Desktop to Cloud Migration of Scientific Experiments

Satish Narayana Srirama[1], Chris Willmore[1], Vladislav Ivanistsev[2], Pelle Jakovits[1]
[1]Institute of Computer Science, University of Tartu
J. Liivi 2, Tartu, Estonia
[2]Institute of Chemistry, University of Tartu
Ravila 14a, 50411 Tartu, Estonia
{srirama, willmore, vladislav.ivanistsev, jakovits}@ut.ee

***Abstract*—Scientific computing applications usually need huge amounts of computational power. The cloud provides interesting high-performance computing solutions, with its promise of virtually infinite resources. However, migrating scientific computing problems to clouds and the re-creation of a software environment on the vendor-supplied OS and cloud instances is often a laborious task. It is also assumed that the scientist who is performing the experiments has significant knowledge of computer science, cloud computing and the migration procedure. Most often, this is not the case. Considering this obstacle, we have designed tools that help scientists to migrate their applications to the cloud. The idea is to migrate the complete desktop environment, with which the scientist is working daily, to the cloud directly. The developed desktop-to-cloud-migration (D2CM) tool supports transformation of virtual machine images, deployment description and life-cycle management for applications to be hosted on Amazon's Elastic Cloud Computing (EC2) or compatible infrastructure such as Eucalyptus. The paper also presents an electrochemical case study which extensively used the tool in drawing domain specific results. From the analysis, it was observed that D2CM tool not only helps in the migration process but also helps in optimizing the experiments, clusters and thus the costs for conducting scientific computing experiments on the cloud.**

***Keywords*-cloud computing; scientific computing; VM migration; electrochemical experiments; GPAW; supercapacitors;**

I. Introduction

Scientific computing is a field of study that applies computer science to solve typical scientific problems. Scientific computing is usually associated with large scale computer modeling and simulation and often requires large amount of computer resources. Cloud computing [1] suits well in solving these scientific computing problems, with its promise of provisioning virtually infinite resources. Cloud computing is a style of computing in which, typically, resources scalable on demand are provided "as a service (aaS)" over the Internet to users who need not have knowledge of, expertise in, or control over the cloud infrastructure that supports them. The provisioning of the cloud services can be at the Infrastructural (IaaS), Platform (PaaS) or Software (SaaS) levels.

Significant research [2], [3], [4] has been performed in migrating scientific computing applications to the cloud. Several science clouds [5], [6] have been established, several simulations and applications have been executed on these hybrid clouds and their performances measured, thus evaluating the efficiency of science on cloud. In this process we observed that the communication latencies and the other latencies added by the virtualization technology are the major hindrances for executing scientific computing applications on the cloud [2], [7]. The cloud computing community is trying to address these issues and several solutions have been proposed [7], [8].

While the research with migrating scientific computing applications to the cloud is interesting, most often it is assumed that the scientist who is performing the experiments has significant knowledge of computer science, cloud computing and migration of his applications to the cloud. However, from our observations this is not the case. For example, scientists who are running their experiments currently on grids and clusters are unaware of the things how the environment is configured, how the queues work on clusters and how his job gets executed. All they are interested in is that they submit a job to some queue and after sometime they can collect the results. Having this assumption, in mind we have designed tools that help scientists to migrate their applications to the cloud. The idea is to migrate the complete desktop environment, with which the scientist is working daily, to the cloud directly.

To support this, we have developed the Desktop-to-Cloud-Migration tool (D2CM) that integrates several software libraries to facilitate VM image transformation, migration and management of experiments running in the cloud. The tool also has support for extracting the results and monitoring the health of the cluster while the application is running on the cloud. Once the prototype of the D2CM tool was ready, we used it in migrating several of our scientific experiments to the cloud. This paper addresses our experience with migrating our electrochemistry case study to the cloud. Grid-base Projector-Augmented Wave Method (GPAW) [9] application was used for computing experiments. GPAW is a popular distributed application which in conjunction with the Atomic Simulation Environment (ASE) provides optimized toolset for simulating atomic structures. The legacy system is being used by researchers in the institute of Physical Chemistry at University of Tartu. They use this system to simulate and study the capacitance of metal electrodes with the aim to improve super-capacitors and batteries. The experiments are explained with detailed performance along with their domain specific

necessity. The rest of the paper is organized as follows.

Section II describes the desktop to cloud migration procedure in detail along with the developed tool. Section III explains the considered electrochemical case study in detail along with the conducted experiments and observed results. Section IV provides overview of the related work. Section V concludes the paper with future research directions.

II. Desktop to Cloud Migration

A major hurdle in porting applications to an IaaS cloud is the re-creation of a software environment on the vendor-supplied (maybe supported) OS and cloud instances. The D2CM tool seeks to mitigate potential problems of this often laborious task by taking the entire environment as-is, provided it is within a virtual environment. The D2CM tool is developed in Python and provides two facilities: 1) Programmatical transformation of user-supplied virtual machine images to the target cloud image format. Currently it has support for the Amazon Machine Image format and Eucalyptus compatible XEN[1] image. 2) Life-cycle management of distributed applications hosted on Amazon's Elastic Cloud Computing (EC2)[2] platform (or compatible infrastructure, such as Eucalyptus[3]). The images must have installed an SSH server, configured to allow root login using a private SSH key.

A. Virtual image transformation

EC2's and Eucalyptus' underlying machine virtualization is XEN (Note: Eucalyptus also supports KVM), a widely used open-source technology which supports full and para-virtualization. While Amazon provides a range of already configured images, it also supports user supplied images, provided they function with a kernel using a Xen compatible kernel. The kernel must have pv-ops with XSAVE hypercall disabled[4]. An image may either be configured to boot with Amazon supplied kernels, known as AKIs, or with a kernel packaged within the image itself. The later option is used by D2CM, which uses Xen's PVGRUB boot process. In the kernel update section of the transformation process, the D2CM tool installs a new kernel, either 32 or 64 bit depending on the detected image type, and modifies GRUB configuration to boot with the new kernel by default. In the current implementation, the tool only supports transformation of a single disk partition. If an image has more than one partition, only the primary one can be migrated to EC2.

D2CM uses the Libguestfs library[5] to inspect and manipulate virtual machine images. Using libguestfs is highly preferred to mounting the images within the user's filesystem as directly mounting the image requires the host OS to support the image's filesystem(s) and requires root privileges. Libguestfs' API includes functions to extract partitions from images and inserts SSH key files, which are required during the image migration process. Libvirt [6] is an abstraction library for interacting with desktop hypervisors. The D2CM tool uses libvirt to access the currently supported hypervisor, VirtualBox. In the future, adding support for other popular hypervisors such as KVM, Xen, and VMWare should be greatly eased by the use of libvirt. The currently supported IaaS platforms, EC2 and Eucalyptus, provide the same Amazon Web Services (AWS) API for managing machine instances. The web service is wrapped by several libraries which provide programming language specific bindings. As we developed the D2CM tool using Python, we used the Boto library [7] which is the Python binding for AWS.

In addition to working with a compliant kernel, an image must also be able to mount devices in the manner supplied by the EC2 virtual machine. EC2 instances have a default device mapping that usually does not match the fstab entries within the user-supplied image. EC2 instances come in a variety of types that may have more or less storage devices than the image is configured for. Some instances do not provide swap devices. Most instances provide additional storage devices that the user may want to utilize. To conform to this new virtual hardware environment, D2CM changes the fstab appropriately.

The first step to application migration requires the user to supply the tool with valid AWS credentials. The user then registers a desktop image with the tool. After this, the user may select to initiate the migration process. The tool proceeds with the above mentioned image transformations, making any changes in a separate image copy, preserving the integrity of the source. Upon transformation completion, the tool uploads the image to Amazon's Simple Storage Service (S3) [8] or to Walrus in case of Eucalyptus. Once uploaded, the tool registers the image with the EC2/Eucalyptus, declaring default settings such as suitable system architecture. All images are currently registered as private, authorizing read and boot access to the account which uploaded and registered it. The duration of the entire transformation and upload procedure varies greatly depending on the image size and user's Internet connection speed. At this point, the user's image is ready to run on the EC2 platform. This process must be repeated for each desktop image required for the distributed application.

Figure 1 shows some of the screenshots of D2CM tool. The first screen lets the scientist to enter his Amazon credentials. The second one (top right) shows the migration procedure and the migrated images. The third one shows the deployment creation, specifying the life cycle scripts and the last screenshot shows the experiment before the execution on the cloud. Cloud execution monitoring using D2CM is presented in figure 4.

B. Deployment life cycle management

D2CM supports non-interactive applications which operate on a finite input data set. A single execution of an application is defined as a deployment. A deployment consists of one or more roles. A role is defined by a deployment unique

[1] http://www.xen.org/
[2] http://aws.amazon.com/ec2/
[3] http://open.eucalyptus.com/
[4] http://aws.amazon.com/articles/3967
[5] http://libguestfs.org/
[6] http://libvirt.org/
[7] https://github.com/boto/boto
[8] http://aws.amazon.com/s3/

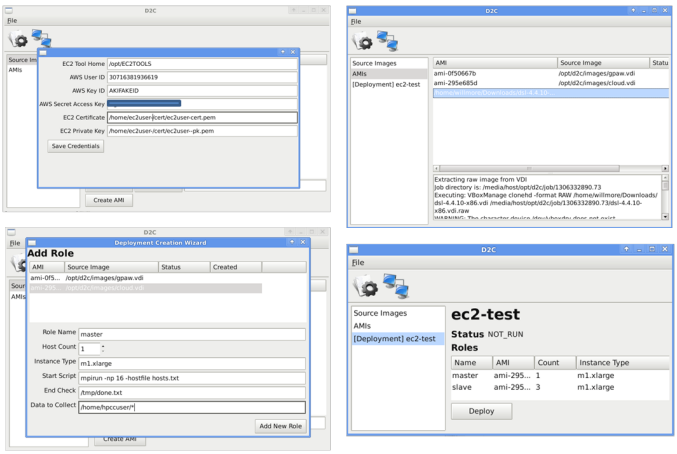

Fig. 1. D2CM screenshots

name, association to an EC2/Eucalyptus image, instance count and type, and life-cycle scripting. There may be a many-to-one relationship between roles and a unique EC2 image. The instance count defines the number of virtual machines of specified type that will be allocated upon deployment start-up. Roles include optional life-cycle scripts that the deployment manager invokes at the appropriate stage. A deployment's life-cycle proceeds through the following phases:

1) Reservation: Upon initiation, the tool iterates through deployment roles and reserves instances matching user specifications. If the request cannot be completely fulfilled by EC2, any successful allocation is released and the process ends. The moment instances are reserved, AWS begins charging the user's account. Prior to actual reservation, the tool prompts the user for confirmation, displaying the per hour rate in USD as determined by the deployment's instance type and count.

2) Contextualization: Processes that operate cooperatively require some discovery mechanism. Contextualization facilitates this discovery. When all instances within a deployment are assigned IP addresses, the tool installs a text file containing all addresses on each host in the deployment. Currently applications must be configured to work with this D2CM specific file, but in the future the location and format of this file will be configurable by the user.

3) Start-up: Once the instances have been acquired, the tool executes any user-defined start actions. If a role contains multiple instances, the start actions will be run on each instance. Currently the tool supplies three classes of start-up actions: synchronous, asynchronous, and upload.

The upload action allows the user to install files on the instances which were not bundled with the image. These files may either correct minor image defects or supply experiment specific input. The synchronous action executes a specified BASH script. The tool executes all scripts for a given role and instance in serial. This functionality allows users to make modifications to deployment hosts that are guaranteed to finish before starting any main scripts, which are run using the asynchronous actions. The asynchronous action executes a script remotely on an instance, returning execution control immediately back to the tool. In contrast to the synchronous action, the user may assume that all asynchronous actions run in parallel. The action is asynchronous in that it logs on to the remote instance, executes the user's script in a daemonizing (with nohup) wrapper, and then immediately ends the remote connection. This is the mechanism by which a user should initiate the main application process. An example asynchronous action script is:

$$mpirun \ -np\ 16\ python\ matrix\_calc.py$$

Once all deployment asynchronous actions have been initiated, the tool changes the deployment state to Completion Monitoring.

4) Completion monitoring: The user must configure the D2CM tool to test for application specific completion. The only option provided currently, is the ability to monitor for the presence of a file on an instance. The tool continually polls the instances of roles with the specified check. If no check is provided for a given role, that role is automatically assumed to be in the finished state. Only when all roles are in the finished state, does the program proceed to finalization.

5) Finalization: The result of supported applications will be a collection of output files. These must be migrated to persistent storage as the cloud instances only provide ephemeral storage, i.e. storage that exists for limited lifespan of the instance. D2CM supports configurable downloading of files from the instances to local storage. This occurs after the remotely executing application has completed and before the instances are terminated.

6) Termination: Upon retrieval of required files, the deployment is shutdown. D2CM terminates all instances associated with the deployment, ending runtime charges.

III. Migration Case Study

To test the migration procedure we ported a computational electrochemistry case study to the cloud. Electrochemistry is an actively studied branch of chemistry, which has been rapidly developing through the late 20th century and 21st century. Among applied solutions provided by electrochemistry different types of batteries are well known. Moreover, devices such as fuel cells, supercapacitors, solar panels are common in mass media in relation to terms as green energy, mobile devices, solar power plants etc. Several noble prizes were awarded during the past decade for discoveries concerning electrochemistry. One of them was shared between Walter Kohn for his development of the "density-functional theory (DFT)" and John A. Pople for his development of "computational methods in quantum chemistry" [9]. The named theory and methods are widely applicable in the field of electrochemisty as part of computational electrochemistry. We applied DFT in order to study room temperature ionic liquid (RTIL) with an ultimate goal to find better design for RTIL based supercapacitors.

[9]http://www.nobelprize.org/nobel_prizes/chemistry/laureates/1998/press.html

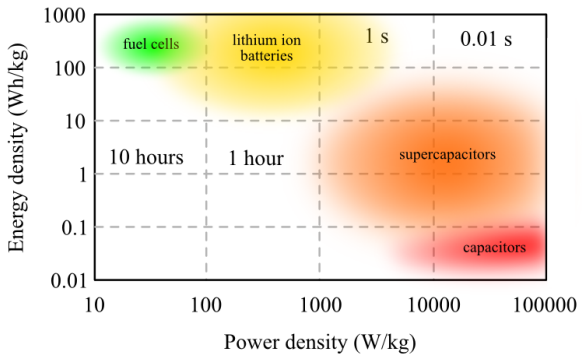

Fig. 2. Comparision of lithium battery and supercapacitor

In the present time, supercapacitors have some definite advantages over batteries. They have impressive charge-discharge time in less than 10 seconds; virtually unlimited, 1000 times larger number of life cycles of 1 million; an order of magnitude higher specific power of 10 kW/kg; excellent operating temperature range from -20 to $65°$C. However, cost per Wh is ten times larger and the amount of energy they can store is ten times lower for supercapacitor compared to a common lithium-ion battery. Capacitors and batteries are marked in the figure 2, representing energy density vs. power density chart for various energy-storage devices. Lithium batteries are somewhere in the upper left corner, and data for supercapacitors are on the right side. "Right and higher" is better in the chart.

Nevertheless, according to the equation:

$$E = \frac{1}{2} C \Delta U^2$$

where E is energy, C is capacitance, and ΔU is potential window, energy and as a result the cost can be improved. Currently potential window is almost the same both for lithium battery and supercapacitor, but may be enlarged for the later by two times from 3 Volts to almost 6 Volts. Capacitance may also be significantly improved for supercapacitors by improving the design of the device. The aim is to improve characteristics of modern supercapacitors and design them in such a way, that they would be commercially competitive with modern batteries and thus would become available at market. Related experiments have been conducted at the University of Tartu over a decade.

A. Experiments and calculations

Modern electrochemistry utilizes experimental data, theoretical models and computational routines. Combining experimental, theoretical and computational approaches allows scientists to compare the characteristics of different structures and material compositions at different levels and magnitudes without actually having access to physical samples. By modifying the model of material composition and structure it is possible to optimize the desirable properties, such as energy, capacitance or power density. Based on experimental data a mathematical model is built, which is then tested by calculations and then the results are treated with developed theoretical model. If the final result is realistic, one may exclude physical experiments from the scheme.

According to the formula, in order to get a higher specific energy, we shall improve capacitance C and potential window ΔU. To do so we first need to understand how these two parameters depend on structure and composition of the system. This goal can be achieved both theoretically by calculations and experimentally. We started with simple models of Au(111) | $EMImBF_4$ interface - representing charged gold surface and single adlayer of anions. Our input model is based on experimental results, which were gathered using scanning tunneling microscope, electrochemical impedance spectroscopy, and cyclic voltampermetry [10], [11], [12]. The results show that at positive potentials a structured layer of anions is formed; at negative potentials cations form an ordered layer structure, and in between both anions and cations are present at the surface. We investigated these cases separately with the help of GPAW. Starting from the given geometry, GPAW finds optimal geometry with the lowest energy and evaluate electron density. Later we use them to recalculate absolute potential. Further we plot energy dependence on the calculated potential and evaluate capacitance value. For hyphotetical Au(111) | F^- interface the evaluated capacitance value is lower compared to the value for Au(111) | BF_4^-. That is a simple dependence on composition of ionic liquid. Our model accounts redistribution of electric charge densities, which are seen in calculations, but not accounted in modern theories and classical molecular dynamics. This encourages us to follow computational approach based on GPAW code.

B. GPAW tool

GPAW tool implements rather novel Grid-based PAW methods [9]. The code and the way it is developed have several distinguished features, which are neccesary for calculations of large electrochemical systems. Firstly, there are two ways of representing the wave functions: one gives less accurate results, however computationally more efficient and provides a good guess for optimized geometry; another is much more accurate, but highly demanding on computational resources. Secondly, the parallelization with MPI can be done over several paremeters: k-points, grid-spacing h etc. Regarding migration to the cloud, it is worth to notice, that GPAW is one of the few codes which is released under the GNU Public License. As an open-source it is constantly developed by a group of developers from Denmark, Finland, Sweden, and Germany, with users from all over the world.

1) Script that is run: In the following script a single optimization is defined via the Atomic Simulation Environment (ASE) python syntax. ASE defines atomic positions, calculators used, optimization procedures and methods, input-output data etc.

```
1  #Define geometry of metal slab + BF4 adlayer:
2  slab = fcc111('Au', size=(2, 2, 3),orthogonal=True
       )
```

```
slab.center(axis=2, vacuum=12)
d=0.8
tFB = Atoms([Atom('B', ( 0,  0,  0)),
             Atom('F', ( d,  d,  d)),
             Atom('F', (-d, -d,  d)),
             Atom('F', (-d,  d, -d)),
             Atom('F', ( d, -d, -d))])
#Transform orientation of BF4 in space
tFB.rotate('y',pi/4,center=(0, 0, 0))
tFB.rotate('x',asin(1/sqrt(3))+pi,center=(0,0,0))
tFB.rotate('z',pi/3,center=(0, 0, 0))
tFB.translate(slab.positions[5]+(0,0,5.118))
#Combine metal slab and BF4
slab += tFB
#Fix second and third layers in the slab (those
    would participate in optimization)
m = [atom.tag > 1 for atom in slab]
slab.set_constraint(constraints.FixAtoms(mask=m))
#Define the initial state
calc = GPAW(
  h=0.16,                  #grip spacing
  kpts=(4, 4, 1),          #number of k-points
  txt='22.txt',            #output file
  parallel={'domain':4},   #parallization option
  xc='RPBE')               #functional
slab.set_calculator(calc)
qn=optimize.QuasiNewton(slab,trajectory='22.traj',
    restart='22.pckl') qn.run(fmax=0.05)
```

The script also specifies a method for geometry optimization (QuasiNewton) and a calculator (GPAW). For the latter we switch from default LDA exchange-correlation functional to a superior RBPE functional. We also specify grid spacing (h) and number of k-points (kpts) needed for calculation of total energy. For each system, functional used and basis set there is a minimal value of energy, which may be reached either with a low degree of accuracy and fast or exactly, but in a long-time. The balance of time and accuracy is dictated by representation of the wave functions, h, kpts and other parameters. Without going into further details, let us note that an optimization finishes when the difference in forces reaches a given value of tolerance (fmax). Calculated energy, optimized geometry and electron charge density are analyzed and used in further modeling. For instance, a special script evaluates electrostatic potential map and also potential drop across modeled interface, which are needed to estimate the capacitance values.

C. Evaluation of the migration

We ran several GPAW experiments on the cloud using the D2CM tool. Figure 3 summarizes some of the experiments showing their scale and cost. Interesting results for the scientists have been observed during the analysis. In fact, the capacitance was evaluated for RTIL | metal interface from DFT results for the first time. Our model is very promising for sensible capacitance predictions and further research is going on in this domain. More detailed discussion of the results is beyond the scope of this paper. We have also tried with varying problem sizes and mixture of Amazon machine types to establish a pattern of scaling for the electrochemistry case.

Figure 4 shows the CPU and memory profile of a GPAW experiment running on 8 m1.large EC2 instances for test ID 34 in figure 3. We instrumented the machine images with

Test ID	Instance Count	Core Count	Total EC2 Units	Total RAM (GB)	Runtime (Hrs)	Cost (USD)
22	1	4	8	15	4.3	3.8
22vdw	2	8	16	30	1	1.52
32	2	8	16	30	45.7	69.92
34	8	32	64	120	46.25	285.76
42	4	16	32	60	26.3	82.08

Fig. 3. Summary of some of the GPAW experiments performed on EC2

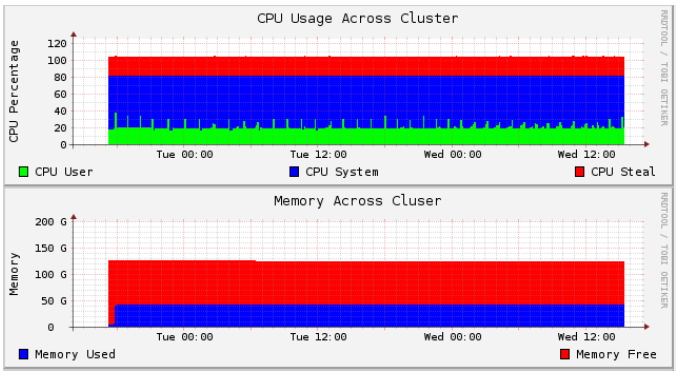

Fig. 4. CPU and memory profile of GPAW experiments

collectd[10], an industry standard in system monitoring. We recorded CPU usage by type (user, system, idle and steal), memory usage, network I/O and system load. The first graph in figure 4 shows average CPU usage of the hosts. The bottom graph depicts total memory usage across the hosts. From the analysis we could observe that CPU is utilized to the full capacity while memory is underutilized for that particular experiment. So the subsequent experiments ran on High-CPU Instances family of EC2. Thus the D2CM tool not only helped in the migration process but also helped us in optimizing the clusters, experiments and thus costs.

IV. Related Work

The D2CM tool provides three core features: migrating desktop virtual machines to IaaS platforms, specifying deployment configurations, and lifecycle management of deployed system. While these features are present to some extent in other tools and systems, D2CM is the only tool we know of that provides all these features in a single tool, which is specifically aimed for performing distributed scientific experiments on the cloud.

The main reason why we have developed the D2CM tool is that the migration of desktop images to cloud has not been a widely supported feature by the cloud industry in the past. However, several tools have been recently published that do aim to provide this service. Amazon has developed VM Import [11] tool for migration of VMware ESX VMDK, Citrix Xen VHD and Microsoft Hyper-V VHD images to Amazon EC2 cloud. In addition to only supporting images from select proprietary systems, the VM Import tool only supports images

[10] http://collectd.org
[11] http://aws.amazon.com/ec2/vmimport/

running Microsoft Windows Server. Amazon has stated their commitment to add more supported platforms, but a concrete roadmap has not been published.

CloudSwitch [12] provides migration and deployment of virtual machines to the cloud and allows to integrate the migrated servers with existing system running locally or in other clouds. Additionally, CloudSwitch provides secure channels and firewall virtual machines that can be deployed to secure the communication and data between local cluster and virtual machines running in different public cloud platforms.

There are multiple providers of tools for IaaS system architecture definition and deployment lifecycle management. The providers of these tools are either IaaS providers themselves, such as EC2, or third parties such as RightScale which only provide management tools. RightScale offers a means of managing systems via deployment definitions, which make use of RightScale ServerTemplates. [13] ServerTemplates defines the machine image and scripts executed during the startup and shutdown phases of a machine.

Amazon provides EC2 system templating through the CloudFormation [14] tool. CloudFormation allows users to describe the images and machine types that comprise the logical roles of a system. While CloudFormation allows for reliable and somewhat flexible deployment of a system, any lifecyle management must be explicitly programmed into the VM images themselves. On Linux, this is supported by Amazon through the CloudInit software package. This contrasts with our approach in D2CM, where we seek to minimize modification of user images.

Birgit Ptzmann and Nikolai Joukov [13] from IBM outline a method for analyzing existing complex IT deployment for the purpose of extracting role templates for efficient migration to the cloud. This differs from the current D2C tool approach which requires the user to build images manually. Such an approach could be adapted to facilitate the migration of complex legacy systems.

V. Conclusions and future work

The paper discusses the migration of scientific computing applications to the cloud. The design and development of D2CM is explained in detail along with the process of running an experiment using the tool. A GPAW based electrochemical case study is considered for evaluating the tool. The conducted experiments are explained in detail. From the analysis of the case study it was observed that D2CM tool not only helps in the migration process but also helps in optimizing the experiments, clusters and thus the costs for conducting scientific computing experiments on the cloud.

We are currently working on extracting scaling parameters for the experiments and using them to predict the cost of experiments in advance. The pricing schemes in public clouds are often very complex to measure. For example Amazon EC2 cloud does not only measure instance (virtual machine) type and running time but also network bandwidth used inside and outside cloud, static IP idle time, size of persistent data stored in the cloud, etc. This is very important for running scientific computing experiments in the cloud as the size and parameters of the experiments are expected to change constantly and thus may require different amount of computing resources every time. For this reason it is important to estimate the computing resources needed for new computing experiments based on the input size and parameters used and estimate the cost of the deployment based on expected deployment time. Currently we are in the process of developing a feasible solution using the linear programming model as part of the European commission funded REuse and Migration of legacy applications to Interoperable Cloud Services (REMICS) project.

[12] www.cloudswitch.com/

[13] www.rightscale.com/products/configuration-framework.php

[14] aws.amazon.com/cloudformation/

Acknowledgment

This research is supported by European Commission via the REMICS project (FP7-257793), Estonian Science Foundation grants ETF9287 and ETF8357, European Regional Development Fund through the Estonian Centre of Excellence in Computer Science and European Social Fund through Mobilitas.

References

[1] M. Armbrust et al., "Above the clouds: A berkeley view of cloud computing," University of California, Tech. Rep., Feb 2009.

[2] S. Srirama, O. Batrashev, P. Jakovits, and E. Vainikko, "Scalability of parallel scientific applications on the cloud," *Scientific Programming*, vol. 19, no. 2, pp. 91–105, 2011.

[3] C. Bunch, B. Drawert, and M. Norman, "MapScale: A Cloud Environment for Scientific Computing," University of California, Computer Science Department, Tech. Rep., 2009.

[4] C. Vecchiola, S. Pandey, and R. Buyya, "High-performance cloud computing: A view of scientific applications," in *Pervasive Systems, Algorithms, and Networks (ISPAN), 2009 10th International Symposium on*, dec. 2009, pp. 4–16.

[5] S. N. Srirama, O. Batrashev, and E. Vainikko, "SciCloud: Scientific Computing on the Cloud," in *The 10th IEEE/ACM Int.l Symposium on Cluster, Cloud and Grid Computing(CCGrid 2010)*, 2010, p. 579.

[6] K. Keahey and T. Freeman, "Nimbus or an open source cloud platform or the best open source ec2 no money can buy," in *Supercomputing*, November 2008. [Online]. Available: http://workspace.globus.org/talks/nimbus-sc08.pdf

[7] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[8] Q. Li and Y. Guo, "Optimization of resource scheduling in cloud computing," in *SYNASC*, 2010, pp. 315–320.

[9] J. Enkovaara, C. Rostgaard, J. J. Mortensen, M. Kuisma, and et al, "Electronic structure calculations with GPAW: a real-space implementation of the projector augmented-wave method," *Journal of Physics: Condensed Matter*, vol. 22, no. 25, p. 253202, Jun. 2010.

[10] G. Pan and W. Freyland, "2D phase transition of PF6 adlayers at the electrified ionic liquid/Au(111) interface," *Chemical Physics Letters*, vol. 427, no. 1-3, pp. 96–100, Aug. 2006.

[11] M. Gnahm, T. Pajkossy, and D. Kolb, "The interface between Au(111) and an ionic liquid," *Electrochimica Acta*, vol. 55, no. 21, pp. 6212–6217, Aug. 2010.

[12] M. T. Alam, J. Masud, M. M. Islam, T. Okajima, and T. Ohsaka, "Differential capacitance at Au(111) in 1-Alkyl-3-methylimidazolium tetrafluoroborate based Room-Temperature ionic liquids," *J. Phys. Chem. C*, vol. 115, no. 40, pp. 19 797–19 804, 2011.

[13] B. Pfitzmann and N. Joukov, "Migration to multi-image cloud templates," in *Proceedings of the 2011 IEEE International Conference on Services Computing*, ser. SCC '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 80–87.