LLNL-PRES-422686

Proposal for an MPI Performance Information Interface

MPIT

MPI Tools WG

MPI Forum, Atlanta, GA, January 2010

Central Idea of MPIT

- Enable tools access to internal MPI information
  - Complementary to MPI Profiling Interface (PMPI)
  - Information about MPI configuration
  - Internal performance information
    - Queue management
    - Memory consumption
- An MPI implementation decides what to expose
  - No restrictions on the implementation
  - Flexibility to expose implementation specific details
  - Option to provide production and debug versions
  - Requirement: API to provide additional information in a portable way to convey some semantics

Three Major Components

- Initialization & Finalization
  - Separate from MPI functionality
- Configuration Variables
  - Any mechanism a user/administrator can use to influence the runtime behavior of MPI
  - Often: environment variables
- Performance Probes
  - Information about MPI internals
  - Data collection during run time
  - Read data at specific locations (calipers)
  - “PAPI/CPU performance counters for MPI”

General Remarks

- Separate name space
  - Allow calls before MPI_Init/after MPI_Finalize
  - New prefix: MPIT_*
- Bindings
  - Tools mostly written in C/C++
  - Provide C bindings only
- PMPI interface also applies to MPIT
- Interface designed for tools
  - Advice to users: discourage use in applications
  - Can lead to portability problems

Return Codes

- All MPIT calls return an error/return code
- Success code: MPIT_SUCCESS
- Failure codes defined as constants
  - Separate section in proposed document
  - Returned failures shall not abort MPI
- MPIT not required to check for all failure conditions
  - Wrong input arguments are the responsibility of the user/tool programmer

String Returns

- API requires return of strings in many places
  - Need to maintain safety
  - Allow flexible sizes
- Proposal based on MPI_Get_processor_name
  - Caller passes char buffer and a pointer to a variable to store the size of the string (on return)
  - Constant to define the maximal size of the buffer
  - Zero termination of string

Initialization

- MPIT_Init and MPIT_Finalize
  - Same structure as for MPI
  - Same arguments
- Separate from MPI_Init/MPI_Finalize
  - Allow MPIT calls before MPI_Init/after MPI_Finalize
  - Same/Similar calling rules within MPIT namespace
- MPIT_Initialized & MPIT_Finalized
  - Same semantics as the MPI counterparts

Type System

- MPIT provides a separate type system
  - Values for configuration variables and probes
  - Needs to be active between MPIT_Init & MPIT_Finalize
  - No need to have complex MPI datatypes
- Base types
  - Similar to native MPI types
  - Equivalent types in MPI to enable communication
- Enumeration types
  - Similar to C enum types
  - Equivalent type is MPI_INT
  - Ability to query names & elements

MPIT Base Types

MPIT Type	MPI Equivalent Type
MPIT_LOGICAL	MPI_LOGICAL
MPIT_SHORT	MPI_SHORT
MPIT_INT	MPI_INT
MPIT_LONG	MPI_LONG
MPIT_CHAR	MPI_CHAR
MPIT_FLOAT	MPI_FLOAT
MPIT_DOUBLE	MPI_DOUBLE

Enumaration Types

- MPIT_Enumtype_query
  - Query the name and number of elements of a type
  - IN Datatype
  - OUT Number of elements in type
  - String with the name of the type

- MPIT_Enumitem_query
  - Query the name for an individual item
  - IN Datatype
  - Item number
  - OUT String with the name of the item

Verbosity Levels

- Number of variables and probes can be large
- MPIs should classify them in verbosity classes
  - MPIT_VERBOSITY_VERY_HIGH
  - MPIT_VERBOSITY_HIGH
  - MPIT_VERBOSITY_MEDIUM
  - MPIT_VERBOSITY_LOW
  - MPIT_VERBOSITY_VERY_LOW
- If MPI does not want to provide classes, it is recommended to assign all variables and probes to MPIT_VERBOSITY_VERY_LOW

Configuration Variables

- Access to variables that help configure MPI
  - E.g., eager send limit
  - Often implemented as environment variables

- Functions
  - Query variables
  - Detect changes
  - Read/write access

Scopes for Configuration Vars.

- MPIT_SCOPE_NEVER
  - Read-only variable, cannot be changed
- MPIT_SCOPE_BEFOREINIT
  - Can only be changed before MPI_Init
- MPIT_SCOPE_DELAYED
  - May be changed at any time, effect can be delayed
- MPIT_SCOPE_IMMEDIATE
  - May be changed at any time

Query Operation

- MPIT_Config_query
  - Query “next” configuration variable
  - IN iterator (MPIT_Configquery_iterator)
  - maxverbosity
  - OUT String with the name of the variable
  - Datatype of variable (MPIT_Datatype)
  - Number of elements / count (int)
  - String with variable description
  - Scope of variable (int)
  - Actual verbosity (int)

Query Operation (cont.)

- Initialization of the query
  - Assign iterator to MPIT_CONFIGQUERY_START
- Return codes
  - MPIT_SUCCESS: query OK, item returned
  - MPIT_END: no more items, no item returned
- MPIT_Config_changed
  - Did the list of configuration variables changes since the last time a query has been initiated?
  - OUT flag (int)
  - Returns true if no query has been started

Read/Write Access

- MPIT_Config_get
  - Read a configuration variable
  - IN name of the variable (string)
  - OUT buffer containing the value of the variable
  - (user responsible to provide right size)

- MPIT_Config_set
  - Set a configuration variable (may return error)
  - IN name of the variable (string)
  - buffer containing the value of the variable

Example

```
MPIT_Configquery_iterator iter;
char name[MPIT_MAX_CONFIGVAR_NAME];
char desc[MPIT_MAX_CONFIGVAR_DESC], char val[4096];
int namelen,desclen,count,scope,verbosity,res;

iter=MPIT_CONFIGQUERY_START;
 do {
      res=MPIT_Config_query(&iter,MPIT_VERBOSITY_NORMAL,
                              (char*)name,&namelen,&datatype,&count,
                        (char*)desc,&desclen,&scope,&verbosity);
      if (res==MPIT_SUCCESS {
            if (count>0)   {
                  res=MPIT_Config_get((char*)name,val);
                  if (res==MPI_SUCCESS)
                        printf("\t%s = %s\n",name,val); }
            } }
while (res==MPIT_SUCCESS);

 if (res==MPIT_END)        printf("\nComplete.\n\n");
                 else printf("\nError: %i\n",res);
```

Performance Probes

- Access to MPI performance information
  - Internal performance data
  - Set of performance probes inside of MPI
  - Divided into classes of probes
  - Optional information to group probes
  - Caliper style instrumentation

- Functions
  - Query probes
  - Detect changes
  - Acquire/release handles
  - Start/stop and read/write/reset probe

Probe Classes

- MPIT_PROBECLASS_STATE
  - Set of discrete states that can change any time
- MPIT_PROBECLASS_UTILIZATION
  - Utilization of a resource (0.0 – 1.0)
- MPIT_PROBECLASS_RESOURCE
  - Absolute value of a resource
- MPIT_PROBECLASS_HIGHWATERMARK
  - High watermark value of a resource
- MPIT_PROBECLASS_LOWWATERMARK
  - Low watermark value of a resource
- MPIT_PROBECLASS_COUNTER
  - Counts occurrence of a particular, individual event (+1)
- MPIT_PROBECLASS_AGGREGATE
  - Aggregate value over time (like counters, but +N)
- MPIT_PROBECLASS_TIMER
  - Aggregate times

Group Information

- Each variable is associated with a set of groups
  - Represented as strings
  - Groups are strings and are defined by MPI
- Two options to group
  - Hierarchies
  - Sets
- Syntax for group sets
  - Group = [A–Z, a–z, 0–9, _, ., :, #]$^{+}$
  - Hierarchy = <group>[-<group>]
  - Groupset = <hierarchy>[,<hierarchy]$^{+}$

Group Information (cont.)

- MPIT_Probe_groups
  - Get additional information for a group (optional)
  - IN group (string)
  - OUT long name the group (string)
  - description of the group (string)

Query Operation

- MPIT_Probe_query
  - Query “next” performance probe
  - IN iterator (MPIT_Probequery_iterator)
  - maxverbosity
  - OUT Class of the probe (int)
  - String with the name of the probe
  - Datatype of variable (MPIT_Datatype)
  - Number of elements / count (int)
  - String with variable description
  - Group of the probe (string)
  - Read only probe, cannot be set/reset (int)
  - Continuous, cannot be started/stopped (int)
  - Actual verbosity (int)

Query Operation (cont.)

- Initialization of the query
  - Assign iterator to MPIT_PROBEQUERY_START
- Return codes
  - MPIT_SUCCESS: query OK, item returned
  - MPIT_END: no more items, no item returned
- MPIT_Probe_changed
  - Did the list of performance probes change since the last time a query has been initiated?
  - OUT flag (int)
  - Returns true if no query has been started

Handle Management

- MPIT_Probe_gethandle
  - Get a handle to access a performance probe
  - IN Probe name (string)
  - OUT Handle of probe (MPIT_Probehandle)
- MPIT_Probe_freehandle
  - Free a probe handle
  - IN/OUT Handle of probe (MPIT_Probehandle)
- No guarantee about what happens to probes while the tool has no active handle

Start/Stop Functions

- MPIT_Probe_start
  - Activate a performance probe
  - IN Handle of probe
- MPIT_Probe_stop
  - Deactivate a performance probe
  - IN Handle of probe
- Continuous probes can't be started/stopped
- All other probes are initialized stopped

Access Functions

- MPIT_Probe_read
  - Read a performance probe (must not be started)
  - IN Handle of probe
  - OUT Buffer for probe variable contents
- MPIT_Probe_write
  - Set a performance probe to a given value
  - IN Handle of probe
  - IN Buffer containing variable contents
- MPIT_Probe_reset
  - Reset a probe to its default starting value
  - IN Handle of probe

Availability / Feedback

- Tools WG is proposing a new tools chapter
  - Replaces the MPI Profiling Chapter
  - PMPI would be a subsection of that new chapter
  - MPIT + debugging proposals new additions
- Proposal text is available
  - SVN: MPI-3.0/wg/tools/new-tools-chapter
  - PDF on tools WG Wiki (potentially slightly outdated)
- Continued discussion during tools WG TelCons
  - Monday 1/25, 8am PT, 11am ET, 5pm MEZ
  - Other topics: debugging APIs