Parallel Computing: Architectures, Algorithms and Applications
Proc. of the Intl. Conference ParCo 2007
4. - 7. September 2007
Editors: G.R. Joubert, C. Bischof, F. Peters, T. Lippert, M. Bücker, P. Gibbon, B. Mohr
to appear.

VirtuaLinux: Virtualised High-Density Clusters with no Single Point of Failure

Marco Aldinucci[1], Marco Danelutto[1], Massimo Torquati[1], Francesco Polzella[1], Gianmarco Spinatelli[1], Marco Vanneschi[1], Alessandro Gervaso[2], Manuel Cacitti[2], and Pierfrancesco Zuccato[2]

[1] Computer Science Department,
University of Pisa, Largo B. Pontecorvo 3, I-56127 Pisa, Italy
E-mail: {aldinuc, marcod, torquati, polzella, spinatel, vannesch}@di.unipi.it

[2] Eurotech S.p.A.,
Via Fratelli Solari 3/a, I-33020 Amaro (UD), Italy
E-mail: {a.gervaso, m.cacitti, p.zuccato}@eurotech.it

VirtuaLinux is a Linux meta-distribution that allows the creation, deployment and administration of both physical and virtualized clusters with no single point of failure. VirtuaLinux supports the creation and management of virtual clusters in seamless way: VirtuaLinux Virtual Cluster Manager enables the system administrator to create, save, restore Xen-based virtual clusters, and to map and dynamically re-map them onto the nodes of the physical cluster. We introduces and discuss VirtuaLinux virtualisation architecture, features, and tools. These rely on a novel disk abstraction layer, which enables the fast, space-efficient, dynamic creation of virtual clusters composed of fully independent complete virtual machines.

1 Introduction

Clusters are usually deployed to improve performance and/or availability over that provided by a single computer, while typically being much more cost-effective than a single computer of comparable speed or availability. A cluster is a network of complete computers (a.k.a. nodes), each of them running its own copy of a – possibly standard – operating system (OS). A range of solutions for the collective and/or centralized management of nodes are available in all OSes, from very low level tools (e.g. `rdist`) to complete software packages (e.g. *Sun Grid Engine*[1]).

Typically, a node of the cluster acts as the master of the cluster, while the others depend on it for several services, such as file sharing, user authentication, network routing and resolution, time synchronization. The master is usually statically determined at the installation time for its hardware (e.g. larger disks) or software (e.g. services configuration). In high-density clusters, disks mounted on nodes are statistically the main source of failures because of the strict constraints of size, power and temperature[2]. This is particularly true on master disk that happens also to be a critical single point of failure for cluster availability since it hosts services for cluster operation. A hardware or software crash/malfunction on the master is simply a catastrophic event for cluster stability.

In addition, rarely a single configuration or even a single OS can be adapted to supply all users' needs. Classic solutions like static cluster partitioning with multiple boots are static and not flexible enough to consolidate several user environments and require an additional configuration effort. Since cluster configuration involves the configuration

Parallel Computing: Architectures, Algorithms and Applications
Proc. of the Intl. Conference ParCo 2007
4. - 7. September 2007
Editors: G.R. Joubert, C. Bischof, F. Peters, T. Lippert, M. Bücker, P. Gibbon, B. Mohr
to appear.

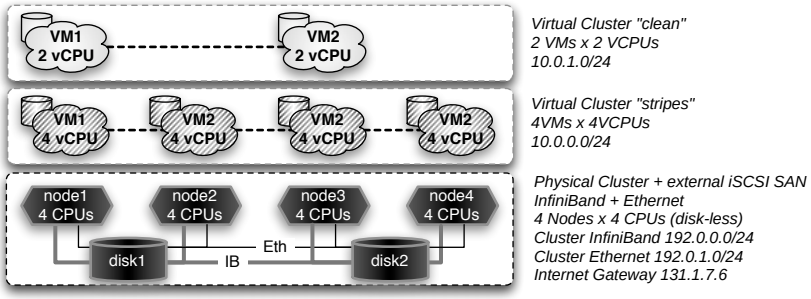

Figure 1. A physical cluster running three Virtual Clusters.

of distinct OS copies in different nodes, any configuration mistake, which may seriously impair cluster stability, is difficult to undo.

We present VirtuaLinux, an innovative Linux meta-distribution able to support the installation and the management of a disk-less, master-less cluster with a standard Linux distribution (e.g. CentOS, Ubuntu). VirtuaLinux address previously mentioned robustness problems of standard cluster installation, and natively supports Virtual Clusters (VCs). In particular, this paper presents VirtuaLinux VCs facility and the software architecture supporting them. We refer back to Aldinucci et al.[3] for a detailed description of other VirtuaLinux features, such as disk-less cluster boot, storage abstraction, and master-less services configuration.

VirtuaLinux improves the management flexibility and configuration error resilience of a cluster by means of transparent node virtualization. A physical cluster may support one or more virtual clusters (i.e. cluster of virtual nodes) that can be independently managed with no impact on the underlying physical cluster configuration and stability. Virtual clusters run a guest OS (either a flavor of Linux or Microsoft Windows™) that may differ from the host OS that handles physical cluster activities.

2 Virtual clustering

The virtualization of the physical resources of a computing system to achieve improved degrees of sharing and utilization is a well-established concept that goes back decades[4,5]. In contrast to a non-virtualized system, full virtualization of all system resources (including processors, memory and I/O devices) makes it possible to run multiple OSes on a single physical platform. A virtualized system includes a new layer of software, called a Virtual Machine Monitor (VMM). The principal role of the VMM is to arbitrate access to the underlying physical host platform resources so that these resources can be shared among multiple OSes that are guests of the VMM. The VMM presents to each guest OS a set of virtual platform interfaces that constitute a virtual machine (VM).

By extension, a Virtual Cluster (VC) is a collection of VMs that are running onto one or more physical nodes of a cluster, and that are wired by a virtual private network. By uniformity with the physical layer, all VMs are homogeneous, i.e. each VM may access a private virtual disk and all VMs of a virtual cluster run the same OS and may access a

Parallel Computing: Architectures, Algorithms and Applications
Proc. of the Intl. Conference ParCo 2007
4. - 7. September 2007
Editors: G.R. Joubert, C. Bischof, F. Peters, T. Lippert, M. Bücker, P. Gibbon, B. Mohr
to appear.

shared disk space. Different virtual clusters may coexist on the same physical cluster, but no direct relationship exists among them, apart from their concurrent access to the same resources (see Fig. 1). Virtual clusters bring considerable added value to the deployment of a production cluster because they ease a number of management problems, such as: *physical cluster insulation* and *cluster consolidation*. Physical cluster insulation ensures that crashes or system instability due to administration mistakes or cursoriness at the virtual layer are not propagated down to the physical layer and have no security or stability impact on the physical layer. Virtualization is used to deploy multiple VCs, each exploiting a collection of VMs running an OS and associated services and applications. Therefore, the VMs of different VCs may be targeted to exploit a different OS and applications to meet different user needs. The main drawback of virtualization is overhead, which usually grows with the extent of hardware and software layers that should be virtualized.

3 VirtuaLinux

VirtuaLinux is a Linux distribution that natively supports the dynamic creation and management of VCs on a physical cluster. VirtuaLinux implementation is arranged in a two-tier architecture: *VM implementation* layer and *VM aggregation* layer. The first one implements the single VM (currently the Xen[6] VMM). The second one aggregates many VMs in a VC, and dynamically creates and manages different VCs. This is realized via the *VirtuaLinux Virtual Cluster Manager* (VVCM). Overall, the VVCM enables the system administrator to dynamically create, destroy, suspend and resume from disk a number of VCs. The VCs are organized in a two-tier network: each node of a VC is connected to a private virtual network, and to the underlying physical network. The nodes of a VC are homogeneous in terms of virtualized resources (e.g. memory size, number of CPUs, private disk size, etc.) and OS. Different clusters may exploit different configurations of virtual resources and different OSes. Running VCs share the physical resources according to a creation time mapping onto the physical cluster. VCs may be reallocated by means of the run-time migration of the VM between physical nodes.

Each virtual node of a VC is implemented by a Xen VM that is configured at the VC creation time. Each virtual node includes: a virtual network interface with a private IP, a private virtual disk, a private virtual swap area and a VC-wide shared virtual storage. The virtualization of devices is realized via the standard Xen virtualization mechanisms.

3.1 VirtuaLinux Storage Architecture

VirtuaLinux uses EVMS to provide a single, unified system for handling storage management tasks, including the dynamic creation and destruction of volumes, which are an EVMS abstraction that are seen from the OS as disk devices[7,8].

The external SAN should hold a distinct copy of the OS for each node. At this end, VirtuaLinux prepares, during installation, one volume per node and a single volume for data shared among nodes. Volumes are formatted with an OS specific native file system (e.g. ext3), whereas shared volumes are formatted with a distributed file system that arbitrates concurrent reads and writes from cluster nodes, such as the Oracle Concurrent File System (OCFS2) or the Global File System (GFS).

Parallel Computing: Architectures, Algorithms and Applications
Proc. of the Intl. Conference ParCo 2007
4. - 7. September 2007
Editors: G.R. Joubert, C. Bischof, F. Peters, T. Lippert, M. Bücker, P. Gibbon, B. Mohr
to appear.

Volumes are obtained by using the EVMS snapshot facility. A snapshot represents a frozen image of a volume of an original source. When a snapshot is created, it looks exactly like the original at that point in time. As changes are made to the original, the snapshot remains the same and looks exactly like the original at the time the snapshot was created. A file on a snapshot is a reference, at the level of disk block, to its original copy.

3.1.1 Snapshots usage in VirtuaLinux

EVMS snapshots can be managed as real volumes that can be activated and deactivated, i.e. mapped and unmapped onto Unix device drivers. However, despite being standard volumes, snapshots have a subtle semantics with respect to activation due to their *copy-on-write* behavior[9]. In fact, the system cannot write on an inactive snapshot since it is not mapped to any device, thus may lose the correct alignment with its original during the deactivation period. EVMS solves the problem by logically marking a snapshot for reset at deactivation time, and resetting it to the current original status at activation time. Since snapshots cannot be deactivated without losing snapshot private data, they all should always be kept active in all nodes, even if each node will access only one of them. Snapshots on Linux OS (either created via EVMS, LVM, or other software) are managed as UNIX devices via the *device mapper* kernel functionality.

Although EVMS does not fix any limit on the number of snapshots that can be created or activated, current Linux kernels establish a hardwired limit on the number of snapshots that can be currently active on the same node. This indirectly constrains the number of snapshots that can be activated at the same time, and thus the number of nodes that VirtuaLinux can support. Raising this limit is possible, but requires a non-trivial intervention on the standard Linux kernel code. VirtuaLinux overcomes the limitation with a different approach, which does not require modifications to the kernel code. Since each snapshot is used as private disk, each snapshot is required to be accessible in the corresponding node only. In this way, each node can map onto a device just one snapshot. The status of an EVMS snapshot is kept on the permanent storage. This information is also maintained in a lazy consistent way in the main-memory of each node. Status information is read at EVMS initialization time *(evms_activate)*, and committed out at any EVMS command (e.g. create, destroy, activate, and deactivate a snapshot). While each snapshot can have just a single global state for all nodes (on the permanent storage), it may have different status on the local memory of nodes (e.g. it can be mapped onto a device on a node, while not appearing on another). Snapshot deactivation consists in unmapping a snapshot device from the system, then logically marking it for reset on permanent storage.

VirtuaLinux extends EVMS features with the option to disable EVMS snapshot reset-on-activate feature via a special flag in the standard EVMS configuration file. In the presence of this flag, the extended version of EVMS will proceed to unmap the snapshot without marking it for reset. VirtuaLinux EVMS extension preserves snapshot correctness since the original volume is accessed in read-only mode by all nodes, and thus no snapshot can lose alignment with the original. One exception exists: major system upgrades, which are performed directly on the original copy of the file system, and that trigger the reset of all snapshots. At the implementation level, the VirtuaLinux EVMS extension requires the patching of EVMS user-space source code (actually just few lines of C code).

Parallel Computing: Architectures, Algorithms and Applications
Proc. of the Intl. Conference ParCo 2007
4. - 7. September 2007
Editors: G.R. Joubert, C. Bischof, F. Peters, T. Lippert, M. Bücker, P. Gibbon, B. Mohr
to appear.

3.2 VC Networking

Xen supports VM networking via *virtualized Ethernet interfaces*. These interfaces can be connected to underlying physical network devices either via *bridged* (OSI model layer 2) or *routed* (OSI model layer 3) networking. Bridging requires less setup complexity and connection tracking overhead as compared to the routing method. On the other hand, bridging impairs insulation among different networks on the same bridge, and it lacks flexibility since it can hardly be dynamically configured to reflect the dynamic creation and destruction of VC-private networks. For this, VirtuaLinux currently adopts the routed networking.

VirtuaLinux sets up VC-private networks in a simple manner: all nodes in the VC are assigned addresses from a private network chosen at creation time, and the VC does not share the same subnet as the physical cluster. In this way, the communications among physical and virtual clusters are handled by setting up appropriated routing policies on each physical node, which acts as a router for all the VMs running on it. Routing policies are dynamically set up at the deployment time of the VM. All VMs of all VCs can be reached from all physical nodes of the cluster and each VC can access to the underlying physical network without any master gateway node. Virtual nodes of a VC are simply VMs on the same virtual subnet. However, each virtual network is insulated from the others. The routing configuration is dynamic, and has a VC lifespan. The configuration is dynamically updated in the case virtual nodes are re-mapped onto the physical cluster.

3.3 VC Disk Virtualization

Typically, VM-private disks are provided via either disk partitions or disk image files. The former method usually provides a speed edge while the latter guarantees a greater flexibility for dynamic creation of VMs. Actually, both methods require the whole root file system of the host OS as many times as the number of nodes in the VC. This leads to a very high data replication on the physical disk, a very long VC creation time. VirtuaLinux copes with these issues by means of the EVMS snapshotting technique described in Sec. 3.1. All private disks of a VC are obtained as snapshots of a single image including the VC guest OS. As discussed in Sec. 3.1.1, this leads to a VC creation time that is independent of the number of nodes in the VC (in the range of seconds) and all benefit discussed in Sec. 3.1. Once created, EVMS volumes are dynamically mounted on physical nodes according to the virtual-to-physical mapping policy chosen for the given VC.

As for the physical cluster, each VC comes with its own VC-private shared storage, which relies on OCFS2 distributed file system to arbitrate concurrent read and write accesses from virtual cluster nodes. However, since Xen does not currently enable the sharing of disks between VMs on the same physical nodes, the VC shared disk cannot be directly accessed from within virtual nodes. VirtuaLinux currently overcomes the problem by wrapping the shared storage with a NFS file system. At VC deployment time, each physical node involved in the deployment mounts the VC shared storage, which is in turn virtualized and make available to virtual nodes.

3.4 VC Management

VirtuaLinux provides two strategies for virtual-to-physical mapping of VMs: *Block* and *Cyclic*. The first one aims to minimize the spread of VMs on the physical nodes. This is

Parallel Computing: Architectures, Algorithms and Applications
Proc. of the Intl. Conference ParCo 2007
4. - 7. September 2007
Editors: G.R. Joubert, C. Bischof, F. Peters, T. Lippert, M. Bücker, P. Gibbon, B. Mohr
to appear.

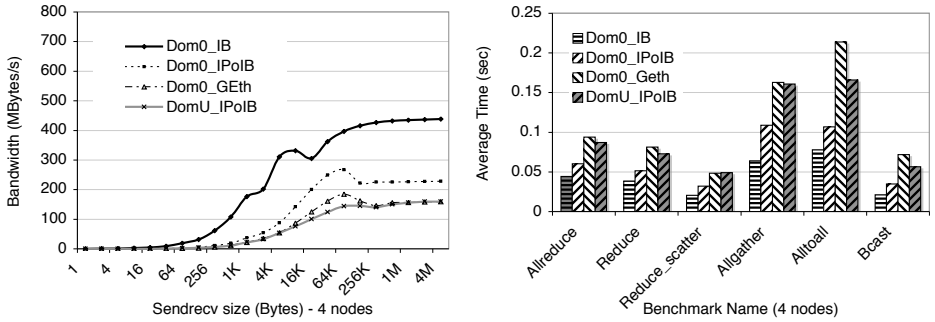

Figure 2. VirtuaLinux: evaluation of network bandwidth (left) and collective communication performance (right) with the Intel MBI Benchmarks[10]. Legend: Dom0_IB: Ubuntu Dom0, Infiniband user-space verbs (MPI-gen2); Dom0_IPoIB: Ubuntu Dom0, Infiniband IPoverIB (MPI-TCP); Dom0_Geth: Ubuntu Dom0, Giga-Ethernet (MPI-TCP); DomU_IPoIB: Ubuntu DomU, virtual net on top of Infiniband IPoverIB (MPI-TCP).

achieved by allocating on the physical node the maximum allowed number of VMs. The second one tries to spread the cluster's VM across all the cluster's physical nodes. The two strategies discussed can be coupled with two modifiers: *Strict* and *Free*. With the first modifier the deployment can be done only if there are enough free cores, with the second the constraint between the number of VM processors and physical cores is not taken into account at all. Notice that the mapping strategy of a VC can be changed after the first deployment provided it is the suspended state.

The VVCM (VirtuaLinux Virtual Cluster Manager) consists of a collection of Python scripts to create and manage the VCs. All the information about the virtual nodes such as the mapping between physical and virtual nodes and the state of each virtual machine are stored in a *database*. The information is maintained consistent between the launch of different clusters. A simple *command-line library* for the creation (*VC_Create* command), the activation (*VC_Control* command) and the destruction of the VCs (*VC_Destroy* command) is provided to the administrator. All the communications used for the staging and the execution of the VMs are implemented on top of the Secure Shell support (ssh). The *VC_Control* command relies on a simple *virtual cluster start-time support* to dynamically configure the network topology and the routing policies on the physical nodes for each virtual cluster.

4 Experiments

Experimental data presented have been collected on a 4U-case Eurotech cluster hosting 4 high-density blades, each of them equipped with a two dual-core AMD Opteron@2.2GHz and 8 GBytes of memory. Each blade has two Giga-Ethernets and one 10 Gbits/s Infiniband NIC (Mellanox InfiniBand HCA). The blades are connected with a Infiniband switch. Experimental data has been collected on two installations of VirtuaLinux: i) a testbed installation Ubuntu Edgy 6.10 with Xen 3.0.1 VMM, Linux kernel 2.6.16 Dom0 *(Ub-Dom0)* and DomU *(Ub-DomU)*; ii) a reference installation CentOS 4.4, no VMM, Linux kernel 2.6.9 *(CentOS)*.

Parallel Computing: Architectures, Algorithms and Applications
Proc. of the Intl. Conference ParCo 2007
4. - 7. September 2007
Editors: G.R. Joubert, C. Bischof, F. Peters, T. Lippert, M. Bücker, P. Gibbon, B. Mohr
to appear.

Two sets of micro-benchmarks have been used: the *LMbench* benchmark suite[11], which has been used to evaluate the OS performance, and the *Intel MBI Benchmarks*[10] with *MVAPICH MPI* toolkit (mvapich2-0.9.8)[12], which has been used to evaluate networking performance. According to LMbench, as expected, the virtualization of system calls has a non negligible cost: within both the privileged domain (Ub-Dom0) and user domain (Ub-DomU) a simple syscall pays a consistent overhead ($\sim +700\%$) with respect to the non-virtualized OS (CentOS) on the same hardware (while the difference between the privileged and the user domain is negligible). Other typical OS operations, such as fork+execve, exhibit a limited slowdown due to virtualization ($\sim +120\%$). However, as expected in a para-virtualized system, processor instructions exhibit almost no slowdown. Overall, the OS virtualization overhead is likely to be largely amortized in a real business code.

The second class of experiments is related to networking. Figure 2 left and right report an evaluation of the network bandwidth and collective communications, respectively. Experiments highlight that the only configuration able to exploit Infiniband potentiality is the one using user-space Infiniband verbs (that are native drivers). In this case, experiment figures are compliant with state-of-the-art performances reported in literature (and with CentOS installation, not reported here). Since native drivers bypass the VMM, virtualization introduces no overheads. As mentioned in Sec. 3.2, these drivers cannot be currently used within the VM (DomU), as they cannot be used to deploy standard Linux services, which are based on the TCP/IP protocol. At this aim, VirtuaLinux provides the TCP/IP stack on top of the Infiniband network (via the *IPoverIB*, or *IPoIB* kernel module). Experiments show that this additional layer is a major source of overhead (irrespectively of the virtualization layer): the TCP/IP stack on top of the 10 Gigabit Infiniband (*Dom0_IPoIB*) behaves as a 2 Gigabit network. The performance of a standard Gigabit network is given as reference testbed (*Dom0_GEth*). Network performance is further slowed down by user domain driver decoupling that require data copy between front-end and back-end network drivers. As result, as shown by *DomU_IPoIB* figures, VC virtual networks on top of a 10 Gigabit network, exhibits a Giga-Ethernet-like performances. Results of extensive testing of VirtuaLinux can be found in Aldinucci et al.[3]

5 Conclusions

VirtuaLinux is a novel Linux meta-distribution aiming at installation and the management of robust disk-less high-density clusters. Among all features of VirtuaLinux, this paper has introduced virtual clustering architecture, features, and tools. These enable the dynamic and seamless creation and management of ready-to-use VCs on top of Xen VMM. Both the physical cluster and the VCs can be installed with a number of pre-defined OSes (e.g. Ubuntu Edgy 6.10 and CentOS 4.4) or easily extended to other Linux distributions. VCs managing tools can be easily extended to manage almost any guest Linux distribution by providing VC tools with a tarball of the OS and a simple configuration file.

VirtuaLinux introduces a novel disk abstraction layer, which is the cornerstone of several VirtuaLinux features, such as the time and space efficient implementation of virtual clustering. Preliminary experiments show that VirtuaLinux exhibits a reasonable efficiency, which will naturally improve with virtualization technology evolution. VirtuaLinux is currently distributed with Eurotech HPC platforms. In this regard, Eurotech laboratory

Parallel Computing: Architectures, Algorithms and Applications
Proc. of the Intl. Conference ParCo 2007
4. - 7. September 2007
Editors: G.R. Joubert, C. Bischof, F. Peters, T. Lippert, M. Bücker, P. Gibbon, B. Mohr
to appear.

experienced a tenfold drop of clusters installation and configuration time. To the best of our knowledge, few existing OS distributions achieve the described goals, and none achieve all of them.

Acknowledgements and Credits

VirtuaLinux has been developed at the Computer Science Department of the University of Pisa and Eurotech S.p.A. with the partial support of the initiatives of the LITBIO Consortium, founded within FIRB 2003 grant by MIUR, Italy. VirtuaLinux is a open source software under GPL available at `http://virtualinux.sourceforge.net/`. We are grateful to Peter Kilpatrick for his help in improving the presentation.

References

1. Sun Microsystems, *Sun Grid Engine*, 2007, `http://gridengine.sunsource.net/`.
2. Eduardo Pinheiro, Wolf-Dietrich Weber, and Luiz Andrè Barroso, *Failure Trends in a Large Disk Drive Population*, in: Proc. of the 5th USENIX Conference on File and Storage Technologies (FAST'07), pp. 17–28, San Jose, CA, USA, Feb. 2007.
3. Marco Aldinucci, Massimo Torquati, Marco Vanneschi, Manuel Cacitti, Alessandro Gervaso, and Pierfrancesco Zuccato, *VirtuaLinux Design Principles*, Tech. Rep. TR-07-13, Università di Pisa, Dipartimento di Informatica, Italy, June 2007.
4. Robert P. Goldberg, *Survey of Virtual Machine Research*, Computer, pp. 34–45, June 1974.
5. Mendel Rosenblum, *The Reincarnation of Virtual Machines*, Queue, **2**, no. 5, 34–40, 2005.
6. Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield, *Xen and the art of virtualization*, in: Proc. of the 9th ACM Symposium on Operating Systems Principles (SOSP'03), pp. 164–177, ACM Press. 2003.
7. EVMS website, *Enterprise Volume Management System*, 2007, `http://evms.sourceforge.net/`.
8. Steven Pratt, *EVMS: A Common Framework for Volume Management*, in: Ottawa Linux Symposium, 2002, `http://evms.sourceforge.net/presentations/evms-ols-2002.pdf`.
9. IBM, *Understanding and exploiting snapshot technology for data protection*, 2007, `http://www-128.ibm.com/developerworks/tivoli/library/t-snaptsm1/index.html`.
10. Intel Corporation, *Intel MPI Benchmarks: Users Guide and Methodology Description*, ver. 3.0 edition, 2007, `http://www.intel.com/cd/software/products/asmo-na/eng/cluster/clustertoolkit/219848.htm`.
11. Larry McVoy and Carl Staelin, *LMbench: Tools for Performance Analysis*, ver. 3.0 edition, Apr. 2007, `http://sourceforge.net/projects/lmbench/`.
12. The Ohio State University, *MVAPICH: MPI over InfiniBand and iWARP*, 2007, `http://mvapich.cse.ohio-state.edu/overview/mvapich2/`.