DAGBENCH: A Performance Evaluation Framework for DAG Distributed Ledgers

Andrew Dong[1], Emma Zheng[1], Young Choon Lee[2] and Albert Y. Zomaya[3]

***Abstract*—Directed Acyclic Graph (DAG) has been emerging as the so-called Blockchain 3.0 after Bitcoin (Blockchain 1.0) and Ethereum (Blockchain 2.0). This new distributed ledger technology is getting significant attention for its high performance and low transaction fee. There have already been several notable implementations, such as IOTA [1], Nano [2] and Byteball [3]. In this paper, we present DAGBENCH as a performance evaluation framework for DAG implementations. DAGBENCH provides a number of sample workloads and adaptors that make effective and easy evaluation of different DAG implementations.It allows any DAG implementation to be evaluated by adding an adaptor. DAGBENCH allows to measure the performance of DAG implementation in terms of throughput, latency, scalability, success indicator, resource consumption, transaction data size and transaction fee. We demonstrate how DAGBENCH can be used to measure the performance of different DAG implementations. In particular, we have conducted experiments, on Amazon EC2, with three popular DAG implementations: IOTA, Nano and Byteball. Our experimental results provide the performance comparison between these implementations that helps developers/users to effectively evaluate different performance characteristics; and, this enables them to identify bottlenecks and accordingly improve performance.**

***Index Terms*—Directed Acyclic Graph, Blockchain, Smart contracts, Distributed ledgers, Benchmark**

I. Introduction

Blockchain technologies have gained popularity for its decentralization nature. Bitcoin [4] and Ethereum [5] are two representative blockchains that have primarily contributed to such popularity gain.These technologies have significantly enabled several secure decentralized applications, such as crypto-currency and smart contracts. However, the current block-based data structure brings a number of issues, such as poor performance and scalability, slow final confirmation and centralization of powerful miners. To resolve these issues, the Directed Acyclic Graph (DAG, see Figure 1) data structure has been emerged and it has been increasingly adopted in the field of Distributed Ledger Technology (DLT) [6].

DAG is a graph data structure that has emerged as a new form of DLT after the popular blockchain technology [7]. It consists of individual transactions linked and formed as a directed acyclic graph, without using the concept of blocks as in blockchains. In particular, DAG removes block packaging process resulting in resolving the well-known block creation bottleneck in, for example, Bitcoin and Ethereum. With such advantages of DAG, several notable DAG implementations have emerged, such as IOTA [1], Nano [2] and Byteball [3]. However, currently, there is a lack of performance evaluation tools for DAG, more precisely, its implementations.

In this paper, we develop and present DAGBENCH (Figure 2) as a performance evaluation framework for DAG implementations. DAGBENCH is a flexible and extensible framework that provides a number of sample workloads and adaptors for different DAG implementations. While the current version of DAGBENCH has supports (i.e., adaptors) for three popular DAG implementations (IOTA, Nano and Byteball), such support can be easily extended for other implementations by adding corresponding adaptors. DAGBENCH uses a number of performance metrics for the evaluation: throughput, latency, scalability, success indicator, resource consumption, transaction data size and transaction fee. We have implemented two workloads: value transfer and transaction query. New workloads can be added by implementing them in the workload layer of DAGBENCH (see Figure 2). To demonstrate the capacity of DAGBENCH, we have conducted an in-depth comparison study of the three DAG implementations with two workloads.

Specific contributions of this paper are as follows:

- We design and implement a performance evaluation framework for DAG distributed ledgers.
- We have released the framework for public use. [1]
- We evaluate DAGBENCH on a real system (Amazon EC2) with three representative DAG implementations: IOTA, Nano and Byteball.
- We give comparative analysis between the three DAG implementations in various performance metrics.

The rest of this paper is organized as follows. In Section II, we introduce DAG distributed ledger and three DAG implementations. In Section III, we present the design and implementation of DAGBENCH. Section IV presents the experimental results and observations. In Section V, we compare and discuss the advantages and disadvantages of the three DAG implementations. We discuss related work in Section VI followed by our conclusion in Section VII.

II. Background

In this section, we give a brief introduction of DAG distributed ledger followed by the description of three promising

[1] Andrew Dong (andrew@tbsx3.com) and Emma Zheng (emma@tbsx3.com) are with TBSx3 Blockchain Lab, L5, 155 Clarence Street, Sydney NSW 2000, Australia.

[2] Young Choon Lee (young.lee@mq.edu.au) is with Department of Computing, Macquarie University, Sydney NSW 2109, Australia.

[3] Albert Y. Zomaya is with Center for Distributed and High Performance Computing, School of Computer Science, The University of Sydney, NSW 2006, Australia.

[1] https://github.com/TBSx3Australia/DAGBENCH

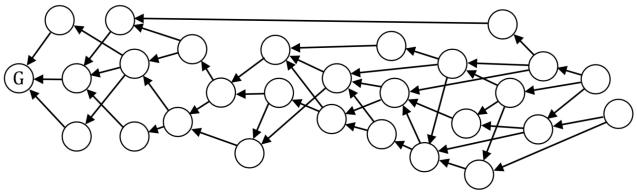


Fig. 1. DAG Structure. G refers the first/genesis transaction of a DAG as in the genesis block of a blockchain.

DAG implementations, IOTA, Nano and Byteball.

Directed Acyclic Graph (DAG) is getting its momentum with its advantages particularly of scalability and resource efficiency over blockchains. The main difference between blockchains and DAGs is that the former bundles transactions in blocks and append them one after another while the latter stores transactions as vertices of a graph [7]. In other words, DAGs are *blockless*.

Transactions can be appended onto a DAG concurrently. In particular, one transaction can have multiple predecessors and successors. The DAG, therefore, progresses transactions immediately and directly without waiting for block composition. It also can add multiple transactions simultaneously to increase system throughput and scalability.

In DAGs, every end user issuing a transaction also validates previous transactions. DAGs remove the need for miners as in block-based distributed ledgers (e.g., Bitcoin and Ethereum) that consume huge amounts of resources, both of computing and energy.

1) IOTA: IOTA is a value transfer platform designed for Internet of Things (IoT). The underlying network is called 'Tangle', which is based on DAG [1]. In IOTA, every participant/node in the network (more generally, DAG network) has to approve two previous transactions in order to issue a new transaction. The two transactions are selected through a tip (a childless unit, similar to a leaf node[2] in the traditional graph/tree data structure) selection algorithm.

While smart contract is not supported by IOTA in its current implementation, roughly 1 kilobyte of arbitrary data can be included in the transaction, which can be utilized in scenarios such as transferring IoT data between devices.

Currently, IOTA introduces 'Coordinator' to confirm transactions. The Coordinator is an entity controlled by the IOTA foundation, which issues zero-value transactions regularly, called a Milestone. Using the Coordinator, consensus is achieved through a rule that only the transaction referenced by a Milestone is confirmed, and the others are not.

2) Nano: Nano is a value transfer platform based on an architecture called 'block-lattice' [8]. With this structure, Nano is not just one long chain like Bitcoin or IOTA; it is a multi-chain structure where each user gets their own chain that only they can append onto. There are two types of transaction: 'send' and 'receive'. The sender needs to create a 'send' transaction and the receiver needs to issue a 'receive' transaction to actually pocket that amount of money. When any conflict occurs, 'representatives' will vote for the conflicts and pick one authentic transaction.

Nano is famous for its high speed, but the transaction object is quite simple. Any additional message is not allowed along with the transaction. Smart contract is not supported, either.

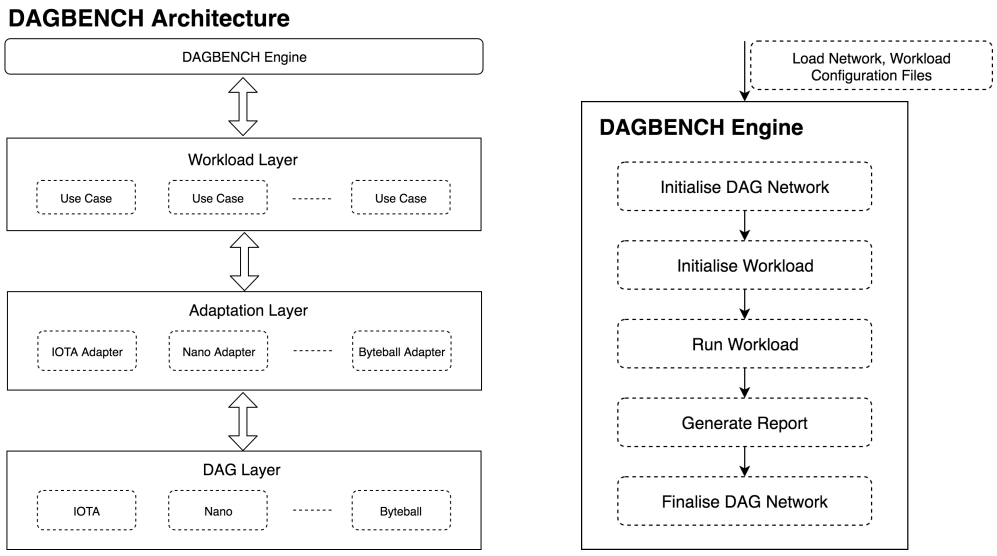
3) Byteball: Byteball is a conditional smart payment platform that enables human readable smart contracts. Data is stored and ordered using DAG, a new transaction has to reference earlier data units created by other users [3]. Different from IOTA's referencing of two units, Byteball encourages referencing all tip transactions by paying more rewards, which helps narrow the width of the network. In this way, users need to pay transaction fee which is used for rewarding later users who reference the unit. The amount of transaction fee is decided by the amount of data the unit includes: 1 Byte (the Byteball's built-in cyptocurrency) is exactly the amount you pay for storing 1 byte of data in the ledger, meaning that the more data you want to post, the more money it costs.

The units in Byteball network have total order, called main chain. To build a main chain, Byteball creates an algorithm that selects one parent of a unit as the 'best parent'. We can select one main chain starting from any tip (a childless unit) of the DAG, then travel backwards in history along the best parent links. If we start from another tip, we will build another main chain. If those main chains ever intersect while they go back in history, they will both go along the same path after the intersection point. This point is called 'stable point' and all the units on this same path are deemed as stable units, meaning that they are confirmed. Once we have a main chain, nonserial units can also have total order.

Since parent units might be created by an attacker, Byteball introduces 'Witnesses' to periodically issue trusted transactions to DAG. Witnesses are non-anonymous reputable people or companies who are expected to behave honestly. When selecting the best parent, the parent unit whose main chain can yield greater witnesses level is considered as the best one. Each user can select different Witnesses list based on their knowledge, but Byteball rules that best parents must be selected only among those parents whose witness list differs from the childs Witness list by no more than one mutation. The Witness is a trusted checkpoint which is similar to IOTA's coordinator, but IOTA only has 1 Coordinator and Byteball has 12 witnesses.

In Byteball, an address represents a definition, which is a boolean expression. When a user signs a unit, s/he also provides a set of authentifiers (usually signatures) which, when applied to the definition, must evaluate it to true in order to prove that this user has the right to sign this unit. Definitions are written in JSON. With the definition, Byteball enables applications such as conditional payment, Peer-to-Peer (P2P) insurance and betting. Byteball supports the publication of

[2]Note that a node in a DAG should not be confused with a 'node' in a DAG network. While the former is a vertex of a DAG data structure, the latter is a compute node (or a server like an Amazon EC2 instance) that is a part of a DAG network.



(a) DAGBENCH architecture. (b) DAGBENCH engine.

Fig. 2. DAGBENCH.

personal assets, which can be used as an Initial Coin Offering (ICO [9]) platform.

III. DAGBENCH

DAGBENCH (Figure 2) is a performance evaluation framework for DAG distributed ledgers, which allows users to test different DAG implementations, with predefined workloads, in a wide range of performance metrics.

A. Design and Implementation

Figure 2(a) illustrates the current DAGBENCH architecture. To make all workloads and DAG implementations extensible, DAGBENCH is designed in a modular fashion consisting of a DAGBENCH engine, a DAG layer, an adaptation layer and a workload layer. The actual implementation of DAGBENCH has been realized using Node.js [10].

- DAGBENCH Engine: DAGBENCH engine reads a configuration file and loads the corresponding workloads and adaptors. In particular, it accepts three arguments: *net*,*work* and *env*. These arguments indicate the network, workload and environment for the test. Then an instance of DAG network is initialized and a workload instance creates the test. The detailed steps performed by the DAGBENCH engine are shown in Figure 2(b).
- Workload Layer: Workloads are defined in this layer. The workload class extends the workload interface and is called by the DAGBENCH Engine. The workload layer contains the tests implemented for typical DAG network scenarios. We implement value/data transfer workload and transaction query workload. Developers can write their own test cases via implementing the workload interface.
- Adaptation Layer: The adaptation layer is a bridge to link a DAG network—created by a particular implementation like IOTA, Nano and Byteball—to DAGBENCH workload and engine. A DAG-interface is defined in adaptation layer, a DAG adaptor implements this interface with its corresponding SDK or RESTful API.
- DAG Layer: A DAG network environment is established in the DAG layer. The configuration files for different workloads are to be defined in this layer.

B. Evaluation Metrics

To evaluate the performance of different DAG implementations, we define the following metrics:

- Throughput: It is measured as the number of processed transactions per second.
- Latency: It is measured as the response time per transaction.
- Scalability: As DAG often deals with potentially a large number of *distributed* participants ('nodes' in the context of DAG network), scalability is one of key performance metrics. The scailability of a given DAG implementation is measured as the changes in throughput and latency when increasing the number of nodes.
- Success indicator:
  - Confirmation rate: It is measured as the number of confirmed transactions per second.
  - Reception rate: It is measured as the number of received transactions per second; this metric is specific to Nano.
- Resource consumption:
  - Memory: It is measured as the total memory usage of a full node[3].

[3]A full node in distributed ledgers is a participating node that contains the 'full' history of transactions in contrast to a light node that contains only hash values of transactions.

  - CPU usage: It is measured as the peak CPU usage rate of a full node.
  - Network traffic: It is measured as the peak network-in and network-out of a full node, in terms of the number of bytes.
- Transaction data size: It is measured as the maximum data size that can be included in a transaction.
- Transaction fee: It is measured as the minimum transaction fee that must be paid for a transaction.

C. Workloads

Similar to blockchain, DAG can also be considered as an append-only distributed database. We are focusing on the performance of data creation and data retrieval.

We implement two workloads which can be executed through client-side APIs to evaluate the performance of DAG implementations: value/data transfer and transaction query.

1) Value/data transfer: This workload is a simple application that transfers coins or arbitrary data from one user to another user.

We implement a set of workload-clients (or simply clients) which function as transaction issuers. Clients are sending transactions during the entire course of test, so that the above metrics are evaluated at an exact request rate. After the first 10% of test duration, which is regarded as a period of unprepared phase, the workload instance starts to count the number of processed transactions and the number of confirmed transactions. Test results including statistical information are reported at the end of the test, i.e., the final report.

2) Transaction Query: This workload considers the performance of a DAG network for historical data retrieval. We implement two queries:

Query 1 (Q1): calculate the number of input transactions and the number of output transactions for a given account.

Query 2 (Q2): calculate the balance for a given account.

In this workload, we pre-load the DAG network with different numbers of transactions. We then use APIs to conduct the defined queries and calculate the query latency.

IV. Performance Evaluation

In this section, we present performance evaluation results of three DAG implementations (IOTA, Nano and Byteball) obtained from experiments conducted using DAGBENCH.

A. Experimental environment

The experiments were run on the AWS cloud platform (Amazon EC2). Each node was configured in an independent EC2 instance allocated with 4 vCPUs, 16 GB of memory, 8 GB storage, running Ubuntu Server 16.04 LTS (HVM) and SSD Volume Type. The results are averaged over 5 independent runs.

- For IOTA, we built the testnet environment with its Docker image v1.4.2.4 and used its JavaScript library to implement adaptors.
- For Nano, we built a Docker image in its 'test' model and used its RPC command to implement adaptors.

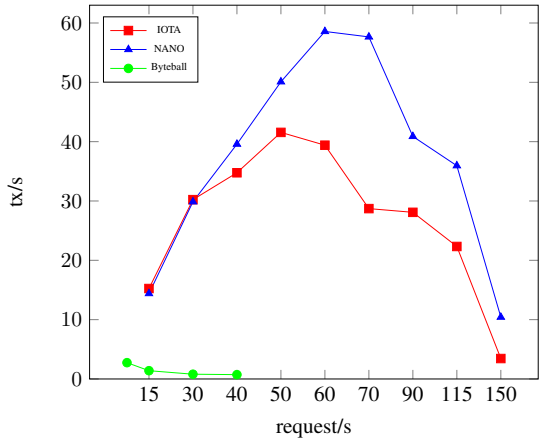


Fig. 3. Throughput with 8 clients and 8 (DAG network) nodes.

- For Byteball, we built the testnet environment with its official git repository for 'hubs' and 'witness', and used its 'byteball.js' to implement adaptors.

B. Results: Value/data transfer

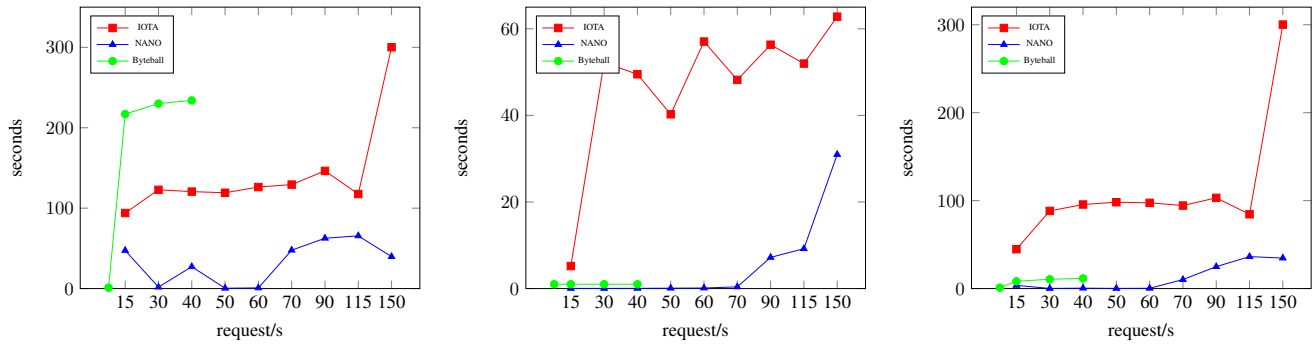
This section presents the test results of our first workload: value/data transfer for three implementations. We measured the performance of the DAG network with 8 nodes (i.e., 8 EC2 servers in our study) and 8 concurrent clients over the period of 300 seconds. Each client sends transactions with a request rate varying from 5 request/second (req/s) to 150 req/s.

1) Throughput and Latency: As shown in Figures 3 and 4, the throughput and latency become in a poorer quality (lower and higher, respectively) after a request rate of 60req/s. In particular, the peak throughput of IOTA—40 transactions/second (tx/s)—is reached at a request rate of 60 req/s. The decreased throughput under high loads is because that: (1) the high load leads to a HTTP TIMEOUT error, which decreases the actual request rate and (2) a high request rate increases the number of new units generated concurrently; it also increases the tip selection delay.

For Nano, the trend shows a similarity with IOTA. Nano reaches 60 tx/s at peak where the request rate is 60 req/s. The throughput is higher and the latency is lower than that of IOTA. The gap between IOTA and Nano is because that while IOTA has a tip selection delay to select two parents for every new transaction, Nano has no tip selection delay due to its block-lattice structure [2].

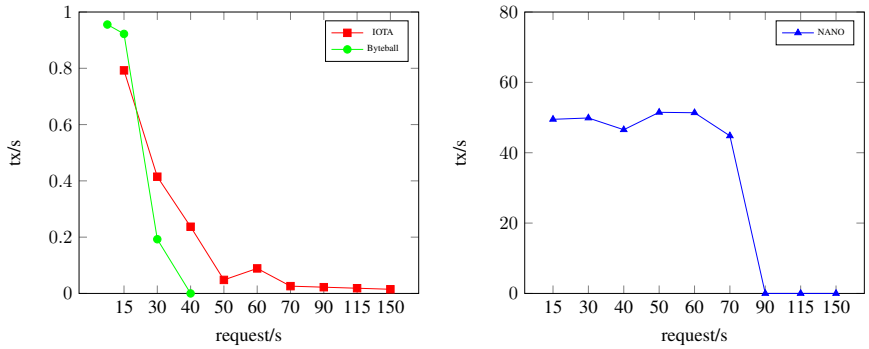
For Byteball, the throughput is much smaller than both Nano and IOTA. For the testnet with 8 full nodes, the peak throughput is around 2.75 tx/s when the request rate is 5 req/s. When the request rate is larger than 40 req/s, the DAG network does not respond. Byteball has lower throughput because local database operations in Byteball implementation increase the transaction processing delay [11].

2) Success indicator: Since there is a big difference between how different DAG implementations confirm a transaction, we use different success indicators for different imple-



(a) Maximum latency with varying request rates. (b) Minimum latency with varying request rates. (c) Average latency with varying request rates.

Fig. 4. Latency with 8 clients and 8 nodes.



(a) Confirmed transaction rates. (b) Reception rates.

Fig. 5. Success indicator with 8 clients and 8 nodes.

mentations to reflect successful transactions. Figure 5 shows the success indicators for different implementations.

For IOTA and Byteball, we use 'confirmation rate' to represent the number of transactions that are confirmed in a second. For Nano, a transaction is confirmed when the recipient's balance increases with an amount of money carried by this transaction. In our experiments, the recipient issues 'receive' transactions after all the transactions are sent to the recipient. We then measured this duration for receiving all the local transactions and used 'reception rate' to represent the number of transactions that can be received in a second.

IOTA introduces *Coordinator* to periodically issue zero-valued transactions and confirm previous transactions. In our experiments, we ran the coordinator every 60 seconds. The result (Figure 5(b)) shows that confirmation rate decreases with the growing request rate. This is because of the increasing latency of the DAG network. The coordinator's transactions take longer delay to be attached by the DAG network. The result also shows that the peak confirmation rate is around 0.8 tx/s, which is much lower than IOTA's throughput. The low confirmation rate is because that not all transactions can be referenced and confirmed by the coordinator. The transactions that are not referenced by any other transactions will remain orphaned as the DAG network grows.

For Nano, the reception rate remains stable around 50 tx/s with request rates below 70 req/s. After 70 req/s, around 18,000 transactions are received by the recipient. Receiving these transactions via APIs leads to a TIMEOUT error.

For Byteball, witnesses issue transactions periodically to validate previous transactions. The maximum confirmation rate is around 0.95 tx/s when the request rate is 5 req/s. This low confirmation rate is because of its low throughput as stated in subsection IV-B1.

3) CPU, memory and network flow: Figure 6 shows the memory usage, CPU usage and network flow. For IOTA, the peaks of memory, CPU and network are in accordance with the results of throughput and latency as discussed in subsection IV-B1. It is shown that when the request rate is higher than the saturated point, the number of transactions that the DAG network can receive decreases. The peak memory usage for a single IOTA full node is around 2,600 MB, the peak CPU usage is around 68% and the peak network usage (for both network-in and network-out) is about 123 MB/s.

For Nano, it consumes less resources compared with IOTA. There is a slight increase for all types of resources with the growing of request rate. The peak memory usage for single Nano full node is around 150 MB, the peak CPU usage is around 12% and the peak network usage is about 23 MB/s .

For Byteball, when request rate is 5 tx/s, the memory it consumed is around 367 MB, the peak CPU usage is around 6% and the peak network usage is about 4.9 MB/s. While Byteball has the lowest throughput, the memory is still higher

than Nano, which shows that the size of Byteball's transaction object is larger than that of Nano.

4) Scalability: For IOTA and Nano, we applied 60 req/s in the experiments while we used 5 req/s for Byteball due to its low throughput. Throughput, success indicators and latency are compared in Figure 7.

For IOTA, throughput increases with growing node count. DAG distributed ledgers have a feature that nodes process the transactions asynchronously, which means that each node can issue transactions concurrently and then broadcast the transactions to other nodes in order to reach consensus. This increases the total throughput of the whole network, and increases the scalability. Confirmation rate, on the other hand, seems no increase with respect to the varying numbers of nodes. Latency decreases from the 2-node (DAG) network to the 4-node network and remains almost stable for the larger networks, which means the latency is at least 100 seconds.

For Nano, the overall throughput for all networks is higher than that of IOTA. There is a slight rise when the number of nodes increases. The result (Figure 7(f)) shows that the reception rate remains stable with increasing nodes since the 'receive' operation is only conducted on a single node.

For Byteball, the throughput increases with growing node count since higher number of nodes can process transactions asynchronously. The confirmation rate increases from the 2-node network to the 10-node network, but remains stable from the 10-node network to the 12-node network. This is because transactions are issued at a fixed rate by the fixed number of witnesses; this leads to a stable confirmation rate.

5) Transaction Fee: IOTA and Nano both have zero transaction fee. The transaction fee of Byteball is based on the size of transaction object, i.e., the transaction issuer needs to pay one Byte (the initial coin of Byteball) for one byte of data.

6) Transaction data size: IOTA can transfer around 1300 Bytes arbitrary data with each transaction object. For Nano, there is no additional field to save any arbitrary data. For Byteball, one can store any size of arbitrary data using 'text' message type or 'data' message type (i.e., unlimited). The transaction fee is based on the size of data that users post: the more data you post the more money you pay.

C. Results: Transaction Query

This section presents the test result of our second workload: transaction query. We measured the query latency for two query scenarios in pre-loaded networks with different numbers of transactions from 10 transactions to 10,000 transactions.

1) Q1: The result of Q1 is shown in Figure 8(a). For IOTA, query latency rises with the increase of pre-loaded transactions. After the number of pre-loaded transactions is larger than 300, the DAG network responds with a TIMEOUT error. For Nano, query latency rises with the increase of pre-loaded transactions and is smaller than that of IOTA. For Byteball, query latency rises with the increase of pre-loaded transactions. It responds with a TIMEOUT error when the number of transactions reaches 10,000.

2) Q2: The result of Q2 is shown in Figure 8(b). Overall, the query latency of each of IOTA and Nano remains nearly the same while that of Byteball increases with respect to the number of transactions. In particular, the query latency for all three is below 50ms, except that of Byteball (0.23s) with 10,000 transactions. This exceptional case is due to the fact that Byteball uses a relational database as its data storage; and this causes a serious bottleneck with a high volume of transactions resulting in such an exceptionally high latency.

V. Discussion

All the three implementations, as DAG distributed ledgers remove mining process which is criticised for its computational waste and the centralization of hash power. The removal of mining enables a lower transaction fee. The DAG structure improves scalability and performance by processing transactions asynchronously and concurrently. All the implementations are pre-mined, meaning that their ledgers start out with a certain amount of cryptocurrency that can never change. While they have different advantages and scenarios, these advantages also lead to new issues.

A. IOTA

Advantages and scenario. (1) In IOTA, money and data can be sent together with a transaction, this is good for machine-to-machine scenario which ranges from data market to supply chain tracking [12]. (2) IOTA has a mechanism to trim ledger size in a process called 'snapshot', this can reduce the ledger size as the network grows.

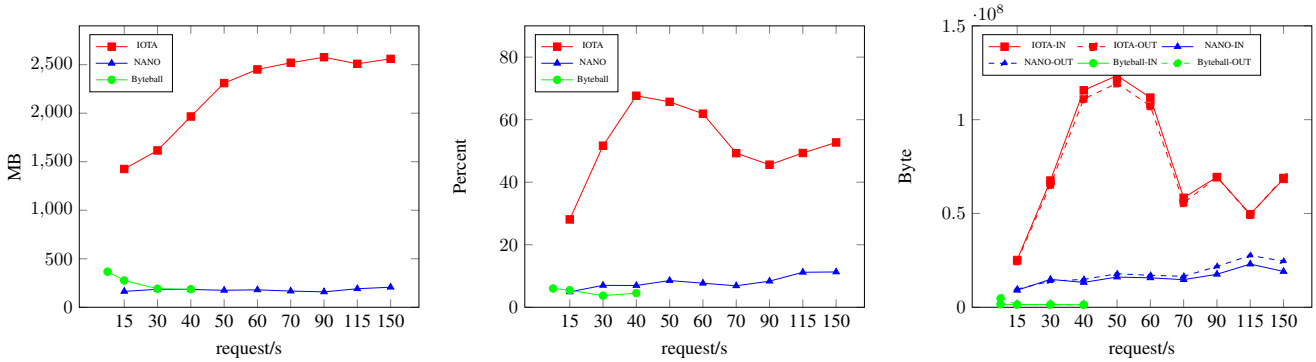
Disadvantages and current issues. (1) Computational and memory resource consumption is the highest among all the three implementations. (2) In theory, the more transactions over the network, the faster transactions get confirmed, but more transactions cause congestion; this leads to delays in confirmation rates and times [12]. (3) Valuable transactions may become orphan when there are a large amount of transactions. A reattachment mechanism is not implemented so users need to reattach unconfirmed transactions themselves. (4) Coordinator will not be removed until the network is large enough. (5) A healthy network needs a large number of full nodes, but there is no incentive for running a full node.

Future development. (1) IOTA foundation keeps optimizing the algorithm such as tip selection algorithm. (2) IOTA has developed its own second layer called Flash Channels, but it will need someone to host, maintain, and facilitate this second layer, which introduces fees [12].

B. Nano

Advantages and scenario. (1) The performance result is the best among the three: highest throughput and lowest resource consumption. (2) The fast speed, ease of use and zero transaction fee are good for the scenario of human-to-human value transfer.

Disadvantage and current issues. (1) It cannot transfer arbitrary data. (2) Representative system is regarded as a

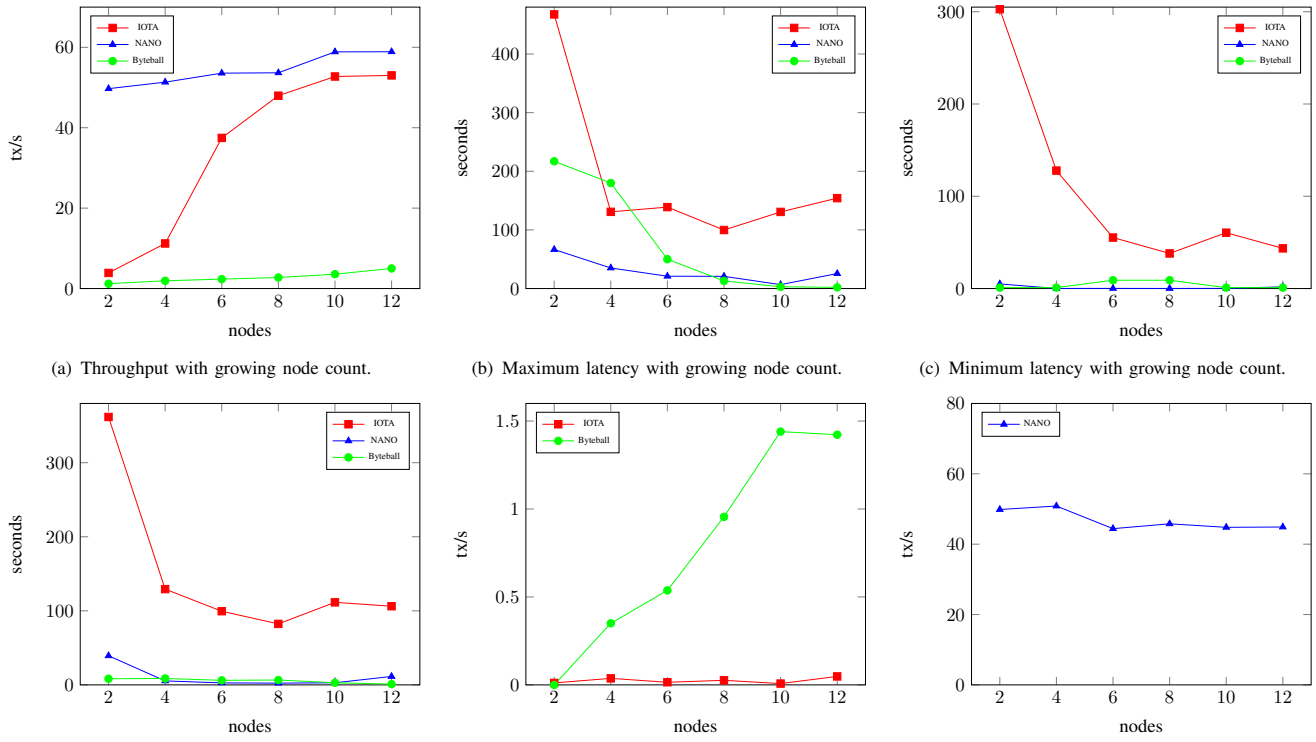


(a) Memory used with varying request rates.

(b) CPU used with varying request rates.

(c) Network in and out with varying request rates.

Fig. 6. CPU, memory and network flow with 8 clients and 8 nodes.



(a) Throughput with growing node count.

(b) Maximum latency with growing node count.

(c) Minimum latency with growing node count.

(d) Average latency with growing node count.

(e) Confirmation rate with growing node count.

(f) Reception rate with growing node count.

Fig. 7. Scalability.

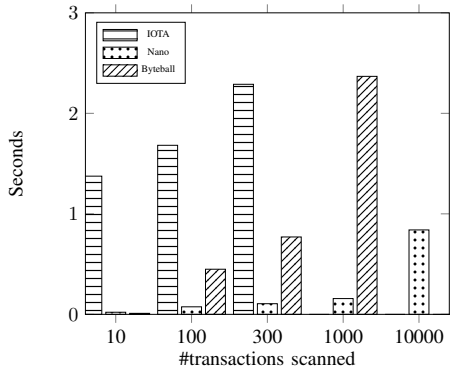
potential attack vector, whereby a malicious entity may buy up millions of dollars and carry out a voting attack [12].

Future development. (1) To reduce the size of database size, ledger pruning has been introduced. It can be done via pruning off old blocks and keeping only the frontier 1 or 2 previous for each account. (2) Instead of using current 4 types of block (i.e., send, receive, change and open), it has proposed the feature to move these 4 types to "Universal Blocks". It will make the programming interface simpler, improve internal data operation and reduce the processing requirement to facilitate light or hardware wallets.

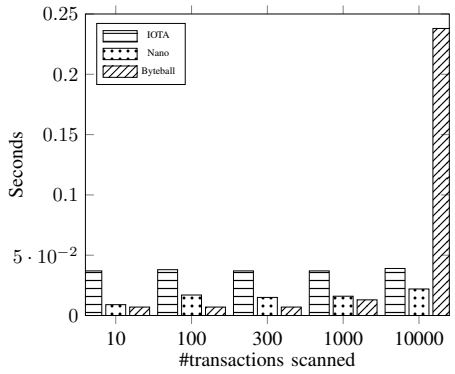
C. *Byteball*

Advantages and scenario. (1) Byteball enables a smart contract which can be used in scenarios such as conditional payment and P2P betting. (2) Byteball offers an untraceable currency called Blackbytes whose transactions are not visible on the public database; these transactions are sent peer-to-peer instead. (3) Byteball enables customized assets; hence, it is also an ICO platform.

Disadvantage and current issues. (1) The performance of Byteball is the worst among the three implementations. (2) There is a low incentive for running full nodes and becoming a witness. As a result, currently most witness nodes are held by the founder.



(a) Query workload Q1.



(b) Query workload Q2.

Fig. 8. Latency of query workload.

Future development. The founder has declared that the low performance is because of the database operations in the source code [11]. Code base is to be engineered and optimized.

VI. Related Work

While the focus of blockchain research has been largely on its applications, crypto-currency applications such as Bitcoin and Ethereum, in particular, there have been only a handful of studies on the performance of blockchains, e.g., [13], [14], [15], [16], [17], [18], [19]. Besides, many of them focus on a particular performance respect, such as scalability or security.

BLOCKBENCH [16] is probably the closest to our work. BLOCKBENCH is a framework to evaluate the performance of private blockchains. It supports Ethereum in a private environment, Hyperledger Fabric, Parity and Quorum. It comes with two types of workload, macro benchmark workload and micro benchmark workload. The macro benchmark workloads is to evaluate the overall performance of blockchains which include the YCSB KVStore and SmallBank OLTP task. The micro benchmark workloads for evaluating performance of individual layers including consensus layer, data model layer and execution layer. The micro benchmark can gather the IO usage and CPU usage during the evaluation process.

Hyperledger Caliper [19] is a blockchain benchmark tool which allows users to evaluate different blockchain and generate the testing report. It supports all Hyperledger blockchain product, including Fabric, Sawtooth, Iroha and Burrow. It has been architected to three layers, Adaptation Layer, Interface and Core Layer and Application Layer. It currently supports the following performance indicators: transaction success rate, transaction throughput, transaction latency and computing resource consumption.

To the best of our knowledge, DAGBENCH is the first to evaluate DAG distributed ledgers in a comprehensive manner. Furthermore, its extensibility for various DAG implementations is also what distinguishes DAGBENCH from other performance evaluation approaches/tools.

VII. Conclusion

In this paper, we present DAGBENCH as the first performance evaluation framework for the DAG distributed ledger. DAGBENCH contains creating new transaction and querying history transactions workloads. We have demonstrated the use of DAGBENCH to evaluate the performance of three popular DAG implementations, IOTA, Byteball and Nano. Our evaluation study has shown DAGBENCH's capacity for the comprehensive performance evaluation of different DAG implementations. The performance evaluation results obtained from our experiments have enabled us to make a number of useful observations and effectively identify advantages and disadvantages of the three DAG implementations. In our future work, we plan to add several test tools to simulate community participants to evaluate the financial sustainability in a crowd computing environment and to detect flaws in decentralized security mechanisms.

References

[1] S. Popov, "The tangle. white paper," 2016. [Online]. Available: https://iota.org/IOTA_Whitepaper.pdf

[2] C. LeMahieu, "Nano: A feeless distributed cryptocurrency network," *URl: https://nano. org/en/whitepaper.(accessed: 04.04. 2018)*, 2018.

[3] A. Churyumov, "Byteball: A decentralized system for storage and transfer of value," *URL https://byteball. org/Byteball. pdf*, 2016.

[4] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[5] "Ethereum blockchain app platform." [Online]. Available: https://www.ethereum.org/

[6] G. B. G. O. for Science, *Distributed Ledger Technology: Beyond Block Chain*. Government Office for Science, 2016. [Online]. Available: https://books.google.com.au/books?id=T3MqnQAACAAJ

[7] F. M. Benčić and I. P. Žarko, "Distributed ledger technology: Blockchain compared to directed acyclic graph," *arXiv:1804.10013*, 2018.

[8] C. LeMahieu, "Raiblocks: A feeless distributed cryptocurrency network," *URL https://raiblocks. net/media/RaiBlocks_Whitepaper__English. pdf*, 2017.

[9] L. Ante, P. Sandner, and I. Fiedler, "Blockchain-based ICOs: Pure hype or the dawn of a new era of startup financing?" *Journal of Risk and Financial Management*, vol. 11, no. 4, 2018.

[10] "Node.js: A javascript runtime." [Online]. Available: https://nodejs.org/

[11] A. During, "Existing problems and improvement directions of byteball." [Online]. Available: https://bbfans.org/2018/08/25/byteball-flaws-and-future-direction/

[12] P. Ryszkiewicz, "IOTA vs NANO (RaiBlocks)." [Online]. Available: https://hackernoon.com/iota-vs-raiblocks-413679bb4c3e

[13] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. Gün Sirer, D. Song, and R. Wattenhofer, "On scaling decentralized blockchains," in *Financial Cryptography and Data Security*, J. Clark, S. Meiklejohn, P. Y. Ryan, D. Wallach, M. Brenner, and K. Rohloff, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 106–125.

[14] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-ng: A scalable blockchain protocol," in *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation (NSDI)*, 2016, pp. 45–59.

[15] E. Kokoris-Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, "Enhancing bitcoin security and performance with strong consistency via collective signing," in *Proceedings of the 25th USENIX Security Symposium (SEC)*, 2016, pp. 279–296.

[16] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan, "Blockbench: A framework for analyzing private blockchains," in *Proceedings of the 2017 ACM International Conference on Management of Data*. ACM, 2017, pp. 1085–1100.

[17] B. V. P Thakkar, S Nathan, "Performance benchmarking and optimizing hyperledger fabric blockchain platform," in *Proceedings of 2018 IEEE 26th International Symp. Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2018, pp. 264–276.

[18] "Hyperledger fabric." [Online]. Available: https://www.hyperledger.org/projects/fabric

[19] "Hyperledger caliper." [Online]. Available: https://www.hyperledger.org/projects/caliper