IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 4, April 2017

A Survey on Performance Evaluation and Optimization of a Linear System Solver in OpenFOAM on Intel MIC

Sumeet Patil[1], Vikas Kumar[2], Vinaya Sivanandan[3]

M.E. Student, Computer Engineering, Pune Institute of Computer Technology, Pune, India [1]

Principal Technical Officer, HPC Scientific and Engineering Application Group, C-DAC, Pune, India [2]

Technical Officer, HPC Scientific and Engineering Application Group, C-DAC, Pune, India [3]

Abstract: OpenFOAM is a scientific library popularly used for solving and simulating Computational Fluid Dynamics (CFD) problem based on the finite volume method. A most recent hardware Intel Xeon Phi coprocessor introduced by Intel which is based on many integrated core (MIC) architecture. The MIC coprocessor has numerous cores than the traditional Intel Xeon processor but each core works at a lesser frequency and is less powerful compared to the Xeon counterpart. In this paper, we studied different hybrid and heterogeneous platform and discussed various methods for achieving good scalability across nodes on Intel MIC. We have discussed a few approaches for evaluating and optimizing the performance of message passing interface (MPI) to reduce the execution time of OpenFOAM solver on Intel MIC architecture.

Keywords: Intel MIC, Linear system solver, OpenFOAM, Optimization.

I. INTRODUCTION

OpenFOAM is written in C++ object-oriented tool-kit for continuum mechanics and published under the license of GNU-GPL. It can be considered as a framework for Computational Fluid Dynamics (CFD) programmers to develop their own code which is more than being a software ready-to-use, as it provides them with the idea sufficient to think of a problem in terms of the underlying mathematical model [2]. OpenFOAM has been ported and evaluated over numerous different architectures which includes massively parallel clusters, as its use became popular over the time.
The arrival of co-processors such as Intel Many Integrated Cores (MIC) is modifying the scenario of high performance computing. Intel's Xeon Phi coprocessor is constructed on the basis of Many Integrated Core (MIC) architecture. It offers 61 cores per card and gives performance of up to 1 TFLOP of double precision. It runs a micro operating system based on linux, also provides x86 downward offering, and for application developers it provides a highly flexible usage model. The MIC is a memory driven architecture where its cores operate at lower clock speed. Additionally, the communication properties between processes on MIC are also dissimilar when compared to communication between the host processes. The predefined communication libraries in many application do not consider the architectural differences and thus cannot deliver good performance in communication metrics. The performance of MPI collective functionality mostly influence the performance of parallel applications.
There are various aspect when considering optimization of OpenFOAM solver. In Section II we will go through various approaches of evaluation and optimization of OpenFOAM on Intel MIC platform and different techniques will be discussed to select the best method for optimization. In Section III we conclude of the review of our work.

II. REVIEW OF LITERATURE

We studied the following related systems to discover their advantages, drawbacks and limitations and these are discussed below.

A. Hybrid and Heterogeneous Platforms

AlOnazi, Keyes, Lastovetsky and Rychkov el at [1][9] proposed optimization aimed at accelerating numerical solver solution on modern and emerging hybrid heterogeneous platforms. Two OpenFOAM solvers icoFoam and laplacianFoam, which are illustrative to numerous low-order discretization of incompressible flow. OpenFOAM CFD package were selected because of it is famous as an open source tool which is adopted widely in industries and research organizations. They studied the behaviour of conjugate gradient method (CG). The conjugate gradient performance per iteration is based on the execution of each single kernel. There are three fundamental kernels in CG: inner product of vector, updates in vector to vector, and multiplication of sparse matrix-vector. MPI is used to parallelize the execution and communications is also introduced on a distributed memory architecture for The parallel-run of the conjugate gradient [1]. Two kind of communications are essential to carry on the execution of algorithm. The first type is communication is point-to-point during the multiplication of matrix-vector to incorporate the parallel interaction between neighbor processors. The second type of communication is collective operation which is used to form inner product by reduce the sum of scalars. CUDA and MPI libraries [9] were merged to design a hybrid conjugate gradient solver and then it was implemented on heterogeneous platform. During the heterogeneous decomposition a load-balancing step is involved, which consider the performance of each computing platform and reducing communication. The main motivation was to improve the performance of the OpenFOAM based CFD applications and effectively using the resources allocated.

B. Bottlenecks in OpenFOAM scalability

OpenFOAM is scalable as such to allow an appropriate usage of architectures. The performance of computation on the single node is restricted by the memory bandwidth accessible itself on the node. The sparse linear algebra core libraries gives way to performance and scalability issues. Massimiliano Culpo el at [2] briefly analyzed Preconditioned Conjugate Gradient (PCG) method. Multiplication of scalar values, preconditioning steps and multiplications of matrix-vector are the most heavily executed operations during each iterative cycle PCG. It is useful to say that these computation are similar to all the methods based on Krylov subspaces, and therefore a entire set of linear solvers will have a definite impact by their optimization [10]. The presence of the MPI_Allreduce routine means that scalar products act as implicit communication barriers during which partial sums are reduced among tasks after products are computed. This limitation, forced by the domain decomposition strategy in order to reduce the amount of replicated data, cannot be simply avoided and contributes to the scalar product the operation that largely restricts the scalability of the execution. Massimiliano Culpo el at [2] conducted analysis after which they suggests two ways to optimize the performance of solvers in OpenFOAM. The first technique is to implement cache blocking to minimize the number of cache misses in the primary operations due to the access patterns that are mostly random. The second technique was the modification to make the basic linear algebra routines multi-threaded using different ways [10].

C. Low Overhead MPI for Intel MIC

The advancement of microprocessors from the single-core en route to the current many-core is a result of the advanced integration solidity that can be achieved through semiconductor processes in modern era. The fresh processing power of many-core processors is a key element in supporting a number of computationally heavy applications, particularly in the multimedia domain. Efficiently converting this fresh processing capability into actual execution performance remains a challenge, weighted on the beneath software and hardware architectures [11]. Applications developed form this model are illustrated as a set of communicating job, with well clarified input and output needs. The communicating tasks run asynchronously on different processing components and swap data using the point-to-point connections. Current MPI system administrate the transfer of messages between processing components using feature-rich software libraries. Therefore, in addition to defining what actual data must be shifted between executing tasks, the developer need to manage the equivalent resource allocation and actual data transfer. Kumar, Djie and Leuken el at [3][11] presented, a message passing system with low overhead for many-core processors. This message passing system achieved performance by reducing state-of-the-art Direct Memory Access (DMA) latencies by at least 30% using the lower end-to-end transfer. Analysis showed that the system was giving scalable throughput and execution speedup on a many-core accelerator with the presented message passing system [3]. The performance earns are rapidly lost as communication overheads approach task execution times in many-cores against the sequential execution time. To improve throughput of the many-core system, it is vital that latencies of message transfer must be kept lower. For data movement and synchronization in message transfers which is controlled directly by software requires processor interruption. DMA based message passing system is used in this system for data transfers. The usage of these functions into hardware implementation have resulted in transfer latencies up to 30% shorter than state of the art MPI derivatives [3].

D. Optimizing MPI on Intel MIC

The performance of OpenMP directives, MPI communication library and different aspects of a Xeon Phi based system were studied and evaluated using microbenchmarks. Technique of inter-node collective communication in MIC clusters is optimized in [5]. In opposite to that, Panigrahi, Kanchiraju, Srinivasan, Baruah, and Sudheer el at [4] have optimized intra-MIC overall communication. Profiling the bandwidth of memory, memory access latency or the cache-to-cache bandwidth is difficult on the Xeon Phi because the latency depends not only on the distance from the core, but also on the distances from the distributed tag directory to the memory controller and from the memory controller back to the requesting core. Panigrahi, Kanchiraju, Srinivasan, Baruah and Sudheer el at [4] used the STREAM benchmark [12] for experiments and to study the memory bandwidth. They considered condition demanding MPI messages of lower sizes,

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 4, April 2017

which are common in many parallel applications. Therefore, they focus mainly more on smaller sizes, where the performance tends to be lesser than for bigger message sizes, and then compared the results with transfer from cache-to-cache. The technique of cache-to-cache transfers take the shortest path on the ring of core in the Xeon Phi. They used POSIX shm open and mmap function provided by shared memory as communication channel between the processes. For each MPI communicator they allocated shared memory of a fixed size. This region is adjoining and is memory mapped at the time of the formation of the communicator and terminated with it. Showing that accessing remote caches in definite ways produces significant performance advantage over memory accesses. The result showed that an improvement in performance by up to a factor of 10 for small message, and factor 4 for large messages can be obtained [12]. The algorithms and methods were also more scalable than the existing MPI implementation provided by Intel.

E. Optimized MPI on Intel MIC for Infiniband Clusters

Networking technologies, such as, InfiniBand, have also quickly developed to offer low latency and high range of bandwidth for communication to convey the growing communication need of current generation of petascale applications. Kandalla, Venkatesh, Hamidouche, Potluri, Bureddy and Panda el at [5][13] anticipated various new levels of communication on out coming Intel MIC clusters: 1) Intra node communication on same MIC device, 2) Intra node communication between processes on MIC and CPU, 3) Inter node communication between processes on MIC and CPU across different nodes, and 4) Inter node communication between processes on MICs across different nodes. Every communication method have different communication characteristics of its own. Therefore, it is important to carefully optimize communication libraries to take advantage of new techniques in networking and communication systems. Here the author [5] propose a collective structure to optimize the performance of main collective operations, like MPI_Bcast, MPI_Reduce and MPI_Allreduce, on Intel MIC clusters. MPI collective operations can be simply used to indicate group communication design. Processes that share the same address are enclosed together [13]. The process whose have the lowest rank within each node is known as the "leader" and term "leader-comm" consists of all such leaders, across all the nodes on a cluster. Kandalla, Venkatesh, Hamidouche, Potluri, Bureddy and Panda el at [ ] proposed a approach MPI_BCast, where they break the original communicator into two new sub-communicators. They created one communicator to consist only the host processes and another communicator that consist only the MIC leader processes [13]. This design enhanced the latency of the MPI_Bcast operation by up to 76% for 4,864 MPI processes. On heterogeneous MIC clusters an 52.4% improvements in the communication latency of the MPI_Allreduce operation with 2K MPI processes was observed.

F. Performance Evaluation of OpenFOAM with MPI-3 based on RMA

A standard OpenFOAM application decomposes the mesh into partitions and then does computation over it by assigning them to different MPI ranks over various nodes. For solving the partial differential equations it requires adjacent nodes to interchange boundary conditions between partitions using communication mechanism. When solving the equation inside every time loop, every MPI rank exchange data to and from its adjacent nodes using asynchronous calls such as MPI_Irecv, MPI_Isend which is followed by MPI_Waitall. Agrawal, Edwards, Pandey, Klemm, Ojha and Razak el at [6][14] aimed to remove these current asynchronous API communication calls with the one-sided RMA API of MPI for communication within a neighbouring ranks with the objective of improving performance of the application [14]. Their implementation strategy targets the Pstream library of MPI for optimization using RMA technique. Their analysis showed an enhancement using the RMA technique in native mode on the Intel Xeon Phi coprocessor by up to 8% for 0.3 million motorbike benchmark and an enhancement of 3% for the 4.2 million motorbike benchmark.

G. RDMA based MPI for Infiniband Clusters

The Message Passing Interface (MPI) has become the standard in making scientific applications which run in parallel on High Performance Clusters [7][15]. MPI gives point-to-point as well as collective communication method which can be used to design parallel application. Mostly scientific applications use the collective communication mechanism to exchange data within nodes/multi-cores. The MPI_Allgather [7] is an important collective operation which is a all-to-all broadcast mechanism is used in many approach such as multiplication of matrix, factorization of lower and upper triangle, solving partial differential equations, and mainly in operations like basic linear algebra. InfiniBand [8] is a hardware feature which is used in high performance interconnect for communication and I/O between the activity. Remote Direct Memory Access (RDMA) which allow a process to straightly access memory which is on a remote node/processor is a feature provided by InfiniBand. To utilize the advantages of this feature Sur, Bondhugula, Mamidala and Panda el at [7] designed a collective action directly using the RDMA. They described the design of various operation such as all-to-all broadcast using RDMA which allowed the application to mostly remove messaging expenses like additional message copies, protocol handshake and added buffer allocation. The design proposed uses the basic choice of algorithms [7] and implemented that over InfiniBand for a high performance model. Performance evaluation of designs showed that the latency of MPI All_gather can be lowered on 32 processes for a message size of 32 KB by up to 30%.

III.CONCLUSION

This survey can be seen as a stepping towards understanding the behavior of OpenFOAM solver. In this paper, we observed that most time consuming portion in solver execution on Intel MIC are message passing interface (MPI). Message passing bound applications, such as the OpenFOAM solvers, can take better advantage of the full hardware potential, if all resources are taken into account a lower execution time and optimized solver can be obtained.

ACKNOWLEDGMENT

I take this opportunity to express my deep sense of gratitude to Center for Development of Advanced Computing (C-DAC), Pune for giving me this splendid opportunity for my dissertation work. I would like to thank my dissertation guide Prof. S. S. Sonawane for her inspiration, priceless suggestions and support for this work. I also thank our dissertation coordinator Dr. A. S. Ghotkar and Head of Computer Engineering Department, Dr. R. B. Ingle for their support and help.

REFERENCES

[1] Amani AlOnazi, David E. Keyes, Alexey Lastovetsky and Vladimir Rychkov, "Design and Optimization of OpenFOAM based CFD Applications for Hybrid and Heterogeneous HPC Platforms", Computers and Fluids 00 (2015) pg 1–12.
[2] Massimiliano Culpo, "Current Bottlenecks in the Scalability of OpenFOAM on Massively Parallel Clusters", PRACE White Paper, pg 1-13.
[3] Sumeet S. Kumar, Mitzi Tjin A Djie and Rene van Leuken, "Low Overhead Message Passing for High Performance Many-Core Processors", First International Symposium on Computing and Networking, pg 345-351, 2013.
[4] Pinak Panigrahi, Sriram Kanchiraju, Ashok Srinivasan, Pallav Kumar Baruah and C D Sudheer, "Optimizing MPI Collectives on Intel MIC Through Effective Use of Cache", International Conference on Parallel, Distributed and Grid Computing, pg 88-93, 2014.
[5] K. Kandalla, A. Venkatesh, K. Hamidouche, S. Potluri, D. Bureddy and D. K. Panda, "Designing Optimized MPI Broadcast and Allreduce for Many Integrated Core (MIC) InfiniBand Clusters", 2013 IEEE 21st Annual Symposium on High-Performance Interconnects, pg 63-70, 2013.
[6] Nishant Agrawal, Paul Edwards, Ambuj Pandey, Michael Klemm, Ravi Ojha and Rihab Abdul Razak, "Performance Evaluation of OpenFOAM with MPI-3 RMA Routines on Intel® Xeon® Processors and Intel® Xeon PhiTM Coprocessors", EuroMPI '15, September 21-23, 2015, Bordeaux, France.
[7] S. Sur, U. K. R. Bondhugula, A. Mamidala, H.-W. Jin and D. K. Panda, "High Performance RDMA Based All-to-all Broadcast for InfiniBand Clusters", Department of Computer Science and Engineering, The Ohio State University, Columbus, Ohio 43210.
[8] Infiniband website [Online] Available : http://www.infinibandta.org/content/pages.phppg=about_us_infiniband
[9] H. G. Weller, G. Tabor, H. Jasak, C. Fureby, A Tensorial Approach to Computational Continuum Mechanics using Object-Oriented Techniques, Computers in Physics 12 (6) (1998) 620 – 631.
[10] H. Jasak, "HPC deployment of OpenFOAM in an industrial setting," in PRACE Seminar: Industrial Usage of HPC, 2011.
[11] J. Psota and A. Agarwal, "rmpi: message passing on multicore processors with on-chip interconnect," in Proceedings of the International Conference on High performance embedded architectures and compilers, pp. 22–37, 2008.
[12] Intel Xeon Phi Coprocessor System Software Developers Guide. http://software.intel.com/en-us/articles/intel-xeon-phi-coprocessor-system-software-developers-guide.
[13] K. Kandalla, H. Subramoni, G. Santhanaraman, M. Koop, and D. K. Panda, "Designing Multi-leader-based Allgather Algorithms for Multi-core clusters," in Proceedings of the 2009 IEEE International Symposium on Parallel and Distributed Processing, 2009, pp. 1–8.
[14] MPI: A Message-Passing Interface Standard. http://www.mpi-forum.org/docs/mpi-11-html/mpi-report.html.
[15] A. Mamidala, J. Liu, and D. K. Panda. Efficient Barrier and Allreduce on IBA clusters using hardware multicast and adaptive algorithms. In IEEE Cluster Computing, 2004.