

# Aplicação do Método de Monte Carlo em OpenMP

**Alunos: Ana Veroneze e Gabriel Cardoso**

**Professora: Andrea Charão**

**Disciplina: ELC139 - Programação Paralela**

# Ambiente de execução do programa

---

**Hardware:** Intel(R) Core(™) i5-4200u @1.6 GHz, 2 cores físicos e 4 virtuais, Cache L1 de 128KB, L2 de 512KB, L3 de 3MB, 6 GB de RAM.

**Software:** Linux Ubuntu 16.04 LTS x64

**Compilador:** gcc version 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.4)

# Alterações no programa:

---

Alteração do Makefile: flag **-fopenmp**

Inclusão da biblioteca **omp.h**

Utilização das **diretivas OpenMP**

Pequenas alterações no código: posição de declaração de variáveis, modificações para os experimentos - cálculo do tempo de execução

# 1ª Solução: Paralelização na main

— — —

**Objetivo:** Dividir o cálculo do percentual de árvores queimadas entre diferentes threads

**Onde?**

```
// 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 sequência 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]);
}
```

# Código alterado

```
1 // para cada probabilidade, calcula o percentual de árvores queimadas
  #pragma omp parallel private (ip, it)
  {
    Forest* forest = new Forest(forest_size);
2  #pragma omp for
    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();
        }

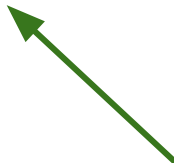
        // calcula média dos percentuais de árvores queimadas
        percent_burned[ip] /= n_trials;

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

# Código alterado

— — —

```
1  #pragma omp parallel private (ip, it)
   {
       ...
   }
```

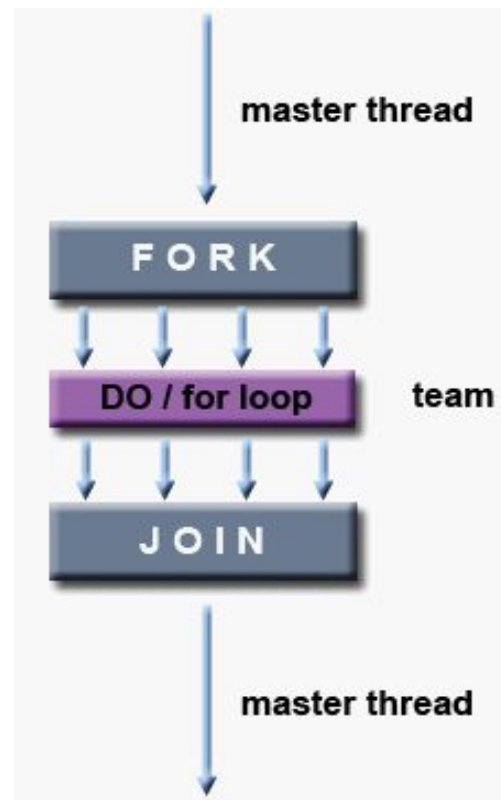


Variáveis de  
controle dos laços

# Código alterado

```
2 Forest* forest = new Forest(forest_size);  
  #pragma omp for  
  for (ip = 0; ip < n_probs; ip++) {  
    ...  
  }
```

```
// queima floresta até o fogo apagar  
forest->burnUntilOut(forest->centralTree(), prob_spread[ip], rand);  
percent_burned[ip] += forest->getPercentBurned();
```





## 2ª Solução: Paralelização na main

---

**Objetivo:** Dividir a execução dos experimentos entre diferentes threads

**Onde?**

```
// 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();
}
```

# Código alterado

```
// executa vários experimentos
#pragma omp parallel 1
{
    2 Forest* forest = new Forest(forest_size);
    #pragma omp for
    for (it = 0; it < n_trials; it++) {
        // queima floresta até o fogo apagar
        forest->burnUntilOut(forest->centralTree(), prob_spread[ip], rand);
        3 #pragma omp critical
        percent_burned[ip] += forest->getPercentBurned();
    }
}
```

# Código alterado

— — —

```
1  #pragma omp parallel
    {
        ...
    }
```

## Código alterado

— — —


```
2 Forest* forest = new Forest(forest_size);  
    #pragma omp for  
    for (it = 0; it < n_trials; it++) {  
        ...  
    }
```

# Código alterado

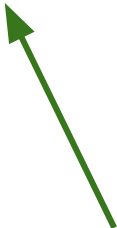
— — —

**3** `#pragma omp critical`  
`percent_burned[ip] += forest->getPercentBurned();`

Solução encontrada



A variável ip controla o laço mais externo,  
que não pertence a região paralela



# Experimentos

— — —

**10** execuções

Programas:

- firesim - **sequencial**
- firesim-omp-1 - **2 e 4 threads**
- firesim-omp-2 - **2 e 4 threads**

Entradas <**tamanho-do-problema**> <**nro. experimentos**> <**probab. maxima**>:

- 30 5000 101
- 15 2500 50
- 7 1500 25

# Resultados

— — —

## Sequencial

	Menor tempo (seg)	Maior tempo (seg)	Média (seg)
30 5000 101	105,696849	114,386657	107,953845
15 2500 50	4,504353	4,979517	4,644869
7 1500 25	0,227899	0,281643	0,253136

# Resultados

---

## Primeira variação

		Menor tempo (seg)	Maior tempo (seg)	Média (seg)
30 5000 101	2 Thread	93,058603	104,390215	100,167741
	4 Thread	117,134443	142,667820	127966102
15 2500 50	2 Thread	4,035692	5,430337	4,636491
	4 Thread	4,499608	4,664057	4,587158
7 1500 25	2 Thread	3,05426	3,27040	3,20229
	4 Thread	2,78781	2,93931	2,84634



# Resultados

— — —

## Segunda variação

		Menor tempo (seg)	Maior tempo (seg)	Média (seg)
30 5000 101	2 Thread	74,832610	78,126565	76,351045
	4 Thread	72,892472	73,542390	73,223783
15 2500 50	2 Thread	4,146894	4,184084	4,169830
	4 Thread	4,007927	4,245759	4,108193
7 1500 25	2 Thread	3,51614	4,51613	3,75467
	4 Thread	2,56453	3,12280	3,04741

**./firesim 30 5000 101**

— — —

Thread	Speedup V1	Eficiência V1	Speedup V2	Eficiência V2
2	1.0778	0.5389	1.4138	0.7069
4	0.8435	0.2108	1.4742	0.3685

**./firesim 15 2500 50**

Thread	Speedup V1	Eficiência V1	Speedup V2	Eficiência V2
2	1.0019	0.5009	1.1307	0.5653
4	1.0126	0.2531	1.1141	0.2785

**./firesim 7 1500 25**

Thread	Speedup V1	Eficiência V1	Speedup V2	Eficiência V2
2	0.7904	0.3952	0.6742	0.3371
4	0.8893	0.2223	0.8306	0.2076

# Conclusão e discussão

— — —

configuração 1 - versão 2

configuração 2 - versão 1 e sequencial

configuração 3 - versão sequencial

= dados pequenos melhor execução sequencial

Melhoras no desempenho? Não como o esperado.