Proc. 2009 Computer Measurement Group Conf, Dallas, TX, Dec. 7-11, 2009.

Understanding Cloud Computing: Experimentation and Capacity Planning

Daniel A. Menascé Paul Ngo

Dept. of Computer Science, MS 4A5
The Volgenau School of IT & Engineering
George Mason University
Fairfax, VA 22030, USA
menasce@gmu.edu pngo1@gmu.edu

Abstract

Cloud computing is based on the notion of shared computational, storage, network, and application resources provided by a third party. This paper explores in detail the concept of cloud computing, its advantages and disadvantages and describes several existing cloud computing platforms. It then discusses the results of quantitative experiments carried out using PlanetLab, a cloud computing platform. The paper also discusses how the methods of capacity planning are impacted by the advent of cloud computing from the point of view of the cloud user and from the cloud provider.

1 Introduction

Cloud computing is a relatively new concept but has its roots in many not so new technologies. One of the main tenets of cloud computing is the sharing of computing resources among a community of users. A successful predecessor of the idea is the Condor project that started in 1988 at the University of Wisconsin-Madison [31]. This project was motivated by the observation that a high percentage of the capacity of user's workstations is idle while their users are away of their offices or doing other tasks such as reading or talking on the phone. These idle cycles can be harvested by the Condor system and made available to users who need more computing power than that available to them at their local workstations.

Another technology related to cloud computing is grid computing. The grid is defined as a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities [14]. It became the main computing paradigm for resource-intensive scientific applications and more recently for commercial applications [11]. Fred Douglis points out that although grid computing and cloud computing are closely related, they are indeed truly distinct [6]. Resource allocation issues are crucial to the performance of applications on the grid. See [12] for a description of heuristic techniques for optimal allocation of resources (i.e., computing, network, service providers, and secondary storage) in grid computing.

One of the technologies that has enabled cloud computing is virtualization because it allows for easy isolation of applications within the same hardware platform and easy migration for purposes of load balancing. Isolation is important for security concerns and load balancing is important for performance considerations. Service oriented architectures (SOA) and Web services are also an important development for building clouds that provide services as opposed to just computing resources.

This paper discusses the concepts of cloud computing as well as its advantages and disadvantages. In order to provide a more concrete example of the benefits of cloud computing, the paper shows results of experiments conducted on PlanetLab, a cloud infrastructure widely used in academia. The paper then discusses how cloud users can optimally select the values of Service Level Agreements (SLAs) to be negotiated with cloud providers in order to maximize their utility subject to cost constraints. A numeric example is thoroughly discussed.

The rest of the paper is organized as follows. Section 2 discusses the definition of cloud computing as well as its advantages and disadvantages. Section 3 briefly describes some examples of cloud computing platforms. The next section discusses the results of experiments carried out with PlanetLab. Section 5 discusses capacity planning issues as they apply to cloud computing. Section 6 presents some concluding remarks.

2 What is Cloud Computing

Cloud computing has many different meanings for different people. However, a basic definition that encompasses virtually all definitions is the following: Cloud computing is

a modality of computing characterized by *on demand* availability of resources in a dynamic and scalable fashion. The term resource here could be used to represent infrastructure, platforms, software, services, or storage. The *cloud provider* is responsible to make the needed resources available on demand to the *cloud users*. It is the responsibility of the cloud provider to manage its resources in an efficient way so that the user needs can be met when needed at the desired Quality of Service (QoS) level. For example, an infrastructure cloud offers computing infrastructure, typically in the form of virtual machines allocated to physical servers, as needed by its users. Cloud users are charged, in most cases, by how much resources they consume (e.g., $ per hour of CPU usage).

An analogy between cloud computing and the power grid is useful to capture some of the similarities but also some important distinctions. Consumers use electric energy on-demand according to their needs and pay based on their consumption. The electric power utilities (analogous to cloud providers) have to be able to determine at each point in time how much energy to generate in order to be able to match the demand. A variety of models that include weather prediction models and historic data on power consumption for each day of the year and each hour of the day drive the decisions made by utilities.

The analogy breaks in some of important aspects. The product delivered by the power grid is homogeneous (e.g., 110 V of alternating current at 60 Hz). On the other hand, computing clouds offer a variety of resources on demand. Another important difference has to do with the interface and plug-compatibility. One can plug any appliance to the power grid and it will work seamlessly as long as it conforms to a very simple specification of voltage and frequency. The same is not true with computing clouds. The APIs offered by cloud providers are not standardized and may be very complicated in many cases. In other words, computing clouds are not "plug-and-play" yet.

The following are some of the advantages of cloud computing:

- *Pay as you go*: Companies can avoid capital expenditures by using cloud resources on an as needed basis. In the owned approach, the total cost of ownership (TCO) is a function of three main components: 1) initial capital investment for hardware, software, networking, facilities infrastructure including cooling and power, 2) operational costs which includes hardware and software maintenance, personnel cost (including system, network, database administrators, capacity planning analysts), power consumption, and depreciation, 3) system upgrades required to cope with the growth of existing workload and/or new workloads.

- *No need to provision for peak loads*: If cloud computing resources are used, the responsibility to support peak loads at agreed upon service levels rests with the cloud computing provider.

- *Time to market*: Because users of cloud computing resources do not need to procure, install, and test all the infrastructure including middleware and applications in many cases, they can be up and running in very little time.

- *Consistent performance and availability*: When services are provided by the cloud under strict SLAs that are specific on response time and availability, users do not need to worry so much about maintaining adequate levels for these metrics. This burden is shifted to the cloud, which, by virtue of managing a typically large infrastructure may be able to autonomically shift resources (e.g., virtual machines) to keep up with varying and unpredictable workloads [7].

The potential drawbacks and or concerns regarding cloud computing are:

- *Privacy and security*: Many organizations may be concerned about having their sensitive data living in the same platforms as that of their competitors. There may be concerns regarding exposing a company's private data to the cloud computing provider. In some cases, a company may be bound to several types of regulations (e.g., HIPPA) whose responsibility cannot be easily delegated to a third party provider.

- *External dependency for mission critical applications*: Even when cloud providers offer to adhere to strict SLAs and pay penalties for non-compliance, cloud users may be concerned about trusting some of their mission critical applications to a third party.

- *Disaster recovery*: Users of resources in a cloud need to have guarantees that the provider has adequate backup and disaster recovery plans that will prevent a disruption of a user's activities in the face of natural or man-made disasters.

- *Monitoring and Enforcement of SLAs*: Negotiating, monitoring, and enforcing SLAs may be challenging in cloud computing because cloud resources and services are shared by a multitude of users and because providers have little control over the workload intensity of the different cloud applications.

3 Examples of Cloud Computing Platforms

This section discusses some examples of cloud computing platforms.

3.1 PlanetLab

PlanetLab is a virtual lab network that builds on top of the concepts of grid computing, distributed computing, and utility computing to support large-scale research and development using the service-on-demand paradigm [24]. Since its establishment in March of 2002, PlanetLab has grown tremendously from just over 40 sites and 100 nodes to 425 active sites [25] with 985 nodes [26] that scatter across 40 countries. PlanetLab currently hosts over 246 active research projects and experiments from various disciplines [27]. Many researchers around the world use PlanetLab as an overlay testbed to conduct scientific experiments, gather statistical data, and validate the results.

PlanetLab affords experimental researchers the possibility of conducting computationally intensive experiments they were not able to conduct before due to the lack of supercomputing power. By taking advantage of shared resources, such as CPU cycles, storage, and memory, combined from multiple nodes in the PlanetLab environment, researchers are able to run their experiments in a distributed and parallel fashion with less cost. Also, because of the distributed nature of cloud computing, PlanetLab is able to improve scalability, availability, reliability, and performance.

PlanetLab provides common abstractions to use its resources via the PlanetLab Central API [29]. This API allows users to create automated scripts to ease the deployment and monitoring of applications running at multiple nodes. Researchers can create a script to monitor node availability and take appropriate actions based on predefined criteria. If the monitoring process detects that a node is down, it can dynamically find another, add it to the set of already allocated PlanetLab resources, install any needed tools, and then deploy the applications to it. This way, an application can be up and running within minutes.

3.1.1 Structure

Figure 1 illustrates the structure of PlanetLab, which consists of a number of physical sites scattered all over the world. Each site has at least one PlanetLab *node*, which runs components of PlanetLab services. A *slice* is a set of allocated resources distributed across PlanetLab nodes. PlanetLab offers no guarantee related to the period of time during which these resources will remain allocated. When a user logs into a PlanetLab node, he or she has to compete for resources with the current users. Therefore, without fixed resources allocated to a slice on a particular time interval, performance measurement of an application on PlanetLab can be quite challenging.

3.1.2 Business Model

The PlanetLab Consortium is formed by trusted academic institutions and research organizations around the world

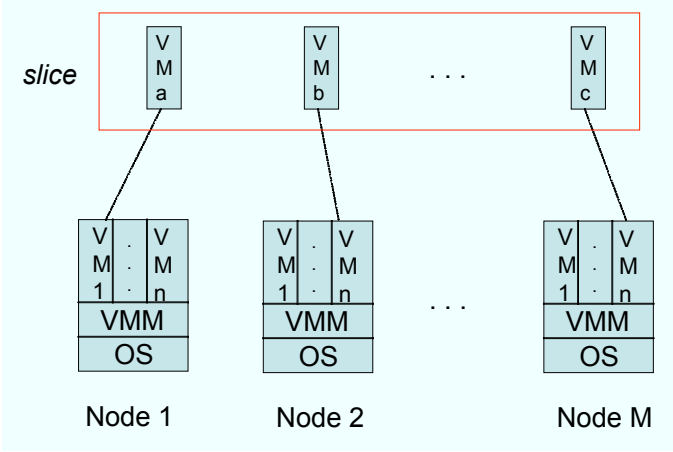

Figure 1: PlanetLab's structure: nodes and slices.

that jointly provide a common platform for conducting research and scientific experiments. Unlike most other cloud computing platforms, PlanetLab does not follow the pay-as-you-go model. The consortium consists of five membership levels [28]:

1. Charter ($300K annual dues): affords a permanent seat on the Steering Committee, unlimited number of slices, access to PlanetLab events, research papers, and working groups.

2. Full ($75K annual dues): provides a rotating seat on the Steering Committee, a maximum of 10 slices, access to PlanetLab events, research papers, and working groups.

3. Associate ($25K annual dues): provides up to 2 slices and access to PlanetLab events, research papers, and working groups.

4. Sponsor ($10k annual dues): provides access to PlanetLab events and research papers.

5. Academic (no annual dues): provides a seat on the Steering Committee by invitation, up to 10 slices, and access to PlanetLab events, research papers, and working groups.

3.2 Amazon's Elastic Compute Cloud (EC2)

Amazon is a leading-edge company in the cloud computing space. In August, 2006, Amazon released Amazon Elastic Cloud Computing (Amazon EC2) to the public. Amazon's EC2 is a virtual site farm that allows a company to outsource its Information Technology infrastructure so that their resources can be concentrated on other initiatives vital to the company's success. Users can dynamically determine

the number and type of compute *instances* needed to support their infrastructure. There are two types of instances: standard and high-CPU. Within each category there are different sizes of instances depending on the amount of main memory, number of compute units, and amount of secondary storage available. For example, a small standard instance has 1.7 GB of main memory, 1 EC2 compute unit, 160 GB of secondary storage, all in a 32-bit platform.

Since its rollout, Amazon has improved the reliability of their cloud infrastructure service with persistent storage capability via Elastic Block Storage (EBS) and the assignment of static IP addresses in the dynamic cloud environment via Elastic IP addresses. Other interesting features include Amazon CloudWatch, a performance monitoring service, and Auto Scaling, which allows the number of instances to be automatically scaled up or down to maintain performance goals.

3.2.1 Structure

With service-on-demand capability built into Amazon EC2, customers have great flexibility to customize their IT needs dynamically as their customer base increases. Amazon's EC2 centers around the idea of providing "Anything as a Service" which is potentially a game-changing technology that could reshape IT [4]. EC2 implements three important basic services: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [3]:

1. Infrastructure as a Service (IaaS): IaaS clouds aim at the hardware level and make the provisioning of customer resources such as servers, connections, storage, and related tools easy and affordable. This allows developers to quickly and cost-effectively build application environments from scratch. Building an IT infrastructure to provide IT services can be quite complex; it becomes even more complex when the infrastructure is large-scale, highly-distributed, and multi-regional-site. Yet, with the complexity of such infrastructure comes a high degree of flexibility. Companies are able to scale up or down their resources depending on the current workloads within minutes.

2. Platform as a Service (PaaS): PaaS is the integration between infrastructure and a commercial development platform to build and launch applications or services from. PaaS clouds, working in combination with IaaS clouds, have the benefit of making deployment and scalability trivial and ensuring that costs are linearly incremental and reasonably predictable. Companies such as Oracle, Microsoft, Google, and SalesForce have enabled their software to be used in the cloud as an increasing number of users perceive the effectiveness and cost saving of cloud computing. Examples of PaaS include Google App Engine (Python), Microsoft's Azure (.Net), and Force.com (Appexchange). Many more are in the process of converting their platform software services to be cloud enabled.

3. Software as a Service (SaaS): SaaS aims at the application-level software used by cloud users to achieve their mission. It ranges from office software to financial software such as tax preparation and budgeting. The term SaaS has been around for quite a while but cloud computing has revitalized the SaaS model by reducing the cost of producing a SaaS application. SaaS serves as a stepping stone in the evolution of cloud computing in terms of resource management and allocation.

3.2.2 Business Model

Amazon adopts a pay-as-you-grow paradigm to lease services provided by EC2. Depending on the instances being requested, Amazon EC2 provides flexible plans of service to meet all types of customers from small startup of a single individual developer to a large-scale corporation that has multiple distributed locations around the world. Usage of resources is calculated and billed on a per hour basis. As of now, Amazon EC2 is strictly for commercial users and not available for academic institutions at a discounted price. However, many academic institutions are considering Amazon EC2 as a promising solution for their future IT needs.

3.3 Google's App Engine

Google's App Engine is a cloud infrastructure that has gained significant attention and market share in the cloud computing space. App Engine helps to eliminate or reduce the IT infrastructure at the local or office level while providing high reliability, availability, scalability, and integrity. Yet, Google provides a wide range of development tools that can be leased as a service and are very elementary and trivial to use for application deployment and maintenance. These tools scale as the needs to serve a higher number of concurrent customers increases.

3.3.1 Structure

Google developed its own distributed storage system called Bigtable [2]. Bigtable's data model is a sparse, distributed, multidimensional map indexed by a row key, a column key, and a timestamp. To achieve Google's goals of providing wide applicability, scalability, high performance and high availability infrastructure, Google developed an SQL-like query language for Bigtable called "GQL". Unlike SQL, GQL does not support join statements and has restrictions on the `where` clause to $>, \leq, <, \leq$ operations on one column only [16]. Google also provides the non-relational Bigtable API, similar but not identical to the SQL API, which supports datastore transaction and query.

App Engine currently supports two runtime environments: Java and Python and provides standard protocols and a number of tools for web development [15]. Applications are deployed and run in the secure environment that permits restricted access to the host operating system. This restricted access allows App Engine to distribute requests to multiple servers and makes it possible to start and stop servers to meet the traffic demand [15].

3.3.2 Business Model

Google applies the "Pay for What You Use" business model. There is no fee to start an App Engine account. The usage of resources such as storage and bandwidth is measured in gigabytes. Google App Engine gives developers total control of the resources that they are using and allows them to set the maximum allowed usage to avoid surprises when they receive the monthly bills. Developers can develop and deploy at no charge applications that utilize at most 500MB of disk space and enough CPU and bandwidth to serve up to 5 million page views a month [15]. Google also allows developers to enable a billing capability that monitors if the usage of the environment exceeds the free limit. Users are only charged for the use of resources that exceeds the free levels [15].

3.4 Microsoft's Azure

Windows Azure is an open cloud computing platform built into the operating system that "serves as the development, service hosting, and service management environment for Azure Services Platform" [17]. Microsoft provides a data center to host web applications that can scale in terms of computing power and data storage on demand. Microsoft plans to release the initial commercial version of Windows Azure at the end of 2009. Since Windows Azure is an open platform, it supports both Microsoft proprietary and other commercial languages and development environments. The Azure Service Platform provides a set of Microsoft core developer services such as Live, .Net, SQL, SharePoint, and Dynamic CRM Services that will be run on Windows Azure operating system [18].

Windows Azure targets all kinds of users ranging from novice computer users and hobbyists to web, software, and corporate developers, or anyone in between, providing scalability and flexibility to an increasing demand of outsourcing the local and corporate development environment [18]. The experience that Microsoft has gained over the years in building operating systems and development tools has converged into the release of Windows Azure. With Microsoft's proprietary .NET environment and Visual Studio tools, developers can create any type of application: gaming, wireless device applications, web authoring, or combinations of everything. Developers can also use non-Microsoft languages such as Ruby, PHP and Python, and environments such as Eclipse and NetBeans to build their applications. Windows Azure not only provides a set of available services for consumers, but also facilitates the composition of more complex services tailored to their needs [18].

3.4.1 Structure

Microsoft Azure relies on several core services: Live, SQL, .NET, SharePoint, and Dynamic CRM services. These core services are used to facilitate, develop, manage, compose, and monitor services and the communication between services.

Live services handle user data and application resources and allow developers to communicate with web audiences via digital devices and access social networking services such as chat and e-mail.

SQL services extend Microsoft SQL Server to cloud-enabled capabilities as a web-based and distributed relational database with enhanced security, availability, scalability, and reliability. Microsoft SQL Data Services (SDS) is a cloud-based relational database platform built on SQL Server technologies. With SDS, customers can maximize the benefits and capabilities to provision and deploy the relational database to highly distributed data centers [21].

.NET services enable hosting, scalability, and developer-oriented capabilities that serve as the foundation for many cloud-based and cloud-aware applications [22]. Microsoft is working on adding more services as core services for .NET Services. It currently supports three core services: Access Control, Service Bus, and Workflow [22].

SharePoint and Dynamic CRM Services allow customers to access SharePoint and CRM functionalities to collaborate and share information among different entities. Developers can utilize the capabilities provided in SharePoint and CRM cloud-enabled services to build applications quickly by extending from these services [18].

3.4.2 Business Model

Windows Azure Service Platform targets all types of customers for its business model, which is built from four principles: pay to use, the price to use the service must be attractive and affordable to the market, the opportunity to grow and expand to become Microsoft service partners, and the ease of tracking and accessing via web interface or the existing channels [19].

Services in the Azure Services Platform will be initially available in the Community Technology Preview (CTP) phase with certain constraints for everyone at no cost. Microsoft will use the feedback from the CTP to evaluate and build an affordable business model to meet everyone's needs in terms of the usage, pricing, and services [19]. After the CTP, Microsoft will launch Azure services commercially, which then will be charged and licensed based on a consumption-based model. Microsoft provides tools in the

Azure Services Platform to monitor, increase and decrease resource usage based on user's business requirements. Furthermore, these tools allows customers to expand or limit to certain capacities the services so that costs do not exceed the allowed budget.

3.5 NSF's Cloud Computing Research Initiative

Recently, the National Science Foundation (NSF) has invested in the area of cloud computing research projects or projects that take advantage of a cloud computing infrastructure to help solve problems that would be impossible to handle with the computing power that resides locally.

NSF just awarded $5 million in grants to 14 universities around the United States through its Cluster Exploratory (CluE) program on April 23rd, 2009 [23]. These universities will participate in the IBM/Google Cloud Computing University Initiative, which started in 2007 to help students from various disciplines utilize cloud benefits and gain invaluable skills to build applications. The awardees are engaged in various research areas in high data-intensive computing including image processing, studies for comparative large-scale data analysis, Internet improvement studies, and medical studies of human genome sequencing [23]. These research projects will utilize the existing cloud infrastructure provided by IBM and Google. The main foci are to explore novel concepts for solving challenging problems using the parallel and distributed cloud environment.

4 Experiments with PlanetLab

To demonstrate the power of a cloud platform we conducted some experiments that are reported in this section. The experiments use a simple application, namely the parallel computation of the number π. The computation of π can be approximated by observing that π is the area of a circle of radius equal to one. Therefore, π is equal to four times the area of a quadrant with radius one. Figure 2 indicates a number of randomly generated points in a square of unit area in the x-y coordinate plane. Also shown in the figure is a quadrant of a circle with unit radius. The Monte Carlo approach for computing the area of the circle quadrant is simply based on determining the fraction of all randomly generated points that fall within the quadrant of a circle.

More specifically, the following Monte Carlo based algorithm can be used to estimate π:

1. NumPointsInQuadrant $\leftarrow$ 0;
2. Repeat (a) and (b) m times.
   (a) Randomly select a point (x, y) such that x and y are random numbers uniformly distributed in [0,1].
   (b) If $\sqrt{x^2 + y^2} \leq 1$ then NumPointsInQuadrant $\leftarrow$ NumPointsInQuadrant + 1.
3. $\pi \leftarrow 4 \times \text{NumPointsInQuadrant}/m$.

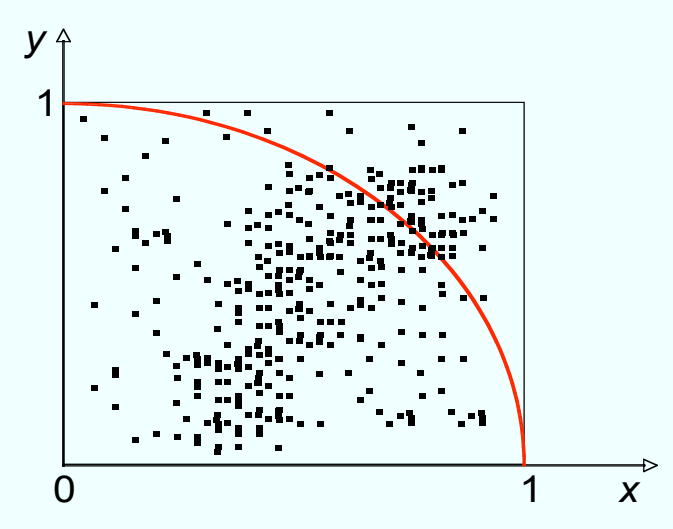

Figure 2: Monte Carlo computation of π.

The larger the number of times m that step 2 is repeated, the better is the approximation for π computed in step 3. The precision of the computation can be increased without significantly increasing the computation time by executing steps 2 at n nodes in parallel and then combining the values of NumPointsInQuadrant reported by each node. Thus, the distributed version of the above algorithm is as follows.

1. (At the master node). Send m and n to all n slave nodes.
2. (At each slave node i, $i = 1, \cdots, n$). $\text{NumPointsInQuadrant}_i \leftarrow 0$.
3. (At each slave node i, $i = 1, \cdots, n$). Repeat (a) and (b) m/n times.
   (a) Randomly select a point (x, y) such that x and y are random numbers uniformly distributed in [0,1].
   (b) If $\sqrt{x^2 + y^2} \leq 1$ then $\text{NumPointsInQuadrant}_i \leftarrow \text{NumPointsInQuadrant}_i + 1$.
4. (At each slave node $i, i = 1, \cdots, n$). Send $\text{NumberPointsInQuadrant}_i$ to the master node.

5. (At the master node).
   $\pi \leftarrow 4 \times \sum_{i=1}^{n} \text{NumPointsInQuadrant}_i/m.$

We implemented the distributed algorithm in PlanetLab with a number of nodes n varying from 1 to 10. The value of m was set to 10^9 (i.e., 1 billion) and each node executes steps 3 of the distributed algorithm m/n times. In our experiments, all nodes have the following characteristics: 2 Intel(R) Core(TM)2 Duo E6550 processors at 2.33 GHz with 3.44 GB of main memory. The ten nodes that participated in the experiments were spread around the country according to Table 1. The last column in the table shows the participation of a node in the experiments. For example, node UP1 participated in all experiments in which n varied from 1 to 10. Node GMU4 participated in the experiments in which n varied from 6 to 10. Node VT was only included in the experiment with 10 nodes. So, when we ran experiments with 5 nodes, the nodes that participated were UP1, UP2, GT1, GT2, and GMU3.

Node Name	Location	Participation
UP1	Univ. Pennsylvania	1-10
UP2	Univ. Pennsylvania	2-10
GT1	Georgetown University	3-10
GT2	Georgetown University	4-10
GMU3	George Mason University	5-10
GMU4	George Mason University	6-10
CT1	Caltech	7-10
CT2	Caltech	8-10
CN4	Cornell University	9-10
VT	Virginia Tech	10

Table 1: Distribution of the 10 nodes for the experiment.

For each value of the number of nodes n, 100 runs were made and the execution time E of the algorithm was measured as the time needed for all n nodes to complete their execution. Figure 3 shows the variation of the average execution time for the 100 runs as a function of the number of nodes used. The figure also shows 95% confidence intervals for the average. A regression analysis shows that a power curve approximates reasonably well (i.e., $R^2 = 0.9055$) the variation of the execution time E with the number of nodes n. The regression curve is

$$E = 124,828 \times n^{-0.7349}. \quad (1)$$

The execution time decreases as n increases according to a power relationship. For example, with a single node, the computation of π takes 2 minutes on average while with ten nodes it takes 20 seconds on average.

Figure 4 shows the variation of the speedup, defined as the ratio between the average execution time with one node E_1 and the execution time E_n with n nodes. For example, when 10 nodes are used, the application runs 6.3

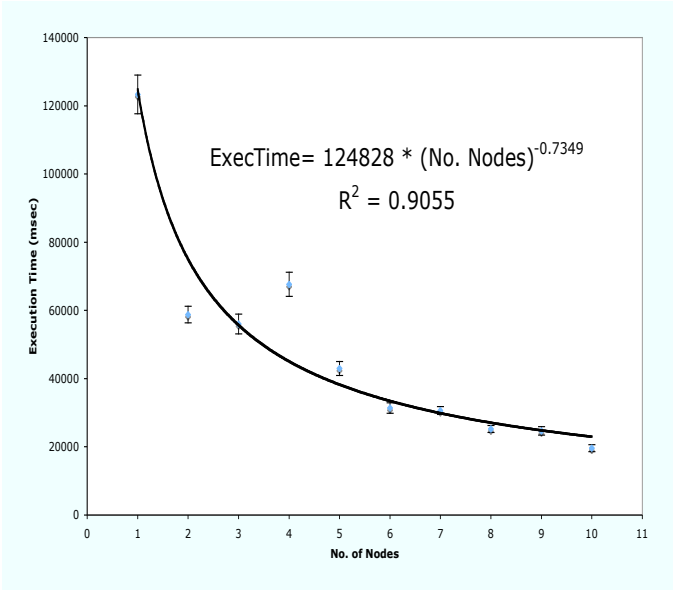

Figure 3: Execution time (in msec) vs. number of nodes.

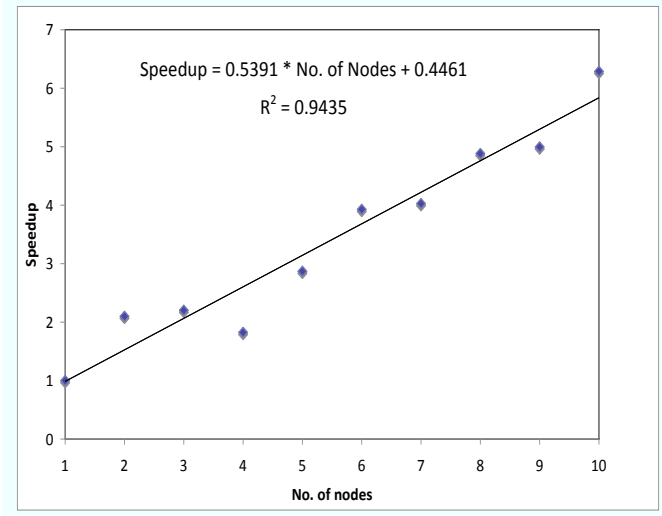

Figure 4: Speedup vs. number of nodes.

times faster than if a single node were used. The speedup increases with n. A linear regression shows that the relationship between the speedup S and the number of nodes n can be reasonably approximated as

$$S = 0.5391 \times n + 0.4461 \quad (2)$$

with a coefficient of determination R^2 equal to 0.9435.

While the example above is relatively simple, it illustrates how computations can be performed on top of a cloud platform. However, many organizations are moving significantly complex applications to the cloud. This requires that capacity planning issues—usually handled within the confines of an organization—be tackled from an external perspective.

5 Capacity Planning for the Cloud

How does the traditional notion of capacity planning change when using cloud computing? There are two points of view to consider: one from the user of cloud services and the other from the provider.

5.1 From the Cloud User's Point of View

When services are executed on demand using cloud resources, the burden of capacity planning shifts to the provider of the cloud services. However, cloud users have to be able to negotiate SLAs with cloud service providers. Since there may be SLAs for different QoS metrics, cloud users should consider using the notion of *utility function* to determine the combined usefulness of cloud services as a function of the various SLAs. Utility functions are used quite often in economics and have been extensively used by the first author of this paper in autonomic computing [1, 8, 9, 10, 13]. However, the use of utility functions to determine the optimal mix of SLAs in cloud computing as presented here is novel.

We introduce the following notation to formalize the problem of optimal selection of SLAs to be negotiated with the provider of cloud services. Let,

- $\mathrm{SLA_r}$: SLA (in seconds) on the average response time per transaction.
- $\mathrm{SLA_x}$: SLA (in tps) on the transaction throughput.
- $\mathrm{SLA_a}$: SLA on the cloud availability.
- $C_r(\mathrm{SLA_r})$: per transaction cost (in cents) when the negotiated response time SLA is $\mathrm{SLA_r}$.
- $C_x(\mathrm{SLA_x})$: per transaction cost (in cents) when the negotiated throughput SLA is $\mathrm{SLA_x}$.
- $C_a(\mathrm{SLA_a})$: per transaction cost (in cents) when the negotiated availability SLA is $\mathrm{SLA_a}$.
- U: global utility. The global utility function is composed of terms that represent the utility for various metrics such as response time, throughput, and availability. The utility is a dimensionless number in the [0,1] range.
- w_r, w_x, w_a: weights associated to response time, throughput, and availability, respectively, used to compute the global utility. $w_r + w_x + w_a = 1$.

The cost functions used in the following example are:

$$\begin{aligned} C_r(\mathrm{SLA_r}) &= \alpha_r e^{-\beta_r\ \mathrm{SLA_r}} \\ C_x(\mathrm{SLA_x}) &= \alpha_x\ \mathrm{SLA_x} \\ C_a(\mathrm{SLA_a}) &= e^{\beta_a\ \mathrm{SLA_a}} - e^{0.9\beta_a},\ \mathrm{SLA_a} \geq 0.9. \end{aligned} \quad (3)$$

As it can be seen, the response time cost decreases exponentially as the SLA for response time increases. The throughput cost increases linearly with the throughput SLA and the availability cost increases exponentially as the availability SLA goes from 0.9 to 1.0.

We used the following utility function:

$$\begin{aligned} U &= w_r\ \frac{2.0\ e^{-\mathrm{SLA_r}}}{1 + e^{-\mathrm{SLA_r}}} + \\ & \quad w_x(1 - e^{-0.1\ \mathrm{SLA_x}}) + w_a(10\ \mathrm{SLA_a} - 9). \end{aligned} \quad (4)$$

The first term in Eq. (4) is the response time utility function, the second term is the throughput utility function, and the third is the availability utility function (see Fig. 5). Because each of these individual utility functions have their values in the $[0, 1]$ range and because $w_r + w_x + w_a = 1$, $U \in [0, 1]$.

A cloud user is now faced with the problem of selecting the optimal values of the SLAs that maximize the utility function subject to SLA and cost constraints. This can be cast as the (generally) non-linear constraint optimization problem shown below.

$$\begin{aligned} maximize\ U &= f(\mathrm{SLA_r}, \mathrm{SLA_x}, \mathrm{SLA_a}) \\ subject\ to & \\ \gamma_r^{\min} &\leq \mathrm{SLA_r} \leq \gamma_r^{\max} \\ \gamma_x^{\min} &\leq \mathrm{SLA_x} \leq \gamma_x^{\max} \\ \gamma_a^{\min} &\leq \mathrm{SLA_a} \leq \gamma_a^{\max} \\ C_r(\mathrm{SLA_r}) &+ C_x(\mathrm{SLA_x}) + C_a(\mathrm{SLA_a}) \leq C_{max} \end{aligned} \quad (5)$$

The optimization problem above indicates that values of the SLAs for response time, throughput, and availability have to obtained so that the utility function is maximized. The SLAs have constraints (minimum and maximum values), which may be inherent to what the cloud provider can offer. There is also a maximum cost constraint C_{max}.

We provide here a numeric solution for this problem with the following set of parameters: $\alpha_r = 0.4, \beta_r = 0.1, \alpha_x =$

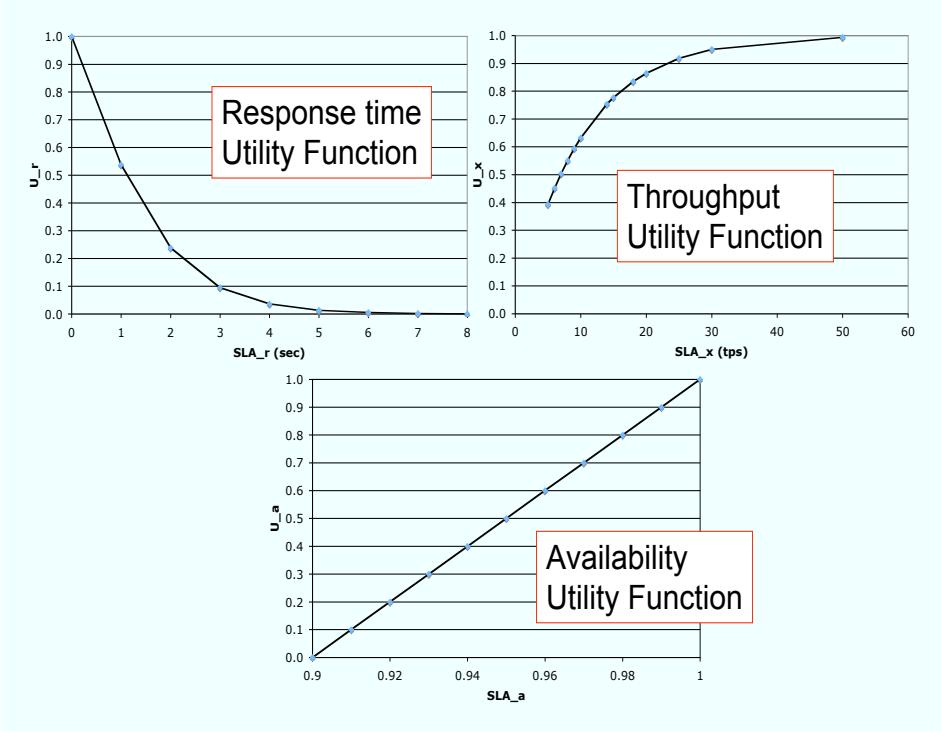

Figure 5: Utility functions for response time, throughput, and availability.

$0.03, \beta_a = 0.8, \gamma_r^{\min} = 1$ sec, $\gamma_r^{\max} = 4$ sec, $\gamma_x^{\min} = 5$ tps, $\gamma_x^{\max} = \infty, \gamma_a^{\min} = 0.92$, and $\gamma_a^{\max} = 0.999$.

Table 2 shows the results of solving the non-linear constrained optimization problem for various values of the maximum cost C_{max}. The values of the weights w_r, w_x, w_a used are 0.4, 0.3, and 0.3, respectively. The maximum per-transaction cost decreases from row 4 to row 1. When the user is willing to spend 0.70 cents per transaction, the response time SLA is the best possible (i.e., 1 second), the throughput SLA is slightly above the worst possible value of 5 tps, and the availability is the best possible (i.e., 0.999). As the user is less willing to spend more money per transaction, the SLAs to be negotiated with the cloud change. For example, for $C_{max} = 0.60$ cents, the user will need to settle for a response time SLA of 3.543 sec. The throughput SLA goes down to its worst value (i.e., 5.0 tps). As the C_{max} is reduced, worse SLAs need to be negotiated as shown by the table. As seen in the table, the cloud utility value decreases as C_{max} decreases.

Many optimization solvers can be used to solve the type of non-linear optimization problem discussed above. The NEOS Server for Optimization (see http://neos.mcs.anl.gov/) provides many such options. MS Excel's Solver (under the Tools menu) can also be used to solve medium scale optimization problems. This is the solver that was used in the numerical example discussed in this section.

C_{max}	Utility	SLA_r	SLA_x	SLA_a
0.70	0.641	1.000	5.625	0.999
0.60	0.438	3.543	5.000	0.999
0.55	0.366	4.000	5.000	0.978
0.50	0.279	4.000	5.000	0.949

Table 2: Numeric results for optimal SLA selection. Cost is in cents, SLA_r in sec, and SLA_x in tps.

5.2 From the Point of View of the Cloud Provider

From the point of view of the cloud infrastructure provider, the workload is very unpredictable, exhibits a high time variability, and tends to be very heterogeneous. The cloud provider's infrastructure tends to be very large and very complex with a myriad of parameter settings that can influence performance in a significant way. Therefore, it is generally very difficult for system administrators to manually change these parameters at run time to keep pace with the variability of the workload in a way that meets customer's SLAs.

The solution is for cloud providers to use autonomic computing techniques [7]. Proposed autonomic techniques for data centers are based on three types of methods: control theory [5], machine learning [30], and combinatorial

search methods combined with queuing network models [1, 8, 9, 10, 13]. The latter set of methods have been successfully applied by Menascé and his students and colleagues in a variety of settings including e-commerce, virtualized environments, and Internet data centers.

A platform that provides cloud computing services must be able to dynamically provision its various resources (e.g., computing, storage, software licenses, and networking) to its various users according to their instantaneous needs and in compliance with negotiated SLAs. Autonomic computing techniques allow for resources to be dynamically shifted without human intervention in a way that optimizes a certain objective function. Once again, utility functions are very useful as an objective function to be maximized by an autonomic controller such as the one in [1]. One can establish a global utility function for the cloud computing service provider. This utility function takes into account the negotiated SLAs for each customer as well as their relative importance.

The objective then is to design an autonomic controller that determines the allocation of resources that maximizes the global utility function of the cloud computing service provider. The controller should run its algorithm periodically to dynamically track customer's constant changes in resource demands.

6 Concluding Remarks

This paper has discussed the concept of cloud computing as well as its advantages and disadvantages. Some examples of cloud computing infrastructures were presented. The list is by no means exhaustive nor is there an implication that the examples discussed are representative of best practices in the industry. We tried though to provide a diverse range of examples by including cloud platforms widely used in academia and research as well as those provided by companies.

In order to provide a more concrete example of the benefits of cloud computing, we developed a parallel application on top of PlanetLab and provided results on execution time and speedup as a function of the number of nodes involved.

The paper then discussed the important issue of how cloud users optimally select the values of SLAs to be negotiated with cloud providers in order to maximize their utility subject to cost constraints. A numeric example was thoroughly discussed.

Acknowledgments

The work of Daniel Menascé is partially supported by award no. CCF-0820060 from the National Science Foundation.

References

[1] M.N. Bennani and D.A. Menascé, "Resource Allocation for Autonomic Data Centers Using Analytic Performance Models," *Proc. 2005 IEEE International Conference on Autonomic Computing*, Seattle, WA, June 13-16, 2005.

[2] F. Chang, J. Dean, S. Ghemawat, W. Hsieh, D. Wallach, M. Burrows, T. Chandra, A. Fikes, R. Gruber. Bigtable: A Distributed Storage System for Structured Data.

[3] Cloud Computing Basics. www.webguild.org/2008/07/cloud-computing-basics.php.

[4] Cloud Computing: Anything as a Service. www.cioinsight.com/c/a/Strategic-Tech/Cloud-Computing-Anything-as-a-Service/.

[5] Y. Diao, N. Gandhi, J. L. Hellerstein, S. Parekh, and D. M. Tilbury, "Using MIMO Feedback Control to Enforce Policies for Interrelated Metrics With Application to the Apache Web Server," *Proc. IEEE/IFIP Network Operations and Management Symp.*, Florence, Italy, April 15-19, 2002.

[6] F. Douglis, "Staring at Clouds," *IEEE Internet Computing*, Vol. 13, No. 3, 2009.

[7] D.A. Menascé and J. Kephart, "Guest Editor's Introduction," Special Issue on Autonomic Computing, *IEEE Internet Computing*, January 2007.

[8] D.A. Menascé and M.N. Bennani, "Autonomic Virtualized Environments," *Proc. IEEE International Conference on Autonomic and Autonomous Systems*, July 19-21, 2006, Silicon Valley, CA, USA.

[9] D.A. Menascé and M.N. Bennani, "Dynamic Server Allocation for Autonomic Service Centers in the Presence of Failures," in *Autonomic Computing: Concepts, Infrastructure, and Applications*, eds. S. Hariri and M. Parashar, CRC Press, 2006.

[10] D.A. Menascé, M.N. Bennani, and H. Ruan, "On the Use of Online Analytic Performance Models in Self-Managing and Self-Organizing Computer Systems," in the book *Self-Star Properties in Complex Information Systems*, O. Babaoglu, et. al., eds., Lecture Notes in Computer Science, Vol. 3460, Springer Verlag, 2005.

[11] D.A. Menascé and E. Casalicchio, "Quality of Service Aspects and Metrics in Grid Computing," *Proc. 2004 Computer Measurement Group Conference*, Las Vegas, NV, December 5-10, 2004.

[12] D.A. Menascé and E. Casalicchio, "A Framework for Resource Allocation in Grid Computing," *Proc. 12th Annual Meeting of the IEEE/ACM International*

Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), Volendam, The Netherlands, October 5-7, 2004.

[13] D.A. Menascé and M.N. Bennani, "On the Use of Performance Models to Design Self-Managing Computer Systems," *Proc. 2003 Computer Measurement Group Conference*, Dallas, TX, Dec. 7-12, 2003.

[14] I. Foster and C. Kesselman, *The grid: Blueprint for a new Computing Infrastructure*," Morgan-Kaufman,1999.

[15] Google, What Is Google App Engine. http://code.google.com/appengine/docs/whatisgoogleappengine.html.

[16] Google App Engine. http://en.wikipedia.org/wiki/Google_App_Engine.

[17] Microsoft, Windows Azure. www.microsoft.com/azure/windowsazure.mspx.

[18] Microsoft, What is the Azure Service Platform. www.microsoft.com/azure/whatisazure.mspx.

[19] Microsoft, Pricing & Licensing Overview. www.microsoft.com/azure/pricing.mspx.

[20] Microsoft, SQL Services. www.microsoft.com/azure/sql.mspx.

[21] Microsoft, SQL Data Services. www.microsoft.com/azure/data.mspx.

[22] Microsoft, .NET Services. www.microsoft.com/azure/netservices.mspx.

[23] National Science Foundation, Cloud Computing Research. www.nsf.gov/news/news_summ.jsp?cntn_id=114686.

[24] L. Peterson, A. Bavier, M. Fiuczynski, and S. Muir, "Experiences Implementing PlanetLab," OSDI 2006, Seattle, WA, November 2006.

[25] PlanetLab Sites. https://www.planet-lab.org/db/pub/sites.php. Last accessed on May 29, 2009.

[26] PlanetLab Nodes. https://www.planet-lab.org/db/nodes/index.php. Last accessed on May 29, 2009.

[27] PlanetLab Slides. http://summer.cs.princeton.edu/status/viz_slices.html. Last accessed on May 29, 2009.

[28] PlanetLab Consortium. http://www.planet-lab.org/consortium.

[29] PlanetLab Central API. http://www.planet-lab.org/doc/plc_api.

[30] G. Tesauro, N.K. Jong, R. Das, and M.N. Bennani, "A Hybrid Reinforcement Learning Approach to Autonomic Resource Allocation," *Proc. IEEE Intl. Conf. Autonomic Computing (ICAC'06)*, Dublin, Ireland, June 13-16, 2006.

[31] D. Thain, T. Tannenbaum, and M. Livny, "Distributed Computing in Practice: The Condor Experience," *Concurrency and Computation: Practice and Experience*, Vol. 17, No. 2-4, pages 323-356, February-April, 2005.