Memory Allocator Kind Info

Jim Dinan

September 28, 2022
HACC Working Group

Goals

Users currently rely on nonstandard methods for enabling and querying MPI library support for accelerator memory (see issue for links)

This proposal uses info to provide users with a portable solution to:

1. Detect whether accelerator memory is supported by the MPI library
2. Request support for accelerator memory from the MPI library (when using Sessions)
3. Constrain usage of accelerator memory to specific communicators, windows, etc.

PR: https://github.com/mpi-forum/mpi-standard/pull/714

Issue: https://github.com/mpi-forum/mpi-issues/issues/580

Proposed Info Keys

mpi_memory_alloc_kind

- Memory kinds supported by the MPI library

mpi_assert_memory_alloc_kind

- Assert memory kinds used by the application

Query Support for Accelerator Memory (World Model)

```
1.  bool cuda_aware = false;
2.  int len = MPI_MAX_INFO_VAL, flag = 0;
3.  char *val = malloc(MPI_MAX_INFO_VAL);

4.  MPI_Info_get_string(MPI_INFO_ENV, “mpi_memory_alloc_kind”, &len, val, &flag);

5.  while (flag && (kind = strsep(&val, ",")) != NULL) {
      if (strcasecmp(kind, "cuda") == 0 || strcasecmp(kind, "cuda:device") == 0) {
        cuda_aware = true;
        break;
      }
    }
```

Query Support for Accelerator Memory (Sessions Model)

```
1.  bool cuda_aware = false;
2.  int len = MPI_MAX_INFO_VAL, flag = 0;
3.  char *val = malloc(MPI_MAX_INFO_VAL);
4.  MPI_Info info;

5.  MPI_Session_get_info(session, &info);
6.  MPI_Info_get_string(info, “mpi_memory_alloc_kind”, &len, val, &flag);

7.  while (flag && (kind = strsep(&val, ",")) != NULL) {
      if (strcasecmp(kind, "cuda") == 0 || strcasecmp(kind, "cuda:device") == 0) {
        cuda_aware = true;
        break;
      }
    }
```

Request Support for CUDA Memory (Sessions Model)

```
1.  bool cuda_aware = false;
2.  int len = MPI_MAX_INFO_VAL, flag = 0;
3.  char *val = malloc(MPI_MAX_INFO_VAL);
4.  MPI_Info info;

5.  MPI_Info_create(&info);
6.  MPI_Info_set(info, “mpi_assert_memory_alloc_kind”, “cuda:device”);
7.  MPI_Session_init(info, MPI_ERRORS_ARE_FATAL, &session);
8.  MPI_Info_free(&info);

9.  MPI_Session_get_info(session, &info);
10. MPI_Info_get_string(info, “mpi_assert_memory_alloc_kind”, &len, val, &flag);

11. if (flag && strcasecmp(val, "cuda:device") == 0) {
        cuda_aware = true;
    }
```

Request Support for CUDA Memory II (Sessions Model)

```
 1.   bool cuda_aware = false;
 2.   int len = MPI_MAX_INFO_VAL, flag = 0;
 3.   char *val = malloc(MPI_MAX_INFO_VAL);
 4.   MPI_Info info;

 5.   MPI_Info_create(&info);
 6.   MPI_Info_set(info, “mpi_assert_memory_alloc_kind”, “cuda:device”);
 7.   MPI_Session_init(info, MPI_ERRORS_ARE_FATAL, &session);
 8.   MPI_Info_free(&info);

 9.   MPI_Session_get_info(session, &info);
10.   MPI_Info_get_string(info, “mpi_memory_alloc_kind”, &len, val, &flag);

11.   // Check mpi_memory_alloc_kind instead of mpi_assert_memory_alloc_kind
      while (flag && (kind = strsep(&val, ",")) != NULL) {
        if (strcasecmp(kind, "cuda") == 0 || strcasecmp(kind, "cuda:device") == 0) {
          cuda_aware = true;
          break;
        }
      }
```

Request Support for SYCL Memory (Sessions Model)

```
 1.  bool sycl_aware = false;
 2.  int len = MPI_MAX_INFO_VAL, flag = 0;
 3.  char *val = malloc(MPI_MAX_INFO_VAL);
 4.  MPI_Info info;

 5.  MPI_Info_create(&info);
 6.  MPI_Info_set(info, “mpi_assert_memory_alloc_kind”, “sycl”);
 7.  MPI_Session_init(info, MPI_ERRORS_ARE_FATAL, &session);
 8.  MPI_Info_free(&info);

 9.  MPI_Session_get_info(session, &info);
10.  MPI_Info_get_string(info, “mpi_assert_memory_alloc_kind”, &len, val, &flag);

11.  if (flag && strcasecmp(val, "sycl") == 0) {
       sycl_aware = true;
     }
12.  …
13.  buffer buf(buf_size); // Application does not know if allocated via Level 0, HIP, CUDA, etc…
14.  MPI_Recv(buf, …);
```

Request Support for OpenMP Memory (Sessions Model)

```
 1.  bool openmp_aware = false;
 2.  int len = MPI_MAX_INFO_VAL, flag = 0;
 3.  char *val = malloc(MPI_MAX_INFO_VAL);
 4.  MPI_Info info;

 5.  MPI_Info_create(&info);
 6.  MPI_Info_set(info, “mpi_assert_memory_alloc_kind”, “openmp”);
 7.  MPI_Session_init(info, MPI_ERRORS_ARE_FATAL, &session);
 8.  MPI_Info_free(&info);

 9.  MPI_Session_get_info(session, &info);
10.  MPI_Info_get_string(info, “mpi_assert_memory_alloc_kind”, &len, val, &flag);

11.  if (flag && strcasecmp(val, "openmp") == 0) {
       openmp_aware = true;
     }
12.  …
13.  void *buf = omp_target_alloc(size, 0); // Application unaware if allocated via Level 0, HIP, CUDA, etc…
14.  MPI_Recv(buf, …);
```

Restrict Memory Allocator Kind on a Communicator

```
// Assuming the user already confirmed that MPI is CUDA-aware, they don’t
// need to check whether the assertion was recognized by the MPI library
1.  MPI_Comm newcomm;
2.  MPI_Info info;

3.  MPI_Info_create(&info);
4.  MPI_Info_set(info, “mpi_assert_memory_alloc_kind”, “cuda:device”);

5.  // MPI library can assume all buffers on newcomm are of kind cuda:device
    MPI_Comm_dup_with_info(MPI_COMM_WORLD, info, &newcomm);

6.  MPI_Info_free(&info);
```

Discussion Items

Edgar - Clarify that info assertions apply to the local process

Maria/Dan - Using an assert to request support for a memory allocation kind is different from the existing usage model for info assertions

- The set of things being restricted is not known at the time of the assertion
- User is still asserting application behavior, but must check if accepted
- Could add a different name info for requesting support on a session

Maria - What about applications that support two models (e.g. CUDA and HIP) and want to select what to use at runtime?

- We could add MPI_Session_set_info so users can check supported memory allocation kinds before making an assertion. Given that libraries don’t enable accelerator support by default, this usage model may need more knobs.
- We could allow an assertion to be partially accepted (even more unusual usage for asserts)

Dan - You can request support for a given memory allocator kind in the world model by setting the assertion when creating a communicator, window, etc.

Discussion Items II

Christoph - Can we use this in MPI_Alloc_mem

- Intentionally avoiding any changes to how MPI allocates memory in MPI 4.1

Rolf - Do we need MPI_INFO_ENV and Sessions level usage or can we have only info assertions on communicators, windows, files, etc.

- Users need to check whether request was accepted

Do we want to put any of these examples into the standard?

- If examples use vendor-specific names, they should go into the side document

Suggest moving “Memory Allocator Info” section to “Common Elements of Both Process Models”

Add info keys / values to A.1.5, ensure that one underlined entry in constants and predefined index

Use infoval LaTeX macro for memory allocation kinds. Put brackets around node and policy.

Request Support for Either CUDA or HIP Memory

```
1.  bool cuda_aware = false, hip_aware = false;
2.  int len = MPI_MAX_INFO_VAL, flag = 0;
3.  char *val = malloc(MPI_MAX_INFO_VAL);
4.  MPI_Info info_in, info_out;

5.  MPI_Session_init(MPI_INFO_NULL, MPI_ERRORS_ARE_FATAL, &session);
6.  MPI_Info_create(&info_in);

7.  // (1) Check for CUDA support
    MPI_Info_set(info_in, “mpi_assert_memory_alloc_kind”, “cuda:device”);
8.  MPI_Session_set_info(session, info_in); // New function!

9.  MPI_Session_get_info(session, &info_out);
10. MPI_Info_get_string(info, “mpi_assert_memory_alloc_kind”, &len, val, &flag);

11. if (flag && strcasecmp(val, "cuda:device") == 0) {
        cuda_aware = true;
    }
    MPI_Info_free(info_out);

12. // (2) Check for HIP support
    if (!cuda_aware) {
        MPI_Info_set(info_in, “mpi_assert_memory_alloc_kind”, “hip:device”);
        MPI_Session_set_info(session, info_in); // New function!
        MPI_Session_get_info(session, &info_out);
        if (flag && strcasecmp(val, "hip:device") == 0) {
            hip_aware = true;
        }
        MPI_Info_free(info_out);
    }
```