Excursionando por OpenMP

Exemplo de uso de OpenMP

```
PROGRAM simple
  IMPLICIT NONE
  INTEGER, PARAMETER :: vSize=1000000000
  REAL :: vec(vSize)
  INTEGER :: i
  DO i = 1, vSize
    vec(i) = REAL(i - vSize/2) ** 2
  END DO
  DO i = 1, vSize
    vec(i) = SQRT(vec(i))
  END DO
  WRITE(*,'(" maximum = ", f9.0,"; minimum =",f9.0)')&
    MAXVAL(vec), MINVAL(vec)
END PROGRAM simple
```

OpenMP Versão 0

```
PROGRAM simple
  IMPLICIT NONE
  INTEGER, PARAMETER :: vSize=1000000000
  REAL :: vec(vSize)
  INTEGER :: i
  DO i = 1, vSize
    vec(i) = REAL(i - vSize/2) ** 2
  END DO
  !$OMP DO PARALLEL
  DO i = 1, vSize
    vec(i) = SQRT(vec(i))
  END DO
  WRITE(*,'(" maximum = ", f9.0,"; minimum =",f9.0)')&
    MAXVAL(vec), MINVAL(vec)
END PROGRAM simple
```

OpenMP Versão 0

Threads	Tempo (s)	Speed-up
1	167,87	1,00
2	128,20	1,31
3	109,62	1,54
4	102,36	1,65
5	97,99	1,72
6	95,08	1,77

OpenMP Versão 0 – O que falta?

```
PROGRAM simple
  IMPLICIT NONE
  INTEGER, PARAMETER :: vSize=1000000000
  REAL :: vec(vSize)
  INTEGER :: i
  DO i = 1, vSize
    vec(i) = REAL(i - vSize/2) ** 2
  END DO
  !$OMP DO PARALLEL
  DO i = 1, vSize
    vec(i) = SQRT(vec(i))
  END DO
  WRITE(*,'(" maximum = ", f9.0,"; minimum =",f9.0)')&
    MAXVAL(vec), MINVAL(vec)
END PROGRAM simple
```

OpenMP Versão 1

```
PROGRAM simple
  IMPLICIT NONE
  INTEGER, PARAMETER :: vSize=1000000000
  REAL :: vec(vSize)
  INTEGER :: i
 !$OMP DO PARALLEL
  DO i = 1, vSize
    vec(i) = REAL(i - vSize/2) ** 2
  END DO
 !$OMP DO PARALLEL
  DO i = 1, vSize
    vec(i) = SQRT(vec(i))
  END DO
  WRITE(*,'(" maximum = ", f9.0,"; minimum =",f9.0)')&
    MAXVAL(vec), MINVAL(vec)
END PROGRAM simple
```

OpenMP Versão 1

Threads	Speed-up
1	1,00
2	1,41
3	1,83
4	2,07
5	2,25
6	2,37

OpenMP Versão 1 – O que falta?

```
PROGRAM simple
  IMPLICIT NONE
  INTEGER, PARAMETER :: vSize=1000000000
  REAL :: vec(vSize)
  INTEGER :: i
 !$OMP DO PARALLEL
  DO i = 1, vSize
    vec(i) = REAL(i - vSize/2) ** 2
  END DO
 !$OMP DO PARALLEL
  DO i = 1, vSize
    vec(i) = SQRT(vec(i))
  END DO
  WRITE(*,'(" maximum = ", f9.0,"; minimum =",f9.0)')&
    MAXVAL(vec), MINVAL(vec)
END PROGRAM simple
```

Aplicando Amdhal

Trecho	% Tempo Exec
Initialização	7,79
SQRT	55,52
Min, Max	36,69

Versão	Fração sequencial (%)	Speed-up maximo
V0	44,48	2,24
V1	36,69	2,72

OpenMP Versão 1

```
PROGRAM simple
  IMPLICIT NONE
  INTEGER, PARAMETER :: vSize=1000000000
  REAL :: vec(vSize)
  INTEGER :: i
!$OMP DO PARALLEL
  DO i = 1, vSize
    vec(i) = REAL(i - vSize/2) ** 2
  END DO
!$OMP DO PARALLEL
  DO i = 1, vSize
    vec(i) = SQRT(vec(i))
  END DO
  WRITE(*,'(" maximum = ", f9.0,"; minimum =",f9.0)')&
    MAXVAL(vec), MINVAL(vec)
END PROGRAM simple
```

OpenMP Versão 2

```
PROGRAM simple
  IMPLICIT NONE
  INTEGER, PARAMETER :: vSize=1000000000
  REAL :: vec(vSize), maxTot(0:7), minTot(0:7)
  INTEGER :: i, this, nThreads
  !$OMP DO PARALLEL
  DO i = 1, vSize
    vec(i) = REAL(i - vSize/2) ** 2
  END DO
  !$OMP DO PARALLEL
  DO i = 1, vSize
    vec(i) = SQRT(vec(i))
  END DO
  maxTot = -1.0; minTot = REAL(vSize) ** 2
  !$OMP DO PARALLEL PRIVATE(this)
  DO i = 1, vSize
    this = OMP_GET_THREAD_NUM() ; nThreads = OMP_GET_NUM_THREADS()
    maxTot(this) = MAX(maxTot(this), vec(i))
    minTot(this) = MIN(minTot(this), vec(i))
  END DO
  WRITE(*,'(" maximum = ", f12.0,"; minimum =",f12.0)')&
    MAXVAL(maxTot(0:nThreads-1)), MINVAL(minTot(0:nThreads-1))
END PROGRAM simple
```

OpenMP Versão 2 - Detalhe

```
maxTot(this) = -1.0
minTot(this) = REAL(vSize) ** 2
!$OMP DO PARALLEL PRIVATE(this)
DO i = 1, vSize
  this = OMP_GET_THREAD_NUM()
  nThreads = OMP_GET_NUM_THREADS()
  maxTot(this) = MAX(maxTot(this), vec(i))
  minTot(this) = MIN(minTot(this), vec(i))
END DO
WRITE(*,'(" maximum = ", f12.0,"; minimum =",f12.0)')&
  MAXVAL(maxTot(0:nThreads-1)), &
  MINVAL(minTot(0:nThreads-1))
END PROGRAM simple
```

OpenMP – Versão 2

Threads	Speed-up
1	1,00
2	2,00
3	2,99
4	3,98
5	4,99
6	6,01

Conclusões da Excursão Inicial - 1

- Paralelismo OpenMP explora independências na codificação do programa
  - Parece fácil
  - Na maioria dos casos, é a forma mais fácil de gerar paralelismo
  - Mas há detalhes complexos (ao longo das aulas)
- Paralelismo OpenMP pode mudar substancialmente o programa
  - Devido ao paralelismo desejado, e não por OpenMP
- Paralelismo OpenMP aplica-se a laços com iterações independentes
  - Mas há exceções (operações de redução, por ex.)

Conclusões da Excursão Inicial - 2

- O paralelismo OpenMP é explícito
  - Diretivas definem trechos do programa executados em paralelo
- Decomposição de Domínio é implícita
  - No programa visitado, qual thread atua sobre determinado elemento do vetor?
  - Mas pode ser programado explicitamente
- Scheduling é implícito
  - No programa visitado, qual thread executa determinada iteração do laço?
  - Mas pode ser programada explicitamente

Atrás das cortinas em OpenMP (Simplificado)

Geração de Paralelismo

- Como o compilador gera paralelismo no trecho:

```
!$OMP DO PARALLEL
DO i = 1, vSize
  vec(i) = REAL(i - vSize/2) ** 2
END DO
```

1. Laço transformado em uma função
2. Cria tantas threads quanto solicitadas, invocando a função
3. As threads são executadas em paralelo
4. Aguarda o término de todas as threads ao fim do laço

- PS: O número de threads é tipicamente definida por variável de ambiente

Divisão de Domínio

- Como o compilador divide o domínio no trecho:

```
!$OMP DO PARALLEL
DO i = 1, vSize
  vec(i) = REAL(i - vSize/2) ** 2
END DO
```

- Suponha variáveis:
  - nThreads: número de threads
  - threadID: número desta thread (0:nThreads-1)

- Então a função realizada pela thread pode ser

```
DO i = 1+threadID, vSize, nThreads
  vec(i) = REAL(i - vSize/2) ** 2
END DO
```

- Como seria a função se a divisão de domínio fosse por blocos (no lugar de cíclica, como acima?)

Variáveis Locais e Globais

- Como o compilador trata as variáveis no trecho:

```
!$OMP DO PARALLEL
DO i = 1, vSize
  vec(i) = REAL(i - vSize/2) ** 2
END DO
```

- Por default, todas as variáveis são globais
  - Basta utilizar a posição existente antes da criação da pilha (stack) da thread

- Entretanto, como tratar o índice do laço

```
DO i = 1+threadID, vSize, nThreads
  vec(i) = REAL(i - vSize/2) ** 2
END DO
```

- O índice não pode ser uma variável global (race condition)
  - O índice é variável local à thread, por default