A Parallel Implementation of the Lattice Solid Model for the Simulation of Rock Mechanics and Earthquake Dynamics

Steffen Abe$^{(1)}$, David Place$^{(1)}$ and Peter Mora$^{(1)}$

(1) QUAKES, Department of Earth Sciences, The University of Queensland, Brisbane, Australia, (e-mail: steffen@quakes.uq.edu.au, mora@quakes.uq.edu.au, place@quakes.uq.edu.au, phone: +61-7 3365 4853 +61-7 3365 2128; +61-7 3365 2176, fax: +61-7 3365 7347).

Abstract

The Lattice Solid Model has been used successfully as a virtual laboratory to simulate fracturing of rocks, the dynamics of faults, earthquakes and gouge processes. However, results from those simulations show that in order to make the next step towards more realistic experiments it will be necessary to use models containing a significantly larger number of particles than current models. Thus, those simulations will require a greatly increased amount of computational resources. Whereas the computing power provided by single processors can be expected to increase according to "Moore's law", i.e. to double every 18-24 months, parallel computers can provide significantly larger computing power today. In order to make this computing power available for the simulation of the microphysics of earthquakes, a parallel version of the Lattice Solid Model has been implemented. Benchmarks using large models with several millions of particles have shown that the parallel implementation of the Lattice Solid Model can achieve a high parallel efficiency of about 80% for large numbers of processors on different computer architectures.

Introduction

Most simulations based on particle dynamics models such as the Discrete Element Model (Cundall and Strack 1979 [1]) and the Lattice Solid Model (Mora and Place 1994 [5]) have been performed using a relatively small number of particles. Simulations of processes requiring a large number of time steps to be simulated such as gouge shear have typically used between several hundreds and a few thousand particles (Morgan and Boettcher 1999 [8], Place and Mora 2000 [12]) while in simulations of processes which require a relatively small number of time steps such as compression and fracturing of solids (Place et al. 2002 [14]) models containing a few tens of thousands of particles have been used.

However, recent laboratory experiments have shown that the behavior of fault gouge not only depends strongly on the particle shape and size distribution (Mair et al. 2002 [4] but also that there is a significant difference in behavior between

2D and 3D models (Frye and Marone 2002 [2]). It is also been suggested that those results explain the differences observed between laboratory experiment and current numerical simulations. In order perform realistic simulations which provide results directly comparable to laboratory experiments it will be thus be necessary to use 3D simulation models with particle shapes and size ranges comparable to those used in the laboratory experiments. Considering the range of particle sizes between 1-800μm used by Mair et al. [4], this would suggest simulation models containing at least several million particles. This is currently not feasible using serial simulation software on a single CPU, however it will be possible if the significantly larger computational power of massively parallel computer systems can be used for the simulations.

Overview over the Lattice Solid Model

The Lattice Solid Model (Mora and Place 1994 [5], Place and Mora 1999 [10]) is a particle based model similar to the Discrete Element Model (DEM) developed by Cundall and Strack 1979 [1]. The model consists of spherical particles which are characterized by their radius r, mass m, position $\mathbf{x}$ and velocity $\mathbf{v}$. The particles interact with their nearest neighbors, e.g. by elastic and frictional forces. The particles can be linked together by elastic bonds or springs (Figure 1), in which case the elastic forces are attractive or repulsive, depending on whether the particles are closer or more distant than the equilibrium distance.

$$\mathbf{F}_{ij}^{linked} = \begin{cases} k_{ij}(r_{ij} - (r_0)_{ij})\mathbf{e}_{ij} & r_{ij} \leq (r_{cut})_{ij} \\ \mathbf{0} & r_{ij} > (r_{cut})_{ij} \end{cases} , \qquad (1)$$

where k_{ij} is the spring constant for the elastic interaction between the particles, r_{ij} is the distance between the particles i and j, $(r_{cut})_{ij}$ the breaking distance for the link between the particles and $\mathbf{e}$ is a unit vector in the direction of the interaction.

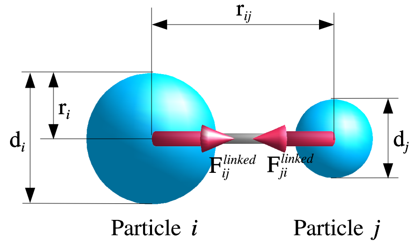

Figure 1: Attractive forces between linked particles. $\mathbf{F}_{ij}$ is the force applied to particle i due to the interaction with particle j whereas $\mathbf{F}_{ji}$ is the force applied to particle j due to the interaction with particle i.

Links are broken if the distance between the particles exceeds the threshold breaking distance $(r_{cut})_{ij}$. If two particles are not linked together (Figure 2) the elastic force $\mathbf{F}_{ij}^{free}$ between the particles i and j is purely repulsive

$$\mathbf{F}_{ij}^{free} = \begin{cases} k_{ij}(r_{ij} - (r_0)_{ij})\mathbf{e}_{ij} & r_{ij} \leq (r_0)_{ij} \\ \mathbf{0} & r_{ij} > (r_0)_{ij} \end{cases} . \qquad (2)$$

An intrinsic friction between particles has been incorporated in the model (Place and Mora 1999[10]). Two unbonded interacting particles can be in static or dynamic

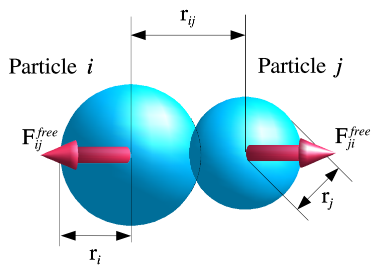

Figure 2: Repulsive forces between particles which are not linked together.

frictional contact. The force on particle i due to the dynamic frictional contact with particle j is given by

$$\mathbf{F}_{ij}^{D} = -\mu F_{ij}^{n} \mathbf{e_{ij}}^{T} \tag{3}$$

where μ is the coefficient of friction between the particles, F_{ij}^{n} is the magnitude of the normal force and $\mathbf{e_{ij}}^{T}$ is a unit vector in the direction of the relative tangential velocity between the particles (Mora and Place 1998 [6]).

Rotational dynamics can be simulated in the Lattice Solid Model both by per-particle rotation (Winter et al. 1997 [17], Sakaguchi and Mühlhaus, 2000 [16]) where each single particle has angular velocity $\boldsymbol{\omega}$ and momentum $\mathbf{I}$ or by collective rotation of groups of irrotational particles (Place 1999 [11]).

Parallel Implementation

There are two fundamentally different approaches to the design of parallel programs: a purely data parallel approach which keeps a single thread of control within the program and the explicitely distributed approach which also distributes the thread of control. While the data-parallel approach has been successful using a relatively small number of CPUs (Place and Mora 1997 [9], Place 1999 [11]), it is not well suited for the parallelization of the Lattice Solid Model on computer systems with a high number of CPUs. The results of performance and scalability tests performed by Place [11] show that a purely data-parallel implementation of the lattice solid is likely to lead to a program which still contains significant serial overhead and will not scale well on a high number of CPUs. Thus the second, explicitely distributed approach has been taken for the parallel implementation of the lattice solid model. Explicit message passing using MPI [1] as underlying communication library has been used to implement the inter-process communication because MPI is a well defined, open standard, thus supporting the portability of the program to a wide range of computer architectures and also because most vendors of current parallel computer systems provide optimized implementations of MPI, thus ensuring good performance on those systems.

The parallel process structure follows a modified master-worker model. A master process provides high level control and external communication such as a user interface or I/O facility. The worker processes perform the computational work. In order to minimize the computation performed by and the communication with the

[1]MPI: **M**essage **P**assing **I**nterface – an open standard for message passing.

master process, thus reducing the serial part of the program. Thus, the master process handles only the global high level control flow and each worker processes then takes control of the local computations. In contrast to a pure master-slave approach, direct communication between worker processes is used instead of communication involving the master process whenever possible (Figure 3).

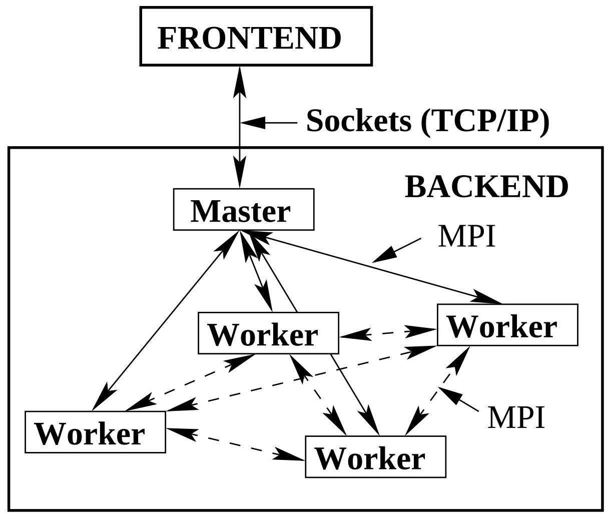

Figure 3: Process structure of the parallel lattice solid model. The communication between the parallel processes are implemented using MPI.

For molecular dynamics algorithms, which are closely related the lattice solid model, three main approaches for the distribution of the work between the parallel processes have been suggested (Rapaport, 1995 [15]). The first approach would be to partition only the computation and share all data between the processors, i.e. each processor as direct access to all data. An efficient implementation of this approach, at least in its pure form, is restricted to shared memory computers. On distributed memory systems it would require the duplication of the whole data set on each node, thus being inefficient in terms of memory use. A second possibility is a partitioning scheme based on the particles, assigning each particle to a particular processor for the entire run time of the simulation, irrespective of the position of the particle. This leads to the efficient use of memory, but the handling of interactions between particles assigned to different processors requires a large amount of communication. The third choice is to partition the problem spatially and assign a subregion to each processor. This processor will then perform the computations for all the particles which are located within the subregion at the current time step. The amount of communication required by this approach is potentially much smaller then in the second approach because interaction between particles assigned to different processors can only occur along the boundaries of the subregions, thus restricting the number of particles involved in such interaction. However, this approach makes it necessary to move particles between processors. In classic molecular dynamics simulations these relatively infrequent movements of particles between processors have no significant impact on the performance of the computation. However, unlike in molecular dynamics simulations, the Lattice Solid Model contains persistent bonds between particles. Thus there is additional connectivity information associated with each particle which would have to be moved between processors if a particle moves from one subregion into another, making the third approach in its pure form less attractive for the Lattice Solid Model. For this reasons, a hybrid method containing elements from all three approaches has been used in the current parallel implementation of the Lattice Solid Model.

The initial distribution of the particles is based on a partitioning of the problem space, but additionally to the particles located in a particular subregion, the data set assigned to each processor also contains all particles interacting with any particle in the subregion. The forces due to interactions which are assigned to more then one processor are computed by each processor, which leads to a small increase in computations, but reduces communication. When particles move between subregions, they remain assigned to the same process but the generation of new particle interactions caused by the particle movement makes it necessary to update the set of particles shared between different processes, leading to an increase in communication cost over time. Thus a redistribution of the particles between the processes will be necessary to restore the purely space-based partitioning of the work between the processes and to minimize inter-process communication. However, as such a redistribution will be performed only infrequently for most simulations it will not have a significant impact on performance.

Verification Tests

In order to verify that the algorithm works correctly, a verification test was performed. A displacement source was located in the center of the lattice and the propagation of the resulting waves was observed. The source was similar to the one used by Place et al. 2000 [13] to test the propagation of seismic waves in the serial implementation of the lattice solid model. The displacement d in x and y-direction is given by

$$\begin{aligned} d_x &= a_x e^{\frac{(t-t_{0,x})^2}{b_x}} && (4) \\ d_y &= a_y e^{\frac{(t-t_{0,y})^2}{b_y}} \quad . && (5) \end{aligned}$$

The constants chosen for the test are $a_x = a_y = 0.1$, $b_x = b_y = 3.0$, $t_{0,x} = 3.0$ and $t_{0,y} = 4.0$. All values are given in model units. Those parameters lead to a characteristic wavelength of the source wavelet of $l_p \approx 12r_0$ for the P-wave and $l_s \approx 7r_0$ for the S-wave. The lattice used for the test was a regular triangular 2D lattice of 64×64 particles with radius $r_0 = 1.0$ and mass $m_0 = 1.0$. A snapshot of the simulation at $t = 29$ (Figure 4) shows the radial propagation of P- and S-wave from the source in the center of the lattice. The wave speeds measured from those tests are

$$\begin{aligned} v_p &= 1.04 \pm 0.02 && (6) \\ v_s &= 0.59 \pm 0.02 \quad . && (7) \end{aligned}$$

The theoretical values of the wave speeds in a 2D regular triangular lattice can be calculated from the particle masses m_0, the particle radii r_0 and the spring constant

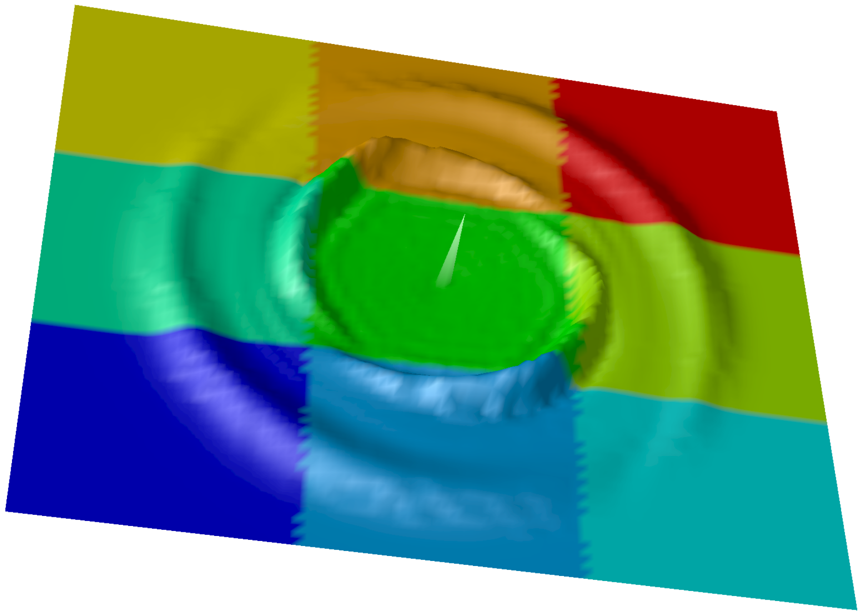

Figure 4: Snapshot of the displacement field generated by P and S-waves propagating in a regular lattice of 96 × 96 particles. The waves are generated by a source described by Equation (4). The simulation was run on 9 nodes, the colors show which part of the lattice was assigned to each node.

of the inter-particle bonds k (Mora et al. 2000 [7]) by the equations

$$v_P = \sqrt{\frac{9}{8}} r_0 \sqrt{\frac{k}{m_0}} \tag{8}$$

$$v_S = \sqrt{\frac{3}{8}} r_0 \sqrt{\frac{k}{m_0}} = \frac{1}{\sqrt{3}} v_P \quad . \tag{9}$$

Using the values of

$$m_0 = 1.0 \tag{10}$$

$$r_0 = 1.0 \tag{11}$$

$$k = 1.0 \quad , \tag{12}$$

the theoretical wave speeds derived from Equation (8) are $v_p = 1.061$ and $v_s = 0.612$ which are in good agreement with the measured values (6), (7).

The correctness of the neighbour search algorithm was verified using small examples and comparing the neighbour lists produced by the algorithm with the theoretically expected neighbour lists.

Performance Evaluation

The main goal of the evaluation of the performance of the parallel implementation of the lattice solid model on different parallel computer systems was to investigate the suitability of the implemented algorithms for large scale parallel simulations. There are a number of commonly used performance measures. The parallel run time, the time elapsed between the moment computation is started and moment the last processor finishes is denoted T_p. The performance gain achieved by parallelizing the program can be described by the speedup S, defined as the ratio between the parallel runtime on a given number of CPUs and the serial runtime (Kumar et al. 1994 [3]), i.e.

$$S = \frac{T_p}{T_s} \quad , \tag{13}$$

where T_s is the runtime of the best serial algorithm for the same problem. Another important performance measure is the efficiency E, giving the ratio between speedup S at a given number of CPUs n and the number CPUs, i.e. describing how well the additional CPUs are used for actual performance gain.

$$E = \frac{S}{n} = \frac{T_p}{T_s n} \tag{14}$$

The speedup S can theoretically never exceed the number of CPUs used. A speedup of more than n on n CPUs could only be achieved if each CPU spends less than $\frac{T_s}{n}$ time running the program. But then a single CPU could emulate the n CPUs and run the program in a shorter time than T_s which would contradict the definition that T_s is the run time for the best serial implementation (Kumar et al. 1994 [3]). Thus, the efficiency E can not be larger than 1.

$$0 \leq E \leq 1 \tag{15}$$

Despite this, speedups larger than the number of processors ($E > 1$), so-called "super-linear" speedup, are sometimes observed in real parallel applications. This is due to the fact that because of the multi-level memory access architecture of modern computers, including one or more levels of high speed cache, the speed of memory access depends on the problem size, i.e. smaller problems, which fit better into the cache result in faster memory access. Thus if a large problem which would not fit into the cache on a given system is partitioned into smaller problems which fit into the cache the result may be an improvement of the per-CPU performance, leading to an apparent "super-linear" speedup.

In order to test the scaling of the force computation algorithm a test was run simulating the propagation of elastic waves in a regular two-dimensional lattice solid. In the tests, the problem size was scaled with the number of processors used, i.e. the amount of computation per processor was constant. This was done in order to avoid variations of single CPU performance with problem size to influence the the results, in particular to avoid apparent super-linear scaling due to cache related effects. A problem size of 65536 particles per processor was used for all tests. The lattice is partitioned so that an area of 256×256 particles is assigned to each processor.

Thus a constant runtime independent of the number of processors would be expected for ideal scaling of the algorithm. However, the amount of communication for each process is dependent of the number of neighbors it has, i.e. how many of the subregions generated by partitioning the problem space share a common boundary with the subregion which is assigned to this process. The communication necessary for each processor is not only dependent on the process topology but it also depends on where in the process structure it is located. However, because the parallel runtime of the program is determined by the runtime of the slowest subprocess, the maximum per processor communication overhead will determine the total runtime of the program. In order to estimate how the communication overhead depends on the number of processors used, two cases have to be considered. First, on computer architectures where the global bandwidth is at least as large as the sum of the bandwidths available to each processors the communication overhead, and thus the parallel runtime, is entirely determined by the maximum per-processor runtime, independently of the total communication by all processors. Assuming the amount of communication between two processors is the same for all pairs of neighbor processors, the theoretical parallel runtime t_r can than be calculated as

$$t_r = t_{calc} + n_{max} t_{comm} \tag{16}$$

where t_{calc} is the computation time, n_{max} is the maximum number of neighbors of a subprocess and t_{comm} is the communication time between a pair of processors. If a partitioning scheme splitting the problem into rectangular 2D-regions is used, this results in an increase of the communication overhead with the number of processors for small numbers of processors due to changes in the process topology and thus the amount of communication per processor. For large numbers of processors ($\geq$ 12) however, the process topology remains the same and thus the communication overhead and the parallel runtime also should be unchanged (Figure 5) .

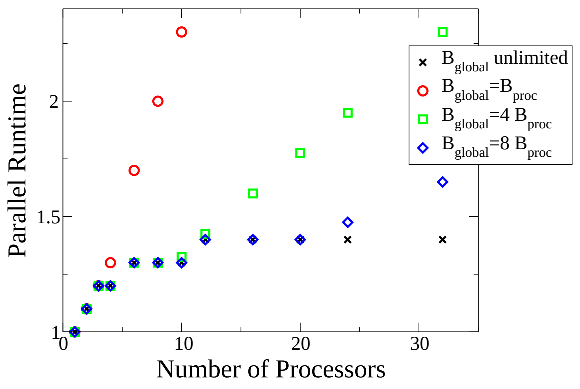

Figure 5: Theoretical parallel runtime for different ratios between the global bandwidth B_{global} and the per-processor bandwidth B_{proc}. The theoretical runtimes are calculated from Equation (18) assuming that $t_{comm} = 0.1 t_{calc}$ and are thus only a qualitative example.

On computer architectures where the global bandwidth is smaller than the sum of the per-processor bandwidths however, the total amount of communication performed by all processors will have an influence on the parallel runtime. If the partitioning scheme described splitting the problem into rectangular 2D-regions is used, the total communication C^{total} necessary in a simulation using $n \times m$ processors can be calculated as

$$C^{total} = C_{proc} \left[n(m-1) + (n-1)m \right] \tag{17}$$

where C_{proc} is the per-processor communication. If, for a given number of processors, the ratio between the total communication and the global bandwidth is smaller than the ration between communication and bandwidth at each processor, i.e. the per-processor bandwidth is saturated before the global bandwidth, the same calculations as for the first case apply (Equation 16). If, however the global bandwidth is saturated before the per-processor bandwidth, the the ratio between the total communication and the global bandwidth will determine the communication overhead. Thus, the theoretical parallel runtime can be calculated as

$$t_r = t_{calc} + \max\left(\frac{C^{total}}{B_{global}}, \frac{C_{proc}}{B_{proc}}\right) \tag{18}$$

where B_{proc} is the per-processor bandwidth and B_{global} is the global bandwidth. A qualitative comparison of theoretical parallel runtimes (Figure 5) for different ratios between the global and per-processor bandwidth shows that while for small numbers of processors the influence of the per-processor communication dominates and thus there is not much difference in parallel runtime depending on the ratio between the global and per-processor bandwidth, for larger numbers of processors used there is a significant difference. In particular, the large increase of the runtime calculated in the case of identical global and per-processor bandwidth shows that the algorithm is not suitable for architectures exhibiting this behaviour. Such architectures would include bus-based SMP [2] systems and clusters using a bus-structured network such as Ethernet [3] or FDDI as interconnect. However, the algorithm appears to be well suited to computer architectures where the communication is not limited by global bandwidth restrictions.

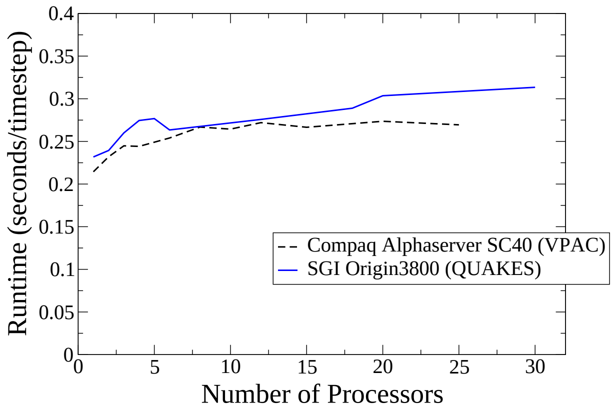

Figure 6: Runtimes for the wave propagation benchmark with constant per-processor problem size of 256×256 particles on a SGI Origin and a Compaq Alphaserver

The Benchmarks performed on up to 30 CPUs on a SGI Origin 3800 (400Mhz MIPS R12000 CPUs) and on up to 25 CPUs on a Compaq Alphaserver SC40 (833Mhz Alpha EV68 CPUs) both show a scaling behavior close to that theoretically predicted for a system without a global communication bottleneck (Figure 5). The runtimes (Figure 6) show an increase for small numbers of CPUs $N \leq 12$ but for larger numbers of CPUs $12 < N < 30$ the runtimes remain nearly constant (Compaq AlphaServer) or show only a small increase (SGI Origin). The parallel efficiency (Figure 7) remains at about 75-80% for these number of CPUs.

In order to test the performence of the algorithms on a very large number of CPUs, additional benchmarks have been performed by Bill Ryder at SGI on an SGI Origin 3800 using a problem size 128×128 particles and a partitioning of the

[2]**S**ymmetrical **M**ulti**P**rocessor

[3]This does not apply to switched Ethernet where each node of the network as acess to the full bandwidth independently of other nodes

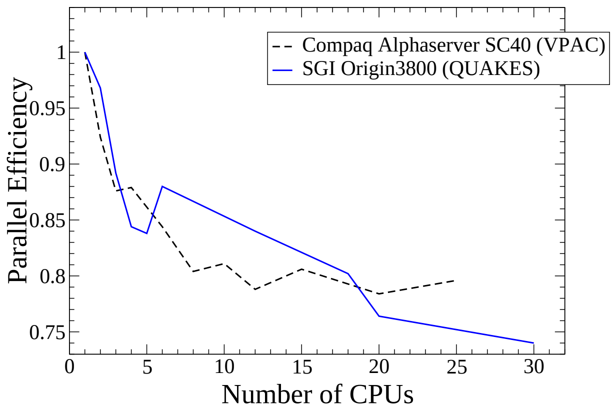

Figure 7: Parallel efficiency for the wave propagation benchmark with constant per-processor problem size of 256×256 particles on a SGI Origin and a Compaq Alphaserver

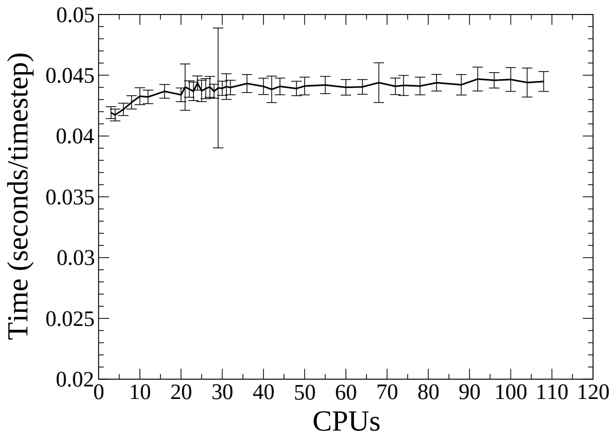

Figure 8: Parallel runtime for the wave propagation benchmark with constant per-processor problem size of 128×128 particles on a 128 processor SGI Origin. The error bars show the standard deviation of the runtime per timestep.

problem in a $1 \times n$ grid. The results have shown that for a constant process topology the runtime remains constant within the resolution of the measurement for ≈ 30 up to the maximum tested 108 CPUs (Figure 8).

Conclusion

The parallel implementation of the Lattice Solid Model shows good scaling for up to 30 CPUs on two different computer architectures (SGI Origin and Compaq Alphaserver), achieving parallel efficiencies $> 75\%$ on ≈ 30 CPUs. Furthermore, benchmarks showing near perfect scaling between 30 and 108 CPUs on the SGI Origin 3800, thus suggesting that the program will run efficiently on very large systems with more than 100 CPUs at least on this architecture, enabling the simulation of large, realistic models.

Acknowledgments

The research was funded by the Australian Research Council, QUAKES and The University of Queensland. The benchmarks would not have been possible without the support of SGI and VPAC, enabling the access to the 128 processor SGI Origin and the Compaq

AlphaServer. Particular thanks go to Mr. Gerald Hofer (SGI Brisbane), Mr. Bill Ryder (SGI New Zealand), Ms. Feng Wang and Mr. David Bannon (VPAC).

References

[1] P. A. Cundall and O. D. A. Strack, 1979, *A Discrete Numerical Model for Granular Assemblies,* Geótechnique, **29,** 47-65

[2] K. M. Frye and C. Marone, 2002 , *The effect of particle dimensionality on granular friction in laboratory shear zones,* Geophys. Res. Lett., **29**

[3] V. Kumar, A. Grama, A. Gupta and G. Karypis, 1994, *Introduction to Parallel Computing* Benjamin/Cummings

[4] K. Mair, K. M. Frye and C. Marone, 2002 , *Influence of grain characteristics on the friction of granular shear zones,* J. Geophys. Res., **107**

[5] P. Mora and D. Place, 1994, *Simulation of the Stick-Slip Instability,* Pure Appl. Geophys., **143,** 61-87

[6] P. Mora and D. Place, 1998, *Numerical Simulation of Earthquake Faults with Gauge: Towards a Comprehensive Explanation for the low Heat Flow,* J. Geophys. Res., **103,** 21067-21089

[7] P. Mora, D. Place, S. Abe and S. Jaumé, 2000 *Lattice solid simulations of the physics of fault zones and earthquakes: the model, results and directions* Geocomplexity and the Physics of Earthquakes, AGU, Washington

[8] J. K. Morgan and M. S. Boettcher, 1999, *Numerical simulations of granular shear zones using the distinct element Method: 1. Shear zone kinematics and the micromechnics of localization,* J. Geoph. Res., **104,** 2703–2719

[9] D. Place and P. Mora, 1997, *Towards Large Scale Simulations of the Physics of Rocks and Earthquakes,* QUAKES Report #2, 209-219, The University of Queensland

[10] D. Place and P. Mora, 1999, *The Lattice Solid Model to Simulate the Physics of Rocks and Earthquakes: Incorporation of Friction,* J. Comp. Physics, **150,** 332-372

[11] D. Place, 1999, *A Refined Lattice Solid Model to Simulate Earthquakes and Localization Phenomena Using Parallel Computers,* PhD Thesis, The University of Queensland

[12] D. Place and P. Mora, 2000, *Numerical Simulation of Localisation Phenomena in a Fault Zone,* Pure Appl. Geoph., **157** 1821-1845

[13] D. Place, 2001, *A random lattice solid model for simulation of fault zone dynamics and fracture processes,* Bifurcation and Localization Theory for Soils and Rocks '99, AA Balkema, Rotterdam/Brookfield

[14] D. Place, F. Lombard, P. Mora and S. Abe, 2002, *Simulation of the microphysics of rocks using LSMEarth,* Pure Appl. Geophys. in print

[15] D. C. Rapaport, 1995, *The Art of Molecular Dynamics Simulations,* Cambridge University Press

[16] H. Sakaguchi and H. B. Mühlhaus, 2000, *Hybrid Modelling of Coupled Pore Fluid-solid Deformation Problems,* , Pure Appl. Geophys.**157,** 1889-1904

[17] M. Winter, D. Place and P. Mora, 1997, *Incorporation of Particle Scale Rotational Dynamics into the Lattice Solid Model* , QUAKES Report #2, 221-229