University of Nevada, Reno

Facilitating Data Driven Research Through a Hardware- and Software-Based Cyberinfrastructure Architecture

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science and Engineering

by

Brianna Blain-Castelli

Dr. Frederick C. Harris Jr., Co-Advisor
Dr. Sergiu M. Dascalu. Co-Advisor

May, 2022

THE GRADUATE SCHOOL

We recommend that the thesis
prepared under our supervision by

Brianna M. Blain-Castelli

Entitled

Facilitating Data Driven Research Through a Hardware- and Software-Based Cyberinfrastructure Architecture
be accepted in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Dr. Frederick C. Harris, Jr., Co-Advisor

Dr. Sergiu M. Dascalu, Co-Advisor

Dr. Scotty Strachan, Graduate School Representative

David W. Zeh, Ph.D., Dean, Graduate School

May, 2022

Abstract

Cyberinfrastructure is the backbone of research and modern industry. As such, to have an environment conducive to research advancements, cyberinfrastructure must be well maintained and accessible by all researchers. Presented in this thesis is a method of centralizing aspects of cyberinfrastructure to allow for ease of collaboration and data management by researchers without requiring these researchers to manage the involved systems themselves. This centralized architecture includes dedicated machines for data transfers, a cluster designed to run microservices surrounding the method, a dashboard for performance and health monitoring, and network telemetry collection. As system administrators are responsible for maintaining the systems in place, a user study was conducted to assess the functionality of the dashboard they would utilize to receive alerts from and utilize to quickly gauge the status of involved hardware. This thesis aims to provide a template for deploying centralized data transfer cyberinfrastructure and a manual for utilizing these systems to support data driven research.

Dedication

I dedicate this thesis to my family, whose love and support was invaluable on this journey.

Acknowledgments

I would like to thank my committee Dr. Harris, Dr. Dascalu, and Dr. Strachan for their invaluable guidance throughout my graduate studies. They have all been incredible mentors who have helped me grow as a professional.

I would also like to thank NSHE Systems Computing Services and NevadaNet personnel, including Tom Fishel and James Hayes for their support. Next, I want to thank the University of Nevada, Reno's CIO Steve Smith and David W. Zeh from the Graduate School for helping support my Graduate Research Assistant position in the Cyberinfrastructure group. This has been an incredible opportunity to work within the Office of Information Technology that I hope continues moving forward for other Graduate students. I would also like to thank members of the Office of Information Technology, including Zachary Newell, Cole Griggs, and Brice Berman, for their guidance and assistance. I sincerely thank you for your hand in aiding me to become a better professional and student while developing new skills on the job. Finally, I would like to thank Sebastian Smith, who provided further professional development opportunities.

I would also like to thank my colleagues Chase Carthen, for his invaluable assistance throughout the development process as well as Vinh Le, Connor Scully-Allison, and Andrew Muñoz for their guidance. Additionally, I would like to thank my boyfriend, Mitchell Martinez, both for his encouragement and his aid in software development. Finally, I'd like to thank my family and my dog, Kovu, for their emotional support throughout my Masters degree.

This material is based in part upon work supported by the National Science Foundation under grant number(s) IIA-1301726, OAC-1827186, OAC-2019164. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In modern history, humanity is more connected than ever before because of the internet. As of January 2022 there are 4.95 billion internet users worldwide, equating to roughly 62% of the global population represented online, and this number is only growing [40]. This connection is both to one another and to data as supported by the internet traffic of users. The top three most visited sites on the internet are Google, YouTube, and Facebook [74], and out of the top 10 most visited sites of 2021, eight are categorized as either social media or a search engine. And, while YouTube itself is neither social media or a search engine, it hosts both educational content and has a social component in the form of stream chats that allow views to communicate with content creators live. People are seeking both information and social platforms to connect with one another. This trend is also present for academic research. Virtual libraries are available to quickly find and share information from institutions and organizations. Journal and conference papers are available and easily searched for relevant fields. This has resulted in a growing opportunity for collaboration in research across regions.

At the same time, the field of big data has grown exponentially in recent years as more information can be readily collected. The field of big data refers to the collection, analysis, and extraction of useful data in increasingly large and complicated data sets. It is estimated that the average person generates roughly 1.7 MB of data a second and internet users as a whole generate 2500 PB of data every day between phone, computer, and internet of things usage [73]. This data then needs to be parsed and filtered so only useful data is collected. Similar explosive data growth is occurring broadly across fields of scientific and engineering research.

Technology in the research space, known as cyberinfrastructure [66, 78], needs to adapt to support collaboration across institutional and organization campuses while adjusting for the scale of data [36]. For example, many researchers will share data by attaching files to emails or through a collaborative space, such as Google Drive [31]. However, this option is not viable in the longterm in part due to growing file sizes and security. Large numbers of files sent will be lost or unorganized, and file sizes that are too large are not attachable. In these cases, few alternative options are utilized. Of these options, fewer have a minimal learning curve, set-up time, and universal application—required features for technology adoption due to observed researcher hesitancy to dedicate time towards learning new tools [42]. For example, Rsync [69], a popular data transfer solution for Linux, has a simple setup, but requires researchers to learn Linux commands and the process of transferring data between Linux machines and other operating systems is convoluted. This time to learn and deploy a data transfer solution is better spent on further research. Instead, solutions must be intuitive or highly facilitated by technology professionals [52]. Additionally, these platforms need to be available and maintained as reliable institutional-scale infrastructure for general use by the research population.

To address these issues, this thesis proposes the Data Driven Research Infrastructure Systems (DDRIS) as an architecture for centrally deployed and maintained data transfer solutions for researchers developed within the University of Nevada, Reno, Cyberinfrastructure Department. This collection of systems hosts a combination of software solutions for data transfer, defined further in Chapter 2 and Chapter 6, and tools for monitoring system health and performance. Health and performance data is then visualized for system administrators responsible for the performance of these transfer solutions. The current deployed version of DDRIS functions as a proof of concept that will be built upon in further iterations as outlined in future work. An overview of DDRIS can be read in Chapter 3.

The rest of this thesis is structured as follows: Chapter 2 provides background knowledge integral to understanding the material of this work. It includes an introduction to research computing concepts, including the Science DMZ, data transfers, and data transfer nodes. A brief history of high performance computing and its evolution into cluster computing is then provided. This is followed by a description of cloud computing, containerization and the combination of the two. Information is then presented on file systems and technologies utilized in the solution proposed in this thesis. Chapter 3 provides a more in-depth analysis of the problem and the solution covered in this thesis. Chapter 4 describes the hardware involved in the final model. This includes their initialization, changes for performance, and unique configuration requirements. Chapter 5 provides a deeper look at the design and implementation of software, including specifications and design. Chapter 6 presents the front-end applications vital to DDRIS, including a manual on utilizing Globus for data transfers, Prometheus visualizations, MaDDash with PerfSONAR, and the DDRIS cyberinfrastructue dashboard. Chapter 7 includes the details of a user study to determine the effectiveness of DDRIS. Chapter 8. Finally conclusions as well as future work are contained in Chapter 9.

Chapter 2

Background and Related Work

In this Chapter, background knowledge is provided on the concepts surrounding the problem and solution proposed in this thesis. First, an overview of the Science DMZ and its place in Research Cyberinfrastructure is discussed in Section 2.1. Data transfer and the hardware necessary to achieve high-speed performance is outlined in Section 2.2 and Section 2.3 respectively. A brief history of High Performance Computing is provided in Section 2.4 (this method of computing resulted in clusters and cloud computing). Section 2.5 and Section 2.6 then provides more detail on these approaches to computing. Next, the concept of containerization is introduced in Section 2.7. Section 2.8 provides information on the practical application of containerization in clusters and the cloud. The basics of file systems, including their advantages and disadvantages are discussed in Section 2.9. Finally, a list and brief description of technologies used in our solution implementation is provided in Section 2.10.

2.1 Science DMZ

A Science DMZ is a section of a university's network dedicated to high-performance research applications as opposed to enterprise systems. The network model was developed by the Energy Sciences Network for handling the transfer of Big Data based on the existing implementations of a "demilitarized zone" (DMZ) [19] and is continuing to evolve from the perspectives of security, routing, access, and real-time parallel inspection with the development of new technologies around software automation and infrastructure. A traditional DMZ is implemented to increase security for an organization by separating external facing systems, including web servers and mail servers, from the internal local area network (LAN) behind a stateful firewall.

There are two types of traditional DMZs, single and dual firewall. In a single firewall environment, the DMZ and institutional LAN share an internal firewall but remain separate through routing. This has higher performance than the dual firewall implementation shown in Figure 2.1. In this model, external traffic is first policed by a border firewall. If traffic is not blocked by the firewall, it then has access to the DMZ. However, a second, internal firewall blocks this traffic from reaching the LAN. Internal traffic must traverse this internal firewall and the border firewall to reach the external wide area network (WAN). This approach has higher security than the single firewall implementation.

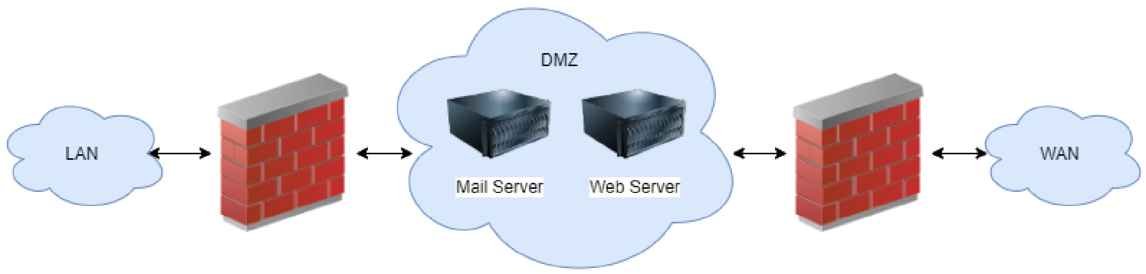


Figure 2.1: A dual firewall implementation of a DMZ separates external facing systems from the LAN through a second firewall, creating a network traffic pipeline.

The Science DMZ takes the DMZ design one step further. As opposed to a traditional DMZ, by design, the Science DMZ does not feature a stateful firewall, which would negatively impact throughput. This subnet is implemented at or near the edge of the university's campus network to reduce the number of network devices between Science DMZ systems and the WAN [19]. Ideally, the Science DMZ router is connected directly to the border router as reflected in Figure 2.2. In this model, the border router connects directly to the WAN and routes traffic to and from campus. If this traffic requires connection to the campus LAN, traffic is routed through the campus border firewall. However, if this network activity only involves communication with Science DMZ systems, it is routed through a high-performance, dedicated Science DMZ switch without being processed by the border firewall. Communications between the Science DMZ and campus LAN are routed through the border firewall, border router, and Science DMZ switch. Through this architecture, traffic is not shared between the Science DMZ and the campus local area network (LAN) thus improving network performance on the Science DMZ. Additionally, the number of connections between Science DMZ systems and the WAN are extremely limited to reduce the risk of dropping TCP packets, which would negatively impact performance.

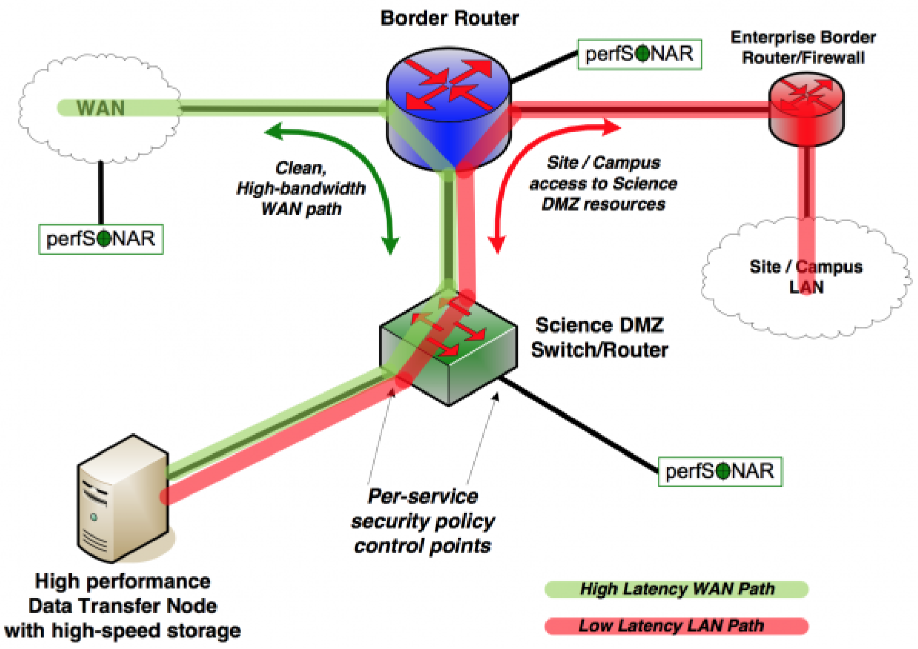

Border Router
perfSONAR
Enterprise Border Router/Firewall
WAN
Clean, High-bandwidth WAN path
Site / Campus access to Science DMZ resources
perfSONAR
Site / Campus LAN
Science DMZ Switch/Router
perfSONAR
Per-service security policy control points
High performance Data Transfer Node with high-speed storage
High Latency WAN Path
Low Latency LAN Path

Figure 2.2: A simple overview of the ideal Science DMZ implementation where red represents campus access to Science DMZ systems and the Science DMZ data flow to the WAN is shown in green, from ESNet [26].

The issue of security is raised without a stateful firewall protecting Science DMZ systems. To address this concern, there are a number of recommendations by ESnet to increase security [22]. It is recommended that router access control lists (ACL) are utilized on the Science DMZ switch. Additionally, port control should be managed on the machines native to the Science DMZ through software such as IPTables. Third, network intrusion detection systems and host intrusion detection systems should be configured for the Science DMZ. Lastly, the Science DMZ is to remain as isolated as possible from campus LAN systems.

2.2 Data Transfer and GridFTP

Data transfer is the primary usage of the system outlined in this thesis. As such, it is paramount to define research data transfer in this context. In general, data transfer is the movement of large files or numerous files between systems or organizations. The first solution, File Transfer Protocol (FTP), was developed by M.I.T. in 1971 based on the Network Control Program and was updated to run using Transmission Control Protocol (TCP) in later iterations [59]. A brief overview of how FTP works is displayed in Figure 2.3. There are three primary pieces to FTP transfers. First, a TCP 3-way handshake is utilized to establish communication between the client and server. In this example, the Workstation serves as the client. Following communication establishment, the user is authenticated against the server machine's users. This step may be skipped to allow anonymous connection if the server is configured to allow this form of connection. Finally, the connection is fully established and data is freely transferred. While this method has been in use several years, at the base level FTP has several limitations. FTP was designed as an insecure solution to data transfer with all data, including usernames and passwords sent as clear-text without encryption. This makes data sent over FTP vulnerable to the most basic attacks. Additionally, the connection can be slow to establish, and due to the security settings of FTP, firewalls can have a difficult time handling FTP traffic. Because FTP is built upon TCP, FTP is also subject to all of the limitations of TCP.

In more recent years, there has been a shift in approaches to data transfer with browsers, including Firefox and Google Chrome removing support for FTP transfers in general [32, 53]. This is in part due to a shift toward HTTP as the primary method of data transfer because this protocol addresses several of the limitations of FTP. However, as a long standing solution, FTP has not fully disappeared but been built upon. Several derivatives of FTP have been developed to address some of the

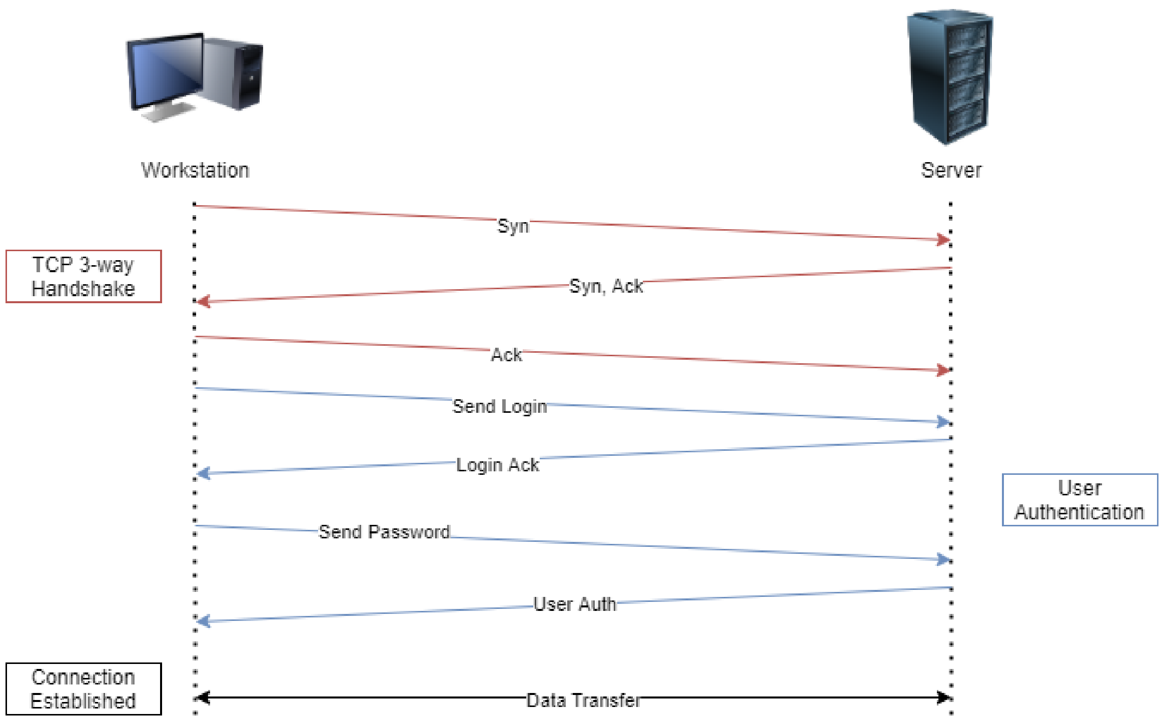


Figure 2.3: An overview of traditional FTP communication establishment built on TCP.

limitations of FTP, including sFTP, the secure evolution of FTP. Another popular development is GridFTP.

GridFTP is an extension to FTP to support failure detection and transfers from multiple data sources while increasing security [4]. It was developed by the University of Chicago with an emphasis on supporting data striping. This allows for data transfers across storage pools that are used in a highly parallel cluster environment. Tested against other existing FTP servers, GridFTP performs faster both in striped and single-process environments [5]. This framework is the platform that Globus, an open source data transfer application defined further in Subsection 2.10.1, is built upon. Despite FTP losing popularity, GridFTP remains highly used among universities and organizations through Globus. However, GridFTP still requires installation on both ends of a task to facilitate the transfer. Managing a server for transfers is time consuming for researchers when they could be focused on their individual research. Instead, a dedicated, campus-wide data transfer node can be configured for data staging and transfer using GridFTP.

2.3 Data Transfer Nodes

A data transfer node (DTN) is a dedicated Linux server for transferring data on a WAN, and the typical use case for the Science DMZ. The full end-to-end path from a laboratory within an institution to a laboratory at a different institution has too many limitations across the various networks and subnets that data crosses, which will eventually cause a large transfer to fail. Instead, researchers stage data by performing a transfer from a laboratory deep within an institution's LAN to a DTN at the edge of the campus network. The DTN is then capable of making expedient, large transfers to another DTN across the internet, from where the collaborator can transfer data to their local machine according to the limitations and security policies of their own network.

DTNs feature local storage for data staging purposes in which users can post data temporarily in a compressed and archived format that will be pulled by a collaborator from a remote location. Some additionally feature integration with external storage resources to manage transfers from multiple locations in a single location. For security reasons, DTNs are not permitted to run any general-purpose tasks such as mail servers [27].

Achieving fast data transfer speeds requires DTN tuning, a process in which the DTN receives a series of system modifications to increase performance at the frame, packet, or disk input/output levels. These system modifications are dependent upon the network on which the device belongs as well as the device itself, not a specific set of specifications that apply regardless of the environment. However, as the engineers behind the architecture of the DTN, the general guidelines provided by the Energy Sciences Network provide a general overview of possible changes to be made to maximize data transfer speeds for a DTN [27]. To understand a DTN is to then understand the components belonging to this hardware so as to make proper decisions regarding modification.

Arguably, the most important components to understand regarding a DTN are the Network interface controller (NIC), Peripheral Component Interconnect Express (PCI-e), and vendor-specific items, such as the Intel QuickPath Interconnect (QPI). The NIC, also known as a network adapter, is responsible for system connection with the network, wired or wireless, and setting the MAC address of the machine. It is a data link layer and physical layer device. A modern network adapter also provides advanced settings for administrators, including interrupts and direct memory access. While connection to the internet is important, the NIC on its own cannot perform a network transfer on a DTN, so other hardware also contributes to this process. Communication between these hardware components on the motherboard is handled by the PCI-e bus. After a request moves from the NIC to the PCI-e, it must then be handled by one of the system's cores. However, depending on the core chosen, there may not be a direct connection between the PCI-e and that core. An excellent example of this can be viewed in Figure 2.4, which shows the Intel Sandy Bridge architecture, including core sockets, PCI-e connections, and the QPI bus. The QPI has been the standard for Intel microprocessor architecture since 2008 to provide high-speed, scalable communication between processors [39]. While often referred to as a bus, the QPI more closely represents point-to-point interconnections. As opposed to a standard front-side bus, the QPI is narrow, resulting in a small but fast platform. According to this architecture, only one cpu socket has a direct connection with the NIC while the other must communicate over the QPI to reach it. The result is a bottleneck between sockets for network communications. This will negatively impact DTN data streaming performance, and requires specific mitigation steps that will be addressed in Chapter 4 of this thesis.

2.4 High Performance Computing

While DTN systems can handle data transfers effectively in manually-controlled and managed environments, new demands of management scaling, automation, and software integration are emerging as cyberinfrastructure systems evolve towards inter-

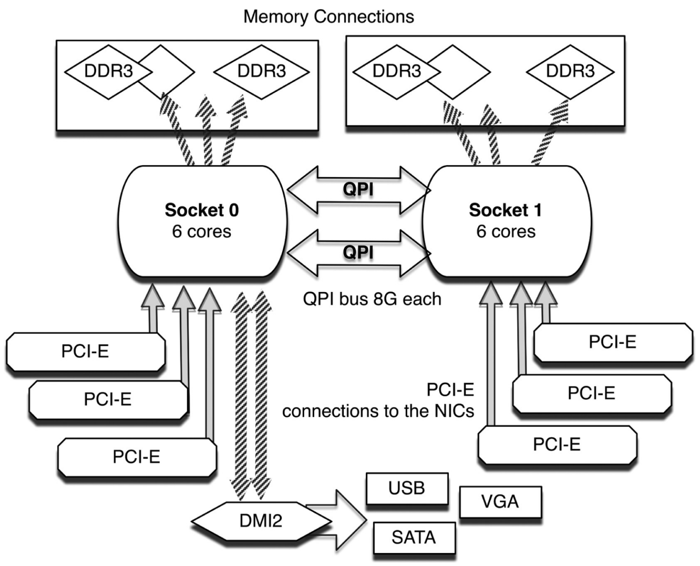


Figure 2.4: Intel's Sandy Bridge Architecture, showing memory connections across a system, from [27].

operability and regional federation. Therefore, it is important to have methods in place to monitor deployments to ensure availability and performance. The system outlined in this thesis that is responsible for monitoring other systems incorporates Cloud Computing, defined in Section 2.6, which finds its roots in High Performance Computing (HPC).

HPC is commonly considered as the practice of aggregating compute power to process data and perform complex calculations faster than a traditional computer. While the concept of HPC has stayed the same, the method through which this has been achieved historically has changed over the years. HPC was originally known for the development of supercomputers before taking a cluster approach. Today, HPC typically follows a hybrid model of powerful computers in a cluster environment.

Historically, the CDC 6600 is generally recognized as the first supercomputer and the beginning of HPC. The CDC 6600 was created by Seymour Cray and the Control Data Corporation (CDC) in the early 1960's. It featured 400,000 transistors, more than 100 miles of wiring, and reached a top speed of 40 Mhz [88]. Cray eventually left the CDC and founded a new company, Cray Research, where he released the Cray-1. The Cray-1, shown in Figure 2.5, was a new form of supercomputer that allowed for limited parallelism [70]. This supercomputer pulled 115 kW of power in exchange for an 80 Mhz processing speed, and was the top supercomputer on the market from when it was released in 1976 to 1982 [18], only to be replaced with the Cray X-MP. The Cray X-MP was among the first computers capable of multiprocessing and was capable of 105 Mhz [8]. This period of time in HPC was known as the era of Clay architecture.

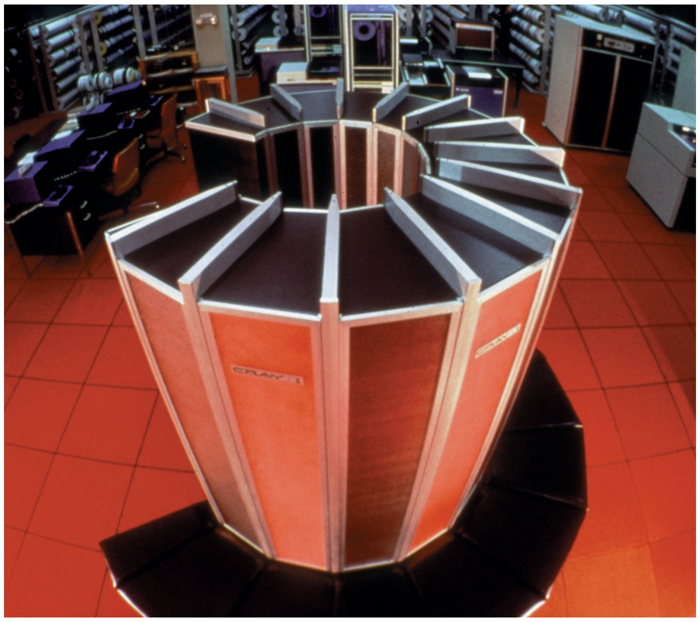


Figure 2.5: The Cray-1, released in 1976, was among the first supercomputers and the first computer to allow parallelization [38].

While each of these machines are historically important and were incredible developments for their time, they all had the same disadvantages. Cray architecture was expensive, with the Cray-1 costing up to $10M in 1977, or upwards of $44M today when adjusted for inflation. Additionally, the computers were large with the Cray X-MP at 2.62 m (8.6 ft) x 1.96 m (6.4 ft). Lastly, there is a limit to scaling up, as explained by Moore's law. Moore's law states that the number of transistors on a microchip will be doubled every two years while the cost of computers is halved [71]. However, Moore's law has become synonymous with the physical, upward bound of transistors and computation. While many are confident in the ability of ingenuity to continue to test these bounds, it requires creativity. Vector supercomputers, like those developed by Cray, have a theoretical maximum performance. The solution proposed by HPC professionals is clustering.

Clusters are a single system of connected computers that communicate with one another to share computations. In the early 1980s clusters of killer micros began to challenge the performance and scalability of vector supercomputers for a fraction of the cost, and companies began to develop methods to improve communication between microprocessors [9]. It wasn't until 1995 that clusters began to be economical though and HPC took a drastic turn. This was the introduction of the Beowulf, showcased in Figure 2.6, a computing cluster developed by NASA where engineers were challenged with creating a cluster capable of 1 Gflops for less than $50,000. The resulting cluster was built on the Pile-of-PCs model, the concept of combining multiple, separate personal computers across a private network [67]. This new Beowulf model emphasized the use of no custom components, replicability, scalability, and providing the design and any improvements to the community [77]. While Beowulf clusters are still utilized, there are disadvantages to this model. All machines in the cluster are homogeneous, and code ran on the cluster must be written to make use of parallelization. Otherwise, it does not take advantage of the benefits of the cluster.

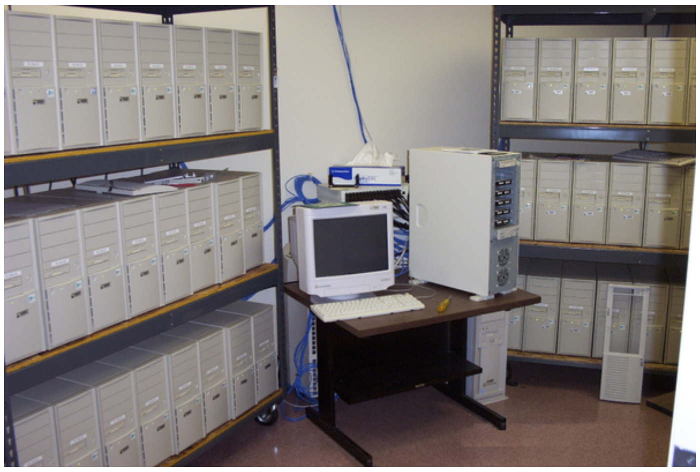


Figure 2.6: A Beowulf cluster developed at Lockheed Martin Space Operations [92].

HPC in recent years has taken a hybrid model approach, known as hybrid clusters, with an emphasis on the cloud and services provided over the internet. Hybrid clusters combine the usage of more powerful nodes with a distributed approach to computing. The servers added to a cluster are chosen based on a number of cores rather than individual processing speed. By providing these assets through the cloud, these compute resources are then available publicly or within an institution. Considerations for modern HPC solutions using hybrid clusters include interconnect speed, shared memory access, and performance, which vary based on implementation as these are specific architectures in the host systems and their connecting internal network interfaces within the cluster environment, not LAN.

2.5 Cluster Computing

Cluster computing typically refers to utilizing multiple identical computers that are either tightly or loosely coupled to shared computational tasks. While HPC was the beginning of clusterization, clusters have expanded to include the three following types:

- High Performance Computing
- Load Balancing
- High Availability

Where HPC clusters are designed to improve multi-node parallel processing performance, load balancing clusters are utilized to distribute a workload across multiple machines that may not be dedicated to the cluster. A machine may be a cluster member while maintaining its ability to be utilized individually. In this instance, resources on the machine may already be heavily utilized and a load balancer would be utilized to ensure traffic is not sent to a machine that is already inundated with requests when another machine is receiving less traffic [12]. High availability clusters are dedicated to maintaining reliability and job status. In practice, most clusters combine all three concepts and all feature a cluster resource management system (CRM), such as Slurm, YARN, or Kubernetes.

The CRM is a distributed computing framework that maintains the cluster state and manages computing job assignments across the cluster. Most CRMs organize the cluster based on master/worker architecture, also known as master/compute architecture, presented in Figure 2.7. Individual computers and virtual machines are split into cluster nodes known as masters and workers. In this model, the master nodes are primarily responsible for maintaining communication with nodes and job scheduling. Rarely is this node responsible for computation on the cluster in addition to this role. While the most simple clusters feature a single master, it is recommended that a cluster have multiple master nodes to have high availability and fault tolerance. Worker nodes are responsible for executing scheduled jobs. In most instances, the status of a job is not sent back to the master. Instead, to address the possibility of scheduling a job on an unavailable node, most CRM feature a heartbeat between masters and workers as utilized by YARN [89]. The heartbeat is a regular ping of each worker node by master nodes. A response from the worker signals to the master that the worker node is online. In the event no response is received, the master will attempt to schedule a job on a different node.

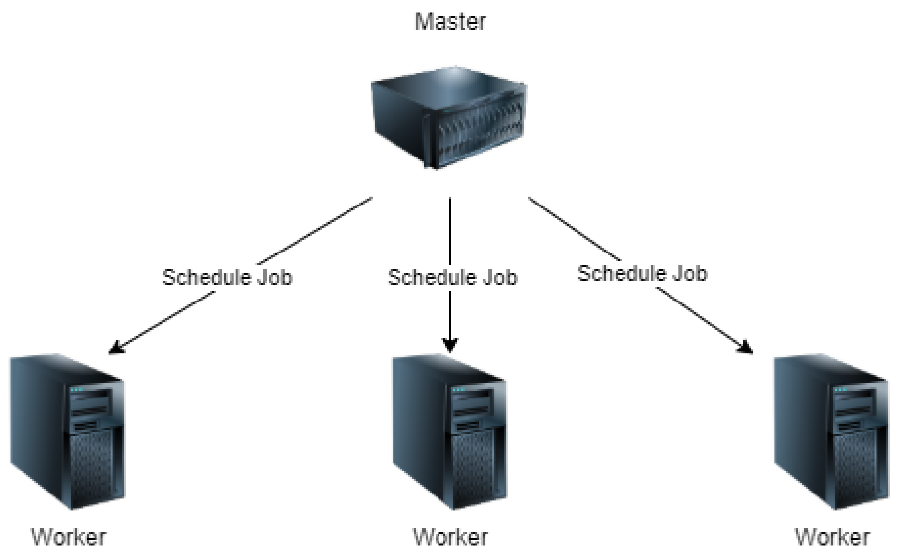


Figure 2.7: An overview of Master/Worker architecture used in cluster environments.

Kubernetes features specific terminology important to understanding how a cluster is managed. This CRM will be discussed in more detail in Subsection 2.10.4.

While this approach is helpful for maintaining a service and providing fast computation, there are two remaining issues when applying this model to provide campus-scale cyberinfrastructure. First, because clusters are homogeneous in nature, this would require the institution to maintain identical hardware across the institution to support this. Second, this would result in downtime for scheduled maintenance if the hardware was upgraded for the cluster. To provide data transfer solutions, the underlying cyberinfrastructure should not be subject to extended maintenance periods.

2.6 Cloud Computing

Cloud computing refers to delivering a hosted service over the internet. While no standard definition exists, it is generally agreed that cloud computing has five characteristics as defined by the National Institute of Standards and Technology, consisting of on-demand self service, broad network access, resource pooling, rapid elasticity, and measured service [49]. There are three primary service models when it comes to defining cloud computing. These are offering access to an application as a service over the internet, known as Software as a Service (SaaS), hardware and software in data centers utilized to support these applications, known as Infrastructure as a Service (IaaS), and Platform as a Service (PaaS) [7]. All three belong to one of two categories. If the application, platform, or infrastructure is available across the general internet, then it is a public cloud. If instead these resources are available only within an organization, institution, business, or other entity, it is a private cloud.

No matter the service model and deployment, the motivation behind becoming a cloud user or cloud provider is to reduce costs while improving performance. Instead of requiring several machines dedicated to a single project, a project can be hosted on the cloud, where a project can be maintained alongside other projects across the same servers with minimal management by users. Cost of entry into compute-heavy research or business models is lowered, allows for immediate access to hardware, and aids in scaling [48]. This allows smaller organizations to gain access to computational resources without personnel to manage these resources, and larger institutions to organize their services.

2.7 Containerization

Containerization is the packaging of software applications such that these applications run within isolated environments called containers, which all share the host operating system (OS). The primary benefit of this is to run several, small jobs that are portable, contain all software dependencies, and are isolated from the hardware environment. As shown in Figure 2.8, containers are traditionally more lightweight to deploy on a system than a whole virtual machine due to their lack of a Guest OS, a dedicated operating system installed only within the virtual environment [21, 50]. This difference is the result of having a Container Engine as opposed to a Hypervisor.

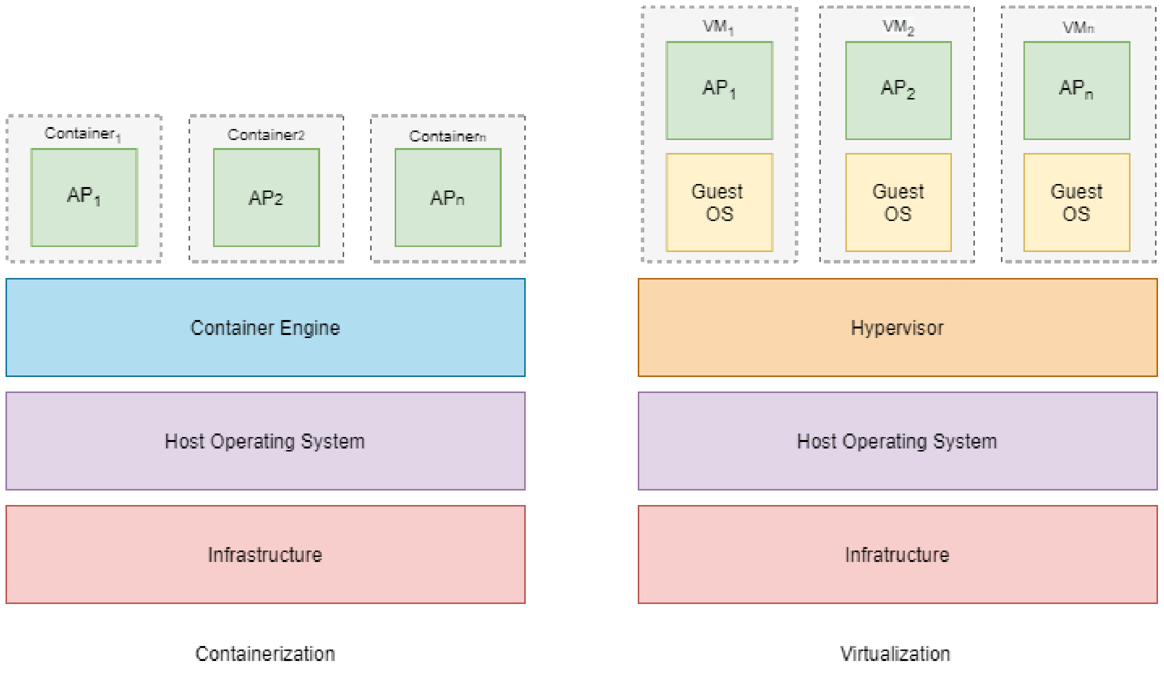


Figure 2.8: This is a high-level comparison between Containerization and Virtualization, showing the higher overhead of virtual machines on a system.

Virtualization works by utilizing software to crate an abstraction layer over the hardware components of a computer. This allows hardware resources of a single machine to be divided among several virtual machines, effectively creating several machines on one computer. In the case of traditional OS virtualization, this is software is called the Hypervisor. Virtual machines were designed with this architecture to maintain full isolation[30]. However, when multiple applications are running with the same operating system, the resulting bloat on the host system slows performance. By comparison, Containers abstract the operating system of the host machine to run applications through the Container Engine, resulting in a virtual environment that is more lightweight than a virtual machine. In avoiding running multiple virtual machines simultaneously, hardware resource usage is less impacted by operating systems, resulting in faster processing within the containers and deployment time [72]. However, while this provides a layer of abstraction, it should be noted that all processes within containers still share the same kernel.

Despite being slower, virtual machines still have several use cases where containers may not be the more effective choice, including when applications need to run together, or in the case of monolithic architecture. However, containers are typically a better choice for microservices, DevOps, web applications, and the cloud through cluster orchestration [57].

2.8 Containerized Clusters for the Cloud

A private cloud environment can be deployed by combining clusters and containerization. Private and public cloud environments are the current primary use case of container orchestration technologies, such as Kubernetes or Docker Swarm, but this is further expanded in Subsection 2.10.4. The architecture of this cloud environment is the same as standard cluster computing where a master node controls workers, however, all jobs and control features are individual containers within nodes apart from a load balancer for master nodes. For example, the scheduler for a master node is a container managed by the container orchestration technology that is limited to running on a master node. This scheduler then assigns jobs with a load balancer to manage job assignment across workers. The external load balancer is only present in clusters where there are multiple master nodes. This external load balancer manages which master assigns jobs.

The idea of scheduling jobs as containers on a cluster comes from the combined ideas of scalability, and high availability previously discussed in clusters as well as isolation and portability from containers. In other words, by deploying jobs as isolated containers, an application can be deployed on heterogeneous systems to maintain up-time without requiring extra dedicated development. This has resulted in applications being split into microservices that can be easily managed within containerized environments across a variety of systems. The process of isolating these applications is referred to by the Cloud Native Computing Foundation as cloud native application development [17]. Further results of cloud native application development include faster release times due to their inherent support of DevOps practices, ease of management, and reduced costs without the need for dedicated systems to a singular application.

2.9 File systems

A file system is how an operating system organizes data. There is no best file system for all situations, which has resulted in many file system solutions being developed. Each file system has a different method for organizing data and metadata, resulting in differences in performance, and some file systems provide additional features for increased scalability or security. In this section, the file systems involved in DDRIS, ext4, ZFS and CephFS, are discussed, including their advantages and disadvantages.

2.9.1 ext4

The embedded file system present on Linux devices is ext4, the fourth version of the extended file system. The ext series of file systems started in 1992 with each iteration building on previous versions. As the most recent version, ext4 has increased reliability against fragmentation, performance and capacity [15]. While ext4 is not the highest rated file system in terms of performance or ease of use, it is considered a good choice for servers where a volume manager is not necessary with comparable performance to other similar solutions [2]. Because it is the default Linux file system, most users and system administrators understand how to configure ext4. This results in better maintainability as system administrators handover responsibility of the systems to incoming professionals.

2.9.2 ZFS

ZFS is a scalable file system that combines volume managers with traditional file systems, allowing for distribution across multiple drives or a storage pool [68]. Memory across drives is pooled into a single cache to be utilized in the file system where ZFS handles partitioning for an administrator. This is done by combining physical drives into virtual devices (VDEV) that are combined into a ZFS storage pool. The number of VDEVs present within a pool is dependent upon system configuration. Figure 2.9 shows how pooling devices is achieved for ZFS with six drives are divided into two VDEVs. One application of splitting these disks is to support two separate RAID arrays in a single file system. More memory can be added to the pool and automatically incorporated into ZFS. This comes with a number of benefits, namely the ability to scale up on a system. However, ZFS lacks the ability to scale outward efficiently with data redundancy being for whole disks, not objects, and once a disk is added to the ZFS pool, the disk cannot be removed.

ZFS includes different levels of software-based RAID, known as RAID-Z. There are three levels of RAID-Z including, RAID-Z1, RAID-Z2, and RAID-Z3. RAID-Z1 is the original level provided and is similar to RAID-5 but with a variable stripe width [35]. The expansion of RAID-Z saw RAID-Z2, a software-based RAID-6. RAID-Z2 has two parity blocks as opposed to the single parity block of RAID-Z1, making it more fault tolerant [64] but slower in performance due to parity block write times [1]. RAID-Z3 includes triple parity. A comparison of RAID-5 and RAID-6 levels can be seen in Figure 2.10, showing differences in handling parity and minimum

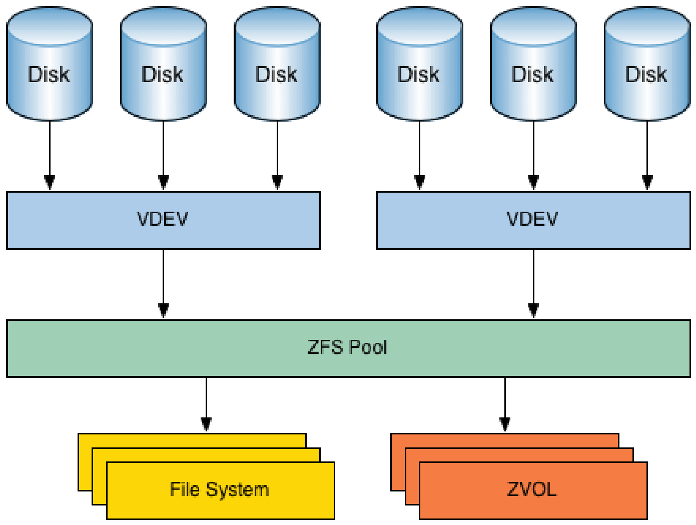


Figure 2.9: An overview of how devices are pooled with ZFS [91].

number of disks. The advantages of using software-based RAID are that these RAID levels do not require the hardware component of a RAID controller and an added level of control over what RAID level is being utilized. A hardware controller cannot be changed to include a level that has not been included by the manufacturer. However, software can emulate RAID without the need for a controller and the software can be chosen based on what levels are supported. The primary disadvantage of using software-based RAID is that it is slower than hardware-based RAID as it utilizes some of the processing power of the computer in order to emulate hardware RAID.

2.9.3 CephFS

CephFS is a POSIX-compliant file system built on the distributed object store provided by Ceph [16]. It provides object, block, and file system storage. The three primary components of the Ceph file system are clients, the object storage cluster (OSD), and metadata servers (MDS) [90]. Clients perform I/O operations directly with OSDs, which store all data and metadata. The MDS manages the namespace, including filenames and directories. After metadata operations have been completed, metadata is journaled by MDS to the OSD.

Ceph, as a distributed file system, combines all available storage into a single

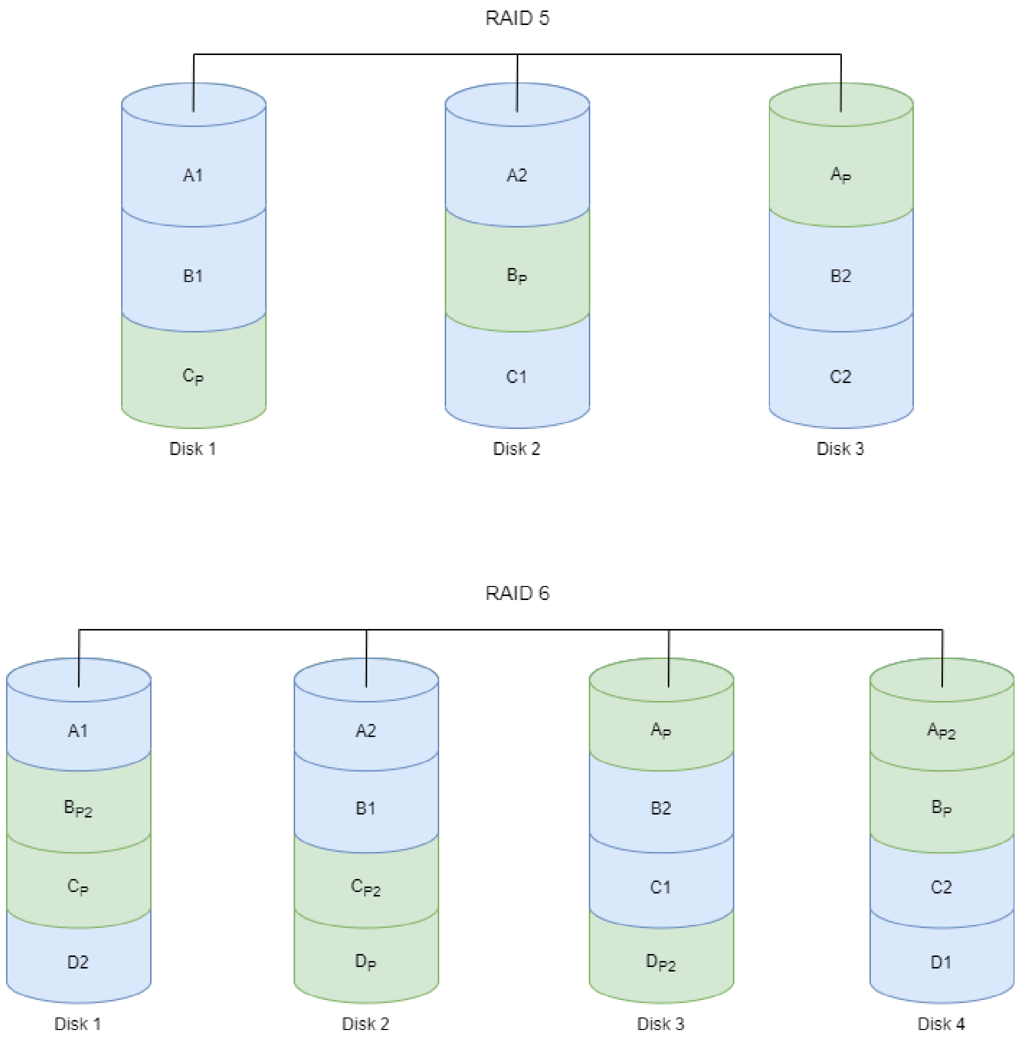


Figure 2.10: A comparison between RAID-5 and RAID-6, the basis for software-based RAID levels included with ZFS where parity blocks are denoted by "P".

storage pool by decoupling the software-based file system from the storage hardware, including across a cluster. However, unlike ZFS, Ceph will automatically rebalance clusters if new nodes are added or removed. This ability to effectively scale outward makes Ceph more appropriate in a cluster environment where nodes may be added, removed, or replaced than ZFS. However, on a single machine, Ceph is outperformed by ZFS. Ceph also does not provide software-based RAID. Instead it provides data redundancy through replication and erasure coding.

2.10 Technologies Utilized

The implementation of DDRIS required a multitude of technologies to promote quick future deployments and reliable data handling. Many tools were chosen due to their preexisting usage on other University systems in order to promote standard tool usage at a campus scale. In this section, a brief overview will be given for many of the different tools utilized in the formation of DDRIS components. It should be noted that other modern frameworks and software can be utilized in replacement of many of these technologies in deployments similar to DDRIS to fit existing standard practices at an institution. A high-level representation of DDRIS architecture, where these technologies have been applied, is available in Chapter 5.

2.10.1 Globus

Globus is a non-profit, "software as a service" solution to data transfer management developed by the University of Chicago [86]. It is designed to make data transfer and collaboration easy through a visual, Dropbox-like application. The model has automatic fault recovery and network tuning. A high-level representation of data workflow for Globus as illustrated in Figure 2.11. Users submit a data transfer request to Globus Online. Globus handles networking handshakes and authentication to transfer between a source and destination. After the transfer is completed, the user who requested the transfer is notified by Globus of the transfer's completion or any errors that may have occured. This model allows a "fire and forget" workflow where

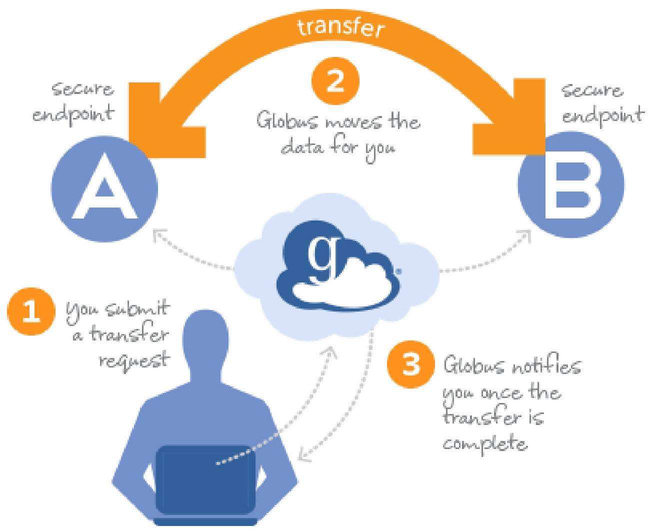

transfer
secure endpoint
secure endpoint
2 Globus moves the data for you
A
B
1 You submit a transfer request
3 Globus notifies you once the transfer is complete

Figure 2.11: A high-level representation of the Globus workflow for researchers [84].

researchers do not need to watch the data transfer. The Globus data management solution has two major components, Globus Online and Globus Connect.

Globus Online is a web portal for researchers that allows for data transfer, deletion, and sharing without local application installation [28]. Instead, it connects with remote locations to allow users to manage their data without accessing the machine directly. Researchers also have the ability to manage data access through this user interface. The learning curve for Globus is shorter than full command line, which allows researchers to spend more time researching and less learning software communication methods. Information on how to use Globus Online will be discussed in Chapter 6.

Globus Connect consists of two types, Globus Connect Server and Globus Connect Personal (GCP). Both are designed as a GridFTP server that communicates with Globus systems [6]. GCP is meant to be easily installed on a local machine, such as a lab computer or laptop, without necessary configuration. Globus Connect Server is a larger application meant to be installed on a managed server for multiple users to connect with.

This data transfer solution has its own limitations. Due to overhead in authentication with GridFTP, numerous small files transfer significantly slower that one large file [46]. This will result in a need for effective data transfer workflows within the institution, and data needs to be archived prior to staging.

Globus is utilized as the primary software behind data transfers utilizing DDRIS. This is due to the easy-to-use front-end interface provided through Globus's online web portal and support across common operating systems. Many other data transfer services require knowledge of command line interfaces or are operating system dependant. The primary advantage of having a GUI is in limiting the amount of time that must be dedicated to learning or remembering how to utilize the software, and as a cross-platform tool, data transfer methodology can be standardized while supporting the needs of individual researchers with different research apparatuses. Another primary advantage is the ease of installing GCP. GCP does not require extensive setting configuration. This allows researchers to begin transfers quickly without concern for deployment time.

2.10.2 Python

Python is an interpreted, general purpose programming language developed under an open source library [63]. As a popular language for back-end system development, Python has an active community of content creators that have lead to several available frameworks for development. Python benefits from the simplicity of its syntax, making code easy to read and comprehend when an application is inherited by a new developer. Globus allows connection to their services through the use of an officially supported Python software development toolkit (SDK). Due to these reasons, Python was chosen as the primary development language for all DDRIS back-end services, including log exporters, micro web framework deployment, and Prometheus exporters.

2.10.3 Docker

Docker is a containerization platform for developing and running applications [20]. It also provides tools to manage container lifecycles. Docker was chosen for DDRIS due to its active development community and support. This platform benefits from this community as members often publishe modules and container images for public use and review. Between the community and easy to learn nature of Docker, it is often chosen by DevOps teams for deployment. Docker also works well with Kubernetes, the central framework involved in DDRIS.

2.10.4 Kubernetes

Kubernetes is an open source solution to managing and deploying containerized applications originally designed by Google [80]. This is ideal for cloud-native applications, which increases the scalability of deployments. It provides a platform to implement and rely on containers in production. A simplified overview of one option for Kubernetes architecture can be seen in Figure 2.12. This architecture is similar to that used by DDRIS. Kubernetes creates a cluster of nodes where a node is a physical or virtual machine. As referenced in Section 2.5, these nodes consist of Master and Worker nodes. Master nodes contain the Kubernetes control plane that controls and schedules jobs as well as monitors the cluster's state. The control plane has four components when simplified, the API server, controller manager, scheduler, and etcd. The API server is responsible for communication across cluster components and is the method through which users access pieces of the cluster [14]. Second, is the Controller Manager. The Controller Manager consists of four controllers, the node controller, replication controller, endpoints controller, and service account & token controller. A brief overview of each controller can be seen in Table 2.1. Third, the Scheduler orchestrates jobs based on resources required by an application, resources available on a node, and if a job has a node preference set in its configuration. Finally, etcd is the data store for the cluster. It saves metadata relating to very object created within the cluster. The Master nodes utilize a Load Balancer, to balance job orchestration amongst all Master nodes on the cluster. These jobs are not containers themselves, but pods. Because Kubernetes supports several container types, it instead manages a wrapper around the container called a pod. Each pod belongs to a namespace, used

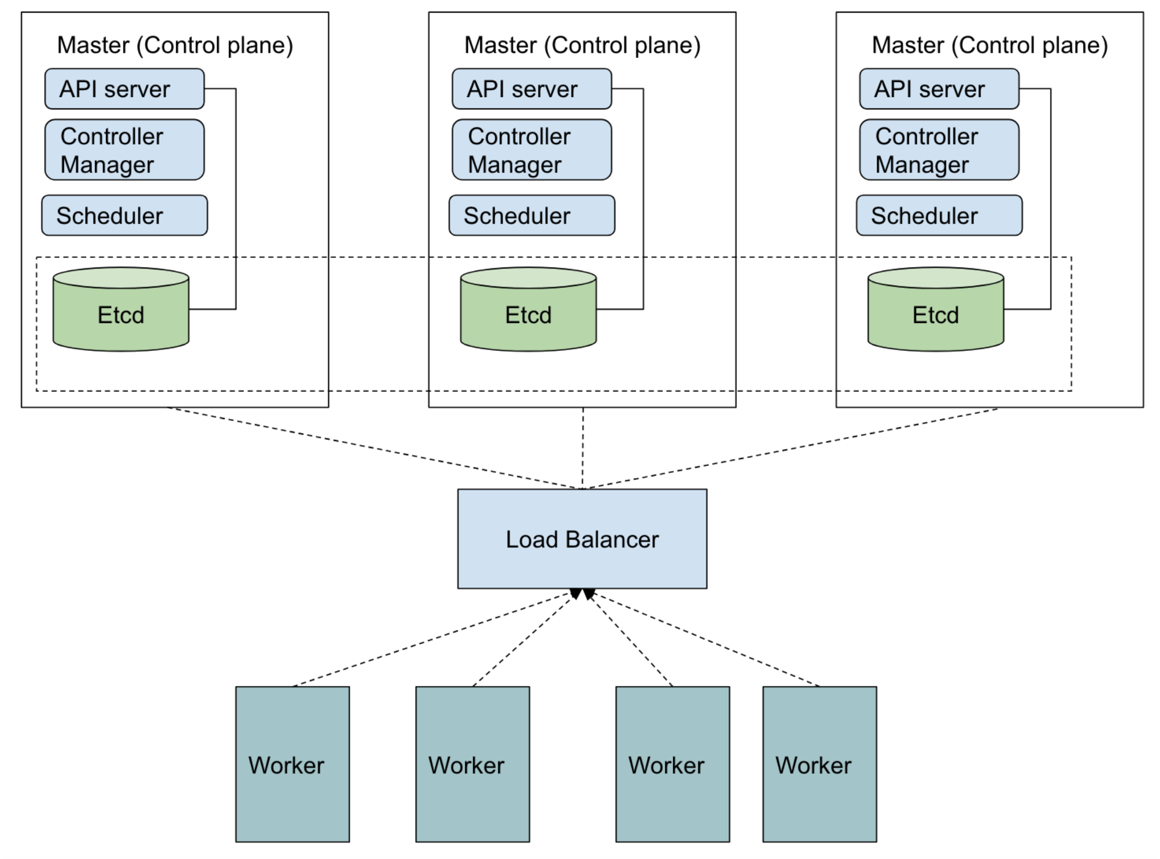


Figure 2.12: The Kubernetes architecture featuring multiple master nodes with co-located Control-Plane and Etcd [3].

to organize pods based on project and limit user activity to specific projects relevant to those users.

Kubernetes is used in DDRIS to run containerized administrative jobs. Prometheus data, measurement archives, log data, and visualization services run through Kubernetes. Kubernetes was chosen for DDRIS due to the portable nature of containerized applications and scalability as the array of DDRIS services continues to expand alongside other centralized services.

2.10.5 Prometheus

Prometheus is a community driven, open source metric collection and reporting software utilized in system administration and cyberinfrastructure for time-series data [61]. The primary components to the Prometheus architecture can be seen in

Table 2.1: Description of controllers native to the Kubernetes Controller Manager.

Controller	Description
Node Controller	Monitors for nodes that are down and responds accordingly.
Replication Controller	Manages pods so that the correct number of pods are always active if pod replication is specified.
Endpoints Controller	Joins services and pods
Service Account & Token Controller	Create accounts and access tokens for new namespaces

Figure 2.13 where the four main components include the Prometheus server, Alertmanager, Pushgateway, and Prometheus web UI. The Prometheus server is responsible for periodically scraping data from Prometheus targets, jobs, and exporters, and the server discovers new targets through service discovery. The pulled data is then manipulated through a set of rules to be formatted as time-series data and checked against alerts. This data is saved in a targeted time-series database to be queried by the Prometheus web UI or other data visualization tools. If the data is flagged by the Alertmanager, notifications are sent through supported plugins, such as Pagerduty or an email server. The exception to this workflow is when a Pushgateway is utilized. Where Prometheus is usually a pull-based architecture, metrics can be lost for short-lived jobs that exist within the scrape interval. The Pushgateway allows metrics to be pushed to the Prometheus server to be stored and analyzed.

DDRIS utilizes the Prometheus server to scrape data off of Prometheus targets, web servers that host exported information to be pulled by Prometheus over HTTP, and the Prometheus web UI as part of its data visualization solution. All metric reporting for DDRIS is completed through the pull model over HTTP as opposed to push over UDP, so no Pushgateways are present in DDRIS. Additionally, alerts are not handled through Prometheus. While Prometheus is an important tool, it is limited by its ability to only collect numerical data. Prometheus is not suited to handle extracting information such as user data.

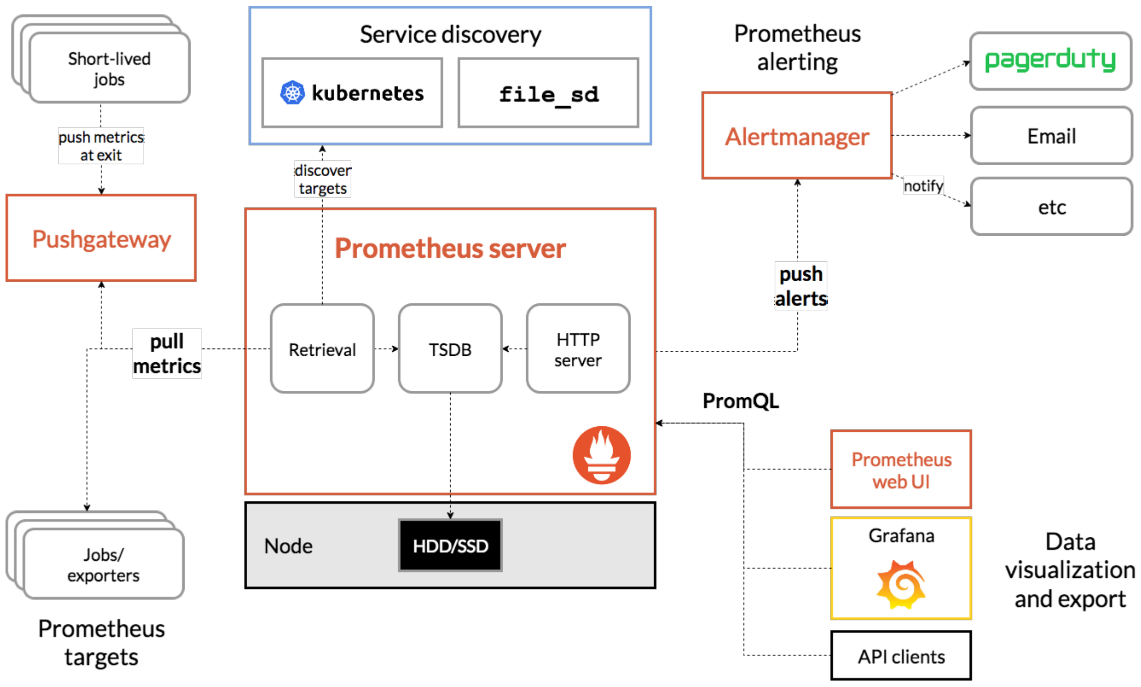


Figure 2.13: The Prometheus architecture including ecosystem components [60].

2.10.6 Grafana

Grafana is an opensource analytics web application written in Typescript and Go [33]. It is capable of pulling from several databases to automatically create interactive data visualizations specified by the user through database queries. These data visualizations allow users to add annotations, and alert management with specified thresholds and integration with communication software, including Discord and Slack, to publish alerts. Grafana is a popular analytics tool utilized in research cyberinfrastructure that allows for expansion through plug-in support developed by the active Grafana community. It can be installed locally to create a private dashboard or served through the public cloud. For use in DDRIS, Grafana is being privately hosted on Kubernetes.

2.10.7 Flask

Flask is a python-based Web Server Gateway Interface (WSGI) microframework [58]. As a microframework, the core of Flask only controls how a webserver connects to web applications. Flask does not have a built in database, data abstraction layer, or form validation. Instead, Flask is highly customizable. It supports extensions and libraries that perform these functionalities and more, such as network calls, leaving their implementation at the discretion of the developer. This makes the core of Flask very lightweight but easily scalable for large projects. Flask supports ssl context specification when declaring Flask environment variables. This allows the application to be run over HTTPS instead of the default HTTP. Due to the use of demilitarized zones throughout DDRIS, this is important to add a level of security to communications.

2.10.8 PerfSONAR

The performance service-oriented network monitoring architecture, known as PerfSONAR, is an open-souce toolkit for network telemetry to discover realistic end-to-end performance expectations [25, 81]. PerfSONAR allows for automatic, scheduled testing within and beyond the boundaries of a network. Through the use of PerfSONAR meshes, a single testing configuration can be shared across multiple nodes to provide a detailed map of network performance. Mesh configuration is hosted on a webserver and pulled by mesh members to be stored locally. The results of network tests are stored within a PerfSONAR Measurement Archive for query and display. The Measurement Archive utilizes Esmond [23], a system for gathering, storing, visualizing, and analyzing data, to turn transfer results into time-series data for storage. Areas of low performance are flagged to be investigated based on location and time of day. This toolkit promotes more efficient network troubleshooting and faster internet access to users. DDRIS includes a local, campus PerfSONAR mesh to provide detailed network telemetry, including long-term storage and historical data visualization to ensure users are able to transfer their data quickly through network oversight. The head of the DDRIS mesh is available for inclusion on external PerfSONAR meshes to provide insight on connections past the university borders for collaboration.

2.10.9 Cassandra

Apache Cassandra is an open-source distributed NoSQL database [79]. It is utilized by DDRIS to store PerfSONAR test results to be visualized. Cassandra was chosen due to it being the default storage for PerfSONAR. Where NoSQL databases are typically not suited for complex queries, Cassandra provides a query model similar to SQL, referred to as Cassandra Query Language (CQL). However, Cassandra is not supported by Grafana at this time, making data visualization more difficult as it will have to pass through a supported platform before being visualized.

2.10.10 Postgres

Postgres is an open-source relational database system [82] and is used for DDRIS table data, and local storage of the PerfSONAR mesh configuration. It is considered to be one of the most popular databases in use today with an active user community. Postgres uses a modified version of SQL known as PostgreSQL that provides the same features as SQL as well as custom datatype and function declaration. These user defined functions can be written in any procedural programming language, including Python. Postgres was chosen as the primary database system for DDRIS due to its status as a well-established, open-source solution.

2.10.11 TimescaleDB

TimescaleDB is an open source, relational, time-series database built on Postgres [83]. This solution offers the scalability of a NoSQL database with high performance. It is the only open-source time-series database with native SQL support. Due to this exclusivity, TimescaleDB is utilized in DDRIS for time-series data storage outside of PerfSONAR results. Prometheus is set to export all data collected by the server to TimescaleDB. This is then queried for display on the DDRIS dashboard in the same method as Postgres data.

2.10.12 Bash Shell Script

`bash` is the default Unix shell and command language for Linux systems. When several bash commands are written together in a single text file, it is considered a shell script [41]. Bash is frequently used by Linux administrators to perform tasks on a Linux system. These tasks include checking system health. While DDRIS allows this task to be performed by users through the dashboard interface, it is important to note that shell scripts were utilized through the deployment process of DDRIS. All system administrative tasks to improve performance were completed through `bash`, including but not limited to interrupt binding, and Maximum Transmission Unit (MTU) changes.

2.10.13 Linux Service

A Linux service, also known as a daemon, is a system application that runs in the background or is waiting to be used [43]. All daemons on a Linux system are managed through `systemd`, a software suite that controls other processes started after boot. Services are used commonly in System Administration to allow for easy control and supervision of processes and parallel job execution. This is particularly useful for deploying background processes and scheduling complex tasks. By creating `.service` files to handle deployment, redeployment of DDRIS requires less manual configuration. Instead, system files can be imported with Ansible, defined futher in Subsection 2.10.15, and enabled on boot. Additionally, several processes can be more easily maintained on DDRIS.

2.10.14 Cron

`cron` is a time-based scheduler available on Linux distributions [45]. Tasks deployed using `crontab`, known as jobs, are capable of setting reoccurring events on a system by time, date, or interval. `cron` jobs are utilized throughout DDRIS to schedule events. Through the use of this tool, services, including Flask, are not required to run at all times and hurt performance. Instead, Flask and services pulling information from

Flask are scheduled to run within time periods that experience low system usage.

2.10.15 Ansible

Ansible is an open source tool utilized by DevOps engineers to automate information technology work in the deployment of new systems, including software package installation, application deployment and server configuration, according to the practices of infrastructure as code [65]. This solution utilizes push-based configuration—which features no monitoring agents on remote hosts and results in less overhead on deployed systems after configuration when compared to pull-based configuration software, such as Puppet [62]. Management scripts, called playbooks, are built on top of "yet another markup language" (YAML), further defined in Subsection 2.10.16, to be easily readable. Ansible converts these playbooks to Python scripts and uses the Secure Shell Protocol (SSH) to push these changes to remote hosts in parallel [37]. Ansible is utilized in the deployment of DDRIS DTNs and PerfSONAR nodes. This configuration management tool was chosen because it would not be installed on these devices and frequent redeployment is a possibility with systems on Science DMZ's.

2.10.16 YAML

YAML is a data serialization language popular for use as configuration files [10]. It takes inspiration from datatypes present in popular programming languages such as Python and Javascript. This allows for the use of dictionaries and lists within configuration files. YAML is utilized in DDRIS as the primary language for all configuration files for production code. This is due to its ability to specify complex configurations and settings simply in an easily readable format. This simplicity but robust capabilities aids in the quick deployment of new systems on DDRIS by allowing for configuration with no modification to production code. DDRIS also utilizes YAML in Ansible playbook formation.

Chapter 3

Problem Statement and Proposed Solution

This chapter describes the problem in detail and outlines the solution proposed. Section 3.1 provides an in depth analysis of the current classic university campus model for handling data driven research, including the challenges unique to this model. Section 3.2 describes an alternative model to address these issues. This includes an overview of the users affected and their needs as well as the primary components of this architecture solution. Each of these primary components are designed to meet the specifications required by users. Additionally, a brief description of the results of this architecture in action is addressed.

3.1 Problem Statement

A common campus model for data transfer relies heavily on the individual researcher or department to configure machines for use. This model dictates that cyberinfrastructure is to be decentralized. A lab or department is responsible for buying, deploying, and updating technology as needed. This includes servers, technologies, and storage. Some cyberinfrastructure services are traditionally provided centrally, for example HPC or general purpose data storage (at UNR, these are the Pronghorn HPC cluster and the Rosalind data storage system). However, handling movement of data in or out of these centralized systems is still the responsibility of the researcher. While this approach has its merits, it makes collaboration between departments and institutions more difficult.

Key downsides of technology de-centralization include uniformity of infrastructure, access and security standards, duplication of management effort, and reliability, to name a few. With this comes the issue of interoperability, connectivity, and performance across systems. Researchers need to share and manage data with other departments and organizations. Some of these data may be local to a lab server, personal endpoint, or on a campus centralized system. Solutions for better data transfer management need to be explore, due to the lack of available systems, guidelines, or services across the UNR campus. Lab or department level solutions require an amount of computer knowledge that should not be expected of all individuals and may involve a learning curve. This time, exploring options and learning solutions can otherwise be spent by a researcher on their own research and actively collaborating, and increases the risk of constant "reinventing the wheel" across campus.

This issue is exacerbated by moderns trends associated with data sizes, formats, volumes, and velocities [44]. Most data transfer solutions require personnel to monitor transfers actively for errors or maintain an open terminal. For smaller transfers, these may finish within a day, but for larger transfers a researcher may feel uncomfortable leaving the transfer unattended overnight.

3.2 Proposed Solution

To address these issues, this thesis presents the Data Driven Research Infrastructure Systems (DDRIS), a collection of hardware- and software-based cyberinfrastructure architecture solutions developed for the University of Nevada, Reno, but conceptually applicable to all similarly-sized research institutions. DDRIS is designed at a campus scale with modular deployments available for smaller scale usage, including within a lab, to connect to the overarching infrastructure centered around a Kubernetes cluster to promote high availability of DDRIS.

There are two main user groups of DDRIS to consider: the researchers, and the administrative professionals responsible for the upkeep of these systems. First, the researcher's needs are to be considered. A researcher requires a fast, easy to learn solution that requires little oversight. Data is to reach its destination and maintain integrity, and the researcher should have access to basic statistics regarding the transfer. These statistics include the following: transfer speeds, amount transferred, warnings and errors in transfers, and a notice of success. Next, the needs of administrative professionals need to be considered. Systems require simple initialization and maintenance. Problems should be quickly recognizable through historical metadata. There should be user accountability in place to protect systems.

The three core user-facing components of DDRIS are designed to handle each of these aspects. These include the following: DTNs configured with Globus as Managed Endpoints to handle data transfers, an on-campus PerfSONAR mesh to handle network telemetry, and a dashboard to display historical data clearly and concisely. To address the requirement that researchers need a fast solution, each DTN was modified to increase performance. Compared to baseline data transfers, there has been a significant increase in performance that will be further explored in Chapter 8. Support for transfer speeds are also provided through network telemetry by allowing for a more proactive response to bandwidth congestion and errors. All of this is to be displayed on the dashboard alongside DTN health and transfer metadata to further proactive responses. The DTNs also provide an easy to learn solution through utilizing Globus Online, a data transfer application developed at the University of Chicago. This software supplies a simple Graphical User Interface to make transfers and access management easy for campus researchers. This software is described in Chapter 6 as a training manual to be utilized within the architectural model. Each of these three core components include sub-components that handle data flow, storage, and standardization.

Chapter 4

Hardware Solution Implementation

This chapter presents an overview of all hardware configured as a part of DDRIS. Section 4.1 discusses the network at a high level and includes a high level network diagram to contextualize hardware pieces and their interactions. Section 4.2 provides an in depth summary of the Science DMZ DTNs. Section 4.3 describes the details of the DTN currently configured on the Research Computing subnet and the Kubernetes environment. Finally, Section 4.4 provides information on the PerfSONAR Network Telemetry Mesh.

4.1 Network Overview

The University of Nevada, Reno has connectivity through the Nevada System of Higher Education (NSHE) System Computing Services (SCS). SCS runs NevadaNet, shown in Figure 4.1, the Wide-Area Network through which NSHE organizations connect to the greater internet. NevadaNet primarily consists of fiber lines but includes leased connections as well. NevadaNet consists of two Dense Wavelength Division Multiplexing rings, North Ring and South Ring. These rings serve as a solution to increasing fiber-optics performance and reliability where the University of Nevada Reno, where DDRIS is hosted, is a spur off of the North Ring. The distribution of leased lines and fiber lines owned by SCS can be seen in Figure 4.2. In the context of DDRIS, once data reaches NevadaNet it is considered to be on the external internet and is beyond the scope of this thesis.

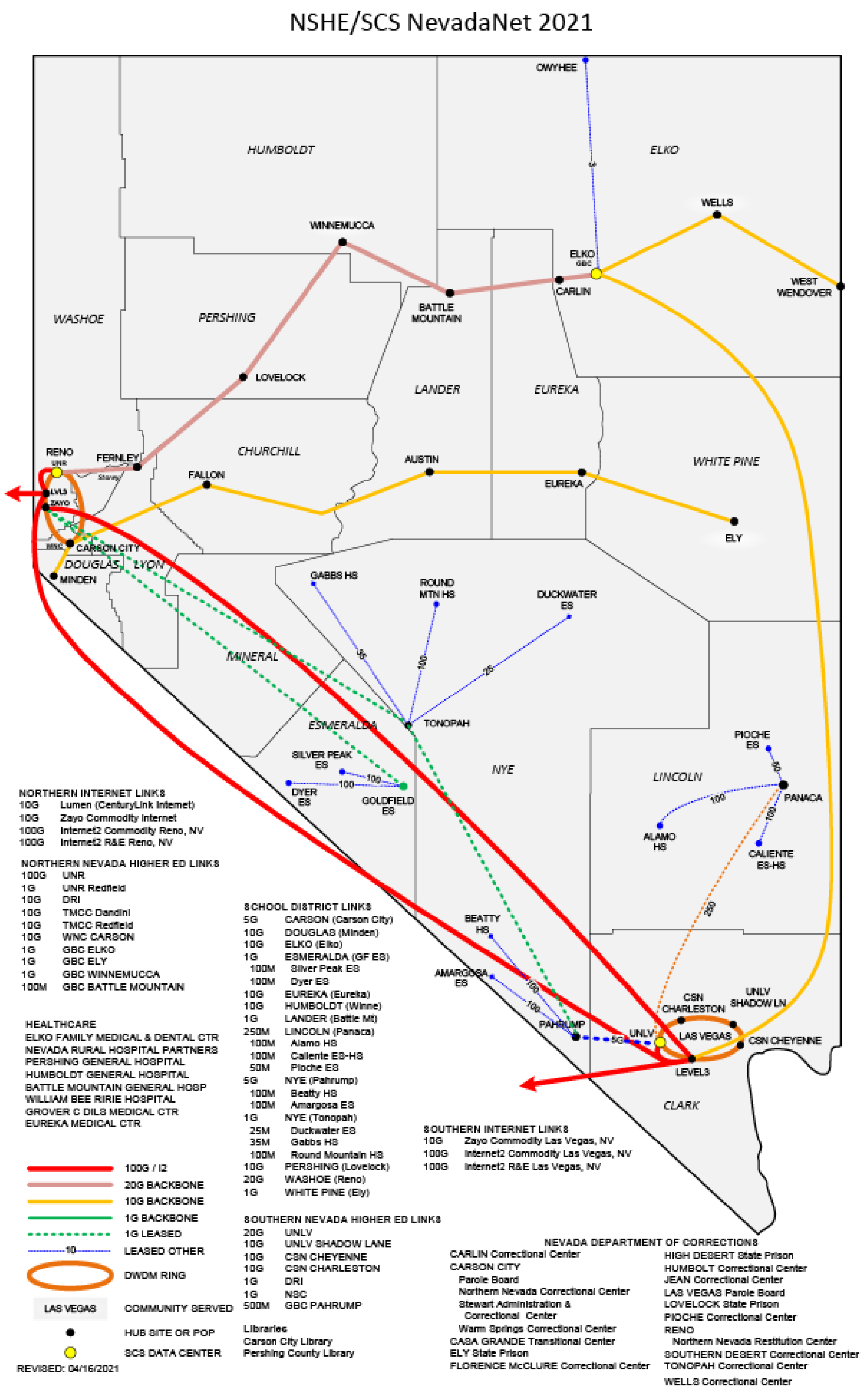


Figure 4.1: A network diagram showing the components of Nevada Net and how University campuses in Nevada connect to the outside internet [54].

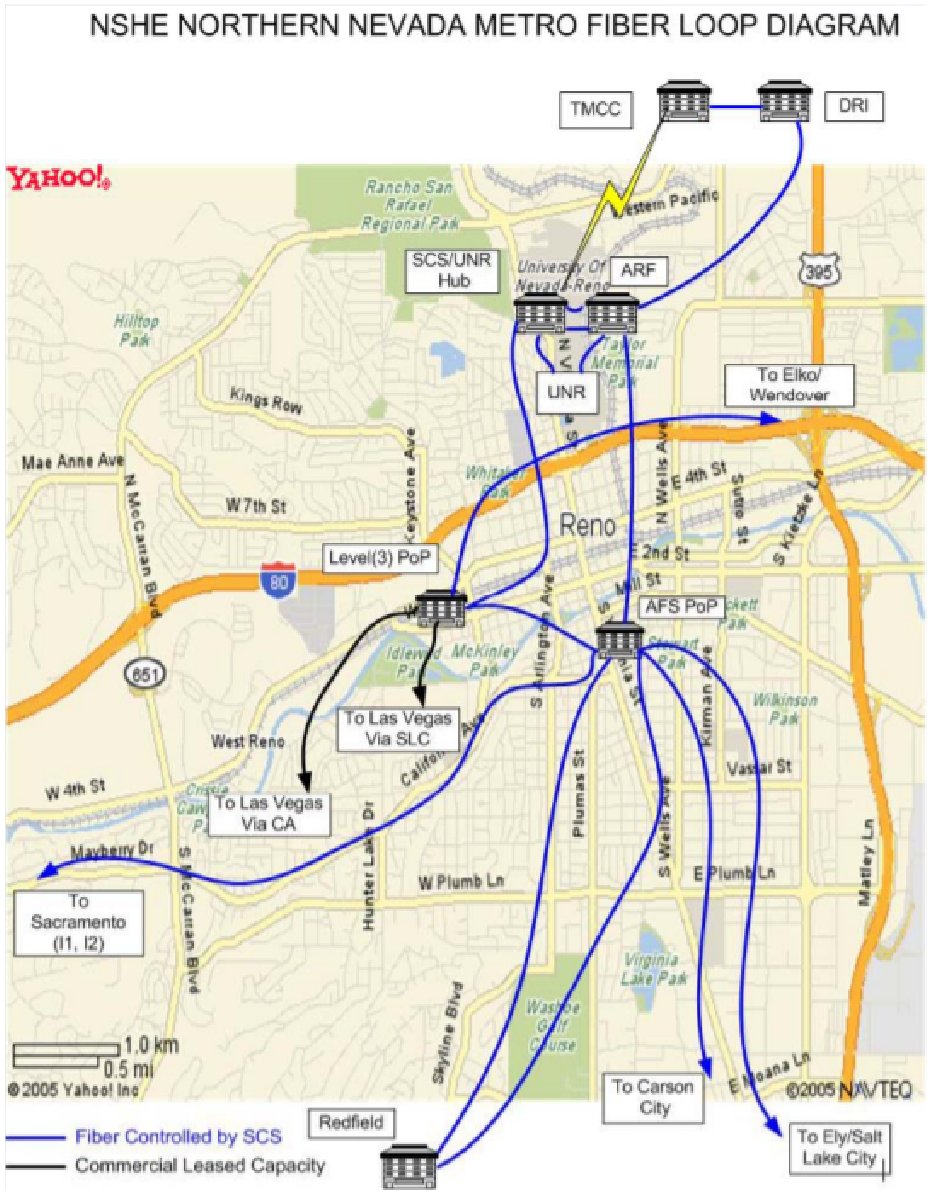


Figure 4.2: A network diagram showing the Northern ring of Nevada Net [34].

DDRIS exists on the University of Nevada, Reno internal campus network. An overview of the configured hardware in relation to the campus network can be seen in Figure 4.3. This shows the relationship and connection between network pieces and systems outlined in this section without security zones. It should be noted that this current UNR campus network topology, and the various locations of research-facing equipment in security space is still not ideal for research workflows, and further development of the campus network for research remains necessary. A high-level, scalable systems model will be provided further in Chapter 5.

The network includes four primary zones involved in DDRIS: the Science DMZ, a Research Computing (RC) subnet, the Field network, and a simplified network region labeled “Campus LAN”. The Science DMZ serves as a fast, easy to access collection of machines dedicated to performance capable of achieving higher-performance by limiting the use of firewalls as defined in Section 2.1. Implementation of DDRIS was completed within fixed networking parameters where security zones were defined through virtual local area networks (VLAN). This means, the Science DMZ uses ports on the “RCDC CS Router” where the firewall is not enabled. However, this speed is traded off with less security hardware. Without the campus border firewall to protect these machines, they need to maintain strict, local rules and are to remain isolated from the rest of the campus network. The only way for researchers to interact with these devices will be through the use of Globus. Machines on the Science DMZ include two DTNs (DTN-S and DTN-F) and a single dedicated PerfSONAR box.

The RC subnet trades performance for increased security and connectivity with the rest of campus. Because the machines belonging to the RC subnet exist behind the campus border firewall, they are permitted direct connection with the Campus subnet. This will allow for network mounting of other university machines. However, Globus will perform significantly slower on machines available on this subnet. DDRIS utilizes three machines on this subnet. One is a DTN, named NCAR-DA-9, and the other two machines belong to a Kubernetes cluster. Additionally, NCAR-DA-9 has network mounts from the campus High-Performance Computing cluster, Pronghorn, which belongs on the RC subnet, and Rosalind, the campus Isilon storage solution.

The Field network is entirely separate from the RC subnet. Overlap between these subnets is expressed to showcase the use of one PerfSONAR testpoint shared between the RC subnet and Field. Field is utilized for in situ research. Within the context of DDRIS, PerfSONAR testpoints have been deployed in this area of the network to monitor network connectivity back to campus. The testpoint shared with

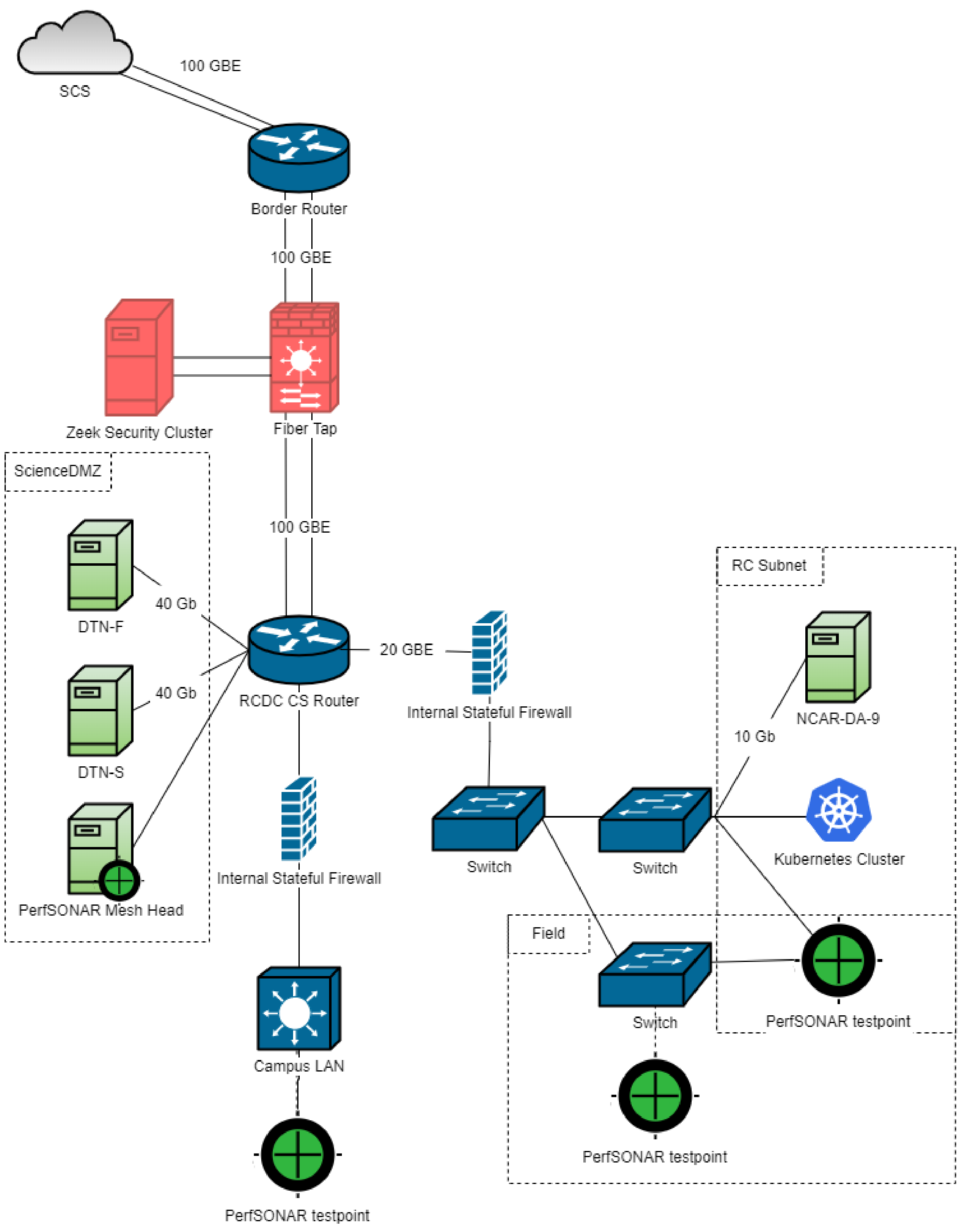


Figure 4.3: A network diagram showing on campus hardware systems involved in DDRIS. Dashed lines indicate a simplified connection to provide a general network location of a device.

the RC subnet, defined further in Section 4.4, serves as the destination for these network tests.

Lastly, the “Campus LAN” refers to any campus machines logically on campus that are utilized by end users directly, including lab spaces and storage systems. These systems are reliable but slow.

4.2 Science DMZ DTNs

Two DTNs were set-up on the campus Science DMZ. These DTNs are DTN-Flash (DTN-F) and DTN-Spin (DTN-S). Both of these campus border DTNs are designed around performance. Their purpose is to serve as a connection between researchers on campus and other institutions. These DTNs do not feature permanent local storage as they will only be utilized for data staging.

DTN-F features 15.36TB of flash memory. DTN-S features 112TB of spinning disk storage. DTN Settings for DTN-F do not include any form of Caching. Initial settings for DTN-S used a 4TB SSD as a cache, configured through ZFS. However, due to boot errors, this cache was later removed from DTN-S with an effect in performance that will be outlined in Chapter 8.

Both DTNs are configured with ZFS to provide a modern solution to data storage that easily scales as more drives are added. DTN Settings include software based RAID 6 emulation, RAID-Z2. This RAID level is typically reserved for read intensive disks above the size of 1TB. File compression is disabled on these systems.

Initially, jumbo frames were disabled on both DTNs; however, the Juniper switch had jumbo frames enabled. Because this can affect transfer speeds, both servers were later configured with jumbo frames enabled.

The PCI slot for the 40Gbps NIC in both servers is only connected to one CPU socket. This forces communication between sockets along the QuickPath Interconnect (QPI) bus in order to access the NIC from the other socket. Effective interrupt binding ensures that cores utilized in Globus transfers are connected to the correct socket. This safeguards against large performance penalties caused by bottle-necking along the QPI bus [24]. For both DTNs, the Linux network interface enp94s0 is connected to socket 0 and cores 0-11. A bash script was written to force all processes for enp94s0 to run on cores 0-11. To avoid errors and reduce future maintenance for the DTNs, the irqbalancer was left enabled. Instead, all irqs for i40e-enp94s0-RxTx-slice are banned interrupts from the irqbalancer, and cores 0-11 have been established as banned CPUs. This will allow the irqbalancer to continue to affect all other interrupts, but assign them to cores 12-23 across the QPI bus.

These DTNs are connected directly to a Juniper 10002 Research router with a 40Gbe fiber line. Their next hop is at the Border Router for campus at 100Gbe. For the security of other University systems, the Science DMZ is isolated from the rest of the network. Due to this, direct communication between these DTNs and the Kubernetes environment require other tools, including Prometheus, that will be addressed in Chapter 5.

4.3 Research Computing Systems

One DTN, NCAR-DA-9, was also established on the Research Computing (RC) subnet. It acts as a campus internal access point for users to view and manage their data. NCAR-DA-9 also allows internal collaboration within lab spaces. NCAR-DA-9 has 18TB of local storage and will have network mount points through NFS to other University systems, such as the HPC cluster, Pronghorn, and centralized campus data storage in its final iteration.

NCAR-DA-9 features hardware-based RAID 5 to maximize temporary storage. The decision was made to increase local storage at the cost of the extra protection of RAID 6 because this DTN is mostly meant to feature network mount points with nonessential local storage. This solution allows more storage to users as the system scales upwards, but essential data will be protected by remaining off machine.

Because hardware RAID is being utilized as opposed to software based RAID emulation, NCAR-DA-9 is configured with ext4 as opposed to ZFS. The primary appeal of ZFS is the raid emulation, which makes it unnecessary for the NCAR-DA-9 device. Instead ext4 was utilized because it is friendly to data recovery with decent performance speeds balanced by its reliability and preexisting affinity for ext4 within the University's Office of Information Technology. Additionally, more information is widely available for tuning DTNs with ext4.

NCAR-DA-9 is only accessible to campus users and is protected by campus firewalls. Due to this extra security, it is able to freely communicate with university systems and can be utilized for a mount point for other data locations. However, because it is behind campus security systems, it has significantly slower performance than either Science DMZ system.

The Kubernetes environment also exists on the RC subnet alongside NCAR-DA-9. It acts as a server to host a Postgres database, the Grafana dashboard, and listeners for Prometheus, the log exporter, and the PerfSONAR central measurement archive. Because this cluster belongs on the RC subnet, it is capable of direct communication with NCAR-DA-9, but it is isolated from Science DMZ systems.

The cluster is configured over two machines with three master nodes with five workers each. They communicate with one another through a single load balancer. The first machine has 16TB of storage and the second has 8TB. This solution was used due to scalability and availability of the environment. The entire cluster is utilizing CephFS. This is due to the scalability of CephFS and its reliability as a distributed filesystem.

4.4 Network Telemetry Mesh

The network behind a data transfer is as important as the DTNs involved, and as such, maintaining a healthy network is critical for data transfers. To collect diagnostic data on the health of the campus network, PerfSONAR was utilized to create a network telemetry mesh. Mesh operations are handled by the Measurement Archive, hosted on Kubernetes, and the mesh head, a mesh member responsible for hosting the mesh configuration and dashboard. The Measurement Archive hosts two databases. The Cassandra database hosted on the Measurement Archive holds network test results for the MadDash dashboard. The second hosted database is a Postgres database to hold mesh configuration metadata.

The mesh consists of five dedicated members and a single workstation as a proof of concept. The proof-of-concept mesh members include: SciDMZ-Perf, SCS-Perf, Roamer-Perf, NCAR-DA-11, Cavehill, and a campus workstation. SciDMZ-Perf is a dedicated PerfSONAR node on the Science DMZ near the DTNs and serves as the mesh head. SCS-Perf is a dedicated node in a SCS datacenter within the UNR network. Roamer-Perf is a portable Flash I/O Network Appliance designed to be moved around campus to run diagnostic tests within a lab as necessary. NCAR-DA-11 serves as a testpoint between field deployments and the campus network, and Cavehill is a field deployment.

All mesh members are initialized through Ansible and automatically updated with new mesh configurations when a member is added.

Chapter 5

Software Design

In this chapter, the software design specifications of subsystems involved in DDRIS will be discussed in detail. Section 5.1 will cover requirements. This includes technical requirements that describe system features and the nonfunctional requirements that describe system complaints. Section 5.2 details the use cases of the DDRIS dashboard with a diagram and a brief overview. Section 5.3 describes the architectural design of these systems. This entails high-level diagrams to describe the systems that are in place, system component diagrams, and application activity diagrams. Section 5.4 describes the intended workflow for researchers on this new infrastructure and on-boarding procedures through activity diagrams.

5.1 Requirements

Dashboard support was designed with three types of requirements in mind. These include base functionality requirements, extended functionality, and constraints. Base and extended functionality are discussed as the system's functional requirements. System constraints are addressed as the nonfunctional requirements. Both are discussed in this section.

5.1.1 Functional Requirements

Each system involved needed to fit a number of parameters. These parameters depend on the intended interaction and where they belong within the data pipeline. All data in this section relates to the dashboard and exporters that control the data pipeline to the dashboard. It will only be viewed and analyzed by system administration professionals and will not be visible to end users.

Because the dashboard is only accessible by system administration staff within the University, it will hold more sensitive information that would otherwise not be shared with a general user. For example, the dashboard should show the files update by each user and what transfer they belong to. This should not be available to everyone because collection information should otherwise only be visible to members of the collection. Additionally, system information, such as temperature, is not necessary to people outside professionals responsible for the health of the DTNs.

All functional requirements of DDRIS are listed in Table 5.1 and belong to one of three tiers. Tier one requirements are needed for basic functionality. All tier one requirements are completed. Tier two requirements include those that are considered desirable but were unnecessary at a base level for the dashboard to be completed. And, tier three requirements include stretch goals, including future development ideas to be implemented at a later time.

Table 5.1: List of functional requirements required by the dashboard.

Requirement ID	**Level**	**Description**
FR1	1	The dashboard will show historical transfer speed data
FR2	1	The dashboard will allow the user to filter time-series data based on transfer
FR3	1	The dashboard will allow the user to filter based on time and date
FR4	1	The dashboard will allow the user the zoom in on transfer speeds
FR5	1	The dashboard will have password authentication
FR6	1	The dashboard will limit editing functionality only to admin accounts
FR7	1	The dashboard will allow the user to view what files were affected
FR8	1	The dashboard will allow the user to view what endpoint was interfaced with

Continued on next page

Table 5.1 – continued from previous page

Requirement ID	Level	Description
FR9	1	The dashboard will allow the user to download data as a CSV
FR10	1	The dashboard will allow the user to view inter-facing IPs on a map
FR11	1	The dashboard will allow the user the identity that initiated the transfer
FR12	1	The dashboard will allow the user the local user account used to read/write
FR13	1	The dashboard will allow administrators to view all users with access
FR14	1	The dashboard will only be accessible through the University's network
FR15	1	The dashboard will display usage of the database
FR16	1	The dashboard will alert the administrator if file system usage meets or exceeds 80%
FR17	1	The dashboard will alert the administrator if database usage reaches 80% of capacity
FR18	1	The dashboard will alert the administrator if CPU temperature meets or exceeds 50 degrees Celsius
FR19	2	The dashboard will allow the user to view system health for campus DTNs
FR20	2	The dashboard will allow the user to view campus network telemetry
FR21	2	The dashboard will allow the user to view histori-cal network health
FR22	3	Kubernetes will automatically archive older histor-ical data on a storage database
FR23	3	The dashboard will provide users the option to browse archived data

5.1.2 Nonfunctional Requirements

In addition to these functional requirements, there are a number of constraints to DDRIS. These nonfunctional requirements total to fewer than half of the previous functional requirements. Each of the eight constraints are displayed in Table 5.2. The first requirement states that the UI of the application will be written through queries in PostgreSQL and PromQL. The second requirement constrains the system to only exporting log table data once a night as a scheduled job on each machine.

The third requirement indicates that the Globus Transfer API will be utilized to provide extra information regarding transfers. Requirement four indicates that the log exporter will be written using the Python programming language. Requirement five asserts that the custom Prometheus exporter will also be written using the Python coding language. The sixth requirement states that all timeseries data, including transfer speeds and system health, will be exported using the Prometheus service. The seventh requirement indicates that the dashboard will be created utilizing Grafana. The eighth requirement states that the Postgres database, Grafana dashboard, and Prometheus server will exist on the Kubernetes environment. This means extensive containerization will be utilized as all Kubernetes platforms will use Docker.

Table 5.2: List of nonfunctional requirements required by the dashboard.

Requirement ID	Description
NFR1	Dashboard queries will be written in PostgreSQL and PromQL
NFR2	Table data will be exported using a nightly cron-job
NFR3	Table data will be supplemented using the Globus Transfer API
NFR4	Log data collection will be written in Python
NFR5	Timeseries data will be exported using Python
NFR6	Timeseries data will be exported through Prometheus
NFR7	The dashboard user interface will be created using Grafana
NFR8	The Postgres database, Grafana dashboard, and Prometheus server will exist on the Kubernetes environment
NFR9	The dashboard will update in near real-time
NFR10	Science DMZ data will only be accessible through secret keys
NFR11	Science DMZ DTNs will host their table data using a flask application

5.2 Use Case Modeling

In this section, use case modeling for DDRIS is covered and the following descriptions are presented in Figure 5.1. In this context, the use cases presented are divided among three actors and discussed briefly.

The first actor is the Sysadmin, or System Administrator. This role includes any personnel responsible for maintaining and onboarding researchers on DDRIS. An administrator has the ability to monitor traffic through two methods. First, they can view recent transfers through the Globus Online GUI and their current or final average transfer speed. They also have the ability to view historical data graphed, including instantaneous transfer speeds through the dashboard interface. This dashboard also provides more detailed transfer metadata, including the IP address the transfer was initiated from, who requested the transfer, what endpoints and collections the transfer was between, what files were modified, and what local user is reported as having modified the files. The Sysadmin also has the ability to monitor system health. Records can be accessed on the machine through command line, but important, historical data is also routinely reported through the dashboard for a more efficient visualization of changes in system health. A system administrator also has the ability to create new collections on a DTN to onboard more users. For testing purposes and in the event administrators have data to transfer, administrators may have their own departmental collections through the same onboarding proceedures researchers have. These system administrators then have the ability to perform their own transfer requests. The next use case for system administrators dictates that they have the ability to cancel transfer requests. This includes any transfer performed on a DTN, but this ability is only to be utilized in specific situations, including if notified of an unauthorized transfer. The Sysadmin also has the ability to manage members on a collection. They are provided the highest collection role of Administrator on every collection on DDRIS.

The second actor is the Quality Control Service, which consists of a series of services on Kubernetes. Like the Sysadmin, the Quality Control Service is capable of monitoring traffic and system health. Unlike the Sysadmin this is an automated process to collect data for the dashboard and compare against thresholds. Another use case for the Quality Control Service is to send alerts through email if a threshold is exceeded. The Quality Control Service also communicates directly with Globus. This is exemplified through its other two use cases. First, it is able to query Globus

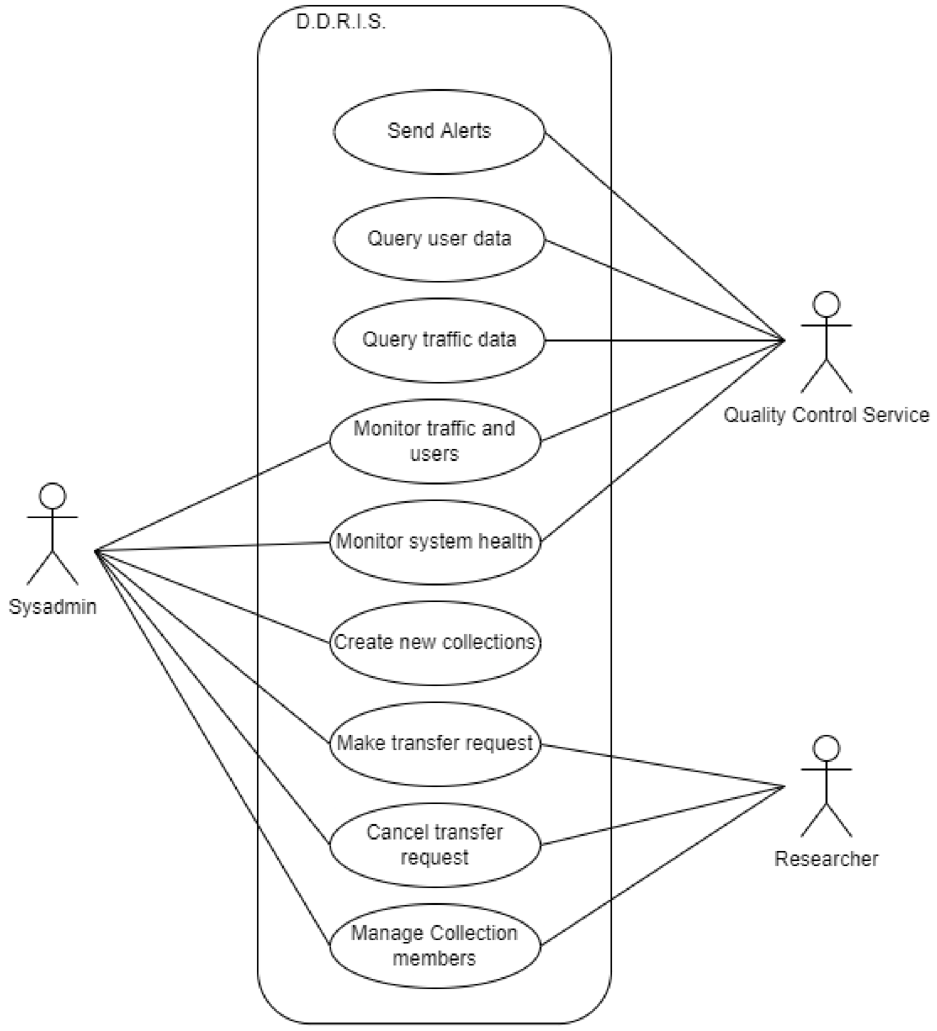


Figure 5.1: Use Case Diagram of DDRIS.

for user data when a user transfers with a monitored system. Lastly, the service queries traffic data from Globus to provide more detailed information about transfers on the machines.

The final actor is the Researcher or campus research lab. A researcher has the ability to transfer to move their data utilizing DDRIS. This means they have the ability to initiate transfer requests on collections they are provided access to and manage these transfers through the ability to cancel their own requests. They should only have the ability to cancel transfer requests to or from collections they own. These users are also able to manage access to their collections through the Globus

Online web interface. Researchers have the ability to add students or members of other organizations to their existing collections with role privileges including, Access Manager, Activity Monitor, Read-Write access, or Read only access. These roles provided through Globus are covered in more detail in Chapter 6.

5.3 Architecture

In this section, the main architecture of DDRIS is discussed in three main segments. The first segment, covered in Subsection 5.3.1, describes DDRIS at a high level with an aiding diagram. This serves as a brief overview of the systems involved to aid in contextualizing each piece of the system. The second segment, Subsection 5.3.2 describes the Prometheus exporters utilized by DDRIS. The third segment, Subsection 5.3.3, provides a more detailed overview of the application design. This includes a brief explanation of log exporter versions, component diagrams and activity diagrams. The fourth segment, Subsection 5.3.4, provides an in depth view of the databases designed for historical metadata storage. This segment is supplemented by an entity relationship diagram to show the layout of these databases.

5.3.1 High Level Design

DDRIS consists of nine primary components. These components are split into two contexts for readability, DDRIS Transfer and DDRIS Networking, but both contexts interact and serve the purpose of aiding data driven research. First, the components of DDRIS Transfer will be discussed. This will be followed by an overview of DDRIS Networking.

A high level representation of the components and primary subcomponents belonging to DDRIS Transfer is shown in Figure 5.2. At the center of DDRIS is the Kubernetes environment. Kubernetes is responsible for DDRIS health monitoring, reporting, and alerting. It has four subcomponents to achieve these goals, including the Prometheus Speed Exporter, Prometheus Server, Log Receiver, and Grafana. The Prometheus Speed Exporter is described in Subsection 5.3.2. The Prometheus

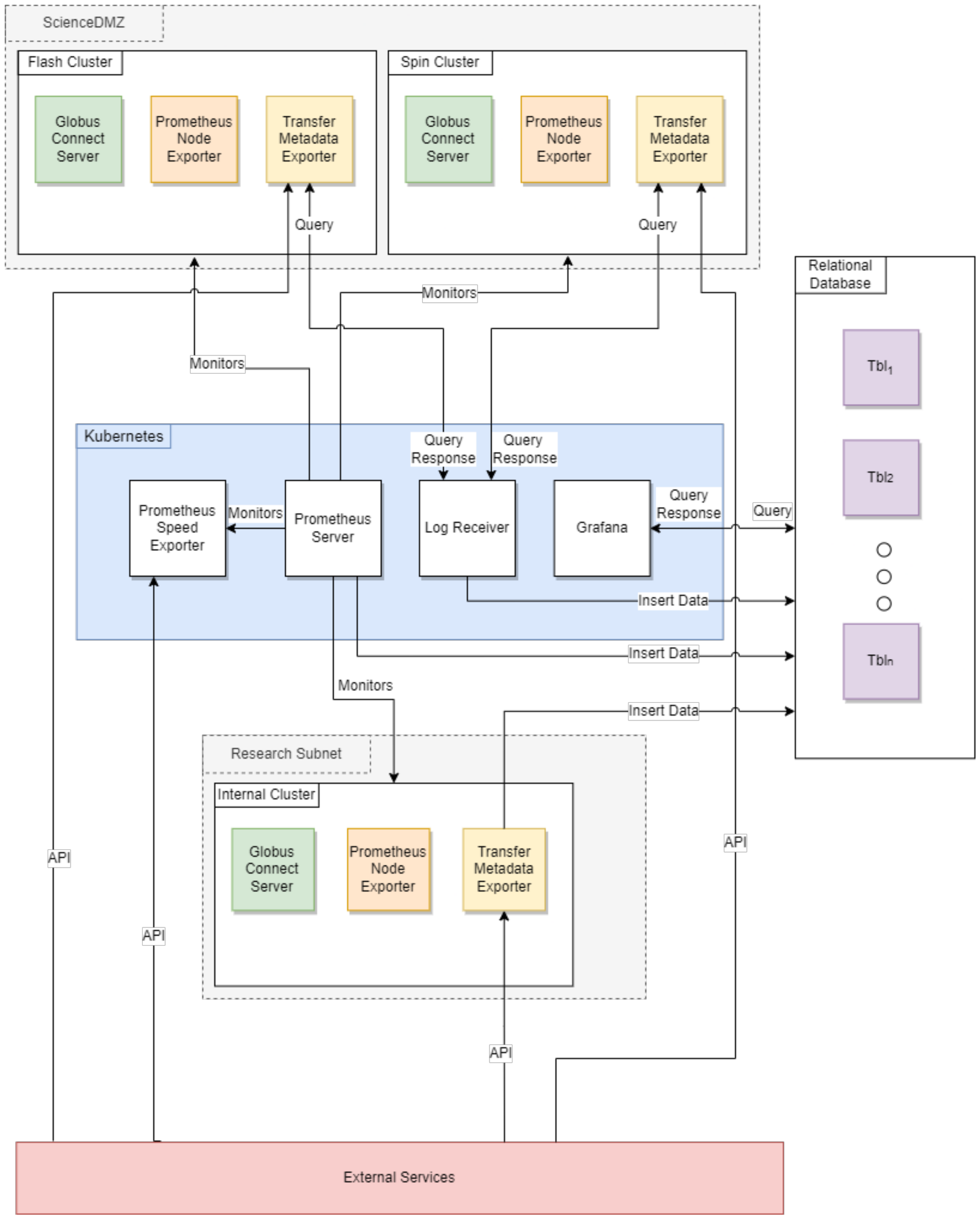


Figure 5.2: High Level Design of DDRIS software communications.

Server pulls records from Flash, Spin, and Internal every six hours to monitor the system health of each component. Records are pulled from the Prometheus Speed Exporter every 15 seconds to provide accurate historical data for data transfers. The Log Receiver queries log data from the Science DMZ components nightly at 03:00. This data is then stored in the Relational Database, another primary component that will be covered in Subsection 5.3.4. The Grafana dashboard queries the Relational Database, visualizes the response, and sends alerts when data thresholds are crossed. The third primary component is the Internal Cluster on the Research Subnet. The Internal Cluster is a hypothetical scalable model built on the current implementation of NCAR-DA-9 and is responsible for internal data transfers. The use of clusters in a more scalable final model is recommended by Foster et al. (2011) [29] due to the native Globus load balancer. Fourth and fifth components are the Flash Cluster and Spin Cluster respectively. These clusters make up the Science DMZ components. Flash and Spin, like the Internal Cluster, are hypothetical models built on the current implementations of DTN-S and DTN-F to allow for more onboarded researchers. The Science DMZ is responsible for external data transfers. Sixth is External Services. This consists of two parts that are both accesible through API calls. These components are Globus Services, and an open source IP to geolocation API.

DDRIS Networking is responsible for network telemetry and the health of systems responsible for collecting this data. A high level diagram of the components in DDRIS Networking is showing in Figure 5.3. As the center of DDRIS, Kubernetes is also at the center of DDRIS Networking for health monitoring, reporting, and alerting. As the central archive for PerfSONAR, Esmond receives test results and stores them in the seventh component, the Relational Results Database. Kubernetes also holds a copy of the mesh configuration in the Mesh Database. The Kubernetes Prometheus server and Grafana are the same across both diagrams, showing the role of both subcomponents in both contexts. While Prometheus is responsible for DTN monitoring, Prometheus also monitors the health of the Kubernetes environment itself, the PerfSONAR Mesh Host, and all mesh members. This data is inserted into the

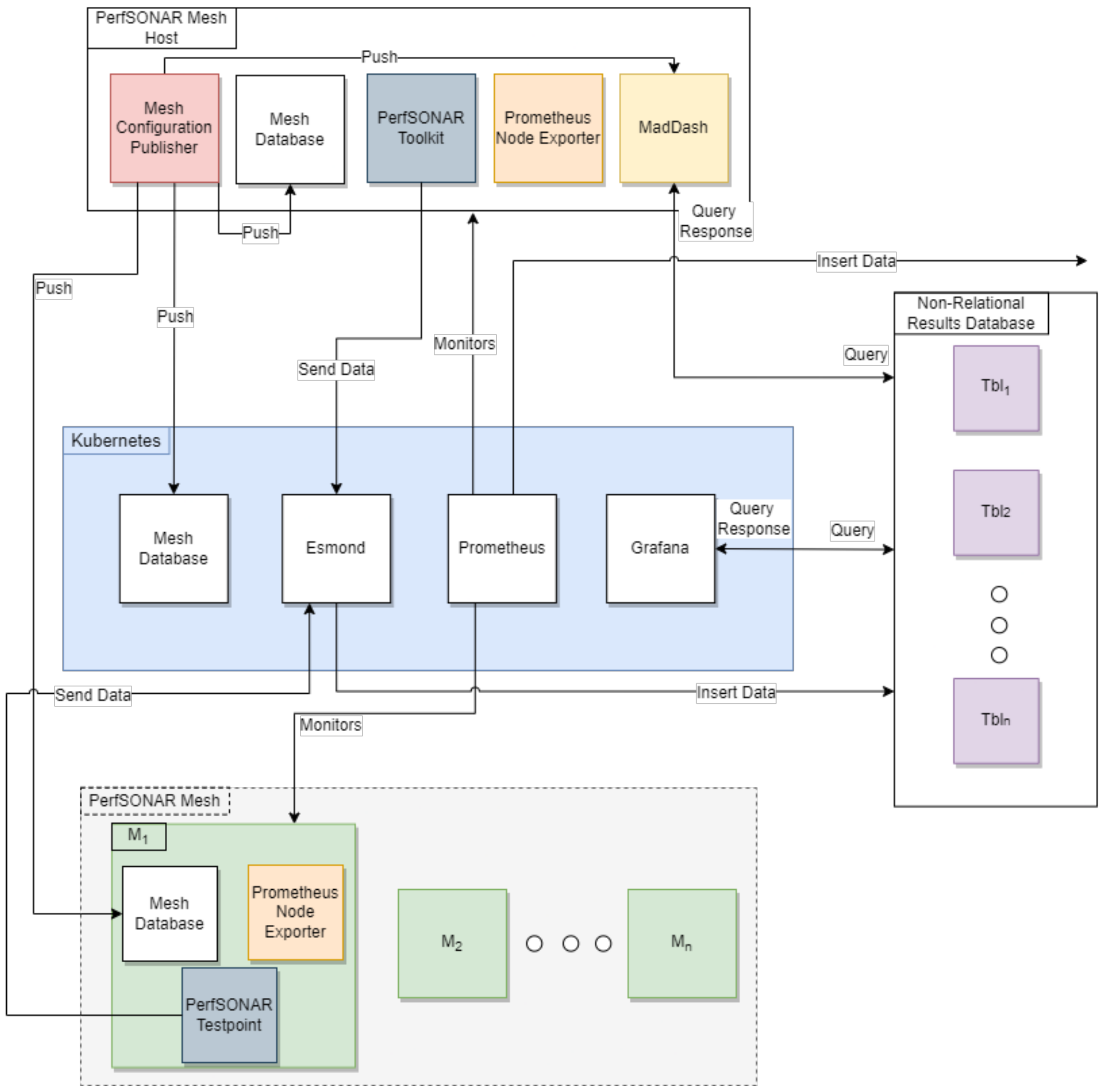


Figure 5.3: High Level Design of DDRIS software communications for PerfSONAR. Each of the members of the PerfSONAR mesh are denoted by M_y where y indicates the count. Each mesh member has DDRIS network telemetry connections identical to M_1. The Prometheus Server inserts data into the Relational Database identified in Figure 5.2.

same Relational Database as that defined previously. Grafana, in addition to pulling transfer and health statistics, pulls historical network telemetry results from the Relational Results Database for display. The eighth component is the PerfSONAR Mesh. This is responsible for running network telemetry tests as specified by the mesh configuration. This configuration information is stored in a local database. The ninth component is the PerfSONAR Mesh Host. While responsible for running tests like a mesh member, the host is also responsible for publishing the mesh configuration and displaying real-time data in MadDash.

5.3.2 Prometheus Exporters

Two classifications of Prometheus exporters exist within DDRIS, node exporters and transfer exporters. Node exporters exist on every DTN within DDRIS and are responsible for collecting system health information. This class of exporter was developed by Prometheus with an official release. The transfer exporter only has one instance. It runs on the Kubernetes environment and is responsible for timeseries metadata regarding Globus transfers. This class of exporter has been custom created for DDRIS.

The system surrounding the custom Prometheus exporter for Globus transfer speeds contains four primary components. A high-level representation of these systems is shown in Figure 5.4. The Prometheus Server orchestrates data exchange through pull-based reporting. This Server queries data from the exporter every 15 seconds for transfer speed metrics. The next piece is the Prometheus Exporter. Exporters are responsible for data collection and formatting data into appropriate metric types. This data collection is achieved through querying the third component, Globus Services, through the Globus Transfer API. This third component exists outside DDRIS as an official service provided by Globus and is the only component accessed through an external internet connection. It responds to the Prometheus Exporter with a Transfer Object that is processed by the Exporter. When the Prometheus Server receives this processed data, it connects to the fourth component, Postgres with TimescaleDB. The data is stored in this database to be queried by the DDRIS

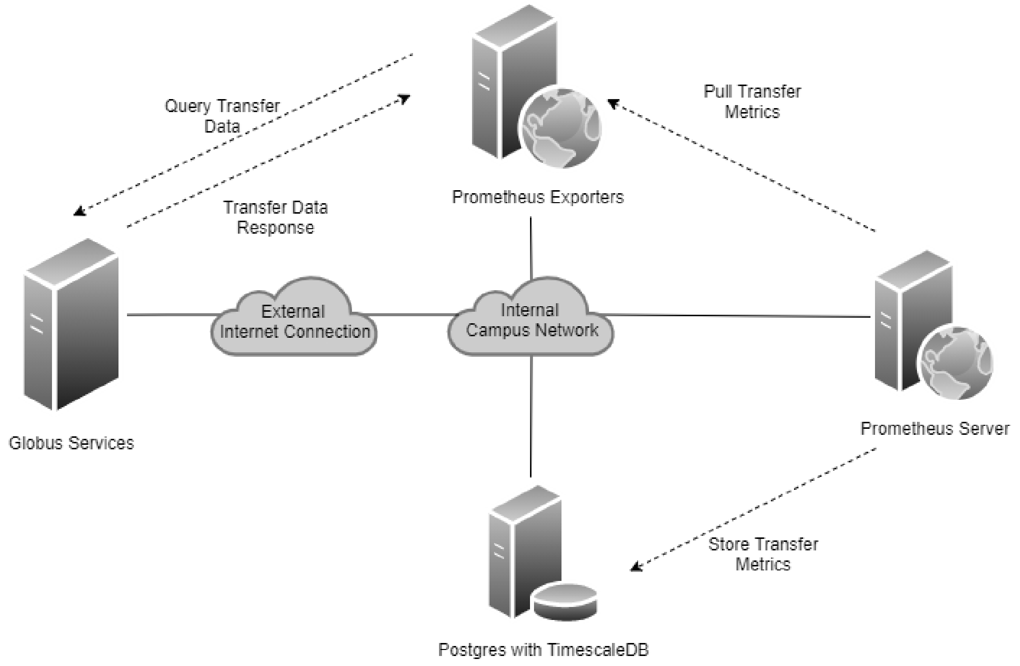


Figure 5.4: High Level Design of the Globus Speed Exporter for Prometheus.

dashboard later.

5.3.3 Application Design - Log Exporters

Due to existing networking standards, different pieces of hardware utilize different code versions and workflows dependent upon their subnet and security. First, the Science DMZ version of the transfer exporter will be discussed in detail. Then the version utilized on the Research Computing subnet by NCAR-DA-9 will briefly be covered with an emphasis on pieces that do not overlap with the Science DMZ version. The workflow of both versions will then be compared with one another.

The Science DMZ version of this application and its critical classes can be seen in Figure 5.5. The log exporter has six main components. These consist of the classes GlobusApp, APIData, SubmissionData, LogData, LocationData, and FlaskHost. More details of these components can be viewed in Table 5.3.

This version features no database connections on the DTN's side. Due to the isolated nature of Science DMZ systems, DTN-F and DTN-S are not permitted a

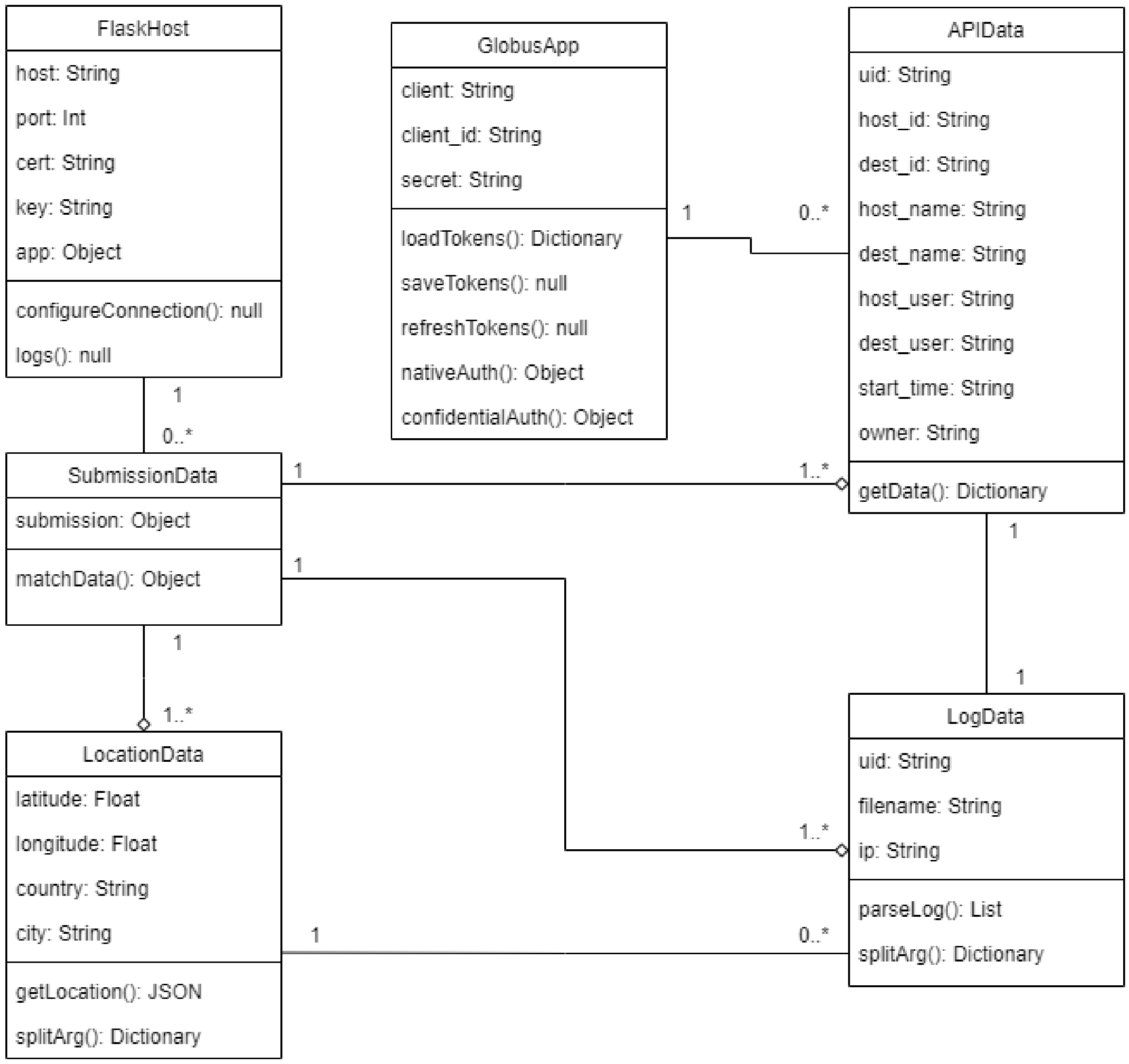


Figure 5.5: Class Diagram showing the relationship between critical classes in the Globus data exporter for Science DMZ systems. Hollow diamonds indicate aggregation between two classes. Lines without arrows represent association. 1s and 0s show cardinality such that 1 indicates exactly one, 0 indicated exactly zero, 1..* is one or more, and 0..* is zero or more. This notation is consistent with UML standards [76].

direct connection with the Kubernetes environment. An alternative workflow was implemented using a Flask server. Where, ideally, the application would connect directly to the database and push data, this version instead hosts all matched data on the local network. API, log, and geolocation data are matched first with no checks to the database to determine if the entries already exist. This operation is instead performed on the client side after pulling the data from the hosted web service. Exported data is posted as a JSON for ease of readability on the client side. To protect the security of the DTNs, the port the web service is running on is locked to only allow traffic from the Kubernetes environment and requires a certificate to connect.

This Flask solution is not ideal due to the overhead in running a Flask server on the machines but is required to maintain isolation from other systems. They will divert computation and throttle network connections on the machines. To address these problems, the time at which the Flask server is running is very limited. The application exists as a service on the machine that has been stopped. A local cronjob starts the service at 03:00 and stops the service at 04:00. This way only a single hour that is expected to have slower traffic is impacted by the exporter. By running the application for a full hour, multiple attempts to connect can be supported in the event of failure or potential local time differences.

Table 5.3: Class descriptions for classes featured in the Science DMZ version of the log exporter.

		Description
API Data		This class handles extracting and formatting data from the GlobusAPI.
	getData()	This function pulls a list of transfers from the GlobusAPI that were preformed on a single, designated endpoint. It then organizes this data into individual transfer instances.
GlobusAPP		This class handles connecting to and authenticating with the GlobusAPI.

Continued on next page

Table 5.3 – continued from previous page

		Description
	loadTokens()	This function loads authentication tokens from a local file.
	saveTokens()	This function saves tokens to a local file on its first run on a new system
	refreshTokens()	This function refreshes local tokens whenever a new authentication session starts.
	nativeAuth()	This function handles authentication for native Globus applications. This is useful during testing and is to be utilized until CILogon can be integrated with the application.
	confidentialAuth()	This function handles authentication for non-native Globus applications. This will be utilized when CILogon can be integrated with the application for increased security of the application
FlaskHost		This class handles posting data to the web service and its security
	configureConnection()	This function runs the flask application, specifying port, ip, and security
	logs()	Hosts a JSON containing an encrypted version of the data to export
LocationData		This class handles pulling data from an online resource to map an ip address to a physical location
	getLocation()	This function gets the location data from host api
	splitArg()	This function splits the location data to keep only the necessary pieces
LogData		This class handles extracting data from the local gridftp logs.
	parseLog()	This function pulls all data from the local machine's gridftp logs. This data is then organized into individual transfer instances
	splitArg()	This function is responsible for pulling important data out of the transfer instances for communication with other classes
SubmissionData		This class organizes data from all three sources into a single object.
	matchData()	Uses unique data in each class to match API, location, and log data to a single transfer

Hardware belonging to the RC subnet is protected by the University border firewall. For this reason, RC subnet systems are not isolated from the rest of the network. This allows for a more direct connection between exporters and the Kubernetes environment. This direct connection was utilized on NCAR-DA-9 to avoid the overhead that would be introduced by Flask.

Critical classes involved in the exporter are included in Figure 5.6. As opposed to the Science DMZ version, this features the DBConnection class instead of the FlaskHost class. This class is responsible for connection initialization and commits to the Postgres database. It is initialized utilizing an INI file to avoid locally storing authorization information and to create a more modular system. In the event that the database changes locations, the INI is to be updated instead of the code.

The details of this class can be seen in Table 5.4. This includes details of the functions defined.

Table 5.4: A description of the class "DBConnection". This class is specific to the RC subnet version of the exporter.

Class: DBConnection	Description: This class handles all database connections. This includes configuring the connection based on the configuration file, handling SQL statements, and closing connections.
configureConnection()	This function parses the INI file for relavent DB connection information. To avoid the use of plaintext password storage, it also handles decoding the password for authentication.
insertData()	This function temporarily opens a connection to the database and performs a SQL insert. It then closes the connection again. This is to avoid connections remaining open.

In addition to this change, comparisons with data already present in the database can occur within the application. This allows for only data that does not exist already to be handled prior to matching the data and limits the number of API calls from LocationData.

Both versions of the application follow a similar workflow. This workflow, shown

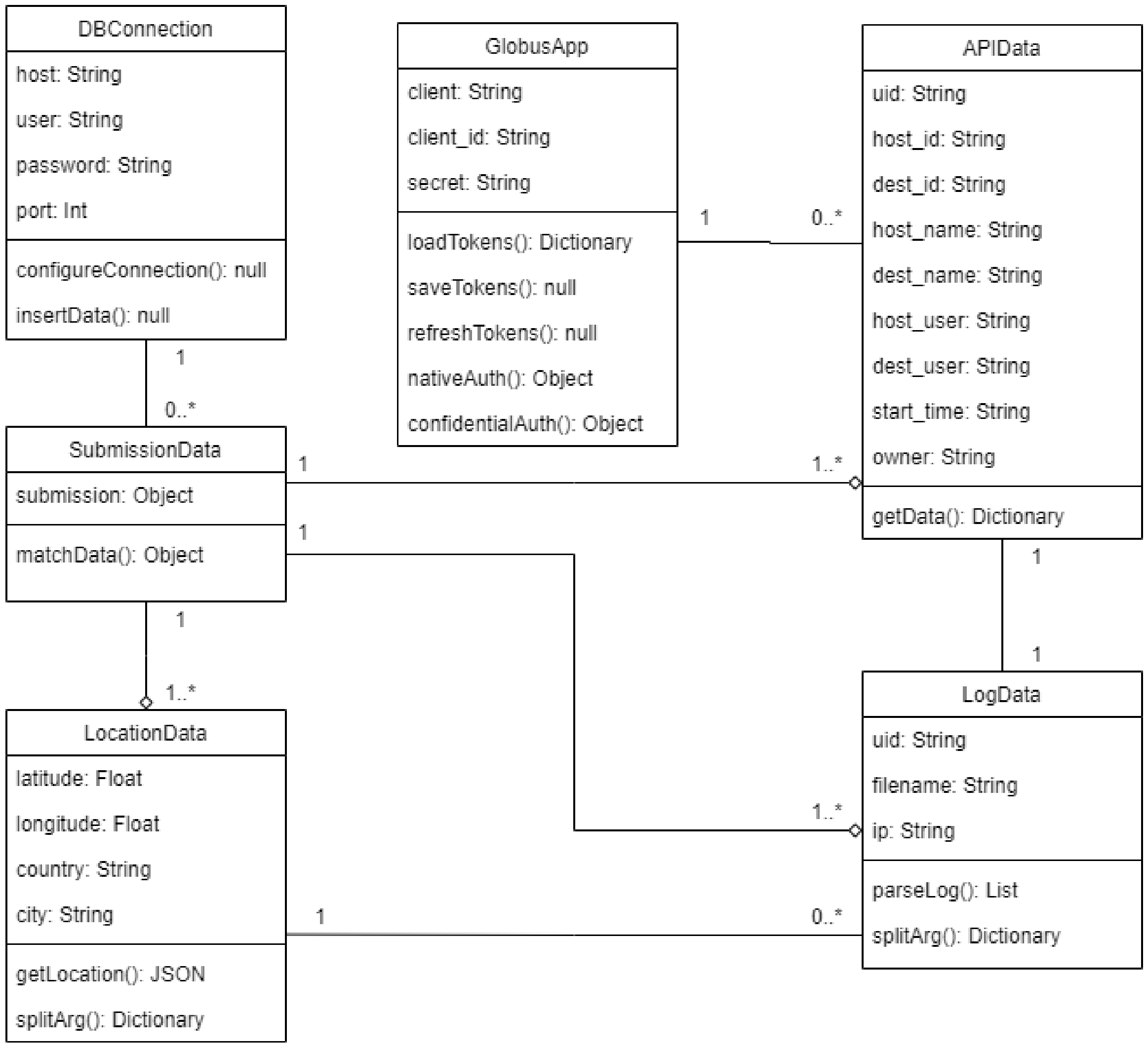


Figure 5.6: Class Diagram showing the relationship between critical classes in the Globus data exporter for NCAR-DA-9 and other future systems off of the Science DMZ.

in Figure 5.7, starts at 03:00 through a cron job. At this start time, a bash script is set to start a Linux service. The service starts the exporter that pulls data from the Globus Transfer API. Local log data is then pulled and parsed into transfer instances. This data serves to supplement API data. The first difference in the code then comes when handling this log data. Because the Science DMZ version cannot directly communicate with the database, it continues with all log data. By contrast, the RC version repeatedly queries the database. All entries already entered in the database are excluded from the list, leaving only new entries. Both versions then query location data and all three data sources are combined into a single object. The second difference is how the collection handles this data. As previously stated, the RC version pushes this data to the database. The Science DMZ version instead hosts this data and continues this workflow until the service is stopped. The service is set to stop at 04:00 through a second cron job.

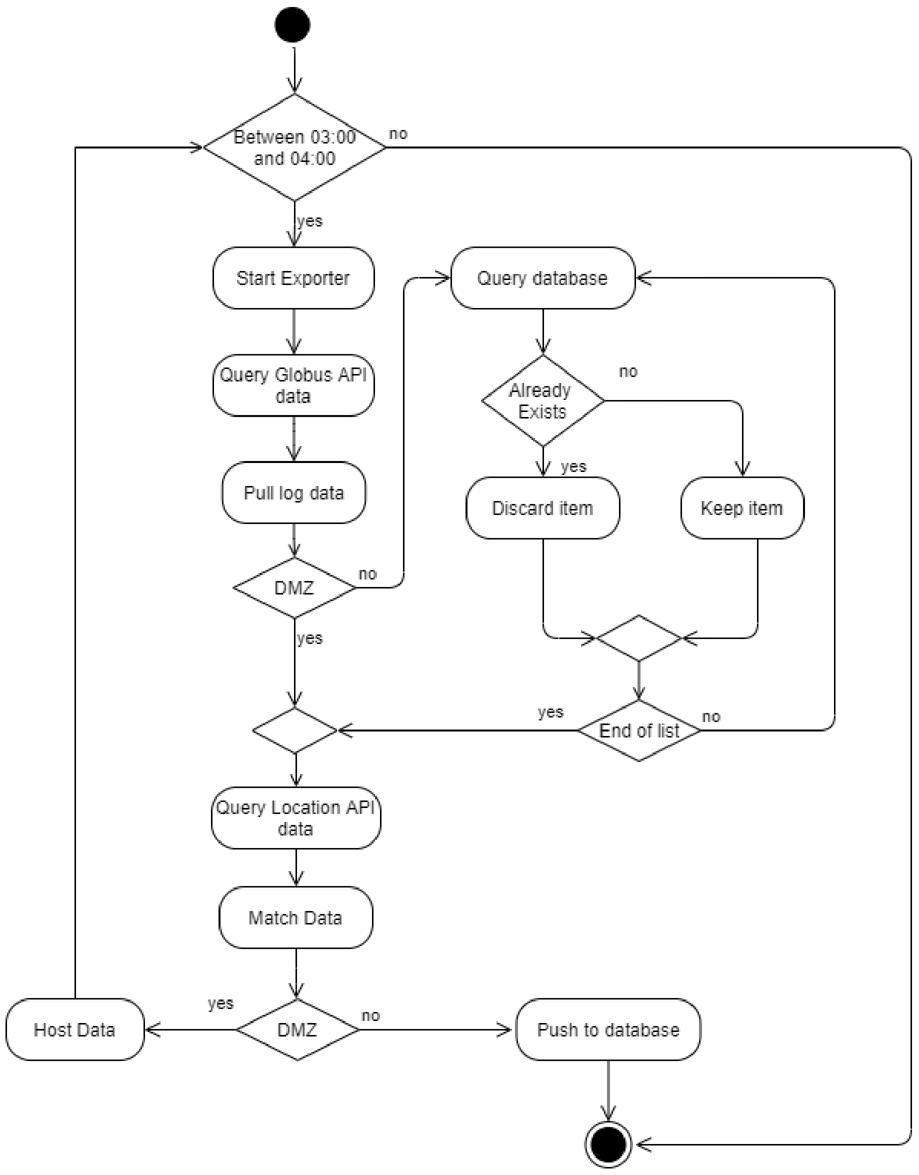


Figure 5.7: Activity diagram showing the general application workflow and differences between the Science DMZ and RC versions of the exporter.

5.3.4 Database Schema

The data collected by the log exporter and displayed on the dashboard is stored to maintain historical records. This database uses Postgres. As shown in Figure 5.8, the database is separated into eight tables: i) `TBL_GlobusUser`, ii) `TBL_LocalUser`, iii) `TBL_Location`, iv) `TBL_TransferData`, v) `TBL_Endpoint`, vi) `TBL_Collection`, vii) `transfercollectionbridge`, and viii) `subcollection`. An explanation of each table can be seen in Table 5.5. These tables are currently being implemented to increase scalability and performance of the systems.

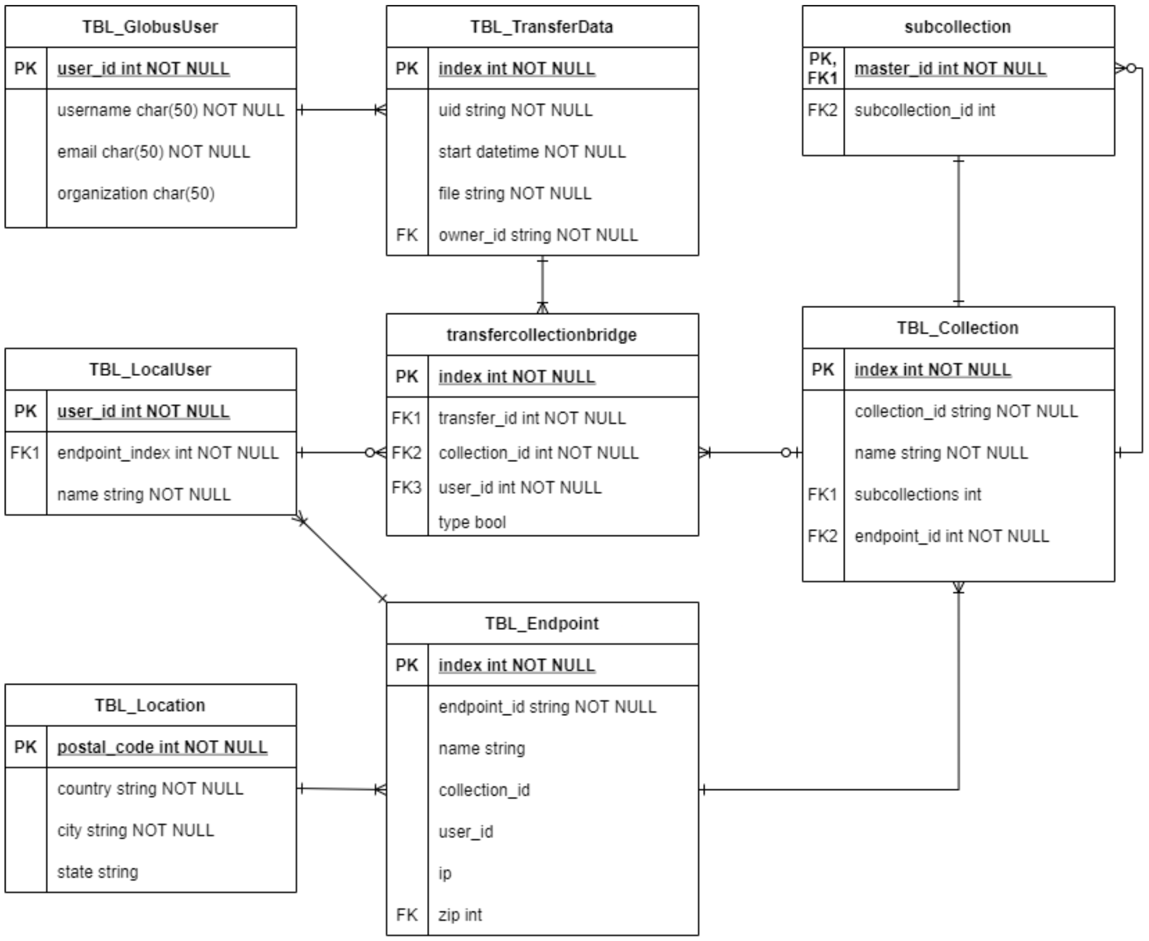


Figure 5.8: Entity Relationship Diagram of the Kubernetes table-based Metadata Storage.

A second database for DDRIS is dedicated to timeseries data. This data is everything collected by the Prometheus exporter, including system health information

Table 5.5: Description of tables included in the Kubernetes table-based Metadata Storage.

Table	Description
`TBL_GlobusUser`	Holds data about the Globus user who initiated the transfer
`TBL_LocalUser`	Holds the local user of each DTN endpoint
`TBL_Location`	Holds the location data of an endpoint
`TBL_TransferData`	Contains metadata about an individual transfer
`TBL_Endpoint`	Contains information regarding a DTN
`TBL_Collection`	Holds data about a Globus collection on a DTN
`transfercollectionbridge`	A bridge table that addresses the many-to-many relationship between `TBL_TransferData` and `TBL_Collection`. This table holds whether the endpoint is the source or destination as well as what local user is reported to have accessed data on the collection.
`subcollection`	A bridge table to hold a list of collections within a collection

and data transfer speeds. The second database utilizes TimescaleDB with Postgres. It includes two tables. The first table is `metric_labels`. This table is an indexed list of all Prometheus metrics. It directly relates to a second table, `metric_values`. The `metric_values` table holds all timeseries data of each Prometheus metric.

5.4 Workflow

Because DDRIS is new to our campus, a workflow needed to be established for proper utilization and the onboarding process. The user transfer workflow provides a standard for DTN usage. This will serve to aid researchers in utilizing the DTN that best suits their specific transfer. The onboarding workflow will streamline the process of adding more researchers to the DTNs to maximize the amount of time that they can spend collaborating. These are discussed in this section.

5.4.1 User Transfer Workflow

Previously, the workflow on campus for research data transfers was ad-hoc. Solutions still vary drastically between departments and labs with no standard, which is not conducive to collaboration, researcher's time, or repeatability. There would be differences in solutions, management, and infrastructure. This new workflow is meant to combine managed infrastructure into an easier, single-interface solution for data transfers, and made available to researchers across campus. It also allows the Office of Information Technology professionals to support and facilitate transfers.

Each DTN utilizes Globus Connect Server (GCS), which handles transfers for researchers with built in alerts. Globus also provides a simple user interface (UI) to view data and start transfers. An overview of the entire workflow can be shown in Figure 5.9.

Transfers internal to the institution have the following workflow:

- A PI accesses the Globus Online web portal
- The PI searches for NCAR-DA-9 and adds collaborators to their dedicated local scratch space. Collaborators are not to be provided access to the entire collection.
- If data is not immediately accessible through NCAR-DA-9, such as if the data is stored on their laptop, they will install Globus Connect Personal (GCP) on their lab machine.
- The PI stages data from storage or Pronghorn on NCAR-DA-9.
- A collaborator will login on the Globus website and locate NCAR-DA-9 under collections.
- The collaborating researcher selects the data they wish to copy and clicks start.
- The collaborating researcher posts data to the collection of the PI on NCAR-DA-9 to share data with the PI

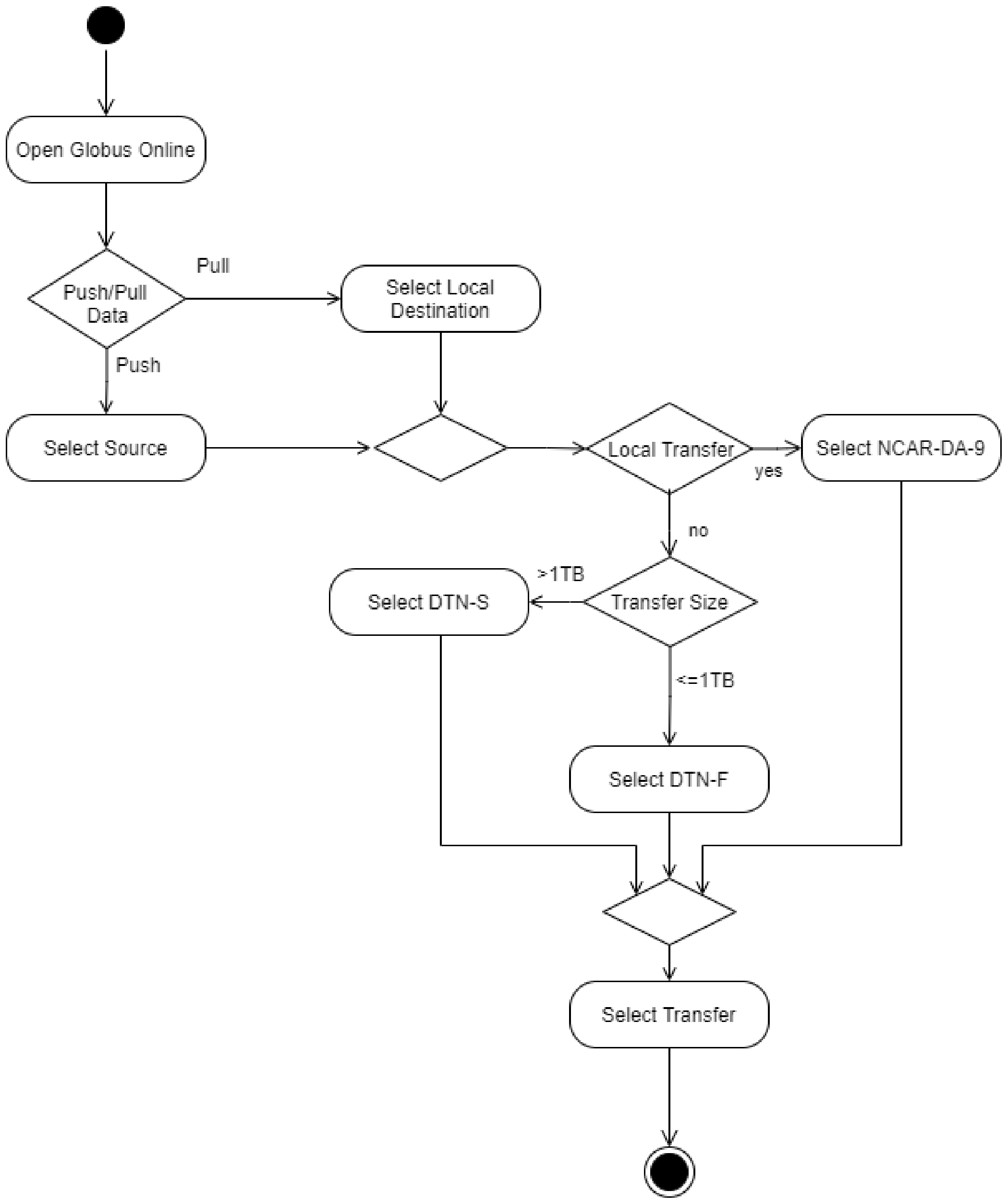


Figure 5.9: Activity Diagram showing the workflow for users utilizing the centralized data transfer solutions.

- The PI pulls data from the scratch space to a more permanent location, such as PH for processing or a local lab computer

In the event that the collaborator is not part of the University or the transfer is to an external machine, such as in the case of remote work, the workflow changes very little. If the transfer is over 1TB, the user will search for DTN-S on the Globus Online web portal. Their secondary endpoint will be where their data currently exists. This can be a lab machine, laptop, a machine connected to NCAR-DA-9, or another workstation. They will then initiate a transfer from this data location to DTN-S. The data staged on DTN-S will then be pulled from the machine to its desired location. Transfers under 1TB will instead utilize DTN-F. This DTN is designed to be the fastest option of the three but has the smallest amount of available space for data staging.

Researchers may use DTN-S or DTN-F for internal transfers as well; however, it is not recommended if the data is connected to NCAR-DA-9. It will create a slower overall transfer process.

5.4.2 User On-boarding Workflow

As of early 2022, user on-boarding for DTN systems is in its initial phase. At this time a user is to contact the Cyberinfrastructure group in OIT requesting DTN access. OIT personnel are then to provide instructions detailing the User Transfer Workflow Outlined in Subsection 5.4.1 and access to documentation on the Globus support page. This details installing GCP on a lab machine as well as use of the Globus online website.

On each DTN, a directory is to be created using the university NetID of the PI. This will serve as the home directory for a shared Globus collection. On DTN-S, DTN-F, and NCAR-DA-9, this collection is to be created as a directory at `/data/gateway/`, the mount point for the filesystem dedicated to Globus. The folder's data group is to be set to globusdata.

From here all changes are to be made through the Globus website for DTN-S and

DTN-F. Using the globusdata user, each DTN is to be accessed through its designated mapped collection. A new shared guest collection is to be created using the name of the PI. This collection's details are to be set to the endpoint on which this collection exists. The default directory is then to be set to the directory created in the previous step.

Unlike DTN-S and DTN-F, collection creation is completed through the command line on NCAR-DA-9. Because NCAR-DA-9 has identity mapping features that connect the DTN to a local user on campus, mapped collections must be utilized to handle local users as opposed to guest collections. To create a new collection, "`globus-connect-server collection create`" must be run utilizing the "`--no-allow-guest-collections`" option. This allows only users local to UNR that can be mapped to the collection. This collection will be based out of the PI directory set in the previous step.

Local permissions are to be changed within the new collection. The group "`unr.edu Globus Admins`" is to be added as administrators on the new collection. The PI is then added to the new collection as an Access Manager. This will give them the necessary permission to add collaborators and lab members to their collection.

In the next phase, direct contact with the cyberinfrastructure group is to be replaced. Instead of sending an email directly to team members, a form will be added to the University's Office of Information Technology website that is to be filled out and submitted as an access request. This will help streamline the process and contribute to better organization.

Chapter 6

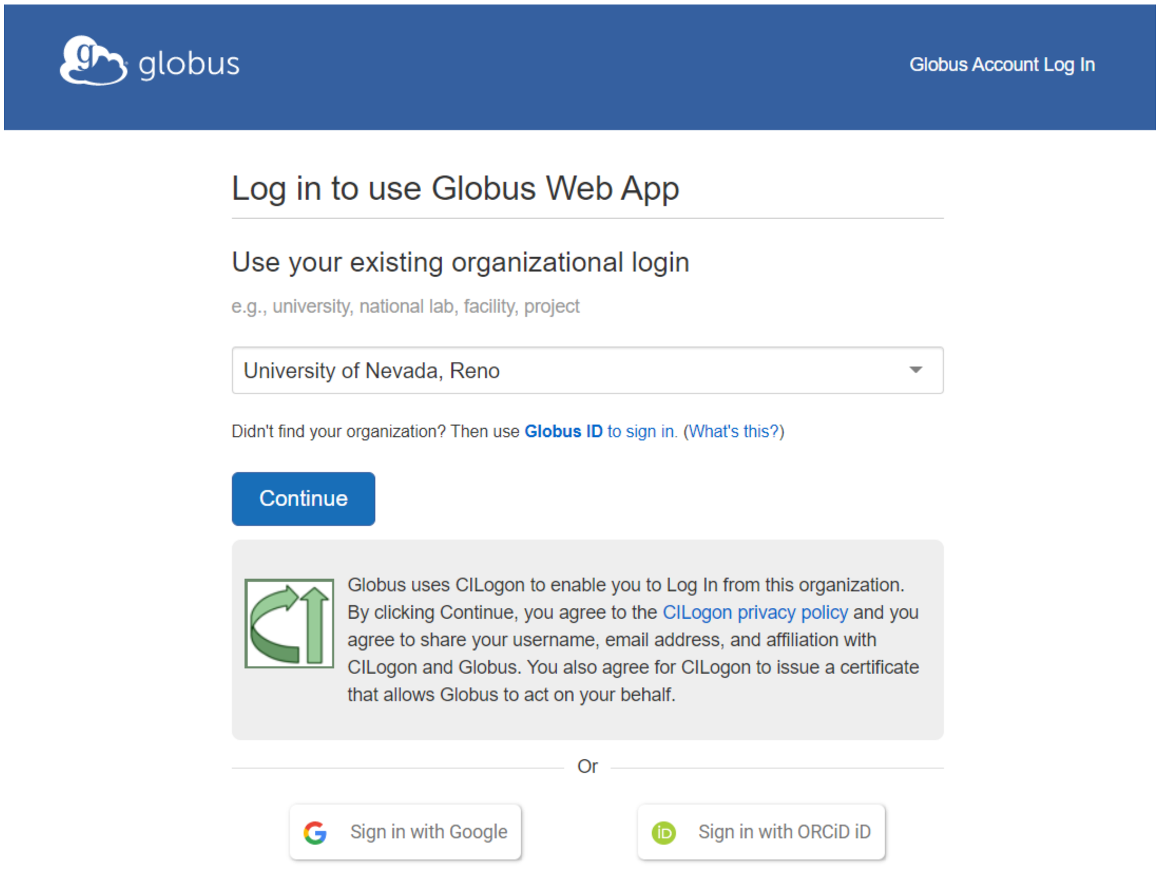
Applications in Action

In this Chapter, the User Interface components for DDRIS will be discussed in detail, including their iterations. First, Globus Online will be discussed in Section 6.1. While Globus was not developed as a part of DDRIS, its incorporation requires training materials for on-boarding new users. This section provides descriptions of how to use Globus Online with accompanying figures. Second, the original DDRIS dashboard, hosted on the Prometheus server, will be discussed in Section 6.2. This original dashboard was automatically generated by Prometheus and continues to run on DDRIS to provide additional information not available in the primary DDRIS Dashboard. Next, Section 6.3 provides information on the data visualization aspect of network telemetry collection. Finally, Section 6.4 provides an in-depth overview of the finalized DDRIS Dashboard, including the organization and the available data visualization charts included in the dashboard.

6.1 Globus

While not developed as a part of this thesis, it is important to understand how to use Globus Online. This section serves as a brief manual for using Globus at the University of Nevada, Reno and is to be utilized in the onboarding process. This section does not include Globus Connect Server (GCS). Installation steps for GCS can be found in the Globus documentation [85].

To begin, users must Login on Globus Online by clicking the “Log in” button



globus Globus Account Log In

Log in to use Globus Web App

Use your existing organizational login

e.g., university, national lab, facility, project

University of Nevada, Reno

Didn't find your organization? Then use **Globus ID** to sign in. (What's this?)

Continue

Globus uses CILogon to enable you to Log In from this organization. By clicking Continue, you agree to the CILogon privacy policy and you agree to share your username, email address, and affiliation with CILogon and Globus. You also agree for CILogon to issue a certificate that allows Globus to act on your behalf.

Or

Sign in with Google

Sign in with ORCiD iD

Figure 6.1: The Globus login client prompting a user to select their organization. As a member of the University of Nevada, Reno, this has been selected as this user's organization.

in the top right corner of the screen. The user is then redirected to a page that allows them to search for an organization, shown in Figure 6.1. As a member of the University of Nevada, Reno, faculty, students, and staff have access to the UNR Globus Subscription. After choosing the University of Nevada, Reno, the user will be prompted to login using their NetID through CILogon.

After being logged on, the user is redirected to the Globus File Manager, as shown in Figure 6.2. The File Manager is the main page of Globus. Here the user can select a collection, specify file paths, and manage files in a Collection.

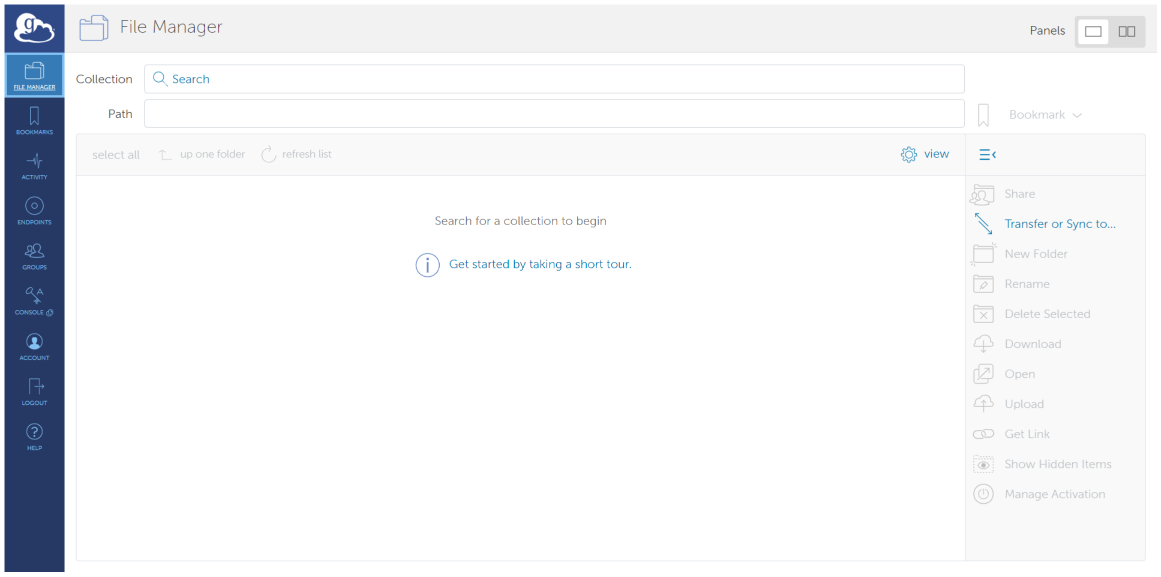
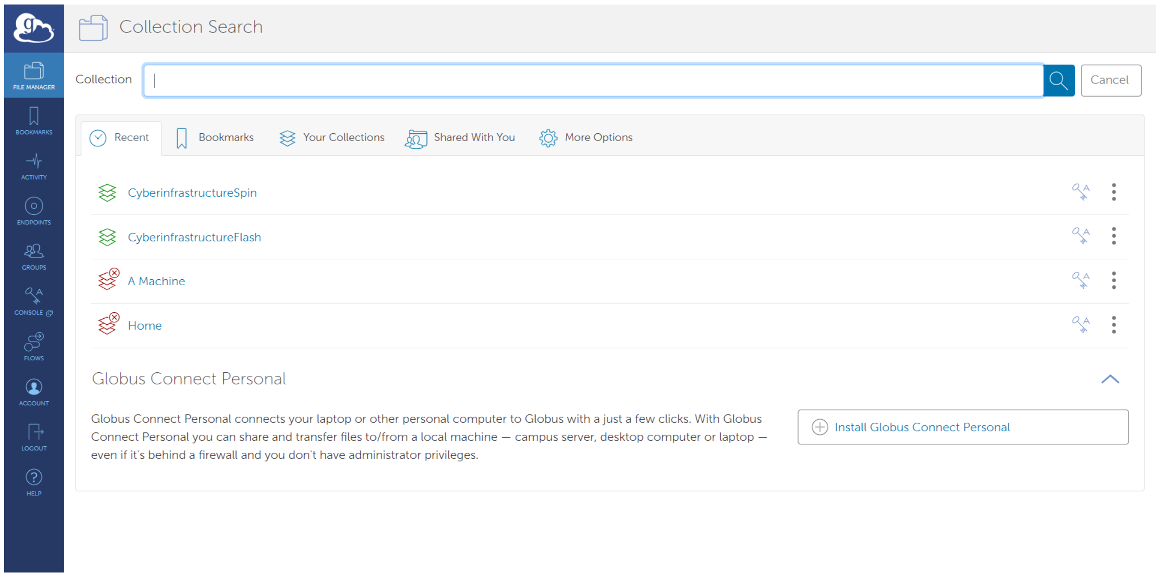


Figure 6.2: The Globus File Manager after logging in on Globus.

To select a collection, the user has two options. First, a user can click on the collection search bar and find their preferred endpoint as shown in Figure 6.3. Users have the option to choose from their recently used collections, collections owned by the user, collections shared with the user, and bookmarked locations. A user can also type the name of an endpoint that is listed publicly to view or request access to a collection on that endpoint.

The second option to select a collection is to navigate to the "Endpoints" page shown in Figure 6.4. Like the first option, users have the ability to view and select recently used collections, collections owned by the user, and collections shared with the user. Users do not have the ability to view bookmarks on this page; however a user can see what collections are in use and which collections are administered by the user. A user can also search for endpoints.

After a collection has been selected from one of these two locations, the user is redirected back to the File Manager. If the user is transferring a file, they will change the File Manager panel view to two pane and select a second collection. The user then selects a file and clicks start to begin a data transfer. The File Manager ready to transfer a file can be seen in Figure 6.5.



Collection Search

Collection | Cancel

Recent | Bookmarks | Your Collections | Shared With You | More Options

CyberinfrastructureSpin

CyberinfrastructureFlash

A Machine

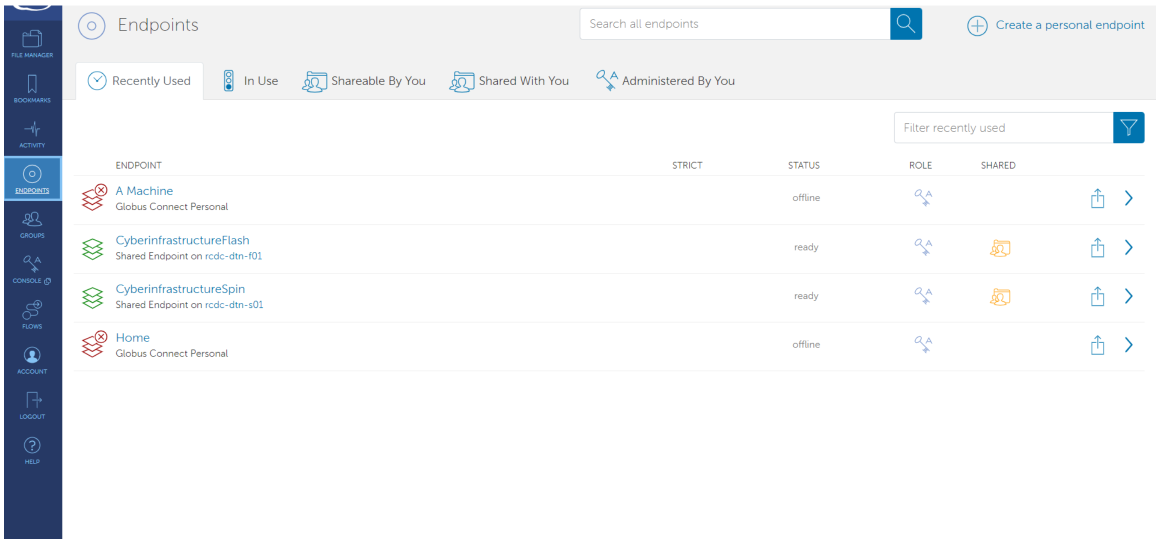
Home

Globus Connect Personal

Globus Connect Personal connects your laptop or other personal computer to Globus with a just a few clicks. With Globus Connect Personal you can share and transfer files to/from a local machine — campus server, desktop computer or laptop — even if it's behind a firewall and you don't have administrator privileges.

Install Globus Connect Personal

Figure 6.3: The Globus Collection discovery shows all available Collections to transfer against.



Endpoints

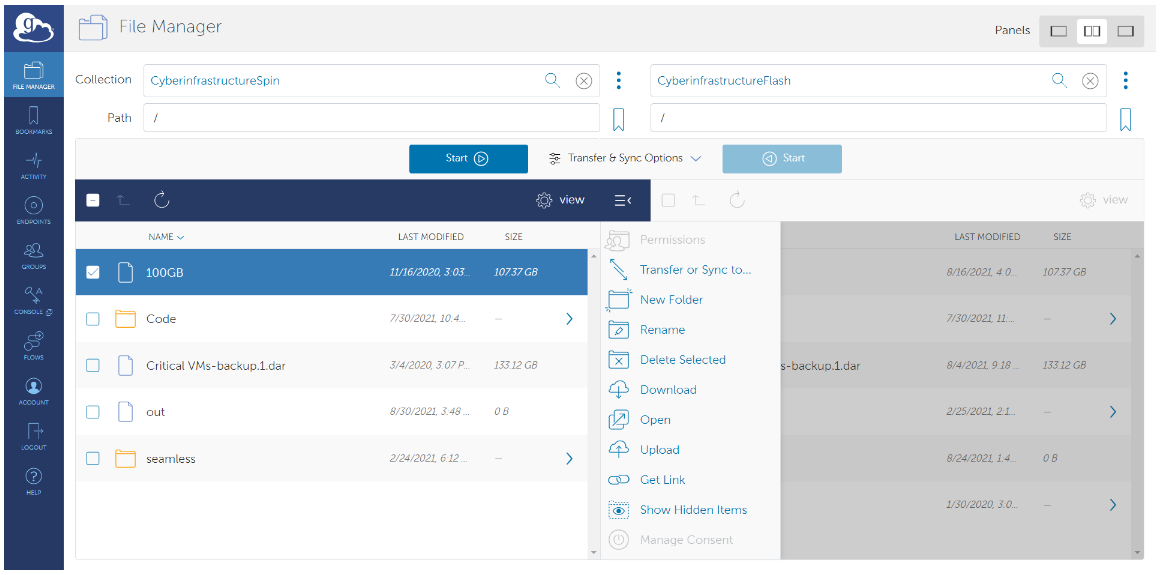
Search all endpoints | Create a personal endpoint

Recently Used | In Use | Shareable By You | Shared With You | Administered By You

Filter recently used

ENDPOINT	STRICT	STATUS	ROLE	SHARED
A Machine Globus Connect Personal		offline		
CyberinfrastructureFlash Shared Endpoint on rcdc-dtn-f01		ready		
CyberinfrastructureSpin Shared Endpoint on rcdc-dtn-s01		ready		
Home Globus Connect Personal		offline		

Figure 6.4: The Globus Online Endpoint List Page allows users to view all endpoints they have access to.



Collection: CyberinfrastructureSpin
Path: /

NAME	LAST MODIFIED	SIZE
100GB	11/16/2020, 3:03...	107.37 GB
Code	7/30/2021, 10:4...	–
Critical VMs-backup.1.dar	3/4/2020, 3:07 P...	133.12 GB
out	8/30/2021, 3:48 ...	0 B
seamless	2/24/2021, 6:12 ...	–

Collection: CyberinfrastructureFlash
Path: /

	LAST MODIFIED	SIZE
	8/16/2021, 4:0...	107.37 GB
	7/30/2021, 11:...	–
s-backup.1.dar	8/4/2021, 9:18 ...	133.12 GB
	2/25/2021, 2:1...	–
	8/24/2021, 1:4...	0 B
	1/30/2020, 3:0...	–

Figure 6.5: The Globus File Manager ready to start a transfer between collections CyberinfrastructureSpin and CyberinfrastructureFlash.

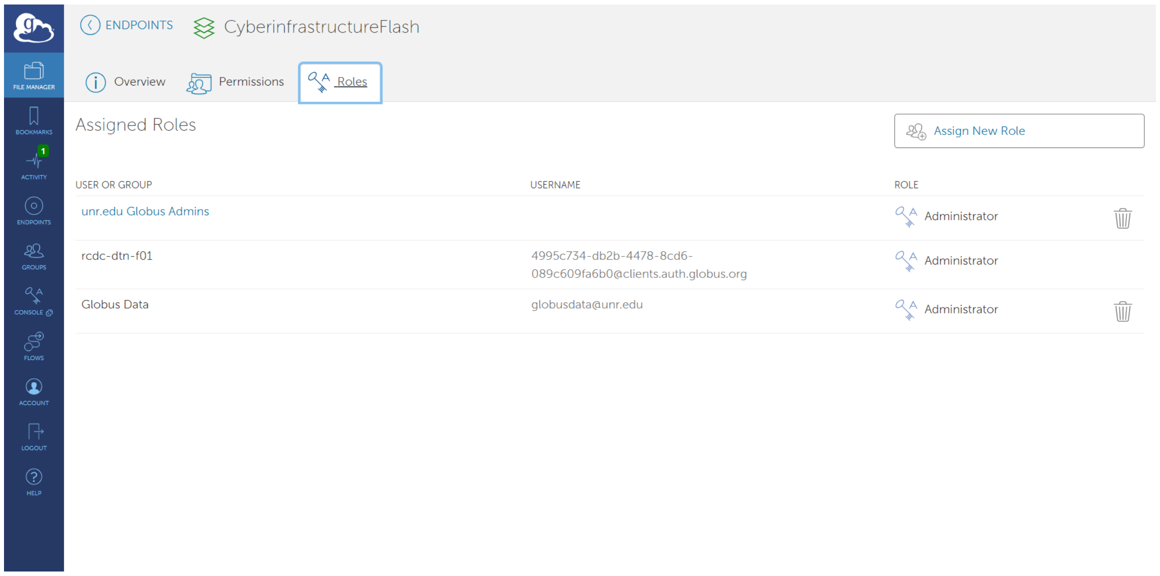
A user also has the ability to upload, download, or delete files from a collection in this view. All of these options are accessible through the middle console.

In addition to transferring data, researchers are responsible for managing access to their data. Globus separates access rules into two types: roles, and permissions. Roles refer to the delegation of collection management capabilities while permissions include read and write access.

To change roles and permissions, users navigate to the Endpoint page and select the collection that is being updated. From here, the user navigates to the Roles tab, shown in Figure 6.6, or the Permissions tab. Then the researcher clicks on “Assign New Role” or “Assign New Permissions” respectively.

When adding new roles, the user specifies who the role will be assigned to. This can be a user or a group. To locate the user or group, the researcher will utilize the global search feature. If the user is associated with UNR, the researcher can search for the user’s Network Identification (NetID) to locate the user quickly. The researcher then chooses a role to assign this user or group. This process is shown in Figure 6.7.

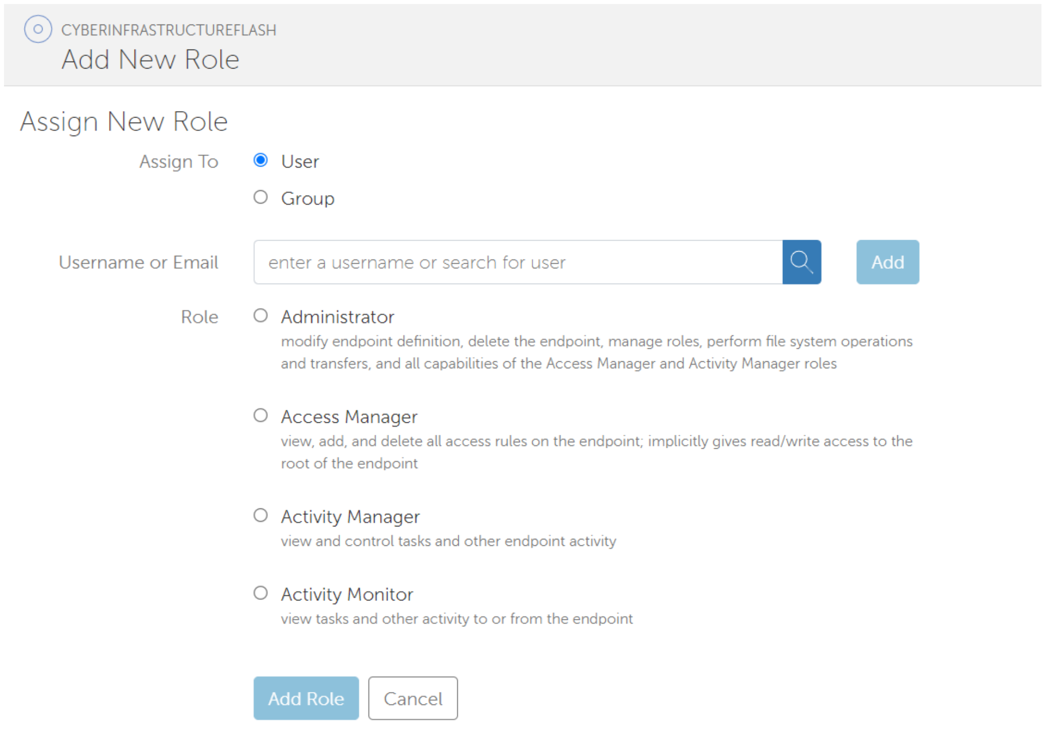
The four roles provided as options to the user include Administrator, Access Manager, Activity Manager, and Activity Monitor. The role of Administrator is



Assigned Roles

USER OR GROUP	USERNAME	ROLE
unr.edu Globus Admins		Administrator
rcdc-dtn-f01	4995c734-db2b-4478-8cd6-089c609fa6b0@clients.auth.globus.org	Administrator
Globus Data	globusdata@unr.edu	Administrator

Figure 6.6: Globus Collection Roles for CyberinfrastructureFlash.



CYBERINFRASTRUCTUREFLASH

Add New Role

Assign New Role

Assign To [✓ User] Group

Username or Email: enter a username or search for user

Role

Administrator
modify endpoint definition, delete the endpoint, manage roles, perform file system operations and transfers, and all capabilities of the Access Manager and Activity Manager roles

Access Manager
view, add, and delete all access rules on the endpoint; implicitly gives read/write access to the root of the endpoint

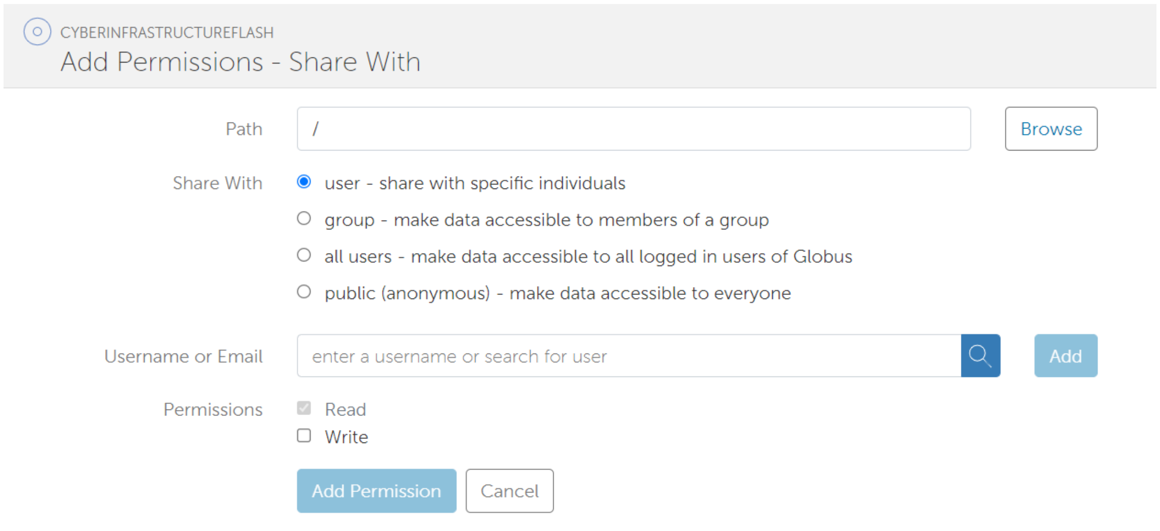
Activity Manager
view and control tasks and other endpoint activity

Activity Monitor
view tasks and other activity to or from the endpoint

Figure 6.7: Adding a new role to a Collection.

reserved for personnel responsible for maintaining the endpoint and will not be accessible by researchers. Access Manager has the ability to view, add, and delete access rules. By assigning the role of Access Manager to a group or user, that group or user is automatically provided read and write permissions across the collection. Activity Managers can control tasks performed on the endpoint. Activity Monitors can view tasks performed on the endpoint. Activity Managers and Activity Monitors do not automatically have read and write permissions.

Permissions can be specified for specific file paths within a collection or the root directory. If permissions are granted for a parent directory, those permissions are included for all child directories. Permissions can be added for users, groups, all Globus users, or be made public to everyone. By adding permissions, the specified users are immediately provided read access, and write access can be specified. Adding new permissions can be seen in Figure 6.8.


CYBERINFRASTRUCTUREFLASH

Add Permissions - Share With

Path: /

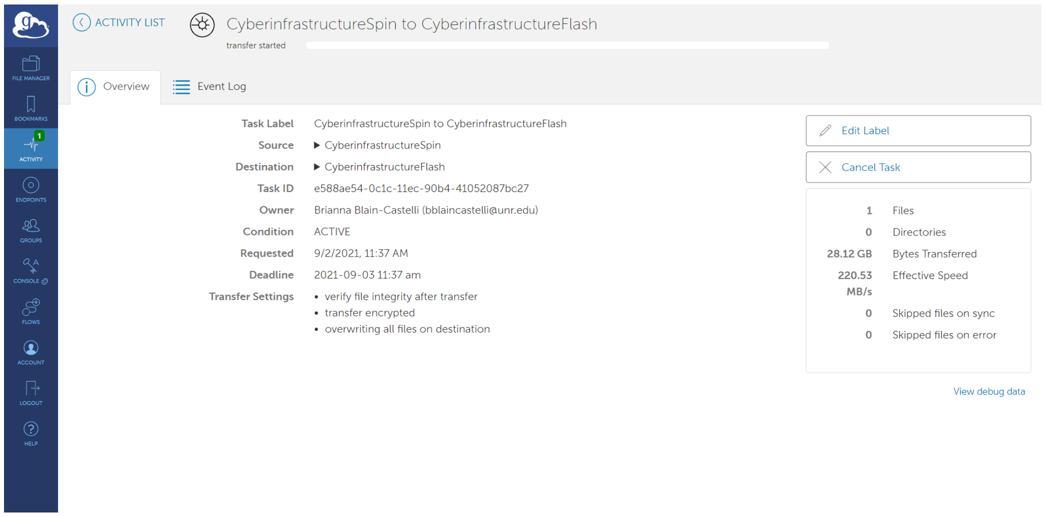
Share With:
- [✓ user - share with specific individuals]
- group - make data accessible to members of a group
- all users - make data accessible to all logged in users of Globus
- public (anonymous) - make data accessible to everyone

Username or Email: enter a username or search for user

Permissions:
- [✓ Read]
- Write

Figure 6.8: Adding new permissions to a Collection.

Regardless of roles, users have the ability to view transfers they submitted through the Activity Monitoring page, shown in Figure 6.9. From here, users can select a transfer to view transfer metadata, including start time, transfer settings, source, destination, owner, bytes transferred, transfer speed, and condition. Users also have the ability to transfer their own tasks or view events surrounding a task.



CyberinfrastructureSpin to CyberinfrastructureFlash

transfer started

Task Label	CyberinfrastructureSpin to CyberinfrastructureFlash
Source	▶ CyberinfrastructureSpin
Destination	▶ CyberinfrastructureFlash
Task ID	e588ae54-0c1c-11ec-90b4-41052087bc27
Owner	Brianna Blain-Castelli (bblaincastelli@unr.edu)
Condition	ACTIVE
Requested	9/2/2021, 11:37 AM
Deadline	2021-09-03 11:37 am
Transfer Settings	• verify file integrity after transfer • transfer encrypted • overwriting all files on destination

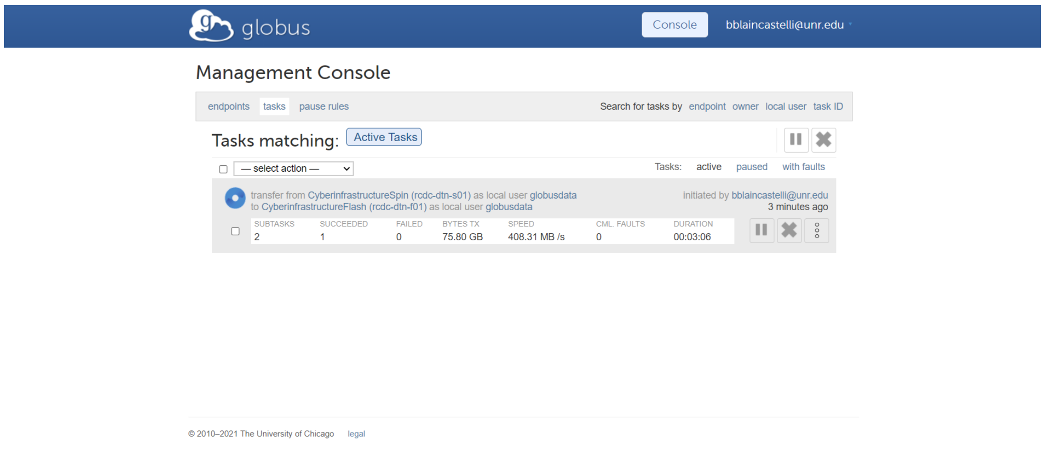
1	Files
0	Directories
28.12 GB	Bytes Transferred
220.53 MB/s	Effective Speed
0	Skipped files on sync
0	Skipped files on error

Figure 6.9: Globus Activity Monitoring shows Bytes transferred, effective speeds, task id, source, destination, and other transfer metadata for recent or current transfers by a user.

Users with the Activity Manager role have access to the Globus Management Console in Figure 6.10. The Management Console, allows Activity Managers to view, pause, or cancel any tasks performed on collections they hold this role for. This view also allows for a more detailed view of the transfer, providing the same information at the Activity Monitoring page.

6.2 Prometheus UI

The dashboard UI was originally limited to that provided by Prometheus reporting by default, as shown in Figure 6.11. This allowed users to view transfer speeds with limited interact-ability. While this method worked well during initial deployment, more information was pertinent towards the transfers than information output by Prometheus. Unfortunately, Prometheus reporting also did not allow for automatic refreshing or metrics that do not have counts. This meant that audit-trail data including user responsible for the transfer, IP address of the transfer, and more would not be able to be collected through this method. Additionally, data from the DTNs and Kubernetes were collected by different, dedicated Prometheus servers, so two


globus
Console
bblaincastelli@unr.edu
endpoints
tasks
pause rules
Search for tasks by endpoint owner local user task ID
Active Tasks
— select action —
Tasks: active paused with faults
© 2010–2021 The University of Chicago legal

Management Console

Tasks matching: Active Tasks

transfer from CyberinfrastructureSpin (rcdc-dtn-s01) as local user globusdata to CyberinfrastructureFlash (rcdc-dtn-f01) as local user globusdata

initiated by bblaincastelli@unr.edu 3 minutes ago

SUBTASKS	SUCCEEDED	FAILED	BYTES TX	SPEED	CML. FAULTS	DURATION
2	1	0	75.80 GB	408.31 MB /s	0	00:03:06

Figure 6.10: The Globus Management Console shows transfer metadata for recent or current transfers on all collections accessible by a user.

different addresses would need to be accessed to view all of the data. As a solution, the transfer speeds graph was output to a Postgres database for more permanent storage and queried using PostgreSQL through the Grafana platform. This also allowed for more metrics to be reported at one time and customization of visualizations and will be covered in detail in Section 6.4. However, all Prometheus web pages continue to run to provide information on changes to system health because a detailed version of all data would not be manageable on the single Grafana interface.

All data collected through Prometheus is available in the drop-down menu under the Graph page, shown in Figure 6.12. After selecting selecting an option, the user clicks on “Execute” to pull records. All data pertaining to that query is then graphed on a line graph where the user may filter by time and metric name. Prometheus also offers a console view, which displays this data in a two column table.

6.3 MaDDash

MaDDash is part of the PerfSONAR package, designed as a tool for visualizing network telemetry data. As previously shown in Figure 5.3, this platform was deployed on SciDMZ-Perf, the head of the PerfSONAR mesh. The primary GUI for MaD-

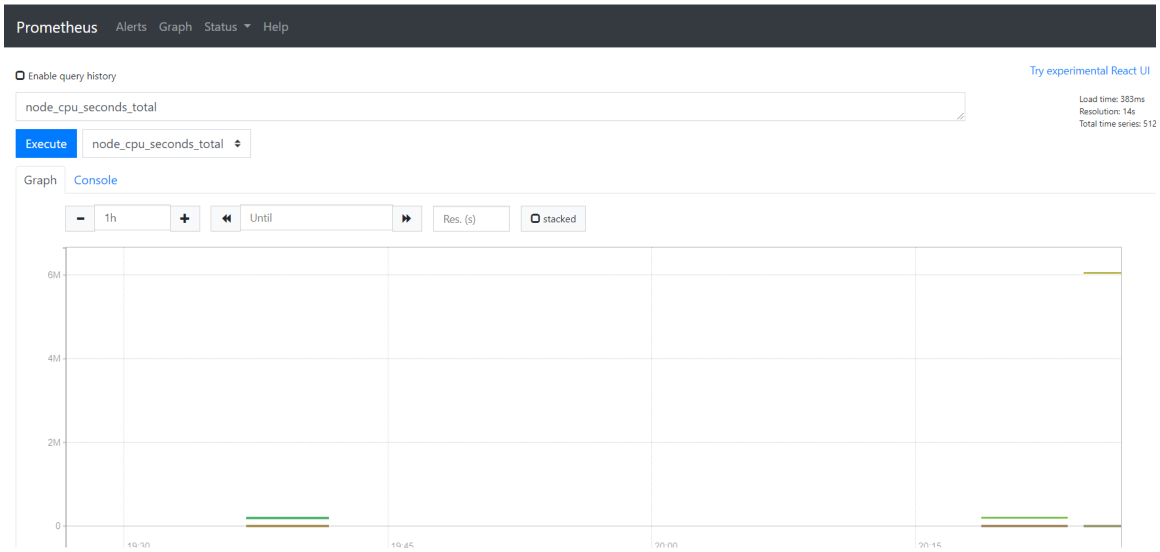


Figure 6.11: An overview of the Prometheus User Interface for DDRIS DTNs.

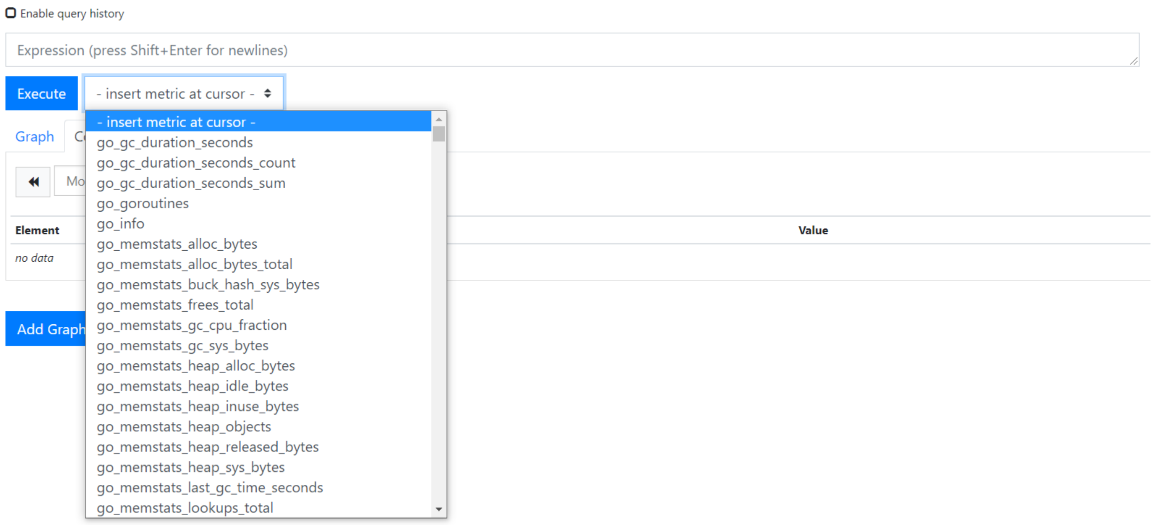
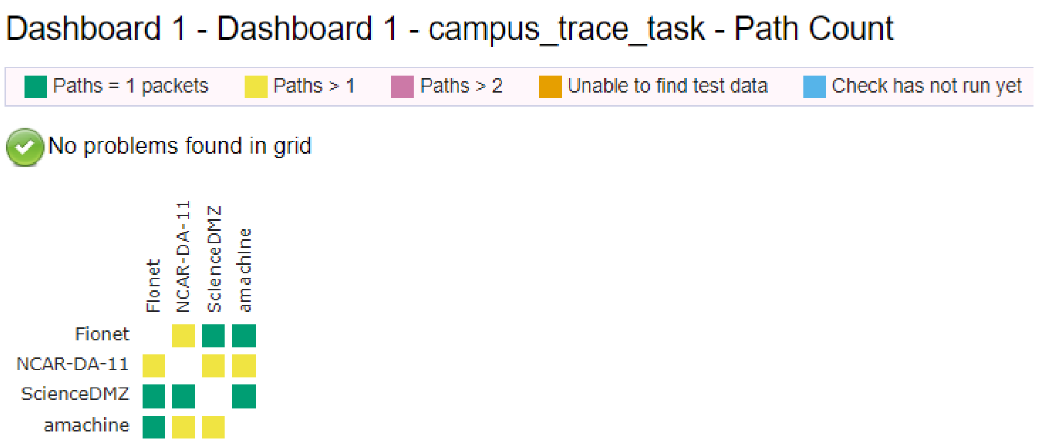

Enable query history
Expression (press Shift+Enter for newlines)
Execute
- insert metric at cursor -
Graph
Element
Value
no data
Add Graph
- insert metric at cursor -
go_gc_duration_seconds
go_gc_duration_seconds_count
go_gc_duration_seconds_sum
go_goroutines
go_info
go_memstats_alloc_bytes
go_memstats_alloc_bytes_total
go_memstats_buck_hash_sys_bytes
go_memstats_frees_total
go_memstats_gc_cpu_fraction
go_memstats_gc_sys_bytes
go_memstats_heap_alloc_bytes
go_memstats_heap_idle_bytes
go_memstats_heap_inuse_bytes
go_memstats_heap_objects
go_memstats_heap_released_bytes
go_memstats_heap_sys_bytes
go_memstats_last_gc_time_seconds
go_memstats_lookups_total

Figure 6.12: The selection menu for Prometheus metrics.

Dash is a collection of heatmaps corresponding with individual tests and groups, as shown in Figure 6.13. These heatmaps feature testpoints on the Y-axis that represent test source and X-axis that represent test destination as well as associated colors for each test. Colors correspond with stoplights to show good, acceptable, and poor connectivity results with an additional orange and blue for missing data.


Dashboard 1 - Dashboard 1 - campus_trace_task - Path Count

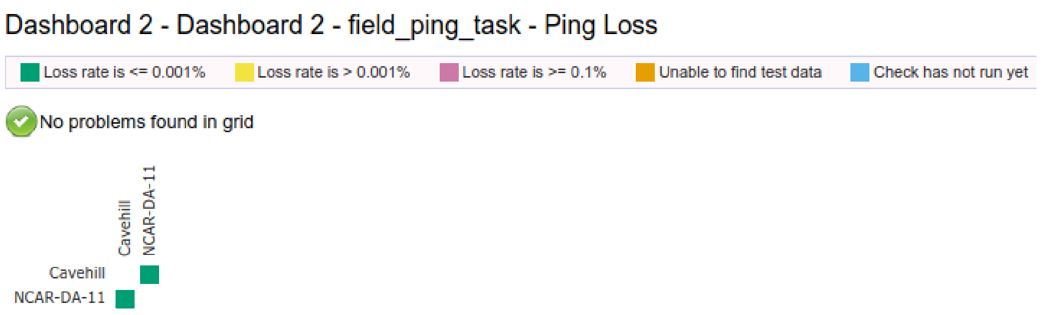
🟩 Paths = 1 packets 🟨 Paths > 1 🟪 Paths > 2 🟧 Unable to find test data 🟦 Check has not run yet

✓ No problems found in grid

	Fionet	NCAR-DA-11	ScienceDMZ	amachine
Fionet		🟨	🟩	🟩
NCAR-DA-11	🟨		🟨	🟨
ScienceDMZ	🟩	🟩		🟩
amachine	🟩	🟨	🟨	

Figure 6.13: MaDDash heatmap visualization showing traceroute results for the campus mesh group.

Each group in a mesh has its own set of heatmaps for each network test. These include, latency, ping, traceroute, and throughput tests that have been individually specified in the mesh configuration. In its current iteration, DDRIS has two groups: campus, illustrated in Figure 6.13, and field, shown in Figure 6.14. The campus group includes machines internal to the campus network while the field group refers


Dashboard 2 - Dashboard 2 - field_ping_task - Ping Loss

🟩 Loss rate is <= 0.001% 🟨 Loss rate is > 0.001% 🟪 Loss rate is >= 0.1% 🟧 Unable to find test data 🟦 Check has not run yet

✓ No problems found in grid

	Cavehill	NCAR-DA-11
Cavehill		🟩
NCAR-DA-11	🟩	

Figure 6.14: MaDDash visualization of ping test results as heatmaps for the field group.

to edge devices deployed as a part of in situ research projects. As the mesh is further expanded, the field group will be separated based on project.

Further visualizations include a historical representation of result metrics, exemplified in Figure 6.15. This view is accessed by selecting a point on the heatmap to specify source and destination. Throughput, ping, and latency share this historical representation of telemetry results with filterable options along the top legend to limit results to one or more test or metric types. Furthermore, the mouse can be utilized to hover over the line graphs to receive exact point information.

Traceroute has its own visualization type as a table. This table, and its search functionality, was developed at the University of Wisconsin, Madison. It allows users to view individual network hops taken to connect from the source to the destination node, including IP and delay time.

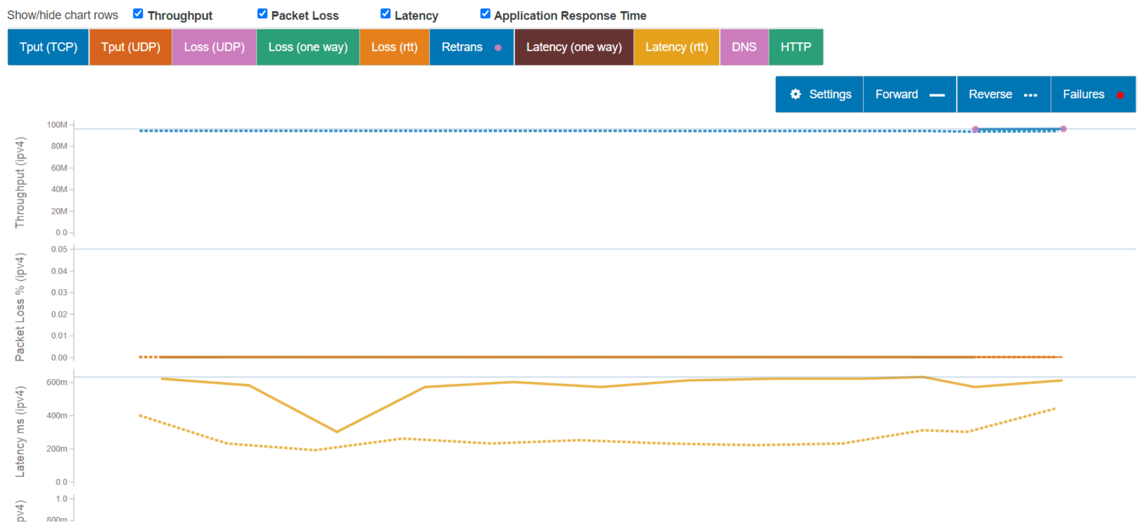


Figure 6.15: MaDDash visualization of network tests from NCAR-DA-11 to Roamer-Perf.

6.4 DDRIS Dashboard

The DDRIS dashboard is the primary GUI for system administrators tasked with maintaining the health of DDRIS and troubleshooting performance related concerns for users. Built utilizing Grafana, the dashboard is meant to be simple but provide long-term historical records to detect trends and anomalies. Subsection 6.4.1 provides an overview of the pages available to dashboard users. Details of the data visualizations provided by the DDRIS dashboard are available in Subsection 6.4.2.

6.4.1 User Interface

The DDRIS custom dashboard was created to provide supplemental information to system administrators for troubleshooting, documentation, and alerts. It was built with simplicity in mind to support the key concepts of Human Computer Interaction, including keeping the dashboard easy to use and easy to learn. To promote this, few pages are used to organize the data and further organization relies on accordion-style menus to filter visualizations. The pages available to a viewer include the login page, dashboard home, user profile, alert rules, the Cyberinfrastructure Data Transfer Dashboard, and the Kubernetes Metadata Dashboard.

When a user navigates to the DDRIS Dashboard, the first page to load will be the login page, created and managed by Grafana as presented in Figure 6.16. The user will be prompted for user credentials that are managed within Grafana and can belong to one of three user types: admin, editor, and user. Each has access to different permissions within the dashboard. User management is specific to the DDRIS Grafana instance and is handled within Grafana.

Following login, the user is redirected to the Grafana home page showcased in Figure 6.17. This page is also managed by Grafana and shows recently viewed dashboards and new announcements from the Grafana developers. There are also links available to receive help or access documentation and tutorials regarding Grafana. The primary purpose of the home page is to provide access to further navigation.

From the “Home” page, users may then access their user profile. This allows a user to update their contact information for alerts or modify their password. This page is also fully managed by the Grafana software. The “Profile” page can be seen in Figure 6.18.

Any alert rules and their current statuses are listed on the “Alerts” page as

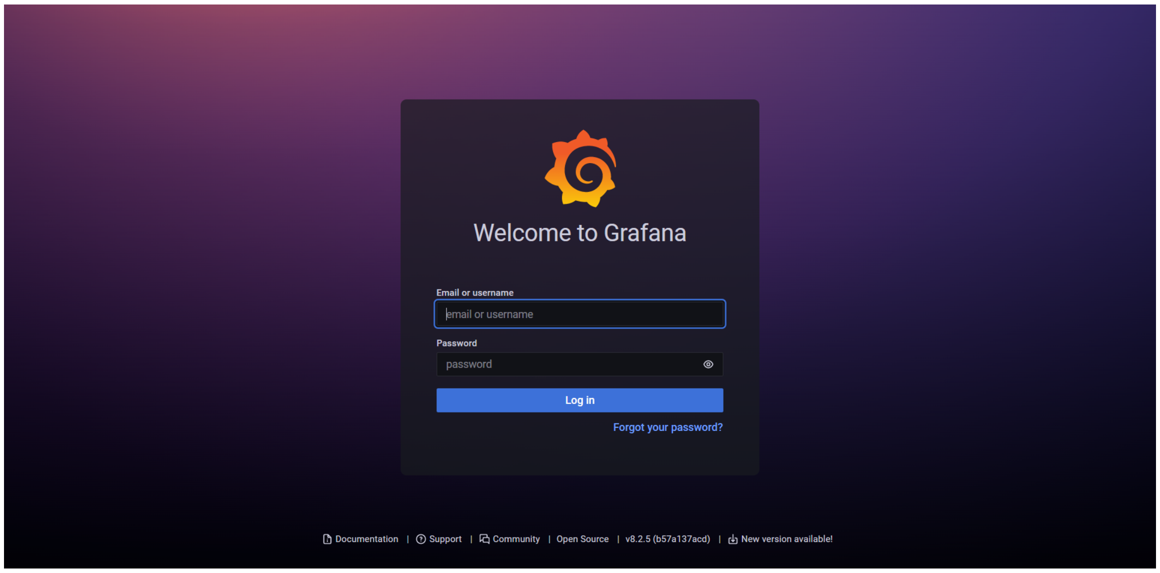
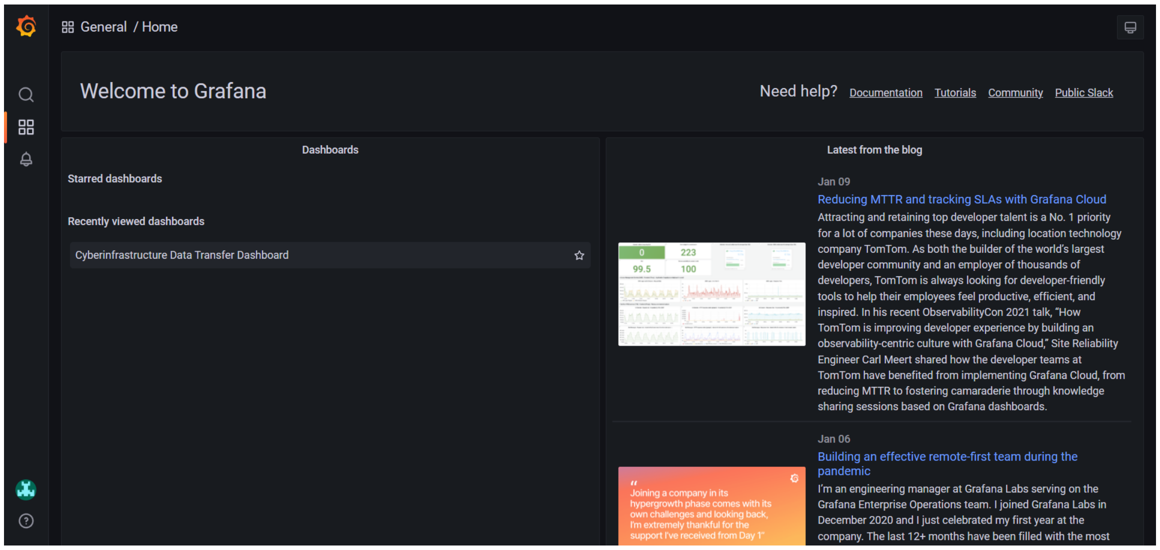

Welcome to Grafana
Email or username
email or username
Password
password
Log in
Forgot your password?
Documentation | Support | Community | Open Source | v8.2.5 (b57a137acd) | New version available!

Figure 6.16: A screenshot of the Login page for the DDRIS dashboard, created and managed by Grafana.


General / Home
Welcome to Grafana
Need help? Documentation Tutorials Community Public Slack
Dashboards
Starred dashboards
Recently viewed dashboards
Cyberinfrastructure Data Transfer Dashboard
Latest from the blog

Jan 09

Reducing MTTR and tracking SLAs with Grafana Cloud

Attracting and retaining top developer talent is a No. 1 priority for a lot of companies these days, including location technology company TomTom. As both the builder of the world's largest developer community and an employer of thousands of developers, TomTom is always looking for developer-friendly tools to help their employees feel productive, efficient, and inspired. In his recent ObservabilityCon 2021 talk, "How TomTom is improving developer experience by building an observability-centric culture with Grafana Cloud," Site Reliability Engineer Carl Meert shared how the developer teams at TomTom have benefited from implementing Grafana Cloud, from reducing MTTR to fostering camaraderie through knowledge sharing sessions based on Grafana dashboards.

Jan 06

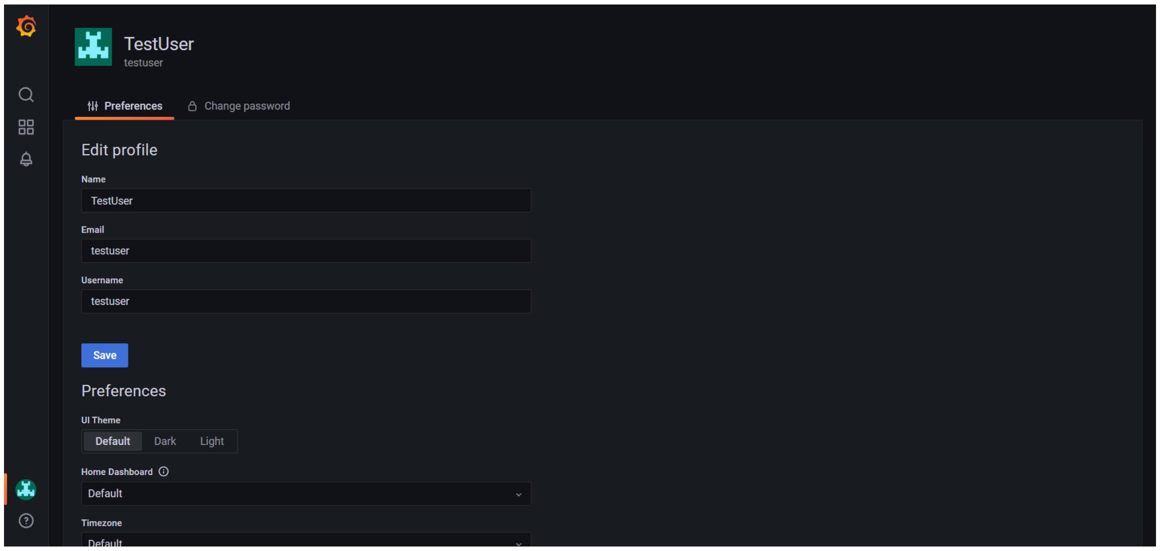
Building an effective remote-first team during the pandemic

"Joining a company in its hypergrowth phase comes with its own challenges and looking back, I'm extremely thankful for the support I've received from Day 1"

I'm an engineering manager at Grafana Labs serving on the Grafana Enterprise Operations team. I joined Grafana Labs in December 2020 and I just celebrated my first year at the company. The last 12+ months have been filled with the most

Figure 6.17: The “Home” page for the Grafana dashboard shows recent dashboards and provides access to navigation.

presented in Figure 6.19. Alerts are only linked to graph-type data visualizations in Grafana and are created by specifying bounds to expected values. When a query is created on a graph, it is then automatically reflected on the “Alerts” page by Grafana. These alerts can then be modified and alert channels are specifiable, including email, Slack [75], or PagerDuty [56].



TestUser
testuser

Edit profile

Name: TestUser

Email: testuser

Username: testuser

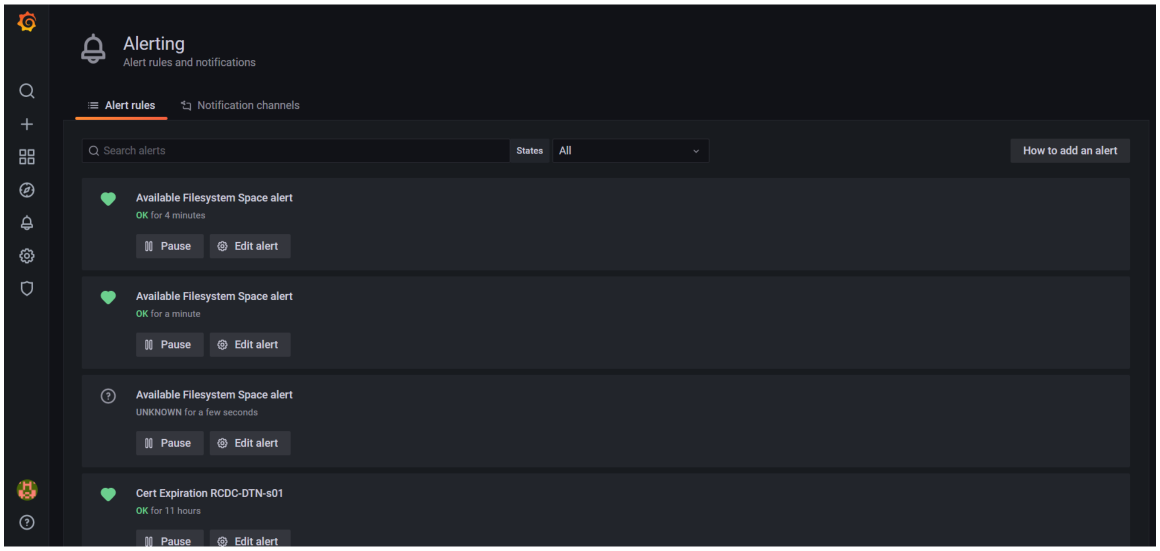
Preferences

UI Theme: Default | Dark | Light

Home Dashboard: Default

Timezone: Default

Figure 6.18: The "Profile" page of the Grafana dashboard where users may change or view their user profile information.



Alerting
Alert rules and notifications

States: All

Available Filesystem Space alert
OK for 4 minutes
Pause | Edit alert

Available Filesystem Space alert
OK for a minute
Pause | Edit alert

Available Filesystem Space alert
UNKNOWN for a few seconds
Pause | Edit alert

Cert Expiration RCDC-DTN-s01
OK for 11 hours
Pause | Edit alert

Figure 6.19: The alert page shows all alerts on the DDRIS dashboard.

Finally, the user has access to the dashboards created through Grafana. The Cyberinfrastructure Data Transfer Dashboard is responsible for displaying information specifically regarding DTNs and their performance. Timeseries data regarding the health of the DTNs is tracked through Prometheus and displayed on this dashboard under an accordion menu dedicated to DTN health as illustrated in Figure 6.20 and Figure 6.21. This data is navigable utilizing the DTN variable above the menu. This variable allows the data to be filtered according to a single DTN.

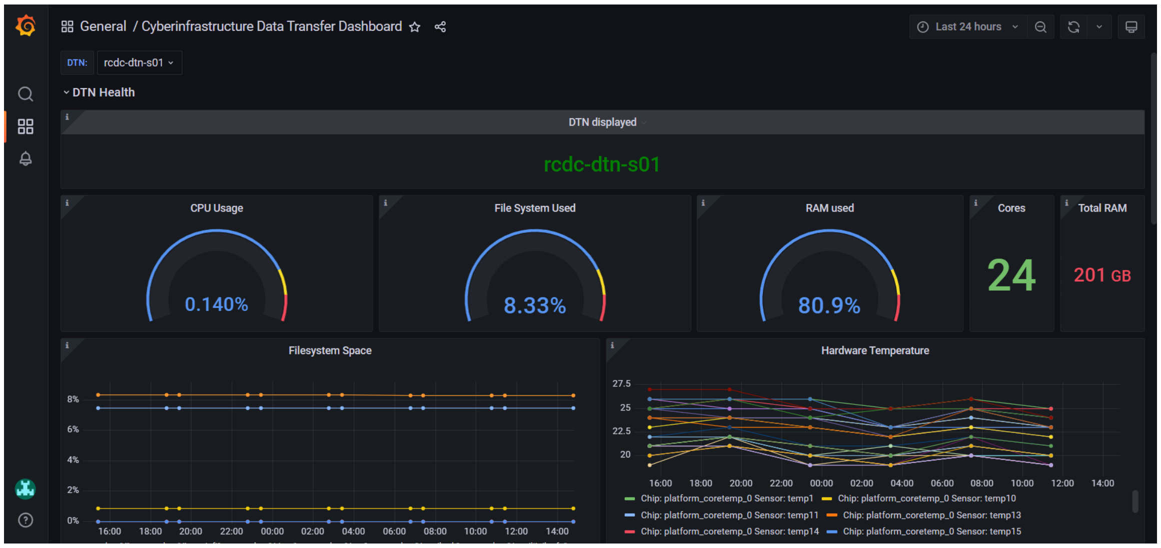


Figure 6.20: A screenshot of the DTN Health tab of the DDRIS dashboard, showing seven data visualizations belonging to DTN-S under the collapsible menu, "DTN Health".

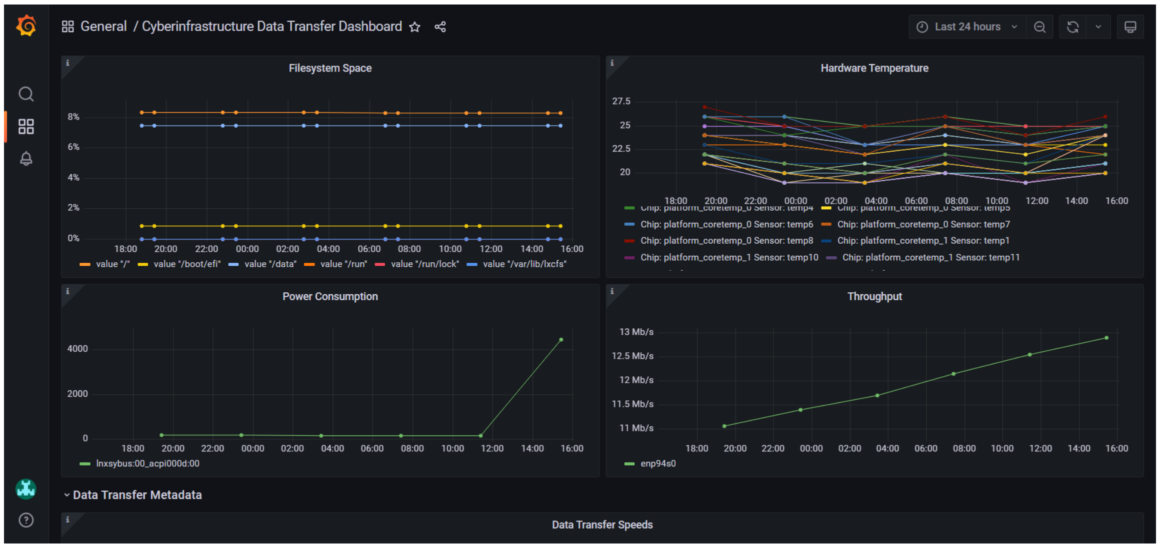
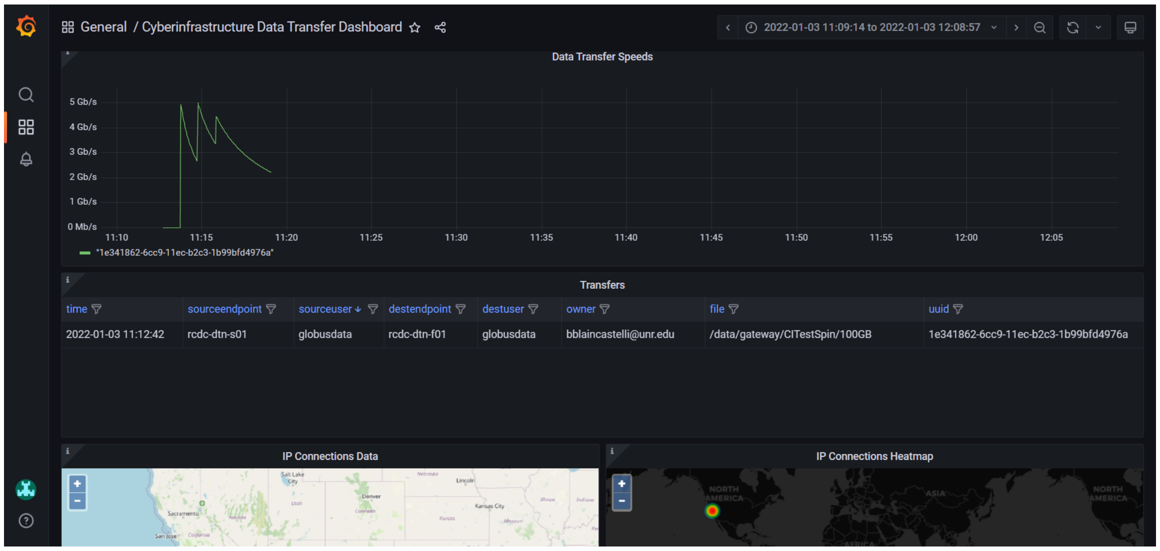


Figure 6.21: A screenshot of further DTN Health on the DDRIS dashboard, showing two additional data visualizations belonging to DTN-S under the collapsible menu, "DTN Health".

Another tab in the menu is "Data Transfer". This tab, shown in Figure 6.22, displays metadata regarding data transfers utilizing the DTNs on campus. Visualizations include maps of transfers, a graph of transfer speeds over time, and a table of



Transfers

time	sourceendpoint	sourceuser	destendpoint	destuser	owner	file	uuid
2022-01-03 11:12:42	rcdc-dtn-s01	globusdata	rcdc-dtn-f01	globusdata	bblaincastelli@unr.edu	/data/gateway/CITestSpin/100GB	1e341862-6cc9-11ec-b2c3-1b99bfd4976a

Figure 6.22: A screenshot of the "Data Transfer" tab on the DDRIS dashboard.

transfer metadata. More information regarding these visualizations will be covered in Subsection 6.4.2.

The remaining tab on the "Cyberinfrastructure Data Transfer Dashboard" is the "Related Dashboards Quicklist". This tab provides direct links to dashboards pertaining to Kubernetes in an organized fashion, allowing for quick navigation between these dashboards without the use of an intermediate page, such as the dashboard search. By limiting the number of pages needed to navigate the dashboards, the ease of use and rate of errors are decreased accordingly.

One relevant dashboard is the "Kubernetes Metadata Dashboard", showcased in Figure 6.23, which provides important metadata describing the health of Kubernetes elements necessary for proper dashboard display and data collection. These graphs show memory usage across multiple pods, including the Postgres database for data transfer and DTN health information, elasticsearch, and the Postgres and Cassandra databases for network telemetry collection. If these databases reach their maximum allocated capacity, then data collection will halt. For this reason, alerts are configured to notify administrators when usage has reached 80% of the maximum capacity.

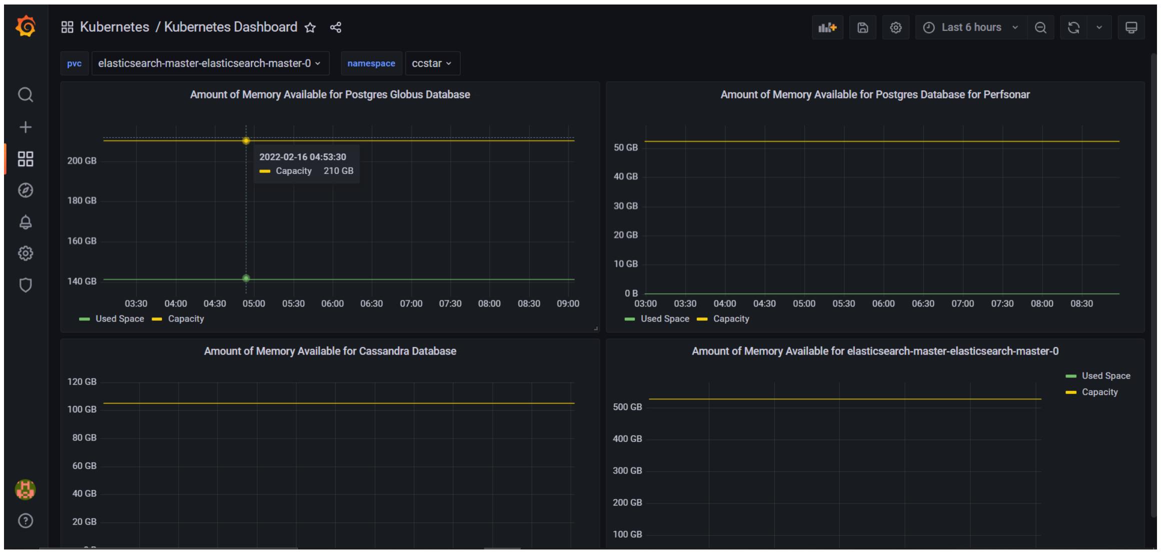

Kubernetes / Kubernetes Dashboard
pvc
elasticsearch-master-elasticsearch-master-0
namespace
ccstar
Last 6 hours
Amount of Memory Available for Postgres Globus Database
2022-02-16 04:53:30
Capacity 210 GB
Amount of Memory Available for Postgres Database for Perfsonar
Amount of Memory Available for Cassandra Database
Amount of Memory Available for elasticsearch-master-elasticsearch-master-0
Used Space
Capacity

Figure 6.23: An overview of the "Kubernetes Metadata Dashboard".

6.4.2 Data Visualization

All data visualizations hosted on the DDRIS dashboard are updated in near-real time with options available for filtering historical records based on time. This allows users to view current data as well as filter through time to find expected values and trends. Eight visualizations belong to the Cyberinfrastructure Data Transfer Dashboard previously defined in Subsection 6.4.1. The first visualization is dedicated to resource usage as percentages, as reflected by Figure 6.24. Total CPU, file system, and RAM usage are all calculated as a percentage and displayed in a gauge format with a Blue, Yellow, Red color scheme. These colors were chosen for color-blind accessibility and additionally have a bar gauge to support alternative methods of conceptualizing the usage percentage. Yellow and Red correspond with threshold values of 80% and 90% respectively.

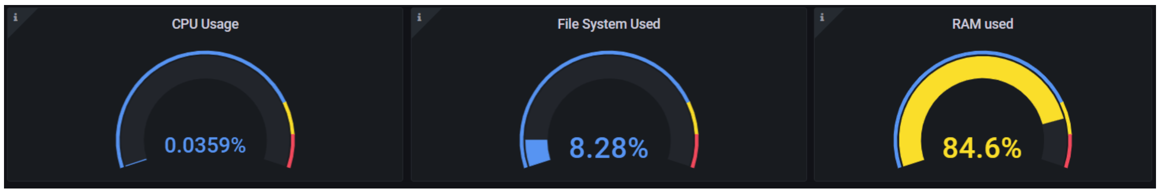


Figure 6.24: DTN health visualizations for resource usage in a gauge format.

The second visualization is a graph of file system usage versus time for each file system mount shown in Figure 6.25. One advantage of displaying this information as a graph is the ability to set alerts on graph-type visualizations in Grafana. File systems are separated by mount point and displayed as the percentage used of each mount point over time. The visualization is filterable by selecting a mount point in the legend and by selecting a time period.

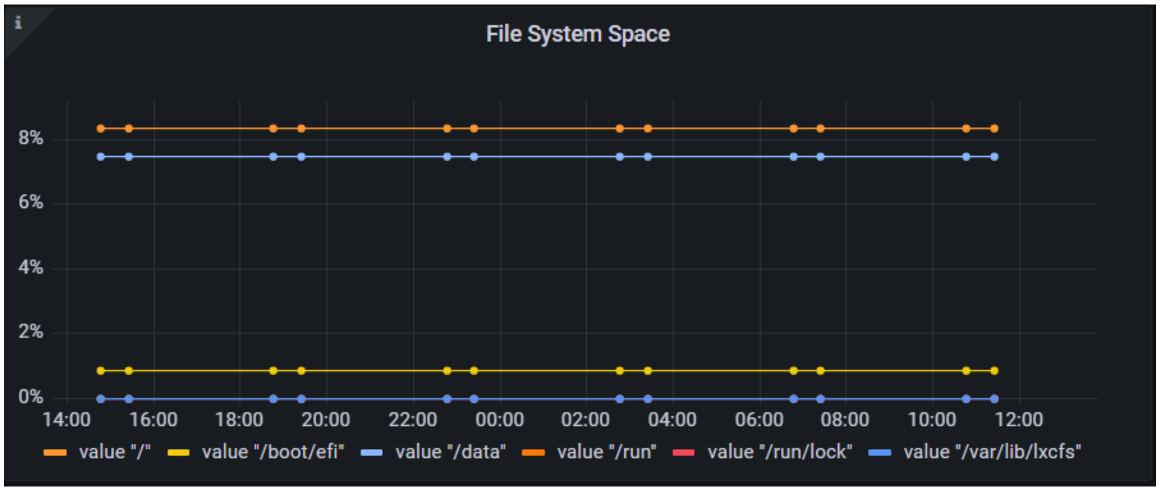


Figure 6.25: Information regarding the DTN's file system capacity displayed in a graph format of percentage used versus time.

Visualization three, presented in Figure 6.26, is hardware temperature over time. Like other graph-type visualizations in Grafana, hardware temperature is filterable by selecting entries within the legend corresponding to the graph. Data is collected for chips `platform_coretemp_0` and `platform_coretemp_1` over multiple sensors. As some sensors are shared between chips and are inconsistent across DTNs, the most meaningful legend display was decided to include both sensor and chip data with all values displayed at once. Data is then filterable by chip and sensor combination as opposed to by chip or by sensor. By displaying chip and sensor as a pair, both are also displayed in alerts, making messages more illustrative of the concerns present at the time of an alert. Insufficient cooling of the device can result in hardware damage, making this graph vital to proactive troubleshooting.

Next is the Power Consumption visualization showcased in Figure 6.27. Power

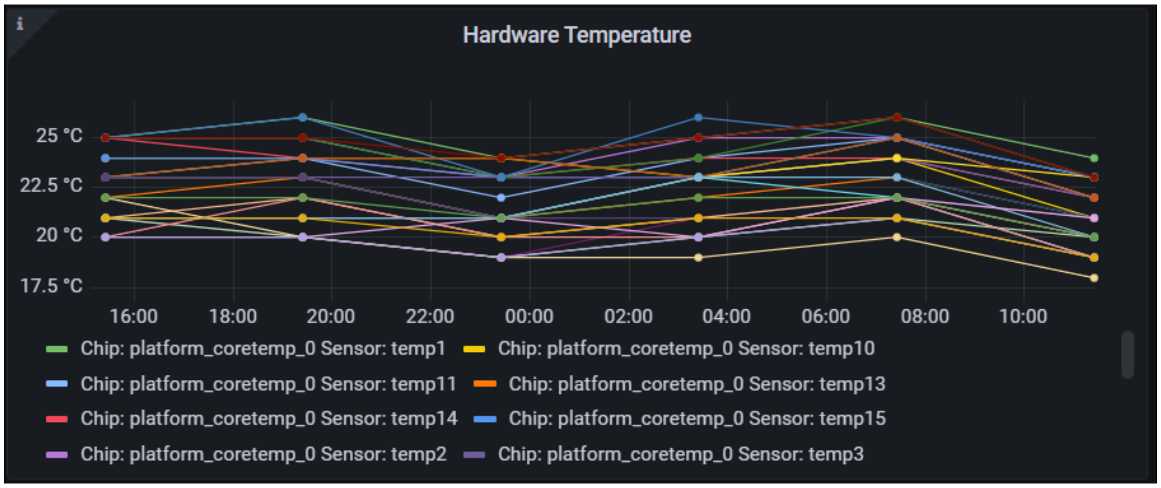


Figure 6.26: Information regarding the DTN's hardware temperature overtime separated by chip and sensor.

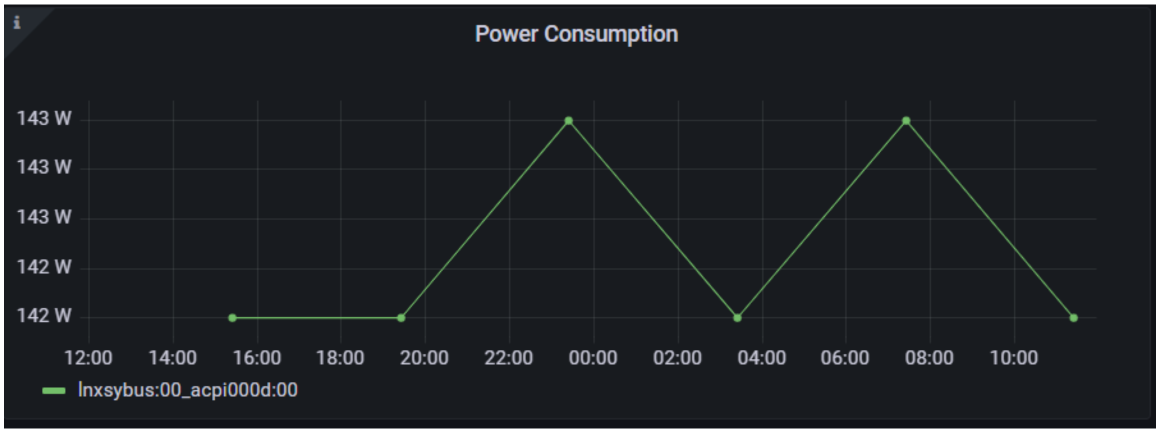


Figure 6.27: DTN power consumption over time in Watts.

Consumption is another graph-type visualization built in Grafana and features a legend for filtering like other graphs. This legend displays the chip information responsible for drawing power. Data is displayed in Watts over the specified time range shared with other visualizations. Power consumption serves as a method of viewing potential power surges and potential malfunctioning behavior from power supplies.

The final "DTN Health" visualization on the Cyberinfrastructure Data Transfer Dashboard is bandwidth usage by the machine overtime, reflected in Figure 6.28. This visualization dynamically filters the data so as to display only the network interface that is active for data transfers on the DTN. At this time, this is the only network interface on each DTN capable of connecting to the internet. Bandwidth usage is imperative for system administrators to view unexpected behavior unrelated to transfers and monitor performance reported by the device during a transfer compared to the reported transfer speed to determine which DTN may be experiencing poor performance.

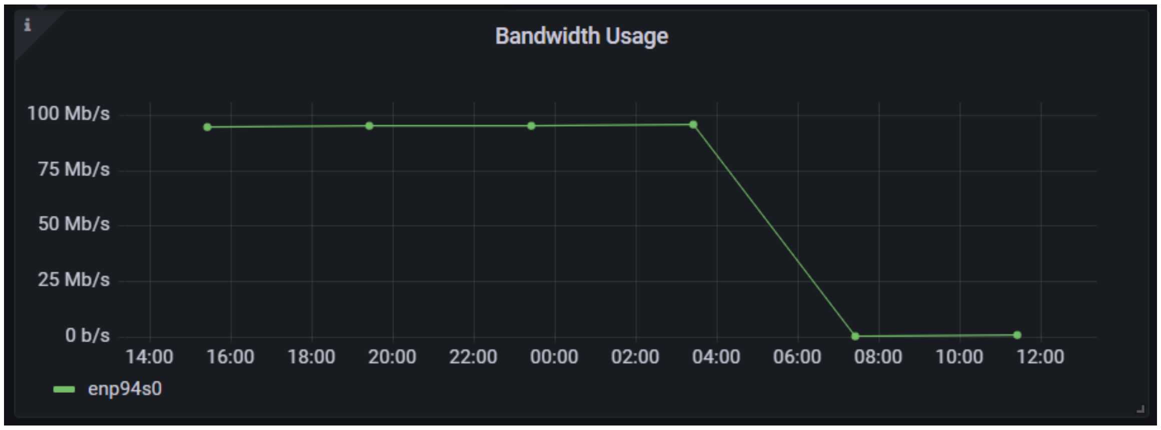
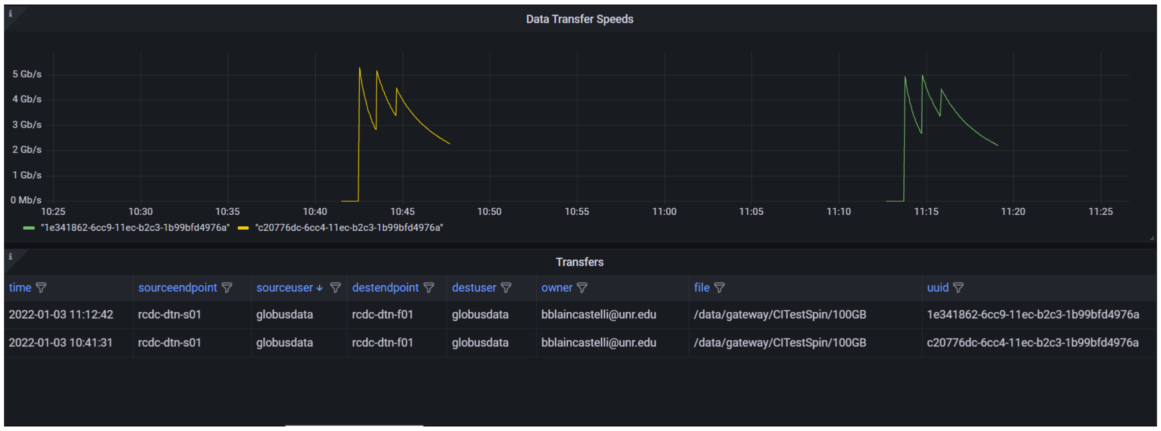


Figure 6.28: Bandwidth Usage overtime displayed as a graph.

Next, the Cyberinfrastructure Data Transfer Dashboard's "Data Transfer" tab has two data visualizations both shown in Figure 6.29. These visualizations show historical data transfer speeds in a graph and transfer metadata in a table. These visualizations display data for all DTNs over a specified range of time together with filter options in the table to filter by DTN, users, or transfer owners. The table provides the UUID of a transfer that can then be utilized to filter the graph through the legend filter options of graph-type visualizations in Grafana.

Finally, further metadata is displayed in a Geomap visualization, a geographical display of data that is dependent upon location. This Geomap is showcased in Figure 6.30 and consists of two parts. The left display, known as "IP Connections Data" utilizes an open street map that provides street names, national parks, and important geographical landmark data for a more detailed description of where transfers are occurring. This street map display was chosen because the left display also provides specific transfer information when a location is hovered over with the cursor, including latitude, longitude, city, and zip code. "IP Connections Data" limits trans-



Transfers

time	sourceendpoint	sourceuser	destendpoint	destuser	owner	file	uuid
2022-01-03 11:12:42	rcdc-dtn-s01	globusdata	rcdc-dtn-f01	globusdata	bblaincastelli@unr.edu	/data/gateway/CITestSpin/100GB	1e341862-6cc9-11ec-b2c3-1b99bfd4976a
2022-01-03 10:41:31	rcdc-dtn-s01	globusdata	rcdc-dtn-f01	globusdata	bblaincastelli@unr.edu	/data/gateway/CITestSpin/100GB	c20776dc-6cc4-11ec-b2c3-1b99bfd4976a

Figure 6.29: Information regarding data transfer performance overtime and metadata surrounding these transfers.

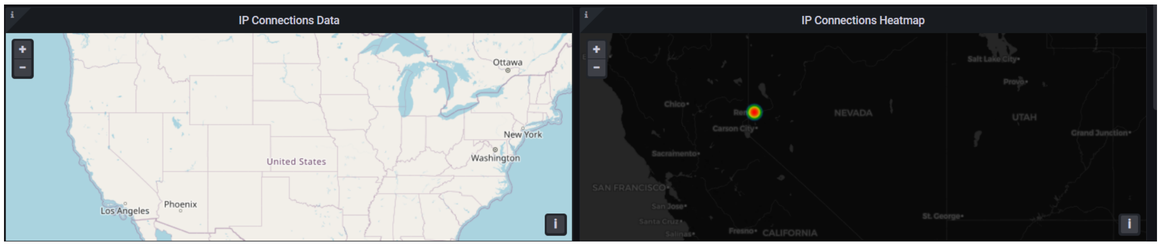


Figure 6.30: A geographical display of where data transfer are occurring. On the left is a time filterable map capable of providing additional metadata regarding specific transfers while the right display is a heatmap showing frequency of transfers in a specific location.

fers displayed according to the same time field as the data transfer table and graph previously defined. On the right is the “IP Connections Heatmap”, which groups transfers according to location. This Geomap provides a less detailed display, using the Grafana default map, due to the aggregated nature of the data. Data displayed on the heatmap is historical and does not filter by time. By providing all data, it is possible to map where the University is collaborating with or spot unusual traffic.

Chapter 7

User Study

This Chapter outlines the user study performed to assess the DDRIS Grafana dashboard. Section 7.1 provides insight on the background of the study, including its importance and the approval process for conducting the study. Next, Section 7.2 summarizes the participants included in the study, including study participant demographics. The hardware and software applications that were utilized in running the study are briefly discussed in Section 7.3. Fourth, Section 7.4 describes the procedure of running the designed user study in detail. Section 7.5 presents the independent and dependent variables collected and assessed as a part of the study to determine effectiveness of the dashboard. Finally, all tasks performed by the users are listed in Section 7.6.

7.1 Introduction

Research with a Human Computer Interaction component is to include a measure of usefulness for an application. In this context, usefulness is a combination of utility, relating to an application having the tools necessary for a user to utilize the application, and usability [55]. Usability extends beyond ease of use to encompass effectiveness, efficiency, ease of learning, error tolerance, and engagement. Software should be designed with all five of these goals for usability in mind. To measure this usefulness requires experimentation with the intended users and assessment to determine appropriate changes to further achieve these goals. The basis for running such an experiment has been outlined by I. Scott MacKenzie in his book "Human-computer interaction: An empirical research perspective" [47], which has been utilized in the design of this user study. This study aims to assess the usefulness of the DDRIS dashboard in respect to these principles, and evaluate future development ideas to improve upon the implemented iteration.

Before a user study may be conducted, the experiment must be approved by the Institutional Review Board (IRB) to ensure the experiment is ethical in nature. First, all instructors participating in conducting the user study must be certified through the Collaborative Institutional Training Initiative (CITI) to train instructors on how user studies involving humans are to be conducted and the rights of any participants. Certification for this user study was acquired through the "CITI Group 1: Social Behavioral Research Investigators and Key Personnel Group" [87] training course. This user study was then sent to the IRB for approval with a form package created of documents found in the Appendices. This package includes the IRB Application Cover Sheet found in Appendix A, the Social Behavioral Educational Research Protocol in Appendix B, the Consent Form found in Appendix C, and the questionnaires and recruitment email found in Appendices D-F.

7.2 Participants

Participants that took part in the user study consisted of current and former University of Nevada, Reno students with experience in the field of computer science. This experience includes computer science degrees, some computer science classes, and vocational experience with system administration. Participant sampling was based on convenience. The participants were contacted utilizing the recruitment email reflected in Appendix F. A total of 13 participants between the ages of 20 and 50 years old engaged in the study. While all participants were familiar with computer science, the level of experience with system administration and data transfers varied between participants to receive responses from a larger range of users. This variation served to properly reflect the differences in levels of experience and domain specific knowledge between potential students and staff that will be responsible for maintaining DDRIS.

7.3 Apparatus

The study was conducted on a Sager Notebook running Windows 10 Home Edition with a 9th Generation i7 64-bit processor and 16 GB of RAM. Due to the Covid-19 pandemic, the study was performed in person with a remote option through Zoom for participants who were unable or uncomfortable meeting in person. The software tools utilized in running this user study include Brave web browser [13], MobaXterm [51], Mousotron [11], and Grafana. Two websites were included in this user study, the DDRIS dashboard and Globus Management Console. The DTN NCAR-DA-9 was used for the command line section of this user study.

All web browsing was conducted on the Brave browser, shown in Figure 7.1. The Brave browser is a fast and secure web browser built on Chromium that serves as the department standard. The DDRIS dashboard, Globus Management Console, and one tab with the Google search engine were open for the user prior to starting the user study. Google was chosen as the search engine available to participants due to its popularity, so most participants would have experience utilizing the search engine. A search engine was provided to the users to more closely represent a typical work environment where questions could be answered through the internet. If a user is unfamiliar with how to complete a task, they then can search for how to complete the task within the task time limit. To compare performance against the Linux command line, a single SSH session was loaded on MobaXterm prior to the user starting the user study. MobaXterm opens a terminal environment for participants to use as shown in Figure 7.2.

Mousotron and Brave are used to aid in collecting accurate qualitative data. Mousotron was set prior to starting the first task. After each task, the output was recorded and the software was reset. The Mousotron interface can be seen in Figure 7.3 Brave's element inspector allows for performance data to be collected with persistence. These logs will be reviewed after a participant has finished the port-usage

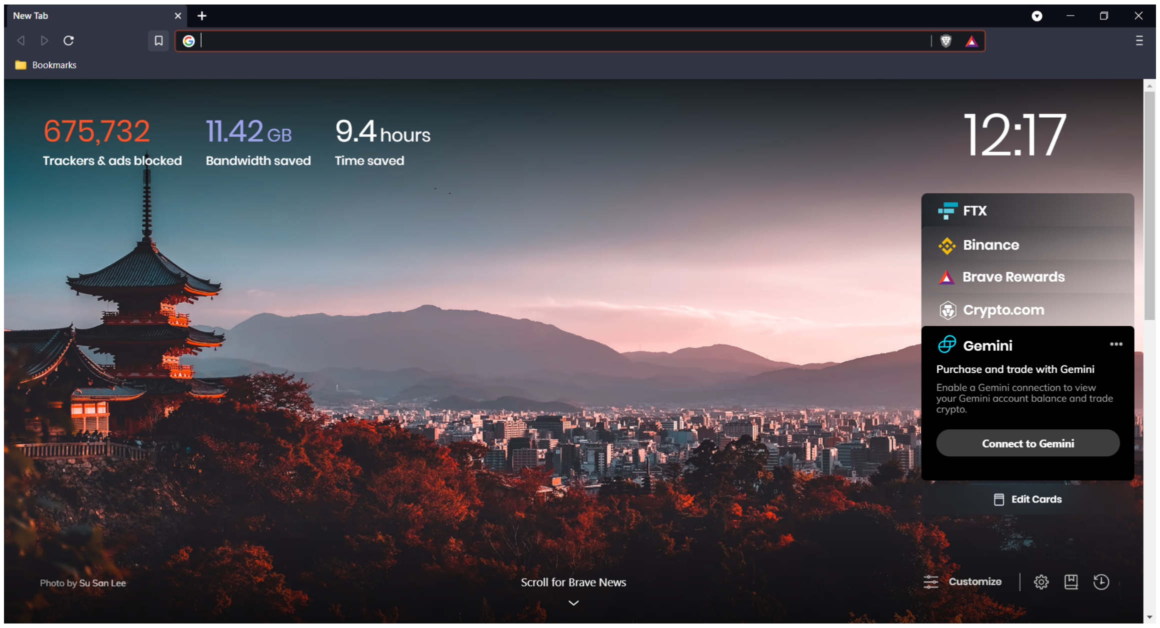
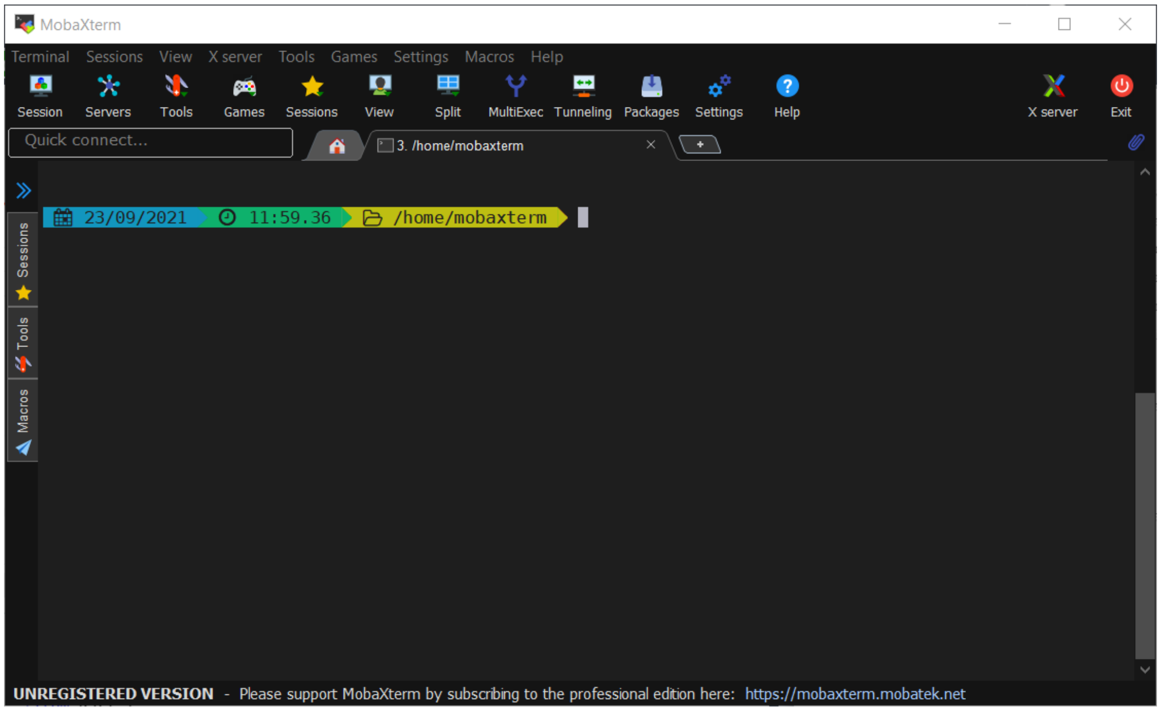


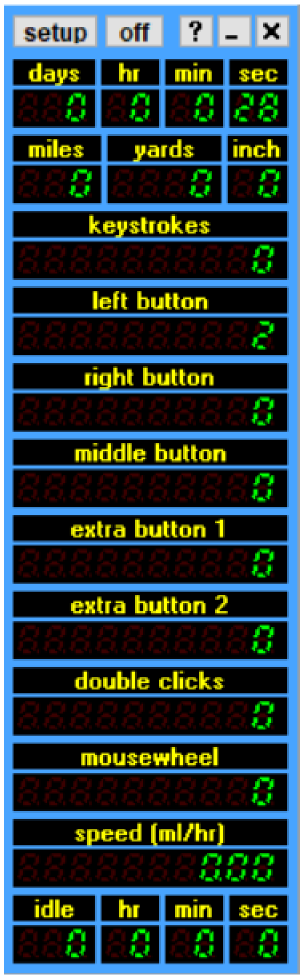
Figure 7.1: A screenshot of a Brave browser window, showing the layout of the browser.



```
23/09/2021  11:59.36  /home/mobaxterm
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

Figure 7.2: A single session open on MobaXterm, showing the terminal interface that was be available to participants.



days	hr	min	sec
0	0	0	28

miles	yards	inch
0	0	0

keystrokes 0
left button 2
right button 0
middle button 0
extra button 1 0
extra button 2 0
double clicks 0
mousewheel 0
speed (ml/hr) 0.00

idle	hr	min	sec
0	0	0	0

Figure 7.3: A screenshot of the Mousotron Interface.

questionnaire before being removed again to view page load times.

7.4 Procedure

The procedure began with recruitment, where potential users were sent an email asking the participant if they would like to participate in the study. There were two versions of this email, shown in Appendix F, based on if the participant was already familiar with the instructor. After a user agreed to participate, a meeting time was scheduled for in person or on Zoom, based on participant preference. In person sessions were completed in Edmund J. Cain Hall (EJCH) room 272C located on the UNR campus. Participants were met in the open lobby area of EJCH and led to the office in which the study was conducted. Upon starting the study, the instructor outlined the goals of the study and what would be expected of participants. It was established for participants that they were allowed to leave at any time and end their performance in the study if they no longer wanted to participate. Participants were then required to complete the consent forms attached in Appendix C to continue in the study.

Following the completion of the consent form, participants were to complete the entrance questionnaire outlined in Appendix D. This questionnaire was designed to collect general demographic information about participants, including age, gender, and education. Additionally, the survey provided insight on familiarity with Linux command line, data transfer solutions, data visualization, and web browsing, all concepts present in the study. After completing the survey, the instructor provided access to their laptop to complete the working session of the study on a controlled platform.

In the working session of this study, participants were asked to complete a series of tasks on three platforms to evaluate the effectiveness of the DDRIS dashboard design. The study consisted of this single working session with two parts. In part one, users performed tasks on the DDRIS dashboard and the native Globus console to evaluate effectiveness in locating data transfer metadata. Both web pages were loaded in different browser windows prior to beginning the study. Users were to perform all tasks on the Globus console prior to viewing the DDRIS dashboard with only one task provided at a time. After all tasks were completed, participants were asked to perform the tasks again on the DDRIS dashboard. Participants were provided four minutes and 30 seconds to complete each task on either platform before the task is considered failed. When a task is failed, users were asked to continue to the next task without completing the previous task. This ensured that too much time was not dedicated to a single task, and to aid in addressing user frustration in the event they were stuck. Participants were instructed to pause between tasks to log each task's completion data using Mousotron. In part two, users performed tasks on the DDRIS dashboard as well as Linux command line to evaluate speed and accuracy of finding system health information for DDRIS systems. Users were led to the MobaXterm console where access to a machine was already established without sudo or root privileges. It was expressed to users they have access to one browser tab to utilize the Google search engine to find correct methodologies and were given information on the operating system to aid in finding the correct commands. Starting on Linux command line, participants were presented a single task at a time until all tasks were completed, like in part one. After each task was presented, users were asked if they already know how to complete the task and reminded to use Google if they did not. Participants were provided only two minutes and 30 seconds before the task was considered failed. When all tasks were completed on command line for part two, the tasks were to be completed on the DDRIS dashboard. As was the case in part one, participants were once again instructed to pause between tasks on command line and the dashboard to output Mousotron metrics.

Once part one and part two were both completed, participants were to complete the exist survey outlined in Appendix E. The post-test survey was designed to aid in assessing the user experience and determine the usefulness of the dashboard through both quantitative and qualitative data. The first six questions of the survey were quantitative where data is collected on a 1-5 Likert scale. On this scale, a 1 represented a strong negative and a 5 represented a strong positive. These questions addressed both the utility and usability of the dashboard by asking the helpfulness of the dashboard in terms of effective data display and ease of navigation and task completion across platforms. The remaining three questions were designed to collect more qualitative data through short answer. These questions allowed for user feedback to determine a future direction for research and development by determining what was lacking from the dashboard.

7.5 Design

Measurements were taken during the experiment as participants completed tasks. First, the variables for Part One of the study will be discussed. Then an overview of the variables for Part Two of the study will be outlined. Lastly, qualitative data belonging only to the DDRIS dashboard will be discussed. For both parts of the study, the within-subjects method defined by MacKenzie [47] was utilized such that test conditions were repeated for all participants as opposed to the between-subjects method where participants would be exposed to only one interface.

Part One of the study contained five dependent variables as shown in Table 7.1. The first dependent variable is the amount of time a participant takes to complete a task measured to the seconds. The second and third measurements are the integer values of left and right mouse clicks respectively. Fourth, task completion status is a boolean value determined by whether or not the participant is capable of completing the task within the four minute time limit. If the participant has not completed the task within four minutes, then the task is considered failed, denoted by a 1. Lastly, the comparative difficulty scores will be assessed. These results are gathered through the Exit Questionnaire and are on a Likert scale of 1-5. Part One had only one independent variable. This was the interface at two levels, the Globus Online Graphical User Interface and the DDRIS Dashboard.

Table 7.1: Overview of variables measured in Part One of the user study.

User Study Variables Part One	
Measurement Number	Description
1	Time to complete the task
2	Number of left mouse clicks
3	Number of right mouse clicks
4	Task completion status
5	Difficulty scores

Part Two of the study had six dependent variables as shown in Table 7.2 where measurements 1-4 are the same as in Part One. Measurement five, task completion status, has an updated threshold of only two minutes and thirty seconds, so users have less time to complete each task before it is considered failed. The difficulty scores are from the Exit Questionnaire like in Part One where they are from a 1-5 Likert scale. Lastly, the number of questions asked by the participant on how to complete a task

Table 7.2: Overview of variables measured in Part Two of the user study.

User Study Variables Part Two	
Measurement Number	Description
1	Time to complete the task
2	Number of left mouse clicks
3	Number of right mouse clicks
4	Task completion status
5	Difficulty scores
6	Number of questions asked

and websites visited will be recorded. These questions include both to the moderator and number of Google searches.

There were two independent variables associated with Part Two of the study. One independent variable is the interface at two levels, Linux (Ubuntu) Command Line Interface (CLI) and the DDRIS Dashboard. The second is preexisting familiarity with the CLI at three levels, not familiar, moderately familiar and very familiar.

7.6 Tasks

As mentioned previously, the study was broken into two parts. Part One of the study compared effectiveness of the DDRIS Dashboard to the Globus Transfer Management Console to obtain transfer records. Part two compared effectiveness of the DDRIS Dashboard to Linux Command Line to obtain system status information. First, the tasks associated with Part One will be discussed in detail. This will be followed by the tasks in Part Two.

Part One of the user study involved four tasks as shown in Table 7.3. The participant was to find the transfer speed of one transfer on DTN-F on both platforms. Because the Globus Management Console does not display historical records, this data needed to be within a month of the user study. Participants were allowed to pick any transfer on this date, but it had to involve DTN-F as the transfer source or destination. After finding the transfer speed of this transfer, the participant was to state the local user on the source and destination recorded as being responsible

Table 7.3: Overview of tasks participants were asked to perform in Part One of the user study.

User Study Tasks Part One		
Task Number	Name	Description
1	Find Transfer Speed	Find the ending transfer speed for one transfer that completed on any DTN
2	Transfer User	Find the local users responsible for a transfer's read and write operations on the source and destination machines
3	Transfer Owner	Find the Globus user that initiated the latest transfer
4	Current Transfers	Find how many transfers are currently running

for the transfer. This is the user on the machine responsible for the read and write operations involved in the transfer. From here, the user had to navigate back to the current date to find the Globus user who initiated the latest transfer. This differs from the local user in that this corresponds with a Globus username, not a local account on the machine. For the final task, the participant was to count how many tasks were currently running. Each task for Part One was limited to four minutes before the task would be considered failed.

Part Two of the study involved the three tasks shown in Table 7.4. All tasks completed in Part Two were to be completed for NCAR-DA-9, where participants would have an ssh terminal to run Linux commands. For the first task, users were told that Globus Collections used the block device "sdb" and were instructed to find how much of the filesystem was left available for Globus Collections. This could be completed several ways, but the recommended method was through the command `df -h` to display filesystem information in a human readable format. This was not provided by the instructor to test knowledge, but if asked, the instructor would recommend the man page for this command. For the second task, users were to find the current temperature of any on-board component. Through command line, this could be completed by running the `sensors` command. Like in the previous task, this was not provided to participants. For this task, instructors would encourage participants

Table 7.4: Overview of tasks participants were asked to perform in Part Two of the user study.

User Study Tasks Part Two		
Task Number	Name	Description
1	Find Available File System Space	Find how much of the file system is available for Globus Collections
2	Find Temperature	Find the current temperature of on-board components
3	Find Memory Usage	Find how much memory is currently in use

to search for the answer on Google. For the final task, the participant was to find how much memory was currently in use. This could by completed through the command `htop` but was once again not provided to participants. All tasks in Part Two were limited to two minutes and 30 seconds before being considered failed to provide sufficient time to complete tasks without forcing participants to remain on the task until completion.

Chapter 8

Results and Data Analysis

The results of the user study and tuning the DTNs are detailed in this Chapter. First, the results from the pre-usage survey will be discussed in Section 8.1. Performance on tasks during he user study are then outlined in Section 8.2. Next, Section 8.3 reviews results from the post-usage questionnaire. A discussion of these results is then provided in Section 8.4. Finally, a detailed comparison of initial transfer speeds and final transfer speeds as a result of DTN tuning are detailed in Section 8.5.

8.1 Results from the Entry Questionnaire

Participants ranged from age 22 to age 50 with the majority of participants in their mid-20s to early-30s. Figure 8.1 shows participant responses when asked their age. Other demographic information collected indicated that the majority of participants held a Bachelor's or Master's degree as their highest form of education with one user currently pursuing their Bachelor's. All participants currently attend or have graduated from the University of Nevada, Reno. The majority of participants identify as male with only two participants identifying as female.

After demographic data was collected, participants were asked to rank their familiarity with various concepts present in the user study. All participants indicated a familiarity with web browsing with 10 participants indicating that they were very familiar, and two participants indicating some familiarity. Specificity increased following this question by asking participants how familiar they were with web dashboards.

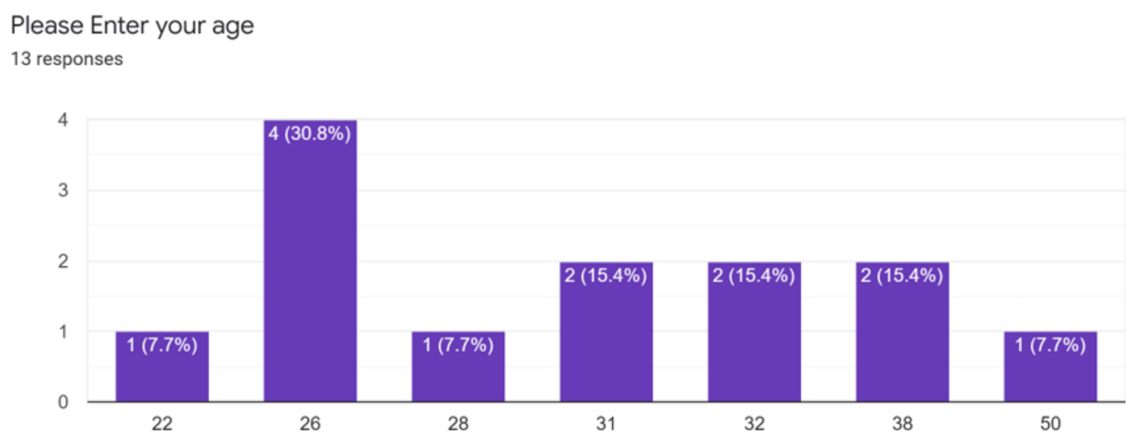


Figure 8.1: Bar chart displaying participant ages.

As shown in Figure 8.2, out of the 13 participants, 11 reported significant familiarity with dashboards. Two participants indicate some familiarity without confidence in the subject. Participants were then asked how often they utilized data visualization. 10 participants indicate frequent usage. One participant responded that they never use data visualization. This was followed by asking participants their familiarity with data transfer solutions. The majority answered they were not familiar with data transfer solutions while those with familiarity mostly responded with knowledge of FTP and SFTP. Finally, participants were asked their familiarity with using Linux Command Line, as presented in Figure 8.3. The majority of participants registered a basic familiarity of command line with one participant indicating a lack of familiarity.

8.2 Results from Tasks

As defined in Chapter 7, following completion of the pre-usage questionnaire, participants were instructed to perform two sets of tasks. In Part One, participants performed the same tasks on the DDRIS dashboard and Globus native dashboard. The results of this section of the study are provided in Subsection 8.2.1. Part Two of the study saw participants performing tasks on the DDRIS dashboard and using a

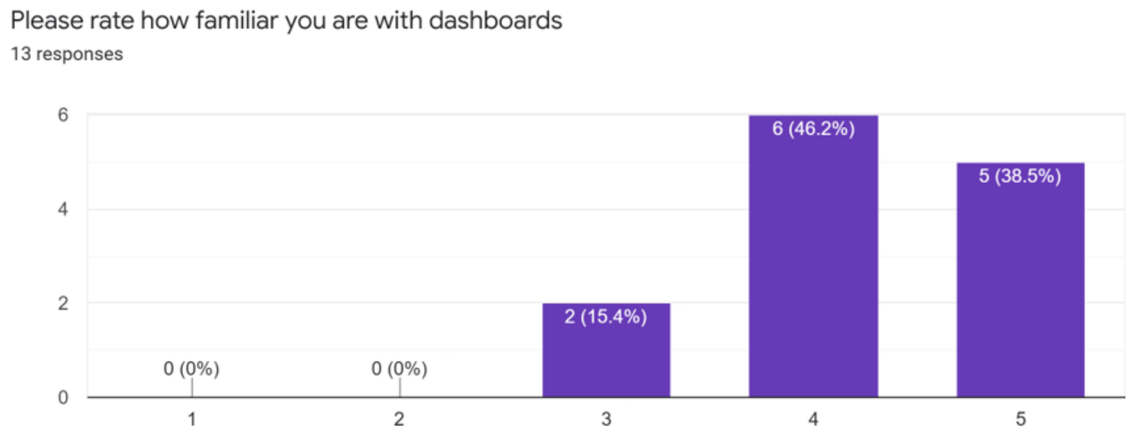


Figure 8.2: Bar chart displaying participant familiarity with dashboards.

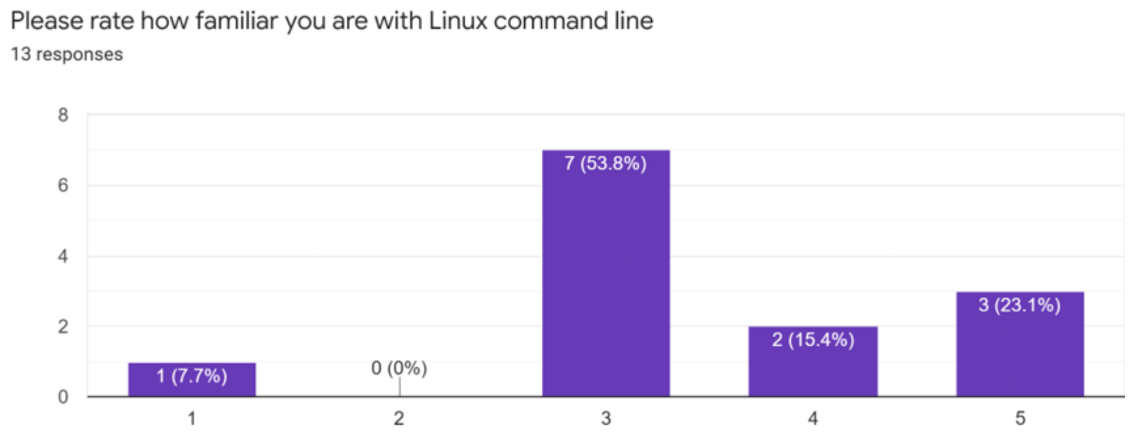


Figure 8.3: Bar chart displaying participant familiarity with using Linux command line.

Linux terminal. These results are shown in Subsection 8.2.2. For both sections of the study, after data collection, results were aggregated and graphed to show skewedness, variance and outliers. The difference in results was then calculated and statistical analysis was performed to determine if the differences were significant.

8.2.1 Part One

During Part One of the user study, participants were asked to perform four tasks on the native Globus dashboard followed by repeating the tasks on the DDRIS dashboard. To maintain consistency between sections, DDRIS measurement results are listed first such that $\Delta = DDRIS - Globus$. All measurement results are discussed separately before evaluating the difference in performance between platforms. The statistical significance to determine a difference in the results between using DDRIS and using Globus to complete tasks was performed utilizing a two-tailed paired t-test with DDRIS as the first data group.

Participant task completion time, listed in seconds, for each task on the DDRIS dashboard is listed in Table 8.1. All participants successfully completed every task in the allotted amount of time. The mean and sample standard deviation (SD) are provided as the last two rows of the table to provide a data summary. Out of the four tasks, the most participants struggled most with Task 2, showing a mean of 83.54 seconds and standard deviation of 61.48 seconds. By contrast, Task 3 had the fastest performance with a mean of 13.31 seconds and standard deviation of 5.88 seconds. Figure 8.4 supplements these results by visualizing this performance data as a Box and Whisker Plot, showing distribution and upper quartiles, median, and lower quartiles for each task. According to this data summary, Task 1 has a large right skew due to a single outlier present on the upper bound. Task 2 and Task 3 have a moderate positive skew towards a higher number of seconds required to complete the task. Task 4 also has a moderate right skewness of 0.73.

In contrast, participant task completion time utilizing Globus is visible in Table 8.2. Due to errors in the data, participant 12's Task 1 has been removed from all calculations in Part One. The results show users struggled the most completing Task 1 with a mean of 119.75 seconds with a wide difference in completion time as supported by the standard deviation of 52.49 seconds. By contrast, participants tended to quickly complete Task 2 and Task 3. According to Figure 8.5, Task 1 shows

Table 8.1: Overview of time taken to complete each task on the DDRIS dashboard in Part One of the user study.

Task Completion Time (in seconds) using DDRIS				
Participant Number	Task 1	Task 2	Task 3	Task 4
1	32	225	22	26
2	45	107	6	42
3	71	17	8	26
4	88	118	18	88
5	5	144	8	25
6	110	101	8	25
7	20	76	26	15
8	22	18	13	34
9	60	130	12	93
10	82	35	11	87
11	35	28	15	49
12	110	38	11	32
13	235	49	15	73
mean	70.31	83.54	13.31	47.31
SD	60.03	61.48	5.88	27.94

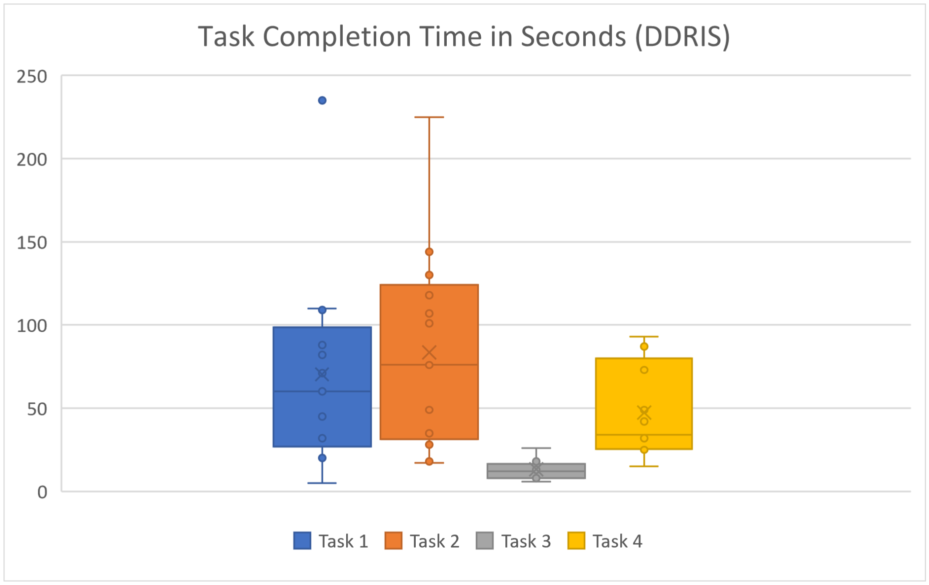


Figure 8.4: A Box and Whisker Plot showing data analysis for task time completion utilizing DDRIS.

the lowest, moderate right skew (0.71). Task 1 also shows the highest variance and highest completion times compared to other tasks. Task 2 and Task 4 are heavily skewed to the right, with Task 4 having two outliers on the upper bound. Task 3 is moderately skewed to the right.

Table 8.2: Overview of time taken to complete each task on the Globus dashboard in Part One of the user study.

Task Completion Time (in seconds) using Globus				
Participant Number	Task 1	Task 2	Task 3	Task 4
1	111	45	39	30
2	227	35	21	32
3	182	16	10	21
4	105	23	12	48
5	114	28	18	19
6	167	14	14	111
7	49	27	19	30
8	66	24	11	18
9	92	31	16	127
10	69	14	25	44
11	106	16	35	19
12	x	54	12	23
13	149	19	30	39
mean	119.75	26.62	20.153	43.15
SD	52.42	12.25	9.48	35.17

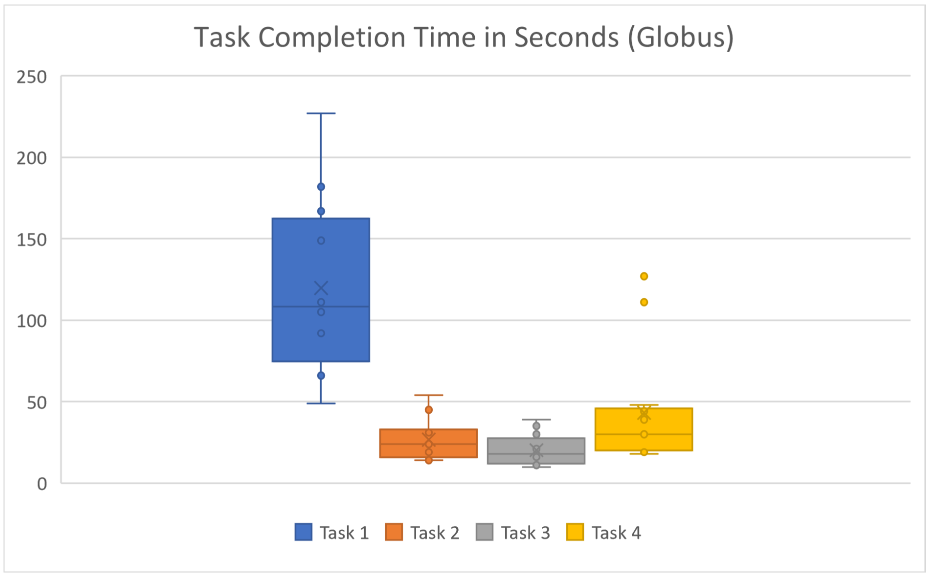


Figure 8.5: A Box and Whisker Plot showing data analysis for task time completion utilizing Globus.

Comparing the results, without Participant 12's Task 1, on average participants were able to complete tasks using DDRIS faster than the Globus native dashboard on two out of four tasks as shown in Table 8.3. Results from Task 1 on DDRIS and Globus indicate that there is a significant statistical difference between the results and that users were capable of completing tasks faster on the DDRIS platform, $t(11) = -2.7$, $p = .02$. Additionally, results from Task 3 on DDRIS and Globus indicate users were also capable of completing this task significantly faster utilizing DDIRS, $t(12) = -2.74$, $p = .02$. However, Task 2's results with DDRIS and Globus show participants' ability to complete tasks faster using the Globus native dashboard are statistically significant, $t(12) = 3.58$, $p = .004$. Lastly, the results of Task 4 show no statistical difference in participant completion times between DDRIS and Globus, $t(12) = 0.43$, $p = .67$.

Table 8.3: Overview of ΔTask Completion Time, where Δ represents the difference between the DDRIS dashboard value and the Globus dashboard value and statistically significant differences are denoted by a boldfaced mean value.

Δ Task Completion Time (in seconds)				
Participant Number	Task 1	Task 2	Task 3	Task 4
1	−79	180	−17	−4
2	−182	72	−15	10
3	−111	1	−2	5
4	−17	95	6	40
5	−109	116	−10	6
6	−57	87	−6	−86
7	−29	49	7	−15
8	−44	−6	2	16
9	−32	99	−4	−34
10	13	21	−14	43
11	−71	12	−20	30
12	x	−16	−1	9
13	86	30	−15	34
mean	**−52.67**	**56.92**	**−6.85**	4.15

The next metric, was a measurement of the number of left mouse clicks from participants when completing a task. Results for users on the DDRIS dashboard can be viewed in Table 8.4, and show a higher number of clicks for Task 1 and Task 2 with an average of 12.54 and 12.46 clicks respectively. This appears to correlate with

Table 8.4: Overview of left mouse clicks to complete each task on the DDRIS dashboard in Part One of the user study.

Left Mouse Clicks using DDRIS				
Participant Number	Task 1	Task 2	Task 3	Task 4
1	8	38	6	4
2	5	16	2	8
3	35	1	1	4
4	8	20	4	5
5	0	8	0	0
6	10	24	0	0
7	4	18	0	6
8	4	8	2	10
9	6	10	3	14
10	10	3	3	13
11	8	3	4	10
12	24	5	4	3
13	41	8	3	12
mean	12.54	12.46	2.46	6.85
SD	12.66	10.48	1.85	4.71

time taken for a task where Task 1 and Task 2 also featured the highest values. The result plot, provided in Figure 8.6, also shows the high spread for both of these tasks. While Task 1 is approximately symmetric ($skewness = 0.3$), Task 1 also features one outlier on the upper bound of 41 clicks. Task 2 has a moderate right skew. Task 3 shows a right skew with a smaller standard deviation and better average performance than the previous tasks. Task 4 has a larger variance and mean than Task 3 while maintaining lower values than the first two tasks, correlating with task completion time.

Globus features low performance for Task 1 and high performance for the remaining three tasks as provided by Table 8.5. The number of clicks for participant 12 in Task 1 has been removed again due to data collection errors. Utilizing the remaining data, it can be seen that Task 1 had the highest number of clicks with a mean of 24.70 clicks until task completion. Further information, illustrated in Figure 8.7, show a large spread in left mouse clicks for Task 1 with no outliers and a moderate, left skew. Tasks 2, 3, and 4 feature little variation in results with a moderate, right

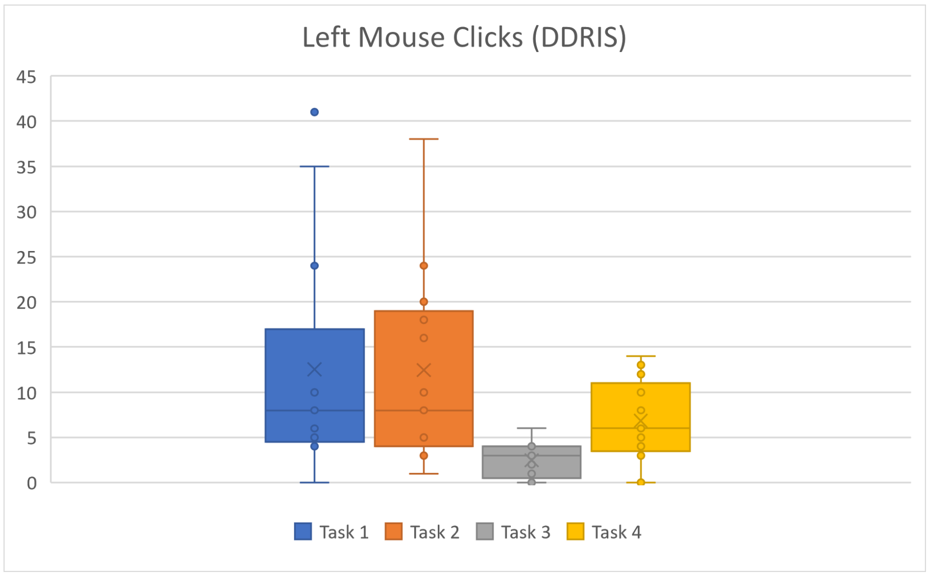


Figure 8.6: A Box and Whisker Plot showing data analysis for left mouse clicks per task utilizing DDRIS.

skewness for Task 2 and near symmetrical distribution for Task 3. Neither Task 2 or Task 3 feature outliers. Task 4 has a moderate, left skew with two outliers on the upper bound and one outlier on the lower bound.

Differences in the results for each task can be viewed in Table 8.6 where it appears participants utilized fewer clicks on DDRIS for Task 1 and Task 3, and participants performed more left clicks on DDRIS for Task 2 and Task 4 when compared to Globus. However, statistical analysis shows that the difference between DDRIS and Globus for Task 1 is not statistically significant, $t(11) = -1.68$, $p > .05$. As such, no conclusion can be drawn regarding the number of left clicks for Task 1. Similarly, Task 3 shows no statistically significant difference in results, $t(12) = -0.10$, $p > .05$, nor does Task 4, $t(12) = 0.92$, $p > .05$. Task 2 shows a statistically significant difference in the number of left mouse clicks used between DDRIS and Globus, supporting that fewer left clicks were utilized to complete Task 2 on Globus, $t(12) = 3.37$, $p = .006$.

Right clicks are a method of viewing error rate, however most participants did not utilize right clicks to navigate the dashboard and would instead choose to navigate through mistakes or left click on the back button in the browser. As such, results for

Table 8.5: Overview of left mouse clicks to complete each task on the Globus dashboard in Part One of the user study.

Left Mouse Clicks using Globus				
Participant Number	Task 1	Task 2	Task 3	Task 4
1	48	9	7	6
2	54	4	2	4
3	35	0	0	5
4	13	0	1	5
5	1	1	0	0
6	21	2	3	9
7	15	8	4	5
8	22	3	3	4
9	12	5	3	13
10	6	4	4	6
11	39	3	3	6
12	x	4	5	5
13	38	3	3	6
mean	24.70	3.48	2.85	5.50
SD	17.08	2.70	1.93	2.95

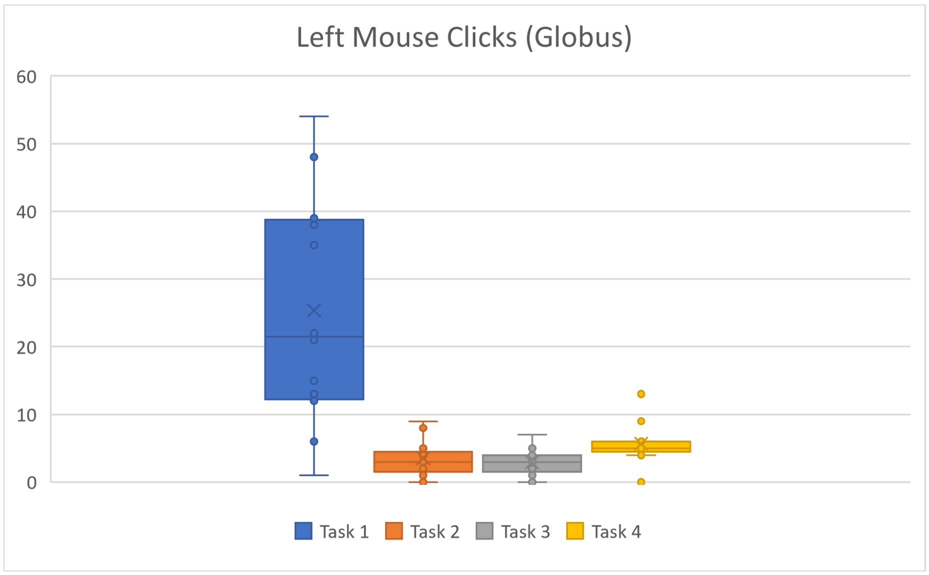


Figure 8.7: A Box and Whisker Plot showing data analysis for left mouse clicks per task utilizing Globus.

Table 8.6: Overview of the difference in left mouse clicks between DDRIS dashboard usage and Globus dashboard usage.

ΔLeft Mouse Clicks				
Participant Number	Task 1	Task 2	Task 3	Task 4
1	−40	29	−1	−2
2	−49	12	0	4
3	0	1	1	−1
4	−5	20	3	0
5	−1	7	0	0
6	−11	22	−3	−9
7	−11	10	−4	1
8	−18	5	−1	6
9	−6	5	0	1
10	4	-1	−1	7
11	−31	0	1	4
12	x	1	−1	−2
13	3	5	0	6
mean	−13.75	**8.92**	−0.46	1.15

right clicks were low across all tasks. Task 1 and Task 4 are the only two tasks that resulted in right click navigation with a mean of 0.38 clicks for Task 1 and 0.15 clicks during Task 4. The Box Plot, visible in Figure 8.8, for this measurement shows that all data points outside of 0 clicks are an outlier. As such, there are two outliers for Task 1 on the upper bound and two outliers for Task 4 on the upper bound.

Similar to when using DDRIS, participants rarely utilized right clicks to navigate the Globus dashboard. Right clicks were used in Task 1 and Task 4, however, as illustrated by the associated Box Plot in Figure 8.9, once again all values other than 0 were outliers. Task 1 and Task 4 both feature two outliers on the upper bound. As such, the difference in values between DDRIS and Globus usage can be assumed to be 0 clicks.

8.2.2 Part Two

Part Two of the study had participants perform three tasks on the DDRIS dashboard followed by the same three tasks using Linux's Command Line Interface. Results for DDRIS measurements are listed prior to the corresponding Linux CLI measurement

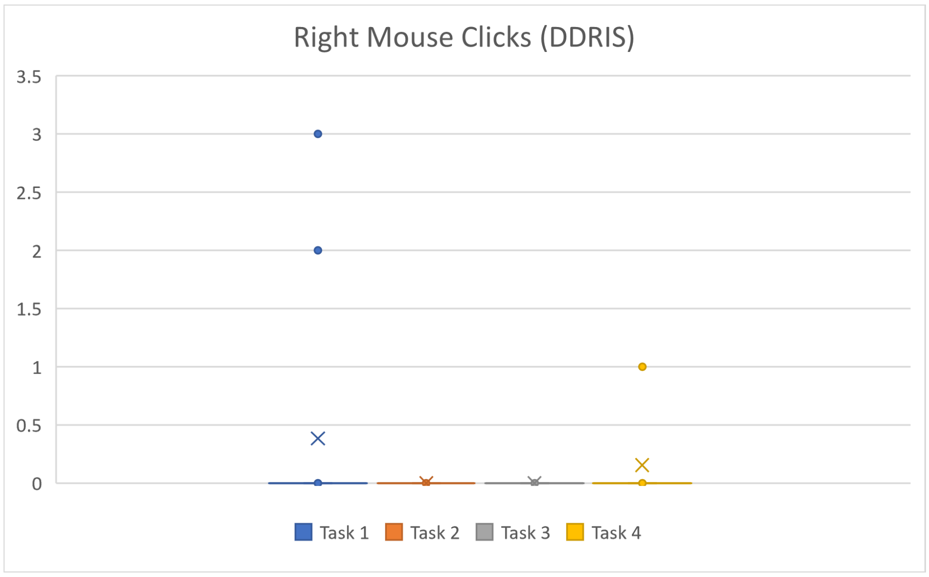


Figure 8.8: A Box and Whisker Plot showing data analysis for right mouse clicks per task utilizing DDRIS.

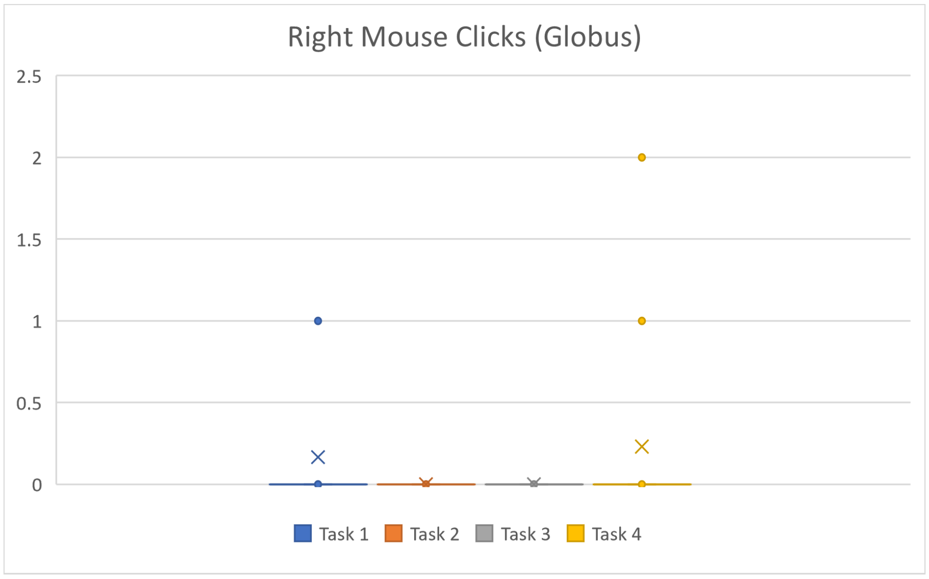


Figure 8.9: A Box and Whisker Plot showing data analysis for right mouse clicks per task utilizing Globus.

such that $\Delta = DDRIS - CLI$. The difference in results is then analyzed, except in the case when no comparative analysis is present. If a task was failed by the participant, the resulting data is not included in calculations and the corresponding task completion measurement on the opposing platform is not included in the statistical analysis.

The first measurement collected in Part Two was time to complete a given task by a participant, displayed in Table 8.7 for DDRIS. Participants performed best on Task 3 with an average completion time of 20.08 seconds and worst on Task 1 with an average completion time of 33.54 seconds. This is supported by the corresponding Box Plot available in Figure 8.10, which shows decreasing amounts of time taken to complete each task. Task 1 has a slight positive skew with no outliers. Task 2 has a slight right skew with one outlier on the upward bound of 48 seconds to complete the task. This task features the smallest variation in measurement results. Task 3 also has a positive skew and has two outliers on the upward bound. This task featured the highest deviation in completion times.

Table 8.7: Overview of time taken to complete each task on the DDRIS dashboard Part Two of the user study.

Task Completion Time (in Seconds) using DDRIS			
Participant Number	Task 1	Task 2	Task 3
1	39	28	18
2	25	11	11
3	20	26	52
4	33	48	64
5	43	25	9
6	65	30	11
7	33	33	12
8	13	20	9
9	45	16	13
10	13	24	13
11	21	25	12
12	63	28	14
13	23	22	23
mean	33.54	25.85	20.08
SD	17.02	8.85	17.41

The amount of time to complete each of the three tasks can be viewed in Table 8.8. All instances where participants were unable to complete a task in the two minutes and thirty seconds allotted for completion have been removed from calculations. Task 1 and Task 2 both had one instance of failure and as such have 12 data points. Two

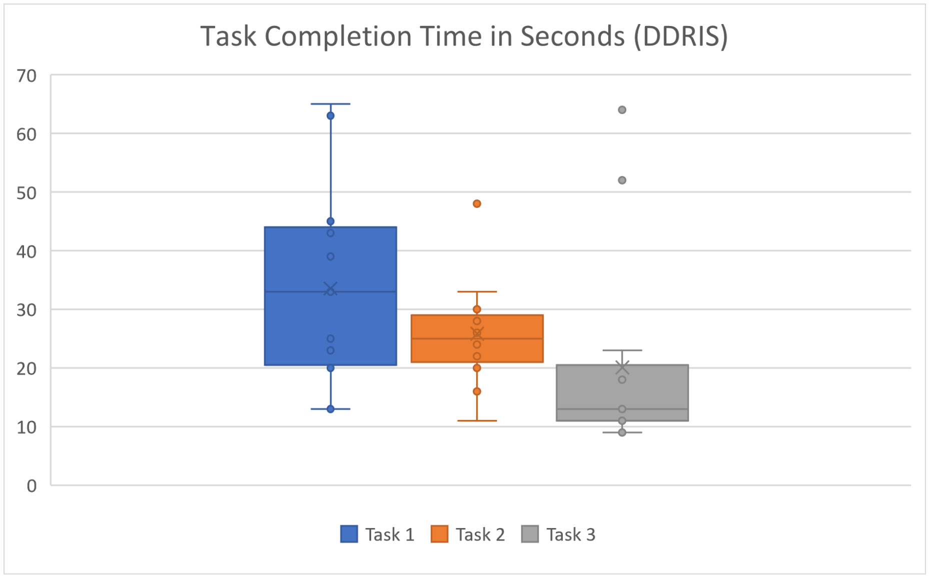


Figure 8.10: A Box and Whisker Plot showing data analysis for task completion times using DDRIS during Part Two of the user study.

participants were unable to complete Task 3 in the allowed time frame, so Task 3 has 11 associated data points. These removed points are removed for each metric. Viewing the remaining data, participants completed Task 3 fastest on average at 65.73 seconds and Task 2 the slowest with an average of 99.75 seconds. Most participants visited multiple sites to find the commands for Task 2 and Task 3. Task 3 saw a large number of participants copy commands from the internet without reading the command or information surrounding it. As a result, several participants first answered with the total amount of memory on the machine before realizing the metric they found was not the metric asked for. These participants were asked to continue looking for the correct answer. This likely contributed to the larger failure rate for Task 3. Task 3 also saw the highest number of participants able to complete the task without the use of the internet, which likely is why the average time taken is the lowest when instances of task failure are removed.

Only successful instances of tasks were graphed in Figure 8.11 as completion of the task is required by this metric. Viewing this Box Plot, Task 1 appears to be approximately symmetric and has no outliers. With failed instances removed, this

Table 8.8: Overview of time taken to complete each task through Linux command line Part Two of the user study where failure to complete the task in the allotted amount of time is marked “X”.

Task Completion Time (in Seconds) using Command Line			
Participant Number	Task 1	Task 2	Task 3
1	69	147	45
2	57	X	69
3	124	150	42
4	X	118	43
5	57	144	140
6	76	86	78
7	45	70	36
8	66	46	57
9	94	121	X
10	94	103	120
11	91	45	76
12	105	68	17
13	110	99	X
mean	82.33	99.75	65.73
SD	24.35	37.51	36.84

task has the lowest variance in response times. Task 2 has the highest variance in responses but no outliers and a normal distribution. This likely correlates with how much time was spent on the internet searching for commands to complete this task. Task 3 has one outlier on the upper bound of 140 seconds to complete the task and a moderate right skew.

Compared task completion times are available in Table 8.9. Comparisons are only present for instances where the task was successful. If a task was failed on command line during a session, the associated results on DDRIS were removed from the associated T-test for that task. The difference is statistically significant for Task 1, $t(11) = -5.42$, $p < 0.001$. Significantly better performance was also displayed during Task 2 with high confidence, $t(11) = -6.68$, $p < 0.001$. Lastly, participants took significantly less time to complete tasks on DDRIS compared to command line for Task 3, $t(10) = -3.19$, $p = 0.01$. This indicates that participants were capable of performing all three tasks faster on DDRIS than using command line.

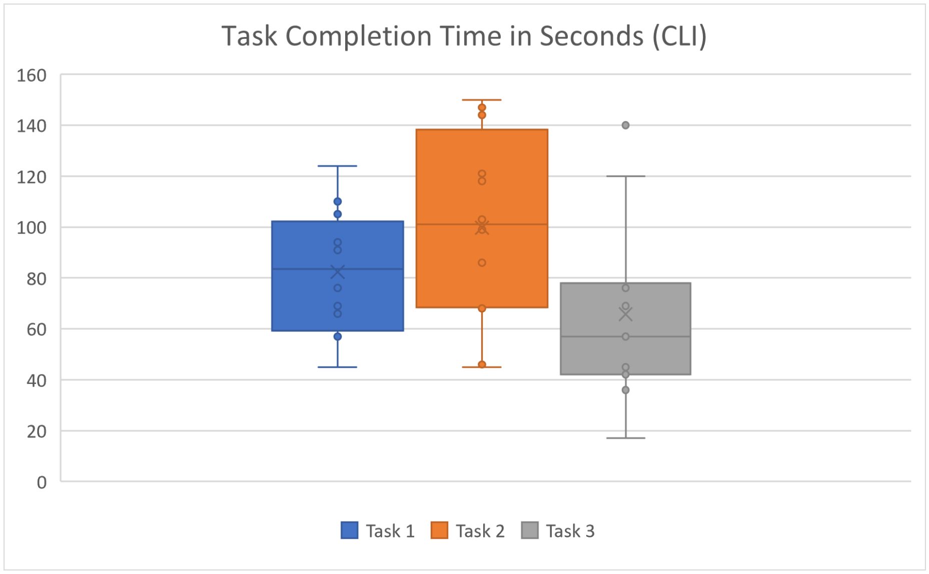


Figure 8.11: Task completion time in seconds using Linux command line.

Table 8.9: Overview of the difference in Task Completion Time between DDRIS dashboard usage and Command Line Interface (CLI) usage.

ΔTask Completion Time (in seconds)			
Participant Number	Task 1	Task 2	Task 3
1	−30	−119	−27
2	−32	X	−58
3	−104	−124	10
4	X	−70	21
5	−14	−119	−131
6	−11	−56	−67
7	−12	−37	−24
8	−53	−26	−48
9	−49	−105	X
10	−81	−79	−107
11	−70	−20	−64
12	−42	−40	−3
13	−87	−77	X
mean	**−48.75**	**−72.67**	**−45.27**

The second measurement collected was left mouse clicks required to complete each task. Results utilizing DDRIS are displayed in Table 8.10 and show participants used the most left mouse clicks to complete Task 1 when compared to other tasks. Task 1 also had the highest variation in results. This is likely due to there being two

Table 8.10: Overview of left mouse clicks to complete each task on the DDRIS dashboard in Part Two of the user study.

Left Mouse Clicks using DDRIS			
Participant Number	Task 1	Task 2	Task 3
1	6	5	4
2	4	2	2
3	2	2	8
4	5	4	4
5	12	4	4
6	3	3	4
7	10	12	4
8	3	3	3
9	5	3	3
10	3	3	3
11	3	3	3
12	3	3	3
13	3	3	6
mean	4.77	3.85	4.15
SD	3.00	2.58	1.63

methods of completing the task. Some participants utilized the gauge visualization while others utilized the graph display and performed mental math to reach the answer. The corresponding Box Plot, available in Figure 8.12, shows a high positive skew for Task 1 due to two outliers on the upper bound of 10 and 12 clicks. Task 2 has the highest skewness of all three tasks with one outlier of 12 clicks on the upper bound. Task 3 also features high positive skewness in the results. This task has no outliers.

Results using Linux command line can be seen in Table 8.11 and show the fewest clicks for Task 1. This reflects the observation that participants visited the fewest websites to complete this task. Task 2 and Task 3 have similar averages; however, Task 3 features a higher standard deviation. This is supported by Figure 8.13, which shows Task 3 to have the highest variance. According to this plot, Task 1 has a moderate left skewness. Task 2 has the highest skewness ($skewness = 3.03$) when compared to other tasks. This positive skewness is due to the single outlier on the upper bound. Due to this, it can be assumed that with more data points, this skew

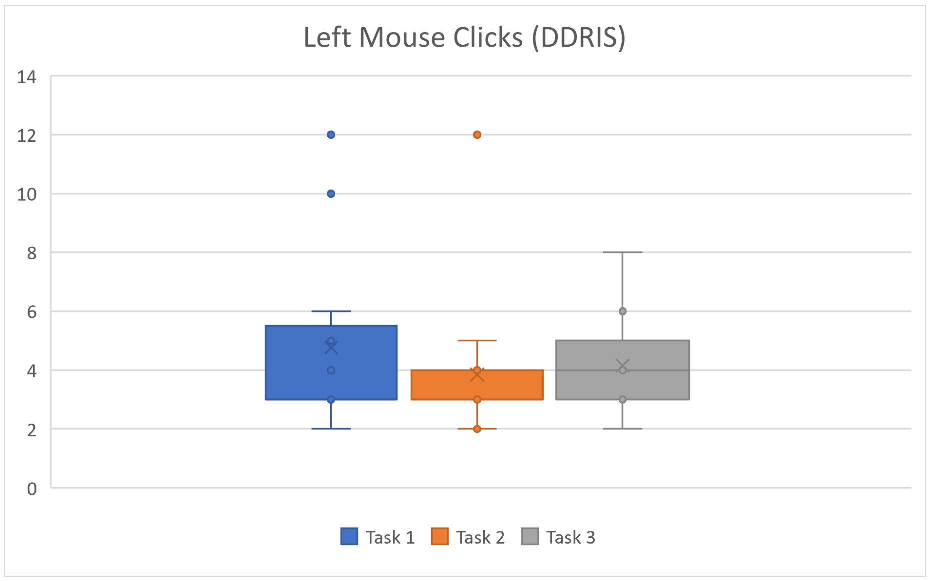


Figure 8.12: Graphical representation of left mouse clicks to complete tasks on DDRIS in Part Two of the user study.

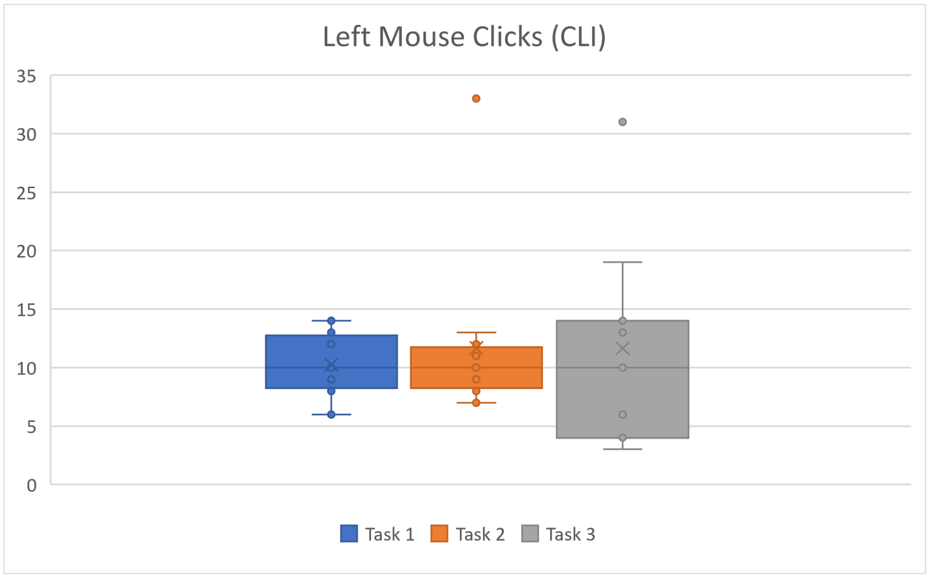


Figure 8.13: Left mouse clicks used to complete tasks through Linux command line.

would be more moderate. Task 3 also has highly skewed data towards the right due to an outlier.

When the results are compared, DDRIS outperforms command line for each task, as shown in Table 8.12. Task 1 had significantly fewer left mouse clicks by participants utilizing DDRIS when compared to command line, $t(11) = -4.37, p = .001$. This was

Table 8.11: Overview of left mouse clicks to complete each task through the Linux command line in Part Two of the user study.

Left Mouse Clicks using Command Line			
Participant Number	Task 1	Task 2	Task 3
1	14	33	31
2	8	X	10
3	12	9	6
4	X	10	4
5	12	7	3
6	9	8	14
7	6	13	10
8	14	7	13
9	10	11	X
10	9	12	14
11	13	10	19
12	10	9	4
13	6	11	X
mean	10.25	11.67	11.64
SD	2.80	6.97	8.19

Table 8.12: Overview of the difference in left mouse clicks between DDRIS dashboard usage and Command Line Interface (CLI) usage.

ΔLeft Mouse Clicks			
Participant Number	Task 1	Task 2	Task 3
1	-8	-28	-27
2	-4	X	-8
3	-10	-7	2
4	X	-6	0
5	0	-3	1
6	-6	-5	-10
7	4	-1	-6
8	-11	-4	-10
9	-5	-8	X
10	-6	-9	-11
11	-10	-7	-16
12	-7	-6	-1
13	-3	-8	X
mean	**-5.5**	**-7.67**	**-7.82**

also true for Task 2, $t(11) = -3.91, p = .002$, and Task 3, $t(10) = -3.02, p = 0.01$.

Similar to Part One, Participants tended to utilize the back button in the web browser rather than right click to correct errors. However, right clicks are also how to paste a command into MobaXTerm. As such, it was hypothesized there would be more fewer clicks using DDRIS than using the CLI. Figure 8.14 presents the results from DDRIS and shows all values other than zero clicks are outliers. This observation extends to command line, as displayed in Figure 8.15. As such, there is no observable difference in right mouse clicks between DDRIS and command line across any task.

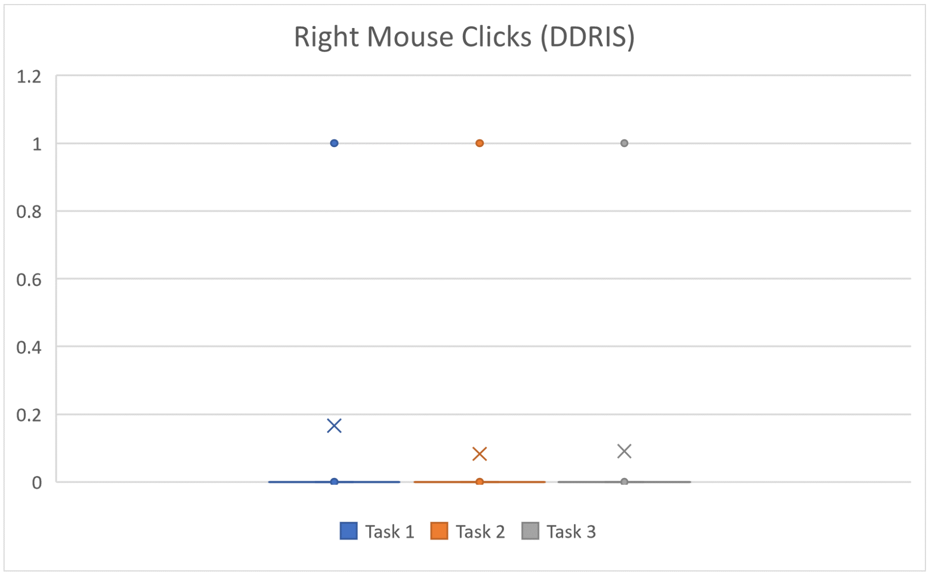


Figure 8.14: Right mouse clicks used by participants to complete tasks on DDRIS during Part Two of the user study.

During this part of the study, participants were allowed access to a single tab of Google to aid in finding information through command line. The number of websites visited have been aggregated across tasks into Table 8.13. A website refers to both new Google searches and links clicked from Google. This is to account for page previews provided by the search engine often having answers in them if thoroughly read by participants. This is to both reflect a job environment where participants would have access to the internet to find commands and provide participants a method of finding answers if they did not have them memorized prior to the study. It was expected that users would not know all commands as participants were meant to reflect an average student with enough experience to maintain systems who would be acting in

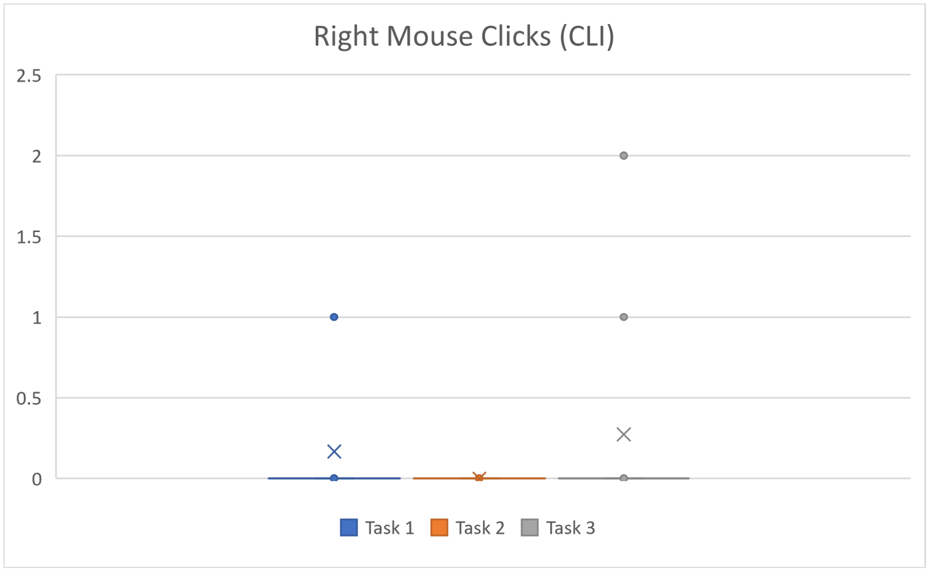


Figure 8.15: Right mouse clicks from participants to complete tasks through Linux command line.

a Junior Systems Administrator role. As reflected by the results, this was necessary as all participants utilized Google at least once to find commands. On average, participants visited 6.23 websites to complete all three tasks. This exemplifies the learning curve of utilizing command line and showcases the need for a platform to aid new administrators in quickly assessing system health while learning. After gaining experience, the dashboard still aids through alert rules by providing relevant system information in the event of a critical event, such as overheating or a full file system, through an alert channel without the administrator to first needing to diagnose a system issue.

8.3 Results from the Exit Questionnaire

Following the completion of all tasks, participants were asked to complete and exit questionnaire to collect qualitative data regarding perceived easy of use, effectiveness, and satisfaction with the dashboard. The first part of the exit survey required participants to respond on a 1-5 Likert scale where a 1 represented a strong negative response, such as that the dashboard was very difficult to navigate or tasks were very

Table 8.13: Overview of the number of websites visited by participants when participants were asked to complete tasks using command line during Part Two of the study.

Number of questions per participant	
Participant Number	Websites visited/Questions asked
1	10
2	9
3	4
4	3
5	7
6	4
7	7
8	6
9	6
10	6
11	5
12	4
13	10
mean	6.23
SD	2.31

difficult to complete, and a 5 represented a strong positive response in favor of the platform, such as that tasks were very easy to complete.

Participants were first asked how helpful they found the DDRIS dashboard in its current state. Results, represented by Figure 8.16, indicate the majority of participants found the dashboard effective with 12 responses of either a 4 or 5. One participant responded with a 3, conveying that this participant found the dashboard somewhat unhelpful. The weighted average for this question was a rating of 4.31, demonstrating users found DDRIS helpful overall.

Ease of navigation was addressed in the second question, as shown in Figure 8.17. Participants indicated that the DDRIS dashboard was also easy to navigate with 6 responses that the dashboard was easy to navigate and 5 responses that the dashboard was very easy to navigate. Two participants indicated that the dashboard was somewhat difficult to navigate. Results show a weighted average rating of 4.23 in favor of the DDRIS dashboard being easy to navigate.

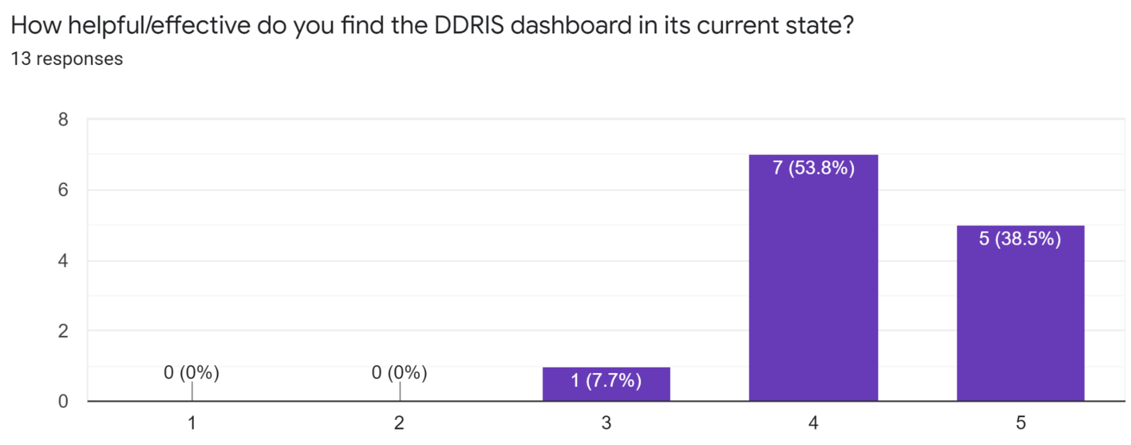


Figure 8.16: User responses when asked to rate the helpfulness of the current iteration of the DDRIS dashboard.

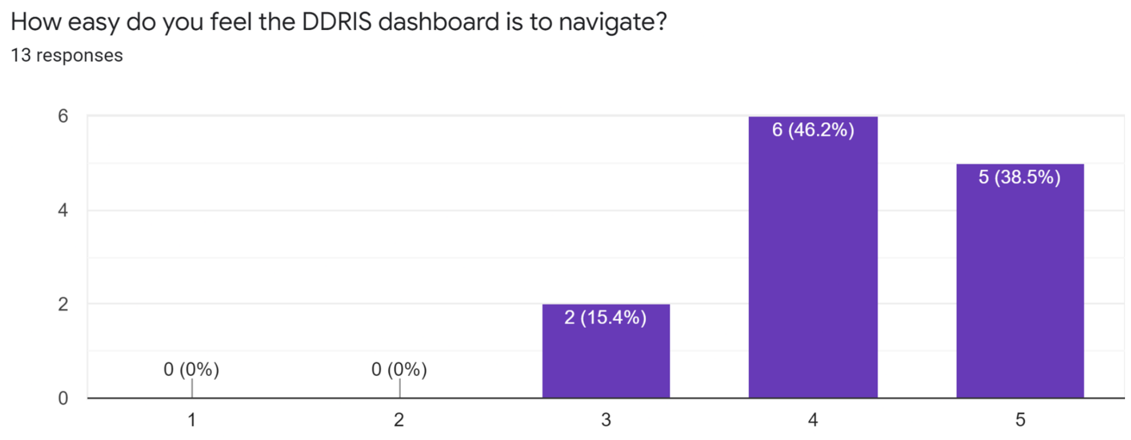


Figure 8.17: User feedback on DDRIS ease of navigation.

The design of the dashboard interface was then critiqued by participants. As illustrated by Figure 8.18, the majority of participants found the dashboard to be well designed. 12 participants responded with either a 4 or a 5, meaning well designed and very well designed respectively, and only one participant responded with a 3, indicating the dashboard was somewhat not well designed. The weighted average rating of 4.23 shows participants found the dashboard well designed overall.

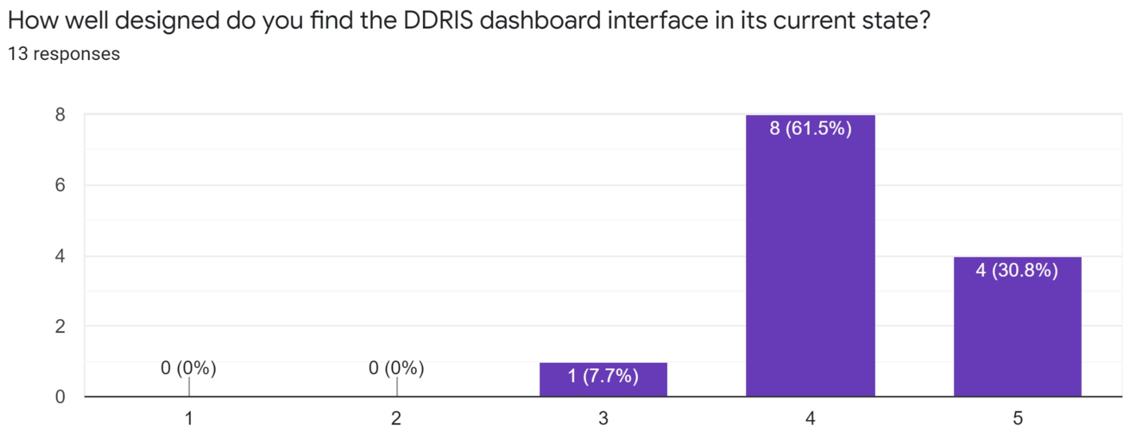


Figure 8.18: Participant satisfaction with the DDRIS dashboard.

The next three questions asked how difficult users found tasks on each of the three platforms. Result data for these three questions have been aggregated into Table 8.14 to directly compare responses. Out of the three platforms, DDRIS had the highest responses with 1 neutral and 11 positive responses of either a 4 or a 5, and a weighted average of 4.42. The Linux terminal had the seconds highest response with 1 participant finding tasks very difficult to complete and 6 participants having responded neutrally. The remaining 6 participants found the command line tasks to be easy with an overall weighted average response of 3.46. Globus had the highest number of negative responses with 6 participants indicating that tasks were difficult on the platform. 3 participants responded neutrally, and only 4 participants responded with either a 4 or a 5. The weighted average for Globus was a response of 2.92.

Table 8.14: Participant responses when asked how difficult tasks were to complete using DDRIS, Globus, and Linux command line where WAVG refers to the weighted average of responses.

Weighted Average of Task Questions						
Platform	Very Difficult	Difficult	Neutral	Easy	Very Easy	WAVG
DDRIS	0	0	1	5	6	4.42
Globus	0	6	3	3	1	2.92
Terminal	1	0	6	4	2	3.46

The remaining three questions in the exit questionnaire were short answer questions. These questions were designed to allow participants an unguided opportunity to express their feelings regarding DDRIS design. This feedback can then be utilized in future iterations to improve design effectiveness. Results from the show answer questions will be discussed in Section 8.4.

8.4 Dashboard Effectiveness Analysis

From the results, it appears most participants preferred the navigation of DDRIS to Globus. This is interesting when comparing performance results against survey responses for, while participants found tasks more difficult on Globus, as shown in Table 8.14, participants were able to complete Task 2 significantly faster on Globus, shown in Table 8.3, with less user input, exemplified in Table 8.6. In addition, Task 4 was completed without a statistically significant difference between platforms across metrics. Participants were able to complete Task 1 and Task 3 significantly faster on DDRIS than Globus; however, there was no significant difference in user input for left mouse clicks, right mouse clicks, or mouse wheel scrolls for these tasks. This implies that the subject satisfaction in favor of DDRIS is higher among participants than Globus, as varying response times and input required to complete tasks is not reflected in the exit questionnaire results.

This is further supported by participant comments in the short answer section of the exit questionnaire. One participants noted that Globus was "hard to navigate" and had "ambiguous wording" that led to a difficulty understanding how to navigate or find data. The same participant commented that DDRIS was "user friendly" with data entries "appropriately named and convenient" to access. Participants tended to become frustrated that Globus terminology was not consistent between pages. For example, some pages referred to a transfer as a task while others still used the term transfer, resulting in confusion from participants. Several participants also expressed that they preferred the dark background. While it is unlikely this affected response times, this further supports participant subjective satisfaction in favor of DDRIS.

Use of DDRIS outperformed the use of the command line across all tasks for task completion times, represented in Table 8.9, and left mouse clicks, shown in Table 8.12. When using the command line, there were also 4 instances of task failure across all tasks and participants while no instances of failure were present for DDRIS. This is further represented by difficulty scores reported by participants for both platforms. Participants perceived tasks to be more difficult using the terminal than using DDRIS 8.14. It should be noted that participants represent an incoming graduate student or junior system administrator. As a result, this time difference is expected to become less significant with increased experience. However, DDRIS still continues to offer additional functionality through alert rules. These allow DDRIS to send messages to Slack channels and emails when a critical event occurs and for proactive responses. For example, if the file system reaches 80% of its capacity, an alert will be sent out to address this issue before users are unable to write data.

When comparing Globus and DDRIS as well as command line and DDRIS, more participants would aid in assessing further metrics. Right mouse clicks comparatively do not show a statistical difference in usage between platforms. Additional participants would be beneficial towards assessing whether this difference is due to browsing methods or truly was not different between platforms.

To improve DDRIS independently of other platforms, participants were asked what changes could be made to increase the ease of use and improve the experience of a system administrator. Participants responded with functionality that is already present within the dashboard but was not used during the user study, including the ability to change colors for visualizations and alerts in the event of overheating. This conveys that this functionality should be better labeled or made more intuitive for users. Additionally, participants responded that the ability to zoom or filter was not intuitively present. To improve this, one participant suggested the use of symbols on either side of the X-axis of graphs, such as a plus-sign and minus-sign to indicate functionality. The addition of a help page could also list this functionality. Lastly, one participant suggested adding a transfer duration column to the table view of transfer metadata.

8.5 Transfer Speeds

To assess tuning of the DTN-S and DTN-F, a series of tests were run between DTNs and between campus machines. Because the NCAR-DA-9 DTN only has a 10 Gb connection, and due to its primary use case of connecting to other locations on campus that are not optimized, the performance of NCAR-DA-9 was not tuned at this time. For this reason, NCAR-DA-9 was only used as a testpoint for DTN-S and DTN-F. Instead of changes in speeds, this section provides a baseline expected performance for NCAR-DA-9. Future work for NCAR-DA-9 includes tuning and changes in hardware as mentioned in Chapter 9.

Initial Disk-to-Disk testing was performed utilizing GridFTP through Globus. Tests of a single file with file integrity checks featured four file sizes, 1.64 TB, 409.21 GB, 100.37 GB, and 53.68 GB to compare transfer rates of different file sizes. The final average speed and time elapsed until transfer completion were recorded. End-points SomeMachine and AMachine are two workstations in EJCH. Endpoint 056255hvlf8y1l is a workstation located in the University of Nevada, Reno's Scrugham Engineering and Mines (SEM). The results of these tests can be seen in Table 8.15. All tests featured similar behavior where the Average Speed suddenly declined at the end of the test. This is due to the file integrity checksum at the end of the transfer.

Further testing was completed to determine the performance change due to file integrity. While this setting should be left implemented in practice, it was important to determine the impact of the checksum on transfer performance to determine theoretical maximum performance opposed to expected behavior. Additionally, due to the negative impact of the border firewall on transfer speeds, these tests were only performed between DTN-S and -F. Tests featured files of size 53.68 GB and only included single file transfers. As shown in Table 8.16, transfer speeds theoretically double without checksums.

To test edge cases, one million files were then tested to simulate an extreme case

Table 8.15: Results from baseline single file disk-to-disk testing with file integrity.

Source	Dest	File Size	Average Speed	Duration
SomeMachine	DTN-F	1.64 TB	58.97 MB/s	7h44m30s
AMachine	DTN-S	409.21 GB	21.28 MB/s	4h09m58s
DTN-F	DTN-S	409.21 GB	172.33 MB/s	0h39m35s
DTN-S	DTN-F	409.21 GB	93.71 MB/s	1h12m47s
056255hvlf8y1l	DTN-F	100.37 GB	53.02 MB/s	0h31m25s
DTN-F	056255hvlf8y1l	100.37 GB	51.02 MB/s	0h32m39s
AMachine	DTN-S	100.37 GB	28.80 MB/s	0h57m52s
DTN-F	DTN-S	100.37 GB	360.51 MB/s	0h04m38s
DTN-S	DTN-F	100.37 GB	592.99 MB/s	0h02m48s
AMachine	DTN-F	53.68 GB	23.04 MB/s	0h38m49s

Table 8.16: Results of single file testing inside Globus without file integrity checks.

Source	Dest	File Size	Average Speed	Duration
DTN-S	DTN-F	53.68 GB	1.12 GB/s	0h00m48s
DTN-S	DTN-F	53.68 GB	1.32 GB/s	0h00m40s
DTN-F	DTN-S	53.68 GB	874.35 MB/s	0h01m02s
DTN-F	DTN-S	53.68 GB	1.20 GB/s	0h00m45s

of improper preparations for a data transfer where many files are transferred without being archived. Files ranged from 1 B to 7 B, but the overall size of the transfer was 4 MB in both cases. Minimal tests were performed solely between DTN-S and -F as the aim of running these tests are to gather a basic understanding of how unarchived data affects performance. Tests were performed without file integrity to obtain the maximum theoretical output. As shown in Table 8.17, having several small files greatly impacts performance with a drop of more than 0.557 GB in performance.

Initial memory-to-memory testing was also conducted between the DTNs. Tests were performed using `iperf` with 10 tests at a 10-second interval and 10 tests at a 60-second interval. Results from these initial memory-to-memory tests are displayed

Table 8.17: Results from edge case testing inside Globus without integrity checks.

Source	Dest	File Size	Number of files	Average Speed	Duration
DTN-S	DTN-F	1-7 B	1 mil	2.03 KB/s	0h56m31s
DTN-F	DTN-S	1-7 B	1 mil	2.02 KB/s	0h56m45s

in Figure 8.19. The average of the 10-second interval trials was a transfer size of 34.78 GB at 29.89 Gb/s with little result variance and no outliers. Transfer speeds over a 60-second interval were significantly slower with an average of 26.72 Gb/s, $t(9) = 2.52, p < .05$. The average amount transferred over 60 seconds was 186.6 GB with no outliers in speed of transfer size.

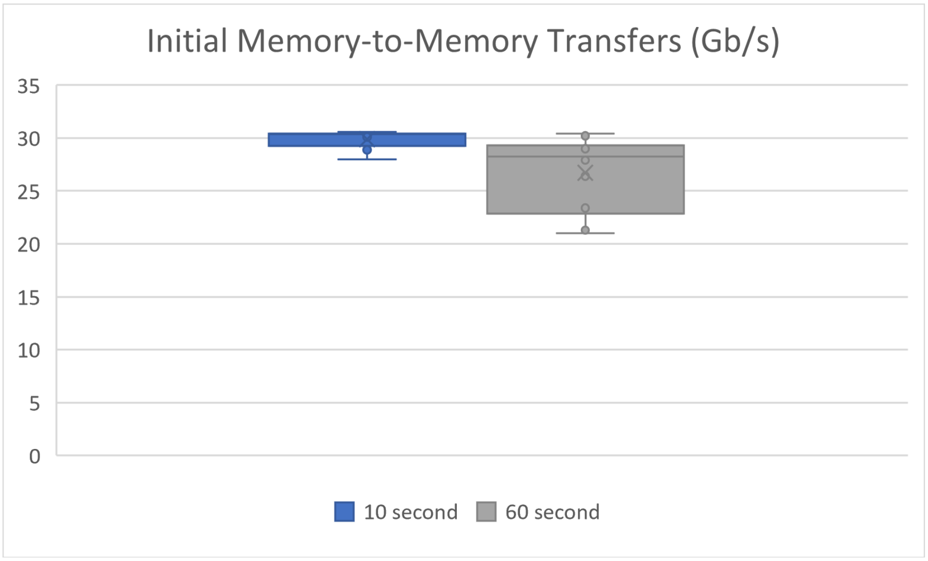


Figure 8.19: Initial results of memory-to-memory testing on the DTNs in Gb/s at intervals of 10 seconds and 60 seconds.

After DTN tuning, initial end results were collected to compare to these baseline tests. Tests were performed using 3 files of size 30GiB each. While these tests were performed both with and without file integrity, they only ocurred between DTN-s and DTN-F. The results from the final tests can be seen in Table 8.18. When compared to initial tests, for similarly sized files of 50GiB and 100GiB, these tests show a 146.4% increase with file integrity enabled. Without checksums, tests also show a 157.8% increase compared to these baseline tests.

Final memory-to-memory testing included 10 tests at a 10 second interval and 10 tests at a 60-second interval. Results show no significant difference between the 10-second and 60-second intervals, $t(9) = 0.66, p > .05$. As displayed in Figure 8.20, neither set of tests featured outliers. Test results from the 10-second interval show an average transfer rate of $35.78 Gb/s$ and transfer size of $41.67 GB$. Transfer rates for

Table 8.18: Results from final transfer results after tuning.

Source	Dest	File Size	Average Speed	Duration	Checksum Status
DTN-S	DTN-F	96.63 GB	1.05 GB/s	0h01m31s	With Integrity
DTN-F	DTN-S	96.63 GB	1.30 GB/s	0h01m14s	With Integrity
DTN-S	DTN-F	96.63 GB	2.91 GB/s	0h00m34s	Without Integrity

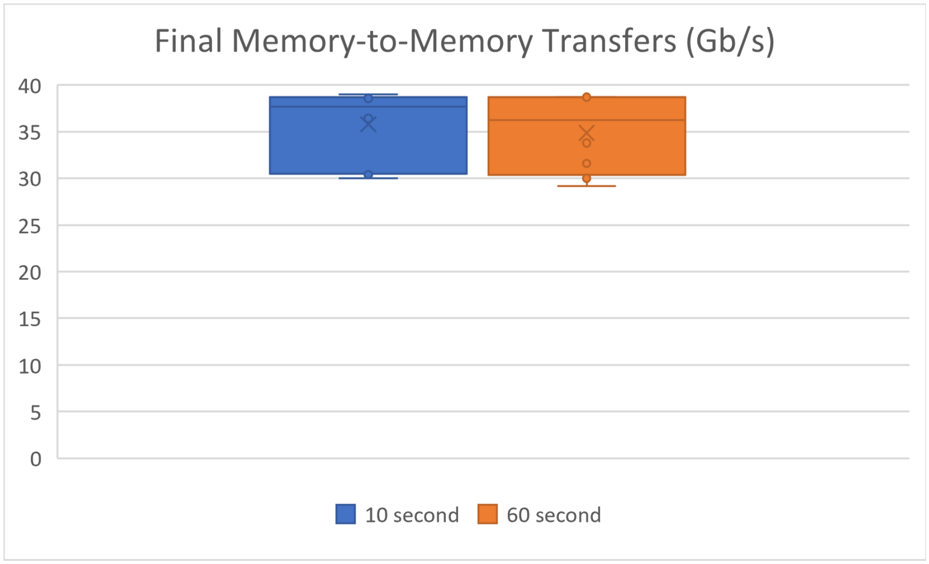


Figure 8.20: Final results of memory-to-memory testing on the DTNs in Gb/s at intervals of 10 seconds and 60 seconds.

the 10-second are significantly faster than the initial rates, $t(9) = -4.69, p = .001$. Results were also significantly faster at the 60-second interval compared to initial results such with an average transfer rate of $34.86Gb/s$, $t(9) = -3.92, p = .003$. Both intervals show transfers are capable of reaching speeds significantly closer to the theoretical limit of $40Gb/s$ despite the onboard clock speed of less than $3.0GHz$.

These results show promise for DTN tuning for DTN-S and DTN-F. However, further stress testing would be preferable to ensure proper behavior and performance prior to and following production release. These future tests are outlined in Chapter 9.

Chapter 9

Conclusion and Future Work

9.1 Conclusions

The architecture discussed in this thesis, Data Driven Research Infrastructure Systems, is a collection of subsystems designed to centralize cyberinfrastructure for researchers across fields of study. The central idea across the subsystems is to provide a single solution to data management and data transfer with further infrastructure to ensure functionality of these systems. DDRIS can be defined by two core components: the underlying hardware, and the applications developed. We proposed specifications for underlying hardware and provided detailed tuning information for this hardware to increase system performance. The engineering behind the applications involved in DDRIS was then explained. These applications of data transfer management, network telemetry collection, and metadata visualization were then described. Data transfers are handled utilizing Globus, software written to provide a simple GUI for data transfers using GridFTP. Network telemetry is collected using PerfSONAR. Finally, metadata visualization is handled with Grafana with underlying systems written primarily in Python to handle data collection. This dashboard was designed to provide a platform with consistent terminology to hold historical data for an indefinite period of time when Globus deletes old metadata as well as assess system health across data transfer nodes and the Kubernetes cluster on which systems are running.

To assess the effectiveness of the metadata dashboard, a user study was conducted to compare the solution against existing applications, including the Globus native dashboard and utilizing the Linux command line. Participants were asked to complete four tasks on both the Globus dashboard and DDRIS followed by three tasks on both the DDRIS dashboard and using command line. During each task, the amount of time elapsed, left mouse clicks used to navigate, and number of right mouse clicks were recorded. Results showed little statistical difference for task completion between Globus and DDRIS, but user responses indicated a subject preference for DDRIS. However, a significant difference between DDRIS and command line in favor of using DDRIS.

This project exposed us to critical skills in research computing and DevOps. The experiences gained in developing and maintaining this system have shown great value to researchers at UNR which we believe would be beneficial to other institutions. Any such institution seeking to collaborate with other entities would benefit from a structure such as this as it would meet the needs of multiple levels of research from small labs to large departments while still being centrally managed and maintained. It is recommended to use security zones defined through hardware as opposed to software for ease of deployment and documentation within an institution. To begin setting up an architecture such as this the general suggestion of the development team is to have a Director of Cyberinfrastructure to oversee the project. They would keep the project within scope and adapt the project according to the needs of individual researchers.

9.2 Future Work

DDRIS has many points for potential future development across elements of the system. Based on feedback from the user study, quality of life changes to address intuitiveness need to be implemented to make the dashboard more effective. This includes adding indicators to graph visualizations and labelling the color selection options. However, this functionality is not provided by Grafana. As a result, a new dashboard should be written in an easily maintainable language, such as Angular, to provide more customization from developers. This dashboard will need to include a new page dedicated to historical PerfSONAR measurements. Maddash provides historical measurements with few customization options; however, it is more beneficial to have all DDRIS measurements available from the same website.

This new dashboard page would require micro-services to handle PerfSONAR measurements. The Grafana community has developed a plug-in to allow queries against Cassandra databases of version 3.0 or newer. However, this plug-in does not provide backwards compatibility for Apache Cassandra versions prior to 3.0, which is necessary for Esmond, the data collection point for PerfSONAR. Esmond relies on the legacy Thrift API, which was replaced by Cassandra Query Language (CQL3) in Cassandra version 3.0. Either a plug-in would need to be written using Go to utilize the legacy API, or a series of micro-services will need to be written to handle results and store them in a supported database, such as Postgres. This solution features a Flask application deployed using Waitress to serve as a measurement archive location in parallel with Esmond. Waitress provides a deployment method more appropriate for production environments, including multi-threading. While Esmond continues to collect data for Maddash, this service is to be dedicated to store data quickly in a staging table within Postgres. A micro-service would then migrate results from the staging location into a more organized database schema. Another benefit of providing an archive solution other than Esmond is that data stored in Cassandra using Esmond is overwritten overtime. One of the key aspects of DDRIS is access to historical data as such, this data should be properly stored indefinitely.

A version of this micro-service should also be developed for Prometheus data, or Promscale should be added to the technology stack. Promscale is a database solution built on Timescale and Postgres to store Prometheus metrics in a more organized format rather than split into a single table of values and table of labels. By organizing the Prometheus metrics, DTN health queries can be improved to provide better performance.

Further services could be written for all databases belonging to DDRIS to address long-term deployment. Overtime, databases will be filled with timeseries and metadata records for transfers, health, and network telemetry. A rolling archive could be implemented to remove data past specified ages that can be unpacked when requested within the dashboard. This microservice would theoretically run as a cronjob twice a year.

DTN stress observation in production would be beneficial in addition to response tuning. This observation would require a large number of users requesting transfers simultaneously to determine when speeds would be throttled. As such, this test requires the production deployment of the machines to gather real user data. Observation data can then be utilized to determine further tuning parameters and increase performance during peak hours.

References

[1] A comparison of software raid types. URL: https://alephnull.com/benchmarks/sata2009/raidtype.html (visited on 10/04/2021).

[2] Mohd Bazli Ab Karim, Jing-Yuan Luke, Ming-Tat Wong, Pek-Yin Sian, and Ong Hong. Ext4, XFS, BtrFS and ZFS Linux file systems on RADOS Block Devices (RBD): I/O performance, flexibility and ease of use comparisons. In *2016 IEEE Conference on Open Systems (ICOS)*, pages 18–23, 2016. DOI: 10.1109/ICOS.2016.7881982.

[3] Rancher Admin. Introduction to kubernetes architecture — suse communities, 2019. URL: https://rancher.com/learning-paths/introduction-to-kubernetes-architecture/ (visited on 10/12/2021).

[4] A. Allcock, J. Bester, J. Bresnahan, A. Chervenak, L. Liming, and S. Tuecke. GridFTP: Protocol Extensions to FTP for the Grid. In 2001. URL: https://www.ogf.org/documents/Drafts/old/GridFTP%20Protocol%20RFC%20Draft2.pdf.

[5] William Allcock, John Bresnahan, Rajkumar Kettimuthu, Michael Link, Catalin Dumitrescu, Ioan Raicu, and Ian Foster. The Globus Striped GridFTP Framework and Server. In *Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*, SC '05, page 54, USA. IEEE Computer Society, 2005. ISBN: 1595930612. DOI: 10.1109/SC.2005.72. URL: https://doi.org/10.1109/SC.2005.72.

[6] Bryce Allen, John Bresnahan, Lisa Childers, Ian Foster, Gopi Kandaswamy, Raj Kettimuthu, Jack Kordas, Mike Link, Stuart Martin, Karl Pickett, and Steven Tuecke. Globus online: radical simplification of data movement via saas, 2011. URL: https://www.globus.org/sites/default/files/globus-online-radical-simplification-of-data-movement-via-saas.pdf.

[7] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, April 2010. ISSN: 0001-0782. DOI: 10.1145/1721654.1721672. URL: https://doi.org/10.1145/1721654.1721672.

[8] Melvin C. August, Gerald M. Brost, Christopher C. Hsiung, and Alan J. Schiffleger. Cray X-MP: the birth of a supercomputer. *Computer*, 22(1):45–52, 1989. DOI: 10.1109/2.19822.

[9] Gordon Bell and Jim Gray. What's next in high-performance computing? *Commun. ACM*, 45(2):91–95, February 2002. ISSN: 0001-0782. DOI: 10.1145/503124.503129. URL: https://doi.org/10.1145/503124.503129.

[10] Oren Ben-Kiki, Clark Evans, and Brian Ingerson. YAML ain't markup language (yaml™) version 1.1, 2005. URL: https://yaml.org/spec/1.1/.

[11] Blacksun Software. Mousotron: Mouse and keyboard usage counter. URL: http://www.blacksunsoftware.com/mousotron.html (visited on 09/29/2021).

[12] Tony Bourke. *Server load balancing.* O'Reilly Media, Inc., 2001. ISBN: 978-0596000509.

[13] Brave Software, Inc. Secure, Fast & Private Web Browser with Adblocker — Brave Browser. URL: https://brave.com/ (visited on 09/23/2021).

[14] Brendan Burns and Craig Tracey. *Managing Kubernetes: operating Kubernetes clusters in the real world.* OReilly, 2019. ISBN: 978-1492033912. URL: https://www.oreilly.com/library/view/managing-kubernetes/9781492033905/.

[15] Mingming Cao, Suparna Bhattacharya, and Ted Ts'o. Ext4: the next generation of ext2/3 filesystem. In *LSF*, 2007. URL: https://www.usenix.org/legacy/event/lsf07/tech/cao_m.pdf.

[16] Ceph authors and contributors. Ceph file system – ceph documentation. URL: https://docs.ceph.com/en/pacific/cephfs/ (visited on 10/06/2021).

[17] Cloud Native Computing Foundation. CNCF cloud native definition v1.0. URL: https://github.com/cncf/toc/blob/main/DEFINITION.md (visited on 11/09/2021).

[18] Computer History Museum. The Cray-1 supercomputer. URL: https://www.computerhistory.org/revolution/supercomputers/10/7 (visited on 10/08/2021).

[19] Eli Dart, Lauren Rotman, Brian Tierney, Mary Hester, and Jason Zurawski. The science dmz: a network design pattern for data-intensive science. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, Denver, Colorado*, SC '13, New York, NY, USA. Association for Computing Machinery, 2013. ISBN: 9781450323789. DOI: 10.1145/2503210.2503245. URL: https://doi.org/10.1145/2503210.2503245.

[20] Docker, Inc. Docker overview — docker documentation, September 2021. URL: https://docs.docker.com/get-started/overview/ (visited on 09/14/2021).

[21] Rajdeep Dua, A Reddy Raja, and Dharmesh Kakadia. Virtualization vs containerization to support paas. In *Proceedings of the 2014 IEEE International Conference on Cloud Engineering*, IC2E '14, 610–614, USA. IEEE Computer Society, 2014. ISBN: 9781479937660. DOI: 10.1109/IC2E.2014.41. URL: https://doi.org/10.1109/IC2E.2014.41.

[22] ESnet: Energy Sciences Network. Best practices for science DMZ security. URL: https://fasterdata.es.net/science-dmz/science-dmz-security/best-practices-for-science-dmz-security/ (visited on 11/12/2021).

[23] ESnet: Energy Sciences Network. Esmond: esnet monitoring daemon. URL: `https://github.com/perfsonar/esmond#:~:text=ESnet%20Monitoring%20Daemon-,At%20this%20time%20esmond%20is%20only%20supported%20as%20part%20of,to%20a%20much%20wider%20audience.` (visited on 04/13/2022).

[24] ESnet: Energy Sciences Network. Interrupt Binding. URL: `https://fasterdata.es.net/host-tuning/linux/100g-tuning/interrupt-binding/` (visited on 11/30/2021).

[25] ESnet: Energy Sciences Network. Network performance and troubleshooting (perfsonar). URL: `https://www.es.net/network-r-and-d/perfsonar/` (visited on 09/29/2021).

[26] ESnet: Energy Sciences Network. Science DMZ architecture. URL: `https://fasterdata.es.net/science-dmz/science-dmz-architecture/` (visited on 11/12/2021).

[27] ESnet: Energy Sciences Network. Science DMZ: Data Transfer Nodes. URL: `https://fasterdata.es.net/science-dmz/DTN/` (visited on 11/30/2021).

[28] Ian Foster. Globus online: accelerating and democratizing science through cloud-based services. *IEEE Internet Computing*, 15(3):70–73, 2011. DOI: `10.1109/MIC.2011.64`.

[29] Ian T Foster, Josh Boverhof, Ann Chervenak, Lisa Childers, Annette DeSchoen, Gabriele Garzoglio, Dan Gunter, Burt Holzman, Gopi Kandaswamy, Raj Kettimuthu, Jack Kordas, Miron Livny, Stuart Martin, Parag Mhashilkar, Taghrid Samak Miller Zachary Samak, Mei-Hui Su, Steven Tuecke, Vanamala Venkataswamy, Craig Ward, and Cathrin Weiss. Reliable high-performance data transfer via globus online. URL: `https://www.mcs.anl.gov/papers/P1904.pdf`.

[30] Robert P. Goldberg. Survey of virtual machine research. *Computer*, 7(6):34–45, 1974. DOI: `10.1109/MC.1974.6323581`.

[31] Google Inc. Personal cloud storage & file sharing platform - google. URL: `https://www.google.com/drive/` (visited on 01/31/2022).

[32] Google Inc. Remove ftp support - chrome platform status. URL: `https://chromestatus.com/feature/6246151319715840` (visited on 10/21/2021).

[33] Grafana Labs. Grafana® features — grafana labs. URL: `https://grafana.com/grafana/` (visited on 09/27/2021).

[34] Frederick C. Harris. Networking and computing infrastructure in nevada: current status and future development. Nevada NSF EPSCoR Climate Change Conference, 2010. URL: `https://digitalscholarship.unlv.edu/epscor/2010/feb02/33/`.

[35] Dominique A Heger. Workload dependent performance evaluation of the btrfs and zfs filesystems. In *Int. CMG Conference*, 2009. URL: `https://picture.iczhiku.com/resource/paper/shIdkpUhfRUjHbcx.pdf`.

[36] Tony Hey and Anne E. Trefethen. Cyberinfrastructure for e-science. *Science*, 308(5723):817–821, 2005. DOI: 10.1126/science.1110410. eprint: https://www.science.org/doi/pdf/10.1126/science.1110410. URL: https://www.science.org/doi/abs/10.1126/science.1110410.

[37] Lorin Hochstein and Rene Moser. *Ansible: Up and Running: Automating configuration management and deployment the easy way*. O'Reilly Media, Inc., 2017. ISBN: 978-1491979808. URL: https://www.oreilly.com/library/view/ansible-up-and/9781491979792/.

[38] William L. Hosch. *Supercomputer*. In *Encyclopedia Britanica*. URL: https://www.britannica.com/technology/supercomputer (visited on 10/08/2021).

[39] Intel Corporation. An introduction to the intel® quickpath interconnect. URL: https://www.intel.com/content/www/us/en/io/quickpath-technology/quick-path-interconnect-introduction-paper.html (visited on 11/30/2021).

[40] Kepios. Digital around the world — datareportal – global digital insights. URL: https://datareportal.com/global-digital-overview#:~:text=Internet\%20use\%20around\%20the\%20world,500\%2C000\%20new\%20users\%20each\%20day. (visited on 01/31/2022).

[41] Brian W. Kernighan and Rob Pike. *The UNIX programming environment*. Prentice-Hall Software Series. Prentice-Hall, 1st edition, 1984. ISBN: 978-0139376818.

[42] Youngseek Kim and Kevin Crowston. Technology adoption and use theory review for studying scientists' continued use of cyber-infrastructure. *Proceedings of the American Society for Information Science and Technology*, 48(1):1–10, 2011. DOI: https://doi.org/10.1002/meet.2011.14504801197. eprint: https://asistdl.onlinelibrary.wiley.com/doi/pdf/10.1002/meet.2011.14504801197. URL: https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/meet.2011.14504801197.

[43] Sinny Kumari. *Linux shell scripting essentials: Learn shell scripting to solve complex shell-related problems and efficiently automate your day-to-day tasks*. Packt Publishing, 2015. ISBN: 978-1785284441.

[44] Yong-Hong Kuo and Andrew Kusiak. From data to big data in production research: the past and future trends. *International Journal of Production Research*, 57(15-16):4828–4853, 2019. DOI: 10.1080/00207543.2018.1443230. eprint: https://doi.org/10.1080/00207543.2018.1443230. URL: https://doi.org/10.1080/00207543.2018.1443230.

[45] Jay LaCroix. *Linux Mint essentials: a practical guide to Linux Mint for the novice to the professional*. Packt Publishing, 2014. ISBN: 978-1782168157.

[46] Zhengchun Liu, Prasanna Balaprakash, Rajkumar Kettimuthu, and Ian Foster. Explaining wide area data transfer performance. In *Proceedings of the 26th International Symposium on High-Performance Parallel and Distributed Computing, Washington, DC, USA*, HPDC '17, 167–178, New York, NY, USA. Association for Computing Machinery, 2017. ISBN: 9781450346993. DOI: `10.1145/3078597.3078605`. URL: `https://doi.org/10.1145/3078597.3078605`.

[47] I. Scott MacKenzie. *Human-Computer Interaction: An Empirical Research Perspective*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2013. ISBN: 978-0124058655.

[48] Sean Marston, Zhi Li, Subhajyoti Bandyopadhyay, Juheng Zhang, and Anand Ghalsasi. Cloud computing — the business perspective. *Decision Support Systems*, 51(1):176–189, 2011. ISSN: 0167-9236. DOI: `https://doi.org/10.1016/j.dss.2010.12.006`. URL: `https://www.sciencedirect.com/science/article/pii/S0167923610002393`.

[49] Peter Mell and Tim Grance. The NIST definition of cloud computing: Recommendations of the National Institute of Standards and Technology. Technical report NIST Special Publication 800-145, Gaithersburg, MD 20899-8930, 2011. URL: `https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf`.

[50] Dirk Merkel. Docker: lightweight linux containers for consistent development and deployment. *Linux Journal*, 2014(239), May 2014. ISSN: 1075-3583. URL: `https://www.linuxjournal.com/content/docker-lightweight-linux-containers-consistent-development-and-deployment`.

[51] Mobatek. MobaXterm free Xserver and tabbed SSH client for Windows. URL: `https://mobaxterm.mobatek.net/` (visited on 09/23/2021).

[52] Henry Neeman, Aaron Bergstrom, Dana Brunson, Carrie Ganote, Zane Gray, Brian Guilfoos, Robert Kalescky, Evan Lemley, Brian G. Moore, Sai Kumar Ramadugu, Alana Romanella, Johnathan Rush, Andrew H. Sherman, Brian Stengel, and Dan Voss. The advanced cyberinfrastructure research and education facilitators virtual residency: toward a national cyberinfrastructure workforce, miami, usa. In *Proceedings of the XSEDE16 Conference on Diversity, Big Data, and Science at Scale*, XSEDE16, New York, NY, USA. Association for Computing Machinery, 2016. ISBN: 9781450347556. DOI: `10.1145/2949550.2949584`. URL: `https://doi.org/10.1145/2949550.2949584`.

[53] Caitlin Neiman. Built-in FTP implementation to be removed in Firefox 90 — mozilla add-ons community blog, 2021. URL: `https://blog.mozilla.org/addons/2021/04/15/built-in-ftp-implementation-to-be-removed-in-firefox-90/` (visited on 10/21/2021).

[54] Nevada System of Higher Education, System Computing Services. Nevadanet backbone. personal communication, September 2021.

[55] Jakob Nielsen. *Usability Engineering*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1994. ISBN: 978-0080520292.

[56] PagerDuty, Inc. Pagerduty — real-time operations — incident response — on-call — pagerduty. URL: `https://www.pagerduty.com/` (visited on 01/12/2022).

[57] Claus Pahl. Containerization and the paas cloud. *IEEE Cloud Computing*, 2(3):24–31, 2015. DOI: `10.1109/MCC.2015.51`.

[58] Pallets. Welcome to Flask – Flask documentation (2.0.x). URL: `https://flask.palletsprojects.com/en/2.0.x/` (visited on 09/27/2021).

[59] Jon Postel and Joyce Reynolds. File transfer protocol, 1985.

[60] Prometheus Authors. Overview: prometheus. URL: `https://prometheus.io/docs/introduction/overview/` (visited on 09/13/2021).

[61] Prometheus Authors. Prometheus - monitoring system & time series database. URL: `https://prometheus.io/` (visited on 09/27/2021).

[62] Puppet. Powerful infrastructure automation and delivery — puppet. URL: `https://puppet.com/` (visited on 05/02/2022).

[63] Python Software Foundation. Welcome to Python.org. URL: `https://www.python.org/` (visited on 09/14/2021).

[64] Zhi Qiao, Jacob Hochstetler, Shuwen Liang, Song Fu, Hsing-bung Chen, and Bradley Settlemyer. Incorporate proactive data protection in zfs towards reliable storage systems. In *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)*, pages 904–911, 2018. DOI: `10.1109/DASC/PiCom/DataCom/CyberSciTec.2018.00-10`.

[65] Red Hat, Inc. Ansible is Simple IT Automation. URL: `https://www.ansible.com/` (visited on 09/11/2021).

[66] David Ribes and Charlotte P Lee. Sociotechnical studies of cyberinfrastructure and e-research: current themes and future trajectories. *Computer Supported Cooperative Work (CSCW)*, 19(3):231–244, 2010.

[67] D. Ridge, D. Becker, P. Merkey, and T. Sterling. Beowulf: harnessing the power of parallelism in a pile-of-pcs. In *1997 IEEE Aerospace Conference*, volume 2, 79–91 vol.2, 1997. DOI: `10.1109/AERO.1997.577619`.

[68] O. Rodeh and A. Teperman. Zfs - a scalable distributed file system using object disks. In *20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies, 2003. (MSST 2003). Proceedings.* Pages 207–218, 2003. DOI: `10.1109/MASS.2003.1194858`.

[69] Rsync(1) - linux man page. URL: `https://linux.die.net/man/1/rsync` (visited on 01/31/2022).

[70] Richard M. Russell. The CRAY-1 computer system. *Commun. ACM*, 21(1):63–72, January 1978. ISSN: 0001-0782. DOI: `10.1145/359327.359336`. URL: `https://doi.org/10.1145/359327.359336`.

[71] R.R. Schaller. Moore's law: past, present and future. *IEEE Spectrum*, 34(6):52–59, 1997. DOI: `10.1109/6.591665`.

[72] Mathijs Jeroen Scheepers. Virtualization and containerization of application infrastructure: a comparison. In *21st Twente student conference on IT*, volume 21, 2014. URL: `http://faculty.washington.edu/wlloyd/courses/tcss562/papers/Spring2017/team3_containers/Virtualization%20and%20Containerization%20of%20Application%20Infrastructure-%20A%20Comparison.pdf`.

[73] Kesha Shah. How much data is created every day in 2020?, 2020. URL: `https://www.linkedin.com/pulse/how-much-data-created-every-day-2020-kesha-shah/` (visited on 01/31/2022).

[74] Similarweb LTD. Most visited websites - top websites ranking for january 2022 — similarweb. URL: `https://www.similarweb.com/top-websites/` (visited on 01/31/2022).

[75] Slack Technologies, LLC. Slack is where the future works. URL: `https://slack.com/` (visited on 01/12/2022).

[76] Ian Sommerville. *Software engineering*. Pearson, 10th edition, 2016. ISBN: 978-0133943030.

[77] Thomas Sterling, Donald J. Becker, Daniel Savarese, John E. Dorband, Udaya A. Ranawake, and Charles V. Packer. Beowulf: a parallel workstation for scientific computation. In *In Proceedings of the 24th International Conference on Parallel Processing*, pages 11–14. CRC Press, 1995. URL: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.53.8824&rep=rep1&type=pdf`.

[78] Craig A. Stewart, Stephen Simms, Beth Plale, Matthew Link, David Y. Hancock, and Geoffrey C. Fox. What is cyberinfrastructure, norfolk, virginia, usa. In *Proceedings of the 38th Annual ACM SIGUCCS Fall Conference: Navigation and Discovery*, SIGUCCS '10, 37–44, New York, NY, USA. Association for Computing Machinery, 2010. ISBN: 9781450300032. DOI: `10.1145/1878335.1878347`. URL: `https://doi.org/10.1145/1878335.1878347`.

[79] The Apache Software Foundation. Apache cassandra. URL: `https://cassandra.apache.org/_/index.html` (visited on 09/29/2021).

[80] The Kubernetes Authors. Kubernetes: production-grade container orchestration. URL: `https://kubernetes.io/` (visited on 09/13/2021).

[81] The perfSONAR Project and contributors. perfSONAR. URL: `https://www.perfsonar.net/index.html` (visited on 09/29/2021).

[82] The PostgreSQL Global Development Group. PostgreSQL: The World's most Advanced Open Source Relational Database. URL: `https://www.postgresql.org/` (visited on 09/24/2021).

[83] Timescale, Inc. Time-series data simplified — Timescale. URL: `https://www.timescale.com/` (visited on 09/23/2021).

[84] University of Chicago, Argonne National Laboratory. Data transfer with globus — globus. URL: `https://www.globus.org/data-transfer` (visited on 10/06/2021).

[85] University of Chicago, Argonne National Laboratory. Globus Connect Server v5 Installation Guide. URL: `https://docs.globus.org/globus-connect-server/v5/` (visited on 09/29/2021).

[86] University of Chicago, Argonne National Laboratory. Globus: what we do. URL: `https://www.globus.org/what-we-do` (visited on 10/06/2021).

[87] University of Nevada, Reno. Research Integrity - Training. URL: `https://www.unr.edu/research-integrity/training` (visited on 11/23/2021).

[88] Steven Vaughan-Nichols. A super-fast history of supercomputers: From the CDC 6600 to the Sunway TaihuLight, November 2017. URL: `https://www.hpe.com/us/en/insights/articles/a-super-fast-history-of-supercomputers-from-the-cdc-6600-to-the-sunway-taihulight-1711.html` (visited on 10/08/2021).

[89] Vinod Kumar Vavilapalli, Arun C. Murthy, Chris Douglas, Sharad Agarwal, Mahadev Konar, Robert Evans, Thomas Graves, Jason Lowe, Hitesh Shah, Siddharth Seth, Bikas Saha, Carlo Curino, Owen O'Malley, Sanjay Radia, Benjamin Reed, and Eric Baldeschwieler. Apache Hadoop YARN: Yet Another Resource Negotiator. In *Proceedings of the 4th Annual Symposium on Cloud Computing*, SOCC '13, Santa Clara, California. Association for Computing Machinery, 2013. ISBN: 9781450324281. DOI: `10.1145/2523616.2523633`. URL: `https://doi.org/10.1145/2523616.2523633`.

[90] Sage A. Weil, Scott A. Brandt, Ethan L. Miller, Darrell D. E. Long, and Carlos Maltzahn. Ceph: a scalable, high-performance distributed file system. In *Proceedings of the 7th Symposium on Operating Systems Design and Implementation, Seattle, Washington*, OSDI '06, 307–320, USA. USENIX Association, 2006. ISBN: 1931971471.

[91] John Paul Wohlscheid. What is ZFS? Why people use ZFS? [explained for beginners], 2021. URL: `https://itsfoss.com/what-is-zfs/` (visited on 10/04/2021).

[92] Judy L Woods, Jeff S West, and Peter R Sulyma. Construction and Utilization of a Beowulf Computing Cluster: A User's Perspective. Technical report TM-2000-210691, NASA, 2000.

Appendix A

UNR IRB Application Cover Sheet

University of Nevada, Reno
Institutional Review Board
Part I, Cover Sheet

Last edited by: Brianna Blain-Castelli

Last edited on: September 21, 2021

[click for checklist]

[1815780-1] Near Real-Time and Historical Cyberinfrastructure Data Dashboard

Answer all questions on this form completely, include attachments and obtain signatures of Principal Investigator and the Responsible official prior to final submission on IRBNet.

I. Principal Investigator

Name:	Sergiu Dascalu, PhD		
Institution:			
Department:	College of Engineering		
Telephone:	(775) 636-5060	**Email:**	dascalus@cse.unr.edu
Address:	1664 N Virginia St, Reno, NV 89557		

COI Disclosure: Any financial interests related to this study?

☐ Yes

☑ No

If yes, **COI Disclosure Explanation:**

Could any external entity benefit financially from the results of this study?

☐ Yes

☑ No

II. Co-Investigator(s) or Research Team Member(s)

N/A ☑

Name:			
Department:			
Institution:			
Telephone:		**Email:**	

COI Disclosure: Any financial interests related to this study?

☐ Yes

☐ No

If yes, **COI Disclosure Explanation:**

III. Student Investigator(s) N/A ☐

Name:	Brianna Blain-Castelli		
Institution:			
Telephone:	9167599502	**Email:**	bblaincastelli@unr.edu

COI Disclosure: Any financial interests related to this study?

☐ Yes

☑ No

If yes, **COI Disclosure Explanation:**

IV. Project Information

Research Type:

☐ Biomedical

☑ Social Behavioral/Educational

Photographing or Video Recording:

☐ Yes
Upload the Photo Release form.

☑ No

Identifiable Information from Education Records:
Note that accessing education records for research purposes invokes FERPA regulations. In the protocol, address how records are accessed, whether researchers will access directory information only, and how written permission will be collected from students or parents for minor students. For more information on FERPA, see IRB Policy 76.

☐ Yes

☑ No

Research Location:

☐ VA Sierra Nevada Healthcare System

☐ Saint Mary's Regional Medical Center

☐ Renown Health

☑ UNR Campus

☐ Other -

International Research:

☐ Yes
For international research, reference IRB Policy 575 for details to address in the protocol.

☑ No

If yes, **specify the countries:**

Renown Health Research Locations:

☐ Renown Regional ☐ Renown Pharmacy

- ☐ Renown South Meadows
- ☐ Renown Pregnancy Center
- ☐ Renown Outpatient Clinic
- ☐ Renown Urgent Care
- ☐ Renown Imaging
- ☐ Renown Lab
- ☐ Renown Emergency Room
- ☐ Renown Skilled Nursing
- ☐ Renown Hospice Care
- ☐ Renown Home Health
- ☐ Renown Rehabilitation

Requested Review Path:

- ☐ Expedited IRB Review
  Complete Protocol - Social Behavioral Educational Research and Records Research or Protocol - Biomedical Research
- ☐ Full Board Review
  Complete Protocol - Social Behavioral Educational Research and Records Research or Protocol - Biomedical Research
- ☑ Exempt Review
  Complete Protocol - Social Behavioral Educational Research and Records Research
- ☐ Requesting a determination about whether a project is human research
  Complete Request for Human Research Determination
- ☐ Requesting authorization to use an external IRB
  Complete Request to Use an External IRB
- ☐ Reporting emergency use of an FDA-regulated drug or device
  Complete Emergency Use Investigational Drug or Device
- ☐ Review of a Humanitarian Use Device
  Complete Protocol - Humanitarian Use Device for Treatment or Diagnosis
- ☐ Research involving existing records or specimens
  Complete Protocol - Social Behavioral Educational Research and Records Research

Risk Level:

- ☑ Minimal risk
- ☐ Greater than minimal risk (requires full board review)
- ☐ No known risk

Involvement of Vulnerable Populations:

- ☑ N/A, research will not involve vulnerable populations
- ☐ Pregnant women and fetuses
  For research with pregnant women and fetuses, reference IRB Policy 210 and 211 for details to address in the protocol.
- ☐ Prisoners
  Complete Research with Prisoners
- ☐ Children (persons under 18 years of age)
  For research with children, reference IRB Policy 230 for details to address in the protocol.
- ☐ Adults with impaired decision-making capacity
  For research with adults with impaired decision-making capacity, reference IRB Policy 240 for details to address in the protocol.
- ☐ People who do not speak English

For research with people who do not speak English, reference IRB Policy 250 for details to address in the protocol.

V. Funding Information

N/A ☑

Sponsor Type:

☐ Federal Government	☐ Other Government (State/Local)
☐ Industry Sponsor	☐ Other Private Funds
☐ Departmental	☐ Subcontract
☐ Other:	

Sponsor Name:

Grant/Contract Title and Number:

VI. Federal Agencies with Additional Requirements to Protect Human Participants

Please see the "Instructions to Researchers" section at the end of this form for a list of required supplemental forms and relevant policies.

- ☐ DoD
- ☐ DoE
- ☐ DoEd
- ☐ DoJ or NIJ
- ☐ EPA
- ☐ NSF
- ☐ VA
- ☑ N/A

VII. FDA-Regulated Research

- ☑ N/A, research does not involve drugs or devices
- ☐ Drug research

Trade Name	Generic Name

- ☐ Device research

Name of Device	Device Manufacturer

VIII. External Committee Approvals

- ☑ Thesis or Dissertation Committee
- ☐ Radiation Safety Committee

- ☐ Biosafety Committee
- ☐ Other:
- ☐ N/A

INSTRUCTIONS TO RESEARCHERS

[top]

You have completed Part I of the application process. **Preview** Part I and correct if needed. Print the last page so you have the list of the researcher forms and additional regulatory requirements expected for this research. Click **Save and Exit**. **Add** the remaining required documents (listed below or referenced in the researcher forms/applications), **address** the necessary regulatory and policy requirements in the protocol and other project documents, and then the PI should electronically **Sign** and **Submit** the project. Make sure to upload training documentation for all researchers listed on this form.
If you have any questions, refer to the IRBNet pages of the Research Integrity website.

Additional required researcher forms and policies/regulations:

- Complete Protocol - Social Behavioral Educational Research and Records Research

Appendix B

UNR IRB Social Behavioral Educational Research Protocol

University of Nevada, Reno
Research & Innovation

Research Integrity
Tel: (775) 327-2368
Web: unr.edu/research-integrity

Protocol – Social Behavioral Educational Research and Record Research

Project ID: [1815780-1]
Title: Near Real-Time and Historical Cyberinfrastructure Data Visualization
Principal Investigator: Sergiu Dascalu, PhD
Co-Investigators / Study Contact: Brianna Blain-Castelli

Please delete the instructions and sample text after you complete each section. Do not delete the section headings; if the heading does not relate to your research insert N/A.

Background:
Campus centralized tools for data management are essential to collaboration for data driven research. Having standardized workflows and available resources allows researchers to easily share data across departments and aids in sharing across universities. However, moving toward this model requires support for the new infrastructure.

The project that this study is based on involves setting up and maintaining hardware- and software-based cyberinfrastructure to support data driven research. To aid in overseeing the health and status of these systems, a dashboard has been created to visualize system data and manage alerts. Data visualization is an important tool as data expands in size to assist in quick data analysis and proactive management.

The main goal of this study is to determine the effectiveness, ease of use, and subjective satisfaction of the design of the dashboard that has been created. This newly developed dashboard, referred to as the Data Driven Research Infrastructure Systems (DDRIS) Dashboard through this study, will be compared against existing solutions for gathering pertinent system data to aid in evaluating the success of the DDRIS Dashboard's design.

Study Aims/Objectives:
This study aims to compare a dashboard developed by the research team to visualize system health, transfer speeds, and network telemetry against current solutions within the context of Human Computer Interaction. The current dashboard will be evaluated in this study to determine its effectiveness, ease of use, and subjective satisfaction through user-verified review.

Study Population:
Participants will be between the ages of 20 and 50 of mixed gender and ethnic background. Participants include current or former University of Nevada, Reno students with a background in computers.

Vulnerable Populations:
N/A

Sample Size:
There will be between 16-24 participants involved in this study with the possibility of more based-on recruitment. This sample size was chosen due to standards in the field of Computer Science paired with expectations of how many responses there will be to the recruitment email.

Recruitment Process:
Potential participants will be chosen from the following: Computer Science labs, former lab members, and people within the university known and recommended to participate by other participants. Recruitment will be conducted through email. The template of the recruitment email has been attached to this application.

Screening Procedures:
Participants will be recruited through Computer Science labs, former lab members, and people recommended to the study by other participants. Participants recruited through the first two means will not be screened as they will meet the minimum requirements to be involved in the study. Those recommended by participants will be screened by asking them if they have a background in computers prior to scheduling a time to complete the study.

Informed Consent Process:
Participants will be fully informed of the nature of the research and what data will be collected. They will be reassured that no personal data that could potentially be linked back to them will be collected through the study. Participants will be given the consent form attached prior to beginning the study. The participant is to review, sign, and date the form.

Data Collection Procedures:
Participants will be recruited through email and schedule a date and time they are available to participate in the study. Participants will meet the instructor in EJCH or be provided a Zoom meeting to complete the study based on participant preference. Zoom meetings will not be recorded. The participant will be asked to review and sign the consent form attached to this study. First, participants will complete a pre-usage/entrance questionnaire to assess existing knowledge and participant demographics through Google Forms. No personal information will be recorded, including name. Second, the participant will complete the Part 1 tasks provided in the application. Through parts one and two, Mousotron will be utilized to capture data such as mouse clicks, mouse scrolls, keys typed, and amount of time taken to complete a task. Third, participants will complete the Part 2 tasks provided in the application. Finally, participants will complete a post-usage/exit questionnaire to provide qualitative data on the thoughts participants had on the dashboard.

Study Duration/ Study Timeline:
The study will be conducted in a single 30-45 minute session.

Study Locations:
Researchers will conduct the study in EJCH room 272C as it already serves as a dedicated lab space for the researchers. The option to complete the study remotely through Zoom will be provided.

International Research:
N/A

Participant Compensation:
N/A

Risk to Participants:
There are no risks associated with this study. N/A

Benefits to Participants:
This research does not present any direct benefit to the participants. However, the research provides an opportunity to gain a better understanding of web browsing and Linux command line.

Privacy of Participants:
Recruitment will begin with lab members known personally by the research team. Participants recommended by previous participants will be required to approach the research team to be included in the study as the research team will not collect any information from the previous participant or ask if they know of any potential participants. Instead, a participant, after completing the study, can approach a colleague and recommend participation to them. This will not be asked of participants.
No sensitive data will be collected regarding the participant. Participants will be interacting with the researcher in a lab environment. Other lab members may be present in the lab when the study is being conducted; however, the participant will be separated from the rest of the lab by participating in a separate cubicle. The additional option to complete the study remotely will not involve recording the user in any way and will allow for the participant to complete tasks privately if they are unable to attend in person or are uncomfortable. Headphones will be utilized by the instructor when conducting the study remotely, so other lab members will be unable to hear participant responses.
If a participant no longer wishes to participate in the study, they may leave at any time, and the data collected will not be utilized in the final analysis and will be destroyed.

Data Management and Confidentiality:
All data will be stored on a password protected computer owned by a member of the research team. Only the research team will have access to this data. No sensitive information about participants will be recorded.

Approach to Analysis:
The effectiveness, ease of use, and subjective satisfaction of the dashboard will be evaluated. Listed below are the variables considered in this study. Values will be compared across independent variables to determine comparatively if the dashboard is more effective than current solutions.

Independent variables include the following:

- Interface at 3 levels:
    - Globus online GUI
    - DDRIS Dashboard
    - Linux (Ubuntu) Command Line Interface (CLI)
- Familiarity with CLI at 3 levels:
    - Not familiar
    - Moderately familiar
    - Very familiar

Dependent variables include the following:

- Time to complete task
- Number of clicks
- Number of mouse-wheel scrolls
- If the task is completed
- Number of questions asked to the moderator or available search engine
- Preexisting knowledge of how to complete a specific given task
- Subjective satisfaction
- Difficulty score provided by the participant in the exit questionnaire

References:

Appendix C

UNR IRB Consent Information Sheet

Consent Information Script or Sheet

We are conducting a research study to learn the ease of use, efficiency, and design of a web dashboard.

If you volunteer to be in this study, you will be asked to complete a pre and post usage questionnaire at the beginning and end of the experiment along with tasks to assess the usability and efficiency of the dashboard. No identifying information will be collected over the course of the study, including recordings, images, or names.

Your participation should take about 45 minutes of your time***.***

This study is considered to be minimal risk of harm. This means the risks of your participation in the research are similar in type or intensity to what you encounter during your daily activities.

Benefits of doing research are not definite; but we hope to learn the effectiveness of the dashboard developed in comparison to existing solutions. There are no direct benefits to you in this study activity.

The researchers and the University of Nevada, Reno will treat your identity and the information collected about you with professional standards of confidentiality and protect it to the extent allowed by law. You will not be personally identified in any reports or publications that may result from this study. The US Department of Health and Human Services, the University of Nevada, Reno Research Integrity Office, and the Institutional Review Board may look at your study records.

You may ask questions of the researcher at any time by calling Sergiu Dascalu at (775) 636-5060 and/or Brianna Blain-Castelli at (916) 759-9502 or by sending an email to Brianna Blain-Castelli at bblaincastelli@unr.edu.

Your participation in this study is completely voluntary. You may stop at any time. Declining to participate or stopping your participation will not have any negative effects on you

You may ask about your rights as a research participant. If you have questions, concerns, or complaints about this research, you may report them (anonymously if you so choose) by calling the University of Nevada, Reno Research Integrity Office at 775.327.2368.

Thank you for your participation in this study!

Appendix D

UNR IRB Pre-Test Survey

University of Nevada, Reno
Social Behavior Research

Title of Study:	Near Real-Time and Historical Cyberinfrastructure Data Visualization
Principle Investigator:	Sergiu Dascalu, Ph.D.
Co-Investigators:	Brianna Blain-Castelli
Study ID Number:	1815780-1
Sponsor:	N/A

Pre-Usage Questionnaire

Participant ID#: _______________

Please Enter your age: ____________________

What is your gender?

A. Male
B. Female
C. Other
D. Prefer not to disclose

What is your highest level of completed education? ________________

Please rate how familiar you are with dashboards
(1 - Not Familiar, 5 - Very Familiar)

Not Familiar	1	2	3	4	5	Very Familiar

Please rate how familiar you are with Linux command line
(1 - Not Familiar, 5 - Very Familiar)

Not Familiar	1	2	3	4	5	Very Familiar

Please rate your familiarity with data transfer solutions
(1 - Not Familiar, 5 - Very Familiar)

Not Familiar	1	2	3	4	5	Very Familiar

What data transfer solution are you familiar with (mention one or more, or leave blank if you are not familiar with any data transfer solutions)?
____
____
____
____

How often do you use data visualization?
(1 - Not Familiar, 5 - Very Familiar)

Not Familiar	1	2	3	4	5	Very Familiar

How familiar are you with web applications/web browsing?
(1 - Not Familiar, 5 - Very Familiar)

Not Familiar	1	2	3	4	5	Very Familiar

Appendix E

UNR IRB Post-Test Survey

University of Nevada, Reno
Social Behavior Research

Title of Study:	Near Real-Time and Historical Cyberinfrastructure Data Visualization
Principle Investigator:	Sergiu Dascalu, Ph.D.
Co-Investigators:	Brianna Blain-Castelli
Study ID Number:	1815780-1
Sponsor:	N/A

Post-Usage Questionnaire

How helpful/effective do you find the DDRIS dashboard in its current state?
(1 - Not Helpful, 5 - Very Helpful)

Not Helpful	1	2	3	4	5	Very Helpful

How easy do you feel the DDRIS dashboard is to navigate?
(1 – Very Difficult, 5 - Very Easy)

Very Difficult	1	2	3	4	5	Very Easy

How well designed do you find the DDRIS dashboard interface in its current state?
(1 - Not Well Designed, 5 - Very Well Designed)

Not Well Designed	1	2	3	4	5	Very Well Designed

How difficult was it to complete the tasks assigned on the DDRIS dashboard?
(1 – Very Difficult, 5 - Very Easy)

Very Difficult	1	2	3	4	5	Very Easy

How difficult was it to complete the tasks assigned on the Linux command line?
(1 – Very Difficult, 5 - Very Easy)

Very Difficult	1	2	3	4	5	Very Easy

How difficult was it to complete the assigned tasks on the Globus management console?
(1 – Very Difficult, 5 - Very Easy)

Very Difficult	1	2	3	4	5	Very Easy

What changes could be made to make the dashboard easier to use?

____

If applicable, how can this dashboard be modified to improve your experience with system administration?

____

Any other comments or suggestions for improvements?

____

Appendix F

UNR IRB Participant Recruitment Email

Recruitment Email Script

General Recruitment Email:

Good [morning, afternoon, etc.],

I am conducting a user study to assess the usability, visual appeal, and effectiveness of a dashboard. I am currently looking for 16-24 participants to be a part of this user study and further this research. Participants of this survey will be asked to browse two websites, utilize Linux command line, and perform between 3 and 4 tasks on each interface. The study should take between 30 and 45 minutes of your time. Prior advanced knowledge of Linux command line is not necessary; however, basic knowledge is preferred. The study will be completed in EJCH but can be completed virtually through Zoom. If you are interested in participating in this user study, please email Brianna Blain-Castelli at bblaincastelli@unr.edu to schedule a time to complete the study.

Sincerely,

Brianna Blain-Castelli

Personal Recruitment Email:

Dear [participant],

I am conducting a user study to assess the usability, visual appeal, and effectiveness of a dashboard. I am currently looking for 16-24 participants to be a part of this user study and further this research. Participants of this survey will be asked to browse two websites, utilize Linux command line, and perform between 3 and 4 tasks on each interface. The study should take between 30 and 45 minutes of your time. Prior advanced knowledge of Linux command line is not necessary; however, basic knowledge is preferred. The study will be completed in EJCH but can be completed virtually through Zoom. If you are interested, please contact me with your availability to complete this user study.

Sincerely,

Brianna Blain-Castelli