Description

Being specified in the `on` or the `from` clause of some directives, the template reference refers to a subset of a node set in which the specified subset of the template resides.

Specifically, the "`*`" symbol that appears as *template-subscript* in a dimension of *template-ref* is interpreted by each node at runtime as the indices of the elements in the dimension that reside in the node. "`*`" in a template reference is similar to "`*`" in a node reference.

Examples

Assume that `t` is a template.

- In the `task` directive, the executing node set of the task can be indirectly specified using a template reference in the `on` clause.

XcalableMP Fortran
```
!$xmp task on t(1:m,1:n)
!$xmp task on t
```

XcalableMP C
```
#pragma xmp task on t[0:n][0:m]
#pragma xmp task on t
```

- In the `loop` directive, the executing node set of each iteration of the following loop is indirectly specified using a template reference in the `on` clause.

XcalableMP Fortran
```
!$xmp loop (i) on t(i-1)
```

XcalableMP C
```
#pragma xmp loop (i) on t[i-1]
```

- In the `array` directive, the executing node set on which the associated array-assignment statement is performed in parallel is indirectly specified using a template reference in the `on` clause.

XcalableMP Fortran
```
!$xmp array on t(1:n)
```

XcalableMP C
```
#pragma xmp array on t[0:n]
```

- In the `barrier`, `reduction`, and `bcast` directives, the node set that is to perform the operation collectively can be indirectly specified using a template reference in the `on` clause.

XcalableMP Fortran
```
!$xmp barrier on t(1:n)
!$xmp reduction (+:a) on t(*,:)
!$xmp bcast (b) on t(1:n)
```

XcalableMP C
```
#pragma xmp barrier on t[0:n]
#pragma xmp reduction (+:a) on t[:][*]
#pragma xmp bcast (b) on t[0:n]
```

4.3.3 `distribute` Directive

Synopsis

The `distribute` directive specifies the distribution of a template.

Syntax

[F] `!$xmp distribute` *template-name* `(`*dist-format [, dist-format]...* `)` `onto` *nodes-name*

[C] `#pragma xmp distribute` *template-name* `(`*dist-format [, dist-format]...* `)` `onto` *nodes-name*

[C] `#pragma xmp distribute` *template-name* `[` *dist-format* `]` *[* `[` *dist-format* `]` *...]* `onto` *nodes-name*

where *dist-format* must be one of:

```
*
block [ ( int-expr ) ]
cyclic [ ( int-expr ) ]
gblock ( { * | int-array } )
```

Description

According to the specified distribution format, a template is distributed onto a specified node array. The dimension of the node array that appears in the `onto` clause corresponds, in order of left-to-right, to the dimension of the distributed template for which the corresponding *dist-format* is not "`*`".

Let `d` be the size of the dimension of the template, `p` be the size of the corresponding dimension of the node array, `ceiling` and `mod` be Fortran's intrinsic functions, and each of the arithmetic operators be that of Fortran. The interpretation of *dist-format* is as follows:

"`*`" The dimension is not distributed.

`block` Equivalent to `block(ceiling(d/p))`.

`block(n)` The dimension of the template is divided into contiguous blocks of size `n`, which are distributed onto the corresponding dimension of the node array. The dimension of the template is divided into `d/n` blocks of size `n`, and one block of size `mod(d,n)` if any, and each block is assigned sequentially to an index along the corresponding dimension of the node array. Note that if `k = p-d/n-1 > 0`, then there is no block assigned to the last `k` indices.

`cyclic` Equivalent to `cyclic(1)`.

`cyclic(n)` The dimension of the template is divided into contiguous blocks of size `n`, and these blocks are distributed onto the corresponding dimension of the node array in a round-robin manner.

`gblock(m)` `m` is referred to as a mapping array. The dimension of the template is divided into contiguous blocks so that the i'th block is of size `m(i)`, and these blocks are distributed onto the corresponding dimension of the node array.

If at least one `gblock(*)` is specified in *dist-format*, then the template is initially undefined and must not be referenced until the shape of the template is defined by `template_fix` directives at runtime.

Restrictions

- `[C]` *template-name* must be declared by a `template` directive that lexically precedes the directive.
- The number of *dist-format* that is not "`*`" must be equal to the rank of the node array specified by *nodes-name*.
- The size of the dimension of the template specified by *template-name* that is distributed by `block(n)` must be equal to or less than the product of the block size `n` and the size of the corresponding dimension of the node array specified by *nodes-name*.

- The array *int-array* in parentheses following `gblock` must be an integer one-dimensional array, and its size must be equal to the size of the corresponding dimension of the node array specified by *nodes-name*.
- Every element of the array *int-array* in parentheses following `gblock` must have a value of a nonnegative integer.
- The sum of the elements of the array *int-array* in parentheses following `gblock` must be equal to the size of the corresponding dimension of the template specified by *template-name*.
- `[C]` A `distribute` directive for a template must precede any of its references in the executable code in the block.
- A template can be distributed only once by a `distribute` directive.
- A template that is not distributed can not be referenced.

Examples

Example 1

XcalableMP Fortran

```
!$xmp nodes p(4)
!$xmp template t(64)
!$xmp distribute t(block) onto p
```

XcalableMP C

```
#pragma xmp nodes p[4]
#pragma xmp template t[64]
#pragma xmp distribute t[block] onto p
```

The template `t` is distributed in `block` format, as shown in the following table.

p(1)	t(1:16)
p(2)	t(17:32)
p(3)	t(33:48)
p(4)	t(49:64)

p[0]	t[0:16]
p[1]	t[16:16]
p[2]	t[32:16]
p[3]	t[48:16]

Example 2

XcalableMP Fortran

```
!$xmp nodes p(4)
!$xmp template t(64)
!$xmp distribute t(cyclic(8)) onto p
```

XcalableMP C

```
#pragma xmp nodes p[4]
#pragma xmp template t[64]
#pragma xmp distribute t[cyclic(8)] onto p
```

The template `t` is distributed in `cyclic` format of size eight, as shown in the following table.

p(1)	t(1:8) t(33:40)
p(2)	t(9,16) t(41:48)
p(3)	t(17,24) t(49:56)
p(4)	t(25,32) t(57:64)

p[0]	t[0:8] t[32:8]
p[1]	t[8:8] t[40:8]
p[2]	t[16:8] t[48:8]
p[3]	t[24:8] t[56:8]

Example 3

XcalableMP Fortran

```
!$xmp nodes p(8,5)
!$xmp template t(64,64,64)
!$xmp distribute t(*,cyclic,block) onto p
```

XcalableMP C

```
#pragma xmp nodes p[5][8]
#pragma xmp template t[64][64][64]
#pragma xmp distribute t[block][cyclic][*] onto p
```

The first dimension of the template `t` is not distributed. The second dimension is distributed onto the first dimension of the node array `p` in `cyclic` format. The third dimension is distributed onto the second dimension of `p` in `block` format. The results are as follows: