ALGORÍTMOS PARALELOS
(Aula 3)

Neyval C. Reis Jr.

OUTUBRO/2004

Laboratório de Computação de Alto Desempenho

DI/UFES

UFES
UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

Programa do Curso

1. Introdução
2. Arquitetura de Computadores
3. Arquiteturas de Sistemas Paralelos
4. Computação de Alto Desempenho
5. Programação Paralela (modelos e paradigmas)
6. Análise de Desempenho e Instrumentação
7. Aplicações

UFES
UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

Programa do Curso

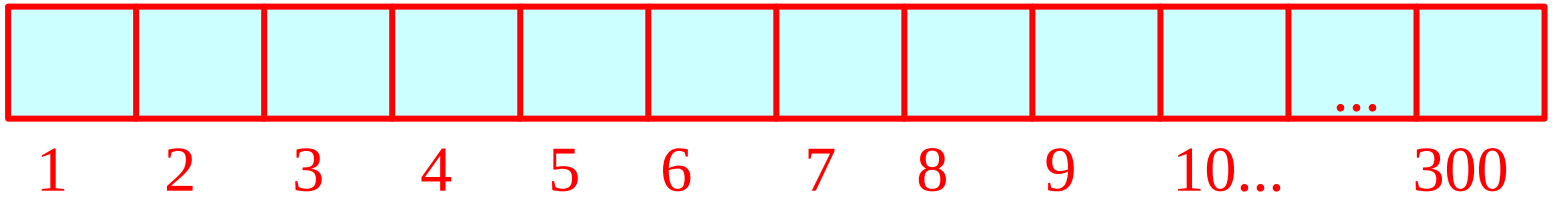
5. Programação Paralela (modelos e paradigmas)
   a) Começando a pensar em paralelo (exemplo)
   b) Metodologia de design
   c) Paradigmas de Programação
   d) Eficiência
   e) Ferramentas

UFES
UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

Começando a pensar em paralelo

Exemplo

Soma dos elementos de um vetor



Poderíamos pensar em uma máquina paralela (com vários processadores) onde cada processador calcularia a soma dos elementos de uma parte do vetor

Solução do problema serial

```
      PROGRAM EXEMPLO
C-----------------------------------------------------
      IMPLICIT NONE
      INTEGER ID
      PARAMETER(ID=300)
      INTEGER VET(ID), I, SOMA
C------------Inicio do programa principal --------------
      SOMA=0
      OPEN (UNIT=10,FILE='b.dat',STATUS='OLD')
      DO I=1, ID
        READ (10,*) VET(I)
      ENDDO
      DO I=1,ID
        SOMA=SOMA+VET(I)
      ENDDO
      WRITE(*,*) 'SOMA DOS ELEMENTOS DO VETOR E:', SOMA
      END
```

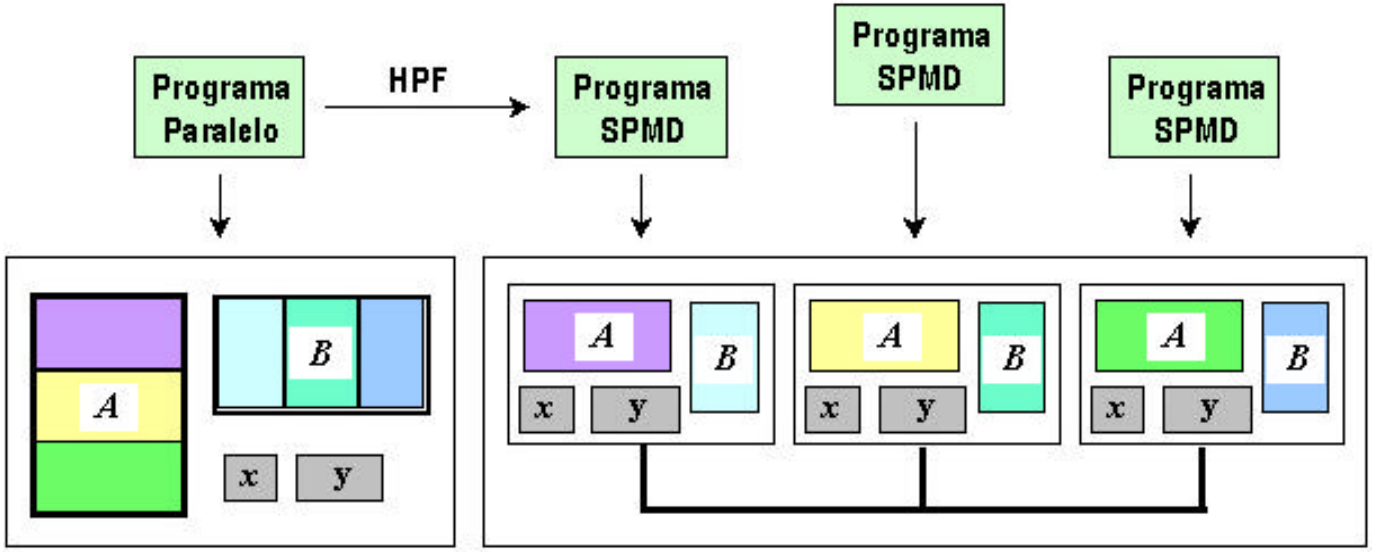
Ambientes de programação paralela

- HPF – High Performance Fortran
- Treadmarks
- MPI – Message Passing Interface

HPF - Características

- Paralelismo Implícito
- Paralelismo mais simples de implementar
- Existem ferramentas que fornecem melhor desempenho

HPF: Modelo SPMD



- O mesmo programa executa em cada um dos processadores (SPMD – Single Program Multiple Data).
- Os vetores são distribuídos entre os processadores.
- A comunicação entre processadores ocorre de maneira transparente para o programador.
- A distribuição de “trabalho” ocorre com base da divisão de dados.

HPF – Implementação paralela

```
      PROGRAM EXEMPLO
C-------------Inicio do programa principal --------------
      IMPLICIT NONE
      INTEGER ID
      PARAMETER(ID=300)
      INTEGER VET(ID), I, SOMA
!hpf DISTRIBUTE VET(BLOCK)
C-------------Inicio do programa principal --------------
      SOMA=0
      OPEN(UNIT=10,FILE='b.dat',STATUS='OLD')
      DO I=1, ID
        READ(10,*) VET(I)
      ENDDO
!hpf independent,reduction(SOMA)
      DO I=1,ID
        SOMA=SOMA+VET(I)
      ENDDO
      WRITE(*,*) 'A SOMA DOS ELEMENTOS DO VETOR E:', SOMA
      END
```

HPF – Implementação paralela

```
      PROGRAM EXEMPLO
C-------------Inicio do programa principal --------------
      IMPLICIT NONE
      INTEGE
      PARAME
      INTEGER VET(ID), I, SOMA
!hpf DISTRIBUTE VET(BLOCK)
C-------------Inicio do programa principal --------------
      SOMA=0
      OPEN(UNIT=10,FILE='b.dat',STATUS='OLD')
      DO I=1, ID
        READ(10,*) VET(I)
      ENDDO
!hpf independent,reduction(SOMA)
      DO I=1,ID
        SOMA=SOMA+VET(I)
      ENDDO
      WRITE(*,*) 'A SOMA DOS ELEMENTOS DO VETOR E:', SOMA
      END
```

!hpf DISTRIBUTE VET(BLOCK)


```
      real, dimension (23) :: A
!hpf$ distribute A (block)
```

HPF – Implementação paralela

```
      PROGRAM EXEMPLO
C-------------Inicio do programa principal --------------
      IMPLICIT NONE
      INTEGER ID
      PARAMETER(ID=300)
      INTEGER VET(ID), I, SOMA
!hpf DISTRIBUTE VET(BLOCK)
C-------------Inicio do programa principal --------------
      SOMA=0
      OPEN(UNIT=10,FILE='b.dat',STATUS='OLD')
      DO I=1, ID
        READ(10,*) VET(I)
      ENDDO
!hpf independent,reduction(SOMA)
      DO I=1,ID
        SOMA=SOMA+VET(I)
      ENDDO
      WRITE(*,*) 'A SOMA DOS ELEMENTOS DO VETOR E:', SOMA
      END
```

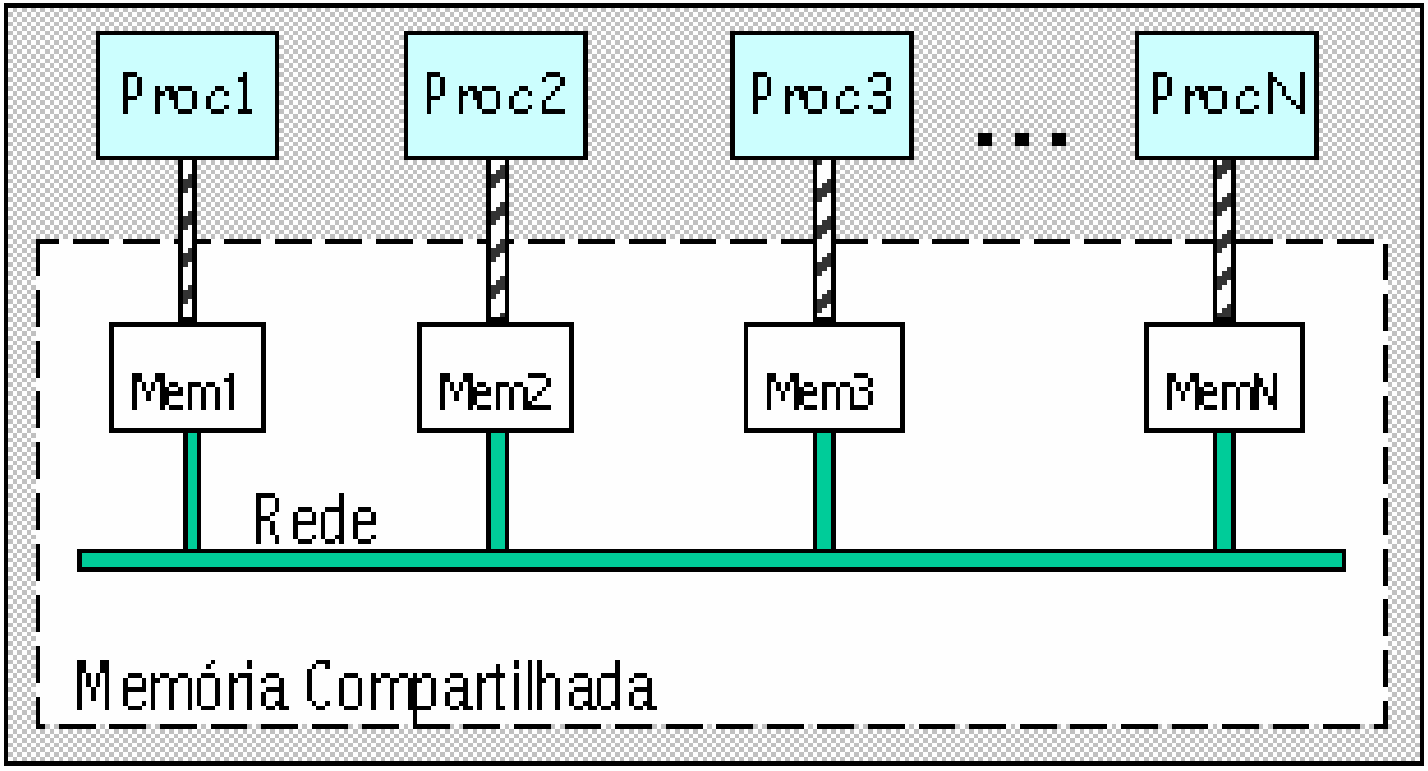
HPF – Implementação paralela

```
      PROGRAM EXEMPLO
C-------------Inicio do programa principal --------------
      IMPLICIT NONE
      INTEGER ID
      PARAMETER(ID=300)
      INTEGER VET(ID), I, SOMA
!hpf DISTRIBUTE VET(BLOCK)
C-------------Inicio do programa principal --------------
      DO I=1, ID
        READ(10,*) VET(I)
      ENDDO
!hpf independent,reduction(SOMA)
      DO I=1,ID
        SOMA=SOMA+VET(I)
      ENDDO
      WRITE(*,*) 'A SOMA DOS ELEMENTOS DO VETOR E:', SOMA
      END
```

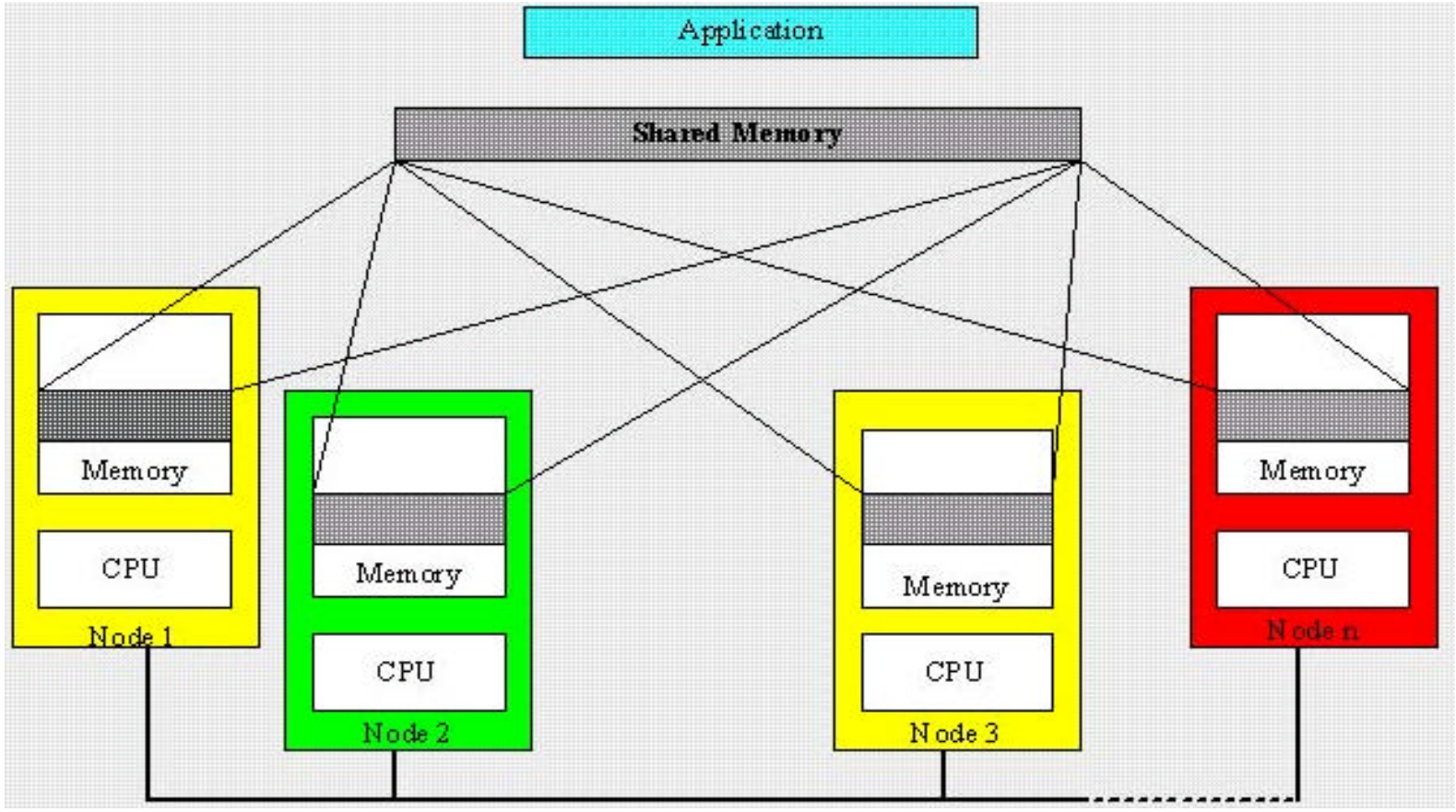
!hpf independent,reduction(SOMA)

Treadmarks

- Implementa DSM – Distributed Shared Memory



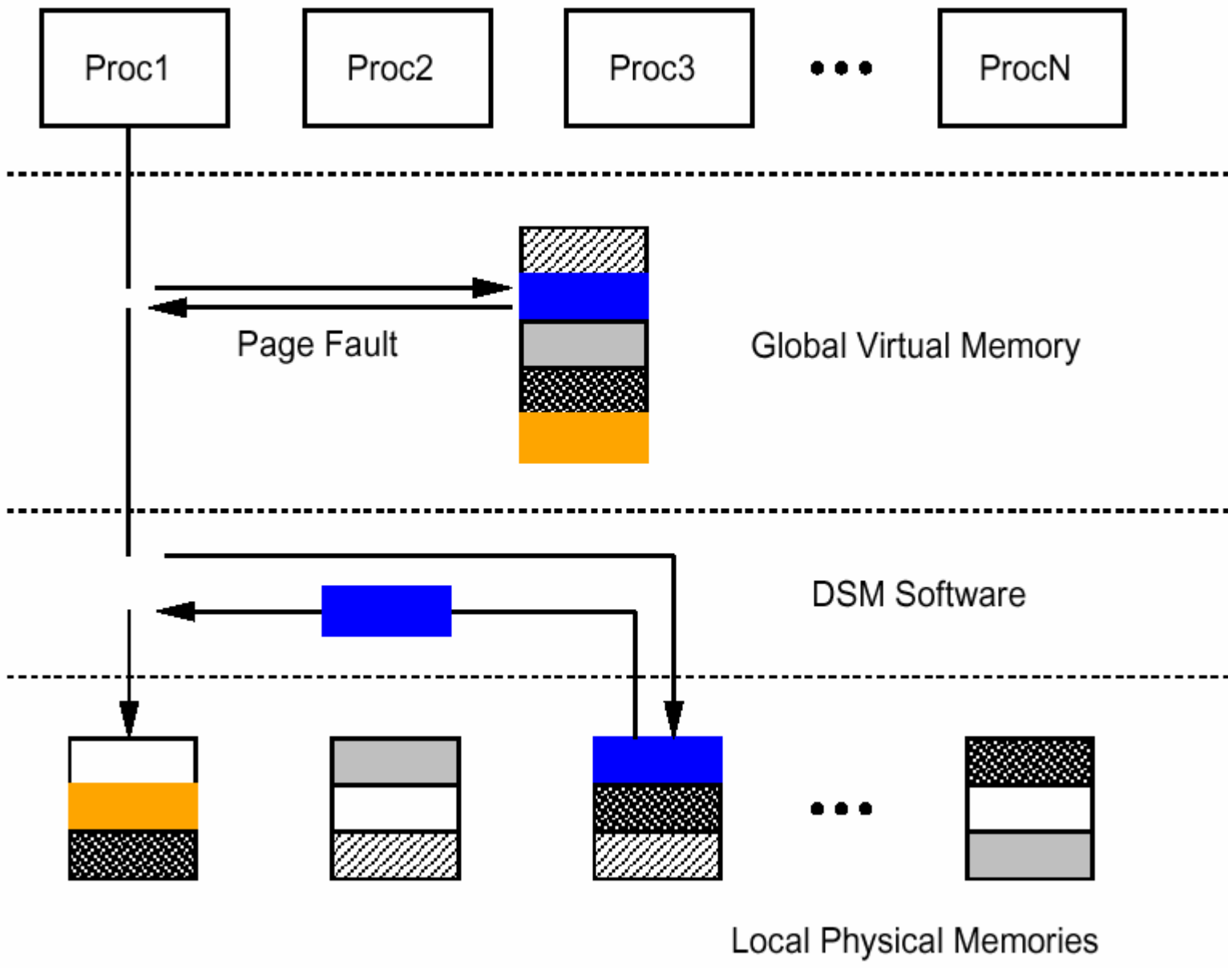
Treadmarks - Características



Treadmarks - Características

- Boa performance (melhor que HPF, por exemplo)
- Paralelismo com dificuldade média
- Divisão dos dados é baseada na divisão de tarefas
- A comunicação entre processadores ocorre de maneira transparente para o programador

Treadmarks


Proc1
Proc2
Proc3
ProcN
Page Fault
Global Virtual Memory
DSM Software
Local Physical Memories

Treadmarks – Implementação Paralela

```
      PROGRAM SOMA_VETOR
C-----------------------------------------------------------
      IMPLICIT NONE
      COMMON /Tmk_shared_common/ VET,SOMAG,SOMAP
      INTEGER ID,NP
      PARAMETER(NP=4)
      PARAMETER (ID=300)
      INTEGER INICIO, FIM, SOMAL, SOMAG, I, VET(ID),SOMAP(NP)
      COMMON /Tmk_shared_common/ VET,SOMAG,SOMAP
C-----------------Inicio do programa principal ---------------
      CALL Tmk_startup()
      INICIO = (ID*Tmk_proc_id)/Tmk_nprocs + 1
      FIM = (ID*(1 + Tmk_proc_id))/Tmk_nprocs
      IF (Tmk_proc_id.EQ.0) THEN
        OPEN (1,FILE='b.dat',STATUS='OLD')
        DO I=1,ID
          READ(1,*) VET(I)
        ENDDO
        SOMAG=0
      ENDIF
      CALL Tmk_barrier(0)
```

Treadmarks – Implementação Paralela

```
      PROGRAM SOMA_VETOR
C-----------------------------------------------------------------
      IMPLICIT NONE
      INCLUDE 'Tmk_fortran.h'
      INTEGER ID,NP
      P
      P
      I                                                          )
      COMMON /Tmk_shared_common/ VET,SOMAG,SOMAP
C-----------------Inicio do programa principal -----------------
      CALL Tmk_startup()
      INICIO = (ID*Tmk_proc_id)/Tmk_nprocs + 1
      FIM = (ID*(1 + Tmk_proc_id))/Tmk_nprocs
      IF (Tmk_proc_id.EQ.0) THEN
        OPEN (1,FILE='b.dat',STATUS='OLD')
        DO I=1,ID
          READ(1,*) VET(I)
        ENDDO
        SOMAG=0
      ENDIF
      CALL Tmk_barrier(0)
```

```
INICIO = (ID*Tmk_proc_id)/Tmk_nprocs + 1
FIM = (ID*(1 + Tmk_proc_id))/Tmk_nprocs
```

Treadmarks – Implementação Paralela

```
      PROGRAM SOMA_VETOR
C-----------------------------------------------------------------
      IMPLICIT NONE
      INCLUDE 'Tmk_fortran.h'
      INTEGER ID,NP
      PARAMETER(NP=4)
      PARAMETER (ID=300)
      INTEGER INICIO, FIM, SOMAL, SOMAG, I, VET(ID),SOMAP(NP)
      COMMON /Tmk_shared_common/ VET,SOMAG,SOMAP
C-----------------Inicio do programa principal ---------------
      CALL Tmk_startup()
      INICIO = (ID*Tmk_proc_id)/Tmk_nprocs + 1
      FIM = (ID*(1 + Tmk_proc_id))/Tmk_nprocs
      IF (Tmk_proc_id.EQ.0) THEN
        OPEN (1,FILE='b.dat',STATUS='OLD')
        DO I=1,ID
          READ(1,*) VET(I)
        ENDDO
        SOMAG=0
      ENDIF
      CALL Tmk_barrier(0)
```

CALL Tmk_barrier(0)

Treadmarks – Implementação Paralela

```
SOMAL=0
DO I=INICIO,FIM
  SOMAL=SOMAL+VET(I)
ENDDO
SOMAP(Tmk_proc_id+1)=SOMAL
CALL Tmk_barrier(0)
IF (Tmk_proc_id.EQ.0) THEN
  DO I=1,Tmk_nprocs
    SOMAG=SOMAG+SOMAP(I)
  ENDDO
ENDIF
CALL Tmk_barrier(0)
WRITE(*,*) 'SOMA GLOBAL:',SOMAG
CALL Tmk_exit(0)
END
```

```
DO I=INICIO,FIM
  SOMAL=SOMAL+VET(I)
ENDDO
```

Treadmarks – Implementação Paralela

SOMAP(Tmk_proc_id+1)=SOMAL

```
SOMAL=0
DO I=INICIO,FIM
  SOMAL=SOMAL+VET(I)
ENDDO
SOMAP(Tmk_proc_id+1)=SOMAL
CALL Tmk_barrier(0)
IF (Tmk_proc_id.EQ.0) THEN
  DO I=1,Tmk_nprocs
    SOMAG=SOMAG+SOMAP(I)
  ENDDO
ENDIF
CALL Tmk_barrier(0)
WRITE(*,*) 'SOMA GLOBAL:',SOMAG
CALL Tmk_exit(0)
END
```

Treadmarks – Implementação Paralela

```
SOMAL=0
DO I=INICIO,FIM
  SOMAL=SOMAL+VET(I
ENDDO
SOMAP(Tmk_proc_id+1)=SOMAL
CALL Tmk_barrier(0)
IF (Tmk_proc_id.EQ.0) THEN
  DO I=1,Tmk_nprocs
    SOMAG=SOMAG+SOMAP(I)
  ENDDO
ENDIF
CALL Tmk_barrier(0)
WRITE(*,*) 'SOMA GLOBAL:',SOMAG
CALL Tmk_exit(0)
END
```

```
IF (Tmk_proc_id.EQ.0) THEN
  DO I=1,Tmk_nprocs
    SOMAG=SOMAG+SOMAP(I)
  ENDDO
ENDIF
```

Troca de Mensagens

- Interface por passagem de mensagem
- Melhor Performance
- Paralelismo com maior grau de dificuldade de implementação
- Padrões de troca de mensagens mais comuns são o **PVM** (parallel Virtual Machine) e **MPI** (Message Passing Interface)

Implementação Paralela MPI

```
PROGRAM SOMA_PARALELA
IMPLICIT NONE
INCLUDE 'mpif.h' !Biblioteca do mpi
INTEGER N, NP
PARAMETER (N=300)
PARAMETER (NP=4)      !Número de processadores

INTEGER I,SOMAL,SOMAG
INTEGER B(N),A(N/(NP-1))
INTEGER RANK, ERR
INTEGER STATUS(MPI_STATUS_SIZE)

INTEGER END_CNT

CALL MPI_INIT(ERR)
CALL MPI_COMM_RANK(MPI_COMM_WORLD, RANK, ERR)
```

```
IF (RANK.EQ.0) THEN
  OPEN(UNIT=10,FILE='b.dat',STATUS='OLD')
  DO I=1,300
    READ(10,*) B(I)
  ENDDO
  CALL MPI_SEND(B(1),100,MPI_INTEGER,1,11,MPI_COMM_WORLD,ERR)
  CALL MPI_SEND(B(101),100,MPI_INTEGER,2,11,MPI_COMM_WORLD,ERR)
  CALL MPI_SEND(B(201),100,MPI_INTEGER,3,11,MPI_COMM_WORLD,ERR)
  END_CNT=0
  SOMAG=0
  DO WHILE (END_CNT.NE.3)
     CALL MPI_RECV(SOMAL,1,MPI_INTEGER,MPI_ANY_SOURCE,
*    MPI_ANY_TAG, MPI_COMM_WORLD, STATUS, ERR)
    END_CNT=END_CNT+1
    SOMAG=SOMAG+SOMAL
  ENDDO
ELSE
```

```
 ! ELSE
      CALL MPI_RECV(A,100,MPI_INTEGER,0,11, MPI_COMM_WORLD,
*       STATUS,ERR)
      SOMAL=0

      DO I=1,100
        SOMAL=SOMAL+A(I)
      ENDDO
      CALL MPI_SEND(SOMAL,1,MPI_INTEGER,0,19,MPI_COMM_WORLD,ERR)
    ENDIF

    WRITE(*,*) 'PROGRAMA FINALIZADO:',RANK
    WRITE(*,*) ‘SOMA FINAL:’,SOMAG

    CALL MPI_FINALIZE(ERR)
    CLOSE(10)
    END
```

Implementação Serial

```
      PROGRAM EXEMPLO
C-------------------------------------------------------
       IMPLICIT NONE
      INTEGER ID
      PARAMETER(ID=300)
      INTEGER VET(ID), I, SOMA
C------------Inicio do programa principal --------------
      SOMA=0
      OPEN (UNIT=10,FILE='b.dat',STATUS='OLD')
      DO I=1, ID
        READ (10,*) VET(I)
      ENDDO
      DO I=1,ID
        SOMA=SOMA+VET(I)
      ENDDO
      WRITE(*,*) 'SOMA DOS ELEMENTOS DO VETOR E:', SOMA
      END
```

Implementação HPF

```
      PROGRAM EXEMPLO
C------------Inicio do programa principal --------------
      IMPLICIT NONE
      INTEGER ID
      PARAMETER(ID=300)
      INTEGER VET(ID), I, SOMA
!hpf DISTRIBUTE VET(BLOCK)
C------------Inicio do programa principal --------------
      SOMA=0
      OPEN(UNIT=10,FILE='b.dat',STATUS='OLD')
      DO I=1, ID
        READ(10,*) VET(I)
      ENDDO
!hpf independent,reduction(SOMA)
      DO I=1,ID
        SOMA=SOMA+VET(I)
      ENDDO
      WRITE(*,*) 'A SOMA DOS ELEMENTOS DO VETOR E:', SOMA
      END
```

Implementação Serial

```
      PROGRAM EXEMPLO
C-------------------------------------------------------
       IMPLICIT NONE
      INTEGER ID
      PARAMETER(ID=300)
      INTEGER VET(ID), I, SOMA
C------------Inicio do programa principal --------------
      SOMA=0
      OPEN (UNIT=10,FILE='b.dat',STATUS='OLD')
      DO I=1, ID
        READ (10,*) VET(I)
      ENDDO
      DO I=1,ID
        SOMA=SOMA+VET(I)
      ENDDO
      WRITE(*,*) 'SOMA DOS ELEMENTOS DO VETOR E:', SOMA
      END
```

Implementação Treadmarks

```
 PROGRAM SOMA_VETOR
C-------------------------------------------------------------
      IMPLICIT NONE
      INCLUDE 'Tmk_fortran.h'
      INTEGER ID,NP
      PARAMETER(NP=4)
      PARAMETER (ID=300)
      INTEGER INICIO, FIM, SOMAL, SOMAG, I, VET(ID),
           SOMAP(NP)
      COMMON /Tmk_shared_common/ VET,SOMAG,SOMAP
C----------------Inicio do programa principal -------------
      CALL Tmk_startup()
      INICIO = (ID*Tmk_proc_id)/Tmk_nprocs + 1
      FIM = (ID*(1 + Tmk_proc_id))/Tmk_nprocs
      IF (Tmk_proc_id.EQ.0) THEN
        OPEN (1,FILE='b.dat',STATUS='OLD')
        DO I=1,ID
          READ(1,*) VET(I)
        ENDDO
        SOMAG=0
      ENDIF
      CALL Tmk_barrier(0)

      SOMAL=0
      DO I=INICIO,FIM
        SOMAL=SOMAL+VET(I)
      ENDDO
      SOMAP(Tmk_proc_id+1)=SOMAL
      CALL Tmk_barrier(0)
      IF (Tmk_proc_id.EQ.0) THEN
        DO I=1,Tmk_nprocs
          SOMAG=SOMAG+SOMAP(I)
        ENDDO
      ENDIF
      CALL Tmk_barrier(0)
      WRITE(*,*) 'SOMA GLOBAL:',SOMAG
      CALL Tmk_exit(0)
      END
```

Implementação Serial

```
      PROGRAM EXEMPLO
C-------------------------------------------------------
       IMPLICIT NONE
      INTEGER ID
      PARAMETER(ID=300)
      INTEGER VET(ID), I, SOMA
C------------Inicio do programa principal --------------
      SOMA=0
      OPEN (UNIT=10,FILE='b.dat',STATUS='OLD')
      DO I=1, ID
        READ (10,*) VET(I)
      ENDDO
      DO I=1,ID
        SOMA=SOMA+VET(I)
      ENDDO
      WRITE(*,*) 'SOMA DOS ELEMENTOS DO VETOR E:', SOMA
      END
```

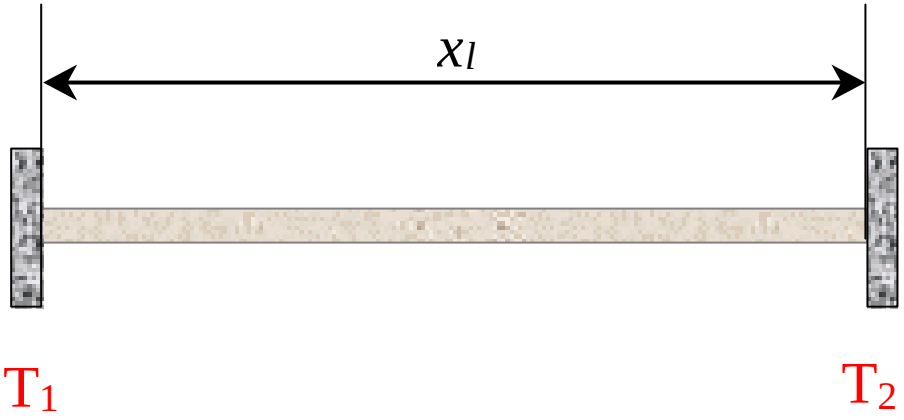
Implementação MPI

```
      PROGRAM SOMA_PARALELA
C-------------------------------------------------------
      IMPLICIT NONE
      INCLUDE 'mpif.h' !Biblioteca do mpi
      INTEGER N, NP
      PARAMETER (N=300)
      PARAMETER (NP=4)      !Número de processadores
      INTEGER I,SOMAL,SOMAG
      INTEGER B(N),A(N/(NP-1))
      INTEGER RANK, ERR
      INTEGER STATUS(MPI_STATUS_SIZE)
      INTEGER END_CNT
C------------Inicio do programa principal --------------
   CALL MPI_INIT(ERR)
   CALL MPI_COMM_RANK(MPI_COMM_WORLD, RANK, ERR)
   IF (RANK.EQ.0) THEN
    OPEN(UNIT=10,FILE='b.dat',STATUS='OLD')
    DO I=1,300
      READ(10,*) B(I)
    ENDDO
    CALL MPI_SEND(B(1),100,MPI_INTEGER,1,11,MPI_COMM_WORLD,ERR)
    CALL MPI_SEND(B(101),100,MPI_INTEGER,2,11,MPI_COMM_WORLD,ERR)
    CALL MPI_SEND(B(201),100,MPI_INTEGER,3,11,MPI_COMM_WORLD,ERR)
    END_CNT=0
    SOMAG=0
    DO WHILE (END_CNT.NE.3)
       CALL MPI_RECV(SOMAL,1,MPI_INTEGER,MPI_ANY_SOURCE,
*      MPI_ANY_TAG, MPI_COMM_WORLD, STATUS, ERR)
      END_CNT=END_CNT+1
      SOMAG=SOMAG+SOMAL
    ENDDO
  ELSE
    CALL MPI_RECV(A,100,MPI_INTEGER,0,11, MPI_COMM_WORLD,
*      STATUS,ERR)
     SOMAL=0
     DO I=1,100
       SOMAL=SOMAL+A(I)
     ENDDO
     CALL MPI_SEND(SOMAL,1,MPI_INTEGER,0,19,MPI_COMM_WORLD,ERR)
   ENDIF

   WRITE(*,*) 'PROGRAMA FINALIZADO:',RANK
   WRITE(*,*) 'SOMA FINAL:',SOMAG

   CALL MPI_FINALIZE(ERR)
   CLOSE(10)
   END
```

Exemplo de comparação de desempenho entre MPI, TreadMarks e HPF



O problema selecionado é uma caso de condução de calor unidimensional em uma barra de comprimento de 1 metro, com propriedades homogêneas, e temperaturas constantes em cada extremidade (T_1=100 ºC e T_2=10º C), conforme apresentado na Fig. acima. É possível escrever a equação governante deste problema como:

$$\frac{?}{?x}\left(k\frac{?T}{?x}\right) + S = 0$$

onde *k* é a o condutividade térmica, *T* é a temperatura e *S* é a taxa de geração de calor por unidade de volume.

Exemplo de comparação de desempenho entre MPI, TreadMarks e HPF

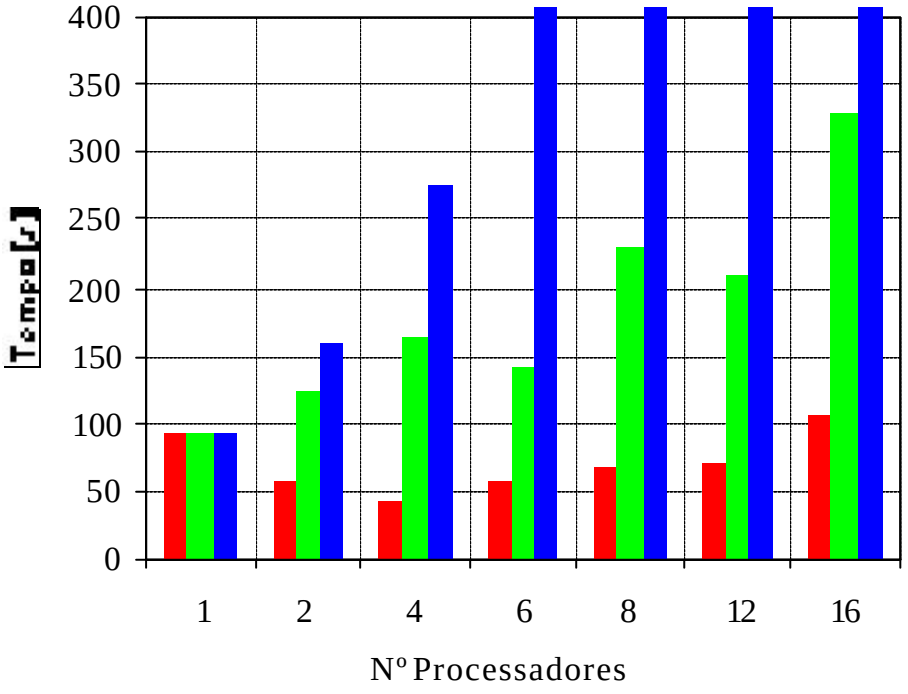
O algoritmo para solução do problema acima foi baseado no método de diferenças finitas e implementado utilizando Fortran 77.

! Leitura dos dados de entrada - condutividade térmica (k), condições de contorno (T1 e T2),
! termo de fonte (S), comprimento do domínio computacional (XL), número de pontos do grid (n);
! Calcula coeficientes independentes A_E, A_W, A_P, b e inicializa matriz de temperatura.
Do while (not stop)
    Do i =2, (n-1)

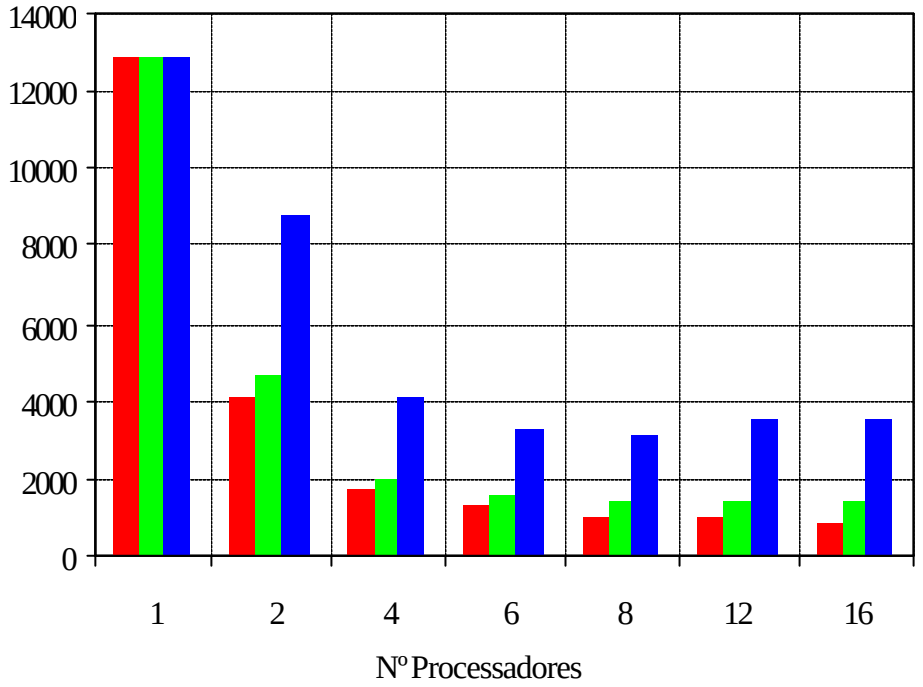
$$T(i) = \text{Relax}\ \frac{(A_E(i) * T(i+1) + A_W(i) * T(i-1) + b(i))}{A_P(i)} + (1 - \text{Relax})\ * T(i)$$

    End-do
    !Calcula resíduo médio
    If (Resíduo < Precisão)
        stop=true;
    End-if
End-do

Análise de Desempenho
tempo de execução



(a)

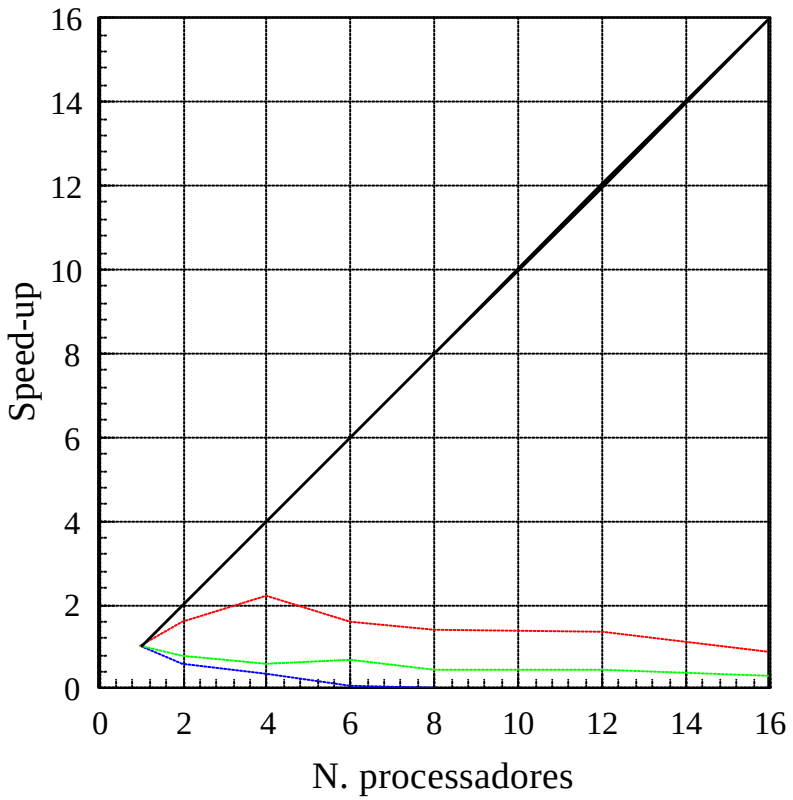


(b)

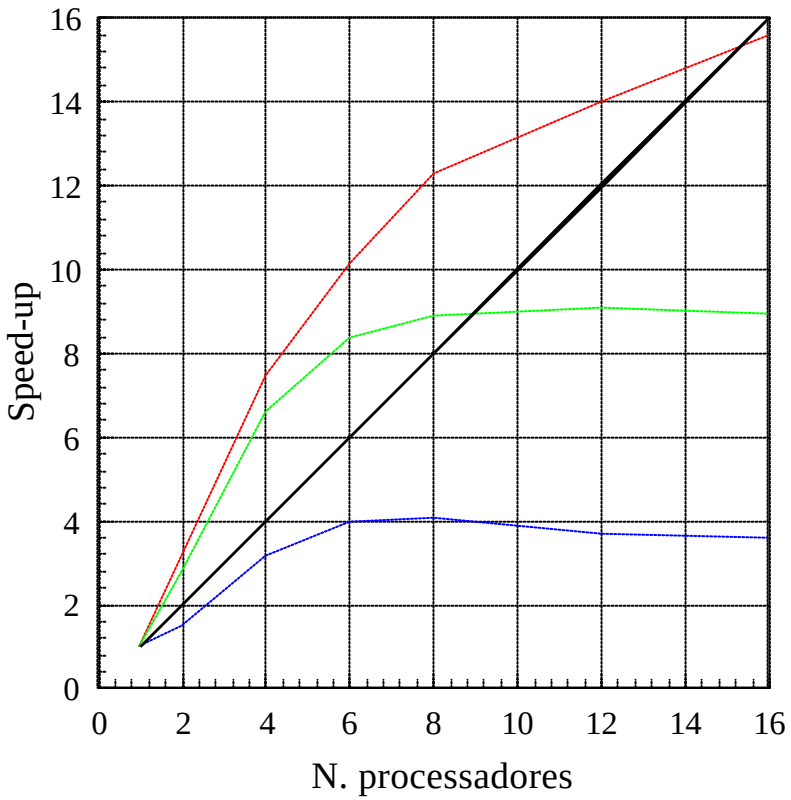
Tempo de execução em segundos para MPI (), TreadMarksTM () e HPF () com (a) 2KVC e (b) 12,5KVC.

Análise de Desempenho

Speed-up (redução do tempo de execução)



(a)

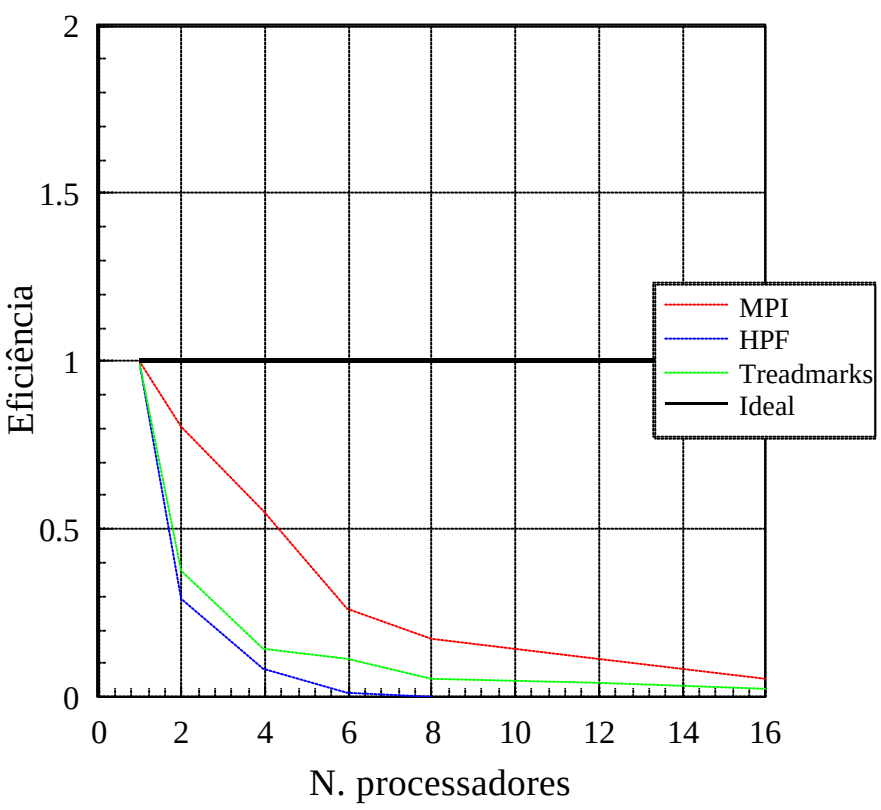


(b)

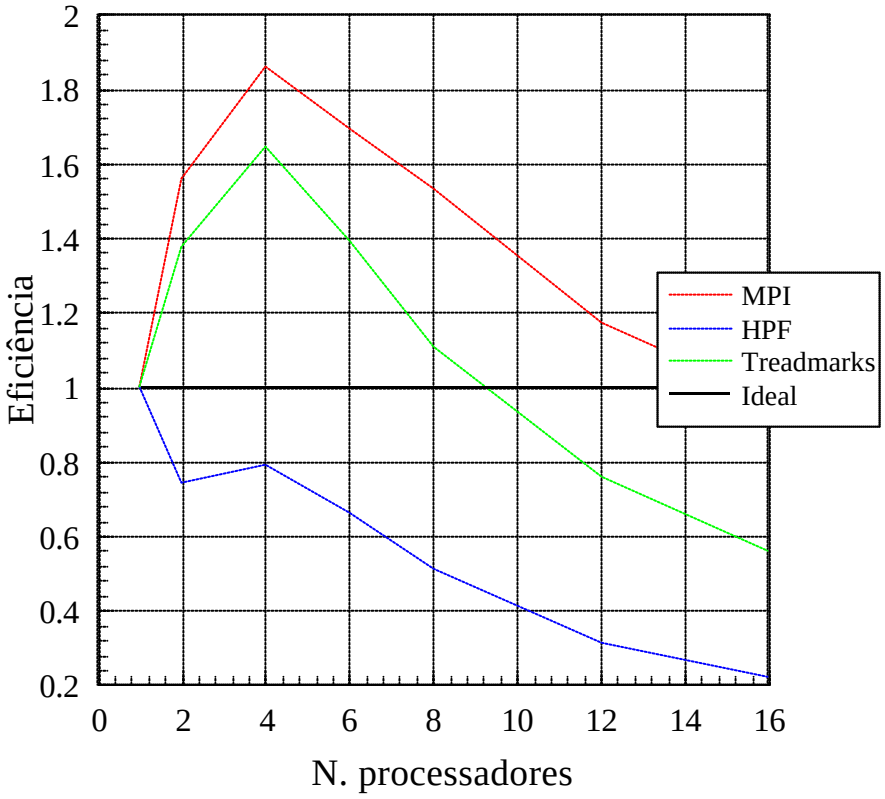
Speed-up obtido para MPI (), TreadMarksTM () e HPF () com (a) 2KVC e (b) 12,5KVC.

Análise de Desempenho

Eficiência



(a)



(b)

Eficiência obtida para MPI (), TreadMarksTM () e HPF () com (a) 2KVC e (b) 12,5KVC.

Conclusão

- Pensar paralelo é uma arte e exige muito do desenvolvedor
- A ferramenta mais adequada vai depender de uma série de fatores:
  - Balanceamento de carga
  - Tamanho do problema
  - Tempo de desenvolvimento requerido
- Necessidade de conhecimento de Hardware

Algoritmos Paralelos (disciplina regular)

Ementa:

1. Modelos de computação paralela:
   - Memória compartilhada: PRAM síncrona e assíncrona, QSP, QWQR, BSP, LogP, SMP
   - Memória distribuída: SMPD
   - circuitos combinatoriais
2. paradigmas de programação: multi-threaded, BSP, passagem de mensagem, funcional, tuple space, CSP
3. desempenho: profundidade de computação, trabalho, custo, speed-up, eficiência
4. Famílias fundamentais de algoritmos: árvore binária balanceada, divide-and-conquer, jumping pointer, compressão, randomização
5. aplicações
   - ordenação
   - grafos
   - processamento de strings
   - álgebra linear
   - otimização
   - Complexidade paralela: P-completeness

UFES
UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO