THE CLOUD COMPUTING RESEARCH ENGAGEMENT PROGRAM

Dennis Gannon, Juan Vargas

Technology and Policy Group

January 24, 2012

Microsoft Co

TABLE OF CONTENTS

THE CLOUD COMPUTING RESEARCH ENGAGEMENT PROGRAM

The Cloud Computing Research Engagement Program was created to introduce cloud computing and the Azure platform to the research community. The goal has been to broaden the research capabilities of scientists and scholars, foster collaborative research communities and, in doing so, accelerate scientific discovery globally. Our vision for the future is one where researchers will be able to easily interact with massively scalable data analysis tools and services directly accessible from their desktops, laptop or mobile devices in the same way they now interact with Web search and other online resources. We believe that the cloud can transform how research is conducted, accelerating scientific exploration, discovery and results. To meet this goal, we engaged the creative minds of the global research community by building partnerships with government-sponsored research agencies and laboratories.

We have established agreements with the following agencies and institutions:

1. US National Science Foundation has created a new program called "Computing in the Cloud" (CiC). There are two phases to their funding. In the first phase 12 existing grants applied for a supplement to their current funding and received an allocation of our Windows Azure resources. In the second, larger phase, 15 new projects were funded with a substantial financial award and significant Azure resource allocation.
2. As part of the European Commission's FP7 funding program, Microsoft initiated a large project to demonstrate how Windows Azure could interoperate with the existing European science grid and with other clouds. The project was funded with 4,500,000 € from the EC. See: http://www.venus-c.eu/Pages/Home.aspx for project details. EMIC is driving the development of this project for Microsoft as primary architect. In early 2011 the project had an open call which added 15 new applications for Venus-C. In addition there are seven initial application projects that are being ported to Windows Azure and are driving the requirements for cloud interoperability. Several of those are described here.
3. The U.K.' Engineering and Physical Sciences Research Council (EPSRC) is funding three 'digital hubs' which will examine how new technologies can be used to enhance quality of life for everyone. The emphasis is on developing innovative, inclusive products and services. The Microsoft agreement is with the Horizon project (the "Digital Economy Hub") is based at the University of Nottingham.
4. INRIA is France's premier information technology research laboratory. This new MSR-INRIA Joint Centre track involves research into neurobiology imaging using Windows Azure.
5. Japan's National Informatics Institute. The participating researchers are part of the NII's "New IT infrastructure for the Information-Explosion Era" project known as InfoPlosion, which is centered on the creation of novel technology for efficient and trusted information retrieval in a world of heterogeneous information sources.
6. Two separate agreements were made in Australia involving the three major national research organizations. The first agreement is with NICTA (National ICT Australia), Australia's Information and Communications Technology (ICT) Centre of Excellence. They are an independent company in the business of research, commercialization and research training. With over 700 people, NICTA is the largest organization in Australia dedicated to ICT research. The second agreement is with the Australian National University (ANU) and the National Computational Infrastructure (NCI). A third

partner that received resources from the project is the Commonwealth Scientific and Industrial Research Organization (CSIRO).

7. Taiwan's Department of International Cooperation of the National Science Council (NSC) announced the first four Taiwanese grant recipients who will receive advanced cloud computing resources as part of the company's Global Cloud Research Engagement Initiative launched last year. The Initiative provides opportunities for NSC-funded research projects to receive access to Window Azure Cloud computing resources. Additional NSC-granted principal investigators looking to join this Initiative are still encouraged to submit proposals for consideration.
8. The most recent agreement is with the Chinese National Academy of Science and their Computer Network Information Center. This agreement and the initial projects inside CNIC are being launched in January of 2012. In addition to the CNIC internal projects, there will be 12 projects in China from the top universities.

STATUS REPORTS FROM THE INITIAL PROJECTS

As of January 23, 2012, the projects in this program have consumed $744,828 in Azure Resources. We solicited reports from the top 30 research projects (as measured by resource consumption). We asked each project to provide us with a brief description of the project and a detailed technical description of how they used Azure. In addition, we were very interested in their experience building applications on Azure. We also asked them the following questions.

1. Many large scientific problems that can be solved on the cloud could also be solved on a large supercomputer. Are there any aspects of your computation or approach that were better suited to solution on the cloud? ("I don't have access to a supercomputer" is a fair answer).
2. The median amount of cloud resource consumed by the group in this study has cost about $25,000 for 2011. If you had a way to request this amount from your funding agency for cloud resources in the future, would you do so? Or would you prefer to request the same amount to purchase new hardware that you would manage yourself? If so, would you be able to accomplish the same research results?
3. What were the biggest challenges of working with the Azure cloud resource?

Twenty nine of the thirty responded to our request. In the following sections of this report we present their full responses.

TWISTER4AZURE: DATA ANALYTICS IN THE CLOUD

Thilina Gunarathne, Xiaoming Gao and Judy Qiu, Indiana University

Genome-scale data provided by next generation sequencing (NGS) has made it possible to identify new species and infer the evolutionary relationships between organisms. These techniques have been applied in medicine, such as to screen for recurrent mutations in cancer for use as biomarkers, to classify diseases and suggest treatment. However, developing a complete understanding of a genomic dataset relies heavily on a set of data analytics tools that are tractable to analyze potentially billions of read sequences, where the challenges are both unprecedented at scale and in complexity. Researchers of Informatics and Computing, Biology and Medicine at Indiana University have worked together on a NIH project to investigate emerging scalable computing systems interoperable of Cloud and HPC that meet the common needs across a series of life sciences problems [1]. This effort is exemplified by our bioinformatics pipeline of data storage, analysis, and visualization [2]. At its core, parallel programming paradigms such as Mapreduce and MPI provide powerful large-scale data processing capabilities. Our research is to clarify which applications are best suited for Clouds; which require HPC and which can use both effectively. The long-term goal is enabling cost effective and readily available analysis tool repository that removes the barrier of research in broader community -- making data analytics in the Cloud a reality.

Integrating Scientific Challenge: A Typical Bioinformatics Pipeline

The study of microbial genomes is complicated by the fact that only small number of species can be isolated successfully and the current way forward is metagenomic studies of culture-independent, collective sets of genomes in their natural environments. This requires identification of as many as millions of genes and thousands of species from individual samples. New sequencing technology can provide the required data samples with a throughput of 1 trillion base pairs per day and this rate will increase. A typical observation and data pipeline is shown in Fig. 1 with sequencers producing DNA samples that are assembled and subject to further analysis including BLAST-like comparison with existing datasets as well as clustering, dimension reduction and visualization to identify new gene families. The initial parts of the pipeline fit the Mapreduce (e.g. Hadoop) or many-task Cloud (e.g. Azure) model but the latter stages involve parallel linear algebra for the data mining (e.g. MPI or Twister). It is highly desirable to simplify the construction of distributed sequence analysis pipelines with a unified programming model, which motivated us to design and implement Azure4Twister. Twister and Twister4Azure interpolate between MPI and Mapreduce and, suitably configured, can mimic their characteristics, and, more interestingly, can be positioned as a programming model that has the performance of MPI and the fault tolerance and dynamic flexibility of the original Mapreduce.

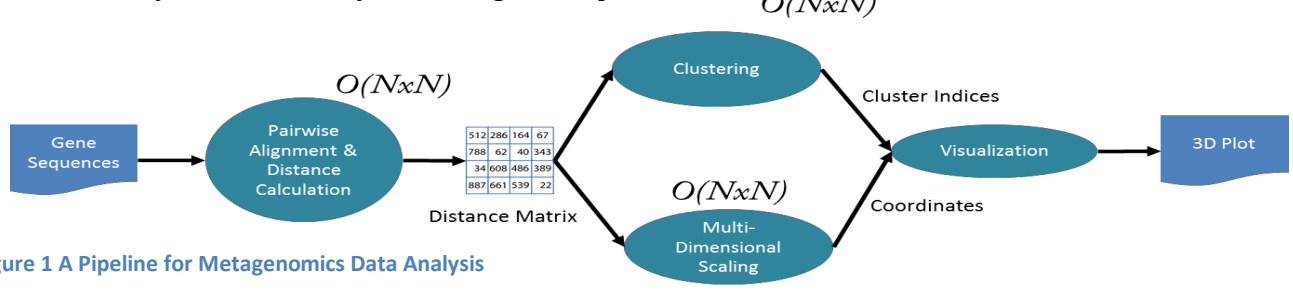


Figure 1 A Pipeline for Metagenomics Data Analysis

Technologies & Applications

Our applications can be classified into three main categories based on their execution pattern, namely pleasingly parallel computations, Mapreduce computations and iterative Mapreduce computations. Twister4Azure [3] distributed decentralized iterative Mapreduce runtime for Windows Azure Cloud, which is the successor to MRRoles4Azure [4] Mapreduce framework and the Classic Cloud pleasingly parallel framework [5], was used as the distributed cloud data processing framework for our scientific computations. Twister4Azure extends the familiar, easy-to-use Mapreduce programming model with iterative extensions, enabling a wide array of large scale iterative as well as non-iterative data analysis and scientific applications to utilize Azure platform easily and efficiently in a fault-tolerant manner, supporting all three categories of applications.

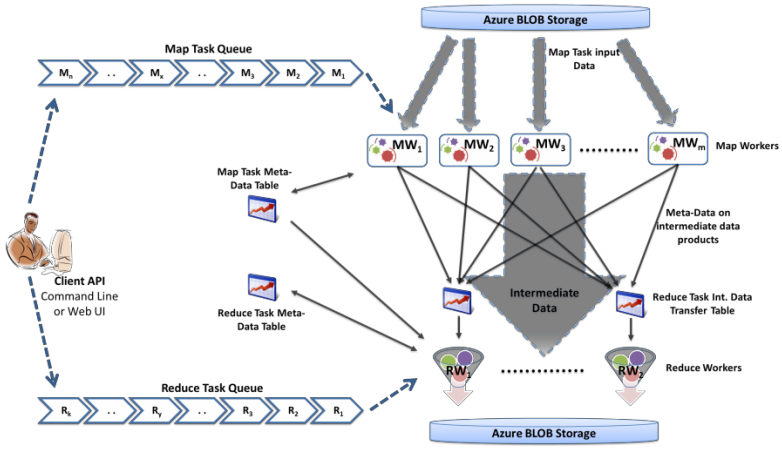


Figure 2 MRRoles4Azure Architecture

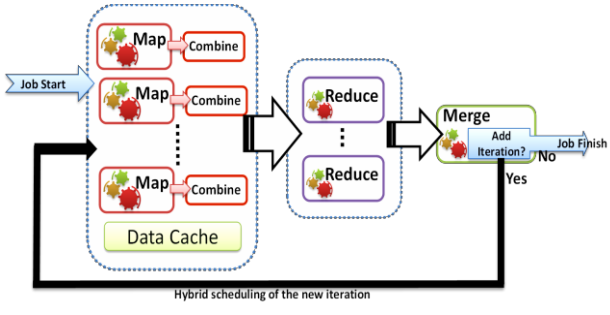


Figure 1 Twister4Azure programming model

Twister4Azure utilize the eventually-consistent, high-latency Azure cloud services effectively to deliver performance comparable to (non-iterative) and outperforming (for iterative computing) traditional Mapreduce runtimes. Twister4Azure supports multi-level caching of data across iterations as well as among workers running on the same compute instance and utilizes a novel hybrid task scheduling mechanism to perform cache aware scheduling with minimal overhead. Twister4Azure also supports data broadcasting, collective communication primitives as well as the invoking of multiple MapRedue applications inside an iteration.

Pleasingly parallel Computations

We performed Cap3 sequence assembly (Fig. 4), BLAST+ sequence search and dimension reduction interpolation computations on Azure using this framework. The performance and scalability are comparable to traditional Mapreduce run times [3]. For Cap3, we assembled up to 4096 FASTA files (each containing 458 reads) in less than one hour using 128 Azure small instances with a cost of around $16. With BLAST+, the execution of 76800 queries using 16 Azure large instances was less than one hour with a cost of around $12.

Mapreduce Type Computations

We performed SmithWatermann-GOTOH (SWG) pairwise sequence alignment computations on Azure [3][4] with performance and scalability comparable to the traditional Mapreduce frameworks running on traditional clusters (Fig. 5). We were able to perform up to 123 million sequence alignments using 192 Azure small

instances with a cost of around 25$, which was less than the cost it took to run the same computation using Amazon ElasticMapReduce.

Iterative Mapreduce Type Computations

The third and most important category of computation is the iterative Mapreduce type applications. These include majority of data mining, machine learning, dimension reduction, clustering and many more applications. We performed KMeans Clustering (Fig. 6) and Multi-Dimensional Scaling (MDS) (Fig. 7) scientific iterative Mapreduce computations on Azure cloud. MDS consists of two Mapreduce computations (BCCalc and StressCalc) per iteration and contains parallel linear algebra computations as its core. Fig. 6 shows performance measurements of Azure are comparable or a factor of 2 to 4 better than those of the traditional MapReduce runtimes deployed on up to 256 instances and for jobs with tens of thousands of tasks.

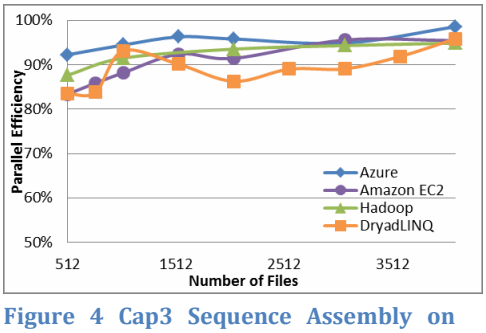


Figure 4 Cap3 Sequence Assembly on 128 instances/cores

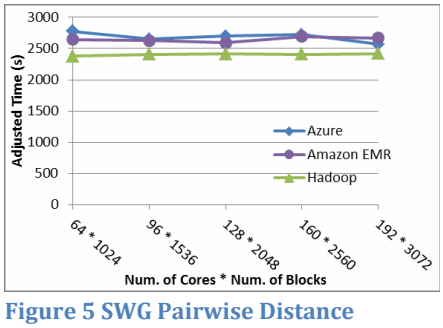


Figure 5 SWG Pairwise Distance Calculation Performance

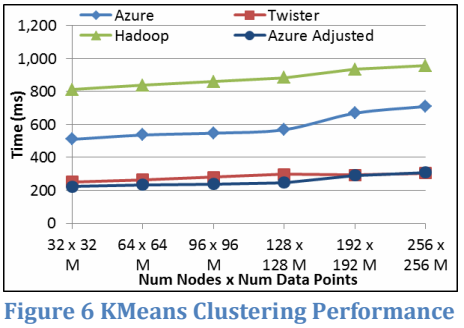


Figure 6 KMeans Clustering Performance

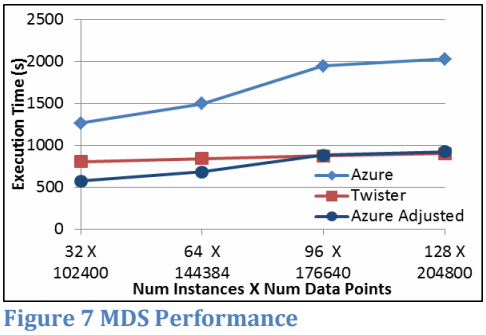


Figure 7 MDS Performance Transmission Speeds

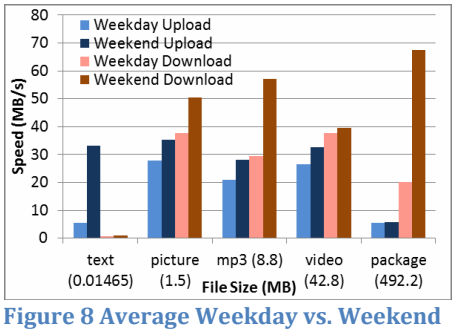


Figure 8 Average Weekday vs. Weekend

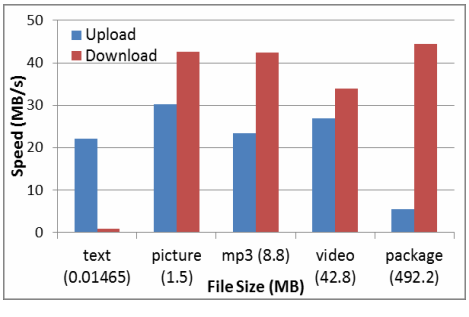


Figure 9 Average Upload vs. Download Speed

Blob storage data transmission performance

We conducted performance tests on the data transmission speed of the Windows Azure Blob service. The tests were done on a virtual machine hosted by the Windows Azure cloud, so the data transmission measured was intra-cloud traffic. Aspects tested include: data transmission speed for objects of different sizes; download vs. upload speed; weekday performance vs. weekend performance, etc. We illustrate our analysis in Fig. 8 and Fig. 9 and summarize three conclusions below.

1. Both upload and download performance fluctuate, but download speed is generally faster than upload, suggesting Azure Blob may be a better option for storing input than saving output. Namely, Azure Blob is not suitable for those scientific applications that generate a huge amount of output.

2. Download speed is generally faster for files of hundreds of MB than for those of tens of MB. So packing scientific input data into larger files may help improve the data transmission efficiency.

3. The top download speed could be high (over 60MB/s), but real-time speed fluctuates a lot. Keeping the computation jobs close to where the Azure data blobs will be important for scheduling scientific jobs in Azure.

Some Lessons Learned

The overall major challenge for this research is building a system capable of handling the incredible increases in dataset sizes while solving the technical challenges of portability with scaling performance and fault tolerance using an attractive powerful programming model. Further, these challenges must be met for both computation and storage. Cloud enables persistent storage like Azure Blob. Mapreduce leverages the possibility of collocating data and compute and provides more flexibility to bring data to compute, bring data close to compute or bring compute to data. In concert with virtualization technology, data center model like Azure Cloud is well suited for hosting data analysis for bioinformatics applications as services on demand. Below we give specific comments on Azure.

Azure Programming Model Issues

1. Intermittent performance inconsistency of Azure instances when executing long running memory intensive applications.

- Fig. 10 shows a sample histogram of computational tasks in a MDS iterative computation. Adjacent blue & red areas represent a single iteration. In ideal conditions, the time taken for tasks in an iteration should be nearly identical across the iteration except for the first iteration. However, after several iterations we started to notice that some tasks randomly take much longer to finish. This result in slowing down of the whole iteration and cause degradation of the computation efficiency as we can see in Fig. 7. We are still investigating the cause for this anomaly and we suspect it to be a behavior of Azure instances after stressing the instance memory for a certain time.

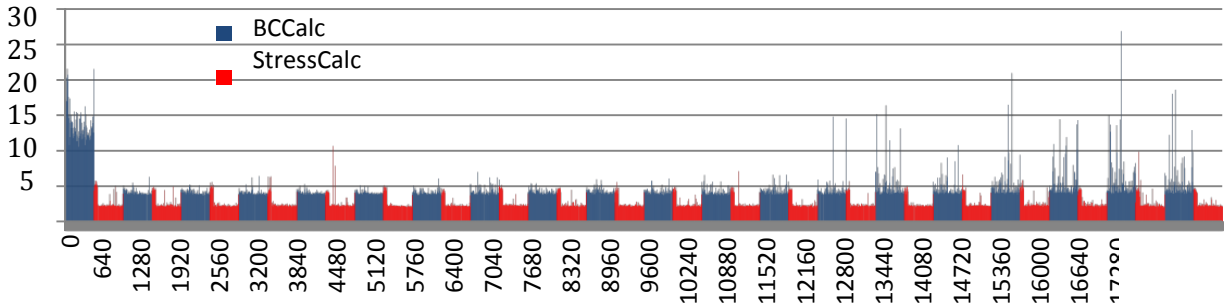


Figure 10 MDS Task Execution Time Histogram for 20 iterations

2. Deployment issues

- Changing the number of roles in a "running" deployment results in a non-responding deployment. This happened to us several times in the past and lately we avoid this by stopping the deployment before changing the number of roles.
- Deployment unreliability. Some roles never come to running state or take a very long time to start. This became a major issue for us when running the performance benchmarks. For an example, typically we had to start 132 or more roles to get 128 working roles.

3.API issues

- More user friendly error messages for the service requests. Most of the error messages we get from Azure services are generic "invalid request" messages, which do not provide information about what really went wrong in the request. This leads to lot of wasted time with trial and error type of development.
- Better coherent documentation

4. Feature requests

- Visibility time limit change support for queue messages. This would allow us to support much richer fault tolerance patterns. Amazon SQS supports this through ChangeMessageVisiblity operation.
- Server-side Count or Sum operation for table entries. It's very inefficient to download all the entries in a table to perform a simple sum or a count operation for a single table field.

Acknowledgements

This work was made possible using the computing usage grant provided by Microsoft Azure Cloud. The work reported in this document is partially funded by NIH Grant number RC2HG005806-02. We would also like to appreciate SALSA group for their support.

References

[1] Judy Qiu, Jaliya Ekanayake, Thilina Gunarathne, Jong Youl Choi, Seung-Hee Bae, Hui Li, Bingjing Zhang, Yang Ryan, Saliya Ekanayake, Tak-Lon Wu, Adam Hughes, Geoffrey Fox Hybrid Cloud and

Cluster Computing Paradigms for Life Science Applications, *Journal of BMC Bioinformatics*. Open Conferences System BOSC Proceedings (BOSC 2010), VOL.11(Suppl 12): p.S3, August 18, 2010.

[2] Judy Qiu, Jaliya Ekanayake, Thilina Gunarathne, Jong Youl Choi, Seung-Hee Bae, Yang Ruan, Saliya Ekanayake, Stephen Wu, Geoffrey Fox, Mina Rho, Haixu Tang, Data Intensive Computing for Bioinformatics, a book chapter of *Data Intensive Distributed Computing*, ISBN13:978-1-61520-971-2, IGI Publishers, 2012.

[3] T. Gunarathne, B. Zhang, T.-L. Wu, and J. Qiu, "Portable Parallel Programming on Cloud and HPC: Scientific Applications of Twister4Azure," presented at the Portable Parallel Programming on Cloud and HPC: Scientific Applications of Twister4Azure, Melbourne, Australia, 2011.

[4] T. Gunarathne, W. Tak-Lon, J. Qiu, and G. Fox, "MapReduce in the Clouds for Science," in *Cloud Computing*
Technology and Science (CloudCom), 2010 IEEE Second International Conference on, 2010, pp. 565-572.

[5] T. Gunarathne, T.-L. Wu, J. Y. Choi, S.-H. Bae, and J. Qiu, "Cloud computing paradigms for pleasingly parallel biomedical applica.

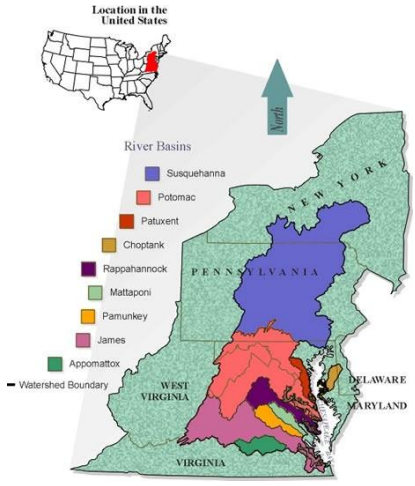
USING THE CLOUD TO MODEL AND MANAGE LARGE WATERSHED SYSTEMS

Jon Goodall, University of South Carolina, and Marty Humphrey, University of Virginia

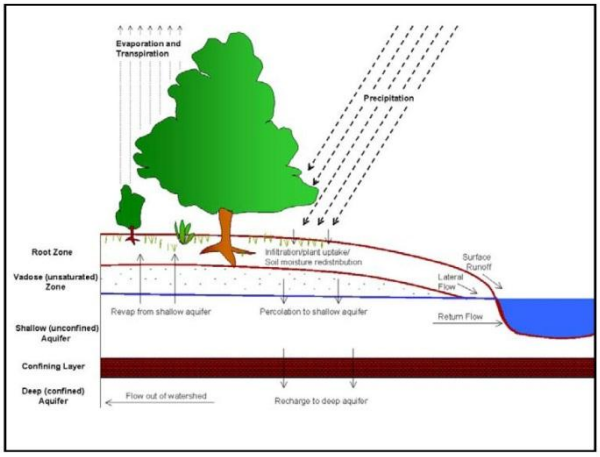
Understanding hydrologic systems at the scale of large watersheds is critically important to society when faced with extreme events, such as floods and droughts, or with concern about water quality. Climate change and increasing population are further complicating watershed-scale prediction by placing additional stress and uncertainty on future hydrologic system conditions. This project advances hydrologic science and water resource management by creating and using a cloud-enabled hydrologic model and data processing workflows to examine the Savannah River Basin in the Southeastern United States. This will provide the detail and scale necessary to address fundamental research questions related to quantifying impacts of climate change on water resources.

Scientific Challenge

Computational tools are critical to hydrologic system analysis and simulation. However, today, models that are used for engineering analysis of hydrologic systems in particular are insufficient for addressing current water resource challenges such as the impact of land use change and climate change on water resources. The challenge that we face extends beyond modeling and includes the entire workflow from data collection to decision making.



Chesapeake Bay Watershed



Watershed Hydrology Model (source: SWAT theoretical documentation)

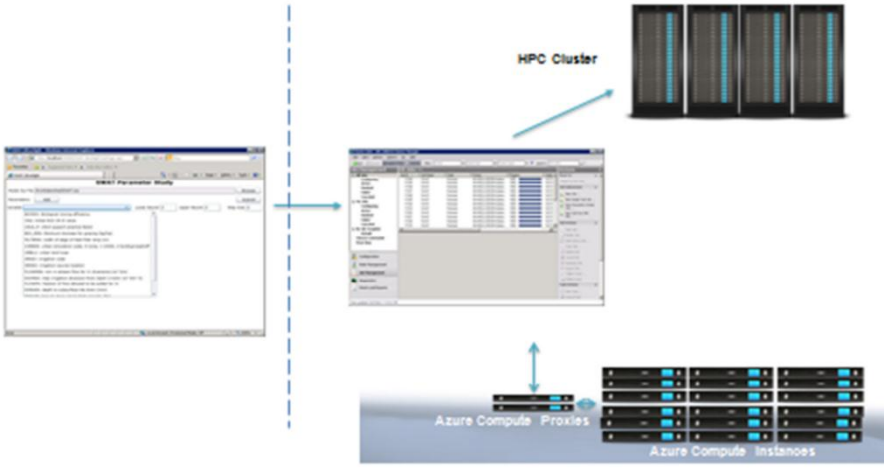
The first challenge is data preparation, which is arguably the most tedious, error-prone and time-consuming step in the overall process of applying a hydrologic model. The second challenge is the ability for the hydrologic researcher to find and manage the necessary computational resources to execute the model – this includes the time necessary to "calibrate" the model against the known physical observations prior to large-scale execution and simulation. A third challenge is the design, creation, and execution of sophisticated models that are necessary to answer policy questions of increasing complexity. For example, a few decades ago it was sufficient to model only point source discharges of pollution to water bodies under water quality regulations. Now it is required to also model the impact from nonpoint sources, i.e. runoff from agricultural and urban lands to waterbodies. This seemingly subtle difference has significant implications in the complexity of the model needed to address policy questions.

We will use Windows Azure in three ways. First, we will create a cloud-enabled hydrologic model. Second, we will improve the process of hydrologic model parameterization by creating cloud-based data processing workflows. Third, in Windows Azure, we will apply the model and data processing tool to a large watershed in order to address a relevant hydrologic research question related to quantifying impacts of climate change on water resources.

We choose Windows Azure instead of a supercomputer-based approach primarily for two reasons. First, not surprisingly, data processing workflows more naturally fit with a cloud model – whereby data can be pulled on-demand from remote sources (perhaps already in the cloud), transformed, and then stored (at least temporarily) in the cloud for use in other computations. Second, to provide a community-based resource, we are creating a Web-based portal for the watershed modeling community, and we anticipate that usage of this portal will be sporadic, which maps well to the cloud's ability to ramp up and ramp down dynamically.

Implementation on Windows Azure

We plan to use Windows Azure for data preparation, model calibration, and large-scale model execution. To date, we have focused on model calibration, whose goal is to search the space of potential model parameters for the best match against observed data. As shown below, the user uploads her model (we are currently supporting the SWAT modeling system) and then specifies the set of parameters and range of values for each parameter, effectively defining the search space. After specifying all the parameters and ranges, our system then uses the DDS algorithm for watershed calibration, whereby essentially a collection of independent executions of the model are performed in parallel, each with a different set of parameter values. When all of these complete, the executions are evaluated against the observed USDA data. The best execution is chosen as the base for the next wave of parallel executions. Currently, the number of model executions in a single wave is a function of the number of cores available (1 per core). These waves continue for 1000 total executions of the model, at which point the best parameter set is presented to the user. The user can then continue with more calibration if necessary.



SWAT Model Calibration using Windows Azure

We highly leverage MODISAzure, which is our previous implementation on Windows Azure for large-scale satellite image processing. Our particular architecture is based on cloudbursting, which uses the Microsoft HPC support extensively. As shown above, when a user submits a job, the first resources sought are the Microsoft HPC Cluster we have at the University of Virginia. When there is too much work in the queue, we "burst" onto Windows Azure. This architecture has shown to be very flexible, as we are able to spin up new compute nodes in Windows Azure (and shut them down) whenever we desire. Within Windows Azure, we use Blob storage to prepare the nodes – i.e., when they boot, they automatically load our generic watershed modeling code – but, because each execution of the model is largely independent, we do not use Blob storage after the VMs have booted (and rather rely on local storage within the VMs). We use the Windows Azure Connect CTP to provide VPN capability between the head node inside the University of Virginia and our Windows Azure compute nodes.

Current Status

We have recently successfully implemented the watershed modeling calibration routines and are now analyzing performance to determine the bottlenecks and potential improvements. Using 8 Medium Windows Azure nodes and no local nodes, we are seeing approximately a 4x speedup as compared to desktop calibration. This is reasonable, but we believe that there are additional performance gains possible.

Currently, our biggest challenges remain application development and debugging, the reliability of Windows Azure itself, and performance analysis. For the latter in particular, there is large variance in duration for each execution of the watershed model, and it is difficult to determine if it's domain-specific variance, networking variance (from Windows Azure to the University of Virginia and back) , or variance within Windows Azure itself. Going forward, we believe that re-architecting the

watershed modeling code to effectively scale-out will be a challenge – e.g., how to run a single watershed model simulation across multiple Windows Azure nodes.

Finally, we continue to be challenged and intrigued by the financial aspects of the cloud. While we would certainly request cloud resources as part of future requests to funding agencies (instead of purchasing machines to be housed locally), the precise economic comparison is difficult to determine. That is, we are very happy with the ability to rent Windows Azure cloud resources on demand, and they're reasonable to manage right now (and only getting easier); however, we cannot say with certainty if there is a 4x, a 2x, or even only a slight financial benefit of renting resources as compared to buying. We are attempting perform this analysis, and we are also looking to the broader community for answers on this issue.

NETWORK3D ON WINDOWS AZURE: WEB PORTAL FOR ECOLOGICAL NETWORK SIMULATIONS & ANALYSIS

Jennifer Dunne[1,2], Sanghyuk Yoon[1], Neo Martinez[1]
[1]Pacific Ecoinformatics and Computational Ecology Lab (Berkeley, CA); [2]Santa Fe Institute (Santa Fe, NM)

A Scientific Challenge: Complex Ecological Network Research

A grand challenge in the science of ecology is explaining and predicting the behavior of complex ecosystems comprised of many interacting species. The long-term scientific goal is the development of theory that accurately predicts the response of different species and whole ecosystems to physical, biological and chemical changes. Our group formalizes aspects of such theory using nonlinear, coupled ordinary differential equations that characterize the bioenergetic feeding dynamics of complex networks of interdependent species. Models comprised of these equations demand substantial computational resources to parameterize, run, and analyze large numbers of model simulations. Such simulations provide the data necessary to develop basic theory of how species co-exist and persist and what stabilizes and destabilizes ecosystems. The simulations also support more applied investigations of the consequences of species loss, species invasion, and environmental change on ecosystem structure, dynamics, and function. Over the last two decades, ecological network theory developed by our group and others has provided important insights about a variety of aspects of these issues, and has motivated a significant expansion in research on complex ecological networks.

Our group's first major accomplishment was the discovery of a more accurate and detailed understanding of how food webs are organized. These networks of species and their feeding relationships are of central importance for the co-existence of species, their dynamics at ecological and evolutionary time scales, and ecosystem function. Improved understanding of the structure of consumer- resource networks provides a foundation for a second accomplishment, the development of theory based on integrated network structure and non-linear dynamical modeling that describes how complex ecological networks persist over various times scales. This theory, in turn, supports the third accomplishment, understanding and predicting the consequences of species loss, invasion, and environmental change on ecosystems as mediated by complex species interactions. One example is that we can successfully predict how the loss of one species will alter the abundance of other species within an ecosystem dominated by feeding dynamics. In addition to refining our basic theory about ecological network structure, dynamics, and persistence, we are using an intensive computational framework to pursue further important issues including habitat-specific modeling (e.g., coral reefs, lakes, forests, etc.), modeling of human subsistence and economic roles and impacts within ecosystems, and modeling the impact of species loss and invasions on ecosystem structure and function. These extensions of our research agenda require a significant expansion of our computational capabilities, particularly as related to mathematical modeling, data management, and visualization.

Implementation

Due to the limited capabilities (e.g., for network visualization), required programming expertise, and relatively slow computational abilities of programs such as MATLAB, our group developed research software called Network3D (with funding from NSF and Microsoft Research) to visualize, analyze, and model the structure and dynamics of populations, communities, and ecosystems. Network3D makes it much easier and faster to analyze, visualize, and simulate complex ecological network structure and dynamics, allowing for simulations that run approximately 10x faster than MATLAB. The results of simulations are easily compared with empirical data within Network3D to evaluate the predictive power of a given mathematical model. We have made the software freely available to the research community and it has been used extensively by other research groups.

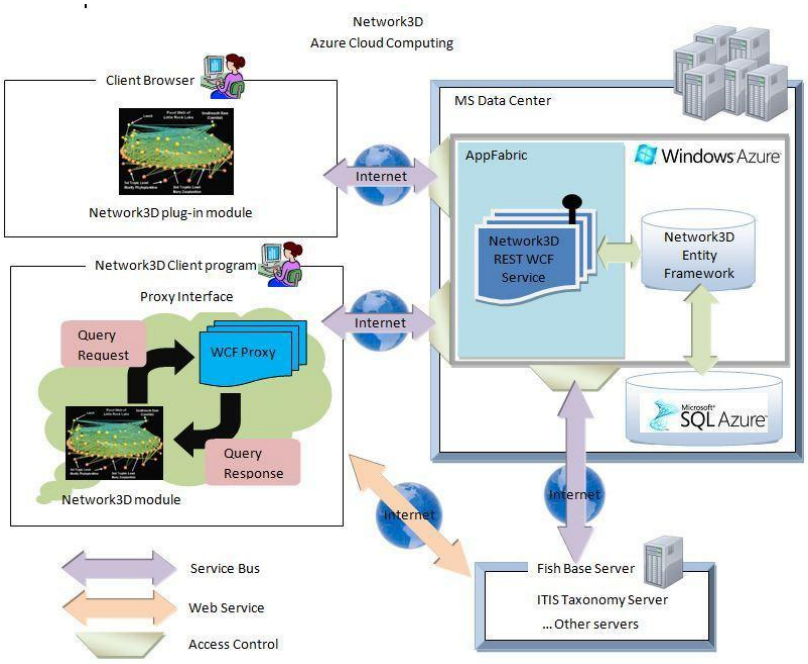
Network3D is written in C# and runs on the Microsoft Windows platform. Network3D enables easy parameterization and running of well-known ecological models of network structure such as our "Niche Model" (published originally in Williams & Martinez 2000 *Nature*) which generates realistic food web structure. Network3D also supports simulation and manipulation of ecological network dynamics as characterized by our "Allometric Trophic Network Model" (as described in Berlow et al. 2009 *Proceeding of the National Academy of Sciences USA*). Any given *in silico* experiment involving population dynamics can be very computationally intensive. Each step of an experiment requires repeated calculation of many different complex aspects of the ecologically dynamical system, such as functional response, and includes hundreds of thousands of time steps. Given the great variability in natural systems, researchers need to conduct many manipulations to discover what are the more and less predictable behaviors and responses of complex ecosystems to particular experimental factors. Of equal importance to the basic ability to run such *in silico* experiments is the ability to store and query the vast volume of data that result from such experiments. Relevant data include things such as the time series of species biomasses, changes in interaction structure, and the amount of energy flowing through the links in the networks.

By porting Network3D to the Windows Azure cloud computing platform, we have been able to decrease the average processing time from one day on a local computer to an hour given 48 4-core instances on Windows Azure. Research projects that typically require 2 weeks of simulation time on a Linux cluster with 50 nodes only take a few hours with Windows Azure. More importantly, the results are stored in a SQL Azure database, making them much easier to analyze by anyone with access to the database. An advantage of this solution over a traditional supercomputer batch system is that it is a web portal that is accessed over the internet and provides web services. Data generated by each user can be accessed and shared with community users.

"Network3D on Azure" is a parallel engine running on Windows Azure that can harness the computing power of Azure instances (see Figure at top of page 3). The Network3D engine uses Windows Workflow Foundation to implement long-running processes as workflows. Given requests which contain a number of manipulations, each manipulation is delivered to a worker instance to execute. The result of each manipulation is saved to SQL Azure. A web role provides the user with an interface where the user can initiate, monitor and manage their manipulations as wells as web

services for other sites and visualization clients. Once the request is submitted through the web role inter- face, the manipulation workflow starts the task and a manipulation is assigned to an available worker to process. The Network3D visualization client communicates through web services and visualizes the ecological network and population dynamics results. Currently, we are testing and using Net- work3D web services for a social ecological game engine called 'BeastReality'. This game allows social game users to participate in stabilizing the dynamics of ecological networks using various possible combinations of system parameters, potentially leading to novel outcomes to modeling.

Effectiveness



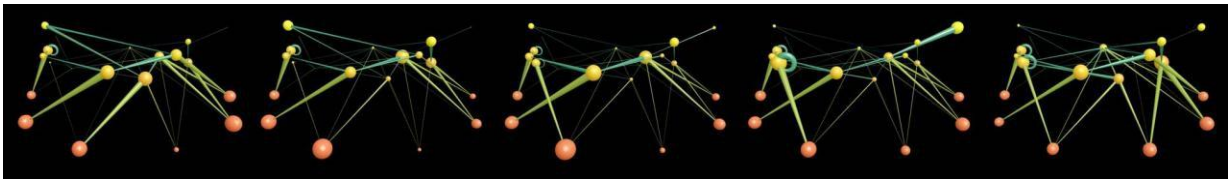
We have yet to thoroughly evaluate the effectiveness of our implementation. It is clear that users with less computational expertise are more able to conduct computational research using our portal. However, it is not clear whether the amount of time developing and maintaining the portal is worth the additional functionality it provides. At this point, its main competition is MATLAB running on Amazon Cloud Computing. This latter platform might be less expensive and frees us of much development time. However, its database and other Network3D specific functionality is lacking. A key issue is up-front development costs.

The Azure platform may provide the best platform for conducting our research but results are significantly delayed by initial development time. Such delays in output may interfere with continued NSF funding for our research. Therefore we may have to reduce our commitment to developing our Azure portal in order to get more immediate research results that will help secure additional re- search funding. As far as cost, our research results and especially the improved accessibility of these results on the Azure platform may well justify including the 100-200K costs for

Azure resources in our research proposals. However, such resource requests need to be balanced against alternative research methods. We will be better able to evaluate such issues within the next six months.

The other exciting thing we are evaluating is the implementation of our social network game 'BeastReality' which uses Network3D on Azure as the backend for playing the game. We are investigating funding sources to develop a business model to provide for Azure costs during the conduct of the game.

Below is a series of images captured from a dynamic Network3D visualization representing a time series of the changes of 18 species' biomasses represented by the size of the nodes and feeding rates represented by the diameter of the cone at the fat end. Our research currently focuses on 30-50 species networks.



DEVELOPING THE FORECAST-AS- A-SERVICE (FAAS) FRAMEWORK FOR RENEWABLE ENERGY SOURCES

Kwa-Sur Tam, Virginia Tech

Accurate forecast is key to effective utilization of weather-dependent renewable energy sources such as wind and solar. Weather forecasting is a complex and data-intensive computing process. To generate wind power forecasts at specific locations, additional data such as orography, land surface condition, wind turbine characteristics, etc., need to be obtained from multiple sources. In addition to the diversity of the types of data and the sources of data, there are different forecasting models that have been developed using different approaches. The goal of this project is to develop a framework – the Forecast-as-a-service (FaaS) framework – that achieves two purposes.

1. Enable the combined use of different types of data from different sources and enhance the synthesis of more accurate forecasts using prediction results from different models.

2. Support on-demand delivery of forecasts of different data types and at different levels of detail for different prices. Widespread utilization of renewable energy can be enhanced by making forecast information available to current or potential renewable energy users with different needs and different budgets.

Figure 1 shows the FaaS framework that can be implemented by using the Azure platform to provide on-demand wind power forecasting services at a customer-specified location for different types of customers. A similar FaaS framework can be developed for solar power forecasting services.

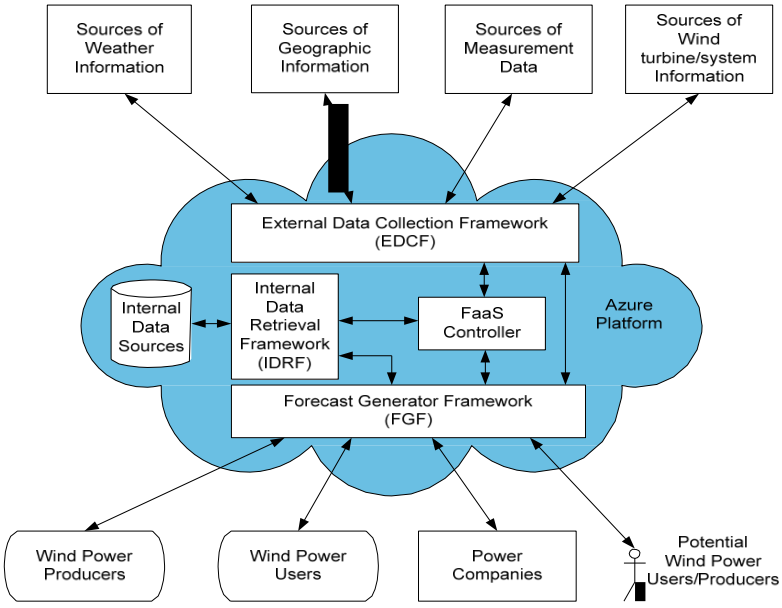


Figure 1. FaaS framework for wind power

This project could have broad impact because the FaaS framework could contribute to the formation of a national forecast cyber-infrastructure to provide technical support to many sectors of the economy. The FaaS framework could contribute to meeting two national needs – energy independence and environmental stewardship. It could help power companies in many states to meet their respective state mandates on the use of renewable energy.

Figure 2 shows the architecture of the FaaS framework that is currently being developed. The FaaS framework consists of the Forecast Generation Framework (FGF), the Internal Data Retrieval Framework (IDRF), the External Data Collection Framework (EDCF) and the FaaS controller. The FGF, IDRF and EDCF all adopt service-oriented architecture (SOA) and the activities of these three frameworks are orchestrated by the FaaS controller. In additional to advancing the understanding of how a network of SOA-based systems will operate, the forecast services/composite services developed can also be useful in other applications that require forecasting (such as economics and transportation).

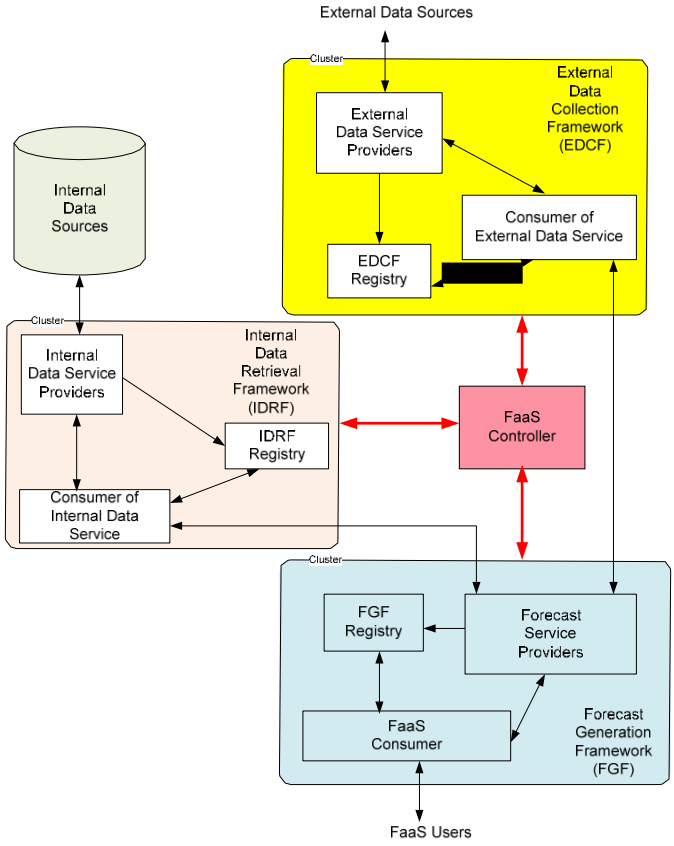


Figure 2. SOA-based architecture for the FaaS framework

Since Windows Azure and .NET technologies support service-oriented architecture and its design principles, this project can focus on achieving its goals rather than dealing with the underlying support infrastructure. The development environment for Windows Azure is fully integrated into Visual Studio, which provides a simulated runtime for Windows Azure for local desktop-based

development and unit testing. This is especially important for this project because students at Virginia Tech are generally familiar with Visual Studio.

As far as research funding is concerned, cloud computing is generally preferred over on-premise hardware because of costs and flexibility. Cloud computing services such as Microsoft Azure is also preferred over supercomputers because generally they are more accessible and they provide better technical support and training.

This two-year project is currently in its first year so many aspects are still in the development stage. We would like to reserve our opinion on the challenges in working with Azure until we finish the project.

XCLOUD: SEAMLESS INTEGRATION OF MULTIPLE CLOUDS

Hakim Weatherpoon, Assistant Professor, Cornell University

*[1. **Describe the scientific goals of your research project**. What is the technical challenge and what will be the impact of the results. This need only be a one or two paragraph statement. Include a nice picture if you have one handy.]*

Cloud computing is often compared to other utility models, like electricity. Similar to power suppliers, cloud providers offer massive amounts of computing resources. Unlike the power grid, these resources are tightly coupled to a provider's infrastructure. For example, the infrastructure and standards that the Microsoft Azure provides are different than Cornell's Red Cloud (http://www.cac.cornell.edu/redcloud). The cloud user (not to be confused with the end user) must be aware, for example, that his/her application is running in Azure or Red Cloud. This tight coupling is problematic as it constantly exposes the underlying provider and the corresponding infrastructure. In particular, at Cornell, we would like to be able to seamlessly use cloud resources provided by Cornell, the National Science Foundation, and/or Microsoft Azure. As a result, we are investigating how to seamlessly migrate a live virtual machine (VM) instance from Cornell's Red Cloud based on Eucalyptus to Microsoft's Azure.

To this end, we will explore an architecture that decouples cloud users from cloud providers and eliminates the need to share a complex hypervisor. At first glance, the idea of totally decoupling the cloud user from the cloud provider might require a radical redesign of today's clouds. In fact, it does not. We have successfully demonstrated this new way of delivering cloud resources through a highly-optimized nested para-virtualization. In our approach, we are able to showcase how one can re-architect today's cloud such that cloud production (i.e., the actual provider's infrastructure) and its consumption are totally decoupled. In our model, a virtual machine can be running across multiple cloud infrastructures without its owner being aware of the underlying hypervisor, the provider, the hardware, the network, or how many replicas are running. The shared, bottom-layer hypervisor can be extremely lean—requiring only support for multiplexing hardware, similar to the NoHype approach—while the second-layer hypervisor can implement functionality pertinent to the cloud application running on top of it. Second-layer hypervisors are likely to be diverse: each cloud user or application may have a different hypervisor suited to their needs.

We call this new model an xcloud (http://xcloud.cs.cornell.edu). It provides four key advantages: (1) it provide a true container that completely abstracts out the underlying infrastructure and provider, thus decoupling cloud providers from users, (2) can implement key atomic operations such as cloning, mirroring, splitting, combining, migrating—independent of the underlying hypervisor; these atomic operations can then be used as building blocks to implement high-level constructs, (3) allow the creation of resiliency and trust zones where monitoring, detection, and quarantine can be deployed without modifications to the applications, OSs, or hypervisors, and (4), xclouds can be overlayed on existing clouds, allowing the creation of a cloud layer on existing technologies.

IMPLEMENTATION

*[2. **How did you approach the technical challenges using the Azure cloud**? Here we are looking for another paragraph or two that describes the programming paradigm you used. Technical details help a lot here.]*

At its core, an xcloud acts as a second hypervisor layer. It relies on a nested virtualization: running a second-layer hypervisor. The reason we focus on a second-layer hypervisor is four-fold. First, we envision xclouds to be deployed across potentially many hypervisors in multiple heterogeneous providers. Diversity in the lowest layer should be embraced, as the development of different cloud approaches appear. Second, if support from the underlying hypervisor is not explicitly required, benefits from our research can be achieved nearer in the future: it is not necessary to roll out a new hypervisors across all data centers. Third, optimizations in the lowest layer, both in terms of performance and security, can be deployed incrementally. Finally, this approach enables experimentation and development on a wide variety of real cloud systems.

Without relying on the lowest-layer hypervisor exposing hardware extensions for virtualization to guests, techniques like para-virtualization (e.g. Xen), binary translation (e.g. VMware), and emulation (e.g. QEMU) must be used. We have selected Xen as the second-layer hypervisor because of its popularity both in the open-source community and public cloud infrastructures. Challenges in running Xen on top of existing clouds generally stem from handling diversity in the underlying interface, which tends to be found in the device I/O subsystem. This is because hypervisor specific para-virtualized device interfaces are used to achieve performance: for example, the interfaces for Xen and KVM differ.

To handle diversity, we propose a Cloud Infrastructure Abstraction Layer (CIAL) populated with a number of drivers for underlying infrastructures, called Blanket drivers. Guests running on top of the second-layer hypervisor all use the same Xen device interface to communicate with a second-layer Domain 0, containing the Blanket drivers. As a consequence, cloud provider heterogeneity from the underlying hypervisors is abstracted away by the CIAL into a homogeneous Xen environment. Therefore, any VM in an xcloud can be run on any physical machine that is running the CIAL. This enables multi-cloud migration.

RESEARCH AGENDA

*[3. Many large scientific problems that can be solved on the cloud could also be solved on a large supercomputer**. Are there any aspects of your computation or approach that were better suited to solution on the cloud**? ("I don't have access to a supercomputer" is a fair answer).]*

The xcloud is an agenda designed for the cloud. It is a cloud within a cloud: It is both a cloud provider and cloud user. It is a cloud provider since it provides a Cloud Abstraction Infrastructure Layer (CIAL) for the seamless integration of multiple clouds. And, it is also a cloud user, since the CIAL sits on top of the bottom layer cloud provider.

*[4. The median amount of cloud resource consumed by the group in this study has cost about $25,000 for 2011. **If you had a way to request this amount from your funding agency for cloud resources in the future, would you do so? Or would you prefer to request the same amount to purchase new***

hardware that you would manage yourself? *If so, would you be able to accomplish the same research results?]*

The xcloud allows us to seamlessly migrate a computation from one cloud provider to another (e.g. from Cornell's Red Cloud to Azure). As a result, requesting funding for cloud resources would be much more advantages than purchasing of new hardware. Further, we will be able to exploit local, national, and commercial cloud resources, which will allow us to optimize cost and performance.

[5. **What were the biggest challenges of working with the Azure cloud resource?]**

To run a second layer Xen instance in Azure, we require hardware virtualization (full virtualization or HVM), which is necessary to run a Windows VM guest instance. Since many cloud providers supply a HVM instances for Windows guests, we plan to make an "installer" for Xen to convert a Windows instance to a Xen instance. With this installer, we will be able to run Xen Azure, as well as many other cloud providers. Further the xcloud layer will enable a Xen instance to seamlessly run in many heterogeneous clouds such as Cornell's Red Cloud and Azure.

USE OF AZURE FOR COMPUTING IN THE CLOUD (NSF)

Douglas Thain, University of Notre Dame

Broadly speaking, my research lab at Notre Dame studies how to make large scale distributed systems such as clusters, clouds, and grids usable and cost-effective for large scale science problems. In this particular project making use of the Windows Azure platform, we took the technical step up porting our applications and frameworks to Windows, and studied the implications of moving a particular family of applications (ensemble molecular dynamics) across a range of large scale distributed systems. The outputs of this project include new software tools now used in production scientific computing, a technical study of the implications of moving from a parallel environment to a cloud environment, and some observations on our experience of using commercial cloud providers.

For our collaborators, the objective is to accomplish research in the most cost effective way possible, not to use a particular computing system. From the beginning, we take the perspective that users will have access to a variety of computing technologies, and will want to evaluate their applications for cost and performance on all available systems. Consequently, this research was not targeted towards Azure, but employed a variety of computing systems, including include traditional clusters (managed by SGE), campus grids (Condor), VM-based cloud providers (Amazon EC2) and framework-based cloud providers (Windows Azure). To enable this, we developed a custom middleware (Work Queue) that allows users to express elastic applications independently of the underlying cloud provider.

To study the implications of moving from a traditional parallel environment to a cloud environment, we focused on the application structure of *replica exchange*, which is a very widely used technique in the study of molecular dynamics. Replica exchange applications are traditionally implemented in a parallel fashion using the Message Passing Interface (MPI). In this technique, replicas of the protein system are created at different temperatures within the range of interest and many configurations are visited by these replicas. At certain points in the simulations, an exchange between the replicas is attempted to allow them to cross energy barriers. Specifically, simulations are run by creating multiple replicas of a protein molecule and executing each over several Monte Carlo steps or iterations at different temperatures. At the end of every iteration, an exchange is attempted between neighboring replicas, where if certain criteria are met, the replicas are swapped with regards to their temperature and the simulation is continued. The simulations of replicas in each iteration are completely independent and can be performed parallel to each other.

To overcome the shortcomings of an MPI based implementation of replica exchange, we built an elastic implementation using Work Queue, a framework for building cloud and distributed applications. In doing so, we also established guidelines for designing and building software frameworks used in building cloud applications. We show that adherence to these guidelines provides better scalability,

run-time resource adaptability, fault tolerance, and portability across multiple cloud platforms. This work was published in CloudCom 2011 [1].

In our scalability evaluations, we employed over 400 cloud instances spread across multiple platforms including a homogeneous cluster environment (SGE) and heterogeneous cloud environments such as Microsoft Azure. We built scripts that communicate to the web role and invoke the Work Queue workers on the worker roles. We also used Cygwin compiled executables in running our implementation on the Azure platform. We employed around 350 CPU cores in the Azure platform for these experiments and consumed over 300,000 core hours in our experimental runs so far.

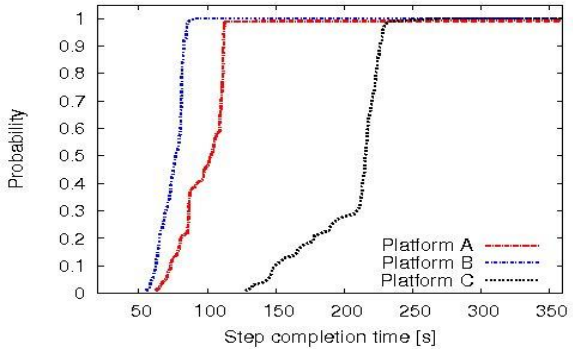


FIGURE 1 DISTRIBUTION OF TASK PERFORMANCE

As an example, Figure 1 compares the completion time of the Monte Carlo steps of each replica. Platform A represents EC2 instances (2*2 x 1.0-1.2 GHZ, 7.5 GB memory), Platform B represents instances from the Sun Grid Engine at Notre Dame (2*2 x 2.6 GHZ, 8-12 GB memory), and Platform C represents Azure instances (2 x 1.6 GHZ, 3.5 GB memory). We observe significant variations in the completion times of the Monte Carlo steps on Platform C as compared to the other two platforms. While these variations can be attributed to one or more of several factors including network latencies and jitter, virtualization effects, firewall, load balancing etc., this is good evidence that our implementation is impervious to any peculiar platform and network characteristics of a cloud computing system. Our elastic implementation thus also demonstrates the ability to hide differences in design, implementation, and behavior of different cloud computing platforms

Figure 2 captures the resource adaptability of our elastic implementation of replica exchange in our experimental evaluations. It illustrates how the run-time of each iteration in the simulation varies as resources (workers) are added and removed. The run-time decreases as resources are added except where it increases when resources from Platform C were added. This is because the running time of simulation steps on Platform C significantly varies from the other platforms as we observe in the figure above. We also observe the run-time to increase whenever resources are removed since the failed tasks are being re-scheduled and re-run on the remaining resources.

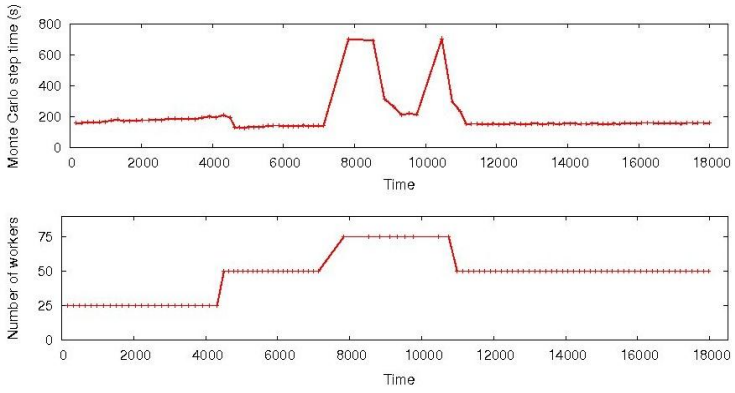


FIGURE 2: TOLERATING RESOURCE VARIATIONS

Our experimental evaluations and results were published in CloudCom 2011 [1] and in the Workshop on Python for High Performance and Scientific Computing at SC 2011 [2].

We also built an asynchronous version of the replica exchange technique that maximizes resource utilization and improves the completion time of the simulation. We are currently running evaluations on this asynchronous replica exchange technique, but early results show that these techniques are much more appropriate for the performance variations that occur in cloud computing systems.

Observations of Azure

- Easy to get started. Overall, we found the Windows Azure service quite easy to get started using for real applications. The most common usage model was to start a certain number of worker roles from the web console, use those to start our own middleware, and then communicate with those machines directly. Very little idling in the way of virtual machine images, network configuration, and so forth was necessary. The system as a whole was generally compatible with how we and our collaborators would like to be able to "cloud burst" from existing computing resources, especially when using Win32 and Cygwin based applications.

- **Variability in performance led to application redesign.** As shown in Figure 1 below, we found that Azure had higher variability in both CPU performance and network latency than a VM-based cloud system such as EC2. We suspect this is due to a high degree of time-sharing of multiple roles on a given physical resource, but we don't have a way of verifying this directly. This variability initially had a significant impact on our applications, in which there was a barrier between completion of parallel steps. This observation led us to redesign the application framework to be more asynchronous, so that variations did not have the same overall effect. This modification also gives us a performance

improvement on other platforms. So, this is not necessarily a negative property of Azure, but may be a design lesson that has broader implications.

- Need delegation of resource management. One significant downside is that there is poor cost visibility for the end user. Our mode of operation (which we suspect is not unusual) is that the manager (in this case the PI) establishes the account for a purpose, and then delegates to an employee (in this case a student) the authority to incur expenses on the account. Although most of the cycles went to good use, some were wasted when the employee left resources in the "Suspended" state. Although the manager received the occasional notice of resources used, he assumed this was due to productive use, and had no visible way to associate this with suspended resources. The employee had no direct notice, and so had no reason to clean them up. What is needed here is that a given allocation should have a max limit on resources, and prominently display the "burn rate" to the manager. When the manager gives access to another employee, the manager should be able to put a limit on what is delegated to the employee and prominently display the burn rate. In short, one use should be able to delegate a certain amount of resources to another, and have visibility into that delegation.

References:

[1] Dinesh Rajan, Anthony Canino, Jesus A Izaguirre, and Douglas Thain, *"Converting A High Performance Application to an Elastic Cloud Application", 3rd IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2011)*, November, 2011.

[2] Peter Bui, Dinesh Rajan, Badi Abdul-Wahid, Jesus Izaguirre, Douglas Thain, *Work Queue + Python: A Framework For Scalable Scientific Ensemble Applications,* Workshop on Python for High Performance and Scientific Computing at SC11, November, 2011.

ARCHITECTING DELAY-SENSITIVE AND SCALABLE CLOUD APPLICATIONS

Mohammad Hajjat, Shankar Narayanan, and Sanjay Rao, Purdue University

RESEARCH AND CLOUD TEST-BED DESCRIPTION

Cloud computing has recently gained significant attention, given its exciting potential to deliver IT services at a lower barrier to entry in terms of cost, risk, and expertise, in addition to enabling enterprises respond to spikes in load in an agile fashion. However, leveraging cloud computing imposes several challenges and a wide range of factors such as performance requirements, workload dynamics, policy constraints, cost savings, and network characteristics. In the Internet Systems Lab (ISL) at Purdue University, we focus on research problems related to cloud computing. In specific, our goal is to tackle challenges that application designers face in architecting scalable and performance-sensitive applications across cloud data-centers, while meeting their performance, cost and policy constraints.

Our research in cloud computing involves i) introducing new frameworks for systematically architecting interactive, large-scale cloud applications across multiple data-centers; and ii) finding novel techniques for dynamically splitting application transactions across data-centers for delay-sensitive cloud applications in response to dynamics in the cloud. We next explain each project and how Microsoft Azure cloud is being used in our research.

Dynamic Request Splitting for Geo-distributed Cloud Applications

While there are several benefits of cloud computing, cloud environments are highly dynamic, and variability in performance of cloud components and networks is inevitable [4]. Thus, a key challenge for enterprises is meeting service level agreements (SLAs) associated with their applications.

In this project[1], we propose a system that handles transient cloud dynamics (tens of seconds up to few minutes). Our system exploits the fact that the response times of the same component in different data-centers are often uncorrelated; creating a potential latency savings if work related to a poorly performing component is dynamically reassigned to a replica in a remote data center. We leverage this insight to build a system that we term Dealer which for each component dynamically splits transactions among its replicas in different data-centers. Dealer seeks to minimize user response times by taking components performance, intra-datacenter and inter-data-center communication latencies into account.

[1] Paper is under review; however, a technical report can be found here: http://docs.lib.purdue.edu/ecetr/416/

Existing schemes such as invoking more resources, or using server-side or DNS redirection are inefficient in dealing with transient cloud dynamics. Invoking more instances may not solve all problems (eg, networking or storage problems), and may take up to tens of minutes in commercial cloud platforms. Further, Dealer reassigns transactions to alternate data-centers at the granularity of individual components, rather than redirecting entire requests. In large applications with potentially hundreds of components (where only a few components might be temporarily impacted), it is possible that not all cloud components in the remote data-center are sufficiently over provisioned to handle the redirected requests.

In evaluating the importance of our system to cloud applications, we make use of Azure cloud by deploying real multi tier applications in the cloud, across multiple data centers. Figure 1 shows the overview of the system for an application consisting of components C1 to Cl, geo-distributed across d data-centers (DC1, …, DCd). We subject the applications to workloads that are generated from test clients spread across PlanetLab network (U1 to Un). Our system keeps monitoring application components performance and links delays with time at each data center. The algorithm re-evaluates how requests should be routed across the different components of the system in response to cloud dynamics (components performance problems, workload surges, network congestions, etc). It then pushes that information to application components across data-centers to instruct them how to route requests at each application component.

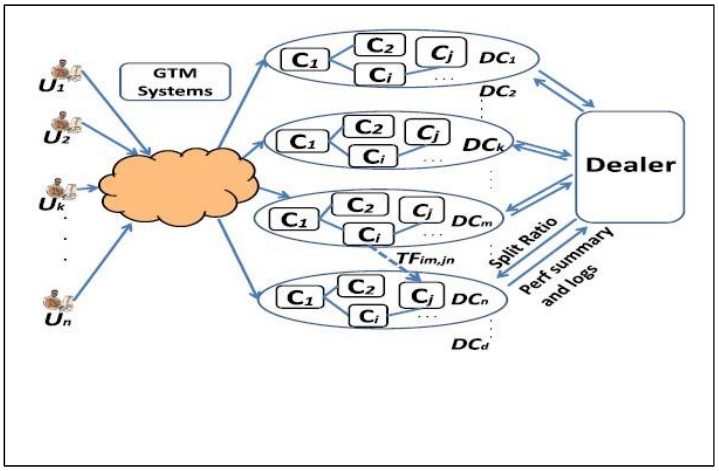


Figure 1 System overview

Architecting Interactive Large-scale Multi-tier Applications for the Cloud

Cloud applications typically consist of several functional components (up to hundreds sometimes [2]), with each component consisting of arbitrarily large number of instances (VMs). The interactions between components are often complex [2]. Further, applications have stringent response time requirements and significant variations in users' response time due to bad application design and cloud dynamics can turn potential customers away, and lead to loss of revenue.

In [2], we study how to systematically achieve the best placement of application components across multiple cloud data-centers. We build a model that takes into account enterprise-specific constraints (for eg, DB should not be migrated to another data-center), cost savings of placement, and changes on transaction and wide-area communication delays. Our model also considers the data flows between application components, and the spread of users (e.g., if they are located within or outside enterprise premises). We show the need for and benefits of a planned approach to deciding how application components should be placed across data- centers and the maximum benefits achieved.

We use Azure cloud to evaluate our algorithms. The algorithm monitors the applications to collect model parameters such as average transaction sizes, components service times and communication delays between the components. It then deploys the application across multiple data centers in a configuration recommended by our planned migration approach to evaluate the change in response time. The results help us experimentally validate the effectiveness of our model in meeting constraints on changes in application response time.

CHALLENGES IN USING AZURE

Most of the challenges dealing with Azure were around instrumentation, the need for more heterogeneous resources with better gauging, and applications migration.

- Instrumentation: the Diagnostics API lacked significant performance metrics we were interested in capturing. Since our algorithms and models need to measure components processing times and communication delays, we had to manually instrument our applications by modifying their code to measure these metrics. We had to use interpolation in many cases since measuring delays directly was not feasible; for eg, measuring queuing delays (the time a message spent inside a queue before being fetched by a worker role), processing times (eg, finding the time needed to process a query in SQL Azure database, or to store a blob) or finding the bandwidth and transmission latency between different Azure data-centers. Having more performance metrics would make building delay sensitive and scalable applications easier on Azure.

- Resource heterogeneity access: we work with a representative set of cloud applications that need a variety of cloud resources. While cloud resources keep continually evolving and new resources emerge, only minimal resources are made available without cost for the National Science Foundation (NSF) project. We need to extensively experiment with new services such as SQL Azure, Caching, Service Bus, CDN, Access Control, etc. Currently, all these resources are not covered through the NSF project and we pay for our usage ourselves from other sources.

- Resource usage: due to the nature of our research experiments, we typically need high peak usage for shorter periods, rather than reserving resources continually for a long period. While we were provided with the ability to run a higher number of cores for short periods, it is not clear what the exact billing model is and how it affects our quota. A simple interface that shows the percentage usage of our quota should help us gauge our usage.

- Migration: Migrating on-premise applications to the cloud is not straightforward and involves significant code re-writing to make applications compatible with Azure environment. Further, existing cloud applications written with older SDK's can be incompatible with newer versions of the SDK; creating the need for a continuous code re- writing effort.

References

[1] R. Torres, M. Hajjat, S. Rao, M. Mellia, M. Munafo "Inferring Undesirable Behavior from P2P Traffic Analysis," In ACM SIGMETRICS'09, Seattle, WA, USA, June 2009.

[2] M. Hajjat, X. Sun, Y. Sung, D. Maltz, S. Rao, K. Sripanidkulchai, and M. Tawarmalani "Cloudward Bound: Planning for Beneficial Migration of Enterprise Applications to the Cloud", SIGCOMM 2010, New Delhi, India.

[3] M. Lee, M. Hajjat, R. Kompella, and S. Rao, "RelSamp: Preserving Application Structure in Sampled Flow Measurements", INFOCOM 2011.

[4] A Li, X Yang, S Kandula, and M. Zhang, "CloudCmp: comparing public cloud providers", IMC '10.

LARGE SCALE ANNOTATION OF GENE TRANSCRIPTION REGULATORY SEQUENCES IN BACTERIAL GENOMES USING CLOUD COMPUTING

Zhengchang Su, Srinivas Arkela and Youjie Zhou
Department of Bioinformatics & Genomics and Computer Science
The University of North Carolina at Charlotte

Our goal is to annotate regulatory sequences in sequenced bacterial genomes using comparative genomics-based algorithms. Regulatory sequences specify when, how much, and where the genes should be expressed in the cell through their interactions with proteins called transcription factors (TFs). Therefore, identifying these sequences, also called TF binding sites (TFBS), in a genome is as important as identifying gene-coding sequences for understanding the biology of the cell and the development of applications in renewable energy production and environment protection as well as the prevention of the diseases they cause.

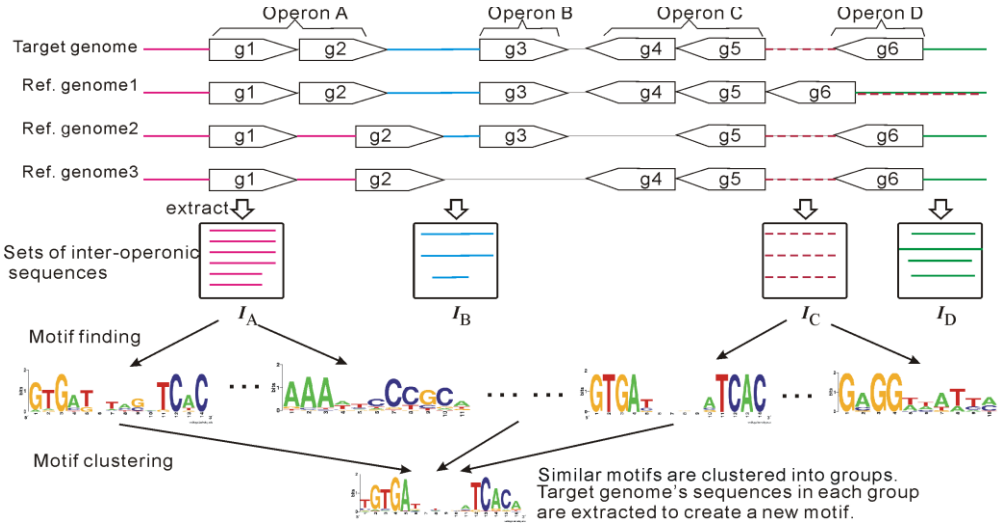


Figure 1. Prediction of transcription factor binding motifs using a comparative genomics approach.

Although comparative genomics-based algorithms allow us to predict TFBSs using only genome sequence information, these algorithms are computationally demanding, in particular, when we are now dealing with the ever increasing number of sequenced bacterial genomes, owing to the rapid advances in massively parallel sequencing technologies that are dramatically reducing the time and cost of sequencing a genome. As of today over 1,800 bacterial genomes have been sequenced and this number is rising exponentially with time. Annotating TFBSs in thousands or even tens of thousands of genomes requires enormous storage space and CPU times, which can overwhelm a typical institutional computer cluster. Cloud computing can be a good solution to the problem.

In our project, we are testing two different kinds of programming paradigms to solve our problem. First, we implemented a basic programming model using Azure model. Web roles are used as interfaces

between the system and the users. After jobs are submitted by the users, web roles build job messages and send them through Azure's queues. Worker roles automatically pull messages from the queues and perform real tasks. Second, we are testing the performance of Map-Reduce framework on Azure. Currently, we already have an implementation on Hadoop framework. So we are now planning to first deploy Daytona (an iterative Map-Reduce framework implemented by Microsoft) on Azure, and then re-implement our existing Map-Reduce codes on Azure.

Like many other large scientific problems, our problem could also be solved on a large supercomputer, however, we currently do not have access to a larger supercomputer. Furthermore, a supercomputer can be used by hundreds of users, and sometimes the priority to run large problems can be low, meaning a long waiting time. In addition, based on our experiences of programming on Hadoop, MPI and Azure frameworks, it is much easier to develop our solution on Azure. Although Hadoop or MPI provides plenty of APIs to encapsulate network operations and job scheduling, those APIs are not uniform and reliable compared with Azure's very high level of abstraction. Moreover, Azure also provides an efficient interaction framework (web roles) with web users, which large supercomputer solutions are hard to achieve. Without easy-to-use interfaces for users, the system can only be used by experts who know implementation details. Especially for our project, we intend to provide this system to biological researchers who may not be familiar with programming implementations. Compared with supercomputer solutions, Azure is more suitable for our project.

Azure also seems to be more cost-effective as the median amount of cloud resource consumed by a group is about $25,000 for 2011. If we had a way to request this amount from our funding agency for cloud resources in the future, we will very glad to do that, because in a foreseeable future with $25,000 hardware, we cannot conduct the scale of computation that we are currently doing.

However, we do face challenges of working with the Azure cloud resource. In particular, in our computational pipeline, we rely on several third-party programs; and we have difficulty to port some of them on the windows platform, so we have to seek alternative solutions.

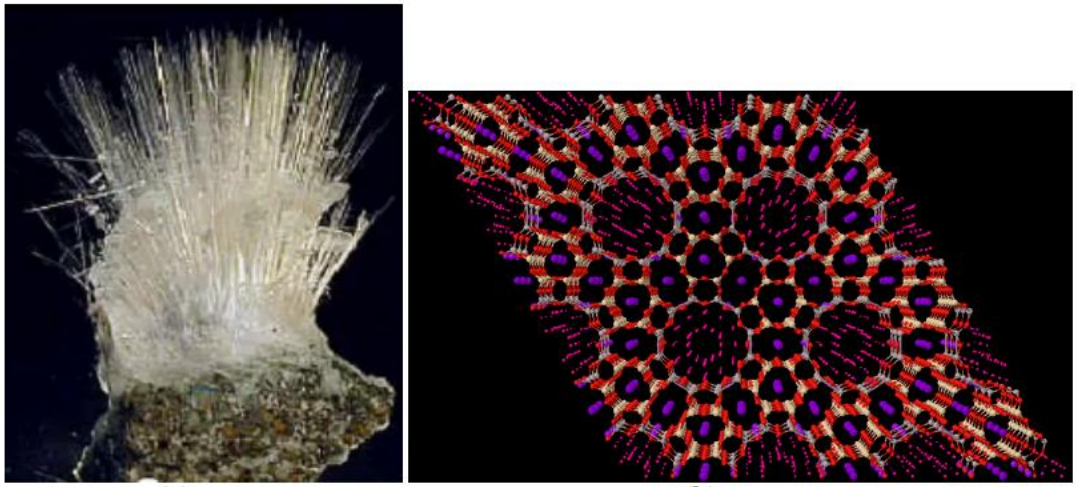
SAMP-COMPUTING IN THE CLOUD USING THE WINDOWS AZURE PLATFORM

Estela Blaisten-Barojas, George Mason University

1. Background

The NSF Structure-Adaptive Materials Prediction (SAMP) project was funded from the CHE-CRIF-Cyberinfrastructure solicitation. Within this project we have developed prior to the CIC supplement a novel machine learning approach for analyzing the structure of zeolite crystals [1-9]. Zeolites are complex crystals that present a variety of pore networks (Fig. 1). The characteristics of these pore networks define the structural features of zeolites. Along our project we have created several in-house software modules for the various machine learning tasks and data pre- and post-processing.

In the CIC supplement we proposed to further develop web-based software for the cloud environment and create appropriate interfaces with the goal of making the execution of our software modules available to a larger community. The cloud environment on the Windows Azure platform is a good choice for this supplement project because the majority of the chemistry and crystallography community that uses the Inorganic Crystal Structure Database (ICSD) [10] where the zeolite data reside works on PC based-Windows environments.



(a) **(b)**

Figure 1. Zeolites: (a) photo of a real zeolite (a frame from the movie at http://www.cmasc.gmu.edu/samp/movie.html); (b) atomic rendering of the zeolite supercell (http://cmasc.gmu.edu/sampproject.html).

2. Schematic representation of the SAMP project in Windows Azure.

The goal of the cloud computation is to provide a faster inspection of the structure of zeolites and other crystals through machine learning. One objective is the development of Web-based computing access for automating the calculation of our crystal descriptors of zeolites. A second objective is to automate the machine learning analysis for classification and clustering based on structural characteristics of the analyzed crystal data.

With these objectives in mind we have developed a number of *services*: supercell, descriptor, ZSP (zeolite structural predictor) [1-9], visualization, and ATMC (adaptive tempering Monte Carlo) [11-15]. The latter is under construction. Figure 2 gives a schematic summary of the work in process [16]. In yellow boxes we are representing the major software that we had to make available in the cloud. Services supercell and descriptor need in the cloud environment the Python compiler plus libraries, the dynamic links and runtime libraries of Fortran, and the runtime libraries of Java in order to run our in-house codes. Service ZSP requires running the Random Forest™ algorithm for classification (supervised learning) and for that we employ the open source package WEKA [17]. Service visualization uses the open source Jmol modeler. The ATMC will require use of a combination of Fortran compiler, WEKA and Jmol.

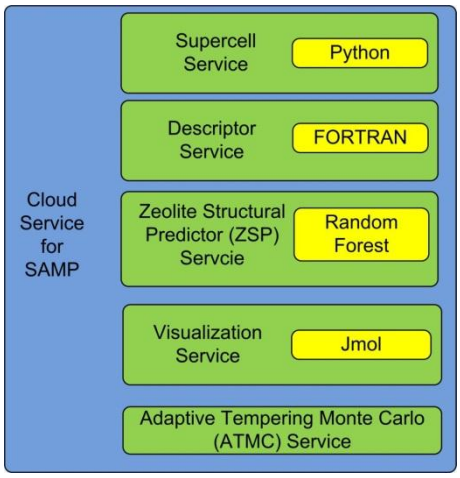


Fig. 2. Summary of the cloud services available through the SAMP project.

3. SAMP Web Role, Worker Roles and Storage in the Windows Azure platform.

The implementation of the services set summarized in Fig. 2 implies the definition of task layers in Window Azure. The first layer is composed of our in-house codes for the calculation of crystal supercells based on data entries from the ICSD. Because this database is proprietary, future users will need to upload their own data in a flat-file ascii format. Windows Azure builds upon three components: a *compute service* that runs applications, a *storage service*, and a *fabric* that supports the compute and storage services. Developers must create a Windows application consisting of Web Role instances and Worker Role instances to use the compute service. C# and ASP.Net are used for this purpose.

A Web Role serves the purpose of a Web application; a user interacting via a web page instructs the system to process his/her desired tasks. On the other hand, a Worker Role is the equivalent of a Windows service that runs continuously. Worker Role instances run fabric to implement execution of in- house codes, launch hosted executable applications, write files, and store results.

Our SAMP project uses one Web Role as an ASP.NET web application that interacts with the user and two Worker Roles, for backbone processing: the WR_server and the WR_client. Our processes are based on using our in-house codes. Three layers of processes are required: i) launching codes and executable files to process data from the ICSD, ii) running step (i) serves to create the SAMP services

(supercell, descriptor, ZSP, visualization), and iii) using 3rd party runtime and dynamic linking libraries. A schematic representation of the processing is given in Fig. 3.

The tandem Web/Worker Roles works as the follows. The Web Role stores the zeolite entries provided by the user, sends them along with messages to Windows Azure Storage (WAS) and delivers them to the Worker Roles for background processing (Fig. 4, purple arrows). The WAS is composed of Blob, Table and Queue. The Worker Roles receive different input from the Web Role. The WR_server organizes the processing, hosts and shares Fig. 3. Schematic representation of Worker Xdrive. The WR_client executes the tasks creating 18 Worker Role instances. When tasks are executed, the WR_client interacts with Xdrive to store results and update the status in the WAS. (Fig. 4, green arrows).

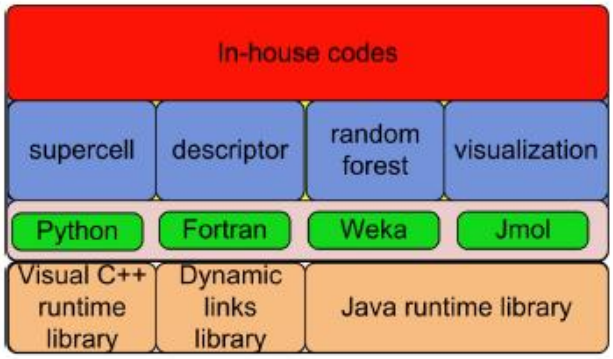


Fig. 3. Schematic representation of Worker Role instances for backbone processing.

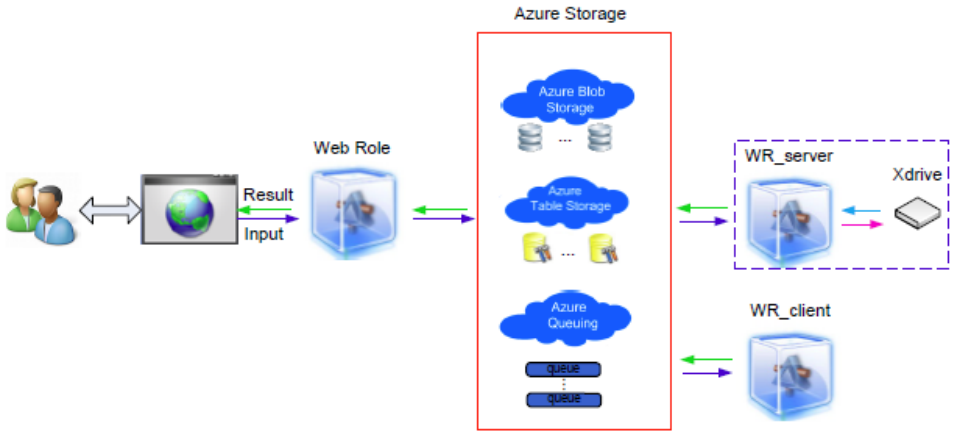


Fig 4. Cartoon of the overall workflow of the SAMP services in the cloud

When the user submits one zeolite entry file, the SAMP Web Role sends a data upload message to a file upload queue. An ICSD number identifies each zeolite entry. However, the user may also submit a merged file containing a set of zeolite entries. Our SAMP Web Role differentiates between these two

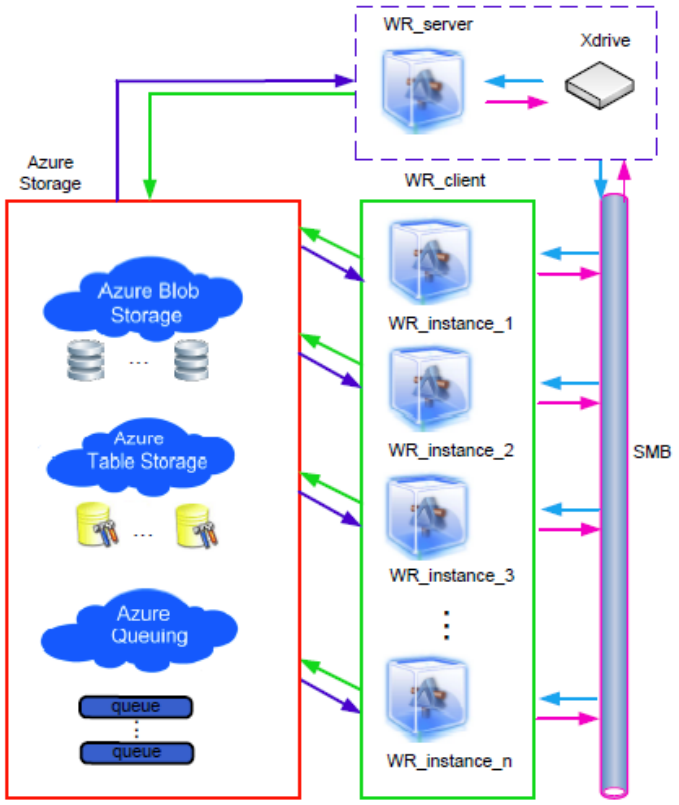
cases. For a single zeolite entry, a new task message is sent directly to another queue referred as global task queue. For a merged file of zeolite entries, the file is first split into separate files and then new task messages are created, each using the zeolite database number as identifier, Finally the messages are sent to the global task queue. Azure Queuing provides a stable buffer necessary for managing workload bursts. We developed our own SAMP project task schedule; a task is serialized into an Azure message and all task messages are then queued into the global task queue that identifies all the uploaded files. All our Worker Role instances compete for tasks from this global task queue. A third queue hosts the WEKA requests, and a fourth queue hosts the download requests.

First the WR_server fetches data from the upload queue. Then the WR_client fetches messages from the global task queue. At this point the WR_client instances execute the supercell service and the descriptor service. These WR_client instances interact with Drive storage where files/ libraries/executables are stored and update data in Blob and Table storage. Finally, the WR_client instances organize the fetching back of results to the Web Role. When one WR_client instance fails, the task in process is restored into the global task queue allowing another WR_client instance to pick it up.

Our data are stored and processed directly within the web application. The processing status (conversion, supercell, descriptor, Random Forest) of each zeolite entry is displayed in the web application. Zeolite entries are sorted according to their starting processing time and their processing status is recorded in each step. Table storage is used to store data with up to 256 entries allowing us to query these data as if they were part of a database. SAMP has a “conversion” table that holds the conversion of the original zeolite entry format to a .csv file. The SAMP “supercell” table stores the results of processing the supercell service in which the X, Y, Z coordinates of a supercell are created from the eolite reduced cell given in the ICSD. The SAMP “descriptor” table stores the values of the descriptors calculated with the descriptor service. The SAMP “random forest” table stores the descriptor values into a unified table in .csv format. It is this table that will be used by the machine learning algorithm. Last, but not least, Drive storage is used to store in-house codes, executable files, as well as all third party libraries.

4. SAMP parallelization through Server Message Block and its Web access

Server Message Block (SMB) is a network protocol that provides shared access to files and communications between nodes on a network. In the SAMP implementation, SMB connects the multiple instances of the WR_client with the WR_server. This virtual share is used to facilitate communication between the processes and the actual computers in the cloud once these computers have been authenticated. Figure 5 shows the scheme that is now operational of the SAMP project in Windows Azure. The Microsoft computer allocation to our NSF supplement allows us to use 20 worker roles. The actual SAMP scheme is using 18 instances of the WR_client and one of the WR_server.



5. SAMP Computing in the Cloud performance

5. SAMP Computing in the Cloud performance

A daily start of any calculations implies deployment of the SAMP system in Windows Azure (application packages upload, system initialization, Web/Worker Role creation, VC++/Java Runtime installation, Xdrive sharing using SMB protocol). This takes up to 40 minutes to be completed. The allocation of the Xdrive sharing time is also a lengthy step. In our assessment, MS needs to improve these two issues, or the cloud will not be an attractive environment for scientific applications.

The computer boxes of the MS cloud are servers with quad-core AMD™ processor 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 4.00GB memory running Windows 7 Enterprise.

(a)

(b)

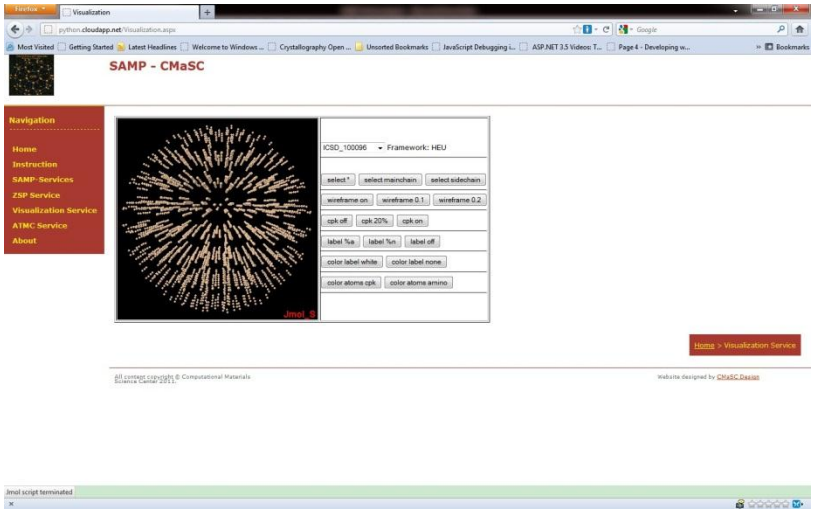


Fig. 6. Two web pages of the SAMP cloud computer user interface

First test is to use 18 WR_client instances, each loaded with one zeolite entry, and the test was repeated six times. Fig. 7 shows the Xdrive sharing time indicating the high dispersion of the time it takes for each WR_client instance to process one entry while sharing Xdrive with SMB. The figure also shows the high variability on the time in the six different trials. This is probably intrinsic to CIC, since we have no way to control the load of the processors and/or storage where the WR_clients do their job. Then, on average for this set of 18 zeolite entries, the sharing time is 9.75 minutes. Meanwhile, a test of processing time alone of these 18 entries yielded a mean of 5.29 ± 0.02 minutes. Additionally, the 18 zeolite entries were sent to one WR_client instance only so the SMB was not used. This test was repeated three times and used processed if 18 WR_client instances were working contemporarily (18 x 5.29), it is clear that the SMB is not providing an ideal speedup, and that instead the implementation degrades this embarrassingly simple parallelization by tagging 1.2 minutes more to each process.

This cloud-based test is to be compared with the time it takes to process the same 18 zeolite entries in a sequential manner (no parallelization) in our dedicated local PC. The run was repeated five times, giving a mean of 66.2 ± 0.2 minutes of total execution cpu and I/O. Thus, locally we have that the process takes on average 3.7 minutes per zeolite entry. As expected, this time is less than the 5.3 minutes it takes to the CIC implementation using one WR_client instance because the local processor is faster. Our jobs are not memory eager, so the discrepancy in memory size doesn't play a role. However, locally, there is no drive sharing time that allows for further speed.

All together, it is seen that the system as it is now implemented allows saving time in the overall execution. However, its efficiency is not the best. We have started to experiment on a different system in which we distribute also a drive to each worker role client instance. It is our hope to gain performance efficiency by eliminating SMB, although the cloud cost might be higher since 18 Xdrives are needed.

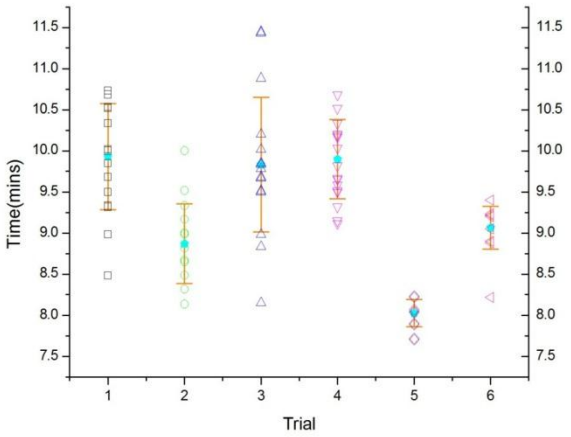


Fig. 7. SAMP sharing time in Window Azure

6. Challenges of using Computing in the Cloud with Azure

Our biggest challenge is the need to create a *compute system* for each number crunching project. In turn, this implies to have a full time person creating such system so that computations could be carried on. This is a very expensive investment in human resources, investment that we cannot price in dollar amount, and that we cannot expect NSF to be able to afford for every future project.

For this specific project, in addition of the time paid to an employee, there are uncountable PIs (principal investigator) hours invested, that will be fully lost when a new project would need to be implemented again in the Azure platform. While it is very simple to port our codes to a linux cluster that has 20 cores, use of the Azure cloud requires the complete creation of a system for only then be able to start production. Any of our students can port our codes to a cluster. However, for the programming paradigm in Azure we need a special person, with knowledge of the Windows/Windows Server/ASP.net, etc environments. Given the choice of receiving funds of $25,000 to acquire our own small cluster, or to have equivalent cpu and storage time in the Azure platform, we would select the funds.

Among our findings, the time to "prepare the cloud environment" is inexcusably long (involves a substantial waste of time). Additionally, uploaded libraries get corrupted often and need to be re-uploaded. Lack of secure access (secure shell, ssh) to load/access your own space is not efficient. Lack of *any* of the tools regularly accessible in a supercomputing center (compilers, libraries, environments, etc) is a drawback. Lack of advanced compilers compatible with Windows makes our applications slower and more bulky in Azure than in local computers. Technical support is basically inexistent in Azure.

References

1. Yang, S., Lach-hab, M., Vaisman, I. I., and Blaisten-Barojas, E., "Machine Learning Approach for Classification of Zeolite Crystals," In Proceedings of the 2008 International Conference on Data Mining, CSREA: Las Vegas, NV, (2008) 702-706.

2. D. A. Carr, M. Lach-hab, S. Yang, I. I. Vaisman, and E. Blaisten-Barojas, "Machine learning approach for structure-based zeolite classification, " Microporous and Mesoporous Materials 117, 339-349 (2009).

3. M. Lach-hab, S. Yang, I. Vaisman and E. Blaisten-Barojas, "Assignment of Framework Types to the Zeolite Crystals in the Inorganic Crystal Structure Database," arXiv:0904.2597 (April 2009).

4. Yang, S., Lach-hab, M., Vaisman, I. I., and Blaisten-Barojas, E., "Machine learning identification of zeolite framework types," In the 2009 International Conference on Artificial Intelligence (ICAI), CSREA, Las Vegas, NV, (2009) 340-344.

5. Yang, S., Lach-hab, M., Vaisman, I. I. and Blaisten-Barojas, E., "A cheminformatics approach for zeolite framework determination," Lecture Notes in Computer Science (LNCS) 5545, 160-168 (2009).

6. S. Yang, M. Lach-hab, I.I. Vaisman, and E. Blaisten-Barojas, "Identifying Zeolite Frameworks with a Machine Learning Approach," J. Phys. Chem. C 113, 21721-21725 (2009).

7. M. Lach-hab, S. Yang, I. I. Vaisman, and E. Blaisten-Barojas, X. Li, V. L. Karen, "Framework Type Determination for Zeolite Structures in the Inorganic Crystal Structure Database," J. Phys. Chem. Reference Data 39, 033102 (2010).

8. M. Lach-hab, S. Yang, I. I. Vaisman, and E. Blaisten-Barojas, "Novel Approach for Clustering Zeolite Crystal Structures." Molecular Informatics 29, 297-301 (2010).

9. S. Yang, M. Lach-hab, E. Blaisten-Barojas, X. Li, V. L. Karen, "Machine Learning Study of the Heulandite Family of Zeolites," Microporous and Mesoporous Materials 130, 300-313 (2010).

10. Inorganic Crystal Structure Database (ICSD): < http://www.fiz-karlsruhe.de/icsd.html>, < http://www.nist.gov/srd/nist84.htm>.

11. X. Dong and E. Blaisten-Barojas, "Adaptive Tempering Monte Carlo Method," J. Computational & Theoretical Nanoscience 3, 118-127 (2006).

12. X. Dong, D. Klimov, and E. Blaisten-Barojas, "Protein Folding with the Adaptive Tempering Monte Carlo Method," Molecular Simulation 33, 577-582 (2007).

13. J. Lyver, IV and E. Blaisten-Barojas, "Effects of the interface between two Lennard-Jones crystals on the lattice vibrations: a molecular dynamics study," J. Phys.: Condensed Matter 29, 345402 (2009).

14. A. Patrick, X. Dong, T. Allison, and E. Blaisten-Barojas, "Silicon Carbide Nanostructures: a Tight Binding Approach " J. Chem. Phys. 130, 244704 (2009).

15. Y. Dai and E. Blaisten-Barojas, "Monte Carlo Study of Oligopyrroles in Condensed Phases," J. Chem. Phys. 133, 034905 (2010).

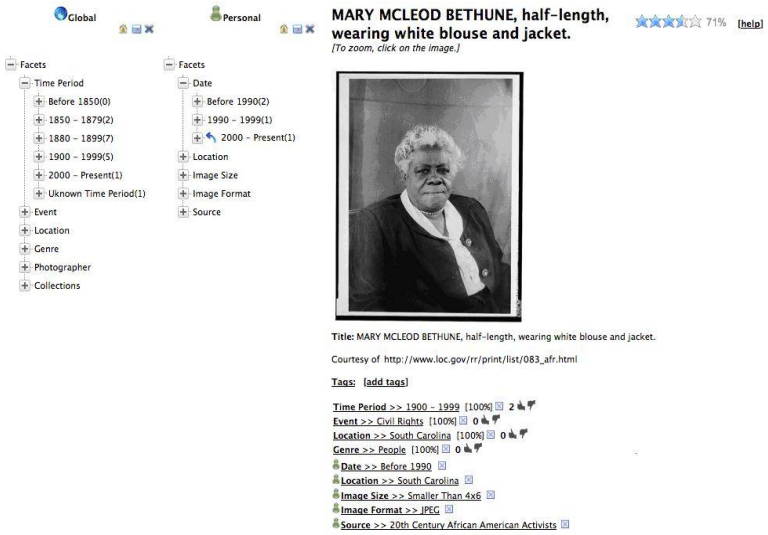
16. Q. Xing and E. Blaisten-Barojas, "The Zeolite Structural Predictor in the Windows Azure Platform," to be submitted.

17. M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten, "The WEKA Data Mining Software: An Update", SIGKDD Explorations 11, Issue 1 (2009(.

FACET ON CLOUD: A SOCIAL CLASSIFICATION SYSTEM

Harris Wu, Kurt Maly, Mohammad Zubair, Old Dominion University

With ever increasing amount of information and the need for classification, harnessing social intelligence is a promising direction for large-scale classification. We are developing a web-based system (FACET) that allows users to collaboratively organize multimedia collections into an evolving faceted classification. The FACET system includes a wiki-like interface that allows users manually classify documents into their personal document hierarchies as well as the global faceted classification schema, and backend algorithms that automatically classify documents and

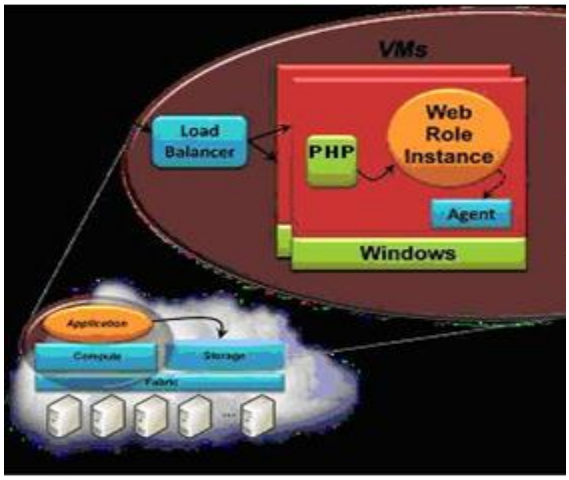


systematically enrich the classification schema.

Evaluating FACET with millions of documents and thousands of users allows us to answer research questions specific to large-scale deployment of social systems that harness and cultivate collective intelligence. For example, how to merge thousands of users' individual document hierarchies into a global schema? How to build a knowledge map of thousands of experts in different domains?

Recently we started evaluation of the FACET system on Microsoft Azure. Azure has the potential to scale the FACET system to virtually unlimited number of users, documents, schema objects and revisions, and user activity logs.

The FACET system is a research prototype built on Joomla, a popoular open-source content management system, and originally on the LAMP stack: Linux, Apache, MySQL and PHP. To deploy the FACET system on Azure, we had to move the metadata repository and session management from MySQL to SQL Azure. To continue to benefit from evolving features in the Joomla community, however, we chose to keep MySQL to support Joomla's core non-data intensive features.



Users of the FACET system are served by multiple Web Role instances. Browsing and classification are supported by queries to SQL Azure. Evaluation has shown that a single Web Role instance of medium size (2 core processors, 3.5GB RAM) can support ~200 concurrent users.

The backend classification algorithms run on Worker Role instances. One of the most computing intensive backend procedures is to compute the similarities (both textual and structural) among user-created categories. By dividing the work into 2 extra large instances, the pair-wise similarity computation of over 10,000 user categories can be computed with 24 hours. We are improving the algorithm to make it more efficient by utilizing incremental updating techniques.

While the backend computation can be arguably implemented on a super computer, the web-based user interface certainly benefits from the dynamic configuration and virtually unlimited bandwidth of Microsoft Azure. We found Azure very effective in addressing dynamic computing needs of the project. When not conducting user experiments, we can easily reduce the number of instances. Recently Microsoft has started providing Virtual Machine based deployment, which greatly simplified the process of deploying a computing-intensive program on Azure. We plan to use Azure in future research projects to have more flexibility in managing the computing costs. If given a ~$20K budget for computing costs, we believe Azure is more cost-effective to support a project with bursty computing needs at the evaluation phase.

The biggest challenges with Azure in our project were 1) deploying our system onto Azure and 2) porting the MySQL-based system to SQL Azure, 3) moving data from local databases to SQL Azure. Deployment requires setting up various pieces of software and quite a bit of research. Initially we let students to set up the environment on their own laptops, which turned out to be challenging. Porting MySQL-based Joomla system to SQL Azure also turned out to be a daunting exercise. After porting Joomla from MySQL to Microsoft SQL Server, We have contributed our code modifications to SourceForge to help others in similar efforts. The lessons we have learned should benefit similar efforts that attempt to deploy open source software on the Azure cloud.

SPATIAL OVERLAY OPERATIONS ON VECTOR-BASED GIS DATA ON AZURE

Sushil Prasad, Georgia State University

Researchers in Geographic information systems and science (GIS) have always perceived large scale vector-data computation as a challenge due to the intensity of the data. When large volumes of data are deployed for spatial analysis and overlay computation, it is a time consuming task, which in many cases is also time sensitive. Additionally, data integrity concerns make it prohibitively difficult to partition such files into smaller pieces, consequently making the processing of these files compute-intensive. Processing of these files is tedious and time consuming due to data-intensive and irregular nature of underlying computation. Moreover, it could be time sensitive as the results could help policy makers make informed decisions. In this project we have designed and implemented Crayons - A Cloud based parallel system for GIS overlay operations. We believe Crayons to be the first distributed GIS system over cloud capable of end-to-end spatial overlay analysis. Crayons scales well for sufficiently large data sets, achieving end-to-end speedup of over 40-fold employing 100 Azure processors. For smaller, more irregular workload, it still yields over 10-fold speedup. The link to Crayons' source code and manuscripts can be found at http://www.cs.gsu.edu/dimos/crayons.html.

MOTIVATION AND CHALLENGES

For a wide range of large scale distributed computing applications from Geosciences, the demand for resources varies significantly during the course of execution. While a set of dedicated resources for such applications could result in under-utilization most often, at other times the system could perform better by utilizing more resources than available. Therefore, a dedicated pool of resources, regardless of the capacity, might not be suitable for such applications. The emerging cloud platforms, such as Microsoft's Azure - with their potential for large scale computing and storage capabilities, easy accessibility by common users and scientists, on demand availability, easy maintenance, sustainability, and portability - have the promise to be the platform of choice for such GIS applications.

THE ARCHITECTURE OF CRAYONS

We have created three different architectures of Crayons. These versions only differ in how the load is balanced among worker roles. We are going to discuss the details of one of the versions in this document. Figure 1 shows the architectural diagram of Crayons with centralized load balanced version employing an extra large virtual machine(VM) as the web role. End users have the option to upload their input files in GML format to cloud or to operate on the existing files. Since uploading is a trivial process we will assume that the files are already available in the cloud storage. User selects the GML files to be processed along with the spatial operation to be performed on these files. First of these two selected files is treated as the base layer and the other file is treated as the overlay layer. The web role immediately starts downloading the files from the Azure cloud storage and translates (parses) the features (polygons) from the input GML files into C# objects.

Since spatial overlay operations are computationally expensive, it is wise to prune the set of polygon pairs needed to be processed together. In order to create this intersection graph, Crayons finds each overlay polygon that can potentially intersect with the given base polygon and only performs spatial operation on these pairs. This is achieved using the coordinates of bounding boxes generated during parsing of input files. Intersection graph creation currently is based on sorting the polygons with (n log n) cost.

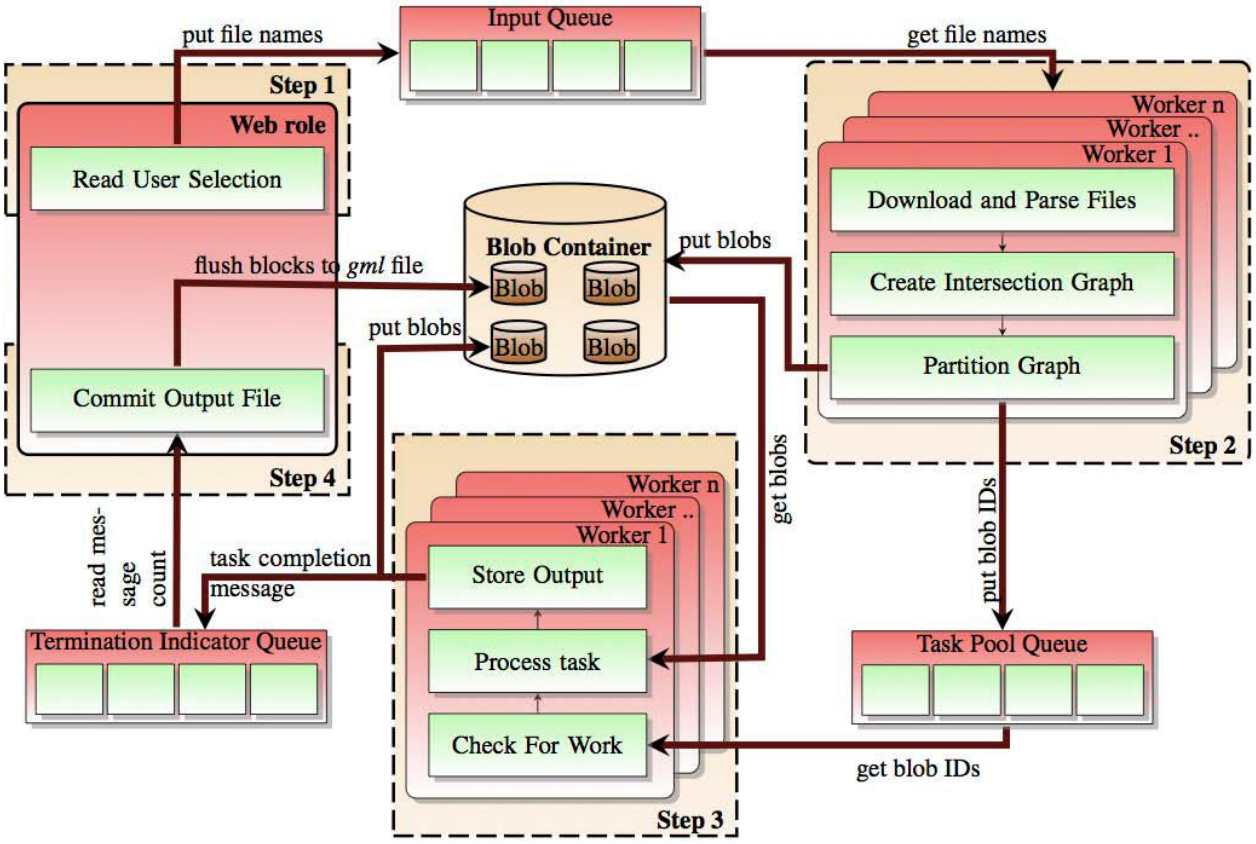


Figure 1. The architecture of Crayons

Intersection graph defines one-to-many relationship between the set of base polygons and overlay polygons. To create an independent task, one polygon from base layer and all related polygons from overlay layer are merged together as a task and stored in the cloud storage as a blob. The web role converts the C#'s polygon objects belonging to a task to their GML representation that gets stored in the blob storage. We prefer in-house serialization against C#'s serialization library to avoid excessive metadata required to convert an object to string. Each task is given a unique ID, this id is communicated to the worker roles using a message over a queue that serves as a shared task pool among workers and thus facilitates dynamic load balancing. Queue storage mechanism provided by Azure platform comes handy here to implement task based parallelism and for fault tolerance. Worker roles continuously check the shared task pool (queue) for new tasks. Since this can throttle the Queue storage, with a limit to support a maximum of 500 requests per second, if there is no message in the Queue we let a worker sleep for a few seconds before sending next request. However, if there is a task (message) in the shared task pool, the worker reads the message and consequently

hides it from other workers, downloads the blob with id stored in this message, converts the content of the downloaded blob to get the original base and overlay polygon objects back (deserialization), and performs the spatial overlay operation by passing a pair of base polygon and one overlay polygon at a time to GPC library for sequential processing.

GPC library returns the resultant feature as a C# polygon object that is converted to its equivalent GML representation and appended as a block to the resultant blob stored in the cloud storage. Azure API PutBlock is used to achieve parallel writing to the output blob. This API facilitates the creation of a blob by appending blocks to it in parallel and if the sequence of the features is not critical this API can significantly improve the performance. Additionally, each worker role puts a message on the termination indicator queue to indicate successful processing of the task. The web role keeps checking the number of messages in termination indicator queue to update the user interface with the current progress of the operation. Logically, when all of the tasks have been processed the number of messages in the termination indicator queue will match the number of base polygons. When this happens web role commits the resultant blob and flushes it as a persistent blob in the blob storage. The resultant blob becomes available for downloading or further processing, user interface is also updated with the URI of resultant blob. To commit a blob created using blocks the Azure API PutBlockList is used. In order to use PutBlockList it is necessary to provide the list of blocks to be committed, this list is maintained at the cloud end and can be downloaded by the web role by using another Azure API GetBlockList.

The queue storage mechanism provided by Azure platform comes handy for fault tolerance during processing. After a worker role reads a message from the task pool, the message disappears from the task pool for other worker roles and is subsequently deleted by the worker role after the processing ends successfully. In the event of a failure, the message does not get deleted and appears in the queue after a stipulated amount of time.

Cloud Computing vs Large Compute-Clusters/Grids:

Feasibility: Lee et. al., [1] present the case of hurricane Katrina in 2005 to conclude that the only answer to the scientific and operational grand challenge problem is enormous computer power. However, it is not economically possible to dedicate the required amount of resources for this single purpose. Therefore 1. resources must be shared but available on-demand, 2. the platform should be scalable on-demand, and 3. resources should be easily accessible to GIS scientists and policy decision makers (not necessarily computer savvy) in a user friendly way over the web. These guidelines are mandatory for the proliferation of such GIS systems. Since a Grid or any large compute cluster cannot adapt to these guidelines, we chose to architect Crayons for cloud computing as cloud computing appears to be the only feasible platform.

Cost: We would like to address the issue of cost for HPC application development on cloud by answering following two questions asked by the funding agencies: (1) The median amount of cloud resource consumed by the group in this study has cost about $25,000 for 2011. If you had a way to request this amount from your funding agency for cloud resources in the future, would you do so?

Answer: Yes. (2) Or, would you prefer to request the same amount to purchase new hardware that you would manage yourself? If so, would you be able to accomplish the same research results? Answer: Probably not, as the scalability experiments can only be performed on cloud with 10s and 100s of processors. On the other hand, we already have a 80-core linux cluster for more controlled, limited experiments. If we did not, first $25-50K will go into purchasing a cluster to obtain initial capabilities as a HPC researcher.

CHALLENGES OF WORKING WITH AZURE CLOUD RESOURCE

We chose Azure cloud platform over other rather mature cloud platforms as Azure platform provided us the opportunity to think-outside-the-box to devise an architecture for systems research for data and compute intensive scientific applications as it currently lacks support for traditional distributed computing design paradigms such as MPI or map-reduce. On the other hand, Azure's robust middleware APIs and artifact enable finer-grained task level fault tolerance which other clouds with system image level control cannot.

HPC program development over the still-emerging Azure platform is very difficult and tedious even for experienced parallel programmers and algorithm developers. There are two primary challenges for application development on any cloud platform that result in the transition from traditional application development to application development on cloud not so satisfying. First and foremost, the time taken to deploy an application to cloud is significantly high compared to a traditional web application development. Secondly, application debugging in cloud is difficult as the only debugging support is through logging and thus a developer needs to check the logs to identify the potential software bugs. There are continuous efforts to enhance debugging support, such as intellitrace and remote logging for role instances, but it is still far from what is available for traditional application development. Moreover, we faced a few technical challenges during the course of engineering Crayons system including

1 Limitation of Table Storage

Azure Table storage provides a service to store large amount of data that needs additional structure. Table storage can be queried to retrieve data based on the underlying structure. However, a table can only store entities that are a maximum of 1 MB in size. Since Crayons (and many other scientific HPC applications) handles and stores fundamental units of operation that are usually larger than 1 MB, Table storage was realized not to be a good fit in this case.

2 Queues - FIFO Behavior

Crayons uses Queue storage service of Azure platform to communicate between web role and worker roles. Azure Queue storage differs from the traditional Queue data structure as it lacks the ability to guarantee a FIFO operation. This lack of guarantee for FIFO operation can create issues if a Queue is to be used to signal the beginning or end of a special event. This is the reason why we had to create a dedicated termination indicator queue where worker roles send messages to indicate successful completion of tasks.

3 Local Azure Simulator vs. Azure Cloud Environment

Microsoft Azure provides a simulator that can be installed on a personal computer so that Azure based applications can be debugged locally. Interestingly, we found that the cloud platform does not necessarily replicate the local Windows OS and .Net platform appropriately. In our case, the issue was absence of a standard Windows dynamic link library (dll) file named msvcrt100.dll on Azure cloud platform; the application started executing as expected once this dll file was manually packed inside the package that was deployed to the Azure cloud.

REFERENCES

[1] C. A. Lee, "A perspective on scientific cloud computing," in Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, ser. HPDC '10. New York, NY, USA: ACM, 2010, pp. 451–459.

SQL SHARE: DATABASE-AS-A-SERVICE FOR LONG TAIL SCIENCE

Bill Howe, Garret Cole, Alicia Key, Nodira Khoussainova, Leilani Battle
University of Washington, Seattle, WA

Science is being transformed by the democratization of automated, high-throughput data acquisition technology: small labs and individual researchers now have ready access to DNA sequencers, high-resolution simulations of the Earth, satellite imagery, terabytes of telescope imagery, and a national network of oceanographic sensors (e.g., OOI, IOOS).

But there has not been a commensurate democratization of automated, high-throughput data analysis technology. Relational database technology remains remarkably underused in science, especially in the *long tail* — the large number of relatively small labs and individual researchers who, in contrast to "big science" projects, do not have access to dedicated IT staff or resources yet collectively produce the bulk of scientific knowledge [4].

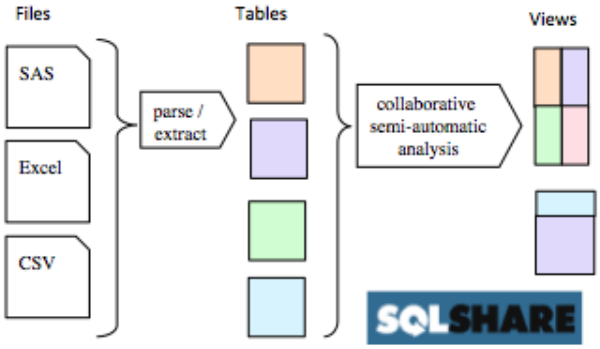


Figure 1: SQLShare simplifies collaborative, semi-automatic data management in the cloud using Windows Azure.

Number of uploaded datasets	> 1000
Number of non-trivial views	> 500
Number of queries executed	> 4000
Number of active users	> 50
Max (tables, views) for a user	(192,133) Total
size of all data	44.5 GB
Size of largest table, in rows	1.1M

Figure 2: Early usage for the SQLShare system during a 6-month pilot period. We use data collected during the pilot to perform preliminary evaluation of advanced features.

The problem persists despite a number of prominent success stories [2] and an intuitive correspondence between exploratory hypothesis testing and the ad hoc query answering that is the "core competency" of an RDBMS. Some ascribe this underuse to a mismatch between scientific data and the models and languages of commercial database systems[1]. Our experience (which we describe throughout this paper) is that standard relational data models and languages can manage and manipulate a significant range of scientific datasets. We find that the key barriers to adoption lie elsewhere:

1. *Setup.* Local deployments of database software require too much knowledge of hardware, networking, security, and OS details.
2. *Schemas.* The initial investment required for database design and loading can be prohibitive. Developing a definitive database schema for a project at the frontier of research, where knowledge is undergoing sometimes daily revision, is a challenge even for database experts. Moreover, the corpus of data for a given project or lab accretes over time, with many versions and variants of the same information and little explicit documentation about connections between datasets and sensible ways to query them.
3. *SQL.* Although we corroborate earlier findings on the utility of SQL for exploratory scientific Q&A, we find that scientists need help writing non-trivial SQL statements.
4. *Sharing.* Databases too often ensconce one's data behind several layers of security, APIs, and applications, which complicate sharing relative to the typical status quo: transmitting flat files.
5. *Visualization.* While SQL is appropriate for assembling tabular results, our collaborators report that continually exporting data for use with external visualization tools makes databases unattractive for exploratory, iterative, and collaborative visual analytics.

As a result of these "S4V" challenges, spreadsheets and ASCII files remain the most popular tools for data management in the long tail. But as data volumes continue to explode, cut-and-paste manipulation of spreadsheets cannot scale, and the relatively cumbersome development cycle of scripts and workflows for ad hoc, iterative data manipulation becomes the bottleneck to scientific discovery and a fundamental barrier to those without programming experience.

Having encountered this problem in multiple domains and at multiple scales at the UW e-Science Institute, we have released a cloud-based relational data sharing and analysis platform called SQLShare [3] (http://sqlshare.escience.washington.edu) that allows users to upload their data and immediately query it using SQL — no schema design, no reformatting, no DBAs. These queries can be named, associated with metadata, saved as views, and shared with collaborators. Beyond the basic upload, query, and share capabilities, we explore techniques to partially automate difficult tasks through the use of statistical methods that exploit the shared corpus of data, saved views, metadata, and usage logs.

The SQLShare platform was designed to address these challenges based on the following observations:

- We find that cloud platforms drastically reduce the effort required to erect and operate a production- quality database server.
- We find that up-front schema design is not only prohibitively difficult, but unnecessary for many analysis tasks and even potentially harmful — the "early binding" to a particular fixed model of the world can cause non-conforming data to be overlooked or rejected. In response, we

postpone or ignore up-front schema design, favoring the "natural" schema gleaned from the filenames and column headers in the source files.

- We find that when researchers have access to high-quality example queries, pertinent to their own data, they are able to self-train and become productive SQL programmers. The SQL problem then reduces to providing such examples. We address this problem by deriving "starter queries" —automatically— using the statistical properties of the data itself (as opposed to the logs, the schema, or user input that we cannot assume access to.) Moreover, we analyze the corpus of user-provided free-text metadata to exploit correlations between English tokens and SQL idioms.
- We find that streamlining the creation and use of views is sufficient to facilitate data sharing and collaborative analysis.
- We find the awkward export-import-visualize cycle can be avoided by eager pre-generation of "good" visualizations directly from the data.

Removing Obstacles to Scientific Productivity

Direct access to query capabilities for researchers accustomed to manipulating spreadsheets can have a remarkable effect. At an early demonstration of our platform, the results of a simple SQL query written "live" in less than a minute caused a post doc to exclaim "That took me a week!" — meaning that she had literally spent a week manually cleaning and pre-filtering a handful of spreadsheets, then using copy-and-paste to compute what was essentially a join. Within a day, the same post doc had derived and saved several new queries. The experience was not isolated: the director of her lab has contributed several of her own SQL queries. She has commented that the tool "allows me to do science again," explaining that she felt "locked out" from personal interaction with her data due to technology barriers, relying instead on indirect requests to students and IT staff. She is not alone — several researchers we have surveyed informally have reported that the ratio of time they spend "manipulating data" as opposed to "doing science" is a staggering 9 to 1. This number should give pause —taxpayer money flows through federal funding agencies to pay senior scientists to spend 90% of their time doing something other than science.

Windows Azure as a Critical Success Factor

The SQLShare platform is implemented as an Azure Web Role that issues queries against a SQL Azure back end. The Azure Web Role implements a REST API and manages communication with the database, enforcing SQLShare semantics when they differ from conventional databases. In particular, the Web Role manages fault-tolerant and incremental upload of large datasets, analyzes the uploaded data to infer types and recommend example queries, manages authentication with external authentication services, operates on the system catalog, parses and formats data for interoperability with external systems, provides asynchronous semantics for all operations that may operate on large datasets, and handles all REST requests.

Windows Azure was essential to the success of the project by allowing a single developer to build, deploy, and manage a production-quality web service. In typical web projects, a dedicated "release engineer" or "build team" may be responsible for releasing versions from test to production safely, robustly, and without downtime. The web-based Windows Azure console completely eliminated the need for such a position; the lead developer was able to unilaterally deploy both new releases and hotfixes as needed with zero downtime and near-zero overhead. Moreover, the test cycle was significantly simplified by the availability of the Azure SDK (for unit tests) and the ability to swap deployments in the Azure console (for system and performance tests). In past projects, managing a

production-quality deployment of a web application, including upgrades and hotfixes, tended to be the major source of user-facing errors. Keeping the development environment and test environments in synch was a challenge. Windows Azure completely eliminated this burden.

As a result, we simply would not have been able to deliver this application to the UW campus with our given resources without the use of Windows Azure.

The "flagship" SQLShare web client (Figure 1) exercises the API to provide upload, query, sharing, and download services. The following features highlight key differences between SQLShare and a conventional RDBMS.

No Schema We do not allow CREATE TABLE statements or any other DDL; tables are created directly from the columns found in the uploaded files. Just as a user may place any file on his or her file system, we intend for users to put any table into the SQLShare "table system." By identifying patterns in the data (keys, union relationships, join relationships, attribute synonyms) and exposing them to users through views, we can superimpose a form of schema post hoc, incrementally — a *schema later* approach.

Tolerance for Structural Inconsistency Files with missing column headers, columns with non-homogeneous types, and rows with irregular numbers of columns are all tolerated. We find that data need not be pre- cleaned for many tasks (e.g., counting records), and that SQL is an adequate language for many data cleaning tasks.

Append-Only We claim that science data should never be destructively updated. We therefore do not support tuple-level updates; errors can be logically replaced by uploading a new version of the dataset. This approach allows aggressive caching and materialization to improve performance.

Simplified Views Despite their utility, we find views to be underused in practice. We hypothesize that the solution may be as simple as avoiding the awkward CREATE VIEW syntax. View creation in SQLShare is a side effect of querying — the current results can be saved by simply typing a name. This simple UI adjustment appears sufficient to encourage users to create views (Table 2).

Metadata Users can attach free-text metadata to datasets; we use these metadata not only to support basic keyword search, but to bootstrap query recommendations by mining correlations between English tokens and SQL idioms.

Unifying Views and Tables Our data model consists of a single entity: the *dataset*. Both logical views and physical tables are presented to the user as datasets. By erasing the distinction, we reserve the ability to choose when views should be materialized for performance reasons. Since there are no destructive updates, we can cache view results as aggressively as space will allow. However, since datasets can be versioned, the semantics of views must be well-defined and presented to the user carefully. We find both snapshot semantics and refresh semantics to be relevant, depending on the use case. Currently we support only refresh semantics.

Provenance The DAG of views used to produce a given result is itself useful to communicate a simple form of provenance to collaborators.

References

[1] P. G. Brown. Overview of scidb: large scale array storage, processing and analysis. In *Proceedings of the*
2010 international conference on Management of data, SIGMOD '10, pages 963–968, New York, NY, USA, 2010. ACM.

[2] J. Gray, D. T. Liu, M. A. Nieto-Santisteban, A. S. Szalay, D. J. DeWitt, and G. Heber. Scientific data management in the coming decade. *CoRR*, abs/cs/0502008, 2005.

[3] B. Howe, G. Cole, E. Souroush, P. Koutris, A. Key, N. Khoussainova, and L. Battle. Database-as-a-service for long tail science. In *Proceedings of the 23rd Scientific and Statistical Database Management Conference (SSDBM '11)*. Springer, 2011.

[4] Big science and long-tail science. http://wwmm.ch.cam.ac.uk/blogs/murrayrust/?p=938. term attributed to Jim Downing.

SCALABLE ALGEBRAIC VISUALIZATION IN THE CLOUD

Bill Howe, University of Washington

The requirements for large-scale scientific visualization systems and large-scale scientific databases are converging. Visualization systems are being equipped with rudimentary query processing capabilities, observing that simply "throwing datasets" through the graphics pipeline ignores the scalable restructuring, manipulation, and filtering required by realistic applications. Further, there is increasing emphasis on in situ visualization —execution of visualization and data processing on a single platform to avoid data transfer costs and afford new optimizations. In particular, the role of cloud computing as both a large-scale visualization platform as well as a large-scale data processing platform is underexplored.

In response to these pressures, we propose a novel approach to the problem of scalable visualization, one informed by the algebraic query processing techniques developed by the database community and coupled with recent advances in data-intensive scalable computing platforms such as MapReduce, Dryad, and their contemporaries. Specifically, we propose to develop a *visualization algebra* analogous to the relational algebra but specialized for manipulating 4D mesh datasets instead of (simpler) relations.

In previous work, we developed a data model and algebra called GridFields for manipulating unstructured grid datasets used in the context of finite element simulation. The resulting framework was general enough to capture complicated gridded datasets and algorithms and afforded several algebraic optimization techniques, but did not support efficient parallel processing and could not easily express certain common visualization tasks. Specifically, GridFields assume a fixed topology and are therefore inappropriate for algorithms such as isosurface extraction and mesh simplification that must mutate or create topological elements. Other existing visualization systems favor depth over breadth, focusing on optimizations for specific visualization algorithms on specific hardware rather than a generic platform for visual analytics that can run on the shared-nothing clusters of commodity computers typically found in the cloud. Our approach, to be developed and evaluated on the Windows Azure platform, provides a core set of scalable primitives for manipulating mesh datasets using shared-nothing architectures, capable of expressing a variety of visualization algorithms, and amenable to algebraic reasoning and optimization.

Application Domain We work extensively in the area of data-intensive oceanography. We have active partners at UW and beyond who are working with the serial GridField model and are interested in using Azure. Further, we have partners at Microsoft Research who, in response to a white paper written by Howe, have developed an Azure-based OPeNDAP server for hosting oceanographic datasets. In independent work, we are working to populate this server with oceanographic simulation results. This system will provide a data source for parallel processing using GridFields. Each worker role will query the OPeNDAP server to extract a piece of the overall dataset, partitioning the data on the fly. In the simplest case, the data will be partitioned on time, and the task may be embarrassingly parallel. For example, a filter or gradient calculation can operate on each time step independently.

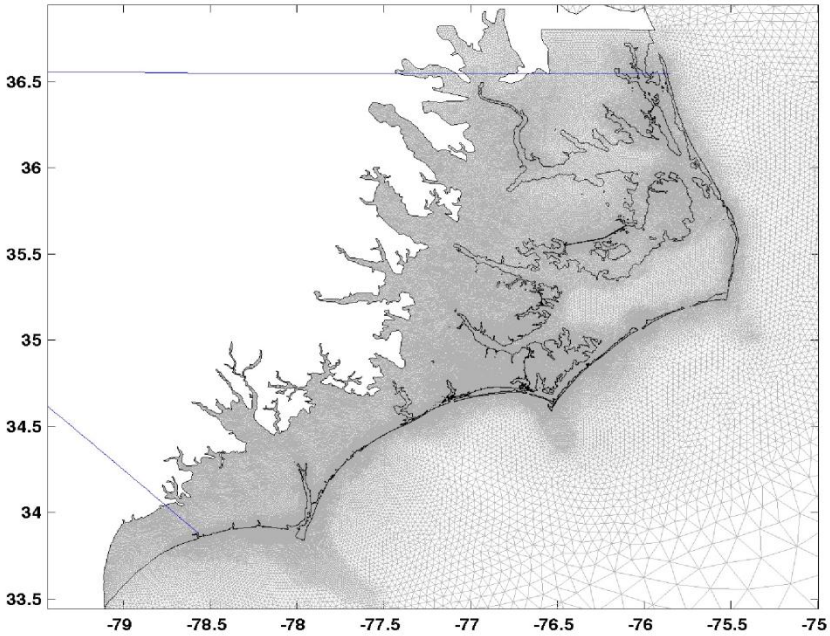


Figure 1: Example of an unstructured model grid from an ADCIRC floodplain analysis and forecasting application, which uses triangular linear finite elements for spatial discretization of the relevant physics. Models that use the unstructured grid approach are able to represent complex boundary features like the coastline and coastal flood plain.

Unstructured Grids

There is a critical need in the oceanographic community for interoperable visualization and analysis for unstructured grid model data. Numerical models are playing an increasing role in coastal oceanography, engineering, and policy, and unstructured grids (e.g. triangular, polygonal, polyhedral) are gaining popularity because of their ability to represent multi-scale domains that may include complex coastlines requiring a high-resolution grid as well as the open ocean for which a high-resolution grid is an inefficient use of computational resources. Figure 1 illustrates an unstructured grid used to model a complex coastline for floodplain analysis.

There are five unstructured grid models used in the United States oceanographic community; ADCIRC [4], FVCOM [1], ELCIRC [5], SELFE [5] and SUNTANS [2]. Each model has particular strengths that make it useful for a variety of scientific research and application domains (e.g., storm surge modeling, tsunamis, search and rescue, etc.). However, each model currently relies on proprietary file formats and a corresponding collection of custom analysis and visualization tools. As a result, the use of these ocean models — despite their power in capturing a range of hydrodynamic phenomena — is usually limited to direct collaborators with the model developers. Their outputs are simply too complicated to analyze and apply unilaterally. This parallel and entirely independent development results in significant duplication of effort, complicates software reuse, and limits the ability to perform model-model and data-model comparisons by the broader community. In contrast, the atmospheric modeling community — who use simpler *structured grids* exclusively — now benefits from a standard data model and common infrastructure investment led by the Climate and Forecast (CF) Metadata Community and by UNIDATA (an NSF-supported facility). The oceanographic community — who increasingly rely on unstructured grids — need an analogous investment in a unifying data model and infrastructure.

Operator	Description
scan/bind	Associate data with an existing grid.
restrict	Cull cells that do not satisfy a predicate.
cross	"Multiply" one gridfield by another.
merge	Combine multiple gridfields over their inter- section.
accrete	"Grow" a gridfield by adding neighboring cells.
regrid/apply	Map one gridfield onto another, aggregat- ing values.
fixpoint	Allow recursive execution of recipes.

Figure 2: List of gridfield operators and their descriptions

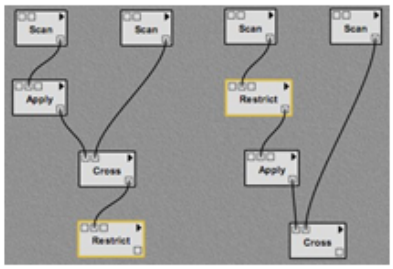


Figure 3: Two equivalent query plans using Grid- Fields. The system can automatically derive the optimized plan at right from the unoptimized plan at left thanks to the algebraic data model.

Algebraic Query-Driven Visualization

The data management community has begun to recognize the need for visual analytics [3], and the visualization community has begun to couple visualization techniques with remote query facilities. However, the "query" capabilities in "query-driven visualization" systems are generally limited to simple subsetting — the user specifies a region of interest as a "working set", and the system retrieves it and feeds it into visualization pipeline. We take a database-centric view of query capabilities, and argue that the more computation you can express in the data management layer, the better. Therefore, we are exploring the use of Hadoop for highly-scalable visualization pre-processing in fewer lines of code.

Database researchers hold these truths to be self-evident:

- It is better to move the computation to the data than the data to the computation.
- Declarative query languages are better than imperative programming languages.
- Reasoning about data manipulation algebraically facilitates automatic optimization and automatic parallelization.

Codd observed in 1970 that all the database products at the time were all processing tabular data, and basically used only about six underlying operations. If these operations could be formalized and made explicit, he reasoned, most of the work in accessing and manipulating data could be off-loaded to the database system instead of the programmer. Thirty years later, relational databases are a $18 billion dollar industry that powers most of the Internet.

A similar revolution may be possible for visualization with an appropriate change to the underlying data model and operators — a topological mesh, not a table, is the unit of data manipulation. We have developed an algebra of mesh data called GridFields [3] to facilitate reasoning and optimization. In previous work, we have formalized, implemented, and deployed the GridField data model for file-based manipulation of oceanographic simulation results. Our current effort involves porting this work to the cloud by designing parallel implementations of the GridField operators using first MapReduce, and now Windows Azure.

The fundamental data structure of the algebra is the Gridfield, an abstract data structure that separates topology from geometry and other data to expose equivalences and afford optimization. The data structure is manipulated by applying a composition of simple operators. Some of our operators are analogous to relational operators, but grid-enabled, while others are novel. For example, the restrict operator implements grid-safe subsetting, while the regrid operator maps data from one grid onto another. A list of operators in the core algebra is enumerated in Figure 2.

To parallelize these operators, we model a mesh as a set of key-value pairs in accordance with the MapReduce data model and the Windows Azure Table data model. Each key-value pair corresponds to a local topological neighborhood called a *stencil*. A stencil can be specified by a simple sequence of integers. The simplest stencil of the mesh vertices is written 0, while a stencil of triangles along with their vertices and adjacent triangles is written 202. Using MapReduce, we can organize and process a large mesh as a stream of stencils. For example, a smoothing filter is implemented over a stream of 020 stencils: the data value at each vertex is replaced by the average value from adjacent vertices.

References

[1] C. Chen, H. Liu, and R. C. Beardsley. An unstructured, finite-volume, three-dimensional, primitive equation ocean model: Application to coastal ocean and estuaries. *Journal of Atmospheric and Ocean- icTechnology*, 20:159–186, 2003.

[2] O. Fringer, M. Gerritsen, and R. L. Street. An unstructured-grid, finite-volume, nonhydrostatic, parallel coastal-ocean simulator. *Ocean Modeling*, 14(3-4):139–278, 2006.

[3] B. Howe, P. Lawson, R. Bellinger, E. Anderson, E. Santos, J. Freire, C. Scheidegger, A. Baptista, , and C. T. Silva. End-to-End eScience: Integrating Workflow, Query, Visualization, and Provenance at an Ocean Observatory. In *4th IEEE International Conference on e-Science (eScience'08)*, Indianapolis, Dec. 2008.

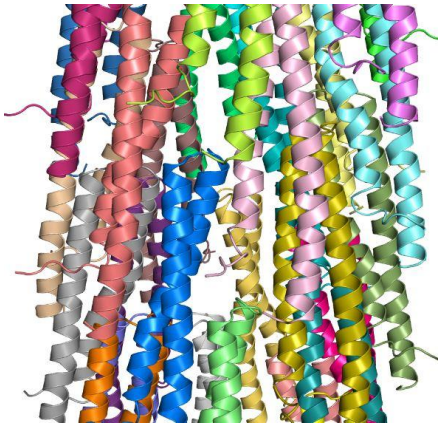
[4] J. J. Westerink, J. R.A. Luettich, J. Feyen, J. Atkinson, C. Dawson, M. Powell, J. Dunion, H. R. and E.J. Kubatko, and H. Pourtaheri. A basin- to channel- scale unstructured grid hurricane storm surge model as implemented for southern Louisiana. *Monthly Weather Review*, 136:833–864, 2008.

[5] Y. Zhang, A. Baptista, and E. Myers. A cross-scale model for 3d baroclinic circulation in estuary-plume- shelf systems: I. formulation and skill assessment. *Continental Shelf Research*, 24:2187–2214, 2004.

PROTEIN FOLDING WITH ROSETTA@HOME IN THE CLOUD

Nikolas Sgourakis, University of Washington, Baker Lab.

THE SCIENTIFIC CHALLENGE



A classic example of embarrassingly parallel computation in science is the Seti@home. This is based on volunteer computing where thousands of people make their pc available for an external agent to download tasks to it when it is not being heavily used. A standard framework for these applications is BOINC from Berkeley. In David Baker's lab at the University of Washington, they have built a protein folding application (Rosetta@home) based on BOINC. The problem with traditional volunteer computing is that volunteers are not very reliable. On the other hand, the cloud can be considered a very large pool of capability that can easily be turned into a "high thoughput" BOINC service. Specifically the scientific challenge was to elucidate the structure of a molecular machine called the needle complex, which is involved in the transfer between cells of dangerous bacteria, such as salmonella, e-coli, and others.

THE IMPLEMENTATION

To demonstrate this we used 2000 Azure cores to run a substantial folding challenge. The implementation was straightforward. The standard BOINC client was ported to a VM role on windows Azure. (This was also a test of this beta capability.) Because the client was designed to run on a Windows host, this was not difficult. The only changes that were needed were modifications to the Rosetta BOINC server in the UW lab. Specifically a special work queue was needed to filter jobs going to Azure so that only Dr. Sgourakis's jobs went there.

One of the main advantages of using Azure is that they did't have to handle support of the system, a very common problem with Rosetta@home. Of course on a real volunteer system the price is free. Consequently one has a choice. You can get the job done quickly, or you can get it for free.

SCALABLE, SECURE ANALYSIS OF SOCIAL NETWORK GRAPH ON THE AZURE PLATFORM

YOGESH SIMMHAN, ALOK KUMBHARE, MARK REDEKOPP AND VIKTOR PRASANNA,

UNIVERSITY OF SOUTHERN CALIFORNIA

The increasing availability of data relating to human activity and interactions at population scales is allowing social science researchers to understand and model human behavior. Political and social scientists are starting to correlate such large scale social media datasets with events that impact society as evidence abound of the virtual and physical public spaces intersecting and influencing each other. Mobile and social media also provide a convenient framework for "human sensors" to broadcast data for behavioral studies, such as understanding and even influencing the use of energy. They form a vehicle to crowd source citizen science that can engage the broader public in STEM activities, and help influence public policy.

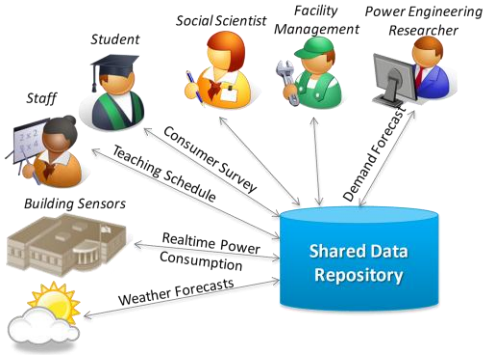
There exist a number of software challenges to support the needs of the under-served social informatics community. Of these, two key aspects that we are investigating are on data collection and sharing, and data analysis. Specifically, our research on Cloud platforms tries to answer the questions:

- *How can we securely collect, host and share personally identifiable social informatics datasets with social scientists and researchers at large scale?*
- *What are programming models and tools necessary to analyze large scale social network graphs in a timely and reliable manner?*

Our work is motivated by behavior analysis problems in the smart power grids domain where customer energy consumption data is used to predict future power demand and response to curtailment efforts. We also address common graph algorithms such as betweenness centrality that are vital for social network analysis. Success of these efforts will lead to better understanding of human activity, help provide more customized services to customers, drive energy policy and help build sustainable systems.

Cryptonite: Secure Storage Repository for Sharing Scientific Datasets on Public Clouds

Social sciences datasets often contain personally identifiable information that can compromise the privacy of human subjects who participate in studies. Institutional review boards pay particular attention to how datasets collected from surveys or behavior monitoring are stored and shared. These issues are exacerbated if when expanded to life sciences where regulations such as HIPAA are tightly enforced. The ability to gather human data has expanded rapidly with subjects actively sharing information through social media feeds or passively through mobile apps that report readings on sensors such as GPS, accelerometers and ambient light. Subjects sharing such data for research projects may wish to impose restrictions on data sharing within the research group or across the institution. For example, students and staff



Sharing consumer power consumption and behavior datasets in a campus microgrid

participating in a campus energy sustainability study may share power consumption, comfort levels, and facility usage information through a mobile app but wish to have controls over sharing of the data. Likewise, social sciences researchers may wish to share some of this information with power engineering researchers. The large scale, continuous update and need for sharing these datasets makes them suitable for hosting in Cloud repositories. However, this also expands the potential for data leakage and widens the attack surface for a security breach. Currently, Service Level Agreements from major Cloud providers have limited liability provisions for loss of security and privacy and is often a best effort. The Cloud Security Alliance's 2010 guidance recognizes “Malicious Insiders” and “Data Loss or Leakage” among the top five security threats in Clouds.

Cryptonite is a scalable storage repository designed to securely share datasets among a collaboration of users on untrusted public Clouds [1]. The design leverages the reliability and availability of Cloud storage while assuring data security and privacy managed by distributed clients. It improves manageability by avoiding setting up and maintenance of a dedicated machine for hosting the storage repository locally and ensuring its backup. Even using a local machine does not address issues of malicious insiders accessing the data.

Some of the design tenets of Cryptonite are:

- *Authorized clients are the only entities that have access to plain text data. Both the Cryptonite service running in the Cloud and Cloud data storage services hosting the (encrypted) datasets do not see plain text dataset.*
- *Any user in the collaboration can get access to the (encrypted) dataset. However, only authorized users can decrypt the data or update the data.*
- *Users can decrypt any data they are authorized to access using their global private key. They do not need to maintain a unique password for each of their files.*
- *The Cryptonite service is trusted to perform storage operations on encrypted datasets. However, an audit trail allows both clients and the service to verify and prove authorized and unauthorized operations.*

These address the shortcomings of more naïve solutions. Using a shared symmetric key for a file means that a unique key has to be shared with the users for each new file. Using ACLs means trusting the Cryptonite service or Cloud storage service to enforce it. Using PKI such as PGP allows many to one sharing (such as emails) but limits many to many sharing of files.

The owner of a data file encrypts it using a unique symmetric key for that file and sends the encrypted data file to the Cryptonite service running as an Azure Cloud Role. The Cryptonite service provides GET, PUT, POST and DELETE operations similar to Cloud storage services, and stores this encrypted file on Azure Blob Storage Service. The owner uses a strongbox file to record the symmetric key used to encrypt and decrypt the data file and uses broadcast encryption to encrypt the strongbox using the public keys of authorized users. The strongbox is also send to the Crypotonite service for storing in the Azure Cloud Blob storage. Authorized users can access an encrypted file by accessing the strongbox for that file, decrypts it using their private key to access the symmetric key, downloads the encrypted data file and decrypts it using the symmetric key.

Our early prototype shows an overhead of less that 10% for the Cryptonite storage repository as compared to directly storing the data file on Azure Blob Storage. We provide .NET clients to access the service though a library. This can be extended to a service similar to “dropbox” while providing security and privacy guarantees. Clients in other languages such as Java can also be implemented using open security algorithms and libraries.

Challenges with Azure Cloud Resources

During our implementation of Cryptonite on Azure, we encountered a few challenges. Transferring large data files led to network connections closing occasionally due to inactivity. This was particularly

exacerbated when using Windows Communication Framework within the Cloud worker roles. This was solved by using an incremental transfer approach to avoid inactive open connections. The bandwidth that is observed for data transfers is not uniform and fluctuates over time. While practically this may have limited impact, it makes reproducible experimental results more difficult. We noticed that the Azure .NET Storage Client API does not fully implement all capabilities of the Azure Storage Service, such as leasing. This required us to extend the client library. We also detected a memory leak bug in the .NET security/cryptographic libraries that has been reported.

Pillcrow: Scalable Graph Analysis on Cloud Platforms

Graphs are a common data structure that can be used to model relationships, and used to represent problems in social networks, flow networks and even genomics. Many domain problems can be reduced to standard graph problems with well-defined algorithms and solutions. Examples include community detection algorithms to cluster vertices that are tightly connected subsets and graph max-cut algorithm for genome haplotyping.

Betweenness centrality (BC) is an important graph problem for social network analysis, and seeks to find key vertices in a graph that appear on shortest paths between all pairs of vertices. BC is used in traffic networks, epidemiology and intelligence analysis, and is memory and compute intensive as a function of the number of vertices in the graph. There exist tightly coupled parallel solutions to BC but these rely on massively parallel shared memory infrastructure that is not readily accessible to most researchers. However, BC can be naively reduced to a distributed, loosely coupled version that is well suited for implementation on Cloud platforms. This approach operates on each vertex independently, and performs a breadth first traversal rooted at each vertex followed by a reverse traversal that aggregates BC scores.

As part of our work, we implement this loosely coupled BC algorithm on Azure to evaluate its performance for graphs of different sizes [2]. A Web Role instance provides a browser based interface for uploading a graph and submitting a BC job to be performed on it. This graph is saved to the Azure Blob Storage Service and a job message placed on an Azure Job Request queue. A *Manager* worker role instance picks jobs from this queue and creates independent tasks from it by partitioning the vertices in the graph. The tasks are placed in an Azure Task queue on which multiple Worker instances are listening. The worker loads the graph from blob storage and for each vertex in the task, it performs a BFS traversal rooted at the vertex and computes the local BC score for it. The BC scores for all vertices in the task are passed back to the Manager either using an Internal Endpoint or by uploading to a blob result file. The manager collects the results for all tasks and computes a global BC score for each vertex.

Our analysis compares the impact of number of worker instances, task size and the use of Internal Endpoint or Blob storage for result exchange on the total runtime of a job. Graphs of up to 100,000 vertices were evaluated on 16 small worker instances. For graphs with large number of vertices, we see a linear speedup as we increase the number of worker instances. Relative to a local machine, the virtualized Azure worker instances deliver only half the performance. However, use of Cloud resources provides ready access to a large number of workers and enables researchers to tackle large sized graph problems.

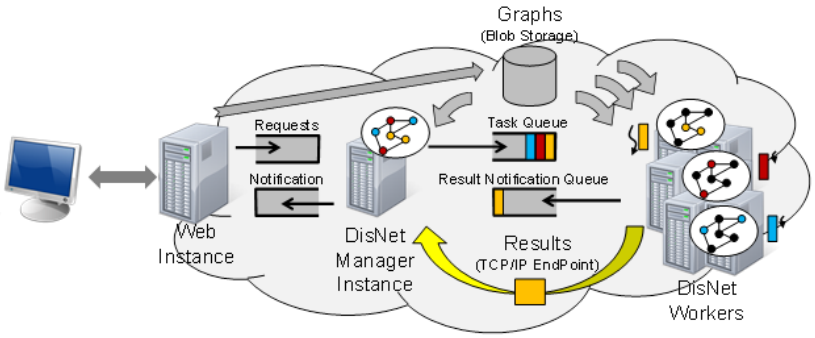
One of the short comings of this initial work is that the entire graph needs to fit in memory. Our ongoing work is examining the use of distributed models such as Microsoft Daytona MapReduce and Google's Pregel for trading of time against space. We are also investigating resilient models for graph applications

that survive failure of worker instances, and incremental graph algorithms for large graphs that are constantly being updated.

Funding for Cloud Resources

The decision on whether to acquire Cloud resources or physical hardware through agency funding will be a function of the applications that are being run and the users they serve. For highly performance driven application that operate on a tightly coupled model, purchasing and managing a rack with ~50 cores is a better model than Cloud resources. Similarly, small research groups familiar with managing a local cluster would be better off hosting their own hardware.

However, much of the research in our group deals with large scale problems rather than high performance problems. In such a scenario, on-demand access to a large number of virtual machine is more useful than round the cloud availability of a captive cluster. In addition, the overhead for installing and maintaining a local cluster is non-trivial unless strong system administration resources available at the local institution. Availability of platform services such as storage and programming abstractions such as .NET or MapReduce reduces the overhead of installing, monitoring and managing such services locally.



Betweenness centrality jobs and tasks running on Azure Worker and Web Roles

References

[1] Designing a Secure Storage Repository for Sharing Scientific Datasets using Public Clouds, Alok Kumbhare, Yogesh Simmhan and Viktor Prasanna, *International Workshop on Data Intensive Computing in the Clouds (DataCloud-SC11)*, 2011

[2] Performance Analysis of Vertex-centric Graph Algorithms on the Azure Cloud Platform, Mark Redekopp, Yogesh Simmhan and Viktor K. Prasanna, *Workshop on Parallel Algorithms and Software for Analysis of Massive Graphs (ParGraph)*, 2011

CIC (FRCC): TOWARDS A MOBILE CLOUD COMPUTING FRAMEWORK TO SUPPORT NEXT-GENERATION MOBILE APPLICATIONS

Department of Computer Science, University of Colorado at Boulder
Principal Investigator: Richard Han

The main thrust of this project is to build a mobile cloud computing infrastructure called SocialFusion to support context-aware mobile social applications. We have finished the preliminary design phase of SocialFusion and are now starting to build the context-aware infrastructure this Spring 2012. In parallel, we have started to research recommender applications and systems during Fall 2011. The results of this research on recommender systems will be integrated to operate on top of our SocialFusion infrastructure, i.e. context-aware mobile social applications require accurate and scalable recommendation algorithms and systems.

The research on recommender systems requires some fairly compute-intensive exploration, hence our use of the Azure cloud in Fall 2011. One of our PhD students at the University of Colorado at Boulder, Mike Gartrell, has been teaming with a researcher at Microsoft Research in Cambridge, England, Ulrich Paquet, to investigate a "Bayesian Treatment of Social Links in Recommender Systems", described in the following section.

A Bayesian Treatment of Social Links in Recommender Systems

Mike Gartrell, University of Colorado Boulder
Ulrich Paquet, Microsoft Research Cambridge

Challenge: Incorporating Social Networks in Recommender Systems

Recommender systems are increasingly driving user experiences on the Internet. This personalization is often achieved through the factorization of a large but sparse observation matrix of user-item feedback signals. In instances where the user's social network is known, its inclusion can significantly improve recommendations for cold start users (users that have rated a small number of items). There are numerous ways in which the network can be incorporated into a probabilistic graphical model. We propose and investigate two ways for including a social network, either as a Markov Random Field that describes a user similarity in the prior over user features, or an explicit model that treats social links as observations.

Implementation

In this project, the two models that we propose are implemented using a Bayesian probabilistic framework for matrix factorization. We performed experiments on a large scale, real world dataset obtained from the Flixster.com social network, and find that our models provide state of the art predictive performance. In order to perform our experiments, we compute predicted ratings for any user and item by performing Bayesian inference using a Markov chain Monte Carlo (MCMC) algorithm known as the Gibbs sampling algorithm.

For high dimensional latent user and item feature vectors, running our Gibbs samplers with the large-scale Flixster dataset is quite CPU and memory intensive. For example, running one of our models for a 20-dimensional latent vector space results in a run time of approximately 15-20 hours and consumes approximately 3 GB of RAM.

Since we needed to repeatedly run our models for a number of different parameter settings to perform our experiments, we quickly exhausted the available compute resources in our local lab environment. Therefore, we turned to Azure to perform our experiments. Using four quad-core Azure instances deployed as worker roles, we were able to complete our experiments in approximately one week.

Since we implemented our models using C# in a Windows environment, we were able to simply run our models on the deployed Azure instances without altering our model code. We did not encounter any difficulties with running on the Azure cloud for this phase of our research. We haven't yet used any of the Azure platform-as-a-service APIs.

SocialFusion

In the future, for our broader research on the entire SocialFusion infrastructure to support mobile social cloud computing, we anticipate that we will need to run some open source binaries and libraries such as Apache Web servers, mySQL, openCV, etc. We hope that Azure will readily support the creation of virtual machine instances for such open source software. Otherwise, we will have to port much of this functionality to run on Azure and .NET.

BetaSIM on the cloud

Michele Di Cosmo, Angela Sanger, COSBI

BetaSIM is a dry experiment simulator, driven by BlenX - a stochastic, process algebra based programming language for modeling and simulating biological systems as well as other complex dynamic systems.

Case study: p53[1]

SreeHarish Muppirisetty, PhD Student, COSBI Federico Vaggi, Researcher, COSBI
Attila Csikasz-Nagy, Principal Investigator, COSBI Alberto Inga, Professor, CIBIO

Models in systems biology are characterized by a wealth a parameters that govern the behavior of the system. For complex systems, most of the parameters cannot be experimentally measured, and must be fitted by means of minimization algorithms. Furthermore, the parameters of systems biology models often display 'sloppy' behavior2. To obtain information about the parameter space of a given model, and to obtain the parameters that give it the best fit to the existing experimental data, we require a high number of simulations. While individual simulations are very hard to parallelize, since each step is sequential and dependent on the state of the system at the previous time step, a lot of optimization algorithms can be parallelized3. In addition, because of the stochastic nature of BetaSIM, in order to provide a good fitting of the results, the number of simulations must be increased even more, resulting in a computing time of the order of magnitude of years for a single worker machine.

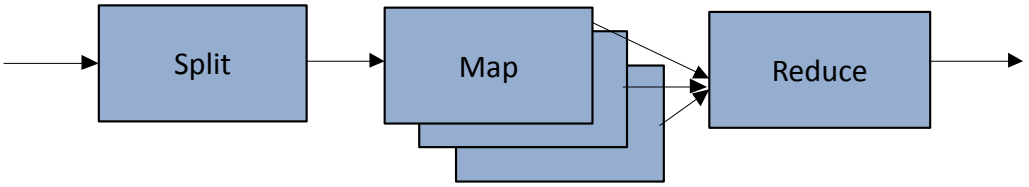
In our case, we are interested in the behavior of p53, one of the most important transcription factors in the genome. Our experimental collaborators in Alberto Inga's lab have developed a system to measure the transcriptional activity of p53 using a luciferase reporter gene. By introducing p53 in budding yeast, where no endogenous targets exist, we can measure the full activity of p53 in different conditions without worrying about downstream feedback loops, and characterize its ability to bind to the genome. We have developed a model that includes all known reactions between p53 and DNA, and are fitting this complex model using experimental data obtained in different conditions and using different mutants of p53. AzureBetaSIM will be used to simulate this model.

The implementation

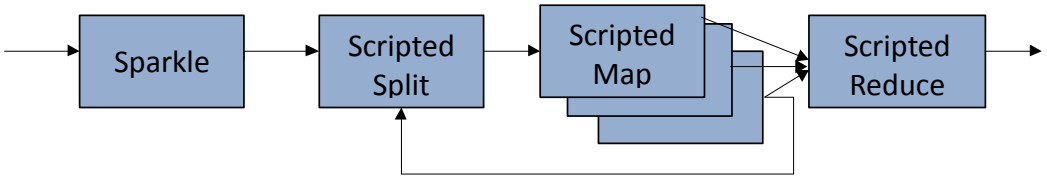
AzureBetaSIM is a data-flow driven parallelization of the BetaSIM simulator, based on the VENUS-C Windows Azure platform. BetaSIM and AzureBetaSIM are simulators for the BlenX language.

The aim of AzureBetaSIM is to provide researchers with the ability to quickly run (despite the length of the job queue) a large number of concurrent simulations and, in return, quickly gather research data. More importantly, this approach enables the execution of complex flows based on the aggregation of a large number of identical or slightly different models that, because of the stochastic nature of BetaSIM, will produce different outputs.

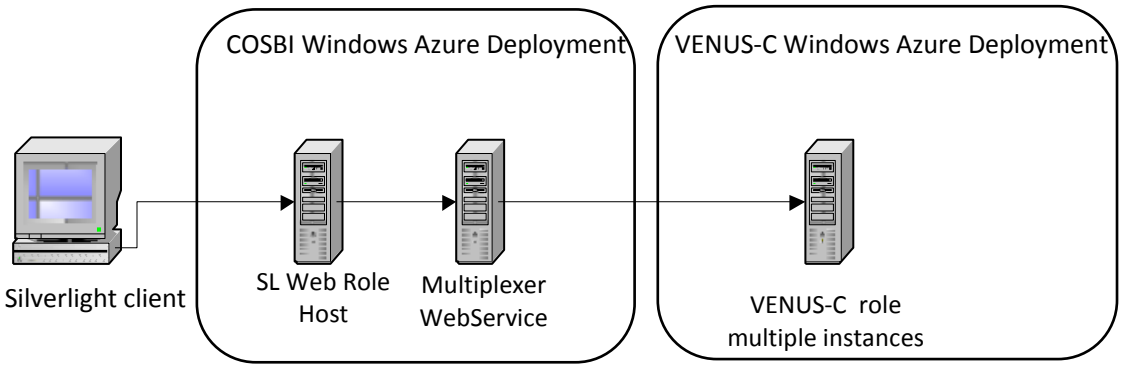
The VENUS-C framework, which can run on top of Windows Azure, provides a data-flow programming model based on jobs that run when input data is made available, and produce the input for the following job. The basic use case of AzureBetaSIM consists of a user uploading input data (a BetaSIM model written in BlenX); then specifying how many executions of that model are to run and how many differences (if any) should be introduced in these cloned executions; finally she instructs the cloud on how she wants to aggregate resulting data in order to have a sneak peek of what happened.



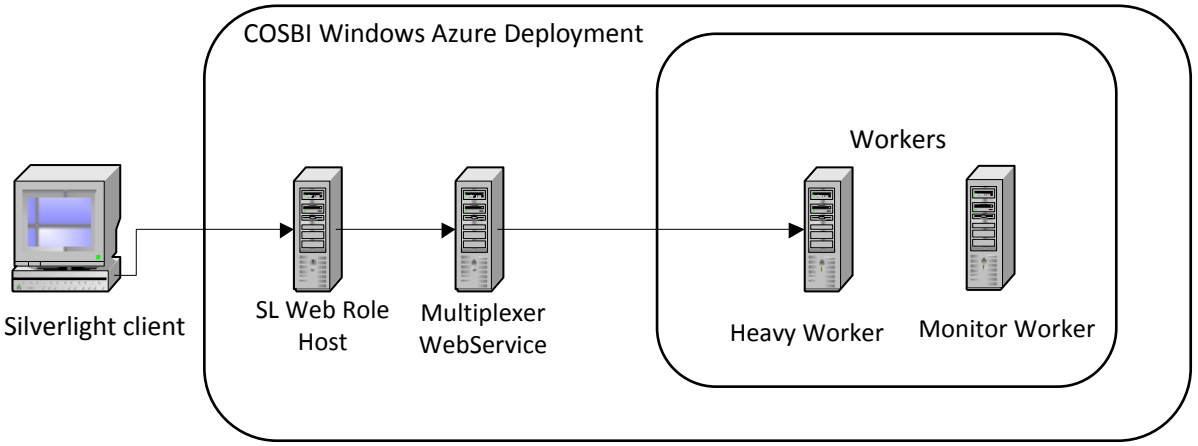
AzureBetaSIM extends the VENUS-C functionalities by giving the user the ability to change and define the data-flow programmatically by the means of simple Python scripts. In its' advanced use case, COSBI's scenario asks the user for a "sparkle" python script that will provide information about the following jobs. In this way, the user can alter the flow dynamically based on the previous outputs and ask the researcher remotely to continue, stop or alter manually the model before a new step. All jobs in this use case are scripted, and the scripts can be provided by the user or generated automatically by another script.



To provide the ability to read and alter input and output files easily, the DOM of common formats are made available to the script. In particular, BlenX input files (the model to simulate), CSV (Comma Separated Files) and SPEC files (proprietary simulation output files). As previously stated, our solution is based on the VENUS-C framework and consists of a role running a simple multiplexer webservice; a role running a simple web server hosting the Silverlight client application; and multiple instances of the VENUS-C role, which act as an STS server used for authentication, as a webservice for querying and submitting jobs, and as the main worker, altogether. All the simulations and scripts are run inside executable independent applications uploaded to the VENUS-C instances by an administrator. User data is stored on Windows Azure blobs only and is accessed through the multiplexer.



In respect to our objectives, we prepared a VENUS-C independent alternative solution which shows a different architecture, but the same programming model. In this case the deployed roles are four: the first two are the same as the previous case (web role and multiplexer); however, two worker roles are present: the "heavy" one is expected to be deployed with a lot of instances, being the one responsible for the simulations and the scripts execution, while the second one is responsible for updating statues and ensuring everything is running smoothly. Everything is hard coded in the two worker roles but the simulator, which resides in an independent executable file. User data is stored on Windows Azure blobs and tables and is accessed through the multiplexer. The VENUS-C framework data-flow paradigm subset used by our scenario is here replicated ad-hoc.



In both cases the simulations are considered atomically and run from the beginning to the end on the same machine.

[1] http://www.nature.com/nrc/journal/v9/n10/abs/nrc2730.html
[2] http://www.lassp.cornell.edu/sethna/Sloppy/index.html.
[3] For example, in a recent model of the EGFR receptor, Kholodenko et al used over 3 years of processor power to estimate the parameters of their model (http://www.nature.com/msb/journal/v3/n1/extref/msb4100188-s7.pdf) (full paper Ref here: http://www.nature.com/msb/journal/v3/n1/full/msb4100188.html).

Technical challenges

The main technical challenges we encountered were:

- Database storage on the cloud is different from the one we are used to: databases are not relational and querying and paging is available on a limited number of fields. Data structure is definitely something that needs attention during the architecture phase of the project.
- The cloud is less stable than a local server or HPC: machines may shut down unexpectedly because of upgrades or because of some unmanaged exception in other processes, plus non-relational DBs require architecting and coding effort to ensure transactional operations in order to preserve consistency - your code may be shut down at any moment.
- Choosing the best programming model for your problem is something to be done in the initial steps of the project, as it strongly impacts the architecture of the solution. In our case, we spent a few weeks researching the best model and found that a data-flow driven model was the best fit for our map-reduce scenario, while giving enough elasticity to enable scripting.

Real benefits of the cloud vs. high performance computing

During our testing, we verified the following major benefits of the cloud infrastructure versus our existing 16 nodes HPC:

- Scalability is a plus for Azure: the usage statistics of our application shows that users running simulations access resources for a limited and concentrated amount of time. Zeroing the costs of idle machine is a significant cost benefit.
- No additional maintenance and administration costs are a plus for the cloud: often nodes' software needs to be synchronized and local storage needs to be cleaned-up due to dirty jobs' trash. This requires time from the Infrastructure team and, more importantly, this time needs to be scheduled, so it is not rare for days to pass before the work is scheduled and completed.
- A theoretically infinite number of machines enable us to approach different scientific problems that require the simulation of large numbers of very similar input models. The stochastic nature of our simulator requires simulating the same input multiple times, so with "unlimited" cloud resources, researchers can gather and analyze larger amounts of data and investigate new sets of problems that, with the usage of an HPC, were not possible.

From a cost and scalability point of view, we would definitively consider requesting funding for cloud resources. Also, the cloud enables us to explore different classes of problems opening new doors to research.

Additional contributors

Angela Sanger, Applications Manger, COSBI
Sonia Martinelli, Infrastructure Manager, COSBI

RUNNING FIRE RISK ESTIMATION AND FIRE PROPAGATION MODELS ON WINDOWS AZURE

Nikos Athanasis, University of Angean

Fire risk estimation and fire propagation models are based on a set of biophysical input data to calculate the probability of wildfire ignition and fire behavior potential. Both problems are taking place in space and time. Therefore, the more often input data change the more often simulation and computation models should be applied to calculate output data. Our scenario addresses emergency management aiming to assist on wildfire early warning, fire control and civil protection. The main objectives focus on adapting and migrating applications and models for prediction of fire risk and fire propagation simulation in the Azure cloud computing infrastructure. These models require a large amount of input data, create a larger amount of output data and have a high processing complexity as a result of increased spatial and temporal resolutions of the input/output data. Output maps are accessible via a web interface by the end-users. Cloud computing perfectly fits into our needs because it provides for reliable data storage and offers scalable workload of the computations.

THE IMPLEMENTATION

Azure is used as the cloud platform for hosting and storage. End-users have access to a large set of output maps that visualize fire risk and fire behavior. End-users interact with the system through a Silverlight-based interface that uses Microsoft Bing Maps for overlaying the output maps.

One of the main challenges of working with Azure was to translate the source code for the fire risk execution from an older Visual Basic code to .NET (VB and C#) and to create the appropriate solutions that were successfully ported into the Azure platform. This transition also excluded dependencies from third parties vendors (e.g. ESRI's ArcGIS scheme) that were used for processing and storing purposes. This process required the modification of the input and output data structure.

We started by decomposing our scenario in three use cases. The first use case is for the calculation of the fire risk based on forecasted weather data. It provides the expected fire risk on an hourly basis for the next 120 hours (i.e. for the next 5 days) and generates one output map for each hour (Fig 1, left). Besides the everyday calculation of fire risk maps for the next 120 hours (5 days interval), the improved model now includes the calculation of weather maps for the same time interval. Maps about temperature, relative humidity, precipitation and wind speed improve the overall functionality and assist on an enhanced early warning, control and civil protection decision support system. In a serial way, this computation required many hours before, while now the computation time is reduced dramatically by having a large set of virtual machines computing a corresponding set of the output maps. The second use case runs once every hour and calculates the fire risk based on real time weather data. While use cases one and two run at specific time periods, a third use case is based on the users' interaction to calculate the expected propagation of the fire propagation front for a given fire ignition point. Users define the simulation start date/hour and the corresponding end date/hour as well as the ignition point(s) of the fire(s) (Fig 1, right). Each fire is simulated by a different machine in

the cloud, thus reducing the computation time dramatically by processing in parallel multiple fire behavior simulations. Fire propagation fronts can be calculated not only based on forecast weather data but also based on archive weather data. This is especially useful to simulate fire fronts of fires already ignited in the past. When the VENUS-C project is completed, firefighting personnel, emergency crews, and authorities will be able to design an operational plan to encompass the forest fire, pinpointing the best ways to put out fires within the proper resources and time.

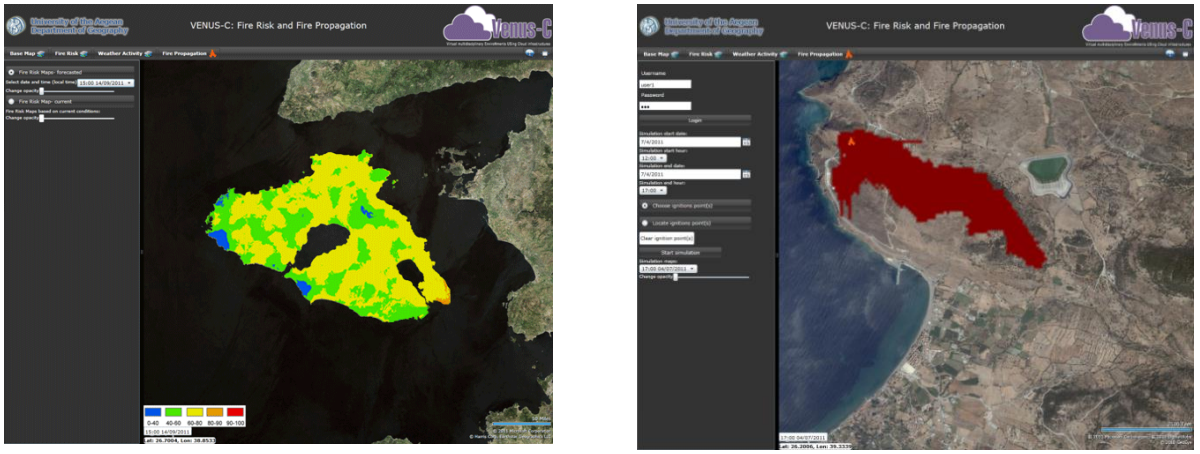


Fig 1: Visualization of fire risk maps and fire behavior simulation

Every one of the three use cases of our VENUS-C Civil Protection and Emergencies scenario corresponds to a binary executable compiled in the .NET environment. An Azure WebRole hosts the Silverlight application while worker instances (i.e. virtual machines running in worker role) processed the binaries and stored the output results in Azure blobs. Workers started their execution by fetching jobs that were stored in an Azure queue (Fig 2).

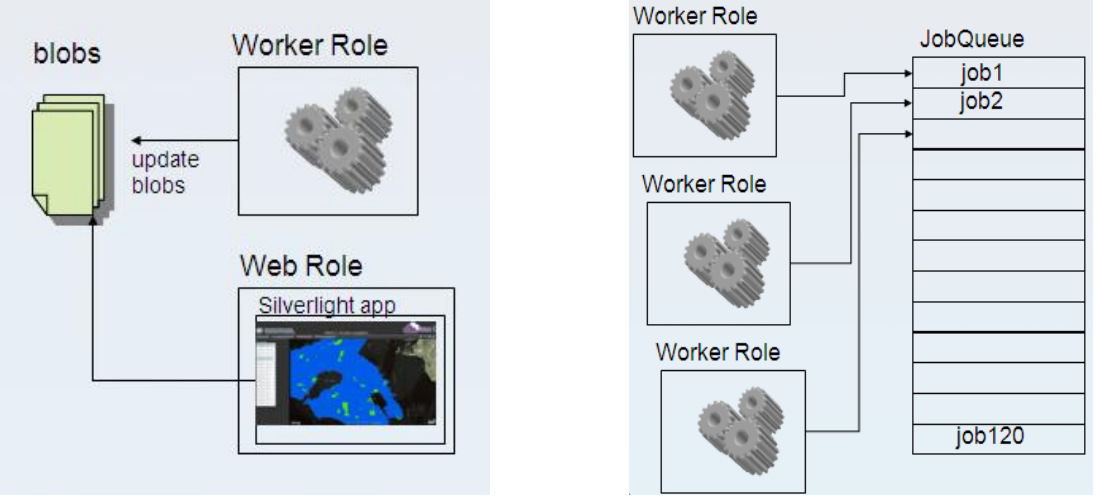


Fig 2: Job execution for fire risk estimation and fire propagation

The number of the allocated virtual machines is not constant, but changes according to different workloads. Thus, elasticity rules are used to increase/ decrease the number of machines. This is very essential, since forest fire protection is mainly needed only from June to September during the regular fire seasons of the northern hemisphere ecosystems. By using scalability rules, authorities can respond to different workload situations that have to deal

with on a day-to-day situations; e.g. unpredictable (fire propagation simulations during a fire event) and/or predictable (daily fire risk estimations).

The instant availability of processing power along with the cost effectives of the cloud-based solutions are key elements that are better suited to our scientific problem. Fire risk estimation and fire propagation algorithms do not require extremely high computational speeds via parallel processing but only a certain amount of processing and storage at specific time intervals during the summer fire seasons. Thus, elasticity is a key aspect of our approach which is better suited in a cloud-based solution than a supercomputer-based solution. For non IT end users, the cloud solution may be more user friendly than supercomputing in which considerable technical experience may be required.

Regarding the cost of the cloud solution, it is more preferable to request this funding from internal or external funding agencies rather than purchase new hardware with the same amount. Fire risk and propagation applications are useful during the fire season that it may extend only for 4 or so months. Therefore, new purchase of hardware is to remain unexploited for almost 2/3 of the year. Furthermore, the maintenance and administration costs will be significantly high. Wildfire management authorities would prefer to invest their resources to firefighting equipment and services rather than in computational resources on premises.

A STRUCTURAL ANALYSIS CLOUD SERVICE IMPLEMENTATION THROUGH AZURE AND VENUS-C

José M. Alonso, Pedro de la Fuente, Vicente Hernández, Pau Lozano
Grid and High Performance Computing Research Group – Institute of Instrumentation for Molecular Imaging – Universidad Politécnica de Valencia – SPAIN.
Contact: José M. Alonso, jmalonso@dsic.upv.es

Introduction

Structural analysis of buildings, or civil engineering structures, is the process to determine the response of a structure to different prescribed applied loads. This response is usually measured by establishing the stresses, tensions and displacements at any point of the structural elements.

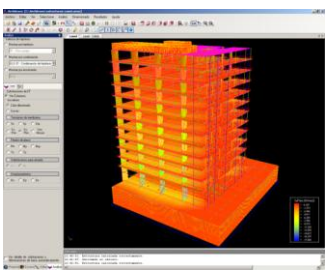
Accuracy in structural simulations is a key factor, where 3D realistic structural models together with accurate and numerically efficient methods of analysis must be applied. In order to find the most appropriate structural design, according to distinct criteria of safety, economic limitations or construction constraints, a large amount of different configurations have to be simulated, following a trial-error process. Each of these alternatives is defined by the structural engineer varying the size of the structural elements, the material that composes them (concrete, steel, etc.), or the external loads applied. For example, the Spanish Earthquake-Resistant Construction Standards (NCSE-02) demands a building to be analysed with at least five different representative earthquakes. Then, all these structural alternatives must be analysed and the results are interpreted, maybe giving place to a new iteration in this trial-error scheme. Obviously, this situation largely increases the computational cost of the problem.

Therefore, the realistic 3D structural dynamic analysis of large scale structures can demand an important computational power, give place a huge volume of data and become one of the most time consuming phases in the design cycle of a building or a civil engineering structure. For this reason, this analysis has been traditionally solved by introducing a variety of simplifications (unsuitable for complex structures) in order to reduce the problem size and the volume of the data, and obtain the results in reasonable simulation times.

Architects and structural engineers need thus powerful software applications able to simulate efficiently the accurate response of the structure. High Performance Computing (HPC) techniques provide powerful numerical and programming tools to develop applications able to simulate, efficiently and in a realistic way, large dimension structures, in very reasonable response times. However, commercial applications just offer traditional approaches, computing sequential structural analysis on the user's local machine. As a result, the size and the complexity of the structure to be analysed, the type of structural analysis employed and the total number of the different structural solutions or even earthquakes evaluated are limited by the performance of the computational resources used by the users.

With the purpose of overcoming these limitations, our research group, in collaboration with the Department of Continuum Mechanics and Theory of Structures (DMMCTE) of the UPV, has developed Architrave® (http://www.architrave.es), an advanced software environment for the design, 3D linear static and dynamic analysis and visualisation of buildings and civil engineering structures. Architrave is composed of these three different and independent components, although interacting among them:

- The Design Component: An interactive GUI AUTOCAD-based application where the user can draw the model and define the structural properties and the external loads.
- The Analysis Component: An interactive GUI application where the user can modify the structural properties and the external loads, analyse the structure and visualise the results.



- The Structural Simulator: A batch HPC application used by the Analysis Component to simulate the response of the structure by means of the Finite Element Method.

Notwithstanding, studios for architecture and engineering rarely own parallel platforms to execute an HPC-based application. Thus, the users install and run Architrave in their personal computers, and the time spent on the calculations by means of the Structural Simulator component depends on the performance of their machines.

Fortunately, Cloud Computing technology has emerged as a solution for the computational requirements of organizations, enabling the usage of non-owned remote resources which delivers enough power and storage space to satisfy the computational and disk requirements of the resource-starved dynamic structural simulations of high-rise buildings. Moreover, Cloud technology allows sharing not only computing power, but also another kind of resources such as storage space, data and even software packages. As known, Windows Azure is one tools used to build, host and scale web applications in the Cloud. Moreover, Venus-C, whose main target infrastructure is Windows Azure, aims at providing components and software libraries to ease the process of porting applications to the Cloud.

The main aim of this work is to develop an innovative Cloud system that offers on-demand high performance static and dynamic structural simulations to the structural community. All the simulations in the Cloud are governed by Architrave. The system is composed of:

- The Architrave Analysis Component: The GUI application in charge of submitting the simulations in the Cloud and visualise the results.
- A high throughput and reliable Structural Analysis Cloud Service, responsible of performing remote simulations over a Cloud infrastructure based on Venus-C and Windows Azure.

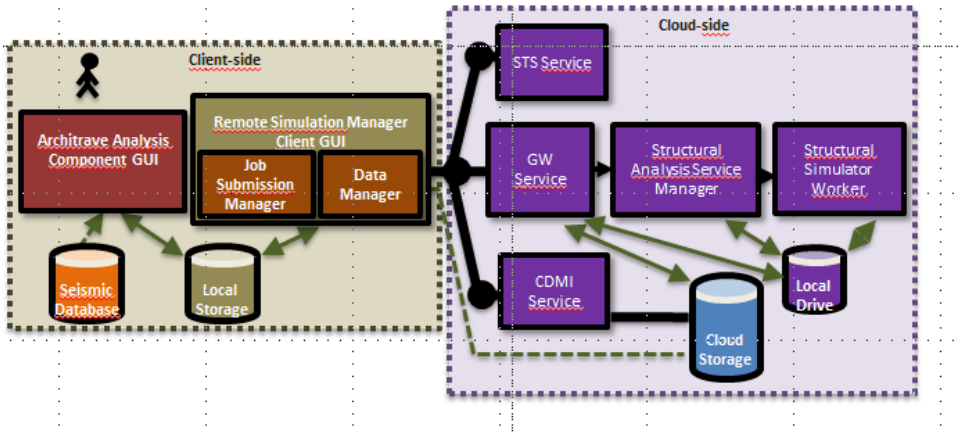
Architecture of the System

On the client-side, the local or remote simulations are defined exactly in the same way from the Architrave Analysis Component, the GUI application where the structural properties can be modified and the response of the structures is visualized. This application is now also composed of a Remote Simulation Manager Client, a comprehensive GUI that includes:

- The Job Submission Manager: The component that submits the simulation and checks its status (submitting, pending, running, finished, failed, downloading, downloaded) and informs the user.
- The Data Manager: The module that uploads the input data and downloads the corresponding results.

On the Cloud-side, the Structural Analysis Cloud Service is composed of:

- The STS Service, in charge of performing the user authentication.
- The Generic Worker Service, which manages the execution of the remote tasks.
- The Structural Analysis Service Manager, which generates the input files needed by the Structural Analysis and launches it with the appropriate parameters and encapsulates the results.
- The Structural Simulator Worker, i.e. the HPC component of Architrave that executes the simulations in the Cloud.
- The CDMI Service, which uploads and downloads the data using the standard CDMI protocol.



More in detail, for each simulation, an input binary file is saved on disk by Architrave Analysis Component. This binary file is composed of the building geometry, the external loads applied, such as an earthquake, and the simulation parameters. The user is authenticated in the Cloud by means of the username and the password. If the user has permission to use the Cloud Service, the Remote Simulation Manager Client GUI reads the input binary file and the remote simulation is registered. Then, the input binary file is uploaded to the Cloud and saved in the Cloud Storage, and the job is submitted to the Cloud Service in the way of a .jsdl document.

For each client request, the associated input binary file is moved from the Cloud Storage to the local drive by the Generic Worker Service, and the Structural Analysis Service Manager is launched. This executable file reads the input archive and generates the input file needed by the Structural Simulator worker. Next, the Structural Simulator worker is executed with the appropriate parameters, depending on the type of analysis, defined by the user. As a result, the structure is analyzed and the output files are periodically generated and saved on the local drive. For each simulation time step, in case of a dynamic analysis with response along the time, all the multiple output files generated are encapsulated in just a single output file by the Structural Analysis Service Manager. The Generic Worker Service moves now these simulation results to the Cloud Storage. A notifications scheme is employed, and the client is appropriately informed with the status of the remote simulations. Thus, the Remote Simulation Manager Client downloads the output files to the local machine when the execution has finished or when enough results are ready to be retrieved. In this way, the remote simulation execution and the result retrieval phase are overlapped in time and, thereby, the whole time involved in the simulation is dramatically reduced.

The Structural Analysis Cloud Service is based on the High Throughput Computing model, since multiple clients will employ concurrently the service and lots of different structural solutions will be analyzed at the same time. The objective is thus to compute as many simulations per time unit as possible.

Advantages for the Structural Community

Thanks to this Structural Analysis Cloud Service, researchers will have available a huge number of computational resources to be on-demand employed and lots of simulations will be launched simultaneously. Thus, more structural experiments will be analysed per time unit, increasing the number of structures simulated and speeding up the research process. If researchers get more results in less time, they can get more and better conclusions in their research and reducing the time-to-publication.

The structural community will be able to solve large-scale problems, increase the complexity of the structure to be analysed, and carry out a larger number of realistic dynamic simulations without simplifications. In this way, the reliability and safety of the results obtained will be improved and new

structural problems will be tackled. Since the time spent on the design of buildings and civil engineering structures will be reduced, the engineering companies and the architectural studios will increase easily their productivity and volume of business.

Finally, there will be no need of acquiring software licenses in property (just pay per use) and expensive hardware for solving large-scale structural problems, and the users will not be worried about new software updates.

Dominic Price (Dominic.Price@nottingham.ac.uk)

About Horizon

Horizon is an inter-disciplinary research institute housed at the University of Nottingham and funded by Research Councils UK (RCUK), in collaboration with over 100 industrial and academic partners. Building on the Digital Britain plan, Horizon research focuses on the role of 'always on, always with you' ubiquitous computing technology. Our aim is to investigate the technical developments needed if electronic information is to be controlled, managed and harnessed — for example, to develop new products and services for societal benefit.

A Toolkit for Crowd-Sourcing Personal Data

As crowd-sourcing is a term that covers more than one single task, and there are many different workflows that a crowd-sourcing task can consist of, developing a single universal application that supports all types of crowd-sourcing is a daunting prospect. Instead, we propose that there are a number of generic elements of crowd-sourcing that can be drawn out and an infrastructure developed to support these. In particular, participant recruitment, registration and management are common to most if not all crowd-sourcing activities. Our proposed toolkit contains support for all these interactions.

As stated above, creating a universal application to support all crowd-sourcing activities is a huge engineering task. What we instead propose is a "marketplace" for crowd-sourcing activities (in essence, crowd-sourcing of crowd-sourcing). Our toolkit will provide an infrastructure in which crowd-sourcing modules that support a particular activity can be developed and then different crowd-sourcing workflows constructed by combining different modules. These modules can then be shared with other users to facilitate their crowd-sourcing activities. This enables non-programmers to reuse existing modules in creating new crowd-sourcing applications.

Architecture

At the heart of the toolkit is the crowd-sourcing factory (see diagram below). This is an application running in the cloud, providing the front-end interface for crowd-sourcing administrators (the user group which requests crowd-sourcing activities) and developers (the user group which develops crowd-sourcing modules) to log in and create crowd-sourcing modules and create an activity from selected modules. Once the workflow has been defined, the factory generates a crowd-sourcing instance, a separate application that is sandboxed from the factory and all other crowd-sourcing instances. This instance contains all of the necessary functionality for recruiting and managing crowd-sourcing participants as well as some storage for storing the results of the activity to be retrieved by the administrator once the activity ends.

Personal data is sourced from a participant's Datasphere: the crowd-sourcing instance knows the data it must request and talks to the participant's *catalog* to determine where that information is kept. The catalog authenticates the request and notifies the datastore that the crowd-sourcing application is allowed to access it.

Supercomputer Vs. Cloud

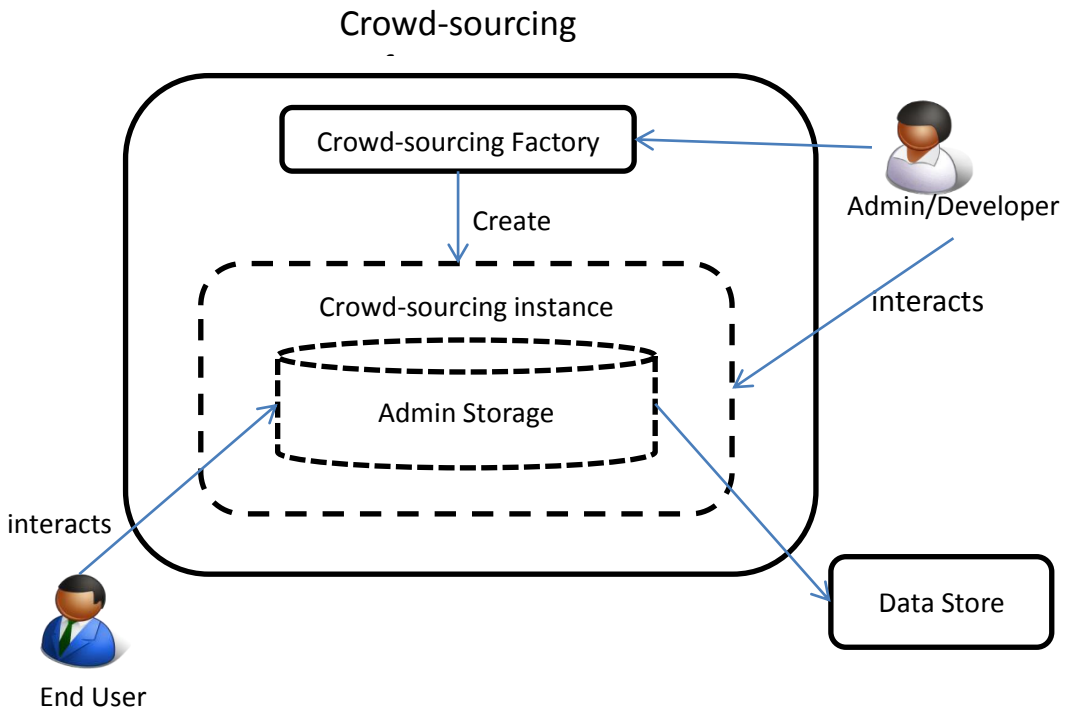
In our research, cloud-computing is far more appropriate than using a supercomputer as we are interested in the distributed nature of computing in the cloud and it's relation to the digital economy. A supercomputer would not offer us the flexibility we require.

Infrastructure as a Service Vs. Owned Hardware

At the outset of the Horizon research institute, we made the decision that we would not purchase any server hardware infrastructure. We believe that cloud-computing is vital to the future of the digital economy and are focusing our research efforts there.

Challenges

By far, the biggest challenge that we have had in working with the Azure platform has been in documentation. There is a lot of folk-knowledge on the Web but a lot of it is out of date. A particular example is setting up diagnostics within a Web or Worker role. The second biggest challenge has been in learning a lot of new technologies at once as many of us in Horizon were new to the Dot Net platform.



PORTING BIOINFORMATICS APPLICATIONS IN AZURE THROUGH VENUS-C

Abel Carrión, Ignacio Blanquer, Vicente Hernández

Grid and High Performance Computing Research Group – Institute of Instrumentation for Molecular Imaging – Universidad Politécnica de Valencia – Spain.

Contact: Ignacio Blanquer, iblanque@dsic.upv.es

Introduction

With the recent changes in the DNA sequencing technologies, the volume of data related to the study of biologic sequences has increased exponentially. This growth is not only consequence of the speed-up in the sequencing techniques, but also on the reduction of costs [1], which has ended up with a situation in which processing costs are larger than sequencing costs [2]. Examples are the Genome 10K[2] project, which aims at obtaining the genome of several individuals from each one of the 10,000 species of vertebrates selected, or the "Medical Genome Project"[3] in Spain, which is collecting genetic samples of patients suffering from rare diseases at a rate of 30 genomes per week. It is therefore crucial to find efficient and economic techniques to provide computing power to bioinformatics analysis centers. In this sense, in the last years many HPC and Grid adaptations of conventional tools have been made.

Despite of the advantages that intensive computing adaptations provide, many of these approaches have not achieved a wide impact. One important reason is the learning curve required for using high-performance or distributed computing infrastructures. This has been minimized in some cases by the use of portals. However, portals normally provide a limited set of options and are not easy to integrate with existing workflows, which directly use the native interfaces of the tools.

Our approach was to prototype and develop a client tool that could provide the same interface as the conventional bioinformatics tools used in local computers, but linking them to a powerful processing service on the cloud. Therefore, once the tool is deployed, biologists used to work with Command-Line-Interface tools (such as BLAST, bwa, fasta, bowtie, BLAT, ssaha, etc.) will be able to use the client interface of our tool without requiring additional knowledge, and they could seamlessly integrate the new tool in the existing processing pipelines they already use.

Architecture

In order to achieve the previous objectives, we used Venus-C components and Windows Azure. Venus-C aims at providing components and software libraries to ease the process of porting applications in the cloud. The main target infrastructure of Venus-C is Windows Azure. In our approach, we use VCDMProxy and Generic Worker components from Venus-C. VCDMProxy is used to upload and download the data using the standard CDMI protocol and Generic Worker deals with the execution of tasks.

2 http://www.genome10k.org
3 http://medicalgenomeproject.com

Our approach has been to develop an architecture-agnostic client, exposing a Command Line Interface (CLI) that mimics the exact behaviour of conventional bioinformatics applications. This CLI will connect to an infrastructure specific service that is deployed on Azure. Our first target has been BLAST.

BLAST searches for homologies (similar sequences) of proteins and nucleotides in an annotated reference database. Matches could be inexact, since natural genetic variability, mutations and sequencing errors do exist. BLAST takes as input a reference database (on the order of GBytes) and a set of input sequences (on the order of MBytes) and performs a comparison based on a Smith-Waterman seeding and a set of biologically-sound heuristics.

The CLI client uploads the input data to the cloud storage if necessary (e.g. the reference database may have been already uploaded in previous studies), using VCDMProxy. The use of VDMProxy hides the particularities of the different cloud storages that could be used in Venus-C. The processing service exposes a set of basic web services that enables the client to submit a processing request.

BLAST service splits the input sequences into chunks and submits one task per chunk to the processing roles of a Venus-C Generic Worker deployment. Generic Worker deals with the synchronisation and restarting of processing instances, as well as with the staging of data to and from the cloud storage. Authentication is performed through the STS service provided by Generic Worker. Figure 1 shows the architectural design for the application.

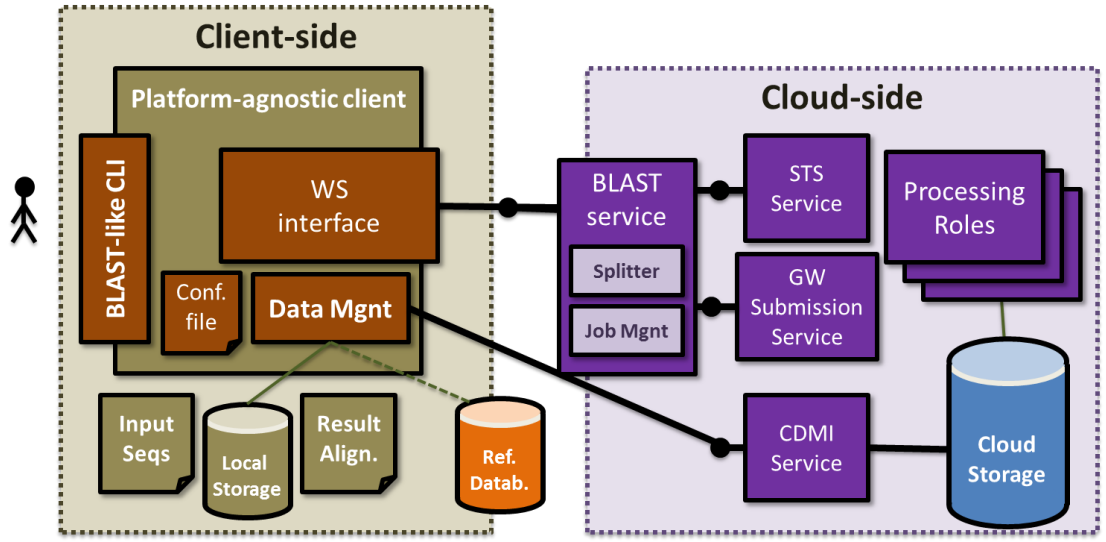


FIGURE 3: ARCHITECTURAL DESIGN OF THE BIOINFORMATICS APPLICATION

By the usage of concurrent instances, processing is performed in parallel and global response time is reduced. Final results are grouped in a single file after a postprocessing that computes the global results for the different output indicators, as well as groups in a single file the sequences matched. Finally, client is notified and results are transferred back through VCDMProxy. Client application remained in the foreground, waiting for the completion of the process, mimicking the behaviour of conventional tools.

Overhead of transferring input files and output results is minimized by the use of compressing (input and output are text files experimenting a good compressing ratio) and reference database is reused in different experiments.

Results

In order to validate the prototype, a large experiment using the Sargasso's Sea metagenome (800 MBytes) and comparing it with respect to a filtered version of the Non-Redundant GeneBank database that only contained samples from prokaryotes (total size around 1.3 GBytes). The algorithm used was blastx with an expected value of 0.01. The estimated execution time in a conventional sequential computer is on the order of 1.3 CPU years.

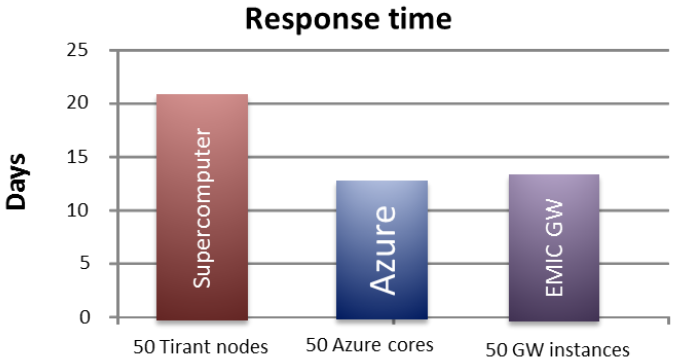


FIGURE 4: RESULTS OF THE PROCESSING OF THE SARGASSOS' SEA METAGENOME WITH RESPECT TO THE NR NCBI DATABASE.

Early results (shown in figure 2) obtained show a good speed-up on the order of 36 with 50 Windows Azure instances. The complete experiment was performed in slightly more than 13 days. The overhead of using Venus-C components with respect of directly using Windows Azure is lower than a 3%, which is neglectable, considering the advantages on the implementation. Comparisons with clusters and supercomputers [3], show that this approach provides very effective results.

The use of a separate and multiplatform client interface improves strongly the usability of the tool, especially on a discipline such as bioinformatics in which the presence of other platforms is quite strong. By including on the service the logic of splitting and scaling (the service checks the job queue when submitting new requests to decide if the number of workers should be scaled-up and decides how many instances should be switched off after the completion of an experiment), the deployment can be fined tuned to different supporting infrastructures.

References

[1] Editorial – Metagenomics versus Moore's Law, Nature methods, September 2009, vol. 6, num 9, pp. 623.

[2] Mihai Pop, Steven L. Salzberg, "Bioinformatics challenges of new sequencing technology", Elsevier, Trends in Genetics 24 (2008) 142-149, doi:10.1016/j.tig.2007.12.006.

[3] Daniele Lezzi, Roger Rafanell, Abel Carrión, Ignacio Blanquer Espert, Vicente Hernández, Rosa M. Badia, "Enabling e-Science applications on the Cloud with COMPSs", proceedings of the Cloud Computing Projects and Initiatives of the Euro Par 2011 Congress.

Jacek Cala, University of New Castle

Scientific goals: In the search for new anti-cancer therapies, the family of kinase enzymes are important biological targets since many are intimately connected to cell division and other important maintenance functions. New drugs that block the action of certain kinase enzymes are being sought by researchers at Newcastle University in collaborations with the chemists who make the new potential drugs, and the biologists who test them against the kinases. The scientists use a method known as Quantitative Structure-Activity Relationships (QSAR)[4] to mine experimental data for patterns that relate the chemical structure of a drug to its kinase activity. If a successful QSAR model can be derived from the experimental data then that model can be used to focus new chemical synthesis, and by creating QSAR models for more than one set of results, for different kinases, the new drugs can be designed to be selective.

Technical challenge: Calculating useful QSAR models requires a lot of processing resources. The process includes: calculating molecular descriptors, selecting features, building and validating models. As the input to the process we use data provided by ChEMBLdb — a database of drug-like molecules published and maintained by European Bioinformatics Institute.[5] Currently, the database offers over one million small molecules of which we extracted over 12,000 data sets that can be used in QSAR modelling. Each of the data sets may result in producing tens of different models. However, only a small percentage of the models found is valid and useful for further QSAR prediction. Therefore, our initial challenge is to find an efficient way to process the data from ChEMBLdb. Given a scalable drug discovery engine we can experiment with different feature selection and model building algorithms in the future.

Impact of results: The results will be published on a website, allowing scientists globally to predict the activity of molecules.

Architecture and Implementation

The basis of our solution is a set of workflows included in a legacy drug discovery system — the Discovery Bus.[6] Our previous work showed that the use of the Discovery Bus to build QSAR models is of limited scalability. The best performance achieved was using about 20 worker nodes and any additional workers did not significantly improve the processing speed. For this reason we decided to model the original drug discovery workflows using our e-Science Central platform (e-SC) and process them with e-SC workflow engines running in Azure. The architecture of the solution is presented in Figure 5.

[4] Hansch C., Leo A., Hoekman D.H. "Exploring QSAR: Fundamentals and Applications in Chemistry and Biology". American Chemical Society, Washington DC, 1995. 13.
[5] https://www.ebi.ac.uk/chembl
[6] Cartmell J., Enoch S., Krstajic D., Leahy D.E., "Automated QSPR through Competitive Workflow," J. of Computer-Aided Molecular Design, vol. 19, pp. 821–833, 2005.

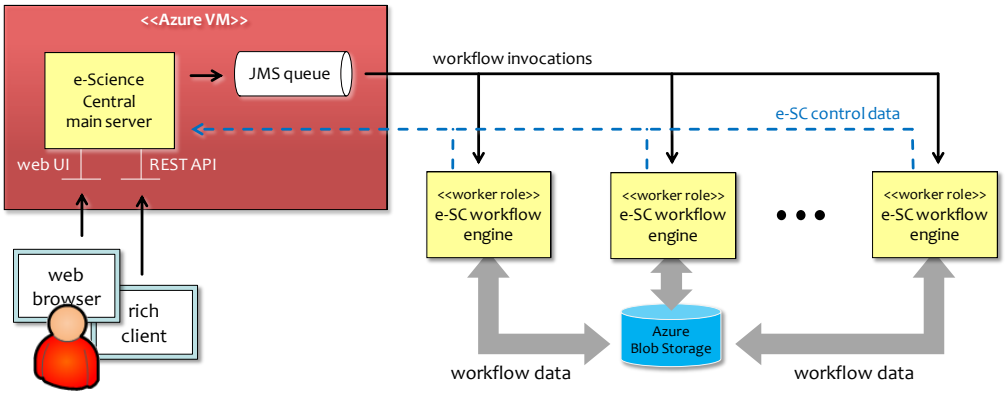


FIGURE 5. ARCHITECTURE OF THE DRUG DISCOVERY SYSTEM BASED ON E-SCIENCE CENTRAL AND AZURE.

Workflows are modeled and stored within the e-Science Central main server which is the central coordination point for all workflow engines. The server dispatches work to a single JMS queue from which it is fetched by the engines. For every input data set the system issues a single top-level workflow invocation, which then results in a number of sub-workflow invocations. All invocations form a large tree. The depth of the tree is constant and reflects the static structure of workflows, whereas the number of tree leaves depends on the input data and for the current workflow design results in about 10-35 invocations for a data set. Altogether a single input dataset generates from 50 to 120 workflow invocations.

The basic unit of work in our system — a workflow invocation — contains a number of tasks. After an engine accepts an invocation from the queue, it executes all included tasks. A task can be as simple as downloading data from blob storage or transposing a data matrix, or as complex as building a QSAR model with neural networks, which can consume over one CPU hour. The ability to model a basic unit of work as a workflow rather than a task enabled us to collocate many short tasks into a single invocation and so resulted in saving on communication overheads. But it also allowed the division of CPU intensive tasks into separate workflows so the work can be processed by many engines in parallel. This seems to be one of the important improvements over the previous approach used in the Discovery Bus, where the basic unit of work was just a task.

Moreover, each e-SC workflow engine is able to process a predefined number of workflow invocations concurrently. We found that the best resource utilization of worker nodes was achieved if a single engine could process up to four invocations at the same time when running on a 2-core machine. The value is a balance between processor and I/O intensive tasks included in our workflows.

Finally, a substantial speed up in processing time was observed after moving almost all data transfers between engines from the central server to the Azure blob storage. This showed good scalability of the Azure storage service and together with all other performance refinements enabled us to run drug discovery workflows with 150 execution nodes working in parallel and over 90% of ideal, linear speed up.

Supercomputer: We don't have access to a supercomputer and, therefore, it is difficult to say if and how the presented architecture would fit into a supercomputer processing model.

Costs and sustainability of results

The cloud computing model is a very good fit for the presented scenario. After initial processing of the whole ChEMBLdb database with all available model building algorithms, further efforts with QSAR analysis will require much less resource. The database is regularly updated with several hundred new input data sets every three months. This is less than 10% of the current database size, and so it will need just a fraction of the infrastructure to be effectively processed. Also, the development of new model building and feature selection algorithms can be tested on a relatively small fraction of the input sets and for only the most promising ones the whole input data will be applied. Again, introducing new drug discovery algorithms will need only a fraction of the resources because some of the results from previous invocations (namely the calculation of molecular descriptors) can be reused for any future model building.

Funding questions: We would certainly prefer to request funding for cloud computing than to buy hardware as we feel clouds give us a much more flexible research environment. Given the nature of our application (processing data that arrives infrequently), we will also get more processing power per dollar through cloud computing than through buying and managing our own hardware. We already do request funding for cloud computing resources in our research applications. For example, the Research Councils UK gave us $150K for cloud computing in the $20M Social Inclusion through the Digital Economy grant.[7]

The challenges

As our scenario is heavily based on e-Science Central and uses Azure in a limited way, we did not observe any real challenges when working with the Azure cloud. The major difficulty was the preliminary stage of the VM Role offering as we started using it soon after it was made available under the Azure Beta Program. The most troublesome issues were the lack of good documentation, the need for users to prepare the base OS image by themselves and inability to store and reuse customized OS images in subsequent deployments. For example, the process of running an Azure VM would be much simpler and quicker if users were able to deploy a predefined Windows Server 2008 R2 image from some Azure repository.

[7] http://www.side.ac.uk

D4SCIENCE BIODIVERSITY ON WINDOWS AZURE: DATA FOR SCIENCE

Marko Mikulicic, Pasquale Pagano, ISTI – CNR

The D4science e-Infrastructure aims at providing the scientific community with a wide range of tools and environments for managing research data, organized in Virtual Research Environments tailored for the specific needs of scientific communities, leveraging the computation power of the Grid, and starting to explore the cloud as an alternative model. In particular we focus here on problematic of the biodiversity community in the context of fishery.

A SCIENTIFIC CHALLENGE: AQUAMAPS

AquaMaps are computer-generated maps that show large-scale predictions of where a marine species may occur. These maps are generated using known occurrences of a species and its environmental preference with respect to depth, salinity, temperature, primary production and its association to land and in some cases, sea ice. The AquaMaps project website (http://www.aquamaps.org) provides a comprehensive description of its purpose and services.

The process can be roughly split into three phases:

1. Generating species envelope (HSPEN) based on models and occurrence points, possibly curated by end user scientists.
2. Generating probabilities of species occurrences in a georeferenced db (HSPEC), based on the Cartesian product of species envelopes (HSPEN) and an authority file of georeferenced data (HCAF) about sea parameters (temperature, salinity, etc).
3. Producing maps, GIS layers, and other geographical entities (such as transects) for selected species and group of species, for whole earth or user defined regions.

Each of these phases has different characteristics in terms of computational and IO costs, and can be parallelized to varying degree.

The georeferenced data about sea parameters is currently based on the Half-Degree Cell Authority File (HCAF), which has a 0.5 deg resolution, yielding 259200 cells, 178204 of which are ocean squares or partially ocean. Considering a species database consisting of 11k species, the Cartesian product grows to 2 billion records.

The next generation 0.1 deg resolution data is on its way, yielding 6480000 cells, 4455100 are ocean cells, increasing our data size by a factor of 25. The species list is also expected to grow, with a predicted size of the HSPEC db of 100 billion records.

We thus need a way to accommodate these growing computational and IO requirements.

THE IMPLEMENTATION

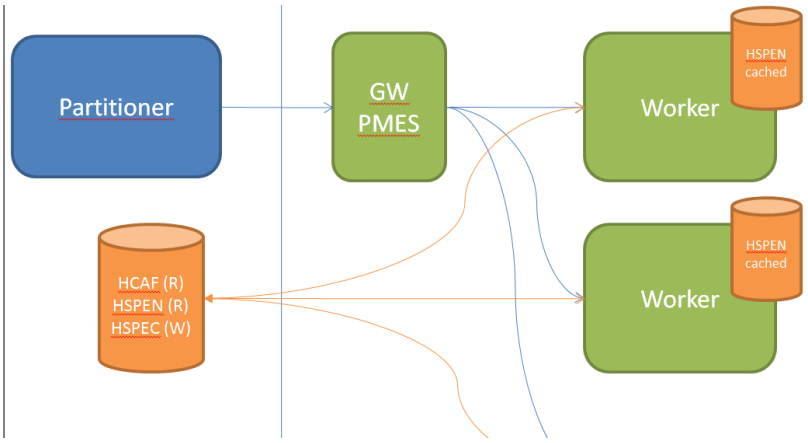
The end user entry point, the D4Science Portal is an existing component which runs on premises and offers other functionalities which are not intended to be ported to the cloud. The execution of the jobs has to support several backends as, among others, the Azure Cloud.

In order to achieve the portability goal we leverage the Venus-C cloud middleware, which supports execution of jobs on the Azure Cloud with the Generic Worker component.

The Generic Worker uses Azure Tables to coordinate job scheduling among a set of worker nodes and Azure Blob to store job executables and job input/output data. The details about working of the Venus-C Generic Worker are out of the scope of this document.

The D4Science e-Infrastructure's component for spawning biodiversity jobs on the cloud, called RainyCloud, divides the input database into partitions and sends jobs to Azure via the Venus-C enactment interface.

The HSPEC generation phase has been ported to Azure. The partitioner splits the HCAF table and spawns a job for each partition passing the whole HSPEN table to each job. The result data is copied back to the database on premises.



COSTS AND ALTERNATIVES

Our group already has a hardware infrastructure on premises, which is dimensioned to handle our normal load. We look for cost effective ways to handle our usage spikes. Buying additional hardware is not always a good solution, not only for the hardware cost but also for the system administration costs and the noticeable expense of time in hardware acquisition, also due to institutional bureaucracy.

We don't have access to a supercomputer. We did use the Grid infrastructure and thanks to the Venus-C middleware the Cloud approach it's quite comparable in terms of additional benefits from a technical point of view, albeit the overall business model is drastically different, offering more flexible capacity planning and direct billing.

LESSONS LEARNED

Our current bottleneck is due to the copying of produced data back to our legacy database system on premises. In order to better exploit the Cloud infrastructure we are improving our existing infrastructure to leverage on data stored directly on the cloud and accessed on demand; for example, the generated maps will be stored on blob storage and accessed directly by end users without the round trip on premises.

TARGETS ON THE CLOUD: A CLOUD-BASED MICRORNA TARGET PREDICTION PLATFORM

Theodore Dalamagas, Thanassis Vergoulis, Michalis Alexakis, "Athena" R.C.

Biologists used to consider proteins and DNA as *movers and shakers* in genomics. This has dramatically changed after the discovery, in early 2000s, of the key role played in gene expression by small RNA molecules, called **microRNAs** (miRNAs). miRNAs can completely silence proteins. They do so, by binding themselves to complementary sequences on mRNA transcripts, called **targets**. The knowledge of **miRNA targets** (i.e., which mRNA transcripts are targeted by a miRNA) is important for therapeutic uses.

There is a lack of high-throughput experimental methods for identifying microRNA targets. Thus, **computational methods** to predict targets have become increasingly important. Our team and the DNA Intelligent Analysis (DIANA) group of B.S.R.C. Alexander Fleming (http://www.fleming.gr/) have developed the DIANA-microT web application (http://diana.cslab.ece.ntua.gr/?sec=software) to provide access to computationally predicted miRNA targets produced by the target prediction methods proposed by DIANA group.

In this project, we will set up target prediction methods on the Microsoft Azure Platform to provide **real-time target prediction services.** Our aim is to meet the requirements for efficient calculation of targets and efficient handling of the frequent updates in the miRNA and gene databases.

A Scientific Challenge: Efficient Sequence Matching for miRNA Target Prediction

The first miRNA molecules were identified in 1993. Since then, there has been a dramatic increase in the number of miRNAs discovered and registered in **miRBase**, the searchable database of published miRNA sequences and annotation (http://www.mirbase.org/). On the other hand, there is a lack of high-throughput experimental methods for identifying microRNA targets. Thus, **computational methods** to predict targets have become increasingly important, and led to the experimental identification of many miRNA targets.

Our team and the DNA Intelligent Analysis (DIANA) group of B.S.R.C. Alexander Fleming (http://www.fleming.gr/) have developed the DIANA-microT web application. It provides access to computationally predicted miRNA targets produced by the target prediction algorithm proposed by DIANA group. Experimental results indicate that the DIANA method is among the top-3 in terms of prediction precision. Furthermore, since its original publication, DIANA-microT has been one of the most widely used web resources for miRNA analysis, counting more than 1,500 users per month based on Google Analytics.

In DIANA-microT, users can upload their own (newly discovered) miRNA sequences, and, then, compute the targets based on the DIANA target prediction method. In terms of efficiency, all target prediction methods are CPU-intensive methods, since they involve heavy sequence matching. However, biologists set the following requirements:

1. As researchers discover new miRNA molecules, the prediction methods must be **re-executed** in order to produce the targets of these new molecules.
2. Genome and miRNA sequences are **updated** (many times in the year) due to more precise biological experiments.
3. Because of the large computational cost, the target prediction algorithms are executed in order to predict targets in genomes of a handful of species (usually, at most 4). There is a need to search for targets **in much more genomes**.

Our aim is to **build the DIANA target prediction methods on the Microsoft Azure Platform** to meet the previous requirements effectively. Porting the application to the cloud, we will be able to adopt MapReduce solutions for the target prediction problem. We plan to design, develop and experimentally evaluate several MapReduce configurations for the efficient computation of miRNA targets. The outcome of the proposal will be real-time target prediction services available to users of DIANA-microT web application.

The Implementation

The goal of this proposal is to set up the DIANA-microT target prediction methods on the Microsoft Azure Platform. The DIANA prediction algorithm launches approximate string matching tasks to locate subsequences of the genome sequence that are similar to the given miRNA sequence. We call them **miRNA recognition elements (MREs)**. Based on the type and the number of such locations in a specific gene, the conservation profile of these locations in many species, and some statistics of the sequences, the algorithm determines a score for that gene. The greater this score is, the greater the probability that this gene is a target for this miRNA. The generation of targets is a CPU-intensive application.

The key scientific challenge of the proposal is to **design and develop MapReduce solutions** for the target prediction problem. Specifically, we will study how the DIANA target prediction method can fit the MapReduce execution model. Since the identification of the MREs of each miRNA on a fragment of the genome can be seen as an independent process, it can be implemented in a MapReduce function as follows: (a) each Mapper computes the MREs for one miRNA in a fragment of the genome, and (b) each Reducer aggregates the MRE scores for each gene in the genome, and produces the final target prediction score for this gene. Note that the fragments can be overlapping. Thus, a more sophisticated organization of input files that takes into consideration the overlaps may improve the performance even more. We plan to design and develop several MapReduce configurations for the efficient computation of miRNA targets.

To validate our methods, we are going to use the following datasets of RNA molecules and genes:

Unregistered miRNA sequences discovered by our users. The output of each execution will contain the targets of these miRNA sequences in the current versions of genome databases supported by DIANA-microT. Each supported genome database corresponds to one of the species covered by EnsEMBL database. A full list of EnsEMBL species can be found here:

http://www.ensembl.org/info/about/species.html.

EnsEMBL currently covers more than 50 species, and an entire EnsEMBL release currently maintains almost a terabyte of MySQL files (the files contain the genome sequences, annotations, and other

related data). The latest version of DIANA-microT includes four of the Ensembl genomes: Human (Homo Sapiens, ~3GB), Mouse (Mus musculus, ~2,7GB), Fruitfly (Drosophila melanogaster, ~175MB) and Caenorhabditis elegans (~100MB). Our intention for the Azure version of DIANA-microT is to cover even more species.

All registered miRNA sequences in miRBase. At the moment, miRBase (http://www.mirbase.org) contains more than 15000 miRNA sequences, accumulating ~20MB of data. The output of each execution will contain the targets of these miRNA sequences in the current versions of the supported genome databases.

GREEN PREFAB: CIVIL ENGINEERING HUB IN WINDOWS AZURE

Furio Barzon

Introduction

Green Prefab (GPF) is a product life-cycle management system for the building industry sector. GPF enables to design and construct a new generation of prefabricated buildings by means of collaborative software and industrial production.

GPF platform is composed so far by:

- a database devoted to classify prefabricated building components and architectural best practices
- collaborative workspaces for the life-cycle management of a green prefabricated building
- interoperable IT standard for 3D models sharing
- cloud computing tools to control building performances at engineering phase (eco-efficiency analysis, structural analysis, immersive rendering)
- a community of stakeholders.

Among the seven phases of the life-cycle of a GPF building (Digitalization, Design, Engineering and testing, Executive output, Production off-site, Realization on-site, Building automation tools), "Engineering and testing" deals with a community of engineers that need to work together in a 3D Master Model of the project to deliver a set of tests. Tests runs in the following fields: Immersive rendering, Structural Analysis, Eco-efficiency Analysis, Electrical plants, Heating and ventilation, Lighting, Production and Realization, Computation of Costs.

For the specific needs of Engineering area, GPF and partners have successfully ported prominent engineering softwares in different cloud infrastructures (MS Windows Azure, MS Generic Worker, COMPSs), specifically for Immersive rendering (LuxRender), Structural Analysis (Architrave), Eco-efficiency Analysis (Energy+, JEplus) softwares. An Engineering Hub website is under production, and since we are growing as a reference point for the building industry community more and more softwares will be ported in the cloud, enlarging the service offered.

The implementation in MS Windows Azure

The so-called Engineering Hub at Green Prefab is a web portal where professional engineers and researchers can easily access to cloud services for their analysis and needs.

GPF has implemented two different accesses to CLOUD, developed in .net programming language for Microsoft Windows AZURE, and Java for COMPSs (open source). The first solution has been conceived for business cases because it can deliver more protection on data, security, and larger computational expandability.

The general specifications of Engineering Hub components used are illustrated in the following table:

Component	Type	Platform
Generic Worker (GW)	Job Submission	Windows Azure
COMPSs	Job Submission	Linux
CDMI Proxy	Storage	Both
Security Token System (STS)	Security	Both
Accounting service	Accounting	Both

Introductory information about porting of LuxRender (tool for 3D rendering visualization) to the Microsoft Windows Azure using Generic Worker are introduced below. Similar porting is undergoing for new engineering softwares as for eco-efficiency analysis and structural analysis. LuxRender porting was successfully completed in late 2011.

Generic Worker (GW) is a SDK developed from Venus-C EU project and it releases some libraries for managing jobs in Microsoft Windows Azure. GW allows to create containers on Storage where service can upload user's files and install software for rendering. It allows to describe the job and submit it to Azure.

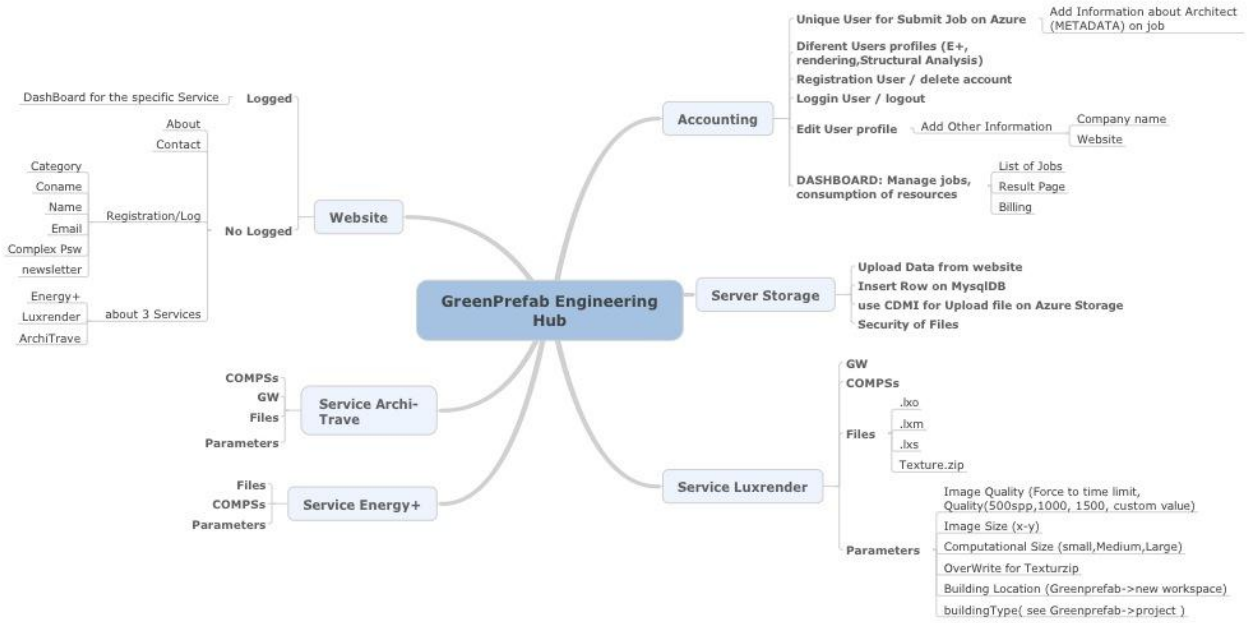


Fig. 1 Engineering Hub Components at Green Prefab system

Our users access to a web portal where they are able to management their jobs with a simple web interface: they can upload, submit, delete their jobs, and download results. At the current time the user set parameters for the execution of job and upload its project files on proprietary Server, these data are saved on mysqldb.

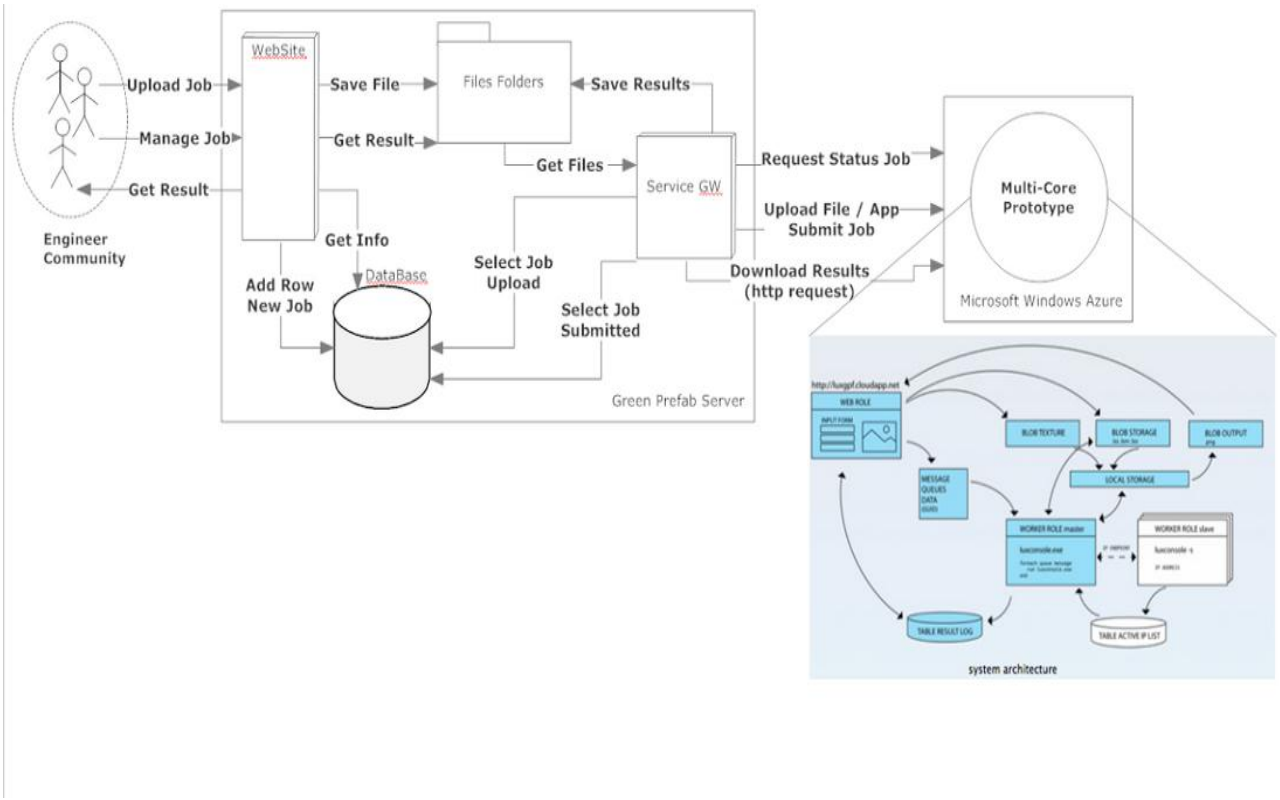


Fig. 1 Engineering Hub architecture for 3D rendering in MS Windows Azure

On Server side there is a service waiting for a communication about submitted jobs from Azure Cloud, after that is waiting for a “job-finished” response, then download the results.

In order to minimize the time spent for uploading files we zip all user's files, the software for rendering, and a .bat file. This last file is used to call LuxRender console (software for rendering) with some parameters like user's file or other configurations.

We use job descriptor in GW library to set job name, job ID, and input and output data location (the container where users upload files and where are output destination of files).

Then with GW we can create a “job management client” that allows to submit jobs to Azure (this is called “submission portal”). Our service uses submission portal looking for the job submitted and their status on Microsoft Windows Azure; if job has been completed we can download output files.

Fig. 2 - Screenshot of Job Result page in GPF website for a 3D visual rendering (LuxRender)

This module accelerates computation in rendering visualization through LuxRender.

LuxRender algorithm for rendering is “unbiased”, the status of the art in the field (in computer graphics, unbiased rendering refers to a rendering technique that does not introduce any systematic error, or bias, into the radiance approximation; this technique replies the effects of the physical reflection of light in the real world) The main problem for unbiased algorithms is that they required a huge amount of time for rendering single images or videos (from hours to many days of jobs, depending on the complexity of design, textures and lights). Cloud computing is going to significantly reduce that time.

An advantage of Engineering Hub solution in the CLOUD over a traditional supercomputer batch system is that Engineering Hub is a web portal that one can access over the internet, so the whole system can scale dynamically to handle requests.

Engineering Hub service doesn’t need to manage directly a supercomputer and its related location so cloud can be an advantage in this perspective too.

An usage plan involving professionals is going to start in early 2012 and will provide detailed information about time saving offered by the solution.

Further development ongoing

Green Prefab is porting prominent softwares for civil engineering running in cloud computing through easy-to-use interfaces and accounting systems: 3D immersive rendering visualization, structural analysis, eco efficiency analysis are the first third fields of applications.

Most of the results has been achieved thanks to the Venus-C project. Venus-C is a project funded under the European Commission's 7th Framework Programme drawing its strength from a joint co-operation between computing service providers and scientific user communities to develop, test and deploy a large, Cloud computing infrastructure for science and SMEs in Europe. Project will end June, 2012.

Please see reference in: http://www.venus-c.eu/Pages/Static/Mission.aspx

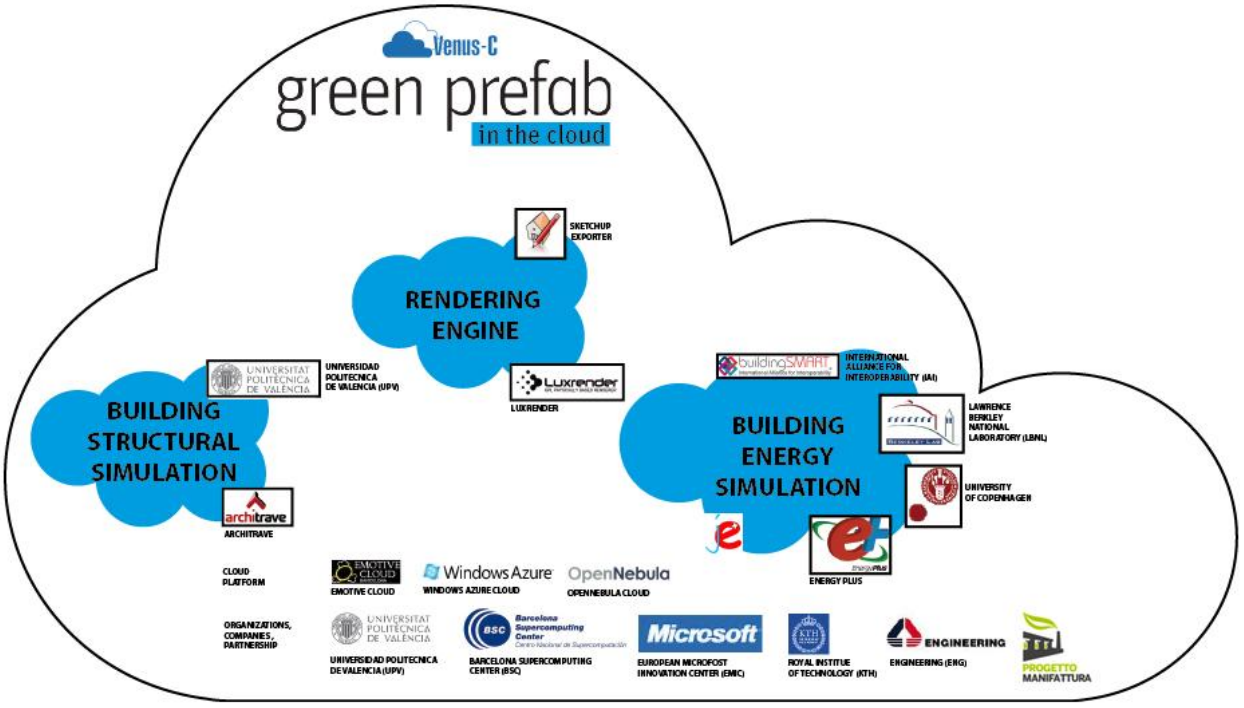


Fig. 1 - GPF prototypes for civil engineering analysis developed so far in Green Prefab

Currently, "energy efficiency analysis" and "structural analysis", beside other rendering softwares, are going to be implemented in CLOUD with similar porting. There are huge unsatisfied computational needs in civil engineers community nowadays, as the following extract from Prof. Emanuele Naboni witnesses.

> "The improvement of new and existing building's environmental performance as a contribution to mitigate climate change, to increase human wellbeing and to promote sustainable development is one of the major problems that the Architecture, Engineering & Construction (AEC) industry is facing. Despite the fact that sustainable design is being strongly promoted, its implementation in practice is still very difficult. Building Performance Simulation (BPS) is a key factor for facilitating sustainable design as it allows in-depth studies of buildings performance without the expense of constructing them, but it requires a big amounts of computational power in order to be implemented smoothly on a design process. In addition to this, parametric studies and multivariate optimization are emerging and promising approaches which allows architects to evaluate design alternatives in relation to performance factors. But the lack of sufficient computational power is the main issue that stops the AEC industry to increase their implementation in practice."

Emanuele Naboni, Associate Professor at the Institute of Technology - Royal Danish Academy - School of Architecture, Copenhagen (Denmark).

Company profile

GPF ties together an International network of real estates, producers of prefab elements, building contractors, designers, engineers, who work together to innovate the building industry. GPF is a product life cycle management platform for the building industry sector. GPF enables the production of a new generation of sustainable buildings by means of online collaborative softwares and industrial production method.

GPF has been incubated in Rovereto (Trento, Italy) at Progetto Manifattura green innovation factory (www.manifactor.it) since September 2011. Legal name is Green Prefab Italia srl (www.greeenprefab.com).

GPF company is a spin-off from two previous EU research projects where GPF was previously incubated within a private micro-incubator called Collaboratorio. The two projects are: MACE Metadata for Architectural Contents in Europe (taxonomies and social data) and Venus-C (cloud computing), see reference below for further details.

GPF is part of IAI International Alliance for Interoperability - Italian Chapter and founding partner of MDM Industrial cluster of ICT companies in Veneto Region (Italy).

Main customers at GPF are: real estate and public entities controlling building life-cycle, industrial prefabricators, designers.

OPTIMIZING DATA STORAGE FOR MAPREDUCE APPLICATIONS IN THE AZURE CLOUD

Radu Marius Tudoran, Gabriel Antoniu, Luc Bouge´ , Alexandru Costan

ENS de Cachan, IFSIC, IRISA, KerData-Team

(The following is abstracted from the IRISA report of June 2011)

APPLICATION STUDY: JOINT NEUROIMAGING AND GENETIC ANALYSIS

Joint genetic and neuroimaging data analysis on large cohorts of subjects is a new approach used to assess and understand the variability that exists between individuals. This approach has remained poorly understood so far and brings forward very significant challenges, as progress in this field can open pioneering directions in biology and medicine. As both neuroimaging- and genetic-domain observations represent a huge amount of variables (of the order of 106), performing statistically rigorous analyses on such amounts of data represents a computational challenge that cannot be addressed with conventional computational techniques. We propose to apply our solutions for a MapReduce framework to address these computational challenges.

Description of the Application

Imaging genetics studies the linking of functional MRI data with Single Nucleotide Polyphormisms (SNPs) data. In the genome dimension, genotyping DNA chips (the process of determining the genes of an individual by examining the individual's DNA sequence), allow to record several hundred thousand values per subject, while in the imaging dimension an fMRI volume may contain 100k-1M voxels (volumetric picture element). Finding the brain and genome regions that may be involved in this link entails a huge number of hypotheses. A drastic correction of the statistical significance of pairwise relationships reduces crucially the sensitivity of statistical procedures that aims at detecting the association. It is therefore desirable to set up as sensitive techniques as possible to explore where in the brain and where in the genome a significant link can be detected, while correcting for family-wise multiple comparisons (controlling for false positive rate).

Let (X, Y) be a joint neuroimaging dataset, i.e. a set Y of brain images after adequate pre- processing and a set X of genetic variables (single nucleotid polymorphisms SNPs and/or Copy Number Variants CNVs of genetic loci), acquired in the same population of S subjects. The dataset may also comprise a set Z of behavioral and demographic observations, such as psychological tests or age. In the tests performed this set is null. X is assumed to comprise n_v variables, e.g. one for each location in the brain image domain, while Y comprises n^g variables; typically $n_v \sim 10^6$, $n_g \sim 10^6$; when available, Z contains typically less variables (of the order of 10^2). These variables cannot be considered as independent, as there exist serial correlations between consecutive SNPs or CNVs, and spatial correlation within neighboring spatial locations in brain images. The main problem that we want to address is the detection of statistically significant links between the sets X and Y. This can be performed using univariate testing, i.e. testing the statistical significance of the correlation or equivalent association measure of all (x, y) pairs for (x, y) in XxY.

The initial data to be processed is given by a (X, Y) pair of a joint neuroimaging dataset. X represents a set of genetic variables. A variable has the dimension n_v , and there are n_s such variables, where ns gives the number of subjects. Considering that for this univariate testing we assume only the set of {0,1,2} values, we get X as a matrix containing one of these values. Y represents a set of brain images obtained with a functional MRI. Each image is divided into voxels and hence the set of values that forms each variable in Y represents a value for a region of the brain image. The number of variables is given by the number of subjects (n_s), thus forming a matrix of real values. After performing the necessary computations (the regression) the correlations between the two are obtained, giving a matrix of size $n_v \times n_g$ containing the p-values that represents the statistical significance of the association. Several regressions are performed, each giving a set of such correlations, and all these intermediate data must be stored in order to compare the values of each simulation in order to keep the most significant p-values. Thus taking into account that we have B matrices of doubles (8 bytes) of size $n_v \times n_g$, the amount of intermediate data that must be stored can reach (80 petabytes =8 bytes $\times 10^4 \times 10^6 \times 10^6$).

THE IMPLEMENTATION

Several computations for the p-values must be performed on random shuffles of the initial data. This computation cannot be performed on standard equipment. However, the problem can be run in parallel because the computation can be slice along many dimensions (neuroimaging, genetic or permutations). For the case in which the parallelization is done with respect to permutation, each worker will take the same initial data, shuffle it and perform the regression between the 2 sets of data (X, Y) to obtain the correlation between them. The regression from computational point of view represents a series of matrix operations, having as result a matrix. All these operations can be simply considered as the job of a mapper if we parallelize the application with the MapReduce programming model. Hence each mapper will perform a regression and will generate as the intermediate output a matrix. The final step, which would be the reducer, must perform a filtering on these values. Therefore the reducer will take all intermediate data, merge them and select the values within a threshold interval. An example of parallelizing this application with respect to the MapReduce model is shown in Figure 1, with 3 mappers and 1 reducer. B shuffles must be performed, thus the maximum parallelization factor for the map phase is B with respect to the parallelization according to the permutations. The reducer parallelization is not so direct. Basically we can launch more reducers that performs filters on intermediate data, in parallel, but all results should be combine to a unique one. We propose an iterative reduce phase and this will be detailed in below.

The public clouds like Azure or Amazon EC2 (Elastic Compute Cloud) offer storage services like Azure Blobs for Azure or S3 for Amazon. These type of storages made up of data nodes are in remote locations with respect to the computation nodes. An existing API based on HTTP protocol permits any application, executed on a cloud computation node or outside the cloud, to perform data operations (put/get). The benefits of using these storage systems are data security, easy access from anywhere and reliability. However, when it comes to data-intensive applications, like some of the scientific ones, the distance between computation nodes and data nodes affects the performance. Both Azure and Amazon offer the possibility to select the data centers in which data will be placed and this allows users to place the computation in the same center. However, the overall performance can be increased by moving data closer, in the same racks or perhaps on the same node.

The idea that we propose is to exploit the data locality principle by having the data and the computation in the same nodes of a public cloud. This approach trades data-availability with efficiency, a good tradeoff for scientific applications that are executed in the cloud, since the data, most often private, is not accessed by the general public. To our knowledge there are no DFSs that can be deployed

in the computation nodes of public clouds like Azure. Our contribution fills this gap, offering an efficient solution for data-intensive applications.

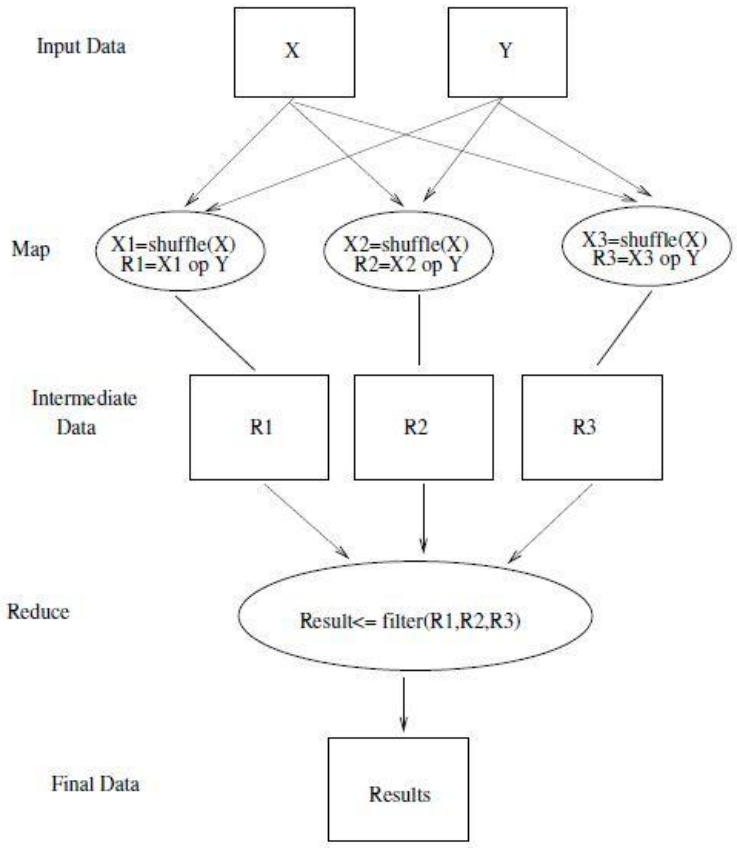


Figure 7: MapReduce Example of the neuroimaging and genetic application

Each virtual machine (node), in both Azure and Amazon clouds, has a local storage space varying from 160GB up to 2TB. This local storage can be used with no additional cost, but the major drawbacks are the fact that it is not persistent (in case of failures all data on it is lost) and is not accessible to the other nodes. Our solution overcomes these limitations and exploits these resources efficiently. It consists of a distributed file system that is deployed on all computational nodes and offers to the application running in each node the same view of a unique, shared file system. The DFS, called A-BlobSeer, uses the local space on each node to store the data and offers the possibility to all nodes to access data from any other nodes. By placing the data on the local storage of the nodes, the data is as close as possible to the computation, thus applying the principle of data locality. Fault tolerance is addressed through replication: having several replicas ensure that even in case a node fails, the remaining nodes can still retrieve the remaining copies of the data. In addition to efficiency of data manipulation, another advantage relates to the cost. By using a DFS instead of the storage offered by cloud providers, the cost for storage and for the bandwidth traffic are eliminated.

A-BlobSeer exploits the principle of data locality, by providing a distributed file system that is deployed on the computation nodes of a public cloud. By moving the data closer to the computation the write performance increased up to 5 times and the read performances increased up to 2 times over the remote storage offered by the cloud. Encouraged by these results we have constructed a MapReduce platform, A-BMR, optimized for data-intensive applications, which uses the A-BlobSeer as storage backend. As a third

contribution, the A-BMR platform was tuned for a particular type of data-intensive applications that are based on univariate analysis. These MapReduce platforms provide fault tolerance both for data and for the jobs, they have runtime scalability and exhibit autonomic properties. Security and privacy are also provided together with the other guarantees offered by a public cloud. These platforms were evaluated with the neuroimaging and genetic application and the computation time was reduced up to 4 times over the case when a general MapReduce platform is used. The results obtained validate our approaches and allow the continuation of the work for studying the variability between individuals using the case study application, and as well, executing new data-intensive applications in public clouds.

EVOLVING INVERSION METHODS IN GEOPHYSICS WITH CLOUD COMPUTING

J Craig Mudge, Pinaki Chandrasekhar
Collaborative Cloud Computing Lab School of Computer Science
Graham S. Heinson, Stephan Thiel
School of Earth and Environmental Sciences
University of Adelaide

Magnetotellurics is a geophysics technique for characterization of geothermal reservoirs, mineral exploration, and other geoscience endeavors that need to sound deeply into the earth – many kilometers or tens of kilometers. Central to its data processing is an inversion problem which currently takes several weeks on a desktop machine. In our new eScience lab, enabled by cloud computing, we parallelized an existing FORTAN program and embedded the parallel version in a cloud-based web application to improve its usability. A factor-of-five speedup has taken the time for some inversions from weeks down to days and is in use in a pre-fracturing and post-fracturing study of a new geothermal site in South Australia, an area with a high occurrence of hot dry rocks. We report on our experience with Amazon Web Services cloud services and our migration to Microsoft Azure, the collaboration between computer scientists and geophysicists, and the foundation it has laid for future work exploiting cloud data-parallel programming models.

Computing resources needed for geophysical data processing and inversion have grown at a rate faster than can be handled by conventional desktop computers. The issues are twofold: first, memory is not sufficient to store information pertinent to the phenomena under investigation. Because the physical memory limits of 32-bit machines (and lower priced 64-bit computers) do not allow a full analysis, the problems are typically broken down into more manageable fractions. The second issue is one of time. Modeling and inversion of electromagnetic data is highly non-linear in its parameters, and thus most computations require significantly sized and repeated matrix operations. When implemented in a sequential program, these are increasingly taking many weeks or even months to perform on a desktop computer. In common with many scientists, we find the parallel execution available on High Performance Computing (HPC) infrastructure available to us to be too hard to use without specialist training.

III. ELECTROMAGNETIC GEOPHYSICS

A. Electrical conductivity

The determination of subsurface electrical conductivity σ (or its inverse resistivity ρ) plays an important role in a variety of applications from tectonic evolution, mineral and geothermal exploration and groundwater studies. In contrary to other geophysical parameters, i.e. seismic wave velocities and density, Fig.1 shows that the electrical conductivity varies over more than seven orders of magnitude. The large spread in conductivity ensures sufficient contrast to image geological structures of elevated conductivities in an otherwise electrically resistive host rock (igneous and metamorphic host rocks in Fig. 1).

In general, earth's lithosphere is quite resistive (>1000 Ωm), but frequently zones of higher conductivity are observed. The causes of enhanced conductivities are diverse, but can be attributed in

most cases with additional geological and other geophysical constraints [3]. Among the common agents for enhanced conductivities are minor conducting phases like graphite and metallic sulfides formed through metamorphic processes frequently involving shear [4]. Aqueous fluids distribute between rock grains of porous rocks and can significantly enhance conductivities if they are well connected and increase with temperature and salinity of the fluid. Fluids play a major role in the characterization of geothermal areas and also in the understanding of ore genesis.

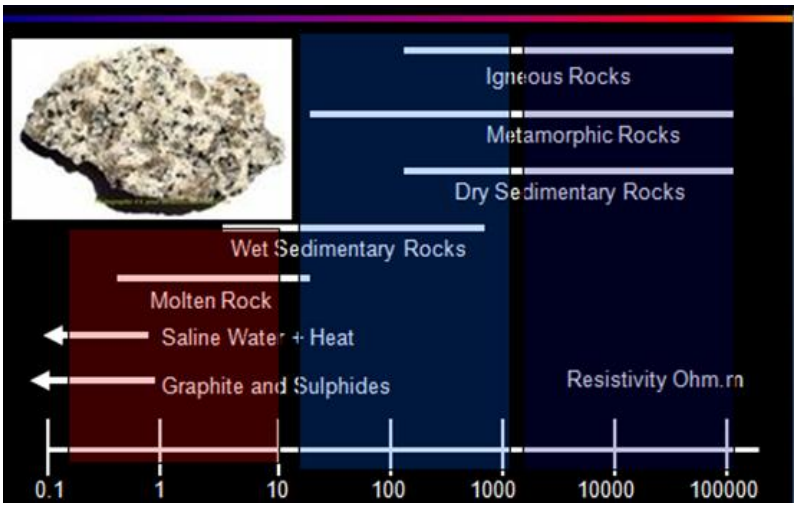


Figure 1. Electrical resistivity as a function of lithology. Dry igneous and metamorphic rocks show several orders of magnitude higher electrical resistivity than rocks with minor conducting agents, such as fluids, graphite and sulfides. In general resistivity also decreases with temperature.

Monitoring fluid injection and reservoir development of Enhanced Geothermal Systems (EGS) is one of the crucial steps in maximizing energy extraction in the renewable geothermal energy sector. The MT project at Paralana is aimed at using the MT method to better understand the processes involved during the stimulation of an EGS fluid-system. The initiation and propagation of the underground fluid-reservoir is dependent on several parameters, including the stress regime, geometry of pre-existing faults, and the reservoir lithology.

The processing is in two stages. The first takes the MT time series data which has been logged in multiple stations in the field. After a cursory inspection by hand, the data are run through the BIRRP step [10]. BIRRP is a robust statistical method that removes outliers along with converting the data, via a Fast Fourier Transform, into sets of impedance tensors. Each set of impedance tensors correspond to a particular desired evaluation frequency. The impedance tensors in turn become the input to the second stage, the MT inversion step as shown in Fig. 3b.

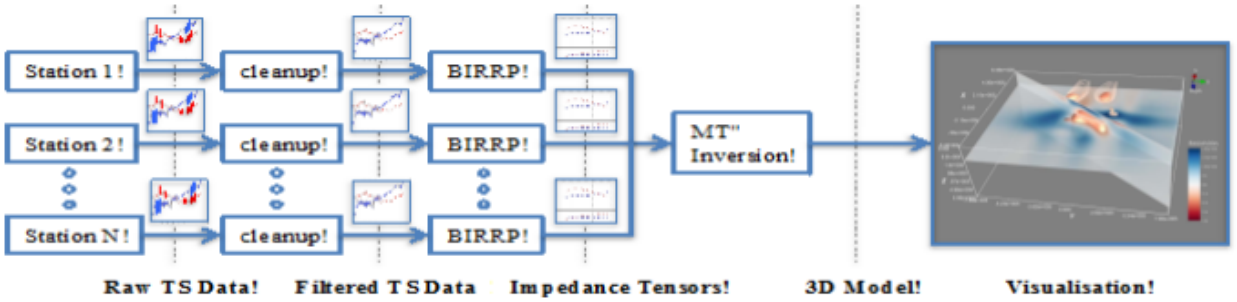


Figure 3b Overview of the MT processing steps

There are two sections of the code: the parallelized FORTRAN code and a web application. The FORTRAN program, 22,000-lines in length, is widely used in geophysics. The parallelized code abstracts the logic which computes the sensitivity and forward modeling calculations per evaluation frequency. The computations for each frequency are run in parallel. To improve the extent of its adoption, and to improve the user experience, we embedded the resulting program in a web application. The web application is implemented using Django, a popular web application framework, written in Python, which follows the model- view-controller architectural pattern.

The remote tasks are executed in remote EC2 instances standing by for this particular inversion. 2 A complete EC2 instance is dedicated for any remote task. Also, there is no shared memory between processes with the necessary state variables required for the separate tasks persisted on file and transferred between the instances.

The entire distributed process is implemented on the Amazon EC2 cloud using standard available EC2 instances. The whole Inversion process is further packaged as a MT 3D inversion software product which is accessible anywhere through a secure login via a common-or-garden browser. Amazon EC2 machines are automatically acquired on demand only for the duration of the inversion and released soon after. Relevant results and configuration data are stored in Amazon S3 storage.

Our lab is exploring different types of cloud services in order to understand costs and benefits of using cloud resources for large scale eScience applications. Microsoft Azure is a Platform as a Service (PaaS) model whereas Amazon Web Services is an Infrastructure as a Service (IaaS) model. We will also use community cloud services, such as those utilities provided by our federal government for university research. Our migration to Azure is complete and has the architecture shown in Figure 6. Note the symmetry of the Amazon and Azure architectures.

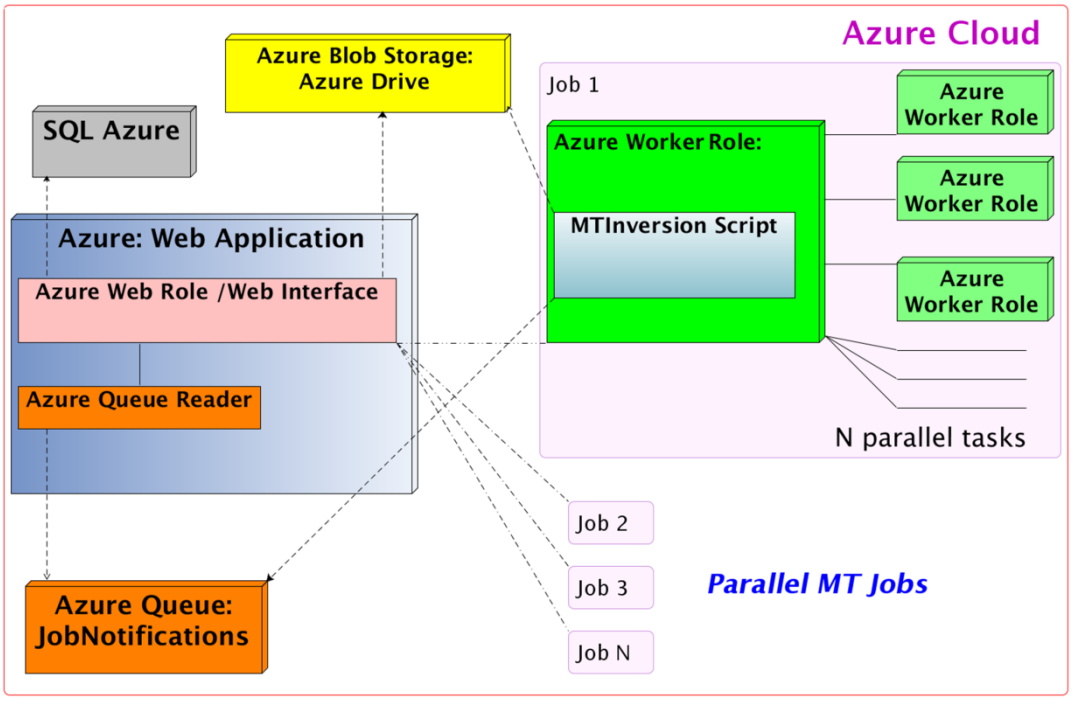


Figure 6. Architecture overview of MT running on Windows Azure cloud.

SPEEDUP FROM PARALLELISATION

The computation time versus a range of frequencies used in the inversion are shown in Fig 7.On average, with our cloud-based implementation, the modeling time was reduced by a factor of four reducing the computation time from weeks to days.

A straightforward approach to converting legacy code to a parallelized distributive process is to transfer all control logic and optimization functions of the 3D Inversion to the control script/master process. The idea is to reduce the FORTRAN code to computational functions and stripped of as much decision logic as possible [11]. In order to do this we segregate optimization, global and local variables in the FORTRAN code to begin with. There are certain optimization logic decisions which if embedded deep inside the FORTRAN makes it very hard to be stripped out. These variables are not removed but are made sure to be consistent and up to date in both the FORTRAN code and the master control script. The idea is to not rework any logic or algorithmic technique in the FORTRAN code which would require it to then go through a rigorous testing cycle.

The control script/master process is implemented using Python. It controls the forward and inverse optimization loops, remote task delegation of parallel & independent tasks, aggregation of remote task outputs, File I/O operations and upload of results, errors & log files onto Cloud storage.

There were 18 FORTRAN routines totaling 22497 lines of code. Four of the 18 routines are unchanged. The Python script has 600 lines. Overall the project took six months. To help those undertaking similar efforts in the future, the time taken (by the one person working on the programming) is given.

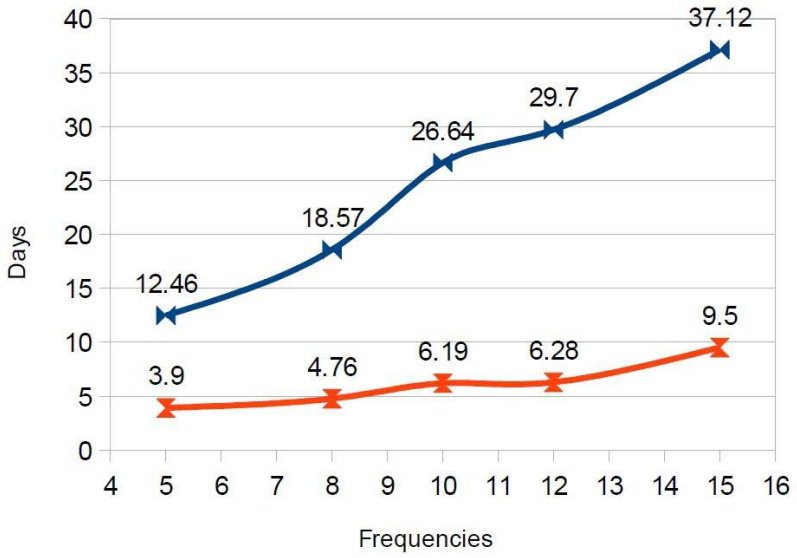
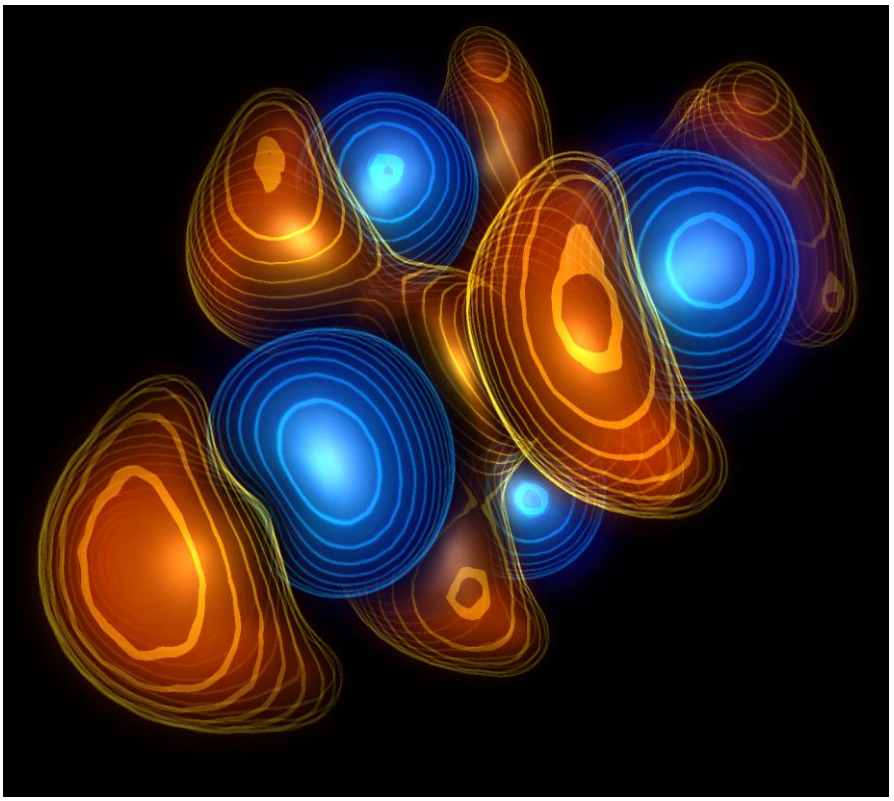


Figure 7. The upper curve shows the execution times for the serial program in days, while the lower curve shows the faster parallel times. The speedup increases with the number of frequencies.

USING WINDOWS AZURE AT THE NATIONAL COMPUTATIONAL INFRASTRUCTURE

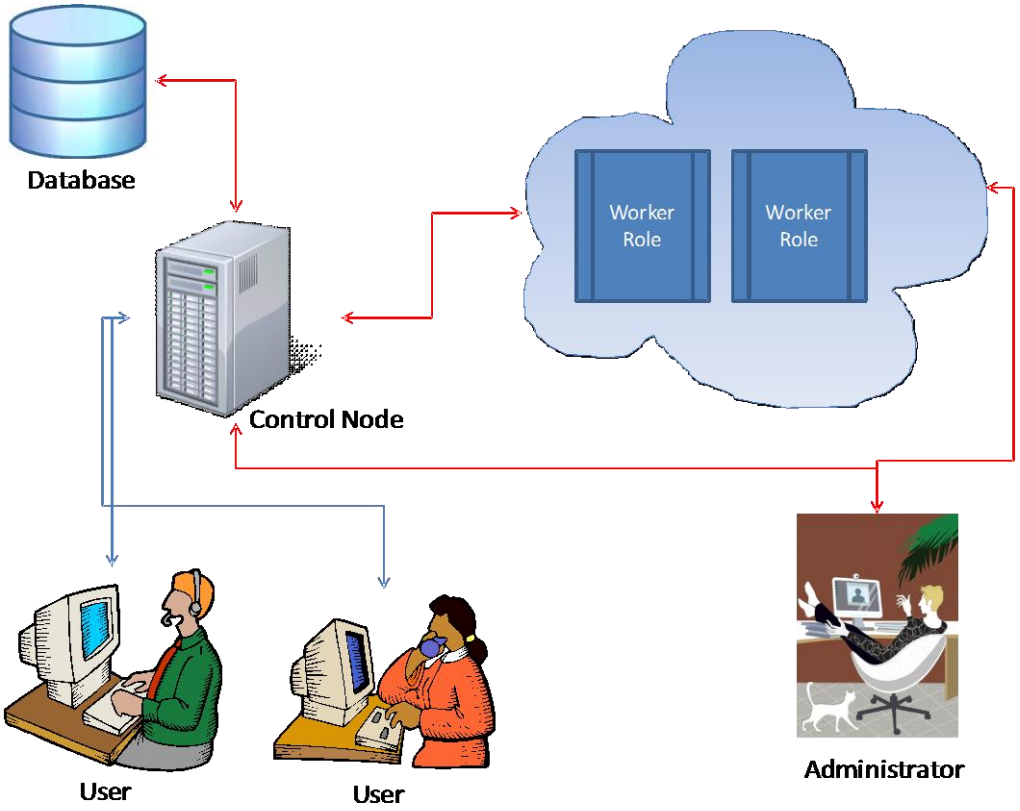
Vladislav Vassiliev and Benjamin Evans, NCI, ANU, Canberra, Australia

At the NCI National Facility, we explore the execution of a selection of packages on the Windows Azure cloud fabric. We are developing a system which supports a typical HPC job submission and management system, and ability to monitor job status and resources monitoring. A system does not target a particular package and is suitable for the generic submission scenario. Part of our goal is make it is easy for users and support staff to access the cloud service in a way familiar to their existing access to HPC systems, including command-line and browser interfaces. The Azure Cloud is also being assessed as a place where common small to medium-scale HPC packages could be executed. The initial candidate packages being tested include several popular Computational Chemistry programs including Games and Gaussian.



THE IMPLEMENTATION

The system we have developed consists of several components: a Control Node, database , and the Azure Cloud fabric including hosted services and persistent storage. In a typical example, a user interacts via a command-line interface to the Cloud through the Control Node (Linux OS). The Control Node provides a set of commands similar to those in typical batch oriented HPC. Doing so simplifies the access for this traditional method of usage, and therefore the Cloud resources are more readily accessible.



The Control Node components are written in Java to ensure their cross-platform capabilities. The job submission consists of copying the input files to the Azure persistent storage and sending the Job Description file to the Azure queue to be processed by the next available Worker Role on the Cloud. The Job Description file contains necessary information to execute the job: the list of required executables and input files, environment variables to be setup, job run pattern, etc. The Worker Role runs the job according to the scenario described in the Job Description file and informs about the job progress execution through the Azure queue. The Control Node components perform administrative and user functions: job submission, extract messages from the Azure queue and update the database on the current job status, download output files for the completed jobs, etc. More details about the implementation are on our web site, http://sf.anu.edu.au/~vvv900/azure/

Would we spend $25,000 for the Cloud?

Current answer is definitely "No"! The Cloud is too slow for the HPC applications and requires many man hours to set it up for the HPC purposes.

Our benchmark calculations showed that for the packages we had used 8 Azure nodes are roughly equivalent to 3.5 2.93GHz Intel Nehalem CPUs of our Oracle/Sun Constellation Cluster. Taking into account that some applications, for example, in Computational Chemistry do not have good parallel efficiency above 8-16 CPUs increasing number of (slow) Azure nodes won't lead to the increased Azure computational performance. Thus, such kind of applications won't benefit much from the Azure.
Probably the Azure Cloud is better suited for so-called "embarrassingly parallel" applications which subtasks rarely or never have to communicate. In this case the slowness of the Azure nodes can be compensated by their large number.

In its current state the Azure Cloud infrastructure is very poorly suited for the HPC applications. The weak points are

1. Prior to starting to use the Azure Cloud one needs to invest a lot of time into learning the Azure Cloud infrastructure and then into the Azure programming to achieve even the very basic things.
2. The Azure Cloud has very poor debugging facilities so even a simple problem with the Azure Cloud application can take hours or even days to be solved.
3. The Azure nodes are slow for the HPC applications.
4. Memory limits are low for some applications.
5. Since practically all development of the HPC applications goes under the Linux/Unix it is a real problem to find the parallel Windows binaries for the Cloud for many applications
6. Building the Windows binaries from the sources developed under the Linux/Unix requires a lot of time.
7. The Fortran compilers for Windows are expensive.
8. One needs to rewrite the Unix scripts either into the DOS batch files or to use Cygwin (which is not installed on the Cloud by default)

LOCATION IMPROVEMENT IN VANETS USING WINDOWS AZURE

Farhan Ahammed, Javid Taheri and Albert Y. Zomaya

Center for Distributed and High Performance Computing, School of Information Technologies, University of Sydney, Australia

GOALS OF THE PROJECT

A Vehicular Ad Hoc Network (VANET) is a mobile wireless network where the nodes of the network are vehicles and/or devices fixed inside or beside the roads. The vehicles can use GPS devices to receive an estimate of their current location. Measuring distance to other vehicles is also possible using communication or other devices. However, a level of inaccuracy can be associated with GPS devices and the accuracy of distance measurement techniques reduces as the distance to be measured increases. One way to improve accuracy of in-vehicle localisation techniques is to share information among adjacent vehicles on a road so that they can all estimate their true location more accurate.

Certain algorithms have been designed to address this issue –improving the accuracy of coordinates through sharing information– such as one of our algorithms named VLOCI2. This project is concerned with finding lower bounds on the amount of improvement possible on GPS-provided coordinates, by simulating a VANET with vehicles travelling along a road, while communicating with each other to improve their location coordinates.

TECHNICAL APPROACH

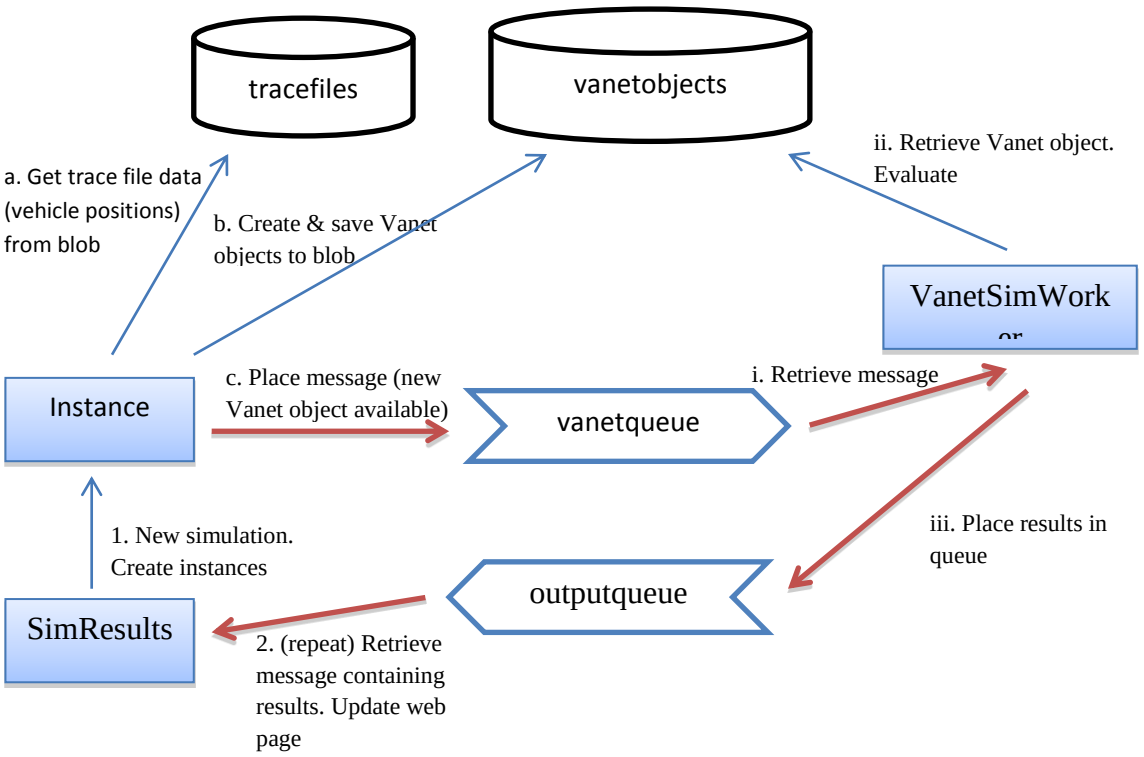
The designed program simulated vehicles within a VANET travelling along a road and communicating with each other. The simulator computes the *location error* (measure of the average inaccuracy of location coordinates) of a VANET before and after applying a location improvement algorithm (e.g. VLOCI2).

The program creates multiple *Instance* objects with the aim of finding a particular Instance object that can produce consistent favourable results. To compute how well an Instance object fits this criterion, multiple *Vanet* objects are created, representing the different states of the vehicles within the vehicular network. Every Vanet object is *evaluated* with the results of this evaluation combined to provide a quantitative measure for Instance objects. This process is very time consuming and may take up to several weeks for a decent size VANET with 1000 nodes/vehicles moving almost for an hour.

Using Win-Azure was one of the best ways to expedite such process. Various trace files, which contain information regarding the position of every vehicle at various times throughout the simulation, are first uploaded to the server. Then, using such data, Vanet objects are created to model this information. These Vanet objects perform the bulk of the calculations required to complete the simulation. For this reason, worker roles are designed to use the Vanet objects.

When an Instance object creates (multiple) Vanet objects, each Vanet object is placed onto the Blob. This is to allow the workers to have access to the Vanet objects. A message is then passed through a queue indicating that a new Vanet object was created.

When a worker receives a message from the queue, it retrieves the corresponding Vanet object from the blob and evaluates it. The result of the evaluation (represented as an array of doubles) is place onto another queue. The program (with a web interface) receives this message and updates the result on the web page. The diagram below describes this process.



ADVANTAGE OF USING THE CLOUD

Since every Vanet object is independent of one another, evaluating Vanet objects can be performed in parallel and in no particular order. Aggregating the results then requires simple arithmetic. The design of multiple workers running in parallel is well suited for this approach, since each worker can evaluate its own Vanet, publishes the result, and then retrieves another Vanet object, regardless of what the other worker objects are doing. The time required to evaluate a Vanet object (and hence the time spent by a worker) varies between a few minutes to an hour. This requirement is well suited to cloud-based applications.

Amount of Cloud Resources Required

Due to the nature of our project, we could literally use any number of cores during our simulation. We however started with one Web-role and two worker-roles and then gradually increase the number of workers. We realized that after a certain number of workers (greatly depends on the size of the uploaded trace file), speeding-up effect of every extra worker becomes almost negligible.

CHALLENGES OF WORKING WITH AZURE

Queues have size limits; and therefore, we had use sluggish blobs to pass on Vanet information and storage data. Also, the workers are stateless and do not know whether a new simulation was started or not. Simulation parameters (common to all Vanet objects) were thus designed to be included in all

Vanet objects. This way, the workers do not wait/check for a new set of simulation parameters; they only have to wait for the next Vanet object to become available.

SECURE HIGH PERFORMANCE COMPUTING ON WINDOWS AZURE

Taheri and Albert Y. Zomaya

Center for Distributed and High Performance Computing, School of Information Technologies, University of Sydney, Australia

Clouds, by definition, provide an infinite computing capacity and storage to hide tedious details of the underlying software and hardware from programmers; such a unique feature makes the cloud paradigm very attractive for many application domains.

This project proposes a High Performance Computing platform with Security guarantees. An example to demonstrate the goal of this project is to try to analyze millions of bank transactions to detect frauds and/or illegal activities. Such computations are currently performed on local clusters for such organization. Lack of portability and extension are common concerns for current solutions.

HPC solutions–either secure or non-secure–are left almost uncovered in all current clouds; to the best of our knowledge, Azure HPC released in Nov-2011, is probably the latest product to address the non-secure version of HPC applications in the cloud. The different nature of applications in HPC and those formerly targeted in clouds as well as many other technical and performance challenges are among the main reasons for such shortcomings. In fact, clouds are majorly designed to provide efficient platforms for handling highly data dependent applications with simple algorithms (e.g., sorting client log-files) rather than complex time-consuming algorithms with lesser amount of data requirements (e.g., aligning bank transactions to detect frauds) usually demanded in HPC applications. As a result, allocating cloud resources for HPC applications is probably much more complicated than traditional data oriented cloud applications. Security is another reason many organizations hesitate to send their confidential computation to the cloud. Regardless of the application design, the fact that all information needs to be sent to the cloud is almost inevitable. A naïve approach to tackle this issue could be the use of sophisticated coding techniques to obscure critical information; such techniques are however completely impractical as the decoding algorithms are also sent to the cloud! Therefore, security is always compromised.

This project is an attempt to separate information from computation for highly secure HPC applications.

Scientific Challenges

Azure schedulers control/allocate requested Virtual Machines (VMs) to their underplaying Physical Machines (PMs). For many data-intensive applications, which are embarrassingly parallel in nature and do not need massive inter-communicating messages among VMs, Azure schedulers are quite enough and most probably well balance PMs' loads. As a by-product of such load balancing, users' applications are also efficiently executed in the cloud. Secure HPC applications are however usually CPU intensive where a collection of parallel modules needs to communicate with each other–through exchanging messages. Because cloud schedulers cannot know the capacity of such messages before scheduling their executing VMs, they may simply allocate them to distant PMs–to balance their executions–with slow commutation links. As a result of such networking overhead, the whole HPC process can be significantly slowed down. Furthermore, security, performance, and speedup are the main concerns in Secure-HPCs; whereas, load and energy are for the current clouds applications. All these differences bring extra complexity to the design of external schedulers that are required to not only schedule their processes to cloud VMs, but also secure the underlying information. The following list briefly explains the most important secure-HPC specific challenges for the cloud that cannot be addressed through current cloud schedulers.

Performance: HPC applications must run faster (or at least fairly the same) in clouds than in current physical clusters. To the best of our knowledge, no mechanism exists to date to measure such speed and/or even guarantees/predicts the future execution time of applications in clouds.

Resource Utilisation: HPC applications usually consist of several tightly interconnected parallel modules. Thus, considering I/O communications between them is at least as important as balancing their CPU utilisation in many cases.

SLA: Cloud providers must satisfy SLAs for HPC applications like current applications. Definition of such SLAs is however inherently different from the current ones as their focus is vastly different.

Workflows: HPC applications usually follow a workflow in which each step is a parallel process. Therefore, cloud providers not only need to optimise resource utilisation for each step, but also between steps where a large amount of data is usually transferred.

Security: Local clusters/clouds are the only current solutions where security of information is more important that the execution time of any algorithm. Clouds rarely focus on such issues.

Our Design and Implementation

This project aims to address all of the above issues and provides a complete solution. The solution is based on the real fact that exclusive privilege to real schedulers in the cloud (Win-Azure in this case) does not exists; and thus, we need to provide our scheduling solutions through connecting an external scheduler to several VMs in the cloud. Through AppFabric, Win-Azure has already provided a platform to develop such hybrid solutions; its optimal deployment however is still open. Figure 1 shows the overall picture of our solution and explains how such hybrid solution can be formed. Here, users/customers submit their jobs to the local server, acting as the main scheduler. Upon receiving any job, the active scheduler pulls information regarding the status of its roles in the cloud and finds the best VM to run the submitted job. Other provisioning mechanisms additional to the main scheduling process to evaluate the current load as well as to predict future loads is to (1) allocate/delete VMs in the cloud and/or (2) change the size of the current ones. This unit is mainly responsible to reduce the total cost of using Win-Azure while satisfying all SLAs of the submitted jobs.

It is also worth noting that, because the whole scheduling process is performed in the local scheduler (external to the cloud), our scheduler is also able to consider the security requirement of each submitted jobs and sends it to either its local underlying cluster or to the public cloud while satisfying its security requirements. Here in particular, because our scheduler only sends part of the data to the cloud, the integrity of information is never compromised.

Having stated the overall framework of our design, the most important scientific challenge of this solution is the design of a collection of smart/intelligent multi-objective scheduling algorithms to concurrently (1) satisfy SLA of all jobs, (2) minimize execution of all jobs, (3) maximize resource utilization of its local cluster, and (4) minimize the overall cost of sending application to the cloud. It is worth nothing that the third objective itself is a proven NP-complete problem; and therefore, the

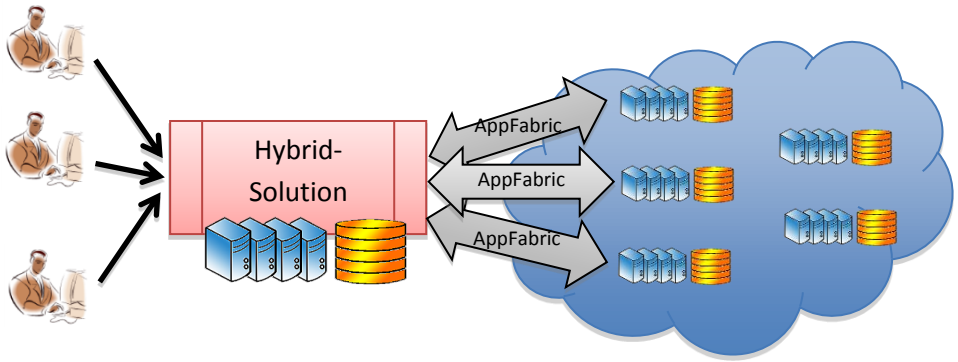


FIGURE 1: SECURE-HPC IN WIN-AZURE

combination of these objectives is strictly NP-complete in nature.

AN ACCOUNTABLE COORDINATION SYSTEM FOR COAL SUPPLY CHAINS IN AUSTRALIA

Shiping Cheng, CSIRO Australia

Project Background and Goals

A typical transport supply chain (called coal chain in the following) in Australian coal industry involves multiple independent business entities that own different resources, such as coal mines, mine loading points, rail connections, coal loading terminals and vessels. For example, the Hunter Valley Coal Chain has 35 coal mines owned by 14 producers, 24 load points, 28 trains with 15,000 trips per years run by 2 rail operators, tracks own by 2 operators, 2 coal loading terminals with 5 dump stations, 1.5Mt of working stockyard and 5 shipping berths, and approximate 1,000 vessels per year.

The producers and operators are independent. They contract with each other to ensure the resources for shipping the coal. However, the individually negotiated contracts may not lead to an optimal (or sometimes even feasible) resource usage in the whole coal chain. The coal chain can run more efficiently with a coordinator. However, the obstacle of setting up a coordinator is not trivial.

Firstly, participants need to make some sensitive information available to the coordinator, e.g., if Miner A discloses that it needs to transport *100* tons coal to stockyard 1 for loading to vessel *v1* that arrives at 9:30AM, its competitors may figure out its business partner and contract size etc. Secondly, the current practices do not guarantee that the information won't flow to the competitors. Thirdly, convincing participants to share information is difficult.

To address this problem, there needs a system that can ensure the shared information is only used for the purpose specified by the information provider, which includes following properties:

1. The system should allow the party that supplies the data to specify who can use the data and how the data should be used.
2. The owner specified policies should be enforced when data flow across administrative boundaries.
3. A user-supplied program that accesses a set of data in the system should label information flow inside the program so that a party can check the following: 1) if the program satisfies the access control and information flow policies of its data owner; 2) if the access to the output produced by an application should also satisfy certain policies specified by the owners of input data.

In this project, CSIRO Researchers exploit the use of cloud computing platform to achieve above goals.

On-Going R&D Activities

Cloud computing technologies provide effective isolation for data and compute of collaborating parties. An independent cloud infrastructure provider can serve as a middleman for monitoring the data exchange among various parties. It is therefore capable of collecting evidences for detecting the violation of information flow policies.

The system architecture is shown in Fig. 1. It consists of the following components:

- Data store: a data store contains information local to each participant. A data store is maintained by its owner and exposes a Web services interface for external parties to access the data.
- Coordinator: a coordinator aggregates data from related data stores for computation. The computing results can be pushed to data stores according to give information flow control policies.
- Monitor: a monitor captures information flows when they cross administrative boundaries.
    - A monitor also generates events for compliance check against information flow policies.
- An event log aggregator and a log analytic engine, which process data from monitors for problem detection.

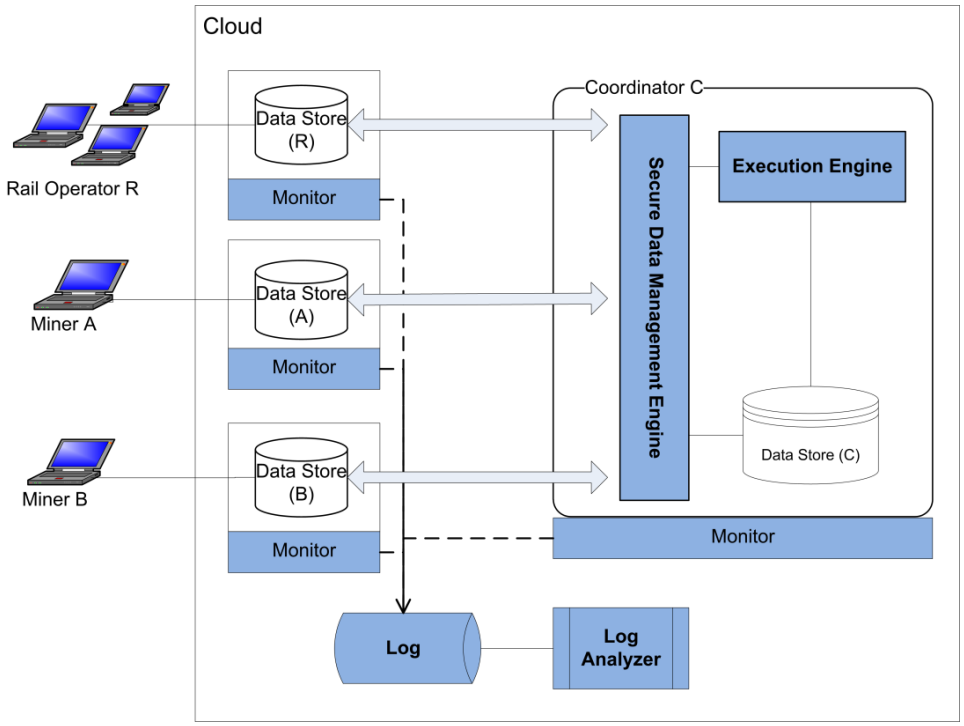


Fig. 1. System architecture.

The processing of log data is a computing and data intensive task. We are currently implementing the log analysis algorithms to enable them to run on Azure.

Throughout the project, we realize that treating "cloud" as a middleman has great potential to enhance the trust among collaborating parties from different administrative domains. More often than not, a well known cloud infrastructure provider is more trustworthy for a party than its collaborative parties that have conflicting interests.

In addition to the above R&D, CSIRO researchers are also conducting a study of performance and scalability of cloud using Microsoft Azure.

ARGUMENT STRUCTURE ANALYSIS OF HUGE WEB CORPORA FOR IMPROVING A SEARCH ENGINE INFRASTRUCTURE

Daisuke Kawahara and Sadao Kurohashi (Kyoto University)

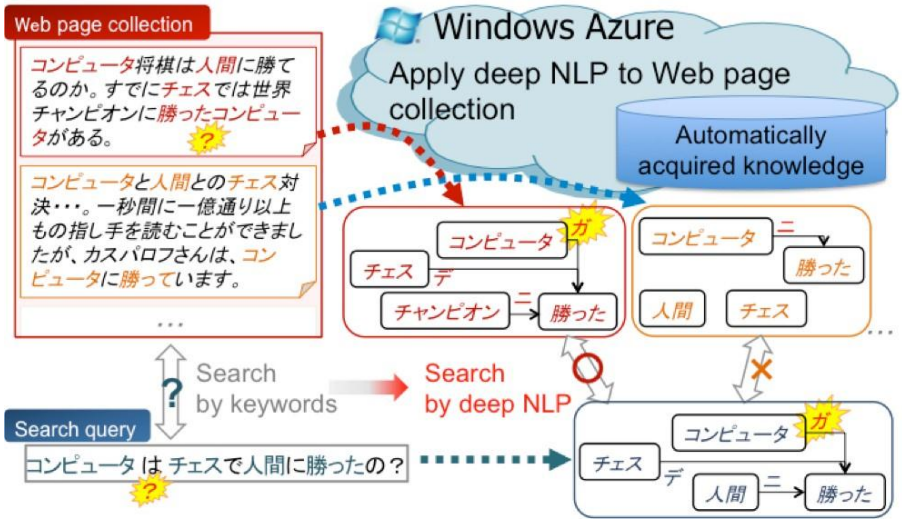


Figure 1 Search by deep NLP enabled by Azure

Web documents have become the far most important information recently. Considering their explosive growth and diversity, conventional ranking-based information retrieval (IR) is obviously insufficient. It is necessary to provide a bird's-eye view towards a given topic by organizing and relating information. We strongly believe that text understanding by machine, or natural language processing (NLP) is the most important for explosive-era IR.

We have been developing a search engine infrastructure, TSUBAKI, which is based on deep NLP. While most conventional search engines register only words to their indices, TSUBAKI provides a framework that indexes synonym relations, hypernym-hyponym relations, dependency/case/ellipsis relations and so forth. These indices enable TSUBAKI to capture the semantic matching between a given query and documents more precisely and flexibly (Figure 1). TSUBAKI also supports natural language search, which directly represents users' information need.

Among these various relations, case/ellipsis relations have not been indexed in a large scale. This is mainly because the speed of these analyses is not fast enough due to the necessity of referring to a large database of predicate-argument patterns (case frames), which is automatically acquired from a large corpus. In this project, we apply case/ellipsis analysis to a huge Web corpus by using the Windows Azure cloud computing environment. To apply case/ellipsis analysis to millions of Web pages of TSUBAKI in a practical time, we needed about 10,000 CPU cores. According to the XCG engagement team in Microsoft, obtaining a single service having 10,000 CPU cores in a hosted service of Azure was a major challenge, therefore we created instead 29 hosted services, each having 350 CPU cores, as the XCG team suggested. Each service processes web pages using a master/worker model. In this model, a master manages workers, which concurrently apply our analysis to an input file. This concurrent execution does not need reciprocal communication and can be performed independently.

In addition, since our case/ellipsis analysis system has been developed using the C language on Linux, it was necessary to port our system to Windows in order to execute on Azure. To do this, we employed a

Unix-like environment, Cygwin. We implemented the above framework and tested our analysis on 1x350, 2x350 and 8x350 step by step. Once we confirmed that we could obtain 29x350 CPU cores, we executed our analysis on these CPU cores. A remaining problem at this moment was the high cost of manually managing 29 hosted services. We then kept developing a manager of 29 hosted services based on the Windows Azure Service Management API. Recently, we completed this development and will run it using 10,000 CPU cores.