Performance Analysis of Fattree Data Centre Networks (DCN)

Pavan Kumar Mangipudi
Electronics and Computer Engineering
University of Florida
Gainesville,FL

Arunabho Basu
Electronics and Computer Engineering
University of Florida
Gainesville,FL

Naga Venkata Sai Varanasi
Electronics and Computer Engineering
University of Florida
Gainesville,FL

***Abstract*—Data centre networks (DCN) play a pivotal role in the operation and inter networking of a Data Centre. There are certain challenges and network performance requirements in order for the DCN to be able to provide seamless connectivity and inter networking for a scalable number of servers and applications that use the Data centre. Of these hurdles, the one of scalability and traffic congestion remain the area of focus of this project. This project implements a Fat tree DCN model and analyses the said challenges by comparing and contrasting them with some of the well known generic parameters of DCN's such as delay and throughput of the network.**

***Index Terms*—Fat tree, Data centre Network, Scalability, Traffic**

I. Introduction

A. Data Centre Networks

Data centres are facilities comprising of networked computers and storage that businesses and organizations use to organize, process, store and disseminate large amounts of data. Although generally referred to as a single entity, they are composed of a number of technical elements such as routers, switches, security devices, storage systems, servers, application-delivery controllers and more. All these constituents of the Data Centre are controlled and interconnected through the Data Centre Network. So, Data Centre Networks (DCN) are a form of interconnection network that facilitate the networking and interconnections between various peripherals of the Data Centres. As can be anticipated, DCN's are presented with several challenges. Some of them include Scability, Reliability and Performance, Security, Energy consumption. To meet these challenges there has been a lot of research into various DCN architectures as shown in Fig.1. Broadly, DCN architectures can be classified into two categories: Fixed and Flexible topologies.Examples of the Fixed topologies are Basic Tree, Fat-tree, Clos Network, D-cell, Bcube. In this Project, we focus on the Fat tree architecture of DCN's 1

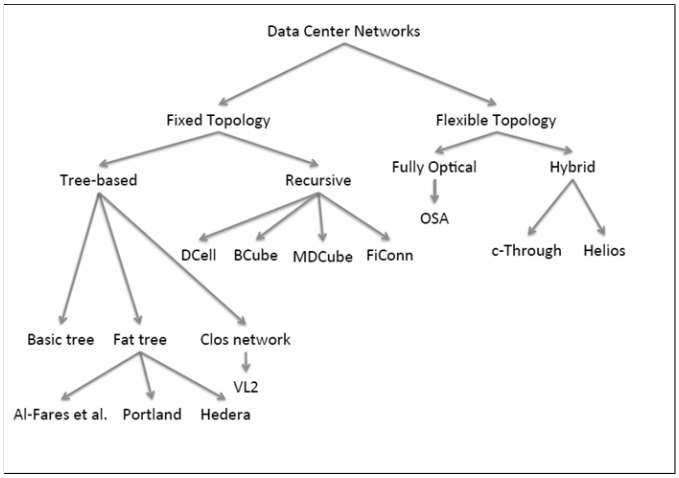

Fig. 1. Types of DCN topologies

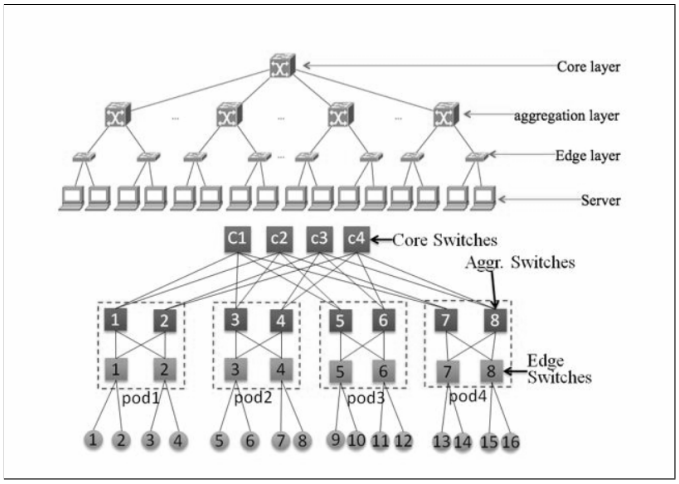

Fig. 2. Fat tree topology

B. Fat Tree Architecture

Fat tree, as the name suggests is a type of tree architecture. This is mainly used to connect a large number of physical servers/ computers in a large data centre. These were originally introduced by Charles Leiserson in 1985 [2]. The idea behind fat-trees is to alleviate the bandwidth bottleneck closer to the root with additional links. In a tree data structure, every branch has the same thickness, regardless of their place in the hierarchy—they are all "skinny" (skinny in this context means low-bandwidth). This problem is overcome in fat tree architecture by having thicker ('fatter') branches (data links) closer to the top of the hierarchy, than those further down. This variation in the thickness (bandwidth) of the branches allows for more efficient and technology-specific use. The topology is as shown in Fig 2.

C. Approach and Objectives

In this project, we aim to analyse a select few performance characteristics and parameters of the DCN. We do this by implementing a scalable Fat tree DCN with a given number of nodes, servers and applications. The parameters or performance metrics we are concerned about are scalability, Traffic Handling and Optimal network. scalabality refers to the effect of a change in the number of applications of the throughput and delay. Traffic handling is concerned with the performance of the DCN when the application traffic is changed for a constant number of servers and how this change effects the delay and throughput across the network. And lastly, optimal network attempts to get the optimal number of servers and switches for which the given network performs the best.

II. Reference Paper

The following publication is used as a reference paper for this project. [1] Towards Reproducible Performance Studies of Datacentre Network Architectures Using an Open-source Simulation Approach by: Daji Wong, Kiam Tian Seow, Chuan Heng Foh, Renuga Kanagavelu,. Published 9-13 Dec. 2013 in '2013 IEEE Global Communications Conference (GLOBECOM), 6 pages. Its abstract is as follows: [1]

The major data centre network research challenges include performance analysis of scalable DCN architectures. This is will support in making wise datacentre investment decisions. NTU-DSI-DCN initiative is based on ns-3 on which performance models of DCN are made.Models of fattree and BCube are implemented and comparetive performance study is reported.

The paper conducts simuations on both Fat tree and Dcell architectures of DCN's and compare various performance parameters and characteristics of the two architectures. In this project we have tried to implement simulations of only the Fat tree architecture.The paper conducts the following performance studies: (1) Average throughput of the Fat tree and the BCube architectures. (2) Average packet delay of the Fat tree and the BCube architectures. In our project, we have implemented similar performance studies. Instead of conducting them for both fat tree and Dcell, we have only done it for Fat tree architecture. We have compared the Average packet delay and the Average Throughput with both the number of servers and number of applications each, so that we get scalability performance characteristics by varying the number of servers, and traffic handling performance when comparing the delay and throughputs with number of applications keeping the servers constant.

III. Simulation description

In this section we will show how we designed and implemented the fat-tree architecture similar to the one described in the previous section using many features available in NS3 network simulator. We will also show the custom experiments created to analyse the performance of Data Centre Network replicating the real world applications. Sections III-A–III-B below show the experimental setup and results respectively.

The reason for selecting NS3 as simulator is due to its wide usage in the research community and also due to its flexibility in design parameter selection along with inbuilt functions for various features. These features include creating custom connection between any nodes using point2pointhelper, using routing protocols like Nix-vector and more. It also have wide variety of inbuilt applications like UDPEchoserver which deals with establishing connection using UDP protocol between nodes.

A. Experimental setup

For performance study of the fat-tree data centre network we used NS3 as simulator on Intel i7 CPU with 2.0GHz and 8GB RAM running on Ubuntu 16.04 OS. We simulate the fat-tree similar to the design discussed before, but for fair understanding of its performance in different conditions the design we need to realize has to be flexible. So the two main design criteria considered are flexibility in tree structure and proper application with relevant performance metrics.

The flexibility we realized in our experiment consists of two types, they are flexibility in tree structure and flexibility in traffic management. The former states that our fat-tree has to be scalable with single parameter change and the later gives the freedom to change number of applications used to increase/decrease traffic in the network.

Table I shows the simulation settings used for our experiments. The application used is OnOffApplication, this application creates an alternate On and Off traffic between two nodes selected. The On-time and Off-time varies according to random variable selection. The traffic generated during the On-state is according to given data-rate and packet-size. We have used 1024 bytes as packet size and 1Mbps as data rate for our application. The maximum datarate we defined between any two devices in network is set to be 1000Mbps. Our application needs two nodes to create the traffic the selection of these nodes/servers is achieved using random variable selector.

Fig 3 shows the algorithm used to achieve the above mentioned goals. Our algorithm contains three nested loops, the first is for flexibility in structure, where we used number of ports as the parameter which defines the structure. The tree creates number of servers based on number of port each switch has using the formula K*K*K/4 where K is number of ports. The second loop is for changing the number of applications acting on one structure to create the traffic, we looped over 2-680 number of applications. Since our application uses random variable based node selection, the metrics which application outputs always changes and is not constant, this is the reason we have used another loop to experiment application multiple times to generate the average performance.

Final step we have to consider before starting the experiment is to decide which parameter to measure and what experiments have to be conducted. NS3 provides wide variety of options to select various metrics, of which we have selected throughput and delay time . We simulated three different problem scenarios which closely resembles the issues we face in realtime data centre networks. These include scalability, optimal network

- Initialize variables
- Loop over number of ports
  for (ports, ports < max_ports ; ports = ports + 1)
- Loop over number of applications
  for(applications, applications<max_applications)
- Loop over iterations
  for(iterations,iterations<5)
  1. Creation of Node Containers
  2. Initialize settings for On/Off Application
  3. Connect hosts to edge switches
  4. Connect edge switches to aggregate switches
  5. Connect aggregate switches to core switches
  6. Start Simulation

Fig. 3. Simulation algorithm

TABLE I
SIMULATION SETTING USED FOR FATTREE DCN

Attribute	Value
Number of pods	K(2-12)
Number of nodes	K*K*K/4(2-432)
Type of application	On-Off Application
Number of applications	(2-680)
Simulation run time	100s
Packet size	1024 bytes
Data rate for packet sending	1Mbps
Data rate for device channel	1000Mbps
Communication pair selection	Random selection
Traffic flow pattern	Random variable traffic
Routing protocol	Nix-vector

structure and traffic handling capacity. Further details about the problem scenarios are discussed in Section III-B

B. Results and analysis

The three experimental scenarios and settings are as follows:

- Scalability: In scalability experiment we intended to observe how scalable the fat-tree structure can be, i.e, how the throughput and delay time changes when structure and applications increase linearly .
- Optimal network: In this experiment we tried to find the optimal network structure, i.e, number of servers and switches to consider to give better performance for given traffic conditions.
- Traffic handling: In critical conditions a data centre network may face heavy amounts of traffic which it can't handle. We tried to find how its performance degrades during such conditions.

The sample values of the output of scalability test are shown in first part of Fig. 4. In this test we linearly increase number of servers and number of OnOffApplications used on the respective fat-tree structure. We calculated the average throughput and time delay occurred due to the application (note: the average is due to the multiple iterations conducted on the same structure and same number of applications). When we plot number of servers Vs average throughput and delay the graphs are as shown in 5. Average delay slowly increases at the initial stages of scaling and it increases exponentially as the number of servers increase. Average throughput has a steady decrease as the number of servers increase. The initial

No. of servers	No. of applications	Avg Delay(s)	Avg Throughput(Mbps)
2	2	0.000338725	65.1494
31	31	0.025547	58.2742
85	85	0.120674	53.0437
182	182	0.391555	47.2925

No. of servers	No. of applications	Avg Delay(s)	Avg Throughput(Mbps)
2	85	0.0702875	10.1726
85	85	0.286395	41.7202
843	85	0.658294	45.607
1228	85	0.750793	46.0767

No. of servers	No. of applications	Avg Delay(s)	Avg Throughput(Mbps)
85	8	0.00368198	61.6214
85	85	0.0844657	51.7799
85	340	0.342658	25.3192
85	680	1.75425	19.924

Fig. 4. Output values

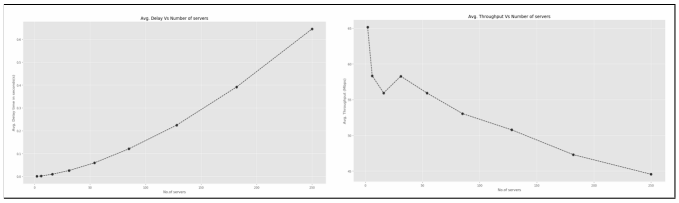

Fig. 5. Scalability performance

dip in the throughput values in graph are due to error caused by random selection of nodes. At initial stages as we considered less number of applications, experiment tends to create heavy traffic in specific part of structure which causes that error.

The sample values of the output of optimal network test are shown in second part of Fig.4. Here we kept number of applications constant and increased the number of servers. The graphs obtained for average delay and throughput are shown in Fig.6. Average throughput rises heavily till some point and there is slowly increasing(almost constant) value after that point. where there is heavy increase in average delay initially and there is linear increase after some point.

The sample values of the output of optimal network test are shown in third part of Fig.4. We increased the number of applications on single tree structure with constant number of servers. This resulted in performance shown in Fig.7. Average delay has heavy effect on performance after some point of traffic, average delay also decreases heavily and goes to almost zero at high traffic levels.

The conclusions we can derive from our individual experiments are as follows: scalability measure shows that performance of fat-tree networks degrades heavily at higher

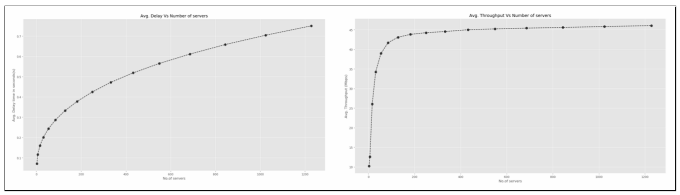

Fig. 6. Optimal Network performance

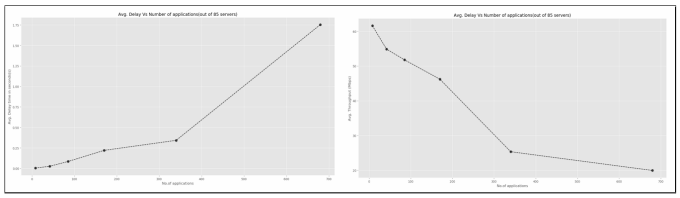

Fig. 7. Traffic handling performance

levels of tree structures, this is because of increase in number of hops between application nodes.Optimal performance test shows at initial parts where number of servers is very less compared to number of applications it is alright to increase the structure as trade-off due to increase in throughput value is more than increase in delay value, but after a point the increase in throughput is very less but delay increases for which increasing the network complexity only degrades performance. This behaviour is due to decrease in number of collisions and increase in number of hops for application. The traffic handling performance shows fat-tree can handle slight variations in sudden traffic rise but fails to perform when heavy traffic occurs, this is due to increase in number of collisions and throughput decreases heavily.

The paper we selected for this project mainly deals with difference in performance of BCube and fat-tree data centre networks, where they did not consider different problem scenarios but conducted experiments for large scale data centre networks to find average difference between Bcube and fat-tree. Our experiment used the structure of Fat tree similar to that used in paper with few modifications to output which included problem specific scenarios. One of that include analysis on traffic handling capability. We got similar performance compared to that of paper for similar testing conditions. Fig.8 shows results of paper, it shows relation between number of servers, throughput and delay. Our project performance is more focused on initial performance of network rather than overall performance as showed in paper.

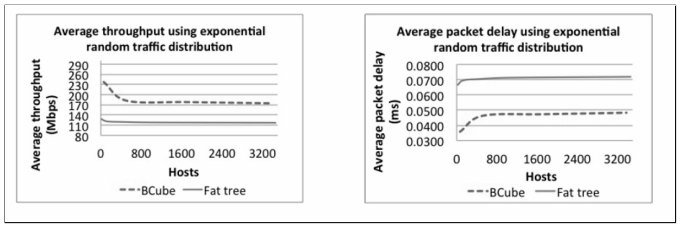

Fig. 8. Results in research paper

CONCLUSION

During our project we explored multiple versions of code. Initially we conducted experiment for only 16 servers with user defined constant IP address for the nodes and switches. This program helped us understand about various features of NS3 more. The flexibility of coding in C++ and many inbuilt tools and functions available made the work easy. Then we expanded our code to dynamically change the structure and traffic.

We faced difficulties during the initial stages of our project as NS3 doesn't have proper support for windows systems and the linux versions effected the packages to be installed for NS3. After we figured the proper configuration changing the code for our requirement has been great learning experience where we used concepts we have learned in class to simulation. The major problem we faced is during the end of our project where use of random variable class for OnOffApplication random selection did not work and we changed to the version of NS3 tried to add the randomvariable.h file to our source to run for our programm.

REFERENCES

[1] Daji Wong, Kiam Tian Seow, Chuan Heng Foh, Renuga Kanagavelu, "Towards Reproducible Performance Studies of Datacentre Network Architectures Using an Open-source Simulation Approach". in '2013 IEEE Global Communications Conference (GLOBECOM)' ,pp. 1373-1378 December 2013

[2] Leiserson, Charles E (October 1985). "Fat-trees: universal networks for hardware-efficient supercomputing" (PDF). IEEE Transactions on Computers. 34 (10): 892–901. doi:10.1109/TC.1985.6312192.