Parallel Computing

Academic year – 2020/21, spring semester

Computer science

Lecture 6

Lecturer, instructor:

Balakshin Pavel Valerievich

(pvbalakshin@itmo.ru; pvbalakshin@hdu.edu.cn)

Assistant:

Liang Tingting

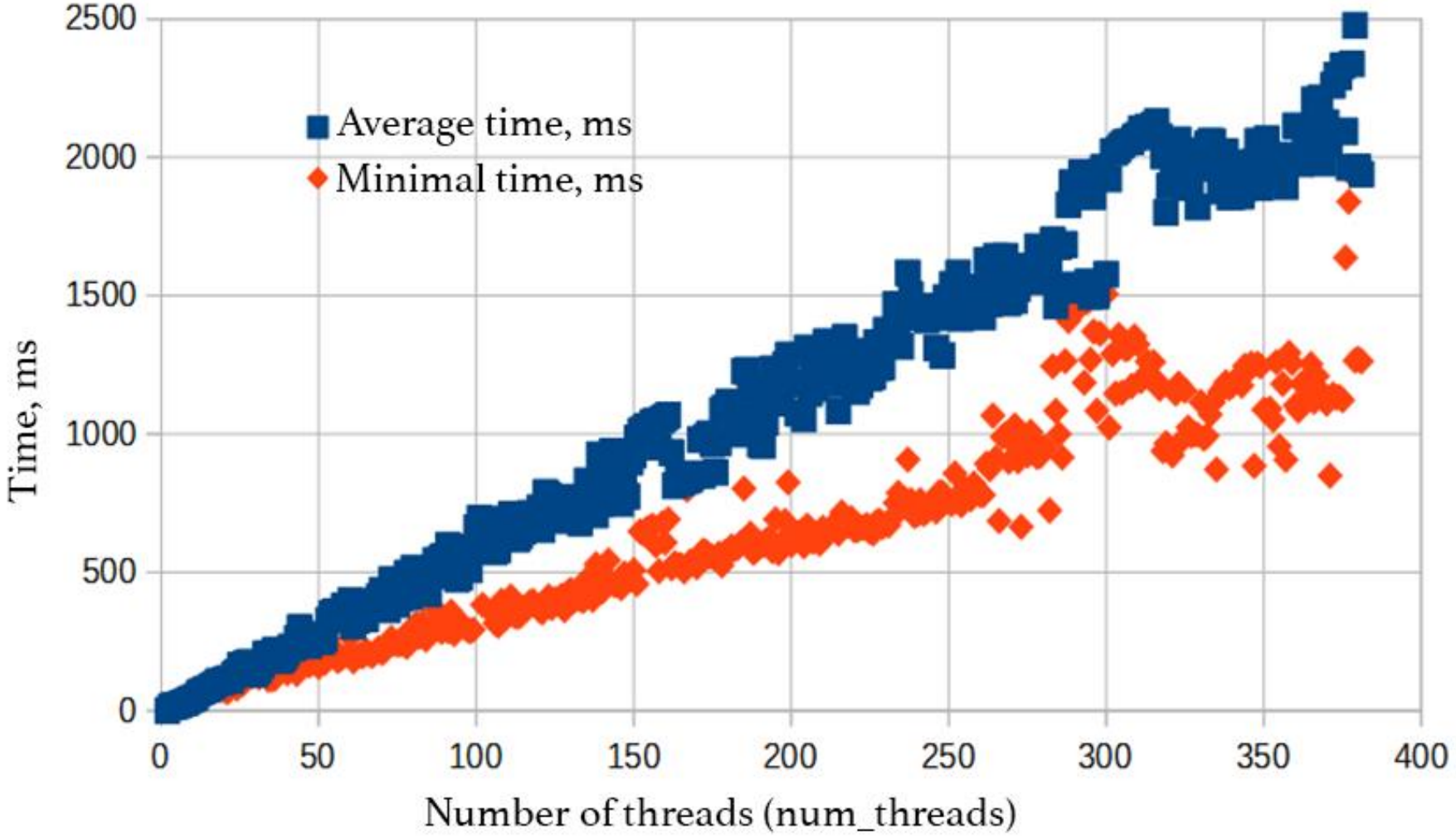
(liangtt@hdu.edu.cn)

Measurement of overhead costs for thread creation in OpenMP

```
for (i = 1; i < 382; i++) {
   omp_set_num_threads(i);
   gettimeofday(&T1, NULL);
   #pragma omp parallel
   #pragma omp master
      s++;
   gettimeofday(&T2, NULL);
   print_delta(T2, T1);
}
```

The omp_get_max_threads function allows to know the maximum number of threads (in the example this number is 381).

Measurement of overhead costs for thread creation in OpenMP (2)



Nested parallelism

Example 1

```
omp_set_nested(1);
#pragma omp parallel num_threads(2)
{
   putchar("!");
   #pragma omp parallel num_threads(3)
   putchar("!");
}
```

Example 2

```
omp_set_num_threads(5);
#pragma omp parallel for
for (i = 0; i < 10; i++) {
  …
   #pragma omp parallel for
   for (j = 0; j < 10; j++) {
      putchar("!");
   }
  …
}
```

Nested parallelism (2)

```
omp_set_num_threads(5);
#pragma omp parallel for collapse(2)
for (i = 0; i < 10; i++) {
   ...
  for (j = 0; 0 < 10; j++) {
    putchar("!");
    printf("%d.%d\n", i, j);
  }
   ...
}
```

```
omp_set_num_threads(5);
#pragma omp parallel for
for (int ij = 0; ij < 10*10; ++ij) {
    int i = ij / 10;
    int j = ij % 10;
    ...
    putchar("!");
    printf("%d.%d\n", i, j);
}
```

Since OpenMP 3.0 only!

Sequential code inside parallel area

Example

```
#pragma omp parallel
 {
   parallel code;
   #pragma omp single
   sequential code (with anticipation of the others);
   ????
   parallel code;
   #pragma omp master
   sequential code (without anticipation of the others);
   parallel code;
}
```

Forced synchronization of threads in parallel area

```
#pragma omp parallel
{
   printf(“Message #1\n");
   printf("Message #2\n");
   #pragma omp barrier
   printf("Message #3\n");
}
```

```
Message #1
Message #2
Message #2
Message #1
Message #1
Message #2
Message #1
Message #2
Message #3
Message #3
Message #3
Message #3
```

All threads have to complete one task prior to execute the next one.

Forced synchronization of threads in parallel area (2)

```
#pragma omp parallel
{
   #pragma omp for nowait
   for (i = 1; i < 100; i++)
   compute_something();
   #pragma omp for
   for (j = 1; j < 200; j++)
   more_computations();
}
```

Usually pragma omp for loops have hidden barrier.
Nowait allows free threads to execute next tasks.

private-variables initialization

```
int n = 1;
printf("Value of n in the beginning: %d\n", n);
#pragma omp parallel firstprivate(n)
{
   printf("Value of n in thread (in the beginning): %d\n", n);
   n = omp_get_thread_num();
   printf("Value of n in thread (at the end): %d\n", n);
}
printf("Value of n at the end: %d\n", n);
```

```
Value of n in the beginning: 1
Value of n in thread (in the beginning): 1
Value of n in thread (at the end): 1
Value of n in thread (in the beginning): 1
Value of n in thread (at the end): 0
Value of n in thread (in the beginning): 1
Value of n in thread (at the end): 2
Value of n in thread (in the beginning): 1
Value of n in thread (at the end): 3
Value of n at the end: 1
```

private-variables storage

```
int n = 0;
printf("Value of n in the beginning: %d\n", n);
#pragma omp parallel for lastprivate(n)
for (i = 0; i < 6; i++) {
   n = i * 10;
   printf("Value of n in thread (iteration): %d\n", n);
}
printf("Value of n at the end: %d\n", n);
```

lastprivate can be used in
“for loops” and sections only!

```
Value of n in the beginning: 0
Value of n in thread (iteration): 40
Value of n in thread (iteration): 0
Value of n in thread (iteration): 10
Value of n in thread (iteration): 50
Value of n in thread (iteration): 20
Value of n in thread (iteration): 30
Value of n at the end: 50
```

Sequential omp for loops

What should be executed faster?

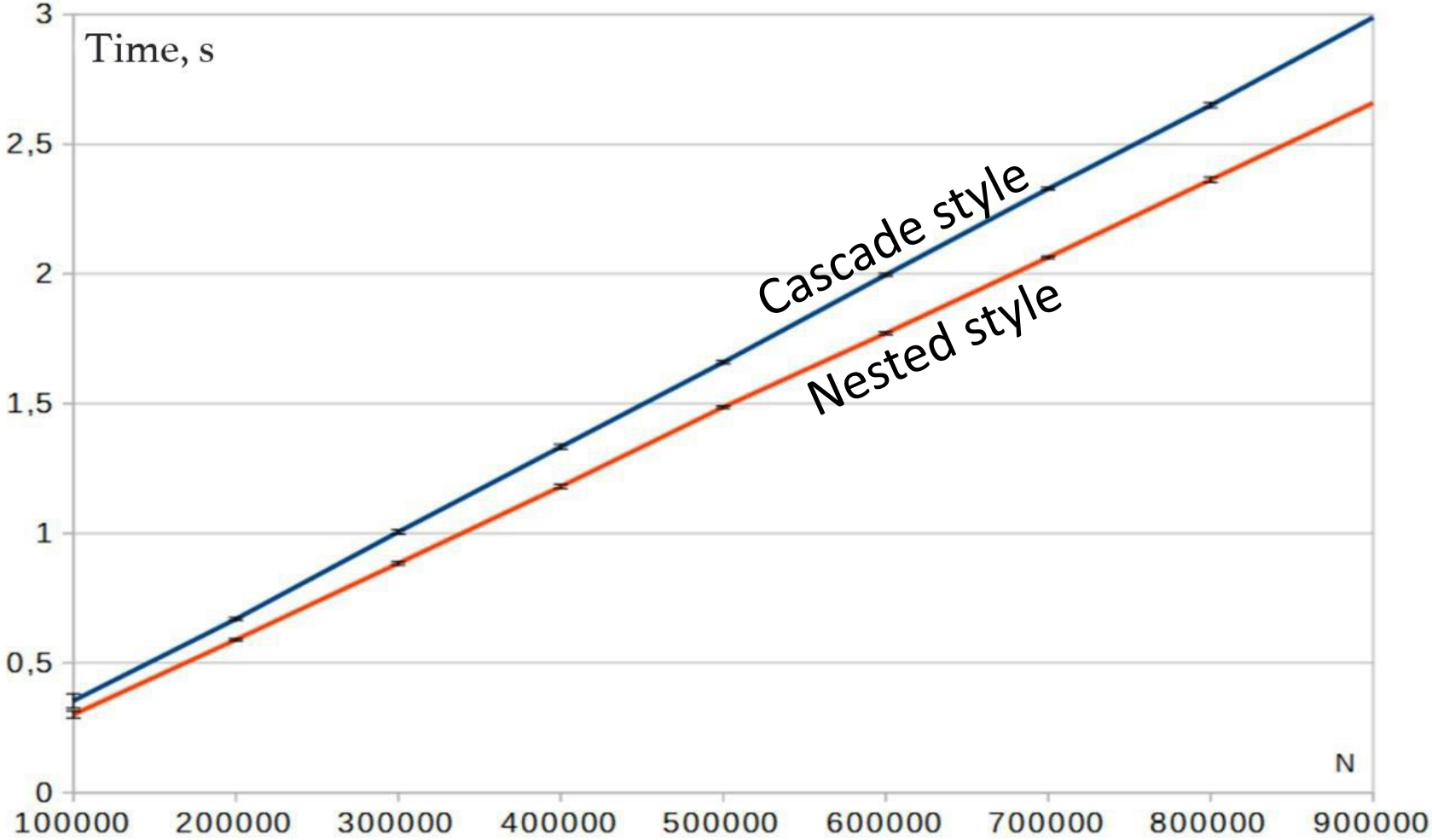
Cascade (waterfall) style

```
for (i = 0; i < N; ++i) {
   #pragma omp parallel for reduction(+:s)
   for (j = 0; j < 100; ++j) s=1;
   #pragma omp parallel for reduction(+:s)
   for (j = 0; j < 100; ++j) s=1;
}
```

Nested style

```
for (i = 0; i < N; ++i) {
   #pragma omp parallel
   {
      #pragma omp for reduction(+:s)
      for (j = 0; j < 100; ++j) s=1;
      #pragma omp for reduction(+:s)
      for (j = 0; j < 100; ++j) s=1;
   }
}
```

Sequential omp for loops (2)



‘parallel’ directive absence

INCORRECT:

```
#pragma omp for
for (int i = 1; i < 10; i++)
    … // code
```

CORRECT:

```
#pragma omp parallel for
for (int i = 1; i < 10; i++)
    … // code
```

CORRECT:

```
#pragma omp parallel
{
    #pragma omp for
    for (int i = 1; i < 10; i++)
        … // code
}
```

According to A. Kolosov, A. Karpov, E. Ryzhikov “32 OpenMP traps and pitfalls when programming in C++”: https://www.viva64.com/ru/a/0054/

‘omp’ directive absence

INCORRECT:

```
#pragma omp parallel num_threads(2)
{
   #pragma single
   {
      printf("Me\n");
   }
}
```

CORRECT:

```
#pragma omp parallel num_threads(2)
{
   #pragma omp single
   {
      printf("Me\n");
   }
}
```

According to A. Kolosov, A. Karpov, E. Ryzhikov “32 OpenMP traps and pitfalls when programming in C++”

‘for’ directive absence

INCORRECT:

```
#pragma omp parallel num_threads(2)
for (int i = 1; i < 10; i++)
    … // code
```

CORRECT:

```
#pragma omp parallel for num_threads(2)
for (int i = 1; i < 10; i++)
    … // code
```

According to A. Kolosov, A. Karpov, E. Ryzhikov “32 OpenMP traps and pitfalls when programming in C++”

Useless parallelism

INCORRECT:

```
#pragma omp parallel num_threads(2)
{
    … // N code lines
    #pragma omp parallel for
    for (int i = 1; i < 10; i++)
        function1();
}
```

CORRECT:

```
#pragma omp parallel num_threads(2)
{
    … // N code lines
    #pragma omp for
    for (int i = 1; i < 10; i++)
        function1();
}
```

According to A. Kolosov, A. Karpov, E. Ryzhikov “32 OpenMP traps and pitfalls when programming in C++”

Redefining the number of threads inside a parallel section

INCORRECT:

```
#pragma omp parallel
{
   omp_set_num_threads(2);
   #pragma omp for
   for (int i = 0; i < 10; i++)
      function1();
}
```

CORRECT (2 options):

```
#pragma omp parallel num_threads(2)
{
   #pragma omp for
   for (int i = 0; i < 10; i++)
      function1();
}
```

```
omp_set_num_threads(2);
#pragma omp parallel
{
   #pragma omp for
   for (int i = 0; i < 10; i++)
      function1();
}
```

According to A. Kolosov, A. Karpov, E. Ryzhikov “32 OpenMP traps and pitfalls when programming in C++”

Undeclared local variables

INCORRECT:

```
int a = 0;
#pragma omp parallel private(a)
{
   a++;
}
```

CORRECT:

```
int a = 0;
#pragma omp parallel private(a)
{
   int a = 0;
   a++;
}
```

According to A. Kolosov, A. Karpov, E. Ryzhikov “32 OpenMP traps and pitfalls when programming in C++”

Local variables are not marked as they should

INCORRECT:

```
size_t i;
#pragma omp parallel sections num_threads(2)
{
   #pragma omp section
   for (i = 0; i != arraySize; ++i)
      array[i].a = 1;
   #pragma omp section
   for (i = 0; i != arraySize; ++i)
      array[i].b = 2;
}
```

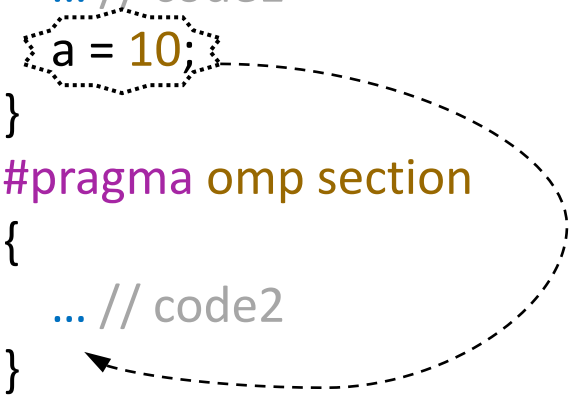
CORRECT:

```
#pragma omp parallel sections num_threads(2)
{
   #pragma omp section
   for (size_t i = 0; i != arraySize; ++i)
      array[i].a = 1;
   #pragma omp section
   for (size_t i = 0; i != arraySize; ++i)
      array[i].b = 2;
}
```

According to A. Kolosov, A. Karpov, E. Ryzhikov “32 OpenMP traps and pitfalls when programming in C++”

Careless use of lastprivate

```
int a = 1;
#pragma omp parallel
{
   #pragma omp sections lastprivate(a)
   {
      #pragma omp section
      {
         … // code1
         a = 10;
      }
      #pragma omp section
      {
         … // code2
      }
   }
 #pragma omp barrier
}
```



Simultaneous usage of shared resource

INCORRECT:

```
#pragma omp parallel num_threads(2)
{
   printf(“Hello World!");
}
```

→ HellHell oo WorWlodrl
d

CORRECT:

```
#pragma omp parallel num_threads(2)
{
   #pragma critical
   printf(“Hello World!");
}
```

According to A. Kolosov, A. Karpov, E. Ryzhikov “32 OpenMP traps and pitfalls when programming in C++”

Unprotected access to shared memory

INCORRECT:

```
int a = 0;
#pragma omp parallel
{
   a++;
}
```

CORRECT (2 options):

```
int a = 0;
#pragma omp parallel
{
   #pragma omp atomic
   a++;
}
```

```
int a = 0;
#pragma omp parallel reduction(+:a)
{
   a++;
}
```

According to A. Kolosov, A. Karpov, E. Ryzhikov “32 OpenMP traps and pitfalls when programming in C++”