# Integration of High-Performance Computing into Cloud Computing Services

Mladen A. VOUK[a, 1], Eric SILLS [a], Patrick DREHER[b]
[a]*North Carolina State University, Raleigh, NC 27695, USA*
[b]*Renaissance Computing Institute, Chapel Hill, NC 27517, USA*

**Abstract.** High-Performance Computing (HPC) services can range from time on peta-flop supercomputers, to access to high-end tera-flop facilities running a variety of operating systems and applications, to mid-range and smaller computational clusters used for HPC application development, pilot runs and staging. What they all have in common is relative isolation - that is traditionally HPC facilities have tended to be isolated from the more general scientific computing operations. Advent of the cloud computing concept has changed that. Even the most avid supporters of HPC and Grid computing are beginning to admit that almost all loosely coupled HPC computing, and a lot of tightly coupled HPC computing, can be done in a cloud. In this article, we will discuss a very successful production-level architecture and policy framework for supporting HPC services within a more general cloud computing infrastructure. This integrated environment has been operating at NC State since fall 2004. It typically delivers over 7,200,000 HPC CPU hours per year to NC State faculty and students. In addition, we present and discuss operational data that show that integration of HPC and non-HPC services in a cloud can substantially reduce the cost of delivering cloud services (down to cents per CPU hour).

## 1 - Introduction

Figure 1. shows a snapshot of a Google trends analysis for keywords: Cloud Computing, High-Performance Computing and Grid Computing. Vertical axis shows the search volume and news reference volume. We see that the popularity of "Grid Computing" has been diminishing for the last six years, that of high-performance computing has been steady (but low), and that cloud computing did not really have any significant visibility until end of 2007. However, after about October 2007 when Google and IBM announced "cloud computing" research directions [Loh07] and IBM announced its cloud computing initiative [IBM07] the interest in the concept started rising and has not waned yet as witnessed by numerous articles, a growing number of conferences, and an increasing use of the term to describe old, existing and some new solutions and services.

*"The concept of cloud computing has become a popular term to describe a flexible system that provides users with access to hardware, software, applications and services. Because there is no one generic user and the hardware, software, and services may be*

[1] Corresponding Author: M. Vouk, Department of Computer Science, Box 8206, North Carolina State University, Raleihg, NC 27695, USA, vouk@ncsu.edu.

*grouped in various combinations, this cloud computing concept quickly fractures into many individualized descriptions and perspectives. As a result, it is very difficult to agree on one common definition of cloud computing"* [Dre09, see also McK09, Amr09, Vou09].

In the context of this chapter, we consider "cloud computing" to refer to a seamless component-based architecture that can deliver an integrated, orchestrated and rich suite of both loosely and tightly coupled on-demand information technology functions and services, significantly reduce overhead and total cost of ownership and **services**, and at the same time empower the end-user in terms of control. Some more obvious advantages of cloud computing are server consolidation, hardware abstraction via virtualization, better resource management and utilization, service reliability and availability, improved security and cost effectiveness, etc.

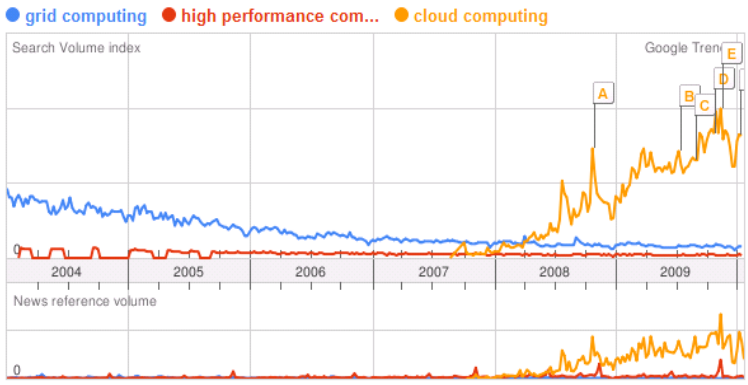

Figure 1. The rise of Cloud Computing

The concept of cloud computing is changing the way we think of information technology (IT) infrastructure in businesses, education, research and government, and as a result there has been a rapid increase in the number and types of cloud computing systems that are being deployed. While for most part organizations are still debating [e.g., Arm09, Gol09] how this technology might be applied, some organizations, such as NC State University, have been providing cloud-based services to their students, faculty and staff with great success since 2004 [Vou09, Dreh09, Vouk08a,b, Ave07, Sea10].

In many instances cloud computing tends to be equated to delivery of a single category of services, such as desktop services, or specific server functionalities, or specific applications or application environments. Where NC State's cloud computing environment, called Virtual Computing Laboratory (VCL, http://vcl.ncsu.edu) differs from other cloud computing implementations (and interpretations) is that it offers capabilities that are very flexible and diverse ranging from Hardware-as-a-Service all the way to highly complex Cloud-as-a-Service. These capabilities can be combined and offered as individual and group IT services, including true High-Performance Computing (HPC) services. Our VCL-HPC service very successfully integrates HPC into the cloud computing paradigm by managing not only resource capabilities and capacity, but also resource topology, i.e., appropriate level of network/communication coupling among the resources.

In the remainder of the chapter we describe NC State University (NC State or NCSU) cloud computing environment (Section 2), we describe its HPC environment and how it integrates into our cloud solution (Section 3), and we discuss performance and economics of the solution (Section 4). Summary is provided in Section 5.

## 2. NC State University Computing Cloud Computing

A cloud computing system should be designed around a service-oriented architecture (SOA) that can allocate resources on-demand in a location and device independent way, incorporate technical efficiency and scalability through appropriate level of centralization and sharing of cloud resource and control functions, and through explicit or implicit self-provisioning of resources and services by users to reduce administration overhead [Vou09, Dre09]. The principal difference between "traditional" and cloud computing is in the level of control delegated to the user. For example, in a traditional environment control of resource use and management lies primarily with the service delivery site and provider, in a cloud environment this control is for most part transferred to users in the form of self-provisioning options and appropriate privileges. Similarly, other traditional IT operations such as operating system and environment specification and mode of access and prioritizations now become explicit user choices. While this can increase management efficiency and reduce provisioning costs, the initial base-line set-up of the cloud is much more complicated and requires much more sophisticated technological expertise, management and security. In our experience, a flexible and versatile cloud environment needs to provide a range of differential services from Hardware-as-a-Service all the way to Cloud-as-a-Service and Security-as-a-Service.

In the context of NC State's VCL we distinguish

- Hardware as a Service (HaaS) – On-demand access to a specific computational, storage and networking product(s) and/or equipment configuration possibly at a particular site
- Infrastructure as a service (IaaS) - On-demand access to user specified hardware capabilities, performance and services which may run on a variety of hardware products
- Platform as a service (PaaS) - On-demand access to user specified combination hypervisors (virtualizations), operating system and middleware that enables user required applications and services running on either HaaS and/or IaaS
- Application as a Service (AaaS) - On-demand access to user specified application(s) and content. Software as a Service (SaaS) may encompass anything from PaaS through AaaS
- Higher level services - A range of capabilities of a cloud to offer a composition of HaaS, IaaS, PaaS and AaaS within an envelope of particular policies, such as security policies – for example Security-as-a-Service. Another example are composites and aggregates of lower-level service such as a "Cloud-as-a-Service" – a service that allows a user to define sub-clouds (clusters of resources) that the user controls in full.

All of the above services are at some level available to NC State VCL users with different privileges [Vou08, Vou09]. HaaS and IaaS are essential if one wishes to construct high-performance computing (HPC) service with a particular topology, or to have the ability to deliver specific end-to-end quality of service, including application performance. We find that a carefully constructed cloud computing implementation that offers the basic services listed above can result in good technical performance and increased productivity regardless of whether the cloud is serving commercial or educational institutions.

Virtual Computing Laboratory (VCL, http://vcl.ncsu.edu) is a *high performance open-source award-winning*[2] ***cloud computing*** *technology* initially conceived and protyped in 2004 by NC State's College of Engineering, Office of Information Technology, and Department of Computer Science. Since then, VCL development has rapidly progressed in collaboration with industry, higher education, and K-12 partners to the point that today it is a large scale, production-proven system which is *emerging as a dominant force in the nascent and potentially huge open-source private-cloud market* [Sae10, Sch10, Vou09].

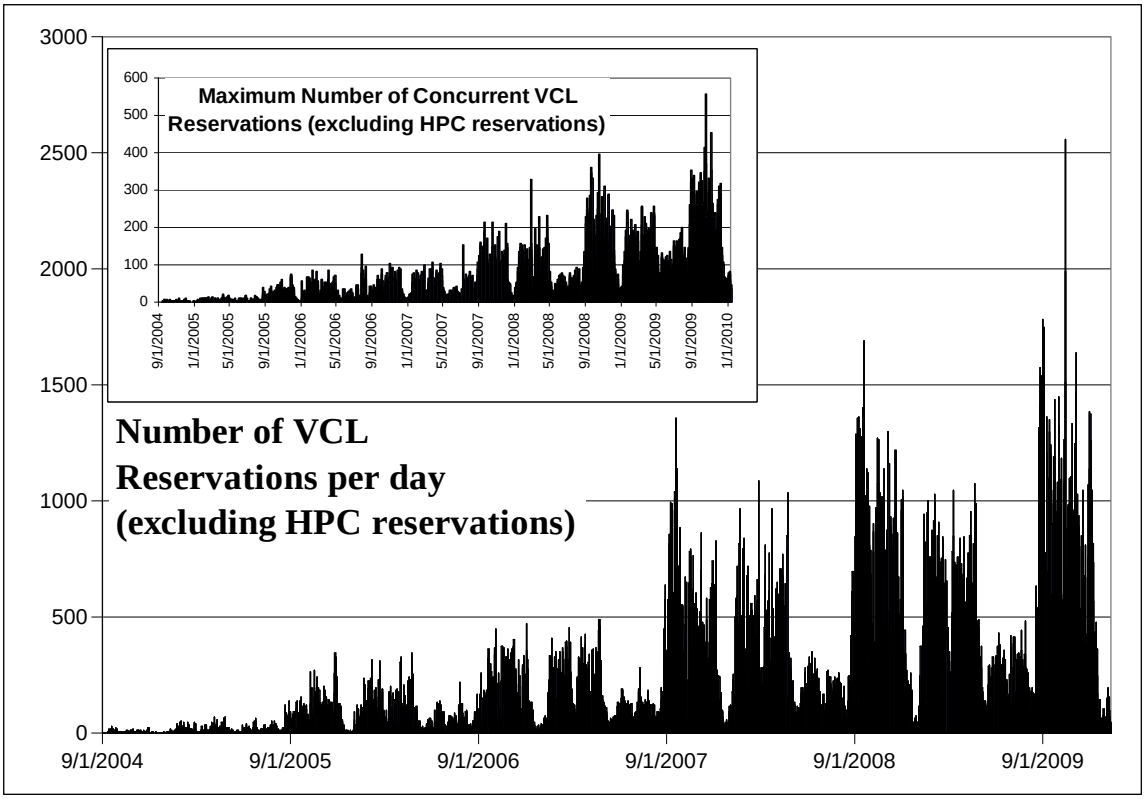


Figure 2. VCL usage

Campus use of VCL has expanded exponentially over the last five years (Figure 2). We now have over 30,000+ users and deliver over 100,000 reservations per semester through over 200 service environments, as well as over 7,200,000 HPC CPU hours annually. In-state initiatives include individual UNC-System universities (e.g., ECU, NCCU, UNC-CH, UNCG, WCU - technically all UNC System campuses which implement Shibboleth

[2] 2007 Computerworld Honors Program Laureate Medal (CHPLM) for Virtual Computing Laboratory (VCL), 2009 CHPLM for NC State University Cloud Computing Services

authentication have access to VCL), the NC Community College System (production and pilots in 15 colleges: Beaufort County, Brunswick, Cape Fear, Catawba Valley, Central Piedmont, Cleveland, Edgecombe, Fayetteville Tech, Forsyth Tech, Guilford Tech, Nash, Sandhills, Surry, Wake Tech), and several K-12 pilots and STEM initiatives.

Regional, national and international interest in VCL has also increased over the past year since VCL has been available through the Apache Software Foundation [VCL10]. Pilots are in progress all over the world. George Mason University (GMU) has become a VCL leader and innovator for the Virginia VCL Consortium, recently winning the 2009 Virginia Governor's award for technology innovation. There are VCL initiatives in a number of other states. For example Southern University Baton Rouge, and California State University East Bay are in the process of implementing VCL-based clouds.

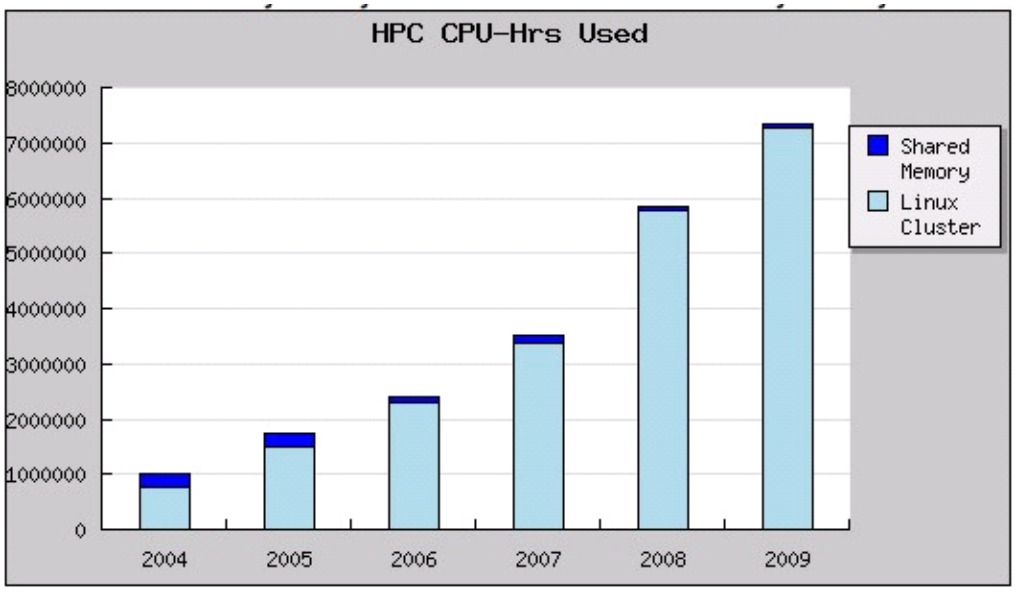


Figure 3. NC State HPC usage over years.

VCL's typical base infrastructure (preferred but not required) is an HPC blade system. The reason for that will become apparent shortly. System's capability can be delivered as a whole or "sliced and diced" dynamically into smaller units/clusters of capability appropriately "packaged" to meet a set of highly individualized requirements – services range from single desktops, to groups of "seats" in classrooms, to servers and server farms, to research clusters and sub-clouds, to true high-performance computing – from hardware, infrastructure and platforms as a service, to different levels of software and application as a service.

Figure 2 shows the number of VCL reservations made per day by users over last five years. This includes reservations made by individual students for Linux or Windows XP desktops along with some specific applications, but also reservations that researchers may

make to construct their own computational sub-clouds, or specialized resource aggregations – including high-performing ones. Figure 2 inset is the number of such concurrent reservations per day. What is not included in the counts shown in these figures are the reservations that deliver standard queue-based VCL HPC services. We therefore call these service non-HPC services (although self-constructed high-performance sub-clouds are still in this category of services). NC State's VCL currently has about 2000 blades distributed over three production data centers and two research and evaluation data centers. About one third of the VCL blades are in this service delivery mode (the one we call non-HPC mode), some of the remaining blades are in our experimental test-beds and in maintenance mode, and the rest (about 600 to 800) operate as part of the VCL-HPC (http://hpc.ncsu.edu) and are controlled through a number of LSF[3] queues.

There three things to note with reference to Figure 2. One is that the usage of VCL, and by implication of the NC State Cloud Computing environment, has been growing steadily. The second is that the resource capacity (virtual or bare-machine loaded) kept on the non-HPC side at any one time is proportional to the needed concurrency. The third thing to note is that non-HPC usage has clear gaps. VCL reservations tend to go down during the night, and during student vacations and holidays. On the other hand, if one looks at the NC State demand for HPC cycles we see that it has been growing (Figure 3), but we also see that demand is much less subject to seasonal variations (Figure 4).

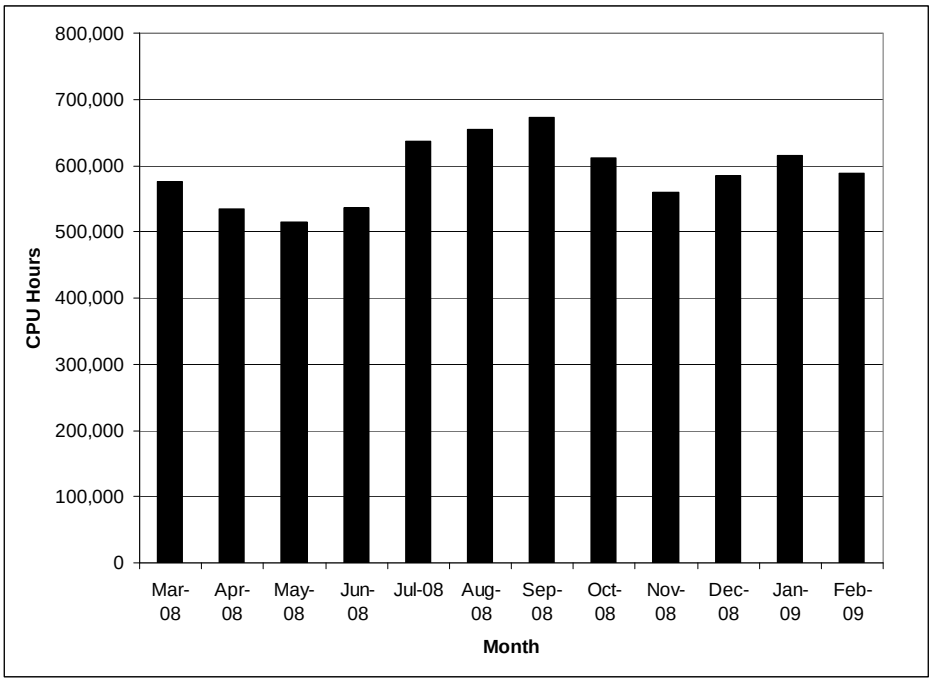


Figure 4. NC State HPC usage over March 2008 – February 2009.

An in depth analysis of the economics of cloud computing [Dre09] shows one of the key factors is average utilization level of the resources. Any of our resources that are in VCL-HPC mode are fully utilized (with job backlog queues as long as two to three times the

[3] http://www.platform.com/Products

number of jobs running on the clusters). In 2009, even with maintenance down time, VCL-HPC recorded over 7.2 million HPC CPU-hours – in the 95+% utilization range. On the other hand, while in the same period desktop augmentation and similar non-HPC usage recorded over 180,000 reservations, it also recorded only about 490,000 CPU-hours – about 10 to 15% utilization.

To satisfy high demand for HPC, allow for peak non-HPC use, and generally balance the workloads, VCL was designed so that during the times when augmentation (non-HPC) use is lower, such as during summer holidays, VCL can automatically move the resources into its HPC mode where they are readily used. When there is again need for non-HPC use these resources are, again automatically, moved back in that use pool. As a result, the combined HPC/non-HPC resource usage runs at about 70% level. Obviously a cost-effective and desirable strategy, but also one that requires an active collaboration of the underlying (cloud) components. VCL offers that capability.

## 3. Integrating High-Performance Computing

So how do we do that? The first level architecture of VCL is shown in Figure 5. A user accesses VCL through a web interface to select from a menu a combination of applications, operating systems and services she needs. If a specific combination is not already available as an "image" – either a bare-metal or a virtual machine service environment consisting of the operating system (with possibly a hypervisor), middleware, application stack along with security management, and access and storage tools and agents -, authorized user can construct one's own from the VCL library components. This customization capability is very much in the spirit of what services engineering and management is all about. VCL manager software then maps that request onto available software application images and available hardware resources and schedules it for either immediate use (on demand) or for later use. VCL manager software was developed by NCSU using a combination of off-the-shelf products (such as IBM xCAT[4]) and in-house developed open-source "glue" software (about 50,000+ lines of code – now available through Apache [VCL10]).

All components of the VCL can be (and are) distributed. A site installation will typically have one resource management node (Node Manager in the figure) for about 100 physical blades. This ensures adequate image load times as well as resource fail-over redundancy. In the context of our architecture we distinguish undifferentiated resources – resources that are completely malleable and can be loaded with any service environment designed for that platform, and differentiated resources – resources which can be used "as is" only without any modifications beyond what the user access permissions allow. For example, during the night VCL users are allowed to use NC State computing lab Linux machines remotely, but they are not allowed to modify their images since lab machines are considered a differentiated resource. In contrast, when they make a "standard" VCL Linux or Windows reservation, students get full root/administrator privileges and can modify the image as much as they wish. However, once they are finished (typically our

[4] http://xcat.sourceforge.net/

student reservations last 1 to 2 hours) that resource is "wiped" clean and a fresh image is reloaded. Students save their data either onto our network-attached corporate storage or on their own laptops.

In this fashion VCL provides the ability to deliver scheduled and on-demand, virtualized and "bare-metal" infrastructures, and differentiated and undifferentiated services consistent with established NCSU policy and security requirements which may vary by user and/or application. VCL dynamically constructs and deconstructs computing environments thereby enabling near continuous use of resources. These environments, consisting of intelligent software-stack image(s) and metadata specifications for building the environment, can be created, modified, and transferred as policy and authorization permit. VCL security capabilities enable wide latitude in the assignment of these permissions to faculty, staff, and students.

VCL code version 2.x has been available for about a year as part of the Apache offering [VCL10]. Version 2.x of VCL represents a major rewrite of the base code and moves VCL to a modular software framework in which functional elements can be modified, replaced or optioned without attendant code changes elsewhere. It greatly empowers both community contribution and non disruptive customization. In addition, its already excellent security profile is being constantly enhanced through a federally funded Secure Open Source Initiative (http://sosi.ncsu.edu).

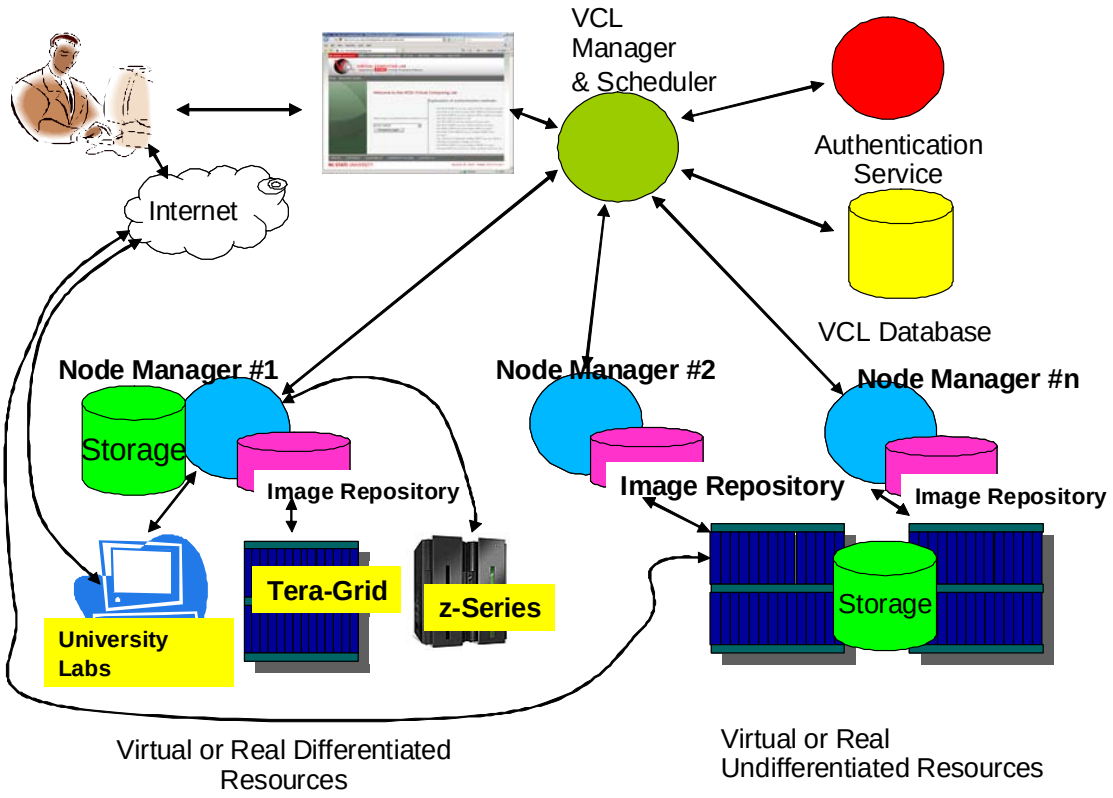


Figure 5. Top level VCL architecture [Vou09].

VCL environments are stored in on-line repositories, providing a low-cost high-volume retention capability that not only supports extreme scaling of access and reuse but also enables the breath of scale and scope required for intelligent real-time sequencing of multi-stage workflows. The benefit of this advance varies depending on the limitations of use imposed by software licensing agreements. Absent these limitations VCL empowers a new paradigm of *build once well and pervasively reuse*. In fact, one of the special characteristics of VCL is that its provenance and meta-data collection is sufficiently fine-grained and thorough that it allow very detailed metering of the software usage – by user, by department, by duration, by location, etc. That in itself offers an opportunity to implement a metering-based license management model that, given appropriate vendor agreements, allows porting and exchange of service environments across/among clouds.

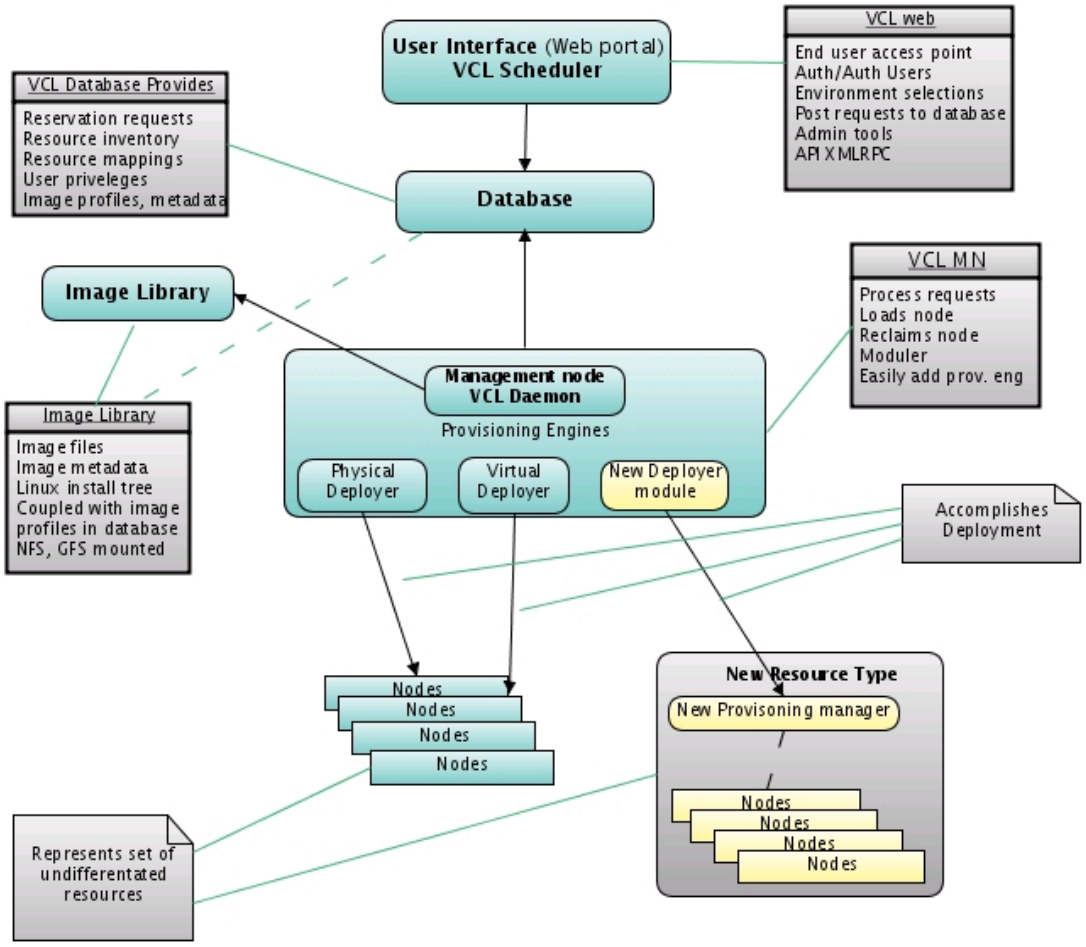


Figure 6. VCL architecture – internal details (http://cwiki.apache.org/VCL/vcl-architecture.html)

### 3.1 Internal Structure

Figure 6 shows more of the internals of the VCL architecture. At the heart of the solution are a) a user interface (including a GUI and a remote service API), b) an authorization,

provenance and service tracking data-base, c) a service environment "image" library, and d) a service environment management and provisioning engine.

Provisioning engine deploys service environments on demand either at the physical layer (bare-machine loads are typically done using xCAT, but other loaders can be used, including IBM Director), or at a virtual layer (e.g., VMWare, Xen, KVM "images"), or through a specially defined mechanism, e.g., a new service interface, a remote service, a service API to another cloud. Deployed service environments can consist of a single bare-metal image or a virtual software stack "image," such as a Windows or Linux desktop loaded onto undifferentiated resources, or it can consist of a collection of images (including their interconnection topology) loaded onto a set of (possibly) hybrid resources, or it can consist of set of HPC images loaded onto VCL resources being moved into differentiated HPC mode, and so on. In the NC State implementation, physical server manager loads images to local disk via kick-start (only Linux environments), copies images to local disk (Linux and Windows), or loads images to local memory (stateless). For loading of virtual machine images, VCL leverages command-line management tools that come with hypervisors.

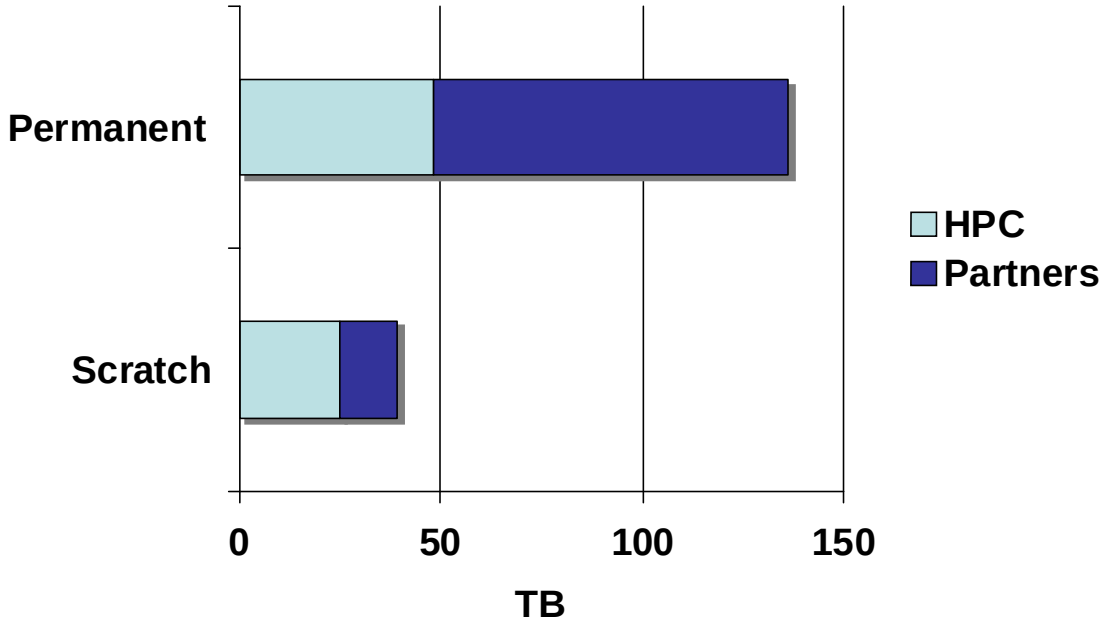


Figure 7. NC State (baseline VCL-HPC services) and Partner storage.

**Storage**. Where information is stored in the cloud is very important. Secure image storage is part of the core VCL architecture, however end-user storage is more flexible. It can range from storage on the physical VCL resource, to secure NAS or SAN accessed via the storage access utilities on the image itself, to storage on the end-user access station (e.g., laptop storage or memory key storage), etc. At NC State most of our images are equipped with agents that can connect in a secure way to our corporate storage (AFS based) and thus access backed-up storage space that students and faculty are assigned, and tools (such as visual sftp) that can access other on-line storage. Our HPC images are

constructed so that they all have access to HPC scratch and permanent storage via NFS. Figure 7 illustrates the current extent of that storage. It is interesting to comment on the part of the storage marked as “Partners”.

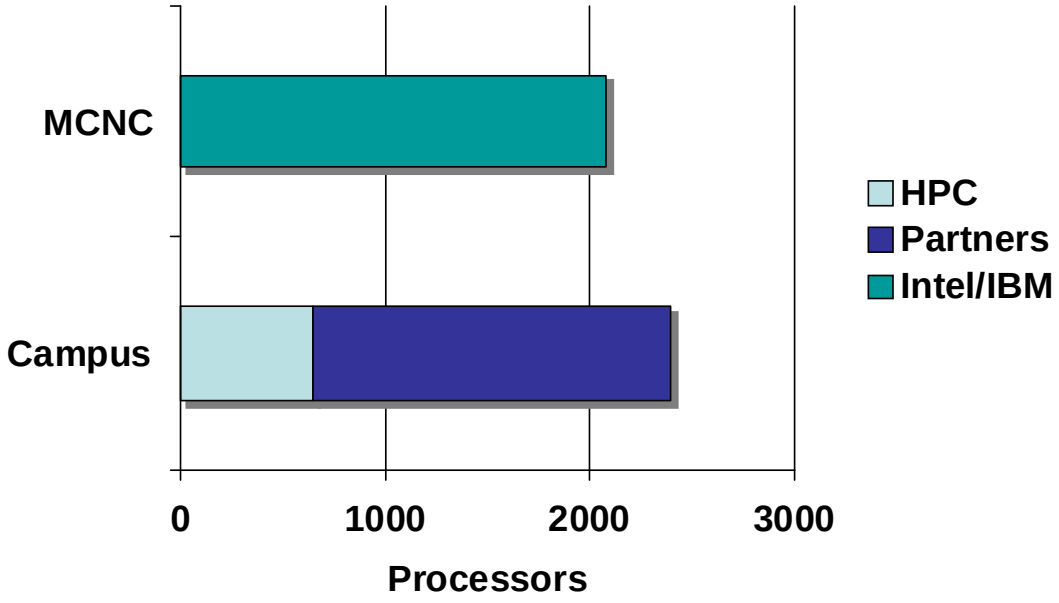


Figure 8. NC State VCL-HPC distributed memory computing resources (HPC), Partner computational resources and resources acquired through gifts and joint projects with IBM and Intel.

**Partner’s Program.** NC State researchers have the option of purchasing VCL-HPC compatible hardware and any specialized or discipline-specific software licenses. NC State Office of Information Technology (OIT) provides space in an appropriate and secure operating environment, all necessary infrastructure (rack, chassis, power, cooling, networking, etc.), and the system administration and server support. In return for infrastructure and services provided by OIT, when partner compute resources are not being used by the partnerm they are available to the general NC State HPC user community. This program has been working very well for us as well as for our researchers. As can be seen from Figure 7 and Figure 8, a large fraction of our HPC resources are partner resources.

### 3.2 Access

As part of our cloud HPC services we have both distributed memory (typically IBM BladeCenter clusters) and shared memory computing resources (typically 4-socket quad core Opteron servers with at least 2 GB of memory per core). We also provide items such as resource manager/scheduler, compilers, debuggers, application software, user training and support, consulting, code porting and optimization help, algorithm development support, and general collaboration. There are two ways of reaching those resources – through VCL-based reservation of one’s own login node, or through the use of a communal login node. Personal login nodes make sense if end-users wish to monitor their

runs in real time. One submits jobs in the usual fashion using job queues, in our case controlled via LSF. Queue priority depends on the resources requested and partnership privileges. Partners get absolute and immediate priority on the resources they own (or an equivalent set of resources), and they get additional priority towards adding common resources beyond what they own.

**Standard**. All VCL-HPC resources run the same HPC service environment (typically RedHat-based), and have access to a common library of applications and middleware. However, users can add their own applications to the computational resources they are given access to. All our standard VCL-HPC nodes are bare-metal loaded for sole use on VCL blades. They are managed as differentiated resources, i.e., users have full control over them, but they cannot re-load them or change them (except for the software in the user's home directories), and they must be used with the NC State maintained scheduler and file system. Most of our HPC nodes operate in this mode and as such they are very similar to any other HPC cluster. Nodes are tightly coupled with 1Gbps or better interconnects. A user with sufficient privileges can select appropriate run queues that then map the jobs onto the same BladeCenter chassis or same rack if that is desired, or onto low latency interconnects (e.g., Infiniband interconnected nodes).

**Special needs**. If a user does not wish to conform to the "standard" NC State HPC environment, a user has the option of requesting the VCL cloud to give him/her access to a customized cluster. In order to do that, the user needs to have "image creation" privileges [Vou08, Vouk09], and the user needs to take ownership of that sub-cluster or sub-cloud service environment. First the user creates a "child image" – a data node or computational node image - running operating system of their choice as well as tools that allow communications with the cluster controller and access to the data exchange bus of their choice, e.g., NFS-based delivery of directories and files. The user saves that image.

Then the user creates a "parent image" in the VCL-cloud aggregation mode. Again the user picks the base-line operating system, and adds to it a cluster controller, such as an HPC scheduler of choice, e.g., PBS, or a cloud controller such as Hadoop's controller, or similar. Now the user attaches to this any number of "child images". Typically child images are of the same type, e.g., a computational HPC Linux image, if the user wishes to operate a homogenous cluster. But, the user can also attach different child images, say 20 computational Linux images, one Linux-based data-base image, one Windows web-services image, and so on. Then the user saves the "parent image". From now on, when the user loads the "parent or control image" all the children are also loaded into virtual or bare-machine resource slots, depending on how the child-images were defined. All Linux-based child images that are part of such a VCL aggregate know about each others IP numbers through a VCL placed /etc file. "Parent image" control software needs to know how to access that information and communicate with the children.

Default custom topology is random and loosely coupled, i.e., VCL maps the "parent" or anchor image, and its children onto resources on which the images can run, but it does not pay attention to inter-node latency or topology. If tight, low latency, inter-image communication coupling is desired, and the image owner has appropriate privileges,

mapping of the images onto nodes that conform to a particular topology or interconnect latency is possible.

### 3.3 Computational/Data Node Network

There are some important differences between the "standard" queue-based batch-mode VCL-HPC offering and a user-constructed user-owned cloud or HPC cluster. Following [Vou09]:

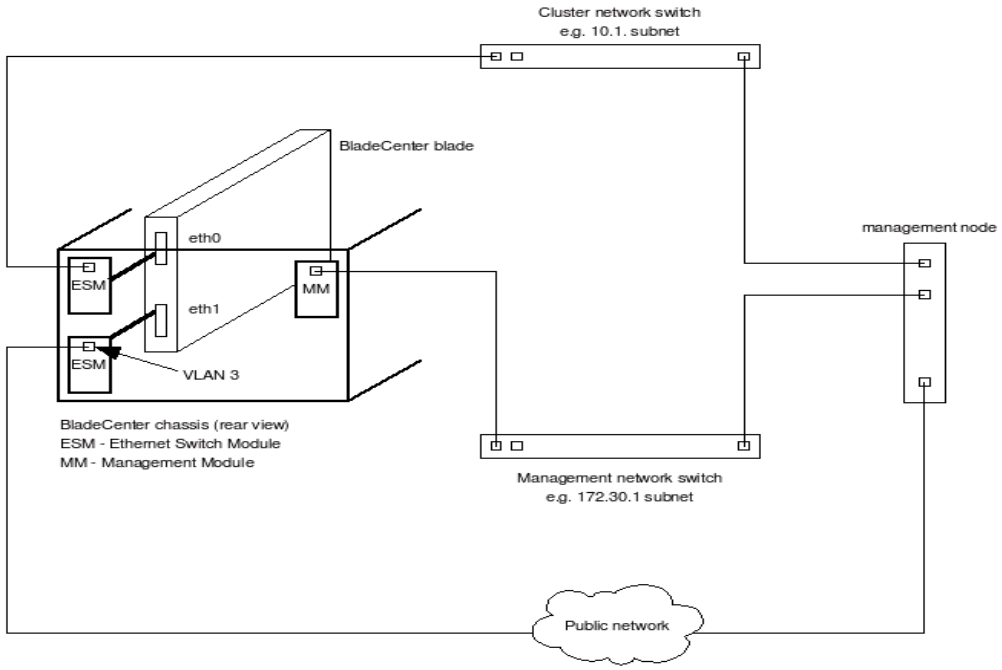


Figure 9. Undifferentiated resource node network configuration.

"One of the key features of the undifferentiated VCL resources is their networking set-up. It allows for secure dynamic reconfiguration, loading of images, and for isolation of individual images and groups of images. Every undifferentiated resource is required to have at least two networking interfaces. One on a private network, and the other one on either public or private network depending on the mode in which the resource operates. Also, for full functionality, undifferentiated resources need to have a way of managing the hardware state through an external channel –for example through the BladeCenter™ chassis Management Module."

"Figure 9 illustrates the configuration where seats/services are assigned individually or in synchronized groups, or when we want to assign/construct an end-user image aggregate or environment where every node in the environment can be accessed from a public network (e.g., an image of a web server, plus an image of a data-base, plus an image of an application, or a cluster of relatively independent nodes). Typically eth0 interface of a blade is connected to a private network (10.1 subnet in the example) which is used to

load images. Out-of-band management of the blades (e.g., power recycling) is effected through the management network (172.30.1 in the example) connected to the MM interface. The public network is typically connected to eth1 interface. The VCL node manager (which could be one of the blades in the cluster, or an external computer) at the VCL site has access to all three links, that is it needs to have three network interfaces. If it is a stand-alone server, this means three network interface cards. If management node is a blade, the third interface is virtual and possibly on a separate VLAN."

"It is worth noting that the external (public) interface is on a VLAN to provide isolation (e.g., VLAN 3 for the public interface in Figure 9). This isolation can take several levels. One is just to separate resources, another one is to individually isolate each end-user within the group by giving each individual resource or group of resources a separate VLAN – and in fact end-to-end isolation through addition of VPN channels. This isolation can be effected for both real and virtual hardware resources, but the isolation of physical hardware may require extra external switching and routing equipment. In highly secure installations it is also recommended that both the private network (eth0) and the MM link be on separate VLANs. Currently, one node manager can effectively manage about 100 blades operating in the non-HPC mode."

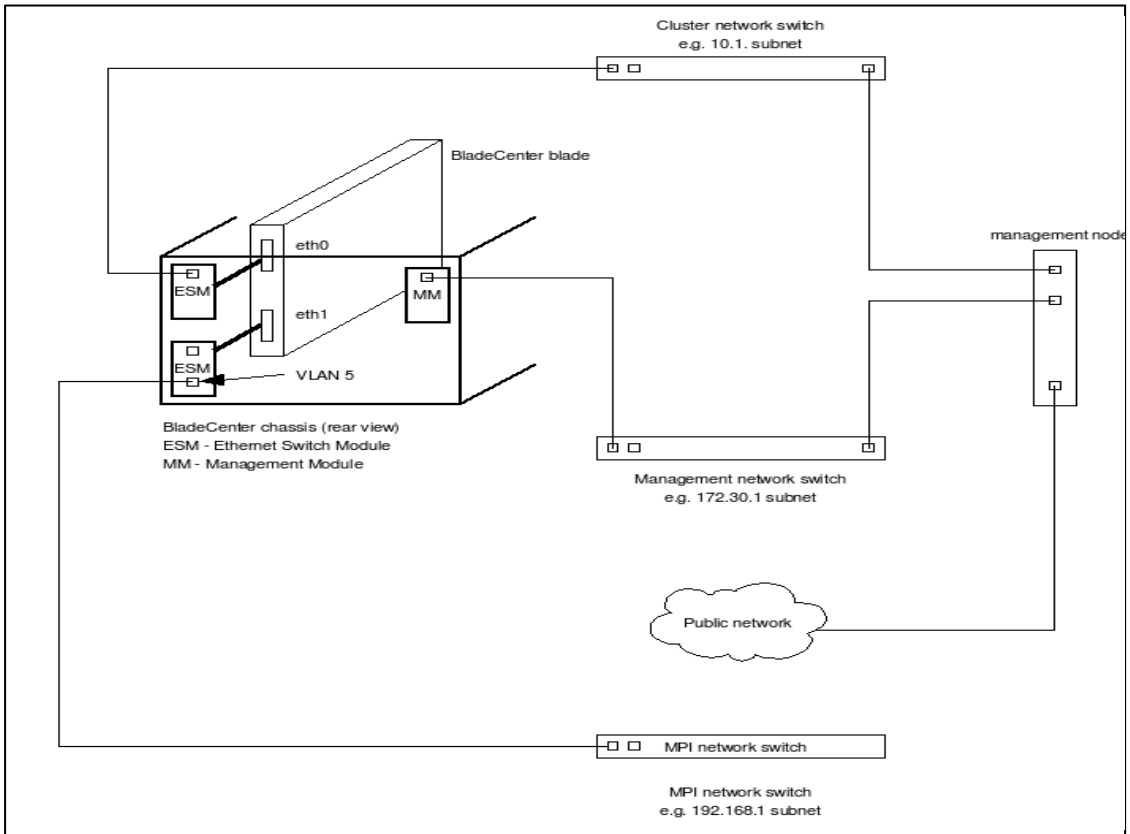


Figure 10. Network set-up for a "standard" VCL-HPC node.

"The second configuration (Figure 10) is used when the blades are assigned to a tightly coupled VCL-HPC cluster environment, or to a large overlay (virtual) "cloud" that has relatively centralized public access and computational management. In this case the node manager is still connected to all three networks – public, management and image-loading and preparation private network, but now eth1 is connected (through VLAN manipulation, VLAN 5 in Figure 10) to what has now become an Message Passing Interface (MPI) network switch. This network now carries intra-cluster communications needed to effect tightly couple computing tasks usually given to an HPC cloud. Switching between non-HPC mode and this HPC mode takes place electronically, through VLAN manipulation and table entries; the actual physical set-up does not change. We use two different VLANs to eth1 to separate Public Network (external) access to individual blades when those are in the Individual-Seat mode (VLAN 3 in Figure 9), from the MPI communications network to the same blade when it is in the HPC mode (VLAN 5 in Figure 10)."

### 3.4 Build Your Own

VCL code is available from apache.org [VCL10]. While the current version of VCL can operate on any X86 hardware, we have been using primarily IBM's BladeCenters because of their reliability, power savings, ease of maintenance, and compact footprint.

When building a "starter" non-HPC version of VCL one could limit the installation to virtual environments only, i.e., all resources operate as VMWare servers, and VCL controls and provisions only virtual images using the virtual version of the VCL non-HPC configuration in Figure 9. This is quick and works well, but is probably less appealing for those who wish to have the HPC option. One reason is performance. HPC community is still wary of having true HPC computations run on virtualized resources. Furthermore, some of the large memory and CPU footprint engineering applications also may not behave best in a virtualized environment. In those cases, installation of VCL's bare-machine load capabilities (via XCat) is recommended.

A small starter blade-based configuration is illustrated in Figure 11. Hardware can be housed in a single BladeCenter chassis. Two Ethernet switch modules are required to accommodate the public and private networks. If more than 7 blades are used, it is necessary to also have additional internal power supplies. Chassis network module is needed to connect the management node to storage – for example via fiber channel (optical pass through) or via iSCSI (copper pass through). A single chassis will house from 2 to 14 blades. At least one blade needs to be configured to attach to external storage for service environment image library function. That same blade could be designated to house the VCL scheduler, database and management daemon, while the rest of the blades would then be delivering VCL services. Storage is typically external. Several terabytes of storage are needed for the image repository, and additional storage needs to be network-accessible for support of HPC activities. Figures 12 and 13 illustrate how the installation can be scaled and what is needed to allow it to operate in both the HPC mode and in non-HPC cloud mode.

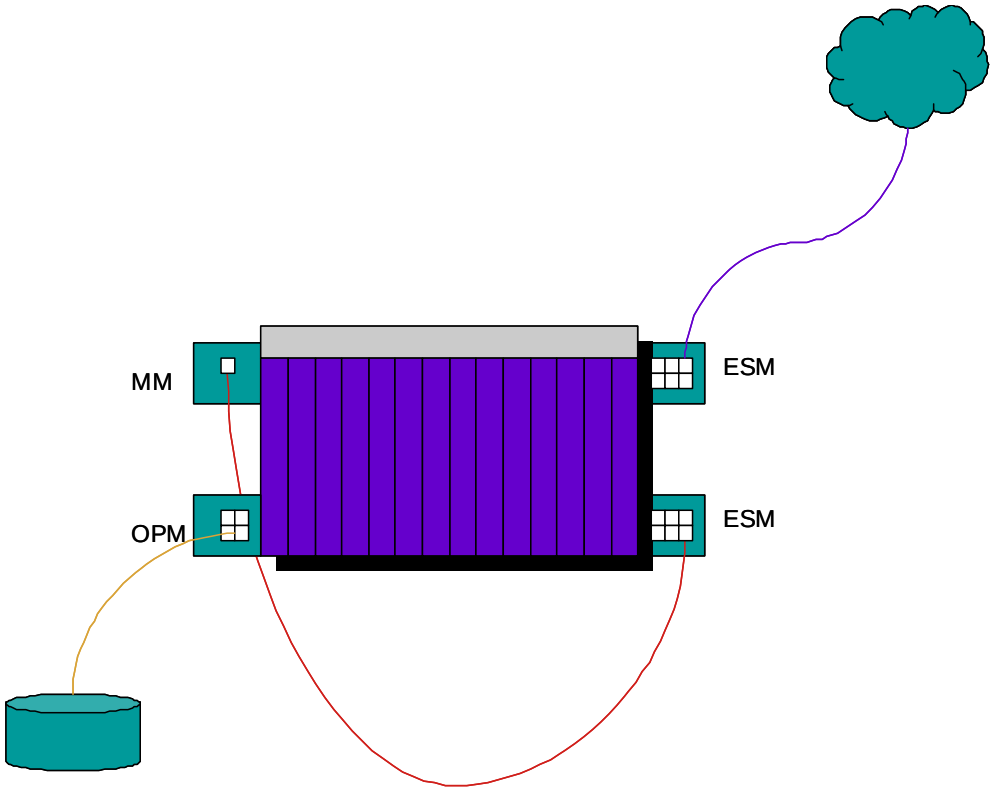


Figure 11. A small “starter” VCL cloud installation.

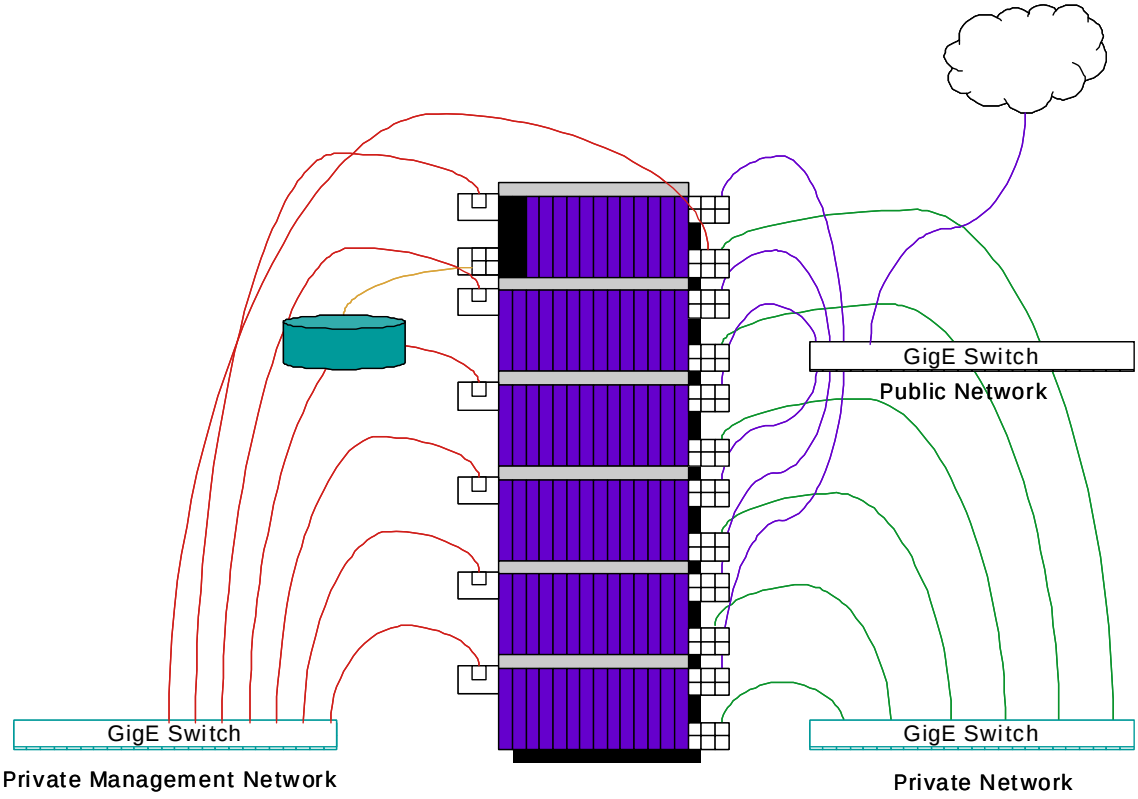


Figure 12. Scaling VCL Cloud.

**How do we scale?** Figures 12 and 13 show a rack of BladeCenter chasses. Additional racks are interconnected in a similar way. The differences between the two images are logical, i.e., switching from one mode to another is done electronically – in software – by VCL depending on the image characteristics. One can mix non-HPC and HPC configurations at the granularity of chasses. An important thing to note is that in order to maintain good performance characteristics, we do not want to daisy-chain internal chassis switches. Instead, we provide an external switch, such as Cisco 6509e (or equivalent from any vendor) that is used to interconnect different chasses on three separate networks and VLANs. In non-HPC mode, one network provides public access, one network is used for managing image loads and for accessing back-end image storage, and the third one is for direct management of the hardware (e.g., power on/off, reboot …). In HPC mode, the public network becomes MPI network, and special login nodes are used to access the cluster from outside. While we can use one VCL web-server and data-base for thousands of blades, with references to Figures 5 and 6, in a scalable environment we need one resource management node for every 100 or so computational blades to insure good image reservation response times – especially when image clusters are being loaded. We also need physical connection(s) to a storage array – we typically run a shared file system (such as GFS[5] or GPFS[6]) for multiple management nodes at one site.

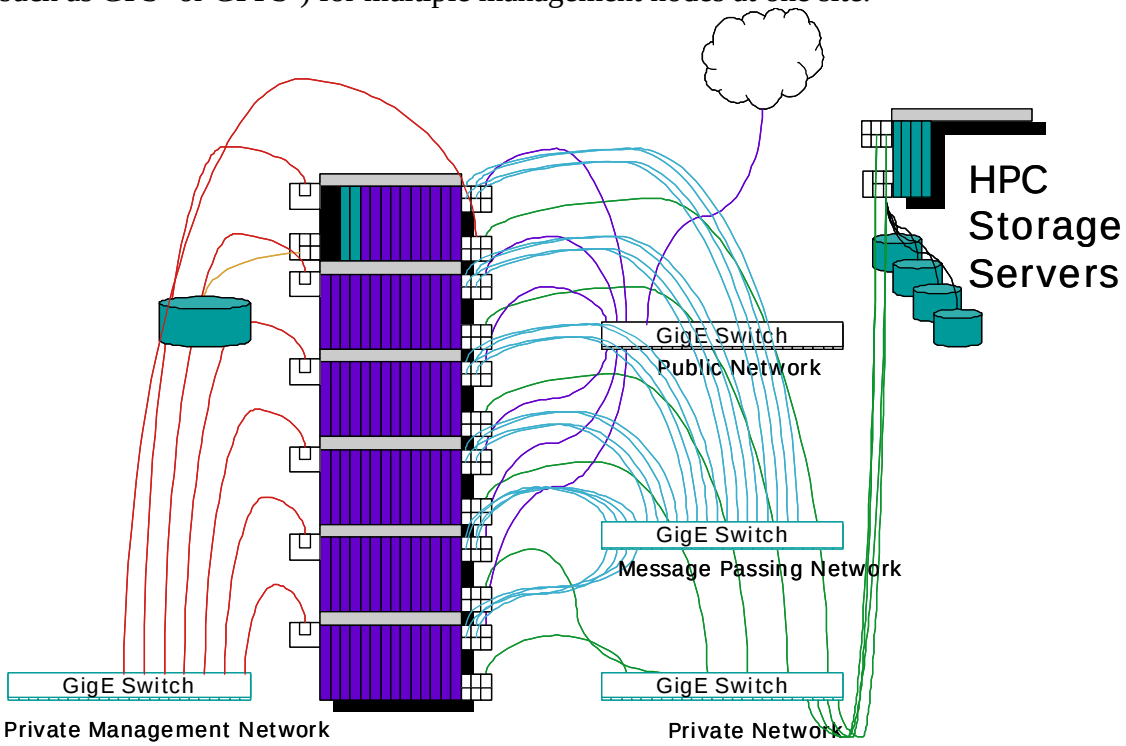


Figure 13. Scalable HPC Configuration

When VCL provides distributed and shared memory compute service for HPC, this is done through tightly coupled aggregation of computational resources with appropriate

[5] http://sources.redhat.com/cluster/gfs/
[6] http://www-03.ibm.com/systems/software/gpfs/index.html

CPU, memory and interconnect capacity. In our case, distributed memory compute services take the form of a logical Linux cluster of different sizes with Gigabit or 10 Gigabit Ethernet interconnects. A subset of our nodes have additional Myrinet or InfiniBand low-latency interconnects. Nodes which can be allocated for shared memory computations have large number of cores and plenty of memory.

To operate VCL in HPC mode, we dedicate one private network to message passing (Figure 10) – for that we use the blade network interface that would have been used for public user access in VCL standard mode (Figure 9). Also on a HPC BladeCenter chassis we configure two VLANs in one switch module, one for public Internent and for message passing interface. VCL management node makes those changes automatically based on image metadata. An HPC service environment image "knows" through its meta-data that it requires VCL-HPC network configuration (Figure 10) and those actions are initiated before it is loaded. VCL-HPC environment consists of one or more login nodes, and any number of compute nodes. LSF[7] resource manager is part of a login node.

Both login nodes and compute nodes are given permanent reservations (until canceled) – as opposed to time-limited resource reservations that typically occur on the non-HPC side. An HPC compute node image consists of a minimal Linux with LSF client that, when it becomes available, automatically registers with the LSF manager. All HPC compute images also automatically connect to user home directory and to shared scratch storage for use during computations. An HPC login node image contains full Linux and LSF server. There are usually two to three login nodes through which all users access HPC facility to launch their jobs. However it is also possible to reserve, using VCL web page, a "personal" login node on a temporary basis. On these "personal" nodes users can run heavy duty visualization and analytics without impacting other users. All login nodes have access to both HPC scratch storage and user HPC home directories (with appropriate permissions), as well as long-term HPC data storage. While compute nodes conform to configuration in Figure 10 – two private networks, one for MPI traffic, the other for image load and management, login nodes conform to Figure 9 topology, and have a public interface to allow access from the outside, and a private side to access and control compute nodes.

If we wish to add low latency interconnects for HPC workloads, we need to make additional changes in chasses and servers that will be used for that. Chassis network modules for low-latency interconnects (Myrinet, InfiniBand) need an optical pass-through and an appropriate external switch is needed (e.g., InfiniBand). Blade servers need to be equipped with a low-latency interconnect daughtercards.

# 4. Performance and Cost

VCL delivers both classroom, lab and research IT services for faculty and students. On the one hand, if users are to rely on VCL, then the VCL system must have sufficient available resources to satisfy the peak demand loads. On the other hand, if VCL is to

[7] http://www.platform.com/Products

operate cost effectively it is essential that it is not over provisioned to the point where the system is totally uneconomical to deploy. In order to provide VCL capabilities to users across widely varying demand loads, NC State decided to make a capital investment to assure that its "on-demand" level of service is available when needed. The user demand for these computing services is governed by the academic calendar of the university, and user expectations are that the availability of the services exceeds 99%. VCL meets that.

One way of assuring this high level of user availability, i.e., servicing of peak loads – see Figure 2 – is for the university to maintain a pool of equipment in standby or idle mode, for long periods of time. The consequence of this policy however would be an overall low average utilization of the resources. This is an expensive and uneconomical total cost of ownership option for the university. Therefore, one of the key VCL design considerations was sharing of HPC and non-HPC resources. In a research university, such as NC State, HPC is a very useful and needed workload. Because HPC jobs are primarily batch jobs, HPC can act as an excellent "filler" load for idle computational cycles thereby providing an option to markedly decrease the total cost of ownership for both systems.

An analysis of the NC State HPC usage pattern shows that researchers who actively use HPC computational systems, do so year round and do not have their computational workloads strongly fluctuating with the academic calendar. Because HPC jobs usually have large requirements for computational cycles, they are an excellent resource utilization backfill, provided that the cloud can dynamically transfer resources between single-"seat" and HPC use modes. Furthermore, it turns out that the demand for HPC cycles increases during holidays, and when classes are not in session, since then both faculty and graduate students seem have more time to pursue computational work. In NC State's case, co-location of complementary computational workloads – on-demand desktop augmentation and HCP computations - results in a higher and more consistent utilization and in overall savings.

In this context, one has to understand that although for most part HPC operates in "batch" mode, HPC requests are demanding as those of "on-demand" users, and rightly so. They expect their clusters to have very low interconnect latency, sufficient memory, and the latest computational equipment. Therefore, running HPC on virtualized resources is not really an option yet. Ability to bare-machine load HPC images is essential, as is the ability to map onto an appropriate interconnect topology. This need has recently been confirmed through a detailed comparison of NAS Parallel Benchmarks run on Amazon's EC2 cloud and NCSA clusters. Results showed wide variations in performance [Wal08]. While the specific results depended on the particular application and the cloud computing option used, the general lesson learned was that one needs to have not only control over what one is running on (virtual or real machines), but also over the interconnect latency among the cluster nodes. VCL was designed to allow sharing of resources while retaining full performance capability for the mode it operates in.

The VCL operational statistics over the past several years strongly support this design choice and suggest that by building a coherent integrated campus IT layer for faculty and student academic and research computational needs allows the institution flexibility in

servicing both of these university functions. It also allows the educational institution itself to maximize the return on their capital investment in the IT equipment and facilities and decrease the total cost of ownership.

IT staff supporting VCL see advantages as well because the systems is scalable and serviceable with fewer customization requests and less personnel (NC State uses 2 FTEs to maintain 2000 blade system). Because the VCL hardware may be remotely located outside the classroom or laboratory, there is also better physical security of the hardware and a more organized program for computer security of the systems.

**Commercial**? Commercial cloud computing firms are beginning to venture into the educational and research space using the pay-per-use model. Examples are Amazon EC2 services, and more recently Microsoft's Azure[8], but none of them, in our opinion, currently offer a viable integration of high-end HPC services into their clouds [Dre10].

For example, "Amazon Web Services has positioned their EC2[9] cloud offering in a way that is strongly focused on marketing rental of physical hardware, storage and network components. Although Amazon's EC2 enables users to acquire on-demand compute resources, usually in the form of virtual machines, the user must configure this hardware into a working cluster at deployment time, including loading and linking the appropriate applications. Using the Amazon web service users can create and store an image containing their applications, libraries, data and associated configuration settings or load pre-configured template images from an image library. Amazon implements "availability zones" to allow users some degree of control over instance placement in the cloud. Specifically, EC2 users can choose to host images in different availability zones if they wish to try and ensure independent execution of codes and protection from a global failure in case of difficulties with their loaded image. They also have choices when to run their images, the quantity of servers to select and how to store their data. Amazon bills customers on a pay-as-you-go basis for the time rented on each component of their cloud infrastructure."

**Open-Source?** There are a number of possible open-source solutions. Very few, if any, except for VCL offer the full HaaS to CaaS set of service. One example is Eucalyptus[10]. Eucalyptus has an interface that is compatible with Amazon Web Services cloud computing but treats availability zones somewhat differently. With Eucalyptus, each availability zone corresponds to a separate cluster within the Eucalyptus cloud. Under Eucalyptus, each availability zone is restricted to a single "machine" (e.g., cluster) where at Amazon, the zones are much broader.

[8] http://www.microsoft.com/azure/windowsazure.mspx
[9] http://aws.amazon.com/ec2/
[10] http://www.eucalyptus.com/

# 5. Summary

High-Performance Computing (HPC) services can range from time on peta-flop supercomputers, to access to high-end tera-flop facilities running a variety of operating systems and applications, to mid-range and smaller computational clusters used for HPC application development, pilot runs and staging. What they all have in common is relative isolation - that is traditionally HPC facilities have tended to be isolated from the more general scientific computing operations. Advent of the cloud computing concept has changed that. Even the most avid supporters of HPC and Grid computing are beginning to admit that almost all loosely coupled HPC computing, and a lot of tightly coupled HPC computing, can be done in a cloud. In this article, we have discussed a very successful production-level architecture and policy framework for supporting HPC services within a more general cloud computing infrastructure. This integrated environment, called VCL, has been operating at NC State since fall 2004. It typically delivers over 7,200,000 HPC CPU hours per year to NC State faculty and students, and about 100,000 non-HPC desktop-type reservations per semester. We have presented and discussed operational data that show that integration of HPC and non-HPC services in a cloud can substantially reduce the cost of delivering cloud services.

**Acknowledgments**

VCL development was supported in part by IBM Corp., Intel Corp., SAS Institute, NetApp, EMC, NC State University, State of North Carolina, and UNC General Administration. The authors would like thank the NC State VCL team for their advice, support and input.

## References


[Arm09] Armbrust, Michael, et al., *Above the Clouds: A Berkeley View of Cloud Computing*, Technical Report No. UCB/EECS-2009028, February 10, 2009 at www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009.html.

[Ave07] Sam Averitt, Michael Bugaev, Aaron Peeler, Henry Schaffer, Eric Sills, Sarah Stein, Josh Thompson, Mladen Vouk, "The Virtual Computing Laboratory," Proceedings of the International Conference on Virtual Computing Initiative, May 7-8, 2007, IBM Corp., Research Triangle Park, NC, pp. 1-16 (http://vcl.ncsu.edu/news/papers-publications/virtual-computing-laboratory-vcl-whitepaper).

[CAS09] Developing a Coherent Cyberinfrastructure from Local Campus to National Facilities: Challenges and Strategies: A Workshop Report and Recommendations, EDUCAUSE Campus Cyberinfrastructure Working Group and Coalition for Academic Scientific Computation February 2009

[Dre09] Patrick Dreher, Mladen A. Vouk, Eric Sills, Sam Averitt, "Evidence for a Cost Effective Cloud Computing Implementation Based Upon the NC State Virtual Computing Laboratory Model," in Advances in Parallel Computing, *High Speed and Large Scale Scientific Computing*, Edited by Wolfgang Gentzsch, Lucio Grandinetti, Gerhard Joubert, ISBN 978-1-60750-073-5, Volume 18, 2009, pp. 236 – 250

[Dre10] P. Dreher, M. Vouk, S. Averitt, E. Sills, "An Open Source Option for Cloud Computing in Education and Research," 2010, to appear.

[Gol09] B. Golden, *The Case Against Cloud Computing*, CIO Magazine, January 2009

[IBM07] IBM, "IBM Introduces Ready-to-Use Cloud Computing," Nov 15, 2007, http://www-03.ibm.com/press/us/en/pressrelease/22613.wss

[Loh07] Steve Lohr, Google and I.B.M. Join in 'Cloud Computing' Research, The New York Times, 8-October-2007,

[San09] Santos, Jack, et al., *The Dark Side of Virtualization*, Burton Group Advisory Program, April 6, 2009.

[Sch09] Henry E. Schaffer, Samuel F. Averitt, Marc I. Hoit, Aaron Peeler, Eric D. Sills, and Mladen A. Vouk, NCSUs Virtual Computing Lab: A Cloud Computing Solution," IEEE Computer, pp. 94-97, July 2009.

[Sea10] Cameron Seay, Gary Tucker, Virtual Computing Initiative at a Small Public University, Communications of the ACM, Vol 53 (3), March 2010, pp. 75-83

[VCL10] Apache VCL, http://cwiki.apache.org/VCL/, last accessed March 2010.

[Vou08a] Mladen Vouk, Sam Averitt, Michael Bugaev, Andy Kurth, Aaron Peeler, Andy Rindos*, Henry Shaffer, Eric Sills, Sarah Stein, Josh Thompson "'Powered by VCL' – Using Virtual Computing Laboratory (VCL) Technology to Power Cloud Computing ." Proceedings of the 2nd International Conference on Virtual Computing (ICVCI), 15-16 May, 2008, RTP, NC, pp 1-10 (http://vcl.ncsu.edu/news/papers-publications/powered-vcl-using-virtual-computing-laboratory-vcl)

[Vou08b] Mladen Vouk, "Cloud Computing – Issues, Research and Implementations," Journal of Computing and Information Technology, **16** (4), 2008, pp 235-246

[Vou09] Mladen Vouk, Andy Rindos, Sam Averitt, John Bass, Michael Bugaev, Aaron Peeler, Henry Schaffer, Eric Sills, Sarah Stein, Josh Thompson, Matthew P. Valenzisi, "Using VCL Technology to Implement Distributed Reconfigurable Data Centers and Computational Services for Educational Institutions," IBM Journal of Research and Development, Vol. 53, No. 4, pp. 2:1-18, 2009

[Wal08] Walker, Edward. 2008. Benchmarking Amazon EC2 for high-performance scientific computing. Usenix Magazine **33** no. 5 (http://www.usenix.org/publications/login/2008-10/openpdfs/walker.pdf).