Pooling Approach for Task Allocation in the Blockchain Based Decentralized Storage Network

Iman Vakilinia*, Shahin Vakilinia†, Shahriar Badsha‡, Engin Arslan‡, Shamik Sengupta‡
*School of Computing, University of North Florida, Jacksonville, FL, USA
i.vakilinia@unf.edu
†PwC Data Analytics Team, Montreal, QC, CA
shahin.vakilinia@pwc.com
‡Computer Science and Engineering Department, University of Nevada, Reno, NV, USA
{sbadsha, earslan, ssengupta}@unr.edu

***Abstract*—Blockchain technology has provided a solid system to develop incentivization algorithms using the smart contract. Blockchain applies the distributed ledger to store transaction histories, and the information is stored across a network of computers instead of on a single server. This facilitates the development of a new set of applications such as distributed file storage systems where users can rent out their storage in return for a premium. The distributed file storage systems provide more privacy and security compared to the centralized storage models as there is no need to have a trusted party. New schemes have been developed for distributed file storage systems on top of the blockchain platform, however, the problem of task/service allocation in these models have not been studied before. In this paper, we study the task/service allocation in the distributed file storage systems considering the challenge of computation cost. First, we formalize the problem of task/service allocation in a decentralized storage network, and then we discuss different approaches to allocate storage tasks to storage servers in an efficient manner. Moreover, we study the benefits of the cooperation (a.k.a pooling) in the storage and retrieval markets of distributed storage networks. The evaluation results show the benefit of our proposed pooling based approach in storage and retrieval markets.**

***Index Terms*—Distributed Storage Network, Storage Market, Blockchain, Filecoin, Pooling**

I. Introduction

Nowadays the Internet plays a critical role in our life. However, the Internet has been influenced by a centralized architecture in which big companies such as Google, Facebook, and Microsoft dominate the daily Internet traffic. This raises security concerns due to the potential misbehavior of these entities. For example in 2018, it was revealed that Cambridge Analytica had harvested the personal data of millions of people's Facebook profiles without their consent and used it for political purposes [1]. To overcome this problem, decentralized models such as peer to peer algorithms have been proposed to relieve the dependency to a central party. Nevertheless, such models fail to motivate players to participate due to the lack of comprehensive incentivization mechanisms.

Recently, blockchain technology has offered a reliable framework to develop incentivization mechanisms using the smart contract which automatically moves digital assets following an arbitrary pre-specified program [2]. Blockchain applies the distributed ledger to store transaction histories, and the information is stored across a network of computers instead of on a single server. This facilitates the development of a new set of applications such as distributed file storage systems. These distributed file storage systems are providing an algorithmic market for storage to clients and servers. The client can store its own data or query public data from the system by making a payment to the network. On the other hand, the servers provide their storage resources to the network in return for a premium. Such distributed file storage systems can replace the central storage system providers. To protect the private data, an end to end encryption method can be used to encrypt the data and each file can be split into blocks which are stored in different nodes of the network. Therefore, it can offer enhanced security and privacy by eliminating the central entity that controls the data. Moreover, network nodes can rent out their excess storage, improving the efficiency of the network by reducing the maintenance cost of data-centers. Furthermore, these models are more transparent as their platforms are open-source compared to opaque operations of current data storage system providers.

There are two markets available in the decentralized storage networks as *Storage Market*, and *Retrieval Market* [3]. In the storage market, the storage providers store client data in return for a premium. This market is analogous to central storage services such as Dropbox, Google Drive, Microsoft OneDrive, and Apple iCloud. On the other hand, in the retrieval market, users issue inquiries for the public data from the network, and the network retrieves the data from servers. Retrieval market is similar to the BitTorrent network with the main distinction that the storage providers earn rewards when they reply to clients' requests. In this model, the data does not belong to a specific party, and storage providers are free to choose what data to store as there is no obligation for providing storage in comparison with the storage market.

While most previous works focused on the development of robust distributed storage systems based on blockchain [3]–[6], the analysis of the economic aspect of such a market has been mostly disregarded by the community. As the normal users are providing retail storage services in such a system, a proper mechanism needs to be deployed to assign tasks to servers based on the service requirements. To clarify the problem, consider the following example. Assume that there is a task requests for 5 TB of data storage with the replicated

978-3-903176-24-9

factor of 3 in the USA, China, and Europe. Although there is no single retail storage provider which can satisfy all of these requirements, the coalition of users can meet the task requirements. Here, the problem is how we can find an efficient coalition which satisfies the requested task with the cheapest cost. On the other hand, as the deployment and execution of a smart contract are costly, the proposed mechanism should be scalable to handle millions of nodes with the minimal cost of deployment and execution such that the cost of using decentralized storage network is affordable compared to the available centralized approaches. Considering this challenge, in this paper, we study the benefits of the cooperation in the storage and retrieval markets of distributed storage networks. First, we formalize the problem of task/service allocation in the decentralized storage network, and then we propose and analyze different approaches to allocate tasks to servers in an efficient manner with the goal of decreasing the cost of smart contract implementation.

The rest of the paper is organized as follows. In the next section, we present the system model. In Section III, we study the benefits of the auction and pooling approaches in the storage market, and in section IV, we study the benefit of pooling approach in retrieval market. We evaluate the benefit of pooling approach in section V. In section VI, we study the related works in the context of decentralized storage networks. Finally, section VII provides concluding remarks of the paper.

II. System Model

In this section, we define the system model. Our system model mainly follows Filecoin structure [3]. There are three main entities as the *Network*, *Servers*, and *Clients*. Network is an abstract entity which is responsible for maintaining the distributed storage service through managing requests and responses. Network is responsible to verify the honest behavior of the Servers and Clients. Network penalizes the users which do not follow the protocol as specified. Using the blockchain technology, Network stores a distributed ledger which is accessible to the participants. Network operations are managed through smart contracts.

Servers are two types as *Storer* (The terms storage provider, and storer are used interchangeably throughout this paper), and *Retriever*. Storer is responsible for storing data, while Retriever is responsible to retrieve data for queries. The honest behavior of Storer is checked by Network through proof of storage and proof of retrievability methods [7]. Storer pledges their service with a collateral which is going to be withdrawn and given to the clients in the case of inability to provide the service as it is obligated in the contract.

There are also two types of Clients as *Storage Requester* and *Data Requester*. The storage client request for a storage service which is specified with a set of service requirements. These service requirements include:

Storage size, the number of replication, backup options, throughput, accessibility time, number of supported users for concurrent download and upload, collateral to be withdrawn in the case of loss, etc.

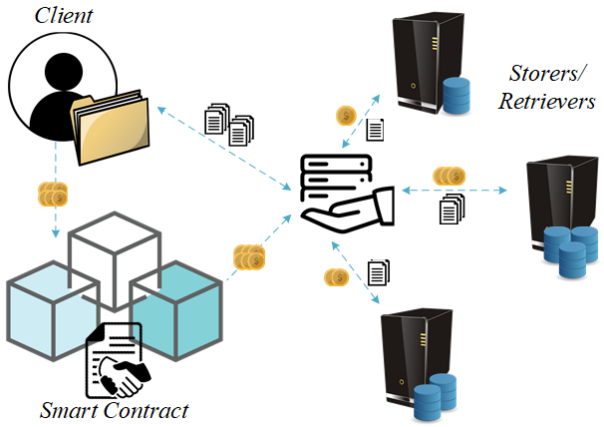

Fig. 1: Blockchain based decentralized storage network

On the other hand, the inquirer client is searching for a specific content. In this case, clients pay service providers in return for the search results. Financial transactions between clients and servers are automatically managed by the smart contract which eliminates the dependency to a central trusted party.

In the storage market, the storage clients submit their requests to the order-book which is a ledger in the blockchain accessible to the public, then the tasks are assigned to servers based on their resources. Note that, the network is responsible for validating the service availability, and the service providers are charged if they do not deliver the committed services. Storer deposits a collateral to the smart contract which will be withdrawn in the case that the service has not been provided as it has been committed.

In the retrieval market, the inquirer client submits its request to the order-book along with the payment. Then, the retriever transfers the data, and Network reimburses it accordingly. All of the smart contract operations are performed automatically and triggered by the users' actions. These operations are executed on a distributed decentralized blockchain network which makes the system independent to any trusted third party. Figure 1 demonstrates the overall picture of the entities and their interactions with each other in the blockchain based decentralized storage network.

III. Storage Market

In this section, we study the problem of service/task allocation in the storage market. In the decentralized storage network where typical users are providing retail services, we need a mechanism to efficiently assign the storage tasks to the users.

Let $S = \{s_1, ..s_N\}$ represent the set of servers and $C = \{c_1, ..., c_M\}$ represent the set of clients. Let $\bar{c}_i$ indicate a set of requirements for a storage task of client c_i as defined in system model. Let $\bar{s}_j$ represent a service features that storage server s_j provides to the system. Let $\preceq$ present the coverage of a task requirements by a server. For example, $\bar{c}_i \preceq \bar{s}_j$ indicates that the services which are provided by storage server s_j, covers the storage requirements of client c_i's task.

Let v_j associated with the $\bar{s}_j$, and indicates the cost that the storage server s_j demands in return for its service. Let $\hat{v}_i$ associated with the $\bar{c}_i$, and indicates a payment that client c_i is willing to make in return for its storage service. Therefore, we can formally model the storage task allocation problem as follows. For a storage request with the requirement of $\bar{c}_i$, an optimal storage allocation algorithm should find a set of service providers $J \subset S$ such that $\bar{c}_i \preceq \bar{s}_J$ with the least cost. Thus we have the following minimization problem:

$$\begin{aligned} \min \quad & f_c(J) = \sum_{j \in J} v_j \\ s.t. \quad & \bar{c}_i \preceq \bar{s}_J \end{aligned}$$

This optimization problem is reducible to the dual of the multidimensional Knapsack problem [8], and it is an NP-hard problem.

On the other hand, for each server with the service $\bar{s}_j$, an optimal task allocation algorithm should find a set of tasks $I \subset C$ such that $\bar{c}_I \preceq \bar{s}_j$ with the most premium. Thus we have the following maximization problem:

$$\begin{aligned} \max \quad & f_s(I) = \sum_{i \in I} \hat{v}_i \\ s.t. \quad & \bar{c}_I \preceq \bar{s}_j \end{aligned}$$

This optimization problem is reducible to the multidimensional Knapsack problem [8], and hence it is an NP-hard problem as well.

Therefore, the main problem is finding a tasks/service mappings which provide the maximum benefits for both clients and servers. The solution to this problem requires exhaustive search where we need to find every combination of task service allocation considering the permutation of tasks. Considering that M and N are in the scale of number of Internet users, then it is impossible to run such an algorithm in very strong servers, let alone a smart contract where it is resource-bounded and we need to make expensive payment for computations. Therefore, in the following, we are studying near-optimal solutions to alleviate the computation cost, considering the large scale of number of users.

A. Storage Auction

An open auction can be used to mitigate the computation cost for an efficient matching of the storage tasks and storage providers. The smart contract implementation of the auction is simple and it is computationally cheap as we describe in section V.

In this model, a client submits a task along with its requirements, maximum price she is willing to pay, and a time interval for the auction. The smart contract then shares the task request with storers. As the blockchain ledger is public, storers can see the current lowest bid, and issue competitive bids. The smart contract checks if the submitted bid is lower than the lowest bid and then updates the ledger accordingly. This approach significantly mitigates the need to solve the difficult task allocation optimization problem as there is no need for solving the optimization problem. In the worst case, the complexity of this auction algorithm is in the order of $O(N)$, where all of the storage providers submit their bids in an decreasing matter assuming that the bidders are bidding truthfully and they do not change their bids strategically based on other bidders' decisions. In the best case, the complexity of the algorithm is $O(1)$ where the first bidder submits the best offer.

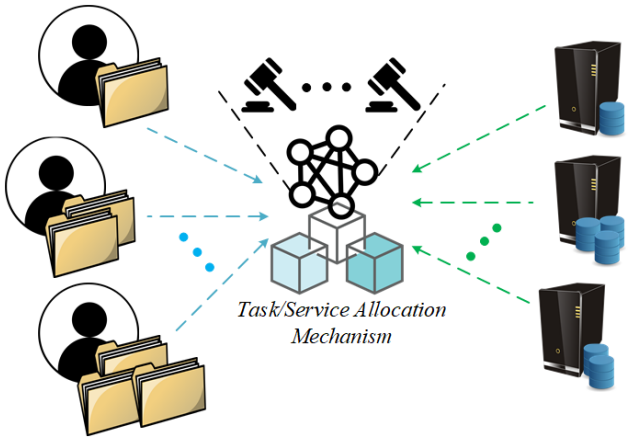

Fig. 2: Task/Service allocation mechanism on top of the blockchain

Another approach is to use a sealed-bid auction to determine the winner. Sealed-bid auction improves the client's utility when the bidders are strategic agents. This is due to the fact that, in the sealed-bid auction the bidders bid based on their true valuations instead of bidding according to the current auction status. This helps clients to find better deals for their tasks. However, such a sealed-bid auction can drastically decrease the overall efficiency of the smart contract, as a larger number of servers trigger the bidding function of the smart contract. Such bidders would not participate in the auction if they aware of the current best offer as in the open auction. However, in the worst case, the complexity of such a sealed-bid auction to find the best offer among the submitted offers is in the order of $O(N)$ as well.

Although auction-based approach solves the problem of computation cost as there is no need to calculate the optimal allocations, it is not an efficient approach in our system model for two main reasons:

- Most of the storage providers in our system model are retail users with limited resources who cannot individually satisfy most of the tasks. For example, a user can offer 10 GB of storage between 10:00 am - 05:00 pm for a month. As most storage requests necessitate uninterrupted availability in the long term, the auction model eliminates such retail users from the system
- The smart contract deployment of auction is costly as Network needs to reach an agreement through a consensus method. Although the deployment overhead cost of smart contract based auction system is low for small number of tasks, such an overhead cost grows as the number of tasks increases.

To allow the retail storers to join to the market, we propose task splitting and pooling solutions.

B. Task Splitting and Pooling

In task splitting, the smart contract can partition coarse-grained tasks to a set of pre-defined fine-grained sub-tasks. For example, if a task is requesting 5 TB of storage, this can be split into 500 subtasks of 10 GB each. This allows small retailer servers to be able to participate in the storage market. This approach is helpful but it is not an ideal model for time-limited resources where storage providers can share their storage services in specific times of a day. For example, consider that a storer is offering storage resources between 08:00 am and 05:00 pm GMT and another storer is providing the storage from 05:00 pm and 08:00 am GMT. Assume each these storers asks for \$1, while a third storer can provide whole day coverage for \$3. Splitting the task time in this example would prefer two short-term storers over the third storer as their total cost would be lower. On the other hand, time splitting could mistakenly favor short-term storers over long term ones when the splitting approach is used for time attribute. To explain this problem, consider the previous example where a storer price is \$3 for a whole day coverage, and this time the first two storers with partial time coverage asks for \$2 each. As there are two auctions for two time-windows, the coalition of two servers will be selected as the winners, and the client would pay \$4, while the \$3 is the better option. Note that as storage servers need to keep the data whole day even if they only commit to a partial time of the day as moving data in and out would be extremely inefficient in most cases. Thus, if a server can provide a whole day service, it is more efficient to dedicate one task to such a server instead of several tasks with various time requirements. This problem is not considered in splitting tasks option which causes losing resources as servers try to compete for fine-grained tasks. Moreover, the computation cost of auction exacerbates with task splitting as more auctions are generated when large tasks are split into many smaller tasks. To handle these problems, pooling approach can be applied [9]–[11].

Pooling allows fine-grained service providers to build coalitions of coarse-grained resource providers. The same approach can be used for tasks such that fine-grained tasks can be combined into a pool to construct coarse-grained tasks. Once the pools of servers and clients have been established, then server pools can compete for coarse-grained tasks. The main benefit here is to outsource the heavy computation of efficient task/service mapping to off-chain pools while keeping the distributed storage network management system on-chain with the smart contract. The cost of smart contract deployment decreases with the rate of pooling as there are smaller number of tasks and storer coalitions. Moreover, managing the users will be easier for pools as the number of users is shrinking compared to a universal approach. In this case, pools are bidding for the tasks considering their available resources. On the other hand, users join the pools based on their demand.

IV. Retrieval Market

In this section, we study the retrieval market and the benefit of the cooperative approach compared to the independent behavior of the nodes in terms of responding to the requests. Recall that, in the retrieval market, clients issue inquiries for the public data from network, and retrievers reply to these requests in return for payments.

For simplicity and without loss of generality, assume the size and the reward value of the files are the same, and each retriever stores one file. Following the system model, upon a successful response to a client, the corresponding retriever is rewarded. Assume the benefit is one unit (e.g. one Filecoin) and if multiple retrievers have the queried file, then they share the benefit. Then a retriever's expected utility can be computed as:

$$E[U_{s_i}] = P_{i,f} \times \bar{P}_{R-i,f} + P_{i,f} \times \sum_{\forall J \in S, |J|>0, i \notin J} P_{J,f} \times \bar{P}_{R-J,f}/|J|$$

Here, $P_{i,f}$ is the probability that the retriever server s_i has the requested file f, and $\bar{P}_{i,f}$ is the probability that the file f is not hosted by s_i. R is the number of retrievers. Note that, when multiple servers host a requested file, they share the benefit. Here, J represents a set of retrievers excluding i, having the requested file f.

Let F represent the number of available files to be stored and queried in the system, and assume the probability of a requested file is uniformly random among the available files, and retrievers are randomly choosing the files to host in their systems. In this model, we can represent the expected utility of a retriever s_i as follows:

$$E[U_{S_i}] = (\frac{1}{F}) * (\frac{F-1}{F})^{R-1} + (\frac{1}{F}) * (\frac{1}{F}) * (\frac{F-1}{F})^{R-2}/2 + ... + (\frac{1}{F}) * (\frac{1}{F})^{R-1}/R = (\frac{1}{F}) * \sum_{k=0}^{R-1} \frac{(\frac{1}{F})^k * (\frac{F-1}{F})^{R-k-1}}{k+1}$$

In order to study the benefits of the pooling approach in retrieval market, we study the price of anarchy in the next section.

A. Price of Anarchy

Price of Anarchy (PoA) measures the degradation of system efficiency when the players behaving selfishly and it is calculated as the ratio between the socially optimal point and the Nash Equilibrium. We use PoA to demonstrate the benefit of joining a coalition for retrieval market. To this end we define PoA as follows:

$$PoA = \frac{E[U_{\bar{S}_R}]}{\sum_{i \in R} E[U_{S_i}]}$$

Here $\bar{S}_R$ represents a pool of R retriever, and we measure the ratio of expected utility for a pool of users and the sum of individual expected utilities without pooling.

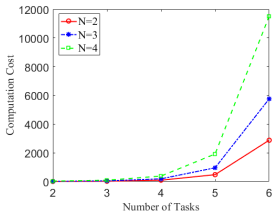

Fig. 3: Computation Cost when Number of Tasks varies

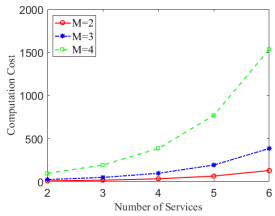

Fig. 4: Computation Cost when Number of Services varies

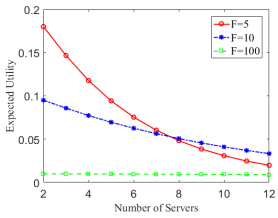

Fig. 5: Expected Utility with the change of Number of Servers

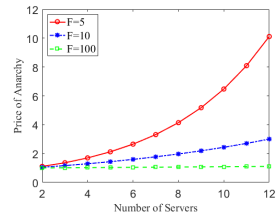

Fig. 6: Price of Anarchy with the change of Number of Servers

V. Evaluation

In this section, we evaluate the auction based task/service allocation in the storage market. We also study the benefits of pooling in storage and retrieval markets.

To evaluate the cost of the auction based task/service allocation, we have used *solidity* language to implement the auction smart contract. We used *remix* to compile the smart contract code. For the sealed-bid auction, the auction implementation includes three phases. The first phase is the setup of an auction by the task's owner. In this phase, the auction parameters are adjusted. These parameters are the time window for bidding, the task requirements, and the maximum value which a task owner would pay for a task. The next phase is bidding, where servers submit their bids in an encrypted format using the commitment scheme. Here, the bid value is hidden while the bidder cannot claim another bid in the revealing phase [12]. Moreover, the smart contract checks that if the bidder has sufficient resources to satisfy the task requirements. Finally, in the revealing phase, the bidders open their submitted commitments, and the smart contract selects the winner which has the lowest bid value. For the open-price auction, bidders submit the plain-text of their bid values while monitoring the current winner's bid. Upon submitting a new bid value, the smart contract first checks if the bid value is smaller than the current winner's bid, then update the winner's bid to the new bid value. We use Etherium to implement the smart contract.

The ether price in 05/22/2019 is $251.93 (https://www.coinbase.com/), and the average gas cost is 18 Gwei (https://ethgasstation.info/) where 1 ether is 10^9 Gwei. Table I demonstrates the cost of deployment, bidding, and revealing operations for a sealed-bid auction, and Table II shows the cost of deployment, and bidding for the open price auction. As it is explained in section III-A, it is expected that as more bidders participate in a sealed-bid auction, the deployment overhead cost increases. In the pooling based approach, several tasks merged to a single auction decreasing the cost of auction deployment and bidding as the costs of the auction deployment and bidding are shared between a set of pool members. Therefore, the implementation of a storage task auction is a practical solution.

TABLE I: Gas cost for a sealed-bid auction

Function	Gas units	Gas cost (USD)
Setup	291351	1.13
Bidding	25728	0.11
Revealing	32293	0.14

TABLE II: Gas cost for an open auction

Function	Gas units	Gas cost (USD)
Setup	215862	0.97
Bidding	35481	0.16

For the storage market, the changes in computation cost in terms of the number of operations for computing the optimal task/service allocation are shown in Figures 3 and 4. We can see that with the growth of the number of tasks and services, the computation cost is growing exponentially, however by pooling we are able to decrease the number of tasks and services which can effectively impact the computation cost. It is worth to mention that, the pooling on the tasks side has more benefit than pooling on the service side.

On the other hand, in order to calculate the benefits of pooling in the retrieval market, we use the price of anarchy concept as introduced in the previous section. Assuming the probability of a request file is uniformly random among the available files, and storers are randomly choosing the files to host in their systems, and the reward of a file hit is one unit which is shareable if multiple stores host the file, then the changes of expected utility and price of anarchy are shown in Figures 5 and 6 when the number of servers varies. We can see that with the growth of the number of servers, the expected utility decreases, and the price of anarchy increases with an increasing rate. On the other hand, with the growth of the number of files in the system, the rate of decrease in expected utility and the rate of increase in the price of anarchy decrease. This means that with the growth of the number of files in such a market, the benefit of pooling is decreasing.

VI. Related Work

Peer-to-peer sharing systems such as BitTorrent were proposed long time ago. However, lack of appealing incentivization mechanisms is a major hurdle in wide adoption as users are reluctant to share their own resources with others. BitTorent [13] uses a tit-for-tat reciprocity strategy to provide positive incentives for nodes which are contributing resources. Recently, several distributed storage networks such as Filecoin [3], swarm [4], MaidSafe [14], sia [6], and storj [5] have been introduced which rely on blockchain technology for the incentivization purpose.

For example, Filecoin [3] is a decentralized storage network that turns cloud storage into an algorithmic market. The market

runs on a blockchain with a native protocol token (also called "Filecoin"), which miners earn by providing storage to clients. Conversely, clients spend Filecoin hiring miners to store or distribute data. As with Bitcoin, Filecoin miners compete to mine blocks with sizable rewards, but Filecoin mining power is proportional to active storage, which directly provides a useful service to clients (unlike Bitcoin mining, whose usefulness is limited to maintaining blockchain consensus). This creates a powerful incentive for miners to amass as much storage as they can, and rent it out to clients. The protocol weaves these amassed resources into a self-healing storage network that anybody in the world can rely on. The network achieves robustness by replicating and dispersing content, while automatically detecting and repairing replica failures. Clients can select replication parameters to protect against different threat models. Such a storage network also provides security, as the content is encrypted end-to-end at the client, while storage providers do not have access to decryption keys. Filecoin works as an incentive layer on top of IPFS [15], which can provide storage infrastructure for any data. It is especially useful for decentralizing data, building and running distributed applications, and implementing smart contracts.

As another example, Swarm [4] is a distributed storage platform and content distribution service. The primary objective of Swarm is to provide a decentralized and redundant store of Ethereum's public record, in particular, to store and distribute decentralized applications' code and data as well as blockchain data. From an economic point of view, it allows participants to efficiently pool their storage and bandwidth resources in order to provide these services to all participants. The goal is a peer-to-peer serverless hosting, storage and serving solution that is DDoS-resistant, has zero-downtime, is fault-tolerant and censorship-resistant as well as self-sustaining due to a built-in incentive system. On the other hand, many research studies such as [7], [16]–[18] are focused on developing an efficient proof of storage and proof of retrievability to verify the honest behavior of the storage providers. In the case of dishonest behavior of the servers, the smart contract can withdraw the server's collateral and pay to the corresponding clients.

In contrast with these works, in this paper, we formalized the problem of optimal task allocation in the decentralized storage network. Then, we proposed a coalitional auction approach for storage market to minimize the cost of smart contract computation cost while exploiting the off-chain computations to find the proper task/service allocation. Moreover, we studied the benefit of pooling approach in retrieval market. To the best of the authors knowledge, this is the first research to investigate the task/service allocation in the storage and retrieval markets for the blockchain based decentralized storage network.

VII. Conclusion and Future Work

Considering the storage market and retrieval market in the blockchain based distributed storage network, we have studied the challenge of task/service allocation. First, we have formally modeled the problem, and then we have proposed different approaches to decrease the cost of smart contract deployment for an efficient task/service mapping. Moreover, we have modeled the retrievers' expected utility and the price of anarchy in the retrieval market. We studied the benefit of pooling in retrieval market as well. The evaluation results show the benefit of the pooling based approach in the storage and retrieval markets. More specifically, by applying pooling in the storage market, the computation cost in the smart contract decreases significantly by outsourcing computations to the off-chain entities. On the other hand, in the retrieval market, the benefit of pooling increases as the number of files and retrievers decrease. As a future work, we plan to develop of new smart contracts which can handle a pooling process life cycle, which includes creation, joining, bidding, verification, nodes management, profit sharing and etc.

Acknowledgement

The work in this study was supported in part by the NSF grant OAC-1850353.

References

[1] M. Rosenberg, N. Confessore, and C. Cadwalladr, "How trump consultants exploited the facebook data of millions," the New York Times. Archived from the original on March 17, 2018.
[2] V. Buterin, "A next-generation smart contract and decentralized application platform," *white paper*, 2014.
[3] Protocol-Labs, "Filecoin: A dicentralized storage networks." [Online]. Available: https://filecoin.io/filecoin.pdf
[4] V. Tron, A. Fischer, D. N. A, Z. Felfldi, and N. Johnson, "swap, swear and swindle: incentive system for swarm," Ethersphere, Tech. Rep., 2016, ethersphere Orange Papers 1. [Online]. Available: https://ethersphere.github.io/swarm-home/ethersphere/orange-papers/1/sw%5E3.pdf
[5] Sia, https://sia.tech/.
[6] Storj, https://storj.io/.
[7] B. Fisch, J. Bonneau, N. Greco, and J. Benet, "Scaling proof-of-replication for filecoin mining," Technical report, Stanford University, 2018. https://web. stanford. edu , Tech. Rep., 2018.
[8] H. Kellerer, U. Pferschy, and D. Pisinger, "Multidimensional knapsack problems," in *Knapsack problems*. Springer, 2004, pp. 235–283.
[9] M. M. Khalili, X. Zhang, and M. Liu, "Public good provision games on networks with resource pooling," in *Network Games, Control, and Optimization*. Springer, 2019, pp. 271–287.
[10] I. Vakilinia and S. Sengupta, "A coalitional cyber-insurance framework for a common platform," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 6, pp. 1526–1538, 2018.
[11] M. M. Khalili, X. Zhang, and M. Liu, "Incentivizing effort in interdependent security games using resource pooling," in *Proceedings of the 14th Workshop on the Economics of Networks, Systems and Computation*. ACM, 2019, p. 5.
[12] I. Vakilinia, S. Badsha, and S. Sengupta, "Crowdfunding the insurance of a cyber-product using blockchain," in *Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), 2018 IEEE 9th Annual*. IEEE, 2018.
[13] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "Do incentives build robustness in bittorrent," in *Proc. of NSDI*, vol. 7, 2007.
[14] "Maidsafe." [Online]. Available: https://maidsafe.net/
[15] IPFS, "Interplanetary file system," http://ipfs.io/.
[16] A. Juels and B. S. Kaliski Jr, "Pors: Proofs of retrievability for large files," in *Proceedings of the 14th ACM conference on Computer and communications security*. Acm, 2007, pp. 584–597.
[17] S. Dziembowski, S. Faust, V. Kolmogorov, and K. Pietrzak, "Proofs of space," in *Annual Cryptology Conference*. Springer, 2015, pp. 585–605.
[18] H. Shacham and B. Waters, "Compact proofs of retrievability," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2008, pp. 90–107.