Characterizing Application Sensitivity to Network Performance

Eli Rosenthal
Brown University
eli_rosenthal@brown.edu

Edgar A. León
Lawrence Livermore National Laboratory
leon@llnl.gov

I. Introduction

Most scientific applications use MPI to leverage parallelism across multiple nodes. As future architectures scale up the number of nodes in the system, the characteristics of the network become increasingly important for application developers and those responsible for future hardware procurements. When procuring a future supercomputing cluster, for example, there is a range of options in terms of processor, memory, and network performance. A question one may ask is whether doubling the network bandwidth is worth sacrificing upgrades in processor speed (assuming a fixed capital budget). The right balance, of course, depends on the suite of applications that will execute on such a system. The more we understand the characteristics of our applications, the better decisions we can make to provide the best performance within a given capital or power budget.

In this work, we provide an empirical study of the network requirements of a representative suite of HPC applications and the impact of network speed on their performance. First, we employ a subset of the CORAL mini-applications[1], which represent U.S. Department of Energy workloads and technical requirements. The CORAL benchmarks are the result of a joint *C*ollaboration between *O*ak *R*idge, *A*rgonne, and *L*awrence Livermore National Laboratories and provide simplified source code that contains the data access patterns and computational characteristics of larger production codes. Second, we characterize the communication requirements of these mini-applications in terms of their point-to-point (P2P) and collective operations and message sizes. And, third, we leverage multirail (multiple network interfaces per node) networking as a vehicle to examine improvements in network performance. When using multirail we evaluate the performance of several policies to understand locality and affinity tradeoffs.

II. MPI Characteristics of Applications

First, we characterize the communication characteristics of our benchmark suite, which is shown in Table I, using mpiP[2] on the Lawrence Livermore National Laboratory supercomputer *catalyst*. Catalyst has 324 dual-socket nodes with Intel IvyBridge Xeon processors; each socket has 12 cores. Catalyst includes dual-rail Quad Data Rate (QDR-80) Intel TrueScale fabrics and runs the TOSS (based on RHEL) operating system.

Figure 1 shows the communication characteristics of our applications using two configurations: MPI (1 task per core) and MPI+OpenMP (1 task per socket with 12 threads per task). The MPI performance characteristics were similar for these two configurations. We observed that most applications spend more than half of their MPI execution time performing collective operations. The exceptions here are AMG and MCB. AMG spends a large amount of time on P2P calls. In addition, it spends by far the highest percentage of execution time on communication. Despite taking a large percentage of execution time, AMG sends mostly small P2P messages. With the exception of MCB, collective operations send significantly less data across the network than P2P operations in the same application. LULESH spends nearly all of its MPI time making calls to Allreduce with a message size of 8 bytes, while a number of Isend calls incur orders of magnitude more bandwidth while requiring roughly one tenth of the time compared to the Allreduce calls.

TABLE I. Our CORAL benchmark suite.

UMT2013	Unstructured Mesh deterministic radiation Transport
AMG2013	Algebraic Multi-Grid linear system solver
MCB	Monte Carlo transport
LULESH	Shock hydrodynamics for unstructured meshes
miniFE	Finite element code

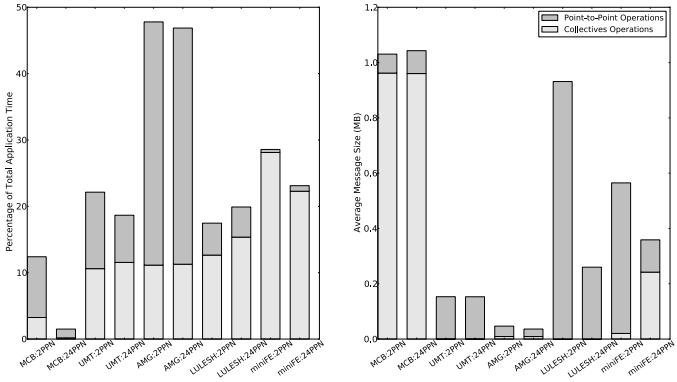


Fig. 1. Collective and P2P breakdown as a percentage of total execution time and their average message size. Experiments executed on 256 nodes for MPI (PPN=24) and MPI+OpenMP (PPN=2) configurations.

These results indicate that the sensitivity of MPI performance to bandwidth is largely application-dependent. In AMG or UMT, we would expect increased bandwidth to be helpful. However, in LULESH or miniFE factors affecting latency such as system noise and message injection rate may impact their performance more than efforts to increase bandwidth.

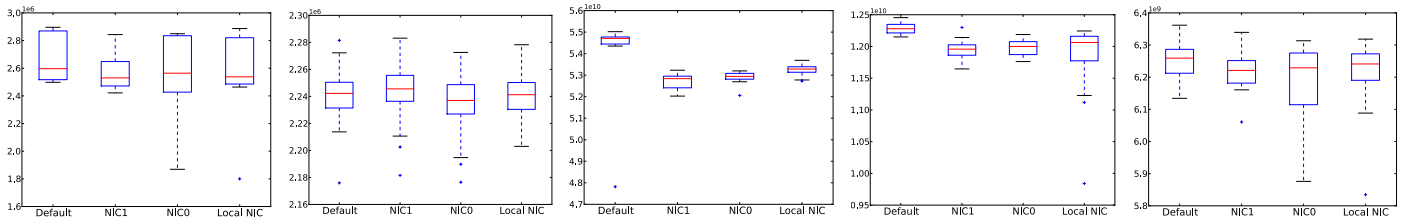
III. Impact of Multirail Networking

Multirail networks consist of multiple network interface controllers (NICs) per node. Catalyst provides two InfiniBand cards per node, each placed in proximity to a socket. The low-level network layer, PSM, provides options for binding

Prepared by LLNL under Contract DE-AC52-07NA27344. LLNL-ABS-658997.

[1]https://asc.llnl.gov/CORAL-benchmarks/

[2]http://mpip.sourceforge.net/



(a) miniFE (b) LULESH (c) UMT (d) AMG (e) MCB

Fig. 2. Application performance, represented as Figure of Merit, as a function of different network configurations.

network traffic from a given task to a given card. To understand the impact of increased bandwidth and affinity to a local NIC, we instrumented the following policies.

Default	PSM default dual-rail policy. It allocates MPI processes to the NICs in an alternating or round robin fashion
NIC-0	Route all traffic through card 0
NIC-1	Route all traffic through card 1
Local-NIC	Route traffic from each socket through its local NIC

Previous work highlights the benefits of striping individual messages across two NICs to increase bandwidth and decrease latency for MPI calls. Most of this literature does not focus on realistic or representative workloads of scientific applications. As demonstrated in Section II, a medium-sized run of 256 nodes shows that many key benchmarks are not network bandwidth-bound.

Before our CORAL analysis, we make the following observations about our policies and network configuration. First, we consider both policies NIC-0 and NIC-1 because even though the hardware configuration is symmetric, NIC 0 is used for system-level traffic. Depending on the application, this network *noise* may have an impact on performance. Second, the PSM driver version installed on catalyst does not instrument striping of messages across NICs or using a local NIC (thus the need for our Local-NIC policy) at the PSM level. And, third, even though MVAPICH provides dual-rail policies, they are not implemented for Intel fabrics. Our goal is not to provide a comprehensive study of multirail policies but to understand the impact of the network on application performance.

A. CORAL suite

Each of the CORAL benchmarks provides a metric called *Figure of Merit (FOM)* indicative of application performance. We measured the FOM of each benchmark under our network policies using both MPI and MPI+OpenMP configurations. In Figure 2, we summarize the pure MPI results. Because of nontrivial levels of variation between runs (see Figure 3), we ran all applications between 80 and 500 times and saw negligible difference in application performance with the different network policies (e.g., single- vs dual-rail). Despite significant time spent on P2P communications, AMG only benefits from the increased bandwidth (multirail) on the order of 2%. This is due to the relatively small average P2P message size, which indicates that MPI performance for AMG is driven more by latency and synchronization overhead than bandwidth.

UMT exhibited the most gains from multirail, but even here performance only increased by approximately 3.2%. As Figure 3 indicates, most of these gains are due to reductions in time spent on P2P operations (which are as much as 18% faster overall), with collectives showing relatively modest improvements. This observation helps explain why miniFE and LULESH showed no appreciable difference in performance. While AMG and UMT spend significant time on bandwidth-bound P2P operations, UMT's messages are larger on average, making it more sensitive to bandwidth increases. In contrast, miniFE and LULESH heavily rely on collective communications, where performance is dominated by issues of synchronization, communicator size, and latency. The outlier here is MCB, which only spends 2% of execution time on MPI calls and does not seem to provide meaningful comparison with other applications.

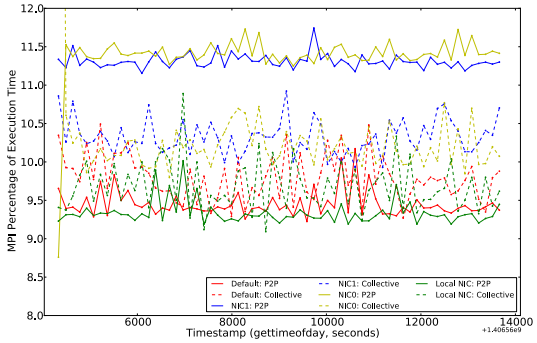


Fig. 3. P2P and collectives breakdown for UMT over successive runs. The variations on a run-to-run basis are largely indicative of system noise.

IV. Conclusions

Increased network bandwidth (multirail) may be of limited benefit for many applications. Despite substantial improvements on microbenchmarks, the communication patterns of representative scientific workloads seem to benefit only slightly or not at all from increased network performance. These applications are not bandwitdh bound; they send mostly small messages, and many of the larger messages are performed asynchronously. Substantial network-level costs lie in the latency for small messages and load imbalances across tasks, which are application dependent.

We performed this investigation on a small–medium scale, low-contention cluster. Future systems will be much larger and have busier networks with more contention. These commodity systems, those running largely unmodified Linux kernels, will also have greater sources of system noise. All of these factors increase the effective latency of communication that could potentially be mitigated by the use of multirail networks and, due to the relatively small scale of the catalyst cluster, will be studied as part of our future work.