Hypercube and Its Generalization

Dissertation submitted to
Jawaharlal Nehru University
in partial fulfillment of the requirements for the award of the degree of

MASTER OF TECHNOLOGY
in
COMPUTER SCIENCE AND TECHNOLOGY

By

VINAY KUMAR

Under the supervision of
PROF. C.P. KATTI

JNU

SCHOOL OF COMPUTER AND SYSTEMS SCIENCES
JAWAHARLAL NEHRU UNIVERSITY
NEW DELHI-110067, INDIA

JULY 2007

जवाहरलाल नेहरू विश्वविद्यालय

JAWAHARLAL NEHRU UNIVERSITY
School of Computer & Systems Sciences
NEW DELHI- 110067, INDIA

<u>CERTIFICATE</u>

This is certify that the dissertation entitled "***Hypercube and Its Generalization***", being submitted by Mr. Vinay Kumar to the **School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi** in partial fulfillment of the requirement for the award of the degree of **Master of Technology** in **Computer Science and Technology**, is a record of original work done by him under the supervision of Prof. C.P. Katti. This work has not been submitted in part or full to any other University or Institution for the award of any degree or diploma.

vinay kumar.
Vinay Kumar
(Student)

[illegible] 24/12
Prof Parimala N.
Dean
School of Computer & Systems Sciences
JAWAHARLAL NEHRU UNIVERSITY
NEW DELHI-110067
(Dean, SC & SS, JNU, New Delhi-67)

[illegible]
Prof. C.P. Katti
(Supervisor)

For

My Parents

Who taught me A B C in childhood

that enabled me to write a few words today...

Acknowledgement

A journey is easier when you travel together. Interdependence is certainly more valuable than independence. This thesis is the result of one year of work, whereby I have been accompanied and supported by many people and blessed by almighty God. It is a pleasant aspect that I have now the opportunity to express my gratitude for all of them.

The first person I would like to thank is my honorable supervisor, Prof. C. P. Katti for his valuable guidance in completing this dissertation successfully. His constant motivation and integral view on research has made a deep impression on me. He could not even realize how much I have learnt from him. I am really privileged that I have come to get know Prof. Katti in my life. Working under his supervision has been an immense learning experience for me.

I am thankful to Prof S. Balasundaram for providing a congenial environment and lab facilities in the school.

For the non-scientific side of my thesis, I particularly want to thank for my parents who formed part of my vision and taught me the good things that really matter in life. I would like to share this moment of happiness with them. I am extremely grateful to my other family members who have always boosted my morale in carrying out my work.

The chain of my gratitude would be definitely incomplete if I would forget to thank the first link of this chain, i.e. friends aiding me all the time in all manner they could.

Vinay Kumar

Contents

List of Figures

List of Tables and Graphs

Abstract

"Many hands make light work"

- John Heywood

A parallel system provides an unambiguous and high level framework for solving a large problem parallely by multiple processors. Many scientific and engineering problems are in the form of system of linear equations. However in the real world, system of linear equations is often quite large. As we know system of linear equations can be written as $Ax = b$, where A is an $N \times N$ matrix. A most common technique for solving this is Gauss Elimination technique. For solving this system of equations time complexity is O (N^3). But If matrix A is already in a triangular form, then solving this system takes time O (N^2).

A parallel version on hypercube for the solution of $Ax = b$ is given by [SA, 86]. Hypercube architecture suffers from a disadvantage that its nodes are in 2^k forms, where k is an integer. We have extended the result of [SA, 86] by considering the solution of $Ax = b$ on a supercube, which does not have restriction on number of nodes and reduces to hypercube, when number of nodes are 2^k. We also consider the solution of $Ax = b$ on a pyramid architecture. We finally compare the performance evaluation of each architecture.

Chapter 1

Introduction

1.1 Parallel Computing

Parallel computing is the simultaneous execution of the same task on multiple processors in order to obtain results faster. The idea is based on the fact that the process of solving a problem usually can be divided into smaller tasks, which may be carried out simultaneously with some coordination.

Traditionally, software has been written for *serial computation*:

- For running the problem having a single Central Processing Unit (CPU).
- Breaking up a large problem into smaller tasks.
- Executing tasks one by one.

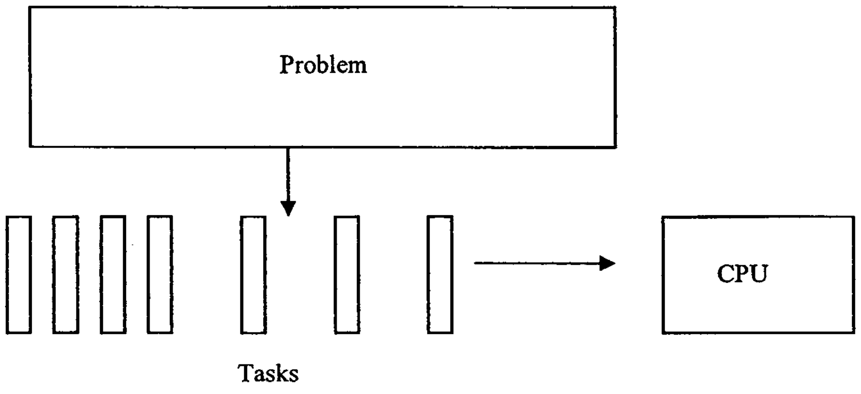


Figure 1(a): *Serial Computation of a problem*

But parallel computing is the simultaneous use of multiple compute sources to solve a computational problem.

- For running the problem using more than one CPUs.
- Breaking up a large problem into smaller tasks.
- Each task is further broken down to a series of instructions.
- Instructions from each part execute simultaneously on different CPUs.

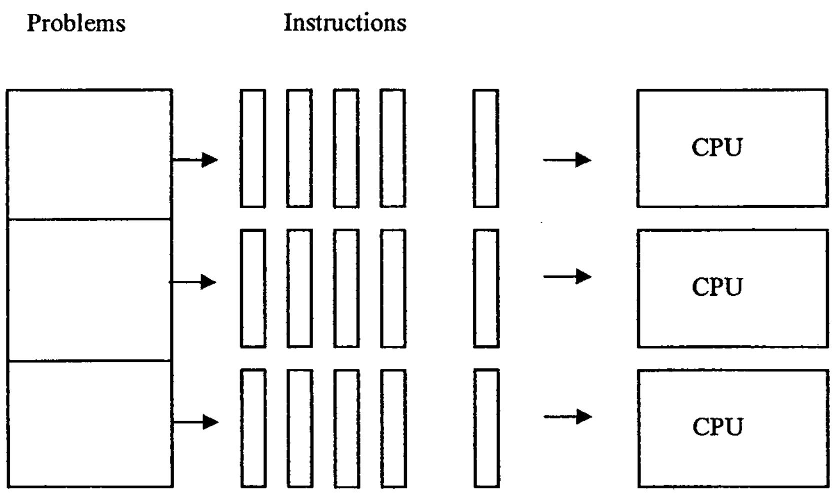


Figure 1(b): *Parallel Computation of a problem*

1.2 Historical View of Parallel Computing

Idea of parallelism is comes when Charles Babbage designed *the Analytic Engine.* In 1842, General L. F. Menabrea wrote a paper on Analytical Engine invented by Charles Babbage, He writes

"*When a long series of identical computations is to be performed, such as those required for the formation of numerical tables, the machine can be brought into play so as to give several results at the same time, which will greatly abridge the whole amount of the processes.*"

That indicates about idea of parallelism. The study of computer architecture involves both hardware organization and programming/software requirements. From the hardware implementation point of view, the abstract machine is organized with CPUs, caches, buses, pipeline, physical memory etc. Therefore architecture covers both instruction set architecture and machine implementation organization. [KH, 93]

1.2.1 Von Neumann Architecture

Over the past four decades, computer has gone through evolution rather than revolution changes. For over 40 years, virtually all computers have followed a common machine model known as the *Von Neumann Computer*, named after the Hungarian mathematician John Von Neumann. A Von Neumann computer uses the stored-program concept. The CPU executes a stored program that specifies a sequence of read and writes operations on the memory

Basic design features includes:

- Both program and data instructions are store in memory.
- Program instructions are coded data which tell the computer to do something.
- Data is used by the program.
- A central processing unit (CPU) gets instructions and/or data from memory, decodes the instructions and then *sequentially* performs them.

A basic *Von Neumann Architecture* is shown in figure.

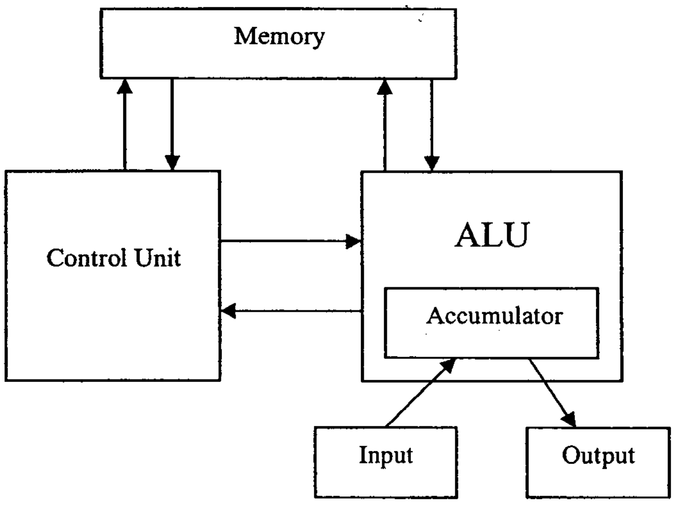


Figure 1(c): *Von Neumann Architecture*

This architecture built as a sequential machine executing scalar data because of this, these machines was slow. Later look ahead techniques were developed to prefetch instructions in order to develop I/E (instruction fetch / decode and execute) operations and to enable functional parallelism. Functional parallelism was supported by two approaches: one is to use multiple functional units simultaneously and the other is to practice pipelining at various processing levels.

1.2.2 Flynn's Classification

- <u>**SISD**</u> – single instruction / single data stream. This is the conventional serial Von Neumann computer in which there is one stream of instructions (and consequently only one processing unit) and each arithmetic instruction initiates one arithmetic operation, leading to a single data stream of logically related arguments and results. It is irrelevant whether pipelining is used to speed up the processing or not. A machine of this call is sometimes referred to as a scalar computer.
- <u>**SIMD**</u> – single instruction / multiple data stream. This is a computer that retains a single stream of instructions but has vector instructions that initiate many operations. Each element of the vector is regarded as a member of a separate data stream. Hence, there are multiple data streams. This classification includes all machines with vector instructions and machines belonging to this class are often called vector computers.
- <u>**MISD**</u> – multiple instruction stream / single data stream. This is essentially a void class because it implies that several instructions are operating on the same data simultaneously. We will not consider this class of machines.
- <u>**MIMD**</u> – multiple instruction stream / multiple data stream. Multiple instructions streams imply the existence of several instruction processing units and therefore, necessarily, several data streams. This class is quite general and includes all forms of multiprocessor configurations from linked mainframes (LANS and WANS) to a large arrays of microprocessors.

One can see that only two of the categories described above are of interest as far as parallel processing is concerned – i.e. the SIMD and MIMD machines.

1.3 Why Parallel Computing?

For decades computer architects have incorporated parallelism into various levels of hardware in order to maximize the performance of computers. To getting the extremely high speed demanded by modern science, architecture's must now incorporate parallelism at the uppermost level of the system. Todays, the fastest computers in the world use high level parallelism concept. These computers are leading to new scientific discoveries.

Traditional serial computers are characterized by the occurrence of a single locus of control that tells about the next instruction to be executed. During the execution of each instruction data is operated. That is fetched from the global memory one at a time. So, only one instruction is execute at a time. In this processing, total computation is slow by the speed of the memory access and the speed of the input-output devices. A number of methods have been developed for alleviating these bottlenecks, cache memories and pipelining are some examples.

1.3.1 Limitations of Serial Computing

Both physical and practical reasons pose major constraints to faster serial computers:

- **Transmission Speed**: How fast data can transmit through hardware? The speed of a serial computer is directly dependent upon answer of this question. Absolute limits are the speed of light is 30 cm/ns, whenever the transmission limit of copper wire is 9 cm/ns. Increasing speeds necessitate growing proximity of processing elements.
- **Limits to Miniaturization**: Increasing the number of transistors in place of using chip allowing by processor technology. However, even with molecular or atomic-level components, a limit will be reached on how miniature components can be.
- **Economic Limitations**: To make faster a single processor is an expensive procedure. Using a larger number of moderately fast commodity processors to achieve the same performance is less expensive then above procedure.

Parallel computing is currently an area of powerful research activity. This activity is motivated by a multiplicity of factors. These have always been a need for the solution of very large computational problems, but it is only recently that technological advances have raised the possibilities of massive parallel computation and have made the answer of such problems possible. Furthermore, the availability of powerful parallel computers is generating interest in the new types of problems. They problems were not addressed in the earlier period.

1.3.2 Benefits of using the Parallel Computing

The prime reasons for using parallel computing

- Minimize time
- Solve bigger problems
- Give concurrency
- Taking help of non-local resources by using available compute resources on a wide area network, or even the Internet.
- Cost savings by using multiple low cost computing resources. This cost can not save in supercomputing.

1.4 Communication Aspect of Parallel Computing

The development of parallel computing gives a new aspect in computer science. Parallel computers can give better performance at lower cost than machines which use especially good processors. In many parallel algorithms, the time spent for inter-processor communication is nothing but it is a fraction of the total time that is required for solving a problem. In that case algorithm experiences a considerable *communication delay.* In definition of communication delay

$$T_{CD} = \frac{T_{total}}{T_{comp}}$$

Where T_{total} is the total time required by the algorithm to solve the given problem, and T_{comp} is the corresponding time that can be attributed just to computation that is the time that would be required if all communication were instantaneous. T_{CD} is communication delay.

Distributed computing system can be looked as a network of processors. That is connected by communication links. For storing some data and results of computation, each processor uses its own local memory. Each processor requires the ability of exchanging information with other processors. This information is in group of bits that is called packets. For this exchanging information, use the communication link of the network. The size of the packet can be widely varying.

A shared memory can also be viewed as a communication network, so when a processor send the information to any other processor. It can be store information in shared memory. [BT, 89]

Communication delays can be separated into four parts:

- **<u>Communication processing Time</u>**: It is time that required to preparing the information for transmission. Some examples of preparing the information are making the information in packets, affix addressing and control information to the packets, selecting a communication link on which to send each packet and moving the packet to appropriate buffers.
- **<u>Queuing Time</u>**: When information is assembled into packets for sending on some communication links, it must wait in a queue for the start of transmission. This wait is called *queuing time*. There are many reasons for waiting e.g. the link may be temporarily unavailable because other information packets are using it or scheduled to use it ahead of the given packet, another reason is that by limitation of needed resources it may be necessary to delay the transmissions of packets. No proper buffer space at the destination processor is the one type of limitation.
- **<u>Propagation Time</u>**: A time is requiring for between the end of transmission of last bit of the packet at the transmitting processor and the reception of the last bit of the packet at the receiving processor. That time is called *propagation time.*
- **<u>Transmission Time</u>**: It is the time that is required for transmission of all the bits of the packets.

One or more of above given times may be negligible. It's depending on system and algorithms.

1.5 Parameters of Parallel Algorithms

Execution time complexity and space time complexity are two major parameter for evaluating a any sequential algorithms. The quality of good algorithm is it has a small running time and uses as little space as possible. But in a parallel algorithm, three criteria for evaluating a parallel algorithm. There are running time, number of processors, and cost. [Q, 94]

They are define as follows

- **Running time:** Since speeding up solution of a problem is the main reason for building parallel computers, running time is an important measure for evaluating a parallel algorithm. It is time taken by the algorithm to solve a problem on a parallel computer. A parallel algorithm is made up of two kinds of steps. One is computation step and other is communication step.

  In the computation step, performing a arithmetic or logic operation by a processor. In the communication step data is transformed between the processors via the shared memory or through the interconnection network.

  Running time = computation time + communication time

  Thus the running time of a parallel algorithm is sum of time spend in computation steps and time spend in communication steps. The running time also depends on the machine on which algorithm execute.
- **Number of processors**: How many processors are used in a parallel algorithm? That is other criteria for evaluating a parallel algorithm.
- **Cost:** The cost of a parallel algorithm is multiplication of the running time of the parallel algorithm and number of processors used in parallel algorithms

  $$Cost = running\ time \times no.\ of processors$$

To be able to review the merits of a parallel algorithm, one needs to count the number of time unit steps needed. For this it is better to introduce the idealized notion that during such a time unit step exactly one arithmetical operation can be carried out in the parallel form.

1.6 Applications of Parallel Computing

In natural world, many complex, interrelated events happening at the same time by parallel computing. Some example of parallel computing is planetary and galactic orbits, automobile assembly line, rush hour traffic, seismic processing, geophysical modeling, daily operations within a business and other activities. [KH, 93]

The big challenging applications areas of parallel computing are

- The magnetic recording industry use of computers. This industry revise magneto static exchange interactions for reducing the noise in metallic thin films to coat high density discs.
- Supercomputing MPP systems can ocean modeling much accurately than other techniques. For analyzing the complex chemical and dynamical mechanism in ocean depletion research need of computers is necessary. Also in ozone modeling, use of parallelism widely.
- Rational drug design is being aided in the study for a acquired Immune deficiency and cure cancer syndrome. In this high performance computing is used.
- A civil transport aircraft are being aided by computational fluid dynamics running on super computers. A engine models, which made through chemical kinetics calculation is better for fuel combustion.
- Many areas as -- pollution reduction through computational modeling, image processing, design of protein structure by computational biologist and digital anatomy in real time medical diagnosis follow the concept of parallel computing.

1.7 Outline of Dissertation

The rest of this dissertation is organized as follows. In Chapter *2,* some necessary definitions, interconnection networks and fundamental properties of interconnection networks introduced. In Chapter *3,* describe hypercube and supercube architecture. In this chapter we apply Gauss elimination technique on hypercube and supercube. In Chapter *4,* explain pyramid architecture and effect of length on a pyramid architecture. Further in this chapter apply Gauss elimination technique on pyramid architecture and effect on this technique when pyramid architecture with length and without length. Finally in this chapter give the comparison between supercube and pyramid architecture. In Chapter *5,* give the conclusion of whole dissertation.

Chapter 2
Background

The topology of an interconnection network can be either static or dynamic. *Static networks* are formed of point to point direct connections which will not change during program execution. *Dynamic networks* are implemented with switched channels, which are dynamically configured to match the communication demand in user programs.
Static networks are used for fixed connections among subsystems of a centralized system or multiple computing nodes of a distributed system. Dynamic networks include buses, crossbar switches and multistage switches which are often used in shared memory multiprocessors. Here we focus on static interconnection networks. [AAGV, 03]

2.1 Evaluation Criteria for a Interconnection Network

There are several parameters often used to estimate the complexity, communication efficiency and cost of a network. In general, a network is represented by a graph of a finite number of nodes connected by directed or undirected edges. Here number of nodes like processors and edges are like communication links. The number of nodes in the graph is called *network size*. [AAGV, 03]

- **Node degree** - The number of edges (channels) incident on a node is called the *node degree*. In case of unidirectional channels, the number of edges into a node is the *in-degree* and that out of a node is the *out-degree*. For unidirectional channels the degree of a node is the sum of two. The node degree reflects the number of I/O ports required per node and thus the cost of a node. Therefore for reducing cost the node degree should be kept a constant, as small as possible. A constant degree is very much desired to achieve modularity in building blocks for scalable systems.

- **Diameter** - In any connected network there is always a shortest path between any two nodes. *Diameter* is the maximum distance shortest path between any two nodes. The network diameter tells the maximum number of distinct hops between any two nodes. Therefore, the network diameter should be kept as small as possible from the communication point of view.
- **Connectivity** - The *connectivity* of a network is a calculate multiplicity of paths between any two processing nodes in the connected network. A desirable things is network should be with high connectivity. It lowers the contention for communication resources. Number of nodes that are require to break a network into two disconnected networks is called the *arc connectivity* of the network.
- **Bisection width** - When a given network is cut into two equal halves, the minimum number of edges along the cut is called channel *bisection width b. Channel width* is the number of bits that can be communicated simultaneously over a link of connecting two nodes. Number of physical wires in each communication link is equal to channel width. The high rate at which a physical wire can deliver bits is called the *channel rate.* The peak rate at which data can be communicated between the ends of a communication link is called *channel bandwidth.*

$$\textit{Channel bandwidth} = \textit{Channel rate} \times \textit{Channel Width}$$
$$B = b\,w$$

  When B is fixed, the channel width (in bits) w = B / b. Thus the bisection bandwidth provides a good indicator of the maximum communication bandwidth along the bisection of a network.
- **Cost** - Cost of the network can be defined in terms of how many links are required by the network.

The performance of an interconnection network is affected by the following factors:

- **Functionality** - This refers to how the network supports data routing, interrupt handling, synchronization, request message combining and coherence.

- **Network latency** - This refers to the worst case time delay for a unit message to be transferred through the network.
- **Bandwidth** - This refers to the maximum data transfer rate in terms of Mbytes transmitted through the network.
- **Hardware complexity** - This refers to the implementation cost such as those for wires, switches, connectors, arbitration and interface logic.
- **Scalability** - This refers to the ability of the network to be modularly expandable with a scalable performance with increasing machine recourse.

2.2 Types of Interconnection Networks

[AAGV, 03], [Q, 94]

2.2.1 Completely-Connected Network

In a *completely-connected network*, each processor has a direct communication link to every other processor in the network. A completely-connected network of five processors shown in figure

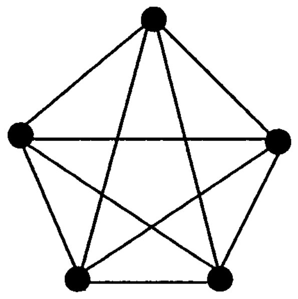


Figure 2(a): *Completely- connected network*

In this interconnection network, a processor can send message to another processor in a single step, because each pair of processor has a direct communication link. The diameter of a completely-connected network is *1*. The maximum number of links per processor is *p-1*, where *p* is the number of processors. If in the network, any communication link not working properly then whole network is not be blocked.

2.2.2 Linear Array and Ring

A simple way to connect processors is illustrated in figure (i). Each processor in this network (except the processors at the ends) has a direct communication link to two other processors. Such an interconnection network is called a *linear array.* A wraparound connection is often provided between the processors at the ends. A linear array with a wraparound connection is referred to as a *Ring*. Figure (ii) shows a ring of five processors.

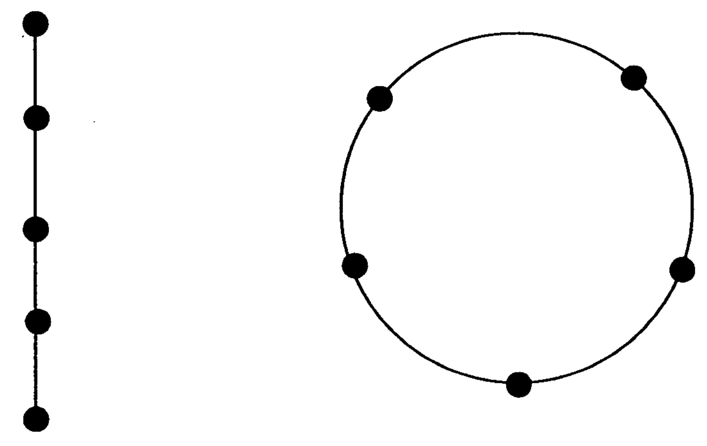


Figure 2(b): *Linear array and Ring Network*

One way of communicating a message between processors is by repeatedly passing it to the processor immediately to the right (or left, depending on which direction yields a shorter path) until it reaches its destination. The diameter of a liner array and a ring is *(p-1)* and $\left\lfloor \frac{p}{2} \right\rfloor$ respectively. Where *p* is number of processors in network. The maximum number of links per processor is *2*.

2.2.3 Mesh Network

In *Mesh network,* processors are arranged into *q*-dimensional array. Interior processors communicate with *2q* other processors. In a two-dimensional mesh, each processor has a direct communication link connecting it to four other processors. Figure (i) show a two-dimensional mesh of nine processors. If both dimensions of the mesh contain an equal number of processors, then it is called a *square mesh*; otherwise it is called a *rectangular mesh.*

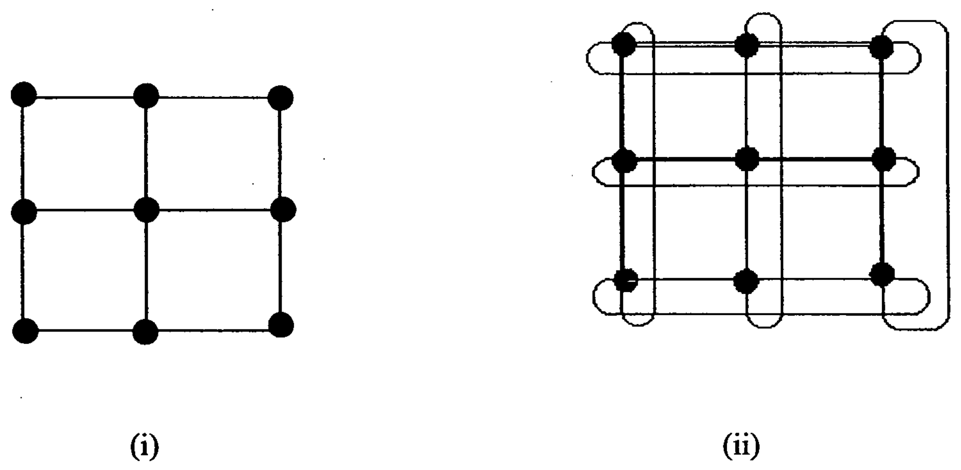


Figure 2(c): *Mesh Network*

Often, the processors at the periphery are connected by wraparound connections. Such a mesh is called a *Wraparound mesh* or *Tours* as in Figure (ii). A message from one processor to another can be routed in the mesh by first sending it long one dimension and then along the other dimension until it reaches its destination. This model allows wraparound connections between processors on the edge of mesh. Many commercial available parallel computers are based on the mesh network. When mesh has no wrap-around connections, the diameter of a q dimensional mesh with k^q nodes is $q\ (k\text{-}1)$. The maximum number of links per processor is $2q$.

2.2.4 Tree Network

A *tree network* is one in which there is only one path between any pair of processors. Linear array network is special case of tree networks. A type of tree network, in which $(2^k\text{-}1)$ processors are arranged into a complete binary tree of depth k, called *binary tree network.* Figure shows networks based on complete binary tree with 7 processors. To route a message in a tree, the source processor sends the message up the tree until it reaches the destination processors. Then the message is sent down the tree toward the destination processor.

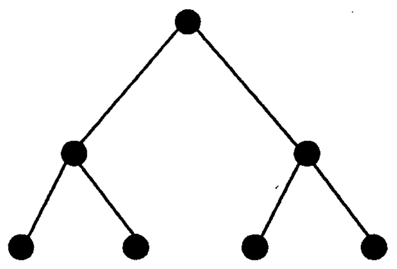


Figure 2(d): *Binary Tree Network*

A binary tree network has diameter *2(k-1)*. The maximum number of links per processor is *3,* two for its children and one for its father.

Tree networks suffer from a communication bottleneck at higher levels of the tree. For example, when many processors in the left subtree of a node communicate with processors in the right subtree, the root mode has to handle all the messages. This problem can be alleviated by increasing the number of communication links between processors that are closer to the root. This network is called a *fat tree.*

2.3 Communication Cost in Parallel Computing

The network topology, programming models semantics, associated software protocols data handling and routing are the features on which communication cost is depend. The time taken to communicate a message between two nodes in a network is the sum of the time to prepare a message for transmission and the message to traverse the network to its destination. [AAGV, 03]

The basic parameters that determine the communication latency are as follows:

- **Startup time** – Time, require for handling a message at sending node and also at receiving node, is called *startup time*. It includes the time to prepare the message, to execute the routing algorithm, and to creating interface between router and nodes.

By preparing a message mines that adding header, trailers and error correction in a message. Startup time is denoted by t_s.

- **Per-hop time** – In two directly connected nodes, time require for traveling between these nodes by the header of a message is called *per-hop time.* It is also known as *node latency.* Node latency is directly related to the latency within the routing switch. It is denoted by t_h.
- **Per–word transfer time** – It is time require for each word to traverse the link. If the channel bandwidth is *r* words per second, then each word takes time (1 / *r)* to traverse the link. This time is called *per word transfer time.* It is denoted by τ.

  So $\tau = 1/r$

There are some routing techniques that have been used in parallel computers are following.

2.3.1 Store and forward Routing

When a message is traversing a path with multiple links, each middle node on the path receives a message and store. Before receiving and store next message it must be forward previous message.

Suppose that a message of size N is being transmitted through stored and forward network and it traverses total l links. At each link, the message incurs a cost t_h for the header and $N\tau$ for the rest of the message to traverse the link. Since total links are l,

So total time is $(t_h + N\tau)\,l$. Therefore the total communication cost for a message of size τ words to traverse l communication links is

$$t_c = t_s + (t_h + N\tau)\,l$$

In modem parallel computers, the per hop time t_h is very small. So we can take communication cost is

$$t_c = t_s + N\tau l$$

2.3.2 Packet Routing

Store and forward routing is not doing proper use of communication resources. A message is sent from one node to the next node only after the entire message has been received. If a message is broken into some equal parts before it is sent, middle node waits for only one parts of the original message to arrive before passing it on. This approach is good for better utilization of communication resources. Clearly packet routing reduced the communication time.

Thus this routing technique offers lower overhead from packet loss and better error correction capability. One other advantage of this technique is free to choosing different paths for packet. But the overhead involved is that each packet must carry routing and sequencing information.

2.3.3 Cut-Through Routing

If we enforce that all the packets should take the same path, we can eliminate the overhead of transmitting routing information with each packet as in-sequence delivery and sequencing information. If we add error information at message level not at packet level, then the overhead can be reduced that is incurred with error detection and correction. [AAGV, 03]

In Cut-Through routing, a message is broken into fixed size units called flow control digits or flits. They flits are much smaller in size than packets because they do not contain the overhead of packets. To establish a connection, a tracer is first sent. Once the connection is established, the flits are sent continuously to the same path. Middle nodes do not wait for the entire message to arrive before forwarding. As soon as flit is received at an intermediate node, it is transferred to the next node. So we not need of large buffer to store the entire message as in case of store and forward technique. This technique wants less memory and memory bandwidth at middle nodes.

Consider a message is traversing l links in the network and t_h is the per hop time, then

lt_h is the time taken by header of the message to reach the destination. If the message is N words long, then the entire message arrives in time $N\tau$ after the arrival of the header of the message. Therefore the total communication time in cut trough routing

$$t_c = t_s + lt_h + N\tau$$

This time is less from time taken in store and forward routing.

2.3.4 A Simplified Cost Model

By previous section, the communication cost of a message between two nodes l hopes away is

$$t_c = t_s + lt_h + N\tau$$

In this equation, for optimize the cost of message transfer, we would need to

- **Communicate in bulk** – In place of sending small messages and give up the startup cost t_s for each, it will be better to aggregate small messages into single large message and amortize the startup latency across a larger message. Because of clusters and message passing machines, the value of startup time is much bigger than per-hop time and per-word transfer time.
- **Minimize the volume of data** - To minimize the overhead paid by means of per-word transfer time τ, it is better to reduce the volume of data communicated as much as possible.
- **Minimize distance of data transfer** – It is good that decrease the number of hops that a message must traverse.

The per-hop time t_h is normally dominated either by per word component ($N\tau$) for large message or by the startup latency (t_s) for small messages. Since the maximum number of hops (l) in most networks is relatively small, so in that per-hop time can be neglect per with small loss in accuracy.

So a simplified cost model in which transferring a message between two nodes takes time is

$$t_c = t_s + N\tau$$

Most recent parallel computers utilize cut through routing. The size of the flit may depend upon network parameters. If flit size is too small for a given bandwidth, required flit becomes very large. This requires routers that can operate at a very high speed. On the contrary if the flit size becomes large internal buffer size increases and thus latency of the message. Flit size is generally kept from 4 bit to 32 bytes. In many parallel programming paradigms that rely predominantly on short messages, the latency of message is critical. For these, it is unfair for along message traversing a link to hold up a short message. Such situation can be addressed using a new technique, which is called multilane cut-through routing. In multilane cut through routing, a single physical channel is break into a number of virtual channels.

Chapter 3

Gauss Elimination on Hypercube and Supercube architecture

3.1 Hypercube Architecture

The hypercube has been used extensively as a network topology for parallel computers. Hypercube has symmetry for nodes and links. It constructed recursively. Hypercube offers high data bandwidth and low message latency. In other advantage, it is a simple architecture, takes limited space. The first experimental hypercube was designed at Caltech in Collaboration with Jet Propulsion lab (1985).

TH-14683

A n-dimensional hypercube consists 2^n nodes, which is numbered by n-bit binary numbers from 0 to $2^n - 1$. A zero cube consist only one node. In recursive way to obtain an n cube, take two (n-1) cubes and link their corresponding nodes. In hypercube two nodes directly connected only if they are differ by exactly one bit. . A 3-dimensional hypercube is shown in figure. In this there are two 2-dimensional hypercube, which are dark. All three dimensional are show by arrow.

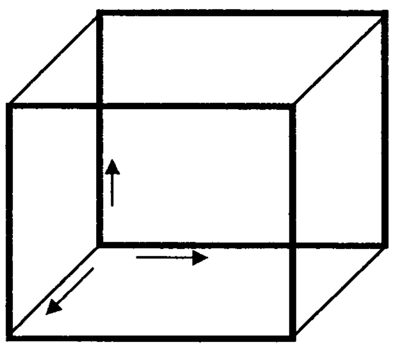


Figure 3(a): *3-dimensional Hypercube*

In any interconnected network, diameter of network and degree of nodes play the major role. A n- dimensional hypercube has diameter n and degree of nodes is n also. Hypercube gives balance between diameter and degree of node. Ring architecture with p-nodes has diameter $\left\lfloor \frac{p}{2} \right\rfloor$. But by means of connectivity, it is less expensive. In another way, a fully connected network has diameter *1*, but most expensive in terms of cost of connectivity. So point of balance factor, hypercube is most efficient architecture.

For identification of nodes of a hypercube, we use a particular notation, called *Gray codes.* Gray code generated by concept of recursion which is defined as follows:

$$G_1 = \{0, 1\}$$

$$G_n = \{0G_{n-1}, 1G^R_{n-1}\} \quad \text{where } n = 2, 3, 4 \ldots\ldots\ldots\ldots\ldots\ldots$$

G_n represents n-bit gray code, G^R_{n-1} represents sequence obtains from G_{n-1} by reversing order of G_{n-1}. Prefixing a zero to each element of G_{n-1} makes $0G_{n-1}$. This prefixing also true for *1*. Next Figure shows labeling the nodes of *3*-dimensional hypercube by Gray code.

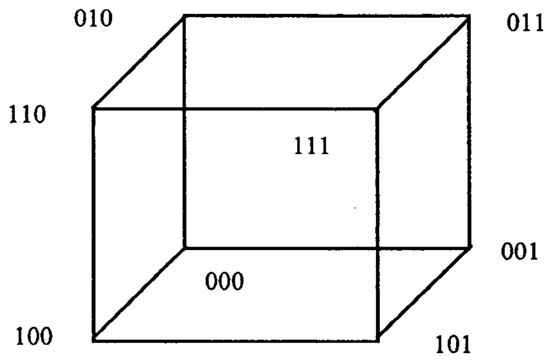


Figure: 3(b): *3-dimensional Hypercube with Gray code*

One other term using in hypercube structure is Hamming distance. Between two nodes X and Y, the number of bits that differ in node X and Y is called *Hamming distance.* We denote it by *HD* (X, Y). By definition of hypercube two nodes directly connected if they are differ by exactly one bit. So, two nodes are directly connected if they have hamming distance *1*.

3.1.1 Properties of Hypercube [SS, 85]

- An n-dimensional hypercube has diameter $\log_2 p$ where p is the total number of node.
- An n-dimensional hypercube is a connected and regular graph with degree n.
- An n-dimensional hypercube has 2^n nodes and $n\,2^{n-1}$ links.
- Total different ways of numbering of n-dimensional hypercube is $(2^n . n!)$.
- No cycles of odd length in an n-dimensional hypercube.

3.2 Communication in Hypercube

In parallel processing, one processor exchange data with other processor. Communication cost passing a message between two processors is

$$t_s + N\tau$$

Where N = size of the message, t_s = startup time and τ = per word transfer time.

All process is identical, so assume that communication cost is hypercube $(t_s + N\tau)$ between any two processors in hypercube architecture. Also assume that all links are bidirectional such that two directly connected nodes send data to each other simultaneously in time $(t_s + N\tau)$, whenever size of data is N.

In broadcasting, when one processor is send data to all other processes, called *one to all broadcasting*. The inverse operation of one to all broadcasting is *all to one broadcasting*, in which data combined from all processes at a single destination process. One to all broadcasting and all to one broadcasting are used in several parallel algorithms as *Matrix Vector multiplication*, *Gaussian elimination* and *Vector inner product.*

One to all broadcasting in hypercube architecture, communication start along a dimension and proceeds along other dimension in subsequent steps. Perform one to all broadcasting in a *3*-dimensional hypercube as following.

Let message is at node *0*. In first step node *0* send message to node *1*. In next step node *0* and *1* send message to node *2* and *3*. And so on.

So it takes total 3 steps for sending message process 0 to all other processes.

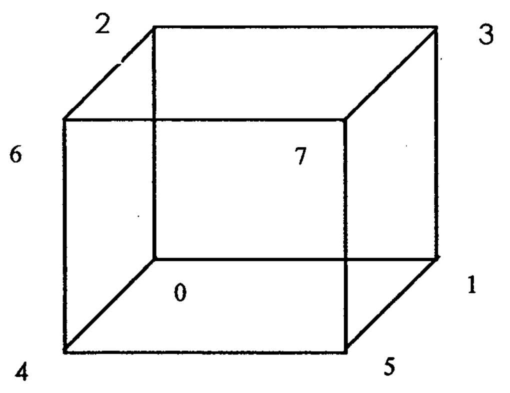


Figure 3(c): *3-dimensional Hypercube for broadcasting*

For n-dimensional hypercube need only n steps for performing one to all broadcasting. because message send from one dimension to next dimension need one step. In other way, can say that p processor hypercube needs only $\log_2 p$ steps. [AAGV, 03]

For analyzing cost of one-to-all broadcasting in hypercube, let hypercube has total p processors and message of size N words. At each step time cost is $(t_s + N\tau)$ and hypercube takes $\log_2 p$ steps. So total time require for one to all broadcasting in hypercube is

$$t_c = (t_s + N\tau)\log_2 p \qquad \text{.........3(a)}$$

In broadcasting, when all processor send different data to all other process. This operation is called *all to all broadcasting*. It is a generalization of one-to-all broadcasting. This broadcasting is used in matrix operations. The inverse operation of all to all broadcasting is *all to all reduction*.

In a *3*-dimensional hypercube, at every step processor exchange data, by which data size is double at every step. This *3*-dim hypercube takes 3 steps, but each step size of data is double.

For analyzing cost of all-to-all broadcasting in hypercube, let hypercube has total p processors and in starting each process has message of size N words. So total time require for all to all broadcasting in hypercube with p processors.

$$t_c = \sum_{k=1}^{\log p} [t_s + (2^{k-1} N) \tau]$$

$$t_c = t_s \log p + N(p\text{-}1)\ \tau \qquad \text{.........3(b)}$$

When a processor send a unique message of size N to every other node. This operation known as *one to all personalized communication.* It is clear that, it is not like one to all broadcasting. In one to all broadcasting duplication of message are allowed but in one to all personalized communication duplication of messages are not allowed. In this send a unique message to all other nodes. The inverse operation of all to all personalize communication is *concatenation.* In it a single node receives a unique message from every node

One to all personalized communication for hypercube at first step source code sends half message to other processor. And next step both processors send half messages to other processor. Similarly perform this process in subsequent steps. [AAGV, 03]

For analyzing cost of one to all personalized communication on hypercube, at every step message size is reduced. Let hypercube has p processor and source process send N size unique messages to all other nodes. So total time require for one to all personalized communication

$$t_c = \sum_{k=\log p}^{1} [t_s + (2^{k-1} N) \tau]$$

$$t_c = t_s \log p + N(p\text{-}1)\ \tau \qquad \text{.........3(c)}$$

3.2.1 Improving Communication in Hypercube

Previous communication method is not better performing when message is large. For the purpose of better utilization of communication network we split the N size message into smaller parts and send these parts through different paths.

Now measure communication time on a p processor hypercube, when a processor has large size message. Let size of the message is m. Now split N size message into p parts. Each message of size N/p *and* total number of message is p. Also total number of processor is p. Perform one to all personalized operation on hypercube. So the communication time require for (N/p) size messages. [From equation 3(c)]

$$t_c = t_s \log p + \frac{N}{p}\,(p\text{-}1)\,\tau \qquad \text{..........} 3(d)$$

Now all nodes have *(N/p)* size unique message perform the all to all broadcasting on hypercube. It is clear that each node has *(N /p)* size message, so need of communication time is

$$t_c = t_s \log p + \frac{N}{p}\,(p\text{-}1)\,\tau \qquad \text{...........} 3(e)$$

For one to all broadcasting on hypercube, add both equation 3(d) and 3(e)

$$t_c = 2\,[t_s \log p + \frac{N}{p}\,(p\text{-}1)\tau\,]$$

$$t_c \approx 2[\,t_s \log p + N\,\tau\,] \qquad \text{...........} 3(f)$$

This equation gives communication time for one to all broadcasting when a message is split into parts. Equation (3.a) gives communication time for one to all broadcasting when message is not split into parts. Compare these two equations, we find startup time t_s is doubled but per word transfer time τ is reduced by $\frac{\log_2 p}{2}$ factor.

3.2.2 Gauss Elimination on Hypercube

Now we perform parallelization of gauss elimination for solving system of linear equations. A system of *N* linear equations with *N* variables

$$\left.\begin{array}{l} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \text{-----------------} + a_{1N}x_N = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \text{-----------------} + a_{2N}x_N = b_2 \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ a_{N1}x_1 + a_{N2}x_2 + a_{N3}x_3 + \cdots\cdots\cdots\cdots + a_{NN}x_N = b_N \end{array}\right\} \text{......} 3(g)$$

It also expressed as *Ax* = *b*, where *A* is an *N* × *N* matrix and *x* and *b* are *N*-element vectors, which containing x_i's and b_i's.

When equation 3(g) is solves sequentially, it takes time complexity O (N^3) to solve this system of linear equations. If matrix A in upper triangular or lower triangular form, then for solving this system takes complexity O (N^2).

As we know, an $N \times N$ matrix A is called upper triangular if

$$i > j \Rightarrow a_{ij} = 0 \text{ in matrix } A.$$

Also an $N \times N$ matrix A is called lower triangular if

$$i < j \Rightarrow a_{ij} = 0 \text{ in matrix } A.$$

A Gaussian elimination for solving system of linear equation $Ax = b$ reduces the matrix A to an upper triangular matrix.

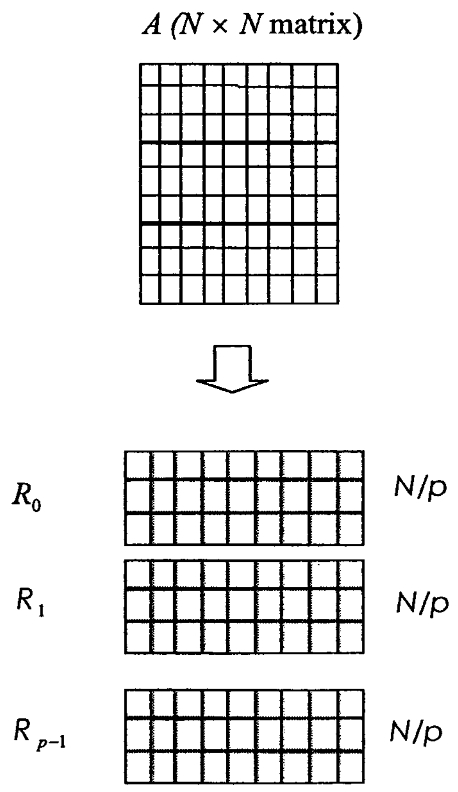


Figure 3(d): *Partition of matrix A in p blocks*

Let a hypercube with p processors (2^n) and processors are denoted by R_0, $R_1 \ldots R_{p-1}$. Matrix A is an $N \times N$ matrix. Assign to each of p processors, $\frac{N}{p}$ rows of A, where A is divided into blocks of $\frac{N}{p}$ rows each.

So Processor R_0 holds the rows from 1 through $\frac{N}{p}$.

Processor R_1 holds the rows from $\frac{N}{p}+1$ through $2\frac{N}{p}$.

In similar way Processor R_i holds rows from $i\,(\frac{N}{p}) + 1$ through $(i+1)\,\frac{N}{p}$.

Where $i = 0, 1, 2 \ldots . p-1$.

At the first step, 1st row of R_0 contain N words to broadcast to all processors and the first column is made zero. During the second step, 2nd row from R_o contain $(N - 1)$ words to broadcast to all processors. Finally we get the last row of R_0 which contains $(N - \frac{N}{p})$ words and broadcasting it to all other processors except R_o.

Similarly processor R_i send row of size $(i\,\frac{N}{p} + 1)$ to processes R_{i+1}, R_{i+2} ....R_{p-1}

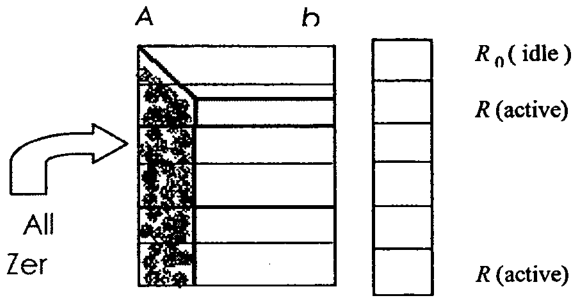


Figure 3(e): *Gauss elimination technique*

Since the data is initially arranged so that block i is in processor R_{i-1}, the processors R_i, R_{i+1}.........R_{p-1} are in the subcube of dimension $\lceil \log_2 (p-i) \rceil$ and therefore we can perform a limited broadcast in $\lceil \log_2 (p-i) \rceil$ steps.

We know that transferring a row of length $(N-J)$ to all processors requires a time bounded by $2[t_s \log p + (N-J)\tau]$.

So by [SA, 86] , the communication time is given by

$$t_c \approx 2\frac{N}{p}\sum_{i=0}^{p-1} \lceil \log_2 (p-i) \rceil\, t_s + 2\sum_{J=1}^{N-1} (N-J)\,\tau$$

$$t_c \approx 2\frac{N}{p}\sum_{i=0}^{p-1} \lceil \log_2 (p-i) \rceil\, t_s + N(N-1)\,\tau \qquad \text{.........3(h)}$$

Take $\quad r = \log_2 p \quad \Rightarrow \quad p = 2^r$

Now equation (3.h) can be written as

$$t_c \approx 2\frac{N}{2^r}\left[\sum_{i=0}^{r-1} [(r-i)\,2^{r-i-1}]\, t_s + N(N-1)\,\tau\right.$$

After solving this expression we get

$$t_c \approx 2\frac{N}{2^r}[(r-1)\,2^r + 1]\, t_s + N(N-1)\,\tau$$

Or

$$t_c \approx 2\frac{N}{p}[(r-1)p + 1]\, t_s + N(N-1)\,\tau \qquad \text{.........3 (i)}$$

For total time we need arithmetic time, to determine the time for arithmetic at each step J, a processor performs at most N/P eliminations, which requires time $\frac{N}{p}(N-(J-1)]\,\omega$, where ω is the time for adding and multiplying a pair of floating point numbers.

Summing over $N-1$ step, we obtain estimated time as

$$t_a \approx \sum_{J=1}^{N-1}\frac{N}{p}[N-(J-1)]\,\omega$$

Or
$$t_a \approx \frac{N\omega}{p}\left[\frac{N^2+N-2}{2}\right] \qquad \text{......... 3(j)}$$

So the total time for performing Gauss elimination on hypercube is

$$T_p = t_c + t_a$$

$$T_p \approx 2\frac{N}{p}\,[(r-1)p+1]\,t_s + N(N-1)\,\tau + \frac{N\omega}{p}\,[\frac{N^2+N-2}{2}]$$

Let time taken by sequentially for Gauss elimination is T_1 with one processor.

Then

$$T_1 = [\sum_{k=0}^{N-1} k^2]\,\omega$$

$$\approx \frac{N(N-1)(2N-1)}{6}\,\omega \qquad \text{.........3(k)}$$

Now speed up can be calculated as

$$S_p = \frac{T_1}{T_p}$$

$$\Rightarrow \frac{\frac{N(N-1)(2N-1)}{6}\,\omega}{2\frac{N}{p}\,[(r-1)p+1]\,t_s + N(N-1)\,\tau + \frac{N\omega}{2p}\,[N^2+N-2]}$$

3.3 Supercube Architecture

Hypercube is a popular network architecture which is widely used in parallel machines. Because it has low message latency and high data bandwidth. But it is not free form drawbacks as well. One most important drawback is number of processors needs to be in the power of *2*. So in case number of processors is not in power of 2 forms, then we need to look for some other network architecture.

A few generalizations of hypercube have been proposed for work number of processors that are not in power of *2* forms.

- **Generalized Hypercube**: A generalized hypercube can be constructed arbitrary number of processors, but they can be written as in product form. So it can not work when number of processors is prime. Since prime number can not be written in product form. Second drawback arises when a new processor is added. In that case significant changes have to be made. [BA, 84]

- **Incomplete Hypercube:** It is incomplete hypercube architecture which removes drawback of generalized hypercube. It has no limit of number of processors. It works any number of processors also for prime numbers. But it leads to a problem of connectivity. Sometimes removing a processor leaves the whole architecture disconnected. So it does not work properly like hypercube. [K, 88]

Sen [S, 89] proposed a generalization of hypercube called supercube which removes all drawbacks of generalized hypercube and incomplete hypercube.

Let in p node supercube, where p lies as $2^{m-1} < p \leq 2^m$. So p is a supergraph of $(m - 1)$ dimension hypercube.

From definition of Supercube, let $G = (V, E)$ be a graph, where V is the set of vertices and E is the set of edges in graph. Assume V contains p vertices, which are numbered from 0 to $p - 1$. So each vertex can be expressed as $x_1 x_2 x_3 \ldots . x_k$ (a k bit sequence).

$$\text{Where } k = \lceil \log_2 p \rceil \text{ and } x_i \in \{0,1\} \qquad \forall i \ , \ 1 \leq i \leq k$$

Vertex set can be partitioned into three subsets V_1, V_2 and V_3 as

$V_1 = \{X \mid X \in V, X = 1u$, where u is a $(k\text{-}1)$ bit sequence$\}$

$V_2 = \{X \mid X \in V, X = 0u, 1u \notin V$ where u is a $(k\text{-}1)$ bit sequence$\}$

$V_3 = \{X \mid X \in V, X = 0u, 1u \in V$ where u is a $(k\text{-}1)$ bit sequence$\}$

The edge set E is union of E_1, E_2, E_3 and E_4. Where

$E_1 = \{(X, Y) \mid X, Y \in V, X = 0u, Y = 0v$, where u, v are $(k\text{-}1)$ bit sequence $\& \ HD(u, v) = 1\}$

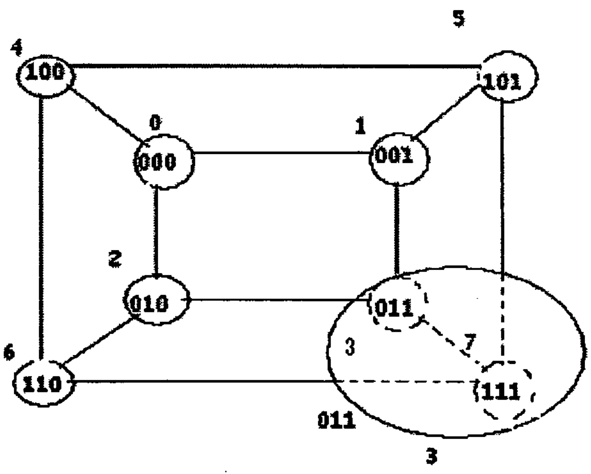
$E_2 = \{(X, Y) \mid X, Y \in V_3, X = 1u, Y = 1v$, where u, v are $(k\text{-}1)$ bit sequence $\& \ HD(u, v) = 1\}$

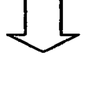
$E_3 = \{(X, Y) \mid X \in V_3, Y \in V_2, X = 1u, Y = 0v$, where u, v are $(k\text{-}1)$ bit sequence & $HD(u, v) = 1$}

$E_4 = \{(X, Y) \mid X \in V_3, Y \in V_1,\ X = 1u, Y = 0u$, where u is a $(k-1)$ bit sequence}

From this definition A p-node supercube can be constructed from an m-dimension hypercube, where $2^{m-1} < p \leq 2^m$

Let nodes in m-dimensional hypercube are labeled from 0 to $(2^m - 1)$. For each node u, $p \leq u \leq (2^m - 1)$. Now merging node u and $(u - 2^{m-1})$ in m dimensional hypercube into a single node labeled as $(u - 2^{m-1})$. And leaving other nodes in m-dimensional hypercube are unchanged. This type we obtain a p-node supercube. Next figure shows the construction of 7 node supercube from a 3 dimensional hypercube.





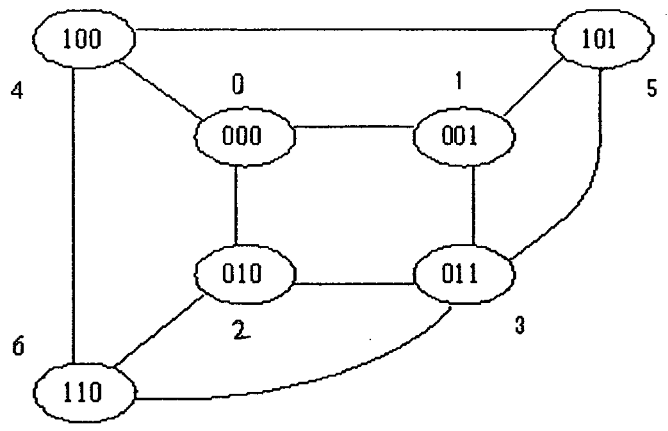


Figure 3(f): *Construction of 7-node supercube from a 8-node hypercube*

3.3.1 Properties of Supercube [Y, 91]

- The diameter of an p-node supercube is $\lfloor \log_2 p \rfloor$ at most.
- Node connectivity of an p-node supercube is at least $\lfloor \log_2 p \rfloor$.
- The node degree of p-node supercube is between $(k - 1)$ and $2(k - 1)$.

Where $k = \lceil \log_2 p \rceil$.

3.3.2 Gauss Elimination on Supercube

A p node supercube is the generalization of hypercube architecture. Supercube is different from hypercube in means number of nodes. So the process of gauss elimination is done in previous way. Then transferring a row of length $(N - J)$ to all processors of supercube require time bounded by

$$2[t_s \log p + (N - J)\tau] \qquad \text{... ....3(l)}$$

Where p is the number of processor in supercube. So the communication time is

$$t_c \approx 2\frac{N}{p}\sum_{i=0}^{p-1} \lceil \log_2 (p-i) \rceil\, t_s + 2\sum_{J=1}^{N-1} (N-J)\,\tau$$

$$t_c \approx 2\frac{N}{p}\sum_{i=0}^{p-1} \lceil \log_2 (p-i) \rceil\, t_s + N(N-1)\,\tau \qquad \text{..........3(m)}$$

Take $r = \lfloor \log_2 p \rfloor$

Now equation 3(m) can be written as

$$t_c \approx 2\frac{N}{p}[\,(r+1)p + 1 - 2^{r+1}\,]\,t_s + N(N-1)\,\tau \qquad \text{.........3(n)}$$

For finding arithmetic time, at each step J a processor requires time $\frac{N}{p}\,[N-(J-1)]\,\omega$, Now summing all steps from 1 to N-1. It gives like previous section

$$t_a \approx \frac{N\omega}{p}\,[\,\frac{N^2+N-2}{2}\,] \qquad \text{.........3(o)}$$

So the total time for performing gauss elimination on supercube is

$$T_p = t_c + t_a$$

$$T_p \approx 2\frac{N}{p}[(r+1)p + 1 - 2^{r+1}\,]\,t_s + N(N-1)\,\tau + \frac{N\omega}{2p}\,[N^2+N-2] \qquad \text{.........3(p)}$$

For performing Gauss elimination on supercube sequentially, it same as for hypercube So from equation (3.k)

$$T_1 \approx \frac{N(N-1)(2N-1)}{6}\,\omega \qquad \text{.........3(q)}$$

Speedup is given by

$$S_p = \frac{T_1}{T_p}$$

$$\Rightarrow \frac{\frac{N(N-1)(2N-1)}{6}\,\omega}{2\frac{N}{p}[(r+1)p + 1 - 2^{r+1}\,]\,t_s + N(N-1)\,\tau + \frac{N\omega}{2p}\,[N^2+N-2]}$$

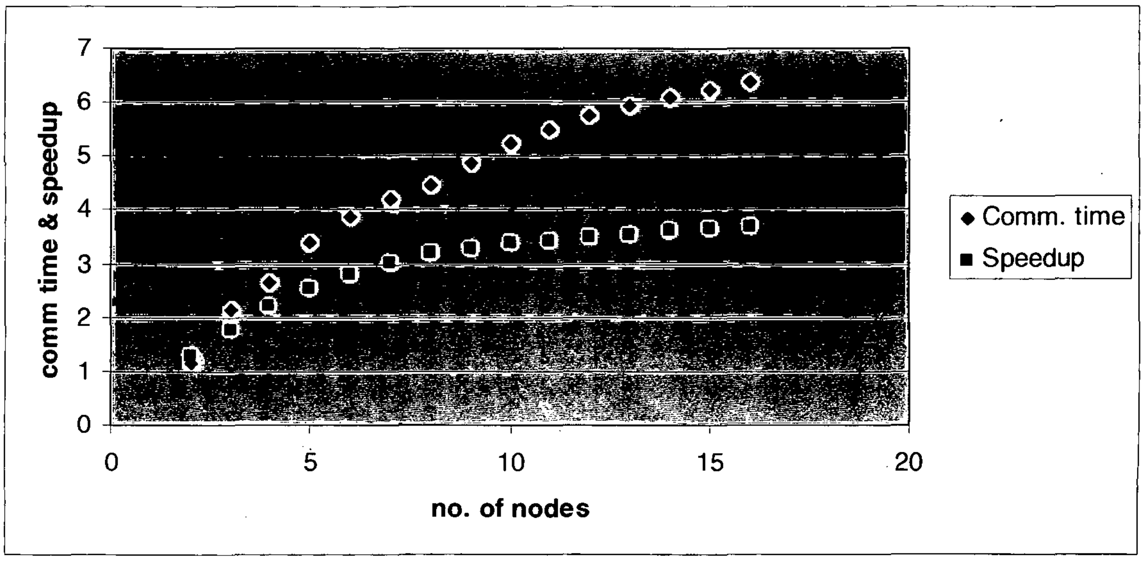
.... 3(r)

3.4 Results for processing time on Hypercube and Supercube architecture

To obtain the performance measure of our parallel system, we use the standard parameters of parallel computing.

Startup time (t_s)	=	1.0×10^{-3} sec
Elemental transfer time (τ)	=	1.0×10^{-7} sec
Size of matrix $N \times N$	=	1024×1024
ω	=	1.0×10^{-7} sec

For result see graph 3(a) and table 3(a) following



Graph 3(a): *Communication time and speedup for Supercube architecture*

Architecture	*No. of nodes*	*Communication time*	*Speedup*
Hypercube	2	1.124	1.276
Supercube	3	2.148	1.781
Hypercube	4	2.66	2.22
Supercube	5	3.376	2.53
Supercube	6	3.854	2.789
Supercube	7	4.196	3.01
Hypercube	8	4.452	3.199
Supercube	9	4.878	3.294
Supercube	10	5.22	3.373
Supercube	11	5.50	3.441
Supercube	12	5.732	3.5
Supercube	13	5.928	3.552
Supercube	14	6.097	3.597
Supercube	15	6.244	3.637
Hypercube	16	6.372	3.672

Table 3(a): *Communication time and speedup for Supercube architecture*

Chapter 4

Gauss Elimination on Pyramid architecture

4.1 Pyramid Architecture

Pyramid architecture combines the properties of a tree architecture and mesh architecture. Pyramid architecture of level λ connects $\frac{4^{\lambda+1}-1}{3}$ nodes. At level *0*, it has one node. At level *1*, it has four nodes which is connected by a 2×2 mesh. Now at next level it has sixteen nodes that is connected by a 4×4 mesh. In similar way at level λ it has a $2^{\lambda}\times2^{\lambda}$ mesh. This $2^{\lambda}\times2^{\lambda}$ mesh is called *base of pyramid architecture*. These meshes are in form of stack, one on top of other. So we can say that a pyramid architecture is of base $n\times n$ is a rooted tree of height $\log_2 n$.In pyramid architecture node at level *0* is called *apex of pyramid architecture*. [Q, 94]

Pyramid architecture of level *2* with *21* nodes shown in figure

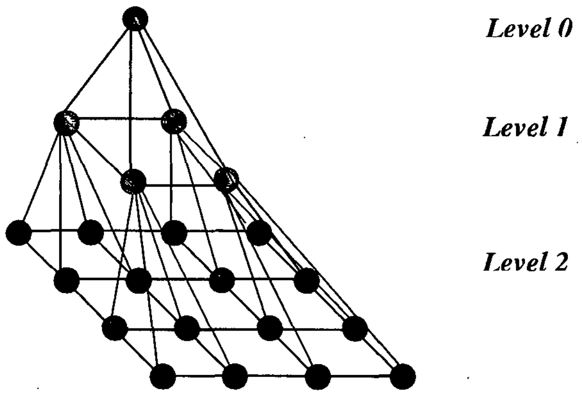


Figure 4(a): *Pyramid architecture of level 2*

Pyramid architecture reduced the diameter over 2D mesh architecture. When a message travel one side to other side in a mesh, then tree traversal approach is better then across the mesh, because the height of pyramid architecture of base $n \times n$ is $\log_2 n$ .So diameter of this pyramid is $2\log_2 n$. In this diameter $\log_2 n$ is for up direction and $\log_2 n$ is for down direction. Whenever in a $n \times n$ mesh, the diameter is $2(n-1)$. In a pyramid architecture every interior node connect with 9 other node, but maximum link per node not greater then 9.

4.2 Gauss elimination on Pyramid Architecture

As in chapter 3, we subdivide the $N \times N$ matrix A into p blocks and each block has N/p rows. Now we perform gauss elimination on pyramid architecture

Case 1: Pyramid of level *1*

Let a $N \times N$ matrix A on apex of pyramid shown in figure

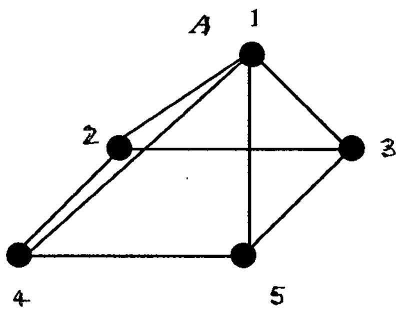


Figure 4(b): *Pyramid of level 1*

Initially transfer the row of length N to nodes 2, 3, 4 and 5 at level 1 from apex node 1. It takes time

$$[t_s + N\tau]$$

Next we perform the arithmetic operations and transfer row of length $(N-1)$ which takes time

$$[t_s + (N-1)\tau]$$

Thus at the end of arithmetic operations on *Apex node*, the total communication time is

$$t_c \text{ (for level } 0) \approx [t_s + N\tau] + [t_s + (N-1)\tau] + \dots\dots\dots\dots$$

$$\dots\dots + [t_s + (N - \frac{N}{5} + 1)\tau]$$

$$\text{Or} \approx \sum_{J=0}^{(4^0)\frac{N}{5}-1} (t_s + (N-J)\tau) \qquad \dots\dots\dots 4(a)$$

Next at node *2*, transfer the row of length (*N* - *N* /*5*) to node *3*, *4* and *5* from node *2* with passes level *0*. It takes time

$$[t_s + (N - \frac{N}{5})\tau]$$

But firstly it goes to node *1* at level *0* and then goes to node *3*, *4* and *5* at level *1*.
So total time require

$$2\,[t_s + (N - \frac{N}{5})\tau]$$

After performing arithmetic operations, transfer the row of length (*N* - *N* /*5* – *1*) which takes total time

$$2\,[t_s + (N - \frac{N}{5} - 1)\tau]$$

At the end of arithmetic operations of node *2*, the total communication time is

$$t_c \text{ (for node } 2) \approx 2\,[t_s + (N - \frac{N}{5})\tau] + 2\,[t_s + (N - \frac{N}{5} - 1)\tau] + \dots\dots$$

$$\dots\dots + 2\,[t_s + (N - \frac{2N}{5} + 1)\tau]$$

$$\text{Or} \approx 2\sum_{J=(4^0)\frac{N}{5}}^{\frac{2N}{5}-1} [t_s + (N-J)\tau]$$

Same thing is repeated for node *3*, *4* and *5*. Therefore the total communication time for nodes at level *1* is

$$t_c \text{ (for level } 1) \approx 2\sum_{J=(4^0)\frac{N}{5}}^{\frac{2N}{5}-1} [t_s + (N-J)\tau] + 2\sum_{J=\frac{2N}{5}}^{\frac{3N}{5}-1} [(t_s + (N-J)\tau] + \dots\dots$$

$$\ldots\ldots + 2 \sum_{J=\frac{4N}{5}}^{(4^0+4^1)\frac{N}{5}-1} [t_s + (N-J)\tau]$$

Or $\quad t_c$ (for level *1*) $\approx 2 \sum_{J=(4^0)\frac{N}{5}}^{(4^0+4^1)\frac{N}{5}-1} [t_s + (N-J)\tau]$ ....4(b)

So the total communication time for all nodes of pyramid is sum of equation (4.a) and (4.b).

$$t_c \approx \sum_{J=0}^{(4^0)\frac{N}{5}-1} (t_s + (N-J)\tau) + 2 \sum_{J=(4^0)\frac{N}{5}}^{(4^0+4^1)\frac{N}{5}-1} (t_s + (N-J)\tau)$$

Or $\quad t_c \approx (1\cdot 4^0 + 2\cdot 4^1)\frac{N}{5} t_s + \sum_{J=0}^{(4^0)\frac{N}{5}-1} (N-J)\tau + 2 \sum_{J=(4^0)\frac{N}{5}}^{(4^0+4^1)\frac{N}{5}-1} (N-J)\tau$

Or $\quad t_c \approx 9\frac{N}{5} t_s + 2 \sum_{J=0}^{(4^0+4^1)\frac{N}{5}-1} (N-J)\tau - \sum_{J=0}^{(4^0)\frac{N}{5}-1} (N-J)\tau$

After solving this approx communication time is

$t_c \approx 9[\frac{N}{5} t_s + (\frac{N^2}{5})\tau + \frac{1}{2}(\frac{N}{5})\tau] - \frac{49}{2}(\frac{N^2}{5^2})\tau$ ......4(c)

For arithmetic time by chapter *3*, at step *k* a node performs at most *N/p* eliminations. Remember that in this case, total number of processors *p* is *5*.

It require time

$$\frac{N}{5}[N-(k-1)]\,\omega$$

Where ω is time for executing a pair of operation consisting of an addition and a multiplication.

For total estimated arithmetic time

$$t_a \approx \sum_{k=1}^{N-1} \frac{N}{5}[N-(k-1)]\,\omega$$

Or $\quad t_a \approx \frac{N\omega}{5}[\frac{N^2+N-2}{2}]$ ....4(d)

So the total time for performing gauss elimination on pyramid of level *1* is

$$T_p = t_c + t_a$$

$$T_p \approx 9[\frac{N}{5} t_s + (\frac{N^2}{5})\tau + \frac{1}{2}(\frac{N}{5})\tau] - \frac{49}{2}(\frac{N^2}{5^2})\tau + \frac{N\omega}{5}[\frac{N^2+N-2}{2}]$$

.... 4(e)

Case 2: Pyramid of level 2

Let a $N \times N$ matrix A on apex of pyramid shown in figure

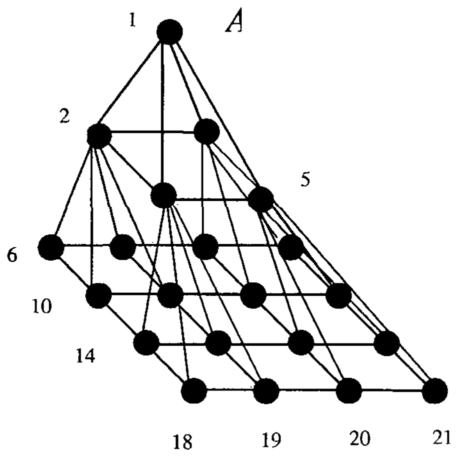


Figure 4(c): *Pyramid of level 2*

Initially transfer the row of length N to nodes *2, 3, 4.......20* and *21* from apex node *1*. It takes maximum time

$$2[t_s + N\tau]$$

Time twice is here because row transfer from level *0* to level *1* and then level *1* to level *2*. Next we perform the arithmetic operations and transfer row of length $(N-1)$ which takes time

$$2[t_s + (N-1)\tau]$$

Thus at the end of arithmetic operations of apex node, the total communication time is

$$t_c \text{ (for level } 0) \approx 2[t_s + N\tau] + 2[t_s + (N-1)\tau] + \dots\dots\dots\dots + 2[t_s + (N - \frac{N}{21} + 1)\tau]$$

$$\approx 2\sum_{J=0}^{(4^0)\frac{N}{21}-1}[t_s + (N-J)\tau] \qquad \dots\dots\dots 4(f)$$

Next at node *2*, transfer the row of length (*N* - *N /21*) to node *3, 4, 5.........20* and *21* from node *2* with passes node *1*. It takes maximum time

$$3[t_s + (N - \frac{N}{21})\tau]$$

thrice is here because route of row transfer from level *1* to level *0*, then level *0* to level *1* and finally level *1* to level *2*. After performing arithmetic operations, transfer the row of length (*N* - *N /21 –1*) which takes total time

$$3[t_s + (N - \frac{N}{21} - 1)\tau]$$

At the end of arithmetic operations of node *2*, the total communication time is

$$t_c \text{ (for node } 2) \approx 3[t_s + (N - \frac{N}{21})\tau] + 3[t_s + (N - \frac{N}{21} - 1)\tau] + \dots\dots\dots$$

$$\dots\dots\dots + 3[t_s + (N - \frac{2N}{21} + 1)\tau]$$

$$\approx 3 \sum_{J=(4^0)\frac{N}{21}}^{\frac{2N}{21}-1} [t_s + (N - J)\tau]$$

Same thing is repeated for node *3, 4* and *5*. Therefore the total communication time for nodes at level *1* is

$$t_c \text{ (for level } 1) \approx 3 \sum_{J=(4^0)\frac{N}{21}}^{(4^0+4^1)\frac{N}{21}-1} [t_s + (N - J)\tau] \qquad \dots\dots\dots 4(g)$$

Now at node *6*, transfer the row of length (*N* - *5N /21*) to node *7, 8, 9.........20* and *21* from node *6* with passes level *1* and level *0*. It takes time

$$4[t_s + (N - \frac{5N}{21})\tau]$$

Time is fourth because route of row transfer from level *2* to level *1*, level *1* to level *0*,then level *0* to level *1* and finally level *1* to level *2*. After performing arithmetic operations, transfer the row of length (*N* - *5N /21 – 1*) which takes total time

$$4[t_s + (N - \frac{5N}{21} - 1)\tau]$$

At the end of arithmetic operations of node *6*, the total communication time is

$$t_c \text{ (for node } 6) \approx 4 \sum_{J=(4^0+4^1)\frac{N}{21}}^{\frac{6N}{21}-1} [t_s + (N-J)\tau]$$

Similarly repeated for node *7, 8........21*. Therefore the total communication time for nodes at level *2* is

$$t_c \text{ (for level } 2) \approx 4 \sum_{J=(4^0+4^1)\frac{N}{21}}^{(4^0+4^1+4^2)\frac{N}{21}-1} [t_s + (N-J)\tau] \qquad ....4(h)$$

So the total communication time for all nodes of pyramid is sum of equation 4(f), 4(g) and 4(h).

$$t_c \approx 2 \sum_{J=0}^{(4^0)\frac{N}{21}-1} [t_s + (N-J)\tau] + 3 \sum_{J=(4^0)\frac{N}{21}}^{(4^0+4^1)\frac{N}{21}-1} [t_s + (N-J)\tau] + 4 \sum_{J=(4^0+4^1)\frac{N}{21}}^{(4^0+4^1+4^2)\frac{N}{21}-1} [t_s + (N-J)\tau]$$

Or

$$t_c \approx (2\cdot 4^0 + 3\cdot 4^1 + 4.4^2)\frac{N}{21} t_s + 2 \sum_{J=0}^{(4^0)\frac{N}{21}-1} (N-J)\tau + 3 \sum_{J=(4^0)\frac{N}{21}}^{(4^0+4^1)\frac{N}{21}-1} (N-J)\tau + 4 \sum_{J=(4^0+4^1)\frac{N}{21}}^{(4^0+4^1+4^2)\frac{N}{21}-1} (N-J)$$

Or

$$t_c \approx 78\frac{N}{21} t_s + 4 \sum_{J=0}^{(4^0+4^1+4^2)\frac{N}{21}-1} (N-J)\tau - \sum_{J=0}^{(4^0+4^1)\frac{N}{21}-1} (N-J)\tau - \sum_{J=0}^{(4^0)\frac{N}{21}-1} (N-J)\tau$$

After solving this approx communication time is

$$t_c \approx 78[\frac{N}{21} t_s + (\frac{N^2}{21})\tau + \frac{1}{2}(\frac{N}{21})\tau] - 869(\frac{N^2}{21^2})\tau \qquad ....4(i)$$

Arithmetic time same as equation (4.d), only change is now p is *21*.

So the total time for performing gauss elimination on pyramid of level *2* is

$$T_p = t_c + t_a$$

$$T_p \approx 78[\frac{N}{21} t_s + (\frac{N^2}{21})\tau + \frac{1}{2}(\frac{N}{21})\tau] - 869(\frac{N^2}{21^2})\tau + \frac{N\omega}{21}[\frac{N^2+N-2}{2}]$$

.... 4(j)

Case 3: Pyramid of level λ

In similar way we can find communication time for λ level pyramid. As we know a λ level Pyramid has $\frac{4^{\lambda+1}-1}{3}$ nodes. Take this number equal to p. So communication time for nodes at level *0* is

$$\left.\begin{aligned}
t_c\,(\text{for level } 0) &\approx \lambda \sum_{J=0}^{(4^0)\frac{N}{p}-1}[t_s+(N-J)\tau] \\
t_c\,(\text{for level } 1) &\approx (\lambda+1) \sum_{J=(4^0)\frac{N}{p}}^{(4^0+4^1)\frac{N}{p}-1}[t_s+(N-J)\tau\,] \\
t_c\,(\text{for level } 2) &\approx (\lambda+2) \sum_{J=(4^0+4^1)\frac{N}{p}}^{(4^0+4^1+4^2)\frac{N}{p}-1}[t_s+(N-J)\tau\,] \\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
t_c\,(\text{for level } \lambda) &\approx (\lambda+\lambda) \sum_{J=(4^0+4^1+\cdots\cdots+4^{\lambda-1})\frac{N}{p}}^{(4^0+4^1+\cdots\cdots+4^{\lambda})\frac{N}{p}-1}[t_s+(N-J)\tau\,]
\end{aligned}\right\}$$

For getting total communication time adding all equations in above brace

$$t_c \;\approx\; [\lambda\cdot 4^0+(\lambda+1)\cdot 4^1+\ldots+(\lambda+\lambda)\cdot 4^{\lambda}]\frac{N}{P}t_s \;+\; \lambda\sum_{J=0}^{(4^0)\frac{N}{P}-1}(N-J)\tau$$

$$+\;(\lambda+1)\sum_{J=(4^0)\frac{N}{P}}^{(4^0+4^1)\frac{N}{P}-1}(N-J)\tau \;+\ldots\ldots\; +\;(\lambda+\lambda)\sum_{J=(4^0+4^1+\ldots+4^{\lambda-1})\frac{N}{P}}^{(4^0+4^1+\ldots+4^{\lambda})\frac{N}{P}-1}(N-J)\tau$$

Or

$$t_c \approx \frac{1}{9}[4(6\lambda-1)4^{\lambda}+(4-3\lambda)]\,(\frac{N}{p}t_s) + (\lambda+\lambda) \sum_{J=0}^{(4^0+4^1+\ldots+4^{\lambda})\frac{N}{P}-1}(N-J)\tau$$

$$- \sum_{J=0}^{(4^0+4^1+\ldots+4^{\lambda-1})\frac{N}{P}-1}(N-J)\tau \text{ - } \ldots\ldots\ldots\ldots\ldots\ldots \text{ - } \sum_{J=0}^{(4^0)\frac{N}{P}-1}(N-J)\tau$$

After solving this we get

$$t_c \approx \frac{1}{9}[4(6\lambda-1)4^{\lambda}+(4-3\lambda)].[\frac{N}{p}t_s+(\frac{N^2}{p})\tau+\frac{1}{2}(\frac{N}{P})\tau\,]$$

$$-\frac{1}{270}\,[16(30\lambda\text{-}1)4^{2\lambda}+40(1\text{-}6\lambda)4^{\lambda}+(15\lambda\text{-}24)]\,(\frac{N^2}{P^2})\tau$$

Arithmetic time same as equation (4.d) also, here number of nodes is p.

So the total time for performing gauss elimination on pyramid of level λ is

$$\left.\begin{aligned} T_p &= t_c + t_a \\ &\approx \frac{1}{9}[4(6\lambda-1)4^{\lambda}+(4-3\lambda)].[\frac{N}{p}t_s+(\frac{N^2}{p})\tau+\frac{1}{2}(\frac{N}{P})\tau\,] \\ &\quad -\frac{1}{270}\,[16(30\lambda\text{-}1)4^{2\lambda}+40(1\text{-}6\lambda)4^{\lambda}+(15\lambda\text{-}24)]\,(\frac{N^2}{P^2})\tau \\ &\quad +\frac{N\omega}{P}\,[\frac{N^2+N-2}{2}\,] \end{aligned}\right\} \quad \ldots 4(k)$$

In all cases we can find the speedup by

$$S_p = \frac{T_1}{T_p} \qquad \ldots\ldots\ldots 4(l)$$

4.3 Length Measure in Pyramid Architecture

Pyramid architecture has requirement of more links to connect nodes over 2D mesh. In this case length of links has increase when level of pyramid is increase. Now we measure length between two nodes in pyramid architecture.

Let each level at distance *l* with other level. This assumption same for all levels, also take two nodes directly connected by link of length *l* in a mesh at any level.

So at level *1* a 2×2 mesh represents in figure

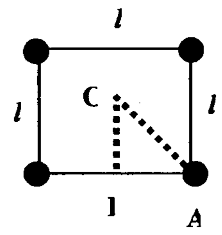


Figure 4(d): *Length for 2×2 mesh*

Half of diagonal length $OA = OP + PA$

$$= \sqrt{\frac{l^2}{4} + \frac{l^2}{4}}$$

$$= \sqrt{\frac{l^2}{2}}$$

The length of link in pyramid architecture between level *0* and level *1* is *AB*. This system show in figure

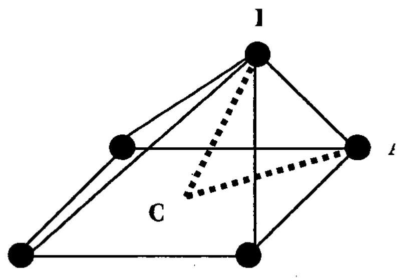


Figure 4(e): *Length for Pyramid of level 1*

Because the distance between level *0* and level *1* is *l*. So the maximum length of link in pyramid architecture between level *0* and level *1* is

$$AB = \sqrt{OB^2 + OA^2}$$

$$= \sqrt{l^2 + \frac{l^2}{2}}$$

$$= \sqrt{\frac{3l^2}{2}}$$

Now we measure length of link between level *1* and level *2*. System architecture for pyramid between level *1* and level *2* shown in figure.

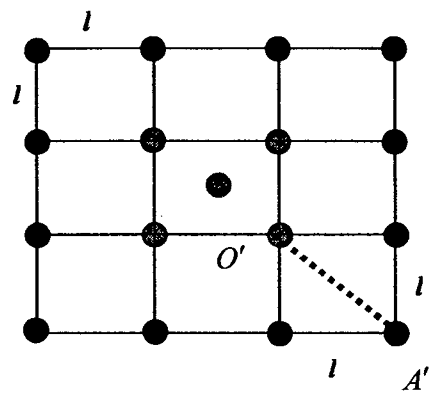


Figure 4(f): *Length for 4×4 mesh*

In figure position of nodes show as, nodes at level *2* represent by black circle, at level *1* and level *0* are represented by red and green circles respectively

At level *2*, distance between O' and A' is

$$O'A' \quad = \sqrt{l^2 + l^2}$$

Or $$O'A' \quad = \sqrt{2l^2}$$

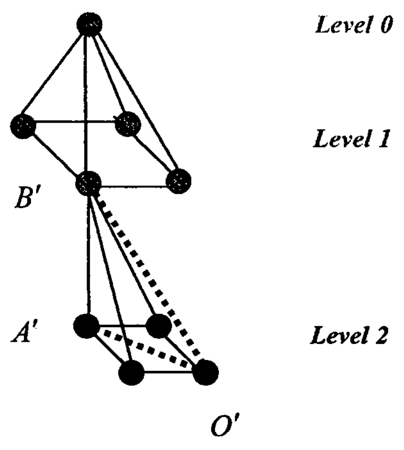


Figure 4(g): *Length for Pyramid of level 2*

As we assume distance between level *1* and level *2* is *l*. So the maximum length of link in pyramid architecture is

$$\begin{aligned} O'B' &= \sqrt{A'B'^2 + O'A'^2} \\ &= \sqrt{l^2 + 2l^2} \\ &= \sqrt{3l^2} \end{aligned}$$

In similar way we can find length of link between any two levels. In generalization, maximum length of link between level (λ-*1*) and level λ is

$$l\sqrt{1+2^{2\lambda-3}}$$

Where $\lambda = 1, 2, 3\ldots$

4.4 Gauss elimination on Pyramid Architecture with length effect

In previous section [4.3] we find length of link between level (λ-*1*) and λ is

$$(\sqrt{1+2^{2\lambda-3}})\, l \quad , \qquad \text{Where } \lambda = 1, 2, 3\ldots\ldots\ldots\ldots\ldots$$

Now we perform gauss elimination on *p* node pyramid architecture for $N \times N$ matrix.

Gauss elimination in this is same as section [4.2]. Here is only difference that communication time is depending on length of link.

Case 1: Pyramid of level *1*

Let an $N \times N$ matrix A on apex of pyramid shown in figure (4.b). Firstly transfer the rows to nodes *2, 3, 4* and *5* at level *1* from apex node *1*. Length of link between level *0* and level *1* is $(\sqrt{\frac{3}{2}})\, l$. Initially transfer the row of length N to nodes *2, 3, 4* and *5* at level 1 from apex node *1*. It takes time

$$[t_s + N\tau(\sqrt{\frac{3}{2}})\, l]$$

In similar way go further. At the end of arithmetic operations on node of apex, the total communication time is

$$t_c \text{ (for level } 0) \approx \sum_{J=0}^{(4^0)\frac{N}{5}-1} [t_s + (N-J)\tau(\sqrt{\frac{3}{2}})l]$$

$$\approx (1.4^0)\frac{N}{5}t_s + \sum_{J=0}^{(4^0)\frac{N}{5}-1} [(N-J)\tau(\sqrt{\frac{3}{2}})l] \qquad \text{........ 4(m)}$$

Now transfer the rows to nodes *3, 4* and *5* at level *1* from node *2* with passes level *0*. At the end of arithmetic operations of node *2*, the total communication time is

$$t_c \text{ (for node } 2) \approx \sum_{J=(4^0)\frac{N}{5}}^{2\frac{N}{5}-1} [2t_s + (N-J)\tau(\sqrt{\frac{3}{2}} + \sqrt{\frac{3}{2}})l]$$

Same thing is repeated for node *3, 4* and *5*. Therefore the total communication time for nodes at level *1* is

$$t_c \text{ (for level } 1) \approx \sum_{J=(4^0)\frac{N}{5}}^{(4^0+4^1)\frac{N}{5}-1} [2t_s + (N-J)\tau(\sqrt{\frac{3}{2}} + \sqrt{\frac{3}{2}})l]$$

$$\approx (2.4^1)\frac{N}{5}t_s + \sum_{J=(4^0)\frac{N}{5}}^{(4^0+4^1)\frac{N}{5}-1} [(N-J)\tau(\sqrt{\frac{3}{2}} + \sqrt{\frac{3}{2}})l] \qquad \text{...... 4(n)}$$

So the total communication time for all nodes of pyramid is sum of equation (4.m) and (4.n).

$$t_c \approx (1.4^0+2.4^1)\frac{N}{5}t_s + \sum_{J=0}^{(4^0)\frac{N}{5}-1}[(N-J)\tau(\sqrt{\frac{3}{2}})l] + \sum_{J=(4^0)\frac{N}{5}}^{(4^0+4^1)\frac{N}{5}-1}[(N-J)\tau(\sqrt{\frac{3}{2}}+\sqrt{\frac{3}{2}})l]$$

Or

$$t_c \approx 9\frac{N}{5}t_s + \sum_{J=0}^{(4^0+4^1)\frac{N}{5}-1}[(N-J)\tau(\sqrt{\frac{3}{2}}+\sqrt{\frac{3}{2}})l] - \sum_{J=0}^{(4^0)\frac{N}{5}-1}[(N-J)\tau(\sqrt{\frac{3}{2}})l]$$

After solving this approx communication time is

$$t_c \approx 9\frac{N}{5}t_s + (9\sqrt{\frac{3}{2}})l\tau\ [(\frac{N^2}{5})+\frac{1}{2}(\frac{N}{5})] - (\frac{49}{2}\sqrt{\frac{3}{2}})l\tau(\frac{N^2}{5^2}) \qquad ....4(o)$$

Arithmetic time same as equation (4.d), here p is *5*. So the total time for performing gauss elimination on pyramid of level *1* is

$$\begin{aligned} T_p &= t_c + t_a \\ &\approx 9\frac{N}{5}t_s + (9\sqrt{\frac{3}{2}})l\tau\ [(\frac{N^2}{5})+\frac{1}{2}(\frac{N}{5})] - (\frac{49}{2}\sqrt{\frac{3}{2}})l\tau(\frac{N^2}{5^2}) \\ &+ \frac{N\omega}{5}[\frac{N^2+N-2}{2}] \qquad ....4(p) \end{aligned}$$

Case 2: Pyramid of level *2*

Let a $N \times N$ matrix A on apex of pyramid shown in Figure (4.g). Firstly transfer the rows to nodes *2, 3, 4 ... 20* and *21* from apex node *1*. Length of link between level *0* and level *1* is $(\sqrt{\frac{3}{2}})$ *l*, level *1* and level *2* is $(\sqrt{3})$ *l*. At the end of arithmetic operations on node of apex, the total communication time is

$$t_c \text{ (for level } 0) \approx \sum_{J=0}^{(4^0)\frac{N}{21}-1}[2t_s + (N-J)\tau(\sqrt{\frac{3}{2}}+\sqrt{3})l]$$

$$\approx (2.4^0)\frac{N}{21}t_s + \sum_{J=0}^{(4^0)\frac{N}{21}-1}(N-J)\tau(\sqrt{\frac{3}{2}}+\sqrt{3})l \qquad ....4(q)$$

Summation of lengths shows maximum length between level *0* and level *2*. Now transfer the rows to nodes *3, 4, 5... 20* and *21* from node *2* with passes level *0*. So it takes length between level *0* and level *1* two times. Further between level *1* and level *2* one times at the end of arithmetic operations of node *2*, the total communication time is

$$t_c \text{ (for node } 2) \approx \sum_{J=(4^0)\frac{N}{21}}^{\frac{2N}{21}-1} [3t_s + (N-J)\tau(2\sqrt{\frac{3}{2}} + \sqrt{3})l]$$

Same thing is repeated for node *3, 4* and *5*. Therefore the total communication time for nodes at level *1* is

$$t_c \text{ (for level } 1) \approx \sum_{J=(4^0)\frac{N}{21}}^{(4^0+4^1)\frac{N}{21}-1} [3t_s + (N-J)\tau(2\sqrt{\frac{3}{2}} + \sqrt{3})l]$$

$$\approx (3.4^1)\frac{N}{21}t_s + \sum_{J=(4^0)\frac{N}{21}}^{(4^0+4^1)\frac{N}{21}-1} (N-J)\tau(2\sqrt{\frac{3}{2}} + \sqrt{3})l \qquad \text{........ 4(r)}$$

Similar way is repeated for node *6, 7, 8... 20* and *21*. Here take length between level *0* and level *1* two times. Further between level *1* and level *2* two times Therefore the total communication time for nodes at level *2* is

$$t_c \text{ (for level } 2) \approx (4.4^2)\frac{N}{21}t_s + \sum_{J=(4^0+4^1)\frac{N}{21}}^{(4^0+4^1+4^2)\frac{N}{21}-1} (N-J)\tau(2\sqrt{\frac{3}{2}} + 2\sqrt{3})l \qquad \text{...........4(s)}$$

So the total communication time for all nodes of Pyramid is sum of equation (4.p), (4.q) and (4.r)

$$t_c \approx (2.4^0 + 3.4^1 + 4.4^2)\frac{N}{21}t_s + \sum_{J=0}^{(4^0)\frac{N}{21}-1} (N-J)\tau(\sqrt{\frac{3}{2}} + \sqrt{3})$$

$$+ \sum_{J=(4^0)\frac{N}{21}}^{(4^0+4^1)\frac{N}{21}-1} (N-J)\tau(2\sqrt{\frac{3}{2}} + \sqrt{3})l + \sum_{J=(4^0+4^1)\frac{N}{21}}^{(4^0+4^1+4^2)\frac{N}{21}-1} (N-J)\tau(2\sqrt{\frac{3}{2}} + 2\sqrt{3})l$$

Or

$$t_c \approx 78\frac{N}{21}t_s + \sum_{J=0}^{(4^0+4^1+4^2)\frac{N}{21}-1} (N-J)\tau(2\sqrt{\frac{3}{2}} + 2\sqrt{3})l - \sum_{J=0}^{(4^0+4^1)\frac{N}{21}-1} (N-J)\tau(\sqrt{3})l$$

$$- \sum_{J=0}^{(4^0)\frac{N}{21}-1} (N-J)\tau(\sqrt{\frac{3}{2}})l$$

After solving this approx communication time is

$$t_c \approx 78\frac{N}{21}\,t_s + (41\sqrt{\frac{3}{2}} + 37\sqrt{3})\,l\tau\,[(\frac{N^2}{21}) + \frac{1}{2}(\frac{N}{21})]$$

$$- \frac{1}{2}(881\sqrt{\frac{3}{2}} + 857\sqrt{3})\,l\tau\,(\frac{N^2}{21^2}) \qquad \text{..........4(t)}$$

Arithmetic time same as equation (4.d), with *p* is *21*. So the total time for performing gauss elimination on pyramid of level *2* is

$$T_p = t_c + t_a$$

$$\approx 78\frac{N}{21}\,t_s + (41\sqrt{\frac{3}{2}} + 37\sqrt{3})\,l\tau\,[(\frac{N^2}{21}) + \frac{1}{2}(\frac{N}{21})]$$

$$- \frac{1}{2}(881\sqrt{\frac{3}{2}} + 857\sqrt{3})\,l\tau\,(\frac{N^2}{21^2}) + \frac{N\omega}{21}\,[\frac{N^2 + N - 2}{2}]$$

$$\text{............... 4(u)}$$

Case 3: Pyramid of level λ

In similar way we can find communication time for λ level Pyramid. As we know a λ level Pyramid has $\frac{4^{\lambda+1}-1}{3}$ nodes. Take this number equal to *p*. So communication time for nodes at level *0* is

$$t_c \text{ (for level } 0) \approx (\lambda.4^0)\frac{N}{P}t_s) + \sum_{J=0}^{(4^0)\frac{N}{P}-1}(N-J)\tau\,(\sqrt{\frac{3}{2}} + \sqrt{3} + \ldots.. + \sqrt{1+2^{2\lambda-3}}\,)\,l$$

$$t_c \text{ (for level } 1) \approx [(\lambda+1).4^1]\frac{N}{P}t_s + \sum_{J=(4^0)\frac{N}{P}}^{(4^0+4^1)\frac{N}{P}-1}(N-J)\tau\,(2\sqrt{\frac{3}{2}} + \sqrt{3} + \ldots.. + \sqrt{1+2^{2\lambda-3}}\,)\,l$$

$$t_c\text{(for level } 2) \approx [(\lambda+2).4^2]\frac{N}{P}t_s + \sum_{J=(4^0+4^1)\frac{N}{P}}^{(4^0+4^1+4^2)\frac{N}{P}-1}(N-J)\tau\,(2\sqrt{\frac{3}{2}} + 2\sqrt{3} + \ldots.. + \sqrt{1+2^{2\lambda-3}}\,)\,l$$

....

....

$$t_c \text{ (for level } \lambda) \approx [(\lambda+\lambda).4^{\lambda}]\frac{N}{P}t_s + \sum_{J=(4^0+4^1+\cdots+4^{\lambda-1})\frac{N}{P}}^{(4^0+4^1+\cdots+4^{\lambda})\frac{N}{P}-1}(N-J)\tau \left(2\sqrt{\frac{3}{2}} + 2\sqrt{3} + \ldots + 2\sqrt{1+2^{2\lambda-3}}\right)l$$

For getting total communication time adding all equations in above and solve it

So the total communication time is

$$t_c \approx \frac{1}{9}[4(6\lambda-1)4^{\lambda}+(4-3\lambda)]\left(\frac{N}{p}\right)t_s + 2\sum_{J=0}^{(4^0+4^1+\cdots+4^{\lambda})\frac{N}{P}-1}(N-J)\tau\left(\sum_{r=1}^{\lambda}\sqrt{1+2^{2r-3}}\right)l$$

$$- \sum_{r=1}^{\lambda}\left[\sum_{J=0}^{\left(\frac{4^r-1}{3}\right)\frac{N}{P}-1}(N-J)\tau\left(\sqrt{1+2^{2r-3}}\right)l\right] \qquad \ldots\ldots. 4(v)$$

Arithmetic time same as equation (4.d) also, here number of nodes is p.

So the total time for performing gauss elimination on pyramid of level λ is

$$T_p = t_c + t_a$$

$$\left.\begin{aligned} T_p \approx\ & \frac{1}{9}[4(6\lambda-1)4^{\lambda}+(4-3\lambda)]\left(\frac{N}{p}\right)t_s + 2\sum_{J=0}^{(4^0+4^1+\cdots+4^{\lambda})\frac{N}{P}-1}(N-J)\tau\left[\sum_{r=1}^{\lambda}\sqrt{1+2^{2r-3}}\right]l \\ & - \sum_{r=1}^{\lambda}\left[\sum_{J=0}^{\left(\frac{4^r-1}{3}\right)\frac{N}{P}-1}(N-J)\tau\left(\sqrt{1+2^{2r-3}}\right)l\right] + \frac{N\omega}{P}\left[\frac{N^2+N-2}{2}\right] \end{aligned}\right\}$$

$$\ldots\ldots\ldots.4(w)$$

Speed up can be evaluated by equation 4(l).

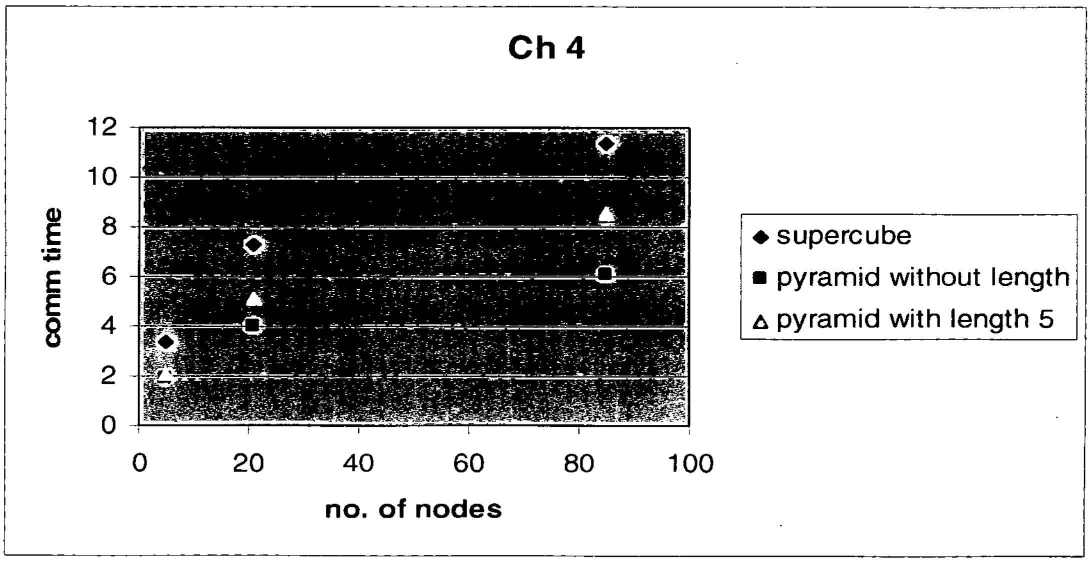
4.5 Comparison of results on Pyramid and Supercube architecture

To obtain the performance measure of our parallel system, we use standard parameters of parallel computing.

Startup time (t_s)	=	1.0×10^{-3} sec
Elemental transfer time (τ)	=	1.0×10^{-7} sec
Size of matrix $N \times N$	=	1024×1024
ω	=	1.0×10^{-7} sec

Architecture	*No. of nodes*	*Length*	*Communication time*	*Speedup*
Supercube	**5**	×	**3.37**	**2.53**
Pyramid (without *l*)	5	×	1.93	2.82
Pyramid (with *l*)	5	5	2.11	2.77
Pyramid (with *l*)	5	10	2.37	2.72
Supercube	**21**	×	**7.32**	**3.62**
Pyramid (without *l*)	21	×	3.98	5.46
Pyramid (with *l*)	21	5	5.13	4.64
Pyramid (with *l*)	21	10	6.46	3.96
Supercube	**85**	×	**11.38**	**2.97**
Pyramid (without *l*)	85	×	6.10	5.31
Pyramid (with *l*)	85	5	8.55	3.89
Pyramid (with *l*)	85	10	11.27	3.00

Table 4(a): *Communication time and speedup for Supercube and Pyramid architecture*



Graph 4(a): *Communication time for Supercube and Pyramid architecture*

Chapter 5
Conclusion

In parallel computing, Hypercube architecture works for limited number of nodes. It only works for 2^n (where n is any non negative integer) number of nodes. It can calculate the processing time in parallel computing if number of nodes is in 2^n forms. But it cannot calculate the exact processing time if number of nodes is not in 2^n forms. If number of nodes is not in the form of 2^n , then it only calculates the upper bound processing time.

This limitation is eliminated by supercube architecture which can works for arbitrary number of nodes. It can calculate the exact processing time in parallel computation. We applied the "**Gauss Elimination Method**" on supercube architecture to calculate the exact processing time in parallel computation.

There is one more architecture which is best suited for $\frac{4^{\lambda+1}-1}{3}$ (where n is any non negative integer) number of nodes. That architecture is called *Pyramid Architecture*. Supercube architecture gives the better result for all the nodes except (1, 5, 21, 85 ....) for this kind of structure, pyramid architecture gives the better result than supercube architecture. "**Gauss Elimination Method**" is used to calculate the processing time for pyramid architecture in parallel computing.

References

[AHS, 77] A.H. Sameh, Numerical parallel algorithms- A Survey, Lawrie, D. and Sameh, A.H. ed., "*High Speed computer and algorithm organization* ", Academic Press, 1977, pp. 207-28

[BA, 84] L. N. Bhuyan and D. P. Agrawal, "*Generalized Hypercube and Hyperbus structures for a computer network*", IEEE Transactions on Computers, vol. C-33, no. 4, pp. 323-333, 1984.

[SS, 85] Y. Saad and M.H. Schultz, "*Topological properties of Hypercube",* Research report 389, Dept. comp. sc., Yale university, 1985

[SA, 86] Y. Saad, "*Gauss elimination on Hypercube*", Parallel algorithms and architecture (1986)

[K, 88] H.P.Katseff, "*Incomplete Hypercube",* IEEE Transactions on Computers. 37(1988), pp. 604-607

[S, 89] A.Sen, "*Supercube:An Optimal fault tolerant network architecture*", Acta inform.,26(1989), pp. 741-748

[BT, 89] Dimitri P. Bertsekas and John N. Tsitsilis, "*Parallel and Distributed computation: Numerical Methods",* Prentice Hall, Englewood ciffs, NJ, 1989

[Y, 91] S.M.Yuan, "*Topological properties of Supercube*", Inform. process letter 37(1991), pp. 241-245

[AL, 91] A. El-Amawy and S.Latifi, " *Properties and Performance of Folded Hypercube* ", IEEE Trans parallel and distributed systems, Vol.2, No.1, Jan.1991, pp.31-42

[TW, 91] N.F. Tzeng and S. Wei, "*Enhanced Hypercubes* ", IEEE Trans. Comput., Vol.40, No.3,March 1991, pp.284-294

[KH, 93] Kai Hwang, "*Advanced Computer Architecture: Parallelism, Scalability, Programmability*", McGraw Hill, 1993

[Q, 94] Michael J.Quinn, "*Parallel Computing: Theory and Practice*", McGraw Hill, 1994

[AAGV, 03] Ananth Grama, Anshul Gupta, George Karypis, Vipin Kumar, "*Introduction to parallel Computing*", Pearson Education Ltd., 2003

[N, 04] Narsingh Deo, "*Graph Theory with applications to engineering and computer science*", Prentice hall of India pl. 2004