Reducing the Impact of Soft Errors on Fabric-Based Collective Communications

José Carlos Sancho, Ana Jokanovic, and Jesus Labarta

Barcelona Supercomputing Center
Barcelona, Spain

Abstract. Collective operations might have a big impact on the performance of scientific applications, specially at large scale. Recently, it has been proposed Fabric-based collectives to address some scalability issues caused by the OS jitter. However, soft errors are becoming the next factor that significantly might degrade collective's performance at scale. This paper evaluates two approaches to mitigate the negative effect of soft errors on Fabric-based collectives. These approaches are based on replicating multiple times the individual packets of the collective. One of them replicates packets through independent output ports at every switch (spatial replication), whereas the other only uses one output port but sending consecutively multiple packets through it (temporal replication). Results on a 1,728-node cluster showed that temporal replication achieves a 50% better performance than spatial replication in the presence of random soft errors.

1 Introduction

Large-scale HPC clusters containing thousands of nodes are usually interconnected using a commodity interconnect such as *InfiniBand* [1] and arranged by a cost-effective slimmed fat-tree topology [2]. One popular example of these systems is the Roadrunner supercomputer [3] built at Los Alamos National Laboratory which was the first supercomputer that achieved one Petaflop of peak performance.

On these systems, MPI is the de facto standard for communication. MPI provides both point-to-point communications as well as collective communications. Collective communications are group communications between many nodes used for different purposes such as combining partial results of computations (such as Gather and Reduce), synchronization of nodes (Barrier), and publication (Broadcast). However, because collectives involve the participation of all the members in the group before it can conclude, any variance in node communication responsiveness and performance for any member of the group have a big impact on the completion time. One major cause of variability is produced by the OS jitter. Recently, *Fabric-based collective communications* [4] have been proposed to address this scalability problem by moving the collective's calculation from nodes onto switches which do not have OS jitter.

Another important factor that introduces a high variability and performance degradation of collectives are soft errors. These errors corresponds to alterations

M. Alexander et al. (Eds.): Euro-Par 2011 Workshops, Part II, LNCS 7156, pp. 262–271, 2012.

in the bit stream received over a communication channel. They are caused by various factors such as channel noise, interference, distortion, bit synchronization and attenuation problems. The frequency of occurring these errors is measured by network manufactures by the *Bit Error Rate* (BER)—the number of bit errors divided by the total number of transferred bits during a studied time interval. Typical BER values found on transmission channels range from 10^{-12} up to 10^{-15} for high-end optical cables. Although, the probability of an error happening on a single channel is small, the large amount of communication channels found in clusters results in a system-wide failure rate quite small. Note that next generation of supercomputers will contain in the order of millions of channels.

Unfortunately, Fabric-based collectives can suffer from soft errors. The general approach to deal with these errors in current interconnection networks is to detect errors at the receiver side with an *CRC* code, and then ask for a re-transmission if the error is positive. However, these re-transmissions are inevitably adding delays to the individual messages involved in each step of the collective's calculation resulting in an overall performance loss. Providing a technique to avoid these re-transmissions or ameliorate their negative impact would be of great interest. One possible technique would be the use of error-correcting codes (ECC), so errors can be detected and also corrected. However, since some of the collective's messages are not fully stored in switches, ECC is not viable on Fabric-based collectives.

In this paper, we propose the use of message replications in order to reduce the degradation caused by soft errors on Fabric-based collectives. Two different replication techniques have been evaluated: spatial and temporal replications. Results on 1,728-node InfiniBand cluster arranged on a slimmed fat-tree shows that temporal replications is the most effective solution to mitigate the negative effects of soft errors on Fabric-based collectives. Performance of up to 50% can be achieved with respect to spatial replications.

The rest of this paper is organized as follows. Section 2 briefly describes the operation of Fabric-based collectives on slimmed fat-tree topologies. Section 3 shows how InfiniBand detects and handles soft errors. Section 4 describes two approaches based on spatial and temporal replications to mitigate the negative effects of soft errors. Section 5 characterizes the impact on collective performance when soft errors are present in the network for both proposed techniques. Section 6 summarizes recent approaches to deal with network errors. Conclusions from this work are given in Section 7.

2 Fabric-Based Collectives

Fabric-based collectives is an approach to accelerating the calculation of collective communications. It uses the switch CPU to perform the collective steps and required calculations instead of using the host CPU as in the traditional approach. Recently, it has been integrated with the popular OpenMPI and Platform MPI message passing libraries and it is fully supported on InfiniBand networks.

Basically, this scheme is composed of a manager that orchestrates the initialization of the collective communication tree and a SDK that offloads the

computation of the collective onto the switches. Today's switches have been redesigned and optimized to support a super scalar FPU hardware engine that performs single and double precision operations in just one single cycle. This technology have been specifically targeted to the $MPI\_Barrier$, MPI_Reduce, and $MPI\_Allreduce$ operations, which in turn are the most frequent operations found on scientific applications.

In essence, the collective calculation is composed on two phases— a reduction phase and a broadcast phase. In the first phase, switches aggregate the collective values from all the computing nodes and switches attached to them, calculate the resulting value, and forward it to higher level switches. The root of the collective tree calculates the final reduction and on the second collective phase the result is broadcasted to computing nodes using multicast operations. Notice that in order to calculate the reduction it implies that messages have to be fully received at the switches before sending the partial result to upper switches in the reduction phase. However, on the final broadcast phase, there is no need to wait until the full message is received to start transmitting it down to computing nodes.

3 Handling Soft Errors on InfiniBand

The InfiniBand's link layer is where most soft errors are detected when packets traverse through the network. According to its specification [5] there is a diverse set of soft errors that can be detected:

- Physical errors. Errors indicative of bit errors at the attached physical link. These are detected by CRC checks.
- Malformed packet errors. Errors indicative of packets transmitted with inconsistent content.
- Switch routing errors. Errors indicative of an error in switch routing.
- Buffer overrun. Error indicative of an error in the state of the flow control machine.

When one of these errors is detected on any single packet at the switches, the immediate action is to discard the packet, record the type of error for further processing, and notify to sender side of the transmission that the packet was corrupted. This last action is performed by the hardware-level ACK messages. Originally, InfiniBand's host channel adapters (HCA) where the ones sending these notifications back to the sender HCAs. However, these notifications had to be supported also at switches on Fabric-based Collectives because they are becoming now the originators of packets.

In case of a corrupted packet it might be just dropped if it has not been yet forwarded to the next switch; or in the case that it has been started the transmission, switches are appending a bad CRC value and the *End Bad Packet* delimiter (EBP) as an alternative to dropping the packet.

InfiniBand implements two different CRC checks in any packet which are invariant (ICRC) and variant (VCRC) CRCs. ICRC is 4 bytes long and covering only the fields of the packet which are invariant from end to end on the network.

The polynomial used is CRC-32. On the other hand, VCRC is a CRC-16 2-bytes long polynomial which covers all fields of the packet and it is computed at every switch. Both ICRC and VCRC are appended at the end of the packets. Additionally, note that there is no support for ECC on InfiniBand.

4 Collective Replication Approaches

In this section we are describing two different approaches to mitigate the impact of soft errors on Fabric-based collectives. The first one is based on replicating collective packets to different output ports on the switches; we called this approach as *Spatial replication*. And the second approach is also based on replicating collective packets, but using the same output port, we refer to it as *Temporal replication*.

4.1 Spatial Replication

This method takes advantage of the multiple number of output ports that might be available to connect to upper level switches on slimmed fat-tree topologies. In essence, it duplicates the resulting collective packets at each switch and retransmit them to each of these output ports. This message duplication provides some resiliency to soft errors— In case of packet corruption in one of these channels, the packet would be delayed, but another packet in the other output channel will still deliver the collective value to upper level switches without no delay. Therefore, with this scheme if the number of soft errors is lower than the number of replications on the output ports, then there is no degradation due to soft errors.

Figure 1 illustrates this approach showing in red lines the links used to calculate the collective on a slimmed fat-tree topology [2] (XGFT(3;4,4,2;1,4,2)) for a 16-node parallel application spread out in four switches in the network. As can be seen, S_0 duplicates the resulting value R_1 to both output ports which connect to two different upper-level switches, S_8 and S_9. In case that there is a soft error occurring in channel S_8 to S_{12}, R_5 can still be delivered to upper level switches through the alternate channel S_8 to S_{13}. Note that in this approach we will have multiple roots for calculating the final result. This is a necessary condition in order to provide also resiliency when broadcasting the final result back to the computing nodes. For example, if there is another soft error on channel from root S_{12} to switch S_8, roots S_{13}, S_{14}, and S_{15} can still transmit down the result R_7.

Also, notice that in order to provide resiliency at the HCAs, there must be also multiple independent output channels available on HCAs. This can be achieved on InfiniBand networks in two ways. The first one may involve the use of dual-port HCAs where each port connects to the same switch or to another switch in the network. This is the most expensive solution because it requires to duplicate the number of switches in the network in order to accommodate the same number of computing nodes. Because of that this solution is not commonly found in current production clusters. The second solution, which is more

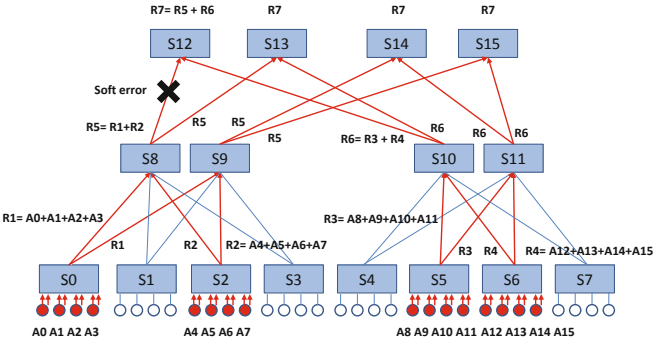

Fig. 1. Spatial replication scheme

cost-effective, relies on single-port HCAs, but re-configuring the port to two or more independent channels. InfiniBand defines three different port speeds, 1X, 4X, 12X. 4X and 12X are achieved by aggregating four and twelve 1X channels together, respectively. Since most production clusters actually deploy 4X-port HCAs we can actually configure them as four 1X independent channels in order to get independent output channels. However, notice that with this solution, soft-error resiliency is provided, but at the expenses of reducing the available HCA bandwidth.

4.2 Temporal Replication

The temporal replication scheme re-transmits multiple times every collective packet but to the same output port on the collective tree. Basically, unlike the spatial replication, during the reduction phase one output port is enough to provide resiliency to soft errors because the same packet is being re-transmitted consecutively multiple times. And during the collective's broadcast phase we also multicast the collective final result to all output ports down towards computing nodes, and also sending multiple packets through these channels.

Figure 2 illustrates this scheme for the same example as before for the collective reduction phase. As it is shown, only one output port at each switch and HCA is used, but on this channel the same packet is being transmitted twice. Note that with this scheme there is no need to split up the HCA's port in multiple independent output channels because we only require one output port. Therefore, there is no reduction on bandwidth at the HCAs.

However, contrary to the spatial replication scheme it may be some penalty when a packet got corrupted. For example, in case that R_5 got corrupted by a soft error over the channel S_8 to S_{12}, S_{12} will discard R_5 and it will wait until the R_5's second packet arrives. The collective's completion would suffer a delay of just one packet transmission. On the other case, if the corrupted packet was

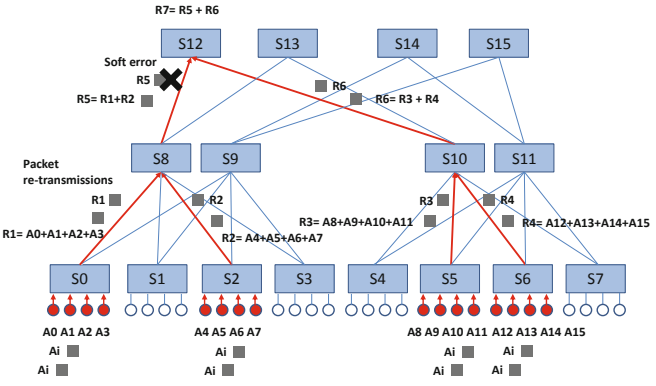

Fig. 2. Temporal replication scheme

the second R_5 instead of the first R_5 then there will be no delay at all because the first packet will still have a valid value and the second packet would be just discarded when it arrives at S_{12}.

5 Experiments

In this section, we have evaluated both schemes, *spatial* and *temporal* replications Fabric-based collectives in the case of having one or multiple soft errors in the network. The evaluation is performed by simulation using the Venus network simulator [6]. In this simulator, it has been implemented the Fabric-based collective technology and also both fault-tolerant schemes.

A large network containing 1,728 computing nodes is used in the evaluation. It is arranged on a 3-level slimmed fat-tree topology, XGFT(3;24,12,6;1,12,6). InfiniBand network is considered using 36-port switches and single port HCAs with a 100ns delay for each HCAs and switches. A 4X SDR (10Gb/s) port configuration is used in the evaluation. On the *spatial* approach the HCA's bandwidth is reduced proportionally to the number of output port replications. Replications of two, four, and six are considered for both *temporal* and *spatial* techniques.

We used the $MPI\_Allreduce$ collective operation in our evaluations. This is the most common collective operation found in scientific applications because it is being used on Conjugate Gradient Solvers. We assume a collective operation with few operands that actually fit on the InfiniBand's minimum transfer unit that consist of 256 bytes.

One and multiple soft errors are injected in the network. It is considered the case of one and two soft errors on specific channels and also multiple soft errors randomly affecting multiple channels. In the latter case it is following a exponential distribution of errors with mean values ranging from 1, 10, 100, up to 1000 µs. In this case, the average time to complete the collective in the presence of soft errors is reported over a thousand collective operations.

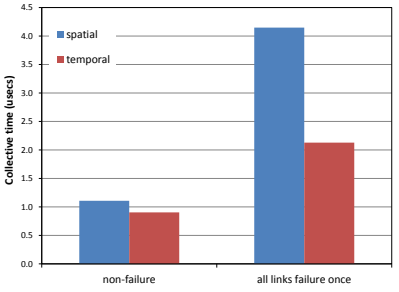

Fig. 3. Non-failure and all channels failure scenarios

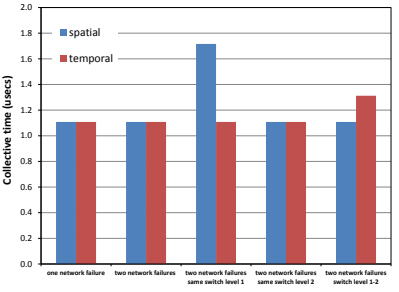

Fig. 4. One and two soft errors on selected network channels

5.1 Results

Figure 3 shows the performance of Fabric-based collectives for *spatial* and *temporal* replications on two extreme cases: the case of a failure-free scenario and the case of experiencing a failure on all the network links once. In both techniques the number of replications used is two. As can be seen, *temporal* significantly outperforms *spatial* by 22% in the case of non-failures. The reason for that is that the HCA's bandwidth had to be divided up to support multiple output channels on *spatial*. For the last extreme case of experiencing a soft-error on every link, *temporal* still significantly outperforms *spatial* because *spatial* is not able to provide a free-failure path, and thus it is been heavily penalized due to multiple re-transmissions of collective packets once soft errors have been detected. In particular, there is a difference in performance of almost a 2X factor.

Figure 4 shows various cases having one and two soft errors occurring in specific network channels. The first case, having only one soft error in a channel significantly degrades *temporal* by 22% with respect to the non-failure case showed before. This is because the waiting time to get the second collective packet. In the case of *spatial* does not suffer additional degradation because another collective packet is still being transmitted through another channel. In this scenario both approaches are achieving the same performance. Similarly, the case to having two simultaneous soft errors in two different network channels and in two different switches, but at the same tree level, is not further degrading the performance of both techniques. However, the interesting case comes when soft errors are occurring in the same switch on the level 1 for *spatial*. In this particular case, both output ports are experiencing soft errors and thus it can not provide a fault-free path suffering a 55% degradation. This case is not happening at higher levels of the fat tree as it can be seen in the next set of results. This is due to the fat-tree topology where replications on different output ports on a switch on level i make the collective going to different switches on the upper level $i+1$. And hence, if one these switches at level $i+1$ is experiencing failures the other switch on the same level can still deliver the collective to upper levels. The last case, shows a worse scenario for *temporal* where soft errors are actually

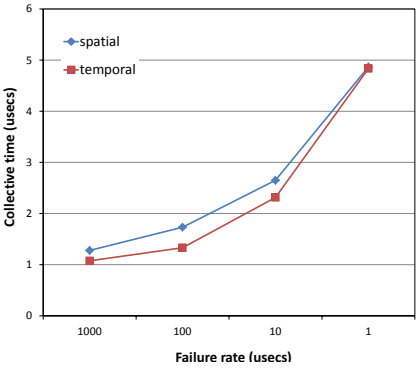

Fig. 5. Multiple random failures

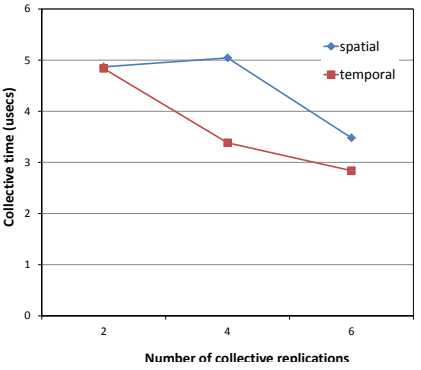

Fig. 6. Various collective replications on the worst scenario (1 µs)

occurring in two connecting switches, but each one sit on a different level. In this failure scenario, *temporal* is suffering higher degradation than *spatial* because the first collective packet is being consecutively delayed in both switches. Therefore, suffering twice the penalty to wait for the second collective packet to arrive.

Figure 5 shows the scenario of experiencing random soft errors over multiple collectives. As can be seen, *temporal* significantly outperforms *spatial*. Specifically, at 1ms, 100 µs, and 10 µs the collective time is reduced by 18%, 30%, and 14%, respectively. The margins are reduced at less frequent failure rate (1,000 µs) because when there is a small number of failures both techniques performs similar as it was shown before. Also, on a very high failure rate (1 µs), both techniques perform the same. The reason for that is that there are too many soft errors that almost any collective packet is having a soft-error, and thus both techniques suffer from having multiple re-transmissions. In order to provide more resiliency to this environment we increased the number of collective replications up to six as it is shown in Figure 6. As can be seen, the collective time significantly decreases as we increases the number of replications, specially for *temporal*. In particular, the collective time drops by 30% and 46% when going to 4 and 6 replications. Note that *spatial* is not decreasing significantly collective time from 2 to 4 because it is also reducing the available HCA bandwidth proportionally. Overall, *temporal* is still outperforming *spatial* for these cases. Performance improvements of 50% and 22% are seen for 4 and 6 replications.

6 Related Work

Providing fault-tolerance to MPI communications is a hot research topic today. Currently, the MPI Forum's Fault Tolerance Working Group is working in that subject. Recently, they have just proposed a run-through stabilization component in MPI to deal with failures [7]. This component provides an application with the ability to continue running and using MPI even when one or more

processes in the MPI universe fail. In this context, it has been analyzed various fault-tolerant algorithms to deal with hard failures for collectives in [8] and [9].

In [8] these algorithms are based on a new MPI function $MPI\_Comm\_validate$ that can check process failures any time. If a process failure is detected then it re-builds the collective tree accordingly, so for the next collective, the tree is already working and optimized. In [9], the collective tree is only re-built when the failure is detected during the collective operation. However, re-building the collective tree is too expensive in order to handle soft errors on Fabric-based collectives.

Additionally, it has been proposed in [10] an enhancement resilient protocol for Eager and Rendezvous point-to-point communications that covers fabric end-to-end hard and soft failures including also HCAs. Unlike our approach, the basic idea is to solely act as soon as a failure is detected, but this approach may lead to a higher degradation. We believe that a pro-active approach—also acting before a failure will occur— is better to reduce the potential harmful degradation of soft errors.

7 Conclusions

Soft errors are having a big impact on the performance of collective communication operations. For these communication operations, solely acting when soft errors occur is not efficient enough, and thus pro-active solutions are highly recommended. We have evaluated two of these pro-active solutions called spatial and temporal replications.

Evaluations show that temporal replications deliver higher performance than spatial replications. In particular, a 50% lower degradation is observed on temporal with respect to spatial in the presence of soft errors. Therefore, temporal replications are effectively diminishing the impact of soft errors. In addition, temporal replications can be seamlessly deployed in current production systems because it does not require the use of special hardware. Note that spatial would require at least 4X HCAs.

We understand that the only benefit of spatial would come from the potential to mitigate also hard failures. However, this work demonstrates that spatial achieves a very poor performance with respect to temporal, and thus it will make it less attractive to be deployed as a stand-alone solution.

Acknowledgments. We thankfully acknowledge the support of the Spanish Ministry of Science and Innovation under grant RYC2009-03989, the European Commission through the HiPEAC-2 Network of Excellence (FP7/ICT 217068), the Spanish Ministry of Education (TIN2007-60625, and CSD2007- 00050), and the Generalitat de Catalunya (2009-SGR-980).

References

1. InfiniBand website: Infiniband trade association, official website on, `http://www.infinibandta.org`
2. Öhring, S.R., Ibel, M., Das, S.K., Kumar, M.J.: On generalized fat trees. In: Proceedings of the 9th International Parallel Processing Symposium, p. 37. IEEE Computer Society, Washington, DC (1995)

3. Barker, K.J., Davis, K., Hoisie, A., Kerbyson, D.J., Lang, M., Pakin, S., Sancho, J.C.: Entering the petaflop era: the architecture and performance of roadrunner. In: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing, SC 2008, pp. 1:1–1:11 (2008)
4. Mellanox: Fabric collective accelerator (2011), http://www.mellanox.com/related-docs/prod_acceleration_software/fca.pdf
5. InfiniBand specification: Infiniband trade association, infiniband architecture specification, vol. 1, release 1.0.a (2001)
6. Minkenberg, C., Rodriguez, G.: Trace-driven co-simulation of high-performance computing systems using OMNeT++. In: Proceedings of the 2nd International Conference on Simulation Tools and Techniques, Simutools 2009 (2009)
7. Fault Tolerance Working Group: Run-though stabilization interfaces and semantics, svn.mpi-forum.org/trac/mpi-forum-web/wiki/ft/runthroughstabilization
8. Hursey, J., Graham, R.: Preserving collective performance across process failure for a fault tolerant. In: 16th International Workshop on High-Level Parallel Programming Models and Supportive Environments (HIPS) held in conjunction with the 25th International Parallel and Distributed Processing Symposium (IPDPS), Anchorage, Alaska (May 2011)
9. Jaros, J.: Evolutionary Design of Fault Tolerant Collective Communications. In: Hornby, G.S., Sekanina, L., Haddow, P.C. (eds.) ICES 2008. LNCS, vol. 5216, pp. 261–272. Springer, Heidelberg (2008)
10. Koop, M.J., Shamis, P., Rabinovitz, I., Panda, D.K.: Designing high-performance and resilient message passing on infiniband. In: Communication Architecture for Scalable Systems Workshop held in conjunction with the 25th International Parallel and Distributed Processing Symposium (IPDPS), Atlanta, Georgia USA (April 2010)