Spring 2015

A Mechanism Design Approach to Resource Procurement in Cloud Computing

Mahesh Kumar Pallati
Governors State University

Follow this and additional works at: http://opus.govst.edu/capstones

Part of the Systems Architecture Commons

Recommended Citation

For more information about the academic degree, extended learning, and certificate programs of Governors State University, go to http://www.govst.edu/Academics/Degree_Programs_and_Certifications/

Visit the Governors State Computer Science Department
This Project Summary is brought to you for free and open access by OPUS Open Portal to University Scholarship. It has been accepted for inclusion in Capstone Projects by an authorized administrator of OPUS Open Portal to University Scholarship. For more information, please contact opus@govst.edu.
A Mechanism Design Approach to Resource Procurement in Cloud Computing

By

Mahesh Kumar Pallati

Masters Project / Graduate Project
Submitted in partial fulfillment of the requirements

For the Degree of Master of Science,
With a Major in Computer Science

Governors State University
University Park, IL 60484.

Spring 2015
INDEX

1. Abstract
2. Introduction
3. System Analysis
4. Software Environment
5. System Specifications
6. Modules
7. System Design
8. System Testing
9. Conclusion
10. Bibliography
We present a cloud resource procurement approach which not only automates the selection of an appropriate cloud vendor but also implements dynamic pricing. Three possible mechanisms are suggested for cloud resource procurement: cloud-dominant strategy incentive compatible (C-DSIC), cloud-Bayesian incentive compatible (C-BIC), and cloud optimal (C-OPT). C-DSIC is dominant strategy incentive compatible, based on the VCG mechanism, and is a low-bid Vickrey auction. C-BIC is Bayesian incentive compatible, which achieves budget balance. C-BIC does not satisfy individual rationality. In C-DSIC and C-BIC, the cloud vendor who charges the lowest cost per unit Qos is declared the winner. In C-OPT, the cloud vendor with the least virtual cost is declared the Winner. C-OPT overcome the limitations of both C-DSIC and C-BIC. C-OPT is not only Bayesian incentive compatible, but also individually rational. Our experiments indicate that the resource procurement cost decreases with increase in number of cloud vendors irrespective of the mechanisms. We also propose a procurement module for a cloud broker which can implement C-DSIC, C-BIC, or C-OPT to perform resource procurement in a cloud computing context. A cloud broker with such a procurement module enables users to automate the choice of a cloud vendor among many with diverse offerings, and is also an essential first step toward implementing dynamic pricing in the cloud.
INTRODUCTION

CLOUD computing is an increasingly popular paradigm of offering services over the Internet. It is also an active area of research, and the popularity of this paradigm is growing rapidly. Many companies like Amazon, IBM, Google, salesforce.com, Unisys, and so on, now offer cloud services. The main advantage of cloud computing is the ability to provision IT resources on demand (thus avoiding the problems of over-provisioning and under-provisioning which are commonly seen with organizations that have widely variable requirements due to growth/shrinkage, seasonal peaks, and valleys, etc.). The resources offered may include storage, CPU processing power, IT services, and so on. These resources are often geographically distant from users.

We can say the following:

. A cloud user is a person or an organization (such as an SME—small and medium enterprise) that uses cloud services.

. A cloud vendor is an organization that offers cloud services for use on payment.

. A cloud broker [2] is a middleware that interacts with service providers on behalf of the user. It is responsible for configuring the user’s settings suitably and for procuring resources. Resource procurement of cloud resources is an interesting and yet unexplored area in cloud computing. Cloud vendors follow a fixed pricing strategy (“pay as you go”) for pricing their resources and do not provide any incentive to their users to adjust consumption patterns according to availability or other factors.

Consider, for example, a user who wants to use a service in the form of an application hosted on a cloud. There are cloud vendors who provide versions of that application at different prices and with varying quality-of-service (QoS) parameters. The user has to go through the specifications of each cloud vendor to select the appropriate one, to obtain the service within budget and of the desired quality. In case of an organization acting as a user, this selection is quite complex and challenging. Also, the companies offering cloud services, and their offerings, change continually. So, given the large and varying multitude of cloud vendors, it is very tedious to select the most appropriate one manually. Hence, there is a need for a scalable and automated method to perform resource procurement in the cloud. Observe that while cloud vendors do not yet offer standardized services, they will need to do so, and that the “federated cloud has huge
potential.” In that event, it would become possible to mix and interchange resources offered by different cloud vendors and to automate the procurement of such resources. If resource procurement is automated, then the challenge would be to find the appropriate location where the solution can be deployed.

**SYSTEM ANALYSIS**

**Existing System:**

Resource procurement of cloud resources is an interesting and yet unexplored area in cloud computing. Cloud vendors follow a fixed pricing strategy (“pay as you go”) for pricing their resources and do not provide any incentive to their users to adjust consumption patterns according to availability or other factors.

Most cloud vendors use the pay-as-you-go model. Many are loath to negotiate contracts as they lack understanding of a sound theoretical basis for dynamic pricing. The default agreement offered by a vendor often contractually benefits the vendor but not the user, resulting in a mismatch with user requirements. Hence, this kind of pricing favors the cloud vendor. Also, there is no clear commitment on SLAs.

**Proposed System:**

Each cloud user has resource requirements. The users perform reverse auctions for procuring resources (which are also called procurement auctions). Cloud vendors offer resources, but with varying costs and quality metrics. The goal of the cloud user is to minimize the total cost of procuring resources without compromising quality of service. To minimize the procurement cost, it is necessary
for the cloud user to know the real costs of cloud vendors. A user announces its specifications for desired resources and quality of service to all cloud vendors, with the broker acting as a middleman. The cloud vendors decide whether to participate in the auction based on the user information and submit their bids to the broker.

**Advantage:**

- Costs and tasks are uniformly distributed. The average procurement cost is calculated in every mechanism and compared.

**SOFTWARE ENVIRONMENT**

**FEATURES OF .NET**

Microsoft .NET is a set of Microsoft software technologies for rapidly building and integrating XML Web services, Microsoft Windows-based applications, and Web solutions. The .NET Framework is a language-neutral platform for writing programs that can easily and securely interoperate. There’s no language barrier with .NET: there are numerous languages available to the developer including Managed C++, C#, Visual Basic and Java Script. The .NET framework provides the foundation for components to interact seamlessly, whether locally or remotely on different platforms. It standardizes common data types and communications protocols so that components created in different languages can easily interoperate.

“.NET” is also the collective name given to various software components built upon the .NET platform. These will be both products (Visual Studio.NET and Windows.NET Server, for instance) and services (like Passport, .NET My Services, and so on).

**THE .NET FRAMEWORK**

The .NET Framework has two main parts:

1. The Common Language Runtime (CLR).
2. A hierarchical set of class libraries.
The CLR is described as the “execution engine” of .NET. It provides the environment within which programs run. The most important features are:

- Conversion from a low-level assembler-style language, called Intermediate Language (IL), into code native to the platform being executed on.
- Memory management, notably including garbage collection.
- Checking and enforcing security restrictions on the running code.
- Loading and executing programs, with version control and other such features.
- The following features of the .NET framework are also worth description:

**MANAGED CODE**

The code that targets .NET, and which contains certain extra information - “metadata” - to describe itself. Whilst both managed and unmanaged code can run in the runtime, only managed code contains the information that allows the CLR to guarantee, for instance, safe execution and interoperability.

**MANAGED DATA**

With Managed Code comes Managed Data. CLR provides memory allocation and deallocation facilities, and garbage collection. Some .NET languages use Managed Data by default, such as C#, Visual Basic.NET and JScript.NET, whereas others, namely C++, do not. Targeting CLR can, depending on the language you’re using, impose certain constraints on the features available. As with managed and unmanaged code, one can have both managed and unmanaged data in .NET applications - data that doesn’t get garbage collected but instead is looked after by unmanaged code.

**COMMON TYPE SYSTEM**

The CLR uses something called the Common Type System (CTS) to strictly enforce type-safety. This ensures that all classes are compatible with each other, by describing types in a common way. CTS define how types work within the runtime, which enables types in one language to interoperate with types in another language, including cross-language exception
handling. As well as ensuring that types are only used in appropriate ways, the runtime also ensures that code doesn’t attempt to access memory that hasn’t been allocated to it.

COMMON LANGUAGE SPECIFICATION

The CLR provides built-in support for language interoperability. To ensure that you can develop managed code that can be fully used by developers using any programming language, a set of language features and rules for using them called the Common Language Specification (CLS) has been defined. Components that follow these rules and expose only CLS features are considered CLS-compliant.

THE CLASS LIBRARY

.NET provides a single-rooted hierarchy of classes, containing over 7000 types. The root of the namespace is called System; this contains basic types like Byte, Double, Boolean, and String, as well as Object. All objects derive from System. Object. As well as objects, there are value types. Value types can be allocated on the stack, which can provide useful flexibility. There are also efficient means of converting value types to object types if and when necessary.

The set of classes is pretty comprehensive, providing collections, file, screen, and network I/O, threading, and so on, as well as XML and database connectivity.

The class library is subdivided into a number of sets (or namespaces), each providing distinct areas of functionality, with dependencies between the namespaces kept to a minimum.

LANGUAGES SUPPORTED BY .NET

The multi-language capability of the .NET Framework and Visual Studio .NET enables developers to use their existing programming skills to build all types of applications and XML Web services. The .NET framework supports new versions of Microsoft’s old favorites Visual Basic and C++ (as VB.NET and Managed C++), but there are also a number of new additions to the family.

Visual Basic .NET has been updated to include many new and improved language features that make it a powerful object-oriented programming language. These features include
inheritance, interfaces, and overloading, among others. Visual Basic also now supports structured exception handling, custom attributes and also supports multi-threading.

Visual Basic .NET is also CLS compliant, which means that any CLS-compliant language can use the classes, objects, and components you create in Visual Basic .NET.

Managed Extensions for C++ and attributed programming are just some of the enhancements made to the C++ language. Managed Extensions simplify the task of migrating existing C++ applications to the new .NET Framework.

C# is Microsoft’s new language. It’s a C-style language that is essentially “C++ for Rapid Application Development”. Unlike other languages, its specification is just the grammar of the language. It has no standard library of its own, and instead has been designed with the intention of using the .NET libraries as its own.

Microsoft Visual J# .NET provides the easiest transition for Java-language developers into the world of XML Web Services and dramatically improves the interoperability of Java-language programs with existing software written in a variety of other programming languages.

Active State has created Visual Perl and Visual Python, which enable .NET-aware applications to be built in either Perl or Python. Both products can be integrated into the Visual Studio .NET environment. Visual Perl includes support for Active State’s Perl Dev Kit.

Other languages for which .NET compilers are available include

- FORTRAN
- COBOL
- Eiffel

Fig1 .Net Framework

<table>
<thead>
<tr>
<th>ASP.NET</th>
<th>Windows Forms</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML WEB SERVICES</td>
<td>Base Class Libraries</td>
</tr>
<tr>
<td></td>
<td>Common Language Runtime</td>
</tr>
<tr>
<td></td>
<td>Operating System</td>
</tr>
</tbody>
</table>
C#.NET is also compliant with CLS (Common Language Specification) and supports structured exception handling. CLS is set of rules and constructs that are supported by the CLR (Common Language Runtime). CLR is the runtime environment provided by the .NET Framework; it manages the execution of the code and also makes the development process easier by providing services.

C#.NET is a CLS-compliant language. Any objects, classes, or components that created in C#.NET can be used in any other CLS-compliant language. In addition, we can use objects, classes, and components created in other CLS-compliant languages in C#.NET. The use of CLS ensures complete interoperability among applications, regardless of the languages used to create the application.

**CONSTRUCTORS AND DESTRUCTORS**:

Constructors are used to initialize objects, whereas destructors are used to destroy them. In other words, destructors are used to release the resources allocated to the object. In C#.NET the sub finalize procedure is available. The sub finalize procedure is used to complete the tasks that must be performed when an object is destroyed. The sub finalize procedure is called automatically when an object is destroyed. In addition, the sub finalize procedure can be called only from the class it belongs to or from derived classes.

**GARBAGE COLLECTION**

Garbage Collection is another new feature in C#.NET. The .NET Framework monitors allocated resources, such as objects and variables. In addition, the .NET Framework automatically releases memory for reuse by destroying objects that are no longer in use. In C#.NET, the garbage collector checks for the objects that are not currently in use by applications. When the garbage collector comes across an object that is marked for garbage collection, it releases the memory occupied by the object.

**OVERLOADING**

Overloading is another feature in C#. Overloading enables us to define multiple procedures with the same name, where each procedure has a different set of arguments.
Besides using overloading for procedures, we can use it for constructors and properties in a class.

MULTITHREADING:

C#.NET also supports multithreading. An application that supports multithreading can handle multiple tasks simultaneously, we can use multithreading to decrease the time taken by an application to respond to user interaction.

STRUCTURED EXCEPTION HANDLING

C#.NET supports structured handling, which enables us to detect and remove errors at runtime. In C#.NET, we need to use Try…Catch…Finally statements to create exception handlers. Using Try…Catch…Finally statements, we can create robust and effective exception handlers to improve the performance of our application.

THE .NET FRAMEWORK

The .NET Framework is a new computing platform that simplifies application development in the highly distributed environment of the Internet.

OBJECTIVES OF .NET FRAMEWORK

1. To provide a consistent object-oriented programming environment whether object codes is stored and executed locally on Internet-distributed, or executed remotely.

2. To provide a code-execution environment to minimizes software deployment and guarantees safe execution of code.

3. Eliminates the performance problems.

There are different types of application, such as Windows-based applications and Web-based applications.

FEATURES OF SQL-SERVER

The OLAP Services feature available in SQL Server version 7.0 is now called SQL Server 2000 Analysis Services. The term OLAP Services has been replaced with the term Analysis Services. Analysis Services also includes a new data mining component. The Repository component available in SQL Server version 7.0 is now called Microsoft SQL Server 2000 Meta...
Data Services. References to the component now use the term Meta Data Services. The term repository is used only in reference to the repository engine within Meta Data Services

SQL-SERVER database consist of six type of objects. They are,

1. TABLE
2. QUERY
3. FORM
4. REPORT
5. MACRO

TABLE:

A database is a collection of data about a specific topic.

VIEWS OF TABLES:

We can work with a table in two types,

1. Design View
2. Datasheet View

1. Design View

To build or modify the structure of a table we work in the table design view. We can specify what kind of data will be hold.

2. Datasheet View

To add, edit or analyses the data itself we work in tables datasheet view mode.

QUERY:

A query is a question that has to be asked the data. Access gathers data that answers the question from one or more table. The data that make up the answer is either dynaset (if you edit it) or a snapshot (it cannot be edited). Each time we run query, we get latest information in the dynaset. Access either displays the dynaset or snapshot for us to view or perform an action on it, such as deleting or updating.

AJAX:
ASP.NET Ajax marks Microsoft's foray into the ever-growing Ajax framework market. Simply put, this new environment for building Web applications puts Ajax at the front and center of the .NET Framework.

SYSTEM SPECIFICATION

Hardware Requirements:

- System : Pentium IV 2.4 GHz.
- Hard Disk : 40 GB.
- Floppy Drive : 1.44 Mb.
- Mouse : Optical Mouse.
- Ram : 512 Mb.

Software Requirements:

- Coding Language : ASP.Net with C#
- Data Base : SQL Server 2008.
MODULES:

- USER
- CLOUD BROKER
- CLOUD PROVIDER

USER:

It contains following steps:

- User Registration
- Login
- File Upload
- View accepted Files
- Request for space
- Download

USER REGISTRATION:

In this module new user register the information in order to use the cloud for space.

LOGIN:

In this module user can login by using his/her username and password.

FILE UPLOAD:

In this module each user can upload the file and requirements to the cloud broker for provider allocation.

VIEW ACCEPTED FILES:

In this module each user can view their own file is accepted or not.
REQUEST FOR SPACE:

In this module each user sent the request to the cloud broker for upload their file in cloud

DOWNLOAD:

In this module user can download their files for future use.

CLOUD BROKER:

- Login
- Accept Files
- View provider space
- Provider allocation

LOGIN:

By this module cloud broker can enter into process by using his name and password.

ACCEPT FILES:

In this module the broker can accept the user by accepting and rejecting their file depends on their cost.

VIEW PROVIDER SPACE:

In this module broker can view available space in each cloud server

PROVIDER ALLOCATION:
This component validates the user resource requirements. The validated requirements are broadcasted to all the cloud vendors. The cloud vendors respond with the assumed QoS parameters and cost. This information is validated and sent to the auction manager.

CLOUD PROVIDER MODULE:

In this module each user can upload the files depending on their cost to upload their files in cloud server (i.e.) cloud provider.

The cloud provider can view the files are upload to server.

SYSTEM DESIGN

Data Flow Diagram / Use Case Diagram / Flow Diagram

The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of the input data to the system, various processing carried out on these data, and the output data is generated by the system.
Use Case Diagram:
Data Flow Diagram:

Sequence Diagram:
Class Diagram:

USER
- User Name
- Password
- Email
- Phonenumber
- City
- REGISTRATION
- LOGIN
- UPLOAD
- DOWNLOAD
- VIEW FILES
- REQUEST OFFER

CLOUD BROKER
- User Name
- Password
- LOGIN
- VIEW REQUESTED DETAILS
- ACCEPT REQUEST
- VIEW PROVIDER SIZE
- PROVIDER ALLOCATION

CLOUD PROVIDER
- User Name
- Password
- LOGIN
- VIEW FILES
- VIEW PROVIDER SIZE
Activity Diagram:
SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS:

UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfied, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.
Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.
Invalid Input: identified classes of invalid input must be rejected.
Functions   : identified functions must be exercised.
Output      : identified classes of application outputs must be exercised.
Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

SYSTEM TESTING

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

WHITE BOX TESTING

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

BLACK BOX TESTING

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.
UNIT TESTING:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

INTEGRATION TESTING

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

ACCEPTANCE TESTING

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.
Currently, the cloud user pays a fixed price for resources or services. This type of pricing is called fixed pricing. Fixed pricing is very popular with telecom providers. On the flipside, there is no provision for incentives for users in the fixed strategy. Resource procurement is not only an important problem in cloud computing but is also an unexplored area. Currently, resource procurement is done manually and there is a pressing need to automate it. To automate procurement, we have presented three mechanisms: C-DSIC, C-BIC, and C-OPT. C-DSIC is a low bid Vickrey auction. It is allocative efficient and individual rational but not budget balanced. If the mechanism is not budget balanced, then an external agency has to provide money to perform procurement. C-BIC is a weaker strategy compared to C-DSIC and it is Bayesian incentive compatible. In C-BIC, vendors reveal the truth only if other vendors reveal the truth, unlike C-DISC where vendors reveal the truth irrespective of others’ choices. C-BIC achieves budget balance and a locative efficiency but not individual rationality’s-OPT achieves both Bayesian incentive compatibility and individual rationality, which the other two mechanisms cannot achieve. This mechanism is immune to both overbidding and underbidding. If a cloud vendor overbids, then the incentive is reduced. If it underbids, then it may not be a winner. C-OPT is more general compared to both C-DSIC and C-BIC—even if cloud vendors use different distributions for cost and QoS, we can safely use C-OPT. Hence, C-OPT is the preferred mechanism in more cases in the real world. The experiments reveal an interesting pattern. The resource procurement cost reduces as the number of cloud vendors increase, irrespective of the mechanism implemented. The cost in C-BIC reduces more significantly, compared to the other two mechanisms.
BIBLIOGRAPHY


