Distributed Electric Field Approximation

D. Trybus, Z. Kucerovsky, A. Ieta, T.E. Doyle
University of Western Ontario, London, Ontario, Canada.
E-mail:dtrybus@uwo.ca, aieta@uwo.ca, tdoyle@uwo.ca

Abstract—Grid or mesh techniques are frequently used to approximate continuous entities that behave in a wave or fluid-like fashion. Partial Differential Equations (PDE’s) are usually involved in the description of such entities or processes. Distributed parallel computation was used in various computer cluster configurations to calculate PDE solutions of electrostatic field. The study of the efficacy of the selected architecture using mesh techniques was intended. The match between the algorithm and the architecture in achieving maximum computational performance was also investigated. The developed architectures, algorithms, and findings are presented in the paper.

Keywords—Cluster computing, Parallel Processing, Grid Techniques, Electric Field, Linux.

I. INTRODUCTION

Numerous problems that arise in engineering can be visualized as a 2-dimensional grid where the values of the individual elements vary over time in response to the values of neighbouring elements. Such problems include electric field intensity, thermal conduction, and atmospheric modeling. Usually these problems can be characterized by the rate of change in two or more independent variables. PDEs describe such processes; the solutions of PDEs can be implemented and computed in a data parallel manner using arrays to store their discrete representation [3].

Most commonly used algorithms compute the grid values by repetitive calculations of the averages of the neighbouring grid points in a sequential manner. Such algorithms require access to all points of the grid and thus are not well suited for Non Uniform Memory Access (NUMA) machines. In order to implement the algorithm on a NUMA machine and compute the grid values in parallel, additional communications are required after each iteration, to ensure that the boundaries of the divided mesh are continuous.

A parallel mesh computing algorithm has been developed and implemented on a six-computer Linux cluster. Various sized grids were computed and their effect on the computation process observed. The main objectives of our experiments were the study of the reduction of the computation time required to calculate the grid values by multiple computers running in parallel; and the assessment of the accuracy.

II. THEORETICAL APPROACH

Problems where each point in the grid has the same computational requirement are quite often called uniform. PDE’s are commonly used to solve uniform grid problems. The Laplace’s equation that governs steady-state distribution of electrical potential on a plane was used to implement the following algorithm [5]:

\[
 u(x, y) \approx \frac{1}{4}[u(x+1,y)+u(x-1,y)+u(x,y+1)+u(x,y-1)]
\]

(1)

The above algorithm can be implemented on a UMA machine. Distributed implementation of this algorithm requires partitioning of the grid and assigning the partitions to every participating computer. This partitioning of the data is usually done by one machine acting as the server, which is aware of the cluster configuration. Since the computed data resides on physically distinct machines, additional communications are also required in order to ensure correct grid values at the partition boundaries. The communications can either take place among the participating machines or be performed between the participants and the machine acting as the server. The later approach was chosen, as it is the server that assigns the data and is aware of the boundaries resulting from the partitioning of the data. The communications can be performed in a synchronous or asynchronous manner. The synchronous type of communication was implemented because the number of data points required to compute the grid values at the boundaries is only \(4N^1\) per participant, and all participants have the same computational power. Figures 1(a) and 1(b) illustrate the Cluster Algorithms.\(^2\)

\[
\begin{align*}
\text{Begin} & \\
\text{Set Communications} & \\
\text{For}(m=0; m<CMs; m++) & \\
\text{SendDimms}(n/CMs, n) & \\
\text{SendMatrix}(n/CMs[m]) & \\
\text{For}(i=0; i<\text{Iters}; i++) & \\
\text{CalcGrid}(\text{Matrix}[n][m]) & \\
\text{SendBounds}(\text{Top2Rows}) & \\
\text{SendBounds}(\text{Bot2Rows}) & \\
\text{RecBounds}(\text{Top2Rows}) & \\
\text{RecBounds}(\text{Bot2Rows}) & \\
\text{End} & \\
\end{align*}
\]

(a) Member

\[
\begin{align*}
\text{Begin} & \\
\text{Set Communications} & \\
\text{For}(m=0; m<\text{CMs}; m++) & \\
\text{SendMatrix}(n/\text{CMs}[m]) & \\
\text{For}(i=0; i<\text{Iters}; i++) & \\
\text{CalcGrid}(\text{Matrix}[n][m]) & \\
\text{SendBounds}(\text{Top2Rows}) & \\
\text{SendBounds}(\text{Bot2Rows}) & \\
\text{RecBounds}(\text{Top2Rows}) & \\
\text{RecBounds}(\text{Bot2Rows}) & \\
\text{End} & \\
\end{align*}
\]

(b) Server

![Fig. 1. Cluster Algorithms.](image)

1Where \(N\) is the size of the grid.
2The algorithms have not been optimized for the sake of clarity.
server). All cluster members’ critical computational hardware components were identical (CPU speed, cache, and memory size). A dual processor system was chosen for the server as it was responsible for the following operations: data generation, data partitioning, assignment of the partitions to the participating cluster members, and collection of the experimental results.

The computation involved calculating the values of potentials at the grid representing the plate as shown in figure 2. To ensure a satisfying convergence result, ten iterations of the calculation were performed. The size of the mesh was varied from 100 to 3000 at 100 point increments. The computations were performed by a Linux cluster whose configuration was varied from two to six computers. The SpeedUp\(^3\) and the Efficiency\(^4\) of each configuration were recorded and analyzed.

![Fig. 2. Grid for difference equation solution.](image1)

![Fig. 3. Six computer mesh calculations.](image2)

IV. DISCUSSION

There were two main objectives of the experiment. Firstly, the grid values calculations are to be optimized by multiple computers running in parallel. Secondly, the correctness of the result produced by the distributed computation needed to be assured. Sample results produced by the six-computer cluster are plotted in figure 3. The execution time was shortened considerably and a speed up factor of 4.5 was observed for large meshes. The collected speedup results are listed in figure 4.

The performance of the cluster improves with the increase of the grid size. For low data sets the time spent on communications is much greater than the computation time and thus the performance of the cluster oscillates. All oscillations disappear for meshes 1500 and greater.

![Fig. 4. System SpeedUp.](image3)

![Fig. 5. Modeled system efficiency.](image4)

V. CONCLUSIONS

The efficiency degrades with the increase of the number of computers in parallel. The degradation is mainly a result of all of the machines sharing the same network. Since the partitioned data is sent simultaneously, the number of collision and network congestion increase, leading to an increase in the time required for the communications. The observed efficiency of the system for the largest data sets is plotted in figure 5. Regression analysis indicate that

\[ \text{SpeedUp}(N) = \frac{\text{Experiment Execution Time of 1 Cluster Member}}{\text{Experiment Execution Time of N Cluster Members}} \]

where the execution time is the wall clock time.

\[ \text{Efficiency}(N) = \frac{\text{SpeedUp}(N)}{N} \times 100\% \]

the system would scale for this particular problem quite well up to twelve machine configuration, where it would still perform at 50% efficiency.

Our research investigated the efficacy of the selected architecture in solving an important engineering problem, namely, the approximation of the electric field using mesh techniques. It was observed that it is be possible to divide the mesh into several parts and compute the grid values in parallel. While additional communication is required to ensure that the boundaries of the divided mesh are continuous, the induced overhead is negligible, especially for large data sets. The research also investigated the importance of the match between the algorithm and the architecture in achieving maximum computational performance. The maximum speedup of a six computer cluster was observed to be 4.5 with the efficiency of 74%.

REFERENCES