1998

Automatic Granularity Control For Load Balancing Of Concurrent Particle Simulations

Marc Rieffel
Syracuse University, Scalable Concurrent Programming Laboratory, marc@scp.syr.edu

Stephen Taylor
Syracuse University, Scalable Concurrent Programming Laboratory, steve@scp.syr.edu

Jerrell Watts
Syracuse University, Scalable Concurrent Programming Laboratory, jwatts@scp.syr.edu

AUTOMATIC GRANULARITY CONTROL FOR LOAD BALANCING OF CONCURRENT PARTICLE SIMULATIONS

Marc Rieffel, Stephen Taylor, and Jerrell Watts
Scalable Concurrent Programming Laboratory
Syracuse University, Syracuse, NY 13244
e-mail: {marc,steve,jwatts}@scp.syr.edu

KEYWORDS

Dynamic load balancing, granularity, irregular applications, particle simulations, concurrent computing.

ABSTRACT

This paper demonstrates the use of automatic granularity control as part of dynamic load balancing for irregular, particle-based simulations. Performance optimization techniques are considered in the context of a concurrent Direct Simulation Monte Carlo method used to study the rarefied gas flow inside three-dimensional plasma reactors. Several computational techniques are used to reduce the overall time to deliver realistic three-dimensional results. The effectiveness of dynamic load balancing and granularity control are presented for large-scale simulations on distributed-memory multicomputers.

INTRODUCTION

Recent advances in microprocessor performance have been driven primarily by improvements in manufacturing technology. New processes and equipment have paved the way for smaller feature sizes and larger wafer sizes. These, in turn, have facilitated the production of microprocessors with more transistors, operating at lower voltages and with higher clock rates. One of the key pieces of equipment in microelectronics manufacturing is the *plasma reactor*, used in 30 to 40 percent of processing steps. Moreover, plasma processing equipment accounts for approximately 20 percent of the cost of each semiconductor manufacturing facility. The cost of these production facilities is escalating, as are the research and development costs associated with the introduction of each new generation of processing technology. It is widely recognized that computational tools for modeling plasma reactors can significantly reduce the costs of validating new reactor designs and can help to improve manufacturing processes.

Due to the extensive computational requirements of the simulation technique, and the imperative of providing results on industrial timescales, the use of large-scale concurrent computer architectures is necessary. The irregular nature of the particle simulations results in complicated load characteristics that may vary over the course of a computation. It is impossible for a static load balancing method to obtain efficient utilization on large numbers of processors. Dynamic load balancing techniques are therefore used in the present work. One critical factor load balancing is the granularity of the computation. Just as static load balancing is ineffective, static partitioning and granularity specification are also inadequate. This work presents and evaluates techniques for *automatic granularity control* as applied to large-scale particle-based simulations.

SIMULATION TECHNIQUE

The simulation technique integrates a variety of ideas taken from computational fluid dynamics and finite-element methods. A central aim is to exploit existing industrial tools, already in use by process engineers, to shorten the design cycle to acceptable engineering timescales. A three-dimensional geometry definition is taken directly from CAD/CAM descriptions already available to process engineers. An unstructured tetrahedral grid is then constructed using automatic grid generation techniques. This grid is subsequently partitioned for execution on multiprocessor systems. Scalable concurrent algorithms are then used to reduce the numerical simulation time. Adaptive gridding is used to automatically maintain the accuracy of the simulation. Dynamic load balancing and granularity control

are used to maximize processor utilization in the presence of both grid adaption and dynamic flow variations. Finally, simulation results are analyzed using standard CFD visualization tools.

The Direct Simulation Monte Carlo (DSMC) method solves the Boltzmann equation by simulating the behavior of individual particles. Since it is impossible to simulate the actual number of particles in a realistic system, a smaller number of simulation particles are used, each representing a large number of real particles. Statistical techniques are employed to reproduce the correct macroscopic behavior. Computational grid cells are initially filled with simulation particles according to density, temperature, and velocity specifications. The simulation takes discrete steps in time, during which a *transport model* is used to move particles, a *collision model* is used for particle-particle interactions, and a *boundary model* is used for interactions between particles and surfaces. Macroscopic properties, such as density and temperature, are computed by appropriate averaging of particle properties including mass and velocity.

Transport Model. The transport model is concerned with moving particles through the computational grid for a specified period of time. It uses ray-tracing techniques to determine the paths of particles during each timestep

Collision Model. The collision model characterizes particle-particle interactions. Only collisions between particles in the same cell must be considered, and collisions can be performed independently within each cell and concurrently in each partition.

Boundary Model. When configuring a simulation, a surface type is specified for each surface of the computational grid. The surface type of a face determines particle-surface interactions on that face. The three typical surface types are *inflow, outflow*, and *accommodating*, modeled according to standard DSMC techniques for gas-surface interactions. During grid partitioning, an additional surface type, *partition*, is created to represent shared boundaries between partitions. A particle arriving at a partition surface is sent to the appropriate neighboring partition.

The concurrent algorithm, executed by each partition of the computational grid, is as follows.

1. Initialize partitions according to initial conditions (locally)
2. While more steps are necessary
   (a) Calculate new particle positions (locally)
   (b) Exchange particles between partitions (local communication)
   (c) Collide particles (locally)
   (d) Compute global information, such as the total number of simulated particles (global communication)
   (e) Determine load imbalance
   (f) Adjust granularity
   (g) Balance load
3. Conclude computation

For the most part, particle transport is local within a partition (2a), though a particle may move across a cell face on the boundary between two partitions. In this case, it is communicated to the appropriate neighboring partition (2b). In a single timestep, a particle may cross several partition boundaries and thus require several rounds of communication. In order to improve communication efficiency, all particles exchanged between a given pair of partitions are combined into a single message. Once all of the particles have been placed in their new cell locations, the collision and boundary models are employed independently within each cell (2c).

The numerical technique, Concurrent Direct Simulation Monte Carlo, is presented in (Rieffel 1997). Validation studies, for neutral flow and gas mixtures in a variety of configurations, are considered in (Gimelshein 1996). A parametric study of reactor configurations is discussed in (Rieffel 1998). The present work presents automatic granularity control techniques and their application to particle simulations on complex three-dimensional geometries.

DYNAMIC LOAD BALANCING

No initial partitioning of a computational grid can provide optimum load balance, or optimal granularity, throughout a dynamic simulation. Dynamic load balancing and granularity control are therefore essential for efficient use of modern computational resources, especially on heterogeneous networks where machines have differing memory and performance characteristics. Portions of the grid must be decomposed at runtime, and exchanged between computers in order to achieve load balance. Exchanges must be selected in order to maintain locality where possible.

The load balancing mechanism is based on the concept of heat diffusion, which provides a scalable, correct mechanism for determining how much work should be

migrated between computers, including computers with different processing capabilities or external workloads. Heat diffusion only gives the ideal work transfer, however; to meet that ideal, neighboring computers must exchange partitions. The selection of which partitions to exchange may be guided by both the sizes of the partitions involved as well as the effect a partition's movement would have on its communication with other partitions. If there are too few or too many partitions in the system, granularity management routines are used to increase or decrease the number of partitions. The end-result is a five-step methodology for load balancing a computation (Watts 1996; Watts 1997):

1. **Load measurement:** The load of each computer is determined, by measuring its resource usage.
2. **Load imbalance detection and profitability calculation:** Based on the total load measured at each computer, the efficiency of the computation is calculated. Load balancing is undertaken only if its estimated cost is exceeded by the estimated reduction in run time that would result from load balancing.
3. **Ideal load transfer calculation:** Using the load quantities measured in the first step, computers calculate the ideal amount of load that they should transfer to or from their neighbors.
4. **Transfer quantity satisfaction:** This phase may be repeated several times until the transfer quantities have been adequately met:
   (a) **Partition selection:** Using the load transfer quantities calculated previously, partitions are selected for transfer or exchange between neighboring computers.
   (b) **Granularity adjustment:** If the granularity is so coarse that not all transfers can be adequately satisfied, partitions may be divided to increase the options available.
5. **Partition migration:** Once the partitions have migrated to their final locations, any data structures associated with those partitions are transferred from their old locations to their new locations, and the computation resumes.

DYNAMIC GRANULARITY CONTROL

A central component of the load balancing approach is the automatic granularity control technique, discussed in the following section. In the transfer quantity satisfaction phase of load balancing, the partitions may be so large, or coarse-grained, that it is impossible to balance the load. It is then necessary to split the partitions into smaller partitions, resulting in a finer granularity. A partition is split if its corresponding load is greater than a certain fraction of the average load. If the division of partitions results in a better, but still inadequate load balance, the threshold is lowered so that more partitions are divided. This continues until an adequate load balance is achieved, until no benefit results from finer granularity, or until partitions can no longer be split.

When the load balancing method determines that a partition must be split, the application is responsible for achieving that split. The grid cells in the partition are traversed in order to compute a bounding box around the partition. The bounding box is then divided into the desired number of new partitions, so as to minimize the surface area of the new bounding boxes. This process can be completed in time proportional to the number of grid cells, though it does not necessarily guarantee minimum communication or even division. New connections are created between the newly-created partitions, and connections to neighboring partitions are updated.

Application support is required for splitting partition-level data structures. For example, counts of the numbers of particles and cells in each partition must be updated. Grid cells and particles are unaffected by partition splitting. Cell faces that lie on the border between the two new partitions must be replicated, and face-level data structures updated accordingly.

Note that the bulk of the granularity control operations are local. Communication is only required for updating connections between the new partitions and their neighbors. This allows for rapid granularity adjustment even on large concurrent computers. For the simulations considered in this study, the process of granularity control was completed in same amount of time as several simulation steps. As typical simulations require tens or hundreds of thousands of timesteps, and load balancing only takes place about once every thousand steps, this cost is negligible.

SIMULATION RESULTS

For the purpose of this study, simulations of the GEC Reference Cell Reactor have been considered. The computational grid for this reactor is shown in Figure 1. Simulations were performed using Argon at an operating pressure of 13.3 Pa (100 mTorr). The reactor walls

are assumed to be accommodating at 300K.

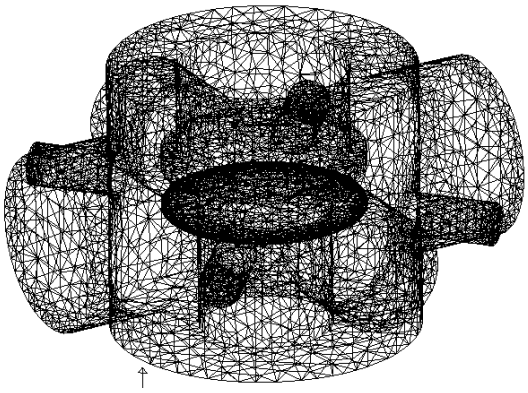

Figure 1: Computational Grid for the GEC Reference Cell Reactor

In order to demonstrate the applicability of these technique to a problem of industrial relevance, a full-flow simulation of the GEC cell was completed. For this simulation, gas was injected through a small port on the side of the reactor, and removed through the large port on the opposite side. Inflowing gas was at 300K, with a particle number density of $2.5 \times 10^{22} m^{-3}$ and a speed of 37.6 m/s. Note that this is a completely three-dimensional flow configuration. This simulation was completed in approximately 2 weeks on a 12-processor Avalon A12 with 500 MB RAM per processor, using 2 million grid cells and 16 million particles.

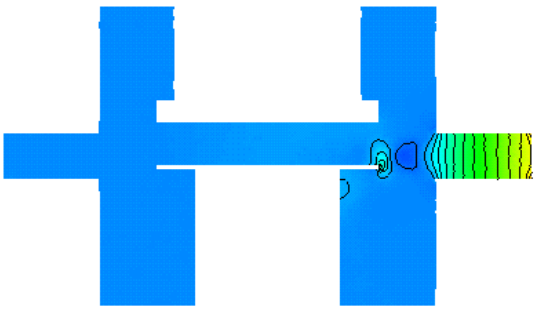

Figure 2: Pressure in vertical plane

Figure 2 shows gas pressure in a vertical slice through the reactor, perpendicular to the wafer. The wafer appears as a thin horizontal surface in the middle. Twenty contours are drawn from 0 to 25 Pa. The prominent features in this plot are the high pressure in the inflow region and a slightly lower pressure in the exhaust region. A boundary layer can also be seen in the inflow pipe, and a shock has formed on the leading edge of the wafer.

PARALLEL PERFORMANCE

Practical DSMC simulations typically involve two phases: startup and statistics-collection. During the startup phase, macroscopic properties change over time as the solution emerges. Once the macroscopic parameters have converged, statistics are collected over several thousand steps in order to obtain smooth and accurate results. During the startup phase, a small number of particles are used (only as many as are required to reach a correct solution). As the number of processors is increased, there is no need to increase the number of particles.

During the statistics-collection phase, however, the goal is to maximize the number of "samples", where a sample is essentially one timestep for one particle. As the ratio of particles to cells increases, the computational overhead associated with each cell is amortized over a larger number of "useful" particle computations. Maximizing the particle processing rate therefore results in the fastest wall-clock-time convergence. On distributed-memory machines, the use of additional processors makes possible the use of additional particles. It is therefore useful to consider a *scaled speedup*, where the number of particles used is proportional to the number of processors.

In both phases of a computation, the rate of productive work can be measured and compared in terms of the number of particles that can be simulated in a given amount of time. Because of the reduced overhead, this processing rate can actually increase super-linearly with the number of processors. This is particularly true on machines with small amounts of memory per processor, where single-processor simulations are only possible with very small numbers of particles. While this metric of performance may be misleading from an algorithmic-scalability perspective, it is nevertheless a meaningful measure of the amount of "useful work" that can be achieved on existing platforms.

In order to investigate the effectiveness of dynamic load balancing with automatic granularity control, a series of GEC simulations was conducted on the Cray T3D. A high-pressure (13.3 Pa / 100 mTorr), uniform-flow case was considered. Due to the relatively large size of the grid (140,000 grid cells), and the small amount of memory per processor (32 MB), this problem could not be run on fewer than 16 processors. For this reason, the uniprocessor speed could not be determined

exactly. An estimate of the uniprocessor speed was obtained by running the full uniprocessor case on one Avalon A12 processor (with 512 MB RAM), then timing small uniprocessor test cases on both the A12 and the T3D. The T3D uniprocessor time was then computed as the A12 time scaled by the ratio of times for the small problem on the two machines. Based on several different tests, this figure is believed to be accurate to within 10%.

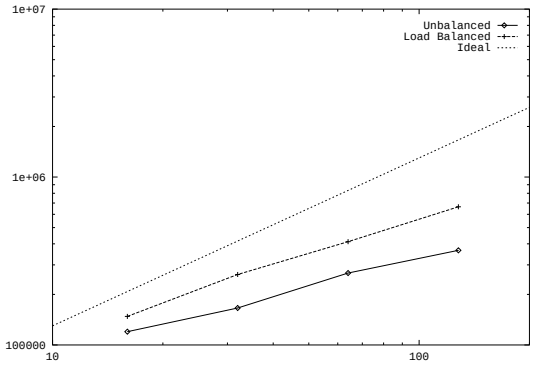

Figure 3: Performance on the unscaled GEC problem

Simulations were conducted on varying numbers of processors for both scaled and unscaled cases. For the unscaled simulations, 200,000 particles were used. The unscaled results are shown in Figure 3. Three lines are shown here: the measured speed without load balancing or automatic granularity control; the measured speed with load balancing and granularity control; and the ideal speed, computed by scaling the estimated uniprocessor speed. For these tests, the combination of load balancing and granularity control improved performance by 50-100%, but performance still dropped below 40% of ideal on 128 processors. This can be attributed to the small number of particles per processor for the unscaled case on large numbers of processors. As the number of particles per processor decreases, the fraction of time spent on computational overhead increases, resulting in poor scaling.

Several scaled-particle simulations were also conducted, using 12,500 particles per processor. These results are shown in Figure 4, again with unbalanced, balanced, and ideal speeds. Here, the unbalanced performance quickly drops to 26% of ideal, but with load balancing and granularity control, performance remains above 70% of ideal. On 128 processors, the combination of load balancing and granularity control resulted in a 3x performance improvement, resulting in performance

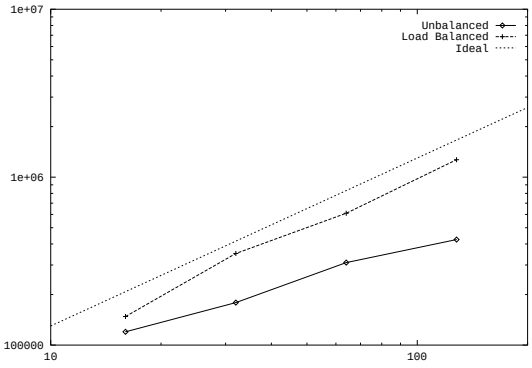

Figure 4: Performance on the scaled GEC problem

that was 77% of ideal.

A final test was conducted in order to evaluate the effectiveness of the automatic granularity control technique. The scaled-particle simulation was executed on 128 processors with 3.2 million particles, both with automatic granularity control, and without, using different numbers of partitions per processor. The GEC Grid was initially statically partitioned for one partition on each of 128 processors, and tests were conducted using the same initial partitioning, both with and without automatic granularity control. For the simulation with automatic granularity control, partitions were automatically divided only when deemed appropriate by the load balancing technique. For the simulations without automatic granularity control, each initial partition was repeatedly split in order to obtain a specified number of partitions per processor (1,2,4,8, or 16), and then the load balancing method continued without any further splits. This approach yields the most uniform granularity possible for the given initial partitioning. In each of these cases, the performance, in particles per second, was measured both before and after dynamic load balancing.

The decrease in performance of the unbalanced case reflects the increased overhead of the additional partitions on the same processor, which is fairly small, as inter-processor communication is not increased. With only one partition per processor, load balancing cannot make any improvement. Up to 8 partitions per processor, performance improves with more partitions per processor, as load balancing has more flexibility in transfer selection. Above 8 partitions per computer, however, the increased overhead of non-local communication is greater than any improvements from load

balancing, resulting in a lower performance. Without automatic granularity control, the best performance is obtained with the use of 8 partitions per processor.

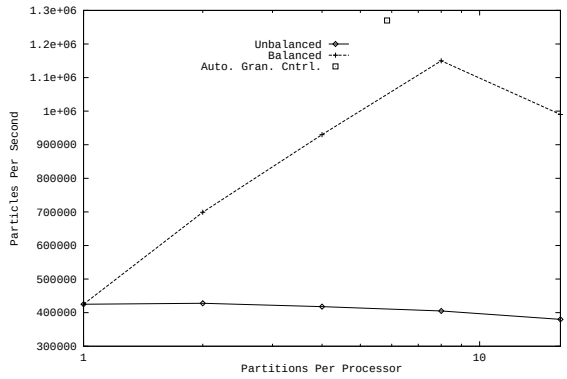

Figure 5: Performance as a function of partitions per processors

The automatic granularity control technique yielded an average of 5.84 partitions per processor, and a 10% better performance, with 27% fewer partitions. The performance improvement is the result of two factors. First, fewer partitions are required and thus the volume of communication is reduced; second, the approach guarantees that no partition is so large as to impede the load balancing method.

In addition to the performance improvement that results from the use of automatic granularity control, it is important to note the reduction of parameters. Without automatic granularity control, it is necessary to specify the desired number of partitions per processor. An optimal value for this parameter can only be determined by extensive tests. A sub-optimal number of partitions per processor could further reduce performance by 12%. In general, dynamic load balancing and automatic granularity control will yield better performance than static manual partitioning.

CONCLUSION

These results demonstrate the effectiveness of automatic granularity control for the purpose of load balancing of particle-based simulations on distributed-memory multicomputers. Due to the irregular nature of particle-based simulations, static techniques are inadequate; dynamic and adaptive techniques must therefore be applied. These techniques have been presented in the context of rarefied gas flow in the GEC Reference Cell reactor. The same tools can also be applied to spacecraft reentry calculations, and are in use at Intel Corporation for the simulation of proprietary reactor systems. While the present work focuses on homogeneous, distributed-memory machines, the same automatic granularity control techniques can be applied to shared-memory machines and even heterogeneous networks of workstations. In fact, granularity control is critical for obtaining efficient use of any concurrent architecture.

ACKNOWLEDGEMENTS

Infrastructure support and computing resources for this research were provided by BMDO under contract DAAH04-96-1-0319, and by Avalon Computer Systems, Inc. The research described in this report is sponsored by Intel Corporation and the Advanced Research Projects Agency under contract number DABT63-95-C-0116. This project includes Russian participation that is supported by the U.S. Civilian Research and Development Foundation under Award No. RE1241. The information contained herein does not necessarily reflect the position or policy of the government of the United States, and no official endorsement should be inferred.

REFERENCES

Bird, G. 1994. *Molecular Gas Dynamics and the Direct Simulation of Gas Flows.* Clarendon Press, Oxford, England.

Gimelshein, S.; G. Markelov; and M. Rieffel. 1996. "Collision Models in the Hawk DSMC Implementation." Technical Report CS-96-16. Department of Computer Science, California Institute of Technology, Pasadena, CA.

Rieffel, M.; S. Taylor; J. Watts; and S. Shankar. 1997. "Concurrent Simulation of Plasma Reactors." *Proceedings of High Performance Computing 1997*, 163-168.

Watts, J.; M. Rieffel; and S. Taylor. 1996. "Practical Dynamic Load Balancing for Irregular Problems." In *Parallel Algorithms for Irregularly Structured Problems: IRREGULAR '96 Proceedings*,1117. Springer-Verlag LNCS.

Watts, J.; M. Rieffel; and S. Taylor. 1997. "A Load Balancing Technique for Multiphase Computations." *Proceedings of High Performance Computing '97.* pp. 15-20.

Rieffel, M; S. Taylor; and S. Shankar. 1998. "Reactor Simulations for Semiconductor Manufacturing." *Proceedings of High Performance Computing '98.*