Scaling Evalutation of the Lattice Solid Model on the SGI Altix 3700

Shane Latham, Steffen Abe and Matt Davies
ACcESS MNRF
Earth Systems Sciences Computational Centre
The University of Queensland
Brisbane, QLD, Australia, 4072
Phone: +61-7 3346 8790, +61-7 3346 9783, +61-7 3365 9773
Fax: +61-7 3365 7347
slatham@esscc.uq.edu.au, steffen@esscc.uq.edu.au, matt@esscc.uq.edu.au

Abstract

The Lattice Solid Model is a particle based method which has been successfully employed for simulating the fracturing of rocks, the dynamics of faults, earthquakes and gouge processes. However, results from initial simulations demonstrate that models consisting of only thousands of particles are inadequate to accurately reproduce the micro-physics of seismic phenomenon. Instead, models with millions or tens of millions of particles are required to produce realistic simulations. Parallel computing architectures, such as the SGI Altix 3700, provide the opportunity to solve much larger computational problems than traditional single processor systems. In order to take advantage of high performance systems, a Message Passing Interface version of the Lattice Solid Model has been implemented. Benchmarks, presented in this paper, demonstrate an 80% parallel efficiency for the parallel Lattice Solid Model on 128 processors of the SGI Altix 3700. These results, for a two–dimensional wave propagation problem, indicate the potential for the Lattice Solid Model to simulate more computationally challenging three-dimensional geophysical processes.

1. Introduction

Most simulations based on particle dynamics models such as the Discrete Element Model [4] and the Lattice Solid Model (LSM) [10] have been performed using a relatively small number of particles. Simulations of processes requiring a large number of time steps, such as gouge shear, have typically used between several hundreds and a few thousand particles [12, 18]. Simulations of processes which require a relatively small number of time steps, such as compression and fracturing of solids [15], have used models containing tens of thousands of particles.

However, recent laboratory experiments have shown that the behavior of fault gouges has a strong dependence on the particle shape and size distribution [8]. These experiments have also shown that there is a significant difference in fault gouge behaviour between 2D and 3D models [5]. In addition, [5, 8] suggest that those results explain the differences between results obtained from laboratory experiment and those from current numerical simulations. In order to perform realistic simulations, which provide results directly comparable to laboratory experiments, it will be necessary to use three–dimensional simulation models with particle shapes and size ranges comparable to those used in the laboratory experiments. Particle sizes between 1-800μm, used by [8], would suggest simulation models containing at least several million particles. This is currently not feasible using serial simulation software on a single CPU.

Results from Section 5 demonstrate high parallel efficiency achieved, using a Message Passing Interface (MPI) implementation of the LSM, when solving a two–dimensional wave propagation problem involving approximately eight million particles. These results demonstrate the potential for this parallel implementation of the LSM to solve more computationally challenging three–dimensional geophysics problems.

The remainder of the paper is organised as follows. Section 2 presents an overview of the LSM and a discussion of the parallel implementation. Section 3 gives a brief overview of the Altix 3700 architecture. Section 4 describes the two–dimensional wave propagation problem which is solved to obtain the benchmark results presented in Section 5.

2. Overview of the Lattice Solid Model

2.1. Model

The LSM [10, 17] is a particle based model similar to the Discrete Element Model (DEM) [4]. 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 neighbours by imparting elastic and frictional forces. The particles can be linked together by elastic bonds (see Figure 1), in which case the elastic forces are attractive or repulsive, depending on whether the particles are closer or more distant than an equilibrium distance. The *linked* force $\mathbf{F}_{ij}^{linked}$ which particle i exerts on particle j is expressed as

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

where k_{ij} is the spring constant for the elastic interaction between particles i and j, 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. The link between particle i and particle j is 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 and is given by

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.

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

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

An intrinsic friction between particles has been incorporated in the model [17]. Two unbonded interacting particles can be in static or dynamic frictional contact. The force on particle i, due to the dynamic frictional contact with particle j, is expressed as

$$\mathbf{F}_{ij}^{D} = -\mu F_{ij}^{n} \mathbf{e_{ij}}^{T} \ , \quad (3)$$

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 [11]. The total force $\mathbf{F}_i$ on particle i can be calculated as

$$\mathbf{F}_i = \sum_{(ij)\in L} \mathbf{F}_{ij}^{linked} + \sum_{(ij)\in T} \left(\mathbf{F}_{ij}^{free} + \mathbf{F}_{ij}^{D} \right) \ , \quad (4)$$

where L is the set of linked pairs of particles and T is the set of all pairs of particles which are in contact but not linked.

Rotational dynamics can be simulated in the LSM both by per-particle rotation [20, 21] where each single particle has angular velocity $\boldsymbol{\omega}$ and momentum $\mathbf{I}$ or by collective rotation of groups of irrotational particles [13].

2.2. Parallel Implementation

There are two 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 explicitly distributed approach which also distributes the thread of control.

While the data-parallel approach has been successful using a relatively small number of CPUs [13, 16], it is not well suited for the parallelization of the LSM on computer systems with large numbers of CPUs. The results of performance and scalability tests [13] show that a purely data-parallel implementation of the LSM is likely to lead to a program which contains significant serial overhead. Thus the second, explicitly distributed approach has been taken for the parallel implementation of the LSM. Explicit message passing using MPI [9] as the underlying communication library has been used to implement the inter-process communication. MPI is a well–defined, open standard, and implementations exist which are tailored for optimum performance on a wide variety of high performance and desktop architectures.

The parallel process structure follows a modified *master-worker* model. A *master* process provides a high level of control and external communication I/O facilities. The *worker* or *slave* processes perform the computational work. This design minimizes the computation performed by the master process and also greatly reduces the amount of communication between master and worker processes. The master process performs high level control and each worker process undertakes 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 LSM. The communication between the parallel processes are implemented using MPI.

For molecular dynamics algorithms, which are closely related to the LSM, three approaches for the distribution of the work between the parallel processes have been suggested [19]. The first approach is to partition only the computation and share all data between the processors, i.e. each processor has direct access to all data. An efficient implementation of this approach, is appropriate for shared memory computers. On distributed memory systems it would require the duplication of the whole data set on each node. A second possibility is a partitioning scheme based on the particles, assigning each particle to a particular processor for the entire run time of a simulation, irrespective of the position of the particle. This leads to the efficient use of memory, but the handling of interactions between particles residing on different processors incurs a large communication cost. The third choice is to partition the problem spatially between processors. A worker processor will then perform the computations for all the particles which are located within the subregion at the current time step. The communication cost of this approach is potentially smaller than in the second approach as interactions between particles assigned to different processors can only occur along the boundaries of the subregions. This third approach has been used in the current parallel implementation of the LSM.

The distribution of the particles, amongst the worker processors, is based on a spatial partitioning of the geometrical domain. In addition 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. Due to the short spatial range of the interactions, the number of duplicated boundary-particles is small in comparison to the total number of particles. The forces due to interactions which are assigned to more than one processor are computed by each processor. This leads to a small amount of duplicated computation, but reduces the communication cost.

3. The SGI Altix 3700 Architecture

The Earth Systems Simulator (ESS) situated at the University of Queensland is a 208-processor SGI Altix 3700 cache-coherent NUMA (ccNUMA), distributed shared-memory (DSM) "supercluster". The system hardware is based on the SGI Origin S^2MP architecture (see [7]) with SGI NUMAflex interconnect technology (first used in the SGI Origin 3000 [3]), the Intel Itanium 2 processor, DDR SDRAM DIMMs and supports a variety of PCI cards and adaptors. The system software administering the system hardware is based on the SGI Advanced Linux Environment with SGI ProPack, configured to operate as one node under a single system image.

The 208 processor elements of the ESS are all Itanium 2 Madison processors, each with a 3Mb L3 write-back cache, 256Kb write-back L2 cache and 16Kb write-through L1 cache, all of which are located on-die. The ESS has 2Gb of memory local to each node board.

The NUMAflex network of the ESS is implemented in a dual-plane "fat tree" topology interconnecting the basic building-block components (or "bricks") in a modular manner as shown in Figure 4. The compute brick (or "C-brick") consists of two node boards, each supporting local SDRAM memory and two Itanium 2 processors connected to an ASIC via a single front side bus. The ASIC acts as a crossbar between the local memory, processors, network and I/O interfaces and is internally connected to the opposing ASIC of the other node board in the C-Brick via a 6.4Gb/s full duplex NUMAlink 4 interconnect. The ASIC is also externally connected to a router brick (or "R-brick") by a 3.2Gb/s full duplex NUMAlink 3 interconnect. The R-brick in-turn acts as a high-speed switch, routing network packets between the C-bricks and other system components including memory bricks (or "M-brick"s), I/O expansion bricks (or "IX-brick"s), PCI expansion bricks (or "PX-brick"s), data bricks (or "D-brick2"s) and other R-bricks in the network. Refer to [3] for further information on the topology.

The global shared memory addressing of the Altix 3700 is implemented in the C-Brick ASIC, interfacing the snooping operations of the Itanium 2 processor to NUMAflex protocol which is directory-based. The ASIC maintains an internal directory containing the most recent cache coherency information for each cache-line the processor fetches. For each cache-line, a bit vector flagging which other node boards have a copy of the particular cache-line is maintained. This arrangement permits direct cache-line status transactions and updates, implemented by way of an invali-

Figure 4. The Dual Plane Fat Tree Topology

dation strategy [22]. An additional 3% of local memory is reserved for inactive directory entries not found in the ASIC. Cache-line data and directory information are loaded simultaneously, reducing delays in the coherency scheme [22].

4. 2D Wave Propagation Problem

A two–dimensional wave propagation problem was solved for increasing numbers of particles to examine the scaling properties of the parallel LSM implementation on the SGI Altix 3700. A displacement source-particle, located in the center of the lattice, was used to propagate waves through the particle array. The type of source displacement was similar to the one used in [14] to test a serial implementation of the LSM and similar to the type used in [1, 2] to test scaling properties of the LSM implementation on other high-performance architectures. The displacement d_x in the x-direction and the displacement d_y in the y–direction of the source particle are given by

$$d_x = a_x e^{\frac{(t-t_{0,x})^2}{b_x}} \quad , \tag{5a}$$

$$d_y = a_y e^{\frac{(t-t_{0,y})^2}{b_y}} \quad . \tag{5b}$$

The constants chosen for the timing tests were $a_x = a_y = 0.1$, $b_x = b_y = 0.5$, $t_{0,x} = 5.0$ 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. Particles of radius $r_0 = 1$ and mass $m = 1$ were initially ($t = 0$) arranged in a two–dimensional regular triangular lattice. Figure 5 illustrates the radial propagation of the P-wave and S-wave, from the source particle in the centre of the lattice, at $t = 100$.

Figure 5. Snapshot of the displacement field generated by P and S-waves in a regular lattice of 256 × 256 particles. The waves are generated by the source displacement given in equations (5).

5. Performance Results

The results from the performance tests enable an assessment of the parallel LSM and its potential to solve computationally demanding three-dimensional geophysics problems on the Altix 3700. This section examines runtimes and the parallel efficiency of the LSM when simulating the two-dimensional wave propagation problem described in Section 4.

The parallel run time T_p is the time elapsed between the moment computation is started and the moment the last processor finishes computation. The performance gain achieved by parallelizing the program can be described by the speedup S [6], which is defined as the ratio between the parallel runtime on a given number of CPUs and the serial runtime T_s. The speedup is expressed as

$$S = \frac{T_s}{T_p} \ , \tag{6}$$

where T_s is the runtime of the optimal serial algorithm for the same problem. The parallel efficiency E is the ratio between speedup S at a given number of CPUs n and the number of CPUs. The efficiency value describes how well the additional CPUs are used for actual performance gain,

$$E = \frac{S}{n} = \frac{T_s}{T_p n} \ , \quad 0 \leq E \leq 1 \ . \tag{7}$$

The inequality is derived from the fact that the speedup S can theoretically never exceed the number of CPUs [6]. In practice, however, speedups larger than the number of processors ($E > 1$), so-called "super-linear" speedups, are sometimes observed in parallel implementations. This is related to CPU hardware architectures involving multiple levels of high-speed cache. Smaller datasets, which can reside entirely within a high-speed cache have better performance characteristics than large datasets which can only partially reside within a high-speed cache. Super-linear speedup occurs when a large dataset is partitioned into a number of smaller datasets which are capable of residing entirely within the caches of CPUs in the parallel architecture.

In order to test the scaling of the parallel LSM implementation, simulations were run for the two–dimensional elastic wave propagation problem (Section 4) for increasing numbers of particles. As in [1, 2], the problem size was scaled with the number of processors used, i.e. the area of the rectangular region assigned to a processor was constant. This problem scaling was performed in order to avoid potential super-linear scaling due to cache related effects. The lattice is geomet-

rically partitioned so that an area containing 256×256 particles is assigned to each processor.

Figure 6. Partitioning of spatial domain amongst processors. Each square represents a worker processor and the digit indicates the number of neighbours of the processor.

A constant runtime, independent of the number of processors, would be desired for ideal scaling of the algorithm. However, the amount of communication for each processor is dependent on the number of its neighbour processors. The number of neighbours for a processor is determined by the partitioning of the problem geometry. Figure 6 illustrates some examples of the partitioning of the problem space amongst processors for different problem sizes. This figure shows the spatial partitioning of particles amongst processors and the communication between neighbouring processors. Each processor is represented as a rectangular area (an area containing 256×256 particles).

Figure 7 plots the runtime per timestep of the particle force calculation (Equation (4)) for the two–dimensional wave propagation problem. The error bars indicate the minimum, mean and maximum runtime per timestep over the 500 timesteps of each simulation on the indicated number of CPUs. The times are

Figure 7. Runtimes for the particle force calculation in Equation (4) of the wave propagation simulation. Error-bars indicate minimum, mean and maximum runtime per timestep over the 500 timesteps of each simulation.

recorded by the master process for each timestep. The master process directs the slave processes to perform the force calculations and waits until all slave processes complete the computation. Therefore, each runtime is the longest time taken, over all slave process, to complete the force calculation for a single timestep. The force calculation is CPU-bound, involving no particle–data communication between worker processes. The force calculation times grow rapidly up to 16 processors because of the increasing maximum number of particles per processor over all worker processors. Each worker processor contains 256×256 particles of its assigned rectangular region, plus an "overlap" of duplicated boundary particles from neighbouring rectangular regions which are linked to particles within the assigned region. In Figure 6, it can be seen that simulations for 16 or more CPUs contain processors that have the maximum 4 neighbours. The mean runtimes for 32, 64 and 128 processors increase by less than 2% of the 16 processor mean runtime.

Figure 8. Runtimes for the exchange of boundary-particle data. Error-bars indicate minimum, mean and maximum runtime per timestep over the 500 timesteps of each simulation.

Figure 8 plots the runtime per timestep for the exchange of boundary particle data. Again, the error bars indicate minimum, mean and maximum runtime per timestep over 500 timesteps. The runtime for exchanging the boundary–particle data consists of both communication time and processing time for physically sending and storing the boundary–particle data. In the single processor case, there is no boundary–particle data communication with neighbours. For 2 processors there is the minimum possible amount of boundary–particle data to exchange. The 4 processor simulation involves each processor exchanging boundary-particle data with two neighbouring processors. In the 8 processor case, four of the processors have three neighbours. Another increase in boundary–particle data exchange occurs for the 16 processor simulation. In this case and the 32, 64 and 128 processor cases, there are processors which must exchange boundary information with four neighbouring processors. For simulations involving 16, 32, 64 and 128 processors, the rate of increase in boundary–particle data exchange time is greatly reduced.

A qualitative discussion of the scalability of the LSM parallel communication implementation for different communication architectures is given in [1]. There it is revealed that the parallel implementation is unlikely to scale well for architectures where the global bandwidth is less than the sum of the per-processor bandwidths. From the above timing results for the boundary-particle data exchange, it appears that parallel LSM communication implementation scales well with regard to the Altix 3700 communication architecture. Given the 3.2 GB/sec bidirectional bandwidth between nodes, and, for 256×256 particles per processor, neighbouring processors exchange less than 100KB of boundary-particle data, it is not expected that actual communication of data forms any significant overhead.

Figure 9 plots the efficiency for the combined force

Figure 9. Parallel efficiency.

calculation and boundary–particle data exchange. The efficiency decreases for increasing number of processors due to the increases in force calculation times and communication times for larger numbers of processors. The plot shows an efficiency of approximately 80% for 128 processors ($256 \times 256 \times 128 = 8388608$ particles).

6. Conclusion

The parallel implementation of the LSM shows good scaling for up to 128 processors on the SGI Altix 3700, achieving a parallel efficiency $\approx 80\%$ for a problem involving over eight million particles. To date, this is the largest number of particles for which a wave propagation simulation has been run using the parallel LSM implementation. The experimental results are encouraging and suggest that the parallel implementation of the LSM, running on the Altix 3700, will be capable of performing three–dimensional simulations involving millions of particles and enable realistic simulation of geophysical processes.

Acknowledgments

Funding support for this work is gratefully acknowledged. Project work is supported by the Australian Computational Earth Systems Simulator Major National Research Facility, The University of Queensland and SGI. The ACcESS MNRF is funded by the Australian Commonwealth Government and participating institutions (University of Queensland, Monash University, Melbourne University, VPAC, RMIT) and the Victorian State Government. Computations were made using the ACcESS MNRF supercomputer, a 208 processor 1.1 TFlops SGI Altix 3700, which was funded by the Queensland State Government Smart State Research Facility Fund and SGI.

References

[1] S. Abe. *Investigation of the Influence of Different Microphysics on the Dynamic Behaviour of Faults using the Lattice Solid Model.* PhD thesis, University of Queensland, 2001.

[2] S. Abe, D. Place, and P. Mora. A parallel implementation of the lattice solid model for the simulation of rock mechanics and earthquake dynamics. In A. Donnellan and P. Mora, editors, *3rd ACES Workshop Proceedings*, pages 201–212. ACES, 2003.

[3] D. Brownell. *SGI Altix 3000 User's Guide*, 2003.

[4] P. A. Cundall and O. A. D. Strack. A discrete numerical model for granular assemblies. *Geótechnique*, 29:47–65, 1979.

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

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

[7] J. Laudon and D. Lenoski. The SGI Origin: A ccNUMA highly scalable server. In *Proceedings of the 24th International Symposium on Computer Architecture (ISCA '97)*, pages 241–251, Denver, Colorado, 1997.

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

[9] Message Passing Interface Forum. *MPI-2: Extensions to the Message-Passing Interface*, 1997.

[10] P. Mora and D. Place. Simulation of the stick-slip instability. *Pure Appl. Geophys.*, 143:61–87, 1994.

[11] P. Mora and D. Place. Numerical simulation of earthquake faults with gouge: Towards a comprehensive explanation for the low heat flow. *J. Geophys. Res.*, 103:21067–21089, 1998.

[12] J. K. Morgan and M. S. Boettcher. 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, 1999.

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

[14] D. Place. A random lattice solid model for simulation of fault zone dynamics and fracture processes. In H. B. Mühlhaus, A. Dyskin, and E. Pasternak, editors, *Bifurcation and Localization Theory for Soils and Rocks '99.* Balkema, Rotterdam/Brookfield, 2001.

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

[16] D. Place and P. Mora. Towards large scale simulations of the physics of rocks and earthquakes. QUAKES Report #2, The University of Queensland, 1997.

[17] D. Place and P. Mora. The lattice solid model to simulate the physics of rocks and earthquakes: Incorporation of friction. *J. Comp. Physics*, 150:332–372, 1999.

[18] D. Place and P. Mora. Numerical simulation of localisation phenomena in a fault zone. *Pure Appl. Geoph.*, 157:1821–1845, 2000.

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

[20] H. Sakaguchi and H. B. Mühlhaus. Hybrid modelling of coupled pore fluid-solid deformation problems. *Pure Appl. Geophys.*, 157:1889–1904, 2000.

[21] M. Winter, D. Place, and P. Mora. Incorporation of particle scale rotational dynamics into the lattice solid model. QUAKES Report #2, The University of Queensland, 1997.

[22] M. Woodacre, D. Robb, D. Roe, and K. Feind. *The SGI Altix 3000 Global Shared-Memory Architecture.* Silicon Graphics, 2003.