ABSTRACT

MATHEW KALLUMKAL, GEORGY. A Micro-cloud Model for Adaptable High Performance Computing. (Under the direction of Dr. Mladen Vouk.)

High performance computing (HPC) and High-performance data (HPD) capabilities are now becoming the norm for advancing science through modeling simulations and data analytics. Often a serious limitation is internal and inter-computational node communication topology. They tend to be fixed. This basically means that applications and problems need to map onto these specialized architectures. This is expensive, takes a long time, and poses a challenge for rapidly rising demands for computation and data analytics. Instead, it may be a much better solution to virtualize the topology, re-arrange the nodes, and offer applications architectures which are dynamically customized for the computation and data analysis problem in hand. This, for example, is a reason why new FPGA accelerated as well as data-flow oriented HPC platforms are gaining interest.

In this project we are investigating a software-based solution to the problem using an open source cloud technology called VCL (Virtual Computing Laboratory) in combination with an open source solution called KittyHawk. As a proof of concept, we have created a soft-version of a reconfigurable architecture residing in a supercomputing environment (specifically IBM BlueGene/P or BG/P). Solution conforms to the Department of Energy secure access standards. Configuration is managed using a root-less version of VCL code and IBM Kittyhawk running on BG/P. Pilot is implemented on the Argonne National Laboratory (ANL) Surveyor BG/P. This implements a distributed cloud computing model we call VCL Micro-cloud, with KittyHawk as its provisioning engine where users are able to request individual BlueGene nodes under topology of their choosing and design that can flexibly map to the computation problem or data analytics requirements. VCL

Micro-cloud has a management node which controls and provisions other nodes in the cluster and executes the requested application on them. A user who wants to have a customized Micro-cloud needs to make a reservation of the image environment in VCL and then request a template which then will be configured as a Micro-cloud on BG/P.

Results indicate that Micro-clouds of different configurations can be easily constructed. Network interfaces between the nodes in the cluster are configured based on templates. We describe the technology, illustrate how to setup such an environment, and we discuss the performance of the solution.

A Micro-cloud Model for Adaptable High Performance Computing

by
Georgy Mathew Kallumkal

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Computer Networking

Raleigh, North Carolina

2013

APPROVED BY:

Dr. Patrick Dreher	Dr. George Rouskas
Dr. David Thuente	Dr. Mladen Vouk Chair of Advisory Committee

DEDICATION

To my parents, my elder sister Krupa and to my friend Shivani for their love and support.

BIOGRAPHY

Georgy Mathew was born in Mumbai, India. He did his primary and secondary schooling from St. Francis D'Assisi High School, Mumbai, India. He received his Bachelors degree in Electronics and Tele-communication from University of Mumbai in May 2011. He joined North Carolina State University in Fall 2011 for pursuing Masters in Computer Networking. Apart from academics he also likes to create cover music.

ACKNOWLEDGEMENTS

I would express my sincere gratitude to my advisers, Dr. Mladen Vouk and Dr. Patrick Dreher for their invaluable guidance in performing research, writing this thesis, and throughout the course of my graduate studies. It is because of their continuous support and assistance I was able to finish this work successfully. I would like to sincerely thank Dr. George Rouskas and Dr. David Thuente for their kind consent to serve on my thesis committee. I am obliged to Dr. Marta Garcia Martinez, Assistant Computational Scientist at the ALCF for providing us all help and assistance regarding the needs and requirements of ACLF resources on this VCL-on-BG/P project. I am extremely thankful to William Allock, ALCF Director of Operations for helping us to get the hardware configuration and dedicated log-in server needed to conduct this project on the Surveyor BG/P machine. I would like to take this opportunity to thank Prof. Jonathan Appavoo, distinguished professor at Boston university one of the Kittyhawk developers, and his graduate student Dan Schatzberg for providing us help and advice in porting the Kittyhawk utility at ALCF. I would also like to thank Mr. Marhn Fullmer, manager of Netlabs at NCSU, for his relentless technical support. I would like to take this opportunity to thank my research group members Vikas Cheruku and Sai Rajesh, my friends Tom George, Leon Dmello, Sanjana, Ishita Shinde, Ushma Thakker and Venisa for all their help and support.

This work was supported in part by NSF grant 0910767, the U.S. Army Research Office (ARO) grant W911NF-08-1-0105 managed by NCSU Science of Security Initiative and Science of Security Lablet, and IBM Share University Research and Fellowships program funding, the Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-06CH11357.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

In 1960 Seymour Cray introduced the first Supercomputer CDC 6600 [10] at Control Data Corporation which provided very high computation peak performance. In early 1970's such high computations were required only in a few domains. But as Internet evolved the demand to solve large scientific problems and data intensive applications significantly increased. Today science and internet have grown to such an extent that huge amounts of data needs to be processed daily. For example Facebook stores more than 30 petabytes of data [9] daily which needs to be analyzed and filtered. High computational power is largely needed in domains such as quantum physics, weather forecasting, climate research, molecular modelling and physical simulations. Apart from these domains we are entering into an age where the entire globe can be considered as a data field. Todays most of the real world objects come with chips or tags built-in in them. All these devices can be connected together using the Internet. Such an interconnected network is called Internet of things [74]. There exists millions of such devices all over the world and these devices process enormous amounts of data every minute. So we need systems that can process large amount of data. These in turn would speed up the data analysis. Such huge amounts of data which are generated daily from various firm and industries is termed as "Big Data" [26]. Companies such as Google, Facebook, Twitter etc. which handle such large amount of data daily depend heavily on such systems.

In order to solve these large data driven problems high performance supercomputers are designed by some of the leading manufacturers such as Cray [11], IBM [22], Fujitsu [16]. These systems can achieve peak speeds of 17.50 pflops, 16.32 pflops and 10.51 pflops respectively. Such systems have nodes with multiple cores and memory. Large number of nodes are aggregated together in racks which offer petaflop operation speed. These nodes are tightly coupled and have fast network interconnects between processor units which ensure high performance. There are many challenges associated in developing HPC hardware, such as huge number of computers, developing an algorithm for scheduling and executing data intensive applications, cost of hardware, cooling system etc. It becomes very difficult to administer and maintain such high performance clusters and supercomputers. Moreover, the underlying architecture on a supercomputer has a fixed network interconnect topology between the nodes. There might be various workflows which, when mapped to a custom topology, performs faster as opposed to a fixed topology.Authors in [12] list various frameworks like Mapreduce [30] etc which can be used to process Big Data.

Cloud Computing technology provides on-demand services of various computing resources which can be provisioned easily with minimum maintenance and administration efforts. These computing facilities when properly configured with a supercomputing environment can be used to provide resources for various scientific computations and Big

Data analytics. Yuichi et.al in [95] describes two methods which can be used for processing big data in a cloud environment. It might be a good idea to construct a software layer which maps the architecture to the problem and thus construct a custom topology based on the application to be executed. The advantage of such a method is that we can have the flexibility of constructing our own domain. Data can be distributed among various nodes and will be processed based on a data flow model. In this chapter, we present the current technologies that are used for high performance computations, how does HPC in a cloud perform, how network interconnects on a supercomputing environment provide much better performance than HPC clusters, how FPGA accelerated are used for HPC, and the motivation behind creating a dynamic network based on Data Flow Computing.

1.1 High Performance Computing

High Performance Computing [23] is very essential for scientific workloads of various Science Technology Engineering and Mathematics (STEM) projects which require very high computing power and low network latency. Such high performances can be achieved by running the applications on shared cluster environment [57] or on a supercomputing hardware. Both these systems have their own advantages and disadvantages. One can argue that shared clusters are loosely coupled and thus their performance will not be sufficient in terms of bandwidth requirement specially when huge data chunks are considered. HPC over the years has evolved from Cray machine in back 1970's to cluster computing and now supercomputers [54]. Cluster computing is technique in which local nodes are grouped together using network connections and thus providing a shared resource [61]. There are many manufacturers like Dell, HP and Intel who provide HPC solutions [13] [20] [24] [19].Techopedia in [40] quotes HPC as the ability in using various parallel processing techniques for solving various scientific problems, analysis and modelling. In simple words, it is about combining various systems together to produce high computing power to cater the increasing demand for processing speed. The various scientific workloads can be classified into following three categories, [64]

- Large-scale tightly coupled systems : This category comprises of tightly coupled HPC applications that require flexibility in computation.
- Mid- Range or Loosely Coupled systems : These are workflows which can perform using virtual clusters.
- High-Throughput : Workflows that require high performance in post processing analysis or calculations.

Since thousands of processors and gigabytes of memory are put together to produce high computing powers, it is very expensive to run applications on such systems. In-spite of advancements in chips, network interconnects and storage technologies the expense incurred while using these platforms haven't really come down so that a normal user can use it from their laptop or desktop. In such a scenario, cluster computers are a useful solution where distributed computing is performed between nodes either in a serial or parallel fashion. It is important to note that all compute intensive problems cannot be mapped to a cluster computing environment as the network connections or underlying network topology might not be fast enough to support the requirements. It is interesting to see the results that are obtained when these systems are integrated with cloud computing technology [86] [50] [108]. Cloud computing provides an on-demand, pay as per use model for shared resources. As mentioned earlier, if we can properly configure a cloud computing environment we can provide HPC services to users. One such classic example is the

HPC facility provided at North Carolina State University, where VCL Cloud Computing Platform is used to provide HPC cluster access [34]. HPC applications which require minimum I/O operations and communication between nodes in a cluster can be used via a Cloud Computing environment. The next section presents the various cloud computing solutions that are used to provide HPC facilities.

1.2 HPC in the cloud

Cloud Computing technology can be used as a management tool for HPC services. It provides the flexibility in offering a user with IaaS, PaaS, SaaS etc [60] [87] [88] by sharing computing resources. Based on the requirements resources can be easily provisioned and deployed through the cloud. Various firms are looking at integrating HPC in the cloud [21] [18] [39]. It is difficult for each and every individual to have the skill set and expertise to develop algorithms for scientific workflows or big data analytics. In such situations cloud solutions come in handy where the user can perform data intensive computations through the cloud accessing the data in a distributed manner. Carlyle et.al [58] discuss cost comparisons of doing scientific computations using a cloud computing in an HPC environment. Amazon EC2 [3] added HPC capability to provide elastic compute clusters for scientific and large scale computing in public cloud [2]. After Amazon added the support for HPC services significant research has been conducted on the EC2 platform to check computational performance [107] [66] [71] [75].

From these results, we can see that EC2 clusters are still lagging in terms of performance as compared to tightly coupled HPC environments. They appear to be more suitable for small scale HPC applications. Excellent network performance is a key criteria needed to support HPC applications, which most of the public cloud lack. The major drawback with EC2 cluster is that the performance is limited due to the network interconnects. An improvement in the network interconnects will likely produce significant improvements for compute intensive applications. There are other cloud solutions like Eucalyptus [14], OpenStack [35], Hadoop MapReduce [30] [17] and Nimbus [33] which focus on loosely coupled environments that can be used for scientific workflows which require less number of I/O operations. Thus we can see that there is a need for research in this area to better support the user demands for both HPC computing and data analytics.

1.3 FPGA based HPC and Data Flow Computing

From the above sections we have seen that traditional HPC environments offer very good performance for compute-intensive applications. Field-Programmable Gate Arrays (FPGAs) [15] were invented by Ross Freeman in 1984. Maxwell designed a FPGA supercomputer and investigated the concept of hardware acceleration for HPC [53]. FPGA devices help to reconfigure High Performance Computing by optimizing the algorithm for improved computation performance. In certain case studies [83], improvements in operating speeds of 20-70 times faster were recorded. Various applications like fluid flow, imaging, geoscience have also used FPGA accelerated hardware to obtain higher performance. Today there are many systems [100] [69] [52] [55] [101] [93] today that use FPGA accelerated hardware to obtain peak performances. As mentioned by Buell et.al in [56] FPGA-based system follow the 10-90 rule (ie FPGA processors execute 10 percent of the code which take 90 percent of the time for execution). Basically these devices support the CPU processor which runs the main application and handles those parts of the kernel that take more execution time [65]. The working and rationale behind high speed working of these systems are explained in [83]. The kernel simplifies the code and converts then into data-flow graph which uses static loops where each node is assigned an arithmetic operation. Thus using FPGA accelerated hardware can increase the performance of the system [72] [109].

Any workflow can be executed on a system in two methods, either using a control flow approach or a data flow approach [63]. In the former approach the program is converted into a set of instructions which is then sent to the processor [103]. The processor executes these instructions sequentially and accesses the data from the memory. In the latter approach the data is moved from one unit to another and the results are collected. Data Flow computing can be done at multiple levels (ie system, architecture, arithmetic etc.) [102]. As mentioned earlier, data streams from the memory are forwarded to the processor units or cores which are assigned to perform one arithmetic operation. Data is then processed like a chain. Data Flow computing is a programming model that was proposed in the early (1980's). It is gaining significant impetus now due to emerging demand for distributed computing and parallel processing of large quantities of data [97] [84]. For example Systolic arrays [81] are used to combine large number of processor together for scientific computations and signal processing. Such an approach provides better performance than a control flow based program and is being considered for providing high performance for compute-intensive applications. Thus, when running applications on supercomputer hardware, we can design a data flow compute model for the application and execute it on a set of nodes. In order to visualize this, we will see how network interconnects are configured between processor units on a supercomputer hardware and how we can use it for a data flow computing.

1.4 Network Interconnects on Supercomputing Hardware

Communication domains in an HPC infrastructre system play an important role in deciding the performance of the system. For compute intensive applications we can use loosely coupled distributed systems or tightly coupled supercomputing environments. Software systems such as Mosix [31], Amoeba [4], Plan9 [36] which are loosely coupled use standard Ethernet networks to establish a communication domain between the nodes thus forming a cluster between them. In compute intensive applications there is a constant exchange of data and messages between nodes and hence it is essential to have high speed interconnection in order to reduce latency and have very high performance. Loosely coupled systems does not fair well in terms of latency and performance.

Supercomputers being tightly coupled systems, focus more on caching, resource management and very high data transfer rates. In such systems communication domains are constructed using custom interconnect networks which provide very high data rates and excellent performance, thus creating an environment suitable for compute-intensive applications. For example Thunderbird [41] uses Infiniband interconnects, Red Storm [38] and Jaguar [27] desinged by Cray uses XT3 Internal Interconnects and IBM BlueGene [7] [90] uses various custom interconnects. These systems are designed considering the dense structure of nodes, high-bandwidth requirements, low power usage and highly reliable network infrastructure. With such a high performance underlying network interconnects, we can implement various heterogeneous scientific workflows by creating a software layer over the nodes and clustering them together based on the problem being posed.

Hybrid communication networks and switching fabric provides the performance required for high-performance computer systems. Point to point and multi-stage communica-

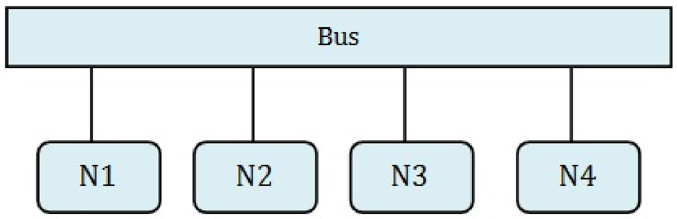


Figure 1.1: Interconnect Networks -Bus Topology

tions, which involves intermediate routers or switches are used to establish communication between nodes. There are many primitive basic interconnection networks which use bus, crossbar swithces for communication between nodes. But in terms of scalability, as the number of nodes increases in a bus topology (Figure 1.1) the latency incurred increases which affects the performance. Same is the case with crossbar switches (Figure 1.2). This stressed on the need for having routers or switches in between nodes which could facilitate fast data transfer. Dynamic networks such as ring (Figure 1.3) networks [67] and multistage (figure 1.4) networks like fat-tree CLOS networks [70] were developed for the same. Low radix routers [62] were designed, which supported multi-stage networks and led to the advent of hypercubes - cosmic cube [6] which uses either 2D Mesh or torus networks.

While designing an interconnection network we need to take into consideration of three aspects, the topology between the nodes in the network, an algorithm which governs the routing between them and finally controlling the flow of packets in the network [78]. There are published references [25] explaining the detailed operation of various interconnection networks. Today science has advanced to such an extent that the performance expected from the underlying hardware has increased. This demand led to the advent of new high

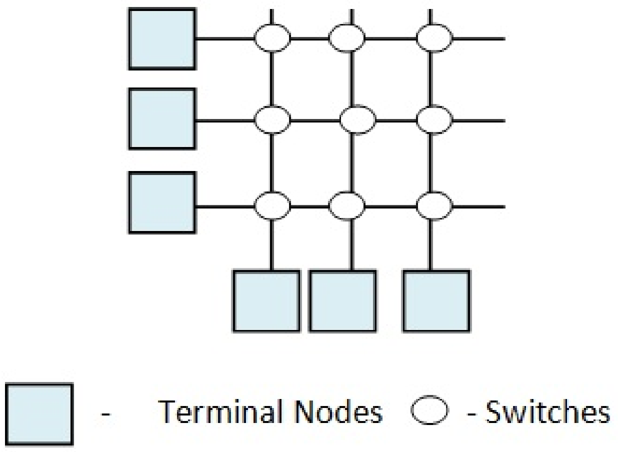


Figure 1.2: Interconnect Networks - Crossbar switches

radix routers [80]. As shown in [62] low radix routers have few number of ports which have fat links whereas high radix routers have more number of ports and thus skinny links.

Topologies used in an HPC environment include Fat-tree (CLOS [8]), Infiniband [89], Mesh and 3D-Torus. Custom topologies [96] can also be constructed to improve the performance by allocating the operations to each network. Let us take a look at these topologies which are used for building a high performance communication domain in a supercomputing environment. Infiniband is a multi-purpose interconnection network which has the capability to connect thousands of nodes or servers together and providing very high performance. Infiniband provides three levels of network bandwidth 10Gbps, 20Gbps and 40Gbps [51]. This topology is limited to cluster supercomputers as the infrastructure software has to be tweaked in order to support it. It is used for I/O operations and was primarily developed by InfiniBand Architecture (IBA) [92] to avoid issues created due to point to point communication. It uses specific message format for communication between

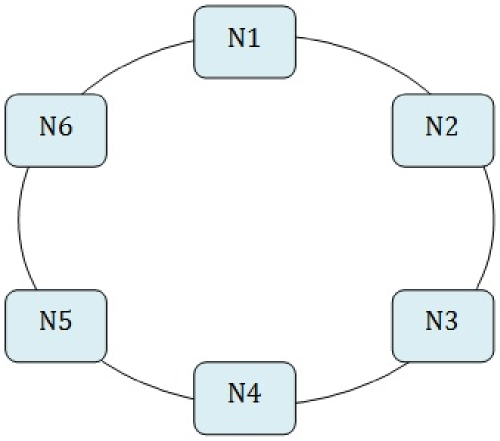


Figure 1.3: Interconnect Networks - Ring Network

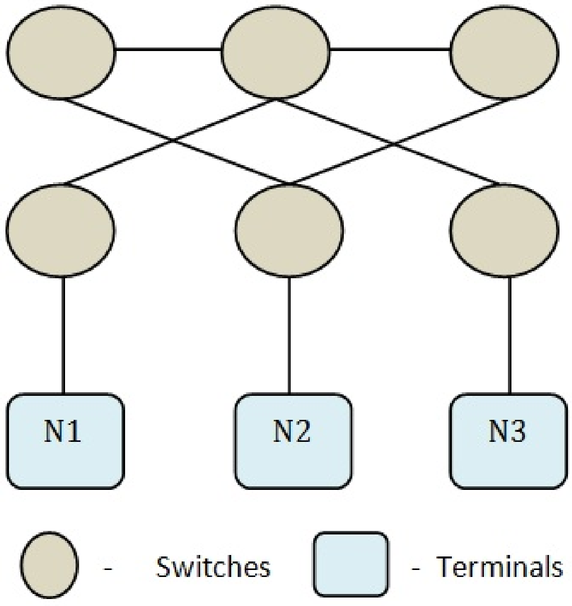


Figure 1.4: Interconnect Networks - Multi-stage Networks

host machines and other devices. A new topology “Flattened butterfly ” which is used in high radix routers is explained in detail in [76] [79]. It is a cost efficient approach and offers lower latency as compared to other network topologies like hypercube and CLOS. Flattened butterfly topology is created by configuring routers on each row in a butterfly topology such that it does not affect the interconnections [76].

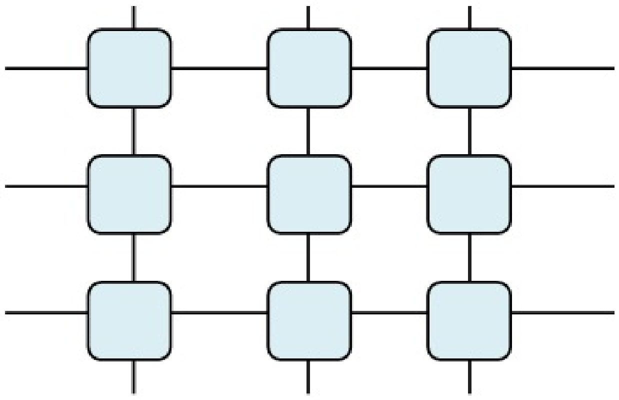


Figure 1.5: Interconnect Networks - Mesh Networks

Mesh networks [44] (Figure 1.5) are 2-dimensional networks where the nodes are arranged in a row and column fashion with each node having a point to point link to its neighbours. When the end nodes in a mesh are connected back to the first node then the network becomes a torus network. When high radix routers are used all the communication between nodes occur through the switches. As shown in (figure 1.6) [1] 3D torus networks offer direct point to point communication between nodes and provide high performance in terms of scalability and speed. Redundant channels are provided for each node which can be used to create different communication domains. Kim et.al introduces a new topology called as Drangonfly [77] which is used in some Cray manufactured supercomputers. This technique uses high radix routers as virtual routers and further more it reduces the number of global channels required for the system and results in reducing the cost. Routers are grouped together using local channels which constitute the virtual router and these groups are then interconnected using global channels. Thus moving from low radix to high radix routers increases the network bandwidth and scalability. These topologies are used in most of the recent supercomputers due to their low latency and high performance.

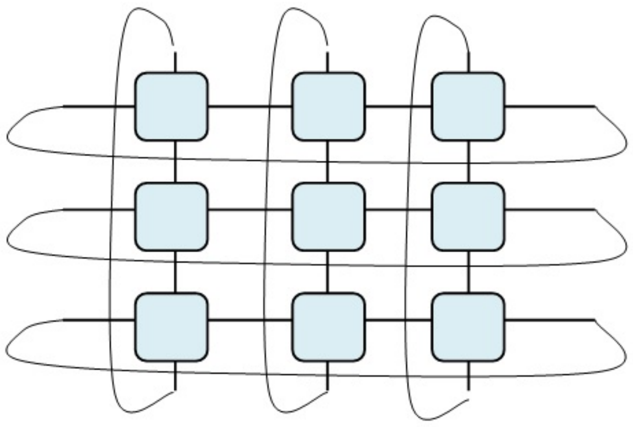


Figure 1.6: Interconnect Networks - Torus Networks

1.5 Motivation for Dynamic Topology Construction

We have seen that the major problem with loosely coupled systems or clusters is that they use Ethernet communication between nodes resulting in sometime unacceptable network latency. It is easy to configure nodes in a cluster based on the requirements, but the underlying architecture should be able to provide high performance specially for compute intensive applications and workflows. In compute intensive applications data needs to be transferred between nodes at a very high rate. Also fast links are needed to access the storage space. We can create clusters which provide the users with high processing power but network connections between nodes should be fast enough to support the fast processing of data. In such cases, tightly coupled systems such as using a supercomputing environment is used for various scientific applications and other HPC applications which require high processing power. As seen in the above section in a supercomputing environment various interconnect topologies are constructed to connect the processor units of nodes. We can implement various scientific problems using such a topological environment. There can also be cases where it might be beneficial to create a data flow workflow of the problem and have different nodes perform different operations. We can have the data distributed between nodes. Apart from this there can also be workflows which needs to implemented over a custom topology where isolated communication domains like VLAN's [43] need to be created.

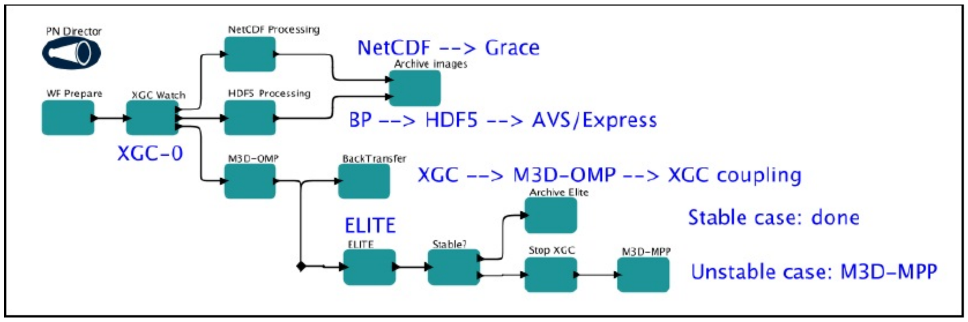


Figure 1.7: Scientific Workflow Topology example [98]

Figure 1.7 shows a scientific workflow which was designed using Kepler [28] in [98]. An interesting thing about the Kepler workflow system is that it supports a number of processing models including simple data flow, distributed data flow, and process networks. This makes it appealing to map workflows written in Kepler, which is used to implement scientific workflows, onto a flexible dynamic topology that would coincide with the data operations the workflow implements. We can see that three are three main streams. In the first part a single node is dedicated to perform NetCDF processing, second part another node is assigned to perform HDFS processing and the third stream performs an M3D OMP process but has multiple nodes involved. The most important functionality that we can infer from Figure 1.7 is that every single node is assigned a unique function to perform and are different from each other. There are three different data processes that occur parallely. Such an implementation can offer better performance in some cases as opposed to using the fixed underlying architecture. This motivates the need of having a software defined networking [85] on the supercomputing hardware which provides the flexibility of configuring networks on the fly than doing them manually based on the requirement. This is analogous to creating a virtualized layer over the hardware thus providing dynamic network configuration. It is essential to have such a technique to cater the rising demands of computation and data analytics. Thus instead of mapping the application and problems to the fixed network architecture we can construct a data flow model of the problem and create a topology to execute them.

In this thesis we create an environment using VCL and IBM KittyHawk which provides the functionality of constructing dynamic network topologies between nodes on IBM BlueGene/P supercomputer. KittyHawk is a tool that runs on the BlueGene Login node which provisions BlueGene nodes and creates custom networks to which nodes can be added. We can consider it like a provisioning agent. Just as we would use Vmware to create virtual machines and configure a network between them, KittyHawk provides the same capability over raw nodes on a supercomputer like BlueGene. We integrate this provisioning agent into VCL which then provides a complete soft-version. VCL provides a well defined user interface to the user through which they can request for specific number of nodes and networks and then construct a topology between them. This information is passed onto the KittyHawk modules. VCL has a very flexible design through which resources are controlled. We assign these raw BlueGene nodes as computer resources in VCL and these computers have KittyHawk as their provisioning engine. A user can reserve the topology from VCL User Interface and run applications or workflows on them. Detailed explanation of the system environment, design and implementation is given in the following chapters.

In chapter 2 we describe the environment used. We first describe about BlueGene/P supercomputer, the hardware architecture, node specifications, how they are organised and finally the interconnection networks between them. Later on we describe features of kittyhawk and how they create a logical communication domain over BlueGene environment. We also provide details about VCL architecture, the services it provides etc. In Chapter 3 we describe the system design. This section is further divided into two parts. First section explains how KittyHawk is configured and designed for our system environment and the next section presents a detailed design of VCL. In chapter 4 we describe the installation and integration of these modules and also go over the system flow based on an example. Chapter 5 talks about the experiments and results. We conclude the discussion in chapter 6. Detailed installation manual steps and figures for the results are mentioned in the appendices.

CHAPTER 2

Environment Used

The main focus of this project is to create an environment on a supercomputer where the interconnectivity between nodes can be reconfigured based on the data flow model of a workflow. Supercomputers have fixed architecture at their node interconnect level. We create a soft-version which will enable a user to create logical custom interconnections between the nodes. We implement the same on IBM BlueGene/P supercomputer. The software layer is created by integrating KittyHawk and VCL. Thus we have three main systems which are integrated together to provide the proposed system. Before we get into the system design and configuration it is essential to understand the system components. In this chapter we will go through each of the components briefly and understand how they are configured. We will first go over the BlueGene/P architecture, the hardware and software configurations and various interconnect models used at the hardware level. In the sections ahead we will go through the various components of VCL and KittyHawk and how they function.

2.1 IBM BlueGene/P Architecture

IBM Bluegene is a project which aims at producing next generation supercomputers which achieve operating speeds in the Petaflops range. There are three ranges of computers BlueGene/L, BlueGene/P and BlueGene/Q.

The heart of BlueGene is a Node which contains these two elements a processing unit which controls all the processing tasks and an interconnect controller which is used to manage interconnection between nodes. These systems are efficient in terms of power consumed. We split the BlueGene Hardware into three sections, first Node Model where we will describe the components of each node and the processing power etc. BlueGene Architecture is made up of different types of nodes such as front end nodes, compute nodes, I/O nodes etc and their functionality is described in the section Node organisation. It is essential to see how these nodes interact with each other using the various Interconnect Models.

2.1.1 Node Model

The central unit in a BlueGene/P computer is a quad-core chip also called as a Processing node. It consists of a 850 Mhz Quad Core PowerPC CPU with 2GB of RAM and an Interconnect Controller. These chips are loaded onto a Compute Card shown in Figure 2.1 with one chip on each card which achieve upto 13.6 GFlops of operating speed. Many such compute cards are put together to form a node card. Each node card shown in Figure 2.2 consists of 32 compute cards. Two I/O nodes are allocated for each node card. I/O nodes consists of an 850Mhz Quad-Core PowerPC CPU with 2GB of RAM and also has a 10G Ethernet Controller for communication with external world. With 32 nodes, each node card has 32 * 2 = 64 GB of memory and an operating speed of 435 GFlops. The next aggregation level is the rack shown in Figure 2.3 where 32 node cards are stacked together to form one rack. There are two mid-planes with 16 node cards on each plane. Putting together each rack contains 1024 nodes ie 4096 cores and a total memory of 2TB and peak operating speed of 13.6 TFlops. A petaFlop operating system is achieved when 72 such racks are combined together.

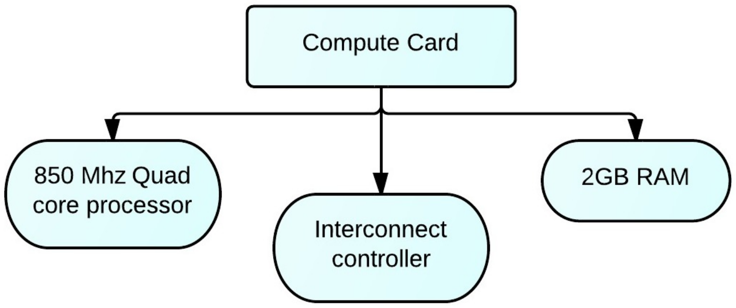


Figure 2.1: BlueGene/P Compute Card Components

2.1.2 BlueGene Node Organization

There are four types of nodes in a BlueGene/P Environment as shown in Figure 2.4 (referred from [90]). They are compute nodes, I/O nodes, front-end nodes and service nodes. As mentioned above these nodes are quad core nodes with 2 or 4GB memory. Users do not have access to the compute nodes which are used to run various applications. They have an operating system called as Compute Node Kernel (CNK) installed on them.

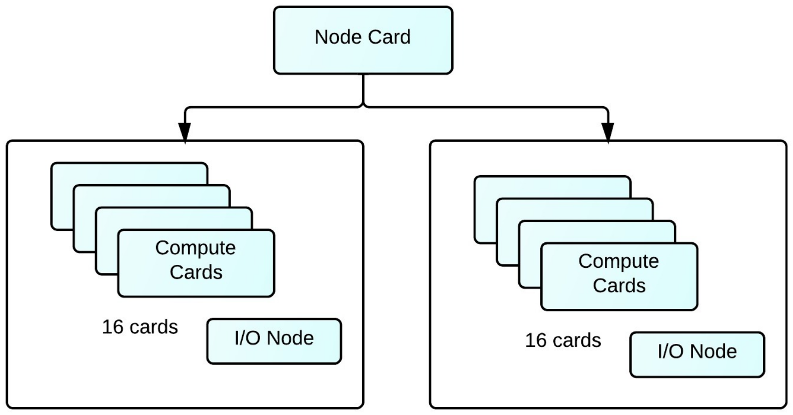


Figure 2.2: BlueGene/P Node card components

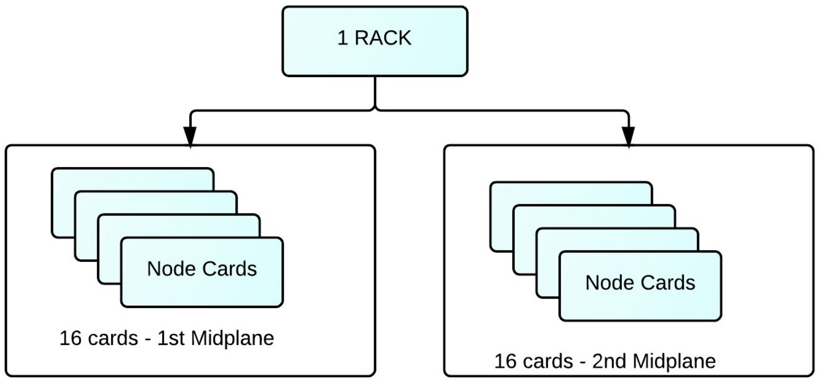


Figure 2.3: BlueGene/P Rack

All the I/O operations from these compute nodes are directed to the I/O node as it has external access through a 10 Gigabit network connection. I/O nodes provide various functionalities such as starting a process, network connections etc. The ability for the users to login into the system and compile and submit jobs to the BlueGene nodes is provided by the front-end nodes. Using this node the user can start, delete and check the queue status of a job that is submitted. For such a complex system there should be constant monitoring of nodes for various security aspects. These services are provided by the service nodes which perform monitoring, synchronization and management services. Analogous to the compute nodes the users do not have access the service nodes.

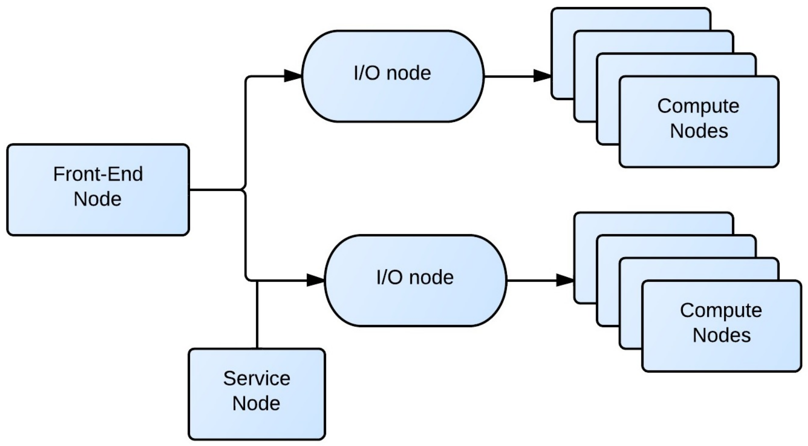


Figure 2.4: BlueGene/P Node Organisation

2.1.3 Interconnect Models

There are 4 types of interconnection networks namely 3-Dimensional Torus Network, Collective Network, Global Control Network and 10Gigabit Ethernet network which are used by the BluGene Nodes to communicate between each other and to the outside world. Let us take a look each of the networks.

- **Global Control Network**: This network is used to send boot configurations when the BlueGene/P nodes are booted up. For example when we use KittyHawk it uses this network to communicate with the nodes and send them with the various boot-related information. The boot-loader firmware is loaded on these nodes using this control network.

- **Torus Network**: In this network every node is connected to every other node. It basically connects all the compute nodes and creates a point to point communication between them. Every node will have links to its 6 neighbours and in total there will be 12 links as we need to count both incoming and outgoing links.

- **Collective Network**: This is analogous a hierarchical network which connects all compute and I/O nodes together and can be used to send broadcast and multicast messaged to the compute nodes.

- **10GBit Ethernet Network**: As mentioned in the section 2.1.2 each I/O node is provided with the network connection so that it can have external access.

2.2 KittyHawk

KittyHawk [48] [37] is a system software that was developed to explore the use of a global scale shared computer to host various applications. It was build on BlueGene/P and thus creates a platform which can be used to run various heterogeneous workloads. It provides a software environment which helps in building various solutions and services on a supercomputing environment which takes into consideration implementation complexity for BlueGene/P. The main goal of Kitty-Hawk is to reduce the software that is needed to provide a service developed on BlueGene/P [47]. It provides an abstraction layer which utilizes the benefits of the underlying hardware. BlueGene/P nodes can be considered as computational resources with specific memory, storage, network interfaces with the ability to load an kernel image through a software like KittyHawk.

2.2.1 KittyHawk Features

Scalable hardware and proper software provisioning are two important aspects required in a global scale system [48]. Today there are two major computational environments: tightly coupled systems and loosely coupled clusters. Kitty-Hawk designers argue that loosely coupled clusters or systems which use shared memory cannot support a wide range of applications [48]. While designing such a system it is important to make sure that the applications running on top of such a hybrid architecture are able to utilize the performance. Loosely coupled clusters do not share memory but use various virtualization techniques which might affect the performance of the application. Thus hybrid computing environments should be able to provide the best performance for heterogeneous workloads. Reliability and sustainability of the system is also very important while developing such systems. In BluGene/p environment all the resources or compute nodes are tightly coupled with custom or torus node interconnects between them. KittyHawk helps to create a logical network between a pool of nodes where nodes do not share memory and can be easily configured for the required application or workflow. Raw computational nodes are loaded with a very basic boot-loader followed by a software image which consists of kernel image and ramdisk image which configures the nodes based on the specified command line arguments. These images can are either loaded using a Push Model or a Pull Model. In Pull model the image to be loaded on the nodes will be stored on an external storage which can be accessed using NFS storage or nbd-server / client where as in a Push model the kernel image is pushed onto the node via the network.

2.2.2 KittyHawk Abstraction : Communication Domains

Communication Domains help to create a flexible abstraction over the hardware interconnects. KittyHawk provides the flexibility to define a communication domain between the nodes. For example if we define a node to have the ability to communicate with with two different networks then it will have two network interfaces at the software level with each interface defined for a corresponding network. This is similar to creating smaller host-only subnets inside a network. KittyHawk enables the implementation of such custom communication domains where users can add nodes to specific networks or remove nodes from a network. As seen in the previous section where we described about the interconnect model of BlueGene/P, at the hardware level each node is associated with an identifier which defines it location on a torus like network. Kitty-Hawk provides the ability to develop a logical layer where users can construct groups of nodes with certain set of access rights which will define the communication domain. KittyHawk also provides a method to exploit the tight interconnections between BlueGene nodes and thus providing high performance for compute-intensive applications. It uses a distributed memory caching system called “memcached (memory cache daemon)” to utilize RDMA (Remote Direct Memory Access) features of BlueGene/P hardware [49]. Memcached [68] is a distributed caching system which provides high performance to various applications such as caching database query results to reduce the load and contention. KittyHawk uses a modified memcached system which at the lower layer can invoke calls to the torus network driver and thus uses the torus network RDMA to transfer data between the nodes.

2.3 Virtual Computing Lab (VCL)

Virtual Computing Laboratory (VCL) is an open source Cloud Computing Platform developed at North Carolina State university. It was a concept desgined by Vouk and Averitt et al [94] and aimed at providing on demand services to the university. Today VCL is the top level Apache Software Foundation project [99] [59] which provides on-demand cloud services which includes physical, virtualized, storage, networking and software resources. VCL provides more than 80,000 reservations and 7 million hours for HPC [32]. It offers wide range of flexible services like Infrastructure (IaaS) [60], Software and Platform (SaaS and PaaS) [87] [88] as well as High Performance Computing Clusters. It provides single seat desktop reservations, hypervisor based platforms with the flexibility of underlying hypervisor such as VMware, KVM, Xen and combination of various services to individuals. HPC services and Network Virtualization are important features in VCL.

2.3.1 Services Provided by VCL

VCL provides the basic service of Infrastructure. It can be either in the form of bare-metal loads or virtualized services. For bare-metal loads it directly loads the image on the hardware. VCL also supports both hardware and software virtulization. It uses IBM Extreme Cloud Administration Toolkit [73] for provisioning bare-metal loads. For software virtulizaton it uses hypervisors like Vmware server, KVM etc. Over this infrastructure it has the capability to provide PaaS, SaaS and AaaS. It provides various Linux and Windows environments as a platform for various research purposes. For example the XINU image environment is a custom made linux kernel and is used by students to modify and develop and create a new platform out of it. VCL has hundreds of images for various software and application environments. Based on the privilege level a user can obtain a fixed time reservation or long term reservation for an image. VCL provides a block allocation service wherein a block of nodes can be allotted for a particular purpose. Reservations can vary from single-seat reservations , to group reservations (Server Profile function added in VCL 2.3). Cluster reservations functionality can be used for accessing HPC services through a login node.

2.3.2 VCL Architecture

As mentioned by Averitt et al in [32] there are seven major parts in the architecture as shown in Figure 2.5,

- End User Interface
- Authentication Mechanisms
- VCL Manager
- VCL Database
- Node Manager

- Image Repository and Storage

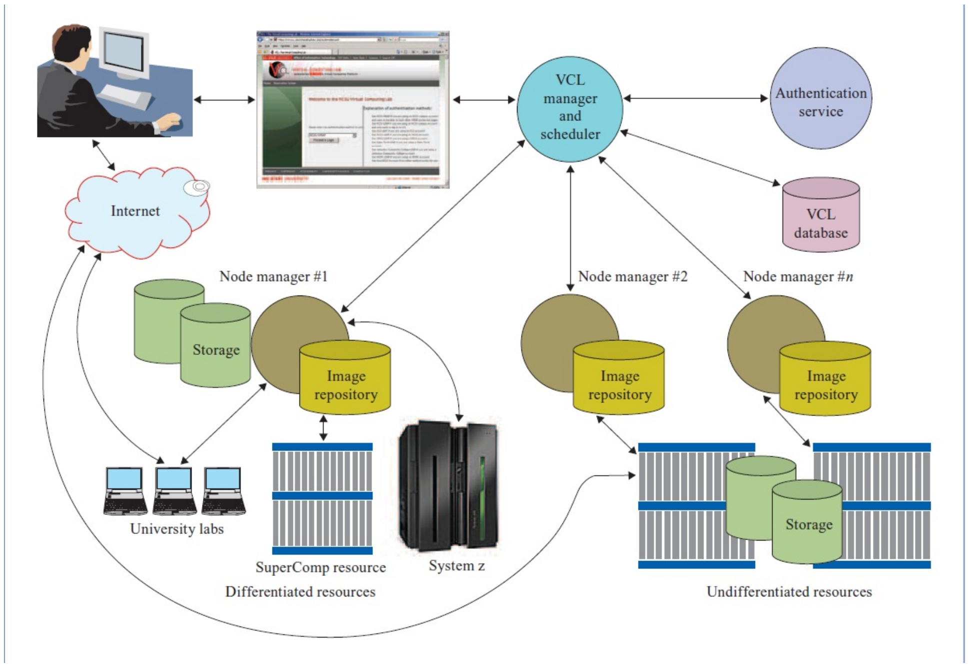


Figure 2.5: VCL Architecture - referred from [32]

End User Interface : This serves as the online portal where users can connect into VCL front end and request for an environment. They have to go through an authentication phase before getting access into the system. Once logged in the user should be able to access the reservation system. The duration for which the image can be requested depends on the privilege level of the user. There are four levels of privileges in VCL going all the way from a basic user to an administrator.

Authentication Mechanisms : VCL uses LDAP authentication [29] for authenticating users. Josh Thompson specifies a set of instructions for adding a new ldap server to VCL in [104]. This mechanism is added to the Web Code by using various php modules. User login details like first name, last name, Ldap server, unity id etc are securely stored in the VCL Database.

VCL Manager : This component manages all the request sent to the scheduler. It also provides the interface to access the User Interface as well as Database through secure authentication services. VCL Manager nodes processes the requests and assigns it to a management node based on the availability of a resource by checking the database.

VCL Database : This is one of the most important component in the VCL Architecture which contains all the necessary information related to resources. VCL classifies its resources into four major categories [106] Images, Computers, Management Nodes and Schedules. These resources are mapped to each other. Information related to each resource is stored in various tables. Apart from resource information it also contains information related to user credentials, requests and reservation that are being processed, or will be processed etc. It also contains various log informations for computers, images etc which can be used for reservation specific analysis.

Node Manager : This node is responsible for managing resources under it. There might be 50 - 100 physical blades which are assigned to each management node which is capable of processing requests for both bare-metal loads and virtual machines. Once the scheduler forwards a reservation request to a management node it is its job to find a computer which can be used for the same. Every management node keeps querying the VCL database at regular intervals of time.

Image Repository and Storage : Storage is an integral part for any cloud environment. VCL uses data-stores on NFS mounted storage and have been looking into using SAN storage for the same. Images created in VCL are saved in these data-stores. Storage disks are also provided for differentiated and undifferentiated resources.

2.3.3 VCL reservation flow

This section describes the basic reservation flow when a user requests for an image in VCL. There are three phases. The first one is a set of steps that are performed by the front end modules to check if the requested resources is available or not. This is explained in Figure 2.6. Once VCL makes sure that the request can be processed, in the second step it again performs various operations as shown in Figure 2.7 and sets up the request and reservation table for the back end process which is described in Figure 2.8.

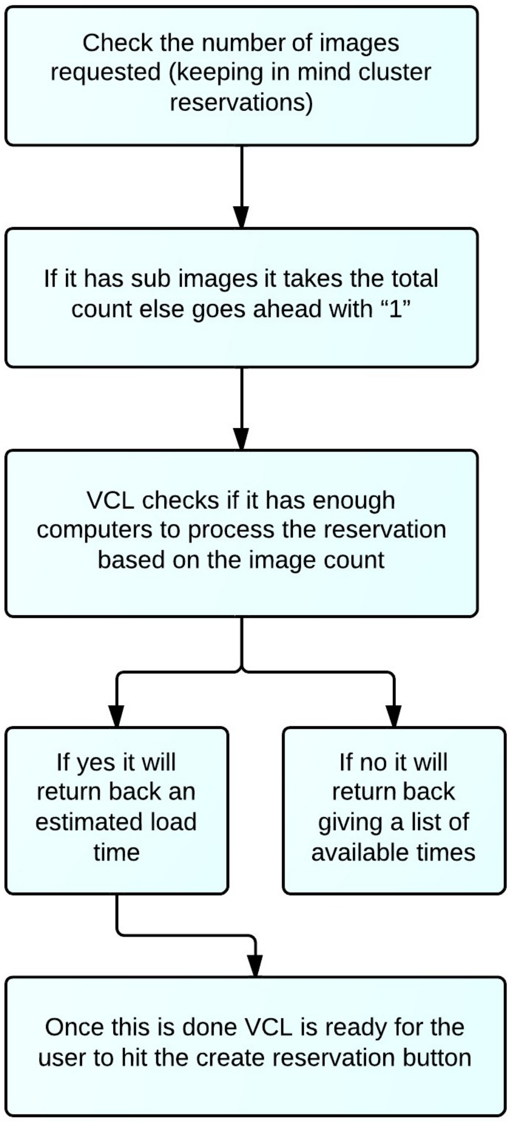

Check the number of images requested (keeping in mind cluster reservations)
If it has sub images it takes the total count else goes ahead with "1"
VCL checks if it has enough computers to process the reservation based on the image count
If yes it will return back an estimated load time
If no it will return back giving a list of available times
Once this is done VCL is ready for the user to hit the create reservation button

Figure 2.6: VCL reservation Flow - Image Selection

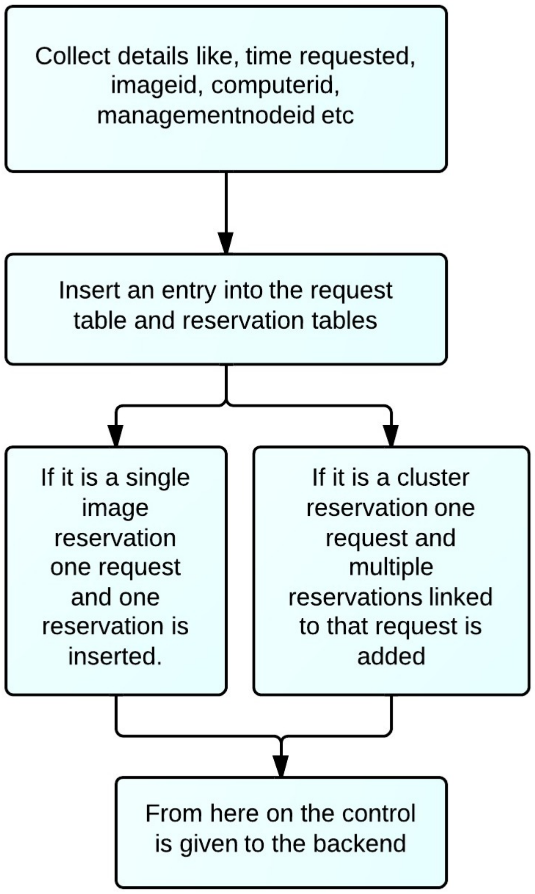

Collect details like, time requested, imageid, computerid, managementnodeid etc
Insert an entry into the request table and reservation tables
If it is a single image reservation one request and one reservation is inserted.
If it is a cluster reservation one request and multiple reservations linked to that request is added
From here on the control is given to the backend

Figure 2.7: VCL reservation Flow - Create Reservation

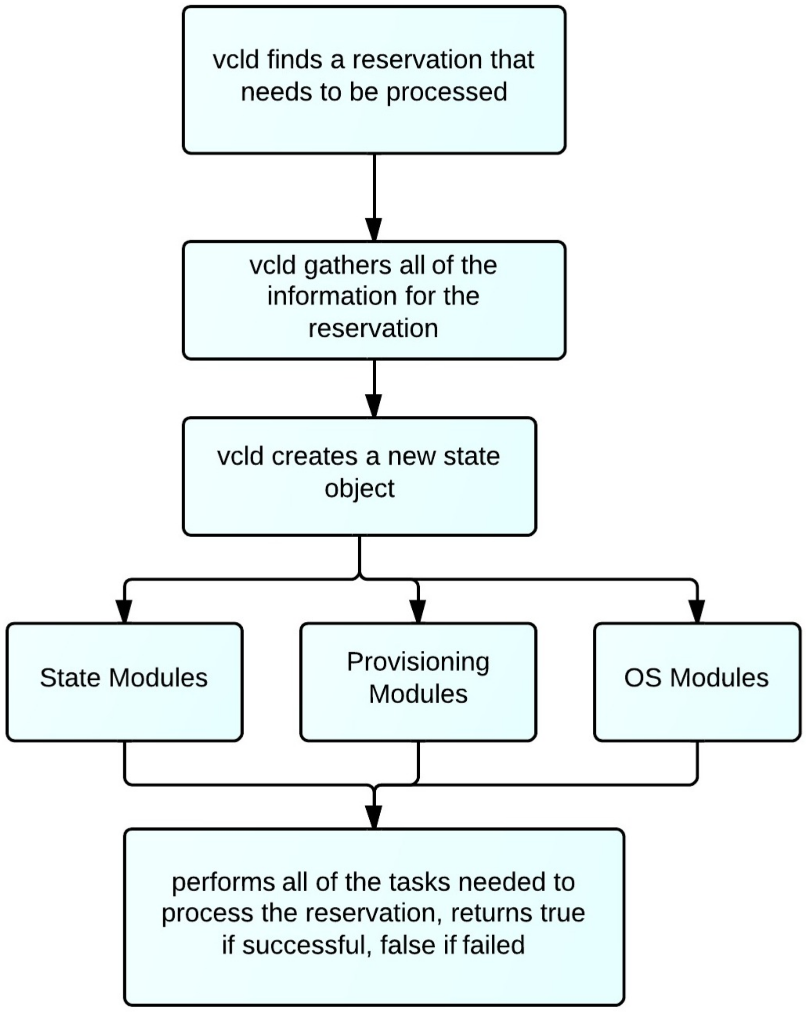

vcld finds a reservation that needs to be processed
vcld gathers all of the information for the reservation
vcld creates a new state object
State Modules
Provisioning Modules
OS Modules
performs all of the tasks needed to process the reservation, returns true if successful, false if failed

Figure 2.8: VCL reservation Flow - Backend Flow

CHAPTER 3

System Design

As we have seen in the previous chapters, supercomputers have fixed node interconnect architectures and thus do not provide the flexibility to reconfigure them. Thus it becomes beneficial to have a software layer over these nodes using which we can define the interconnectivity between them. In our implementation we use VCL and KittyHawk to develop a software layer over these nodes and thus being able to run scientific workflows which require a custom network to be configured on the underlying nodes. KittyHawk is used as a provisioning agent which is driven by VCL to collect nodes, load an image on them, connect the nodes together in the specified way. For example when we use Vmware hypervisor to load virutal machines it would follow steps like creating a container based on the configuration parameters passed to it, create network interfaces as required and finally load an operating system onto it. A pool of nodes are reserved on a BlueGene environment by running a Job. VCL passes the necessary configuration parameters like the total number of nodes, networks etc to KittyHawk and based on this information a cluster is generated. There are many steps taken by both KittyHawk and VCL during this operation.

As seen in Figure 3.1 and Figure 3.2 we design the system in two phases. As listed in Phase I when we consider a Micro-cloud we need certain configuration parameters like the total number of nodes required, the interconnection between for example topologies like mesh, star, totally connected or even custom networks, the image that needs to be loaded on these nodes etc. Once this configuration is saved, KittyHawk scripts will be triggered to create the environment request. Based on the number of nodes requested KittyHawk decides the number of Jobs to be executed. KittyHawk utility scripts in turn communicate back with VCL and updates on resource details. It will kickstart a management node for each of the Micro-cloud. All these details are recorded in the database.

Phase II describe post management node reservation process. Once phase I is completed, dedicated set of management node groups, computer groups are added in VCL and all the BlueGene nodes are added into these groups. The next part will be to go through the vcl template reservation phase. Users should have the flexibility of choosing the template they wish to reserve from the pool of templates. Once the reservation button is hit, the management node running on the BlueGene node is notified that a reservation needs to be processed. When the nodes are reserved as a cluster reservation, one of the nodes will be assigned as parent node and the rest as children. New provisioning and operating system modules which are specific for BlueGene nodes interact with KittyHawk scripts and process the reservation. Once all the requested nodes are reserved vcl web interface returns with a ”connect” button which provides the IP address and login information for each of the nodes in the cluster. In the following sections we will see a detailed explanation

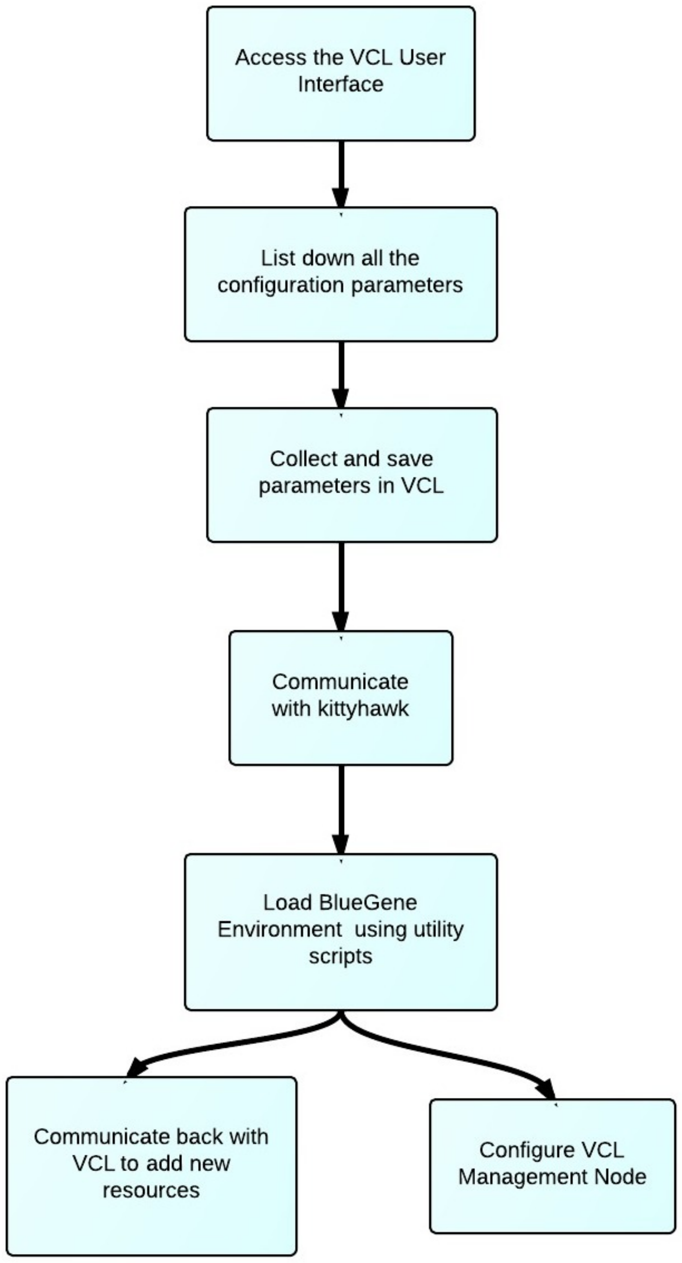

Access the VCL User Interface
List down all the configuration parameters
Collect and save parameters in VCL
Communicate with kittyhawk
Load BlueGene Environment using utility scripts
Communicate back with VCL to add new resources
Configure VCL Management Node

Figure 3.1: Design Flow Phase I

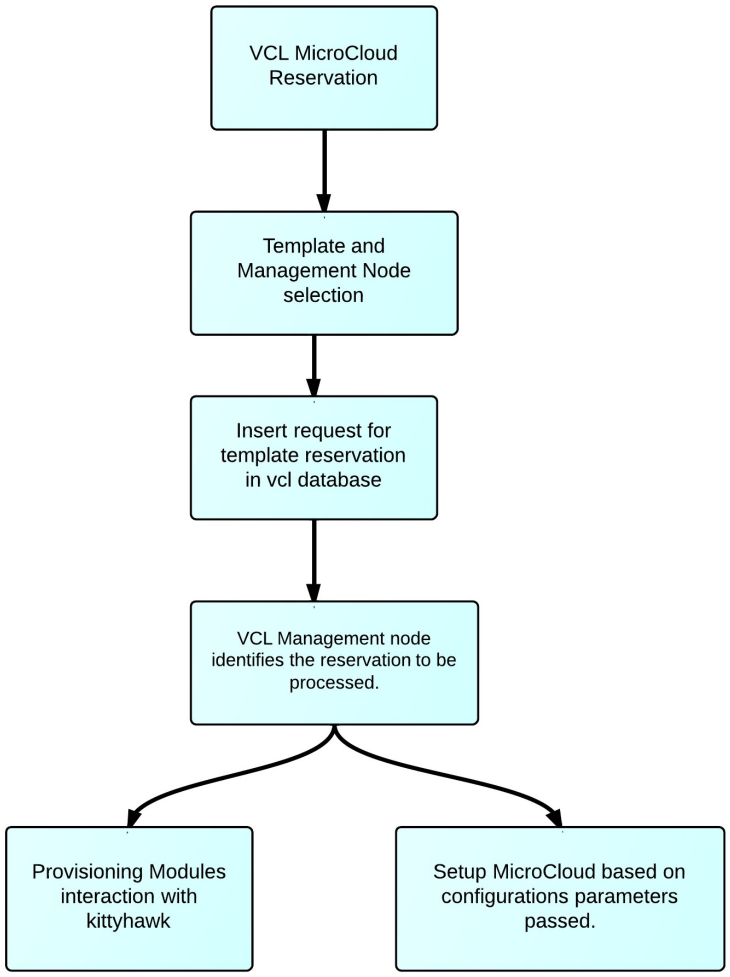

VCL MicroCloud Reservation
Template and Management Node selection
Insert request for template reservation in vcl database
VCL Management node identifies the reservation to be processed.
Provisioning Modules interaction with kittyhawk
Setup MicroCloud based on configurations parameters passed.

Figure 3.2: Design Flow Phase II

of how both KittyHawk and VCL modules have been designed and integrated with the already existing system.

3.1 KittyHawk Design

KittyHawk provides a software platform over BlueGene/P over which we can run heterogeneous workloads. Using it we can load numerous instances of Linux on BlueGene nodes. A debian linux image is loaded on all the nodes. These nodes can be considered as resources with a few cores, memory and network interface for communicating with the control server and other nodes. Along with the kernel image a small bootstrapped ramdisk is loaded onto these nodes. A ramdisk is an image of the root file system. This has the bare minimum applications installed on it for the specific purpose. All these nodes are raw nodes have nothing installed on them. KittyHawk loads a firmware on these nodes and a bootloader uBoot is brought up on these nodes and then finally an operating system to get the nodes operational. All these functionalities are performed by various KittyHawk utility scripts. Lets have a look at them in detail.

3.1.1 Workflow on BlueGene/P

In this section first we will walk through as to how KittyHawk sets up a BlueGene node as fully functional resource. It performs the following steps for the same.

- Job Submission on BlueGene Environment
- Loading the KittyHawk control server
- Configuring nodes with custom interfaces

- Loading kernel image on the nodes
- Loading ramdisk on the nodes
- Loading Lenny / L4 kernel image on the nodes (Lenny is a stripped down debian image with basic essential linux packages for a powerpc architecture)

Job Submission on BlueGene Environment : In BlueGene Environment an application can be executed on the nodes through a Job submission. Cobalt-mprium job submission is used for the same and the command to be issued is “qsub (number of nodes) (time in minutes) (project name) (other options) (script)”. As we can see we can mention various details while requesting a pool of nodes. Script is the file that needs to be executed on the nodes. Other options include configuration options like the network connectivity like torus networks, mesh etc, execution mode, queue to be used. All these details are explained in [82]. KittyHawk uses script “khqsub” for Job submission. This is a script which is written as a wrapper around qsub. When executed we just need to mention the total number of nodes and time in minutes. For eg. if we need a pool of 64 nodes for 60 minutes the command would be
Command: khqsub 60 64

The script then builds the cobalt-mprium submission command based on the command line arguments and environment variables and the command issued is
Command: qsub -n 64 -t 60 -e stderr -o stdout –debuglog debuglog -q default -A VCL-ON-BG_P –kernel kh /home/georgy/null.elf

The option -A is used to specify the project which in our case is VCL on BGP. This is picked up by khqsub script from the KittyHawk environment variable that are loaded into the shell before running this command. This job is executed on the login node and it returns back with a compute node partition exclusively for the user.

Loading the KittyHawk control server : At the end of the job, a KittyHawk control server node is configured to control all the further requests. As mentioned above a compute node partition of 64 nodes is allocated to a particular user and is not shared with other users who are also running a BlueGene Job. khqsub returns back the IP address of the KittyHawk control server which needs to be exported at the shell.
Command: export khctlserver=(IP)
The environment variable khctlserver is now set and thus can be used by other utility scripts for further processes. Interaction between the scripts on the login node and the khclserver uses key based authentication. During the Job submission itself, the keys are loaded into khctlserver for password-less ssh access.

Configuring nodes with custom interfaces : Once we have the pool of nodes we can provide a custom configuration for each of the nodes and load them with a linux kernel on them. The first step is to get nodes and pass network interfaces parameters. The command used for this "khget (options) (number of nodes)". The options are as follows

- -x : to add the node to the external network
- -i : to add the node to the internal network
- user : user to get the nodes for
- -p : Number of new private networks to be created
- -n : (netid) Assigning the node to already existing private networks

Thus if we want to request for a single node the command to be issued is node="$(khget -x -i $USER 1)". node is an shell environment variable assigned and $USER will configure the node for the currently logged in user. The options provided are -x and -i, this will add two network interfaces for the node one for the external network and the second for external network. Option -p is used when we want to add private custom networks. Each time a private network is created there is a "netid" associated with each of the network. Thus later on if we wish to add a node to the existing private network we can just specify this netid using the -n option. In our implementation these options are extensively used as we need to created custom private networks and add nodes to them.

Loading kernel image : Once the node is requested, KittyHawk loads a BlueGene/P specific firmware on each node using the Global secure control network. Once the firmware is loaded the boot loader is brought up on the nodes. The next step is to load a kernel image on these nodes "uImage". This is again loaded using the "khdo" utility script which has a function called as loadkernel. These images are used with uBoot Boot-loaders and they contain information about the linux kernel, root file system, firmware etc. Once this is done, the node is up and running for use. We can transfer the public key into the node and then perform ssh based on public-private key based authentication.

Loading ramdisk on the nodes : As mentioned earlier ramdisk is an image of the root file system. We can configure any application as a ramdisk file and then load it on the node. For example we can have a ramdisk with sshd configured on it and thus after loading it on the node we will be able to ssh into it. In the basic setup KittyHawk loads an nbd ramdisk onto the nodes. nbd is network block device which is used in cases where we have low disk space on the nodes and like NFS we can use it to load a file system remotely. nbd server is intitated on the login surveyor node and an nbd-client ramdisk is loaded on the Bluegene Nodes which connects to the server. It gives the notion of a disk loaded locally on the node. The debian lenny image is loaded on the disk which is then mounted and accessed by the nodes using nbd-client. This is a persistent disk and thus any changes made to the image will be changed. Loading the ramdisk is managed by "khdo" utility script which has a function called loadramdisk.

khdev utility script : This entire procedure can be done through a single utility script called as "khdev". It does all the functions that are mentioned above like requesting node, loading uImage, loading ramdisk and finally setup nbd-server and client and load lenny image to create a remote disk for the nodes. keys are also loaded in the nodes via the script so that further ssh communication can happen via key based authentication. The command that is issued is : khdev -K "$(cat /.ssh/id_rsa.pub)" (path-to-lenny image). -K option specifies the key to be loaded on the node and the second parameter is the path of the lenny image. But there is a limitation in this script. It configures a node with only two interfaces, one for public interface and other for internal network. We wont be able to configure custom private network interfaces. Figure 3.3 provides a diagrammatic representation of the entire process flow.

3.1.2 Configuring Modules

In the previous section we have seen the KittyHawk process flow and the various steps it undertakes to configure a node completely. All the individual steps are put together into a single utility script "khdev". But we need to create new scripts based on the system

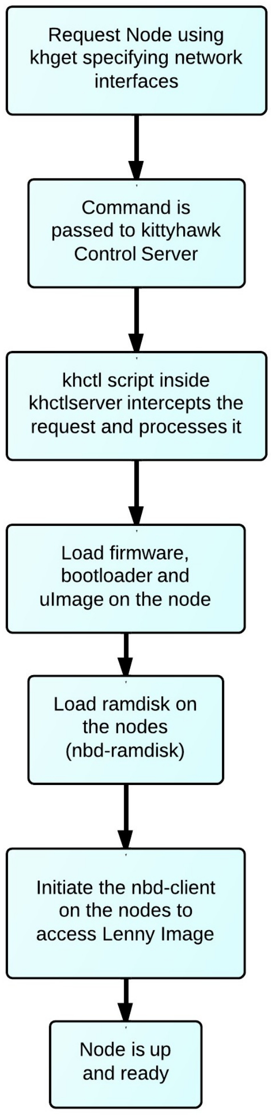

Request Node using khget specifying network interfaces
Command is passed to kittyhawk Control Server
khctl script inside khctlserver intercepts the request and processes it
Load firmware, bootloader and uImage on the node
Load ramdisk on the nodes (nbd-ramdisk)
Initiate the nbd-client on the nodes to access Lenny Image
Node is up and ready

Figure 3.3: KittyHawk workflow on BlueGene/P

requirement. The default options which khdev provide are as follows:

- -d : Perform Debug during script execution
- -k : kernel image to be loaded overriding default one.
- <root disk image>: Location of the Lenny image
- -r : ramdisk to be loaded overriding default one.
- -K : Public ssh key to be loaded on the node to provide root access

By default this assumes that the node needs to be added on the public (-x) and private (-i) network. But in a cluster network this is not true we need more custom private networks and any node can be a part of any network.

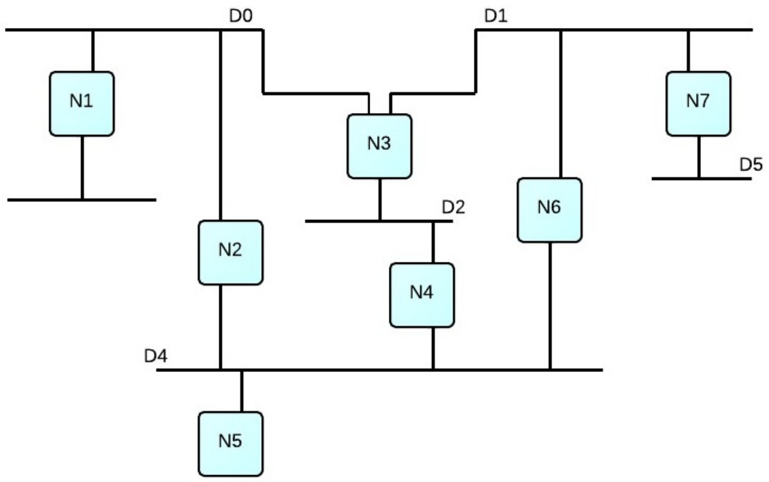


Figure 3.4: Cluster Example on BlueGene/P

As we can see in Figure 3.4 there are 7 nodes are each of them are connected to different networks. In order to design such a configuration we need to assign these custom networks as private networks in KittyHawk. Also extending further VCL uses two networks for managing resources, one public for external access to users and the second an internal private network for back-door management. Thus considering the integration with VCL we need to design the networks based on it. The public network on BlueGene (-x option) is used for external access and the internal private (10.x.x.x) network (-i option) is used for back-door management network. In Figure 3.4 there are 6 custom networks, first we need to create these networks using -p option and a specific netid will be assigned to each network and later we can assign the netid's to nodes. We add these option features and create new khdev scritps.

While creating a cluster one node is assigned as the management node which runs VCL and configures the remaining cluster. Cluster template configuration is provided by the user form VCL Web Interface. This specifies the total number of private networks. These networks are configured when the management node is created. We create a new script called as "khdev_mn", with new options added for "-p" which can be specified as command line argument while initiating the script. Command : khdev -p (no of custom networks) -K "$(cat /.ssh/id_rsa.pub)" (path-to-lenny image). Thus when the management node is requested the function gethost() while requests for a node send the following command to the khct control server. Command:node="$(khget -x -i -p (number)$USER 1)". Once the management node is up and running it will have all the networks configured on it. During further template reservation which is processed by the management node, the new nodes needs to be attached to the custom private networks by using the netid's. In order to provide this we add another options field "-n" in the new khdev script. For eg. khdev -n 2,3,4 -K "$(cat /.ssh/id_rsa.pub)". This will request for a node which needs to be added to custom networks with network id 2,3 and 4. Hence based on the information specified in the template nodes will be configured based on these scripts controlled by the

VCL provisioning modules.

3.2 VCL Desgin - Front End and Back End

As seen in the VCL Architecture in the previous chapter there are three main components. MySQL database which contains all the information about various physical and virtual resource, images, user groups and role based privilege access lists. We need to add new tables into database to support the new functionality. Details will be selected and inserted from and into the database by both the front-end and back-end VCL modules. VCL front-end provides User Interface functionality for users. Apart from the regular tabs in VCL dashboard we will add a couple of tabs and access to these can be restricted using privilege tree functionality in VCL. Front end modules are php based and dynamic update of contents on a page are done using JavaScript and AJAX. We will stick to the modularized architecture and add new modules based on the existing code structure. VCL uses continuations [105] to control the sequence of pages. This provides additional security where users will not be able to access pages where they are not authorized to or perform man in the middle attacks. We will see that BlueGene Nodes are added into VCL as specific computers. In VCL we have a type field associated with each computer. Type "virtual machine" is assigned to virtual resources, "blade" is assigned to physical resources. Each type is associated with a specific provisioning and OS modules for back-end provisioning. In our system we create a new computer type called "bgp" and create new provisioning modules which are linked to BlueGene Nodes. Finally we need to add modules which will manage the interaction between VCL and KittyHawk utility scripts and thus providing an environment which helps to configure software defined networking.

3.2.1 User Interface

VCL provides a very structured User Interface for managing the resources. Resources are grouped into computers, images, management nodes, schedules etc. The various resource attributes are mentioned in [106]. As we can see in Figure 3.5 there are tabs that are provided for managing each of these resources. Computers are managed through “Managed Resources”, images are managed through “Manage Images” etc.

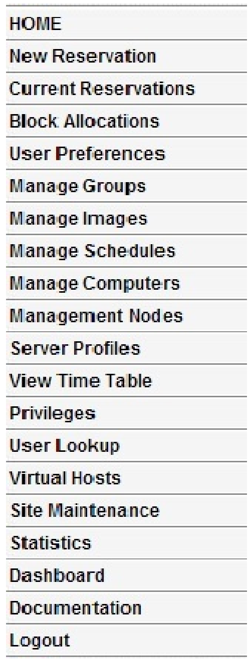


Figure 3.5: VCL User Interface Tabs

Each tab has its own php file which displays and manages the different functionalities. For dynamic update JavaScript functionality is used. For our system we will create two additional tabs. First one called as ”HPC-BGP” and the second one ”HPC-Template”. All the functionalities will be provided by these two tabs and they are managed by a php file which contains all the functions and a JavaScript to take care of dynamic updates. Already existing modularized architecture of VCL code is maintained while adding these two tabs. Once these tabs are created we need need to decide the functions that they need to provide. For our system we know that before we request for a pool of nodes we need to create a topology or a template which will have all the custom configuration parameters. Once this is done a pool of nodes can be requested and the management node can be configured. Finally we should be able to reserve the template that we created initially which is send as a cluster reservation to the management node. Hence we need to provide four functions in total which are as follows:

- Create Template
- Management Node and Pool Reservation
- Template Reservation
- Delete Base Node and Pool

The first functionality to be added is ”Template Creation”. While defining the template the user needs to provide two details one the total number of nodes required and second the total number of custom networks. Based on the information provided a two dimensional matrix of nodes v/s networks is provided to the user. The user can now map between the nodes and the corresponding network need for each of them. Table 3.1 shows the mapping between nodes and networks. As mentioned in the section above we have two mandatory network interfaces for each node external network and internal network.

This mapping information is saved in the database table. The second functionality to be provided is triggering a BlueGene Job and reserving a Management Node. It is important to note that every-time a template is created a corresponding management

Table 3.1: Mapping between nodes and custom networks

Nodes	Ext Net	Int Net	Network 1	Network 2	Network 3	Network 4
1	✓	✓	✓	-	-	✓
2	✓	✓	-	✓	✓	-
3	✓	✓	✓	-	✓	-
4	✓	✓	-	✓	-	✓

node is inserted into the queue for reservation. In this section the user can select any management node from the queue and trigger a BlueGene Job. Once the management node is up and running it can process the reservation for the template linked with it. Every template will have its own management node. But it is possible that we have a pool of 64 nodes and we are only using 10 out of them. Thus it is not ideal to waste the remaining 54 nodes and request for another pool for a new template. Here we add an additional feature of linking the management node to an already running BlueGene Job. This is implemented using JavaScript where it will query the database and return the time remaining for the existing Job. Finally the template can be reserved using the option "Template Reservation". Here the user will have to select the Base Node / Management Node which will load the template linked with the same. We can then select the application which to be loaded on these nodes. Delete Base Node provides the functionality of deleting the nodes that were reserved and thus freeing up the resources once the scheduled work is done. The figures below list the various functionalities.

Now that we have an overview of the various functions that are incorporated at the User Interface level let us see the process flow in detail. Function flows and passing arguments over function calls are managed by continuations in VCL. For every continuation we can specifically mention which function it needs to go and which php file. This provides a great flexibility while designing. As shown in Figure 3.6 we have four basic functions and Table

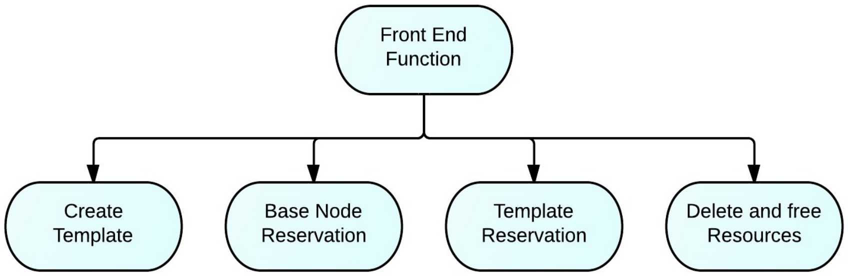


Figure 3.6: HPC-BGP Tab functions

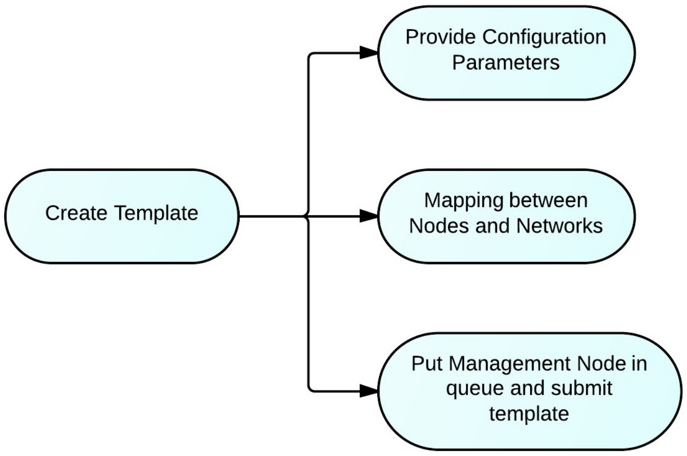

Create Template
Provide Configuration Parameters
Mapping between Nodes and Networks
Put Management Node in queue and submit template

Figure 3.7: HPC-BGP Function 1 - Template Creation

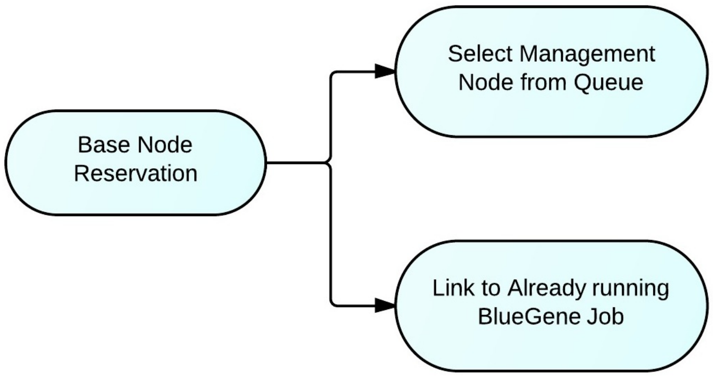

Base Node Reservation
Select Management Node from Queue
Link to Already running BlueGene Job

Figure 3.8: HPC-BGP Function 2 - Based node Reservation

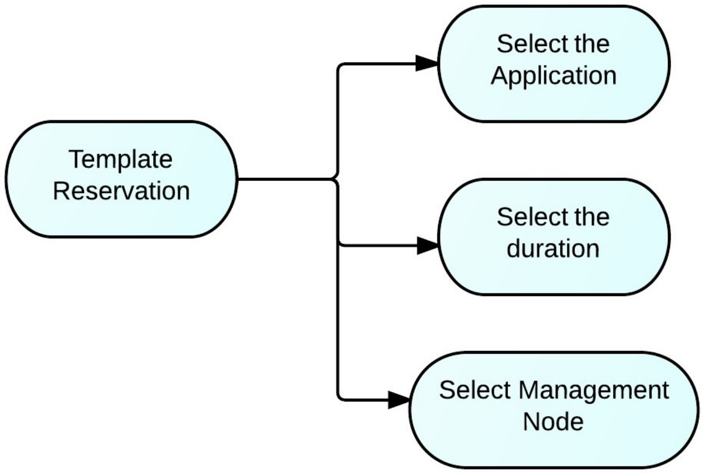


Figure 3.9: HPC-BGP Function 3 - Template Reservation

3.1 depicts how continuations are mapped. For eg. when a user selects the create template option, a new continuation entry "option1" is added in the list and it is already defined the function to which the control should be redirected when the continuation is submitted. In this case the user will be taken to a new page function - Template_Creation. We can use the same functionality to even pass arguments to functions. Figure 4.1, Figure 4.2, Figure 4.3, Figure 4.4 explain the process flow for each of the functionalities and what happens at each step.

Table 3.2: Using Continuations for Function calls

Option	Continuation	Function called
create template	option1	Template_Creation
Reserve Base Node	option2	Base_reserv
Reserve Template	option3	Template_Reservation
Delete Job	option4	Delete

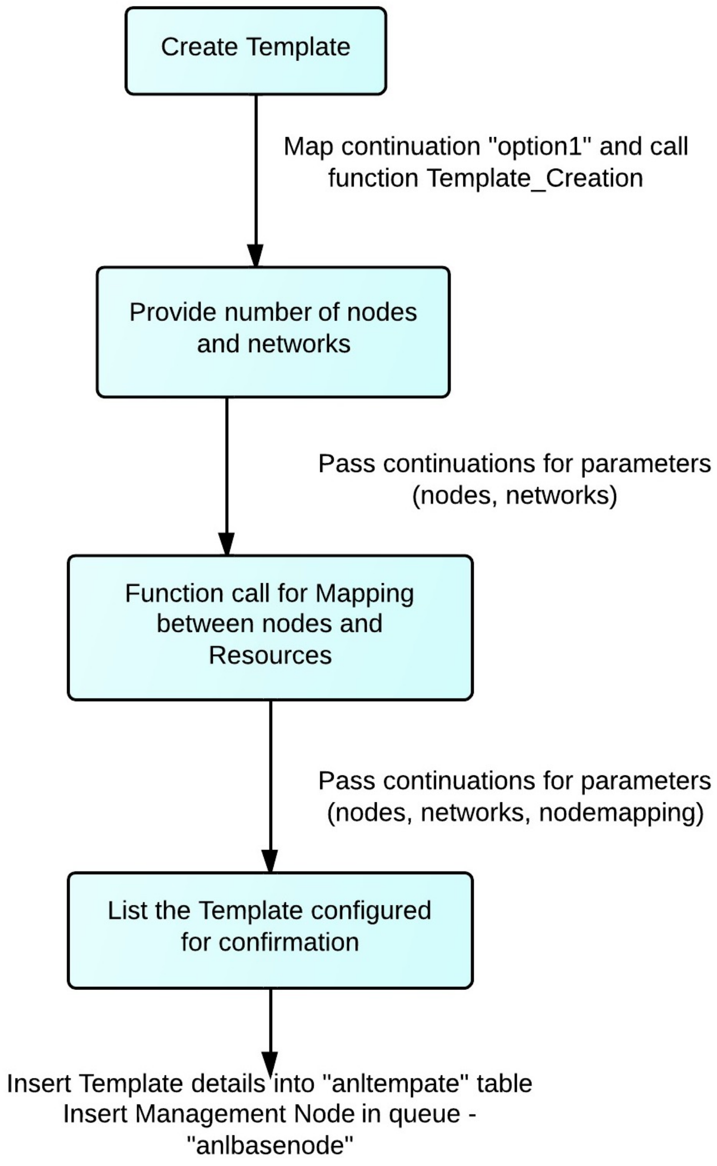

Create Template
Map continuation "option1" and call function Template_Creation
Provide number of nodes and networks
Pass continuations for parameters (nodes, networks)
Function call for Mapping between nodes and Resources
Pass continuations for parameters (nodes, networks, nodemapping)
List the Template configured for confirmation
Insert Template details into "anltempate" table
Insert Management Node in queue - "anlbasenode"

Figure 3.10: HPC-BGP Function Flow 1 - Template Creation

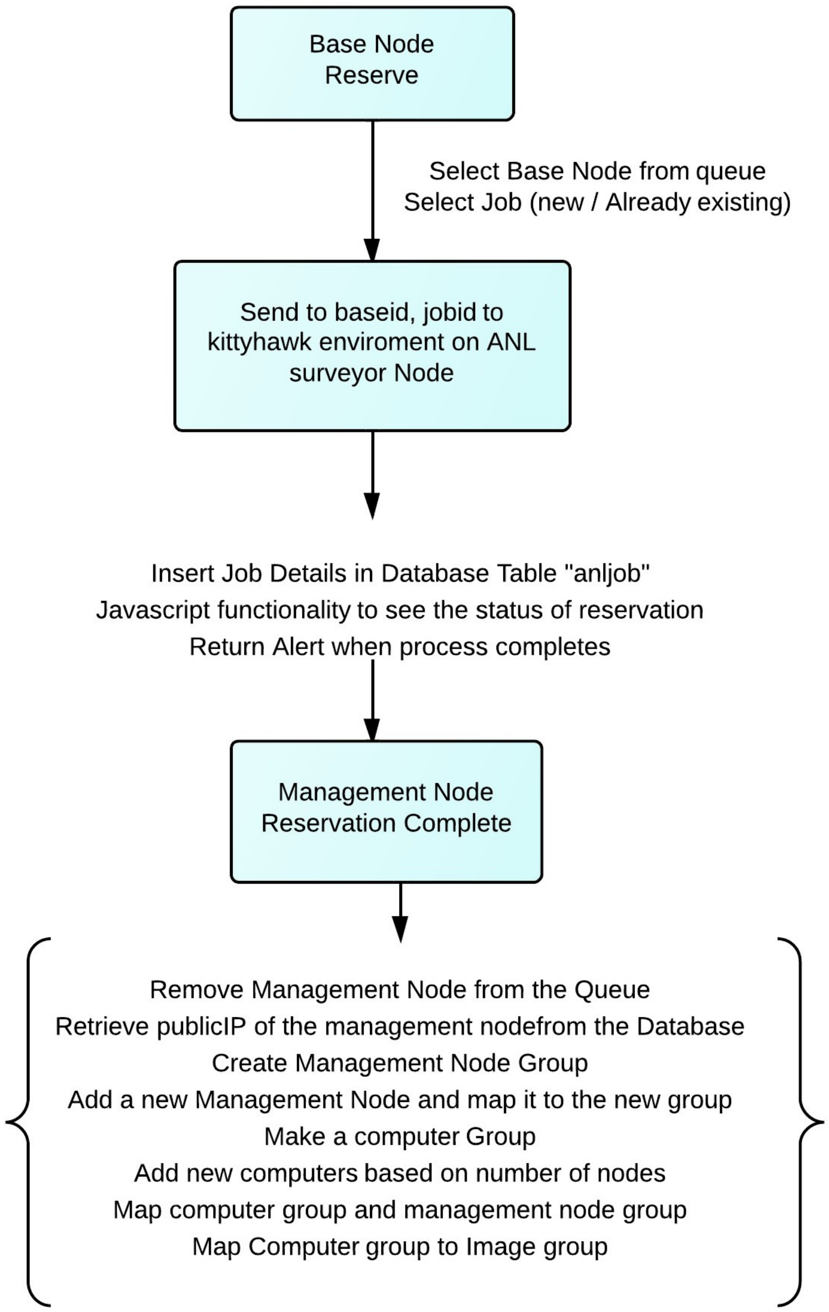

Base Node Reserve
Select Base Node from queue
Select Job (new / Already existing)
Send to baseid, jobid to kittyhawk enviroment on ANL surveyor Node
Insert Job Details in Database Table "anljob"
Javascript functionality to see the status of reservation
Return Alert when process completes
Management Node Reservation Complete
Remove Management Node from the Queue
Retrieve publicIP of the management nodefrom the Database
Create Management Node Group
Add a new Management Node and map it to the new group
Make a computer Group
Add new computers based on number of nodes
Map computer group and management node group
Map Computer group to Image group

Figure 3.11: HPC-BGP Function Flow 2 - Base Node Reservation

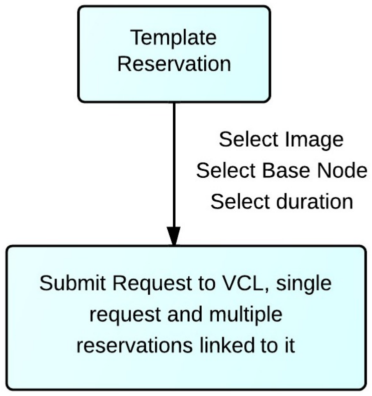

Template Reservation
Select Image
Select Base Node
Select duration
Submit Request to VCL, single request and multiple reservations linked to it

Figure 3.12: HPC-BGP Function Flow 3 - Template Reservation

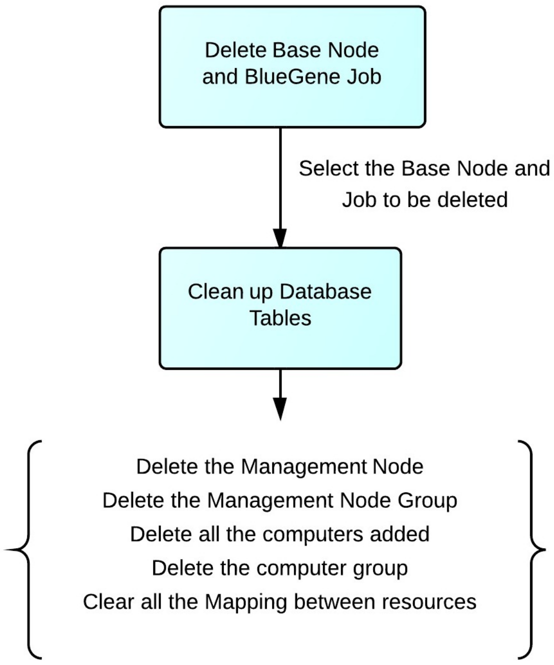

Delete Base Node and BlueGene Job
Select the Base Node and Job to be deleted
Clean up Database Tables
Delete the Management Node
Delete the Management Node Group
Delete all the computers added
Delete the computer group
Clear all the Mapping between resources

Figure 3.13: HPC-BGP Function Flow 4 - Delete Resources

3.2.2 Database Modules

In this section we will look at the various tables that have to be configured. Database serves as a mid interface for the communication between Front End and Back End Modules. All the details that are provided by the user from the Front End are saved into the database which is then fetched by the backend modules and utilized. As mentioned in the previous section each template is managed by a management node. Table 3.3 shows the various details that are recorded for Management Node. Once the user creates a template, the new management node will be added to this table with the queue field as "1". Once the Management Node is reserved details like PublicIP, PrivateIP etc are added. The queue status is changed from "1" to "2". Every time a BlueGene Job is triggered on the surveyor node it is assigned a job id and the management node being a node in the pool the jobid field signifies the job linked to it. nbdid is nothing but the Network Block Device id of the server that is triggered to load the lenny image on the node.

Table 3.3: Table - anlbasenode : Details of ANL Management Node

id	PublicIP	PrivateIP	image	inqueue	reserved	jobid	nbdid
1	172.16.13.2	10.10.0.14	Lenny_MN	1/2	0/1	1	4563

Next Table 3.3 contains all the details of the currently running BlueGene Job. Every job has a maximum running time of 60 minutes which is specified in the starttime and endtime details. The total number of nodes assigned are 64 and nodesused helps in figuring out how many nodes or resources are left that can be used. khcltserver is the IP Address of the KittyHawk control server which processes all the request. While the base node reservation is being processed we have added the javascript functionality to

Table 3.4: Table - anljob : Details of ANL BlueGene Job

id	khctlserver	starttime	startstamp	endtime	nodes	nodesused	qsubid
1	172.16.13.7	00:00:00	1568	00:00:00	64	10	3356

provide the load status information. Table 3.4 consists of all the log messages that needs to be displayed based on the current status. “status” field when set to 0 indicates that the step is yet to be done and 1 indicates that it is completed. Anl provision table is used

Table 3.5: Table - anlloadlog : Details of status of Management Node Reservation

id	message	status
1	Loading the KittyHawk Environment Paths	0/1

to maintain the provisioning status of nodes that are provisioned by the management node. If the process still is going on then the “status section” will reflect a value 1 instead of the default 0. All the details regarding the template configured by the user is stored

Table 3.6: Table - anlprovision : Provisioning Status of Template Reservation

provision	status
khdev	0/1

in Table 3.7. It contains the management node id to which the template is linked. The total number of nodes and custom networks requested. The “members” field contains the mapping between nodes and networks. Once this template is reserved the field “added” gets updated to 1.

Table 3.7: Table - anltempalte : Tempalte Configuration Details

id	basenodeid	nodes	networks	members	added
1	2	10	5	array()	0/1

Once the template reservation request is placed in the database tables 3.8 and 3.7 are used to manage the details. When a cluster is reserved the reservation is processed as parent and child reservations. So we store the parentid linked to the management node. And the computer details like network block device id which is the process id which runs

Table 3.8: Table - anlrequest : Reservation Request Details

id	basenodeid	parentid
1	0/1	5

the network block server to load the lenny image, and the network interfaces id specific to the computer being reserved are stored in “anltocomputer”. We add two new modules in

Table 3.9: Table - anltocomputer : ANL Computer Details

id	computerid	basenodeid	nbdid	netid
1	2	1	5689	interface list

VCL one for Operating System Management and Provisioning the resources. They need to be configured and specified in the database tables. We will see in the next section how they are configured and designed to suit our requirements. As we can see in Table 3.10 provisioning_bgp module is the provisioning module use for bringing up the BGP nodes

Table 3.10: Table - Modules List

id	name	prettyname	description
1	provisioning_bgp	bgp provisioning	Module for BGP Computers
2	os_bgp	bgp OS Module	Module for OS

and os_bgp will control the post load processes on these resources.

3.2.3 Provisioning Modules

These modules control the image load flow in VCL. Once we request for a reservation it is managed by state variables which passes the control to the designated provisioning modules. Once the node is up and the Operating System has booted up the control passes to OS Modules which perform various post load functions. For eg while creating a reservation for an image which is loaded on a virtual machine, VCL uses the provisioning module VMware.pm to create a virutal machine with the specified configurations and then load the image. Once the node is up all post load processing is performed by the OS.pm module like adding a user, creating user group etc. In the same manner we need to configure new modules for our system. We consider the BlueGene Nodes as computer resources in VCL. There are three primary computer types, virtual machine, blade and lab. Each type has a corresponding Provisioning Module associated with it. Computers of the type virtual machine are controlled by VMware modules or KVM modules. Once we add a new computer in the database and assign the type virtual machines, VCL understands the modules to be loaded for the computer once it is assigned a reservation. All these informations are stored in various tables like Modules, Provisioning etc. We need to interact with KittyHawk control server in order to boot BlueGene Nodes and load lenny images onto them. We load these commands into a new provisioning module called

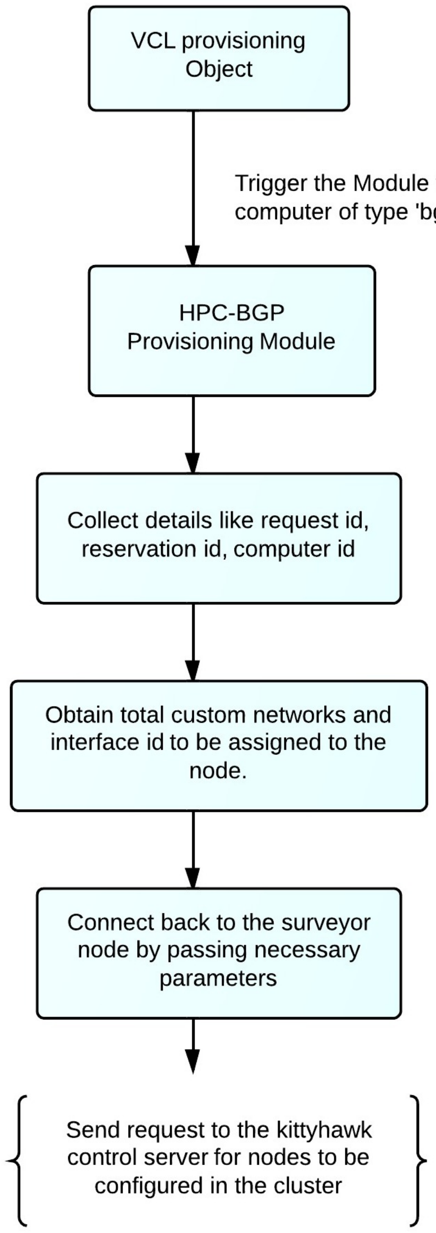

VCL provisioning Object
Trigger the Module for computer of type 'bgp'
HPC-BGP Provisioning Module
Collect details like request id, reservation id, computer id
Obtain total custom networks and interface id to be assigned to the node.
Connect back to the surveyor node by passing necessary parameters
Send request to the kittyhawk control server for nodes to be configured in the cluster

Figure 3.14: HPC-BGP Provisioning Module Flow

as “bgp provisioning” which does the necessary functions for reserving BlueGene Nodes. We add a new computer type in VCL called as ’bgp’ for BlueGene Computer Resources. The computers which represent BlueGene Nodes that are added into the database are assigned this new type ’bgp’. This will automatically set the new provisioning model BGP-HPC for the new resource which will be used during the image load flow reservation process. Figure 3.14 explain the flow of the Provisioning Module

CHAPTER 4

Implementation And System Flow

In the previous chapter we described the design of various components. In this chapter we will see how these modules are put together and integrated and then go over the entire system flow based on a small example. The final goal is to have nodes connected via custom private networks to form a personalized MicroCloud. VCL and KittyHawk form the software abstraction layer over BlueGene/P. It is very important that each module is correctly installed and configured based on the system design. Installation is done keeping in mind security concerns. All the interaction between VCL and KittyHawk happens through secure tunnels via port forwarding using key based authentication. All the processes running on ANL Surveyor node are local and cannot be accessed from the external network. Modules are installed in local space and do not require root access. There are two main sections in this chapter. First we will see how the individual modules are configured and integrated together. In the next section we will consider a small example and run through the entire system flow and which will provide an understanding on what happens at each stage.

4.1 Configuration and Implementation

In this section we will see how various modules are installed and configured. As we have seen, VCL Design is done in three sections: front-end, database and back-end along with provisioning modules. New modules and tabs are added to the VCL Front-end code following the modularized architecture of VCL. Back-end modules are also configured in the same manner. KittyHawk provides various utility scripts for configuring BlueGene Nodes. These scripts were tweaked based on VCL requirements. It is a one to one exchange of commands between VCL and KittyHawk while creating a Micro-cloud. Failure at any step should be properly intercepted and necessary actions need to be taken by both VCL and Kitthawk. Let us take a detailed look.

4.1.1 Module Installation and Configuration

Detailed Installation user manual is included in the Appendix A.1. MySQL database is installed on the ANL Surveyor Login node in user local space. VCL has installation instructions for CentOS and Fedora Distributions. The surveyor node is an openSUSE distribution and thus installation steps had to be tweaked. MySQL database is installed from a binary source and is configured to run on 127.0.0.1 local interface on port 3306. This is done by editing the my.cnf configuration file. Password-less access is given to the user account and root user is able to access the database using password. A new mysqld daemon is also created to suit the new installation. Path Environment variables are also edited so that MySQL installed in the local space is picked up instead of the one installed under root user. A new database called ”vcl” is created and all the tables along with the newly designed ones are loaded into this database. A stripped down debian image “Lenny Image” is loaded on the BlueGene Nodes. Each configured Micro-cloud being managed by a management node needs to have a specific lenny image with VCL and necessary perl modules installed on it. Installation of these modules are based on the instruction mentioned in Appendix A.2. VCL daemon installed on the lenny image keeps polling the database on the surveyor node for template reservation requests. Provisioning Modules interact with the KittyHawk control server while provisioning BlueGene nodes. KittyHawk scripts are modified to add the feature for adding custom network options. In VCL the management node uses two networks to control resources. One a public interface for external connectivity and second a private network as a back-door interface to manage resources. In BlueGene Environment it uses the 172.x.x.x interface as public and uses 10.x.x.x as the back-door interface Each node that is booted has these two mandatory interfaces apart from the custom network interfaces. VCL web interface is installed on a sandbox Image at NC state. On this node, apache and all the necessary packages and libraries required for VCL front-end functionality are installed. Configuration files are changed in order to direct php modules to access the database installed on ANL surveyor node. In the next section we will see how these components are integrated together.

4.1.2 System Integration

As explained in the above section, the MySQL database runs locally. First VCL user interface needs to be integrated with the database. We first create an ssh tunnel from the web interface node which is installed at North Carolina State University. ssh tunnel is

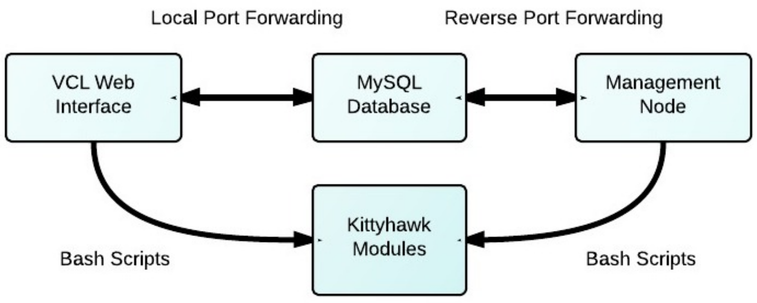


Figure 4.1: Comuunication between Modules

created based on 2048-bit RSA key based authentication between the web interface node and surveyor login node. Local port forwarding for port 3306 is done. All the connections on port 3306 on the web node will be directed to port 3306 of the surveyor node on the IP address 127.0.0.1. This is nothing but the connection to mysql Database. We configure the database login details in vcl secrets.php file. Thus php modules believe that mysql database is installed on the same machine but all the requests are actually forwarded through the tunnel to surveyor node. Similarly the connectivity between vcld on the management node and database is also based on port forwarding. But this time, a reverse port forwarding is performed. After loading the lenny image on the management node, an ssh tunnel is created between the surveyor node and management node with reverse port forwarding on 3306 port. Again the ssh tunnel is configured using key based authentication. Bash scripts control the communication between vcl modules and KittyHawk utility scripts. The first set of command chains is sent from the web interface which is intercepted by bash scripts which then passes the information to KittyHawk scripts. Provisioning modules on the management node also interact with KittyHawk modules in the similar fashion. Once the installation and integration is done the system is ready for an experimental workflow.

4.2 System Flow

In chapter 3, we have seen the main modules of both VCL and KittyHawk and how they have been modified to suit our design. By integrating both of them together we were able to create a software abstraction layer over BluGene/P which provides the flexibility of having a software defined networking over a supercomputer architecture. KittyHawk acts like a provisioning agent for BlueGene Nodes which is configured and controlled through VCL. There are three main phases for creating a Micro-cloud. After making a VCL reservation for "VCL to Surveyor" Image which provides User Interface Functionality, the user has to create a template. This template is based on the problem that needs to be mapped to the BlueGene Nodes. The user has the flexibility of configuring the network interconnection between nodes during template creation which is then stored in the VCL Database. As the template is saved, the user can always come at a later time to reserve the template. As we have seen in the earlier chapter each MicrCloud is managed by a VCL management node. Management Node is configured on one of the BlueGene Nodes out of the pool of 64 nodes that are reserved. This is done using the "Base Node Reservation" functionality added in VCL. During this process the VCL Front End Modules will issue a command to KittyHawk to trigger the process and also keeps a check on the steps that are performed. Once the Base Node has been successfully reserved, user can go ahead with the reservation of Micro-cloud based on the template that he created. This adaptable cloud is configured by the management node which through Provisioning modules interacts with KittyHawk modules and sets up the environment. Let's consider an example. Suppose we want to configure a Micro-cloud which is connected as shown in Figure 4.2.

There are 7 nodes N1 to N7 and there are 6 private networks D0 - D5. Each node is connected to a specific set of networks. Apart from these custom networks, all the nodes

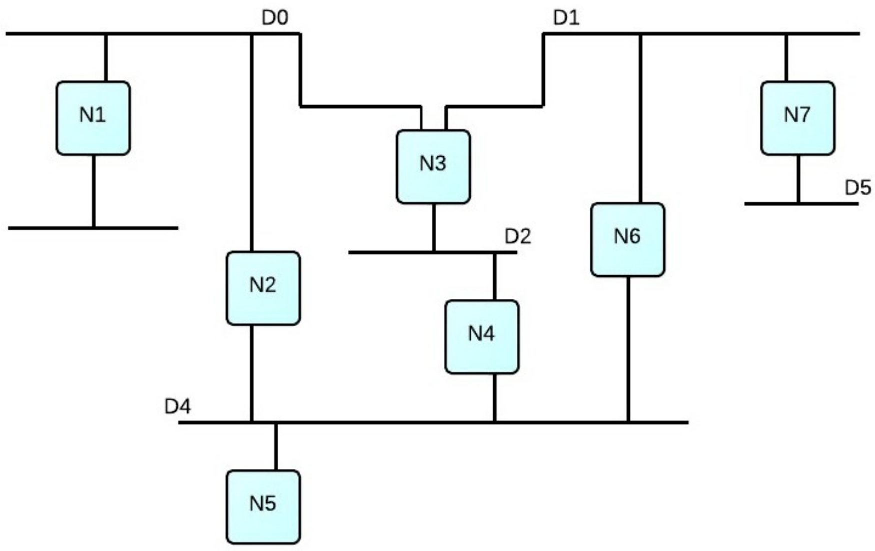
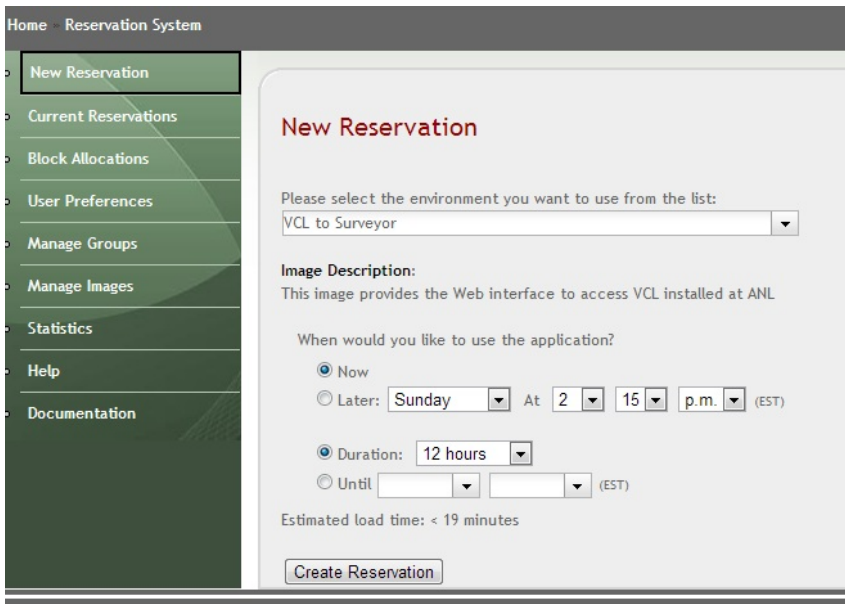


Figure 4.2: Micro-cloud with custom Network

will have two more networks configured in them, an internal public interface (172.x.x.x) and an internal private (backdoor interface) used by the management node to talk to all the nodes in the cluster. As mentioned earlier we have to first create a template where we specify all the configurations and then save it. After this we kickstart a BlueGene environment where a management node is configured which then reserves the cluster network. Let's look at each of these steps one by one.

4.2.1 Template Creation

Here we need to reserve the Image "VCL to Surveyor" from vcl.ncsu.edu HomePage as seen in Figure 4.3. The interaction between Web Interface installed at NC State and Surveyor Login Node at ANL happens through a secure tunnel. Hence before we start the Web Interface, we need to login into the Web Node via shell and execute the tunnel creation step as mentioned in Appendix A.1.3 Web Interface Configuration . We can now connect to the Surveyor Web Login Page by pointing the browser to the IP address returned by the reservation.



New Reservation

Please select the environment you want to use from the list:
VCL to Surveyor

Image Description:
This image provides the Web interface to access VCL installed at ANL

When would you like to use the application?
● Now
○ Later: Sunday At 2 15 p.m. (EST)

● Duration: 12 hours
○ Until (EST)

Estimated load time: < 19 minutes

Figure 4.3: Virtual Computing Lab Homepage

There are two sections that are created in the Dashboard to Manage the entire functionality: HPC-BGP and HPC-Template. HPC-BGP tab performs functions as seen in Figure 4.4.In Create Template section we can specify the number of nodes we require in our cloud and how many custom network interfaces we wish to add. We can thus configure a cluster of nodes in a specific topology. Each template is managed by a management node which has VCL running on it and will provision the requested cluster. These details are inserted “anltemplate” table in VCL database. In this table, information such as management nodeid / basenode id, the total number of nodes requested for the cluster, number of private networks, and mapping between nodes and networks are stored.

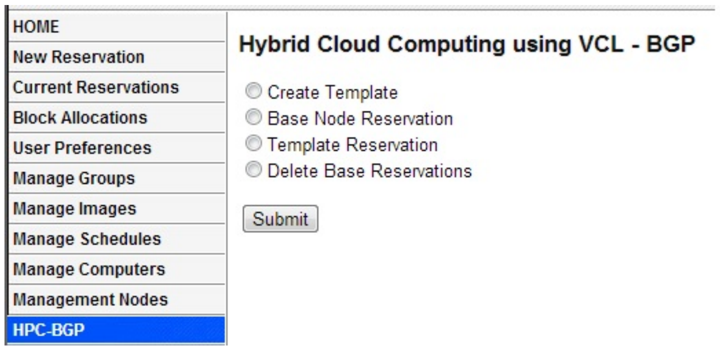
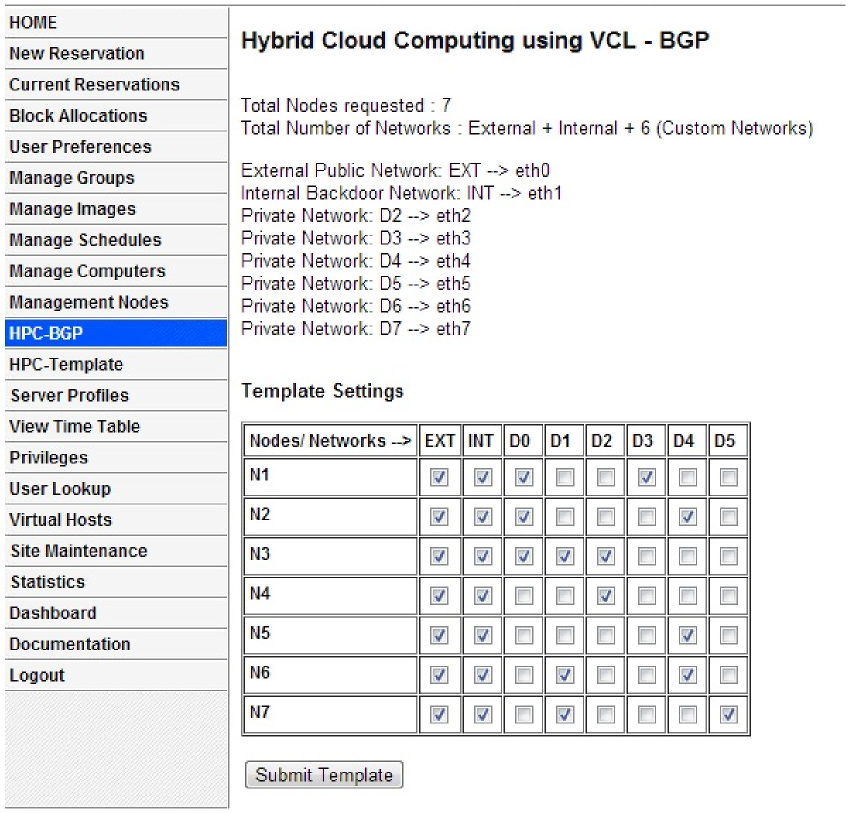


Figure 4.4: HPC-BGP Functions

In our example we have 7 nodes in our cluster and 6 private networks. Thus, we request a template for the specified network and perform the mapping between nodes and network. As we can see in Figure 4.5, there are in total 8 networks in the template. Two of them being assigned for mandatory public and back-door interfaces which will be attached to every node. A matrix mapping is created. A user just needs to check the check-box for every network that needs to be added to a node.

In the network node N1 is connected to network interfaces D0 and D3, node N2 is connected to interfaces D0 and D4, node N3 is connected to network interfaces D0,D1 and D2 and so. This interconnection is mapped onto Template settings. On clicking the submit template button, the configuration is saved in the database and can be used later. The template is again listed for confirmation so that the user can cross check the mappings that are done. As mentioned earlier, one management node is assigned to to each cluster. Thus before reserving the template, we need to reserve the Management node / Base node which the next functionality provided by HPC-BGP section.



Hybrid Cloud Computing using VCL - BGP

Total Nodes requested : 7
Total Number of Networks : External + Internal + 6 (Custom Networks)

External Public Network: EXT --> eth0
Internal Backdoor Network: INT --> eth1
Private Network: D2 --> eth2
Private Network: D3 --> eth3
Private Network: D4 --> eth4
Private Network: D5 --> eth5
Private Network: D6 --> eth6
Private Network: D7 --> eth7

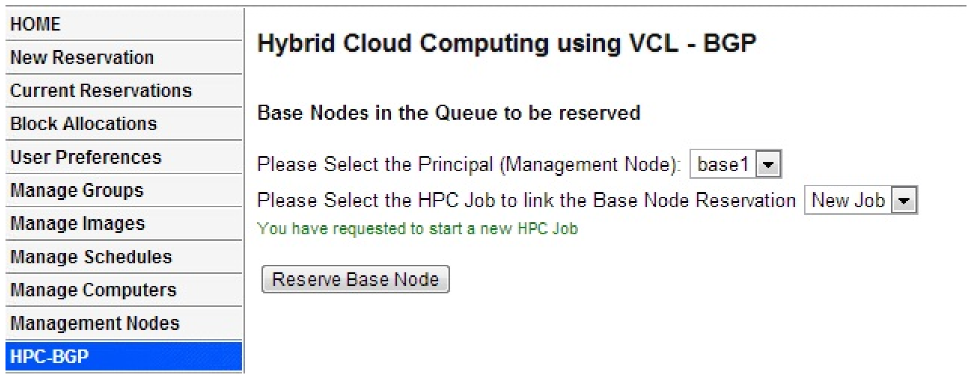
Template Settings

Nodes/ Networks -->	EXT	INT	D0	D1	D2	D3	D4	D5
N1	✓	✓	✓	☐	☐	✓	☐	☐
N2	✓	✓	✓	☐	☐	☐	✓	☐
N3	✓	✓	✓	✓	✓	☐	☐	☐
N4	✓	✓	☐	☐	✓	☐	☐	☐
N5	✓	✓	☐	☐	☐	☐	✓	☐
N6	✓	✓	☐	✓	☐	☐	✓	☐
N7	✓	✓	☐	✓	☐	☐	☐	✓

Figure 4.5: Cluster Node and Network Mapping

4.2.2 Loading BlueGene Environment

This is the next step to be performed after creating a template. We need to start a BlueGene Job and request for a pool of nodes. For default privileges we will be able to reserve a pool of 64 nodes for 60 minutes. Out of this pool one node will be configured as the management node for the template.Figure 4.6 lists all the management nodes which are currently present in the queue to be reserved. These are the management nodes that are assigned during template creation. Using the drop down menu we can select the management node. This node needs to be linked with an BlueGene Job. We can go ahead with a new Job.



Hybrid Cloud Computing using VCL - BGP

Base Nodes in the Queue to be reserved

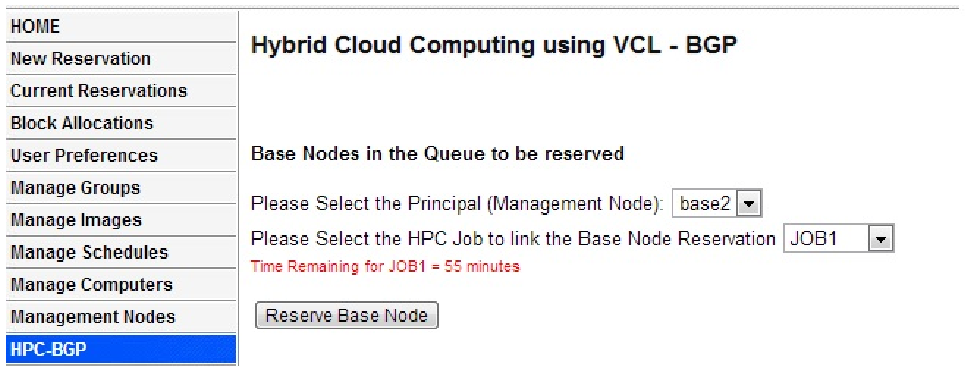
Please Select the Principal (Management Node): base1

Please Select the HPC Job to link the Base Node Reservation New Job

You have requested to start a new HPC Job

Figure 4.6: Base Node Reservation linked to new BlueGene Job

It is also possible that out of the 64 nodes only 7 or 8 will be used for one template. Thus, around 90% of the nodes in the pool are left unused. If the same user decides to create one more cluster of 10 nodes, he should be able to simply add the template under the already running BlueGene Job. This new cluster will definitely be managed by a different management node. As seen in Figure 4.7 user has the option of selecting the BluGene Job to which the template is linked. By selecting an already running Job it shows time left for the Job to end. If the time is sufficient he can go ahead with the currently running Job.



Hybrid Cloud Computing using VCL - BGP

Base Nodes in the Queue to be reserved

Please Select the Principal (Management Node): base2

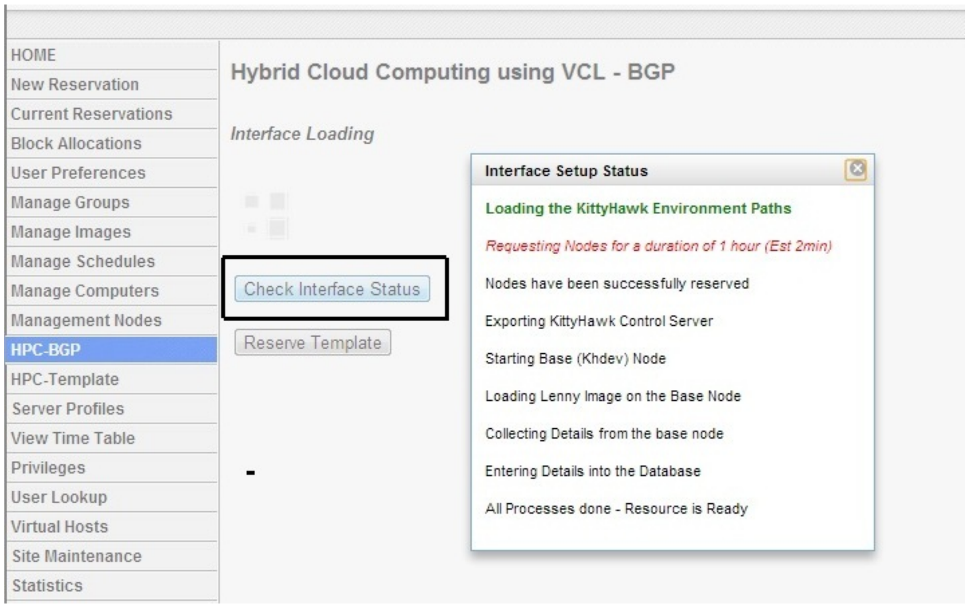
Please Select the HPC Job to link the Base Node Reservation JOB1

Time Remaining for JOB1 = 55 minutes

Figure 4.7: Linking Base Node Reservation with existing BlueGene Job

Base Node Reservation takes around 2 minutes to complete. The user can check the status by click the "Check Interface Status". It opens up a dialog box which lists the steps that are being executed in the background. As seen in Figure 4.8 there are 8 steps before the resource is ready. Steps that are marked in green are complete, the ones in red is currently going on and the ones in black are still left. Once all the steps are done, the web interface will pop up with an alert message saying "Interface Setup Complete". Lets go through the steps. First all the necessary KittyHawk environment variables are loaded. Once this is done a pool of 64 nodes are requested of a period of 60 minutes.

KittyHawk control server is initiated to control request for nodes. One of the nodes is booted with Lenny Image which has VCL Management Node modules installed. Once the node is booted details like public, private IP addresses are entered into the database and the base node is removed for the reserve queue. At this point of time the management node is up and running and ready to process template reservation.



Hybrid Cloud Computing using VCL - BGP

Interface Loading

Interface Setup Status

Loading the KittyHawk Environment Paths

Requesting Nodes for a duration of 1 hour (Est 2min)

Nodes have been successfully reserved

Exporting KittyHawk Control Server

Starting Base (Khdev) Node

Loading Lenny Image on the Base Node

Collecting Details from the base node

Entering Details into the Database

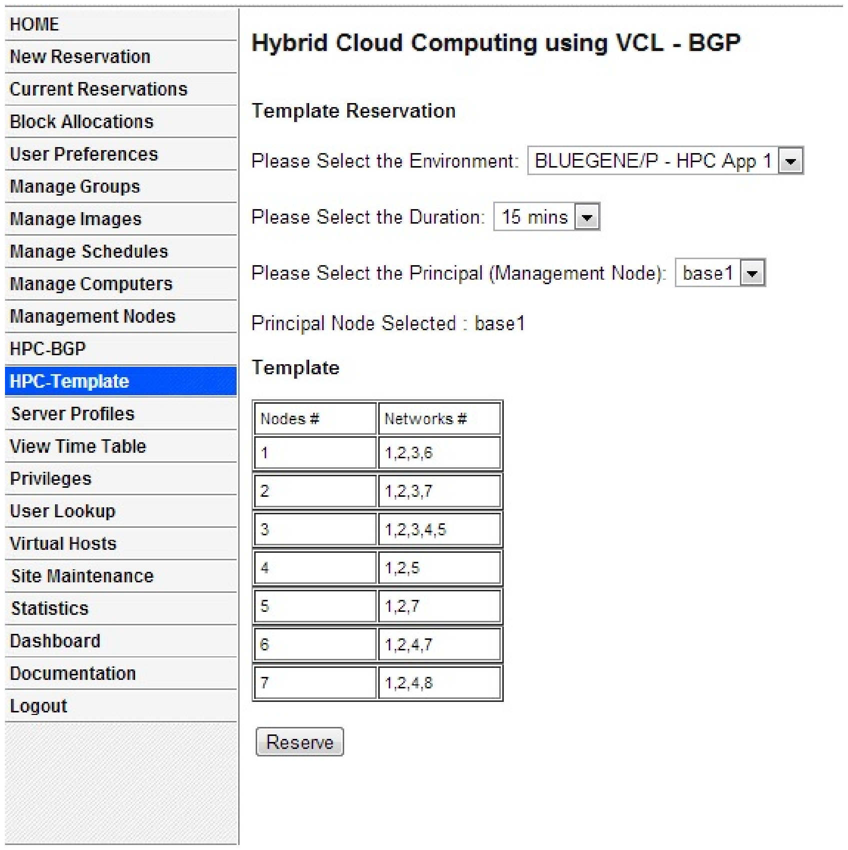
All Processes done - Resource is Ready

Figure 4.8: BluGene Environment Status

4.2.3 Reserving Template / Micro-cloud

Once the above steps are performed the user can go ahead and reserve the template that was created and construct a personalized Micro-cloud. This can be done by either going to HPC-BGP and selecting Template Reservation or directly going to HPC-Template. As seen in Figure 4.9 the user can select the application / Image to be loaded on the nodes in the Micro-cloud. As mentioned in the previous sections each Micro-cloud is controlled by a management node which can be selected using the drop down menu. Once the management node is selected the template linked with it is fetched from the VCL database. Thus the user can once again cross the Node interconnect configurations. Along with this he can also specify a reservation time for the cluster of nodes.

When the reserve button is clicked, the control is transferred to management node. The reservation is processed as a cluster reservation, where multiple reservations are linked to a single request. Provisioning modules interact with KittyHawk control server,



Hybrid Cloud Computing using VCL - BGP

Template Reservation

Please Select the Environment: BLUEGENE/P - HPC App 1

Please Select the Duration: 15 mins

Please Select the Principal (Management Node): base1

Principal Node Selected : base1

Template

Nodes #	Networks #
1	1,2,3,6
2	1,2,3,7
3	1,2,3,4,5
4	1,2,5
5	1,2,7
6	1,2,4,7
7	1,2,4,8

Figure 4.9: Template Reservation

configure the custom network interfaces on the nodes and load them with a stripped down version of debian image. After the node boots, its information is inserted into the database computer tables. After all the reservations linked to the request id is complete, VCL user interface returns back with the connect button which provides IP address and login information for each of the nodes. Using this information we can login into each of the nodes and check the interfaces configured for each of them. Thus, summarizing the above sections we can see that there are three phases for constructing a personal Micro-cloud. First creating a template which specifies the number of nodes and custom networks in the cluster and defines the interconnectivity between them. Before reserving the configured template, a vcl management node needs to reserved (second step) which then handles the template reservation requests which is the third step. Each of the nodes are represented as specific computers in VCL database with their corresponding information. Once the reservation is complete the requested application will be executed on the nodes. In the next chapter we will see specific experiment cases and the results obtained.

CHAPTER 5

Experiments and Results

In this chapter we we show examples of topologies that were tested on the VCL Mirco-cloud. While advanced parallel compilers [91] parallelize codes and try to distribute independent paths to different computational nodes in order to speed up processing, that is done at code level. When one wants to try to do the same thing with larger data oriented workflows the problem is more complex. One would not only want to parallelize pieces of code and computations, but would also want to operate on data chunks that are independent. For instance one of the reasons why Hadoop [5] like solutions are so successful in the cloud is that the large chunks of data are independent of each other with respect to the operations (programs, codes) we want to run on them. Therefore an ideal topology is a star like architecture where the computational elements receive the data that need to be analysed, they do not need to communicate with each other while processing and are relatively immune to communication latencies between them but may need good communication links to the Hub or where the results will be collected. Basically what it means is that the problem, of the so called embarrassingly parallel category, maps very well onto a Hub and Spoke architecture of loosely coupled computational nodes. For each of the example we first model the data flow graph of the problem and then look at the number of arithmetic operations performed. This gives an idea on how many nodes are required in a cluster to compute the problem. We then design a topology diagram to show how the network connections between the nodes would be configured. The management node controls and monitors the flow.

5.1 Example 1

Consider the following example pseudo-code. It illustrates possible micro parallelization but more importantly if each of the variables is considered as a data chunk, or a property of a larger workflow element, then it can indicate how such a problem graph can be processed in a topology that maps onto the problem. An example of a workflow development and management system that could be employed at that level would be Kepler [45] [46] [98].

```
    EXECUTE()
{
A=5;
B=0;
i=1;
if(i <A) do
i=i+1
endif
B=A+7
print A,B,i
}
```

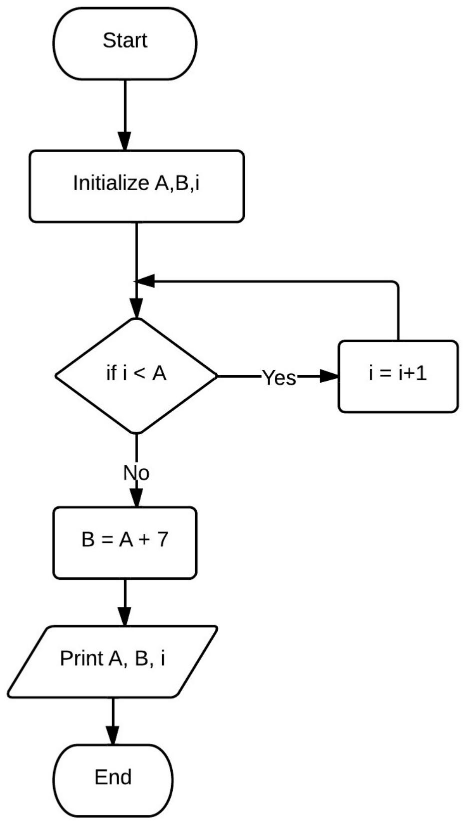


Figure 5.1: Example 1: Control Flow Diagram

Figure 5.1 shows the control flow model for the example we are considering. We can see that all the operations are executed in a sequential manner. We can identify the data flows for A,B and i and construct a data flow model for the same which is shown in Figure 5.2. As we can see the flow of data for each of the variables are different we can assign

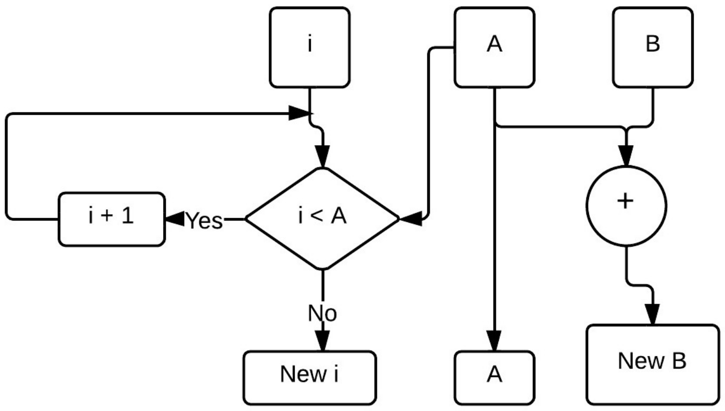


Figure 5.2: Example 1: Data Flow Diagram

each of the arithmetic operations to nodes which will perform the required operation and return the result. Mapping this problem into a network topology we have three data variables A,B and i and in the program we have three operations, first summation of the values of A and B which, second a comparison between values of i and A and third, incrementing the value based on the result of comparison operation. We can allocate each of these operations to three different nodes and provide them with the required values. After the program is completed we will print the final values. Figure 5.3 shows the topology diagram. We will design a cluster with three nodes and a VCL management node to control the node which will serve as the master node. We assign different networks for each of the operation and thus we require four custom network for implementing this example. As described in Section 4.2 to execute the above mentioned application we need to configure a template, request a reservation for it and then execute the program. Figure A.1 shows the template configuration where we can create a mapping between nodes and networks. Once the template is submitted and the template is reserved VCL returns back

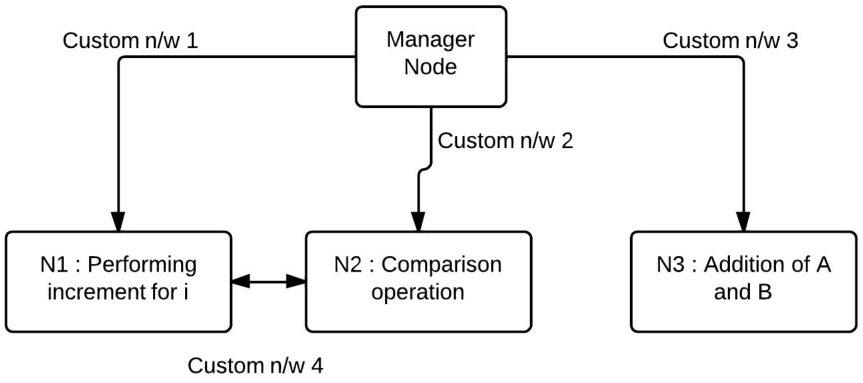
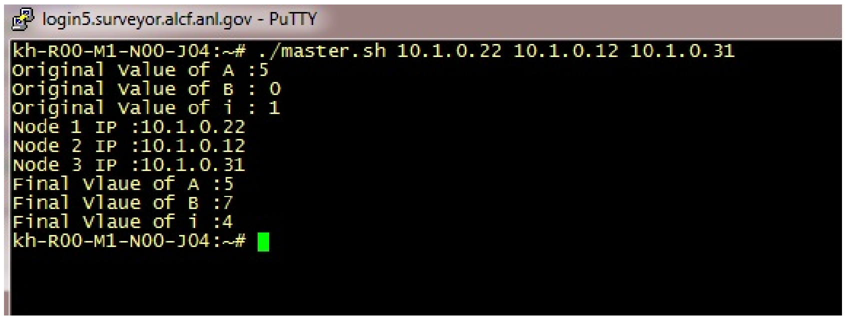


Figure 5.3: Example 1: Node Topology Diagram

a “Connect” interface which provides the connection details for every individual node as shown in Figure A.2. The management node being the master node will have the network interfaces and can communicate with all the three nodes in the cluster. Each node is assigned to perform one operation and return back the result to the master node where the final results will be printed. Figures A.3, A.4, A.5 shows details about the network interfaces of nodes. Based on the template configuration the number of interfaces will vary from node to node.

We execute the program on the master node and the result is collected back at the master node and printed. The result is shown in Figure 5.4.



```
kh-R00-M1-N00-J04:~# ./master.sh 10.1.0.22 10.1.0.12 10.1.0.31
Original Value of A :5
Original Value of B : 0
Original Value of i : 1
Node 1 IP :10.1.0.22
Node 2 IP :10.1.0.12
Node 3 IP :10.1.0.31
Final Vlaue of A :5
Final Vlaue of B :7
Final Vlaue of i :4
kh-R00-M1-N00-J04:~#
```

Figure 5.4: Example 1: Result

5.2 Example 2

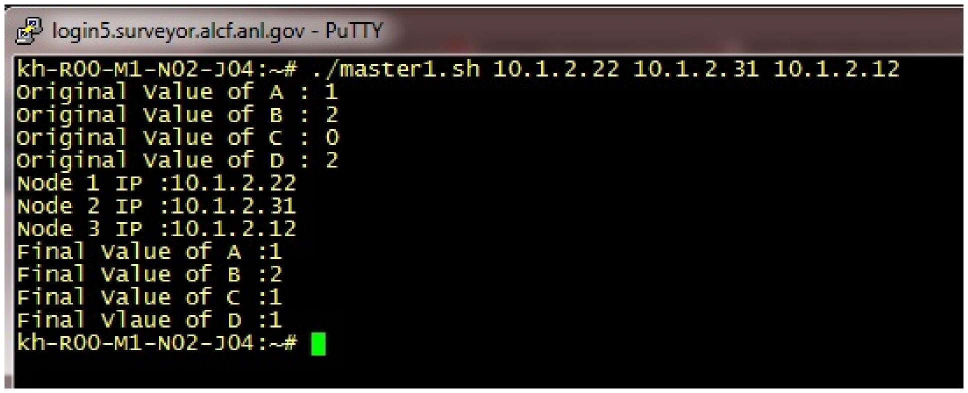
Let us consider a second example. The pseudo-code is listed below. The control flow model for the same is shown in Figure 5.6. In this example we consider four variables A,B,C and D. There are two comparison operations performed in the code. The first one is between A and B, and based on the result the value of C is updated correspondingly. After this is done, value of D is updated based on the comparison of the updated value of C. Once these operations are done we print the new values of all the variables. As seen in Example 1, these operations are performed in a sequential manner. We can identify the data flow model for the problem as shown in Figure 5.6.

```
EXECUTE()
{
A=1;
B=2;
C=0;
D=2;
if(A <B)
C=B-A
else
C=A-B
    if(C <5)
D = D-1
else D =D+1
print A,B,C,D
}
```

As we can see the flow of data for each of the variables are different we can assign each of the arithmetic operations to nodes which will perform the required operation and return the result. Mapping this problem into a network topology we have three data variables A, B, C and D. In the program we have three operations that are performed, comparison, increment and decrement. We can allocate each of these operations to three different nodes and provide them with the required values. After the program is completed we will print the final values of the variables. Figure 5.8 shows the topology diagram. We will design a cluster with three nodes and a VCL management node to control the node which will serve as the master node. We assign different networks for each of the operation and thus we require three custom network for implementing this example. One network is used for all the comparison operations between the variables passed, and the other networks are used for increment and decrement operations respectively. Based on the topology diagram we will first create a template as shown in Figure A.6. Once the template is submitted and the template is reserved VCL returns back a “Connect” interface which provides the connection details for every individual node as shown in Figure A.7. The management node being the master node will have the network interfaces and can communicate with all the three nodes in the cluster. Each node is assigned to perform one operation and return back the result to the master node where the final results will be printed. Figures

A.8, A.9, A.10 and A.11 shows details about the network interfaces of nodes. Based on the template configuration the number of interfaces will vary from node to node.

We execute the program on the master node and the result is collected back at the master node and printed. The result is shown in Figure 5.5.



```
kh-R00-M1-N02-J04:~# ./master1.sh 10.1.2.22 10.1.2.31 10.1.2.12
Original Value of A : 1
Original Value of B : 2
Original Value of C : 0
Original Value of D : 2
Node 1 IP :10.1.2.22
Node 2 IP :10.1.2.31
Node 3 IP :10.1.2.12
Final Value of A :1
Final Value of B :2
Final Value of C :1
Final Vlaue of D :1
kh-R00-M1-N02-J04:~#
```

Figure 5.5: Example 2: Result

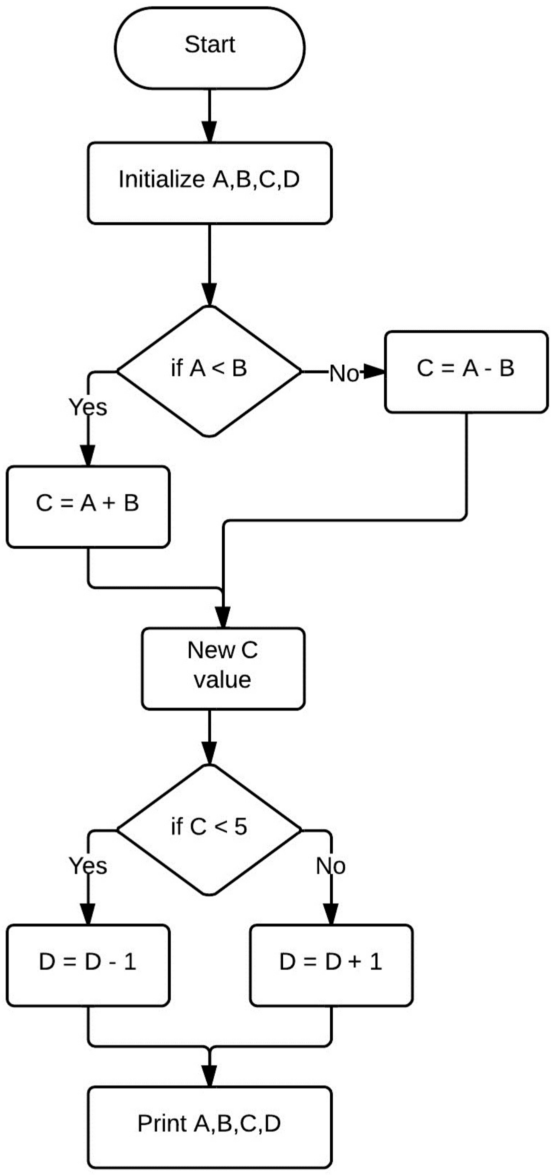


Figure 5.6: Example 2: Control Flow Diagram

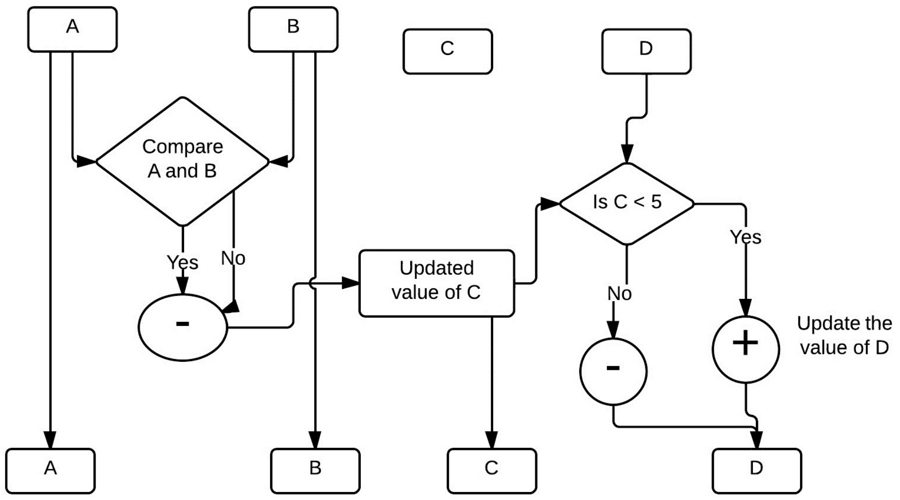


Figure 5.7: Example 2: Data Flow Diagram

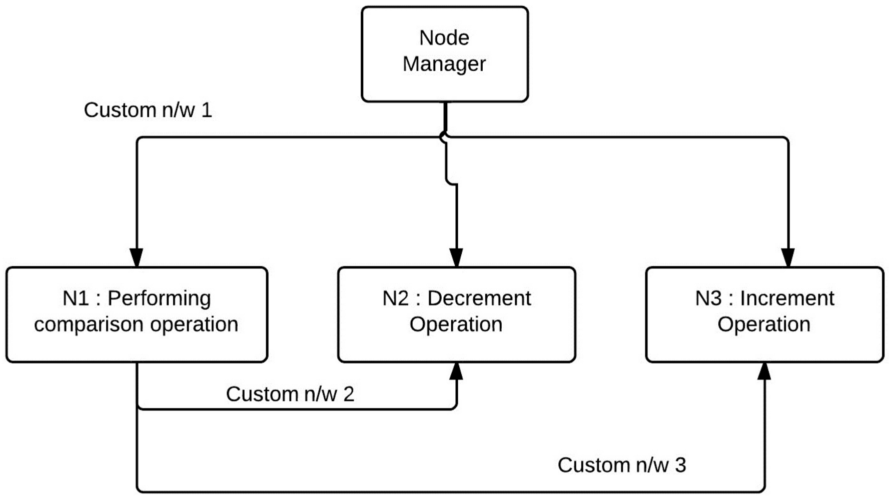


Figure 5.8: Example 2: Node Topology Diagram

CHAPTER 6

Conclusion and Future Work

As we are entering into the era of Big Data, the demand for processing power keeps rising. Supercomputer manufacturing companies like IBM, Cray and public clouds like Amazon EC2 are at the fore front of bringing together large scale analytics and high performance computing and data facilities. Traditional supercomputing environment provide very high performance but are very expensive and thus not available to a broader community. They are mostly used for various scientific workflows in domains like quantum physics, weather forecasting, climate research and physical simulations. An open question is, what kind of topology in compute architecture in general is most appropriate for different applications. In some cases it is appropriate to rely on control flow view of the problem and parallelize it, but in some cases it is more appropriate to consider the data flow view of the particular problem and attempt to exploit that to deliver improved performance.

Traditionally supercomputers are control flow oriented and problems need to map onto the supercomputer architecture. More recently data flow oriented computing has again engaged interest along with FPGA, GPU accelerators that can considerably help with parts of the code that map well onto the architecture they offer. This project is concerned with exploration of software defined internode topology that may be of use in mapping parts of supercomputers onto a problem or problem parts that run on it.

We have used VCL [42] and KittyHawk [37] and developed a software defined topology layer for IBM BlueGene/P supercomputer. VCL acts as a management system which uses KittyHawk as a provisioning engine to control and configure BlueGene/P resources. We added capability into VCL which enables a user to create topology (as a Mirco-cloud) based on the workflow under consideration. KittyHawk collects all the information by interacting with VCL modules and adds custom network interfaces on the nodes and configures the requested topology . Stripped down Linux kernel images are loaded on BlueGene Nodes. Each Micro-cloud is managemed by a VCL management node which uses a backdoor network connection to manage BlueGene nodes. Topology for the Micro-cloud is constructed based on the workflow to be executed. We first create a data flow model for the program and figure out the number of data nodes that need to be used. We consider a couple of simple examples which can be computed using a few number of nodes. As mentioned earlier we create data flow model for the same and construct a topology and reserve a Micro-cloud. Once the nodes are set-up we execute the workflow from the master node which computes the program over multiple nodes and then collects the results. Visualizing such an environment from a high level we can have different operating systems running on different nodes, where each node is dedicated to perform a particular function and have the results sent back to the master node or directed to a storage space. VCL has the capability to provide a well defined management interface through which each node can be configured in terms of the image to be loaded on the nodes, add custom network interfaces etc.

From the experiments we can see that it is very easy to create a data flow model for workflows. It gives a user a broader perspective to realize the flow of data during execution as opposed to a control flow model. A software system like VCL and KittyHawk when put together provides the flexibility of creating the environment required. Through this research we can find that specialized supercomputer like BlueGene/P when integrated with a software management system provides a cloud computing model to execute compute-intensive workflows. This can also be extended by configuring a cluster on BluGene and running other software solutions like Hadoop, Openstack etc and utilizing the resources for HPC applications.

REFERENCES

[1] 3D Torus network topology. http://www.eurotech.com/en/hpc/hpc+solutions/3d+torus. "Last Accessed : March 10, 2013".

[2] Amazon adds HPC capability to EC2. http://www.hpcinthecloud.com/hpccloud/2010-07-13/amazon_adds_hpc_capability_to_ec2.html. "Last Accessed : March 13, 2013".

[3] Amazon EC2. http://aws.amazon.com/ec2/. "Last Accessed : March 13, 2013".

[4] Amoeba - Cluster Method. http://www.acsu.buffalo.edu/~geojared/tools.htm. "Last Accessed : March 10, 2013".

[5] Apache Hadoop. http://hadoop.apache.org/. "Last Accessed : March 12, 2013".

[6] Birth of Hypercube - Cosmic Cube. http://www.netlib.org/utk/lsi/pcwLSI/text/node13.html. "Last Accessed : March 10, 2013".

[7] Bluegene/Q, PowerPC. http://www.top500.org/system/177720. "Last Accessed : March 10, 2013".

[8] CLOS Networks. http://www.stanford.edu/class/ee384y/Handouts/clos_networks.pdf. "Last Accessed : March 10, 2013".

[9] Comprehensive List of Big Data Statistics. http://wikibon.org/blog/big-data-statistics/. "Last Accessed : March 12, 2013".

[10] Control Data Corporation, CDC 6600. http://ed-thelen.org/comp-hist/vs-cdc-6600.html. "Last Accessed : March 12, 2013".

[11] Cray, the supercomputer company. http://www.cray.com/Home.aspx. "Last Accessed : March 12, 2013".

[12] Data procesing techniques in era of big data. http://www.cse.unsw.edu.au/~ssakr/BigDataBook/. "Last Accessed : March 12, 2013".

[13] Dell : High Performance Computing. http://www.dell.com/Learn/us/en/555/hpcc?c=us&l=en&s=biz. "Last Accessed : March 13, 2013".

[14] The Eucalyptus Open-source Cloud-computing System. https://dspace.ist.utl.pt/bitstream/2295/584876/1/CCGrid2009_Eucalyptus.pdf. "Last Accessed : March 13, 2013".

[15] Field-Programmable Gate Array. http://en.wikipedia.org/wiki/Field-programmable_gate_array. "Last Accessed : March 13, 2013".

[16] Fujitsu: Supercomputer. http://www.fujitsu.com/global/services/solutions/tc/hpc/products/primehpc/. "Last Accessed : March 12, 2013".

[17] Hadoop HDFS. hadoop.apache.org/docs/r0.18.0/hdfs_design.pdf. "Last Accessed : March 13, 2013".

[18] High performance computing cloud offerings from ibm. http://www-03.ibm.com/systems/technicalcomputing/solutions/cloud/cloud_offerings.html. "Last Accessed : March 13, 2013".

[19] High Performance Computing on AWS. http://aws.amazon.com/hpc-applications/. "Last Accessed : March 13, 2013".

[20] HPC : High Performance Computing. http://h20311.www2.hp.com/hpc/us/en/hpc-index.html. "Last Accessed : March 13, 2013".

[21] HPC] in the Cloud, note ="Last Accessed : March 13, 2013", howpublished = http://www.hpcinthecloud.com/.

[22] IBM: Bluegene supercomputing solution. http://www-03.ibm.com/systems/technicalcomputing/solutions/bluegene/index.html. "Last Accessed : March 12, 2013".

[23] Inside HPC]: What is High Performance Computing, note ="Last Accessed : March 13, 2013", howpublished = http://insidehpc.com/hpc-basic-training/what-is-hpc/.

[24] Intel : High Performance Computing. http://www.intel.com/content/www/us/en/high-performance-computing/server-reliability.html. "Last Accessed : March 13, 2013".

[25] Interconnection Networks. http://www.ece.eng.wayne.edu/~czxu/ece7660_f05/network.pdf. "Last Accessed : March 10, 2013".

[26] The Internet of Things. http://www.mckinsey.com/insights/mgi/research/technology_and_innovation/big_data_the_next_frontier_for_innovation. "Last Accessed : March 12, 2013".

[27] Jaguar - Cray XT3, 2.4 ghz. http://www.top500.org/system/174479. "Last Accessed : March 10, 2013".

[28] The kepler project. https://kepler-project.org/. "Last Accessed : March 13, 2013".

[29] LDAP linux HOW TO. http://tldp.org/HOWTO/LDAP-HOWTO/whatisldap.html. "Last Accessed : March 6, 2013".

[30] Mapreduce. http://wiki.apache.org/hadoop/MapReduce. "Last Accessed : March 12, 2013".

[31] Mosix - Cluster Operating System. http://www.mosix.cs.huji.ac.il/. "Last Accessed : March 10, 2013".

[32] NCSUs Virtual Computing Lab: A Cloud Computing Solution, author=Henry E. Schaffer, Samuel F. Averitt, Marc I. Hoit, Aaron Peeler, Eric D. Sills, and Mladen A. Vouk,, journal=Published by the IEEE Computer Society, year=2009.

[33] Nimbus. http://www.nimbusproject.org/nimbus_cloud. "Last Accessed : March 13, 2013".

[34] North carolina state university - hpc service. http://www.ncsu.edu/itd/hpc/main.php. "Last Accessed : March 13, 2013".

[35] Open stack : Cloud software. http://www.openstack.org/. "Last Accessed : March 13, 2013".

[36] Plan9 - Clustering Method. http://thor.cs.ucsb.edu/~ravenben/papers/prelims/plan9_new.pdf. "Last Accessed : March 10, 2013".

[37] Project Kittyhawk: A Global Scale Computer. http://kittyhawk.bu.edu/kittyhawk/Kittyhawk.html. "Last Accessed : March 9, 2013".

[38] Red Storm Cray. http://www.top500.org/system/174240. "Last Accessed : March 10, 2013".

[39] Smart ibm solutions for high performance engineering clouds. ftp://ftp.software.ibm.com/common/ssi/ecm/en/dcw03003usen/DCW03003USEN.PDF. "Last Accessed : March 13, 2013".

[40] Techopedia - high performance computing. http://www.techopedia.com/definition/4595/high-performance-computing-hpc. "Last Accessed : March 13, 2013".

[41] Thunderbird - Poweredge 1850. http://www.top500.org/system/174736. "Last Accessed : March 10, 2013".

[42] Virtual Computing Lab. `http://vcl.ncsu.edu/`. "Last Accessed : March 14, 2013".

[43] Virtual local area networks. `http://www.cse.wustl.edu/~jain/cis788-97/ftp/virtual_lans/index.htm`. "Last Accessed : March 11, 2013".

[44] Vikram S. Adve and Mary K. Vernon. Performance analysis of mesh interconnection networks with deterministic routing. *Parallel and Distributed Systems, IEEE Transactions on*, 5(3):225–246, 1994.

[45] I Altintas, R Barreto, JM Blondin, Z Cheng, T Critchlow, A Khan, Scott A Klasky, J Ligon, B Ludaescher, PA Mouallem, et al. Automation of network-based scientific workflows. Technical report, Oak Ridge National Laboratory (ORNL), 2007.

[46] Ilkay Altintas, Oscar Barney, Zhengang Cheng, Terence Critchlow, Bertram Ludaescher, Steve Parker, Arie Shoshani, and Mladen Vouk. Accelerating the scientific exploration process with scientific workflows. In *Journal of Physics: Conference Series*, volume 46, page 468. IOP Publishing, 2006.

[47] J Appavoo, V Uhlig, A Waterland, B Rosenburg, D Da Silva, and JE Moreira. Kittyhawk: enabling cooperation and competition in a global, shared computational system. *IBM Journal of Research and Development*, 53(4):9–1, 2009.

[48] Jonathan Appavoo, Volkmar Uhlig, and Amos Waterland. Project kittyhawk: building a global-scale computer: BlueGene/P as a generic computing platform. *ACM SIGOPS Operating Systems Review*, 42(1):77–84, 2008.

[49] Jonathan Appavoo, Amos Waterland, Dilma Da Silva, Volkmar Uhlig, Bryan Rosenburg, Eric Van Hensbergen, Jan Stoess, Robert Wisniewski, and Udo Steinberg. Providing a cloud network infrastructure on a supercomputer. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, pages 385–394. ACM, 2010.

[50] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[51] Infiniband Trade Association. Infiniband Specifications. `http://www.infinibandta.org/content/pages.php?pg=technology_faq`. "Last Accessed : March 10, 2013".

[52] Rob Baxter, Stephen Booth, Mark Bull, Geoff Cawood, James Perry, Mark Parsons, Alan Simpson, Arthur Trew, Andrew McCormick, Graham Smart, et al. The FPGA high-performance computing alliance parallel toolkit. In *Adaptive Hardware and*

Systems, 2007. AHS 2007. Second NASA/ESA Conference on, pages 301–310. IEEE, 2007.

[53] Rob Baxter, Stephen Booth, Mark Bull, Geoff Cawood, James Perry, Mark Parsons, Alan Simpson, Arthur Trew, Andrew McCormick, Graham Smart, et al. Maxwell-a 64 fpga supercomputer. In *Adaptive Hardware and Systems, 2007. AHS 2007. Second NASA/ESA Conference on*, pages 287–294. IEEE, 2007.

[54] Gordon Bell and Jim Gray. What's next in high-performance computing? *Communications of the ACM*, 45(2):91–95, 2002.

[55] F. Belletti, M. Cotallo, A. Cruz, L.A. Fernandez, A. Gordillo-Guerrero, M. Guidetti, A. Maiorano, F. Mantovani, E. Marinari, V. Martin-Mayor, A. Muoz-Sudupe, D. Navarro, G. Parisi, S. Perez-Gaviro, M. Rossi, J.J. Ruiz-Lorenzo, S.F. Schifano, D. Sciretti, A. Tarancon, R. Tripiccione, J.L. Velasco, D. Yllanes, and G. Zanier. Janus: An fpga-based system for high-performance scientific computing. *Computing in Science Engineering*, 11(1):48–58, Jan.-Feb.

[56] Duncan Buell, Tarek El-Ghazawi, Kris Gaj, and Volodymyr Kindratenko. High-performance reconfigurable computing. *COMPUTER-IEEE COMPUTER SOCIETY-*, 40(3):23, 2007.

[57] Rajkumar Buyya et al. High performance cluster computing: Architectures and systems (volume 1). *Prentice Hall, Upper SaddleRiver, NJ, USA*, 1:999, 1999.

[58] A.G. Carlyle, S.L. Harrell, and P.M. Smith. Cost-effective hpc: The community or the cloud? In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 169–176, 30 2010-Dec. 3.

[59] North Carolina State University. Computer Science Department - news. `http://www.csc.ncsu.edu/news/1348`. "Last Accessed : March 4, 2013".

[60] Alex Cerier, Matt Visich, Chris Bork, Jon Brooks, Brian W. Perrault, Philip Haddad, and Xinli Wang. Infrastructure-as-a-service clouds in a professional environment. In *Proceedings of the 2011 conference on Information technology education*, SIGITE '11, pages 301–302, New York, NY, USA, 2011. ACM.

[61] Tufts Linux Research Cluster. Introduction to clusters and parallel computing. `http://uit.tufts.edu/at/downloads/cman8-05.pdf`. "Last Accessed : March 13, 2013".

[62] William J. Dally. From Hypercubes to Dragonflies - history of interconnect. `http://www.csm.ornl.gov/workshops/IAA-IC-Workshop-08/documents/wiki/dally_iaa_workshop_0708.pdf`. "Last Accessed : March 10, 2013".

[63] Jack B. Dennis. Data flow supercomputers. `http://www.cs.ucf.edu/courses/cop4020/sum2010/Lecture10.pdf`. "Last Accessed : March 13, 2013".

[64] Dr. Patrick Dreher. Big data and cloud computing. Cloud Computing Technology - North Carolina State University. "Last Accessed : March 13, 2013".

[65] Tarek El-Ghazawi, Esam El-Araby, Miaoqing Huang, Kris Gaj, Volodymyr Kindratenko, and Duncan Buell. The promise of high-performance reconfigurable computing. *Computer*, 41(2):69–76, 2008.

[66] Constantinos Evangelinos and C Hill. Cloud computing for parallel scientific hpc applications: Feasibility of running coupled atmosphere-ocean climate models on amazons ec2. *ratio*, 2(2.40):2–34, 2008.

[67] R. Fernandez and A. Kanevsky. Generalized ring interconnection networks. In *Parallel Processing Symposium, 1994. Proceedings., Eighth International*, pages 30–34, Apr.

[68] Brad Fitzpatrick. Distributed Caching with Memcached - Linux Journal. `http://www.linuxjournal.com/article/7451`. "Last Accessed : March 9, 2013".

[69] James Greco, Grzegorz Cieslewski, Adam Jacobs, Ian A Troxel, and Alan D George. Hardware/software interface for high-performance space computing with fpga coprocessors. In *Aerospace Conference, 2006 IEEE*, pages 10–pp. IEEE, 2006.

[70] Zhiyang Guo and Yuanyuan Yang. On nonblocking multirate multicast fat-tree data center networks with server redundancy. In *Parallel Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International*, pages 1034–1044, May.

[71] Qiming He, Shujia Zhou, Ben Kobler, Dan Duffy, and Tom McGlynn. Case study for running hpc applications in public clouds. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, pages 395–401, 2010.

[72] M.C. Herbordt, T. VanCourt, Yongfeng Gu, B. Sukhwani, A. Conti, J. Model, and D. DiSabello. Achieving high performance with fpga-based computing. *Computer*, 40(3):50–57, March.

[73] IBM. Extreme Cloud Administration Toolkit. `http://www-03.ibm.com/systems/software/xcat/`. "Last Accessed : March 5, 2013".

[74] IBM. The Internet of Things. `http://www.ibm.com/smarterplanet/us/en/overview/article/iot_video.html`. "Last Accessed : March 12, 2013".

[75] K.R. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, Harvey J. Wasserman, and N.J. Wright. Performance analysis of high performance computing applications on the amazon web services cloud. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 159–168, 30 2010-Dec. 3.

[76] J. Kim, J. Balfour, and W.J. Dally. Flattened butterfly topology for on-chip networks. *Computer Architecture Letters*, 6(2):37–40, Feb.

[77] J. Kim, W.J. Dally, S. Scott, and D. Abts. Technology-driven, highly-scalable dragonfly topology. In *Computer Architecture, 2008. ISCA '08. 35th International Symposium on*, pages 77–88, June.

[78] John Kim. High-Radix Interconnection Networks. `http://cva.stanford.edu/publications/2008/jkim_thesis.pdf`. "Last Accessed : March 10, 2013".

[79] John Kim and William J. Dally. Flattened butterfly: A cost-efficient topology for high-radix networks. In *in Proc. of the Intl. Symp. on Computer Architecture*, 2007.

[80] John Kim, William J. Dally, Brian Towles, and Amit K. Gupta. Microarchitecture of a high-radix router. In *IN ISCA 05: PROCEEDINGS OF THE 32ND ANNUAL INTERNATIONAL SYMPOSIUM ON COMPUTER ARCHITECTURE*. IEEE Computer Society, 2005.

[81] H. T. Kung. Systolic array. In *Encyclopedia of Computer Science*, pages 1741–1743. John Wiley and Sons Ltd., Chichester, UK.

[82] Argonne National Laboratory. Running Jobs at ANL. `https://www.alcf.anl.gov/resource-guides/running-jobs#job-submission`. "Last Accessed : March 2, 2013".

[83] Olav Lindtjorn, Robert Clapp, Oliver Pell, Haohuan Fu, Michael Flynn, and Haohuan Fu. Beyond traditional microprocessors for geoscience high-performance computing applications. *Micro, IEEE*, 31(2):41–49, 2011.

[84] Jorg Lotze. Dataflow an effective model for high performance computing. `http://blog.xcelerit.com/dataflow-an-effective-model-for-high-performance-computing-2/`. "Last Accessed : March 14, 2013".

[85] Nick McKeown. Software-defined networking. *INFOCOM keynote talk, Apr*, 2009.

[86] Peter Mell and Timothy Grance. The nist definition of cloud computing (draft). *NIST special publication*, 800:145, 2011.

[87] Ivanka Menken and Gerard Blokdijk. *Cloud Computing Best Practice Specialist Guide for SaaS and Web Applications: Software as a Service*. Emereo Pty Ltd, London, UK, UK, 2009.

[88] Ivanka Menken and Gerard Blokdijk. *Cloud Computing Best Practice Specialist Guide for Storage Management and Platform as a service (PaaS): Understanding and Applying PaaS Solutions*. Emereo Pty Ltd, London, UK, UK, 2009.

[89] Bradley Mitchell. Infiniband. `http://compnetworking.about.com/cs/clustering/g/bldef_infiniban.htm`. "Last Accessed : March 10, 2013".

[90] Virali Morozov. Blue gene/p Architecture. `http://press.mcs.anl.gov/gs11/files/2011/01/Vitali_WSS11.pdf`. "Last Accessed : March 2, 2013".

[91] David A Padua and Michael J Wolfe. Advanced compiler optimizations for supercomputers. *Communications of the ACM*, 29(12):1184–1201, 1986.

[92] Gregory F . Pfister. An introduction to Infiniband Architecture. `http://gridbus.csse.unimelb.edu.au/~raj/superstorage/chap42.pdf`. "Last Accessed : March 10, 2013".

[93] Justin L Rice, Khalid H Abed, and Gerald R Morris. Design heuristics for mapping floating-point scientific computational kernels onto high performance reconfigurable computers. *Journal of Computers*, 4(6):542–553, 2009.

[94] Aaron Peeler Henry Shaffer Eric Sills Sarah Stein Josh Thompson Mladen Vouk Sam Averitt, Michael Bugaev. Virtual Computing Laboratory (VCL). *In the proceedings of the International Conference on Virtual Computing*, 2007.

[95] Yuichi Tsuchimoto Vivian Lee Satoshi Tsuchiya, Yoshinori Sakamoto. Big data processing in a cloud environment. `http://www.fujitsu.com/downloads/MAG/vol48-2/paper09.pdf`. "Last Accessed : March 12, 2013".

[96] Gilad Shainer. Network topologies: How to design. `http://www.hpcadvisorycouncil.com/events/2011/switzerland_workshop/pdf/Presentations/Day%201/10_HPCAC.pdf`. "Last Accessed : March 10, 2013".

[97] John A Sharp. *Data flow computing*. Ablex Pub, 1992.

[98] Arie Shoshani, Ilkay Altintas, Alok Choudhary, Terence Critchlow, Chandrika Kamath, Bertram Ludäscher, Jarek Nieplocha, Steve Parker, Rob Ross, Nagiza Samatova, et al. Sdm center technologies for accelerating scientific discoveries. In *Journal of Physics: Conference Series*, volume 78, page 012068. IOP Publishing, 2007.

[99] Apache Software Foundation. Project VCL. http://projects.apache.org/projects/vcl.html. ”Last Accessed : March 1, 2013”.

[100] Nasri Sulaiman, Zeyad Assi Obaid, MH Marhaban, and MN Hamidon. Design and implementation of fpga-based systems-a review. *Australian Journal of Basic and Applied Sciences*, 3(4):3575–3596, 2009.

[101] Song Sun and J. Zambreno. A floating-point accumulator for fpga-based high performance computing applications. In *Field-Programmable Technology, 2009. FPT 2009. International Conference on*, pages 493–499, Dec.

[102] Maxeler Technologies. Multiscale dataflow programming. http://www.doc.ic.ac.uk/~kc1810/docs/maxcompiler-tutorial.pdf. ”Last Accessed : March 13, 2013”.

[103] Maxeler Technologies. Multiscale performance computing solutions. ”Last Accessed : March 13, 2013”.

[104] Josh Thompson. LDAP Authentication for partner institutions. https://vcl.ncsu.edu/help/for-partner-institutions/ldap-authentication. ”Last Accessed : March 5, 2013”.

[105] Josh Thompson. VCL Web Code Overview. https://cwiki.apache.org/confluence/display/VCL/Web+Code+Overview. ”Last Accessed : March 8, 2013”.

[106] Apache VCL. VCL Resources Attributes. https://cwiki.apache.org/VCL/for-vcl-users.html#ForVCLUsers-ResourceAttributes. ”Last Accessed : March 8, 2013”.

[107] Yan Zhai, Mingliang Liu, Jidong Zhai, Xiaosong Ma, and Wenguang Chen. Cloud versus in-house cluster: Evaluating amazon cluster compute instances for running mpi applications. In *High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for*, pages 1–10, Nov.

[108] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1):7–18, 2010.

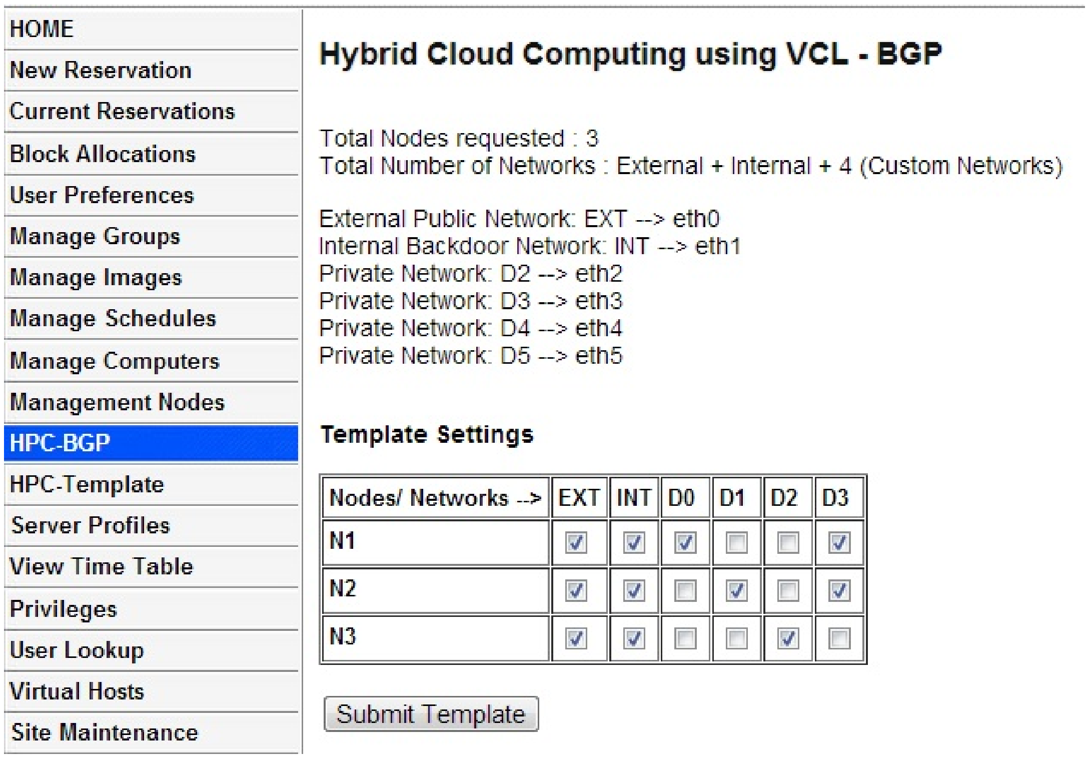
[109] Xingjun Zhang, Yanfei Ding, Yiyuan Huang, and Xiaoshe Dong. Design and implementation of a heterogeneous high-performance computing framework using dynamic and partial reconfigurable fpgas. In *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*, pages 2329–2334, 29 2010-July 1.

APPENDICES

APPENDIX A

Experiments and Results: Figures

A.1 Example 1



Hybrid Cloud Computing using VCL - BGP

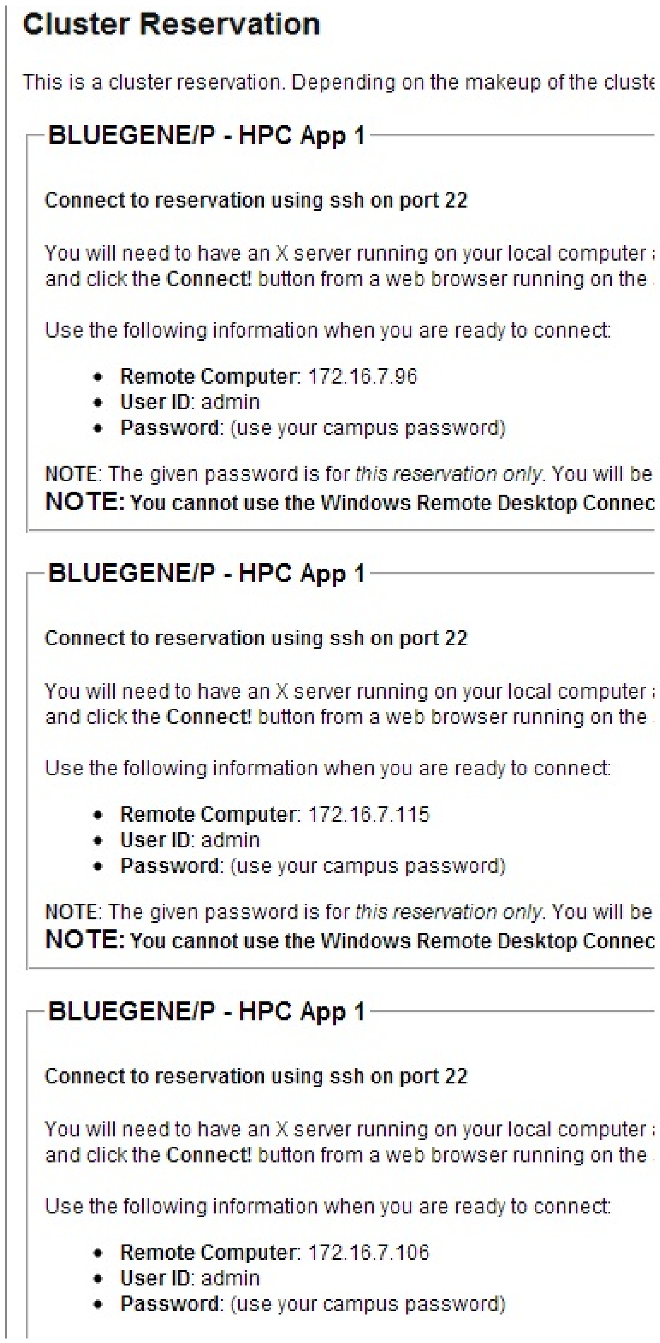
Total Nodes requested : 3
Total Number of Networks : External + Internal + 4 (Custom Networks)

External Public Network: EXT --> eth0
Internal Backdoor Network: INT --> eth1
Private Network: D2 --> eth2
Private Network: D3 --> eth3
Private Network: D4 --> eth4
Private Network: D5 --> eth5

Template Settings

Nodes/ Networks -->	EXT	INT	D0	D1	D2	D3
N1	✓	✓	✓	☐	☐	✓
N2	✓	✓	☐	✓	☐	✓
N3	✓	✓	☐	☐	✓	☐

Figure A.1: Example 1: Cluster Configuration


Cluster Reservation

This is a cluster reservation. Depending on the makeup of the cluste

BLUEGENE/P - HPC App 1

Connect to reservation using ssh on port 22

You will need to have an X server running on your local computer
and click the **Connect!** button from a web browser running on the

Use the following information when you are ready to connect:

- **Remote Computer**: 172.16.7.96
- **User ID**: admin
- **Password**: (use your campus password)

NOTE: The given password is for *this reservation only*. You will be
NOTE: You cannot use the Windows Remote Desktop Connec

BLUEGENE/P - HPC App 1

Connect to reservation using ssh on port 22

You will need to have an X server running on your local computer
and click the **Connect!** button from a web browser running on the

Use the following information when you are ready to connect:

- **Remote Computer**: 172.16.7.115
- **User ID**: admin
- **Password**: (use your campus password)

NOTE: The given password is for *this reservation only*. You will be
NOTE: You cannot use the Windows Remote Desktop Connec

BLUEGENE/P - HPC App 1

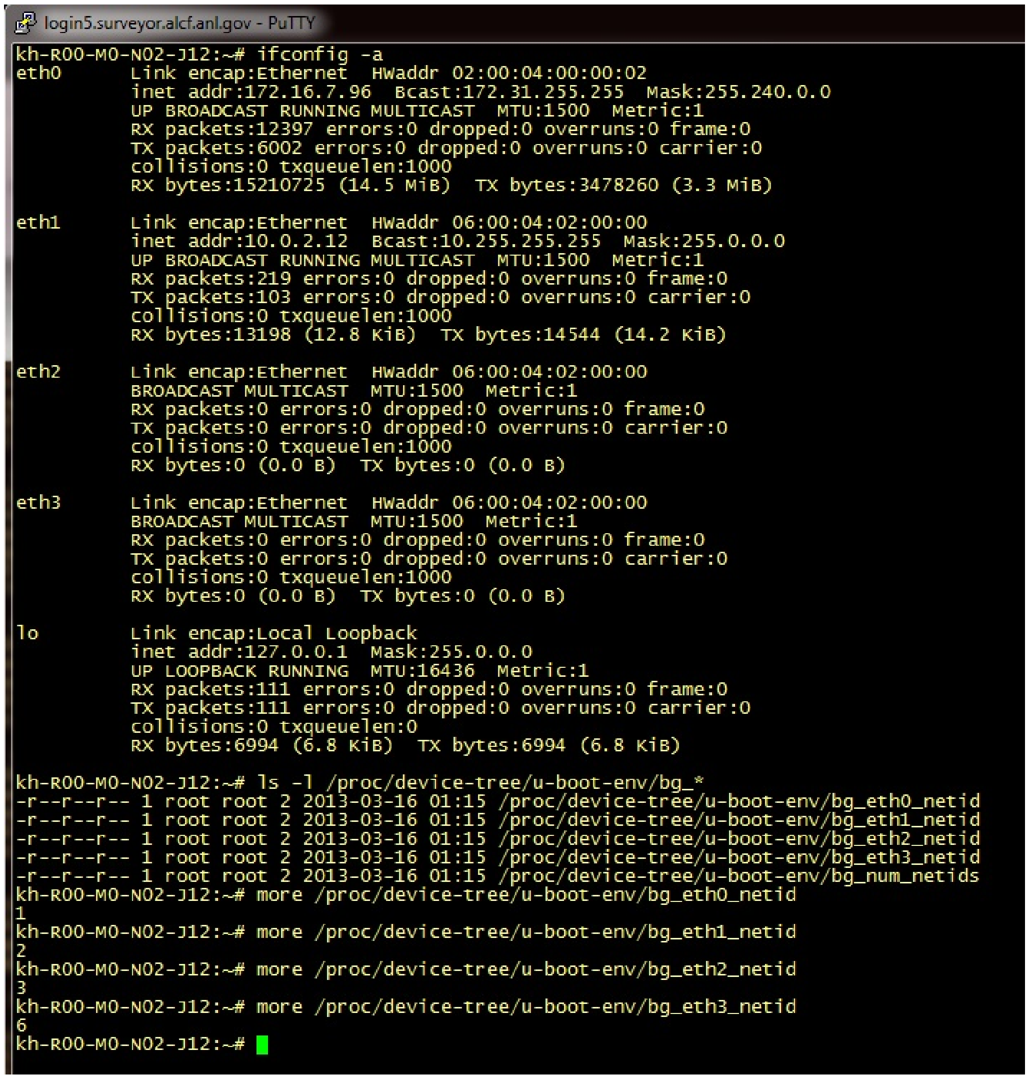
Connect to reservation using ssh on port 22

You will need to have an X server running on your local computer
and click the **Connect!** button from a web browser running on the

Use the following information when you are ready to connect:

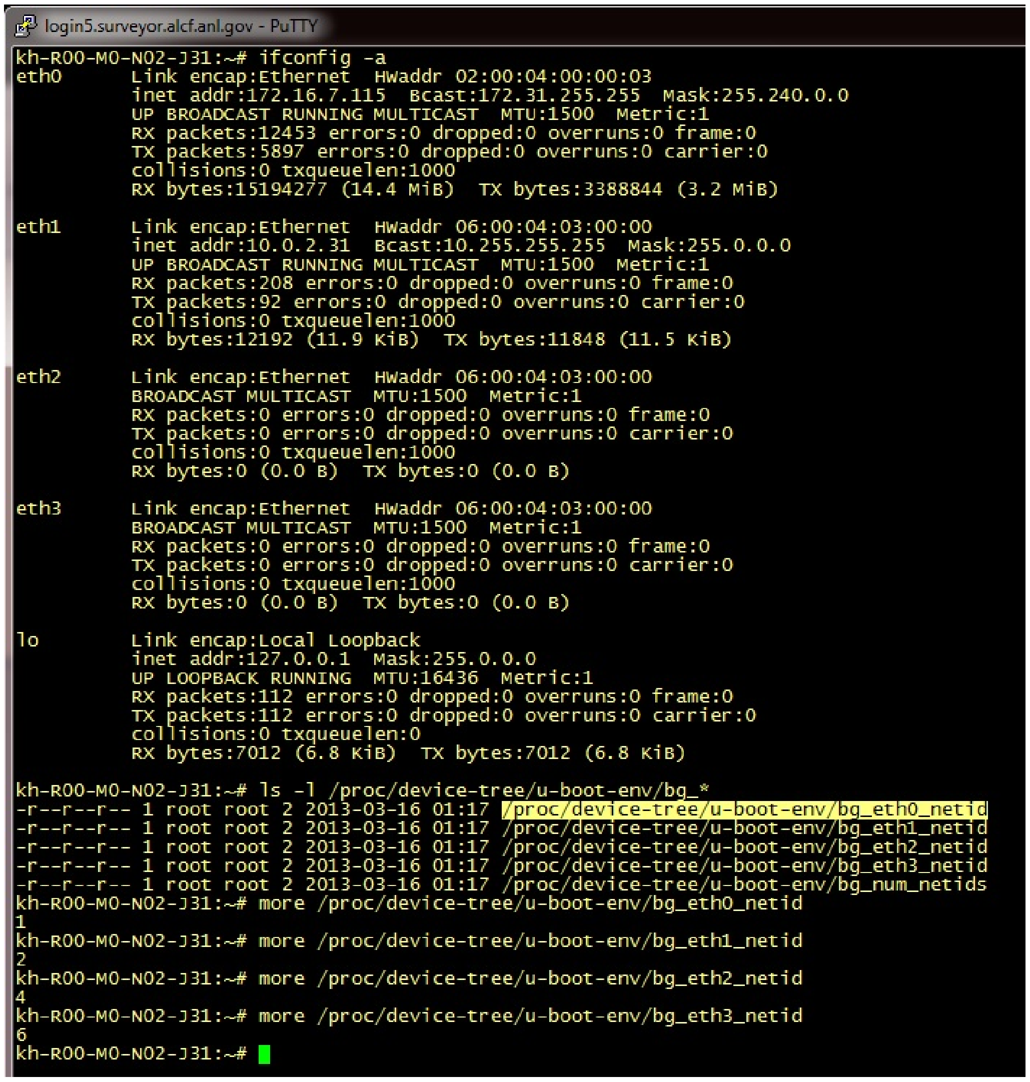
- **Remote Computer**: 172.16.7.106
- **User ID**: admin
- **Password**: (use your campus password)

Figure A.2: Example 1: Cluster Reservation Connect Details



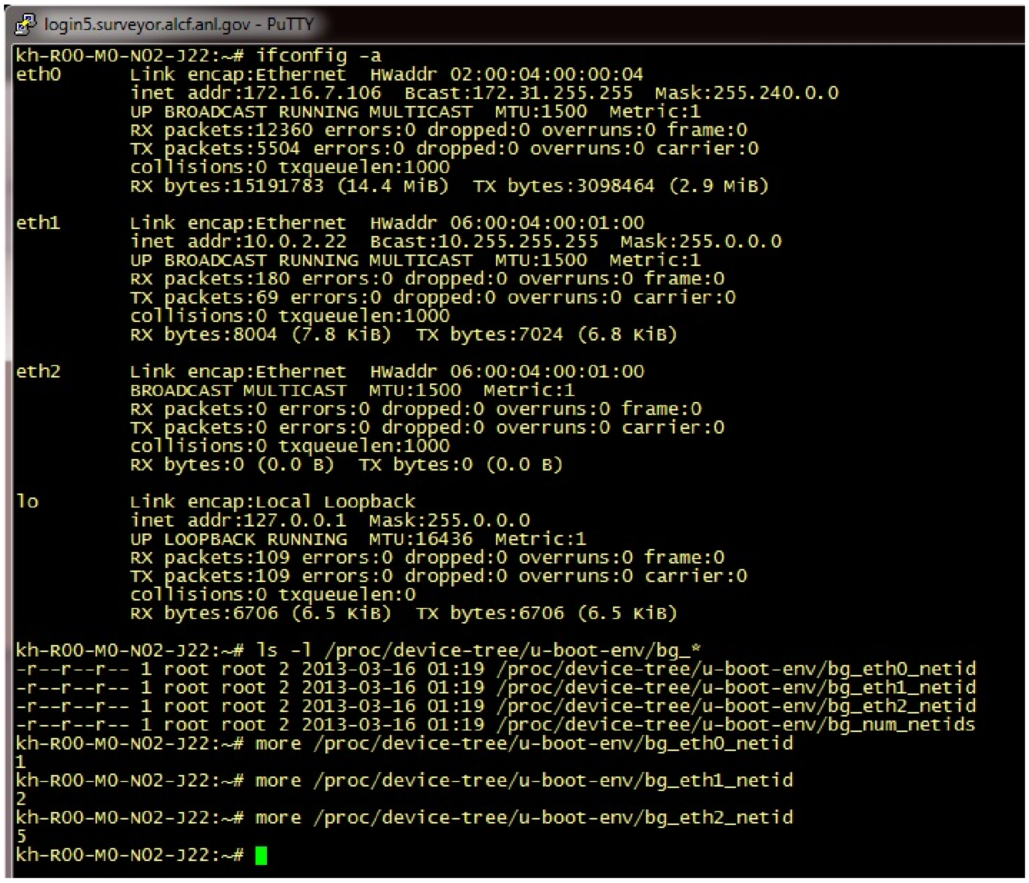
```
kh-R00-M0-N02-J12:~# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 02:00:04:00:00:02
          inet addr:172.16.7.96  Bcast:172.31.255.255  Mask:255.240.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:12397 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6002 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:15210725 (14.5 MiB)  TX bytes:3478260 (3.3 MiB)

eth1      Link encap:Ethernet  HWaddr 06:00:04:02:00:00
          inet addr:10.0.2.12  Bcast:10.255.255.255  Mask:255.0.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:219 errors:0 dropped:0 overruns:0 frame:0
          TX packets:103 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:13198 (12.8 KiB)  TX bytes:14544 (14.2 KiB)

eth2      Link encap:Ethernet  HWaddr 06:00:04:02:00:00
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth3      Link encap:Ethernet  HWaddr 06:00:04:02:00:00
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:111 errors:0 dropped:0 overruns:0 frame:0
          TX packets:111 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:6994 (6.8 KiB)  TX bytes:6994 (6.8 KiB)

kh-R00-M0-N02-J12:~# ls -l /proc/device-tree/u-boot-env/bg_*
-r--r--r-- 1 root root 2 2013-03-16 01:15 /proc/device-tree/u-boot-env/bg_eth0_netid
-r--r--r-- 1 root root 2 2013-03-16 01:15 /proc/device-tree/u-boot-env/bg_eth1_netid
-r--r--r-- 1 root root 2 2013-03-16 01:15 /proc/device-tree/u-boot-env/bg_eth2_netid
-r--r--r-- 1 root root 2 2013-03-16 01:15 /proc/device-tree/u-boot-env/bg_eth3_netid
-r--r--r-- 1 root root 2 2013-03-16 01:15 /proc/device-tree/u-boot-env/bg_num_netids
kh-R00-M0-N02-J12:~# more /proc/device-tree/u-boot-env/bg_eth0_netid
1
kh-R00-M0-N02-J12:~# more /proc/device-tree/u-boot-env/bg_eth1_netid
2
kh-R00-M0-N02-J12:~# more /proc/device-tree/u-boot-env/bg_eth2_netid
3
kh-R00-M0-N02-J12:~# more /proc/device-tree/u-boot-env/bg_eth3_netid
6
kh-R00-M0-N02-J12:~#
```

Figure A.3: Example 1: Node N1 Network interfaces


login5.surveyor.alcf.anl.gov - PuTTY

```
kh-R00-M0-N02-J31:~# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 02:00:04:00:00:03
          inet addr:172.16.7.115  Bcast:172.31.255.255  Mask:255.240.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:12453 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5897 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:15194277 (14.4 MiB)  TX bytes:3388844 (3.2 MiB)

eth1      Link encap:Ethernet  HWaddr 06:00:04:03:00:00
          inet addr:10.0.2.31  Bcast:10.255.255.255  Mask:255.0.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:208 errors:0 dropped:0 overruns:0 frame:0
          TX packets:92 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:12192 (11.9 KiB)  TX bytes:11848 (11.5 KiB)

eth2      Link encap:Ethernet  HWaddr 06:00:04:03:00:00
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth3      Link encap:Ethernet  HWaddr 06:00:04:03:00:00
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:112 errors:0 dropped:0 overruns:0 frame:0
          TX packets:112 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:7012 (6.8 KiB)  TX bytes:7012 (6.8 KiB)

kh-R00-M0-N02-J31:~# ls -l /proc/device-tree/u-boot-env/bg_*
-r--r--r-- 1 root root 2 2013-03-16 01:17 /proc/device-tree/u-boot-env/bg_eth0_netid
-r--r--r-- 1 root root 2 2013-03-16 01:17 /proc/device-tree/u-boot-env/bg_eth1_netid
-r--r--r-- 1 root root 2 2013-03-16 01:17 /proc/device-tree/u-boot-env/bg_eth2_netid
-r--r--r-- 1 root root 2 2013-03-16 01:17 /proc/device-tree/u-boot-env/bg_eth3_netid
-r--r--r-- 1 root root 2 2013-03-16 01:17 /proc/device-tree/u-boot-env/bg_num_netids
kh-R00-M0-N02-J31:~# more /proc/device-tree/u-boot-env/bg_eth0_netid
1
kh-R00-M0-N02-J31:~# more /proc/device-tree/u-boot-env/bg_eth1_netid
2
kh-R00-M0-N02-J31:~# more /proc/device-tree/u-boot-env/bg_eth2_netid
4
kh-R00-M0-N02-J31:~# more /proc/device-tree/u-boot-env/bg_eth3_netid
6
kh-R00-M0-N02-J31:~#
```

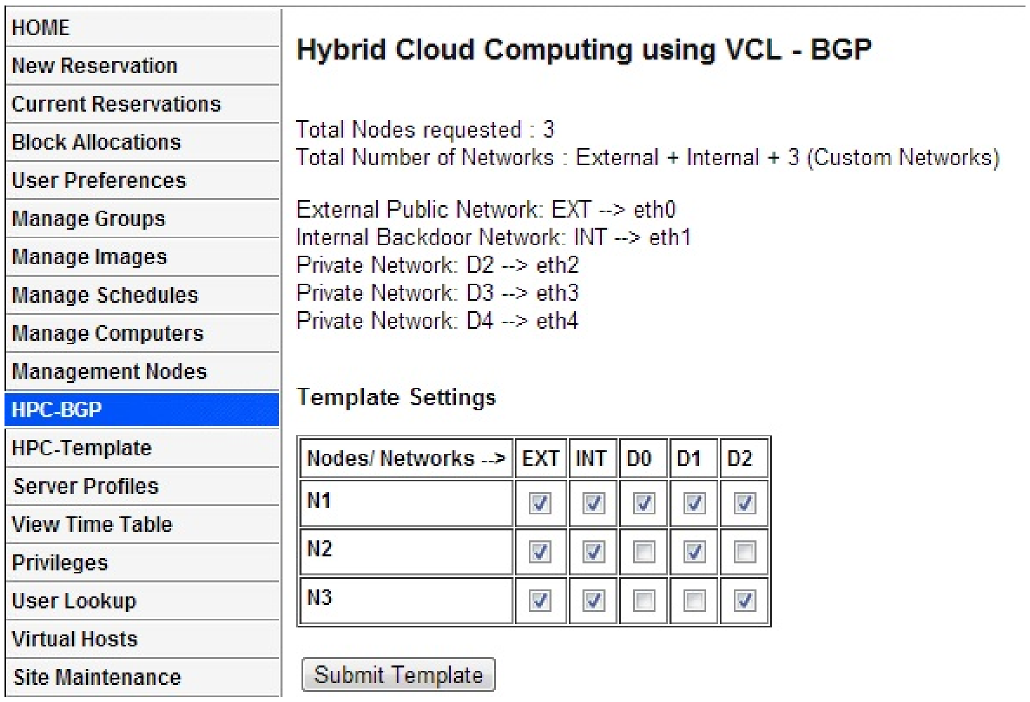
Figure A.4: Example 1: Node N2 Network interfaces



```
login5.surveyor.alcf.anl.gov - PuTTY
kh-R00-M0-N02-J22:~# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 02:00:04:00:00:04
          inet addr:172.16.7.106  Bcast:172.31.255.255  Mask:255.240.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:12360 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5504 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:15191783 (14.4 MiB)  TX bytes:3098464 (2.9 MiB)

eth1      Link encap:Ethernet  HWaddr 06:00:04:00:01:00
          inet addr:10.0.2.22  Bcast:10.255.255.255  Mask:255.0.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:180 errors:0 dropped:0 overruns:0 frame:0
          TX packets:69 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:8004 (7.8 KiB)  TX bytes:7024 (6.8 KiB)

eth2      Link encap:Ethernet  HWaddr 06:00:04:00:01:00
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:109 errors:0 dropped:0 overruns:0 frame:0
          TX packets:109 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:6706 (6.5 KiB)  TX bytes:6706 (6.5 KiB)

kh-R00-M0-N02-J22:~# ls -l /proc/device-tree/u-boot-env/bg_*
-r--r--r-- 1 root root 2 2013-03-16 01:19 /proc/device-tree/u-boot-env/bg_eth0_netid
-r--r--r-- 1 root root 2 2013-03-16 01:19 /proc/device-tree/u-boot-env/bg_eth1_netid
-r--r--r-- 1 root root 2 2013-03-16 01:19 /proc/device-tree/u-boot-env/bg_eth2_netid
-r--r--r-- 1 root root 2 2013-03-16 01:19 /proc/device-tree/u-boot-env/bg_num_netids
kh-R00-M0-N02-J22:~# more /proc/device-tree/u-boot-env/bg_eth0_netid
1
kh-R00-M0-N02-J22:~# more /proc/device-tree/u-boot-env/bg_eth1_netid
2
kh-R00-M0-N02-J22:~# more /proc/device-tree/u-boot-env/bg_eth2_netid
5
kh-R00-M0-N02-J22:~#
```

Figure A.5: Example 1: Node N3 Network interfaces

A.2 Example 2



Hybrid Cloud Computing using VCL - BGP

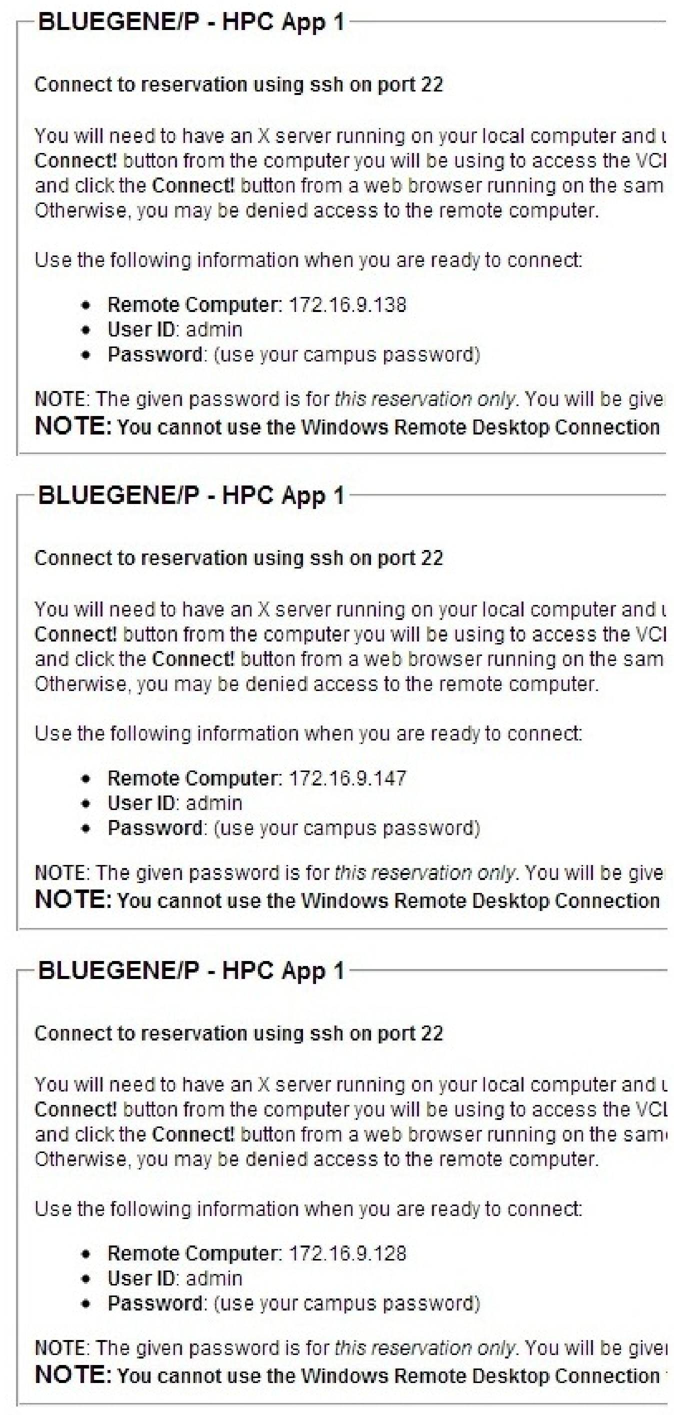
Total Nodes requested : 3
Total Number of Networks : External + Internal + 3 (Custom Networks)

External Public Network: EXT --> eth0
Internal Backdoor Network: INT --> eth1
Private Network: D2 --> eth2
Private Network: D3 --> eth3
Private Network: D4 --> eth4

Template Settings

Nodes/ Networks -->	EXT	INT	D0	D1	D2
N1	✓	✓	✓	✓	✓
N2	✓	✓	☐	✓	☐
N3	✓	✓	☐	☐	✓

Figure A.6: Example 2: Cluster Configuration Template



BLUEGENE/P - HPC App 1

Connect to reservation using ssh on port 22

You will need to have an X server running on your local computer and
Connect! button from the computer you will be using to access the VCL
and click the **Connect!** button from a web browser running on the sam
Otherwise, you may be denied access to the remote computer.

Use the following information when you are ready to connect:

- **Remote Computer**: 172.16.9.138
- **User ID**: admin
- **Password**: (use your campus password)

NOTE: The given password is for *this reservation only*. You will be give
NOTE: You cannot use the Windows Remote Desktop Connection

BLUEGENE/P - HPC App 1

Connect to reservation using ssh on port 22

You will need to have an X server running on your local computer and
Connect! button from the computer you will be using to access the VCL
and click the **Connect!** button from a web browser running on the sam
Otherwise, you may be denied access to the remote computer.

Use the following information when you are ready to connect:

- **Remote Computer**: 172.16.9.147
- **User ID**: admin
- **Password**: (use your campus password)

NOTE: The given password is for *this reservation only*. You will be give
NOTE: You cannot use the Windows Remote Desktop Connection

BLUEGENE/P - HPC App 1

Connect to reservation using ssh on port 22

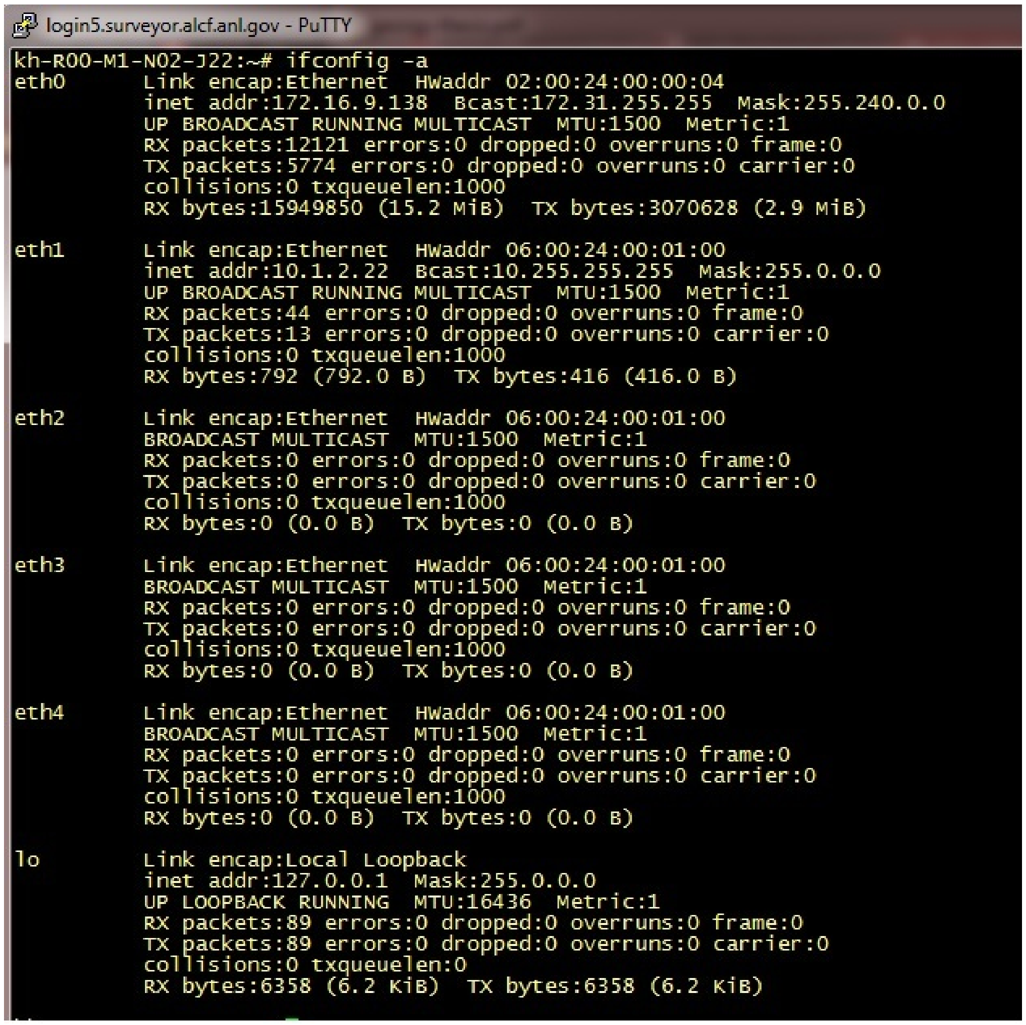
You will need to have an X server running on your local computer and
Connect! button from the computer you will be using to access the VCL
and click the **Connect!** button from a web browser running on the sam
Otherwise, you may be denied access to the remote computer.

Use the following information when you are ready to connect:

- **Remote Computer**: 172.16.9.128
- **User ID**: admin
- **Password**: (use your campus password)

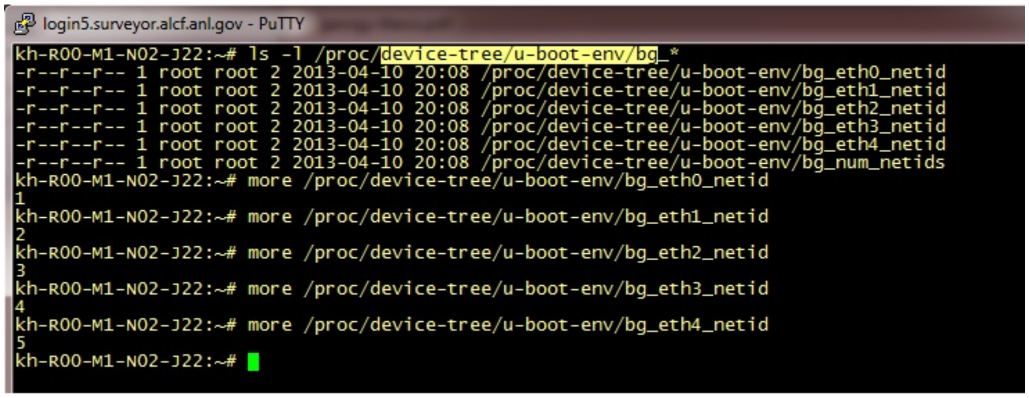
NOTE: The given password is for *this reservation only*. You will be give
NOTE: You cannot use the Windows Remote Desktop Connection

Figure A.7: Example 2: Cluster Reservation Details



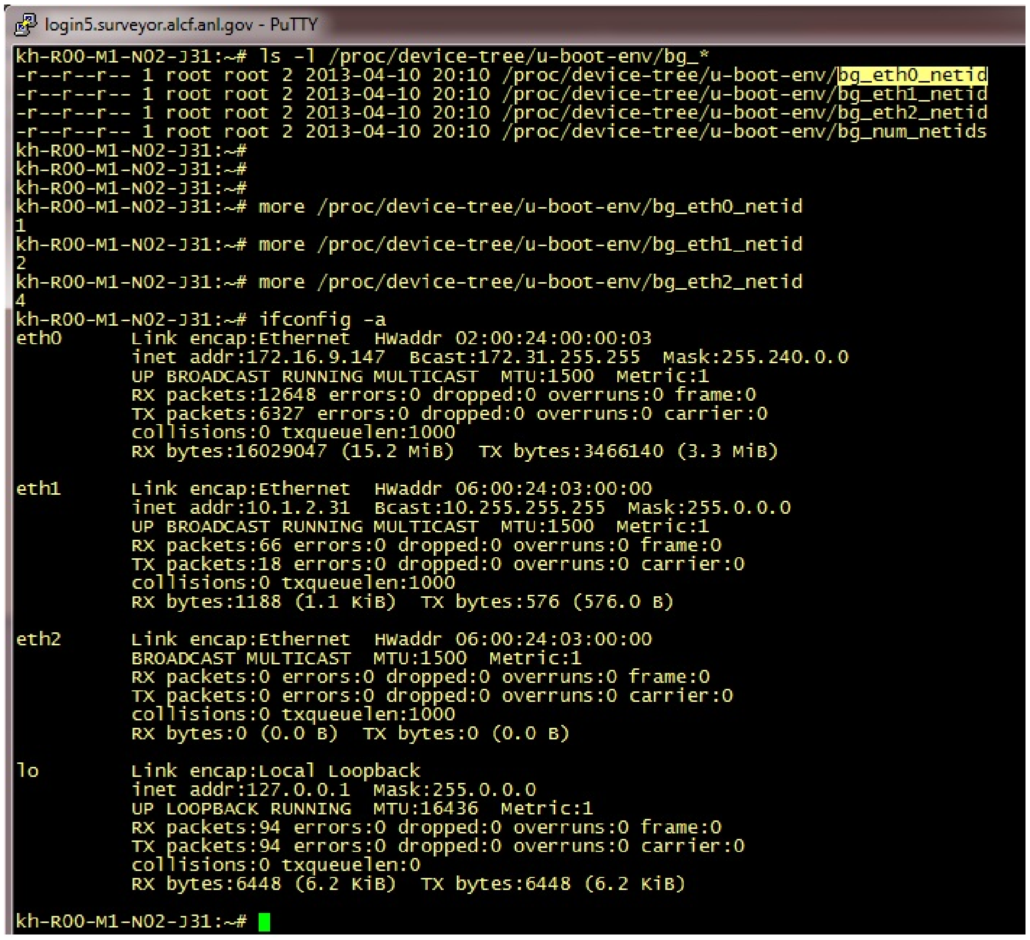
```
kh-R00-M1-N02-J22:~# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 02:00:24:00:00:04
          inet addr:172.16.9.138  Bcast:172.31.255.255  Mask:255.240.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:12121 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5774 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:15949850 (15.2 MiB)  TX bytes:3070628 (2.9 MiB)

eth1      Link encap:Ethernet  HWaddr 06:00:24:00:01:00
          inet addr:10.1.2.22  Bcast:10.255.255.255  Mask:255.0.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:44 errors:0 dropped:0 overruns:0 frame:0
          TX packets:13 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:792 (792.0 B)  TX bytes:416 (416.0 B)

eth2      Link encap:Ethernet  HWaddr 06:00:24:00:01:00
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth3      Link encap:Ethernet  HWaddr 06:00:24:00:01:00
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth4      Link encap:Ethernet  HWaddr 06:00:24:00:01:00
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:89 errors:0 dropped:0 overruns:0 frame:0
          TX packets:89 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:6358 (6.2 KiB)  TX bytes:6358 (6.2 KiB)
```

Figure A.8: Example 2: Node N1 Ifconfig details



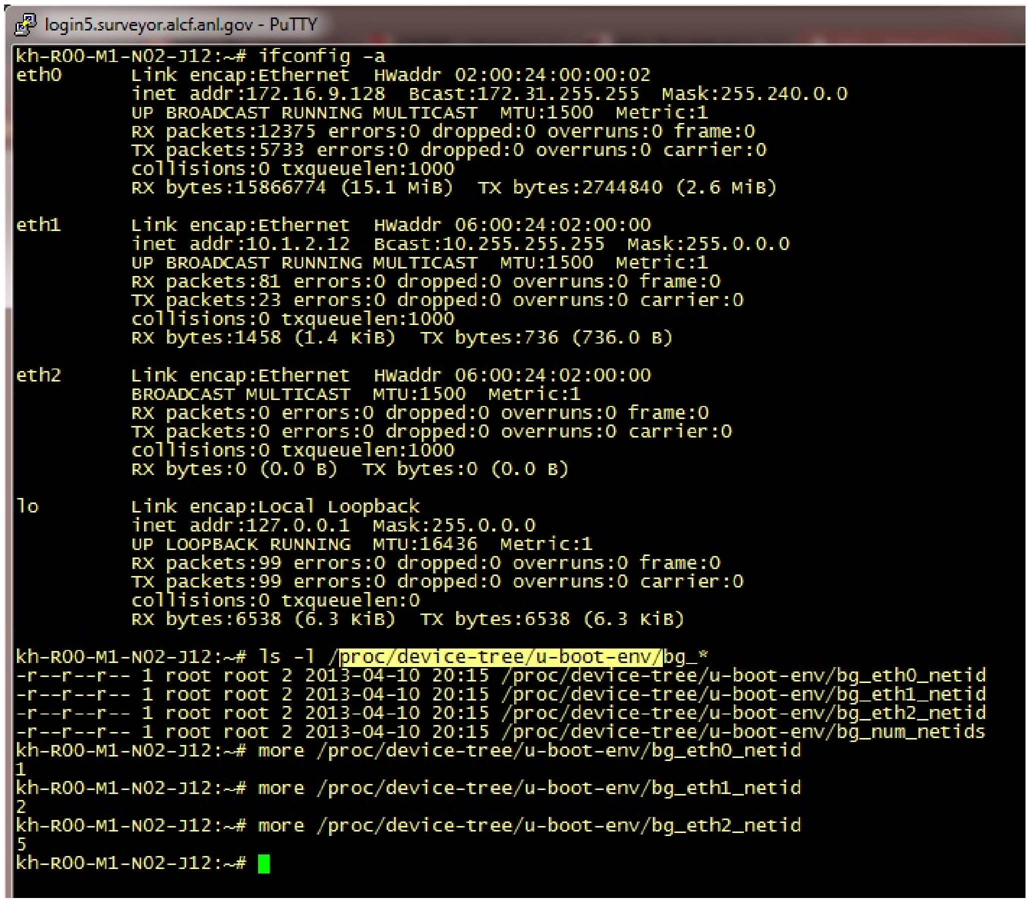
```
kh-R00-M1-N02-J22:~# ls -l /proc/device-tree/u-boot-env/bg_*
-r--r--r-- 1 root root 2 2013-04-10 20:08 /proc/device-tree/u-boot-env/bg_eth0_netid
-r--r--r-- 1 root root 2 2013-04-10 20:08 /proc/device-tree/u-boot-env/bg_eth1_netid
-r--r--r-- 1 root root 2 2013-04-10 20:08 /proc/device-tree/u-boot-env/bg_eth2_netid
-r--r--r-- 1 root root 2 2013-04-10 20:08 /proc/device-tree/u-boot-env/bg_eth3_netid
-r--r--r-- 1 root root 2 2013-04-10 20:08 /proc/device-tree/u-boot-env/bg_eth4_netid
-r--r--r-- 1 root root 2 2013-04-10 20:08 /proc/device-tree/u-boot-env/bg_num_netids
kh-R00-M1-N02-J22:~# more /proc/device-tree/u-boot-env/bg_eth0_netid
1
kh-R00-M1-N02-J22:~# more /proc/device-tree/u-boot-env/bg_eth1_netid
2
kh-R00-M1-N02-J22:~# more /proc/device-tree/u-boot-env/bg_eth2_netid
3
kh-R00-M1-N02-J22:~# more /proc/device-tree/u-boot-env/bg_eth3_netid
4
kh-R00-M1-N02-J22:~# more /proc/device-tree/u-boot-env/bg_eth4_netid
5
kh-R00-M1-N02-J22:~#
```

Figure A.9: Example 2: Node N1 Interface ID's


login5.surveyor.alcf.anl.gov - PuTTY

```
kh-R00-M1-N02-J31:~# ls -l /proc/device-tree/u-boot-env/bg_*
-r--r--r-- 1 root root 2 2013-04-10 20:10 /proc/device-tree/u-boot-env/bg_eth0_netid
-r--r--r-- 1 root root 2 2013-04-10 20:10 /proc/device-tree/u-boot-env/bg_eth1_netid
-r--r--r-- 1 root root 2 2013-04-10 20:10 /proc/device-tree/u-boot-env/bg_eth2_netid
-r--r--r-- 1 root root 2 2013-04-10 20:10 /proc/device-tree/u-boot-env/bg_num_netids
kh-R00-M1-N02-J31:~#
kh-R00-M1-N02-J31:~#
kh-R00-M1-N02-J31:~#
kh-R00-M1-N02-J31:~# more /proc/device-tree/u-boot-env/bg_eth0_netid
1
kh-R00-M1-N02-J31:~# more /proc/device-tree/u-boot-env/bg_eth1_netid
2
kh-R00-M1-N02-J31:~# more /proc/device-tree/u-boot-env/bg_eth2_netid
4
kh-R00-M1-N02-J31:~# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 02:00:24:00:00:03
          inet addr:172.16.9.147  Bcast:172.31.255.255  Mask:255.240.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:12648 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6327 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:16029047 (15.2 MiB)  TX bytes:3466140 (3.3 MiB)

eth1      Link encap:Ethernet  HWaddr 06:00:24:03:00:00
          inet addr:10.1.2.31  Bcast:10.255.255.255  Mask:255.0.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:66 errors:0 dropped:0 overruns:0 frame:0
          TX packets:18 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1188 (1.1 KiB)  TX bytes:576 (576.0 B)

eth2      Link encap:Ethernet  HWaddr 06:00:24:03:00:00
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:94 errors:0 dropped:0 overruns:0 frame:0
          TX packets:94 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:6448 (6.2 KiB)  TX bytes:6448 (6.2 KiB)

kh-R00-M1-N02-J31:~#
```

Figure A.10: Example 2: Node N2 Interface Details



login5.surveyor.alcf.anl.gov - PuTTY

```
kh-R00-M1-N02-J12:~# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 02:00:24:00:00:02
          inet addr:172.16.9.128  Bcast:172.31.255.255  Mask:255.240.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:12375 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5733 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:15866774 (15.1 MiB)  TX bytes:2744840 (2.6 MiB)

eth1      Link encap:Ethernet  HWaddr 06:00:24:02:00:00
          inet addr:10.1.2.12  Bcast:10.255.255.255  Mask:255.0.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:81 errors:0 dropped:0 overruns:0 frame:0
          TX packets:23 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1458 (1.4 KiB)  TX bytes:736 (736.0 B)

eth2      Link encap:Ethernet  HWaddr 06:00:24:02:00:00
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:99 errors:0 dropped:0 overruns:0 frame:0
          TX packets:99 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:6538 (6.3 KiB)  TX bytes:6538 (6.3 KiB)

kh-R00-M1-N02-J12:~# ls -l /proc/device-tree/u-boot-env/bg_*
-r--r--r-- 1 root root 2 2013-04-10 20:15 /proc/device-tree/u-boot-env/bg_eth0_netid
-r--r--r-- 1 root root 2 2013-04-10 20:15 /proc/device-tree/u-boot-env/bg_eth1_netid
-r--r--r-- 1 root root 2 2013-04-10 20:15 /proc/device-tree/u-boot-env/bg_eth2_netid
-r--r--r-- 1 root root 2 2013-04-10 20:15 /proc/device-tree/u-boot-env/bg_num_netids
kh-R00-M1-N02-J12:~# more /proc/device-tree/u-boot-env/bg_eth0_netid
1
kh-R00-M1-N02-J12:~# more /proc/device-tree/u-boot-env/bg_eth1_netid
2
kh-R00-M1-N02-J12:~# more /proc/device-tree/u-boot-env/bg_eth2_netid
5
kh-R00-M1-N02-J12:~#
```

Figure A.11: Example 2: Node N3 Interface Details

APPENDIX B

Installation Manual

B.1 User Manual for VCL Installation

B.1.1 MySQL Installation and Configuration

1. Login into ANL Surveyor Node. You will be in your home directory (use pwd command to check)

2. Next you have to change directory to /pvfs-surveyor/georgy

   Command: cd /pvfs-surveyor/georgy

3. Do command ls -l. The output should be as follows, (5 directories)

   drwxr-xr-x 1 georgy users 4096 2013-01-24 02:51 lenny

   drwxr-xr-x 1 georgy users 4096 2013-02-14 03:36 MYSQL

```
drwxr-xr-x 1 georgy users 4096 2013-01-24 15:07 perl
drwxr-sr-x 1 georgy users 4096 2013-01-24 12:32 vcl
drwxr-xr-x 1 georgy users 4096 2013-02-14 03:48 vcl_config
```

4. Change directory into MYSQL. (command : cd MYSQL). There will be a tar ball called (mysql.tzr.gz)

5. Copy this tar ball into your home directory.

   cp mysql.tar.gz ∼/

6. Untar the mysql.tar.gz file. This will create a new directory called mysql

7. Change directory into mysql (command: cd mysql)

8. Do an ls -l. It should have the following

   ```
   -rw-r–r– 1 georgy users 3308551 2013-02-14 03:48 clean.sql
   -rwxr-xr-x 1 georgy users 1646 2013-01-23 19:40 installer
   drwxr-xr-x 1 georgy users 4096 2013-01-23 21:06 mysql-5.1.60
   -rwxr-xr-x 1 georgy users 873 2012-10-13 02:03 mysqld
   drwxr-xr-x 1 georgy users 4096 2013-01-23 19:30 ncurses
   ```

   clean.sql : the database file to be loaded after mysql is completely and correctly installed.
   installer : file listing the commands for installing mysql
   mysql-5.1.60 : Directory which consists of mysql installation files
   mysqld : daemon to turn on and turn off mysql database
   ncurses : library required for correct functionality

9. change directory into mysql-5.1.60 (command : cd mysql-5.1.60)

10. Run the command :

    CFLAGS=”-O3” CXX=gcc CXXFLAGS=”-O3 -felide-constructors -fno-exceptions -fno-rtti” ;

11. Run the command:

```
./configure –prefix=/home/$USER/MySQL/mysql \
–enable-assembler \
–enable-thread-safe-client\
–with-mysqld-user=$USER \
–with-unix-socket-path=/home/$USER/MySQL/mysql/tmp/mysql.sock \
–localstatedir=/home/$USER/MySQL/mysql/data \
–with-named-curses-libs=/home/$USER/mysql/ncurses/lib/libncurses.a
```

12. Run the command:

    make

13. Run the command:

    make install

    *****(very important to check the ending message on running the above commands for leaving with error) this means the installation was in error due to some problem**

14. This will install mysql in the home directory. It will be in the following directory ~/MySQL/mysql

15. Change into the directory /MySQL/mysql (cd ~/MySQL/mysql)

16. Now in this directory you need to make two directories (log and etc)

Command: mkdir log ,mkdir etc

log directory will have have thesql log files, etc directory for configuration file

17. Change into the etc directory (Command: cd etc)

    Touch a new file called as my.cnf (this is mysql configuration file). If the service needs to be initiated on a different port, this file needs to be edited.

18. Run the following command to load the contents into my.cnf

```
cat<<EOF >/home/$USER/MySQL/mysql/etc/my.cnf
[mysqld]
user=$USER
basedir=/home/$USER/MySQL/mysql
datadir=/home/$USER/MySQL/mysql/data
port=3306
socket=/home/$USER/MySQL/mysql/tmp/mysql.sock
bind-address=127.0.0.1
[mysqld_safe]
log-error=/home/$USER/MySQL/mysql/log/mysqld.log
pid-file=/home/$USER/MySQL/mysql/mysqld.pid
[client]
port=3306
user=$USER
socket=/home/$USER/MySQL/mysql/tmp/mysql.sock
[mysqladmin]
user=root
port=3306
```

```
socket=/home/$USER/MySQL/mysql/tmp/mysql.sock
[mysql]
port=3306
socket=/home/$USER/MySQL/mysql/tmp/mysql.sock
[mysql_install_db]
user=$USER
port=3306
basedir=/home/$USER/MySQL/mysql
datadir=/home/$USER/MySQL/mysql/data
socket=/home/$USER/MySQL/mysql/tmp/mysql.sock
EOF
```

19. Open my.cnf if the configuration file is correct

20. Come back to your home directory (Command: cd ∼/)

21. Export the path variable for the new installed mysql database. Open .bashrc file in your home directory and add the following line,

    export PATH=/home/(username)/MySQL/mysql/bin:$PATH

    ***** replace (username) with your userid**

22. Load the .bashrc file to reload the environment variables

    Command: . /.bashrc

23. Now check if it picks up the new mysql that is installed.

    Command: which mysql

    (you should see /home/(userid)/MySQL/mysql/bin)

24. If this is not the output then the path variable is not loaded properly and thus you wouldnt be able to access the new installed database.

25. Go back to the mysql directory command: cd ∼/mysql. Copy the mysqld daemon from the extracted tarball in to the ∼/MySQL directory.

26. Now run the following command: /home/$USER/MySQL/mysql/bin/mysql_install_-db

    ***** replace $USER with your userid**

27. Now start the mysql service:

    Command: /home/(username)/MySQL/mysql/bin/mysqld_safe –user=(username) &

    ***** replace (username) with your userid**

28. After this is done, we need to set the root password for root user to access the database. Issue the following command:

    Command: /home/(username) /MySQL/mysql/bin/mysqladmin -u root password (****** password you wish to give)

    This will ensure that root login can be done into the database only using the given password.

29. Now try logging into the database, since all the users are removed only root will be able to login

    Command: mysql u root p

    On prompt for the password enter the password you configured for root.

30. Quit out of the databse.

31. Run the command from your home directory:

    /home/(username)/MySQL/mysql/bin/mysql_secure_installation

    On running the above command you can follow these answers

    Change the root password? [Y/n] - n

    Remove anonymous users? [Y/n] - y

    Disallow root login remotely? [Y/n] - y

    Remove test database and access to it? [Y/n] - y

    Reload privilege tables now? [Y/n] - y

32. Now for your username you can give a passwordless access to the database,

    To give access to the Home User:

    Login into the database: mysql u root p (enter password on prompt)

    Now type the following command: (** replace username with your own username)

    GRANT ALL PRIVILEGES ON *.* TO '(username)'@'localhost' WITH GRANT OPTION;

33. Quit out of the database

34. Now check if you can directly access the database

    Command: mysql

35. Quit out of the database

36. Do a ps ef — grepmysqland kill the mysql database which is running.

37. Now you can use the mysql daemon to start and stop the mysql service

38. Change to ~/MySQL directory. This will contain the mysqld file

    Command: mysqld (start/stop)

39. Creating the vcl database

40. Login into the mysql database

    Command: mysql

    Command: CREATE DATABASE vcl;

41. Exit out of the database.

42. Load the vcl database tables; (the clean.sql file included in the extracted tar ball, copy it to you home directory)

    Command: mysql vcl <clean.sql

43. To check the correctness of the above step

    Login into the mysql database

    Command: mysql

    use vcl;

    show tables;

44. The above command should list 75 tables which are present in VCL Database.

B.1.2 Perl Module Installation and VCL Configuration

1. change directory to /pvfs-surveyor/georgy

   Command: cd /pvfs-surveyor/georgy

2. Do an ls -l

   perl directory contains all the perl modules to be installed and the installation scripts vcl directory all the vcl modules and code base

vcl_config directory contains all vcl configuration files, like vcld daemon

3. **The management node is not installed on the login node rather it is installed on one of the BlueGene nodes on the lenny image**

4. Inside the directory /pvfs-georgy/georgy/ there will be another directory called lenny which consists of the lenny image.

5. Change directory into lenny

   Command: cd /pvfs-surveyor/georgy/lenny

6. There will be a file called as lenny.tar.gz. You can copy this into your home directory and then untar it.

   cp lenny.tar.gz ∼/

7. Now go back to you home directory and untar the lenny image. This will produce a new file called lenny.img in the home directory.

8. Now you need to start a blugene job before we go ahead with perl modules installation, as these installations should be done on the lenny.img

9. Load the Kittyhawk environment variables. This file is called as kh.env. This will either be there in your Home directory itself or inside a directory called as Work. If the file does not exist create a new file with the following contents, and save it as kh.env

   ```
   export BGPPROFILEDIR=/bgsys/argonne-utils/profiles
   export BGPROFILE=kh
   ```

export PATH=$PATH:$BGPPROFILEDIR/$BGPROFILE/opt/bin

export PATH=$PATH:$BGPPROFILEDIR/$BGPROFILE/scripts

export PATH=$PATH:$BGPPROFILEDIR/$BGPROFILE/bin

export PATH=$PATH:$BGPPROFILEDIR/$BGPROFILE/khdev/bin

export PATH=$PATH:$BGPPROFILEDIR/$BGPROFILE/khdev/kernels

export KH_Project="VCL-ON-BG_P"

Exporting the file command: .kh.env

10. Come back to your home directory, command: cd ∼/

11. Now you have to start the BlueGene Job

    Command: khqsub 60 64

    (your requesting for 64 nodes for a period of 60 minutes.)

    This will take around 2 mins, In between it will ask for a password prompt. The password is kh

12. Export the Kittyhawk control server. For this copy paste the last line which is returned by executing the khqsub command.

    For eg, export khctlserver=172.16.5.8

13. Creating a public private key pair. Command: ssh-keygen -t rsa

    This will save the key pair id_rsa and id_rsa.pub in /.ssh directory. Make sure that it does not have a passphrase associated with it.

14. Now we need to trigger a khdev node for creating a management node image. Command:

    khdev -K "$(cat ∼/.ssh/id_rsa.pub)" **/pvfs-surveyor/georgy/lenny/lenny.img**

(The last parameter is basically the path where you have untared the image :lenny.img, Wherever it is situated please give that path.)

15. After the khdev command is over you will be directly loaded into the khdev node. Note down the eth0 ip address of the khdev node which will be useful in step 15.

16. You will have to exit out of that node come back to the surveyor node and then scp all the necessary directories into the khdev node.

17. Now you need to scp three directories into the khdev node,
    perl, vcl and vcl_config which is inside /pvfs-surveyor/georgy. Commands are as follows:
    scp -r /pvfs-surveyor/georgy/perl root@(IP address from step 14):∼/
    scp -r /pvfs-surveyor/georgy/vcl root@(IP address from step 14):∼/
    scp -r /pvfs-surveyor/georgy/vcl_config root@(IP address from step 14):∼/

18. Now ssh back into the khdev Node, to setup the gatech repositories in order to install some packages.
    Open /etc/apt/sources.list and change the line deb http://localhost/debianlenny main into deb http://localhost/debian-archive/debian/lenny main

19. Now exit out of the khdev node again.
    Command: exit

20. Now when we ssh back into the khdev node we need to mention a reverse tunneling option in order to install a few packages.
    Command: ssh R 80:debian.gtisc.gatech.edu:80 root@(IP_address of khdev node)

21. In order to check the functionality do a sudo apt-get update to check whether it works.

22. If the above step works successfully, go ahead and install mysql-server. This is done for solving some package dependencies for the vcl code modules.

    Command: apt-get install mysql-server

23. It will ask you to enter a root password, enter a simple one, as it is not going to be used.

24. After the installation go and kill the process running the mysqld service on the khdev node.

    Command: ps -ef | grep mysqld

    Kill the process id.

25. Update the startup file on the lenny.img to avoid the mysqld starting at boot. This is done because the vcld installed on the image should connect to the DB installed on the login node and not the local one.

    Command: update-rc.d -f mysql remove

26. Now after making sure that the mysqld service is stopped and not running we can go ahead and install the perl modules.

27. Go to the home directory in khdev node.

28. Do a command ls, You should find the directory perl - which has all the modules

29. Change directory into perl

    Command: cd perl

30. There are three script files in that directory : script1, script2 and script3

    Run all of them in order script1, script2, script3

    ./script1

    ./script2

    ./script3

31. Script 1 and 2 will be executed pretty fast, but Script 3 will take some amount of time.

32. Exit out of the perl directory and come back to the home directory.

33. After the perl module installation is done now move the vcl directory into /usr/local/

    Command: mv vcl /usr/local

34. Now we have to prepare the vcl configuration files. Change into vcl_config directory.

    Command: cd vcl_config

35. Do an ls -l in that directory. You will find that there is file called vcld. This is nothing but the vcld daemon, which will be used to start and stop the vcld service. But before that we have to configure the vcld.conf file

36. Open the vcld.conf file. Go down in the file there will be a section called as wrtpass= Whatever password was configured for root user which installing mysql database needs to be mentioned here, for eg. if the password is abcd

    wrtpass=abcd

37. Save the vcld.conf file

38. Exit out of the khdev node. Now we need to test whether the vcld can connect to the database and work properly.

39. Now we need to do a reverse tunneling for the database while we log back into the khdev node which from now on is our management node (perl modules and vcl installed).

    Command: ssh R 3306:127.0.0.1:3306 root@IP of khdev node

    If an alternate port is used for firing up mysql service, the above mentioned command will change. For example consider mysql is fired up on port number 3508, the command will be as follows:

    Command: ssh R 3306:127.0.0.1:3508 root@IP of khdev node

40. Now we need to change the name of the vcl_config directory to vcl

    Command: mv vcl_config vcl

41. Now change directory into vcl.

    Command: cd vcl

42. Start the vcld process: Command: perl vcld start

43. Tail the vcld log file to see if it can read from the database and there is last checkin time updated every 5 seconds

    Command: tail f /vcl/vcld.log

44. Use halt command to stop the khdev node.

45. Now the lenny image will have all the necessary modules installed on it. You can rename the image as MN_lenny (image specific for Management Node).

B.1.3 Web Interface Configuration

1. Go to vcl.ncsu.edu

2. Take a reservation for VCL to Surveyor Image
3. Hit connect button and login into a ssh shell using the IP address provided.
4. When you login perform an ls -la to see the files
5. There will be one tun file and other .scratch
6. Edit the tun file for port forwarding changes and username changes.

   Command: ssh -t -f -N -i .scratch -L 3306:127.0.0.1:(mysql-port number) (username)@login-node

   mysql-port number : Port number at which mysql service is running on the login node of the surveyor machine.

   username : anl account username
7. .scratch should be the private key file to login into your ANL account. This key should be without a pass-phrase.
8. Give correct permission to the .scratch file (chmod 600 .scratch)
9. So remove the existing one. Make a new file called .scratch and copy paste your private key into that file.
10. Now go to the location /var/www/html/vcl/.ht-inc/
11. Here you should find a file called secrets.php. Open it and we have to edit the password.

    $vclpassword=csc591team4mogambo571820. This is the password that it uses to connect to the mysql database that is installed on the ANL surveyor node. Change it to the password which you configured while mysql installation

    $vclpassword = assign the new pass

12. Go back to the home directory. Command: cd ~/
13. Once this is done, try running the tun file (remember the mysql should be running on the surveyor node).
    Command: ./tun
14. Now go to your web Browser and enter the Image reservation IP. You should get a login Interface
15. The username is : admin and password is: csc591team4mogambo571820
16. You can change this password by going into User Preference tab in VCL.
17. Once the passwords are changed, you can go to the login node and remove a clean mysql dump which can be used later on.
    Command: mysqldump vcl >clean(time_stamp).sql
18. You should be able to see two tabs HPC-BGP and HPC-Template
19. Copy the private key .scratch from your home directory to /var/www/html/vcl/.ht-inc/
20. Rename the key in /var/www/html/vcl/.ht-inc directory as keyan
21. Make sure the ownership of the keyan file is correct
    Command: chown apache:apache keyan
22. Edit the files t1.sh, t2.sh and t.sh. Change the username to your account username.
23. Go to /var/www/html/vcl - There will be two files jb.php and jp.php
    In jb.php , go to line 144, rename from (username) to (account-username)
    In jp.php go to line 2 and rename from (username) to (account-username)

24. Now go to /var/www/html/vcl/.ht-inc, There will be file hpc.php
    Line no: 704, rename from (username) to (account-username)

25. In /var/www/html/vcl/.ht-inc, next there will be a file template.php
    Line 262: rename from (username) to (account-username)

26. Now you can go ahead and make an Image VCL to surveyor_username and use it for your own purpose as it has got your private key. Image creation can be done using Manage Image functionality in vcl.

B.2 Editing the Scripts

1. trigger.sh : Line 124 (list the path of the image where you have the management node lenny image stored)
   spawn ./khdev.copy -p $net -K "$(cat /.ssh/id_rsa.pub)" (path of the image)

2. khdev.copy : Line 589 (port forwarding for database access for the vcld, -R 3306:127.0.0.1:3306) please change based on the port you have selected.

3. khdev_template: Line 591 (same as above change for the port)

4. provision.sh : Line 48 Again insert the path of the image as in step1

B.3 Source Code

The source code can be found in the GIT Repository(https://github.com/georgymathew/VCL-on-BlueGene-P.git).