ISSN 1330-3651(Print), ISSN 1848-6339 (Online)
DOI: 10.17559/TV-20140819115232

MPI APPLICATIONS' PERFORMANCES IN NATIVE VS VIRTUALIZED ENVIRONMENTS USING INFINIBAND IPOIB VIRTUALIZATION AND LIVE MIGRATION

Marko Kobal, Zlatan Car, Milan Ojsteršek

Original scientific paper
Over recent years we have seen a constant evolution of High-Performance Computing moving into commodity servers cluster types of systems. At the same time we have seen Cloud Computing being a proven, stable and reliable IT paradigm, utilising virtualized clusters of commodity servers as key components. This paper presents a face-to-face native-to-virtual HPC MPI application performance analysis on a range of general-use HPC applications by virtualizing InfiniBand via IPoIB at the guest virtual machine level on VMware ESXi as well as KVM hypervisor with production ready live migration support. Applications running on various small and medium HPC clusters are targeted to address users, considering full IT infrastructure virtualization by merging general purpose with HPC IT infrastructure. Live migration is a key component within a virtualized environment as it enables dynamic virtual IT infrastructure, therefore live migration performance impact is evaluated through a pluggable multi-platform architecture on a proactive fault-tolerance use case.

Keywords: *cloud computing; HPC; InfiniBand; IPoIB; live migration; MPI application; performance analysis; virtualization*

Izvorna svojstva MPI aplikacije prema virtualnom okruženju uporabom InfiniBand IPoIB virtualizacije i aktivne migracije

Izvorni znanstveni članak
Zadnjih je godina vidljiv ulazak High-Performance Computing u klaster tipove robnih servera sustava. U isto vrijeme svjedočimo računarstvu u oblacima kao dokazanoj, stabilnoj i pouzdanoj IT paradigmi koja rabi virtualizirane klastere robnih servera kao ključne komponente. U radu se daje analiza svojstava kod primjene direktne prirodne u odnosu na virtualnu HPC MPI aplikaciju na području HPC aplikacija opće uporabe virtualizacijom InfiniBanda via IpoIB na nivou virtualnog stroja na VMware ESXi kao i KVM hypervisora uz podršku aktivne migracije spremne na proizvodnju. Aplikacije za razne male i srednje HPC klastere usmjerene su na korisnike, imajući u vidu potpunu virtualizaciju IT infrastrukture povezujući infrastrukturu opće namjene s HPC IT infrastrukturom. Aktivna migracija je ključna komponenta u okviru virtualiziranog okruženja budući da omogućuje dinamičku virtualnu IT infrastrukturu. Stoga se djelovanje svojstava aktivne migracije procjenjuje kroz arhitekturu s više platformi na slučaju proaktivne uporabe greška-tolerancija.

Ključne riječi: *aktivna migracija; analiza svojstava; aplikacija MPI; HPC; InfiniBand; IpoIB; računarstvo u oblaku; virtualizacija*

1 Introduction

The diversities of HPC applications range from shared-memory, distributed memory, PGAS (Partitioned Global Address Space) and other architecture-dependent and architecture-independent types of applications. Nevertheless, the majority of currently widely-used distributed memory applications are based on MPI (Message Passing Interface) [1]. High speed interconnect is crucial for efficient execution of MPI applications - and InfiniBand is today's high speed interconnect of choice. Virtualizing a system for MPI applications also requires virtualizing high speed interconnection which is not a trivial task. While 10GbE (10-gigabit Ethernet) and the emerging 40GbE are also valid options, we focus on InfiniBand interconnect as it is already present in many HPC systems. There are several options for virtualization of an InfiniBand network, however, most of them impose limitations or do not support some of the advanced virtualization features on production-ready platforms, especially live migration - we discuss these issues in Section 4. Virtualization of InfiniBand network, based on IpoIB [2] provides production-ready support on different platforms without additional modifications and preserves hypervisor's advanced features such as live migration. This paper discusses the technological details behind IPoIB virtualization and presents performance analysis with results that should answer the question whether running MPI applications on a virtualized system with IPoIB virtualization yields an acceptable performance. This would enable seamless transformation of HPC clusters into virtualized environments, merging general purpose with HPC IT infrastructure and would help users to decide whether cloud solutions, based on IPoIB virtualization would be appropriate for running MPI applications. Our performance analysis methodology, application selection and testbed environment configuration can also be used in future work to evaluate other modes of InfiniBand virtualization and compare results.

This paper focuses on performance analysis for widely used general-use MPI applications. With widely used general-use we are referring to applications that are used on a day-to-day basis, in academia, public services as well as in enterprise and SME industry and are present in hundreds of HPC systems all over the globe. The global HPC market is facing fast adoption of HPC within the SME industry with strong funding from international and especially EU projects such as e.g. FORTISSIMO [3], therefore the need for consolidation of IT infrastructure within a single multi-purpose virtualized environment is a fast emerging concept. That said we are especially, yet not exclusively, focusing on a private virtualized HPC system; the results of our analysis can also be used by a public HPC cloud provider. We have developed performance analysis methodology that covers all aspects of native to virtual transformation: from flexible deployment to programmable management and multi-platform orchestration.

To be able to address a wide range of users we used two of the more common virtualization platforms, commercial VMware ESXi [4] and open source KVM[5]

hypervisor. VMware has currently the biggest market share between commercial virtualization platforms [6] and KVM is the default and preferred hypervisor within many open-source operating systems, including CentOS and other Red Hat Enterprise Linux derivatives.

This paper is organized as follows. Section 2 presents the related work, Sections 3 and 4 discuss the motivation of using virtualization with HPC and an overview of the current InfiniBand virtualization options. Our extensive performance analysis is presented in Section 5. In Section 6 we propose a pluggable implementation of multi-platform application-transparent, proactive fault-tolerance as one of the highlighted features of virtualized HPC environments. This feature is used as the evaluation of live migration impact. Finally, Section 7 presents our conclusions.

2 Related work

In the related work various HPC virtualization benchmarking, case studies and performance analysis are presented and can be divided into three major groups: 1) related to public cloud offering, 2) applicable to a particular virtualization technology or private virtualized infrastructure and 3) general common studies. General HPC virtualization studies, e.g. [7] cover general benefits and pitfalls related to High-Performance Cloud Computing outlining, many still open up questions such as scalability issues and performance variations depending on a particular application. Work, related to public cloud offering, most often discusses Amazon EC2. Early benchmarks reports low reliability and performance, outlining network performance issues in [8] with further analysis in [9] showing improved but still low performance compared to native systems. Other studies evaluating HPC in the Cloud [10, 11] highlight the benefits of on-demand and elastic resources and on the other hand the drawbacks of non-optimal performances, mainly due to interconnect limitations, brought by a virtualization layer and in some cases due to lower performance servers, used in general-use public cloud environments.

Case studies and performance analysis on various virtualization technologies mainly deal with synthetic benchmarking. Very efficient virtualization can be achieved in single node performance, resulting in acceptable performance for embarrassingly parallel HPC applications (where there is low or no dependency − or communication − between parallel tasks) and poor performance for message-passing-based applications with heavy inter-node communication when an interconnection network is inefficiently virtualized or limited to Gigabit Ethernet [12]. Multi-platform performance analysis in [13] reports KVM as one of the more promising open source hypervisors. Due to many performance issues with a high-speed network, many alternatives for InfiniBand virtualization have been proposed [14 ÷ 18]. However, these alternatives either have limitations of advanced features such as live migration or have not (yet) been included in production-ready versions of hypervisors.

This paper extends the related work with native to virtual performance analysis where research is not limited to synthetic benchmarks but also evaluates the performances of general-use applications to address real world application usage which has not been thoroughly addressed in previous work. Finally IPoIB virtualized InfiniBand is used with live migration support -we believe that this is the first extensive performance analysis of general-use MPI HPC applications on InfiniBand IPoIB virtualization with production ready live migration support. Furthermore we discuss the benefits of a virtualized HPC system with live migration support on a fault-tolerance use case.

3 Motivation for HPC system virtualization

Server virtualization is already a proven and globally accepted standard approach for IT resource provisioning in mainstream computing. With mainstream computing we are referring to the web, business, database and other generally used applications. Many HPC application users already use virtualization as a key component in their mainstream computing IT infrastructure. However, the vast majority of HPC users still use bare-metal (native) provisioning for their HPC systems. It makes sense for an HPC system to deliver maximum performance and squeeze the last bit of power from every single component, whereas virtualization, without any doubt, brings inevitable performance overhead and even some restrictions on hardware usage. The benefits and caveats of running HPC applications within a virtualized environment is discussed in [19].

In the light of virtualization advantages we would like to point out three important concepts which need to be taken into account when dealing with today's increasingly demanding IT: 1) consolidation, 2) overall utilisation and 3) ease of management for an organisation when dealing with its IT infrastructure. Mixed native and virtual environments are harder to maintain and provide non-ideal utilisation. Our motivation is to find out whether users of widely-used MPI HPC applications could virtualize their existing HPC clusters or migrate to new virtualized environment and thus use a global, unified virtualized environment for general use as well as for HPC computing. This would be a key enabler for combining both HPC and other general use computing infrastructures into a single software defined datacentre. A unified virtualized system would also bring better energy efficiency and power savings. The same hardware could be used for both HPC and other applications thus enabling us to schedule workloads depending on the current needs. Another important aspect would be that today's virtualization software is very diverse. There are many commercial providers of virtualization software (market leaders with highest market share) and also many open-source solutions enabling low-cost virtualization deployments. As mentioned before we evaluated commercial (VMware ESXi) as well as open source virtualization software (KVM) to cover different type of users.

We set some prerequisites for our performance analysis. One of the key components of virtualization advanced techniques that enables ease of management, dynamic utilisation and optionally enables proactive fault tolerance, is virtual machine live migration [20]. Therefore it should be fully supported even when running

HPC applications on the virtualization platform. The other obvious prerequisite comes with interconnect bandwidth and latency. Our focus was on MPI applications, running on distributed memory clusters. These kinds of applications require high-speed, low-latency interconnect. At a minimum a 10-gigabit Ethernet or preferably QDR or FDR InfiniBand is a standard in modern distributed memory clusters, supporting MPI applications. Virtualization software should support virtualization of an InfiniBand adapter whilst maintaining high-throughput and low-latency with the ability to do live migration of the virtual machine. As we are targeting production-ready virtualization platforms, these features should be available out-of-the-box in current production versions. Of course, as mentioned before, virtualization brings inevitable performance overhead therefore it is to be expected that the penalties should be significant when virtualizing a high-performance interconnect with drivers' translation techniques, such as IPoIB virtualization.

4 Methods of virtualizing InfiniBand

There are four main concepts for virtualizing an InfiniBand adapter: PCI Pass-through, SR-IOV, Para-Virtualization and IPoIB translation. **PCI Pass-through** grants a guest operating system direct access to a dedicated InfiniBand device. Because each native adapter card is used exclusively by a single OS, severe limitations are applied to this virtual machine, most notably the inability of live migration. Edwin Zhai, et al. presented live migration with a pass-through device for Linux VM [21], however this experimental work was excluded from the later production versions of Linux based hypervisors. **Single Root I/O Virtualization** (SR-IOV) [14] allows a PCI Express device to appear as multiple, separate devices, called Virtual Functions (VF). Each guest operating system may gain exclusive access. Again, support for some features is limited and live migration is not supported in production versions of widely-used open-source virtualization software, or in VMware. **Para-Virtualization** is the most suitable concept for virtualizing the InfiniBand adapter. If allows the creation of a virtualized RDMA device that supports RDMA semantics and provides complete transparency to guest operating systems thus allowing live migration and other features. Currently there is no production implementation of a para-virtualized InfiniBand adapter with the exception of Microsoft Hyper-V, which is currently only supported on Microsoft's implementation of the Message Passing Interface (MS-MPI), limited to Microsoft operating systems. Similar to this approach experimental work has been done for the VMM-bypasses (OS-bypasses) [18] virtualization technique, however this prototype has not been used in production grade hypervisors. This brings us to the IP over InfiniBand **IPoIB virtualization** method. The IPoIB driver allows a spanning IP network on top of an InfiniBand high-speed network. This allows guest operating systems to use a para-virtualized high-speed Ethernet adapter by translating network communication from a guest high-speed para-virtualized Ethernet adapter to IPoIB interface on native host and further down to InfiniBand network devices. As para-virtualization is used, live migration and other features are fully supported. This mode is supported in current production versions of KVM, Xen and VMware, therefore we chose to evaluate the performance of MPI applications when running on top of IPoIB virtualized InfiniBand.

4.1 IPoIB for virtualization

IPoIB is IP encapsulation over InfiniBand as described in RFC 4391/4392. It provides IP services over InfiniBand fabric. The benefits of IPoIB are that it acts as a data-link within the TCP/IP stack; it allows users to run IP-based unmodified applications on InfiniBand and supports network interface offloads. However, IPoIB also has limitations – it is limited to IP applications only and as such does not provide any Ethernet Header encapsulation. The IPoIB network interface itself does not act as a standard Ethernet Network Device and cannot be used in fully-virtualized or para-virtualized environments. This is why an additional layer has to be added within the network flow from undelaying a physical device to the guest virtual machine para-virtual network adapter. VMware ESXi IPoIB drivers' registers "vmnic" interfaces. It can then be attached to the virtual switch and used as a "vmxnet" para-virtualized network device inside a guest operating system. Whilst implementation details for VMware ESXi are not disclosed due to its closed-source nature we describe the implementation in Linux. A new eIPoIB "shim layer" (a small library that transparently intercepts API calls and changes the arguments passed, handles the operation itself, or redirects the operation elsewhere) has been introduced by Mellanox [22]. eIPoIB creates a new Ethernet Network Device over IPoIB interface, managed by an eIPoIB kernel module and register a standard Ethernet Interface into the Operating System. The basic eIPoIB frame flow is shown in Fig. 1.

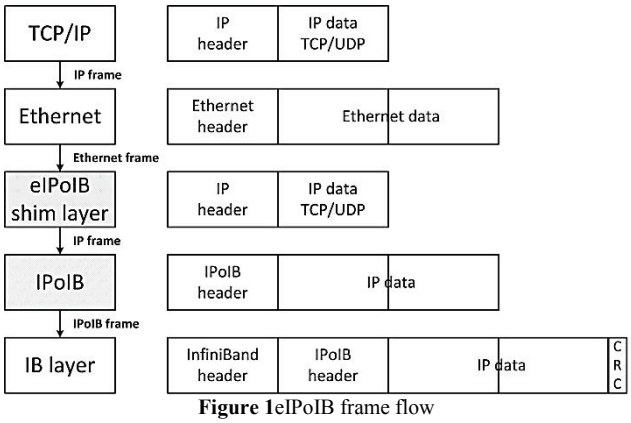

Figure 1eIPoIB frame flow

The eIPoIB driver creates an eIPoIB interface which is then assigned to the Linux bridge network device and later attached to a virtual machine as the "virtio" para-virtual network device [23].

Virtual machine network device to IPoIB InfiniBand translation is done at the driver level therefore this mode fully supports live migration of virtual machines using an IPoIB para-virtualized high-speed Ethernet adapter. This kind of virtualized InfiniBand network retains low latency but also reduces throughput and brings TCP/IP stack

overhead. We discuss the network's performance implications and test the results in Section 5.

5 Performance analysis

Our performance analysis methodology was based on the boundary conditions set by the requirements of MPI applications where network interconnect performance is crucial. Firstly, we performed bandwidth and latency tests, followed by HPL Linpack synthetic benchmarking [24] and lastly we evaluated the performances of three general-use MPI applications. We identified three fields of general-use MPI applications: weather prediction application as an example of public service, molecular dynamics application as an example of a widely-used scientific tool and the CFD software package as a widely-used application throughout Enterprise and SME industries. All applications were compute and memory intensive, but were not heavily storage I/O intensive, thus any storage performance evaluation could be left out. It is presumed that storage performance is sufficient for application needs and affects performance in native environments in the same way as in virtual environments and as such did not become a factor for our performance analysis. ALADIN/ALARO (version 36t1) was selected for weather prediction application. Aladin [25] is a high resolution numerical weather prediction project. It provides software application itself as well as numerical models as the basis for the forecasting tools of modern meteorology. With its enviable development over the last two decades it has become a standard and essential tool for weather prediction in many European and North African countries. We used real test datasets provided by the Slovenian Environment Agency and ran the integration of meteorological models as benchmark. The integration was done in steps, where full integration was done in N steps. The average time to complete one step was measured. For the molecular dynamic benchmarks we used GROMACS [26] (version 4.6.3) − a well-known versatile package for performing molecular dynamics for systems with hundreds to millions of particles. We used a real-life example on a DPPC membrane system from GROMACS benchmark suite and measured time to complete one simulation with 60,000 steps. Open FOAM [27] (version 2.1.1) was chosen as the leading open source Computational Fluid Dynamics (CFD) software. Open FOAM has a large user base across most areas of engineering and science, from both commercial and academic organisations and is also used by many SMB enterprises. A parallel lid-driven cavity 3D flow simulation example (provided by OpenFOAM package) was used as the benchmarking example and the runtime in seconds was measured. All benchmark datasets, synthetic and real-life are publically available thus the test can be easily repeated in other scenarios for further work.

5.1 Testbed

All tests were performed on the exact same hardware to avoid any potential hardware-related implications − the same compute nodes were transformed into native, KVM and ESXi, respectively. The test bed consisted of standard HPC components that can be found in the vast majority of small and medium (also large) HPC clusters, based on distributed memory architecture. We used eight native compute nodes (servers); each configured with dual Intel Xeon X5650 2.66 MHz 6-core CPUs, 32 GB main memory, a single hard drive, and a Mellanox QDR InfiniBand adapter. We used three networks, also quite common; one GbE for administration, one GbE dedicated to live migration and InfiniBand QDR for the MPI communication network. Dynamic and programmable infrastructure is crucial for a successful multi-platform performance analysis, therefore we prepared centralised deployment and management of operating systems and middleware using Cobblerand Puppet [28]. This allowed us to make efficient and unified configuration changes across all tests and across all platforms, native and virtual servers. Fig. 2 outlines our performance analysis methodology and test bed architecture design.

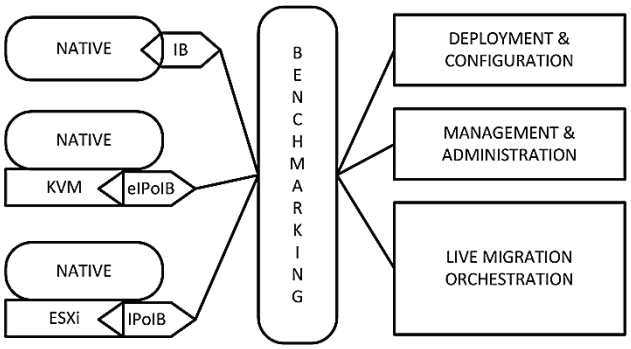

Figure 2 Performance analysis methodology and test bed architecture design

Cobbler deployment settings, Puppet recipes and application configurations were centrally managed and stored which enabled us to easily repeat system reconfigurations or reinstallations in a programmatic manner for further performance analysis on different platforms with the same settings and configurations. As virtualization brings memory overhead and the hypervisor itself requires some memory, we limited the operating system to 24 GB of usable main memory (kernel option mem=26,500M) when running benchmarks in native mode. When running in virtual mode, VM's were configured using the same configuration as in the native mode: each virtual server with 12 virtual CPUs and 24 GB allocated memory (thus leaving enough memory for the hypervisor and optional virtualization overhead). Network interfaces were configured in para-virtualized mode and InfiniBand adapter via IPoIB virtualization, as described in Section 4.1. The operating system of choice was CentOS 6.5 64 bit as all our test applications were fully supported on this OS. The native tests were done on native nodes running CentOS 6.5 64 bit. The KVM virtualization environment was configured on top of the native CentOS 6.5 64 bit based on kernel 2.6.32 together with libvirt 1.2 and QEMU 0.12. The VMware virtualization environment was based on ESXi 5.1. The guest operating system in the virtual machines, running on top of KVM and ESXi was configured within the same environment as the native nodes' using Puppet configuration management. All applications were compiled from source with default settings, no optimisations had been applied, in most cases SSE 4.1

was the highest processor optimisation used by the default compile environment. For the best cross-compatibility across different applications we used OpenMPI 1.6.

5.2 Performance analysis results

Latencies were measured over a range of message sizes using the OFED's[29] ib_send_lat (Send/Receive) utility with polling completions for native InfiniBand (*ib_send_lat --iters=10000 -a --tx-depth=300 -c RC --ib-port=1 target-host*). MPI over the virtualized IPoIB Ethernet adapter uses TCP as its transport; therefore we used the netperf TCP_RR (TCP Request/Response) test for the virtualized InfiniBand adapter (*netperf -H target-host -t TCP_RR -- -r 1*). In the following figures "native GbE" and "native IB" represent benchmarks in a native (bare metal) environment, "esx IPoIB" represents benchmarks in VMware ESXi IPoIB virtualized environment and "kvm eIPoIB" represents benchmarks in a KVM IPoIB virtualized environment.

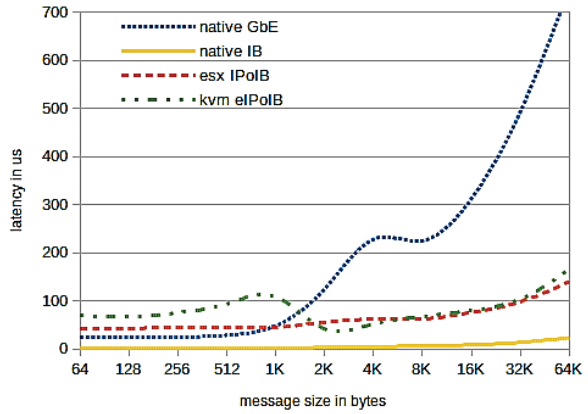

Figure 3 Latency benchmark (lower is better)

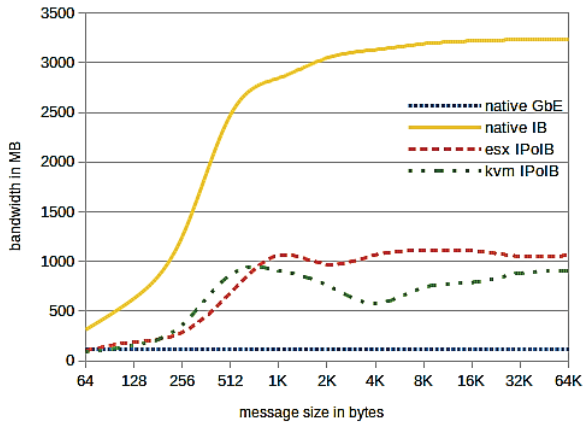

Figure 4 Bandwidth benchmark (higher is better)

IPoIB translation provides satisfactory low latencies for larger message sizes, but not for smaller message sizes. Poor latency performance at small message size, combined with higher TCP/IP stack overhead when dealing with high volume small messages suggests lower performance in MPI applications.

Bandwidth was measured over a range of message sizes using the OFED ib_write_bw (Send/Receive) with polling completions for native InfiniBand (*ib_write_bw --iters=10000 -a --tx-depth=300 -c RC --ib-port=1 target-host*) and with a netperf bulk data transfer performance test for the virtualized InfiniBand adapter (*netperf -H target-host -- -m 65536 -M 65536*). As expected, bandwidth is significantly reduced within virtualized environments compared to native InfiniBand. We also noticed an unusual drop in performance for bandwidth in KVM of around 4K message size which is probably related to the eIPoIB drives issue, because we noticed the same pattern in multiple test runs. We also included a test on native GbE performance for comparison.

HPL benchmark [24] is a very good tool for maxing-out the memory usage and thus taking advantage of the InfiniBand high throughput capability because of heavy MPI communication load.

In our results the measured HPL GFLOPS speed in single node configuration averaged only a 1,7 % decrease for virtualized mode which confirms very good CPU and memory efficiency within virtualized environments [30]. The performances of HPL on ESXi and KVM were significantly worse than the one seen in the native environment when we scaled up. Besides the orders of magnitude lower bandwidth capability of the virtualized IPoIB network, there is also the TCP/IP overhead which requires significant system resources. CPU cycles, spent for TCP/IP processing cannot be used for application processing which led to an increasing drop in performance as we were adding more nodes. Similar results were obtained in related work [10] on 10 GbE virtualized interconnects.

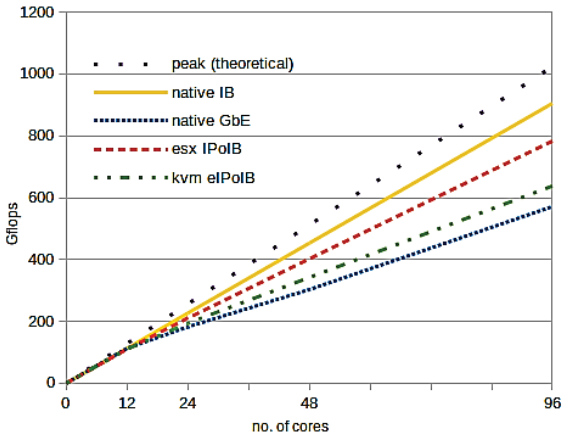

Figure 5 Linpach HPL benchmark

GROMACS application on the DPPC membrane system uses a less intensive MPI communication pattern. As shown in Fig. 6 the DPPC benchmark, as well as the Aladin model integration application, scaled well on the virtualized platform, although the performance overhead was significant. The results for OpenFOAM showed the most significant drop in performance within the virtualized environment. Even in a single node test OpenFOAM performed poorly compared to the native's performance. Further analysis with basic runtime profiling revealed a drop in performance when "icoFoam", the OpenFOAM solver heavily utilised the CPU's Streaming SIMD Extensions (SSE). Hypervisors were configured to expose the needed subset of CPU features to the guest operating system. However the translation at the virtualization layer for the specific set of CPU features was obviously done inefficiently. We were able to identify a different set of CPU optimisation that was needed for better compatibility with KVM and ESXi

for OpenFOAM and did get better but still not satisfactory results. It was because we wanted to retain the compatibility and overall face-to-face comparison with the defaults that we kept the original results. Tests with finer tuning of compilation options for each application on separate environments (ESXi and KVM) would most probably mitigate SSE issues and this is left for our future work.

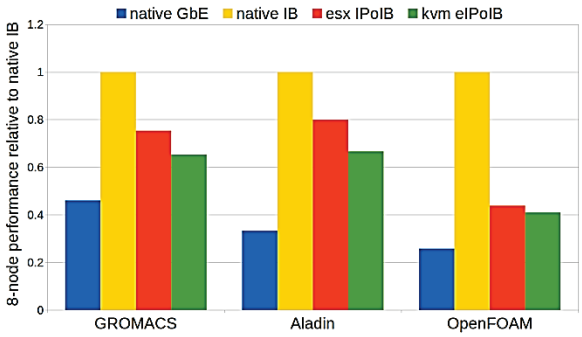

Figure 6 8-node performance relative to native InfiniBand environment for GROMACS, Aladin and OpenFOAM applications

There are two aspects that need to be evaluated when dealing with network interconnect: performance and scalability. The important fact, learned from our performance analysis is that virtualized network significantly reduces performance, but at least at some degree retains scalability. We analysed 8-node performance ratios relative to native performance where we evaluated performance ratio PR as

$$PR(platform) = \frac{8nodeperf(native)}{8nodeperf(platform)}$$

and 8-node scalability ratio relative to 1-node performance where we evaluated scalability ration SR as

$$SR(platform) = \frac{1nodeperf(platform)}{8nodeperf(platform) \times 8}$$

Table 1 Performance and scalability ratio for general-use applications

	GROMACS		Aladin		OpenFOAM	
	PR %	SR %	PR %	SR %	PR %	SR %
native IB	100,0	84,65	100,0	94,83	100,0	93,76
native GbE	46,05	38,94	33,33	34,38	34,78	33,73
esx IPoIB	75,36	65,41	80,00	82,50	55,28	57,19
kvm eIPoIB	65,37	59,94	66,67	70,83	59,11	53,62

Tab. 1 shows both our performance and scalability ratios for evaluated applications. In the 8-node 96-core configuration performance decrease was significant. OpenFOAM showed the biggest decrease in performance and scalability due to SSE issues, as mentioned above.The smallest impact on performance and scalability is observed in 8-node 96-core configuration for Aladin whereas decrease in 8-node GROMACS benchmark is still significant.

6 Live migration impact and proactive fault-tolerance use-case

The most important feature that enables consolidation, overall utilisation and ease of management within a virtualized environment is the ability to move virtual machines across different native hosts. This feature provides the flexibility of moving running virtual machines between native hosts providing a dynamic environment. Live migration also enables usage of a proactive fault tolerance system which adds additional value for running popular general-use applications in virtualized environments, of course, if we can trade performance downsides with management and fault-tolerance benefits. In this section we discus multi-platform fault-tolerance architecture and analyse performance overhead with live migration for our virtualized platforms.

The vast majority of widely used general-use MPI applications are not resilient to hardware errors. If a cluster node suffers from hardware errors, a non-fault tolerant MPI application faces a complete termination and has to be restarted. There are some known workarounds, for example, checkpoint/restart technique [31] e.g. as implemented for OpenMPI [32]. Techniques such as checkpointing are expensive to implement due to significant I/O performance requirements, even though they can be very efficient with checkpointing done over InfiniBand [33]. However, various implementations are likely to be bound to specific MPI versions or MPI feature sets. Significant effort has been put into in-application fault-tolerance support, e.g. for Monte Carlo methods [34] and to implement fault-tolerance support directly into MPI specifications by MPI 3.0 Fault Tolerance Working Group. This is undoubtedly the right direction for future MPI application development. However, it will take significant effort and time for widely used general-use MPI applications to be ported or rewritten to support new MPI versions, therefore the need for an in-place solution is obvious. Checkpointing has also been evaluated within virtual environments with the proposal of BlobCR, a checkpointing implementation at the virtual disk snapshot level [35]. Live migration can also be used as a transparent multi-platform fault tolerance. It allows live migration of a running virtual machine from one to another native host. If we are able to predict a hardware failure by constantly monitoring a native host, we can take advantage of the live migration technique and execute a live migration of a running virtual machine to a stand-by native host, thus preventing MPI application crash/termination in the case of predictable native host hardware failure. Obviously this technique does not provide a 100 % fault-tolerance as we are unable to predict all hardware failures in advance, nevertheless with a comprehensive set of monitoring parameters we are able to achieve an acceptable percentage of successful predictions as has been originally presented by Arun Babu Nagarajan et al. [36]. In this section we present the concept of a plug-in architecture, allowing us to use the proactive fault tolerance on any virtualization platforms by simply adding platform specific plug-ins, where Linux based virtualization plug-in can be based on, e.g. Arun's work, VMware plug-in can be based on the existing VMware tools and plug-in for other virtualization platforms to be added later on. As a proof of concept we analysed live migration impact on VMware ESXi and KVM. A multi-platform fault tolerance system would consist of 1) a health monitoring module and 2) a migration orchestrator module. The implementation of

health monitoring plug-ins is left for future work as this paper focuses on performance aspects. Therefore we only describe the migration orchestrator.

The migration orchestrator holds a list of available spare nodes. Spare nodes are native servers that are online with a near zero load (CPU, memory and I/O) over the past time interval, indicating a free native host with no running or only paused stand-by virtual machines. Such a host is elected as a migration target candidate and the migration orchestrator will live-migrate virtual machines from source host to the available free target host. The libvirt [37] library is used as the base of the migration orchestrator − it supports a very wide range of virtualization platforms (hypervisors) therefore it can be used to execute live migration for KVM, ESX, Xen and many more. This enables a multi-platform fault tolerance system or migration orchestrator as a standalone tool to be used on the majority of popular virtualization platforms. Load detection is simplified to the extent of the proof-of-concept, as needed for the scope of this paper; Ganglia monitoring system [38] output is used for Linux based virtualization and esxtop/restop utilities in case of VMware virtualization. Simplified load detection is used by only gathering a few attributes: past 30-minute CPU, memory and I/O load parameters where each of the attributesmust have less than 1 % load for a node to elect as a live migration target candidate. Only a basic Ganglia setup is needed for the basic load parameters to be available and esxtop/restop utilities provide basic load parameters out of the box.

6.1 Live migration performance overhead

Live migrations of the running applications should be executed without interruption and with as small as possible performance overhead on the running application. Live migration is defined as the process of moving a running guest operating system from one native host to another, where the process is completely transparent to the guest operating system without any functional interruption. All CPU, memory and I/O states have to be transferred and synchronised to the target native host. Under heavy CPU load with significant memory utilisation the synchronisation requires enough network bandwidth to support live migration traffic. Usually a dedicated network interface is used for live migration. Fig. 7 shows our benchmarking results on Aladin application. We used our migration orchestrator to choose a spare free node and initiated a live migration process of a random running loaded virtual machine. Aladin is very handy for these kinds of tests with its possibility of monitoring performance in real time. It outputs time recordings, the need for a one calculation step (see also Section 5). When live migration took place, the time for calculation steps, executed during live migration, increased because of the performance overhead. The host had to process the live migration thus using CPU cycles that could not be used for the application itself. Processing latency was also introduced due to the states' synchronisation requirements from source to target host.

Tests were run using live migration traffic over dedicated GbEnetwork as well as over IPoIB interface, thus sharing bandwidth with MPI network traffic. In our previous tests we observed that IPoIB throughput on the native host had not been saturated by MPI traffic, therefore we could afford to use it in shared mode for live migration tests. As shown in Fig. 7, live migration was very efficient. Even on a very short run at only 100 steps in total of approximately 15 minutes of full run, the total run time was increased by less than 6 % when running over GbE and less than 2 % when over shared IPoIB. We observed very similar results with live migration when running other applications as well. Since live migration overhead is absolute for applying runtime, projecting these results to even longer-running applications, the overhead of migration can become almost insignificant. Our results show that live migration is very efficient with negligible impact on medium or long running applications.

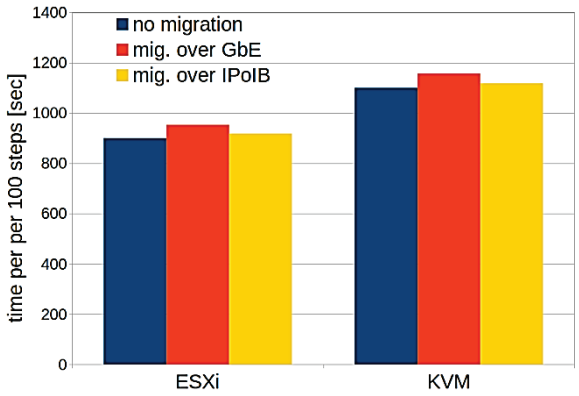

Figure 7 Live migration impact on Aladin with 8-node configuration

7 Conclusion

This paper presented MPI application performance analysis within a virtualized environment using para-virtualized IPoIB InfiniBand network. Performance analysis was done on proven and production-ready commercial and open source virtualization platforms, VMware ESXi and KVM, respectively. IPoIB was used for virtualizing InfiniBand as one of the few technologies which provides out-of-the-box production support for virtualizing InfiniBand without modifications in virtualization software, middleware or hardware and at the same time fully supports transparent live migration of virtual machines thus allowing consolidation, overall utilisation and ease of management of a converged fully virtualized IT infrastructure. Furthermore we discussed and proposed a plug-in based multi-platform proactive fault tolerance system which provides fault tolerance for MPI applications by utilising virtual machine migration on a variety of virtualization platforms as a use case for a live migration performance overhead benchmark. Our performance analysis results show that IPoIB InfiniBand virtualization brings significant performance overhead, even though scalability is preserved to a certain degree. Moreover, we showed that the performance overhead in live migration is negligible for medium and longer application runs. Due to the high performance overhead we can conclude that IPoIB virtualization is not suitable for large systems. Nevertheless, small HPC users, especially in smaller research groups and SMB

Enterprises may find IPoIB InfiniBand virtualization performance acceptable by leveraging converged dynamic virtualized IT environments with the possibility of automatic live migrations, load balancing and optional fault tolerance of MPI applications. Our performance analysis on synthetic benchmarks shows what kind of bandwidth and latency performance we can expect when using IPoIB virtualization – this data can be used as a reference when one is deciding on expected performance for a particular application. Furthermore we have shown that performance and scalability can vary a lot dependingon a particular application. Also, hypervisor limitations in exposing advanced CPU features (such as SSE and AVX) must be evaluated when using a particular application within a virtualized environment.

For a truly efficient execution of MPI applications in virtualized environments, different technologies such as SR-IOV, full native para-virtualization and other OS-bypass techniques have to be used for efficient InfiniBand network virtualization. The performance evaluation of the latter is left for our future work when these kinds of technologies with full live migration support will already be in production versions of various virtualization platforms.

8 References

[1] Walker, D.; Dongarra, J. MPI: a standard message passing interface. // Supercomputer. 12, (1996), pp. 56-68.

[2] IP over InfiBand Working Group. URL: http://www.ietf.org/wg/concluded/ipoib.html.

[3] Fortissimo EU Project. URL: http://www.fortissimo-project.eu.

[4] VMware ESXi hypervisor. URL: http://www.vmware.com/products/vsphere/features/esxi-hypervisor.html.

[5] Kivity, A.; Kamay, Y.; Laor, D. kvm: the Linux virtual machine monitor. // Proceedings of the Linux Symposium / Ottawa, 2007, pp. 225-230.

[6] International Data Corporation (IDC). URL: http://www.idc.com/.

[7] Tomić, D.; Ogrizović, D.; Car, Z. Cloud solutions for high performance computing: oxymoron or realm? // Tehnicki vjesnik-Technical Gazette. 20, 1(2013), pp. 177-182.

[8] Jackson, K. R.; Ramakrishnan, L.; Muriki, K.; Canon, S.; Cholia, S.; Shalf, J.; Wasserman, H. J.; Wright, N. J. Performance Analysis of High Performance Computing Applications on the Amazon Web Services Cloud. // 2010 IEEE Second International Conference on Cloud Computing Technology and Science / Indianapolis, 2010, pp. 159-168. DOI: 10.1109/CloudCom.2010.69

[9] Zhai, Y.; Liu, M.; Zhai, J.; Ma, X.; Chen, W. Cloud versus in-house cluster: evaluating amazon cluster compute instances for running MPI applications. // 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC) / Seattle, 2011, pp. 1-10. DOI: 10.1145/2063348.2063363

[10] Expósito, R. R.; Taboada, G. L.; Ramos, S.; Touriño, J.; Doallo, R. Performance analysis of HPC applications in the cloud. // Future Generation Computer Systems. 29, 1(2013), pp. 218-229. DOI: 10.1016/j.future.2012.06.009

[11] He, Q.; Zhou, S.; Kobler, B.; Duffy, D.; McGlynn, T. Case study for running HPC applications in public clouds. // Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing - HPDC '10 / New York, 2010, pp. 395-401. DOI: 10.1145/1851476.1851535

[12] Regola, N.; Ducom, J. Recommendations for virtualization technologies in high performance computing. // 2010 IEEE Second International Conference on Cloud Computing Technology and Science / Indianapolis, 2010, pp. 409-416. DOI: 10.1109/CloudCom.2010.71

[13] Younge, A. J.; Henschel, R.; Brown, J. T.; von Laszewski, G.; Qiu, J.; Fox, G. C. Analysis of Virtualization Technologies for High Performance Computing Environments. // 2011 IEEE 4th International Conference on Cloud Computing / Washington, 2011, pp. 9-16.

[14] Dong, Y.; Yang, X.; Li, J.; Liao, G.; Tian, K.; Guan, H. High performance network virtualization with SR-IOV. // Journal of Parallel and Distributed Computing. 72, 11(2012), pp. 1471-1480. DOI: 10.1016/j.jpdc.2012.01.020

[15] Jose, J.; Li, M.; Lu, X.; Kandalla, K. C.; Arnold, M. D.; Panda, D. K. SR-IOV Support for Virtualization on InfiniBand Clusters: Early Experience. // 2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing / Delft, 2013, pp. 385-392.

[16] Ma, Y.-M.; Lee, C.-R.; Chung, Y.-C. InfiniBand virtualization on KVM. // 4th IEEE International Conference on Cloud Computing Technology and Science / Taipei, 2012, pp. 777-781. DOI: 10.1109/CloudCom.2012.6427589

[17] Ramakrishnan, L.; Canon, R. S.; Muriki, K.; Sakrejda, I.; Wright, N. J. Evaluating interconnect and virtualization performance for high performance computing. // Proceedings of the second international workshop on Performance modeling, benchmarking and simulation of high performance computing systems - PMBS '11 / Seattle, 2011, pp. 76-86. DOI: 10.1145/2088457.2088459

[18] Liu, J.; Huang, W.; Abali, B.; Panda, D. High Performance VMM-Bypass I/O in Virtual Machines. // Proceedings of the annual conference on USENIX '06 Annual Technical Conference / Boston, 2006, pp. 255-269.

[19] Mauch, V.; Kunze, M.; Hillenbrand, M. High performance cloud computing. // Future Generation Computer Systems. 29, 6(Aug. 2013), pp. 1408-1416. DOI: 10.1016/j.future.2012.03.011

[20] Clark, C.; Fraser, K.; Hand, S.; Hansen, J. G.; Jul, E.; Limpach, C.; Pratt, I.; Warfield, A. Live Migration of Virtual Machines. // 2nd Symposium on Networked Systems Design & Implementation / Boston, 2005, pp. 273-286.

[21] Zhai, E.; Cummings, G. D.; Dong, Y. Live Migration with Pass-through Device for Linux VM. // Linux Symposium / Ottawa, 2008, Volume Two, pp. 261-268.

[22] Mellanox InfiniBand Drivers. URL: http://www.mellanox.com/page/software_overview_ib.

[23] Motika, G.; Weiss, S. Virtio network paravirtualization driver: Implementation and performance of a de-facto standard. // Computer Standards & Interfaces. 34, 1(2012), pp. 36-47. DOI: 10.1016/j.csi.2011.05.002

[24] Dongarra, J. J.; Luszczek, P.; Petitet, A. The LINPACK Benchmark: Past, Present and Future. // Concurrency and Computation: Practice and Experience. 15, 9(2003), pp. 803-820. DOI: 10.1002/cpe.728

[25] Aladin high resolution numerical weather prediction project. URL: http://www.cnrm.meteo.fr/aladin.

[26] Berendsen, H. J. C.; van der Spoel, D.; van Drunen, R. GROMACS: A message-passing parallel molecular dynamics implementation. // Computer Physics Communications. 91, 1-3(1995), pp. 43-56. DOI: 10.1016/0010-4655(95)00042-e

[27] Jasak, H.; Jemcov, A.; Tukovic, Z. OpenFOAM: A C++ library for complex physics simulations. // International Workshop on Coupled Methods in Numerical Dynamics / Dubrovnik, 2007, pp. 1-20.

[28] Kanies, L. Puppet - Next-Generation Configuration Management. // Login. 31, 1(2006), pp. 19-25.

[29] OpenFabrics Alliance. URL: https://www.openfabrics.org.

[30] Ali, Q.; Kiriansky, V.; Simons, J.; Zaroo, P.Performance evaluation of HPC benchmarks on VMware's ESXi server. // Euro-Par'11 Proceedings of the 2011 international conference on Parallel Processing / Bordeaux, 2012, pp. 213-222.
[31] Duell, J., The Design and Implementation of Berkeley Lab's Linux Checkpoint / Restart. // Lawrence Berkeley National Laboratory, 2000.
[32] Hursey, J.; Squyres, J. M.; Mattox, T. I.; Lumsdaine, A. The Design and Implementation of Checkpoint/Restart Process Fault Tolerance for Open MPI. // 2007 IEEE International Parallel and Distributed Processing Symposium / Long Beach, 2007, pp. 1-8. DOI: 10.1109/IPDPS.2007.370605
[33] Panda, D. K. Application-Transparent Checkpoint/Restart for MPI Programs over InfiniBand. // 2006 International Conference on Parallel Processing (ICPP'06) / Ohio, 2006, pp. 471-478.
[34] Pauli, S.; Kohler, M.; Arbenz, P. A fault tolerant implementation of Multi-Level Monte Carlo methods. // Parallel Computing: Accelerating Computational Science and Engineering (CSE), 2014, pp. 471-480.
[35] Nicolae, B.; Cappello, F. BlobCR: Virtual disk based checkpoint-restart for HPC applications on IaaS clouds. // Journal of Parallel and Distributed Computing. 73, 5(May 2013), pp. 698-711. DOI: 10.1016/j.jpdc.2013.01.013
[36] Nagarajan, A.; Mueller, F. Proactive fault tolerance for HPC with Xen virtualization. // Proceedings of the 21st annual international conference on supercomuting / Seattle, 2007, pp. 23-32. DOI: 10.1145/1274971.1274978
[37] libvirt, the virtualization API. URL: http://libvirt.org.
[38] Massie, M. L.; Chun, B. N.; Culler, D. E. The ganglia distributed monitoring system: design, implementation, and experience. // Parallel Computing. 30, 7(2004), pp. 817-840. DOI: 10.1016/j.parco.2004.04.001

Authors' addresses

Marko Kobal
Arctur d.o.o.
Industrijska cesta 5
5000 Nova Gorica, Slovenia
E-mail: marko.kobal@arctur.si

Zlatan Car
Faculty of Engineering
University of Rijeka
Vukovarska 58
51000 Rijeka, Croatia
E-mail: car@riteh.hr

Center for advanced computing and modelling
University of Rijeka
Radmile Matejčić 2
51000 Rijeka
Email: zlatan.car@uniri.hr

Milan Ojsteršek
Faculty of electrical engineering and computer science
University of Maribor
Smetanova ulica 17
2000 Maribor, Slovenia
E-mail: milan.ojstersek@um.si