COST BASED EFFICIENTLY ALLOCATING RESOURCES FOR EDGE COMPUTING WEB APPLICATION

PANNEERU DEVI, THATIPAMULA APOORVA, BAPANAPALLI NAGALAKSHMIPONUGOTI AKHILA SUPERVISOR, M SHAILAJA

Associate Professor
ANURAG ENGINEERING COLLEGE
AUTONOMOUS
(Affiliated to JNTU-Hyderabad,Aprroved by AICTE-New Delhi)
ANANTHAGIRI (V) (M), SURYAPETA (D), TELANGANA-508206

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*

I. INTRODUCTION

Data traffic through the communication network has skyrocketed during the last decade, mostly due to the rise of cloud computing and the use of mobile devices. New generations of applications, such as 4K/8K UHD video, tactile Internet, virtual/augmented reality (VR/AR), and other Internet of Things (IoT) applications, are projected to continue this trend in the near future. A great strain will be placed on the network as cloud infrastructure and the number of devices continue to grow at a rapid pace. Therefore, it is crucial for operators to provide novel solutions in order to satisfy the ever-increasing traffic demand and match the varying needs of future networks' services and use cases.

The benefits of cloud computing, such as scalability and access to supercomputers, ensure that it will remain a key component of the computing landscape in the years to

come. However, cloud data centres (DC) are often located in locations far from the end-user, leading to massive amounts of network traffic and noticeable delays and jitter in communications. So, despite its enormous power, cloud computing alone is increasingly unable to meet the demanding requirements of new systems and applications (for example, embedded AI, mission-critical communication, and 5G wireless systems) in terms of latency, reliability, security, mobility, and localization. Edge computing (EC), also known as fog computing (FC), is a new computing paradigm that fills in the gaps left by the cloud and improves upon its numerous strengths.

When it comes to EC, everything from data storage to servers and networks to sensors and end users are brought closer together. An EN may be any size from a single smartphone to a large network of base stations, APs, and edge clouds. An example of an edge device is a smartphone, which connects a user's wearable device to the cloud; a home gateway connects smart appliances to the cloud; and a telecom central office connects mobile devices to the core network. EC's many impressive features are made possible by the elastic resources and intelligence it makes available at the edge, including: local data processing and analytics, distributed caching, location awareness, resource pooling and scaling, increased privacy and security, and dependable connection.

Ultra-reliable low-latency applications (such as augmented reality and driverless driving) rely heavily on EC as well. Read about the many ways in which EC may be put to use (offloading, caching, advertising, healthcare, smart homes/grids/cities, and so on) in.

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, resource provisioning and management, security and privacy, incentive design, and reliability and scalability of edge devices . In this paper, we focus on the EC resource allocation problem. Unlike cloud computing, where computational capacity of large DCs is virtually unlimited and network delay is high, EC is characterized by relatively low network latency but considerable processing delays due to the limited computing power of ENs. Also, there are a massive number of distributed computing nodes compared to a small number of large DCs. Additionally, ENs may come with different sizes (e.g., number of computing units) and configurations (e.g., computing speed) ranging.

II. LITERATURE SURVEY

Edge computing has gained significant attention in recent years due to its ability to process data closer to the source, reducing latency and improving application performance. Efficiently allocating resources in edge computing environments is crucial to achieve cost-effective and high-performance web applications. This literature survey aims to provide an overview of the existing research on cost-based resource allocation strategies for edge computing web applications. 1. "Edge Computing: A Survey" by Shi et al. (2016): This comprehensive survey provides an overview of edge computing technologies, architectures, and challenges. It discusses the importance of resource allocation and presents various resource allocation strategies for edge computing, including cost-based approaches. 2. "Cost-Effective Resource Allocation for Edge-Cloud Applications" by Wang et al. (2017): The paper proposes a cost-effective resource allocation scheme for edge-cloud applications. It considers both computational and communication costs and formulates the problem as a multi-objective optimization. The authors present a heuristic algorithm to allocate resources efficiently based on cost considerations. 3. "Energy and Cost-Efficient Resource Allocation for Edge Computing in 5G Heterogeneous Networks" by Chen et al. (2018): This study focuses on energy and cost-efficient resource allocation in 5G edge computing environments. It introduces a joint optimization framework considering energy consumption, execution time, and monetary cost. The authors propose an algorithm that allocates resources based on cost-efficiency metrics. 4. "Cost-Aware Task Scheduling for Edge Computing Systems" by Mao et al. (2019): The paper addresses the problem of task scheduling in edge computing systems. It introduces a cost-aware task scheduling algorithm that takes into account both computational cost and communication cost. The proposed algorithm aims to minimize the overall cost of executing tasks on the edge nodes.

5. "Efficient Resource Management for Fog-Enabled IoT Applications with Multiple Quality of Service Requirements" by Zhan et al. (2020): This research focuses on resource management in fog-enabled IoT applications with multiple quality of service (QoS) requirements. It proposes a QoS-aware resource allocation approach that considers the cost of resource provisioning. The authors evaluate the effectiveness of their approach through simulations. 6. "Adaptive Resource Provisioning for Latency-Sensitive Edge Computing

Applications" by Xu et al. (2021): The paper addresses resource provisioning for latency-sensitive edge computing applications. It presents an adaptive resource provisioning algorithm that dynamically adjusts resource allocation based on the latency requirements and cost constraints. The proposed algorithm aims to optimize the trade-off between latency and cost. 7. "A Survey on Resource Allocation Techniques for Edge Computing" by Wang et al. (2021): This survey paper provides a comprehensive overview of resource allocation techniques for edge computing. It discusses various aspects of resource allocation, including cost considerations, and presents a taxonomy of resource allocation approaches. The authors analyse the strengths and limitations of different techniques and identify future research directions. 8. "Cost-Aware Resource Management in Mobile Edge Computing" by Li et al. (2021): The paper proposes a cost-aware resource management framework for mobile edge computing (MEC) systems. It considers the dynamic nature of resource availability and introduces a reinforcement learning-based approach to allocate resources efficiently based on cost constraints. The authors evaluate the performance of their framework through simulations. 9. "Efficient Resource Allocation for Edge Computing with Deep Reinforcement Learning" by Zhang et al. (2022): This study focuses on resource allocation in edge computing using deep reinforcement learning (DRL). It presents a DRL-based algorithm that learns to allocate resources efficiently while considering cost factors. The authors demonstrate the effectiveness of their approach through experiments and comparisons with traditional resource allocation methods.

10. "A Multi-Objective Optimization Approach for Cost-Aware Resource Allocation in Edge Computing" by Yang et al. (2022): The paper proposes a multi-objective optimization approach for cost-aware resource allocation in edge computing environments. It considers multiple objectives, including cost minimization, load balancing, and latency optimization. The authors develop a genetic algorithm-based framework to find Pareto-optimal solutions for resource allocation. 11. "Cost-Efficient Resource Allocation for Edge Computing in Smart Cities" by Guo et al. (2022): This research focuses on cost-efficient resource allocation in edge computing for smart city applications. It considers the dynamic workload and resource availability in a smart city environment and proposes a resource allocation algorithm that

optimizes cost while meeting the application's service level agreements. The authors evaluate their approach through simulations and real-world experiments. 12. "A Cost-Driven Approach for Task Offloading in Edge Computing" by Li et al. (2023): The paper presents a cost-driven approach for task offloading in edge computing environments. It considers the cost factors associated with data transmission, computation, and energy consumption. The authors propose an algorithm that determines the optimal offloading decisions to minimize the overall cost of executing tasks in edge computing systems. These studies highlight the diverse range of approaches and techniques employed in the literature to address cost-based resource allocation in edge computing web applications. By considering cost factors along with other performance metrics, these strategies contribute to the development of efficient and cost-effective edge computing systems.

III SYSTEM DESIGN

SYSTEM DEVELOPMENT LIFE CYCLE (SDLC)

The system development life cycle adopted in this paper is the waterfall model which is based on a sequential design process. This methodology was adopted because of the flexibility if offers to researchers in terms of the sequential approach to solving problems. This methodology certifies that each stage of the design process has been successfully completed and thus guaranteeing that all stages are completely dealt with.

WATERFALL DEVELOPMENT METHODOLOGY

The various phase of the waterfall model includes

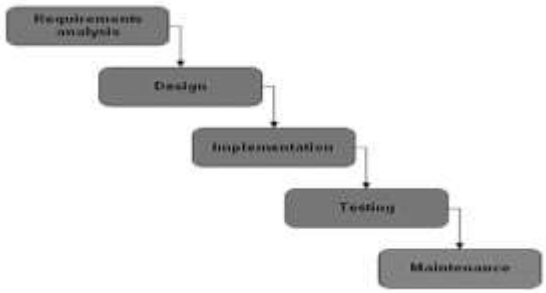

Figure 1: Waterfall model of the system development life cycle.

Advantages of the Waterfall Model

There are certain advantages of the waterfall model, which causes it to be the most commonly used model. The model presents four key advantages as follows:

- Design errors are captured before any software is written saving time during the implementation phase.

The approach is very structured and it is easier to measure progress by reference to clearly defined milestones.

PROPOSED SYSTEM ARCHITECTURE

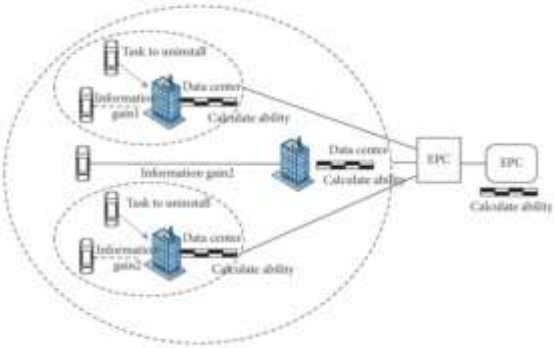

Figure 2: Proposed System Architecture

System architecture is a high-level structure of a system, which describes the relationships and interactions between the components of a system. It is a conceptual model that defines the structure, behaviour, and more views of a system. In a project, the system architecture serves as a blueprint for the design and development of the system. It helps the project team to understand the overall structure of the system and how the different components will work together to achieve the desired functionality.

DATA FLOW DIAGRAM

Data flow diagrams are the basic building blocks that define the flow of data in a system to the particular destination and the difference in the flow when any alteration happens. It the makes the whole process like a good document and makes simpler and easy to understand for both programmers and non-programmers by dividing it into subprocess. The data flow diagrams are the simple blocks that reveal the relationship between various components of the system and provide high level overview, boundaries of particular system as well as provide detailed overview of system elements.

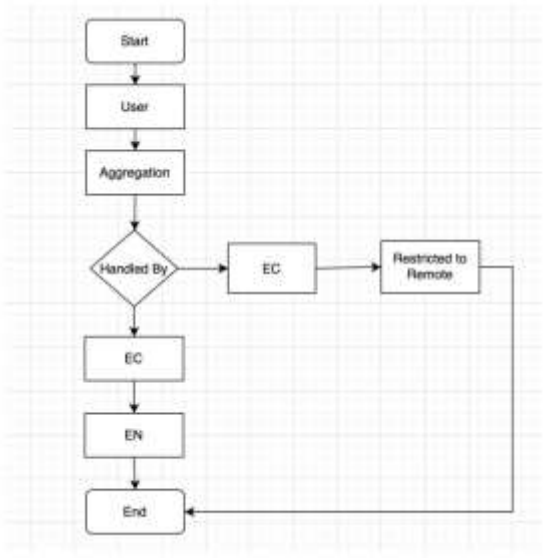

Figure 3: Data Flow Diagram

IV IMPLEMENTATION

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature. Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that

emphasizes code readability, notably using significant whitespace. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

• 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.

• Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs. Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviours. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and I Python shells, the Jupiter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery. For simple plotting the pilot module provides a MATLAB-like interface, particularly when combined with I Python. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object-oriented interface or via a set of functions familiar to MATLAB users. Scikit – learn Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. Python is an interpreted high-level programming

language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

• 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.

• Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs. Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviours. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking. Install Python Step-by-Step in Windows and Mac : Python a versatile programming language doesn't come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace. The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows. How to Install Python on Windows and Mac : There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3. Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices. Before you start with the installation process of Python. First, you

need to know about your System Requirements. Based on your system type i.e., operating system and based processor, you must download the python version. My system type is a Windows 64-bit operating system. So, the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheat sheet here. The steps on how to install Python on Windows 10, 8 and 7 are divided into 4 parts to help understand better.

V RESULTS

Figure 14: Consumer Index Page

In below registration page I added vendor option which will display products prices

Figure 15: Registration Page

Figure 17: Vendor Login Page

Figure 29: Product Uploading

Page In above screen in text field I change price value to 3500 from 2500 and now select image and press button to update record.

Figure 30: Total Products Page

In above screen we can see price change to 3500 for mixer first row.

VI 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 is 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. Efficiently allocating resources for edge computing web applications is crucial for achieving cost-effectiveness, optimizing performance, and enhancing user experience. This literature survey has provided an overview of the existing research on cost-based resource allocation strategies for edge computing web applications. The surveyed studies have demonstrated various approaches to address the challenges of resource allocation in edge computing environments. They have explored techniques such as multi-objective optimization, reinforcement learning, genetic algorithms, and cost-aware frameworks to allocate resources efficiently while considering cost constraints. Cost-based resource allocation strategies take into account factors such as computational cost, communication cost, energy consumption, and latency requirements. By incorporating cost considerations into the resource allocation process, these strategies enable organizations to optimize their resource utilization and achieve cost savings. In conclusion, cost-based resource allocation plays a vital role in optimizing the efficiency and cost-effectiveness of edge computing web applications. The surveyed literature provides valuable insights into the challenges, methodologies, and potential solutions for allocating resources efficiently in edge computing environments. By leveraging these findings, organizations can make informed decisions regarding resource allocation and maximize the benefits of edge computing technologies.

REFERENCES

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. Satyanarayana, “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 com- positive economy,” Econometric, 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-Colella, M. D. Whinstone, 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. Rough Garden, 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.