CMST SI(1) 7-11 (2006) DOI:10.12921/cmst.2006.SI.01.07-11

COMPUTATIONAL METHODS IN SCIENCE AND TECHNOLOGY **Special Issue 2006**, 7-11

Efficient Programming and Optimization of Parallel Applications for Cluster Systems

Ullrich Becker-Lemgau

Intel GmbH

(Rec. 10 October 2006)

Abstract: Designing parallel applications and parallelizing sequential applications for cluster systems require powerful development tools to achieve scalable, reliable and highly efficient applications. The Intel Cluster Toolkit offers a toolset for development of MPI based applications and enables the developer to build, analyze, and optimize the parallel application. This article will describe the basic functionality and use of these tools.

Key words: Development Tools, Intel Cluster Toolkit, Cluster System, MPI Library, Performance Analysis, Performance Optimization, HPC

1. INTRODUCTION

Since several years parallel systems are becoming available for more and more users, applications, and application domains. This change is driven mainly by cluster systems build out of standard components. Due to the release of new dual- and multi-core processors for server, desktop, and laptop we nowadays can find parallel systems in any application area.

This trend states a big challenge for application developers: To benefit from parallel resources and thereby increase performance all serial applications have to be parallelized. This is true for server and desktop applications in equal measures. For desktop application very often thread level parallelism is used which is supported by compilers (OpenMP) and hardware (hyper-threading).

For cluster applications a different parallel paradigm is needed as a cluster consists of independent systems (often called nodes). Especially the memory of a cluster is distributed over the nodes and very often only accessible by the specific node. The connection between the nodes is very often realized by an Ethernet connection. But more and more high speed networks like Infiniband, Myrinet, Quadrics or similar solutions are used to enable higher throughput and lower latency.

For technical applications on cluster systems with distributed memory very often the Message-Passing-Interface[1] (MPI) is used as the parallel programming model. MPI is a standard describing the programming interface to a communication library which realizes the explicit distribution of data over the parallel processes and the necessary communication between these processes. A MPI application is a collection of processes with function calls to send data to other processes and function calls to receive data from other processes. This is the main difference to the OpenMP/thread model which is based on transparent programming language extensions (pragmas).

Within the MPI model the application programmer has to take care of the data exchange and synchronization between different processes of the application by calling functions of the MPI library. Thus, to parallelize an application with MPI the application itself has to be restructured. Most likely this will require more effort than a parallelization with the thread model. But the additional effort will lead to a portable, parallel application because MPI is a common standard with implementations for nearly every platform. MPI parallelized applications feature high scalability – adding new processes and processors will speed up the application accordingly. That's why MPI is very often used for parallel applications on cluster systems.

Lots of technical applications use MPI as the parallel programming model. Among them are applications like Fluent, Star-CD, Pam-Crash, LS-Dyna, Eclipse, MSC.Marc, *etc*. Because the cluster market is growing and already reached 50% of the overall server market the need for efficient and scalable parallel applications is growing too. More and more applications are made available for cluster systems. These trends are big challenges for the software developers and the developer tools. Lots of developers that don't have years of experience with MPI need to be able to write efficiently parallelized applications.

With the Intel Cluster Toolkit[2] in its upcoming version 3.0 Intel is offering an integrated collection of development

[1] http://www.mpi-forum.org

[2] http://www.intel.com/cd/software/products/asmona/eng/cluster/clustertoolkit/index.htm

tools that simplifies the programming and optimization of MPI based parallel applications. Besides libraries for code generation it contains analysis tools for performance optimization of parallel applications and systems.

2. MPI LIBRARY

The main component of the Intel Cluster Toolkit is the Intel MPI Library in its upcoming version 3.0. The Intel MPI Library is an implementation of the MPI standard and includes the full MPI-1.2 functionality and main aspects of the MPI-2 standard like one-sided communication and parallel I/O. All common high speed networks are supported which enables users to use the same MPI library for different applications on different cluster systems and different high speed networks (see Fig. 1).

Fig. 1. Intel MPI Library

For the software vendor this concept offers a very similar advantage: the parallel application does not have to be validated and released for each high speed network separately. Thus to support Infiniband, Myrinet, Gigabit Ethernet and other high speed networks the ISV (independent software vendor) only has to validate and release one of these. The network is no longer a differentiating factor for the release.

When starting the application the user decides which network should be used. For example the command

```
mpiexec –n 2 –env I_MPI_DEVICE sock a.out
```

starts the application with communication via TCP-IP on Ethernet. With

```
mpiexec –n 2 –env I_MPI_DEVICE rdma a.out
```

the same application is launched on Infiniband. The same executable is used in both cases. Thereby the performance advantage of high speed networks is directly available to the user with the same executable. For the ISV this implies lower development and maintenance costs while the application is supported on a wider variety of cluster systems.

Intel MPI Library uses different devices to support different networks. In the new version 3.0 all devices are integrated into one intelligent device that includes support for TCP/IP, shared memory communication and high speed networks via the DAPL interface. The DAPL interface provides the advantage that the Intel MPI Library will be decoupled from the underlying software stack. Thereby updates can be performed independently from the Intel MPI Library. Even new networks can be introduced without changing the application or the Intel MPI Library.

Intel MPI Library also supports a hybrid approach for the communication. A user might use the shared memory communication on the node level and a high speed network between the nodes to get the best of both models. This becomes more important if the nodes are fat nodes with lots of processors like the nodes of the SGI Altix series. With Intel MPI Library the user does not need to rebuild the application or get a new version from the ISV because Intel MPI Library supports this hybrid mode inside the intelligent devise.

3. MATHEMATICAL LIBRARY

Another building block for generating high performance applications is the cluster edition of the Intel Math Kernel Library (MKL). This cluster edition includes all functionality of the sequential MKL Library like BLAS, LAPACK, sparse matrix solvers and FFTs. Additionally the cluster edition provides an implementation of SCALAPACK functions for distributed, parallel applications. Thereby the well-known superior performance of MKL is also available for cluster systems and MPI applications.

4. PERFORMANCE ANALYSIS

While the MKL cluster edition is already optimized for the cluster platform this is not true for MPI applications per se. Performance optimizations of MPI applications can be a time consuming and difficult task without the right tool. At the same time MPI offers big potential for performance optimization of parallel applications because it is a powerful but complex parallel programming paradigm.

During the development process initially debuggers like Totalview[3], DDT[4] or IDB[5] with special functionality for parallel applications will be used. These tools offer great help during the analysis of incorrect code. But once the application is running without errors that doesn't mean the application scales well or is well optimized performance wise.

Here the Intel Trace Analyzer and Collector 7.0[6] sets in and enables a fundamental analysis of the parallel application and its communication structure. Like the name is indicating the Intel Trace Analyzer and Collector consists of two separate components. The Intel Trace Collector is a library that has to be linked with the parallel application. If the so linked application is executed Intel Trace Collector will record all MPI events like MPI function calls and logs them together with a time stamp in the trace file. In a second step – the post-mortem analysis – this file is read by the Intel Trace Analyzer and can now graphical and numerically analyzed by the developer.

[3] http://www.etnus.com
[4] http://www.allinea.com
[5] http://cache-www.intel.com/cd/00/00/21/92/219299_intel_debugger.pdf
[6] http://www.intel.com/cd/software/products/asmona/eng/cluster/tanalyzer/index.htm

Like all other tools out of the Intel Cluster Toolkit the Intel Trace Analyzer and Collector is available for Linux systems while the Intel Trace Analyzer can also be executed on a Windows system. Thus the performance analysis can even happen on the own Windows laptop. Windows version of all other Intel Cluster Tools are in preparation.

Intel Trace Analyzer offers a wealth of different methods to graphically analyze the collected data. The main instrument is the event timeline: For each process the event timeline shows in one row which function the process is executing at a certain time. Hereby the timeline distinguishes between MPI functions (red) and application time (blue), see Fig. 2.

The black lines indicate messages. It is easy to identify which process has send this message at which time and which process is receiving it at which time. Collective operations for global communication and synchronization are displayed as blue lines.

With the help of the event timeline the communication structure of the parallel application can be easily visualized and analyzed. In the example of Fig. 2 one can see a typical stair structure which shows a sequentialized region and indicates that the parallel resources are not very well used and the application is suffering a performance loss. This can also be detected by the long red bar in the function profile: the time spend in MPI functions exceeds the time consumed by the application itself. The displayed time interval can be adjusted by zooming or moving in the time axis. Intel Trace Analyzer offers lots of different ways to filter, group or select the displayed data. Thereby the developer is able to analyze even the smallest details of the parallel application and find performance bottlenecks.

Besides the event timeline and function profile like shown in Fig. 2 the Intel Trace Analyzer offers several other charts like the quantitative timeline and the qualitative timeline.

The quantitative timeline (Fig. 3) displays for each function how many processes are executing this function at a certain time. This chart enables the programmer to visualize the number of processes executing MPI function over time while the function profile shows the amount of time consumed over a time interval. So with the quantitative timeline the developer is able to detect local performance problems.

The qualitative timeline (Fig. 4) displays different attributes for functions, messages or collective operations. For example the developer can analyze transfer rates for messages over time. By zooming an interesting interval can be chosen and displayed in any detail. With the context menu the developer can get specific information for each message.

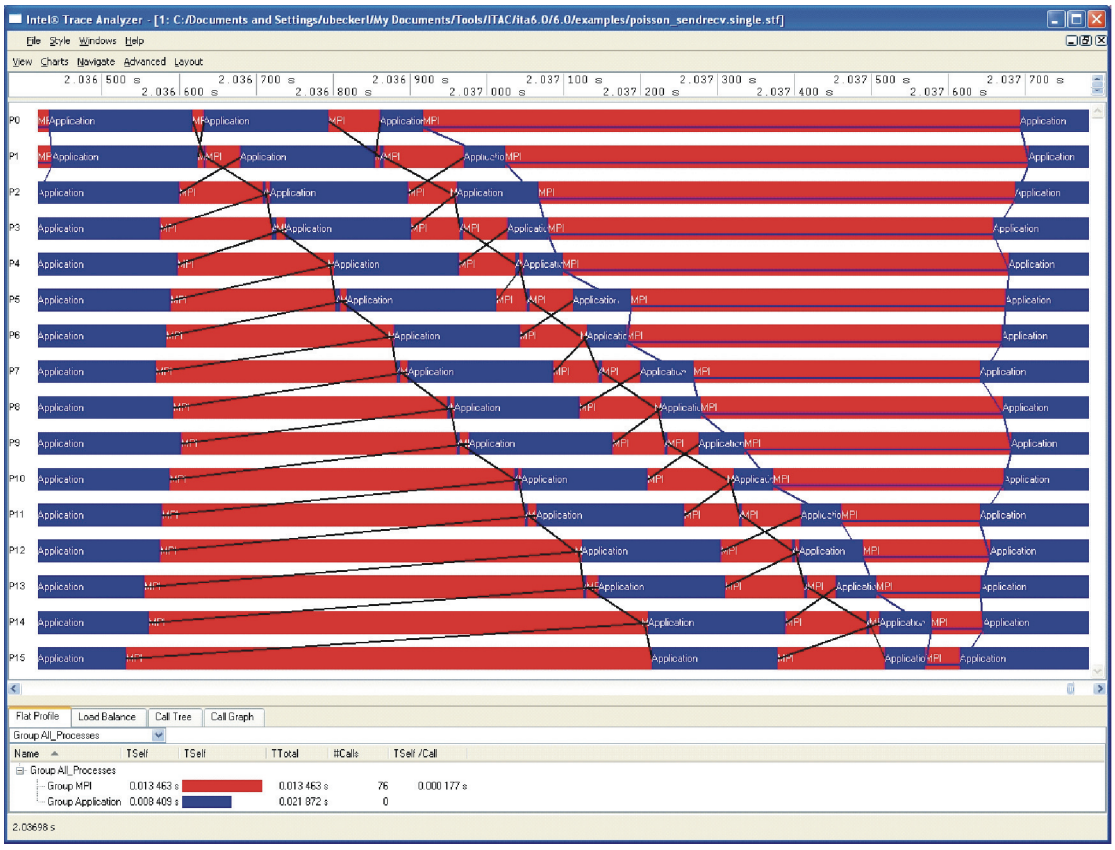

Fig. 2. Inter Trace Analyzer – Event Timeline

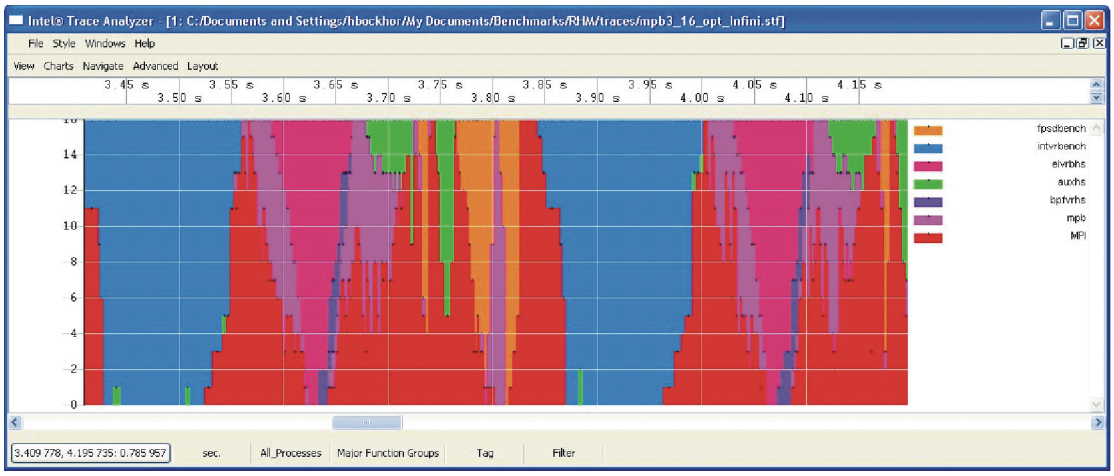

Fig. 3. Intel Trace Analyzer – Quantitative Timeline

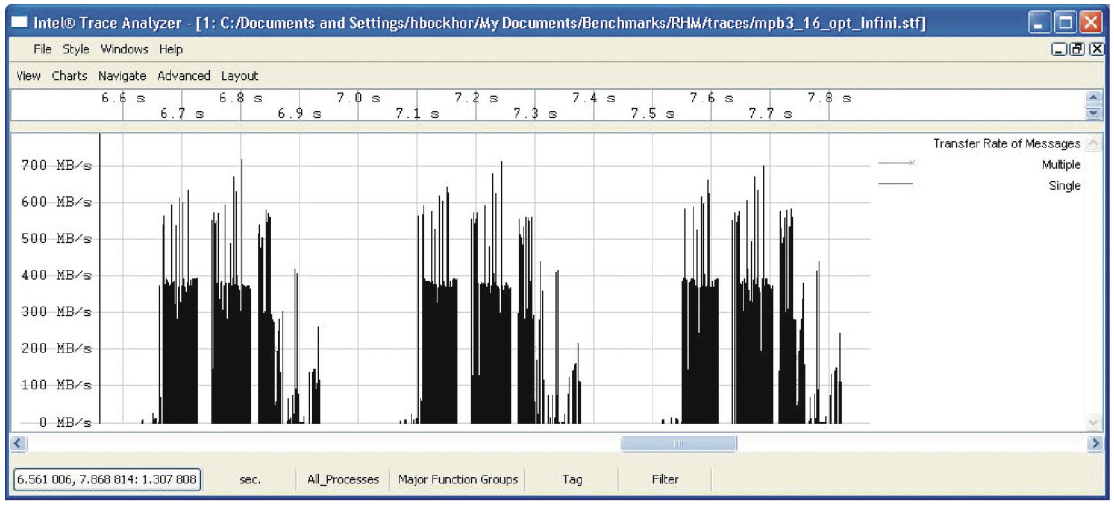

Fig. 4. Intel Trace Analyzer – Qualitative Timeline

The different timelines are complemented by several profiles which show statistical information or accumulated data over time. We already mentioned the function profile. Additionally the massage profile displays different attributes in matrix form for example the amount of data send between two processes. With the message profile the developer can easily detect irregularities and can analyze them in combination with the other charts.

Intel Trace Analyzer is not only displaying information about MPI communication. In case the application is instrumented with the API of Intel Trace Collector one can display the functional calls of and inside the parallel application. Intel Trace Analyzer is showing which process is executing which function of the application at what time. The developer defines through the level of instrumentation the level of detail for the application data to be recorded in the trace file. On 32-bit and 64-bit Pentium and Xeon systems additional information can also be recorded by using the binary instrumentation of the executable.

The upcoming version 7.0 of the Intel Trace Analyzer and Collector will introduce some very interesting enhancements for analysis of MPI applications. One of the main advantages is the new comparison chart which enables the developer to compare two trace files and zoom through the data in a synchronized way. Herewith the developer is able to see the impact of changes done to the communication structure and to do a before-after comparison.

The Intel Trace Collector 7.0 will also include the new Intel Message Checker functionality. The Intel Message Checker detects the most common MPI errors like possible deadlocks. Even if the application is running smoothly the MPI application might contain errors that only effect the execution or result for a specific workload. Intel MPI Checker improves code quality, stability, and correctness.

The Intel Cluster Toolkit includes also the Intel MPI Benchmarks as source code. This set of functions enables performance measurements of MPI implementations and cluster systems.

5. SUMMARY

With the upcoming version 3.0 of the Intel Cluster Toolkit Intel is offering a very valuable tool set for efficient developing, analyzing and optimizing MPI applications. The tool set contains libraries for code generation like the Intel MPI Library and the MKL Library Cluster Edition. To ensure an effective and scalable parallel application and to enable an easy development/optimization/maintenance phase tools like the Intel Trace Analyzer and Collector and the Intel MPI Benchmarks are included. The new Intel Message Checker functionality will add correctness checking to enhance quality and reliability of MPI applications.

ULLRICH BECKER-LEMGAU holds a diploma in mathematics from Justus-Liebig-University in Giessen and a doctoral degree in engineering from Technical University in Clausthal-Zellerfeld. He is active in the area of numerical simulation, technical computing and HPC since 16 years in various positions. Three years ago he joined the Software and Solutions Group at Intel as a Technical Marketing Engineer for Cluster Software.