Insper

SuperComputação

Aula 10 – Efeitos colaterais e paralelização

2019 – Engenharia

Luciano Soares <lpsoares@insper.edu.br>
Igor Montagner <igorsm1@insper.edu.br>

Aulas passadas

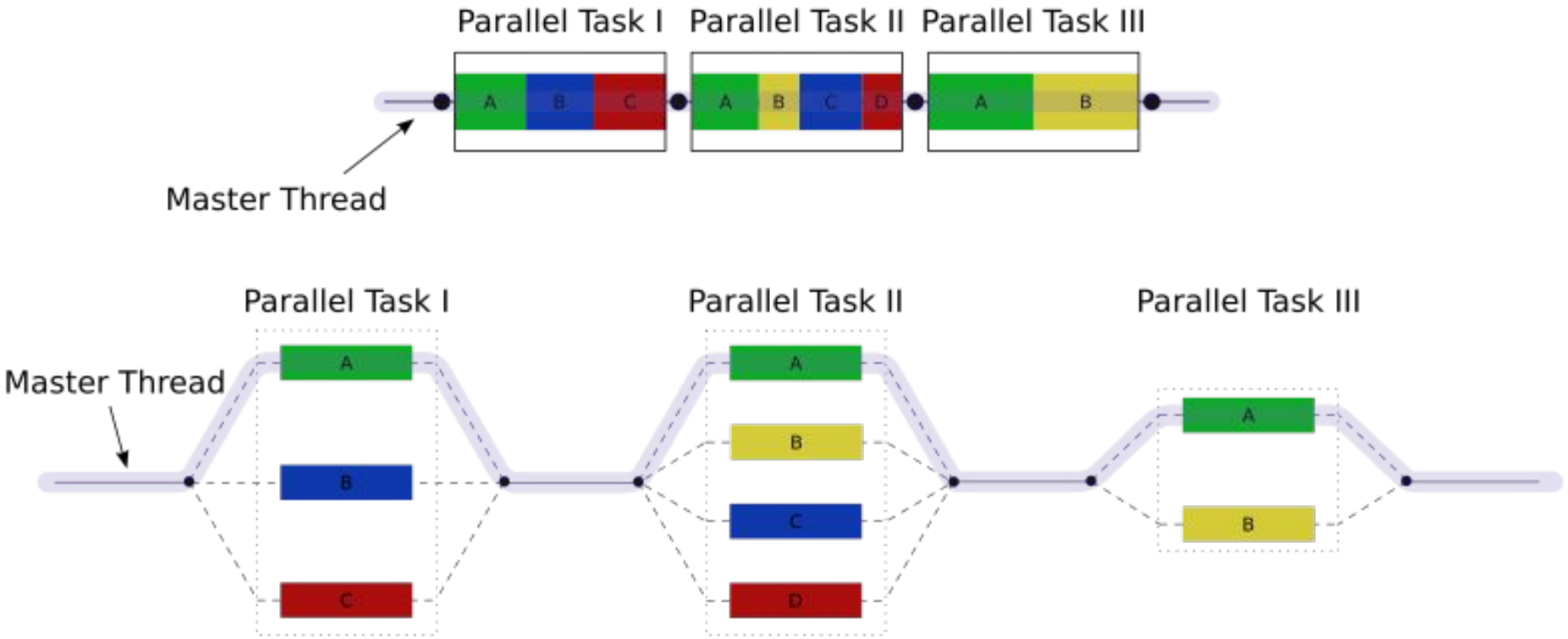


Figura: https://en.wikipedia.org/wiki/File:Fork_join.svg

Aulas passadas

Modelo fork-join:

```
#pragma omp parallel
{
    int id = omp_get_thread_num();
    long nt = omp_get_num_threads();
    double d = 0;
    for (long i = id; i < num_steps; i+=nt) {
        double x = (i-0.5)*step;
        d += 4.0/(1.0+x*x);
    }
    #pragma omp atomic
        sum += d;
}
```

Aulas passadas

Modelo fork-join:

```
#pragma omp parallel for reduction(+:sum)
for (long i = 0; i < num_steps; i++) {
    double x = (i-0.5)*step;
    sum += 4.0/(1.0+x*x);
}
return sum * step;
```

Hoje

- Comentários *mandel.c*
- Códigos com efeitos colaterais
- Mini-projeto: cálculo do pi usando sorteios aleatórios

Exercício *mandel.c* (solução)

```c
# define NPOINTS 1000
# define MAXITER 1000

struct d_complex{
  double r;
  double i;
};
struct d_complex c;
int numoutside = 0;

void testpoint(struct d_complex);

int main(){
  int i, j;
  double area, error, eps  = 1.0e-5;
  #pragma omp parallel for default(shared) private(c,j) firstprivate(eps)
     for (i=0; i<NPOINTS; i++) {
       for (j=0; j<NPOINTS; j++) {
         c.r = -2.0+2.5*(double)(i)/(double)(NPOINTS)+eps;
         c.i = 1.125*(double)(j)/(double)(NPOINTS)+eps;
         testpoint(c);
       }
     }

  area=5.625*(double)(NPOINTS*NPOINTS-numoutside)/(double)(NPOINTS*NPOINTS);
  error=area/(double)NPOINTS;
  printf("Area of Mandlebrot set = %12.8f +/- %12.8f\n",area,error);
  printf("Correct answer should be around 1.510659\n");
}
```

```c
void testpoint(struct d_complex c){
  struct d_complex z;
  int iter;
  double temp;
  z=c;
  for (iter=0; iter<MAXITER; iter++){
     temp = (z.r*z.r)-(z.i*z.i)+c.r;
     z.i = z.r*z.i*2+c.i;
     z.r = temp;
     if ((z.r*z.r+z.i*z.i)>4.0) {
       #pragma omp atomic
       numoutside++;
       break;
     }
  }
}
```

Agora as respostas são constantes e corretas.

Exercício *mandel.c*

Qual a nota de vocês para

- organização do código?
- facilidade de leitura?
- boas práticas de programação?

Exercício *mandel.c*

Exercício 4 pedia para reorganizar o código.

Isso diminuiu os problemas de paralelização?

Exercício *mandel.c*

Efeitos colaterais: função que lê ou modifica o estado global do programa.

1) Seu resultado não depende somente dos argumentos passados;
2) A função escreve seus resultados em outros lugares além do seu valor de retorno.

Exercício *mandel.c*

Mundo ideal: nenhuma função modifica o estado global do programa, facilitando muito a paralelização

Mundo real: eliminar todos efeitos colaterais pode tornar o código menos claro, menos eficiente e muito menos legível

Exercício *mandel.c*

Linguagens funcionais

Mundo ideal: nenhu
estado global do pr
a
paralelização

Mundo real: eliminar todos efeitos colaterais pode tornar o código menos claro, menos eficiente e muito menos legível

Exercício *mandel.c*

Mundo ideal: nenhuma função modifica o estado global do programa, facilitando muito a paralelização

Mundo real: **diminuir** efeitos colaterais na **parte paralela** do código pode

1) evitar problemas de compartilhamento de dados e de concorrência por recursos
2) melhorar organização do código

Exercício *mandel.c*

Objetivo é escrever código *threadsafe*:

manipula dados de modo que nenhuma thread interfira na execução de outra.

- Evitar estado compartilhado/efeitos colaterais
- Utilizar primitivas de sincronização para acessar os dados compartilhados

Atividade prática

- Cálculo do PI usando algortimo probabilístico
- Diversas opções de paralelização

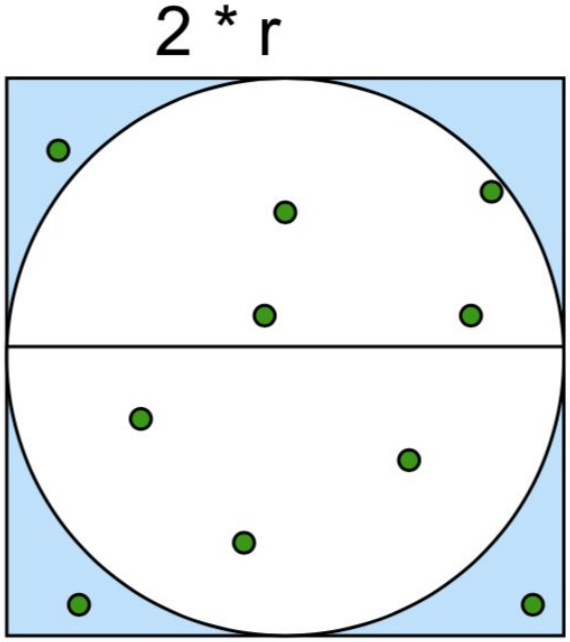
Geração de números aleatórios

- Em um gerador aleatório não é possível prever qual será o próximo número
- Podemos criar geradores pseudo-aleatórios: sequência depende de uma regra conhecida, mas possui propriedades parecidas com uma sequência aleatória de verdade
- A “aleatoriedade” da sequência depende do método e dos parâmetros usados
- Possibilidade de realizar simulações estatísticas

Método de Monte Carlo

Aproximar algum valor baseado em sorteios aleatórios

Exemplo:



Lance dardos aleatoriamente no quadrado
Probabilidade de cair no círculo é proporcional às áreas:

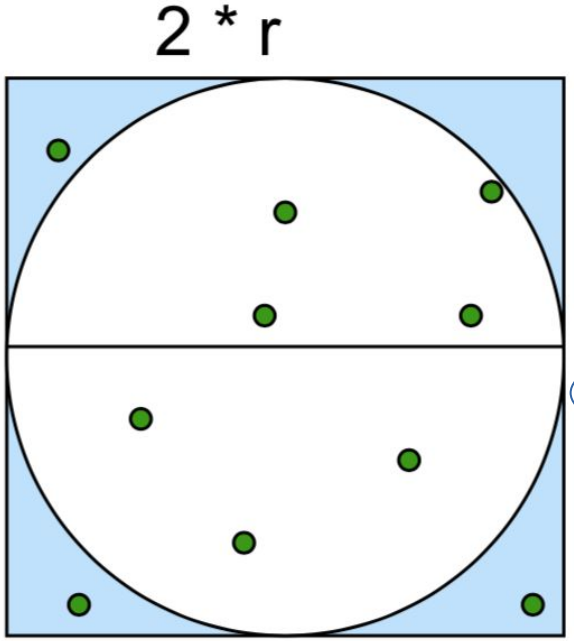
$A_c = \pi * r^2$
$A_q = (2*r) * (2*r) = 4 * r^2$
Prob. = $A_c/A_s = \pi/4$

N= 10	π = 2.8
N=100	π = 3.16
N= 1000	π = 3.148

Método de Monte Carlo

Aproximar algum valor baseado em sorteios aleatórios

Exemplo: Cálculo do PI



Define qualidade da aproximação!

Probabilidade de um ponto cair dentro do círculo:

Método:

1) Sorteia N pontos
2) Conta pontos dentro (I) e fora (F)
3) pi = 4 * (I/F)

Atividade prática

Cálculo do PI usando algortimo probabilístico

Objetivo: explorar diferentes estratégias de paralelização

1. exclusão mútua
2. dividir tarefas paralelizáveis e não paralelizáveis
3. transformar em tarefas independentes

Atividade prática

Vamos trabalhar hoje no roteiro 06 – Efeitos colaterais e geração de números aleatórios

Objetivo: tornar a implementação do LCG usada no exemplo *thread safe.*

Seu trabalho será o código para que cada thread gere uma sequência independente de números aleatórios.

Referências

- Livros:
  - Hager, G. ; Wellein, G. **Introduction to High Performance Computing for Scientists and Engineers**. 1ª Ed. CRC Press, 2010.
- Artigos:
  - Dagum, Leonardo, and Ramesh Menon. "OpenMP: an industry standard API for shared-memory programming." *IEEE computational science and engineering* 5, no. 1 (1998): 46-55.
- Internet:
  - https://www.youtube.com/playlist?list=PLLX-Q6B8xqZ8n8bwjGdzBJ25X2utwnoEG
  - http://www.openmp.org/wp-content/uploads/omp-hands-on-SC08.pdf
  - http://extremecomputingtraining.anl.gov/files/2016/08/Mattson_830aug3_HandsOnIntro.pdf

Insper

www.insper.edu.br