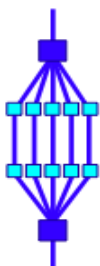


# Instituto de Informática

## OpenMP: Uma Introdução

*Cláudio Geyer*

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

# Sumário

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

# Organizações OpenMP

The OpenMP logo features the word "OpenMP" in a large, teal, sans-serif font. A horizontal teal bar is positioned above the text, and another horizontal teal bar is positioned below the "Open" part of the text. The letters "MP" are significantly larger than "Open". A small "TM" trademark symbol is located to the right of the "P".

<http://www.openmp.org>

The OMPunity logo consists of the word "OMPunity" in a teal, sans-serif font. The "OMP" part is larger and bolder than the "unity" part. A horizontal tan bar is positioned above the text, and another horizontal tan bar is positioned below the text.

<http://www.compunity.org>

# OpenMP: 1ª página



The screenshot shows the OpenMP.org website in a web browser. The browser's address bar displays <http://openmp.org/wp/>. The website features a large "OpenMP" logo with the tagline "THE OPENMP API SPECIFICATION FOR PARALLEL PROGRAMMING".

**OpenMP News**

»IWOMP 2009 in June - Important Dates

International Workshop on OpenMP  
IWOMP 2009  
Evolving OpenMP in an Age of Extreme Parallelism

June 3rd - June 5th, 2009  
Dresden, Germany  
<http://www.iwomp.org>

The 2009 International Workshop on OpenMP (IWOMP 2009) will be held on the campus of Technische Universität Dresden, Germany. IWOMP is the premier event focusing on parallel programming with OpenMP. The workshop serves as a forum to present the latest research ideas and results related to this shared memory programming model. It also offers the opportunity to interact with OpenMP users, developers and the people working on the next release of the standard.

The first day consists of tutorials focusing on topics of interest to current and prospective OpenMP developers, suitable for both beginners as well as those interested in learning of recent developments in the evolving OpenMP standard.

The second and third day consists of two keynotes, one invited talk, 15 technical papers and a poster session during which research ideas and results will be presented and discussed. The keynotes "Is OpenMP the right approach for future generation architectures?" by Jose Duato Marin, Universidad Politecnica de Valencia and "Can OpenMP be extended to deal with Hardware Accelerator?" by François Bodin, CAPS Enterprise provide the framework for the other presentations.

To enable the exchange of information regarding latest developments it is still possible to submit a poster for all registered workshop participants until May 20 via email to [iwomp09@zih.tu-dresden.de](mailto:iwomp09@zih.tu-dresden.de).

**Important Dates**

APRIL 22: Downtown Hotel Deadline  
APRIL 24: Guesthouse Deadline  
MAY 18: Early Registration Deadline  
MAY 20: Poster Deadline  
MAY 30: Late Registration Deadline  
Tutorial and Workshop in Dresden: June 3-5, 2009

Posted on April 21, 2009

»Download Book Examples and Discuss

**The OpenMP API**

supports multi-platform shared-memory parallel programming in C/C++ and Fortran. OpenMP is a portable, scalable model with a simple and flexible interface for developing parallel applications on platforms from the desktop to the supercomputer.

»Read about OpenMP

**Get**

»OpenMP specs

**Use**

»OpenMP Compilers

**Learn**

Using OpenMP

PORTABLE SHARED MEMORY PARALLEL PROGRAMMING

SARABHA CHAKRABARTY, GABRIEL CILIC, AND KENNETH R. COLE

»Using OpenMP – the book  
»Using OpenMP – the examples  
»Using OpenMP – the forum  
»Wikipedia  
»OpenMP Tutorial  
»More Resources

**Discuss**

**Subscribe to the News Feed**

» Specifications

» About OpenMP

» Compilers

» Resources

» Discussion Forum

**Events**

The 5th International Workshop on OpenMP - Evolving OpenMP in an Age of Extreme Parallelism - will take place June 3-5, 2009 in Dresden, Germany.

» [iwomp.org](http://iwomp.org)

ISC 09 - Hamburg, Germany - June 23-26.

» [www.supercomp.de/isc09/](http://www.supercomp.de/isc09/)

**Input Register**

Alert the OpenMP.org webmaster about new products, events, or updates and we'll post it here.

» [webmaster@openmp.org](mailto:webmaster@openmp.org)

**Search OpenMP.org**

Google Custom Search

Search

**Archives**

- April 2009
- March 2009
- February 2009
- January 2009
- November 2008



# OpenMP Community: 1ª página

The screenshot shows the OpenMP Community website. The browser's address bar displays <http://www.compunity.org/>. The website features a dark blue header with the 'compunity' logo and a navigation bar with links to 'compunity', 'OpenMP ARB', and 'OpenMP Specification'. A left sidebar contains a 'Home' menu with links to News, ARB (Architecture Review Board), Specifications, Bylaws, Resources, Training, Research, Events, Futures, and Contacts. The main content area is titled 'The Community of OpenMP Users, Researchers, Tool Developers and Providers' and is divided into three sections: 'News...', 'About...', and 'Past...'. The 'News...' section includes links to 'OpenMP 3.0 specifications released!' and 'The OpenMP ARB's Website has been considerably overhauled', with accompanying text. The 'About...' section describes the community and the development of OpenMP. The 'Past...' section mentions the 'SC08, USA, Texas, Austin Convention Center, November 15 -21, 2008'. The browser's bookmark bar shows various links, and the status bar at the bottom indicates the page is from 'UFRGS'.

**compunity**

hosted by  
**RWTH AACHEN UNIVERSITY**  
Center for Computing and Communication

COMPunity OpenMP ARB OpenMP Specification

**The Community of OpenMP Users, Researchers, Tool Developers and Providers**

**News...**

[OpenMP 3.0 specifications released!](#)

The final 3.0 specs is available. The major achievement is the new tasking concept, which considerably increases OpenMP's applicability for a large class of applications. Furthermore it improves the support of nested parallelization and includes quite a few clarifications and corrections.

[The OpenMP ARB's Website has been considerably overhauled](#)

Please find plenty of up-to-date information around OpenMP there.

**Soon...**

[International Workshop on OpenMP IWOMP 2009 3rd to 5th of June in Dresden](#)

The 5th International Workshop on OpenMP: Evolving OpenMP in an Age of Extreme Parallelism - will take place in [Dresden](#) (Germany) from the **3rd June through the 5th June 2009**. The workshop will be hosted by the [Center for Information Services & High Performance Computing \(ZIH\)](#) at the Dresden University of Technology. To ensure the IWOMP 2009 attendees are able to find accommodations that best suit their needs, we have compiled an overview of many of the hotels and guest houses in Dresden. This overview also includes estimates of the time it takes to reach the university from the various accommodations. The following clickable map will show you detailed information about the accommodation chosen by you. Please find further information here under [accommodation](#)

[>>>](#)

**Past...**

[SC08, USA, Texas, Austin Convention Center, November 15 -21, 2008](#)

**About...**

**compunity** is the community of OpenMP researchers and developers in academia and industry. It is a forum for the dissemination and exchange of information about OpenMP. compunity is also a forum for discussing our experiences with this programming API and for debating ideas that might improve it. **compunity** was formally incorporated as a non-profit organization on November 14, 2001

**OpenMP** was developed by a consortium of computer vendors to enable the creation of portable, high-level shared memory parallel programs under Fortran, C and C++.

The **compunity website** will provide up-to-date information on OpenMP-related news and events. It is a repository of resources for learning about OpenMP, reading about the experiences of application developers who use the language, and for finding out about recent research activities. It will also be the site for obtaining OpenMP freeware and benchmarks.

# 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



# 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

# 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 portátil
  - 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

# 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



# 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

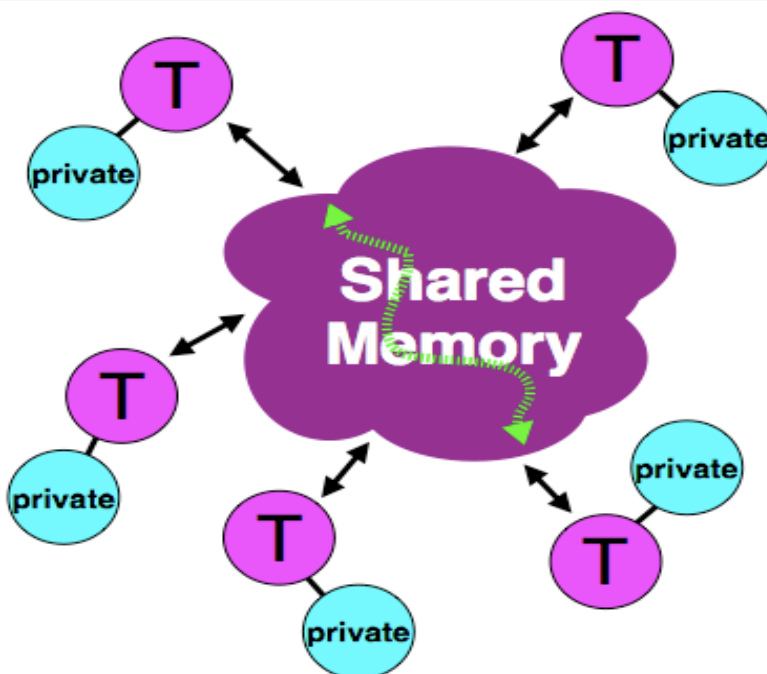
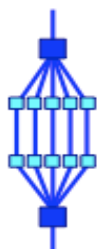


# Modelo de Memória de OpenMP

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

RvdP/V1

An Overview of OpenMP





# 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





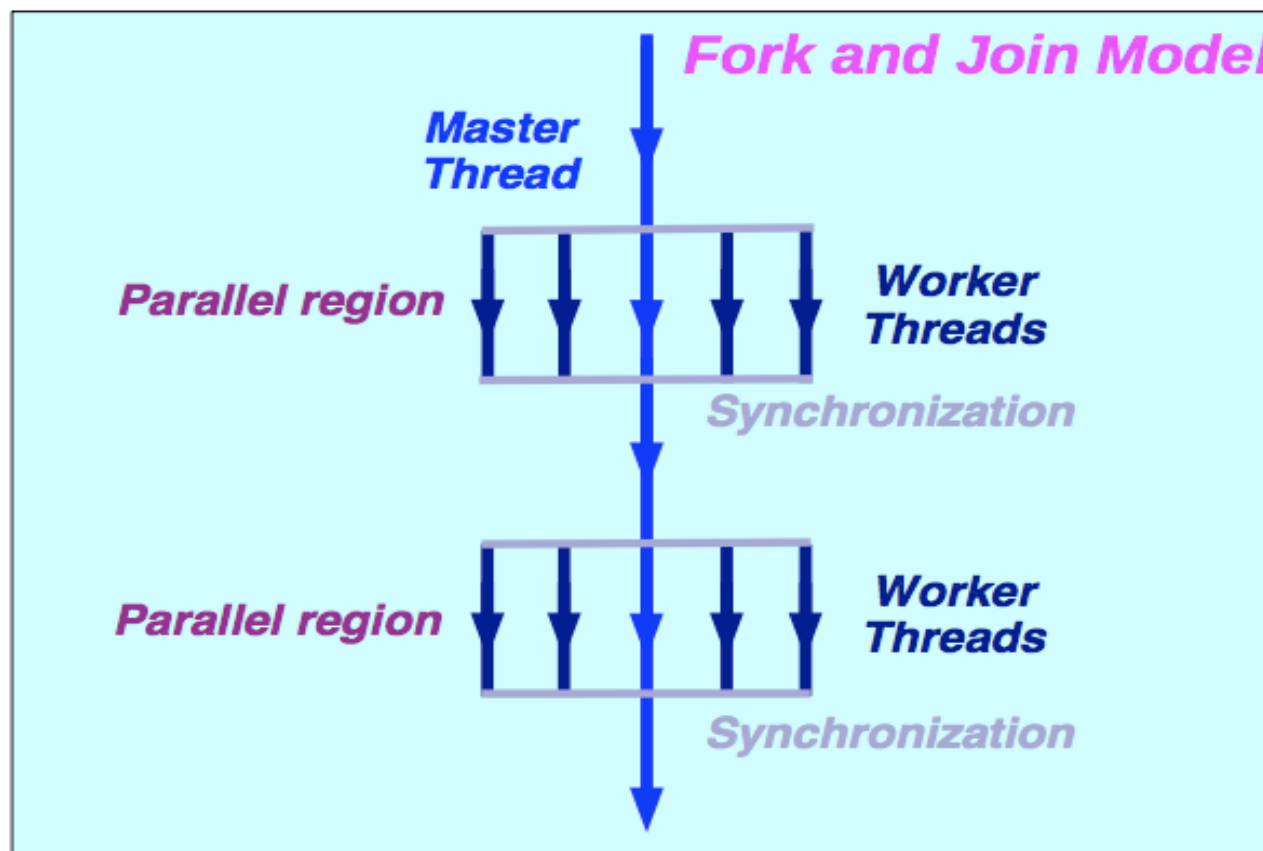
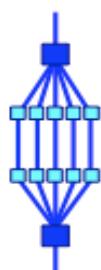
# Atributos de Dados Compartilhados

- 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

## 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

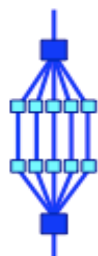
- Exemplo de execução de loop
  - Ver próximo slide com figura
  - 5 threads (fluxos de execução)
    - Numeradas de 0 a 4
  - Código único nas threads
  - Vetor de 1000 elementos
  - Cada thread operando sobre um subconjunto distinto de partes do vetor
    - Vetor parcial com índices consecutivos
 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

# Execução Paralela do Exemplo

NTU Talk  
January 14  
2009

14

## Example parallel execution



| Thread 0<br>i=0-199 | Thread 1<br>i=200-399 | Thread 2<br>i=400-599 | Thread 3<br>i=600-799 | Thread 4<br>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]                  |

RvdP/V1

An Overview of OpenMP



# 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

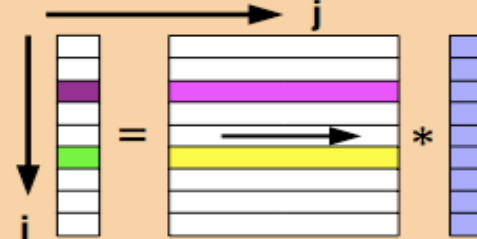
NTU Talk  
January 14  
2009

16

### Example - Matrix times vector



```
#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 =  $\sum b[i=0][j]*c[j]$

a[0] = sum

i = 1

sum =  $\sum b[i=1][j]*c[j]$

a[1] = sum

TID = 1

for (i=5,6,7,8,9)

i = 5

sum =  $\sum b[i=5][j]*c[j]$

a[5] = sum

i = 6

sum =  $\sum b[i=6][j]*c[j]$

a[6] = sum

... etc ...

RvdP/V1

An Overview of OpenMP



# Exemplo de Avaliação de Desempenho

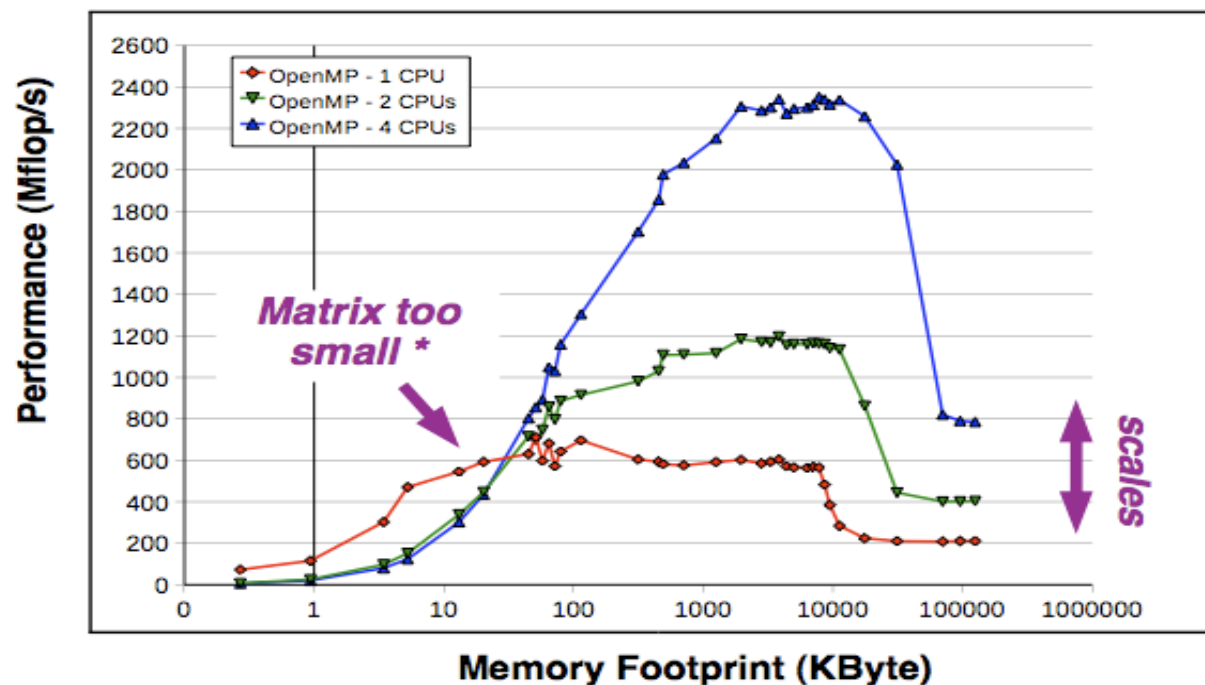
- Exemplo de avaliação de desempenho
  - Execução do mesmo problema com variação do número de cpus
  - Uma curva para cada quantidade de cpus
  - Eixo X:
    - Tamanho do problema
  - Eixo Y:
    - Tempo de execução (tempo paralelo)
  - Notar a perda de desempenho para entradas pequenas
    - 1 core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

# Desempenho de OpenMP

NTU Talk  
January 14  
2009

17

## OpenMP performance



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

RvdP/V1

An Overview of OpenMP



# 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
      - $n > \text{limit}$
  - 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

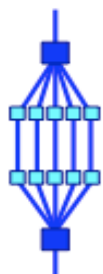
# Um Exemplo Sintético +

NTU Talk  
January 14  
2009

18



## A more elaborate example



```
#pragma omp parallel if (n>limit) default(none) \
    shared(n,a,b,c,x,y,z) private(f,i,scale)
```

```
{
    f = 1.0;
```

```
#pragma omp for nowait
```

```
    for (i=0; i<n; i++)
        z[i] = x[i] + y[i];
```

```
#pragma omp for nowait
```

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

```
#pragma omp barrier
```

```
    ....
    scale = sum(a,0,n) + sum(z,0,n) + f;
    ....
```

```
} /*-- End of parallel region --*/
```

Statement is executed  
by all threads

parallel loop  
(work is distributed)

parallel loop  
(work is distributed)

synchronization

Statement is executed  
by all threads

parallel region



# OpenMP em mais Detalhes

# Súmula

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

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

# 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 --*/
```

# 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 --*/
```

# 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 --*/
```



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

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

***wait !***

***barrier***

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

## Barreira /3

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

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

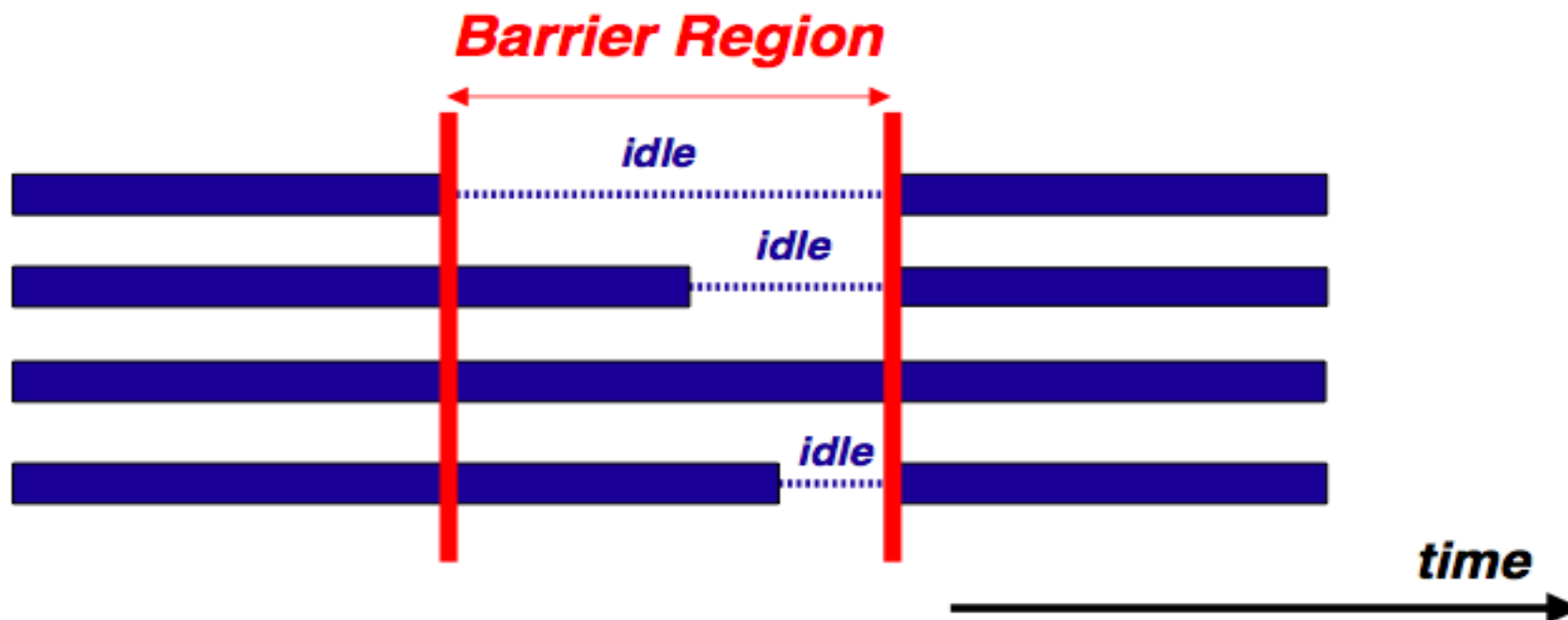
*wait !*

*barrier*

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

## 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]
```

# Construtor for/do

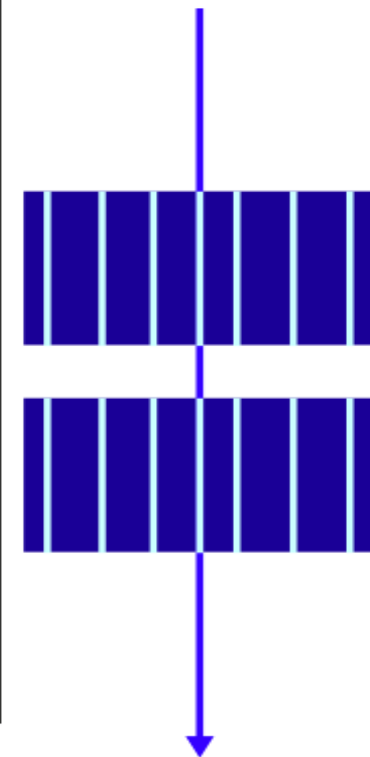
- Construtor “for/do”
  - Cláusulas suportadas
    - private
    - firstprivate
    - lastprivate
    - reduction
    - ordered
    - schedule
    - nowait

# 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*



# 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*)

# Resumo

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

# Exercícios

- Exercícios:
  - A)

# Revisão

- Revisão

# Referências