Smooth particle hydrodynamics on ATM-connected distributed PC clusters

Pete Wyckoff* and Rob Armstrong
Sandia National Laboratories
MS 9011, P.O. Box 969, Livermore, CA, 94551, USA
{wyckoff,rob}@ca.sandia.gov
Phone 925 294 3503 FAX 925 294 1225

Abstract *The simulation of a fluid flow using smooth particle hydrodynamics is performed on clusters of inexpensive personal computers. These clusters achieve connectivity in the wide area using ATM across links of a variety of capacities. Utilization of a full OC12 (622 Mb/s) of bandwidth is obtained using commodity OC3 network interface cards by striping the application data across multiple nodes in parallel.*

Keywords ATM, cluster, SPH.

1 Introduction

In the past supercomputers were expensive to build, maintain, and operate. This level of specialization naturally made it quite tough to do real parallel programming, and an elite club was formed. Now the existence of hardware standards allows for a wide selection of components in building a machine, and expensive parts need only be applied where they will have the most benefit. The effect of this development is that massively parallel computers can be assembled out of commodity parts to perform the largest simulations. Further, the standards-based commodity computers are well suited for rapid deployment of new technologies. When faster processors arrive on the market, for instance, they can quickly be integrated into the existing machine without redesigning the entire set of original, say, PCI components.

Another fading remnant of history is the idea that all of the supercomputer must be in the same room. With the ever-increasing speeds of wide-area networks, it is becoming feasible to locate pieces of the same conceptual machine at multiple sites around the world. Commodity hardware is also available which allows compute nodes of the machines to utilize the network backbone directly without additional hardware.

The following sections explain the use of two distributed clusters of personal computers with ATM providing the remote interconnectivity in the development of a smooth particle hydrodynamics code for the simulation of a Navier-Stokes flow. Details of a demonstration at SC97 serve as an example.

2 Simulation environment: Cluster computing

The future of high-performance computing lies in massively parallel computers, which are unfortunately becoming too expensive on a per-flop ratio when compared to their smaller commodity brethren. These cheap, widely available personal computers (PCs) can be purchased whole or piecewise from a variety of vendors and, to a large extent, conform to the same basic set of hardware standards such as having a PCI bus. Networking components required to assemble many PCs into a single system are likewise cheap and plentiful, although the cost of the interconnect fabric can be a significant part of the overall purchase price of the machine.

Sandia embarked on a program in 1997 to build its next-generation supercomputer out of commodity components. The idea behind the Computational Plant (CPlant) is that installations of new hardware will be attached to the base tree every so often, and older growths will be pruned. The development cycle of computer-related technologies has been shown to be about 18 months, making it important not to rely solely on one manufacturer or system for long-range computing. Instead, by sticking with some well-defined set of standards for hardware, recent evolutions in hardware design can

*Presenting author

be added to the current computing platform, which still remains viable.

The necessary pieces to make this plan work are the software infrastructure. The concept is unattractive if application programmers must learn a new operating system, or even a new set of compiler flags, to use the newest growth of the machine. To meet this goal a freely available, widely ported operating system and set of tools is necessary. We chose Linux as an operating system which will likely be around for a few years, and run on the current computer of choice. The parallel programming interface seen by the application is MPI, which suits most codes that run on the machine. PVM is also available, as are other more experimental message passing systems.

A major stumbling block for cluster builders has been the failure to ensure that the system would scale. This is distinct from the usual requirement that applications must scale well to be efficient, and speaks to the reliability and usability of the system itself. CPlant is designed as an arbitrarily connected aggregation of individual quantized pieces called scalable units. These SUs are fully functional stand-alone clusters and can be administered and used in isolation from the rest of the system. Each SU consists of some number of compute nodes (16 for the present machine), a service node, and an input/output node, and is shown in Figure 1. The compute nodes run minimally configured kernels and message passing traffic is done exclusively on the high-speed interconnect. They may have local disks, but do not *depend* on them; instead, the nodes boot across the network and use NFS or local disk caching as needed. The service node is the "manager" for the SU, performing tasks such as booting the other nodes, monitoring the nodes for failure, and providing connectivity to the local LAN. The I/O node provides disk service for its own SU and can be used to implement a parallel filesystem across the entire machine. The SUs are combined into a single machine using utilities at a higher level which dynamically map tasks onto available physical machines [1].

As an example, the first piece of CPlant is based on nodes which are DEC Miatas, each of which contains a 433 MHz EV5 Alpha processor, 196 MB of SDRAM, a 2 GB IDE drive, on-board 100 Mb/s ethernet, and a Myrinet card. The Myrinet switches are wired to form a cube of eight switches, attached to each vertex of which are two compute nodes, and form the basis of the local-area high-speed interconnect. This single unit cube is extended in the x and y directions by connect-

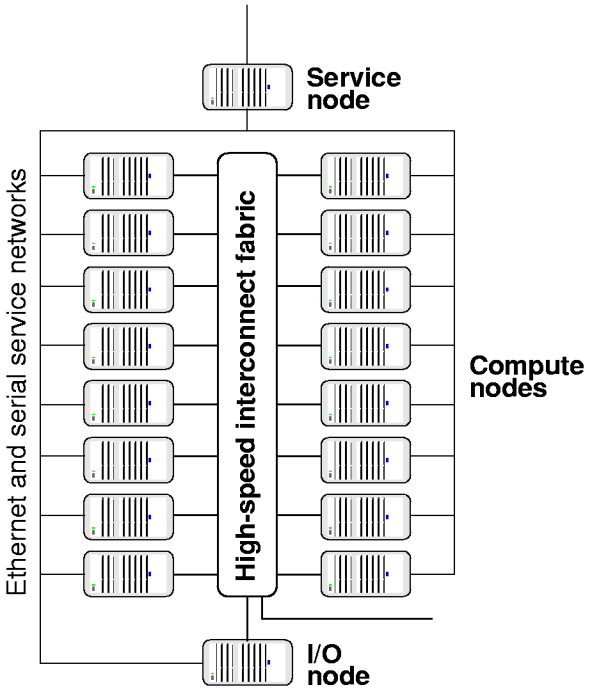

Figure 1: CPlant scalable unit consisting of 16 compute nodes and one input/output node, controlled by a service node. Three wiring networks connect the SU to itself, and to the outside world.

ing adjacent scalable units. Currently there are ten SUs: two Intel-based and two Alpha-based in Livermore, and six Alpha-based SUs in Albuquerque [2].

3 Wide-area networking

To be able to perform simulations of the largest scales, the computational resources of multiple physical sites must be used in concert. However, today the connectivity across the wide-area interconnect is quite poor compared to that inside the machine room, by many orders of magnitude. Assuming that one day the long-distance phone companies would be willing to sell 100 Gb/s of guaranteed bandwidth between two sites, there will certainly be no single network adapter card which can deliver that performance. Instead, multiple smaller connections into the computational fabric will be aggregated in stages up to the eventual 100 Gb/s bandwidth of the wide-area. The scenario described here is a demonstration of the feasibility of such a scaled network design, using only commodity off-the-shelf components.

The two components which make up CPlant are located at two sites 1100 miles away from each other. The sites are connected by an OC3 link (155 Mb/s) on which can be arranged dedicated time for CPlant application programming. We also have the capability of using an OC12 link (622 Mb/s) to connect to other sites around the San Francisco Bay area using the LLNL-operated National Transparent Optical Network [3].

At the SC97 conference in San Jose, CA, the full OC12 bandwidth offered by NTON was used in a live demo. One rack of the four from Livermore was moved to the show floor in San Jose, and it was connected back to Livermore through NTON. To sustain the bandwidth, four Efficient Networks OC3 ATM cards were used concurrently, striped together into the OC12 pipe. The network configuration is shown in Figure 2.

The software used to drive the ATM cards comes from the Linux community [4], but had to be heavily modified to work on the 64-bit architecture of the Alpha processor in the Miatas of CPlant [5]. Permanent virtual circuits were established in all the switches to enable ATM connectivity, and one switch was elected as an ATMARP server for classical IP. Standard IP routing protocols were used to arrange for traffic to flow across the ATM links, and application-level partitioning of the problem guaranteed that the links would be relatively balanced during the course of a run.

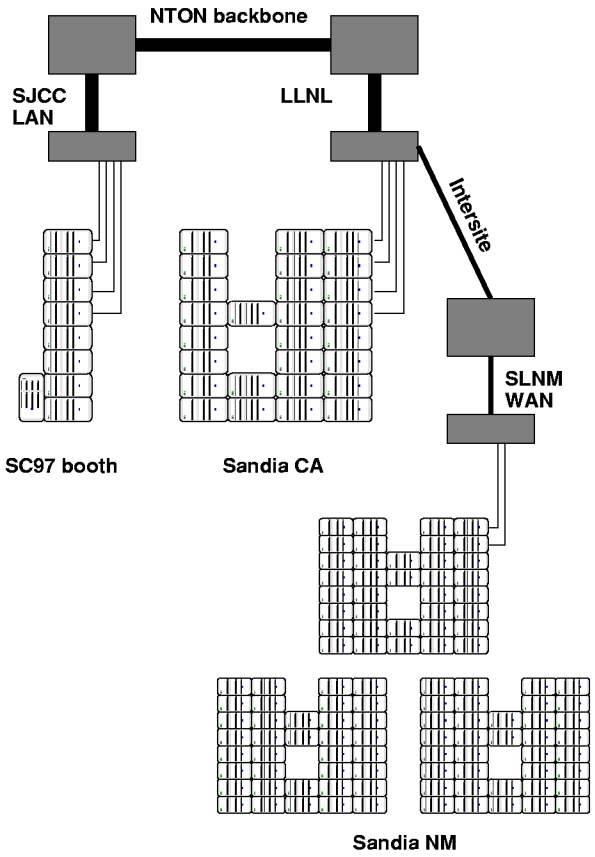

Figure 2: Connectivity diagram showing the three computational sites and ATM links between them.

In particular, of the eight nodes at the SC97 booth, only nodes 1 through 4 had ATM cards, and were configured to route IP across the ATM links back to the Livermore and Albuquerque clusters, each of which had its own IP class C subnet. The nodes without ATM cards used as gateways those with did possess them. For example, node 5 used node 1 as its gateway, and node 6 used node 2. Similar arrangements at the other two sites ensured that ATM wide-area link usage would be balanced as long as the application maintained a balanced communication load.

The SPH application ran continuously for hours using up to 64 processors across all three sites. Since at the time of the demonstration the link between Livermore and Albuquerque was only a T3 (OC1), this was the main impediment to utilization of the entire set of 128 processors, as a great imbalance existed between the large bandwidth connection from San Jose to Livermore against the narrow connection from there to Albuquerque.

4 Smooth particle hydrodynamics

Smooth particle hydrodynamics (SPH) is a method which has its origins in astrophysics [6] and is useful in boundary-free fluid mechanics involving high deformations or non-linear constitutive equations. It is derived from purely continuum equations from which a particle representation is generated. The particle approximation has not been rigorously shown to converge to the continuum equations, but in practice the agreement can be very good depending on the flow conditions. The "particles" in the name SPH have no physical counterparts. They are similar to collocation points in finite element calculations, in that each particle represents the values in some domain around it. These contributions to the continuum field are summed using weights based on the masses of the particles.

The calculation of interactions between the particles does not require a mesh, so large deformations of the physical substance can be modeled in a computationally efficient way. This is the main reason SPH has found prominence in the fields of astrophysics (star formation) [7] and fluid mechanics (shock and high-strain flows) [8].

The particular problem modeled here is that of a viscous incompressible Navier-Stokes fluid flow in three dimensions. The boundary conditions are parallel plates on the top and bottom of the domain, and circular (or periodic) in the x and y directions. Particles are seeded in the domain randomly and allowed to evolve in time while statistics are gathered to show the density distribution, velocity profile, and other quantitative measures of the flow.

The fluid is taken to satisfy Stokes' relation, which expresses the stress tensor in terms of velocity gradients. The equation being solved, then, is

$$\frac{d\mathbf{v}}{dt} = -\frac{1}{\rho}\nabla p + \frac{\mu}{\rho}\left(\nabla^2\mathbf{v} + \frac{1}{3}\nabla(\nabla\cdot\mathbf{v})\right).$$

Averages are taken over each term of the equation using the definition

$$\langle f(\mathbf{x})\rangle = \int W(\mathbf{x}-\mathbf{x}')f(\mathbf{x}')\,d\mathbf{x}'$$

where W is a normalized weighting function, that has compact support over a fairly small region, compared to the global domain. Integral quantities in the resulting expression are then expanded as sums over particles in the local neighborhood of a point.

The particular weighting function used in the code for this series of simulations is the simple truncated Gaussian. Its form is

$$W(x) = \frac{1}{2\pi}\left(\frac{\exp(-d_{ij}/h_i^2)}{h_i^2} + \frac{\exp(-d_{ij}/h_j^2)}{h_j^2}\right)$$

where d_{ij} is the Euclidean distance between the two particles i and j, and h_i is the smoothing length for particle i, a measure of the "size" or "influence" of the particle on its surroundings. This value is derived from the mass of the particle. The Gaussian interaction term is truncated by not calculating the interaction between two particles which are more than a certain distance apart, as specified by an error tolerance on omitting that term (10^{-5} here).

5 Parallelization strategy

The code is parallelized in the traditional domain-decomposition style, where each processor is assigned the calculation of velocity gradients for the particles which happen to be in its domain. The gradients arise from interactions between particles which may be in other domains, requiring message passing at every step of the calculation. Only the information on particles near the boundary of each cell must be transferred to the neighboring cells, as illustrated in Figure 3. (The terms "cell" and "domain" are synonymous here, although one could

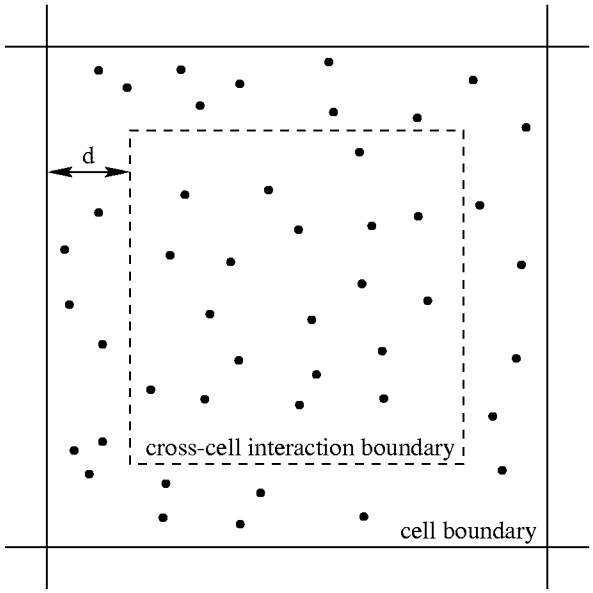

Figure 3: Particles inside the cross-cell interaction boundary described by the distance d do not affect particles outside that cell, and do not need to be transferred to another processor.

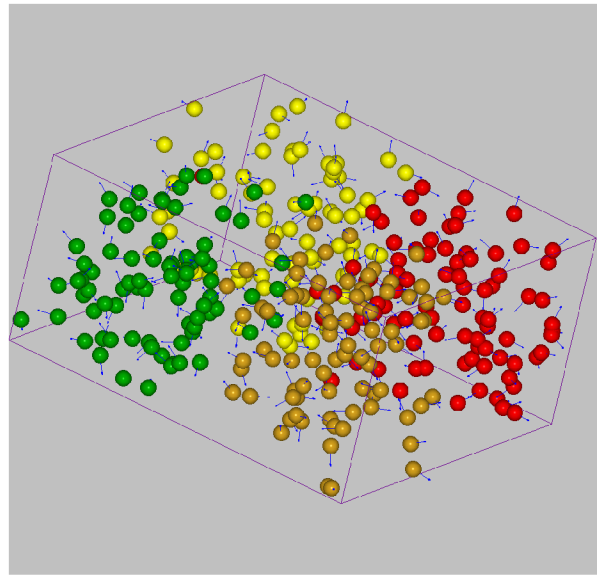

Figure 4: Smooth particle hydrodynamics simulation snapshot. Particles are the shaded to indicate the processor responsible for their calculation.

imagine a case where an aggregation of multiple cells on a single processor would be useful for load balancing or the calculation of long-range forces.)

The domain sizes are constrained at a minimum to be larger than the smoothing length used for the simulation. That length is derived from the current density of particles and the shape of the smoothing function. This minimum domain size guarantees that only nearest-neighbor processors must communicate. The calculation of this minimum domain size is rather complex, as it must consider interactions through the Gaussian weighting, as well as through its first and second derivative. The shapes of W, $\frac{dW}{dx}$, and $\frac{d^2W}{dx^2}$ are searched to find the largest separation x which satisfies the error tolerance mentioned above. The resulting value is shown as d in Figure 3.

As the computation progresses, particles which were inside the domain of one processor may migrate into the domain of another. The current "owning" processor of the particle continues to integrate its changes even though it may have moved out of its domain by enlarging the overlap it requires of the neighboring processor's domain. Eventually the owning processor would be forced to acquire information about particles more than one domain away, but instead, a repartition is triggered based on a tracking of computational efficiency. Extending the domain overlaps requires more message passing and should be minimized, but repartitioning the particles requires the communication of extra per-particle information that is not normally sent at each time step of the simulation, resulting in a tradeoff between these two effects.

Load balancing is performed based on each processor's elapsed total time required to calculate a few iterations. This is necessary because processors which abut the top and bottom solid boundaries require fewer neighbors to compute the motion of their particles, because the effects of the boundary are accounted for analytically. Only a few load balancing steps are required to smooth out these differences which arise from the initial equal-area distribution. The physical domains of the under-utilized processors are increased until the parallel computation is balanced. In the event that this machine is used in a time-sharing (as opposed to space-sharing) mode, load balancing can account for the effects of other users as well.

6 Simulation results

An example snapshot of the simulation is shown in Figure 4 where each particle has attached to it an arrow representing its velocity. The particles are color-coded to indicate which processor is responsible for computing the particle's dynamics. For the

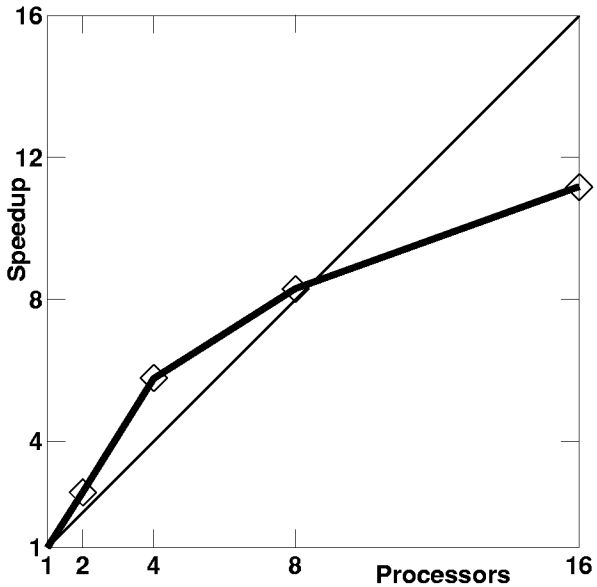

Figure 5: Speedup values for various numbers of processors, for a relatively small fixed-size problem.

sake of clarity, only one out of every twenty particles participating in the calculation is shown, and only four processors are used in the simulation. The Reynolds number of the flow is 10, and the Mach number is 0.5, as calculated by taking averages over all the particles.

It is interesting to see that the code shows superlinear speedups for small problem sizes of interest. Figure 5 shows this effect. The curve quickly levels off and the efficiency decreases as the number of participating processors gets large. This effect is only visible since the problem size is kept fixed while the number of processors is varied, and the corresponding amount of work per processor decreases, possibly leading to increased cache performance.

"Interesting" simulation sizes are much larger than those used here for illustration. Memory requirements are the limiting factor determining the number of particles which can be maintained by a single processor. To handle the state requirements of this simple Navier-Stokes flow, 80 bytes of memory are required for each particle. Further storage is needed to hold copies of some of each processor's neighboring cells, depending on the average density and the physical size of the domain, say one "ghost" cell for each cell to be calculated as a conservative estimate. On a machine with 256 MB of main memory per processor this gives an upper limit of 1.5 million particles, compared with the runs using 60 000 particles used for descriptive purposes here. More advanced uses of the SPH algorithm would certainly include auxiliary state information such as chemical composition, and would further restrict the usable problem sizes.

7 Future work

It is not surprising that SPH, for all its value in application to large and odd-shaped deformations, has difficulties when boundary conditions must be applied. The most common approach to this is to place layers of particles having the shape of, and moving with the velocity of the boundary, with the thickness of the layer usually chosen to be one or several of the interaction distances. Still difficulties arise. There are no "hard" surface approximations natural to the method, so particles still squeeze through a boundary that is meant to be impenetrable. Various heuristics have been developed to address these problems such as specially crafted artificial viscosity, but these are, well, artificial. The particular simulation described above was special in that there are analytical expressions for the planar impenetrable boundary conditions using a Gaussian weighting function.

As alluded to earlier, the SPH method differs only slightly from a classical molecular dynamics (MD) application using a finite cut-off radius. SPH derives most of its advantage from its ability to do large deformation mechanical simulations. However due to the finite particle interaction length, SPH suffers from particles or clumps of particles "breaking free" from the main body. Although, this can be physical, the model does not support free boundary conditions and there is no "free particle" analogy in SPH *per se*. Future work will center on flexible boundary conditions that are derived from molecular analogies and converge to the phenomenological equations for a free surface.

In addition to the mathematical similarity between SPH and MD, there is an almost a one-to-one algorithmic similarity. Future work will focus on creating a toolkit for particle-based applications, including both SPH and MD. Other application domains that can benefit from this toolkit are Probability Density Function, Particle In Cell, and Cohort methods. For example, algorithms and communication patterns outlined here for SPH load-balancing and particle motion are identical in MD calculations. Currently these portions of the code described here are being converted into SPMD (single-program multiple-data) parallel

components for use in a parallel framework [9].

The hope is that components that are machine dependent or difficult to create can be shared among applications, reducing the often considerable time needed to develop a working simulation. It is characteristic of SPH, and all particle methods, that problem setup, checkpointing, load-balancing and input/output are equally as burdensome as the actual computation of the particle motions. Naturally, the equations of motion for each application will have to be rewritten on a case by case basis, but if components that do these necessary but time-consuming tasks can be reused, engineers and scientists will be more productive on advanced parallel computers [10].

8 Summary

The application of smooth particle hydrodynamics in the calculation of a Navier-Stokes fluid flow was presented, using a parallel, geographically distributed cluster of commodity personal computers. The ability to stripe across multiple ATM links allows the use of a large amount of network bandwidth without the need for specialized connectivity components. CPlant shows great promise as being the scalable computing platform of the future.

This demonstration shows both the viability of commodity components for supercomputing applications and the feasibility of WAN connectivity to unite distributed parallel resources of various sorts of locations and bandwidths. Combined, these distributed computing resources can be used to solve individual problems of immense proportions.

References

[1] D. S. Greenberg, R. Brightwell, L. A. Fisk, A. B. Maccabe, and R. Riesen. A system software architecture for high-end computing. In *Proceedings of SC97*. ACM/IEEE, 1997.

[2] J. Laroco. CPlant web pages. `http://rocs-pc.ca.sandia.gov/CPlant/CPlant.html`.

[3] P. D. Sargis, B. D. Henderer, and M. E. Lowry. 10 Gb/s subcarrier multiplexed transmission over 490 km of ordinary single-mode fiber without dispersion compensation. Technical report, LLNL, August 1997. UCRL-JC-127376.

[4] W. Almesberger. ATM on Linux. `ftp://lrcftp.epfl.ch/pub/linux/atm/papers/atm_on_linux.ps.gz`, July 1996.

[5] T. C. Hu and P. Wyckoff. Connecting remote clusters with ATM. Sandia National Laboratories SAND report, in preparation, February 1998.

[6] J. J. Monaghan. Smoothed particle hydrodynamics. *Annual Review of Astronomical Astrophysics*, 30:543–74, 1992.

[7] L. B. Lucy. A numerical approach to the testing of the fission hypothesis. *The Astronomical Journal*, 82(12):1013–24, 1977.

[8] H. Takeda, S. M. Myiama, and M. Sekiya. Numerical simulation of viscous flow by smoothed particle hydrodynamics. *Progress of Theoretical Physics*, 92(5):939–60, November 1994.

[9] R. Armstrong. Frameworks for parallel computing. `http://gs.ca.sandia.gov/~rob/poet`, May 1998.

[10] S. J. Vinay, M. C. Kim, P. Wyckoff, R. Armstrong, and M. S. Jhon. A novel computational method in fluid mechanics. Subtmitted to *AIChE Journal*, March 1998.