Sessions & Reinit

Ignacio Laguna

Meeting 02/05/24

Reinit Model

- We removed the automatic long-jump from Reinit
  - Too many issues; it could lead to memory leaks
  - In the new model, the user will control jumping to a rollback location

- Semantics of MPI_Test_failure()
  - Automatically recovers MPI_COMM_WORLD
  - Cleans up state of MPI (communicators are deleted, queues freed, etc.)
  - If a failure occurred, it returns a code to indicate that MPI was cleaned up, e.g., MPI_RESTARTED
  - Maybe need to be collective? (not sure, but probably yes)

Two possible ways for Reinit + Sessions

1. Very restrictive:
   - Session handle is valid after a failure, but we can’t not do anything with it, except finalizing the session.
   - Local operations are not allowed.
   - Objects derived from the session are invalid.

2. Less restrictive:
   - Session handle is valid after a failure
   - Everything that is local is fine.

Session handle is valid after a failure, but we can't not do anything with it, except finalizing the session. Local operations are not allowed. Objects derived from the session are invalid.

```
restart:

MPI_Session_init(..., &shandle);
// compute...

while (!done) {
  // Load checkpoint if needed
  // compute...

  err = MPI_Test_failure();
  if (err == MPI_RESTARTED) {
    MPI_Session_finalize(&shandle);
    goto restart;
  }
}

MPI_Session_finalize(&shandle);
```

Session handle is valid after a failure; everything that is local is fine.

```
restart:
MPI_Session_init(..., &shandle);
// compute...

while (!done) {
  // Load checkpoint if needed
  // compute...

  err = MPI_Test_failure();
  if (err == MPI_RESTARTED) {
    // Since this is local, it's valid here
    MPI_Group_from_session_pset(shandle,...);
    ...
    MPI_Session_finalize(&shandle);
    goto restart;
  }
}

MPI_Session_finalize(&shandle);
```

- Not sure if there is a use case for this.
- For Reinit, at the end everything is restarted.

Questions

- Do sessions allow comparing sets?
  - Is the mpi://WORLD set comparable after a failure?

- Can the user derive **any** kind of object from a session, e.g., communicators, windows, etc.?
  - Are there objects that **cannot be** derived from a sessions?

Meeting 02/26/24

Discussion – 02/26/24

- Semantics of MPI_Test_failure()
  - Not synchronizing (processes may return even if others didn't call it yet)
  - Collective (all processes in World must call it)
    - All alive processes
  - When does it stop returning **`MPI_RESTARTED`**?
    - If infinite failures occur, it may return MPI_RSTARTED an infinite number of times
    - If a period of time T, one failure occurred, it will return MPI_RESTARTED once.

- Semantics of MPI_Init()
  - Should MPI_Init be non synchronizing after a failure?
  - New process need to bypass code to get to "restart" label

ReInit and the World model

```
int main() {
  MPI_Init();
restart:
  make_subcomms_files_windows(MPI_COMM_WORLD, &my_comm);
  load_checkpoint(my_comm);
  while (!done) {
    err = MPI_Test_failure();
    if (MPI_ERR_RESTARTED == err) {
      { // (good) design choice – no user actions needed
        // derived MPI objects no longer useful without ULFM
        // MPI_COMM_WORLD can be different
      }
      goto restart;
    } else if (++steps % freq) {
      save_checkpoint(my_comm);
    }
    // compute with message passing and RMA
    done = compute_a_step(my_comm);
  }
  MPI_Finalize();
}
```

ReInit and the Sessions model

```
int main() {
  MPI_Session_init(&session);
restart:
  make_subcomms_files_windows(session, &my_comm);
  load_checkpoint(my_comm);
  while (!done) {
    err = MPI_Test_failure();
    if (MPI_ERR_RESTARTED == err) {
      if (MPI_SESSION_VALID_AFTER_RESTART) { // (poor) design choice
        MPI_Session_finalize(&session); // no longer useful without ULFM
        MPI_Session_init(&session); // even mpi://world can be different
      }
      goto restart;
    } else if (++steps % freq) {
      save_checkpoint(my_comm);
    }
    // compute with message passing and RMA
    done = compute_a_step(my_comm);
  }
  MPI_Session_finalize(&session);
}
```

Dan's example – 02/26/24

```
void make_subcomms_files_windows(session, &my_comm) {
  MPI_Group world_group;
  MPI_Group_from_session_pset(session, "mpi://world", &world_group); // local
  MPI_Comm_from_group(world_group, &my_comm); // collective over my_comm
  MPI_Group_free(world_group);
}

int main() {
// Reinit & sessions, option (2)
restart:
  MPI_Session_init(&session); // local
  make_subcomms_files_windows(session, &my_comm); // collective
  load_checkpoint(my_comm);
  while (!done) {
    err = MPI_Test_failure();
    if (MPI_ERR_RESTARTED == err) {
      goto restart;
    } else if (++steps % freq) {
      save_checkpoint(my_comm);
    }
    // compute with message passing and RMA
    done = compute_a_step(my_comm);
  }
  MPI_Session_finalize(&session);
}
```

Aurelien's example (1) – 02/26/24

````
## Variant with explicit resync/reinit

```c
int main() {

static bool restart = false;
MPI_Session_init(&session); //local – with this model, open sessions may be able to remain
open across faults? (Discuss implications wrt. Universe and connectivity)
Restart:
if(restart) {
#if 1callform
  MPI_Reinit(use_wpm); // collective, fixup MPI_COMM_WORLD etc. if use_wpm is true,
otherwise only synchronizes the content of mpi://world and other sets
#else
  MPI_Sessions_fence(); // collective, synchronizes the content of mpi://world and other
sets for all sessions
  if(use_wpm) MPI_Reinit(); // collective, mixup MPI_COMM_WORLD etc. no effect on sessions
  restart = false;
}
create_comms(session); //collective
load_checkpoints(comms); //user-dependent if collective
while(!done) {
  MPI_Irecv();
  MPI_Isend();
  MPI_Allreduce(); // what is rc when an error has already occurred here?

  rc = MPI_Test_failure() // implicit variant for the wpm needed?
  If (MPI_ERR_REINIT == rc) {
    Restart = true;
    goto restart;
  …
}
```
````

Aurelien's example (2) – 02/26/24

Variant with implicit resync/reinit using a predefined errhandler

```c
int main() {

MPI_set_errhandler(MPI_ERRORS_REINIT); // this API is new – this should be global state
// if any session sets it, it's implicit in other sessions?

MPI_Init();
MPI_Test_failure(); // this synch with the test_failure below

Restart:
MPI_Session_init(&session); //local
~~MPI_Session_set_errhandler(session, MPI_ERRORS_REINIT); //problem: this must be global~~
create_comms(sessions); // collective
Load_checkpoints(comms); // user-dependent if collective

while(!done) {
  rc = MPI_SUCCESS;
  rc |= MPI_Irecv(); // |= not correct but you get the idea
  rc |= MPI_Send();
  rc |= MPI_Allreduce();
  // add reinit check
  rc = MPI_Test_failure(); // scope is global
  if(rc == MPI_ERR_REINIT) goto restart;
}
```