Asociación Argentina de Mecánica Computacional

Mecánica Computacional Vol XXIX, págs. 2985-2997 (artículo completo)
Eduardo Dvorkin, Marcela Goldschmit, Mario Storti (Eds.)
Buenos Aires, Argentina, 15-18 Noviembre 2010

PERFORMANCE COMPARISON OF SCIENTIFIC APPLICATIONS ON LINUX AND WINDOWS HPC SERVER CLUSTERS

Albino A. Aveleda[a], Renato N. Elias[b], José J. Camata[a] and Alvaro L.G.A. Coutinho[a]

[a]*NACAD, High Performance Computing Center, Federal University of Rio de Janeiro, P.O.Box 68506, 21945-970, Rio de Janeiro, RJ, Brazil, {bino, camata, alvaro}@nacad.ufrj.br, http://www.nacad.ufrj.br*

[b]*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*

Keywords: HPC, Cluster, Performance, Benchmark, Parallel Computing.

Abstract. The TOP500 project ranks the 500th. most powerful computer systems in the world. This rank is based on High-Performance LINPACK (HPL), a portable implementation of the LINPACK benchmark for distributed-memory computers. According to the TOP500's list published in June, 2010, most of the systems are Linux based clusters. However, the number of Windows based systems tends to grow in the next lists. In this paper the performance of a Linux and a Windows HPC Server cluster using the same hardware and scientific applications is evaluated. To assess performance, the HPL and NAS Parallel Benchmarks (NPB), as well as a real-world multi-physics application, named EdgeCFD, are used. This application is developed at NACAD-COPPE/UFRJ and it is an implicit edge-based coupled fluid flow and transport solver for large-scale problems optimized for modern clusters. EdgeCFD adopts peer-to-peer non-blocking communication pattern among processes. Tests were conducted for three MPI distributions MVAPICH2 (based on MPICH2 with Infiniband support), OpenMPI (High Performance Message Passing Library) and MS-MPI (Microsoft MPI based on MPICH2) on all benchmarks and applications of this study. All performance measurements show that Windows HPC Server is a viable option.

1 INTRODUCTION

In recent years, modern parallel supercomputers are allowing engineers, scientists and industries to solve a wide range of complex, real world problems at scales that were considered impossible few years ago. National laboratories and universities want even more powerful machines to solve difficult large-scale problems with greater fidelity. On the industry, as in aerospace and automotive engineering companies, better performing systems allow to grow from running component-level jobs (such as analyzing the stress on an engine block) to conducting more complex, multi-parameter studies. It is known that the performance of these systems depends on several factors including processors, memory system and local and global networks (Saini et al; 2009a,b). However, we have observed that operational system influence on these systems has a limited discussion.

Since 1993 and twice a year, a group of scientists in Europe and the United States release a list of the world's 500 most powerful computing systems. The TOP500 (TOP500, 2010) list is the most prestigious ranking of its kind, with vendors and users leveraging favorable rankings to promote their work. Over the years, the TOP500 list has served as a tool for documenting and analyzing technological and architectural changes in the HPC arena, including the tremendous growth in performance and the increase in the number of computer clusters. The number of clusters in the TOP500 list has increased dramatically over the 15 years since the list has been compiled. In the most recent list, released in June 2010, Linux is by far the most frequently used operational system in high performance computing. Considering the numbers: 405 machines (or 91%) run some Linux distribution, 22 run UNIX (4.5%) and five run Windows (1%). .

Although Linux has a long history in HPC, there is a concern about the future. Microsoft continues to make advancements with its Windows HPC server. In addition, Windows is by far the most popular operational system for personal computers, which potentially makes it a comfortable platform to new HPC users. In light of this, we present an evaluation of the Linux and Windows HPC server environments using MPI-based benchmarks such as High-Performance LINPACK (Dongarra et al, 2003), NAS Parallel Benchmark (Bailey et al, 1994 and Bailey et al, 1994), and a real world application, EdgeCFD (Elias and Coutinho, 2007; Elias et al, 2009). For a general overview of computer benchmarking, please see Hockney (1995).

The High-Performance LINPACK (HPL) solves a dense linear system in double precision on distributed-memory computers (Dongarra et al, 2003). Although, there is much debate about how relevant the HPL benchmark is to the industry, it remains an excellent “burn in” test for very complex HPC systems. HPL is a good tool for validating a system: it works as a check besides stressing the system more than typical applications would. Keeping this in mind, the HPL benchmark is used in this work to validate the system. The NAS Parallel Benchmarks (NPB) (Bailey et al, 1994) suite, developed by NASA, is based on applications as the Conjugate Gradient Method, Fast Fourier Transform and others. Finally, a real-world multi-physics application, named EdgeCFD (Elias and Coutinho, 2007; Elias et al, 2009) is also used in the performance tests. This application is developed at the High Performance Computing Center at COPPE/Federal University of Rio de Janeiro (NACAD-COPPE/UFRJ).

In the present work, both operational systems were installed in the same hardware and performance studies between Linux and Windows HPC server are summarized by:

- Overall performance evaluation using the benchmarks applications;
- Performance evaluation of different MPI implementations;
- Performance analysis of MPI process placement in multi-core processors.

The remainder of this paper is organized as follows: Section 2 details the computing system architectures; Section 3 describes the HPL and the NPB benchmarks suite, while Section 4 describes the real-world application; Section 5 presents and analyzes results from running these benchmarks and the application; finally Section 6 contains a summary and conclusions of the study and future work.

2 COMPUTING SYSTEM

This section briefly describes the Dell PowerEdge system and software used in this work.

2.1 Hardware

The Dell PowerEdge PE2950III system is composed by: two heap nodes and 16 computing nodes. Each heap node has two 3.0 GHz Intel Quad core Xeon E5450 and 12MB shared cache L2 processors. Each computing node is composed by a blade M600 with two Intel 3.0 GHz quad-core Xeon EX5450 with 12 MB L2 shared cache; 6MB per core pair. The 1333 MHz Front Side Bus (FSB), which transports data among memory and the four cores, is a quad-pumped bus running off a 333 MHz system clock making 10.66 GBytes per second data transfer rates possible. The blades are interconnected by a Double Data Rate (DDR) InfiniBand (IB) network with a peak bidirectional bandwidth of 20GB/s. Each node has 16GB of memory RAM; 2GB per core. Finally, a PowerVault MD3000 and MD1000 form the storage system. The machine specifications are summarized in Table 1.

Head node	Dell PowerEdge PE2950III with two processors Intel Quad core Xeon EX5450, 3.0GHz, 2x6MB Cache 1333MHz FSB.
Computing node	Blade PE M600 with two processors Intel Xeon E5450, 2x6MB, 3.0GHz, 1333FSB
Storage	PowerVault MD3000 and MD1000
Private network	Gigabits Ethernet (1 Gb/s)
Application network	Infiniband DDR (20 Gb/s)

Table 1: Cluster Hardware

In the present paper we use the same computing node hardware on the Linux and the Windows HPC based Cluster. Thus, performance differences between the Linux and Windows HPC environments should not be attributed to the hardware. In addition, each head node on the cluster is assigned by each operational system; one for the Linux based cluster and the other for the Windows HPC based cluster. Considering all computing nodes, the cluster has 128 cores.

2.2 Software

The Linux Cluster uses RedHat Enterprise Linux distribution (version 5.4). The system's IB network uses Open Fabrics Enterprise Distribution (OFED) software. All applications were compiled by Intel compilers (version 11.1). Control over batch jobs and schedule them over the computer nodes are provided by Torque. The software used on the Linux Cluster is summarized in Table 2.

Operational System	RedHat Enterprise Linux 5.4 (RHEL 5.4)
Infiniband	OFED (OpenFabrics Enterprise Distribution)
MPI Libraries	MVAPICH2 – v. 1.4.1 OpenMPI – v. 1.4.1
Compilers	Intel Compiler – v. 11.1
Others	Torque – v.2.4.8 MOAB- v. 5.4.1 Ganglia – v 3.1.2

Table 2: Software used for the Linux Cluster

For the Windows HPC Cluster, it is used Windows HPC server 2008 SP2 with HPC pack SP1 (MS-HPC, 2010). The Windows HPC Pack provides a complete, integrated set of middleware and tools for running high performance computing clusters. For instance, it includes the job scheduling system, system management and network and message passing interface (MPI) (MS-HPC, 2010). The Job Scheduler supports a Service-Oriented Architecture (SOA) mode that provides access to interactive applications through the Windows Communication Foundation (WCF). In addition, core Job Scheduler functionality is exposed using the Open Grid Forum's HPC Basic Profile Web service. The IB network software is Mellanox WinOF VPI. Again, the compilers are provided by Intel vendors integrated into Microsoft Visual Studio 2008. Table 3 reports a short description of system.

Operational System	Windows HPC Server 2008 SP2 + HPC Pack SP1
Infiniband	Mellanox WinOF VPI v. 2.1.1
MPI Libraries	MS-MPI
Compilers	Intel Compiler – v. 11.1 Visual Studio 2008

Table 3: Software used for the Windows HPC Server 2008

The benchmarks and the real-world application use the Message Passing Interface (MPI) library. The MPI interface is meant to provide essential virtual topology, synchronization, and communication functionality between a set of processes (which have been mapped to nodes/servers/computer instances) in a language-independent way. In this work three MPI libraries are available:

- MVAPICH2 (2010): an open source MPI-2 implementation based on MPICH2 (2010) supporting Infiniband interface.
- OpenMPI (2010): an open source MPI-2 implementation that is developed and maintained by a consortium of academic, research, and industry partners.
- MS-MPI (2010): Microsoft MPI is a proprietary distribution by Microsoft. This distribution is also based on MPICH2.

The MVAPICH2 and the OpenMPI are used on the Linux based cluster while the MS-MPI is used on the Windows HPC Server based cluster only.

3 BENCHMARKS AND SCIENTIFIC APPLICATION

Our evaluation approach recognizes that the application performance is the ultimate measure of system capability; however, understanding an application's interaction with a computing system requires a detailed understanding of the performance for each part of the system. Keeping this in mind, the High-Performance LINPACK (HPL) (Dongarra et al, 2003) benchmark that measures processor, memory, and network performance of the architecture at the subsystem level, is used to validate the system. Then, insights gained from the HPL benchmark are employed to guide and interpret performance analysis of the NPBs in a full-scale application.

3.1 High Performance Linpack

High Performance Linpack (HPL) (Dongarra et al, 2003) is a portable implementation of the Linpack Benchmark (Dongarra et al, 1979). It can be seen as a software package that generates and solves a dense linear system of equation of order n:

$$Ax = b;\ A \in R^{nxn};\ \ x,b \in R^{n} \tag{1}$$

by performing LU factorizations. In addition, the package checks and times the solution process on distributed-memory computers. The package uses 64-bit floating-point arithmetic and portable routines for linear algebra operations and message passing. The Figure 1 illustrates the two-dimensional block-cyclic data distribution used by HPL. The matrix is partitioned in NB x NB submatrices (i.e. blocks). Blocks are distributed to (MPI) processes where each process performs factorization on P × Q blocks assigned to it.

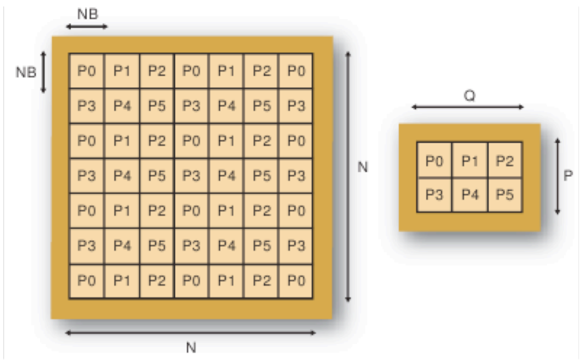


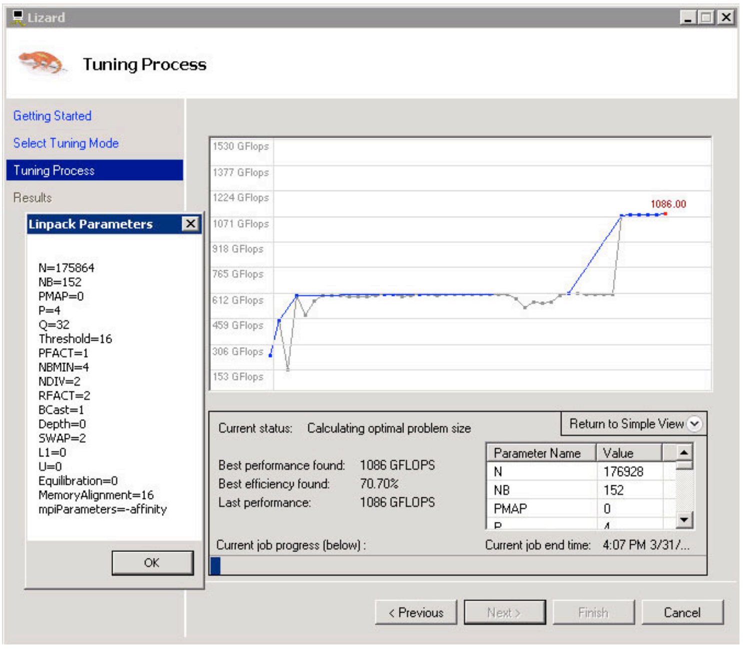
Figure 1: HPL two-dimensional block-cyclic data distribution

The performance of HPL is improved by tuning the following parameters:

- *Broadcast parameter:* LU factorizations are performed iteratively. At the end of each iteration, resulted data (i.e. matrix columns) are broadcasted to every processor. There are many possible broadcast algorithms implemented on HPL.
- *Block Size NB:* used for the data distribution as well as for the computational granularity.
- *N, P e Q:* To ensure that each process is allocated the same amount of data, the value of N should be divisible by P, Q and NB. The value of P and Q depends of the network interconnection. Assuming a topology, it is recommended that P and Q should be approximately equal, with Q slightly larger than P and P taken as a power of 2.

Finding the best computational performance and efficiency involves finding the better parameters for our system. In some cases, this is a trial-and-error process. The Microsoft HPC 2008 Tool Pack, installed in our system, has a tool called Lizart that helps determine the computational performance and efficiency that can be achieved by the Windows HPC Server 2008 cluster. It calculates and reports a performance peak value for the HPC cluster in billions of floating-point operations per second (GFLOPS), and a percentage value for the efficiency that is achieved at peak performance. Lizard calculates the performance and efficiency of the

HPC cluster by automatically running the LINPACK Benchmark several times and tuning the parameters that governs the LINPACK Benchmark solution between runs. Figure 2 shows the HPL algorithm performance evolution. To further optimize performance, all non-essential services are turned-off to run HPL. Eventually, Lizard is able to provide optimal performance measurements of the HPC cluster, which is available as a report at the end of the tuning process.



```
N=175864
NB=152
PMAP=0
P=4
Q=32
Threshold=16
PFACT=1
NBMIN=4
NDIV=2
RFACT=2
BCast=1
Depth=0
SWAP=2
L1=0
U=0
Equilibration=0
MemoryAlignment=16
mpiParameters=-affinity
```

Current status: Calculating optimal problem size

Best performance found: 1086 GFLOPS
Best efficiency found: 70.70%
Last performance: 1086 GFLOPS

Parameter Name	Value
N	176928
NB	152
PMAP	0
P	4

Current job progress (below) :
Current job end time: 4:07 PM 3/31/...

Figure 2: Evolution of HPL algorithm performance.

This tool was used to compare the performance of Gigabit Ethernet and Infiniband interconnects. MS-MPI implementation was used for message exchanges. For each network, three different cases with 4, 8 and 16 compute nodes are used, totalizing 32, 64 and 128 cores respectively. The results are shown in Table 4 and the performance plot is shown in Figure 3. We clearly see that InfiniBand yields the best performance.

Cores	**32**	**64**	**128**
Theoretical Peak Performance	384.0	768.0	1536.0
Ethernet Network	277.6	528.8	1092.0
Infiniband Network	316.0	641.1	1289.0
Ethernet (%) Efficiency	72.29	68.85	71.09
Infiniband (%) Efficiency	82.29	83.48	83.92

Table 4: HPL results (GFlop/s)

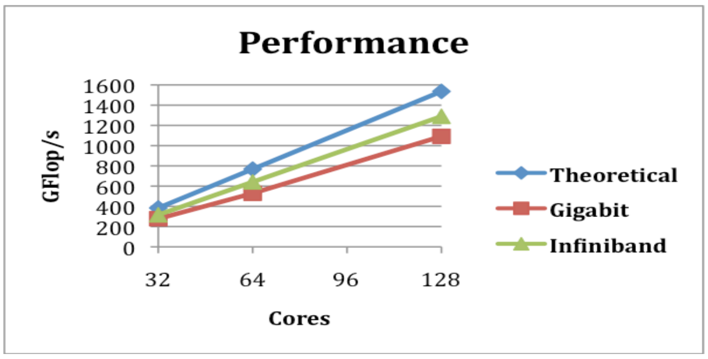


Figure 3: Performance for different interconnects.

3.2 NAS Parallel Benchmarks

The NAS Parallel Benchmark (Bailey et al, 1994 and Bailey et al, 1994) (NPB) suite is composed of well-known codes for testing the capabilities of parallel computers and parallelization tools. The benchmarks were derived from Computational Fluid Dynamics (CFD) codes and are widely recognized as a standard indicator of parallel computer performance. The original NPB suite contains eight benchmarks, five kernels (CG - Conjugate Gradient, EP - Embarrassingly Parallel, MG - Multigrid, FT - Fast Fourier Transform and IS - Integer Sort) and three compact applications (BT - Block Tridiagonal, LU - Lower-Upper symmetric Gauss Seidel and SP - Scalar Pentadiagonal). The MPI version of the NPB suite (NPB3.3.1 distribution) is used in this study. Each application has five problems sizes, dubbed Classes A, B, C, D and E, being Class E the biggest. The Intel Compiler is used on the Linux based cluster and on the Windows HPC based cluster because some codes were written in Fortran Language and Microsoft does not have support for this language. Thus, the only difference in all tests is the operational system. The compiler options used on both platforms are presented in Table 5.

Platform	Compiler Option
Linux	–O3 –axSSE2
Windows HPC Server 2008	/O3 /QaxSSE2

Table 5: Intel compiler options.

3.3 Scientific Application: EdgeCFD

EdgeCFD is a finite element system for complex fluid-structure interactions designed for offshore hydrodynamics. Green-water decks and wave impact on floating devices are a few examples of problems that can be solved with EdgeCFD. The fluid component of the software consists of an incompressible finite element edge-based flow solver able to treat free-surface flow problems by a VOF approach (Elias and Coutinho, 2007; Elias et al, 2009) EdgeCFD is based on the Streamline-Upwind Petrov-Galerkin/Pressure-Stabilized Petrov-Galerkin with the Least Squares Incompressibility Constraint (SUPG/PSPG/LSIC) finite element formulation. Turbulence in EdgeCFD has been treated by a Smagorinsky model. More recently, the Residual-Based Variational Multiscale method to turbulence has been incorporated into EdgeCFD with success (Lins et al, 2009; Lins et al, 2010). The fluid-structure problem is treated by the Arbitrary Lagrangian Eulerian (ALE) formulation in which parts of the computational mesh move attached to material particles in a Lagrangian description, while other parts of the mesh remain fixed in space, in an Eulerian description. In between these parts, there is a transition region, connecting them, where the mesh nodes move arbitrarily, so to say, irrespective of material particles motion.

In EdgeCFD special attention is given to the description of the hydrodynamics of an immersed body. The mesh updating scheme is based on the node repositioning in the neighborhood of a moving body in three-dimensions and is accomplished by the solution of a scalar diffusion problem in each spatial coordinate. Problem statement for mesh updating process includes a variable and adaptive local diffusion coefficient, dependent on a relative geometric quality index, computed for each element throughout the mesh. Boundary conditions involve the motion of the immersed body´s surface, i.e., the fluid-structure interface, taken as the Lagrangian portion of the domain in the overall problem. Time integration in EdgeCFD is performed with a predictor-multicorrector algorithm with adaptive timestepping by a Proportional-Integral-Derivative (PID) controller (Valli et al, 2009). Within the flow solution loop, the multi-correction steps correspond to the Inexact-Newton method. In this method the tolerance of the linear solver is adapted according to the history of the solution residua. As a linear solver, EdgeCFD employs the Preconditioned Generalized Minimal Residual Method (GMRES) for both flow and VOF equations. Mesh movement equations are solved by preconditioned conjugate gradients. Most of the computational effort spent in the solution phase is devoted to matrix-vector products. In order to compute such operations more efficiently, EdgeCFD uses an edge-based data structure. This data structure, when applied to problems as those described in this work, is able to reduce indirect memory access, memory requirements to hold the coefficients of the stiffness matrices and the number of floating point operations when compared to other traditional data structures such as element-by-element (EBE) or compressed sparse row (CSR). EdgeCFD computational kernel is a full parallel Fortran90 finite element code. The computations are performed in parallel using a distributed memory paradigm through the message passing interface library (MPI). Collective and point-to-point communication between subdomains are currently supported. The parallel partitions are generated by Metis/Parmetis library, while the information regarding the edges of the computational grid is obtained from the EdgePack library. EdgePack also reorders nodes, edges and elements to improve data locality, exploiting efficiently the memory hierarchy of current processors. EdgeCFD's Graphical User Interface is integrated within ParaView, which supports several mesh formats, including ICEM-CFD, CFX and EnSight. The parallel nature of data generated by EdgeCFD allows full use of advanced parallel visualization capabilities in ParaView.

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 (Kessler, 1985). 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 as shown in Figure 4. The fluid properties are set to result in Rayleigh and Prandt numbers of 30,000 and 0.71 respectively. For the performance test we use a mesh formed by 178,605 tetrahedra elements and 39,688 nodes (MSH1). 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}.

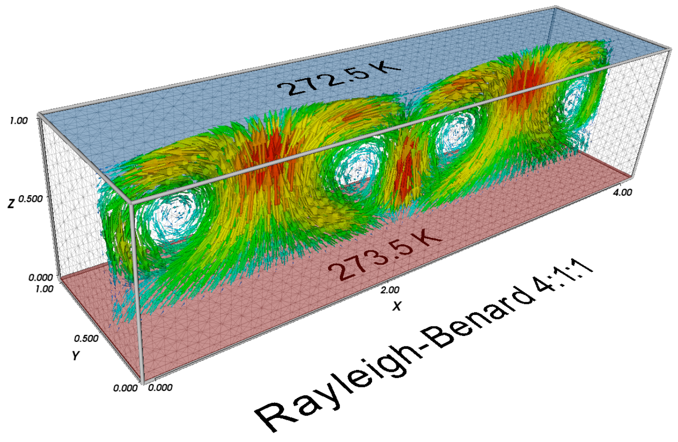


Figure 4: Model multi-physics problem: Problem description and solution at steady state

4 RESULTS

In this section, performance results of selected NPBs benchmarks and the multi-physics application code are presented. To mimic a fully operational environment, all tests are done in production mode, i.e., all services are running.

4.1 NAS Parallel Benchmarks

The results shown below present the performance ratio for six benchmarks (IS, EP, CG, FT, MG and LU) of the NPB-MPI. Figure 5 shows the performance in million operations per second (Mflop/s) on 128 cores for these benchmarks using the NPB problem size Class C. We compare the results with three MPI implementation: MVAPICH2 and OpenMPI on Linux and, MS-MPI on Windows HPC server. MVAPICH2 has the best performance in three of these applications (IS, EP and LU). MS-MPI was better in FT and MG while the OpenMPI has better performance in CG. However, the performance differences were not significant in the most of cases.

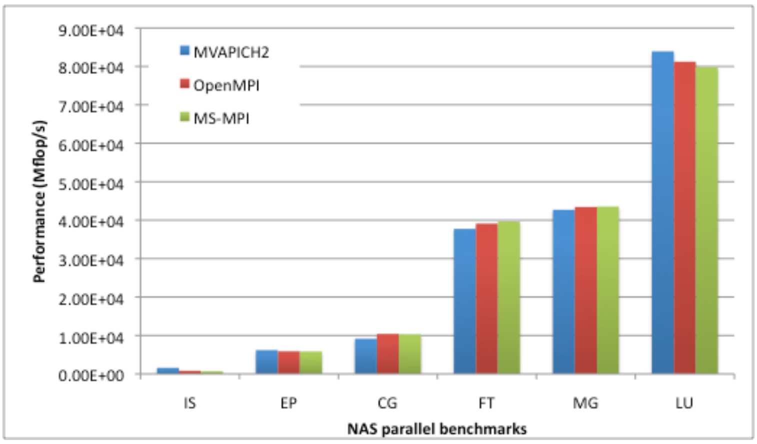
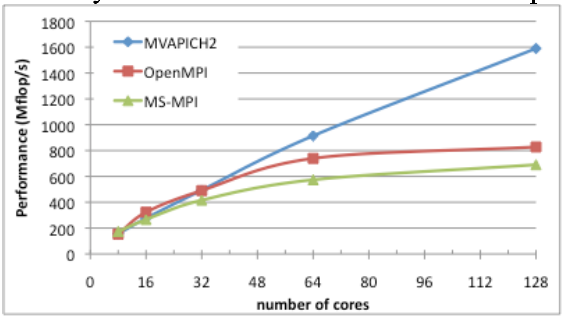
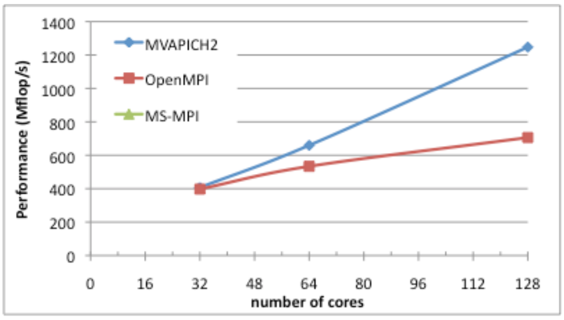


Figure 5: NPB-MPI Class C benchmarks on 128 cores

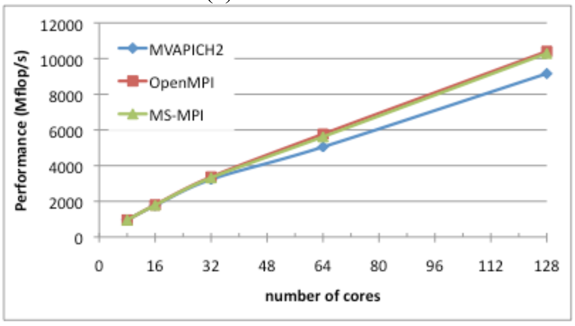
In order to better understand the performance issues on Linux and Windows HPC server, new tests were performed using different arranges of core counts. Figure 6 displays the performance for IS, CG, MG and LU NPB benchmark on 8, 16, 32, 64 and 128 cores. In addition, two problem sizes classes C and D are used. Not all results are obtained because in some cases problem size is bigger than the resources available on the compute nodes, for instance, IS class D do not run on 8 and 16 cores. The results show that the biggest difference between the three MPI distributions is in the IS (Integer Sort) application. This performance difference is not important in the present context, because the kernel of standard science or engineering application is 64-bit floating-point arithmetic. The MVAPICH2 on the Linux cluster system is the MPI distribution that presents a linear behavior in almost all cases.



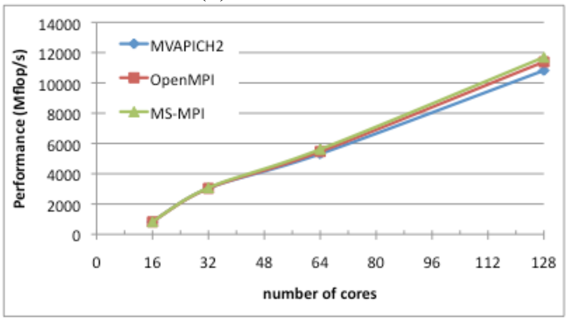
(a) IS - Class C



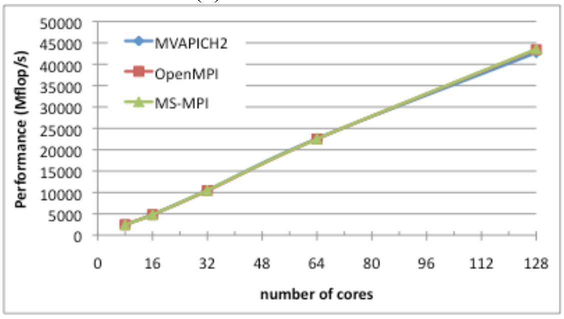
(b) IS – Class D



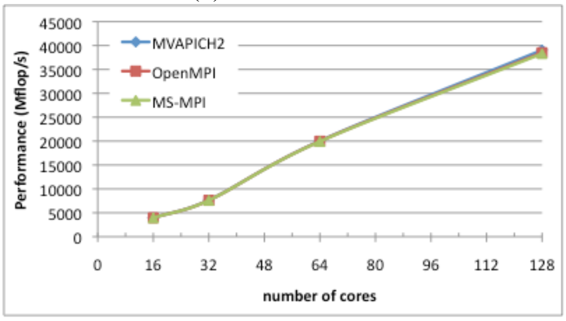
(c) CG - Class C



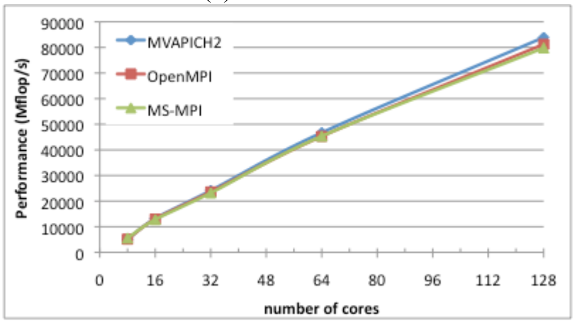
(d) CG – Class D



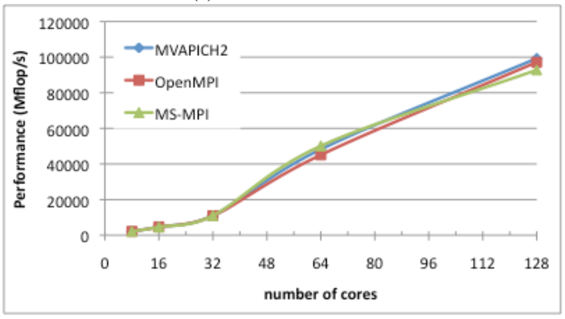
(e) MG-Class C



(f) MG – Class D



(g) LU – Class C



(h) – Class D

Figure 6: NPB-MPI Classes C and D on several cores counts

4.2 Multi-physics Application

This section discusses EdgeCFD's performance on the multi-core nodes considering the two operational systems. We evaluate different combinations of cores per nodes, that is, the effect of process placement in MPI communication. This issue has been studied recently on Linux systems it was observed that process placement has a strong influence on performance (Elias et al, 2010, Diamond et al, 2010). Figure 7 shows the elapsed wall time spent to solve the Rayleigh-Benard problem in parallel (message passing with point-to-point scheme as shown in Sahini et al (2009)), using 8 cores for several arrangement of cores per compute nodes. The code was compiled with Intel ifort version 11.1 and optimized by flag –O3. Note that, diminishing the number of cores per node, performance increases substantially. We can see that both systems present similar behavior, indicating that Microsoft HPC Server does not negatively affect this highly optimized application nor solved the MPI process placement problem identified earlier.

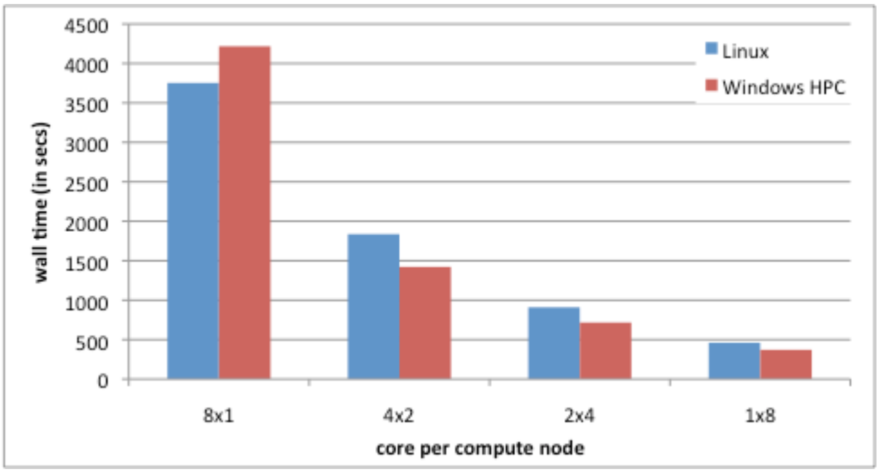


Figure 7: EdgeCFD Performance

5 CONCLUSIONS

In this work we present a performance evaluation of a 128 cores cluster running Windows HPC Server 2008. Tuning HPL using Microsoft Lizart tool we achieved 1.29 TFlop/s with efficiency of 83.9% when using the Infiniband network. Next we run the NPB benchmarks examining the effects of three MPI distributions on both the Linux and the Windows systems. We observe that except for the IS application on Class C, the three MPI distributions are equivalent and platform independent. None of them are better than other distributions in all tests. The difference between them in some cases is 10%, but in general this difference oscillates around 5%. The MS-MPI and MVAPICH2 that are based on MPICH2 presented similar behavior. The following step of the present study is to evaluate MPI process placement on both operational systems using a multi-physics model application. We have found that performance is equivalent on both operational systems and diminishing the number of cores per node improves wall time on both cases. Therefore, our experiments suggest that it is feasible to use Windows HPC server tools on clusters for scientific applications.

Acknowledgements

This work is partially supported by Dell, Intel, ANP and Adaptive Computing.

REFERENCES

Bailey D., Barszcz E., Barton J., Browrning D., Carter R., Dagum L., Fatoohi R., Frederickson P., Lasinski T., Schreiber R., Simon H., Venkatakrisnan V., and Weeratunga S., The NAS parallel benchmarks, *International Journal for Supercomputer Applications*, 5(3): 63-73, 1991.

Bailey D., Barszcz E., Barton J., Browrning D., Carter R., Dagum L., Fatoohi R., Fineberg S., Frederickson P., Lasinski T., Schreiber R., Simon H., Venkatakrisnan V., and Weeratunga S., The NAS parallel benchmarks, RNR Technical Report, RNR-94-007, March 1994.

Diamond, J., Kim, B. D., Burtscher, M., Keckler, S., Pingali, K., Browne, J., Multicore optimization for ranger, *In TeraGrid 2009 Conference*, June 22-25, Arlington, Virginia.

Dongarra J., Bunch J., Moler C., and Stewart G. W., LINPACK User's Guide. SIAM: Philadelphia, PA, 1979.

Dongarra J., Luszczek P., and Petitet A., The Linpack Benchmark: past, present and future, *Concurrency and Computation: practice and experience*, 15:803-820, 2003.

Elias, R. N., Coutinho, A.L.G.A., "Stabilized Edge-based Finite Element Simulation of Free-Surface Flows", International Journal for Numerical Methods in Fluids, v. 54, p. 965-993, 2007.

Elias, R. N., Gonçalves Jr., M. A., Coutinho, A L. G. A., Esperança, P. T. T., Martins, M. A. D., Ferreira, M. D. A. S., "Computational Techniques for Stabilized Edge-Based Finite Element Simulation of Nonlinear Free-Surface Flows", Journal of Offshore Mechanics and Arctic Engineering, v. 131, p. 041103, 2009.

Elias, R. N., Camata J. J., Aveleda A., Coutinho A. L. G. A., Evaluation of message passing communication patterns in finite element solution of coupled problems. *In proceedings on 9th international meeting high performance computing for computational science VECPAR'10*, Berkeley CA, 2010.

Hockney R. W., "The Science of Computer Benchmarking", SIAM, 1995.

Kessler, R., "Nonlinear transition in three-dimensional convection", 1985, Journal of Fluid Mechanics, 174:357-379.

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, v. 61, p. 1-22, 2009.

Lins, E. F., Elias, R. N., Rochinha, F. A., Coutinho, A. L. G. A. , "Residual-based variational multiscale simulation of free surface flows", Computational Mechanics, v. 46, p. 545-557, 2010.

MPICH2, High-Performance and Widely Portable MPI, http://www.mcs.anl.gov/research/projects/mpich2, 2010.

MS-HPC, Microsoft HPC server 2008, http://www.microsoft.com/hpc/en/us/default.aspx, visited in 13-09-2010.

MS-MPI, Microsoft MPI, http://msdn.microsoft.com/en-us/library/bb524831(VS.85).aspx, 2010.

MVAPICH2, MPI over Infiniband, 10GigE and RoCE, http://mvapich.cse.ohio-state.edu, 2010.

OpenMPI, Open Source High Performance Computing http://www.open-mpi.org, 2010.

TOP 500; TOP500 Supercomputing Sites, http://www.top500.org, 2010.

Saini S., Talcott D., Jespersen D., Djomehri J, Jin H., and Biswas R., Scientific Application-Based Performance Comparison of SGI Altix 4700, IBM POWER5+, and SGI ICE 8200 Supercomputers, NAS Technical Report NAS-09-001, February, 2009a.

Saini, S., Naraikin, A. Biswas, R. Barkai, D. and Sandshom, T. Early performance evaluation of a Nehalem cluster using scientific and engineering applications, *NAS Technical Report NAS-09-005*, November 2009b.

Sahini O, Zhou M., Shephard M. S. and Jansen K., Scalable Implicit Finite Element Solver for Massively Parallel Processing with Demonstration to 160K cores, *Proceedings of the ACM/IEEE Conference on High Performance Computing, SC 2009*, November 14-20, 2009, Portland, Oregon, USA.

Valli, A. M. P. ; Elias, R. N. ; Carey, G. F. ; Coutinho, A. L. G. A. . PID adaptive control of incremental and arclength continuation in nonlinear applications. *International Journal for Numerical Methods in Fluids*, v. 61, p. 1181-1200, 2009.