2016 IEEE 22nd International Conference on Parallel and Distributed Systems

Modeling Traffic of Big Data Platform for Large Scale Datacenter Networks

Zhen Xie*†, Zheng Cao*, Zhan Wang*, Dawei Zang*, En Shao*, Ninghui Sun*
*Institute of Computing Technology, Chinese Academy of Sciences
†University of Chinese Academy of Sciences, China
{xiezhen, wangzhan, zangdawei, shaoen, cz, snh}@ncic.ac.cn

***Abstract*—Prior to deployment, network designers often use simulators to pre-evaluate the performance of designed network with artificial network traffic. The traditional way of separating network design from real applications will not only result in over-designed network configurations, wasting money and energy, but also miss the real network demands of applications, degrading system performance. In this paper, we provide a method to model the network traffic of current popular big data platforms, which can observably improve the matching between network design and applications. The new method extracts communication behavior from the popular big data applications and replays the behavior instead of the packet traces. Experiments show that the traffic generated by the model is almost match the real traffic and the model can easily scale to thousands of nodes.**

***Keywords*-modeling traffic; big data platform; datacenter; hadoop; storm;**

I. Introduction

The interconnection network in datacenter significantly impacts the systems overall performance and cost, so a comprehensive evaluation of the network design is badly demanded prior to actual deployment. Traditionally, the evaluation is done through simulation, which comprises two key components: workload and simulator. Workload refers to the traffic that is supplied to the network terminals over time, and simulator refers to the certain detail model those analogs the actual networks behavior. From network designers perspective, workloads should reflect real applications traffic patterns in order to guarantee the accuracy of evaluation results. Now, big data analysis has become one of the most important applications in datacenter, so a well-constructed accurate big data workload is very important for the simulator to evaluate the performance of designed network.

For the sake of development and management efficiency, big data applications are often developed based on several big data framework, such as Hadoop, Storm and so on. Due to the inherent feature, the applications based on same framework have similar network communication pattern. Based on above consensus, this paper focuses on modeling the network behaviors of specific framework instead of certain applications.

Our main contributions are summarized as follows:

1) We propose a new methodology to model the traffic pattern of big data frameworks. It combines function fitting and behavior simulating to model the traffic phases and patterns.
2) We collect the traces of several big data frameworks and analyze their communication pattern in detail. In addition, we give a traffic model for each framework based on the extracted pattern.
3) We evaluate some topologies by simulation to verify the accuracy and scalability of our model. In the simulation, four topologies, including FatTree, HyperX, FB-Clos and TorFT, are created and some detail comparisons between real deployment performance and simulation performance are given.

The rest of the paper is organized as follows. Section 2 describes related work. Section 3 elaborates our traffic modeling methodology. Section 4 gives the complete traffic model construction of two representative frameworks: Hadoop (MapReduce) and Storm (Stream computing). Section 5 realizes the traffic model with a network simulator, and compares the result with real deployment. Section 6 evaluates several mainstream topologies with our workloads. Section 7 concludes the papers.

II. Related Work

One way to create network workload is using full-system simulation. For example, we can use QEMU to run real applications and inject the network packets to a network simulator.However, full-system simulation takes large amounts of CPU and memory resources, it is not practicable to use it to simulate a large scale data center.

Another way to create workload is trace replay. By running specific applications on an existing network topology and using collection tool to preserve the traffic at runtime, the collected traffic can be replayed in the network simulator. Related works of trace replay include Owezarski [3] and Sheldon [4]. However, for the reason that traffic patterns of applications are affected by network characteristics, such as topology and routing algorithm, trace replay would generate irreconcilable results with different network parameters.

Synthesis is a more widely used method to generate network simulation workloads. Junior method is generating several constant traffic patterns, such as broadcast, all-to-all uniform random, and is always used to evaluate a designs upper limit. Senior method is synthesizing workload according to real applications history traffic. Microsoft [5] and

1521-9097/16 $31.00 © 2016 IEEE
DOI 10.1109/ICPADS.2016.36

Facebook [6] adopt distribution fitting model which has been previously shown to be useful to capture the behavior of conventional network workloads.In prior work[7], Feitelson aparts from characterizing network loads and presents an overview of methods to model network request distributions. He presents preliminary considerations on the correlation between task size, arrival rate and execution time when combining network and CPU modeling without presenting any validation of the proposed techniques against actual applications. Lidescribes a method to characterize network and CPU-intensive applications running on large-scale grids[8]. It analyzes features like job arrival rate, size, pseudo periodicity and correlation of these features with execution time. This method is a feasible way to fitting various traffic of datacenters.But in real evaluation,it still meets some problems: the construction of artificial network method is global average traffic or function fitting, and as real data center network traffic is not always average random or consistent function, this leads to the authenticity problems of the existing artificial traffic. In addition, the function fitting method based on the historical data of the network is running on a certain scale data center node, when adding more network nodes and deploying on different topologies, fitting function will have deviation with real deployment. Y.Joo [9] proposes network traffic modeling method to identify and resolve performance bottlenecks in a small machine cluster. They compare two different models, an infinite source-based model that is user-invariant, i.e., all users send same amount of data, and a SURGE-based model, where traffic varies per user. They observe that ignoring user variation and information about network topology causes significant inaccuracies in the generated workloads.

Through this study, source-based model and SURGE-based model are inaccuracies on a larger scale, so we must alter current modeling methods to achieve scalability requirements. While application-based workloads give us the most accurate modeling of the network, it is often difficult achieve a through coverage of expected traffic with these method exclusively. Alternatively, a molding method which merges application-based and fitting can capture the demands expected for date center network, while also remaining flexible.

III. Network Modeling Methodology

Due to the inherent feature, the applications based on same framework have similar network communication pattern.In this section, we analyze the network transmission behavior on the coarse-grained platform-level and the finer-grained application-level respectively. . Specifically, we aim to answer the following questions: (1) what are the aggregate communication characteristics of phases, patterns, and packet distribution for platform-level? (2) what are the particular characteristics, such as execution time and data size for application-level in same platform.

Our detailed modeling methods are the following:

1. Platform communication model

Platform Phases: Big data platforms are often composed of different phases to support corresponding computing. Our model extracts platforms phases, which is obligatory to be covered within any given applications in all the data centers, even in the worst case where application instances are too adequate and spread across racks rather than confined within a rack. For instance, the MapReduce applications seem relatively higher similarity owing to they all need to run map and reduce functions. From this, by using the collected data and analysis of the platform, we build the execution times and transmission sizes of each function for the platform. These function models can help generate the traffic phases, given the platform patterns used.

Platform Patterns: A few recent proposals [5, 6] have discussed for network patterns and hotspot to more effectively engineer data center traffic. Our model analysis the communication patterns of platforms, namely, all-to-all, many-to-many or peer-to-peer, that centralized approaches could achieve parallel and authentic data transmission while supporting the application traffic phases which we observe in the first step, and heuristics to scale the size of data centers.

Packet Distribution: We need examine the distribution of packet sizes in the studied platforms which will be modeled for generating real-size packet. For example in Hadoop, the packet sizes exhibit an average pattern, with most packet sizes clustering between 200 bytes and 1400 bytes. Surprisingly, we found application keep-alive packets as a major reason for the small packets, with TCP acknowledgments, as expected, being the other major contributor. In this case, through the results of platform phases and platform patterns method, we use the ”packet distribution” method to simulate the real packet.

2. Application communication parameters

Different applications running on the same platform have general similarities, but because of different processes and data sets, there also exist limited differences, we need to analysis applications source code, obtain difference of applications characteristics, and change and adjust platform communication models with special communication parameters. Phases Parameters Instantiation: Since the front platform phases method has built the communication phases for platforms, we also need analyze the process of the real implementation of applications and test the actual network traces to instantiate the non-configured parameters in models. In addition, the phases parameters can be achieved by the function fitting or event-driven method. Finally, by profiling the application types and data sets, this instantiated platform models can obtain application-based communication phases under any conditions.

Patterns Parameters Instantiation: Application affect the communication pattern of platform in a certain degree because instances will be deployed in distribution node. We

recognize that there may be other applications may or may not share all the patterns parameters that we have observed. So our work points out that it is worth closely examining the different patterns, as there are important differences for applications.

Next we will apply our modeling approach to the two kinds of representative big data platforms, and in the subsequent, instantiate communication parameters by the two popular applications. Finally, the correctness and extensibility of the network simulator will be proved.

IV. Platforms and Applications Analysis

As our network traffics are generated by specific platforms and applications, these specific network traffics also depend on the type of platforms and applications. We comprehensive analysis the network interfaces and the communication mechanisms of platforms and applications, merge the platform characteristics and the application run-time difference, and finally, recreate traffic on specially platform and application.

The current widely used data center calculation framework: Hadoop [10] and Storm [11]. Hadoop is represented by MapReduce with high reliability, high scalability and high fault tolerance. Mahout [12] is implemented by Hadoop, including the clustering, classification, recommend filter, frequent items mining and retrieval functions. In real-time streaming computing field, as a distributed and fault-tolerant real-time computing system, Storm guarantees each message will be processed, and it can be deployed soon in a cluster, which can handle millions of messages per second. Storm Email Benchmark [13] as typical open source application of real-time mail processing, the system relies on statistics to distinguish mail, usually be used for preprocessing subsystem in large junk mail handling system. The following will set up the platform communication model of the two big data platforms and achieve application communication parameters of applications.

A. Hadoop and Web Search Benchmarks

1) Hadoop Platform Communication Model: According to the description of our understanding in the relevant Hadoop papers, The process of Hadoop is divided into four phases: Input, Map, Shuffle and Reduce. The DAG can be consisted of a dozen functions or operator compositions, namely block, split, map, spill, combine, reduce and etc. Fig.1 is the schematic diagram of the Hadoop framework. According to the characteristics of the Hadoop execution stages, and we detailed analyze four phases (shown in Fig.1):

Input phase (T1): the preparation stage for data and programs, data exchange will produce in unbalance distribution.

Map phase (T2): the Map function execution phase, at this time, it requires a large number of CPU computing without network transmission.

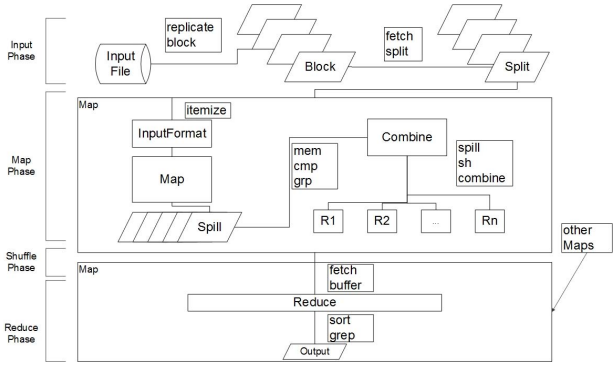

Figure 1. Schematic diagram of the Hadoop framework

Shuffle phase (T3): the stage starts after the Map stage finished, temporary file transfers to the Reduce nodes which need data, and preparing for the Reduce stage.

Reduce phase (T4): Reduce function process stage, at this time, it produces a large number CPU computing, no network transmission. At this point, Reducers integrate with the data and store it in the disk, preparing for the next job.

Then, the transmission component in Hadoop is Netty which mainly occurs in the Shuffle stage, so we analyze the traffic transfer mechanism in the Shuffle phase, and construct traffic communication model by simulating the behavior for Netty in Hadoop platform.

As Hadoop includes a number of mappers and reducers, every map task generates an intermediate result file, each intermediate result file contains more than one partition, the partition number is equal to the number of the reduce task. Each mapper need transfer intermediate results to the additional reducer, so the communication pattern is a kind of manytomany communication relationship. A completed platform model will combine above-mentioned methods.

2) Web Search Benchmarks Communication Parameters: For web search benchmarks work load, PageRank [14] and Nutch Indexing [14] represent two of the most important system implementations (that is, large-scale search index system). PageRank workload uses a test case-SmartFrog (an open source framework) to test data of the distributed management system. It is an open source implementation of the web page ranking algorithm, and widely calculates web link based on the number of reference links for the web pages. PageRank workload is constituted with a series of works. Hadoop job will iterate until constraints of convergent condition were satisfied. The process of program execution is shown in Fig.2. Nutch indexing subsystem is a popular open source search engine workload (Apache). We use the subsystem in the Nutch crawler to climb an internal mirror of Wikipedia and generate about 2.4 million web pages of input data, the program execution process is shown in Fig.3.

Table I
AVERAGE RESULT OF WEB SEARCH TEST

Phase / Node Number	Data size (GB)	Stage1				Stage2			
		Input phase (T1)	Map phase (T2)	Shuffle phase (T3)	Reduce & output (T4)	Input phase (T5)	Map phase (T6)	Shuffle phase (T7)	Reduce & output (T8)
2	100GB	41	477	996	113	55	620	1087	65
4	100GB	21	238	501	55	28	310	547	33
8	100GB	10	122	248	28	14	156	274	16
8	500GB	55	613	1240	146	72	787	1379	82

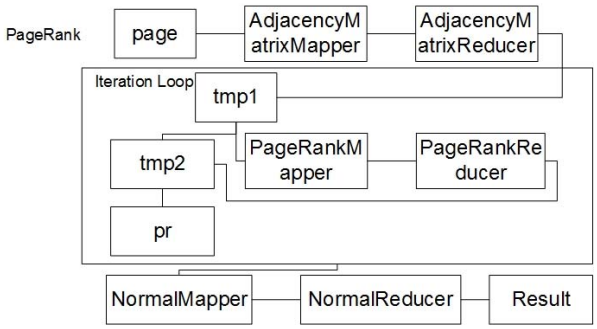

Figure 2. Implementation process of PageRank

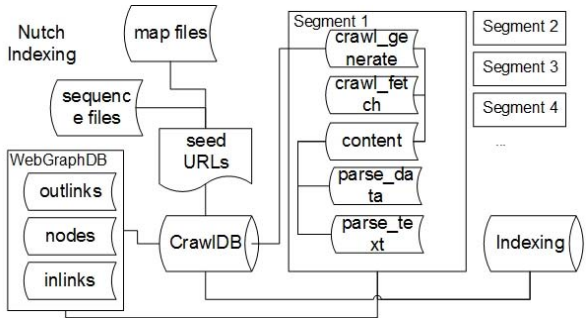

Figure 3. Implementation process of Nutch Indexing

Through the 9000 groups of test for 100G and 500G input data on 2 4 8 nodes respectively, average result is showed in Table1.

We fit the five stages of Hadoop, and we found that correlation coefficient between T1-T8, data size (D) and node number (N) is higher, so we use the linear fitting method of fitting the stages of execution time. The results are as follows:

The Shuffle data quantity of PageRank and Nutch Indexing as follows:

By the time fitting, we verify time function in the larger scale, different nodes and data size, and then we found tolerance is below 5

B. Storm and IBM Storm Benchmark

1) Storm Platform Communication Model: Storm implements a data flow model in which data flows continuously through a network of transformation entities (see Figure 1). The abstraction for a data flow is called a stream, which is an unbounded sequence of tuples. The tuple is like a structure that can represent standard data types (such as ints, floats, and byte arrays) or user-defined types with some additional serialization code. Each stream is defined by a unique ID that can be used to build topologies of data sources and sinks. Streams originate from spouts, which flow data from external sources into the Storm topology.

The sinks (or entities that provide transformations) are called bolts. Bolts implement a single transformation on a stream and all processing within a Storm topology. Bolts can implement traditional things like MapReduce functionality or more complex actions (single-step functions) like filtering, aggregations, or communication with external entities such as a database. A typical Storm topology implements multiple transformations and therefore requires multiple bolts with independent tuple streams.

Compare with Hadoop, Storm framework traffic is closely related with applications topologies.

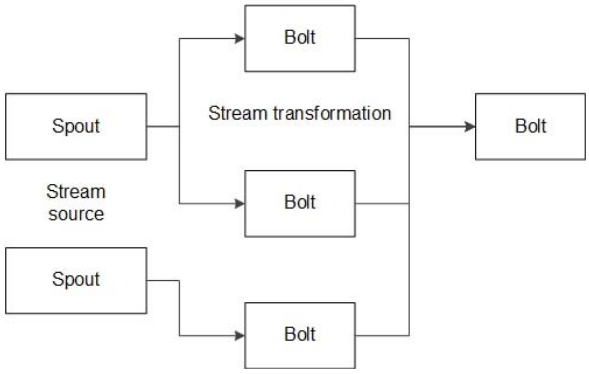

Figure 4. Components of the Storm framework

2) IBM Storm Benchmarks Communication Parameters: According to the different function, this mail classification system benchmark process [13] is divided into seven stages, and uses some strategies to optimize parallel data trans-

mission. As mentioned previously, this process realized the function of email classification by statistics. For example, the word and character count in the email body represents the overall situation of this emails content, which helps to distinguish spam and normal. The results of the email classification system can be used for spamming detection, the program execution process is shown in Fig.5. According

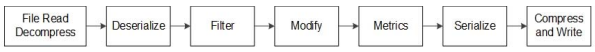

Figure 5. Implementation process of the IBM Storm Benchmark

to our analysis before, the Storm execution process is divided into two components: Spout and Bolt. The number of each component and relation are given in application source code, their respective operation also have contact with each other, each process send message by a mechanism called "Grouping" to constitute the data exchange.

According to analyze logical topology of the application, we find the connection relationship between the logical topology is determined, and component connection relationship is defined by "Grouping" mechanism. The mainly function file is EnronTopology.java. For example ReadEmailDecompressSpout is located at the head of the whole topologies, and it directly related with the external input data, which has a close relationship with the external input data size and rate of the application, therefore the size and rate of external input data set can be set as parameters in the model. There is still a similar relationship for all of the components in the following Spout. In order to get a greater performance, every Spout and Bolt in logical topology will be extended to physical topology. The process is shown in Fig.6.

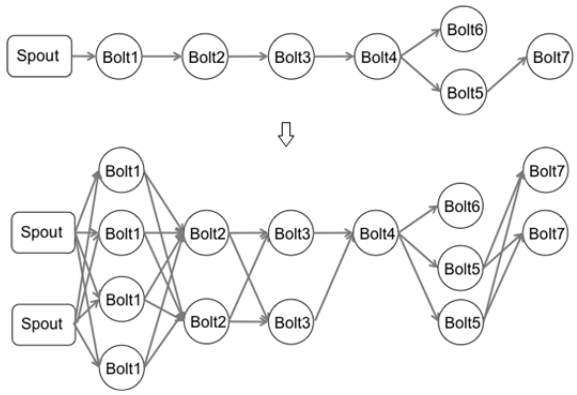

Figure 6. Convert logical topology into physical topology

Because each Spout/Bolt handle a tuple after the last tuple by way, and the time of processing a tuple is very short, so the traffic between Spout/Bolt to Spout/Bolt is very stable. In the processing process of each component, the size and rate of input and output are similar to linear correlation below the permitted processing power.

After the analysis above, we use traffic collection tools to obtain the size of traffic between any input and output in the logical topology, and find out the rate as parameter between two nodes. In our traffic model, for any condition, if we has the input size and rate, according to the DAG of logical topology in Fig.5 and the corresponding parameters in Table 1, we can completely calculate the output size and rate.

Storm exchange data between nodes by Netty, and Netty is a relatively simple data transmission mechanism. So we also realize the data transmission mechanism in the simulator, and according to the size and rate of input and output, replay the IBM Storm Benchmark execution process and network traffic transmission.

V. Network Traffic Simulator and Verification

A. Network Traffic Simulator

Through the traffic model above, we achieve a network traffic simulator. The network traffic simulator can configure node number, application type and data set size, and automatic generate network data package for network request.

Project is open source on GitHub.

https://github.com/ncsg123/Big-Data-Traffic-Simulator.git

B. Verification

By running Web Search Benchmark in simulator and physical machines with 1TB data set within 16 nodes, we compare the traffic generated by our simulator and the traffic collected by system capture tools. We find that the difference is very small. The comparison result showed in Fig.6. The calculated T1-T8 by a function in the 4.1 section are: 55, 621, 1271, 148, 73, 802, 1413 and 84. Experiments show that the standard deviation between those two types of traffic is 3% of the total amount of transmitted data.

VI. Performance and Proposal

A. Mainstream Topology Performance

Through the front of the network traffic simulator, and due to space limitations, we run two workloads in four different topologies, and test performance.

By contrasting the completion time of the test application in different topologies and scale, the stand or fall of topology could be judged. Under the condition with the total application data is unchanged, and the total amount is 100 GB. The two traffic generated by the simulator, and we compare the completion time under four different topology. As shown in Fig.7. The results show the application completion time with the node size decrease in four topologies.

By the performance test example in 6.1. Take the Hadoop test as an example, we can see that, when running Web Search 100G in the Hadoop, the network topology difference

Table II
INPUT AND OUTPUT RATE AND SIZE

Component	Output_Rate
Spout	Spout_N*SpoutOutputRate
Bolt1	(PreOutputRate<Bolt_N*BoltOutputRate)?PreOutputRate&Bolt1_N*BoltOutputRate
Bolt2	(PreOutputRate<Bolt2_N*BoltOutputRate)?PreOutputRate&Bolt2_N*BoltOutputRate
Bolt3	(PreOutputRate<Bolt3_N*BoltOutputRate)?PreOutputRate&Bolt3_N*BoltOutputRate
Bolt4	(PreOutputRate<Bolt4_N*BoltOutputRate)?PreOutputRate&Bolt4_N*BoltOutputRate
Bolt5	(PreOutputRate<Bolt5_N*BoltOutputRate)?PreOutputRate&Bolt5_N*BoltOutputRate
Bolt6	(PreOutputRate<Bolt6_N*BoltOutputRate)?PreOutputRate&Bolt6_N*BoltOutputRate
Bolt7	(PreOutputRate<Bolt7_N*BoltOutputRate)?PreOutputRate&Bolt7_N*BoltOutputRate
Component	**Input_Size**
Spout	0
Bolt1	Readinputspout_N*EmailPartsize
Bolt2	Readinputspout_N*EmailPartsize*0.997
Bolt3	Readinputspout_N*EmailPartsize*0.997*0.723
Bolt4	Readinputspout_N*EmailPartsize*0.997*0.723*0.98
Bolt5	Readinputspout_N*EmailPartsize*0.997*0.723*0.98*0.996*0.053
Bolt6	Readinputspout_N*EmailPartsize*0.997*0.723*0.98*0.996
Bolt7	Readinputspout_N*EmailPartsize*0.997*0.723*0.98*0.996*1.035
Component	**Output_Size**
Spout	Readinputspout_N*EmailPartsize
Bolt1	Readinputspout_N*EmailPartsize
Bolt2	Readinputspout_N*EmailPartsize*0.997*0.723
Bolt3	Readinputspout_N*EmailPartsize*0.997*0.723*0.98
Bolt4	Readinputspout_N*EmailPartsize*0.997*0.723*0.98*0.996
Bolt5	Readinputspout_N*EmailPartsize*0.997*0.723*0.98*0.996*0.053*0.428
Bolt6	Readinputspout_N*EmailPartsize*0.997*0.723*0.98*0.996*1.035
Bolt7	Readinputspout_N*EmailPartsize*0.997*0.723*0.98*0.996*1.035*0.0425

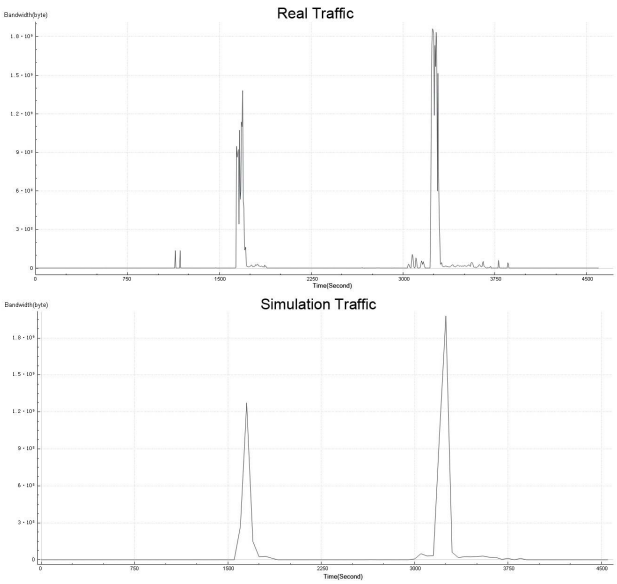

Figure 7. Comparison between real traffic and simulation traffic

is not obvious on 1024 nodes. Communication performance rising along with the network scale, and the application completion time into ”inverse log characteristic”. As Hadoop Job uniform distribution characteristics, the greater network size brings the lower the degree of polymerization. For the application, Performance in turn is HyperX, Fattree, FB-Clos and TorFT. So when the size is in 128 nodes, we recommend Fattree or HyperX. Recommend the HyperX or FB-Clos select in 128-432 nodes. In more than 432 nodes, four kinds of topologies are in the same weight.

B. Scalability Test for Network Congestion

In different size of the network scalability testing, the application of big data transfer a fixed amount of data, the large-scale network will have the problem of uneven distribution of network traffic, leading to network deviation evaluation results in large-scale network.Therefore, in the test of network-oriented scalability for network traffic performance, the amount of data transmitted is proportional to the growth of the network scale.At the meanwhile, the experiment will be performed in parallel by setting a multi-application, making the network been in a state of saturation, so that network congestion phenomenon will been demonstrated in a evident way. Through experiments, we found that ten applications running in parallel way in the network will create full load phenomenon.

From the phenomenology of the test we can see the decreasing of the maximum reception bandwidth of four networks with the growth of network size , so the average utilization of network links gradually decreased too in Fig.

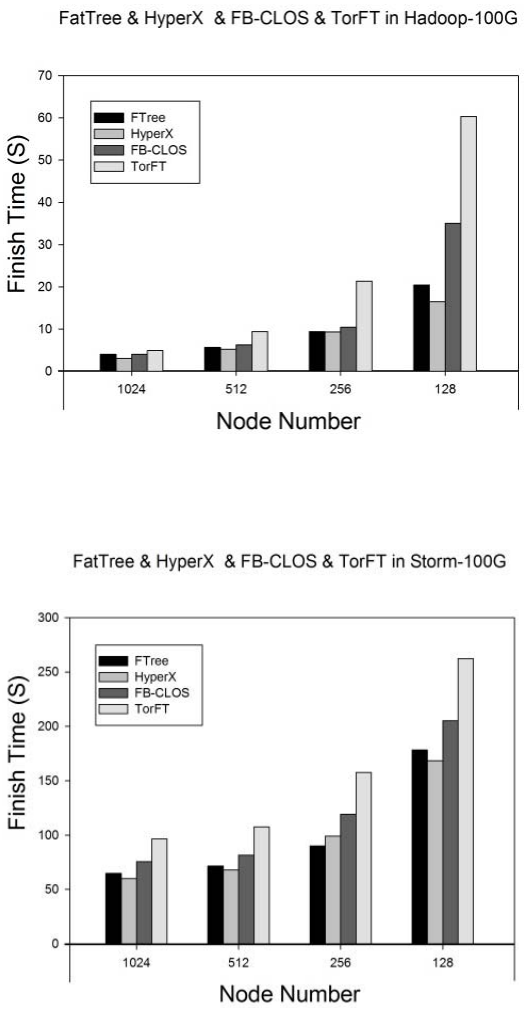

Figure 8. Finish time in different topologies for Hadoop and Storm

8. Network Communications differentiating features still exist, from the test data of four different network. From a performance point of view, with the network size increases, the FB-clos(70% to 46% to 25%) communication performance decrease faster than Fattree (83% to 75% to 37%). The performance of FB-clos under the node size of one hundred thousand, still lower than Fattree about 10% in link utilization. Due to link issues of reduction, TorFT maintain a high degree of congestion. Especialy under one hundred thousand size, the link utilization only reached the 10% of the whole bandwidth , the network congestion phenomenon of this kind of network is most obvious. In summary, during scalability test for network congestion of four mainstream topology, the performance sort of those topology is ”Fattree

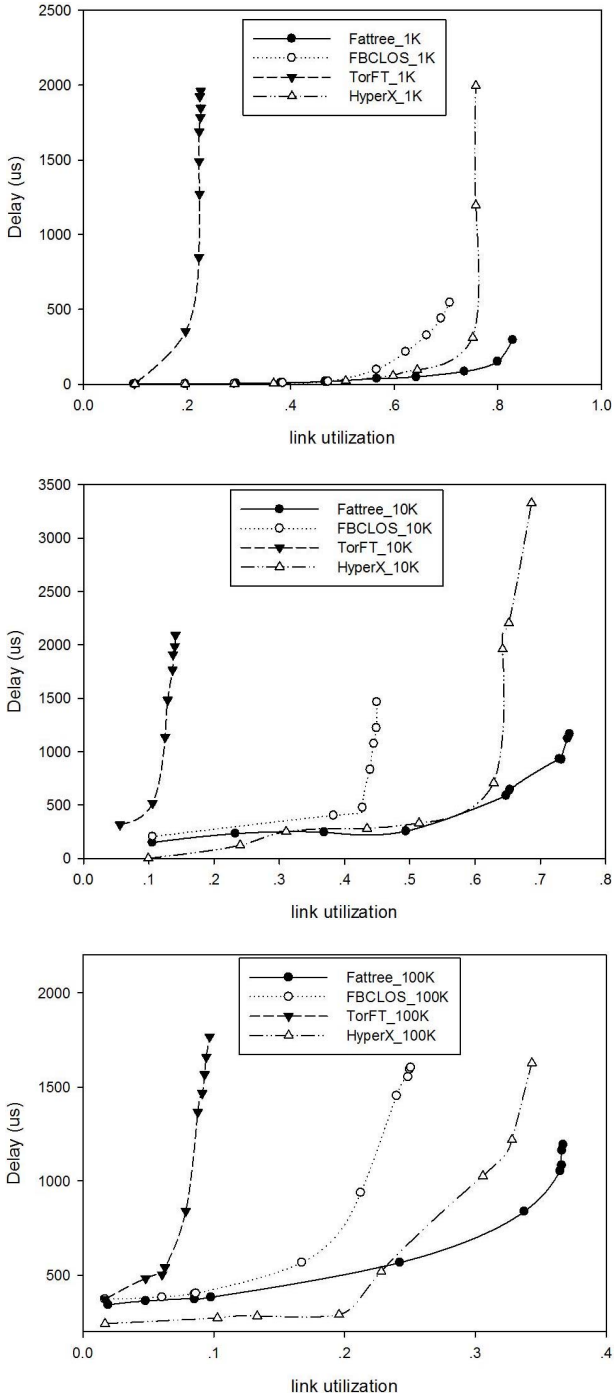

Figure 9. Link utilization in scalability test

>HyperX>FB-clos>TorFT". The Fattree may have the best proformance than others.

VII. Conclusions

In conclusion, we put forward a network simulator which based on platforms and applications, and other related work including platform and application analysis, rather than a function. This work may be useful to network design and deployment. And it can greatly improve the real application and network topology matching. Network model is in accordance with the actual operating behavior of applications data, so the model can completely simulation actual traffic.

Network traffic modeling is an advanced network analysis method, and the theoretical research of this field continues to improve, in this case, in addition to tracking and reference network traffic modeling in the field of the latest research results, traffic modeling more need combine the actual network production environment which based on large amount of network traffic statistics, the practicability and validity of network traffic model also need to be improved. By gradually refining and improving the range of the application and modifying the existing network traffic model, traffic model can be used in more places. To further improve the ability of network data collection and analysis, and extract the original operating data from all levels network devices, seek the breakthrough by using data analysis method, find the traffic data statistical rule, and build the real network traffic model. The model also needs execute more testing, analysis more applications, deploy more network topology to improve applicability.

VIII. ACKNOWLEDGMENT

This paper is supported by the National Natural Science Foundation of China (NSFC) under Grant No.61572464 and No.61331008.

References

[1] QUME homepage: http://wiki.qemu.org/Main_Page

[2] ns-3 homepage: https://www.nsnam.org/

[3] Owezarski, Philippe, and Nicolas Larrieu. "A trace based method for realistic simulation." Communications, 2004 IEEE International Conference on. Vol. 4. IEEE, 2004.

[4] Sheldon M, Weissman G V B. Retrace: Collecting execution trace with virtual machine deterministic replay[C]//Proceedings of the Third Annual Workshop on Modeling, Benchmarking and Simulation (MoBS 2007). 2007.

[5] Christina Delimitrou, Sriram Sankar, Aman Kansal, Christos Kozyrakis:ECHO: Recreating network traffic maps for datacenters with tens of thousands of servers. IISWC 2012: 14-24

[6] A. Roy, H. Zeng, J. Bagga, et al. Inside the Social Networks (Datacenter) Network. In Proc. ACM SIGCOMM, 2015.

[7] D. Feitelson. Metric and Workload Effects on Computer Systems Evaluation. In Computer, 36(9):18-25, 2003.

[8] H. Li . Realistic Workload Modeling and Its Performance Impacts in Large-Scale eScience Grids. In IEEE TPDS, vol. 21, no. 4, 2010.

[9] Y. Joo, V. Ribeiro et al. TCP/IP traffic dynamics and network performance: A lesson in workload modeling, flow control, and trace-driven simulations. In Proc. of SIGCOMM, 2001.

[10] Hadoop - Apache Software Foundation project home page. http://hadoop.apache.org/

[11] Storm, distributed and fault-tolerant real-time computation. http://stormproject.net/

[12] Mahout - Apache Software Foundation project home page. http://lucene.apache.org/mahout.

[13] IBM Research Dublin, IBM Software Group Europe WHITE PAPERS, https://developer.ibm.com/streamsdev/wp-content/uploads/sites/15/2014/04/Streams-and-Storm-April-2014-Final.pdf

[14] Huang S, Huang J, Liu Y, et al. Hibench: A representative and comprehensive hadoop benchmark suite[C]//Proc. ICDE Workshops. 2010.