Instituto de Informática

OpenMP:
Uma Introdução

Cláudio Geyer

UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

Informática
UFRGS

Fonte

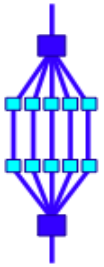
- Fontes
  - Slides inicialmente baseados em palestra de Ruud van der Pas
    - Janeiro de 2009
    - Na Nanyang Technological University, Singapore
  - Afiliação autor
    - Sun Microsystems

Informática UFRGS

Fonte

NANYANG TECHNOLOGICAL UNIVERSITY

Sun microsystems

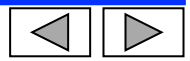


An Overview of OpenMP

Ruud van der Pas

Senior Staff Engineer
Technical Developer Tools
Sun Microsystems, Menlo Park, CA, USA

Nanyang Technological University
Singapore
Wednesday January 14, 2009



UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

Informática UFRGS

Sumário

- OpenMP:
  - O que é? Para que serve? ...
  - Quando usar
  - Vantagens
  - Mecanismos básicos de paralelização
    - Foco em loops

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Informática UFRGS

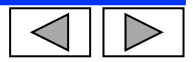
Organizações OpenMP

OpenMP™

http://www.openmp.org

cOMPunity

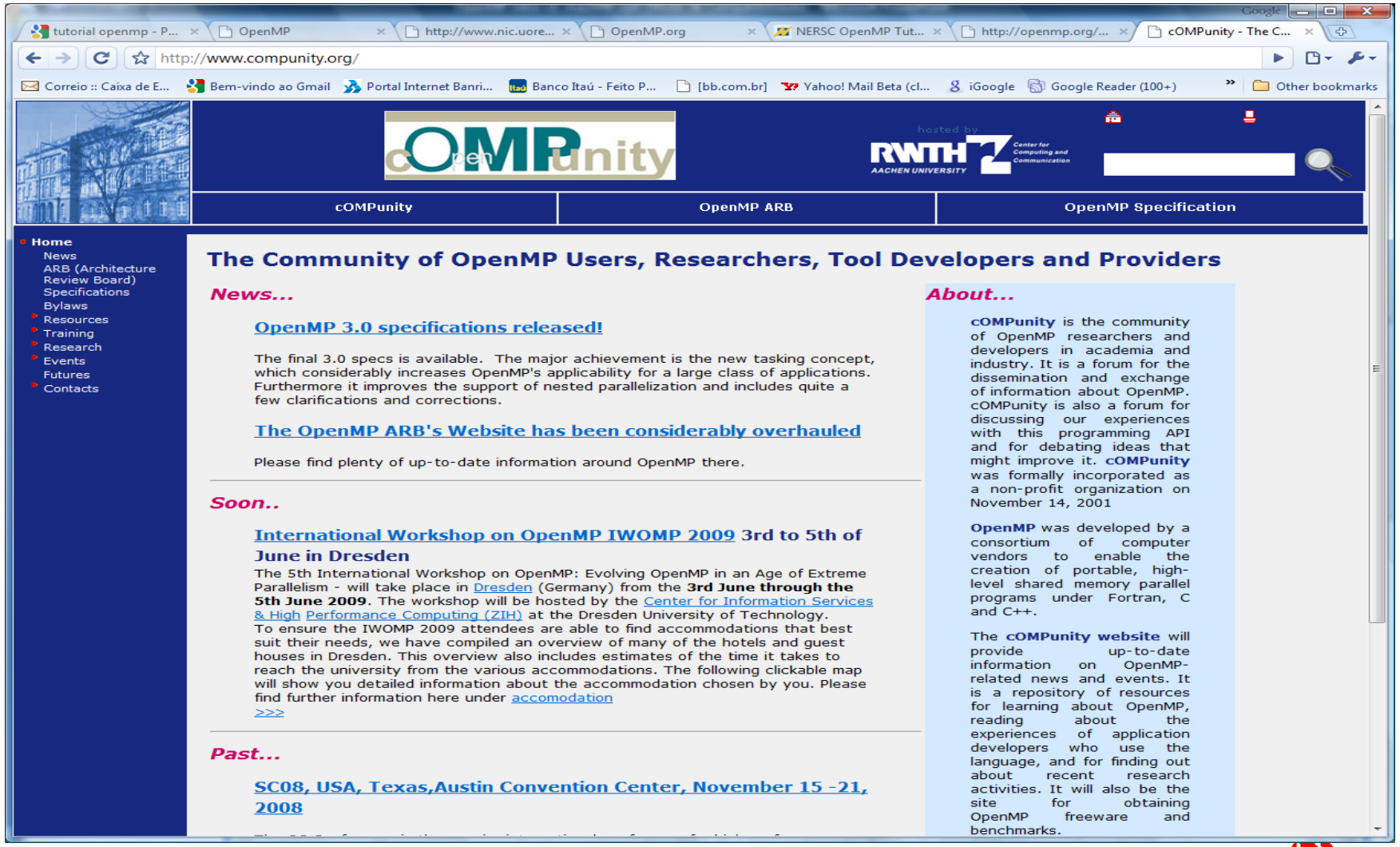
http://www.compunity.org



UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

OpenMP: 1ª página

OpenMP Community: 1ª página



O que é OpenMP?

- Especificação para programação paralela em memória compartilhada
- Padrão “de fato”
- Mantida por ***OpenMP Architecture Review Board***
  - http://www.openmp.org
- Versão 3.0 produzida em maio de 2008
- Consiste de
  - Diretivas de compilação
  - Rotinas de execução
  - Variáveis de ambiente

Informática UFRGS

O que é OpenMP?

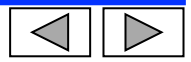
- Versões (especificação)
  - Versão 2.5: 2005
  - Versão 3.0: maio de 2008
  - Versão 3.1: julho de 2011
  - Versão 4.0: em preparação

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Informática UFRGS

Quando usar OpenMP

- Comparando com compiladores paralelizantes sem diretivas
- O compilador não consegue obter o paralelismo desejado pelo programador
  - Porque não consegue "ver" o paralelismo
    - A análise de dependências não tem certeza se pode paralelizar
  - A granularidade das tarefas não é suficiente
    - O compilador não tem informações para paralelizar no mais alto nível
- A paralelização explícita via OpenMP pode resolver esses problemas



Vantagens de OpenMP

- Bom desempenho e escalabilidade
  - Se o programador fizer "a coisa certa"
- Padrão de fato e maduro
- Programa OpenMP é portável
  - Suportado por vários compiladores
  - IBM, Intel, ...
- Requer pouco esforço do programador
  - Comparando com Posix ou Java threads
- Permite paralelização incremental
  - Por partes (loops) do programa

Vantagens de OpenMP

- Programa sequencial (quase) = paralelo
  - Facilita manutenção
  - Facilita depuração
    - Por exemplo, em caso de erro (?) na versão paralela: executar a versão sequencial com mesma entrada

Informática UFRGS

OpenMP e Multicore

- OpenMP é especialmente indicado para processadores multicore
  - Modelos de memória e de threads podem ser mapeados de forma natural
  - Leve
  - Maduro
  - Muito usado e disponível

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Modelo de Memória de OpenMP

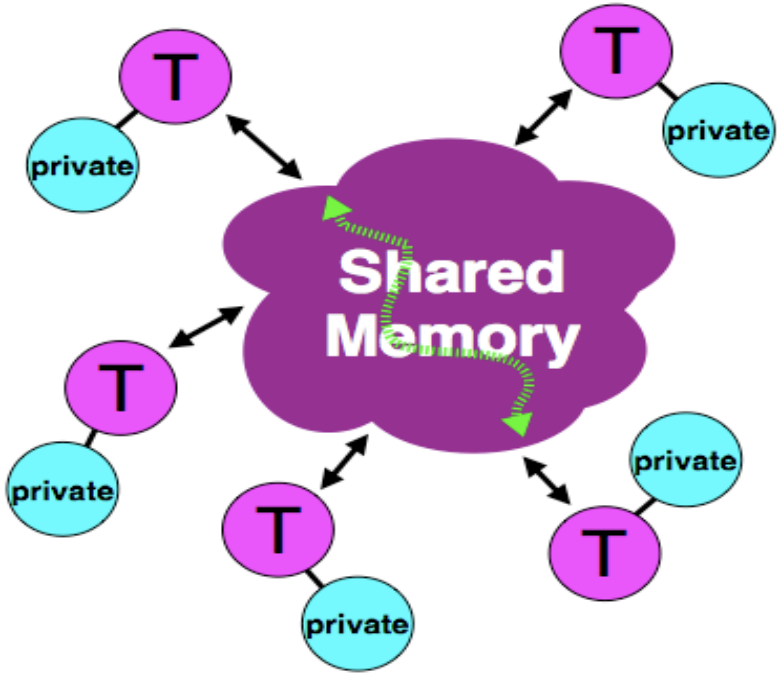
- Memória global compartilhada
  - Todas as tarefas (threads) têm acesso a essa memória
- Dados podem ser compartilhados ou privados
- Dados compartilhados podem ser acessados por todas as tarefas
- Dados privados somente podem ser acessados pela tarefa proprietária dos dados
- Transferência de dados é transparente ao programador
- Há mecanismos de sincronização implícitos

Modelo de Memória de OpenMP

NTU Talk
January 14
2009

10

The OpenMP Memory Model



- ✔ *All threads have access to the same, globally shared, memory*
- ✔ *Data can be shared or private*
- ✔ *Shared data is accessible by all threads*
- ✔ *Private data can only be accessed by the thread that owns it*
- ✔ *Data transfer is transparent to the programmer*
- ✔ *Synchronization takes place, but it is mostly implicit*

Atributos de Dados Compartilhados

- Em um programa OpenMP, os dados precisam receber um atributo ("labelled")
- Há dois tipos básicos
  - Shared
  - Private
- Shared
  - Só há uma instância do dado
  - Todas as tarefas podem ler e escrever nesses dados concorrentemente
    - Exceções se construtores são usados
  - Todas as alterações são visíveis a todas as tarefas
    - Mas não imediatamente a não ser se forçadas

Atributos de Dados Compartilhados

- Private
  - Cada tarefa tem uma cópia do dado
  - Nenhuma outra tarefa pode acessar o dado
  - Alterações são visíveis somente à tarefa proprietária do dado

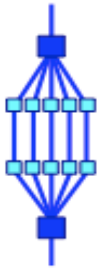
- Mais informações sobre a semântica de variáveis
  - Curso sobre OpenMP na ERAD 2010
  - Anais das ERADs: na biblioteca do INF

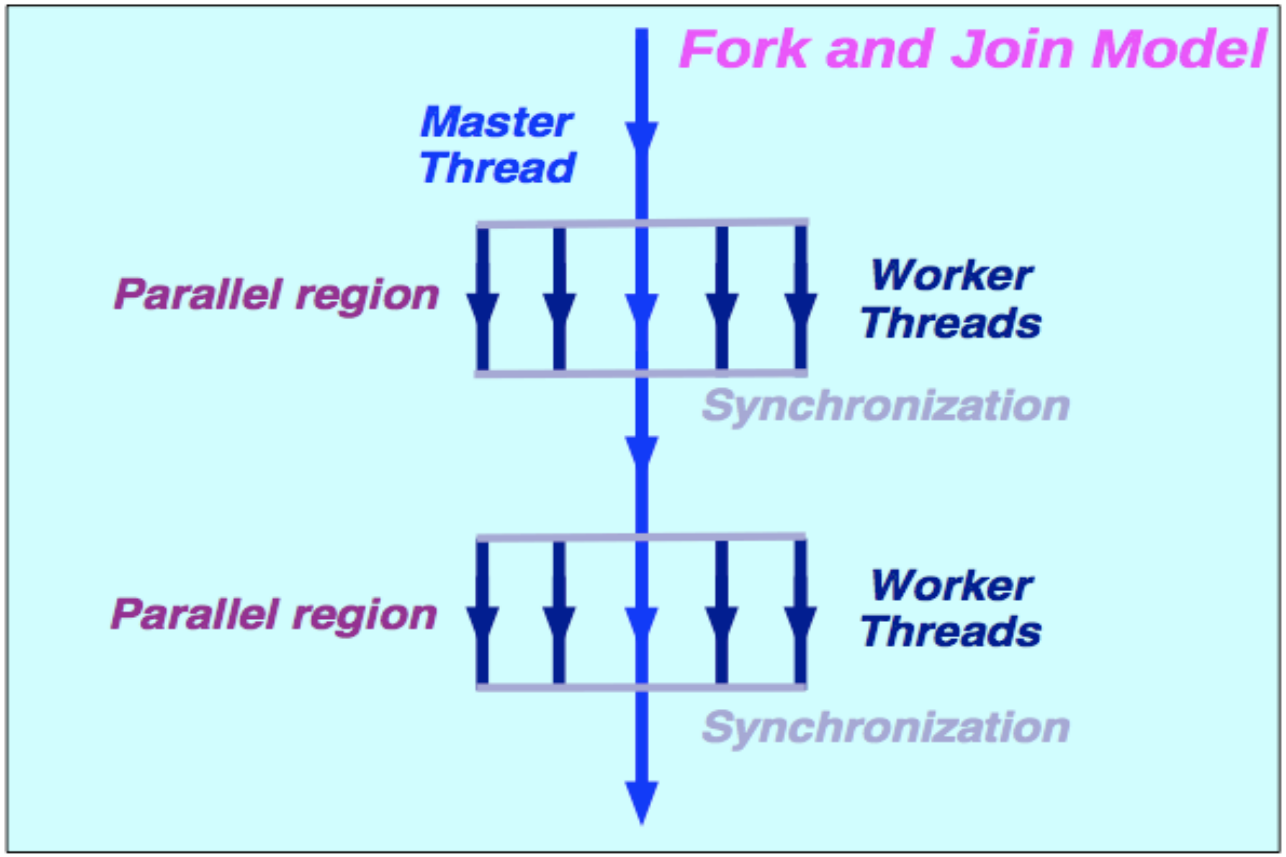
Modelo de Fluxos de Execução

- Modelo de fluxos de execução
  - Segue um modelo parbegin/parend repetido
  - Tipos de threads
    - Master: nas partes sequenciais
    - Workers: nas partes concorrentes
  - Região paralela
    - Parte concorrente
    - Com threads workers

Modelo de Execução OpenMP

The OpenMP Execution Model





Primeiro Exemplo OpenMP

- Loop sequencial com iterações independentes

```
for (int i=0; i<n; i++)
   c[i] = a[i] + b[i];
```

- Loop paralelizado usando diretiva OpenMP

```
#pragma omp parallel for
for (int i=0; i<n; i++)
   c[i] = a[i] + b[i];
```

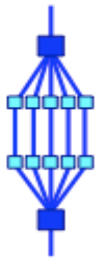
```
% cc –xopenmp source.c
% setenv OMP_NUM_THREADS 5
% a.out
```

Exemplo de execução de loop

- Exemplo de execução de loop
  - Ver próximo slide com figura
  - 5 threads (fluxos de execução)
    - Numeradas de 0 a 4
  - Código único nas threads
  - Vetor de 1000 elementos
  - Cada thread operando sobre um subconjunto distinto de partes do vetor
    - Vetor parcial com índices consecutivos
  - Por exemplo, thread 1 sobre índices 200 a 399

Execução Paralela do Exemplo

Example parallel execution



Thread 0	Thread 1	Thread 2	Thread 3	Thread 4
i=0-199	i=200-399	i=400-599	i=600-799	i=800-999
a[i]	a[i]	a[i]	a[i]	a[i]
+	+	+	+	+
b[i]	b[i]	b[i]	b[i]	b[i]
=	=	=	=	=
c[i]	c[i]	c[i]	c[i]	c[i]

Componentes da versão 2.5

- Diretivas
  - Região paralela
  - Worksharing
  - Synchronization
  - Atributos de dados compartilhados
    - private
    - firstprivate
    - lastprivate
    - Shared
    - reduction
  - Orphaning

Componentes da versão 2.5

- Ambiente de execução
  - Quantidade de tarefas
  - Id da tarefa
  - Ajuste dinâmico de tarefas
  - Paralelismo aninhado
  - Tempo de parede
  - Bloqueios
- Variáveis de ambiente
  - Quantidade de tarefas
  - Tipo de escalonamento
  - Ajuste dinâmico de tarefas
  - Paralelismo aninhado

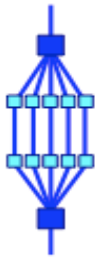
Exemplo Produto Vetorial (M*V)

- Exemplo mais elaborado
  - Produto vetorial: M*V
  - Visto como exemplo em algoritmos PRAM
  - Conjuntos de variáveis shared e private
  - Cada thread calcula um subconjunto de elementos do vetor resultado

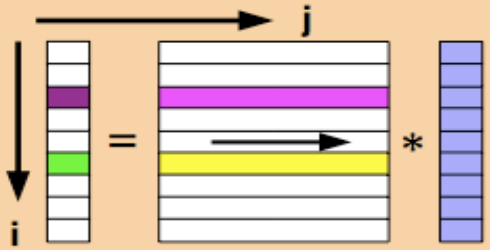
2º Exemplo: M*V

Example - Matrix times vector

Sun microsystems



```
#pragma omp parallel for default(none) \
            private(i,j,sum) shared(m,n,a,b,c)
for (i=0; i<m; i++)
{
   sum = 0.0;
   for (j=0; j<n; j++)
     sum += b[i][j]*c[j];
   a[i] = sum;
 }
```



TID = 0

```
for (i=0,1,2,3,4)
 i = 0
sum = Σ  b[i=0][j]*c[j]
   a[0] = sum
 i = 1
sum = Σ  b[i=1][j]*c[j]
   a[1] = sum
```

TID = 1

```
for (i=5,6,7,8,9)
 i = 5
sum = Σ  b[i=5][j]*c[j]
   a[5] = sum
 i = 6
sum = Σ  b[i=6][j]*c[j]
   a[6] = sum
```

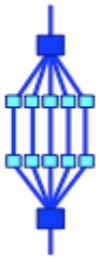
... etc ...

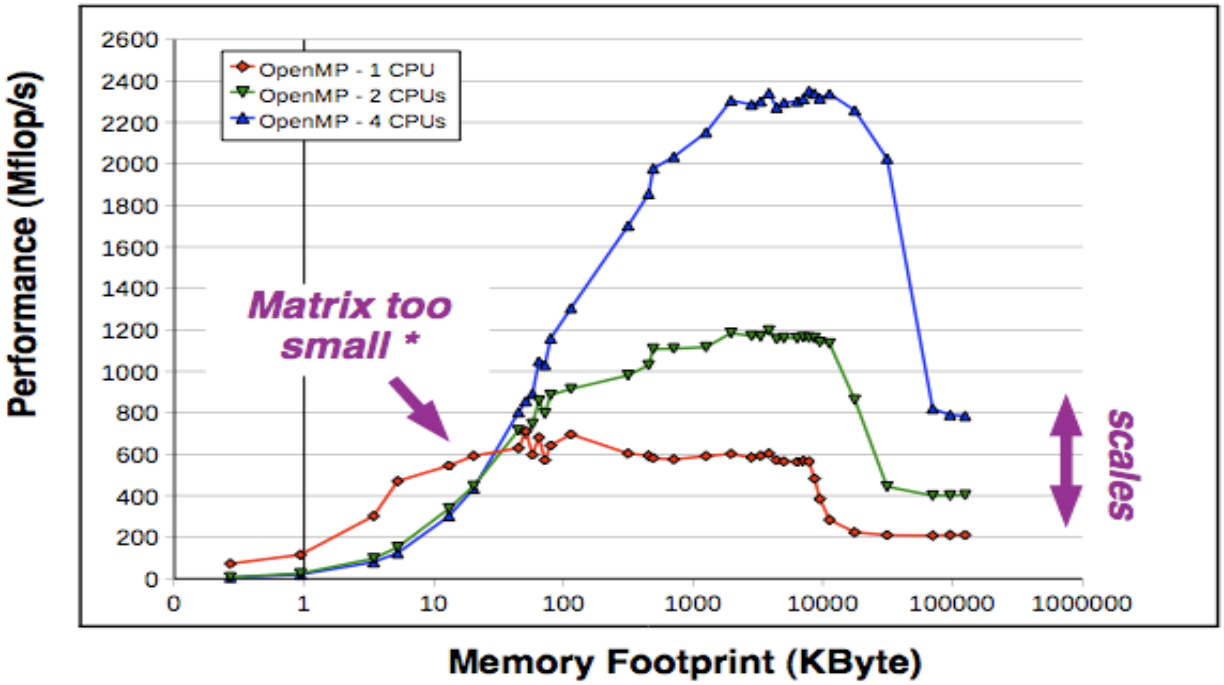
Exemplo de Avaliação de Desempenho

- Exemplo de avaliação de desempenho
  - Execução do mesmo problema com variação do número de cpus
  - Uma curva para cada quantidade de cpus
  - Eixo X:
    - Tamanho do problema
  - Eixo Y:
    - Tempo de execução (tempo paralelo)
  - Notar a perda de desempenho para entradas pequenas
    - 1 core é mais eficiente que 2 e 4 cores
  - Notar a perda de desempenho para ... grandes

Desempenho de OpenMP

OpenMP performance





*) With the IF-clause in OpenMP this performance degradation can be avoided

Outro Exemplo Sintético

- Exemplo de programa sintético
  - Ver slide com código adiante
  - Exemplo abstrato
  - Com várias blocos paralelizados
  - Uso de barreira

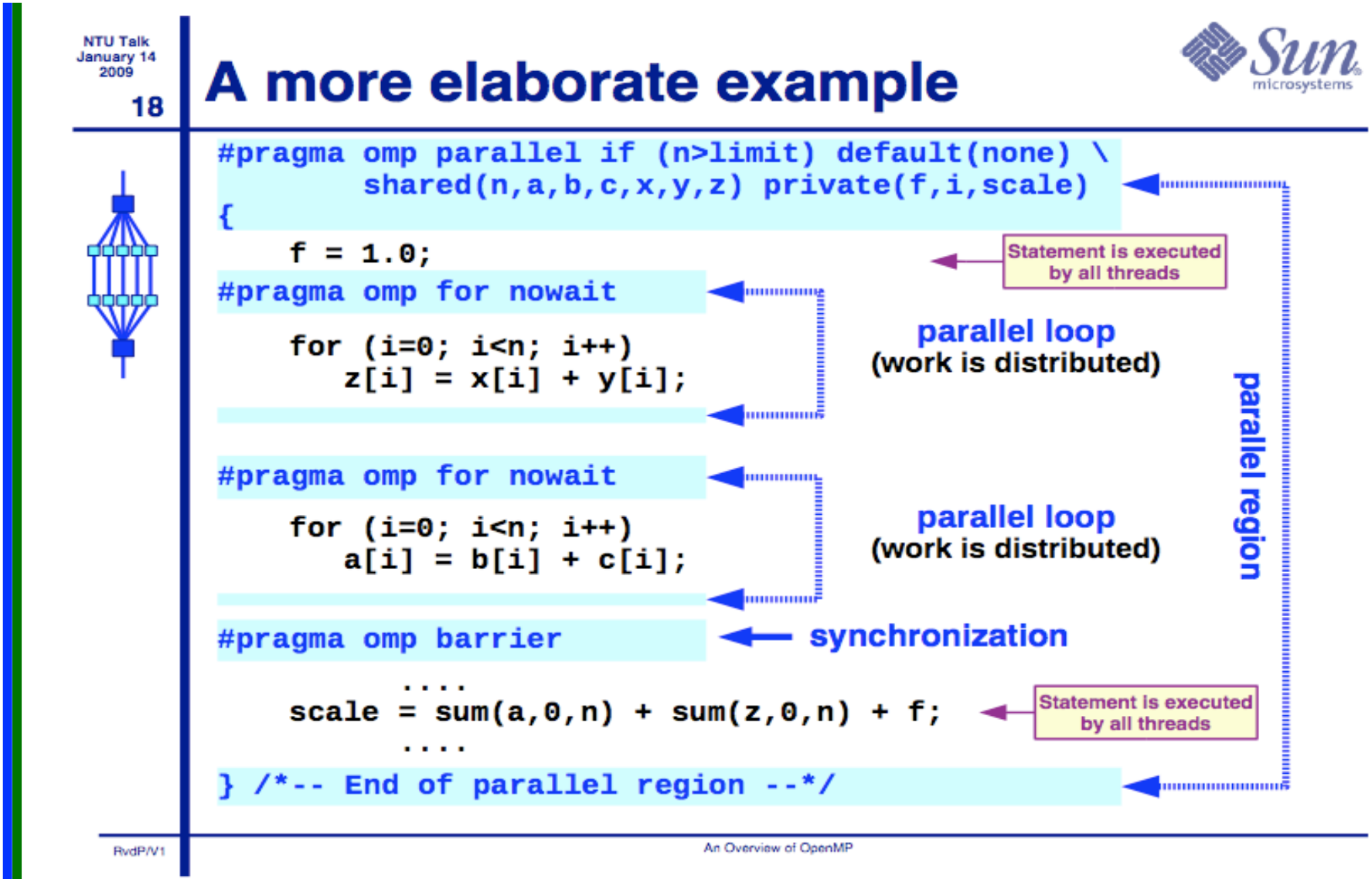
Outro Exemplo Sintético

- Descrição detalhada
  - 1º pragma
    - Define uma região paralela
    - Executada em paralelo se condição verdadeira
      - n > limit
  - Comando sem pragma
    - Executado por todas as threads
  - 2º pragma
    - Define um loop paralelo

Outro Exemplo Sintético

- Descrição detalhada
  - 3º pragma
    - Define um 2º loop paralelo
  - 4º pragma
    - Define uma barreira

Um Exemplo Sintético +



Informática UFRGS

OpenMP em mais Detalhes

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Informática UFRGS

Súmula

- Conceitos gerais
  - Região, condição geral, work-sharing
- Mecanismos básicos: conceito, sintaxe e exemplo
  - Cláusula *if*
  - Cláusula *shared*
  - Cláusula *private*
  - Barreira
  - Região
  - Cláusula *nowait*

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Informática UFRGS

Súmula

- Mecanismos básicos: (cont.)
  - Work-sharing
  - Construtores *work-sharing*
    - for, sections, single

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Termos e Comportamento

- Time OpenMP := Master + Workers
- Região Paralela
  - Bloco de código executado por todas as threads simultaneamente
  - Thread master sempre tem ID = 0
  - Ajuste de threads (se permitido) é realizado somente antes do início da execução da região
  - Regiões podem ser aninhadas mas esse recurso é dependente de implementação
  - Uma cláusula "if" pode ser usada como guarda
    - Se avaliada como "falsa", o código da região é executado sequencialmente

Termos e Comportamento

- Construção “work-sharing”
  - Divide a execução do código da região entre os membros do team

Cláusula If

- Cláusula If
  - Sintaxe
    - if (expressão-escalar)
  - Somente executa em paralelo se expressão é avaliada em “true”
  - Caso contrário, executa sequencialmente

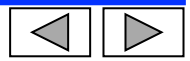
```
#pragma omp parallel if (n > threshold) \
        shared(n,x,y) private(i)
  {
    #pragma omp for
     for (i=0; i<n; i++)
        x[i] += y[i];
  } /*-- End of parallel region --*/
```

Informática UFRGS

Cláusula Shared

- Cláusula Shared
  - Sintaxe
    - shared (lista-de-variáveis)
  - Dados são acessíveis a todas as threads
  - Dados acessados no mesmo endereço

```
#pragma omp parallel if (n > threshold) \
        shared(n,x,y) private(i)
  {
    #pragma omp for
     for (i=0; i<n; i++)
        x[i] += y[i];
  } /*-- End of parallel region --*/
```



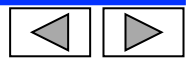
UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Informática UFRGS

Cláusula Private

- Cláusula Private
  - Sintaxe
    - private (lista-de-variáveis)
  - Todas as referências são locais
  - Valores indefinidos na entrada e saída da região
  - Não há associação com dado (variável) original

```
#pragma omp parallel if (n > threshold) \
        shared(n,x,y) private(i)
  {
    #pragma omp for
     for (i=0; i<n; i++)
        x[i] += y[i];
  } /*-- End of parallel region --*/
```



UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Barreira /1

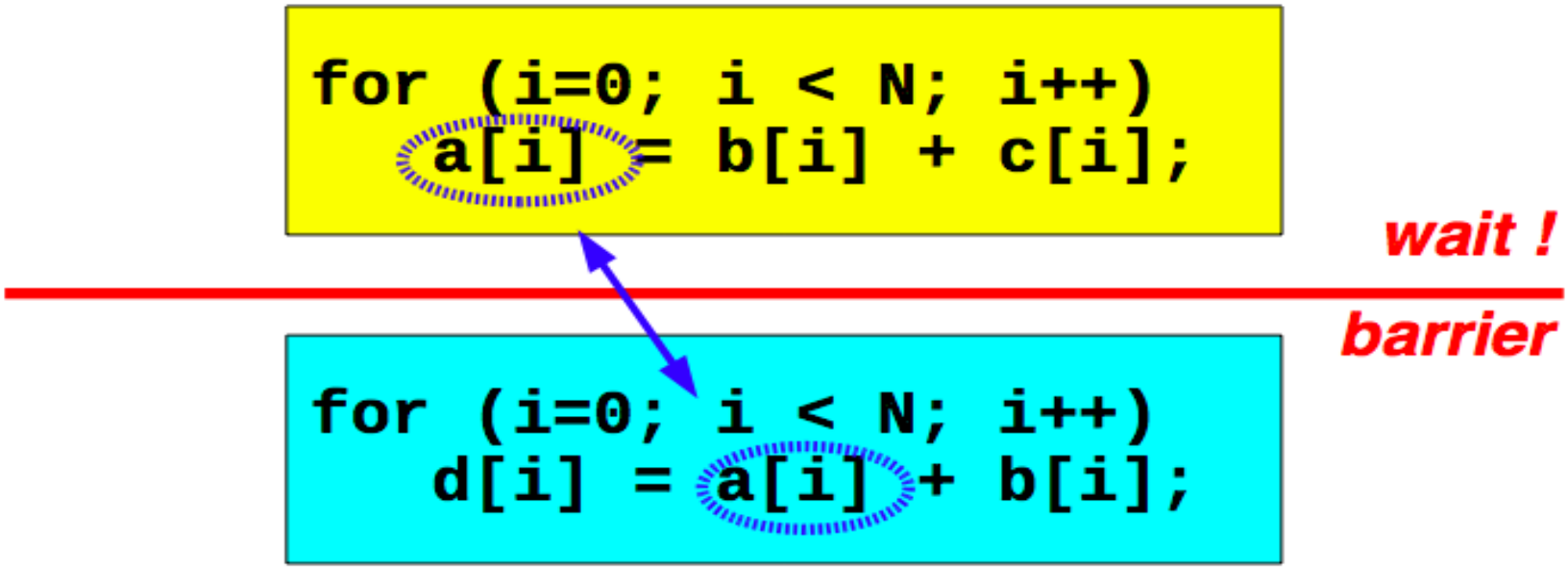
- Barreira
  - Supondo execução em paralelo dos 2 loops abaixo
  - Uma execução pode gerar algum erro (resultado inconsistente)?
  - Porque?

```
for (i=0; i < N; i++)
   a[i] = b[i] + c[i];
```

```
for (i=0; i < N; i++)
   d[i] = a[i] + b[i];
```

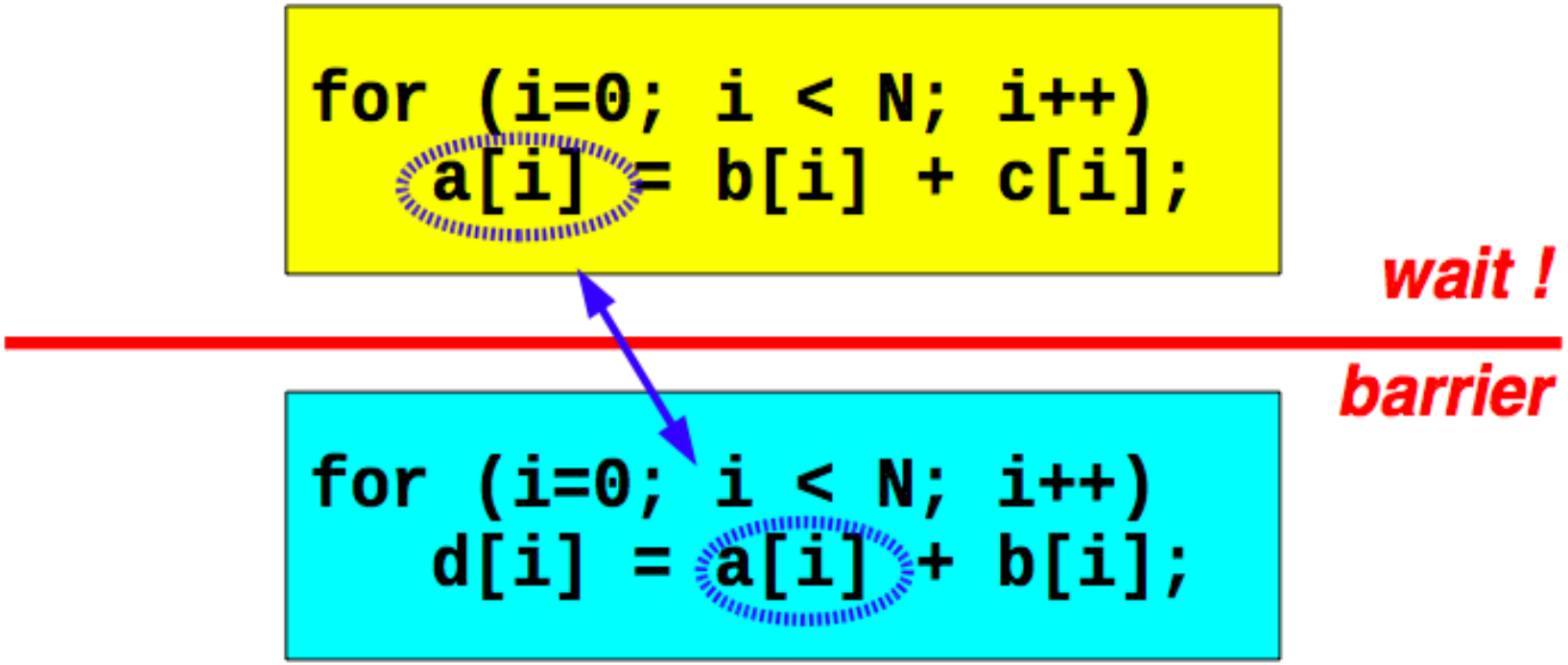
Barreira /2

- Barreira
  - É neccessário atualizar todo o *a[]* antes de usa-lo
  - Seria possível se as iterações de ambos os loops fossem mapeadas às mesmas threads



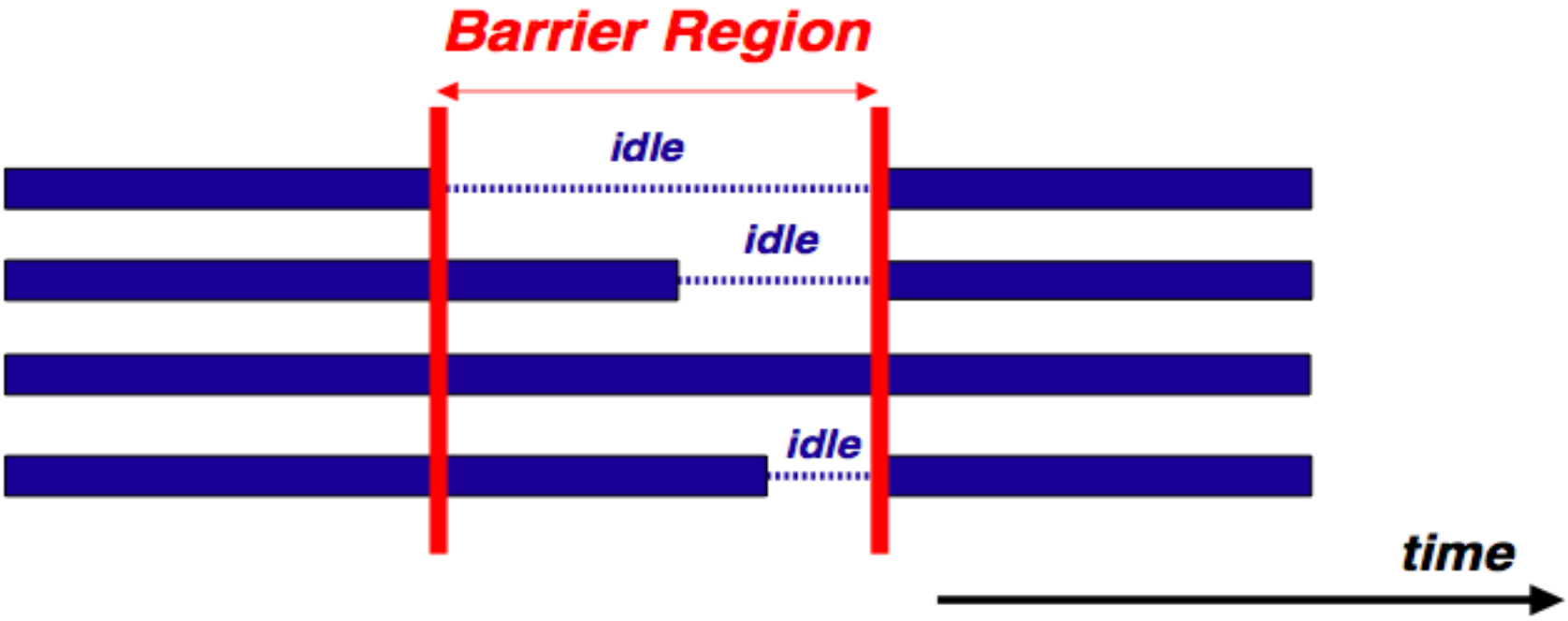
Barreira /3

- Barreira
  - Todas as threads esperam na barreira e só continuam após todas terem atingido a barreira



Barreira /4

- Barreira
  - Diagrama de tempo
  - Algumas threads (cores?) podem ficar ociosas



Barreira /5

- Barreira
  - Sintaxe

```
#pragma omp barrier
```

```
!$omp barrier
```

Cláusula Nowait

- Cláusula Nowait
  - Para minimizar custo de sincronização, algumas diretivas OpenMP suportam a cláusula “nowait”
  - A cláusula é opcional
  - Se usada, threads não esperam (sincronizam) ao final da construção (associada à “nowait”)
  - Obs.:
    - A barreira implícita ocorre ao final de construtores de OpenMP

Cláusula Nowait

- Cláusula Nowait
  - Sintaxe
    - Em Fortran, é colocada no final da diretiva
    - Em C, é uma cláusula ao lado do pragma

```
#pragma omp for nowait
{
        :
}
```

```
!$omp do
        :
        :
!$omp end do nowait
```

Região Paralela

- Região Paralela
  - É um bloco de código executado por múltiplas threads simultaneamente
  - Sintaxe (Fortran e C)

```
!$omp parallel [clause[[,] clause] ...]

    "this is executed in parallel"

!$omp end parallel (implied barrier)
```

```
#pragma omp parallel [clause[[,] clause] ...]
{
    "this is executed in parallel"

} (implied barrier)
```

Construtores em Work-Sharing

- Construtores (diretivas) em Work-Sharing
  - for, sections, single

```
#pragma omp for
{
    ....
}

!$OMP DO
    ....
!$OMP END DO
```

```
#pragma omp sections
{
        ....
}

!$OMP SECTIONS
        ....
!$OMP END SECTIONS
```

```
#pragma omp single
{
        ....
}

!$OMP SINGLE
        ....
!$OMP END SINGLE
```

Construtores em Work-Sharing

- Construtores (diretivas) em Work-Sharing
  - O trabalho é distribuído sobre as threads
  - O trabalho deve estar embutido na região paralela
  - Deve ser "encontrado" por todas as threads do time ou nenhuma
  - Não há barreira implícita na entrada
  - Há uma barreira implícita na saída
    - Possível exceção com cláusula "nowait"
  - Um construtor "work-sharing" não dispara novas threads

Construtores em Work-Sharing

- Construtores (diretivas) em Work-Sharing
  - Fortran possui mais um construtor: “workshare”
  - Sintaxe

```
!$OMP WORKSHARE

    <array syntax>

!$OMP END WORKSHARE [NOWAIT]
```

Construtores em Work-Sharing

- Construtores (diretivas) em Work-Sharing
  - Fortran construtor: “workshare”
  - Exemplo
    - Soma de 2 vetores é executada como um loop paralelo

```
!$OMP WORKSHARE
    A(1:M) = A(1:M) + B(1:M)
!$OMP END WORKSHARE NOWAIT
```

Construtor for/do

- Construtor “for/do”
  - As iterações do loop são distribuídas para as threads
  - Sintaxe

```
#pragma omp for [clause[[,] clause] ...]
   <original for-loop>
```

```
!$omp do [clause[[,] clause] ...]
        <original do-loop>
!$omp end do [nowait]
```

Informática UFRGS

Construtor for/do

- Construtor “for/do”
  - Cláusulas suportadas
    - private
    - firstprivate
    - lastprivate
    - reduction
    - ordered
    - schedule
    - nowait

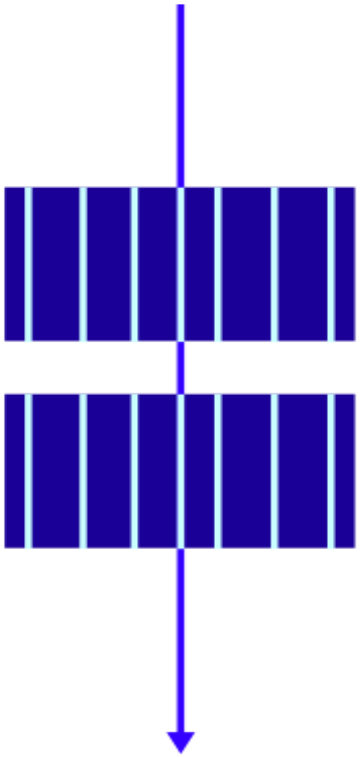
UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Construtor for/do

- Construtor “for/do”
  - Exemplo

```
#pragma omp parallel default(none)\
        shared(n,a,b,c,d) private(i)
  {
    #pragma omp for nowait
      for (i=0; i<n-1; i++)
          b[i] = (a[i] + a[i+1])/2;
    #pragma omp for nowait
      for (i=0; i<n; i++)
          d[i] = 1.0/c[i];

  } /*-- End of parallel region --*/
                          (implied barrier)
```



Instituto de Informática

OpenMP:
Uma Introdução

Cláudio Geyer

UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

Informática
UFRGS

Informática UFRGS

Resumo

- Uma especificação para programação paralela
- Mantida por grupo de empresa e instituições
- Em memória compartilhada
- Misto de diretivas, compilador, runtime, ferramentas
- Estende versão sequencial (programa) de problema
  - Mantendo código único eficiente
- Principal recurso de paralelização
  - Paralelização de loops
  - Uma thread para cada grupo de passos
- Controle de nível mínimo de paralelização (cláusula *if*)

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Informática UFRGS

Resumo

- Outras diretivas:
  - Regiões paralelas
  - Compartilhamento de variáveis
  - Barreiras implícitas e explícitas
  - *for, sections e single*

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Exercícios

- Exercícios:
  - A)

Revisão

- Revisão

Informática UFRGS

Referências

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL