Auction-Based Networks

1st Du Duong
Vanderbilt University
du.k.duong@vanderbilt.edu

1nd Binh Ho
Vanderbilt University
binh.t.ho@vanderbilt.edu

1rd Jeffrey Pan
Vanderbilt University
Jeffrey.w.pan@vanderbilt.edu

***Abstract*—This paper investigates the efficacy of auction-based versus first-come-first-serve (FCFS) resource allocation strategies in managing server service slots within a simulated network environment, particularly under varying levels of network load. The core of the study involves a detailed simulation framework, where the auction-based strategy incorporates a virtual currency system, allowing clients to bid for resources, thereby aligning resource allocation with user utility. Our analysis focuses on comparing the performance of auction-based and FCFS methods under different network load scenarios, ranging from underloaded to heavily overloaded conditions. We find that while both allocation strategies perform similarly under low to moderate loads, the auction-based method significantly outperforms the FCFS approach in high-load situations. This advantage is attributed to its ability to dynamically prioritize and allocate resources based on bid values and user utility, effectively managing resource contention and maximizing system throughput under peak demand. These results highlight the potential of auction-based allocation systems in distributed networks, especially in scenarios characterized by fluctuating and high resource demands. The study provides valuable insights into the design of efficient and adaptive resource allocation mechanisms, suggesting a shift towards more strategic and user-centric models in network resource management. Future work will explore the application of these strategies in real-world network environments and examine the scalability and adaptability of the auction-based approach.**

***Index Terms*—auction-based, round-robin, FCFS, utility, resource allocation, scheduling policy, requests**

I. Introduction

In the realm of large-scale, federated computing infrastructures, efficient resource allocation emerges as a pivotal challenge, particularly under conditions of synchronized demand patterns and shared resource ownership. Traditional allocation methods, such as proportional share in time-shared systems and batch scheduling in space-shared systems, often falter under the complexities of modern computing demands. These challenges include over-demand for resources leading to decreased efficiency, difficulty in accounting for user heterogeneity, and the inability to explicitly consider user utility in scheduling decisions.

Market-based mechanisms offer a promising alternative, aiming to allocate resources among competing interests while maximizing the overall utility of the users. Here, utility is understood as the measure of satisfaction or benefit that a user derives from consuming a particular resource, a crucial aspect often overlooked in traditional allocation methods. These market mechanisms operate on the principle of equating supply and demand through economic transactions, thereby encouraging users to express their resource needs in terms of utility. However, the application of such market-based designs to computing systems is not without its challenges. Computing environments typically lack natural market components like a clear concept of currency or resource production, and end-users often prefer simplicity over strategic behaviors. Addressing these challenges, our study explores an innovative approach to resource allocation in a secure gRPC network environment, utilizing auction-based and round-robin strategies within a simulation framework. This research aims to not only enhance the understanding of resource allocation dynamics in complex networked systems but also to contribute to the development of more efficient and equitable resource distribution methodologies.

II. Background/Motivation

The emergence of federated computing infrastructures, exemplified by PlanetLab, has highlighted the limitations of traditional resource allocation methods [1]. PlanetLab, a distributed, wide-area testbed consisting of over 800 machines hosted globally, experienced significant challenges in 2004 when synchronized academic deadlines led to a dramatic spike in resource demand. The existing proportional-share scheduling policy, which allocated an equal time-slice of resources to each user, proved inadequate under such extreme conditions, leading to sub-optimal resource utilization and reduced system usability.

Traditional approaches, such as proportional-share in time-shared systems and batch scheduling in space-shared systems, fall short in addressing the complex dynamics of modern federated systems, particularly in their inability to quantify and incorporate user utility into allocation decisions. Proportional-share scheduling, while offering equal access, often fails in high-demand scenarios, resulting in decreased individual efficiency and neglecting the varying levels of utility that users might derive from their resource allocation [2]. Batch scheduling, suitable for applications requiring significant resource shares, struggles with user heterogeneity and fails to explicitly consider the differing utility values [3].

In response to these challenges, market-based mechanisms have emerged as a promising alternative, aiming to allocate resources among competing interests while maximizing the overall utility of the users. These mechanisms operate on the principle of equating supply and demand through economic transactions, thereby encouraging users to express their resource needs in terms of utility [4]. Our study explores a market-based allocation model, utilizing virtual currency and auction-based mechanisms, within a secure gRPC network environment. This approach introduces components such as resource discovery, a virtual currency managed by a central bank, auction settlement procedures, and load balancing, aiming to create a system where users can express their resource needs in terms of utility, encouraging efficient and equitable resource distribution. The simulation uses a secure gRPC network environment, incorporating different types of computing nodes to emulate varied resource demands and assess the performance of allocation strategies. By analyzing the effectiveness of these strategies in a controlled simulation, this study seeks to contribute to the development of more robust resource allocation methodologies for real-world distributed systems, drawing lessons from the experiences of platforms like PlanetLab and the critical role of utility in resource allocation.

III. Design and Implementation

The overarching design of the auction-based resource allocation system that we are building can be split into 2 main parts: a server, which manages all operations related to resource allocation, and clients, who seek to consume the resources as permitted by the server. The server is consisted of several moving parts, including a load balancer, a central bank, an auction settler, and computing resources, which will be further explained in the upcoming subsections. A high-level design sketch of the system that we are building can be seen in Figure I.

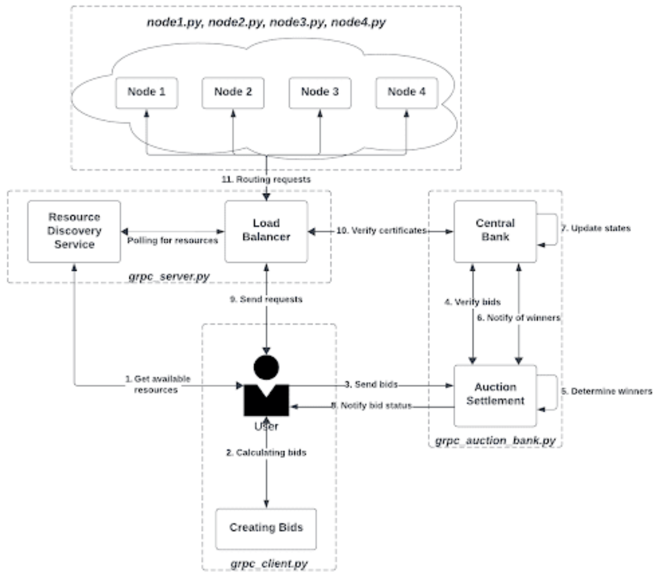

Fig. 1. High-level Design of an Auction-based Allocation System

A. Load Balancer

In the server of the system, a load balancer is implemented as a tool to efficiently and conveniently manage the state of computing resources. Given a resource, the load balancer can be used to query which user has access to that resource, how much time remains for their access, and the current load factor of the resource. Additionally, the load balancer can also serve as a resource discovery service, providing users with information about which resources are currently available for bidding

In our simulation, we specify the server to manage 5 different resources, numbered from 1 to 5. In order to simulate how computing resources vary in real life, we let resource 1 to be the ”least powerful” resource, and resource 5 to be the ”most powerful” resource. Details about how we incorporate this concept into our implementation of the system will be explained later on in the paper.

When a user sends a processing request to the server, the server will use the load balancer in order to route the request to the appropriate computing resources that the user has access to for processing. The more computing resources that the user has access to, the faster their requests can be processed. In our simulation, we use the following formula to determine how long it takes for the server to process a request given the computing resources of the user. Let R be the set of resources that the user has access to (for example, $R = \{2, 3, 5\}$), then the number of seconds it takes for the server to process a request is

$$T(R) = \frac{10}{1 + \sum_{r \in R} r}$$

.

B. Central Bank

The next component of the auction-based allocation system is the central bank. The central bank is designed to incentivize users by rewarding those who limit their system usage during peak demand periods, while penalizing those who either use resources excessively when they are scarce or underutilize them when resources are plentiful. In order to bid for resources, users must have a form of virtual currency to use in auctions, and the central bank is the entity within the server that manages the virtual currency balance of all users in the system. When a new user joins the system, the bank will automatically reward them with 100 units of currency to begin bidding. In addition, when a user uses the resource discovery service provided by the load balancer, they will also be notified of their current bank balance so that they can make better-informed bidding decisions. Moreover, when a user submits a bid, the bank will verify whether or now they have enough currency in their bank balance to complete the bid. If they do, then the bid is deemed to be verified and thus the user is allowed to participate in the auction. If the bid price exceeds the user’s bank balance, then the bid is deemed to be invalid, and the user is not allowed to participate in the auction. After the auction has settled, the bank is in charge of decreasing the winning users’ bank balance by the amount that they bid, and the aggregate sum of all of the collected amount of currency from this action will be equally redistributed to the losing users. This mechanism is implemented to incentivize users to save and accumulate currency for future use, thereby maximizing the utility of the system by avoiding allocating resources for low-priority jobs.

The central bank is also responsible for tax collection in the auction-based allocation system. In our simulation, every 30 seconds, the central bank will determine a threshold for tax collection purposes, which is defined using the following formula. Let B be a function that maps users to their bank balances and U be the set of all users in the system, then the threshold for tax collection is

$$TH = \frac{\sum_{u \in U} B(u)}{|U|}$$

After determining the threshold, the bank will collect tax from users whose balance is higher than the threshold and redistribute it back equally to users whose balance is lower than the threshold. In our simulation, the tax rate is imposed to be 10%. In other words, given a threshold TH, if a user u has $B(u) > TH$, then the new balance of the user u is $B_{new}(u) = 0.9 * B(u)$. The aggregate amount of tax currency collected from these users is then redistributed equally to users that have their bank balance below TH. This mechanism is implemented in order to allow an exhausted bank account to recover and keep contributing to the utility of the system and dissuade users from hoarding currency over long periods of time.

C. Auction Settler

The auction settler is a component in the system's server that is responsible for handling bids from users and determining the winning bids. The format of the auction is a sealed-bid, combinatorial auction, which means that the users will not be able to see other bidders and the users can bid on a combination of different resources. In particular, the format of a bid is

$$\{R, D, P\}$$

, where R is the set of resources that the user wants to bid for, D is the time duration for which the user wants to gain access to those resources, and P is the price that the user is willing to pay for the bid. Auction settlement is carried out at fixed time intervals, and in our simulation, auction settlement is carried out every 10 seconds. During the 10 seconds time period in between settlements, the auction settler accumulates bids from users to prepare for the upcoming settlement. When auction settlement is carried out, the auction settler locks the server from receiving more bids until settlement has completed in order to prevent race conditions.

After accumulating bids from users, the auction settler will determine the winners using a heuristic greedy algorithm. This is because the problem of determining the most optimal combination of resources to allocate is NP-Hard, so it would take a lot of time and computing power to determine the winners if we solve it optimally. In this case, a heuristic greedy algorithm is faster and more efficient, and it also ensures that the allocation is almost optimal, which is enough for the system to operate. More specifically, when a user submits a bid, the auction settler, after verifying that the bid is valid with the central bank, calculates the utility score for the bid using the following formula. Let W_r be the contention index of a particular resource R (valued from 0 to 1, computed based on how many users bid for that resource historically). Then the utility score of a bid is

$$U = \frac{P}{D(\sum_{r \in R} W_r)}$$

If a utility score for a particular bid is high, then that means the bid should win the auction and resources should be allocated to the corresponding user. Intuitively, this means that the auction settler favors users who are willing to pay more for their requests and who are wanting to gain access to uncontested resources in short periods of time. This way, the system ensures that all resources are utilized efficiently to process high-priority requests, not just resources with high computing power, and thus maximizing the overall utility of the system. In addition, by including duration in the formula, the system also incentivizes users to refrain from hoarding resources to themselves for long periods of time in order to ensure that the resources are fairly accessible by all users. In this case, 1 unit of duration corresponds to 10 seconds, which happen to be the time in between 2 settlements.

In terms of the implementation, the auction settler keeps track of a maximum heap in order to efficiently sort the bids in descending order of utility score from a stream of bids. When the time for settlement comes, the auction settler goes through the bids in the heap one by one. If the resources in the bid can be allocated to the corresponding user, then the auction settler contacts the load balancer to update the states of the resources to mark them as being accessible to the corresponding user. If the resources in the bid cannot be allocated to the corresponding user (i.e. when some resources have already been allocated to another user earlier in the heap), then the bid is skipped. This algorithm can also be referred to as a "first-fit" scheduling algorithm.

D. Client

The last component of the auction-based allocation system is the client. The client represents a user who is wanting to participate in the system by bidding for resources so that they can have their requests processed. When the client joins the system, it starts by querying for available computing resources from the server as well as its current bank balance via the resource discovery service exposed by the server. After receiving a list of available computing resources, it then determines which resources it wants to bid for. In order to simulate the behavior of a real user, the client selects a random subset of the set of all available resources using an aggression index. The higher the aggression index, the more computing resources the client wants to bid for. After choosing a list of resources that it wants to bid for, the client then determines how much it wants to pay for the bid. In order to simulate the behavior of a real user, the client values resources with high computing power more than resources with lower computing power, and it also determines the price based on its current bank balance. More specifically, the price of a bid can be computed as

$$P = B(u)\frac{\sum_{r\in R_A} r}{\sum_{r\in R} r}$$

where $B(u)$ is the current bank balance of the user, R_A is the list of all available resources, and R is the list of resources that the user wants to bid for.

After submitting the bid to the server, the client waits to be notified of the result of the settlement process. If the bid does not win, then the client waits for some time until it repeats the process again (i.e. from querying the resource discovery service). If the bid wins the auction, then the client starts to send requests to the server in order to be processed. The number of requests that the client sends is

$$(1 + \sum_{r\in R} r) \cdot D$$

This formula is used because it takes the server $\frac{10}{1+\sum_{r\in R} r}$ seconds to process the request. Thus, the total number of seconds that the client will use the allocated resources is

$$(\frac{10}{1+\sum_{r\in R} r}) \cdot (1 + \sum_{r\in R} r) \cdot D = 10 \cdot D$$

which is consistent with the definition of D (1 unit of duration corresponds to 10 seconds, which happen to be the time in between 2 settlements).

IV. Experiment Setup

A. Overview

Our experiment uses a simulated network environment to compare auction-based and first-come-first-serve (FCFS) resource allocation strategies [5]. The codebase consists of ‘grpc_server_auction.py‘, ‘grpc_client_auction.py‘, ‘grpc_server_FCFS.py‘, and ‘grpc_client_FCFS.py‘.

B. Network Configuration

We conducted the simulation in a Mininet topology, distributed between two machines on the Chameleon cloud platform. Configurations were 3 hosts (1 server, 2 clients), 5 hosts (1 server, 4 clients), and 7 hosts (1 server, 6 clients).

C. Client-Server Interaction

Clients send mock requests to the server, processed based on class/priority. The auction-based system uses a bidding mechanism, while FCFS allocates resources based on request order.

D. Data Collection and Metrics

The server records and outputs the number of responses completed within a set period. Clients log the processing time of their requests. These times are averaged to evaluate the efficiency of each strategy.

V. Results

In our experimental network setup, we utilized a total of seven nodes, while our testing environment was equipped with five distinct resources. This arrangement allowed us to thoroughly evaluate the system’s performance under various load conditions. Specifically, we analyzed three different scenarios: an underloaded system with three nodes, a fully loaded system with five nodes, and an overloaded system with seven nodes. To ensure a comprehensive assessment and to mitigate the impact of random fluctuations, each scenario was subjected to three separate trials.

Table I above present the aggregated results of these trials, showcasing the average values at specific timestamps throughout the experiment. This data offers valuable insights into the system’s behavior under different load conditions.

TABLE I
RESULTS OVERVIEW FOR FCFS AND AUCTION-BASED METHODS

First Come First Serve (FCFS) Method			
Timestamp	**3 Nodes**	**5 Nodes**	**7 Nodes**
5	37.000	48.000	44.333
15	124.33	147.33	148.000
25	197.33	204.667	219.667

Auction-Based Method			
Timestamp	**3 Nodes**	**5 Nodes**	**7 Nodes**
5	35.667	39.33	45.0
15	130.667	147.000	157.667
25	208.667	209.33	236.33

Following this, we have included a series of graphs that depict the number of requests processed over time. These visual representations show the dynamic performance of our network across the various timestamps, providing a clear depiction of how the system responds to underloaded, fully loaded, and overloaded states.

A. Graphs

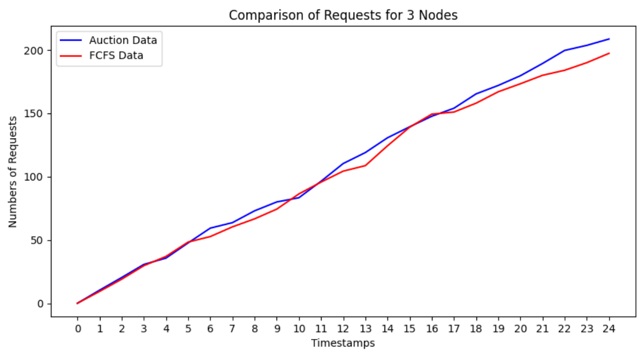

Fig. 2. Auction-based vs. FCFS for 3 nodes

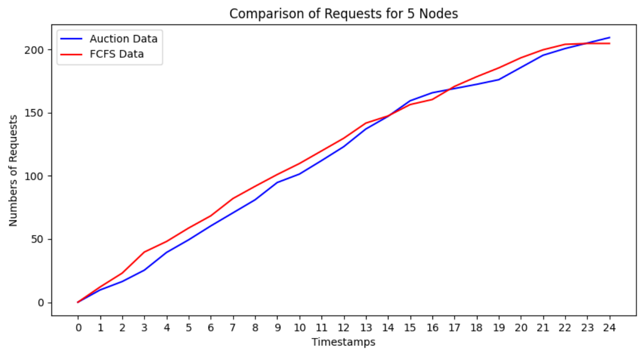

Fig. 3. Auction-based vs. FCFS for 5 nodes

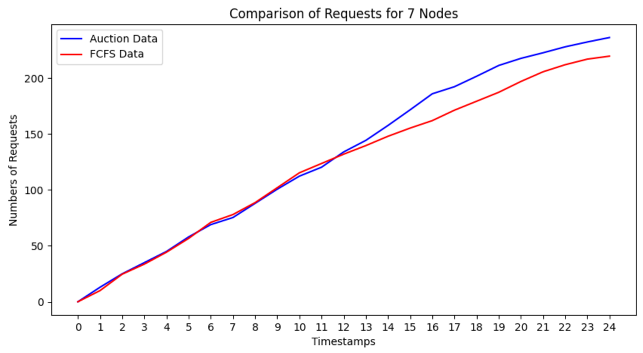

Fig. 4. Auction-based vs. FCFS for 7 nodes

VI. Discussion/Analysis

The provided tables and graphs display the performance trends of the First Come First Serve (FCFS) and Auction-based resource allocation methods across a network of varying node counts. Analyzing the processed request patterns, we can infer critical behavioral dynamics of our system under different load conditions.

For the underloaded scenario with 3 nodes, both methods demonstrate a similar upward trend in the number of processed requests, indicating that when resources are abundant, the system's allocation method does not significantly impact the performance. This convergence suggests that both FCFS and Auction-based methods are efficient when the demand for resources is not at its peak. Most of the difference in requests is most likely due to randomness in the trials.

As we transition to the fully loaded system with 5 nodes, the graphs show a marginal divergence between the two methods. This suggests that as we approach the system's capacity, the Auction-based method begins to display a slightly more consistent performance in request handling compared to FCFS. The Auction-based method's capability to prioritize requests based on user bids may contribute to this efficiency, ensuring that resources are allocated to the most valued requests during times of balanced demand and supply.

Lastly, in the overloaded scenario with 7 nodes, the Auction-based method's advantage becomes more pronounced, as indicated by the wider gap between the two methods' performance curves. The FCFS method shows signs of strain, processing fewer requests as it adheres to a strict sequential order without regard for the urgency or importance of the requests. Conversely, the Auction-based method appears to manage the resource contention more effectively, potentially by optimizing the allocation according to the users' expressed utility, therefore maximizing the throughput even when resources are stretched thin.

These results underscore the Auction-based method's robustness and adaptive efficiency, particularly in scenarios where resource contention is prevalent. The FCFS method, while simpler and more straightforward, may not be as well-equipped to handle high demand levels as the Auction-based method, which can dynamically adjust to the system's state and users' needs.

VII. Conclusion

This study provides a comparative analysis of auction-based and first-come-first-serve (FCFS) resource allocation strategies, particularly under varying network load conditions. Our results, derived from a simulation that includes a sophisticated auction system with a virtual currency mechanism, clearly demonstrate the superiority of the auction-based method in high-load scenarios.

The auction-based approach, with its strategic bidding and utility-focused resource distribution, consistently outperforms the FCFS method under heavy network loads. It adeptly manages resource contention, ensuring more efficient utilization of server service slots when the demand peaks. While both strategies perform comparably under low to moderate loads, the auction-based method's advantage becomes distinctly pronounced as network load intensifies.

These insights suggest that auction-based allocation is particularly advantageous in environments experiencing fluctuating and high demand, offering a dynamic and responsive solution for resource management in distributed systems. Future research should further investigate these strategies in varied real-world contexts, enhancing the adaptability and efficiency of network resource allocation.

In summary, our study underscores the effectiveness of auction-based resource allocation in high-load network environments, marking a significant stride in advancing resource management strategies in complex computing infrastructures.

References

[1] N. Spring, L. Peterson, A. Bavier, and V. Pai, "Using planetlab for network research: myths, realities, and best practices," *ACM SIGOPS Operating Systems Review*, vol. 40, no. 1, pp. 17–24, 2006.

[2] I. Stoica, H. Abdel-Wahab, K. Jeffay, S. K. Baruah, J. E. Gehrke, and C. G. Plaxton, "A proportional share resource allocation algorithm for real-time, time-shared systems," in *17th IEEE Real-Time Systems Symposium*. IEEE, 1996, pp. 288–299.

[3] N. Ye, T. Farley, X. Li, and H. Bashettihalli, "Batch scheduled admission control for computer and network systems," *Information Knowledge Systems Management*, vol. 5, no. 4, pp. 211–226, 2005.

[4] A. AuYoung, P. Buonadonna, B. N. Chun, C. Ng, D. C. Parkes, J. Shneidman, A. C. Snoeren, and A. Vahdat, "Two auction-based resource allocation environments: Design and experience," 2009.

[5] S. Kaur, K. Kumar, J. Singh, and N. S. Ghumman, "Round-robin based load balancing in software defined networking," in *2015 2nd international conference on computing for sustainable global development (INDIACom)*. IEEE, 2015, pp. 2136–2139.

[6] M. E. Fiuczynski, "Planetlab: overview, history, and future directions," *ACM SIGOPS Operating Systems Review*, vol. 40, no. 1, pp. 6–10, 2006.

[7] D. Niyato, N. C. Luong, P. Wang, and Z. Han, "Auction theory for computer networks," 2020.

[8] M. Dramitinos, G. D. Stamoulis, and C. Courcoubetis, "An auction mechanism for allocating the bandwidth of networks to their users," *Computer Networks*, vol. 51, no. 18, pp. 4979–4996, 2007.

[9] Y. Zhang, C. Lee, D. Niyato, and P. Wang, "Auction approaches for resource allocation in wireless systems: A survey," *IEEE Communications surveys & tutorials*, vol. 15, no. 3, pp. 1020–1041, 2012.

[10] E. L. Hahne, "Round-robin scheduling for max-min fairness in data networks," *IEEE Journal on Selected Areas in communications*, vol. 9, no. 7, pp. 1024–1039, 1991.