OPEN Chain: Scalability Through Data Parallelization

Abstract

OPEN is designed to be the next generation scalability protocol by utilizing transaction threading through the OPEN Blockchain Load Balancing Protocol and OPEN ORapid Consensus. We present a new technology we call “Scaffolds.” Scaffolds represent an application’s payment schema and translates on-chain data into an OPEN State. When a developer deploys applications on OPEN Chain, they are automatically deployed across multiple chains each with their own consensus mechanisms via the OPEN Cluster. Transactions are initially processed on the OPENChain with our high-throughput ORapid Consensus mechanism. As transaction volume approaches the throughput limit, OPEN utilizes our unique OPEN Blockchain Load Balancing Protocol to route transactions between a number of different blockchains. This allows us to parallelize data processing. The Scaffolds that exist on the respective chains all point the data to the same application state, called an OPEN State. An application database then only has to point towards one source in order to update changes, while still leveraging clusters of blockchains. This paper outlines a unique approach for powerful throughput scalability with application data parallelization across multiple blockchains.

Introduction

Due to Amdahl’s law, even protocols that are considered to have passable levels of security and decentralization suffer from orders of magnitude less throughput in comparison to what exists in systems designed with centralized control. Until blockchains can solve this core problem, mainstream adoption of blockchains will continue to be hampered because the cost from higher latency and reduced throughput can be translated into dollars lost for applications.

Taken as a whole, the most powerful supercomputer in the world is the Bitcoin Network, followed closely by the Ethereum network. Parallel computing enables modern day data processors to work in conjunction with one another - enabling processing speeds far beyond what is capable by the most advanced processor do by themselves due to limits of the single component itself, i.e. Amdahl’s law.

Much like multiple processing units in a single system, or in the perspective of CPUs as multiple cores on one die, the same principle applies to databases, servers and there needs to exist an equivalent in decentralized technologies. Such a technological analogy is especially relevant given the history behind distributed computing.

The motivation that created the first distributed systems was to have more computational power, have more resource sharing in order to increase efficiency, and the rapid growth of network infrastructure.

However one cannot simply take a workload designed for single threaded, serialized computation and toss it into a massively parallel system. There is no benefit to doing this and it can greatly reduce the available compute capacity to the workload and also to the other pending workloads that must now wait for this non-optimal load to complete and release resources. This is true of all distributed compute systems and the decentralized space afforded by blockchain networks share those same concerns specifically as well as scalability.

In order to compete with centralized solutions, decentralized networks need to overcome the learning curve of usability that application developers and their end users have, while simultaneously maintaining or improving on the convenience and functionality of centralized technologies. Current
blockchain solutions fail to meet the requirements crucial to mainstream adoption.

At this moment in time, nearly all public chains exist in silos, unable to access efficiencies gained by sharing resources with other networks. The exceptions to this are “mergemined” coins, which are sidechains and apps that are traded as currency. An example of this would be NameCoin which provides a blockchain backed DNS service and runs on top of the Bitcoin network. However it is only on the bitcoin network and this is what we mean when we refer to such network silos.

When thinking about the future of blockchain networks, it is prudent to understand the history of distributed processing and apply it to decentralized technologies. The primary difference between distributed and decentralized being in the level of trust required between nodes.

Typically a distributed system is still owned by a single party or group and there is inherent trust implied.

However a decentralized network is generally peer to peer and each worker node has a different owner. You cannot explicitly trust the nodes. You need to be able to verify the results and these results have to be verified with less computational effort than was used to obtain the result in the first place. This makes decentralized computing ideal for NP Hard problems. An NP Hard problem is one in which it is difficult to compute the answer initially, but once the answer is known, it is correspondingly easier to verify.

Not all workloads can be broken down into a series of parallelizable NP Hard steps. But many can including any sort of “solve for x” operation, where x is a totally unknown number in a broad search space. OPEN applies these methodologies to blockchain networks. By treating these networks as processors through which we can thread transactions and have them processed in parallel, we allow throughput in the utilization of OPEN Chain to expand far beyond what any individual blockchain is capable of doing.

Scaffolds function as an on-chain structure for translating data across these diverse networks into one OPEN State. The Scaffold enables the processing of transactional data from multiple networks into an “OPEN State” that updates information about a user to reflect information from the decentralized payment.

OPEN Chain integrates these disparate blockchains and offers transactional scalability and data interoperability for applications, functioning as a load balancing protocol for blockchain networks.

**Scalability**

Scalability and throughput of blockchain networks have held back developers from replacing centralized applications with decentralized counterparts since the conception of Bitcoin. Without achieving a level of scalability which rivals centralized technologies, it may be infeasible for application developers to utilize blockchain or other decentralized technologies effectively.

In a traditional distributed system, each node offers computational capabilities, increasing the resources of the system as a whole.

For example, if a file is hosted as a BitTorrent, each node hosting the file increases the speed of the file download for the end user. It does so by offering the file in smaller more manageable chunks. These smaller chunks are more likely to be “accelerated” by the ISP than a single large file. The smaller chunk sizes are also less likely to run into interference and trigger backoff effects which would slow the transfer.

The same does not hold true for blockchains and decentralized infrastructure. Each full node stores a copy of the entire blockchain history, and they must come to an agreement over an acceptable state of all transactions between all parties before submitting the next block to be appended to the chain.

Once a new block has been found, each block must then be replicated in all nodes in the system before those nodes can resume computing new blocks.

The net effect is that, rather than adding resources to
the system, each node adds to the work needed to be done by the system. This workload is not immediately obvious, but it is a penalty paid in terms of lack of consensus. Since each node must know the entire state of the system, nodes which are out of consensus with the majority present a nuisance hazard that scales with the number of non-compliant nodes.

Current scalability solutions attempt to solve the problem through new consensus mechanisms, but even these have the theoretical ceilings of capacity. Multiple new consensus mechanisms such as the Delegated Proof of Stake [2] and Federated Byzantine Agreements [3] models, are attempts in development to address scalability of the consensus model by changing what consensus means. There are also off-chain scalability solutions like Lightning and Plasma that allow users to move transactions off-chain, but these solutions are still theoretical, isolated to specific blockchains, incompatible with developer needs, and not interoperable between blockchains.

**Interoperability**

Blockchain networks have started out by representing virtual currencies, but have since expanded to include additional features such as storage of financial data or optimization for privacy. In doing so, these chains naturally exist in isolated environments, i.e. silos.

The fact that they are unable to communicate with each other is a serious problem for application developers. Developing a scheme in which developers are able to utilize one or many chains for their core efficiencies in a particular solution, while minimizing the reliability or risk of failing components or data of a single chain, is the subject of much research.

**Cosmos**

Cosmos [7] employs a hub-and-spoke architecture in order to facilitate interoperability between blockchains. At a high level, the Cosmos Hub and Zones are smaller chains, that can be utilized to create inter-chain value transfer. Additionally, Zones within Cosmos must employ the Tendermint Consensus Protocol. The Hub-and-spoke network topology is shown below:

There is a problem with this approach. For interoperability to occur, each existing blockchain must be rebuilt in the Cosmos ecosystem. Cosmos does not achieve true interoperability between blockchains as it only has the ability to attain interoperability in its own network.

Other protocols serve to facilitate information and value transfers between blockchains that typically cannot communicate with each other. These protocols connect blockchains by creating sidechains, structures that allow for transactions between a parent chain and child chains through a two-way peg. This method often utilizes two blockchains, one to connect the traditional blockchain data structure, and one to accommodate the Directed Acyclic Graph data structure. The level of interoperability is also limited to solely value exchange, making it more of an exchange as opposed to a truly interoperable blockchain. There is also a difficulty in ensuring finality in side-chains because the tokens must be tracked on all chains and verified for correctness for
each chain. This takes much longer to verify because a different protocol is dealt with on each chain.

**OPENChain Architecture**

OPENChain is unique in that it utilizes its on-chain and off-chain architecture to solve issues with interoperability and scalability while introducing a mechanism for transaction processing interoperability.

OPEN Chain's core technology is focused on utilizing a combination of its own interoperability protocols to achieve a new level of scalability. In order to solve throughput issues, OPENChain uses a consensus mechanism known as ORapid Consensus, in conjunction with OPEN’s cross-chain interoperability architecture. This enables us to parallelize transactions across multiple chains and use the OPEN Scaffold in order to translate data and maintain synchronicity. By combining these two technologies, OPEN is creating a highly scalable and flexible blockchain allowing application developers to integrate backends into decentralized technologies. The ORapid Consensus mechanism is a significant improvement upon the Delegated Proof of Stake model. Through this mechanism, OPENChain token holders continuously elect a set of 27 block operators, approving the next 243 blocks, 9 blocks per 27 operators, who have staked a specific amount of OPEN Token. If any of the producers act maliciously they can be replaced by the next round of voting by OPEN token holders. It achieves high throughput by having elected block operators submit and validate blocks.

Once the operators are elected, each one gets a chance to submit a block for validation. Once a quorum of 19 operators has validated a block, it is considered immutable and added to the longest chain. OPEN Chain has a target of an average block time between 1.5 seconds and a TPS of 1000. Additions may be necessary to ensure correct consensus. These additions will work to implement the Operator oriented ORapid consensus, where token holders ‘vote’ for operators that they trust. This entails having votes broadcasted via cryptographic signatures [7]. Eventually, OPEN operators will be able to perform additional services for the network and then be compensated for such services in a stacked ORapid Consensus mechanism.

**OPEN Cluster**

The Blockchain Load Balancing Protocol is able to perform load balancing by grouping nodes from several blockchains into one OPEN Cluster. The OPEN Cluster acts as a multi-chain node that can query and broadcast transactions to a variety of chains. This enables the OPEN Cluster to act as a gateway for sending transactions and as a protocol-level implementation of the threading used in traditional networks.

![OPEN Cluster Diagram]

OPEN Cluster is constantly accessing the network load of OPEN Chain and participant blockchains. Once a user initiates a transaction, it goes through the OPEN Chain only if OPEN Chain’s network load is not at its capacity and other blockchains in the OPEN Cluster are not engaged with. However, if OPEN Chain has hit its network capacity, the OPEN Cluster sends the transaction through another blockchain in the Cluster through the Interoperable Blockchain Client. Thus we are able to achieve parallel processing allowing us to access scalability competitive with centralized architectures.

Clustering is used in modern Network Load Balancing implementations. Compute clusters came about through the existence of several high-speed networks and the need for high performance processing in applications. Notable implementations include Microsoft Windows Cluster Server 2003 that was built upon the Windows Server platform. This was used to support multiple high performance computing tasks such as MSMP Ilibrary and management tools. Similar trends driven by the same technological problems are emerging now.

**Blockchain Load Balancing Protocol**

Blockchain Load Balancing Protocol (BLBP) is a mechanism that OPEN Chain uses to parallelize transactions across a variety of chains, by using each
chain as a potential means to send a transaction. This distributed chain approach means that the throughput of transactions scales with the number of chains connected by the protocol. To increase the speed of transaction throughput, OPEN Chain routes transactions through the least latency blockchain. This will be blockchains like Tendermint which is fortunate because blockchains like Tendermint also tend to cost the least.

Nevertheless there will be times when every network has forked or is encountering congestion or has its fees driven up. Our protocol enables transactions to be sent through the chain with the fastest predicted transaction time and broadcasts transactions to this chain, thus taking advantage of the security and compute capacity offered by the larger networks, during times when they are price competitive.

In this manner, OPENChain combines the power of every chain. Instead of competing solely on scalability, OPENChain strives to be the best in versatility which is what enables true scalability.

**Base Mathematical Model for BLBP**
Assume a blockchain in the load sharing network $(U,V,N)$ as having the following characteristics a demand vector $\lambda$, and a positive real number $\gamma$, we consider the stochastic description of blockchain network dynamics as:

1. Each $u \in U$ users arrive by a Poisson process rate of $\gamma \lambda(u)$
2. Assume each network has non-trivial capacity, therefore can accommodate every user
3. Each user has a holding time that is exponentially distributed with unit mean regardless of past action
4. For the base model, we must assume each user does not change their transaction type until their transaction is recorded.

Given $X_u(v)$ denotes the load at blockchain $v \in V$ at time $t$, and set $X_t = X_u(v) : v \in V$ the transaction arrival and departure times, combined with the OPEN Cluster allocation policy and an initial condition, determine the load process $(X = X_u : t \geq 0)$. In a static load balancing protocol, we make assumptions about $\Phi$, where $\Phi : R^V \rightarrow R$ be a strictly convex, differentiable function which is symmetric in its argument. In addition to that assumption, for the base dynamic case, we assume $\Phi(c_x) = c^T \Phi(x)$ for all $c > 0$ and $x \in R^V$ for some $p > 1$. Thus this leads to $\Psi(c\lambda) = c^T \Phi(\lambda)$ for $\lambda \in R^{|V|}$. A potential definition of $\Phi$ is $\Phi(x) = \sum(x(v))^p$. The performance metric for a base balancing policy $\pi$ is the long term average cost of $J_\pi$ defined as:

$$J_\pi = \liminf_{T \rightarrow \infty} E \left[ \frac{1}{T} \int_0^T \Phi(X_t) dt \mid X_0 = x_0 \right]$$

The Dynamic Blockchain Load Balancing Protocol minimizes $J_\pi$, which OPEN does through Least Traffic Routing (LTR).

Let $L_u(t)$ denote the number of $u \in U$ as before in the network at time $t$, and set $L = (L_u(u) : u \in U)$. The value problem of Static Blockchain Load Balancing $(L, \Phi)$ gives us a stable lower bound of balancing costs at time $t$ under any allocation policy, upon which OPEN Blockchain Load Balancing improves upon by using the least-traffic balancing principle as described below. The process $(L_u(v) : t \geq 0)$ for fixed $u$ is represented by an $M/M/\infty$ waiting time given the previously mentioned load factor of $\gamma \lambda(u)$ allowing us to give the equilibrium distribution of $L$ as a vector described by $L_v(u) : u \in U$ of Poisson random variables with mean vector $\gamma \lambda$. This gives us the lower bound on general load balancing protocols, of which Least Traffic Routing is modeled below:

$$J_\pi \geq \liminf_{T \rightarrow \infty} E \left[ \frac{1}{T} \int_0^T \Psi(L_t) dt \mid X_0 = x_0 \right]$$

$$= E \left[ \Psi(L_t) \right]$$

$$\geq \Psi(\gamma \lambda) = \gamma^p \Psi(\lambda).$$

**Other Balancing Implementation Considerations**

LTR is the implementation discussed in this whitepaper, but others were considered. Other Load Balancing implementations considered are discussed below because they bear mentioning.
**Round Robin**
Under Round Robin (RR), Chain 1, 2, and 3 would receive the first, second, and third transactions respectively. Once Chain N receives transaction N, the cycle is reset and transactions are then routed accordingly again.

While Round Robin may be simpler to implement, several issues exist with this routing method that are counter to our implementation. For example routing does not take into account the traffic of a specific chain before routing a specific transaction which is critical. Additionally, RR is static and more centralized in nature.

**Weighted Round Robin**
Under weighted RR, each chain would be given a static rating. This rating would determine how much traffic a particular chain receives over a set period of time.

However, one downside to this approach is the static nature of the load balance technique. Chain traffic is dynamically changing requiring a dynamic load balancing method to allocate transactions efficiently.

**Agent Based Adaptive Load Balancing**
In this implementation, each blockchain has an agent that reports on the load that the blockchain is experiencing. Such real time information is used in determining to which blockchain we should route transactions to.

Agent Based Adaptive Load Balancing is typically used in conjunction with other load balancing protocols, and OPEN Chain uses a modified version of it as well with LTR. OPEN Chain utilizes a call to a blockchain to gain necessary information to determine the load via the wallet - we do not have the agent itself report the load in order to ensure veracity.

**Chained Failover**
Chained failover is another simplistic implementation of load balancing as different networks are put into a predetermined order to route transactions. All requests are sent to blockchain A, if that is full, then blockchain B, and so on.

While OPEN Chain employs a flavor of this, in that OPEN Chain is always the first blockchain network considered, OPEN Chain uses LTR to determine the ordering of the rest of the blockchains. By doing so as a parallel process, the latency in routing is minimized.

**Weighted Response Time**
This method is similar to LTR, but its implementation to determine latency is by calling a network and judging the response time to determine which has the least load. This is done iteratively, by sending a request to each server sequentially, and then calculating based on the response.

Such an implementation suffers from low throughput in routing as it depends on a sequential call and response model. Furthermore, it necessitates a trusted entity constantly being online to make requests and responses which threatens implementation security.

**Least Traffic Routing**
OPEN’s BLBP mechanism leverages the Least Traffic Routing model. Under LTR, resource allocation is optimized for high traffic scenarios. Through LTR, transaction u arrives at the OPEN Cluster and is then routed onto the path with least traffic in locations identified in N(u). If all locations in N(u) are at the same capacity, then transaction u is routed arbitrarily. LTR can be implemented in a distributed manner and each incoming transaction can be routed based on partial information about the OPEN Cluster state.

The function used to understand a particular location’s current state and will be discussed in more detail below. The aggregation of location states provides a view of the Cluster state. LTR is effective in routing transactions in high traffic states, allocating resources more efficiently than traditional static routing.

Under LTR, the load process X is Markov on the state space $Z^V$. For $v \in V$, we can simply define the operator $T_v : Z^V \rightarrow Z^V$, as
function that takes in chain-dependent variables to give a weight for each chain. Chains with lower weights correspond to faster transaction completion time. The current evaluation function for finding the best service relies on predicting the amount of time that it will take to clear all of the pending transactions. The chain that will take the least time is given the transaction to broadcast.

It is important to note that these calculations must remain simple and deterministic for two main reasons:

1. They must be efficient to compute thousands of times to maintain an accurate picture of the chains involved.
2. They must be deterministic because the calculations need to be verified in case of auditing or achieving consensus between nodes viewing the same chain.

A simplified example of such is presented below in pseudocode:

```java
evaluate(block_time, pending_transactions, block_size) {
    //this is can be updated to include functions as well
    return block_size/block_time * pending_transactions;
}
```

**Auto Failover**

With the token sale boom that happened last year, we saw several occasions of the Ethereum network nearly grinding to a halt - an aspect of the network that makes it incompatible with modern day applications. We saw it again in late 2017/early 2018 with “Cryptokitties” - a basic application that drastically increased Ethereum transaction prices and stalled the network due to load.

A key feature of decentralized technologies is that there is no single point of failure, however such thinking needs to be expanded to a network level perspective. An application integrating the blockchain can not have its system functionality tied to a single network through which it routes all
transactions. There needs to be redundancy in capacity in case one goes down as we have seen several times.

In computing and networking, the same issue is solved by introducing the concept of failover. Failover is when one switches to a redundant or stand-by server or network in the event that the main one is compromised. OPEN Chain’s Blockchain Load Balancing protocol effectively enables this to exist for decentralized technologies. In the event that one blockchain goes down, traffic can be automatically re-routed to different member blockchains in the OPEN Cluster. Thus ensuring system reliability in decentralized technologies that exist in centralized ones. This is a key feature of OPEN Chain that will central to adoption.

**OPEN Cross-Chain Interoperability**

OPEN Chain provides a powerful interoperability protocol that is able to link transaction data across all blockchains to application databases. One instance of such a system enables the flexibility of any application to have full utilization of multiple blockchains on a single application build. Cross-chain propagation of data is enabled by creating blockchain-agnostic gateways, then collecting, serializing, and propagating blockchain payments to a distributed data layer that can be utilized by an application’s existing databases. These gateways, called Scaffolds, can be changed to update specifications on the payment schema or data collection variable types. Any updates on the Scaffold can be reflected upon the data networks throughout a variety of chains.

The OPEN Distributed Data Network allows for a global data reference across chains. This data is routed to Scaffolds as they receive and broadcast transactions to different chains via the OPEN Chain Universal Client.

The OPEN Distributed Data Network facilitates global data referencing across chains. By querying data requests through the OPEN Distributed Data network, the data propagated by any Scaffold can be made global. Global data is especially important for cross-chain interoperability and scalability because applications require a single source of truth that is independent of the chain that the payment is sent on. Interoperability means that transaction data can be routed and updated from Scaffolds as they receive and broadcast transactions to different chains via the OPEN Chain Universal Blockchain Client.

**Enhanced Data Interoperability**

OPEN Chain has been designed with a protocol to provide seamless data flow between chains. This protocol uses a specialized network, the OPEN Distributed Data Network, to address and update data in a blockchain-agnostic manner. Using OPEN Chain, it becomes possible to reference data from every chain, thus enabling a secure and scalable two-way communication flow between applications and blockchains. Applications that use the OPEN Chain can connect the transaction data from multiple different chains into one source of truth. This functionality is of paramount importance when compiling user receipts and states for applications.

**OPEN Distributed Data Network**

The OPEN Distributed Data Network allows off-chain data to be referenced and updated by Scaffolds on any chain. The Protocol stores data in a distributed network so that it can be referenced and updated from a variety of chains. This network is enabled by using our enhanced version of the Kademlia Distributed Hash Table [8] (DHT). This system employs address-based ids to ensure that
application Scaffolds can always be looked up regardless of the data changes within them.

Kademlia Distributed Hash Table
The Kademlia Distributed Hash Table is optimized for interactions in peer-to-peer networks, and has wide industry acceptance with BitTorrent and Gnutella. Kademlia is composed of a distributed set of nodes that store and broadcast information throughout the network. Each node is referenced in the network by a cryptographic hash of their node value, known as their node id.

- Nodes are responsible for routing within the network, by maintaining a ledger of routes to nodes that they have referenced before or that are close to them known as their neighbor nodes.
- Distance between nodes is measured with the XOR function, where node ids are closer with a lower number of differing bits.
- Resource tables are updated as nodes encounter other nodes.
- Retriever nodes have the potential to store the data that they connect to.

These properties grant Kademlia some power attributes:

- Queries can be done in $O(\log(N))$ time because of the XOR distance metric and $k$-bucket structure of the network itself. This enables the efficient search of large networks.
- There is minimal vector for attacks because retrievers can store the data. This scales the available resources with the number of requests.

The OPEN Distributed Data Network builds on the Kademlia Distributed Hash Table by adding specialized database level utility:

Push-Pull Data Relationships
Similar to version control systems, a user can submit data and request that the data be updated by the leader, capturing the changes in a filesystem tree. An immutable object represents all of the files, directories, and changes. The data is divided up into equal size pieces of data, and stored as hashes in Merkle DAG, which represent the pieces of the file as a whole. When changes are made to the file, only the hash representing the data being changed needs to be updated. Architecturally, immutable objects represent files, directories with trees, and changes as commits. Objects (files) are content-addressed by the SHA-256, or other cryptographic hashes, of the content in the Object. Following the structure of a Merkle DAG, links to other objects are embedded providing integrity and work-flow properties that can be checked and computed efficiently. That is partly because versioning metadata are trivial pointer references and thus inexpensive to compute and update. Version changes, of course, only update references or add objects. Distributing version changes to other users is only transferring objects and updating remote references. This design allows users to request data changes for their states and to have developers algorithmically accept these changes based on transactions [9].

This by itself would appear to make Kademlia an ideal solution for any distributed data store. Except that Kademlia is highly susceptible to Sybil attacks. Thus, using Kademlia as an ultimate source of truth
could result in chain poisoning. We enhance it to make it less susceptible. This is beyond the scope of this whitepaper, but will be discussed in detail in future publications.

**Universal Ledger for Data**

Since the data in the OPEN Distributed Data Network is independent of any blockchain, it can be referenced by all of them through the SHA-256 hash of the scaffold address. This means that Scaffolds on a variety of different chains can reference and update the same piece of data. The result of this data interoperability and synchronicity enables the OPEN Distributed Data Network to act as a universal data layer for applications. Applications have the ability to compile one source of truth from the transaction data from multiple different chains.

**Interoperable Blockchain Client**

An element of Ethereum's popularity is that they have built a toolset around their blockchain that makes it easier for developers and non-technical users to interact with their chain. OPEN Chain relies on an analogous mechanism, but instead of having a single chain client, OPEN Chain boasts a universal blockchain client that facilitates the interactions between multiple chains. It is capable of interacting with a number of different chains, doing for all chains what the web3.js package does for Ethereum. Through the Universal Blockchain Client it becomes possible to interact with local, remote and native nodes on every blockchain through an HTTP or IPC connection. The functionality of an Interoperable Blockchain Client is necessary for load balancing amongst different chains. It is important to note that not every chain supports the same feature set. Bitcoin and derived currencies have a somewhat limited feature set, that while able to do much, are still very limited compared to ETH and its children. In cases where certain features are required from one

network that is not available on others, the business logic is submitted to a lower cost fork of the coin. For example ETH has features only found in ETH and ETC. An application seeking to take advantage of those features would be routed to ETC when ETH is congested, thus lowering total cost of ownership.

**High Level Network Flow**

Transactions first hit OPEN Chain, but after the OPEN Chain network capacity is hit, transactions are then routed through the OPEN Cluster and threaded into different blockchains. All transactions are translated by the Scaffold and pushed into the OPEN State. Application backends will pull updates from the OPEN State to improve upon their own blockchain.

For example, when a particular network like Ethereum is experiencing higher than normal block write times as well as gas fees, a transaction can take place through chains with less pending transactions or chains with faster consensus mechanisms for application specific processes. The potential payment chains can be set before the transaction is broadcast, and then the cluster can send the fast transaction. Load balancing enables a level of scalability beyond what any other protocol is capable of doing by leveraging capabilities of all blockchains, as opposed to being stuck at the ceilings of isolated protocols. Through load balancing, applications can guarantee users that they will be placed in the fast lane for transaction verification.

**Conclusion**

OPEN Chain represents a protocol that is made to appeal to a wide set of applications with its focus on utilizing multiple consensus mechanisms at once and interoperability of data across multiple chains; features that are sure to appeal practical use cases of blockchain technology. It uses interoperability to provide higher transaction throughput, and thus
scalability. The cross-chain communication design that OPENChain supports, is meant to act in tandem with the chain’s global data interoperability. This enables further responsive cross-chain interactions, and can be used to provide a layer of privacy that does not infringe upon security. The future of the OPENChain is focused on high throughput transaction needs surrounding and will evolve around application requirements. It will be iterated upon to provide additional functionality around the evolution of various unique consensus throughput mechanisms with a focus on utilizing its own consensus mechanism in tandem with other high throughput mechanisms on various chains to achieve the most efficient level of transaction processing possible.

References