Fault tolerant Master-Worker type applications in MPI

Edgar Gabriel, Terry Dontje, Josh Hursey, Rich Graham

Assumptions

- Master job consists of *n* processes, $n \geq 1$
- Master job spawns *k* worker jobs, each job having *p(i)* processes, $p(i) \geq 1$, $i=1,\ldots,k$
- Both master job has set all error handlers to `MPI_ERRORS_RETURN` or a user defined error handler which does not abort all processes
- Within an MPI_COMM_WORLD, the error behavior is identical to the default error behavior of the MPI specification
  - i.e. no guarantee that the application can continue execution and communication between processes in the same MPI_COMM_WORLD

Goals

- Define semantics for the master job to recognize and handle a failed worker job/process
  - Not dealing with failed master job.
- Define operations allowed on a object handle that raised the error

Requirements (I)

- Need to recognize a failed worker job through a unique error code (e.g. `MPI_ERR_xxx`)
  - Error code has to clearly indicate a failed process in the worker job, and has to be distinguishable from a process failure in the master job itself, in order to take the appropriate actions and from other, non-fatal errors
  - Error code can also be raised on an
    - intra-communicator (e.g. `MPI_Intercomm_merge`)
    - An `MPI_Win`
    - An `MPI_File`

Requirements (II)

- Operations allowed on the MPI handle where the error has been raised
  - Communication: `MPI_Comm_free`

    (Note: `MPI_Comm_disconnect` is not allowed due to its collective syntax).
  - (Window: `MPI_Win_free` -> is collective)
  - (File: `MPI_File_close` -> is collective)

Requirements (III)

- An error code shall only be returned by MPI functions using a communicator/window/file which involves processes from the failed worker job
  - Required to identify the handles which shall be freed
  - Required to continue communication between master job and worker jobs which have not failed.

Requirements (IV)

- In case the child processes disconnected from the parent using `MPI_Comm_disconnect` and would like to reconnect using `MPI_Comm_accept/connect` : in case of `MPI_Comm_connect/accept` returns an error, need to distinguish whether the remote job has failed or doesn't have a matching call (yet) posted, i.e.
  - Introduce separate error codes
    - `MPI_ERR_PORT`: job is alive but does not have a matching accept/connect call posted (already existing)
    - `MPI_ERR_xxx`: status of worker job not known/failed
  - Introduce a reserved info key `timeout` in Connect/Accept (chapter 10.3.4 in MPI 2.1)

Requirements (V)

- 'Stronger' definition of the behavior of `MPI_Abort` for connected processes in MPI-2.1, chapter 10.5.4 might be required:

  ```
  "As in MPI-1, it [MPI_Abort] may abort all
    processes in MPI_COMM_WORLD (ignoring its
    comm argument). Additionally, it may abort
    connected processes as well, although it
    makes best attempt to abort only the
    processes in comm."
  ```

  e.g. to

  ```
  "As in MPI-1, it [MPI_Abort] may abort all
    processes in MPI_COMM_WORLD (ignoring its
    comm argument). MPI_Abort should however…"
  ```

Remarks

- Restricting the size of worker jobs to only 1 process:
  - Requirements (I-IV) are still valid.
  - Requirement (V) might be less relevant.
- Restricting additionally the size of the master job to 1 process:
  - Requirement (I) could be simplified.
- Restricting additionally the set of functions allowed to be used, e.g.
    - No disconnect/reconnect: remove Requirement (IV)
    - No one-sided + file operations: significant simplification of Requirement (II)

PARALLEL SOFTWARE TECHNOLOGIES LABORATORY PSTL

Questions

- Do we need to define a restricted set of functions allowed for Master-Worker type applications, if they want to take advantage of a simple FT scheme?
- Single chapter vs. distributing the required statements in the according sections?

- Side note:
  - For efficient execution of large-scale master-worker style applications, we need to speed up process spawning, e.g. asynchronous dynamic process management functions (`MPI_Comm_ispawn`) (definitely MPI-3)

CS@UH