Parallel Computation of Electric Potential in the EHD Ion-Drag Micropump and the Performance Analysis of the Parallel System

[1]Shakeel Ahmed Kamboh, [2]Jane Labadin, [3]Andrew Ragai Henry Rigit
[1,2]Faculty of Computer Science & Information Technology
[3]Faculty of Engineering
Universiti Malaysia Sarawak
94300 Kota Samarahan, Sarawak, Malaysia
[1]shakeel.maths@yahoo.com, [2]ljane@fit.unimas.my, [3]arigit@feng.unimas.my[3]

Ling Teck Chaw
Faculty of Computer Science & Information Technology
University of Malaya
50603 Kuala Lumpur, Malaysia
tchaw@um.edu.my

***Abstract*—The numerical solution of a computationally intensive model becomes more complex in terms of execution time required by a single processor. To speedup the computation, a suitable parallel computing architecture is required. This paper attempts to achieve a fast finite difference solution of electric potential in an EHD ion-drag micropump. A 2D Poisson's equation is solved on a cluster of low cost computers using MATLAB. Numerical solution is obtained for the different mesh refinements and then the execution time, communication time, speedup and efficiency of parallel system are analyzed. The results showed that the speedup and efficiency of the system increases by increasing the grid points. The results also reveal that for each data size there is an optimum number of workers for obtaining the parallel numerical solution in minimum processing time.**

Keywords-Poisson's equation; parallel numerical solution; finite difference methods; modeling and simulation

I. INTRODUCTION

The accuracy of numerical solution of the partial differential equations depends on the fine discretization of the computational domain and the large number of iterations. Particularly, when the geometry is nonlinear or irregular, the size of data set increases significantly. In such situation, the complexity of the numerical algorithms increases and demands greater computational resources. Computational scientists, who have conventionally run their numerical algorithms on a single computer, are experiencing unacceptably long run-times [1]. The objective of this work is to investigate and analyze the computational time required for different grid sizes. A simple parallel computing algorithm was implemented on MATLAB parallel computing environment based on a single program multiple data (SPMD) model.

This paper begins with the introduction of the problem and the motivation of the study. The rest of the paper is arranged as follows: In Section II, some related works are reviewed. Section III describes methodology and data parallelization of the problem. The results are discussed in Section IV. Finally, the concluding remarks and directions for future work are given in Section V.

II. BACKGROUND

Parallel computing is a type of computation in which complex and large calculations are carried out simultaneously by dividing the problem into smaller ones, which are then solved in parallel [2]. Among the various parallelism methods, the most commonly used are the task parallelism, pipeline parallelism, data parallelism or combination of these three methods [3, 4]. For the numerical solution of partial differential equations that is obtained at each grid point or for each discrete element, the data parallelism is a suitable choice. Data parallelism is especially useful for pixel (points) and block-wise operations that can be efficiently parallelized when the number of parallel nodes is large [5]. The initial data is distributed among the different computers also called the workers followed by a proper distribution or partitioning algorithm (Fig.1). The same task is executed by all the workers such that each worker computes its own piece of data.

Parallelism has been successfully used in many domains of science and technology, such as high performance computing (HPC), servers, graphics accelerators, and many embedded systems. A variety of computational problems have been solved by using different high performance computing tools and architectures. Although, many [6-11] high speed computing architectures reported in the literature have been used for HPC but in case of unavailability of supercomputing infrastructure the cluster of available computers can be utilized. The clusters are useful for

978-1-4799-1092-2/13/$31.00 ©2013 IEEE

parallel and/or distributed computing and have the advantages of low cost, flexibility of configuration and upgrade, and scalability to meet the size and time requirements for the specific workloads [12]. This paper focuses on the solution of computationally intensive applications on a remote cluster of computers using MATLAB.

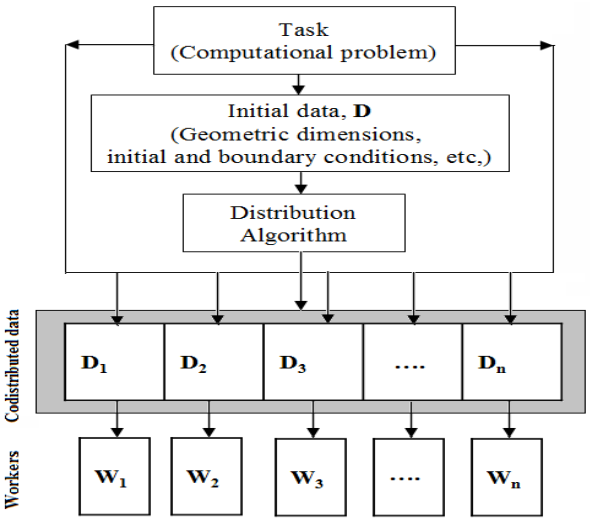

Fig.1 Data parallelism for a computational problem

III. PARALLEL NUMERICAL SOLUTION OF POISSON'S EQUATION

The Poisson's equation is an elliptic type partial differential equation and is one of the basic governing equations of Electrohydrodynamics. This equation is a direct consequence of Gauss's law for electricity and relates the electric potential with the space charge density [13]. Finding the electric potential and the electric field distributions is an important practical problem in EHD ion-drag pumping. The general form of Poisson's equation in Cartesian coordinates is given as follows,

$$\Delta V = -\frac{q_e}{\varepsilon}, \tag{1}$$

where V is the electric potential (Volts), q_e the space charge density (C/m^3) and ε the relative permittivity (F/m). For a two dimensional case the above equation can be written with respect to its coordinates as,

$$\frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} = -\frac{q_e}{\varepsilon}, \tag{2}$$

where both V and q_e are the functions of x and y. In order to obtain the numerical solution the Eq. (2) is discretized using central finite difference schemes and is given by,

$$\frac{V_{i+1,j} - 2V_{i,j} + V_{i-1,j}}{h^2} + \frac{V_{i,j+1} - 2V_{i,j} + V_{i,j-1}}{k^2} = -\frac{q_{e\ i,j}}{\varepsilon} \tag{3}$$

where i and j represent the nodes location while h and k are the step sizes in x and y directions respectively. The description of charge density and initial and boundary conditions can be found in [14-15]. Eq. (3) is expressed explicitly for $V_{i,j}$ and solved by using the Gauss-Siedel iterative method [16].

To compute parallel solution, Eq. (3) is parallelized using data parallelism technique such that the computational domain is partitioned into p horizontal strips (sub-domains), where p is the number of processors or workers. Each subdomain is distributed to each distinct worker and the solution task is passed on the corresponding codistributed data. The interior points of each subdomain are computed locally by each worker while there is a data dependency for border elements (halo points) between each two workers that need necessary communication. This task is done by using message passing functions along all the neighborhoods between the sub-domains. The borders elements are located by their indices values i and j and are updated simultaneously using the message passing functions. The process remains continue until the solution values reach to a predefined error tolerance.

In order to implement the solution algorithm on a parallel system the MATLAB distributed and parallel computing environment was used [17-18]. A remote cluster of eighteen desktop computers was configured as one **Master** or Client; one **Job Manager** and the rest of them were set as **Workers**. The description of the system is given in the table 1. The node status of the system was tested for interactive parallel computation and then the numerical solution algorithm was implemented. The average execution time and the maximum communication time were noted down for the different mesh refinements such as 64x64, 128x128, 192x192, 256x256 and 320x320.

The performance of the parallel system was evaluated by the following metrics [19];

speedup, $S = T_1 / T_p$, (4)

parallel efficiency, $E_p = S / p$, (5)

computational cost, $C_{par} = pT_p$, (6)

where T_1 (sec) is time spent for the algorithm for the solution of a given problem on a single processor (worker), T_p (sec) is the time, which is needed for the algorithm on a parallel architecture with p number of processors of the same type.

Table 1. Hardware and software used for the parallel computing

CPU	Intel ® Pentium (R) D, DELL
Processor	2.80 GHz
Memory	2.00 GB
Interconnection speed	1.00 Gbps Ethernet
Operating System	Windows 7-Professional 32-bit
Parallel Environment	MATLAB Parallel/Distributed Computing