Argonne NATIONAL LABORATORY

The "MPI_T" Tools Interface in MPI-3 and MPICH2

Dave Goodell
goodell@mcs.anl.gov
Mathematics & Computer Science Division
Argonne National Laboratory

U.S. DEPARTMENT OF ENERGY

Questions for You

- Who here maintains a tool for MPI programs?
- Who knows MPI-3 is in development?
- Who has heard of MPI_T before now?

Motivation

- Tons of info trapped inside MPI libraries
- Inaccessible through any standard interface (except PMPI)
- Potentially brand new tool needed for each MPI implementation
- PERUSE deemed too rigid, not widely implemented

MPI_T: a new interface for tools

- MPI_T will allow standardized API access to:
  - "control variables"
  - "performance variables"
- *How*, not *what*
- Supports multiple simultaneous tools
- MPI-3 is slated for ratification this September
  - Will include MPI_T
  - Chapter draft at: http://bit.ly/M4BvNu

Control Variables

- Similar to environment variables (or Open MPI MCA params)
- MPI only specifies API access
- Read-only / Read-write both supported
- Global scope, no per-object binding
- Each var has:
  - An integer index identifier
  - A name
  - A datatype
  - A description
  - Some other properties…

Control Variable Examples

- MPI collective algorithm selection
- Eager limit thresholds
- Selectable threading strategies
- Network transport selection

Performance Variables

- Think PAPI (sort of) for MPI information
- Several types:
  - Discrete states
  - Counters
  - High/low water marks
  - Timings
  - Percentages
  - A few others…
- Optionally reset-able
- May be bound to specific objects (communicators, datatypes, etc)

Performance Variable Examples

- Current unexpected queue (UQ) length
- Total match attempts in posted queue (PQ) since last var reset

Variable Categorization

- Variables can be grouped into categories
- Categories may contain categories
- Forms a tree of categories
- Talk to me later if you want more info

API Look & Feel

- C bindings only
- Uses small subset of predefined MPI datatypes:
  - MPI_INT
  - MPI_UNSIGNED
  - MPI_UNSIGNED_LONG
  - MPI_UNSIGNED_LONG_LONG
  - MPI_COUNT
  - MPI_CHAR
  - MPI_DOUBLE
- Usable before MPI_Init and after MPI_Finalize
  - Separate refcounted MPI_T_Init_thread/MPI_T_Finalize routines

API Look & Feel

- Variables referenced by integer index
- Dense index space
- Strictly grows
- Control var indices separate from perf var indices
- MPI_T_Cvar_get_num tells how many “right now”
- Use a _get_info call to ask about a specific var index
- “PMPI_T_” for meta-tools ☺

Code example: changing MPI_Allreduce algorithm thresholds

```
MPIX_T_init_thread(MPI_THREAD_SINGLE, &provided);
MPI_Init(&argc, &argv);
MPIX_T_cvar_get_num(&num);
for (i = 0; i < num; ++i) {
  name_len = desc_len = STR_SZ;
  MPIX_T_cvar_get_info(i, name, &name_len, &verb, &dtype, &enumtype, desc,
                       &desc_len, &bind, &scope);
  if (0 == strncmp(name, "ALLREDUCE_SHORT_MSG_SIZE", STR_SZ)) {
    MPIX_T_cvar_handle_alloc(i, NULL, &handle, &count);
    assert(dtype == MPI_INT && count == 1);
    MPIX_T_cvar_read(handle, &val);
    val *= 2;
    MPIX_T_cvar_write(handle, &val);
    MPIX_T_cvar_handle_free(&handle);
    break;
  }
}
if (i == num) { printf("ERROR: could not find short msg param\n"); return 1; }
/* … now do allreduce … */
MPI_Finalize();
MPIX_T_finalize();
return 0;
```

Code example: sampling UQ length

```
MPIX_T_pvar_get_num(&num);
for (i = 0; i < num; ++i) {
  name_len = desc_len = STR_SZ;
  MPIX_T_pvar_get_info(i, name, &name_len, &verb, &varclass, &dtype,
                       &enumtype, desc, &desc_len, &bind, &readonly,
                       &continuous, &atomic);
  if (0 == strcmp(name, "unexpected_recvq_length"))
    uq_idx = i;
}

MPIX_T_pvar_session_create(&session);
MPIX_T_pvar_handle_alloc(session, uq_idx, NULL, &uq_handle, &count);
assert(count = 1);

MPI_Isend(buf, 1, MPI_INT, 0, 0, MPI_COMM_WORLD, &rreq);
MPIX_T_pvar_read(session, uq_handle, &unexpected_qlen);
printf("unexpected_qlen=%d\n", unexpected_qlen);

MPIX_T_pvar_handle_free(session, &uq_handle);
MPIX_T_pvar_session_free(&session);
```

MPICH2's MPI_T Support

- All API functions implemented (as "MPIX_T_*") in 1.5b1
- All environment vars available via control variable interfaces
- A few performance vars available, mainly for matching queue info
- Multithreading limitations
- Should hit downstream MPIs in 6-24 months (Intel MPI, IBM MPIs [BG/Q and some others], Cray >=XE, MVAPICH2)
- **Need tool writer suggestions on most useful way to spend time in this area!!!**