Programação Paralela

Introdução

Prof. Carlos Bazilio
http://www.puro.uff.br/bazilio

Departamento de Computação
Instituto de Ciência e Tecnologia
Pólo Universitário de Rio das Ostras
Universidade Federal Fluminense

PURO
UNIVERSIDADE FEDERAL FLUMINENSE
PREFEITURA MUNICIPAL DE RIO DAS OSTRAS

Motivação

- Máquinas paralelas eram arquiteturas raras antigamente
- Este quadro se modificou por, pelo menos, 2 razões:
  - Barateamento dos componentes de hardware
  - Avanço da tecnologia
- Com isso, a computação paralela, atualmente, pode ser considerada como ubíqua (*ubiquitous computing*)

uff PURO
PÓLO UNIVERSITÁRIO DE RIO DAS OSTRAS
UNIVERSIDADE FEDERAL FLUMINENSE
PREFEITURA MUNICIPAL DE RIO DAS OSTRAS

Exemplos de Aplicação

- Animação por computador
  - Animação criada através frames (24 por segundo para filmes) que são resultado da geração de imagens 2D à partir de modelos 3D
  - Toy Story (1995)
    - Primeiro filme completamente criado por computador
    - 100 máquinas com 2 processadores cada
  - Toy Story 2 (1999)
    - Sistema com 1400 processadores
  - Monstros SA (2001)
    - 250 servidores, cada com 14 processadores

uff PURO
PÓLO UNIVERSITÁRIO DE RIO DAS OSTRAS
UNIVERSIDADE FEDERAL FLUMINENSE
PREFEITURA MUNICIPAL DE RIO DAS OSTRAS

Exemplos de Aplicação

- Ciências Biológicas
  - Genoma: descoberta e registro de DNA de organismos em geral
  - Projeto com mais êxito (Celera Corp.) possui 150 servidores com 4 caminhos de execução + 1 servidor com 16 processadores e 64GB de memória
- Projeto SETI@home
  - Procura evidências de inteligência extra-terrestre através da varredura do céu
  - Utiliza computadores ligados a internet como nós processadores

PURO
POLO UNIVERSITÁRIO DE RIO DAS OSTRAS
UNIVERSIDADE FEDERAL FLUMINENSE
PREFEITURA MUNICIPAL DE RIO DAS OSTRAS

Problemas

- Computação em menor tempo tem se descoberto crucial em algumas áreas
- Criar softwares que atendam à estas demandas, usualmente, não é tarefa fácil
- Há escassêz de programadores, analistas e projetistas com experiência em paralelismo
- Estes foram educados a pensar sequencialmente

Exemplo Simples

- Suponha que parte de uma computação envolve a soma de um grande conjunto de valores
- Ao invés destes serem adicionados sequencialmente, podemos:
  - Dividir o conjunto em subconjuntos e distribuí-los entre os processadores
  - Combinar as somas parciais (resultados de cada processador) para obter a soma total

PURO
PÓLO UNIVERSITÁRIO DE RIO DAS OSTRAS
uff
UNIVERSIDADE FEDERAL FLUMINENSE
PREFEITURA MUNICIPAL DE RIO DAS OSTRAS

Exemplo Simples (Soma)

- Qual seria a estratégia adotada e o custo computacional envolvido se:
  - As unidades processadoras tivessem acesso à uma memória compartilhada?
  - As unidades contivessem apenas uma memória local e fossem totalmente interconectadas?
  - As unidades contivessem memórias global (compartilhada) e local (privada)?

uff PURO
PÓLO UNIVERSITÁRIO DE RIO DAS OSTRAS
UNIVERSIDADE FEDERAL FLUMINENSE
PREFEITURA MUNICIPAL DE RIO DAS OSTRAS

Background e Jargão

- Concorrência em Programação Paralela e SOs
  - Concorrência surgiu em SOs com o objetivo de compartilhar ou melhor aproveitar recursos
  - Multitarefa é um exemplo imediato de concorrência
  - SOs ainda oferecem recursos específicos, como *pipes* no UNIX
  - Tratamento de janelas (GUIs) também é inerentemente concorrente
  - Com isso, o SO não precisa identificar concorrência
  - Entretanto, este precisa suportá-la de forma robusta e segura

PURO
PÓLO UNIVERSITÁRIO DE RIO DAS OSTRAS
uff
UNIVERSIDADE FEDERAL FLUMINENSE
PREFEITURA MUNICIPAL DE RIO DAS OSTRAS

Background e Jargão

- Concorrência em Programação Paralela e SOs
  - Isolamento de processos é crítico em SOs
  - Desempenho pode ser sacrificado em respeito à robustez e segurança em SOs
  - Na programação, desempenho é o objetivo principal

PURO
PÓLO UNIVERSITÁRIO DE RIO DAS OSTRAS
UNIVERSIDADE FEDERAL FLUMINENSE
PREFEITURA MUNICIPAL DE RIO DAS OSTRAS

Background e Jargão

- Taxonomia de Flynn
  - SISD
    - Modelo de von Neumann
  - SIMD
    - Um fluxo de instruções e vários de dados
    - Encontrado, principalmente, em máquinas paralelas antigas
  - MISD
    - Formato desconhecido
  - MIMD

PURO
uff
PÓLO UNIVERSITÁRIO DE RIO DAS OSTRAS
UNIVERSIDADE FEDERAL FLUMINENSE
PREFEITURA MUNICIPAL DE RIO DAS OSTRAS

Background e Jargão

- Taxonomia de Flynn
  - MIMD
    - Vários fluxos de instrução e de dados
    - A maioria das arquiteturas paralelas atuais se enquadra nesta classificação
    - Devido à sua popularidade e diversidade, esta classificação possui subclassificações com respeito à organização da memória

PURO
PÓLO UNIVERSITÁRIO DE RIO DAS OSTRAS
UNIVERSIDADE FEDERAL FLUMINENSE
PREFEITURA MUNICIPAL DE RIO DAS OSTRAS

Background e Jargão Subclassificação de MIMD

- Memória Compartilhada
  - Processos compartilham um único endereço de memória
  - Estes se *comunicam* através de *leituras e escritas na memória*
  - Tipos:
    - SMP (Symmetric Multiprocessors) ou UMA (Uniform Memory Access)
      - São mais simples de se programar pois os dados não precisam ser distribuídos
      - Não são facilmente expandidos (escaláveis) devido ao acesso concorrente à memória

uff PURO PÓLO UNIVERSITÁRIO DE RIO DAS OSTRAS
UNIVERSIDADE FEDERAL FLUMINENSE
PREFEITURA MUNICIPAL DE RIO DAS OSTRAS

Background e Jargão Subclassificação de MIMD

- Memória Compartilhada
  - Tipos:
    - NUMA (Nonuniform Memory Access)
      - A memória é divida em blocos
      - Cada bloco pode estar localizado fisicamente mais próximo de um processador, o que torna o acesso à endereços de memória não-constante
    - ccNUMA (cache coherent NUMA)
      - Cada processador possui uma cache de dados, de forma a minimizar o tempo de acesso à memória
      - Estas arquiteturas também prevêem a existência de um protocolo para manter a coerência entre as caches

PURO
uff
PÓLO UNIVERSITÁRIO DE RIO DAS OSTRAS
UNIVERSIDADE FEDERAL FLUMINENSE
PREFEITURA MUNICIPAL DE RIO DAS OSTRAS

Background e Jargão Subclassificação de MIMD

- Memória Distribuída
  - Cada processador possui seu espaço de endereçamento próprio
  - A *comunicação* entre estes processadores é através de *troca de mensagens*
  - A velocidade da comunicação dependerá da topologia da arquitetura paralela, podendo ser de muito rápida (supercomputadores totalmente integrados) a muito lenta (cluster de PCs numa rede Ethernet)
  - O programador precisa programar toda a comunicação, assim como a distribuição de dados

PURO
UNIVERSIDADE FEDERAL FLUMINENSE
PREFEITURA MUNICIPAL DE RIO DAS OSTRAS

Background e Jargão Subclassificação de MIMD

- Memória Distribuída
  - Tipos
    - MPP (Massively Parallel Processors): toda a infra-estrutura de rede é fortemente conectada e especializada para o uso do sistema paralelo
    - Cluster: são computadores (PCs, em sua maioria) completos conectados numa rede; tem se tornado comuns e bastante poderosos dado seu baixo custo
- Sistemas Híbridos
  - Clusters de nós com espaço de memória separado, onde cada nó contém diversos processadores que compartilham sua memória

Background e Jargão
Subclassificação de MIMD

- Grids
  - São sistemas que utilizam recursos distribuídos e heterogêneos através de LANs e/ou WANs
  - Estes têm utilizado, mais comumente, a internet como rede de interconexão
  - Foram pensados, inicialmente, como uma forma de interconectar supercomputadores para a resolução de problemas ainda maiores
  - Uma das diferenças entre Grids e Clusters é que estes necessitam de um ponto comum de administração

uff PURO
PÓLO UNIVERSITÁRIO DE RIO DAS OSTRAS
UNIVERSIDADE FEDERAL FLUMINENSE
PREFEITURA MUNICIPAL DE RIO DAS OSTRAS

Jargão de Programação Paralela

- Tarefa
  - Conceito lógico
- Unidade de Execução
  - Processo e Thread
- Elemento Processador
- Balanceamento de Carga
- Granularidade

PURO
PÓLO UNIVERSITÁRIO DE RIO DAS OSTRAS
UNIVERSIDADE FEDERAL FLUMINENSE
PREFEITURA MUNICIPAL DE RIO DAS OSTRAS

Jargão de Programação Paralela

- Sincronização
- Síncrono x Assíncrono
- Condições de Corrida
- Deadlocks / Livelocks

uff PURO
PÓLO UNIVERSITÁRIO DE RIO DAS OSTRAS
UNIVERSIDADE FEDERAL FLUMINENSE
PREFEITURA MUNICIPAL DE RIO DAS OSTRAS

Análise Quantitativa

- Tempo de execução de um programa numa máquina sequencial:

$$\text{Ttot}(1) = \text{Tconfig} + \text{Tcomput}(1) + \text{Tfinaliz}$$

- Se consideramos uma máquina paralela (P processadores):

$$\text{Ttot}(P) = \text{Tconfig} + \text{Tcomput}(1)/P + \text{Tfinaliz}$$

- Apesar de ideal, as equações acima refletem uma situação realística:
  - Existem porções sequenciais que nunca serão paralelizáveis, enquanto outras sempre poderão executar mais rápido (adição de processadores)

PURO
UNIVERSIDADE FEDERAL FLUMINENSE

Análise Quantitativa

- Medida de análise da adição de processadores é o *speedup*

$$S(P) = Ttotal(1) / Ttotal(P)$$

- A medida de eficiência *E* é o speedup normalizado pelo número de processadores

$$E(P) = S(P) / P = Ttotal(1) / (P * Ttotal(P))$$

- Idealmente, o que gostaríamos era de ter um *speedup* linear perfeito; Entretanto, normalmente, o tempo de configuração e finalização não são diminuídos com a adição de processadores

PURO
PÓLO UNIVERSITÁRIO DE RIO DAS OSTRAS
UNIVERSIDADE FEDERAL FLUMINENSE
PREFEITURA MUNICIPAL DE RIO DAS OSTRAS

Análise Quantitativa

- A fração do tempo sequencial é dada por:

Tserial = (Tconfig + Tfinaliz) / Ttotal(1)

- Naturalmente, a fração paralela é *(1 – Tserial)*
- Reescrevendo a fórmula para *Ttotal(P),* temos:

*Ttotal(P)=Tserial*Ttotal(1)+(1-Tserial)*Ttotal(1)/P*

- Substituindo em *S(P)*, temos:

S(P) = 1 / (Tserial + (1-Tserial)/P)

- Esta é a famosa lei Amdahl's!

PURO
PÓLO UNIVERSITÁRIO DE RIO DAS OSTRAS
UNIVERSIDADE FEDERAL FLUMINENSE
PREFEITURA MUNICIPAL DE RIO DAS OSTRAS

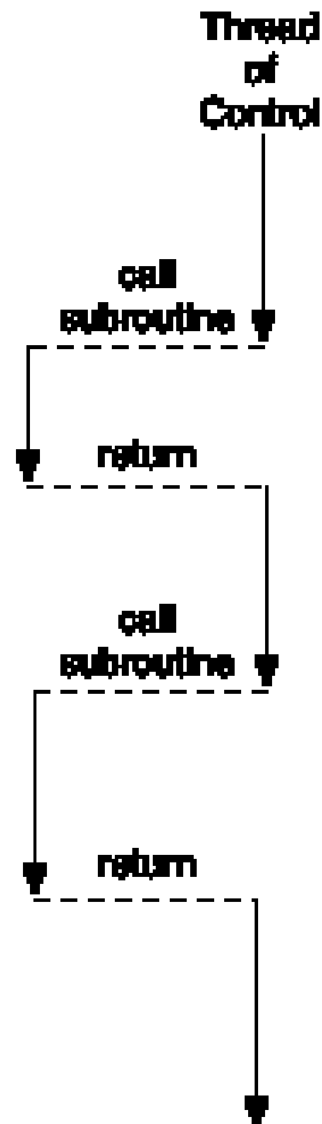
Linguagens para Desenvolvimento Paralelo / Distribuído

- OpenMP
- MPI
- Threads (java.lang.Thread, PThread, Win32, C#, …)
- CUDA
- ...

PURO
UNIVERSIDADE FEDERAL FLUMINENSE
PREFEITURA MUNICIPAL DE RIO DAS OSTRAS

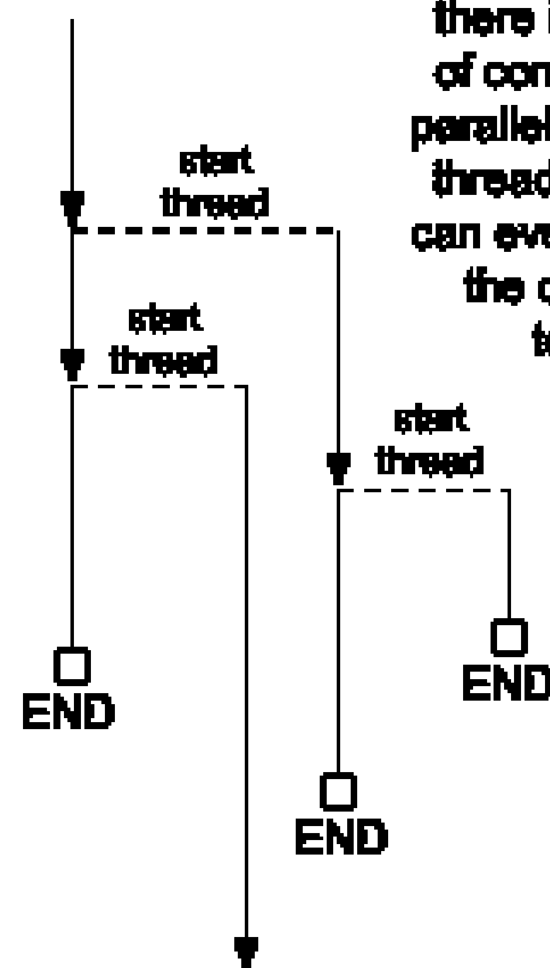
Threads

When a thread calls a subroutine, there is still only one thread of control, which is in the subroutine for a time, then returns to the thread that called the subroutine.



Time

Thread of Control



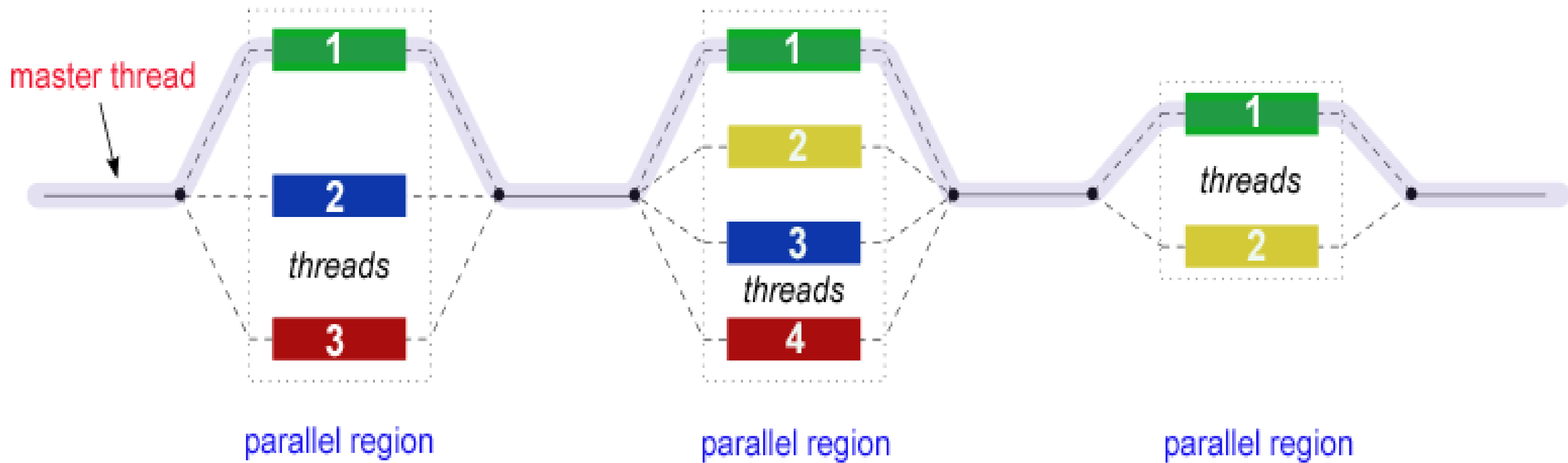
When a thread starts another thread, there is a new thread of control that runs in parallel with the original thread of control, and can even continue after the original thread terminates.

Java Threads – Hello World

```
public class ExampleThread extends Thread {
    private String name;
    private String text;
    private final int REPEATS = 5;
    private final int DELAY = 200;

    public ExampleThread( String aName, String aText ) {
        name = aName;
        text = aText;
    }

    public void run() {
        try {
            for ( int i = 0; i < REPEATS; ++i ) {
                System.out.println( name + " says \"" +
text + "\"");
                Thread.sleep( DELAY );
            }
        }
        catch( InterruptedException exception ) {
            System.out.println( "An error occured in " +
name );
        }
        Finally {
            // Clean up, if necessary
            System.out.println( name + " is quiting..." );
```

```
public class ThreadTest {
    public static void
main( String[] args )
    {
        Thread et1 = new
ExampleThread( "Thread #1", "Hello
World!" );
        Thread et2 = new
ExampleThread( "Thread #2", "Hey
Earth!" );

        et1.start();
        et2.start();

        // et1.interrupt();
    }
}
```

OpenMP



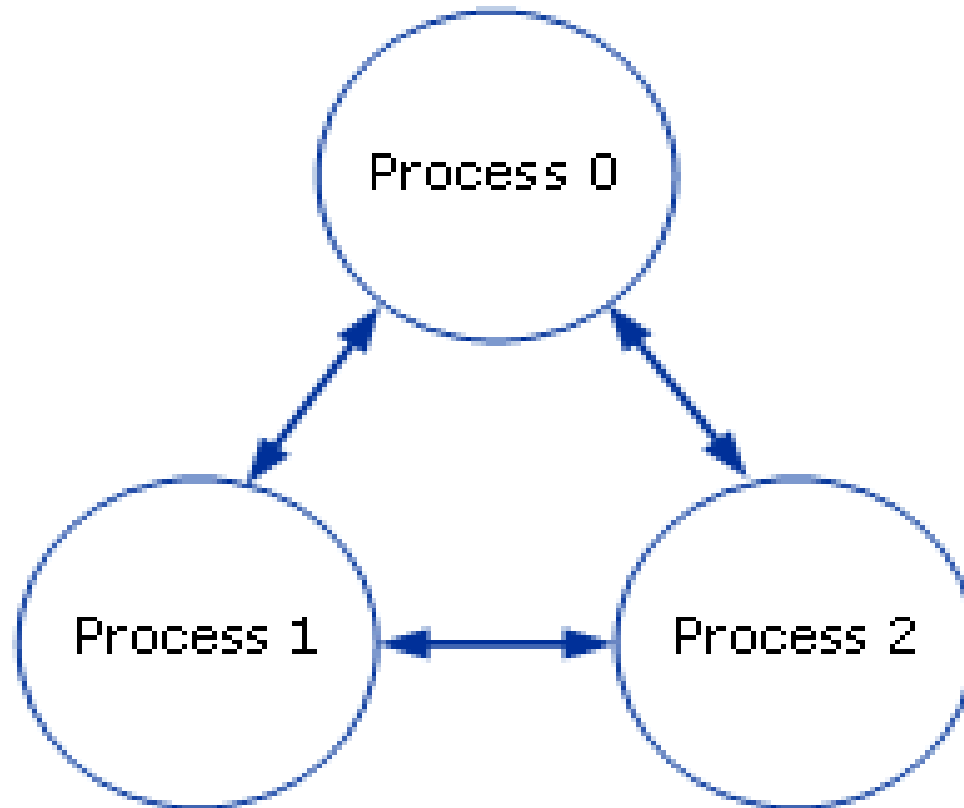
uff PURO
PÓLO UNIVERSITÁRIO DE RIO DAS OSTRAS
UNIVERSIDADE FEDERAL FLUMINENSE
PREFEITURA MUNICIPAL DE RIO DAS OSTRAS

OpenMP – Hello World

```
#include <omp.h>
#include <stdio.h>

int main (int argc, char *argv[]) {
  int th_id, nthreads;
  #pragma omp parallel private(th_id)
  {
    th_id = omp_get_thread_num();
    printf("Hello World from thread %d\n", th_id);
    #pragma omp barrier
    if ( th_id == 0 ) {
      nthreads = omp_get_num_threads();
      printf("There are %d threads\n",nthreads);
    }
  }
  return 0;
}
```

PURO
uff
PÓLO UNIVERSITÁRIO DE RIO DAS OSTRAS
UNIVERSIDADE FEDERAL FLUMINENSE
PREFEITURA MUNICIPAL DE RIO DAS OSTRAS

MPI



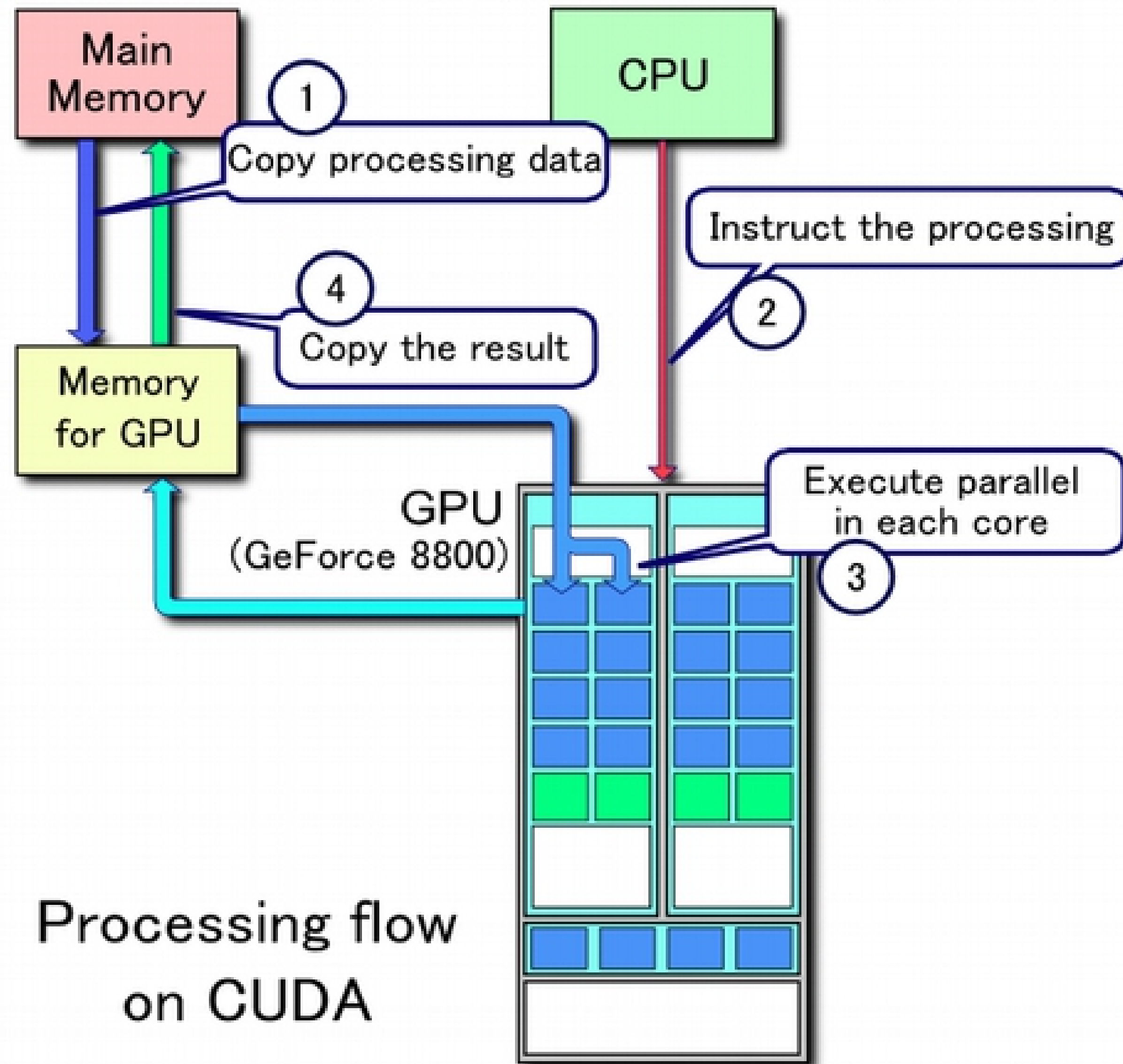
PURO
PÓLO UNIVERSITÁRIO DE RIO DAS OSTRAS
UNIVERSIDADE FEDERAL FLUMINENSE
PREFEITURA MUNICIPAL DE RIO DAS OSTRAS

MPI – Hello World

```
/* C Example */
#include <stdio.h>
#include <mpi.h>
int main (argc, argv)
     int argc;
     char *argv[];
{
  int rank, size,
  MPI_Init (&argc, &argv);      /* starts MPI */
  MPI_Comm_rank (MPI_COMM_WORLD, &rank);        /* get current
process id */
  MPI_Comm_size (MPI_COMM_WORLD, &size);        /* get number of
processes */
  printf( "Hello world from process %d of %d\n", rank, size );
  MPI_Finalize();
  return 0;
}
```

PURO
PÓLO UNIVERSITÁRIO DE RIO DAS OSTRAS
UNIVERSIDADE FEDERAL FLUMINENSE
PREFEITURA MUNICIPAL DE RIO DAS OSTRAS

CUDA



PURO
UFF
PÓLO UNIVERSITÁRIO DE RIO DAS OSTRAS
UNIVERSIDADE FEDERAL FLUMINENSE
PREFEITURA MUNICIPAL DE RIO DAS OSTRAS

CUDA – Hello World

```
// This is the REAL "hello world" for CUDA!
// It takes the string "Hello ", prints it, then passes
it to CUDA with an array
// of offsets. Then the offsets are added in
parallel to produce the string "World!"
// By Ingemar Ragnemalm 2010

#include <stdio.h>

const int N = 7;
const int blocksize = 7;

__global__
void hello(char *a, int *b) {
 a[threadIdx.x] += b[threadIdx.x];
}

int main() {
 char a[N] = "Hello ";
 int b[N] = {15, 10, 6, 0, -11, 1, 0};

 char *ad;
 int *bd;
 const int csize = N*sizeof(char);
 const int isize = N*sizeof(int);

 printf("%s", a);
 cudaMalloc( (void**)&ad, csize );
 cudaMalloc( (void**)&bd, isize );
 cudaMemcpy( ad, a, csize,
cudaMemcpyHostToDevice );
 cudaMemcpy( bd, b, isize,
cudaMemcpyHostToDevice );

 dim3 dimBlock( blocksize, 1 );
 dim3 dimGrid( 1, 1 );
 hello<<<dimGrid, dimBlock>>>(ad, bd);
 cudaMemcpy( a, ad, csize,
cudaMemcpyDeviceToHost );
 cudaFree( ad );

 printf("%s\n", a);
 return EXIT_SUCCESS;
}
```

Links Interessantes

- http://delicious.com/carlosbazilio/parallelism
  - Meus favoritos
- http://www.top500.org/
  - Lista dos top 500 supercomputadores

PURO
PÓLO UNIVERSITÁRIO DE RIO DAS OSTRAS
UNIVERSIDADE FEDERAL FLUMINENSE
PREFEITURA MUNICIPAL DE RIO DAS OSTRAS