Saint Petersburg State University of Information Technologies, Mechanics and Optics
Department of Telecommunication Systems
PERCCOM Master Program

By
Georgiou Stefanos Istvan

Implementing Green IT approach for transferring Big Data over Parallel Data Link

Examiners: Examiner 1 (to be defined by PERCCOM consortium)
Examiner 2 (to be defined by PERCCOM consortium)

Supervisors: Professor Andrey Y. Shevel
Professor Eric Rondeau

20 of May 2015

Saint Petersburg State University of Information Technologies, Mechanics and Optics
Department of Telecommunication Systems
PERCCOM Master Program

By
Georgiou Stefanos Istvan

Implementing Green IT approach for transferring Big Data over Parallel Data Link

Examiners: Examiner 1 (to be defined by PERCCOM consortium)
Examiner 2 (to be defined by PERCCOM consortium)

Supervisors: Professor Andrey Y. Shevel
Professor Eric Rondeau

20 of May 2015

This thesis is prepared as part of an European Erasmus Mundus programme PERCCOM - Pervasive Computing & COMmunications for sustainable development.

This thesis has been accepted by partner institutions of the consortium (cf. UDL-DAJ, n°1524, 2012 PERCCOM agreement).
Successful defense of this thesis is obligatory for graduation with the following national diplomas:

- Master in Master in Complex Systems Engineering (University of Lorraine)
- Master in Pervasive Computing and Computers for sustainable development (Lulea University of Technology)
- Master of Science in Technology (Lappeenranta University of Technology)

Abstract

The whole research of the current Master Thesis project is related to Big Data transfer over Parallel Data Link and my main objective is to assist the Saint-Petersburg National Research University ITMO research team to accomplish this project and apply Green IT methods for the data transfer system. The goal of the team is to transfer Big Data by using parallel data links with SDN Openflow approach. My task as a team member was to compare existing data transfer applications in case to verify which results the highest data transfer speed in which occasions and explain the reasons. In the context of this thesis work a comparison between 5 different utilities was done, which including Fast Data Transfer (FDT), BBCP, BBFTP, GridFTP, and FTS3. A number of scripts where developed which consist of creating random binary data to be incompressible to have fair comparison between utilities, execute the Utilities with specified parameters, create log files, results, system parameters, and plot graphs to compare the results.

Transferring such an enormous variety of data can take a long time, and hence, the necessity appears to reduce the energy consumption to make them greener. In the context of Green IT approach, our team used Cloud Computing infrastructure called OpenStack. It’s more efficient to allocated specific amount of hardware resources to test different scenarios rather than using the whole resources from our testbed. Testing our implementation with OpenStack infrastructure results that the virtual channel does not consist of any traffic and we can achieve the highest possible throughput. After receiving the final results we are in place to identify which utilities produce faster data transfer in different scenarios with specific TCP parameters and we can use them in real network data links.

Keywords: Big Data, Linux, Utilities, transfer, network, cloud.

Acknowledgments

I would like to express my gratitude to my advisors Professor Andrey Y. Shevel for his continuous guidance, support, and patience during the whole research. As always he was very helpful giving me a lot of information with a quick response to all my emails and he always took time to make every question I had as clear as possible. Special thanks also to Prof. Eric Rondeau for all the advices about how to keep the topic close to the research field. In addition , I would like to thank Prof. Jari Porras for his guidance, valuable help and knowledge, insightful comments and being always present to answer my questions also for my thesis report he gave me the most crucial steps how to prepare a professional looking Master Thesis Report. Also about the Big Data team of ITMO State University which was there to help me and guide me with any question I had.

Last but not least, I would like to thank my family and all my friends for their support and significant help during one of the hardest periods in my student life. For all my friends who motivated me and be by my side for every step I took. Special thanks to Baptiste Louis, Alexandre De Masi, Iqbal Ahmed, Fisayo Caleb, Rohan Nanda, Ana Kalisnik, Constandinos Fasoulis, Theodoros Anagnostopoulos, Marat Akhmadeev, Nenad Zoric, and Elena Shkuratova. During this period also some lectures who inspired me and motivated me to become better and more professional in my field of study, my deepest gratitude for each one of them. Many special thanks also to all the members of PERCCOM and to also for those who organized the whole program, it was a life-time experience that I will never forget. The knowledge and memories I have acquired though the whole time of 2 years by visiting different countries as France, Finland, Sweden and Russia will stay always in my mind as the biggest gain for accomplishing this Master Program. For last I would like to thank Erasmus Mundus program for giving me this opportunity and unique life experience to study into 4 different countries.

Table of Contents

List of Symbols and abbreviations

PERCCOM	PERvasive Computing and COMmunication for Sustainable Development
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
FDT	Fast Data Transfer
FTS	File Transfer Service
VM	Virtual Machine
RTT	Round Trip Time
MSS	Maximum Segment Size
LAN	Local Area Network
WAN	Wide Area Network
LHC	Large Hadron Collider
HDD	Hard Disk Drive
K,M,G,T	Kilo, Mega, Giga, Terra
IP	Internet Protocol
SOA	Service Oriented Architecture
OS	Operating System
QoS	Quality of Service
CERN	European Organization for Nuclear Research
I/O	Input/Output
SSH	Secure Shell
BASH	Bourne Again SHell
FTP	File Transfer Protocol
CLI	Command Line Interface
DTN	Data Transfer Node
SDN	Software Defined Network
PS	Post Script
PDF	Portable Document Format
NAT	Network Translation Protocol

RSA	Ron Rivest, Adi Shamir and Leonard Adleman
GUI	Graphic User Interface
RAM	Random Access Memory
NFS	Network File System
VCPU	Virtual Central Processing Unit
ACK	Acknowledgement
RWIN	TCP Receiver Window
ARQ	Automatic Repeat-Request
RoCE	RDMA over Converged Ethernet
RDMA	Remote Direct Memory Access
SCP	Secure CoPy
Data Link	By the term data link, in the related thesis work is meant from one point of the Internet to another point of the Internet, and not from point to point connation.
ITMO	Information Technologies, Mechanics and Optics
PNPI	Petersburg Nuclear Physics Institute located 40Km away from ITMO server
TLS	Transport Layer Security
ACL	Access Control Lists
RTT	Round Trip Time
AQM	Active Queue Management

1 INTRODUCTION

At this part of the thesis an introduction is given about the topic, motivation and purpose of the current thesis work. Delimitations are also expressed and have been described in details and finally a thesis structure is given where brief information can be found what each part includes.

1.1 Background

The purpose of this Master Thesis Project is to implement green IT approach for the Big Data Transfer over Parallel data link by using tools and methods that we have learned during our PERCCOM Master Program from different Universities. As it is widely spread from most recent results presented by climate scientists alarming, the greenhouse gas (GHG) in the atmosphere is growing faster than predicted and the need to reduce the emissions is even more essential. Scientists, economists and policy makers are calling for emissions target of at least 20% below 1990 levels in 2020 as mentioned in the Smart 2020 report from (The Climate Group, 2008). To keep the environment saves and healthy is responsibility for all of us. Thus the necessity to develop Green Pervasive and Sustainable Systems is even more crucial nowadays.

The topic main concern is the Big Data (Beal, 2015) which is a huge and complex set of data which makes it hard to process; analyze it by using on-hand database management tools (White, 2012). The world's technological per-capita capacity to store information has roughly doubled every 40 months since the 1980s as mentioned (Hilbert, and Lopez, 2011) and as of 2012, every day 2.5exabytes (2.5*10^18) of data were created (Taylor, 2011). Also challenging action is to store, maintain, search, share, transfer and visualize these Big Data sets. Big Data has a variety of usage and huge impact in our lives.

To achieve our goal we will run our test bed Big Data System and compare it with different existing Big Data Systems based on multiple parameters and using different utilities in the context of data transfer. Our main aim of comparison is to transfer as much data volume is possible in a short period of time. Generally speaking there are many hi-speed network of 10GBits or 100Gbits/second, but our aim is to transfer Big Data to less conventional University connectivity for this project, and also testing transfers in Virtual environment as the cloud which is one of the most evolving topics nowadays. As well known that data transfer speed over network depends on many parameters but we will focus on specific ones which are in the Transport Layer and more specifically at the TCP protocol which is responsible to provide reliability, in-order data and error check delivery as mentioned from (Hunt, 2002). After using those parameters conclusion is necessary to define which (parameters) and why are affecting our transfer taking into account the scenarios and the data transfer applications which we are going to use. Large variety of Linux tools have been used to accomplish this task by any means. As mentioned above the lowest level tools used in these Master Thesis are the TCP. Energy consumption has essential impact on this project because Big Data Systems are running for a long period of time until the "job is done", and sometimes 24/7. The current project is focused on Green IT which has an aim the sustainable development. In case to achieve sustainability as explained by (Drouant, Rondeau, Georges, Lepage, 2014) in the current work experiments will we executed to identify the most optimal parameters for the data transfer applications since using a large amount of resource will not always transfer datasets faster but it will consume needlessly system resources.

1.2 Goals and Delimitations

Goal of this Master Thesis Project is to test the test bed implementation which is developed at ITMO Saint-Petersburg State University in order to transfer huge volume of data as fast as possible from the source to destination (source has 2 servers). Both source and destination are running Linux Red Had more details about the scenarios and implementation will be given at chapter 3. As mentioned above in LINUX systems there are numerous network parameters (watch parameters `/sbin/sysctl -a | grep net | wc -l`) which can be modified. The test bed was implemented by a team of researchers here in ITMO State University of Saint Petersburg and one of my objectives is to work close with the team to offer my knowledge and my help to achieve the necessary results and conclusion for the above project.

Main objective is to focus on the transport layer, and because reliability and data in order which is important for Big Data Systems our target of modification is TCP protocol (Hunt, 2002). However for Big Data the network link parameters are changing with the duration of time, and in addition the network link (channel) bandwidth is shared with other tasks and users. As a result of that link parameters will constantly change through the duration of time, in that case TCP protocols parameters also need to change to adapt to the new network (link) specifications.

A future aim of the current project as mentioned is to transfer Big Data over parallel data links. For the current implementation we are focus to transfer over single data link of 1Gbps bandwidth, in case to receiver deeper knowledge about the data transfer applications and their implementation.

Some of the most important TCP parameters for TCP turning as mentioned by (Pillai, 2013) which are taken into account for this thesis are window size, packet loss parameters which are dependent at throughput and RTT and maximum segment size (MSS). Final expectation from the whole project is to test different utilities and suggest which work better for each case/scenario and explain why we have this behavior regarding different scenarios.

For the Big Data transfer file we are using five different Utilities (More detailed information about the utilities and how to use them are given in chapter 3):

i) Fast Data Transfer known as fdt (FDT Team, 2013) which is a written in java and is capable to read and write at speed of disk in wide area networks.

ii) BBCP (Hanushevsky, 2015) is a utility for point to point network copying data written from Andy Hanushevsky at Slac as tool for BaBar collaboration. It is capable of transferring files at approaching line speeds in the WAN.

iii) BBFTP is file transfer software. It implements its own transfer protocol, which is optimized for large files (larger than 2GB) and secure as it does not read the password in a file and encrypts the connection information. (IN2P3 group, 2013)

iv) Globus Toolkit is an open source software toolkit used for building grids. It is being developed by Globus Alliance and many others all over the world. A growing number of projects and companies are using the Globus Toolkit to unlock of grids for their cause. (Grid Alliance, 2014)

v) FTS3 is a service responsible for globally distributing the majority of the LHC data across the WLCG infrastructure. Is a low level data movement service, responsible for reliable bulk transfer for files from one site to another while allowing participating sites to control the network resource usage. (Cern IT_SDC group, 2014)

In case to run all those utilities to transfer data a number of scripts/programs were created for that purpose. From all the executions data are need in case to compare the different scenarios that we are going to test, by collection log files and plotting some graphs which make it more visual for the research.

A large number of tests will be executed in purpose to get results to compare with different parameters of the utilities, for that reason if would not be efficient to have a normal server for this research and to execute scenarios one by one.

Cloud is one of the technologies which enter the scene as a main actor nowadays. Interesting fact is to compare what is going on in the virtual environment by using data transfer applications for Big Data before to test them in the global network of computers. For this case OpenStack which is a free and open-source cloud computing software platform as said by (Red Hat, 2014) was suggested. (More discussion about OpenStack later chapter) We have found this specific software easy to manage and handle many VMs, especially for our purpose. Since we can launch many "Instances" of VMs we can assign different job to different VMs and at the end extract the different results and compare then. Simultaneously we complete our tests faster using OpenStack and also using less energy and no infrastructure is needed. As mention in a PhD publication by (Guazzone, and Anglano, 2015) using Cloud infrastructure you can maximize the profit by minimizing the amount of violations of the QoS levels agreed with service providers and at the same time to lower the infrastructure cost, but taking into account that reducing QoS violations and reducing the energy consumption is a really challenging problem.

Delimitation we had during the testing is that not all of the utilities are using compression algorithms, some of them are using it and some do not. For testing purposed we would like to have more “fairness” between the test subjects and a solution has to be provided. A script has to be implemented which will create random uncompressible data where the user can define the destination where the data will created, directory size, file size, desperation and the block size (more details in Chapter 3.3.1). On the other hand since we cannot compress the random binary data we have to push on the link the raw data as it is which it would not be efficient and energy, but it can be considered as a trade-off for the research “fairness”.

Other delimitation we have is the specific number of parameters we can use while executing the current Utilities. As mentioned above in the current Chapter the number of TCP parameters in a LINUX system (Scientific Linux 6.5) is 649 but the most of the Utilities are accepting Number of Parallel Streams, TCP window size, sender and receiver buffer size. In case we would like to change more parameters we have to change them manually located in the path /proc/sys/net/ipv4 store them change in /etc/rc.d/init.d/network and restart every time and then execute again the testing scenarios which will be too much of time consuming procedure.

Also another reason which can delimitate our tests is the hardware resources. A primary thought was to create many senders and receivers to transfer and transfer to each other data, each sender was going to have one receiver, but having a single server to run all this tests we realized that this idea would be impossible because there are not many data lines on the server to transfer all those data and the results would not be clear to us. That is the reason why we decide to schedule all the transfer using a single instance at the time.

Most of the utilities as mentioned will be executed by the usage of scripts. Since it’s necessary when a user use a large amount of parallel TCP streams and relatively high window size the allocation of main memory which is needed is an equation of parallel streams multiplied with the window size. In case the main memory is not sufficient the scripts will hang there without giving any error message and a user may start a large script which may need days until it’s finished.

While testing and running the scripts in Virtual Environment other users can test and run different executions at the same time. In that case the executions may be affected by each other. In that case a good communications and scheduling would be really important.

1.3 Research Questions

Number of different research questions:

- Which is the behavior of different data transfer applications inside a overr virtual and real network channel.
- Which are the system parameters which affect the data transfer applications the most.
- Define the data transfer applications parameters which are most optimal and achieves the highest transfer speed in case to transfer Big Data.

1.4 Structure of the Thesis

This thesis is divided into five parts, which are structured as follows.

Introduction Part. In the first part called "Introduction", we provide the basic information and purpose of this Master Thesis Implementation and we explain why is an important topic especially nowadays. Definitions and main concepts are been defined also in this part but explained later on with more details. A brief description is given which solutions we provide to achieve the current implementation and what we expect to achieve. We define also the goals and the delimitations of the thesis, also narrowing the goal to make it more interesting for the readers.

Thesis Part. For the second part called "Thesis", we are giving explanation about a number of different citing and references which we found useful because they are giving scientific proofs and answers for our questions. At the end we provide a quick references summary from what we consider important to be used in our Implementation.

Implementation Part. For the third part called "Implementation", we are showing to the readers how we implemented the whole idea we had, in general the research methodology which was used. We are giving more in depth scientific, technical definitions and term about the tools, scenarios, tests. Important to explain is also why we did it using that approach and what we actually achieve by using different approaches, and why it was helpful.

Results and Discussions Part. For the fourth part of the thesis called "Results and Discussion", where we summarize the work described in this thesis, comparing the results we acquired, explain the results and giving answers to unanswered scientific questions, discuss features about the limitations of our evaluation.

Conclusion and Future Plans Part. For the last part of the thesis called "Conclusion and Future Plans", we are explain the whole concept of the thesis work, what we did and what we had achieve by working on this topic and how it can be used for further more research. A conclusion is given about which reason was that we gather the specific results and what can be done to improve them. Future plans are also been discussed on this part.

References Part. All the citing and references that was used for this thesis work written in Harvard **style** referencing.

2. THESIS

For this section of the thesis description is given about important paper which used as citing and referencing for the current work. At the end of this chapter a summary of the papers can be found.

2.1 Literature Review

Big Data: Many companies nowadays are using Big Data, the proper definition of big data is the data which exceeds the level of Giga Bytes (10^9B) produced daily. Although after a company produced such amount of data they have to store it, analyze it, search it, transfer it, visualize it and etc. Day by the day information and data size is increasing and it becomes challenge to handle all these data. In case to handle Big Data special tools are needed. Also it cannot be handled by using any of the most "relational database management systems" and desktop statistics and visualization packages, requiring instead "massive parallel software running on tens, hundreds, or even thousands of servers" as the author explains in (Beal, 2015). Big Data is also a crucial domain because of its usage fields. Many fields in the world require Big Data like spot business trends, determine quality research, prevent diseases, link legal citations, combat crime, determine real-time roadway traffic conditions.

Transport Control Protocol (TCP): For the current Master Thesis implementation TCP protocol stack will be our lowest level tools. As widely known TCP is one of the core protocols of the Internet Protocol suite (IP). TCP main advantage is to provide reliable data transfer through the network, ordered and error-checked delivery. Thus the load on the networks using TCP protocol the applications is higher than using UDP because of the packet header that TCP apply, UDP which is rather simple protocol but not reliable. Also TCP has numerous parameters which can be changed (/sbin/sysctl -a | grep net | wc -l). At first TCP protocol establish connection with the host and the server by using the three-way-handshake which send a connection request message and the receive replay with an acknowledgement (ACK) message to the sender and the sender replays back with another ACK message and the data transfer is begin. While receiving data TCP ACKs the data in case to avoid and packet lose by sending accumulated ACK not to load the link bandwidth further more. In case of message loss retransmission of the lost data will be done. Before ending the connection end connection request is send from the host to the server and an ACK message is received to by the host that the connections has been terminated.

Transport Control Protocol tuning: It's considered as the main mechanism, technique to adjust the network congestion avoidance parameters of a TCP connection for high-bandwidth, high-latency networks. As mentioned from (Pillai, 2013) a well-tuned networks can perform up to 10 times faster in some cases. Main attributes that we need to consider while tuning a network are: Bandwidth-Delay product (BDP), Buffers, Window size, and Packet Loss:

Bandwidth-Delay Product: term which is used in conjunction with the TCP as the number of bytes needed to fill a TCP "path". High performance network is equal to very high BDP. Networks with large BDP are also known as Long Fat Networks (FTNs). Bandwidth-Delay Product is the product of a data link's capacity (in bits per second) and its end-to-end delay as said by (Plankers, 2013), and it’s a simple way to calculate the RWIN size.

Buffers: For high performance network systems large buffers are required to handle delays in the system, original TCP configurations supported buffers up to 64KiB-1 receive window size. Thus buffer size need to be scalable depending on the amount of data arrived at the received node.

Window Size: also known as TCP Receive Window (RWIN) is the amount of data which the receiver can receive without acknowledging to the sender. In case the sender will not receive ACK for the first packet it stops and waits until a timeout event occurs and then retransmits. Important fact to mark is that even without packet loss windowing is a limit for the link throughput. Full network bandwidth sometimes may not been used because TCP waits for ACK messages before transmitting new data packets.

Packet Loss: When packet loss events are occurring in the network congestion avoidance algorithm is initiated. Depending on the congestion control algorithm the MSS will be modified and that most probably will cause limitation on the network performance.

Efficient Data Transfer Protocols for Big Data from: Brian Tierne, Ezra Kissel, Martin Swany, Eric Pouyoul, Lawrence Berkeley National Laboratory, Berkeley, CA 94270 School of Informatics and Computing, Indiana University, Bloomington, IN 47405 - As the world evolves the need of data is growing day by day and its necessary to use more efficient data movement protocols. Most tools which are used to move data are using TCP over sockets which gives the data flow limitation at 20Gbps. RMDA over Converged Ethernet (RoCE) is a high performance network data movement which is using minimum CPU. At the these paper the authors are comparing performance of TCP, UDP, UDT, and RoCE over high latency network paths of 10 and 40 Gbps, also shows how the Linux zero-copy system calls is improving the Transmission Control Protocol(TCP) performance on Intel "Sandy Bridge" PCI-Express 3.0 hosts. It is known that TCP suffer and have low performance over long-distance, high-bandwidth networks (Barakat, Altaman, and Dabbous, 2000) (Molnar, Sonkoly, and Trinh, 2009). Thus the need of proper tuning, an appropriate congestion control algorithm and the low-loss paths are necessary. Although the link paths we will use for our implementation are consists of 1Gbps we will try to use the feature with and without the Linux zero-copy system calls. Expectations are to reduce the energy consumption of our Big Data System by saving energy from the processor usage.

How to transfer large amount of data via network: Harry Mangalam - This specific reference talks and explains the difficulties which are currently opposed of transferring large amount of data through the network. By the meaning of large amount of data we mean TBs which using simple or every day tools would take as big portion of time. Different tools and protocols are proposed for this reference where the author explains the pros and cons of each of the tools. Also a brief description is given about the usage and at the end we can also read comparison for the tools and difficulties that the author encountered during the testing. Details are given about the size of the files which are wished to be transferred through the network as the required time (real, user, and system time) for the whole procedure for each of the tools that the author is using. Although we have to take into account that most of these tools are Linux and Mac OSX compatible. Tools which are mentioned are bbcp, bbftp, Fast Data Transfer (fdt), Globus Online, netcat, Aspera ascp.

A TCP Socket Buffer Auto-tuning Daemon: Shao Tao, School of Computing National University of Singapore, Lillykutty Jacob, School of Computing National University of Singapore, A.L. Ananda, School of Computing National University of Singapore. In the context of LFN (Long Fat Networks) the paper explains the necessity to have different buffer size instead of default as is known to be used for the TCP implementation. Well-tuned networks can perform up to 10 times faster in some cases as said by (Rapier, Stevens, Bennett, and Tasota, 2012). The implementation of a TCP socket buffer auto-tuning daemon (Steinberg, and Pants, 2009) which is a computing program(software) running as a background process seems to be the most convenient and appropriate solution to tackle this issue. The possibility of the tuning daemon to run on different end devices and communicate between each other in purpose to send and receive tuning information makes the communications easy. Tuning Daemon consists of two parts which are the Auto-tuning and Communicator. By receiving the ICMP Echo Request packets it can calculate, measure the bandwidth and the delay which depends on the response of the tuning exchange messages and calculate the Bandwidth Delay Product (BDP) to adjust the probing intervals and the optimal socket buffer size for each of the TCP connections (connections are located at /proc/net/tcp). The results clearly shows the increased throughput which is achieved through the daemon usage.

Transport Protocols for Large Bandwidth-Delay Product networks: Rui Policarpo Duarte, Instituto Superior Tecnico (IST) explains first of all the reason why the TCP protocol in general is the main responsible mechanism for under-usage of the available bandwidth for Large Bandwidth-Delay Product networks. Important to notice is while the BDP (Bandwidth-Delay Product) continues to grow TCP performance became bottleneck itself as mention by

(Duarte, 2008). A brief introduction and description is given for the congestions control algorithms which are available or under development and an explanation about their pros and cons for each situation. Thus for the BDP limitation for the networks different protocols are given and compared for different scenarios and conditions. Five categories are classified: 1) loss based, 2) delay based, 3) loss & delay based, 4) explicit congestion notification and 5) split connections.

Self-tuning Price-based Congestion Control Supporting TCP Networks: Hao Wang, Zuohua Tian, Department of Automation Shanghai Jiao Tong University of Shanghai, Qinlong Zhang, Department of Economics University of Florence Italy. The publication focus a method called AQM which main responsibility to improve the performance of end-to-end congestion control on routers. SPC which is our main focus is the proposed self-tuning congestion control scheme. An enhanced price with proportional-integral-derivative control property is introduced into SPC to improve its capability of detecting and controlling network congestion.

Power and Performance Management in Cloud Computing Systems: Marco Guazzone, Prof Cosimo Anglano ,Armedeo Avogadro University of Eastern Piedmont. Nowadays the higher ground in the IT industry is taken for the Service Oriented Architecture and to thread "Everything as a Service". As it's known the aim of the cloud is to maximize the profit by minimizing the amount of violations of the Quality-of-Service (QoS) levels agreed with service providers and at the same time to lower the infrastructure cost. Among these costs one of the most crucial is the energy consumption, and by running the on the cloud it plays a primal role. In case to minimize the QoS violations and at the same time to reduce the energy consumption is a conflict and challenging issue. In this thesis is proposed a framework to automatically manage computing resources of cloud infrastructures. Through simulation, it’s shown that significantly reduce QoS violations and energy consumption with respect to traditional static approaches.

Performance Modeling Power Consumption and Carbon Emissions for Server Virtualization of Service Oriented Architectures: Paul Brebner, Liam O'Brien, Jon Gray, NICTA, Canberra Research Laboratory & Computer Science Laboratory, RSISE, ANU Canberra, Australia. In this publications the authors main aim to provide information and metrics how SOA (Linthicum, 2015) and Server Virtualization is more efficient and decreases the greenhouse gas emissions. On the other hand using SOA combined with Server Virtualization may significantly increase the risks such as saturation and Service Level Agreement (Rouse, 2014) violations. A conclusion with an overview about potential problems and benefits of SOA virtualization like reduction of throughput around 15%, 20-60% increase of response time, also it lowers the total cost of deployment, enable maintenance of multiple versions of services, and provides also emissions and power saving.

OpenStack: is a free and open-source cloud computing software platform that's also a reason why is more suitable to be used for our test purpose. Users are developing their OpenStack infrastructure as a service known as IaaS solution. OpenStack cloud software composed of a series of interrelated projects that control pools of processing, storages, and networking resources through a data center which users manage through a web-based "dashboard", command line tools or a RESTful API. An OpenStack user can launch simultaneously numerous different VMs (Virtual Machine) Instances with different OS (Operating Systems). OS can be uploaded as images or installed on an Instance as .ISO, vmdk, qcow2, raw, VDI, VHD and etc. Apart from uploading images users can also add a link from where OpenStack Instance will receive the image to initiate an Instance. For each Instance users can select the appropriate resource like number of VCPU main memory and HDD. Network can also be created and it's important to define that the network runs NAT to provide more security from the incoming traffic of OpenStack Network. Also security groups can be assigning in case to allow specific incoming or out coming traffic. Snapshot is a feature which you can create exactly the same needed image in case to launch a same instance more than one time.

OpenFlow: is a feature which enables to remote controllers to set from which part of the network packets will move through network of switches. The developers are recommending at least two main controllers - a primary, and a secondary which will act as a backup. Separating the control from the forwarding allows more sophisticated traffic management by using the ACLs and routing protocols. Switches from different suppliers can be managed remotely using a single open protocol by using the OpenFlow. (Gibilisco, 2012) Its inventors consider OpenFlow an enabler of SDN. Network Administratiors can remotely access different switches (in their network) in case to access their packet forwarding tables, for adding, modifying and removing packet matching rules and actions. These rules/actions which are being defined by the Network Administrator can be configured with lifespan and that leaves the forwarding of matched packets to the switch at wire speed for the duration of those rules. Controller's main responsibility is managed the unmatched packets which rules were not defined how to handle them. Ability of the controller is also to decide what's going to happen with those packets, like discard them, create new forwarding flow in the table rules to prevent a structural flow of traffic between switch and controller. Controller has the feature also to decide to forward the traffic by itself. OpenFlow protocol is found on the top of the TCP and prescribes the TLS.

The Practical Obstacles of Data Transfer: Why researchers still love scp: Hai Ah Nam, Jason Hill, Suzanne Parete-Koon from Oak Ridge National Laboratory focus on transferring large datasets over network by using parallel streams. Researchers often under-utilize the network and resort to painfully-slow single stream transfer methods such as scp to avoid the complexity of using multiple stream tools as GridFTP and bbcp, that is a main reason why these research focus on showing the difference between using multiple stream tools. Through this research it was shown that multiple stream tools can achieve higher results in the context of transfer speed. As a conclusion researchers should not spend countless hours trying to achieve peak performance, rather, they should optimize their productive hours by allowing for data transfer to occur seamlessly. Using the available utilities, data transfer can become much less difficult and partially automated through scripts.

Transfer of large volume data over Internet with parallel data links and SDN: **by**Khoruzhnikov S.E., Grudinin V.A., Sadov O.L., Shevel A.Y, Kairkanov A.B. The transfer of large volume data over computer network is important and unavoidable operation in the past, now and in any feasible future. There are a number of methods/tools to transfer the data over computer global network (Internet). In this paper the transfer of data over Internet is discussed. Several free of charge utilities to transfer the data are analyzed here. The most important architecture features are emphasized and suggested idea to add SDN Openflow protocol technique for fine tuning the data transfer over several parallel data links.

Exploiting Network Parallelism for Improving Data Transfer Performance: by Gunter D, Kettimuthu R., Kissel E., Swany M. Many scientific applications, including bulk data transfer, can achieve significantly higher performance from vir- tually loss-free dedicated resources provisioned on shared links, than from opportunistic network use. Research and Education (R&E) backbones, including the Energy Sciences Network and Internet2, provide general-purpose services to allocate dedi- cated bandwidth. However, in order to fully take advantage of this technology, applications need to move from coarse-grained "reservation" strategies, to more sophisticated control based on software defined networking (SDN) with technologies such as OpenFlow. We propose here, as one practical step in this direction, using multiple paths for the same application transfer session. This can add bandwidth from "best effort" and dedicated networks, and can also facilitate performance with applications using multiple 10G NICs over 100G capable paths.

2.2 References Summary

From all the above research paper concerning the Big Data we can see that there is not only a single field of research or single solution without "trade-offs". Since we are trying to be efficient in mater of energy consumption and on the other hand transfer all dataset on the network as fast as possible it's hard to achieve in the fullest this goal without sacrifice performance or energy consumption. As well said and mentioned in the publication in (Brebner, O'Brien, and Gray, 2009) the Server Virtualization may significantly increase the risk of saturation and Service Level Agreement violations.

From the research which was done by (The Climate Group, 2008) we realize the importance of taking into account the sustainable and eco-design developing. That is a reason why we are taking into account the green parts which makes our development more sustainable. We believe if we transfer our datasets faster system can enter ideal mode to save energy. As explained by (Gunter, D., Kettimuthu, R., Kissel, E., Swany, M., Yi, J., and Zurawski, J., 2012) in they are publications by using data transfer application over multiple paths for the application layer and SDN OpenFlow approach they could achieve improved throughput. For the test approach the research team transferred the dataset over WAN.

Using cloud infrastructure like OpenStack will help to compare different scenarios with different testbed resources. Since OpenStack is running on the server there are is no need for any user to have a physical access to the server but can work remotely from any place feels comfortable.

By using multiple streams we can increase our transfer speed as mentions by Hai Ah Nam, Jason Hill, Suzanne Parete-Koon in the research. Some of the multiple streams utilities are been introduced for that purpose. Important is also to tuned the system parameters to help achieve higher transfer speed, although there are many in the Linux systems.

3. IMPEMENTATION

Implementation is the part of the thesis which is described the research methodology which has been used to approach the scientific problem. A brief introduction and explanation is given how to use the file transfer applications and what are their features. Also description is given about scripts which were implemented and how to use them. The testbed specifications are given in details about which hardware and software was used. Apart from that introduction and usage of the cloud infrastructure "OpenStack" is given.

3.1 File Transfer Utilities

The amount of time to transfer over global computer network (Internet) depends on the real data link bandwidth and volume of the data mentioned by (Khoruzhnikov, Grudinin, Sadov, Shevel, and Kairkanov, 2015). As mentioned from (O'Luanaigh, 2013) in CERN experiments generating one Petabyte of data every second which said they do not keep all of them but only the most interesting. CERN is storing 25PB of data every year. Imagine a case that we would like to transfer all this data from one point to another in case to analyze them or to store them over a network link of 1 Gbit capacity. It will give us about 100MB/sec, hence 25TB/100MB=250000 secs = 69.4 hours = 2.9 days in case we use a link with no traffic and either any event will occur which may slow down our transfer. Thus is the reason why is important to use and test some of the freely available data transfer tools/utilities. In this section we discuss about those software, the way to use them and their features.

3.1.1 Fast Data Transfer (FDT)

FDT is a Java written Application which is used for Efficient Data Transfers. Since is written in Java theoretically it can be executed in any platform and it is easy to be used. It can run as a SCP or a client/server application. FDT is based on an asynchronous, flexible multithreaded system and is using the capabilities of the Java NIO library as mentioned from (FDT Team, 2013). NIO library also known as Non-blocking I/O is a collection of Java programming language APIs that features of intensive I/O operations as said by (Jenkov, 2012).

By using standard I/O APIs you work with byte streams and character streams, but in NIO you work with channels and buffers. Always the data is read from a channel into a buffer, or written from a buffer to a channel. Because of the non-blocking IO while the channel reads data into the buffer, the tread can do something else. Once the data is read into the buffer, the thread can continue processing it. Same goes for writing data to channel. Thus is reason which makes the FDT efficient to be used. Some other advantages of FDT are:

- Transfer data in parallel on multiple TCP streams, when necessary
- Use appropriate-sized buffers for disk I/O and for the network
- Restores the files from buffers asynchronously
- Resumes a file transfer session without loss, when needed
- Uses independent threads to read and write on each physical device
- Streams a dataset continuously, using a managed pool of buffers through one or more TCP sockets

While FDT is transferring a large amount of different datasets it can send and receive at full speed, without the network transfer restarting between files.

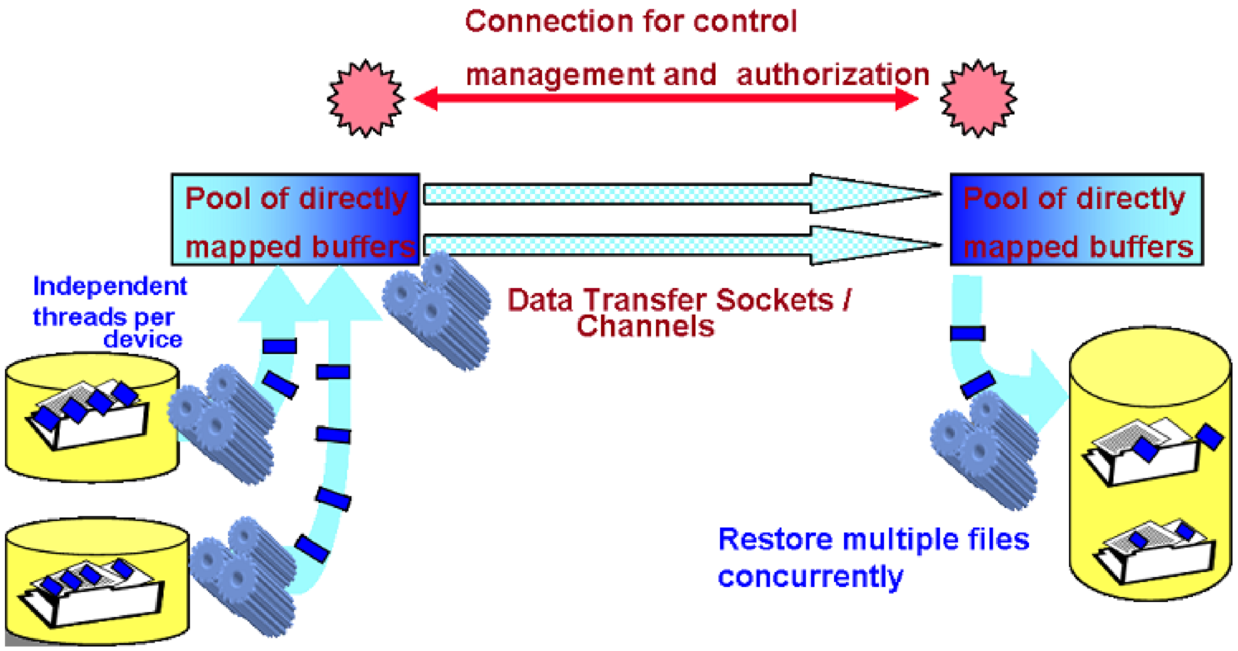

Connection for control
management and authorization
Pool of directly mapped buffers
Pool of directly mapped buffers
Independent threads per device
Data Transfer Sockets / Channels
Restore multiple files concurrently

Figure 3.1.1: FDT Architecture (FDT Team, 2013)

We can see that there is a connection control channel with is responsible for management and authorization. Since we are going to run a lot of test we would not like to authorize a user each time he is trying to transfer thus is the reason we establish Secure Shell (ssh) for known host without the need of password (it will be explain on the following Sub Chapter 3.2). As we see from figure 3.1 FDT keeps a pool with buffers which is going to assign to specific executions.

FDT can be used in one of these following models.

Server: java -jar fdt.jar [OPTIONS]

Client: java -jar fdt.jar [OPTIONS] ***-c*** *<host> [file1 ...]|[**-fl** <fileList>]* ***-d*** *<destinationDirectory>*

SCP: java -jar fdt.jar [OPTIONS] [[[user@][host1:]]file1 [[[user@][host2:]]file2

At first to start the transfer from a sender to receiver, receiver will act as a Server node which is going to receive all the datasets from the client (the sender node). Clients cannot start transferring data if the server is not running which is not the same for SCP. SCP can start by itself the Server and transfer the datasets. For the client part you have to declare the -c parameters which defines the host, the -d which defines the destination of the transmitted datasets, and also you can add the -fl (file list) and add a list of different files or a single file.

3.1.2 BBCP

BBCP is a multi-streaming utility an alternative to Gridftp when transferring large amount of datasets, its capable of breaking up transfers into multiple simultaneous transferring streams, thereby transferring data much faster than any single streaming utilities such as SCP or SFTP. This copy dataset application was written by Andy Hanushevsky at SLAC as a tool for BaBar collaboration. The source code is in C programming language and it's capable of transferring files at approaching line speeds in the WAN. BBCP versions are available for Linux and Solaris platforms. Since BBCP is a peer-to-peer application, no server process is necessary and transfers can be done with an easy just sending the data to the target machine. Tuned up parameters for BBCP are mentioned and explained in Chapter 3.2. It is assumed that bbcp is running on both sides in case to transfer data. Some of BBCP many features which can include the settings are:

- TCP Window size
- Multi-stream transfer
- I/O buffer size
- resuming failed data transfer
- ssh authentication

Way to run bbcp is:

$ bbcp [-options] filename user_name@target_machine_name:filename

There are also different number of options like *-a* to continue the previous failed transfer, *-P [seconds number]* to print every specific defined (by user) second the progress report, *-r [directory name]* to recursive copy a directory which useful for our case, *-w [size]* the TCP window size (default one is 64K) and the *-s [number of streams]* the number of parallel network streams(default 4). Important to note, BBCP is very slow at copying deep directory trees of small files. In case we would like to copy such trees, we should first tar up the trees and then use BBCP to copy the tarball. Use it this method it will increase the data transfer speed. Also the most recent version of BBCP uses option -N named pipelines to use external programs or pipes to feed the network stream. By this way it allows to specify an external program such as tar to provide the data stream for BBCP.

3.1.3 BBFTP

BBFTP is file transfer software which has dynamically adjustable window sizes and can also transmit simultaneously multiple numbers of streams of data. This file transfer software was developed by Gilles Farrache at IN2P3 Computer Center in Lyon, it was written in C programming language, and is open-source software. It is also compatible with most of the Linux distribution systems, but not with Windows OS. Since BBFTP is implements its own transfer protocol, which as mentioned by the developers (IN2P3 group, 2013) it is optimized for large file (larger than 2GB) and also it does encrypt the username and password information but on the other hand it does not encrypt the data being transferred. BBFTP is a Server/Client application which means the receiver node must act as a Server to receive the data from the Client node. It also implements an automatic retry in case of failure. Two methods of connection to the remote host are allowed and implemented in BBFTP. First method, is by starting a BBFTPD daemon on the remote host side and all the user information like username, password are encrypted and transmitted safety to the daemon. Second method, BBFTPD binary has to be accessible from somewhere on the remote host and all control data are transmitted by using SSH tunnel. Some of the BBFTP main features as mentioned by (IN2P3 group, 2013) are list below:

- Encrypted username and password at connection
- SSH and Certificate authentication modules
- Multi-stream transfer
- Big window as defined in RFC1323
- On-the-fly data compression
- Automatic retry
- Customizable time-outs
- Transfer simulation
- AFS authentication integration
- RFIO interface

Ways to use BBFTP is illustrated below:

*$ bbftp [Options] **-i** ControlFile [**-u** RemoteUsername] [**RemoteHost**]*
*$ bbftp [Options] **-e** ControlCommands [**-u** RemoteUsername] [**RemoteHost**]*

For the first command user can add a Control file which will include a number of commands to be executed by BBFTP and its defined by using the argument ***-i***. By adding the argument ***-e*** they user defines the need of a single Control Command which can be a **cd | get | lcd | mget | mkdir | mput | put | dir | rm | stat | df** which are more or less basic Linux system commands but it can be done from a remote host by using BBFTP utility. For our research purpose arguments which are important to be known and used is only **put**. Also **-p** is defined as the number of TCP Parallel Streams and argument **-m** which is the number of buffer size per stream.

3.1.4 Globus Toolkit - GridFTP

Globus Toolkit is open source software developed and provided by the Globus Alliance, which is an enabling technology for the "Grid", allowing to user to share their computing resources, provides securely online across corporations, institutions, and limits the geographical boundaries without reducing the autonomy of the Grid. This toolkit was developed both on Java and C programming languages. Globus Toolkit has a variety of different libraries; some are used for resource monitoring, discovery, resource managements, data management, communication, fault detection, security, file management, and etc. Taking into account that different organization has their own mode of operation it could be impossible to collaborate between multiple organizations due the fact of incompatibility of resources such as data archives, computers and networks. Thus Globus Toolkit was conceived to remove obstacles that prevent seamless collaboration as mentioned in (Grid Alliance, 2014).

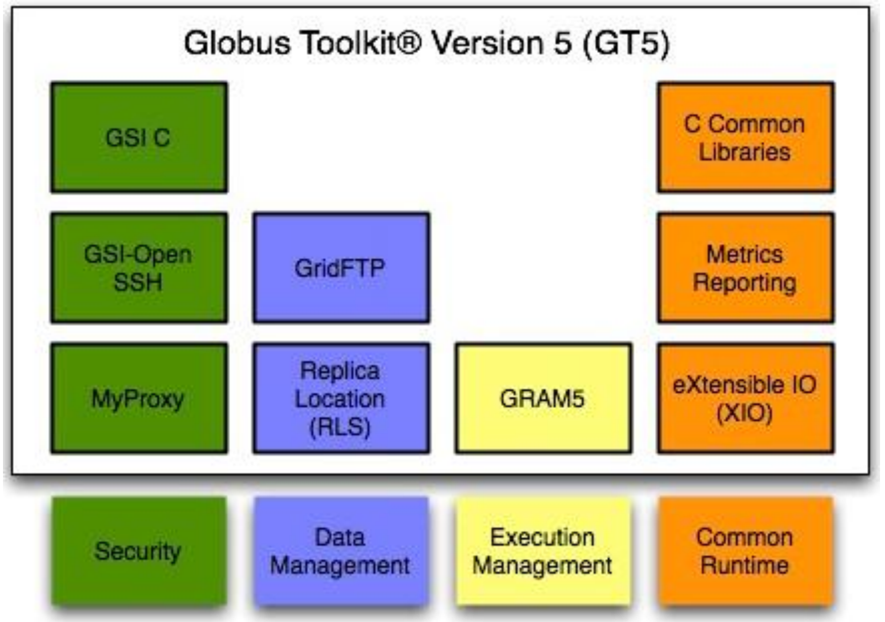


Figure 3.1.2: Globus Toolkit Version 5 libraries (Grid Alliance, 2014)

As we can see from Figure 3.1.2 Globus Toolkit provides a variety of different programs. For example the GRAM5 is the Globus Resource Allocation Manager which converts a request for resource into commands that local computes can understand. GSI which stands for Globus Security Information is responsible to authenticate users and determine their privileges. GridFTP which is the most important for our consideration and the purpose of our research stands for Grid File Transfer Protocol and offers high performance, secure, and robust data transfer across the global network of computers. In fact Globus consist of many other programs as we can see from the figure 3.1.2 but for our case we will focus only on GridFTP. Features of Globus Toolkit are mentioned below:

- Two security flavors: Globus GSI and SSH
- Multi-stream transfer
- Ability to tune I/O buffer size
- retry failed data transfer
- Certificates are needed to authorize users

In case to transfer file to a GridFTP server SSH authentication need to configured, also client (globus-url-copy) has to be configured to support authentication. Commands for transferring file over a Grid using Globus Toolkit:

$ Globus-url-copy [options] ***sourceURL destinationURL***

Different number of options is available for globus-url-copy but for our case we are going to mention only the important and those we used in our research. Starting from **-r** [directory name] in case we would like to transfer an amount of different files, **-f** [file name] in case we would like to transfer a single file, **-p** [number of parallel streams] define the number of needed TCP parallel streams, and the **-bs** [buffer size] to specify the size of the buffer to be used by underlying transfer methods. Since for Globus Toolkit certificate is needed to make file transfers across the global network, we created and used our own simple certificates for the research purpose.

3.1.5 File Transfer Service (FTS3)

FTS3 has been developed by CERN IT-SDC group to address the data transfer needs across WLCG infrastructure. FTS3 is considered to be one of the new and advanced utility for transferring data over the global network. It's open source software which is responsible to transfer data reliably, in a robust way and at large scale between storage systems using different protocols as HTTPS, WebDav, GridFTP and SRM. Data management is also a feature of FTS3 since deleting files efficiently; it will optimize the resources usage by enabling adaptive concurrency, reuse of connections and a retry mechanism in occasion of failure. Apart from these feature web portal is a web monitoring system of FTS3 which provides a visualized manner and easy way to use web interface for transfer management and monitoring. A FTS3 client can use this utility by 3 different ways, Command Line Interface, Python Bindings, and REST. Each way has different installation and configuration steps that are necessary to be done.

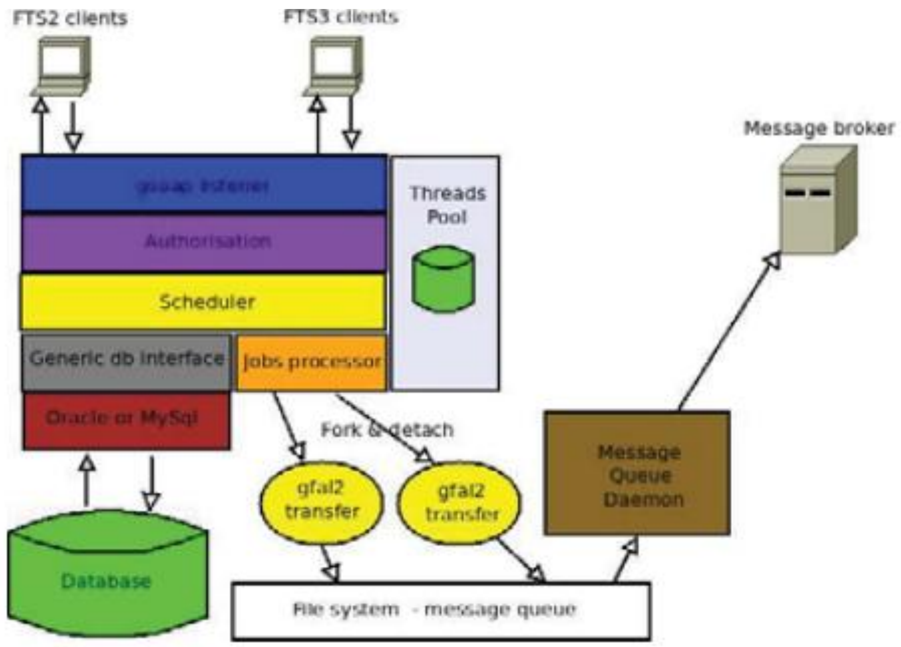


Figure 3.1.3: FTS3 service architecture from (Ayllon, Salichos, Simon, and Keeble, 2014)

The FTS3 interface in general give the opportunity to the user to efficiently schedule data transfers, gaining the maximum of the network bandwidth, availability and storage resources policy limitations are ensured and respected. That feature of FTS3 offers a huge advantage in concept of abstraction and automation.

About the FTS3 architecture as mentioned in (Ayllon, Salichos, Simon, and Keeble, 2014) the components (shown in Figure 3.1.3) are : CLI clients (Command Line Interface), a daemon process for transfer submission, other processes as status retrieval and general VO (Virtual Organization) and service configuration exist, another daemon which exist for bulk stage-in of file from archive using the SRM (Scalable Reliable Multicast)protocol, and all of the data and log are store in an Oracle or MySQL database (defined by user during installation). Some of FTS3 main features are listed below:

- Transfer auto-tuning/ adaptive optimization
- Endpoint-centric VO configuration
- Transfer multi-hop
- REST-style interface for transfer submission and status retrieval
- retry failed transfer mechanism
- staging file for archive
- Support for Oracle and MySQL database
- TCP Multi-stream transfer

To transfer datasets from a local to a remote host it's done by this way:

3.2 Testbed Implementation

At the beginning of the whole research project "Transfer of large volume of data over Internet with parallel data links and SDN" a testbed was implemented in case to run all the scripts that ITMO's BigData team prepared. The importance of the testbed is the large amount of resources that offers to its users to run many test and different scenarios, in case to execute such amount of test by a simple server it would be hard during the lack of resources and time consuming. The need for testbed is to measure results which performed by authors in (Ah Nam, Hill, and Parete-Koon, 2013), where data transfer was performed with Data Transfer Nodes (DTN). The utilities which were used are rsync, scp, bbcp, GridFTP and concrete files have been used (11KB, 3.5MB, 158MB, 2.8GB, and 32GB) in each case. During the experiment it was shown that after the number of 8 streams there were not any changes in the transfer speed. IMTO's University BigData team aims is the also get information about the Linux kernel parameters just to have a more clear image in which data speeds and how each system parameters affect's the data transfer rate. Basically the testbed implementation will provide more precise and clear answers. The idea of the testbed is to compare a number of utilities using SDN Openflow mechanism.

3.2.1 Testbed Characteristics

The current testbed hardware characteristics are illustrated on table 3.2.1 and it consists of two end nodes. The distances between the two end nodes is around 40 Km and the network connection is a public Internet link of 1 Gbit maximum bandwidth.. On this testbed cloud infrastructure OpenStack (version Icehouse) was deployed to manage the VMs. Perfsonar has also been deployed on both sides. The implementation idea is shown at figure 3.2.1

Hardware Type	CPU	Main Memory (RAM)	Hard Disk Drive	Operating System
Server 1 (SV)	2 x Intel 6 Core Xeon E5-2640v2 @2.5GHz	64 GB DDR3		Scientific Linux CE 6.5
Server 2 (SM)	2 x Intel 4 Core Xeon E5-2609v2 @2.4GHz	32 GB DDR3	4 x SATA 500GB raid 5 1000 GB	Scientific Linux CE 6.5
Storage Area Network			HP 16 HDD SAS – 450GB	

Table 3.2.1 : Testbed hardware characteristics

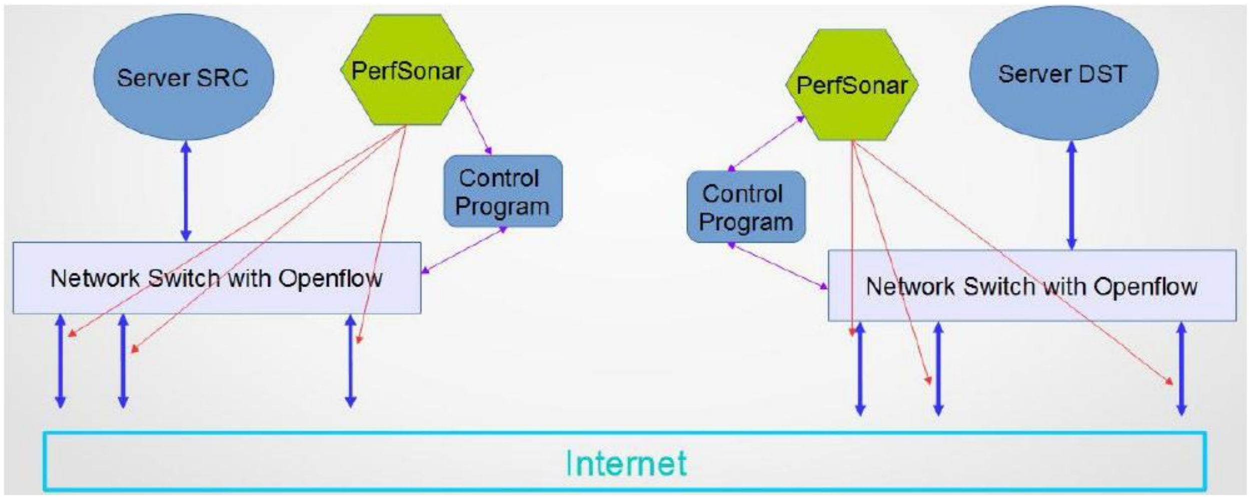


Figure 3.2.1: Testbed Architecture

The reason that have implemented a control program which is running perfsonar is to observe the data link which may experience interruption, control program using perfsonar will collect data and store them in a database. Network switches are running Openflow open-source software which implements the SDN part of the network. On the top of the testbed cloud infrastructure OpenStack is running. Accounts have been created for BigData teams which providers amount of resources from the server itself. More about OpenStack cloud infrastructure on chapter 3.5. For the current research goal we have as aim to transfer Big Data as faster as possible using the “Virtual Channel”. By the term “Virtual Channel” we mean the network which consists between the VMs we deployed using the OpenStack dashboard.

3.3 Scripts Implementation

In case to launch different number of tests to transfer data with different utilities it will be very slow and too much time consuming by typing one-by-one all the commands changing the arguments, and then keep track of the system parameters. Thus the reason occurs to create a number of different scripts for each utility which will launch a data transfer procedure, create log files for the system variable, information, the network link information, and at the extract the results in a view of graphs. Also script was implemented which will create the data which will be transferred through the network. The scripts were written using BASH (Bourne Again Shell). All scripts implementation can be found in https://github.com/itmo-infocom/BigData, as a research team we found github a handy tool to transfer and synchronize the team's work.

3.3.1 Test-data Script

It's responsible to generate directory with files of random sizes with defined size and standard deviation. The reason is because we would like to have a comparative results with the utilities, and by using these data utilities which have compression algorithm implantation it will not allow them to compress their data and send it through the link, but send them as it is. A user can add a list of different parameters to execute the "create-test-directory.sh", list of parameters can be viewed below:

```
"-h"|"--help"         show this text
"-q"|"--quite"        by default false
"-l"|"--logfile"      by default $0\.log
"-n"|"--dirname"      by default ./test_directory
"-z"|"--dirsize"      by default 1024 KiB
"-f"|"--filesize"     by default 102 KiB
"-d"|"--dispersion"   by default 10 KiB
"-s"|"--sample"       by default /dev/urandom
"-b"|"--block-size"   by default 1K
```

The files name which will created is an linear arithmetic count starting from 1,2,3, etc. Also it uses normal distribution (gauss). This script before creating the data it will check the HDD (Hard Disk Drive) free space before creating the random length data (if the user will run out of HDD space avahi daemon cannot start and GUI will not show to the user after next login).

3.3.2 CopyData [utilityName]

This is our main script implementation which will launch a transfer. The CopyData script is written in BASH and each CopyData script exist for each utility. There are no main differences between these five scripts apart from some syntax differences to run each utility with a specific way. The aim of these scripts is not only to launch a dataset transfer but also to take measurements, create log files, trace route files, in case for the user to see what really happened in the network during the dataset transfer. In case to launch the scripts a number of arguments are necessary to be added in the command line of the terminal window as shown below in table 3.3.1.

After the execution of this script a directory will created with the following name "CopyData(UtilityName).LocalHost.RemoteHost.Date.Y.M.D_H:M:S_configFileParameters". The Utility name it can be fdt | bbcp | bbftp | gridftp | fts3, the symbols after the Date will symbolize Y the year, M the month, D the current date number, H the hours, M the minutes and S the current seconds when the directory will be created. Inside the CopyData directory the script will create the following files: Ping, Traceroute, Comments, Abstract, Log files and Sosreport.

Args	Description of each argument
$0	Name of the executable file exp. ./CopyData.(fdt|bbcp|bbftp|gridftp|fts3)
$1	Configuration file, can be empty then default parameters will be added or can be added the parameters "--streams #", "--windowsz #" , "-D[minport:maxport]" (for BBFTP)
$2	Set the full path were the Log directory with all the log files it will created
$3	Set the full path were the test directory is locate (dataset which will be transferred)
$4	Set the username exp. root
$5	Set the remote host combined with the username should look like exp. root@192.168.1.50
$6	Set the destination of the dataset , default value is set as /dev/null to save space
$7	Set user comments, optional parameter

Table 3.3.1: CopyData arguments table

Information	Value Type
Start Time	<date time>
Command Line	<utility command line parameters>
Total dataset size to transfer	<value in KiBs>
Number of files	<value>
Source directory with file	<source directory path>
Average file size	<value in KiBs>
File size dispersion	<value>
Local host name	<value of localhost>
Remote host name	<value of remotehost>
Data size transferred	<value in KiBs>
End time	<date time>
Completion	<YES/ABNORMAL>
Average transfer speed	<value in KiB/sec>

Table 3.3.2 : Abstract file Content

The Abstract file has the most essential information about the dataset transfer, they information is shown in the table 3.3.2. From the abstract file essential information will be extracted in case to plot graphs with the results we got from each transfer. The calculation of Average Transfer Speed was done by a simple equation which was capturing the time before executing the utility to transfer the dataset and after that the utility finishing time was captured which was subtracted by the start time and then the whole dataset size was divided by the elapse time of the transfer. The equation can be viewed below. Inside parameter STARTIME will be stored the current time converted into string so it will be divided by ENDTIME. Argument ${DirSize} has the size of the dataset in Bytes.

```
STARTTIME=$(date +"%s")
Starting BBCP
ENDTIME=$(date +"%s")
ELAPSEDTIME=$((${ENDTIME}-${STARTTIME}))
TransferRate=`expr ${DirSize} / ${ELAPSEDTIME}`
```

The Comments file is created based on the command line arguments the user is going to add for the execution of the utility (number of streams, window size, etc) of CopyData script and the comments which will be added as a last argument for CopyData. From comments file we can see which parameters produced the current dataset transfer results.

The Log file is filled in with all the output of the utility which was executed.

The Ping file includes the output of the command 'ping -c 10 RemoteHost' to check if the remote host which will receive our dataset is reachable from the local host.

The traceroute it has the output of the path which the packets are following to reach the end node. The Top file is filled with the 'TopRun.sh' output.

The sosreport is a tar file which includes all the system parameters as directories. Is a copy of the */proc* directory which has all the necessary system information.

The main reason we are creating all this sosreport, log files, abstract, comments, and etc which is quite large around 6-7 MB in total is to keep a history about how we obtained this results, which parameters have been used and under which circumstances.

3.3.3MultipleExecutionsCopyData [utilityName]

The purpose of this script in general is to execute Multiple times the CopyData script using different configuration files for as the second arguments ($1) for the CopyData script. Thus the script will create a number of different configuration files which will include the different number of parallel streams and window sizes. For each utility the configuration files are different in matter of content. After that the configurations files will be used by the MultipleExecutionsCopyData to launch CopyData script and results will be stored in a log directory defined by the user. In table 3.3.3 can be found the number of arguments, which is pretty similar with the CopyData script apart from the second, third, and forth arguments. For the second argument user can define what the test should be. User have 2 choices, can write the letters of "**-pwl**" or "**-pwr**", where the letter **p** stands for parallel streams, **w** stands for window sizes, **l** stands for list, and **r** stands for the range of number which can be given from the user as the third and the forth argument. List arguments should be separated by commas and range arguments should stand from 2 numbers separated by semicolon.

Args	Description of each argument
$0	Name of the executable file exp. ./ MultipleExecutionsCopyData.(fdt |bbcp |bbftp |gridftp |fts3)
$1	Set the full path were the Log directory with all the log files it will created
$2	User can add a parameter like "***-pwl***" to execute the CopyData with different amounts of parallel streams window sizes in form of list (ex. 1, 2, 4, 6, 7) or "***-pwr***" in form of range numbers (ex.1:10).
$3	Could be a list or range values of parallel streams separated by commas for list or the min and max of a range of parallel streams separated by semicolons, depends on the second argument ($2) value if includes **r** or **l**
$4	Could be a list or range values of window sizes separated by commas for list or the min and max of a range of window sizes separated by semicolons, depends on the second argument ($2) value if includes **r** or **l**
$5	Set the full path were the test directory is locate (dataset which will be transferred)
$6	Set the username exp. root
$7	Set the remote host combined with the username should look like exp. root@192.168.1.50
$8	Set the destination of the dataset , default value is set as /dev/null to save space
$9	Set user comments, optional parameter

Table 3.3.3: MultipleExecutionsCopyData arguments

Apart from the execution of multiple scripts with different configuration files this script will create its own directory and also add at the end of each of the executions the configuration form the configuration file so users can identify them easier. Except from launching the transfers also creating data file and from those data files graphs will be plotted. The whole procedure is illustrated below:

i. MultipleExecutionsCopyData.[UtilityName] will be executed with the specified arguments given by the user.

ii. MultipleExecutionsCopyData.[UtilityName] script will call the CopyData.[UtilityName] script.
iii. CopyData.[UtilityName] will produce the Log Directory in the path which the user specified with command line arguments.
iv. MultipleExecutionsCopyData.[UtilityName] script will call the createData.[UtilityName] script.
v. createData.[UtilityName] script will create the file which will be suitable for gnuplot to create graphs.
vi. MultipleExecutionsCopyData.[UtilityName] script will call the plotGraphs.[UtilityName] script.
vii. plotGraphs.[UtilityName] script will plot the graphs from the data file which was created by the createData.[UtilityName].

User can pass as command line arguments to the Java program a number of file in case to prepare a file to compare different utilities with different number of parallel scripts. In general the executions of MultipleExecutionsCopyData should look like:

./MulitpleExecutionsCopyData.[UtilityName] full/path/log/Directory [-p|-w] [-l|-r] [values1,....,valuesN | Min:Max] full/path/of/Dataset [username]@[hostname] /path/destination

Apart from the execution style it was given above user can also execute the MultipbeExecutionsCopyData.[UtilityName] by the way which is below:

./MulitpleExecutionsCopyData.[UtilityName] full/path/log/Directory [-pwl\-pwr] [ParallelStreams: values1,....,valuesN | Min:Max] [WindowSize: values1,....,valuesN | Min:Max] full/path/of/Dataset [username]@[hostname] /path/destination

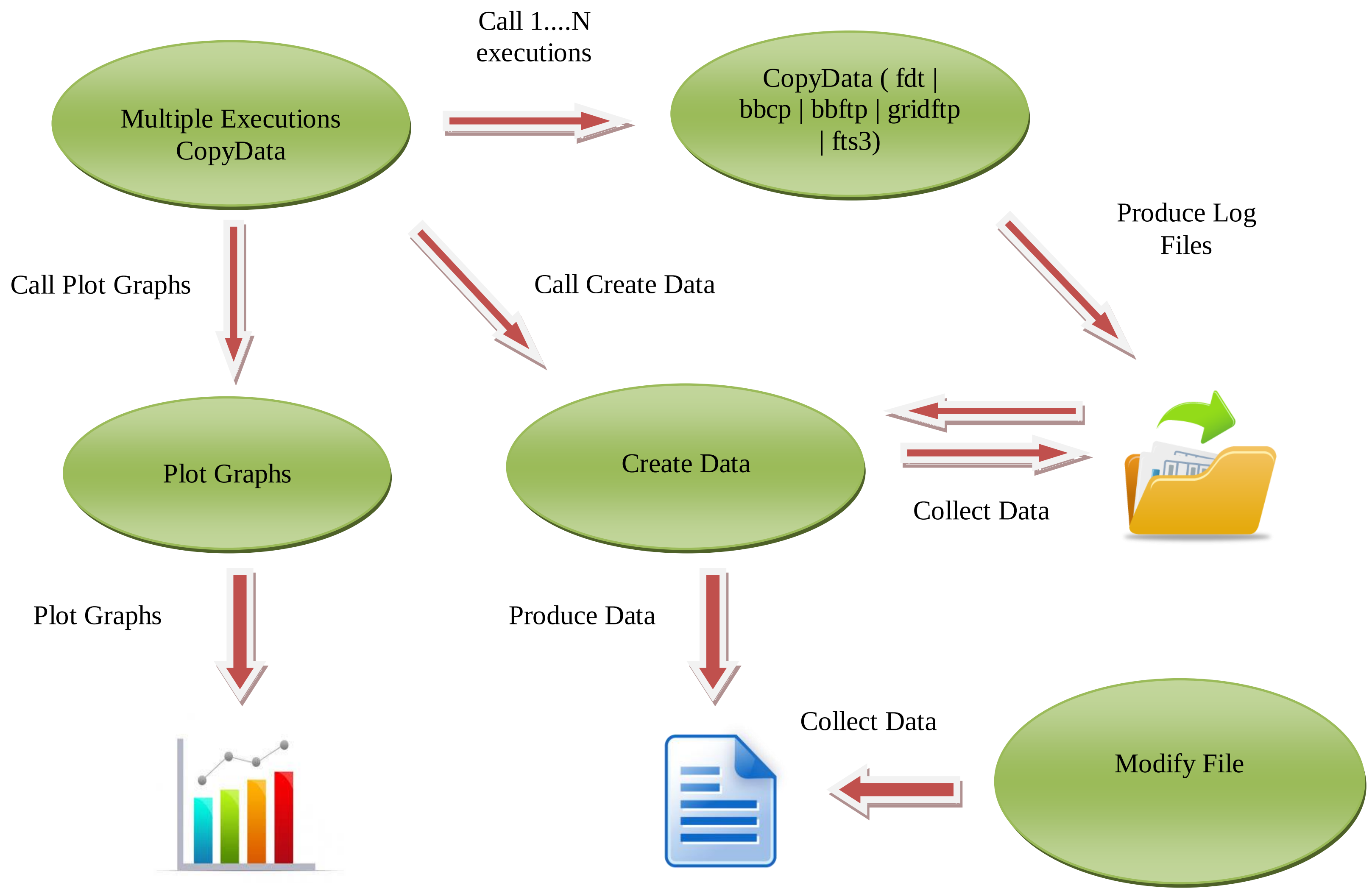


Figure 3.3.1: Function call of Multiple Execution of Copy Data

3.3.4 createData [utilityName]

The script named createData is rather a simple script but useful for extracting data like the Average Transfer Speed in KBs from the abstract file of each log file which was created by the MultipbeExecutionsCopyData script, also grapping the configuration from each configuration file from each (for this situation the parallel streams). In the end a new file name plot_data.txt will be created with two columns. The first column which is that parallel TCP streams is the axis X and the second column is the axis Y. This script will be launch from script MultipbeExecutionsCopyData.[UtilityName] after the end of the multiple executions and command line arguments will be added from the script which could be "streams" for parallel streams and "windowsz" for window size.

3.3.5 modifyFile [utilityName]

Is a script which will capture the createData file (filename:plot_data) and modify it into columns and plotGraphs can used it to plot lines for each column as a different number of Streams and the results(Average Transfer Speed) we have for each one of them. As default use we have used 7 number of streams (1,2,4,8,16,32, and 64), 9 specific window size (131072, 262144, 524288, 1048576, 2097152, 4194304, 8388608, 16777216, and 33554432) The file which will create will have 8 columns the first is for the different window sizes and the other 7 are the results of Average Transfer Speed for each different number of TCP parallel streams we used.

3.3.6 plotGraphs [utilityName]

The script named plotGraphs is running a script using gnuplot, will have as an input the file from the modifyFile. [UtilityName] named comperative_datat.xt. From this file it will get the columns and it will plot the graph and it will give a PostScript (ps) and a Portable Document Format (PDF). The script plotGraphs.[UtilityName] will be executed by the script MultipbeExecutionsCopyData.[UtilityName] after it executed the script createData.[UtilityName] and modifyFile.[UtilityName]. This specific script was created to compare only specific utility arguments like parallel streams based on the transfer speed and the window size. Example of the graph view can be seen in figure 3.3.2. From the X axis are located the window size in Bytes and from axis Y are the results of Average Transfer Speed in Kilo Bytes. On the left corner of the graph it can be found the streams with the representative colors. After on each execution of CopyData.[UtilityName] the title of the graph will change depending on the dataset the user wants to transfer. On the down left corner a timestamp is added to keep track of the days which the graph was plot.

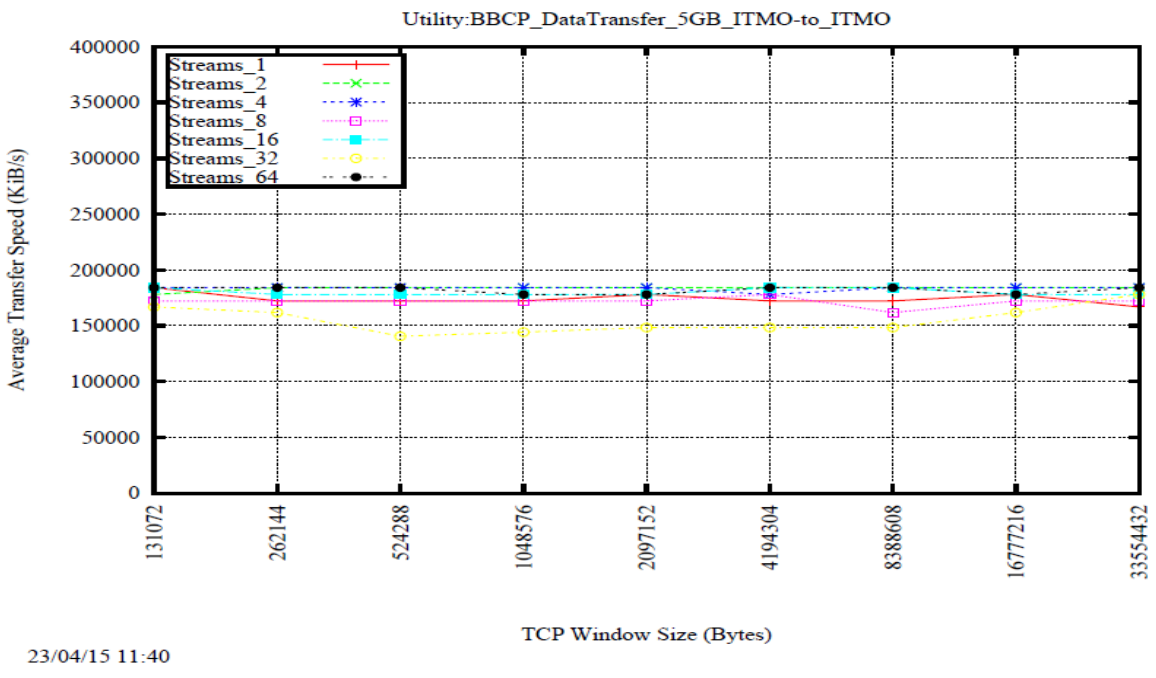


Figure 3.3.2: plotGraphs Example

3.3.7 set-passwordless-ssh

Important to note is that all the scripts implementation are using Secure Shell to connect to the remote host and transfer their data. Since our script is running multiple times to complete the task which was assign to execute for each dataset transfer authentication will be required and thus the reason why is necessary to implement password less ssh connection. At first this script will check if an RSA (Ron Rivest, Adi Shamir and Leonard Adleman cryptosystem) key exist in .ssh directory. If not it will create one and will pass the content of the id_rsa.pub (file which is created after ssh-keygen -t rsa) to the remote host in case to allow the connection from local host without the need of any root access password. The content of the id_rsa.pub will be stored in .ssh directory under the name of authorized_keys file. Then user has to manually enter the instance and change the permission of the authorized_keys ($chmod 700 authorized_keys) and the .ssh ($chmod 600 authorized_keys) directory and the user can start the file transmission scripts without then need to enter manually the password.

3.4 Cloud Infrastructure

As a cloud infrastructure we have used OpenStack. OpenStack is open-source cloud infrastructure software which allows to user to share among them server resources which are going to deploy VMs for their own usage. An organization which is having a server can install OpenStack on the top of the server to allow authorized users to manage a specific portion of resources through the Dashboard. Administrator of the System (OpenStack) can set the limits of resources to each user. Nowadays a lot of data transfers are done between the cloud and more and more companies are joining the cloud era, which is the main reason in ITMO we decided to install cloud infrastructure on the testbed and focus on transferring Big Data over virtual channel. Also since the testbed we host here is a really powerful server even if we used the utilities with their maximum resources we will still have enough left. By using the cloud infrastructure we can divide the testbed resources into many instances to run different scenarios (each instance is different scenario) and then analyze our results.

3.4.1 Virtual Machines

From the figure 3.5.1 we can see the scenario which can be created through the dashboard and which we run our tests. We can see in table 3.4.1 the instances which are located in the virtually in the same LAN which both sender and receiver are quite resourceful in terms of hardware resources. We can see that all the instances are attached on myNetwork (Address Domain 10.0.0.0/24) . Users can hover on each instance and view instance information like instance ID, current status like active, shut down, rebooting, pause, or suspend ; also a possibility is given to open a console and handle the instance through the installed OS GUI and terminated it if it’s necessary. User has also the feature to create snapshots of a VM in case to clone it and have it more than once. The instances specifications can be viewed from table 3.4.1. Important to mention is that OpenStack used NAT (Network Address Translation) protocol which makes the outgoing traffic easy for the network but on the other hand it does not announce to public the sender route address. Security groups can be defined for each instance restricting or permissions can be added on ports and protocols allowance for the instance. At table 3.4.2 we can see the instance specification which was used to run the tests over real network and not virtual environment.

Instance Name	Location	VCPUs	RAM size	HDD size
Sender	ITMO	8	16 GB	160 GB
Receiver	ITMO	8	16 GB	160 GB

Table 3.4.1: Instances Specifications

Instance Name	Location	VCPUs	RAM size	HDD size
Sender	ITMO	8	16 GB	160 GB
Receiver	PNPI	8	16 GB	160 GB

Table 3.4.2: Instance Specifications

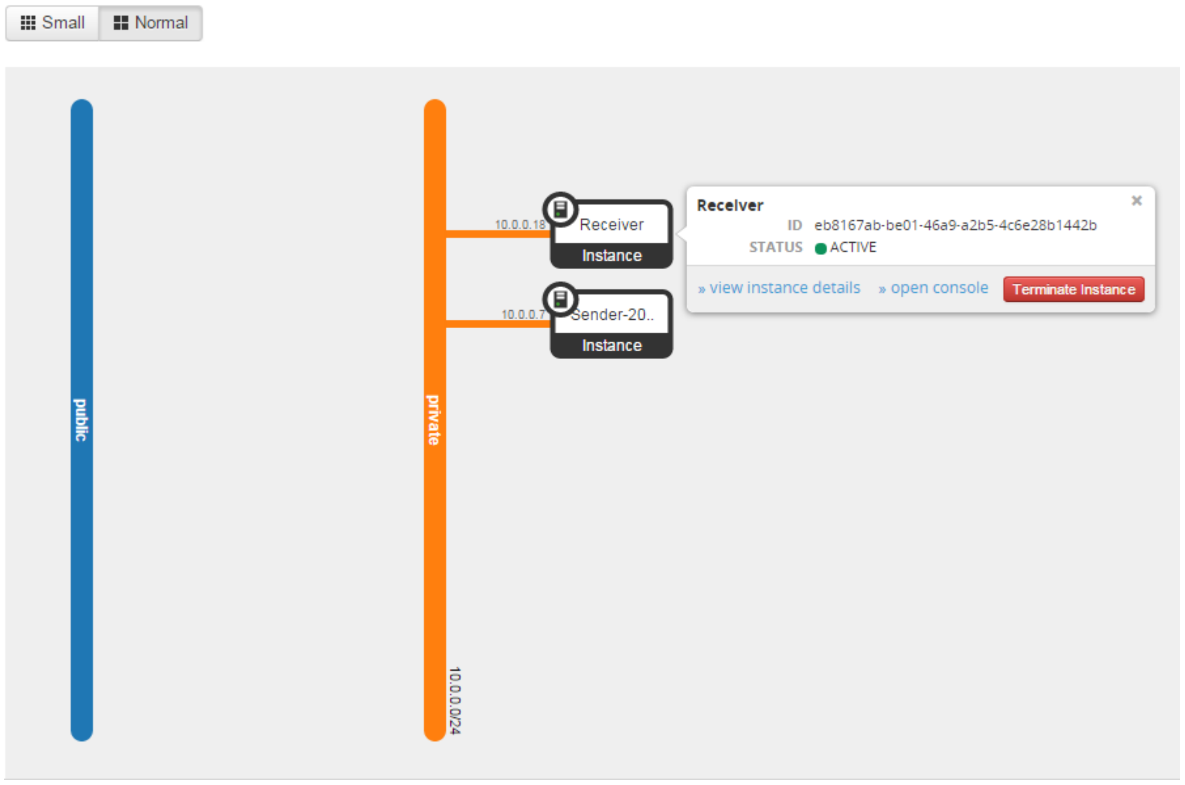


Figure 3.4.1: OpenStack Network

3.4.2 Instances Volumes

After the creation of the instances it was necessary to attach volumes on each of the instances. The reason is to have a large amount of space in case to create the random data which are going to be used for the transfer purposes. As it has been explained on the previous subsection dataset will be 25 Giga Bytes and also after each transfer Log Files will be created where each one of them has size at least 5BM. That is reason why we keep the instance volumes close to 100GB.

First thing we have to do after the creation of volume is to manually attach them to each instance through the dashboard. Second we have to logging and format each volume manually at each instance. All volumes have been formatted in ext4 File System using the command *$mkfs.ext4 /dev/vdb* where /dev/vdb is the volume. The next step is to add a directory which will used of the user as the free space of 200GB. Inside the file /etc/fstat using an editor such as VIM or Nano user can add the location of the mounted device (ex. /dev/vdb) , the path where can be found (ex. /root/free-space-200GB) and File System Format (ex. ext4), forth parameter as “defaults” and the next two as “0” and ”1” as default values. After rebooting changed may take effect.

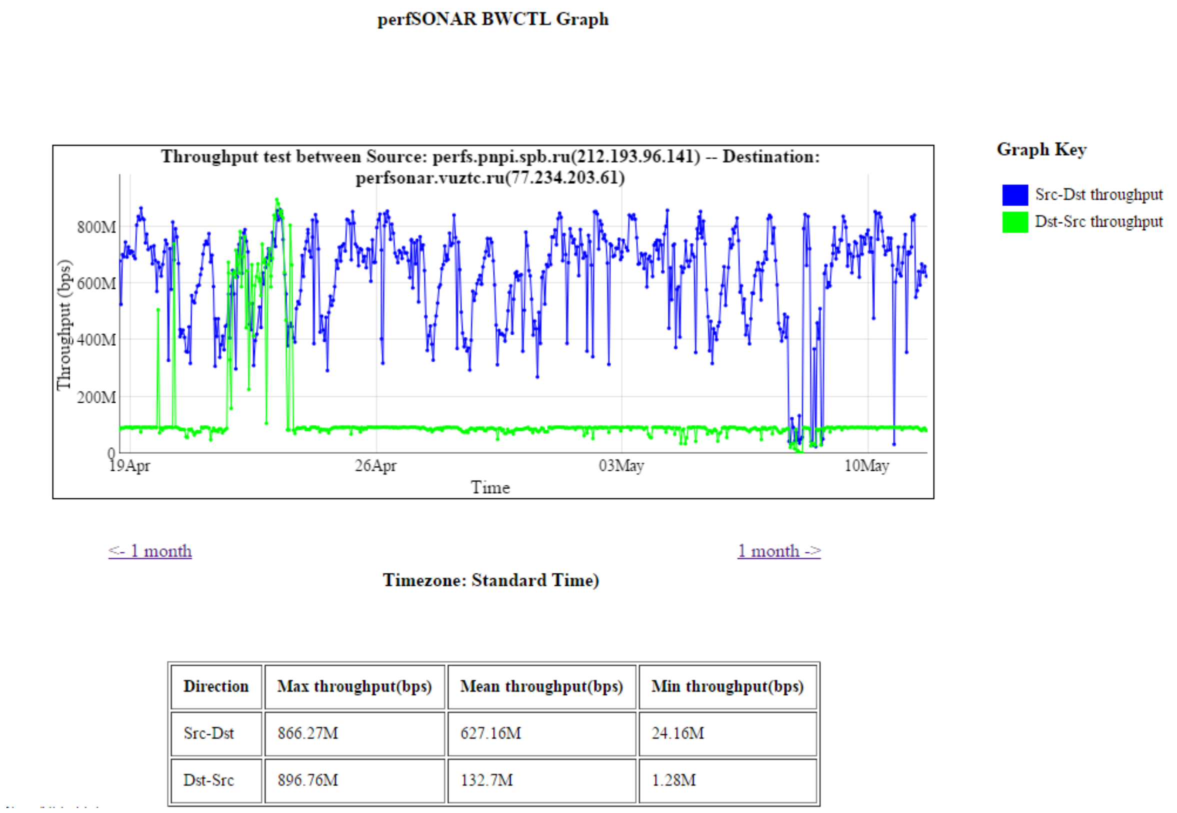
3.5 Scenarios

Some scenarios which was decided to be used with different number of the parameters. In general test could be done by using all the number of different parallel streams but that would consume a huge amount of time that is why decision has been made to use specific ones.

3.5.1 Scenarios Description

A number of different scenarios have been defined for the purpose of this project. The scenarios mostly are defined by the different parameters which are going to be used to transfer datasets. Starting from the number of different streams we are going to use 7 streams. Parallel Streams number 1, 2, 4, 8, 16, 32, and 64. As number of window size we are going to use 131072, 262144, 524288, 1048576, 1048576, 2097152, 4194304, 8388608, 16777216, and 33554432 in Bytes. As dataset to be transferred we created from the create-test-directory script datasets of 25 Giga Bytes.. Apart from that we have 5 different utilities to test all the scenarios. In general all this different combinations will produce the about of 7(Parallel Streams) * 9(Window Sizes) = 63 different scenarios for each of the utilities. All the above scenarios were tested in virtual and real network.

The tests on a real network transfer are taking place from PNPI to ITMO having the instance which is located on PNPI server as a sender node and the instance which is located on ITMO server as a receiver node. The reason we decided to implement our scenario with this way is of the fact that we have higher throughput from PNPI to IMTO rather the opposite. PNPI server is located 40 km away from ITMO server, and through the network are located most of the times around 5 to 7 public routes from where our datasets will pass until reaching the destination. This information (about throughput and trace route) can be found on http://212.193.96.141/serviceTest/index.cgi?eventType=bwctl using the Perfsonar web measurement service which is installed on the servers of IMTO and PNPI and runs as a web service. (See figure 3.5.2) As we can see from the figure 3.5.2 the blue line indicates the available throughput of PNPI, and the green line indicates the available throughput of ITMO. It’s clear to us that in case we would like to achieve the highest transfer rate to compare the utilities we have to use the server which is located at PNPI as the sender node and the server which is located in ITMO as a receiver node.


Throughput test between Source: perfs.pnpi.spb.ru(212.193.96.141) -- Destination: perfsonar.vuztc.ru(77.234.203.61)
Throughput (bps)
800M
600M
400M
200M
0
19Apr
26Apr
03May
10May
Time

perfSONAR BWCTL Graph

Graph Key

🟦 Src-Dst throughput
🟩 Dst-Src throughput

<- 1 month   1 month ->

Timezone: Standard Time)

Direction	Max throughput(bps)	Mean throughput(bps)	Min throughput(bps)
Src-Dst	866.27M	627.16M	24.16M
Dst-Src	896.76M	132.7M	1.28M

Figure 4: PerfSonar throughput Graph

Scenarios that we tests are similar to the scenarios we tested in virtual environment. Parallel Streams number 1, 2, 4, 8, 16, 32, and 64. As number of window size we are going to use 131072, 262144, 524288, 1048576, 1048576, 2097152, 4194304, 8388608, 16777216, and 33554432 in Bytes. As datasets to be transferred we created from the create-test-directory script dataset size of 25Giga Bytes which consists of 244 different files with average size of 100 MegaBytes .

To execute all the above scenarios a large number of time is required. Important to mention is that all the test data we have are located in the hard disk drive. If we consider that we need to execute 630 different tests for each utility and the average time needed to complete a transfer is around 3 to 4 minutes; then the total time is 63*4 = 252 minutes / 60 = 4.2 hours for each utility to finish all the scenarios and that is for a single utility. As we can see to test and transfer Big Data over virtual channel is a time consuming process. In case more testing parameters will be added, like more parallel streams and more window sizes the duration of the execution will increase exponentially.

3.5.2 Alternative Transferring Data ways

After executing all the tests we said in subchapter 3.5.1 and during the results we got, we decided to test the utilities with an alternative way

To transfer data from the memory to the TCP buffer it can be distinguished with two different ways, transferring datasets from HHD or directly from RAM. By executing in BASH the command $ dd if=/full/path/of/dataset of=/dev/null which is a data disk cloning command we are going to receive the transfer speed. During some testes we find out that the speed was in range of 150 – 200 Mbps. If the datasets we wish to transfer is located in the RAM our system will require less time to search for the data since the main memory is faster rather the HHD.

In case to have all the necessary data into the main memory NFS had to be installed and mount the data from a remote host. Mount is a command which allows the user to mount a shared NFS from another machine.

In figure 3.5.3 it is shown that all the sender instances are mounted with NFS from vlan2-net which is located in domain 10.10.20.0/24. In our case all the data which we would like to transmit is located on the memory of a remote host and by mounting the NFS we can transfer data directly from the RAM instead of the HDD. Since the physical memory which is found on the server exited far the amount of data we like to transfer its possible to have them all on the RAM. Instructions about the whole procedure are given by (Timme, 2013).

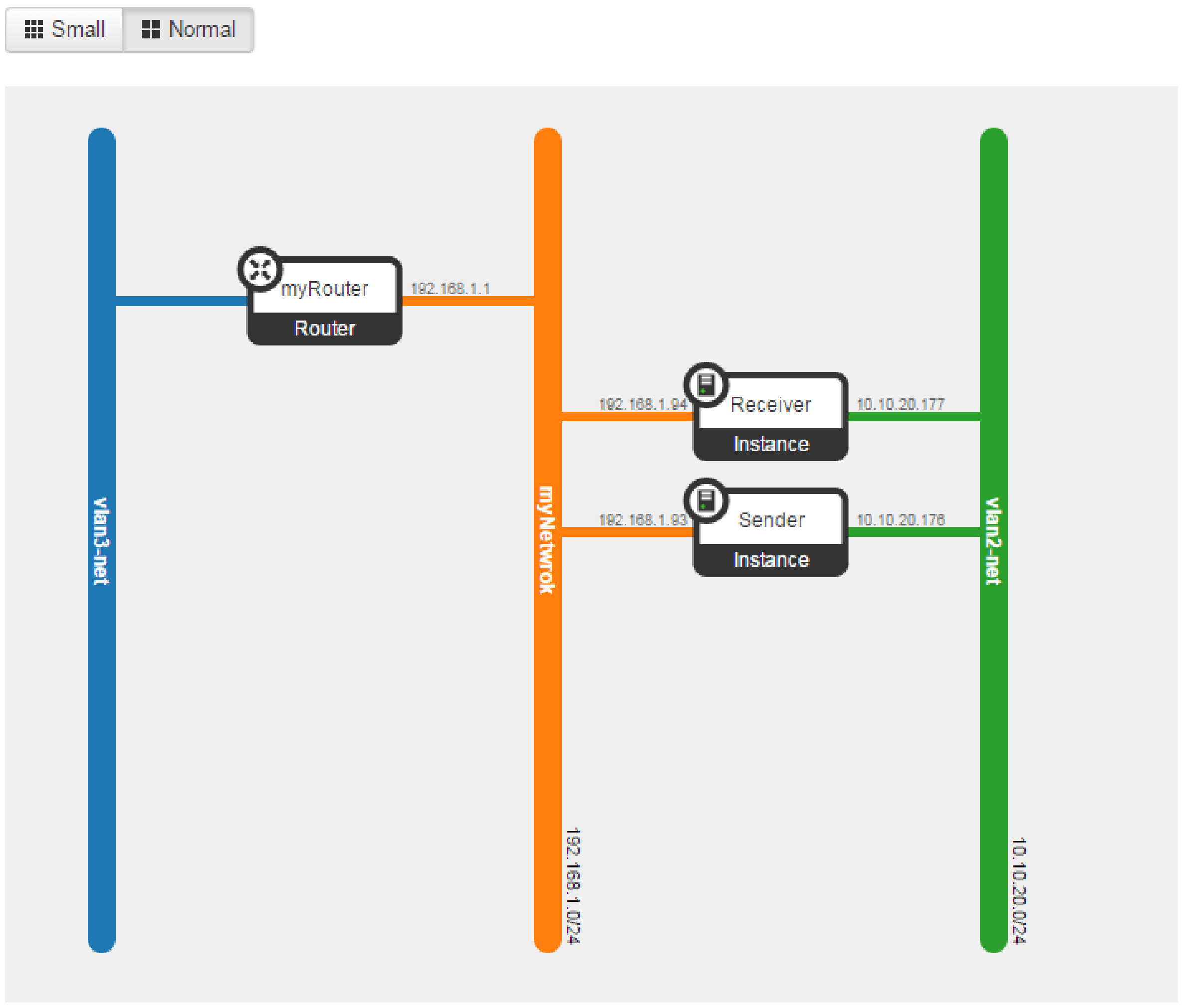


Figure 3.5.3: OpenStack Mount NFS

4. RESULTS

In this section of the thesis results which acquired while executing the scripts, which was mentioned in Chapter 3, are presented and a discussion for the results we have. Results have been acquired while running scripts for BBCP, FDT,BBFTP and for GridFTP. FTS3 was the utility which not tested yet in the context of the Master thesis.

4.1 Explanation about results and Graphs

The graphs sample which plotted can be viewed in figure 4.1.1 with the parameters and description about each one of them. Axis X represent the average transfer speed in Kilo Bytes per second, which was calculated after finishing the dataset transfer, about axis Y there are the number of different TCP window sizes in Bytes, that we used for the tests.

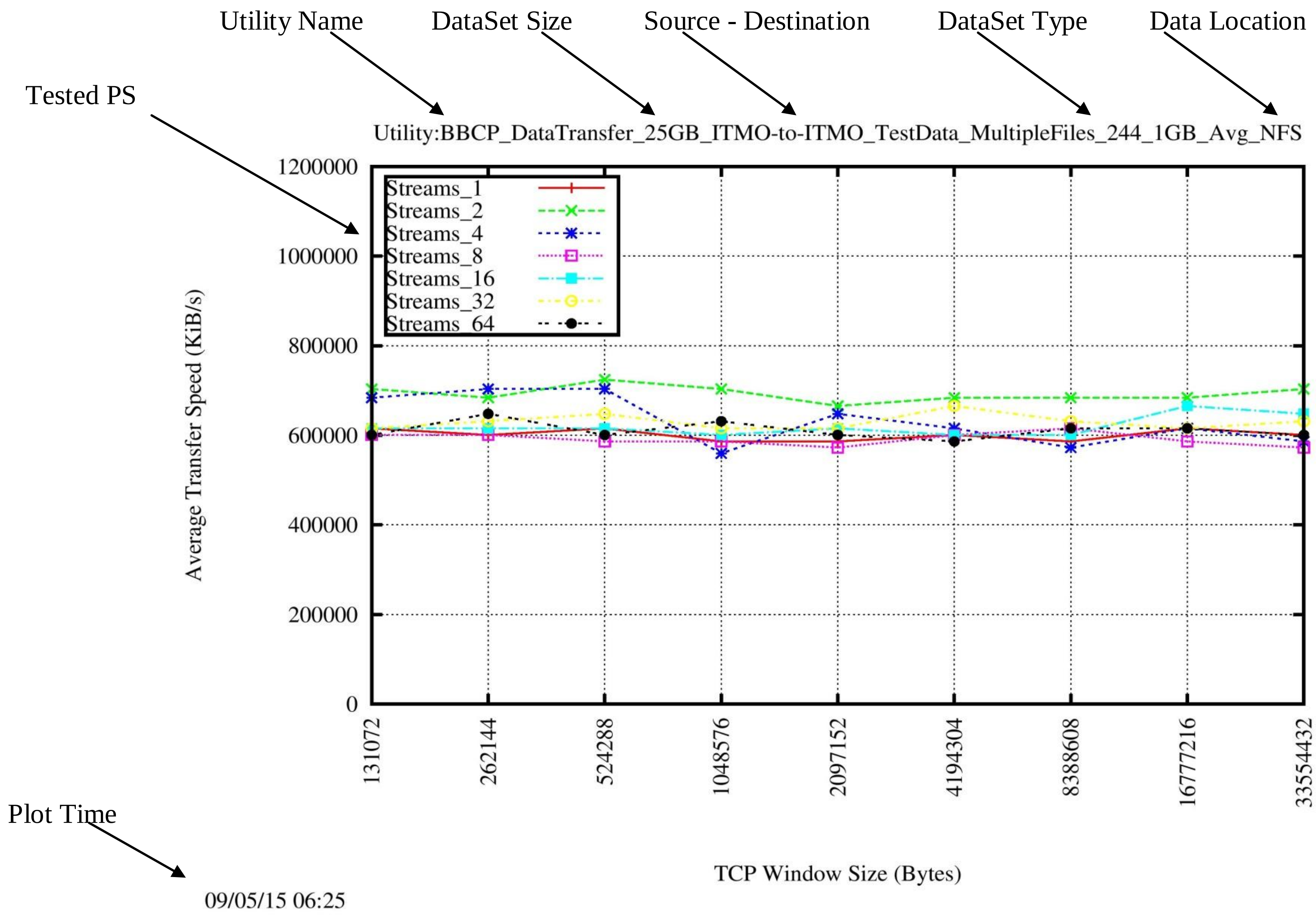


Figure 4.1.1: Graph Parameters

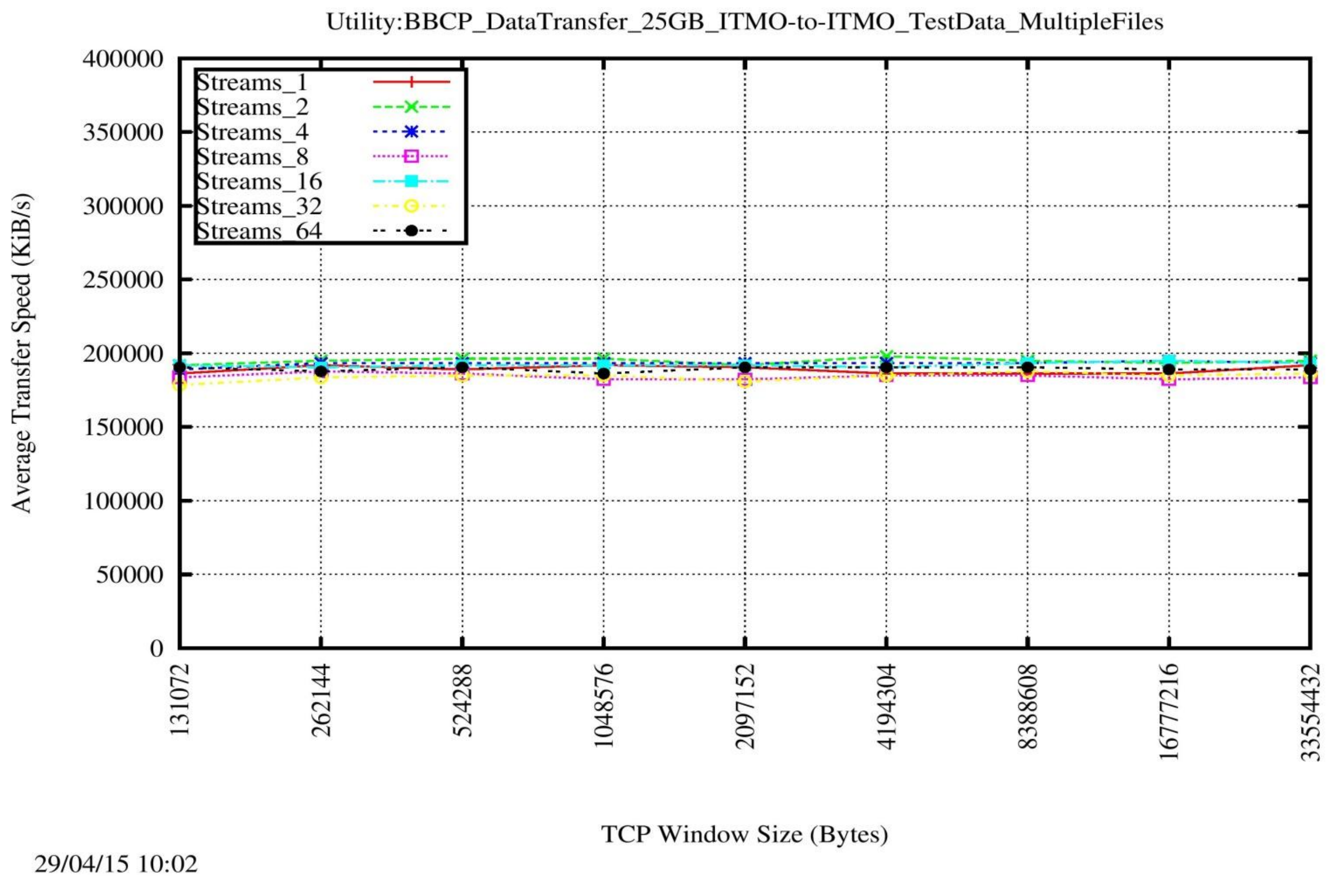


Figure 4.1.2: BBCP ITMO - ITMO from HDD

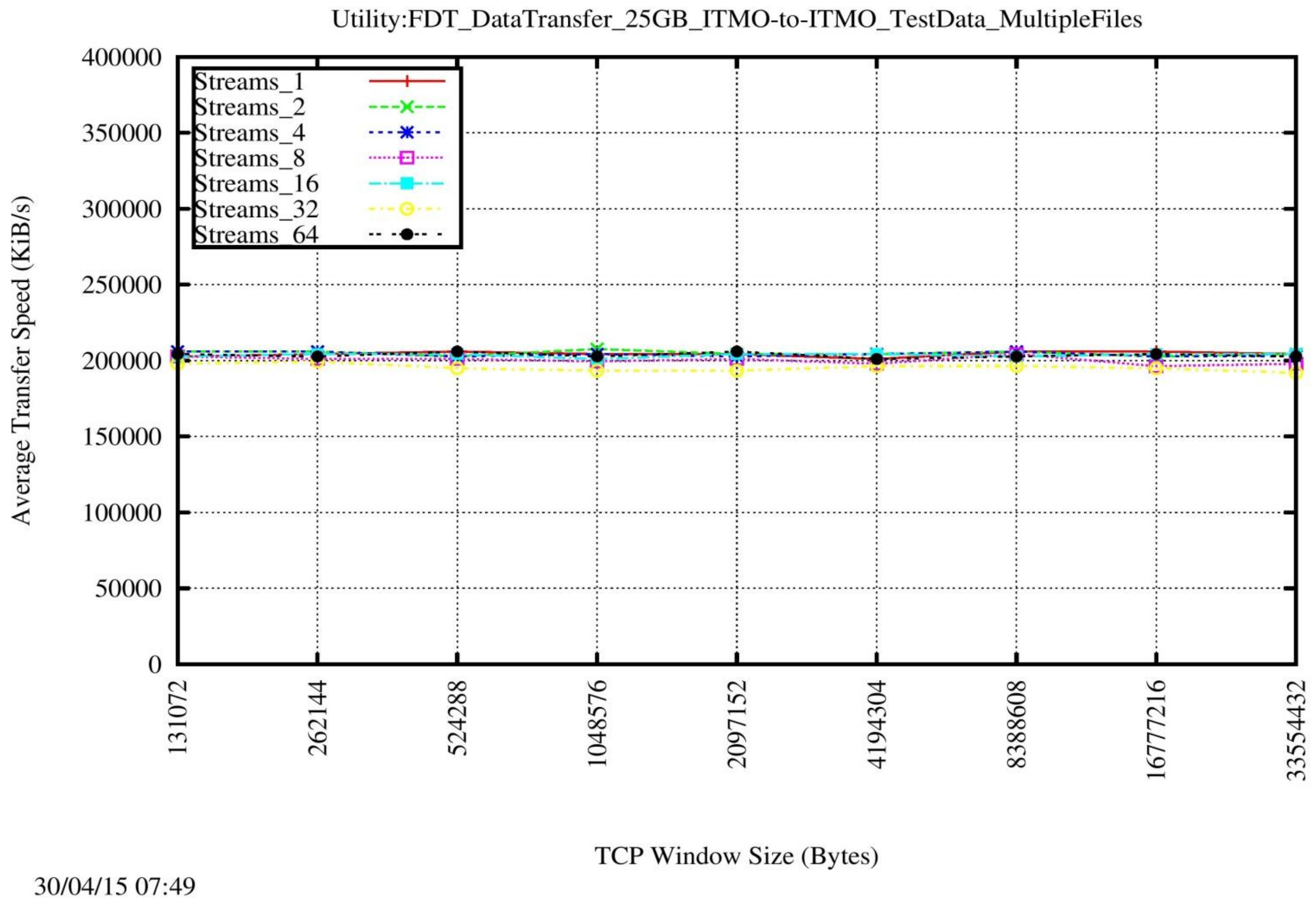


Figure 6: FDT ITMO - ITMO using HDD

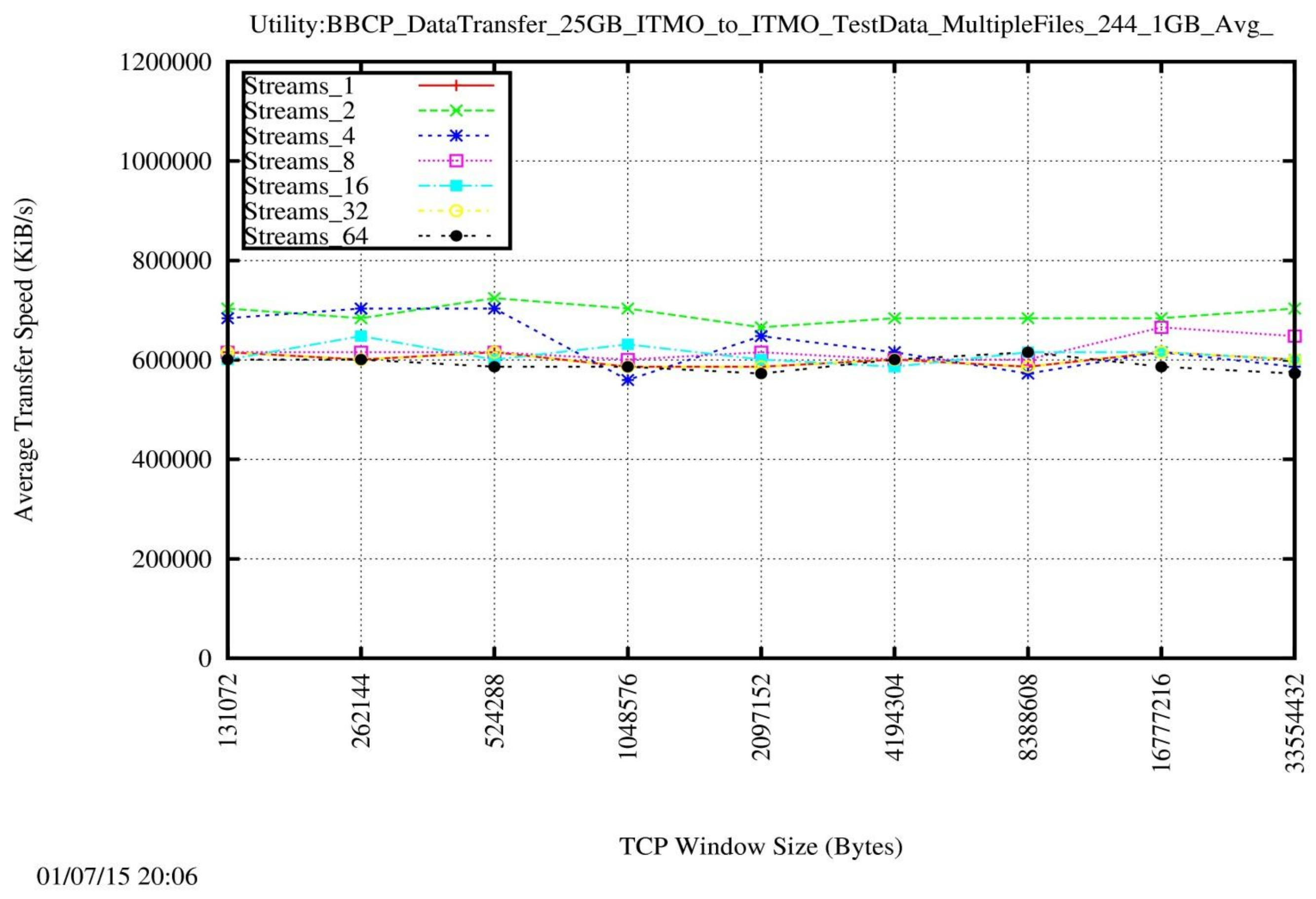


Figure 4.1.4: BBCP ITMO - ITMO using NFS

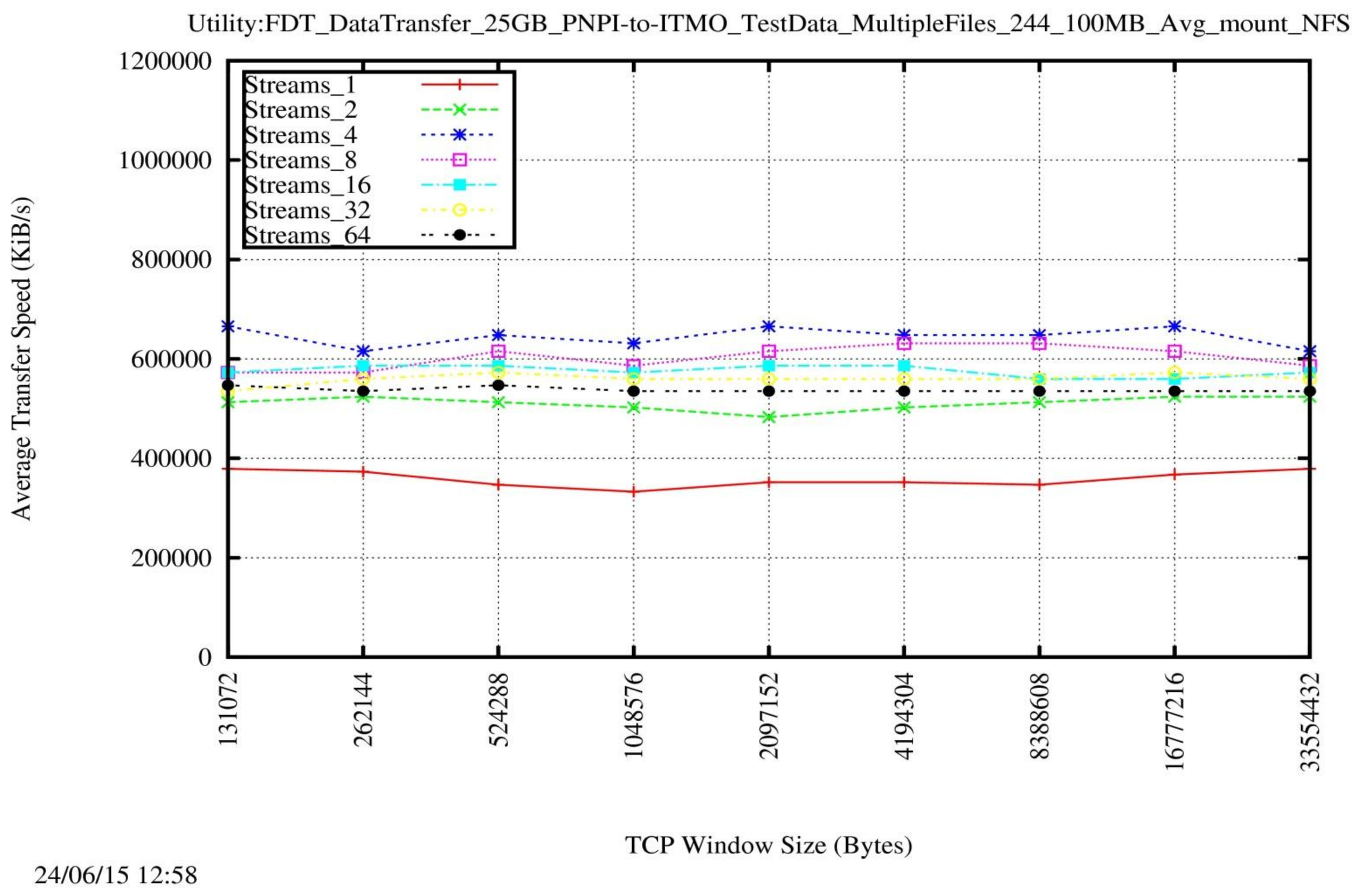


Figure 4.1.5: FDT ITMO - ITMO using NFS

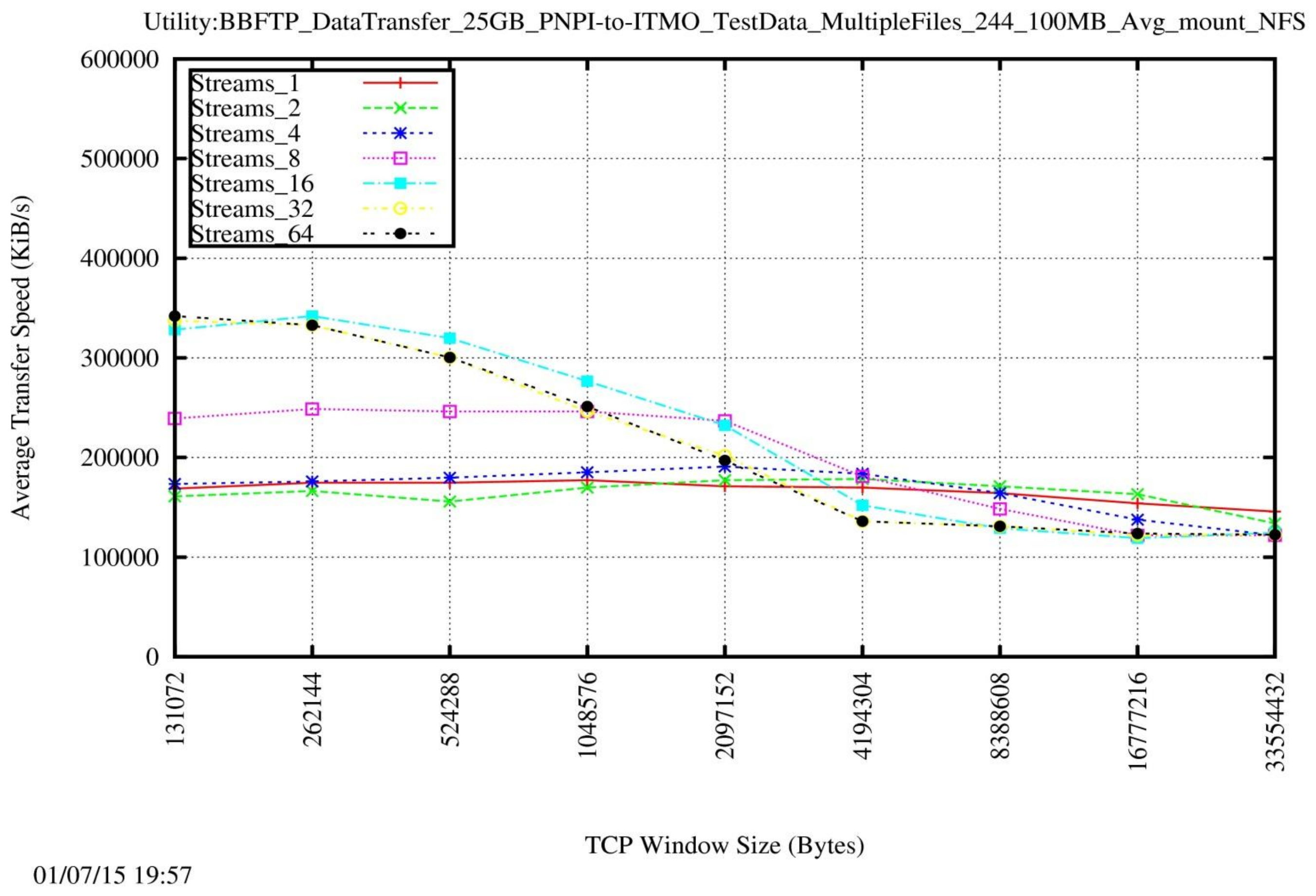


Figure 4.1.6: BBFTP ITMO - ITMO using NFS

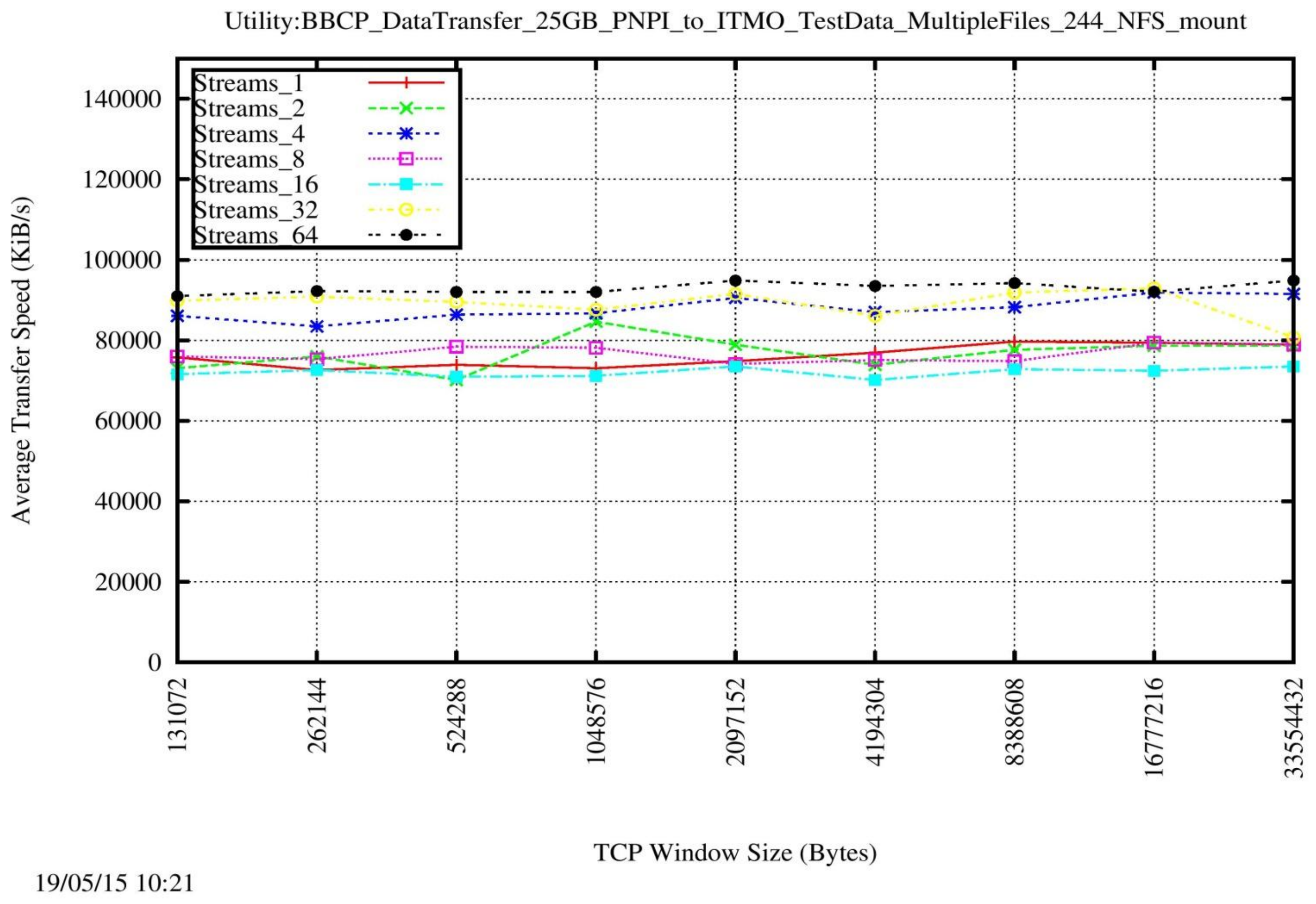


Figure 4.1.7: BBCP PNPI - ITMO using NFS

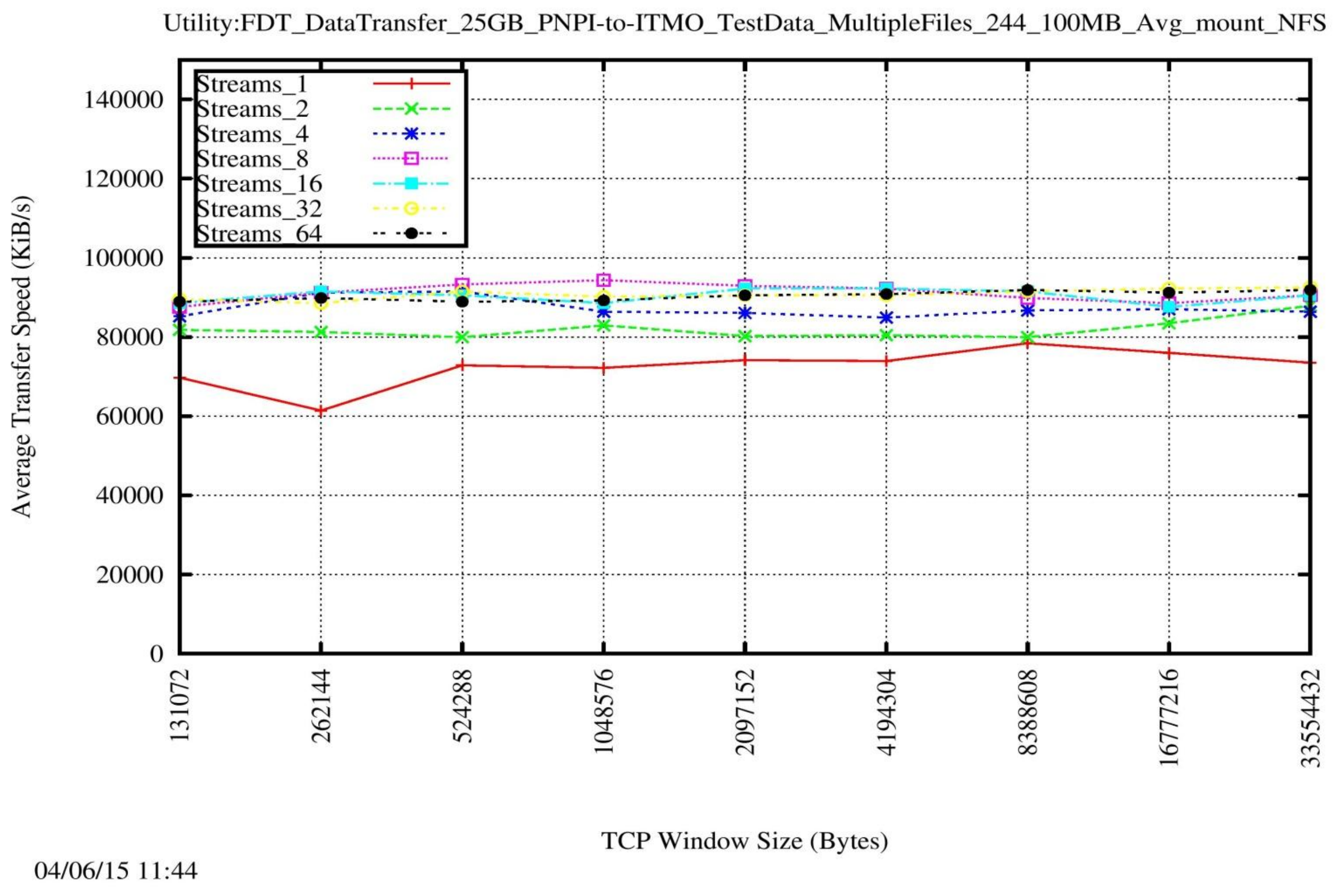


Figure 4.1.8: FDT PNPI - ITMO using NFS

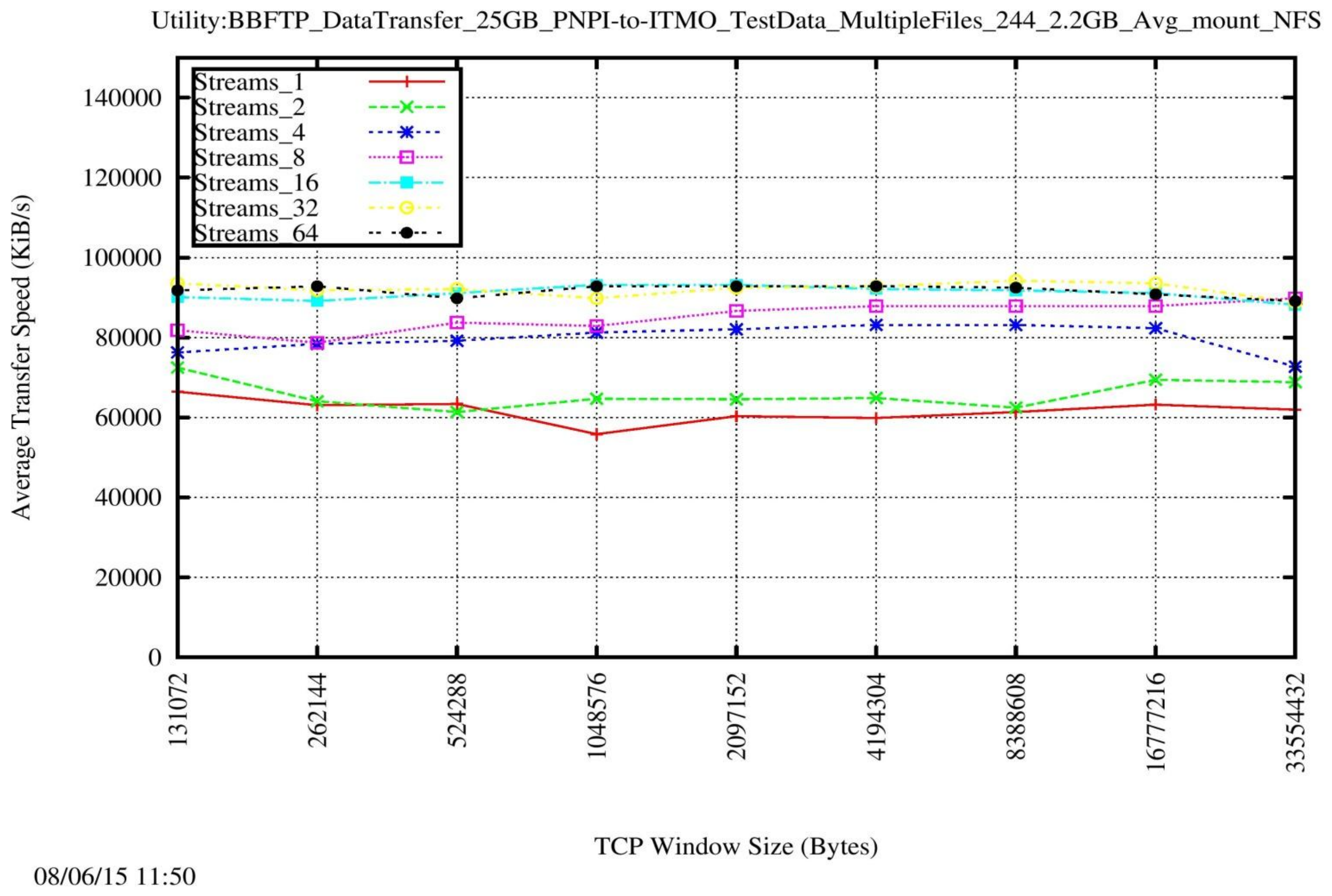


Figure 4.1.9: BBFTP PNPI - ITMO using NFS

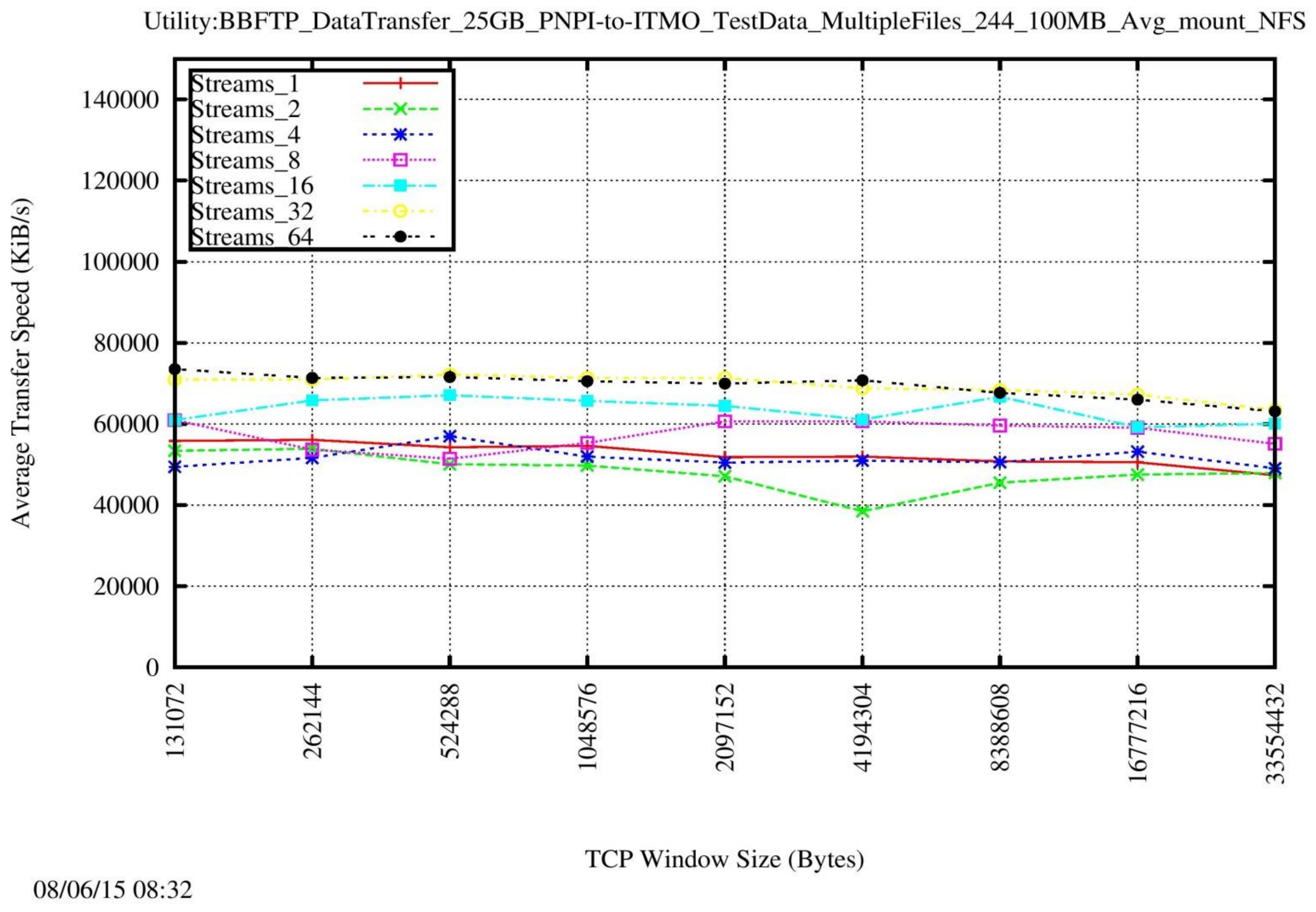


Figure 4.1.10: BBFTP PNPI - ITMO using NFS

5. DISCUSSION, CONCLUSION AND FUTURE WORK

At this part a conclusion of the current work is been given thought the results which have been acquired with explanation why we have received those results. A future work is been described for further research purposes.

5.1 Discussion

In figure 4.1.2 we can see the results we acquired while testing the BBCP utility, and in figure 4.1.3 we see the results from FDT utility. After receiving the following results we can clearly see that the maximum transfer speed which we could achieve was more or less around 200 MBps. Having these results we decided not to go any farther testing with other utilities (BBFTP, GridFTP, and FTS3) the proposed scenarios but to change our approach and having our data from NFS as explained in Chapter 3.5.2.

From the result received we can see from figure 4.1.4, 4.1.5 and 4.1.6 that the Average Transfer speed has been increased and in some cases more than 2 times. The access to the HDD comparing the NFS is showing big difference in our utilities transfer speed. Another observation which we can have based on the results it's while increasing the TCP window size and the number of parallel steams it will not give higher Average Transfer Speed. From figures 4.1.6 are the results for BBFTP utility, where we can see the average transfer speed is relatively high by using small size of buffer size , send/recv window size for streams 1,8, and 32. The same test (figure 4.1.6) we executed more than 5 times and we received similar results. In the context of BBFTP features which are provided by the utility where disabled.

While running the tests in real network which results as slower transfer speed we observed more scalable results in the aspect of increasing the numbers of parallel streams. But as expected after a certain point while increasing the number of parallel streams we cannot see any crucial changes in the average transfer speed for the results on Fig.s 4.1.7 , 4.1.8, 4.1.9 and 4.1.10. We can see from the results which was received from Fig.s 4.1.4 and 4.1.5 BBCP achieve slightly higher transfer speed from FDT which does not happened in Fig.s 4.1.7 and 4.1.8, but they achieve very similar speed. For the Fig.s 4.1.9 and 4.1.10 we experienced increased of Average Transfer Speed the moment we have increased the size each files. By increasing the files to 2,2 GB instead of 100MB each we can see that BBCP achieves higher transfer speed as mentioned by (Hector, 2014).

5.2 Conclusion

As we can see from the graphs which was acquired from the test of BBCP figure 4.1.4, FDT in figure 4.1.5 and BBFTP in figure 4.1.6 we can say for sure that HDD is one of the limitations factor which affects in a big scale the transfer rate of Big Data. By comparing the results which we received from figure 4.1.2 and 4.1.3 that the difference of the transfer rate in the most cases is more than the double. For figures 4.1.2 and 4.1.3 the utilities could achieve maximum speed of 200 MBps the most. By experiencing this fact during the test decision has been made to add of the dataset into main memory and mount NFS so other instances located in the same network could access the data in case to transfer them.

Comparing the results which was receive for figure 4.1.4 and 4.1.5 for BBCP and FDT we can see that BBCP achieves highest Average Transfer Speed compare to FDT. As we know from Chapter 3.1.1 and 3.1.2 BBCP is a utility written from C language and FDT is a java utility. As mentioned by (Mirsky, 2013) C most of the times has faster execution time compared to Java, also in C user is responsible to manage the memory which makes it also risky but on the other had faster.

As we can see from the results by having the datasets into main memory and mount by NFS we increased the transfer speed of our data which results as saving energy since our system will transmit data for less time. From the results we acquired we can see from the figures 4.1.4 and 4.1.5 that for BBCP and FDT even if we increase the TCP window size or the number of Parallel Streams it does not result as a big difference in the context of virtual environment. On the other hand it means we do not really need to allocate such big amount of resources for our virtual machines if you want to transfer over virtual data links.

From figure 4.1.6 which illustrates the BBFTP results in form of graphs we can see a quite unexpected results. As normally expected using bigger size of TCP window and buffer with more streams normally it would increase the average transfer speed. After having tested BBFTP for multiple times we experienced the same results over and over again. Streams 1,8, and 32 with lowest buffer size, send/recv window size can achieve speed of 300MBps and then while the window size and buffer size is increasing they are transfer speed is dropping. For the situation we do not have any clear answer why BBFTP is behaving that way, for that reason further investigation is necessary to understand more the utility and use it more properly.

The internal architecture (and features) of the data transfer utility affects the transfer speed. Focusing on the results of the BBFTP on Figs. 4.1.9 and 4.1.10 based on the implementation we can see by using larger data files we can see increased average transfer speed.

Any long term data transfer task would require careful study, for which utility with which parameters might help to get maximum data transfer speed.

Energy consumption cannot be calculated on the data link, but we can identify which are the parameters to achieve maximum transfer speed to consume less system resources.

The whole testbed platform which was deployed during the project can be used to compare any existing data transfer utilities or any upcoming utilities for further research.

5.3 Future Work

As described in subchapter 5.1 one of the major limitations we had during the testing was the HDD. Since we are running the results from cloud environment and we used NFS to have all the necessary datasets into the main memory we could run our tests, but the whole purpose is to run those scenarios into a real and not virtual network. For this case a raid system is required which can write/read data from multiple HDD parallel. Using the raid system it will be possible to achieve higher transfer speed for our data. By this method we can limit the obstacle of the HDD reading/writing speed.

For this thesis not all the utilities which mentioned above were tested, although is necessary for all of them to be test and be compared together, by changing more of the kernel parameters. Apart from kernel parameters more scenarios with higher volume of datasets could be tested. All the scenarios which were tested should be executed also in a real network through a public line. This test will take more time since it’s not a virtual environment and the transfer speed will affected by traffic of other users or one of the links may go down, in that case we can see more changes in the utilities behaviors.

In case more test scenarios are going to be executed in public network channel which will be more time consuming since the transfer speed will be lower and the traffic and collisions more in the channel it’s important to add mechanism to check the available system resources before launching a test. As mentioned in subchapter 1.2 if the utility will not be able to allocate the needed amount of resources the script will stack and will not be able to continue the executions. For this case a safe mechanism is necessary to be added to avoid this fault which will cost a lot of time.

Energy monitoring tool has to be developed for the Virtual machines which can be used to get information about the which utility consumes more energy. PowerTop is a Linux command which could be used by its not implemented for Virtual Machines.

As mentioned in subchapter 1.2 delimitation for the project was the fact that many users could have access on the server resources and execute tests simultaneously which would have as effect to lower the performance of the execution scripts. An online scheduling application could be implemented which will allow the users to launch different scripts to transfer datasets with different parameters. By using such an application user will be in place to know for each time which user is running tests and when to schedule their own tests so their tests will not be affected.

References

Beal, V. 2015. What is Big Data. [online document]. [Accessed 1 January 2015]. Available at http://www.webopedia.com/TERM/B/big_data.html

White, T. 2012. Hadoop: The Definitive Guide. 3rd ed. O'Reilly Media

Hilbert, M. & Lopez, P. 2011. The World's Technological Capacity to store, Communicate, and Compute Information. *Science*, Volume: 332, no. 6025, Pages: 60-65

Taylor, J. 2011. IBM What is big data? – Bringing big data to the enterprise. [online document]. [Accessed 1 January 2015]. Available at http://www-01.ibm.com/software/data/bigdata/

Hunt, C. 2002. TCP/IP Network Administration. 3rd ed. O'Reilly Media

Pillai, S. 2013. Linux Network (TCP) Performance Tuning with Sysctl. [online document]. [Accessed 5 January 2015]. Available at http://www.slashroot.in/linux-network-tcp-performance-tuning-sysctl

Tierne, B., Kissel, E., Swany, M., Pouyoul, E. 2012. Efficient Data Transfer Protocols for Big Data. 8th International Conference on E-Science, 8 – 12 Oct. 2012, Illinois, Chicago.

The Climate Group 2008. Smart 2020: Enabling the low carbon economy in the information age. [online document]. [Accessed 20 February 2015]. Available at http://www.smart2020.org/_assets/files/02_Smart2020Report.pdf

Barakat, C., Altman, E. and Dabbous, W. 2000. On tcp performance in a heterogeneous network: A survey. IEEE Communications Magazine, Volume: 38, Issue: 1, Pages: 40-46.

Jacobsen, V. Braden, R. and Borman, D. 1992. TCP extensions for high performance. [online documents]. [Accessed 9 February 2015]. Available at https://tools.ietf.org/html/rfc7323.

Barnabas, K. T., Maute, Y. Y., Ezell, M. N., Jaimes, A., Rosas, R., Motaghi, A., Kaplan, H., & Jamshidi, M. 2013. Modeling of System via Data Analytics – Case for "Big Data" in SoS. 8th International Conference on System of Systems Engineering (SoSE), 2 – 6 June 2013, Maui, HI.

Molnar, S., Sonkoly, B., & Trinh, T. A. 2009. A comprehensive TCP fairness analysis in high speed networks. ELSEVIER, Volume: 32, Issues: 13-14, Pages: 1460-1484.

Bogdan, G., Alexandru, I., & Epema, D. H. J. 2013. Towards an Optimized Big Data Processing System. 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 13 – 14 of May 2013, Delft, Netherlands.

Steinberg, R., & Pants, S. 2009. The Origin of the word Daemon. [online document]. [Accessed 29 March 2015]. Available at http://ei.cs.vt.edu/~history/Daemon.html.

Rapier, C., Stevens, M., Bennett, B., and Tasota, M. 2012. PSC/CMU High Performance Enabled SSH/SCP [PSC]. [e-mail] hpn-ssh@psc.edu 7 May 2012.

Baiocchi, A., Castellani, A., and Vacirca, F.,2007. YeAH-TCP : Yet Another Highspeed TCP. Proceedings of the 5th International Workshop on Protocols for Fast Long-Distance Networks. Pages: 37 – 42.

Duarte, P. R. 2008. Transport Protocols for Large Bandwidth-Delay Product Networks: TCP Extensions and Alternative Transport Protocols. Conference on Electronics Telecommunications and Computers, November 2008, Lisbon, Portugal.

Plankers, B. 2013. Account for the Bandwidth – Delay Product with Larger Network Buffers. [online document]. [Accessed 16 February 2015]. Available at https://lonesysadmin.net/2013/12/19/account-bandwidth-delay-product-larger-network-buffers/

Wang, H., Zuohua, T., & Qinlong, Z. 2010. Self – Tuning Price – Based Congestion Control Supporting TCP Networks, Proceedings of the 19th International Conference on Computer Communications and Networks (ICCCN), 2 – 5 August 2010, Zurich, Switzerland, IEEE.

FDT Team, 2013. Fast Data Transfer. [online document]. [Accessed 1 March 2015]. Available at http://monalisa.cern.ch/FDT/

Hanushevsky, A., 2015. BBCP. [online document]. [Accessed 1 March 2015]. Available at https://www.slac.stanford.edu/~abh/bbcp/

IN2P3 group, 2013. BBFTP [online document]. [Accessed 1 March 2015]. Available at http://doc.in2p3.fr/bbftp/

Grid Alliance, 2014. Globus Toolkit [online document]. [Accessed 1 March 2015]. Available at http://toolkit.globus.org/toolkit/

Cern IT-SDC froup, 2014. File Transfer Service [online document]. [Accessed 1 March 2015]. Available at http://fts3-service.web.cern.ch/

Gnuplot, 2015. Gnuplot homepage [online document]. [Accessed 1 March 2015]. Available at http://www.gnuplot.info/

Guazzone, M. & Anglano, C. 2015. Power and Performance Management in Cloud Computing Systems. PhD thesis. Doctoral School in Science and High Technology specialization in Computer Science, University of Torino.

Red Hat, 2014. What is OpenStack [online document]. [Accessed 4 March 2015]. Available at http://opensource.com/resources/what-is-openstack

Energy Science Networks, 2014. Linux Tuning [online document]. [Accessed 4 March 2015]. Available at http://fasterdata.es.net/host-tuning/linux/

Gibilisco, S. 2012. What is Openflow? [online document]. [Accessed 5 March 2015]. Available at http://whatis.techtarget.com/definition/OpenFlow

Vineet, M. 2012. How to SSH without password [online document]. [Accessed 5 March 2015]. Available at http://www.vineetmanohar.com/2009/07/howto-ssh-without-password/

Khoruzhnikov S.E., Grudinin V.A., Sadov O.L., Shevel A.Y., & Kairkanov A.B. 2015. Transfer of large volume of data over Internet with parallel data links and SDN. 6th International Conference on Swarm Intelligence and 2nd BRICS Congress on Computational Intelligence (ICSI – CCI 2015), June 26-29 2015, Beijing, China.

O’Luanaigh, C. 2013. CERN Data Centre passes 100 petabytes [online document]. [Accessed 20 March 2015]. Available at http://home.web.cern.ch/about/updates/2013/02/cern-data-centre-passes-100-petabytes

Jenkov, J. 2012. Java NIO Tutorial [online document]. [Accessed 22 March 2015]. Available at http://tutorials.jenkov.com/java-nio/index.html

Ayllon, A. A., Salichos, M., Simon, K. M., and Keeble, O. 2014. FTS3: New Data Movement Service For WLC. Journal of Physics: Conference Series, Volume: 513, Issue: 3 (2014) .

Sciaba, A. 2010. Critical services in the LFC computing. Journal of Physics: Conference Series 219 (2010), Volume: 219, Issue 6 (2010).

Ah Nam, H., Hill, J., & Parete-Koon, S. 2013. The Practical Obstacles of Data Transfer: Why researchers still love scp. NDM'13 Proceedings of the Third International Workshop on Network-Aware Data Management, 17 – 21 November 2013, Denver, CO, USA

Brebner, P., O'Brien, L., & Gray, J. 2009. Performance modeling power consumption and carbon emissions for Server Virtualization of Service Oriented Architectures (SOAs). 13th Conference on Enterprise Distributed Object Computing Conference Workshops, 1 – 4 Sept. 2009, Auckland, New Zealand.

Timme, F. 2013. Setting Up An NFS Server And Client on Scientific Linux 6.3 [online document]. [Accessed 15 April 2015]. Available at https://www.howtoforge.com/setting-up-an-nfs-server-and-client-on-scientific-linux-6.3

Linthicum, D. 2015. Chapter 1: Service Oriented Architecture (SOA) [online document]. [Accessed 27 April 2015]. Available at https://msdn.microsoft.com/en-us/library/bb833022.aspx

Rouse, M. 2014. Service-Level agreement (SLA) [online document]. [Accessed 27 April 2015]. Available at http://searchitchannel.techtarget.com/definition/service-level-agreement

Mitchell, B. 2015. What Is a Default Gateway [online document]? [Accessed 27 April 2015]. Available at http://compnetworking.about.com/od/ internetaccessbestuses/f/ default_gateway.htm

Mangalam, H. 2015. How to transfer large amount of data via network [online document]. [Accessed 28 April 2015]. Available at http://moo.nac.uci.edu/~hjm/ HOWTO_move_data.html

Raiciu, C., Barre, S., Pluntke, C., Greenhalgh, A., Wischik, D., & Handley, M. 2011. Improving Datacenter Performance and Robustness with Multipath TCP. SIGCOMM '11 Proceedings of the ACM SIGCOMM 2011 conference, SIGCOMM '11, Volume: 41, Issue: 4, Pages: 266-277, New York, USA.

Gunter, D., Kettimuthu, R., Kissel, E., Swany, M., Yi, J., and Zurawski, J. (2012) Exploiting Network Parallelism for Improving Data Transfer Performance. DOI: 10.1109/SC. Companion.2012.337 Conference: High Performance Computing, Networking, Storage and Analysis (SCC), November 10-16, Salt Lake City, Utah, USA.

Mirsky, I. 2013. Performance: Java Vs. C [online document]. [Accessed 19 May 2015]. Available at http://beautynbits.blogspot.ru/2013/01/performance-java-vs-c.html

Hector, 2014. HECToR: UK National Supercomputing Service [online document]. [Accessed 20 May 2015]. Available at http://www.hector.ac.uk/support/documentation/guides/bbftp/

Drouant, N., Rondeau, E., Georges, JP., Lepage, F., 2014. Designing green network architectures using the ten commandments for a mature ecosystem. Elsevier, vol. 42, pp. 38-46, 2014