Adaptive MPI Multirail Tuning for Non-Uniform Input/Output Access

Stéphanie Moreaud, Brice Goglin, Raymond Namyst

Adaptive MPI Multirail Tuning for Non-Uniform Input/Output Access

Stéphanie Moreaud, Brice Goglin and Raymond Namyst

Université de Bordeaux - INRIA - LaBRI
351, cours de la Libération F-33405 Talence cedex.
{smoreaud,goglin,namyst}@labri.fr

Abstract. Multicore processors have not only reintroduced Non-Uniform Memory Access (NUMA) architectures in nowadays parallel computers, but they are also responsible for non-uniform access times with respect to Input/Output devices (NUIOA). In clusters of multicore machines equipped with several Network Interfaces, performance of communication between processes thus depends on which cores these processes are scheduled on, and on their distance to the Network Interface Cards involved. We propose a technique allowing multirail communication between processes to carefully distribute data among the network interfaces so as to counterbalance NUIOA effects. We demonstrate the relevance of our approach by evaluating its implementation within OpenMPI on a Myri-10G + InfiniBand cluster.

Keywords : *Multirail*, *Non-Uniform I/O Access*, *Hardware Locality*, *Adaptive Tuning*, *OpenMPI*.

1 Introduction

Multicore processors are widely used in high-performance computing. This architecture trend is increasing the complexity of the compute nodes while introducing non-uniform memory topologies. A careful combined placement of tasks and data depending on their affinities is now required so as to exploit the quintessence of modern machines. Furthermore, access to the networking hardware also becomes non-uniform since interface cards may be closer to some processors. This phenomenon has been known to impact networking performance for a long time [1] but current MPI implementations do not take it into account in their communication strategies.

One way to take affinities between processes and communication into account is to modify the process placement strategy so as to offer a privileged networking access to communication intensive processes. In this article, we look at an orthogonal idea: optimizing communication in a predefined process placement. We study multirail configuration and show that MPI implementation should not blindly split messages in halves when sending over multiple rails. *Non-Uniform I/O Access* should be involved in this splitting strategy so as to improve the overall performance.

The remaining of the paper is organized as follows. Section 2 presents modern architectures and the affinities of NICs. Our proposal and implementation of a NUIOA-aware multirail MPI is then described in Section 3 while its performance is presented in Section 4.

2 Background and Motivations

In this section, we describe the architecture of modern cluster nodes and introduce *Non-Uniform Input/Output Access* as a consequence that must definitely be taken into account in the design of communication algorithms and strategies.

2.1 Multicore and NUMA architectures

Multicore processors have represented over 90 % of the 500 most powerful computing systems in the world[1] for the last five years. This hardware trend is currently leading to an increasing share of NUMA architectures (*Non-Uniform Memory Access*), the vast majority of recent cluster installations relying on Intel Nehalem or AMD Opteron processors that have introduced scalable but non-uniform memory interconnects. AMD HyperTransport and, more recently, Intel QPI were designed towards this goal by attaching a memory node to each processor socket, as depicted on Figure 3. These increased complexity and hierarchical aspects, from multiple hardware threads, cores, shared caches to distributed memory banks, raise the need for carefully placing tasks and data according to their affinities. Once tasks are distributed among all the cores, an additional step is to optimize communication and synchronization between tasks depending on their topological distance within the machine [2].

In addition, the increasing number of cores in machines causes network interfaces and I/O buses to become potential bottlenecks. Indeed, concurrent requests from all cores may lead to contention and may thus reduce the overall application performance significantly [3]. Regarding this problem, multirail machines are now commonly considered as a workaround since their multiple NICs scale better with the number of cores. However, such complex architectures, interconnecting numerous hardware components, also raise the need to take affinities and locality into account when scheduling network processing [4].

2.2 Non-Uniform Input/Output Access

NUMA architectures have been the target of numerous research projects in the context of high-performance computing, from affinity-based OpenMP thread scheduling to MPI process placement [5, 6]. The impact of process placement in NUMA machines on high-speed networking has been known for several years already. As shown on Figure 3 network interfaces may be closer to some NUMA nodes and processors than to the others, causing their data transfer performance to vary. However, this phenomenon is almost only taken into account for microbenchmarks by binding processes as close as possible to the network interface.

As depicted by Figure 1, we demonstrated with previous Opteron architectures and several network technologies that the actual throughput is dramatically related to process placement with regards to network interfaces [1]. While latency depends only slightly on process placement (usually less than 100 nanoseconds), we observed up to 40% throughput degradation when using multiple high-performance interconnects. This behavior, called *Non-Uniform Input/Output Access* (NUIOA), induces the need for careful placement of tasks depending on their communication intensiveness.

[1] Top500, http://www.top500.org.

The phenomenon has been observed for various memory interconnects. It still appears with latest Intel Nehalem processors and QPI architectures. It is sometimes also referred to as *Non-Uniform Network Access* but it is actually not specific to network devices. Indeed, we observed DMA throughput degradation by up to 42% when accessing a NVIDIA GPU from the distant NUMA node of a dual-Xeon 5550 machine. Moreover, in the presence of multiple devices, it becomes important to carefully distribute the workload among devices. Since these NUMA architectures are now spreading into high-performance computing, we intend to look at adapting the MPI implementation to these new constraints.

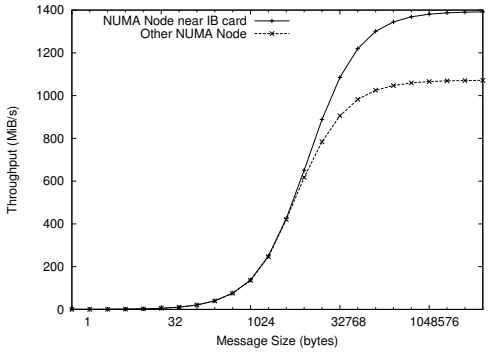

Fig. 1. Influence of the locality of processes and network cards on the RDMA Write throughput between 2 dual-Opteron 265 machines with InfiniBand DDR cards.

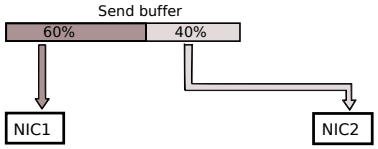

Fig. 2. Multirail using a 40% splitting ratio. 100% means that the whole message is sent through the NIC1.

3 NUIOA-aware Multirail

We introduce in this section our proposal towards a MPI implementation that adapts multirail communication to Non-Uniform I/O Access. We then describe how we implemented it in Open MPI.

3.1 Proposal

Dealing with affinities inside NUMA machines usually requires to place tasks with intensive inter-communication or synchronization inside the same NUMA node or shared cache. Meanwhile, distributing the workload across the whole machine increases the available processing power and memory bandwidth. Finding a tradeoff between these goals is difficult and depends on the application requirements. Adding the locality of network interfaces to the problem brings newer constraints since some cores may have no I/O devices near them. It leads to the idea of keeping these cores for tasks that are not communication intensive. Other processes may be given a privileged access to all or only some of the interfaces (they may be close to different cores). Moreover, detecting which tasks are communication-intensive may be difficult. And numerous MPI applications have uniform communication patterns since most developers try to avoid irregular parallelism so as to exploit all the processing cores.

While binding communication-intensive tasks near the network interfaces is not easy, we look at an orthogonal problem: to optimize the implementation of

communication within a predefined process placement. This placement may have been chosen by the MPI process launcher depending on other requirements such as affinities between tasks [6]. Given a fixed distribution of processes on a NUIOA architecture, we propose to adapt the implementation of MPI communication primitives to better exploit multiple network interfaces.

3.2 Distributing Message Chunks according to NICs localities

Several MPI implementations may use multiple network interfaces at the same time. For throughput reasons, large messages are usually split across all available *rails* and reassembled on the receiver side. Open MPI [7] and MPICH2/NewMad [8] may even use different models of interfaces and wires and dynamically adapt their utilization depending on their relative performance. For instance, a 3 MiB message would be sent as a 1 MiB chunk on a DDR InfiniBand link and another 2 MiB chunk on a QDR link. As explained above, the actual throughput of these network rails depends on the process location. We thus propose to adapt the size of the chunks to the distance of each process from network interfaces.

We implemented this idea in Open MPI 1.4.1. Each network interface is managed by a BTL component (*Byte Transfer Layer*) that gathers its expected bandwidth by looking at its model and current configuration. By default, these bandwidths are accumulated in the BML (*BTL Multiplexing Layer*) so as to compute a *weight* for each BTL. Sending a large message then results in one chunk per BTL that connects the processes, and a *splitting ratio* is determined so that each chunk size is proportional to the BTL weight, as depicted on Figure 2.

We modified the R2 BML component so as to take hardware locality into account when computing these weights. The expected bandwidth of each BTL is adjusted by looking at the current process and BTL physical device locations. This change is specific to each process since it depends on its actual binding. It must thus take place late in the initialization phase, usually after the `paffinity` component did the actual binding of processes.

3.3 Gathering NIC and process locality information

Our NUIOA-aware tuning of BTL weights relies on the knowledge of the location of each networking device in the underlying hardware topology. Most BIOSes tell the operating system which NUMA node is close to each I/O bus. It is thus easy to determine the affinity of each PCI device. However, user-level processes do not manipulate PCI devices, they only know about software handles such as InfiniBand devices in the Verbs interface. Some low-level drivers offer a way to derive these software handles into hardware devices thanks to `sysfs` special files under Linux. For other drivers such as Myricom MX, a dedicated command had to be added to retrieve the locality of a given MX endpoint.

We implemented in the hwloc library (*Hardware Locality*[2]) the ability to directly return the set of cores near a given InfiniBand software device. Open MPI will switch to using hwloc for process binding in the near future since it offers an extensive set of features for high-performance computing [9]. The BML will thus easily know where Open MPI bound each process. Until this is implemented, we let the `paffinity` component bind processes and later have the BML

[2] http://www.open-mpi.org/projects/hwloc/

retrieve both process and network device locations through HWLOC. Once this information has been gathered, the BML adjusts the splitting ratio according to the actual performance of each BTL in this NUIOA placement.

4 Performance

We present in this section the performance evaluation of NUIOA effects, from single rail ping-pong to multi-rail MPI collective operations.

4.1 Experimentation Platform

The experimentation platform is composed of several quad-socket hosts with dual-core OPTERON 8218 processors (2.6 GHz). As depicted by Figure 3, this NUMA architecture contains four NUMA nodes, two of them being also connected to their own I/O bus. Each bus has a PCIe 8x slot where we plug either a MYRICOM MYRI-10G NIC or a MELLANOX MT25418 CONNECT-X DDR INFINIBAND card. These hosts run the *Intel MPI Benchmarks* (IMB) on top of our modified OPEN MPI 1.4.1 implementation, using the MX or OPENIB BTLs.

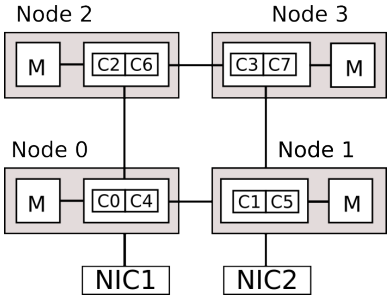

Fig. 3. Quad-socket dual-core host with two I/O chipsets connected to NUMA nodes #0 and #1.

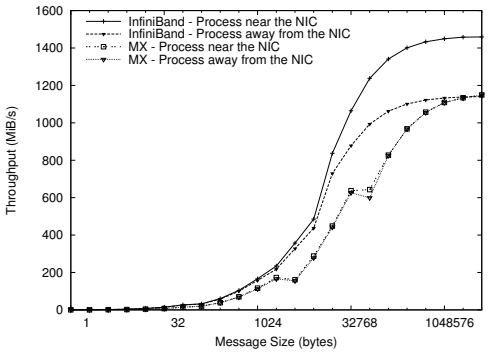

Fig. 4. Influence of the locality of processes and network cards on a single-rail IMB ping-pong with OPEN MPI.

4.2 Single-rail micro-benchmark

Figure 4 summarizes NUIOA effects on our experimentation platform by presenting the MPI throughput of a ping-pong between two hosts depending on the process binding. It confirms that NUIOA effects are indeed significant on our platform, whenever messages contain dozens of kilobytes.

However the actual impact depends a lot on the underlying networking hardware. Indeed, the throughput over MX with MYRI-10G cards varies only very slightly while the INFINIBAND throughput decreases by 23% when the process is not bound near the network interface. The raw INFINIBAND throughput being larger, it may be more subject to contention, but we do not feel that this fact would induce such a difference. Instead we think that these NICs may be using different DMA strategies to transfer data inside the host, causing the congestion to differ. Indeed, if INFINIBAND uses smaller DMA packets, more packets are in-flight at the same time on the HYPERTRANSPORT bus, causing more saturation of the HYPERTRANSPORT request and response buffers.

Moreover, we also observed that increasing the NUMA distance further does not further decrease the InfiniBand throughput: once the process is not near the card, the throughput does not vary anymore when binding it farther away.

4.3 Point-to-point multirail

Figure 5 now presents the multirail throughput for large messages depending on process placement when an InfiniBand NIC is connected to each I/O bus. It clearly shows that when the process is bound near one of the cards (NUMA node #0 or #1), the MPI implementation should privilege this card by assigning about 58% of the message size to it. This ratio is actually very close to the ratio between mono-rail throughputs that we may derive from Figure 4 (56.6%). Such a tuning offers 15% better throughput than the usual half-size splitting. Moreover, when the process is not close on any NIC (NUMA node #2 or #3), the messages should be split in half-size chunks as usual. Again, this could have been derived from Figure 4 since mono-rail throughput does not vary from with the binding when not close to the NIC.

When Myri-10G NICs are used, the closest NIC should only be very slightly privileged (51%). It is also expected since Myri-10G performance varies only slightly with process placement. Finally, combining one Myri-10G and one InfiniBand NIC also matches our expectations: when a process is close to the InfiniBand NIC, it should privilege it significantly (57%), while a process near the Myri-10G NIC should only privilege it slightly (51%).

These results confirm that combining mono-rail throughputs from the given process bindings is an interesting way to approximate the optimal multirail ratio for point-to-point operations, as suggested in earlier works [10].

4.4 Contention

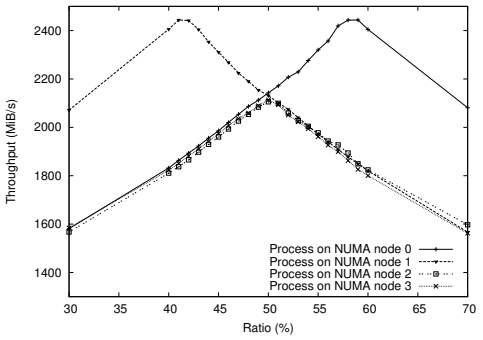

Fig. 5. Multirail IMB ping-pong throughput for 1 MiB messages, depending on the ratio between two InfiniBand cards and on the process placement.

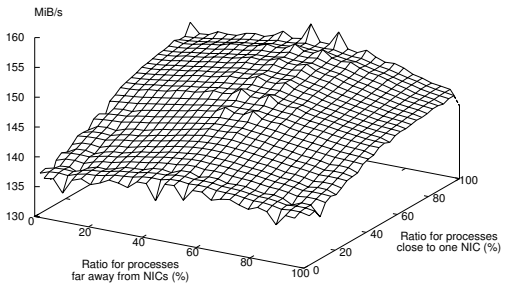

Fig. 6. IMB all-to-all, throughput per process between 16 processes on 2 hosts with 2 InfiniBand NICs, depending on multirail splitting ratios.

We now look at the impact of contention on the memory bus on our NUIOA-aware multi-rail. Indeed, previous results used idle hosts where data transfers between NUMA nodes and the NICs were optimal. In the case of real applications, some processors may access each others' memory, causing contention on the HyperTransport links.

We added such contention on some HyperTransport links during our aforementioned multirail IMB ping-pong. This reduces the network throughput but does not modify the splitting ratio. This is a surprising result since we had carefully chosen which HyperTransport link should get contention so as to disturb the data path towards a single NIC and not the other (thanks to the HyperTransport routing table). If the ratio does not change, then it means that our contention on the single link reduces the overall memory bandwidth instead of only the bandwidth on this link.

4.5 Collective operations

Previous sections showed that an interesting splitting of large point-to-point messages across multiple rails may be derived from each rail NUIOA throughput. We now look at collective MPI operations. Since processes are now communicating from all NUMA nodes at the same time, we should now find the splitting ratio of each running process simultaneously. Figure 6 presents the performance of the all-to-all operation depending on the splitting ratios. We distinguish processes depending on whether they are close to one NIC or not.

The optimal tuning that we obtained first reveals that processes that are not close to any NIC should send one half of each message on each NIC. This result matches our earlier point-to-point observations since NUIOA effects do not matter once processes are far from NICs, but it seems less significant here. Then, the interesting result is that processes close to one NIC should only use this NIC. This result contradicts the previous section since contention now appears as critical for performance. We assume that contention in this all-to-all benchmark were more intensive than in the previous section, causing the ratio to vary. In the end, this all-to-all tuning outperforms the default splitting strategy by 5%.

When using one InfiniBand and one Myri-10G NIC, we obtained similar results. However, when using two Myri-10G NICs we again observed less NUIOA effects since the ratio almost does not matter (the variation of results among multiple runs is larger than the variation due to splitting ratios). Looking at other collectives, we observed that communication intensive operations tend towards all-to-all ratios while other operations behave similarly to point-to-point results.

5 Conclusion and Future Works

The increasing number of cores and the widespread use of NUMA architectures in cluster computing nodes leads to the multiplication of connected hardware components. It raises the need to take affinities and localities into account in the design of communication strategies. Indeed, the performance of communication over high speed networks is directly related to the relative location of processes and network interfaces.

In this paper, we propose to optimize the implementation of MPI primitives by adapting the use of multiple network interfaces to their locations with regards to processes. Thanks to the knowledge of process/NIC affinities in hwloc, we determine a splitting ratio that increases the throughput by up to 15% over the standard multirail strategy. Communication-intensive patterns such as all-to-all even show that processes that are close to one NIC should not use other NIC so as to avoid contention on the memory bus.

Combined with per-core sampling of the interfaces, or even auto-tuning [11], this approach lets us envision a far better utilization of multiple rails. We plan to integrate our work in the mainline Open MPI implementation and further experiment with it on real applications. We also intend to integrate this knowledge about NIC affinities in collective algorithms, where the role of each process may differ (e.g. local root of a reduction) and thus where each process would certainly use different thresholds.

References

1. Moreaud, S., Goglin, B.: Impact of NUMA Effects on High-Speed Networking with Multi-Opteron Machines. In: The 19th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2007), Cambridge, Massachussetts (2007)
2. Buntinas, D., Goglin, B., Goodell, D., Mercier, G., Moreaud, S.: Cache-Efficient, Intranode Large-Message MPI Communication with MPICH2-Nemesis. In: Proceedings of the 38th International Conference on Parallel Processing (ICPP-2009), Vienna, Austria, IEEE Computer Society Press (2009) 462–469
3. Narayanaswamy, G., Balaji, P., Feng, W.: Impact of Network Sharing in Multi-core Architectures. In: Proceedings of the IEEE International Conference on Computer Communication and Networks (ICCCN), St. Thomas, U.S. Virgin Islands (2008)
4. Jang, H.C., Jin, H.W.: MiAMI: Multi-core Aware Processor Affinity for TCP/IP over Multiple Network Interfaces. In: Proceedings of the 17th Annual Symposium on High-Performance Interconnects (HotI'09), New York, NJ (2009) 73–82
5. Rabenseifner, R., Hager, G., Jost, G.: Hybrid MPI/OpenMP Parallel Programming on Clusters of Multi-Core SMP Nodes. In: Proceedings of the 17th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP 2009), Weimar, Germany (2009) 427–436
6. Mercier, G., Clet-Ortega, J.: Towards an Efficient Process Placement Policy for MPI Applications in Multicore Environments. In: EuroPVM/MPI. Volume 5759 of Lecture Notes in Computer Science., Espoo, Finland, Springer (2009) 104–115
7. Gabriel, E., Fagg, G.E., Bosilca, G., Angskun, T., Dongarra, J.J., Squyres, J.M., Sahay, V., Kambadur, P., Barrett, B., Lumsdaine, A., Castain, R.H., Daniel, D.J., Graham, R.L., Woodall, T.S.: Open MPI: Goals, concept, and design of a next generation MPI implementation. In: Proceedings, 11th European PVM/MPI Users' Group Meeting, Budapest, Hungary (2004) 97–104
8. Mercier, G., Trahay, F., Buntinas, D., Brunet, É.: NewMadeleine: An Efficient Support for High-Performance Networks in MPICH2. In: Proceedings of 23rd IEEE International Parallel and Distributed Processing Symposium (IPDPS'09), Rome, Italy, IEEE Computer Society Press (2009)
9. Broquedis, F., Clet-Ortega, J., Moreaud, S., Furmento, N., Goglin, B., Mercier, G., Thibault, S., Namyst, R.: hwloc: a Generic Framework for Managing Hardware Affinities in HPC Applications. In: Proceedings of the 18th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP2010), Pisa, Italia, IEEE Computer Society Press (2010)
10. Aumage, O., Brunet, E., Mercier, G., Namyst, R.: High-Performance Multi-Rail Support with the NewMadeleine Communication Library. In: Proceedings of the Sixteenth International Heterogeneity in Computing Workshop (HCW 2007), held in conjunction with IPDPS 2007, Long Beach, CA (2007)
11. Pellegrini, S., Wang, J., Fahringer, T., Moritsch, H.: Optimizing MPI Runtime Parameter Settings by Using Machine Learning. In: EuroPVM/MPI. Volume 5759 of Lecture Notes in Computer Science., Espoo, Finland, Springer (2009) 196–206