Optimal Resource Scaling for HPC in Windows Azure

Sasko Ristov, Marjan Gusev, Selvir Osmanovic, and Kujtim Rahmani

Ss. Cyril and Methodious University, Faculty of Information Sciences and Computer Engineering,
Rugjer Boshkovikj 16, 1000 Skoipje, Macedonia
sashko.ristov@finki.ukim.mk, marjan.gushev@finki.ukim.mk,
selvir_sk@hotmail.com, kujtim.rahmani@gmail.com

Abstract. Microsoft Windows Azure Cloud offers scalable resources to its customers. The price for renting the resources is linear, i.e. the customer pays exactly double price for double resources. However, not always all offered resources of virtual machine instances are most suitable for the customers. Some problems are memory demanding, others are compute intensive or even cache intensive. The same amount of resources offered by the cloud can be rented and utilized differently to speedup the computation. One way is to use techniques for parallelization on instances with more resources. Other way is to spread the job among several instances of virtual machine with less resources. In this paper we analyze which is the best way to scale the resources to speedup the calculations and obtain best performance for the same amount of money needed to rent those resources in the cloud.

Keywords: Cloud Computing, HPC, Matrix Multiplication

1 Introduction

Cloud computing makes the virtualization and grid computing commercially available. It offers infinite available computing resources (processing power or storage capacity) organized in different virtual machine (VM) instances. Customers can rent as many VMs and use them as long as they need. Scientists can collaborate with each other sharing the data in the cloud [1].

Standard types of instances that are currently offered on the market have linear price / performance ratio. Scaling the resources (CPU, RAM memory) with factor x scales the price with the same factor x. More details about current offers for renting VMs can be found in [9, 3, 2].

However, all problems do not scale well and evenly. Fixed-size speedup (Amdahl's law) bounds speedup to the amount dependent of sequential fraction of the algorithm; the fixed-time speedup is less than scaled speedup bounded to the linear speedup [7]. Also, the performance for particular algorithm does not depend only on CPU and RAM memory. There are a lot of other parameters that impact the performance such as I/O, storage capacity, CPU cache architecture, communication latency runtime environment, platform environment etc. The authors in

[8] discovered several pitfalls resulting in waste of active virtual machines idling. They examine several pitfalls in Windows Azure Cloud during several days of performing the experiments: Instance physical failure, Storage exception, System update. Windows Azure does not work well for tightly-coupled applications [11]. The cloud virtualization generates less cache misses for cache intensive algorithms and thus better performance in both single-tenant and multi-tenant resource allocation for certain workload [5]. Virtualization performance is even better than traditional host for distributed memory, but it provides huge performance drawback in shared memory [4].

The authors in [6] show that dense matrix multiplication algorithm (MMA) runs better on Windows operating system with its C# and threading runtime environment than Linux operating system with OpenMP for parallelization that are hosted in Windows Azure. In this paper we continue the research to analyze the best resource orchestration among the VM instances on the better Windows platform hosted also on Windows Azure. Since the prices of standard VMs grows linearly as resource growth, we determine how to achieve maximum performance for the same price executing dense MMA. We analyze both sequential execution on one core and parallel on maximum 8 cores.

The rest of the paper is organized as follows: Section 2 presents used testing methodology and different hardware environments in Azure Cloud. Section 3 shows the results of the experiments realized to find the performance of the parallelization on particular infrastructure. The final Section 4 is devoted to conclusion and future work.

2 Testing Methodology

This section describes the testing methodology based on 4 different infrastructures with same platform.

2.1 Testing Algorithm

Dense MMA is used as test data. For simplification, we multiply square matrices of same sizes $C_{N \cdot N} = A_{N \cdot N} \cdot B_{N \cdot N}$. Each element c_{ij} of matrix C is calculated as $c_{ij} = \sum_{k=0}^{N-1} a_{ik} * b_{kj}$ where a_{ik}, b_{kj} are correspondingly elements of matrices A, B, for all $i, j, k = 0, \ldots, N-1$.

Matrix elements are stored as double precision numbers with 8 bytes each. One thread multiplies the whole matrices $A_{N \cdot N}$ and $B_{N \cdot N}$ for sequential test cases. For parallel test cases each thread multiplies row matrix $A_{N \cdot N/c}$ and the whole matrix $B_{N \cdot N}$ where $c = 2, 4, 8$ denotes the total number of parallel threads.

2.2 Testing Environments

Testing environment is hosted in Windows Azure. The authors in [10] presents Windows Azure Platform, its components and architecture in details. We use the same platform environment in each test case with different resource allocation. Windows 2008 Server is used as operating system in each VM instances. Runtime environment consists of C# with .NET framework 4 and threads for parallelization.

The same hardware resources are organized in different Windows Azure VMs:

- 1 x Extra Large VM with total 8 CPU cores;
- 2 x Large VM with total 4 CPU cores per VM;
- 4 x Medium VM with total 2 CPU cores per VM;
- 8 x Small VM with total 1 CPU core per VM.

AMD Opteron 4171 HE processor(s) is used in each VM. It has 6 cores, but maximum 4 of 6 cores are dedicated per VM instance. Each core possesses 64 KB L1 data and instruction caches dedicated per core, 512KB L2 dedicated per core. L3 cache with total 5 MB is shared per chip.

2.3 Test Cases

We realize the experiments in each test case varying matrix size to analyze performance behavior upon different VM resources and variable cache requirements.

Test Case 1: 1 VM with 1 process with 8 (max) threads per process on total 8 cores. In this test case one Windows Azure Extra Large VM is activated allocated with 8 cores as depicted in Figure 1 a). One process in VM executes matrix multiplication with 8 parallel threads. Each thread runs on one core multiplying a row block of matrix $A_{N \cdot N/8}$ and the whole matrix $B_{N \cdot N}$.

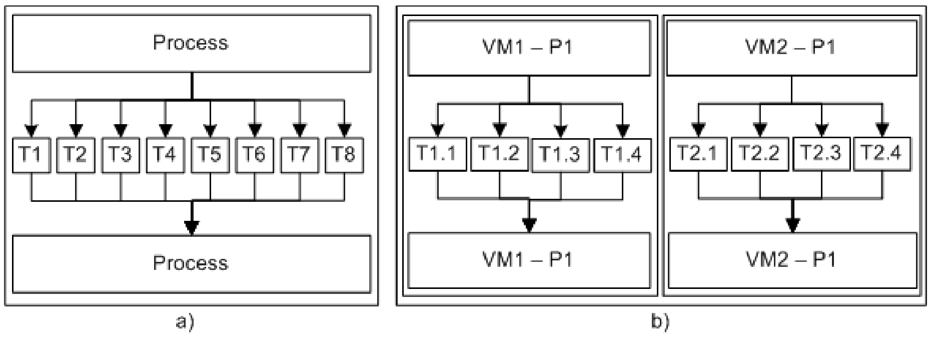

Fig. 1. Test Cases 1 (a) and 2 (b)

Test Case 2: 2 concurrent VMs with 1 process per VM with 4 threads per process on total 8 cores. Two concurrent Windows Azure Large VMs are activated allocated with 4 cores per VM as depicted in Figure 1 b). One process in each VM executes matrix multiplication concurrently with 4 parallel threads per process (VM). Each process (in separate VM) multiplies the half of matrix $A_{N \cdot N}$ divided horizontally, i.e. a row matrix $A_{N \cdot N/2}$ and matrix $B_{N \cdot N}$. Each thread multiplies a quarter of half matrix A, i.e. $A_{N \cdot N/8}$ and matrix $B_{N \cdot N}$.

Test Case 3: 4 concurrent VMs with 1 process per VM with 2 threads per process on total 8 cores. In this test case four concurrent Windows Azure Medium VMs are activated allocated with 2 cores per VM as depicted in Figure 2 a). One process in each VM executes matrix multiplication concurrently with 2 parallel threads per process (VM). Each process (in separate VM) multiplies the quarter of matrix $A_{N \cdot N/4}$ divided horizontally and matrix $B_{N \cdot N}$. Each thread multiplies a half of quarter of matrix A, i.e. $A_{N \cdot N/8}$ and matrix $B_{N \cdot N}$.

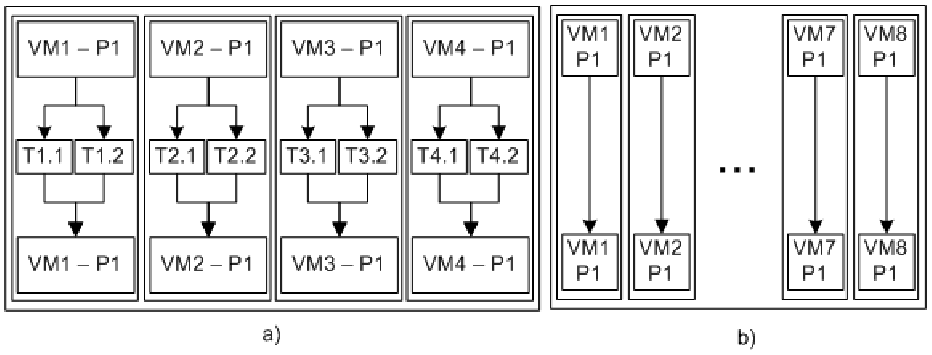

Fig. 2. Test Cases 3 (a) and 4 (b)

Test Case 4: 8 concurrent VMs with 1 process per VM with 1 thread per process on total 8 cores. In this test case eight concurrent Windows Azure Small VMs are activated allocated with 1 core per VM as depicted in Figure 2 b). One process in each VM executes matrix multiplication concurrently with 2 parallel threads per process (VM). Each process (in separate VM) multiplies the quarter of matrix $A_{N \cdot N/4}$ divided horizontally and matrix $B_{N \cdot N}$. Each thread multiplies a half of quarter of matrix $A_{N \cdot N/4}$, i.e. $A_{N \cdot N/8}$ and matrix $B_{N \cdot N}$.

Test Cases 5-8: sequential execution on only one core. Test cases 5-8 execute matrix multiplication sequentially on the testing environments as test cases 1-4 correspondingly. Only one core is used in each of these test cases and all other seven cores are unused and free. The process runs on one core multiplying the whole matrix $A_{N \cdot N}$ with the whole matrix $B_{N \cdot N}$.

2.4 Test Data

Speed V is measured for each test case as defined in (1). Average execution time of all processes per test case is measured to compare the speed of different test cases.

$$V = 2 \cdot N^3 / AverageExecutionTime \qquad (1)$$

2.5 Tests Goal

The test experiments has the goal to determine which hardware resource allocation among tenants and threads provides best performance for HPC application in Windows Azure.

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

3 The Results of the Experiments

This section presents the results of the experiments that run test cases. We measure the average speed for each test case and analyze their dependencies of different hardware resource allocation, that is, we compare the results of test cases 1 and 5, 2 and 6, 3 and 7, and 4 and 8.

Figure 3 depicts the speed of test cases 1 and 5 for different matrix size N. We determine two main regions with different performance. For $N < 572$ (L_3 region) the whole matrices can be placed in L3 cache and performance are much better than for $N > 572$ (L4 region) where L3 cache misses are generated.

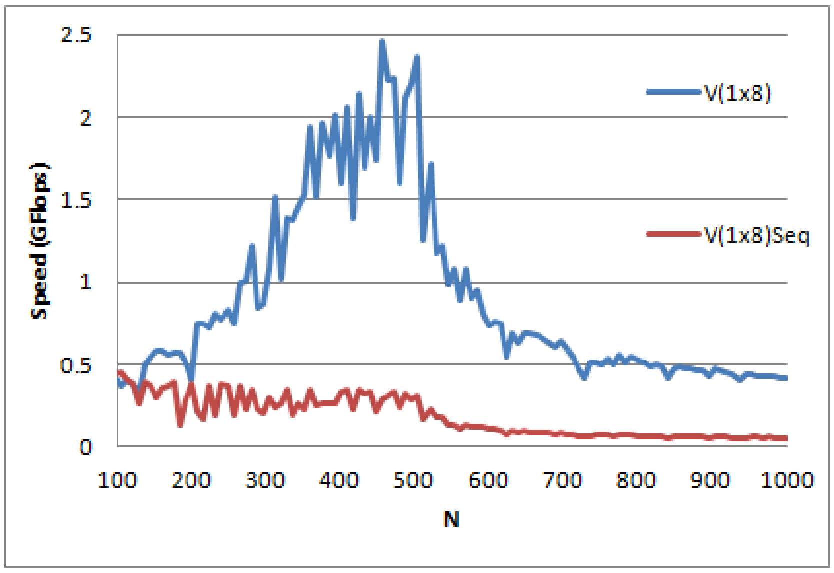

Fig. 3. Speed for test cases 1 and 5

Figure 4 depicts the speed of test cases 2 and 6 for different matrix size N. The same regions with different performance are found also.

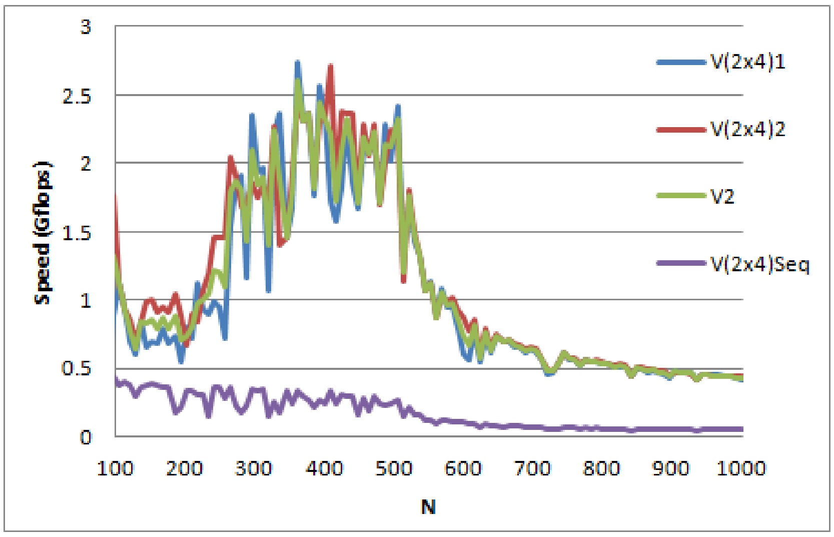

Fig. 4. Speed for test cases 2 and 6

Figure 5 depicts the speed of cases 3 and 7 for different matrix size N. As depicted, there is a huge performance discrepancy in L_3 region among the processes and the average speed.

Figure 6 depicts the speed of test cases 4 and 8 for different matrix size N. We also found a performance discrepancy but more emphasized in L_1 and L_2 regions which are dedicated per process and thread in test case 4 since each process has only one thread, each VM has only 1 core and L1 and L2 caches are dedicated per that core.

4 Conclusion and Future Work

Dense MMA is compute intensive, memory demanding and cache intensive scaled and granular algorithm. Therefore it is optimal algorithm for high performance computing. In this paper we analyze the performance of MMA in Windows Azure Cloud using single-tenant and multi-tenant environments with single-threading and multi-threading hosted on the same hardware resources but differently spreaded among virtual machines.

The results of the experiments for sequential execution are as expected. Extra Large VM achieves maximum speed in front of Large, Medium and Small in L_2 region. MMA algorithm achieves maximum speed when executed parallel on 8

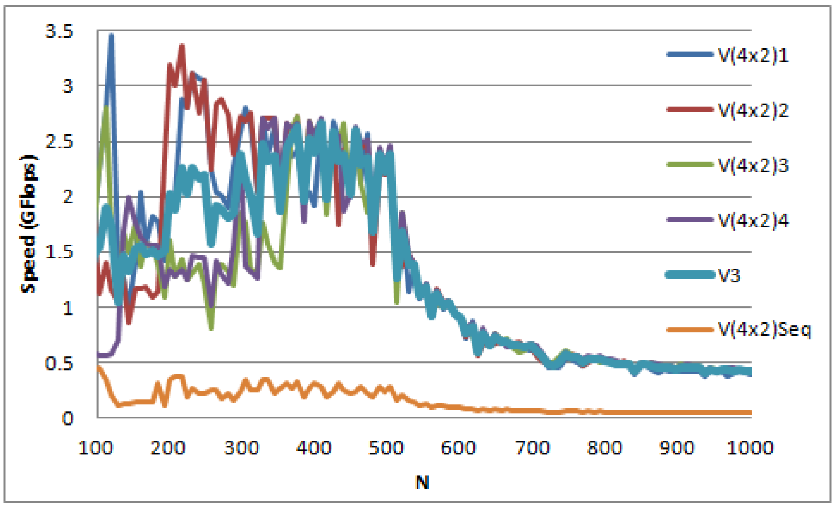

Fig. 5. Speed for test cases 3 and 7

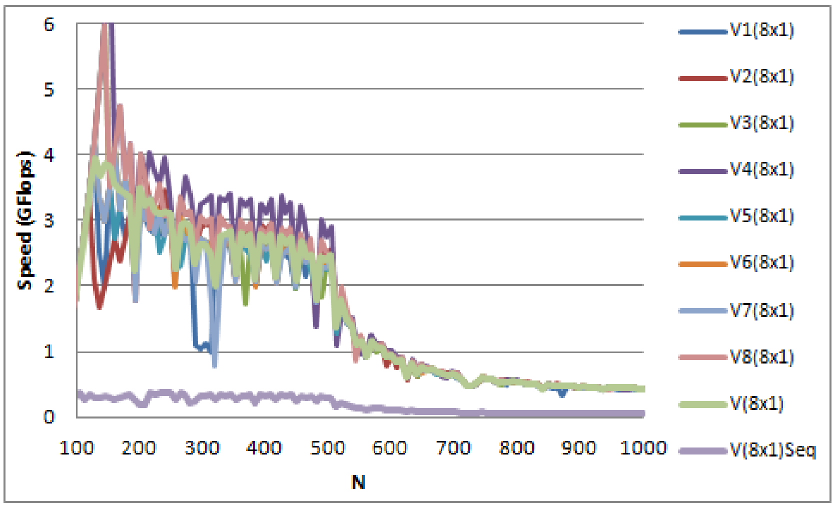

Fig. 6. Speed for test cases 4 and 8

x Small instances, in front of 4 x Medium, 2 x Large, and 1 x Extra Large in L2 and L3 regions, and almost all observed L4 region.

We will continue with research on other hardware architectures and different clouds since MMA depends on CPU and cache memory.

References

1. Ahuja, S., Mani, S.: The state of high performance computing in the cloud. Journal of Emerging Trends in Computing and Information Sciences 3(2), 262–266 (Feb 2012)
2. Amazon: Ec2 (July 2012), `http://aws.amazon.com/ec2/`
3. Google: Compute engine (July 2012), `http://cloud.google.com/pricing/`
4. Gusev, M., Ristov, S.: Matrix multiplication performance analysis in virtualized shared memory multiprocessor. In: MIPRO, 2012 Proc. of the 35th International Convention, IEEE Conference Publications. pp. 264–269 (2012)
5. Gusev, M., Ristov, S.: The optimal resource allocation among virtual machines in cloud computing. In: Proc. of 3rd Int. Conf. on Cloud Computing, GRIDs, and Virtualization (CLOUD COMPUTING 2012). pp. 36–42 (2012)
6. Gusev, M., Ristov, S.: Superlinear speedup in windows azure cloud. Tech. Rep. IIT:06-12, University Ss Cyril and Methodius, Skopje, Macedonia, Faculty of Information Sciences and Computer Engineering (July 2012)
7. Gustafson, J., Montry, G., Benner, R.: Development of parallel methods for a 1024-processor hypercube. SIAM Journal on Scientific and Statistical Computing 9(4), 532–533 (July 1988)
8. Lu, W., Jackson, J., Ekanayake, J., Barga, R.S., Araujo, N.: Performing large science experiments on azure: Pitfalls and solutions. In: CloudCom'10. pp. 209–217 (2010)
9. Microsoft: Windows azure (July 2012), `http://www.windowsazure.com/pricing/`
10. Padhy, R.P., Patra, M.R., Satapathy, S.C.: Windows Azure Paas Cloud: An Overview. International J. of Computer Application 1, 109–123 (Feb 2012)
11. Subramanian, V., Ma, H., Wang, L., Lee, E.J., Chen, P.: Rapid 3d seismic source inversion using windows azure and amazon ec2. In: Proceedings of the 2011 IEEE World Congress on Services. pp. 602–606. SERVICES '11, IEEE Computer Society, Washington, DC, USA (2011)