International Environmental Modeling and Software Society (iEMSs)
7th Intl. Congress on Env. Modeling and Software, San Diego, CA, USA,
Daniel P. Ames, Nigel W.T. Quinn and Andrea E. Rizzoli (Eds.)
http://www.iemss.org/society/index.php/iemss-2014-proceedings

The Virtual Machine (VM) Scaler: An Infrastructure Manager Supporting Environmental Modeling on IaaS Clouds

Wes J. Lloyd[ab], Olaf David[ab], Mazdak Arabi[b], James C. Ascough II[c], Timothy R. Green[c], Jack R. Carlson[b], Ken W. Rojas[d]

[a] Colorado State University, Dept. of Computer Science
Fort Collins, Colorado 80523 USA
wlloyd@acm.org
[b] Colorado State University, Dept. of Civil and Environmental Engineering
Fort Collins, Colorado 80523 USA
[c] USDA-ARS-NPA, Agricultural Systems Research Unit
2150 Centre Ave., Bldg. D, Suite 200, Fort Collins, Colorado 80526 USA
[d] USDA-NRCS Information Technology Center
2150 Centre Ave., Building A, Fort Collins, Colorado 80526 USA

Abstract: Infrastructure-as-a-service (IaaS) clouds provide a new medium for deployment of environmental modeling applications. Harnessing advancements in virtualization, IaaS clouds can provide dynamic scalable infrastructure to better support scientific modeling computational demands. Providing scientific modeling "as-a-service" requires dynamic scaling of server infrastructure to adapt to changing user workloads. This paper presents the Virtual Machine (VM) Scaler, an autonomic resource manager for IaaS Clouds. We have developed VM-Scaler, a REST/JSON-based web services application which supports infrastructure provisioning and management to support scientific modeling for the Cloud Services Innovation Platform (CSIP) [Lloyd et al. 2012]. VM-Scaler harnesses the Amazon Elastic Compute Cloud (EC2) application programming interface to support model-service scalability, cloud management, and infrastructure configuration for supporting modeling workloads. VM-Scaler provides "cloud control" while abstracting the underlying IaaS cloud from the end user. VM-Scaler is extensible to support any EC2 compatible cloud and currently supports the Amazon public cloud and Eucalyptus private clouds versions 3.1 and 3.3. VM-Scaler provides a platform to improve scientific model deployment by supporting experimentation with: hot spot detection schemes, VM management and placement approaches, and model job scheduling/proxy services.

Keywords: Environmental Modeling; Scientific computing; Cloud computing; IaaS; Virtualization; Resource Management and Performance;

1. INTRODUCTION

The advent of modern multi-core CPUs, allow today's compute servers to process many tasks in parallel. These multi-core processors can provide increased performance for environmental modeling when: (1) model source code is architected to perform parallel computations to execute using multiple cores, or (2) multiple distinct related or unrelated model runs can be computed in parallel. Service based computing involves hosting a computational engine to perform complex domain-specific calculations sometimes referred to as business logic or middleware. Services operate using web based TCP ports and are therefore referred to as ***web services***. Deployment of environmental models as web services involves migration of the model computation from the user's client computers to run in a centralized modern datacenter to reap the benefits of faster hardware and server scalability. Users submit model service requests by describing the model parameterization including required inputs for the model run. Many service requests can be processed in parallel. Updating model code is made easier with a centralized deployment model. We refer to deployment of scientific models as web services as ***modeling-as-a-service***.

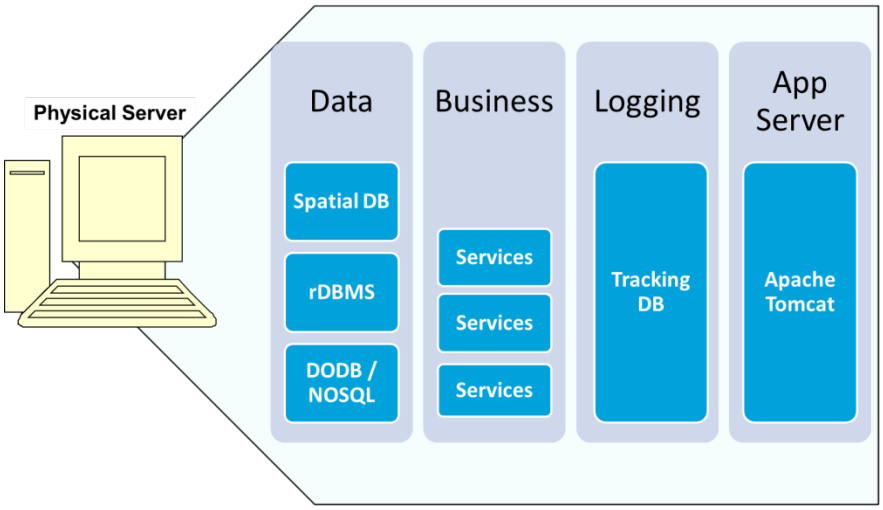

Figure 1. Traditional Service Oriented Application Deployment

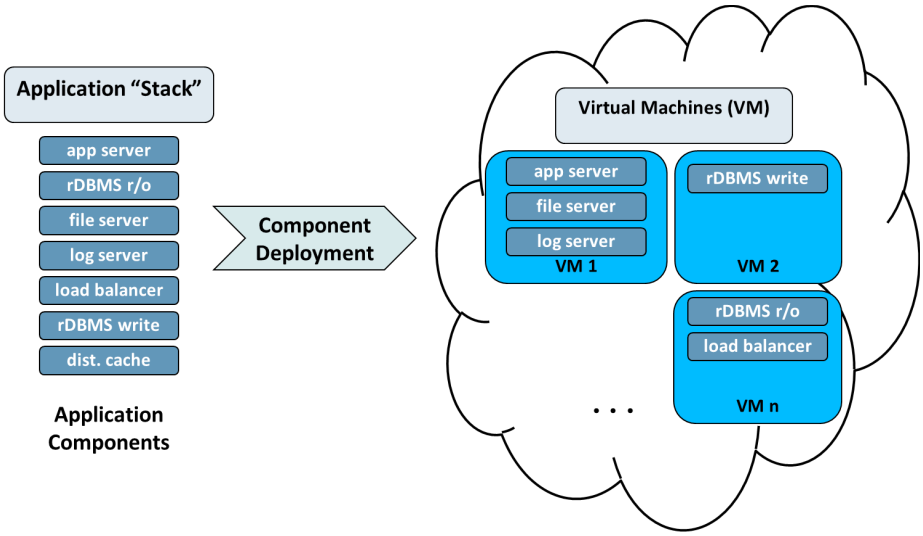

Figure 2. IaaS Cloud Service Oriented Application Deployment

Infrastructure-as-a-Service (IaaS) is a type of web service. Specifically, IaaS provides compute infrastructure, on demand, as a service, to end users. IaaS is one of the fundamental service types enabled by ***cloud computing***. IaaS clouds allow ***service oriented applications*** (SOAs) to have elastic infrastructure where resource allocations can be scaled up or down in real time to meet application demand. IaaS clouds provide ideal server infrastructure for providing modeling-as-a-service. One key challenge of providing scientific modeling-as-a-service on demand to end users is the requirement to provide both good modeling ***performance*** and service ***availability***. Availability is the notion that the modeling service is always available to perspective users. If a large spike in demand for a scientific model occurs, the modeling service should not reject new requests, but continue to accept and process them in a timely manner.

As the number of CPU cores has increased in modern servers, the overall idle time has increased. Today dual and quad processor servers can support 40+ individual processing cores. These cores frequently employ ***hyper-threading***. Hyper-threads provide two execution threads per CPU core, each of which with its own processor architectural state. These hyper-threads, referred to as logical processors, can be individually halted, interrupted or directed to execute a specified program, independent from the other logical processor. Logical processors share execution resources, allowing one processor to borrow resources from the other if stalled waiting for I/O. The benefit seen from CPU hyper-threading depends on the application's balance of computation vs. I/O, but is often 30% or

better. Hyper-threading enables fast execution of many more model runs in parallel, enabling higher throughput than single threaded CPUs.

It has become difficult for a single operating system instance to fully utilize so many cores. To support service scalability, modern multi-core servers support server ***virtualization***. Virtualization allows a single physical server to host many ***virtual machines*** (VMs). VMs are implemented using a software program known as the ***hypervisor*** which supports sharing the physical computer's processor(s), network and disk resources. Each virtual machine has its own operating system instance providing isolation. Server virtualization provides partitioning of server resources with the overall goal of increasing utilization. Increasing server utilization supports datacenter consolidation as redundant idle servers can be removed saving physical space and electricity. Popular virtualization hypervisors include kernel-based VMs (KVM), Xen, and the VMware ESX hypervisor. [Barham 2003] [Kivity 2007] [Camargos 2008].

Historically the components of multi-tier SOAs were deployed on one or more physical servers as in figure 1. In a cloud based setting, SOAs are now deployed across a set of virtual machine images (figure 2) instead of being consolidated on a single physical server. Multiple VM instances can be provisioned for each application tier using these virtual machine images enabling the application infrastructure to scale based on service demand.

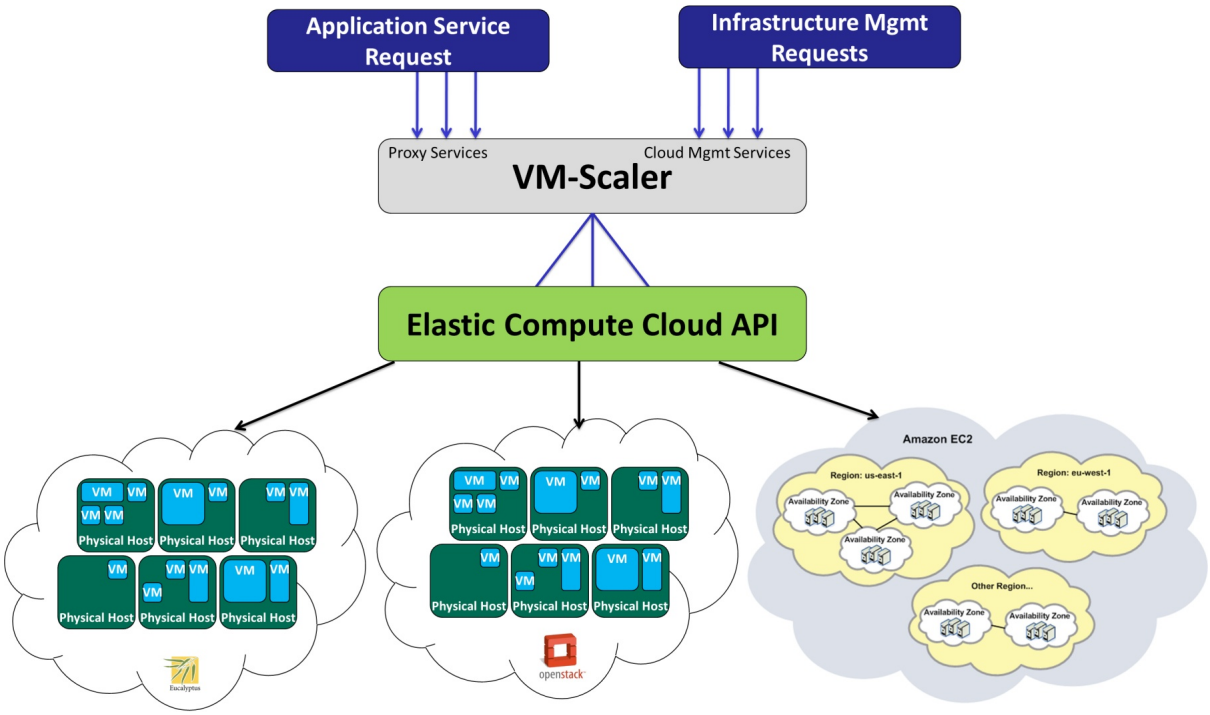

Figure 3. VM-Scaler Cloud Abstraction

Environmental models can be deployed as web services to IaaS clouds using Amazon's public IaaS cloud Elastic Compute Cloud application programming interface (***EC2-API***). The EC2-API enables programmatic management of IaaS clouds enabling dynamic management of the server infrastructure used to host model services [1]. The EC2 API is supported to varying degrees by most open source private clouds including: Apache CloudStack [Cloudstack 2013], Eucalyptus [Nurmi 2009], OpenNebula [OpenNebula 2014], and OpenStack [OpenStack 2013].

To support environmental modeling using IaaS Cloud application scaling and infrastructure management, we have developed the Virtual Machine Scaler (VM-Scaler), a REST/JSON-based web services application. VM-Scaler supports IaaS cloud infrastructure management for environment modeling as part of the US Department of Agriculture and Colorado State University's Cloud Services Innovation Platform (CSIP) [Lloyd et al. 2012]. VM-Scaler's support of model infrastructure scalability

for CSIP has been evaluated using the Revised Universal Soil Loss Equation – Version 2 (RUSLE2) [27], and the Wind Erosion Prediction System (WEPS) [28] as described in Lloyd (2014). RUSLE2 and WEPS are the US Department of Agriculture–Natural Resource Conservation Service standard models for soil erosion used by over 3,000 county level field offices across the United States. RUSLE2 and WEPS are used to provide soil erosion modeling services to end users. RUSLE2 contains both empirical and process-based science that predicts rill and interrill soil erosion by rainfall and runoff. RUSLE2 was developed primarily to guide natural resources conservation planning, inventory erosion rates, and estimate sediment delivery. The Wind Erosion Prediction System (WEPS) is a daily simulation model which outputs average soil loss and deposition values for selected areas and periods of time to predict soil erosion due to wind. WEPS consists of seven sub-models including: weather, crop growth, decomposition, hydrology, soil, erosion, and tillage.

CSIP provides a common Java-based framework for providing REST/JSON based modeling-as-a-service to end users. VM-Scaler harnesses the Amazon EC2 API to support scaling server infrastructure and management of the underlying clouds to enable modeling-as-a-service [Amazon EC2 API reference, 2014]. VM-Scaler currently supports Amazon EC2, and Eucalyptus versions 3.1 and 3.3. VM-Scaler provides cloud control while abstracting the underlying IaaS cloud and can be extended to support any EC2 compatible cloud (figure 3). VM-Scaler provides a platform for supporting scalable environmental model services and supports: (1) profiling resource requirements of modeling workloads, (2) experimentation with hot spot detection schemes, (3) investigation of VM management/placement approaches, and (4) development of custom model request scheduling/proxy services. The remainder of this paper describes features provided by VM-Scaler which help support environmental modeling services using public, private, and hybrid IaaS cloud resources.

2. THE VIRTUAL MACHINE SCALER

VM-Scaler is a REST/JSON Java-based web application installed to any web application container such as Apache Tomcat or Glassfish. VM-Scaler can be hosted by a virtual machine (VM) or physical machine (PM) having network connectivity to the managed cloud. Upon initialization VM-Scaler probes the host cloud and collects metadata including location and state information for all PMs (private clouds only) and VMs. An object model is constructed in memory to represent the state of the cloud. The Eucalyptus implementation also determines the Eucalyptus round-robin VM launch sequence to identify which node is expected to receive the next VM launch request. VM-Scaler service requests are formulated using JSON objects. Service specific JSON objects are used to describe meta-data for the requested operations. VM-Scaler is easily extensible to support new services as needed. Table 1 describes existing VM-Scaler services.

2.1. Resource Utilization Data Collection

An agent is installed to all infrastructure VMs and PMs (if accessible) to send resource utilization (RU) data to VM-Scaler at fixed intervals. The default interval is 15-seconds. The RU data collection agent is extensible and presently collects resource utilization statistics for (18) parameters as described in table 2. RU data is used to calculate resource use for: (1) the last 15-second interval, (2) the previous one minute average, and (3) a historical lifetime average. One minute averages are presently used to perform hot spot detection. (see section 2.4)

2.2 Model Workload Resource Utilization Check-pointing

VM-Scaler supports resource utilization checkpoint for environmental model workloads. Resource utilization check-pointing can be used to obtain the total resource utilization profile for a modelling workload. RU profiles can help determine the required machine resources to accomplish similar modeling workloads. RU profiles help quantify the heft or weight of modelling workloads in terms of the resource requirements needed to execute. RU profiles quantify resource usage for all eighteen resource utilization statistics described in table 2. Understanding the total CPU time, disk, and network I/O required for a batch of modelling is particularly useful if looking to schedule many similar model runs as is the case for calibration or monte carlo simulations.

Service Name	Description
CheckCurrentActivity	Reports if current modeling activity
CreatePool	Creates pool for worker VMs
CurrentPM	Reports next PM in RR launch queue
CyclePool	Resets and reconfigures worker VMs in pool
DescribePool	Describes a pool of worker VMs
DestroyPool	Destroys pool of worker VMs
GetAllPMRUData	Reports RU Data for all PMs
GetAppBusyMetricRelation	Provides BusyMetrics and performance data (CSV)
GetBusyMetrics	Report of BusyMetric values for all PMs
GetPMRUData	Report of PM resource utilization
GetPMs	Provides list of PMs
GetR2AppBehavior	Provides RU data for app being scaled (CSV)
GetRUFromCheckpoint	Provides RU data from checkpoint (CSV)
GetTestResults	Provides test results (CSV)
GetVMBetweenLaunchTimes	Provides scaling idle times (CSV)
GetVMLaunchTimes	Provides VM average launch times (CSV)
GetVMRUData	Report of VM resource utilization
LaunchVM	Launches a VM at specified location
LaunchVMSpot	Launches an Amazon spot VM instance
PostConfig	Receives service configuration object
PostTestResult	Updates test result in VM-Scaler
PostVMRUData	Receives RU data from a PM/VM
PostVMs	Post JSON object describing VMs to VM-Scaler
ScaleVM	Registers a VM to scale
SetRUCheckpoint	Creates RU checkpoint for future reporting
SkipPMs	Skips a PM in RR launch queue
StopScale	Stops scaling a VM
TerminateVM	Terminates a VM, removes associate data

Table 1. VM-Scaler Services

Statistic		Description
P/V	CPU time	CPU time in ms
P/V	cpu usr	CPU time in user mode in ms
P/V	cpu krn	CPU time in kernel mode in ms
P/V	cpu_idle	CPU idle time in ms
P/V	Contextsw	Number of context switches
P/V	cpu_io_wait	CPU time waiting for I/O to complete
P/V	cpu_sint_time	CPU time servicing soft interrupts
V	Dsr	Disk sector reads (1 sector = 512 bytes)
V	Dsreads	Number of completed disk reads
V	Drm	Number of adjacent disk reads merged
V	Readtime	Time in ms spent reading from disk
V	Dsw	Disk sector writes (1 sector = 512 bytes)
V	Dswrites	Number of completed disk writes
V	Dwm	Number of adjacent disk writes merged
V	Writetime	Time in ms spent writing to disk
P/V	Nbr	Network bytes sent
P/V	Nbs	Network bytes received
P/V	Loadavg	Avg # of running processes in last 60 sec

Table 2. Resource Utilization Statistics

2.3 Scaling Tasks

VM-Scaler provides horizontal scaling of application infrastructure by increasing the allocated number of VMs to service a particular tier of a multi-tier SOA. When application hot spots are detected one or more VMs can be launched in parallel in response. A service request is issued to describe the requested scaling task using a JSON object. The JSON object identifies the base VM, the initial VM which provides implementation of a particular application tier. The JSON object includes a VM-type meta-data tag to identify VMs launched to support the tier. An image-id identifies which virtual machine image to launch in response to hot spots. The VM-size attribute specifies the size and type for new VMs, for example m1.xlarge, m2.4xlarge. An access key is included and also the host zone/region and the VM security group are identified.

Three additional configurable scaling parameters include: `min_time_to_scale_again`, `min_time_to_scale_after_failure`, and `max_VM_launch_time`. `Min_time_to_scale_again` provides a time buffer before scaling again, allowing time to consider the impact of recent resource additions. This parameter helps to eliminate the ping-pong effect described in [Kejariwal 2013] and is equivalent to Amazon Scaling Group cool-down periods [Auto Scaling Concepts 2013]. `Max_VM_launch_time` provides a maximum time limit before terminating launches that appear to have stalled. This supports handling launch failures by reissuing stalled launch requests. `Min_time_to_scale_after_failure` provides an alternate wait time when VM launch failures occur.

2.4 Hot Spot Detection

VM-Scaler supports both resource utilization threshold and application performance model-based hot spot detection. Threshold based scaling is triggered when resource utilization variables exceed configured thresholds. This application agnostic approach is reactive to current system conditions and supports experimentation because the hot spot detection scheme can remain constant while VM scheduling algorithms or the application being tested are changed. By default scaling thresholds can be specified for one-minute averages of: maximum CPU time, minimum CPU idle time, maximum number of context switches, and maximum load average. Application performance model hot spot detection uses trends in resource utilization to predict average model execution time. Predictions are

made for average model execution time for 1, 2, and 3 time steps in the future where a time step is 15 seconds. Scaling thresholds trigger hot spot detection when future predicted model execution time exceeds set values.

2.5 Least-Busy VM Placement

For Eucalyptus private IaaS clouds, VM-Scaler supports controlling the placement of new VM's to specific physical hosts. New VM launches can specify a specific host, use the default host provided by Eucalyptus round-robin, or harness VM-Scaler's Least-Busy VM placement algorithm. VM placement to the Least-Busy physical machine is based on using our BusyMetric which aggregates total host resource utilization to determine the best candidates for hosting new VM's by quantifying host busyness [Lloyd 2014]. The busy metric double weights CPU time for environmental modelling since most models are CPU-bound in nature. Disk sector reads/writes, network bytes received/sent and host occupancy are also included in the Busy Metric calculation. Busyness is quantified relative to the observed maximum system value for each resource utilization measure. Maximums are determined through stress testing.

For example:

$$cputime_n = \frac{cputime_{obs_{1sec}}}{cputime_{max_{1sec}}} \quad (1)$$

Our Busy-Metric is expressed as:

$$\frac{\left(2 \cdot cputime_n\right) + dsr_n + dsw_n + nbr_n + nbs_n + \left(\frac{2 \cdot Hosted_{VMs}}{PM_{cores}}\right)}{7} \quad (2)$$

Each additional VM hosted linearly increases the value of the Busy-Metric by:

$$e^{\left|\ln PM_{cores} - 1.2528\right|} \quad (3)$$

The Busy-Metric provides an approach to rank available capacity of physical host machines. Our goal has been to develop a general metric which supports new VM placements based on quantifying the total shared load of private cloud host machines. Many variations of our busy metric are possible by using unique resource variable weights based on specific resource requirements of different environmental models.

2.6 Model Request Job Scheduling

VM-Scaler supports using the Busy-Metric described in section 2.4 to perform model run scheduling/proxy services. Incoming model requests can be routed to the Least-Busy VM. This provides an alternative to both round-robin load balancing and least-connection load balancing. Round-robin load balancing is supported using the HAProxy load balancer [HAProxy 2014], by evenly distributing model requests to the pool of modelling-engine VMs. Least-connection load balancing supported by HAProxy, distributes model requests by evenly balancing the number of active concurrent sessions at each modelling engine VM at any given time. VM-Scaler's Least-busy load balancing routes incoming model requests to run on the modelling engine VM with the most available resources as quantified using the Busy-Metric. Least-Busy job scheduling is a black-box job scheduler which does consider details of the model parameterization to perform the scheduling. Future work plans to investigate the development of white-box job schedulers which harness model parameterization details of incoming model requests to predict model resource and execution time requirements before execution. Harnessing predictions should improve model execution scheduling and reduce model execution times by minimizing server idle time during model workload execution which occurs between scheduled jobs that presently goes to waste.

2.7 VM Pools

VM-Scaler supports VM pools to support recycling VMs in cases when the ***launch latency*** time is high. For environmental modelling, VM launch latency is the time required to launch and initialize a new modelling engine VM before it is ready to perform model computations. Launch latency time varies based on the type and size of the VM image, as well as the host cloud and its underlying hardware. For example, on Amazon EC2, using faster VM types such as c3.xlarge generally enables more rapid VM launch and initialization times compared to slower instance types such as m1.large. Similarly, on a private cloud, VM instance types assigned more computational and memory resources typically initialize more rapidly. Launch latency will vary by cloud and the specifics of the VM being provisioned and should be benchmarked to establish baseline time requirements for dynamic scaling.

When dynamically scaling the modelling tier of an SOA it may be necessary to rapidly increase the number of worker VMs in response model demand. To support dynamic scaling when new VMs cannot be launched fast enough, VMs can be prelaunched and reserved for later use using VM-Scaler ***VM pools***. Prelaunched VMs are referred to as spare VMs. A key cost / performance trade-off concerns identifying the number of spare VMs to allocate versus the supported magnitude of model service demand spikes. When too many spare VMs are provisioned hosting costs are high, but scalability performance is excellent. Conversely when too few spare VMs are provisioned the model service may become slow, unavailable, or **crash** in response to demand spikes.

VM pools help support the use of Amazon public cloud spot instances for environmental modeling. Amazon Spot instances are low-cost VMs which are billed at a fluctuating auction price rather than the standard going rate. Prices may be as low as 1/8 to 1/9 the cost of full dedicated instances with the caveat being these instances may terminate at any instant when the bid price is exceeded due to heavy demand. Amazon spot instances have very long launch latency times since two separate Amazon EC2 operations are required to provision a VM. An initial call places a spot market bid, and if the bid is successful a VM launch operation occurs. For CSIP, launch latency times of 4-5 minutes per VM is not unusual. With such long launch latency times dynamic scaling using amazon spot instances is generally not practical without prelaunching VMs.

VM pools also support reusing VM instances for dynamic scaling in lieu of Amazon's billing model. Amazon bills hourly for VM usage. Even if a VM is only needed for 1 minute, the user is charged for an entire hour. For dynamic scaling it is useful then to recapture idle VMs for the duration of the billing cycle in case there is a future opportunity for use.

3. SUMMARY AND CONCLUSIONS

By harnessing infrastructure-as-a-service cloud computing, server infrastructure supporting environmental model services can dynamically scale based on user demand to deliver: (1) high availability, (2) high throughput (requests/second) and (3) fast model execution times.

In this paper we have presented the VM-Scaler, a cloud agnostic autonomic resource manager which supports infrastructure management for multi-tier service oriented applications. VM-Scaler supports dynamic infrastructure scaling for both new and legacy environmental models supporting their deployment as web-based model services enabling model computation "as-a-service", on demand, for users. VM-Scaler supports model service scalability on both private and public clouds by providing key features including: resource utilization data collection, model workload resource utilization check-pointing, dynamic scaling tasks, hot spot detection, Least-Busy VM placement, Job Scheduling, and VM pools. VM-Scaler supports many features in-gratis which are typically pay-for-use infrastructure services in public clouds including resource utilization data collection and dynamic scaling. The utility of VM-Scaler has been demonstrated in support of the USDA's Cloud Services Innovation Platform (CSIP) and development remains ongoing.

4. REFERENCES

Amazon Cloudwatch Namespaces, Dimensions, and Metrics Reference, 2014, http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/CW_Support_For_AWS.html (last accessed 20.03.14)

Amazon Elastic Compute Cloud API Reference, 2014, http://docs.aws.amazon.com/AWSEC2/latest/APIReference/ (last accessed 20.03.14)

Auto Scaling Concepts – Auto Scaling, 2013, http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/AS_Concepts.html (last accessed 20.03.14)

Barham, P., et al., 2003. Xen and the art of virtualization. In: Proc. 19th ACM Symposium on Operating Systems Principles (SOSP '03), Bolton Landing, NY, USA, Oct 19-22, 2003, 14 p.

Camargos, F., Girard, G., Ligneris, B. Virtualization of Linux servers. In: Proc. 2008 Linux Symposium, Ottawa, Ontario, Canada, July 23-26, 2008, pp. 63-76.

CloudStack Admin. Guide, 2013, http://incubator.apache.org/cloudstack/docs/en-US/Apache_CloudStack/4.0.1-incubating/html/Admin_Guide, last accessed (20.03.14)

Hagen, L. A wind erosion prediction system to meet user needs. In: Journal of Soil and Water Conservation Mar/Apr 1991, vol. 46, iss. 2, pp. 105-111.

HAProxy - The Reliable, High Performance TCP/HTTP Load Balancer, http://haproxy.1wt.eu/, last accessed (20.03.14)

Kejariwal, A. Techniques for Optimizing Cloud Footprint. In: Proc. 1st IEEE Int. Conf. on Cloud Eng (IC2E 2013), Mar 25-27, 2013, pp. 258-268.

Kivity, A. et al. kvm: the Linux Virtual Machine Monitor. In: Proc. 2007 Ottawa Linux Symposium (OLS 2007), Ottawa, Canada, June 27-30, 2007, pp. 225-230.

Llorente, R et al. 2011. On the Management of Virtual Machines for Cloud Infrastructures (ch. 6), in Cloud Computing: Principles and Paradigms, J Wiley & Sons, Inc., Hoboken, NJ, USA.

Lloyd, W., David, O., Lyon, J., Rojas, K., Ascough, J., Green, T., Carlson, J. The Cloud Services Innovation Platform – Enabling Service-Based Environmental Modelling Using IaaS Cloud Computing. In: Proc. iEMSs 2012 Int. Cong on Env.Modeling and Software, Germany, July 2012, 8 p.

Lloyd, W., Pallickara, S., David, O., Arabi, M., Rojas, K. Dynamic Scaling for Service Oriented Applications: Implications of Virtual Machine Placement on IaaS Clouds. In: Proc. 2nd IEEE Int. Conf. on Cloud Engineering (IC2E 2014), Mar 10-14, 2014, pp. 271-276.

Nurmi, D. et al. The Eucalyptus open-source cloud-computing system. In: Proc. IEEE International Symposium on Cluster Computing and the Grid (CCGRID 2009), Shanghai, China, May 18-21, 8p.

OpenNebula – Flexible Enterprise Cloud Made Simple, 2014, http://opennebula.org/documentation/ (last accessed 20.03.14)

OpenStack Compute Admin. Manual-Essex (2012.1), 2013, http://docs.openstack.org/essex/openstack-compute/admin/content/index.html (last accessed 20.03.14).

U.S. Department of Agriculture - Agricultural Research Service, Revised Universal Soil Loss Equation Ver. 2 (RUSLE2), 2008, http://www.ars.usda.gov/SP2UserFiles/Place/64080510/RUSLE/RUSLE2_Science_Doc.pdf (last accessed 20.03.14).