ISSN: 0369-8963

A High Availability Algorithm for HSDC Architecture using Pipelining Technique

[1] R.Kanimozhi ,Assistant Professor,
CSE Department, Bharathiyar Institute of Engineering for Women, Attur, India
[2] Dr.A.Neela Madheswari, Professor,
CSE Department, Mahendra Engineering College, Namakkal, India

Abstract— The rapidly expanding network connection requires that the data be maintained with more concurrently linked ports being switched in order to solve the storage issue, which can be overridden by physical systems to prevent storage allocation. Additionally, introduce the idea of n-port switching here. Using the high availability algorithm, this system can determine the optimal time for the system as well as automatically adapt the storage. In the past, they could allocate storage by using an n-port switch, but the proposed notion might take precedence and work in parallel. With the aid of intelligence, it serves as its predefined storage device. Consider a real-time data centre that has created invoices for its clients and sold things. In this situation, management must maintain a lot of data on the products, their prices, and customer information in addition to managing all the data about the resources. To store this amount of information, they therefore require a lot of storage space in addition to an on-premise server. Here, our algorithm will employ the idea of the on-premises server with decision-making and zero downtime to lower the entire infrastructure. This idea can greatly benefit from the high availability algorithm in terms of lowering downtime. Each data storage unit can be set up as a port so that several checks can be made on the storage and storage space can be allocated to store the data. We can also avoid downtime in the vast majority of situations. Use the fault tolerance algorithm in existing systems. It should be taking care of each component, minimising errors, preventing incorrect communication, and showcasing the high availability technique that was just presented.

Index Terms—Data center, Interconnection network, Network topology, Hypercube, Incremental scalability, Routing algorithm.

INTRODUCTION

Data centres, which are exploding in bulk, are the foundation of cloud computing over the Internet. The design of the network topologies and protocols necessary to connect tens of thousands or even hundreds of thousands of servers, storage devices, and networking components within a single data centre constitutes the data centre network. While this is going on, the data centre can provide inexpensive equipment costs, a large and balanced network capacity, ease of expansion, scalable communication performance, and robust fault tolerance.

While offering a wide range of online applications including search, gaming, Web mail, and others, data centres typically entail infrastructure services like GFS Map-reduce and Dryad. Server-centric and switch-centric are the two broad categories into which the data centre network (DCN) architectures can be categorised. Servers act as both a network or warder and a server in server-centric designs. These approaches increase the server's packet relay overhead. A switch-centric network often uses multiple switch trees to link the servers together. The interconnection intelligence in switch-centric systems depends on switches, whereas servers do not need to be changed for connectivity purposes. This group includes the networks Fat-Tree and VL2. Recursively specified network architectures for data centres such as FiConn, DCell, and BCube show that the kth level structure is built by connecting multiple. As the layers rise in these architectures, the number of servers also rises dramatically. For instance, if we build the D-Cell structure using 16-port switches, there are 272 servers when k = 1. The number of servers can rise to 74256 when k is raised from 1 to 2. As a result, the network structures that are defined recursively are unable to achieve a fine-grained expansion so that servers can be progressively added to the system in accordance with the needs of applications.

Another server-centric data centre network architecture is HCube. Enhancing the effectiveness of searching for related data items in datacenters similar to Torus is the main driver behind HCube. When the Hamming distance between the identifiers of two data items is low, HCube considers them to be comparable. On physically close-by servers, related data items can be kept in HCube. By doing this, all related data items stored on these servers may be found using a single search request. A new class of data centre network topologies, known as dual-centric architectures, has just been proposed.

The routing intelligence in a dual-centric data centre network architecture can be installed on servers as well as switches. The dual-centric architectures combine the benefits of switch-centric and server-centric designs, and they provide a number of useful features for real-world data centres. To accommodate the exponential development of data, several data centres have been built recently all over the world. A large number of businesses, including Microsoft, Google, Facebook, Yahoo, and Amazon, have made considerable investments to build data centres. These massive data centres are capable of dynamically allocating vast quantities of hardware, software, and database resources to millions of Internet users at once. More storage space is required to keep the growing volume of data. Adding new components as opposed to swapping out gold ones is a traditional method of increasing storage. In reality, a few storage hosts are usually added on occasion as opposed to introducing a large number of servers all at once. Expect little to no impact on the system operator or the system itself during the addition.

By adding a small number of servers, a data centre architecture with high incremental scalability can be made

https://doi.org/10.37896/pd91.5/91556

ISSN: 0369-8963

larger while maintaining its topological characteristics. In order to retain the topological features of modern data centre network architectures as the size of the data centres continuously increases, it is becoming increasingly difficult to achieve incremental scaling. The performance of data centres is known to be greatly impacted by the network topology. An interconnection network should have certain characteristics, such as low diameter, scalability, large bisection width, and others.

One of the most adaptable and effective networks for parallel computation is the hypercube, which can effectively replicate any other network of a comparable size. The hypercube and its associated networks, including the Twisted, Folded, Generalized, and Hierarchical cubes, as well as the Exchanged, Crossed, and Crossed cubes, are all strong contenders for the architecture of parallel computer systems. By combining two n-dimensional hypercubes, one can create a (n + 1)-dimensional hypercube. The n-dimensional hypercube can only link exactly 2n vertices, severely limiting the range of system sizes and creating a significant discrepancy between the two permitted sizes. As a result, incomplete hypercube, a modified hypercube topology with greater system size flexibility, is suggested. By combining numerous hypercubes of smaller dimensions, an incomplete hypercube is created. The main topological characteristics of the incomplete hypercube are comparable to those of the complete hypercube, and it can have any number of vertexes. Additionally, the incomplete hypercube can grow by continually adding hypercubes of various dimensions.

RELATE WORK

We describe many data centre network topologies in this section. [1] Sprint Net, a cutting-edge server-centric network architecture for data centres, was designed, put into use, and evaluated. In terms of network bandwidth, fault tolerance, and latency, Sprint Net performs well. Sprint Net is a scalable, low-diameter network architecture that is independent of the number of layers and has a maximum shortest distance between any pair of servers that cannot exceed four.

[2] Data centre networking aims to connect a large number of server machines while requiring little in the way of capital expenditures, while also offering a high network capacity and high bisection breadth. It is common knowledge that the way servers are currently connected through a tree architecture of network switches cannot satisfy these needs. [3] Many big data applications depend on data aggregation; for instance, during the shuffling phase of a map-reduce assignment, data from multiple source racks (mappers) must be combined into one or more specified racks called aggregators (reducers) in the data centre network. In this study, we investigate methods for data aggregation to two aggregators in a data centre network under the restriction that data must be transported from a source rack to each aggregator via a single path.

Energy consumption makes up a sizable portion of operating costs in modern data centres. Only workload- or thermal-based job distribution amongst computing servers is the focus of current data centre energy optimization research. The communication fabric's impact on data centre energy use is highlighted in this research, which also introduces the DENS scheduling method, which combines network awareness and energy efficiency. The DENS technique strikes a balance between a data center's energy usage, employee productivity, and traffic demands. The suggested method maximises the trade-off between traffic pattern distribution and job consolidation (to use fewer compute servers) (to avoid hotspots in the data centre network).

THE PARALLEL ARCHITECTURES

The two types of parallel architectures—complete architecture and incomplete architecture—can be created by m-port switches and 2-port servers.

In order to speed up the processor, pipelining method is added. Pipelining is a method for carrying out the final function in parallel. Using the pipelining technique, high speed is possible.

A type of computation known as parallel computing involves running several calculations or processes concurrently. Large issues are typically divided into smaller ones that can be resolved all at once. Bit-level, instruction-level, data, and job parallelism are some of the several types of parallel computing. Due to the physical limitations prohibiting frequency growth, parallelism, which has long been used in high-performance computing, is now attracting more attention from the general public. Parallel cloud computing, mostly in the form of multi-core processors, has emerged as the dominant paradigm in computer architecture as power consumption and, subsequently, heat generation by computers have been a concern in recent years.

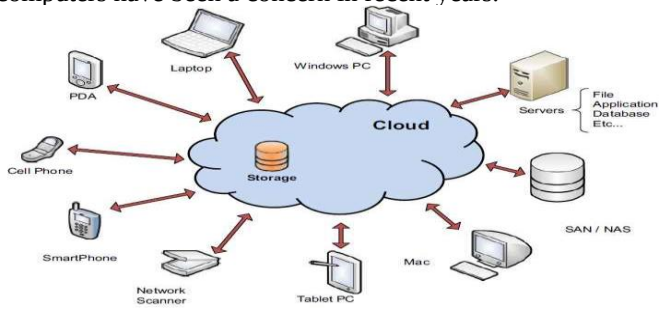

Fig 1 User Devices or Application of Cloud Storage

HSDC ARCHITECTURE

HSDC is a hypercube network-based system. 2-port servers and inexpensive commodity m-port switches make up the HSDC. After determining the internode distance that is the shortest, we create a fault-tolerant routing method for the

https://doi.org/10.37896/pd91.5/91556

ISSN: 0369-8963

HSDC. Any vertex in the network can use the routing method to build a path between any two vertices. With the addition of the High availability Algorithm, we additionally examine the topological characteristics of HSDC.

The architectural model that describes a system design's structure, behaviour, and other aspects. It is made up of a number of modules that will operate in concert to implement the whole system. Process owner, user, create storage, port, and search module are among the modules that are included.

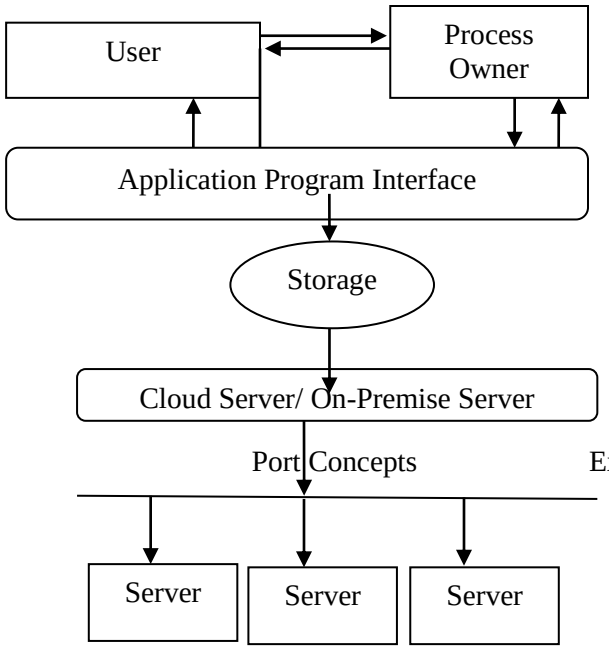

Fig 2 High Scalable Data Center (HSDC) Architecture

End user and process owner authentication is present in the module. It is capable of adding new users to our application interface, registering them, and enabling them to do other functions. It might be able to generate an end user and separate their user role at the same time. Normally, admin has the authority to perform this kind of function.

The registration data that the administrator can create can be stored in the user module. The end user can be used to refer to these users. They can take the whole game infrastructure for applications and other things. The event handler can carry out this action.

The end user's storage allocation can be served by the storage module. This might be the centralised application component. Only our major process begins here. Once the user has access to the designated storage, he or she may be able to carry out the necessary tasks for using the application.

The port module specifies how a server can carry out the necessary operation and switch from one server to another. Moreover, how the files that the end user registered were managed. It might be able to adjust the general application state as well as the current circumstance.

The system makes it possible for a data owner to effectively share his data with a defined group of users who adhere to a sharing policy. The data will retain its searchability, but the related search keyword(s) can also be altered after the data sharing. Users obtain the searchable keyword from the owner and use specific keywords to search documents, returning the results to the user.

Exceed Storage Limit

Complete HSDC Architectures

A graph is typically used to represent an interconnection network, with the vertices being processors and the edges denoting links residing between processors. A set V of vertices and a set E of directed edges make up a graph, where V = (V, E). A subset of the elements (u, v) of V V is the set E. If (u, v) E, (v, u) E, then E is said to be symmetric, in which case the undirected edge is represented by these two opposing arcs (u, v) and (v, u) (u, v). If all of a graph's edges are undirected, the graph is said to be undirected.

We define an m-dimension hypercube Hm as follows since the entire HSDC architecture is built using hypercube networks.

Definition 1. In H_m, defines the vertices and edges are as follows:

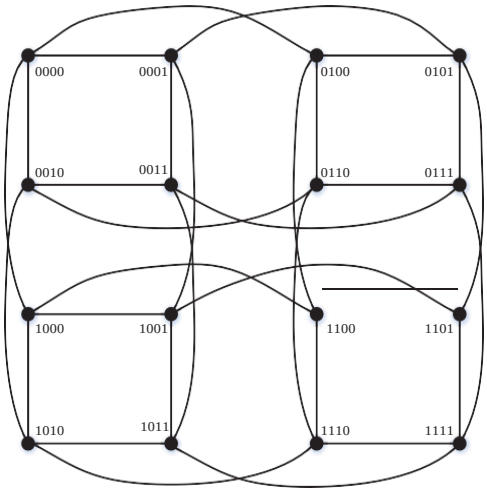

Fig 3 HYPERCUBE H4

https://doi.org/10.37896/pd91.5/91556

ISSN: 0369-8963

i) The vertices are identified as $(x_m \ldots x_1)$;

ii) The edges are defined as $((x_m \ldots x_1),(x_m \ldots x_{y+1} x_y x_{y-1} \ldots x_1))$;

where x_i 0, 1 , 1 y m and x_y the complement of x_y.

Definition of Complete HSDC Architecture

The HSDCm(n) architecture is built utilising m-port switches and is based on the n-dimension hypercube.

Since every server port in the complete HSDC architectures is fully utilised, the HSDC architecture built using m-port switches has achieved its maximum number of servers, and we are unable to add any more servers to it.

We first define the term "full HSDC architecture" in this section. We suggest three different types of incomplete HSDC architectures after thoroughly examining the topological characteristics of the complete HSDC architecture. Based on the topological characteristics of the full architecture, the topological characteristics of the incomplete architectures are inferred.

Algorithm 1: The routing algorithm for $HSDC_m(m)$.

```
1  Function Route((x_m...x_1; y), (u_m...u_1; z))
2      if d == 0 then
           ; /* The vertices n and t is in the same block, and
            the path can be constructed in the block.      */
3          if y! = 0 then
4              if Valid(x_m...x_1; 0) then
5                  return (x_m...x_1; 0)
6              else
7                  return (x_m...x_y...x_1; y)
8          else
9              return (x_m...x_1; z)
10     if d >= 1 then
           ; /* The vertices n and t is in different block, and
            the path will make the hamming distance d
            reduce to zero.                                */
11         if y! = 0 then
12             if Valid(x_m...x_y...x_1; y) then
13                 return (x_m...x_y...x_1; y)
14             else
15                 return (x_m...x_1; 0)
16         if y == 0 then
17             p = VDB(x_m...x_1, u_m...u_1)
18             return (x_m...x_1; p)
```

Routing Algorithm

The routing algorithm is a crucial communication method for any network of connections. In this part, we propose a fault-tolerant routing method for HSDCm(m), as illustrated in Algorithm 1, to handle switches or servers failing in a real-world simulation. This approach is fully distributed, therefore it can be applied to any vertex n = (xm...x1; y) and may quickly identify the following unflawed vertex on the path.

The hamming distance between xm...x1 and um...u1 is represented by the parameter d. The function Valid(xm...x1; y) returns false if the vertex (xm...x1; y) has a bug; otherwise, it returns true. The server (xm...x1, $\overline{p}$) is not defective, and the switch (xm...x1, 0) executes the function VDB(xm...x1, um...u1) and returns a different bit of p between xm...x1 and um...u1.

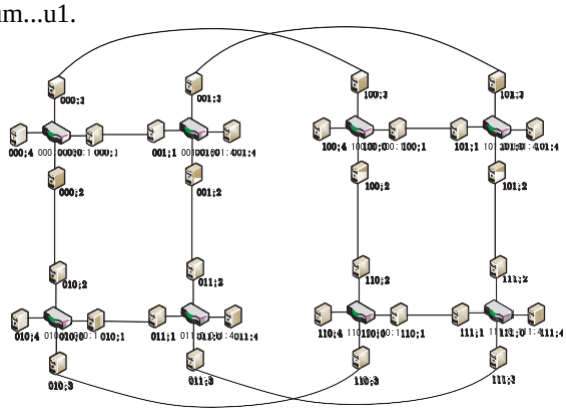

Fig 4 Hamming Distance

Algorithm 2: The algorithm for $HSDC_m(n)$.

```
1  Function InCom Routing1((x_n...x_1; y), (u_n...u_1; z))
       /* Except the special case of y > n or z > n, the path
          can be constructed in HSDC_m(n)
          according to algorithm Route.                    */
2      if y > n then
3          return (x_n...x_1; 0)
4      else if z > n then
5          Route((x_m...x_1; y), (u_n...u_1; 0))
6      else
7          Route((x_n...x_1; y), (u_n...u_1; z))
```

COMPARISON

The number of switches and links, scalability, diameter, and bisection breadth of the topologies of the Fat-Tree, DCell, BCube, and FiConn have all been thoroughly investigated in [8], [9], [14], and [10]. According to the number of switches, edges, diameter, bisection width, and scalability, HSDC is compared to various well-known data centre network architectures as Fat-Tree, DCel-l, BCube, and FiConn in Table 1. The performance of a data centre network design is significantly impacted by the characteristics used to do the comparison.

A measurement metric for network communication latency could be diameter. One of the desirable characteristics of an interconnection network is low diameter. The minimum number of edges that must be eliminated for a network to be divided into two identically sized sections is known as the bisection width. High network capacity and a more resilient structure against faults are implied by a wide bisection. The quantity of switches and links can be used to calculate the cost of building a data centre network when there are a set number of servers in the architecture. We also provide a simulation to assess how much money and energy HSDC uses in comparison to other standard data centre network architectures.

We use the assumption that all five structures support the same number of servers (N) in data centres and make use of the same kind of switches in order to conduct a fair comparison (m ports).

https://doi.org/10.37896/pd91.5/91556

ISSN: 0369-8963

Diameter

The diameter of the Fat-Tree is 2log2N, which is equal to two times its height. DCell and FiConn's exact diameters are currently unknown. These two structures' upper bounds are both 2logmN 1. The diameter of BCube, which is equal to logmN, is the smallest of these server-centric systems. The diameter of the HSDC, which was examined in Section 3, is 2 m, making it relatively modest but significantly larger than the diameters of DCell and FiConn. Applications having real-time requirements can be supported by the HSDC.

Cost and energy consumption

In this part, we contrast HSDC's cost and energy usage with those of the other data centre network architectures when we employ them to build a network with an identical number of servers. We only consider the cost and energy consumption of the switches and NICs because the overall cost and energy consumption of the servers are the same.

SIMULATION RESULTS

A. Memory Pipelining

Figure 5 demonstrates how four inputs—datain1, datain2, datain3, and datain4—along with four address lines, four read/write pins, and four erase pins are stored simultaneously in memory. If the write/read pin is set to 0, writing has occurred; otherwise, reading has occurred.

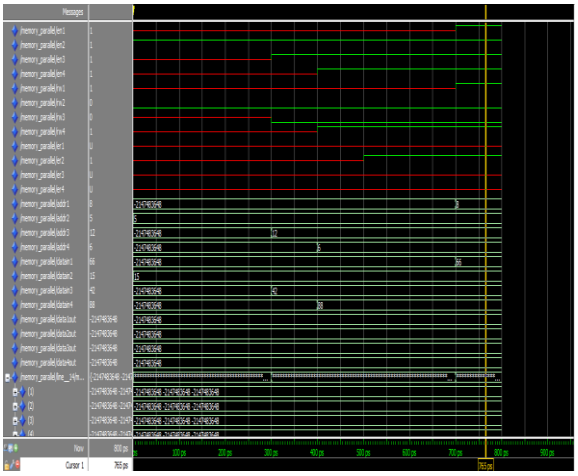

Fig 5 Memory Pipelining

B. File Handling

In Fig. 6, file handling manages the inputs stored in the system path and produces the output based on the user input file that was previously stored.

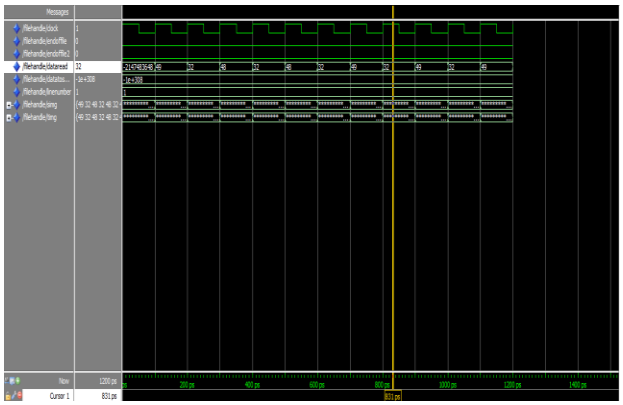

Fig 6 File Handling

CONCLUSION

In this study, we present HSDC, a brand-new hypercube network-based data centre network architecture. 2-port servers and inexpensive commodity m-port switches make up the HSDC. We create a fault-tolerant routing strategy for the HSDC after determining the shortest internode distance. 2-port servers and inexpensive commodity m-port switches make up the HSDC. Create a fault-tolerant routing strategy for the HSDC after determining the shortest inter-node distance. Any vertex in the network can use the routing method to build a path between any two vertices. Offer the HSDC a brand-new class of algorithm dubbed High availability. the HSDC's topological characteristics once the High Availability Algorithm was added.

This idea can greatly benefit from the high availability algorithm in terms of lowering downtime. Each data storage unit can be set up as a port so that several checks can be made on the storage and storage space can be allocated to store the data. Can also avoid downtime in the vast majority of situations. Use the fault tolerance algorithm in existing systems. It should be taking care of each component, minimising errors, preventing incorrect communication, and showcasing the high availability technique that was just presented.

REFERENCES

[1] T. Hoff, “Google architecture,” http://highscalability.com/ google-architecture, 2008, online document accessed on June 28, 2008.

[2] L. Rabbe, “Powering the yahoo! network,” http://yodel.yahoo.com/2006/ 11/27/powering-the-yahoo-network, 2006, online document accessed on June 28, 2012.

[3] Y. Zhang and N. Ansari, “On architecture design, congestion notification, tcp incast and power consumption in data centers,” *IEEE Communica- tions Surveys & Tutorials*, vol. 15, no. 1, pp. 39–64, 2013.

[4] Y. Deng, “What is the future of disk drives, death or rebirth?” *ACM Computing Surveys*, vol. 43, no. 3, p. 23, 2011.

[5] S. Ghemawat, H. Gobioff, and S.-T. Leung, “The google file system,” in *ACM SIGOPS operating systems review*, vol. 37, no. 5. ACM, 2003, pp. 29–43.

[6] J. Dean and S. Ghemawat, “Mapreduce: simplified data processing on large clusters,” *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[7] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, “Dryad: distributed data-parallel programs from sequential building blocks,” in *ACM SIGOP- S Operating Systems Review*, vol. 41, no. 3. ACM, 2007, pp. 59–72.

[8] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu,

 https://doi.org/10.37896/pd91.5/91556

ISSN: 0369-8963

[9] "Dcell: a scalable and fault-tolerant network structure for data centers," in *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4. ACM, 2008, pp. 75– 86.

[10] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "Bcube: a high performance, server-centric network architecture for modular data centers," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 63–74, 2009.

[11] D. Li, C. Guo, H. Wu, K. Tan, Y. Zhang, S. Lu, and J. Wu, "Scalable and cost-effective interconnection of data-center servers using dual server ports," *IEEE/ACM Transactions on Networking*, vol. 19, no. 1, pp. 102– 114, 2011.

[12] D. Li and J. Wu, "On data center network architectures for interconnect- ing dual-port servers," *IEEE Transactions on Computers*, vol. 64, no. 11, pp. 3210–3222, 2015.

[13] T. Wang, Z. Su, Y. Xia, J. Muppala, and M. Hamdi, "Designing effi- cient high performance server-centric data center network architecture," *Computer Networks*, vol. 79, pp. 283–296, 2015.

[14] Z. Li and Y. Yang, "Gbc3: A versatile cube-based server-centric network for data centers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 10, pp. 2895–2910, 2016.

[15] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *ACM SIGCOMM Computer Communi- cation Review*, vol. 38, no. 4. ACM, 2008, pp. 63–74.

[16] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Vl2: a scalable and flexible data center network," in *ACM SIGCOMM computer communication review*, vol. 39, no. 4. ACM, 2009, pp. 51–62.

[17] G. Qu, Z. Fang, J. Zhang, and S.-Q. Zheng, "Switch-centric data center network structures based on hypergraphs and combinatorial block designs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 1154–1164, 2015.

[18] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano *et al.*, "Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 183–197, 2015.

[19] R. D. S. Villaca, R. Pasquini, L. B. De Paula, and M. F. Magalhaes, "Hcube: Routing and similarity search in data centers," *Journal of Network and Computer Applications*, vol. 59, pp. 386–398, 2016.

[20] D. Li, J. Wu, Z. Liu, and F. Zhang, "Towards the tradeoffs in designing data center network architectures," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 1, pp. 260–273, 2017.

[21] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: Networking data centers, randomly." in *NSDI*, vol. 12, 2012, pp. 17–17.

 https://doi.org/10.37896/pd91.5/91556