Achieving Breakthrough MPI Performance with Fabric Collectives Offload

A Voltaire White Paper

Authors

Aviv Cohen (avivc@voltaire.com)

Yiftah Shahar (yiftahs@voltaire.com)

Yaron Haviv (yaronh@voltaire.com)

Asaf Wachtel (asafw@voltaire.com)

Contents

Abstract

This paper introduces a breakthrough in capability computing over x86 standard server clusters. The economic benefits of using commodity based clusters over monolithic Massively Parallel Processing (MPP) and Symmetric Multi-Processing (SMP) supercomputers are well known. The growth of x86 based clusters on the TOP500 list of the fastest supercomputing sites (TOP500.org) is evidence of this trend. Between June 2005 and June 2010, the number of x86 based clusters on the TOP500 list grew from 61% to 85%.

However, while MPP supercomputers comprise 15% of the TOP500, the MPP share of the TOP100 supercomputers is 50%. Scaling application performance over large cluster-based computers is a known challenge due to the inherent limitations of parallel communication, i.e. Message Passing Interface (MPI).

Despite extensive research, no practical solution has been found to date to improve the performance of MPI communication on a large scale. In this paper, we will review the inhibitors to application scalability and introduce a new technology that enables a quantum leap in MPI performance over x86 based clusters. In addition to accelerating MPI collective run time by two orders of magnitude, this solution requires no additional hardware, is compatible with current versions of MPI and does not require any changes to the application code.

Introduction

What are MPI collectives?

In High Performance Computing (HPC), MPI is the de facto standard for communication among processes that model a parallel program running on a distributed memory system.

MPI functions include point-to-point communication — for example, exchanging data between process pairs — and group communication between many nodes. Group communications are used for combining partial results of computations (Gather and Reduce), synchronizing nodes (Barrier), and publication (Bcast).

It is important to note that for some collectives, the operation involves a mathematical group operation that is performed amongst the results of each process, such as summation or determining the min/max value. For example, the MPI_Reduce call can take data from all processes in a group, performs an operation (such as summing), and stores the results on one node. Reduce is often useful at the beginning or end of a large distributed calculation, where each processor operates on a part of the data and then combines it into a final multi-node result.

What prohibits x86 cluster application performance scalability?

There are several reasons why MPI based applications do not scale well in x86 distributed clusters, despite the hundreds of man-years and millions of dollars invested in improving different MPI routines in various MPI distributions.

Both the root cause and the solution lay within the cluster's network and the collective operations. Collective operations command group communication, which by its nature has to wait on all the members of the group to participate before it can conclude. The result is that the weakest link in the chain or group determines the pace of the entire group. In addition, any variance in node communication responsiveness and performance can add up to create considerable system compute variance and unpredictability.

Applications can spend up to 50%-60% of their computation time on collectives, and this percentage increases as the number of nodes involved in the computation goes up. Considering the inefficiencies in collective networking explained below, this presents an enormous opportunity for improving HPC resource utilization.

Problems With Collective Scalability

There are several reasons for the inefficiency of collectives today and the lack of linear scalability that are worth explanation.

Cluster hotspots and congestion

Fabrics, particularly lossless HPC fabrics, suffer from congestion or "hot spots" when send/receive operations over sporadic communication links reach the link capacity limit and cause a delay in communication. A common myth is that a non-blocking configuration can eliminate this problem by providing a higher overall average IO capacity that is sufficient for the server throughput. In reality, application communication patterns are rarely evenly distributed and hot spots occur more often than we would wish to imagine.

Since collective messages share the same wire as the rest of the cluster traffic, they are affected by congestion and must wait in the various network queues and buffers while being sent or received. They are also susceptible to congestion due to the "many-to-one" nature of group communication and the overall large amount of collective messages traveling over the fabric. As mentioned above, the slowest link of the group collective operation determines

the time spent for the entire operation. The larger the environment, the more switch hops the message needs to pass through, and the chance for congestion and jitter increases. In blocking topologies such as blocking-CLOS, 3D torus and mesh, scalability issues are intensified. Only by isolating and shielding collective operations from the rest of the traffic in the network can this issue be resolved.

Collective operations disconnect of the physical topology

Collective operations, which by nature are group operations, are treated as discrete point-to-point communication operations in the lower level MPI code. A logical binary tree is built where each pair performs a collective sub-procedure. The result is further computed as point-to-point with the result of another pair, and so on. The number of stages for a cluster with (n) server nodes for an AllReduce operation would be 2×log(2n) and the number of message-hops sent over the wires in a five-tier Clos would be an order of magnitude 2O(12n). While each message enters the fabric amongst two arbitrary nodes, hot spots and buffers add delays and the overall collective operation time increases.

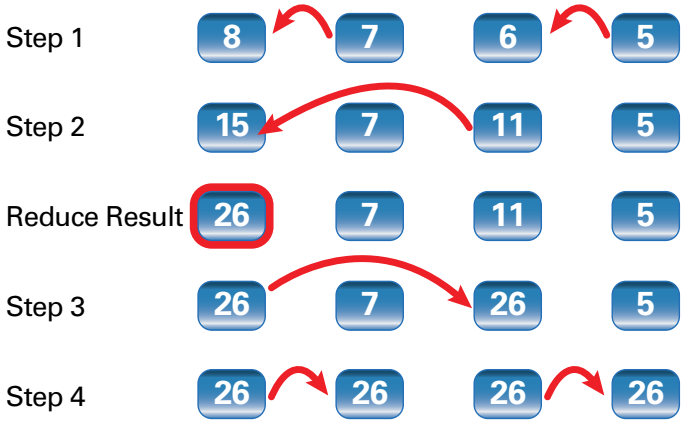


Figure 1: Collective example computational steps: Allreduce (SUM)

Efforts have been made to optimize the collective logical tree and create trees with more branches than a binary tree per each computational step. However, since the algorithm has no knowledge of the actual fabric topology, the pairs of ranks for collective operations are selected arbitrarily. The cores communicating may be located at distant parts of the fabric. Without the ability to match the collective logical tree with the cluster's physical rank topology and physical rank proximity, this inefficiency cannot be overcome.

Server OS "noise"

In non-real-time operating systems like Linux or Windows, there are many tasks and events that can cause a running process to perform a context switch in favor of other tasks and return after some time. Some of the main contributors to operating system noise include hardware interrupts, page faults, swap-ins, and preemption of the main program. Assuming the collective operations or messages are due to processing, this mistimed context switch may put them on hold until the original process resumes. In large scale operations, this can extend the collective operation dramatically as delays and context switch mismatches add up and increase computation time, while the majority of the fabric is still waiting for all the nodes to conclude the collective computation.

Shielding collective operations from OS noise is instrumental to reducing the time spent on collective operation communication as well as the overall application runtime.

Introducing Fabric-Based Collective Offload

Today, there is a new solution that takes an innovative, system-wide approach to accelerating collectives. This approach offloads the computation of collectives onto the fabric switches, thus accelerating collective computation by 10X-100X and providing a drastic reduction in job runtime. Implementing this solution is simple, and the transparency and bottom-line benefits makes it an obvious choice for large scale clusters.

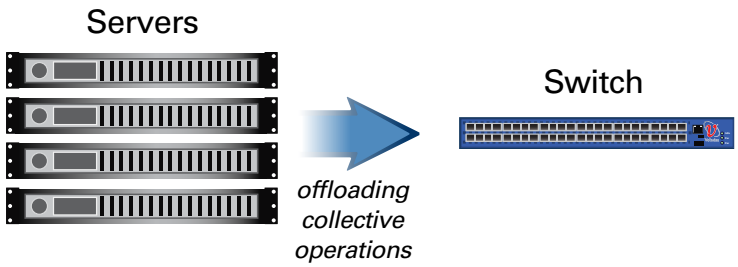
There are three main principles that form the fabric-based collective offload solution:

- Network offload
- Topology-aware orchestration
- Communication isolation

The following sections describe the concept and benefits of this innovative solution, followed by several benchmarks.

Network offload

When offloading floating point computation from the server CPU to the network switch, there is another compute resource being utilized—the switch CPUs. The collective operations can be easily handled by the switch CPUs and their respective cache memory for best performance.



Topology-aware orchestration

The fabric Subnet Manager (SM) has complete knowledge of the fabric physical topology and can ensure that the collective logical-tree optimizes the collective communication accordingly. It constructs the collective logical tree using the following steps:

1. Ensure that the collective communication is optimized within the node by utilizing fast inter-core communication to the greatest extent possible.
2. Build the collective tree in a way that ensures that the first-tier, top-of-rack switch aggregates the collective operations of all the nodes attached to it.
3. All top-of-rack (TOR) switch collectives are aggregated to a core switch of choice that performs the final collective stage and distributes the results as needed.

The correlation between the logical collective tree and the physical topology minimizes the total number of messages. In fact, for each collective operation only a single message goes over each wire, not only avoiding congestion but also reducing the amount of overall traffic on the network.

Using this method, the number of collective computational steps for a cluster with (n) server nodes is reduced from O(log(2n)) to O(3) regardless of the fabric size, and delivers a quantum leap in overall collective runtime. Having a constant number of steps regardless of the fabric size is key to ensuring application performance linearity in medium to large environments.

Communication isolation

Collective communication is isolated from the rest of the fabric traffic by making use of a private virtual network (VLane) to assure complete isolation from other Inter-Process-Communication (IPC) and storage traffic. Additionally, when the computation is performed by the switch CPU, the collective is shielded from the OS noise as described above.

Adapter-based collective offload

Another approach for improving MPI collective performance is the adapter-based (also known as NIC-based) collective offload approach. This approach aims to move the progression of the collective messaging from the main CPU onto the network interface card and thus reduce "OS noise."

The adapter-based offload approach delegates collective communication management and progress, as well as computation if needed, onto the Host Channel Adapter (HCA). This approach addresses the issue of OS noise shielding, but cannot be expected to improve the entire set of collective inefficiencies, such as fabric congestion and topology awareness. The InfiniBand HCA can be programmed to offload a number of collective operations by making use of the HCA messaging queues and the mathematical processor, in conjunction with the host server DRAM memory.

Published references[1] on this approach are limited to a small scale. Additionally, these references do not include any standard MPI network benchmarks (e.g. IMB) or commercial application benchmarks.

Looking at published results, the HCA-based collectives (marked MQ-RDMA) show similar latency as regular OpenMPI solutions (marked PTP-RDMA). In some cases, the performance is 50% worse (32µs with 64 ranks vs 20µs with software only). The steep incline of the graph also raises questions about the scalability of the solution. In this example, when the number of ranks doubled (from 32 ranks to 64 ranks) the latency tripled (from ~10µs to ~30µs).

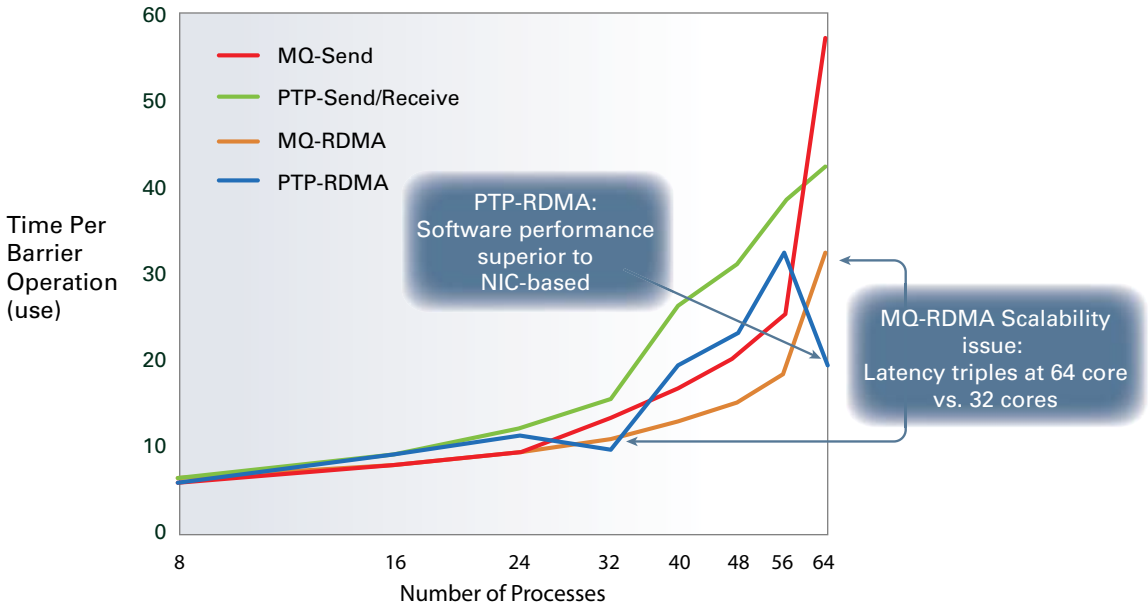


Figure 2 - Scalability limitation with adapter-based offload

A look at the implementation mechanisms of the adapter-based approach suggests questionable scalability. The HCA has limited memory and therefore requires reads and writes onto the host buffers through the PCI bus for every operation and message. As the size of the job increases, so too does the number of HCA resources used, like QPs and memory elements. This in turn increases memory consumption and cache misses, resulting in added latency for the collective operation. These factors combined with the PCI express transactions inherent to this mechanism are likely to cripple scalability on a larger scale.

1 Source: "Overlapping Computation and Communication: Barrier Algorithms and ConnectX-2 CORE-Direct Capabilities," Gilad Shainer, April 2010

In addition, HCA offload is dependent on InfiniBand point-to-point Reliable Connection (RC) QP. As a result, this approach does not take advantage of the benefits of multicast communication, and suffers from network point-to-point congestion—which exists even in a non-blocking topology. This problem becomes even more severe in fat tree blocking topologies and 2D and 3D torus networks.

The HCA offload solution also relies on non-standard InfiniBand kernel driver components and hardware programming for more operations. In the larger MPI ecosystem, incorporating this solution into leading MPI open source and commercial packages that are used by ISV's would take a very long time.

Below is a summary table comparing the fabric-based offload approach with the adapter-based offload approach.

	Fabric-based offload	Adapter-based offload
OS "noise" reduction	Yes	Yes
Support non-blocking collectives	Yes	Yes
Topology aware	Yes	No
Network congestion elimination	Yes	No
Fabric switches offload computation	Yes	No
Blocking topologies	Not affected	Performance degradation
Result distribution based on IB multicast	Yes	No
Supported collective operations	Allreduce, Reduce, Barrier, Bcast	Barrier only
HCA compatibility	Any	Connect-X2 only
Host stack	OFED	Proprietary
Host resources (QPs, memory, HCA cache, etc)	Light use	Heavy use
Performance	13us@64 ranks; 22us@5K ranks	32us@64 ranks
Scalability	O(Log18)	Poor
Maturity	Several production customers	Lab trials
Scalability	O(Log18)	Poor
Maturity	Several production customers	Lab trials

Table 1: A comparison of the fabric-based offload approach and the adapter-based offload approach

Voltaire Solution

Voltaire has designed a unique patent-pending technology based on the the fabric-based collective offload approach described above called **Fabric Collective Accelerator (FCA) software.** This non-blocking collective architecture finally allows InfiniBand to scale collective communication to thousands of nodes better than any interconnect (standard or proprietary) on the market.

The solution is composed of a manager that orchestrates the initialization of the collective communication tree and an MPI library that offloads the computation onto the Voltaire switch CPU, which is built into each of Voltaire's 40 Gb/s InfiniBand switches. This CPU has a super scalar FPU hardware engine that performs 32/64 floating point operations (single cycle, single and double precision).

To enable users of leading MPI technologies to benefit from this solution, a simple SDK is provided. The SDK includes the initialization/destruction APIs as well as the relevant collective operations within a dynamically linked library, enabling a simple, standard integration point for linking this functionality into the MPI code. The complex protocols for initialization, identification, communicator establishment and destruction, reliability, and order handling, among others, are all invisible to the end user.

Currently this technology is integrated with OpenMPI and Platform MPI. Additional integration efforts are underway with other MPI technologies and will be announced in the near future.

Main benefits of Voltaire FCA

The benchmarks that follow demonstrate a substantial reduction in MPI collective runtime (up to 100 times faster) and an overall reduction of application runtime. This performance boost does not require any additional hardware and there is no space/power/cooling penalty. Furthermore, there is no need to change the application once the MPI technology used is in compliance with the offloading library.

Fabric collective offload reduces congestion in the fabric by minimizing the amount of collective traffic that travels over the wires. Particularly in blocking cluster configurations that are created with cost in mind, this solution can deliver significant performance improvement and raise compute capability levels closer to those of a non-blocking configuration.

As described above, the product is scale-proof and affected only by the number of tiers in the fabric – not the number of servers or cluster size. In fact, the larger the fabric size the more beneficial the solution.

Benchmark Comparisons

The solution described above has been proven to show dramatic improvement in collective and overall application runtime performance. These results were achieved without any additional hardware and provide tremendous benefit over the network connectivity features.

This section describes specific examples of improved application and collectives run time using benchmarks performed by Voltaire, its partners and customers.

IMB Pallas

One customer from a major technical university in Europe has tested FCA extensively and is already enjoying higher productivity rates as a result of the solution.

The cluster in use has the following topology:

Switching	Voltaire® Grid Director™ 4036 QDR InfiniBand Switch on the core
Server	NEC HPC 1812Rb-2
Total cores	2048
CPU	2 x Intel X5550
Memory	6 x 2GB
OS	CentOS, 5.4 kernel: 2.6.18-164.el5
Open MPI	1.4.1
HCA	DDR on board

Table 2 - Customer Benchmark Configuration

Measurements were made in job sizes of 256, 512, 768, 1600 and 2048 ranks for 4 Byte messages within the IMB benchmark. The topology involved 24 DDR rate downlinks to the servers with 12 QDR uplinks going to the core switches as depicted in the following diagram:

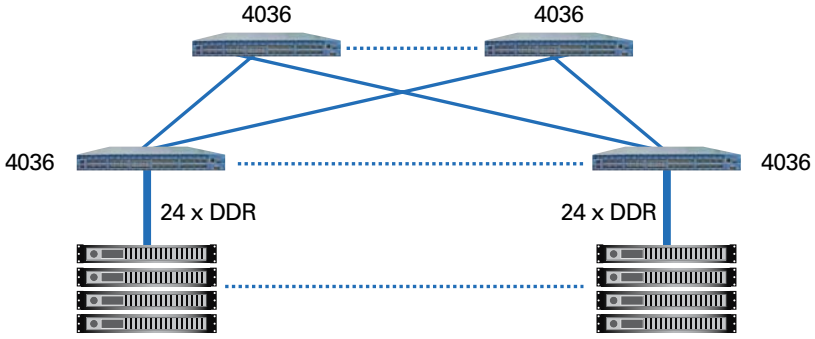


Figure 3 - Customer Benchmark Configuration

The findings of these runs pertaining to Barrier and Allreduce collectives are shown below. Note that the scale of the y-axis is logarithmic.

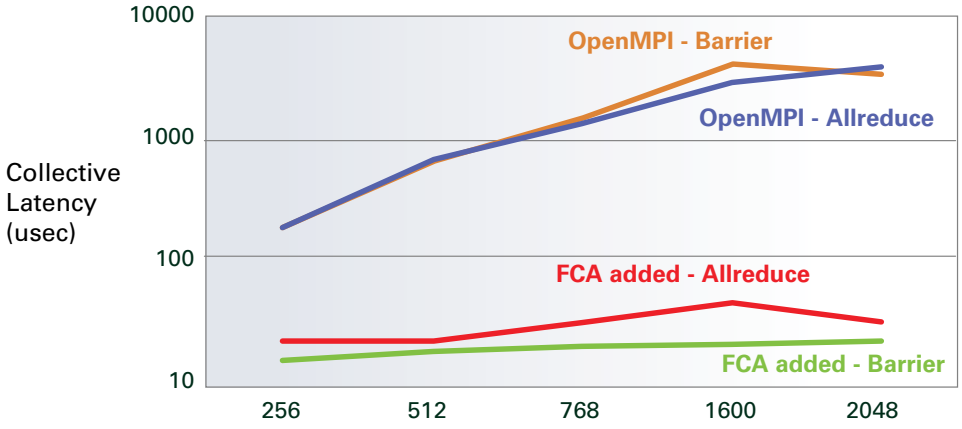


Figure 4 - Customer IMB Benchmark Results

	Number of Ranks				
Allreduce	**256**	**512**	**768**	**1600**	**2048**
Vanilla OMPI	200	669	1343	2894	3638
OMPI + FCA	24	24	33	48	34
Acceleration Multiplier	8x	28x	40x	60x	107x
Barrier	**256**	**512**	**768**	**1600**	**2048**
Vanilla OMPI	201	671	1516	4019	3467
OMPI + FCA	17	19	22	23	24
Acceleration Multiplier	12x	35x	69x	179x	145x

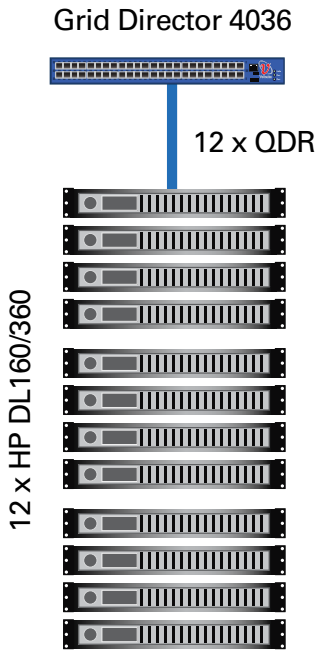
Table 3 - Customer IMB Benchmark Results

Result Analysis

As the benchmark results above show, FCA can accelerate IMB Pallas collectives up to 100X and beyond on a very large scale. The effect on the application run time is determined by the portion of collectives within the application. Benchmarks of additional collectives such as Bcast and Reduce have also shown similar results. It's important to note that as the size of the computation increases, so does the relative acceleration of FCA.

Ansys Fluent (12.1.4)

The cluster used within Voltaire's Cloud Center of Excellence (CCoE) features the following configuration:



Switching	Voltaire® Grid Director™ 4036 QDR InfiniBand Switch
Server	HP DL160 / DL360
Total cores	64
CPU	Intel(R) Xeon(R) X5550 @ 2.67GHz
OS	CentOS, 5.4 kernel: 2.6.18-164.el5
Open MPI	1.4.1
OFED	1.5.3

Table 4 - Voltaire Cloud Center of Excellence Configuration

Based on the configuration above and Fluent standard benchmarks, the following performance improvements were observed:

Application Module Name	OpenMPI Rating	Voltaire FCA Rating	FCA Acceleration
aircraft 2m	3150	3639	15.5%
eddy 417k	3469	4586	32.2%
sedan 4m	3702	4062	9.7%
truck 111m	47	55	16.3%

Table 5 - Fluent Benchmark Results

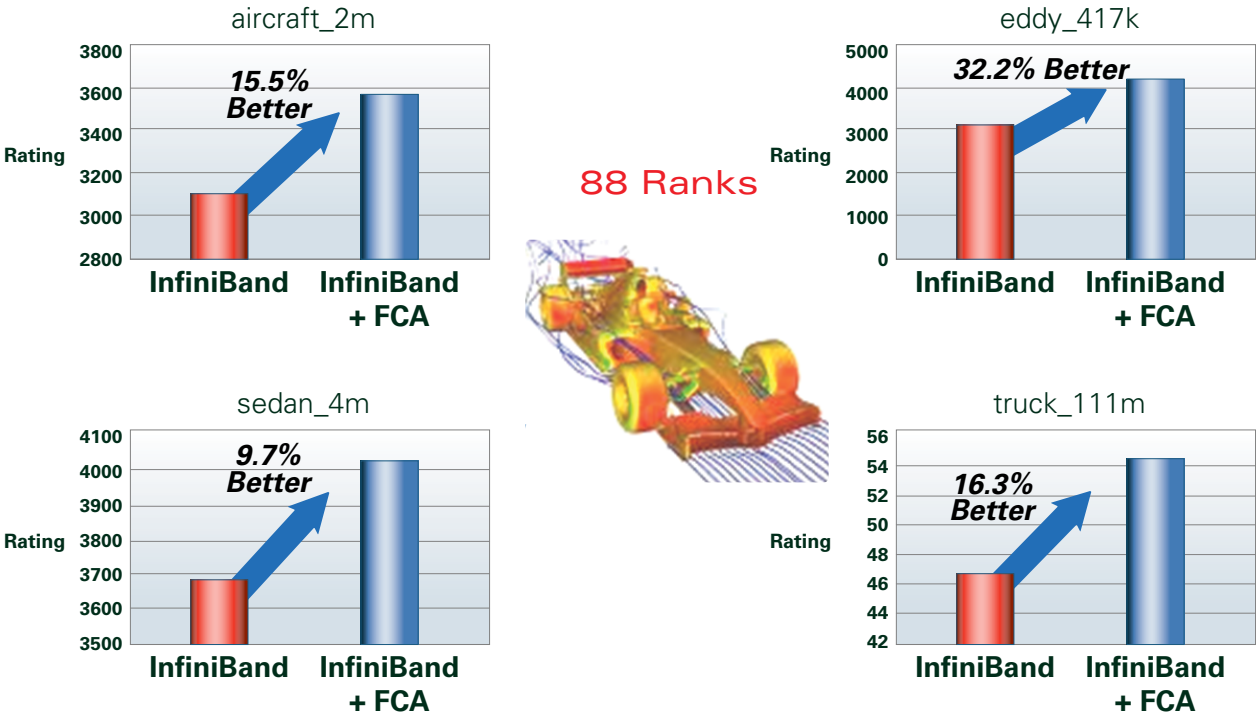


Figure 5 - Fluent Benchmark Results

Result Analysis

One can immediately see a gain of 10%-30% runtime acceleration by using FCA fabric collective offloading technology at the application run time level.

OpenFOAM

This benchmark uses the cluster at Voltaire's Cloud Center of Excellence (CCoE) described in Table 4, with 64 cores using HP DL160 / DL360 servers.

The following performance improvements were observed:

	oodles/ pitzDaily			simpleFoam/pitzDaily		
	OpenMPI	Voltaire FCA	diff	OpenMPI	Voltaire FCA	diff
Clock time [seconds] (Wall clock time of test execution)	598	523	**12.5%**	51	47	**7.8%**
Execution time [seconds] (clock time CPU_load)	491	392	**20.2%**	34	28	**18.2%**

Table 6 - OpenFOAM Benchmark Results (64 cores)

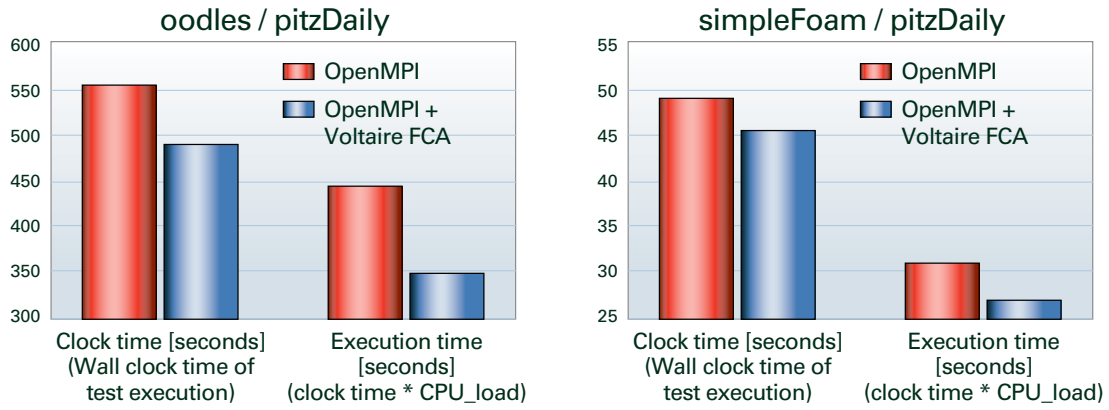


Figure 6 - OpenFOAM Benchmark Results (64 cores)

Result Analysis

With only 64 cores, this configuration demonstrates an impressive acceleration of 8%-12% run time improvement. The rates are even higher when the CPU load of the computation is included. Additionally, as the size of the computation increases, so does the relative acceleration of FCA.

Looking Towards The Future

Voltaire Fabric Collective Accelerator (FCA) will continue to evolve in line with the following objectives.

First, integration efforts are underway to increase the number of MPI technologies that use FCA. In the future, any application will be able to benefit from the performance boost delivered by FCA, regardless of the MPI technology in use. Intel MPI and MVAPICH are both candidates for future integrations.

Second, FCA will evolve to cover more collective operations, data types, message sizes and unique topologies. This evolution will be driven by customer needs and these new features will be added in future versions of FCA.

Third, the ability to use fabric-computing in service of other applications such as map/reduce and in applicability to PGAS-based applications will also be added to FCA in the future.

Summary

Voltaire Fabric Collective Accelerator (FCA) enables the first fabric computing infrastructure to accelerate MPI computation in HPC clusters in a pragmatic, easy and ready-to-use solution.

In a number of benchmarks, FCA has been proven to show impressive performance improvement over fabrics supplemented by only a firmware upgrade. Configurations using FCA have shown 10X to 100X collective performance acceleration and a 10% to 40% improvement in overall application runtime.

The FCA offloading engine is embedded within Voltaire's QDR (40Gb/s) InfiniBand switches, including the Grid Director™ 4036, Grid Director™ 4200 and Grid Director™ 4700, and is activated by software.

FCA addresses a wide range of applications with pragmatic, out-of-the-box integration with leading MPI technologies and job schedulers. In addition to delivering bottom line performance improvement, FCA shows linear scalability without any space/power/cooling CAPEX or OPEX penalties.

About Voltaire

Voltaire (NASDAQ: VOLT) is a leading provider of scale-out computing fabrics for data centers, high performance computing and cloud environments. Voltaire's family of server and storage fabric switches and advanced management software improve performance of mission-critical applications, increase efficiency and reduce costs through infrastructure consolidation and lower power consumption. Used by more than 30 percent of the Fortune 100 and other premier organizations across many industries, including many of the TOP500 supercomputers, Voltaire products are included in server and blade offerings from Bull, Fujitsu, HP, IBM, NEC and SGI. Founded in 1997, Voltaire is headquartered in Ra'anana, Israel and Chelmsford, Massachusetts. More information is available at www.voltaire.com or by calling 1-800-865-8247.

Contact Voltaire to Learn More

1.800.865.8247
info@voltaire.com
www.voltaire.com