VIRTUALIZATION IMPACT ON COMSOL PROCESSOR DETECTION

Valeriu Manuel Ionescu, Dumitru Cazacu
University of Pitesti, ROMANIA
valeriu.ionescu@upit.ro

Keywords: virtualization, COMSOL, CPU detection, NUMA node

Abstract: *Running operating systems in a virtual environment is necessary to allow isolated systems to share the same hardware resources. The hypervisor handles the resource management and presents to the virtual operating system a user configurable hardware structure that can differ from the physical hardware structure. Some applications, like COMSOL Multiphysics, need to detect correctly the underlying hardware in order to obtain the best performance. This paper investigates how virtualized hardware to real hardware mapping affects the performance of the software running in a virtualized operating system.*

1. INTRODUCTION

Virtualization is a technology that existed for a long time [1] and allows applications and operating system to run in isolation. We will call the operating system that runs on top of the virtual machine guest OS and the one that runs directly on the hardware and runs the virtualization system host OS. The virtualization in the host operating system is handled by the hypervisor application.

The virtualization system modifies the way a guest operating system detects the hardware, therefore applications that are dependent on the proper detection of the hardware structure can behave abnormally, from generating errors to low performance.

COMSOL Multiphysics is an application that presents to the user a simulation environment that gives results very close to the effects observed in reality related to electrical, optics, mechanical, acoustics and other domains. This application is dependent on the proper detection of the existing hardware in order to give its maximum performance, as versions earlier then 4.0 are ignoring hyper threaded CPU cores [2].

The introduction of multi core computers and the evolution of computer hardware according to Moore's law brought enough performance to allow running virtualized operating systems on affordable low cost hardware.

Previous research [2, 3] has shown that in multi core systems it is important to correctly assign the processing resources to get the best performance.

This paper analyses how COMSOL Multiphysics performance will be influenced by running in a guest OS with multiple CPU virtualization options (by varying the number of CPUs and the number of CPUs/core is presented to the virtual machine) and what are the necessary steps to improve the application performance.

2. HARDWARE VIRTUALIZATION

Hardware virtualization is a component of the virtualization that allows the creation of a software version of a hardware system. In this way it is possible to run multiple isolated operating systems on a single computer, improving the hardware utilization and controlling their interaction.

Virtualization offers major advantages:

-machine management is easier and faster because it is centralized and automation is usually available;

-reduces power consumption and capital expenditure as the number of virtual machines is independent of the number physical machines;

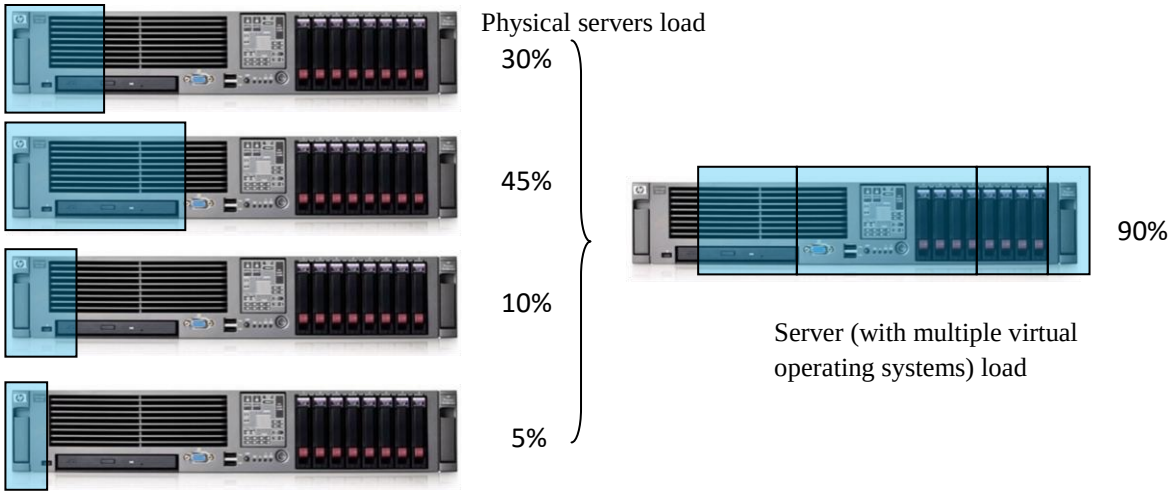

Fig. 1 A single server running virtualized operating systems can perform the tasks of multiple physical machines at a fraction of the hardware cost, necessary space and power consumption

-backup and restore actions are easy to perform and ensure non-stop operation;

-software deployment time is reduced as the environment independent from the actual hardware particularities;

-software testing is sped up because it can be performed in a controlled and repeatable environment.

If used as an application server, ideally a single application or service should run on the operating system because all the available resources are available for that service and problems are easier to detect and solve. As server resources are not used at all times, it is possible to place multiple servers on the same physical machine. Because the relation to the number of physical machines is no longer 1:1, the resources are better used (Fig. 1).

Virtualization, however, has its own problems and limitations:

- while a guest OS can migrate to a new server with a different underlying (host) CPU architecture, problems happen when live system migration is tried between servers running hypervisors on different CPUs types;
- the speed of an applications running in a guest OS is reduced compared to a classical application by a small percentage;
- as many applications have specific requirements related to the hardware characteristics (for CPUs this may involve different feature/flags that specify the instruction set which may be necessary for the application to run) they may run with performance penalties or not run at all. For example the NX flag that allows the marking of memory area as dedicated to code or to data (non-executable memory areas) can be easily presented or hidden to the guest operating system [4]. VirtualBox hypervisor has an "Enable PAE/NX" option to present or not this bit to the guest OS.
- CPU type presented by the hypervisor to the guest operating system is different. By default, the CPUs reported by the virtualization system to the guest CPU are seen as separated physical CPUs not hyper-threaded CPU cores [5]. Also in the case of AMD Fx line of processors, the relation is 2 integer cores to 1 floating point core but the virtualization system reports them as full processors to the guest OS.

3. NON-UNIFORM MEMORY ACCESS (NUMA) RESOURCE ALLOCATION TO THE GUEST OS

Some operating systems have limitations to the number of physical CPUs that can be used (For example Windows Server 2003 and Windows Server 2003 Standard can use a maximum of 4 CPU sockets). Hypervisors offer the option to report a different number of cores per CPU and set the CPU affinity for the guest operating system for compatibility purpose.

As almost any combination of CPUs and cores is possible, but not all combinations present the same performance [3]. This happens because on systems with large number of processors, per core memory allocation and

memory bandwidth are key factors in the performance.

Non-uniform memory access (NUMA) is a computer design that presents non-local memory to the nodes via a fast connection. Each node has its own local memory that can be accessed very fast but can also access data from non-local memory (memory available on other nodes through the fast connection) called foreign memory, but the access to it is slower compared to the local memory because it needs to be moved between processor caches.

The memory access times through the NUMA connection differ, depending on the node's location relative to the accessed memory location.

For example a 4 core CPU, each core with 4 physical processors moves data faster inside a CPU core then between different cores.

As users can specify any CPU resource configuration for a virtual system, if two CPUs from different cores are grouped by the hypervisor in a single logical (virtual) CPU core, moving the data between the two real cores can become extremely costly and lead to performance loss.

4. INTEL HYPERTHREADED CPU

Hyperthreading [6, 7] was introduced by Intel in their processors as the technology that allows implementing two logical processors by sharing the execution resources of a single physical CPU core. This architecture allows just several components to be implemented independently in hardware (processor logic, interrupts) while the hardware components that take a lot of the die space (like caches, the system bus interface).

Introducing hyper-threading resulted in an increase of CPU performance because threads that were blocked while waiting for resources could release the shared execution resources for use in other threads.

The necessity for hyper-threads was due to the fact that many tasks were able to operate in multi-CPU systems and their performance scaled well with an increase in CPU count.

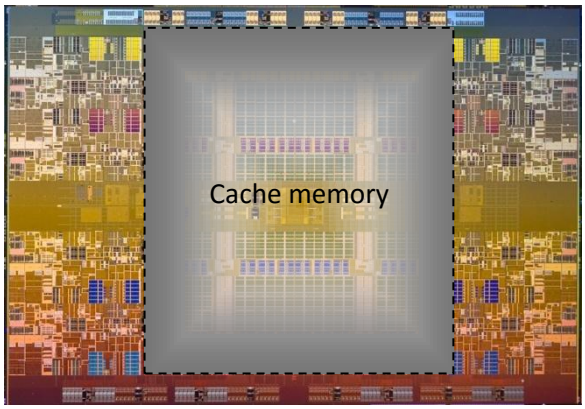

Fig. 2 Intel Xeon processor 7500 series die has a large area for cache memory that is shared between cores

Operating systems were updated to work with this new type of processors. It was observed that the best performance was obtained when a single thread was allocated per core because there was no need to share resources between threads. This means that it is better for the operating system to allocate the odd CPUs first (1,3,5…), then the even CPUs (2,4,6…) because this way you get the maximum performance boost at first and then you get the extra performance from hyper-threading.

Virtualizations systems however present to the guest OS CPUs that give the impression that they are a single CPU per core.

The hypervisor is the one that allocates the CPU cores as needed and by default this is happening automatically. It is possible however to specify a processor affinity for the virtual machines, so that certain real CPUs are used for the guest OS.

5. THE PROBLEM ANALYZED IN COMSOL

The problem we considered for the COMSOL tests is a non homogenous elliptic PDE (Poisson equation) defined on a on a unit square domain $\Omega = (0,1)^2 \subset R^2$, bounded by $\partial\Omega$ and given by:

$$\begin{aligned} -\Delta u &= f && \text{in } \Omega \\ u &= 1/(x+y+1) && \text{on } \partial\Omega \end{aligned} \tag{1}$$

The right hand side term f of (1) is:

$$f(x,y) = -4/(x+y+1)^3 + 2\pi^2 \sin(\pi x)\sin(\pi y) \quad (2)$$

Equation (1) admits as analytical solution the following function:

$$u(x,y) = 1/(x+y+1) + \sin(\pi x)\sin(\pi y) \quad (3)$$

In order to evaluate the convergence of the finite element solution we computed the $L^2(\Omega) - norm$, using the exact solution and the finite element solution u_h defined by the following expression:

$$\|u - u_h\| = \int\int_{\Omega} (u - u_h)^2 dxdy,$$

where h is the mesh spacing parameter.

By refining several times an initial coarse mesh the convergence of the solutions can be evaluated.

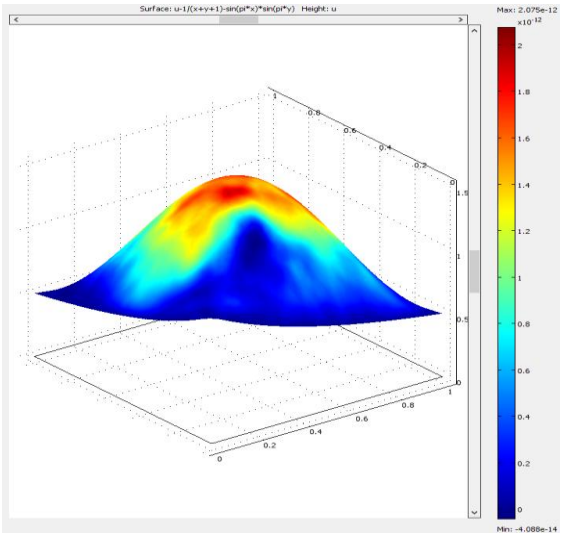

Fig. 3 The graphical result of the simulated problem

From the performance standpoint, the process of refining offers the conditions to use all the CPU cores and needs fast memory access, therefore stressing the system and the NUMA

6. TESTING AND RESULTS

The software used for the tests is COMSOL version 3.4. The operating system is Windows Server 2008 R2 that is installed in a virtual machine. This version has support for NUMA detection.

The operating system will first be tested on a ESXi 5.5 (trial) hypervisor running on a Xeon Processor E7420, 4x4CPU, 16GB RAM, 500GB SAS storage. The trial license was used as the free license for ESXi hypervisor has a limit of 8 vCPUs per virtual machine.

The test related to hyper-threading detection inside the guest OS was made on a i7-4720HQ CPU that has Hyper-threading enabled.

The tests involve:

-inspecting the way the operating system detects the CPUs and reports them to the application;

-inspecting how the NUMA resources are presented to the operating system;

- running the COMSOL application in the guest OS and using the np parameter that allows specifying the number of processors; the effect will be tested on Intel's hyper-threads and AMD's modules.

-setting the processor affinity for the virtualization system.

7. CPU DETECTION IN A GUEST OS

The tests were made using Intel i7-4720HQ CPU (with 4 cores, 2 threads per core) and Windows Management Instrumentation (WMI). WMI can be used (among other purposes) as a monitoring tool.

The following command was run in the Powershell to get the necessary information:

```
WmiObject -class win32_processor -Property
Name, NumberOfCores,
NumberOfLogicalProcessors | Format-List -
Property Name, NumberOfCores,
NumberOfLogicalProcessors
```

For a non virtual operating system running on a system with hyper-threads enabled, the result would show a difference between the number of cores and the number of logical processors:

```
Name                    : Intel(R) Core(TM) i7-4720HQ
CPU @ 2.60GHz
NumberOfCores           : 4
NumberOfLogicalProcessors : 8
```

For a guest OS configured as a 2 CPU system with 4 cores/CPU, all the cores are detected as real cores:

```
Name                    : Intel(R) Core(TM) i7-4720HQ
CPU @ 2.60GHz
NumberOfCores           : 4
NumberOfLogicalProcessors : 4

Name                    : Intel(R) Core(TM) i7-4720HQ
CPU @ 2.60GHz
NumberOfCores           : 4
NumberOfLogicalProcessors : 4
```

This behavior is present not only in Windows, but also for the Linux operating system. The results obtained by inspecting the "/proc/cpuinfo" file for a guest OS configured in a 2 CPU with 4cores/CPU, running the Ubuntu 14.04 OS are:

```
Physical id: 0 siblings: 4 cpu cores: 4
Physical id: 1 siblings: 4 cpu cores: 4
```

The equal number between the number of siblings and the number of CPU cores shows that the cores are perceived as physical cores. In the case of hyper-threaded CPUs the number would have been different (double).

This test showed that if a software runs inside a guest OS, it will be unable to detect the best configuration (number of threads to run on) and will assume that each CPU has no shared components (hyper-threading) that will decrease its performance.

8. NUMA RESOURCE ALLOCATION AND MACHINE PERFORMANCE

One problem is that if the virtualization server is a multi-CPU system, the cores can be reported to the guest OS in different ways (for example for a system with Xeon Processor E7420, 4 sockets with 4 processors each, you can have a 1x16, 2x8, 4x4, 8x2 or 16x1 guest CPUs).

As virtual CPUs are mapped on logical CPUs (which are hardware execution contexts) by the hypervisor, this could lead to performance problems as the hypervisor sometimes cannot map correctly the task schedule on the physical processors [8].

The application used to detect the NUMA configuration detected by the guest OS was Coreinfo v3.31 [9].

COMSOL has a number of command line options that allows the control over the way it handles the number of threads it runs via the "–np" parameter. The tests were run on the Xeon Processor E7420 system.

By running COMSOL for the level 3 refinement level, sub domain selection element settings set to Lagrange – Quintic, the results presented in Table 1 were obtained.

Table 1 Test results for the Quintic configuration, 37760 mesh elements, 473201 Degrees of Freedom

CPU configuration	2x8	5x3	4x4
NUMA nodes detected by the guest OS	2	5	4
Execution time (s) *no –np setting*	42.121	41.9	38.891
Execution time (s) *-np 2 setting*	35.2	35.506	34.819

The software was run initially without command line options, and all the cores are used, and while the tasks are scheduled for 16 CPUs, the performance is not the best. Then the tests were repeated with the –np switch that was found to give the best performance.

The 16x1 CPU configurations is a flat configuration where the resource management is made automatically by the hypervisor.

This would offer a good performance level but was not tested as it is incompatible with the installed operating system (Microsoft Windows Server 2008 R2 Standard Edition supports 4 physical sockets and Microsoft Windows Server 2008 R2 Enterprise Edition supports 8 physical sockets). Only Windows Server 2008 R2 Datacenter allows for more than 8 processor sockets.

For 4x4 CPU configurations, 4 NUMA nodes are presented to the guest operating system and they map perfectly on the hardware.

```
Logical Processor to NUMA Node Map:
****------------  NUMA Node 0
----****--------  NUMA Node 1
--------****----  NUMA Node 2
------------****  NUMA Node 3
```

Approximate Cross-NUMA Node Access Cost (relative to fastest):

```
     00  01  02  03
00: 1.0 1.0 1.2 1.2
01: 1.0 1.0 1.0 1.1
02: 1.0 1.0 1.0 1.0
03: 1.1 1.0 1.0 1.0
```

This configuration also offers the best compatibility and performance level as the virtual to physical mapping is made by the user therefore the Cross-NUMA node access cost is low.

Finally we tested a non-standard configuration 5 nodes x 3CPUs each (15 CPUs), that does not map well to the hardware.

```
Logical Processor to NUMA Node Map:
***---------- NUMA Node 0
---***--------- NUMA Node 1
------***------ NUMA Node 2
---------***--- NUMA Node 3
----------*** NUMA Node 4
```

Approximate Cross-NUMA Node Access Cost (relative to fastest):

```
     00  01  02  03  04
00: 1.4 1.3 1.3 1.2 1.0
01: 1.3 1.3 1.5 1.3 1.4
02: 1.3 1.3 1.3 1.3 1.3
03: 1.4 1.3 1.3 1.3 1.3
04: 1.4 1.3 1.3 1.3 1.3
```

This configuration offers the lowest performance level, with large NUMA resource access costs, as the hypervisor is unable to make a good mapping to the hardware resources.

This investigation shows that there are two possibilities to get the maximum performance from a virtual machine:

- Presenting to the operating system all the cores as individual CPUs. This option will allow the hypervisor to present to the operating system the best configuration;
- Presenting a configuration that matches the physical structure of the server so that performance problems do not occur.

In the performance tests below, the second choice was made, that matched the physical server structure of 4 cores with 4CPUs per core.

Tests were then made to see how the –np setting affects the simulation performance for the 4x4CPU (4 NUMA) setup. This was necessary because, if the application detects all the cores as individual processors, it will allocate the processing tasks for a larger number of threads then the NUMA resources and the performance will decrease.

Table 2 Processing speed relation to –np setting for Q37 (Quintic configuration, 37760 mesh elements, 473201 Degrees of Freedom) and (Quintic configuration, 151040 mesh elements, 1890401 Degrees of Freedom)

np setting	16	8	4	2	1
Q37 (s)	42.972	38.891	36.279	34.819	37.394
Q65 (s)	185.21	169.088	153.083	150.945	156.1

The results show that the best performance is reached at 2 and 4 threads. During this time the CPU were not used at 100% showing that there is a limit in the data that is sent to the processor and not the processor processing power. Using a larger number of threads leads to performance loss as the system has more work to do handling the thread results and when a single thread is used, the results are also bad as the CPU reaches 100% and more performance is needed.

The default CPU allocation and the CPU thread allocation in the guest OS shows that the CPU cores are all equal and are used in order, stressing a single NUMA resource, therefore not having the best performance (Fig. 4-6).

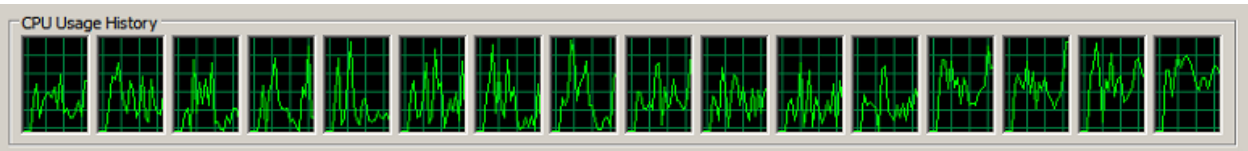

Fig. 4 The usage of all the 16 cores by COMSOL (default setting) leads to lower performance

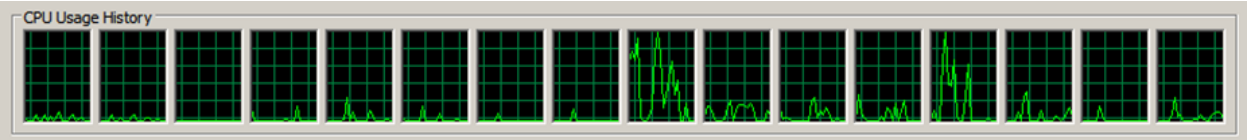

Fig. 5 Using the –np 1 setting for COMSOL. The extra CPU usage is related to other tasks executed by the operating system.

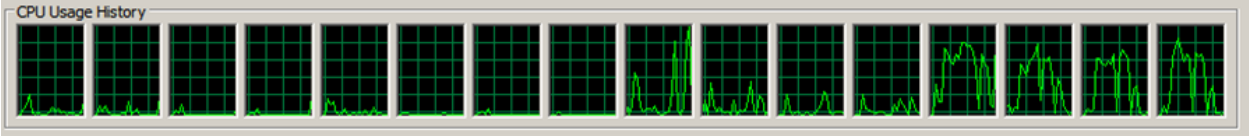

Fig. 6 Using the –np 4 setting for COMSOL. The extra CPU usage is related to other tasks executed by the operating system. The COMSOL execution is limited to the last 4 processors on the right.

As the resource mapping is direct in a 4x4 CPU configuration, if we want to further direct on which of the CPU's available to the operating system is directed the processing, we can use the "Set affinity" option of the Windows operating system. This setting is recommended if a virtualization system is used for the machine running COMSOL, because the software (and for that matter even the operating system) do not know exactly how the real CPU's are allocated by the virtualization system.

It is also possible to set the virtual machine (VM) CPU affinity from the hypervisor (ESXi) in the advanced options, "Scheduling Affinity". This can give a 1:1 real processor to guest OS relation if we want to further speed up the processing for a particular VM.

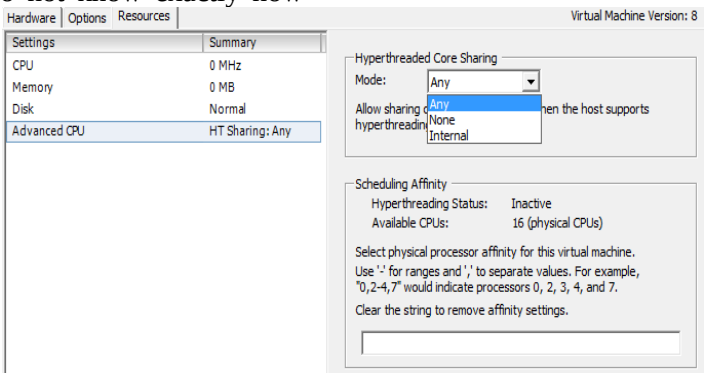

Fig. 7 Setting the processor affinity in the hypervisor's advanced options.

9. CONCLUSIONS

This paper set the goal of testing the performance of an application that is dependent on proper CPU detection and NUMA nodes for giving the best performance.

On a real operating system, the test showed that the software detects the correctly the hardware resources and uses them in order to give the best performance.

In a virtualization system, the resources are presented to the guest operating system according to the user specification. As the software is unable to detect the correct configuration, it uses the processors in order, and the user becomes responsible of the resource usage.

The test showed that a configuration that matches the real hardware setup is advisable for best performance and offers the best compatibility depending on the used operating system.

It was observed that the best performance was obtained not for the number of threads equal to the number of processors but for a lower number, the cause being the number of NUMA nodes available on the system and the system reaching its memory bandwidth limitations as the processors are not used at their full potential.

The results show that if it is necessary to obtain the best performance from a system running in virtual environment test are necessary to obtain the best that include a physical to virtual resource mapping assessment.

BIBLIOGRAPHY

[1] Uhlig, R. et al.; "Intel virtualization technology," IEEE Computer , vol.38, no.5, pp. 48-56, May 2005

[2] Joseph Dieckhans. The Importance of VM Size to NUMA Node Size, Posted on February 8, 2012, Available at: https://blogs.vmware.com/vsphere/2012/02/vspherenuma-loadbalancing.html , Accessed: 11.11.2015

[3] J.D. Freels, I.T.Bodey, R.V. Arimilli. Exploiting New Features of COMSOL Version 4 on Conjugate Heat Transfer Problems, Proceedings of the COMSOL Conference Boston, 2010

[4] VMware, Inc. Workstation 12 Pro Documentation Center, 2015, Available at: http://pubs.vmware.com/workstation-12/topic/com.vmware.ICbase/PDF/workstation-pro-12-user-guide.pdf, Accessed: 11.11.2015

[5] VMware, Inc. Setting the number of cores per CPU in a virtual machine (KB1010184) Available at: http://kb.vmware.com/selfservice/microsites/search.do?language=en_US&cmd=displayKC&externalId=1010184 Accessed: 11.11.2015

[6] Michael E. Thomadakis, The Architecture of the Nehalem Processor and Nehalem-EP SMP Platforms, Supercomputing Facility, Texas A&M University, March, 17, 2011, Available at: http://sc.tamu.edu/systems/eos/nehalem.pdf Accessed: 11.11.2015

[7] D. A. Patterson, J. L. Hennessy. Computer Organization and Design: The Hardware/Software Interface, 4th ed. Morgan-Kaufmann Publishers Inc., 2009, ISBN: 978-0-12-374493-7.

[8] VMware Inc, Performance Best Practices for VMware vSphere® 5.5 VMware ESXi™ 5.5 vCenter™ Server 5.5, May 14, 2014, Available at: http://www.vmware.com/pdf/Perf_Best_Practices_vSphere5.5.pdf, Accessed: 11.11.2015

[9] Mark Russinovich, Coreinfo v3.31, https://technet.microsoft.com/en-us/sysinternals/cc835722.aspx, Published: August 18, 2014