T5: Aplicação do Método de Monte Carlo em OpenMP

Disciplina: ELC139 - Programação Paralela
Professora: Andrea Schwertner Charão
Aluna: Marinara Rübenich Fumagalli

1ª SOLUÇÃO

- ESTRATÉGIAS
- DESCRIÇÕES
- RESULTADOS

- Ambas as estratégias para melhorar o desempenho foram implementadas na main() no trecho onde são calculados os percentuais de acordo com as probabilidades...

```
// para cada probabilidade, calcula o percentual de árvores queimadas
for (int 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 seqüência de números aleatórios

   // executa vários experimentos
   for (int 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 média 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]);
}
```

- As linhas de código que foram incluídas são:

```
// para cada probabilidade, calcula o percentual de árvores queimadas
#pragma omp parallel num_threads(2) private(it, ip)
{
    Forest* forest = new Forest(forest_size);
    #pragma omp for schedule(auto)
    for (int ip = 0; ip < n_probs; ip++) {
...
```

- Elas fazem a divisão dos cálculos das médias dos percentuais de árvores queimadas entre as threads existentes.
- Nesse exemplo:
  - num_threads(2) - o cálculo está sendo dividido entre 2 threads;
  - schedule(auto) - é o compilador quem decide como será a divisão das iterações entre as threads (melhora consideravelmente o desempenho);
  - private(ip, it) - devem ser privadas pois cada thread individualmente deverá tratar as posições que irá percorrer, senão resultará num erro de cálculo.

Análise de Desempenho – Problema G

<60, 2000, 75>

Tempo Sequencial = 176581446µ (2,94m)

Tempo OpenMP (2 threads) = 150996391µ (2,51m)

Speed Up = 1,16

Tempo OpenMP (4 threads) = 104436946µ (1,74m)

Speed Up = 1,69

Análise de Desempenho - Problema M

<50, 1500, 65>

Tempo Sequencial = 72019259µ (1,20m)

Tempo OpenMP (2 threads) = 60697432µ (1,01m)

Speed Up = 1,18

Tempo OpenMP (4 threads) = 39032066µ (39,03s)

Speed Up = 1,84

Análise de Desempenho – Problema P

<30, 1000, 50>

Tempo Sequencial = 9116163µ (9,11s)

Tempo OpenMP (2 threads) = 7829765µ (7,82s)

Speed Up = 1,16

Tempo OpenMP (4 threads) = 5332034µ (5,33s)

Speed Up = 1,70