Beowulf Cluster Design for Scientific PDE Models

B. McGarvey, R. Cicconetti, N. Bushyager, E. Dalton, M. Tentzeris
School of Electrical and Computer Engineering, Georgia Institute of Technology
http://www.athena-em.gatech.edu/Beowulf/index.html

Abstract

This paper details some of the basic issues that arise when designing a Beowulf cluster for particular types of scientific simulations. The general problems of interest are partial differential equations (PDEs). These equations describe natural phenomenon and can be numerically solved. Finite differencing is used to solve the system of equations. This method naturally delineates the problem into four distinct categories based on the span of the data stencil used to update the next point: Independent, Nearest Neighbor, Quasi-Global, and Global. This delineation also relates the communication requirements between the independent processes. Other important factors include computation and memory requirements of the algorithm and particular problem. Design tradeoffs are presented with regards to algorithm development, communication architecture, and node configuration.

1.1 Introduction

Prior to the advent of Beowulf clusters [1], researchers simulating complex systems were limited to time-sharing on traditional "big iron" centralized in supercomputing centers. The limited availability of resources greatly restricted researchers' ability to experiment and limited the number of projects considered. Clusters changed the paradigm by offering distributed computational horsepower to more researchers than was previously possible. Beowulf clusters have expanded the field of numerical modelers by yielding a single user supercomputer at 1/100th-1/10th the cost of a traditional "big iron" with comparable performance.

This paper details some of the basic issues that arise when designing a Beowulf cluster for particular types of scientific simulations. A Beowulf cluster is usually a group of computers networked together on a private lan to perform distributed tasks. The general problems of interest are systems defined by partial differential equations (PDE). These types of problems are commonly found describing physical systems such as electrodynamics or fluid flow. Due to the complex nature of the problems under scrutiny, numerical approximation is often the only method of finding an acceptable solution. The methods for finding solutions to these systems are quite varied. Focusing on a particular method that is relatively simple to understand: finite-difference time-domain solutions.

1.2 Background

The computational electromagnetics lab (Athena-EM) at the Georgia Institute of Technology needed more computational capacity to adequately perform research. Our basic choices were renting time at traditional supercomputing centers, purchasing proprietary specialized workstations, or building a Beowulf cluster. We chose the Beowulf cluster as it provided the most flexibility and control at an affordable price. This paper came out of our initial experiences from building our Beowulf cluster. During the investigation and planning stages, we found that there were no easy answers to the question of optimization, and if a cluster would help with our needs.

The main difficulty in planning our cluster was our lack of experience and established design guidelines. Asking the experts often yielded the standard response, "it depends on your problem." Another difficulty we encountered was the informational gaps between the cluster architecture designers and the numerical modelers. Each group understands the difficulties associated with their specialty, but no common language seemed to exist. By bringing the disparate groups together new and novel designs for clusters may evolve for solving this class of problem.

2.1 Partial Differential Equation Modeling

In an effort to describe the physical world, scientists and physicist use differential equations to approximate it. The mathematical study of natural phenomena is

performed using differential equations. It generally describes the relative rates of changes of variables in a system. PDEs are continuous functions with infinite resolution, but a computer has finite resolution. Therefore, the PDE must be converted to a discrete approximation of the continuous function, before a solution can be found. For example, an electromagnetic simulator [2] was used as a benchmark for this paper [3]. In the electromagnetic simulator, one of the 2D governing Maxwell's equations has the continuous form (1), but becomes (2) after discretization.

$$\frac{\partial E_z}{\partial t} = \frac{1}{\varepsilon}\frac{\partial H_y}{\partial x} \quad (1)$$

$$E_i^{z,n+1} = E_i^{z,n} + \frac{\Delta t}{\varepsilon \Delta x}\left(H_{i+1/2}^{y,n+1/2} - H_{i-1/2}^{n+11/2}\right) \quad (2)$$

The notation used is to express the location in time and space of a particular variables' value. In this case *n* represents the time step and *i* represents the spatial location of the variable. The sub- and super- scripting are shortcuts used to help develop the models by compacting the diverse information.

2.2 Basic Algorithms

The numerical details of the finite-difference time-domain algorithm are complicated, but a simplified version is very easy to understand and covers the majority of the reason one needs a cluster. A simple case provides that there are N points on a grid and two variables associated with these points, each inter-dependent. The algorithm fixes one variable while changing the other, and then marches forward in time a small amount to update the other variable, in a leapfrog fashion. So the number of data points always remains constant, N. Basis functions affect the update algorithm, by determine the number of elements in the previous time-step's set that are required to update a single point in the next time step. As the basis function expands, the communication requirements grow.

2.3 Basis Functions

The basis functions and method used to transform the continuous space into a discrete one greatly affects the stencil of the update algorithm. The stencil is how these sets of problems are being delineated. The name of the problem type relates how many adjacent elements of the set are required for updating the next data point.

Independent

This type of problem is included for completeness. It is not normally used in PDEs, but it is of interest to the Beowulf community at large. It commonly referred to as "embarrassingly parallel," because the update algorithm requires the data only from the previous time step or initial conditions and the general algorithm, i.e. co-located models (Fig 1).

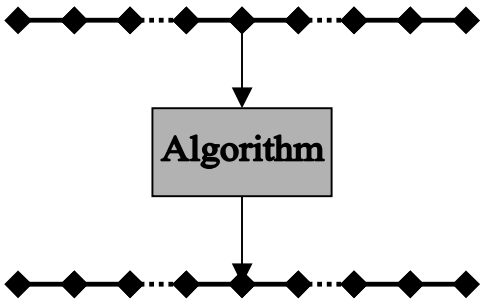

Fig. 1: Independent point update methodology

Nearest Neighbor

One method of approximating derivatives is to use central differences. First derivatives calculated using central differences only use values from neighboring points. This method can be used to solve several classes of equations, such as Maxwell's electromagnetic equations using the Yee-FDTD scheme [4].

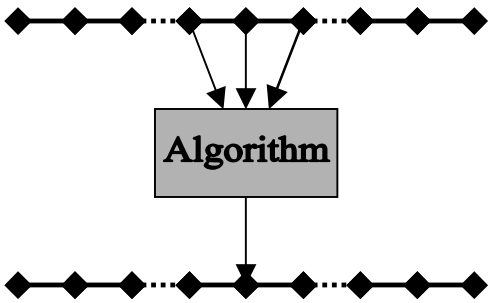

Fig. 2: Nearest Neighbor point update methodology

Quasi-Global

Other higher-order FDTD schemes and methods for discretization result in the need for more than just the adjacent points, the scheme determines the number of points required from the original point. One such scheme is the Battle-LeMarie Multi-Resolution Time-Domain (MRTD) method [5].

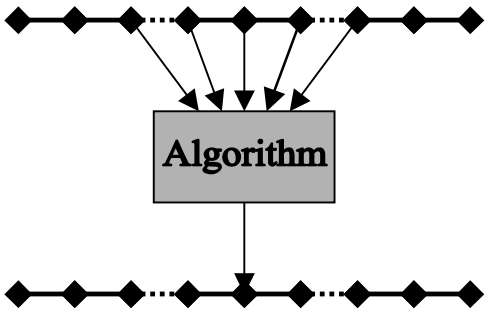

Fig. 3: Quasi-Global point update methodology

Global

The moment method with a global basis function requires that all the other data points in the space be known to update any single point, This is generally the limiting case of data interaction; updating any point depends on the data from every point on the grid [5]. Matrix inversion loosely fits into this category.

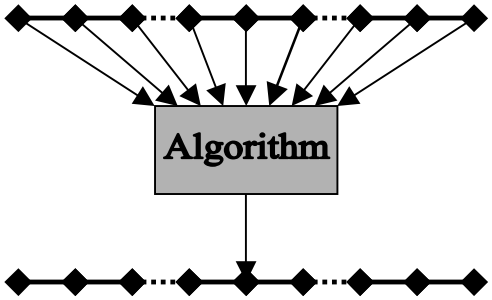

Fig. 4: Global point update methodology

3.1 Basic Performance Characteristic

Amdahl's Law

Parallel computing performance is often measured with reference to speedup (3).

$$Speedup = \frac{time(1\ processor)}{time(N\ processors)} \qquad (3)$$

The ratio of the time it takes a single process to complete a task to the time it takes N processes to complete the task. For a perfect program one would expect a speedup of N linear with the number of increasing processors. Amdahl's Law (3) states, the best possible performance increase in time, given that a particular program is perfectly parallelizable, is 1/N, where N is the number of processors [6]. But most real world applications do not fare that well. An alternate method for calculating Speedup was presented by Amdahl in (4), where s is the serial fraction of the program and p is the processors. In every program that one is attempting to parallelize there are sections that can be parallelized and others that must remain serial for various reasons. The fraction of the program that is serial determines the maximum speedup and is limited by $1/s$ even as p goes to infinity [7]. As is obvious, keeping the serial portion small is highly desirable in order to keep speedup near linear.

$$Speedup = \frac{1}{s + \frac{1-s}{p}} \qquad (4)$$

3.2 Real World Issues

Amdahl's Law relates the theoretical limit of possible speedup. Traditional supercomputers are built with ultra-high speed interconnection networks with very low latency in order to keep the serial fraction small. Beowulf clusters suffer from relatively slow inter-processor communication in comparison. To achieve significant improvement the communication time needs to be minimized while the computation time is maximized.

Computation Time

The differential equations governing the system and the method used for discretization predetermine the relative complexity of the update algorithm. This in turn determines the average time required to calculate the next value of a single data point. Not only does the complexity of the update sequence have an impact, but the type of operations required can also have a significant impact upon performance. For example, a floating-point divide is a non-pipelined function of many modern processors, thereby greatly reducing performance.

Communication Time

The effect of the communication time requirements becomes obvious as the data dependency increases along with the domain of the basis function and more data is required to be passed from one unit to another.

3.3 Granularity

Early in the development of Beowulf clusters people began to talk about granularity of problems. Granularity can be related to the computation/communication ratio.

By keeping this number small, the theoretical maximum speedup can nearly be reached. Granularity issues can impact any type of problem if a good ratio is not achieved. For the theoretical limit to be approached, the inequality must be satisfied. Otherwise, the ratio will determine the maximum speedup possible for that configuration of the problem.

$$\frac{Computation\ Time}{Communication\ Time + Serial} \geq Number\ of\ processors$$

Several of the experiments performed will illustrate this inequality.

4.0 Experiments

All experiments were performed on our homogeneous cluster as described using the fast Ethernet configuration except where noted. The experiments were designed to test the four basic types of problems.

4.1 Independent

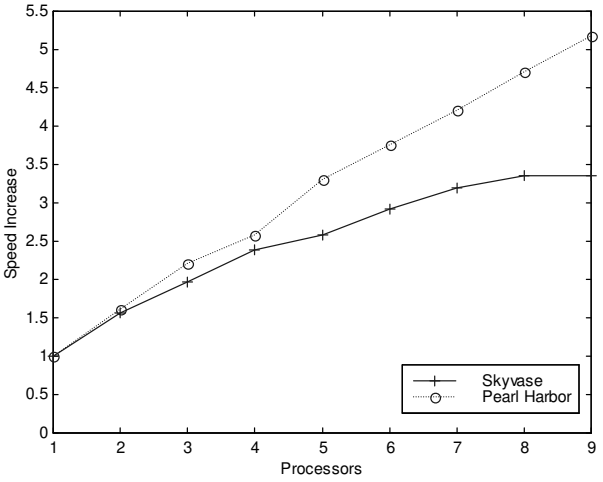

Fig. 5: PVMpov Results

PVMpov [8] satisfies the requirement of only requiring one data point and an algorithm to evaluate the next data point. Two separate well-known scenes were rendered, *skyvase.pov* [9] and *pearlharbor.pov* [10].

Skyvase was selected because it is the benchmarking scene PVMpov has used for many years. *Pearlharbor* was selected in an attempt to increase the granularity. Fig 5 shows the results.

The PVMpov program divides the computational space into small, equal-size blocks. The two scenes vary in complexity. The *Pearlharbor.pov* is significantly more complex than *skyvase.pov*, but as the number of nodes increases the communication overhead becomes the limiting factor. The small dataset is a trade off. According to the makers, PVMpov was designed for non-homogenous clusters [8].

4.2 Nearest Neighbor

For this experiment, an in-house parallel FDTD electromagnetic simulator was used [3]. We studied a small data set and a large dataset. The grid size was limited to one that could remain resident in RAM of a single node for comparison purposes. Blocking calls were used in swapping data between the nodes for illustration. The results are presented in Fig 6.

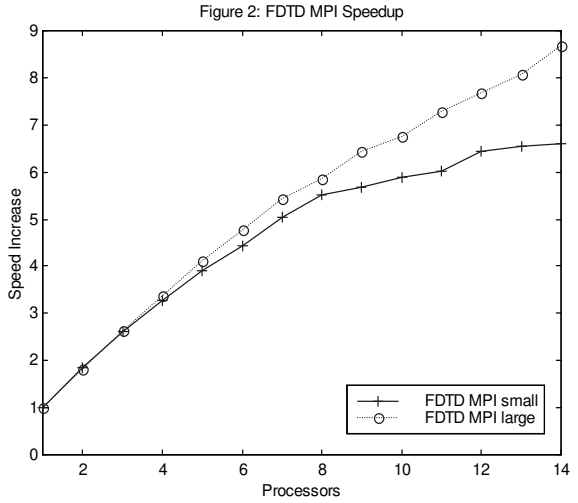

Fig. 6: Hydra-FDTD Results

For the small problem, the speedup tapers off as the subdivided blocks become too small, but the larger problem still maintains acceptable granularity and the slope of the speedup curve only decreases slightly. Similar results are shown as for the independent problem, but as the granularity is better, the theoretical limit is more closely approached.

4.3 Quasi-Global

The results for GADGET were inconclusive. An alternative program is required for conclusive results [11].

4.4 Global

Two separate methods were used to evaluate our cluster's performance for global problems. The first is a

basic gravity simulator that was developed for evaluating performance and implementing a MPI program [12]. The other method used to evaluate the performance of the cluster was LINPACK [13]. For many scientific problems, inverting large matrices is the preferred method for finding the solution.

GravSim Results

The each line in Fig. 7 represents the problem size for the n-body gravity simulator, $O(N^2)$. The problem is symmetrically divided. It shows dramatically the results of poor granularity. For 128 points when the number of processors reaches eight the performance significantly degrades and does not recover, but the larger problems scale close to the theoretical limit and nearly linear. The numerical artifact for the 128-body, 16-processor case was caused by a one of the nodes having a faster processor.

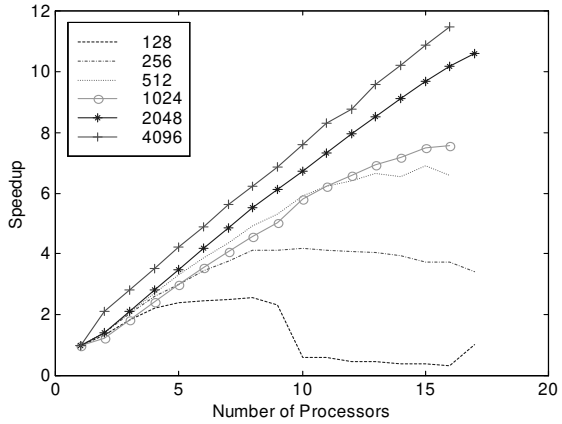

Fig. 7: GravSim Results

It is important to understand that for the large problems that demonstrate linear speedup with this small cluster would behave in the same manner as the small problems.

LINPACK

As many global problems involve matrix inversion, LINPACK was chosen as a test case, since it solves a dense system of linear equations by inverting a matrix. It is another method for finding a solution for a global problem. In our testing, we found that 4 nodes, using a low-latency, high-bandwidth switching fabric, nearly outperformed 8 nodes on a standard fast Ethernet connection, showing that matrix inversion does benefit significantly from the high bandwidth low-latency switches that GigaNet CLan provides.

5.0 Results

To acquire valid test cases for comparison, tradeoffs had to be made. This required that the data set under test be limited to what could remain resident in RAM without having to swap to disk. As the number of processors increased the data set size remained constant yielding smaller data sets for the larger cluster and increasing the communication penalty. This excludes one of the main advantages of a cluster; the cluster can have N times the amount of available memory than just one individual node may have, enabling large complex devices to be designed and optimized.

The communication method was also limited to blocking calls. The experiment design called for this method to test the inherent latency and bandwidth of the system. For this class of problems where the data interaction is predetermined, an algorithm to eliminate the wait for data swapping is relatively straightforward. By taking advantage of non-blocking calls, the programmer can simultaneously swap data and update cells by working from the inside out.

6.0 Conclusions

Our Beowulf cluster has not only functioned as a test bed for cluster research, it has provided the computational needs of our lab and research into new and novel RF packaging devices.

Design Tradeoffs

Designing an effective cluster involves designing for a specific type of problem or making tradeoffs to satisfy the requirements of multiple problems while staying within budget constraints. Fast Ethernet can handle all the algorithms fairly well. A high-speed low latency interconnectivity fabric is only required for those systems that have large basis functions or very small data sets under investigation. These types of networks tend to be very expensive and have limited support.

Design Issues

If the users are processor bound, it may well be wise to choose symmetric multi-processing computers (SMP). Other choices that can predetermine the number of processors per computer may be the air-conditioning and space considerations. Individual nodes take up a lot of space and produce considerable heat. Our cluster is in the center of our workspace. This makes maintenance and security of the system trivial. Our lab

temperature suffers from the changing load of the system generating significant temperature swings.

Cluster Network Recommendations

For Independent, Nearest Neighbor, and Quasi-Global (with small basis functions) the optimal design appears to be a cluster with N nodes and a fast Ethernet network. For Global and Quasi-Global (with large spanning basis functions), a low-latency high bandwidth has significant advantages and provides speedup where Fast Ethernet was unable to do so.

Cluster Node Recommendations

The node selection is determined by whether the algorithm is memory- or processor-bound; if the program is processor-bound, SMP machines that share memory is an efficient method to increase the number of available processors for the same spatial footprint. SMP machines make excellent nodes when the problems under investigation are not memory bound. More nodes increase the total available memory, but increases information swap time and heat production.

Acknowledgement

The authors would like to acknowledge the support of DELL Computers, the Yamacraw Initiative of the State of Georgia, NSF career award, and NSF packaging research center.

Beowulf Cluster Specifications

8 –Nodes: Configuration
- Dell PowerEdge™ 1300
- Dual PII 500MHz (512k cache off die)
- 512MB SDRAM
- SCSI II 9.1GB drives
- IDE CDROM
- Floppy
- 1 – 3COM Tulip™ based fast Ethernet card
- 4 – Starfire™ base fast Ethernet cards
- (4 of the nodes) GigaNet CLan

Network Switch
- HP ProCurve 4000M 40 port

GigaNet Switch
- CLan 5000 8 port

References

[1] Beowulf: A Parallel Workstation for Scientific Computation, D. Becker, T. Sterling, D. Savarese, J. Dorband, U. Ranawak, C. Packer, Proceedings, International Conference on Parallel Processing, 95, *www.beowulf.org/papers/ICPP95/icpp95.html*

[2] A. Taflove and S. Hagness, *Computational Electrodynamics, the finite difference time domain approac*h, 2 nd ed., Boston, Artech House, 2000.

[3] Hydra-FDTD, in house electromagnetic simulator based on designs presented in [2] and [4],

[4] K. S. Yee, "Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media," IEEE Transactions on Antennas and Propagation, vol. AP-14, May, pp. 302-307, 1996.

[5] E.Tentzeris, R.Robertson, A.Cangellaris and L.P.B.Katehi, "Space and time adaptive gridding using MRTD",*Proc. of the 1997 IEEE Symposium on Microwave Theory and Techniqu*es, pp.337-340, Denver, CO.

[6] Maximizing Beowulf Performance, R Brown, Proceedings 4th Annual Linux Showcase & Conference, Atlanta GA, w*ww.usenix.org/publications/library /proceeding s/als20 00/full_papers/brownrobert/ brownrobert.pdf*

[7] Parallel Computer Architecture, D. Culler, J. Singh, A Gupta, Morgan Kaufmann Publishers Inc, San Francisco, CA, 1999.

[8] PVM patch for POV-Ray – PVMPOV, A. Dilger, H. Deischinger: PVMpov Version 3.0, Jakob Flierl: PVMpov patch to work with POV-Ray 3.1, *http://pvmpov.sourceforge.net/*

[9] POVBENCH, A.Haveland, www.haveland.com

[10] First Strike at Pearl, N.B. (beliaev@utu.fi), Glenn McCarter (gmccarter@hotmail.com), *www.geocities.com/SoHo/Workshop/9193/index.html*

[11] GADGET: A code for collision less and gas dynamical cosmological simulations, Springel V., Yoshida N., White S., 2001, New Astronomy, 6, 51,

[12]GravSim, in house tool written by R. Cicconetti

[13] www.top500.org/lists/linpack.html