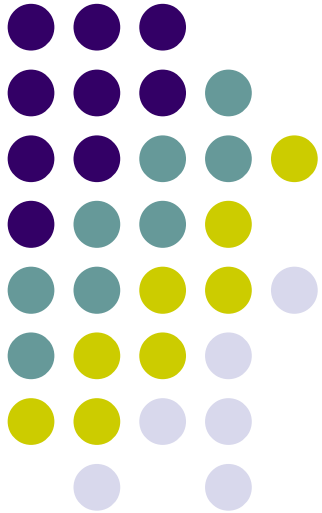
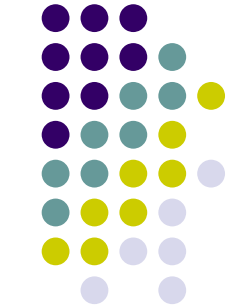
High Performance Fortran

Rodrigo Malara
IFSC – Março de 2004

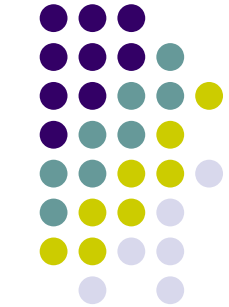


HPF



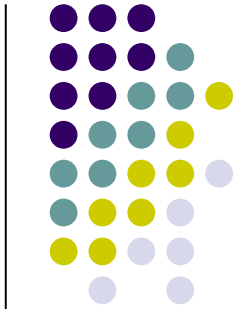
- High Performance Fortran
- Definido pelo HPF Fórum
  - http://www.crpc.rice.edu/HPFF/
- Extensão do Fortran 90
- Paralelização de programas
- SPMD (Single Program Multiple Data)

PGHPF



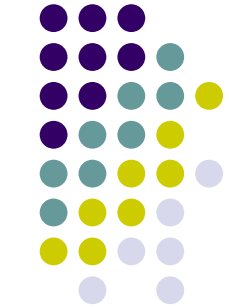
- Portland Group HPF
- Disponível na corto (corto.if.sc.usp.br)
- Não é free software
  - Adere aos padrões definidos pelo HPF Fórum

HPF



- Vantagens
  - Relativa abstração dos aspectos de arquitetura
  - Simplicidade
  - Programa seqüencial/paralelo
    - É possível compilar o programa sequencialmente
    - Não exige o desenvolvimento de variantes (manutenção)
  - Fácil de compilar e de executar
  - É possível analisar post-mortem da execução facilmente

HPF

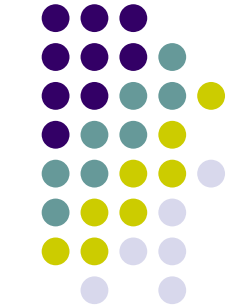


- Desvantagens
  - Ordem das diretivas de paralelização não é muito clara
  - Poucos exemplos e tutoriais
  - A paralelização feita internamente só é conhecida em tempo de execução
  - Problemas na passagem de argumentos na invocação do programa.

HPF

- Princípio básico de funcionamento:
  - Dados são distribuídos (vetores)
  - O mesmo programa atua sobre os dados concorrentemente

Como se programa em HPF?

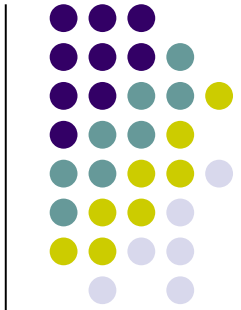


- Programa praticamente idêntico aos seqüenciais:
  - Linhas que contenham diretivas HPF começam com: `!HPF$` ou `cHPF$`
  - As diretivas devem ser colocadas nos locais corretos
    - Ex: após declaração de variáveis

Exemplo de código-fonte

```
      PROGRAM SOMAMAT
      INTEGER, PARAMETER :: TAM = 1000000000
      REAL ITIME, ETIME
      REAL :: V1(TAM), V2(TAM), RES(TAM)
!HPF$ PROCESSORS PROC(NUMBER_OF_PROCESSORS())
!HPF$ TEMPLATE TNP(TAM)
!HPF$ ALIGN WITH TNP :: V1,V2,RES
!HPF$ DISTRIBUTE (BLOCK) ONTO PROC :: TNP
      CALL RANDOM_NUMBER( V1 )
      CALL RANDOM_NUMBER( V2 )
      CALL CPU_TIME( ITIME )
      RES = V1 + V2
      CALL CPU_TIME( ETIME )
      WRITE( *, * ) "Tempo de operacao: ", ETIME -
  ITIME, "segundos"
      ENDPROGRAM SOMAMAT
```

Ex: Diretiva Independent

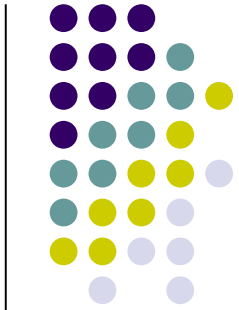


```
      PROGRAM SOMAMAT2
      INTEGER, PARAMETER :: TAM = 1000000000
      INTEGER :: I
      REAL :: V1(TAM), V2(TAM), RES(TAM), SOMATOTAL
!HPF$ PROCESSORS PROC(NUMBER_OF_PROCESSORS())
!HPF$ TEMPLATE TNP(TAM)
!HPF$ ALIGN WITH TNP :: V1,V2,RES
!HPF$ DISTRIBUTE (BLOCK) ONTO PROC :: TNP
      CALL RANDOM_NUMBER( V1 )
      CALL RANDOM_NUMBER( V2 )
!HPF$ INDEPENDENT ON HOME (RES(I)),
  REDUCTION(SOMATOTAL), NEW (I)
      DO I=1,TAM
        RES[I] = V1[I] + V2[I]
        SOMATOTAL = SOMATOTAL + RES[I]
      ENDDO
      ENDPROGRAM
```

Compilando e Executando

- Requisitos mínimos para rodar um programa:
  - O programa sequencial deve estar OK
  - As diretivas HPF devem estar corretas
    - O compilador também faz verificações
  - Utilizar o compilador corretamente

Compilação



- Linha de compilação

```
# pghpf -o arq arq.hpf
```

- Parâmetros que precisam ser setados:
  - Arquivo com nomes das máquinas

```
Ex: /home/rodrigo/hpfhosts
   corto1
   corto2
   corto3
   corto4
   corto5
   corto6
```

- Nome do binário a ser gerado

```
-o <nome do binário>
```

Execução

- Parâmetros podem ser setados
- Variáveis de ambiente

```
# export
  PGHPF_HOST='-file=/home/rodrigo/hpfhost
  s'
```

- Número de processadores

```
# arq -pghpf -np 4
```