AMD 6234 Interlagos *vs.* Intel E5-2680 Sandy Bridge. Benchmark of different computational codes.

Simone Giusepponi, Agostino Funel, Fiorenzo Ambrosino, Guido Guarnieri, Giovanni Bracco
ENEA UTICT-HPC

1 Introduction

In this paper we report results concerning the performances of the AMD 6234 Interlagos 2.4 GHz and Intel E5-2680 Sandy Bridge 2.7 GHz processors by running standard application benchmarks. The tests were conducted on an experimental node with 24 cores, 64 GB RAM and 16 cores, 32 GB RAM for AMD and Intel processors, respectively. The Sandy Bridge processor supports Hyper Threading (HT) [1] technology which makes a single physical processor appear as two logical processors. Thus by enabling HT the Sandy Bridge is seen by the OS as a node with 32 cores. With HT each logical processor maintains a complete set of the architecture state but share all other resources on the physical processors (caches, execution units, branch predictors, control logic and buses). Instead of using a single benchmark we have considered several applications which refer to actual usage cases as CFD (Computational Fluid Dynamics) and MD (Molecular Dynamics). We also tested the HPL (High Performance Linpack) code which is a standard benchmark used in the HPC (High Performance Computing) community. In the case of CFD the open source code OpenFOAM and the commercial code Ansys Fluent have been tested. For the MD simulation we used the CPMD code. We also tested both architectures with Intel MPI Alltoall benchmark. Usually this benchmark is used to provide a measure of network performances testing a set of MPI functions; in this case it was used to have a measure of shared memory access performances.

2 HPL benchmark

The HPL [2,3] is a benchmark widely used in the HPC community for ranking supercomputers according to their computing power in terms of floating point operations per second (flops). The flops rate is evaluated by solving in parallel a dense linear system of order n:

$$\mathrm{A}\cdot\mathrm{X}=\mathrm{B},\quad A\in\Re^{n\times n},\quad X,B\in\Re^{n}$$

using LU factorization [4-6] with row partial pivoting of the n by n+1 coefficient matrix:

$$P[A,B]=[[L,U],Y].$$

P is the permutation matrix representing the row pivoting and L is the lower triangular matrix. P and L are applied to B as the factorization progresses. The solution is obtained by solving the upper triangular system:

$$U\cdot\mathrm{X}=Y.$$

The factorization requires $2n^3/3 - n^2/2$ operations and the solving phase $2n^2$ thus if t_s is the time to solution, the theoretical peak performance p_{th} (in Gflops) is given by:

$$p_{th}=\frac{\frac{2n^3}{3}+\frac{3n^2}{2}}{t_s}\cdot 10^{-9}.$$

Under very general assumptions it can be shown that the HPL algorithm makes the leading term of time to solution t_s of order $O(n^3)$. The time spent for communication is of order $O(n^2)$. The HPL benchmark scales very well with the size n and the scaling does not depend strongly on the communication volume.

2.1 HPL results

The HPL benchmark was compiled with the Open64 (ver. 4.5.1) compiler and ACML 64 bit (ver. 5.0.0) libraries with fma4 support for AMD Interlagos, and with the Intel compiler suite (ver. 12.1.4) and MKL libraries for the Intel Sandy Bridge. We also made a compilation with the GCC compiler (ver. 4.6.3) for both AMD and Intel processors. In all cases the Open MPI 1.5.4 libraries were used for parallel runs. The compiler flags used for optimization are shown in Table 1.

Open64 (ver. 4.5.1)	-O3 -OPT:Ofast -fno-math-errno -ffast-math march=bdver2 -mfma4 -mavx
Intel mpicc (ver. 12.1.4)	-fomit-frame-pointer -unroll-aggressive -O3 -simd –msse3
GCC (ver. 4.6.3)	-fomit-frame-pointer -O3 -funroll-loops -mavx -mfma4

Table 1: Optimization flags used for the HPL benchmark.

In Table 2 are reported the best results for serial, 8 and 24 cores. We see that, considering the same clock frequency normalization, the Intel Sandy Bridge processor achieves in the serial (parallel) case a higher rate of about 30% (18%). Also the amount of used node RAM is higher for Intel Sandy Bridge. The same behavior has been observed by using the same compiler and libraries (GCC + ACML) for runs involving all node cores of both processors. Also for HPL with HT on (24 cores case) the performance of Sandy Bridge is 18% higher than Interlagos in spite of using 30% more memory with respect the total RAM of the node.

Best rate (Gflops) The percentage refers to the used node RAM		
Cores	**AMD 6234 Interlagos 2.4 GHz, 64 GB RAM (Open64+ACML)**	**Intel E5-2680 Sandy Bridge 2.7 GHz, 32 GB RAM (Intel +MKL)**
1	16.4 (13.4) 20%	26.3 (19.5) [23.4] 40%
8	59.4 (49.5) 61%	182.1 (134.9) [161.9] 90%
24	170.6 (142.5) 61%	234.6 (173.8) [208.5](*) 90%

Table 2: HPL best results obtained on AMD Interlagos and Intel Sandy Bridge processors. The results in round [square] brackets are normalized to 2.0 [2.4] GHz clock frequency. (*) means HT switched on.

3 OpenFOAM code

OpenFOAM [7,8] is a free, open source toolbox for CFD. The software has many features for solving anything from complex fluid flows, including chemical reactions, turbulence, and heat transfer to solid dynamics and electromagnetics. OpenFOAM supports parallel computations and is capable of running on very large HPC clusters. The code is written in C++ and can be extended by users who need to write their own solvers and libraries. The core technology includes numerical methods, linear and ODE (Ordinary Differential Equations) system solvers and dynamic mesh functionalities.

3.1 Benchmark description

We present a case test which has been performed to analyse the performances of OpenFOAM on Intel Sandy Bridge and AMD Interlagos processors. We made a 64 bit compilation for Linux on x86_64 systems with the GCC (ver. 4.6.3) compiler. The Open MPI 1.5.4 libraries were used for parallel runs. The case is well described in the OpenFOAM tutorial. We studied a 2 dimensional multiphase problem of simplified dam break. The feature of the problem is a transient flow of two fluids separated by a sharp interface. The OpenFOAM solver uses a specie transport equation to determine the relative volume fraction of the two phases. The geometry and the initial setup are shown in Fig. 1.

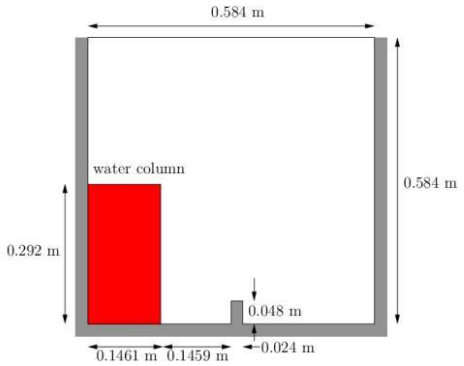

Figure 1: The geometry and initial set up of the OpenFOAM benchmark. The solver uses an incompressible transport Newtonian model and a laminar approximation for turbulence.

A column of water at rest is located behind a membrane on the left side of a tank. At time $t = 0$ the membrane is removed and the column of water collapses. During the collapse, the water impacts an obstacle at the bottom of the tank and creates a complicated flow structure, including several captured pockets of air. The computational domain consists of 123200 cells.

Execution Time (s)		
Cores	**AMD 6234 Interlagos 2.4 GHz, 64 GB RAM**	**Intel E5-2680 Sandy Bridge 2.7 GHz, 32 GB RAM**
1	45069.0 (54082.8)	19069.9 (25744.4) [21453.6]
24	2718.5 (3262.2)	1923.0 (2596.0) [2163.4](*)

Table 3: Execution time for the OpenFOAM case test on AMD Interlagos and Intel Sandy Bridge processors. The results in round [square] brackets are normalized to 2.0 [2.4] GHz clock frequency. (*) means HT switched on.

3.2 OpenFOAM results

The results are reported in Table 3. We see that in the case of a serial run the Intel Sandy Bridge is about twice faster than AMD Interlagos. This is due to the fact that two cores of the AMD Interlagos processor share the same FPU. The discrepancy, however, is less significant with parallel runs and with 24 cores. This may be due to the HT switched on. However, this effect does not impact too much and the Sandy Bridge still takes about 20% less time than AMD Interlagos for completing the run.

4 Ansys Fluent code

ANSYS Fluent software contains the broad physical modeling capabilities needed to model flow, turbulence, heat transfer, and reactions for industrial applications ranging from air flow over an aircraft wing to combustion in a furnace, from bubble columns to oil platforms, from blood flow to semiconductor manufacturing, and from clean room design to wastewater treatment plants [9].

The code is commercial and to make the benchmarks, the academic research licenses are been used to made the simulations. The code, being not open source, is provided precompiled for a generic Linux x86_64 architecture therefore it was not possible to recompile it with optimization techniques as has been done with other codes discussed in this report. The software used for the benchmarks is Ansys Fluent version 12.1.

Commento [FA1]: Aggiunto spazio

4.1 Ansys Fluent benchmark

Three different test cases of CFD problems have been simulated. These benchmarks are part of a pool of nine different CFD problems provided by Ansys in the version 6.3 of Fluent; these are arranged by computational demand that is mainly influenced by the size of the mesh and the complexity of the equations modeled. For sake of brevity we want to avoid the detailed description of the physical model of each benchmark; the main characteristics are:

- M2: about 250000 cells; turbulence modeling with k-ε, steady simulation;
- M3: about 350000 cells; turbulence modeling with k-ε, steady simulation, chemical species transport;
- L1: about 850000 cells; turbulence modeling, compressible fluid.

4.2 Ansys Fluent results

The temporal element that identifies the performance, t_n expressed in seconds, was obtained by measuring only the time required to perform 50 iterations of the solver and therefore does not include loading the software in memory, reading and writing data from the disk and any of the domain partitioning calculation.

Commento [FA2]: among

For each of the three benchmarks several simulations have been conducted: the number of cores is varied from one to the maximum allowed on the single working node. In particular the simulations can be summarized in:

- AMD 6234 Interlagos: serial, 8 cores, 16 cores, 24 cores (full node);
- Intel E5-2680 Sandy Bridge: serial, 8 cores, 16 cores (full node), 24 cores with HT, 32 cores with HT (full node).

Results are showed in terms of *Rating*, *Speed-up* and *Efficiency* as proposed by Ansys in Ref. [10]. This is a short description of their meaning:

Rating: is an index that represent the "speed" or absolute performance of the simulation; it is the number of simulations that can run in a day:

$$\frac{24 \times 60 \times 60}{t_n}$$

it is inversely proportional to the time t_n.

Speed-up (S): for parallel simulations with n cores, is the ratio between the serial (t_s) and the parallel (t_n) execution time:

$$S = \frac{t_s}{t_n}.$$

Efficiency (E): is the speed-up normalized to the number of cores:

$$E = \frac{S}{n}.$$

Results of the benchmarks can be summarized as follows in the Tables 4, 5 and 6.

Execution time t_n (s) M2 case		
Cores	**AMD 6234 Interlagos 2.4 GHz, 64 GB RAM**	**Intel E5-2680 Sandy Bridge 2.7 GHz, 32 GB RAM**
1	123.3	62.1
8	22.2	9.80
16	12.7	5.75
24	9.6	7.14
32	-	5.73

Table 4: Execution time for M2 benchmark at different number of cores on the tested architectures.

Execution time t_n (s) M3 case		
Cores	**AMD 6234 Interlagos 2.4 GHz, 64 GB RAM**	**Intel E5-2680 Sandy Bridge 2.7 GHz, 32 GB RAM**
1	553.8	235.5
8	95.3	39.1
16	56.3	23.5
24	41.0	30.0
32	-	23.2

Table 5: Execution time for M3 benchmark at different number of cores on the tested architectures.

Execution time t_n (s) L1 case		
Cores	**AMD 6234 Interlagos 2.4 GHz, 64 GB RAM**	**Intel E5-2680 Sandy Bridge 2.7 GHz, 32 GB RAM**
1	1068.6	532.5
8	194.8	77.3
16	115.0	43.8
24	81.6	55.2
32	-	43.4

Table 6: Execution time for L1 benchmark at different number of cores on the tested architectures.

The Fig. 2 show the graphs of the comparisons of the three characteristics previously defined; the graphs are very similar for all the considered benchmarks so we show only the L1 benchmark in order to do not overload

overcome these drawbacks. One of these is tailoring Mg nanoparticles in view of an enhancement on the reaction kinetics and thermodynamics of the Mg-MgH2 phase transformation [18-20]. Because metallic nanoparticles often show size dependent behavior different from bulk matter, a better understanding of their physical-chemical properties is necessary. To approach numerical studying of Mg nanoparticles, starting from the magnesium bulk, we have considered the cluster built from the Mg atoms inside a 12 Å radius sphere centered on a Mg atom. To take in consideration the Kirkendall effect [21], we then have removed the inner Mg atoms inside a 5.6 Å radius sphere. The system is composed of a hollow nanoparticle of 266 magnesium atoms. In view of a better understanding of the Mg nanoparticles physical-chemical properties, we want to set up a numerical model to perform first principle calculations based on the density functional theory, using CPMD code.

5.2. Computational details

The Kohn-Sham method of DFT simplifies calculations of the electronic density and energy of a system of *Ne* electrons in an external field without solving the Schrödinger equations with *3Ne* degrees of freedom, but it takes into consideration the electronic density as the fundamental quantity (with only 3 degrees of freedom). The total ground-state energy of the system can be obtained as the minimum of the Kohn-Sham energy which is an explicit functional of the electronic density. This leads to a set of equations (Kohn-Sham equations) that has to be solved self-consistently in order to yield density and the total energy of the electronic ground-state. In the calculations, starting from an initial guess for the electronic density, the Kohn-Sham equations are solved iteratively until convergence is reached. We employed for all the calculations the CPMD code with Goedecker-Teter-Hutter pseudopotentials for magnesium, together with Pade approximant LDA exchange-correlation potentials [22-24]. The preconditioned conjugate gradient with line search method was used in order to yield density and the total energy of the electronic ground-state. We considered two cases: A) The electronic wave functions cutoff equal to 20 Ry, the density cutoff equal to 80 Ry, the number of plane waves for wavefunction equal to 218524, the number of plane waves for density equal to 1747687 and real space mesh 192x192x192. B) The electronic wave functions cutoff equal to 45 Ry, the density cutoff equal to 180 Ry, the number of plane waves for wavefunction equal to 737585, the number of plane waves for density equal to 5899402 and real space mesh 288x288x288.

Execution total time t_{tot} and average time for iteration t_{avg} (s) System A					
Cores	**PM (MB)**	**AMD 6234 Interlagos 2.4 GHz, 64 GB RAM**		**Intel E5-2680 Sandy Bridge 2.7 GHz, 32 GB RAM**	
		t_{tot}	t_{avg}	t_{tot}	t_{avg}
2	4037.1	10h57m32s	383.13	4h01m34s	142.39
4	2024.6	6h49m35s	238.21	2h10m32s	76.84
6	1353.8	5h32m14s	193.28	-	-
8	1018.5	4h08m33s	144.69	1h19m44s	46.77
12	683.3	2h46m30s	96.97	1h03m40s	37.17
16	515.4	2h06m18s	73.57	1h00m38s	35.28
24	347.7	1h27m22s	50.89	1h02m41s	36.46
32	263.7	-	-	1h10m42s	41.25

Table 11: Execution total time t_{tot} and average time for iteration t_{avg} for system A in function of number of cores on the tested architectures. PM is the Peak Memory for core.

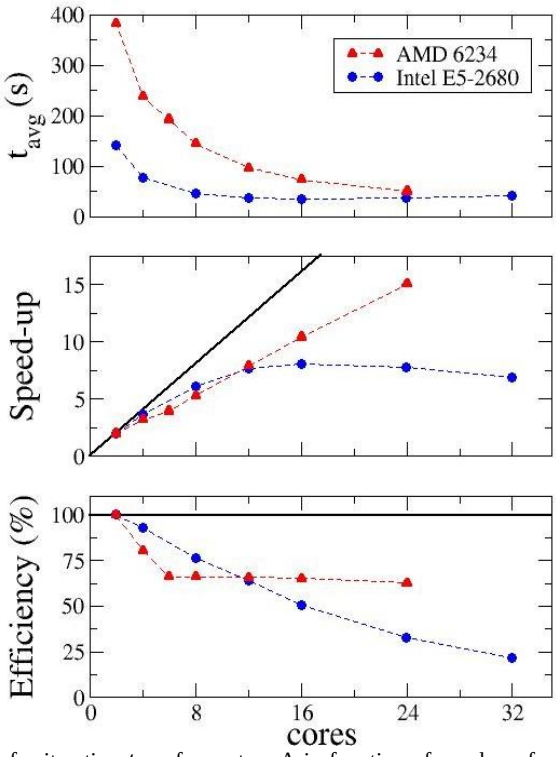

Figure 3: Average time for iteration t_{avg} for system A in function of number of cores for AMD 6234 and Intel E5-2680 processors. We also report the speed-up and the efficiency. Solid black lines are for ideal performance.

Execution total time t_{tot} and average time for iteration t_{avg} (s) System B					
Cores	**PM (MB)**	**AMD 6234 Interlagos 2.4 GHz, 64 GB RAM**		**Intel E5-2680 Sandy Bridge 2.7 GHz, 32 GB RAM**	
		t_{tot}	t_{avg}	t_{tot}	t_{avg}
2	13574.2	8h27m39s	1311.84	3h25m16s	561.04
4	6795.4	5h19m36s	806.35	1h37m32s	266.47
6	4535.6	4h23m50s	652.68	-	-
8	3406.0	3h19m13s	490.38	1h02m24s	166.65
12	2275.7	2h10m46s	330.05	50m01s	130.41
16	1711.3	1h36m26s	247.54	47m58s	122.36
24	1145.9	1h04m50s	167.91	47m11s	121.98
32	863.1	-	-	42m31s	107.61

Table 12: Execution total time t_{tot} and average time for iteration t_{avg} for system B in function of number of cores on the tested architectures. PM is the Peak Memory for core.

5.3 CPMD benchmark results

In Table 11 and 12 are shown the results for the CPMD benchmarks for system A and system B respectively. Because both the systems are very compute demanding, we limit our calculation to the first 100 iterations in the resolution of the Kohn-Sham equations for system A, and to the first 10 iterations for system B. We reported the total time required to complete the calculations and the average time for iteration. In Fig.3 and Fig.4 is shown t_{avg} in function of the number of cores. We also showed the speed-up and the efficiency for both systems. From the results we observe a better performance of the Intel Sandy Bridge processor compared to AMD Interlagos. The differences are more evident when little number of cores are used (about 50% less time).

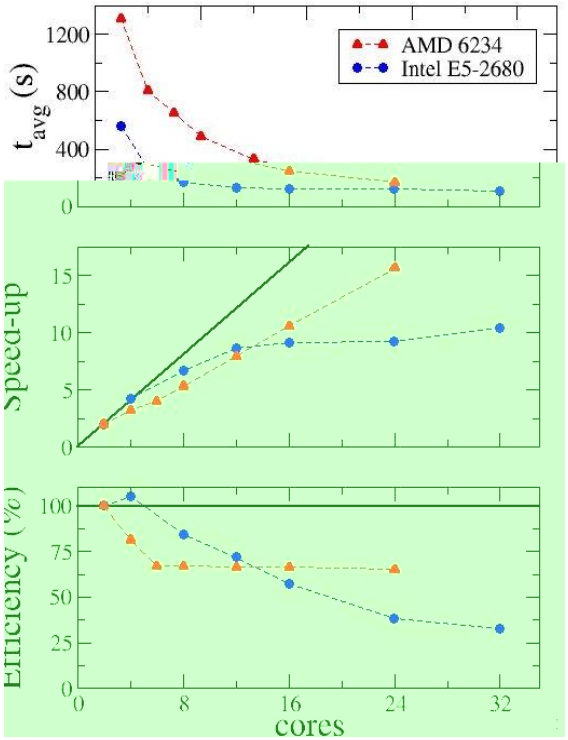

Figure 4: Average time for iteration t_{avg} for system B in function of number of cores for AMD 6234 and Intel Sandy E5-2680 processors. We also report the speed-up and the efficiency. Solid black lines are for ideal performance.

6 IMB-MPI1 Alltoall benchmark

The Alltoall is part of the Intel MPI Benchmarks family [25]. The Intel MPI Benchmarks is a concise, easy-to-use set of MPI benchmarks. It compares the performance of various computing platforms or MPI implementations and is very useful to test the interconnection network in HPC systems. In this work the test was used to compare the performances in terms of access to the shared memory between AMD 6234 Interlagos and Intel E5-2680 Sandy Bridge processors.

The Alltoall benchmark uses the MPI_Alltoall function and during the test each parallel process sends buffers of different lengths to all other processes in the same application. This procedure is iterated several times for each buffer length (from 0 to 4MB). The minimum, the maximum and the mean value of execution times are reported in the standard output for each step of the test. We performed the test on both architectures involving 24 processes, compiling both with GCC ver.

4.6.3, and Intel compiler ver 12.1.4 for Intel Sandy Bridge and Intel compiler ver 12.1.3 for AMD Interlagos.

6.1 IMB-MPI1 Alltoall benchmark results

We have observed that AMD Interlagos performs better than Intel Sandy Bridge Intel for buffers with length <= 32 bytes; instead Intel Sandy Bridge performs better than AMD Interlagos for buffers with length between 32 bytes and 4 MB. In order to compare the architectures, we computed for each step of the test the ratio between the mean value of the execution times on Intel Sandy Bridge and the mean value of the execution times on AMD Interlagos and then the mean value among these ratios, obtaining:

- test compiled with GCC: R=0.762;
- test compiled with Intel: R=0.768.

Whereas the clock frequency of Intel Sandy Bridge is 2.7 Ghz and the AMD Interlagos one is 2.4 Ghz we can normalize the results to 2.4 Ghz, obtaining:

- test compiled with GCC: R=0.857;
- test compiled with Intel: R=0.864.

The mean values of the execution times for the iteration on each buffer length are reported below in Table 13.

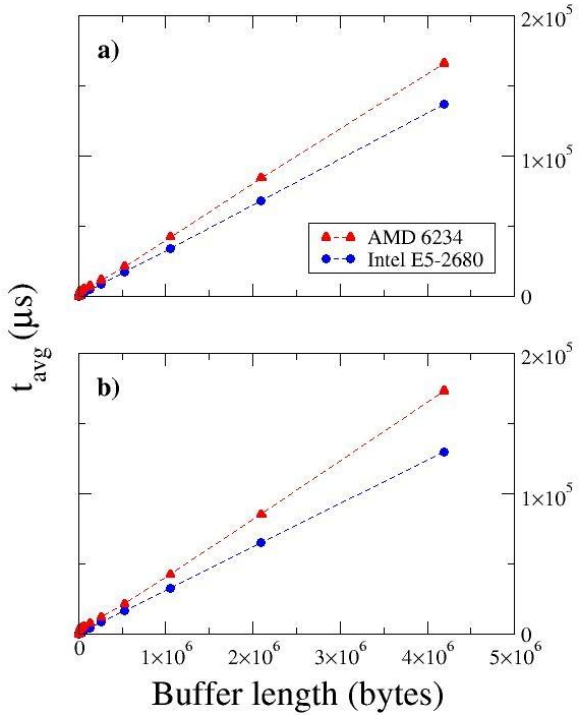

Figure 5: Intel MPI Alltoall benchmark. AMD 6234 vs. Intel E5-2680. a) Test compiled with . b) Test compiled with Intel.

Mean value of execution times t_{avg} (μs)				
Buffer Length (bytes)	**AMD 6234 Interlagos 2.4 GHz, 64 GB RAM**		**Intel E5-2680 Sandy Bridge 2.7 GHz, 32 GB RAM**	
		Intel		Intel
0	0.25	0.24	0.13	0.14
1	34.95	34.95	40.08	42.86
2	35.44	34.46	39.79	42.93
4	34.57	35.11	39.86	43.3
8	34.51	35.21	39.84	43.17
16	40.54	40.4	39.78	42.86
32	40.75	41.14	40.94	43.98
64	50.92	51.53	40.41	43.5
128	56.21	56.56	41.04	44.15
256	64.34	64.7	44.84	47.51
512	79.37	79.24	38.65	39.13
1024	110.47	109.72	44.48	45.93
2048	167.12	165.07	59.78	59.84
4096	314.9	313.57	153.14	153.34
8192	602.47	601.16	967.08	895.13
16384	3940.7	3864.2	1006.5	845.99
32768	4549.5	4504.6	1107.7	934.09
65536	5693.7	5637.4	2720.1	2517.6
131072	7633.2	7935.4	4943.4	4574.1
262144	11936	12232	9057.2	8537
524288	21541	21713	17463	16515
1048576	42399	42459	34252	32482
2097152	84406	85194	68542	65070
4194304	1.6603e+05	1.7315e+05	1.369e+05	1.297e+05

Table 13: Intel MPI Alltoall benchmark. AMD Interlagos vs. Intel Sandy Bridge. Mean values of execution time among the iterations on each buffer length.

7 Conclusions

In this report we have presented the results of different types of computational benchmarks realized to test the performance of the AMD 6234 Interlagos 2.4 GHz and Intel E5-2680 Sandy Bridge 2.7 GHz processors. The tests range from typical HPC benchmark to simulations of different physical systems. Moreover, we have used the Intel MPI benchmark to compare the performances in terms of access to the shared memory.

The HPL runs have made it possible to analyze the floating point performances under varying memory load. The Intel Sandy Bridge can achieve on average (from serial to 24 cores) 24% higher flops rate by using 24.5% more of the available node RAM. Also the results of OpenFOAM with 24 cores show that the Intel Sandy Bridge processor performs better than AMD Interlagos by taking about 20% less time to complete the run even with HT switched on. The results of Ansys Fluent show that Intel Sandy Bridge architecture appears to be faster for the proposed benchmarks in the serial simulation, and even without HT, Intel shows a parallel efficiency greater of the AMD. Intel performances are the same if the node is fully loaded without dependence of the HT. Instead, the efficiency decrease when HT is active (minimum is at 24 =16 + 8 cores) but absolute performance is greater of the corresponding Interlagos. The performances of Interlagos always grow at higher number of cores but the efficiency decreases and reaches the minimum at around 60% at full node loaded. The CPMD tests confirm the previous results. When little number of cores are used, for Intel Sandy Bridge the execution times are halved compared to those of AMD Interlagos. The differences are attenuated with increased number of cores. From the Intel MPI benchmark we have observed that AMD Interlagos performs better than Intel Sandy Bridge for buffers with length <= 32 bytes; instead, Intel Sandy Bridge performs better than AMD Interlagos for buffers with length between 32 bytes and 4 MB.

Bibliographic references:

[1] Binns F., et al. *"Hyper-Threading Technology Architecture and Microarchitecture"* Intel Technology Journal Q1 (2002).

[2] Petitet A., Whaley R.C., Dongarra J.J., Cleary A. *"Hpl: A portable implementation of the high-performance linpack benchmark for distributed-memory computers"* Innovative Computing Laboratory, Available online (2000). URL http://icl.cs.utk.edu/hpl/ and www.netlib.org/benchmark/hpl/.

[3] Dongarra J., Luszczek P., Petitet A. *"The linpack benchmark: past, present and future. Concurrency and Computation"* Practice and Experience **15**(9) (2003) pp. 803-820.

[4] Dongarra J., van de Geijn R.A., Walker D.W. *"Scalability issues affecting the design of a dense linear algebra library"* J. Parallel Distrib. Comput. **22**(3) (1994) pp. 523-537.

[5] Dongarra J., Eijkhout V., Luszczek P. *"Recursive approach in sparse matrix LU factorization"* Scientific Programming **9**(1) (2001) pp. 51-60.

[6] Petitet A.P., Dongarra J. *"Algorithmic redistribution methods for block-cyclic decompositions"* IEEE Transactions on Parallel and Distributed Systems **10**(12) (1999) pp.201-220.

[7] Weller H.G, Tabor G., Jasak H., Fureby C.*"A tensorial approach to computational continuum mechanics using object orientated techniques"* Computers in Physics **12**(6) (1998) pp.620-631.

[8] www.openfoam.com.

[9] www.ansys.com.

[10] www.ansys.com/Support/Platform+Support/Benchmarks+Overview/ANSYS+Fluent+Benchmarks.

[11] Car R., Parrinello M. *"Unified approach for molecular dynamics and Density-Functional Theory"* Phys. Rev. Lett. **55** (1985) p. 2471.

[12] www.cpmd.org.

[13] Hohenberg P., Kohn W. *"Inhomogeneous Electron Gas"* Phys. Rev. **136** (1964) p. B864.

[14] Kohn W., Sham L.J. *"Self-Consistent Equations Including Exchange and Correlation Effects"* Phys. Rev. **140** (1965) p. A1133.

[15] Andreoni W., Curioni A. *"New Advances in Chemistry and Material Science with CPMD and Parallel Computing"* Paral. Comp. **26** (2000) p. 819.

[16] Hutter J., Curioni A. *"Dual-level parallelism for ab initio molecular dynamics: Reaching teraflop performance with the CPMD code"* Paral. Comp. **31** (2005) p. 1.

[17] Giusepponi S., Celino M., Cleri F., Montone A. *"Hydrogen storage in MgH_2 matrices: a study of Mg-MgH_2 interface using CPMD code on ENEA-GRID"* Il Nuovo Cimento C **32** (2009) p. 139.

[18] Pasquini L., Callini E., Piscopiello E., Montone A., Vittori Antisari M. *"Hydrogen sorption in Pd-decorated Mg–MgO core-shell nanoparticle"* Appl. Phys. Lett. **94** (2009) 041918.

[19] Krishnan G., Kooi B.J., Palasantzas G., Pivak Y., Dam B. *"Thermal stability of gas phase magnesium nanoparticles"* J. Appl. Phys. **107** (2010) 053504.

[20] Pasquini L., Callini E., Brighi M., Boscherini F., Montone A., Jensen T.R., Maurizio C., Vittori Antisari M., Bonetti E. *"Magnesium nanoparticles with transition metal decoration for hydrogen storage"* J. Nanopart. Res. **13** (2011) p. 5727.
[21] Smigelskas A.D., Kirkendall E.O. Trans. AIME **171** (1947) p. 130.
[22] Perdew J.P., Burke K., Ernzerhof M. *"Generalized Gradient Approximation Made Simple"* Phys. Rev. Lett. **77** (1996) p. 3865.
[23] Goedecker S., Teter M., Hutter J. *"Separable dual-space Gaussian pseudopotentials"* Phys. Rev. B **54** (1996) p. 1703.
[24] Hartwigsen C., Goedecker S., Hutter J. *"Relativistic separable dual-space Gaussian pseudopotentials from H to Rn"* Phys. Rev. B **58** (1998) p. 3641.
[25] http://software.intel.com/en-us/articles/intel-mpi-benchmarks.