1. INTRODUCTION

1.1 GENERAL

The generations of computers are recognized corresponding to change in hardware building blocks from relays and vacuum tubes (1939-1950), to discrete diodes and transistors (1950-1960), to small and medium scale integrated circuits (1960 - mid 1970), to large and very large scale integrated devices (mid 1970 to 1990), to ultra-large-scale integrated devices and powerful microprocessors (1990 - present). The tendency of mankind to achieve more than, what is available, caused the major developments in science and technology. The same is true in case of computational technology and its application to structural engineering. The various generations of computer have increased the speed by more than a trillion times during last five decades with dramatical reduction in cost. The continual demand for greater computational speed resulted into powerful processors with faster memories. But the increase in speed of computers may be limited due to factors like energy dissipation and packing of transistors [1]. This has caused use of more number of processors for solving a single problem. The process of splitting overall problem into smaller parts and assigning each part to separate processors to increase the throughput is known as parallel processing. The multiple processor system operating closely on a problem is generally called, as parallel processing while use of separate computers for solving a problem is specifically known as distributed processing.

The major area, which requires greater computational speed, includes numerical modeling and simulation of scientific and engineering problems including artificial intelligence, numerical analysis, image processing, etc. [2]. Such problems often require huge repetitive calculation on large amounts of data to get valid results. The engineering computation and simulation must be achieved within reasonable time period; otherwise simulation requiring more time to reach solution is unacceptable. Some of the examples requiring increased computational speed are modeling large DNA structures, weather forecasting, prediction of motion of astronomical bodies in space, seismic modeling, finite element analysis of large complex structures, computational fluid dynamics, natural language processing, speech reorganization, applications of virtual reality and many more [3].

Recent advances in computer hardware and software have made multiprocessing in general and parallel processing in particular a viable and attractive technology for computational intensive problems [4]. Apart from obtaining increased speed on an existing problems, the use of multiple computers / processors often allows a larger or more precise solution of a problem to be solved in reasonable amount of time.

In this chapter the advances in computer hardware and networking responsible for parallel and distributed processing is discussed. Also various computer architectures and their performance evaluation are covered. Various algorithms for parallel and distributed computing with regards to structural analysis and design are discussed. Considering latest trends in computing, finally objectives and scope of the present work is presented.

1.2 REVIEW OF ADVANCES IN HARDWARE & NETWORKING

1.2.1 Microelectronics and Semiconductor Technology

The advances in hardware components in the last three decades have occurred as a result of developments in microelectronics [5]. Instead of connecting discrete components together by wires to produce a circuit, complete circuit patterns, components, and interconnections are placed on a small chip of semiconductor material. The predominant semiconductor material in use to date is silicon. A better understanding of this material, as well as better processing, tooling and packaging techniques, enabled the design of fast, dense circuitry. The principal advantages of microelectronic circuits are their reliability, low cost, and low power consumption. The ever-increasing number of devices packaged on a chip has given rise to the acronyms SSI, MSI, LSI, VLSI, ULSI, and GSI, which stand for small-scale, medium-scale, large-scale, very large-scale, ultra large-scale, and giga-scale integration, respectively. The full range of hardware components are now available on microelectronic chips; these include memory units, addressing units, complete central processing units (CPUs) called microprocessors, and even complete microcomputers (which include the CPU, memory, and input/output functions, all residing on a single chip). The net effect of the aforementioned developments has been a sustained reduction in the cost of computing.

1.2.2 Memory Systems

Performance of a memory system is measured in terms of latency and bandwidth. The latency of a memory is time required by memory system to produce the result of the request. The bandwidth is the rate of the memory system to accept requests and produce results. Recent progress in memory systems includes development of an entire hierarchy of addressable memories, and of high-speed, random-access memory chips with many bits of data. Instead of just a few registers in the CPU and a single-level memory, a typical machine may now have: (a) a number of high-speed, general-purpose registers in the CPU; (b) a cache memory (or instruction buffers) for very rapid access to small amounts of data; (c) standard main (local or remote) memory; (d) hardware-implemented virtual memory, extending the amount of addressable space, with the help of an auxiliary (backup) memory such as magnetic disks and tapes, solid state and optical disks, disk arrays, and mass robotic media.

The several types of semiconductor memories are random-access and read-only memories. In random-access memory (RAM), data can be written into, or read of, any storage location without regard to its physical location relative to other storage locations. Read-only memory (ROM) contains a permanent data pattern stored during the manufacture of the semiconductor chip in the form of transistors at each storage location that are either operable or inoperable. RAM can be either static or dynamic. Dynamic RAM (DRAM) requires constant refreshing to maintain its data, while static RAM (SRAM) does not. However, advances in DRAM permit double the amount of RAM at about the same cost as static RAM, so DRAM is used more frequently. The high-end microprocessors are likely contain intelligent random access memory (IRAM)-an entire computer on a single chip.

1.2.3 Secondary Storage Devices

The mass storage industry has developed technologies for handling petabytes (10^{15} bytes) of data today and is developing technologies for handling exabytes (10^{18} bytes) in the future. Technology advancements include increasing the density of storage media such as disks and tapes, RAID (Redundant Arrays of Inexpensive Disks); and robotic tape storage systems.

1.2.4 User Interface Hardware and Software

The evolution of computer systems has been tightly connected with the advances in the interaction techniques, aimed at improving the productivity of the analyst and designer. Current user-interface software includes DOS, OS/2, Unix / Linux based systems, Windows NT / 95 / 98 / 2000 / XP etc. For the PC sector, the standard interfaces are the Windows, Icons, Mouse, Pull-down menus (WIMP). Multimedia workstations and virtual reality facilities that embrace all forms of human communication and provide elaborate graphics, video, animation and visualization capabilities to convey enriched information to the user have been developed. Future advanced human-computer interface facilities will increase the communication bandwidth between the computer and the human mind and will provide the user with the freedom to choose from a variety of communication modes (e.g. voice, electronic pad that responds to handwriting, sensors that track the eye movement, and a glove that enables the wearer to manipulate objects on the screen). They will incorporate HD technologies and sonification facilities for mapping of data to a sound domain.

1.2.5 Distributed Computing and Networking

For high-performance engineering systems to support collaborative computing, layers of communication lines and devices such as local-area networks (LAN); backbone networks connecting supersystems, mass storage, and general purpose mainframes; and wide area networks (WAN) connecting geographically remote computers and terminals are used.

Local-area networks facilitate the interconnection of a variety of computer-based equipment (including personal computers, workstations, and superservers) within a small area. Backbone networks can provide a communication service between several local-area networks, as well as between different supersystems in a large industrial complex, or a university campus. The wide-area network (WAN), connecting a number of government supported and commercial packet backbone, demonstrated the feasibility and practicality of distributed computing, as well as of communication technology based on packet switching. Current work is directed towards increasing the transmission rates of local-area, backbone and wide-area networks. The Internet is becoming the global infrastructure these days. The recent developments are in direction of ATM / SONET (Asynchronous

Transfer Mode / Synchronous Optical Network) for gigabit speed, all-optical networking technology, networking technology to support portable computing and communication. As ATM is protocol independent, it can integrate LANs and WANs and is the technology of choice for LANs, wireless access and backbone carriers.

1.3 CLASSIFICATIONS OF COMPUTER ARCHITECTURES

There are different types of architectural model employing some kind of parallelism. The most well known classification of computer architecture is based on the concurrency in instruction control and concurrency in execution as proposed by Flynn [6]. A stream is defined as a sequence of items (instructions or data) as executed or operated on by a processor. Four broad classes can be identified according to whether the instruction or data streams are single or multiple as follows:

1.3.1 SISD – Single Instruction Single Data

This is the traditional sequential computer consisting of one processor, memory and one communication channel as shown in Fig. 1.1. This computer architecture will always be restricted by the part of the three elements with the least capacity. Present technology provided almost unlimited memory and extremely fast central processing units but the bottleneck is the communication channel.

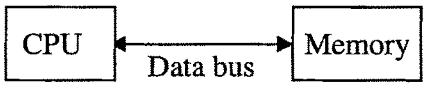


FIG. 1.1 SEQUENTIAL COMPUTER ARCHITECTURE (SISD)

1.3.2 SIMD – Single Instruction Multiple Data

In this only one instruction stream operate on multiple data streams, which are active at a time. This architecture may be available in shared and distributed memory systems as shown in Figs. 1.2 and 1.3. This architecture overcomes bottleneck of SISD system by implementing multiple processors and memory module / modules working on single instruction set. Many large parallel computers use this principle, taking advantage of the simplified programming task as the number of data set are program independent and same instruction set is performed on all processors at any given time synchronously. This leads to data parallelism where problem should be highly regular. As each processor must share the same communication channel, the host communication may prove to be troublesome. Vector computers and array processors are example of this category.

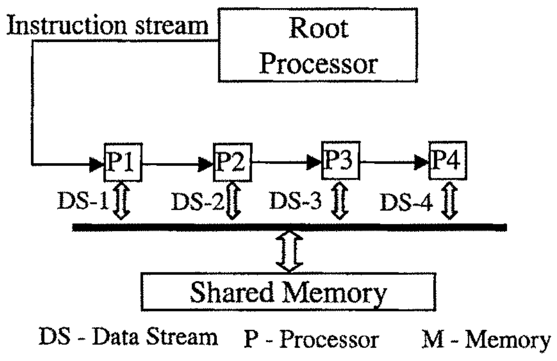


FIG. 1.2 SIMD MODEL WITH SHARED MEMORY

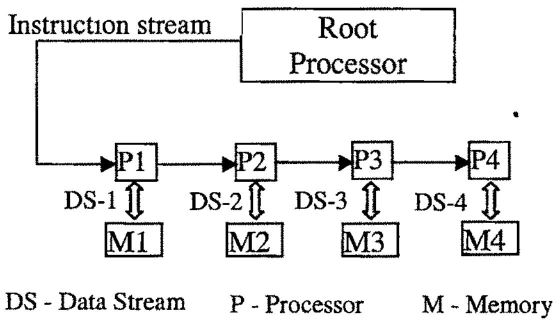


FIG. 1.3 SIMD MODEL WITH DISTRIBUTED MEMORY

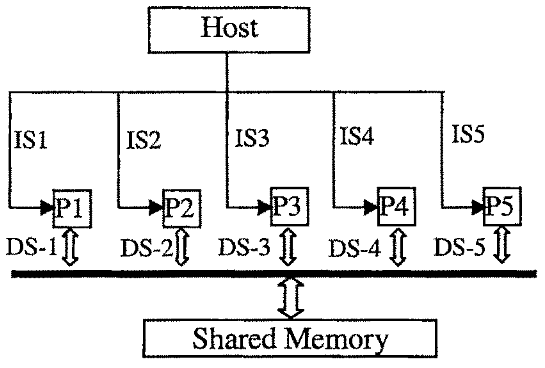
1.3.3 MISD - Multiple Instruction Single Data

This is the opposite of SIMD. By definition it means multiple instruction on a single stream of data. This appears counter intuitive and hence many people believe that MISD is impossible category. However some are placing systolic array architecture in this category.

1.3.4 MIMD - Multiple Instruction Multiple Data

The other end of spectrum would be to let loose both the streams. There are multiple data and multiple instructions and arbitrary combination of instruction and data stream is possible. This is the most general model in Flynn's classification. This architecture may again be subdivided into shared memory and distributed memory systems. The shared memory system as shown in Fig 1.4 has disadvantage that the processors may have to queue to access the same data of memory. The distributed memory system as shown in Fig 1.5 has number of processors with its own local memory and communicates with other processors through channels. MIMD computers are more versatile and more widespread. They may have same or different programs running on each processor. Programs may execute asynchronously. The distributed memory implementation overcomes both the host communication problem in case of

SIMD and bottleneck created by several processors queuing to access the same data in case of shared MIMD.



IS - Instruction Stream P- Processor DS - Data Stream

FIG. 1.4 SHARED MEMORY MIMD MODEL

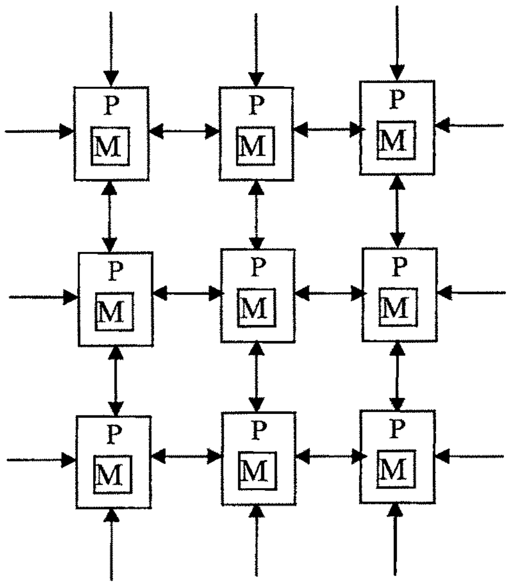


FIG. 1.5 DISTRIBUTED MEMORY MIMD MODEL

In MIMD category the system can be either multiprocessor or multicomputer. A multiprocessor system is single computer with multiple processors which communicates and cooperates at different levels in solving a given problem. The communication may occur by sending messages from one processor to the other or by sharing common memory. A single operating system governs the activities in such a computer. A multicomputer on other hand is a collection of multiple interconnected computer systems, each of which is an autonomous machine having private local memory, resources and an operating system controlling its operations. Communication takes place across the computers by message passing. A parallel program using multicomputer consists of multiple processes executing on these computers and communicating to achieve a common goal. Depending on cost incurred in communicating between processes running on the processors in multiprocessors and multicomputers , these machines are also classified as tightly coupled parallel machines and loosely coupled parallel machines. In former case the cost of communication is much lower than that in the later. The contention for memory is much higher in the case of tightly coupled machines compared to loosely coupled machines. As there is no shared memory across processors in loosely coupled machines, there is no memory contention in it.

1.4 PERFORMANCE EVALUATION OF COMPUTER SYSTEM

The performance of sequential computers is usually measured by three terms:

(1)The peak computational speed measured in millions (or billions) of floating point operations per second (Mflop/s or Gflop/s);

(2)The peak execution rate of arithmetic, logic, and program control instructions, measured in millions (or billions) of instructions per second (Mips or Gips);

(3)The peak rate of knowledge processing in millions (or billions) of logical inferences per second (Mlips or Glips).

In the case of SIMD and MIMD machines, the peak computational speeds in terms of millions (or billions) of floating-point operations per second can be estimated. However, the sustained computational performance in these machines is difficult to estimate since it varies with the level of parallelism achieved, and the overhead incurred in exploiting the parallelism in the particular application. These, in turn, are functions of the formulation used, the numerical algorithm selected, the computer implementation, the compiler and the operating system used, as well as the architecture of the hardware [7].

Parallel processing can be applied at four distinct levels, namely: job level, program level, inter-instruction level, and intra-instruction level, which can be achieved by hardware and software as shown in Table 1.1. The hardware role increases as the parallel processing goes from high (job) level to low (intra-instruction) level. On the other hand, the role of software implementations increases from low to high levels.

1.5 NUMERICAL ALGORITHMS AND COMPUTATIONAL STRATEGIES

Suitable computational strategy and numerical algorithms should be selected depending upon architecture of the computer, to achieve high performance from any computing system, or to select the architecture, which may effectively map the computational strategy and execute the numerical algorithm [8].

TABLE 1.1 LEVELS OF PARALLELISM

Level	Means to achieve parallelism	Performed by
Job level	Multiprogramming - overlapping and interleaving the Operating system execution of more than one program (I/O and O/P operations). Multiprocessing - running two or more CPUs concurrently on different applications or on independent job streams of the same general application.	Operating system
Program level	Multitasking decomposition of a program into two or more tasks (program segments) that can execute concurrently. This requires the tasks to have no data or control dependence. Microtasking - which permits more than one CPU to work on a program at the Do-loop level.	Software
Inter-instruction level	Concurrency among multiple instructions-this requires an analysis of the data dependency and is accomplished by dividing each instruction into suboperations, and overlapping the different suboperations on different instructions.	Compiler
Intra-instruction level	Pipelining by dividing the instruction into a sequence of operations, each of which can be executed by a specialized hardware stage that operates concurrently with other stages in the pipeline. Very-long instruction word (VLIW) performing multiple operations per instruction, each with its own address field	Hardware

In parallel algorithms, independent computations are performed in parallel (i.e. executed concurrently). To achieve this parallelism, the algorithm is divided into a collection of independent tasks which can be executed in parallel and which communicate with each other during the execution of the algorithms. Parallel algorithms can be characterized by the following three factors:

- Maximum amount of computation should be performed by a typical task before communication with other tasks or process;
- Interprocess communication topology, which is the geometric layout of the network of task modules;
- Executive control to schedule, enforce the interactions among the different task modules and ensure the correctness of the parallel algorithm.

The design of a parallel algorithm must deal with a host of complex problems, including data manipulation, storage allocation, memory interference, and in the case of parallel processors, interprocessor communication. In general, the parallel numerical algorithms fall into two categories: reformulation (or restructuring) of serial algorithms into concurrent algorithms, and algorithms developed especially for parallel machines. Generally parallel numerical algorithms belong to the first category, i.e. decomposition of serial algorithms into concurrent tasks. The second category includes the algorithms whose development was spurred by performance criteria for parallel processing which is referred to as uniquely parallel.

There are two approaches for design of parallel algorithms: Divide and conquer, and reordering. The divide and conquer approach involves breaking a task up into smaller subtasks that can be treated independently. The degree of independence of these tasks is a measure of the effectiveness of the numerical algorithm, since it determines the amount and frequency of communication and synchronization. Reordering refers to restructuring the computational domain and/or the sequence of operations in order to allow concurrent computations. For example, the assembly strategy, node-by-node or element-by-element assembly, may change the degree of parallelism that can be achieved in the solution of the resulting algebraic equations.

The comments concerning parallel algorithms and their implementation are as follows:

- Effective parallel algorithms are not necessarily effective on sequential computers. In fact, some parallel algorithms involve additional (redundant) floating point operations which make them inefficient on sequential machines. Also, restructured serial algorithms may not be the most efficient on parallel processing machines.

- The rate of convergence and numerical stability of serial and parallel iterative techniques can be different.

- The numerical algorithm and its implementation depend on the type of computing system to achieve high performance, which raises the question of portability of parallel programs. It is not practical to develop algorithms and programs for each new computer. Also, it is not desirable to achieve portability at the expense of performance.

- On most of the multiprocessing systems, vectorization offers greater performance improvement over multitasking (i.e. parallelization). Consequently, if multitasking conflicts with efficient vectorization, then the algorithm should be vectorized rather than parallelized.

1.6 ALGORITHMS FOR STRUCTURAL ANALYSIS AND DESIGN

Several attempts have been made in recent years to exploit the potential of parallel processing machines in the solution of various structural analysis and design problems. These include finite element computations on SIMD vector computers, shared memory multiprocessors, and message-passing multicomputers [9].

Different phases involved in the steady-state finite element analysis, and their suitability for vectorization and parallelization is listed in Table 1.2. For time-dependent or transient problems, several parallel temporal integration techniques have been proposed for structural dynamics problems. Explicit schemes are well suited to both vector and parallel processing. This is especially true when a lumped mass matrix is used. A number of special strategies can be used to increase the degree of parallelism and/or vectorization in finite element computations. These strategies are applications of the principle of divide and conquer, based on breaking a large (and/or complex) problem into a number of smaller (and/or simpler) subproblems which may be solved independently on distinct processors. The degree of independence of the subproblems is a measure of the effectiveness of the algorithm since it determines the amount of frequency of communication and synchronization. The computational strategies developed can be classified into four major categories and possible combinations: Domain-wise strategies, Operator-splitting techniques, Column-wise (or row-wise) algorithms; and Node- or Element-wise algorithms. The last two categories of numerical algorithms allow only small granularity of the parallel tasks. On the other hand, the first two categories allow large granularity, which can improve the performance of the numerical algorithms.

TABLE 1.2 DIFFERENT PHASES OF STATIC STRUCTURAL ANALYSIS

Phase	**Suitability for Parallelization / Vectorization**
Input problem characteristics, element and nodal data, and geometry	Can be parallelized
Evaluation of element characteristics	Easy to parallelize and can be vectorized
Assembly	Requires special care for parallelization (e.g. node-by-node strategy) and difficult to vectorize
Incorporation of boundary conditions	Easy to parallelize
Solution of algebraic equations	Important to vectorize and parallelize
Postprocessing	Can be parallelized and vectorized

Three commonly used strategies are discussed below:

- **Domain Decomposition and Substructuring**

  The basic idea of domain decomposition is to divide the domain into a number of (possibly overlapping) regions. The initial/boundary-value problem is decomposed into one that involves solution of initial/boundary-value problems on subdomains, thereby introducing spatial parallelism. Since the solution is not available at the interfaces between regions, it is modified iteratively as part of the solution procedure. Substructuring techniques are closely related to domain decomposition. They can also be identified at the algebraic level by partitioning the associated matrices in an appropriate way to separate the degrees of freedom that are to be eliminated (the internal degrees of freedom in different substructures) from those to be retained (interface degrees of freedom). Substructuring techniques were found to be effective in reducing both the total computational effort and the storage requirements. The performance of substructuring techniques depends on balancing the computational load across processors in a way that minimizes interprocessor communication.

- **Operator Splitting**

  The notion of splitting has long been used to synthesize the solution of a complicated problem from that of a simpler problem (or a sequence of simpler problems). Among the different applications of splitting are the breaking of a multidimensional problem into a sequence of one-dimensional problems, and the development of iterative (and semi-iterative) techniques for solution of algebraic equations. Splitting can be used as a means of partitioning the computational task into a number of subtasks that are either independent, or only loosely coupled, so that the computations can be made on distinct processors with little communication and sharing.

- **Element-by-Element Solution Strategies**

  The modular element-by-element logic inherent in the finite element analysis procedure has been used to develop solution strategies which do not require the explicit generation of the global stiffness matrix. The frontal elimination method was originally proposed by Irons to bypass the assembly process. In recent years element-by-element strategies have been developed for the solution of static and dynamic problems as well as adaptive grid generation. Element-by-element strategies are well-suited to vector and parallel processing and, therefore, should be seriously considered for use in parallel processing machines.

1.7 LATEST TRENDS IN HIGH PERFORMANCE COMPUTING

1.7.1 Cluster Computing

Most computing today is done on computers that are connected to other computers [2]. Computers are networked locally to share facilities such as printer and common software. Usually computers in an educational institute and in industry are networked. The emergence of workstation and networked computers has led to possibility of using such network for parallel programming.

It is widely recognized as cluster of workstation (COW) or network of workstation (NOW), which offers a very attractive alternative to expensive supercomputers and parallel computer systems for high performance computing. A cluster of workstations has number of significant and well-enumerated advantages over specifically designed multiprocessor system such as:

- Very high performance workstations and PCs are available at low cost.

- The latest processors can easily be incorporated into the system
- Existing software can be used or modified.

Use of workstations for parallel computing became interesting because network of workstation existed for general purpose computing and workstations were used for programming and computer related activities.. Software tools available for using network of workstation are Parallel Virtual Machine (PVM) and Message Passing Interface (MPI), which consist of message passing functions for data communication between computers. The communication network for workstations has commonly been an Ethernet typ, single wire to which all the computers attach as shown in Fig. 1.6.

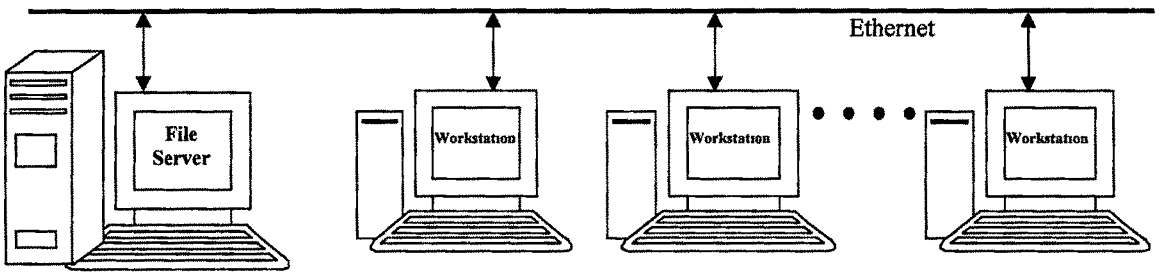


FIG. 1.6 ETHERNET TYPE SINGLE WIRE NETWORK

In ethernet type of connection, all transfers between a source and a destination are carried in "packet" along the single wire which carries the source address, the destination address and the data. Since each workstation is operating independently and could send messages at any time, the ethernet line may be needed by one computer while it is already being used to carry packets sent by other computers.

For parallel programming, workstations could be collected together into a dedicated parallel programming cluster. The computers forming this cluster need not have displays or keyboards but are linked with suitable communication medium. A single ethernet is the simplest method, but multiple Ethernet lines for overlapping connectivity can also be used.

1.7.2 Internet Computing

Internet is becoming common medium of communication widely used for browsing information, email, and online communication. Usually the CPU of computers connected through Internet is idle and utilized at most by about 10%. Computing power of all hosts connected through Internet can be harnessed for solving computationally intensive problems. The Internet can be the ultimate distributed system. The client can be any machine with a browser, and it can be located anywhere through a phone line or wireless satellite link can be established. Any client, once connected to the Web, can link to any server with a public URL. The server can be NT, Unix, or a mainframe running virtually any language. The key to the Internet is the small collection of Internet protocols like TCP/IP, HTTP, HTML, XML.

Client server approach can be used for distributed computing through computers connected through Internet. The problem is divided into subtasks and each task is distributed to various client machines and the management of various tasks i.e. initiation of subtasks on various computers, communication of data and synchronization of various subtasks is done by server. Various issues like networking, security, data communication, and load balancing among the computers depending on their processing capabilities and usage and synchronization among the tasks needs to be considered for Internet computing.

1.7.3 Grid Computing

Grid offers a way to solve Grand Challenge problems like Drug discovery, Financial modeling, Earthquake simulation, Climate / weather modeling. Grids offer a way of using the information technology resources optimally in an organization. They also offer a means to offer information technology as a utility bureau for commercial clients -- clients pay only for what they use, as with electricity or water.

Grid computing uses the Internet to borrow unused CPU cycles and storage from millions of systems across a worldwide network. This flexible, readily accessible pool can then be harnessed by anyone who needs it, much as power companies and their users share the electrical grid. Grid computing leans more to dedicated tasks, such as single large medical and engineering problems, rather than for general, everyday jobs. The computers on a Grid can be of many different OS and hardware platforms.

Grid computing is made up of computational and data intensive problems. The computational aspect focuses on reducing execution time of applications that require large amounts of computer processing cycles. Data intensive problems require large scale data management methods to transfer the data needed for solving the problem to the machine assigned to solve it. Data intensive applications such as High Energy Physics and Bioinformatics require both computational and data management solutions to be present in Grid computing solutions. Depending on function grids can be classified as data grid or computational grid. Grid computing requires the use of software that can divide and farm out pieces of a program to as many as several thousand computers.

Grid computing can be thought of as distributed and large-scale cluster computing and as a form of network-distributed parallel processing. It can be confined to the network of computer workstations within a corporation or it can be a public collaboration using the Internet.

1.7.4 Wireless Computing

Computer networking technology through wireless computing, promises users access to information anywhere and at any time. Wireless computing allows users with a portable computer, often called a personal digital assistant or a personal communicator, to have a wireless connection to information network [10].

The more dominant applications being supported are mail enabled applications and information services. In mail enabled applications the users carrying personal communicator are able to receive and send electronic mail. Information services can provide access to popular database and bulletin board type applications.

A general architecture for wireless computing includes: (a) the network of fixed hosts and (b) the mobile hosts (personal communicator). The mobile hosts communicate with specific fixed hosts called mobile support stations. These mobile support stations are fitted with a wireless interface to communicate with the mobile hosts. This technology further increases the opportunities for information exchange and management but it also brings its own special design challenges.

1.8 SCOPE AND OBJECTIVES OF PRESENT WORK

Solution of various problems of structural engineering like static, dynamic and nonlinear analysis of large size structure is computational intensive. Such problems have been generally attempted using either Direct Stiffness Approach or Finite Element Method of analysis. For better computational efficiency use of parallel and distributed processing has been successfully attempted by number of researchers.

From the literature review it was observed that most of the time consuming applications were implemented through supercomputer or network of transputers or dedicated distributed processing hardware. Use of these tools however, is not only costly but difficult to maintain and also requires theoretical background of operating system, networking and concepts of parallel processing. The structural engineers are usually interested in applications of technology rather than understanding various fields of computer engineering.

The main focus in the present work is on finding a cost effective processing method which may use network of computers available within organization but does not require any additional resources. For implementation of applications, user without any background of parallel processing should not face any difficulty. The objective is also to investigate various applications like Static and Dynamic analyses of Skeletal and Continuum structures, Nonlinear analysis of plates, analysis of Laminated Composites, Optimum design of trusses using Genetic Algorithm etc. The main aim is to critically examine the various aspects of distributed implementation like suitability of algorithm, user friendliness of environment, resources required, computational efficiency related issues, etc.

To achieve above objectives computers connected through local area network (LAN) are used as hardware resources. For implementation of applications, WebDedip environment developed using JAVA technology, based on client server approach is critically examined. WebDedip helps in configuration of application, distribution of data, synchronization of various processes and profiling the entire application. The present work includes following applications, which are developed in C++ language.

1. **Static analysis of skeletal structure using direct stiffness method:** In this application static analysis of microwave tower is included. The structure is divided into number of substructures and processing of each substructure is distributed to different computers.

2. **Static finite element analysis of plane stress and plate bending problems:** Various problems of plane stress analysis like deep beam, beam subjected to pure bending, plate with hole subjected to uniform tensile force using triangular and quadrilateral elements is discussed. Also plate bending problems like annular plate and skew plate subjected to distributed load are included. For distributed processing, finite element mesh is divided into number of parts and each part is assigned to a separate computer. Computational efficiency considering processing time and communication time is studied with different number of computers.

3. **Dynamic analysis of plane frame structures:** Static and dynamic condensation methods used in substructure technique are studied. Example of dynamic analysis of regular two-dimensional rigid frame is covered. Subsequently one large size problem is solved using different number of computers and computational efficiency is examined.

4. **Nonlinear analysis of plate:** Two approaches for distributed nonlinear analysis are discussed. In first approach calculation of geometric stiffness matrix, initial stress stiffness matrix and residual force vector of entire structure is distributed over number of computers and in second approach calculation of tangential stiffness matrix and load vector of substructure is distributed over number of computers. An example of clamped square plate subjected to uniformly distributed load is included and results are compared.

5. **Finite element analysis of laminated composites:** Higher order shear deformation theory for analysis of laminated composite is covered. For finite element implementation eight nodded isoparametric element with six degrees of freedom at each node is considered. The finite element mesh is divided into

substructures and each substructure is assigned to a different computer. For study, an example of four ply laminated simply supported square plate subjected to sinusoidal load, is considered.

6. **Training of artificial neural network for design of columns:** Artificial neural network can be used in problems where direct relationships between parameters do not exist. As neural network is based on learning function of brain, it can be trained through sets of input and desired output. The training of neural network using large number of data set can be made faster using distributed training. An example of training neural network for design of rectangular column subjected to biaxial bending is discussed to examine computational efficiency obtained through distributed processing.

7. **Distributed genetic algorithm for optimization of structures:** For optimization of large structure Genetic Algorithm based on survival of fittest concept, in which evaluation of fitness function is time consuming, is used. For better computation efficiency, fitness function evaluation of population is distributed on different computers. Application of distributed Genetic Algorithm is discussed through examples of plane and space trusses.

1.9 ORGANIZATION OF THESIS

The work carried out in present work is covered in various chapters. The organization of thesis is as under:

After brief introduction, factors responsible for parallel and distributed processing are discussed in **Chapter 1**. It also covers algorithms for parallel structural analysis and latest trends in computing. Finally objectives and scope of present work are enumerated.

Chapter 2 covers the literature review related to applications of parallel processing in structural engineering. It covers literature about general aspects of parallel processing, static, dynamic and nonlinear analysis of skeletal and continuum structures using finite element method, neural networks and genetic algorithms. It reviews type of hardware used, problem solved, algorithm and computational strategy used and results obtained in terms of computationally efficiency.

Various issues related to parallel and distributed processing like hardware, software, operating systems etc. are covered in **Chapter 3**. Terminology related to performance evaluation is discussed.

Chapter 4 discusses about WebDedip environment used in present work for implementation of distributed processing in various applications. It covers general facilities, various components, installation and application development using WebDedip.

Application of distributed processing in matrix analysis of skeletal structure using direct stiffness method is covered in **Chapter 5**. Use of substructure technique for implementing distributed processing is discussed. Feasibility of WebDedip environment over LAN is evaluated.

Distributed finite element analysis considering plane stress and plate bending problems is covered in **Chapter 6**. Implementation over different number of computers using substructure technique is illustrated and after comparing time spent in computation and communication, speedup is presented.

Chapter 7 covers distributed dynamic analysis of plane frames. After discussion of static condensation and dynamic condensation method used for substructure technique, various examples of regular plane frames with different number of substructures are covered to understand difference between static and dynamic condensation. Finally a large size example is implemented with different number of computers to study speedup.

Training of artificial neural network in WebDedip environment is the subject matter of **Chapter 8**. Counter propagation learning algorithm is discussed and its application to design of rectangular column subjected to biaxial bending is covered. More than 7000 data sets are used for training of neural network, which is distributed over different number of computers to study the computational efficiency.

Use of Internet technology for application of distributed processing in structural engineering is discussed in **Chapter 9**. One application for utilizing software on remote computer and other application for distributed finite element analysis using computers connected through Internet are studied.

Chapter 10 is devoted to use of genetic algorithm for optimization of structure. After introduction of genetic algorithm, steps followed for the same are described. Feasibility of distributed implementation is explored through examples of plane and space trusses.

Distributed nonlinear analysis of plate using finite element method is discussed in **Chapter 11**. It covers steps for nonlinear finite element analysis and various approaches for distributed implementation. Distribution of various parts of tangential stiffness matrix and residual force vector over number of computers and substructure approach are covered and compared.

Chapter 12 is devoted to distributed analysis of laminated composite plate. Higher order shear deformation theory and its implementation in finite element method are discussed. Substructure technique is employed for distributed computing over network of computers. Speedup results are included after implementing over different number of computers.

Finally **Chapter 13** covers summary and conclusions of the present work. After highlighting the contributions, scope for future work is also included.