Empirical Evaluation of Conservative and Optimistic Discrete Event Execution on Cloud and VM Platforms

Srikanth B. Yoginath and Kalyan S. Perumalla
Computational Sciences and Engineering Division,
Oak Ridge National Laboratory, USA
yoginathsb@ornl.gov, perumallaks@ornl.gov

ABSTRACT

Virtual machine (VM) technologies, especially those offered via Cloud platforms, present new dimensions with respect to performance and cost in executing parallel discrete event simulation (PDES) applications. Due to the introduction of overall cost as a metric, the choice of the highest-end computing configuration is no longer the most economical one. Moreover, runtime dynamics unique to VM platforms introduce new performance characteristics, and the variety of possible VM configurations give rise to a range of choices for hosting a PDES run. Here, an empirical study of these issues is undertaken to guide an understanding of the dynamics, trends and trade-offs in executing PDES on VM/Cloud platforms. Performance results and cost measures are obtained from actual execution of a range of scenarios in two PDES benchmark applications on the Amazon Cloud offerings and on a high-end VM host machine. The data reveals interesting insights into the new VM-PDES dynamics that come into play and also leads to counter-intuitive guidelines with respect to choosing the best and second-best configurations when overall cost of execution is considered. In particular, it is found that choosing the highest-end VM configuration guarantees neither the best runtime nor the least cost. Interestingly, choosing a (suitably scaled) low-end VM configuration provides the least overall cost without adversely affecting the total runtime.

Categories and Subject Descriptors

I.6.1 [**Computing Methodologies**]: Simulation and Modeling – *Discrete*; I.6.8 [**Computing Methodologies**]: Simulation and Modeling – *Types of Simulation (Discrete Event, Parallel)*

General Terms

Algorithms, Measurement, Performance, Design, Experimentation

Keywords

Virtual Machines, Cloud Computing, Parallel Discrete Event Simulation, Optimistic and Conservative Synchronization, Performance, Performance Study

1. INTRODUCTION

Historically, PDES has largely assumed the luxury of picking the highest-end among the set of computer configuration choices one could access, and proceeding to achieve the highest possible speeds on the chosen high-end system. However, with the introduction of the Cloud platforms, the new dimension of *price* is introduced into consideration. Since there is a price one must pay for all compute cycles used by the application, a "dollar value" is now attached to each PDES run. The most interesting aspect about this new dimension is that the price variation is non-linear. The user might have to pay more than double the price for double the performance. Alternatively, doubling the cost does not guarantee double the performance. In this new milieu, little is known about the price-performance features of PDES execution on Cloud platforms, and about the configuration choices of PDES over VM platforms in general.

There are several questions that arise. What, if any, is the level of performance penalty taken by a PDES application when moving from a traditional native execution to a VM? Is there any performance gain obtained by insisting that the VM be a privileged one versus the default, unprivileged mode of VMs? Does the highest-end VM/Cloud hardware configuration always deliver the least total execution time, and at what overall cost? If the highest-end VM configuration is too expensive for the user, what is the next best configuration to choose, considering overall cost? How well does a Cloud platform designed primarily for embarrassingly parallel jobs execute tightly coupled PDES applications? Are runtime and dollar value largely opposed to each other as one might expect? In general, how do the total execution time and total cost vary with different VM/Cloud instance configuration options?

Here, an empirical approach is undertaken to help answer such questions. Results and findings are reported to understand the new configuration space for PDES enabled by the introduction of new, Cloud-specific concepts such as abstracted speeds of virtual processors, normalized units of processors and memories, price per packaged compute-unit, and overall "bottom-line" cost. Using actual PDES application runs executed on a Cloud platform and on high-end VM hosts, we study the configuration space to uncover new insights, trends, and guidelines on the problem of economically executing PDES applications in Cloud environments. For experimentation purposes, we chose the popular Amazon AWS (EC2) [1] Cloud computing platform, and gathered data from a variety of configurations with varying price-performance characteristics. The empirical study makes use of two benchmarks: one is the popular synthetic PHOLD benchmark, and the other is a complex disease spread model at the individual level in a large population. Both optimistic and conservative synchronization schemes are exercised, with varying levels of locality of events.

The rest of the article is organized as follows. Section 2 briefly introduces virtual machine systems, Cloud infrastructure, and PDES execution. Section 3 introduces the PDES applications, their scenario configurations, and the hardware used for the performance study. This is followed by the performance results

SIGSIM-PADS'13, May 19–22, 2013, Montréal, Québec, Canada.

and their analyses in Section 4. The work is summarized and future work identified in Section 5.

2. BACKGROUND

2.1 Virtual Machines and Cloud Computing

Virtual Machine (VM) technology moves the traditional operating system (OS) away from the actual hardware interface and transplants it to work over a software interface. The decoupling enables entirely new modes of execution from a user's point of view, and provides many benefits such as flexibility, multiplexed use, fault tolerance, dynamic migration, automated load balancing, and cost sharing. Anyone can exploit the benefits of VM technology by deploying the VM implementations on their own hardware. Cloud computing is a term generally used to refer to such installations that provide the advantages of virtualized computing (and storage) interfaces. Due to economies of scale, only large commercial, dedicated installations provide the most cost-effective provisioning of VM technologies and make them accessible over the Internet via Web-based interfaces for very attractive prices. They provide on-demand access to compute resources without the burden of housing, installation, maintenance, and upgrading needs.

Since commercial offerings of Cloud computing are profitable mostly due to the ability to multiplex many smaller units of virtual hardware on larger units of physical hardware, one can expect a price structure that permits the most flexibility for multiplexing. In particular, the price structure favors smaller virtual units, and, more importantly for PDES, charges a non-linearly larger price for the highest-end virtual units. Thus, virtual machines whose resources (e.g., speeds and numbers of virtual processors) are close to the capacity of the underlying physical hardware can be expected to be the most expensive.

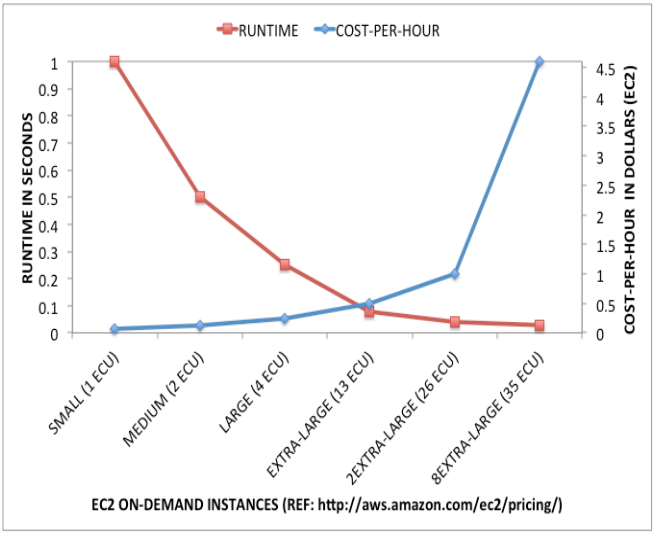

Figure 1 EC2 cost-value model

Figure 1 illustrates the cost model for the Amazon AWS-based EC2 Cloud service that is based on the allotted hardware resource sizes. Suppose the user requires the Cloud for executing a parallel job, and further suppose the user's application enjoys an ideal parallel execution by which the runtime decreases in proportion to the number of processors. The runtime for a parallel job is plotted against different computational units offered by the Cloud platform. Along the abscissa, the size of each indivisible virtual computational unit increases moving from left to right. On the left ordinate, the ideal parallel runtime is plotted. On the right ordinate, the cost for the computational unit is plotted. The non-linear aspect of the price is notable. Also, additional non-linear effects can be expected due to shared-memory effects, shared network effects, scheduling effects, and inter-VM communication effects. Thus, unless the parallel job is embarrassingly parallel in nature, it is rather difficult to predict the trends of runtime and overall cost of parallel jobs, and an empirical study is inevitable for properly understanding the overall tradeoffs.

2.2 Hypervisor

The VM technology relies on a component called the *hypervisor* that replaces the OS to be the lowest-level software running directly over the hardware. By doing so it the segregates the hardware and the OS by virtualizing the hardware components to the OS, and relieves the tight hardware-OS coupling. As an important consequence, OS instances from different vendors co-exist and run on a same physical resource, interacting with the virtual counterparts of the physical hardware. In general there are two different types of virtualization: full-virtualization, and para-virtualization. In full-virtualization, the hypervisor supports any unmodified guest OS to run. In para-virtualization, the hypervisor provides *hypercalls* to the OS for accessing hardware services; this requires the modification of a guest OS. In the empirical study here, we use the Xen® hypervisor platforms for our performance evaluation. The Amazon's EC2 Cloud infrastructure also uses the Xen hypervisor.

Xen Hypervisor

The Xen [2] hypervisor is a popular open source industry standard for virtualization, supporting a wide range of architectures including x86, x86-64, IA64, and ARM, and guest OS types including Windows®, Linux®, Solaris® and various versions of BSD OS. A VM in Xen terminology is referred to as Guest Domain or simply DOM. Each DOM has a unique identifier called its Domain ID (DOM ID). The first DOM called DOM0 is a privileged one with special management rights. System administrative tasks such as suspension, resumption, and migration of DOMs are managed via DOM0. Any other VM is called a DOMU.

Each DOM has its own set of virtual devices, including virtual multi-processors called virtual CPUs (VCPUs). The hypervisor's scheduling functionality mainly deals with the efficient mapping (multiplexing) of all the VCPUs of multiple VMs onto the available physical processor cores (PCPUs). The *credit-based scheduler* is the default Xen scheduler, which schedules VCPUs on to PCPUs based on the principle of fair sharing of computational cycles. This scheduler is very widely used, and works excellently for a very large variety of virtualization users.

2.3 Parallel Discrete Event Simulation

A discrete event simulation model of a system emulates the system's behavior over time in terms of events and state changes. An event is a discrete point on the virtual timeline that corresponds to a change in the evolution of the physical system state. A parallel execution of the discrete event model involves a coordinated evolution of several serial discrete event timelines in parallel. The modeled system state is divided into a set of encapsulated states; each encapsulated state with its own simulation timeline is generally referred to as Logical Process (LP) in the PDES literature. The timelines are synchronized according to the timestamps of the events spanning all the timelines, guaranteeing a global timestamp-ordered execution. Two distinct synchronization mechanisms namely, conservative and optimistic synchronizations are popularly used in the PDES

literature. In conservative synchronization, no timestamp order violation is allowed. In contrast, in optimistic synchronization, LPs are allowed to incur transient violations of timestamp order, which are detected at runtime and corrected on the fly using various algorithms. More details on PDES and its synchronization mechanisms can be found in [3]. In our experiments, we use a parallel/distributed simulation engine that is optimized for very fast execution on native hardware. The engine supports the notion of a Federate that hosts one or more LPs. Each Federate is essentially a UNIX process with its own simulation loop that processes the events of all LPs mapped to that Federate. There is exactly one Federate launched per virtual core in each VM.

2.4 Related Work

Evaluating the HPC applications performance on Cloud infrastructure has been reported in [4], but these applications are largely high-performance scientific applications such as Community Atmospheric Model (CAM), and are not PDES applications. Network performance on Amazon EC2 data-centers has been studied and an evaluation of the impact of virtualization on network parameters such as latency, throughput, and packet-loss was discussed in [5], again in non-PDES context. There is a good overview and discussion of generic utilization of Cloud infrastructures for PDES applications focusing on the advantages and challenges it poses [6], which also serves as a good motivation and background for PDES on Cloud platforms. The Master-Worker approach to distributed (and fault tolerant) PDES has been explored in multiple places [7][8][9], and is somewhat related, although it is different from the traditional PDES execution view in which all processors are equal. Recently, an evaluation of a set of conservative synchronization protocols on EC2 was reported [10]. Overall, the area is nascent, and much additional research in PDES execution is needed to explore the space opened by the new metrics of VM/Cloud computing beyond the raw speed execution.

3. EMPIRICAL STUDY SETUP

3.1 Synchronization Protocols

Our empirical study uses an optimized implementation of parallel discrete event simulation that provides the option of conservative as well as optimistic synchronization. The synchronization protocols are the blocking-based variant of the set of highly scalable global virtual time (GVT) algorithms recently tested at very large scale on supercomputers [11]. The algorithms are implemented using the blocking collective call `MPI_Allreduce()` inside the iterations of the GVT algorithm to account for all transient messages. Since this is a Mattern-style [12][13] epoch-based framework that colors messages and uses reduction-based counting of transient messages, no NULL messages are needed. The parallel implementation is competitive with sequential execution, with a high efficiency.

3.2 Benchmark Applications

PHOLD

The de-facto standard PHOLD [14] benchmark application is used as one of the applications for our performance evaluation. In all our experiments we utilize the following variants for performance evaluation.

- Synchronization: Type of synchronization used [OPT=optimistic, CONS= conservative]
- Number of LPs per Federate (NLP): [NLP=100 means the Federate hosts100 LPs]
- Number of messages (NMSG): [NMSG=1000 means 1000 messages/LP]
- Locality of the LP generated message destination (LOC): In terms of intra-processor event communication [LOC=50, or LOC=90]. Values of 50 and 90 suggest respectively that 50% and 90% of the messages generated by an LP are local to its Federate. Hence, a value of 50% for LOC involves more LP message exchanges across the network and results in increased network traffic.

Disease Spread Simulation

For the second application, we use an epidemiological disease spread model [16] that defines a discrete event model for the propagation of a disease in a population of individuals in groups called locations and aggregates of locations called regions. Each region is mapped to a Federate. Multiple locations are contained in each region. Each location is housed in an LP. Multiple individuals are instantiated at each location, and they not only interact with individuals within the same location but also periodically (conforming to an individual-specific time distribution function) move from one location to another. Similar to PHOLD, the number of individuals per location are varied (e.g., 1000 individuals/location), and the number of locations per region (e.g., 10 locations/region).

3.3 Benchmark Configurations

Using the PDES applications listed in the previous section, four benchmark applications, namely, conservative and optimistic executions for each of *PHOLD Simulation Benchmark* (PSB) and *Disease Spread Benchmark* (DSB) were designed. In all the benchmarks using PHOLD a *lookahead* of 1 was used.

PHOLD Scenarios (PSB)

With PHOLD, we used scenarios with 100 LPs/Federate, 100 and 1000 messages/LP, with 32 Federates for 50% and 90% LOC values. With performance data gathered for both optimistic and conservative synchronization scenarios, a total of 8 sets of readings are gathered for this benchmark.

For 100 LPs/Federate and 100 messages/LP on 32 Federates, 3200 LPs are hosted on 32 DOMs to simulate exchanges of 320,000 PHOLD messages over 100 units of simulation time. Similarly, for 100 LPs/Federate and 1000 messages/LP on 32 Federates, 3200 LPs are hosted on 32 DOMs to simulate exchanges of 3,200,000 PHOLD messages over 100 seconds of simulation time. With a locality value LOC of 50% half of the messages generated are destined to LPs on remote Federates (outside the VCPU). With a locality value LOC of 90%, only 10% are destined to LPs on remote Federates. Thus, LOC 50 is much more taxing on the network than LOC 90.

Disease Spread Benchmark (DSB)

DSB simulates the disease spread across regions. Each Federate is mapped to a region, which are formed of number of locations that are mapped to LPs. In the experiments 32 Federates and 10 locations per Federate are instantiated (representative of a small city sized scenario for disease propagation) and each such location has a population of 1000 people. Hence the benchmark involves simulation of spread of disease across 320 locations across a population of 320,000 for a simulation time of 7 days. The mobility of the population can be set to a certain percentage, similar to PHOLD. In the DSB we experiment with 50% and 90%. The LOC 50 mobility suggests that 50% of the trips tend to travel across regions, while LOC 90 suggests only 10% of the trips travel across regions. DSB is slightly I/O intensive

generating around 32M of output data compared to less than 200 KB of output data of PSB.

3.4 Test Platforms

We utilize two platforms for the VM-based experiments. One is a local high-end machine in our laboratory, and the other is a commercial Cloud offering. The details of these two platforms are provided next.

Local Test Platform (LTP)

LTP is our custom-built machine with a Supermicro® H8DG6-F motherboard supporting two 16-core (32 cores in total) AMD® Opteron 6276 processors at 2.3 GHz, sharing 256GB of memory, Intel Solid State Drive 240GB and a 6TB Seagate constellation comprising 2 SAS drives configured as RAID-0. Ubuntu-12.10 runs with Linux® 3.7.1 kernel runs as DOM0 and DOMUs, over Xen 4.2.0 hypervisor.

All DOMUs are para-virtual and networked using a software bridge in DOM-0. DOM-0 is configured to use 10GB of memory and the guest DOMs were configured to use at least 1GB memories each, which were increased as necessitated by the application benchmarks. Each guest DOM uses 2GB of LVM-based hard disk created over SAS drives, while the DOM-0 uses an entire Solid State Drive (SSD). OpenMPI-1.6.3 (built using gcc-4.7.2) was used to build the simulation engine and its applications. A machine-file listing the IP addresses of the VMs was used along with *mpirun* utility of OpenMPI to launch the MPI-based PDES applications onto VMs.

EC2 Cloud Platform

We also ran our benchmarks on Amazon's EC2 Cloud platform. We built a cluster of para-virtual VM instances of Ubuntu 12.04 LTS. The following are the VM clusters used to run the benchmarks (typical offerings available to Amazon EC2 users).

- *m1.small* is a single-core VM with compute power of 1-ECU and has a memory of 1.7 GB.
- *m1.medium* is a single-core VM with compute power of 2-ECUs and has a memory of 3.7 GB.
- *m1.large* is a 2-core VM with compute power of 4 ECUs and has a memory of 7.5 GB.
- *m1.xlarge* is a 4-core VM with compute power of 8 ECUs and has a memory of 1.5 GB.
- *m3.2xlarge* is an 8-core VM with compute power of 26 ECUs and has a memory of 30 GB.
- *hs.8xlarge* is a 16-core VM with compute power equivalent to 35 ECUs and has a memory of 117 GB.

The term ECU here refers to a "EC2 Compute Unit" which is an abstraction defined and supported by Amazon as a normalization mechanism to provide a variety of virtual computation units independent of the actual physical hardware support that they use/maintain/upgrade without user intervention. OpenMPI-1.6.3 was built on the virtual instance, which was used to build the simulation engine and all the PDES applications. A machine-file listing the DNS names of the allotted instances was used to launch the MPI-based PDES applications using *mpirun*.

4. PERFORMANCE STUDY

4.1 Results from Local Test Platform (LTP)

PDES being a parallel computing application, two important factors, namely, computation and communication, determine the overall application performance. The hypervisor essentially virtualizes the hardware resources and hence a VM running over hypervisor uses a virtual CPU (VCPU) and a virtual network interface for computation and communication, respectively. The hypervisor essentially maps the VCPUs onto the available CPUs, while networking is performed using front-end and back-end virtual interfaces. Hence, the hypervisor essentially introduces some overhead due to its presence.

Virtual Computational Performance

We know that the hypervisor is a necessity to realize Cloud computing. However, this implies that native execution of PDES is not possible in the presence of a hypervisor, which may introduce overheads such as context switching costs and system call (and hypercall) trap costs. Hence, a performance comparison between the native and VM runs is essential to determine the amount of degradation, if any, that PDES suffers simply for the fact that the execution is moved from native to VM platforms.

In Figure 2, the performance results of both PSB and DSB benchmarks runs are presented from *Native*, *DOM0* and a single *DOMU*. The *Native* readings correspond to a setup where Linux® runs directly over the hardware as usual, without the hypervisor. The *DOM0* readings correspond to a setup where the control-DOM with Linux® runs over the Xen hypervisor as the only running instance and is configured to use all 32 CPUs. A single DOMU readings correspond to a setup where a user-DOM with Linux® runs in the presence of control-DOM (dOM0); however DOM0 is not loaded with any load during performance runs. These results demonstrate how the presence of the hypervisor affects the compute-performance of a PDES application. As seen from Figure 2, somewhat surprisingly, the results from all the three setups are almost identical across all the runs, suggesting that the overhead of the Xen® hypervisor in delegating the CPU resources is almost negligible.

This data is helpful in addressing the issue of native *vs*. VM-based performance of PDES execution, and may encourage the community to move towards a Cloud environment by allaying uninformed fears of incurring a significant performance penalty.

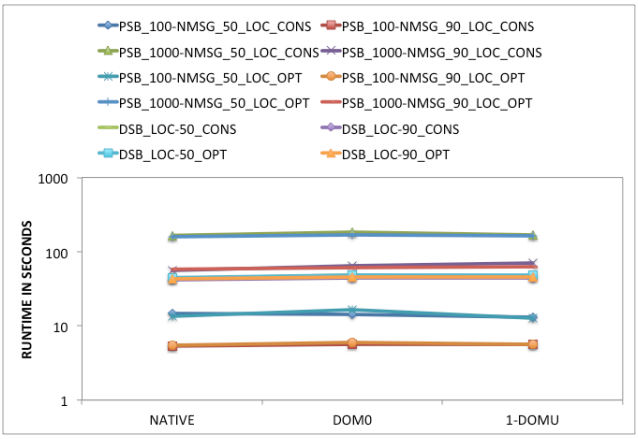

Figure 2 Native, DOM-0 and single DOMU performance comparison on LTP

Effects of Virtual Communication and I/O via DOM0

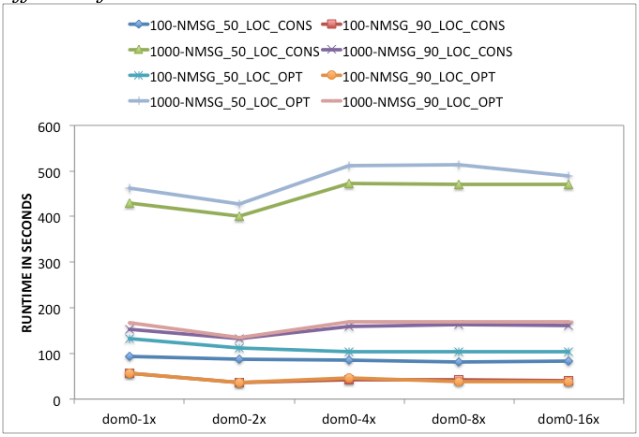

Figure 3 PSB runtime performance in 32 VM setup scenario for varying weights of DOM0 on LTP

Since DOM0 is partially involved in servicing the network communication and input/output (I/O) for all DOMU, DOM0 may need to receive sufficient number of CPU cycles (or a higher priority weight). The need for higher DOM0 weights for better performance was demonstrated in [17] using an older version (3.4.2) of the Xen distribution. To understand such requirements in the current version, the weight assigned to DOM0 relative to all DOMU is varied from 1 to 16. For example, a factor of 4 implies that DOM0 has four times more credits than any DOMU. This provides an overburdened DOM0 more CPU cycles compared to DOMUs.

The runtimes with varying weights are plotted in Figure 3 for the PSB, and in Figure 4 for the DSB. The seemingly flat curves of Figure 3 suggest very slight or no impact of higher weights for DOM0 on the runtime performance with the newer version of Xen contrary to the observation in [17]. This change can be attributed to the incorporation of Netchannel2 [18] in Xen networking, which transfers the burden of copying network data from DOM0 to the DOMUs. However, for DSB, a good improvement in the performance as the weight of DOM0 is doubled is seen in Figure 4. DSB being I/O intensive and also due to the fact that DOM0 services the I/O, the additional weight provided for DOM0 helps significantly in speeding up the I/O functionality during DSB runs.

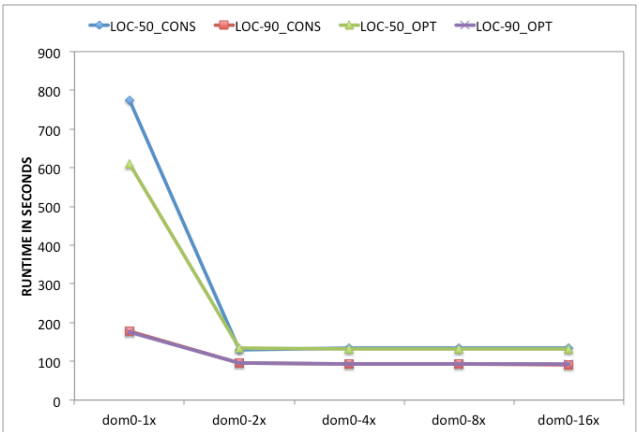

Figure 4 DSB runtime performance in 32 VM setup scenario for varying weights of DOM0 on LTP

Virtual Computation and Communication Performance

To study the impact of combined virtual computation and communication effects on PDES applications, we ran the benchmarks on our LTP, varying the number of VMs in the experiments from a single DOMU with 32 VCPUs to 32 DOMUs with 1 VCPU, keeping the total number of VCPUs constant.

Figure 5 and Figure 6 show the benchmark results obtained from varying the number of DOMs hosted on LTP using PSB and DSB, respectively. For the same benchmark the figures show how the performance varies with the increase in the number of DOMs. Note that as the number of DOMs running the benchmark increases, the number of VCPUs within each DOM also decreases. Also note that in each of these benchmark runs, the number of Federates is equal to the number VCPUs, i.e., Federates have a 1:1 mapping to the VCPUs. Hence, in a fast network environment, we expect the runtime across all types of DOM configurations to be largely identical, since the same number of VCPUs are involved in the computation.

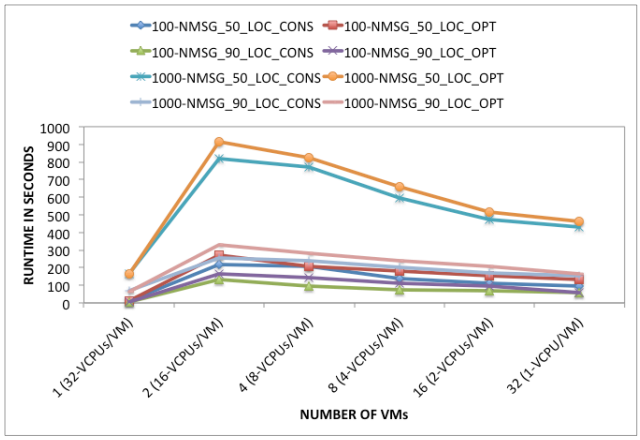

Figure 5 Performance comparison with increase in number of DOMs using PSB on LTP

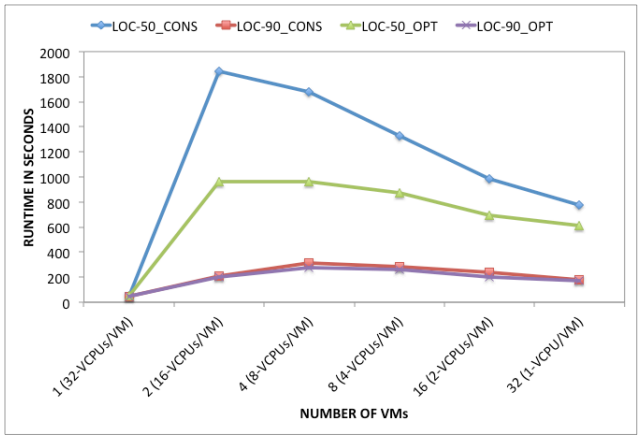

Figure 6 Performance comparison with increase in DOMs using DSB on LTP

A very interesting and common trend across all the benchmark runs is the degradation of performance with fewer numbers of VMs beyond 1, and its betterment with the increase in number of VMs hosted. In other words, there is a steep rise in runtime when moving from 1 VM to 2 VMs but a gradual drop from 2 to 32 VMs. The effect is predominant in the cases where the network

traffic is high (LOC=50), as seen in both Figure 5 and Figure 6, for PSB and DSB, respectively.

This trend is counterintuitive to a general parallel-computing user because a better performance is expected in scenarios involving VMs with more VCPUs. The intuition is that the parallel processing libraries like MPI generally use shared-memory to communicate across processes within the same VM. Hence, one expects to observe better performance by increasing the communication across Federates within a DOM (shared memory) and reducing the inter-DOM messages by reducing number of DOMs.

The underlying reason for the counterintuitive trend is as follows. When a VM contains many VCPUs, a bottleneck is created at the virtual network interface card (NIC) because of serialization. Further, DOMUs doing most of the networking work as observed in the Figure 3 in previous section adds on to this performance degradation.

In real hardware, the communication is often highly optimized via direct memory accesses, cache coherence mechanisms between NIC and CPU, and so on. However, in the case of VMs, the NIC is a software implementation, and all synchronization is performed in software, which significantly reduces the network speed. This degradation increases in a quadratic nature with the number of VCPUs sharing the virtual NIC. Unfortunately, there is little that can be done regarding this issue other than reduce the network traffic generated per VCPU or reduce the number of VCPUs per VM.

As observed earlier in Figure 1, the Cloud operators charge a lower price for low-end machines and higher cost for high-end machines. The benchmark performance results suggest that PDES applications can in fact take very good advantage of the lower cost of smaller sized VMs *and* gain lower execution time simply by moving to the other extreme of 1 VCPU/VM, and greatly benefit from the existing cost model offered by the Cloud infrastructures, like EC2.

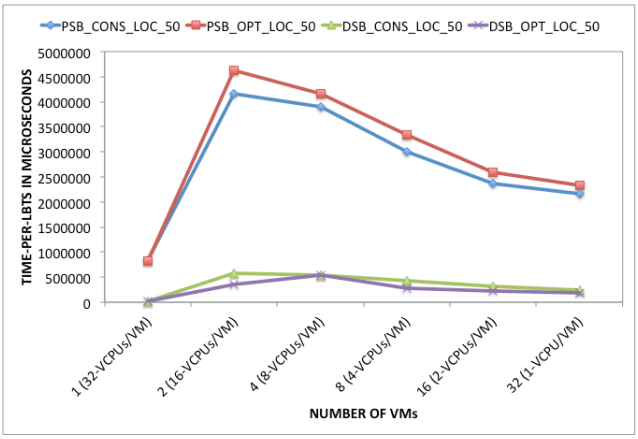

Figure 7 Time taken per LBTS computation with increase in the number of VMs on LTP

In Figure 7, we show the time that the simulator takes to compute a lower bound on incoming timestamps (LBTS) for the most-affected PSB runs namely, 1000-NMSG_50_LOC_CONS and 1000-NMSG_50_LOC_OPT runs along with LOC-50_CONS and LOC_50_OPT DSB runs. However, the number of LBTS computations remained the same across the same benchmark runs even while the number of VMs is changed. Hence, within PDES, it is the prolonged LBTS computation that affects the overall runtime of the simulation application.

4.2 Results from EC2 Cloud

To deal with possible variance of performance in the Cloud due to periodicity of loads and other uncontrollable phenomena, each data point in the results is derived as an average from three runs executed on three different days and times. Note that each request for VMs from Cloud assigns a different set of machines and hence, the virtual cluster built for every run is different from the other. This averaging for variance applies to the Cloud performance results in Figure 8 through Figure 12.

Note also that all the machines that the EC2 provides are VMs. This provides the Cloud operator an ability to multiplex multiple VMs more numerous than the available hardware resources. However, by overloading the host machine, the compute cycles of the physical machine are shared among the VM instances. By defining the Elastic Compute Unit (ECU) that is always lesser or equal to the compute cycles offered by physical CPU-core, the Amazon EC2 is able to overload the host machine and still guarantee the provision of the assured ECU worth of computational service.

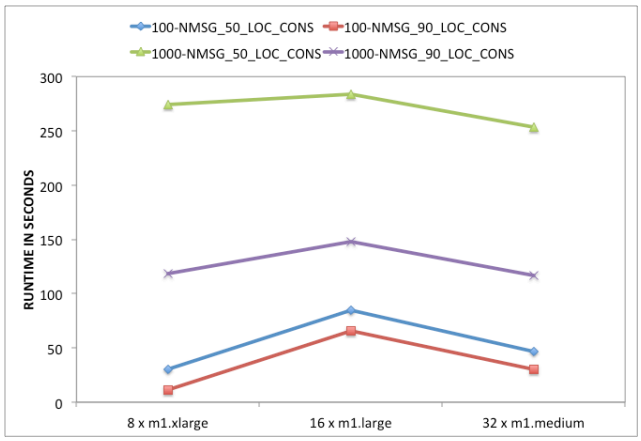

Figure 8 Runtime performance of PSB with conservative synchronization on EC2 Cloud

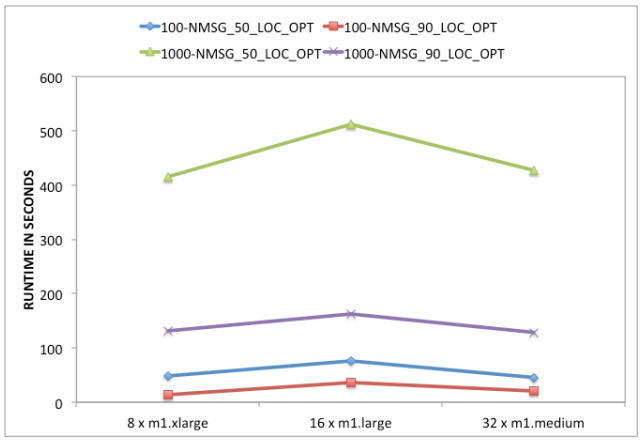

Figure 9 Runtime performance of PSB with optimistic synchronization on EC2 Cloud

With a Cloud infrastructure, the user is not guaranteed in advance specific details of the physical hardware. The user is only assured of the ECU, number of cores and amount of memory for an instance created. The performance unit of the CPU-core is provided in terms of ECUs. For example: *m1.small* and *m1.average* are both single-core VMs but with compute units of 1 ECU and 2 ECUs, respectively. The *m1.small* instance's assured

compute cycles (in ECUs) can be compared to a low priority task that can be migrated across physical nodes or multiplexed flexibly at the Cloud runtime's discretion. Hence, a good performance from *m1.small* instances is not guaranteed. Since, we would not be able to characteristically determine *a priori* the hardware-specific details of a physical CPU-core from the Cloud, the next best option is to choose the VM configuration with a single core with high number of ECUs and use it as a baseline for selecting multi-core machines, if needed.

To observe the performance trend observed with PSB and DSB benchmarks on our LTP we use *m1* set of machines comprising *m1.small*, *m1.medium*, *m1.large* and *m1.xlarge*. Of this set the *m1.medium*, *m1.large* and *m1.xlarge* VMs are *single*, *dual* and *quad* support VCPUs, respectively. Further, the compute cycles of these increase by a factor of 2; i.e., 2 ECUs, 4 ECUs and 8 ECUs in the same specified order. The *m1.small* that provides 1ECU worth of compute cycles is not considered for this set of runs because it is difficult to use it for a fair comparison with other configurations. We built three virtual clusters using these VM instances. *32×m1.medium*, *16×m1.large* and *8×m1.xlarge* are the 3 virtual clusters built using 32, 16 and 8 instances of *m1.medium*, *m1.large* and *m1.xlarge* VMs, respectively.

The conservative and optimistic synchronization-based PSB runtimes from EC2 runs are plotted in Figure 8 and Figure 9, respectively. Figure 10 presents the results for DSB runs on EC2. Interestingly, similar to LTP results, we observe a consistent trend across all the plots. The runtime in most of the cases is at its best with *8×m1.xlarge* virtual cluster, which worsens with *16×m1.large* virtual cluster and gets better with *32×m1.medium* virtual cluster runs. Note that in each of these benchmark runs the number of Federates is equal to number VCPUs.

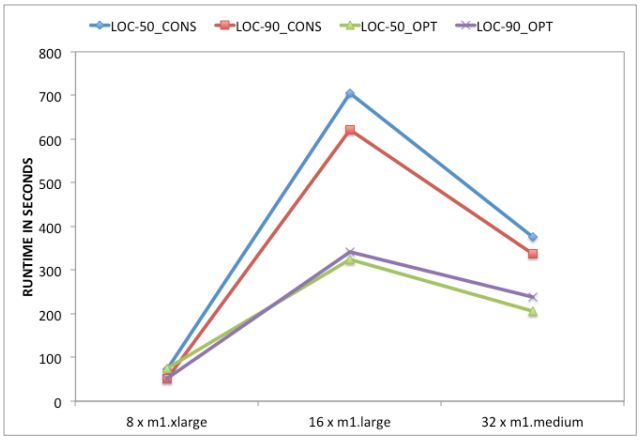

Figure 10 Runtime performance of DSB on EC2 Cloud

VM *m1.xlarge* is the most powerful among all the other offered VMs in the *m1* set. The observed counterintuitive behavior on the Cloud can be reasoned using our prior understanding of the benchmark behavior on LTP, with *m1.xlarge* considered as the physical node capacity on which the all VMs belonging to *m1* can run. Given, the lack of *a priori* guarantees about the physical hardware properties of machinery that hosts EC2 VMs, this is a fair assumption for all *m1* set of VMs. In this case, the runtime reduction in the *8×m1.xlarge* virtual cluster can be attributed to the very low involvement of the virtual network as the quad-core VMs occupying entire physical node mainly use the high-speed physical inter-connect during parallel computation. The increase in runtime in *16×m1.large* setup (instead of decrease as observed in LTP runs) can be attributed to the presence and active utilization virtual-networking as more than one *m1.large* could have been hosted on a physical node. The reduction in the runtime with increase in number of VMs in the *32×m1.medium* setup is consistent with our observations on our LTP.

4.3 LTP and EC2 Results Comparison

In comparing the results from LTP and EC2, note that all VMs running on LTP use the virtual network, whereas an indeterminate combination of real and virtual-network is typical of EC2 environment. While the LTP uses 32 CPU-cores of AMD Opteron at 2.3 GHz, the *m1* set of EC2 is perceived to use Intel Xeon CPU-cores at 2.6 GHz (*cpuinfo* of Cloud instance).

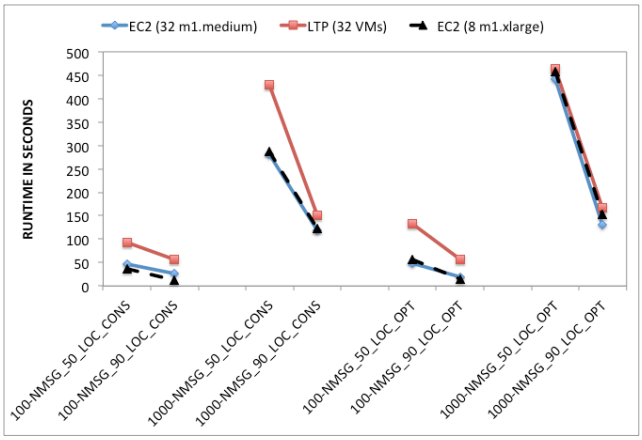

Figure 11 LTP and EC2 Cloud runtime comparison for PSB

The PSB's LTP and EC2 runtime comparisons using 32 VM scenarios are shown in Figure 11; similar runtime comparisons for DSB are shown in Figure 12. An important aspect of PSB plots in Figure 11 is the close similarities for the LTP and EC2 trends.

However, the DSB runtime plot comparisons in Figure 12 differ from this view, especially in 32 VM runs. Note that LTP runs are highly affected by network load as suggested by huge drop in runtimes as LOC value changes from 50 to 90. However, the corresponding 32 *m1.medium* EC2 runs seem unaffected, essentially suggesting the absence or minimal utility of virtual network. Further, 32 *m1.medium* EC2 runtime during LOC 90 is greater than its LTP peer, suggesting EC2 performance being effected by distribute I/O. The 8 *m1.xlarge* EC2 runtimes provide the best runtimes on EC2, suggesting that distribute I/O affects VM dispersed across many nodes more than on fewer nodes. This observation seems to be consistent with LTP, where runtimes are more affected by network performance than I/O.

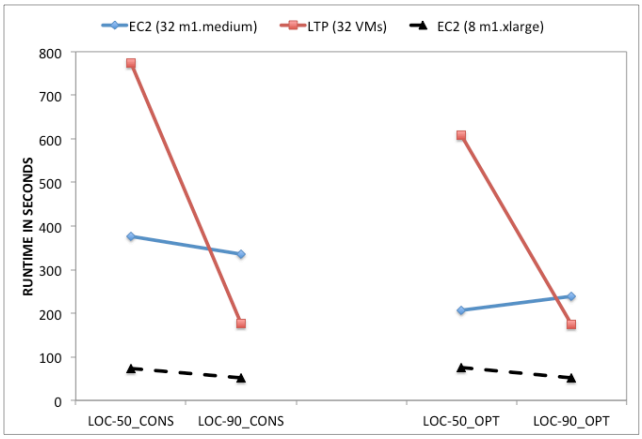

Figure 12 LTP and EC2 Cloud runtime comparison for DSB

Finally, we note that these observations can be helpful in determining an upper bound runtime on the Cloud environments utilizing processors of similar clock speeds.

4.4 Cost-Value Evaluation on EC2

In Table 1, the relevant details of on-demand VMs provided by the Amazon EC2 service are tabulated. For cost-value evaluation we selected set of VMs based on the specified ECU value. To run the PSB and DSB, we built 4 clusters of VM instances. The cheapest VM instance for the least compute unit of 1-ECU is *m1.small* and a VM cluster formed using 32 such instances is called *32×m1.small*. Similarly, VM clusters of *16×m1.medium, 8×m1.large and 2×m3.2xlarge,* are formed using 16, 8 and 2 instances of *m1.medium, m1.large, m3.2xlarge* VMs, each of these VMs have an ECU of 2, 4 and 26, respectively. The *hs1.8xlarge* is the most powerful and most expensive VM that EC2 offers and, it assures an ECU of 35.

Table 1 Details of EC2 on-demand instances

EC2 Instances	Cost/hour in Dollars	Number of VCPUs	Assured Performance in ECUs
m1.small	0.06	1	1
m1.medium	0.12	1	2
m1.large	0.24	2	4
m1.xlarge	0.48	4	8
m3.2xlarge	1.00	8	26
hs.8xlarge	4.60	16	35

PSB and DSB Runtime Performance

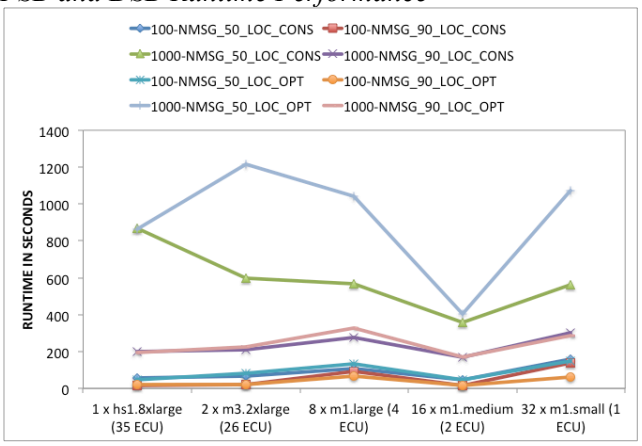

Figure 13 PSB runtime performance on EC2 Cloud

Figure 13 plots the runtimes of various PSB scenarios. While the runs with lower network traffic are almost flat, the optimistic and conservative curves vary significantly across different VM clusters. Three significant observations can be made from the 1000_NMSG-50_LOC_CONS and 1000_NMSG-50_LOC_OPT runtime plots. They are (a) contrary to the trend observed in the LTP runs the runtime of both OPT and CONS curves worsen on *32×m1.small*, (b) the runtime on high-end *hs1.8xlarge* VM, where the Federates are hosted on a single node and in the absence of network utilization during parallel computing, the runtime is the worst among all (c) the best performance across almost all runs is obtained with *16×m1.medium* cluster setup, where 32 Federates are hosted on 16 instances of single-core VMs.

A VM with ECU 1 on a hypervisor running on CPU-cores whose compute capacity is often multiple of ECUs, can be realized either in scenarios where *m1.small* VMs are overloaded on the hypervisor or on nodes where it's often run as VM with lower weight and are generally capped so that they do not exceed their provision. Either of these cases is detrimental for highly asynchronous parallel computing PDES applications. Hence, the poor performance with *32×m1.small* is expected.

Regarding the poor performance on *hs1.8xlarge* runs, note that the *hs1.8xlarge* is a 16-core VM and is loaded with 32 Federates. In overloaded scenarios such as these the hypervisor VCPU scheduler in quest of ensuring fairness in physical CPU utilization among all VCPUs affects the performance. This is a known problem [17].

Further, good runtime performance can be expected from *16×m1.medium* virtual cluster runs based on our previous observations both on LTP and EC2.

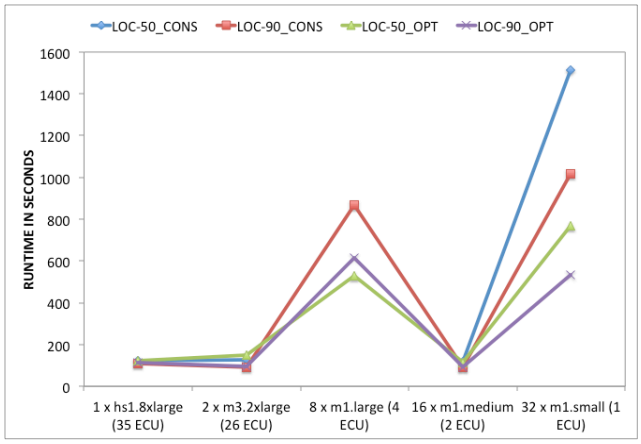

Figure 14 DSB runtime performance on EC2 Cloud

Figure 14 plots the runtimes of various DSB scenarios. Here, the runtimes are almost same on *hs1.8xlarge*, *2×m3.2xlarge* and *16×m1.medium* virtual clusters. Same reason as stated for PSB explains the bad performance of DSB on *32×m1.small.* The readings for *8×m1.large* are consistent with the observations seen with *16×m1.large* virtual cluster runs shown in Figure 10 and the performance degradation can be attributed to virtual-networking.

Cost factoring to PSB and DSB Scenarios

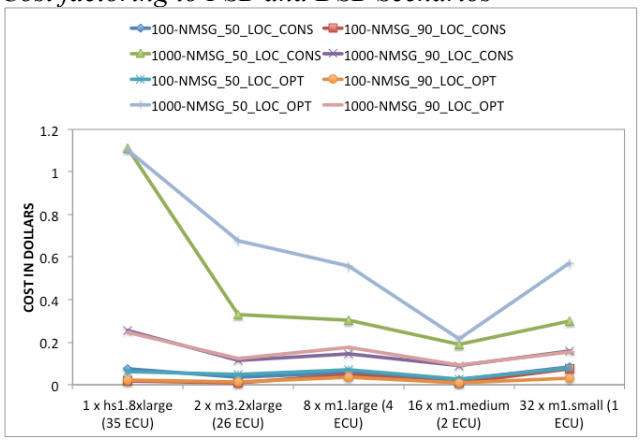

Figure 15 Overall cost of PSB on EC2 Cloud

After obtaining the runtime from the PSB and DSB runs and the cost-per-hour from the EC2 specifications, we computed the overall cost for the PSB and DSB scenarios. Figure 15 and Figure 16 plot the cost of execution in terms of dollars on various virtual clusters. For most of the runs it was found that the *16×m1.medium* virtual cluster provided the best cost-value across almost all runs for both PSB and DSB scenarios.

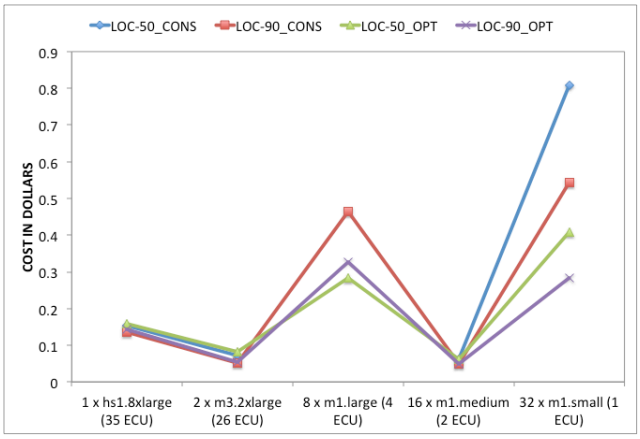

Figure 16 Overall cost for DSB on EC2 Cloud

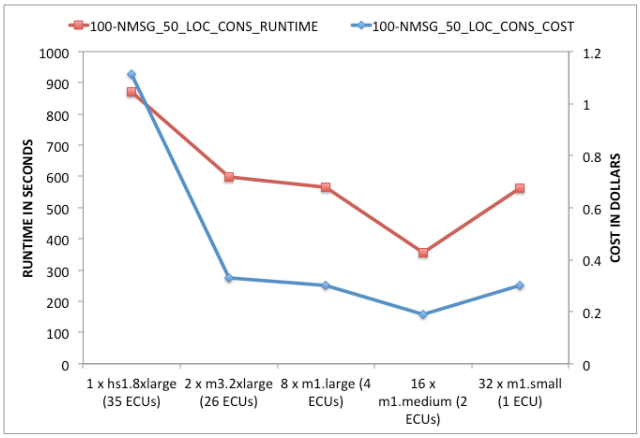

Figure 17 Cost and runtime plots of PSB 1000-NMSG 50-LOC scenario with conservative synchronization on EC2 Cloud

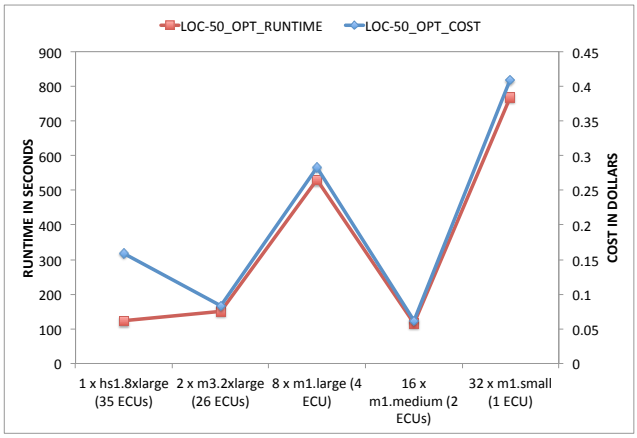

Figure 18 Cost and Runtime plots of DSB with LOC-50 using optimistic synchronization on EC2 Cloud

To compare the runtime and costs, we pick the better-performing large-scale scenarios with high-network traffic from PSB (100-NLP_1000-NMSG_50-LOC_CONS) scenario and DSB (LOC-50_OPT) scenario, as shown in Figure 17 and Figure 18, respectively. The PSB plot in Figure 17 shows best runtime and best cost associated with virtual cluster of decently compute-intensive instances of VM, i.e. *16×m1.medium*, which is against popular belief. The cost and runtime on the expensive high-end resource is far higher than that on the *16×m1.medium* cluster.

Similar to PSB, the DSB plot shown in Figure 18 also provides better runtime performance and cost at *16×m1.medium*. Even though the runtime provided by the expensive high-end compute resource compares well with runtime, the cost of computation is higher than that of *16×m1.medium*.

4.5 Performance Summary

From the benchmarks and scenarios, we find that VM-based execution can be as fast as native execution, with little perceivable performance degradation. Also, privileged and unprivileged VMs deliver the same runtime, indicating that it is not worthwhile to elevate privileges with the goal of increasing performance for PDES runs.

On dedicated machines in which the number of virtual cores is exactly the same as the number of physical cores, the fastest execution is obtained by using only a single VM that contains all the virtual cores. However, such a dedicated allocation of virtual to real cores is almost impossible to ensure in a typical Cloud environment because the underlying physical machine is opaque and also subject to change. Thus, the fastest execution that is competitive with native execution cannot be obtained on the Cloud. In fact, due to complex scheduler artifacts that arise due to a fundamental mismatch between virtual time order and fair scheduling order, the PDES execution on the highest end VM configuration in the Cloud suffers from degraded performance. To make matters worse, since the computational cycles on the highest-end configuration also cost significantly more than other lower end configurations, the overall cost can be much higher, hence less competitive, than execution on lower end configurations. Thus, on the Cloud, it seems to be more economical to choose some of the least expensive configurations (which have only one or two virtual cores per VM), which deliver a dramatic reduction in cost coupled with good runtime relative to the high-end configurations.

On dedicated VM hosts outside the Cloud, there is also an interesting tendency towards the extremes: while the best runtime is obtained on one VM with all the virtual cores, the next best is obtained on the other extreme of the spectrum in which each VM has only one virtual core. In other words, to obtain the best performance, either 1xN or Nx1 should be chosen (N is the number of physical cores), but all other configurations in between should be avoided as they suffer from worse performance.

Recommendations to Cloud users

We find that an analogy to automobiles is appropriate here. An automobile engine may be designed for speed (e.g., 0-60mph time) and/or gas mileage (miles per gallon, mpg). While racecars may exclusively focus on speed, consumer market automobiles have to find a good tradeoff between speed and mileage. Similarly, while PDES has largely focused so far on speed, it may be time to visit the counter part of miles per gallon, namely, the dollar value, when the PDES applications are executed in a Cloud environment.

Taking the preceding observations into account, it is clear that low cost and small runtime are not always opposed to each other, and that trade-offs exist.

- On a node with *N* physical processor cores, the overheads of the virtual network interface should be avoided either by using the entire physical node with a single-VM using *N* VCPUs or or by using *N* VMs each with only one VCPU.
- When a PDES simulation is executed using a single VM (that uses the entire physical node), the host node must avoid not be overloaded with more VCPUs than physical cores; i.e., the number of federates per VM should equal the number of VCPUs (not the number of ECUs). This avoids the undesired effects of VM scheduling on PDES performance.

5. CONCLUSIONS AND FUTURE WORK

Virtual Machine-based systems form an appealing computational platform that is a strong alternative to high-end native execution that has been the traditional focus of PDES. However, due to the introduction of new dimensions such as configuration options and corresponding price points, a PDES user is faced with a new problem of having to decide on the most cost-effective configuration. Overall, the dollar-value of PDES runs has now become an important metric. In this context, relatively few studies exist as guidance to help contemplate, understand and decide on the various factors and effects for effective use of VM-based platforms such as commercial Cloud offerings. To help bridge this gap, an empirical study was conducted on a wide range of VM configurations, PDES applications, and scenarios. The results helped uncover counter-intuitive effects, the significant among which being that it may be not only economical but also faster to split the simulation into fine units; dividing the simulation into many VMs may in fact provide a better overall dollar-value than using the highest-end or second-to-highest VM configurations. Also, PDES runs may benefit from the inclusion of a network metric in the specification of the abstract computational unit, the absence of which seems to leave the computation to be highly sensitive to the vagaries of virtual network devices. The study and results presented here are among the first to evaluate the characteristics of PDES in detail. The results are also timely due to the great appeal of commercial Cloud offerings that many find to be very user-friendly and convenient to access and manage. Additional work is needed to evaluate even larger VM configurations, and more PDES applications with a wider variety of event granularity and event loads. Also, Cloud-specific synchronization algorithms may also be needed to be resilient to variations in virtual network latencies.

ACKNOWLEDGMENTS

This research was performed as part of a project sponsored by the U.S. Army Research Laboratory. This paper has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the U.S. Department of Energy. Accordingly, the United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

REFERENCES

[1] *Amazon Elastic Compute Cloud User Guide* http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/Welcome.html

[2] D. Chisnall, "The Definitive Guide to the Xen Hypervisor," ISBN 978-013-234971-0, Prentice Hall, 2008

[3] R. M. Fujimoto, "Parallel and Distributed Simulation Systems," Wiley Interscience, 2000

[4] K. R. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H. J. Wasserman, and N. J. Wright, "Performance Analysis of High Performance Computing Applications on the Amazon Web Services Cloud," *IEEE 2nd International Conference on Cloud Computing Technology and Science (CloudCom),* pages 159-168, December 2010

[5] G. Wang and T.S.E. Ng, "The Impact of Virtualization on Network Performance of Amazon EC2 Data Center," *IEEE INFOCOM*, pages 1-9, March 2010

[6] G. D'Angelo, "Parallel and distributed simulation from many cores to the public cloud," *High Performance Computing and Simulation (HPCS),* pages14-23, July 2011

[7] R. M. Fujimoto, A. W. Malik, and A. J. Park, "Parallel and Distributed Simulation in the Cloud," *SCS Modeling and Simulation Magazine*, Vol. 1, No. 3, July 2010

[8] A. Malik, A. Park, and R. M. Fujimoto, "Optimistic Synchronization of Parallel Simulations in Cloud Computing Environments," *IEEE International Conference on Cloud Computing,* pages 49-56, September 2009

[9] A. J. Park, "Master/Worker Parallel Discrete Event Simulation" in College of Computing. PhD thesis, *Georgia Institute of Technology*, 2008

[10] K. Vanmechelen, S. De Munck, and J. Broeckhove, "Conservative Distributed Discrete Event Simulation on Amazon EC2," *12th International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pages 853-860, May 2012

[11] K. S. Perumalla, A. J. Park, and V. Tipparaju, "GVT Algorithms and Discrete Event Dynamics on 129K+ Processor Cores," *International Conference on High Performance Computing (HiPC)*, December 2011

[12] A. O. Holder, and C. D. Carothers, "Analysis of Time Warp on a 32,768 Processor IBM Blue Gene/L Supercomputer," *European Modeling and Simulation Symposium (EMSS)*, 2008

[13] F. Mattern, "Efficient Algorithms for Distributed Snapshots and Global Virtual Time Approximation," *Journal of Parallel and Distributed Computing*, Vol. 18, No. 4, 1993.

[14] R. M. Fujimoto, "Performance of Time Warp Under Synthetic Workloads," *Distributed Simulation Conference*, pages 23-28, 1990

[15] K. S. Perumalla, "μsik - A Micro-Kernel for Parallel and Distributed Simulation Systems," *Workshop on Principles of Advanced and Distributed Simulation (PADS)*, 2005

[16] K. S. Perumalla, and S. K. Seal, "Discrete event modeling and Massively Parallel Execution of Epidemic Outbreak Phenomena," *Simulation*, Vol. 88, No. 7, pages 768-783, 2012

[17] S. B. Yoginath, and K. S. Perumalla, "Optimized Hypervisor Scheduler for Parallel Discrete Event Simulations on Virtual Machine Platforms," *International ICST Conference on Simulation Tools and Techniques (SIMUTools)*, 2013

[18] S. J. Renato, G. Janakiraman, Y. Turner, and I. Pratt. "Netchannel 2: Optimizing network performance." *Proceedings of the XenSource/Citrix Xen Summit,* 2007.