MPI application tuning based on trace replay

E. Leksikov, A. Timonova, D. Durnov

Intel Corporation

Abstract - In this paper we present a new approach to MPI runtime library tuning based on an application communication profile. The approach allows the user to get a dramatic reduction in tuning time. The tuning approach based on a communication profile uses the same methodology as the application specific tuning, which uses a real application run on each tuning iteration. However, instead of the application we use only the "replay" of its communication pattern, which doesn't include computation, IO and other application-level time consuming parts. We describe some implementation details, results of our experiments, challenges and issues we have faced and the way they may be addressed in the future.

Index terms – MPI, autotuning, optimization, HPC.

Summary

MPI implementations have many optimization parameters that are intended to ensure optimal cooperation of a parallel system with the hardware. Such cooperation can be achieved by adjusting defaults for those parameters. Due to the fact that there are many optimization parameters and their tuning requires a lot of time, special tools were developed to automate the process. However, the existing tools spend a lot of time and resources searching for the "optimal" values. This article proposes a new approach to the problem of finding optimal environment settings. The new approach is meant to reduce the considerable tuning costs without significant loss of quality, which exist now.

Problem/Opportunity

Let us consider a few of the existing systems for application environment configuration. The Intel® MPI Library's[1] mpitune utility[2] supports several operation modes, but in this paper, we will consider only two of them: "Application Specific" and "Fast Application Specific".

The first mode ("Application Specific") is aimed at finding the optimal configuration of Intel® MPI Library for a specific application. This tuning mode uses the actual application and provides the most precise values, but the method is too expensive in terms of cluster time.

"Fast Application Tuning" runs Intel® MPI Benchmarks[3] instead of running real applications and focuses on those cases which are not covered by the out of the box settings for optimal performance of the application. For example, if an application uses non-typical message sizes.

Multiple experiments have shown that the existing tools handle the task very slowly or with fairly low quality. This means that the main drawback of the existing environment configuration systems is the enormous amount of time spent on the job to find the most optimal settings.

Solution

Since cluster time is an extremely expensive resource, it is proposed to perform tuning using an application that reproduces the communication pattern of the original application. Such an application doesn't execute the user code, which will significantly reduce the tuning time.

Figure 1 shows the structure of the tuning workflow based on the new methodology. According to the new methodology "to create what will be then re-played", it is necessary to create a trace of the original application. Among all the strategies of research, tracing was selected as an instrument allowing us to

retain the most complete picture of the analyzed application. Once the trace is ready, we can re-play it and get the desired "tuned" settings.

Adding heuristics for selecting small repeating patterns (independent parallel tasks or blocks) will reduce the trace replay execution time. For example, by reducing the number of iterations in loops or by executing only one instance of identical parallel subtasks.

Additionally, the trace replay technology can be used to create a benchmark that measures MPI performance taking into account communication pattern of the real application.

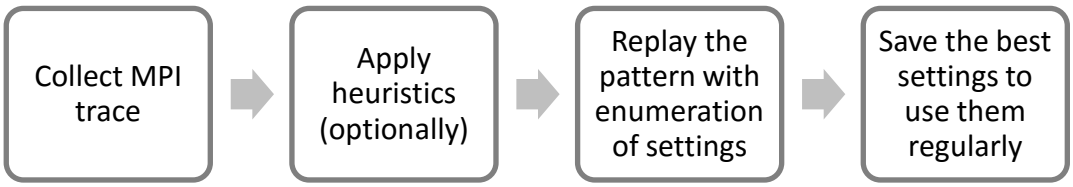

Fig. **1**. MPI tuning workflow based on the "replay" of pattern

Measurement results

Currently there is a prototype system that can handle some MPI calls. Figure 2 shows the trace of the original application (top graph) and the trace of replay work (bottom graph). The original application contains the MPI_Send and MPI_Recv operations, which are invoked between two processes.

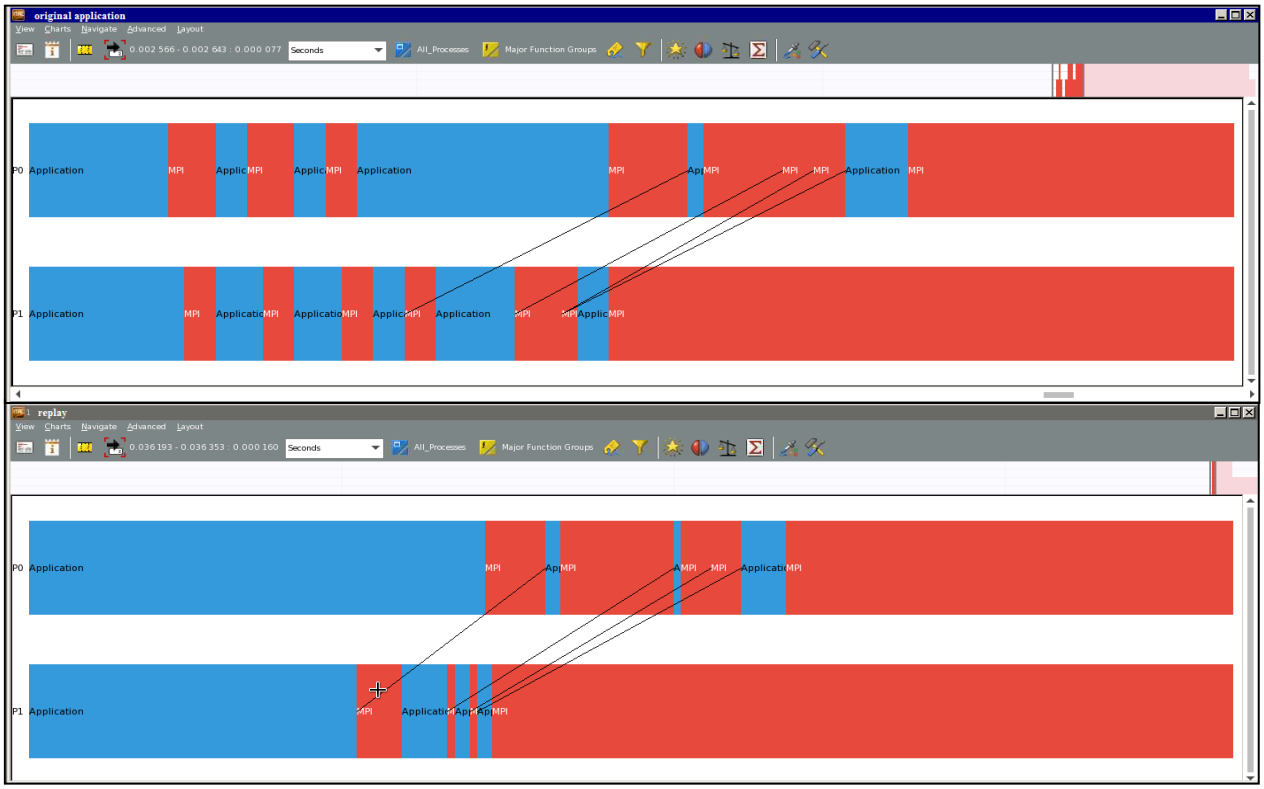

Fig. 2. Original application (top) and trace replay's work (bottom)

Point-to-point operations create one of the simplest communication patterns. The traces show that all of four MPI calls were retained. Besides, you may notice that some sections with MPI calls were placed very close one another. This happened because the user code did not execute.

The experiments have shown that tuning using the new methodology takes less time compared to the application specific tuning and simple enumeration of all parameters.

For testing purposes, a “handmade” micro benchmark was taken, which makes use of popular MPI operations. User’s part of code was emulated using the sleep() function, which pauses program execution for the specified time.

Figure 3 compares different tuning methodologies based on two criteria: speedup of the original application (dark bars) and tuning time (light bars):

1. Search of optimal settings for the original application using linear search of each parameter independently, the original application was used;
2. Search of optimal settings using the same methodology as above but the trace replay was used;
3. Search of optimal settings using complete enumeration of parameter combinations, the trace replay was used;
4. Search of optimal settings using the same methodology as in the 2nd approach but only for two functions – MPI_Bcast and MPI_Allgather.

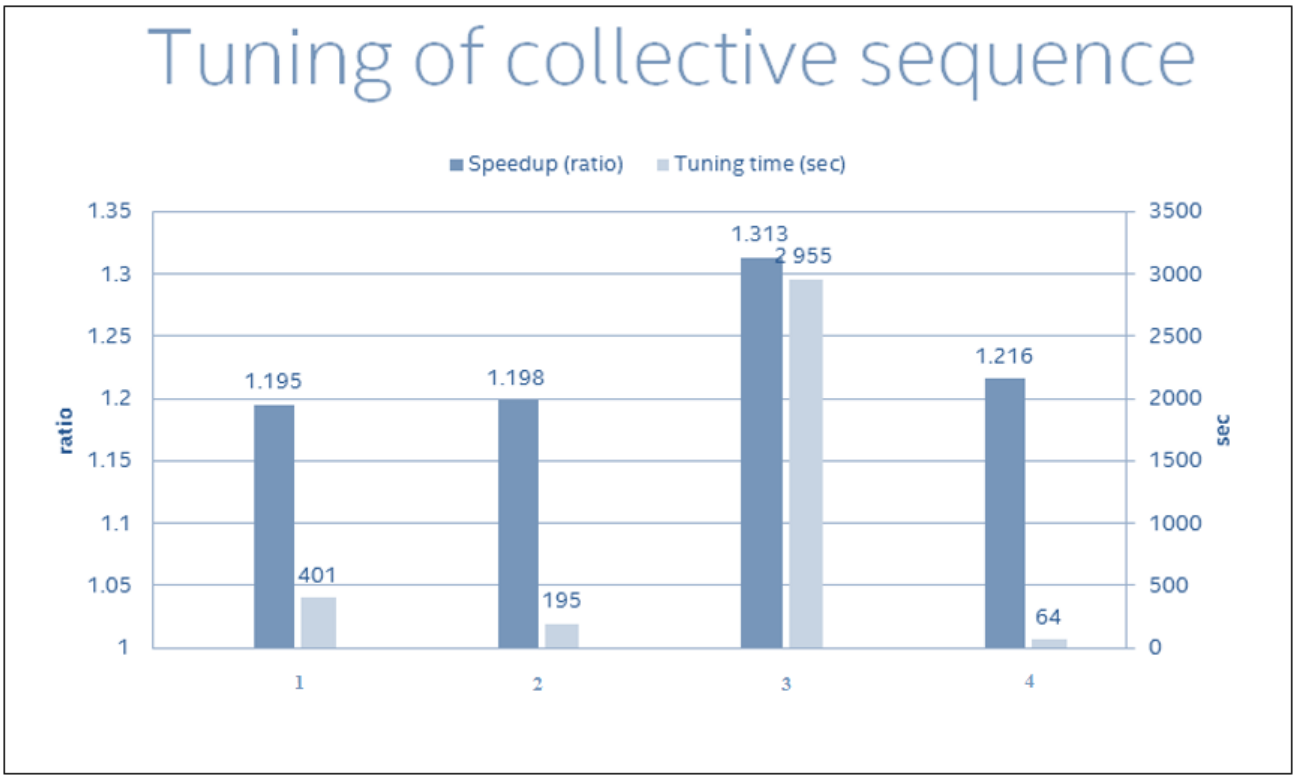

Fig. **3**. Comparison of autotuning methodologies

The data shows that the use of the new methodology significantly reduces time for finding the optimal settings for the application. Besides, the results of tuning using the new approach are comparable quality-wise with the results based on the original application execution: the speedup ratio varies only by the third number of the fractional part (1.195 vs 1.198).

Conclusions

The existing tools for tuning MPI applications consume a lot of time, which is an inadmissible use of resources for large problems. The new approach involves the use of an application trace replay instead of the original application, which will allow us to reduce time of search for optimal settings without loss of quality.

References

1. Intel® MPI Library for Linux* OS Developer Reference
   https://software.intel.com/en-us/mpi-refman-lin-html

2. Tutorial: Using MPI Tuner for Intel® MPI Library
   https://software.intel.com/en-us/mpi-tuner-tutorial-lin

3. Intel® MPI Benchmarks User Guide and Methodology Description
   https://software.intel.com/en-us/imb-user-guide