MPIT Breakout Group @ CScADS 2010

Participants:
Dorian Arnold, Bronis de Supinski, Markus Geimer, David Montoya, Heidi Poxon, Martin Schulz

During the MPIT breakout discussion the group focused on two main issues: (1) the definition of use cases that help to further motivate the inclusion of MPIT into the MPI standard and (2) a discussion on the actual API to understand its impact on tool design and possible extensions to cover additional needs from tool developers.

For the first part, the group collected several use cases, mainly driven by real-world experiences from the group participants, in which information like the one provided by MPIT would have been helpful. In particular we discussed the use of counters in “low-efficiency” paths in MPI (i.e., code segments in MPI that have to use lower efficient mechanisms or algorithms to work around resource contentions or similar constraints). This would help analysts to understand the segments of the code that stress the MPI implementation. Further, we discussed the option to time internal operations such as matching times or startup times between the call to MPI and the actual start of a message transfer. In particularly for timings, high watermarks, as intended by MPIT would be useful.
During the second part of the working group discussions, we looked at several of the API calls of MPIT with the goal of how tools could make use of them (and whether they are adequate) and how tools could transparently extend the offerings through MPIT, e.g., through additional instrumentation – a requirement voiced by several tool groups.

For the latter, we decided to suggest the inclusion of a profiling interface also for MPIT functions. This, combined with some extensions in the handle interface and a simplification of the variable detection mechanism, will allow tools to extend the MPIT offerings without much effort, but yet without increasing the complexity of the interface.

Second, the group felt that the option to set configuration and control variables is useful and should be included as well. We devised an enhanced scheme to deal with both read-only variables that cannot be set as well as with variables that require a global (or communicator specific) synchronization.

Additionally, we discussed many details of the MPIT proposal and suggested changes to namings (to reach a higher consistency), to allow a more flexible verbosity setting and to simplify the initialization and finalization. Finally, the group also addressed the question of whether Fortran bindings should be included. We concluded that we will need more feedback from the MPI/Fortran community through the MPI forum.

Moving forward, Martin Schulz will take these suggestions, use cases and comments back to the MPI working group on tools and will try to get them included into the MPIT draft, which will be presented later this year to the MPI forum. Further, discussions at CScADS have lead to plans to provide a prototype implementation through OpenMPI as well as an MPIT component for PAPI (in discussions with the PAPI group). The latter is the best path to common names for events across MPI implementations, which is a high priority concern for both tool developers and the MPI forum.