CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. (2012)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.2912

A cloud computing system in windows azure platform for data analysis of crystalline materials

Qi Xing [1] and Estela Blaisten-Barojas [1,2,*,†]

[1] *Computational Materials Science Center, George Mason University, Fairfax, VA 22030, USA*
[2] *School of Physics, Astronomy, & Computational Sciences, George Mason University, Fairfax, VA 22030, USA*

SUMMARY

Cloud computing is attracting the attention of the scientific community. In this paper, we develop a new cloud-based computing system in the Windows Azure platform that allows users to use the Zeolite Structure Predictor (ZSP) model through a Web browser. The ZSP is a novel machine learning approach for classifying zeolite crystals according to their framework type. The ZSP can categorize entries from the Inorganic Crystal Structure Database into 41 framework types. The novel automated system permits a user to calculate the vector of descriptors used by ZSP and to apply the model using the Random Forest™ algorithm for classifying the input zeolite entries. The workflow presented here integrates executables in Fortran and Python for number crunching with packages such as Weka for data analytics and Jmol for Web-based atomistic visualization in an interactive compute system accessed through the Web. The compute system is robust and easy to use. Communities of scientists, engineers, and students knowledgeable in Windows-based computing should find this new workflow attractive and easy to be implemented in scientific scenarios in which the developer needs to combine heterogeneous components. Copyright © 2012 John Wiley & Sons, Ltd.

Received 23 April 2012; Revised 11 July 2012; Accepted 15 July 2012

KEY WORDS: cloud computing; Windows Azure; heterogeneous scientific workflow; machine learning; zeolite structure predictor

1. INTRODUCTION

Cloud computing [1–4] is a model that enables on-demand network access to configurable computing resources that are supplied to the user without service provider intervention [5]. Indeed, a cloud computing platform packages information technology (IT) resources and fetches services to customers such that they simply have to access the interface of the cloud platform to use the services. This is a new paradigm for scientific applications that do not require sophisticated parallelization and are currently performed in small computer clusters [6, 7]. Despite the existence of public clouds [8] such as Amazon [9–11], Google [12, 13], Microsoft [14, 15] providing customers with measured services through the Internet, the potential of cloud computing for scientific applications remains largely unexplored [16–20]. Currently, there is lack of open source workflows geared toward research groups in the sciences, applied mathematics, and most of the engineering communities. This deficiency is a fact because cloud developers have primarily targeted customers in business, government departments, and individuals. The absence of science and engineering consumers using public clouds is recognized by organizations such as the US National Science Foundation [21]. This organization funds fundamental research and can adopt a pay-per-use funding mechanism, if the

*Correspondence to: Estela Blaisten-Barojas, Computational Materials Science Center, George Mason University, 4400 University Dr, MS 6A2, Fairfax, VA 22030, USA.

†E-mail: blaisten@gmu.edu

sciences, engineering, and mathematics communities embrace the cloud computing paradigm. A large number of educational and small-to-medium research laboratories would benefit. Cloud computing differentiates from grid computing because instead of batch job queues, the user receives virtual resources. Of particular importance for scientific research where numerical accuracy is important is that cloud computing offers deployment and control of applications, thus reducing compatibility issues between the application and the hosting environment [22]. However, for cloud computing to become efficient for a given science application, a specific-to-problem computer system workflow needs to be created to link the user application with the IT cloud resources [23–25].

Windows Azure (WA) is a public cloud based on the deployment model *Platform as a Service* (PaaS) that runs on servers located in Microsoft data centers [26]. This cloud became available to developers from the public in April 2010. The WA PaaS is connected to the Internet and consists of a scalable cloud operating system, data storage fabric, and services delivered by physical or logical Windows Server instances. Within PaaS, users are capable of deploying onto the cloud their user-developed or acquired applications created with programming languages, libraries, services, or tools supported by the cloud provider. This implies that the user has a high level of control over the deployed applications and configuration settings for the application-hosting environment, but does not manage or control the underlying cloud infrastructure that includes network, servers, operating systems, and storage [5]. Recent corporate efforts by Microsoft are making WA PaaS attractive for scientists interested in implementing applications with *Software as a Service* (SaaS) [27] and for storage applications [28]. The PaaS approach is different from Amazon's EC2 *Infrastructure as a Service* (IaaS) where the user is provided a host for a virtual machine with processing, storage, networks, and other fundamental computing resources such that he/she can deploy and run any software, including operating systems and applications [29]. In this case, the user has control over operating systems, storage, deployed applications, and limited control of select networking components, but does not manage or control the underlying cloud infrastructure [5].

To address the bottleneck that scientific consumers encounter to use public clouds, in this paper, we describe a new cloud-based scientific computing system, referred to as Structure-Adaptive-Materials-Prediction (SAMP) for bringing the Zeolite Structure Predictor (ZSP) [30] to an automated access through the Web. Our computing system is developed for the WA platform. ZSP is a machine learning model that classifies zeolite crystals according to their framework type (FT) into 41 FTs. The ZSP model is trained on 1473 zeolite entries [31] from the Inorganic Crystal Structure Database (ICSD) [32], is built from a nine-descriptor vector, and uses Random Forest™[33] for the classification task. The accuracy of the ZSP model is 98%. This is the first time a cloud-based computing system is built for automating the calculation of descriptors and their use in machine learning classification and clustering. The cloud infrastructure offered by WA is a good choice for SAMP workflow because of the increased work efficiency that a PaaS-based platform offers to the developer as compared with the work investment needed in an IaaS-based platform. Communities of materials scientists, chemists, engineers, and STEM (Science, Technology, Engineering, and Mathematics) students knowledgable in PC-based Windows environments have a growing interest in the heterogeneity of runtimes used in services and architecture of the WA domain. In particular, the researcher community that employs the ICSD will find that the SAMP compute system allows a fast analysis for classifying new zeolite entries or other inorganic crystals. The computing workflow of SAMP is generalizable for different science applications. It suffices to change the services (codes to be executed) and the third-party software packages (data analysis and visualization) by those adequate to another science application for the compute-system to be usable in a different scientific scenario. The novelty of this workflow is its full implementation in the WA PaaS combining very heterogeneous user-supplied software, which makes it unique for scientific applications. This new compute system will be very useful for applications that are both data intensive and computationally intensive benefiting by the parallelization scheme that supports SAMP. For example, with minor modifications, the full compute-system workflow of SAMP will serve users interested in the scientific open-source software packages LAMMPS [34], NAMD [35], SIESTA [36], CPMD [37], just to mention a few.

This paper is organized as follows. Section 2 gives a description of the cloud environment preparation, the data parsing and manipulation, the four cloud services developed, and the parallel

DOI: 10.1002/cpe

environment developed employing the Server Message Block (SMB) protocol. Section 3 gives performance results of SAMP in WA and comparison with local execution. Section 4 concludes this work.

2. THE STRUCTURE-ADAPTIVE-MATERIALS-PREDICTION COMPUTE SYSTEM IN WINDOWS AZURE

The WA PaaS provides developers with four cloud technologies crucial for developing applications to be run in the cloud. These components are illustrated in Figure 1(a): (i) WA (which requires a Windows-based local environment) for running applications and storing data on servers located at the data centers, (ii) data services in the cloud based on SQL, (iii) distributed infrastructure services to cloud-based and local applications through .NET, and (iv) access to data from Windows Live applications.

The WA component of PaaS has three elements [38] that work together: a compute service that runs applications, a storage service, and a fabric that supports the compute and storage services. For the compute service to use PaaS, developers must create a Windows application consisting of *web roles* and *worker roles*. A web role is a web application in which a user interacting via a web page instructs the system to process certain desired tasks. A worker role is the equivalent of a Windows

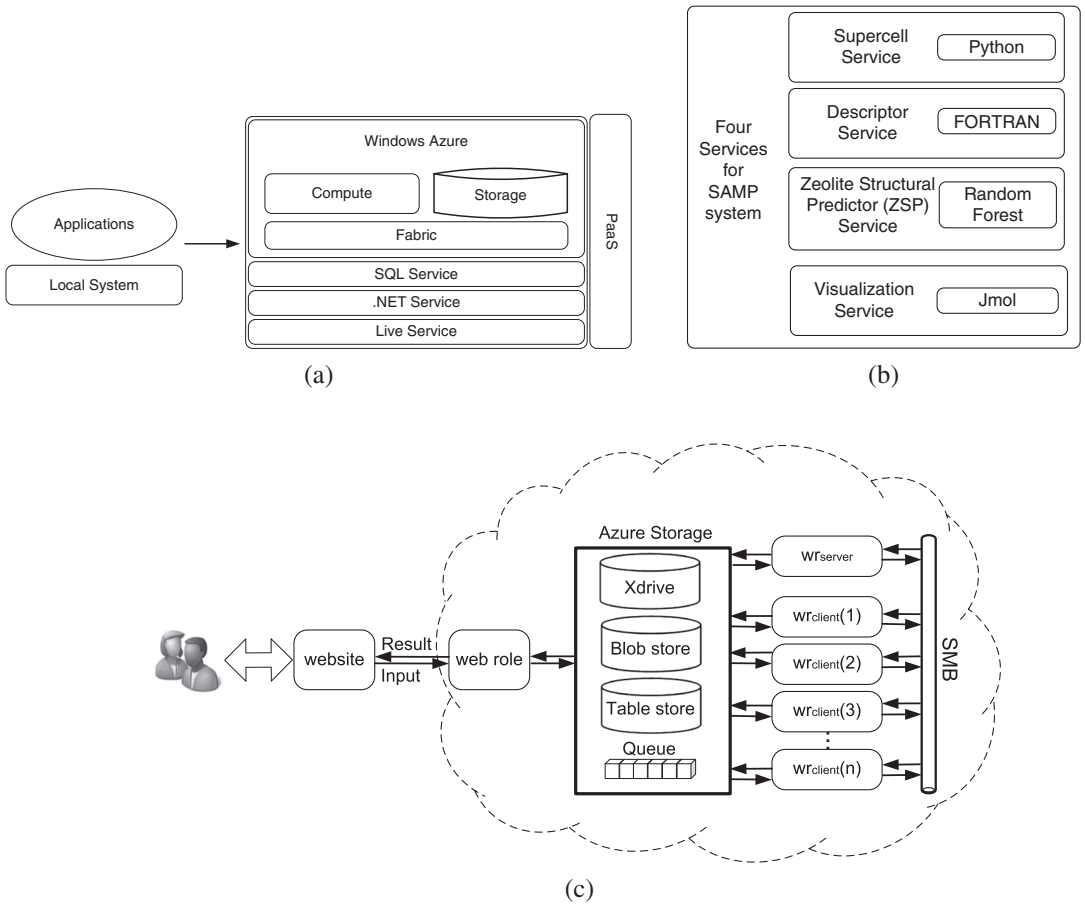


Figure 1. Structure-Adaptive-Materials-Prediction (SAMP) compute system in Windows Azure platform: (a) the four components of PaaS: SQL Service, .NET Service, and Live Service, (b) the four services for the SAMP compute system, supercell, descriptor, ZSP, and visualization that allows use of a battery of in-house codes, (c) a diagram of SAMP compute system workflow.

DOI: 10.1002/cpe

service that can be programmed to execute and terminate desired tasks. For example, a worker role may implement the execution of in-house codes, launch hosted executable applications, write files, and store results.

The WA storage (WAS) [39] provides four services (BLOB, Table, Queue, Drive) [40] that are secure [41], scalable, and easy to access making storage activities durable in the cloud. The BLOB (binary large object) service is the file system providing storage for large datasets such as images, video, documents. BLOB is the simplest way to store text or binary data on WA. The Table storage provides structured storage in the form of tables and supports simple queries on partitions, row keys, and attributes. The Queue service provides a basic queue model to deliver messages asynchronously. This service also enables the communication between the other services. The Drive service (Xdrive) provides NTFS volumes for WA applications. An important characteristic of WAS is that a virtual machine can access the storage facilities with its proper authentication. Therefore, Xdrive can be made available for storing results of a computational task. In addition, WAS may be used for shared storage among several virtual machines.

The WA fabric consists of a large group of servers, all of which are managed by the fabric controller [26]. WA applications and data residing in WAS are distributed through the fabric controller to every server in the fabric. The fabric controller does several useful tasks such as monitoring all running applications, managing the operating system in the cloud, deciding where new applications should run, and choosing servers to optimize hardware utilization.

2.1. The compute system

The ZSP permits automatic determination of the framework types of zeolite crystals. This model is based and trained on crystal data of zeolites cataloged in the ICSD [30, 31, 42]. The purpose of setting the ZSP in the WA cloud is to open up the predictive capabilities of the model to whoever wants to use it with a web browser access. Our expectation is that the WA PaaS should minimize the work of developers when compared with services based on IaaS. One objective is the development of Web-based computing access for automating the calculation of the descriptors that enter in ZSP. The descriptors are needed for analyzing any new dataset of interest to a user. A second objective is to automate the machine learning analysis for classification of the new entries in one of the 41 framework types that ZSP can classify.

With these objectives in mind, we have developed the SAMP compute system. SAMP possesses four services for distributing results from our battery of serial in-house codes: supercell, descriptor, ZSP, and visualization [30, 42]. Figure 1(b) shows schematically these four services and the third-party software uploaded to the cloud required to implement them. Service supercell uses the Python compiler and libraries. Service descriptor requires the dynamic links and runtime libraries of Fortran. Service ZSP uses the classification algorithm Random Forest™[33] as implemented in the open source package WEKA [43]. Service visualization employs the Jmol modeler [44].

The SAMP compute system is a parallel engine running on the WA. A schematic workflow is given in Figure 1(c). Currently, tasks are parallelized into 20 processes. When the user provides a dataset of entries to SAMP, then SAMP deploys one web role and 19 worker roles. The web role stores the user provided zeolite entries, sends them along with messages to WAS and delivers them to 19 worker roles for background processing. The 19 worker roles receive different inputs from the web role. One worker role is designated as server, $\mathrm{wr}_{\mathrm{server}}$, to organize the processing tasks, to host, and to share Xdrive. The remaining worker roles, $\mathrm{wr}_{\mathrm{client}}$(i) with $i = 1 - 18$, execute the tasks in parallel. The input queue contains a number of messages. SAMP extracts one message from the input messages and then dispatches the message to each wr_{client} for execution. Once these tasks are executed, the 18 wr_{client} interact with Xdrive to store results and update the status in the Shared Access Signature (SAS). In addition, SAMP uses the query data-parallel pattern such that several queries can be done simultaneously.

To start any calculations, we deploy the compute system in WA. This requires upload of our in-house codes, compilers, libraries, and software packages to our assigned disk in the cloud where they subsequently reside as long as desired. The uploading time depends on the size of the package to be uploaded. Our 2-GB software package requires a one-time loading time of about 100 min that

DOI: 10.1002/cpe

do not involve computations or invoking services. The deploying time is used for completing five steps: (i) uploading our software package plus VC++ and Java runtime libraries to WA, (ii) initializing the system in WA, (iii) creating the web role, $\mathrm{wr}_{\mathrm{server}}$, and $\mathrm{wr}_{\mathrm{client}}$ instances, (iv) installing the VC++ and Java runtime libraries, and (v) sharing Xdrive among the 18 $\mathrm{wr}_{\mathrm{client}}$ instances using SMB protocol. Unfortunately, VC++ and Java runtime libraries do not reside in WA permanently, otherwise the uploading time will be considerably shorter. During the uploading time, there are several tasks needed as described in Section 2.2.

2.2. Preliminary definitions and data conversion

A WA project containing associations between the web/worker roles and the expected solution is created as first step. The project sets the service definition file and the service configuration file [45]. The service definition file defines the runtime settings for the application, including those worker roles to be used, endpoints, startup, Xdrive information, and Xdrive sharing definition. The service configuration file gives configuration for the web role, defines the number and type of worker roles ($\mathrm{wr}_{\mathrm{server}}$, $\mathrm{wr}_{\mathrm{client}}$), assigns setting values to the various roles, and contains Xdrive information (name, size, disk letter). Windows Azure Tools of Visual Studio [45] provide templates for different types of web roles. Steps to create the $\mathrm{wr}_{\mathrm{server}}$ include: (i) creation of a worker role project [45], (ii) assignment of configuration parameters for connecting to WAS, for mounting Xdrive (where all the in-house codes reside), for initializing services, compilers and libraries, (iii) Xdrive mounting, and (iv) Xdrive sharing by its assigned name (TCP port 445 enabled as an internal endpoint) so that it can receive requests from other roles through the SMB protocol [45–47]. The latter involves: user accounts creation that authenticate the $\mathrm{wr}_{\mathrm{client}}$ instances (user name/password are assigned in the service configuration), granting full access to user accounts, and enabling inbound SMB traffic.

The SAMP compute system uses an ASP.NET (.NET Framework 4.0) web role for building an ASP.NET application with web front-end. The newly created web role specifies the ASP.NET pages and the services included in SAMP. The web role also configures the service definition files and service configuration files for connection to WAS. SAMP defines two types of worker roles for performing background processing: the worker role server $\mathrm{wr}_{\mathrm{server}}$ and the worker role client $\mathrm{wr}_{\mathrm{client}}$. The tasks of the $\mathrm{wr}_{\mathrm{server}}$ are organization, processing, and mounting/sharing the Xdrive. The task of the $\mathrm{wr}_{\mathrm{client}}$ instances is to assign a drive letter (specified in the configuration file) to the shared Xdrive mounted by the $\mathrm{wr}_{\mathrm{server}}$. At start up, the $\mathrm{wr}_{\mathrm{client}}$ instances locate the $\mathrm{wr}_{\mathrm{server}}$, identify the address of the (SMB) endpoint, and assign a letter to the shared drive. At this point the $\mathrm{wr}_{\mathrm{client}}$ instances are ready to read and write to the shared drive. This task emulates writing to a local drive. The global workflow in SAMP proceeds as follows. The web role hosts the front-end logic, handles user requests, and the visualization service. The back-end processing services (supercell, descriptor and ZSP) are implemented through the worker roles. The communications (message, data, entry) between web role and worker roles are connected through WAS. A diagram of such workflow is given in Figure 1(c). We developed our own SAMP task scheduler; a task is serialized into an Azure message and all task messages are afterwards queued into the global task queue that identifies all the uploaded zeolite entries.

A significant portion of the SAMP setup initial time is due to the preparation of Xdrive to be shared by the $\mathrm{wr}_{\mathrm{client}}$ instances. Figure 2 shows Xdrive sharing time for the 18 $\mathrm{wr}_{\mathrm{client}}$ instances. This time indicates how long it takes for each $\mathrm{wr}_{\mathrm{client}}$ instance to share Xdrive through SMB. Seven trials at different moments along a day were performed. As seen in the figure, the time dispersion is significant, even when the process is done at night (trial # 7). Also visible is the high variability on the time in the seven different trials. Thus, we conclude that the dispersion is built into the WA cloud because there is no way of controlling where the $\mathrm{wr}_{\mathrm{client}}$ instances do their work. On average, the sharing time of the 18 $\mathrm{wr}_{\mathrm{client}}$ instance is 9.10 min. We note that the $\mathrm{wr}_{\mathrm{client}}$ instances start to accept tasks when the Xdrive sharing is finished.

A user request involves uploading of a zeolite dataset to the website. The web role creates a message identified by the uploaded file name and pushes it into the file upload queue. Simultaneously, the web role creates a container in Blob storage and stores there the zeolite dataset. Meanwhile, the $\mathrm{wr}_{\mathrm{server}}$ fetches the message from the file upload queue and downloads the zeolite

DOI: 10.1002/cpe

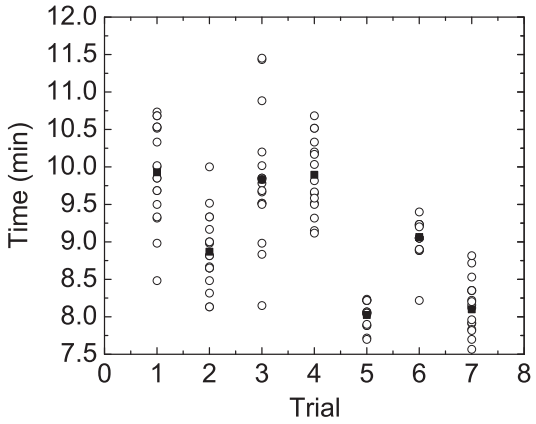


Figure 2. Xdrive sharing time needed to set up the 18 $\mathrm{wr}_{\mathrm{client}}$ instances in seven different trials. The mean sharing time of the 18 $\mathrm{wr}_{\mathrm{client}}$ instances in each trial is reported with a filled square.

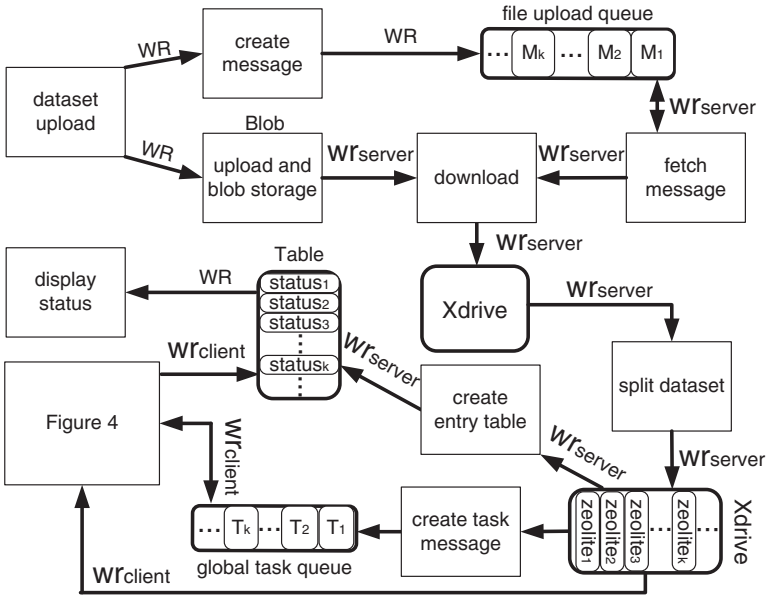


Figure 3. Workflow of the dataset conversion in WAS showing the tasks executed by the web role (WR), the $\mathrm{wr}_{\mathrm{server}}$ and the 18 $\mathrm{wr}_{\mathrm{client}}$ instances.

entries in the Blob container to Xdrive naming the file with the message identifier. If the dataset contains multiple zeolite entries, the $\mathrm{wr}_{\mathrm{server}}$ splits the original file into individual zeolite entry files identified by the zeolite ICSD number. In this case, a task message for each individual zeolite entry is created and pushed into the global task queue. Each message has the zeolite ICSD number as identifier. Next, a table of zeolite entries is created in the Table storage containing the entry ID as its entry index. In addition, the process status of the data conversion, supercell, and descriptor services for each zeolite entry is stored in that table. This entry table is shared with the 18 $\mathrm{wr}_{\mathrm{client}}$ such that the status is updated when processes are finished. The workflow of the data conversion is shown schematically in Figure 3.

2.3. Supercell, descriptor, ZSP, and visualization services

The 18 $\mathrm{wr}_{\mathrm{client}}$ instances compete for tasks from the global task queue. Once a $\mathrm{wr}_{\mathrm{client}}$ instance gets a task message, it parses the zeolite ID from the message and then interacts with Xdrive where the zeolite dataset is stored. Next, the $\mathrm{wr}_{\mathrm{client}}$ instance executes the supercell service and

DOI: 10.1002/cpe

records the calculated supercell of each zeolite in Xdrive. Subsequently, the $\mathrm{wr}_{\mathrm{client}}$ instance creates a supercell container in Blob storage and uploads the supercell dataset to it. When one $\mathrm{wr}_{\mathrm{client}}$ instance successfully executes the supercell service, it updates the process status for that zeolite in the entry table to indicate that execution of the supercell service is concluded. In the event that the $\mathrm{wr}_{\mathrm{client}}$ instance fails to execute the supercell service, the task is restored into the global task queue allowing another $\mathrm{wr}_{\mathrm{client}}$ instance to pick it up. A workflow for these multiple steps is given schematically in the upper portion of Figure 4.

Once supercell service is executed successfully, the $\mathrm{wr}_{\mathrm{client}}$ instance proceeds to execute the descriptor service by fetching the supercell dataset of one entry from Xdrive. Next, the descriptor service is executed, meaning that a fortran code executable and corresponding dynamic link libraries are used in Xdrive. The output is a vector of nine descriptors for each zeolite that is initially stored in Xdrive. To end the descriptor service process, each $\mathrm{wr}_{\mathrm{client}}$ instance updates the status of that zeolite in the entry table, stores the descriptor vector in the table, creates a descriptor container in Blob storage and uploads the descriptor vector to it. The bottom portion of Figure 4 shows schematically this workflow. SAMP compute system provides an interface that binds descriptors stored in the table to the web page managed by the web role. This allows the user to check the calculated values of the descriptors. Furthermore, the user has the ability to request a download of these values, which will be executed by the web role by parsing and fetching the descriptor dataset in Blob storage, and finally completing the downloading request (Figure 5(a)). From the website, the user has the option to request visualization of the supercells once these are generated. SAMP compute system parses and specifies the ICSD identifier of the zeolite supercell, fetches the supercell dataset from the supercell Blob storage and loads it into Jmol (stored in the web role) for visualization directly in the website. A workflow is shown in Figure 5(b).

The ZSP service is tasked to run Random Forest™ using the values of all descriptors stored in the table. When the user requests the ZSP service, the web role creates a ZSP request queue and queues a new message. The $\mathrm{wr}_{\mathrm{server}}$ instance fetches the message from the ZSP request queue, loads the descriptors stored in the Xdrive, and executes Random Forest (running WEKA in Xdrive). This task renders the classification of the zeolite data into 41 framework types, each framework type is a different class. Once this task is finished, the $\mathrm{wr}_{\mathrm{server}}$ creates a ZSP container in Blob storage and uploads there the classification results. The web role binds the web page to this ZSP blob container and results are uploaded and displayed in the Web page every 8 s. A workflow is shown in Figure 5(c).

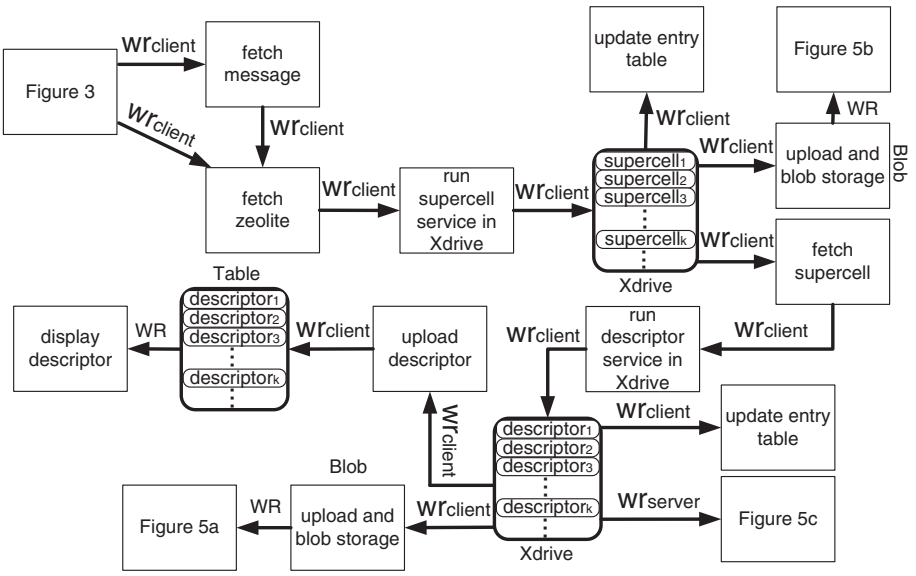


Figure 4. Workflow of supercell service (top) and descriptor service (bottom). WR stands for web role.

DOI: 10.1002/cpe

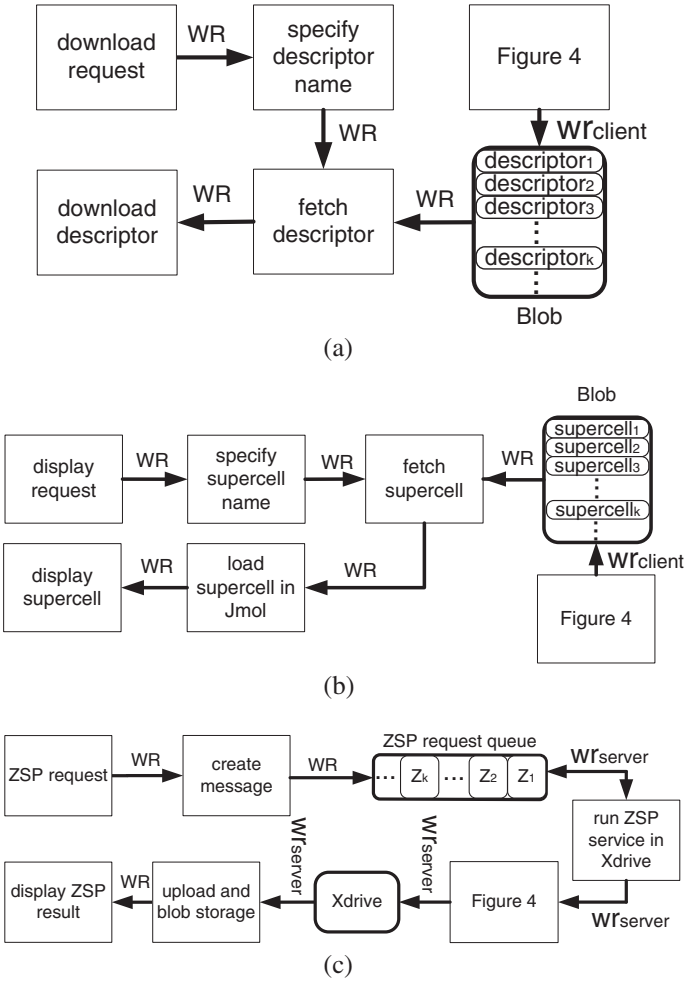


Figure 5. Workflow of how the user accesses the display of (a) the table of descriptors, (b) the visualization service, and (c) the Zeolite Structure Predictor service. WR stands for web role.

3. SYSTEM PERFORMANCE

Several performance tests were carried out to assert time needed for processing and comparison with local execution.

For the processing performance test, the 1473 zeolite entries were split into 11 separate datasets of variable sizes as specified in Table I. Datasets 6, 7 and 8 contain 500, 500 and 473 zeolite entries, respectively, a mere split of the 1473 available entries into three groups. Datasets 1 through 5 contain variable numbers of entries from subset 6, whereas datasets 9 through 11 are formed by merging dataset 6 with 8, dataset 7 with 8, and dataset 6 with 7, respectively. Dataset 12 contains all of the 1473 zeolite entries. These 12 datasets were processed independently in the SAMP compute system to measure the following: (i) the efficiency of the $\mathrm{wr_{client}}$ instances as a function of number of entries processed (mean of time employed by each $\mathrm{wr_{client}}$ to process each dataset <time>); (ii) the processing load used by each $\mathrm{wr_{client}}$ as a function of dataset size (mean of number of entries processed by each $\mathrm{wr_{client}}$ <load>); and (iii) the elapsed processing time used by the 18 $\mathrm{wr_{client}}$ instances for each dataset (from when one $\mathrm{wr_{client}}$ instance begins to process the first zeolite entry to the time when another $\mathrm{wr_{client}}$ instance finishes the process of the last zeolite entry). Based on these measurements, mean time <time>, mean load <load>, their standard deviation $\mathrm{SD_{time}}$, $\mathrm{SD_{load}}$, and coefficient of variation $\mathrm{CV_{time}}$, $\mathrm{CV_{load}}$ are reported in Table I. For dataset 12, the ZSP service

DOI: 10.1002/cpe

Table I. Pertinent quantities monitored in Structure-Adaptive-Materials-Prediction system for assessing performance.

Set	# entries	<time> (min)	SD_{time} (min)	CV_{time} %	<load>	SD_{load}	CV_{load} %	Total time (min)
1	50	10.89	1.53	14.05	2.78	0.55	19.78	13.95
2	100	23.23	1.46	6.28	5.56	0.98	17.63	25.82
3	200	40.28	1.31	3.25	11.11	1.41	12.69	43.72
4	300	60.16	1.96	3.26	16.67	1.37	8.22	63.97
5	400	80.05	1.02	1.27	22.22	1.26	5.67	81.63
6	500	102.36	1.28	1.25	27.78	1.83	6.59	105.37
7	500	104.83	1.45	1.38	27.78	1.63	5.87	106.95
8	473	131.91	1.2	0.91	26.28	1.87	7.12	134.13
9	973	246.47	1.69	0.69	54.06	2.24	4.14	249.28
10	973	243.42	3.19	1.31	54.06	5.54	10.25	255.28
11	1000	206.9	1.43	0.69	55.56	3.96	7.13	208.93
12	1473	356.04	3.69	1.04	81.83	4.79	5.86	361.95

generates our machine learning model, same as that stored in the cloud, allowing users to use our trained model to classify any new zeolite entry [30, 31, 42].

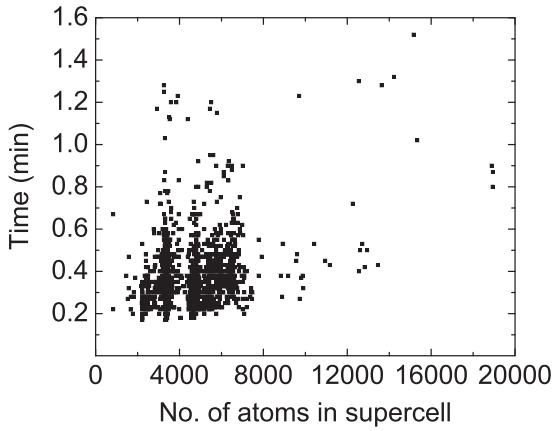
From the statistics summarized in Table I, we conclude that the load on each wr_{client} instance does not degrade the processing time. Small deviations are fully within the error indicated by both SD and CV. Analysis of the SD_{time} and CV_{time} shows that the full workflow time is about 2 min and execution time is homogeneous, independently of the average load on each of the 18 wr_{client} instances. Dataset #10 behaved somehow differently than the others. However, this is expected because dataset #8 contains several entries that are more complex than the rest, and their execution is longer. The processing time of the supercell service is considerably smaller than the descriptor service. Figure 6(a) shows the processing time of the supercell service for each zeolite entry in dataset # 12, as a function of the number of atoms in the supercell and Figure 6(b), shows the processing time of the descriptor service. These two figures clearly show that the majority of the processing time is taken by the descriptor service. Supercell service spends less than 2 min to create any supercell regardless of the number of atoms. It is also interesting to note that in this sample, most of the supercells created have less than 8000 atoms. There are only a few zeolites with supercells containing 8000 to 19,500 atoms, and the descriptor service processing times for them was between 8 and 17 min. This is expected because several descriptors are based on calculations of distances between atoms and their number scales as the square of the number of atoms [30, 31, 42].

By comparing the processing time of the SAMP compute system in the WA using one and 18 wr_{client} instances, we calculated the speedup of the parallel implementation. Speedup was obtained for datasets 1 through 6 and 11 as shown in Table II. A significant improvement of the speedup is obtained for datasets of 400 entries or more, reaching an almost perfect value of about 17.5. The slight degradation is due to the SBM protocol and bandwidth limitation between the 18 wr_{client} instances.

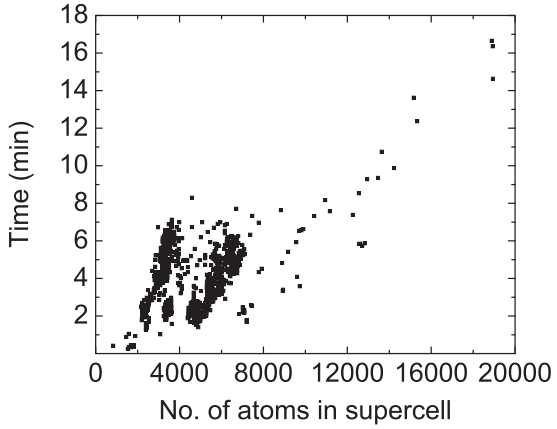
In addition, a comparison was done with the processing time of the same datasets in one dedicated local PC. The WA computers are quad-core AMD™2373 EE at 2.10 GHz with 1.75 GB memory running Windows Server® 2008 Enterprise. Our dedicated local PC is an Intel® core 2 Duo™E8400 at 3.00 GHz with 4 GB memory running Windows 7 Enterprise. Results are summarized in Table II. Although the hardware of the two systems is different, it is obvious that a significant speedup of about 11.0 is rendered by the parallel implementation of SAMP compute system over the local PC. Our services are serial codes and not memory eager, so the discrepancy in memory size is not important.

Signal communication, data transmission, and I/O among the 18 wr_{client} instances affect the bandwidth. WA defines bandwidth based on the number of cores that each wr_{client} instance uses [48]. SAMP compute system assigns only one core to each wr_{client} because our codes and software packages [30, 31, 42] are serial. Thus, the available bandwidth is 100 Mbps. Parallelizing the codes

DOI: 10.1002/cpe



(a)



(b)

Figure 6. Processing time of the 1473 zeolite entries: (a) processing time of the supercell service, (b) processing time of the descriptor service.

Table II. Effect of load on the speedup of the SAMP computing system.

Dataset	18 wr_{client} (min)	1 wr_{client} (min)	Speedup 1 wr_{client}/18 wr_{client}	PC (min)	Speedup PC PC/18 wr_{client}
1	13.95	176.28	12.64	122.13	8.7
2	25.82	380.18	14.7	258.31	10.0
3	43.72	631.67	14.4	443.41	10.1
4	63.97	961.43	15.0	688.75	10.8
5	81.63	1440.82	17.7	917.92	11.2
6	105.37	1842.43	17.5	1145.13	10.9
11	208.93	3545.45	17.0	2254.50	10.8

with openMP to use the quad cores efficiently would upgrade SAMP compute system bandwidth significantly. In turn, our current performance would be enhanced.

4. CONCLUSION

This paper provides a new cloud-based compute system to use with zeolite entries from the ICSD and obtain automatically their classification into 41 framework types. Our novel automated system is composed of four services: supercell, visualization, descriptor, and ZSP, which execute the packages

DOI: 10.1002/cpe

and codes needed for building the machine learning model that classifies zeolites based on their structure. In addition, the underlying model of SAMP compute system employs a minimum of resources in the cloud. In fact, the SMB protocol used to share the services and Xdrive among the $\mathrm{wr}_{\mathrm{client}}$ instances enables an excellent parallelism such that when enough data is analyzed, an almost perfect speedup is obtained. For applications that require embarrassingly parallel workloads, our compute system renders excellent performance at low cost. In addition, the pay-as-you go cloud paradigm united with an efficient compute system may gain popularity as a green and clean approach to computing in the sciences and engineering [49].

We were unable to test the scalability of our model because the resources in WA provided by our sponsor were limited to 20 processors. It would be interesting to test the upper limit at which the speedup of the parallel implementation starts to degrade. In addition, a larger impact in the science community would be gained by Windows Azure platform, if certain software such as VC++/Java runtime libraries would reside permanently in the cloud. This would reduce significantly the time to load the system to the cloud, that at 100 min, is currently too long. An alternative implementation in which a drive is additionally distributed to each worker role client instance is currently under development for improving SAMP performance.

The cloud-based compute system developed is easily generalizable. It suffices to change the services (codes to be executed) and other science and engineering applications that manipulate data can use this cloud compute system. Such applications are usually data intensive and computationally intensive. Both will benefit by the parallelization scheme that supports the SAMP compute system. In particular, the researcher community that employs classical and quantum scientific open-source packages for atomistic simulations (LAMMPS [34], NAMD [35], SIESTA [36], CPMD [37], among others) will find that the SAMP compute system allows access to resources in the WA cloud that otherwise might be difficult to procure with local hardware.

In summary, SAMP compute system is one of the very few cloud implementations currently available to the science community. Its use in Windows Azure will prove simple and robust, offering high parallelism and scalability.

ACKNOWLEDGEMENTS

This work was supported under the National Science Foundation grant CHE-0626111. The computing time allocation in the Windows Azure cloud was awarded by Microsoft Research Division QX gratefully acknowledges the fellowship support from Southwest Jiangton University.

REFERENCES

1. Buyya R, Yeo CS, Venugopa S, Broberg J, Brandic I. Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computing Systems* 2009; **25**:559–616.
2. Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, Zaharia M. A view of cloud computing. *Communications of the Association for Computing Machinery* 2010; **53**:50–58.
3. Watson P, Hiden H, Woodman S. e-Science Central for CARMEN: science as a service. *Concurrency and Computation: Practice and Experience* 2010; **22**(17):2369–2380.
4. Yin JW, Ye YM, Wu B, Chen ZN. Cloud computing oriented network operating system and service platform. In *IEEE Int. Conf. on Pervasive Computing and Communications Workshops*. IEEE: Seattle, WA, March 21, 2011; 111–116.
5. Mell P, Grance T. The NIST Definition of Cloud Computing. Publication 800-145, National Institute of Science and Technology, 2011. (Available from: http://csrc.nist.gov/SP800-145.pdf) [Accessed date: July 10, 2012].
6. Srirama SN, Jakovits P, Vainikko E. Adapting scientific computing problems to clouds using MapReduce. *Future Generation Computing Systems* 2012; **28**:184–192.
7. Yuan D, Yang Y, Liu X, Chen JJ. A data placement strategy in scientific cloud workflows. *Future Generation Computing Systems* 2012; **26**:1200–1214.
8. Peng JJ, Zhang XJ, Lei Z, Zhang BF, Zhang W, Li Q. Comparison of several cloud computing platforms. In *2nd. Int. Symp. on Information Science and Engineering (ISISE 2009)*. IEEE: Shanghai, China, Dec. 26, 2009; 23–27.
9. Amazon elastic compute cloud *(Amazon EC2)*. (Available from: http://aws.amazon.com/ec2/) [Accessed date: April 09, 2012].
10. Akioka S, Muraoka Y. HPC benchmarks on Amazon EC2. In *Proc. IEEE 24th Int. Conf. on Advanced Information Networking and Applications Workshops (WAINA 2010)*. IEEE: Perth, WA , Australia, April 20, 2010; 1029–1034.
11. Walker E. Benchmarking Amazon EC2 for high-performance scientific computing. *Login* 2008; **33**:18–33.
12. Ciurana E. *Developing with Google App Engine*. Apress: New York, 2009.

DOI: 10.1002/cpe

13. Widera P, Krasnogor N. Protein models comparator: scalable bioinformatics computing on the Google App Engine platform. *Computing Research Repository* 2011; **1**:1–8.
14. Lu W, Jackson J, Barga R. AzureBlast: a case study of developing science applications on the cloud. In *Proc. 19th ACM Int. Symp. on High Performance Distributed Computing (HPDC '10)*. ACM: Chicago, IL, June 20, 2010; 413–420.
15. Hill Z, Li J, Mao M, Ruiz-Alvarez A. Early observations on the performance of Windows Azure. In *Proc. 19th ACM Int. Symp. on High Performance Distributed Computing (HPDC '10)*. ACM: Chicago, IL, June 20, 2010; 135–146.
16. de Oliveira D, Ogasawara E, Ocana K, Baiao F, Mattoso M. An adaptive parallel execution strategy for cloud-based scientific workflows. *Concurrency and Computation: Practice and Experience* 2011; [Online]. DOI: 10.1002/cpe.1880.
17. Gunarathne T, Wu T-L, Choi JY, Bae S-H, Qiu J. Cloud computing paradigms for pleasingly parallel biomedical applications. *Concurrency and Computation: Practice and Experience* 2011; **23**(17):2338–2354.
18. Yuan D, Yang Y, Liu X, Zhang G, Chen J. A data dependency based strategy for intermediate data storage in scientific cloud workflow systems. *Concurrency and Computation: Practice and Experience* 2010; **24**:956–976.
19. Vockler J-S, Juve D, Deelman E, Rynge M, Berriman B. Experiences using cloud computing for a scientific workflow application. In *Proc. of the 2nd Int. Workshop on Scientific Cloud Computing (ScienceCloud '11)*. ACM: San Jose, CA, June 8, 2011; 15–24.
20. Li X, Wang Y, Chen X. Cold chain logistics system based on cloud computing. *Concurrency and Computation: Practice and Experience* 2011; [Online]. DOI: 10.1002/cpe.1840.
21. National Science Foundation, Computing in the Cloud. (Available from: http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=503291) [Accessed date: July 10, 2012].
22. Iosup A, Ostermann S, Yigitbasi MN, Prodan R, Fahringer T, Epema DHJ. Performance analysis of Cloud computing services for many-tasks scientific computing. *IEEE Transactions on Parallel and Distributed Systems* 2011; **22**:931–945.
23. Toews E, Satchwill B, Rankin R, Shillington J, King T. An internationally distributed cloud for science: the cloud-enabled space weather platfrom. In *Proc. of the 2nd Int. Workshop on Software Engineering for Cloud Computing (SECLOUD '11)*. ACM: Waikiki, Honolulu, HI , USA, May 21–28, 2011; 1–7.
24. Li Q, Hao QF, Xiao LM, Li ZJ. An integrated approach to automatic management of virtualized resources in cloud environments. *The Computer Journal* 2011; **54**:905–919.
25. Armstrong D, Djemame K. Performance issues in clouds: an evaluation of virtual images propagation and I/O paravirtualization. *The Computer Journal* 2011; **54**:836–849.
26. Chappell D. Introducing the Azure services platform. *Technical Report*, DavidChappell & Associates, San Francisco, California, USA, 2008.
27. Li J, Humphrey M, Agarwal D, Jackson K, van Inger C, Youngryel B. eScience in the cloud: A MODIS satellite data reprojection and reduction pipeline in the Windows Azure platform. In *IEEE Int. Symp. on Parallel & Distributed Processing (IPDPS)*. IEEE: Atlanta, TX, April 19, 2010; 1–10.
28. Zhou L, Varadharajan V, Hitchens M. Enforcing role-based access control for secure data storage in the Cloud. *The Computer Journal* 2011; **54**:1675–1687.
29. Ostermann S, Iosup A, Yigitbasi N, Prodan R, Fahringer T, Epema D. A performance analysis of EC2 cloud computing services for scientific computing. *Cloud Computing* 2010; **34**:115–131.
30. Yang SJ, Lach-hab M, Vaisman II, Blaisten-Barojas E. Identifying zeolite frameworks with a machine learning approach. *Journal of Physical Chemistry* 2009; **113**:21721–21725.
31. Yang SJ, Lach-hab M, Vaisman II, Blaisten-Barojas E. Framework-type determination for zeolite structures in the inorganic crystal structure database. *Journal of Physical and Chemical Reference* 2010; **39**:33102–33146.
32. FIZ/NIST Inorganic Crystal Structure Database. (Available from: http://www.nist.gov/srd/nist84.cfm) [Accessed date: April 12, 2012].
33. Breiman L. Random forests. *Machine Learning* 2001; **45**:5–32.
34. LAMMPS: molecular dynamics simulator, Sandia National Laboratory, US Department of Energy. (Available from: http://lammps.sandia.gov) [Accessed date: July 10, 2012].
35. NAMD: scalable molecular dynamics, theoretical and computational biophysics group, Beckman Institute for Advanced Science and Technology, University of Illinois at Urbana-Champaign. (Available from: http://www.ks.uiuc.edu/Research/namd/) [Accessed date: July 10, 2012].
36. SIESTA: Spanish initiative for electronic simulations with thousands of atoms. (Available from: http://www.icmab.es/dmmis/leem/siesta/) [Accessed date: July 10, 2012].
37. CPMD: parallelized plane wave/pseudopotential density functional theory for ab-initio molecular dynamics. (Available from: http://www.cpmd.org/) [Accessed date: July 10, 2012].
38. Jennings R. *Cloud Computing with the Windows Azure Platform*. Wrox: UK, 2009.
39. Calder B, Wang J, Ogus A, Nilakantan, Skjolsvold A, McKelvie S, Xu Y, Srivastav S, Wu JS, Simitci H, Haridas J, Uddaraju C, Khatri H, Edwards A, Bedekar V, Mainali S, Abbasi R, Agarwal A, ul Haq MF, ul Haq, Bhardwaj D, Dayanand S, Adusumilli A, McNett M, Sankaran S, Manivannan K, Rigas L. Windows Azure storage: a highly available cloud storage service with strong consistency. In *Proc. 23rd ACM Symp. on Operating Systems Principles (SOSP 2011)*. ACM: Cascais, Portugal, October 23, 2011; 143–157.

40. Windows Azure Storage Team. (Available from: http://blogs.msdn.com/b/windowsazurestorage/) [Accessed date: April 09, 2012].
41. Xi K, Tang Y, Hu JK. Correlation keystroke verification scheme for user access control in cloud computing environment. *The Computer Journal* 2011; **54**:1632–1644.
42. Carr DA, Lach-Hab M, Yang SJ, Vaisman II, Blaisten-Barojas E. Machine learning approach for structure-based zeolite classification. *Microporous and Mesoporous Materials* 2009; **117**:339–349.
43. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH. The WEKA data mining software: an update. *Association of Computing Machinery SIGKDD Explorations Newsletter* 2009; **1**:10–18.
44. Jmol: an open-source java viewer for chemical structures in 3D. (Available from: http://www.jmol.org/) [Accessed date: April 09, 2012].
45. Windows Azure Service. (Available from: http://msdn.microsoft.com/en-us/library/windowsazure/) [Accessed date: April 09, 2012].
46. Sharpe R. Just what is SMB? (Available from: http://www.samba.org/cifs/docs/what-is-smb.html) [Accessed date: April 09, 2012].
47. Using SMB to share a Windows azure drive among multiple role instances. (Available from: http://blogs.msdn.com/b/windowsazurestorage/archive/2011/04/16/using-smb-to-share-a-windows-azure-drive-among-multiple-role-instances.aspx) [Accessed date: April 09, 2012].
48. Kommalapati H. Windows azure capacity assessment. (Available from: http://blogs.msdn.com/b/hanuk/archive/2011/02/01/windows-azure-capacity-assessment.aspx) [Accessed date: April 09, 2012].
49. Berl A, Gelenbe E, Di Girolamo M, Giuliani G, De Meer H, Dang MQ, Pentikousis K. Energy-efficient cloud computing. *The Computer Journal* 2010; **53**:1045–1051.

DOI: 10.1002/cpe