Use Case Evaluation of the Proposed MPIT Configuration and Performance Interface

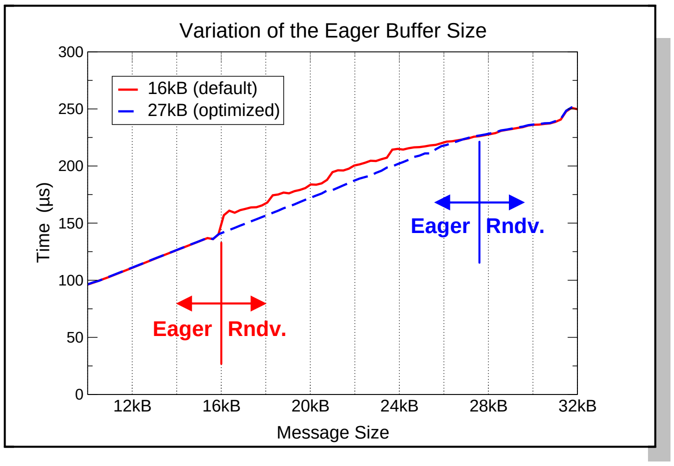
Carsten Clauss, Stefan Lankes, and Thomas Bemmerl

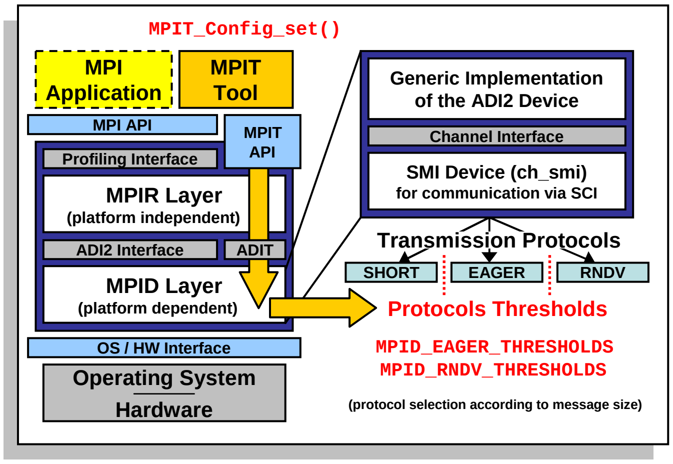
1. Introduction

Currently, the definition of the upcoming MPI 3.0 standard is under active discussion by the respective working groups of the MPI-Forum.
One of these working groups deals with the definition of an additional interface that should help to enhance the interaction between MPI implementations and additional tools like debugger and profiler.
In this context, it is intended to offer an additional set of tool-oriented functions within a separate namespace, namely the MPIT configuration and performance interface.
In contrast to the traditional PMPI profiling interface, that simply offers alternate entry points to MPI functions, the MPIT configuration and performance interface is rather a generic information interface that allows for querying MPI-internal configuration variables and performance counters.
While the set of routines of this new interface is to be defined by the new MPI standard, names and intent of the accessible configuration and performance variables are left to the respective MPI implementation.
For that reason, variable names and data types have to be retrievable via the MPIT interface, too, as well as their actual meaning.
However, the question arises how a certain tool should cope with these library-specific configuration and performance variables.

2. MP-MPICH

In order to evaluate the potential of this new information interface, we have prototyped an essential part of the proposed MPIT functions on top of an MPI library called MP-MPICH. MP-MPICH, which stands for Multi-Platform MPICH, is a modification and extension to the well-known MPICH distribution. It covers several subprojects like SCI-MPICH (support for SCI cluster interconnects), NT-MPICH (support for Windows operating systems) and MetaMPICH (a Grid-enabled MPI library) and conforms to most parts of the MPI-2 standard (including process-spawning and one-sided communication on certain platforms).
Since MP-MPICH is derived from the original MPICH, it also makes use of the three common message transmission protocols *Short*, *Eager* and *Rendezvous*.
The decision which of these is to be used is based on the respective message size.
Usually, the thresholds between the protocols are set as static configurations before application start, either depending on certain resource restrictions or according to communication characteristics of the MPI applications.
For example, when using SCI-MPICH, the optimal threshold between Short and Eager protocol depends on the size of the largest *atomic* data package within the SCI network (e.g. 64, 128 or 1024 Bytes, depending on the SCI Link Controller used). On the other hand, for an optimal threshold between Eager and Rendezvous protocol, the fraction of *unexpected messages* occurring during an application run can be the decisive factor.





3. Prototyping the MPIT Interface

By implementing the MPIT interface on top of MP-MPICH, it is now possible to query and even to adjust these protocol thresholds via configuration variables (`MPID_EAGER_THRESHOLDS` and `MPID_RNDV_THRESHOLDS` on verbosity level `MPIT_VERBOSITY_TUNER_BASIC`) for each pair of processes at runtime.
For this purpose, an application (or rather an adjustment tool utilizing the MPIT interface) can call the proposed `MPIT_Config_get()` function, returning an array that contains the respective threshold values for all remote ranks.
In turn, via the `MPIT_Config_set()` function, it can be attempted to modify these values at runtime. However, it's up to the underlying communication device (the so-called *channel device* of the ADI2 layer) to accept such a modification, or to reject it e.g. for resource reasons.

4. A Simple Tool on top of MPIT

In order to evaluate the impact of varying these configuration values, we have developed a small tool that makes use of the MPIT interface by adjusting the protocol thresholds during a simple Ping-Pong benchmark.
At first, the tool measures the communication time for messages of a length equal to the considered threshold. Then, the tool shifts the threshold (if possible), so that during a second measurement the alternative protocol is applied for the same message length.
By comparing the measurement results and repeating this procedure in a nested manner, the tool tries to optimize the threshold values.
An example for such an optimized protocol transition is shown in the Figure on the left side of this box.
Here, the measured communication times are plotted over the message sizes, once for a default threshold between Eager and Rendezvous of 16kByte, and once for an adjusted threshold of 27kByte. Obviously, messages of a size between these two values will benefit from shifting the transition from 16kByte to 27kByte in this case.
Although we have developed this simple threshold adjustment tool on top of MP-MPICH, we have designed the tool independent from the underlying MPI library.
For this reason, the actual name of the MPIT configuration variable representing the threshold to be optimized has to be passed to the tool by the user.

Announcement:

We are not part of the MPI 3.0 Tools Working Group, but we are observing the respective standardization process with great interest.

LEHRSTUHL FÜR BETRIEBSSYSTEME
Univ.-Prof. Dr. habil. Thomas Bemmerl

RWTH AACHEN UNIVERSITY