Performance of OpenMP simulations on the cloud

Natalia Seoane[1], Raul Valin[2] , Antonio J. Garcia-Loureiro[1], and Tomás F. Pena[1]

Abstract—**In this paper we present an analysis of the impact of the hyperthreading, the number of virtual machines employed per host, the number of cores and the I/O on the performance of two parallel scientific benchmarks. The virtual machines have been managed through the KVM hypervisor included in the CloudStack platform. The applications selected as benchmarks are the Linpack and a two–dimensional Multisubband Ensemble Monte Carlo semiconductor device simulator that has been parallelised using OpenMP.**

The obtained results show for 4 cores lower simulation times than for 2 cores when just 1 or 2 VMs are used per host. The influence of enabling the hyperthreading is insignificant for a low number of virtual machines per host (1 or 2), but for a higher number of VMs there is a considerable reduction in the simulation time when the hyperthreading is activated, that is more substantial when there is a considerable I/O or in the Linpack benchmark. Moreover, we observe an important degradation in the performance when the number of VMs is increased, that is more serious when the hyperthreading is disabled and 4 cores are being used.

Keywords— **Cloud computing, virtualisation, KVM, benchmarking, CloudStack, semiconductor device simulation, Monte Carlo, OpenMP**

I. Introduction

Cloud is drawing the attention from the industry, research centres and the IT community in general since it places the computing infrastructure in the network reducing the management costs of hardware and software resources. Because of such interest there has been substantial development of open source Cloud management platforms such as CloudStack [1], Open Nebula [2] or Eucalyptus [3]. These platforms provide support for several hypervisors like KVM [4] or Xen [5].

In order to test a platform based on CloudStack, we have selected, as examples of scientific parallel computing applications, a two–dimensional Monte Carlo semiconductor device simulator (2D MSB–EMC) parallelised using OpenMP and the Linpack numerical library [15]. Using these applications we have evaluated the effect of over–provisioning on several key factors in cloud computing: the performance, the influence of the hyperthreading, the number of virtual machines running on the same physical host and the hard disk I/O. To manage the virtualised services we have utilised the CloudStack platform.

This paper is organized as follows. Section II describes the main features of the CloudStack platform.

[1]Departamento de Electrónica e Computación, Universidade Santiago de Compostela, e-mails: natalia.seoane@usc.es, antonio.garcia.loureiro@usc.es, tf.pena@usc.es

[2]Supercomputing Center of Galicia (CESGA), e-mail: rvalin@cesga.es

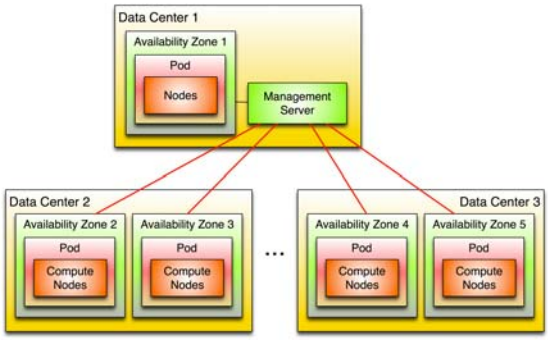

Fig. 1. Scheme representing the CloudStack architecture.

The two scientific applications used as benchmark tests are introduced in Section III. The simulation results are presented in Section IV and conclusions are drawn up in Section V.

II. CloudStack platform

CloudStack [1] is an open source application designed to deploy and manage large networks of virtual machines as a cloud computing platform. The CloudStack software works with a variety of hypervisors including Oracle VM, KVM, vSphere and Citrix XenServer. It provides a massively scalable centralized management server that enables downtime–free management server maintenance and reduces the workload of managing a large–scale cloud deployment. It implements a web interface on top of the CloudStack API to supply a real–time view of the storage, IP pools, CPU, memory and other resources in use. The main benefits of CloudStack are the cutting in IT operation costs and the reduction in complexity and variability by using standard workloads, ensuring the consistency with each application and service deployment [7].

Fig. 1 presents an scheme of the basic components in the CloudStack architecture. The infrastructure is governed by the management server. The virtual machines are deployed in the compute nodes, which are hypervisor–enabled hosts. A group of compute nodes is known as a Pod, and a collection of Pods forms an availability zone. These zones are in sight of the end users, who choose one of them to start a virtual machine.

III. Selected benchmarks

A. 2D Parallel Multisubband Ensemble Monte Carlo simulator

The 2D MSB–EMC simulator is based on mode–space approach of the transport [8] and solves the

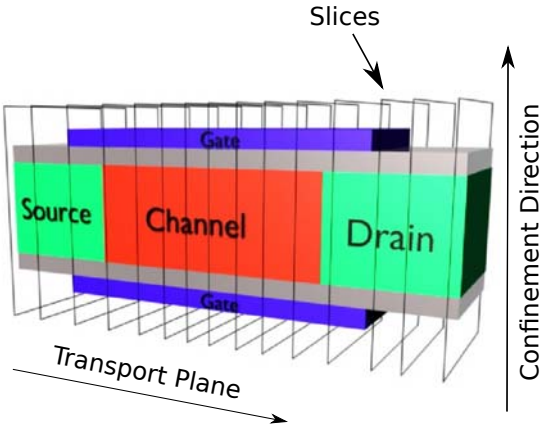

Fig. 2. Representation of the Double Gate MOSFET. The BTE equation is solved in the transport plane and the 1D Schrödinger equation is solved in the confinement direction for each grid point in the transport direction. The drain and source doping concentration is $N_D=10^{20}$ cm^{-3} and the channel doping concentration is $N_A=10^{15}$ cm^{-3}.

Boltzmann Transport Equation (BTE) in the transport direction using the Ensemble Monte Carlo (EMC) method [9]. Quantum effects are introduced via the self–consistent solution of the Schrödinger equation in the confinement direction, perpendicular to the transport plane.

The BTE equation (1) describes the transport phenomena in a semi–classical approach where the fundamental quantity is the carrier distribution function $f(\vec{r},\vec{v},t)$.

$$\frac{\partial f}{\partial t} + \vec{v}\cdot\nabla_r f + \frac{\partial k}{\partial t}\cdot\nabla_k f = \left.\frac{\partial f}{\partial t}\right|_{coll} \qquad (1)$$

The left–hand side describes the time evolution in the phase–space of the carrier distribution function. $\vec{v}$ is the group velocity of carriers and $\partial k/\partial t$ is proportional to the driving field. According to the mode–space approach, the drift field is calculated from the derivative of $E_{i,\nu}(x)$, e.g. the driving force is different for each subband corresponding to a given valley.

The right–hand side represents the change of $f(\vec{r},\vec{v},t)$ due to collisions, in our case, phonon and interface roughness scattering processes. The probabilities of the scattering mechanisms are tabulated in the simulator and they are calculated during the simulation [10].

For thin body devices, quantum confinement effects are quite important and must be considered properly. The MSB–EMC simulator takes these effects into account by coupling the BTE equation with the Schrödinger equation in the confinement direction [11]. From the simulation point of view, our transistor is considered as a stack of slices perpendicular to the transport direction as seen in Fig. 2. The electrostatics of the system is calculated from the self-consistent solution of the 2D Poisson and the 1D Schrödinger equations for each slice and conduction band valley [12].

Fig. 3 depicts a block diagram of the 2D MSB–EMC simulator. As first step, a starting solution

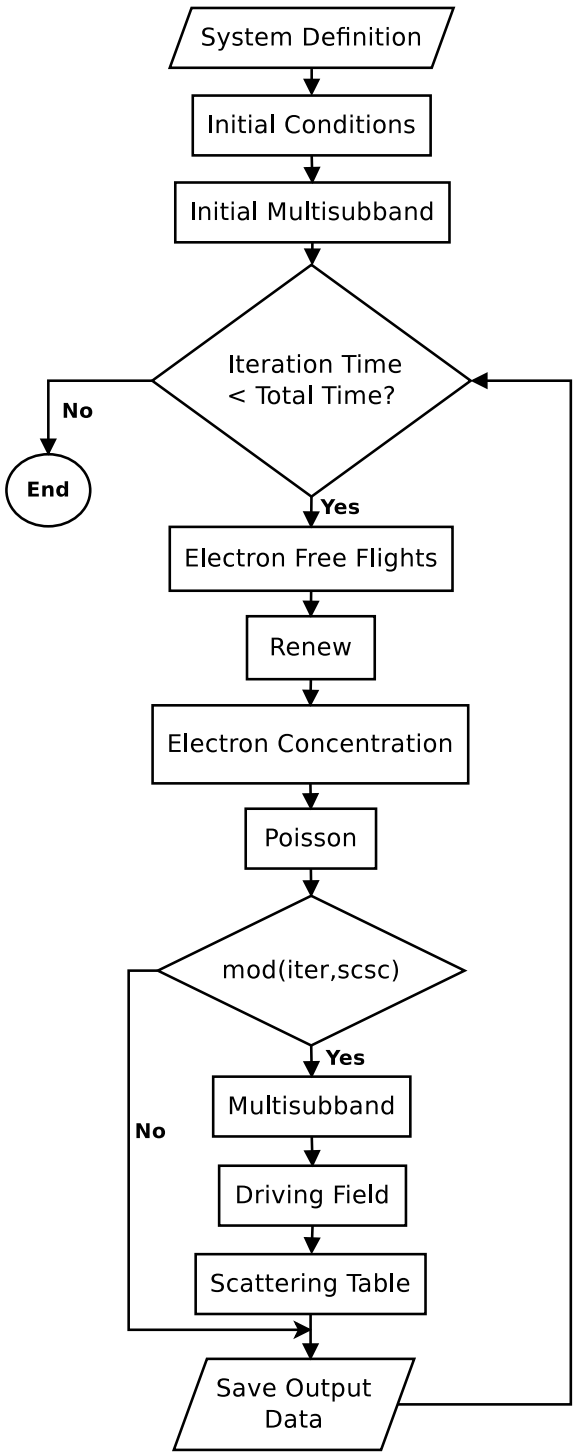

Fig. 3. Flow diagram of the 2D Parallel Multisubband Ensemble Monte Carlo simulator.

of the simulated device (*Initial Conditions* and *Initial Multisubband* blocks) is required in order to obtain the initial values of the electric field and the scattering table needed for the first iteration of the MC simulation. This initial guess is calculated through the self–consistent solution of the Poisson and Schrodinger equations at equilibrium.

The time domain of the MC simulation is discretised in short intervals, namely iteration times, where the motion of the electrons is simulated. After the flight of all superparticles a new electron distribution is obtained from the evaluation of the new po-

sitions and properties of them, and therefore, the electrostatic potential has to be updated solving the 2D Poisson equation in the transport plane. Furthermore, it is also necessary to evaluate the wave–function by solving the 1D Schrödinger equation for each slice in the confinement direction. This will allow us to update the scattering table. The 1D Schrödinger equation and the scattering table have to be solved self–consistently with the updated value of the electrostatic potential.

A drawback of this simulator (compared with semi–classical EMC codes) is its computational burden due to the calculation of the scattering table for each slice for every new solution of the Schrödinger equation to keep the self–consistence of the simulation [13]. In the last step of the simulation flow, the obtained values of the driving field and the probability rates of the scattering tables are employed as initial conditions for the next iteration. This iterative process is repeated up to the end of the simulation time. Bearing in mind the high computational load of these calculations, the 2D MSB–EMC simulator has been parallelised using the OpenMP API [14] to reduce the computational time required to simulate 2D MOSFET transistors.

B. Linpack benchmark

The Linpack benchmark [15] measures the floating point rate of execution of a computer by solving a dense system of linear equations. The user can scale the size of the problem and optimise the software in order to achieve the best performance for a given machine. Linpack uses very intensively floating point operations, so its results are very dependent on the FPU of the system. This benchmark is widely accepted and used in the scientific community as a general method of evaluating computer and supercomputer performance. Currently, it is possible to find performance figures available for most of the systems [16]. Information about the parallel implementation of the Linpack benchmark can be found in [17].

IV. Simulation results

The simulation results presented in this section have been obtained using as a host an Asus P6-P8H61E with 4 cores and 8GB of RAM and an Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz processor. The interconexion network is a GigaBit Ethernet. The operative systems of the host and of the virtual machines are CentOS 64Bit release 6.1 and Debian 6.0.4 64Bit respectively. The network file system used is NFS V3 with the following configuration:

nfs rw, relatime, vers=3, rsize=1048576, wsize=1048576, namlen=255, hard, proto=tcp, timeo=600, retrans=2, sec=sys,

To perform this analysis we have deployed several virtual machines in the same host using the KVM hypervisor (version qemu-kvm-0.12.1.2). The virtualisation API [18] is libvirtd (version 0.9.4). We have

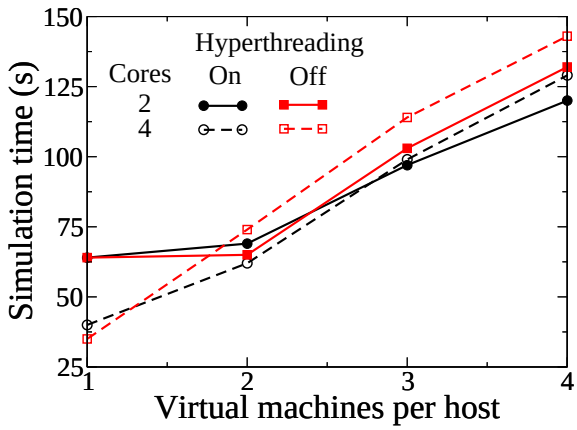

Fig. 4. 2D MSB–EMC simulation time versus the number of virtual machines per host and the number of cores when no I/O is generated. The influence of the hyperthreading is also shown.

used VMs with either 2 or 4 virtual cores, each of them with 2GB of RAM available. The virtual CPU is QEMU Virtual CPU (version cpu64-rhel6).

A. 2D MSB–EMC

Initially, it is worth noting that a realistic simulation of a MOSFET transistor will require performing studies of stationary states of at least 10 ps. Such studies need over 10 hours of computational time in a Xeon processor. In this work we use as a test a shorter simulation of 1 ps and 100000 eps (electrons per superparticle), although the results will still be valid for longer simulation times.

A realistic test using the 2D MSB–EMC simulator will generate an important amount of I/O that needs to be read from or written to hard disk. To avoid the interference of this kind of operations in the results we initially remove all I/O from the simulator. Fig. 4 shows the 2D MSB–EMC simulation time versus the number of virtual cores and VMs deployed per host when there is no I/O generated. The influence of the hyperthreading is also shown in the figure. For 2 cores, with 3 or less VMs deployed, results show similar simulation times whether the hyperthreading is enabled or not. For 4 cores, this is only true when we deploy a single VM in the host. However, for a larger number of VMs, the effect of enabling the hyperthreading is up to a 10% decrease in the simulation time for both 2 and 4 cores.

When comparing the number of cores employed, if the hyperthreading is enabled, simulation times are lower for 4 cores when 1 or 2 VMs are deployed, or for just 1 VM if the hyperthreading is disabled. For a higher number of VMs, the tendency is inverted. As expected, there is an increase in the simulation time when the number of VMs per host is increased. For 2 cores, when the hyperthreading is enabled, this increase is lower than 10% when 2 VMs are being used and it reaches 51% and 87% respectively, for 3 and 4 VMs. For 4 cores, this increase in the simulation time with the number of virtual machines is more pronounced, ranging from a 55% increase when we

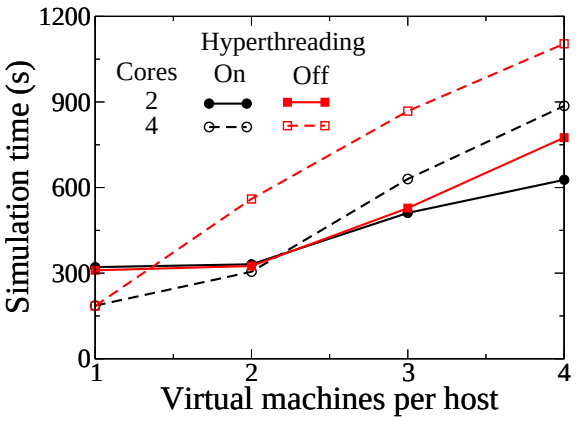

Fig. 5. 2D MSB–EMC simulation time versus the number of virtual machines per host and the number of virtual cores when a realistic I/O is generated. The influence of the hyperthreading is also shown.

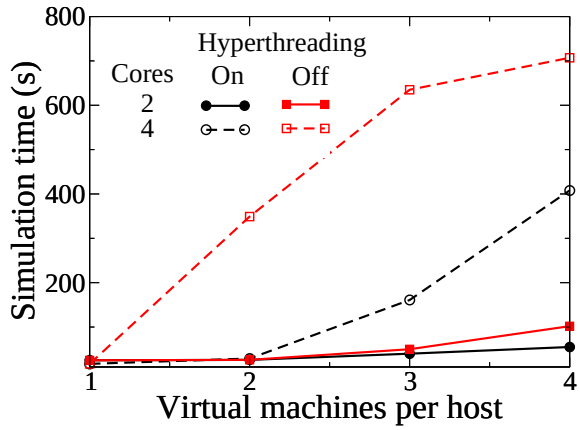

Fig. 6. Linpack simulation time versus the number of virtual machines per host and the number of virtual cores. The influence of the hyperthreading is also shown.

deploy 2 VMs to a 222% for 4 VMs. On the other hand, when the hyperthreading is disabled we observe similar or slightly larger increases in the simulation time for 2 cores. However, the increase in the simulation time when 4 cores are used is more marked, varying from 111% for 2 VMs to over 300% for 4VMs.

We are going to present next an analysis of the impact of the I/O on the performance of the 2D MSB–EMC simulator. In a 1ps long simulation, we perform 1000 iterations (i.e. the time step is 1 fs). Every 10 time steps, we write to a file the values of the different variables that are being calculated: the drain current, the electrostic potential for each node mesh, the subband energy and the population of each subvalley and band. Note that this test is more realistic than the previous one. In order to make this analysis we need to incorporate an extra virtual machine with 4 available cores deployed in an Intel(R) Core(TM) i7-2600 CPU @ 3.4GHz host. This new VM exports an NFS file system that is being shared among the other VMs to store the simulation results.

Fig. 5 shows for this test, the simulation times versus the number of cores and VMs deployed per host. The impact of the hyperthreading is also shown in the figure and it follows the same trend as in Fig. 4. We can see independently of the hyperthreading being enabled or not, minor differences in the execution times for 2 cores, when using 3 or less VMs, or 4 cores, when using 1 VM. In this case, the effect of enabling the hyperthreading is also noticed for a larger number of VMs, with reductions of up to 24% in the simulation time for both 2 and 4 cores.

With respect to the number of cores employed, we obtain lower simulation times for 4 cores when just 1 or 2 VMs are deployed. For a higher number of VMs per host, we observe the oppposite behaviour. Although these results follow a similar tendency than the previously observed in Fig. 4, a much larger deterioration in the performance is found for 4 cores when the number of VMs per host is increased.

We need to bear in mind that an increase in the number of processes that use the disk shared through NFS will lead to interruptions in the application since the I/O operation will require more and more time due to the rise of the number of processes compiting for getting acess to the disk. Consequently, there is a notable increase in the simulation time when the number of VMs is increased. When the hyperthreading is ON and two cores are used, it ranges from an increase of 3% when 2 VMs are deployed to 95% with 4 VMs. If we utilise 4 cores, this increase in the execution time varies between 63% with 2 VMs to 376% with 4 VMs. If the hyperthreading is OFF and we deploy 4 VMs we can see an increase in the simulation time of up to 150% for 2 cores and of almost a 500% for 4 cores. These percentages are notably larger than those observed when there was no I/O written to disk.

B. *Linpack*

In this section, we present a similar analysis using the Intel parallel Linpack benchmark. In this test we use a problem size of 5000, with alignment values of 4KBytes. Linpack provides the average performance in Gflops by all the Linpack runs (5 in our case) for a single test. In order to make a proper comparison between the results obtained when using Linpack and the 2D MSB–EMC simulator, Fig. 6 shows the execution time versus the number of cores and VMs for the Linpack test. We compare these results with the ones presented in Fig. 4 since Linpack just uses the CPU of the system and does not have I/O dependencies. In these two figures the impact of the hyperthreading has also been shown.

First, we examine the effect of activating the hyperthreading on the results. For 2 cores, whether we enable the hyperthreading or not, we obtain the same simulation times when we deploy 1 or 2 VMs. The same happens for 4 cores with 1 VM per host. However, the benefits of enabling the hyperthreading are evident for a larger number of VMs per host, observing for instance, over a 70% drop in the simulation time for both 2 and 4 cores when 4 VMs are used.

For the Linpack test, simulation times for 4 cores are also lower than for two cores when only 1 VM is utilised per host. There is an abrupt decay of the performance when the number of virtual machines is increased as seen in Fig. 6, which is more prominent when the hyperthreading is disabled.

Finally, we evaluate the impact on the performance of the number of virtual machines. When using two cores, simulation results follow a similar trend to the observed in the 2D MSB–EMC simulator. The increase in the simulation time due to an increase in the number of VMs per host is around 4% for 2VMs, and when using 4VMs it reaches 120% and 300% if the hyperthreading is ON or OFF, respectively. For 4 cores, there is a deterioration of the performance when the number of virtual machines is increased that can be seen in the dramatic increase in the simulation time.

V. Conclusion

Cloud technologies are drawing the attention from the industry, research centres and the IT community since they offer the possibility to virtualise services, providing virtual machines to users for their execution. Currently, there is a wide range of open-source solutions for building private, public or even hybrid Clouds.

In this paper we have used the KVM hypervisor included in this platform to manage the virtualised services. The objective is to analyse the influence of the hyperthreading, the number of virtual machines employed per host and the I/O on the performance of two parallel benchmarks: a two–dimensional Multisubband Ensemble Monte Carlo semiconductor device simulator that has been parallelised using OpenMP and the Linpack.

For 4 cores we obtain lower simulation times than for 2 cores but just when 1 or 2 VMs are used per host. The influence of enabling the hyperthreading is imperceptible for 1 or 2 virtual machines per host, but for a higher number of VMs there is a noticeable drop in the simulation time when the hyperthreading is ON, that is more substantial when there is I/O or in the Linpack benchmark. Furthermore, there is a clear deterioration in the performance when the number of VMs is increased, that is more patent when the hyperthreading is OFF and 4 cores are being used.

Acknowledgement

The work developed in this paper has been supported in part by the Ministry of Education and Science of Spain and FEDER funds under contract TEC2010–17320 and by the Xunta de Galicia under contracts 2010/28 and 09TIC001CT.

References

[1] Cloudstack, http://cloudstack.org/
[2] OpenNebula, http://opennebula.org
[3] Eucalyptus, http://open.eucalyptus.com
[4] KVM, http://www.linux-kvm.org
[5] Xen, http://xen.org
[6] F. Gomez-Folgar, J. López Cacheiro, C. Fernández Sánchez, A. Garcia-Loureiro and R. Valin, *An e-Science infraestructure for nanoelectronic simulations based on grid and cloud technologies*, Spanish Conference Electron Devices (CDE), pp. 1-4, 8-11, 2011
[7] CloudStack, http://cloudstack.org/software/features.html
[8] R. Venugopal, Z. Ren, S. Datta, M. S. Lundstrom, and D. Jovanovic, *Simulating quantum transport in nanoscale transistors: Real versus mode–space approaches*, Journal of Applied Physics, Vol. 92, No. 7, p. 3730, 2002
[9] Carlo Jacoboni and Paolo Lugli, *The Monte Carlo Method for Semiconductor Device Simulation*, Springer–Verlag Wien New York, 1989
[10] Mark S. Lundstrom, *Fundamentals of Carrier Transport*, Cambridge University Press, 2000
[11] B. Winstead and U. Ravaioli, *A quantum correction based on Schrödinger equation applied to Monte Carlo device simulation*, IEEE Transactions on Electron Devices, Vol. 50, No. 2, pp. 440-446, 2003
[12] C. Sampedro, F. Gámiz, A. Godoy, R. Valín, A. García-Loureiro, and F.G. Ruiz, *Multi–Subband Monte Carlo study of device orientation effects in ultra–short channel DGSOI*, Solid–State Electronics, Vol. 54, No. 2, pp. 131-136, 2010
[13] C. Sampedro, F. Gámiz, A. Godoy, R. Valín, A. García-Loureiro, N. Rodríguez, I.M. Tienda-Luna, F. Martinez-Carricondo, and B. Biel, *Multi–subband ensemble Monte Carlo simulation of bulk MOSFETs for the 32nm-node and beyond*, Solid–State Electronics, Vol. 65-66, pp. 88-93, 2011
[14] The OpenMP Specifications for Parallel Programming, http://www.openmp.org
[15] Linpack, http://software.intel.com
[16] Linpack report, http://performance.netlib.org/performance/html/PDSreports.html
[17] Intel Math Kernel Library, User's guide, 2008
[18] The virtualization API, http://libvirt.org/