A STUDY ON PARALLELIZED SOLVERS FOR MOVING PARTICLE SEMI-IMPLICIT METHOD (MPS)

Fabio Kenji Motezuki, fabio.motezuki@poli.usp.br
Liang-Yee Cheng, cheng.yee@poli.usp.br
Department of Construction Engineering of Escola Politécnica – University of São Paulo.
Av. Prof. Almeida Prado, trav. 2, n. 83 Edifício de Engenharia Civil – Cidade Universitária, São Paulo, Brazil.

Marcio Michiharu Tsukamoto, michiharu@tpn.usp.br
Department of Naval Architecture and Ocean Engineering of Escola Politécnica – University of São Paulo.
Av. Prof. Mello Moraes, n. 2231 – Cidade Universitária, São Paulo, Brazil.

Abstract. *The Moving Particle Semi-implicit (MPS) method uses a lagrangean particle representation to simulate incompressible fluid flows. In the implicit part of this method a linear system formed by Poisson's equation is solved. The matrix used to solve the linear system is symmetrical, sparse, unstructured and may change from one time-step to another.The size of the matrix is proportional to the number of particles squared and a typical case have hundreds of thousands particles Thus the time to solve the linear system become the bottleneck for the simulation. To deal with, iterative parallelized solvers are an option to speed-up the simulation. Two implementations of Conjugate Gradient method and one implementation of Algebraic Multi-Grid method were tested and the resuls are compared by the time consumed in solving a typical MPS linear equation system of 300,000 particles. The results indicates that the Algebraic Multi-Grid method is better suited for the kind of matrix obtained in the MPS method*

Keywords*: Parallel Processing, MPS, CFD*

1. INTRODUCTION

The MPS method, presented by Koshizuka and Oka (1996), is a fully lagrangean method for simulation of incompressible fluid flows based upon particles and the interaction between them. The lagrangean approach of the method simplifies the equations and eliminates one of the main difficulties in flow simulation which is the numerical diffusion also, as a meshless method, it is very effective for highly non-linear problems that involves free-surfaces, large deformations of fragmentation.

The method uses a matrix to solve the linear equation system of the implicit part that is symmetrical, sparse and may change from one time-step to another. The size of the matrix is proportional to the number of particles squared that for a typical case that can reach hundreds of thousands. In addition to this there is an increasing requirement for faster results, more complex and larger simulations using this method. These reasons make the solution of the linear system become the bottleneck for the simulation, to deal with this there is a necessity to find more efficient solvers.

This work has the objective to analyze the performance of two implementations of the Parallelized Conjugate Gradient method and one Algebraic Multi-Grid method in a simulator using the MPS method. In section 2 the main characteristics of the MPS method is presented, in section 3 the comparison method is presented and in section 4 the behavior of the solvers are analyzed in two distinct parts of simulation using a hydrostatic tank, measuring the influence of matrix sparsity. Also a 2D dam breaking test case is used to analyze the processor influence on the processing time of the solvers.

2. THE MPS METHOD

The governing equations are the continuity and Navier-Stokes, Eq. (1) and Eq. (2) respectively.

$$\frac{\partial \rho}{\partial t} = 0 \tag{1}$$

$$\frac{Du}{Dt} = -\frac{1}{\rho}\nabla P + f \tag{2}$$

Where:
ρ is the density,
t is the time,
u is the velocity in the direction u,

P is the pressure,

f represents the external forces.

The method is composed by an explicit part, whose calculation is simple from the computational point of view, where the external forces (f) are calculated.

In MPS the interaction between particles is defined by a kernel function $w(r)$ given by Eq. (3).

$$w(r) = \begin{cases} \dfrac{r_e}{r} - 1 & (0 \le r < r_e) \\ 0 & (r_e \le r) \end{cases} \tag{3}$$

Where:

r_e is the effective radius,

r is the distance between the particles.

The effective radius r_e defines the limits of the influence of one particle and its neighbors.

The kernel function is also used to calculate the particle number density n given by Eq. (4). The particle number density has a direct relation with the density of the fluid, by keeping the particle number density constant and equal to the initial particle number density, MPS method assures the incompressibility of the flow.

$$\langle n \rangle_i = \sum_{i \neq j} w(|r_j - r_i|) \tag{4}$$

In the implicit part, the pressure is calculated by solving the Poisson equation of pressure for each particle, given by Eq. (5).

$$\langle \nabla^2 P^{n+1} \rangle_i = -\frac{\rho}{\Delta t^2} \frac{\langle n^* \rangle_i - n^0}{n^0} \tag{5}$$

Where:

P is the pressure,

Δt is the simulation time-step

n^* is the particle number density for the particle

n^0 is the initial particle number density

By using a model of interaction between particles, Koshizuka and Oka (1996) modeled the laplacian operator present on the left hand side of Eq. (5) as following:

$$\langle \nabla^2 P \rangle_i = \frac{2d}{n^0 \lambda} \sum_{j \neq i} (P_j - P_i) w(|r_j - r_i|) \tag{6}$$

Where:

d is the number of dimensions in the study case (2 for bidimensional, 3 for tridimensional),

λ is the weighted average of distance between particles squared, represented by Eq. (7),

$w(|r_j - r_i|)$ is the weight function between the particles i and j given by Eq. (3).

$$\lambda = \frac{\int_V w(r) r^2 dV}{\int_V w(r) dV} \tag{7}$$

As shown by Eq. (3) to (7), despite MPS is a very flexible meshless method, it is much more time consuming than the conventional methods because, instead using a mesh, the topology of the computing points is defined by the particle interaction model described in Eq. (3) and the volume of calculation increase as r_e increases.

On the other hand, as the neighborhood may change from one time to another the matrix of coefficients of the linear equations formed by Eq. (5) may change during the simulation, which represents and additional challenge for speed-up of the processing time.

The effective radius, where only the influence of the particles inside this radius is considered in the calculus, used by MPS reduces significantly the data in the linear equations system, resulting in a sparser matrix to be solved. Due to the reciprocity of interactions between the particles, the linear system is symmetrical and has the order of the number of particles that, for a typical case, can achieve hundreds of thousands.

Due to the characteristics of the linear equation system presented (large, sparse, unstructured and symmetric) the processing time of the solver in a typical case consumes almost 40% of the total simulation time, as can be seen in Figure 1, a research in solvers that can perform faster and reliable solutions is required. The figure also shows that the neighbor search algorithm also takes a long processing time. However its optimization will not be discussed in this paper.

Also considering the popularization of multi-core machines parallelized solvers can take advantage of these machines, splitting the work into tasks to solve the linear system using the processors full power.

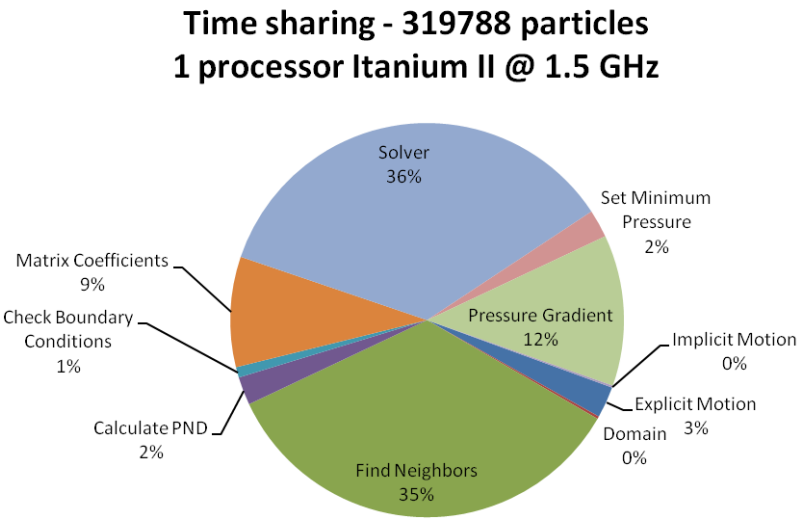


Figure 1 – Time sharing of various parts of MPS simulation program for a case with 319788 particles

In this way, in order to reduce the processing time regarding the solution of the system of equations, different types of solvers was tested. Direct solvers have been tested but the solution time was not good as with the iterative solvers and, since the iterative ones can make a compromise solution between precision and solution time, they were chosen for the study.

3. METHOD

In this work two Parallelized Conjugate Gradient (PCG) implementations by Intel and Hypre were tested and compared, a promising approach using Algebraic Multi-Grid (AMG) implemented by Hypre called BoomerAMG is also tested (Falgout and Yang, 2002; Henson and Yang, 2002).

The solvers were implemented in a modular mode, so only the solver is changed in the tests and all other parts of the simulation program remains untouched.

The solvers are implemented in the MPS simulator using a standard interface to the function, by this method of implementation, the solvers are changed by only substituting the function were they are implemented, minimizing the impact on other parts of the simulator.

The methods are compared based on the time spent in the solver function that concentrates all the required steps to use the solver in the program. This time is measured using internal C+ time function, registering the time the solver function starts and ends and by difference calculates the time spent in solver function.

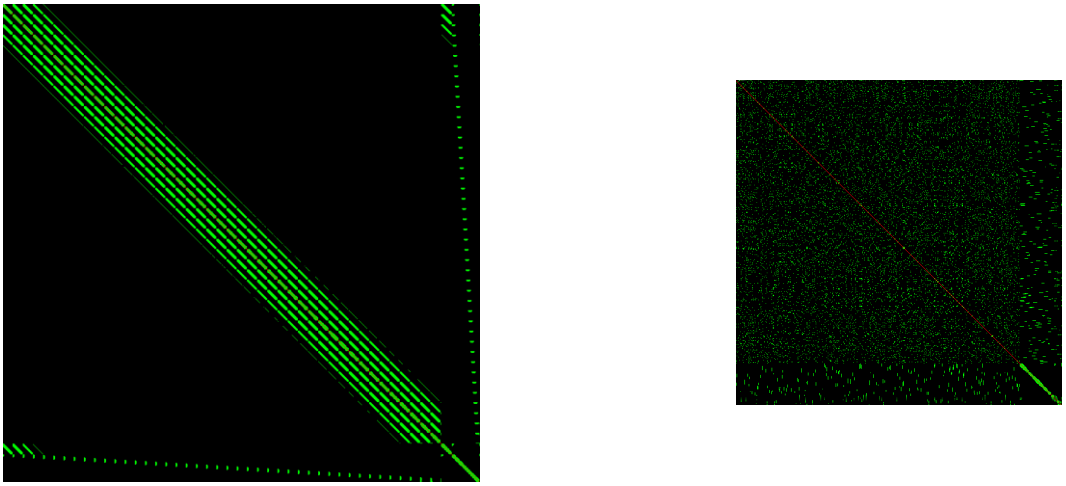
The machine used in simulations is a SGI-Altix computer with 16 Intel Itanium II processors running at 1.5 GHz and 32Gb of RAM memory. Up to 8 processors of the computer was used for the tests due to simultaneous utilization by other researchers.

After that an analysis of the influence of the processor type is carried out using three different processors: two intel Core2 and one intel Corei7, and the speed-up and scalability of the BoomerAMG method are tested and analyzed.

4. RESULTS AND DISCUSSION

4.1. The influence of the MPS linear equation system

The MPS is a meshless method and each particle does not have a fixed or a predictable relationship with other particles. The resulting matrix to be solved from the beginning of the simulation, due the order of particle creation process, looks like a band type matrix illustrated by Figure 2a. While the simulation advances, the position of the particle changes modifying the neighborhood around the particles and resulting in a completely sparse matrix, as shown in Figure 2b, this behavior do not allows the use of topology optimized solvers. The size of the matrices shown as example in Figure 2a and 1b differs due to the increase on the particles of free surface, in which pressure is imposed as boundary condition and no calculation is required.



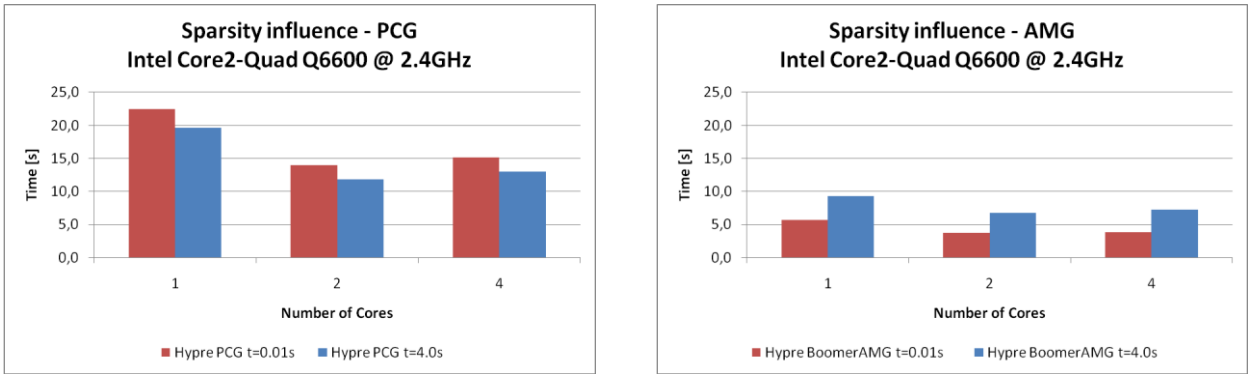
(a) time = 0.01s (b) time = 4.0s

Figure 2 – Matrix obtained from MPS simulation of 2D dam break with 1512 particles at start (a) and after 4s. (b). Light pixels indicates location with values different of zero. The size differs due to increase of free surface.

To measure the performance of solvers with a common start base, the first 200 time-steps were used since the particle organization depends mostly on the creation order, showing the behavior of the solver for the nearly diagonal matrix topology. More advanced time-steps will depend on the simulation development and can arrive to slightly different matrices to be solved. The test case used is a hydrostatic tank with 319788 particles, which was chosen to minimize the matrix reduction influence caused by increasing the number of free surface particles.

Depending on the solver, the change of the matrix during the simulation can be an advantage or a drawback to the method. Figure 3 shows the effects of the matrix modification on the solution time for Parallel Conjugate Gradient (PCG) and Algebraic Multi-Grid (AMG).

In this analysis, two matrices was taken from a simulation, one at the start of simulation (0.01s) when the particles are organized and other when the particles are scrambled (4.0s), to grant equal simulation conditions for both solvers and calculated in a external program made for this purpose. The machine used is equipped with a processor Intel Core2-Quad Q6600 @ 2.4GHz, with 4 processing cores and 4Gb of memory RAM, the results are measured in time consumed by the solver.



(a) (b)

Figure 3 – Influence of matrix sparsity on the solving time for (a) PCG and (b) AMG methods.

From the Figure 3 can be seen that PCG method obtained some advantage when solving the completely sparse matrix while the AMG method is less efficient for sparse matrices, but on overall it is faster than PCG and matrix reordering schemes can be applied to bring it to a near diagonal topology.

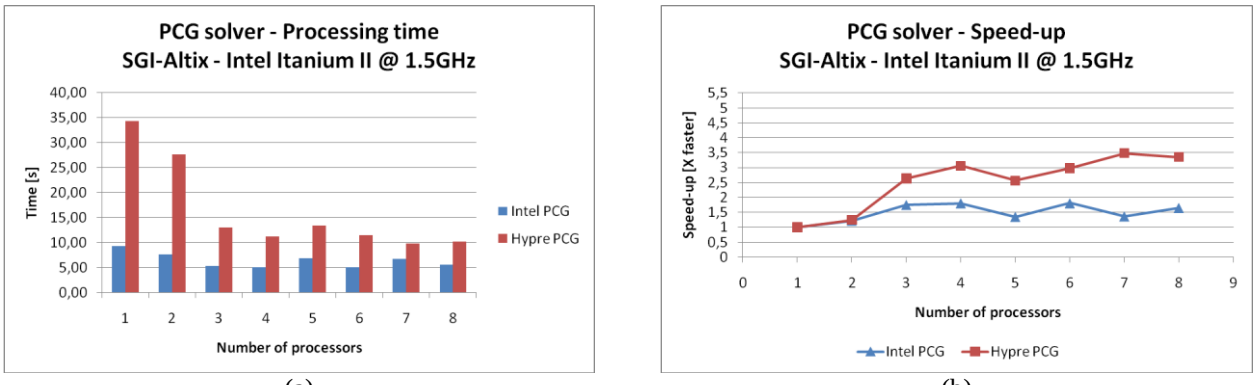
4.2. Parallel Conjugate Gradient

The Conjugate Gradient Method is one of the most commonly used Krylov-based iterative methods for symmetric matrices, the parallel implementation of this method by Intel and Hypre is used in this work.

The Intel solver uses the concept of threading, better suited for shared memory systems, where all data is available to all processors in one single memory poll. It is implemented in the Math Kernel Library (MKL), a specialized library of high performance math routines for science, engineering and financial applications (Intel, 2008).

The Hypre implementation is based on the concept of distributed memory systems, or clusters, where the data is spread between the various nodes of the system, and each node stores only its significant data. The communication between the multiple processes are handled by Message Passing Interface (MPI) a de facto standard in communication for clusters (Gropp et al. 2008)

The measured time spent for these two implementations are shown on Figure 4a and the speed-up of the solvers, that is how much the solver gained in time reduction compared with the 1 processor time as base, is calculated and showed in the Figure 4b.



(a) (b)

Figure 4 – Comparison between Intel and Hypre PCG implementations (a) wall clock time spent in the solver function and (b) speed-up achieved.

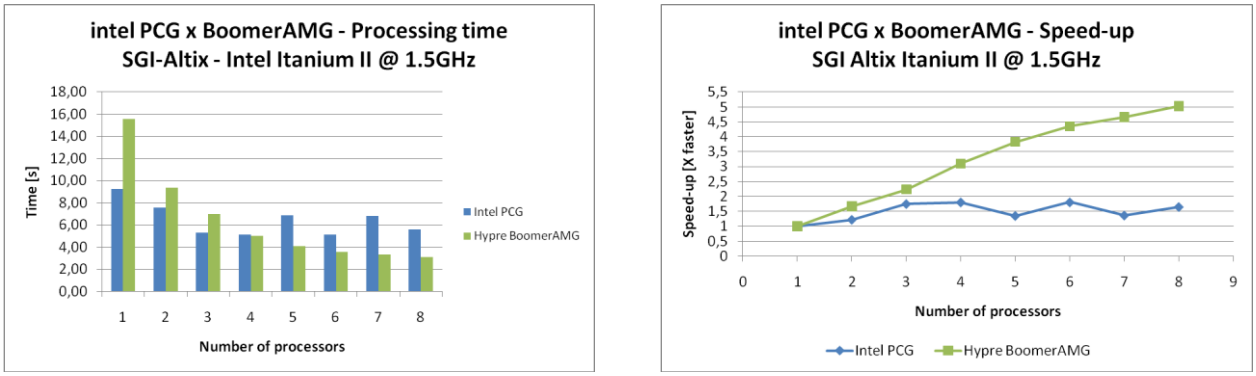
It is notable that Intel PCG implementation of the solver is far faster than Hypre's one, what is expected since the Intel solver is highly optimized for Intel processors, but is also noticeable that, for this case, the speed-up does not necessary occurs when using more than 4 processors, this is observed by the speed-up curve in the Figure 4b that stalls to about 1.5 for 5 processors.

On other hand Hypre's PCG implementation have a reasonably good speed-up, reaching 3 times faster for 4 processors, but the times measured for the solve time are longer than Intel.

4.3. Algebraic Multi-Grid - BoomerAMG

The Algebraic Multi-Grid was introduced in the 1980s and grabbed the attention of scientists needing to solve large and unstructured grid problems. Currently there is a great interest in applying the method in extremely large problems in the million or billion number unknowns and the BoomerAMG, presented by Henson and Yang (2002) and part of Hypre library, is a parallel implementation of AMG.

This method was implemented on the SGI-Altix computer, and simulated with the same 319788 particles case from PCG the results for up to 8 processors are shown in the Figure 5.



(a) (b)

Figure 5 – Comparison between Intel PCG implementation and Hypre's BoomerAMG (a) wall clock time spent in the solver function and (b) speed-up achieved.

The PCG method is faster than the BoomerAMG for up to 4 processors, when the PCG reaches its maximum speed-up. For more processors the tendency of PCG speed-up curve is almost constant while for BoomerAMG the speed-up continue to increase, this better scalability of BoomerAMG is relevant for the simulation of bigger cases with more particles, since the task may be divided between more processors to reduce processing times.

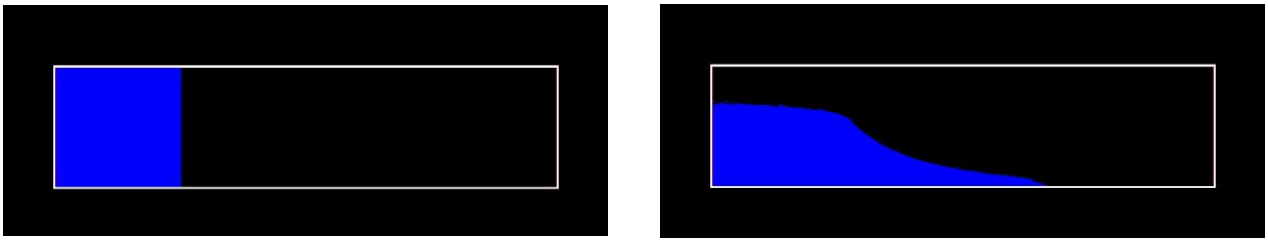
4.4. Influence of the processors

The result obtained in the test run using the Core2-Quad processor showed in section 4.2 was far faster than the obtained in the SGI-Altix environment. In this way to analyze the influence of the processor a series of tests were conducted using the hypre BoomerAMG solver, which showed good scalability, in two processors of the intel Core2 family and one in the new Corei7 family, a summary of the processor specifications is given in Tab. 1 . The case studied is a 2D dam breaking, with 80069 and 245702 particles.

Table 1. Summary of the processor specifications used in the tests

Processor	technology	Processing cores	Cache (MB)	Clock speed (GHz)	System memory (GB)	Memory frequency (MHz)
Core2 Q6600	65 nm	4	8	2.40	4	800
Core2 Q8200	45 nm	4	4	2.33	4	800
Corei7 920	45 nm	4	8	2.66	8	1066

The time consumed in the solver is measured at the beginning of the simulation (a), when the matrix for linear system have the diagonal band topology, and at the middle of the simulation. The measure at the middle of simulation was made at 0.78s of simulation time, at this point the water column already collapsed as shown in the b, and the linear equation matrix is completely sparse. An average of the 200 time-steps is considered, which correspond to 0.02s of simulated time.



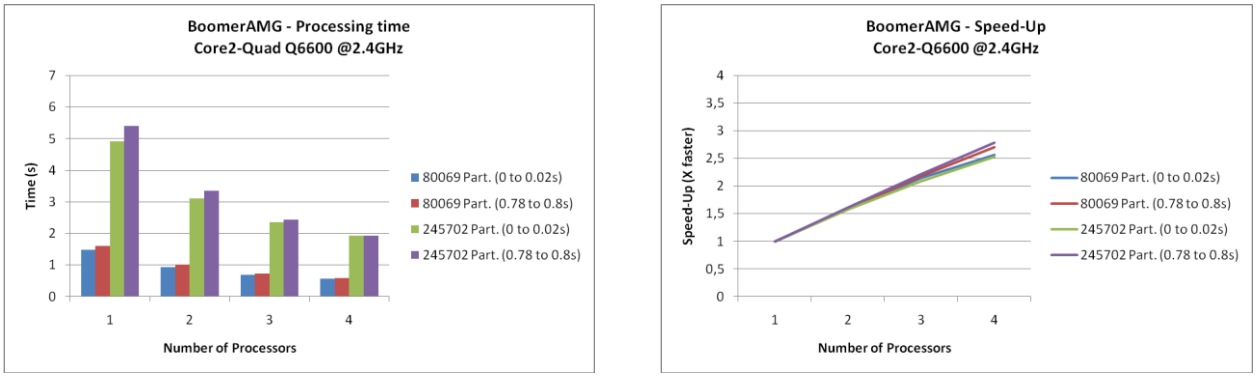
(a) (b)

Figure 6 – Instants where the processing time of the solver are measured, (a) at the start of simulation from 0s to 0.02s and (b) at the middle of simulation from 0.78s to 0.80s

In the Figure 7a, Figure 8a and Figure 9a, dam breaking tests were executed with 80069 and 245072 particles, approximately 3 times bigger. As the figures show, the larger test case consumed 3 times more processing time, showing a direct relation between the size and the time consumed.

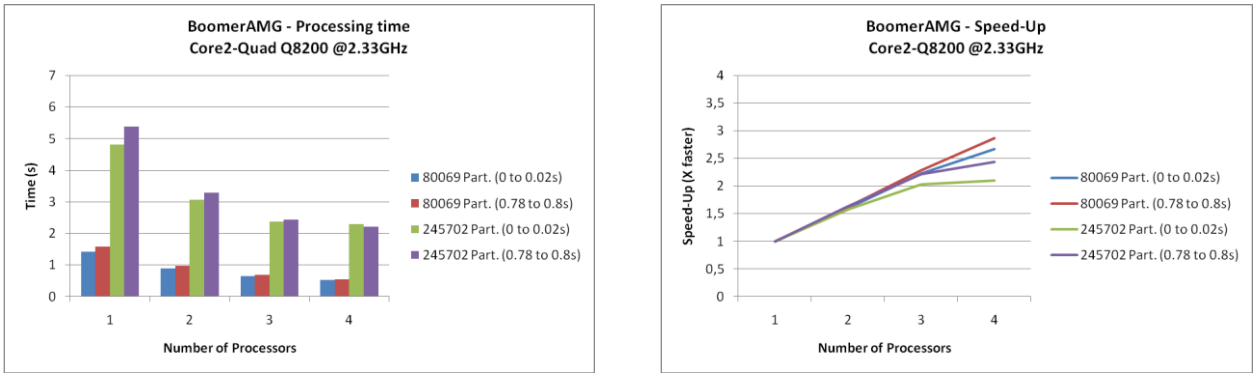
The scalability of the method is shown in the speed-up graphs, Figure 7b, Figure 8b and Figure 9b, the Core2 processors realized a speed-up of near 3 times using 4 processors and the Corei7 processor speeds-up above 3,5 times what shows the good scalability of the method for the different processors analyzed.

For these cases the 0.78-0.8s average processing time was just around 10% slower than at the beginning of simulation, which is less than the difference showed in the Figure 3. The increase of the number of free-surface particles during the dam breaking reduces the size of the matrix to be solved, and this reduces the processing time.



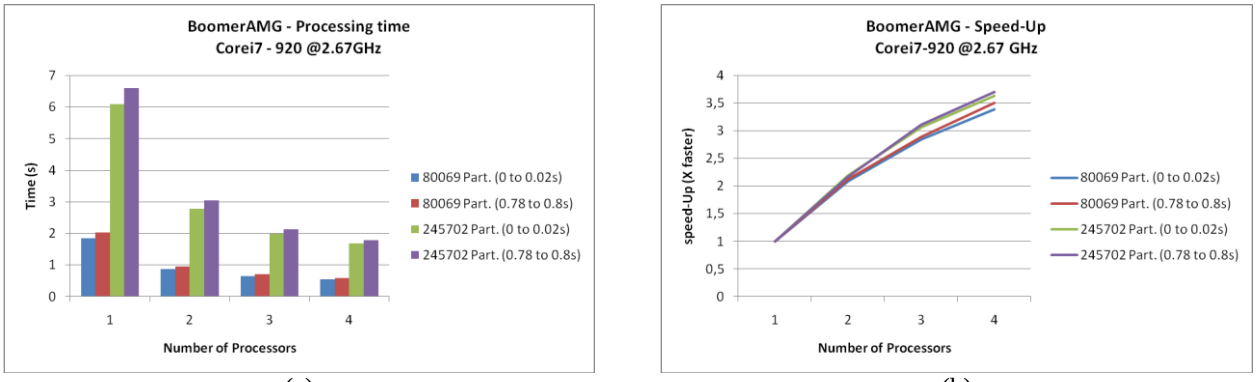
(a) (b)

Figure 7 – Comparison of method scalability for 80069 and 245702 particles and respective speed-up for intel Core2 Q6600 processor



(a) (b)

Figure 8 – Comparison of method scalability for 80069 and 245702 particles and respective speed-up for intel Core2 Q8200 processor.



(a) (b)

Figure 9 – Comparison of method scalability for 80069 and 245702 particles and respective speed-up for intel Corei7 920 processor.

The Figure 9 shows that intel Corei7 processor is slower than other tested for one processor, but for more than two processors there is a slight advantage, which can indicate a better suitability for multiprocessing, also for 2 processors there is a speed-up higher than 2 that have to be better investigated. On other hand the advantage to Corei7 processor was expected but the difference presented in current tests is more related to the higher clock speed than to the rest of system performance.

5. CONCLUDING REMARKS

The results show that depending on the implementation of the PCG method the difference between the solution times may vary largely, about 3 times in the worst case.

The AMG method showed to be more suitable one for the MPS type of linear system, mainly for the bigger ones where the parallelization is needed, having a good speed-up and fast solver times.

Regarding the performance of the Corei7, the set solver/compiler/math libraries is slower if only one processor is used, however for parallel processing the speed-up is slightly better than the older Core2 processor, what makes the difference between these two families of processors be lower than the expected.

6. REFERENCES

Falgout, R.D. and Yang, U.M., 2002, “hypre: a Library of High Performance Preconditioners”, in Computational Science – ICCS 2002 Part III, 07 May 2009 <https://computation.llnl.gov/casc/linear_solvers/pubs.html>

Gropp, W. et al., 2008 “MPICH2 User’s guide: Version 1.0.8”, Mathematics and Computer Science Division – Argonne National Laboratory.

Henson, V.E. and Yang, U.M., 2002, “BoomerAMG: A parallel algebraic multigrid solver and preconditioner”, Applied Numerical Mathematics, Vol. 41, pp. 155-177.

Intel, 2008, “ Intel® Math Kernel Library for the Linux OS”, 9 may 2009 <http://software.intel.com/en-us/articles/intel-math-kernel-library-documentation/>

Koshizuka, S. and Oka, Y., 1996, “Moving-Particle Semi-Implicit Method for Fragmentation of Incompressible Flow”, Nuclear Science and Engineering, Vol.123, pp. 421-434.

7. RESPONSIBILITY NOTICE

The authors are the only responsible for the printed material included in this paper.