# Evaluation of Message Passing Communication Patterns in Finite Element Solution of Coupled Problems

Renato N. Elias[1], Jose J. Camata[2], Albino Aveleda[2] and Alvaro L.G.A. Coutinho[2]

[1]IM/DTL, Multidisciplinary Institute,
Federal Rural University of Rio de Janeiro,
Av. Governador Roberto Silveira, s/n,
26210-210, Nova Iguaçu, RJ, Brazil.
rnelias@gmail.com

[2]NACAD, High Performance Computing Center,
Federal University of Rio de Janeiro, PO Box 68506,
21945-970, Rio de Janeiro, RJ, Brazil.
{camata, bino, alvaro}@nacad.ufrj.br

**Abstract.** This work presents a performance evaluation of single node and subdomain communication schemes available in EdgeCFD, an implicit edge-based coupled fluid flow and transport code for solving large scale problems in modern clusters. A natural convection flow problem is considered to assess performance metrics. Tests, focused in single node multi-core performance, show that past Intel Xeon processors dramatically suffer when large workloads are imposed to a single node. However, the problem seems to be mitigated in the newest Intel Xeon processor. We also observe that MPI non-blocking point-to-point interface sub-domain communications, although more difficult to implement, are more effective than collective interface sub-domain communications.

**Keywords:** Parallel Computing, Message Passing, Communication Patterns, Coupled Problems, Edge-Based.

## 1 Introduction

In 2008 the petascale barrier has been broken. According to Kogge [5] such systems can carry out real computations 1,000 times more challenging than those computable by early terascale systems. The size of such systems raises particular challenges, including performance on each node, scalable programming models, performance and correctness debugging, and improving fault tolerance and recovery. On the applications side, Gropp [6] stresses the fact that when discussing such systems researchers often overlook the increasing complexity of individual nodes, processors and the underlying network. Particular applications may benefit from the sheer power of such systems, but the majority of them have to be re-examined. Again, according to Gropp [6], researchers are creating new tools to develop, debug, and tune applications, as well as creating new programming models and languages that could enhance scalability by reducing communication overhead. The Computational Fluid Dynamics (CFD) community is aware of these new developments [7].

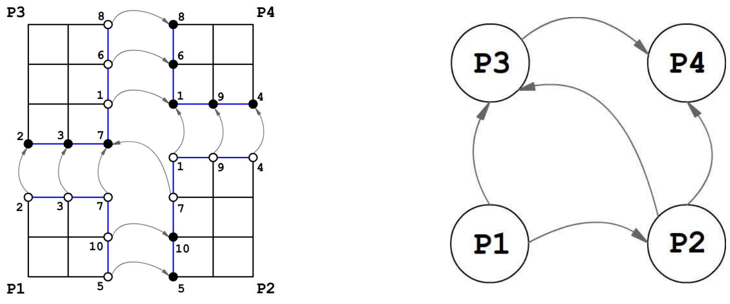
In Brazil there is a growing need to understand complex processes in the oil and gas industry. Particularly, understanding these processes is therefore critical to effective exploration for oil and gas in the recently discovered pre-salt fields in ultra-deep waters offshore in Brazil. Several of such complex processes can be recast in the general framework of fluid-structure interaction and coupled fluid flow and transport problems, involving multiple spatial and temporal scales. This paper presents a parallel performance evaluation of computation and communication models implemented in EdgeCFD, an implicit edge-based coupled fluid flow and transport solver for large-scale problems in modern clusters. EdgeCFD currently supports stabilized and multiscale finite element formulations and has been used in problems ranging from Newtonian and non-Newtonian fluid flows, free-surface flow simulations with fluid-structure interaction, gravity currents and turbulence (details available in [2] and references therein). Of particular interest in the present work is EdgeCFD's performance in the current multi-core processors, particularly process placement within processors and the impact of several subdomain communication models. The target machines are modern clusters with the latest processor and network technologies, paving the way towards sustained petascale performance. Following [4] and [10], where strategies for massive parallelism computations in unstructured grids are discussed, EdgeCFD adopts peer-to-peer non-blocking communication among processes.

The remainder of this paper is as follows. Next section details the benchmark software and communication models currently supported. The natural convection problem used to access parallel performance metrics is given in Section 3. The paper ends with a summary of our main conclusions.

## 2 EdgeCFD: The Benchmark Software

EdgeCFD was chosen to evaluate performance in several aspects, such as: parallel models, system architecture and processors. EdgeCFD was developed to exploit parallel architectures in four different ways, which broaden the range of machines that can be efficiently used. Three of them rely on message passing interface (MPI) implementations while the fourth one is based on threaded parallelism for shared memory systems or system components such as many-cores processors with shared memory at cache levels.

For the message passing implementations, the three variants differ in how data are split and messages exchanges are scheduled among processors. The simplest case is based on collective communication calls. In this strategy, nodes are divided in two groups: parallel interface and internal nodes. Interface nodes, all hollow and solid vertices in Figure 1a, are known by all processors. On the other hand, nodes owned exclusively by a processor are internal nodes. Computation of matrix-vector products and residuals are performed in just one, blocking, step where the parallel interface values are combined using MPI_ALLREDUCE operations. This collective communication model was further extended to remove the need of excessive data exchange as well as redundant information storage. In this case, globalization operations are also performed in one step but, now, using MPI_ALLGATHER calls. The more complex message passing parallel model employs peer-to-peer (p2p) message exchanges among processors and takes advantage of communication and computation overlapping. Following [4], the point-to-point (p2p) communication strategy is based on a master-slave relationship between processors. This relationship is established by creating a hierarchy based on host partition numbers. Thus, the processor $P_i$ is slave of $P_j$ if $P_i$ and $P_j$ are neighbors and $i < j$. Note that a processor can be slave and master at the same time, depending only on the number that identifies it in relation to its neighbors. Figure 1a illustrates a two dimensional mesh which is decomposed into four partitions. The hollow vertices denote the nodes, or degrees of freedom of the system of equations, that will be sent to the receiver (a master processor). On the other hand, the solid vertices represent the nodes or degrees of freedom that will be received from donors (a slave processor). Figure 1b shows the communication graph corresponding to this mesh. In this case, $P_1$ is slave of $P_2$ and $P_3$. $P_2$ is slave of $P_3$ and $P_4$ but it is master of $P_1$. $P_3$ is slave of $P_4$ and master of $P_1$ and $P_2$. Finally, $P_4$ is master of $P_2$ and $P_3$.



(a) Mesh Partition (b) Communication Graph

**Fig. 1.** Master-Slave relationship

The information exchange among neighboring processors is implemented in two stages: in the first stage, slaves processes send their information to be operated by masters (where the interface contributions are accumulated) and, in the second stage, the solution values are copied back from masters to slaves. In addition, EdgeCFD uses non-blocking send and receive MPI primitives, which allow communication and computation overlapping.

## 3 Performance tests

The three dimensional Rayleigh–Benard convection problem is used to investigate code performance in situations ranging from small to large scale simulations in different architectures and system configurations. The problem consists in a fluid, initially at rest, contained in a 3D rectangular domain with aspect ratio 4:1:1 (Lenght:Depth:Height) and subjected to an unity temperature gradient [3]. For a 4:1:1

container aspect ratio, with no-slip boundary conditions at walls, the flow is three dimensional and must gives rise to four convection cells. The fluid properties are set to result in Rayleigh and Prandt numbers of 30,000 and 0.71 respectively. For the performance tests two meshes with different discretizations are used. The coarser mesh (MSH1) is formed by 178,605 tetrahedra elements and 39,688 nodes while the finer one (MSH2) is made by splitting the domain in $501\times125\times125$ divisions, which gives rise to 39,140,625 tetrahedral elements. In both cases the solution is evolved towards steady-state using EdgeCFD's block sequential implicit time-marching scheme. In this scheme the Navier-Stokes block is solved by the Inexact-Newton method and the temperature block by simple multi-correction iterations. The inner iterative driver for both Navier-Stokes and temperature transport is an edge-based preconditioned GMRES method. A nodal block-diagonal preconditioner is used for the Navier-Stokes equations while a simple diagonal preconditioning is employed for the temperature equation. GMRES tolerance for the temperature is fixed at $10^{-3}$ while the maximum tolerances for the inexact Newton method is set to 0.1. For both, flow and transport, the number of Krylov vectors is fixed in 25. We consider that steady state is achieved when the relative velocity increment differs by less than $10^{-5}$.

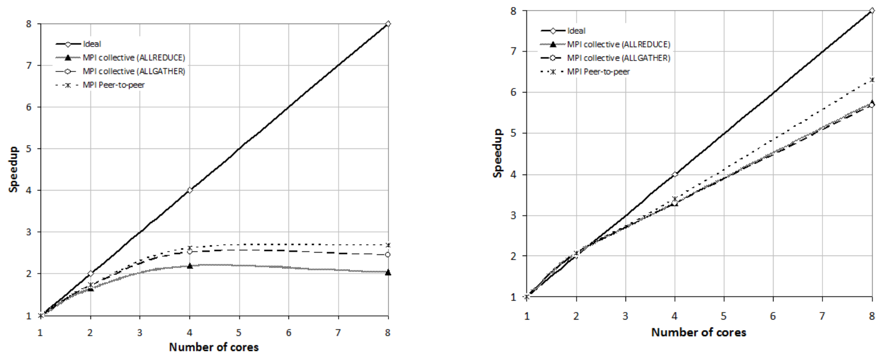
Tests are carried out in three different Intel Xeon-based HPC systems. All systems are equipped with quad core CPUs.

- *SGI Altix ICE cluster* with 32 compute nodes. Each node has eight 2.66 GHz cores (in two Intel Xeon Processor Quad-Cores, Clovertown - X5355). L2 cache size 8MB on-die for Quad-Core; 4MB per core pair; shared by the two cores. Memory Blade: 16GB. All nodes are interconnected using InfiniBand technology in a Hypercube topology.
- *DELL cluster PowerEdge M1000e* with 16 compute nodes (M600). Each node has eight 3.00 GHz cores (in two Intel Xeon Processor Quad-Cores, Harpertown - E5450). L2 cache size 12MB on-die for Quad-Core; 6MB per core pair; shared by two cores. Memory Blade: 16GB. All nodes are interconnected using InfiniBand technology in a full-CLOS topology.
- *Intel Nehalem server* with eight 2.8 GHz cores (in two Intel Xeon Processor Quad-Cores, Nehalem – X5560). L3 cache size 8 MB shared by all cores. Memory: 12 GB.

All systems have similar configurations in terms of number of sockets per node and cores per socket. Tests were performed using Intel Fortran compiler using the same compilation options in all cases. The cache sharing scheme in older Intel Xeon processors, although behaving as Quad-core are, in fact, two Dual-core processors put together. Mesh entities are ordered to improve data locality as described in [1].

In the performance tests, initially, we evaluate single node performance and multi-core performance according to the parallel models described in Section 2. Tests are primarily conducted in the coarser mesh (MSH1), for runs using up to 8 intra-node cores. In other words, no network connection was employed in order to reveal intrinsic CPU aspects, such as: cache size, memory sharing, load balance, process placement, etc.... Figure 2 shows the speedup for EdgeCFD, running the parallel schemes presented in section 2, using 2 different systems, SGI Altix ICE 8200 (Figure

2a) and the Nehalem server (Figure 2b). The systems are chosen to show the performance evolution between 2 Xeon family processors, X5355 and X5560 respectively.

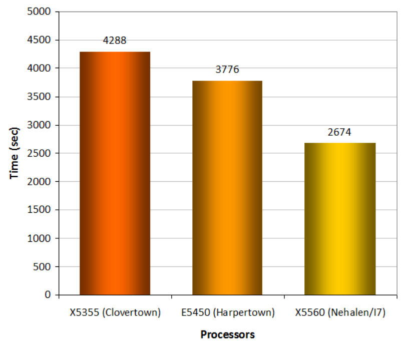


(a) SGI Altix ICE (clovertown) (b) Nehalem server (i7)

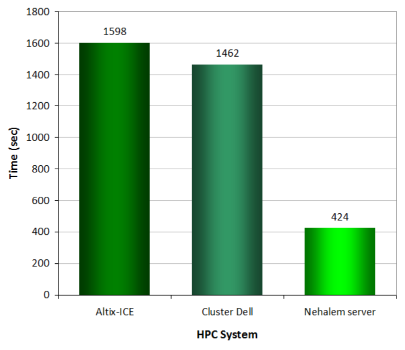
**Fig. 2.** Speedup for two Xeon systems running up to 8 intra-node cores

As can be seen from Figure 2, the overall performance is much better in the newer Xeon processor (Nehalem or i7) than in previous version (Clovertown/Harpertown). Regarding the parallel scheme, the peer-to-peer model resulted in the best performance, as expected, reaching a particularly good speedup in the Nehalem processor. However, for the Clovertown CPU, the poor performance led us to investigate this issue from other aspects, such as: cache size and sharing, node architecture, etc. In both architectures peer-to-peer message passing is clearly superior. Earlier experiments [11] on a HP ProLiant DL145 G3 cluster with 912 cores powered with Opteron 2218 dual core processors and Gigabit network have shown that p2p was also faster than collective communication when using more than one computational node.

Figure 3a show a raw comparison among the benchmark systems when running serial cases (using one core). It clearly shows the performance increase for the Intel Xeon processor between releases from 2006 (Clovertown) until 2009, when Nehalem processor was launched. The wall time reduction, in our tests, reached 37% for processors with 3 years of difference. The performance gains are even more pronounced when we analyze the multi-core case running in a single node (Figure 3b). Other interesting result is the parallel performance shown by Nehalem which is faster than the Cluster Dell and SGI Altix-ICE systems, where older Intel Xeon CPUs are present. It seems that these results are mainly influenced by the cache memory system that Nehalem processor has.



(a) CPU (serial run)



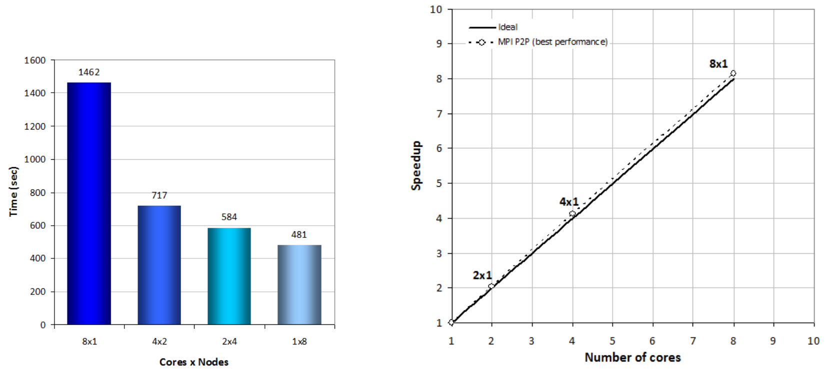
(b) System (8 cores, 1 node)

**Fig. 3.** System comparisons for CPU performance in serial runs (a) and peer-to-peer MPI performance using 8 cores in 1 node (b).

Motivated by the results presented so far, tests were also conducted considering different combinations of cores per nodes. Figure 4a shows the elapsed wall time spent to solve the Rayleigh-Benard problem in parallel (message passing with peer-to-peer scheme), using 8 cores for several arranges of cores × nodes. Tests were performed in Cluster Dell but similar results were also obtained in the SGI Altix ICE 8200.

From Figure 4a, we note that diminishing the number of cores per node, the performance increases substantially, which points out that older Xeon processors suffer when all cores are simultaneously busy. It is important to remember that the main EdgeCFD's kernels (matrix-vector product, stiffness matrix build up and assembly of elements residua) strongly relies in indirect memory addressing operations and are, thus, influenced by how mesh entities are accessed and used during these operations. In EdgeCFD, mesh entities are reordered to makes efficient use of cache memory as explained in details in [1]. However, due to the complexity of the main loops of the software, cache misses are expected even for reordered meshes. To better understand the meshing ordering effect we have also run this problem considering two mesh configurations: original nodal ordering and nodes reordered using Reverse Cuthill Mckee (RCM). In the latter case, edges and elements were ordered in ascending order of edge (element) nodes. All tests were made in a single node. For the first case, it was necessary 24:26 (mm:ss) to solve the Rayleigh-Benard problem on MSH1, while for the reordered, the wall time decreased to 17:57 (mm:ss). The parallel profiling information was obtained using TAU (Tuning Analysis Utilities [9]) and, for the case using one core per node, where all communication is made through InfiniBand network, the time spent in MPI_WaitAll calls was around 3.2% of the total wall time. For the case using all cores available in one node, where all communication is made using memory bus, the largest MPI cost, due to MPI_WaitAll routine, was around 2.4%. This may be an indication of the MPI inability to provide efficient communication in non-homogeneous systems (here memory bus/InfiniBand) as described in [8].

Figure 4b presents the speedup curve obtained with p2p communication pattern and using the best combination of cores per nodes in the Cluster Dell system. Comparing with Figure 2a, which presents the same metric for the SGI Altix ICE 8200, but using only one node, we can conclude that using a large number of cores per node dramatically reduces performance. Note that using one core per node, 12.5% of the theoretical processing power, an ideal speed-up is reached. This also supports the previous argument, because in this case, communication is homogeneous and uses only InfiniBand network.



(a) CPU (serial run) (b) Best speedup - one core per node

**Fig. 4.** Performance impact according to cores x nodes distribution on Cluster Dell

In order to illustrate the impact of the performance issues discussed in previous sections, tests are also conducted for the same problem in the finer mesh (MSH2) described in section 3. For this test, 64 cores are employed to solve 31,140,625 flow equations per nonlinear iteration per time step and 7,843,248 transport equations for each multi-correction iteration per time step. The number of time steps considered was enough to make the initialization process negligible. Two runs with different combinations of cores × node are used. In the first run, all cores of 8 nodes are used in order to exhaust nodes resources. In the second case, only 4 cores are used from 16 nodes, 50% of each node resources. In the first case, it is necessary 01:49:13 (hh:mm:ss) to solve 10 time steps while in the second case, considering only 50% of each node capacity, the walltime substantially decreased to 00:57:05. All runs are performed in the Cluster Dell, which uses Intel Xeon E5450.

## 4 Concluding Remarks

This work presented several performance tests for different versions of the Intel Xeon processor family running EdgeCFD, a stabilized finite element software for solving incompressible flows coupled (or not) to advection-diffusion transport problems. Tests focused in single node multi-core performance showing that past Intel Xeon processors dramatically suffers when large workloads are imposed to a single node. However, the problem seems to be mitigated in the newest Intel Xeon processor (codename Nehalem or commercial name Core I7) by the inclusion of a third level (L3) of shared cache memory. Other important change made by Intel, in its newest Xeon processor, was the construction of a fast linking channel among processors called Quick Path Interconnect, QPI for short. As a consequence, performance dramatically decreases when systems built with older Intel Xeon processors are subjected to large workloads. In the other hand, excellent performances may be reached when placement policies, such as using a smaller number of cores per node, are adopted as shown in Figure 4.

We also investigated message-passing parallelism. We observed that peer-to-peer two-stage information exchange among neighboring processors, using non-blocking send and receive MPI primitives, present the best performance. Experiments also demonstrate the difficulty of MPI to handle heterogeneous communication. Moreover, by setting a suitable MPI process distribution, execution time can be reduced by more than one half as we observe in the large run with the finer mesh on the Cluster Dell. A possible direction to tackle such difficulties is to set-up an architecture aware mesh partition, but this remains to be explored in EdgeCFD.

**Acknowledgements.** This work is partially supported by CNPq, Petrobras, ANP, Dell and Intel.

## References

1. Coutinho, A.L.G.A., Martins, M.A.D., Sydenstricker, R.M., Elias, R.N. Performance comparison of data-reordering algorithms for sparse matrix–vector multiplication in edge-based unstructured grid computations. International Journal for Numerical Methods in Engineering, 66:431–460, 2006.
2. Lins, E.F., Elias, R.N., Guerra, G.M., Rochinha, F.A., Coutinho, A.L.G.A. Edge-based finite element implementation of the residual-based variational multiscale method. International Journal for Numerical Methods in Fluids, 61(1):1-22, 2009.
3. Kessler, R., Nonlinear transition in three-dimensional convection, Journal of Fluid Mechanics, 174:357-379.
4. Sahni, O., Zhou, M., Shepard, M. S., Jansen, K. E., Scalable Implicit Finite Element Solver for Massively Parallel Processing with Demonstration to 160K Cores, Proceedings of the Supercomputing Conference, Portland, OR, USA, 2009
5. Kogge, P., The Challenges of Petascale Architectures, IEEE Computing in Science Engineering, pp 10-16, Nov-Dec 2009.
6. Gropp, W.D., Software for Petascale Computing System, IEEE Computing in Science Engineering, pp 17-21, Nov-Dec 2009.
7. Biswas, R., Proceedings of the 21st International Conference on Parallel Computational Fluid Dynamics, Moffet Field, CA, May 2009.
8. G. Berti, J.L. Traff, What MPI Could (and Cannot) Do for Mesh-Partitioning on Non-homogeneous Networks. In: Bernd Mohr, Jesper Larsson Träff, Joachim Worringen, Jack Dongarra (Eds.): Recent Advances in Parallel Virtual Machine and Message Passing Interface, 13th European PVM/MPI User's Group Meeting, Bonn, Germany, Lecture Notes in Computer Science 4192 Springer 2006.
9. S. Shende and A. D. Malony. The TAU Parallel Performance System International Journal of High Performance Computing Applications, 20(2): 287-311, 2006.

10. Houzeaux, G., Vázquez, M., Aubry, R., Cela, J. M., A Massively Parallel Fractional Step Solver for Incompressible Flows, Journal of Computational Physics, 228:6316-6332, 2009
11. Elias, R.N., Camata, J.J., Paraizo, P., Coutinho, A.L.G.A., Communication and Performance Evaluation of Edge-Based Coupled Fluid Flow and Transport Computations, ECCOMAS, Coupled Problems, Barcelona, 2009