Fast and Simple Deployment of a Linux® Cluster Data Warehouse

by

Belal Tassi

Department of Electrical and Computer Engineering

University of Waterloo

In fulfillment of the project requirement for the degree of

Master of Applied Science

in

Computer Engineering

Abstract

Collections of distributed computers, termed clusters, have been used for decades to help solve some of the world's most complicated problems. The popularity of these clusters, particularly those built with "off-the-shelf" commodity components, has been steadily increasing over the years, but the cluster-related skills and time budgeted to properly deploy them have not. This paper examines the current research in the area of Linux cluster-deployment and associated deployment tools. Subsequently, it discusses one such approach in detail that was successfully used to automate and simplify deployment of Linux clusters running an important application that these clusters capably support – namely, a distributed data warehouse application. Requirements, design decisions, and process flow of the approach are examined as well as technical details about the tool itself. This tool has successfully been used by IBM® to deploy clusters at a multitude of sites all over the world. Lessons learned about this technique from these deployments, as well as potential future work stemming from these experiences, are discussed.

Contents

List of Figures

List of Tables

1.0 Introduction

It has oft been mentioned that in computing there are three basic approaches to improving performance [1]:

1) Use a better algorithm.
2) Use a faster computer.
3) Divide the calculation among multiple computers.

Since options one and two are not technically or financially feasible in many cases, the third choice – normally termed parallel or distributed computing – has become the choice for many large computing problems. To use a house construction analogy, distributed computing allows a whole team of workers to simultaneously work on building the house faster – without the need to invent a better house-building technique or to buy the latest most expensive house-building tool.

The computer cluster is the modern workhorse of the distributed-computing approach – and for many applications, Linux clusters in particular, are the workhorse of choice. One application where Linux clusters have really shown their mettle is running distributed data warehouse systems. Data warehouses are large repositories of an organization's historical data – used to contain the raw information needed for decision support systems and business-intelligence tools (such as data mining), which are an increasingly critical part of the modern enterprise. Linux clusters are making these large distributed systems feasible from a

performance, technical, and financial perspective for numerous enterprises, educational, and research organizations.

With large numbers of these Linux clusters being deployed, the ease, speed, and simplicity of deployment have become critically important issues – especially for organizations such as IBM which are deploying a multitude of these clusters daily. There is a need for simple tools to speed deployment, increase consistency, and decrease the technical skills needed to be able to rapidly deploy a data warehouse system running on a Linux computer cluster.

This paper covers the aforementioned issue in detail. Chapter 2 takes a panoramic survey of current research in the area of Linux-cluster-deployment approaches and associated deployment tools. This is followed by details about one such tool - the DB2ISERVER deployment tool - which was built to help make deployment of Linux cluster's running the IBM DB2® data warehouse system as simple as possible. Chapter 3 describes the requirements, software and hardware design decisions, and overall approach taken by the DB2ISERVER tool. In Chapter 4, implementation details about the DB2ISERVER approach and tool are discussed. The tool has been used extensively by IBM to deploy clusters in the field; lessons learned from these deployments are discussed in Chapter 5. The chapter ends with a brief discussion of future work that is currently being implemented in the next generation of this tool. Chapter 6 offers a summary and some concluding thoughts. Finally, Appendix A contains detailed reference information about the DB2ISERVER tool and the DB2ISERVER configuration file which is the primary input required by the tool.

2.0 The Linux Cluster Landscape

This chapter is a survey of current research in the area of Linux cluster deployment and associated deployment tools.

2.1 What is a Cluster?

The term *cluster* is one of those overloaded computing terms (like "node") that can have a plethora of meanings based on context, and hence, should always be explicitly defined when used. In the context of this paper, a computer cluster is defined as "a group of loosely coupled computers that work together to accomplish a specific task or tasks but is viewed externally as though it is a single computer" [2]. One important differentiation is that the term *cluster computing* is different from the related, but distinct, field of grid computing. The term *grid computing* is generally used to describe multiple computers in different locations working together across a wide-area network (WAN), typically the Internet. In contrast, clusters are generally in one location and restricted to computers on the same sub-network. In general, the significant differences between cluster computing and grid computing are that computing grids typically are a much larger scale, tend to be used more asynchronously, and have much greater access, authorization, accounting and security concerns [1].

Why use a cluster instead of just using a larger, more powerful computer to accomplish the same task? In addition to the fact that for some very large tasks a computer that could accomplish it on its own has yet to be created, clusters are usually deployed to improve performance and availability over that provided by a single computer, while typically being much more cost-effective than a single computer of comparable speed or availability [2].

2.2 The Many Faces of the Linux Cluster

Any computer cluster contains four fundamental parts:

1) The collection of computer hardware that provides the processing power of the cluster.

2) The network that connects these computers together.

3) The I/O storage system used to store the cluster's data.

4) The distributed software applications that are being run on the cluster.

What components are used for each of these four parts, and how these components are put together will determine the primary characteristics of the cluster: processing power, I/O throughput, price, capacity, and overall performance.

Over the past four decades, clusters have been put together in many different ways and have produced many different types of computing clusters. One very important development in this progression of clustering technology was performed by Thomas Sterling and Donald Becker while working for a NASA contractor in the mid-1990s. At that time, when they created a 16-PC cluster, they effectively pioneered the commodity Linux cluster, under a name that has in most people's minds become synonymous with Linux clusters: a "Beowulf" cluster [3,4]. The Beowulf concept commonly involves using commodity servers and open-source software to cluster large numbers of inexpensive Linux computers together, effectively to enable supercomputer power inexpensively. These "commodity clusters", with all four fundamental parts of the cluster built from relatively inexpensive and commonly available components, have brought parallel computing, supercomputing power to the masses. Academic institutions and other similar organizations that have large and complex

problems to solve, but limited financial resources, were quick to start building their own Beowulf-like clusters.

How successful have these commodity clusters become? Today, not only can one build one's own Beowulf Linux cluster - and many educational facilities and companies are doing just that – but even more easily, they can be purchased from IBM, HP, Sun Microsystems, and a host of other major computer vendors. As far back as 2001, there were already 28 Beowulfs in the Top500 supercomputers list, a running list of the most powerful computers in the world [5]. It has been proclaimed that Beowulf's economics and sociology are currently poised to kill off the other cluster architectural lines – and will likely affect traditional supercomputer centers as well [5]. Since then, the popularity of Beowulf clusters has only grown (see current Top500 list at http://www.top500.org), and it is now not unheard of for even secondary-school classrooms to have their own Beowulf clusters "just to kick the tires" [5].

Originally, these Beowulf Linux clusters were exclusively used to run applications in the high-performance computing (HPC) space. This has since expanded to include other primary uses, in particular, high availability (HA) and load balancing.

High availability clusters are composed of multiple machines but only a subset of these machines are needed to actually provide the service the cluster is providing [2]. For example, if 2 out of the 5 machines that make up a database server cluster fail, the distributed database would continue to run and service requests on the 3 remaining machines. In fact, in most

cases, this failure should be transparent to the connected users, except for potentially reduced performance.

Load-balancing clusters are built to provide better performance by dividing the work among the group of computers in the cluster. The classic example is the Web-server cluster, sometimes termed a "server farm", in which incoming queries are distributed among all the servers in the cluster based on a distribution algorithm. The reality is that, in most cases, there is considerable overlap with these 3 primary uses of clusters: HPC, HA, and load balancing. For example, a standard HPC cluster would, in many cases, also provide some high-availability characteristics as well.

2.3 A Survey of Current Linux Cluster Deployment Research

An extensive literature search of published technical papers pertaining to Beowulf Linux clusters was performed.

Numerous papers dealt with research issues around stateless or "diskless" clusters. Stateless clusters run without requiring persistent storage from every node in the cluster. The idea is for this to reduce cost, cut down on noise, lessen cooling requirements, curtail power consumption, and minimize the number of moving parts in a cluster, thereby improving reliability [6]. These classes of clusters and associated deployment technologies were ignored for the purposes of this paper because they are unable to satisfy the requirements of a data warehouse system.

Many of the research papers focused on performance issues in distributed cluster computing [7,8]. Other papers focused on the application of these Linux clusters to specific application domains [9,10]. Moreover, many of the papers focused on deployment and management of wide-area Grids – normally composed of non-dedicated machines [11]. These topics will also not be addressed in this paper, as they are unrelated to our specific problem of rapid deployment of data warehousing clusters.

One area of research that is strongly linked to this paper's commodity-cluster deployment focus is that of cluster administration and configuration.

2.3.1 Cluster Administration

System administrators of medium and large-sized clusters are often faced with the need to perform administrative tasks hundreds or thousands of times – and the traditional way of doing this is the time-consuming manual approach [12,13].

The team at Argonne National Laboratory have laid out a "scalable cluster management approach" that they used to automate many of the day-to-day administrative tasks for their 314-node Linux cluster [12]. They define specific nodes in their cluster to be "management servers" and have them run common administration support services such as DHCP, NFS, HTTP, TFTP, DNS, and FTP services to the rest of the nodes in the cluster. Their focus was on how many of these management servers are needed, and how to efficiently distribute the services among them. Elements of the architecture they found work well include: making the management servers dedicated (i.e., not available for any other user applications), specialized hardware configuration of the management servers (i.e., more disk, more CPU, adequate RAM, etc.), having a "master" management server from which all commands are

issued, and having remote power control across all nodes in the cluster. They also recommended the following design changes to simplify administrating commodity clusters that contain hundreds of nodes: demand-based software distribution, eliminating single point of service management, and a management-server topology, among others [12].

The Los Alamos National Laboratory team looked at a more novel approach to simplifying cluster administration tasks [13]. They note that administration is typically done via the "system software stack", typically composed of the communication library, resource manager, parallel file system, and system monitor. As the cluster size increases, the role of this system software becomes increasingly more important. They argue that this system software stack can be defined as a small set of network primitives available on all nodes in the cluster. These primitives, essentially a least-common denominator of the system software components, can subsequently be implemented directly in the hardware on each node "greatly reducing the complexity of most system software". As one example, they look at implementing the popular Message Passing Interface (MPI) communication library directly on the system Network Interface Card (NIC) for each node. Based on the results of their experimental case studies, notably not performed on hardware but emulated in their software environment, they promise some real simplification of the current system software stack on these clusters [13]. Note that one major drawback of this new approach is that it must be coupled with future changes made to the Linux kernel that would take advantage of this new hardware layer.

Researchers at the Swiss Institute of Technology decided to tackle the difficult and presently time-consuming problem of installing and maintaining software levels across all the nodes in

a cluster [14]. They propose a new aggressive block-based disk-partitioning imaging approach built on top of their recommended "partition repository." This approach is already in daily use by their current system administrators and has shown a substantial decrease of administrative time in their environment [14].

As the current commodity clusters increase in size, new improved approaches will continue to be needed to grapple with the time-consuming and tedious administrative tasks needed to maintain these cluster. However, in order to even have a cluster, another time-consuming and tedious task, which also requires a great deal of technical skill, must first be conquered: that of cluster deployment.

2.3.2 Cluster Deployment

Since the introduction of the commodity Linux cluster, it has been recognized that the number and variety of skills that were needed to effectively deploy and administer a cluster was one limiting factor to the widespread use of these clusters [1,14]. To deploy a Linux commodity cluster, the installer would normally have to have technical skills in at least five of the following areas:

1. Linux
2. Networking
3. Hardware
4. Storage
5. Installing and configuring the distributed application
6. High availability (if applicable)

It has been noted by experts that the skills requirement is a limiting factor and that the current "effort to adopt, understand, and train computer scientists about clusters and distributed computing has been minimal" [5].

Dozens of papers have discussed issues around manually building a Beowulf cluster [15-21]. However, after building a few manually, researchers maintain that "it is apparent that the process of building and maintaining a cluster needs to be automated" [14]. In fact, "the need for automated installation and configuration tools has been obvious ever since the first appearance of networked clusters of workstations" [20].

2.3.3 The OSCAR and Rocks Projects

OSCAR (Open Source Cluster Application Resources) is a software package that is designed to simplify Linux cluster installation [21-23]. The design goal of OSCAR is to include all the current best-of-class open-source software that would be useful for a cluster in one bundle and ultimately to move toward the standardization of clusters [1]. Today, OSCAR dramatically cuts the time needed to successfully deploy a fully functional HPC Linux cluster as compared to a manual installation where one has to download, install, and configure the software themselves. OSCAR does this by providing a set of tools that are installed on a running Linux system that will serve as the "head node" of the cluster. Once installed, these tools allow the creation of an OSCAR client image that is subsequently used to easily deploy new nodes in the cluster. OSCAR effectively supports multiple distributions of Linux and is supported by a large and active community of contributors and users, including many representing commercial computer vendors [21]. OSCAR also integrates a large number of

diverse software packages, making it a one-stop shop when deploying a fully featured HPC cluster.

Rocks is a similar Linux-cluster-deployment tool built by NPACI (National Partnership for Advanced Computational Infrastructure) [24]. Rocks' major design goal is ease of deployment and as such automates many of the steps that are required in OSCAR. It comes on a CD that can be used to deploy the "front-end" node (i.e., head node) directly without the need to perform this operating-system installation manually. In addition, for the sake of simplicity, it supports only one Linux distribution, namely Red Hat. Rocks also includes many open-source cluster tools, many being the same as OSCAR. Furthermore, it also has the added benefit of supporting deployment on a cluster with heterogeneous hardware [1].

OSCAR, and to a lesser extent Rocks, is very promising and has aided a great deal toward the standardization of commodity cluster deployment. However, as great as both OSCAR and Rocks are, at this point they are still not ideal solutions for all Linux-cluster-deployment scenarios. As can be expected, many of their limitations stem from the design decisions and assumptions that the toolkit makers have made.

The first and most significant assumption is that these packages are targeted to clusters set up in a "standard" HPC clustering architecture. This architecture is depicted in Figure 1.

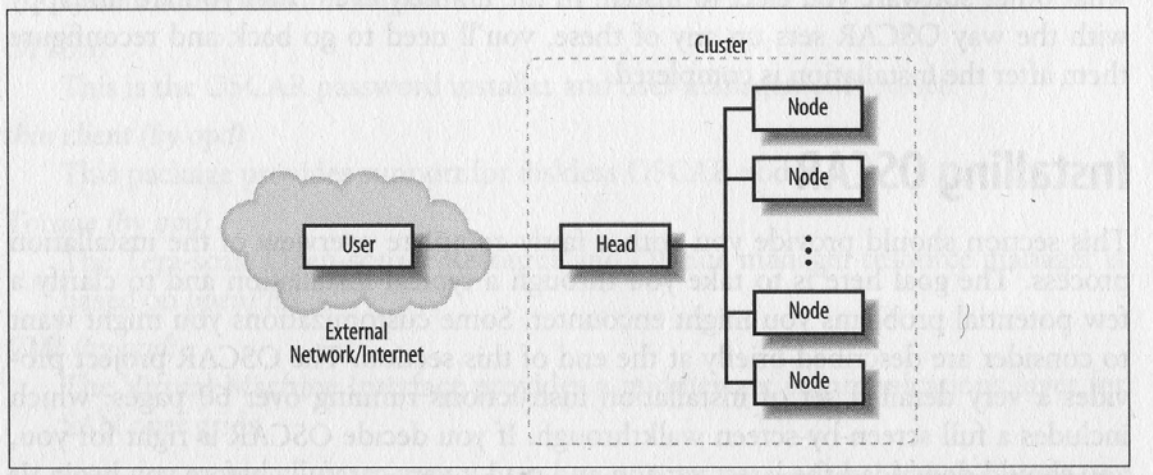

Figure 1: OSCAR and Rocks Cluster Architecture [1, Figure 6-1]

This architecture contains one master or head node and a group of, normally identical, compute nodes. If one is not running in this configuration, for example in the case of a DB2 "shared-nothing" data warehouse, then things become a lot more difficult[1]. See Section 3.2.1, Cluster Topology and Node Types, for details about the typical DB2 shared-nothing cluster topology.

The second issue stems from the fact that each kit is basically a large, complex collection of individual cluster software [1]. This means that a choice must be made during installation: keep deployment simple and install everything from the default package, or take the time and energy to understand what each piece of software does, which packages are actually needed for each environment, and the toolkit software dependencies. In the latter case, this means the skill level of the installer must be relativity high; i.e., the individual must be able to understand what all the packages in the toolkit are, when they are and are not used, etc. This last point should not be trivialized: the large software set is installed and configured in a specific order with dependencies taken care of by the toolkits in the default cases. If an error

[1] It should be noted that with effort one can stretch the toolkits to make them work in other configurations but, in general, this requires detailed technical knowledge of the toolkits.

occurs or a mistake is made during deployment, it can be quite unforgiving [1]. In the former install-everything case, a lot of extra software is often installed and configured that must subsequently be managed on the system. This adds unnecessary complication and potential security holes to the environment, and slows down execution unnecessarily.

Third, since both OSCAR and Rocks are pre-selected collections of open-source software, they do not include any additional proprietary software that would need to be installed. In some cases, these replacement applications are used to replace an inferior application that was included in the kits. For example, many proprietary Linux-cluster-administration toolkits exist, such as the IBM CSM (Cluster Systems Management), that are arguably more powerful than the included open-source cluster-management tool in either toolkit. If they are to be included, these replacement choices normally have to be installed and configured manually afterwards, in addition to having to uninstall the program it is replacing. Commonly the additional software that must be installed afterwards includes the primarily distributed application that the cluster will be running. In the case of a dedicated data warehouse cluster, this is the distributed data warehouse application itself. After successfully using the toolkit, work must still be performed to install and configure this distributed application on all nodes in the clusters manually.

Lastly, although in the last few years both toolkits have made great strides in simplifying the installation, it is still a rather elaborate process [1]. In the case of OSCAR, it is currently an eight-step process, with that must be done in order - after one manually performs the Linux installation on the head node. Although none of these steps is extremely onerous, neither is the process trivial. Rocks is generally easier than OSCAR to use, and does not require a

manual installation of the operating system on the head node or any other of the nodes, but is still far from a trivial process [1].

2.3.4 Other Cluster-Deployment Tools

In addition to OSCAR and Rocks, other cluster-deployment tools have been developed by researchers to help deploy various aspects of their clusters in their particular domain.

Researchers at Laboratoire ID-IMAG have developed their own Perl-based tool called Kadeploy2, which provides automated software installation and reconfiguration mechanisms on all layers of the software stack [20]. This software, currently implemented as a prototype, allows users to concurrently deploy computing environments to the same cluster exactly fitted to their experiment needs, even as their experiment needs change on different nodes [20]. They place a strong emphasis on fast installation performance and support multiple deployment methods with measured execution time performance numbers for each type.

Hardware differences within heterogeneous clusters have historically been quite difficult to deal with. The researchers at the University of California – who developed the NPACI Rocks clustering toolkit itself - have been researching a new approach that they call "Description-based installation," some of which has subsequently been added to their toolkit [26]. This approach will enable the ability to intelligently share configuration across different hardware in a heterogeneous cluster – and to not share configuration across hardware when it should not be shared. Their new XML, Red Hat, Python, and HTTP-based approach does not use the standard replication-of-configuration-files approach, nor does it require building a golden image reference [26]. The approach works on nodes in a cluster in groups, which they term *appliances*. Their approach enables the tool to intelligently deal with hardware

differences between the various appliances. This includes nodes containing different hard-drive types (IDE versus SCSI drives) and even major architecture differences, such as x86 versus Intel™ Itanium™ (IA64) hardware [26]. They go on to validate their approach with over a 100 clusters that have "significant" hardware and configuration differences among them [26].

For the remainder of this paper, one specific approach that was successfully used to simplify the deployment of Linux clusters running a distributed data warehouse application will be discussed.

3.0 A Linux Cluster Data Warehouse Deployment Approach

One common distributed application that commodity Linux clusters can efficiently support is a distributed data warehouse application. A data warehouse is the main repository of an organization's historical data – effectively a corporation's digital memory [27]. Data warehouses contain large amounts of raw information to be used for management's decision-support systems (DSS), which are used to answer important business questions. Data warehousing also enables the business to use automated statistical-analysis techniques - generally termed *data mining* – to discover important new trends and patterns of behaviour in the historical information that previously went unnoticed. This potentially vital intelligence can then be used in a predictive manner for a variety of applications. For example, based on past historic sales trends in the region for males over the age of 50, what would be the estimated sales volume for dentures in Asia in 2007? This gathering and harnessing of historical information has fundamentally changed how important business decisions are made by modern enterprises.

Not surprisingly, these databases routinely grow to very, very large sizes. The largest database warehouses have reached into the hundreds of terabytes (TB) in size, and today it is common to have 5 TB to 20 TB in an organization's data warehouse [28]. To effectively handle this amount of data processing and the complex queries that are typically used for decision-support requests, a distributed database application running on some type of cluster of computers is required.

One of the major leaders in the current data warehouse space, both from a technology and market-share perspective, is IBM DB2 [28].

3.1 The Application: DB2 Data Warehouse

The DB2 database partitioning feature (DPF) uses a massively-parallel-processing (MPP) shared-nothing architecture to allow the database, and the database manager, to be distributed across up to 1000 physical machines - to handle the thousands of terabytes of data that could not possibly be handled by a single machine [29].

A distributed DB2 database can consist of one or more database partitions distributed over the computer cluster. Each database partition is, in essence, a mini-database in its own right. It has responsibility for its own (and only its own) data, logs, data locking, and other essential elements that make up a database [29]. This shared-nothing architecture can be seen in Figure 2.

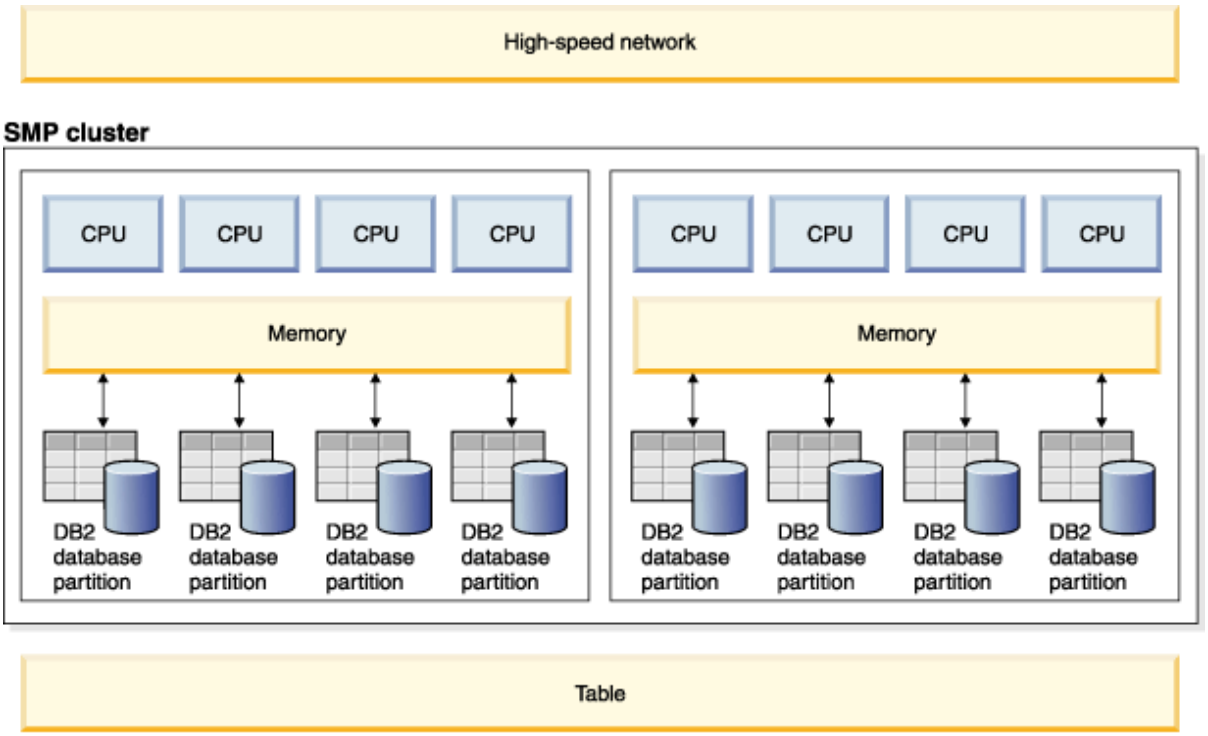

Figure 2: The DB2 MPP Shared-Nothing Architecture [29]

DB2 has supported running in a distributed clustered-computing environment since the mid- 90's [30] and has supported this environment on a Linux Custer since its Version 7.1 release in 2000. In early 2003, IBM released an integrated hardware and software offering called the DB2 Integrated Cluster Environment (DB2 ICE) that offered a pre-installed and configured, price-sensitive, Linux cluster to its customers. Since then, updated versions of DB2 ICE as well as other IBM DB2 Linux-cluster data warehouse offerings – most significantly its Balanced Configuration Unit (BCU) offerings - have been developed that hinge on being able to provide a fast and efficient deployment of a Linux cluster data warehouse.

One of the approaches successfully used in hundreds of cases to make this fast and simple deployment a reality, was the DB2 ICE Installation Server (DB2ISERVER) approach which will be detailed in the remainder of this paper.

3.2 Requirements and Design Decisions

The five major requirements[2] that drove the design of this approach are the following:

Requirement #1: Laydown and configure all software that is needed during one install – no further steps should be required post-install to successfully start the DB2 application on all nodes.

One of the greatest frustrations of the manual process of setting up a Linux cluster - as well as for some of the automated cluster kits - is the number of steps that one must successfully complete in transforming the cluster from a group of networked hardware to a working system. It is not an exaggeration to put this number of steps in the hundreds, and for everything to work, each individual step must be performed perfectly. Making this more difficult and error prone adds to the many dependencies that exist between these steps. One mistake during networking setup on one machine will prevent one from being able to start the distributed database instance many steps later. The installation tool should automate all required steps including software installations, networking configuration, user configuration, etc., and manage the dependencies between them automatically. After the installation is complete, the installer should immediately be able to start the distributed database manger and begin creating the database.

[2] During development of the tools, these were affectionately referred to as "the big 5."

Requirement #2: Required skill level of installer should be minimal.

The individual installing a Linux cluster data warehouse has historically had to be very skilled, not only with cluster-related skills described earlier – hardware, networking, storage, clustering tools – but in this case, also Linux, Data Warehousing, and DB2 skills. This has been a major inhibitor in the speed with which these clusters can be deployed and ultimately to the number of clusters being deployed. While it is true, to an extent, that one will ultimately have to understand many of these areas to properly administer the cluster, the same demands should not be required from the individual performing the installation. It should be possible for a person who has no experience in clusters and DB2, and very little experience using Linux, to successfully deploy a cluster in a short period of time (see Requirement #5) using this tool.

Requirement #3: The tool must enable successful deployment on a prescribed set of bare hardware with no need for any initially installed operating system, software, additional drivers, or software patches to complete.

One of the frustrating consequences of the rapid rise and flexibility of the Linux operating system is that in many instances Linux drivers or modules need to be updated to enable successful deployment and/or efficient and secure operation of the distributed application. This stage has historically been the stumbling area for many deployments; in some cases one missing or obsolete hardware driver leaves the novice installer looking at a hung Linux install with no idea how to continue. This is one of the major reasons the decision was made to use a fully tested "master image" (also commonly referred to as a "golden image") instead of using the install-from-scratch approach. These master images will contain all required drivers and patches and be pre-tested on the supported hardware.

Requirement #4: The tool should deploy a pristine working system every time. A simple method of verifying a successful installation must be included.

With so many configuration options – colloquially referred to as "knobs" - available at the operating system and application level, it was decided that to ensure pristine working system every time, a "master image" approach should be taken. In addition, to minimize any chance of expensive and hard to isolate mis-configurations, the tool should take responsibility for all aspects of the deployment; from the complete disk partitioning, to the Linux operating system installation, to the system-network setup, to the DB2 installation. If there is any type of error during the installation, then the installer would just re-install that specific node using the tool, without a need to diagnose and try to recover from this error.

As well, as a consequence of Requirement #2, it is not sufficient to rely on the installer to have the skill set to ensure that the system is working properly after deployment. Not only is it not always clear what a "properly" installed system actually means to a non-data warehouse expert, but also, depending on the method used, this can be a highly time-consuming process. For this reason, a popular Linux open-source host and network-monitoring application named Nagios® was integrated into the tool to enable easy visual verification of a working cluster and required DB2 data warehouse services across the cluster post-installation. See Section 3.4, Software Details, for more information about the software stack.

Requirement #5: Rapid Deployment

The deployment of even a moderate-sized Linux cluster can be a very time-consuming task. Deployment durations of weeks are common in this area. One of the major benefits of automating the deployment is to greatly shorten this time of deployment. During design of this approach and the DB2ISERVER tool every attempt was made to make this as rapid as possible. The rule of thumb target was for the tool to shorten the deployment time by an order of magnitude – weeks to days and days to hours – when compared to performing a manual deployment. Thus, if a manual deployment of the cluster normally takes 3 days, using the tool an installer – regardless of skill level – should be able to deploy the cluster in roughly 3 hours. In almost all cases, this requirement was successfully met.

3.2.1 Cluster Topology and Node Types

In addition to the five major requirements that drove the design of this approach, another important element is the defining of the cluster topology.

A best-practice DB2 shared-nothing data warehouse has some notable differences from a tradition HPC cluster setup [31]. The compute nodes, called "data nodes", each fully own a complete portion of the database. In addition to these data nodes, three other node types exist: System nodes, Administration nodes, and Load nodes. Each of these four types of nodes is configured differently.

When compared with the traditional HPC head node, this functionality is spread across the System and Administration nodes in a DB2 data warehouse. The Administration nodes also host important DB2-specific services, such as the DB2 coordinator functionality that accepts and processes remote requests to the warehouse from the external application(s). Again, this

means the topology used with a DB2 shared-nothing data warehouse is in many ways different from a traditional HPC setup. An example of a typical DB2 Data warehouse cluster configuration can be found in Figure 3.

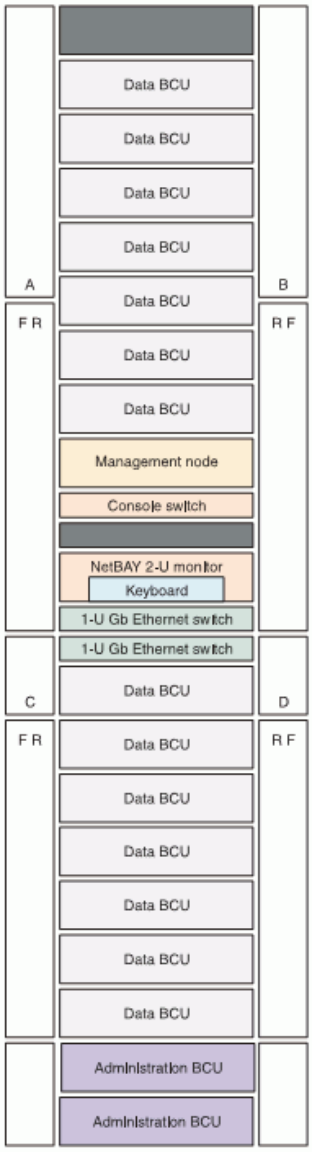

Figure 3: Typical DB2 Shared-nothing cluster topology [25, fig. 24]

The DB2ISERVER tools must allow the installer to define each machine in the Linux cluster as a specific type of node as understood by DB2 data warehouse best practices described above. This in turn defines how each node will be exactly configured by the tool during deployment. This will allow the creation of flexible data warehouse topologies that make the most sense for each data warehouse cluster being deployed. Generally, these decisions are made by the data warehouse design team and communicated to the installer.

Details of each node type that is supported are [31]:

System Node: is analogous in many ways to a traditional management or head node. In a HPC cluster, it is responsible for managing the overall cluster and to be available as the system management, installation, and configuration arm of the cluster. It can also be used as the failover node for the first administration server in a high-availability setup. Note that although very similar to a traditional management node, the system does not perform scheduling nor does it hold the coordinating management information for DB2 that resides on the first administration node.

Administration Node: is responsible for the DB2 coordinator functionality, that is to say, the node by which the external users will connect to the cluster. The first administration node defined will also contain the DB2 system catalogues (the important meta-data tables that are created and maintained for each database); as well, it runs the NFS service needed to export the distributed directories as needed by the DB2 DPF feature. It is recommended to have the administration node equipped with more memory then the rest of the nodes especially in the case of a large number of concurrent users connecting to the cluster.

Data nodes: are the workhorses of the DB2 Linux cluster. They each operate and effectively own one subset of the overall data warehouse cluster. Users normally do not connect to those nodes, and they need no external connectivity whatsoever. In many cases, for simplicity in management and security, it is best not to connect those systems to the existing external network.

Load nodes: are optional nodes that are responsible for processing all of the extract, transformation, and load (ETL) work for the cluster. It is a recommended best practice to isolate ETL on a node in the cluster to be used exclusively for ETL work, to ease the burden of workload management especially when a dedicated ETL tool is being used.

An example nodes definition and resulting cluster topology for a six-node cluster is shown in Figure 4.

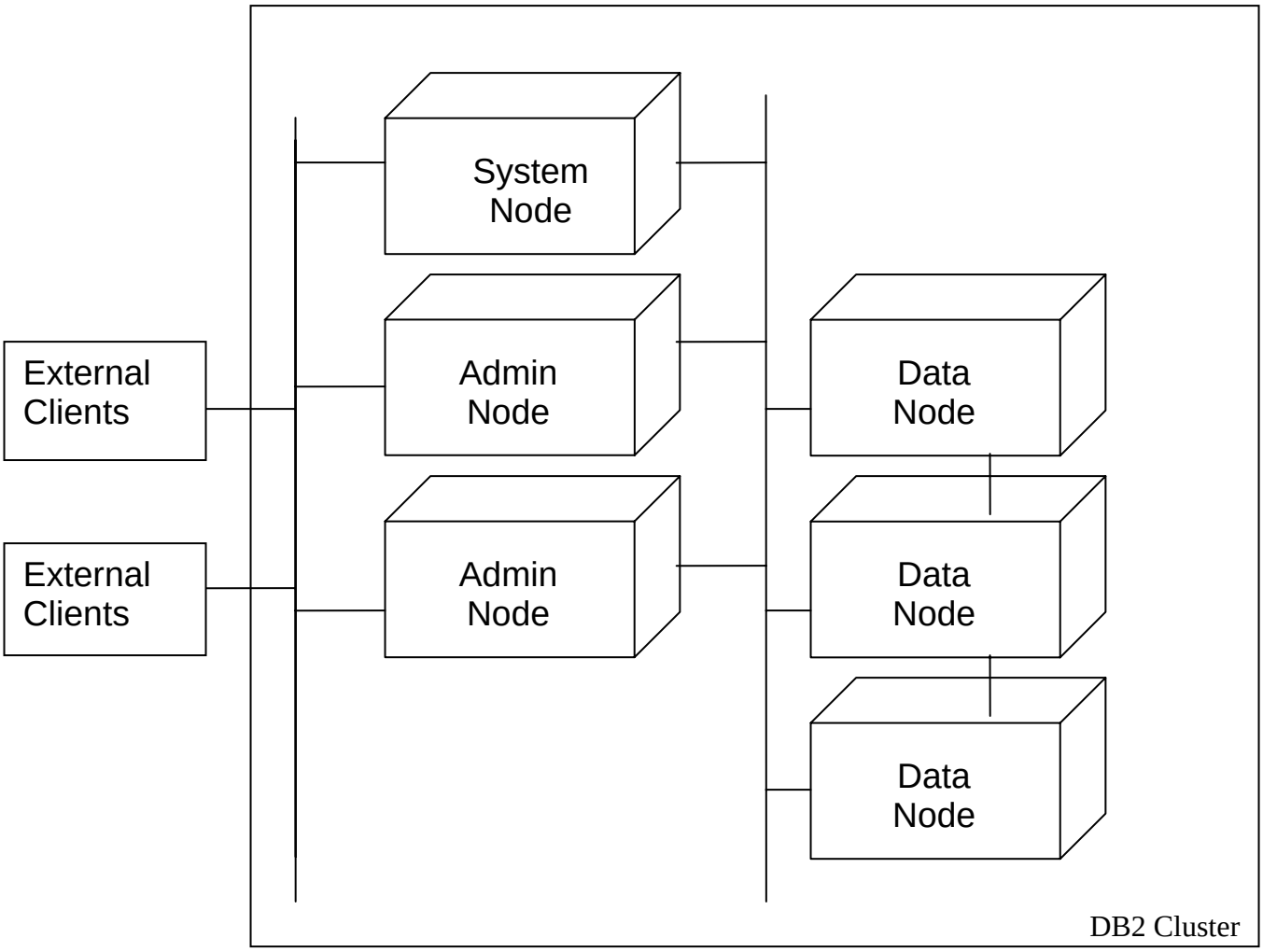

Figure 4: A Six-Node DB2 Data Warehouse Cluster Definition and Topology

3.3 Hardware Details

The DB2ISERVER tool is built to officially support a pre-determined set of hardware as prescribed by the DB2 Data Warehouse best-practice team for Linux [31]. Among other benefits, strictly defining support to an official set of pre-determined hardware allows users of the tool kit to rapidly deploy a DB2 Linux data warehouse cluster into a pre-tested environment, where they are assured of rigorous pre-testing of the deployment performed in-house before it being performed live on the customer site. Pre-identification of supported hardware, in conjecture with the general pre-built master-image approach applied by the tool kit, assures the number of failed field deployments will be close to zero.

The specific hardware configurations supported have changed very frequently – and will continue to do so – based on the current market conditions and the extensive experience of the recommending IBM team [31].

The number of types of servers that are currently supported is more than 10 and will continue to grow. The servers have all been IBM System Servers running on Intel x86 and AMD x86_64 architectures. The server form factors that were specifically supported were:

- Rack servers with SCSI internal disks,
- Rack servers with external disk storage,
- Blade servers with internal SCSI disk storage, and
- Blade servers with external disk storage.

As part of the deployment, the DB2ISERVER tool will perform the disk partitioning of the first internal hard drive. This first hard drive will contain the operating system, as well as all installed applications including DB2. A consequence of this automatic partitioning is that the size of this hard drive becomes a minimum system requirement for each node in the cluster. Currently the requirement is for this first hard drive to be at least 73[3] Gigabytes in size before disk formatting.

The networking type supported by DB2ISERVER is either Ethernet, typically gigabit variety although all varieties are supported, or Infiniband (IB) high-speed interconnect. Note that the type of each network must be specified in the xcluster.cfg configuration file at time of deployment and this must correspond to the hardware found on the machines. With respect to network security, it is generally recommended to have a company's data warehouse placed in a secure data canter with no Internet conductivity directly to any of the nodes. Typically only the cluster administrator and the DB2 client applications will have the ability to access the cluster both directly from the corporate intranet via the System node and Administration node, respectively.

There is no minimum requirement on the amount of RAM memory on the nodes imposed by the DB2ISERVER tool, although DB2 itself does impose a 512 MB minimum to run.

[3] 73 GB is a common SCSI hard drive size

Lastly, it should be noted that although a specific set of hardware is documented to be officially supported, the DB2ISERVER tool has successfully been used on a host of other x86_64 hardware of varying configurations, including those from hardware companies other then IBM. As described above, the primary requirement is a minimum prerequisite disk drive size on the first hard drive of each of the node in the cluster. However, if non-official hardware is used, there is no guarantee that all required hardware drivers and security fixes are included in the kit, which increases the likelihood of some type of failure or problem during or after deployment.

3.3.1 Additional Hardware Requirements

In order to develop and maintain the DB2ISERVER tool, two important machines, or most commonly virtual machines, are required by this approach. Note that these two machines are **not** per cluster – it is sufficient to have two machines (or virtual machines) for an unlimited number of clusters being deployed.

These machines are:

- **Master-Image Machine:** This machine will hold the master image. Having the master image pre-installed on a dedicated machine, or virtual machine, greatly simplifies any future modifications that must be made. Changes are applied directly to the master-image machine; this machine is subsequently copied into a new image which then replaces the older image on the Installation Server. See Section 4.1, Creating the Master Image, for details about this process.

- **Installation Server:** This machine is configured as a standard Linux installation server[4] as well as holding the master image and the system-configuration code that is used to image and configure each node in the cluster. The Installation Server can either be automatically generated on a deployed system node or can be a different machine external to the cluster – such as a native Linux notebook or commonly a VMware® virtual machine. To use the system node as an Installation Server, it must first be created itself through the use of an external Installation Server. An external Installation Server is made available and distributed as a VMware virtual machine that can be run on any existing VMware Workstation 5 system**.** Installers normally receive and start using the DB2ISERVER tool by downloading this pre-configured Installation Server VMware virtual machine. See Section 4.3, Details of the Installation Server, for more information about the Installation Server.

3.4 Software Details

As a core requirement (requirement #1), all software must be laid down on the cluster during an integrated installation; with all software installation and configuration performed in one automated installation with no manual intervention needed.

The current software architecture used by the DB2ISERVER tool and deployed onto the Linux clusters is shown in Figure 5 below.

[4] The technology used varies between Linux distributions. Currently, DB2ISERVER supports Novell's SUSE Enterprise Linux (SLES) distribution and hence uses its AutoYast Linux Installation server.

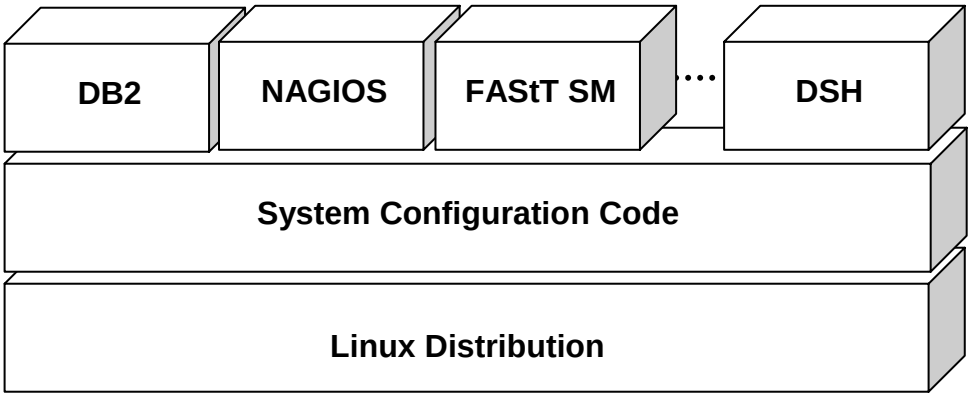

Figure 5: DB2ISERVER Software Architecture

At the base of the software architecture is the Linux operating system. The currently supported Linux distribution that the tool supports is Novel SUSE Enterprise Server versions 9 and 10. Other than the time and resources necessary to produce and maintain a new master image, there is no technical reason why other Linux distributions cannot also be supported by this approach.

On top of the Linux distribution setup sits a set of DB2ISERVER system-configuration code that automatically customizes each specific node for their function in the cluster, performing configurations such as the setup of hostnames, TCP/IP addresses, necessary gateway functions, time setup, as well as setting up secure user access across the clusters. This system-configuration layer can be used any time afterwards to change the setup, if necessary, post-deployment; for example, when moving a cluster between data centers.

At the top of the architecture, is a minimal set of application blocks, each with a specific function that will vary as time progresses and the needs of the cluster changes. At this time, the software includes the DB2 distributed database application itself, as well as the Nagios system-management application, IBM FAStT Storage Manager, ServeRAID manager, db2top, LSI® Megaraid Storage Manager, among others. A simple distributed shell (dsh) is also included in this layer.

Nagios is a popular Linux open-source system and network-monitoring application [32]. Nagios allows monitoring of the nodes in a cluster as well as services (distributed and otherwise) running on these nodes and will send an automatic alert if service or host problems occur. Contact notifications can be sent via e-mail, pager, SMS, or any user-defined method. One of the strengths of Nagios is its high level of customization: it provides a plug-in design that allows development of custom checks by using standard programming languages such as Perl, Python, and C++, among others. System Administrators can monitor the hosts and services and manage Nagios through an optional Web interface. Nagios also has the ability to implement redundant monitoring hosts.

A Linux cluster deployed using the DB2ISERVER tool will have the Nagios monitoring host automatically installed and configured on the System node and is used by the one deploying to easily verify successful cluster installation.

The complete software stack is provided as a single image that contains all necessary services and software components described above. The specific building blocks are then configured, enabled or disabled accordingly, during deployment of each node in the cluster. Since the software stack will be installed from a master image, the exact contents of this stack can and does change over time. To make an update to a current software version, or to add some new software to the stack, all that is required is a one-time installation or update to the master-image machine. This update is performed once by the DB2ISERVER maintenance team, and distributed as an updated Installation Server virtual machine to all users of the kit. See Section 4.1, Creating the Master Image, for additional details about the software version used by the current version of the tool.

4.0 The DB2ISERVER Installation Tool

This chapter will detail specifics about the DB2ISERVER installation tool developed at IBM to ease deployment of a DB2 data warehouse Linux cluster.

4.1 Creating the master image

The ultimate target of the deployment is the creation of a fully running and properly configured DB2 data warehouse Linux cluster. The creation of a master image is the first step towards this goal. The master image used by the DB2ISERVER tool has been designed with the following criteria:

- Ensure the broadest possible support for boot devices in the Linux kernel including the normal Adaptec SCSI controller used by IBM, the IPS driver for the ServeRAID adapter, and support for BusLogic® and QLogic® network drivers.

- All needed packages are on one single image that is used by all node types to simplify the setup and keep the distributed Installation servers as small as possible.

- All components and drivers are updated to the latest stable, tested and validated level as recommended by the DB2 for Linux design team.

The disk layout created for each node in the cluster will be identical to each other as well as to the disk layout on the Master-Image Machine. It is detailed in Table 1.

Disk/Partition	Size (MB)	Mount point	Purpose
All nodes			
/dev/sda1[5]	100	/boot	Kernel and boot loader partition
/dev/sda2	5000	/ (RESCUE)	Small rescue system for each node
/dev/sda3	20000	/ (PROD)	Production OS core
/dev/sda5	16000	Swap	Swap space 2x default memory
/dev/sda6	10000	/var	Independent /var to preserve the OS stability
/dev/sda7	10000	/tmp	Independent /tmp to preserve the OS stability
/dev/sda8	Rest of sda	/db2data	DB2 home directory on this machine
Data node			
/dev/sdb1	360000	/db2data1	RAID5 array plus one hot spare
/dev/sdc1	360000	/db2data2	
/dev/sdd1	360000	/db2data3	
/dev/sde1	360000	/db2data4	

Table 1: Current Disk Partitioning (for all nodes in the cluster)

The master image copy that is used by the DB2ISERVER Installation server is built by taking a very simple full tar backup of a Master-Image system. tar is a simple yet very powerful UNIX™ utility that allows a file 'archive' copy to be created of any set of directories on a system – even the entire file system itself. The following tar command is run on the Master-Image machine to create the master image copy:

[5] Note that in Table 1, the term /dev/sda is generically used to refer to all disk device types.

tar –cpzvf image001.tgz / --exclude=proc/kcore

This single "golden" image file is then used as the base image for nodes in the cluster, regardless of the node type. At this time, this file contains all the software installed during a default SUSE Enterprise Linux 9 or 10 install in addition to the files listed in Table 2.

Package	Source/Owner	Current Version
IBM DB2 UDB ESE V9.1	IBM	DB2 V9.1 (Fixpack 1)
ServeRAID Manager	IBM	8.12
ServeRAID agent	IBM	RaidMan-8.12
IB driver	Voltaire	ibhost-v3.4.5_12
IB driver	Mellonox	1.8.0
Distributed Shell (dsh)	IBM CSM	csm.dsh-1.6.1
Apache	Apache Foundation	apache-1.3.29-71.15
Nagios	http://www.nagios.org	Nagios-1.2-73.1
FAStT Storage Manager	IBM	9.1 V15
IBM® SDK for Java™	IBM	IBM SDK for Java 2-1.5.1
SanSurfer Pro 2.0.30 build 58	QLogic	2.0.30 build 58
AutoYast 2 (SUSE ONLY)	Novell	autoyast2-2.9.53-0.2
Q Logic Drivers	QLogic	qla2xxx 8.01.01 qla2300 8.01.01 qla2xxx_conf 8.01.01
Open IPMI	http://ipmitool.sourceforge.net/	OpenIPMI-1.3.11-0.2
IPMI Scripts	IBM	1.0

Table 2: Current Software versions in the Master Image

In addition to installing the additional software to satisfy Requirement #1, other changes were made to the master image as required to support Requirement #3 – namely, the automatic changes applied to any system configuration or system parameter to ensure an optimal running and secure DB2 data warehouse Linux cluster. Table 3 below lists some current pre-install changes made directly to the Master-Image machine – and hence to the master image – to fulfill this requirement.

File Changed	Change
SLES OS files	Applied all current SLES maintenance and security fixes to the system.
~db2inst1/.ssh/id_rsa ~root/.ssh/id_rsa /sbin/.ssh/id_rsa	Generate ssh keys for these two users and the system daemon
/usr/RaidMan/RaidAgnt.pps	Ensure this line exists: agent.enable.security=false
/etc/sysctl.conf	Ensure these lines exist: vm.swappiness = 0 vm.dirty_ratio = 10 vm.dirty_background_ratio = 5
Nagios configuration files	Many changes were made to support DB2 out of the box.
/etc/sysconfig/kernel	Ensure this line exists: INITRD_MODULES = "mptbase mptscsih ips qla2xxx_conf qla2xxx qla2300"
/etc/modprobe.conf.local	Change the line "options qla2xxx ql2xfailover=0 configrequired=0" to: "options qla2xxx ql2xfailover=1 ConfigRequired=0"

Table 3: Pre-install related changes made to the Master Image

4.2 Cluster-deployment

Following the physical assembly of the hardware, four simple deployment steps - described below - are used to rollout a Linux cluster using the DB2ISERVER tool.

Before installation has begun, a quick hardware check should be performed. It is recommended to quickly check over and write down the current BIOS levels of the machines and verify that no critical fixes are missing. During this quick check, all the RAID volumes must be created and a short power-up test is recommended.

The actual installation works as follows:

1.) The first step involves ensuring the proper services are started on the Installation Server. To do so, run the script called "**start.sh"** found in the /tftpboot directory off the Installation Server.

2.) The second step is the gathering and validation of the existing hardware, cluster topology, and network layouts. To do so, the primary MAC addresses of the Ethernet network adapter is located and used to uniquely identify each node in the cluster (it is highly recommended to have this information written on the front of a large cluster anyway). Subsequently, each node is defined as a system, administration, data, or load node, depending on its intended function. The list is entered into the DB2ISERVER configuration file – called the xcluster.cfg file - with the selected parameters for the network and function inside the cluster.

3.) Next, the nodes can be powered on. When booted, they will automatically connect to the installation server and start their installation (see below for details). For logical flow, it is recommended to begin installation of the system node first, followed by the administration nodes, before beginning all the data and load-node installations. This will ensure that the nodes will be complete and bring themselves up in a proper fashion (i.e., with the system and administration nodes powered up first).

4.) During installation, the operating system, DB2, and all higher-level building blocks are installed, while all provided information is configured, and the setup is completed. A check through the Nagios console on the system node (available from any Web browser) should show all the correctly installed nodes up and running properly. In the case of wrong TCP/IP addresses - or, in fact, errors of any kind - each node can be re-installed as many times as necessary until the setup is satisfactory.

Analyzing steps 3 and 4 from above in greater detail, each node goes through the following seven steps:

1. The node is booting cold into a network boot (depending on the BIOS of the system, this happens automatically or needs to be initiated manually *once* usually by pressing F12 during bootup).

2. The node picks up a DHCP network boot address from the Installation Server together with the direction for the network boot to load the boot file "pxelinux.0" - which it does via TFTP.

3. The PXE boot strap initiates the download from the PXE configuration file, which itself is a pointer to the Linux kernel, INITRD file, and the appended boot parameter.

4. The Linux kernel and the INITRD file is transferred via TFTP to the machine and the boot strap initiates the installation based on the pre-set installation parameters without any user interaction.

5. As part of the installation of the rescue system, the primary (i.e., First) hard drive is partitioned as described in Table 1.

6. After the core installation of the rescue system, the production system master image is copied to the proper partition. In addition, the xcluster.cfg configuration file is transferred from the Installation server to each node in the cluster to be used during configuration[6].

7. All system configuration is performed based on the information provided in the xcluster.cfg file. Section 4.5, Details of the System Configuration, looks at this stage in more detail.

4.3 Details of the Installation Server

The installation server uses three important services to deploy each node in the cluster.

All these services can be configured for permanent operation with the Linux command:

chkconfig <service name> on

[6] This configuration file– as well as the complete installation logs - can be found in the /root/.db2ice directory post install.

This command will start the services at the next reboot automatically. The following command will turn off the service permanently:

chkconfig <service name> off

Note when using the Installation Server VMware produced by the author, all services can be enabled by running the script "**start.sh"** found in the /tftpboot directory.

These three important services are:

Service #1: DHCP server

The DHCP daemon provides the initial boot addresses during the installation cycle. The provisioning of the DHCP addresses is important, especially if the cluster is installed with its own private network. The DCHP address also comes with the critical information for the BOOTP/PXE process-related information (where to get the pxelinux.0 file). The service can be provided from an outside resource. The configuration for the DHCPD server is located in the file /etc/dhcpd.conf.

The service uses the range 192.168.254.1 to 192.168.254.250 for the setup process. The system node should have the default boot address 192.168.254.254. Of course, those addresses can be easily changed in the setup scripts. All DHCP addresses get logged in the file /var/log/messages for debugging purposes. The DHCPD daemon is not needed for normal cluster operation and can be switched off at this point.

No other DHCP servers should be running in the environment because this will conflict with the installation server's DHCP server[7].

Service #2: TFTP server

The TFTP server is the most rudimentary file server provider. It is needed for the transfer of the boot image. Per definition, this server can be created as a standalone service or as part of the XINETD service. The TFTP server resides in the directory /tftpboot, which is also the default share for the NFS server-based installation. The critical files in the directory are:

1. pxelinux.0 (pxelinux boot file)
2. pxelinux.cfg/default (pxelinux configuration file)
3. *.profile files

Service #3: NFS server

The NFS server is used to provide access to the software on the installation server and does this by sharing the /tftpboot directory. It contains the production master image, the actual installed software, as well as the System-Configuration perl scripts. This sharing is only needed during installation and upgrades, and can be disabled afterwards.

[7] This is particularly easy to miss when running the installation server inside VMware, which might be running its own DHCP server available to each virtual machine. In this case, the VMWare DHCP server must be turned off.

4.4 Details of the Configuration File

The sole input that is required from the installer to deploy the Linux cluster is to fill in one configuration file: namely the xcluster.cfg file. This file can be found in the /tftpboot directory. This important file contains a list of all the machines that will make up the cluster – each uniquely identified by the MAC address of its first Ethernet adapter (eth0). In addition to identifying the machines in the cluster, it allows specification of the configuration for each machine in four major areas:

- **Cluster Node Type**: Indicates whether the machine should be configured as a System, Administration, Data, or Load node. See Section 3.2.1, Cluster Node Types.

- **Network Information:** Indicates the requested network settings for each node in the cluster. This means the hostname, network adapter type, TCP/IP addresses, net masks, and default gateway will be identified for one or two networks on the cluster. The system configuration code will automatically configure these network settings on each node during deployment.

  While the network can be configured in any way chosen by the person performing the installation, it is normally recommended to configure the cluster as a standalone cluster with two networks – one an Ethernet management network (e.g., 192.168.x.y), and one dedicated high-speed network for DB2 Fast Communication Manager (e.g., 172.16.x.y). The connection to the corporate user network is established – usually only on selected machines in the cluster - through additional available network ports.

- **DB2 Configuration:** Indicates the mandatory user names and new passwords for the three default DB2 users (instance owner, DAS owner, and fenced user ID) as well as the DB2 partition numbers for each node. Note, the user names must match the user names that exist in the installation image being used.

- **Nagios Configuration:** If Nagios is used in the Linux cluster, the mandatory Nagios settings needed to deploy it must be indicated. Currently, this consists of only the Nagios administrator user name, password, and a contact e-mail address. Using this information, the post-imaging System Configuration code will automatically configure Nagios to monitor all machines in the cluster on the system node (if present in the xcluster.cfg file, of course; if no system node is present, no configuration will occur). The Nagios administration console can then be run on any Web browser on this System node.

Note that the text file format has been designed to be simple and flexible and is structured to be extendable for future needs. The xcluster.cfg file specifications, as well as a sample completed xcluster.cfg file, can be found in Appendix A of this paper.

4.5 Details of the System Configuration

The System Configuration program is a Perl program that will run automatically after the master image has been laid down and un-tarred on each node. Its purpose is to perform all configurations needed for each node based on the node type. The system configuration code is automatically launched after successful imaging of the production operating system on each node. After the configuration code has completed successfully, the node is rebooted to

apply all the new system settings and verify proper hard-disk booting of the newly configured system. This reboot ends the installation for each node.

Currently the following elements are configured on each node on the cluster:

- Set IP address for first network
- Set IP address for second network (optional)
- Set hostname for first network
- Set hostname for second network (optional)
- Set default network gateway
- Set up the name resolution for all machines in the cluster
- Set up the machine timezone
- Set up root, DB2, and other required users and their passwords on the system
- Set up ssh no password logon between machines
- Set required environments variables for root (.bashrc)
- Set up dsh configuration for root

In addition, the following elements are configured based on the type of node that is being deployed:

SYSTEM NODES

- Enable the proper services for a SYSTEM NODE (rsh, nfs, nfslock, nfsserver, xinetd, rlogin, apache , nagios).
- Set up the .rhosts file in DB2 instance owner
- Configure the Nagios application to monitor the cluster

FIRST ADMINISTRATOR NODE (First node of type ADM in the file)

- Set up the db2nodes.cfg file in instance home
- Set up the .rhosts file in instance owner
- Set up the exports file to export instance home
- Set up the DB2 registry values for admin tools (default.reg)
- Enable the proper services for a FIRST ADMINISTRATOR NODE (rsh, nfs, nfslock, nfsserver, xinetd, rlogin).
- Mask out passwords from xcluster.cfg file

NON-FIRST ADMINISTRATOR NODES

- Mount the instance directory in the fstab file
- Enable the proper services for a NON-FIRST ADMINISTRATOR NODE (rsh, xinetd, rlogin)
- Mask out passwords from xcluster.cfg file

DATA & LOAD NODES

- Mount the instance directory in the fstab file
- Enable the proper services for a DATA NODE or a LOAD NODE (rsh, xinetd, rlogin)
- Mask out passwords from xcluster.cfg file

5.0 Lessons Learned and Future Work

5.1 Lessons learned from the field

The DB2ISERVER tool has been used successfully in many engagements around the world by different IBM personnel to deploy Linux clusters running DB2 varying in size from 2 to more then 64 nodes. Many lessons were learned during these engagements.

The most important is that the approach did end up fulfilling almost all of its objectives. Requirements #1, #3 and #4 were all met by the nature of the design decisions made during the development of the tool and subsequently verified during the tool's usage in the field. The tool installed and configured a pristine working system, in one install, onto the supported bare hardware quite easily. The Nagios visual method of verifying a successful installation was well received by both those deploying and users of the cluster alike, with Nagios's effective historical logging and visual report-generating features viewed as an added bonus by cluster users.

Using the DB2ISERVER tools has drastically reduced the time needed to deploy a cluster: generally by an order of magnitude, from days to hours. A cluster installation that normally took 2 or 3 days can now be performed in 2 or 3 hours. In addition, once a successful install has completed, the imaging approach means that the installed system has worked properly every time. There are no further steps necessary in order to start creating the database on the data warehouse cluster. The time to install each node, or a group of nodes up to eight, is anywhere from 5 to 15 minutes, depending primarily on the speed of the network and the

speed of the local disk on each system. This was all deemed to easily meet and exceed the rapid deployment requirement (i.e. requirement #5).

The results with respect to Requirement #2, namely the required skill level of the installer, were mixed. In general, as per the stated objective, it was shown over and over again that it is possible to have a person with minimal cluster, Linux, and DB2 skills successfully perform the cluster-deployment. In one case, an internship student who had only been using Linux for two weeks, had minimal UNIX, DB2, and TCPIP networking experience, and had never deployed a cluster before, successfully performed a 16-node Cluster-deployment using the tool. However, the one important caveat to this is that it was found the installer **must** be trained on proper usage of the DB2ISERVER tool itself. It was found that a two-hour training session on how the approach works and using the tools, most importantly how to start the installation server and properly fill in the configuration file, was critical to the success of the deployment. Without this training, the probability of failure by first-time installers, at some stage in the deployment, is very high.

What were the problems that the installers ran into? By far the most common cause of failure was networking-related issues. This is perhaps not surprising since networking is generally regarded as one of the most difficult aspects of clustering technology. In addition to the always present issues of knowing how to choose proper networking values and physically wiring all the networks properly, there was the issue of proper network-switch configuration. Depending on how it was configured out of the factory, one might need to update or remove some preconfigured VLANs (Virtual Local Area Networks) and might need to configure the switch to prevent it from suppressing the DHCP server on the

network. Since, in general, these steps are very switch specific, and the configuration from the factory tends to be different in different locations of the world, it was found that this road block was difficult to overcome via automation or even documentation. As well, if the installation server being used is a VMWare virtual image (which is how the installation server is deployed at IBM), those new to VMware commonly ran into problems with the VMWare-specific networking between the guest virtual machine and the host machine and other machines in the cluster.

In addition to the networking tool-usage wrinkle above, there were some negative aspects of the approach itself. The single master image helped keep things simple – mostly for the one developing the master image and other aspects of tool development – but in some cases, the "single master image" approach limited the flexibility of what can be performed on different nodes. If anything needed to be added to the master image that was only useful for one of the node types – for example, a cluster-administration tool for the system node - it would also be placed on all the other nodes in the cluster, and although it does not have to be automatically configured (since the configuration code is node aware), it would be present on the data, administration, or load nodes as well. This type of master image "bloating" occurred as time progressed and produced a type of resistance to node-specific additions to the kit.

Lastly, some users of the tool commented that the collection of the first MAC addresses from each node in the cluster and having to enter them into the configuration file was a time-consuming and error-prone step.

5.2 Future work

Overall, the DB2ISERVER approach and the tool itself were found to be effective and useful. One consequence of this is that new-feature requests have been solicited by users of the tool. The most requested features are those that will further automate current time-consuming tasks:

- Integrate automatic High-Availability setup and configuration into the tool.
- Integrate automatic external-storage provisioning into the tool.
- Automate the creation and configuration of the production data warehouse database, table spaces, and other higher-level data objects into the tool itself.
- Automate the configuration of a cluster Network Time Protocol (NTP) server.

Work has already started on the next generation of the DB2ISERVER tool. This new tool will fix some of the limitations of the current tool described in Section 5.1, Lessons Learned from the Field.

The largest modification to the current approach that will be found in the new tool is that it will no longer image each server using the same master image: instead, only the system node will be imaged via a master image. This master image, and hence the system node, will be a preconfigured install server that can be used to install (not image) the remaining nodes of the cluster. IBM's Cluster Systems Management (CSM), a cluster management tool for IBM System p™ and System x™ clusters, will be integrated into the system server and used to quickly deploy all the other nodes in the cluster. A single script will be automatically built at the system node during the deployment time that will initiate the installation on the

remaining nodes. This approach will give the flexibility to deploy each type of node differently, based on mixing and matching the available software on the system node, and still preserve the simplicity of setup and the little clustering skill required, as the system node will be setup after its deployment.

Some of the other smaller, new features, of the new tool will include:

- The ability to deal with more than two network adapters on each node
- The automatic setup of the Baseboard Management Controller (BMC) network on each BMC-enabled cluster. BMC is a specialized embedded chip that is found on most IBM System x servers that supports the Intelligent Platform Management Interface (IPMI) architecture used for system management.
- The automatic setup of a NTP server on the management node to automatically keep the system time consistent across the cluster
- Simplification of how the network information is entered into the xcluster.cfg file to minimize chance of error
- Support of a larger set of hardware, including the newer dual and quad-core Intel™ and AMD™ based servers. The DB2ISERVER tool will automatically configure an optimal number of DB2 database partitions on each data node based on the number of cores that are present on the server.

6.0 Conclusion

As the modern work horse, used to efficiently and inexpensively run software applications requiring intensive computing power, commodity Linux clusters have come a long way in the last decade. Linux clusters are now being deployed in ever-increasing numbers all over the globe. The ease, speed, and simplicity of deployment have become critically important issues for organizations deploying them in large numbers. There is a need for automated tools to speed deployment, increase consistency, and decrease the skills needed to be able to effectively deploy a dedicated Linux cluster.

After surveying the current research in the area of Linux-cluster-deployment, it was concluded that the current state of the art in this area has managed to automate, and to a lesser extent standardize, much of the deployment process. However, there is still work to be done. These tools and approaches still require the one deploying to posses a minimum skill set; and, especially when deploying a custom distributed application, require a great deal of work to be performed by the installer.

The specific tool that is the focus of this paper, DB2ISERVER, was developed by IBM to help make deployment of Linux clusters running their distributed DB2 data warehouse system as simple and fast as possible.

Five major requirements drove the design of the DB2ISERVER tool and, based on the extensive field usage, four of them (R#1, R#3, R#4, and R#5) were completely fulfilled successfully. The tool installed and configured a pristine working DB2 data warehouse system, in one install, onto the supported bare hardware effectively. The remaining requirement (R#2), that the required skill level of the one installing the cluster be minimal, had mixed results. It was found that the one deploying can have a minimum skill set from those skills that are usually needed to effectively deploy a cluster, namely hardware, networking, Linux, and DB2 skills, but that they must now be trained in a new skill: basic knowledge about how to use the DB2ISERVER tool itself. This skill can usually be successfully obtained during a two-hour hands-on training session.

The DB2ISERVER tool has been used extensively by IBM to deploy clusters in the field; lessons learned from these deployments are many and were discussed in detail. One major finding is that although the "master image" approach provided many benefits around simplifying the deployment and ensuring a pristine and working system, it did limit some of the flexibility of the tool with respect to differentiating the software stack on nodes with different roles in the data warehouse cluster.

A new version of the DB2ISERVER tool is currently in development. This new version will fix some of the shortcomings of the current tool as well as include some new features requested from those using the tool in the field.

Appendix A: Configuration File Details

A.1 Configuration File Specifications

The xcluster.cfg file is composed of three sections, each delimited by the following labels:

[KEYWORDS]

[NETWORK]

[NAGIOS]

The sections [KEYWORDS] and [NETWORK] are required. The [NAGIOS] section is optional; Nagios will be configured for use on the System node if, and only if, this section is present in the xcluster.cfg file. The order of the sections is important and should always follow the order above. The file itself is terminated by an empty section entitled: [END]

Section: [KEYWORDS]

The [KEYWORDS] section contains a list of keywords and their associated values that are used for DB2 UDB configuration as well as general network configuration for all nodes in the cluster. Currently valid keywords include:

KEYWORD	EXAMPLE VALUE	REQ?	DESCRIPTION
NODE_PREFIX	XCLUSTER	No	A prefix that can be appended to each hostname in the cluster. It

			is used as a typing shortcut when all nodes in the cluster will be given an identical prefix (recommended).
NETWORK_GATEWAY	192.168.1.1	Yes	This address will be configured as the default gateway on each node in the cluster.
DB2_PRIMARY_NETWORK	eth0	Yes	This identifies the network, identified by this network adapter name, that will be used as the primary network for DB2. This is the network that will be identified in the db2nodes.cfg file as well as the one that will be used by rsh/db2_all/nfs/db2start etc. If only one network adapter is provided in the [NETWORK] section it must match this adapter. If two networks are provided, one of them must match this adapter.

FCM_SECONDARY_NETWORK	ipoib0	No	This identifies the network that is used as the network for DB2 Fast Communication Manager (FCM). Usually this is a high-speed interconnect network. This keyword should only be specified if two networks have been specified.
TIMEZONE	US/Central	Yes	This is the time zone that will be configured on each node. The value must be provided in the standard Linux format as found in the "clock" file.
ROOT_PASSWORD	root9man	Yes	This is the password that will be set for the root user on each node. Note: This will automatically be masked out of the xcluster.cfg file after installation.
INST_USERNAME	db2inst1	Yes	This is the username of the DB2 instance owner. This must match the username found in

			the installation image.
INST_PASSWORD	db2admin01	Yes	This is the password that will be set for the DB2 instance owner. Note: This will automatically be masked out of the xcluster.cfg file after installation.
DAS_USERNAME	dasusr1	Yes	This is the username of the DB2 Administration Server (DAS) owner. This must match the username found in the installation image.
DAS_PASSWORD	db2das01	Yes	This is the password that will be set for the DB2 Administration Server (DAS) owner. Note: This will automatically be masked out of the xcluster.cfg file after installation.
FENCED_USERNAME	db2fenc1	Yes	This is the username of the DB2 fenced user. This must match the username found in the installation image.
FENCED_PASSWORD	db2admin01	Yes	This is the password that will be set for the DB2 fenced user.

			Note: This will automatically be masked out of the xcluster.cfg file after installation.

Section: [NETWORK]

The [NETWORK] section contains a list of all machines in the cluster. This is a very critical part of the xcluster.cfg file. If a user attempts to deploy an image on a machine that is not listed, or not correctly listed, in this section of the xcluster.cfg file (i.e., its eth0 MAC address does not exactly match one found in the file), no configuration will occur. In most cases, this means the user should correctly add this machine to the xcluster.cfg file and re-run the installation.

Each line in the [NETWORK] section consists of six space-separated columns. The last column, which represents a second network, is the only optional column in the line. Note that the last column can optionally be separated by a comma. Below is an explanation of valid values for each column in the line:

COL.	CONTAINS	VALID VALUES	DESCRIPTION
1	DB2 Partition Number	Integer Number	DB2 Database Partition number for all nodes except for the system node (i.e., nodes of type

			"SYS"). For SYS node, set this to be a negative number.
2	Node Type	SYS ADM DATA LOAD	Indicates the node type of this machine in the DB2 cluster and how it will be configured during deployment. Note that for the ADM nodes, the first node of this type read in from the file will be considered the FIRST ADMINISTRATION NODE on the system and will be configured as such (i.e., it contains the instance home directory). The remaining ADM nodes will only contain coordinator functionality.
3	Identifying MAC address	A valid MAC address (colon delimited)	MAC Address of first Ethernet adapter (eth0) on the machine. This uniquely identifies the machine in the cluster.
4	Hostname	A string - optionally containing the value "<NODE_PREFIX>"	The hostname that will be assigned to this node. Note that if the value "<NODE_ PREFIX>" is included anywhere in this string, it will be substituted with the value of the NODE_PREFIX keyword found in the [KEYWORKS] section.

			If two networks are provided for this node, the second hostname will be generated by adding a suffix to this hostname, depending on the second adapter type : • eth1 -> <HOSTNAME>_E1 • iboip -> <HOSTANAME>_IB
5	Network Adapter 1 Configuration	adapter:IPADDRESS/XX Adapter can be either: • ethX • ipoibX where X is an integer number starting at 0.	The adapter for the first (required) network as well as the IPADDRESS and NETMASK that will be configured for this adapter. Note that in most cases, this adapter name will correlate with the adapter name specified in the DB2_PRIMARY _NETWORK keyword in the [KEYWORDS] section.
6	Network Adapter 2 Configuration (OPTIONAL)	adapter:IPADDRESS/XX Adapter can be either: • ipoibX • ethX	The adapter for the second (optional) network as well as the IPADDRESS and NETMASK that will be configured for this adapter. Note that in most cases this adapter name will correlate with the adapter name specified in the FCM_SECONDARY_NETWORK

		where X is an integer number starting at 0.	keyword in the [KEYWORDS] section.

Section: [NAGIOS]

The [NAGIOS] section contains a list of keywords and their associated values that are required during Nagios configuration on the system node. If this section is not present, specifically if the section header "[NAGIOS]" is not present in the file, Nagios will not be configured.

The network information that is provided in the [NETWORK] section of this file contains a majority of the information needed to configure Nagios on the cluster; however, there are a few Nagios specific keywords that are required in this section. Currently the required keywords include:

KEYWORK	**EXAMPLE VALUE**	**DESCRIPTION**
NAGIOS_ADMIN_USERNAME	nagiosadmin	The username of the primary Nagios administrator. Note that in addition to this administrator, the DB2 instance owner will also be added as a Nagios

		administrator.
NAGIOS_ADMIN_PASSWORD	nagios01adm	The password that will be set for the Nagios administrator. This will automatically be masked out of the xcluster.cfg file after installation.
NAGIOS_ADMIN_EMAIL	tassi@server.com	The contact e-mail address for all Nagios administrators.

A.2 Network Configuration

One of the most important aspects of the xcluster.cfg file is the network information provided. It is critical that this information be specified correctly; otherwise, network connectivity, and everything in the cluster built on top of it (DB2, Nagios, etc.), will not function. In particular, note the following points:

- Two adapter types are supported: ethX (Ethernet) and ipoibX (infinband).
- No more then two networks can be specified for each machine (in column 5 and 6).
- The first network is mandatory; having a second network is optional.
- One of these networks must be an Ethernet network (eth0).
- The keyword DB2_PRIMARY_NETWORK must be set to match one of these two networks.
- The keyword FCM_SECONDARY_NETWORK is optional, and will only be used if it matches one of the networks specified in the [NETWORK] section.

Following these rules means that there are three possible ways to configure networking in the xcluster.cfg file:

One Ethernet network

DB2_PRIMARY_NETWORK = eth0

0 ADM 00:0C:29:E8:D4:C9 HOSTNAME1 eth0:192.168.99.2/24

0 SYS 00:0C:29:E8:D4:CA HOSTNAME2 eth0:192.168.99.3/24

Two Ethernet networks

DB2_PRIMARY_NETWORK = eth0 or eth1

FCM_SECONDARY_NETWORK = eth1 or N/A

0 ADM 00:0C:29:E8:D4:C9 HOSTNAME1 eth0:192.168.99.2/24, eth1:192.168.98.2/24

0 SYS 00:0C:29:E8:D4:CA HOSTNAME2 eth0:192.168.99.3/24, eth1:192.168.98.3/24

One Ethernet network and one InfiniBand network

DB2_PRIMARY_NETWORK = eth0 or ipoib0

FCM_SECONDARY_NETWORK = ipoib0 or N/A

0 ADM 00:0C:29:E8:D4:C9 HOSTNAME1 eth0:192.168.99.2/24, ipoib0:192.168.98.2/24

0 SYS 00:0C:29:E8:D4:CA HOSTNAME2 eth0:192.168.99.3/24, ipoib0:192.168.98.3/24

Note that the order of Ethernet and InfiniBand network can also be switched.

A.3 Configuration File Sample

```
# DO NOT DELETE THE SECTION HEADERS!
#
# VALID REQUIRED SECTIONS INCLUDE:     [KEYWORDS] [NETWORK]
# VALID NON-REQUIRED SECTIONS INCLUDE:  [NAGIOS]

[KEYWORDS]

NODE_PREFIX                 = TEST
NETWORK_GATEWAY      = 192.168.99.3
TIMEZONE                     = US/Eastern

DB2_PRIMARY_NETWORK        = eth0
FCM_SECONDARY_NETWORK = eth1

ROOT_PASSWORD          = password
DAS_USERNAME           = dasusr1
DAS_PASSWORD           = db2admin
INST_USERNAME          = db2inst1
INST_PASSWORD          = db2admin
FENCED_USERNAME        = db2fenc1
FENCED_PASSWORD        = db2admin

[NETWORK]

#
# NODE  NODETYPE    MAC                  HOSTNAME   IPADDRESS (up to 2)
#
-1          SYS            00:0C:29:E8:D4:CA  XCLUSTER0  eth0:192.168.99.10/24  eth1:192.168.12.50/23
0           ADM            00:0C:29:E8:D4:C9  XCLUSTER1  eth0:192.168.99.11/24  eth1:192.168.12.51/23
1           DATA           00:0C:29:AB:52:2F  XCLUSTER2  eth0:192.168.99.12/24  eth1:192.168.12.52/23
2           DATA           00:0C:29:AB:52:2B  XCLUSTER3  eth0:192.168.99.13/24  eth1:192.168.12.53/23
3           LOAD           00:0C:29:AB:52:2C  XCLUSTER3  eth0:192.168.99.14/24  eth1:192.168.12.54/23

[NAGIOS]

NAGIOS_ADMIN_USERNAME     = nagiosadmin
NAGIOS_ADMIN_PASSWORD     = password
NAGIOS_ADMIN_EMAIL           = tassi@system.com

[END]
```

Appendix B: Acronyms

- BMC – Baseboard Management Controller
- CSM – IBM Cluster Systems Management
- DAS – DB2 Administration Server
- DB2 ICE – DB2 Integrated Cluster Environment
- DB2 BCU – DB2 Balanced Configuration Unit
- DHCP - Dynamic Host Configuration Protocol
- DNS - Domain Name System
- DPF - Data Partitioning Feature
- DSH – Distributed Shell
- DSS - Decision Support System
- ETL – Extract Transform and Load
- FTP – File Transfer Protocol
- GB – Gigabyte
- HTTP - Hypertext Transfer Protocol
- HPC – High Performance Cluster
- IA64 – Intel Itanium architecture
- IB – Infinband
- IDE - Integrated Drive Electronics
- MPI – Message Passing Interface
- MPP – Massive Parallel Processing
- NFS – Network File System

- NIC – Network Interface Card
- NPACI – National Partnership for Advanced Computing Infrastructure
- OSCAR - Open Source Cluster Application Resources
- RSH – Remote Shell
- SCSI - Small Computer System Interface
- SMS – Short Message Service
- Ssh – Secure Shell
- TCPIP – Transmission Control Protocol and Internet Protocol
- TFTP – Trivial File Transfer Protocol
- x86_64 – x86 compatible 64-bit Architecture (AMD64 & Intel 64)

References

[1] J. D. Sloan, “High Performance Linux Clusters”, O’Reilly, 2005.

[2] “Computer Cluster”, Dec. 2006, http://en.wikipedia.org/wiki/Computer_cluster

[3] D. J. Becker, T. L. Sterling, D. F. Savarese, J. E. Dorband, U. A. Ranawak, and C. V. Packer, “Beowulf: A Parallel Workstation for Scientific Computation”, Proceedings of the International Conference on Parallel Processing, 1995.

[4] T. L. Sterling, J. Salmon, D. J. Becker, and D. F. Savarese, “How to Build a Beowulf: A Guide to the Implementation and Application of PC Clusters”, MIT Press, 1999.

[5] G. Bell, and J. Gray, “What’s Next in High-Performance Computing?”, Communications of the ACM, Vol. 45, No. 2, Feb. 2002, pg. 91-95.

[6] J. Layton, “The Coming of Diskless Clusters”, Oct. 2005, http://www.linux-mag.com/content/view/2269/

[7] H. Tang, A. Gulbeden, J. Zhou, W. Strathearn, T. Yang, and L. Chu, “A Self-Organizing Storage Cluster for Parallel Data-Intensive Applications”, Proceedings of the 2004 ACM/IEEE Conference on Supercomputing, IEEE, Nov. 2004.

[8] V.Cardellini, E. Casalicchio, M. Colajanni, and P. S. Yu, “The State of the Art in Locally Distributed Web-Server Systems”, ACM Computing Surveys, Vol. 34, No. 2, June 2002, pg. 263-311.

[9] J. Chapin, A. Chiu, and R. Hu, “PC Cluster for Signal Processing: An Early Prototype”, Proceedings of the IEEE International Conference on Sensor Array and Multichannel Signal Processing Workshop, IEEE, Mar. 2000, pg. 525-529.

[10] R. Gollan, A. Denman, and M. Hankel, “Clustering is not Rocket Science”, Linux Journal, Vol. 2006, No. 149, Sept. 2006.

[11] F. D. Sacerdoti, S. Chandra, and K. Bhatia, “Grid Systems Deployment & Management using Rocks”, Proceedings of the IEEE International Conference on Cluster Computing, IEEE, Sept. 2004, pg. 337-345.

[12] J. P. Navarro, R. Evard, D. Nurmi, and N. Desai, “Scalable Cluster Administration – Chiba City I Approach and Lessons Learned”, Proceedings of the IEEE International Conference on Cluster Computing, IEEE, Sept. 2002, pg. 215-221.

[13] J. Fernadez, E. Frachtenberg, F. Petrini, K. Davis, and J. C. Sancho, “Architectural Support for System Software on Large-Scale Clusters”, Proceedings of the IEEE International Conference on Parallel Processing, IEEE, 2004, pg. 519-528.

[14] C. Pettey, R. Butler, B. Rudnick, and T. Naughton, “Simple Maintenance of Beowulf Clusters in an Academic Environment”, Journal of Computing Sciences in Colleges, Vol. 18, No. 2, Dec. 2002, pg. 208-214.

[15] G. Otero, “Building Linux Clusters”, Linux Journal, Nov 2000.

[16] S. Steiner, “Building and Exploring a Beowulf Clusters”, Journal of Computing Sciences in Colleges, vol. 17, no. 2, Dec. 2001, pg. 78-87.

[17] D. C. Bergen, and B. P. Miller, “Building an MPI Cluster”, Crossroads, ACM Press, Vol. 8, No. 5, Aug. 2002.

[18] P. De Palma, A. Wiborg, and A. Withers, “Super Computing on a Budget”, Journal of Computing Sciences in Colleges, Vol. 17, No. 2, Dec. 2001, pg. 71-77.

[19] S. Vaidya, and K. J. Christensen, “A Single System Image Server Cluster using Duplicate MAC and IP Addresses”, Proceedings of the 26th Annual Conference on Local Computer Networks, IEEE, Nov. 2001, pg. 206-214.

[20] Y. Georgiou, J. Leduc, B. Videau, J. Peyrard, and O. Richard, “A Tool for Environment Deployment in Clusters and Light Grids”, Proceedings of the IEEE International Conference on Parallel and Distributed Processing Symposium, IEEE, April 2006.

[21] OSCAR (Open Source Cluster Application Resources) Homepage, http://oscar.openclustergroup.org/

[22] M. Meredith, T. Carrigan, J. Brockman, T. Cloninger, J. Privoznik, and J. Williams, “Exploring Beowulf Clusters”, Journal of Computing Sciences in Colleges, Vol. 18, No. 4, April 2003.

[23] R. Ferri, “The OSCAR Revolution”, Linux Journal, Vol. 2002, No. 98, June 2002.

[24] Rocks Homepage, http://www.rocksclusters.org/

[25] IBM Manual, “Balanced Configuration Unit for Linux: Guide and Reference”, Version 1, 2005, pg 68.

[26] M. J. Katz, P. M. Papadopoulos, and G. Bruno, “Leveraging Standard Core Technology to Programmatically Build Linux Cluster Appliances”, Proceedings of the IEEE International Conference on Cluster Computing, IEEE, Sept. 2002, pg. 47-53.

[27] “Data Warehouse”, Dec. 2006, http://en.wikipedia.org/wiki/Data_warehouse

[28] D. Feinberg, and M. Beyer, “Magic Quadrant for Data Warehouse Data Management Systems 2006”, Gartner Inc., Sept. 2006.

[29] DB2 Information Center (Online), V9.1, http://publib.boulder.ibm.com/infocenter/db2help/index.jsp

[30] C. K. Baru, G. Fecteau, A. Goyal, H. Hsiao, A. Jhingran, S. Padmanabhan, G. P. Copeland, and W.G. Wilson, “DB2 Parallel Edition”, IBM Systems Journal , Vol. 34, No. 2, 1995, pg 292-322.

[31] IBM Manual, “Balanced Configuration Unit for Linux: Guide and Reference”, Version 2.1, 2006, pg 15-19.

[32] Nagios Homepage, http://nagios.org/

Trademarks Attribution Statement

IBM and **DB2** are registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

Intel is a trademark of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product or service names may be trademarks or service marks of others.