Mat-2.4177
Seminar on case studies in operations research

Simulating Insurance Portfolios using Cloud Computing

Final report
May 18, 2011

Client: Model IT

Jimmy Forsman (Project Manager)
Tuomo Paavilainen
Topi Sikanen
Taneli Silvonen

Contents

Introduction

In this report, cloud computing refers to all services in which the capacity is fully scalable, does not require up-front commitments and the users pay by the hour. The benefits of cloud computing are closely related to these features. It allows users, who do not own the necessary resources, to operate their services without heavy infrastructure investments. It can lower the risk entrepreneurs take when starting their business.

The scalability of cloud computing is attractive for solutions which require a lot of computing power momentarily. This includes the mSII product of this project and scientific computing applications. The question is, whether other properties of the cloud, such as data transfer and input/output performance, are suitable for high performance computing. The user does not always know what they are getting. The precise specifications of the hardware paid for are not usually known. Performance might depend on the usage rate of the particular instance the user gets.

Although services, which are now referred to as cloud computing, have existed for a while, the concept of selling hardware capacity via Internet is fairly recent. It started with selling un-utilized capacity (for example Amazon) and is now an industrialized business with huge data centers (for example Microsoft). For this reason, it is constantly changing and developing. Service providers, enterprises and research groups are thinking of new ways to utilize the properties of the 'cloud infrastructure'. Some efforts have already been made and these will be discussed later in the report.

This report starts with a background section in which financial modeling, insurance regulations, cloud computing and mSII are discussed. Then we present results from previous work, when the high performance computing abilities of the cloud have been examined. That is followed a short review of cloud computing economics. The 'Service providers' chapter consists mostly of discussing the properties of Windows's and Amazon's cloud services. After that, we present our approach to and findings of the technical side of this project.

Background and motivation

Modeling

The Monte Carlo method can be used to calculate various financial parameters, such as prices and risks. Monte Carlo simulations in security pricing consist of three parts: (i) simulate sample paths of the underlying variable (ii) evaluate the discounted cash flows on each sample path (iii) average the discounted cash flows over sample paths. The standard error of this basic Monte Carlo method depends on the amount of simulations, $\sigma \sim 1/\sqrt{n}$. (1) This is roughly the algorithm our client Model IT uses in its msII product to simulate insurance portfolio cash flows.

There are also other ways for modeling cash flows of insurance companies. For example, Koivu et al. (2005) introduce a vector equilibrium correction (VEqC) model which estimates the parameters of the model from time series data, or they can be user-specified. The VEqC model is a generalization of the vector autoregressive (VAR) model which can be also used for this purpose. (2) The above-mentioned model is utilized when Hilli et al. (2007) suggested an optimization method for the asset liability management of the insurance company with stochastic programming (3).

Solvency II

Solvency II is a set of regulatory requirements set by the European Union for insurance companies. It consists of three pillars. The first pillar contains the quantitative requirements which are two capital requirements. These will be introduced more thoroughly later since they are the main motivation for the msII product. The other two pillars cover qualitative requirements, such as risk management, and transparency, such as public reporting and disclosure. (4)

The two capital requirements of Solvency II are Available Capital and Solvency Capital. In the case of life insurance companies, neither of these is straightforward to calculate because of the complex financial structure of the life insurance contracts. Usually, life insurance companies rely on simulations to estimate the capitals. The simulations for the market value of liabilities are conducted according to an internal model or a standard external model. The implementation of these models has been difficult for some companies because the methods of the internal model are inefficient and the standard model does not reflect accurately the company's risk situation. (5)

There is a need for more accurate models, which also reflect the individual nature of the insurer, under Solvency II. Since closed-form analytical solutions are not possible, the Monte Carlo method offers a way to estimate the cash flows of each contract, and thus define the Available and Solvency Capitals. The Monte Carlo method is fairly easy to implement, but requires a lot of computing power. Even a small Finnish insurance company has tens of thousands clients

which can have multiple contracts. The cash flows of these contracts are simulated, say, a thousand times for a time period of 60 years.

Cloud computing

The term cloud computing means various different things depending on who you ask. The National Institute of Standards and Technology (NIST) defines it as “a model for enabling convenient, on-demand network access to a shared pool of computing resources” which has such characteristics as broad network access and rapid elasticity. The NIST definition also mentions three different service models: Software, Platform and Infrastructure as a Service. (6) Armbrust et al. (2009) from Berkeley University eschew using the different service models because the borders between them are so vague. They use simply the term Utility Computing when talking about cloud computing. (7)

The biggest providers of cloud services are Amazon Web Services, Microsoft Azure and Google AppEngine. These and other cloud-related services and providers are depicted in Figure 1 using the service division mentioned earlier. As we shall see, Amazon offers infrastructure and lets the user have control over operating systems, storage and deployed applications. Microsoft Azure will be the service provider used in this project, and it is introduced in more detail later in this report.

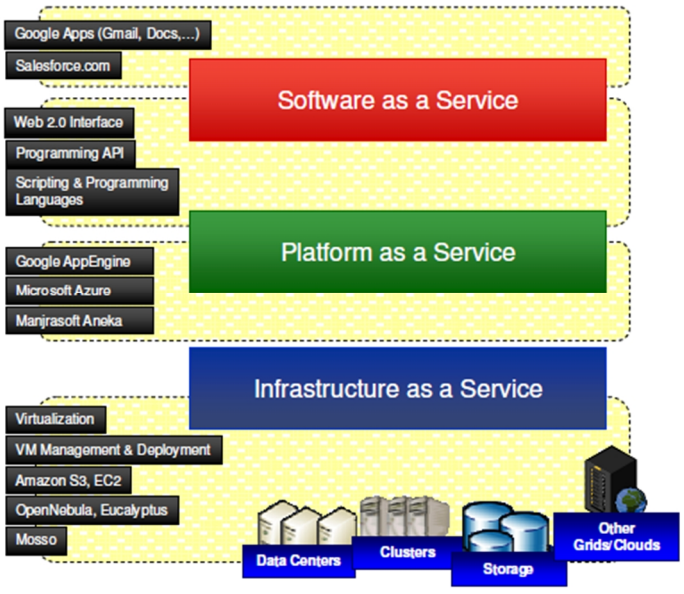

Figure 1. Different cloud services and service providers. (8)

mSII

Our client, Model IT, has developed a product called mSII to evaluate the cash flows of insurance portfolios. An insurance contract is modeled as a cash-flow exchange contract between the company and the customer. Cash flows can be repetitive, single or recurrent. The numerical contract-level Monte Carlo simulations produce a full distribution of the cash flows, which can be extracted

and reviewed more closely. There is no need for the complex option theory. Instead, options are priced with Monte Carlo as a by-product of the simulations. (9)

mSII allows importing the company's decision-making rules and external economic scenarios. Implementation of mSII includes defining the product cash flows, reading in the existing portfolio and adding the business logics. The algorithm goes through all the time steps and customers, and simulates the random events, usually a hundred times. This and other essential phases of the algorithm are depicted in Figure 2.

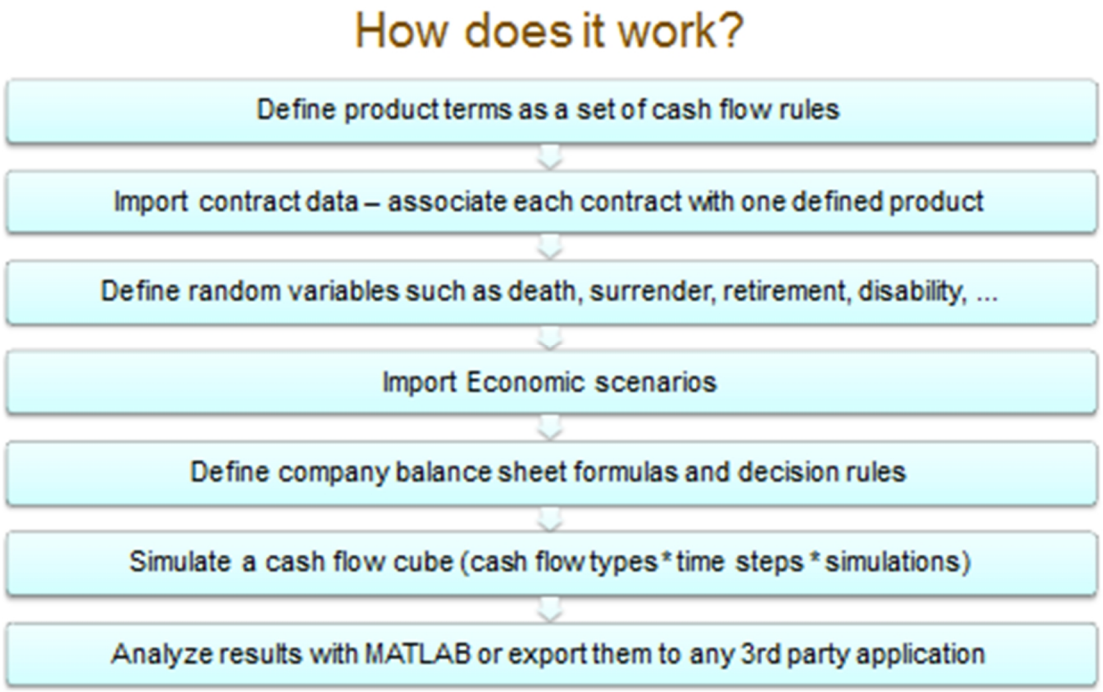

Figure 2. A flow diagram of mSII's operational principle. Source: Model IT

High performance computing in the cloud

The architecture of cloud platforms in networking and storage is designed for scalability and cost efficiency. Scientific and computation-intensive commercial applications have quite different system requirements than the web applications, for which the cloud platforms were initially designed for. Some require a lot of fast processors and have little communication during the process. For other applications parallel computations require intensive communication, that it is dominating factor for performance. The primary determinants for performance in a grid computing system, such as the one CSC has in Otaniemi, are thus communications latency and bisectional bandwidth.

Microsoft Azure

A group from Microsoft Research has investigated the feasibility of Azure for parallel computations. They are using an algorithm called Blast, which is one of the most widely used bioinformatics algorithms. For the implementation they developed a dedicated task parallel library, which provides concurrency and coordination support. They also suggested dividing the data into sufficiently small partitions in order to improve load balancing. (10)

The team tested the performance of different and multiple Azure instances. The performance grew super-linearly as the size of the problem instance grew. Because the cost grows linearly, it is most cost-effective to use the biggest instance. The reason for the super-linear behavior in this case was mostly due to increased memory capacity. The speedup of using 64 instances instead of one was almost linear; the calculations took 57 times longer with one instance. Although, the authors mention that the Blast algorithm is especially suitable for task parallelism, since the computations can be run independently without heavy communication. (10)

Amazon EC2

Rehr et al. (2010) examined the performance characteristics of Amazon's EC2 platform with different physics-related algorithms. First they noticed that the scaling of EC2's virtual machines was similar to physical clusters. Then they tested the serial performance with an n-body gravitational code, which has a lot of inter-process communication. Comparing to a similar HPC cluster, the extra-large EC2 instance performed on a same level. The authors speculated that the virtual instances might have had more memory capacity. This cannot be known for sure, because there are a variety of memory speeds deployed with the extra-large instance. Network characteristics measured by bandwidth and latency were slightly worse than those of a 100-Mbps Ethernet connection. (11)

Implications

All in all, the previous examples tell us that cloud computing is in principle a feasible way for applications, which are data and compute-intensive. It has some questions regarding the network performance and the uncertainty of the

resources the user gets each time. Using a lot of instances should however average the variance of, for example, different memory speeds. The algorithm of this project, mSII, needs inter-process communication only from time to time so the network characteristics should not be a big factor affecting performance

Economics of cloud computing

From economic perspective, perhaps the greatest promise of cloud computing is removing the large initial costs associated with building a compute clusters. Ideally the cloud computing environment enables the user to use as many compute instances as are needed for the task at hand. On the other hand, in conventional cluster installations, the number of computers must be decided beforehand. If the cluster is designed according to peak demand, most of the computing power will sit idle most of the time.

Most of the costs associated with traditional compute clusters are fixed and thus the cost per hour goes down as the compute-load increases. The pricing logic of the cloud computing service providers is completely opposite to the costs of operating a compute cluster. Pricing of cloud computing instances is based on usage in terms of time or amount of data. As such one might suspect that for predictable steady workload traditional compute clusters will emerge as the most cost-effective application. However, this is not immediately obvious. Costs incurred from operating a compute cluster include

- Up-front cost of hardware and continuous maintenance
- Electricity for operation and cooling
- Price of floor space for the computers

A large service provider such as a cloud computing service provider will most probably be able to get better deals on hardware and space. Large computing centers also tend to be more efficient in their power use. Table 1 presents a comparison between the costs incurred from cloud computing vs. those incurred from operating a HPC cluster.

How important is scalability and availability? Cloud computing provides an easy way to scale up a company's computing power to meet variable demand. However since the cloud resources are shared with many other users there is no guarantee that there are always enough resources available to deploy. This might mean that one needs to settle for fewer compute instances than would be optimal. The traditional HPC cluster on the other hand is not really scalable at all.

Table 1. Comparison of cost structures between cloud computing and traditional HPC cluster

	Fixed Costs	Variable costs
Cloud computing	Adapting existing software for the cloud	Costs incurred from using the cloud include: • Computing instance hours • Data transfer
HPC Cluster	Acquisition costs of • Hardware • Software • Server room	Costs incurred from HPC Cluster • Electricity • Maintenance

Things that have not been considered in this comparison are software license fees, possible costs of adapting the software for use in the cloud. License fees are not likely to change the outcome of the comparison, since licenses need to be purchased also for the cloud. The network usage of an application is difficult to estimate beforehand but should be taken into account when comparing local HPC server to Cloud computing.

These issues are however not likely to change the outcome of this simple cost-effectiveness analysis. For applications where demand for computing capacity has sharp spikes, the cloud seems very appealing. However, as the utilization of the computing resources grows, the price advantage of cloud computing shrinks. If the demand for computing capacity is relatively steady, owning a HPC server is likely to be the more cost-effective solution.

Table 2. Total cost of ownership estimate for a single HPC server.

Total cost of ownership		
Up-front	5000	Eur
Service contract per year	1000	Eur
Service years	5	years
Power consumption	0.5	kW/h
Power price	0.17	Eur/kWh
Cost per Year	2000	Eur

Table 3. Hourly cost estimates for single HPC server

Hours/Year	Eur/h
100	20.085
200	10.085
500	4.085
1000	2.085
2000	1.085
4000	0.585

Service providers

Microsoft Azure

The Windows Azure Platform consists of a cloud services operating system and a set of developer services. It claims to provide the functionality to build applications that span from consumer web to enterprise solutions. It contains four technologies which can be used individually or together to build solutions which run "in the cloud". The key technologies are

- Windows Azure
- SQL Azure
- Windows Azure Platform AppFabric
- Windows Azure Marketplace

Windows Azure provides, what is commonly called Platform as a Service (PaaS). Solutions can either run entirely on the platform or as a hybrid, with some of the solution running on-premise or elsewhere on the Internet. The code is run and the data is stored in datacenters. There are in total six datacenters, which are located quite evenly all over the world. (12)

Windows Azure is the cloud services operating system for the platform and it provides developers with on-demand compute and storage capacity. It is the service hosting and management environment. Azure integrates with Visual Studio (VS), which makes VS a convenient development environment for Azure hosted applications. Azure is supposed to be an open platform that supports both Microsoft and non-Microsoft languages and technologies. Third-party tools are, in principle, compatible with Azure.

Microsoft SQL Azure provides **r**elational data services in the cloud based on SQL Server. It delivers the capabilities of Microsoft SQL Server to Azure applications or applications running outside of the Azure Platform. SQL Azure is a cloud database built on SQL Server technologies, offered as a service, and running in Microsoft datacenters. It is scalable, with built in high-availability and fault tolerance. (13)

AppFabric in turn is a cloud-based infrastructure service for applications running in the cloud or on-premise. It provides connectivity as a service to help developers bridge cloud, on-premise, and hosted deployments. Marketplace is an online service for purchasing cloud-based data and applications. It consists of two parts which are DataMarket and AppMarket. It is also a channel for dataset and application providers to make their products available.

To summarize, Azure is an integrated development, service hosting and management environment maintained at Microsoft datacenters (see Figure 3).

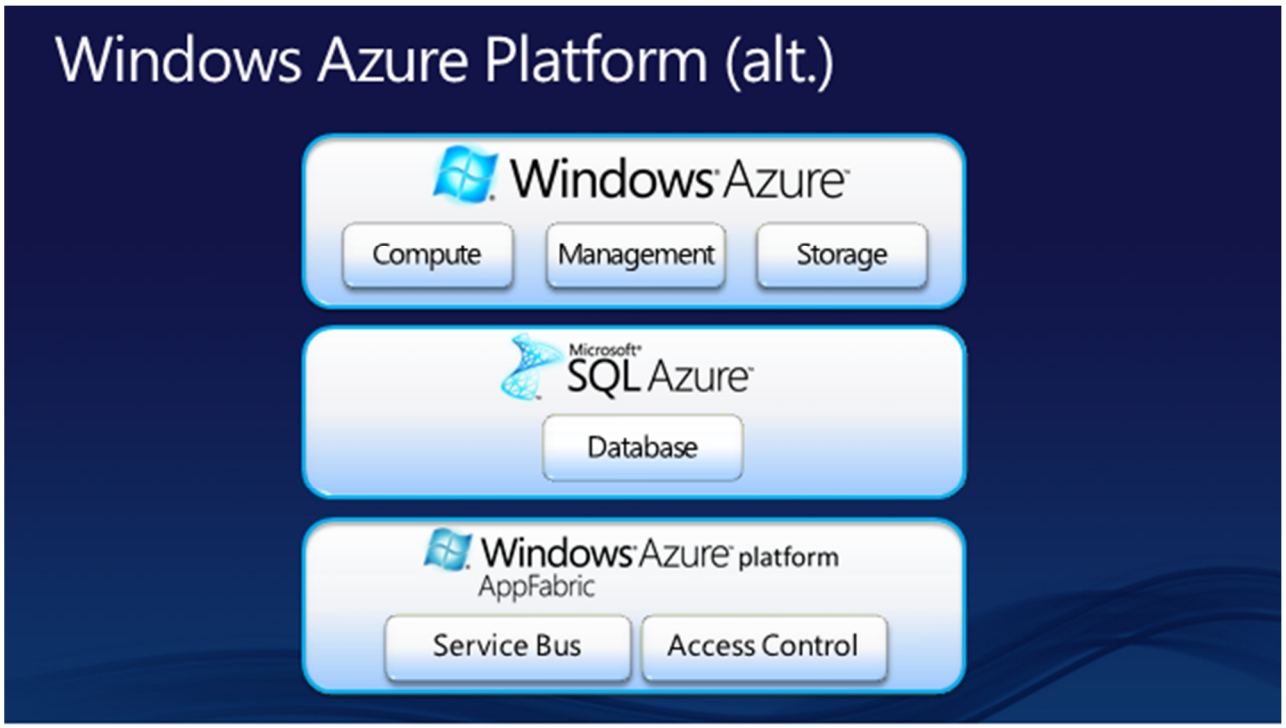

Figure 3. The structure of the Windows Azure platform and the functions of its components. Source: http://www.zdnet.com/blog/microsoft/microsofts-windows-azure-what-a-difference-a-year-makes/7051

Roles

An application built on the Windows Azure service is structured as one or more roles. When it executes, the application typically runs two or more instances of each role, with each instance running as its own virtual machine (VM). There are three kinds of roles:

- Web roles
- Worker roles
- VM roles

Web roles are primarily for running Web-based applications. Web role instances have Internet Information Services (IIS) pre-configured to run inside them. This feature makes the creation of, for example, ASP.NET applications straightforward.

Worker roles, as the name suggests, are designed simply to run code. A worker role might for example run a simulation or nearly anything else. A common implementation method for an application is, that it interacts with the user through a web role, and then hands tasks off to a worker role for processing.

VM roles can run images provided by the user with Windows Server 2008 R2. This role can be the right choice, if the intention is to move some on-premise Windows Server applications to Windows Azure.

When the user gives Azure an application to run, configuration information is submitted along with it. This information tells, among other things, the platform how many instances of each role to run. Then a VM is created for each instance, and the code is run for the appropriate role in each VM. Requests from the outside world are load-balanced across all instances of a role. (12)

For role instances, developers can choose from several VM sizes, each with a specific number of processor cores and memory (see Table 4). Since each instance can be assigned one or more cores, applications have predictable performance. (13)

Table 4. Windows Azure Compute Instance specifications and pricing. (14)

Compute instance size	CPU	Memory	Instance Storage	I/O Performance	Cost per hour
Extra Small	1.0 GHz	768 MB	20 GB	Low	$0.05
Small	1.6 GHz	1.75 GB	225 GB	Moderate	$0.12
Medium	2 * 1.6 GHz	3.5 GB	490 GB	High	$0.24
Large	4 * 1.6 GHz	7 GB	1 000 GB	High	$0.48
Extra Large	8 * 1.6 GHz	14 GB	2 040 GB	High	$0.96

Storage

Blobs are the simplest way to store data in Azure. There are two kinds of blobs to store text and binary data. Block blobs are optimized for streaming, and page blobs are ideal for random read/write operations. A storage account can have one or more containers, each of which holds one or more blobs (see Figure 4). Blobs can be big, up to 1 TB, and they can be subdivided into blocks, what makes transferring large blobs easier. (15)

Tables are another way to store data, and sometimes a more convenient option than blobs, which are somewhat unstructured. The data tables contain is stored in a set of entities with properties. Properties can have various types such as integer, string or Boolean. A single table can be quite large, with billions of entities holding terabytes of data. **Queues** are the third option, but they are used for a different purpose. The main use of queues is to let web role instances to communicate with worker role instances. (16)

Regardless of how it is stored—in blobs, tables, or queues—all data is replicated three times. This allows the fault tolerance, since losing one copy is not fatal. Storage can be accessed either by an Azure application or by an application running somewhere else. Compute and storage resources are charged independently. This means that an on-premises application can use just storage. (13)

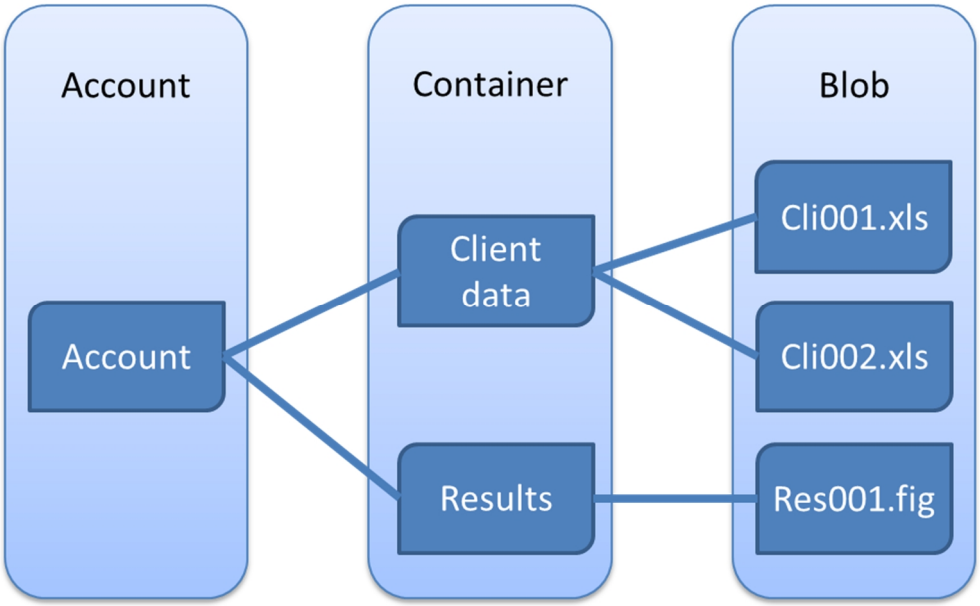

Figure 4. An illustration of a storage account

Amazon Elastic Computing Cloud

Amazon Elastic Computing Cloud (EC2) is a part of Amazon.com's cloud computing platform, Amazon Web Services (AWS). EC2 allows users to rent virtual computers on which to run their own computer applications. A web service is used to launch and manage Linux/UNIX and Windows server instances in Amazon's data centers. In this project EC2 has been an alternative for Windows Azure. But as our client has a partnership with Microsoft, Azure has remained as the alternative to take a closer look at.

EC2 communicates with the user through a web interface. It is a virtual computing environment quite independent from the operating system, whatever it may be. The application environment can be customized, or pre-configured settings can be used. The user can include libraries, applications and data. These are included in an Amazon Machine Image (AMI). Also security settings and network access should be relatively adjustable. After instance types and an operating system have been chosen, the user can start, terminate and monitor as many instances as needed. Multiple locations to run the applications can be determined. (17)

Amazon EC2 is advertised to be very elastic in the sense that computing capacity can be increased or decreased within minutes. The user should also have

complete control, root access, of her instances. Multiple instance types, operating systems and software packages are said to make the service flexible. Capacity can be automatically scaled according to conditions given by the user. AMIs and instances, storage and available user interfaces are discussed next in more detail.

Amazon Machine Images and instances

AMI is a template that contains a software configuration which means the operating system, the application server, and the applications, and the user can have as many AMIs as needed. Instances, on the other hand, are running copies of the AMI and are launched from AMIs. Multiple instance types can be launched from a single AMI. The choice of the instance type should be based on the amount of memory and computing power needed. Most common software configurations for AMIs are published by Amazon. The user can also use his own configurations. (18)

Instance types are specifications that define the memory, CPU, storage capacity and hourly cost for an instance (see Table 5). Some instance types are designed for standard applications, whereas others are designed for CPU-intensive applications, or memory-intensive applications.

Table 5. Amazon EC2 instance specifications and pricing. (19) More information about the instances can be found from: http://aws.amazon.com/ec2/instance-types/

Instance	Linux/UNIX Usage ($/h)	Windows Usage ($/h)
Micro	0.02	0.03
Small	0.085	0.12
Large	0.34	0.48
Extra Large	0.68	0.96
XL (hi-memory)	0.50	0.62
Double XL (hi-memory)	1.00	1.24
Medium (hi-CPU)	0.17	0.29
XL (hi-CPU)	0.68	1.16
Cluster Compute	1.60	N/A
Cluster GPU	2.10	N/A

Storage

The two most commonly used storage types, when using EC2, are Simple Storage Service (S3) and Elastic Block Store (EBS).

Amazon S3 provides a simple web service interface that enables the user to store and retrieve any amount of data from anywhere on the Web. It is intentionally built with a minimum feature set. Objects can be written, read and deleted, and each object can contain 5 terabytes of data. The number of objects is unlimited. Each object is stored in a bucket which is located in one of the datacenters. S3

gives access to the same scalable infrastructure that Amazon uses to run its own global network of web sites. (17)

Amazon EBS Volumes provides instances with persistent, block-level storage. They are essentially hard disks that can be attached to a running instance. In fact, multiple EBS volumes can be attached to a single instance. Volumes are especially suited for applications that require a database, a file system, or an access to raw block-level storage. Snapshots (sort of backups) can be created of EBS volumes to S3. (17)

Databases are also supported. There are two common ways to implement a database for the application: Amazon Relational Database Service (RDS) or to launch an instance of database AMI and use that EC2 instance as the database. RDS gives access to the capabilities of a familiar MySQL database, so that user's existing MySQL databases should work with Amazon RDS. (18)

User Interfaces

There are three different interfaces to access EC2. The AWS Management Console, Command Line Tools (API Tools) and Programmatic Interface. The AWS Management Console is a simple web-based GUI that serves as a point-and-click web-based interface. Command line tools provide a Java-based command-line client. EC2 can also be accessed programmatically, as service development kits are provided e.g. for Java and .NET. There are third-party libraries available for these programmatic interfaces. (17), (20)

Azure versus EC2

The biggest difference between Azure and EC2 is that Azure is classified as Platform as a Service (PaaS) and EC2 is Infrastructure as a Service (IaaS). Azure offers effectively a virtual server on which to load software, which can be accessed and managed through a web browser. EC2 should be able to offer same things but with greater flexibility, as there are many operating system and application options. One of the biggest benefits with EC2 is the ability to package own images with whatever custom software needed. Amazon pricing is maybe a little more complex than that of Microsoft's.

Both services are designed to be scalable, but the way they achieve this, is different: Azure users have the choice of scaling up the number of Virtual Machines on which an application runs, or by increasing the power of the VM they use. EC2 scales in blocks called Elastic Compute Units (ECU), which include a specific number of Amazon Machine Images and storage volumes (Elastic Block Stores). Azure is more suitable for developers who are experienced in Microsoft environments and technologies such as .NET and Visual Studio. While Azure offers only Microsoft operating systems, Amazon supports a variety of operating systems. (21)

Other cloud providers

There are also other service providers among Microsoft and Amazon. Google's AppEngine (GAE) is quite similar to these two big players, but the difference is in the infrastructure it offers. GAE gets rid of the whole concept of an instance. User does not have to predetermine or pre-configure the number of instances needed. In order to keep costs reasonable, GAE s provides quotas that the user administers. Service development kits are available only for Java and Python. (22)

Other providers, which offer IaaS (Infrastructure), are for example Rackspace Cloud, GoGrid and FlexiScale. In addition, there are a lot of services who offer SaaS or PaaS (Software and Platform). The choice of the cloud provider depends highly on the computing problem in hand.

Approach

Theoretical limits of program speed-up

Perhaps the most common reason for running programs in parallel is to achieve faster running times. This gain from using N cores or computers instead of a single core is commonly measured by speedup

$$S = \frac{T_1}{T_N},$$

where T_1 and T_N are respectively the running times of the program on a single core and N cores.

Each program or algorithm consists of parts that can be run in parallel and those that cannot be run in parallel. The latter parts of the program make up the parallel fraction of the program. If the program has a parallel fraction P the maximum theoretical speed-up that we can receive from running the code on N processors given by Amdahl's Law (23)

$$\frac{1}{1 - P + \frac{P}{N}}.$$

In the limiting case of infinitely many processors, the speedup is directly inversely proportional to the fraction $(1 - P)$ of the code that is not parallelizable. Figure 5 shows this theoretical speedup as a function of number of processors for various fractions. It can be seen that even for highly parallelizable program, with $P = 0.9$, the additional speedup from adding more processors diminishes quickly. Here P is assumed to be constant for all values of N, but in reality this is not necessarily the case. For example, as the number of processors grows, various communications related latencies are also likely to increase.

The Monte Carlo simulation methodology employed in mSII means that we can expect a value of P close to 1. However there are parts of the program that are not parallelizable: After each time step (one year or more of simulation) data is gathered from all the worker instances and analyzed. Business decisions are made with this data, and new tasks are distributed.

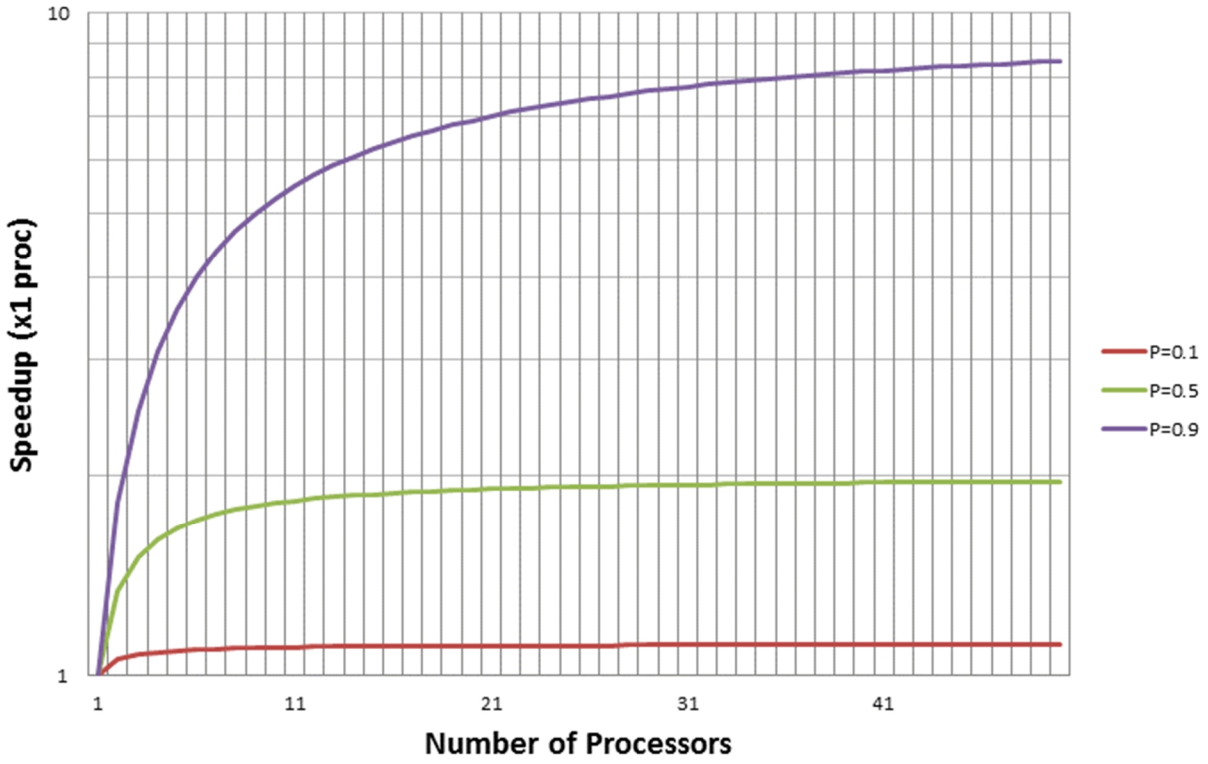

Figure 5. Speedup as a function of parallel fraction *P* and number of processors N

Parallel computing issues

In the previous section it was shown that under certain assumptions the speedup resulting from parallelization can be significant. In practice, several problems need to be solved in order to achieve these theoretical speedups. The most important problem is how to distribute the computational tasks across the worker nodes. Consider two nodes working on N computational tasks. A naïve approach would be to give $N/2$ tasks to each node and wait until they are finished. However, if the computational tasks take a variable amount of time to finish, it is possible that one of the nodes will finish all its tasks long before the other and will then sit idle, while the other node calculates its remaining tasks. To avoid this situation, some kind of load balancing should be used.

Communication issues become important, especially when transitioning from shared memory parallelism to distributed memory parallelism. In distributed memory systems, the computational nodes do not share memory. Most commonly this means that the nodes reside on different physical computers. In this case, all communication between nodes must pass through a network interface of some sort. This brings significantly greater latencies than using shared memory. It is therefore important to minimize the amount of messaging between nodes.

Another issue, which might arise with running massively parallel programs, is that of nodes crashing. To see how this could be an issue, assume that a single instance will fail during the simulation with probability p. When running N

machines in parallel, the probability that none of them will fail is $(1-p)^N$. The failure may be due to hardware failure or network failure, and the failure probabilities are assumed to be independent. Some form of error recovery strategy might be useful for very large N, or if the calculations take a very long time to complete.

Parallel computing with MATLAB

MATLAB is a widely used environment for data analysis and for developing numerical algorithms. Many of these algorithms can benefit greatly from parallelization. Monte Carlo simulations and parametric sweep applications are particularly amenable to solving in parallel.

MATLAB provides the 'parfor' construct for running loops in parallel. This means that, ideally, parallelizing a MATLAB program only requires changing one for-loop into a parfor loop. The same construct can be used for both shared memory and distributed memory systems. However, in the latter case the MATLAB Distributed Computing Server is needed.

MATLAB has not always supported parallel computing and there exists several legacy software packages for parallel computations on MATLAB. One notable software package is MATLABMPI, which implements in MATLAB a subset of the Message Passing Interface (MPI) library commonly used for parallelization of HPC applications. MPI is intended for distributed memory parallelism and represents a relatively low level of parallelism.

Modifying programs to use MPI is a considerably more involved exercise than using the MATLAB Parallel Toolbox, since the user now has to explicitly handle distributing the tasks and gathering the results. The upside is that the user retains more control over the execution of the program, and MATLAB Parallel Toolbox or MATLAB Distributed Computing Server is not needed.

MATLAB and Azure

In theory, parallelism in the Azure environment can be implemented using the MATLAB Distributed Computing Server and MATLAB Parallel Toolbox. At the time of writing this, the MATLAB Distributed Computing Server does not work on Azure. This leaves the Parallel Toolbox for shared memory parallelism on a single computing instance which does not require the MATLAB Distributed Computing Server.

Since we are interested in running Monte Carlo simulations utilizing a large number of instances, we must turn our attention to Message Passing solutions. In this approach, several instances of the program are launched in the cloud. One of the instances is deemed the 'master' and the rest are 'workers'. The master instance handles distributing the computing tasks to the workers.

As the term 'Message Passing' implies, all this is handled by passing messages between the instances. MPI libraries running on traditional HPC clusters use TCP/IP communications. This is hard to do with MATLAB, and even harder to accomplish in Azure. The Azure provides several methods for passing data between instances, and to and from the cloud. These methods include queues and blobs.

Test scenarios

Several parameters need to be considered when programs are run in parallel. These things include distributing the tasks across the worker nodes, and data transfer methods between the worker and master nodes. Attention should also be paid to the amount of physical memory on each worker node.

It can be difficult to determine the parameter P in Amdahl's law theoretically. However, we can construct a series of experiments to determine the parallel fraction P, and from there the maximum possible speedup by Amdahl's law. It must be noted that the Amdahl's law does not take in to account any issues with program scaling. For example, most cloud computing service providers provide instances with at most 8 cores. Going from doing the calculation on a single machine to distributing it across several physical machines will introduce new delays in form time needed to copy the data across the network. Thus we should be careful to differentiate between simulations running on a single machine and simulations running on several physical or virtual machines.

Effect of data transfer on calculation times can be determined for example, by running the same case first on 8 cores with shared memory and followed by a run on 8 separate instances connected through a network. This can be repeated for different numbers of cores and instances. This way we can determine the parallel fraction P for simulations running on networked computers. The decrease in parallel fraction can be attributed completely to the communications routines. Depending on how the parallelization is implemented, a test of different communications strategies should be tested. If possible, the nodes should talk to each other by TCP/IP connections, since these are usually free (at least up to a data transfer limit). However, Windows Azure also provides queues and blobs for communications purposes.

Azure experimentations

The emphasis of this project was on the technical implementation. Projects and packages along with their configuration files were built using Visual Studio, which means that C# programming skills were essential. Another part of the problem was compiling MATLAB files into a format compatible with Azure. The MATLAB runtime library had to be included in the Azure project, which increased the size of the project significantly. There were two approaches to the runtime library matter: to add the runtime library installer to the project, or to add the files included in the runtime library directly to the project.

Most of the resources were directed to technical configuration and implementation. Nonetheless, problems encountered during this phase caused major delays to the revised project plan. Since our team had only a limited time reserved for the project, we had to abandon the testing phase. Technical issues were mentioned as the biggest risk of the project, but there were also unexpected problems. Some of the programs were 32-bit and some 64-bit, which caused compatibility issues.

The projects were tested and debugged in Visual Studio environment (screen capture of the graphical interface is in Figure 6). They could be run in a specific emulator before trying to deploy them in the cloud. After getting acquainted with Azure, our aim was to include our own .m-files in the cloud projects. Approaches applied are discussed more thoroughly in the next chapters.

Web and worker roles were experimented and some differences with functionality emerged. It would be of great importance to get better and more thorough error messages by default, if a deployment should fail. Also the size of the runtime library (several hundred megabytes) slowed the deployment process, which lasted generally from 60 to 90 minutes.

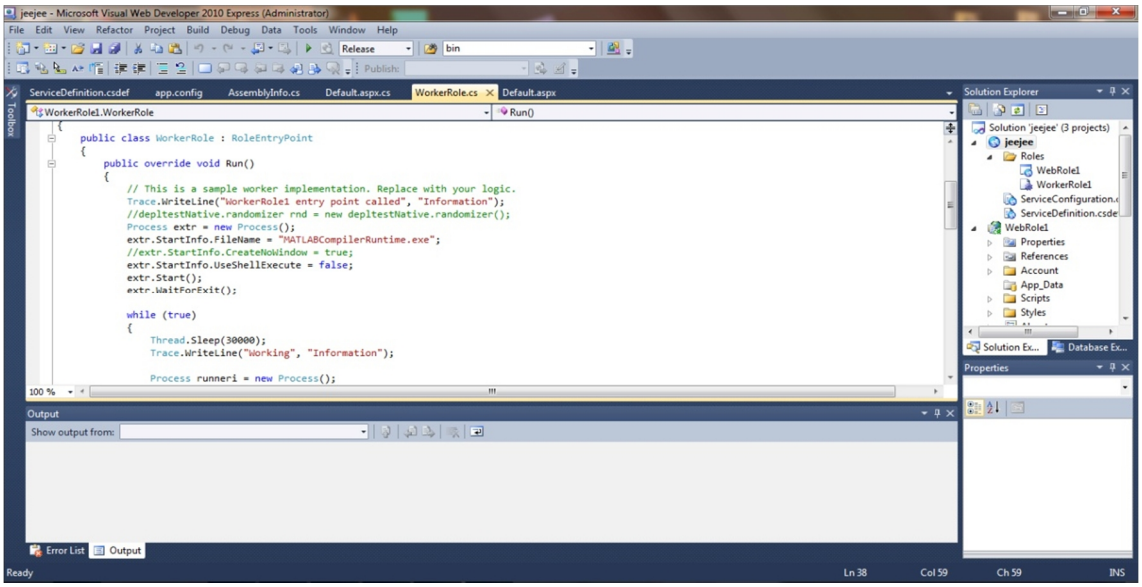

Figure 6. Screen capture of the Microsoft Visual Web Developer.

Tutorial project

In order to get used to Azure, we completed a small tutorial project called HelloCloud provided by Microsoft. The purpose was to learn how to create a basic ASP.NET application, and then how to deploy the application to Azure. Before the tutorial, we naturally had to make an Azure account and install the necessary tools, which include Azure SDK for Visual Studio, and also extensions to Visual Studio for debugging and packaging. A screen capture of Microsoft Azure Platform can be found in Figure 7.

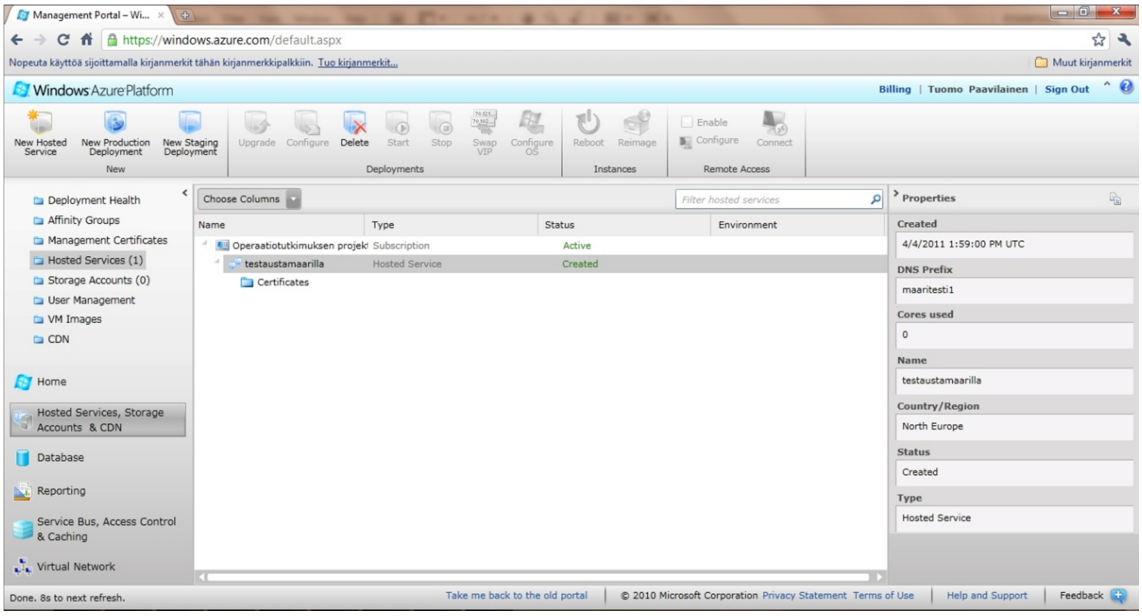

Figure 7. Screen capture of Windows Azure Platform.

The HelloCloud project itself was essentially an ASP.NET web role. It just printed “Hello Cloud” that was shown in web interface. The project was actually a Visual C# template. HelloCloud was first compiled, and run in the local development environment. The Azure compute emulator allows users to debug the application locally. This enables tasks such as viewing a running instance of the local deployment. The compute emulator should start automatically when debugging an Azure project. (24)

Two important files in a project are .csdef and .cscfg. These files are automatically created when the project itself is created. The .csdef extension is for the cloud service definition files. It contains metadata used by Azure when hosting the user’s application. Metadata tells Azure for example, which roles are in the deployment. The .cscfg extension is for the cloud service configuration files. It provides configuration settings for the application as well as the number of instances to run for each role. (24)

To deploy the application to Azure, a service package and a service configuration file must be uploaded. When the project is successfully compiled, also a .cspkg file is created. The extension .cspkg is for the service package files and a package file contains the service definition as well as the binaries and other items for the

application being deployed. After this the deployment process is quite straightforward.

MATLAB files into .NET library files

MATLAB files must be compiled into a suitable format before uploading into Azure. Our first idea was to compile .m-files into a compatible format with the .NET framework. MATLAB has a deploy tool for this purpose, called deploytool. The result of the compiling process was a .dll file, which is Microsoft's implementation of the shared library concept in the Windows operating system.

The first major problem encountered when trying to build Azure projects including these .dll files was the lack of MATLAB runtime library. Our approach to this was to add the runtime library installer to the project and configuring the project so that the installer was run. In the emulator this worked but we could not get it running correctly on the Azure platform. The reason for this was that the emulator could use the MATLAB and its libraries already installed on the computer. The cloud application on the other hand could not find the necessary libraries to perform the simple calculations defined in the .m-file.

As opposed to Amazon's EC2, users do not have administrative rights to the instances in Azure. This means that there is no root access and we could not install software in the traditional sense. Also, the debugging process in the cloud was laborious due to the size of the project and it was hard to get adequate error messages after the deployment had failed. After failure, the deployment had to be done again from the beginning. This process is illustrated in Figure 8.

At first we worked using the web role, since it was familiar from the tutorial and easier to use. The web interface was not a necessity for our eventual purpose, so we turned our attention to the worker role. In theory, everything should have worked analogically with respect to web role, but some problems emerged nonetheless. These problems could be traced to compatibility issues between 32 and 64-bit systems. We got the worker role finally running correctly with the emulator but not in the cloud. The same reasoning applies with this problem as with the web role.

The difficulties with this approach seemed insurmountable and we had to look for a different solution. We decided to try compiling the MATLAB files into executable files (.exe).

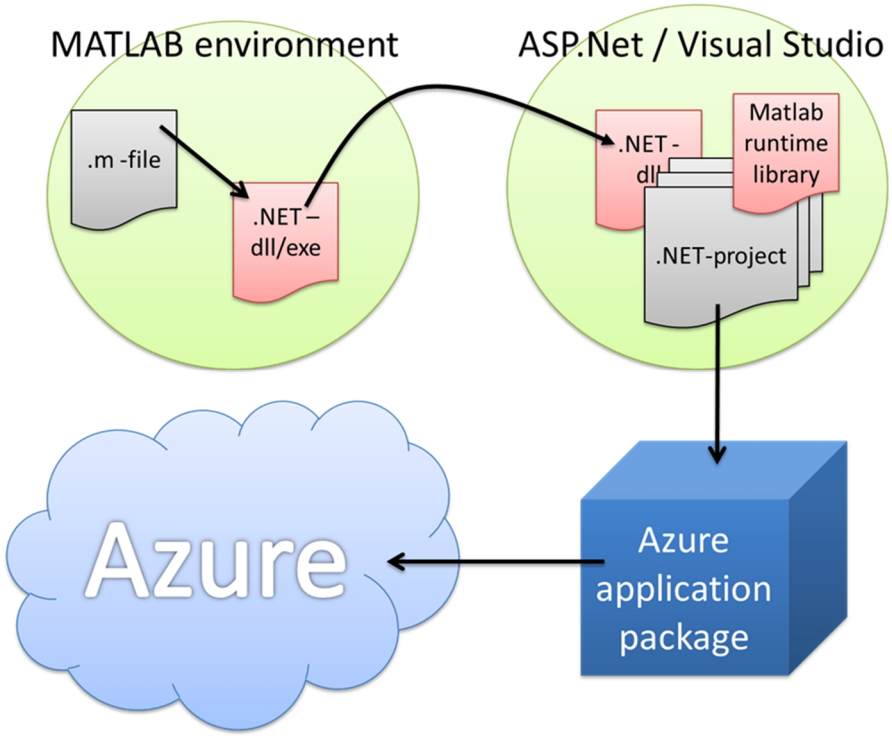

Figure 8. The deployment process

MATLAB files into Executable files

The MATLAB deploytool can also compile .m-files into executables. For this purpose, we had to first install a new C++ compiler. Once again, we had problems with 32 and 64-bit systems but reinstalling some of the tools with consistent bit systems helped. We tried to include the runtime library files in the .exe-files using the deploytool. This did not succeed, although giving the process certainly enough time by waiting overnight. This could have been a result of memory problems, as there were a lot of files to be included in the .exe.

Self-extracting package

Finally we decided to move the MATLAB runtime libraries to Azure using a self-extracting package. The idea was to include two executable files in the Azure package: an exe-file compiled using the MATLAB-compiler and a self-extracting package for transferring the runtime libraries needed by the other. The execution of these program files was controlled by an Azure worker role written in C#.

The reason to try this approach was that it was practically impossible to include the runtime libraries directly into the Azure project due to inadequate user interface and documentation of Windows Azure. The other reason behind this approach was the fact that a self-extracting package is much smaller than the directory structure of the runtime library, thus speeding up the obsequious upload process. Unfortunately this approach also failed to run in the production

environment in the cloud. It seems that the MATLAB .exe-file was unable to locate the necessary files in the folder structure of the runtime library. At this point, we were under a heavy time table pressure to proceed to starting the final report.

Conclusions and discussion

Financial aspects

Cloud computing is not always the best option for satisfying ones computing needs. For applications where computational capacity is needed in short bursts, the cloud offers a very cost-effective solution. However if there is need for constant computing capacity then it will likely be cheaper to own a HPC server or cluster. Our clients' application is both highly amenable to parallelization and needs computing capacity in relatively short bursts. While more careful financial calculations and technical tests still need to be run, it seems that cloud services would be a good fit for applications such as the mSII.

Technical aspects

Due to the large number of technical difficulties encountered during the course of are project, we do not have first-hand information on running MATLAB programs on the cloud. However, during our literature review we found white papers describing other groups' efforts at running their simulations on the cloud.

One important difference between several cloud instances and a HPC cluster is the lack of low latency communications. This means that for cloud computing to be realistic alternative, communications between processes must be minimized. This is especially true if communications is implemented, for example, by using Azure queues or Tables. These methods are not only slow compared to direct network connections, they also cost more.

Based on our own experiences

Our thoughts about the usability of Azure are somewhat mixed. The user interface is designed to be as simple as possible, which makes Azure approachable. Visual Studio has debugging tools which help while developing the project. However in case of difficulties with project Deployment, adequate information about errors can be difficult to obtain.

One frustrating feature occurs while deploying project on Azure. If the project is big enough, it can take over an hour get it running. During that time no detailed information about the progress of the deployment process is given. One can only guess how many additional hours it is going to take or has the deployment crashed.

Documentation provided on the Azure web page is useful and fairly extensive. It covers well all the principles behind the service, such as main features and instance types. There are white papers available and also some example projects. One can also try Microsoft's technical support forums or some other developer community in case of impasse. Naturally it would be impossible to provide Azure users with detailed instructions in every situation, so

documentation did not really help us with our problems with MATLAB and Azure.

As we have no experience with other cloud service providers, it is impossible to make any comparisons regarding usability. Still, our understanding is that there is considerably space for improvement.

Next steps

If MathWorks publishes its Distributed Computing Server for Microsoft Azure, lot of the problems encountered in this project will most likely go away. To overcome the problems encountered in using the MATLAB Compiler, more cooperation with MathWorks is most likely necessary.

The Microsoft Azure Platform seems to be mainly intended for hosting web based services and not for high performance computing. Other cloud computing service providers, such as Amazon, seem to offer more flexibility and control over the computing instances. They also provide a wider range of instance types with different amounts of memory and most importantly for us: faster processors. In the future other options besides Microsoft Azure should be explored.

Bibliography

1. **P. Boyle, M. Broadie, P. Glasserman,** 1997. *Monte Carlo methods for security pricing*. Journal of Economic Dynamics and Control, pp. 1267-1321.

2. **M. Koivu, T. Pennanen and A. Ranne,** 2005. *Modeling assets and liabilities of a Finnish pension insurance company: a VEqC approach.* Scandinavian Actuarial Journal, pp. 46-76.

3. **P. Hilli, M. Koivu and T. Pennanen,** 2007. *A stochastic programming model for asset liability management of a Finnish pension company*. Ann Oper Res, pp. 115-139.

4. **European Commission,** 2011**.** *Solvency II: Basic architecture*. [Online] [Cited: 3 30, 2011.] http://ec.europa.eu/internal_market/insurance/solvency/architecture_en.htm.

5. **D. Bauer, D. Bergmann and A. Reuss,** 2009. *Solvency II and nested simulations – a least-squares Monte Carlo approach.*

6. **Grance, P. Mell and T,** 2009**.** *The NIST definition of cloud computing*. [Online] [Cited: 4 4, 2011.] http://csrc.nist.gov/groups/SNS/cloud-computing/.

7. **M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, Ar. Rabkin, I. Stoica and M. Zaharia,** 2009**.** *Above the clouds: A Berkeley view of cloud computing*. [Online] [Cited: 4 4, 2011.] http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html.

8. **C. Vecchiola, S. Pandey and R. Buyya,** 2009**.** *High-performance cloud computing: A view of scientific applications.* [Online] [Cited: 4 4, 2011.] http://arxiv.org/abs/0910.1979.

9. **Model IT,** 2010. *Insurance analysis.* [Online] [Cited: 4 12, 2011.] http://www.modelit.fi/web/en/Vakuutusanalyysi.

10. **W. Lu, J. Jackson and R. Barga,** 2010**.** *AzureBlast: A case study of developing science applications on the cloud.*

11. **J. J. Rehr, F. D. Vila, J. P. Gradner, L. Svec and M. Prange,** 2010. *Scientific computing in the cloud.* Computing in Science & Engineering.

12. **Microsoft,** 2011**.** *Windows Azure platform FAQs.* [Online] [Cited: 4 26, 2011.] http://www.microsoft.com/windowsazure/faq.

13. **Microsoft,** 2011. *Whitepapers. Introducing the Windows Azure platform.* [Online] [Cited: 4 26, 2011.] http://www.microsoft.com/windowsazure/whitepapers/introducingwindowsazureplatform/default.aspx.

14. **Microsoft,** 2011. *Windows Azure compute*. [Online] [Cited: 3 15, 2011.] http://www.microsoft.com/windowsazure/compute/default.aspx.

15. **Microsoft,** 2010. *The Windows Azure platform: Articles from the trenches*.

16. **Wikipedia,** 2011. *Azure Services platform*. [Online] [Cited: 4 22, 2011.] http://en.wikipedia.org/wiki/Microsoft_Azure.

17. **Amazon Web Services,** 2011**.** *Amazon Elastic Compute Cloud (Amazon EC2)*. [Online] [Cited: 4 22, 2011.] http://aws.amazon.com/ec2/.

18. **Amazon Web Services,** 2011**.** *Amazon Elastic Compute Cloud, User guide. API version 2011-02-28*.

19. **Amazon Web Services,** 2011. *Amazon EC2 pricing*. [Online] [Cited: 4 26, 2011.] http://aws.amazon.com/ec2/pricing/.

20. **Wikipedia,** 2011. *Amazon Elastic Compute Cloud*. [Online] [Cited: 4 22, 2011.] http://en.wikipedia.org/wiki/Amazon_EC2.

21. 2011. *5 key differences. Microsoft vs. Amazon cloud*. [Online] [Cited: 4 22, 2011.] http://velocha.com/5-key-differences-micrososft-vs-amazon-cloud/.

22. 2009. *Comparing clouds: EC2, Azure, App Engine*. [Online] [Cited: 4 22, 2011.] http://loosexaml.wordpress.com/2009/11/25/comparing-clouds-ec2-azure-app-engine/.

23. **G. Amdahl.** 1967. *The validity of the single processor approach to achieving large-scale computing capabilities*. Proceedings of the AFIPS Spring Joint Computer Conference. pp. 483-485.

24. **Microsoft,** 2011**.** *Creating and deploying an ASP.NET application in Windows Azure*. [Online] [Cited: 4 22, 2011.] http://msdn.microsoft.com/en-us/library/gg651132.aspx.

25. **Microsoft,** 2011. *Whitepapers*. [Online] [Cited: 4 26, 2011.] http://www.microsoft.com/windowsazure/whitepapers/.