CLUSTER AND NETWORK MANAGEMENT INTEGRATION
An SNMP-based Solution

Rodrigo S. Alves, Clarissa C. Marquezan, Philippe O. A. Navaux, Lisandro Z. Granville
Institute of Informatics, Federal University of Rio Grande do Sul, Av. Bento Gonçalves 9500, Porto Alegre,Brazil

Keywords: High Performance Clusters, Network Management, Cluster Management.

Abstract: The management of high-performance clusters, in several situations, needs to be integrated with the management of computer networks. In order to achieve such integration, a network management architecture and protocol, such as the SNMP, could be used for cluster management. This paper investigates the required integration through the implementation of SNMP agents deployed in the cluster infrastructures, as well as the development of a management system for these SNMP agents.

1 INTRODUCTION

Cluster Computing research has been growing in recent past years, mainly when clusters are used as substitutes for supercomputers in some applications that require high performance processing (Buyya, 1999). The popularization of clusters has occurred due to their relative low price and their capacity of easily increase the amount of processing power through the addition of new worker nodes.

The difficulty in maintain a cluster infrastructure is usually proportional to the number of nodes that compose the cluster. Managing a cluster without any tool that automates the management tasks can be prohibitive, mainly when the number of nodes is high. Thus, it is essential the use of management tools that help the administrator to deal with the growing number of nodes.

Today, there are some tools that automate cluster management. However, many of them have their own associated language and their own way of work: currently there is no standardized way to support cluster management procedures.

This problem turns the integration of cluster management tools and others management tools hard. This situation is still more difficult because clusters are network components usually located in computer networks universities and companies. So, is not rare the case in which the network administrator is the same person that administrates a cluster. Considering the difficulties to integrate management tools, the network/cluster administrator is forced to use a set of tools to manage the network, and other set of tools to manage clusters.

Thus, there is the need of a new approach for integrated cluster and the network management. Since the traditional and widely used solution for network management is based on the SNMP (Simple Network Management Protocol) framework (Harrington et al., 2002), the integration with cluster management could be obtained through the use of SNMP in the cluster domain. This work presents an SNMP-based cluster management system used to manage two clusters of our testing labs. The system is composed by SNMP agents and one Web-based management system. The work presented in this paper is the evolution of the paper (Alves, 2004).

The remainder of this paper is organized as follows. Section 2 presents related work concerning common cluster management issues. In Section 3 we present two cluster MIBs (Management Information Base) supported by the implemented SNMP cluster agents. Section 4 shows a case study where the Web-based cluster management system and its main operations are presented. Finally, we finish this paper with conclusions and future work in Section 5.

2 MANAGEMENT OF HIGH PERFORMANCE CLUSTERS

According to Buyya (Buyya, 1999), a cluster consists of a collection of interconnected stand-alone computers (nodes) that work together like a single integrated computational resource. The front-end is the cluster node that presents to the user the integrated and single resource image. The front-end

S. Alves R., C. Marquezan C., O. A. Navaux P. and Z. Granville L. (2005).
CLUSTER AND NETWORK MANAGEMENT INTEGRATION - An SNMP-based Solution.
In *Proceedings of the Second International Conference on e-Business and Telecommunication Networks*, pages 318-323
DOI: 10.5220/0001417503180323

is crucial for the cluster operation because it implements a single access point from where the users submit applications to be executed in nodes. Often, nodes are personal computers interconnected with each other through the cluster network. This network is responsible to allow the communication among processes running in different nodes.

Differently from what happens in the network management area, there is no consolidated and widely accepted notion of what cluster management exactly means. In this scenario, we try to organize the current cluster management tools in three broad and general classes: cluster monitoring, user task management, and administrative tools. Since the functionalities found in these tools may be very similar, is not rare that some tools implement functionalities that would fit in more than one class.

Cluster monitoring tools are used to check the internal status and utilization of the cluster resources. Ganglia (Massie, 2005), the most spread monitoring tool, is a distributed system able to monitor inter-cluster and intra-cluster information. Another tool is SIMONE (SNMP-based Monitoring System for Network Computing) (Subramanyan et al., 2000), that offers node description, processes information (e.g. CPU time, memory usage), nodes information (e.g. percentage of CPU usage, memory usage, network interfaces) and information about traffic on each node. Basically, SIMONE implements most of the management information that Ganglia does.

User task management tools are designed for job scheduling. Such tools allow the allocation of a subset of the cluster nodes for the execution of the tasks of a user. The most spread tool here is Open PBS (Portable Batch System) (PBS, 2005). The basic PBS operation is implemented as a FIFO queue. It offers mechanisms to view and interact with the execution queue, and uses policies to control the abusive use of clusters resources by just one user, harming others users.

A*dministrative tools* automate cluster operation tasks such as node image replication and parallel commands. Basically, the main objective of these tools is to offer scalable operations for clusters. System Imager (System-Imager, 2005) is a tool of this group. It offers functionalities for maintenance of a same operating system in all cluster nodes.

From a network management perspective, all these cluster tools may not always be identified as management tools in the strict word mean, but in cluster terms it is not rare to find such denomination. A more critical problem is the fact that these tools can not be integrated with network management without complex adaptation work. SIMONE is the unique cited tool that offers facilities for SNMP integration, however it is focused in the management of general distributed systems, and does not deal with specific cluster issues such as nodes allocation and internal/external network interation.

3 CLUSTER MANAGEMENT MIBS AND SNMP AGENT

The developed management system handles SNMP agents inside clusters. The cluster management information supported for each SNMP agent is defined in MIBs for clusters. These defined MIBs relay on the analysis of information required to manage a cluster and on the analysis of other existent MIBs. Two already standardized MIBs were important in this work: MIB-II (Mccloghrie and Rose, 1991) and HostResources (Grillo and Waldbusser, 1993). They allow us to use several data of these MIBs, instead of redefining them on the clusters MIBs. From MIB-II we borrowed the *system* group, which provides general machine information. The network information is obtained from the *interfaces* group. From the HostResources MIB we borrowed information from three following groups: (a) *hrSystem*: provides information related to the system status; (b) *hrStorage*: provides information about disk and memory usage; (c) *hrDevice*: provides data about disk partitions.

Complementarily, the specific information related to clusters, not covered by MIB-II and HostResources, were defined in two new MIBs: one referring to nodes management and the other referring to front-end management. The functionalities supported in the current version of these new MIBs include some features of two management groups (described in the previous section): monitoring tools and users tasks management tools.

3.1 Cluster MIBs Description

Figure 1 presents the MIBs designed to manage cluster nodes and front-ends. As showed in Figure 1, ClusterNode MIB is composed by four groups: *clAllocation*, *clCpu*, *clTemperature* and *clProcesses*.

Node allocation and deallocation are performed through *clAllocation* group. The read-and-write object *clNodeAllocUser* stores the name of the user who has allocated a node or the special value “ALL” if the access to the node is free. After allocation, any user, except the one that allocated the node, is unable to login in the node. The node deallocation is performed by the system administrator, whom must attribute the value “ALL” in *clNodeAllocUser* object. The *clNodeAllocDuration* should be adjusted

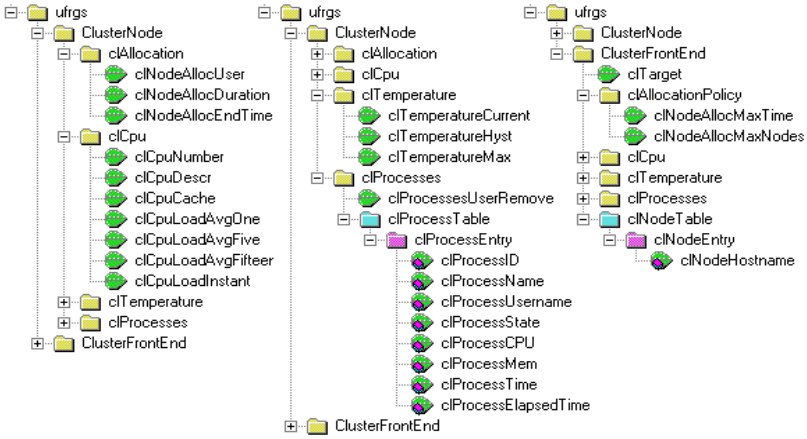

Figure 1: ClusterNode and ClusterFrontEnd MIBs.

to indicate the number of hours the allocation is valid. Such adjustment must be performed before the effective allocation. Finally, the reading object *clNodeAllocEndTime* inform the date and time that the allocation will end up according the *clNodeAllocDuration* value.

The *clCpu* group exposes data related to node CPU(s). *clCpuNumber* inform the number of CPUs the node possesses, and *clCpuDescr* presents a description of each CPU. The amount of cache memory of each processor is presented in *clCpuCache*. CPU utilization data are obtained with four objects: (i) *clCpuLoadAvgOne*, (ii) *clCpuLoadAvgFive*, (iii) *clCpuLoadAvgFifteen* and (iv) *clCpuLoadInstant*. The objects (i), (ii), (iii) present the processing load average in the past minute, past five minutes and past fifteen minutes, respectively. The last object (iv) informs the processing load in a given moment. These data are computed based on an arithmetic average among the CPUs usage percent of each node.

The *clTemperature* group informs the values obtained through temperature sensors of the node motherboard. The reading object *clTemperatureCurrent* indicates de current processor temperature in Celsius scale. The reading object *clTemperatureHyst* indicates an alert limier, which over it some proactive actions must be executed. *clTemperatureMax* inform the maximum temperature limier allowed from which the machine must be turned-off. As temperature limier can be just changed by complex manual interventions in each node, *clTemperatureHyst* and *clTemperatureMax* can only be used to obtain such values, but can not be used to perform changes.

Information related to nodes processes is located in the *clProcesses* group, which is composed by *clProcessesUserRemove* and *clProcessTable*. The leaf *clProcessesUserRemove* is used to finish all user processes that are running in a node (specifying the username) or to finish the processes of all users that are running some process in the node (specifying the string “ALL”). The processes can be finished individually using the *clProcessState* object. The *clProcessTable* contains information about the processes running in a node. A part of this information is provided by the HostResources MIB, however, other important data (e.g. owner of process and elapsed time the process is running) are not provided by this MIB. This leads to the necessity to extend the information contained in HostResources by defining the *clProcessesTable* in ClusterNode MIB. An optimization inserted in this work is based on the fact that, concerning cluster context, generally the interest in the running process are related to users processes and not system processes. Thus, we decided to implement in ClusterNode MIB data already supported by HostResources MIB. Nevertheless, only user processes will be exported in our MIB, providing an economy in network traffic.

Each line in clProcessTable refers to a process executing in a node. clProcessID index each process in the table and gives the process identifier (PID). The *clProcessName* object indicates the process name and *clProcessUsername* object informs the username of the process owner. Process status (e.g. running, IO, stoped or zoombie) is given by the read/write *clProcessState* object. It is also used to enable the ending of a process, when the “kill” value was set in this object (this is based on HostResources MIB). In addition, *clProcessCPU* and *clProcessMem* objects inform the percentage of CPU and memory the process is consuming. The object *clProcessTime* indicates the amount of processor time used by the process, and the *clProcessElapsedTime* object indicates elapsed time the process is running.

The ClusterFronEnd MIB is too similar to ClusterNode MIB, beging formed by an scalar object (*clTarget*), by four groups (*clAllocationPolicy*, *clCpu*, *clTemperature*, *clProcesses*) and by a table (*clNodeTable*). The differences between these two MIBs are related to the existence, in the ClusterFrontEnd MIB, of the following elements: *clTarget* object, *clAllocationPolicy* group and *clNodeTable* table. The *clTarget* object will be discussed in section 3.2.

The *clAllocation* group from ClusterNode MIB was substituted by *clAllocationPolicy* in ClusterFrontEnd MIB. That was required due to it makes no sense allocate the front-end for a user. On the other hand, it is the front-end responsibility to store the cluster allocation policy. An allocation policy is a set of rules that aims to provide a better sharing of the cluster resources; allowing most users as possible to use the cluster at the same time and trying to reduce the waiting time to get access to the resources. In this way, *clAllocationPolicy* group stores the allocation policy and is composed by the read/write leaves *clNodeAllocMaxTime* and *clNodeAllocMaxNodes*, which indicate, respectively, maximum hours and maximum nodes per allocation. Finally, *clNodeTable* stores in *clNodeHostname* object the *sysName* value from MIB-II of the nodes. *clTarget* and *clNodeHostname* are key objects in the routing of SNMP messages to target nodes.

3.2 SNMP Agent Implementation

The SNMP agents have been developed in Linux environment, implemented in C language and using the facilities of the NET-SNMP package (NET-SNMP, 2005) to build agents based in SNMPv3 protocol (Blumenthal and Wijnen, 1998), which enables the support for several security issues, such as user authentication and data integrity assurance.

Two different, but complementar, SNMP agents were implemented: (i) one running inside each cluster node and (ii) another running in the front-end. In this way, the cluster components (nodes and front-end) have SNMP agents. Nevertheless, only the front-end agent is accessible directly through the management system. Node agents can only be accessed through the front-end agent. In this case, the front-end, besides providing support for managing itself, also acts as an SNMP proxy.

The need to implement an SNMP proxy is derived from a difficulty intrinsic of most clusters configurations: generally the single machine accessible from the external to the cluster is the front-end, while its nodes are confined inside the cluster internal network, hidden behind the front-end. Therefore, the goal of our SNMP proxy is to allow the front-end to perform the interaction between cluster internal and external networks.

Figure 2 presents an example of integration through the SNMP proxy. In this example, a manager software needs to get the value of *clCpuNumber* object from ClusterNode MIB. However, before performing this operation the manager needs to manipulate *clTarget* object from the ClusterFronEnd MIB to prepare the front-end SNMP agent to handle SNMP messages designed to cluster nodes. In this example the manager must set *clTarget* to forward the SNMP messages to node 2.

Receiving an SNMP request the front-end agent must, before proceeding with the standard SNMP message processing, examine the value of *clTarget* object. Whenever the *clTarget* value is the same of one of the values stored in *clNodeHostname* of *clNodeTable*, the front-end agent forwards the SNMP message to the node identified in *clTarget* and then waits the reply from the target node, to sent it back to the management software.

This indicates that the manager software needs, before interacting with a node, change the *clTarget* value in the front-end to inform the node that will effectively take care of the request. In Figure 2, (1) the cluster manager asks the writing of “Node 2” in *clTarget* object and (2) receives a reply confirming the operation. After, all SNMP messages arriving in the front-end will be forwarded to the node 2 of the cluster. In this example, (3) the manager asks the value inside *clCpuNumber*. This request is forwarded (4) to the agent of node 2 that answers (5) informing the number of CPUs from node 2. Finally the reply (6) is sent back to the software manager.

4 THE SNMP-BASED CLUSTER MANAGEMENT SYSTEM

The SNMP-based cluster management system is composed by a set of PHP scripts (PHP, 2003) that

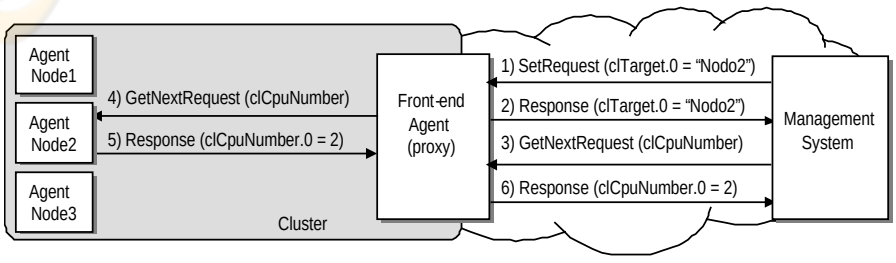

Figure 2: Basic operation schema of SNMP proxy

the cluster administrator can access via a Web interface. Web-based management is largely discussed and its advantages are well-known in the network management area.

The graphical interface allows a progressive interaction between the administrator and the system. Initially, information allows the verification of global cluster data (Figure 3), such as front-end hostname, the number of nodes and processors (obtained in *clNodeTable* from ClusterFrontEnd MIB and in *clCpu* from ClusterNode MIB), and the cluster total amount of memory and hard disk (obtained via HostResources MIB).

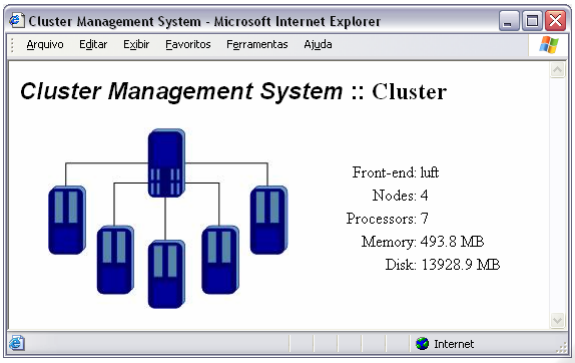

Figure 3: Interface of Management System.

Clicking in the cluster image, the administrator accesses an HTML page where the nodes are presented (Figure 4). It presents images that show two basic information of each node: the running workload and the allocation status. According with the node CPU workload, the node image assumes a different colour (green, yellow or red) that represents, respectively, a low (fewer than 30%), a medium (between 30% and 80%) or a high workload level (higher than 80%). The other information presented is the node allocation status. If the node is currently allocated to a user, then the node image will be changed adding to it the figure of a small lock. This information is obtained accessing *clCpu* and *clAllocation* groups of ClusterNode MIB.

The "*Allocate Nodes*" presents a new interface (Figure 5) where the user can allocate cluster nodes and obtain allocation status of all nodes, including the user that has allocated a node and the time when the allocation will expire. With this information, obtained from ClusterNode MIB, it is possible to perform an allocation choosing desired duration and number of nodes. The allocation duration options are offered according to the cluster allocation policy, and the nodes limit is informed below the nodes table (this information are obtained accessing *clAllocationPolicy* group from ClusterFrontEnd MIB). If an allocation request exceeds the cluster allocation policy, the request is performed partially.

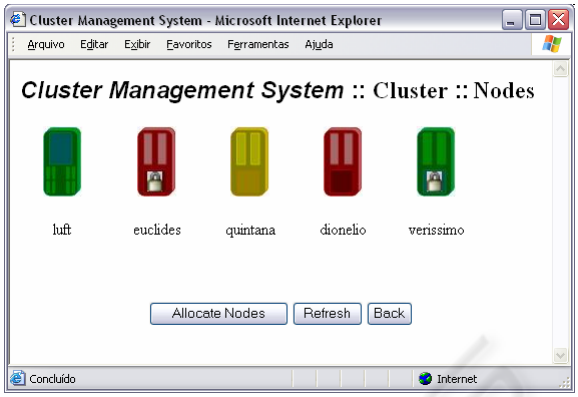

Figure 4: Interface showing front-end and nodes.

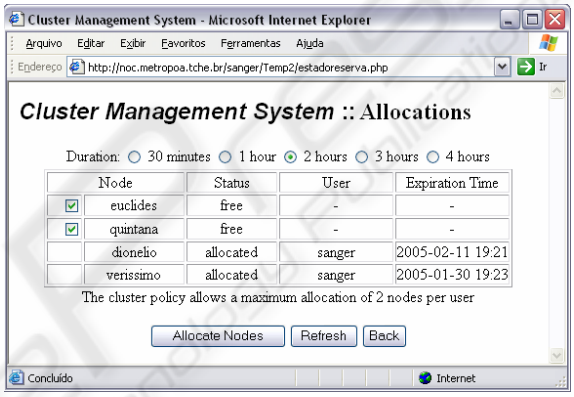

Figure 5: Multiple allocation of nodes.

The user interface is different for cluster administrator and cluster user, since the first can, for example, change the cluster allocation policy. Thus, the button "*Change Policy*" is available only for administrators. Selecting it, it is showed a new interface where it is possible to set the cluster allocation policy.

In the interface presented in Figure 4 it is also possible to obtain specific information about nodes and front-end. Clicking on this figure, an HTML page is presented with the six main categories of information offered by the management system. These categories are: hardware, running processes, temperature, resources utilization, node allocation and node disk partitions (Figure 6).

The first option is "*Hardware*". In this option, the user visualizes details of machine hardware and gets information about CPU nodes; information about memory, disks and swap memory size, and a list of the network interfaces available. Through this interface, the user obtain a global characterization of the node. These information are obtained from HostResources MIB and ClusterNode MIB.

The "*Processes*" option presents a list of user processes running in the machine. This interface presents the process name, the owner (user), the

identifier (PID), the state, the CPU usage percentage, the memory usage percentage, the time elapsed since execution started and the execution time. It is also possible to kill processes, selecting the correspondent checkbox and clicking on the "*Kill*" button. The data presented in this option are obtained from *clProcesses* in ClusterNode MIB.

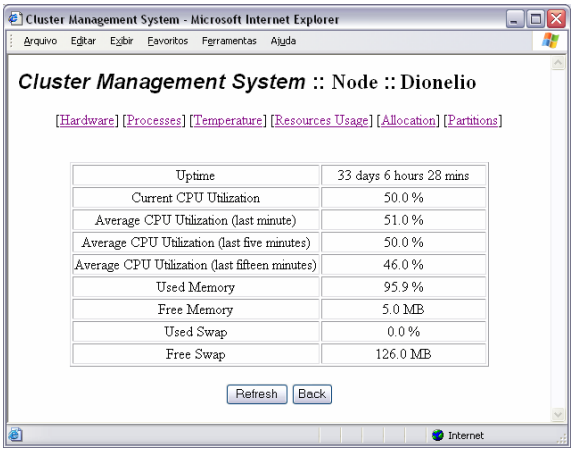

Figure 6: Interface for resource usage information.

The "*Temperature*" option presents the current temperature information of a node, the temperature threshold from which the node requires care and the maximum temperature supported by the node, obtained from the *clTemperature* group of the ClusterNode MIB. This group information is available only for nodes with hardware support for temperature sensors properly configured.

The fourth option is "*Resources Usage*". In this option, the user gets information (obtained from HostResources and ClusterNode MIBs), about the nodes resources utilization such as CPU, main memory usage, hard disk usage and swap memory usage. Considering the CPU usage, four parameters are presented: the first indicates the percentage of CPU usage in that moment and the second, the third and fourth present the average CPU utilization (*loadavg*) during the last one, five and fifteen minutes, respectively. The free main memory and free swap memory are presented in terms of percentage and megabytes.

The "*Allocation*" option deals with node allocation individually, instead of global allocation shown in Figure 5. If the node is already allocated, the user is notified about the time when the allocation will expire. Finally, in the "*Partitions*" option are presented information about active disk partitions of the computer. The interface presents, for each partition, the total length and the partition usage (kilobytes and percentage), obtained from HostResources MIB.

5 CONCLUSIONS

The management objects defined by ClusterNode and ClusterFrontEnd MIBs encloses the necessities of a cluster manager. Other necessities are covered, complementally, by MIB-II and HostResources MIB. The developed solution allows us to conclude that cluster management can be based in a network management protocol. The agent proposed allows the easy integration between cluster and network management, since it uses SNMP to access devices, and now, to access clusters as well.

The use of an SNMP proxy imposes a CPU overhead in the front-end. However, this overhead is tolerable, since the front-end, in general, has only administrative functionalities. An important argument in favor of the SNMP proxy approach is that it allows the installation of other SNMP MIBs in the nodes, without changes in the front-end agent.

Finally, the continuous cluster monitoring is a work in progress. The goal is to verify the resources usage history through polling-based management.

REFERENCES

Alves, R. S., Marquezan, C. C. and Granville, L. Z., 2004. "Experiences in the Implementation of an SNMP-Based High Performance Cluster Management System", In: *The Ninth IEEE Symposium on Computers and Communications - ISCC 2004*.

Blumenthal, U. et. al., 1998. "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", IETF, RFC 2264.

Grillo, P. and Waldbusser, S., 1993. "HostResources MIB", IETF, RFC 1514.

Harrington, D. et. al., 2002. "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", IETF, RFC 3411.

Massie, M., 2005. "Ganglia - distributed monitoring and execution system". Retrieved April 15, 2005, from http://ganglia.sourceforge.net.

Mccloghrie, K. and Rose, M., 1991. "Management Information Base for Network Management of TCP/IP-based internets: MIB-II", IETF, RFC 1213.

NET-SNMP, 2005. "The NET-SNMP Project". Retrieved April 2, 2005, from http://net-snmp.sourceforge.net.

PBS, 2005. "PBS - Portable Batch System". Retrieved April 8, 2005, from http://www.openpbs.org.

PHP, 2005. Retrieved March 12, from http://www.php.net.

Subramanyan, R. et. al., 2000. "Design and evaluation of a SNMP-based monitoring system for heterogeneous, distributed computing", School of Electrical and Computer Eng, Purdue University, TR-ECE. (00-11).

System Imager, 2005. Retrieved April 8, 2005, from http://systemimager.org.