Algoritmo da Soma no modelo CGM

Algoritmo da Soma no modelo CGM

- Entrada: n elementos $v(1), v(2), \cdots, v(n)$
- Saída: $S = v(1) + v(2) + \cdots + v(n)$

  (1) p processadores

  (2) Cada processador recebe n/p elementos,
  efetua a soma localmente e envia o resultado para p_1

  (3) p_1 efetua a soma de $S = S_1 + S_2 + \cdots + S_p$

Algoritmo da Soma no modelo CGM

- Entrada: número do processador i;
  número p de processadores; $B = A((i-1)r + 1 : r)$;
  $r = n/p$
- Saída: p_i calcula $z = B(1) + B(2) + \cdots + B(n)$ e envia o resultado para p_1. p_1 calcula $S = z_1 + z_2 + \cdots + z_n$

(1) $z = B(1) + B(2) + \cdots + B(n)$

(2) se $i = 1$ então $S = z$
  caso contrário $envia(z, p_1)$

(3) se $i = 1$ então
  para $i = 2$ até p faça
    $receba(s[i], p)$

(4) $S = S_1 + S_2 + \cdots + S_p$

Algoritmo da Soma no modelo CGM

complexidade:

Passo 1: cada p_i efetua r operações

Passo 2: p_1 efetua uma operação e os demais p_i's enviam uma msg.

Passo 3: p_1 recebe $p-1$ mensagens.

Passo 4: p_1 efetua $p-1$ operações.

Um rodada de comunicação.

tempo : $O(n/p)$

Soma no modelo CGM - implementação

```
#include<mpi.h>

  // Passo 1. Inicializar
  // Passo 2. Envie os dados para as tarefas diferente de 0
  // Passo 3. Receba uma msg da tarefa 0
  // Passo 4. Calcule a soma
  // Passo 5. Receba as somas parciais
  // Passo 6. Imprima os valores da soma
  // Passo 7. Finalize o MPI
```

Passo 1

```
// Passo 1. Inicializar
MPI_init(&argc, &argv);
MPI_Comm_size(MPI_COMM_WORLD, &size);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);

tam = (int)TAMMAX/size;      // tamanho do subvetor
```

Passo 2 - Enviando dados

```
// Passo 2. Envie os dados para as tarefas diferente de 0
if (rank == 0){
   for (i = 1; i < size; i++)  {
      for (j = 0; j < tam; j++) {
         subvetor[j] = vetor_dados[tam * i + j];
      }
      MPI_Send(subvetor, tam, MPI_INT, i, tag, MPI_COMM_WORLD);
   } // fim for

   for (j = 0; j < tam; j++) { // vetor da tarefa 0;
      subvetor[j] = vetor_dados[j];
   }
}
```

Passo 3 - Recebendo dados

```
// Passo 3. Receba uma msg da tarefa 0
if (rank != 0) {
   MPI_Recv(subvetor, tam, MPI_INT, 0, tag,
                           MPI_COMM_WORLD, &status);
}
```

Passo 4 - Efetuando a soma

```
// Passo 4. Calcule a soma
 for(i = 0; i < tam; i++) {
    somap = somap + subvetor[i];
 }
```

Passo 5 - Recebendo somas parciais

```
// Passo 5. Receba as somas parciais
MPI_Reduce(&somap, &soma, 1, MPI_INT,
                             MPI_SUM, 0, MPI_COMM_WORLD);
```

Passo 6 e 7 - Finalizando

```
// Passo 6. Imprima os valores da soma
if (rank == 0) {
   printf("soma: %d\n", soma);
}


// Passo 7. Finalize o MPI
MPI_Finalize();
return 0;
```

Compilação e Execução

Para compilar

```
$ mpicc soma.c -o soma
```

Para executar

```
$ mpirun -np 4 soma
```

Soma de prefixos

Soma de prefixos

Dado um vetor A de n elementos $A[0], A[1], \cdots, A[n-1]$ a computação de prefixos calcula os valores:

```
A[0]
A[0] op A[1]
A[0] op A[1] op A[2]
...
A[0] op ... op A[n-1]
```

onde op é uma operação binária associativa.

Soma de prefixos

- Exemplo: op é adição
- Vetor de entrada:

$$[3, 7, 5, 1, 2, 4, 0, 9]$$

- Vetor de saída:

$$[3, 10, 15, 16, 18, 22, 22, 31]$$

Soma de prefixos - CGM

- Entrada: n elementos $v(1), v(2), \cdots, v(n)$
- Saída: $S(k) = v(1) + \cdots + v(k)$, $1 \leq k \leq n$
  - (1) p_0 envia n/p elementos para cada p_i
  - (2) Cada p_i calcula a soma $T(i)$ de seus elementos e envia o resultado para os processadores $j > i$
  - (3) p_0 calcula a soma $ST(i) = T(1) + \cdots + T(i)$, $1 \leq i \leq p$
  - (4) Cada p_i calcula a soma de prefixos local
    - $S(k) = ST(i-1) + v((i-1) * r + 1) + \cdots + v(k)$,
    - $(i-1) * r + 1 \leq k \leq i * r$, $r = n/p$

Soma de prefixos - implementação

```
#include <mpi.h>


  // passo 1. Inicialização
  MPI_init(&argc, &argv);
  MPI_Comm_size(MPI_COMM_WORLD, &size);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);

  tam = (int)TAMMAX/size;       // tamanho do subvetor
```

Soma de prefixos - implementação

```
// passo 2. Envia os dados as tarefas filhos
MPI_Scatter(vetor_dados, tam, MPI_INT, subvetor, tam,
                              MPI_INT, root, MPI_COMM_WORLD);


// passo 3. Calcula a soma em cada tarefa
for (i = 0; i < tam; i++)
   somap = somap + subvetor[i];
```

Soma de prefixos - implementação

```
// passo 4. Receba as somas parciais
MPI_Scan(&somap, &soma, 1, MPI_INT, MPI_SUM, MPI_COMM_WORLD);


// passo 5. Calcula as somas parciais em cada tarefa
soma_pre[0] = soma - somap + subvetor[0]
for (i = 1; i < tam; i++)
   soma_pre[i] = soma_pre[i-1] + subvetor[i];
```

Soma de prefixos - implementação

```
// passo 6. Imprima o valor da soma
for (i = 0; i < tam; i++)
   printf("rank %d e soma_pre[%d] = %d\n", rank, i, soma_pre[i]);



// passo 7. Finalize o MPI
MPI_Finalize();
return 0;
```

Fim