Wired and Wireless Parallel Simulation of Fluid Flow Problem on Heterogeneous Network Cluster

Fariha Quratulain Baloch,
SE Development Switching, PTCL, Hyderabad, Sindh, Pakistan

Mahera Erum Baloch
Institute of Computer Engineering, University of Duisburg-Essen, Campus Duisburg, Germany

Abstract

In this research Homogeneous and Heterogeneous networks will be analyzed using parallel processing technique for highly sophisticated research problem. During the analysis sample networks will be considered and QoS parameters will be observed using simulation for end-to-end services.

Analysis of the wireless LAN system is carried out and its efficiency is measured compared to the wired systems, for the purpose of designing and implementing the parallel processing techniques used to solve general purpose engineering problems. The basic purpose for this research is to provide improved QoS in networks for research and academic learning.

Keywords: *Wired and Wireless, Parallel Simulation, Finite Element Method, Rotating Mixing Flow, Non-Newtonian Fluids.*

1. Introduction

In recent years, parallel computation has become increasingly more important for solving large-scale computational fluid dynamics problems, which arise in many areas of science and engineering involving both compressible and incompressible flow regimes. Particular interests are in incompressible complex flows of non-Newtonian fluids, of immediate relevance to the processing industries, associated with electromagnetic, electrostatic, polymer, foods, cosmetics, oil products and etc. The mathematical modelling of such flows, typically, generates complex three-dimensional systems of partial differential equations of mixed-type. Common discretisation approaches adopts, such as finite difference, finite element, finite volume or spectral element formulations, to transform these systems from differential to algebraic form, generating large numbers of degrees-of-freedom. Over the preceding decade, with the advancement of computer hardware and developments in sophisticated numerical algorithms, it has become easier to solve complex flows, albeit of limited size. To make satisfactory progress in this area often fine resolution meshes are required, that may also involve adaptive meshing. This places practical limitations upon the range and scope of the problems that may be tackled in terms of size, necessitating a shift from a sequential to a parallel mode of computation.

Parallel computation may be viewed in a distributed manner, where memory and processors are distinct, or in a combined form where memory is shared. The distributed model involves sending and receiving messages and configuring a processor network (e.g. Master/Slave). Over the last few years there has been an increase in the availability of software to perform such message passing. Recent advancements in parallel computing different message passing protocols have been developed, such as Network Linda (LINDA), Message Passing Interface (MPI), Parallel Virtual Machine (PVM) and many more. These protocols have given impetus to the design of parallel algorithms to solve very large tedious and time consuming problems. Present study, following the study of [1 and 2], PVM version 3.4.4 is adopted for using a Master/Slave configuration.

In the solution of complex flows, inverting and resolving linear systems of equations constitutes a large proportion of CPU time overhead. The nature of the problem to be solved may be coupled, leading to large system matrices or alternatively, decoupled which may be handled iteratively. Some examples of using PVM for parallelising numerical codes are described in [1, 2].

2. Problem Definition

Here the problem investigated, is two-dimensional mixing flows of Newtonian fluids, of relevance to the food industry such as occurs in dough kneading. Such flows are rotating, driven by the rotation of the outer containing cylindrical-shaped vessel. The two stirrers are securely-held in place by being attached to the lid of the vessel. In reality, within the industrial process, the lid of the vessel would rotate with stirrer attached with a double stirrers, an eccentric configuration is adopted.

The problem is analysed for rotating flow with stationary stirrers in rotating cylindrical vessel, the numerical predictions are compared against the results obtained in previous investigations [1, 2].

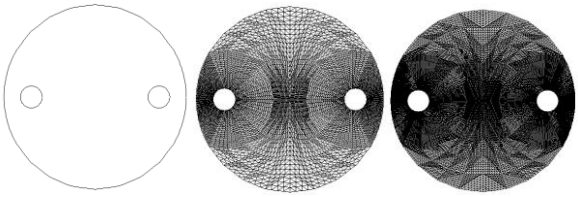

Fig. 1: Domain of interest and finite element meshes [Baloch]

Domain and finite element mesh for the problem involved is displayed in Fig. 1. In this investigation two consecutive refined meshes are used to demonstrate scalability of the parallel performance. For first mesh the total number of elements, nodes and degrees-of-freedom are 3840, 7840 and 17680 respectively. However, for second mesh these parameters are four times [6].

To provide a well-posed specification for each flow problem, it is necessary to prescribe appropriate initial and boundary conditions. A quiescent initial state inaugurates the simulations. Boundary conditions are taken as follows. For stationary stirrers the fluid stick to the solid surfaces, so that the components of velocity vanish on the solid inner stirrers sections of the boundary ($v_r = v_\theta = 0$) [6].

3. Parallelisation Strategy

For any fixed mesh, the performance of these parallel implementations are presented demonstrating monotonically increasing speed-up and monotonically decreasing efficiency with increasing number of domains (processors). Running with a fixed number of processors illustrates the increase in speed-up and efficiency with an increasing number of mesh elements.

Parallelisation has been achieved using the PVM. For parallel computations, PVM version 3.4.3 developed at Oak Ridge National Laboratory has been employed as message passing mechanism. PVM supports programs written in both C/C++ and FORTRAN by providing a library of low-level communication routines.

For parallel computation, both [1, 2] have used PVM as the message passing mechanism in master/slave style. Shared memory on DEC-alpha clusters was used by [1] and distributed memory was used by [2] on Intel Linux. In present investigation the hardware platform consists of heterogeneous network systems involving a number of workstations. Initially, wired cluster was designed through three Intel processors one Intel Pentium–IV 1.7 GHz, 256 MB memory as master processor and two, one Intel Centrino 512 MB memory and one Intel Centrino core 2 Duo with 1GB memory, as slave processor. The same network cluster was also used for wireless adhoc network. These workstations communicate through a fast 100 Mbps Ethernet network for wired network and *802.11 b/g* for wireless network cluster.

The parallelisation strategies and associated test results are applicable to a wide range of CFD applications. One of the major areas of this research can be that of mathematical modelling and simulation. Another area is that of wireless sensor networks were a cluster of sensors are utilised to collect data of physical nature, such as environment monitoring of remote systems. Modelling the behaviour of human body or its parts is also a major issue and with the use of modern day technology as well as methods such as parallel processing, systems can be designed to solve hard problems.

3.1 Domain decomposition

The domain decomposition method embodies (incorporates/includes) large potential for parallelisation of finite element methods. In this approach, the domain of interest is partitioned into smaller sub-domains of desired size according to the specification of the available processors. The overall computational load may be equi-partitioned and assigned uniformly among the available processors, resulting in a uniform balancing of computational load [6].

The inter-processor communications can considerably influence the overall efficiency. In this coarse granularity implementation, each sub-domain is assigned to a processor that computes simultaneously the corresponding subsection of the sub-domain, independently. Such a

configuration would yield optimal performance when there is no communication amongst the processors [6].

One of the key issues being addressed in parallel computing these days is load balancing. At this stage, issues of dynamic load balancing are yet to be investigated. Here, static uniform load distribution is ensured, irrespective of processor speed, using a Recursive Spectral Bisection method. This, with proper granularity of parallelism, enables us to handle synchronisation of processes, sending and receiving messages, and distributing data efficiently. Finite element algorithm is inherently suitable for parallelisation through a variety of paradigms [6].

Focusing on the paradigm of domain decomposition, as finite element meshes are structured, adopting a domain decomposition approach, the meshes are partitioned into a number of equal-sized sub-domains according to the number of processors available. On each processor, calculations are performed simultaneously for each sub-domain over a set of slave processors. On the periphery of each sub-domain, shared boundary nodes are computed by a central master (control) processor. The master processor is used to gather the contributions from shared nodes that result from sub-domain processes on each processor, and subsequently, redistribute the combined information to each processor [6].

4. Numerical Scheme

The present study adopts a so called semi-implicit Taylor-Galerkin/Pressure-Correction finite element time-marching scheme that has been developed and refined over the last two decades. This scheme, initially implemented sequentially, is appropriate for the simulation of incompressible isothermal Newtonian, fibre suspension, generalised inelastic and viscoelastic flows (Townsend and Webster [1], Carew et al. [2], Baloch and Webster [3], Baloch et al. [4] and Matallah et al. [5]). Parallel implementations of this algorithm are described in (Grant et al. [1998] and Baloch et al. [6] and [7]).

This scheme is based on a fractional-step formulation that involves discretisation, initially in the temporal domain, by adopting a Taylor series expansion in time and a pressure-correction operator-split, to build a second-order time-stepping scheme. This scheme was made available for researcher. For ready reference parallel algorithm is presented in Table 1 [6].

5. Results and Discussion

In this section, parallel results are presented in the form of speed-up and loss of efficiency. Fig. 2 shows the decomposition of domains:

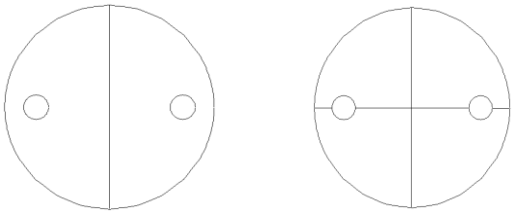

Fig. 2: Decomposed Domains

Table 1: Parallel Taylor-Galerkin Algorithm

Master processor: Enter pre-processing information; Setup Parallel Virtual Machines; Input domain decomposition information from RSB; Spawn process on slave processors; Input numerical, fluid and algorithm parameters; Inputs mesh information, and initial and boundary conditions; Decompose domain and reorder node numbering after band-width reduction, and pack all information; Distribute information to slave processors after rearranging the information; Synchronise the machines and hand-shake with slave processors; *While not converged*
Stage 1a Receive: right-hand-side for stage-1a from each slave processor; Redistribute: after combining; Loop over Jacobi iteration; Solve stage-1a for interfacing nodes;
Stage 1b Receive: right-hand-side from each slave processor for stage-1b; Redistribute: after combining; Loop over Jacobi iteration; Solve stage-1bfor interfacing nodes;
Stage 2 Build: right-hand-side for interfacing nodes on stage-2; Solve stage-2 for pressure difference using Choleski on interfacing nodes;
Stage 3 Receive: right-hand-side from each slave processor for stage-3; Redistribute: after combining; Loop over Jacobi iteration; Solve stage-3 for interfacing nodes;
Compute error norm for interfacing nodes; Test for convergence; Synchronise: the machines and hand-shake with slave processors; Receive: results from slave processors; Print combine final result;

Slave processor:
Receive: pre-processing information from master processor and unpack all information;
Synchronise: the machines and hand-shake with other processors;
While not converged

Stage 1a
Build: right-hand-side for stage-1a for internal nodes;
Send: only information of interfacing nodes to master processor;
Receive: combined information from master processor;
Loop over Jacobi iteration; Solve stage-1a for internal nodes;

Stage 1b
Build: right-hand-side for stage-1b for internal nodes;
Send: only information of interfacing nodes to master processor;
Receive: combined information from master processor;
Loop over Jacobi iteration; Solve stage-1b for internal nodes;

Stage 2
Build: right-hand-side for stage-2 for internal nodes;
Solve stage-2 for pressure difference using Choleski on internal nodes;

Stage 3
Build: right-hand-side for stage-3 for internal nodes;
Send: only information of interfacing nodes to master processor;
Loop over Jacobi iteration; Solve stage-3 for internal nodes;

Compute error norm for internal nodes, send information of interfacing nodes to master;
Synchronise: the machines and hand-shake with master processor;
Send: result to master processor;

5.1 Speed-up and Efficiency

Numerical computed results are presented for the performance of the parallel TGPC scheme by measuring the speed-up factor and efficiency, defined as $S_n = \frac{T_s}{T_n}$, $\eta_n = \frac{S_n}{n}$, where T_s is the CPU time in seconds (s) for sequential algorithm and T_n is the CPU time for the parallel algorithm for number of processors (n), while S_n is the total speed-up factor and η_n is the total efficiency for the parallel computation. CPU time T_n of parallel computation can be decomposed into computation time (T^{comp}) and communication (T^{comm}). Timing accords to the total job run-time, inclusive of the input-output and communication latency. For a fixed mesh with an increasing number of partitions, the cost of communication increases and this decreases the total efficiency of computation. The computation on a fixed number of domains and upon increasing the size of problem (mesh) increases the over-all efficiency [6].

In Table 2, timings for systems comprising of the above network clusters is reported. These results are cross-checked on other heterogeneous networks.

Table 2: Computation Time

S. #	Processors	Elements	Nodes	DOF	Time (secs)	
					Wired	Wireless
1	2	3840	7840	17680	1440	1350
2	4	15360	31220	70280	5130	4710
3	8	61440	62120	155460	10260	7830

The row 2 and 3 of this table, it is observed that by doubling the number of processors and degrees of freedom will give the same order of computation time. This is true also if we extrapolate the time from row 1, taking the size of problem to be half that in row 2. This demonstrates the scalability of our parallel algorithm. In parallel computation, better scalability depends upon ideal speed-up, whilst speed-up depends on problem size [6].

5.2 Scalability

A parallel algorithm is said to be scalable if the computation time remains unchanged when the total number of degrees-of-freedom is increased simultaneously with an increase in the same ratio, as the number of slave processors. Heterogeneous network combination is constructed to show scalability. To establish two, four and eight slave processor heterogeneous networks, with equal numbers of Intel processors windows workstations are taken in each case [6].

5.3 CPU Speedup and efficiency

Table 3: CPU Speed-up of Heterogeneous Cluster

Number of Processors	Wired Cluster	Wireless Cluster
1	1.00	1.00
2	1.92	1.90
4	3.60	3.52

The given table 3 CPU's speedup is shown as the number of processors are increased (for both wired and wireless clusters) and the result is shown in the Fig. 3.

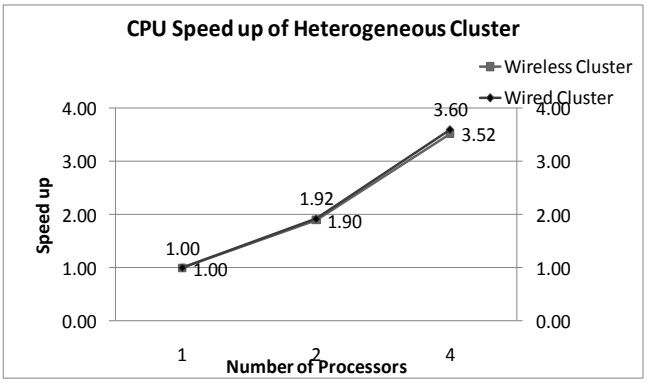

Fig. 4: Graph of CPU Speed-up of Heterogeneous Cluster

As the number of processors are increased, the computation time decreases, hence, the speed up of the network is increased.

Table 4:Efficiency of Heterogeneous Cluster

Number of Processors	Wired Cluster	Wireless Cluster
1	1.00	1.00
2	0.96	0.95
4	0.90	0.88

The given table 4 shows, efficiency of the cluster as the number of processors are increased and the result is also mapped in fig. 4.

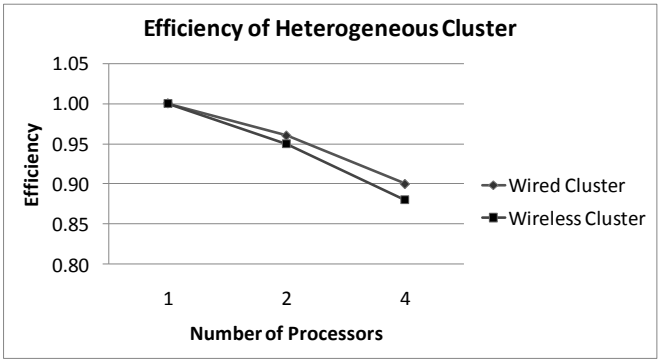

Fig. 4: Graph showing efficiency of Heterogeneous cluster

Efficiency may decrease if the same size problem is decomposed to much smaller parts, thus, increasing the communication over-head of the network.

6. Conclusions

A problems solving cost on a parallel heterogeneous cluster is the product of overall parallel runtime and the number of processing nodes that are used. Cost can also reflect the sum of the time that each processing node spends in solving the problem. Efficiency is also expressed in terms of, the ratio of the execution time of the fastest known sequential algorithm for solving a problem to the cost of solving the same problem on a number of **p** processing nodes. The cost of solving a problem on a single processing node is the execution time of the fastest known sequential algorithm. A parallel system is cost-optimal if the cost of solving a problem on a parallel node has the same asymptotic growth as a function of the input size as the fastest-known sequential algorithm on a single processing node. Since efficiency is the ratio of sequential cost to parallel cost, a cost-optimal parallel system has an efficiency of η (1) [8].

The efficiency of the parallel wireless heterogeneous networked system depends on the problem size, number of processors and the distance/media between the processing nodes.

In other words: Increasing partitions/processors will increase the cost of the communication and decrease the total efficiency of the computation. Thus, load balancing very essential. If the size of the problem is increased, but the number of processors remain constant, the processing time will increase compared to communication time; therefore increasing efficiency. Communication time depends on the type of network, its media, data rate and distance. Moving the processing nodes at a longer distance will also increase the communication time, thus, a reduction in efficiency of the network.

Acknowledgments

Authors greatly acknowledge the Mehran University of Engineering and Technology, Jamshoro for providing computing facilities and Professor A. Baloch for his valuable guidance.

References

[1] Baloch, A., Grant, P. W. and Webster, M. F., "Homogeneous/Heterogeneous Distributed Cluster Computation of Two and Three-Dimensional Viscoelastic Flows", Int. J. Numer. Meth. Fluids, 40, 1347-1363, 2000.

[2] Baloch, A., Grant, P. W. and Webster, M. F., "Parallel Computation of Two-Dimensional Rotational Flows of the Viscoelastic Fluids in Cylindrical Vessel", Int. J. Comp. Aided Eng. and Software, Eng. Comput., 19(7), 820-853, 2002.

[3] Townsend, P. and Webster, M. F., "An Algorithm for the Three-dimensional Transient Simulation of Non-Newtonian Fluid flows", in: G. Pande, J. Middleton (Eds.), Proc. Int. Conf. Num. Meth. Eng.: Theory and Applications, NUMETA, Nijhoff, Dordrecht, pp. T12/1–11, 1987.

[4] Carew, E. O., Townsend, P. and Webster, M. F., "Taylor-Petrov-Galerkin algorithm for Viscoelastic flow", J. Non-Newtonian Fluid Mech., 50, 253–287, 1994.
[5] Baloch, A. and Webster, M. F., "A Computer Simulation of Complex Flows of Fibre Suspensions", Int. J. Computers and Fluids, 24(2), 135–151, 1995.
[6] Baloch, A., Townsend, P. and Webster, M. F., "On the highly elastic flows", J. Non-Newtonian Fluid Mech. 59, 111–128, 1995.
[7] Matallah, H., Townsend, P. and Webster, M. F., "Recover and stress-splitting schemes for Viscoelastic flows", J. Non-Newtonian Fluid Mech., 75, 139–166, 1998.
[8] Grama, A., Gupta, A., Karypis, G., and Kumar, V., Introduction to Parallel Computing (2nd edn). Addison-Wesley, Harlow, UK, 2003.

Fariha Quratulain Baloch graduated in Telecommunication Engineering in 2006 and did her M. Eng. in Communication Systems and Networks in 2008 from Mehran University of Engineering and Technology, Jamshoro, Sindh, Pakistan. She is currently working as a Senior Engineer at PTCL, Pakistan.

Mahera Erum Baloch graduated in Computer Systems Engineering in 2008 and received her Master's degree in Communication Systems & Networks from Mehran University of Engineering & Technology, Jamshoro, Pakistan in 2011. She is currently studying towards a PhD degree in Computer Engineering at the University of Duisburg-Essen, Germany. Her research interests include Distributed & Parallel Computing, Performance Analysis & Evaluation techniques, Computer Supported Collaborative Work, & Artificial Intelligence.