

Price-Based Resource Allocation For Edge Computing: A Market Equilibrium Approach

Dr.B.Ratnakanth¹, M.Karuna², Valakanti Vamshi³, Rathod Rohith⁴, Pooja Yadav⁵, Eslavath Vinod Kumar⁶

¹ Professor, Department of CSE& HOD, Sri Indu Institute of Engineering & Technology, Hyderabad

² Assistant Professor, Department of CSE, Sri Indu Institute of Engineering & Technology, Hyderabad

^{3,4,5,6} IVthBtech Student, Department of CSE, Sri Indu Institute of Engineering & Technology, Hyderabad

ABSTRACT

The emerging edge computing paradigm promises to deliver superior user experience and enable a wide range of Internet of Things (IoT) applications. In this paper, we propose a new market-based framework for efficiently allocating resources of heterogeneous capacity-limited edge nodes (EN) to multiple competing services at the network edge. By properly pricing the geographically distributed ENs, the proposed framework generates a market equilibrium (ME) solution that not only maximizes the edge computing resource utilization but also allocates optimal resource bundles to the services given their budget constraints. When the utility of a service is defined as the maximum revenue that the service can achieve from its resource allotment, the equilibrium can be computed centrally by solving the Eisenberg-Gale (EG) convex program. We further show that the equilibrium allocation is Pareto-optimal and satisfies desired fairness properties including sharing incentive, proportionality, and envy-freeness. Also, two distributed algorithms, which efficiently converge to an ME, are introduced. When each service aims to maximize its net profit (i.e., revenue minus cost) instead of the revenue, we derive a novel convex optimization problem and rigorously prove that its solution is exactly an ME. Extensive numerical results are presented to validate the effectiveness of the proposed techniques.

1. Introduction

1.1 Introduction

The last decade has witnessed an explosion of data traffic over the communication network attributed to the rapidly growing cloud computing and pervasive mobile devices. This trend is expected to continue for the foreseeable future with a whole new generation of applications including 4K/8K UHD video, tactile Internet, virtual/augmented reality (VR/AR), and a variety of IoT applications. As the cloud infrastructure and number of devices continue to expand at an accelerated rate, a tremendous burden will be put on the network.

Cloud data centers (DC) are often geographically distant from the end-user, which induces enormous network traffic, along with significant communication delay and jitter. Therefore, despite the immense power, cloud computing alone is facing growing limitations in satisfying the stringent requirements in terms of latency, reliability, security, mobility, and localization of new systems and applications (e.g., embedded artificial intelligence, mission critical communication, 5G wireless systems). To this end, edge computing (EC), also known as fog computing (FC), has emerged as a novel computing paradigm that complements the cloud and addresses many shortcomings

in the traditional cloud model.

In EC, storage, computing, control, and networking resources are placed closer to end-users, things, and sensors. Today, EC is still in the developing stages and presents many new challenges, such as network architecture design, programming models and abstracts, IoT support, service placement. On the other hand, different services may have different requirements and properties. Some services can only be handled by ENs satisfying certain criteria. Furthermore, different services may be given different priorities.

The basic idea behind our approach is to assign different prices to resources of different ENs. In particular, highly sought-after resources are priced high while prices of under-demanded resources are low. We assume that each service has a certain budget for resource procurement.

In these scenarios, it is important to consider both fairness and efficiency. Thus, conventional schemes such as social welfare maximization, max min fairness, and auction models may not be suitable. To strive the balance between fairness and efficiency, we advocate the General Equilibrium Theory, with a specific focus on the Fisher market model, as an effective solution concept for this problem.

1.2 Modules:

There are four modules can be divided here for this project they are listed as below

- **UPLOAD PRODUCTS**
- **PRODUCT REVIEW BASED ORDER**
- **RATINGS AND REVIEWS**
- **DATA ANALYSIS**

1.2.1 Upload Products

Uploading the products is done by admin. Authorized person is uploading the new arrivals to system that are listed to users.

1.2.2 Product review based order

The suggestion to user's view of products is listed based on the review by user and rating to particular item. Naïve bayes algorithm is used in this project to develop the whether the sentiment of given review is positive or negative

1.2.3 Ratings and reviews

Ratings and reviews are main concept of the project in order to find effective product marketing. The main aim of the project is to get the user reviews based on how they purchased or whether they purchased or not.

1.2.4 Data analysis

The main part of the project is to analysis the ratings and reviews that are given by the user. The products can be analysis based on the numbers which are given by user.

2. Literature Survey

2.1 Introduction

1. Fog and IoT: an overview of research opportunities:

Fog is an emergent architecture for computing, storage, control, and networking that distributes these services closer to end users along the cloud-to-things continuum. It covers both mobile and wireline scenarios, traverses across hardware and software, resides on network edge but also over access networks and among end users, and includes both data plane and control plane.

2. The emergence of edge computing:

Industry investment and research interest in edge computing, in which computing and storage nodes are placed at the Internet's edge in close proximity to mobile devices or sensors, have grown dramatically in recent years.

3. Existence of equilibrium for a competitive economy:

Wald has presented a model of production and a model of exchange and proofs of the existence of an equilibrium for each of them. Here proofs of the existence of an equilibrium are given for an integrated model of production, exchange and consumption

4. CloudFog: leveraging fog to extend cloud gaming for thin-client MMOG with high quality of service:

With the increasing popularity of Massively Multiplayer Online Game (MMOG) and fast growth of mobile gaming, cloud gaming exhibits great promises over the conventional MMOG gaming model as it frees players from the requirement of hardware and game installation on their local computers.

2.2 Existing system

In existing system, we study how to allocate resources from multiple ENs to multiple services. We exploit the General Equilibrium, a Nobel prize-winning theory, to construct an efficient market-based resource allocation framework. Although this concept was proposed more than 100 years ago, only until 1954, the existence of an ME was proved under mild conditions in the seminal work of Arrow and Debreu.

However, their proof based on fixed-point theorem is no constructive and does not give an algorithm to compute equilibrium.

2.2.1 Drawbacks in existing system

- Complexity
- Lack of Control
- Market Manipulation
- Potential for Unfairness

2.3 Proposed System

In proposed system, the models are inspired by the Fisher market which is a special case of the exchange market model in the General Equilibrium theory. An exchange market model consists of a set of economic agents trading different types of divisible goods.

Each agent has an initial endowment of goods and a utility function representing their preferences for the different bundles of goods. The goal of the market is to find the equilibrium prices and allocations that maximize every agent's utility respecting the budget constraint, and the market clears.

In the Fisher market model, every agent comes to the market with an initial endowment of money only and wants to buy goods available in the market. We cast the EC resource allocation problem as a Fisher market. We not only show appealing fairness properties of the equilibrium allocation, but also introduce efficient distributed algorithms to find an ME.

ADVANTAGES

- Efficient Resource Allocation.
- Fairness
- Scalability
- Incentivizes Innovation.

3. Feasibility Study

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are:

- **ECONOMICAL FEASIBILITY**
- **TECHNICAL FEASIBILITY**
- **SOCIAL FEASIBILITY**

3.1 Economical Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The

expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

3.2 Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement; as only minimal or null changes are required for implementing this system.

3.3 Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

4. Implementation

SYSTEM ARCHITECTURE



4.1 Python

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

1. Python is Interpreted – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

2. Python is Interactive – You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

3. Python is Object-Oriented – Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

4. Python is a Beginner's Language – Python is a great language for the beginner level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

4.1.1 Features

1. Easy-to-learn – Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.

2. Easy-to-read – Python code is more clearly defined and visible to the eyes.

3. Easy-to-maintain – Python's source code is fairly easy-to-maintain.

4. A broad standard library – Python's bulk of the library is very portable and cross platform

compatible on UNIX, Windows, and Macintosh.

5. Interactive Mode – Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

6. Portable – Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

7. Extendable – You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

8. Databases – Python provides interfaces to all major commercial databases.

9. GUI Programming – Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

10. Scalable – Python provides a better structure and support for large programs than shell scripting.

4.2 Django

Django is a Web framework written in Python. A Web framework is a software that supports the development of dynamic Web sites, applications, and services. It

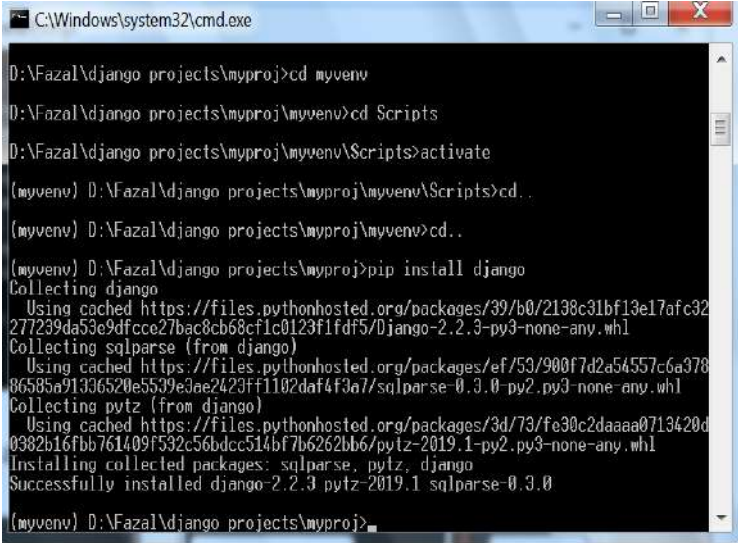
provides a set of tools and functionalities that solves many common problems associated with Web development, such as security features, database access, sessions, template processing, URL routing, internationalization, localization, and much more.

Using a Web framework, such as Django, enables us to develop secure and reliable Web applications very quickly in a standardized way. The development of Django is supported by the Django Software Foundation, and it's sponsored by companies like JetBrains and Instagram.

4.2.1 Installation

Now that we have the venv activated, run the following command to install Django:

```
pip install Django
```



```
C:\Windows\system32\cmd.exe
D:\Fazal\django projects\myproj>cd myvenv
D:\Fazal\django projects\myproj\myvenv>cd Scripts
D:\Fazal\django projects\myproj\myvenv\Scripts>activate
(myvenv) D:\Fazal\django projects\myproj\myvenv\Scripts>cd ..
(myvenv) D:\Fazal\django projects\myproj\myvenv>cd ..
(myvenv) D:\Fazal\django projects\myproj>pip install django
Collecting django
  Using cached https://files.pythonhosted.org/packages/39/b0/2138c31bf13e17afc32277239da53e9dfcce27bac8cb68cflc0123f1fd5/Django-2.2.3-py3-none-any.whl
Collecting sqlparse (from django)
  Using cached https://files.pythonhosted.org/packages/ef/53/900f7d2a54557c6a37806585a91306520e5539e3ae2423ff1102daf4f3a7/sqlparse-0.3.0-py2.py3-none-any.whl
Collecting pytz (from django)
  Using cached https://files.pythonhosted.org/packages/3d/73/fe30c2daaaa0713420d0382b16fbb761409f532c56bdcc514bf7b6262bb6/pytz-2019.1-py2.py3-none-any.whl
Installing collected packages: sqlparse, pytz, django
Successfully installed django-2.2.3 pytz-2019.1 sqlparse-0.3.0
(myvenv) D:\Fazal\django projects\myproj>
```

To start a new Django project, run the command below:

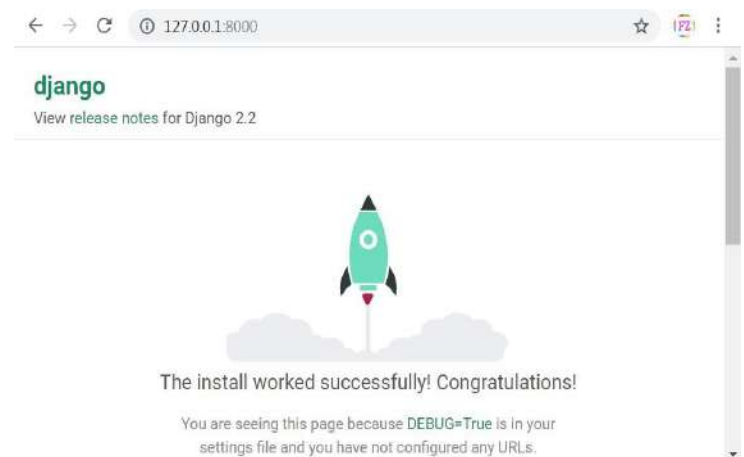
```
django-admin startproject myproject
```

The command-line utility `django-admin` is automatically installed with Django. After we run the command above, it will generate the base folder structure for a Django project. Our initial project structure is composed of five files:

1. `manage.py`: a shortcut to use the `django-admin` command-line utility. It's used to run management commands related to our project. We will use it to run the development server, run tests, create migrations and much more.
2. `__init__.py`: this empty file tells Python that this folder is a Python package.
3. `Settings.py`: this file contains all the project's configuration.
4. `urls.py`: this file is responsible for mapping the routes and paths in our project. For example, if you want to show something in the URL `/about/`, you have to map it here first.
5. `wsgi.py`: this file is a simple gateway interface used for deployment. You don't have to bother about it. Just let it be for now.

Django comes with a simple web server installed. It's very convenient during the development, so we don't have to install anything else to run the project locally. We can test it by executing the command: `python manage.py run server`. For now, you can ignore the migration errors; we will get to that later. Now open the following URL in a Web browser:

`http://127.0.0.1:8000` and you should see the following page:



5. Analysis

The main part of the project is to analysis the ratings and reviews that are given by the user. The products can be analysis based on the numbers which are given by user. The user data analysis of the data can be done by charts format. The graphs may vary like pie chart, bar chart or some other charts.

5.1 System Specification

5.1.1 Software Requirements:

- Operating System :
Windows 7.
- Coding Language :
Python.
- Front-End :
Python.
- Designing :
Html,css,javascript.
- Data Base :
MySQL.

5.1.2 Hardware Requirements:

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

- Monitor : 14'
Colour Monitor.

- Mouse :
Optical Mouse.

- Ram :
512mb

6. Testing

6.1 Introduction

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.

6.2 Type of tests

6.2.1 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.

6.2.2 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 satisfactory, as shown by successful 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.

6.2.3 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.

6.2.4 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.

6.2.5 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 used to test areas that cannot be reached from a black box level.

6.2.6 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, such as specification or requirements document.

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

7. Conclusion

In this work, we consider the resource allocation for an EC system which consists geographically distributed heterogeneous ENs with different configurations and a collection of services with different desires and buying power. Our main contribution is to suggest the famous concept of General Equilibrium in Economics as an effective solution for the underlying EC resource allocation problem. The proposed solution produces an ME that not only Pareto-efficient but also possesses many attractive fairness properties. The potential of this approach are well beyond EC applications. For example, it can be used to share storage space in edge caches to different service providers. We can also utilize the proposed framework to share resources (e.g., communication, wireless channels) to different users or groups of users

(instead of services and service providers). Furthermore, the proposed model can extend to the multi-resource scenario where each buyer needs a combination of different resource types (e.g., storage, bandwidth, and compute) to run its service. We will formally report these cases (e.g., network slicing, NFV chaining applications) in our future work. The proposed framework could serve as a first step to understand new business models and unlock the enormous potential of the future EC ecosystem. There are several future research directions. For example, we will investigate the ME concept in the case when several edge networks cooperate with each other to form an edge/fog federation. Investigating the impacts of the strategic behavior on the efficiency of the ME is another interesting topic. Note that N. Chen et. al. have shown that the gains of buyers for strategic behavior in Fisher markets are small. Additionally, in this work, we implicitly assume the demand of every service is unlimited. It can be verified that we can add the maximum number of requests constraints to the EG program to capture the limited demand case, and the solution of this modified problem is indeed an ME. However, although the optimal utilities of the services in this case are unique, there can have infinite number of equilibrium prices. We are

investigating this problem in our ongoing work. Also, integrating the operation cost of ENs into the proposed ME framework is a subject of our future work. Finally, how to compute market equilibria with more complex utility functions that capture practical aspects such as task moving expenses among ENs and data privacy is an interesting future research direction. It is also interesting to test the performance of the proposed approach on real datasets of an EC system when EC is widely deployed.

8. Reference

- [1] M. Chiang and T. Zhang, “Fog and IoT: an overview of research opportunities,” *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec. 2016.
- [2] M. Satyanarayanan, “The emergence of edge computing,” *Computer*, vol. 50, no. 1, pp. 30–39, Jan. 2017.
- [3] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge computing: vision and challenges,” *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.
- [4] K.J. Arrow and G. Debreu, “Existence of equilibrium for a competitive economy,”

Econometrica, vol. 22, no. 3, pp. 265–290, 1954.

[5] W.C. Brainard and H.E. Scarf, “How to compute equilibrium prices in 1891,” Cowles Foundation, Discussion Paper, no. 1272, 2000.

[6] A. Mas-Colell, M. D. Whinston, and J. R. Green, “Microeconomic Theory”, 1st ed. New York: Oxford Univ. Press, 1995.

[7] H. Moulin, “Fair division and collective welfare,” MIT Press, 2004.

[8] N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, “Algorithmic Game Theory”, Cambridge, U.K.: Cambridge Univ. Press, 2007.

[9] E. Eisenberg and D. Gale, “Consensus of subjective probabilities: The pari-mutual method,” Annals of Mathematical Statistics, vol. 30, pp. 165–168, 1959.

[10] E. Eisenberg, “Aggregation of utility functions,” Manage. Sci. 7, PP. 337–350, 1961.