Events

Events are useful if one or more images need to do something before another image can continue.

For example, in the multifrontal method for factorizing a sparse matrix, work at a node of the assembly tree has to wait for all the work at its child nodes to be completed.

Event variable

An event variable is a scalar coarray of type **`event_type`**. It contains a count which increases by 1 each time the event is “posted”.

```
use iso_fortran_env
type(event_type), save :: event[*]
  :
event post(event[i])
  :
event wait(event) ! Wait until count >= 1
       ! then decreases it by 1 and continue
```

Wait for count

Can wait until count reaches given value:

```
event wait(event,until_count=value)
   ! Waits until count >= value and
   ! then decreases it by value
```

Useful if a number of images need to perform their actions before the executing image can continue.

Query count value

`call event_query (event, count)`

Sets **`count`** to the count value of **`event`**.

It is an intrinsic subroutine that is atomic.

Direct-access parallel i/o

A direct-access file has records all of the same length and each can be accessed directly. We used to have the possibility of opening such a file on a team of images.

If an image writes a record, no other image can access it unless

- the writing image executes a **`flush`** statement for the file and
- the access is in a segment that follows the segment containing the **`flush`** statement.

Failed images

The probability of a particular image failing is small, but if the number of images is huge, the probability that one or more fails is significant.

Hence the concept of continuing execution in the presence of failed images.

If an image is considered to be failed, it remains so for the rest of the program execution.

`failed_images` intrinsic function

`failed_images()`

Returns an integer array holding image indices of failed images in the current team.

`failed_images(team)`

Returns an integer array holding image indices of failed images in **`team`**.

Testing for failed image

```
use :: iso_fortran_env
   :
change team (team_a)
        :
    sync_all(stat=st)
    if (st==stat_failed_image) exit
end team
sync_all(stat=st)
if (st==stat_failed_image) then
    : Deal with failure
end if
```

`fail_image` statement

The statement

`fail_image`

causes an image to behave as failed.

It may be needed for debugging.

Knock-on failures

If the statement

```
a = b[image]
```

is executed and **`image`** has failed, the executing image will stall and will eventually be regarded as failed.

Should there be a way to get this image back into use?

One possibility is that it should exit the change team construct.

Do we need another state? Stalled?

More intrinsic atomic subroutines

The intrinsic subroutines

`atomic_add`

`atomic_and`

`atomic_cas`

`atomic_or`

`atomic_xor`

have been added.

Summary of features in TS

- Teams
- Collective intrinsic subroutines
- Events
- Failed images
- More atomics

Reference

The draft TS that was current at the time of this talk is visible here

ftp://ftp.nag.co.uk/sc22wg5/N2001-N2050/N2007.pdf

Acknowledgement

I would like to thank Dinshaw Balsara for providing these video facilities and for his patience while I was recording these lectures.

The sides and the codes associated with these lectures are available on the web site :

http://www.nd.edu/~dbalsara/Numerical-PDE-Course