Simulation Environment for Studying Overlap of Communication and Computation

Vladimir Subotic
Barcelona Supercomputing Center
vladimir.subotic@bsc.es

Jesus Labarta
Barcelona Supercomputing Center
Universidad Politecnica de Catalunya
jesus.labarta@bsc.es

Mateo Valero
Barcelona Supercomputing Center
Universidad Politecnica de Catalunya
mateo.valero@bsc.es

***Abstract*—Overlapping communication and computation allows both processors and network to be utilized concurrently and leads to two clear benefits: overall speedup and a reduction in network performance requirements. Still, it remains unclear how much overlap can be actually achieved in practice – in real-world applications.**

This work designs a precise simulation environment that measures how much a scientific MPI application can profit from overlapping communication and computation. The simulation takes into account a wide range of application properties and allows to study overlap on the configurable platform. Additionally, the environment can visualize the simulated time-behaviors, so the non-overlapped and overlapped executions can be compared both quantitatively and qualitatively, providing new insights into the mechanism and potential of overlap. We found that the overlapping potential is very limited by pattern by which an application computes on the communicated data. Finally, we identified as the the biggest benefit of overlap the fact that it can highly relax network constraints without consequently degrading performance.

I. Introduction

As a parallel machine grows larger, its interconnect becomes increasingly expensive, therefore the community must find a way to use network resources more efficiently. In an effort to hide communication delays, the network designers deliver interconnects with better technological parameters, causing the cost of the interconnect to become a significant share in the total cost of the parallel machine. Still, due to bursty traffic, the overall network utilization in a real application remains quite low. As a novel approach to increase networks' efficiency, state-of-the-art networks provide support for overlapping communication and computation.

Although the research so far shows that overlap is a perspective avenue for advancing parallel execution, it remains unclear how much a real-world application can benefit from this technique. Prior studies mostly explored the implementation issues of *automatic overlap*, a mechanisms that is able to overlap communication and computation without a need to restructure the legacy code. The main idea of this mechanism is: to partition every original message into independent *chunks*; to send every chunk as soon as it is produced; and to wait for every chunk in the moment when it is needed for consumption. So far, there was little research to identify the potential for automatic overlap in legacy applications. Sancho *et al.* [1] theoretically estimated the potential overlap in scientific codes by modeling a real application with one iterative loop and parameters that roughly describe the computation pattern. We continue Sancho's work by designing a simulation environment that takes into account a much wider scope of application behaviors and reveals more information about the mechanism of overlap.

II. Simulation

This work is the first study that uses a simulation methodology to evaluate the potential of automatic overlap, thus offering a new, wider perspective on the overlapping mechanism. Prior research on the problem theoretically quantified the overlapping potential by defining a representative mathematical model of the applications [1]. Although useful for initial studies, such modeling of scientific codes misses some important execution properties and their influence on the final result. In particular, Sancho *et al.* [1] assumed that computation patterns by which an application accesses the communicated data are always ideal (sequential). Our simulation especially challenges this assumption by reconstructing the overlapped execution considering both real (measured) and ideal (assumed) computation patterns. Also, our methodology takes into account a much wider range of application behavior, without requiring to know the application's algorithm. Additionally, the simulated time-behaviors can be visualized, allowing to qualitatively inspect differences between the non-overlapped and overlapped execution and learn more about the mechanism of overlap.

A. Simulation environment

Our simulation environment consists of a new tracing tool that we designed and a legacy trace-file based network simulator. An MPI application executes in parallel, with each process running on its own Valgrind [2] virtual machine. Each of these virtual machines implements a designed tracing tool. The tool traces the original application and extracts the trace of the original (non-overlapped) execution, while at the same time, it generates what would be the trace of the potential (overlapped) execution. Then, the Dimemas simulator [3] uses the traces obtained from each MPI process and off-line reconstructs the application's time-behavior on a configurable parallel platform. Finally, the comparable time-behaviors can be visualized using Paraver visualization tool [4], allowing to profoundly study the

978-1-4244-6022-9/10/$26.00 ©2010 IEEE

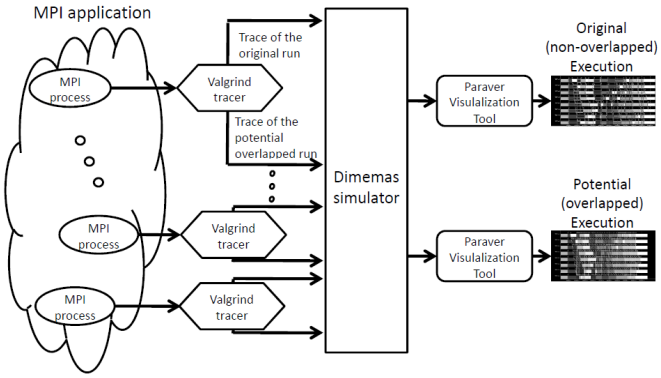

Fig. 1. Simulation environment

effects of automatic overlap. Figure 1 illustrates the described environment.

B. Implementation of the tracing tool

Our instrumentation leverages two key Valgrind functionalities for dynamic analysis of applications: wrapping function calls and tracking memory activities (loads and stores). Also, the tool needs additional data structures to keep track of the transfer's state and of the production/consumption progress of every chunk.

To control the instrumentation easier, the tool obtains timestamps in terms of the number of instructions executed in computation bursts. To represent time, the number of instructions is scaled by the average MIPS rate observed in a real run. The adopted notion of time ignores many real world effects such as: MPI routines overhead, memory hierarchy misses (cache, TLB...), CPU preemptions, etc. Therefore, the model captures the execution properties of interest and isolates the system from the undesirable and hard-to-control effects. However, the model can be extended to address these omitted effects.

From a single real run, the tracing tool generates various Dimemas traces – one non-overlapped (original) and several overlapped (potential), each of them addressing different overlapping mechanism. For the non-overlapped trace, the tool traces the original execution of a legacy code by emitting two types of Dimemas trace records: computation records specifying the length of the original computation burst; and communication records specifying the message parameters. For the overlapped trace, the tool identifies the points where partial data can be sent/is needed. Then it splits each original message into partial messages and injects the partial transfers at the identified points where data is finally produced (for sends) or actually needed (for receptions). Furthermore, in order to stress the influence of production/consumption patterns, the tracer generates the second overlapped trace in which partial transfers are uniformly distributed throughout the original computation burst, therefore modeling linear computation pattern. Moreover, due to its flexibility, the tool can make traces for executions that enforce only a subset of the overlapping mechanisms, so each of the mechanisms can be studied separately.

III. Results and findings

We found that the overlapping potential can be very limited by pattern by which the processes internally compute on the data involved in communication. Considering the real computation patterns, the potential for automatic overlap in the applications is negligible. Still, if the computation phases were restructured such that the data was produced and consumed in an ideal sequential order, automatic overlap could achieve benefits in a wide range of network bandwidth.

For ideal patterns, automatic overlap can achieve benefits in different ranges of bandwidth. For intermediate bandwidths, where time spent in communication is comparable to time spent in computation, overlapping can achieve a significant speedup, such as: 30% in NAS-BT, 10% in NAS-CG, 10% in POP, 40% in Alya, 65% in SPECFEM and 160% in Sweep3D. Indeed, one can expect the system to be exactly in this range of bandwidth, because the "realistic" network resources will result in communication delays comparable to computation time.

Finally, we found that the biggest benefit of overlap is that it can highly relax the expensive trend of advancing network bandwidth. Our results show that in the range of high bandwidths, the overlapped execution will need less bandwidth than the original execution to achieve the same performance. In fact, for achieving the performance of the original execution on some high bandwidth, the overlapped execution needs bandwidth that is couple of orders of magnitude lower.

IV. Conclusions and future work

Our study can be useful for researchers in the field to understand better the potential and the mechanism of overlap. The simulation framework can be useful to researchers that try to actually overlap communication and computation since it allows them to estimate the impact of their proposed implementations. On the other hand, our environment offers us to visually inspect the effects of overlap, therefore providing a profound insight into the overlapping mechanism.

We plan to continue developing the simulation environment in order to model more state-of-the-art network and MPI properties and to estimate the potential of new to-appear features of network systems. In order to have better control of the simulation methodology we have already started modifying the Dimemas simulator. We want to extend our environment to address state-of-the-art features of both network and MPI implementation. Further on, we believe that our simulation environment can become a powerful tool to model new ideas in network and MPI design and test their potential in real-world applications.

References

[1] J. Sancho, K. Barker, D. Kerbyson, and K. Davis, "Quantifying the Potential Benefit of Overlapping Communication and Computation in Large-Scale Scientific Applications," in *Supercomputing, SC 06*, 2006.
[2] N. Nethercote and J. Seward, "Valgrind," http://valgrind.org/, 2008.
[3] S. Girona, J. Labarta, and R. Badia, "Validation of Dimemas communication model for MPI collective operations," in *EuroPVM/MPI'2000*, 2000.
[4] V. Pillet, J. Labarta, T. Cortes, and S. Girona, "PARAVER: A Tool to Visualize and Analyze Parallel Code," in *WoTUG-18*, 1995.