Evaluation of Virtual Machine Scalability on Distributed Multi/Many-core Processors for Big Data Analytics

Amril Nazir, Yaszrina Mohamad Yassin, Chong Poh Kit, Ettikan Kandasamy Karuppiah
MIMOS Bhd, Technology Park Malaysia
57000 Kuala Lumpur, Malaysia
amrilnurman.nazir,yaszrina.yassin, pk.chong,ettikan.karuppiah@mimos.my

***Abstract*—Cloud computing makes data analytics an attractive preposition for small and medium organisations that need to process large datasets and perform fast queries. The remarkable aspect of cloud system is that a nonexpert user can provision resources as virtual machines (VMs) of any size on the cloud within minutes to meet his/her data-processing needs.**

In this paper, we demonstrate the applicability of running large-scale distributed data analysis in virtualised environment. In achieving this, a series of experiments are conducted to measure and analyze performance of the virtual machine scalability on multi/many-core processors using realistic financial workloads. Our experimental results demonstrate it is crucial to minimise the number of VMs deployed for each application due to high overhead of running parallel tasks on VMs on multicore machines. We also found out that our applications perform significantly better when equipped with sufficient memory and reasonable number of cores.

I. Introduction

Using the Infrastructure-as-a-service (IaaS) cloud computing model, users today can request dynamically provisioned, virtualized resources such as CPU, memory, disk, and network access in the form of virtualized resources. The user typically requests resources based on computational needs and pays for virtualised resources based on their capacity and time utilized. Virtualisation enables users to deploy their application workloads on multiple virtual machines (VMs). The VM can be customized with a specific operating system, specific applications and/or services.

This possibility make data analytics an attractive preposition for small and medium organisations that need to process large datasets and perform fast queries in cloud environment. The remarkable aspect of such cloud system is that a nonexpert user can provision virtual machines (VMs) of any size on the cloud within minutes to meet his/her data-processing needs. This feature gives tremendous power to the end user.

Typically, data analysis is being carried out over large data sets. Hence, users still need to be aware and concerned of how the datasets should be distributed across compute resource nodes or virtual machines. This involves important steps such as vm provisioning (how many VMs to deploy), data partitioning (how to partition data), and data scheduling (how to schedule data across VMs). More recently, the MapReduce framework has been introduced by Google to eliminate the burden of the user by enabling automatic and dynamic partitioning of data set across multiple virtual machines in a fault-tolerant manner. For example, an open-source implementation of MapReduce paradigm is Hadoop, which has been widely used for large-scale distributed analysis e.g., web indexing and data mining. However, the performance of running application can vary according to the capacity of the physical machine, the number of virtual machines (VMs) deployed on the machine, the number of cores assigned to each VM, the amount of memory assigned to VMs and any several other factors. Therefore, mapping these virtual resource requests to physical hardware could vary and this could potentially cause variations in the performance of applications deployed on such resources.

In this paper, we evaluate the effects of VM provisioning on multi-core machines of which dual and quad cores are now commonplace. Using realistic financial processing workloads, we experimentally study the effects of VM provisioning behaviours and their impact on application performance. A series of experiments are conducted to measure and analyze the performance of performing large-scale financial distributed analysis in cloud environment where financial tasks run on different size/configuration of the VMs.

This paper is organised as follows. Section II provides an overview of the overall architecture and the crucial steps involved in real-world financial application. Section III summarises specific computational needs and resource requirements for the financial application. Section IV describes our methodology and experimental setup for performance evaluation and we subsequently present our benchmark results. Section V and VI discuss related work and future work. Finally, section VII concludes the paper.

II. A case study of real-world distributed financial application

In this section, we present the crucial steps involved in running a real-world distributed financial application. We demonstrate how such application can be executed in distributed manner to perform large-scale data analysis on very large data sets. Figure 1 illustrates an overall architecture of our financial application. In summary, the financial application comprises three main components: data extraction, data preprocessing, and data analysis. In the next subsection, we will

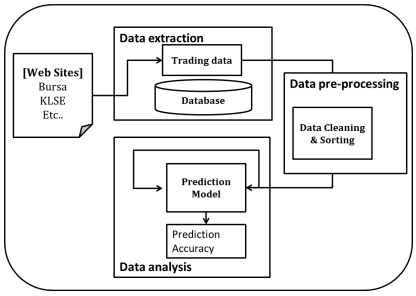

Figure 1. Overall Architecture of Financial Application

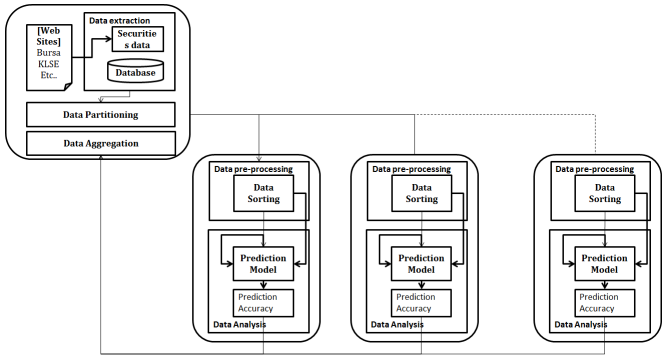

Figure 2. An overview of the distributed financial application architecture

briefly describe each of these components.

A. Data Extraction

The first step involves identifying the source and collecting the data. In this case, the source of financial data is extracted and collected from various of websites that provide historical data and quotes on equities, futures and REITs (Real Estate Investment Trusts). The financial websites are normally available from official Stock Exchange websites (e.g., Bursa, NASDAQ, LSE etc.) any other reliable financial data service providers such as KLSE, and Yahoo! Finance. The financial data is mined directly from the websites and they are extracted from raw html pages.

B. Data Pre-processing

After collecting the raw data, an important step is to transform it into the appropriate format for data analysis. As a matter of fact, data cleaning often consumes a great deal of computational power since the raw data in html and text format needs to be cleaned and streamlined as transactional records based its primary keys (i.e., security code and date), and any mismatched records and duplications must be identified and eliminated. The data pre-processing also involves text matching of different fields of records from different web sources. Furthermore, the records need to be sorted and partitioned based on a number of primary keys such as the security code and date of transactions. Therefore, the effort to organise raw data for even a few hundreds of megabytes can take considerable amount of time.

C. Data Analysis

Once the data has been cleaned and pre-processed, the data will be further analysed based on specific financial model or algorithm. In practice, even after data pre-processing, the data to analyse can still be too large to analyse in a centralised manner. For example, the same financial model may need to be computed on different securities. The size of historical records for each security can reach up to a few thousands, and multiple securities will need to be processed in parallel before final computation can be computed. Based on the complexity of the algorithm and the chosen parameter, each financial model or prediction model can run from a few seconds up to a few hours for each security. This does not include the time it takes to merge all the intermediate values from multiple transactional records and compute the final value. This is where we expect the cloud resources could be useful to distribute these tasks in parallel for on-demand financial analysis. It is expected that cloud resources would further improve the data analysis by exploiting the parallelism and fault-tolerance in the most effective manner.

Figure 2 further illustrates an overview of the financial application architecture for distributed processing. We can observe two new components: data partitioning and data aggregation. At the completion of data extraction, the data is stored in the database. At this point, the data partitioning component retrieves the data sets from the database, then partition the data into a number of smaller segments to be distributed over multiple virtual machines. Each VM will be responsible to perform data pre-processing and data analysis based on the segmented data. The result or the output of the analysis (e.g., prediction accuracy) is sent to the data aggregation component. The data aggregation component collects and retrieves the computation result from each VM upon task completion.

In the next section, we summarise the computational needs for all the crucial steps and processes we have described above.

III. Computational Needs for Financial Application

In the previous section, we have described the crucial steps needed for large-scale distributed financial data analysis. In this section, we summarise the specific computational requirements and features inherited from our financial application.

A. Data-Intensive

Specific to financial application, the most common requirement is the extraction of financial data from source. The source of data here is the websites which publish latest financial data such as equities, bonds, and futures on daily basis. The data extraction process is data-intensive because there is a need for high-memory capable machines to extract the relevant data from various websites and consolidate the data in specific format that can be used directly by the application to perform data analysis.

B. Compute-intensive

Since there is huge amount raw html and text data which is needed to be processed and transformed into the appropriate

format for data analysis, the data pre-processing process is also computationally intensive. However, the most intensive part of computation is the data analysis process because many of the financial algorithms involve computationally intensive operations and complex processes.

C. Summary

In this section, we have identified the need for high-memory capable machines for data extraction process. We also identified the need to execute financial tasks in distributed manner due to their highly computationally-intensive operations. Financial applications are therefore known to be highly computationally and data intensive. It is envisaged that cloud computing would enable users to run their financial application workloads in a distributed manner providing faster results for on-demand financial analysis.

IV. Evaluation

A. Methodology and Experimental Setup

Our experimental hardware consists of a four node cluster. The test-bed comprises four physical machines, namely phy_1, phy_2, phy_3, and phy_4. Physical machine phy_1 is equipped with 8 processors with Intel(R) Xeon(R) 2.4GHz, RAM of 10.5 GB, running Ubuntu 12.04 with Xen hypervisor. Physical machine phy_2 has quad-core Intel® Xeon® 2.53GHz with 12GB RAM whereas physical node phy_3 has 8 Intel® Core™ i7 3.2 GHz processors with 3GB RAM. Both phy_2 and phy_3 run Ubuntu 10.04 with KVM hypervisors. On the other hand, physical machine phy_4 is equipped with four 2.00GHz Xeon processors, 3.2GB of memory and 14GB of disk, runs Ubuntu Release 10.04 (lucid) with kernel 2.6.33, and is connected with 1 Gigabit Ethernet.

In VM-based environments, we use Virtual Machine Manager 0.8.2 to monitor running VMs. The VMs are running with RHEL5 with kernel 2.6.22. The same cluster nodes is used to obtain performance results for both the VM-based environment and the native, non-virtualized environment. Each physical machine is installed with Xen hypervisor and the Virtual Machine Manager initiates the provisioning and deployment of a new VM. On each physical node, we assume the VM image is persistent thereby there is no need to transfer VM image from a storage server to the physical node each time a new VM is deployed.

Our datasets contain historical financial data of Malaysian securities from Bursa Malaysia, KLSE, and Yahoo! Finance. Initially, we run each experiment under different size of securities portfolio (i.e., 4, 8, 16, and 32 securities) as to capture the effects of increasing workload. At later stage of our experiments, we hold the number of securities fixed for each experiment to investigate the behaviours of mixed VM configurations under the same workload.

Our financial application is run on RapidMiner 5.2. RapidMiner, formerly YALE (Yet Another Learning Environment) provides data mining and machine learning procedures including: data extraction and transformation, data preprocessing and visualization, modelling, evaluation, and deployment. The data mining processes can be made up of arbitrarily nestable operators, described in XML files and created in RapidMiner's graphical user interface (GUI). RapidMiner is written in the Java programming language.

All core components of our financial application: data extraction, data pre-processing, and data processing, data partitioning, and data aggregation are modelled as operators to form a single workflow in RapidMiner. Hence, unlike any other MapReduce applications that require users to operate at API-level (i.e., map and reduce functions), RapidMiner enables all financial steps to be created as a workflow by using graphical user interface (GUI). However, to leverage parallelism on a single VM with more than one virtual core (multi-core virtual CPUs), we made some modifications to RapidMiner to spawn n multiple threads based on n number of virtual cores available on the VM.

All results described in this paper are obtained using RapidMiner 5.2, while the data is stored as csv files in the database repository. Each experiment is repeated three times and an average is obtained. For a RapidMiner workflow that runs in parallel on multi-core virtual CPUs, we also repeat the experiment three times and obtain an average of its minimum and maximum execution time to capture variations in processor contention. In particular, we are interested in the maximum execution time as that would give indication of the worst-case scenario when running in multi-core virtual CPUs.

B. Benchmarks

In this section, we conduct performance test on our financial application in a cloud environment involving multi-core physical machines. Performance evaluation was conducted to answer the following question: what is the performance impact when running financial tasks on a variety of virtual machine configurations?

The first experiment is to measure (i) the amount of time it takes for a financial task to be served with an increasing data sets (financial securities data) and (ii) the amount of time it takes for the financial task to complete its execution under a variety configurations of VMs. Each experiment is repeated 3 times and an average is obtained. For a parallel task running on multi-core VMs, we also repeated the experiment 3 times to obtain an average of its minimum and maximum execution times. To leverage parallelism on multi-core VM, we enable our financial application to spawn n multiple threads based on n number of physical cores available.

Size of Securities	1 core	4 core	1 core per VM	4 cores per VM	4 VMs 1 core each
4	10	(6,7)	12	(8,9)	(11,13)
8	19	(13,14)	24	(16,17)	(22,24)
16	38	(19,20)	46	(27,28)	(49,50)
32	66	(20,21)	104	(25,34)	(36,128)

Table I
THE IMPACT OF TIME EXECUTION (IN SECONDS) UNDER INCREASING SIZE OF SECURITIES. (x, y) DENOTE THE MINIMUM EXECUTION TIME x AND MAXIMUM EXECUTION TIME y

Figure I presents the breakdown of the total time for running different security sizes under physical machines and different configurations of VMs with increasing size of securities. The experiment is run on phy_4 machine, equipped with 4-cores,

3.2GB RAM, and 96GB hard disk capacity. We carry out benchmark with 3 different VM configurations: a single VM with 1-core (1-core VM), 4-cores on a single VM (4-core VM), and 4 VMs assigned 1 core each. From the measurments, we can observe that the execution time of running on 4-core VM is significantly faster that running the same financial task sequentially on 1-core VM. We can see almost linear improvement in execution time (approximately 3.2x faster) as the financial task is running on parallel on 4-core VM. This is to be expected as the application is utilising all processors for computation.

When comparing the performance of sequential run on a physical machine versus a sequential run on a VM (1-core VM), it can be seen that performance on the VM is reduced by 58%. Similarly, when comparing the performance of a parallel run on a physical quad-core machine versus a parallel run on a single VM with 4 cores (4-core VM), the performance of a parallel run on 4-core physical machine is slightly better by 29%. We can see that the performance gap between the physical and virtual machine is getting less significant as the number of cores is increased on the VM.

Size. of Securities	1 core per VM	4 cores per VM	8 cores per VM
32	73	(16,17)	(119,1430)

Table II
COMPARISON OF TIME TAKEN FOR DIFFERENT NUMBER OF VIRTUAL CORES PER VM.

For subsequent experiments, we fix the number of securities to 32 units then measure the total execution time against different VM configurations. We run the experiment on phy_1 which is twice as powerful compared to phy_4. The total amount of memory allocated to all VMs for each experiment is fixed to 2048MB while the number of cores assigned to each VM is also varied. Furthermore, for each run, the 32-units of securities size are split equally in parallel based on the number of physical cores.

Table II presents our results for constant size of 32 securities with 3 different set of VM configurations: 1 core per VM (1-core VM), 4 cores per VM (4-core VM), and 8 cores per VM (8-core VM). We can observe that it would take approximately 73 seconds to execute 32 units of securities on a single VM with 1 core. On the other hand, as we increase the number of cores to 4 in a single VM, the execution time reduces greatly by approximately 3.3 times to execute the same task. However, as we increase the number of virtual cores beyond four cores, we can observe that the execution time increases significantly. This shows when we increase the number of cores from 4 to 8 on a single VM on a quad-core machine. It can be seen that the minimum execution time for an 8-core VM is 1 minute and 59 seconds whereas the maximum execution time significantly reaches up to 23 minutes 50 seconds. We are seeing a huge difference in performance when compared to the execution on a 4-core VM. This shows that there is a point of bottleneck where the performance somewhat degrades significantly as we increase the number of virtual cores on a single VM. At first instance, the result seems to be counter-intuitive as the performance should not be affected adversely as the virtual cores do not overrun the number of physical cores. The total number of virtual cores i.e., 8-cores is in fact proportional to the number of physical cores – the phy_1 has 8-cores. However, the best plausible explanation for this poor performance is most likely due to the low memory assigned to each core. As we increase the number of cores, we still retain the memory capacity to 2048MB. Hence, as the number of cores is increased up to 8, each core is allocated with 256MB memory only. As a result, this has a significant impact on the performance as the financial task is struggling to execute efficiently when the machine is low on memory. This is possibly due to the frequent memory swaps between the physical hard disk and virtual machine memory.

Size. of Securities	1 core per VM	4 cores per VM	8 cores per VM
32	71	(12,13)	(14,18)

Table III
COMPARISON OF TIME TAKEN FOR DIFFERENT NUMBER OF VIRTUAL CORES PER VM FOR A HIGHER MEMORY OF 10GB ALLOCATED TO THE PHYSICAL MACHINE.

To test our hypothesis, we repeated the same experiment but we assign a much higher memory of 10GB to the VM. Similar to our previous experiment, the number of virtual cores is set to 8 and the size of securities is fixed at 32 units. Figure III illustrates our result. We can observe that our hypothesis is indeed true. When assigning 10GB of RAM memory with 8-cores (proportion to the number of real cores on physical machine) on a single VM, we can observe tremendous improvement of execution – the minimum execution time is 14 seconds whereas the maximum time is only 18 seconds. This demonstrates that assigning sufficient amount of memory is highly important when assigning higher number of virtual cores. Interestingly, we can observe that the performance of 4-core VM with 2048MB virtual memory is almost equivalent to that of 8-core with 10GB virtual memory. This is somewhat interesting which indicates that the number of virtual cores assigned to a single VM does not guarantee a linear increase in performance even when there is sufficient memory. In fact, from our results, it can be seen that there is only a minor improvement in performance. Therefore, from this, we conclude that when determining where to schedule VM, sufficient memory allocation should be given higher priority than the number of virtual cores. This holds true for our example of financial data analysis application.

We may also ask this question: does actual capability of the physical machine has a significant impact on application performance if we were to allocate two identical VMs with the same number of virtual cores and the same amount of virtual memory on two separate machines with different physical machine capabilities? To do this comparison, we compare the total execution time for running a fixed 32 unit size of securities on the powerful phy_1 machine running a single VM with 4 virtual cores and compare it with the less powerful phy_4 machine running a single VM with 4 virtual cores. Each machine is allocated with 2048MB of virtual memory for the experiment. Interestingly, we observe noticeable differences on performance. As for the low-end phy_4 machine, we can

observe that average time it takes to complete the task is within 34 seconds. However, for the powerful phy_1 machine, it takes in average 17 seconds to complete the same task. This demonstrates that the actual physical core capacity and memory of the machine does indeed play an important factor in determining how fast task can finish its execution. As can be observed in our case, the time it takes to complete the task is two times faster on phy_1 in comparison to phy_4 machine even when both VMs are identical. The most plausible explanation is due to the fact that phy_1 has twice as number of cores and almost four times higher memory in comparison to phy_4 machine. Therefore, we conjecture that the mapping of virtual machines on different physical machine configurations have significant impact on the application performance.

Size. of Securities	4 cores per VM	2 VMs with 2 cores each
32	(29,32)	(37,112)

Table IV
THE TOTAL AMOUNT OF TAKEN TO EXECUTE 32 UNITS OF SECURITIES (IN SECONDS) FOR 4 VIRTUAL CORES PER VM VERSUS 2 VMS WHICH ARE ASSIGNED 2 VIRTUAL CORES EACH

Next, we investigate the impact of assigning number of virtual cores to multiple VMs. We try to answer the following question: can we get similar performance when assigning a single VM with 4 virtual cores in comparison to assigning 2 VMs with 2 virtual cores each? Similarly, we conducted the experiment on phy_4 physical machine. Table IV presents our results. We can clearly observe that 4 virtual cores on a single VM outperforms the performance of 2 VMs with 2 virtual cores each. Interestingly, when we compare the minimum execution time between the two, the performance gap is not significant: there's only a difference of 8 seconds. However, the overall performance between the two becomes very significant when we consider the maximum execution time. We can observe that the maximum execution time reaches up to 1 minute and 52 seconds for 2VMs with 2-cores whereas the maximum execution time of a single VM with 4-cores only reaches up to 32 seconds. In this case, there is a 2.5x degradation in performance. Hence, we advocate that it is more efficient to minimise the number of running VMs if it is all possible (if it can be avoided) since running multiple VMs have significant effects on the performance of task execution, as we have observed from our example of financial task execution. Moreover, from our performance results on a wide variety of VM configurations, we have observed that it is in fact more efficient to allocate sufficient memory to each physical core when running our financial application.

So far, we have conducted our experiments by initiating VM provisioning directly to the Xen hypervisor. We now aim to further test our conjecture by carrying out similar experiments on OpenNebula Cloud to compare the performance of non-optimized VM provisioning in standard cloud environment against an approach that makes intelligent decision based on the discovery of physical resource capacity (i.e., current available memory and number of CPU cores etc). In OpenNebula Cloud, the standard VM provisioning is based on a round robin scheduling strategy. OpenNebula aims to select the first physical node that is found in its database in such way that nodes are selected one after another until one node is found that can run the VM.

Based on our experiments earlier, we identified that each processor core should run at least on 1024MB of memory for optimal task execution. Hence, the discovery and scheduling should be made based on the following rules: schedule and deploy a single VM with n virtual cores to any of the physical machine as long $av \geq m{\times}n$ where av is the available machine memory and m is the optimal memory set by the user. In our case, m is set to 1024MB. Hence, if the number of virtual cores available on the machine is 4, the VM should only schedule on a machine with at least 4096MB of memory. On the other hand, we compare it to the approach where each virtual core is assigned to a single VM with at least 1024MB.

Again, the test-bed comprises four physical machines, namely phy_1, phy_2, phy_3, and phy_4. We timed certain major cloud operations, measuring the cost of individual operations, as well as the total overhead time required to schedule and provision VMs. These include the measurement of the invocation costs (delays) of the discovery time, scheduling time and VM deployment time, as well as the execution time.

	1 virtual core per VM	**Schedule if $av \geq m{\times}n$**
Discovery+Scheduling Time	0.6 (600 ms)	0.6 (600 ms)
Deployment Time	3	3
Execution Time	104	8
Total Completion Time	**107.6 secs**	**11.6 secs**

Table V
COMPARISON OF TIME TAKEN FOR 1 VIRTUAL CORE PER VM PROVISIONING METHOD VS. OUR INTELLIGENT VM PROVISIONING METHOD THAT PROVISIONS VMS BASED ON AVAILABLE MEMORY AND THE NUMBER OF AVAILABLE VIRTUAL CORES.

Table V presents the breakdown of the total overhead time taken to discover, schedule, deploy VM, and execute task on VM for standard VM deployment versus our intelligent discovery approach. As can be observed, the total completion time for 1-core VM provisioning method is significantly higher (10 times higher) than our intelligent provisioning method. This demonstrates that in cloud environment, the way how we provision VMs even have greater impact on the application performance due to higher variations of physical machine capacity in cloud environment.

V. RELATED WORK

Ever since the advent of cloud-enabling technologies such as virtualization, a lot of studies [2], [1], [4], [10], [9] have been devoted to their applicability and gauging performance for various application domains as well as scientific and High Performance Computing (HPC) applications.

Venkatraman et. al. conducted a benchmarking effort of virtual machines on multi-core machines [12]. The performance evaluation was conducted using a series of independent benchmarks, testing the performance of a database server, java server and web server. The multi-workload benchmark is used to evaluate the two server machines and study how the different server workloads interact to utilize the hardware resources.

Some studies have reported successful porting of traditional HPC workloads onto public cloud environments [6], [7], [5]. The effect of virtualization on new generation programming models and environments for data-intensive applications like Hadoop has been explored in [8]. Schad et. al. carried out an extensive study on long-term performance variance of big data analytics on Amazon EC2 [11]. Moreover, Dejun et. al show that different supposedly identical VM instances often have very different performance in cloud environment, up to a ratio 4 from each other [3]. From their experiments, they also reported that they could not see homogeneous performance of an application even across identical VM instances.

In our work we conducted our experiments using realistic financial workloads at the hypervisor level and on a private OpenNebula cloud test-bed. This also allows us to design and test various VM configurations on physical multi-core machines in a controlled environment which enables us to determine the behaviours of different VM configurations on a variety of physical multi-core machines. To our knowledge, our work is one of the earliest attempts that evaluate the virtual machine scalability for large-scale distributed data analysis using realistic financial workloads as the base of our study.

VI. Future Work

There are several directions for future work. We have evaluated the performance of real-world financial applications and have found that effective discovery, scheduling, and load balancing of virtual machines are essential. Therefore, in future work, we aim to investigate various mechanisms that enable effective discovery, scheduling and load balancing in virtualised environment. However, understanding the nature of application (parallelizable) is important to optimally utilize available multi/many core machines. For instance, different application domains impose different loads on available physical resources which may effect on how we do discovery, scheduling, and load balancing. Examples vary between compute, memory or potentially I/O bound applications.

Moreover, hardware accelerators like Graphical Processing Units (GPUs) are highly attractive for computationally intensive tasks since they consist of many-core processors, and support thousands of concurrent threads. GPUs have been reported to deliver substantial speedups (between 10X and 100X) over multi-core CPUs for highly computational tasks. For our financial scenario, we identify that some financial processing such as data extraction and sorting operations can be ideally outsourced to GPUs. Therefore, we also aim to investigate further on the applicability and mechanisms of leveraging GPUs for our financial tasks.

VII. Conclusion

This work presents preliminary results on the true impact of provisioning VMs in cloud environment for large-scale distributed data analysis, focusing mainly on financial applications. We hope to continue our investigation using more different types of applications in the future and evaluate the effects of VM provisioning in much larger VM cluster deployments and possibly public cloud services.

Our experimental results demonstrate it is crucial to minimise the number of VMs deployed for each application due to high overhead of running parallel tasks on VMs on multi-core machines. We also found out that our applications perform significantly better when equipped with sufficient memory and reasonable number of cores.

Guided by these observations, we propose that applications should be able to initiate resource discovery and directly influence VM scheduling and load balancing to take advantage of the number of cores and memory available by dedicating cores and memory for specific application operations. We further conclude that there is a need for more effective discovery, scheduling and load balancing than the typical round robin approaches employed by current Cloud middleware systems such as OpenNebula and Eucalyptus.

References

[1] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, SOSP '03, pages 164–177, New York, NY, USA, 2003. ACM.

[2] Ludmila Cherkasova, Diwaker Gupta, and Amin Vahdat. Comparison of the three cpu schedulers in xen. *SIGMETRICS Perform. Eval. Rev.*, 35(2):42–51, September 2007.

[3] Jiang Dejun, Guillaume Pierre, and Chi-Hung Chi. Ec2 performance analysis for resource provisioning of service-oriented applications. In Asit Dan, Frédéric Gittler, and Farouk Toumani, editors, *Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops*, volume 6275 of *Lecture Notes in Computer Science*, pages 197–207. Springer Berlin / Heidelberg, 2010.

[4] Constantinos Evangelinos and Chris N. Hill. Cloud Computing for parallel Scientific HPC Applications: Feasibility of Running Coupled Atmosphere-Ocean Climate Models on Amazon's EC2. Cloud Computing and Its Applications, October 2008.

[5] Robert Grossman and Yunhong Gu. Data mining using high performance data clouds: experimental studies using sector and sphere. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 920–927, New York, NY, USA, 2008. ACM.

[6] Scott Hazelhurst. Scientific computing using virtual high-performance computing: a case study using the amazon elastic computing cloud. In *Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: riding the wave of technology*, SAICSIT '08, pages 94–103, New York, NY, USA, 2008. ACM.

[7] Keith R. Jackson, Lavanya Ramakrishnan, Krishna Muriki, Shane Canon, Shreyas Cholia, John Shalf, Harvey J. Wasserman, and Nicholas J. Wright. Performance analysis of high performance computing applications on the amazon web services cloud. In *Proceedings of the 2010 IEEE Second International Conference on Cloud Computing Technology and Science*, CLOUDCOM '10, pages 159–168, Washington, DC, USA, 2010. IEEE Computer Society.

[8] Karthik Kambatla, Abhinav Pathak, and Himabindu Pucha. Towards optimizing hadoop provisioning in the cloud. In *Proceedings of the 2009 conference on Hot topics in cloud computing*, HotCloud'09, Berkeley, CA, USA, 2009. USENIX Association.

[9] Amril Nazir, Hong Ong, Sidek Salleh, S.Selvi, and Rajendar K. Intelligentgrid: Rapid deployment of grid compute nodes for immediate execution of batch and parallel applications. In *ICOS*, pages 180–184. IEEE Open Systems, 2011.

[10] M. Suhail Rehman and Majd F. Sakr. Initial findings for provisioning variation in cloud computing. In *CloudCom*, pages 473–479, 2010.

[11] Jörg Schad, Jens Dittrich, and Jorge-Arnulfo Quiané-Ruiz. Runtime measurements in the cloud: observing, analyzing, and reducing variance. *Proc. VLDB Endow.*, 3(1-2):460–471, September 2010.

[12] Aparna Venkatraman, Vinay P, Beth Plale, and Shing shong Shei. Benchmarking effort of virtual machines on abstract multicore machines.