HPC.NRW

Introduction to OpenMP

Dr. Christian Terboven

Tasking and Scoping

Dr. Christian Terboven

Introduction to OpenMP

Tasks in OpenMP: Data Scoping

HPC.NRW

- Some rules from *Parallel Regions* apply:
  - Automatic Storage (local) variables are private
  - Static and Global variables are shared

- Tasking:
  - Variables are `firstprivate` unless `shared` in the enclosing context
    - Only `shared` attribute is inherited
    - Exception: Orphaned Task variables are `firstprivate` by default!
      - See an example later on

Example: Data Scoping

Dr. Christian Terboven

Introduction to OpenMP

Data Scoping Example (1/7)

HPC.NRW

```
 1  int a = 1;
 2  void foo()
 3  {
 4      int b = 2, c = 3;
 5      #pragma omp parallel private(b)
 6      {
 7          int d = 4;
 8          #pragma omp task
 9          {
10              int e = 5;
11
12              // Scope of a:
13              // Scope of b:
14              // Scope of c:
15              // Scope of d:
16              // Scope of e:
17  } } }
```

Data Scoping Example (2/7)

```
 1  int a = 1;
 2  void foo()
 3  {
 4      int b = 2, c = 3;
 5      #pragma omp parallel private(b)
 6      {
 7          int d = 4;
 8          #pragma omp task
 9          {
10              int e = 5;
11
12              // Scope of a: shared
13              // Scope of b:
14              // Scope of c:
15              // Scope of d:
16              // Scope of e:
17  } } }
```

Data Scoping Example (3/7)

HPC.NRW

```
 1  int a = 1;
 2  void foo()
 3  {
 4      int b = 2, c = 3;
 5      #pragma omp parallel private(b)
 6      {
 7          int d = 4;
 8          #pragma omp task
 9          {
10              int e = 5;
11
12              // Scope of a: shared
13              // Scope of b: firstprivate
14              // Scope of c:
15              // Scope of d:
16              // Scope of e:
17  } } }
```

Data Scoping Example (4/7)

```
 1  int a = 1;
 2  void foo()
 3  {
 4      int b = 2, c = 3;
 5      #pragma omp parallel private(b)
 6      {
 7          int d = 4;
 8          #pragma omp task
 9          {
10              int e = 5;
11
12              // Scope of a: shared
13              // Scope of b: firstprivate
14              // Scope of c: shared
15              // Scope of d:
16              // Scope of e:
17  } } }
```

Data Scoping Example (5/7)

```
 1  int a = 1;
 2  void foo()
 3  {
 4      int b = 2, c = 3;
 5      #pragma omp parallel private(b)
 6      {
 7          int d = 4;
 8          #pragma omp task
 9          {
10              int e = 5;
11
12              // Scope of a: shared
13              // Scope of b: firstprivate
14              // Scope of c: shared
15              // Scope of d: firstprivate
16              // Scope of e:
17  } } }
```

Data Scoping Example (6/7)

HPC.NRW

```
 1  int a = 1;
 2  void foo()
 3  {
 4      int b = 2, c = 3;
 5      #pragma omp parallel private(b)
 6      {
 7          int d = 4;
 8          #pragma omp task
 9          {
10              int e = 5;
11
12              // Scope of a: shared
13              // Scope of b: firstprivate
14              // Scope of c: shared
15              // Scope of d: firstprivate
16              // Scope of e: private
17  } } }
```

Hint: Use default(none) to be forced to think about every variable if you do not see clear.

Data Scoping Example (7/7)

```
 1  int a = 1;
 2  void foo()
 3  {
 4      int b = 2, c = 3;
 5      #pragma omp parallel private(b)
 6      {
 7          int d = 4;
 8          #pragma omp task
 9          {
10              int e = 5;
11
12              // Scope of a: shared,           value of a: 1
13              // Scope of b: firstprivate,     value of b: 0 / undefined
14              // Scope of c: shared,           value of c: 3
15              // Scope of d: firstprivate,     value of d: 4
16              // Scope of e: private,          value of e: 5
17  } } }
```

Scoping – Lifetime (1/2)

HPC.NRW

– How long do private / firstprivate instances exist?

```
int i = 5;
#pragma omp parallel \
firstprivate(i)
{
   // private copy per thread
   // initialized with 5
   // alive until end of
      parallel region
}<
```

```
#pragma omp parallel
#pragma omp single
{
   int i = 5; // alive until end of single
   #pragma omp task
   {
      // firstprivate copy of i for task
      // alive until end of task
   }<
}
```

Scoping – Lifetime (2/2)

HPC.NRW

– How long do private / firstprivate instances exist?

```
int i = 5;
#pragma omp parallel \
firstprivate(i)
{
   // private copy per thread
   // initialized with 5
   // alive until end of
      parallel region
}<
```

```
#pragma omp parallel
#pragma omp single
{
   int i = 5; // alive until end of single
   #pragma omp task
   {
      // firstprivate copy of i for task
      // alive until end of task
   }<
}
```

➢ Alive until end of assigned

`structured block` or `construct`

Orphaned Task Variables

HPC.NRW

– Arguments passed by reference are `firstprivate` by default in orphaned task generating constructs, example:

```
void task_body (int &);
void gen_task (int &x) { //
  #pragma omp task       //
  task_body (x);
}

void test (int &y, int &z) {
  #pragma omp parallel private(y)
  {
    y = z + 2;
    gen_task (y); // no matter if the argument is determined private
    gen_task (z); // or shared in the enclosing context.
    y++;          // each thread has its own int object y refers to
    gen_task (y);
} }
```

- **Question:** What is the scoping of x?
- **General rule:** `firstprivate` if not `shared` before
- **Problem:** Due to call by reference it might or might not be `shared`

- **Solution:** Special OpenMP rule for orphaned task has to be applied.

Example taken from the OpenMP 4.5 Examples

Questions?