International Environmental Modelling and Software Society (iEMSs)
2012 International Congress on Environmental Modelling and Software
Managing Resources of a Limited Planet, Sixth Biennial Meeting, Leipzig, Germany
R. Seppelt, A.A. Voinov, S. Lange, D. Bankamp (Eds.)
http://www.iemss.org/society/index.php/iemss-2012-proceedings

The Cloud Services Innovation Platform – Enabling Service-Based Environmental Modelling Using Infrastructure-as-a-Service Cloud Computing

W. Lloyd[ab], O. David[ab], J. Lyon[b], K.W. Rojas[d], J.C. Ascough II[c], T.R. Green[c,] J.R. Carlson[d],

[a] Dept.of Computer Science,[b] Dept. of Civil and Environmental Engineering, Colorado State University Fort Collins, CO 80523 USA, [c] USDA-ARS, ASRU, [d] USDA-NRCS, 2150 Centre Ave., Bldg. A, Fort Collins, CO 80526 USA

Abstract: Service oriented architectures allow modelling engines to be hosted over the Internet abstracting physical hardware configuration and software deployments from model users. Many existing environmental models are deployed as desktop applications running on user's personal computers (PCs). Migration to service-based modelling centralizes the modelling functions to service hosts on the Internet. Users no longer require high-end PCs to run models and model updates encapsulating science advances can be disseminated more rapidly by hosting the modelling functions centrally via an Internet host instead of requiring software updates to user's PCs. In this paper we present the Cloud Services Innovation Platform (CSIP), an Infrastructure-as-a-Service cloud application architecture, used to prototype development of distributed and scalable environmental modelling services. CSIP aims to provide modelling as a service to support both interactive (synchronous) and batch (asynchronous) modelling. CSIP enables cloud-based computing resources to be harnessed for both new and existing environmental models supporting the disaggregation of work into subtasks which execute in parallel using a scalable number of virtual machines. This paper presents CSIP's implementation using the RUSLE2 model as a prototype model. RUSLE2 model service benchmarks are presented to demonstrate performance gains from using cloud resources. We also provide benchmarks for virtualization overhead observed using popular virtual machine hypervisors and demonstrate how application profile characteristics significantly impact performance when virtualized.

***Keywords**:* Environmental Modelling, Infrastructure-as-a-Service, Virtualization

1. INTRODUCTION

Computer hardware continues to improve as central processing units (CPUs), disks, and memory become faster, more powerful, and less expensive than ever before. Since the early 2000s, as a result of being limited by heat dissipation issues CPU design has shifted from increasing clock frequency to adding multiple processing cores on each physical chip to gain speed enhancements. Initially CPUs had two cores, then four with today's CPUs featuring up to 10 processing cores per chip. Additionally, many servers support two or more physical CPUs. The introduction of multi-core CPUs has lead to server virtualization, which enables multiple, separate operating system instances to run on a single physical server with the goal of achieving higher overall server utilization.

To take advantage of modern CPUs, software applications must be redesigned to harness multiple CPU cores. Scientific models must be re-architected to perform parallel computation. Many scientific models, particularly discrete event simulations or models which operate on a time-step interval, are inherently

sequential by nature and difficult to compute in parallel. Dependencies among modelling steps require computations to complete before they can proceed, making disaggregation of computations into pieces difficult and even impossible for the core model.

To harness computational capacity of multi-core processors three techniques hold promise: (1) decomposition of modelling algorithms to enable computation of parts of the calculation in parallel, (2) splitting the computation into pieces based on a geospatial or time stepping and then merging results together (MapReduce), and (3) for composite models which depend on one or more (sub) models, supporting models to execute independently in parallel.

Splitting the work of a computation into pieces so that intermediate results can be computed in parallel and then merged together have been led by Google with the development of their MapReduce framework and the open source version known as Hadoop developed by Yahoo [Dean and Ghemawat 2008]. MapReduce is a framework which takes a large compute and/or data intensive workload and splits the task into sub-problems which are "mapped" to different computers. Sub-problems are computed in parallel with results merged together during the "reduce" operation. As opposed to requiring expensive supercomputers, the MapReduce framework operates using networks of cheap commodity x86-based computers making high performance computing-like capabilities available with much lower infrastructure costs. MapReduce can be harnessed for environmental modelling by splitting model computations based on geospatial regions or time steps.

In this paper, we introduce CSIP, the Cloud Services Innovation Platform which uses a modelling engine deployed using Infrastructure-as-a-Service Cloud (IaaS) based virtual machines (VMs) to enable multi-core parallel model computation. This approach provides models as services and supports executing many simultaneous model runs for existing legacy models in parallel without re-architecting model code. Further, CSIP's architecture consisting of a scalable pool of distributed worker VMs is positioned to support map-reduce style disaggregation of modelling computation. Models with independent time and spatial stepping, such as discrete event simulations, are excellent candidates to harness the distributed parallel features of CSIP. The Object Modelling System version 3 framework within CSIP provides the basis for supporting parallel distributed modelling computation for individual computational steps within a model [David 2010].

2. RELATED WORK

Prior to cloud computing, scientific computing and modelling had largely been supported by supercomputers, grids, and computer clusters. Supercomputers incorporate many thousands interconnected processors in close proximity which support massively parallel computation. Grid computers are loosely coupled, heterogeneous, and geographically dispersed computers unified together using common middleware and the internet. A computer cluster is similar to a grid computer except that computers are co-located in close proximity and interconnected with a high speed local network. Cloud computing is similar in that cloud systems consist of a large number of interconnected computers supported by middleware, but individual computers have varying physical characteristics and variable geographic proximity. Further cloud systems use VMs to partition multi-core and multi-processor servers and share disk and network resources.

High-performance computing (HPC) involves the use of supercomputers and computer clusters to support modelling and solve advanced computational problems with varying degrees of coupled sub-processes running in parallel. Several studies have investigated migration of HPC applications to cloud-based environments. Regola and Ducom [2010] compared performance of three types of VMs: Open-VZ, EC2 XEN, and KVM for running HPC/MPI applications. They

identified KVM's I/O performance was sub-optimal and only Open-VZ exhibited I/O performance comparable to physical computers. El-Khamra et al. [2010] observed a large variability in network I/O performance across EC2 nodes running HPC/MPI applications. Jackson et al. [2010] compared performance of running a number of HPC applications on 2 compute clusters, a super computer and Amazon's Elastic Compute Cloud (EC2). Jackson et al. reported that their cloud based EC2 system was several orders of magnitude slower than both cluster systems. However closer analysis of their experimental design shows this was not a fair test as their EC2 system had substantially less computational resources than the cluster systems. Their shared file system consisted of only a single VM hosting an elastic block storage (EBS) volume shared with Linux's network file system (NFS) for all VMs. This design is not scalable or capable of performance comparable to the cluster computing systems leading the authors' claims of poor HPC application performance under EC2 being questionable. Collectively these studies identified the inability to control where EC2 VMs are physically located when launched in the cloud. Often VMs were not on the same local area network resulting in a large variance for inter-VM communication performance.

Ailamaki et al. [2010] identified key scientific data management challenges for modern distributed systems including complexity of data representation and processing, and support for large volumes of collected data and meta-data. Cloud based modelling systems hosting scientific data must make design tradeoffs between consistency and availability. Data replicated across VMs to improve access latency and throughput is difficult to update. When all copies of changed data are updated, availability and consistency are sacrificed. Scientific modellers must make design decisions balancing availability and consistency with performance when developing these distributed data systems. Data which does not require replication to improve throughput can use a simple approach of sharding (splitting) data across VMs to distribute database load across separate VMs. Queries operating on multiple separate shards require special techniques to complete. Research and development of distributed relational databases is active and ongoing having been encouraged by the proliferation of cloud computing [Bernstein et al. 2011].

Ostermann et al. [2009] identified challenges of harnessing cloud computing for scientific computing including integration of cloud resources within existing environments (Grids, Clusters, Hybrid clouds), virtualization overhead, scheduler awareness of VM start-up, security, and re-architecting applications for cloud environment(s). Our work with CSIP involves re-architecting environmental models for cloud deployment, quantification of overhead resulting from the use of virtual computers, and development of data services to provision required data to support real-time modelling key challenges identified by Ailamaki and Ostermann.

3. CLOUD SERVICES INNOVATION PLATFORM

To prototype and develop the Cloud Services Innovation Platform (CSIP), two private clouds were built using Eucalyptus, an IaaS virtual infrastructure manager [Nurmi et al. 2009]. Eucalyptus is an open source framework which provides an implementation of the IaaS architecture. Eucalyptus supports two common cloud application programming interfaces (APIs) developed by Amazon, elastic compute cloud (EC2) and simple storage service (S3). EC2 is an API which enables management of virtual computing infrastructure. VMs can be launched, destroyed, modified, etc. as needed programmatically using the EC2 API. S3 is an API which supports a non-SQL, non-relational simple storage system and is is essentially a cloud-based key value datastore. Recent advances in cloud computing have encouraged the evolution of distributed database technologies as the need for these new data systems has become increasingly important [Bernstein et al. 2011; Brantner et al. 2008]. Harnessing Eucalyptus as an open source private cloud technology has enabled CSIP development off-line, free from pay-for-use services.

Eucalyptus has enabled low cost experimentation with VM image compositions, development of resource scaling approaches, performance benchmarking/testing, and security design/implementation. With a design based on the industry standard EC2 API, CSIP can be deployed and hosted publicly by a number of IaaS cloud providers.

3.1 CSIP Design

Two Eucalyptus 2.0 IaaS private clouds were built and hosted by Colorado State University's Civil Engineering department in cooperation with the US Department of Agriculture (USDA). These private clouds, unlike Amazon EC2, were installed using the engineering college's local area network and were isolated to prevent outside access. One cloud consisted of 9 SUN X6270 blade servers on the same chassis sharing a private 1 Giga-bit VLAN services. Each blade server was equipped with dual Intel Xeon X5560-quad core 2.8 GHz CPUs, 24GB ram, and two 15000rpm hard disk drives of 145GB and 465GB capacity. A second cloud was built using a variety of surplus DELL Poweredge servers and commodity PCs. Both clouds employed a single server to host cloud services including the Eucalyptus cloud-controller (CC), cluster-controller (CLC), walrus (virtual machine image) server, and storage-controller (SC). All other machines were configured as Eucalyptus node-controllers (NCs) to support hosting one or more VMs using either the XEN or KVM hypervisor [Kivity et al. 2007; Barham 2003]. A hypervisor is a virtual machine monitor which manages running multiple operating systems separately on a single physical host computer. XEN and KVM are two common open source hypervisors. XEN supports *paravirtualization* (partial-virtualization) which enables VMs to have nearly direct access to the host computer's physical disk and network devices to enable faster performance [Camargos et al. 2008; Armstrong and Djemame 2011; Barham 2003]. A disadvantage of XEN's paravirtualization is that all guest VMs must use operating systems with special modifications to run. Virtualization of Microsoft Windows using XEN paravirtualization, for example, is not supported. The kernel-based virtual machine (KVM) hypervisor supports *full virtualization* of the underlying operating system which allows VMs to run any operating system without requiring a special patched version. KVM has gained popularity with recent enhancements to Intel/AMD x86-based CPUs which provide special extensions to enhance performance and better support full virtualization of guest operating systems without modification. These extensions are required by KVM and allow device simulation overhead to be reduced to provide improved performance similar to XEN [Kivity 2007; Raj et al. 2009].

The CentOS 5.6 Linux (2.6.18-274) 64-bit was used as the host operating system for cloud nodes running the XEN hypervisor, and Ubuntu 10.10 Linux (2.6.35-22) was the host operating system for cloud nodes running the KVM hypervisor. VM guests ran Ubuntu Linux (2.6.31-22) 32 and 64-bit server 9.10. One cloud used Eucalyptus-based managed mode networking with a managed Ethernet switch providing isolating VMs on their own private VLANs. The other cloud used Eucalyptus-based managed mode networking without VLAN support due to the absence of a managed Ethernet switch.

3.2 Model Prototype and Testing

The Revised Universal Soil Loss Equation – Version 2 (RUSLE2) (2008), an erosion model, was deployed as a cloud-based web service and used as a proof of concept prototype for the development and testing of CSIP. RUSLE2 is a field to small watershed model of soil movement by sheet and rill erosion processes. It uses empirical equations to calculate detachment of soil particles by the impact of rain and the force of water as it moves over a hillslope. RUSLE2 calculates erosion on a collection of linear segments representing areas of uniform surface properties (management, soil, and geometry). These segments are connected into a flow

network and process-based equations are used to track sediment as it moves downhill through this network, as well as through each segment – deposition is handled by a system of coupled non-linear equations which are solved iteratively. Calculation is done on a daily time step using either long-term average climate data, or real or simulated data on individual storms. These calculations are broken up into several hundred functions which are dynamically scheduled on a flow control engine. RUSLE2 was primarily developed to guide conservation planning, inventory erosion rates, and estimate sediment delivery and is the USDA-NRCS agency standard model for sheet and rill erosion modelling used by over 3,000 field offices across the United States.

RUSLE2 was originally developed using Microsoft Visual C++ as a Microsoft Windows desktop application. To operate as a cloud-based web service a command line modelling engine called RomeShell was added to RUSLE2. The Object Modelling System 3.0 (OMS 3.0) framework [Ajuha 2005; David, 2010] provides middleware to facilitate interaction with the RUSLE2 modelling engine. OMS was developed by the USDA–ARS in cooperation with Colorado State University and supports component-oriented simulation model development in Java, C/C++ and FORTRAN. OMS provides numerous tools supporting data retrieval, GIS, graphical visualization, statistical analysis and model calibration. The windows emulator WINE [WineHQ 2012] is used to run RUSLE2 on the Linux platform. RUSLE2 model services have been developed as JAX-RS RESTful web services hosted by the Apache Tomcat [Apache 2012] application server. JavaScript Object Notation (JSON) is used to encode input and output data objects.

The RUSLE2 web service supports both individual model runs and ensemble runs which consist of groups of modelling requests bundled together. To invoke the web service a client sends a JSON object including parameters for management practice, slope length, steepness, latitude, and longitude. Model results are computed and returned as a JSON object. Ensemble runs are processed by dividing groups of modelling requests into individual requests which are then resent to the web service, similar to the “map” function of MapReduce [Dean and Ghemawat 2008]. A configurable number of worker threads concurrently executes individual runs of the ensemble, and upon completion results are combined (reduced) into a single JSON response object and returned. A random test generation program written in Java was used to generate ensemble tests consisting of 100 randomized model-runs. Latitude and longitude coordinates were randomly selected from a bounding box encompassing most of the U.S. state of Tennessee. Slope length, steepness, and the management practice parameters were randomly selected. Randomization of latitude and longitude for slope location resulted in various spatial query execution times due to the varying complexity of geometry present at random points. To test ensemble complexity we generated 20 ensemble test sets of 100 model runs each. We observed the characteristics of the ensemble execution speed (e.g. slow, medium, or fast) of different ensembles was preserved when repeating the tests indicating that ensemble tests exhibit a complexity or difficulty characteristic (R^2=.914, df=18, $p=5 \cdot 10^{-11}$). Before executing each ensemble test, a randomized 25 model-run ensemble test was run to warm up the system and results were discarded. The warm-up test was warranted after observing that during initialization PostgreSQL performance was consistently slower for initial spatial queries performed on start-up .

4. EVALUATION

Available versions of the XEN and KVM hypervisors were tested to determine which version(s) and configurations provided optimal performance. Virtualization tests are important to understand the implications of modelling using virtual machines and the results presented here can help guide others wishing to harness cloud-based virtualization to host computations. Ten trials of an identical 100-

model run RUSLE2 ensemble test were executed and the average ensemble execution times are shown in Table 1. Version 3.4.3 of the XEN hypervisor was the fastest among those tested resulting in approximately 49% overhead versus the physical hardware. For KVM using disk virtio drivers provided the best observed performance but overall KVM was significantly slower than XEN for the RUSLE2 model showing more than 100% overhead for all KVM configurations tested.

Table 1. Hypervisor performance testing.

Hypervisor	Average Time (sec)	Normalized Performance
Physical server	15.65	100.00%
XEN 3.1	25.39	162.24%
XEN 3.4.3	23.35	149.20%
XEN 4.0.1	26.2	167.41%
XEN 4.1.1	27.04	172.78%
XEN 3.4.3 w/ full virtualization	32.1	205.11%
KVM disk virtio	31.86	203.58%
KVM no virtio	32.39	206.96%
KVM net virtio	35.36	225.94%

VM resource utilization statistics were captured using a profiling script to capture CPU time, disk sector reads and writes (disk sector=512 bytes), and network bytes sent/received. To calculate total application resource utilization, statistics from all VMs hosting application components were added. For experimentation we used two versions of RUSLE2 which perform differently from a machine resources perspective. We identify the standard RUSLE2 model as the model-bound (m-bound) model, because model performance was largely bound by model computations. The second variant is known as the database-bound (d-bound) model, where model performance was bound by spatial database queries. For the "d-bound" model two spatial database queries were modified to perform an unnecessary join against a nested query, as opposed to a table, and this greatly slowed spatial database performance. Application profiles for the "d-bound" vs. "m-bound" models are shown in figure 1.

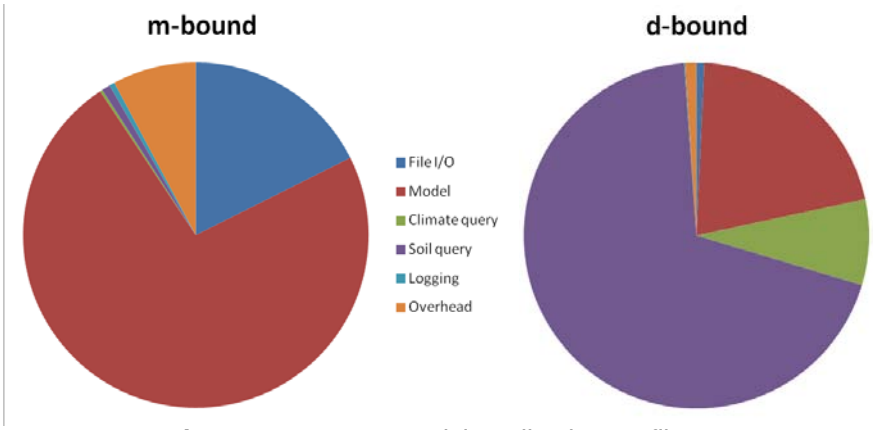

Figure 1. RUSLE2 model application profiles.

Virtualization performance of a model depends on each model's unique application profile. An application's profile consists of the CPU, disk input/output (I/O), and network I/O requirements to perform model computation. Virtualization performance of the RUSLE2 model was largely limited by the extensive quantity of

required disk I/O operations. Scientific models with less disk I/O requirements which are primarily processor intensive should have less virtualization overhead than what we observed for the RUSLE2 model. Our "d-bound" model running under KVM required 111.7% (115.98 sec) to complete a 100-model run ensemble, whereas our "m-bound" model required 189.2% (29.5 sec). The "m-bound" model, as shown in Figure 1, had more file I/O operations which explains the higher overhead.

Figure 2 shows RUSLE2 model performance when the number of model worker VMs was scaled from 1 to 16. For this test, 8 physical host computers were used and each worker VM was allocated 8 virtual CPU cores. Correspondingly we allocated 8 worker threads per worker VM in support of parallel model computations. We initially placed one worker VM on each physical host node. After scaling to 8 worker VMs (64 threads), a second worker VM was assigned to each physical host. The second set of worker VMs ran in contention with the initial set reducing performance gains. Ideally we would have had 16 physical machines to run each worker in isolation. With 16 worker VMs each worker was assigned only 6 or 7 individual model runs from the 100-run ensemble to calculate. Scaling beyond 12 worker VMs did not appear to provide performance improvements. Scaling the number of VMs did not provide linear performance gains as a point of diminishing returns was quickly reached. This result indicates the presence of scaling bottlenecks, which require application and or virtual infrastructure configuration changes to overcome (Lloyd et al. 2011). Deploying environmental models using cloud infrastructure such as Amazon EC2 enables scaling to a large number of VMs, models must be re-architected to take advantage of this capacity.

Figure 2. RUSLE2 scaled model performance.

5. CONCLUSIONS

CSIP provides application infrastructure to support migration of environmental models to operate as cloud-based model services. RUSLE2 has been deployed as a prototype model and we have benchmarked VM hypervisor performance, virtualization overhead, and computational scalability. IaaS cloud technology shows promise for improving model performance by harnessing rapid scaling of computational resources to harness capabilities of today's CPUs.

REFERENCES

Ailamaki, A., Kantere, V., Dash, D., Managing Scientific Data, Communications of the ACM, 53(6), 68-78, 2010.

Apache Tomcat – Welcome, 2012, http://tomcat.apache.org/

Armstrong, D., Djemame, K., "Performance Issues In Clouds: An Evaluation of Virtual Image Propagation and I/O Paravirtualization," The Computer Journal, June 2011, 54(6), 836-849, June 2011.

Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A., "Xen and the Art of Virtualization," Proc. 19th ACM Symposium on Operating Systems Principles (SOSP '03), Bolton Landing, NY, USA, 14 p., Oct 2003.

Bernstein, P., Cseri, I., Dani, N., Ellis, N., Kalhan, A., Kakivaya, G.,. Lomet, D., Manne, R., Novik, L., Talius, T, "Adapting Microsoft SQL Sever for Cloud Computing," Proc. IEEE Int. Conf on Data Engineering (ICDE '11), Hanover, Germany, 1255-1263, April 2011.

Brantner, M., Florescu, D., Graf, D., Kossmann, D., Kraska, T., "Building a Database on S3", Proc. ACM 2008 SIGMOD Conference (SIGMOD '08), Vancouver, B.C. Canada, 251-263, June 2008.

Camargos, F., Girard, G., Ligneris, B., "Virtualization of Linux servers," Proc. 2008 Linux Symposium, Ottawa, Ontario, Canada, 73-76, July 2008.

Chieu, T., Mohindra, A., Karve, A., Segal, A., Dynamic Scaling of Web Applications in a Virtualized Cloud Computing Environment, Proc. IEEE Conf. e-Business Engineering (ICEBE 2009), Macau, China, 281-286, October 2009.

David, O., Ascough II, J., Leavesley, G., Ahuja, L., Rethinking modeling framework design: Object Modeling System 3.0, Proc. iEMSs 2010 Intl. Congress on Environmental Modeling and Software, Ottawa, Canada, 8 p., July 2010.

Dean, J., Ghemawat, S., MapReduce: simplified data processing on large clusters, Communnications of the ACM, 51 (1), 107-113, January 2008.

El-Khamra, Y., Kim, H., Jha, S., Parashar, M., Exploring the Performance Fluctuations of HPC Workloads on Clouds, Proc. 2010 IEEE 2nd Intl. Conference on Cloud Computing Technology and Science (CLOUDCOM '10), Indianapolis, IN, USA, 383-387, Nov. 2010.

Jackson, K., Ramakrishnan, L., Muriki, K., Canon, S., Cholia, S., Shalf, J., Wasserman, H., Wright, N., Performance Analysis of High Performance Computing Applications on the Amazon Web Services Cloud, Proc. 2010 IEEE 2nd Intl. Conference on Cloud Computing Technology and Science (CLOUDCOM '10), Indianapolis, IN, USA, 159-168, Nov. 2010.

Kivity, A., Kamay, Y., Laor, D., Lublin, U., Liguori, A., "kvm: the Linux Virtual Machine Monitor," Proc. 2007 Ottawa Linux Symposium (OLS 2007), Ottawa, Canada, 225-230, June 2007.

Lloyd, W., Pallickara, S., David, O., Lyon, J., Arabi, M., Rojas, K., Migration of Multi-tier Applications to Infrastructure-as-a-Service Clouds: An Investigation Using Kernel-based Virtual Machines, Proc. of the 12th IEEE/ACM International Conference on Grid Computing. Lyon, France, Sept. 2011.

Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., Zagorodnov, D., "The Eucalyptus Open-source Cloud-computing System," Proc. IEEE Intl. Symposium on Cluster Computing and the Grid (CCGRID 2009), Shanghai, China, 8p., May 2009.

Ostermann, S., Prodan, R., Fahringer, T., Extending Grids with Cloud Resources Management for Scientific Computing, Proc. 10th Intl. IEEE/ACM Conference on Grid Computing (GRID '09), Banff, Alberta, Canada, 42-49, 2009.

Raj, H., Nathuji, R., Singh, A., England, P., "Resource Management for Isolation Enhanced Cloud Services," Proc. 2009 ACM Cloud Computing Security Workshop (CCSW '09), Chicago, IL, USA, 77-84, Nov 2009.

Regola, N., Ducom, J., Recommendations for Virtualization Technologies in High Performance Computing, Proc. 2010 IEEE 2nd Intl. Conference on Cloud Computing Technology and Science (CLOUDCOM '10), Indianapolis, IN, USA, 409-416, Nov. 2010.

RUSLE2: Revised Universal Soil Loss Equation Version 2, Science Documentation, United States Department of Agriculture - Agricultural Research Service (USDA-ARS), Washington, D.C., USA, 2008. http://www.ars.usda.gov/SP2UserFiles/Place/64080510/RUSLE/RUSLE2_Science_Doc.pdf

WineHQ – Run Windows applications on Linux, BSD, Solaris, and Mac OS X, 2012, http://www.winehq.org/