

T5: Aplicação do Método de Monte Carlo em OpenMP


Disciplina: ELC139 - Programação Paralela

Professora: Andrea Schwertner Charão

Aluna: Marinara Rübenich Fumagalli



2ª 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)
```

```
{
```

```
 Forest* forest = new Forest(forest_size);
```

```
 ...
```

```
 // executa vários experimentos
```

```
 #pragma omp for schedule(auto)
```

```
 for (int it = 0; it < n_trials; it++) {
```

```
 ...
```

- Elas fazem a divisão da execução de experimentos 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(it) - deve ser privada 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 = 180184833 $\mu$  (3,003m)

Tempo OpenMP (2 threads) = 113292756 $\mu$  (1,88m)

Speed Up = 1,59

Tempo OpenMP (4 threads) = 80520237 $\mu$  (1,34m)

Speed Up = 2,23



# *Análise de Desempenho - Problema M*

<50, 1500, 65>

Tempo Sequencial = 71486135 $\mu$  (1,19m)

Tempo OpenMP (2 threads) = 43776993 $\mu$  (43,77s)

Speed Up = 1,63

Tempo OpenMP (4 threads) = 32360309 $\mu$  (32,36s)

Speed Up = 2,20



# *Análise de Desempenho - Problema P*

<30, 1000, 50>

Tempo Sequencial = 8903295 $\mu$  (8,90s)

Tempo OpenMP (2 threads) = 5661428 $\mu$  (5,66s)

Speed Up = 1,57

Tempo OpenMP (4 threads) = 4305575 $\mu$  (4,30s)

Speed Up = 2,06

