Interferences between Communications and Computations in Distributed HPC Systems

Philippe Swartvagher

Interferences between Communications and Computations in Distributed HPC Systems

Philippe Swartvagher

Inria Bordeaux – Sud-Ouest, 33405 Talence, France
philippe.swartvagher@inria.fr

Abstract. Overlapping communications with computations in distributed applications should increase their performances and allow to reach better scalability. This implies, by construction, communications are executed in parallel of computations. In this work, we explore the impact of computations on communication performances and *vice-versa*, with a focus on the role of memory contention. One main observation is that highly memory-bound computations can have a severe impact on network bandwidth.

Keywords: HPC · MPI · Memory Contention

1 Introduction

Doing in parallel communications and computations (with non-blocking MPI calls or even more complex systems such as task-based runtime systems) is an increasing trend to get higher performances. It has been observed [3, 2], that, sometimes, when computations and communications are executed side by side, communications are slower than nominal performances and computations can also be degraded.

Since possible interactions between communications and computations, and especially the impact on communication performances, are not well detailed in the literature (but only mentioned), we propose in this work to study the possible causes of these interferences and measure their impact on both communication and computing performances. Reported observations here deal with the impact of memory contention. Since we target HPC systems, we consider only fast networks (here, InfiniBand) as well as inter-node communications.

We describe here our experimental protocol, provide first results and then discuss future work.

Advisers: Alexandre Denis and Emmanuel Jeannot
Inria Bordeaux – Sud-Ouest, 33405 Talence, France
{alexandre.denis,emmanuel.jeannot}@inria.fr

2 Methodology

Our goal is to measure performances of communications and computations when they are run side by side. To achieve this, we have designed a multithreaded and parallel benchmark using MPI+OpenMP. Per MPI process, one thread is dedicated to communications (it submits communication instructions and ensures MPI progression) and other threads (driven by OpenMP) do computations. This configuration mimics how some runtime systems work. The communication benchmark performs ping-pongs to measure network latency and bandwidth.

2.1 Benchmarking protocol

We need to compare performances of communications and computations when they are executed alone and when they are executed together. Therefore we decomposed our benchmark into the following steps:

1. Computation without communication
2. Communication without computation
3. Computation with side by side communication

Computations and communications use different data and hence are completely independent. Plots to represent results compare performance of communications and computations when they are executed separately or simultaneously. The former are represented by plain curves and the later by dashed curves.

Regarding performances of communications, we use two metrics: *latency* by exchanging 4 bytes of data (one `float`), and *bandwidth* evaluated for 64 MB message size.

According to which effect we want to observe, instructions done by computing threads will be different. Computations have to be embarrassingly parallel to always get the maximum level of parallelism and avoid an overhead caused by scheduling and dependencies. Moreover we use weak scalability to easily change the number of computing cores (and thus change the level of parallelism).

To see the impact of memory contention caused by data used for computations and data used for communications, we generate memory contention by computing cores doing memory-bound kernels: `COPY` ($b[i] \leftarrow a[i]$) and `TRIAD` ($c[i] \leftarrow a[i] + C \times b[i]$) from the STREAM benchmark suite [4]. Moreover, to really produce memory contention, we allocate memory on a single NUMA node, to increase the traffic on the memory bus between cores belonging to different NUMA nodes. The performance of the computing benchmark is measured using the memory bandwidth *per core* (hence higher is better).

2.2 Experimental environment

We ran our own benchmark suite[1] on several clusters with different characteristics: from small experimental clusters to large production ones. We present

[1] Available on https://gitlab.inria.fr/pswartva/memory-contention

here the results obtained on **henri** nodes which are dual Intel Xeon Gold 6140 at 2.3 GHz with 36 cores split across 4 NUMA nodes and 96 GB of RAM and equipped with InfiniBand ConnectX-4 EDR. **henri** nodes run Linux 5.7.7 with Debian 10, gcc 9.3. We show results obtained with MadMPI, the MPI interface of NewMadeleine [1] ; we observed similar results with other MPI implementations, such as OpenMPI 4.0.

3 First Observations

Our first hypothesis was that data accessed for computations and for communications may create contention on the memory bus, thus reducing performances. To test this hypothesis, we applied protocol described in previous section.

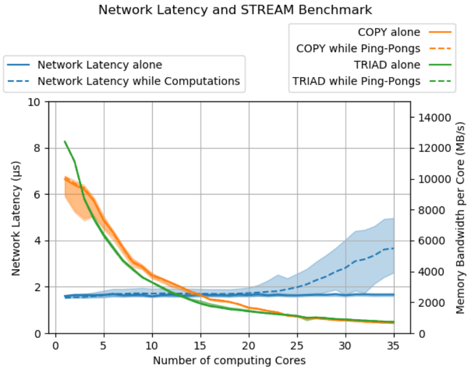

Fig. 1. Network latency (for 4 B) with memory-bound computations in parallel

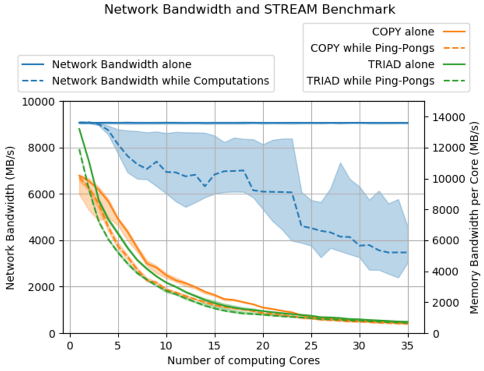

Fig. 2. Network bandwidth (for 64 MB) with memory-bound computations in parallel

Fig. 1 shows that network latency is impacted by the STREAM operations when there are at least 22 computing cores and this impact can double the regular latency when all available cores are computing. However, STREAM operations are not impacted by the ping-pong benchmark. The network bandwidth is impacted sooner, from 3 computing cores (Fig. 2). When all available cores are computing, the network bandwidth is reduced by almost two third from the regular network bandwidth. Memory bandwidth measured by the STREAM benchmark is lower when network bandwidth is measured at the same time as when network latency is measured, which is expected because one bandwidth ping-pong transfers more data than a latency ping-pong (64 MB *vs* 4 B).

4 Conclusion

Overlapping communications with computations is a well-known technique to increase performance of applications. However, our first experiments report memory contention caused by data used for communications and for computations can reduce performances of communications and computations.

The rest of our study explores parameters impacting interferences between communications and computations such as the placement of data and threads, the size of the data transmitted through the network and the arithmetic intensity of instructions executed by computing threads. We also measure the effect of the variations of processor frequencies and the effect of using a task-based runtime system.

As future works, we would like to better understand origins of these interferences to predict and quantify them. We would also like to measure the impact of data movements between main memory and GPUs. This knowledge could then be used in runtime systems to better predict performances and thus better schedules tasks to minimize the impacts of the interferences between computations and communications.

References

1. Aumage, O., Brunet, E., Furmento, N., Namyst, R.: NewMadeleine: a Fast Communication Scheduling Engine for High Performance Networks. In: Workshop on Communication Architecture for Clusters (CAC 2007). Long Beach, California, United States (Mar 2007), https://hal.inria.fr/inria-00127356
2. Groves, T., Grant, R.E., Arnold, D.: NiMC: Characterizing and Eliminating Network-Induced Memory Contention. In: 2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS). pp. 253–262 (2016). https://doi.org/10.1109/IPDPS.2016.29
3. Langguth, J., Cai, X., Sourouri, M.: Memory Bandwidth Contention: Communication vs Computation Tradeoffs in Supercomputers with Multicore Architectures. In: 2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS). pp. 497–506 (2018). https://doi.org/10.1109/PADSW.2018.8644601
4. McCalpin, J.: Memory bandwidth and machine balance in high performance computers. IEEE Technical Committee on Computer Architecture Newsletter pp. 19–25 (12 1995)