The Optimal Resource Allocation Among Virtual Machines in Cloud Computing

Marjan Gusev
Faculty of Information Sciences and Computer Engineering
Ss. Cyril and Methodius University
Skopje, Macedonia
Email: marjan.gushev@finki.ukim.mk

Sasko Ristov
Faculty of Information Sciences and Computer Engineering
Ss. Cyril and Methodius University
Skopje, Macedonia
Email: sashko.ristov@finki.ukim.mk

***Abstract*—Virtualization is a key technology for multi-tenant cloud computing enabling isolation of tenants in one or more instances of virtual machines and sharing the hardware resources. In reality, modern multi-core multiprocessors also share the last level cache among all cores on one chip. Our goal will be to enable an optimal resource allocation by avoiding cache misses as much as possible, since this will lead to performance increase. In this paper, we analyze the performance of single and multi-tenant environments in cloud environment installed on a single chip multi core multiprocessor with different resource allocation to the tenants. We realize a series of experiments with matrix multiplication as compute intensive and memory demanding algorithm by varying the matrix size to analyze performance behavior upon different workload and variable cache requirements. Each experiment uses the same resources but it is orchestrated differently. Although one might think that virtualization and clouds include software overhead, the results show how and when cloud computing can achieve even better performance than traditional environment, both in a single-tenant and multi-tenant resource allocation for certain workload. The conclusions show that there are regions where the best performance in the cloud environment is achieved for cache intensive algorithms allocating the resources among many concurrent instances of virtual machines rather than in traditional multiprocessors using OpenMP.**

***Keywords-Cache memory*; *Cloud Computing*; *Matrix Multiplication*; *Shared Memory*; *Virtualization*.**

I. INTRODUCTION

Cloud Service Providers (CSPs) rent on-demand scalable hardware resources. The customers can use CPU, memory, and storage with arbitrary size and type in virtual machines (VMs) whenever they need. This flexibility results in dynamic resource workload. CSPs foster it even more by consolidating VMs on smaller number of physical servers in order to save power consumption. In such dynamic environment, customers' VMs are not totally isolated. They share same physical resources, especially CPU, memory and network. This paper focuses on CPU utilization when sharing among many concurrent VMs.

Cache memory is the CPU's key element in compute and memory intensive algorithms. Due to the performance impact of the cache, we define these algorithms as *cache intensive algorithms*. Matrix multiplication is an example of such algorithm that today's computations are using. This algorithm is compute intensive $O(n^3)$ and memory demanding $O(n^2)$.

Producers of modern multiprocessors must adopt caches for cloud computing especially in the multitenant, multiprocess and multithreading dynamic environment. For example, Intel introduces Intel Smart Cache [1] to improve the performance. Sharing the last level cache among multiprocessor's cores allows each core dynamically use the cache up to 100%. This technology can be used to increase the overall performance in cloud computing multi-tenant environment. Machina and Sodan in [2] developed a model that describes the performance of the applications as a function of allocated cache size, even if the cache is dynamically partitioned.

The fundamental driver for Multi-tenancy is Virtualization. It introduces additional layer and can provide better performance. The cache intensive algorithms run faster in distributed than shared cache memory virtual environment. Gusev and Ristov in [3] found that matrix multiplication algorithm can run faster in virtual environment compared to traditional, both by sequential and parallel executions (for problem sizes that fit in distributed L1 and L2 caches correspondingly). However, virtualization produces huge performance drawback for shared cache memory, even if it is dedicated per chip in multi chip multiprocessor. In this paper, we continue the performance analysis in cloud solution, compared to both virtual environment in guest operating system and traditional operating system. We expect that there are regions where the experiments will prove that cloud virtualization produces better performance and achieves better performance.

Koh et al. [4] describe the phenomenon that running the same VM on the same hardware at different times among the other active VMs will not achieve the same performance. They predict the performance scores of the applications under performance interference in virtual environments. VM granularity has a significant effect on the workload's performance for small network workload [5].

The experiments performed in this paper address several VM instances in a cloud system using different number of CPUs (assuming all cores are utilized). The introduction of a virtualization in the cloud is supposed to decrease the performance [6]. Our plan is to check validity of the

 ISBN: 978-1-61208-216-5

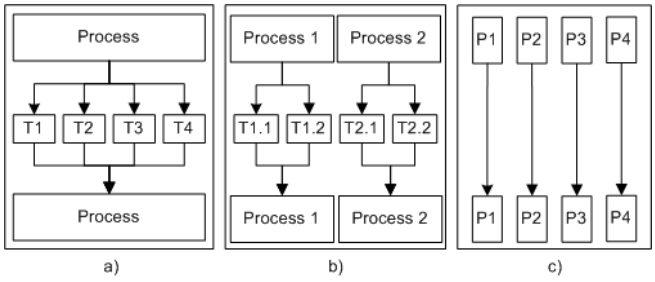

Figure 1. Test Cases in Traditional Environment

following hypotheses:

- Is there a region where cloud environment achieves better performance than traditional and virtual environment, and
- What is the performance of cloud computing with multi-VM environment in comparison to allocation of all resources to only one VM?

The rest of the paper is organized as follows: The testbed for three workload environments is described in Section II. Sections III and IV present the results of the experiments performed to determine the best environment for cache intensive algorithm and best resource allocation among process, threads and tenants correspondingly, while Section V presents the performance when the algorithm is executed sequentially on a single core. The results of the cache misses analysis are presented in Section VI to prove the causes for better / worse performance in L2 / L3 region for traditional and cloud environment. The final Section VII is devoted to conclusion and future work.

II. The Workload Environments

This section describes the testing methodology and defines the workload environments for experiments. Matrix multiplication algorithm is used as test data for both sequential and parallel execution. For all different environments, we plan to use the same hardware and operating system. The only difference is inclusion of virtual machines and enabling cloud environment.

A. Traditional Environment

This environment consists of Linux Ubuntu Server 11.04 installed on Dell Optiplex 760 with 4GB DDR2 RAM and Intel(R) Core(TM)2 Quad CPU Q9400 @ 2.66GHz [7]. The multiprocessor has 4 cores, each with 32 KB 8-way set associative L1 cache dedicated per core and 8-way set associative L2 cache with total 6 MB shared by 3MB per two cores.

Three different parallel executions are defined as test cases 1.1, 1.2 and 1.3 in this environment, as depicted in Fig. 1. The sequential execution is determined as test case 1.4.

1) Case 1.1: 1 process with 4 (max) threads on total 4 cores: In this test case the matrix multiplication is executed by one process using 4 parallel threads as presented in Fig. 1 a). Each thread runs on one core multiplying the whole matrix $A_{N \cdot N}$ and a column block of matrix $B_{N \cdot N/4}$.

2) Case 1.2: 2 different processes with 2 threads per process on total 4 cores: In this test case two concurrent processes execute matrix multiplication. Each process uses two parallel threads as shown in Fig. 1 b). Each process multiplies the whole matrix $A_{N \cdot N}$ and a half of matrix $B_{N \cdot N/2}$ divided vertically. Each thread multiplies matrix $A_{N \cdot N}$ and half of $B_{N \cdot N/2}$, i.e., $B_{N \cdot N/4}$.

3) Case 1.3: 4 different processes with 1 thread per process (sequentially) on total 4 cores: In this test case 4 concurrent processes execute matrix multiplication as depicted in Fig. 1 c). Each process multiplies the whole matrix $A_{N \cdot N}$ and a quarter of matrix $B_{N \cdot N/4}$ divided vertically.

4) Case 1.4: 1 process sequentially on 1 core: In this test case, one process executes matrix multiplication sequentially on one core, i.e., three cores are unused and free. The process runs on one core multiplying the whole matrix $A_{N \cdot N}$ with the whole matrix $B_{N \cdot N}$.

B. Virtual Environment

This environment consists of the same hardware and operating system as described in Section II-A. Additionally new VM is installed with same Linux Ubuntu Server 11.04 using VirtualBox and Kernel-based Virtual Machine virtualization standard (KVM). All available resources (4 cores) are allocated to the only one VM for parallel execution and only one core for sequential execution.

Two test cases are performed in this environment one with parallel and the other with sequential execution.

1) Case 2.1: 1 VM with 1 process with 4 (max) threads on total 4 cores: In this test case one process executes matrix multiplication by 4 parallel threads, all in the VM. Each thread runs on one core multiplying the whole matrix $A_{N \cdot N}$ and a column block of matrix $B_{N \cdot N/4}$.

2) Case 2.2: 1 VM with 1 process sequentially on total 1 core: In this test case one process executes matrix multiplication sequentially in VM on one core, i.e., three cores are unused and free. The process runs on one core multiplying the whole matrix $A_{N \cdot N}$ with the whole matrix $B_{N \cdot N}$.

C. Cloud Virtual Environment

Cloud virtual environment is developed using OpenStack Compute project [8] deployed in dual node as depicted in Fig. 2. KVM virtualization standard is also used for VMs. One Controller Node and one Compute Node are used.

This cloud virtual environment consists of the same hardware and operating system as described in Section II-A for Compute Node server. Virtual Machine described in Section II-B is instantiated in one or more instances for the four test cases that are performed in this environment.

 ISBN: 978-1-61208-216-5

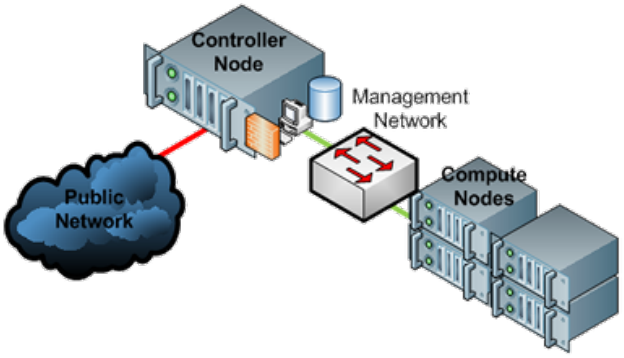

Figure 2. OpenStack dual node deployment [9]

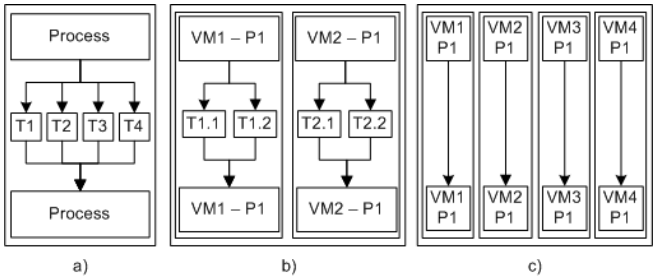

Figure 3. Test Cases in Cloud Virtual Environment

Three test cases 3.1, 3.2 and 3.3 are performed as parallel executions in this environment depicted in Fig. 3. The test case 3.4 for sequential execution is defined as one instance of VM with one sequential process.

1) Case 3.1: 1 instance of VM with 1 process with 4 (max) threads per process on total 4 cores: This case is similar as cases 1.1 and 2.1, i.e., one instance of VM is activated in the Cloud allocated with all 4 cores as depicted in Fig. 3 a). One process in VM executes matrix multiplication with 4 paralllel threads. Each thread runs on one core multiplying the whole matrix $A_{N \cdot N}$ and a column block of matrix $B_{N \cdot N/4}$.

2) Case 3.2: 2 concurrent instances of VM with 1 process per VM with 2 threads per process on total 4 cores: In this test case two concurrent instances of same VM are activated in the Cloud allocated with 2 cores per instance as depicted in Fig. 3 b). One process in each VM executes matrix multiplication concurrently with 2 parallel threads per process (VM). Each process (in separate VM) multiplies the whole matrix $A_{N \cdot N}$ and a half of matrix $B_{N \cdot N/2}$ divided vertically. Each thread multiplies matrix $A_{N \cdot N}$ and half of $B_{N \cdot N/2}$, i.e., $B_{N \cdot N/4}$.

3) Case 3.3: 4 concurrent instances of VM with 1 process per VM with 1 thread per process (sequentially) on total 4 cores: In this test case, 4 concurrent instances of same VM are activated in the Cloud allocated with 1 core per instance as depicted in Fig. 3 c). Each process (in separate VM) multiplies the whole matrix $A_{N \cdot N}$ and a column block of matrix $B_{N \cdot N/4}$.

4) Case 3.4: 1 instance of VM with 1 process sequentially on total 1 core: This case is similar as test case 3.1. The difference is that only one core is dedicated to the only VM, i.e., three cores are unused and free. The process runs on one core multiplying the whole matrix $A_{N \cdot N}$ with the whole matrix $B_{N \cdot N}$.

D. Test Goals

The test experiments have two goals:

- The first goal is to determine if the additional virtualization layer in cloud drawbacks the performances compared to traditional or virtualized operating system when all the resources are dedicated to only one tenant and multi-threading is used.
- The second goal is to determine which resource allocation among tenants and threads provides best performance in the traditional environment and in the cloud.

Different sets of experiments are performed by varying the matrix size changing the processor workload and cache occupancy in the matrix multiplication algorithm.

III. Traditional vs Virtual vs Cloud Environment Performance with all Resources Allocated

This Section presents the results of the experiments performed on three workload environments when all the resources (CPU cores) are rented to one tenant, i.e., test cases 1.1, 2.1 and 3.1 as described in Section II.

Fig. 4 depicts the speed in gigaflops that matrix multiplication achieves for different matrix size N when executing one process concurrently using 4 threads on 4 cores on three same hardware resources, but different system environments as described in Section II. The curves are identified by V(4)T for traditional environment, V(4)V for environment with virtual and V(4)C with cloud environment. Fig. 5 shows only the differences of achieved speeds in Fig. 4 using relative presentation of the ratio to the default speed value obtained by traditional environment.

Two regions with different performance for all three test cases are clearly depicted in Fig. 4; the left one with higher speed and the right one with lower speed. The first region is the L2 region as defined in [3] (the region for such matrix size N that will enable storage of all memory requirements in L2 cache and avoid generation of cache misses for reusing the same data on L2 level). The second region is the region where the matrices can not be stored completely in the L2 cache and many L2 cache misses will be generated due to re-using of data, but memory requirements will fit in the L3 cache (if it exists). This region is called the L3 region. We must note that those matrices that fit in L1 region are too small to produce higher speed.

 ISBN: 978-1-61208-216-5

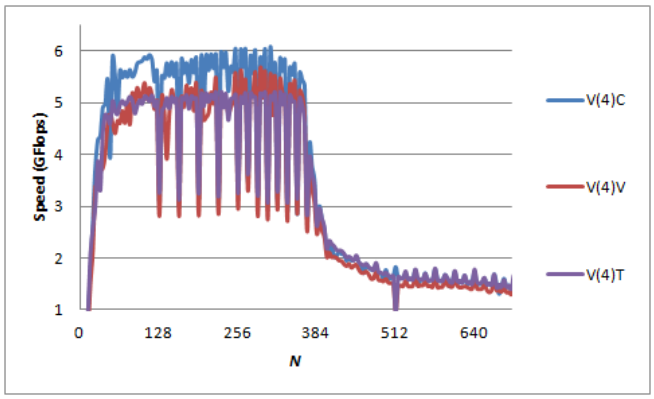

Figure 4. Speed comparison for traditional / virtual machine allocated with all hardware resources (4 threads)

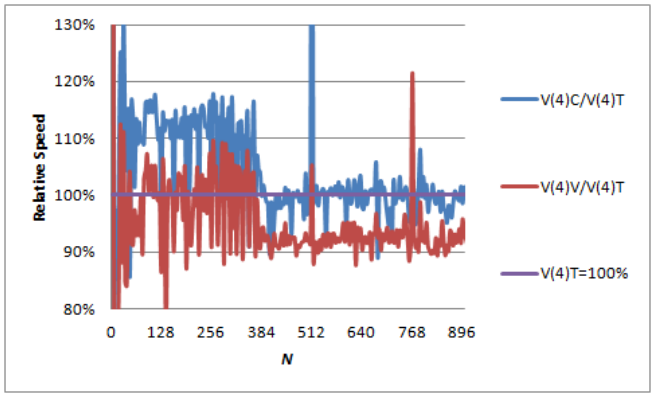

Figure 5. Relative speed comparison for Fig. 4.

Analyzing the performance by comparing the three curves in figures 4 and 5, we can conclude that cloud virtualization performs the algorithm better than other two environments in the L2 region. Virtualization also performs better than traditional environment in the same L2 region, but produces worse performance in points where performance drawbacks appear due to cache set associativity described in [10]. Cloud and traditional environments provide similar performance in L3 region, i.e., shared main memory, much better than virtual environment. The conclusion is that in this region virtualization provides the worst performance and cloud environment achieves the best performance.

Another important conclusion is the fact that the speed increases in the L2 region where the cache memory is dedicated per core (group of 2 cores) for virtual and cloud environments. However, the speed decreases in the shared memory L3 region when matrix size N increases demanding more memory requirements, generating higher cache miss penalty and increasing the overall memory access time.

Based on results of these experiments, we can conclude that cloud virtual environment achieves better performance compared to traditional environment for cache intensive algorithms in the L2 region using dedicated L2 cache per core and shared L3 cache and main memory. Section VI describes the causes for this phenomenon.

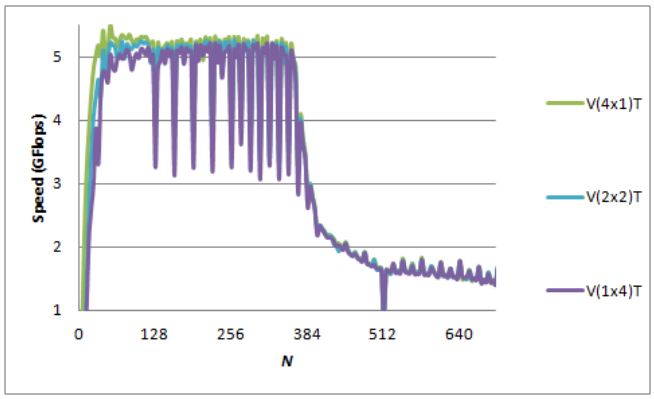

Figure 6. Speed comparison for traditional machine allocated with different resources per thread

IV. Multiprocess, Multithread and Multitenant Environment Performance

This section presents the results of the experiments performed on traditional and cloud workload environment when the resources (cores) are shared among processes, threads and tenants in different ways.

A. Multiprocessing and Multithreading in Traditional Environment

This Section presents the results of the experiments that run test cases 1.1, 1.2 and 1.3 described in Section II, i.e., different resource allocation per process in traditional environment.

The achieved speed for the matrix multiplication algorithm is presented in Fig. 6 in gigaflops for different matrix size N executing with 1, 2 and 4 processes using total 4 threads on all 4 cores on the same traditional environment. By V(1x4)T, we denote the results obtained for environment defined in the test case 1.1, V(2x2)T the test case 1.2 and V(4x1)T the test case 1.3.

The same two regions (L2 and L3) are depicted in Fig. 6 identified by different speed performance for all 3 test cases.

The relative ratio of achieved speeds in comparison to the traditional environment defined in test case 1.1 with 1 process and 4 parallel processes is presented in Fig. 7.

Comparing the obtained curves in figures 6 and 7 we can conclude that environment for test case 1.3 is the leader in the speed race in front of case 1.2 and 1.1 for the L2 region. All test cases provide similar performance in the L3 region where the best performance is achieved by test case 1.3.

The fact that the speed is almost linear in the L2 region where cache memory is dedicated per core (group of 2 cores) is also an important conclusion. However, the speed decreases for all 3 test cases in the shared memory L3 region

 ISBN: 978-1-61208-216-5

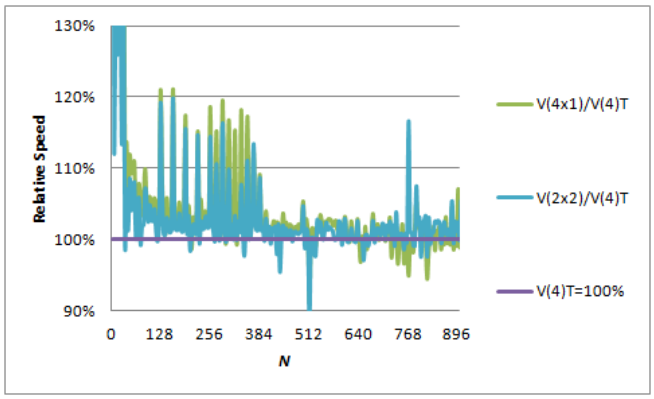

Figure 7. Relative speed comparison for Fig. 6.

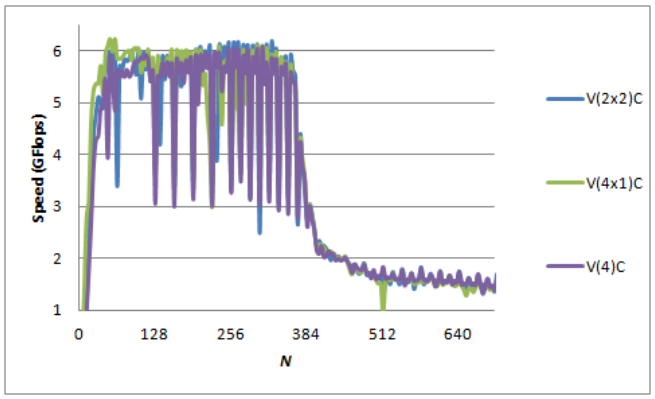

Figure 8. Speed comparison for virtual machine(s) in cloud allocated with different resources per machine and per thread

when the matrix size N is increased and higher cache miss penalty is generated.

We can conclude that dividing the problem in separate processes is the best solution for cache intensive algorithms in the L2 region. The OpenMP handles better in the L3 region by allocating all the resources to one process that executes concurrently with maximum number of threads equal to the number of cores.

B. Multi-tenant / Multi-threading in Virtual Cloud Environment

This section presents the results of the experiments that run test cases 3.1, 3.2 and 3.3 described in Section II with different resource allocation per tenant in cloud virtual environment.

The speed achieved for the matrix multiplication algorithm is presented in Fig. 8 for different matrix size N of the matrix multiplication executing on one, two and four VM using total 4 threads on all 4 cores on the same cloud virtual environment. The curves are identified by V(4)C for test case 3.1, V(2x2)C for test case 3.2 and V(4x1)C for test case 3.3. The relative differences to the default speed V(4)C are presented in Fig. 9.

Fig. 8 presents that the same two regions L2 and L3 can be identified by different performance for all 3 test cases.

Analyzing the performance behavior presented in figures 8 and 9 we can conclude that the environment defined by test case 3.3 is the leader in the speed race in front of the test cases 3.2 and 3.1 for the left part of the L2 region, and the environment for test case 3.2 is the leader for the speed race in front of the test cases 3.3 and 3.1 in the right part of the L2 region. All test cases provide similar performance in the L3 region with test 3.1 as a leader.

We can also conclude that the speed increases in the L2 region where cache memory is dedicated per core (group of 2 cores) for all three test cases. However, the speed decreases for all test cases in the shared memory L3 region when the matrix size N is increased enough and higher cache miss

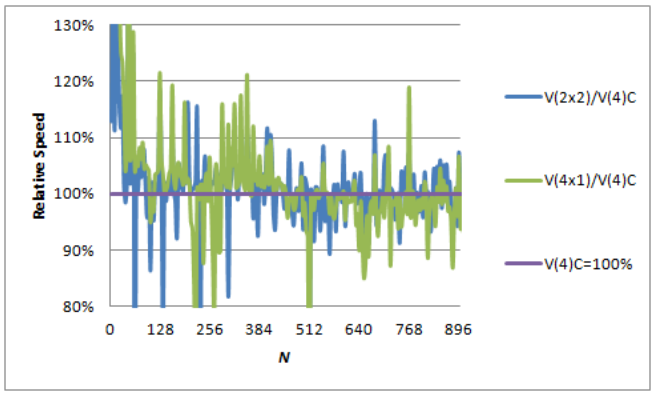

Figure 9. Relative speed comparison for Fig. 8

penalty is generated increasing the overall memory access time.

Dividing the problem in separate concurrent VMs is the best solution for cache intensive algorithms in the L2 region for dedicated L2 caches. The best solution for the L3 region with shared main memory is to allocate all the resources to one process (VM) to be executed concurrently with maximum threads as number of cores.

V. Traditional vs Virtual vs Cloud Environment Performance for Sequential Execution

This section presents the results of the experiments performed on three workload environments for sequential execution, i.e., test cases 1.4, 2.2 and 3.4 as described in Section II.

The achieved speed for execution of the matrix multiplication algorithm is shown in Fig. 10. The figure depicts the speed in gigaflops for different matrix size N when executing one process sequentially on one core on three different system environments as described in Section II. The curves are identified by V(1)T for traditional environment,

 ISBN: 978-1-61208-216-5

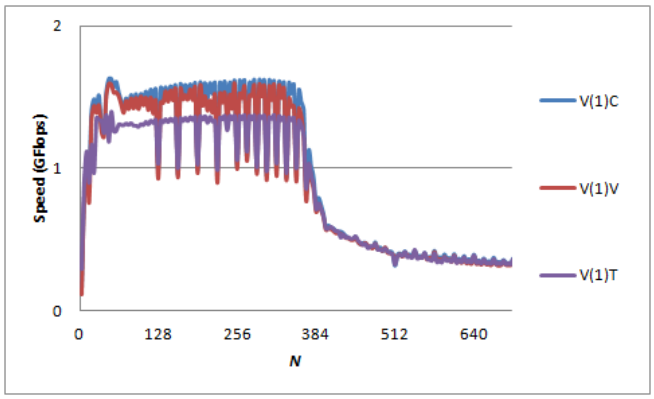

Figure 10. Speed comparison for sequential execution in the three environments

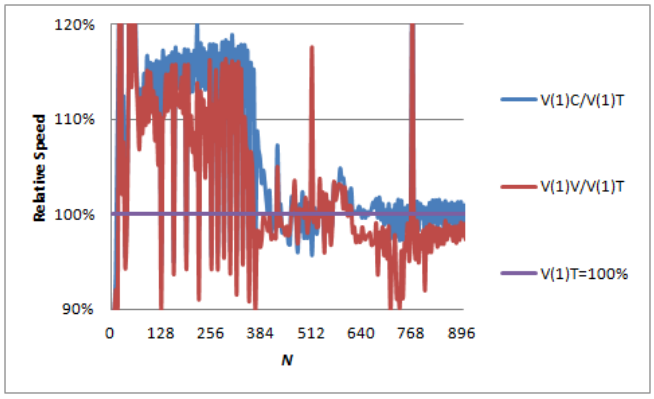

Figure 11. Relative speed comparison for Fig. 10.

V(1)V for environment with virtual and V(1)C with cloud environment.

The performance analysis of the curves in figures 10 and 11 shows that cloud virtualization achieves better performance for the algorithm execution in the L2 region. Virtualization also performs better than traditional environment in the same L2 region. Cloud and traditional environments provide similar performance in the L3 region, better than virtual environment. The conclusion is that in this region virtualization provides the worst performance and cloud environment the best performance.

VI. Cache Miss Analisys

This section presents the results of the experiments realized using Valgrind [11] to prove why the algorithm runs better in cloud environment in L2 region and runs better in traditional environment in L3 region. L1 and L2 cache misses are analyzed for both L2 and L3 regions for sequential execution in traditional and cloud environment. Table VI presents the results of these experiments. L1DT and L1DC identifies the number of L1 data cache misses for traditional and cloud environment correspondingly, and

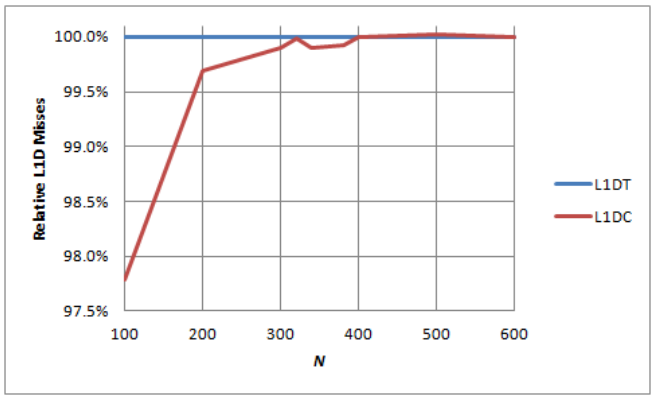

Figure 12. Relative comparison for L1 data cache misses

L2DT and L2DC for the number of L2 data cache misses for both environments correspondingly.

Table I
Number of L1 and L2 data cache misses in cloud and traditional environment in some points in L2 and L3 regions

N	L1DT	L1DC	L2DT	L2DC
100	145,572	142,344	11,553	9,176
200	1,039,954	1,036,676	22,807	20,432
300	3,448,511	3,445,020	41,580	48,158
320	33,600,548	33,597,359	46,329	72,598
340	5,011,438	5,006,561	51,501	83,472
360	5,941,929	5,936,645	60,225	94,021
380	7,093,110	7,087,590	100,675	106,888
400	68,438,786	68,435,517	113,818	119,676
500	113,364,842	113,385,000	187,027	666,141
600	244,545,355	244,541,890	765,609	27,234,248

A. L1 Data Cache Misses

The relative ratio of L1 data cache misses in comparison to the traditional environment is depicted in Fig. 12. We can conclude that cloud environment achieves smaller number of L1 data cache misses than the traditional environment in the L2 region, and comparable number of L1 data cache misses in the L3 region.

B. L2 Data Cache Misses

The relative ratio of L2 data cache misses in comparison to the traditional environment is depicted in Fig. 13. We can conclude that cloud environment achieves smaller number of L2 cache misses than the traditional environment in the L2 region, but much more than traditional environment in the L3 region.

VII. Conclusion and Future Work

Several experiments including sequential and parallel executions are performed with different resource allocation in traditional, virtual and cloud environments on the same multiprocessor. The testing methodology addresses each environment with full utilization to all CPU cores with

 ISBN: 978-1-61208-216-5

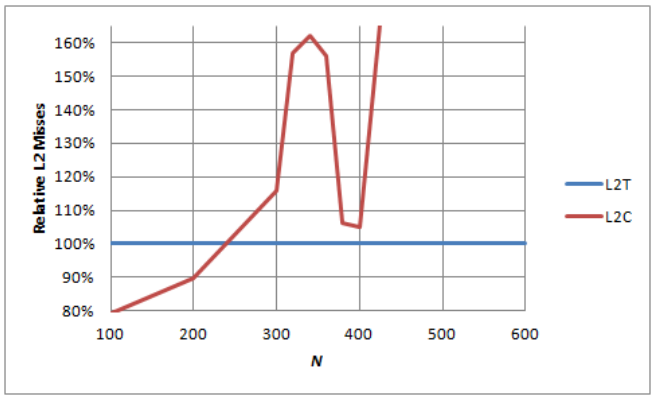

Figure 13. Relative comparison for for L2 data cache misses

different techniques: mono-process with multi-threading, multi-processes with multi-threading and multi-processes with single threads.

Cache intensive algorithm is the algorithm which is computationally intensive and memory demanding, i.e., utilizes the cache with data reuse to perform several computations. Simple matrix multiplication algorithm is used for sequential execution and 1D blocking matrix B for parallel execution to efficiently utilize cache performance. Our goal is not to create a new algorithm which exploits super linear speedup in cloud environment, but to examine the cache memory usage phenomenon and its performance impact in cloud.

The conclusions brought from the experiments performed in this paper are summarized to the impact of the resource allocation. Dividing the algorithm to parallel cores enables usage of more L1 and L2 cache for parallel version in comparison to the traditional environment, phenomenon explained in [3] for L1, L2 and L3 regions for multiprocessors using shared L2 cache and distributed L1 cache.

The experiments performed in this paper address several virtual machine instances in a cloud system using different number of CPUs (assuming all cores are utilized). Each experiment orchestrates the CPU cores differently. The contribution of the paper can be summarized as:

- The experiments prove that there is a region (L2 region) where cloud environment achieves better performance than traditional and virtual environment, both for parallel and sequential process execution, and
- The experiments prove that cloud computing provides better performance in a multi-VM environment, rather than allocating all the resources to only one VM.

The best resource allocation for traditional environment for cache intensive algorithms is the usage of multiple processes with single threads. Multiple VMs with single threads is the best resource allocation for cloud environment. Comparing the environments, cloud computing provides the best performance.

Future multiprocessors will have more cores and cache on chip with different cache types and results of this research will have higher impact. Our plan for further research is to continue with performance analysis of cloud computing on different hardware and cloud platforms with different hypervisors to analyze CPU behavior with different cache organization and the best platform for cache intensive algorithms. Experiments with MPI for inter VM instance communication are planned as future research.

REFERENCES

[1] Intel. Intel smart cache. [retrieved: May, 2012]. [Online]. Available: http://www.intel.com/content/www/us/en/architecture-and-technology/intel-smart-cache.html

[2] J. Machina and A. Sodan, "Predicting cache needs and cache sensitivity for applications in cloud computing on cmp servers with configurable caches," in *Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing*, ser. IPDPS '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 1–8. [Online]. Available: http://dx.doi.org/10.1109/IPDPS.2009.5161233

[3] M. Gusev and S. Ristov, "Matrix multiplication performance analysis in virtualized shared memory multiprocessor," in *MIPRO, 2012 Proc. of the 35th Int. Convention*, 2012, pp. 264–269.

[4] Y. Koh, R. Knauerhase, P. Brett, M. Bowman, Z. Wen, and C. Pu, "An analysis of performance interference effects in virtual environments," in *Performance Analysis of Systems Software, 2007. ISPASS 2007. IEEE International Symposium on*, april 2007, pp. 200 –209.

[5] P. Wang, W. Huang, and C. Varela, "Impact of virtual machine granularity on cloud computing workloads performance," in *Grid Computing (GRID), 2010 11th IEEE/ACM International Conference on*, oct. 2010, pp. 393 –400.

[6] B. Xiaoyong, "High performance computing for finite element in cloud," in *Future Computer Sciences and Application (ICFCSA), 2011 International Conference on*, June 2011, pp. 51 –53.

[7] cpu world. Intel(r) core(tm)2 quad cpu q9400. [retrieved: May, 2012]. [Online]. Available: http://www.cpu-world.com/sspec/SL/SLB6B.html

[8] Openstack. Openstack compute. [retrieved: May, 2012]. [Online]. Available: http://openstack.org/projects/compute/

[9] ——. Openstack dual node. [retrieved: May, 2012]. [Online]. Available: http://docs.stackops.org/display/documentation/Dual+node+deployment

[10] S. Ristov and M. Gusev, "Achieving maximum performance for matrix multiplication using set associative cache," in *Next Generation Information Technology (ICNIT), 2012 The 3rd International Conference on*, 2012, pp. 542–547.

[11] Valgrind. Valgrind. [retrieved: May, 2012]. [Online]. Available: http://valgrind.org/

 ISBN: 978-1-61208-216-5