Campus Compute Co-operative (CCC): A Service Oriented Cloud Federation

A DISSERTATION
PRESENTED TO
THE FACULTY OF THE SCHOOL OF ENGINEERING AND APPLIED SCIENCE
UNIVERSITY OF VIRGINIA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY(COMPUTER SCIENCE)

BY
MD ANINDYA TAHSIN PRODHAN
MAY 2019

Thesis advisor: Andrew Grimshaw Md Anindya Tahsin Prodhan

Campus Compute Co-operative (CCC): A Service Oriented Cloud Federation

Abstract

With the ever growing demand for computational resources, universities struggle to provide the cyber-infrastructure (CI) required to satisfy the needs of their researchers both in terms of quantity and diversity. Purchasing resources for peak demand for all resource types is not always plausible, and more importantly not at all efficient. Renting capacity from the commercial clouds can be an alternative, however, commercial clouds expect to be paid and are often very expensive. In this research, we present the Campus Compute Cooperative (CCC), which provides an alternative to commercial providers that yields increased value to member institutions at reduced cost. In CCC, member institutions trade their resources with one another in a market based environment to meet both local peak demand as well as provide access to resource types not available on the local campus. CCC also supports multiple differentiated qualities of service for the users.

Contrary to prior market based grid approaches, we focused our study towards a production level deployment. As a prelude to production deployment in multiple campuses we analyzed the challenges and prospects of such a federation with simulation. In this research, we simulated the CCC in a multi-campus environment with real production data traces from multiple institutions and evaluated the performance of the CCC with the current non-federated alternatives. Our simulation included data traces from three (3) institutions over the last quarter of 2017. From the results, we observe that, all the participating institutions can benefit from both the differentiated QoS and federation features of CCC and achieve a better net value individually and combined.

Our simulation shows that CCC increases the overall net value of the federation by $350,000 in a month. This research also provides some interesting insights about the data traces which would help other prospective institutions to decide weather they could benefit from joining the federation or not.

Contents

List of figures

List of Tables

To the three individuals I wanted to make proud with my work

my beloved Mom and Dad,

and my wonderful little angel, Aairah Seher Prodhan.

Acknowledgments

I thank, first and foremost, my advisor Professor Andrew Grimshaw. He has been an absolute angel for me through his guidance and support over the last 6 years.

My wife for her unconditional love and support. My sister for being a source of my inspiration.

My fellow graduate student, like my brother in the US, Muhammad Yanhaona, who has never shied away from helping me whenever I needed him.

My fellow student Abu Sayeed Mondol for always being their for me. My group mates Vanamala Venkataswamy and Swaroopa Dola for their constant support and encouragement.

Scott Ruffner and Essex Scales, who have helped me tremendously by making software and hardware resources available whenever I needed them.

My respected committee members for their support, guidance, and feedback.

My present and previous friends among the graduate students and office staff. I have learned a lot from them, their feedback have made valuable contributions to my work, and their company have made my staying in the US bearable.

From Indiana University Craig Stewart for leading the Campus Bridging effort, and Matt Allen for gathering the Big Red job files, Mark Gardner from Virginia Tech for his continuous support on the project and gathering the VT traces for us.

Finally, I thank this wonderful country for giving me the opportunity to do the study.

1

INTRODUCTION

1.1 Motivation

With the perpetual growth in computing and storage systems, computational techniques have become an essential resource in virtually all aspects of today's research. Universities are faced with the strategic challenge of providing researchers with the necessary computational infrastructure while managing costs.

The challenge in providing affordable computational resources is even more acute right now because the use of cyber-infrastructure (CI) has become ubiquitous across all the sciences and increasingly widely used in humanities research. Moreover, there are now a multitude of computational modalities to support research. Researchers perform many different types of simulation and analysis within and between disciplines, which requires different types of CI systems, i.e. GPU based systems, large memory systems, hadoop clusters, commercial clouds, tightly coupled HPC clusters etc. To make matters more complicated researchers may make use of multiple computational modalities at different points in a research project or even a single distributed workflow making use of resources at a variety of scales with a variety of temporal access patterns (continuous versus bursty usage). As a result, even for institutions with huge existing infrastructure, it is not possible to meet the demands of their researchers all the time.

Most universities maintain a set of shared resources to support the computational needs of their researchers. The typical scheduling model for these shared resources is first come first serve (FCFS). However "FCFS" policies on shared resources often lead to organizationally sub-optimal outcomes as not all jobs have the same value for the researchers and often the jobs with higher value have to wait on the queue while the regular or low priority jobs are running. This leads the funded researchers to buy their own cluster[1] rather than using the shared infrastructure, which are frequently underutilized.

Another alternative sometimes used by researchers is the use of commercial clouds (Amazon AWS, Microsoft Azure, Google cloud). With the commercial cloud however, it is not possible for the institutions to trade their resources in return. Which makes commercial cloud a very expensive alternative.

[1] During our survey of some of the cluster owners [66] [65], we found that the in-ability to immediately start jobs that are of higher value to them was a show-stopper for using shared university resources

Henceforth, for the majority of the institutions the obvious next step is to share the CI resources among each other to increase resource volume and diversity. However, sharing without proper accounting can lead to the tragedy of the commons where one institution can keep using resources from other institutions while nor allowing others to use their resources (or allowing very little), making the institution a chronic debtor.

Trading resources with each other rather than plain sharing can take care of this problem. This is where CCC [60] fits in. The basic idea of the CCC is simple: you can run your jobs to other institutions in exchange of allowing their researchers to run jobs in your institution. For example, your institution can trade in GPU cycles with other institution in exchange of HPC cycles. The remaining balances would later be paid by institutions with real currency at the end of each payment cycle. The use of a pay-as-you-go market mechanism avoids the free-rider problem that has plagued the earlier grid markets and cloud federation systems.

1.2 Shape of the Solution

To address the problem, we have developed the CCC (Campus Compute Cooperative), a secure, standard based, open source, federated, computational resource trading marketplace for research institutions. The CCC is built on XSEDE architecture [20] and leverages Genesis II [86] as its software stack. CCC combines three basic ideas into a production grid environment.

Resource Market The first idea is that resource providers charge for the use of their resources and resource consumers pay for the use of resources. Note that buying and selling does not necessarily involve the exchange of real money. Instead allocation units are exchanged between participating institutions. The mechanism for resolving chronic debits is discussed

in section IV. Charging for compute cycles is not novel in academia. Creating a market for cycles, where actors are both buyers and sellers is novel.

Differentiated Quality of Service (QoS) The second idea is providing differentiated quality of service levels that allow users to specify the quality of service they need. Quality of service attributes include the urgency of the job. Does the job need to run right now (urgent)? Or can the job run later (best effort)? Other quality of service attributes include the type of resource requested. Does the application have large memory requirements? Does it require a very fast interconnect? Does it require many cores on the same node? Or does it have a small memory footprint? Users select, and pay for, the quality of service they desire. By allowing users to express the value of the job in terms of what they are willing to pay we ensure that the CCC always executes the most important job next, increasing overall institutional value.

Resource Federation The third idea is federation: combining the resources of multiple institutions into a single, larger compute environment. This is akin to using an M/M/K queue versus K independent M/M/1 queues. By increasing the resource pool size we can provide better quality of service.

In a nutshell, the CCC allows users to select the urgency (quality of service) for their jobs and it provides institutions ability to securely trade their existing compute and data resources among each other. Users define their job information (resource requirements, data and environment setup) using a standard job description language (JSDL [47]). Once defined, jobs can be submitted to and executed on one of three different gridQueues (one for each QoS supported by CCC) based on their urgency. Each gridQueue is linked to one or more Basic Execution Services (BESes) [57].

Each BES is configured to submit jobs to a particular partition or queue in a local load management system such as SLURM, PBS, or SGE. For each BES that is linked to the gridQueue is configured with the maximum number of jobs it may concurrently submit to the BES as well as the maximum number of cores it may consume at any given instance. The gridQueue uses job resource requirements and BES resource properties to match jobs to BESes that are candidates to execute the job. Finally, The gridQueue schedules jobs to matching BESes in FCFS order.

Over the last decade, there have been several other attempts to create such market or QoS based approaches with limited success, except for the Open Science Grid [49]. We, however, believe that the CCC can result in an effective use of each participating organization's resources, and benefit the researchers at all participating organizations. This is because, we believe such market economies can succeed now and in the coming years within academia because the diversity of these resources needed by researchers is too great for even the best funded universities to maintain and support all of the existing modalities of advanced computing resources on one campus. For example, in the case of UVA and IU, we have a relatively greater abundance of HPC systems based on CPUs at UVA and a relatively greater abundance of HPC systems with GPUs at IU. So, a federation between the universities will allow researchers from both universities to access these diverse resources.

1.3 Campus Compute Co-operative Goals

There are two goals for the CCC. The first is to provide university scientists the computational resources they need when they need them in such a manner as to increase overall institutional value while reducing costs. We achieve this goal by deploying the CCC on participating institutional resources and setting up a mechanism in which universities (or university units such as departments and research labs) trade the use of their computational resources with other universities, e.g., com-

pute clusters, large shared memory machines. The CCC uses production quality software already available as open source created by the project participants. Three university computing resources serve as the pilot hardware – the Rivanna cluster at University of Virginia (6000 cores), Computer Science department resources at UVA (600 cores), and Big Red II at IU (20,000 cores).

The second goal of CCC is to test whether CCC results in a greater value and better ability for researchers to complete their most critical analyses for all the institutes individually and overall, compared to using local resources only, which leads to our hypothesis statement:

"A federation of resources among Universities with differentiated QoS and market based resource allocation will result in a greater value and better ability for researchers to complete their most critical analyses on-demand, for all the participating institutes individually and overall."

A corollary to our hypothesis is that by not federating their resources and by not distinguishing different qualities of service, universities realize a reduced return on their IT investment dollars. The null hypothesis is that the institutional value from computation and storage does not increase when participating in a market-based federated computational economy.

1.4 Data Calibration and Simulation

The CCC is not yet operational. As a prelude to production we wanted to model the market and evaluate the potential of such a market in a multi-campus environment. To evaluate and analyze the performance of CCC in a market based grid federation, we needed an input set of jobs that accurately reflects the system load we expect from CCC. Specifically, we needed job traces that would reflect user behavior in the presence of a federated environment and differentiated QoS. However, as CCC is not in production yet, such traces do not exist. Hence, we collected data traces from existing production environment and calibrated the data so that it reflects the expected system load

as in CCC. Specifically the data traces were not annotated with any quality of service requirement. So, we calibrated the data traces to annotate the jobs with defined job priorities. Finally, we conducted our simulation using ALEA [74] job simulator with the calibrated data traces to test the above mentioned hypothesis. We also analyzed calibrated the job traces to extract different patterns which justifies the prospect of market with CCC.

For the simulation, we used production data traces from three (3) different institutions: University of Virginia, Indiana University, and Virginia Tech, for the last quarter of 2017. The goal of our simulation is to identify the impact of the features of CCC, i.e.- resource federation and differentiated quality of service on different performance metrics. For that, we conducted simulation to measure the net value for each institution with or without the federation and differentiated QoS. We also measured the on demand access capability (which represents the ability to schedule the urgent jobs within a very short threshold) and resource utilization for each of the participating institution separately and as a federation.

Our results highlight that, with differentiated QoS and federation, all three (3) institutions gain significant value from CCC, which results a combined benefit of about $350,000 per month for the federation. Also with CCC, almost all cases more than 90% high priority jobs were scheduled within the 10 min threshold, which is about 20-30% improvement on the legacy systems.

1.5 Research Contribution

The most important contribution of our work is the implementation of a real market-based collaborative to explore the concepts and challenges of economic grid and their prospects in a production environment. The idea of "Grid economy or Grid Market" have been widely studied for last two decades. However, to the best of our knowledge none of these approaches being used in a pro-

duction environment, specially in academic domain. We believe our work lays the groundwork for deploying a production level market based grid federation in a multi-campus environment.

Our data analysis and simulation results establishes the potential of a multi-campus federation with differentiated qualities of service in an academic environment, Three (3) peer reviewed published papers[2] establishes the fact that our results were convincing enough for the peers. On the basis of our experience and these publications, the idea of CCC has stated to pick up some pace recently. This last quarter we have had eight (8) new institutions joining the collaboration- i) George Mason University, ii) University of Utah, iii) University of Nevada, Reno, iv) Texas Tech, v) North Dakota State University, vi) University of Texas, Dallas, vii) Rutgers University, and viii) Stanford University. Now we are in the process of setting up the infrastructure for CCC in a multiple campuses. We believe now the conditions are right for us to move forward and very soon we will have a production level CCC deployed in 10 institutions and start collecting real data from CCC [3].

One of the major challenges with CCC was to run the same jobs from users in different platforms (due to the diversity added with CCC). So far we have had some success stories from users at UVA, GMU and Utah in this regard. We have been successful in running applications like: LAMMPS, Amber, Keras in different platform for users, which encourages us to believe that CCC will become a useful work on academic super-computing. In addition, we believe our work will open up new dimensions in marked based grid research like: dynamic pricing, total cost of ownership analysis, accounting and settlements, job portability etc. In future, we are also considering using docker containers and Singularity containers to make the job-porting simpler.

[2] and a ready to submit paper to be submitted at SC 2019

[3] As of now we have already have the infrastructure setup at University of Virginia, Virginia Tech, George Mason University, Texas Tech, and University of Utah and users from some of those institutions have already started using the infrastructure.

1.6 Structure of the Dissertation

The remainder of the dissertation is structured in six Chapters-

Chapter 2 highlights some of the major work in the grid computing, cloud computing, grid economy, and federated clouds. Specially works that influenced our research (i.e.- Amazon AWS, Open Science Grid (OSG), and federated clouds) are covered with more detail.

Chapter 3 provides an overview the Campus Compute Co-operative (CCC). We present the system architecture of the grid middleware Genesis II, on which CCC is build upon, along with a brief on micro-economic models and the main features of CCC in this chapter.

For the simulation we collected job traces from three (3) different institutions over the last quarter of 2017. In Chapter 4 we summarize these data-set. We also present the calibration and analysis on the traces to setup our simulation.

Chapter 5 describes the simulation setup . Here we discuss the simulator (ALEA) that we have used and the modifications on the simulator to incorporate for CCC scheduling algorithm. We also provide the major model definitions and evaluation plan in this chapter.

Chapter 6 discusses the experimental results for CCC under 3 different setups. The chapter also highlights the breakdown of the result in terms of the market and analyzes the performance of CCC compared with the current setup.

Finally, Chapter 7 concludes with a reflection on the research findings, experience conducting the research, and what should be the short and long term plans for the future of this work.

2

RELATED WORK

There is a vast related literature in distributed systems, distributed scheduling systems, on-demand computing, cloud computing, grids, and grid economies going back over 40 years. Here we will focus on contemporary cloud and on-demand computing, grid economic models, distributed scheduling, pilot jobs, and one of the better known alternative systems, the Open Science Grid.

2.1 On Demand & Cloud Computing

Cloud computing has had a much more significant impact in the commercial space than either Grid computing or earlier metasystems efforts. This is the result of a number of factors including the use of virtual machines as sand-boxes that gave users tremendous flexibility; the rise of pay per use on line media (movies, music), social networking, and other areas that paved the way for market acceptance; and the entrance of Amazon into the market place, a vendor that was not only trusted but assumed to know what it was doing. Regardless of the cause of the change, the change is clearly upon us.

The National Institute of Standards and Technology (NIST) defines clouds as having five essential characteristics [81]: on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service. The CCC is a cloud under the NIST definition. NIST further defines three service models, Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). The CCC straddles the definition of a PaaS and IaaS cloud. The CCC allows users to deploy applications that may or may not depend on CCC services, and in the near future will include the ability to deploy and execute virtual machines.

There are clearly many other vendors of cloud services. The most representative examples are Amazon with the Elastic Computer Cloud (EC2) and S3 services and Microsoft with the windows Azure Cloud. The CCC differs from these commercial cloud providers in two significant ways. First, in the CCC economic actors are both producers and consumers. When using Amazon or Azure it is clear that Amazon is selling and the user is buying. Thus, when interacting with commercial cloud providers there will always be a bill to be paid. In the CCC, bills can be paid in-kind. Second, new resource providers can join the community (enlarging the resource pool) without any

change required on the client side.

Note that within Amazon EC2 there are many different "instance types" that correspond to qualities of service[7]. The Amazon instance types are partitioned in one dimension by the relative scale of the instance: micro, small, medium, large; in another dimension by characteristics such as "general purpose", "compute optimized", "memory optimized", "GPU optimized", "IO optimized", etc.; and also by whether the instances are fixed price and will last as long as you want, or whether they are "spot" instances that are cheap – but which can be terminated at any time. The CCC uses a similar model.

2.2 Grid Scheduling and Grid Economics

There is a rich literature in both computational grids and grid economic models [29, 37, 39, 49, 51, 57, 61–63, 88, 92, 109, 110]. The CCC builds on and extends these earlier systems. The CCC is based on open standards developed in the Open Grid Forum [13] and elsewhere that could only be developed once the underlying concepts of authentication, job description and management, data transfer, and so on had been settled. One of the defining characteristics of most early metasystems and grid efforts was that resources were "free" or that users already had to have permission to use the resources.

In terms of grid economics [22, 39, 91] the work has been done in simulation, or developed but not used in a production environment. Several research systems like: Spawn [108], AppLes [28], Popcorn [97], G-commerce [110], OCEAN [110], Nimrod [21], GridSim [38], and GridEcon [22] have explored the use of different economic models for managing resources in grid environment. In each of these systems a market is established to trade resources between the resource providers and resource users. But none of these actually considered the quality of service requirements of the

users. Buyya et al. [39] proposed a QoS based scheduling algorithm but there the authors only considered the deadlines as a QoS measure, not the value of the job .

The CCC is different in that our explicit goal is to construct and operate a production quality market where the actors buy and sell computational services to test the hypothesis that markets combined with differentiated quality of service result in a win-win for all actors.

2.3 Comparison of CCC and Open Science Grid

The Open Science Grid [92] is the best known community execution environment today. With technology initially developed primarily for the high energy physics community in the late 90's and early 2000's, the OSG is transnational high-throughput computing resource used by several science communities. From the OSG web page [14]:

> "The OSG facilitates access to distributed high throughput computing for research in the US. The resources accessible through the OSG are contributed by the community, organized by the OSG, and governed by the OSG consortium."

There are a number of differentiators between the CCC and the OSG. The most compelling though is the difference in economic models. The underlying model in the OSG is that the community contributes resources in an altruistic manner, and that whole communities are allocated a portion of the resources for execution. There is no explicit quid quo pro. The motivation for an institution to contribute something concrete, particularly in this era of tighter resources and cost accounting is not clear. In the CCC on the other hand the motivation to make resources available is explicit.

A second difference is the differentiated quality of service mechanism that explicitly models the economic value of jobs. The CCC differentiates between less important jobs and urgent jobs, and

schedules them appropriately. The goal is to increase institutional value as measured by what users (and their respective institutions) are willing to pay. Thus, urgent requests, or requests from well-funded research teams, can be moved to the head of the line in a neutral, consistent manner.

A third difference is the class of applications that are supported. OSG is explicitly designed for high throughput sequential jobs. The CCC on the other hand is built on a standard extensible resource and execution model [24, 50] that does not care what the "activities" are. The CCC currently supports sequential, threaded, and MPI jobs. Extending to VMs will not be technically difficult.

Fourth is extensibility and openness. The CCC is based upon open, publically discussed, extensible standards developed in the Open Grid Forum. The OSG on the other hand is based upon Condor, a tightly controlled proprietary system developed at the University of Wisconsin.

2.4 Federated Clouds

The standard Cloud computing model [2, 6, 9, 15], where a client utilizes a single cloud data center, introduces several challenges. A cloud service unavailability can leave thousands of customers without access to essential and paid for resources. Also relying on a single cloud data center makes it hard to implement adequate responsiveness and usability to clients distributed worldwide. These factors lead to the usage of federation of clouds to achieve better QoS, reliability and flexibility.

The concepts of Inter-Clouds are relatively novel, and many works in the area use several terms interchangeably. Federated Clouds or Inter-Cloud computing has been formally defined as [42]: "A cloud model that, for the purpose of guaranteeing service quality, such as the performance and availability of each service, allows on-demand reassignment of resources and transfer of workload through a interworking of cloud systems of different cloud providers based on coordination of each consumer's requirements for service quality with each provider's SLA and use of standard inter-

faces."

Last few years there have been several state-of-the-art Inter-Cloud developments which include both academic and industry projects. The InterCloud [37] project developed at the University of Melbourne is one of the first initiatives in this field. It is a federated cloud computing environment that addresses the issue of provisioning application services in a scalable computing environment, achieving QoS under variable workload, resource and network conditions. The proposed architecture is centralized and is built around a central entity called Cloud Exchange (CEx). In essence, CEx acts like a marketplace where clouds can sell resources. The buyers may be other clouds or application brokers. Thus, the architecture is clearly market oriented and facilitates pricing aware application brokering. Each cloud data center participating in an InterCloud Federation needs to have an agent called Cloud Coordinator (CC) installed. CCs manage the federation memberships of the clouds by communicating with CEx on their behalf. The Cloud Broker identifies suitable cloud service providers published on the CEx, negotiating with CCs for an allocation of resources that meets QoS needs of users. The main difference between CCC and InterCloud is that the prices of the resources are somewhat fixed for CCC, while in case of InterCloud the prices are decided based on some auction mechanism. The reason of choosing a static pricing scheme in the CCC is that, the expenses from the user's point of view will be more predictable with CCC. Moreover, in CCC the CCC consortium decides on the prices for each QoS, so there would be more control of the central body in the system. Also in InterCloud the main focus of the QoS attribute is the availability and reliability of the resources, where as in CCC the QoS attribute dictates which job to run earlier and whether a job can be preempted or not. And finally there is no empirical study with InterCloud deployed in a real system, which on the contrary is one of the main focus for CCC.

In Dynamic Cloud Collaboration (DCC) [67, 68], the central entity that facilitates the feder-

ation is called the primary cloud provider (pCP). The pCP is the first cloud provider in the federation – the one that actually established it. The other cloud providers are called collaborating clouds. The primary cloud (pCP) maintains a registry of the services of the collaborating clouds. Application brokering is done via SLA contract with pCP. Cloud clients submit requests to the pCP, which on the basis of the requests' characteristics allocates resources within the federation. It has not been specified whether clients are allowed to declare requirements about the geographical location of the allocated resources. However, there has been work on the facilitation of a cloud market place where auctions for resources are held. One concern with the genetic algorithm model is that the convergence time during the auction face, which can lead away researchers with immediate need of resources. On the contrary, our CCC grid queue would be able to dispatch jobs immediately if the resources are available.

Reservoir [99] is an open federated cloud computing platform. In the Reservoir model, each resource provider is an autonomous entity with its own business goals. A provider can choose the providers with which to federate. There is a clear separation between the functional roles of service providers and resource providers. Service providers are the entities that matches the user needs by finding resources that their application need. However, service providers do not own the resources. They lease such resources from resource providers. Reservoir succeeds in defining a reference architecture capable of dealing with common IaaS requirements and even new ones, such as service-orientation and separation between infrastructure and services. The Reservoir architecture does not feature a central entity and is a peer to peer federation. The clouds communicate directly with each other to negotiate resource leasing. CCC on the contrary is architecturally more centralized through the grid Queue.

The Federated Cloud Management (FCM) architecture relies on a generic repository called

FCM Repository to store virtual appliances for all federated services [78]. It is replicated to the native repositories of the different IaaS providers. Clients interact only with the Generic Meta-Broker Service (GMBS) describing the requested service, and as far as they are concerned, further provisioning and scheduling are transparent [78]. For every IaaS provider in the architecture, there is a correspondent broker called CloudBroker managing the allocation and deallocation of VMs and dispatching incoming application calls to appropriate VMs. GMBS has access to the FCM Repository and can communicate with the CloudBroker components of the federated clouds. When GMBS receives a request from a user, it performs matchmaking between the request and an appropriate CloudBroker. The matchmaking is based on information from the FCM Repository and runtime metrics provided by the CloudBrokers. Each CloudBroker maintains an internal queue of incoming application calls and a separate priority queue for every virtual appliance. VM queues represent the resources that can serve a virtual appliance-related service call. The priority of each VM queue is based on the currently available requests in the queue, their historical execution times, and the number of running VMs. On the basis of the VM queues, the CloudBroker needs to perform appropriate VM creation and destruction. A CloudBroker also handles the aforementioned queue of incoming calls by redirecting them to the VMs created as a result of the management of the VM queues. Application brokering in FCM is done transparently to the client. In the definition of the architecture, nothing has been said about user-level control over the location of the used resources and how cost optimization can be achieved [78]. We could speculate that location-specific requirements can become a part of the SLA between the clients and the GMBS. In a separate work, Kecskemeti et al. extend the FCM approach to facilitate self-adaptable and autonomous management of the federation [73]. One of their goals is to achieve resource optimization (and consequently costs optimization) without violating the SLAs. However, this optimization concerns the

resource utilization of the whole federation and does not necessarily lead to cost optimizations for all federation's clients.

One of the major difference between CCC and FCM is that, FCM mainly focuses on IaaS system, hence the FCM architecture only works for VMs, whereas CCC works in both VMs and real systems making it a viable option for both PaaS and IaaS system. Also there is no performance evaluation with FCM, so it is hard to quantify the benefits of such cloud federation.

The European project Contrail [41] is built around a centralized composite entity that acts as a single entry point to a federation of cloud providers. It is responsible for periodically observing the states of the cloud providers and facilitating a federation-wise SLA. It also provides single sign on, so that users need to authenticate only once to work with the entire federation. A special architectural component called Federation Runtime Manager (FRM) is dedicated to mapping users' requests to cloud resources. It implements cost and performance optimization heuristics and ideally should have access to the geographical location and other meta-information about every cloud provider [41]. Application brokering is achieved by specifying a detailed application SLA. It is the responsibility of the Federation Runtime Manager module to provision in accordance with the SLA and to minimize the costs.

Our Grid Queue performs similar tasks as the FRM in Contrail, however in Contrail, there is no benefit for sharing. In that regard Contrail is somewhat similar to OSG. But with these type of systems it is often difficult to motivate providers to share their resources. Also there is no experimental evidence (performance measure) of value addition with Contrail project.

Bernstein et al. envision a worldwide federation of cloud providers, rather than separate small-scale federations [34]. They draw analogies from previous experience in integrating separate systems into large-scale utility services, for example, electricity and phone systems and the Internet.

They propose a new Inter-Cloud architecture following some of the main design principles of the public Internet. In a global Inter-Cloud environment, it is important for cloud providers to discover each other and to match their needs for resources. Bernstein et al. propose the usage of a global resource catalogue called Cloud Computing Resource Catalogue. It contains information about the shared resources within the federation and should be hosted by several community governed Intercloud Root servers. Root servers replicate the catalogue among each other to provide better performance and availability. Negotiation between providers is facilitated by Intercloud Exchanges – distributed servers that allow cloud providers to discover suitable resources. Intercloud Exchanges perform match making between cloud providers on the basis of specified preferences and constraints. Intercloud Exchanges use the resource catalogue stored on the Intercloud Roots to provide a Distributed Hash Table suitable for fast querying. After the negotiation is done, cloud providers continue to communicate directly with each other.

To facilitate all phases of the described global Inter-Cloud system, each participant should have an entity called Intercloud Gateway. These entities allow cloud providers, Intercloud Roots and Exchanges to communicate with each other using a set of Inter-Cloud protocols and standards. To date, such interoperability protocols and standards have not been widely utilized, although there are already ongoing works in the area [30–33].

Arjuna Agility [1] is a commercial framework for establishing Inter-Cloud Federations. It is mostly tailored for federations of in-house data centers but can also facilitate cloud bursting (usage of public clouds) if the demand increases beyond in-house resource capabilities. Each federation site needs to install an Agility software agent that governs the interaction with the other sites. Each site has its own policy regarding resource sharing and can define what resources in terms of hardware and software are shared. Being targeted mostly at in-house cloud federations, Arjuna Agility

addresses provider specific problems like reducing the power consumption [53, 79]. To this point, it does not feature cost optimization. Priority usage of local resources is addressed as a data centre borrows resources only if it cannot meet its own demands. Provisioning decisions are governed by provisioning policies set by an administrator. They are federation specific, not application specific.

Several companies provide paid independent Multi-Cloud services [5, 16, 17]. In essence, their features are similar, and they compete against each other on the basis of the performance overhead they add and the variety of cloud providers they can integrate with. RightScale offers a service for deploying and managing applications across clouds. Users can manage VMs on multiple clouds through the RigthScale console [16]. Application brokering is achieved through the alert-action mechanism, similar to the Trigger-Action mechanism [16]. All RightScale VMs have predefined 'hooks', which continuously send information back to the RigthScale console. This facilitates the check of the alert conditions and the execution of the predefined actions. Actions can define scale up/down policies but can also be administrative in nature (e.g. sending email to the system administrator). As an action, a user is allowed to specify in which cloud and location resources should be provisioned. Thus, scaling within RightScale can be location aware. Users can add private clouds within the RigthScale console to facilitate local resources utilization.

The services of EnStratius [5], and Scalr [17] are comparable with those of RightScale. Naturally, there are difference in the pricing, the set of supported cloud vendors and technologies and some terminology. Similar to RightScale, they allow users to deploy VMs across different public and private clouds. Application brokering is achieved through automated triggers, similar to RigthScale's alerts, which can trigger scale up/down actions.

Federated cloud is a novel area of intensive research and development whose body of knowledge has not been well established yet. However, all these projects and applications on federated clouds

actually support our motivation claiming the heterogeneous resource and computing requirements in the real world and how a single organization (even as big as Amazon Web Services or Microsoft Azure) is not suitable of handling all these requirement. Nevertheless, most of these projects or research have been focusing on the theoretical domain of the problem with hardly any practical experimentation with real user data. It is very understandable as these real data are often very hard to come by. Also none of these works on federated cloud actually provides real empirical data to quantify the benefits of cloud federation.

We, on the other hand, believe that we can actually get data of real applications run by real researchers in a real production quality test-bed as two universities are already committed to the idea and hopefully a few more universities to follow suit. Also with these federated cloud environments, any resource user cannot be a resource provider in the market, in our case any user with even a small desktop computer can join in the market as a provider, which introduces more dynamism and heterogeneity in the market.

3

SYSTEM MODEL

The CCC builds upon the XSEDE Execution Management Services (EMS) [26], the Global Federated File System (GFFS) [59], and the Campus Bridging use cases [105]. It is a secure, standards-based, open-source, federated cloud environment that provides execution management services, data access and management services, and identity and group management services. The CCC uses Genesis II [86] [10] [75] as its software stack. Genesis II is a web-services, container-based

architecture that uses open standards, (primarily from the Open Grid Forum [13]) such as Basic Execution Services, JSDL, RNS, byteIO and WS-Secure Token Services.

In CCC, we combine a market based resource trading with differentiated QoS on top of the software stack. Before moving into the details of CCC system model, we need to introduce the platform where CCC is implemented and some basics about micro-economics which is a major building block for CCC. This chapter presents a brief overview of the CCC software stack and micro-economics basics, followed by a description of the features that CCC includes to improve the state of the art.

3.1 CCC Software Environment

The CCC software environment consists of client-side tools that are used to interact with the CCC services and Web Services containers that are deployed to proxy access to back-end compute, data, and identity resources on participating institution machines. We are using Genesis II for both the client and server software. Client and container installers are available for Linux, Windows, and MacOS. RPM's are available for Linux as well. The Genesis II implementation along with underlying web services "stack" are all open source and are part of the XSEDE software stack. More details about Genesis II software is available in [10] [75]. We will use the phrase GORM in this document to refer to the Genesis II OmniBus Reference Manual [75].

Genesis II was developed at UVA and is the implementation of the XSEDE Global Federated File System (GFFS) [59] and the XSEDE Execution Management Services (EMS). The XSEDE architecture is described in detail in the XSEDE Level 3 Decomposition [26] (hereafter the L3D) and the GORM.

The GFFS and EMS are based upon standard, open, Web Services components that interact as a

service cloud. The "object model" used is one in which all objects are defined by a WSDL interface, have state, have associated metadata accessible via WS Resource Framework [27] get/set resource properties, have an address represented by a WS-Addressing End Point Reference (EPR) [36], and have a globally unique WS-Naming Endpoint Identifier (EPI) [55].

Below we examine four features of the CCC software system: the GFFS directory structure, the security model, the execution management architecture, and the accounting mechanism. For each we give a brief description followed by CCC-specific configuration information.

3.1.1 Directory and Discovery Service

A central feature of the Genesis II and the XSEDE architecture is the Global Federated File System name-space. The GFFS name-space is modeled on the Unix directory structure. Like a Unix file system maps path names to inodes the GFFS maps path names from a root "/" to resource endpoints known as EPRs that support the WS-Resource Framework model, and that implement some service type. The basic service types are WS-Trust authentication services [87], ByteIO file services [84], RNS directory services [85], directory export services (L3D), BES execution services [50], grid queue services (L3D), DAGMAN workflow execution services (L3D), and running jobs (activity endpoints) (L3D). In this proposal we will focus on Execution Management Services. For details on any of the services and how they interact see the L3D, the GORM, and the use case architectural responses [20].

The XSEDE GFFS name-space has a set of conventions on how the name-space is organized, e.g. a compute resource in the XSEDE name-space is */resources/xsede.org/BigRed2*. A running job is */resources/CCC/RegularQueues/jobs/mine/running/simulation-1-4-best*. A directory made available on my desktop is */home/xsede.org/prodhan/desktop-dir*. Keep in mind that an endpoint may

have many aliases, i.e., there may be many paths that point to the same endpoint; the Regular-Queue above may also exist as */resources/xsede.org/queues/normalQ*. The CCC uses the XSEDE name-space. The Genesis II client supports access to the GFFS name-space via a command line interface, via a GUI, via APIs, and by "mounting" the GFFS name-space as a file system on a Linux system using FUSE [106], or in Windows via a SMB mount. Once the GFFS is mounted as a file system by the operating system, existing operating system tools such as bash or user applications can directly operate on remote data and compute resources, e.g., cd into the working directory of a running job and **tail -f** an output file.

CCC Configuration

The CCC has its own directory entries in the XSEDE name-space, */groups/CCC*, */home/CCC*, */users/CCC*, */docs/CCC*, */resources/CCC* and */bin/CCC*. Beneath each of groups, users, and home there are links to institution specific, and institution maintained directories that point to the institutions users, their home directories, and any institution-specific and public groups.

3.1.2 Security Model Services

The CCC security model is described in the XSEDE L3D and has been extensively reviewed. In a nutshell on-the-wire data integrity is provided by https (using TLS), authentication is via signed delegated SAML certificates with a base of the certificate chain signed using an X.509 certificate, and authorization is via access control lists(oarwx) in which the principles are the signers of the base SAML assertions. Authentication tokens are transferred in the SOAP header of a web services call as specified in the Web Services Interoperability Basic Security Protocol, known as the WSIBSP [80]. Clients, acting on behalf of users, accumulate a "credential wallet" that is a set of signed SAML

assertions, that is passed on every web services call.

The Genesis II server implements several WS-Trust Secure Token Services (STS) for different back-end authentication protocols. The current back-end STSs implement a username / password using an internal database, a Kerberos backend that authenticates against Kerberos servers, and an InCommon Enhanced Client [11] or Proxy back-end that interacts with CILogon [3] and InCommon services. STS endpoints can also represent groups (or roles). Instead of signing an assertion stating that a particular user is JOAN or SARAH, the assertion states that the properly delegated holder-of-key is a member of the group.

Genesis II clients interact with STS services to acquire delegated signed SAML user and group assertions that are placed in the clients credential wallet for subsequent outcalls. The user can add/remove tokens from their credential wallet.

CCC Configuration

We have constructed a set of groups for the CCC and placed them in */groups/CCC*. Each participating institution manages its own group of people authorized to use CCC resources. These groups are named */groups/CCC/<inst-name>/<inst-name-users>*. Authorization to acquire credentials to institutional groups is managed by the participating institution not by CCC staff. CCC staff maintain a group */groups/CCC/CCC-users*. Authorization to acquire ccc-users credentials will be given to holders of group credentials from participating institutions.

To use CCC compute resources, an individual first authenticates and acquires an end-user credential from an STS that his/her institution trusts, such as the institution's InCommon server. The credential is placed in his/her credential wallet. The individual then requests an institutional group certificate from */groups/CCC/<inst-name>/<inst-name-users>* and places that credential

in his/her wallet. Finally the individual requests the ccc-users credential from */groups/CCC/CCC-users* and places it in his/her wallet. Assuming the user has been properly authorized at every step, the user now has the CCC-user credential and can use CCC resources. These steps are configured to execute immediately once the user credential is available. Participating institutions choose who will use their credentials and can revoke access at any time. Similarly, CCC staff control which institutions are permitted to acquire CCC-user credentials. Finally, note that any resource owner can stop accepting CCC jobs on a resource any time with a single command on a resource end-point, grid *chmod-rx<path-toresource>CCC-group*

3.1.3 Execution Service Model

Execution Management Services are described in detail in the L3D and GORM. EMS services are concerned with instantiating and managing to completion units of work that may consist of single activities, sets of independent activities, or workflows. EMS addresses problems with executing units of work including their placement, "provisioning", and lifetime management. These problems include, but are not limited to, the following: finding candidate execution locations, selecting the execution location, preparing the execution location for execution, initiating the execution, and managing the execution. The solution to these five problems consists of a standard job description mechanism and the use of set of services that decompose the EMS problem into multiple, replaceable components that all enable specific architecture functions. These components include the Job Submission Description Language [27] documents to describe jobs, OGSA Basic Execution Services (BES) [28] to execute jobs, Grid Queues to manage large sets of jobs and to schedule them onto BESes, GFFS directory paths [30] and resource registries [29] to discover resources, and job managers to implement application-specific functionality.

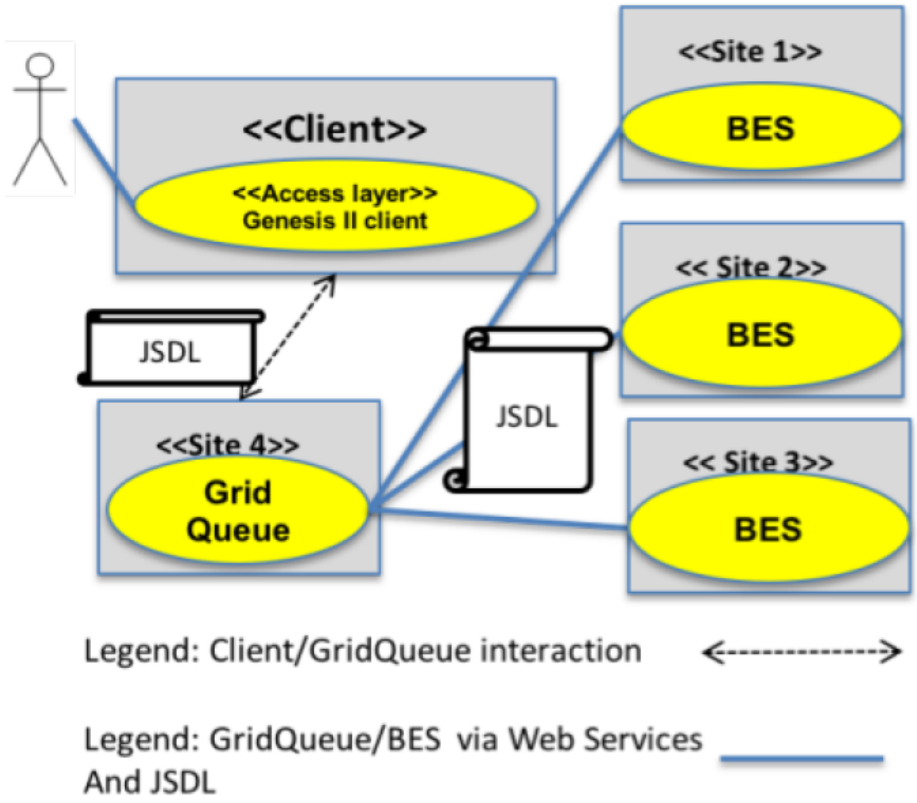

Figure 3.1.1: Clients send jobs to gridQueues. GridQueues send jobs to BESes. BESes wrap compute resources such as clusters. BESes may execute the job directly or delegate to other BESes.

Client tools

Client GUI tools include the Job-Tool for creating and submitting JSDL files and the queue manager tool. End-users can manage (list/start/stop/re-start clean up) their jobs with the queue manager tool. Users can also examine detailed log histories to understand why a particular job may have failed. Users can also interact with their running jobs using the GFFS by cd'ing into the job's working directory and reading (or tailing) output files being generated by the job regardless of where it

is running. Administrators can change scheduler parameters and manage end user jobs. Command line tools to submit and manage jobs are available as well for those who prefer scripting.

JSDL

JSDL [47] [70] is a standard XML based language used to describe jobs. A JSDL 1.0 document has three main components: a resource requirements section, an application information section, and a data staging section. The JSDL resources section contains application requirements such as operating system version, minimum amount of memory, number of processors and nodes, wall clock time, file systems to mount, and so on. It consists both of a standardized set of descriptions, as well as an open-ended set of matching requirements that are arbitrary strings.

The JSDL application information section includes items such as the command line to execute, the parameters, the job name, account to use, and so on.

The JSDL staging section consists of a set of items to stage-in before the job is scheduled in the local environment, and a list of items to stage-out post-execution. Each staging defines the protocol to use, the local file(s) to use as the source or target, and URIs for the corresponding source or target. Supported protocols include http(s), ftp, scp, sftp, GridFTP, mailto, and the XSEDE GFFS.

JSDL++ [100] is a non-standards track JSDL extension developed to address the short-coming that each JSDL document describes exactly one set of possible resource matches with exactly one corresponding application execution description. For example, "the job requires 8 nodes, each with 8 cores, 64 GB memory, and MPICH 1.4: in that environment stage-in executable Y and execute 'Y 1024 -opt1' ". Suppose an equally suitable option is "the job requires 1 node, each with 64 cores, 256 GB memory, and pthreads: in that environment stage-in executable Z and execute 'Z -opt2' "? JSDL++ allows the specification of an arbitrary list of options and the JSDL processing agent is free

to use any one of the options for which it can find the resources. The diversity of compute resources and tuned, platform-specific implementation for common packages, has led to use JSDL++ in the CCC.

OGSA Basic Execution Services (BESs)

OGSA BES service [50] [102] endpoints represent the ability to execute jobs, specifically execute JSDL documents. The BES interfaces combined with JSDL create a virtual execution environment for XSEDE and the CCC in which all execution resources, desktops, department servers, campus clusters, clouds, and supercomputers provide the same interface.

GridQueues

The basic idea for a grid queue is simple and is shown in Figure 3.1.1. GridQueue services are instantiated and configured to use particular BES services at each site. For example, the highPriority gridQueue will be configured to use the highPriority BESes at each site. Clients (users) send their jobs defined in JSDL to gridQueues. The gridQueue matches job resource requirements specified in JSDL against BES resource properties and schedules jobs onto BESes. In other words it provides execution services on-demand. The gridQueue monitors the job until it completes. If the job or BES fails, the gridQueue reschedules it on another resource.

CCC Configuration

To add their computing resources (clusters) to the CCC resource owners download and bring up a Genesis II container on a head node or login node that can submit jobs to their cluster. Once the container is operational the resource owners instantiate one or more BES endpoints on their

new container. Each BES resource is configured to submit jobs to a particular local queue, for example the "high priority" queue, "medium priority" queue or the "economy" queue. Each BES is given a path, e.g., */resources/CCC/BigUniveristy/FooClusterHighPriority*. The BESes have their access control lists initialized (rx) to allow members of the */groups/CCC/CCC-users* group to run and manage jobs.

Once the BES endpoints are instantiated and initialized they are added to the appropriate grid queue endpoint, e.g., */resources/CCC/queues/highPriority* or */resources/CCC/queues/regularPriority* or */resources/CCC/queues/economy* using the link command in the grid command line shell.

cd /resources/CCC

ln BigUniv/FooClusterHighPriority queues/high-Priority/resources/BigUHighPriority

Once linked in, end-users jobs that match the FooCluster resource properties are eligible to be scheduled on FooCluster.

3.1.4 Accounting

Whenever a job is run on the CCC an accounting record is created on the local BES that contains the time the job started/ended, the command line, the resource used, how much resource was used, and the signed SAML assertions that indicate who the user was, and what user and group credentials were used in the execution. The signature chains provide non-repudiation. Different resources types at different qualities of service have different costs in accounting units. This conceptually similar to the Amazon EC2 pricing model. Note that resource owners "earn" accounting units when jobs run on their resources and users home institutions "pay" for resource consumed by their users.BES resource usage records are swept up into the accounting database periodically. The accounting database can be sliced many ways; by resource to show all jobs run by users on

the resource; by user showing which resources they used; by institution as a resource consumer and producer, and so on. The objective is to provide institutions the ability to clearly understand the costs (their resources being used by others) and benefits (their users using other institutions resources) of participating in the CCC.

3.2 Micro-economics Basics

Before we dive into the discussion of CCC a bit of background on microeconomics[101] is appropriate. At the center of micro-economic theory are the market supply and demand functions. Both supply and demand are a function of the price of the item (usually a widget) being traded. A market equilibrium is reached when *supply=demand*.

The demand function for widgets is a decreasing function of the price of an object. In other words, when price of widgets goes up, the demand for widgets goes down. The difference between what the buyers would have paid, i.e. how much they value a widget, and the price is the buyers' surplus. If the surplus is not positive, the buyer will not buy as the buyer will only pay what an extra widget is worth to them.

Similarly on the suppliers side there is a supply curve that is an increasing function of the price, i.e., if the price is higher more widgets will be produced. Ignoring sunk capital costs, when the cost of producing a widget falls below the price, suppliers will cease producing widgets. The difference between the cost of producing widgets and the price one receives for selling widgets is known as the sellers' surplus.

Supply and demand are the foundation on which modern market economies are based. We posit that supply and demand applies to computational resources as well as widgets; that core hours right now are often worth more than core hours later, and that different types of core hours, e.g., those

with an associated GPU, are worth more to some users. It also follows that as the price of ”right now” core hours goes up relative to ”later” that the demand of ”right now” hours will decline.

Supply is also a function of the market. If the price of right now cycles goes up there will be more right now cycles available. We further argue that resources should be allocated to those who are willing to pay more as their willingness to pay directly reflects the value they place on the item.

3.3 CCC Features

Based on these micro-economics fundamentals and a very basic queuing theory principle, CCC introduces three features to increase the institutional value of the participating universities: i) Differentiated QoS, ii) Heterogeneous Resources, and iii) Larger Resource Pool. This section presents these features in the above mentioned order.

3.3.1 Differentiated Quality of Service

In a university there are funded researchers, unfunded researchers, new professors, graduate research groups, and others, each with different computational needs. Each user values particular runs of their jobs differently than others. Some are more important for example jobs needed for a conference deadline, others less. Further, some researchers' jobs are more important to the institution than others. We therefore assume that users have different utility function, i.e. users are willing to pay different amount for different jobs based on the importance of the job and the resources available to them. This is a core concept of micro-economics. The pricing structure for the popular cloud infrastructures such as Amazon EC2 or Windows Azure also serves as concrete evidence that the willingness to pay for different jobs for different users varies significantly.

In order to deal with this, in CCC we propose a differentiated QoS based market, where the QoS

will be defined by the user's willingness to pay for the job. Initially, we are considering three quality of services classes: high priority, regular priority, and low priority. We can add more service classes in the future based on interest from the users and the participating organizations.

Definition 3.3.1 (High Priority / Immediate Access Jobs)**.** This is the highest quality of service and the cost is maximum in this mode. In the immediate access mode, the job is guaranteed to be started within a predefined threshold maxDelay with probability onTimeStart from the time of submission. Further, the job will be allowed to run up to the time period specified by the user. In this mode, if the resource fails before the completion of the job, the resource provider will have to compensate the user for the failure. While if the job fails due to coding error or any other error from the user's part, the user will have to pay for the resource consumption nevertheless. Moreover, for these kind of jobs there would be a penalty to pay if the providers fail to start the job within maxDelay.

Definition 3.3.2 (Regular Priority / Long Uninterrupted Access Jobs)**.** This quality of service would provide best effort service without preemption. So the service will be inferior to high priority service as there is no guarantee of on-demand access, but better than the low priority jobs as there is no preemption. That is, in this mode, the job will be run on the available resources as long as no high priority jobs can be scheduled, but once the job is started it cannot be preempted without any penalty. In this mode too if the resource fails before the completion of the job, the resource provider will have to compensate the user for the failure.

Definition 3.3.3 (Low Priority / Best Effort Access Jobs)**.** This is the lowest quality of service and the cost of resource access will be the minimum in this mode. In the best effort mode, the job will be run on any available resource only if no other high priority jobs are waiting in the resource

queue and the job will be executed until any new high quality job shows up or it completes. If any high quality job shows up during the execution of the current job, the job will be immediately terminated.

In a nutshell, the CCC allows users to select a quality of service (i.e. urgency) for their jobs and the institutions to securely trade their existing compute and data resources. Users define their job information (resource requirements, data and executable files, command line) using a standard job description language. Once defined, jobs can be submitted to and executed on one of three different quality of service queues, high, medium, or low priority. In the Figure 3.3.1 users submit their job to one of three different quality of service (QoS) *gridQuues*. Each gridQueue is linked to one or more Basic Execution Services (BESes) [57]. Each BES is configured to submit jobs to a particular partition or queue in a local load management system such as SLURM, PBS, or SGE. For each BES that is linked to the gridQueue is configured with the maximum number of jobs it may concurrently submit to the BES as well as the maximum number of cores it may consume at any given instant. The gridQueue uses job resource requirements and BES resource properties to match jobs to BESes that are candidates to execute the job. The gridQueue schedules jobs to matching BESes in FCFS order. Now, how does differentiated QoS lead to better institutional value? The trick is, always run the high priority jobs whenever available, this way institutions can achieve more value from the same set of resources. So, jobs high priority BESes are always scheduled first and they can preempt jobs from low priority BESes.

3.3.2 Heterogeneous Resource

The resources at different universities are heterogeneous in terms of structure and architecture. Some universities provide cloud-based infrastructure for the researchers, other universities pro-

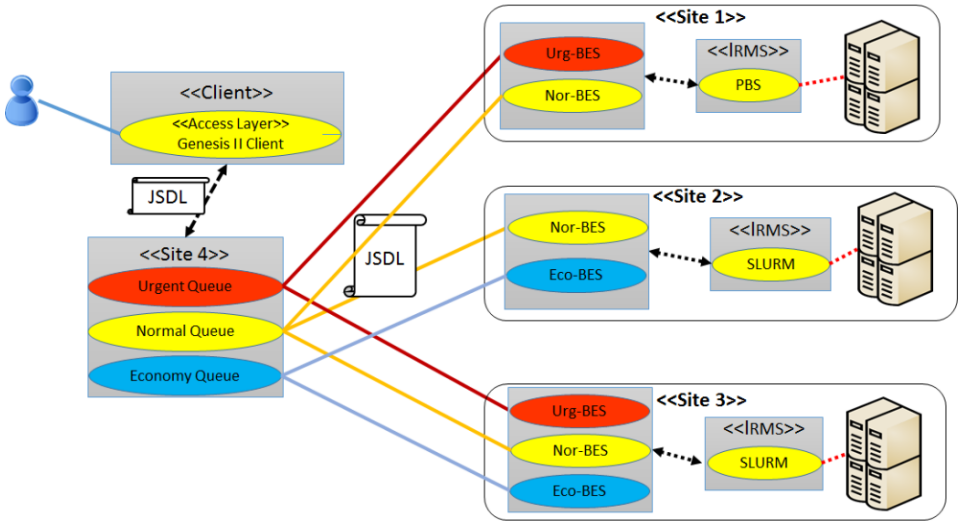

Figure 3.3.1: In the CCC, Clients send jobs to a set of gridQueues (each corresponding to a QoS specification). gridQueues then match the jobs to resources and forward the jobs to appropriate BESes based on the job specification. BESes wrap compute resources such as clusters and will manage the execution of the job on the underlying clusters.

vide tightly-coupled high performance computing infrastructure, while others provide GPU based computing infrastructure or a combination of the above. With the current surge in computational research, the diversity of computing resources needed by researchers is too great for even the best funded universities to maintain and support the requirements of their own researchers.

That is why, in CCC we propose a federation of resources among the interested universities so that researchers of each of the institute will have access to a more diverse set of resources to conduct their research. Individual research groups with their own clusters or even individuals with stand-alone desktop machine will be able to join the market to further the extent of heterogeneity.

With the current setup, often due to the job mix, many core-hours on the special (more expensive) resources are consumed by jobs that do not require the special feature. For example, single core jobs get scheduled in a parallel machine (machine with superior interconnection network),

regular jobs get scheduled in a GPU machine, etc. With the added heterogeneity in CCC, we will provide a better utilization of these expensive resources.

3.3.3 Larger Resource Pool

Another advantage of federation of resources is that with CCC researchers at each site will now have access to a greater resource pool which could lead to more value for the institutes. This is because with the larger resource pool, the institutions can now off-load some of their jobs to other institutions during peak load (rather than waiting on the queuing system). Thus the overall utilization of the system will increase and as will the institutional values of all concerned parties. The basics of queuing theory [9] also supports the benefits of federation. The idea behind federation is simple: replace K independent M/M/1 queues at each site with a single M/M/K queue. We believe that, just as utilizing an M/M/K queue out performs using k independent M/M/1 queues as k increases, so too will the benefit to the CCC community.

3.4 CCC Use-cases

These ideas (i.e.- federation and differentiated QoS) have been around for some time. The CCC combines them into an open-membership production compute environment for academic research as a means for addressing the computational challenges facing universities.

Three simple use cases illustrate how the CCC provides value to participating institutions-

First, well-funded Dr. Chemistry's research group at University 1 (U1) has a major grant review in three weeks and urgently needs immediate access to 5,000 cores for a week to prepare. U1 does not have 5,000 free cores. Most of their machine is tied up with another urgent job. Fortunately, U2 has no urgent jobs running and has plenty of capacity not being used by urgent jobs. Non-

urgent jobs can be terminated or suspended and Dr. Chemistry's jobs can now be run on a mix of resources, some at each school. By selling capacity to U1, U2 gains the ability to acquire resources from U1 when it needs them. U1 is able to meet its immediate need without over-provisioning resources.*Value of CCC: CCC provides a mechanism for one participating institution to obtain "burst capacity" from other institutions now, without paying funds to a commercial provider (which it cannot afford to do or is prohibited from doing for policy or regulatory reasons)*.

Second, Dr. Biology's application at University 2 (U2) needs 1 TB of memory. Dr. Biology only periodically needs to execute the jobs, but when executed dozens of 1 TB machines are needed. Unfortunately U2 only has two 1 TB nodes. U1 has two dozen such nodes. U2 could buy two dozen expensive 1 TB nodes and use them for smaller memory jobs when they are not needed for Dr. Biology. But the nodes are quite expensive. By using U1's nodes a capital expense is avoided. *Value of CCC: exchange of one type of computational resource for another, among two different institutions with different local resources saves on capital expenditure.*

Third, U1 economics professor Dr. MacroEcon models the world economy using large numbers of ensemble calculations, each one of which is itself a large scale Monte Carlo simulation. The total execution time is expected to be in the millions of CPU hours. Dr. MacroEcon does not have a large budget and is going to be in Europe for the next three months and does not particularly care when any given job completes. Instead, Dr. MacroEcon wants them to complete before returning from Europe. Dr. MacroEcon's jobs can be executed with a low priority and fill CPU slots when no higher priority jobs are ready. *Value of CCC: By being flexible about completion speed, researchers can get access to more resources for less.*

4

SIMULATION DATA

Due to the growing need for computational infrastructure, it is getting ever so difficult for most institutions to meet all the diverse requirements for all their researchers and computational scientists. That is where CCC comes in. We believe, through the federation in CCC, all the institutions can achieve more benefits compared to the current setup, both individually and combined. To test the hypothesis ahead of the actual deployment we chose to simulate the proposed cooperative. Us-

ing the simulation allows us to predict the costs and benefits to institutions as well as measure the sensitivity of the costs and benefits to changes in key variables and parameters. To do the most accurate simulation we require data-traces from a production system. However, as CCC is not in production yet, we had to rely on existing data traces from production queues at the member institutions. To have the available data reflect the expected system loads of CCC accurately we need to calibrate the data before it can be used.

In this section, We describe the data collection process followed by the calibration and analysis on the available data. We performed three (3) calibrations on the input data sets. These are:

1. *Collapsing the array jobs:* In CCC, we expect parameter sweep jobs to be submitted as one large array job. Hence, we identify the array jobs in the input set, so that they can be treated as a single sweep job.

2. *Setup job priority:* The jobs in the input set have no QoS information specified. Hence, we need to annotate the data set with job priority.

3. *Adding Additional load:* Federation increases both the resource capacity and resource diversity available to the researchers. So, we model additional loads to consume the surpluses.

4.1 Data Collection

To evaluate a market-based model with different resource classes and differentiated quality of service requires an input set of jobs that accurately reflects the system load we expect in production. Specifically, we needed a set of job traces that reflect user behavior in the presence of a federated environment with differentiated quality of service and diverse resources. The problem is that the

CCC is not yet in production, so such traces do not exist. Instead we have traces from existing, non-federated environments without differentiated quality of service (QoS).

A time-honored method would be to generate synthetic loads or to use job traces from existing systems. The problem with purely synthetic loads is that the simulation results are only as good as the method of generating synthetic loads. The alternative of using synthetic job traces is that the job traces are collected from existing systems which on the other hand, do not have jobs that reflect new capabilities with federation and QoS. For example, if system X has no large memory nodes then there will be no large memory job requests in the traces. Similarly, without priorities in the traced system there will be no priorities in the job traces.

Our solution is to start with job traces from real systems and then transform the traces to:

1. Have differentiated quality of service queues; and

2. Have increased load to reflect new capacity and capabilities.

We collected job traces from three institutions that are planning on participating in the CCC: UVa, VT, and IU. We collected 3 months of job traces from October 2017 to December 2017. The Table 4.1.1 summarizes the job specific information collected from the traces. In order to do analysis on the data (i.e.- group similar jobs together) we needed the ids of the users or groups along with the name of the job. However, it was also very important to preserve the privacy of the users while collecting the data. Hence, the user id, the group id and the job name was anonymized using an one-way hash function on the traces. The use of the hash function allows us to group users, user-groups, and jobs even when we do not have the details. Job arrival times, start times, execution times and other job related information was stored to recreate the traces in the simulation as accurately as possible.

Table 4.1.1: The job-specific information that is gathered from the job-traces of different institutions. This information is used to construct an accurate simulation under different set-up to evaluate the effectiveness of CCC.

Field Name	Description
Job Arrival Time	When the job was submitted by the user
Job Start Time	When the execution of the job was started
Name of the Job (Anonymized)	The name of the submitted job
Wall-clock Time Requested	The expected execution time of the job as specified by the user
Job Execution Time	Actual execution time of the job (Job Finish Time – Job Start Time)
Memory Requested	The amount of memory requested for the job by the user
Memory Used	The amount of memory actually used by the job
Resource Id	The id of the resource where the job was executed
Number of Cores Requested	Number of CPU/GPU cores requested by the user for the job
Number of Cores Used	Actual number of CPU/GPU cores on which the job was executed
Job Queue	Specifies the queue was the job was submitted in the local scheduler
User Id / Group Id (Anonymized)	Id of the user who submitted the job. This id will be used for accounting purposes.
User Institute	Accounts of the user/research group who submitted the job

From UVA, we collected the job traces from *Rivanna*[1], the High Performance Computing (HPC) system at UVA. *Rivanna* has 240 nodes (20 cores each) equipped with infini-Band interconnect for high-performance parallel jobs. In addition to the basic nodes, *Rivanna* also has 14 nodes with GPUs (10X 4 K80s, 4X 4 P100s), 5 large memory nodes with 1TB memory each and 8 with Intel Knight's Landing systems. Over three months, Rivanna had 243,592 jobs submitted through SLURM on 7 different queues. UVA traces includes 1,173 GPU jobs and 11,204 large memory jobs.

Table 4.1.2 provides a high level summary of the UVA traces. The researchers at UVA can submit

[1] https://arcs.virginia.edu/rivanna

their computational jobs at Rivanna in one of several different **SLURM** Queues. Most of the jobs are submitted in the *standard* Queue (about 85.5%). This queue is mostly used for general purpose sequential and HPC jobs. There is specialized queues for large memory, GPU and KNL resources. Researchers, who need these resources are required to have access to these queues. The parallel jobs are submitted in the *parallel* queue. And finally debug jobs are submitted to the *dev-part* queue. There was also a specialized queue for researchers in the Bio-Medical Engineering (BME) labs.

To get a better overview of the data we wanted to group the array jobs together. The third column of the Table 4.1.2 shows the job count after the array jobs are collapsed as a single parameter sweep job. We will describe this process in more details in Section 4.2.

Finally, we also summarized the core-hrs of the jobs executed on each queue. From that we can observe that, for UVA, most of the core-hrs were used on the parallel queue. Which is pretty much what we expected, as parallel jobs use multiple cores in parallel. The data shows about 2/3rds of the actual CPU hours are executed by parallel jobs.

Table 4.1.2: Summary of UVA job traces

Queue	Number of Jobs	Post collapsing Array Jobs	Core-hr
dev-part	25	22	4
eqa-bme4550	216	177	174
gpu	1,173	954	41,749
knl	10	10	24
largemem	11,204	1,914	26,948
parallel	22,473	6,176	2,263,557
standard	208,491	15,115	807,666
Grand Total	**243,592**	**24,368**	**3,140,120**

From IU, we collected the data traces from *Big Red II*[2] which is the main system for high per-

[2]https://kb.iu.edu/d/bcqt

formance parallel computing at IU. We also collected data traces from IU's large memory machine *Mason*[3]. However, Mason has recently been retired and replaced by a new large memory machine named Carbonate. During the data collection process Mason was in the process of retirement. As a result the utilization of Mason was in general very low. Due to this reason, we believed that the data actually does not reflect the real requirements of the large memory machine users and hence we decided not to use Mason traces for our simulation. We do not use Carbonate logs either the users were still in transition from Mason to Carbonate.

Big Red II is comprised of 344 XE6 (CPU-only) compute nodes and 676 XK7 "GPU-accelerated" compute nodes, all connected through Cray's Gemini scalable interconnect, providing a total of 1,020 compute nodes, and 21,824 processor cores. Each XK7 node is equipped with one NVIDIA Tesla K20 GPU accelerator. IU traces consists of 106,109 jobs over 3 months including 38,119 GPU jobs.

Table 4.1.3 summarizes the IU traces collected from the PBS logs at IU. BigredII machine had seven (7) **PBS** queues where the researchers submitted their computational jobs. There were 3 different queues for serial jobs- *normal*, *serial* and *long* queue, each providing different maximum wall clock time for individual jobs. The normal queue is for regular HPC and parallel (MPI) jobs, while *serial* and *long* queues runs significantly longer serial jobs (jobs that runs for about a week or more). The maximum wall clock time allowed on these queues are 2 days, 7 days and 14 days respectively. There were also specialized queues for GPU jobs and debug jobs. After collapsing the array jobs the job count for IU was reduced to 25,687 which shows that there were many array jobs present in the IU traces. Finally, when we summarized the core hours, we observed that most of core hours were spent for GPU jobs. This is why many of the GPU jobs at IU observed a very

[3]https://kb.iu.edu/d/aolc

high latency before they could start. Jobs submitted to the *normal* queue was the second largest contributor of executed core-hours.

Table 4.1.3: Summary of IU job traces

Queue	Number of Jobs	Job Count after Collapsing the Array Jobs	Core-hr
cpu16	817	455	51,499
debug_cpu	3,369	2,959	21,921
debug_gpu	1,291	1,113	6,953
gpu	38,119	5,011	16,818,291
long	551	324	2,226,224
normal	32,998	6,239	10,160,416
serial	28,964	9,586	6,126,348
Grand Total	**106,109**	**25,687**	**35,411,652**

At Virginia Tech, we gathered logs from the Splunk database. The logs contain traces from four (4) distinct HPC systems[4] available at VT: *Cascades, Dragonstooth, NewRiver* and *Blueridge*. These systems contain 817 nodes and 18,220 processor cores. Fifty-five of the nodes were equipped with Nvidia GPUs, while 130 of them had two 60 core Intel Xion Phi co-processors (mic). Four of the nodes were suitable to run large-memory jobs with a total of 3TB of memory each. VT job traces were collected from the MOAB logs for the 3 months period which includes around 5,000 GPU jobs and 350 large memory jobs and a total of 125,772 jobs.

The summary of VT job traces are presented in Table 4.1.4. From the table we can observe that VT researchers had access to nine (9) different **PBS** queues where they could submit their jobs to. Three (3) of them were for regular HPC and parallel jobs: *normal_q*, *open_q* and *large_q*. The *normal_q* was the main queue which included more than 90% of the jobs. The *open_q* and *nor-*

[4]https://www.arc.vt.edu/computing

Table 4.1.4: Summary of VT job traces

Queue	Number of Jobs	Job Count after Collapsing the Array Jobs	Core-hr
deq_q	45	39	11,749
dev_q	6,515	3,982	67,212
hxin_lab	6,368	3,212	2,128,424
ikoutrom_lab	1,086	479	567,328
large_q	148	9	1,821
largemem_q	350	220	264,413
nmayhall_lab	84	78	8,276
normal_q	102,755	41,838	39,099,938
open_q	3,003	1,654	184,568
p100_dev_q	2,282	2,079	9,695
p100_normal_q	2,272	1,797	637,311
rpollyea_lab	191	158	87,326
srinivasan_lab	140	52	55,899
stamps_geo_lab	130	120	8,940
vis_q	339	310	6,598
Wang_AOE_lab	64	61	109,135
Grand Total	**125,772**	**56,088**	**43,248,633**

mal_q essentially served the same purpose but as access to *open_q* was basically free (researchers do not need an allocation to submit to the *open_q*), hence the wall time allowed on *open_q* was significantly smaller than that of *normal_q*. On the other hand, jobs that run over a week or longer were submitted to the *large_q*. There were also specialized queues for large memory nodes (*largemem_q*) and GPU nodes (*vis_q*). Some of the nodes were equipped with NVIDIA Tesla P100 GPUs. Researchers could access those nodes through the *p100_normal_q* and *p100_dev_q*. There were several special queues for specific research labs (we assume they own part of the cluster or had their condo nodes on the cluster).

Finally, when we observed the core-hours, we found out that almost 95% of the core-hrs were spent on *normal_q* jobs. Which actually points to the importance of having multiple QoS, as oth-

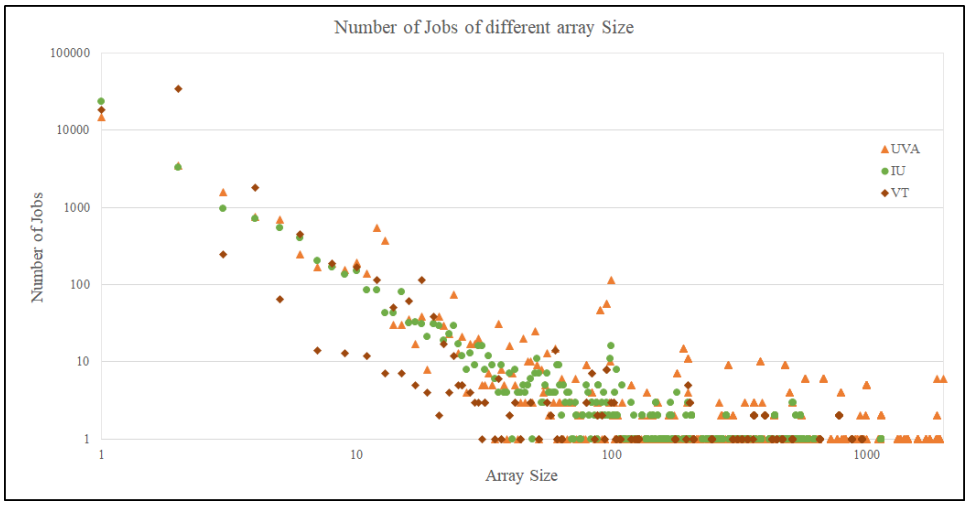

Figure 4.2.1: The Range of Array jobs in the traces for different institutions. Both the array size on the x-axis and the job count on the y-axis are represented in log scale to improve the visibility of the data. UVA has the larger array jobs among the three institutions. While there were over a hundred array jobs at UVA with array size bigger than 1024, the maximum array size at IU is 1144 and at VT is 958. These lower maximum numbers may be the result of load policy.

erwise the high priority jobs at would have to wait on the *normal_q* for a awfully long time.

4.2 Data Calibration & Analysis

With the available data traces, we first wanted to identify the array jobs in the system and collapse them as a single job. This is because, we expect CCC users to submit a single job to perform the array jobs (mostly parameter sweeps) and the Grid Queue will be responsible for expanding them before submitting to the appropriate queue. Also, while choosing job priority or inserting additional load, we wanted to consider the whole array as a single parameter sweep job.

From the available job traces, we found that many jobs look like part of a single array job. In CCC, we expect the array jobs to appear as a single parameter sweep job in the queue. So, we

wanted to identify array jobs present in the system and collapse them as one job for our analysis To do that, we iterated through the job traces and identified all the jobs with the same name (ignoring the job number in some cases), same user, and same job profile (number of cores required, similar execution time etc) which are submitted within a time threshold $array_{th}$. For the current traces we chose the $array_{th}$ to be 120 secs (2 mins). Once we identified the array jobs in the traces, we collapsed each sweep into one job with array width of $size_{array}$.

Figure 4.2.1 presents the number of jobs of different array sizes present on the data traces from all the three institutions. From the traces we observe that, many jobs from the UVA logs were array jobs or high throughput computing (HTC) jobs. As a result, after collapsing the array jobs reduced the UVA job count significantly from 243,592 to 24,368. UVA researchers also have the larger array jobs. We can identify many array jobs over the size of 1024 from the UVA traces. VT traces, on the other hand had the least number of HTC jobs. After collapsing the VT job count was reduced to 56,088 from 125,772 where the array size for the largest job was 958. IU job traces also contained a few very large HTC jobs (with a maximum size of 1144) and collapsing them reduced the number of jobs from 106,109 to 25,687.

At UVA, there are a number of resources which are only suitable for single node jobs and these resources are usually quite underutilized and cheap. As a result, there is an opportunity to move many of the single node jobs from within UVA or across multiple institutions to these resources, leaving the more expensive and oversubscribed resources to be used for specialized jobs. Hence, we broke down the different array jobs based on the width (number of cores requested for the job) of each individual job. Table 4.2.1 to Table 4.2.3 shows the count of jobs with different width categorized by the array size. Analyzing the available data traces, we can observe that most of the larger HTC jobs would fit in a single node, hence many these jobs can take the advantage of these

Table 4.2.1: Distribution of Array jobs in terms of job width (number of cores requested) at UVA. The data show that most of the larger array jobs usually fit in a single node, on the other hand most of the wider parallel jobs have a smaller array size (not HTC jobs).

Array Size	Job width	Count of Jobs	Total
1-16	1-20	20,062	23,211
	21-40	681	
	41-60	119	
	61-80	763	
	81-100	272	
	101-120	22	
	121-140	129	
	141-160	318	
	161-180	164	
	181-200	29	
	201-220	19	
	221-240	34	
	241-260	49	
	281-300	3	
	301-320	19	
	>320	528	
17-32	1-20	360	365
	21-40	2	
	41-60	1	
	61-80	2	
33-48	1-20	131	133
	21-40	1	
	>320	1	
49-64	1-20	90	103
	>320	13	
65-80	1-20	115	115
>96	1-20	404	404

underutilized and inexpensive resources and can be scheduled at these resources.

From Table 4.2.1 we can observe more than 50% of the jobs are single node jobs. With the smaller array jobs we actually see more diversity in the number of cores (width) requested, while

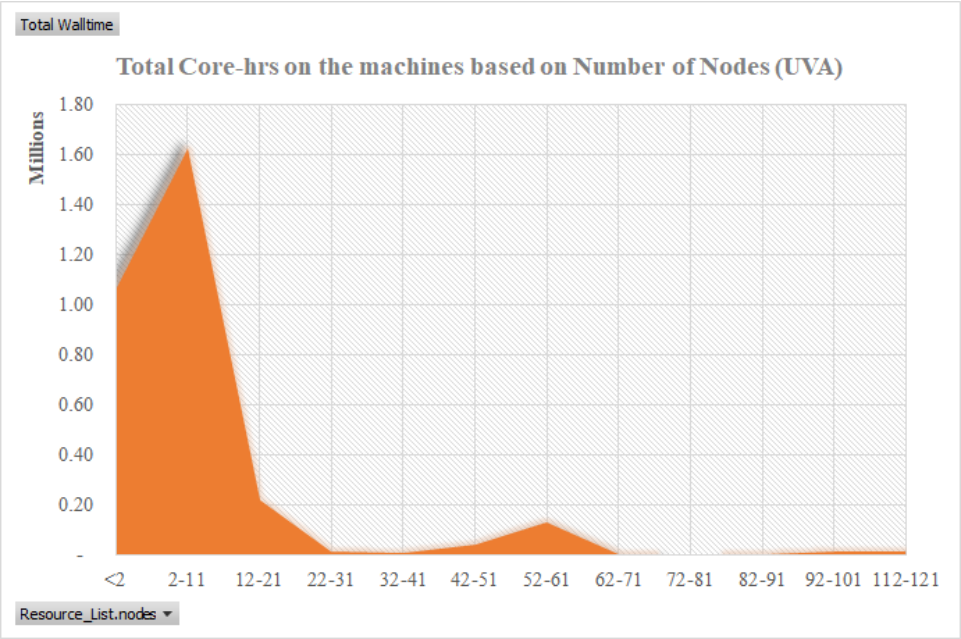

Figure 4.2.2: Total machine time (core-hours) occupied by different jobs in terms of the on the number of nodes requested at UVA. From the figure it is quite visible that more than a million machine hours are expend on single node jobs. There is also a smaller spike at about 50-60 nodes showing the presence of some parallel jobs at UVA.

most of the larger array jobs (HTC jobs) at UVA fit in a single node. Interestingly, from the table we can observe that, at UVA there were 13 jobs which have an array size between 49-64 requested more than 320 cores ($>$ 16 nodes).

From Table 4.2.2, we can observe that about 70% of the jobs from IU traces are single node jobs. There are quite a few parallel jobs as well, with 345 jobs asking for more than 16 nodes. Most of the parallel jobs has smaller array size.

Table 4.2.3 shows that at VT however, the percentage of single node jobs is about 50%, Which means about half of the jobs are parallel jobs. There are also more than 600 jobs that require more

Table 4.2.2: Distribution of Array jobs in terms of job width (number of cores requested) at IU. Also at IU, most of the larger array jobs are single node jobs while most of the wider parallel jobs have a smaller array size (not HTC jobs).

Array Size	Job width	Count of Jobs	Total
1-16	1-32	17,105	25,017
	33-64	1,381	
	65-96	3,591	
	97-128	1,716	
	129-160	131	
	161-192	136	
	193-224	10	
	225-256	172	
	257-288	22	
	289-320	106	
	353-384	9	
	417-448	5	
	449-480	3	
	481-512	286	
	>512	344	
17-32	1-32	227	252
	33-64	19	
	129-160	6	
33-48	1-32	80	89
	33-64	9	
49-64	1-32	69	79
	33-64	10	
65-80	1-32	22	30
	33-64	8	
81-96	1-32	26	32
	33-64	5	
	>512	1	
>96	1-32	115	188
	33-64	73	

than 16 nodes at a time. Compared to VT, IU and UVA have more larger array jobs (HTC jobs).

The number of single node jobs can often be a little misleading, as there could be many short jobs

Table 4.2.3: Distribution of Array jobs in terms of job width (number of cores requested) at VT. Interestingly VT had many parallel jobs (multi-node jobs) compared to IU or UVA and has very few very large array jobs.

Array Size	Job width	Count of Jobs	Total
1-16	1-20	23,518	55,775
	21-40	15,609	
	41-60	21,69	
	61-80	5,587	
	81-100	2,069	
	101-120	333	
	121-140	3,577	
	141-160	903	
	161-180	11	
	181-200	379	
	201-320	1021	
	>320	599	
17-32	1-20	157	220
	21-40	30	
	41-60	4	
	61-80	5	
	81-100	3	
	121-140	6	
	141-160	3	
	181-200	1	
	201-320	3	
	>320	8	
33-48	1-20	13	17
	61-80	2	
	81-100	1	
	301-320	1	
49-64	1-20	19	22
	81-100	1	
	>320	1	
>64	1-20	54	67
	21-320	11	
	>320	2	

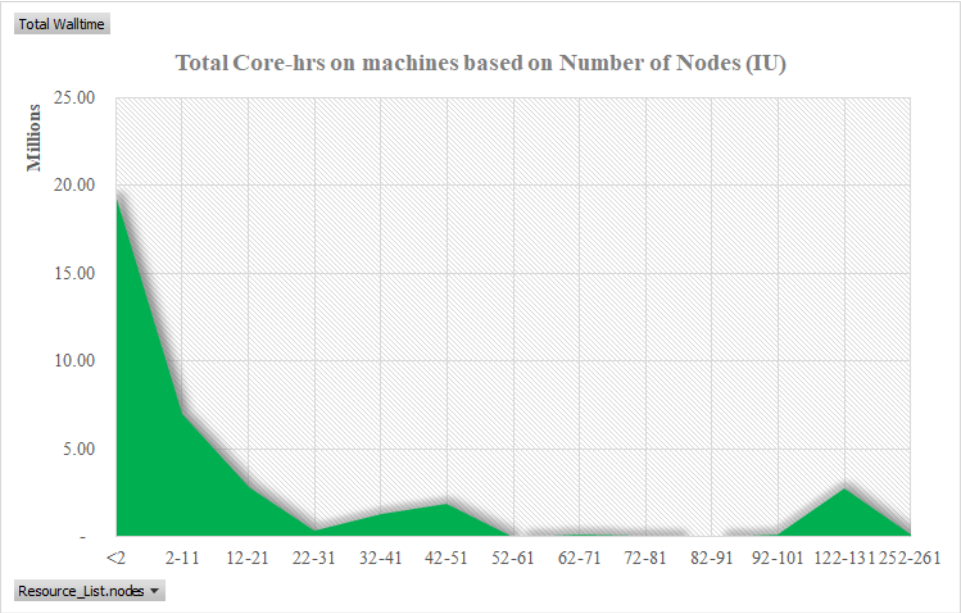

Figure 4.2.3: Total machine time (core-hours) occupied by different jobs in terms of the number of nodes requested at IU. From the figure we can easily observe that a significant part of the machine load is actually due to the single node jobs. We can also see a small spike at the end of the figure, which tells us that IU had a few very wide jobs (more than 100 nodes).

in the mix to inflate the number of jobs. We analyzed the amount of machine time each institution has for single node jobs and MPI jobs, so that we can get an idea of how much machine load can be off loaded to the HTC resources at UVA. To do that, we calculated the machine time ($walltime \times no\_of\_cores$) for the jobs categorized into the number of nodes requested for different institutions.

Figure 4.2.2, 4.2.3 and 4.2.4 presents the machine time for single node and MPI jobs for UVA, IU and VT respectively. From the figures we can observe that for UVA and IU, about 35-40% of the overall machine cycles are consumed by single node jobs, which is significant enough that both institutions can gain a lot from CCC. For VT however, most of the machine time is utilized for multi-node (MPI) jobs. Only about 10-15% of the machine cycles are utilized with single node

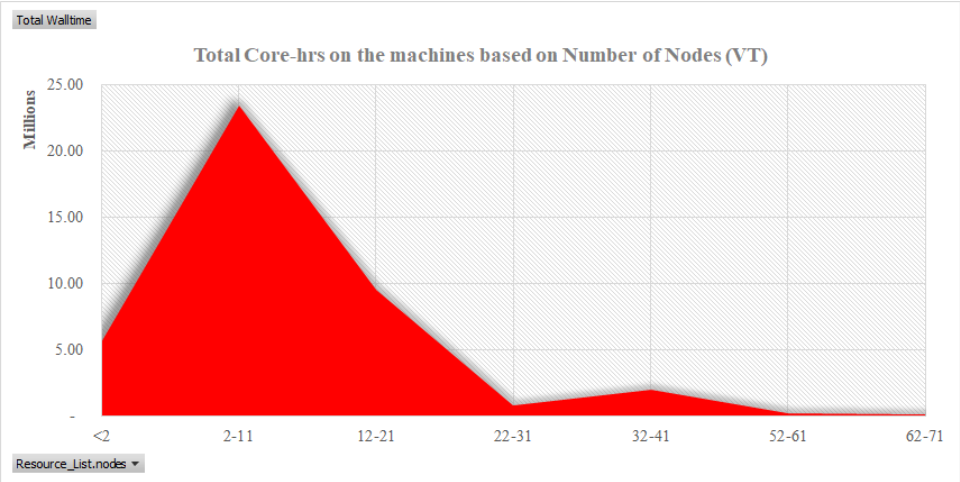

Figure 4.2.4: Total machine time (core-hours) occupied by different jobs in terms of the number of nodes requested at VT. The spike in the figure is more spread-out than it was for IU or UVA. Which means that there are few more parallel jobs at VT compared to the other two institutions.

jobs.

In terms of width however, we can observe that IU has more wider parallel jobs compared to the other two institutions (perhaps because they have the largest machine). The small spike at the end of Figure 4.2.3 is representative of that. The widest job in the IU traces was a 256 node job. UVA also had a few jobs requiring more than 100 nodes. The widest job at VT however had only 64 nodes.

4.3 Setup Job Priority

At the beginning of our project, we conducted several interviews [54, 65, 66, 69, 103] with different types of users in the compute resource user community of University of Virginia. We focused on three different types of users:

- the high end users (the users who use up the most shared resources)
- researchers with their own clusters and
- researchers who do not have the funds to buy cycles on the shared resources

As our available data traces did not have information about the priority of the jobs, our focus was to identify jobs which are more important to users and those for which the user can wait a little longer if they get the resources at a cheaper rate.

From our discussion with the community, we found that when people submit a large number of wide jobs together in the system, they often do not expect a quick turnaround time. On the other hand, from the discussion with the researchers that own their own cluster, we found out that most often researchers buy their own machine so that they can run their [5] jobs right away (which perfectly fits into our definition of high priority jobs) and/or control the schedule of their own jobs based on their priority. Hence, the jobs from these special queues are inherently high priority jobs. Conventionally these queues would be named after the research lab of the owner. Similarly, the debug jobs are by definition short and users usually expect them to finish right away for some quick result or a proof of concept. So they are typically high priority job.

Since none of the job traces come with defined job priorities for the jobs, the next piece of the puzzle is to assign priorities to the jobs. Based on our discussion with the above mentioned potential stakeholders, in CCC, we decided to categorize jobs into three priority groups: high priority, medium priority and low priority and came up with two heuristics to assign the priorities for the jobs on the traces-

Heuristics 4.3.1. *Large parameter sweep jobs which are also wide, are usually low priority*

[5] by their jobs we include the jobs from their post-docs and students

Heuristics 4.3.2. *The debug jobs and jobs submitted to special queues (i.e. queues that are exclusive to specific research labs) are usually high priority jobs.*

Based on these heuristics, from the collapsed traces with sweep jobs, we identified the jobs that have an array size greater than 16 and requested more than 640 cores and annotated them as low priority jobs. We also annotated the debug jobs and special queue jobs as high priority.

However, from the traces we can observe that the debug jobs and the special jobs account for only 5% of the overall job count. Moreover, the debug jobs being typically very short, in terms of core-hr, these jobs contribute even less with respect to the machine time.

In practice, from our experience and discussion with the stakeholders, we expect the high priority jobs to have a much larger share on the job mix than just the debug jobs and special queue jobs. Because people inherently have deadlines and right now users can't give the option to specify a job as higher priority. But we believe, given that CCC provides the support of multiple differentiated QoS, people will submit more jobs as high priority jobs with CCC. So, we decided to chose a percentage of the jobs from the rest of the mix and annotate them as high priority jobs.

The problem however is, how do you pick the numbers? The solution we propose is to choose 10%, 20%, 30% or 40% of the jobs as high priority jobs and then conduct a sensitivity analysis on our assignment. We want to note here that, these estimates has been reviewed and approved by the peers in our papers [60, 94–96]

4.4 Insert Additional Load

Finally, the introduction of resource federation will provide more resources to the researchers of each institution both in terms of quantity and diversity. That means users from different institutions will not only have access to a larger pool of resources but also have access to resource types that may

not have been currently available at their own institution. Moreover, as CCC will employ a single M/M/K queue in place of k M/M/1 queues, we would typically expect shorter delays for the jobs. From our discussion with several stakeholders [54, 66], we found out that a very long wait-time for the jobs would often lead researchers not to use the resources as much as they would have liked or in response buy their own clusters.

However, with CCC, because of the larger resource pool we expect a much quicker turnover (shorter wait time) for jobs. To add to that, the diversity of resources will introduce new opportunities for researchers to explore more with additional jobs. Hence, we expect more load from the institutions (specially from the ones where the resources are generally fully-subscribed or higher queue delays discourages the scientists to explore with more options). The problem here is, we do not have a way to predict these additional loads until the CCC is functional.

To deal with that, we decided to increase the loads from the institutions with some additional jobs. However, creating synthetic load completely out of whim can compromise the credence of the simulation. Hence, rather than creating completely random load, we decided to randomly duplicate a percentage of the existing jobs and insert them in the system to increase the overall load. For example, if we want to increase the load by 5%, we will randomly chose one out of each 20 jobs to duplicate. This way we can increase the load by 5%, 10%, 15%, 20% and more to conduct another sensitivity analysis.

Now the final peace of the puzzle is to identify how much additional load would we expect with CCC. Again, we can not predict it perfectly until we have real users submitting their jobs to CCC, but we can obviously come up with some educated guess. As we mentioned before, the average wait time is one of the major limiting factors for researchers to add more jobs into the mix. With CCC, the average wait time is expected to reduce. Hence, we can concoct a heuristics that-

Heuristics 4.4.1. *Users would submit additional jobs as long as the average wait times for jobs does not exceed the average wait time of the current job mix.*

So, we would increase the load at different institutions until the wait time with CCC exceeds the wait time for the current job mix. From our analysis, we found that we can add 5% of additional load at IU and 25% of additional load at VT without increasing the wait times at either of the institutes. We have discussed these results in more details in Section 6.5.

5

SIMULATION SETUP

In the CCC we plan to leverage the current cyber-infrastructure to build a market based resource federation for academic computing. We believe with a market based federation will yield a better net value for all the participating institutions individually and combined. To test this hypothesis, we choose simulation to compare the institutional values for each institution as a separate entity with the institutional value accrued by the same institution through CCC. We setup our simula-

tion with multiple institutions and multiple differentiated QoS. We used an off-the-shelf simulator called ALEA [74] to simulate the job traces collected and calibrated as mentioned in Chapter 4 to conduct our simulation. The simulator comes with a built in implementation of PBS-PRO [90] scheduler and we leverage that to develop the CCC scheduler for our simulation. In this chapter we discuss the simulation model and the scheduling algorithms for the simulation, followed by a brief introduction on the model definitions and parameters, and finally we conclude the chapter with a evaluation plan to test our hypothesis.

5.1 Simulation Model

The goal of our simulation is to evaluate the CCC, which is a market based federation among multiple institutions. We setup the simulation based on the system model described in Chapter 3. For our simulation we use the data traces available from three (3) different institutions. Each institution has a set of resources with different resource properties and a set of queues that we can submit our jobs to. We do not control the resources of the institutions. They are managed by their own system administrators.

The CCC has three (3) queues for three different quality of Service (refer to the Figure 3.3.1). Users will submit their jobs in one of these three queues and the CCC-scheduler will schedule the job to the appropriate resource.

5.2 Scheduling

We used ALEA v4.0 to conduct our simulation. ALEA is based on the GridSim [38] simulation toolkit which was extended to provide a simulation environment that supports simulation of vary-

ing job scheduling problems. ALEA has a centralized job scheduler which uses advanced scheduling techniques for schedule generation. A few local search-based optimization algorithms as well as classical queue-based policies such as FCFS, SJF or Easy Backfilling [48] are already supported in ALEA. The scheduler is capable of handling dynamic situations when jobs appear in the system during simulation. In this case the generated schedule is changing through time as some jobs are already finished while the new ones are arriving.

For our simulation, we started with the PBS-PRO scheduling algorithm available in ALEA and modified that to implement the CCC scheduler. Before diving into the CCC-scheduler let us first descuss the PBS-PRO scheduler that was readily available with ALEA.

5.2.1 PBS-PRO

In ALEA, scheduling algorithms in general include two functions, i) adding a new job to the queue, ii) scheduling the next job. Algorithm 1 shows the pseudo code of these two functions for PBS-PRO workload manager.

Adding a new job

In a typical PBS-PRO workload manager (similar for SLURM too) there would be several different scheduling queues for users to submit the jobs. Each queue will have access to different set of resources and sometimes may have different priorities on those resources. For example, there would be parallel queue for parallel jobs, serial queues for serial jobs, largemem queue, GPU queue etc. In order to schedule a job in the PBS-PRO queuing system (which represents deployment of current cyber-infrastructure), the scheduler just has to put the job on the specific queue. However, since we are expecting the jobs from the queues for specific research labs and the debug queues to

Algorithm 1: Scheduler for simple queue based workload manager (PBS/SLURM)

```
Function add_new_job(job_info: JobInformation) : void is
    Result: The job is added to one of the scheduling queues;
    /* Each institute has a several different scheduling queues   */
    if job_info.get_job_id() is a private job or debug job then
        move job_info.get_job_id() to the HighPriorityQueue;
    else
        add the job to the requested queue;
    end
    /* Different queues have access to different set of resources */
    return;
end
Function select_job(): int is
    Result: If resource is available which can fit a new job, the job gets scheduled;
    Resource_List: List of the available resources;
    Queue_List: List of Queues sorted in the order of priority;
    foreach queue ∈ Queue_List do
        while queue is not empty do
            job_id ← job at the top of the queue;
            sort(Resource_List, Resource_Cost);
            foreach ResourceID res_i ∈ Resource_List do
                if matches(res_i, job_id) and res_i.can_run_now(job_id) then
                    remove job_id from the queue;
                    submit_job(res_i, job_id);
                    return job_id
                end
            end
        end
    end
end
```

be high priority, we have added a HighPriorityQueue on the standard PBS-PRO implementation and put all the high priority jobs there. Jobs from this queue are always scheduled as soon as a new resource become available.

Scheduling the next job

When a new scheduling event is triggered, the scheduler first goes through the HighPriorityQueue and selects a job from the head of that queue. Then it iterates through the available resources (*Resource_List*) in an ascending order of costs (so that expensive resources are not used if they are not necessary) and checks if the resource matches the job requirement and if the resource have enough capacity left to run the job at that point. If these conditions suffice, the job will be scheduled on that resource. However, if no such resource is available at this moment, the scheduler fetches the next job in the queue and tries to get that scheduled. The non-priority jobs will be scheduled if there are no high-priority jobs available on the queue that can run on the available resources currently.

5.2.2 CCC-Scheduler

For the CCC we simulate a federated queue of queues priority scheduler in which jobs are placed into one of three global queues, high priority, regular priority, or low priority. We modified the PBSPro algorithm to schedule the jobs in multiple differentiated quality of service queues. The resource pool consists of all the resources from different institutions that have been made available. Each resource is annotated with a set of resource properties which specifies the special features of the resource (GPU, largemem, KNL, inifi-Band), and the institution it belongs to. The per unit cost of a resource is determined based on these properties. The idea is to schedule the job at a resource which has the minimum unit cost and fulfills the resource requirements of the job.

Again we implemented the two scheduling functions for adding a new job to the available queues and scheduling the next job on the available resources.

Algorithm 2: Scheduler for multiple qos based CCC scheduler - Part I

```
Function add_new_job(job_info: JobInformation) : void is
    Result: The job is added to one of the scheduling queues;
    /* In CCC there will one queue for each quality of service    */
    if job_info.get_priority() == "HIGH" then
        add job_info.get_job_id() to the HighPriorityQueue;
    else
        if job_info.get_priority() == "MEDIUM" then
            add job_info.get_job_id() to the MediumPriorityQueue;
        else
            add job_info.get_job_id() to the LowPriorityQueue;
        end
    end
    /* All the three queues have access to all set of resources
       unless otherwise specified                                 */
    return;
end
```

Add a new job

For the CCC-scheduler, we have three (3) global queues, one for each priorities of service. To add a new job for a CCC-scheduler is straight forward (Algorithm 2). In the calibrated job traces, each job is annotated with the a job priority and the scheduler just matches the jobs to the appropriate priority queue.

Scheduling the next job

Scheduling the next job to run is more complicated. To select the next job the scheduler traverses the HighPriorityQueue first in a FCFS manner and checks if the job is schedulable at that moment. If none of the high-priority jobs are schedulable, the scheduler selects the next job from the RegularPriorityQueue, and finally from the LowPriorityQueue. Within each priority queue, jobs are

Algorithm 3: Scheduler for multiple qos based CCC scheduler - Part II

```
Function select_job(): int is
    Result: If resource is available which can fit a new job, the job gets scheduled;
    Resource_List: List of the available resources;
    Queue_List: List of Queues sorted in the order of priority;
    foreach queue ∈ Queue_List do
        while queue is not empty do
            job_id ← job at the top of the queue;
            sort(Resource_List, Resource_Cost, job_institution) ;      // sort first by
             cost then by job institution
            foreach ResourceID res_i ∈ Resource_List do
                if matches(res_i, job_id) and res_i.can_run_now(job_id) then
                    remove job_id from the queue;
                    submit_job(res_i, job_id);
                    return job_id
                end
            end
            if job_info.get_priority() == "HIGH" and !job_info.isExpired() then
                foreach ResourceID res_i ∈ Resource_List do
                    if matches(res_i, job_id) then
                        /* check if the job can be scheduled by removing
                           one or more low priority jobs                  */
                        if can_run_after_preemption(res_i,job_id) then
                            while low priority job(s) running on res_i do
                                preemp_low(res_i) ;          // preempt the most
                                 recently started low priority job
                                if res_i.can_run_now(job_id) then
                                    submit_job(res_i, job_id);
                                    return job_id
                                end
                            end
                        end
                    end
                end
            end
        end
    end
end
```

selected for scheduling in a FCFS fashion. To improve the scheduler efficiency, whenever a job arrives on the queue we decided to calculate a candidate set of resources for the job on which the job can run. This list will act as a cache for the job for subsequent cases as the scheduler may need to check the schedulability several times before it can finally schedule the job.

However, schedulability in CCC is different than that is in PBS-PRO. Specially for the high priority jobs as we need to pay attention to some special details:

1. In CCC, we support the preemption of low priority jobs to schedule a high priority job sooner. The question is which low priority job to be preempted first?

2. The federation in CCC allows us to move the the jobs from one campus to another if need be. However, the scheduler has to make sure that no unnecessary movement of the job is happening as there are some overheads for moving job around from one place to another (i.e.- data movement).

3. The high priority jobs come with a defined deadline, the question is if we can not meet the deadline for a high priority job, do we still preempt low priority jobs to get that job scheduled sooner?

Algorithm 3 provides our answer to these questions.

1. There is a literature regarding preemption approaches [71] [46]. For our project we did not dive into the details of the preemption approach as it is somewhat orthogonal to the goal of the project. We choose to preempt a low priority job, we select the one that started most recently. This way the overhead for preemption would be minimum.

2. To schedule a job we want to select the cheapest available resource that is able to run the job. However, there are similar resources in different institutions that have the same resource

cost. So just selecting based on cost may in fact introduce some unnecessary movement of jobs. To mitigate that, we add a secondary parameter to sort the resources- the job institution. Which would make sure if multiple resources are available for a job with the same cost, then the job is scheduled to the resource that belongs to the same institution as the job submitter.

3. If the deadline of a high priority job is expired, we simply treat it as a medium priority job. That is it can not be preempted once it is scheduled and it can not preempt any other job. Further it is charged and valued as a medium priority job.

5.3 Model Definitions

Before diving into the results let us introduce some definitions and metrics-

Definition 5.3.1 (Resource Pricing)**.** For resource pricing, we use a basic node configuration and assign a base price to the configuration. Then, we put additional charges for each additional feature, e.g. GPU, added to the base configuration. Resource pricing is a major factor in job value, so prices should reflect prices in the real world. We follow the Amazon AWS on-Demand pricing scheme.

Table 5.3.1 summarizes the reference instance chosen from AWS, its price and ratio and our multiplier chosen from the ratio. For our experiment, we chose a multiplication factor of 5 for GPU resources as in the CCC the GPU resources have more cores per machine. The large memory machines in the CCC have a factor of 3. Since there was no reference to valuate infini-band interconnect, we chose a factor of 2 for that. For special features like mic or KNL, we chose a multiplication factor of 3, as they are not as powerful as a GPU, however they provided extra co-processors for faster execution.

Table 5.3.1: Resource Pricing based on Amazon AWS on-Demand Pricing

Function	Ref Instance	Cores	Mem. (GB)	GPUs	Price/ core-hr	Ratio	Multi-plier
General Purpose	m5.4xlarge	16	64	0	0.0480	-	1
GPU	p2.xlarge	4	61	1	0.2250	4.68	5
Large Memory	x1.32xlarge	128	1952	0	0.1042	2.17	3

Definition 5.3.2 (job Price)**.** Job price is defined as the price that the user must to pay[1] for executing a job on the market resources. The price of a job depends on the requested resource type, and execution time on the resources.

$$job\_price = req\_res\_cost \times exec\_time \times priority\_factor \tag{5.1}$$

$$priority\_factor = \begin{cases} 2, & \text{for high priority} \\ 1, & \text{for medium priority} \\ \frac{1}{2}, & \text{for low priority} \end{cases} \tag{5.2}$$

For the experiment, we chose a *priority_factor* of 2, 1, and $\frac{1}{2}$ for high, medium and low priority jobs respectively

Definition 5.3.3 (Job Value)**.** Job value is defined as the value the user gets by running a job. Job price is a lower bound on the value that the researcher gets as the researcher is at least willing to pay *job price* for running the job. (The surplus value could be higher if the PI would have been willing to pay more.)

Definition 5.3.4 (Institutional Value)**.** Institutional value (*ins_val*) is the sum of the values of all

[1] Whether users pay or not is up to their institution. The users' institution is responsible for paying in kind or paying with cash.

the jobs run through the shared resources. Formally,

$$ins\_val = \sum_{all\_jobs} job\_value \geq \sum_{all\_jobs} job\_price. \tag{5.3}$$

However, by definition, the high priority jobs will yield the additional value (through priority factor) value to the researcher only if it starts immediately (within a threshold). For our experiment we choose the the threshold as 10 mins. Equation 5.4 defines the deadline of the *ith* job.

$$deadline_{job_i} = \begin{cases} 10 * 60 + exec\_time_{job_i}, & \text{if } job_i \text{ is a high priority job} \\ \infty, & \text{otherwise} \end{cases} \tag{5.4}$$

Hence the *job_value* and *ins_val* can be formally presented by equation 5.5 and 5.6 respectively

$$job\_value = \begin{cases} job\_price, & \text{if } (Priority \neq high) \text{ OR} \\ & (Priority = high \text{ AND } finish \leq deadline) \\ job\_price\_medium\_priority, & otherwise \end{cases} \tag{5.5}$$

$$ins\_val = \sum_{job_i \in all\_jobs} job\_value_{job_i} \tag{5.6}$$

At the beginning of the project we considered job prices and values in allocation units [94]. However, as allocation units are an abstract value, it is often hard to relate with the real world monetary value and often difficult to conceptualize. Hence, we decided to measure prices and values in dollars rather than allocation units. While we wanted to try different dollar units for resource cost and job values and conduct a sensitivity analysis, the members of CCC wanted to compare the prices and values based on Amazon pricing, as Amazon has been the market leader of commercial

clouds for over a decade now.

Thus, we used AWS pricing as a reference to convert *ins_val* into dollar value. Since the AWS does not provide the same QoS breakdown as CCC, and AWS on-Demand execution would be comparable to a high priority execution of a job, we chose the value for a high priority job with no special features to be 4.8 cents-per-core-hr. So, the value for a general-purpose medium or low priority job would be 2.4 cents and 1.2 cents per-core-hr respectively. One thing to note here is that, we are ignoring the storage cost and data transfer cost of Amazon AWS for our calculations. Hence, we are actually under-valuing the jobs slightly. From here on we will use the dollar value as the institutional value for clarity.

Definition 5.3.5 (Outsourcing Cost)**.** Since the market in CCC is designed based on resource trading, we need a way to measure the cost to run jobs on a resource owned by a different institution. We define the cost of running a job remotely as the value the researcher will get by running that job. So, if a researcher from institution A runs a job at institution B with value X, then institution A will owe X dollars to institution B. On the other hand, the institution A can earn by allowing researchers from other institution to run their jobs on resources at institution A. So the total outsourcing cost for institution A is the difference between the cost of jobs run remotely by the researchers at institution A and the amount earned by an institution A by lending their resources to other institutions. Formally,

$$cost\_outsource_A = \sum_{jobs-remotely-run-by-A} job\_value - \sum_{jobs-remotely-run-at-A} job\_value \qquad (5.7)$$

Definition 5.3.6 (*wait_time*)**.** The *wait_time* of a job can be defined as the difference between the *queue_time*, that is the time when the job was placed in the queue and the *start_time*, that is the

time when the job started execution. So, the average wait time can be expressed by the following Equation 5.8

$$avg\_wait\_time = \frac{\sum_{job_i \in all\_jobs}(start\_time_{job_i} - queue\_time_{job_i})}{num\_of\_jobs} \tag{5.8}$$

Definition 5.3.7 (The percentage of high priority jobs scheduled within the defined threshold)**.** The percentage of high priority jobs scheduled within the defined threshold (%high) is used to evaluate the on-demand access facility for the high priority jobs and can be expressed with the Equation 5.9.

$$\%high = \frac{\text{number of high priority jobs finished within deadline}}{\text{total number of high priority jobs}} \times 100\% \tag{5.9}$$

5.4 Evaluation Plan

Table 5.4.1: Different alternative approaches to evaluate the performance benefits of CG-Market based on Differentiated QoS and Federation of resources. Our current system is executing model 1 and CCC implements model 3

	Federation	**Differentiated Qos**
Baseline		
with Qos		✓
with Federation	✓	
CCC	✓	✓

Our hypothesis for CCC is- (a) CCC will lead to a better accumulated value from the organization's point of view compared to the value accumulated by the organizations separately with typical FCFS or Simple Priority based scheduling techniques, and (b) More high priority jobs will

be started within the predefined deadline with CCC compared to FCFS or Simple Priority. We can formalize them as two null hypotheses-

- Institutional value from computation and storage does not increase when participating in a market-based federated computational economy with differentiated QoS.
- Number of 'run immediately' jobs started within the deadline does not increase in a market-based federated computational economy with differentiated QoS.

Hence, the goal of the experiment is to reject these two null hypotheses while controlling for campus resource changes over time. As we are interested in measuring the performance improvements due to the federation of resources and the differentiated QoS, we will compare the performance of four systems by turning these two features on and off. Table 5.4.1 identifies the four (4) system models that we are going to compare for our experiment.

5.4.1 Baseline

This is the system model that is currently implemented in all the test sites. In this set up, jobs are not categorized into mutliple differentiated Qos and for simulation we treat all the jobs as medium priority jobs (except for the debug jobs and jobs from special labs which are treated as higher priority and always moved to the head of the queue). Also in this model jobs originated from an institution are always executed at that same institution.

5.4.2 with Qos

In this approach we want to verify the performance improvements achieved by using differentiated QoS in each of the participating sites. So, in this setup, all the jobs will be scheduled on their original

sites. However, jobs will be categorized into multiple QoS groups (3 for our current simulation). Highest priority jobs will be scheduled as soon as possible, some low priority jobs can be preempted if needed to ensure that the high priority jobs are scheduled within the predefined threashold.

5.4.3 with Federation

This setup is used to quantify the benefits of the federation. In this model, jobs are be submitted to single large shared queue and scheduled to resources at any of the institutions based on resource specification for the job. All the jobs except for the debug jobs and jobs from special labs (who own their condo nodes) are treated as medium priority.

5.4.4 CCC - QoS & Federation

This is the standard CCC set-up. In this setup, jobs will be submitted to three (3) shared QoS queues (one for each quality of service) and can be scheduled to resources at any of three institutions. This setup shows the total benefit of using CCC.

We also measure the balances for each institution due to the federation (as institutions will borrow cycles from each other).

6

RESULTS & ANALYSIS

In this chapter we will present the results from running the simulation models described in Chapter 5 using the data described in Chapter 4. The goal is to test the hypothesis presented at the beginning of Section 5.4.

6.1 Metrics

CCC combines three (3) basic ideas into a production grid environment- i) Differentiated QoS, ii) Resource Federation, and iii)Resource Trading. The goal here is to analyze the benefits of each of these ideas individually and separately. We begin by introducing all the metrics of interest.

Institutional Value Since we are evaluating a grid economy concept, it is paramount that we measure the added values each institution is gaining with CCC. Hence, we measure institutional value as the value the researchers are willing to pay to run their jobs. Institutional value as defined in 5.3.4 gives us a high level picture of the benefits that can be gained with different features of CCC. Note, A job is worth at least what the PI is willing to pay.

On-Demand Access From our discussion with several researchers who own the condo nodes in institution data centers, we determined, one of the main reason funded researchers want to buy their own condo node is because they want their jobs to be run immediately rather than waiting in the shared queue. Which means, getting their jobs scheduled on-Demand is an important metric for them. Hence, we wanted to measure what percentage of high priority jobs are scheduled right away (within 10 mins for our experiment).

Wait-time Another metric that we are going to use is the time users have to wait before their job gets scheduled on a resource. It is expected that with federation the average wait time will decrease significantly. And since people seem to be content with the wait-times for the baseline approach (Section 5.4.1), we believe we can allow additional loads in the system until the wait-times for CCC exceeds that of the baseline approach.

Utilization Resource utilization is another important metric that we are going to measure as it

is important for institutions to have a reasonable utilization for their resources. This is one metric that the system administrators care about. We understand that the preemption of low priority jobs will have a negative effect on the utilization for CCC and hence we wanted to quantify this metric.

Additionally, in order to evaluate the effects of the resource trading, we measure the costs each institution incurs to outsource their jobs to other institutions. In other words, provide a net value measure for each of the CCC participants.

In the next three sections, we are going to break-down the results in terms of value gained, cost incurred and net value with different simulation set-ups. All the values are represented as a delta from the institutional value for the baseline approach described in Section 5.4.1.

6.2 Value Added

At first we want to focus on how much individual institutions are expected to gain (or lose) through different features of CCC. Figure 6.2.1 represents the value for each institution compared to the value of the Baseline approach (as delta) for 3 months (October 2017 to December 2017) in different QoS distributions. These results does not include any payments, that is these values reflect on the effects of sharing among the institutions in case of federation (not trading).

From the figure we observe that both IU and VT gain significant value with differentiated QoS even if they are running this feature without federation. As we increase the percentage of high priority jobs in the mix, the value gains increase for both the institutes as expected. This is because, as more high priority jobs are inserted in the mix, more often the resources will be occupied by high priority jobs (yielding more value). For IU jobs the gained value increases from about $170,000 to

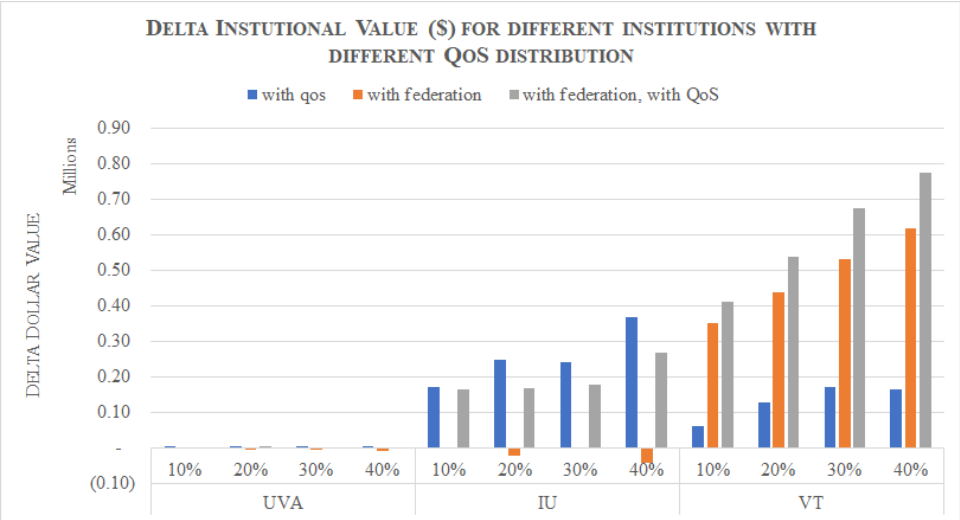

Figure 6.2.1: Dollar value of all the completed jobs as a delta from Baseline approach in 3 months (Oct 2017 to Dec 2017) for different institutions based on different QoS distribution. The x axis represents the percentage of high priority jobs in the mix for the given institution and the y axis represents the dollar value gained through different alternatives.

$368,000 with differentiated QoS only, on the other hand, gains for VT increases from $59,000 to $165,000. With differentiated QoS, gains for IU are significantly higher than that of VT, because with about 75% utilization IU has more opportunity to schedule high priority jobs as soon as possible. Also, IU has more low priority jobs compared to VT, which can be preempted to run high priority jobs immediately. Also we observe that, with about 40% of high priority jobs the gains for VT stops increasing. We believe this is because most of the resources are saturated with high or medium priority jobs at VT, so adding more high priority jobs means more jobs will fail to fulfill the deadline. For UVA however, almost all the jobs are scheduled right away, because of it's very low utilization. Hence, we hardly see any gains for UVA with differentiated QoS.

With federation however, only VT gains value, both UVA and VT lose some value with federation only approach. This is because VT runs at about 90% utilization as itself, while the utilization of UVA and IU resources are 15% and 75% respectively. Since with federation the loads are sched-

uled in a more balanced way, many VT jobs get scheduled at IU and UVA, allowing more jobs to finish before the deadline. IU and UVA, on the other hand, lose some value as some of their cycles are then occupied by the jobs from VT and hence some of the high priority jobs at UVA and IU start missing their deadline. For VT, as the percentage of high priority jobs in the mix increases, the value gained with federation increases significantly (from about $350,000 to $610,000 as we go from 10% to 40%). With federation, the loss for UVA in terms of value is very small, less than $10,000. In the case of IU, the loss of value increases slightly as the high priority job increases. The value loss for IU increases from $500 to $41,000 as we increase from 10% to 40% of high priority jobs. This is expected as more high priority jobs are in the mix, more of the VT jobs will be outsourced to IU and VT, which would not be preemptable when new high priority jobs arrive at IU or UVA. Hence, the loss for both UVA and IU increases as the gain for VT increases.

Finally with the CCC setup, both IU and VT gain significant value over the baseline. As expected, the gains for IU are mainly due to differentiated QoS. However, it is not as high as it was with only differentiated QoS as with the federation in CCC some of VT jobs causes a few of IU high priority jobs to miss their deadline. So as the percentage of high priority jobs increases from 10 to 40, the value gain increases from about $160,000 to $265,000. For VT, on the other hand CCC yeilds the most benefit in value. As VT was gaining through both differentiated QoS and federation, the value gain for CCC with 40% of high priority jobs reaches to about $775,000. Because of its very low utilization, like the previous two cases the impact of CCC on UVA in terms of value is very minimum. With CCC, the delta for UVA is less than $1,500 for 3 months for any of the job mix.

6.3 Cost Incurred

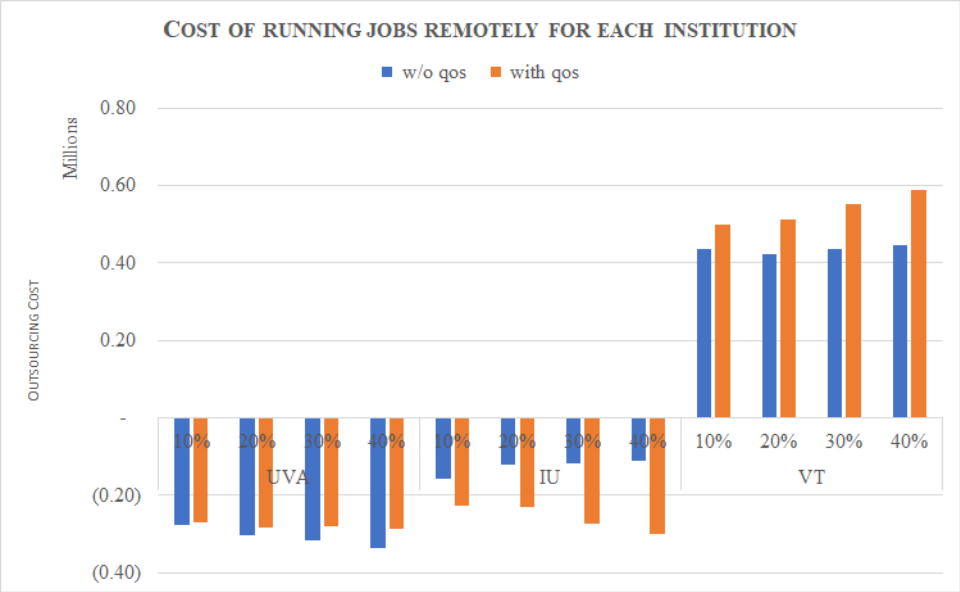

Figure 6.3.1: Cost for different institutions to run their jobs at other institutions over the 3 months (Oct 2017 to Dec 2017) of simulation period. Here negative cost means, other institutions used more cycles from the institution compared to what the institution has borrowed from others.

Figure 6.3.1 summarized the outsourcing cost (Def: 5.3.5) for different institutions with the federation. From the figure we can observe that, both IU and UVA are actually earning from the federation by allowing VT researchers to run their job on IU or UVA (here negative cost means payments received). VT on the other hand are the one with the most debt from the federation. This is in general very reasonable as among the 3 institutions, VT has the most existing resource utilization (about 93%). Although UVA has the least amount of resources as those resources are only 15% utilized, both IU and VT can use those unused cycles to run their jobs. Hence UVA earns the most from the federation both with or without differentiated QoS.

Without differentiated QoS, UVA earns significantly more than IU from the federation. Also, the earnings for UVA increase as the percentage of high priority jobs increases in the job mix (from about $275,000 to $335,000). This is mainly because of the very low utilization at UVA. Jobs from IU and VT can get scheduled at UVA most of the time. For IU, the earnings with federation without QoS is very limited and decreases as the high priority jobs increases in the mix. We believe, this can be explained by the fact that even though the utilization of resources at IU is about 75%, the average wait time for many of the jobs (particularly jobs that need special features, i.e. gpu) is very high. That means these resources are in demand, and so the scope for scheduling jobs from VT or UVA are mostly limited to regular jobs. Because of the very high load at VT, many of their jobs actually get outsourced to IU and UVA and hence the cost for outsourcing is very high for VT. And without differentiated QoS the cost for outsourcing is almost invariant to the percentage of high priority jobs available in the mix.

With differentiated QoS the earnings for UVA reduces a little compared to the non-differentiated QoS case. This is because with differentiated QoS, high priority jobs can preempt low priority jobs to get scheduled right away, and as IU has a much larger resource pool, it has much more chance to host a high priority job (by preempting low priority jobs if needed). As a result the earning for IU with differentiated QoS is significantly higher compared to that of without QoS approach. For the same reason, as the number of high priority job increases in the system, IU earns even more. The earnings IU reaches from $220,000 to $300,000 with differentiated QoS, as the percentage of high priority job increases from 10% to 40%. Similarly, for VT, as the high priority jobs increase in the system, the amount they owe to the federation increases as more high priority jobs get outsourced to fulfill their deadline.

6.4 Net Value

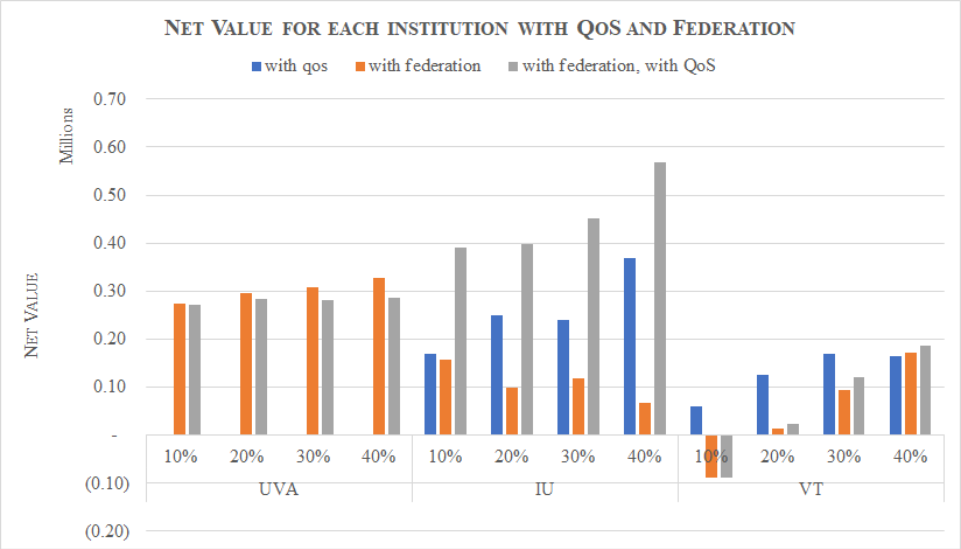

Figure 6.4.1: Net value for different institutions over the 3 months (Oct 2017 to Dec 2017) of simulation period with different simulation setup. Here negative net value means, the institution has to pay more to the federation (as they outsourced their jobs to other institutions) than the value they earned through it.

From Figure 6.2.1 and 6.3.1 we can gather that UVA and IU can earn quite a handsome amount from the federation, which accounts for the small loss they suffer in value. VT, on the other hand, has to pay about half a million dollars to the federation per quarter. Which as itself is such a hefty amount that even with all the gain on institutional value, it will be very hard to make a case for VT to join the federation. The obvious next step would be to compare the value gained with the cost incurred by the institutions for all the setup. Hence we define the net value (Equation 6.1) for an institution as the difference between the gained value over the baseline and the cost incurred to run

one's jobs remotely. Figure 6.4.1 presents the net value for different institutions.

$$net\_value_A = value_{gained_A} - cost\_outsourcing_A \tag{6.1}$$

For the case of differentiated QoS only, since there is no federation (that is each institution is running only on their own resources), the net value will be equal to the value gained. So, as explained in Section 6.2, the net value for UVA would be negligible, however, IU and VT realize significant net value gains through occupying their resources with higher priority jobs.

For the federation only case, UVA gets the most benefit out of the federation as it can trade its unused resource to the other members of the federation. IU also earns some net return from the system under federation only, but that is significantly lower than what IU earns with QoS only approach. The net value for VT, on the other hand is comparatively lower, as many of their jobs are scheduled to other locations. This means even though VT researchers get more value out of the federation, those values are negated by the cost VT has to bear to outsource their jobs. In fact, from Figure 6.4.1 we can observe that 10% of high priority jobs in the mix, VT suffers a loss of about $87,000 with federation. However, as the high priority jobs in the system increases, VT starts to gain some value and with 40% high priority jobs VT can gain a net value of about $170,000.

For the full CCC setup, IU tends to get the most value for similar reasons explained in Section 6.3. UVA also gains significant value (about $280,000) from the CCC federation. The net value for VT is again much lower because of the high cost they incur. However, as more high priority jobs are inserted in the mix the net value margins for VT gets bigger.

From the Figure 6.4.1, the benefits for IU and UVA with CCC are very clear. However, for VT it seemed their accrue more net value with QoS only approach compared to the federation only or CCC approach (specially when the percentage of high priority jobs in the mix is not very high).

Which begs the question, "what is the benefit for VT to join in the collaboration?" The answer lies in the fact that with federation a lot of VT jobs are outsourced to IU or UVA. Which reduces the load at VT resources causing the utilization to go down to about 70% from 90% with lower wait times. That means the VT researchers now have more capacity to introduce additional jobs in the system. Hence, we wanted to simulate the effects of additional load on the system and analyze the net value for different institutions with the additional resource setup.

6.5 Additional Load

Before adding additional load, we have to answer two very important questions.

1. How much additional load to be added?
2. How to add additional load to the system?

To answer the first question, we want to make sure users are submitting all they need and so, focus on the factors that limits the number of jobs users submit. We believe the major limiting factors here are: long delay before the job starts, and availability of special resources.

Long wait time for jobs In general, users tend to submit fewer jobs if they have to wait for a long time for their jobs to be scheduled. For example, if a user has to wait for a day to get his/her job scheduled in a GPU resource, He/she will not be less inclined to add new GPU jobs to the system.

Availability of special resources Most often, the users are limited by the resources available to their own institution or the resources they have access to within the institution. For example, BigRedII at IU does not have any large memory nodes, so users of BigRedII will not submit a large memory job in general.

For UVA the average load of the system is only 25% and the average wait time is less than a minute. Which means UVA researchers already have a lot of unused resources which they are not using, so it would hardly make sense to introduce additional load at UVA.

On the other hand, the average system utilization for VT and IU are about 94% and 73% with a very high average wait-time (about 506 mins and 246 mins respectively for the baseline approach) and IU logs do not contain large-memory data. Hence these two institutions are potential candidates for adding additional loads.

With the federation, however, the average wait-time for VT and IU reduces to about 163 mins and 150 mins respectively. As the wait time with the federation is significantly lower and researchers are currently fine with the current wait time, we will insert loads at these two institutions until the wait time at either of the institutions goes over the wait time with baseline approach. In other words, add load until the delays are comparable. We have to keep in mind that increasing load at any one institution can actually lead to a increase in the wait time for all the institutions due to the federation. So, we keep track of the wait time for both the campuses whenever we change loads at either of the campus.

Table 6.5.1: Effects of adding additional loads at IU and VT on the average wait times on these institutions. Scenarios where the wait time exceeds the baseline wait times are struck-out and are not acceptable. The best acceptable scenario combination is 5% additional load at IU and 25% additional load at VT

		additional load						
IU	baseline	0%	10%	5%	5%	5%	***5%***	5%
VT	baseline	0%	10%	5%	10%	20%	***25%***	33%
awt IU (mins)	246.2	163.6	~~336.1~~	176.2	194.8	216.1	***240.3***	~~260.7~~
awt VT (mins)	506.0	149.4	197.2	154.5	170.4	195.5	***237.8***	263.1

We have used a trial and error approach to find a good solution. So the goal is to find an com-

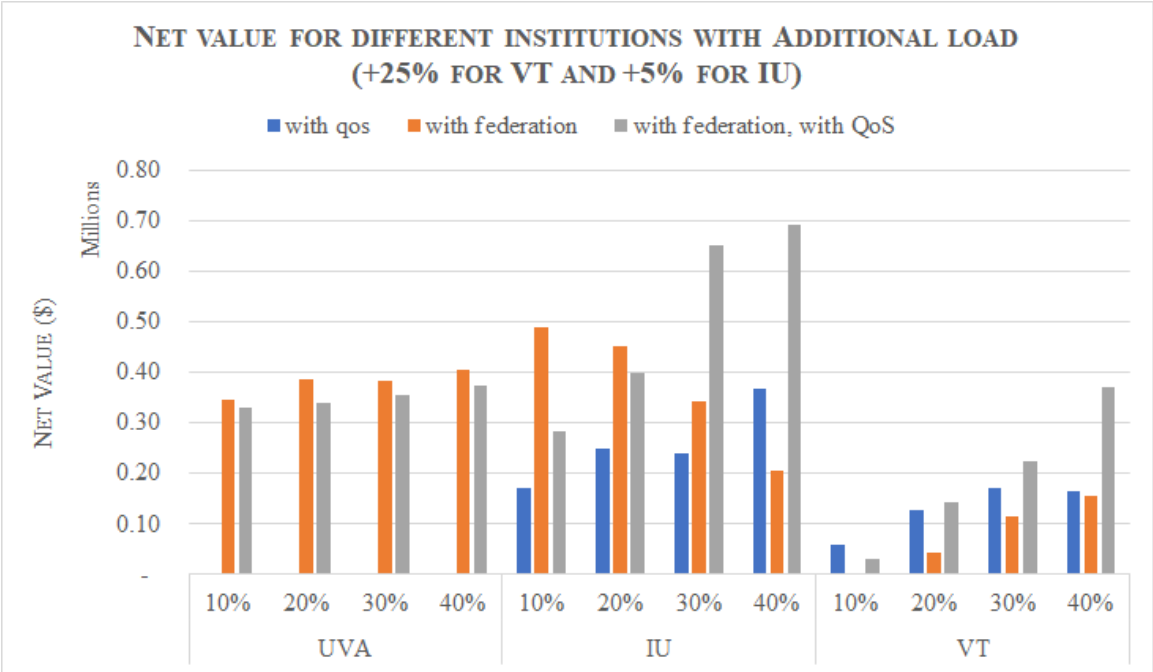

Figure 6.5.1: Net value for different institutions over the 3 months (Oct 2017 to Dec 2017) with additional hypothetical loads.

bination of additional load where the wait time for both institutions will be below their baseline value. So we first tried a 10% increase on both institutions. However, the wait time for IU with 10% additional load exceeds that of it's baseline. So in the next attempt we tried 5% of additional load at each institution. Which yielded favourable wait time for both institutions. From these two attempts we figured that if we add any more additional load at IU the wait time would exceed the baseline value. Hence we tried more loads at VT and found that if we add an additional load of 33% at VT, the wait time at IU exceeds the baseline limit. So from Table 6.5.1 the best acceptable combination for additional load at IU and VT is 5% and 25% respectively.

Now that, we have identified the best possible additional load scenarios for IU and VT, we can simulate the generated net value with the additional load. Figure 6.5.1 presents the net value for

each of the institution in CCC with additional loads. Comparing these results to net value without additional load (Figure 6.4.1), we found that the net value for VT moves into the positive with federation only approach even with lower percentage of high priority jobs in the mix. This is due to the fact that IU is the largest of the three in terms of available resources. Also the utilization of IU being about 75% allows both its additional loads to get values by running some more of the VT loads.

For the CCC case, in Figure 6.4.1, we found that VT was losing some value when there was only 10% of high priority jobs in the mix. But with additional loads VT has positive net value for all the cases. And as expected the positive net value is increasing as the percentage of high priority jobs increases with CCC.

Finally with CCC, the net value margin for UVA is in the range of $330k to $370k, for IU is in the range of $285k to $690k and for VT is in between $31k to $370k for the 3 months of simulation period. Which yield a combined benefit of about $350k per month on average.

6.6 On Demand Access Capability

On demand access (ODA) in one of the more important metrics that many of the stakeholders care about. Figure 6.6.1 presents the on demand access capability with different approaches by varying the priorities in the job mix. From the results we can observe that approaches with QoS achieve better performance with respect to ODA. This is what we would expect as these approaches preempt low priority jobs to schedule high priority jobs as soon as possible. When we add the federation component of CCC with QoS, the ODA improves significantly for VT from 66% to 97% on average. As VT runs at around 94% utilization, many of the high priority jobs would miss

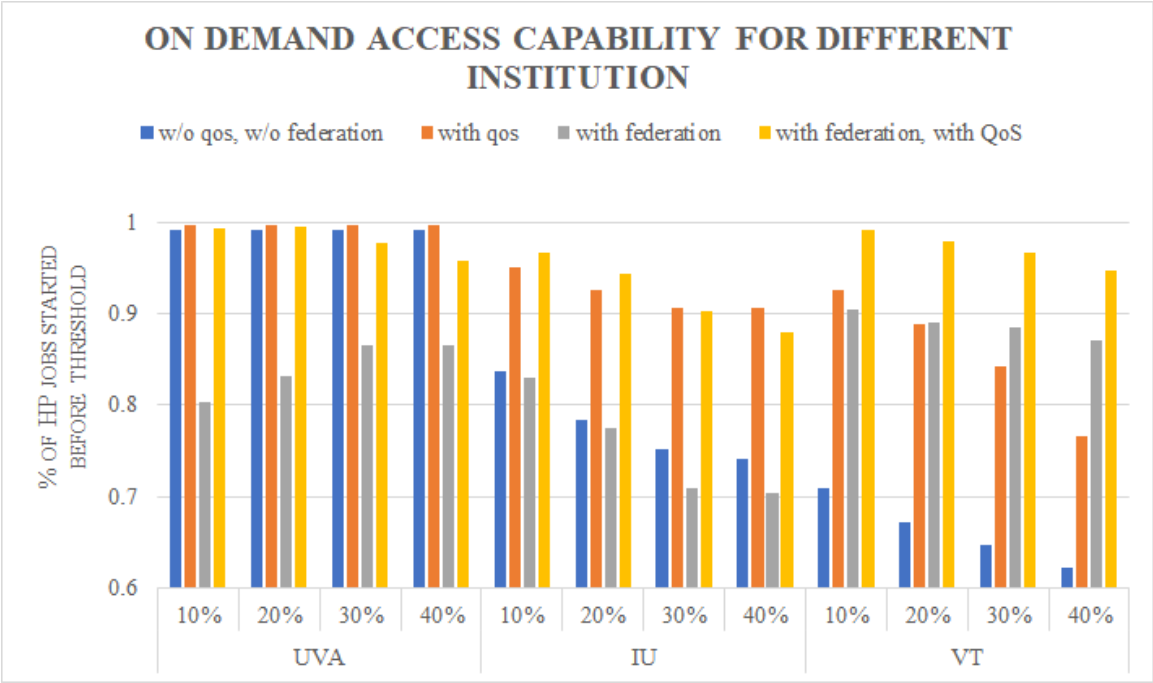

Figure 6.6.1: Percentage of high priority jobs started within the predefined threshold for all three (3) institutions, under different priority mix. The figure shows that in most cases the CCC provides the best on demand access capability with more than 90% high priority jobs scheduled within 10 mins.

their deadline without QoS. From the result, we can also infer that, with federation, many of the VT high priority jobs get scheduled at IU or UVA, which means some of IU or UVA jobs miss their deadline and hence the average ODA for UVA and IU drops a bit with CCC. However, both IU and UVA gain significant benefits from the federation, so they can easily offer cloud bursting services during peak load.

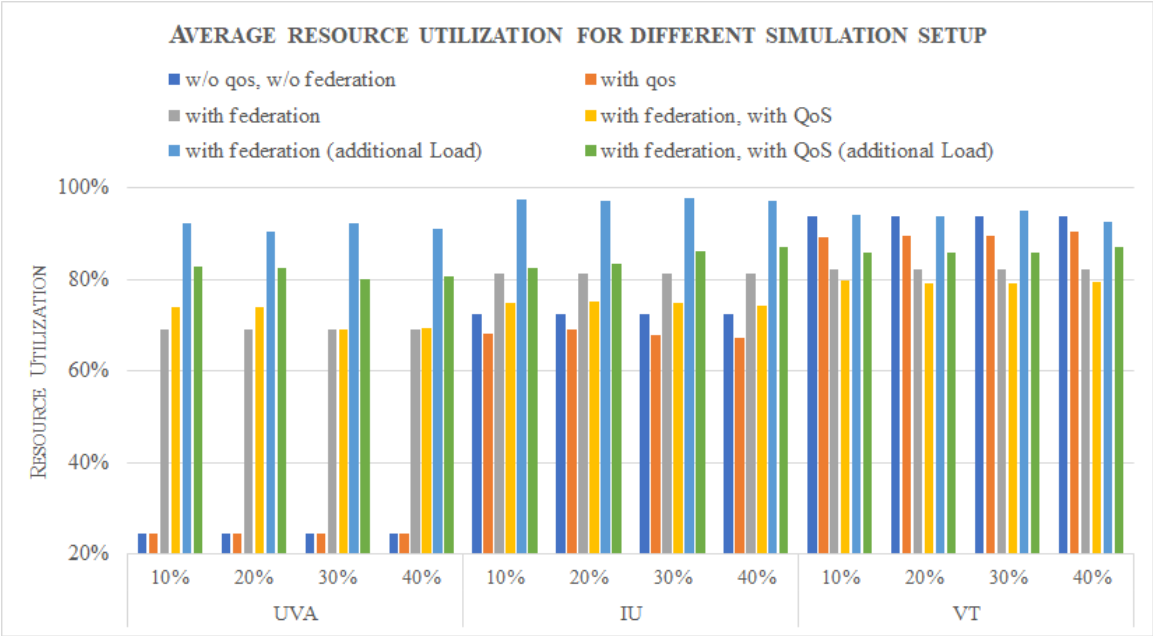

Figure 6.7.1: Average resource utilization for different institutions with varying high priority load. With the federation approach, the utilization at VT drops a little bit, however with additional load the utilization is at par with the baseline resource utilization. Additional load is added due to the fact that federation creates more opportunity for the researchers to add new and /or diverse load on the system. Therefore additional load is only applicable for federation approaches.

6.7 Utilization

Utilization is one metric where differentiated quality of service (more specifically preemption) can have a negative impact. So it is important to quantify the resource utilization for different institutions under different simulation setup. Figure 6.7.1 presents the resource utilization for the three (3) months of simulation period. From the figure, we can observe that, the stand alone utilization for UVA, IU and VT are about 25%, 73% and 94% respectively. However, with QoS only approach the utilization reduces to 68% and 89% for IU and VT. UVA does not incur any reduction as, because of it's very low resource utilization.

However, with the federation only approach, the utilization for both UVA and increases significantly (69% and 81% respectively) showing that VT researchers are using cycles from UVA and IU. Hence, the utilization of VT resources reduces to 82%.

With federation and QoS, the overall load is more or less balanced between the three institutions (utilization is 70%, 74% and 79% at UVA, IU and VT respectively). This actually explains why IU and UVA have a higher positive net value than VT as utilization at VT reduces from 94% to 79%. Again that gives researchers at VT to opportunity to add more load without increasing the wait time.

Finally with additional load, the utilization at all three (3) institution increase. Without QoS, the utilization with additional load at UVA, IU and VT all goes over 90% while with QoS the utilization are 82%, 85% and 86% respectively. Although the utilization is lower with differentiated QoS, more high priority jobs are run on the resources. In other words, "*with QoS, resources are effectively occupied less times, however, they are running more important jobs more often*". Moreover, some preemptible jobs still get value as some standard application packages usage check pointing for restarts (or if the user check points the code himself). Thus our results are pessimistic both in terms of values gained and resource utilization.

7

CONCLUSIONS AND FUTURE WORK

The Campus Computer Co-operative (CCC) uses an open source, standards-compliant, software stack that is a part of the NSF-funded XSEDE infrastructure. The software has been extensively vetted and tested by XSEDE via XSEDE's Software Development and Integration group. The CCC is a realization of the XSEDE Campus Bridging use case CBUC-6, a Shared Virtual Compute Facility [105]. The CCC exists within the XSEDE GFFS namespace.

The CCC provides a market-based resource sharing model that rewards resource providers for sharing their resources and charges users based on the attributes of the resource they use as well as on the quality of service provided. The goal is to provide resources to users when they need them, and provide increased institutional value at reduced cost.

Our hypothesis is: *"A federation of resources among Universities with differentiated QoS and market based resource allocation will result in a greater value and better ability for researchers to complete their most critical analyses on-demand, for all the participating institutes individually and overall"*. In this dissertation, we used simulation on production data traces as described in section 4 to test the hypothesis using the simulation model described in section 5. Our results show that with the federation and differentiated QoS in CCC, all the participating institutions can get significant improvement on their institutional value and on demand access capability. Overall as a federation the benefit is over 350,000 USD in a month on average.

We used production data traces from three (3) of the member institutions: University of Virginia, Indiana University, and Virginia Tech for the simulation. Along with the evaluation of CCC, in our work we also presented analysis on the available traces that would be helpful for future participants and interested institutions to identify whether and how they can benefit from the federation.

To conclude the dissertation, we summarize the key contributions from our simulation and results and discuss limitations. Finally we conclude the chapter and the dissertation with some potential next steps and research directions for the future.

7.1 Key Contributions

Grid economies and market based grid have been widely studied for more than two decades now. The obvious next step of these studies should be the implementation of these approaches in a real

production environment. However, with our extensive study of the state of the art, we have not found any of these approaches being used in any major production environment. Most institutions still use off the shelf workload managers (i.e.- Slurm, PBS, etc), and in some cases like Open Science Grid [14] some level of sharing without any market. In our research, we evaluated the benefits of using a market based grid approach in a real production environment to identify the potential of a grid market in an academic environment. Also we studied some real world production data traces so that we can identify some features and patterns that can help potential candidates to decide whether CCC would be useful for them or not. Based on our results, we can identify two major contributions from our research- a) Establishing the benefits of Campus Compute Co-operative with simulation, b) Analysis of data traces to identify trends in data that can be helpful for future candidates to decide about the potential of CCC.

7.1.1 Establishing the benefits of the Campus Compute Co-operative

The main goal of our thesis is to present and evaluate a market based resource trading in a federated compute environment. Specially we evaluated the impact of different features (i.e. differentiated QoS, resource federation, and resource market) of CCC with different simulation setups. The simulation focuses on net institutional value, average wait-time, on demand access capability, and resource utilization to evaluate the performance of CCC compared to the current approaches. The objective was to test the hypothesis- "*market based resource trading with differentiated QoS in a federated environment can provide better value for each of the institutions with their existing set of shared resources*". To test the hypothesis we collected production data traces from 3 universities and conducted simulation to identify the net value (positive net value means a gain in institutional value while negative net value represents a loss) for each institution under different simulation setup.

With our simulation the benefits of differentiated QoS and federation is very clear as we get positive net value for all the institutions and a combined net value of over a million per quarter for the federation. Further, the on demand capability and resource utilization improves with QoS and federation respectively. Due to preemption, utilization may take a small hit with differentiated QoS, however the overall increase in value justifies the approach. Moreover, better software development practice like check-pointing the code can definitely mitigate some of these loses. Many applications like Amber, Gromacs do that automatically.

We believe the most striking contribution of our research is that, with our data, analysis, results and peer review papers, the case for CCC has become so strong that eight (8) new institutions have decided to join CCC in 2018. We are currently in the process of adding these new institutions in CCC and hopefully in the very near future we will have a functional market based federation among many US schools.

7.1.2 Analysis of Available Traces

For the simulation, we used production data traces of different clusters from three (3) of the member institutions. To the best of our knowledge most of the prior work in market based grid or grid economies used synthesized data or small scale deployment to generate their data. Hence we wanted to analyze these production data traces to explore patterns in the data and features in the traces that would benefit the prospects of CCC.

Specially, we have a number of comparatively cheaper and under-utilized resources at University of Virginia which are only suitable for single node jobs. We wanted to identify whether there are single nodes jobs from other institutions that can be shifted to these resources.

From our analysis on the available data traces from the three institutions, we found some useful

insights, as- a) a large portion of CPU load from each institution is occupied by single node High Throughput Computing jobs, b) all the institutions have a significant number of large parameter sweep jobs which are typically low priority jobs. Thus, having a differentiated QoS will lead to better value for these institutions (even without federation).

High Throughput Computing (HTC) jobs

From our experience with the researchers at different institutions and conversation with the stakeholders, we found that the high throughput computing jobs (long running sequential jobs) and the single node jobs are the easiest to move around. Thus we wanted to identify the High Throughput Computing jobs from different institutions.

From the data analysis, we found that significant portion of jobs at each institution are high throughput or single node jobs. In terms of job count, around 50% of the jobs at UVA and VT are single node jobs and at IU about 70% are single node jobs. In terms of machine cycles, at IU and UVA, more than 1/3rd of the CPU cycles are consumed by single node jobs. At VT, the percentage is slightly lower (about 10-15%). Which shows there is significant scope of moving around these HTC / single node jobs to reap the benefits of the federation.

Parameter sweep jobs of different size

Another interesting feature from the data traces is the size of the parameter sweep jobs. We are interested in parameter sweep jobs because we need to estimate the quality of service specification from the users, which was not available in the current traces. The size and width of the parameter sweep jobs are a good indicator of the potential low priority jobs.

To identify parameter sweep jobs, we iterated through job traces and identified all the jobs with

the same job profile that were submitted within a 2 minute threshold. From the analysis, we observed that UVA has many large parameter sweep jobs, with many of them as large as 1000 jobs. Most of the larger parameter sweep jobs are single node jobs. As a result even though the number of jobs at UVA for the three (3) months were more than 243,000, the overall resource utilization was merely 25%. At IU and VT, the size and count of parameter sweep jobs are comparatively lower. However, IU had a few very large sweep jobs with the maximum size being 1144 jobs.

From these results we can presume that, UVA and IU will potentially have more low priority jobs compared to VT in CCC. Which means the benefit from differentiated QoS would be less for VT compared to that of IU.

7.2 Limitations

There are needless to say some limitations with this approach.

QoS Specification

One major challenge for our simulation was that none of the institutions supported differentiated QoS *per se*. Hence the job traces did not come annotated with job priority information. As a result, we had to use an educated guess to assign the job priorities.

The benefits for differentiated QoS hinges upon the conjecture that users will have different priorities for different jobs. There is little to gain if all the sites are full of high priority jobs all the time. When all the resources are fully utilized there are essentially no degrees of freedom for the scheduler. Jobs cannot be moved around to increase net value, so there is no increase in net value. Similarly if all the jobs are low priority jobs the net value will not change with differentiated QoS.

Additional Load

Due to resource diversity and reduced wait time with CCC, we expect that there will be additional load from different institutions. For example, without the CCC researchers using Bigred II will not submit any large memory jobs as Bigred II does not include any large memory nodes. However, with CCC these researchers can now submit large memory jobs which can be executed on any of the partner facilities (that have large memory nodes). Moreover, with federation the average wait time for typical jobs reduces significantly, so we expect users to submit more jobs as they have a quicker turnaround time.

The issue is- *how to predict these additional loads accurately?*. In this work, rather than guessing the additional load, we duplicated a portion of the load from each institution to account for the expected additional demand. For a real environment, the actual load can be somewhat different which will cause the reality to diverse from simulation. However, since we have randomized the selection for each run and took the average of 5 runs, we expect the result to be significant.

Data Movement Time

Another potential issue in our simulation is that in case of remote execution we have ignored the data movement time prior to the execution of job. The reason we did not include data transfer time in our simulation is because the job traces did not include the input and output size of data for the jobs. Without file size information it is very difficult to predict the actual data transfer time for the jobs when they are executed remotely. However, with CCC we can easily log the input and output sizes for the jobs and that can later be used to model the data transfer time for future simulations.

The data transfer time can really have an impact if jobs require to transfer a huge input data set for it's execution or need to move a huge output data back to the user. Nonetheless, with the current

advent in wide area networks and bandwidth, today the network data transfer time is often comparable to the disk i/o rate [56]. Today we can easily transfer gigabytes of data over the network. Hence, even if the data to be transferred is in a few GB, it just adds a few minutes to the execution time. Particularly if the job execution time is in hours then these few extra minutes for data transfer can actually be ignored. With our experience with real users so far, we have observed that the input and output data size ranges from a few megabytes to a few gigabytes in most cases. Thus in most case they can be ignored.

However, the problem becomes little more complicated if the job execution time is very small or more precisely, the ratio of compute time to data transfer time is too small. For example if a job requires 100 GB of data transfer and only runs for 60 seconds it is clearly not worth moving the execution to another site.

This is often referred to as the computation granularity measured in compute-seconds/megabyte. Fortunately most jobs have relatively large granularity. Further, JSDL (the job submission language) supports the notion of local scratch spaces where users may cache large input data sets that may be reused across many runs. In other words, if ten jobs to be run at a site each need the same 100 GB file it only needs to be transferred once to the site. We are considering adding a JSDL scheduling comment that would indicate the granularity to make it possible for the user to inform the Grid Queue scheduler of the approximate computational granularity.

Different execution time due to Diversity of Resources

One of the benefits of resource federation is the diversity of resources that we can get from different institutions. However, this can lead to a potential concern in our simulation as with different resource configuration the execution times for the jobs may differ. Specially due to the variation in

the CPU architecture or GPU memory or available cache memory etc. the execution time of a job can vary significantly.

However, we have benchmarked the performance of a few common applications (i.e.- lammps, theano, etc.) on different machines and the execution times were comparable. So in our simulation we used the same execution time for a job regardless of where it is executed. In future though, as the number of institutions grow, we plan to do a more extensive benchmark for different applications in different resources and from that use a multiplication factor for the execution time based on where the job is executed and a price scaling based on application performance vector.

7.3 Future Research Directions

In future, we will focus on addressing the limitations specified in Section 7.2. Specially now that the idea of CCC is catching on (10 institutions have decided to come forward and join the consortium), once CCC is deployed, we will conduct these simulations with real CCC traces that would contain the differentiated QoS and additional loads embedded into it. With these enhanced traces we will be able to predict the net value gained for each institution, in particular for those institutions who have not yet joined the federation.

The CCC is still in its development stage. Next steps are to move more applications onto the CCC, build up the user base, and to expand the set of participating institutions; in other words to expand the size of the market. The CCC is an open organization that other institution or research labs are invited to participate. Expanding the market will provide additional value both for the new market participants and existing market participants. Why adding new members helps the new members is clear. A larger market adds value to existing market participants by expanding the pool of resources that can be used and expanding the set of users that will use the resources.

Technically we want to

- examine moving to a dynamic pricing model where resource providers advertise their prices for different qualities of service rather than using fixed prices
- provide users with the option of creating their own custom gridQueues that optimize their own objective functions, e.g. trade-off performance for cost
- allow users to add desktop VMs into the resource mix to earn additional allocation
- support starting VM's and container for users, not just jobs

The deployment phase of CCC has just started. We believe in future this work would be a stepping stone for the market based federations in an academic environment and will shepherd new research areas in this domain.

References

[1] Arjuna agility. http://www.arjuna.com/agility [accessed May 2018].

[2] Windows azure. http://www.windowsazure.com/en-us/ [accessed Jan 2014].

[3] cilogon. http://www.cilogon.org/ [accessed May 2018].

[4] Dagman applications. $https://www.cl.cam.ac.uk/manuals/condor-V6_8_3-Manual/2_11DAGMan_Applications.html$ [accessed May 2018].

[5] Dell hybrid cloud. https://www.dellemc.com/en-us/cloud/hybrid-cloud-computing/index.htm [accessed May 2018].

[6] Amazon elastic compute cloud. http://aws.amazon.com/ec2/ [accessed May 2018], .

[7] Amazon ec2 instance types. https://aws.amazon.com/ec2/instance-types/ [accessed May 2018], .

[8] Futuregrid. https://portal.futuregrid.org/ [accessed Jan 2014].

[9] Google clooud platform. https://cloud.google.com/ [accessed May 2018].

[10] The genesis ii project, virginia center for grid research. http://genesis2.virginia.edu/wiki/ [accessed May 2018].

[11] Incommon: Security, privacy and trust for the research and education community. http://www.incommon.org/ [accessed May 2018].

[12] Lammps. http://lammps.sandia.gov/ [accessed May 2018].

[13] Open grid forum. https://www.ogf.org/ [accessed May 2018].

[14] Open science grid. www.opensciencegrid.org [accessed May 2018].

[15] Rackspace: Managed dedicated cloud computing services. https://www.rackspace.com/ [accessed May 2018].

[16] Rightscale: Cloud portfolio management. https://www.rightscale.com/ [accessed May 2018].

[17] Scalr enterprise cloud management platform. https://www.scalr.com/ [accessed May 2018].

[18] Extreme science and engineering discovery environment. https://www.xsede.org/ [accessed May 2018], .

[19] Xsede usecase registry. https://software.xsede.org/registry-dev/index.php [accessed May 2018], .

[20] Xsede campus bridging l3 architectural response. Technical report, Technical report, XSEDE, Tech. Rep, 2016.

[21] David Abramson, Rok Sosic, Jonathan Giddy, and B Hall. Nimrod: a tool for performing parametrised simulations using distributed workstations. In *Proceedings of the Fourth IEEE International Symposium on HPDC, 1995*, pages 112–121. IEEE, 1995.

[22] Jörn Altmann, Costas Courcoubetis, George D Stamoulis, Manos Dramitinos, Thierry Rayna, Marcel Risch, and Chris Bannink. Gridecon: A market place for computing resources. In *Grid Economics and Business Models*, pages 185–196. Springer, 2008.

[23] David P Anderson. Boinc: A system for public-resource computing and storage. In *Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on*, pages 4–10. IEEE, 2004.

[24] Ali Anjomshoaa, Fred Brisard, Michel Drescher, Donal Fellows, An Ly, Stephen McGough, Darren Pulsipher, and Andreas Savva. Job submission description language (jsdl) specification, version 1.0. In *Open Grid Forum, GFD*, volume 56, 2005.

[25] Felix Bachmann, Ian Foster, Andrew Grimshaw, Dave Lifka, Morris Riedel, and Steve Tuecke. Xsede architecture–level 1 and 2 decomposition, 2012.

[26] Felix Bachmann, Ian Foster, Andrew Grimshaw, David Lifka, Morris Riedel, and Steven Tuecke. Xsede architecture level 3 decomposition. *version 0.972, Jun*, 2013.

[27] Tim Banks. Web services resource framework (wsrf)–primer v1. 2. *OASIS committee draft*, 2006.

[28] Fran Berman, Rich Wolski, Silvia Figueira, Jennifer Schopf, and Gary Shao. Application-level scheduling on distributed heterogeneous networks. In *Proceedings of the 1996 ACM/IEEE Conference on Supercomputing, 1996*, pages 39–39. IEEE, 1996.

[29] Fran Berman, Geoffrey Fox, and Anthony JG Hey. *Grid computing: making the global infrastructure a reality*, volume 2. John Wiley and sons, 2003.

[30] David Bernstein and Deepak Vij. Intercloud directory and exchange protocol detail using xmpp and rdf. In *Services (SERVICES-1), 2010 6th World Congress on*, pages 431–438. IEEE, 2010.

[31] David Bernstein and Deepak Vij. Simple storage replication protocol (ssrp) for intercloud. In *Proceedings of the 2nd International Conference on Emerging Network Intelligence (EMERGING'10)*, pages 30–37, 2010.

[32] David Bernstein and Deepak Vij. Using semantic web ontology for intercloud directories and exchanges. In *International Conference on Internet Computing*, pages 18–24, 2010.

[33] David Bernstein, Erik Ludvigson, Krishna Sankar, Steve Diamond, and Monique Morrow. Blueprint for the intercloud-protocols and formats for cloud computing interoperability. In *Internet and Web Applications and Services, 2009. ICIW'09. Fourth International Conference on*, pages 328–336. IEEE, 2009.

[34] David Bernstein, Deepak Vij, and Stephen Diamond. An intercloud cloud computing economy-technology, governance, and market blueprints. In *SRII Global Conference (SRII), 2011 Annual*, pages 293–299. IEEE, 2011.

[35] Volker Böhm and Hans Haller. Demand theory. *The New Palgrave Dictionary of Economics: Volume 1–8*, pages 1311–1320, 2008.

[36] Don Box, Erik Christensen, Francisco Curbera, Donald Ferguson, Jeffrey Frey, Marc Hadley, Chris Kaler, David Langworthy, Frank Leymann, Brad Lovering, S. Lucco, S. Millet, N. Mukhi, M. Nottingham, D. Orchard, J. Shewchuk, E. Sindambiwe, T. Storey, S. Weerawarana, and S. Winkler. Web services addressing (ws-addressing), 2004.

[37] Rajkumar Buyya and Kris Bubendorfer. *Market-oriented grid and utility computing*. Wiley Online Library, 2010.

[38] Rajkumar Buyya and Manzur Murshed. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurrency and computation: practice and experience*, 14(13-15):1175–1220, 2002.

[39] Rajkumar Buyya, David Abramson, and Srikumar Venugopal. The grid economy. *Proceedings of the IEEE*, 93(3):698–714, 2005.

[40] Rajkumar Buyya, Rajiv Ranjan, and Rodrigo N Calheiros. Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 13–31. Springer, 2010.

[41] Emanuele Carlini, Massimo Coppola, Patrizio Dazzi, Laura Ricci, and Giacomo Righetti. Cloud federations in contrail. In *European Conference on Parallel Processing*, pages 159–168. Springer, 2011.

[42] Use Cases. Functional requirements for inter-cloud computing. In *White Paper, Global Inter-Cloud Technology Forum*, 2010.

[43] Steve J Chapin, Dimitrios Katramatos, John Karpovich, and Andrew Grimshaw. Resource management in legion. *Future Generation Computer Systems*, 15(5):583–594, 1999.

[44] Grid Computing. A practical guide to technology and applications, 2004.

[45] Karl Czajkowski, Donald F. Ferguson, Ian Foster, Jeffrey Frey, Steve Graham, Igor Sedukhin, David Snelling, Steve Tuecke, and William Vambenepe. The ws-resource framework, version 1.0. Technical report, 2004.

[46] Erik Leuven Demeulemeester and Willy S Herroelen. *Project scheduling: a research handbook*, volume 49. Springer Science & Business Media, 2006.

[47] M Drescher, A Anjomshoaa, G Williams, and D Meredith. Jsdl parameter sweep job extension. *Report no. GFD*, 149, 2008.

[48] Dror G Feitelson, Larry Rudolph, and Uwe Schwiegelshohn. Parallel job scheduling—a status report. In *Workshop on Job Scheduling Strategies for Parallel Processing*, pages 1–16. Springer, 2004.

[49] I Foster, H Kishimoto, A Savva, D Berry, A Djaoui, A Grimshaw, B Horn, F Maciel, F Siebenlist, R Subramaniam, J. Treadwell, and J.V. Reich. The open grid services architecture, version 1.5. *Open Grid Forum*, 2006.

[50] I Foster, A Grimshaw, P Lane, W Lee, M Morgan, S Newhouse, S Pickles, D Pulsipher, C Smith, and M Theimer. Ogsa basic execution service, version 1.0, gfd. 108. Technical report, 2007.

[51] Ian Foster and Carl Kesselman. *The Grid 2: Blueprint for a new computing infrastructure*. Elsevier, 2003.

[52] Ian Foster, Carl Kesselman, Jeffrey M Nick, and Steven Tuecke. Grid services for distributed system integration. *Computer*, 35(6):37–46, 2002.

[53] Clive Gerrard, Paul Haldane, Sindre Hamlander, Stephen McGough, Paul Robinson, Dave Sharples, Dan Swan, Paul Watson, and Stuart Wheater. Intelligent power management over large clusters. 2010.

[54] Avik Ghosh. Private Communication, Aug, 2013.

[55] A Grimshaw, D Snelling, and M Morgan. Ws-naming specification. In *Open Grid Forum, GFD-109*, 2007.

[56] Andrew Grimshaw and Vanamala Venkataswamy. Performance of the global federated file system (gffs). Technical report, University of Virginia, 2018.

[57] Andrew Grimshaw, Mark Morgan, Duane Merrill, Hiro Kishimoto, Andreas Savva, David Snelling, Chris Smith, and Dave Berry. An open grid services architecture primer. *Computer*, (2):27–34, 2009.

[58] Andrew Grimshaw, Mark Morgan, and Karolina Sarnowska. Ws-naming: location migration, replication, and failure transparency support for web services. *Concurrency and Computation: Practice and Experience*, 21(8):1013–1028, 2009.

[59] Andrew Grimshaw, Mark Morgan, and Avinash Kalyanaraman. Gffs-the xsede global federated file system. *Parallel Processing Letters*, 23(02), 2013.

[60] Andrew Grimshaw, Md Anindya Prodhan, Alexander Thomas, Craig Stewart, and Richard Knepper. Campus compute co-operative (ccc): A service oriented cloud federation. In *e-Science (e-Science), 2016 IEEE 12th International Conference on*, pages 1–10. IEEE, 2016.

[61] Andrew S Grimshaw and Anand Natrajan. Legion: Lessons learned building a grid operating system. *Proceedings of the IEEE*, 93(3):589–603, 2005.

[62] Andrew S Grimshaw, Anh Nguyen-Tuong, Michael J Lewis, and Mark Hyett. Campus-wide computing: Early results using legion at the university of virginia. *International Journal of High Performance Computing Applications*, 11(2):129–143, 1997.

[63] Andrew S Grimshaw, Wm A Wulf, et al. The legion vision of a worldwide virtual computer. *Communications of the ACM*, 40(1):39–45, 1997.

[64] Martin Gudgin, Marc Hadley, and Tony Rogers. Web services addressing 1.0-core. *W3C, W3C Recommendation, May*, 2006.

[65] Hossain Haj-Hariri. Private Communication, Jul, 2013.

[66] Ira Hall. Private Communication, Jul, 2013.

[67] Mohammad Mehedi Hassan and Eui-Nam Huh. A novel market-oriented dynamic collaborative cloud service platform. In *Handbook of Cloud Computing*, pages 407–434. Springer, 2010.

[68] Mohammad Mehedi Hassan, Biao Song, Changwoo Yoon, Hyun Woo Lee, and Eui-Nam Huh. A novel market oriented dynamic collaborative cloud service infrastructure. In *Services-II, 2009. SERVICES-2'09. World Conference on*, pages 9–16. IEEE, 2009.

[69] Ugo Hincelin. Private Communication, Aug, 2013.

[70] Marty Humphrey, Chris Smith, Marvin Theimer, and Glenn Wasson. Jsdl hpc profile application extension. In *Open Grid Forum*, 2007.

[71] Lori Ann Kaplan. *Resource-constrained project scheduling with preemption of jobs*. UMI, 1996.

[72] Katarzyna Keahey, T Fredian, Qian Peng, David P Schissel, M Thompson, Ian Foster, M Greenwald, and Douglas McCune. Computational grids in action: the national fusion collaboratory. *Future Generation Computer Systems*, 18(8):1005–1015, 2002.

[73] Gabor Kecskemeti, Michael Maurer, Ivona Brandic, Attila Kertesz, Zs Nemeth, and Schahram Dustdar. Facilitating self-adaptable inter-cloud management. In *Parallel, Distributed and Network-Based Processing (PDP), 2012 20th Euromicro International Conference on*, pages 575–582. IEEE, 2012.

[74] Dalibor Klusáček and Hana Rudová. Alea 2 – job scheduling simulator. In *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques (SIMUTools 2010)*. ICST, 2010.

[75] Chris Koeritz. Genesis ii omnibus reference manual. Technical report, Genesis II Group, University of Virginia, 2012.

[76] R Mach, S Jackson, L McGinnis, and Pittsburgh Supercomputing Center. Usage record–format recommendation. 2006.

[77] Dennis A Margosan, Joseph L Smilanick, Gilbert F Simmons, and Delmer J Henson. Combination of hot water and ethanol to control postharvest decay of peaches and nectarines. *Plant Disease*, 81(12):1405–1409, 1997.

[78] Attila Csaba Marosi, Gabor Kecskemeti, Attila Kertesz, and Peter Kacsuk. Fcm: an architecture for integrating iaas cloud systems. IARIA, 2011.

[79] Andrew Stephen McGough, Matthew Forshaw, Clive Gerrard, Paul Robinson, and Stuart Wheater. Analysis of power-saving techniques over a large multi-use cluster with variable workload. *Concurrency and Computation: Practice and Experience*, 25(18):2501–2522, 2013.

[80] Michael McIntosh, Martin Gudgin, K Scott Morrison, and Abbie Barbir. Basic security profile version 1.0. *WS-I Standard*, 30, 2007.

[81] Peter Mell and Tim Grance. The nist definition of cloud computing. *Communications of the ACM*, 53(6):50, 2010.

[82] Daniel A Menasce and Emiliano Casalicchio. Qos in grid computing. *Internet Computing, IEEE*, 8(4):85–87, 2004.

[83] D Merrill and A Grimshaw. Genesis ii security architecture. Technical Report CS2009-07, Department of Computer Science, University of Virginia, September 2009.

[84] M Morgan, N Chue-Hong, and M Drescher. Byteio specification 1.0. Technical report, 2006.

[85] M Morgan, Andrew Grimshaw, and O. Tatebe. Rns specification 1.1. In *Open Grid Forum*, 2006.

[86] Mark M Morgan and Andrew S Grimshaw. Genesis ii-standards based grid computing. In *Cluster Computing and the Grid, 2007. CCGRID 2007. Seventh IEEE International Symposium on*, pages 611–618. IEEE, 2007.

[87] Anthony Nadalin, Marc Goodner, Martin Gudgin, Abbie Barbir, and Hans Granqvist. Ws-trust 1.3. *OASIS Standard*, 19:2007, 2007.

[88] Anand Natrajan, Marty A Humphrey, and Andrew S Grimshaw. *Capacity and capability computing using Legion*. Springer, 2001.

[89] Anand Natrajan, Marty A Humphrey, and Andrew S Grimshaw. Grid resource management in legion. In *Grid resource management*, pages 145–160. Springer, 2004.

[90] Bill Nitzberg, Jennifer M Schopf, and James Patton Jones. Pbs pro: Grid computing and scheduling attributes. In *Grid resource management*, pages 183–190. Springer, 2004.

[91] Pradeep Padala, Cyrus Harrison, Nicholas Pelfort, Erwin Jansen, Michael P Frank, and Chaitanya Chokkareddy. Ocean: the open computation exchange and arbitration network, a market approach to meta computing. In *International Symposium on Parallel and Distributed Computing*, page 185. IEEE, 2003.

[92] Ruth Pordes, Don Petravick, Bill Kramer, Doug Olson, Miron Livny, Alain Roy, Paul Avery, Kent Blackburn, Torre Wenaus, W Frank, I. Foster, R. Gardner, M. Wilde, A. Blatecky, J. McGee, and R. Quick. The open science grid. In *Journal of Physics: Conference Series*, volume 78. IOP Publishing, 2007.

[93] A Prodhan and A Grimshaw. Prospect of market bagsed computing resource allocation in university of virginia.

[94] Md Anindya Prodhan and Andrew Grimshaw. Market-based on demand scheduling (mbods) in co-operative grid environment. In *Proceedings of the 2015 XSEDE Conference*, page 26. ACM, 2015.

[95] Md Anindya Prodhan and Andrew Grimshaw. An evaluation of a market based resource trading in a multi-campus compute co-operative (ccc). In *15th International Conference on theEconomics of Grids, Clouds, Systems, and Services, GECON*, 2018.

[96] Md Anindya Prodhan and Andrew Grimshaw. Modeling a market based federated cloud for academic computing. In *2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. IEEE, 2019 (submitted).

[97] Ori Regev and Noam Nisan. The popcorn market. online markets for computational resources. *Decision Support Systems*, 28(1):177–189, 2000.

[98] Xiaojuan Ren, Seyong Lee, Rudolf Eigenmann, and Saurabh Bagchi. Resource failure prediction in fine-grained cycle sharing systems. In *Proc. of Fifteenth IEEE International Symposium on High Performance Distributed Computing (HPDC-15)*, pages 19–23, 2006.

[99] Benny Rochwerger, David Breitgand, Eliezer Levy, Alex Galis, Kenneth Nagin, Ignacio Martín Llorente, Rubén Montero, Yaron Wolfsthal, Erik Elmroth, Juan Caceres, et al. The reservoir model and architecture for open federated cloud computing. *IBM Journal of Research and Development*, 53(4):4–1, 2009.

[100] R. Ryzhkov. Jsdl++: Extending the jsdl specification used in grid computing environments. Technical report, Department of Computer Science, University of Virginia, 2013.

[101] Paul Anthony Samuelson, William D Nordhaus, and John McCallum. *Macroeconomics. 19th Edition*. McGraw-Hill Ryerson, 2009.

[102] A Savva, H Kishimoto, S Newhouse, and D Pulsipher. Ogsa® ems architecture scenarios, 2007.

[103] Steven Stern. Private Communication, Jul, 2013.

[104] Craig A Stewart, Richard Knepper, James Ferguson, Felix Bachmann, Ian Foster, Andrew Grimshaw, Victor Hazlewood, and David Lifka. What is campus bridging and what is xsede doing about it? In *Proceedings of the 1st Conference of XSEDE*, page 47. ACM, 2012.

[105] Craig A Stewart, Richard Knepper, Andrew Grimshaw, Ian Foster, Felix Bachmann, David Lifka, Morris Riedel, and Steven Tueke. Xsede campus bridging use cases. Technical report, Technical report, XSEDE, Tech. Rep, 2012.

[106] Miklos Szeredi. Filesystem in userspace. https://github.com/libfuse/libfuse, 2005.

[107] Katsumi Toda, Yasushi Okada, Mohamad Zubair, Ken-ichiro Morohashi, Toshiji Saibara, and Teruhiko Okada. Aromatase-knockout mouse carrying an estrogen-inducible enhanced green fluorescent protein gene facilitates detection of estrogen actions in vivo. *Endocrinology*, 145(4):1880–1888, 2004.

[108] Carl A Waldspurger, Tad Hogg, Bernardo A Huberman, Jeffrey O Kephart, and W Scott Storn. Spawn: A distributed computational economy. *IEEE Transactions on Software Engineering*, 18(2):103–117, 1992.

[109] Derek Weitzel, Brian Bockelman, Dan Fraser, Ruth Pordes, and David Swanson. Enabling campus grids with open science grid technology. In *Journal of Physics: Conference Series*, volume 331, page 062025. IOP Publishing, 2011.

[110] Rich Wolski, James S Plank, Todd Bryan, and John Brevik. G-commerce: Market formulations controlling resource allocation on the computational grid. In *Proceedings of 15th International Symposium on Parallel and Distributed Processing*, pages 8–pp. IEEE, 2001.

Colophon

This thesis was typeset using LaTeX, originally developed by Leslie Lamport and based on Donald Knuth's TeX. The body text is set in 11 point Arno Pro, designed by Robert Slimbach in the style of book types from the Aldine Press in Venice, and issued by Adobe in 2007. A template, which can be used to format a PhD thesis with this look and feel, has been released under the permissive MIT (X11) license, and can be found online at github.com/suchow/ or from the author at suchow@post.harvard.edu.