Louisiana State University
LSU Digital Commons

LSU Doctoral Dissertations | Graduate School

2010

Choosing between remote I/O versus staging in distributed environments

Ibrahim Hakki Suslu
Louisiana State University and Agricultural and Mechanical College

Follow this and additional works at: https://digitalcommons.lsu.edu/gradschool_dissertations

 Part of the Computer Sciences Commons

Recommended Citation

Suslu, Ibrahim Hakki, "Choosing between remote I/O versus staging in distributed environments" (2010). *LSU Doctoral Dissertations*. 2525.
https://digitalcommons.lsu.edu/gradschool_dissertations/2525

This Dissertation is brought to you for free and open access by the Graduate School at LSU Digital Commons. It has been accepted for inclusion in LSU Doctoral Dissertations by an authorized graduate school editor of LSU Digital Commons. For more information, please contactgradetd@lsu.edu.

CHOOSING BETWEEN REMOTE I/O VERSUS STAGING IN DISTRIBUTED ENVIRONMENTS

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Department of Computer Science

by
Ibrahim H. Suslu
B.S., Marmara University, Istanbul, 1993
M.S., Marmara University, Istanbul, 1995
M.S., Southern University, Baton Rouge, 2001
August, 2010

Acknowledgments

The following dissertation, while an individual work, benefited from the insights and direction of several people. First, my dissertation chair, Dr. Tevfik Kosar, exemplifies the high quality scholarship to which I aspire. In addition, Dr. Kosar provided timely and instructive comments and evaluation at every stage of the dissertation process, allowing me to complete this dissertation on schedule. I would like to thank all my advisory committee members, Dr. Jim Van Scotter (Minor Professor), Dr. Gabriel Allen, Dr. Bijaya B. Karki, and Dr. Sherif Ishak (Dean Representative).

Next, this dissertation would not be possible without several contributions. It is a pleasure to thank my current colleagues at the Distributed Systems Laboratory (DSL) and CCT (Center for Computation & Technology) for providing me with a pleasant working environment and friendship. I also would like to thank all my collaborators from LONI, TeraGrid, Gilda, and CCTOOLS project team.

In addition to the technical and instrumental assistance above, I received equally important assistance from family and friends. My wife, Dilek Suslu, provided on-going support throughout the dissertation process, as well as technical assistance critical for completing the dissertation in a timely manner.

This project is in part sponsored by the National Science Foundation (NSF) under award numbers CNS-0846052 (CAREER), CNS-0619843 (PetaShare), OCI-0926701 (Stork) and EPS-0701491 (CyberTools), by the U.S. Department of Energy under Award Number DE-FG02-04ER46136 (UCoMS), and by the Board of Regents, State of Louisiana, under Contract Numbers DOE/LEQSF (2004-07) and NSF/LEQSF (2007-10)-CyberRII-01.

Table of Contents

List of Tables

List of Figures

Abstract

Today, scientific applications and experiments have become increasingly complex and more demanding in terms of their computational and data requirements. The amount of data generated and used has grown at a very rapid rate. As tens or hundreds of terabytes of data for a single application is very common today; petabytes and even exabytes of data will be very common in a few years. One of the major challenges in distributed computing environments is how to access these large datasets remotely over the network.

Data staging and remote I/O are the most widely used data access methods for distributed applications. Application developers generally chose one over the other intuitively without making any scientific comparison specific to their applications since there is no generic model available that they can use.

In this thesis, we develop generic models and set guidelines for the application developers which would help them to choose the most appropriate data access method for their application. We define the parameters that potentially affect the end-to-end performance of the distributed applications which need to access remote data.

To achieve our goal, we implement a series of synthetic benchmark applications to simulate different data access patterns. We run these benchmark applications on different distributed computing settings with different parameters, such as network bandwidth, server and client capabilities, and data access ratio. We also use different remote I/O protocols to show the importance of the protocol in making a decision. We use regression analysis to develop applicable generic models for comparing different data access methods, and test our models in a real life application.

The main contribution of our thesis is generic models that can be applied to most data-intensive distributed applications to decide the best data access technique for those applications. Our models provide the scientists and application developers an opportunity to choose the best data access method before actually running the application.

Chapter 1
Introduction

Distributed resources and collaboration between multiple institutions have been inevitable with the increased computational and data requirements of scientific applications. This has led to computer and data resources being shared over widely distributed systems. Usually, data is no longer locally available to the distributed application, and the application developers need to find efficient ways to access distributed and remote data resources.

Data-aware distributed applications usually take place in two phases: data generation/collection and data analysis. In the data generation/collection phase, large amounts of data are generated by applications running on distributed resources or collected from remote instruments. In the data analysis phase, collected data is analyzed and a bigger amount of data may be generated. So, these require reliable and efficient data access mechanisms that can keep up with the volumes of data involved.

There are mainly four different data access techniques for distributed computing environments. These are: (i) remote I/O, (ii) moving application close to data, (iii) moving data close to the application (staging), and (iv) moving both data and application to an intermediate location (hybrid model). Each data access technique has advantages and disadvantages based on the characteristics of the environment and applications. Therefore, based on application needs and distributed environments, an appropriate data access technique should be used.

In most cases, the datasets required by the application are either transferred to a temporary space close to the computation site (staged-in) or accessed remotely over the network (remote I/O). Due to the nature of applications and the interconnects between distributed components, there are many factors affecting end-to-end performance such as I/O access pattern of the application, size of dataset needed, and network characteristics.

In **data staging**, the input data for the application is transferred as streams or files from the remote storage to a location close to the computation node. This process is called the "stage-in"

step. After the computation is finished successfully, the results are generally transferred back to a remote storage site, which is called the "stage-out" step. Separating the stage-in and stage-out steps from the computation allows all data access during the computation to be performed on the local disk.

On the other hand, some distributed applications prefer using **remote I/O** to directly access (read and write) data on the remote storage. Remote I/O does not require extra steps for staging the data in and out, but it slows down the actual computation step since the computation now has to use a remote resource for I/O instead of using the local disk.

Although both data access techniques are widely used in data intensive distributed applications, the application developers generally choose one over the other intuitively without making any scientific comparison specific to their application since there is no generic model available that they can use. To the best of our knowledge, there has not been an extensive study comparing both data access techniques and providing clear guidelines of which method to use in which particular case.

1.1 Overview

In this thesis, the goal is to develop models and set guidelines for the application developers which help them to choose the most appropriate data access method for their application. In this work, we define the parameters that potentially affect the end-to-end performance of the distributed applications which need to access remote data. We have developed some synthetic applications to simulate the defined parameters. We have run the applications on different distributed environments and collected the data for the models. We have used regression analysis for modeling, and analyzed the defined variables. After the regression analysis, we came up with one regression model for each remote data access technique.

We have also used different remote I/O protocols to show how important it is to use the appropriate remote I/O protocol. We have tested our model in a real world application.

1.2 Major Contributions

This thesis contributes to the distributed systems community in several ways.

The main contribution of our thesis is: *"development of generic models that can be applied to most data-intensive distributed applications to decide the best data access technique for those applications."* Since data-aware distributed applications spend most of the time accessing the data before and after the computation, choosing the best way to access the remote data is imperative. Our models provide an opportunity to choose the data access method before running the application. Application designers can use our models to develop their application when they decide which data access technique is right for them.

The current approach to find the best data access method is based on active learning. First, the application needs to be run in the same environment with all possible combinations. Once the combination that is the best fit for that environment is discovered, the correct data access technique can be found. Our models, however, provide best data access technique before running the application.

1.3 Thesis Outline

The rest of this thesis is organized as follows. In Chapter 2, we present an overview of background and basic concepts about remote data access techniques. In Chapter 3, we present an overview of related works. Our overall methodology is described in Chapter 4. Experiment design is presented in detail in Chapter 5. We present our experiment results with the analysis in Chapter 6. Our models are presented in Chapter 7. We have included the implementation results in Chapter 8. We apply our models to a real-life application in Chapter 9, and conclude our work in Chapter 10.

Chapter 2
Background and Basic Concepts

2.1 Data Staging

Staging the data means moving the data closer to the application before the actual processing starts. Staging makes the remotely placed data available to the computation node for processing. This involves placing the input data sets either into the local disks of the computation node or to a location close to the computational node. Generated output is moved back to remote storage after the actual processing finishes.

Application developers use different tools and techniques for data staging. The following subsection will cover the data staging tools.

2.1.1 Staging Techniques

GridFTP [27] is a widely used transport protocol on distributed environments. It is an extension of the default FTP [68] protocol and provides features of security, reliability and efficiency for the distributed computing environments. It also allows changing the sizes of the TCP buffers and congestion windows to improve transfer performance. FTP divides the process of data transfer into two channels which are: i) the control channel used for sending commands and replies between a client and a server ii) the data channel through which the actual transfer takes place.

The Reliable File Transfer (RFT) [59] tool from the Globus [78] project uses GridFTP protocol to stage in or out multiple files simultaneously. RFT provides features like failure detection and restart of file transfer. The Lightweight Data Replicator (LDR) [19] can replicate data sets to the member sites of a Virtual Organization or DataGrid. It was primarily developed for replicating LIGO data, and it makes use of Globus tools to transfer data. Its goal is to use the minimum collection of components necessary for fast and secure replication of data.

The Internet Back Plane protocol (IBP) [66] allows applications to optimize data transfer and storage operations with a store-and-forward protocol to move data around the network. Kangaroo [80] is another data movement protocol which manages data movement as a background process so

that failures do not affect the application but are handled transparently by the underlying monitoring system.

Stork [53] is a specialized scheduling tool which aims to make all data movement related tasks (including data staging) first-class entities in a distributed environment. Stork can schedule, manage, and monitor data movement. It supports multiple protocols, storage and network reservations before data transfers.

PBS [49] transfers files between user accounts. As in all staging techniques, the user should specify the files to be transferred, and transfer files back when the computation is done. Meanwhile, Legion [92] supports on-demand transfer. Also, Punch virtual file system (PVFS) [44] supports on-demand file transfer without requiring dynamically-linked libraries. GASS [35], a data movement and access service for wide area computing systems, transfers an entire file when it is needed. NeST [33] is a Grid enabled storage appliance. NeST supports for restricted subset of NFS protocols and supports for anonymous accesses is the differences among others. Unlike the other file systems, SFS [60], a secure file system that avoids internal key management, needs key management to a map file.

2.2 Remote I/O

Remote I/O allows accessing data directly on the site where it is located without moving the complete data sets. Remote file access can be divided into four categories: i) distributed file systems, ii) parallel file systems, iii) remote execution systems, iv) remote I/O.

2.2.1 Remote I/O Techniques

Some distributed file systems such as NFS [72], and AFS [62] as well as parallel file systems such as GPFS [63], Lustre [15] and Vesta[42] provide mounted file system solutions for remote file access. They allow a process to access remote files over a network as easily as if the files were on its local disks. All of these central-server models have limited scalability in wide area because the number of clients is limited by the aggregate bandwidth provided by the central server. Distributed file systems have performance and administrative problems which often render them inappropriate for distributed computing applications [45]. Distributed file systems mount the source directory, and it is often undesirable for all file systems to be cross mounted on all machines, or even impossible

to cross mount specialized file systems such as tape archives. Even when disks are cross mounted, performance may be poor when using conventional I/O techniques [45].

There are also web-based distributed file systems such as WebFS [89], and UFO [25], which have improved implementation and lowered administration costs, but they have problems with performance [45]. The network disk servers such as DPSS (Distributed Parallel Storage Systems) [85] and MARS (Massively-parallel and Real-time storage) [39] provide high-speed streaming access to distributed data, but they do not provide support for access from parallel programs.

Parallel file systems such as GPFS (Galley Parallel File systems) [63] and Vesta parallel file systems [42] are designed to provide parallel file access to application programs running on multiple computers with parallel I/O subsystems.

Remote execution systems redirect Unix file system calls to a home file system. It enables a remote computer to execute a task and access the remote data. Condor [58] and WebOS [89] are examples of these kind of systems. WebOS, however, does not support parallel I/O interfaces to access the parallel file systems [45].

Specialized remote I/O systems provide high-performance I/O libraries and a unique environment for distributed computing environment. One of the remote I/O implementation is RIO [52]. RIO uses client server architecture and ADIO (Abstract Device I/O) [82] for portability in ROMIO [83] which is a popular implementation of the MPI-IO specification in the MPI-2 standard, so it hides different file system implementation details. RIO requires a certain processor configuration that can cause inefficiency and relies on a legacy communication protocol [57].

RFS (Remote File System) [56] is another implementation of remote I/O which uses dedicated forwarder nodes for message aggregation and asynchronous I/O. There are also some I/O libraries such as Jovian [32], MPI-IO [41] which solve the performance issues by optimizing collective I/O operations, but they are not designed for wide area network or distributed computing environment.

Parrot is a tool that enables remote I/O operations on the selected data set [81]. It supports a variety of storage systems and acts as an interposition agent between the data consumer and the storage server. One of the storage systems Parrot supports is Chirp. Chirp enables the users to create distributed file systems without requiring special privileges [79]. Although it causes performance

degradations, Parrot is described as a very convenient remote I/O client especially for the Chirp file system.

Information Power Grid (IPG) [48] has been working to build a fully distributed computing and data aware management environment by NASA [20]. It includes a network-aware MPI that supports remote I/O, grid communication libraries. A run time library SRB-OL [74] provides various optimizations for HPSS [5], tertiary storage access, such as collective I/O, data sieving remote I/O optimizations. Simultaneously PASSION [40], PANDA [73] also provide collective I/O and data sieving on top of parallel file systems for many popular access patterns. But, these systems cannot scale well when the application size increases.

xFS [28] and Coda [16] are other distributed file systems that provide easy access to distributed resources. When high performance parallel data access is required by parallel applications, they have performance problems. MAPFS [65], a flexible multi-agent parallel file system for clusters, allows applications to access remote data in a flexible and efficient way with flexible I/O architecture. BLUNT [71], like DataCutter [36], and DPSS uses replication to improve I/O performance and reliability.

2.3 How Researchers Choose

Application developers generally choose one of these data access techniques for their applications intuitively, based on their past experiences or their expectations.

2.3.1 Reasons to Favor Remote I/O

In [52], Kohr et al. say that remote I/O allows programmers to execute at remote sites without programmer management of data transfer, and remote I/O can improve performance relative to staging by overlapping computation and data transfer or by reducing communication requirements. According to them, remote I/O simplifies matters (Check-point/Restart) when computation may be moved between computers, and it has the advantage of making intermediate results available before execution is complete. Kohr et al. also say that staging is clumsy, prevents overlapping of communication and computation, and can result in excessive data transfer in situations where a program accesses only part of a file.

Bent et al. claim in [34] that remote I/O can improve turnaround time relative to staging. They also say that when using remote I/O, only needed data is transported across the network which minimizes the network traffic and improves throughput. According to Bent et al., another advantage of Remote I/O is that the throughput of a data-intensive workload will be drastically reduced.

In [38], Blanchet et al. claim that remote I/O does not have to worry about the free local space when dealing with large data sets. Blanchet et al. also claim that remote I/O does not need to know about non-declared input and output files.

Lee [57] et al. claim that supporting remote I/O via MPI-IO enables many applications to perform remote I/O transparently without code changes. Lee et al. say that staging can cause consistency problems, and excessive data transfer for partial file access. According to them, staging is often done manually and this is not convenient.

According to Blanchet [38] et al., there must be enough free space on the local storage area of the computing node for input data, and generated output data when doing staging. Blanchet et al. also claim that the agent has to identify which program will be launched, and to transfer the non-declared files. That means to bring knowledge about the program within the agent.

2.3.2 Reasons to Favor Staging

In [66], Plank et al. say that locality is important for reducing data access overhead. They also claim that staging data near where it is used improves application performance.

Wang et al. claim in [90] that when managing I/O in a client-server model where the data is written to/read from that are located on a remote site, parallel access to remote datasets can be extremely expensive and will hurt the overall performance of the user applications.

Thain et al. say in [80] that due to the large number of users, the size of the data, and the distance involved, remote I/O is not universally applicable and suffers from a scalability problem. Network and storage capacities limit both the number of replicas that may be made as well as the number of jobs that may use each replica.

In [26], Ali et al. claim that even on a high-bandwidth wide area network, local data access is at least an order of magnitude faster than remote I/O which would make advanced staging of data advantageous.

2.3.3 Remark

Some of these claims may be true, and some may not. In any case, most of them are based on experience and intuitions, not scientific reasoning and analysis. In this work, we develop scientific models which can be applied to most data intensive distributed applications to decide the best data access model for them.

Chapter 3
Related Work

In this chapter, we provide the related work which compares staging and remote I/O. Then, we highlight the studies on the modeling staging and remote I/O to point out similarities and differences. In the end, we list some performance modeling tools which can be used in a distributed environment to analyze different data access techniques.

3.1 Comparison of Data Staging and Remote I/O

According to Stockinger [76], the entire resource selection problem requires detailed cost models with respect to data transfer. A cost model for data-intensive applications is discussed in [77] where theoretical models for data intensive job scheduling are presented. In this study, a cost model is created that can determine if it is more efficient to transfer the data to a job or vice versa. The metric for measuring efficiency is the effective time seen by the client application. The model includes all important factors in a distributed Data Grid and takes various storage and access latencies into account to determine optimal data access.

More general performance engineering approaches are discussed in [55]. In this work, they analyze a typical Grid system and point out performance analysis aspects in order to improve the overall job execution time of the system. They give a detailed look at the following two domains: data access and networking.

Data staging and remote I/O have been compared by different studies from different aspects. For instance, GridFTP as a staging technique and RFIO as a remote I/O technique have been compared by Kalmady and Tierney [50] on wide area networks. According to Kalmady and Tierney, the performance of RFIO is better than gsiftp for one stream. gsiftp performs better on multiple streams. With tuning, RFIO becomes pretty close to gsiftp which means that proper tuning can make the difference. They came up with the following observations: i) setting the TCP buffer size a proper value is the most important factor for a good performance; ii) 2-3 parallel streams will gain an additional 25% performance over a single tuned stream; iii) a mechanism of dynamically varying the

buffer sizes during data transfer is needed, because of sensitivity of the variation in network traffic; iv) the same throughput can be gained with tuned buffers using untuned TCP buffers with enough parallel streams;

White et. al [91] compared some Legion data access techniques Legion, Legion basic IO, and Legion nfs with Globus. The Legion provides remote data access capability, and work has showed that Legion has around 60 percent better write, and 85 percent better read performance over FTP. On the other hand, for transfer sizes less then one MB, Legion performance suffers. GASS stages remote file to a locally-accessible place. According to the study, Legion basic I/O interface outperforms again GASS. The Legion I/O model offers an opportunity for files to gain specific file access patterns. This could be done by attribute value tags, such as read-only or single-writer. This property improves the performance significantly.

Thain and Livny compared performance of parrot/chirp with other staging and remote I/O techniques on the study [81] with Andrew-like benchmark. According to the authors, separating computation from storage makes I/O cost high. Copying data gets slower over the network, but the slowdown in the network makes staging acceptable because of increased throughput via remote parallelization. Also, authors point out that the differences in the performance between Chirp, ftp, and NeST are because of the cost of the metadata lookups.

3.2 Modeling the Grid Environment and the Other Modeling Techniques

A Grid environment has two major challenges for modeling. The first challenge is the very complex infrastructure that the Grid environments have: Grid resources are heterogeneous, and may be distributed over a wide area; Grid applications may have strict service requirements; and the Grid network infrastructure supports different traffic protocols. Therefore, the Grid environment is very difficult to model and analyze. The second challenge is that the modeling tools have limitations for distributed performance analysis.

Realistically, widely distributed Grid cannot be analyzed with a single performance modeling tool. Tools like GridSim [70] may be appropriate for some Grid components, however, they are hard to scale to the number of events that a wide area Grid will generate, even if parallel or distributed

extensions are used [46]. Simulating a single job submitted to wide area Grid may produce thousands of events. Also, analytical techniques such as queuing theory [88] and Markov processes may be appropriate for modeling some Grid components at an abstract level, but cannot model a dynamic and multi-component Grid in great detail. Clearly, new modeling methods and techniques are required to analyze the Grid environments.

Modeling the data access techniques in a distributed Grid environment is not a well-studied area. Staging the data before it is needed can be very complicated because of the dynamic nature of the distributed Grid environment such as real-time network traffic and congestion. Limitations of bandwidth, storage space at certain sites, and data specifications can affect the staging of the data.

A mathematical model for a basic data staging problem is studied by Theys et al. [84]. All parameter values for the network and data request a stay fix in the scheduling process. However, the parameter values for the model are changed temporarily to reflect the dynamic nature of the distributed environment. The parameters can be categorized by the node storage capacity and node number, the link availability starting and ending time, bandwidth, latency, source node, and destination node. Also, every request has data size, list of sources, and list of destinations. Elwasif et al. [43] developed a model for farming applications with and without server side staging to analyze the effect of staging, and verify the model with experiments and simulations.

A remote I/O performance model [75] can estimate remote I/O cost before performing the application, so the application can be evaluated better. The paper also presented the design of a remote I/O performance predictor that gives the user a concept how much remote I/O costs for the application, so the appropriate parameters can be chosen for the application. A practical remote data access model is presented in [29] which describes a tightly coupled, a data oriented infrastructure approach with building a leading edge technology to provide very high-speed, and widespread access to large data storage. Antoniu et al. [30] provide a transparent data access model which enables users to access data via global identifiers. This model manages data persistence dynamically and transparently in distributed environment using two approaches which are distributed shared memory and peer to peer systems.

Shivam et al. [31] present the Non-Invasive Modeling for Optimization (NIMO) system which automatically learns cost models for predicting the execution time of computational science applications on distributed environments. NIMO first generates training samples for distributed applications, then by using these samples learns cost models with statistical learning techniques. NIMO is an active system, so it deploys and monitors the sampled application under different conditions. NIMO is also non-invasive so it collects training data from passive streams without effecting not only the operating system or the application, but also the application source, or library.

On the other hand, there are some challenges that arise based on NIMO. According to Shivam et al. [31], sampling acquisition may have high overhead. Also, the number of samples needed, given the level of accuracy, increases exponentially. The training sample set may not represent the entire operating range of the system.

Although our approach and NIMO can suggest which data access method to use, there are significant differences between them. Researchers can use our approach before the design phase, so they can design the application based on the best data access technique. NIMO does not only evaluate the data access methods, but also the active system elements such as CPU, cache, and I/O system behaviors. Sometimes, NIMO can prefer a data access technique which is worse than others in terms of overall throughput, but preferable in terms of other active elements such as CPU, cache, and I/O.

Performance Prophet [67]is a performance modeling and prediction tool for parallel and distributed environments. The main advantage of this methodology is reducing the time needed to evaluate the model by model simplification and the combination of mathematical modeling with discrete event simulation. The performance model is generated based on the UML (Universal Model Language) [10] model.

Performance Prophet mainly contains two components: i)Teuta Estimator, ii) Performance Estimator. Teuta is a UML-based modeling platform independent tool for distributed jobs. Teuta is developed with Java programming language based on the Model-View-Controller (MVC) paradigm [54]. The main specification of the MVC is to separate the user interface and the rest of the application. Teuta mainly consists of model checking component and a model traversing component. The model checking component checks the correction of the model. The model traversing compo-

nent provides a way to walk through the model and access the model properties. The Performance Estimator evaluates the performance of the distributed program on the running system. A set of C++ classes is needed to be developed to model basic program and machine components. Teuta and the Performance Estimator communicate via Data Repository. Data Repository is implemented on PostgreSQL [7] open-source relational database system.

3.3 Performance Modeling Tools

Distributed Grid environments need advanced measurement tools and techniques. Each environment has its own structure and requirements, so it is getting harder to develop a unique performance tool that is essential for end-users, developers, administrators, and researchers.

Distributed environment measurement tools can be categorized into two categories: i) fully Grid enabled tools, ii) partially Grid enabled tools. In the next two subsections; we evaluate some of these tools.

3.3.1 Fully Grid Enabled Performance Tools

Performance tools in this category are specially designed and developed for distributed Grid environments. Most of them have started for a particular need of a particular project, and this makes most of them not suitable for general use. For instance, some measurement tools do not require security measurement, but in some distributed applications security may be a very important issue.

NetLogger [86] (Networked Application Logger) is a package that contains a set of tools that can be used to monitor the behavior of all aspects of the applications, operating system, host, and network. For example, it contains a tool that generates time-stamped event logs. These logs can be used to find out the application end-to-end performance. It also has tools to visualize the logged data with a real time stamp. Netlogger can be applied to different types of distributed system environments and is independent of any particular architecture. However, Netlogger has been used only in loosely-coupled architectures so far.

NetLogger has two phases: i) event logging and ii) log manipulation. In the first phase, NetLogger logs as much raw information as possible about the state of the system. In the second phase, NetLogger has tools to generate and manipulate the logs. NetLogger event logs have high-resolution,

synchronized timestamps. Logs need to be taken before and after each interested event which may be application, operating system activity, network related activity, or network condition.

NetLogger generates a very huge amount of logging data, so all logs should be in a common logging format. NetLogger follows the IETF draft standard Universal Logger Message format (ULM) [9]. A sample NetLogger ULM event:

ts=2008-11-22T20:14:15.507306Z event=doit.start level=INFO index=0

This indicates that event named "doit" started on a particular time.

Logging events on different systems need to be synchronized. All systems that are involved in the event are synchronized by the Network Time Protocol (NTP) [61]. After a log is generated, NetLogger's Toolkits can be used to analyze it. Toolkits can be categorized into three tool sets: i) a library to simplify logging (C, C++, and Java), ii) a set of operating system, managing, and filtering utilities, and iii) a set to visualize and analyze the log tools. A library has been developed for the supported languages, which are C, C++, and Java. Each library contains some routines such as open, write, and close the log file. Events can be written to the system log, separate file, or TCP port.

NetLogger has modified netstat and vmstat unix utilities to get contents of various network-related data structures and reports statistics on virtual memory, disk, and CPU activities. Also, NetLogger uses a broker which is a special type of agent that collects system state information, and filters them for clients.

NetLogger also has a graphical representation toolkit, nlv (NetLogger Visualization), for interactively viewing NetLogger event files. Nlv can be run on log files after application is finished, or it can be run in real-time to analyze live applications. NetLogger also has script tools which are written in perl for analyzing. It also can create a script that can be used with Gnuplot for graphics generation.

Zentrino [69] is an experiment management environment for performance analysis, software testing, and parameter studies. Zenturio can be used on cluster, and Grid environments as well. Zenturio is based on ZEN language which is a novel directive-based language, and user portal to simplify complex programming. ZEN enables to substitute strings for performance information. Zenturio has three main Grid services: i) a register service that registers the location of the Grid services, ii) an experiment

generator that parses files with ZEN for performance analysis, and iii) an experiment manager which compiles and manages execution on remote systems.

A graphical user portal can be used to manage, monitor, and visualize the programs and output data across multiple systems. Zentrino has been implemented by Java/Jini and support cluster jobs through PBS and grid jobs via Globus scheduler.

ZEN provides for the programmer to invoke jobs for arbitrary value range of any parameter, such as file names, compiler options, machine sizes, target systems, program variables, scheduling information. ZEN can be used to request performance metrics, such as execution, communication, load balance, cache, and time synchronization information even for specific needs, such as procedures, loops, statements, and the entire program.

Zentrino receives directives via ZEN files, such as the application source file, input files, output files, comments, and host name. Zentrino runs ZEN files through ZTS (ZEN Transformation System) which parses ZEN files and generates a set of ZEN file instances. Thereafter, it transfers, compiles, and runs the jobs on the target system. Staging the data is optional, and the application can use remote I/O to access remote data. After the job has completed, the output data can be staged out from target systems. A GUI-based portal system enables the process to be managed, monitored and visualized remotely.

3.3.2 Partially Grid Enabled Performance Tools

Unix environment provides some benchmarking tools for the Grid users to monitor and tune their applications. These tools can be divided into two categories: i) synthetic benchmarking tools, and ii)application benchmarking tools.

3.3.2.1 Synthetic Benchmarking Tools

Synthetic benchmarking tests are simple tests to measure a certain aspect of the performance. Such as ttcp [21]: measures the point-to-point bandwidth over a network connection, and hdparm [14]: set and view hard disk hardware parameters. "-t" and "-T" options can be used to measure disk-to-memory (disk reads) transfer rates. Synthetic benchmarking tools used in distributed environments: i)UnixBench [2] is a fundamental high-level Linux benchmark suite. Unixbench integrates CPU and file I/O tests, as well as system behavior under various user loads; ii) AIM9 [12] (Independent Resource

Benchmark) can test and measure each component of a UNIX computer system independently. The AIM9 benchmark uses some subtests to generate absolute processing rates, I/O transfers, function calls, and UNIX system calls; iii) Netperf [6] is a sophisticated and well known network and filesystem benchmarking tool; iv) IOzone [18] is useful for performing a broad filesystem analysis. The IOzone benchmarks file I/O performance operations, such as read, write, re-read, re-write, and random read. More information can be found in Chapter 5.

3.3.2.2 Application Benchmarking Tools

Application benchmark tests are sophisticated tests to measure all aspects of the performance. Application benchmarking tools are used in distributed environments: i) High Performance Group (HPG) [8] set of benchmarking tools created for high-performance computers by The Standard Performance Evaluation Corporation (SPEC). These tools use industry standard parallel application programming interfaces (APIs), OpenMP and MPI. They also support shared-memory and message passing programming paradigms. HPG benchmarking suites contain MPI Benchmark Suite which measures performance of compute intensive applications using the Message-Passing Interface (MPI) across a wide range of cluster and SMP hardware, and OMP Benchmark Suite which evaluates the performance of OpenMP applications and shared-memory systems; ii) The NAS Parallel Benchmarks (NPB) [20] are a set of benchmarks targeting performance evaluation of highly parallel supercomputers. They are developed and maintained by the NASA Advanced Supercomputing (NAS) Division based at the NASA Ames Research Center; iii) OProfile [4] OProfile is a system-wide profiler for Linux systems. OProfile can profile all running code at low overhead. Many CPUs provide hardware registers that can count events; such as, cache misses, or CPU cycles. OProfile provides profiles of code based on the number of these occurring events.

Chapter 4
Methodology

Performance analysis of the distributed systems can be divided into three stages: i) observation and experimental results ii) performance framework and analytical simulation model, and iii) a more realistic model and theory.

For the first stage, we have created synthetic programs to simulate the effects of different parameters that will be described later in this chapter. We define the parameters that can potentially affect the performance of the distributed data intensive applications. The application specific values for these parameters should be provided by the user. Our model will evaluate these parameters not only individually, but also as a combination of related ones to find out which technique is the best for the application.

The parameters of interest are listed below:

- *Data size* of the application. This can be both input data needed as well as the output data generated by the application. Four different data sizes are categorized by our models which are full, half, quarter, and eighth.

- *Proportion* of data needed by the application versus the size of the entire dataset.

- *Disk Speed* shows how fast local vs remote disk is. It is related to both input and output data characteristics.

- *Total turnaround time* is the elapsed time between the submissions of the first task until the last task is completed.

- *Data access pattern* is the way of accessing the data by the application. We use three different types of data access patterns which are sequential, jump, and random.

- *Network architecture* is the design of a communication network where the application and the data take place. We have used ten different types of network architecture on the model.

- *Bandwidth* is the network data transmission rate.

After we collect the parameter results, we develop two analytical simulation models for each data access technique. We use statistical techniques to capture application performance with data access methods accurately.

The model we design suggests the best data access strategy in different scenarios for the application, data and the resources being used. To keep it simple, we are assuming that we know the resources that are being used. We might initially know the proportion of read and write, and their size involved from previous simulations, but the access patterns of the application and the requirements of the actual data needed for the computation versus the input data set being used are not easily known in advance. However, the details of the data access characteristics of an application and required subset of the data from an entire dataset are important for making any kind of suggestion by the model.

The model would rely on the following functions and strategies:

- Analyzing the application characteristics, data characteristics (input, intermediate and output), and resource characteristics (actual hardware resources available or selected and their capacity and performance estimates).

- Listing out the sequence of elementary events related to data access and writes during the initial application run, timely availability of the data that can be staged out.

For the last stage, we test the models with real world application.

Chapter 5
Experiment Design

This chapter presents the overall experiment design. Network architectures, the systems, benchmarking tools, and the synthetic programs are described in detail.

5.1 Experiment Systems

Louisiana Optical Network Initiative (LONI) [11] is the main networking infrastructure used in these experiments. LONI provides connections between Louisiana and Mississippi universities with a 10 Gbps optical network. LONI also includes a statewide 40 Gbps fiberoptic network linking four major systems (QueenBee, Eric, Poseidon, Painter). For the wide-area high-bandwidth tests, TeraGrid [23] is used. Grid Infn Laboratory for Dissemination Activities (GILDA) [13] is a high-speed network infrastructure on the Italian Istituto Nazionale di Fisica Nucleare (INFN) carried on in the context of the some Eurupian Grid enabled projects such as eela, EGEE, BioinfoGRID, etc. GILDA is used for overseas wide-area tests.

Also, two local systems in DSL Lab are used to demonstrate the performance in the local area. The details of each testbed system are given in the following subsections.

5.1.1 DSL Lab Testbed

Two of DSL Lab workstations are used to demonstrate performance with staging and remote I/O without high-speed network connectivity. These workstations are in the same local area network (LAN) with 100 Mbps network connection. The used workstations are:

- Dsl-stork: HP workstation with AMD 32-bit processor Fedora core 11 operating system and 1GB RAM.
- Dsl-tie: HP workstation with 2.8 GHz Core Duo Intel 32-bit processor Fedora core 7 operating system and 1GB RAM.

5.1.2 Loni Testbed

The following Loni systems are used in these experiments.

- Queen Bee: A 50.7 TFlops Peak Performance, 680 node, 2 Quad-Core processor Red Hat Enterprise Linux (RHEL) v4 cluster from Dell with 2.33 GHz Intel Xeon 64bit processors and 8 GB RAM per node. Housed at ISB. According to the June, 2007 Top500 listing, Queen Bee ranks the 23rd fastest supercomputer in the world.

- Eric: A 4.772 TFlops Peak Performance, 128 node, 2 Dual-Core processor Red Hat Enterprise Linux (RHEL) v4 cluster from Dell with 2.33 GHz Intel Xeon 64bit processors and 4 GB RAM per node. Housed at LSU.

- Poseidon: A 4.772 TFlops Peak Performance, 128 node, 2 Dual-Core processor Red Hat Enterprise Linux (RHEL) v4 cluster from Dell with 2.33 GHz Intel Xeon 64bit processors and 4 GB RAM per node. Housed at UNO.

- Painter: A 4.772 TFlops Peak Performance, 128 node, 2 Dual-Core processor Red Hat Enterprise Linux (RHEL) v4 cluster from Dell with 2.33 GHz Intel Xeon 64bit processors and 4 GB RAM per node. Housed at LaTech.

- Spider: 4 Dual-Core processor Red Hat Enterprise Linux (RHEL) cluster with 2.6 GHz AMD 2218 Model 64bit processors and 16 GB RAM. Housed at LSU.

- Oliver: A 4.772 TFlops Peak Performance, 128 node, 2 Dual-Core processor Red Hat Enterprise Linux (RHEL) v4 cluster from Dell with 2.33 GHz Intel Xeon 64bit processors and 4 GB RAM per node. Housed at ULL.

- Louie: A 4.772 TFlops Peak Performance, 128 node, 2 Dual-Core processor Red Hat Enterprise Linux (RHEL) v4 cluster from Dell with 2.33 GHz Intel Xeon 64bit processors and 4 GB RAM per node. Housed at Tulane.

5.1.3 TeraGrid Testbed

Two TeraGrid systems are used in these experiments:

- Lonestar: Dell PowerEdge Linux Cluster is configured with 5,840 compute-node cores, 11.6 TB of total memory and 106TB of local disk space. The peak performance rated is 62 TFLOPS. The

system supports a 70TB globally accessible, Lustre parallel file system. Nodes are interconnected with InfiniBand technology in a fat-tree topology with a 1GB/sec point-to-point bandwidth. Also, a 2.8 petabyte archive system and a 5TB SAN are available through the login/development nodes.

- Steele: 812 node Dell PowerEdge 1950 cluster running the Red Hat Enterprise Linux 4 operating system. Each node contains two Quad Core Intel Xeon 2.33GHz 64-bit processors and 16-32 GB of memory. The cluster is interconnected primarily with Gigabit Ethernet and has 180 TB of NFS storage provided by BlueArc NAS systems.

5.1.4 Gilda Testbed

One system is used for overseas tests:

- gGlite-tutor: 4 Dual-Core processor Red Hat Enterprise Linux (RHEL) cluster with 1.8 GHz AMD 265 Model 64bit processors and 4 GB RAM. Housed at Italy.

Spider is used as an execution node and the other systems are used as a data server. The only exception is the LAN tests. The LAN tests are done by DSL Lab systems.

5.2 Benchmarks

Some benchmarking tools are used to evaluate data access techniques on distributed environments. This section gives a brief description of the benchmarking tools.

5.2.1 IOzone

IOzone [18] is a file system benchmarking tool, which generates and measures a variety of file operations. The benchmark tests file I/O performance for the read and random read from local disk to memory, write and random write from memory to local disk, read and random read from remote disk to local memory, and write and random write from local memory to remote disk operations.

The following IOzone command is used to test staging disk performance:

iozone -r 16 -s 558m -p -i 0 -i 1 -i 2

- -r 16 used to specify the record size, in Kbytes, to test
- -s # used to specify the size, in MBytes, of the file to test

- -p purges the processor cache before each file operation
- -i 0 -i 1 -i2 Used to specify which tests to run. 0=write/rewrite, 1=read/re-read, 2=random-read/write

The following IOzone command is used to test remote disk performance:

iozone -r 16 -s 558m -p -f /chirp/poseidon1.loni.org/iozone.tmp -i 0 -i 1 -i 2

the only difference is -f which is used to specify the filename for the temporary file under test for remote file location.

5.2.2 GridFTP

GridFTP [27] uses Grid Security Infrastructure, which is known as gsiFTP. It provides authentication and encryption to file transfers with user specified levels of confidentiality and data integrity. Gridftp is used for staging in/out data in our experiments.

5.2.3 CCTools

The Cooperative Computing Tools (cctools) [17] are software collections which help to share resources and get along with each other in a complex, heterogeneous, unreliable computing environment for any type of users. Cctools provide reliable services without requiring kernel changes or special privileges for the user.

Some of the cctools utilities are used to implement remote I/O for the grid applications. Parrot/Chirp and Parrot/gsiftp are two combinations that are used to differentiate remote access protocols.

Parrot is a tool for attaching existing programs to remote I/O systems through the file system interface. It can be used to access remote storage devices with different protocols. Parrot can communicate with http, ftp, Gridftp, iRODS, srb, hdfs, rfio, dcap, and Chirp protocols. It traps the program system calls through the ptrace debugging interface, and replaces them with remote I/O operations. Parrot can be installed and operated by any user without any kernel-level changes.

Chirp is a personal file system and remote I/O protocol. Any user can run the program to start Chirp server on the system. It supports different type of security methods like Globus authentication, kerberos, etc. Since it is developed with cctools, it works preferably with Parrot and allows users to

have custom distributed file systems on a Grid environment. Chirp protocol provides easy of use, transparency, and easy of deployment of file system.

Cctools package was installed at all the data server systems we have used. Parrot bash shell is run to access remote data using chirp data server on the execution side. It mounts the remote chirp directory as a local directory on local system.

Parrot supports gsiftp protocol to access remote data as long as there is gridftp server running and there is valid globus certificates. In these experiments, parrot/gsiftp combination is used for remote I/O implementation and gridftp is used for staging implementation.

5.3 Experiment Setup

We have created synthetic programs to simulate the effects of different parameters such as input and output data size, data access pattern, and data access ratio on the performance of staging vs remote I/O. These programs can simulate three different data access patterns:

- read and write the blocks sequentially (seq)
- read and write every other block in the dataset sequentially (jump)
- read and write all of the dataset in random blocks (random)

with four different data access ratios which are:

- read all blocks, process it, and write all blocks to a new file (full)
- read half of the blocks, process it, and write the processed half to a new file (half)
- read quarter of the blocks, process it, and write the processed quarter blocks to a new file (quarter)
- read eighth of the blocks, process it, and write the processed eighth blocks to a new file (eighth)

Also, all the tests are run for two different data access techniques:

- Staging

- Remote I/O

Each experiment is run in ten different network topologies which are listed as increasing size in distance:

- Local area network (LAN) (Dsl-tie & Dsl-stork)
- Campus area networks (CAN) (Spider & Eric)
- Metropolitan Area network (MAN) (Spider & Queen Bee)
- Wide area networks (WAN1) (Spider & Oliver)
- Wide area networks (WAN2) (Spider & Louie)
- Wide area networks (WAN3) (Spider & Poseidon)
- Wide area networks (WAN4) (Spider & Painter)
- Wide area networks (WAN5) (Spider & Lonestar)
- Wide area networks (WAN6) (Spider & Steele)
- Wide area networks (WAN7) (Spider & Gilda)

First, the data is staged in from the remote data server to local directory with globus-url-copy. After the execution, the data is staged out back to data server with globus-url-copy. To measure how long it takes to transfer data from local hard disk to memory, IOzone is used to transfer the same amount of data from local hard disk to local memory. Also, IOzone is used to measure how long it will take to transfer data from the remote hard disk to remote memory. The real data and the data that IOzone creates and uses for tests have similar characteristics like amount and record size.

For remote I/O, two different access protocols are used with the parrot, which are chirp and gsiftp. Mainly gsiftp is used to create consistency with the staging tests. Chirp is used to show how the performance changes with a different protocol. The data is stored on data server side where the GridFTP server is running. The execution node runs the simulation code to access data using parrot/gsiftp combination.

Chirp server is run on a data server to mount the remote data directory to local directory structure with parrot on parrot/chirp tests, so, the simulation program can access the remote data as if accessing the local data with some system.

For standardization and to minimization of the network effects, all the tests are repeated five times at a different time and the average is taken.

A Perl script which is written to run all simulations and measure the time in seconds is given in Appendix A.

5.3.1 Parameters

The parameters we measure are:

- **Staging**
  - **Rrin:** Time to read from remote hard drive to remote memory.
  - **Nsin:** Time to transfer from remote host to local host over network.
  - **Wlin:** Time to write from local memory to local hard drive.
  - **Rlin:** Time to read from local hard drive to local memory for execution.
  - **Execution**: Time to execute the program.
  - **Wlout:** Time to write the generated data from local memory to local hard drive after execution.
  - **Rlout:** Time to read the generated data from local hard drive to local memory for transfer.
  - **Nsout:** Time to transfer the data from local host to remote host over network.
  - **Wrout:** Time to write the generated data from remote memory to remote hard drive after transfer.
  - **Ps:** Total time passed for all staging processes.

  Remote I/O

  - **Rrin:** Time to read from the remote hard drive to remote memory.

- **Execution:** Time to execute the program.
- **Wrout:** Time to write the generated data from remote memory to remote hard drive after transfer.
- **Nr:** Time to transfer the data between local host and remote host over network.
- **Pr:** Total time passed for all remote I/O processes.

5.4 Synthetic Programs

We have developed four different applications for this project. The first one is data generation application. This application generates the preprocessed data randomly. We use this data for our synthetic applications. The second one, reads and writes the blocks sequentially (seq). The third one reads and writes every other block in the dataset sequentially (jump). The last one, reads and writes all the dataset in random blocks (random). Last three applications have four different versions based on the data ratio used which are, full, half, quarter, and eighth.

The application reads and writes all the data in full-mode. We have made small changes to the program in order to read and write half of the data in half-mode, a quarter of the data in quarter-mode, and an eighth of the data in eighth-mode. Also, we have made small changes to source code to access data remotely in remote mode for all data ratios.

5.4.1 Data Generation Program

We have developed a data generation application to generate data. We need to produce data with defined transaction size and data size. Then, the program uses these inputs to generate our data file. The items in each transaction are sorted and duplicates are eliminated.

5.4.2 Sequential

We have developed a synthetic application that reads all the data sequentially, then manipulates the data, and writes the results as a separate file sequentially. We outline the key features of the implementation details in the following lines. We also describe how the program works:

1. Read all the data from file into memory sequentially. The data file named "transacs.dat" has the item codes sorted within each transaction.

2. Count the frequency for each item.

3. Build a tree from the data on memory.

4. Write all the data from memory to a separate file.

5.4.3 Jump

Our second developed synthetic application (jump) reads and writes every other block in the dataset sequentially. We outline the key features of the implementation details in the following lines.

1. Read every other blocks from file into memory. The data file named "transacs.dat" has the item codes sorted within each transaction.

2. Count the frequency for each item.

3. Build a tree from the data on memory.

4. Write all the data from memory to a separate file.

5.4.4 Random

Our third developed synthetic program reads and writes data in random blocks in the dataset which we called random, processes the data, and writes the results as a separate file randomly.

1. Read all the dataset in random blocks from file into memory. The data file "transacs.dat" has the item codes sorted within each transaction.

2. Count the frequency for each item.

3. Build a tree from the data on memory.

4. Write all the data from memory to a separate file.

Chapter 6
Results and Analysis

We have used parrot/gsiftp results to develop our models. Therefore, parrot/gsiftp results will be evaluated in details. Experiment results can be divided into two main categories based on which remote I/O protocol is used. Since GridFTP/gsiftp combination is used for staging data transfer protocol, parrot/gsiftp combination is chosen as a main remote I/O implementation combination. Parrot is used to mount the data on the GridFTP server with gsiftp protocol. Parrot/chirp results are used to compare the performance with the parrot/gsiftp to show how important it is to choose the right remote I/O protocol.

6.1 Parrot/gsiftp Results

Quarter data size is chosen to represent four data sizes' characteristics on both remote data access techniques. The size of the data and the performance are inversely proportional, due to the fact that four data sizes have the similar performance characteristics. Because of the gap between the minimum and the maximum values, logarithmic scale is used for y scale on figures.

Spider is the execution node on all tests. The data server node is changed based on the network architecture is used. The node list can be seen in Chapter 5. There is an exception on LAN tests that DSL-Lab systems are used for both execution and server node.

Data move in from the server node to the execution node and after execution the results are moved out from server node to the execution node on data staging. Running simulation on execution node, accesses the remote data on the running node by using the parrot/gsiftp protocol using the remote I/O.

Staging figures show the stage in (Rrin, Nsin, Wlin) values, the execution (Rlin, Run, Wlout), the stage out (Rlout, Nsout, Wrout) values, and the total (Ps). Also, Remote I/O figures show the remote read performance (Rin), the execution (Run), remote write performance (Wrout), remote network performance (Nr), and the total (Pr) value.

Sequential data access usually has the best performance in total, and jump data access follows it. Usually random has the worst performance comparing the other data access techniques.

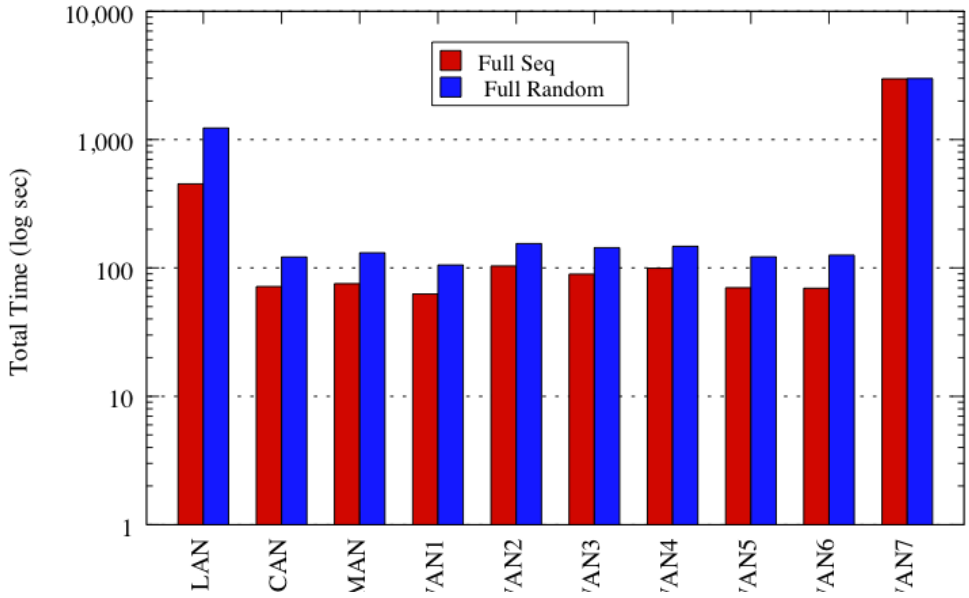

FIGURE 6.1: Full Seq vs Full Random for All Network Architectures with Staging

Figure 6.1 indicates that full sequential always performs better then full random on all network architectures. It also shows that high-speed networks improves the performance dramatically. Figure 6.2 explains that why random remote I/O performs worse. Random on remote I/O is not benefitting from high-speed network architectures.

6.1.1 Local Area Network (LAN)

Dsl-Lab workstations are used for Local Area Network (LAN) tests. Dsl-tie is the execution node and dsl-stork is the data server node for both LAN tests.

Table 6.1 shows an example staging average results of LAN tests. Table 6.2 also shown remote I/O results in the LAN architecture. Tables indicates that remote I/O is better on sequential and staging but worse on random.

Figure 6.3 shows the total numbers for both data access methods with all type of data sizes and data access techniques. Remote I/O performs better then staging in all of the sequential and jump data access patterns. Staging performs better than remote I/O in random data access technique with all data sizes. The differences on sequential and jump is less than the differences on random. So,

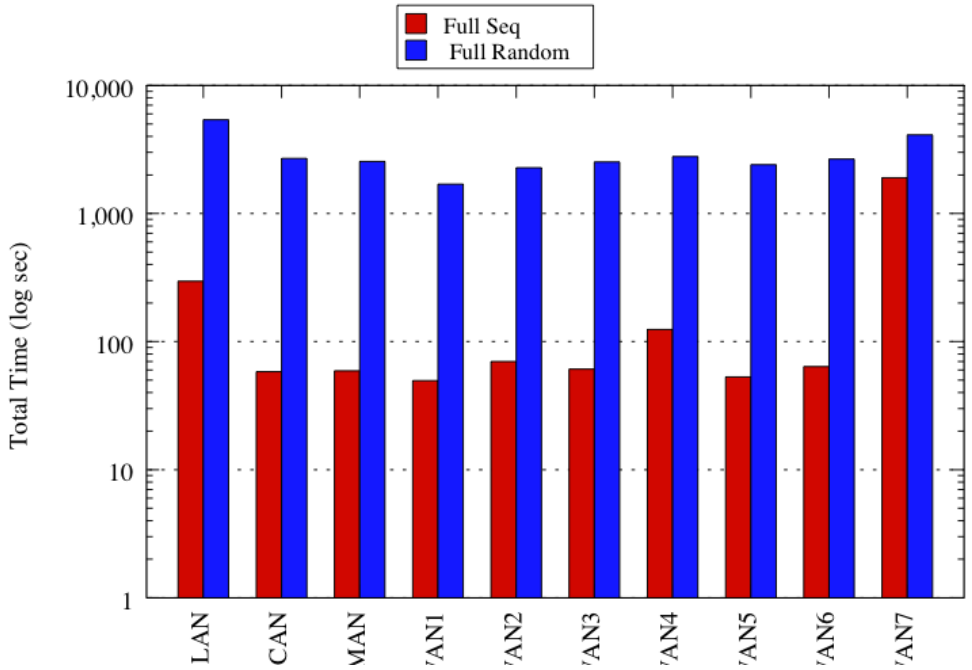

FIGURE 6.2: Full Seq vs Full Random for All Network Architectures with Remote I/O

TABLE 6.1: Staging Results in LAN Network Architecture

Method	Rrin	Nsin	Wlin	Gridftp-in	Rlin	Run	Wlout	Rlout	Nsout	Wrout	Gridftp-out	Total
Full Seq	5.41	127.95	12.63	145.99	16.66	111.95	12.63	16.66	143.95	4.22	164.84	452.06
Full Jump	5.41	129.96	12.63	148.00	16.66	122.83	12.63	16.66	142.22	4.22	163.10	463.22
Full Random	5.41	129.22	12.63	147.26	16.66	894.20	12.63	16.66	139.53	4.22	160.41	1231.16
Half Seq	5.41	129.77	12.63	147.81	8.33	44.21	6.31	8.33	77.55	2.11	87.99	294.65
Half Jump	5.41	128.27	12.63	146.31	8.33	71.75	6.31	8.33	76.79	2.11	87.23	319.93
Half Random	5.41	128.21	12.63	146.25	8.33	164.73	6.31	8.33	73.48	2.11	83.92	409.55
Quarter Seq	5.41	130.38	12.63	148.42	4.16	24.87	3.16	4.16	42.81	1.06	48.03	228.64
Quarter Jump	5.41	129.74	12.63	147.78	4.16	40.98	3.16	4.16	42.52	1.06	47.74	243.83
Quarter Random	5.41	129.59	12.63	147.63	4.16	108.95	3.16	4.16	41.27	1.06	46.49	310.39
Eighth Seq	5.41	131.79	12.63	149.83	2.08	19.13	1.58	2.08	26.09	0.53	28.70	201.33
Eighth Jump	5.41	130.11	12.63	148.15	2.08	34.12	1.58	2.08	26.65	0.53	29.26	215.19
Eighth Random	5.41	127.48	12.63	145.52	2.08	65.58	1.58	2.08	24.36	0.53	26.97	241.73

TABLE 6.2: Remote I/O Results in LAN Network Architecture

Method	Rrin	Execution	Wrout	Nr	Total
Full Seq	12.31	272.13	11.38	160.19	295.82
Full Jump	12.31	281.80	11.38	158.98	305.49
Full Random	12.31	5326.63	11.38	4468.33	5386.22
Half Seq	5.70	139.23	4.79	95.03	149.72
Half Jump	5.70	157.27	4.79	85.52	167.76
Half Random	5.70	2418.19	4.79	2295.93	2471.15
Quarter Seq	1.23	75.44	2.36	50.58	79.04
Quarter Jump	1.23	94.06	2.36	53.07	97.65
Quarter Random	1.23	1238.16	2.36	1131.87	1244.41
Eighth Seq	0.85	45.92	1.33	26.78	48.09
Eighth Jump	0.85	64.70	1.33	30.58	66.88
Eighth Random	0.85	592.00	1.33	525.67	593.43

researcher should be more careful when remote I/O was chosen. SInce all data should be staged in before the execution, decreasing the data ratio improves the remote I/O performance.

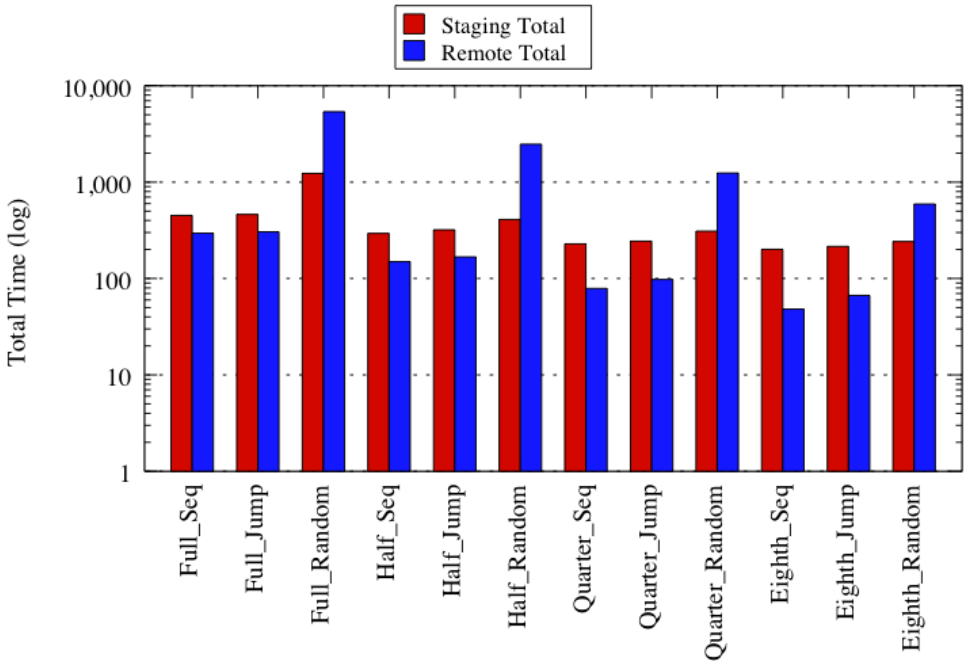

FIGURE 6.3: LAN Total results for Staging and Remote I/O

Figure 6.4 (a) shows performance with quarter data size and all types of data access techniques.Stage in and stage out take the most of the time, because LAN doesn't have high speed network architecture. Execution also takes a lot of time, because DSL_Lab workstations are not super-computers.

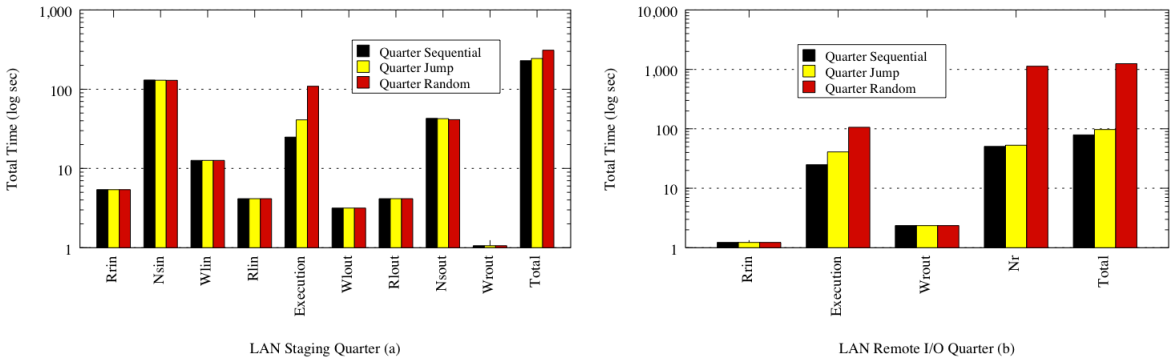

FIGURE 6.4: LAN Quarter Data Size on Staging and Remote I/O Figures

Figure 6.4 (b) shows the performance with quarter data size and all type of data access techniques on remote data access. Nr is taking most of the time, because of slow network.

6.1.2 Campus Area Network (CAN)

Loni systems are used for Campus Area Network (CAN) tests. Both systems are located in the LSU campus.

The Figure 6.5 shows the total numbers for both data access methods with all types of data size and data access techniques. Remote I/O performs better then staging in sequential and jump data access patterns. Staging performs better than remote I/O on random data access technique with all data sizes.

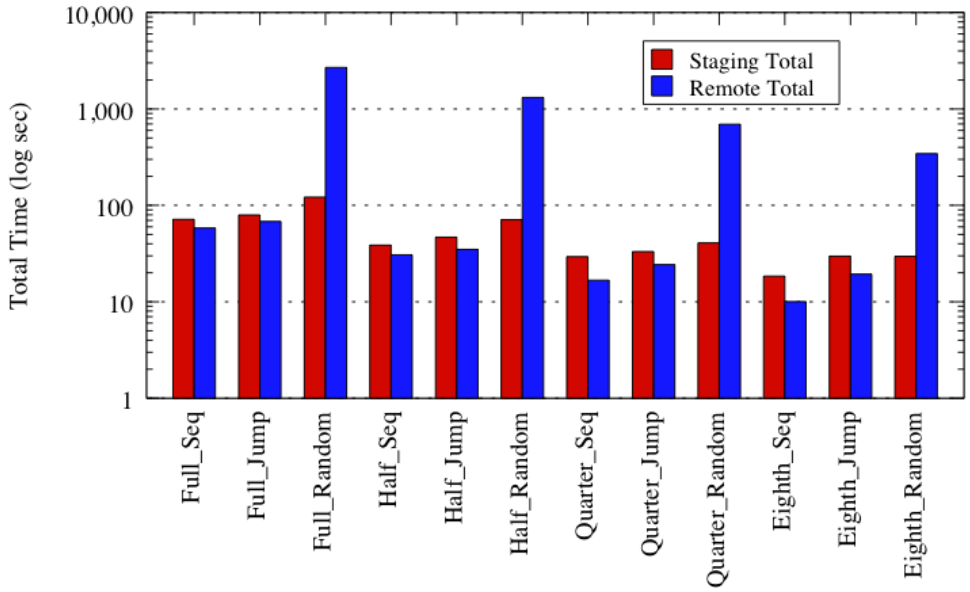

FIGURE 6.5: CAN Total for Staging and Remote I/O

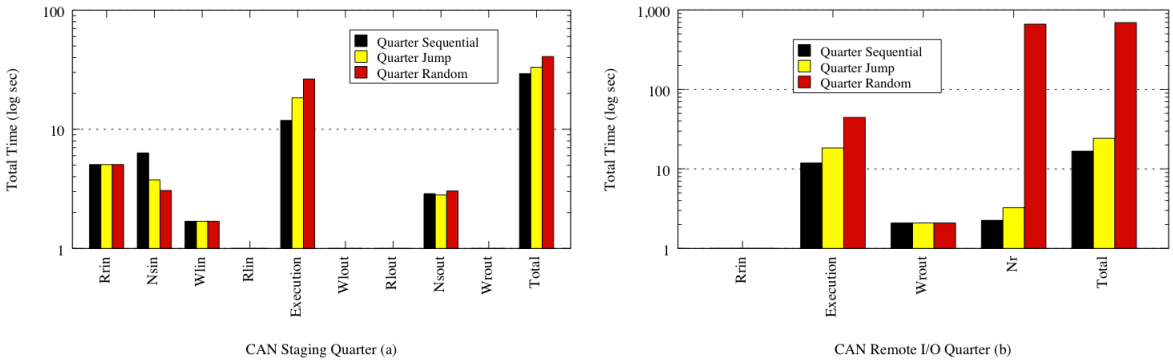

FIGURE 6.6: CAN Quarter Data Size on Staging and Remote I/O Figures

According to the Figure 6.6 (a) full random run performance is worse than the other data access techniques. The differences between remote and the staging getting worser on random data access. Each time random data access needs to access remote data, network overhead decreases the performance.

Figure 6.6 (b) shows the performance with quarter data size and all type of data access techniques on remote data access. It also shown that how network affects the performance in remote I/O. Network overhead plays important role on random remote I/O. If the application can use advanced programming techniques to handle network overhead, remote random performance can be improved.

6.1.3 Metropolitan Area Network (MAN)

Loni systems are used for Metropolitan Area Network (MAN) tests. Both systems are located in Baton Rouge.

Figure 6.7 shows that remote I/O performs better than staging in sequential and jump data access patterns. Staging performs better than remote I/O on random data access technique with all data sizes.

According to Figure 6.8 (a) has similar performance characteristics with CAN. Nsin jump has the best performance that is unusual because of network.

Figure 6.8 (b) shows performance figure with quarter data size and all type of data access techniques on remote data access.

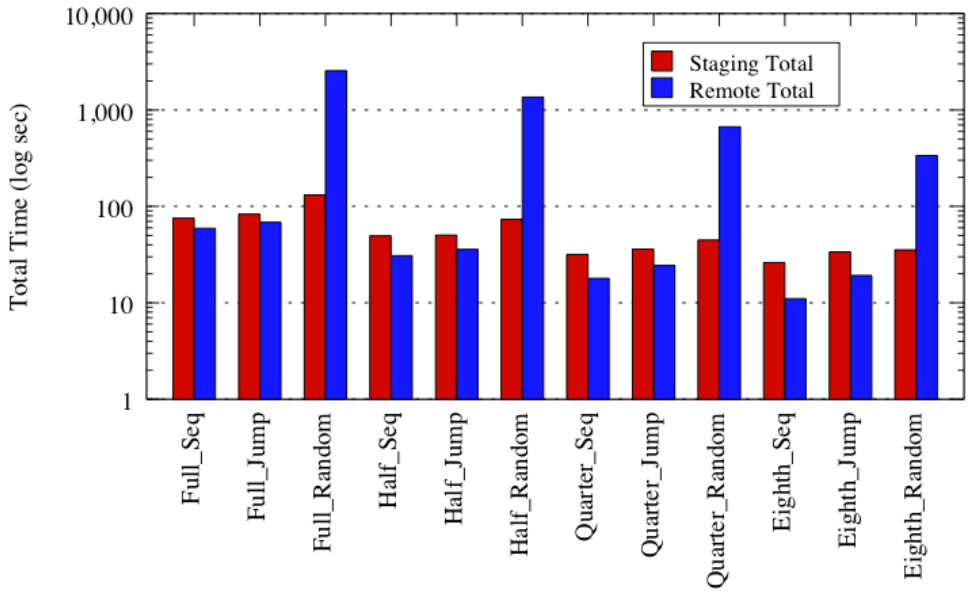

FIGURE 6.7: Man Total for Staging and Remote I/O

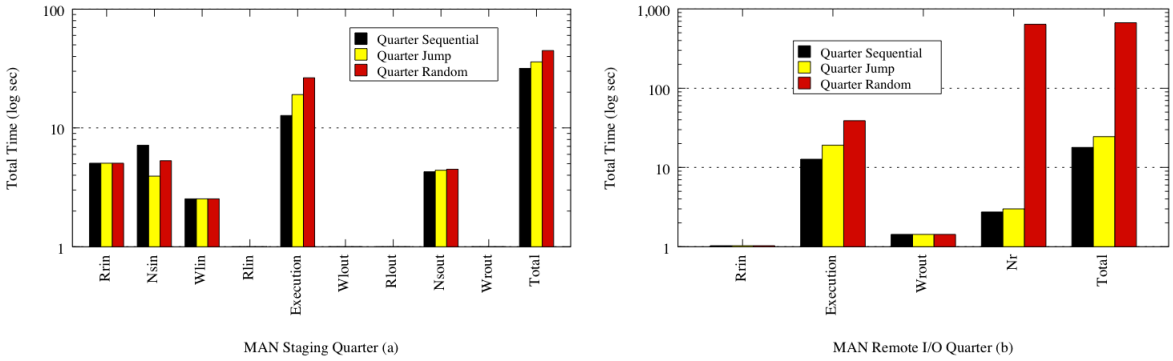

FIGURE 6.8: MAN Quarter Data Size on Staging and Remote I/O Figures

6.1.4 Wide Area Network 1 (WAN1)

Loni systems are used for Wide Area Network 1 (WAN1) tests. Execution node is located in Baton Rouge and data storage node is located in Lafayette.

Figure 6.9 shows total numbers for both data access method. Remote I/O performs better than staging in sequential and jump data access patterns. Staging performs better than remote I/O in random data access technique with all data sizes.

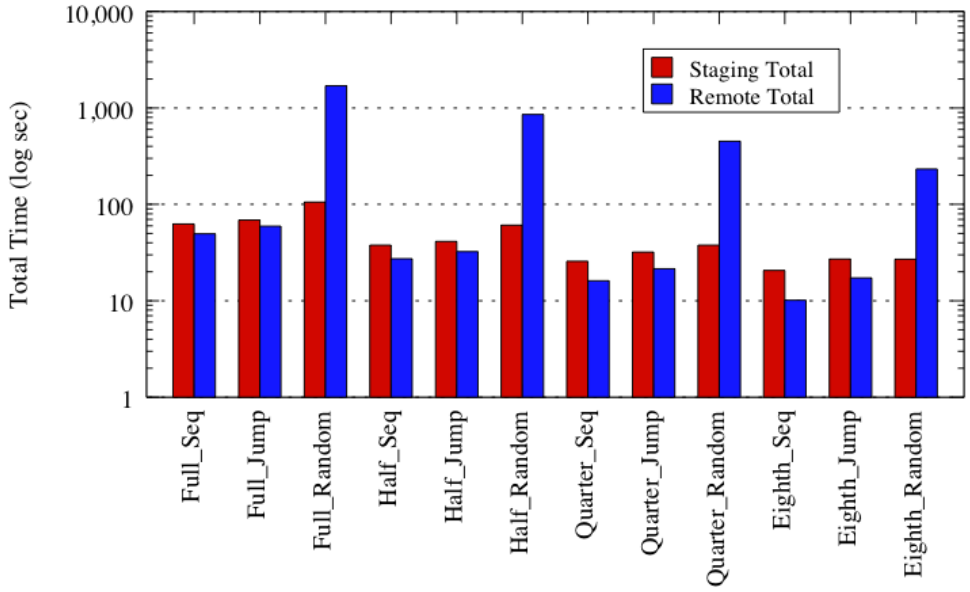

FIGURE 6.9: WAN1 Total for Staging and Remote I/O

According to Figure 6.10 (a) Nsin unusually has the worst performance because of network performance.

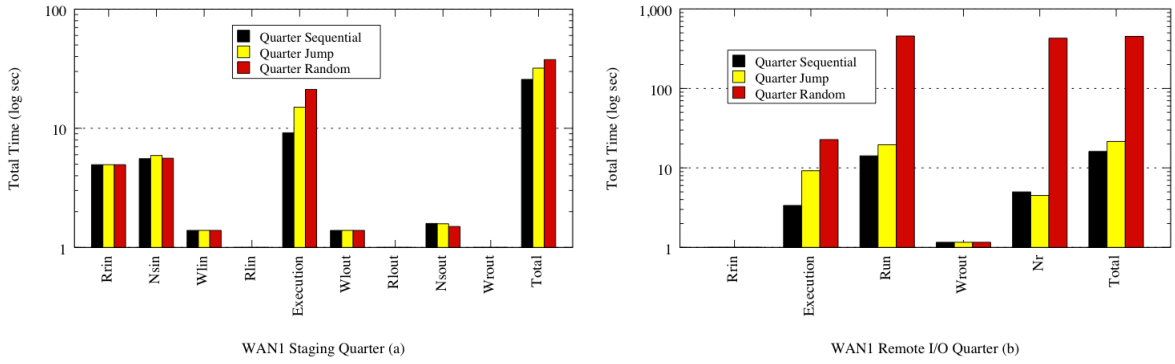

FIGURE 6.10: WAN1 Quarter Data Size on Staging and Remote I/O Figures

Figure 6.10 (b) shows the performance with quarter data size and all type of data access techniques on remote data access. It also shown that how network affects the performance in remote I/O.

6.1.5 Wide Area Network 2 (WAN2)

Loni systems are used for Wide Area Network 2 (WAN2) tests. Execution node is located in Baton Rouge and data storage node is located in New Orleans (Tulane).

Figure 6.11 shows remote I/O performs better than staging all sequential and jump data access patterns. Staging performs better than remote I/O in random data access technique with all data sizes.

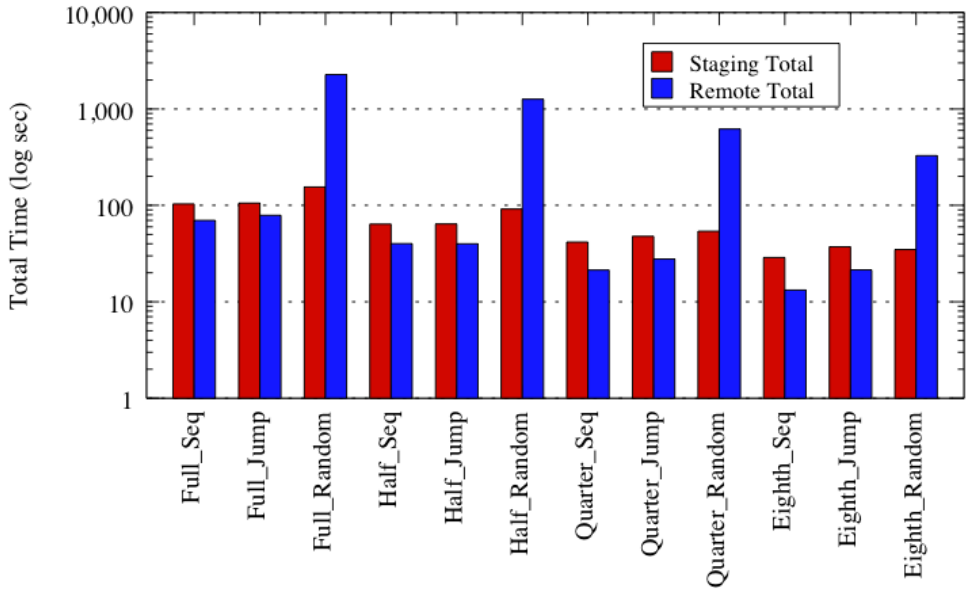

FIGURE 6.11: WAN2 Total for Staging and Remote I/O

According to Figure 6.12 (a) Nsin jump unusually has the best performance because of network. Other three data sizes have almost the same performance characteristics. Full random run performance is worse than the other data access techniques.

Figure 6.12 (b) shows the performance with quarter data size and all type of data access techniques on remote data access.

6.1.6 Wide Area Network 3 (WAN3)

Loni systems are used for Wide Area Network 3 (WAN3) tests. Execution node is located in Baton Rouge and data storage node is located in New Orleans (UNO).

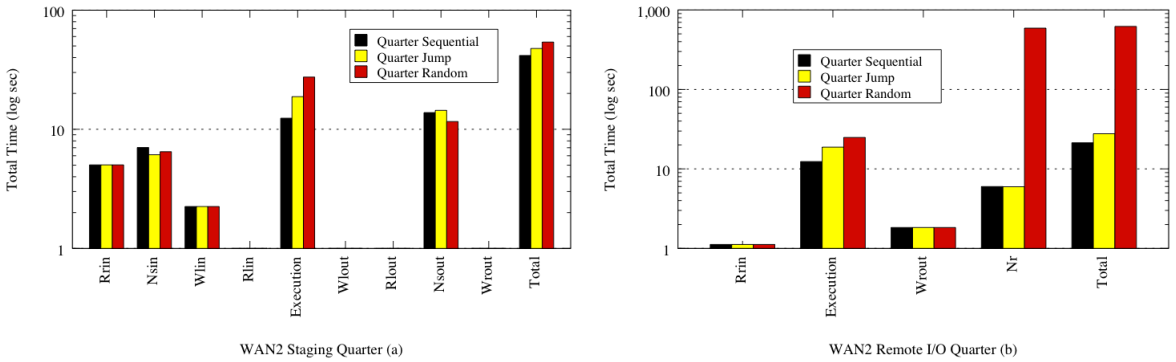

FIGURE 6.12: WAN2 Quarter Data Size on Staging and Remote I/O Figures

The figure 6.13 shows total for both data access methods with all type of data size and data access techniques. Remote I/O performs better then staging all sequential and jump data access patterns. Staging performs better than remote I/O on random data access technique with all data sizes.

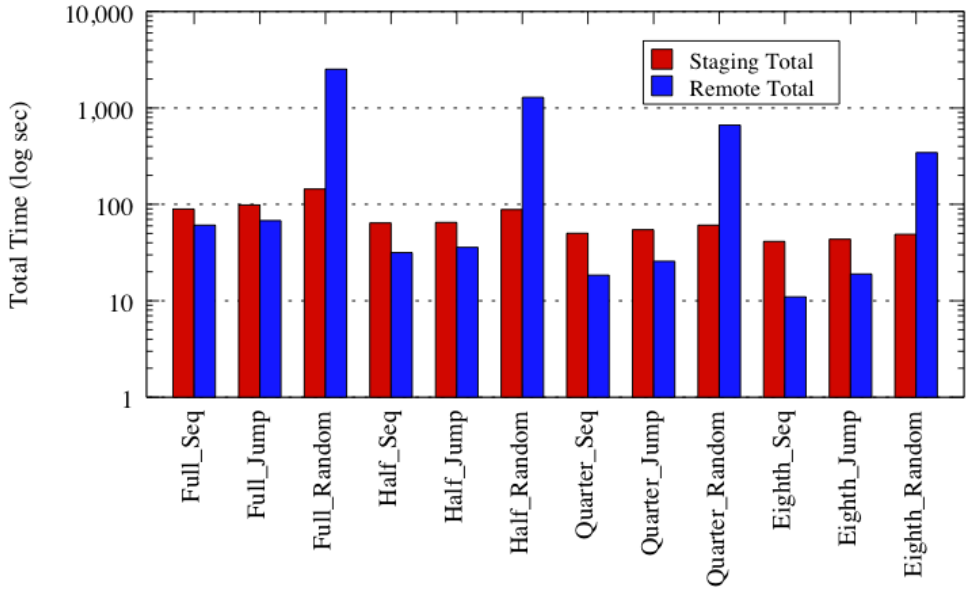

FIGURE 6.13: WAN3 Total for Staging and Remote I/O

Figure 6.14 (a) shows performance figure with quarter data size and all type of data access techniques. Sequential data access has the best performance in total, then jump data access has the second performance. Finally, random data access has the worst performance. These differences come from Run performance. Because of network, Nsin jump has the best performance.

Figure 6.14 (b) shows performance figure with quarter data size and all type of data access techniques on remote data access.

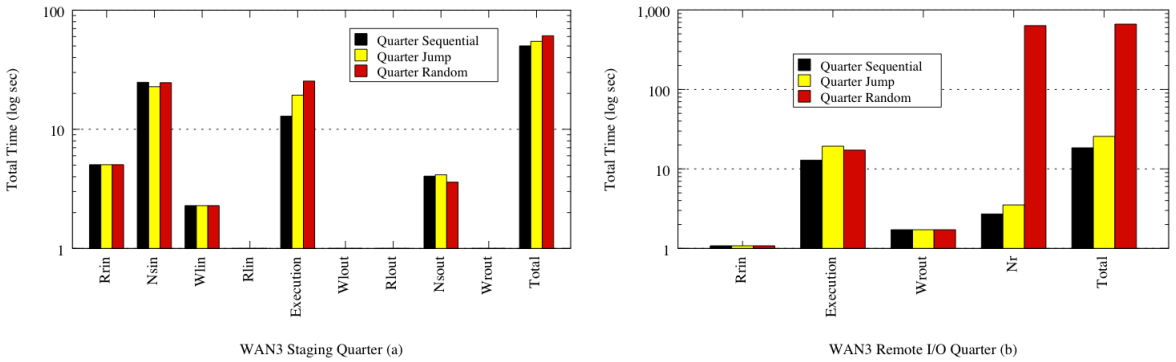

FIGURE 6.14: WAN3 Quarter Data Size on Staging and Remote I/O Figures

6.1.7 Wide Area Network 4 (WAN4)

Loni systems are used for Wide Area Network 4 (WAN4) tests. Execution node is located in Baton Rouge and data storage node is located in Ruston.

Figure 6.15 shows that remote I/O performs better than staging all sequential and jump data access patterns. Staging performs better than remote I/O in random data access technique with all data sizes. The differences on sequential and jump are decrease because of the distance involved.

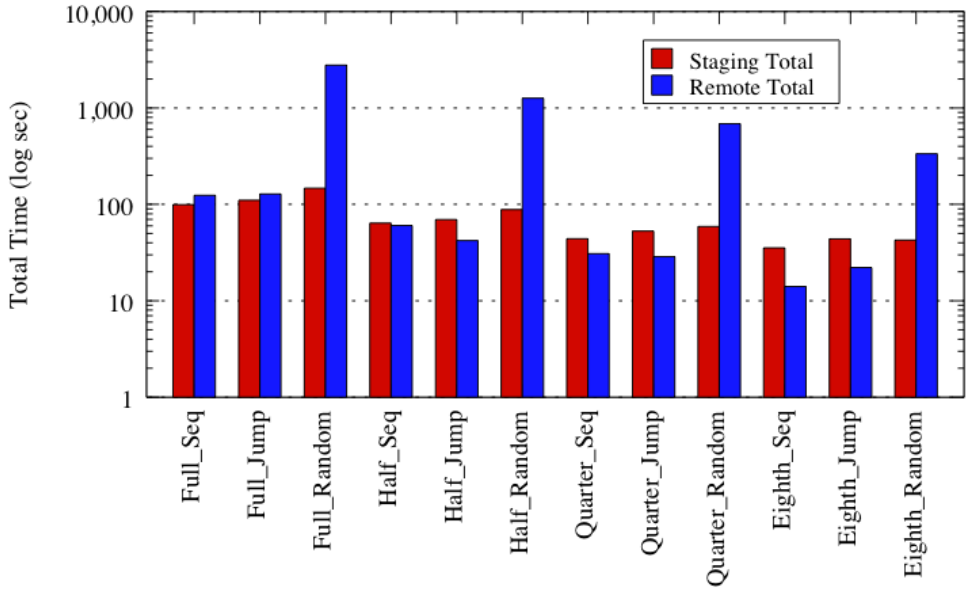

FIGURE 6.15: WAN4 Total for Staging and Remote I/O

Figure 6.16 (a) shows performance figure with quarter data size and all type of data access techniques. Because of network performances, Nsout jump has the worst performance.

Figure 6.16 (b) shows performance figure with quarter data size and all type of data access techniques on remote data access.

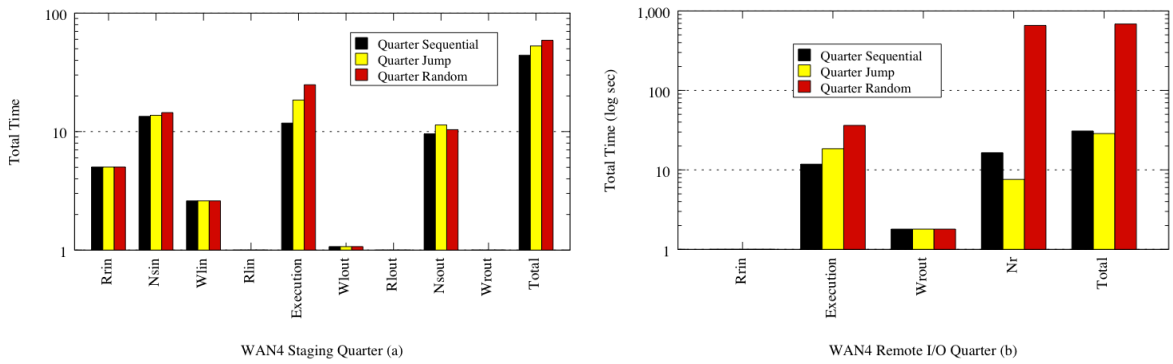

FIGURE 6.16: WAN4 Quarter Data Size on Staging and Remote I/O Figures

6.1.8 Wide Area Network 5 (WAN5)

Loni systems are used for Wide Area Network 5 (WAN5) tests. Execution node is located in Baton Rouge and data storage node is located in Austin, TX.

Figure 6.17 indicates total numbers for both data access methods with all types of data size and data access techniques.

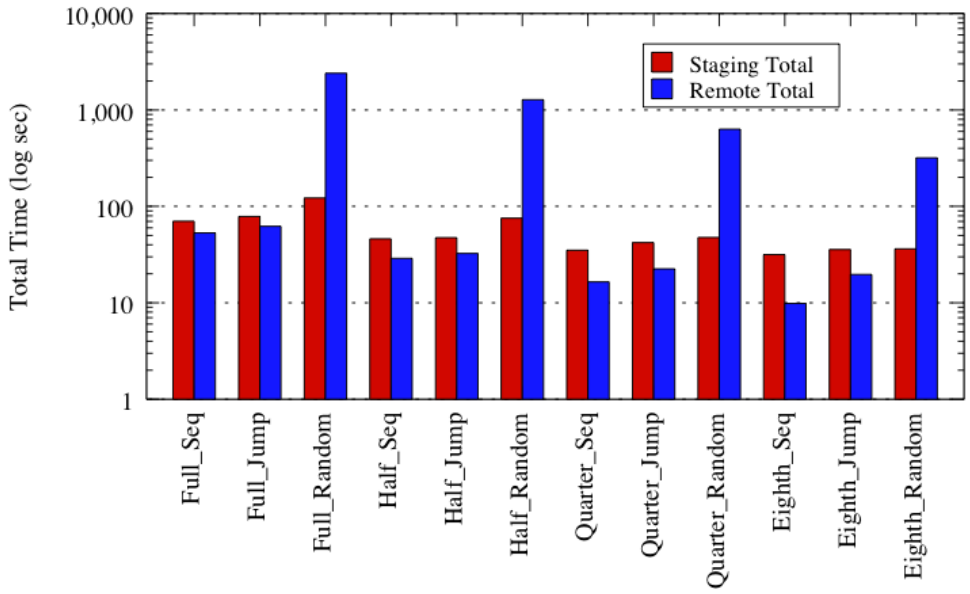

FIGURE 6.17: WAN5 Total for Staging and Remote I/O

Figure 6.18 (a) shows that because of network performances, Nsout jump has the worst performance.

According to Figure 6.18 (b) shows performance figure with quarter data size and all type of data access techniques on remote data access.

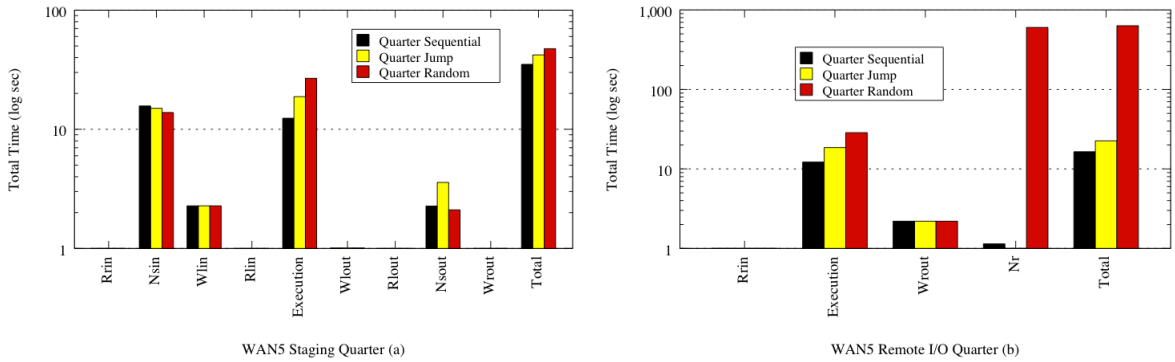

FIGURE 6.18: WAN5 Quarter Data Size on Staging and Remote I/O Figures

6.1.9 Wide Area Network 6 (WAN6)

Loni systems are used for Wide Area Network 6 (WAN6) tests. Execution node is located in Baton Rouge and data storage node is located in West Lafayette, IN.

Figure 6.19 shows that remote I/O performs better than staging in sequential and jump data access patterns. Staging performs better than remote I/O in random data access technique with all data sizes. We have same pattern here, increasing the distance decrease the the gab between sequential and jump performance.

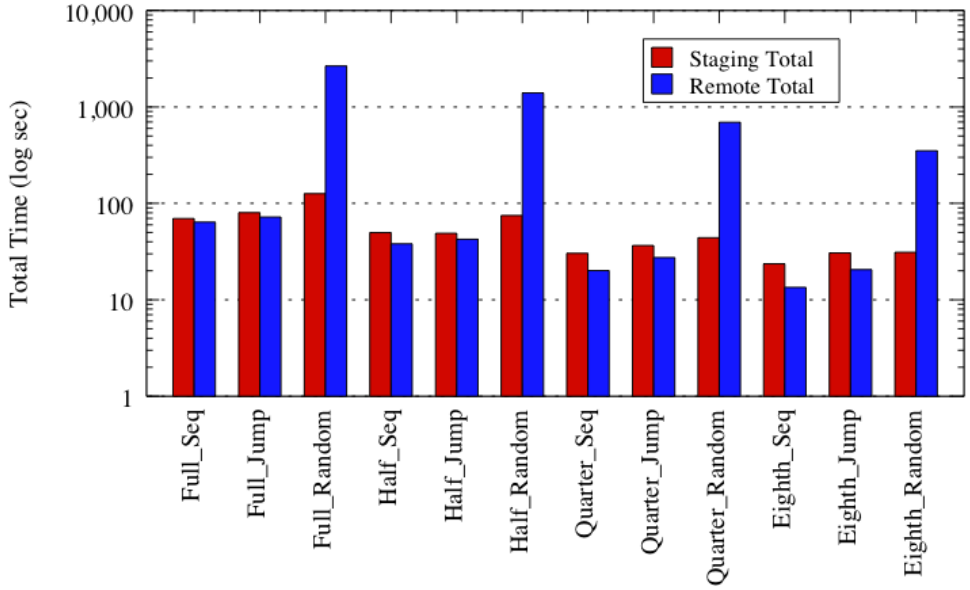

FIGURE 6.19: WAN6 Total for Staging and Remote I/O

Figure 6.20 (b) shows performance figure with quarter data size and all type of data access techniques on remote data access.

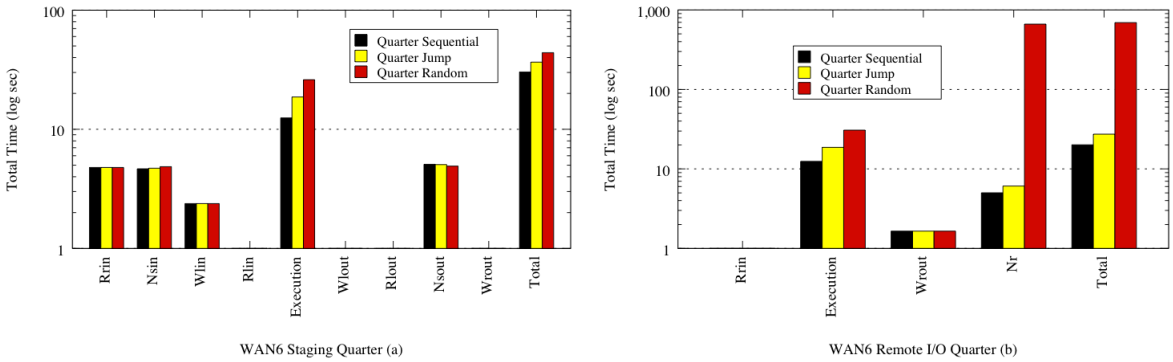

FIGURE 6.20: WAN6 Quarter Data Size on Staging and Remote I/O Figures

6.1.10 Wide Area Network 7 (WAN7)

Loni systems are used for Wide Area Network 7 (WAN7) tests. Execution node is located in Baton Rouge and data storage node is located in Italy.

According to Figure 6.21 WAN7 has different characteristics because of the distance between the execution node and the data server node. Staging is better in full random and half random tests, but is worse on quarter random and eighth random tests. Data ratio also plays a big role in this architecture. Less data ratio performs better in random. Without high-speed network, it decrease the gap between remote I/O and staging on all data access techniques. On the other hand, it increases the gap on smaller data ratios.

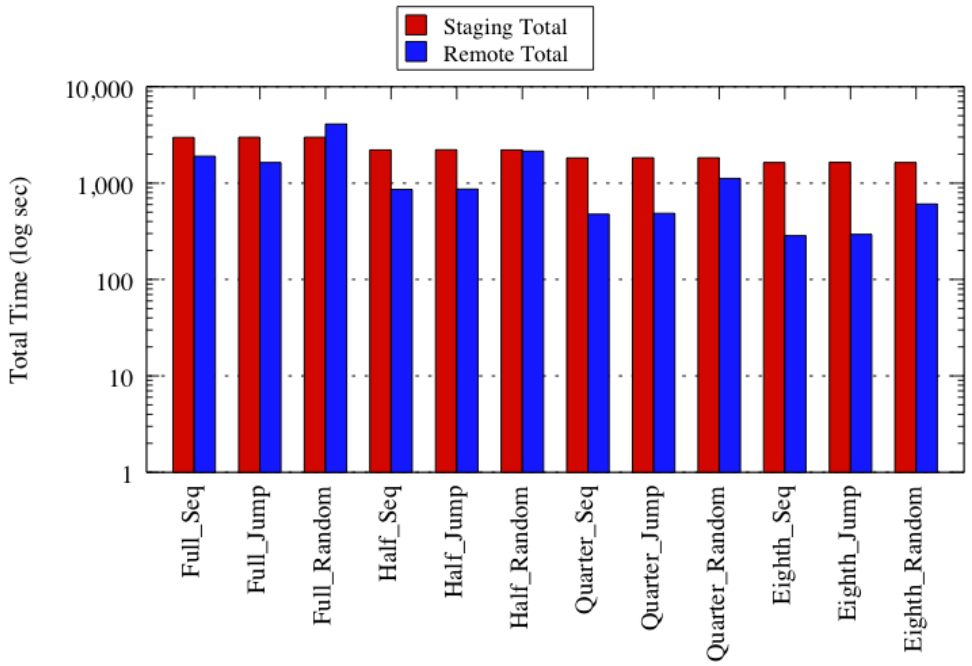

FIGURE 6.21: WAN7 Total for Staging and Remote I/O

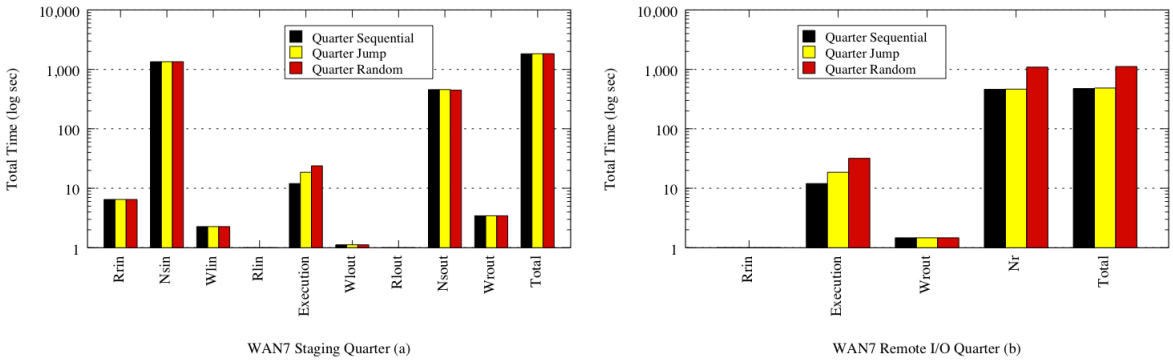

FIGURE 6.22: WAN7 Quarter Data Size on Staging and Remote I/O Figures

Figure 6.22 (a) shows the differences come from Run performance.

6.1.11 Comparison Between Network Architectures on the parrot/gsiftp Combination

All network architectures are discussed in this section. We would like to show the bigger picture on staging and remote I/O. Because of the gap between the minimum and the maximum values, logarithmic scale is used for y scale on the figures. The network architectures are listed from shortest distance to longest distance like LAN, CAN, MAN, WAN1 .. WAN7 on all figures.

Figure 6.23 shows the big picture of parrot/gsiftp experiments. Figure 6.23 (a), (c), and (e) indicates staging with sequential, full, and jump access techniques. The rest shows remote I/O performances with all data access techniques. WAN7, which is the longest in distance, has the worst performance among the all network architectures. On the other hand, LAN has shortest in distance, but the second worst performance because of network speed. LAN has the slowest network architecture among the rest. Eighth has the best performance among other techniques, then quarter, half follows it. Full has the worst performance among the rest. LAN has worst performance on random data access. It is worser than WAN7.

Figure 6.23 also indicates that high-speed network performances are close to each other and they have better performance than LAN and WAN7. In remote I/O random performance, all results are close to each other because of the distance.

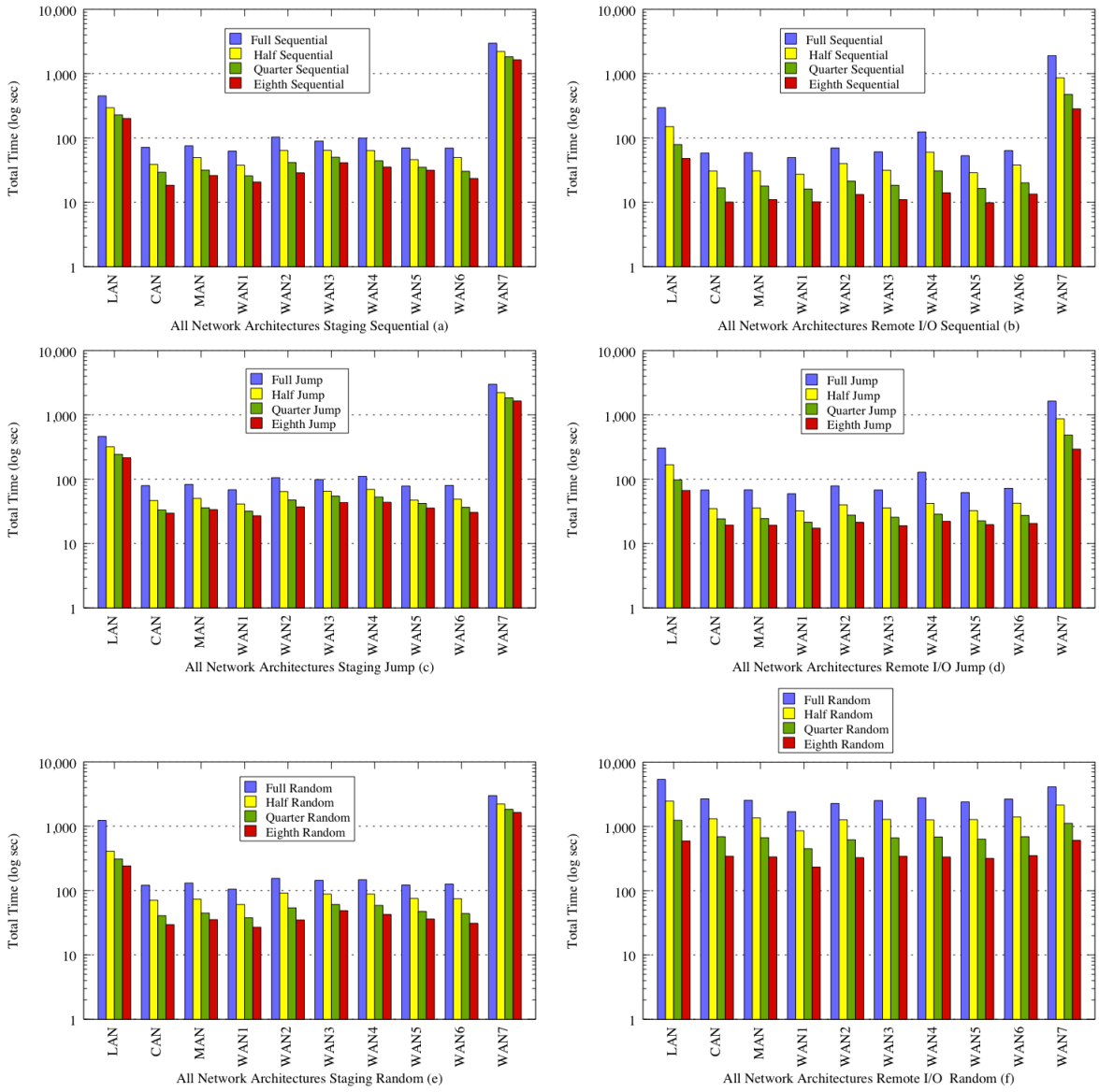

FIGURE 6.23: All Network Architectures Data Access Techniques Performance on Staging and Remote I/O

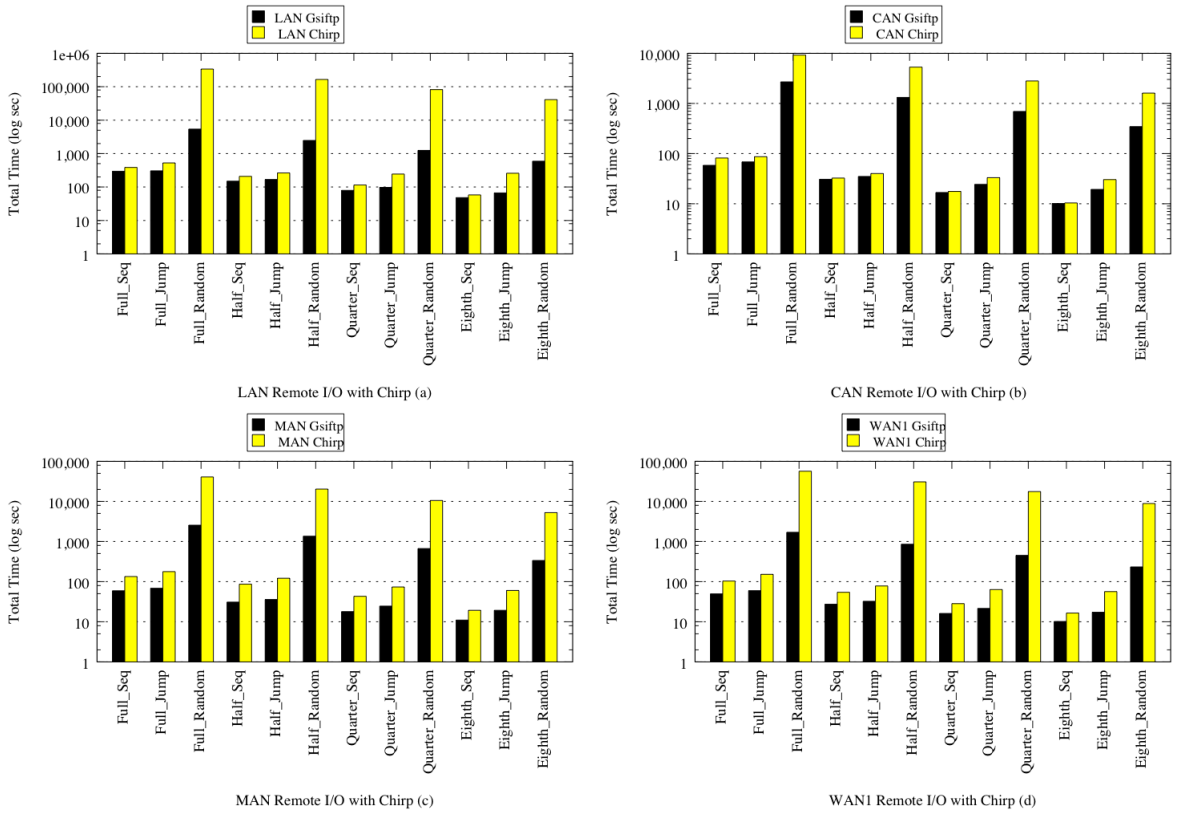

FIGURE 6.24: Remote I/O Performances with Chirp Protocol

6.2 Comparison Between parrot/gsiftp and parrot/chirp Combination

Choosing proper remote I/O protocol is crucial decision for the application performance. To show the performance differences, we setup the same experiment design with the parrot/chirp remote I/O protocol. We present some sample figures to give the big picture.

6.2.1 Remote I/O Results

Figure 6.24 shows the the comparison results between parrot/gsiftp and parrot/chirp performance on remote I/O. It can be seen that gsiftp performs better than chirp in all the network architecture, data sizes, and data access patterns.

6.2.2 Staging Results

Figure 6.25 shows the the comparison results between parrot/gsiftp and parrot/chirp performance on staging. It can be seen that staging performs almost equal on both remote I/O techniques. Small differences comes from the network performance.

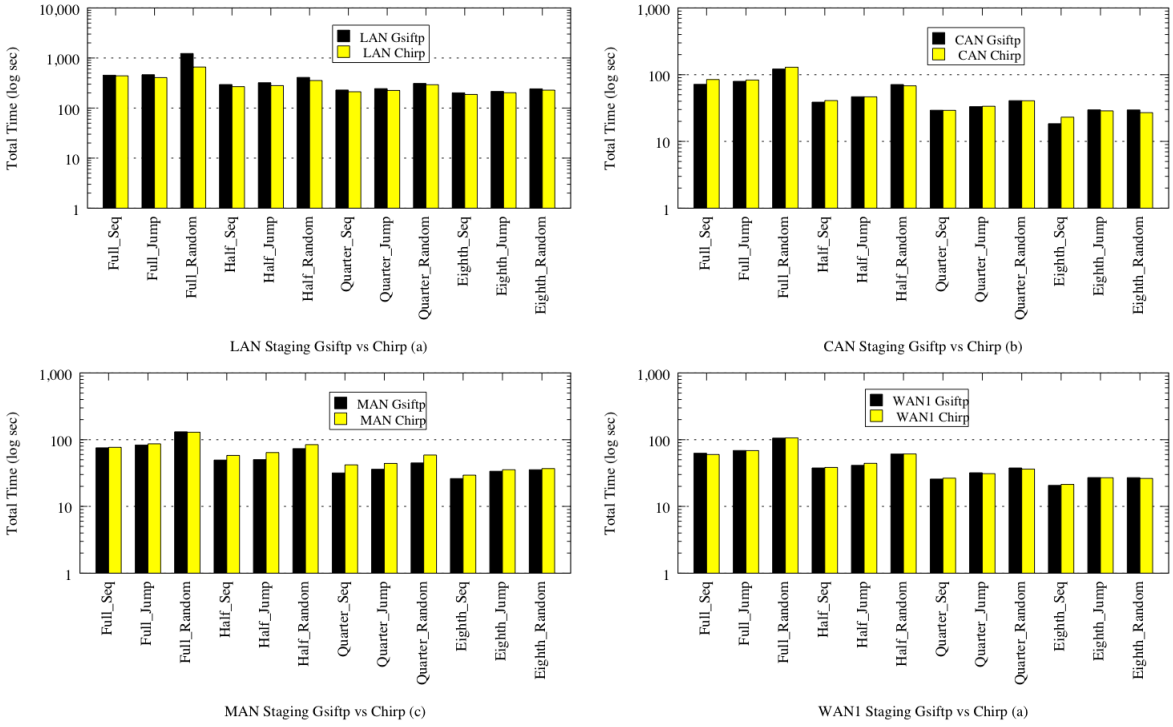

FIGURE 6.25: Staging Performances with Chirp Protocol

Chapter 7
Cache Impact

7.1 Caching

Cache improves performance by storing data for future requests, so data can be served faster. The data that is stored within a cache might be values that have been computed earlier or duplicates of original values that are stored elsewhere. Some caches such as CPU, HDD, etc. are generally managed entirely by hardware, a variety of software manages other caches. The page cache in main memory, which is an example of disk cache, is managed by the operating system kernel. Linux kernels 2.6.16 and newer provide a mechanism to have the kernel drop the page cache and/or inode and dentry caches on command.

To show how much the cache has affect on our tests, we have set up a new experiment.

7.2 Cache Experiment Setup

LAN network architecture is used for cache tests, because disabling cache requires root privileges. We setup an experiment between dsl-tie and dsl-stork with all possible cache combinations which are cache on both, no cache on both, cache on dsl-tie, and cache on dsl-stork. We have used dsl-tie as an execution node and dsl-stork as a data server node. Since we have used parrot/gsiftp as a main remote I/O protocol, we used the parrot/gsiftp combination to run remote I/O tests on cache too. We have created a bash script as a root user with the following commands:

```
1 #!/bin/sh
2 sync
3 echo 3 > /proc/sys/vm/drop_caches
```

This script saves the cache to local disk and frees pagecache, dentries, and inodes. We also created a cron job to run this script every 2 minutes as a root.

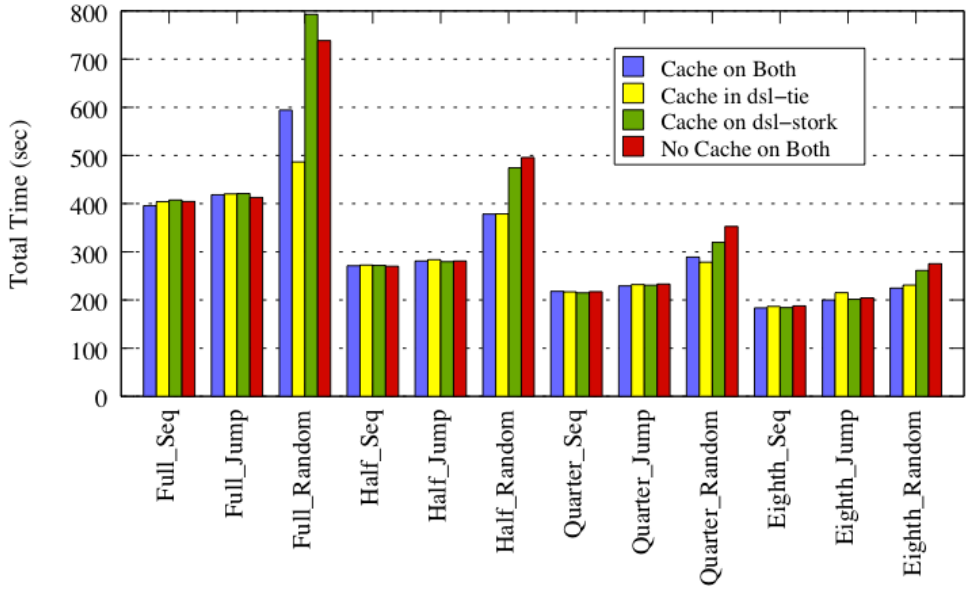

FIGURE 7.1: Cache Impact

7.3 Cache Experiment Results

Figure 7.1 shows the caching results for all data ratios and data access techniques combinations. The figure shows that cache on both has slightly better performance than the rest as it is expected. Then, cache on dsl-tie follows it. Because dsl-tie is the execution node, it is a normal result as well. No cache on both has worst result. Also, sequential data access technique with all data sizes have slightly better performance than jump data access technique. Since network performance plays an important role on random tests, random results are not balanced.

Chapter 8
Model

8.1 Initial Model

We would like to start with a limited set of parameters for simplicity and then extend the upon this. The parameters we use initially are:

- Rl: time to read from local disk to local memory
- Rr: time to read from remote disk to remote memory
- Wl: time to write from local memory to local disk
- Wr: time to write from remote memory to remote disk
- Ns: time to send data over network via a staging protocol
- Nr: time to send data over network via a remote I/O protocol
- E: time spent for computation
- P: total time spent for the application

The total time spent for the application via staging would be:

$$P_s = P_{in} + E + P_{out} \tag{8.1}$$

where

$$P_{in} = Rr_{in} + Ns_{in} + Wl_{in} + Rl_{in} \tag{8.2}$$

and

$$P_{out} = Wl_{out} + Rl_{out} + Ns_{out} + Wr_{out} \tag{8.3}$$

On the other hand, the total time spent for the application via remote I/O would be:

$$P_r = Rr_{in} + Nr_{in} + E + Nr_{out} + Wr_{out} \tag{8.4}$$

The actual computation time for both methods would be the same. The time difference in total (end-to-end) application time comes from network and disk I/O operations. For simplicity, we compare Input and Output operations separately.

In such a setting, for remote I/O to be more efficient than staging, we should have:

$$Rr_{in} + Nr_{in} < Rr_{in} + Ns_{in} + Wl_{in} + Rl_{in} \tag{8.5}$$

and

$$Nr_{out} + Wr_{out} < Wl_{out} + Rl_{out} + Ns_{out} + Wr_{out} \tag{8.6}$$

From Equation 5, we would get:

$$Nr_{in} - Ns_{in} < Wl_{in} + Rl_{in} \tag{8.7}$$

and similarly from Equation 6, we would get:

$$Nr_{out} - Ns_{out} < Wl_{out} + Rl_{out} \tag{8.8}$$

which means the time difference coming from using a specialized data transfer protocol versus a remote I/O protocol should be less than the overhead of extra read/write to the disk in staging. In other words, if your remote I/O library performs good in data transfer over network, or your local disk performance is slow, remote I/O might be advantageous over staging. Otherwise, staging method would perform better.

According to authors knowledge, Nrin and Nrout can not be measured separately. Since, these variables are plays important role in our initial mode, we would like to start with two separate models for each data access technique using our measurements with regression analysis in the following section.

8.2 Regression Models

In order to define the models for remote I/O and staging data access techniques in a distributed environment, a standard multiple regression analysis was performed using PASW Statistics 18, Release Version 18.0.0 (SPSS, Inc., 2009, Chicago,IL) [24].

Multiple regression analysis is a useful technique for predicting a dependent variable from several independent variables. Regressing y variable on several x variables is the main component of this approach. Regression equation symbolizes the best linear combination of independent variables and their associated weights to predict the dependent variable.

$$Y = a + b_1X_1 + b_2X_2 + ... + b_nX_n + e \tag{8.9}$$

where;

- Y is the predicted values of dependent variable
- a is the intercept/constant that the value of the Y when all the X values are equal to zero
- Xs are the independent variables
- b is the regression coefficients for each independent variable

Regression analysis uses the least square criterion that is the sum of the squares of errors should be minimum. Errors are described as the difference between observed and predicted values of the dependent variable.

Regression analysis requires continuous dependent variable and allows nominal (dichotomous) and continuous scale variables as independent variables [47]. Therefore, it is an appropriate method for the current study as regression coefficients of the each independent variable will help to determine the best models for staging and remote I/O techniques in distributed environments.

8.2.1 Regression Model for Data Staging

Performance of the application (Ps) is a dependent variable whereas Rrin, Nsin, Wlin, Rlin, Wlout, Rlout, Nsout, Wrout , DS (data size), AT(Access Techniques), NA (Network Architectures) were the

independent variables for data staging technique model. Full, half, quarter, and eighth are grouped as the data size (DS). NA variable indicates the network architectures such as LAN, CAN, MAN, WAN1, WAN2, WAN3, WAN4, WAN5, WAN6, WAN7. AT variable which is data access techniques (sequential, jump, and random) indicates categorical type of variable. Therefore, (j-1) approach of regression, when including categorical variable in the model, applied [64]. It is also suggested by Kleinbaum [51] et al. when including a nominal independent variable since it can help to index categories of the nominal variable in regression analysis. This method is also called reference cell coding and it is about using dummy variables for one less of the number of categories (j-1). In this study, AT variable which has 3 categories is recoded into 2 dummy variables (ATj and ATr) that have values of 0 or 1. ATj dummy variable is for jump access technique and Atr dummy variable is for random access technique. The sequential access technique is not coded in a separate variable since it is defined when both ATj and ATr variables equal to 0. The intercept of the regression model will indicate the coefficient for the sequential category.

The initial investigation of the variables is showed that dependent variable Ps and other independent variables are not normally distributed. In fact, as it can be seen in Figure 8.1 variables are severely skewed in the positive direction and logarithmic transformation was applied to make the variables normally distributed and meet the multiple regression assumptions. In multiple regression, observations should be independent, variables should be normally distributed, dependent variable and independent variables should be linearly correlated, and standard deviations of errors are equal for all predicted dependent variable scores that is called homoscedasticity.

Logarithmic transformation was used for variables Ps, Nsin, Wlin, Rlin, Wlout, Rlout, Nsout, and Wrout. Before logarithmic transformation, reflect method was applied for Rrin variable. Because of its relatively negative skewness reflect method helped to transform the distribution into the positive direction then logarithm applied. Table 8.1 shows that after the transformations all the skewness and kurtosis levels approached to the normal distribution values. In fact in perfectly normal distributions skewness and kurtosis levels are 0. Before the transformation skewness and kurtosis for the dependent variable Ps were 2.845 and 7.046 respectively. After the logarithmic transformation skewness and kurtosis values improved and became 1.488 and 1.310 respectively. This was evident for the other

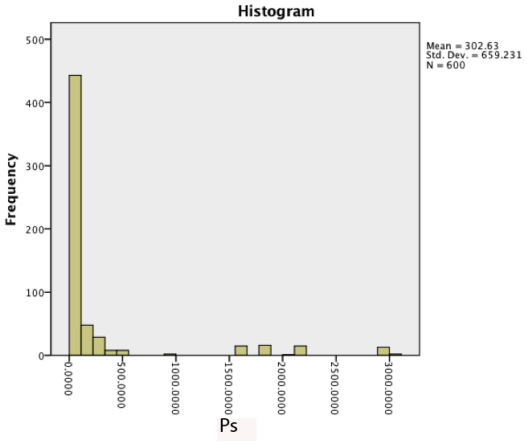

FIGURE 8.1: Staging Data Distribution Before the Transformation

variables. PASW Explore procedure was used to check each variables whether or not they improved in terms of normal distribution. Table 8.1 indicates the improvement of the data in terms of normality assumptions.

TABLE 8.1: Staging Skewness and Kurtosis Table

Variables	Skewness Before	Kurtosis Before	Skewness After	Kurtosis After
Ps	2.845	7.046	1.488	1.31
Nsin	2.634	5.034	1.336	0.969
Wlin	2.609	4.972	2.108	3.684
Rlin	4.451	20.156	1.026	1.366
Wlout	3.987	17.819	0.185	0.473
Rlout	4.451	20.156	1.026	1.366
Nsout	4.197	17.966	1.192	0.939
Wrout	2.351	5.049	0.439	-0.608
Rrin	-2.125	3.817	0.407	1.839

In addition to PASW Explore procedure, PASW Regression procedure was also used to check other assumptions of multiple regression. As mentioned above normality, linearity and homoscesdaticity are the assumptions of the standard regression procedure.

Kleinbaum et al. noted that normality and homoscedasticity was investigated by many researchers using the errors or in other words, residuals. Tabashnick and Fidell [47] stated that examination of residuals scatterplots provide information about assumptions of normality, linearity and homoscedasticity. The residuals (errors) are the differences between obtained and predicted dependent variable values. In fact, scatterplot between predicted dependent variable (DV) scores and error scores of

the prediction provides helpful information about these assumptions. Assumptions of the regression analysis can be summarized also as residuals are normally distributed about the predicted dependent variable scores, residuals have linear relationship with predicted dependent variable and variance of the residuals is same for all predicted values.Therefore, residuals were investigated for the regression assumptions. After transformation and removal of the outliers, residuals indicated normal distribution as it can be seen in Figure 8.2.

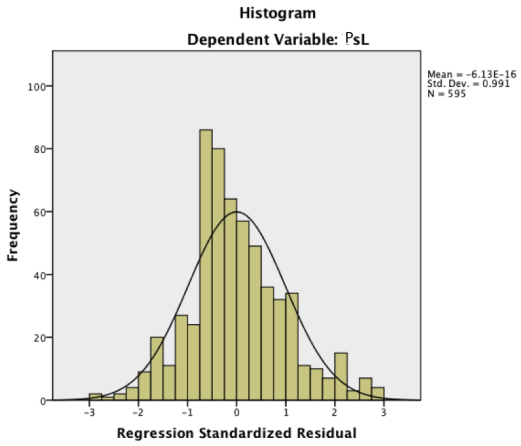

FIGURE 8.2: Staging Histogram After the Transformations

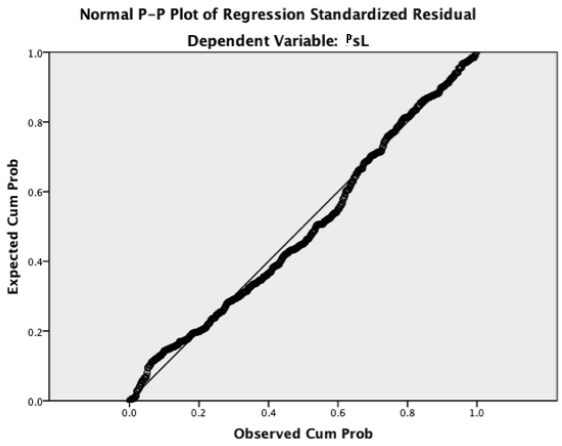

FIGURE 8.3: Normal P-P Plot of Regression Standardized Residual Dependent Variable PS After the Transformations

Figure 8.3 indicates that the residuals are normally distributed after the transformation.

Figure 8.4 shows the normality, linearity and homoscesdaticity assumptions. According to Tabashnick and Fidell [47], if scatterplot of residuals with standardized predicted dependent variable values display the scores scattered around the zero line it indicates that the normality, linearity and ho-

moscesdaticty assumptions are met. As values of the residuals scattered along the 0 line the assumptions were assumed to be met. Moreover, Tate [87] stated that moderate violations of the normality assumption may often be ignored with larger sample size as it does not affect the analysis negatively. Kleinbaum et al. also stated if normality assumption is not badly violated, the results reached by regression analysis will generally be reliable and accurate. In addition, Tabshnick and Fidell [47] noted that violations of linearity and homoscedasticity do not invalidate the analysis. Regarding the 595 observations, the sample was large enough and it was concluded that assumptions of regression analysis were met.

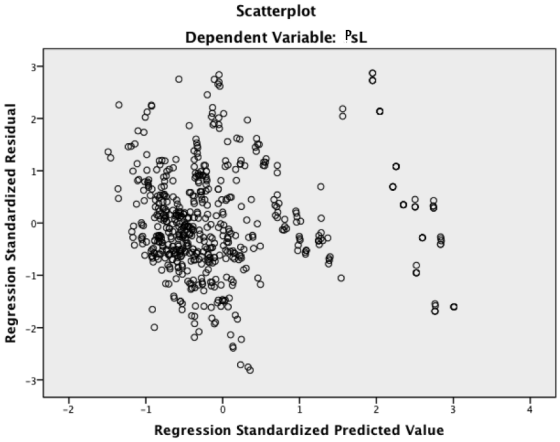

FIGURE 8.4: Staging Regression Standardized Predicted Values After the Transformations

8.2.1.1 Multivariate Outliers

By using Mahalanobis distance with p<.001, multivariate outliers were checked for a number of variables in the model and no multivariate outliers are detected in the data. In fact, maximum Mahalanobis distance were 31.366 and it was less than the X^2 critical value. N= 595 and no missing data observed.

8.2.1.2 Multicollinearity

Table 8.2 indicated multicollinearity (high correlation among independent variables) was an issue. However, some of the independent variables were highly correlated each other: NsoutL and NsinL (.832) and , RlinL and RloutL (1.), WloutL was highly correlated with RlinL (.912) and with RloutL (.912).

TABLE 8.2: Staging Correlation Table (N=595)

Corelations	Variables	PsL	RrinL	NsinL	WlinL	RlinL	WloutL	RloutL	NsoutL	WroutL	DS	NA	ATj	ATr
Correlation	PsL	1.000	-.663	.908	.363	.406	.464	.406	.933	.746	.298	.274	-.013	.093
	RrinL	-.663	1.000	-.589	-.101	-.093	-.166	-.093	-.695	-.358	.016	-.038	.002	-.004
	NsinL	.908	-.589	1.000	.426	.273	.286	.273	.832	.567	.003	.308	-.002	-.001
	WlinL	.363	-.101	.426	1.000	.681	.468	.681	.388	.195	-.016	-.350	.008	-.014
	RlinL	.406	-.093	.273	.681	1.000	.909	1.000	.367	.645	.602	-.380	.010	-.016
	WloutL	.464	-.166	.286	.468	.909	1.000	.909	.415	.728	.722	-.272	.010	-.015
	RloutL	.406	-.093	.273	.681	1.000	.909	1.000	.367	.645	.602	-.380	.010	-.016
	NsoutL	.933	-.695	.832	.388	.367	.415	.367	1.000	.602	.289	.284	.010	-.022
	WroutL	.746	-.358	.567	.195	.645	.728	.645	.602	1.000	.631	.039	.009	-.009
	DS	.298	.016	.003	-.016	.602	.722	.602	.289	.631	1.000	.006	.007	-.009
	NA	.274	-.038	.308	-.350	-.380	-.272	-.380	.284	.039	.006	1.000	-.001	.003
	ATj	-.013	.002	-.002	.008	.010	.010	.010	.010	.009	.007	-.001	1.000	-.499
	ATr	.093	-.004	-.001	-.014	-.016	-.015	-.016	-.022	-.009	-.009	.003	-.499	1.000
Sig. (1-tailed)	PsL	.	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.380	.011
	RrinL	.000	.	.000	.007	.012	.000	.012	.000	.000	.000	.347	.178	.482	.463
	NsinL	.000	.000	.	.000	.000	.000	.000	.000	.000	.000	.472	.000	.483	.492
	WlinL	.000	.007	.000	.	.000	.000	.000	.000	.000	.000	.350	.000	.420	.366
	RlinL	.000	.012	.000	.000	.	.000	.000	.000	.000	.000	.000	.000	.400	.344
	WloutL	.000	.000	.000	.000	.000	.	.000	.000	.000	.000	.000	.000	.408	.359
	RloutL	.000	.012	.000	.000	.000	.000	.	.000	.000	.000	.000	.000	.400	.344
	NsoutL	.000	.000	.000	.000	.000	.000	.000	.	.000	.000	.000	.000	.401	.296
	WroutL	.000	.000	.000	.000	.000	.000	.000	.000	.	.000	.000	.172	.413	.418
	DS	.000	.347	.472	.350	.000	.000	.000	.000	.000	.	.440	.428	.417
	NA	.000	.178	.000	.000	.000	.000	.000	.000	.172	.440	.	.492	.471
	ATj	.380	.482	.483	.420	.400	.408	.400	.401	.413	.428	.492	.	.000
	ATr	.011	.463	.492	.366	.344	.359	.344	.296	.418	.417	.471	.000	.

When regression analysis was applied it was observed that Rlin variable is excluded from the analysis by the statistical program due to perfect correlation with the Rlout variable. Since Rlout and Rlin are identical in their relationship with the performance (dependent variable) no further solution was attempted to include Rlin in the model. Moreover, including perfectly correlated independent variables in the model causes greater standard errors and coefficients will be shown as not significant in the model.

Other than the access technique jump, all the variables were significantly correlated with the dependent variable so multiple regression can be reliably used for this study.

Figures 8.4 and Figure 8.5 indicate that an overall model of eleven predictors that significantly predict the performance in staging data technique access, R^2 =.984 and adjusted R^2= .983 F(11,583)=3197.962 p<.001. This model is accounted for 98 % of the variance in performance of staging data access technique. In other words, 98 % of the variance in the PsL was explained by Rrin, Wlin, Wlout, Nsout, Wrout , DS, AT, and NA.

The coefficient of the intercept (constant) captures the coefficient of the sequential access technique. It was not coded separately as dummy variable in the equation regarding the (j-1) dummy variable approach. The coefficient of jump access technique(Atj) is the difference in means between jump and sequential. Similarly, the coefficient of random access technique for the Atr is the difference in means between random and sequential.

TABLE 8.3: Staging Descriptive Statistics

Variables	Mean	Std. Deviation	N
PsL	1.955210	.5491335	595
RrinL	.389748	.1883968	595
NsinL	1.250370	.7718463	595
WlinL	.407076	.2414208	595
RlinL	-.418420	.5205512	595
WloutL	.059109	.3614962	595
RloutL	-.418420	.5205512	595
NsoutL	1.047176	.7538912	595
WroutL	-.280588	.5636430	595
DS	2.49	1.117	595
NA	5.52	2.868	595
ATj	.33	.472	595
ATr	.33	.471	595

TABLE 8.4: Staging Model Summary

R	R^2	Adjusted R^2	Std. Error of the Estimate
.992	.984	.983	.0707732

TABLE 8.5: Staging ANOVA

	Sum of Squares	df	Mean Square	F	Sig.
Regression	176.199	11	16.018	3197.962	.000
Residual	2.920	583	.005		
Total	179.119	594			

Table 8.22 displays, the unstandardized regression coefficients (B) and intercept (constant), t test statistics and significance levels, tolerance, and VIF.

TABLE 8.6: Coefficients for Staging Model Variables

Variables	B	t	p	Tolerance	VIF
(Constant)	1.12	38.207	0		
RrinL	-0.266	-8.717	0	0.256	3.906
NsinL	0.27	25.72	0	0.129	7.77
WlinL	0.281	7.082	0	0.092	10.919
WloutL	-0.013	-0.529	0.597	0.106	9.469
RloutL	-0.179	-7.132	0	0.05	20.131
NsoutL	0.258	20.928	0	0.098	10.231
WroutL	0.295	22.484	0	0.154	6.501
DS	0.057	8.284	0	0.143	6.997
NA	0.003	1.96	0.05	0.381	2.625
ATj	0.052	7.307	0	0.751	1.332
ATr	0.146	20.561	0	0.749	1.335

Since there was a perfect relationship (1.) between the RlinL and RloutL, RlinL was excluded from the analysis by a statistical program due to the multicollinearity. High multicollinearity can be detected from variance inflator factor (VIF) or Tolerance. Coefficients Table 8.22 provides information about both VIF and Tolerance. VIF value higher like 10 indicates the possible multicollinearity whereas values that are closer to 0 for the Tolerance indicates the collinearity. Coefficients table display high multicollinearity for Rlout which was greater than 20. Since the synthetic application creates the same amount of data, Rlout and Rlin operate the same amount of data. It is not surprising that both variables have strong relationship. T test statistics results indicate that coefficients of the all independent variables are significant except WloutL. This insignificant b(-0.013) for WloutL and significant F (11,583)=3197 is a sign of multicolliniarity. However, since multicollaniarity is normal with this type of data. It doesn't affect the inclusion of this coefficient in the regression equation.

PsL is estimated by the following regression model.

$$Ps = 1.133 + (-.240)(R_{rinL}) + .268(N_{sinL}) + .254(W_{linL}) + (-.009)(W_{loutL}) +$$
$$(-.157)(R_{loutL}) + (.269)(N_{soutL}) + .290(W_{routL}) + (.051)(D_S) + .003(N_A) +$$
$$.054(A_{Tj}) + .154(A_{Tr}) \quad (8.10)$$

8.2.1.3 Clarify Program

Clarify program is developed by Gerry King et al. [3]. It is useful for predicting the dependent variable using one of the independent variables while controlling for the effects of the all of the other independent variables constant in the model. Table 8.7 shows the magnitude of the effect of the independent variables: i) data size; ii) network architecture; iii) access technique on the dependent variable (PsL) when all the other independent variables are kept constant at their mean scores. Clarify program shows us that how each variable affects the PsL while keeping the other variables constant on their mean scores.

TABLE 8.7: Clarify Program for Staging

Variables	TsL Mean
DS Full	2.041586
DS Half	1.984506
DS Quarter	1.927427
DS Eighth	1.870347
LAN	1.8559
CAN	1.859095
MAN	1.862291
WAN1	1.865486
WAN2	1.868682
WAN3	1.871877
WAN4	1.875073
WAN5	1.878268
WAN6	1.881464
WAN7	1.884659
ATj	1.919147
ATr	2.017057
ATs	1.818865

8.2.2 Regression Model for Remote I/O

Performance of the application (Pr) was the dependent variable whereas Rrin, Nr, and Wrout, DS(data size), AT(Access Techniques), NA (Network Architectures) were the independent variables for remote data access techniques in distributed environments.

DS variable indicated transaction and data size including Full, half, quarter, and eighth (DS). NA variable includes the network architectures such as LAN, CAN, MAN, WAN1, WAN2, WAN3, WAN4, WAN5, WAN6, and WAN7. AT variable which were data access techniques (sequential, jump, and random) indicated categorical type of variable. Therefore (j-1) approach of regression or in other term reference cell coding, when including categorical variable in the model was applied in PASW.

The initial investigation of the variables showed that dependent variable Pr and other independent variables were not normally distributed. In fact, most of the variables were positively skewed as shown in the figure 8.5.

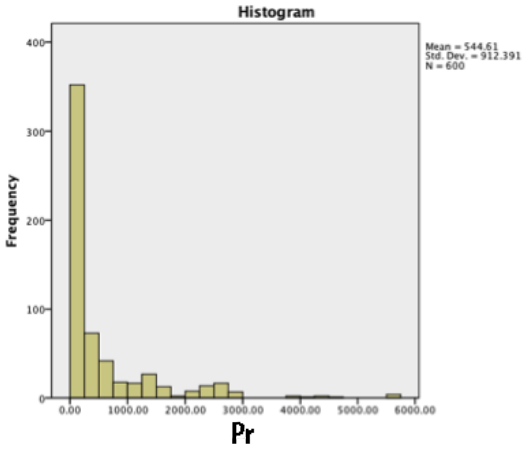

FIGURE 8.5: Remote I/O Histogram Before the Transformations

Several transformations were performed such as square root, inverse and logarithmic. Logarithmic transformation was the best among the others it was applied to make the variables closer to normal distribution and meet the multiple regression assumptions which are the variables have normal distribution, the dependent and the independent variables are linearly related (linearity) and homoscesdaticity (variance of errors for every values of independent variables is equal).

Logarithmic transformation was used for variables Pr, Rrin, Nr, and Wrout. After the transformations, all the skewness and kurtosis levels approached to the normal distribution values. Also,

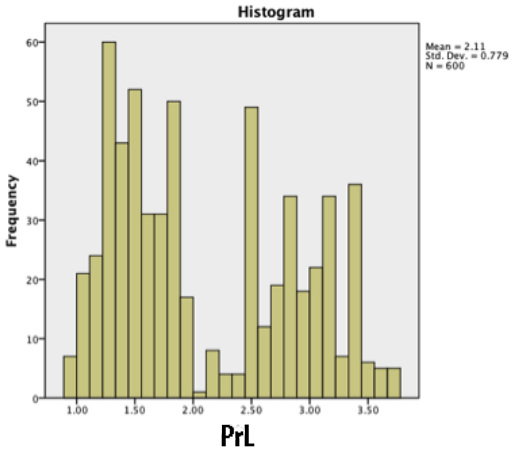

FIGURE 8.6: Remote I/O Histogram After the Transformations

"L" is added to end of the variable name to indicate log(logarithmic) transformation. In perfectly normal distribution, skewness and kurtosis levels are supposed to be 0. Before the transformations, skewness and kurtosis for the dependent variable Pr were 2.845 and 7.046 respectively. After the log transformation, skewness and kurtosis values improved and became .360 and -1.270 respectively. PASW Explore procedure was used to check each variable whether or not they improved in terms of normal distribution. The table 8.8 indicates the improvement of the data in terms of normality assumptions.

TABLE 8.8: Skewness and Kurtosis Table for Remote I/O Model

Variables	Skewness Before	Kurtosis Before	Skewness After	Kurtosis After
Pr	2.570	7.903	.344	-1.280
RrinL	3.278	12.966	.679	-.127
NrL	2.335	6.020	.187	-1.509
WroutL	1.770	4.479	.165	-.827

Normality, linearity and homoscesdaticity assumptions of the multiple regression analysis procedure were checked through residuals analysis. The residuals are the differences between obtained and predicted dependent variable values. And they assumed to be normally distributed, have a linear relationship with predicted dependent variable, and variances of the residuals are the same.

It can be seen from Figures 8.7 and Figure 8.8 that after the log transformation, histogram of residuals indicated improved normal distribution.

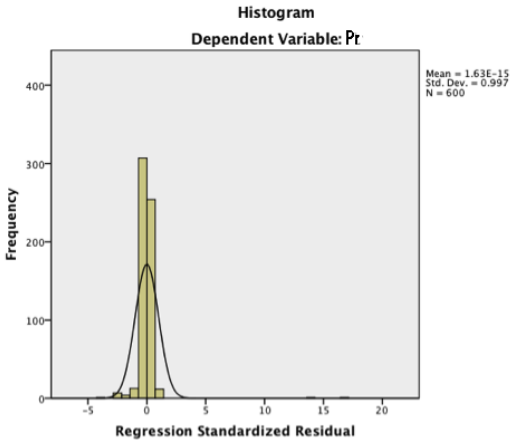

FIGURE 8.7: Standardized Residuals for Remote I/O Before Transformation

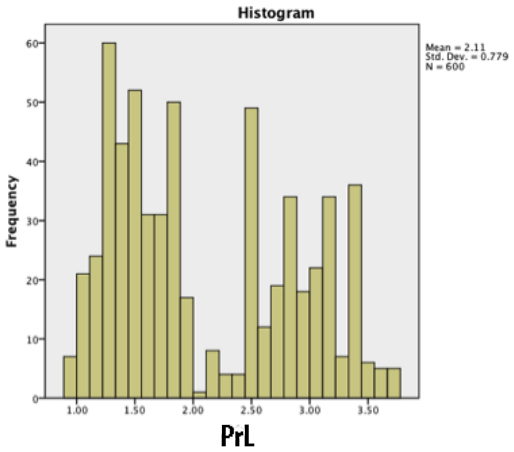

FIGURE 8.8: Standardized Residuals for Remote I/O After Transformation

Moreover, Figures 8.9 and Figure 8.10 also display the improved normality after the log transformation. If the observations are located on the straight line, it indicates the perfect normal distribution.

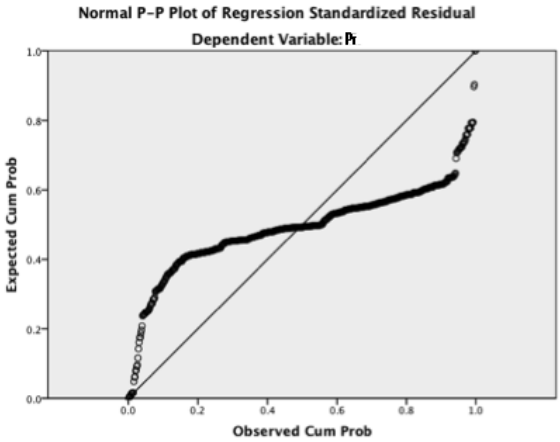

FIGURE 8.9: Normal PP Plots for Remote I/O Before Transformation

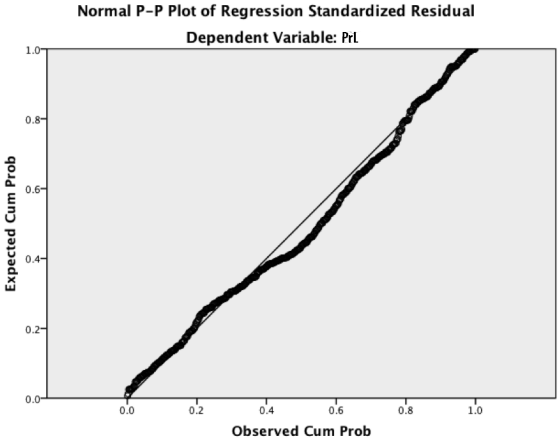

FIGURE 8.10: Normal PP Plots for Remote I/O After Transformation

The scatterplots of the standardized residuals and standardized predicted values were improved after the log transformation. In fact, scatterplots of residuals with predicted values are helpful in checking the linearity, normality and homosesdaticity.

8.2.2.1 Multivariate Outliers

By using Mahalanobis distance value from the regression analysis with $p<.001$, multivariate outliers were checked and no multivariate outliers are detected in the data.

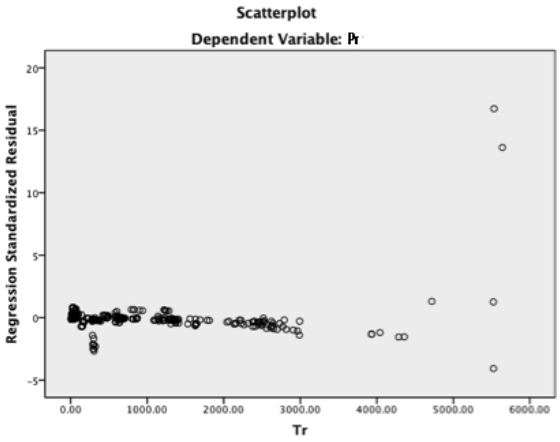

FIGURE 8.11: Scatterplot Before Transformation

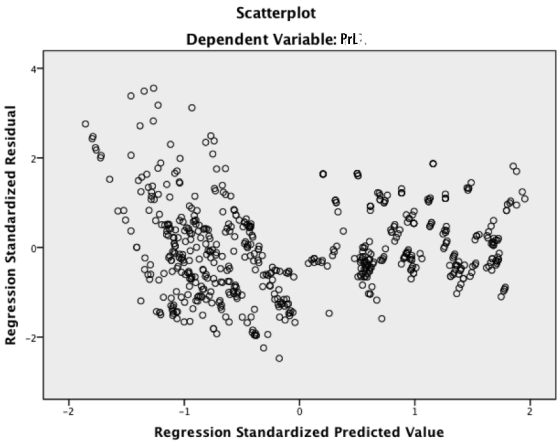

FIGURE 8.12: Scatterplot After Transformation

8.2.2.2 Multicollinearity

Correlations between dependent variable PrL and independent variables (RrinL, NrL, WroutL, DS, NA, Atj, and Atr) were checked and it was observed that all the independent variables were significantly ($p<0.05$) correlated with the dependent variable (PrL) ranging from .111 to .966. However, multicolliniearity also detected between RrinL and WroutL since both of these independent variables were highly correlated (.907) and DS variable also were highly correlated with WroutL variable (.911). However tolerance value and VIF indicated acceptable levels of Multicollinearity since the tolerance values were not closer to 0 .All the variables remained in the model.

TABLE 8.9: Descriptive Statistics for Remote I/O

Variables	Mean	Std. Deviation	N
TrL	2.118533	.7783999	593
RrinL	.125295	.3420950	593
NrL	1.672715	1.1469137	593
WroutL	.367521	.2912515	593
DS	2.49	1.120	593
NA	5.47	2.875	593
ATj	.34	.474	593
ATr	.33	.470	593

TABLE 8.10: Correlations for Remote I/O (N=593)

Correlation	Variables	PrL	RrinL	NrL	WroutL	DS	NA	ATj	ATr
Correlation	PrL	1.000	.357	.973	.358	.355	.115	-.340	.799
	RrinL	.357	1.000	.243	.909	.880	-.152	-.004	.009
	NrL	.973	.243	1.000	.213	.210	.112	-.374	.801
	WroutL	.358	.909	.213	1.000	.912	-.076	.001	.007
	DS	.355	.880	.210	.912	1.000	-.006	-.003	.012
	NA	.115	-.152	.112	-.076	-.006	1.000	.014	-.008
	ATj	-.340	-.004	-.374	.001	-.003	.014	1.000	-.503
	ATr	.799	.009	.801	.007	.012	-.008	-.503	1.000
Sig. (1-tailed)	PrL	.	.000	.000	.000	.000	.003	.000	.000
	RrinL	.000	.	.000	.000	.000	.000	.461	.412
	NrL	.000	.000	.	.000	.000	.003	.000	.000
	WroutL	.000	.000	.000	.	.000	.032	.487	.436
	DS	.000	.000	.000	.000	.	.440	.475	.388
	NA	.003	.000	.003	.032	.440	.	.363	.427
	ATj	.000	.461	.000	.487	.475	.363	.	.000
	ATr	.000	.412	.000	.436	.388	.427	.000	.

TABLE 8.11: Remote I/O Model Summary

R	R^2	Adjusted R^2	Std. Error of the Estimate
.991	.982	.981	.1062674

TABLE 8.12: ANOVA

	Sum of Squares	df	Mean Square	F	Sig.
Regression	352.090	7	50.299	4454.058	.000
Residual	6.606	585	.011		
Total	358.697	592			

Regression results indicated an overall model of nine predictors that significantly predict the performance in remote I/O data access technique, R^2=.975, F(7,592)= 3322.606, p<.001. This model is accounted for nearly 98 % of the variance in performance of remote I/O data access technique. The coefficient of the intercept (constant) captures the coefficient of the sequential access technique as it was not included as a dummy variable in the equation because of the (j-1) dummy variable approach. The coefficient of jump access technique(Atj) is the difference in means between jump and sequential. Similarly, the coefficient of random access technique for the Atr is the difference in means between random and sequential.

$$Pr = .735+(-.160)(R_{rinL})+.527(N_{rL})+.474(W_{routL})+(.065)(D_S)+.011(N_A)+.080(A_{Tj})+.322(A_{Tr}) \tag{8.11}$$

Table 8.13 displays the unstandardized regression coefficients (B) and intercept (constant), the R^2 and adjusted R^2. R^2 for regression significantly differ from zero =.975 F(7,592)= 3322.606 p<.001. This indicates that the combination of the independent variables (the model) explained nearly 98 % of the variation in the remote performance (Pr). In other words, R^2 at .975 indicated that almost 98 % of the performance was predicted by Rrin, Nr, Wrout , DS, AT, and NA.

Pr for remote I/O data access technique is estimated by the following regression model. This model helps to explain almost 98 % of the variance in the performance regarding remote I/O data access technique.

TABLE 8.13: Coefficients for Remote I/O Model Variables

Variables	B	t	p	Tolerance	VIF
(Constant)	0.73	35.926	0		
RrinL	-0.147	-4.258	0	0.136	7.36
NrL	0.553	76.052	0	0.274	3.65
WroutL	0.412	9.435	0	0.118	8.469
DS	0.068	6.633	0	0.145	6.915
NA	0.007	4.327	0	0.822	1.216
ATj	0.082	7.702	0	0.744	1.344
ATr	0.281	15.538	0	0.264	3.785

8.2.2.3 Clarify Program

Table 8.14 shows the affect of the following variables: i) data size; ii) network architecture; iii) access technique over PrL when all the other independent variables are kept constant at their mean scores. Clarify program shows us that how each variable affects the PrL while keeping the other variables constant on their mean scores.

TABLE 8.14: Clarify Program for Remote I/O

Variables	PrL
DS Full	2.221657
DS Half	2.153256
DS Quarter	2.084855
DS Eighth	2.016454
LAN	1.984251
CAN	1.99146
MAN	1.998669
WAN1	2.005878
WAN2	2.013087
WAN3	2.020296
WAN4	2.027504
WAN5	2.034713
WAN6	2.041922
WAN7	2.049131
ATj	2.166752
ATr	2.331526
ATs	1.863906

8.2.3 Alternative Regression Model for Data Staging

Alternatively, another regression analysis was run using the variables network architecture, data size, and access method in a dummy coded format. When a variable dummy coded the number of the dummy coded category should be (J-1) that is 1 less number of categories in the respective variable. For instance, access method variable has 3 categories and 2 dummy coded variables generated excluding the sequential method, data size (4 categories: full, half, quarter, and eighth) 3 dummy coded variables were generated excluding the full category, and 9 dummy variables were generated for the network architecture variable. After removing the outliers, the residuals showed the normal distribution as it can be seen from Figure 8.13 histogram and P-P plot and Figure 8.14 .

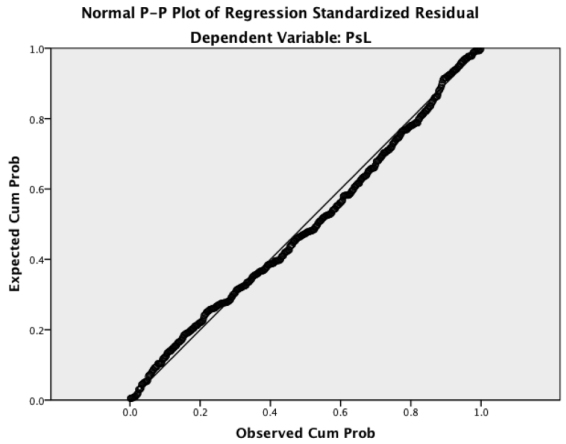

FIGURE 8.13: Alternative Normal PP Plots for Staging After Transformation

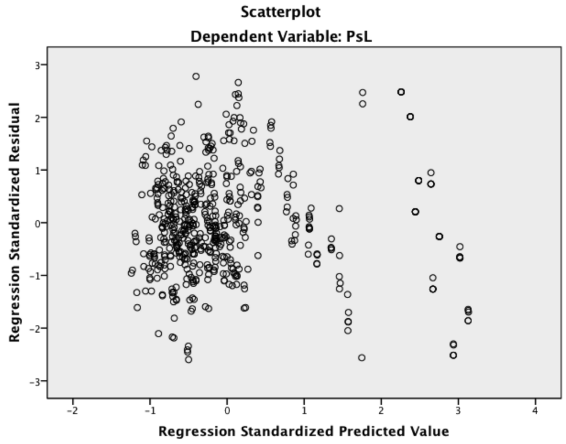

FIGURE 8.14: Alternative Staging Scotterplot After the Transformations

TABLE 8.15: Alternative Staging Descriptive Statistics

Variables	Mean	Std. Deviation	N
PsL	1.930786	.5220259	585
RrinL	.397162	.1837848	585
NsinL	1.218786	.7381837	585
WlinL	.407983	.2433845	585
RlinL	-.415333	.5208294	585
WloutL	.060068	.3622168	585
RloutL	-.415333	.5208294	585
NsoutL	1.014974	.7218033	585
WroutL	-.293726	.5549477	585
NaCan	.10	.301	585
NaMan	.10	.304	585
NaWan1	.10	.304	585
NaWan2	.10	.299	585
NaWan3	.10	.304	585
NaWan4	.10	.304	585
NaWan5	.10	.304	585
NaWan6	.10	.304	585
NaWan7	.09	.280	585
ATj	.34	.475	585
ATr	.32	.469	585
DSh	.26	.437	585
DSq	.26	.437	585
DSe	.25	.431	585

All the variable except ATj were significantly correlated with the PsL (dependent variable) since all the p values were less than 0.05.

The model summary Table 8.16 shows how much variation in the dependent variable was explained by the model. R^2 value of .992 indicates 99% of the variance in the PsL was explained by the independent variables that are in the model.

TABLE 8.16: Alternative Staging Model Summary

R	R^2	Adjusted R^2	Std. Error of the Estimate
.996	.992	.991	.0465396

TABLE 8.17: Alternative Staging ANOVA

	Sum of Squares	df	Mean Square	F	Sig.
Regression	157.925	20	7.896	3645.659	.000
Residual	1.222	564	.002		
Total	159.146	584			

ANOVA Table 8.17 shows that this model is significant since p value is so low even it is less than 0.0001. F (20, 564)= 3645.659 and p<0.05. Since F test statistic is significant. It can be concluded that this model is significantly predicting the PsL.

$$\begin{aligned} Ps = 1.306 + .014(R_{rinL}) + .144(N_{sinL}) + .819(W_{linL}) + .018(W_{loutL}) + (-.090)R_{loutL} + \\ .098(N_{soutL}) + .084(W_{routL}) + .124(Na_{Can}) + (-.003)Na_{Man} + .157(Na_{Wan1}) + \\ .092(Na_{Wan2}) + 0.92(Na_{Wan3}) + 0.33(Na_{Wan4}) + .029(Na_{Wan6}) + 1.038(Na_{Wan7}) + \\ .056(AT_j) + .153(AT_r) + (-.177)DS_h + (-.295)DS_q + (-.383)DS_e \end{aligned} \quad (8.12)$$

The coefficients for dummy coded variables indicated the difference between the excluded category and the respective dummy coded variable. RlinL and NaWan5 are excluded from the analysis due to the multicollinearity.

TABLE 8.18: Alternative Coefficients for Staging Model Variables

Variables	B	t	p
(Constant)	1.306	6.413	0
RrinL	.014	.070	.944
NsinL	.144	10.267	0
WlinL	.819	6.392	0
WloutL	.018	.960	.337
RloutL	-.090	-2.968	.003
NsoutL	.098	5.191	0
WroutL	.084	5.828	0
NaCan	.124	1.186	.236
NaMan	-.003	-.040	.968
NaWan1	.157	1.387	.166
NaWan2	.092	1.021	.308
NaWan3	.092	1.052	.293
NaWan4	.033	.404	.686
NaWan6	.029	.372	.710
NaWan7	1.038	5.821	0
ATj	.056	12.011	0
ATr	.153	31.985	0
DSh	-.177	-15.901	0
DSq	-.295	-15.494	0
DSe	-.383	-13.353	0

8.2.4 Alternative Regression Model for Remote I/O

Regression analysis was run using the variables network architecture, data size, and access method in a dummy coded format. Access method variable has 3 categories and 2 dummy coded variables generated excluding the sequential method, data size (4 categories: full, half, quarter, and eighth) 3 dummy coded variables were generated excluding the full category, and 9 dummy variables were generated for the network architecture variable. After removing the outliers, the residuals showed the normal distribution as it can be seen from Figure 8.15 histogram and P-P plot and Figure 8.16 .

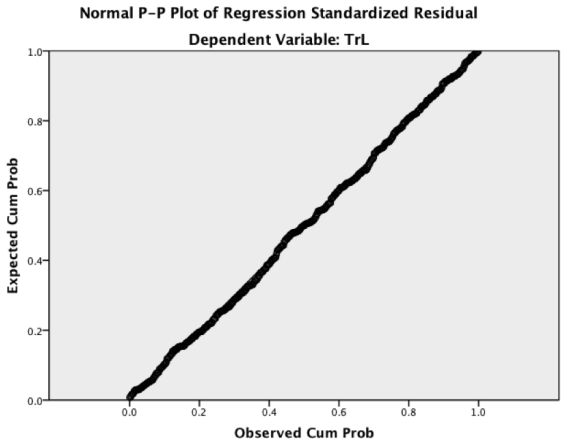

FIGURE 8.15: Alternative Normal PP Plots for Remote I/O After Transformation

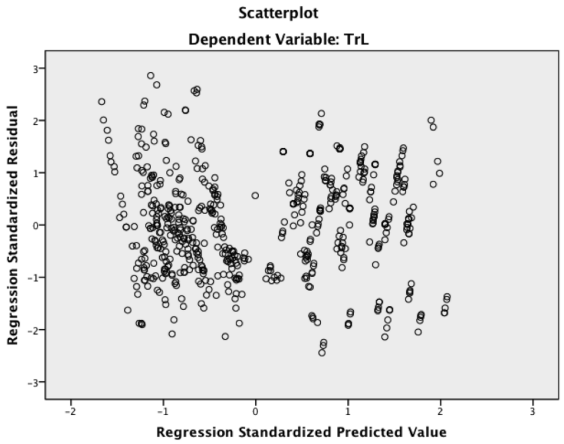

FIGURE 8.16: Alternative Remote I/O Scatterplot After the Transformations

Table 8.16 shown scatterplot of the standardized residuals with the standardized predicted value allows us to check homoscesdaticity, linearity and normality. The regression assumptions are met since the scatterplot does not show any pattern like a funnel.

TABLE 8.19: Alternative Staging Descriptive Statistics for Remote I/O

Variables	Mean	Std. Deviation	N
PrL	2.1239	.78170	584
RrinL	.1253	.34228	584
NrL	1.6900	1.13953	584
WroutL	.3650	.29150	584
NaCAN	.10	.302	584
NaMAN	.10	.304	584
NaWAN	.10	.302	584
NaWAN2	.10	.304	584
NaWAN3	.10	.302	584
NaWAN4	.10	.304	584
NaWAN5	.08	.275	584
NaWAN6	.10	.304	584
NaWAN7	.10	.302	584
ATj	.34	.474	584
ATr	.33	.471	584
Dsh	.25	.433	584
Dsq	.25	.433	584
Dse	.25	.435	584

The model summary Table 8.20 shows how much variation in the dependent variable was explained by the model. R^2 value of .992 indicates 99% of the variance in the PrL was explained by the independent variables that are in the model.

TABLE 8.20: Alternative Remote I/O Model Summary

R	R^2	Adjusted R^2	Std. Error of the Estimate
.996	.992	.991	.07065

TABLE 8.21: Alternative Remote I/O ANOVA

	Sum of Squares	df	Mean Square	F	Sig.
Regression	353.421	17	20.789	4164.793	.000
Residual	2.825	566	.005		
Total	356.246	583			

ANOVA Table 8.21 shows that this model is significant since p value is so low even it is less than 0.0001. Since F test statistic is significant, it can be concluded that this model is significantly predicting the PrL.

TABLE 8.22: Alternative Coefficients for Remote I/O Model Variables

Variables	B	t	p
(Constant)	1.214	25.878	0
RrinL	.075	2.335	.020
NrL	.502	68.593	0
WroutL	.113	2.423	.016
NaCAN	.017	.840	.401
NaMAN	-.019	-1.003	.316
NaWAN1	-.094	-3.798	0
NaWAN2	-.092	-5.304	0
NaWAN3	-.013	-.660	.509
NaWAN4	-.094	-5.149	0
NaWAN5	.112	5.566	0
NaWAN6	-.066	-3.842	0
NaWAN7	.200	9.988	0
ATj	.083	11.602	0
ATr	.390	24.199	0
DSh	-.102	-6.685	0
DSq	-.177	-6.713	0
DSe	-.269	-7.467	0

$$Pr = 1.214 + .075(R_{rinL}) + .502(N_{rL}) + .113(W_{routL}) + .017(Na_{Can}) + (-.019)Na_{Man} + (-.094)(Na_{Wan1}) + .092(Na_{Wan2}) + (-.013)(Na_{Wan3}) + (-.094)(Na_{Wan4}) + .112(Na_{Wan5}) + (-.066)(Na_{Wan6}) + .200(Na_{Wan7}) + .083(AT_j) + .390(AT_r) + (-.102)DS_h + (-.177)DS_q + (-.269)DS_e \quad (8.13)$$

RlinL and NaWan5 are excluded from the analysis due to the multicollinearity.

Chapter 9
Model Validation

Two real-life applications, Hurricane Data Archive and Blast, are used to test the validity of our model. In addition to these real-life applications, we have also used modified versions of our synthetic applications to test some extreme cases. The following sections provide the tests and their results.

9.1 Real-Life Applications

9.1.1 Data Archive for Coastal Science

Coastal/Hurricane research group on CCT has been developing a Simulated Hurricane Database [37] hosted on Petashare[22] containing data produced from ADCIRC [1] application. Initially, archive database is populated by ADCIRC runs for hurricanes and tropical storms that have occurred in the Golf of Mexico over the past 50 years. Then, archive database provides information to additional application for hypothetical storm events. Some applications are available to analyze the simulated hurricane database We will use the application program to find out which remote data access technique is the best method.

9.1.1.1 Data Archive Experiment Design

The simulated hurricane database is around 38 GByte. Spider is used as an execution node, and eric is used as a data server node. So, we have used CAN network architecture for this experiment. We have staged in all the data files from eric to spider by globus-url-copy. After the execution, we have staged out all the files from spider to eric back in staging tests. We have repeated the tests five times to eliminate network affects. Application program has accessed the simulated hurricane database which is on eric with parrot/gsiftp remote I/O protocol. This application accesses small portion of the database, so we have used eighth data size. The application reads the database sequentialy.

9.1.1.2 Experiment Results

Figure 9.1 provides both staging and remote I/O results. Remote I/O has better performance than staging.

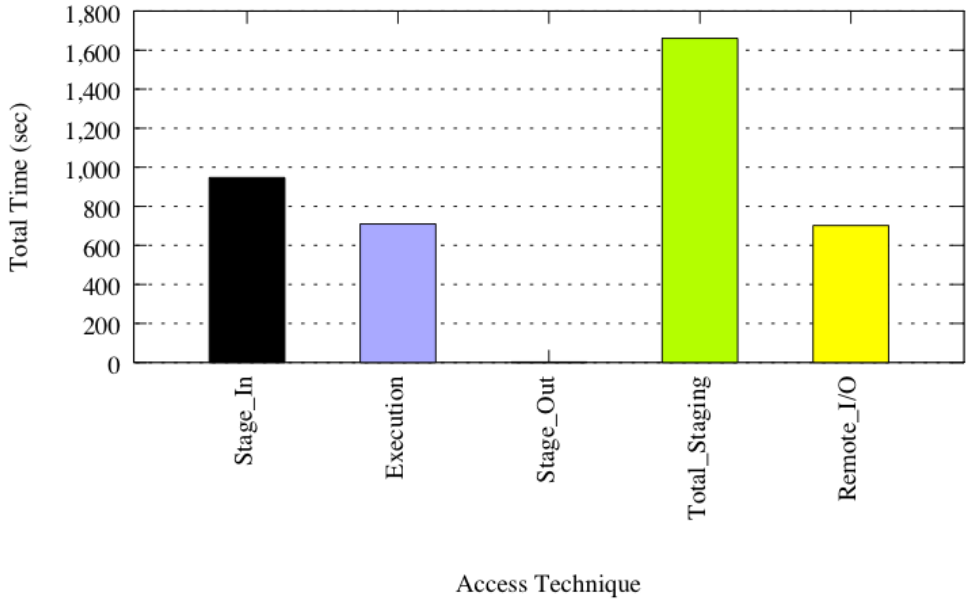

FIGURE 9.1: Simulated Hurricane Database Results

9.1.1.3 Model Results

We applied our models as follows:

Staging

$$Ps = 1.133 + (-.240)(R_{rinL}) + .268(N_{sinL}) + .254(W_{linL}) + (-.009)(W_{loutL}) + (-.157)(R_{loutL}) + (.269)(N_{soutL}) + .290(W_{routL}) + (.051)(D_S) + .003(N_A) + .054(A_{Tj}) + .154(A_{Tr}) \quad (9.1)$$

When we replace the variables with their corresponding values, the equation becomes:

$$Ps = 1.133 + (-.240)(350.85) + .268(358.02) + .254(180.29) + (-.009)(22.54) + (-.157)(7.84) + (.269)(614.58) + .290(56.43) + (.051)(1) + .003(2) + .054(0) + .154(0) \quad (9.2)$$

$$Ps = 239.126 \quad (9.3)$$

Remote I/O

$$Pr = .735+(-.160)(R_{rinL})+.527(N_{rL})+.474(W_{routL})+(.065)(D_S)+.011(N_A)+.080(A_{Tj})+.322(A_{Tr}) \tag{9.4}$$

When we replace the variables with their corresponding values, the equation becomes:

$$Pr = .735+(-.160)(89.99)+.527(103.22)+.474(108.91)+(.065)(1)+.011(2)+.080(0)+.322(0) \tag{9.5}$$

$$Pr = 92.64 \tag{9.6}$$

Our model also shows that remote I/O has better performance score. Real-life application confirms the validity of our model.

9.1.2 Blast

Blast compares a sequence with those contained in nucleotide and protein databases by aligning the sequence with previously characterized genes. It finds regions of sequence similarity, which will yield functional and evolutionary clues about the structure and function of this sequence.

Blast is used for the second real-life application. Blast DNA database refseq_rna is used, it is around 1.3 GByte. Spider is used as an execution node and Queen Bee is used as a data server node (MAN). This application accesses all database, so we have used full data size The application reads the database sequentially.

9.1.2.1 Experiment Results

According to the Figure 9.2, remote I/O also performing better then staging in our second real-life application.

9.1.2.2 Model Results

We applied our models as follows:

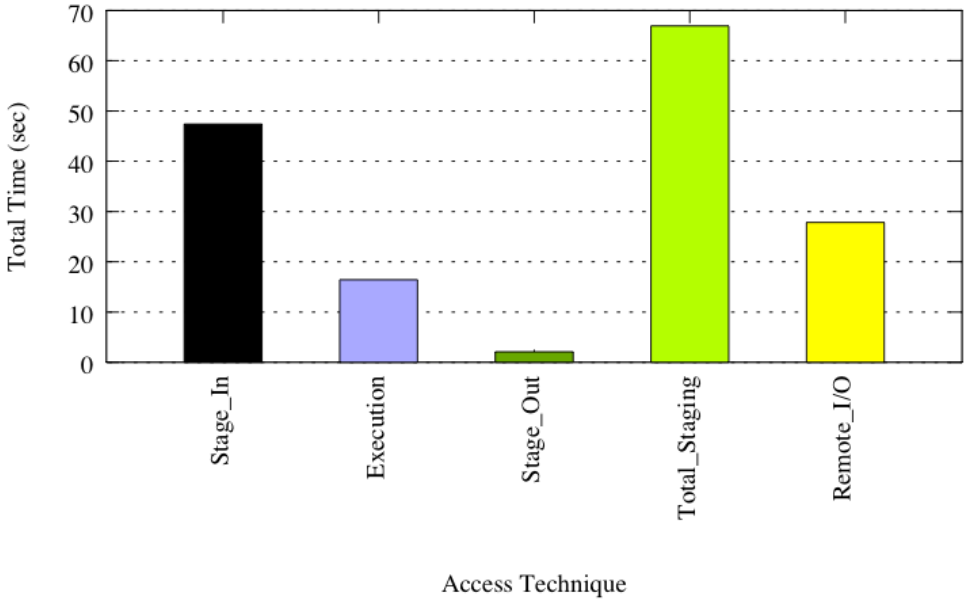

FIGURE 9.2: Blast Results

Staging

$$Ps = 1.133 + (-.240)(R_{rinL}) + .268(N_{sinL}) + .254(W_{linL}) + (-.009)(W_{loutL}) + (-.157)(R_{loutL}) + (.269)(N_{soutL}) + .290(W_{routL}) + (.051)(D_S) + .003(N_A) + .054(A_{Tj}) + .154(A_{Tr}) \quad (9.7)$$

When we replace the variables with their corresponding values, the equation becomes:

$$Ps = 1.133 + (-.240)(11.23) + .268(30.55) + .254(5.62) + (-.009)(0.01) + (-.157)(0.001) + (.269)(2.06) + .290(0.03) + (.051)(4) + .003(2) + .054(0) + .154(0) \quad (9.8)$$

$$Ps = 8.83 \quad (9.9)$$

Remote I/O

$$Pr = .735 + (-.160)(R_{rinL}) + .527(N_{rL}) + .474(W_{routL}) + (.065)(D_S) + .011(N_A) + .080(A_{Tj}) + .322(A_{Tr}) \quad (9.10)$$

When we replace the variables with their corresponding values, the equation becomes:

$$Pr = .735 + (-.160)(6) + .527(7.29) + .474(0.03) + (.065)(4) + .011(3) + .080(0) + .322(0) \quad (9.11)$$

$$Pr = 5.84 \quad (9.12)$$

Our model also shows that remote I/O has better performance score. Real-life application confirms the validity of our model.

9.2 Synthetic Applications

We have used modified versions of our synthetic applications to test the validity of our models with different use cases. We will report two use cases in this section: full ratio and eighth ratio.

9.2.1 Synthetic Application with Full Ratio

We have used our synthetic application and synthetic data. Spider is used as an execution node and Painter is used as a data server node (WAN4). This application accesses all data, so we have used full data size. The application reads the database randomly.

9.2.1.1 Experiment Results

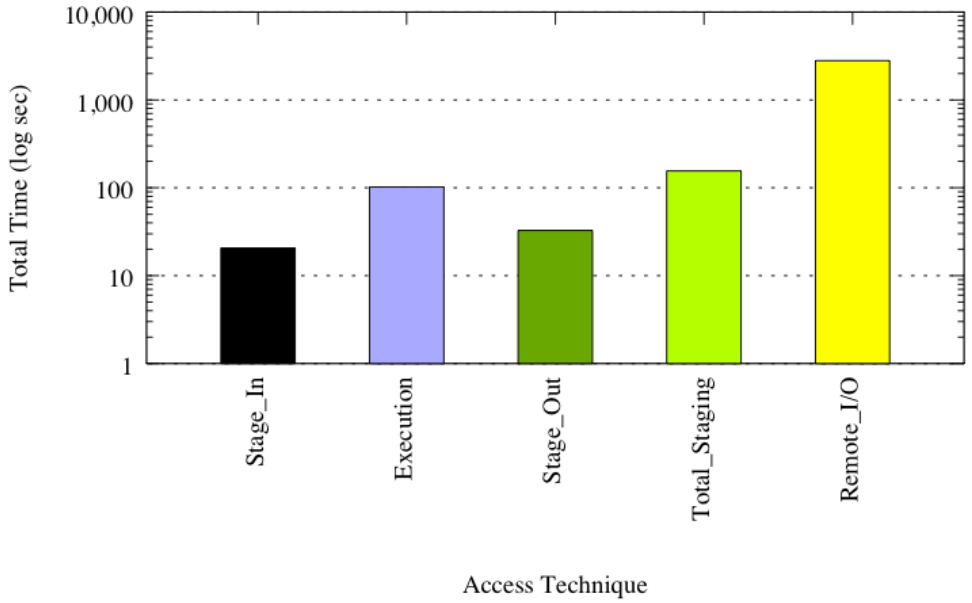

FIGURE 9.3: Synthetic Application Results with Full Ratio

According to the Figure 9.3, staging is performing better then remote I/O in our first case synthetic application.

9.2.1.2 Model Results

We applied our models as follows:

Staging

$$Ps = 1.133 + (-.240)(R_{rinL}) + .268(N_{sinL}) + .254(W_{linL}) + (-.009)(W_{loutL}) + (-.157)(R_{loutL}) + (.269)(N_{soutL}) + .290(W_{routL}) + (.051)(D_S) + .003(N_A) + .054(A_{Tj}) + .154(A_{Tr}) \quad (9.13)$$

When we replace the variables with their corresponding values, the equation becomes:

$$Ps = 1.133 + (-.240)(5.03) + .268(13.00) + .254(2.6) + (-.009)(2.6) + (-.157)(0.9) + (.269)(31.02) + .290(0.78) + (.051)(4) + .003(7) + .054(0) + .154(0) \quad (9.14)$$

$$Ps = 12.70 \quad (9.15)$$

Remote I/O

$$Pr = .735 + (-.160)(R_{rinL}) + .527(N_{rL}) + .474(W_{routL}) + (.065)(D_S) + .011(N_A) + .080(A_{Tj}) + .322(A_{Tr}) \quad (9.16)$$

When we replace the variables with their corresponding values, the equation becomes:

$$Pr = .735 + (-.160)(4.00) + .527(2687.60) + .474(5.00) + (.065)(4) + .011(7) + .080(0) + .322(0) \quad (9.17)$$

$$Pr = 1419.17 \quad (9.18)$$

Our model shows that staging has better performance score. Synthetic application confirms the validity of our model.

9.2.2 Synthetic Application with Eighth Ratio

We have used our synthetic application and synthetic data. Spider is used as an execution node and Painter is used as a data server node (WAN4). This application accesses eighth of the data, so we have used eighth data size. The application reads the database randomly.

9.2.2.1 Experiment Results

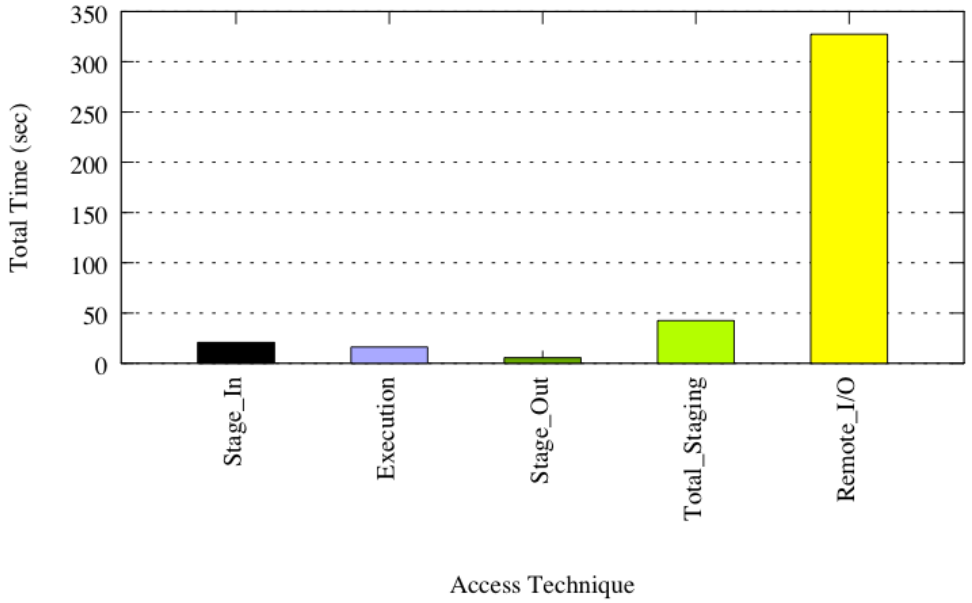

FIGURE 9.4: Synthetic Application Results with Eighth Ratio

According to the Figure 9.4, staging is performing better then remote I/O in our second case synthetic application.

9.2.2.2 Model Results

We applied our models as follows:

Staging

$$Ps = 1.133 + (-.240)(R_{rinL}) + .268(N_{sinL}) + .254(W_{linL}) + (-.009)(W_{loutL}) + (-.157)(R_{loutL}) + (.269)(N_{soutL}) + .290(W_{routL}) + (.051)(D_S) + .003(N_A) + .054(A_{Tj}) + .154(A_{Tr}) \quad (9.19)$$

When we replace the variables with their corresponding values, the equation becomes:

$$Ps = 1.133 + (-.240)(5.03) + .268(13.17) + .254(2.59) + (-.009)(0.53) + (-.157)(0.09) + (.269)(5.3) + .290(0.09) + (.051)(1) + .003(7) + .054(0) + .154(0) \quad (9.20)$$

$$Ps = 5.61 \quad (9.21)$$

Remote I/O

$$Pr = .735 + (-.160)(R_{rinL}) + .527(N_{rL}) + .474(W_{routL}) + (.065)(D_S) + .011(N_A) + .080(A_{Tj}) + .322(A_{Tr}) \quad (9.22)$$

When we replace the variables with their corresponding values, the equation becomes:

$$Pr = .735 + (-.160)(0.56) + .527(309.84) + .474(1.08) + (.065)(1) + .011(7) + .080(0) + .322(0) \quad (9.23)$$

$$Pr = 164.585 \quad (9.24)$$

Our model also shows that staging has better performance score. Synthetic application confirms the validity of our model.

9.3 Extreme Cases

We have used modified versions of our synthetic application to test some extreme use cases as well. We will report two extreme use cases in this section. For each case, we run the simulation for two different data access techniques (sequential and random). On the first case, our simulation reads all data and produces 1/100 output data ratio. On the second case, our simulation reads 1/100 input data ratio and produces the same amount of data.

9.3.1 Full Ratio Input, 1/100 Ratio Output

We have used our synthetic application and synthetic data. Spider is used as an execution node and Quinbee is used as a data server node (MAN). This application accesses all data, so we have used full data size.

9.3.1.1 Experiment Results with Sequential

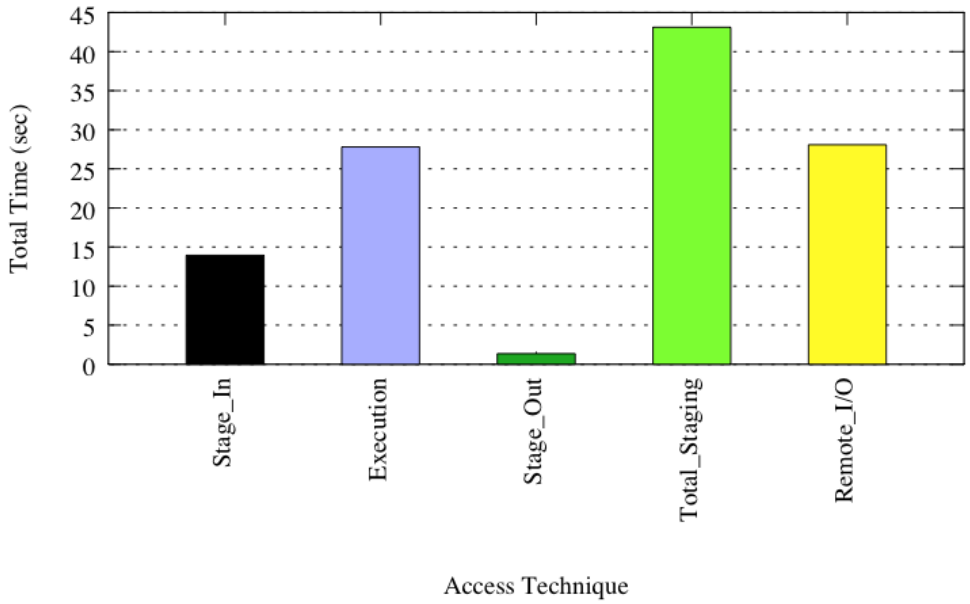

FIGURE 9.5: Extreme Case Synthetic Application Results with Full Input 1/100 Output Sequential

According to the Figure 9.5, remote I/O is performing better then staging in our first extreme case synthetic application.

9.3.1.2 Model Results

We applied our models as follows:

Staging

$$Ps = 1.133 + (-.240)(R_{rinL}) + .268(N_{sinL}) + .254(W_{linL}) + (-.009)(W_{loutL}) + (-.157)(R_{loutL}) + (.269)(N_{soutL}) + .290(W_{routL}) + (.051)(D_S) + .003(N_A) + .054(A_{Tj}) + .154(A_{Tr}) \quad (9.25)$$

When we replace the variables with their corresponding values, the equation becomes:

$$Ps = 1.133 + (-.240)(5.05) + .268(9.56) + .254(2.53) + (-.009)(2.53) + (-.157)(0.86) + (.269)(6.06) + .290(5.85) + (.051)(4) + .003(3) + .054(0) + .154(0) \quad (9.26)$$

$$Ps = 6.50 \quad (9.27)$$

Remote I/O

$$Pr = .735 + (-.160)(R_{rinL}) + .527(N_{rL}) + .474(W_{routL}) + (.065)(D_S) + .011(N_A) + .080(A_{Tj}) + .322(A_{Tr}) \quad (9.28)$$

When we replace the variables with their corresponding values, the equation becomes:

$$Pr = .735 + (-.160)(2.70) + .527(8.18) + .474(1.14) + (.065)(4) + .011(3) + .080(0) + .322(0) \quad (9.29)$$

$$Pr = 5.45 \quad (9.30)$$

Our model shows that remote I/O has better performance score. Synthetic application confirms the validity of our model.

9.3.1.3 Experiment Results with Random

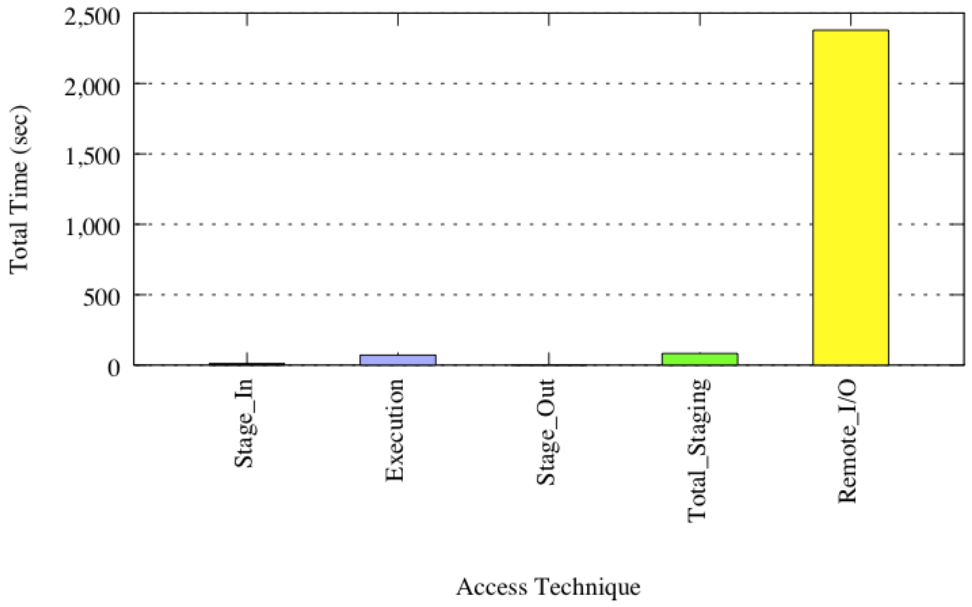

FIGURE 9.6: Extreme Case Synthetic Application Results with Full Input 1/100 Output Random

According to the Figure 9.6, staging is performing better then remote I/O in our first extreme case synthetic application with random data access.

9.3.1.4 Model Results

We applied our models as follows:

Staging

$$Ps = 1.133 + (-.240)(R_{rinL}) + .268(N_{sinL}) + .254(W_{linL}) + (-.009)(W_{loutL}) + (-.157)(R_{loutL}) + (.269)(N_{soutL}) + .290(W_{routL}) + (.051)(D_S) + .003(N_A) + .054(A_{Tj}) + .154(A_{Tr}) \quad (9.31)$$

When we replace the variables with their corresponding values, the equation becomes:

$$Ps = 1.133 + (-.240)(5.05) + .268(8.43) + .254(2.43) + (-.009)(2.53) + (-.157)(0.86) + (.269)(4.58) + .290(5.85) + (.051)(4) + .003(3) + .054(0) + .154(0) \quad (9.32)$$

$$Ps = 5.78 \quad (9.33)$$

Remote I/O

$$Pr = .735 + (-.160)(R_{rinL}) + .527(N_{rL}) + .474(W_{routL}) + (.065)(D_S) + .011(N_A) + .080(A_{Tj}) + .322(A_{Tr}) \quad (9.34)$$

When we replace the variables with their corresponding values, the equation becomes:

$$Pr = .735 + (-.160)(2.70) + .527(2458.03) + .474(1.14) + (.065)(4) + .011(3) + .080(0) + .322(0) \quad (9.35)$$

$$Pr = 1296.52 \quad (9.36)$$

Our model shows that staging has better performance score. Synthetic application confirms the validity of our model.

9.3.1.5 Alternative Model Results with Sequential

We applied our alternative models as follows:

Staging

$$Ps = 1.306 + .014(R_{rinL}) + .144(N_{sinL}) + .819(W_{linL}) + .018(W_{loutL}) + (-.090)R_{loutL} + .098(N_{soutL}) + .084(W_{routL}) + .124(Na_{Can}) + (-.003)Na_{Man} + .157(Na_{Wan1}) + .092(Na_{Wan2}) + 0.92(Na_{Wan3}) + 0.33(Na_{Wan4}) + .029(Na_{Wan6}) + 1.038(Na_{Wan7}) + .056(AT_j) + .153(AT_r) + (-.177)DS_h + (-.295)DS_q + (-.383)DS_e \quad (9.37)$$

When we replace the variables with their corresponding values, the equation becomes:

$$Ps = 1.306 + (.014)(5.05) + .133(9.56) + .819(2.53) + (.018)(2.53) + (-.090)(0.86) + (.098)(6.06) + .084(5.85) + (-.003)(1) \quad (9.38)$$

$$Ps = 5.73 \quad (9.39)$$

Remote I/O

$$Pr = 1.214 + .075(R_{rinL}) + .502(N_{rL}) + .113(W_{routL}) + .017(Na_{Can}) + (-.019)Na_{Man} + (-.094)(Na_{Wan1}) + .092(Na_{Wan2}) + (-.013)(Na_{Wan3}) + (-.094)(Na_{Wan4}) + .112(Na_{Wan5}) + (-.066)(Na_{Wan6}) + .200(Na_{Wan7}) + .083(AT_j) + .390(AT_r) + (-.102)DS_h + (-.177)DS_q + (-.269)DS_e \quad (9.40)$$

When we replace the variables with their corresponding values, the equation becomes:

$$Pr = 1.214 + (.075)(2.70) + .502(8.18) + .113(1.14) + (1.19)(1) + .390(1) + (-.269)(1) \quad (9.41)$$

$$Pr = 4.54 \quad (9.42)$$

Our model shows that remote I/O has better performance score. Synthetic application confirms the validity of our model.

9.3.1.6 Alternative Model Results with Random

We applied our models as follows:

Staging

$$Ps = 1.306 + .014(R_{rinL}) + .144(N_{sinL}) + .819(W_{linL}) + .018(W_{loutL}) + (-.090)R_{loutL} + .098(N_{soutL}) + .084(W_{routL}) + .124(Na_{Can}) + (-.003)Na_{Man} + .157(Na_{Wan1}) + .092(Na_{Wan2}) + 0.92(Na_{Wan3}) + 0.33(Na_{Wan4}) + .029(Na_{Wan6}) + 1.038(Na_{Wan7}) + .056(AT_j) + .153(AT_r) + (-.177)DS_h + (-.295)DS_q + (-.383)DS_e \quad (9.43)$$

When we replace the variables with their corresponding values, the equation becomes:

$$Ps = 1.306 + .014(5.05) + .144(8.43) + .819(2.43) + .018(2.53) + (-.090)(0.86) + (.098)(4.58) + .084(5.85) + (-.003)(1) \quad (9.44)$$

$$Ps = 5.59 \quad (9.45)$$

Remote I/O

$$Pr = 1.214 + .075(R_{rinL}) + .502(N_{rL}) + .113(W_{routL}) + .017(Na_{Can}) + (-.019)Na_{Man} + (-.094)(Na_{Wan1}) + .092(Na_{Wan2}) + (-.013)(Na_{Wan3}) + (-.094)(Na_{Wan4}) + .112(Na_{Wan5}) + (-.066)(Na_{Wan6}) + .200(Na_{Wan7}) + .083(AT_j) + .390(AT_r) + (-.102)DS_h + (-.177)DS_q + (-.269)DS_e \quad (9.46)$$

When we replace the variables with their corresponding values, the equation becomes:

$$Pr = 1.214 + .075(2.70) + .502(2458.03) + .113(1.14) + (-.019)(1) + .390(1) + (-.269)(1) \quad (9.47)$$

$$Pr = 1234.36 \tag{9.48}$$

Our model shows that staging has better performance score. Synthetic application confirms the validity of our model.

9.3.2 1/100 Ratio Input, 1/100 Ratio Output

We have used our synthetic application and synthetic data. Spider is used as an execution node and Quinbee is used as a data server node (MAN). This application accesses 1/100 data, so we have used eighth data size.

9.3.2.1 Experiment Results with Sequential

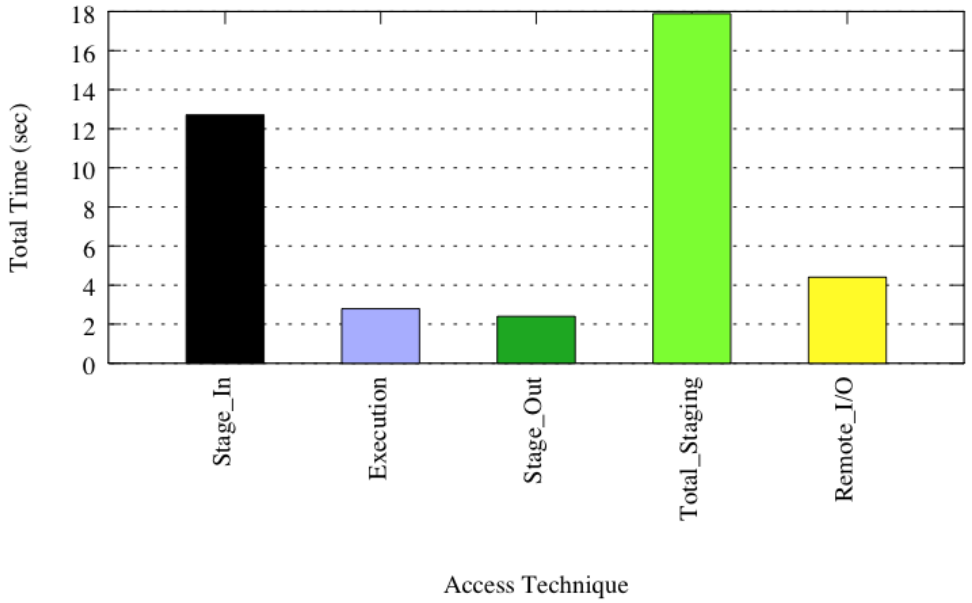

FIGURE 9.7: Extreme Case Synthetic Application Results with 1/100 Input 1/100 Output Sequential

According to the Figure 9.7, remote I/O is performing better then staging.

9.3.2.2 Model Results

We applied our models as follows:

Staging

$$Ps = 1.133 + (-.240)(R_{rinL}) + .268(N_{sinL}) + .254(W_{linL}) + (-.009)(W_{loutL}) + (-.157)(R_{loutL}) + (.269)(N_{soutL}) + .290(W_{routL}) + (.051)(D_S) + .003(N_A) + .054(A_{Tj}) + .154(A_{Tr}) \quad (9.49)$$

When we replace the variables with their corresponding values, the equation becomes:

$$Ps = 1.133 + (-.240)(5.05) + .268(7.52) + .254(2.53) + (-.009)(0.56) + (-.157)(0.10) + (.269)(3.19) + .290(0.09) + (.051)(1) + .003(3) + .054(0) + .154(0) \quad (9.50)$$

$$Ps = 3.50 \quad (9.51)$$

Remote I/O

$$Pr = .735 + (-.160)(R_{rinL}) + .527(N_{rL}) + .474(W_{routL}) + (.065)(D_S) + .011(N_A) + .080(A_{Tj}) + .322(A_{Tr}) \quad (9.52)$$

When we replace the variables with their corresponding values, the equation becomes:

$$Pr = .735 + (-.160)(0.56) + .527(2.36) + .474(1.11) + (.065)(1) + .011(3) + .080(0) + .322(0) \quad (9.53)$$

$$Pr = 2.51 \quad (9.54)$$

Our model shows that remote I/O has better performance score. Synthetic application confirms the validity of our model.

9.3.2.3 Experiment Results with Random

According to the Figure 9.8, staging is performing better then remote I/O with random data access.

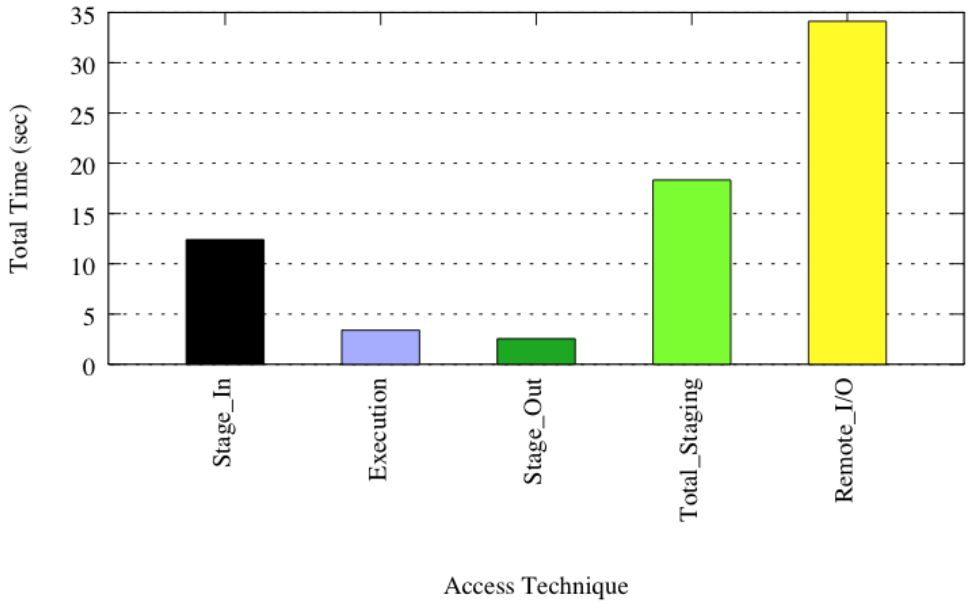

FIGURE 9.8: Extreme Case Synthetic Application Results with 1/100 Input 1/100 Output Random

9.3.2.4 Model Results

We applied our models as follows:

Staging

$$Ps = 1.133 + (-.240)(R_{rinL}) + .268(N_{sinL}) + .254(W_{linL}) + (-.009)(W_{loutL}) + (-.157)(R_{loutL}) + (.269)(N_{soutL}) + .290(W_{routL}) + (.051)(D_S) + .003(N_A) + .054(A_{Tj}) + .154(A_{Tr}) \quad (9.55)$$

When we replace the variables with their corresponding values, the equation becomes:

$$Ps = 1.133 + (-.240)(5.05) + .268(9.81) + .254(2.53) + (-.009)(0.56) + (-.157)(0.10) + (.269)(3.26) + .290(0.09) + (.051)(1) + .003(3) + .054(0) + .154(0) \quad (9.56)$$

$$Ps = 4.13 \quad (9.57)$$

Remote I/O

$$Pr = .735+(-.160)(R_{rinL})+.527(N_{rL})+.474(W_{routL})+(.065)(D_S)+.011(N_A)+.080(A_{Tj})+.322(A_{Tr}) \tag{9.58}$$

When we replace the variables with their corresponding values, the equation becomes:

$$Pr = .735+(-.160)(0.56)+.527(305.88)+.474(1.11)+(.065)(1)+.011(3)+.080(0)+.322(0) \tag{9.59}$$

$$Pr = 162.50 \tag{9.60}$$

Our model shows that staging has better performance score. Synthetic application confirms the validity of our model.

9.3.2.5 Alternative Model Results with Sequential

We applied our alternative models as follows:

Staging

$$\begin{aligned} Ps = 1.306 + .014(R_{rinL}) + .144(N_{sinL}) + .819(W_{linL}) + .018(W_{loutL}) + (-.090)R_{loutL} + \\ .098(N_{soutL}) + .084(W_{routL}) + .124(Na_{Can}) + (-.003)Na_{Man} + .157(Na_{Wan1}) + \\ .092(Na_{Wan2}) + 0.92(Na_{Wan3}) + 0.33(Na_{Wan4}) + .029(Na_{Wan6}) + 1.038(Na_{Wan7}) + \\ .056(AT_j) + .153(AT_r) + (-.177)DS_h + (-.295)DS_q + (-.383)DS_e \end{aligned} \tag{9.61}$$

When we replace the variables with their corresponding values, the equation becomes:

$$\begin{aligned} Ps = 1.306 + (.014)(5.05) + .133(7.52) + .819(2.53) + (.018)(0.56) + (-.090)(0.10) + \\ (.098)(3.19) + .084(0.09) + (-.003)(1) + (-.383)(1) \end{aligned} \tag{9.62}$$

$$Ps = 4.40 \tag{9.63}$$

Remote I/O

$$Pr = 1.214 + .075(R_{rinL}) + .502(N_{rL}) + .113(W_{routL}) + .017(Na_{Can}) + (-.019)Na_{Man} + (-.094)(Na_{Wan1}) + .092(Na_{Wan2}) + (-.013)(Na_{Wan3}) + (-.094)(Na_{Wan4}) + .112(Na_{Wan5}) + (-.066)(Na_{Wan6}) + .200(Na_{Wan7}) + .083(AT_j) + .390(AT_r) + (-.102)DS_h + (-.177)DS_q + (-.269)DS_e \quad (9.64)$$

When we replace the variables with their corresponding values, the equation becomes:

$$Pr = 1.214 + (.075)(0.56) + .502(2.36) + .113(1.11) + (0.019)(1) + .390(1) + (-.269)(1) \quad (9.65)$$

$$Pr = 1.45 \quad (9.66)$$

Our model shows that remote I/O has better performance score. Synthetic application confirms the validity of our model.

9.3.2.6 Alternative Model Results with Random

We applied our models as follows:

Staging

$$Ps = 1.306 + .014(R_{rinL}) + .144(N_{sinL}) + .819(W_{linL}) + .018(W_{loutL}) + (-.090)R_{loutL} + .098(N_{soutL}) + .084(W_{routL}) + .124(Na_{Can}) + (-.003)Na_{Man} + .157(Na_{Wan1}) + .092(Na_{Wan2}) + 0.92(Na_{Wan3}) + 0.33(Na_{Wan4}) + .029(Na_{Wan6}) + 1.038(Na_{Wan7}) + .056(AT_j) + .153(AT_r) + (-.177)DS_h + (-.295)DS_q + (-.383)DS_e \quad (9.67)$$

When we replace the variables with their corresponding values, the equation becomes:

$$Ps = 1.306 + .014(5.05) + .144(9.81) + .819(2.53) + .018(0.56) + (-.090)(0.10) + (.098)(3.26) + .084(0.09) + (-.003)(1) + .153(1) + (-.383)(1) \quad (9.68)$$

$$Ps = 4.95 \tag{9.69}$$

Remote I/O

$$\begin{aligned} Pr = 1.214 + .075(R_{rinL}) + .502(N_{rL}) + .113(W_{routL}) + .017(Na_{Can}) + (-.019)Na_{Man} + \\ (-.094)(Na_{Wan1}) + .092(Na_{Wan2}) + (-.013)(Na_{Wan3}) + (-.094)(Na_{Wan4}) + \\ .112(Na_{Wan5}) + (-.066)(Na_{Wan6}) + .200(Na_{Wan7}) + .083(AT_j) + .390(AT_r) + \\ (-.102)DS_h + (-.177)DS_q + (-.269)DS_e \end{aligned} \tag{9.70}$$

When we replace the variables with their corresponding values, the equation becomes:

$$Pr = 1.214 + .075(0.56) + .502(305.88) + .113(1.11) + (-.019)(1) + .390(1) + (-.269)(1) \tag{9.71}$$

$$Pr = 153.82 \tag{9.72}$$

Our model shows that staging has better performance score. Synthetic application confirms the validity of our model.

Chapter 10
Conclusions

Over the years, scientific applications and experiments have become increasingly complex and more demanding in terms of the ir computational and data requirements, and the amount of data generated and used has grown at a very rapid rate. One of the major challenges for these applications in distributed computing setting has been how to access large datasets remotely over the network. Data staging and remote I/O have been the most widely used data access methods for distributed applications. Application developers generally chose one over the other intuitively without making any scientific comparison specific to their applications since there is no generic model available that they can use.

In this thesis, we have developed generic models and set guidelines for the application developers which would help them to choose the most appropriate data access method for their application. We defined the parameters that potentially affect the end-to-end performance of the distributed applications which need to access remote data. We have implemented a series of synthetic benchmark applications to simulate different data access patterns. We run these benchmark applications on different distributed computing settings with different parameters, such as network bandwidth, server and client capabilities, and data access ratio. We have also used different remote I/O protocols to show the importance of the protocol in making a decision. We have used regression analysis to develop applicable generic models for comparing different data access methods, and test our models in a real life application.

The main contribution of our thesis is generic models that can be applied to most data-intensive distributed applications to decide the best data access technique for those applications. Our models provide the scientists and application developers an opportunity to choose the best data access method before actually running the application. Since data-intensive distributed applications spend most of the time accessing the data before and after the computation, choosing the best way to access

the remote data is imperative. Application designers can use our models to develop their application when they decide which data access technique is right for them.

The existing approaches to find the best data access method has been based on active learning. First, the application needs to be run in the same environment with all possible combinations. Once the combination that is the best fit for that environment is discovered, the correct data access technique can be found. Our models, however, provide best data access technique before running the application.

Sequential and jump data access always performs better then random data access on all network architectures. High-speed networks improve the data transfer ratio dramatically. Random data access on remote I/O is not benefitting from high-speed network architectures, so the performance decreases dramatically. The differences on sequential and jump is less than the differences on random. So, researcher should be more careful when remote I/O was chosen. Since all data should be staged in before the execution, decreasing the data ratio improves the remote I/O performance.

High performance computers not only increase the execution performance, but also the data transfer performance. Network overhead plays important role on random remote I/O. If the application can use advanced programming techniques to handle network overhead, remote random performance can be improved.

Increasing the distance between execution node and data server node with high-speed network decrease the the gap between sequential and jump performance. Without high-speed network, it decreases the gap between remote I/O and staging on all data access techniques. On the other hand, it increases the gap on smaller data ratios. Also, choosing proper remote I/O protocol is a crucial decision for the end-to-end application performance.

References

[1] http://adcirc.org/.

[2] http://freshmeat.net/projects/unixbench.

[3] http://gking.harvard.edu/clarify/docs/clarify.html.

[4] http://oprofile.sourceforge.net/about/.

[5] http://www.hpss-collaboration.org/.

[6] http://www.netperf.org/netperf/.

[7] http://www.postgresql.org/.

[8] http://www.spec.org.

[9] ulm. ftp://ds.internic.net/internet-drafts/draft-abela-ulm-02.txt.

[10] http://foldoc.org/uml, 2002.

[11] http://www.loni.org/, 2009.

[12] http://aimbench.sourceforge.net/, 2010.

[13] https://gilda.ct.infn.it/, 2010.

[14] http://sourceforge.net/projects/hdparm/, 2010.

[15] http://wiki.lustre.org/index.php/, 2010.

[16] http://www.coda.cs.cmu.edu/, 2010.

[17] http://www.cse.nd.edu/ ccl/software/, 2010.

[18] http://www.iozone.org/, 2010.

[19] http://www.lsc-group.phys.uwm.edu/ldr/, 2010.

[20] http://www.nasa.gov/, 2010.

[21] http://www.pcausa.com/utilities/pcattcp.htm, 2010.

[22] http://www.petashare.org/, 2010.

[23] http://www.teragrid.org, 2010.

[24] www.spss.com, 2010.

[25] A. D. Alexandrov, M. Ibel, K. E. Schauser, and C. J. Scheiman. Extending the operating system at the user level: The ufo global file system. In *In 1997 Annual Technical Conference on UNIX and Advanced Computing Systems (USENIX'97)*, January 1997.

[26] N. Ali and M. Lauria. Improving the performance of remote i/o using asynchronous primitives. In *HPDC-15, Paris*, June 2006.

[27] W. Allcock. *Gridftp protocol specification.* Global grid forum, 2003. GFD.20.

[28] Thomas E. Anderson, Michael D. Dahlin, Jeanna M. Neefe, David A. Patterson, Drew S. Roselli, and Randolph Y. Wang. Serverless network file systems. *ACM Trans. Comput. Syst.*, 14(1):41–79, 1996.

[29] P. M. Andrews, T. Sherwin, and B. Banister. A centralized data access model for grid computing. In *Proceedings on 20th IEEE/11th NASA Goddard Conference*, 2003.

[30] L. G. Antoniu and M. Jan BougB. Juxmem: An adaptive supportive platform for data sharing on the grid. In *Scalable Computing: Practice and Ezperience*, 2005.

[31] S. Babu, P. Shivam, and J. Chase. Active and accelerated learning of cost models for optimizing scientific applications. In *International Conferance on Very Large Data Bases (VLDB)*, 2006.

[32] R. Bennett, K. Bryant, A. Sussman, R. Das, and J. Saltz. Jovian: A framework for optimizing parallel i/o. In *In Proceedings of the 1994 Scalable Parallel Libraries Conference*, pages 10–20. IEEE Computer Society Press, October 1994.

[33] J. Bent, V. Venkataramani, N. LeRoy, A. Roy, J. Stanley, A.C. Arpaci-Dusseau, R.H. Arpaci-Dusseau, and M. Livny. Flexibility, manageability, and performance in a grid storage appliance. In *High Performance Distributed Computing, 2002. HPDC-11 2002. Proceedings. 11th IEEE International Symposium on*, pages 3–12, 2002.

[34] John Bent, Douglas Thain, Andrea Arpaci-Dusseau, Remzi Arpaci-Dusseau, and Miron Livny. Explicit control in a batch aware distributed file system. In *Proceedings of the First USENIX/ACM Conference on Networked Systems Design and Implementation, San Francisco, CA*, March 2004.

[35] Joseph Bester, Ian Foster, Carl Kesselman, Jean Tedesco, and Steven Tuecke. Gass: a data movement and access service for wide area computing systems. In *IOPADS '99: Proceedings of the sixth workshop on I/O in parallel and distributed systems*, pages 78–88, New York, NY, USA, 1999. ACM.

[36] M. D. Beynony, R. Ferreiray, T. Kurcy, A. Sussmany, and J. Saltzyz. Datacutter: Middleware for filtering very large scientific datasets on archival storage systems. In *In Proceedings of the 2000 Mass Storage Systems Conferance*. IEEE Computer Society Press.

[37] Ram Sri Harsha Bhagawaty, Lei Jiang, Swathi Dubbaka, Kelin Hu, Sreekanth Pothanis, Gabrielle Allen, Nathan Brener, Qin Chen, and Tevfik Kosar. Design, implementation and use of a simulation data archive for coastel science. In *CLADE*, 2010.

[38] C. Blanchet, R. Mollon., D. Thain, and G. Deleage. Grid deployment of lagesy bioinformatics applications with transparent data access. In *IEEE Conference on Grid Computing*, September 2006.

[39] M. Buddhikot, G. Parulkar, and J. Cox. Design of a large scale multimedia storage server. In *INET '94*, 1994.

[40] Alok Choudhary, Rajesh Bordawekar, Michael Harry, Rakesh Krishnaiyer, Ravi Ponnusamy, Tarvinder Singh, and Rajeev Thakur. Passion: Parallel and scalable software for input-output. Technical report, 1994.

[41] P. Corbett, D. Feitelson, Y. Hsu, J.-P. Prost, M. Snir, S. Fineberg, B. Nitzberg, B. Traversat, and P. Wong. Mpi-io: A parallel file i/o interface for mpi. Technical Report NAS-95-002, NAS, NASA Ames Research Center, Mofiett Field, CA, January 1995. Version 0.3.

[42] P. Corbett, D. Feitelson, J.-P. Prost, and S. Baylor. Overview of the vesta parallel file system. In *In IPPS '93 Workshop on Input/Output in Parallel Computer Systems*, pages 1–16, 1993.

[43] W. Elwasif, J. Plank, and R. Wolski. Data staging effects in wide area task farming applications. In *Proceedings of the 1st International Symposium on Cluster Computing and the Grid*, 2001.

[44] Renato Figueiredo, Nirav H. Kapadia, and Jose A. B. Fortes. The punch virtual file system: Seamless access to decentralized storage services in a computational grid. In *In Proceedings of the Tenth IEEE International Symposium on High Performance Distributed Computing. IEEE Computer*. Society Press, 2001.

[45] I. Foster, D. Kohr, R. Krishnaiyer, and J. Mogill. Remote i/o: Fast access to distant storage. In *Workshop on I/O in Parallel and Distributed Systems (IOPADS)*, pages 14–25, 1997.

[46] R. Fujimoto. Parallel and distributed simulation systems. *Wiley Interscience, New York*, 2000.

[47] Tabachnick Barbara G. and Fidell Linda S. *Using multivariate statistics*. Pearson Education Inc., 2007.

[48] William E. Johnston, Dennis Gannon, and Bill Nitzberg. Grids as production computing environments: The engineering aspects of nasa's information power grid. *High-Performance Distributed Computing, International Symposium on*, 0:34, 1999.

[49] Jones and James Patton. Pbs: portable batch system. pages 369–390, 2002.

[50] R. Kalmady and B. Tierney. A comparison of gsiftp and rfio on a wan. In *Proceedings of Computers in High Energy Physics*, 2001.

[51] David G. Kleinbaum. *Applied Regression Analysis and Other Multivariable Methods*. Duxbury Press, 1998.

[52] D. Kohr, R. Krishnaiyer, I. Foster, and J. Mogill. Remote i/o: Fast access to distant storage. In *Workshop on I/O in Parallel and Distributed Systems (IOPADS)*, pages 14–25, 1997.

[53] Tevfik Kosar and Miron Livny. Stork: Making data placement a first class citizen in the grid. In *In International Conference on Distributed Computing Systems*, March 2004.

[54] G. Krasner and S. Pope. A cookbook for using model view controller interface paradigm. *Journal of Object Oriented Programming*, 1988.

[55] Erwin Laure, Heinz Stockinger, and Kurt Stockinger. Performance engineering in data grids. *Journal of Concurrency and Computation: Practice and Experience, Wiley Press*, 17(2-4), 2005.

[56] Jonghyun Lee, Xiaosong Ma, Robert Ross, Rajeev Thakur, and Marianne Winslett. Rfs: Efficient and flexible remote file access for mpi-io. In *In Proc. of the IEEE Int'l Conference on Cluster Computing (Cluster 2004)*, September 2004.

[57] Jonghyun Lee, Robert Ross, Scott Atchley, Micah Beck, and Rajeev Thakur. Mpi-io/l: Efficient remote i/o for mpi-io via logistical networking. In *In Proc. of the 20th IEEE Int'l Parallel and Distributed Processing Symposium (IPDPS 2006)*, April 2006.

[58] M. Litzkow, M. Livney, and M. Mutka. Condor – a hunter of idle workstations. In *In Proc. 8th Intl Conf. on Distributed Computing Systems*, pages 104–111, 1988.

[59] Ravi K Madduri, Cynthia S. Hood, and William E. Allcock. Reliable file transfer in grid environments. In *Proceedings of the 27th Annual IEEE Conference on Local Computer Networks (LCN'02)*, 2002.

[60] David Mazières, Michael Kaminsky, M. Frans Kaashoek, and Emmett Witchel. Separating key management from file system security. *SIGOPS Oper. Syst. Rev.*, 33(5):124–139, 1999.

[61] D. Mills. Simple network time protocol (ntp). *RFC 1769, University of Delaware*, March 1995.

[62] J. Morris, M. Satyanarayanan, M. Conner, J. Howard, D. Rosenthal, and F. Smith. Andrew: A distributed personal computing environment. *Communications of the ACM*, 29(3):184–201, 1986.

[63] Nils Nieuwejaar and David Kotz. The galley parallel file system. *Parallel Computing*, pages 23(4):447–476, June 1997.

[64] Marija J. Norusis. *SPSS 16.0 Guide to Data Analysis*. prentice hall Inc., 2008.

[65] María S. Pérez, Jesús Carretero, Félix García, José M. Peña, and Víctor Robles. Mapfs: A flexible multiagent parallel file system for clusters. *Future Generation Computer Systems*, 22(5):620 – 632, 2006.

[66] J. Plank, M. Beck, W. R. Elwasif, and T. Moore. The internet backplane protocol: Storage in the network. In *In Proceedings of the 1999 Network Storage Symposium NetStore99, Seattle, WA, USA*, 1999.

[67] S. Pllana and T. Fahringer. Performance prophet: A performance modeling and prediction tool for parallel and distributed programs. In *The 2005 International Conference on Parallel Processing (ICPP-05)*, 2005.

[68] J. Postel and J. Reynolds. *File Transfer Protocol (ftp)*, 1985.

[69] Radu Prodan and Thomas Fahringer. Zenturio: An experiment management system for cluster and grid computing. In *In Proceedings of the 4th International Conference on Cluster Computing (CLUSTER 2002*, pages 9–18. IEEE Computer Society Press. http://www.par.univie.ac.at/project/zenturio, 2002.

[70] Buyya R. and M. Murshed. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *J. Concur, and Computation (CCPE)*, 2002.

[71] Manuel Rodríguez-Martínez and Nick Roussopoulos. Mocha: a self-extensible database middleware system for distributed data sources. *SIGMOD Rec.*, 29(2):213–224, 2000.

[72] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, and B. Lyon. Design and implementation of the sun network file system. In *In Proc. Summer USENIX*, pages pages 119 – 130, June 1985.

[73] K. E. Seamons, Y. Chen, P. Jones, J. Jozwiak, and M. Winslett. Server-directed collective i/o in panda. In *Supercomputing '95: Proceedings of the 1995 ACM/IEEE conference on Supercomputing (CDROM)*, page 57, New York, NY, USA, 1995. ACM.

[74] X. Shen, W. Liao, and A. Choudhary. Remote i/o optimization and evaluation for tertiary storage systems through storage recource vroker. In *In Proc. of IASTED Applied Informatics, Innsbruck, Austria*, 2001.

[75] Xiaohui Shen, Wei keng Liao, and Alok Choudhary. Remote i/o optimization and evaluation for tertiary storage systems through storage resource broker. *IASTED Applied Informatics, Innsbruck, Austria*, 2001.

[76] Heinz Stockinger. Data management in data grids - habilitation overview. Technical report, Research Lab for Computational Technologies and Applications, May 2005.

[77] Heinz Stockinger, Kurt Stockinger, Erich Schikuta, and Ian Willers. Towards a cost model for distributed and replicated data stores. In *9th Euromicro Workshop on Parallel and Distributed Processing (PDP 2001), IEEE Computer Society Press, Mantova, Italy*, February, 2001.

[78] N. T. B. Stone, D. Balog, B. Gill, B. Johanson, J. Marsteller, P. Nowoczynski, D. Porter, and R. Reddy. Pdio: High-performance remote file i/o for portals enabled compute nodes. In *In Proceedings of the 2006 Conference on Parallel and Distributed Processing Techniques and Applications, Las Vegas, NV*, June 2006.

[79] D. Thain. Chirp: An architecture for cooperative storage. *Workshop on Adaptive Grid Middleware*, 2005.

[80] D. Thain, J. Bent, A. Arpaci-Dusseau, R. Arpaci-Dusseau, and M. Livny. Gathering at the well: Creating communities for grid i/o. In *International Conference for High Performance Computing and Communications*, 2001.

[81] D. Thain and M. Livny. Parrot: Transparent user-level middleware for data-intensive computing. *Univ. of Notre Dame, Computer Science and Engineering Dept.*, 2003.

[82] R. Thakur, W Gropp, and E. Lusk. An abstract-device interface for implementing portable parallel-i/o intervaces. In *In Proceeding of the Symposium on the Frontiers of Massively Parallel Computation*, 1996.

[83] R. Thakur, W. Gropp, and E. Lusk. Data sieving and collective i/o in romio. In *In proceeding of the Symposium on the Frontiers of Massively Parallel Computation*, 1999.

[84] Mitchell D. Theys, Howard Jay Siegel, Noah B. Beck, Min Ta, and Michael Jurczyk. A mathematical model, heuristic, and simulation study for a basic data staging problem in a heterogeneous networking environment. In *Proceedings of the Seventh Heterogeneous Computing Workshop*, 1998.

[85] B. Tierney, W. Johnston, L. Chen, H. Herzog, G. Hoo, G. Jin, and J. Lee. Distributed parallel data storage systems: A scalable approach to high speed image servers. In *ACM Multimedia 94*. ACM Press, 1994.

[86] Brian Tierney, William Johnston, Brian Cowley, Gary Hoo, Chris Brooks, and Dan Gunter. The netlogger methodology for high performance distributed systems performance analysis. In *In Proc. 7th IEEE Symp. on High Performance Distributed Computing*, pages 260–267, 1998.

[87] R. Tite. General linear model applications, 1996.

[88] R. L. Tweedie. *Operator-Geometric Stationary Distributions for Markov Chains, with Application to Queueing Models*, volume 14-2. Applied Probability Trust, 1982.

[89] A. Vahdat, P. Eastham, and T. Anderson. Webfs: A global cache coherent flesystem. Technical report, Department of Computer Science UC Berkeley, 1996.

[90] Y. Wang and D. Kaeli. Load balancing using grid-based peer-to-peer parallel i/o. In *Proceedings of the IEEE Cluster Computing Conference*, September 2005.

[91] B. White, A. Grimshaw, and A. Nguyen-Tuong. Grid-based file access: The legion I/O model. In *Proceedings of the 9th IEEE Symposium on High Performance Distributed Systems*, 2000.

[92] Brian S. White, Andrew S. Grimshaw, and Anh Nguyen-Tuong. Grid-based file access: The legion i/o model. In *HPDC '00: Proceedings of the 9th IEEE International Symposium on High Performance Distributed Computing*, page 165, Washington, DC, USA, 2000. IEEE Computer Society.

Vita

The author, Ibrahim H Suslu, was born in Trabzon, Turkey, in 1969. The author enrolled at Marmara University, Technical Education Faculty, in 1988 and received a bachelor's degree in computer and control in 1993. During the college education, he had been honored with the scholarship from the Istanbul Commerce Organization for two years. After the graduation, he has worked as a graduate assistant at Marmara University, Technical Educational Faculty for 6 years.

He received his first master's degree in 1995 in the field of computer technologies. His area of study was Intel 80196 Micro Controller Simulation in Education. He had been in the Ph.D. program of the same university until 1999 for three years. He fulfilled all the course requirements and was working on his thesis. He had been employed as a teaching assistant between 1993 and 1999.

The author enrolled in the Science and Math Education (SMED) Ph.D. Program at Southern University, Baton Rouge in January, 2000. Then he switched his program from the SMED to the Computer Science Master program on 2001. He has worked as a graduate assistant at Southern University, Baton Rouge, from January 2000 to fall 2001. He has worked for a joint data mining project between Southern University (SUBR) and University of Louisiana of Lafayette (ULL) under Dr. Miroslav Kubat. He has finished his master's on fall 2001.

The author enrolled in the computer science doctoral program at Louisiana State University in 2002. While he was working for his PhD, he also worked at the Southern University computer science department as a system manager for 5 years. To fulfill residency requirement, he resigned from his system manager job and became a full time student and started to work as a graduate assistant at CCT under Dr. Tevfik Kosar.

He has contributed to four research projects, participated in five different conferences and three workshops. He has presented a paper at four different conferences.

His research is in distributed computing with emphasis on remote data access techniques. He is also interesting in networks, operating systems and data mining.

His most significant publications are:

- Tevfik Kosar, Mehmet Balman, Ibrahim H Suslu, Esma Yildirim, and D. Yin. Data-Aware Distributed Computing, Submitted to the Journal of Parallel and Distributed Computing (JPDC), 2010.

- Tevfik Kosar, Mehmet Balman, Ibrahim H Suslu, Esma Yildirim, and D. Yin Data-Aware Distributed Computing with Stork Data Scheduler, In Proceedings of SEE-GRID-SCI'09, Istanbul, Turkey, December 2009

- Ibrahim H Suslu, Wan Huang and Tevfik Kosar, Choosing Between Remote I/O versus Staging in Large Scale Distributed Applications, Extended Abstract and Poster in Louisiana EPSCoR RII Cybertools/Science Drivers Symposium, Baton Rouge, LA, USA, May, 2009

- Ibrahim H Suslu, Fatih Turkmen, Mehmet Balman, Tevfik Kosar Choosing Between Remote I/O versus Staging in Large Scale Distributed Applications, in Parallel and Distributed Computing and Communication Systems, New Orleans, LA, 2008.

- Esma Yildirim, Ibrahim H Suslu, Tevfik Kosar. Which Network Measurement Tool is Right for You? A Multidimensional Comparison Study, in The 9th IEEE/ACM International Conference on Grid Computing (Grid 2008), Tsukuba, Japan, 2008

- Ibrahim H Suslu, Ismail Akturk, Xingqi Wang and Tevfik Kosar. Distributed Data Management in CyberTools, Poster in Louisiana Cyberinfrastructure and Science

- Mehmet Balman, Ibrahim Suslu, and Tevfik Kosar. Distributed Data Management with PetaShare, Poster in the 15th Mardi Gras Conference, Baton Rouge, LA, USA, February, 2008

- Ibrahim H Suslu and Tevfik Kosar. Balancing the use of Remote I/O versus Staging in Distributed Environments In 9th International Conference on Enterprise Information System, (ICEIS'07), Funchal, Madeira - Portugal, June, 2007

- Osman Kandara, Mehtap Kandara, and Ibrahim H Suslu. Association Rules to Predict the Likelihood of Recoverability of an Attribute in a Database, In the 8th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2004), Orlando, USA, July, 2004