Communication forecasting for large-scale applications

Nikela Papadopoulou*,Georgios Goumas*,Nectarios Koziris*

* *Computing Systems Laboratory, School of Electrical and Computer Engineering, National Technical University of Athens, Greece*

ABSTRACT

In this work, we deal with the challenging problem of forecasting the communication time of parallel applications. Performance models for parallel applications are significant to multiple decision-making processes, albeit the complexity of modern systems impedes their development. We approach modern large-scale systems in a topology-agnostic manner, we present several metrics of performance and we develop a performance model for point-to-point communication using statistical analysis and supervised learning for the Vilje supercomputer. Using our model, we predict the communication time of a 3D-Jacobi solver with a mean error of 23%.

KEYWORDS: Performance Prediction; MPI Applications; Interconnection Networks; Supervised Learning; Multiple Variable Regression; Benchmarking

1 Motivation

The enormity of compute power provided by parallel computing systems comes with a cost; collaboration between distinct processing elements, namely communication or synchronization, is a feature of most parallel applications that cannot be avoided. On large-scale systems, where applications execute on hundreds or thousands of processors, this cost can rise in a degree high enough to compromise parallel performance and scalability. The bottleneck in performance is aggravated by yet another constraint of parallel systems, resource sharing. Communication takes place over a limited amount of memory or set of network components, on which data streams of different sources and destinations interfere, while the complexity of the system prohibits the user of making any prior assumptions as to the impact of this interference on the parallel efficiency of an application.

Hoefler et al. ([HGTT10]) extensively discuss the significance of performance models for the tuning of parallel scientific applications. In principle, the ability to forecast performance of parallel applications enhances decision making at many levels: users' decisions for resource allocation, scheduling policies for large-scale systems and developers' options for applying and enabling code optimizations.

Interconnection networks for large-scale systems come with various architectures, topologies, configurations, routing protocols and protocol optimizations for specific MPI imple-

[1]E-mail: {nikela,goumas,nkoziris}@cslab.ece.ntua.gr

mentations, constituting a complex entity. The traditional metrics of latency and bandwidth do no longer suffice for characterizing modern interconnects. Several previous works attempt to assess complex features of interconnection networks and how different communication workloads respond to them. Bhatelé et al. attempt to quantify network contention by constructing a set of interesting benchmarks [BK09]. Hoefler and Snir [HS11] assess the impact of network topology, process mapping and congestion on performance and formulate new metrics for dilation, traffic and congestion with respect to topology mapping. In a recent work, Jain et al. [JBR+13] try to predict application performance for different topology mappings with supervised learning on a BlueGene/Q machine.

In this work, we present a supervised learning approach to performance modeling for point-to-point communication of MPI applications. In particular, we utilize a lightweight benchmark to extract knowledge about the system's architecture and interconnection network and their sensitivity to traffic flowing through various points of the system. We perform statistical analysis to identify possible predictors and their relation to observed communication times. We construct a model for communication time with selected predictors as independent variables and compute the model's coefficients with multiple variable regression, utilizing measurements from our micro-benchmark. Our approach diversifies from existing communication performance models, as our target is to model the time of a communication phase of an application, instead of modeling communication primitives separately. Moreover, we derive our predictors from the application's characteristics, while obscure network features are hidden within the model's coefficients.

2 A supervised learning approach to performance modeling

The case study of our work is MPI applications with iterative phases of computation and point-to-point communication, a pattern that applies to numerous real-life scientific applications, e.g. Jacobi solvers, finite element methods, molecular dynamic simulations, sparse matrix-vector multiplication. At present, we assume that computation and communication are two discrete phases and no overlapping takes place. We also assume that communication is non-blocking and that MPI primitives are placed in an order that minimizes unexpected messages. Our study excludes MPI datatypes, since their modeling falls to modeling computation and not communication, thus we adopt a programming model where MPI messages are manually packed and unpacked.

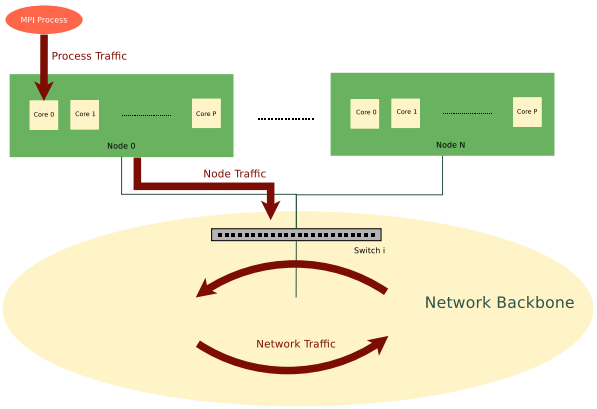

Figure 1: Traffic on large-scale systems

Despite the multitude of network configurations and their special characteristics, only a few topologies dominate the landscape of supercomputing: tori, fat trees and hypercubes and the corresponding system architectures bare similarities. Usually, a bunch of multiprocessors -the compute nodes- are assembled on a switch chip and the switch chips, along with links and possibly other network components, as in the case of fat trees, form the network backbone. This abstraction is depicted in Figure 1. When an MPI application executes

on the system, during the communication phase, streams of data of different volume flow through network points, which allows us to define three new metrics for traffic: *process traffic* is the product of the number of messages exchanged by an MPI process and the message size, *node traffic* is the product of process traffic and the number of processes residing on a node during execution and *network traffic* is the product of node traffic and the number of nodes of the allocation. In accordance to our topology-agnostic approach, we also denote two allocation-specific metrics as indicators of communication performance: *processes per node* that may indicate parallelism, but could also indicate contention for shared resources on the end-node and the switch on which it is attached and *distance* defined as the maximum number of hops, assuming shortest-path routing, a message could travel within the allocation. The latter is a measure for latency, but could also be a measure for contention on network links, since a message traveling a long distance would cause congestion on more links. However, a large distance on k-ary n-cubes also implies more available paths for routing, thus parallelism. The application's communication pattern and problem size introduce two additional variables: *message sizes*, which are relative to traffic metrics and can encapsulate the effect of protocol changes on communication time and *messages per process*, that indicate the overlapping capacity of the network.

To build and train a performance model, it is necessary to obtain information about the underlying architecture. To achieve this, we have incrementally devised an MPI micro-benchmark, based on the WOCON benchmark described in [BK09], where each process sends and receives n messages of equal size to n randomly selected processes. Communication takes place simultaneously for all processes, creating contention and congestion effects that we wish to include in our measurements. This burst of data exchange is a common behavior of MPI applications with iterative kernels of discrete computation and communication phases. The benchmark utilizes non-blocking point-to-point MPI communication primitives and its output is the maximum round-trip time observed among the processes. The metrics described earlier can easily be computed for all tested configurations, if the message size and the allocation's parameters are known.

Elementary variable selection is performed with correlation analysis between the metrics and the observed communication time. Process, node and network traffic are highly-correlated with communication time and thus are the de facto predictors to be included in the model. However, correlation analysis itself can be deceptive, as metrics that appear to be non-correlated with the result can be very useful for prediction when interacting with other variables. This can easily be the case of discrete variables and their effect on communication time can be identified using clustering for one of the highly-correlated variables, according to the values of the non-correlated variable. These variables can then be included in the model as interaction variables.

After the selection of predictor variables, our model takes the form of a linear model with interaction terms. To compute the model's coefficients, we perform multiple variable regression, using the micro-benchmark's results as the training set. Once the coefficients are computed, the model can be used to predict new values.

3 Evaluation and future work

We applied our methodology on Vilje supercomputer (#82 at Top500) at NTNU, an SGI Altix ICE X distributed memory system that consists of 1440 nodes. The network topol-

ogy is an enhanced hypercube. We executed our micro-benchmark on the machine for all possible configurations for 8 to 64 nodes, 1 to 16 processes per node, 1 to 4 messages and message sizes varying from 1B up to 16MB. We built a performance model following the methodology described in section 2 and used the LinearModel class of MATLAB R2012b to perform multiple regression. To evaluate our model's ability to predict the communication time of real-life applications, we executed a 3D-Jacobi solver, an application with an iterative 3D-halo exchange communication pattern, for various problem sizes and various configurations from 16 up to 4096 cores. We then predicted the per-phase communication time of the application utilizing our model. Our results demonstrate that our model can predict the communication time of the application with an error of 30% for about 70% of the different configurations, while the mean error of all predictions is 23%.

As a future work, we intend to improve the accuracy of our model, by studying and modeling more complex features of parallel communication and possibly improving our benchmarking methodology. We also wish to extend our methodology for building communication performance models on more supercomputers with different network architectures and topologies and on predicting the communication time of applications with more intricate communication patterns. More importantly, we hope to automate the performance modeling process, in order to construct a generic, portable tool for performance prediction.

4 Acknowledgements

This research was partly funded by project I-PARTS: "Integrating Parallel Run-Time Systems for Efficient Resource Allocation in Multicore Systems" (code 2504) of Action ARISTEIA, co-financed by the European Union (European Social Fund) and Hellenic national funds through the Operational Program Education and Lifelong Learning' (NSRF 2007-2013). We would like to thank NTNU for granting us access to Vilje Supercomputer.

References

[BK09] Abhinav Bhatelé and Laxmikant V Kalé. Quantifying network contention on large parallel machines. *Parallel Processing Letters*, 19(04):553–572, 2009.

[HGTT10] Torsten Hoefler, William Gropp, Rajeev Thakur, and Jesper Larsson Träff. Toward performance models of MPI implementations for understanding application scaling issues. In *Recent Advances in the Message Passing Interface*, pages 21–30. Springer, 2010.

[HS11] Torsten Hoefler and Marc Snir. Generic topology mapping strategies for large-scale parallel architectures. In *Proceedings of the international conference on Supercomputing*, pages 75–84. ACM, 2011.

[JBR+13] Nikhil Jain, Abhinav Bhatele, Michael P Robson, Todd Gamblin, and Laxmikant V Kale. Predicting application performance using supervised learning on communication features. In *Proceedings of SC13: International Conference for High Performance Computing, Networking, Storage and Analysis*, page 95. ACM, 2013.