Parallel Computing 000 (2017) 1–19

Contents lists available at ScienceDirect

Parallel Computing

journal homepage: www.elsevier.com/locate/parco

Prediction of the impact of network switch utilization on application performance via active measurement☆

Marc Casas[a,*], Greg Bronevetsky[b]

[a] Barcelona Supercomputing Center, Universitat Politècnica de Catalunya, Spain
[b] Google Corporation, United States

ARTICLE INFO

Article history:
Received 8 March 2016
Revised 19 June 2017
Accepted 20 June 2017
Available online xxx

Keywords:
Performance modeling
Resource sharing
Measurement techniques

ABSTRACT

Although one of the key characteristics of High Performance Computing (HPC) infrastructures are their fast interconnecting networks, the increasingly large computational capacity of HPC nodes and the subsequent growth of data exchanges between them constitute a potential performance bottleneck. To achieve high performance in parallel executions despite network limitations, application developers require tools to measure their codes' network utilization and to correlate the network's communication capacity with the performance of their applications.

This paper presents a new methodology to measure and understand network behavior. The approach is based in two different techniques that inject extra network communication. The first technique aims to measure the fraction of the network that is utilized by a software component (an application or an individual task) to determine the existence and severity of network contention. The second injects large amounts of network traffic to study how applications behave on less capable or fully utilized networks. The measurements obtained by these techniques are combined to predict the performance slowdown suffered by a particular software component when it shares the network with others. Predictions are obtained by considering several training sets that use raw data from the two measurement techniques. The sensitivity of the training set size is evaluated by considering 12 different scenarios. Our results find the optimum training set size to be around 200 training points. When optimal data sets are used, the proposed methodology provides predictions with an average error of 9.6% considering 36 scenarios.

1. Introduction

HPC applications demand very capable communication networks to support their high message and data volumes and/or tight synchronizations. Indeed, constraints on available network bandwidth or latency as well as network hotspots induced by specific communication patterns are often the key bottleneck that limit application performance [2,4,16,17,37,42]. Looking into the future, it is expected that the computational capabilities of individual computing nodes will continue to rise

☆ With the support of the Secretary for Universities and Research of the Ministry of Economy and Knowledge of the Government of Catalonia and the Cofund programme of the Marie Curie Actions of the 7th R&D Framework Programme of the European Union (Expedient 2013 BP_B 00243). The research leading to these results has received funding from the European Research Council under the European Union's 7th FP (FP/2007-2013) / ERC GA n. 321253. Work partially supported by the Spanish Ministry of Science and Innovation (TIN2012-34557).

* Corresponding author.
E-mail address: marc.casas@bsc.es (M. Casas).

http://dx.doi.org/10.1016/j.parco.2017.06.005
0167-8191/

faster than the capabilities of the networks that connect them [27]. This means that application performance will become increasingly bottlenecked on the capabilities of the network, making it even more imperative for application developers to optimize their applications taking network performance into account. Specifically, developers will need to (i) predict how their applications will perform on future systems with poorer network-to-node performance ratios and (ii) develop ways to assign computing work to available resources to effectively balance network communication and on-node computation. To achieve these tasks developers will require powerful tools to enable them to understand the interactions between their applications and the networks they run on and how these interactions ultimately affect application performance. Specifically, two directions of this interaction will need to be quantified for developers. First, tools must quantify how the application's communication utilizes the network and whether the application's needs are approaching the limits of the network's capabilities. Second, tools must measure how the capabilities of the network influence application performance and most importantly, whether the network is the application's performance bottleneck. These analyses must apply to both current and future systems, as well as to both static and highly configurable applications (e.g. where the space of possible configurations is too large to be explicitly enumerated and analyzed).

This paper eluates a new approach to measure the relationship between network capacity and application performance [9]. Our basic insight is that this relationship should be modeled as the application consuming a resource provided by the network. As more of this resource is available, the application runs monotonically faster, with reduced improvements as application performance becomes bottlenecked on other resources. Further, if multiple software components (entire applications or individual tasks such as processes, threads or Charm++ chares [21])) run concurrently on the same network, they will share its resources. This sharing can be modeled as one component consuming some amount of network resources, making it unavailable to others and thus causing them to behave as if they were running on a less capable network. The heart of this idea is a "performance relativity" principle, that "from the perspective of software components less capable networks behave very similarly to networks that are partially utilized by other software components".

When several software components share a network there can be many more potential issues than when just a single software component runs on a dedicated but slow network. These issues have been measured in detail [5,20,22] concluding that interference strongly contributes to performance degradation. Since the slowdown suffered by each particular software component can be emulated by running each one of these components in a dedicated and slow network, the "performance relativity" principle holds even though the plethora of issues that can be raised by network interference. This principle enables two novel measurement techniques that can answer the above questions:

Impact experiments measure a software component's use of the network based on the latency of a few additional packets sent over the network while the component runs. These measurements directly quantify the network's ability to carry application communication and can be used to determine whether the network is congested and measure how close the application is to fully utilizing the network. The additional packets are triggered by extra tasks running on dedicated cores and they do not impact applications' performance as the extra load is very low. We presented this idea in a previous paper [9] and some subsequent similar ideas have been proposed [15].

Compression experiments measure the relationship between network capability and a software component's performance. The component is executed concurrently with a micro-benchmark that runs on cores connected to the same network and sends varying amounts of communication. As the effective network capability is varied we observe the component's resulting performance, which corresponds to how it will perform on less capable networks or when more software components are executed on the same network. The idea of inserting messages and study its subsequent performance impact at parallel applications has been previously explored [36].

Finally we present several techniques that combine the two measurements to predict the performance degradation that a given combination of software components would suffer when executed concurrently on the same network. Each technique is based on a particular description of the available network capability when an application is running. Data from Impact measurements is used to compute latencies of the triggered packets. We consider four different approaches to describe the available network capacity when a particular parallel application is running: i) The average latency of all the packets triggered by the application ii) The average latency and the standard deviation of the packets triggered by the application iii) the histogram of the latencies of the packets triggered and iv) a mathematical queue [40]. By measuring the network capability that is left available while a given application or the Compression benchmark runs we can estimate the effect of multiple concurrent software components on each other as they share a network. The experimental and analytic procedures presented in this paper are focused on single-switch networks that connect multiple computing nodes.

This paper extends our previous work [9,10] by building performance models from multiple data sets, that is, by providing a complete evaluation of the performance prediction sensitivity with respect to the training set size. Our approach improves upon the state of the art in network performance modeling and measurement in at least 4 ways. The first 3 were already presented [9,10] and are revisited in this paper, while the fourth one is completely new:

i) Impact experiments of network utilization and contention are significantly faster than similar analysis performed inside simulators and apply to real physical networks for which precise models may not exist due to intellectual property restrictions. Further, unlike indirect measurement techniques, Impact experiments directly probe the network's ability to carry out the application's communication requests. Since they focus on just the network and quantify its effective capabilities in terms of a generic queue-oriented metric, these experiments provide a simple and unfiltered view onto this resource.

Please cite this article as: M. Casas, G. Bronevetsky, Prediction of the impact of network switch utilization on application performance via active measurement, Parallel Computing (2017), http://dx.doi.org/10.1016/j.parco.2017.06.005