ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Monthly Peer Reviewed Open Access International e-Journal

An Innovative Fault Tolerant Searching Algorithm in Hypercube Network Environment for Better Throughput

Laxman Kumar D
M.Tech Student ,
Computer Science Engineering Department,
B V Raju Institute of Technology, Narasapur.

K Karthik
Asst Professor
Computer Science Engineering Department,
B V Raju Institute of Technology, Narasapur.

Abstract:

Parallel processing is the ability to simultaneously process incoming stimuli of differing quality. Parallel processing is the simultaneous use of more than one CPU or processor core to execute a program or multiple computational threads. Ideally, parallel processing makes programs run faster because there are more engines (CPUs or cores) running it. The interconnection network is responsible for fast and reliable communication among the processing nodes in any parallel computer.

The demands on the network depend on the parallel computer architecture in which the network is used. In present day, there is higher demand for processing of large data in short period of time. We need advanced search algorithms in order to meet the increasing demand. In this paper, we have studied various search methods employed, compared them and come up with a parallel search algorithm which has better performance over legacy models.

Keywords:

Interconnect Networks, Parallel Processing, Search, performance, throughput.

Introduction:

Parallel Processing is a form of computation in which many calculations are carried out simultaneously, operating on the principle that large problems can often be divided into smaller ones, which are then solved concurrently (“in parallel”). In practice, it is often difficult to divide a program in such a way that separate CPUs or cores can execute different portions without interfering with each other.

Most computers have just one CPU, but some models have several, and multi-core processor chips are becoming the norm. There are even computers with thousands of CPUs.

With single-CPU, single-core computers, it is possible to perform parallel processing by connecting the computers in a network. However, this type of parallel processing requires very sophisticated software called distributed processing software.

Parallelism Vs Concurrency:

Concurrency is a term used in the operating systems and databases communities which refers to the property of a system in which multiple tasks remain logically active and make progress at the same time by interleaving the execution order of the tasks and thereby creating an illusion of simultaneously executing instructions.

Parallelism, on the other hand, is a term typically used by the supercomputing community to describe executions that physically execute simultaneously with the goal of solving a problem in less time or solving a larger problem in the same time. Parallelism exploits concurrency.

Parallel processing is also called parallel computing. In the quest of cheaper computing alternatives parallel processing provides a viable option. The idle time of processor cycles across network can be used effectively by sophisticated distributed computing software.

The term parallel processing is used to represent a large class of techniques which are used to provide simultaneous data processing tasks for the purpose of increasing the computational speed of a computer system.

Advantages:

- Faster execution time, so higher throughput.

Disadvantages:

- More hardware required, also more power requirements.
- Not good for low power and mobile devices.

Interconnection networks :

When more than one processor needs to access a memory structure, interconnection networks are needed to route data—

• from processors to memories (concurrent access to a shared memory structure), or

• from one PE (processor + memory) to another (to provide a message-passing facility). Inevitably, a large bandwidth is required to match the combined bandwidth of the processing elements.

Measures of interconnection performance :

Several metrics are commonly used to describe the performance of interconnection networks:

•Connectivity, or degree, the number of nodes that are immediate neighbors of a node (i.e., the number of nodes that can be reached from it in one hop).

•Diameter, the maximum number of nodes through which a message must pass on its way from source to destination. Diameter measures the maximum delay in transmitting a message from one processor to another.

•Average distance, where the distance between two nodes is defined by the number of hops in the shortest path between those nodes. Average distance is given by

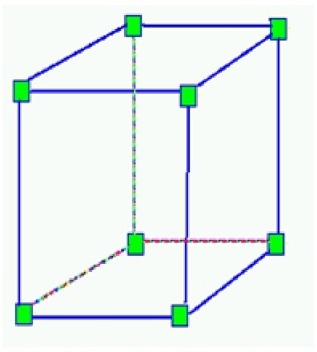
$$d_{avg} = \frac{\sum_{d=1}^{r} (d \cdot N_d)}{N-1}$$

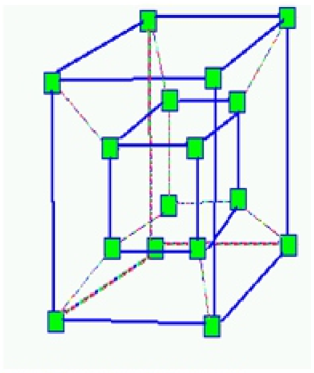
where N is the number of nodes, Nd is the number of nodes at distance d apart, and r is the diameter.

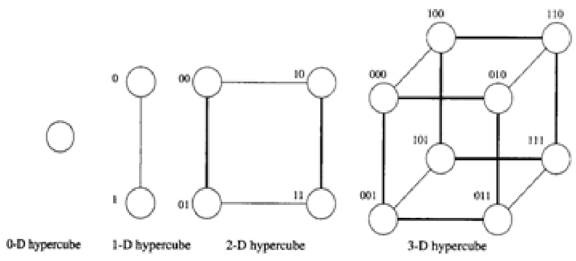
•Bisection width, the smallest number of wires you have to cut to disconnect the network into two equal halves (±1).

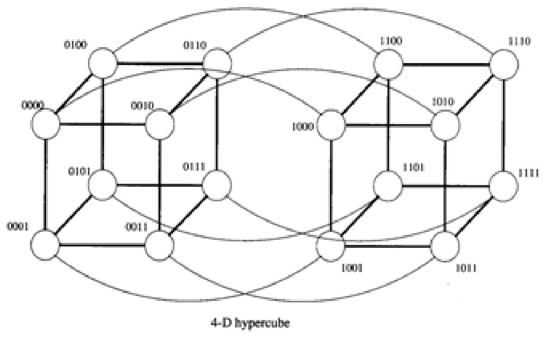
Hypercube Network Topologies:

Visualizing N-Dimensional Hypercubes:


Three-Dimensional Hypercube


Four-Dimensional Hypercube

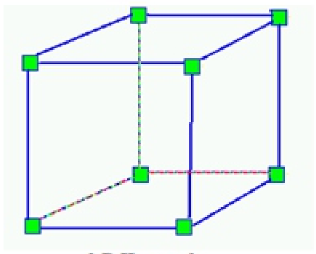




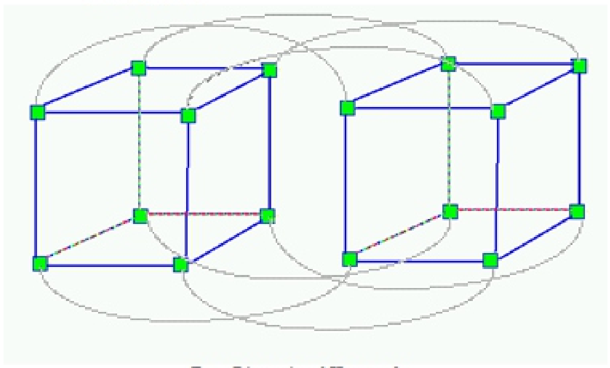
Constructing an N-Dimensional Hypercube:

For dimension N+1, just create two hypercubes of dimension N, and connect analagous vertices.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Monthly Peer Reviewed Open Access International e-Journal

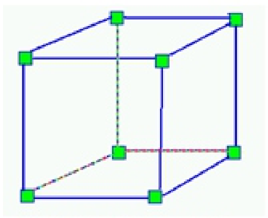

3-D Hypercube

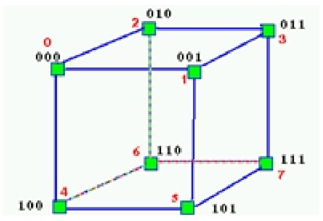
Dimensions:	N	3
Nodes:	2^N	$2^3 = 8$
Links per Node:	N	3
Longest Path:	N	3


Four-Dimensional Hypercube

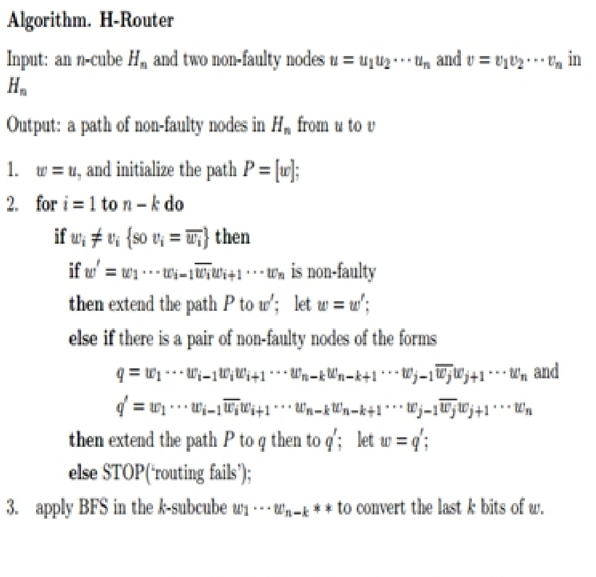
Node Numbering:

Each node must differ from an adjacent node by at most one binary digit.


Three-Dimensional Hypercube



Adjacent nodes differ by one binary digit.


Algorithm. H-Router

```
Input: an n-cube $H_n$ and two non-faulty nodes $u = u_1u_2\cdots u_n$ and $v = v_1v_2\cdots v_n$ in $H_n$

Output: a path of non-faulty nodes in $H_n$ from $u$ to $v$

1.  $w = u$, and initialize the path $P = [w]$;
2.  for $i = 1$ to $n - k$ do
        if $w_i \neq v_i$ {so $v_i = \overline{w_i}$} then
          if $w' = w_1\cdots w_{i-1}\overline{w_i}w_{i+1}\cdots w_n$ is non-faulty
          then extend the path $P$ to $w'$;   let $w = w'$;
          else if there is a pair of non-faulty nodes of the forms
                $q = w_1\cdots w_{i-1}w_iw_{i+1}\cdots w_{n-k}w_{n-k+1}\cdots w_{j-1}\overline{w_j}w_{j+1}\cdots w_n$ and
                $q' = w_1\cdots w_{i-1}\overline{w_i}w_{i+1}\cdots w_{n-k}w_{n-k+1}\cdots w_{j-1}\overline{w_j}w_{j+1}\cdots w_n$
          then extend the path $P$ to $q$ then to $q'$;   let $w = q'$;
          else STOP('routing fails');
3.  apply BFS in the $k$-subcube $w_1\cdots w_{n-k}**$ to convert the last $k$ bits of $w$.
```

A routing algorithm for hypercube networks

Interconnect comparison at-a-glance:

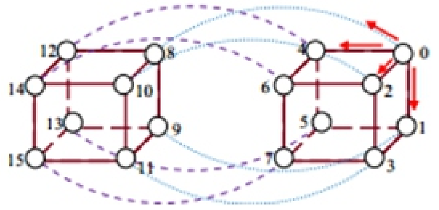
Network Topology	Number of Nodes	Node Degree
Linear and Ring	d	2
Shuffle-Exchange	2^d	3
2D Mesh	d^2	4
Hypercube	2^d	d
Star	m!	m-1
De Bruijn	2^d	4
Binary Tree	2^d - 1	3
Butterfly	$(d+1)* 2^d$	d+1
Omega	2^d	2
Pyramid	$(4^{d+1} – 1)/3$	9

Parallel Search on Hypercube:

Presume n is the quantity of data elements and N is the amount of Processor Elements (PE) in this algorithm. Every node is a processing element, which just does comparison operation. It is considered that all data elements are located in processing elements previously that are nodes of hypercube.

The plan is to seek for the key value of k among data elements that are obtainable and distributed in the nodes of hypercube. The hypercube is understood to have two cubes for this example, which have 16 processing element. The subsequent steps will be performed the algorithm to be carried out.

Step 1: At first, k is fed into PE0 and this PE compare its data eleements with the value of k, if it has a value equal to k, it sends one with k to PEs 1, 2, 4, and 8. If PE0 doesn't have the value k, it will send zero to the mentioned PEs.



Step 2: PEs 1, 2, 4, and 8 compare their data elements with k. If these two data were equal. they send one, which means the data exists; otherwise they send zero. PE1 sends the results to PEs 3, 5, and 9. PE2 transfers the results to PEs 3, 6, and 10. PE4 gives its results to PEs 5,6, and 12. PE8 sends the results to PEs 9 , 10, and 12.

Step 3: PEs 3, 5, 6, 9, 10, and 12 do the same operation and compare their data elements with k. PE3 transfers its results to PEs 7, and 11. PE5 sends the results to PEs 7, and 13. PE6 transfers the results to PEs 7, and 14. PE9 gives its results to PEs 11, and 13. PE10 transfers the results to PEs 11, and 14. PE12 sends the results to PEs 13, and 14. These activities are performed simultaneously.

Step 4: PEs 7, 11, 13, and 14 accomplish the same operation and compare their data elements with k. If each of them had the data, they will transfer one otherwise will send zero. These PEs transfer their results to PE15.

Step5: PE15 receives the results from mentioned processing elements. These data, which are ones or zeros comprise a number. If this number is zero, it means the key k, doesn't exist and if the number becomes greater than zero, it means that data exists. This number also specify which processing elements had the value of k

Conclusion:

In this paper, we studied various topologies and proposed a new fault tolerant search algorithm on hypercube using a new technique. This search algorithm has an acceptable order using a appropriate hardware cost where execution time is O(logN) with fault tolerrance property. The pipelined of this search algorithm will acquire more throughput which leads to better performance.

References:

[1] Masumeh Damrudi, Kamal Jadidy Aval, A Parallel Search on Hypercube Interconnection Network, Journal of Computer Science & Computational Mathematics, Volume 2, Issue 1, January 2012

[2] H. El-Rewini and M. Abd-El-Barr, Advanced Computer Architecture and Parallel Processing. USA: John Wiley & Sons Publishing, 2005.

[3] D. Jiang and J. P. Singh, “A methodology and an evaluation of the SGI Origin2000,” SIGMETRICS Perform. Eval. Rev., vol. 26, pp. 171-181, 1998.

[5] Ahmed Louri, “Optical Interconnection Networks for Scalable High-performance Parallel Computing Systems,” Optical Interconnects Workshop for High Performance Computing, Oak Ridge, Tennessee, Nov 8-9, 1999.

[
4] Ahmed Louri and Hongki Sung, “An Optical Multi-Mesh Hypercube: A Scalable Optical interconnection Network for Massively Parallel Computing,” Journal of Lightwave Technology, Vol. 12, No. 4, pp: 704-716, Apr. 1994.

[7] A. Ferna´ndez, “Homogeneous product networks for processor interconnection,” PhD thesis, University of Southwestern Louisiana, Lafayette, October 1994.

[8] Zhaoyang Li, Yi Zhang, Yu Chen and Ruichun Tang, “Design and implementation of a high-performance interconnection network,” Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies, PDCAT’2003, 27-29, Aug. 2003.

[9] N. Gopalakrishna Kini, M. Sathish Kumar and Mruthyunjaya H. S., “Analysis and Comparison of Torus Embedded Hypercube Scalable Interconnection Network for Parallel Architecture,” International journal of Computer Science and Network Security, vol. 9, No.1, pp. 242-247, Jan. 2009.

[10] N. Gopalakrishna Kini, M. Sathish Kumar and Mruthyunjaya H.S., “A Torus Embedded Hypercube Scalable Interconnection Network for Parallel Architecture,” IEEE explore conference publications, 2009.