OpenMP: Sincronização de Threads e Impacto no Desempenho

Paulo Sérgio Lopes de Souza
pssouza@icmc.usp.br

Universidade de São Paulo / ICMC / SSC – São Carlos
Laboratório de Sistemas Distribuídos e Programação Concorrente

CNPq Conselho Nacional de Desenvolvimento Científico e Tecnológico
CAPES
FAPESP

Barreiras em C/OMP

- Barreira: ***#pragma omp barrier***
  - Força todas as *threads* esperarem até que todas elas tenham atingido a barreira
  - As execuções das *threads* continuam após a última *thread* chegar
  - Evita, por exemplo, que uma *thread* acesse prematuramente dados que são definidos por outra(s) *thread(s)*

- Exemplos de uso
  - Barreira é necessária se mapeamento de iterações às *threads* não for preservado entre os dois *for's*
  - Versões mais novas do OpenMP garantem esse mapeamento (escalonamento *static*)
  - Junção dos dois loops pode eliminar a necessidade da barreira

```
 1 #pragma parallel shared(n,a,b)
 2 {
 3    #pragma omp for
 4    for (int i=0; i<n; i++)
 5        a[i] = i;
 6
 7    // The implied barrier on the for-loop above is needed there
 8
 9    #pragma omp for nowait
10    for (int i=0; i<n; i++)
11        b[i] += a[i];
12 } // End of parallel region
```

```
1 #pragma parallel for shared(n,a,b)
2 for (int i=0; i<n; i++)
3 {
4     a[i] = i;
5     b[i] += a[i];
6 } // End of the parallel region
```

Van der Pas et al. (2017)

Limitando Execuções a uma *Thread* Apenas

- Lembrando... Diretiva ***single***: ***#pragma omp single***

#pragma omp single *[clause[[,]clause] ...]*
structured-block
clause:
private(*list*)
firstprivate(*list*)
copyprivate(*list*)
nowait

- Diretiva ***master***: ***#pragma omp master***
  - Semelhante à diretiva *single*, porém, bloco estruturado executado pela *thread* mestre

```
int main( )
{
   int a[5], i;

   #pragma omp parallel
   {
      // Perform some computation.
      #pragma omp for
      for (i = 0; i < 5; i++)
         a[i] = i * i;

      // Print intermediate results.
      #pragma omp master
         for (i = 0; i < 5; i++)
            printf("a[%d] = %d\n", i, a[i]);

      // Wait.
      #pragma omp barrier

      // Continue with the computation.
      #pragma omp for
      for (i = 0; i < 5; i++)
         a[i] += i;
   }
}
```

https://docs.microsoft.com/pt-br/cpp/parallel/openmp/reference/openmp-directives?view=vs-2019#master

Protegendo a Região Crítica com Critical e Atomic

- Relembrando as diretivas *critical* e *atomic*

```
1 #pragma parallel
2 {
3               ...
4    #pragma omp critical (c_region1)
5    {
6       sum1 += ...
7    }
8               ...
9    #pragma omp critical (c_region2)
10   {
11      sum2 += ...
12   }
13                ...
14 } // End of the parallel region
```

```
1 #pragma omp parallel
2 {
3       ......
4    #pragma omp atomic
5        x += 1;
6       ......
7 } // End of parallel region
```

Van der Pas et al. (2017)

OMP: Estrutura de um programa C

- Cláusula ***nowait***
  - Remove a barreira implícita no final das diretivas *for*, *sections* e *single*

  - No exemplo abaixo a variável *name* é buscado em duas listas
  - Não há necessidade de esperar todas as *threads* do primeiro *for* terminarem para procurar na segunda lista

```
1       #pragma omp parallel
2       {
3           #pragma omp for nowait
4               for (i = 0; i < nmax; i++)
5                   if (isEqual(name, current_list[i])
6                       processCurrentName(name);
7           #pragma omp for
8               for (i = 0; i < mmax; i++)
9                   if (isEqual(name, past_list[i])
10                      processPastName(name);
11      }
```

Grama et al. (2003)

OMP: Estrutura de um programa C

- *Locks* no OpenMP

Function name	Description
`omp_init_lock`	Initialize a lock variable.
`omp_init_nest_lock`	Similar, but for a nestable lock.
`omp_set_lock`	Blocking request to acquire the lock.
`omp_set_nest_lock`	Similar, but for a nestable lock.
`omp_unset_lock`	Release the lock.
`omp_unset_nest_lock`	Similar, but for a nestable lock.
`omp_destroy_lock`	Change the state of the lock to be unitialized.
`omp_destroy_nest_lock`	Similar, but for a nestable lock.
`omp_test_lock`	Non-blocking request to acquire the lock.
`omp_test_nest_lock`	Similar, but for a nestable lock.

Van der Pas et al. (2017)

OMP: Estrutura de um programa C

- *Locks* no OpenMP: ***omp_set_lock*** e ***omp_unset_lock***

```
#include <stdio.h>
#include <omp.h>

omp_lock_t my_lock;

int main() {
   omp_init_lock(&my_lock);

   #pragma omp parallel num_threads(4)
   {
      int tid = omp_get_thread_num( );
      int i, j;

      for (i = 0; i < 5; ++i) {
         omp_set_lock(&my_lock);
         printf("Thread %d - starting locked region\n", tid);
         printf("Thread %d - ending locked region\n", tid);
         omp_unset_lock(&my_lock);
      }
   }

   omp_destroy_lock(&my_lock);
}
```

https://docs.microsoft.com/pt-br/cpp/parallel/openmp/reference/openmp-functions?view=vs-2019#omp-init-lock

OMP: Estrutura de um programa C

- *Locks* no OpenMP: ***omp_set_lock*** e ***omp_unset_lock***

```
#include <stdio.h>
#include <omp.h>

omp_nest_lock_t my_lock;
void Test() {
   int tid = omp_get_thread_num( );
   omp_set_nest_lock(&my_lock);
   printf("Thread %d - starting nested locked region\n", tid);
   printf("Thread %d - ending nested locked region\n", tid);
   omp_unset_nest_lock(&my_lock);
}

int main() {
   omp_init_nest_lock(&my_lock);

   #pragma omp parallel num_threads(4)
   {
      int i, j;
      for (i = 0; i < 5; ++i) {
          omp_set_nest_lock(&my_lock);
           if (i % 3)
              Test();
          omp_unset_nest_lock(&my_lock);
        }
    }
    omp_destroy_nest_lock(&my_lock);
}
```

https://docs.microsoft.com/pt-br/cpp/parallel/openmp/reference/openmp-functions?view=vs-2019#omp-init-lock

OMP: Estrutura de um programa C

- Exemplos de códigos com diferentes sincronizações
  - Encontrar o maior elemento em um vetor de inteiros
    - Sequencial: veja tempo de resposta
    - OpenMP:
      - 1ª Versão: variável maior compartilhada;
      - 2ª Versão: variável maior compartilhada, mas código otimizado;
      - 3ª Versão: variável maior como um vetor para as *threads*; e
      - 4ª Versão: variável maior obtida com cláusula reduce.

- Exercício solicitado aos grupos
  - Ver especificação disponibilizada no edisciplinas

Referências

Pas, Ruud van der.; Stotzer, Eric; Terboven, Christian. ***Using OpenMP-the next step: affinity, accelerators, tasking and SIMD***. The MIT Press, Cambridge, MA, 2017. ISBN 9780262534789

GRAMA,A.; KUMAR, U.; GUPTA,A.; KARYPIS, G. Introduction to Parallel Computing, 2nd Edition, 2003.

OpenMP API 5.0 C/C++ Syntax Quick Reference Card. 2018. Disponível em https://www.openmp.org/wp-content/uploads/OpenMPRef-5.0-111802-web.pdf . Último acesso em 28/09/2020.

OpenMP Application Program Inferface, Version 5.0 – 2018, Disponível em https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-5.0.pdf. Último acesso em 28/09/2020.

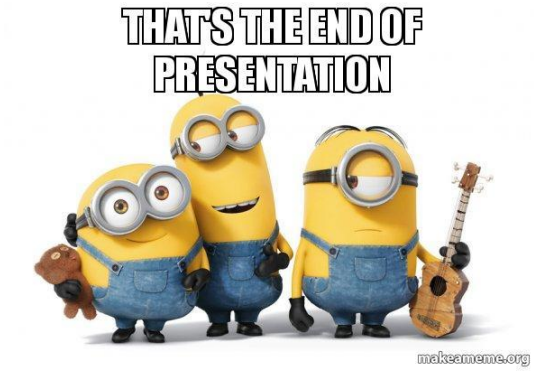
Reinders, James (Ed.) The Parallel Universe: flow Graphs, Speculative Locks, and Tasks Arenas. Intel. 2014. Disponível em https://www.openmp.org/about/whos-using-openmp/ Útlimo acesso em 28/09/2020

OpenMP Resources. NERSC Documentation (National Energy Research Scientific Computing Center. Berkeley Lab. Disponível em: https://docs.nersc.gov/development/programming-models/openmp/openmp-resources/. Último acesso em 28/09/2020.

OpenMP: Sincronização de Threads e Impacto no Desempenho

Paulo Sérgio Lopes de Souza
pssouza@icmc.usp.br

Universidade de São Paulo / ICMC / SSC – São Carlos
Laboratório de Sistemas Distribuídos e Programação Concorrente



CNPq Conselho Nacional de Desenvolvimento Científico e Tecnológico
CAPES
FAPESP