Deploying MPI applications in a cluster of Xen VMs: A Networking Perspective

Anastassios Nanos, Georgios Goumas and Nectarios Koziris
{ananos,goumas,nkoziris}@cslab.ece.ntua.gr

Computing Systems Laboratory
National Technical University of Athens

Motivation

Nowadays, seeking optimized data paths that can increase I/O throughput in Virtualized environments is an intriguing task, especially in a high performance computing context. We try to address this issue by evaluating methods for optimized network device access using scientific applications and microbenchmarks. We examine the network performance bottlenecks that appear in a Cluster of Xen VMs using both generic and intelligent network adapters. We study the network behavior of MPI applications. Our goal is to:

(a) explore the implications of alternative data paths (direct or indirect) between applications and network hardware and

(b) specify optimized solutions for scientific applications that put preasure on network devices.

Preliminary results show that a combination of these techniques is essential for scientific applications to achieve near-native performance in VM environments.

Process Placement

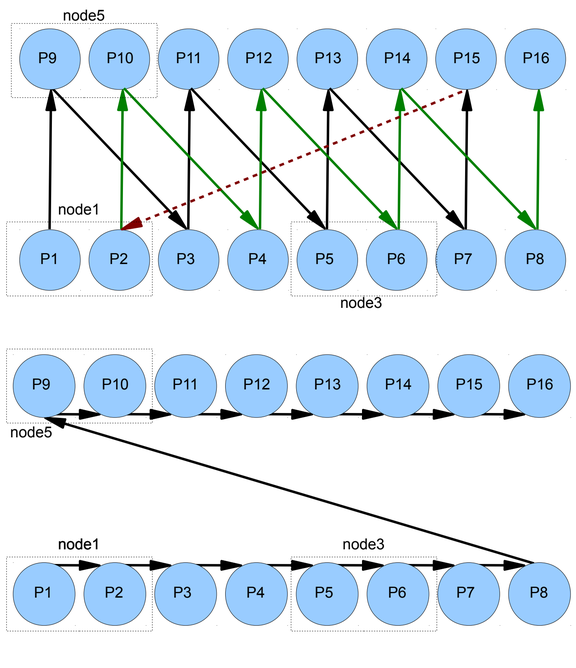


Figure 3: Process Placement in Physical Machines and VMs (top: case (a), bottom: case (b))

Network Configuration (BRIDGED)

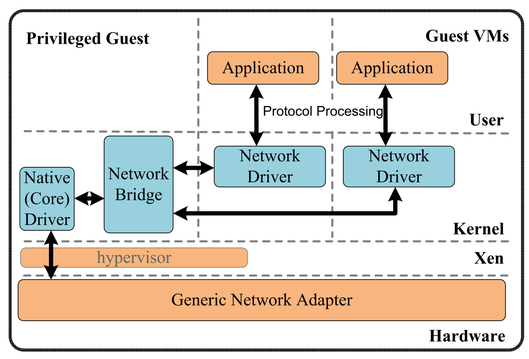


Figure 1: BRIDGED Configuration

This is the default configuration provided by Xen. All guest VMs share a common bridge, setup by the privileged guest. Data flow from applications to the privileged guest via copying or page flipping and thus the software bridge becomes a bottleneck in an HPC context.

Network Configuration (IOV)

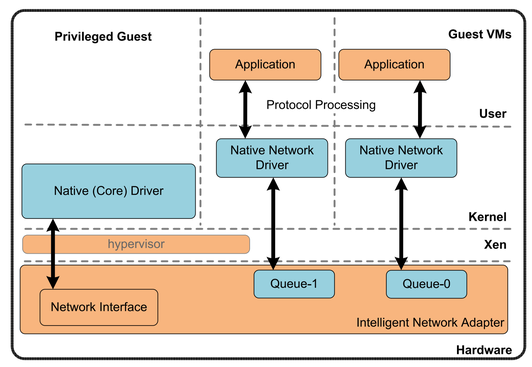


Figure 2: IOV Configuration

The Network Adapter exports Virtual Functions that are setup by the Privileged Guest. Each of these Functions provide Virtual Interfaces that are assigned to a guest VM and as a result, a direct data path is installed between applications and the network hardware (I/O Virtualization).

Application

Computes an advective process in a XxYxZ space for a time window T [4]. We choose a fixed grid size ($512x512x512$, $T = 512$), distributing X, Y or Z dimension across all processes (16 total processes).

Deploying the Application

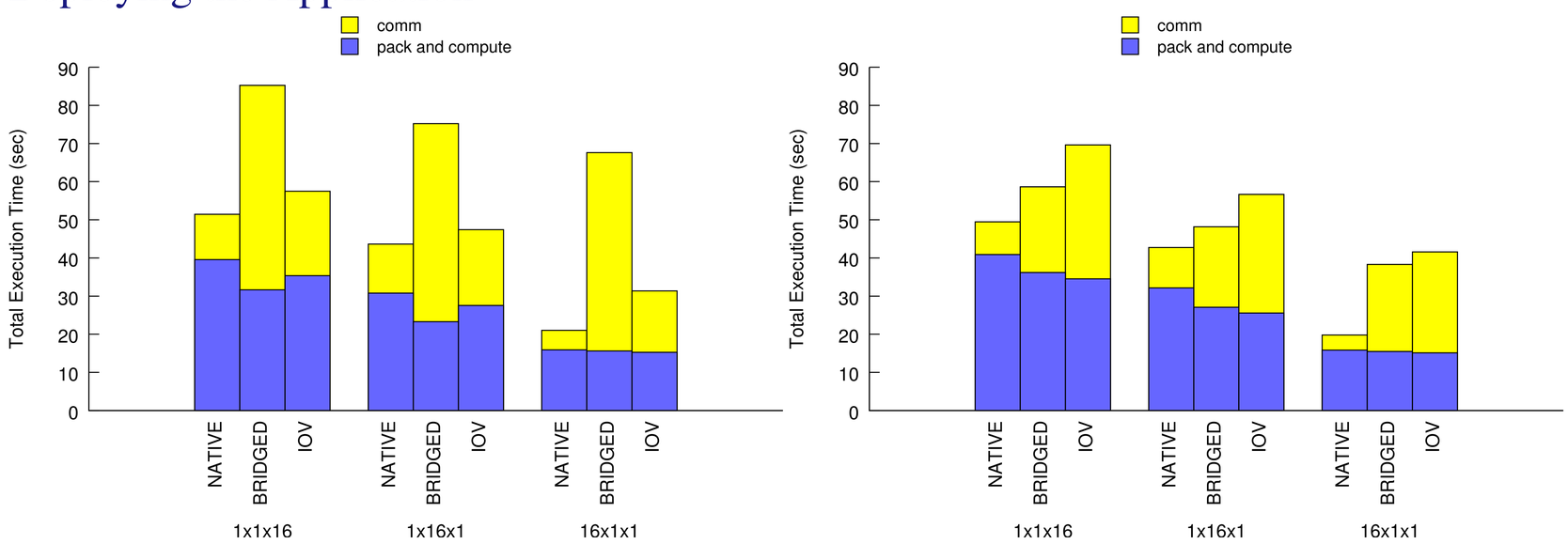
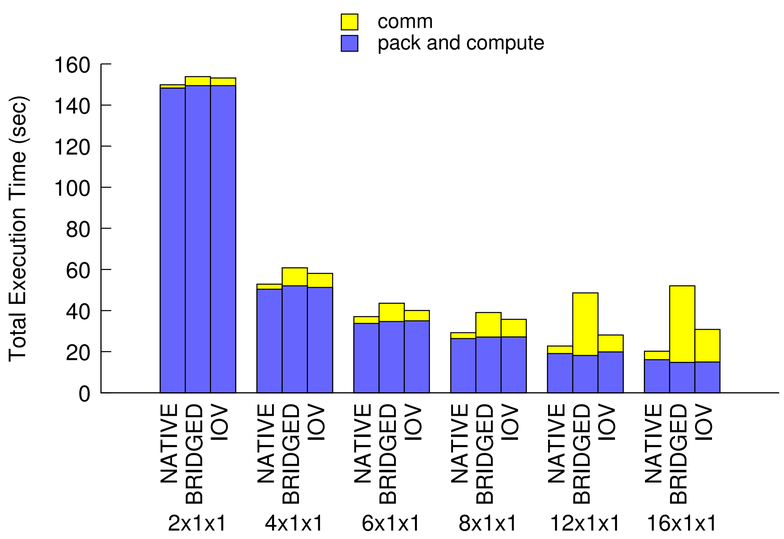


Figure 4: Total Execution Time (left: placement case (a), right: placement case (b))

Scaling



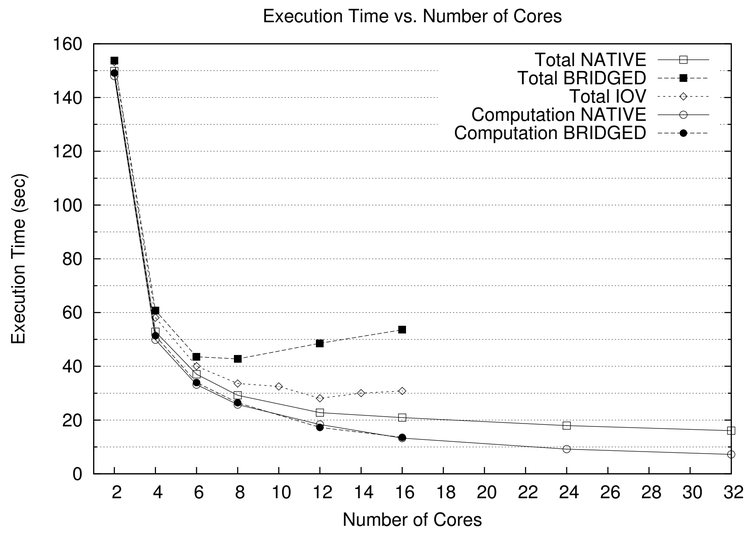


Figure 5: Execution Time Breakdown for 2. . .16 cores and scaling (placement case (a)).

This Figure presents the execution time breakdown for the $\{X\text{x}Y\text{x}Z\} = \{2 \ldots 16\text{x}1\text{x}1\}$ process distribution using the first communication pattern (case (c)). In the BRIDGED case (2^{nd} bar), the negative scaling factor is obvious as we add cores to the application. This negative factor is due to the communication part of the execution (light part); the computation part (dark part) remains constant. On the other hand, the IOV case follows the scaling pattern of the NATIVE case, with a constant overhead due to virtualized communication layers.

Speedup

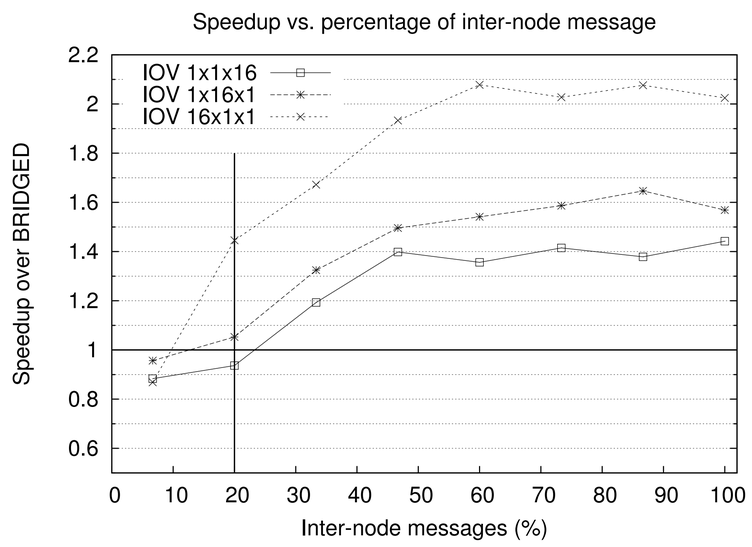


Figure 6: Speedup over BRIDGED vs. % of inter-node messages

We observe that when 50% of MPI operations traverse the network, IOV outperforms the BRIDGED case by at least 40%. The only case where one should choose the BRIDGED case, is when network operations are lower than 20% of all MPI operations.

Conclusions

- present preliminary performance evaluation results of a real scientific application running in a cluster of Xen VMs
- demostrate the need for profiling application behavior prior to deploying HPC applications in Virtualized environments
- explore alternative data paths for network communication between HPC applications that run on clusters of VMs
- show that for a given parallel HPC application, its communication pattern has to be examined before placing processes in VMs
- justify that the computation part of the application execution is not altered when migrating to a VM environment
- suggest HPC applications *can* be executed in VM environments with very little overhead, provided that their communication pattern is examined and that all parallel processes are distributed in a way that data flow through the optimum ad-hoc data-path (direct or indirect)

Future Work

- evaluating message passing using shared memory techniques when processes co-exist in VM containers
- evaluating higher-level frameworks for application parallelism based on MapReduce and its extensions in VM execution environments

References

1. PCI SIG: SR-IOV (2007) http://www.pcisig.com/specifications/iov/single_root/.
2. Nanos, A., Koziris, N.: MyriXen: Message Passing in Xen Virtual Machines over Myrinet and Ethernet. In: 4th Workshop on Virtualization in High-Performance Cloud Computing, The Netherlands (2009)
3. Youseff, L., Wolski, R., Gorda, B., Krintz, C.: Evaluating the Performance Impact of Xen on MPI and Process Execution For HPC Systems. In: 1st Intern.Workshop on Virtualization Technology in Distributed Computing. VTDC 2006.
4. Goumas, G., Drosinos, N., Koziris, N.: Communication-Aware Supernode Shape. IEEE Transactions on Parallel and Distributed Systems 20 (2009) 498511