Diskless Image Management (DIM) for Cluster Administration

Peter Morjan, Greg Rodgers, and Ross Aiken
IBM, Inc.

George Turner, Dave Hancock
Indiana University

Laurie Feinswog
WW Wordsmiths

Diskless Image Management (DIM) is a centralized management tool developed by Peter Morjan at IBM for large scale computing systems. This report discusses the technology behind DIM and its features and benefits.

1 Diskless Image Management Overview

Large computing systems have large administration needs. But just as technologies have evolved to take advantage of certain parallelisms of large scale computing, administrating these technologies must evolve to take advantage of the associated operational efficiencies.

Using a straightforward push technology, and scalable to thousands of blades, Diskless Image Management (DIM) allows system administrators to boot, patch, or modify one, several or all distributed images in minutes from a single management console.

DIM was prototyped on the MareNostrum cluster with 2406 blades, but is scalable to 7000 blades. Using IBM JS20 blade technology MareNostrum consists of 172 BladeCenters.

1.1 Introduction to DIM concepts

While saving administrators from the significant legwork of changing cluster operating systems one by one is important, speeding up management tasks is only one benefit to be realized from any large scale cluster administration tool. DIM was developed to satisfy many administrative needs:

- Maintaining multiple distributions on a cluster
- No distribution modification requirements
- Space savings with shared directories
- Protection of shared resources
- Synchronous or asynchronous image management
- High speed image updates
- Simple (single command) distribution swapping
- Simple hardware management

- Scalable performance

We discuss how DIM addresses each of these needs in the following sections.

1.2 Multiple Distribution Maintenance

With DIM, images are managed from image servers, with each server supporting a defined set of blades in the cluster. In a distributed network architecture, communication between the image servers and the blades they support occurs over a specified boot network, and does not interfere with other network communication. See Figure 1.

An image server receives a clone of the image from a master image residing on the primary image server, or management node. Depending on the hierarchical structure on which you choose to implement your solution, this master can be on an actual disk-based node, or can be obtained from another image server. As we will discuss later, this two-level hierarchy scales better for very large clusters.

One master image server is needed for each personality/distribution. In this way, the image server can manage different personalities of a Linux distribution and/or multiple Linux distributions on a single cluster.

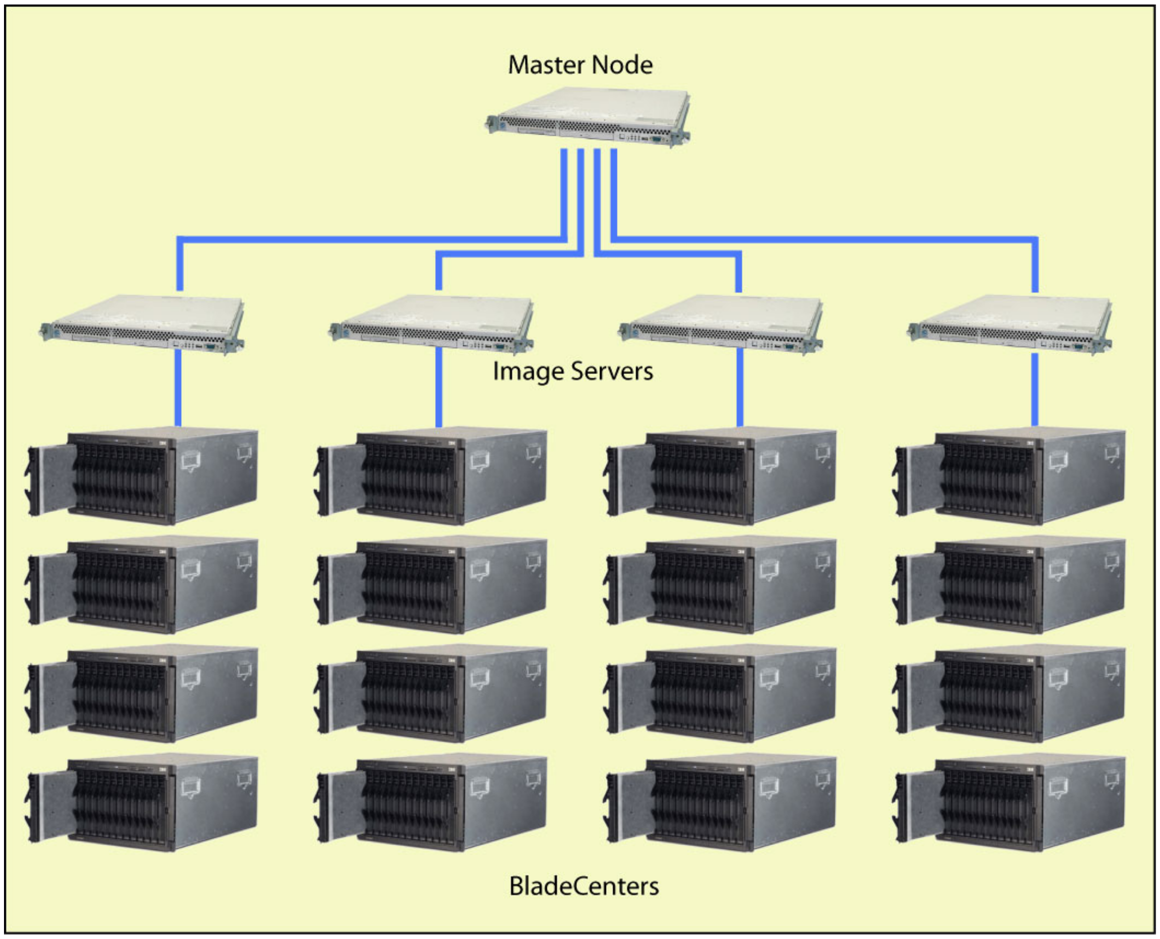

Figure 1

1.3 Distribution Integrity

DIM uses common Linux utilities, such as *dhcpd*, *snmp*, *nfs*, *tftp*, and *rsync*, without modification, and has been tested on native RedHat and SuSE masters. It does require at minumum Linux 2.6 as the *linuxrc* feature introduced in 2.6 is a core component of DIM that allows each blade to mount the image boot from the image server. For more information, see section 2.4 which details the DIM Process.

The distribution and installation of DIM is equally non-invasive. DIM comes as an RPM package of Perl scripts and man pages. It is installed only on the image server, while the master images are left unaltered.

1.4 Disk Space Savings

There is a large parallelism in cluster management that can be leveraged specifically for the purpose of reducing disk space needs and therefore operational expenses. Each blade image managed by the image server is composed of two parts: the read-only image that is shared across all blades served by the image, and the read-write image that is unique to each blade. The read-only portion of the image can only be updated by the image server, whereas the files in the read-write portion can be updated by the image server, and also written to by the individual blade as they would for log files. This differentiation provides significant overall space savings by storing only one copy of first level directories such as /usr/bin, yet still provides flexibility for image individualization.

1.5 Protection of Shared Resources

In addition to preventing write access to the shared image files, each blade image is a single distinct file on the image server, and is accessed by its blade as a loop-back mounted filesystem. This establishes an automatic quota for the image filesystem, and prevents rogue blades from corrupting shared resources.

1.6 Flexible (Synchronous or Asynchronous) Image Management

Because clusters are shared resources themselves, and are rarely used as homogenous computational platforms, it is necessary to be able to manage blade images in a flexible manner. We have already seen that DIM is capable of managing several flavors of Linux on a single cluster. DIM also can be used to send administrative commands to any one or many blades in a cluster using a blade taxonomy we describe in section 2.2.

Images can also be updated incrementally from the image server, regardless of whether the blade is on or off.

1.7 High Speed Image Updates

Although the time savings from pushing image updates to a cluster is already quite substantial, DIM technology uses the Linux *rsync* facility, which means that incremental updates are fast. If changes are contained in a first level read-only directory, updating a few files across thousands of notes takes only a few seconds. If you are performing a global synchronization, an update can take as little as 2-6 minutes if only a few files require updating. The more files that require updating, the more time it will take. For example, to push four months of images updates across 2000 blades, it could take 40 minutes.

1.8 Simple Distribution Swapping

DIM manages images with a very simple set of commands. If a system administrator wants to perform a parallel operation from the management node, he can manage the DHCPD configuration file on every image server with a single command called dim_dhcp. The dim_dhcp command assigns an image to an actual physical blade by creating or updating an entry for the MAC address of that blade. Thus, a single command can update images across the cluster. For more information on DIM commands, see section 2.3.

1.9 Simple Hardware Management

Along with simple software management, it is important for a cluster management tool to be able to easily manage hardware modifications. To handle identification of thousands of blades, DIM uses an XML definition file to describe the geography of the network, and a plug-in mechanism to get the MAC addresses for cluster components.

The XML definition file can handle any type of hierarchy of names. In the case of a cluster of BladeCenters, for instance, we use names with three modifiers to identify the rack number, the BladeCenter within the rack, and the blade number within the BladeCenter. See section 2.2 for more details.

When a new blade gets plugged in, the management module picks up its MAC address using SNMP, and maps it to a component name in the definition file.

1.10 Scalable Performance

A single image server performs adequately for clusters up to 56 blades. When the number of blades in a cluster becomes large, however, DIM makes it easy to add a layer in the hierarchy. The management node pushes information over a separate network, the bootnet, to the slave layer of image servers, which in turn push images to the cluster subset it is responsible for. Using a hierarchical network architecture reduces the possibility of network contention during high throughput times, and also distributes the risk of cluster failure, as the loss of a single image server would only affect the rack, or set of blades it is responsible for.

2 DIM Technical Details

In the following sections, we review DIM technology, including network architecture, blade taxonomy, DIM commands, and the DIM process, in greater detail and in the context of a large blade cluster.

2.1 Architecture

As we discussed in section 1.10, it is possible to run DIM on a cluster from a single image server and a global network. However, a distributed bootnet architecture – two or more layers of image servers, and a distributed network – is recommended for large clusters due to its superior scalability. We refer to the primary image server as the management node.

In the implementation shown below in Figure 2, there are four separate networks used to communicate within the cluster. For the purposes of this technical report, we are concerned primarily with the bootnet and the usernet networks.

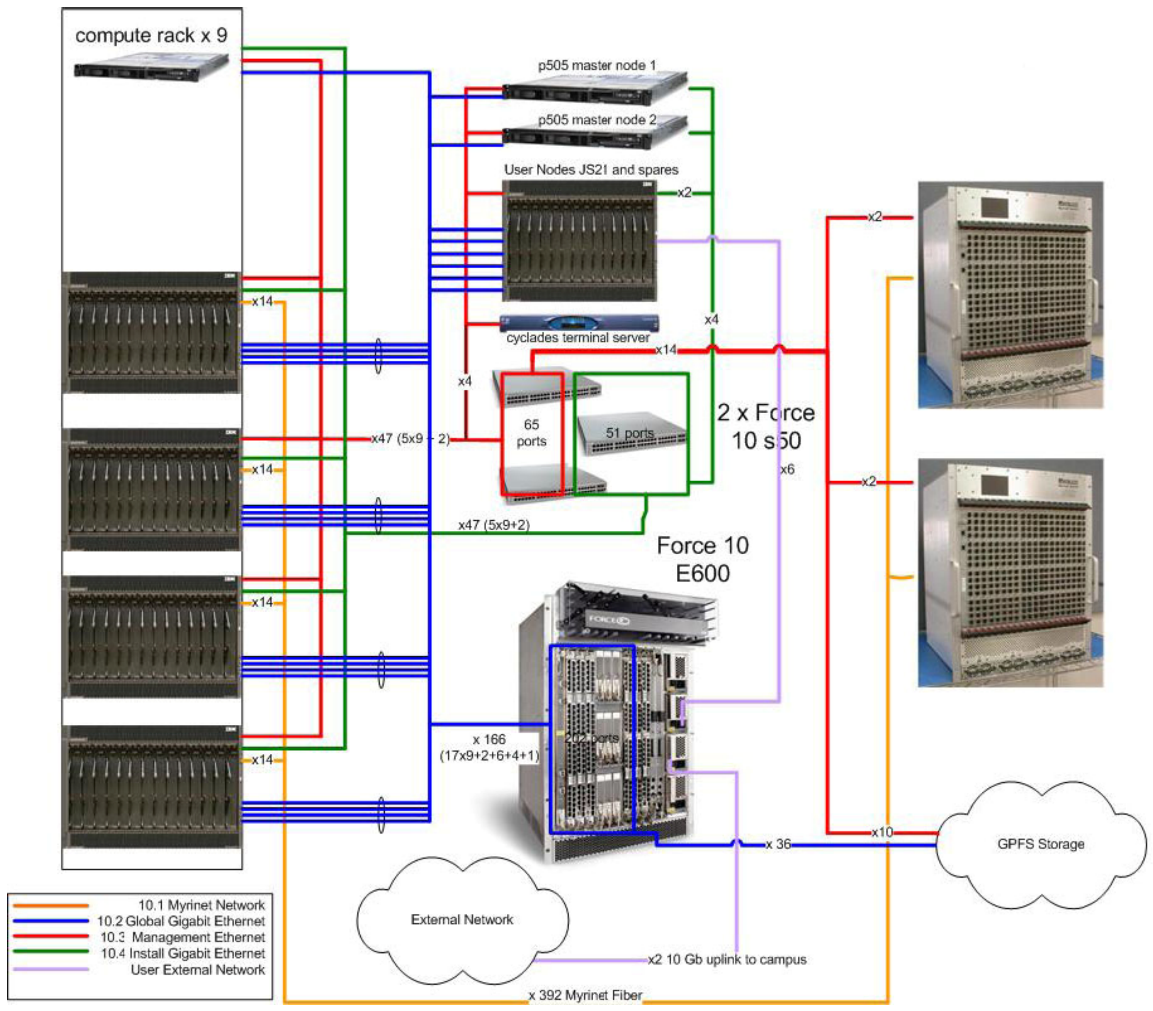

Figure 2

When we talk of a distributed bootnet, we are talking of a network that is isolated to a given image server and the BladeCenters it manages. Typically the first ethernet interface (eth0) and switch module 1 is used to form this network on each BladeCenter. The dhcpd on the image server is told to listen to only eth0, and the images are exported to the blades over this network.

The global ethernet is formed with eth1 and switch module 2. No system administration or image support is required on this network that we call usernet.

The benefits of a distributed bootnet can be summarized as follows:

- No network congestion when booting the entire cluster
- User traffic does not create contention with image maintenance
- Global network (usernet) failure does not result in reboot of entire cluster
- More IO bandwidth into the blades

2.2 Taxonomy

The issue of usably naming thousands of blades in a cluster is addressed by the creation of an XML definition file. You can define any hierarchy of names you require for your cluster. For BladeCenters, we have found that a three-level structure is most effective.

In the taxonomy we use for MareNostrum, each blade is assigned three numbers in the cluster coordinate system – the server number, the BladeCenter number, and the blade number – and is represented as sXXcYbZZ. The variable XX ranges from 1 through 42, the number of image servers deployed in the cluster, Y ranges from 1 through 4, the number of BladeServers managed by an image server, and ZZ ranges from 1 through 14, the number of blades in a BladeCenter. In order to perform an operation on an entire BladeCenter, for instance to power it up or down, we have created a component name that is represented as sXXcYmm.

While this three-dimensional coordinate system is useful for pinpointing a specific component geographically, there are times when you will need to cycle through a sequence of components. For this case, we have assigned a sequential, or linear name to each component as well as a logical name. The linear equivalent of s1c1b1 is b1, and the equivalent of s1c2b14 is b28. Similarly, for BladeCenter components, s1c1mm is equivalent to mm1, and s23c2mm is equivalent to mm90.

The nameserver is programmed to reflect the same naming schema as the XML definition file.

2.3 Commands and Scripts

DIM has only nine administrative commands, and comes with a small sample script for synchronizing images. The commands and their descriptions are listed below:

Command	**Description**
dim_dhcp	Manage dhcpd.conf file on management node and image servers.
dim_image	Create image files.
dim_buildzimage	Build a network bootable file, zImage.
dim_nfs	Export image files using nfs.

dim_bctool	Utility to manage BladeCenter via SNMP. This command is for blade specific operations, such as "power on blade 6".
dim_bcadmin	Utility to manage BladeCenter via SNMP. This command is for BladeCenter wide operations, such as "power on whole chassis".
dim_sync_master	Pull image changes from master blade (master blade to management node).
dim_sync_server	Push changes from management node to other image servers.
dim_sync_image	Make the copies for each served blade and distribute.

The last three commands listed, *dim_sync_master*, *dim_sync_server*, and *dim_sync_image*, make up the three step process of globally distributing image updates from the master image to each blade. We have created a short, customizable script, *syncit*, to handle these commands. It should be run by the administrator on the primary image server, or management node.

Because granular image updates are common, *syncit* takes a directory or file to synchronize as an optional argument. Without the argument, the entire image is incrementally synchronized with the master. You can customize the script for your specific environment by hardcoding network and distribution information, including the distribution name, the hostname of the master blade, and the list of images servers, at the beginning of the script.

2.4 DIM process

The following steps describe the key points in the DIM image distribution and boot process, and are illustrated in Figure 3.

1) Copy master distribution image from the management node to the image server.
2) Run dim_image to create images from directory master. This step creates the image files for each blade image.
3) Run dim_nfs to make images accessible to blades (via NFS export/mount)
4) Run dim_build_zimage to create boot zImage in /tftpboot. This image is the native distribution kernel, but with a custom linuxrc. This step creates the bootable file for the individual blades.
5) Run dim_dhcp to assign blades to distributions (update dhcp.conf)

Once the images have been distributed, the blades boot via the following process:

1) Blade network boots zImage from Image Server
2) Blade mounts root filesystem from storage server during custom linuxrc execution.
3) CHROOT to mounted new root filesystem, continue standard Linux boot process.

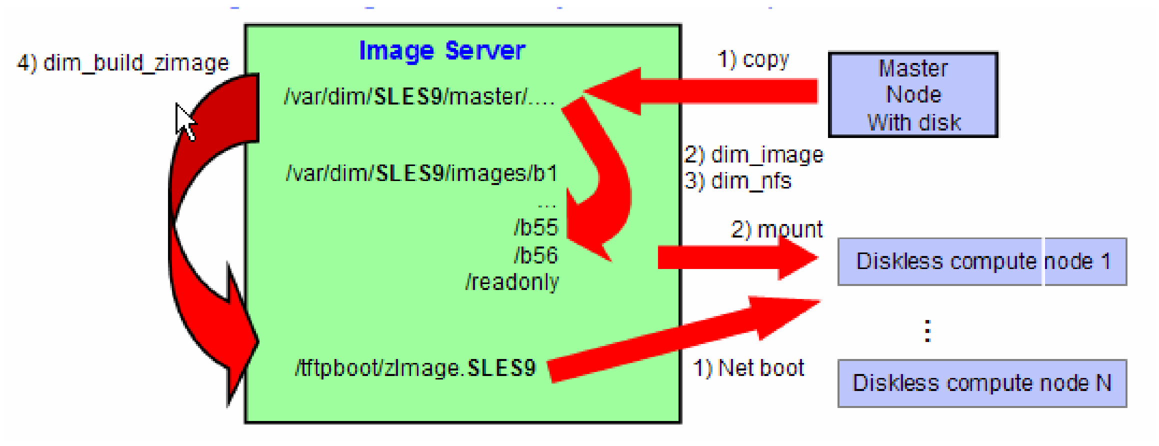

Figure 3

Synchronizing the distributed images to changes made to the master image is simply a matter of applying the *syncit* script, or equivalently, the execution of the following three steps. These steps are illustrated in Figure 4.

1) Run dim_sync_master to pull image changes from the management node.
2) Run dim_sync_server to push the changes from the management node to the other image servers.
3) Run dim_sync_image to make image copies for each served blade.

Since these files are already mounted to their respective blades, once the images are copied, the blades see the changes immediately.

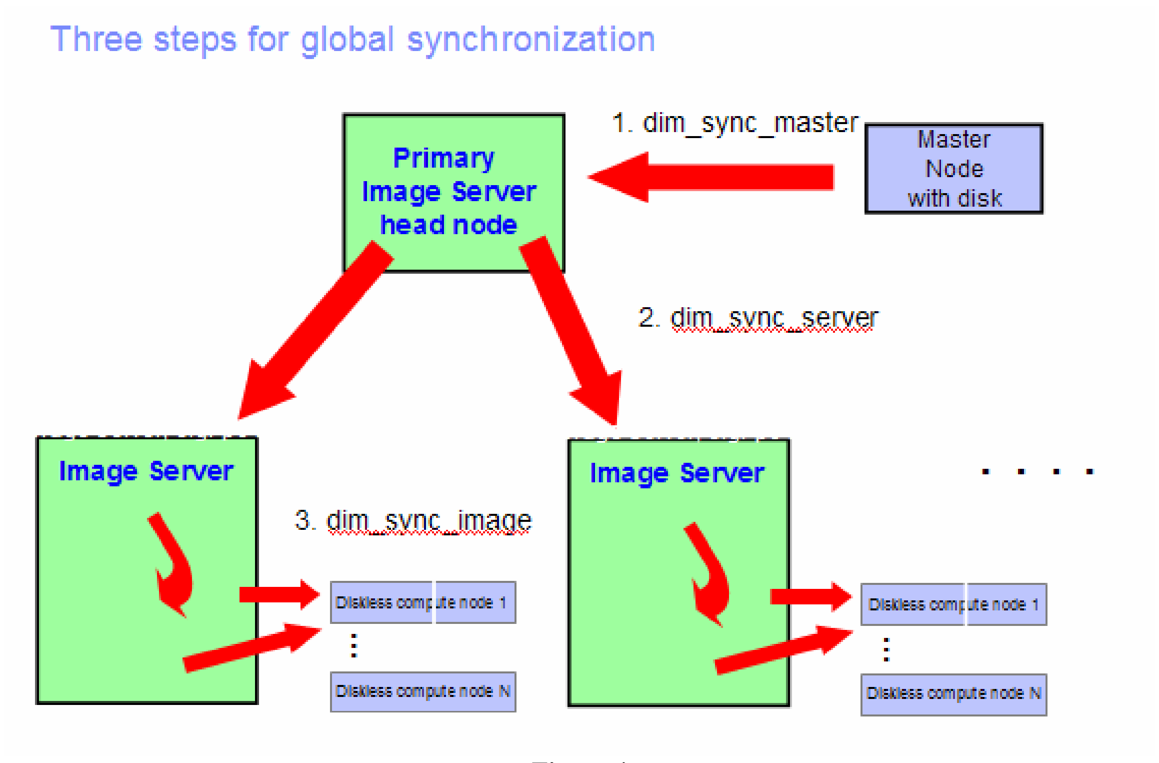

Figure 4

3 The advantages of DIM

Diskless Image Management (DIM) is a superior tool for large scale cluster management. For system management issues, DIM provides:

- Speed and simplicity of updates
- Scales to thousands of blades with simple 2-level hierarchy
- Dynamic update
- No changes to Linux required
- Asynchronous image management
- Multiple distributions supported
- Saves space with shared read-only directories

In terms of reliability, DIM has:

- Fewer moving parts
- No contention in bandwidth
- No single point of failure with hierarchical architecture