Abstract:

MPI is a library that is widely-used to develop applications to run in parallel. It provides an efficient profiling interface - PMPI - which is used by many profiling tools based on MPI. However, PMPI supports a single profiling tool at a time and the application must be re-linked with the tool each time a new tool is to be used. This one tool at a time nature of PMPI leads to monolithic tool designs, that are tailored for a certain MPI application and profiling goal. Hence, it discourages the reusability and modularity of the code leading to a very complex tool nature. On the other hand, if tools could work in coordination and could be utilized concurrently, desired tool functionalities could be achieved by combination of simpler tools. This approach would enhance testability, reusability and modularity of tools. The major question related to such an approach is, how to make tools work concurrently and in coordination with each other. In this thesis a prototype of an interface - called QMPI- , that enables concurrent and collaborative execution of tools is designed, implement and tested. The final design of the interface is made by enhancing and extending other previous designs, which are also a part of this thesis work. The
final design aims to achieve concurrent execution of tools by successive calls of tool wrappers in a nested fashion. The order of the tools in the toolchain is determined by a vector data structure, in which each tool has their corresponding instance. The final design is fully implemented along with some simple tools, that are used for testing the final design implementation. Finally, some performance tests are conducted to prove the efficiency of our QMPI prototype. It is shown that, linking QMPI with the MPI application - without using tools- causes negligible time overhead. Since QMPI provides such a low time overhead it can be linked with applications by default, in order to enable simple activation or deactivation of tools. Furthermore, it is also shown that our prototype implementation has linear time overhead scaling with the increasing size of the toolchain. It provides low time overhead for a realistic size of toolchain.

Übersetzter Abstract:
MPI is a library that is widely-used to develop applications to run in parallel. It provides an efficient profiling interface- PMPI - which is used by many profiling tools based on MPI. However, PMPI supports a single profiling tool at a time and the application must be re-linked with the tool each time a new tool is to be used. This one tool at a time nature of PMPI leads to monolithic tool designs, that are tailored for a certain MPI application and profiling goal. Hence, it discourages the reusability and modularity of the code leading to a very complex tool nature. On the other hand, if tools could work in coordination and could be utilized concurrently, desired tool functionalities could be achieved by combination of simpler tools. This approach would enhance testability, reusability and modularity of tools. The major question related to such an approach is, how to make tools work concurrently and in coordination with each other. In this thesis a prototype of an interface - called QMPI- , that enables concurrent and collaborative execution of tools is designed, implement and tested. The final design of the interface is made by enhancing and extending other previous designs, which are also a part of this thesis work. The final design aims to achieve concurrent execution of tools by successive calls of tool wrappers in a nested fashion. The order of the tools in the toolchain is determined by a vector data structure, in which each tool has their corresponding instance. The final design is fully implemented along with some simple tools, that are used for testing the final design implementation. Finally, some performance tests are conducted to prove the efficiency of our QMPI prototype. It is shown that, linking QMPI with the MPI application - without using tools- causes negligible time overhead. Since QMPI provides such a low time overhead it can be linked with applications by default, in order to enable simple activation or deactivation of tools. Furthermore, it is also shown that our prototype implementation has linear time overhead scaling with the increasing size of the toolchain. It provides low time overhead for a realistic size of toolchain.