CAP-387(2016) – Tópicos Especiais em Computação Aplicada: Construção de Aplicações Massivamente Paralelas

Aula 41: Linguagens PGAS

Celso L. Mendes, Stephan Stephany

LAC / INPE

Emails: celso.mendes@inpe.br, stephan.stephany@inpe.br

Modelo PGAS

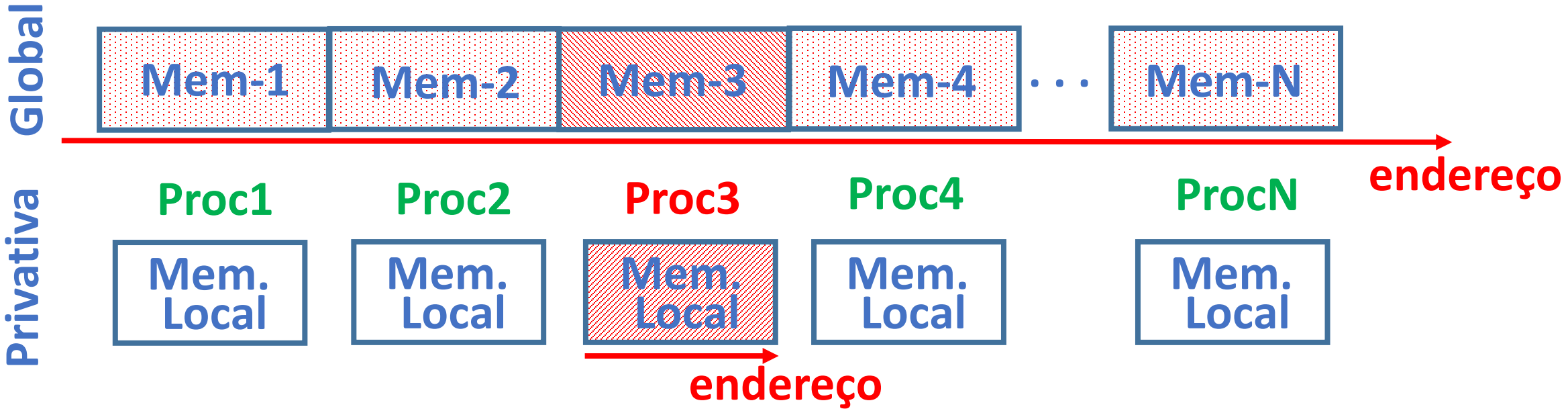
- PGAS: modelo de programação paralela
  - *Partitioned Global Address Space*
  - Oferece a cada processador a *ilusão* de memória global
  - Combina estilo SPMD (como em MPI) com a semântica de programação em memória compartilhada (acesso uniforme a todas as variáveis)
- Objetivos:
  - Elevar o nível de abstração da programação em sistemas massivamente paralelos
  - Evitar ter que lidar com troca de mensagens no programa
- Implementações: Linguagens PGAS
  - CoArray Fortran, Unified Parallel C (UPC), Chapel, X10, Fortress, Global Arrays, etc, etc

Modelo PGAS

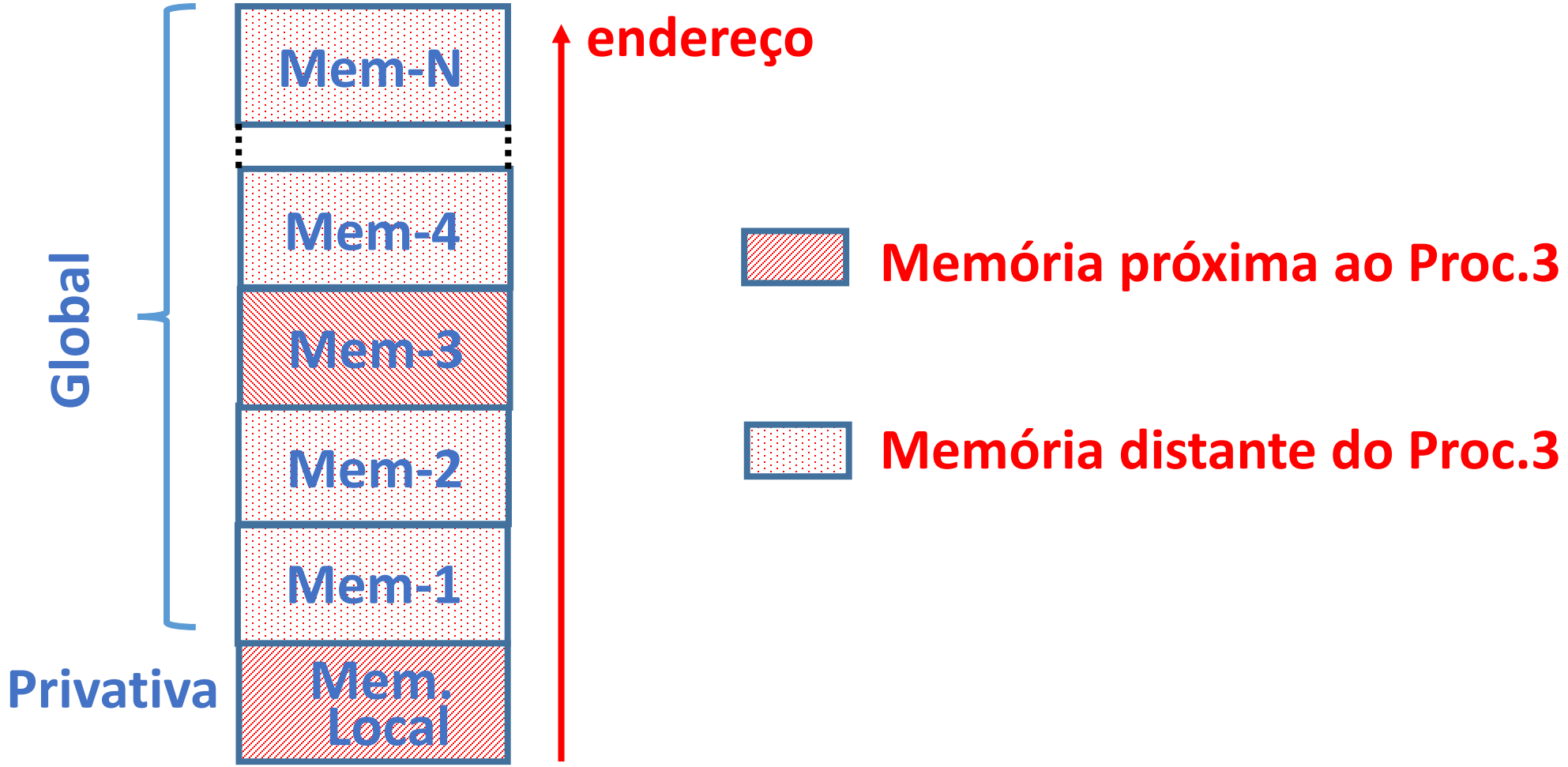
- Princípios:
  - Todas as memórias de todos os nós fazem parte do espaço de endereços de cada processador
  - Para cada processador, uma faixa de endereços corresponde à memória do nó local (acesso mais rápido), as outras faixas são memórias de nós remotos (lentas)
  - Para um dado processador ProcK:
    - Mem-K = memória local (acesso rápido)
    - Mem-j (j≠K) = memórias remotas (acessos lentos)

Modelo PGAS

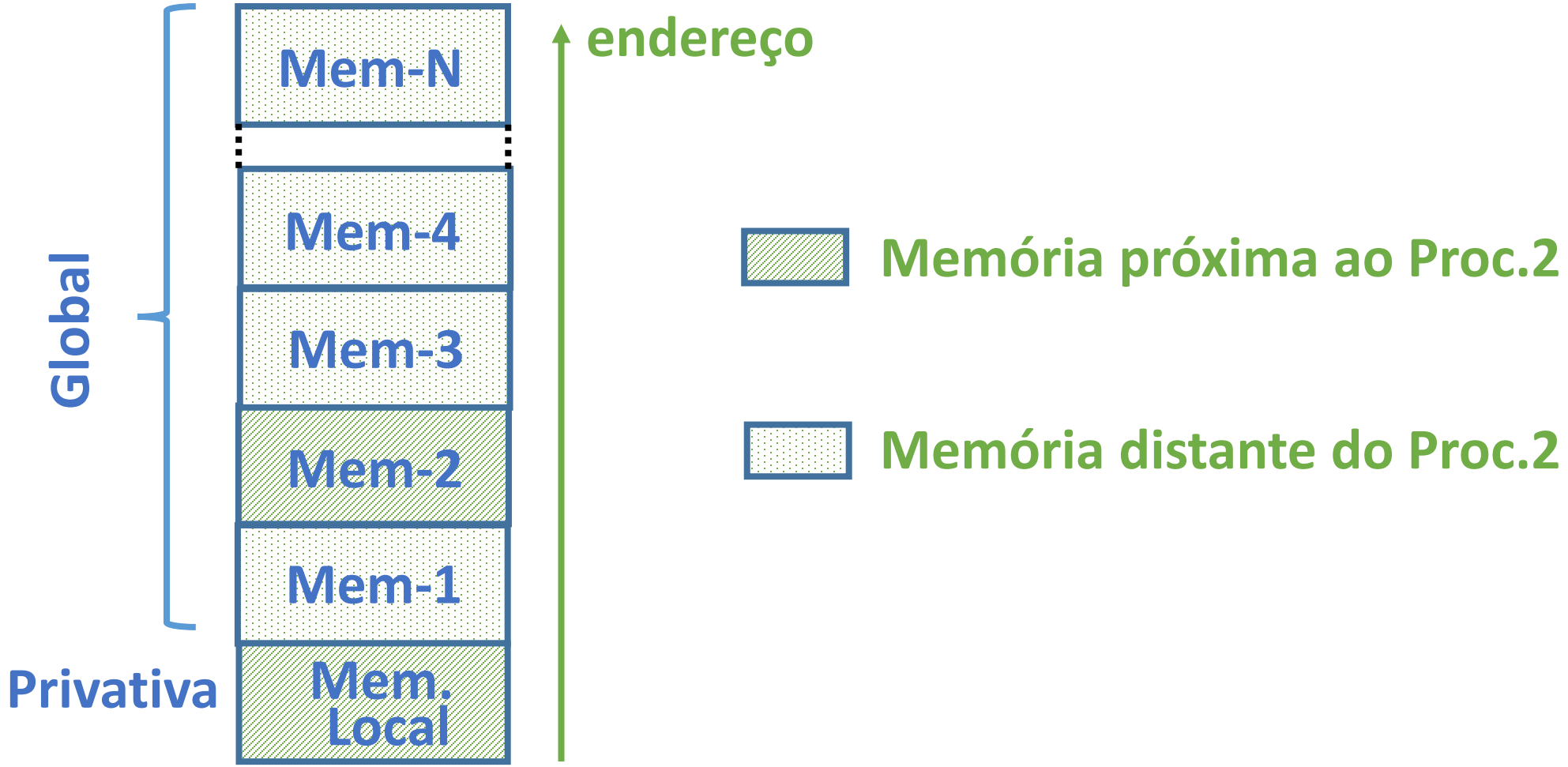
Visão Lógica: Mem. {Global,Compartilhada} ou {Privativa,Local}



Visão da Memória – Proc.3



Visão da Memória – Proc.2



Linguagens PGAS: UPC

- UPC: Unified Parallel C
  - Criada em 1999, v.1.0 em 2001; hoje: v.1.3
  - Extensão da linguagem C para Proc. Alto Desempenho
  - Várias implementações, abertas ou comerciais
  - Disponível em máquinas de memória compartilhada ou distribuída
  - UPC Consortium: governo, universidades, indústria
  - Ref: https://upc-lang.org/
    - Documentação, publicações, etc.

UPC: Modelo de Execução

- Coleção de *threads* executando em modo SPMD
  - Número total de threads: THREADS
  - Índices dos threads: 0,1,…,THREADS-1
  - Índice do thread local a um processador: MYTHREAD
  - Implementação: *thread* pode ser um processo
- Sincronização:
  - Barreiras, locks, controle de consistência de memória
- Ponteiros:
  - Podem apontar para área privativa ou compartilhada

UPC: Localização de Variáveis

- Vetores compartilhados:
  - *shared int x[THREADS];* um elemento por thread
  - *shared int y[10][THREADS];* 10 elementos por thread
- Escalares:
  - *shared int a;* escalar na área compartilhada do thread-0
  - *int b*; escalar na área privativa – um em cada thread
- Ponteiros:
  - *shared int *p*; ponteiro para a área compartilhada

UPC: Loops Paralelos

- *upc_forall*: extensão de *for* em C

  *upc_forall (init; condition; increment; **affinity**) { …}*
  - *init, condition, increment:* análogos ao caso de C
  - ***affinity**:* declaração de "afinidade" das iterações; controla a execução do corpo do loop. Duas formas possíveis:
    - *affinity=inteiro:* loop é equivalente a

      for (…) if (MYTHREAD==(i%THREADS)) {<corpo-do-loop>}

      → iteração *i* é executada pelo thread *i%THREADS*
    - *affinity=endereço:* loop é equivalente a

      for (…) if (MYTHREAD==upc_affinity(endereço) {<corpo-loop>}

      → *iteração i é executada pelo thread que armazena o endereço*

UPC: Exemplo 1

```
#include <stdio.h>
#include <upc.h>
#define NperTHREAD 4
#define SIZE  (NperTHREAD * THREADS)
main ()
{
  int i;
  // "affinity" is an int so it is (i mod THREADS)
  upc_forall( i=0;                    // Init field
         i< SIZE;                     //Condition field
         i++, printf(">%d %d\n", MYTHREAD, i); //Increment field
         i ){                         // Affinity field
        printf("MYTHREAD is %d and I am on iteration %d\n", MYTHREAD, i);
  }
}
```

← Corpo do loop

UPC: Exemplo 1 (cont.)

- Compilação e execução num Cray-XE6:
  - *cc –h upc –o prog1 prog1.c*
    → Utiliza compilador Cray, com flag "*–h upc*" para UPC
  - *aprun –n 4 –N 2 prog1*
    → Solicita execução em 4 processadores, 2 em cada nó
    De acordo com o código-fonte, esta execução terá:
    THREADS = 4 (número de processadores em uso)
    *SIZE = (4*4) = 16*

UPC: Exemplo 1 (saída)

```
>1 1
MYTHREAD is 0 and I am on iteration 0
MYTHREAD is 1 and I am on iteration 1
>0 1
>1 2
>0 2
>1 3
>0 3
>1 4
>0 4
>1 5
MYTHREAD is 0 and I am on iteration 4
MYTHREAD is 1 and I am on iteration 5
>0 5
>1 6
>0 6
>1 7
>0 7
>1 8
>0 8
>1 9
MYTHREAD is 0 and I am on iteration 8
MYTHREAD is 1 and I am on iteration 9
>0 9
>1 10
>0 10
>1 11
```

```
>0 11
>1 12
>0 12
>1 13
MYTHREAD is 0 and I am on iteration 12
MYTHREAD is 1 and I am on iteration 13
>0 13
>1 14
>0 14
>1 15
>0 15
>1 16
>0 16
>2 1
>3 1
>2 2
>3 2
MYTHREAD is 2 and I am on iteration 2
>3 3
>2 3
MYTHREAD is 3 and I am on iteration 3
>2 4
>3 4
>2 5
>3 5
>2 6
>3 6
```

```
MYTHREAD is 2 and I am on iteration 6
>3 7
>2 7
MYTHREAD is 3 and I am on iteration 7
>2 8
>3 8
>2 9
>3 9
>2 10
>3 10
MYTHREAD is 2 and I am on iteration 10
>3 11
>2 11
MYTHREAD is 3 and I am on iteration 11
>2 12
>3 12
>2 13
>3 13
>2 14
>3 14
MYTHREAD is 2 and I am on iteration 14
>3 15
>2 15
MYTHREAD is 3 and I am on iteration 15
>2 16
>3 16
```

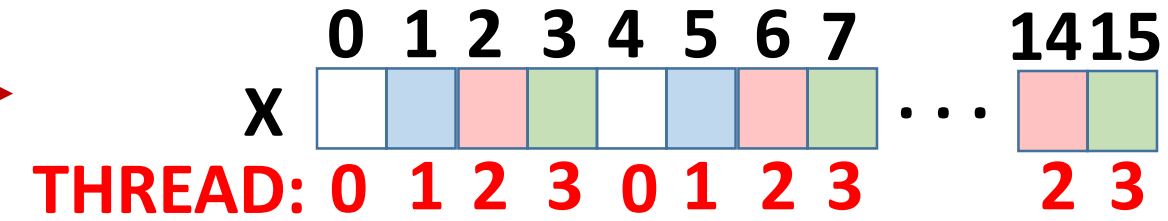
Todos os threads iniciam todas as iterações:
i++ = 1,2,3,...,16

Cada thread só executa o corpo do loop (*printf("MYTHREAD is...")*) nas iterações onde *MYTHREAD==i%THREADS*

UPC: Exemplo 2

```
#include <stdio.h>
#include <upc.h>
#define NperTHREAD 4
#define SIZE  (NperTHREAD * THREADS)
shared int x[SIZE];
main ()
{
  int i;
  // "affinity" is an address
  upc_forall( i=0;                    // Init field
         i< SIZE;                     //Condition field
         i++, printf(">%d %d\n", MYTHREAD, i); //Increment field
         &x[i] ){                     // Affinity field
        printf("MYTHREAD is %d and I have affinity to x[%d]\n", MYTHREAD, i);
  }
}
```

Cenário com THREADS=4, SIZE=16:



UPC: Exemplo 2 (saída)

```
MYTHREAD is 0 and I have affinity to x[0]
>1 1
>0 1
MYTHREAD is 1 and I have affinity to x[1]
>0 2
>1 2
>0 3
>1 3
>0 4
>1 4
MYTHREAD is 0 and I have affinity to x[4]
>1 5
>0 5
MYTHREAD is 1 and I have affinity to x[5]
>0 6
>1 6
>0 7
>1 7
>0 8
>1 8
MYTHREAD is 0 and I have affinity to x[8]
>1 9
>0 9
MYTHREAD is 1 and I have affinity to x[9]
>0 10
>1 10
>0 11
```

```
>1 11
>0 12
>1 12
MYTHREAD is 0 and I have affinity to x[12]
>1 13
>0 13
MYTHREAD is 1 and I have affinity to x[13]
>0 14
>1 14
>0 15
>1 15
>0 16
>1 16
>3 1
>2 1
>3 2
>2 2
>3 3
MYTHREAD is 2 and I have affinity to x[2]
MYTHREAD is 3 and I have affinity to x[3]
>2 3
>3 4
>2 4
>3 5
>2 5
>3 6
>2 6
```

```
>3 7
MYTHREAD is 2 and I have affinity to x[6]
MYTHREAD is 3 and I have affinity to x[7]
>2 7
>3 8
>2 8
>3 9
>2 9
>3 10
>2 10
>3 11
MYTHREAD is 2 and I have affinity to x[10]
MYTHREAD is 3 and I have affinity to x[11]
>2 11
>3 12
>2 12
>3 13
>2 13
>3 14
>2 14
>3 15
MYTHREAD is 2 and I have affinity to x[14]
MYTHREAD is 3 and I have affinity to x[15]
>2 15
>3 16
>2 16
```

UPC: Exemplo 3

- Adição de dois vetores:

```
#include <upc.h>
#define N 100*THREADS
shared int v1[N], v2[N], v1plusv2[N];
void main(){
        int i;
        for(i=0; i<N; i++)
                if (MYTHREAD==i%THREADS)
                        v1plusv2[i]=v1[i]+v2[i];
}
```

UPC: Exemplo 4

- Outra forma para a adição de dois vetores:

```
#include <upc.h>
#define N 100*THREADS
shared int v1[N], v2[N], v1plusv2[N];
void main()
{
        int i;
        upc_forall(i=0; i<N; i++; i)
                v1plusv2[i]=v1[i]+v2[i];
}
```

upc_forall: Loop com *afinidade* a *i*
→ iteração *i* só será efetivada onde
MYTHREAD=i%THREADS

UPC: Exemplo 5

- Operação de redução em UPC:

```
#include <stdio.h>
#include <upc.h>
#include <upc_collective.h>
#define BLK_SIZE 1
#define NELEMS 1
shared [BLK_SIZE] long A[NELEMS*THREADS];
shared long result;
int main()
{
      int i;
      if(0 == MYTHREAD){
            for (i=0 ; i<NELEMS*THREADS; i++)
                  A[i] = ((i%2)==1) ? i : i * (-1);
      }
      upc_barrier;
      upc_all_reduceL( &result, A, UPC_ADD, NELEMS*THREADS, BLK_SIZE, NULL, UPC_IN_NOSYNC | UPC_OUT_NOSYNC );
      upc_barrier;
      if(0 == MYTHREAD) printf("UPC_size: %d  result: %ld \n",THREADS,result);
      return 0;
}
```

Inicialização do array *A*:

- ***A[i]=i* se *i* for ímpar**
- ***A[i]=-i* se *i* for par**

A: {0,1,-2,3,-4,5,-6,...,NELEMS*THREADS}

UPC: Exemplo 5 (cont.)

- Compilação e execução num Cray-XE6:
  - *cc –h upc –o prog prog.c*
    → Utiliza compilador Cray, com flag "*–h upc*" para UPC
  - *aprun –n 1024 prog*
    → Solicita execução em 1024 processadores, com 32 nós do XE6 (32 núcleos por nó)

    De acordo com o código-fonte, esta execução terá:
    THREADS = 1024 (número de processadores em uso)
    Conteúdo do array *A: 0, 1, -2, 3, -4, 5, …, -1022, 1023*
- Saída da execução: soma dos elementos de *A*

  UPC_size: 1024 result: 512