# Analyzing Performance Improvements and Energy Savings in Infiniband Architecture using Network Compression

Branimir Dickov*†, Miquel Pericàs‡, Paul M. Carpenter*, Nacho Navarro*†, Eduard Ayguadé*†
*Barcelona Supercomputing Center
†Universitat Politècnica de Catalunya, BarcelonaTech
‡ Chalmers University of Technology
{branimir.dickov, paul.carpenter, nacho.navarro, eduard.ayguade}@bsc.es
miquelp@chalmers.se

***Abstract*—One of the greatest challenges in HPC is total system power and energy consumption. Whereas HPC interconnects have traditionally been designed with a focus on bandwidth and latency, there is an increasing interest in minimising the interconnect's energy consumption. This paper complements ongoing efforts related to power reduction and energy proportionality, by investigating the potential benefits from MPI data compression.**

**We apply lossy compression to two common communication patterns in HPC kernels, in conjunction with recently introduced InfiniBand (IB) power saving modes. The results for the Alya CG kernel and Gromacs PME solver kernels show improvements in both performance and energy. While performance improvements are strongly influenced and changable depending on the type of communication pattern, energy savings in IB links are more consistent and proportional to achievable compression rates. We estimated an upper bound for link energy savings of up to 71% for the ALYA CG kernel, while for the Gromacs PME solver we obtained savings up to 63% compared to nominal energy when compression rate of 50% is used. We conclude that lossy compression is not always useful for performance improvements, but that it does reduce average IB link energy consumption.**

***Keywords*-Parallel Applications; Data Compression; Supercomputing; MPI Performance; Network Energy Savings**

## I. Introduction

High-Performance Computing (HPC) is increasingly constrained by total system peak power and energy consumption. Cumulative improvements in micro-architecture and silicon process technologies have dramatically improved the energy efficiency of the processing nodes, leading to an interest in the energy efficiency of the interconnect. One way to reduce the energy consumption of any resource is through *energy proportionality*, meaning that its energy consumption is proportional to its utilization. Current interconnects are not energy proportional, since high-speed links consume the same power, whether or not data is actively being exchanged. Network vendors are, however, beginning to implement basic power saving techniques. Examples include powering down interconnection links and entire switches during idle periods [1].

Performance and scalability are, however still first-class requirements in HPC, so they should not be compromised by any proposed energy saving technique. We therefore have two goals: maximising energy efficiency and minimising performance overheads. Our focus is on InfiniBand networks [2], which account for 45% of the systems in the TOP500 [3] and most non-custom interconnects in the top one hundred systems. We focus on the energy consumed by the interconnection links, which consume the majority of the network power; e.g. for an IBM InfiniBand 8-port 12X switch, the links are estimated to take 64% of the total switch power [4].

This paper complements ongoing efforts in interconnect energy proportionality, by investigating the energy savings from data compression. Significant prior work has focused on the use of data compression to improve performance. We tried to reproduce the large improvements in performance found by previous studies, but when we applied compression to real application kernels on realistic modern high-performance networks, the performance gains were low. As discussed in the related work section, previous studies generally used benchmarks or low-performance networks. To the best of our knowledge, this paper is the first to analyse the role of data compression in reducing network energy consumption. We found that lower interconnect utilisation resulting from data compression led to significant energy savings.

We apply compression to the data sent and received by MPI [5] library calls. Our techniques are designed for double-precision floating-point (FP) data, which is the most common data type passed in the messages of scientific applications. Specifically, our techniques implement lossy compression by truncating the mantissa. Previous work has shown that certain applications are able to tolerate this loss of precision [6]. To isolate the energy savings from data compression while preserving performance, the link bandwidth is effectively scaled with the compression ratio, by changing the number of active InfiniBand link lanes. Switching between different power states of the links can be implemented as macros directly in MPI code [7].

We chose two parallel scientific applications, Alya [8] and Gromacs [9] , both of which use the Single Program Multiple Data (SPMD) programming paradigm, in which application processes have synchronised computation and communication parts. These applications represent the most commonly used communication patterns in HPC, nearest-neighbor and all-to-all exchange. For Alya, the main kernel to be exercised is Conjugate Gradient [10]. Data compression is applied to transfers that occur during the Sparse Matrix Vector Multiplication,

which has a nearest-neighbor communication pattern. In the case of Gromacs, we applied data compression to transfers in the 3D FFT part of the Particle Mesh Ewald (PME) [11] computation, which has an all-to-all exchange pattern.

The energy results fully justified the use of data compression for system energy efficiency. We estimated link energy savings of up to 71% for Alya CG kernel, while for Gromacs PME we got savings of up to 63% compared to nominal energy, with 50% compression. In contrast with previous work, however, we found the performance improvement from data compression to be small. The blocking nature of the point-to-point MPI calls in the nearest-neighbour pattern of Alya CG led to a maximum speedup of just 2.3%. However, for the Gromacs PME kernel with an all-to-all exchange pattern, we obtained somewhat better result with a maximum speedup of 14%.

The rest of the paper is organized as follows. In Section II, we provide the necessary background in interconnect power consumption and link power optimization techniques. Section III introduces the proposed technique. Section IV describes the evaluation methodology. In Section V, we evaluate the performance gains, and in Section VI, we provide the IB link energy saving analysis. Section VII discusses the related work. We conclude this work in Section VIII.

## II. Background

### A. Data Compression

Data compression, both lossless and lossy, is widely used, across many application domains, to reduce the demands on storage capacity and communication bandwidth. In HPC, data compression has been applied to messages on the network in order to minimise application execution time [12], [13], [14], [15], [16]. This paper is the first, as discussed above, to apply data compression to the problem of energy efficiency.

There is an important distinction between lossless and lossy compression. Although lossless compression can be applied to scientific applications indiscriminantly, always leading to correct results, previous studies have shown that the resulting compression rate is low [16], [6]. This is because scientific applications pass high entropy floating-point data, which is hard to compress. In contrast, applying lossy compression requires the involvement of the scientific application or library developer. Previous studies have, however, found that these experts have a good understanding of the numerical stability of their algorithms, and can determine the required data precision. For the applications chosen in this work, a previous study concluded that no significant divergence occurs in the final results when using 50% lossy compression [6].

### B. Interconnection Network Power Consumption

The network fabric power consumption is due to the switch fabric, Host Channel Adapters (HCAs), and interconnect links. The power consumption of the HCAs and switches varies with the data injection rate, being dominated by the active power of the memory elements. In contrast, the power consumption of an interconnect link is almost constant, whether or not it is actively exchanging data [17], since both ends stay active to maintain synchronization. It has been shown that majority of the total network fabric power consumption is due to interconnect links [4]. For this reason, we focus on link power.

InfiniBand links support two physical mechanisms that can be used for power saving. Firstly, the link's signalling rate can be changed, for example between Quad Data Rate (QDR), at 10 Gb/s per serial lane, and Single Data Rate (SDR), at 2.5 Gb/s per lane. This gives a tradeoff between bandwidth and energy consumption. The switching time between rates is small, on the order of hundreds of nanoseconds [18], but the power savings are also small.

A single InfiniBand link is implemented by aggregating one or more serial lanes. It is common to use more than one lane, in order to increase the link's bandwidth, though power consumption is also multiplied by the same factor. Mellanox has recently announced Width Reduction Power Saving (WRPS), a technique that allows the link's number of active lanes to be dynamically reconfigured, assuming firmware support is enabled in the HCAs and switches [1]. The energy to transfer a single message remains the same, because energy is power multiplied by time. The energy savings come from relatively short idle periods between messages.

After a broad investigation, we found that the worst case power consumption is 0.3 W for the transmitter and 0.2 W for the receiver [17]. In our investigation we therefore assumed that every lane consumes 0.5 W. For a link with four lanes this would be 2 W per port.

## III. Proposal: Data Compression for Energy Savings

### A. Implementing Data Compression

Data compression and decompression often have high computational costs. Since our lossy compression scheme is simply truncation of least-significant bits (LSB) of the floating-point mantissa, it is, however, relatively inexpensive. If implemented naively in software its cost is still similar to two additional copies per message transfer (one on send and one on receive). In some cases these software copies could be merged with existing data copies. Since each extra message copy is still a significant overhead, we propose to implement truncation in streaming hardware [19], [20], while the data is copied from main memory to the memory on the Host Channel Adapter (HCA). When message words arrive at the HCA, they are stored in an input buffer; the compression rate is chosen by activating the appropriate bit lines. On the receiving side, decompression is done by padding with zeros to restore to the original double-precision format. Since compression is done in the HCA, there is no data compression on messages between MPI processes mapped to the same node.

### B. Power switching

In order to benefit from the potential energy savings, we used the following policy. When there is no communication, all lanes except one are switched off. If all lanes of a link were shut down, the forwarding tables would have to be updated. By having one lane remain on, management and control traffic will be always available. Also by having one lane always on, we avoid the need for complex adaptive routing schemes. When communication is about to happen, the appropriate number of lanes are powered up, depending

on the compression rate. For example, if compression is not being used, then all lanes (four, in our experiments) would be powered up. Alternatively, if 50% compression is applied, half of the lanes (two lanes) are powered up, preserving the original performance. On 4× links, the supported compression rates are therefore 50% and 25%. On 8× links, the granularity is finer, giving a wider range of possible compression rates. It is expected that the number of lanes will continue to rise. Thus, the idea is that we can have more tuned and optimized support for power savings supported through compression techniques.

To be sure that links are active when needed, we use two new MPI primitives, one to activate the appropriate links and one to deactivate unused links [7]. Each MPI primitive can use WRPS method to tune the links to required width. For turning on/off lanes we take a typical delay of up to 10 microseconds ($\tau$ = 10 µs) [21]. By recognizing the communication patterns or group of patterns (regions) in the parallel application, the unnecessary overheads that can appear by power switching can be avoided. This allows changes in link bandwidth (also link power consumption) to be done at coarser granularity than individual communication calls. Therefore, we apply the following policy regarding the MPI primitives for activation and deactivation:

- Activate the link to the required number of lanes before each MPI call and deactivate after the MPI call has finished.
- If the MPI call is part of larger loop where more communication exchanges are done (nearest-neighbour pattern), activate the link before the loop and deactivate when all MPI calls in the loop finish.

## IV. Methodology

To measure the impact of data compression on application execution time and quantify the link energy savings, we use the Venus–Dimemas simulator [22], [23]. Dimemas is an event-driven simulator, which replays a trace of the application's computation bursts and MPI activity, preserving causal relationships and timings. Venus is a detailed network simulator, which models the complete network architecture including topology and routing, with an accurate switch/adapter model. Computation is not simulated in detail, instead being represented by the original duration from the trace.

Traces of the Alya and Gromacs applications were recorded on a machine based on Bull B505 nodes, with six-core Intel Xeon E5649 processors at 2.53 GHz with 24 GB RAM. In this architecture, allocating multiple MPI processes to the same processor implies sharing of various resources including the HCA. Data compression could alleviate contention on the HCA, especially if MPI processes communicate at similar times, which is often the case for scientific applications. In order to measure this effect, we generated traces in two configurations, first with one MPI process per processor, and second with six MPI processes per processor (one per core). Also, we generated strong scaling traces, so that as the number of processes was varied, the workload remained the same.

Table I gives the parameters of the simulated system. We first ran the simulation without any changes in the trace, replicating the original execution times. For the performance analysis in Section V, we reduced the sizes of the MPI messages in accordance with the compression rate. Using lossy compression the message size was therefore reduced by 25%, 37.5% or 50%, equivalent to compressing the double-precision FP data by truncating the 16, 24 or 32 least-significant bits. Since this is done in hardware, only a few additional cycles would be needed, which can be considered negligible, validating the assumption of zero overhead for hardware lossy compression. When multiple MPI processes are mapped to one node, communication inside the same node is done without compression. Finally, we simulate the new traces on Venus–Dimemas, and quantify the performance.

TABLE I
Parameters used in Simulations

| Parameter | Value |
|---|---|
| Simulator | Dimemas–Venus |
| Connectivity | XGFT(2;18,14;1,18) |
| Topology | 2-level Extended Generalized Fat Tree |
| Switch technology | InfiniBand |
| Network Bandwidth | 40 Gbits/s |
| Segment Size | 2 KB |
| MPI latency | 1$\mu$s |
| CPU Speedup | 1 |
| Routing scheme | Random routing |
| 4X-IB link port power | 2W peak |

For the energy analysis in Section VI, we assumed that the link bandwidth was always reduced in accordance with the compression rate. Rather than implementing new code in the simulator to dynamically modify the link bandwidths in proportion to the new message sizes, the same effect was achieved simply by keeping the original MPI message sizes and interconnect bandwidths. It was only necessary to model the switching overheads by introducing appropriate delays in the traces.

## V. Performance Analysis

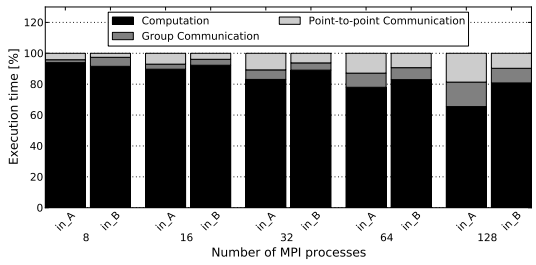
### A. Case Study: Alya-CG kernel

We first investigate the performance benefits of lossy compression for the communications in Alya [8]. Specifically, we applied compression to the Sparse Matrix–Vector Multiplication (SMVM) kernel in the Conjugate Gradient (CG) algorithm. We use two input sets, a small one denoted Input A and a large one denoted Input B.

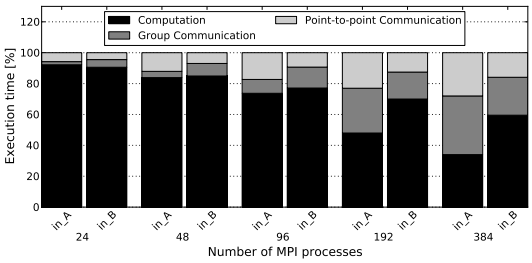
*1) Performance using lossy compression with one MPI process per node:* Figure 1(a) gives execution profiles of the Alya CG kernel for the two input data sets. For this configuration, all communication is inter-node, so all MPI messages must pass through HCAs. Compression leads to smaller MPI messages, meaning that MPI messages arrive sooner. Table II gives the

TABLE II
Average size of MPI messages(kB) with 1 MPI process per node

| Number processes | Alya CG Input A | Alya CG Input B | Gromacs PME Input A | Gromacs PME Input B |
|---|---|---|---|---|
| 8 | 15.4 | 47.7 | 139.9 | 1036.0 |
| 16 | 7.9 | 25.8 | 147.8 | 259.0 |
| 32 | 3.9 | 17.4 | 55.4 | 350.4 |
| 64 | 2.2 | 9.2 | 18.4 | 135.5 |
| 128 | 1.4 | 5.0 | 7.9 | 53.2 |




(a) 1 MPI processes per node.

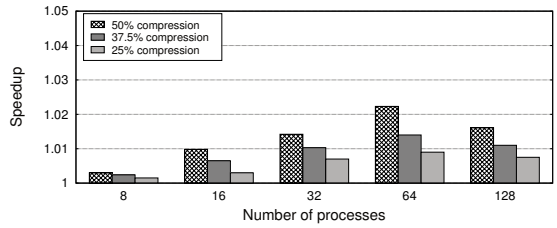



(b) 6 MPI processes per node.

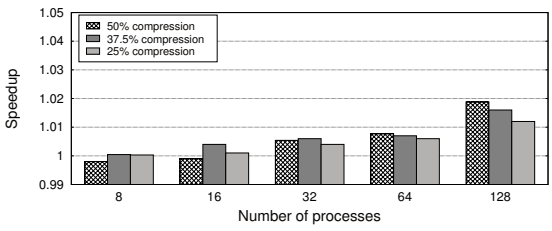
Fig. 1. Profile of the Alya CG kernel

average MPI message sizes, in kilobytes, for the two input sets varying the number of processes.

Figure 2 shows the speedup for the Alya CG kernel, for both input sets from eight to 128 processes. Speedup is measured in comparison with the corresponding baseline run without data compression. The general trend is that the speedup increases as the number of processes is increased, since the application becomes more communication-heavy, due to strong scaling. For Input A, the speedup factor slightly decreases for 128 MPI processes. This is because the size of MPI messages decreases making the application sensitive to latency rather than bandwidth. For Input A reduced precision in the computation does not result in an increase in the number of CG iterations until convergence, even at 50% compression when 32 LSB of mantissa are truncated. For Input B, reduced precision results in a slightly larger number of iterations, giving low performance improvement for runs with a small number of processes. For runs with 8 and 16 MPI processes,




(a) Input A.




(b) Input B.

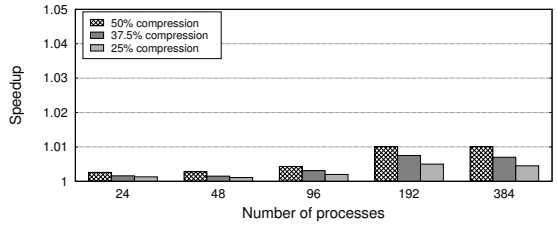
Fig. 2. Speedup factors from applying lossy compression on MPI messages in SMVM kernel of CG with 1 MPI processes per node

TABLE III
AVERAGE SIZE OF MPI MESSAGES(KB) WITH 6 MPI PROCESSES PER NODE

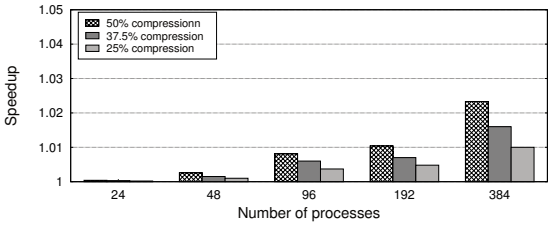
| N proc | Alya CG Input A | Alya CG Input B | Gromacs PME Input A | Gromacs PME Input B |
|---|---|---|---|---|
| 24 | 4.7 | 20.8 | 73.5 | 540.3 |
| 48 | 2.8 | 12.8 | 26.0 | 192.5 |
| 96 | 1.8 | 6.6 | 10.9 | 78.6 |
| 192 | 1.1 | 3.8 | 4.4 | 26.2 |
| 384 | 0.7 | 2.1 | 1.7 | 9.7 |

the performance benefits of 50% compression are insuffient to compensate for the greater number of iterations, giving a degradation in performance. For the lower compression rate of 37.5% there is still a small speedup.

*2) Performance using lossy compression with six MPI processes per node:* For runs with a small number of processes, most communication in the CG kernel is between processes on the same node. As the total number of processes increases, a greater proportion of messages is between processes on different nodes. Figure 1(b) shows the profiles of the CG kernel with six MPI processes per node. Table III gives the average sizes of the MPI messages.




(a) Input A.



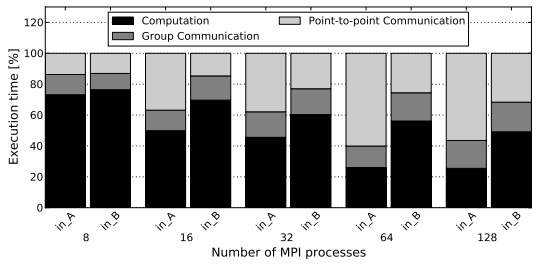

(b) Input B.

Fig. 3. Speedup factors from applying lossy compression on MPI messages in SMVM kernel of CG with 6 processes per node

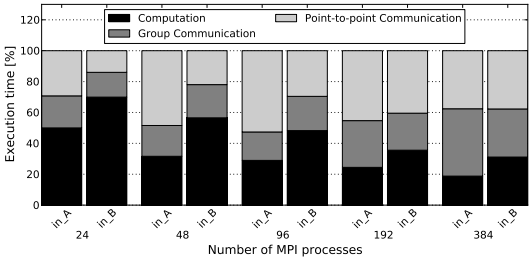
Since compression is only applied to data communicated between different nodes, multiple processes per node leads to smaller performance benefits from compression. Figure 3 shows the observed speedups. For Input A, as the inter-node to intra-node communication ratio increases, we observe larger speedup factors, especially for 192 and 384 MPI processes. Although the MPI messages become small, all inter-node communication is done via the HCAs, and the aggregate size of the MPI messages is significant. For Input B, we see that the speedup factors increase, but slowly. With 384 MPI processes, the largest performance improvement factor of just 2.3%.

### *B. Case Study: Gromacs-PME kernel*

The second case study is Gromacs [9]. Data compression is applied to the all-to-all exchange pattern in the Particle




(a) 1 MPI processes per node.




(b) 6 MPI processes per node.

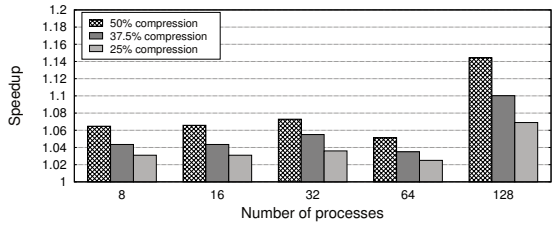
Fig. 4. Profile of Gromacs PME kernel

Mesh Ewald (PME) algorithm. The 3D FFT kernel in PME is notable for requiring dense communication, due to multiple MPI_Alltoall operations.

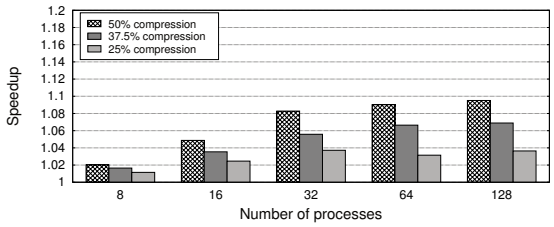
*1) Performance using lossy compression with one MPI process per node:* Figure 4(a) gives execution profiles for the PME kernel for the two input data sets. Figure 5 shows that speedup factors tend to increase with the number of MPI processes, but for Input A with 64 MPI processes there is a small decrease in the speedup. The reason is that for this particular workload, Gromacs uses different domain decompositions for 32 and 64 MPI processes. This results in an inconsistent execution profile for 64 MPI processes. For the run with 128 MPI processes, we observe the maximum improvement of a little over 14%. From Table II, we see that Gromacs has much larger messages than Alya, suggesting a greater performance benefit from data compression. In addition, MPI_Alltoall tends to synchronize the tasks, further increasing the performance benefits from compression.

*2) Performance using lossy compression with six MPI processes per node:* With six MPI processes per node, all cores are occupied. The all-to-all exchange pattern implemented in MPI_Alltoall uses non-blocking sends and receives, so each process sends data to all recipients before waiting to receive data. These non-blocking sends and receives can create contention on the HCAs. Figure 4(b) shows the profile of PME with six MPI processes per node. It is clear that a greater percentage of time is spent in communication, compared with one MPI process per node. We expect data compression to have a greater impact on application performance.

Figure 6 shows the speedup results, compared with the baseline with no compression. We see that for Input A, the peak speedup is for 192 processes; for 384 processes we get a lower speedup. For the largest run, the MPI messages become small ($\approx$ 1 KB) and the application becomes latency sensitive, giving a lower performance benefit. Since Input B is larger, the improvements remain consistent, even with 384 processes, giving a performance improvement of 12%.

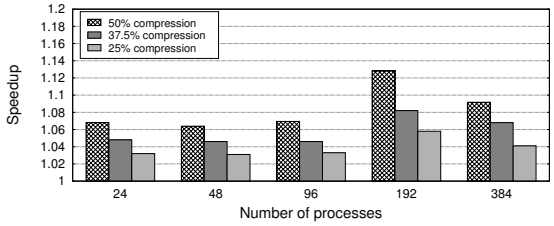



(a) Input A.

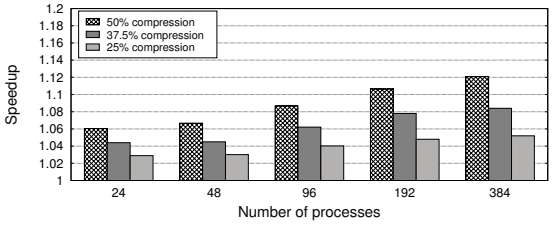



(b) Input B.

Fig. 5. Speedup factors from applying lossy compression on MPI messages in the 3D FFT part of PME with one MPI process per node




(a) Input A.



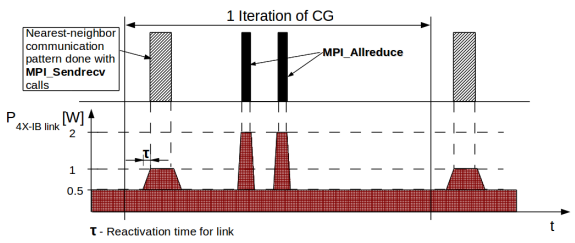

(b) Input B.

Fig. 6. Speedup factors from applying lossy compression on MPI messages in the 3D FFT part of the PME kernel with six MPI processes per node

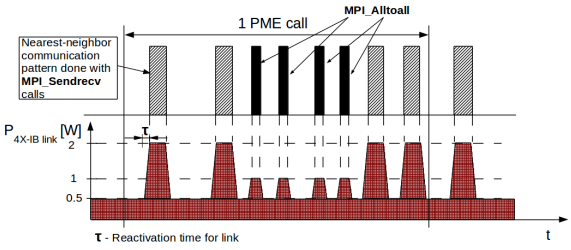
## VI. Analysis of Link Energy Savings

The previous section investigated the effect of data compression on application performance, finding little performance benefit for real application kernels on modern high-performance networks. We now turn to the problem of energy efficiency. Previous work on network energy proportionality has focussed on powering down communication links when idle. In this section, we dynamically adjust the link bandwidth, by varying the number of active lanes during communication periods. During idle periods, one lane remains active. At the start of each communication, the number of lanes is increased to two, three, or four, depending on the compression rate. In any case, there is a reactivation penalty of 10 µs.

Figure 7 illustrates the behaviour of the Alya CG and Gromacs PME kernels. The upper traces show whether execution is in the application (no bar) or MPI library (grey or black bar). The lower part of each subfigure shows the link power consumption, using the proposed link power reduction technique,




(a) Alya CG kernel.

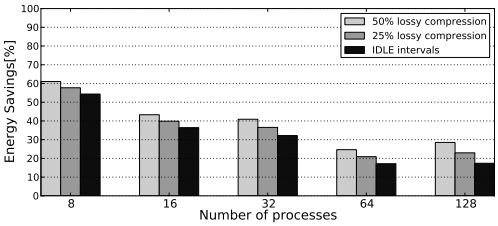



(b) Gromacs PME kernel.

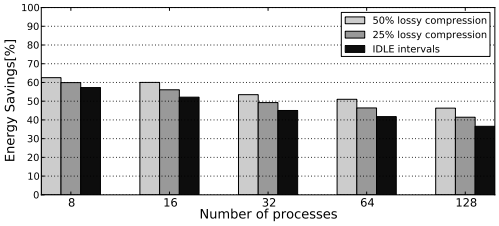
Fig. 7. Traces showing whether the program is in the application or MPI library (grey or black). The lower traces show 4X-IB link power using the proposed technique

applying our policy for link activation and deactivation. We see that link power reduction is possible during computation phases and, if compression is used, also during communication phases. Whenever compression cannot be used; e.g. for the application-driven communication in Gromacs, the links are fully operative, consuming full power.

Figure 8 shows the energy savings for the Alya CG kernel. The energy reduction is about 70% for eight processes, reducing to about 50% for 128 processes. The total energy savings decrease with the number of processes, since, assuming strong scaling, the computation phases become shorter, reducing the lengths of the idle periods. There are, however, greater relative energy savings from data compression. With 50% compression (Input A), the difference in energy savings between the smallest and largest runs is just 10%. Considering only the benefits from idle link periods, this difference increases to about 20%.




(a) INPUT A.

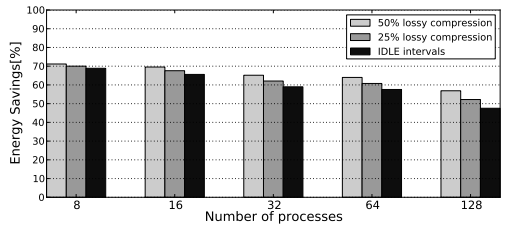



(b) INPUT B.

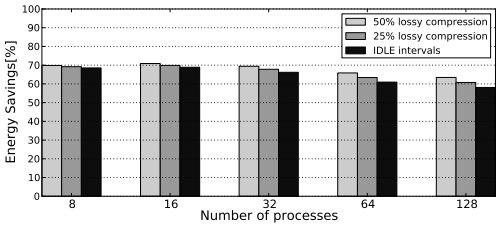
Fig. 9. Link energy savings for Gromacs PME kernel for one MPI process per node

The larger Input B has a lower drop in energy savings from increasing the number of processes. Data compression will, however, still be important if the number of processes is increased much above 128. The same observation is relevant for Gromacs, but to a lesser extent, as shown in Figure 9. The communication phases for which compression can be used make up a smaller part of the overall execution time.

If more than one MPI process is allocated to a node, the link will be powered up if any of these processes needs to communicate; i.e. it is powered up when the first process on the node starts to communicate and powered down when the last process finishes communication. This should lead to higher link utilization, decreasing the potential for link energy savings, compared with a single process per node. Figure 10 shows the results for the Alya CG kernel with six processes per node, which are little different from the results with one process per node. In contrast, Figure 11 shows the results for the Gromacs PME kernel, with six processes per node. Here, a higher link utilization leads to lower link energy savings.

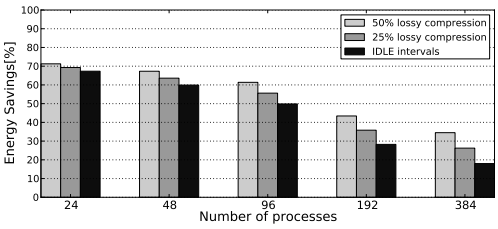



(a) INPUT A

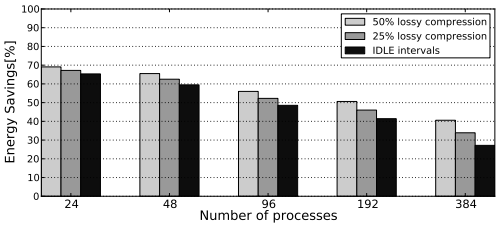



(b) INPUT B

Fig. 8. Link energy savings for Alya CG kernel for one MPI process per node

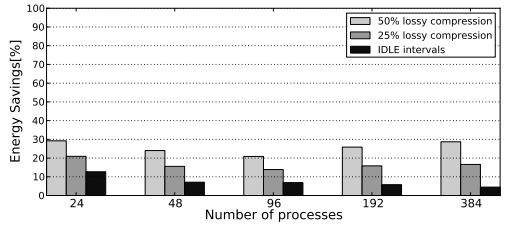



(a) INPUT A.

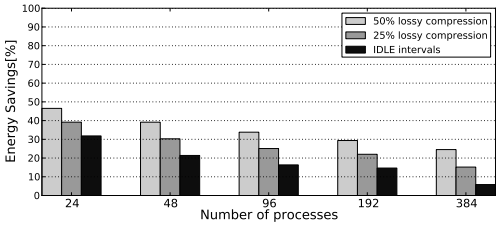



(b) INPUT B.

Fig. 10. Link energy savings for Alya CG kernel for six MPI processes per node




(a) INPUT A.

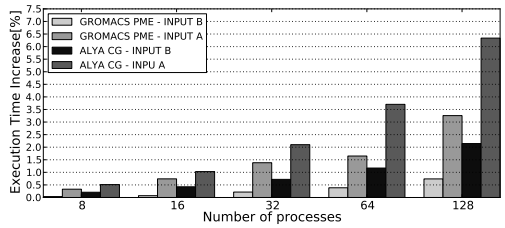



(b) INPUT B.

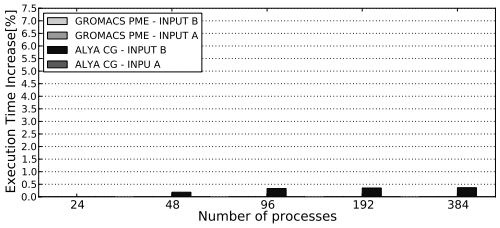
Fig. 11. Link energy savings for Gromacs PME kernel for six MPI processes per node

This is clearest for Input A, which is considerably smaller, soon reaching the limits of strong scaling. In this case, the link energy savings from compression, in percentage terms, increase with the number of processes. This is simply because communication phases account for an increasing percentage of the total execution time.

Finally, we investigate the performance overheads caused by the link deactivation and activation penalties. With one process per node, as shown in Figure 12(a), the maximum increase in execution time was 6%, but it was usually considerably less. With six processes per node, in Figure 12(b), the execution time was increased by less than 0.5%. This is because the reactivation penalties were each time paid by the first process to communicate, which is generally the fastest process, and therefore not on the critical path. The communication on the critical path incurs no penalty, preserving the original performance. The performance loss of Alya CG for Input A is because lossy compression provokes a small increase in the number of CG iterations until convergence.




(a) One process per node




(b) Six processes per node

Fig. 12. Applications kernels execution time increase due to reactivation time penalty and lossy compression

## VII. Related Work

A large body of related work revolves around the idea of using data compression for additional gain in terms of power and performance. Some works have applied compression to improve the bandwidth to the I/O subsystem during large scale molecular dynamic simulations [24], while others works focus on compressing MPI messages in distributed systems.

Burtscher et al. [14] proposed the FPC compression algorithm for double precision data. The FPC is lossless algorithm that works well for pattern based data sets. Previously, the authors proposed the DFCM compressor [15] for double precision data which is a subpart of the later developed FPC compressor. They incorporated DFCM in an MPI library to speed up MPI programs running on a cluster of workstations. This cluster used a low performance network (Fast Ethernet) which created high communication to computation ratio in application when executed on a large number of processors. Therefore, they achieved good speedups, up to 98%.

Another algorithm based on arithmetic prediction was proposed by Katahira et al. [13]. The algorithm is implemented in hardware and is used to enhance the memory bandwidth of LBM (Lattice Boltzman Method) stream-computing accelerators. They achieved compression rates up to 3.5×.

Filguera el al. [12] used different types of compression algorithms which were applied to MPI transfers at runtime depending on the type of data. They achieved good results, but mostly for integer and single precision FP data.

The work most similar to ours was done by Kumar et al. [16]. In their work, they also try to achieve performance gains by using both lossless and lossy compression schemes for double precision floating point data. Lossless compression schemes are similar to proposed by Burtscher et al [14] whereas lossy does the reduction in mantissa parts [6]. They applied both compression schemes in communication intensive part of Community Atmospheric Model application. With lossless compression scheme run on 32 processes, they achieve a speedup of 53.58%. These large speedups are obtained primarily because they run the tests on a cluster that uses a low performance network (Gigabit Ethernet). This made that cluster much more sensitive to message size.

Das et al. [20] proposed the similar approach like in our work but applied in NOC architectures. Simple compressor/decompressor was integrated in the NIC, to compress the network communication traffic.

Lately power consumption optimization of the network has become very important target for HPC system designs. Several reduction technique has been proposed, where most of them are based on Dynamic Voltage Scaling (DVS) approach. Shang et al. [25] proposed history-based DVS policy, where past network utilization has been used to predict future traffic. Soteriou at al. [26] propose software techniques that form an extension to the parallelizing compiler flow, which statically generate DVS instructions that later will direct run-time network power reduction.

Other techniques are based on turning off unused communication links or links with low utilization. Alonso et al. [27] propose a power saving mechanism for regular interconnection networks that are built with a high degree switches, where each network dimension is formed of various links in parallel. The idea is to turn off and on the links that compose trunk link as a

function of network traffic. All links but one can be turned off. Therefore, connectivity in the network is maintained, which allows the use of same routing algorithm regardless of power reduction level. In the work of Kim et al. [28] DVS technique is complemented with powering down under-utilized links. The use of adaptive routing algorithm is required in order to avoid deadlocks.

Lim et al. [29] use MPI run time system to reduce CPU power consumption during the communication phases. The run time system identifies communication regions and select processor frequency in order to minimize energy-delay product.

Jian Li et al. [7] work is focused on non-prediction power-saving techniques. Links are powered up just before they are needed, by relying on hints from the built-in system events or from macros in MPI source code. A separate control network is needed to exchange link activation messages. This network is always active. This approach is similar to ours in the way of powering on the data links. The difference is that we rely on IB architecture with links that offer a dynamic range in terms of performance and power. Using the compression, we don't need to power up full link width but just the width that is based on achievable compression rate of transmitted data. Therefore, the link power is saved while original performance of an application is preserved.

In the work of Abts et al. [18], authors propose energy proportional datacenter networks. Link data rates are selected on the basis of traffic intensity in the network. They use the congestion sensing heuristic to sense traffic intensity, dynamically activating links as they are needed. While this work is focused on datacenter applications, which can tolerate small changes in latency, the HPC applications can not afford such performance loss.

## VIII. Conclusions

In this paper, we evaluated the benefits of MPI data compression on performance and energy. We observed that lossy compression generally has a limited effect on performance. The blocking nature of point-to-point MPI calls in the nearest-neighbour pattern, where only a single message is outstanding in communication between each pair of processes, does not overload network resources at the HCA. More time is spent on scheduling and synchronization inside the communication pattern than on the actual data transfer. Also, when the size of the messages and the number of neighborhood processes for each process are variable, the total time of communication is also affected. On the other hand, patterns like all-to-all tend to synchronize the tasks, leading to a larger speedup. This communication pattern loads the HCA channel with multiple MPI messages, so a reduction in size improves performance.

To the best of our knowledge, we are the first to apply data compression to link energy savings. Using compression allows the number of active lanes to be reduced in proportion to the compression rate. Thanks to compression, even with reduced network bandwidth, the application performance is not affected. Reactivation delays typically increased execution time by just a few percent. Using 50% compression, we obtained total link energy savings of up to 71% for the Alya CG kernel and 63% for Gromacs PME. We also show that strong scaling runs, in particular, have a large benefit from data compression.

## IX. Acknowledgements

This work has been supported by the Spanish Severo Ochoa award SEV-2011-00067 and project TIN2012-34557 from the Ministry of Science and Innovation, by the Generalitat de Catalunya 2014-SGR-1051 award and the EU Mont-Blanc project (FP7/2007-2013 under grant agreement n° 288777 and 610402). Branimir Dickov was supported by Catalan Government with Pre-doctoral scholarship FI (reference no. 2009FI-B00077).

## References

[1] "Power savings features in mellanox products," Mellanox, Sunnyvale, CA, USA, january 2013, http://www.mellanox.com/related-docs/whitepapers/.
[2] "Infiniband trade assoc." 2012, http://www.infinibandta.org/.
[3] TOP500, "Top500 Supercomputer Sites," http://www.top500.org/, June 2014.
[4] "Ibm. ibm infiniband 8-port 12x switch," 2011, http://www-3.ibm.com/chips/products/infiniband.
[5] "MPI forum: MPI: A message-passing interface standard," *The International Journal of Supercomputer Applications and High Performance Computing*, vol. 8(3/4), pp. 165–414, 1994.
[6] B. Dickov, M. Pericas, G. Houzeaux, N. Navarro, and E. Ayguade, "Assessing the impact of network compression on molecular dynamics and finite element methods," in *HPCC*, june 2012, pp. 588 –597.
[7] J. Li, W. Huang, C. Lefurgy, L. Zhang, W. Denzel, R. Treumann, and K. Wang, "Power shifting in thrifty interconnection network," in *HPCA*, feb. 2011.
[8] "Alya system - large scale computational mechanics," Barcelona Supercomputing Center, Barcelona, Spain, 2011, http://www.bsc.es/plantillaA.php?cat_id=552.
[9] "Gromacs(GROningen MAchine for chemical simulations)," University of Groningen, The Netherlands, 2011, http://www.gromacs.org.
[10] J. R. Shewchuk, "An introduction to the conjugate gradient method without agonizing pain," Carnegie Mellon University, Tech. Rep., 1994.
[11] C. Sagui and T. A. Darden, "Molecular dynamics simulations of biomolecules: long-range electrostatic effects," in *Annu Rev Biophys Biomol Struct*, vol. 28, 1999, pp. 155–179.
[12] R. Filgueira, D. E. Singh, A. Caldern, and J. Carretero, "CoMPI:enhancing mpi based applications performance and scalability using run-time compression," in *EUROPVM/MPI*, Espoo, Finland, September 2009.
[13] K. Katahira, K. Sano, and S. Yamamoto, "FPGA-based lossless compressors of floating-point data streams to enhance memory bandwidth," in *ASAP*, 2010.
[14] M. Burtscher and P. Ratanaworabhan, "FPC:a high-speed compressor for double-precision floating-point data," *IEEE Transactions on Computer*, vol. 58(1), pp. 18–31, January 2009.
[15] J. Ke, M. Burtscher, and E. Speight, "Runtime compression of mpi messages to improve the performance and scalability of parallel applications," ser. SC '04, Washington, DC, USA, 2004, pp. 59–.
[16] V. S. Kumar, R. Nanjundiah, M. Thazhuthaveetil, and R. Govindarajan, "Impact of message compression on the scalability of an atmospheric modeling application on clusters," *Parallel Computing*, vol. 34(1), pp. 1–16, February 2008.
[17] V. Soteriou and L.-S. Peh, "Design-space exploration of power-aware on/off interconnection networks," in *ICCD*, oct. 2004, pp. 510 – 517.
[18] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu, "Energy proportional datacenter networks," *SIGARCH Comput. Archit. News*, vol. 38, no. 3, pp. 338–347, Jun. 2010.
[19] V. Sathish, M. J. Schulte, and N. S. Kim, "Lossless and lossy memory i/o link compression for improving performance of gpgpu workloads," in *Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques*, ser. PACT '12. ACM, 2012, pp. 325–334.
[20] R. Das, A. K.Mishra, C. Nicopoulos, D. Park, V. Narayanan, R. Iyer, M. S. Yousif, and C. R. Das, "Performance and power optimization through data compression in network-on-chip architectures," in *HPCA*, Februar 2008.
[21] T. Hoefler, "Software and hardware techniques for power-efficient hpc networking," *Computing in Science Engineering*, vol. 12, no. 6, pp. 30–37, 2010.
[22] C. Minkenberg and G. Rodriguez, "Trace-driven co-simulation of high-performance computing systems using omnet++," in *Proceedings of the 2Nd International Conference on Simulation Tools and Techniques*, ICST, Brussels, Belgium, 2009.
[23] J. Labarta, S. Girona, V. Pillet, T. Cortes, and L. Gregoris, "Dip: A parallel program development environment," in *Euro-Par'96 Parallel Processing*. Springer Berlin Heidelberg, 1996, pp. 665–674.
[24] L. Yongpeng, Z. Hong, L. Yongyan, W. Feng, and F. Baohua, "Parallel compression checkpointing for socket-level heterogeneous systems," in *HPCC*, September 2011.
[25] L. Shang, L.-S. Peh, and N. K. Jha, "Dynamic voltage scaling with links for power optimization of interconnection networks," in *HPCA*, 2003.
[26] V. Soteriou, N. Eisley, and L.-S. Peh, "Software-directed power-aware interconnection networks," *ACM Trans. Archit. Code Optim.*, vol. 4, no. 1, Mar. 2007.
[27] M. Alonso, S. Coll, J.-M. Martínez, V. Santonja, P. López, and J. Duato, "Power saving in regular interconnection networks," *Parallel Comput.*, vol. 36, no. 12, pp. 696–712, Dec. 2010.
[28] E. J. Kim, K. H. Yum, G. M. Link, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, M. Yousif, and C. R. Das, "Energy optimization techniques in cluster interconnects," in *ISLPED 2003*. New York, NY: ACM, pp. 459–464.
[29] M. Y. Lim, V. W. Freeh, and D. K. Lowenthal, "Adaptive, transparent frequency and voltage scaling of communication phases in mpi programs," in *SC*, New York, NY, 2006.