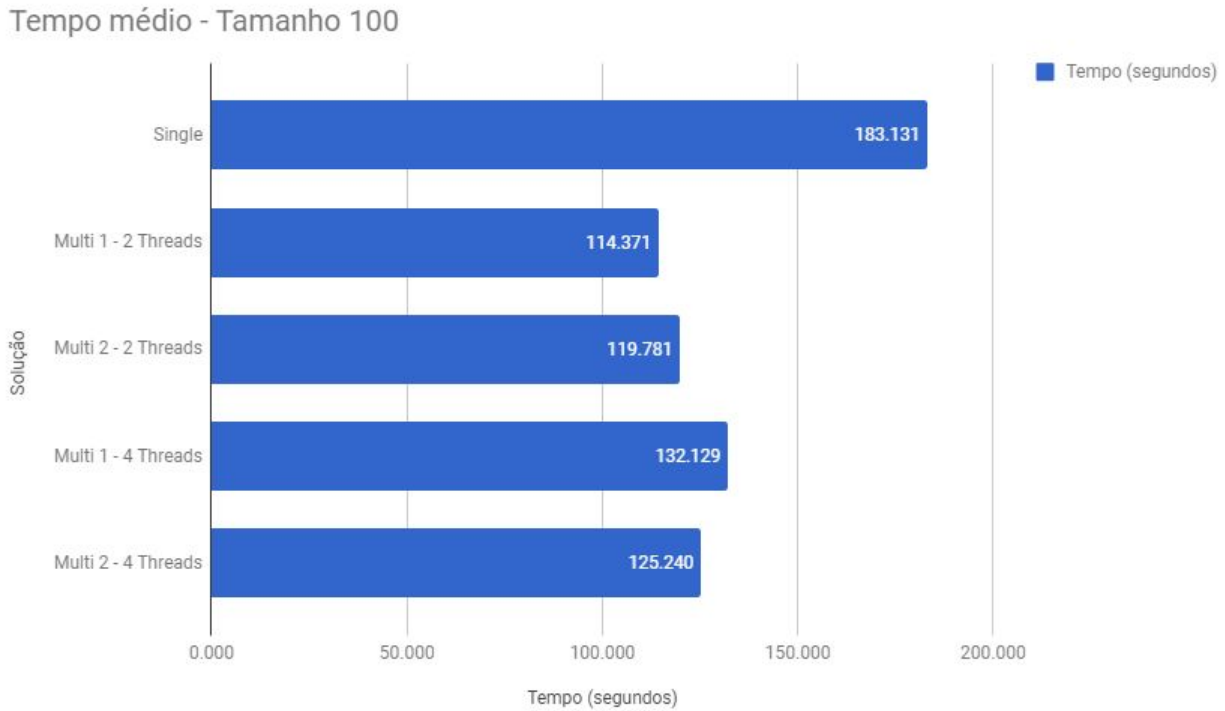
- Paraleliza o laço de experimentos
- Cada thread executa n experimentos de maneira estática

```
// para cada probabilidade, calcula o percentual de árvores queimadas
for (ip = 0; ip < n_probs; ip++) {
   prob_spread[ip] = prob_min + (double) ip * prob_step;
   percent_burned[ip] = 0.0;
   rand.setSeed(base_seed+ip); // nova sequência de números aleatórios
   //paraleliza os experimentos
   #pragma omp parallel shared(prob_spread, percent_burned) private(forest, it)
   {
   forest = new Forest(forest_size);
   // executa vários experimentos
   #pragma omp for schedule(static)
   for (it = 0; it < n_trials; it++) {
      // queima floresta até o fogo apagar
      forest->burnUntilOut(forest->centralTree(), prob_spread[ip], rand);
      #pragma omp critical
      percent_burned[ip] += forest->getPercentBurned();
   }
   }
```

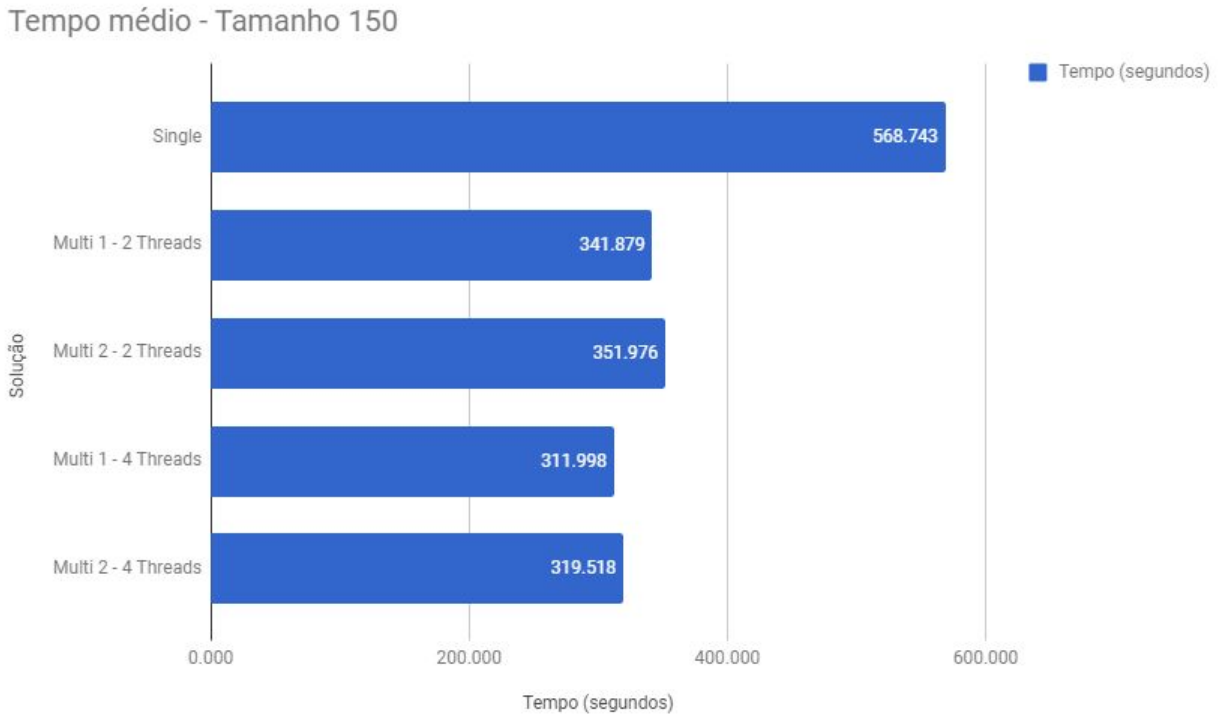
Tempo de execução - Tamanho 50



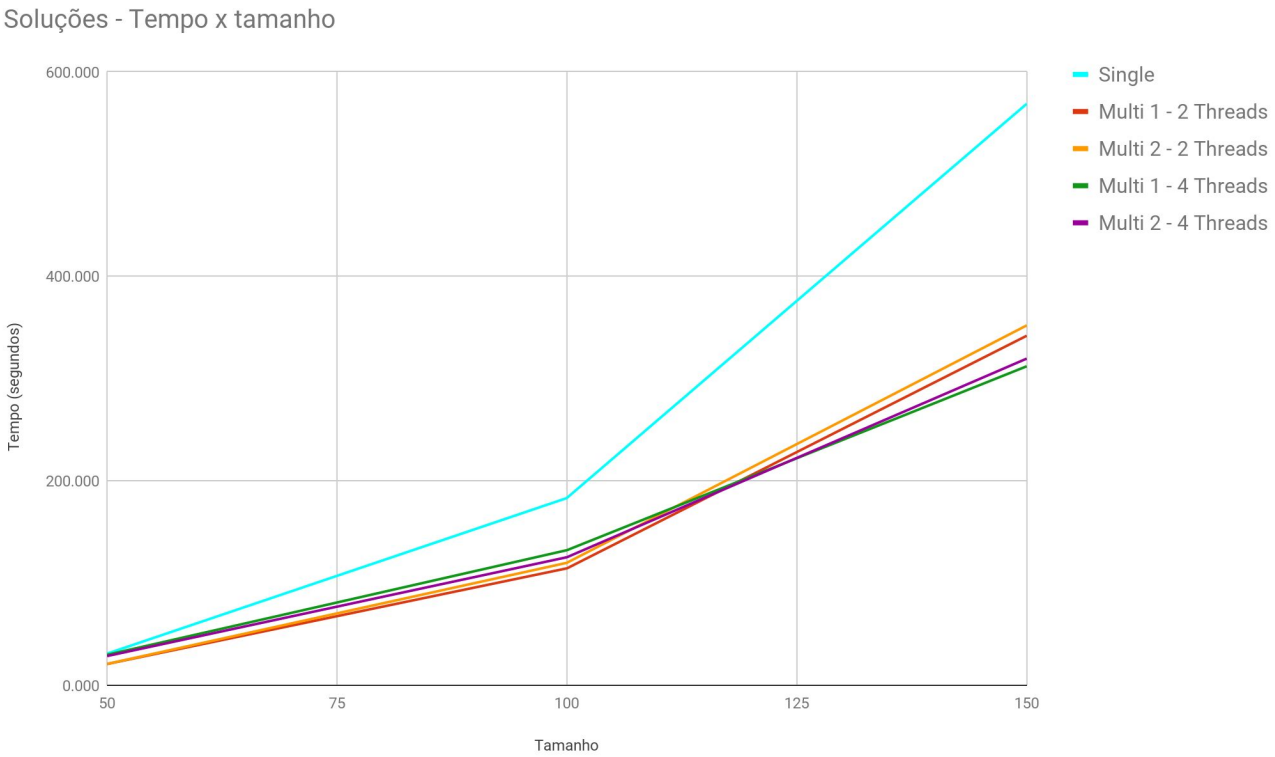
Tempo de execução - Tamanho 100



Tempo de execução - Tamanho 150



Soluções - Tempo x Tamanho



Conclusões

- Ambas as soluções com OpenMP reduzem o tempo de execução
- Um maior número de threads apresenta melhor desempenho conforme aumenta o tamanho da matriz