International Journal of Advanced Computer Technology (IJACT)

ISSN:2319-7900

EVALUATING THE USE OF VIRTUAL MACHINES IN HIGH PERFORMANCE CLUSTERS

R L Warrender, J Tindle, D Nelson
Department of Computing, Engineering and Technology, Faculty of Applied Sciences,
University of Sunderland
Sunderland, UK
robert.warrender@sunderland.ac.uk

Abstract

It seems reasonable to assume that the highest performance from a computer node occurs when the basic operating system has been installed directly onto a 'bare-metal' machine. The idea of loading an operating system within another supporting virtual machines, at best sounds convoluted and at worst, a likely drain on the overall performance of the resulting system. However, by comparing the performance of multicore processors in 'virtual' machine environments with those in 'bare-metal' machines the authors explore areas where virtual machines could well result in higher overall levels of efficiency within a cluster environment.

Keywords: High performance computing HPC, parallel computing, cluster, virtual computing, contention, multicore processors.

Introduction

Since 2004, multicore processors have become the norm. This has come from the need to restrict clock speeds in order to prevent excessive power dissipation [1]. While symmetric multiprocessing (SMP) has been a phenomenal success in areas like laptops and workstations, this success has not translated itself evenly to areas like cluster computing where high performance and parallelism are necessary to achieve a significant impact. Their success in laptops and workstations is primarily in allowing multitasking of independent applications with core load balancing. This gives a richer fuller experience to the user rather than the ability to run any individual program any faster. Using SMP to achieve performance and parallelism has resulted in contention within the multicores which has hampered their usefulness (see Figure 1). The aim of this paper is to explore the performance of multicore processors in 'virtual' machines (VMs) compared with 'bare-metal' machines both in terms of how they deal with contention as well as their possible use within a cluster environment.

Throughout this paper the authors use the term 'bare-metal' to signify a computer system where the operating system has been installed directly onto a physical hard disk and would be the only operating system in use within the hardware computer box. Virtual Machines (VMs) are also operating systems but have been installed within the context of a normal operating system. Managed by a 'hypervisor', a computer box can be made to support multiple VMs all apparently running independently (and concurrently) within their own space.

Contention in multicore processors is something that has been the subject of many research papers [2-4]. Contention is generally associated with both bandwidth and memory resources. Bandwidth resource contention comes in the shared bus structure between the processor cores necessitating that the concerned cores arbitrate any resulting congestion. This has the effect of delaying the requests for all but one application until the congestion is clear. Memory resources can also give rise to congestion this time in shared memory resources (such as processor cache) where one core may alter a memory value currently being used by another core. This has the effect of slowing down the operation of one application at the expense of another.

The authors have installed, commissioned and operated a cluster computer for a period of six years. The University of Sunderland Cluster Computer (USCC) is a small but none-the-less powerful general purpose dual boot system that is able to run application programs within the Windows and Linux operating environments. The USCC compute nodes consist of some forty servers plus two headnodes (one used for Windows and the other used for Linux). Each node contains twin dual core Intel Xeon 5150 processors fitted with 8 GB of RAM.

The program being used for all tests carried out in this paper is the 32-bit Windows version of Gaussian_09. This is a state-of-the-art modelling suite based on the work of Professor John Pople who was awarded the Nobel Prize in Chemistry in 1998 for his work in this field [5]. Gaussian_09 is designed to model a broad range of molecular systems under a variety of conditions. All computations performed start from the basic laws of quantum mechanics and can be used to build up a 3D view of the compound. It can also predict energies, molecular structures, vibrational frequencies and numerous molecular properties for systems both in solution and the gas phase.

In order to form a common comparison between results, the Gaussian's command-line software using the same chemical compound (an azobiphenylboronate) was used throughout. This sample job was run directly on the machine in question storing the results locally to ensure no external network issues arose. In this way the authors were able to compare the results from both the existing cluster machines as well as from a new six (hex) core workstation.

Gaussian_09 is not just a single program but a suite of modules, which Gaussian terms 'links', that are used to achieve its goals. The Gaussian website lists the function of each of these links [6]. During normal execution of a Gaussian job, the same program code (i.e. the link) could be called several times from within the Gaussian job before the job is considered to be complete. This would suggest that the Gaussian_09 software is primarily sequential in its execution but makes as much use as it can of any declared hardware resources to utilise parallelism within each 'link' module.

Gaussian publish their own assessment of hardware performance for 1, 2, 4 and 8 core machines for the compound Alpha Pinene [7]. Throughout this study, Gaussian_09 has been set up to use the same numbers of cores in its calculations so that direct comparisons can be made between different configurations. The results relate to the total time taken by the software within a node (or operating system instance), and have been 'normalised' and shown as a speed-up factor. The speed-up factor compares this measured time against that taken for Gaussian_09 software to complete a job using the same physical machine but with a single core processor.

'Bare-metal' performance

The hardware tests were run on two classes of machine.

- Dell 2950 twin dual core server currently installed on the USCC. They consist of twin dual core Xeon 5150 64 bit CPUs (four cores) running at 2.66 GHz with 8 GByte of installed RAM. This processor supports Intel Virtualisation Technology but not Intel Hyper Threading.
- Viglen workstation utilising a single Intel Core i7 3930K Sandy Bridge-E (Hex core) running at 3.2 GHz with 32 GByte of installed RAM. This processor supports both Intel Virtualisation Technology as well as Intel Hyper Threading Technology.

The results of these tests are set out in Table 1.

Intel has two different technologies which merit some comment:

1. Virtualisation Technology (VT)
2. Hyper Threading Technology (HT)

Virtualisation Technology in Intel terms is hardware support for the more commonly known software-based virtualisation solutions such as Microsoft Hyper-V, VMWare, IBM Zen, etc.

Hyper Threading Technology is a feature that under certain conditions will allow two threads to run concurrently within the processor when calling two different CPU functions. This results in the operating system seeing twice as many processors (e.g. a four core processor would appear as eight processors – four real and four virtual). Intel claims that this can account for up to 30% performance enhancement but is dependent on the scheduling of tasks to the cores. Under certain conditions, HT can actually end up degrading the performance [8].

Table 1. Comparative times to execute the test job on different machines using various numbers of specified cores

	Cores	Time (secs)	Speed-Up
Dell 2950 Server (Twin Dual Core)	1	56470	1
	2	29158	1.9
	4	15612	3.6
Viglen Workstation (Hex Core)	1	29812	1
	2	15172	2
	4	6266	3.6
Perfectly Linear	1	n/a	1
	2		2
	4		4
	8		8
Published Gaussian Results	1	n/a	1
	2		1.9
	4		3.4
	8		4.3

Note: All timings shown are an average of several experimental runs where the repeatability was found to be within a range of 1.5% of the average value.

The results as displayed in Figure 1 show a slightly better than predicted speed-up for both of the hardware machines compared to published Gaussian results. However, it has to be taken into account that the compound used in tests carried out by the authors is not the same as that used in the Gaussian published results. It is unfortunate that we were unable to run the test job further than four processors. The authors believe that Gaussian_09 software checks the number of physical cores available and rejects any attempt to run a job with a higher number of specified cores.

The Gaussian published results are interesting in that they show a marked drop in the expected speed-up as the number of cores increases (i.e. a speed-up of only 4.2 when using an eight core processor). The authors believe that this is due to resource contention issues in a multicore system based on virtualisation results discussed later in this paper.

There is a fundamental difference in how the user executes applications on cluster compute nodes as compared to workstations and infrastructure servers such as DHCP, DNS, Database and Web servers. Cluster compute nodes are generally run as close to their full processor utilisation as is physically possible for long periods of time (perhaps several weeks) to maximise the cluster performance and minimise job run times. Nodes are generally allocated a single

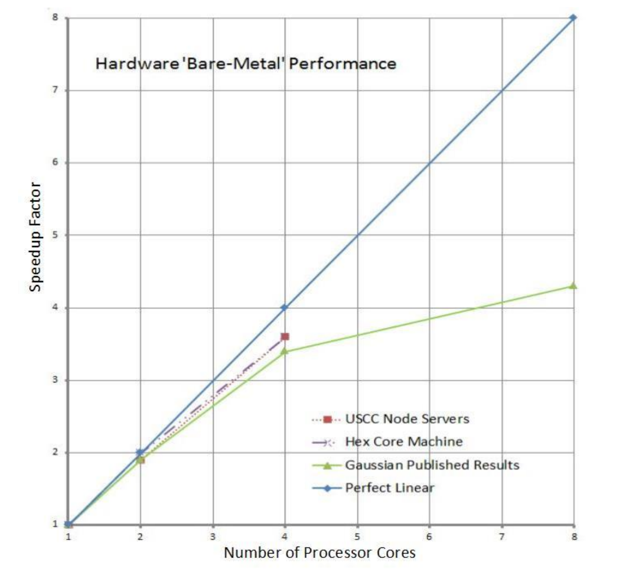

Figure 1 – Hardware 'Bare-Metal' Performance

software job to execute at any time (under control of the headnode) and are not expected to undertake any other task until this is complete. Workstations and infrastructure servers, on the other hand, have to remain vigilant to ensure that any requested task is dealt with without interruption to its current activities. This requires the processor to have sufficient resources 'in hand' to deal with these requests as and when required.

Although contention is an issue in workstations and infrastructural servers, the effect on compute cluster nodes performance is far more marked due to the desire to always operate at the maximum speed-up possible.

'Virtual Machine' performance

All virtual machine performance tests were carried out on a Viglen six (hex) core workstation. The base operating system was Windows Server 2012 R2. Unlike the tests carried out for 'bare-metal' performance, a machine was setup enabling Hyper-V services where a batch of machines could be installed and started separately. The same amount of RAM was specified for each machine (2 GBytes), only the allocated number of processors was varied creating 1, 2, 4 and 8 core virtual machines onto which further installations of Windows Server 2012 R2 were made. This general arrangement is shown in Figure 2.

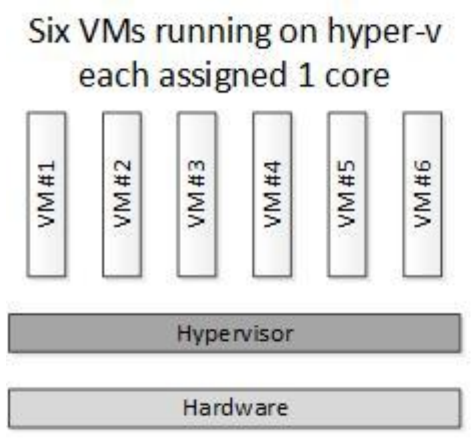

Figure 2 – Typical VM Arrangement

An overall total of twelve VMs were created, however only those involved in the particular test were started for the test. This structural arrangement enabled the authors to run separate Gaussian_09 jobs in parallel on separate virtual machines each within their own program and memory space.

Speed-up calculations take into account the overall time taken to complete a number of parallel jobs compared with that taken to run a job on a single core 'bare-metal' machine

The results of these tests are set out in Table 2.

The results displayed in Figure 3 show a completely different speed-up curve from that shown for the hardware only case. Having a six (hex) core processor allowed the creation of multiple VM configurations. The following were set up and tested as four separate configurations:

1. Six VMs each consisting of one virtual processor
2. Three VMs each consisting of two virtual processors
3. Two VMs each consisting of four virtual processors
4. One VM consisting of eight virtual processors

Table 2. Comparative times to execute the test job on different virtual machines using various numbers of specified virtual cores

No of VMs	Cores/VM	Time (secs)	Speed-Up
6	1	34616	5.2
3	2	17445	5.1
2	4*	11171	5.3
1	8*	5852	5.1

* In both these cases, the total number of virtual cores exceeded the hardware cores within the processor.

The hardware 'bare-metal' configuration allowed only one program instance to be executed or run at a time. This processor configuration creates a (theoretical) linear relationship between the system speedup and the number of cores available for processing.

The authors have observed that for the virtual machines, the performance speedup remains almost constant for active processor cores in the range of up to eight virtual cores. The measured speed-up obtained using VMs over this range of cores was roughly constant (around 5.2).

Due to the way Intel implement their multi-threading strategy, it was noted that this theoretical performance applies to twelve cores (six real and six virtual) whereas the theoretical speedup is only equal to the number of real cores. The shaded area shown in Figure 3 represents the overall loss of speed-up between the measured performance and the theoretical performance and represents the portion of processor activity necessary to implement the hypervisor.

Discussion of Results

Although the authors were not able to measure the speed-up for 'bare-metal' machines having more than four cores, even at this number of cores, contention is already evident reducing the speed-up from a theoretical 4 to an actual value of 3.6.

The speed-up results found for virtual machines reinforce our belief that the massive drop-off in performance noted in the Gaussian results appears to be as a result of contention within the multicore processor rather than any software or Gaussian_09 issue. Had this drop-off been program related, then an equally substantial drop in speedup factor for the 8-core VM implementation would also have been evident.

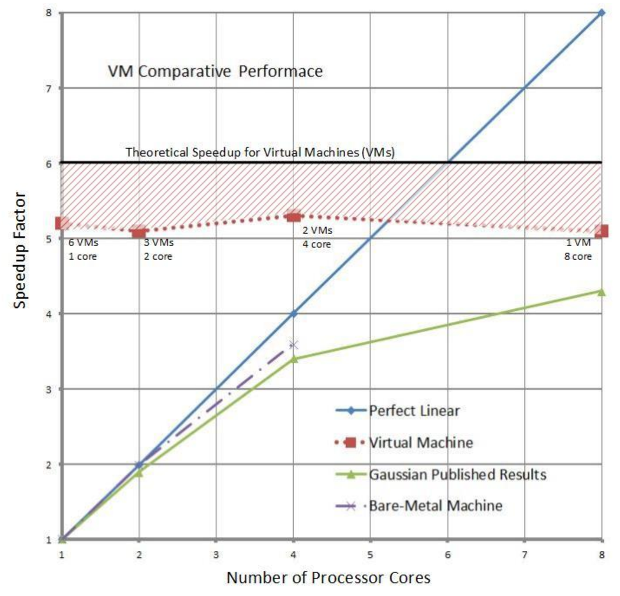

Figure 3 – VM Comparative Performance

According to the authors' findings, a six (hex) core processor runs more efficiently and hence gives better results when run using virtual machines (VMs) rather than in bare-metal mode. However much of this argument centres round the fact that VMs allow much better granular control over the available resources. Gaussian_09 32-bit software only appears to allow the use of four cores in SMP mode although clearly there are six cores available. The authors believe that the 64 bit Linux version would have allowed the use of six cores and hence would partly nullify this argument.

HPC Performance

Carrying out tests on individual machines as well as running the likely target software is necessary to obtain essential design data for any new cluster. However, it is the eventual performance of that cluster as a complete unit that is crucial to the success or otherwise of any high performance computing (HPC) cluster system. HPCs tend to be operated in two distinct modes representing the needs of the user. Compare the following cases:

1. In this case the user aims to obtain the results for a single or particular job as soon as possible.
2. In this case the user aims to obtain the results for a large batch of jobs as soon as possible.

In Case 1, the user attempts to complete a job in a fraction of the time it would otherwise take on a conventional machine without regard to the overall efficiency. The system is made into a single machine through the use of symmetric multiprocessing (SMP) between cores in a node and message passing interfaces (MPI) between nodes to ensure everything works cooperatively under the general guidance from the headnode.

In Case 2, the user wishes to obtain the result for a batch of jobs in the shortest possible time. The user aims to configure the system to maintain a peak level of efficiency during batch processing. In that way the overall time of a large batch of jobs can be minimised.

The Application Framework for Computational Chemistry (AFCC) is a framework specifically developed by the authors to process large batches of Gaussian_09 jobs, refer to Figure 4.

Separate instances of the command-line version of Gaussian_09 solver (G09.exe) run on each compute node. Each instance is passed information as to where to pick up input configuration details along with the location of the necessary input files (.gjf) and possible check files (.chk) for processing. These are then collected by the appropriate compute nodes for processing as well as returning the modified check files and output files (.out) back to a central collection point after processing. The node would then be 'flushed' to ensure a clean start for the next job on that node. This prevents any error in the processing of one job from affecting another.

As part of the AFCC framework, four directory structures were created on the headnode (g09-work1, g09-work2 and so forth) to process batches of jobs. New batches are loaded into one of these directories and the process started. A suite of six programs were written in C# to automate the process, namely:

i. G09prep - used to prepare the .gjf file to run efficiently on the cluster
ii. G09submit - used to send jobs to the scheduler
iii. G09local - accepts scheduler data and invokes local G09.exe program
iv. G09archive - isolate the results into an archive directory to allow re-use of the directory for the next batch
v. G09analyse - analyses the success and time taken by the node
vi. G09rerun - same as G09Prep but allows reuse of .chk files

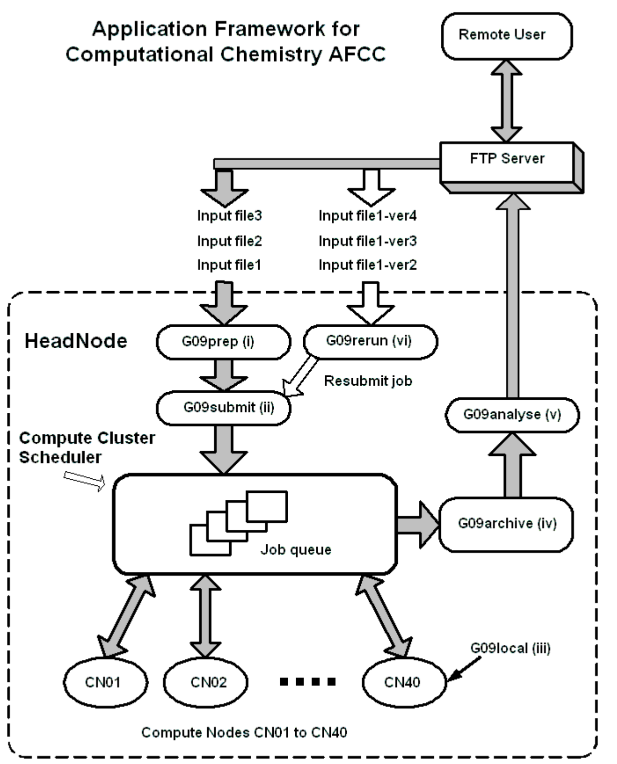

Figure 4 – Application Framework AFCC

The AFCC is not a job scheduler. However the six component programs comprising the AFCC described above operate in conjunction with the scheduler built into the Microsoft HPC server software to affect a highly efficient tailored solution to cluster management. The AFCC prepares jobs for execution and automatically loads large batches of jobs (typically 500 to 1000 jobs) onto the HPC scheduler under program control. The preparation programs (G09Prep and G09rerun) are used to configure the Gaussian_09 '.gjf' files such that each individual job can be processed in the most efficient manner at any of the cluster nodes.

While G09archive performs the necessary 'housekeeping' tasks to archive work on the cluster, G09analyse measures the success of the work as well as computing the ‘cost’ incurred for each job in terms of time spent within the nodes.

Access to these various programs is made via scripts that act upon specific working directories. In this way only a minimal set of high level directives are used to process many different batches of work on the HPC cluster.

Conclusions

It is the contention issue where the real interest lies and whether or not real progress can be gained using virtual machines rather than bare-metal machines for high performance clusters in the future. The interesting part lies in the fact that the speedup factor for virtual machines is essentially flat showing no sign of any contention issues as the number of virtual cores increase.

Virtual machines offer other potential advantages when used within a cluster environment. They can be kept minimal avoiding a great deal of overhead that goes with larger installations. They can also avoid known incompatibility issues where, for example, different versions of Java need to be installed. Also the fact that a VM can be readily backed up means that re-deployment back to a new VM installation can be quick and painless.

With a whole new breed of multicore processors soon to become readily available, it will be interesting to see if contention continues to dominate the ‘bare-metal’ implementations of these 16 core and 64 core processors. The authors are of the opinion that virtual machines (VMs) may well have a place in the newer high performance computing (HPC) clusters.

References

[1] S. H. Fuller and L. I. Millett, "Computing Performance: Game Over or Next Level?," *Computer*, vol. 44, pp. 31-38, 2011.

[2] S. Zhuravlev, S. Blagodurov, and A. Fedorova, "Addressing shared resource contention in multicore processors via scheduling," *SIGARCH Comput. Archit. News*, vol. 38, pp. 129-142, 2010.

[3] R. Hood, J. Haoqiang, P. Mehrotra, J. Chang, J. Djomehri, S. Gavali, D. Jespersen, K. Taylor, and R. Biswas, "Performance impact of resource contention in multicore systems," in *Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, 2010, pp. 1-12.

[4] A. Abel, F. Benz, J. Doerfert, B. Dörr, S. Hahn, F. Haupenthal, M. Jacobs, A. Moin, J. Reineke, B. Schommer, and R. Wilhelm, "Impact of Resource Sharing on Performance and Performance Prediction: A Survey," in *CONCUR 2013 – Concurrency Theory*. vol. 8052, P. D’Argenio and H. Melgratti, Eds., ed: Springer Berlin Heidelberg, 2013, pp. 25-43.

[5] J. Van Houten, "A Century of Chemical Dynamics Traced through the Nobel Prizes. 1998: Walter Kohn and John Pople," *Journal of Chemical Education*, vol. 79, p. 1297, 2002/11/01 2002.

[6] Gaussian. (2012, November 22). *Gaussian 09 Links* [Online]. Available: http://www.gaussian.com/g_tech/g_ur/m_linklist.htm

[7] Gaussian. (2012, November 22). *Get Your Gaussian Results Sooner* [Online]. Available: http://www.gaussian.com/g_prod/parallel.htm

[8] T. Leng, R. Ali, J. Hsieh, V. Mashayekhi, and R. Rooholamini, "An empirical study of hyper-threading in high performance computing clusters," *Linux HPC Revolution*, 2002.

Bibliography

R L WARRENDER is a Senior Lecturer at the University Of Sunderland within the Department of Computing Engineering Technology in the Faculty of Applied Sciences. Currently studying for a professional doctorate related to his work on cluster computing, he is working on collaborative research projects with different groups around the University who have high computational requirements. As well as being involved in the development and upgrade of the cluster, he has introduced cluster computing to the undergraduate teaching curriculum at the University.