High-performance aspects in virtualized infrastructures

Vitalian A. Danciu, Nils gentschen Felde, Dieter Kranzlmüller, Tobias Lindinger
Munich Network Management Team
Ludwig-Maximilians-Universität München
Oettingenstr. 67, D-80538 München
{danciu,felde,kranzlm,lindinge}@nm.ifi.lmu.de

***Abstract*—In this paper we analyse the suitability of computing clouds, i.e. large-scale virtualized infrastructures for high-performance applications that are normally executed on specialised clusters or supercomputers. We classify such applications according to their requirements on different system components and present measurements of virtualization software overhead for these components. Based on the results, we discuss cloud-tolerant problems and address surprising performance effects observed in different hypervisors.**

I. Motivation

Today, while Peta-FLOP systems are being installed around the globe, the planning for exascale systems is already underway, with such systems expected around 2019. Such high-end computing (HEC) systems are projected to contain in excess of $1000,000,000$ CPU cores and consequently present enormous challenges because of their dramatically increased parallelism and complexity. The cost for acquiring and operating such a system through its life-time is accordingly exuberant. Clearly, this development will encounter hard financial (if not technical) limits for publicly funded systems.

At the same time, computing power is being offered as a commodity provisioned by virtualized infrastructures operated in a cloud model. Provided this emergent business model pans out, it does indeed make sense to explore the option of solving computing problems hitherto reserved for HEC systems on the growing platform of offerings.

On the other hand, host virtualization as a technology exhibits properties that may be instrumental to the management and operation of very large scale systems. Virtual machines (VMs) are already being used successfully as a base for the execution of programs in massive HEC clusters. Such use of virtualization techniques in HEC poses interesting research questions. In this paper, however, we will focus on the formerly mentioned issue: the use of virtualized environments for solving HEC problems.

II. Background and related work

Current roadmaps [7] project one billion cores in exascale systems, four orders of magnitude more parallelism than the approximately $100,000$ cores in current petascale systems. This vastly increased parallelism presents what appears to be an even greater challenge for both systems software and application software design.

The use of heterogeneous systems, i.e. a combination of traditional general-purpose cores with GPUs (graphics processing units) and accelerators, will further complicate applications, algorithms and associated programming models. The increased component count, power and cooling requirements of these systems will present significant resilience and fault tolerance challenges. Finally, the move to radically different system architectures raises important questions about legacy applications and system software that, viewed globally, comprise financial investments in the range of the budgets of small countries.

Given the physical size and raw component count in an exascale machine, fault tolerance may very well be a show stopper. The solution can no longer be the traditional checkpoint-restart, but instead built-in system and application resilience.

The decoupling of the physical machine from the program run by means of a virtualization layer relocates fault, configuration, accounting, performance and security (FCAPS) management functions to dynamic relocatable entities—the VMs in cloud-like environments. In addition to being able to compensate physical failures by means of migration techniques, a virtualization layer enables dynamic and fine grained partitioning of a cluster at run-time. More efficient distributions of virtual machines, load balancing and even ad hoc tapping into spare capacity of other data centres are conceivable. At the same time, a higher security baseline is provided implicitly as a consequence of sandboxing applications within their own VM.

Given these facts and projections, it might seem peculiar that virtualization has not been embraced at once in the domain of high-end, high-performance computing. The single most important issue that keeps virtualization out of the TOP500 list is the *performance overhead* introduced by the virtualization facility, e.g. the hypervisor.

It is this overhead that we examine in this paper, from both qualitative and quantitative perspectives.

A. Related work

The issue of performance inherent to host virtualization has indeed been pointed out early on; in particular, the role

of I/O in performance considerations has been noted [3]. Examinations of performance have been carried out for some hypervisors (Xen is certainly a preferred candidate, probably due to being released as Free Software, with its sources readily available)[4], [13], and opportunities for improvement have been pointed out with respect to CPU and I/O performance[12], [2].

In contrast, this paper does not focus on a specific virtualization technology instance, but instead evaluates VM-based computing per se for use in high-performance and high-throughput applications.

B. Structure of this paper

Section III outlines the requirements on virtualization use in High-End Computing (HEC) based on the common problem classes in parallel computing. Based on these requirements, we select measurement setups and present the results of those measurements in Section IV. We interpret these results in Section V to determine the suitability of virtualization facilities for different computational problems and summarise our conclusions in Section VI.

III. Performance impact factors

The goal of this work is to determine the applicability of large scale virtualized infrastructures, such as clouds, to high-performance computing problems. Requirements on the performance profile of such infrastructures are consequently derived from HEC problems in the following.

A. Problem classes

The problems addressed in our analysis require massive parallel execution in general, yet different classes of computational problems impose different requirements on the systems they are computed on, in terms of *component* performance. We differentiate roughly between a *High-Performance Computing* (HPC) and *High-Throughput Computing* (HTC) problem classes. Table I characterises these problem classes according to architectural criteria.

a) HPC problems: are tightly coupled and need to transmit intermediate results between nodes frequently. Hence, the performance of their computation is highly dependent on locality and on the characteristics of a common memory and/or the network interconnecting computing nodes.

b) HTC problems: are loosely coupled computations that include the class of so called embarrassingly parallel problems. Computations are often organised in tasks within a workflow and/or pipelined through a hierarchy of nodes. The performance of their computation is determined primarily by the raw CPU power available.

In general, we can thus identify several impact factors that have relevance to both VM-based computing and to the HEC problems delineated:

- Effective CPU performance and support for special instruction sets, especially extensions supporting host virtualization [15].
- Memory organisation (addressing scheme, size, latency, "jitter")
- Latency and throughput of the interconnect.

c) Job management.: Virtualization technology affords several novel mechanisms of job (read: VM) control previously unavailable in purely physical computing environments. However, these mechanisms will influence job execution at runtime in a positive or negative manner: Job *submission* (as VM images) allows (but also forces) a-priori specification of VM resource requirements known at deployment or relocation time. The *migration* of VMs across network induces a change in locality that implies higher or lower interconnect latencies. Changes in resource assignment may result in higher or lower performance, depending on the program run.

B. Summary of requirements

In order to be suitable for all HEC problems, an execution environment must meet the following requirements simultaneously:

1) High aggregate CPU performance
2) Low interconnect latency
3) High interconnect transmission capacity
4) Contiguous memory addressing
5) Uniform latency of memory access
6) Conservation of locality once program replicas have been deployed
7) Predictable assignment of resources
8) Fixed assignment of resources after program replicas have been deployed

C. Benchmarking the cloud

Due to the sheer amount of processing time, the program code of HEC applications is optimised for specific architectures. Hence, for good results, knowledge about hardware characteristics is necessary at design time. With specialised clusters or supercomputers such knowledge is available beforehand and allows adaptation of program code for execution on a specific installation, often even before that installation is deployed.

In contrast, the opaque nature of clouds obviates the acquisition of such knowledge deliberately. The type of virtualization, i.e. *para-virtualization*, *full virtualization* or *OS virtualization*, constitutes a determinant to performance overhead. Due to the lack of prior knowledge of the hypervisor class in use, we determine virtualization overhead by virtual component (CPU, memory, network or disk), as these are virtualized using a single virtualization technique instead mix of techniques employed to create a VM.

The degree of fulfilment of some of the requirements is pre-determined by the business model of compute clouds: they consist of commodity server machines interconnected by means of commodity network technology. They cannot realise a contiguous, uniform-latency addressing scheme due to the isolation property of VMs. Locality may be sacrificed for the sake of load balancing, and resource assignment might change for the same reason.

Criterion	High-Performance	High-Throughput
coupling	tight	loose
CPU impact	co-determinant	critical
interconnect impact	critical	less
RAM addressing	sensitive; prefers contiguous, uniform latency addressing	less sensitive
program structures	inter-communicating program replicas	workflows; pipelining of computing tasks
Examples	fluid dynamics problems, crash codes	high-energy physics (e.g. CERN LHC experiments), general parameter variation studies

Table I: Characterisation of problem classes

Thus, we consider the remaining two dimensions of performance in comparison to the performance of a native OS:

1) The overhead of the virtualization facility with respect to CPU and interconnect.
2) A representative set of virtualization facility products (hypervisors) that employ different virtualization strategies.

IV. Measurements

The measurements presented in this paper constitute an excerpt of a larger sequence of measurements (see [10]) that would overtax the available space. Thus, we will focus on the performance *overhead of CPU* and the *throughput of network* devices as the determining components for HPC and HTC problems. This section will present our approach to create a benchmark suite and present the results obtained when running the benchmarks on several of the common hypervisor implementations including Xen 3.2, VMware ESXi 3.5, MS Hyper-V and Virtuozzo 4 / OpenVZ.

A. Designing a performance measurement suite

When running benchmarks on virtual machines one has to take into account mainly two issues that do not arise when the same benchmark is executed on a physical machine. Firstly, results of virtual machines cannot be interpreted stand-alone, but rather must be compared to a reasonable result obtained in an identical physical environment. Secondly, measurements of time within a virtual machine are error prone due to several layers of independent scheduling. While attempts to synchronise clocks of virtual and physical machines are underway [8], [14], the large number of eligible counters requires a careful selection of those best suited for the desired benchmark. This implies that a benchmark executed on a virtual machine should either be proven to yield correct results or its source code must be obtainable to verify (and if necessary, to fix) timing related issues. In addition, we require the benchmarks selected to be part of our test suite to be executable on both Windows and Linux and on both 32-bit and 64-bit platforms.

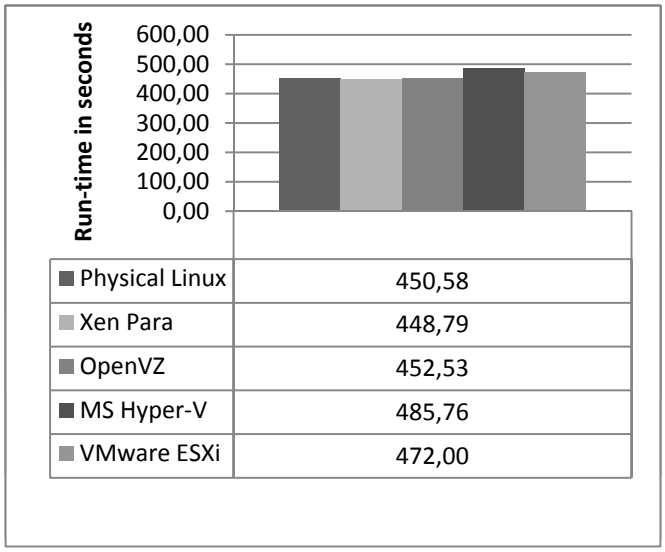
1) CPU: We have chosen the *Linpack* benchmark [9] to be the benchmark for CPU-related measurements. It fulfils the requirements stated above and is well known in the HEC community as the benchmark ranking the TOP500 list of supercomputers. Unfortunately, the time measurement used by the Linpack benchmark is based on the wall clock time of the application which is not even defined properly for virtual machines. Consequently, we extended the benchmark by adding an external clock updated over UDP instead of asking the internal clock counter. This introduces a delay compared to the original meassurement, but since the same benchmark is used on a physical machine one can compare the results. We ran the modified Linpack on different matrix-sizes beginning from 1000x1000 up to 8000x8000 measuring the absolute run-times of each pass. We aggregated these values to compute the average of all time measurements.

2) Network: For our networking tests we selected *Iometer* [6] which can fulfil our requirements and can be used without further changes. It consists of two components: the dynamo client, which is responsible to produce the network load, and the Iometer control instance, which realises time measurements. Placing the latter component on a physical machine effectively eliminates time-related measurement problems. To eliminate such effects due to network message sizes, we performed the same experiment using different segment sizes between 1 kB and 32 MB to measure the throughput.

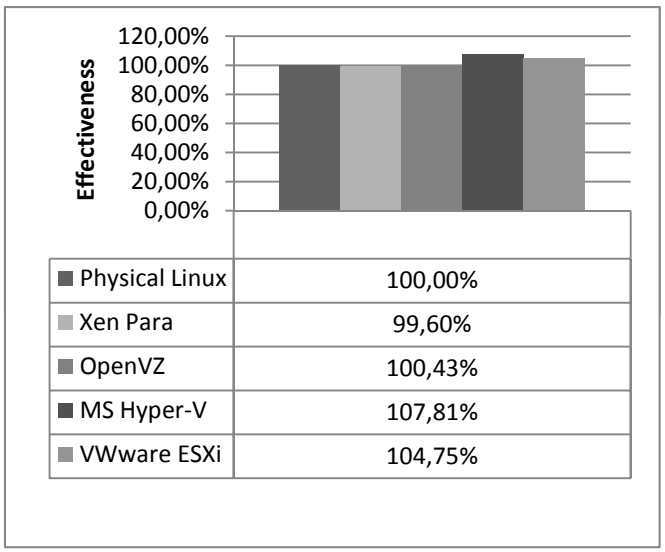
B. Applying the test suite

Single runs of Linpack and Iometer with one benchmark instance in a single active virtual machine yield efficiencies close to 100% across hypervisors, client platforms and operating systems. Thus, we only present the more interesting concurrent measurements. Sample results showing the Linpack runtime for a physical machine running Ubuntu Linux (kernel 2.6.18) as well as corresponding Linux VMs that are virtualized using the four different hypervisor products are illustrated in Figure 1. We encountered an anomaly in the doubled run-time of Linpack on Windows, as compared to the same tests on Linux. We attribute this to the treatment of different segment sizes in Windows's UDP stack, as this behaviour could not be replicated with the unmodified version of Linpack. The behaviour of the Xen virtual machine is even better then the result of the physical maschine. We believe this to be caused by larger page sizes within the hypervisor leading to a smaller number of page faults.

1) Concurrent CPU tests: On our two-core machine, we ran the adapted Linpack benchmark in one, two and three concurrent virtual machines. As the underlying physical system had only two CPU-cores to bind virtual machines to, we expected

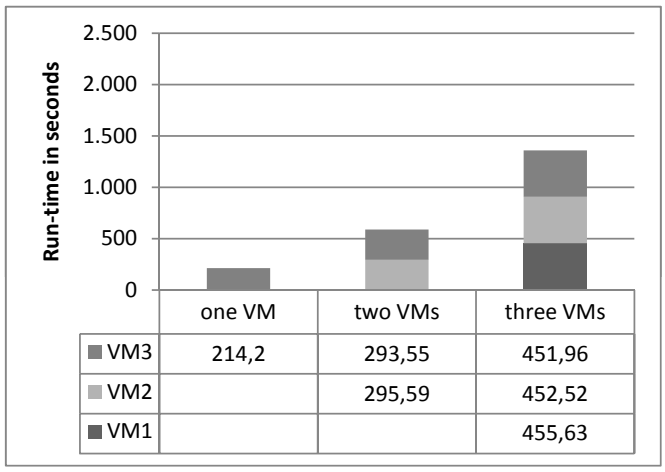


(a) Absolute run-time

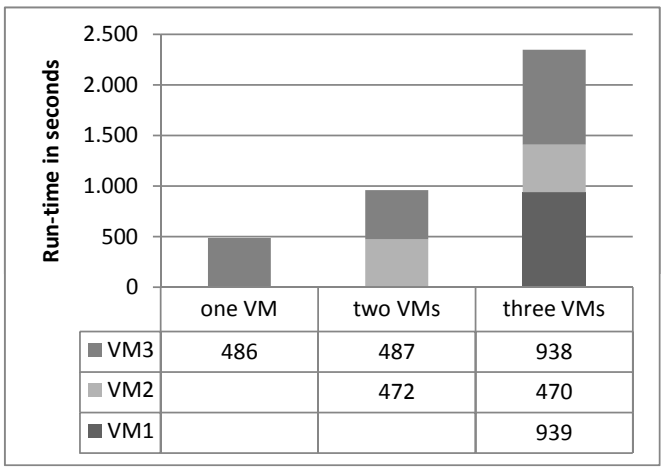


(b) Relative run-time

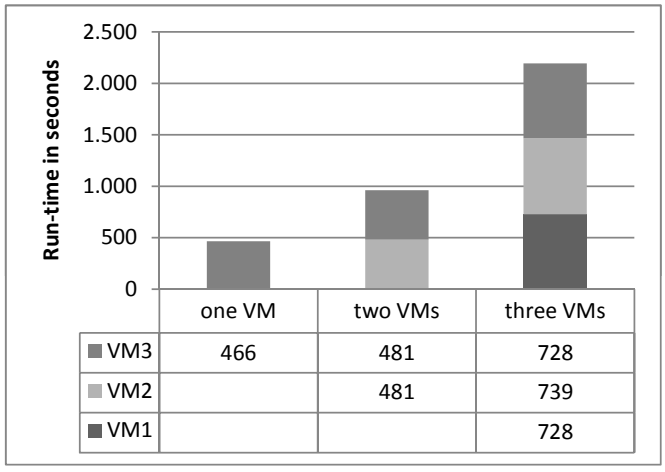
Figure 1: Effectiveness of current hypervisors (Linpack)



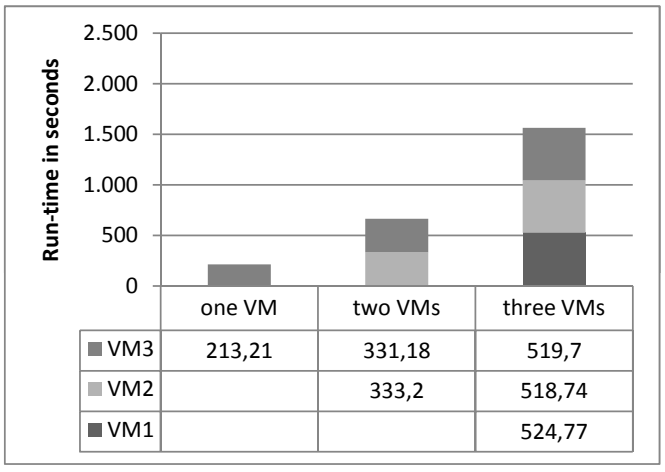
(a) Xen



(b) Virtuozzo



(c) Hyper-V



(d) ESXi

Figure 2: Concurrent CPU runs

the most significant results in the category of running three instances of Linpack simultaneously. The results, shown in Figure 2 exhibit a linear increase of run-time in relation to the amount of machines and therefore the amount of Linpack instances in all scenarios except when using Virtuozzo as a hypervisor. Virtuozzo seems to implement a strict binding of virtual machines to physical cores since one machine finishes calculation twice as fast than the other ones. While this implementation is a good idea to make most efficient use of caches, it is prone to distribute CPU time in an unfair manner.

2) Concurrent network I/O: This test measures the network throughput (both reading from and writing to the network) of one to three virtual machines executed on the same physical host communicating with a second physical host. The virtual machines constitute sending or receiving entities. During the tests packet sizes were varied. The results are illustrated in Figures 3, 4, 5 and 6.

We observed an interesting effect with respect to read vs. write operations according to the VM clock. While clients (e.g. a benchmark program) can only send data when the VM is scheduled to be active, reception of data from the network is assured even while the VM is scheduled to be inactive. Thus, the VMs time frame for the reception of data may be smaller than real time.

3) Locality of communication peers: Inter-communicating VMs can be located on the same physical machine, or on different physical machines. We did not measure communication between two virtual machines located on different physical hosts, as the receiving time-frame anomaly due to scheduling effects would appear in both directions. Instead, we interpret the results in comparison with the former experiment, where a VM sends or receives data from the physical network. In this symmetric experimental setup we need not distinguish between sending and receiving scenarios as they are obviously equivalent. We can see a significant increase of the throughput observed in the virtual scenario across all hypervisors with the exception Virtuozzo. We determined this effect to be due to Virtuozzo's network driver implementation. The quantitative results are shown in Figure 7.

V. Discussion

Based on the measurements of the efficience of single virtual components, we can conclude that the overhead introduced by the virtualization layer is relatively small and will be acceptable against the background of the administrative benefits gained by the use of host virtualization in large-scale scenarios. However, our measurements have revealed several non-linear aspects of virtual component performance, that are due to multiple, decoupled scheduling procedures. As some of these effects entail a high load of the CPU during network transmissions, they cannot be discounted when assessing the suitability of host virtualization for use in High-End Computing. In the following, we discuss our findings with respect to the HEC problem classes identified in Section III.

A. Observations

Our experiments suggest that modern virtualization approaches can be highly efficient in abstracting single virtual components (such as a virtual CPU, RAM or NIC). Within limitations, the VMs themselves can be considered to be very efficient against the baseline of the physical machine supporting it. Virtual component types are not equally efficient regarding costs, however. Specifically, virtualizing I/O-components such as network interfaces or host-bus adapters entails emulation of their properties in software; thus, creating virtual I/O-components may be highly compute intensive. The introduction of hardware support for I/O-virtualization may lessen this impact in the domain of single physical machines [5], but an actual improvement would naturally require corresponding end-to-end transmission capacity on the interconnecting network. We consider it highly improbable that IT-installations intended for cloud computing will be equipped with interconnects featuring latencies and throughput characteristics as demanded by tightly coupled HPC applications, as such features may not be marketable to non-HPC customers.

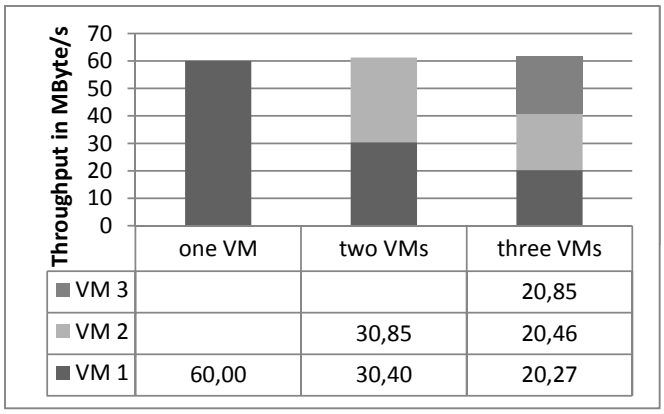
This limitation applies to a much lesser extent to the loosely coupled HTC problems; while computing nodes will communicate occasionally, program execution speed is determined to a much higher extent by the performance of virtual CPUs, and in some main memory-intensive cases by the speed of the CPU-memory connectors. As our experiments show, virtual CPUs are efficiently abstracted across virtualization software brands, and the impact of the memory access latencies is highly dependent on the locality within the application programs.

Virtualized infrastructures, by their very nature, do not provide contiguous memory address space. This does not preclude effective execution of parallel programs, but does imply the use of message passing libraries, such as the Message Passing Interface (MPI) [1], which are optimized for most effective communications between peers using the shortest path available. However, depending on the construction of the cluster, the network topology may change (unexpectedly, for MPI) due to live migration of VMs and "confuse" this highly optimised message passing.

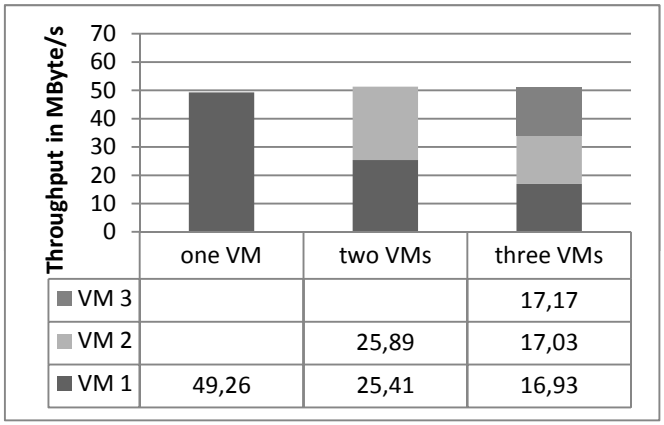
Where usable, virtualization will enable flexible management of resources and a high degree of automatisation; this is fortunate, as a flexible management will become an essential aspect in emerging HEC environments due to the high amounts of resources, e.g. CPUs in a magnitude greater than 10^6, committed to these calculations will not be manageable manually as before. Another aspect to watch, while operating a high performance cloud cluster, is the overcommitment of resources. This will create large performance penalties due to the overhead created by scheduling of virtual machines and especially in consequence due to the loss of locality and cache poisoning issues.

B. Verdict

We conclude that it is indeed possible to virtualize loosely coupled, HTC problems in cloud environments, but that such

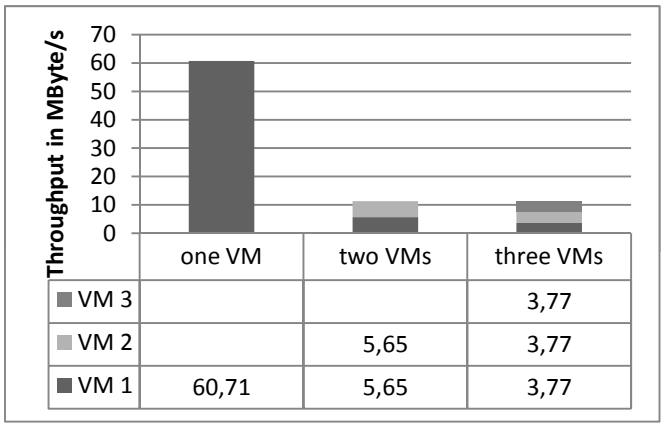


(a) Receive data from network

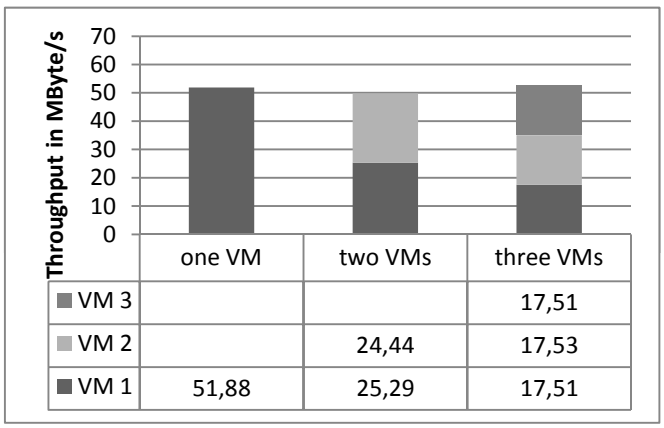


(b) Send data to network

Figure 3: Concurrent I/O on network - Xen

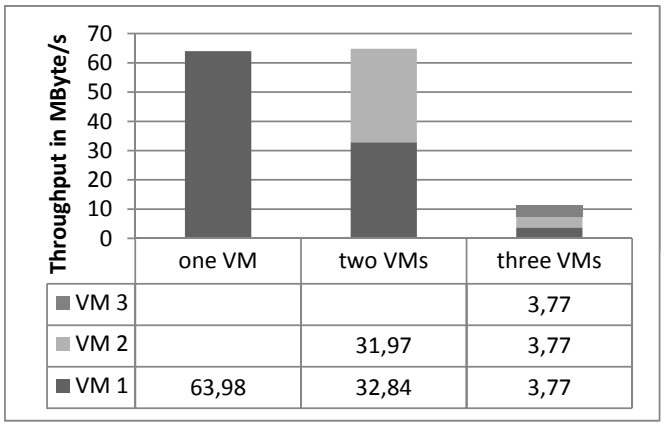


(a) Receive data from network

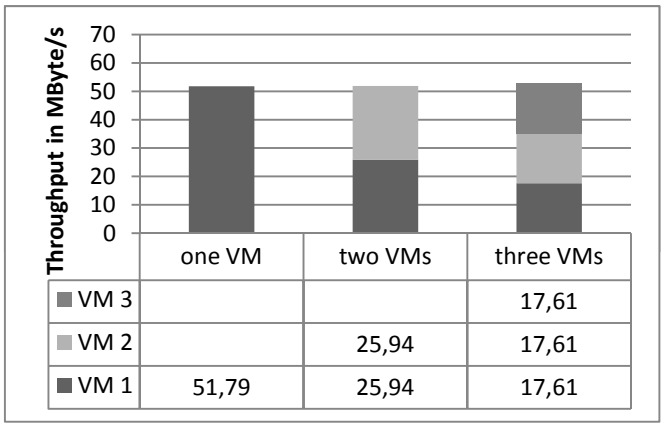


(b) Send data to network

Figure 4: Concurrent I/O on network - Virtuozzo

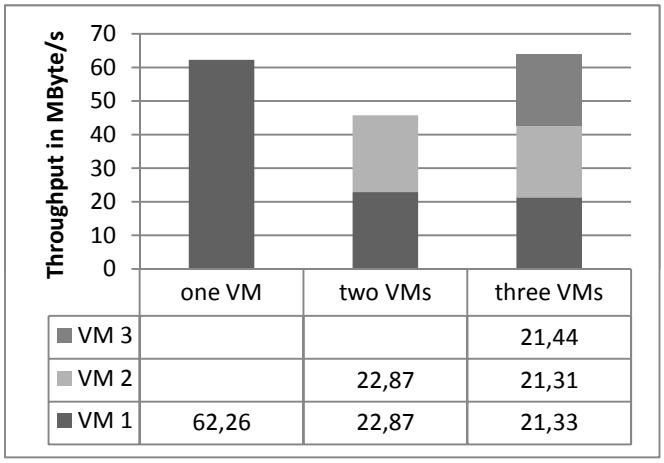


(a) Receive data from network

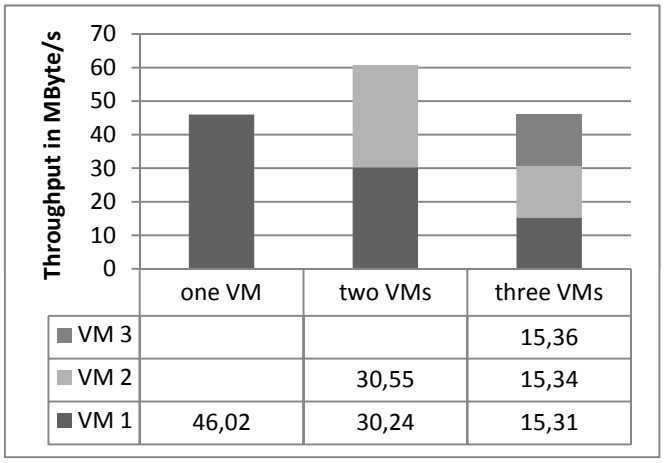


(b) Send data to network

Figure 5: Concurrent I/O on network - Hyper-V

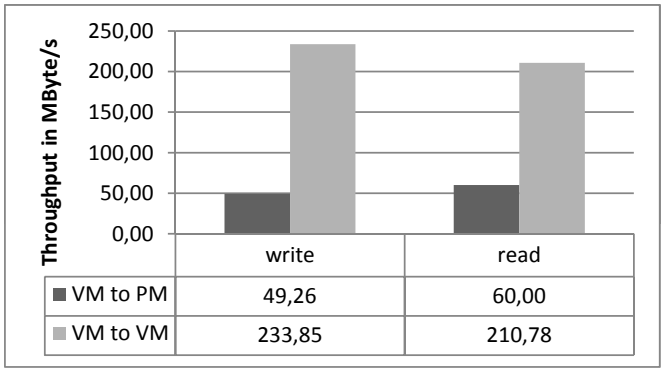


(a) Receive data from network

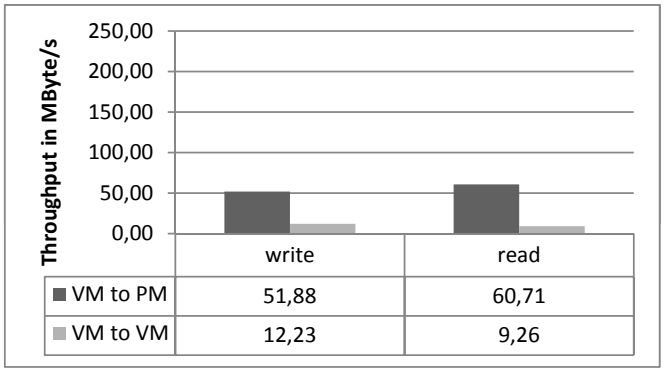


(b) Send data to network

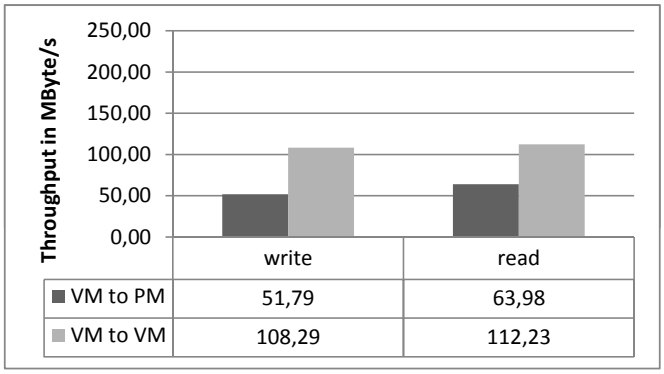
Figure 6: Concurrent I/O on network - ESXi



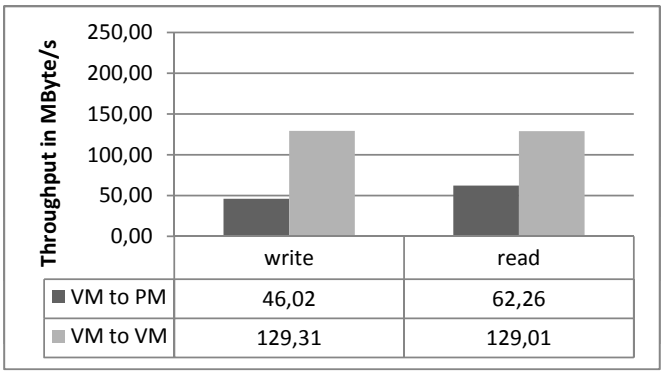
(a) Xen



(b) Virtuozzo



(c) Hyper-V



(d) ESXi

Figure 7: Physical vs. virtual communication peers

environments are unsuitable for the execution of tightly coupled parallel programs due to the overhead incurred by the virtualization of I/O-components and the requirements on the network interconnecting the computing nodes for up-to-date virtualization techniques.

VI. Conclusions

The success of virtualization technology in general, and its deployment in cloud installation in particular, are driven by novel, flexible provisioning and load distribution options. The decoupling of machine images from the actual physical hardware allows flexible assignment of resources in the same manner as it is provided by specialised HEC middleware facilities. At the same time, a virtualized environment can prove more tolerant against localised faults, and it can be reconfigured without service interruption, to allow maintenance. However, the architecture of virtualized infrastructures and the manner in which they are loaded lead to performance bottlenecks being exhibited by common PC-like machine architectures. On one hand, "performance leaks" occur due to the virtualization software layer, on the the other hand, the co-location of machines may overtax subsystems such as I/O.

In this paper, we have presented performance measurements of VMs to determine their suitability to support HEC applications. To avoid having to extrapolate from specific computational problems, we have adapted standard HEC benchmarks (e.g. Linpack) to be usable in a virtualized environment and measured the performance index of separate and aggregate virtual system components.

Our results indicate that, while some computation problems are well suited for execution on VM environments, e.g. in cloud offerings, other problem classes suffer from performance issues induced by the aforementioned bottlenecks. High-Throughput Computing problems, as encountered in the particle physics appear suitable to be computed on large sets of VMs; in contrast, High-Performance Computing (HPC) problems as in fluid mechanics simulations appear problematic due to their need to pass results quickly between computing nodes. In addition, the flexibility of VMs in terms of topological arrangement and resource assignment (through migration and resource pool concepts) forces programs to be constructed without assumptions with regard to locality (and thus latencies) or node resources. Thus, HEC applications are computable on virtualized, or cloud, infrastructures only to limited degrees.

A. Outlook on future research

Computability of the problematic HPC applications can be enhanced either by taking into account the characteristics of virtualized environments at development time, or by demanding from the virtualization platform that it take into account locality requirements of the applications. Either of these strategies may lead to the enlargement of the problem space tolerant to VM-based computing.

Another interesting and related field is the introduction of host virtualization in specialised HEC environments, in order to make use of the resource management capabilities associated with VMs while retaining the hardware characteristics necessary for tightly coupled problems. This entails the operation of virtualization facilities "out of their depth" in massively parallel environments

As the number of virtualization software brands is small compared to the plethora of programs in the HEC domain, the second avenue seems more promising for practise; in addition to primitive functions of hypervisors already supporting such functions, locality control appears to be a technique suitable for problems outside the HEC domain, as well.

Acknowledgment

The authors wish to thank the members of the Munich Network Management Team (MNM Team) for helpful discussions and valuable comments on previous versions of this paper. The MNM Team directed by Prof. Dr. Dieter Kranzlmüller and Prof. Dr. Heinz-Gerd Hegering is a group of researchers at Ludwig-Maximilians-Universität München, Technische Universität München, the University of the Federal Armed Forces and the Leibniz Supercomputing Centre of the Bavarian Academy of Science. http://www.mnm-team.org

References

[1] MPI: A Message-Passing Interface Standard – Version 2.2. http://www.mpi-forum.org/docs/mpi-2.2/mpi22-report.pdf, September 2009.

[2] Matthew Arnold, Adam Welc, and V. T. Rajan. Improving virtual machine performance using a cross-run profile repository. *SIGPLAN Not.*, 40(10):297–311, 2005.

[3] Peter M. Chen and Brian D. Noble. When virtual is better than real. *Hot Topics in Operating Systems, Workshop on*, 0:0133, 2001.

[4] Ludmila Cherkasova and Rob Gardner. Measuring cpu overhead for i/o processing in the xen virtual machine monitor. In *ATEC '05: Proceedings of the annual conference on USENIX Annual Technical Conference*, pages 24–24, Berkeley, CA, USA, 2005. USENIX Association.

[5] Vitalian A. Danciu and Martin G. Metzker. On I/O virtualization management. In *Proceedings of the 3rd International DMTF Workshop on Systems and Virtualization Management*, Wuhan, China, September 2009. Distributed Management Task Force.

[6] Daniel Scheibli and Ming Zhang. *Iometer Project*, September 2009.

[7] Jack Dongarra et al. International exascale software project roadmap (draft 0.93), November 2009.

[8] Scott Drummonds. *Time-based Measurements in Virtual Machines*, May 2008.

[9] Jack Dongarra, Jim Bunch, Cleve Moler, and Gilbert Stewart. *Linpack Benchmark*, June 2010.

[10] Tobias Lindinger. *Optimierung des Wirkungsgrades virtueller Infrastrukturen*. Dissertation, Ludwig–Maximilians–Universität München, February 2010.

[11] Bruce McCready, Kenneth C. Barr, and Kiran Tati. *ESX Server Best Practices for Performance*, September 2008.

[12] Aravind Menon, Alan L. Cox, and Willy Zwaenepoel. Optimizing network virtualization in xen. In *ATEC '06: Proceedings of the annual conference on USENIX '06 Annual Technical Conference*, pages 2–2, Berkeley, CA, USA, 2006. USENIX Association.

[13] Aravind Menon, Jose Renato Santos, Yoshio Turner, G. (John) Janakiraman, and Willy Zwaenepoel. Diagnosing performance overheads in the xen virtual machine environment. In *VEE '05: Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments*, pages 13–23, New York, NY, USA, 2005. ACM.

[14] Andreas Stiller. *Magazin für Computertechnik: Xen und die virtuelle Zeit*, March 2007. Seite 180.

[15] Ken Strandberg. *Performance Impacts with Optimized Virtual Environments on Intel Virtualization Technology-based Platforms*, 2006. Intel Whitepaper.

[16] VMware, Inc. *Performance Benchmarking Guidelines for VMware Workstation 5.5*, April 2006.

[17] VMware, Inc. *Performance Tuning Best Practices for ESX Server3*, January 2007.