IT Center | RWTH AACHEN UNIVERSITY

Debugging opaque MPI handles

New attempt towards a spec

Joachim Protze

Opaque MPI handles?

```
int MPI_Send( const void *buf,
              int count,
              MPI_Datatype datatype,
              int dest,
              int tag,
              MPI_Comm comm );
```

- **MPI_Datatype: MPI_INT, …, or user defined derived data type**
- **MPI_Comm: MPI_COMM_WORLD, or user defined communicator to select a subset of processes**
- **Integer value in Fortran, implementation specific in C**
  - → Typically int (e.g. sgimpt) or pointer (e.g. OpenMPI)

Opaque MPI handles (cont.)

- **All the handles:**
  - → MPI_Comm, MPI_Group
  - → MPI_Datatype, MPI_Op
  - → MPI_File, MPI_Win, MPI_Info, MPI_Errhandler, MPI_Keyval
  - → Special: MPI_Request, MPI_Status
- **Most have constructors and a destructor function:**
  - → E.g. MPI_Comm_dup, MPI_*_free
- **MPI_Request special:**
  - → „Constructor“ is async communication
- **MPI_Status special:**
  - → Data structure is application owned, data layout is MPI implementation specific

Context from our side

- **Experience from working on debugging interface for OpenMP**
  - → Implemented an OpenMP debugging DLL prototype
- **Datatype „pretty-printer“ for gdb (next slide)**
- **Master student to write his master thesis on this topic**
  - → 6 month worth of work to implement a prototype, document the work in form of a spec and write-up the thesis
  - → (btw: where is EuroMPI‘17? ;)

Our previous work: gdb plugin

```
66        MPI_Bcast (buffer, 1, struDT, 0, MPI_COMM_WORLD);
(gdb) p struDT
$2 = (MPI_Datatype) 0xbbdf20
(gdb) source mpipp/__init__.py
MPI-PP GDB Support loaded
(gdb) p struDT
$3 = { kind = STRUCT,
       count = 5,
       entries = { {blocklength = 1, offset = -4, type = "MPI_LB"},
                   {blocklength = 5, offset = 0, type = "MPI_INT"}, …}
(gdb) p buffer
$4 = {{"MPI_LB"}, {"(MPI_INT)(0x1b164) 1024", "(MPI_INT)(0x1b168)
2048", "(MPI_INT)(0x1b16c) 3072", …}, … {"MPI_UB"}}
```

J. Protze, T. Hilbrich, A. Knüpfer, B. R. de Supinski and M. S. Müller, "**Holistic Debugging of MPI Derived Datatypes**", Parallel & Distributed Processing Symposium (**IPDPS**), **2012** IEEE 26th International, Shanghai, 2012, pp. 354-365. doi: 10.1109/IPDPS.2012.41

The master plan

- **The basic assumption:**
  - → Knowledge that can be collected by an observing PMPI wrapper can serve as the common base line for knowledge available in an MPI runtime.

- **Basing the prototype on a PMPI wrapper**
  - → The wrapper collects the information during execution, the debugging library reads the information from the wrapper.
  - → The interface would provide information as put into MPI opaque handle „constructor“ calls

- **Later, a runtime could implement an own debugging DLL and provide the information using implementation specific internal data structures.**

Available work on MPI debugging

- **Jeff‘s mpihandles_interface.h**
  - → For the thesis, I would start from scratch and compare the outcome.

- **Msgq interface**
  - → Should we stick with this callback interface spec?
  - → @JDS: the interface in OMPD is more lightweight – adopt this?

Use cases

What can a debugger do with the information from the interface?

- **Provide information on MPI handles in the code during debugging**
  - → As in slide 3
- **Provide information on the data to be transmitted based on buffer address and type information**
- **Identify the group of processes specified by a comm-handle**