Distributed Virtual Computers (DVC): Simplifying the Development of High Performance Grid Applications

Nut Taesombut and Andrew A. Chien
Department of Computer Science and Engineering
University of California at San Diego
{ntaesomb,achien}@ucsd.edu

Abstract

Distributed Virtual Computer (DVC) is a computing environment which simplifies the development and execution of distributed applications on computational grids. DVC provides a simple set of abstractions to simplify application management of naming, security, communication, and resource, easing use of highly dynamic and heterogeneous resource environments. These abstractions enable complex collections of grid resources to be used in a fashion similar to private user or workgroup resources. The DVC model is attractive for lambda-grids with circuit-switched optical networks, providing a structure for exploiting unique communication and security properties. Examples of DVC's include virtual clusters and virtual heterogeneous resource collections. We introduce the concept of a DVC, its system structure and mechanisms. We discuss the potential benefits of DVC's for application programmers.

1. Introduction

The past decade has seen dramatic growth in e-science applications on distributed cyber-infrastructures in terms of number, scale, and complexity. Emerging scientific and engineering applications, such as analysis and virtualization of neuroscience, geophysical, or other forms of scientific data, require an aggregation of large-scale computing resources and geographically dispersed scientific data. The vision of "grid" computing [1] has emerged to meet the needs of such applications. Grid resources, such as computing clusters, petabyte data stores and other high-end scientific instruments in multiple organizations, can be securely shared through a *Virtual Organization* (VO) [2], enabling far greater computing and collaboration capabilities. A VO is a set of relationships and sharing policies that grant users access to resources across traditional organizational boundaries. VO's are configured cooperatively amongst the IT administrators of the organizations, so their lifetime is long, months or even years.

When compared to either sequential or parallel programming, the difficulties in developing grid applications which efficiently exploit these complex and heterogeneous infrastructures are daunting [3]. Grid applications (and consequently grid application programmers) must contend with the complexity of a dynamic and untrustworthy resource environment spanning multiple administrative domains. Grid resources are heterogeneous, varying in performance, security, availability, and runtime behavior. In the face of these challenges, to be efficient, applications must identify and select appropriate resources rapidly, use them to achieve secure and robust performance, and even tolerate asynchronous changes in resource performance and availability. Though a number of research projects on programming tools are underway [4], these systems are far from delivering mature solutions to designers of grid applications who have little knowledge of grid environments and distributed systems issues. In the current environment, the system knowledge and programming effort implied are slowing the spread and deployment of grid applications.

We propose a new approach based on the notion of a *Distributed Virtual Computer* (DVC) which presents to the application a simple computing environment with complexity of use comparable to a local distributed environment. Our objective is to make it easy for inexperienced users to build high-performance, robust and secure grid applications without the need for them to understand complex grid resource environments. DVC's use a distributed virtual machine model to simplify application models of naming, security, and performance while hiding the complexity that attends these issues in typical grid computing environments. A key distinguishing feature of DVC's is to separate the configuration of resources from the application programming and execution. DVC's allow users to

bind a collection of grid resources prior to execution time, facilitating efficient resource discovery and selection.

The primary contributions of this paper are the conception, design, and analysis of the benefits of distributed virtual computers (DVC's). Specifically these include:

- a distributed virtual computer, expressed as a resource pool request, which provides the execution context for a grid application,
- a simple namespace which enables convenient application process management,
- an integrated set of communication primitives which coordinate with resource monitoring and management to provide simpler communication semantics, and
- a simple security model which enables DVC's to leverage existing Virtual Organization structures, but allows applications to manage distributed security within a single trust domain.

The advent of dynamic lambda circuits provides an opportunity to exploit optical lightpaths to establish dedicated multi-gigabit communication channels. These lambda-based networks are private and can be effectively formed on-demand. The DVC computing model can exploit such novel communication capabilities to dynamically construct high-performance and secure communication structures among grid resources, providing computing environments suitable for sophisticated scientific applications. Our work is part of the OptIPuter project [5], one such research effort to exploit availability of such networking technology in establishing a high-performance computing infrastructure.

The remainder of this paper is organized as follows: In Section 2, we discuss the challenges of developing applications on grids. In Section 3, we present the DVC model, comprising its fundamental concepts and system structure. In Section 4, we outline the key elements in a DVC. Section 5 gives two examples of DVC to make the concept concrete, and Section 6 relates DVC's to other technologies in the grid space. We survey related work in Section 7, and close with a summary and future work in Section 8.

2. Grid Programming Challenges

Many difficulties have emerged as application developers and users move from a local programming environment to a grid computing environment. Application developers must contend with grid environments that are distributed, heterogeneous, dynamic and even untrusted in terms of resources and networks involved. Grid resources vary in their types, capabilities, and runtime behaviors. They are drawn from a range of distributed resource providers that represent multiple administrative domains and may have different security and resource management policies and mechanisms imposed.

Grid application developers want to build applications that deliver high quality of service, capability, security, and are efficient in the use of resources. However, this is difficult for a number of reasons:

- *Shared Network*: Communication cannot be trusted, necessitating the use of security mechanisms such as certificates, digital signatures, data encryption, etc. to ensure correct application behavior, data integrity and confidentiality. Managing a security infrastructure and correct use of these protocols is difficult, significantly increasing programming effort.
- *Best-effort Network*: Availability and performance of the network is unpredictable, forcing applications to monitor performance and react or adapt to changes in its behavior (e.g. network congestion, unreachable end-point, etc.) to deliver predictable application service. This complicates the design and development of robust grid applications.
- *Heterogeneous Resource Naming*: Grid resources often span multiple sites which impose diverse, site-dependent resource naming mechanisms (e.g. full naming, dynamic and internal network address, and security credentials). Managing these heterogeneous names for resources complicates application programming.
- *Runtime Resource Allocation and Management*: Applications must discover, select and allocate resources at run time. This activity is complex and time-consuming, involving heterogeneous access to multiple resource providers and negotiation which optimizes over diverse policies.

Directly grid applications need to achieve high levels of quality and capability, but must do so dealing with complex, heterogeneous, and badly behaving underlying services and resources. This makes grid programming even more difficult than traditional sequential, parallel, and even distributed computing. Our objective is to develop programming abstractions and tools which enable grid applications to be built and deployed in a more convenient way like in a local private distributed computing environment.

3. Distributed Virtual Computer Overview

Distributed Virtual Computer (DVC) is a simple computing environment for developing distributed

applications on computational grids. The DVC provides an abstraction layer that insulates application developers from the full complexity of building an adaptive, robust and secure application on a highly dynamic and heterogeneous grid. DVC's provide applications with a single namespace, a single security domain, simple communication primitives, and simple resource management. Altogether, the resulting programming complexity is comparable to a workgroup or subnet. Key DVC abstractions include:

- *Single Namespace*: A DVC provides a simple flat namespace for resources, masking physical location, network connection structure, protocol structure, or any site-specific naming mechanisms. This namespace can be grown, shrunk, subsetted or combined, allowing flexible usage.
- *Single Security Domain*: The DVC computing model assumes each DVC is private to a single or group of users; DVC implementation mechanisms ensure it is secured as a private local distributed environment. Within the DVC, users have full control over their computing environments, and may choose to set various security management policies appropriate for specific users, applications, etc.
- *Simple Communication Primitives*: A DVC provides a simple set of communication primitives defined against the simple namespace. These operations are coupled to DVC management and monitoring, and thus have a variety of convenient return codes in exceptional circumstances and enable novel communication capabilities to be exposed.
- *Simple Resource Management*: DVC's are initiated with a base set of resource requirements (end resources and network elements). These requirements are realized by configuration modules and runtime libraries that support DVC formation, freeing applications from these concerns. The resources bound into a DVC can be increased or reduced via explicit application control, and in some cases, resources can be transparently replaced in response to errors, revocations, or faults.

Figure 1 illustrates a high-level view of a DVC computing environment (shown at the bottom-left). The DVC environment can be viewed as a collection of computing resources assembled from several remote sites. These sites can span multiple administrative domains which each enforce distinct security and resource management policies. The DVC abstraction consists of a simple computing environment where the collected resources (or *DVC resources*) are tightly connected via a reliable, private network and controlled under one administrative domain (by *DVC domain controller*). The complexity of low-level site-specific management systems and diversity of resources are mitigated through the DVC abstractions.

A DVC domain controller serves as a resource manager, a security administrator, a communication mediator and a job manager of the user's DVC environment. It also provides a virtualization service that enables consistent naming and access methods for DVC resources. A users can construct secure group communication structures among them, set various resource and security policies and mechanisms, as well as submit distributed jobs to run, via the DVC controller. Grid resource are reserved and bound into the DVC environment before an application is actually executed. During execution, a user may allocate or deallocate additional grid resources for the DVC.

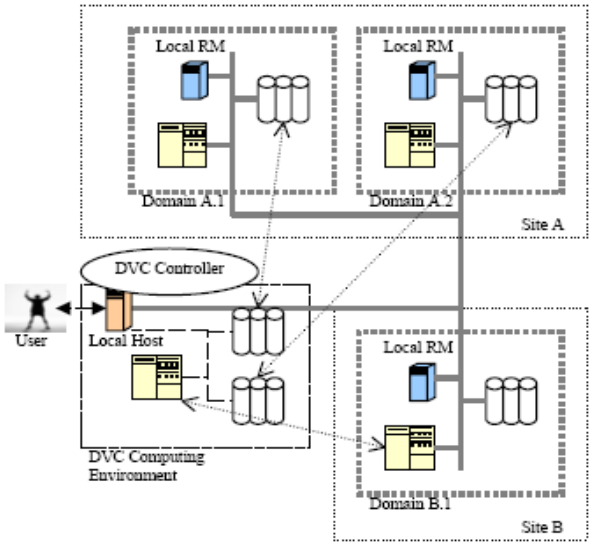

Figure 1. High-level view of DVC environment (RM represents resource manager)

4. DVC Components

Realizing the DVC abstractions requires the cooperation of a number of high-level services and libraries, including resource naming, event monitoring, as well as security, communication, resource, and job execution management. In addition, the DVC computing environment provides application developers a set of convenient routines and their application interfaces (DVC APIs) to interact with these services (via the DVC domain controller). This section first presents a DVC domain controller, and then discusses each of these elements in further detail.

4.1 DVC Domain Controller

The purpose of a DVC controller is to reduce the burden of a user in directly interacting with underlying

complex management systems. It instead offers a set of simplified services that the user can use to configure and manage his computing environment appropriately to meet his application needs. The DVC domain controller is realized by a group of daemon processes cooperatively running on a user's local host and remote resources. Once the user starts the DVC system on his local machine and initiates a new DVC environment, a single daemon, called a *DVC manager*, is created and associated with his DVC. The DVC manager takes a major responsibility in controlling and managing DVC. It acts as a resource broker that discovers, acquires, and binds resources into the DVC environment. It also serves as a DVC domain security authority, managing trust relationships and implementing security policies and mechanisms. In addition, the DVC manager monitors resources to detect failure or unreachability. When new resources are allocated into the DVC environment, the DVC manager spawns another daemon process, called a *ghost manager*, to run on each resource. The major function of the ghost manager is to enable the DVC computing environment at its attached resource and periodically report resource status (e.g. utilization, availability and currently running tasks) back to the DVC manager. The DVC manager and ghost managers periodically exchange DVC configuration and status information. If there are many users working on the same host, different DVC managers will be created and associated to individual users so that they cannot compromise one another's computation.

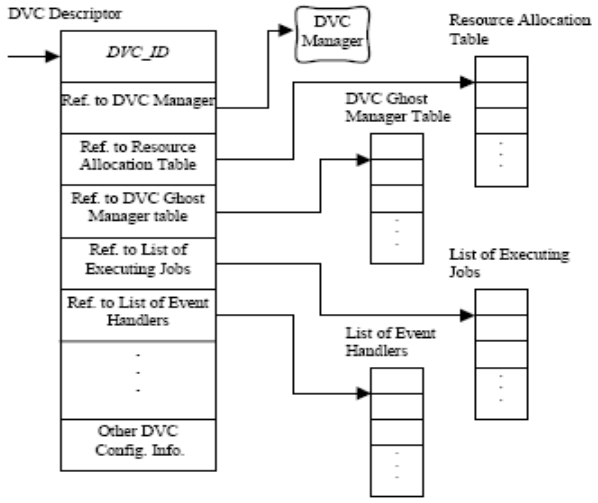

Figure 2. DVC Descriptor Structure

The DVC manager maintains a DVC descriptor which contains the state of a DVC (see Figure 2). The descriptor includes information about DVC resources, the ghost managers on those resources, the active jobs, and subscribed event handlers. In addition, the descriptor maintains DVC configuration, including resource, communication, and security.

4.2 Resource Naming and Namespace Management

DVC naming provides a flat namespace, and a set of naming operations, simplifying computation over a highly complex grid computing environment. Specifically, to reduce the complexity of handling diverse low-level naming mechanisms, a new unique name, called *ZoRN* (Zero-origin Resource Name), is generated and assigned to each newly allocated resource. The ZoRN is a simple and location-independent name used internally within the DVC environment. Users and applications can access and employ grid resources via their respective ZoRNs and query their detailed information (e.g. physical location and supported services) via simple APIs. In grid environments, computing resources are widely available and they are often interchangeable. To compete with dynamic application needs and unexpected system failures, DVC resources can be effectively and transparently replaced. A ZoRN can be associated with different physical resources over time while its presence remains intact. Hence, from the applications' perspective, grid resources within a DVC environment are as reliable and easily accessible as they are in a local distributed environment.

Initially, all resources are bound into a single namespace. Users can create hierarchical or other structures using these names. DVC's allow operations on sets of these names, enabling convenient description of relationships among resources. For example, a set of names is a natural basis for describing trust or communication mechanisms (see Section 4.3). A set of names is also a natural way to limit a scope in which security, performance or other policy control domains are applied.

4.3 Communication

DVC provides a simple set of communication primitives defined against the simple namespace. There are numerous potential novel network capabilities that can be expressed through the DVC model; here we focus on three. First, many dedicated optical networks and virtual private network [6] technologies enable easy construction of groups of connected endpoints. This type of private network is directly relevant to many applications anticipated for DVC's, and enables a range of optimized group protocols for fast transport [7, 8]. Second, researchers are exploring optical/photonic level multicast techniques which could also be naturally expressed and managed as part of the groups of endpoints [9], though IP multicast groups and naming could also be employed as an interface. Third, radical

communication architectures, such as LambdaRAM [10] could also be expressed in our DVC communication architecture, supporting remote memory access (RMA) via a high-speed network through a simple put/get model. We believe there are numerous other special communication capabilities which DVC's can express in a fashion that enables easy use, and underlying implementation.

The ability to collect sets of names allows novel communication semantics to be exposed. For instance, to establish a private network, a set of end-points can be declared as requiring a private connection and choose a private network technology to apply. In response, the DVC controller checks the possibility of such configuration and, if feasible, it manages to establish the private connection with the required properties. In case of creating a physically private network like a dedicated optical network, the DVC controller may need to contact the controller of the underlying network infrastructure to negotiate, allocate and reserve for network resources, including photonic switches and light paths. Note that DVC's separate configuration and execution, so DVC's can support legacy applications while exploiting on-demand optical network capabilities.

4.4 Security Management

Each DVC is private to a single or a group of users. Under a single privilege, a user has full control over the DVC. Security domains can be formed across a subset of resources in a namespace. Using DVC primitives, a user can define the level of security and trust amongst resources, including the network. Three security options for network trust are available: 1) trusted network and resources; 2) trusted network and untrusted resources; or 3) untrusted network and resources. Users may also also set finer grained access control for individual resources and security domains. Based on the configured trust relation, the DVC domain controller selects and implements appropriate security mechanisms prior to application runtime. During actual execution, the applications can assume a secure communicating environment, as comparable to a closed private network.

4.5 Resource Management

A DVC includes a resource management service, whose main tasks are to allocate/deallocate grid resources and to manage their utilization within the DVC environment. The DVC supports runtime libraries that application developers can use to interact with this service. These libraries allow the application developers to set various resource management policies for their computing environments and to explicitly allocate and manage individual resources. The resource management service is realized by the cooperation of the DVC and ghost managers. In the resource allocation process, the DVC manager serves as a resource broker that inputs a specification of desired resources from a user and subsequently discovers, reserves and binds the matching resources with the DVC. The DVC model supports an aggregation of any kind of network-enabled resources that allow creating of a process on them. Once a new resource is allocated, a lightweight-process ghost manager is created to run on it. The ghost managers enable a simple view of resources as they mask resource diversity and complexity and provide a consistent way to interact with them. Throughout application execution, the DVC manager and the ghost managers monitor utilization and availability of DVC resources. It appropriately adjusts the computing environment, allocating/deallocating DVC resources, in response to the dynamic application needs. Furthermore, DVC resources can be transparently replaced in case of resource failure or connection breakdown.

All these services shield application developers from interacting with remote resource providers and handling low-level management systems. Even though it requires an effort of a user in the resource configuration process, once constructed the DVC environment can be viewed as a collection of computing resources bound up under a single resource management domain, as in a typical local distributed environment.

4.6 Event Monitoring

To enable the development of robust applications on grids, DVC supports an event subscribe/notification and asynchronous message services. These services allow applications to respond to asynchronous changes in networks and remote resources or let the DVC controller handle them based on the previously agreed policy. In the first case, developers can request the DVC controller to send them and/or their applications a notification message on an occurrence of particular events. The user can also specify the tasks to be executed in response to these events. In the latter case, the DVC controller has ability to dynamically perform process migration and resource reallocation, as appropriate. These capabilities will help hide the unpredictable and dynamic nature of grid environments and present a view of reliable computing environments, as is our goal.

4.7 Job Execution Management

The DVC manager facilitates efficient and reliable execution of a user's jobs. It schedules jobs to run on

distributed resources under the DVC environment based on their priority, submission time and other applicable policy. It also mediates communication among remote jobs and supports job restart and migration in case of unexpected execution failure or resource unavailability.

When an application is started, the DVC manager selects computing resources from the DVC resource pool and binds them up with the application. When the application spawns computing tasks to run on remote resources, the DVC manager contacts the corresponding ghost managers to invoke processes on the chosen resources.

4.8 DVC Application Interface

A DVC provides a set of convenient routines and their interfaces for application developers to interact with the DVC domain controller and the DVC services. These include DVC configuration modules and runtime libraries we have discussed in the section. Besides DVC-related routines, a DVC supplies fundamental programming tools necessary for developing parallel and distributed applications on grids.

5. Example Distributed Virtual Computers

We discuss several use cases for DVC, describing their structures and potential advantages.

5.1 Virtual Cluster

In a typical cluster environment, a collection of computing resources are logically grouped under a single administrative domain and controlled by a centralized resource management system. These computing resources, such as computing nodes and disk storages, are generic and interchangeable, though they may be heterogeneous in capability.

DVC can allow traditional cluster computing models to be used on the grid. A cluster DVC can assemble a large set of distributed computing resources from multiple organizations, coupled with a high-speed wide-area network, and allow them to be centrally managed under a single security domain. Additionally, a cluster DVC can supply virtualization and naming services that hide the complexity of highly dynamic and heterogeneous resource environment and simulates the view of a typical resource cluster, coupled by a close, reliable network. Potential advantages include ease of use, and exploitation of geographically distributed resources for both larger-scale computing cluster and geographic failure tolerance.

5.2 Heterogeneous Resources (Virtual Collection)

For many grid applications, an assemblage of heterogeneous resources (e.g. special data, scientific instrument, virtualization devices, etc.) is useful. These resource aggregations are similar to clusters, but these resources are not always interchangeable because of their unique capabilities.

With DVC, a heterogeneous resource collection can be effectively and conveniently created on the grid for a group of users participating in a run. The virtual collection DVC would use the specification from the user to discover and select remote grid resources, and bind them into the DVC. The DVC also includes naming and virtualization services (see Section 4.2 and 4.5) that enable standard view of these distributed set of diverse resources, and to establish a complex secure communicating structure among them. Potential advantages include simplicity of use and convenient expression of a shared configuration for addressing various performance and security needs.

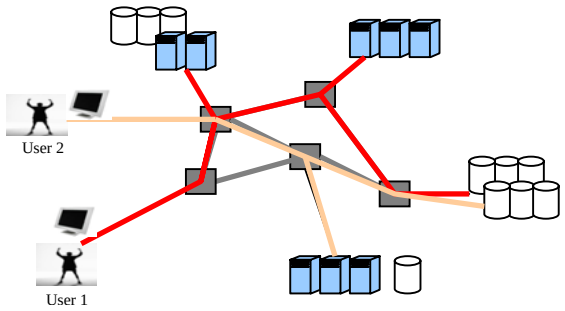

Figure 3. Two DVC computing environments established from two sets of dedicated lambda links

6. DVC's and Other Grid Technologies

The DVC model can be applied to all types of grids, but our initial focus is to employ it on lambda-based grids, where a collection of distributed resources are interconnected by dedicated dense wavelength division multiplexing (DWDM) optical paths (or lambda network). These communication channels are private and provide paths without routers or switches. As compared to a shared, packet-switched network, like open Internet, the lambda network allows higher-speed and more reliable data communication. The circuit-switched lambda can be configured on-demand to form private end-to-end and multi-endpoint networks, suitable for creating high-performance and secure computing environments. We anticipate DVC's as secure collections of computing and storage resources that span multiple administrative domains and are coupled by these high-speed dedicated optical connections. Figure 3 illustrates two DVC environments established from two lambda networks. In this scenario, two users allocate resources dynamically to form DVC's including computing resources from multiple sites. Within DVC's,

applications can utilize remote resources directly and securely with high-performance.

As discussed earlier, we are exploring several new primitives to expose novel communication capabilities provided by bandwidth-rich lambda networks. These include high-speed optical multicast and group communication. We believe that these primitives will be needed in the future and will enable a wide range of new high-performance distributed applications.

The DVC model relies on grid technologies in many areas, including resource management, security, communication, and data movement. We leverage existing grid middleware for basic resource access, but innovate to extract the maximum benefits from the opportunity of lambda-based grids. In our implementation of DVC's we exploit the Globus Toolkit [11] which provides fundamental grid services for resource discovery [12], remote resource allocation [13] and data movement [14]. It also defines Grid Security Infrastructure (GSI) [15] that provides standard mechanisms for authentication, authorization and secure job invocation. DVC's leverage these grid components to implement DVC abstractions and make more application-oriented services available to developers.

7. Related Work

A wide variety of work is relevant to DVC; here we briefly survey the most relevant.

PVM [16] is a portable message-passing system which provides a simple user-environment for distributed computing. Both DVC and PVM provide an abstraction layer that enables the development of distributed applications on diverse computer systems. Traditional PVM assumes a single administrative domain, and does not explicitly address security and unique communication capabilities. In contrast, DVC is designed to span multiple administrative domains, and also to support unique communication structures.

A Globus "Virtual Organization" [2] is a set of relationships and sharing policies that permit coordinated use of grid resources from multiple organizations by a community of users. However, its construction requires agreement of all participating organizations, so change is slow. The Community Authorization Service (CAS) [17] is a VO-enabled service that eliminates the needs of direct contact between resource providers, providing a community as a first class identity. The DVC model essentially assumes an existence of VO and CAS, but is a dynamic application instance oriented structure. It is easily instantiated within a VO by a single user, and comes and goes dynamically with single application runs.

A number of grid programming tools have emerged from traditional parallel and distributed computing paradigms. Tools, such as MPICH-G2 [18] and GridRPC [19], use the Globus services to operate in grid environments. These systems are limited in their capabilities due to their heritage. For instance, MPICH-G2 assumes a static environment and does not support secure inter-process communication. GridRPC doesn't support convenient use of process collections. In contrast, the DVC enable flexible construction and adaptation of groups of processes.

Condor-G [20] is a computation management system for compute-intensive jobs on multi-organizational grids. Both the DVC and Condor-G systems leverage the Globus system in harnessing grid resources and support hosting environments and job execution on remote resources. In Condor-G all resources share the same level of trust and communication is implicit, in direct contrast to the DVC model which enables description of complex multi-party communication and trust. Unlike DVC applications, those in Condor cannot take advantages of topology-aware and unique communication capabilities.

The European DataGrid [21] is developing and deploying grid middleware to support computation and management of large-scale scientific datasets. Their middleware makes use of Condor-G for submission and management of batch and interactive jobs. Parallel (MPI) jobs are allowed, but they run across local computer elements within an administrative domain. This is in direct contrast to the DVC model which spans multiple administrative domains. Furthermore, the interaction among jobs in DataGrid is only pairwise and asynchronous, and thus well matched to the underlying GSI capabilities.

The GrADS [22] and GridLab [23] projects share the same general goal as DVC – to simplify the development of grid applications. The GrADS project supports the building of configurable object programs, while the GridLab project aims to provide application-oriented grid services. These are innovative, ambitious efforts. The DVC takes a lower-level approach, providing only an execution environment which provides a simple view of a local distributed computing environment.

8. Summary and Future Work

We have presented DVC as a computing environment that provides a set of abstractions to simplify the development and execution of secure and robust applications on grids. Grid resources can be bound up within a DVC environment and expressed through a single administrative domain. This enables

applications to view their environment as a local private distributed computing environment with predictable computation performance. The major benefits arising from using DVC's include: simplified view of security and naming mechanisms as well as efficient runtime resource selection and binding.

Our effort in developing the DVC model is motivated by the novel communication capabilities provided by lambda circuit-switched networks. Specifically, the dynamic lambda circuits allow on-demand construction of bandwidth-rich dedicated networks that can effectively interconnect geographically distributed resources, forming high-performance and secure grid computing environments.

Our future work is to make the DVC design more concrete and to build a full implementation of DVC's. These implementations will build on existing grid services and a wide range of new protocol, optical circuit switching, real-time, and security technology being developed in the OptIPuter project [5].

Acknowledgements

Supported in part by the National Science Foundation under awards NSF EIA-99-75020 Grads and NSF Cooperative Agreement ANI-0225642 (OptIPuter), NSF CCR-0331645 (VGrADS), NSF NGS-0305390, and NSF Research Infrastructure Grant EIA-0303622. Support from Hewlett-Packard, BigBangwidth, Microsoft, and Intel is also gratefully acknowledged.

9. References

[1] I. Foster and C. Kesselman, editors, "The Grid: Blueprint for a New Computing Infrastructure," Morgan Kaufmann, 1999.

[2] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of Grid: Enabling Scalable Virtual Organizations," *International Journal of Supercomputer Applications*, 15(3), 2001.

[3] C. Lee, S. Matsuoka, D. Talia, A. Sussman, M. Mueller, G. Allen, and J. Saltz, "A Grid Programming Primer," Technical Report, Global Grid Forum Programming Model Working Group, August, 2001.

[4] C. Lee and D. Talia, "Grid Programming Models: Current Tools, Issues and Directions," Grid Computing: Making the Global Infrastructure a Reality, Wiley, pp.555-578, 2003.

[5] L. Smarr, A. Chien, T. Defanti, J. Leigh, and P. Papadopoulos, "The OptiPuter," in *Communications of the Association for Computing Machinery (CACM)*, 47(11), November 2003. http://www.optiputer.net/.

[6] B. Gleeson, A. Lin, J. Heinanen, T. Finland, G. Armitage, and A. Malis, "A Framework for IP Based Virtual Private Network," RFC 2764, February 2000.

[7] R. Wu and A. Chien, "GTP: Group Transport Protocol for Lambda-Grids," to appear in *Proceedings of the 4th IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2004.

[8] E. He, J. Leigh, O. Yu, and T. Defanti, "Reliable Blast UDP: Predictable High Performance Bulk Data Transfer," in *Proceedings of IEEE International Conference on Cluster Computer*, 2002.

[9] Leigh et al, "An Experimental OptIPuter Architecture for Data-Intensive Collaborative Visualization," in *Proceedings of the 3rd Workshop on Advanced Collaborative Environments*, 2003.

[10] C. Zhang et al, "Terascope: Distributed Visual Data Mining of Terascale Data Sets over Photonic Networks," *Journal of Future Generation Computer System* 19, pp. 935-944, Aug. 2003.

[11] The Globus Toolkit. http://www-unix.globus.org/toolkit/.

[12] K. Czajkowski et al, "Grid Information Services for Distributed Resource Sharing," in *Proceedings of the 10th IEEE International Symposium on High-Performance Distributed*, 2001.

[13] K. Czajkowski et al, "A Resource Management Architecture for Metacomputing Systems," In *Proceedings of the forth Workshop on Job Scheduling Strategies for Parallel Processing*, 1998.

[14] J. Bester, I. Foster, C. Kesselman, J. Tedesco, and S. Tuecke, "GASS: A Data Movement and Access Service for Wide Area Computing Systems," in *Proceedings of the sixth Workshop on I/O in Parallel and Distributed Systems*, 1999.

[15] I. Foster, C. Kesselman, G. Tsudik, S. Tuecke, "A Security Architecture for Computational Grids," in *Proceedings of the fifth ACM Conference on Computer and Communication Security Conference*, 1998.

[16] A. Geist, et al, "PVM Parallel Virtual Machine: A Users' Guide and Tutorial for Networked Parallel Computing," the MIT Press, 1994.

[17] L. Pearlman, V. Welch, I. Foster, and C. Keselman, "A Community Authorization Service for Group Collaboration," in *Proceedings of IEEE Workshop on Policies for Distributed Systems and Networks*, 2002.

[18] N. Karonis, B. Toonen, and I. Foster, "MPICH-G2: A Grid-Enabled Implementation of the Message Passing Interface," *Journal of Parallel and Distributed Computing (JPDC)*, 63(5), pp. 551-563, May 2003.

[19] H. Nakada, S. Matsuoka, K. Seymour, J. Dongarra, C. Lee, and H. Casanova, "GridRPC: A Remote Procedure Call API for Grid Computing," in *Proceedings of the third International Workshop on Grid Computing*, 2002.

[20] J. Frey et al, "Condor-G: A Computation Management Agent for Multi-Institutional Grids," in *Proceedings of the 10th IEEE Symposium on High Performance Distributed Computing*, 2001.

[21] F. Gagliardi at el, "European DataGrid Project: Experiences of Deploying a Large Scale Testbed for E-Science Applications," in *Proceedings of Performance Evaluation of Complex Systems: Techniques and Tools (Performance 2002 Conference)*, 2002.

[22] F. Berman et al, "The GrADS Project: Software Support for High-level Grid Application Development," *Internal Journal of High Performance Computing Applications*, 15(4), pp. 327-344, 2001.

[23] G. Allen et al, "Enabling Applications on the Grid – A GridLab Overview," *Internal Journal of High Performance Computing Applications*, Aug 2003.