On spatio-temporal blockchain query processing

Qu, Qiang; Nurgaliev, Ildar; Muzammal, Muhammad; Jensen, Christian S.; Fan, Jianping

Published in:
Future Generation Computer Systems

DOI (link to publication from Publisher):
10.1016/j.future.2019.03.038

Publication date:
2019

Document Version
Accepted author manuscript, peer reviewed version

Accepted Manuscript

On spatio-temporal blockchain query processing

Qiang Qu, Ildar Nurgaliev, Muhammad Muzammal, Christian S. Jensen, Jianping Fan

PII: S0167-739X(18)31421-3
DOI: https://doi.org/10.1016/j.future.2019.03.038
Reference: FUTURE 4863

To appear in: *Future Generation Computer Systems*

Received date : 11 June 2018
Revised date : 15 March 2019
Accepted date : 18 March 2019

Please cite this article as: Q. Qu, I. Nurgaliev, M. Muzammal et al., On spatio-temporal blockchain query processing, *Future Generation Computer Systems* (2019), https://doi.org/10.1016/j.future.2019.03.038

On Spatio-temporal Blockchain Query Processing

Qiang Qu[a,e], Ildar Nurgaliev[e], Muhammad Muzammal[a,b,*], Christian S. Jensen[c], Jianping Fan[d]

[a] *Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China*
[b] *Department of Computer Science, Bahria University, Pakistan*
[c] *Department of Computer Science, Aalborg University, Denmark*
[d] *University of Chinese Academy of Sciences, Beijing 100049, China*
[e] *Xinlian Technology Co.,Ltd, China*

Abstract

Recent advances in blockchain technology suggest that the technology has potential for use in applications in a variety of new domains including spatio-temporal data management. The reliability and immutability of blockchains combined with the support for decentralized, trustless data processing offer new opportunities for applications in such domains. However, current blockchain proposals do not support spatio-temporal data processing, and the block-based sequential access in blockchain hinders efficient query processing. We propose spatio-temporal blockchain technology that supports fast query processing. More specifically, we propose blockchain technology that records time and location attributes for the transactions, maintains data integrity, and supports fast spatial queries by the introduction of a cryptographically signed tree data structure, the Merkle Block Space Index (BSI), which is a modification of the Merkle KD-tree. We consider Bitcoin-like non-uniform block generation, and we process temporal queries by means of a block-DAG data structure, called Temporal Graph Search (TGS), without the need for temporal indexes. To enable the experiments, we propose a random graph model to generate a block-DAG topology for an abstract peer-to-peer network. We perform a comprehensive evaluation to offer insight into the applicability and effectiveness of the proposed technology. The evaluation indicates that TGS-BSI is a promising solution for efficient spatio-temporal query processing on blockchains.

Keywords: Blockchains, spatio-temporal data, authenticated data structure, block-DAG

1. Introduction

Blockchain as a transformative technology has found early use in the financial domain [1], but has recently found application in a variety of other domains. A blockchain is a distributed, decentralized, and trustless ledger that supports the reliable and secure recording of transactions. Blockchain technology has demonstrated its applicability to business solutions in sectors such as finance, healthcare, and education [1, 2, 3]. Consider, for instance, a supply chain scenario where an object is tracked as it is undergoing transportation. The tracking mechanism requires not only that the spatio-temporal information is updated continuously, but that queries regarding the object's time-varying location are also supported. Typical queries include, for example, 'list all objects at location l at time t,' or 'list all objects that moved within radius r of location l during time interval $[t_1, t_2]$.' The support for such queries is desirable, for example, for logistic decisions or product monitoring. However, a blockchain implementation of such a business scenario is challenging. For example, spatio-temporal data grows at a higher rate than does the transaction data currently supported by financial blockchain systems. Further, consensus protocols for spatio-temporal data require proof-of-location processing. Also, the sequential access mechanism in blockchains does not support efficient query processing.

The value proposition of blockchains over traditional databases is the data integrity through cryptographically signed historical data. For example, financial institutions require a signed 'append-only' data structure that is auditable and traceable [1, 4]. Large enterprise service providers such as Google or Amazon require spatio-temporal analytics on user data for providing continuous services in given time and space [5]. Therefore, a spatio-temporal blockchain system design should take into account two considerations, (i) the scale at which such a system is to be used, and (ii) the kind of query support that is required of the system.

It is not straightforward to directly adapt database concepts to a blockchain system. A spatio-temporal blockchain system design should consider secure data storage and efficient query processing simultaneously. This work provides a conceptual block design for efficient queries in block directed acyclic graphs (Block-DAG). Block-DAG is the blockchain alternative that makes possible to achieve high throughput by way of fast block creation.

More specifically, we consider spatio-temporal data processing in a blockchain setting and enable querying of blockchains without expensive local indexing. We integrate the Merkle-tree [6] with the spatial indexing such that spatio-temporal queries are supported without the need for additional indexes. Also,

*M. Muzammal is the corresponding author.
Email addresses: qu@xinlian.ai (Qiang Qu), ildar@xinlian.ai (Ildar Nurgaliev), muzammal@bui.edu.pk (Muhammad Muzammal), csj@cs.aau.dk (Christian S. Jensen), jp.fan@siat.ac.cn (Jianping Fan)

we include additional timestamp information in block headers that enables temporal queries directly on blocks. We assume a spatio-temporal blockchain with an abstract consensus such that the credibility of the data is maintained by the consensus algorithm. We assume that a typical transaction has the following attributes: timestamp, longitude, latitude, and hashed account identifier. Overall, we enable blockchain for efficient query processing without the requirement of additional local indexes.

To summarize, data about moving objects can be stored in a spatio-temporal blockchain. Location services such as object tracking can enable many real-life tasks such as finding lost items [7], enabling autonomous delivery vehicles [8], and providing a foundation for next generation supply chains [9]. The advantages of distributed ledger technology are, however, often blocked by the technology's limitations in relation to real-life application domain requirements. One key such limitation is the lack of support for efficient queries directly on spatio-temporal blockchains. To the best of our knowledge, this work is the first attempt to enable efficient queries on spatio-temporal blockchains following the preliminary study [10].

We make the following specific contributions.

1. We propose the concept of block-DAG with pair-wise block order for spatio-temporal data storage that offers advantages over sequential blockchain access. We also propose improved block header organization that supports efficient spatio-temporal queries.

2. We introduce the Block Space Index (BSI) for blocks in order to maintain the integrity of a block's body. We propose a variant of the Merkle Patricia tree to maintain the global current position change per account and also propose the TGS-BSI algorithm to enable spatio-temporal queries on DAG-chains directly by traversing block headers.

3. We also propose simplified current position verification on mobile clients that facilitates tracking of a product associated with a particular account. We use a Merkle-Patricia-trie on the peer side and local block header information on the client side to authorize peer responses.

4. We propose a random-graph model to generate a block-DAG topology for an abstract peer-to-peer network and demonstrate the effectiveness of the solution with the help of a detailed experimental study.

The rest of the paper is organized as follows. Section 2 covers related concepts, and Section 3 provides an overview of the proposed approach. Details about block construction and authenticated spatial indexing are presented in Section 4. Spatio-temporal query processing is covered in Section 5. A detailed empirical evaluation is presented in Section 6, and Section 7 concludes.

2. Preliminaries

Usecases of blockchain technology are motivated by business requirements. The initial considerations for crypto-currencies were based on requirements such as decentralized authority, pseudo-anonymity, censorship-resistance, and reduced transaction fees. The resulting blockchain technology offers many benefits to users. These include, to inspect the quality of products and services [11], to obtain efficient and tamper-resistant digital content services [12], to realize tamper-resistant real-time supply chains [13], to secure IoT devices [3, 14, 15], and to enable privacy preserving behavior analytics [16].

The problem of indexing multi-dimensional data is well studied in the database community, but lacks consideration in the context of blockchain systems. An existing study [17] offers a detailed coverage of spatio-temporal query processing. In this section, we briefly describe location encoding systems, spatio-temporal indexing, and authenticated spatial indexes. We also comment on the advantages of block-DAG for spatio-temporal blockchain data management.

2.1. Location Encoding Systems

Location information is encoded in a variety of ways to enable location sharing. In geographic coordinate systems, latitude and longitude are typically used to capture a point location. A latitude is an angular distance between −90° and 90° and represents a location South or North of Earth's equator. A longitude angular distance ranges between −180° and 180° and represents a location East or West of an imaginary line through *Greenwich*. An alternative representation is to use a space filling curve. The idea is to discretize space into cells, typically by means of a uniform grid. Then the cells are numbered by means of a curve that traverses all cells. A location is then represented by the number of the cell that it belongs to. A different approach, *Geohash* [18], encodes latitude-longitude pairs in a hierarchical data structure as unique strings.

2.2. Spatial Indexes

Spatial indexes typically store spatial locatons into a hierarchical data structure [19, 20, 21]. For example, the R-tree [22] and its variants, including the RT-tree [23] and the 3D R-tree [24] is a popular spatial index. The kd-tree performs better under the assumption of an initial bulk load and no subsequent data changes. This property makes it best suited for cases where the data is static. The kd-tree exhibits many favorable properties and has proven to be efficient in practice for low-dimensional data [25].

Another important aspect of data processing is data verification. Many authenticated data structures have been proposed for indexing spatial and spatio-temporal data to support verifiable queries such as range queries, *k*-NN queries, reverse *k*-NN queries, and skyline queries [26]. Authenticated versions of spatial indexes include the Merkle kd-tree [27] and the Merkle R-tree [28]. Proposals also exist for *k*-NN-based spatial queries [29].

2.3. NoSQL Multi-dimensional Indexing

Spatio-temporal indexing structures enable efficient query processing by maximizing the de-normalization capabilities [30].

ACCEPTED MANUSCRIPT

Full-text search based database solutions are used widely to enable spatio-temporal analytics. Data storage is by way of specialized data structures similar to Bkd-trees [31]. GeoMesa [32] is an open-source distributed database system that supports spatio-temporal indexing using the Z-order curve to index space and time. Fox et al. [30] enable spatio-temporal indexing in NoSQL solutions where the data is de-normalized by way of column *families* and *qualifiers* that are implemented in the *Accumulo* NoSQL solution.

2.4. Blockchain Cryptography and Authenticated Index Basics

A collision-resistant hash function $\mathcal{H}$ maps a string s to a bit vector of a fixed-length such that $\mathcal{H}(s)$ is fast to compute and it is computationally infeasible to find a collision as $\mathcal{H}(s_1) = \mathcal{H}(s_2)$ if $s_1 \neq s_2$ [33]. The Merkle Hash Tree (MHT) has proven to be a general base for a varity of authenticated DAG structures [34]. The MHT hierarchically organizes hashes to verify the integrity and validity of blocks by way of providing a tiny number of hashes or Verification Objects (VOs) [6]. These properties facilitate the applicability of MHT in blockchain systems [35] with use case specific adaptations. For example, Ethereum [36] uses a Merkle Patricia trie to maintain the integrity of the global key-value states where the key is a 32-byte account identifier and the value is the account's state.

2.5. Blockchain for location

Location data has also been considered using the Blockchain technology. FOAM, as described by the authors [37], is a protocol for decentralized geo-spatial data markets designed to empower users to build a consensus-driven map of the world that can be trusted by applications. Another interesting example is ChainSQL that integrates blockchain and database technology, thus enabling support for SQL in a blockchain setting [3]. The study [15] proposes a spatio-temporal protocal to prove the locations in the setting of blockchain.

2.6. Blockchain and block-DAG

Bitcoin [1] and Ethereum [36] are well-known blockchain systems where transactions are publicly accessible in an anonymous way. The transaction accuracy is driven by the fact that consensus creates truth. However, throughput in blockchain systems is a bottleneck as the transaction confirmation times are not comparable with those in database systems. Blockchain performance is strongly coupled with the lying consensus protocol and hard-coded limitations on computations per block. The concept of block-DAG [4] is based on the idea of multiple references from every block to its predecessor blocks with possible conflicting transactions. As a consequence, this leads to changed transaction acceptance rules, where the graph topology is used directly as votes that help identify a robust subset of a block-DAG with no conflicting transactions. A block references all the blocks that its miner was aware at the time of block creation. Thus, the blocks that take a long time to propagate are prone to be rejected by the system. A block is considered valid only if all of its predecessors are valid and are known by the node. Thus, if a block B_i references some block B_j, there is no need to reference the predecessors of B_j; this is implied. To ensure the stable working of block-DAG based systems, miners (i) must reference the valid points of the DAG, and (ii) should quickly broadcast blocks they create or receive. In a block-DAG, there is no assumption that all miners share the exact same view of the DAG at all times.

Table 1: Frequently used notation

Symbol	Meaning
B	Block; B_{header}: block header; B_{size}: block size
D	Dataset $D = \{x \mid x \in R^d\}$, d: dimensionality
$\mathcal{D}$	Network delay
$G(V, E)$	Topology of a block-DAG; V: set of blocks; E: set of references
$\mathcal{H}$	Crypto graphic Hash function
$\mathcal{H}_R$	Merkle hash root function
$\mathcal{T}$	Transaction; $\mathcal{T}_r$: Transaction rate per second; $\mathcal{T}_n$: Total transactions
$g_{\mathcal{T}}$	Cryptographically signed $\mathcal{T}$ by user's PRIVKEY
k	Answer points in k-NN and bounded k-NN
q	Hyper-rectangle space range $q = [x, y]$ or point $q = \{x\}$ where $x, y \in D$
r	Radius; r_b: bounding radius
α	Standard deviation s.t. $\mathcal{N}(\mathcal{T}_r, \alpha^2)$
β	Time range s.t. $[\beta.start\_time, \beta.end\_time]$
σ	Deterministic search procedure
	σ_β: Time range search on $G(V, E)$ by β
	σ_q: Range search; σ_{q,r_b}: Ball-point search
	$\sigma_{k,q}$: k-NN search; σ_{k,q,r_b}: Bounded k-NN
(ϕ, λ)	(Latitude, Longitude)

3. Problem Formulation

In this section, we present related concepts, definitions, and assumptions. We rely on the block-DAG paradigm as an alternative to blockchain and propose a block header that aims to allow efficient spatio-temporal query processing. Table 1 provides an overview of frequently used notation.

3.1. Related Concepts

We first formalize concepts including transaction, block, and block header.

Transaction. A transaction, formally $\mathcal{T}$, is a set of attributes including *longitude*, *latitude*, *timestamp* (t), and *hashed account identifier* (uid) or the public-key of the transaction creator. We assume that the system records only spatio-temporal information. A transaction is a single instruction that is verified through a cryptographic-signature of the transaction hash $g_{\mathcal{T}} = Enc(\mathcal{H}(\mathcal{T}))$ while the block formation is on peer side. To include $\mathcal{T}$ to B_{body} a peer receives from a user the transaction data, the digital signature of the transaction $g_{\mathcal{T}}$, and the user's public key PUBKEY.

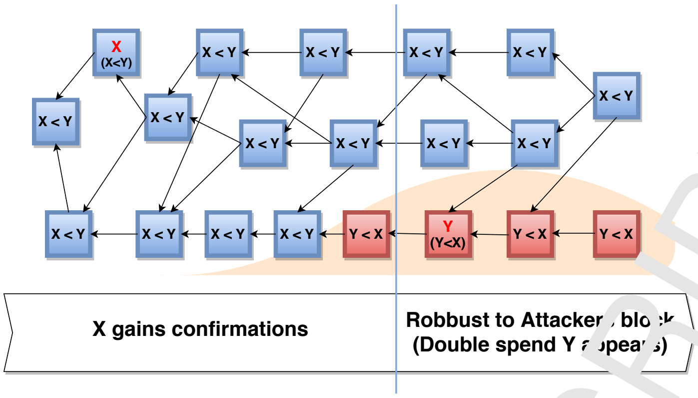


Figure 1: In a block-DAG, each block references all blocks to which its miner was aware at the time of its creation. The blocks, that take a long time to propagate (conflicting transactions) are ignored (red color).

Block. A block, formally $B = (B_{header}, B_{body})$, includes a block header B_{header} and an associated list of transactions B_{body} ordered w.r.t. the leaves in Merkle kd-tree that is organized by a bulk load algorithm which is applied to B_{body}.

Block header. A block header, formally B_{header}, contains information related to transactions and a set of hashes of other block headers. The block header maintains a spatial index per block. For the spatial indexes per block, following properties should hold: (i) header must be authenticated to guarantee that every other ledger holder has the similarly built index structure, (ii) spatial query support similar to range query, nearest neighbor query, ball query, etc., and (iii) fast access and verification from the last position of a particular item that is associated with its hashed account identifier. To avoid additional indexing of the whole block-DAG, we use Merkle kd-tree for its bi-functionality, i.e., block integrity verification similar to Bitcoin and fast spatial queries on a block. For simplified current position verification, we use Merkle-Patricia-trie and *location root* hash value from a particular block header. The block header contains the following information:

blockID: a unique identifier $\mathcal{H}(B_{header})$.

orphanHashes: a list of hashes of referenced block headers.

locationRoot: a 256-bit hash of the root node of the Merkle patricia trie populated with all the account identifiers and associated with the most recent location, time and number of records per account.

merkleSpaceRoot: a 256-bit hash of the root node of the Merkle kd-tree populated with each transaction of the block.

startTime: a scalar value equal to $\forall \mathcal{T} \in B_{body} : min(\mathcal{T}.t)$.

endTime: a scalar value equal to $\forall \mathcal{T} \in B_{body} : max(\mathcal{T}.t)$.

timestamp: is the time of block creation.

nonce: a 64-bit hash which proves that sufficient amount of computation has been carried during block creation.

3.2. Definitions

We formally denote block-DAG as G. G enforces a causal relation among blocks which states that if block B_i includes the hash of block B_j, then B_i must have been created after B_j. Although block-DAG supports fast block creation and short transaction confirmation time, a highly conflicting environment reduces the speed of transaction to be securely confirmed in a local image of G for a particular ledger holder. The SPECTRE protocol introduces $GetRobustAccepted(G)$ that is a function over G and it returns a subset of securely accepted transactions.

Definition 1 (input). *A subset of confirmed transactions in* G *are the input of* $\mathcal{T}$ *and belong to the owner of* $\mathcal{T}$.

Definition 2 (conflict). *Transactions* $\mathcal{T}_1$ *and* $\mathcal{T}_2$ *are conflicting transactions iff* $\mathcal{T}_1.uid = \mathcal{T}_2.uid$ *and* $\mathcal{T}_1.t = \mathcal{T}_2.t$.

Property 1 (Adjusted Consistency). $\mathcal{T}_1$ *is accepted iff* $\forall \mathcal{T}_i \in input(\mathcal{T}) : \mathcal{T}_i \in GetRobustAccepted(G)$*, all conflicts are rejected and the time-stamp of transaction* $\mathcal{T}$ *is not more than* δ*-away from current time-stamp.* δ *is system defined.*

The Adjusted Consistency property ensures a real system clock as the time-stamp in user transactions is user-defined and the system accepts transactions satisfied by δ only.

Property 2 (Weak Liveness). *If transaction* $\mathcal{T}_i$ *is published in* G *and no conflicting* $\mathcal{T}_j$ *is published, then it is included in* $GetRobustAccepted(G)$.

Property 3 (Pairwise ordering). *Once block B is published in G, the system guarantees that G contains blocks published before or exactly after B.*

Property 4 (Result Completeness). *A response set for a user query must not have any missing results.*

Property 5 (Block Soundness). *No modification takes place in the* B_{body}*, neither by adding non-existence transactions nor by modifying existing ones.*

A miner is expected to maintain the SPECTRE voting protocol to reveal the real order between each pair of blocks on the local image of G. The protocol holds these properties and states for fast block creation (at least five blocks per second). The aforementioned properties and a fast block creation rate allow the temporal queries to proceed over block-DAG topology without additional temporal indexes. Note that we also consider the case of a Light Node such as a mobile client which does not store the entire block-DAG [1].

Remark. The spatio-temporal block-DAG does not require a linear order among all the transactions because the tracking actions of an account don't affect other account actions. The linear ordering limits throughput in a blockchain and results in low block creation rate. This is a concern as the spatio-temporal data grows massively as a tracking object reports spatio-temporal information on a regular basis. Block-DAG is an alternative that makes possible to handle high throughput by implementing a fast block creation scheme. For the rest of this work, we use the terms blockchain and block-DAG, interchangeably. In this work, we assume that the transactions associated with an object have a linear order.

4. Enabling Block-DAG for Spatio-Temporal Queries

A transaction typically includes geodesic coordinates, *latitude* ϕ and *longitude* λ, to represent a location.

4.1. Geo-spatial Representation and Spatial Index

For queries such as k-NN, range, and ball-point queries, we need a Cartesian coordinate system represented by orthogonal axis. For the purpose, 'map projection' is used to convert the geodesic system to two-dimensional coordinates on the map. Nonetheless, it does not matter which coordinate system or geographic standard is used, as it is not possible to make a projection from a sphere onto a rectangle, i.e., a two-axis system, and save all data, i.e., angles and distances, simultaneously.

Example. Haversine is a measure to get approximate distance on a sphere. To use it on Earth with an approximate radius $r_e = 6371km$, consider three reference points $A = \{50°, 50°\}$, $B = \{50°, 51°\}$, and $C = \{51°, 50°\}$. For points A, B, and C, 2D coordinate system tells that they are equidistant based on Euclidean distance. But given latitudinal difference $\delta\phi = \phi_2 - \phi_1$, and the longitudinal difference $\delta\lambda = \lambda_2 - \lambda_1$ for pairs (A, B) and (A, C) according to Haversine (Eq. 1), the distance from point A to B is 111.19 km, but the distance from A to C is 71.47 km. That is why taking latitude and longitude values directly to a kd-tree is incorrect as it fails to compute k-NN and range queries by using euclidean distance. We compute Haversine distance as follows:

$$dist_{harv} = 2\, r_e\, arcsin(min[1, \sqrt{sin^2(\frac{\delta\phi}{2}) + cos\phi_i cos\phi_j sin^2(\frac{\delta\lambda}{2})}]) \quad (1)$$

For the k-NN and bounded k-NN search problems, a magnitude-comparable distance can be substituted for the relative distance since the relative ordering of the distances is more important than the actual distances [38]. The tunnel-through distance for points i and j is Eq. 3. For each index, we pre-compute Cartesian coordinates given by the latitude and longitude using Eq. 2 and store these on a Merkle kd-tree.

$$\left.\begin{aligned} x_i &= r_e cos(\lambda_i) cos(\phi_i) \\ y_i &= r_e sin(\lambda_i) cos(\phi_i) \\ z_i &= r_e sin(\phi_i) \end{aligned}\right\} \quad (2)$$

$$dist = \sqrt{(x_i - x_j)^2 - (y_i - y_j)^2 - (z_i - z_j)^2)} \quad (3)$$

4.2. Spatial Index per Block

The Merkle Hash Tree, or MHT, is generally used as a base for arbitrary authenticated directed acyclic graph structures. The scheme to do verification proposed on MHT is as follows: recompute the hash value incrementally by recreating Merkle tree root for a particular block as shown in Eq. 4.

$$\mathcal{H}_R(v_i) = \begin{cases} \mathcal{H}(byte(v_i)), & v_i\text{: leaf node} \\ \mathcal{H}(v_i, \mathcal{H}_R(v_{i,1}), ..., \mathcal{H}_R(v_{i,n})), & v_{i,j}\text{: successors of } v_i \end{cases} \quad (4)$$

As Merkle tree is designed for spatial queries only, we require a data structure that is able to process spatio-temporal queries. The SPECTRE protocol states that the block creation is typically within one second and the size of a block is expected to be within 40-70 transactions. We consider a three-dimensional space, i.e., a sphere in a Cartesian space along with an additional set of points, and use kd-tree for the purpose. A kd-tree has the advantage that it fits in the main memory and avoids a complicated structure similar to 3D R-tree with additional minimum bounding box coordinates. The kd-tree at each level has the Euclidean distance in one dimension and therefore has a reasonable performance for processing k-NN, range, and ball-point queries.

In this work, we consider a kd-tree authenticated by the Merkle scheme, i.e., an Mkd-tree, and propose a modification of Mkd-tree which we call Block Space Index or BSI. BSI stores one hash value for every node which is computed from the left and right hashes using Eq. 4. Location points are stored in the internal nodes of BSI. The leaves of BSI store transaction hash values where each leaf-hash is associated with the hash of internal node using Depth First Search as shown in Figure 3.

The bulk loading of BSI from the prepared list of transactions $\mathcal{T}_{list}$ for a block formation is presented in algorithm 1 where t is a timestamp of $\mathcal{T} \in \mathcal{T}_{list}$. The resultant reordered list of transactions $\mathcal{M}$ is written to the block and BSI is formed from $\mathcal{M}$.

Lightweight Client. We also consider the case of a Light Node, for instance, a mobile device, which does not store the entire block-DAG. A light client only needs to query node headers from the peer-to-peer network and select locally the valid parts by topological voting procedure over the network that is handled by SPECTRE protocol. A typical proof of transaction inclusion is handled by BSI. For the same, we send a verification object, VO, with the results of spatio-temporal queries to a lightweight client.

Algorithm 1 An outline of the steps for BulkLoad routine

Require: $\mathcal{T}_{list}, \mu, \mathcal{M} \leftarrow \emptyset$ ▹ μ: recursion depth; $\mathcal{M}$: empty kd-tree
Ensure: $\mathcal{T}_{list} \leftarrow \{\mathcal{T}.(X,Y,Z) \leftarrow \mathcal{T}.(\phi,\lambda) \forall \mathcal{T} \in \mathcal{T}_{list}\}$; ▹ $\mathcal{T}_{list}$ to Cartesian space
$\sigma \leftarrow \mu \; mod \; 3$ ▹ Choose splitting point
$\mathcal{T}_{list} \leftarrow sorted(\mathcal{T}_{list}, by\; \mathcal{T}.(X,Y,Z)[\sigma] \; and \; \mathcal{T}.t)$ ▹ Sort rest of transactions
$\mathcal{T}_{left}, m, \mathcal{T}_{right} \leftarrow$ MEDIANSPLIT($\mathcal{T}_{list}$) ▹ m: root of splitting
$\mathcal{M} \leftarrow m$;
BULKLOAD($\mathcal{T}_{left}, \mu+1, \mathcal{M}$) ▹ Recursion build left subtree
BULKLOAD($\mathcal{T}_{right}, \mu+1, \mathcal{M}$) ▹ Recursion build right subtree
return $\mathcal{M}$

4.3. Simplified Last Position Verification

In this section, we discuss account management and account location tracking.

4.3.1. Account Management

An *account* is a personal data that is associated with a hashed public key of a user's initially generated *key-pair*. The account includes the last record of space-time location and the number of stored transactions per account. The state of all accounts is the state of the whole block-DAG network. In then system, accounts are needed for tracking entities associated with the accounts. The presented block-DAG platform is a public transaction-based state machine that initiates from the genesis block, or *genesis state*, and incrementally updates the location position of associated anonymous accounts up to the last location or final state.

The main challenge of account management in a blockchain is frequent updates of values for each account. Therefore an authenticated data-structure that holds all the account information is different from managing transaction history in binary trees. We utilize Merkle Patricia-trie (MPT) that reflects associations between each account identifier and their actual data. In case of spatio-temporal data, the account information includes geodesic coordinates of the most recent geolocation, timestamp, and nonce where *nonce* is the total number of stored transactions for an account.

To efficiently support account state, MPT has the following features: (i) it is an authenticated data structure and is able to quickly recalculate a tree root after an insert or update operation such as create or update account or update the last geolocation and timestamp, (ii) the root is dependent on data and not the order in which updates are made, (iii) fast roll-back is supported to construct global block state that only has the fresh account information and history in the block, and (iv) it enables answering account specific queries, for example, 'what is the most recent position of account x?' or 'does account with id xxx-xx exists?'.

The value of MPT root or location root in the block header reflects a distinct version of the global state per block. The MPT implementation introduces a value driven data-structure through referencing each node by its hash, therefore, key-value is stored in a levelDB database where the value is the string representation of a node and the key is its hash. Thus, multiple copies of the historical states of each node allow fast roll-back. MPT consist of three types of nodes: (i) a leaf node that stores key-value pairs, (ii) an extension node that stores a hash of another node, and (iii) fixed length sets of branch nodes typically having 17 elements. The first 16 elements correspond to the sixteen possible hex characters in a key, and the final element holds a value if there is a key-value pair where the key ends at the branch node. A sample part of MPT version for a block is shown in Figure 2.

Note that in a blockchain, the state of MPT changes just from one block to another, but the block-DAG allows several references from a block to other blocks that adds to the complexity of the construction of MPT. In Algorithm 2, we introduce steps for a snapshot of MPT formation per block.

Algorithm 2 An outline of steps for CreateSnapshotMPT

Require: B_{header}, B_{body}
▹ prepare MPT version
Ensure: ROLLBACK(MPT,B_{header}.orphanHashes)
$S \leftarrow \emptyset$
$S \leftarrow \{\mathcal{T}_i : \forall \mathcal{T}_i \in B_{body} \nexists \mathcal{T}_j \in B_{body}$ s.t. $(\mathcal{T}_i.uid = \mathcal{T}_j.uid \wedge \mathcal{T}_i.t < \mathcal{T}_j.t)\}$;
$\mathcal{T}_{orphanes} \leftarrow$ getBlocksTransactions(B_{header}.orphanHashes)
$S \leftarrow \{\mathcal{T}_i : \forall \mathcal{T}_i \in \mathcal{T}_{orphanes}, \forall \mathcal{T}_b \in B_{body}, \nexists \mathcal{T}_j \in \mathcal{T}_{orphanes}$ s.t. $(\mathcal{T}_i.uid = \mathcal{T}_j.uid \wedge \mathcal{T}_i.uid \neq \mathcal{T}_b.uid \wedge \mathcal{T}_i.t > \mathcal{T}_j.t)\}$
MPT $\leftarrow S$ ▹ Extend MPT
▹ Save fingerprint of MPT version
B_{header}."Location root" $\leftarrow \mathcal{H}_R$(**MPT**)

Theorem 1. *For each block from GetRobustAccepted(G), the last position of an account queried from MPT has the completeness of query answer (property 4) .*

Proof. Property 1 guarantees absence of conflicting transactions. The properties of MPT ((ii)-(iii) above) ensure a unique tree root value. Property 3 guarantees a monotonically decreasing time from a block to referenced blocks. □

4.3.2. Account Location Tracking

We use MPT and a *location root* from a block header for most recent position of an object. We assume that the block headers are already available. A lightweight client tracks items by hashed account identifier (*uid*) that is a hashed from the public part of a key pair. For most recent location verification of an object, we consider the following steps:

(1) A lightweight client has a fresh state of the block-DAG.

(2) The client requests from the peers to get an *account state* by the predefined *uid* that stores the most recent position and a time-stamp.

(3) The request is handled on peer side by MPT authenticated data structure. Request result includes the account state and additional information for authorization. The

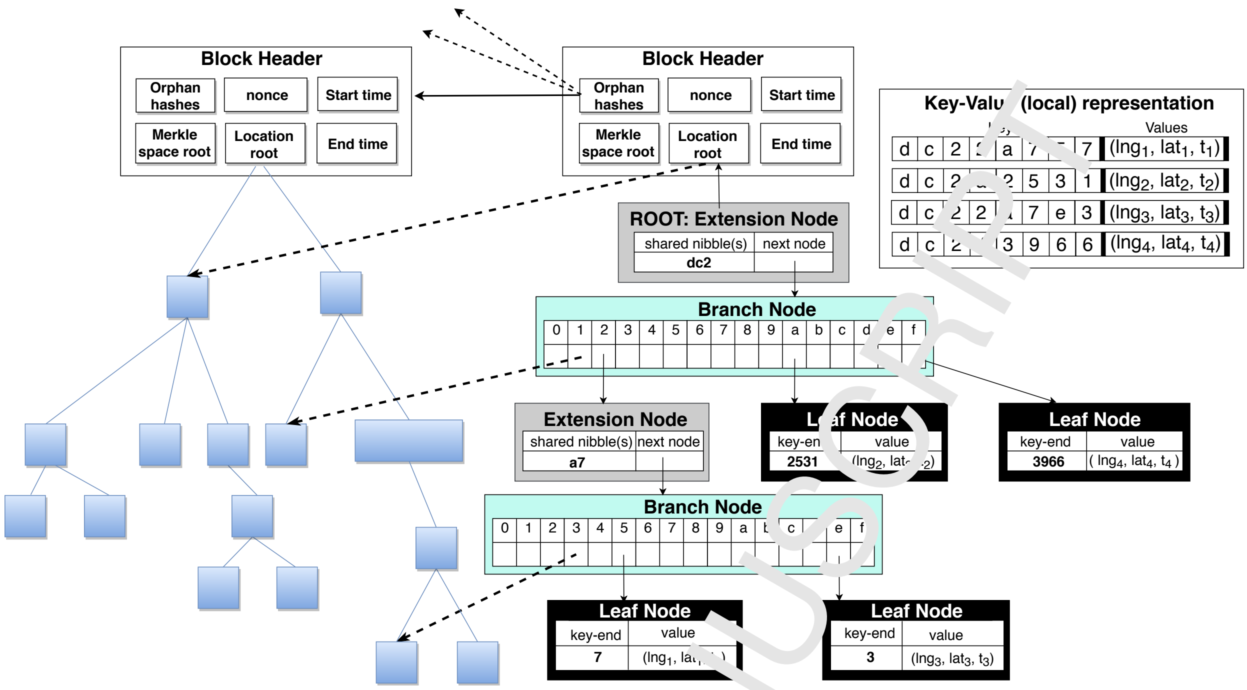


Figure 2: The last positions of objects related to accounts are represented by key-value pairs and encoded in Merkle Patricia-trie.

authorization information consists of an array of verification objects, VOs, reported by MPT in time-order and the block identifier of a B_{header} that includes *location root* hash-value corresponding to a fresh version of the MPT on the peer side.

(4) The client computes the 'Location root' of an object by hashing $\mathcal{H}$(obtained account state) with the VOs.

(5) The client locally verifies the result on local block-DAG image with the help of block identifier and Location root from the previous step. If the verification is successful, we have the verified most recent location along with a time-stamp for the client.

5. Spatio-temporal Queries on block-DAG

In this section, we present the queries for the data structures described in section 4. We first discuss temporal queries on block-DAG.

5.1. Temporal Queries on block-DAG

To enable a fast temporal search over the block-DAG, we introduce a temporal meta-information included in each block header. The temporal range query with a given time range β over bloc-DAG is the deterministic search procedure P over block-DAG topology $G(V, E)$ that we name as *Temporal Graph Search* (TGS). G has multiple source nodes s, which are recent nodes with a reference to the previous points, for each $v \in V$. The temporal range search (*searchtime*) is implemented in a breadth-first (BFS) manner, the steps of which are listed below:

(1) Given a time range $\beta = (start\_time, end\_time)$, start BFS from the tips of the $GetRobustAccepted(G)$.

(2) When reach a block-header B_{header} such that $[B_{header}.start\_time, B_{header}.end\_time] \cap [\beta.start\_time, \beta.end\_time] \neq \emptyset$, the B_{header} is included in the result set.

(3) The BFS runs until no new B_{header} occurs in range β and all the next $B_{header}.end\_time < \beta.start\_time$.

Theorem 2. *Temporal Graph Search procedure has the completeness of query answer (property 4) over GetRobustAccepted(G) $\subset$ G of block-DAG topology (G).*

Proof. Property 3 guarantees monotonically decreasing timestamps of $B_{header}.start\_time$ and $B_{header}.end\_time$ for each step of TGS procedure. The σ_β accepts blocks as partial intersection of time ranges. Therefore, the algorithm has the property 4, nonetheless, a tiny number of outline transactions will be in result set that leads to an unsound answer. □

5.2. Spatial Queries per Block

The spatio-temporal query is the combination of the two consequent procedure calls. The first part is the TGS procedure that results in a set of block headers $\mathcal{M}_\beta$. The second part is the spatial query over BSI of each block header. A general spatial search result is denoted as $\sigma(\mathcal{M}_\beta) = \{\sigma(BSI(B_{header})) \; \forall B_{header} \in \mathcal{M}_\beta\}$. We now present the standalone mechanism of spatial queries.

The BSI is the extension of Merkle kd-tree, where each internal node of the data structure is the final space point. In this work, we use a three dimensional BSI for the three distinct coordinates in the Cartesian coordinate system. With BSI, we are

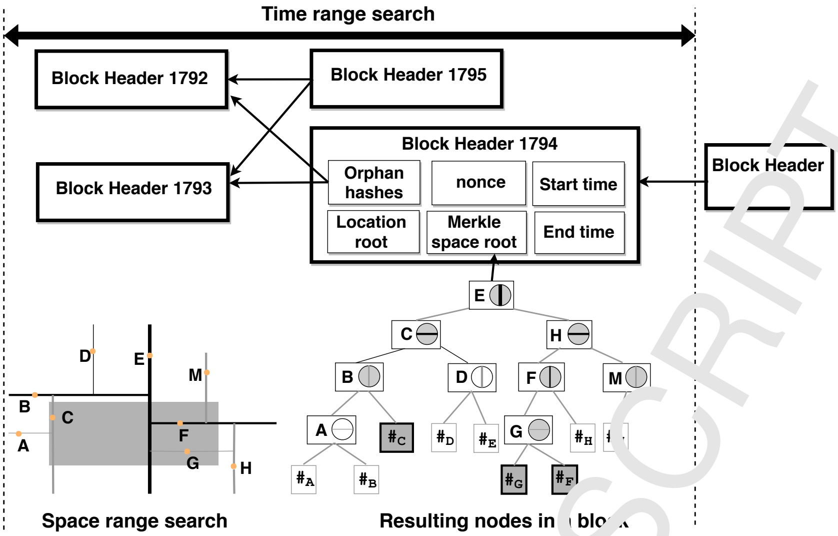


Figure 3: Spatio-temporal range search on block-DAG by TGS-BSI procedure: first, filter by temporal range procedure σ_β, and given time range β, apply space range query on each BSI by a given hyper-rectangle $q = \{x, y\}$.

able to handle efficient space query per block for the following types of queries: (i) *range query*, (ii) *k-NN query*, (iii) *bounded k-NN query*, and (iv) *ball-point query*, the details of which are presented below.

5.2.1. Range query

Range query, σ_q, is stated as follows: Given a dataset of points D, and a range $q = \{x\}$ where $x \in \mathcal{R}^d$, range query seeks each $s \in D$ that is located inside the hyper-rectangle constituted by q. The computational cost of a range query for the BSI of a particular block is $O(\sqrt{B_{size}} + k)$ where k is the number of result points.

Given a query hyper-rectangle q, the range search for each $B_{header} \in \mathcal{M}_\beta$ starts at the root and recursively traverses the tree pruning a subtree if its root does not intersect with q. An example of a spatio-temporal range query on block-DAG is shown in Figure 3. We show a 2D BSI for ease of illustration.

5.2.2. K-nearest neighbors

k-NN query, $\sigma_{k,q}$, is defined as follows: Given a dataset of points D, a scalar value k, and a query point $q = \{x\}$ where $x \in \mathcal{R}^d$, $\sigma_{k,q}(D)$ returns a subset of k points from D that are closest to q. The average computational complexity of a k-NN query for a block is $O(3B_{size}k)$ assuming that the number of dimensions of BSI is fixed (3D).

The k-NN algorithm maintains a priority queue to keep k closest points. The first k points are en-queued and the algorithm traverses down the tree skipping bounding boxes where is no chance to get a point closer than the points in k. Thus, the performance of the algorithm depends on quickly reaching nearby points. The final stage is to aggregate the top-k points from $\sigma_{k,q}(\mathcal{M}_\beta)$.

5.2.3. Bounded k-nearest neighbors

Bounded k-NN, σ_{q,r_b}, is defined as follows: Given a dataset of points D, a scalar value k, a scalar value of the bounding radius r_b, and a query point $q = \{x\}$ where $x \in \mathcal{R}^d$, $\sigma_{k,q,r_b}(D)$, a k-NN query returns a subset of points in D of size $\leq k$ that are closest to the q inside a hyper-sphere of radius r_b centered at q. The average case complexity of a bounded k-NN query is also $O(3B_{size}k)$ considering that the number of dimensions in BSI is fixed at 3D.

The query algorithm includes additional boundary to a maximum radius that creates a limiting hyper-sphere for exploration during the BSI tree traversal. Bounded k-NN maintains k current best points and search branches in BSI when they can't have points closer than any of the k current points, or if the radius is more than a boundary radius r_b. The final stage is to aggregate the top-k points from $\sigma_{k,q}(\mathcal{M}_\beta)$.

5.2.4. Ball-Point Query

Given a dataset of points D, radius r_b, and a point $q = \{x\}$ where $x \in \mathcal{R}^d$, a ball-point query, σ_{k,q,r_b}, seeks each $s \in D$ that is in a sphere of radius r_b centered at the point q. The average case computational complexity of a ball-point query average for a particular block is $O(3\sqrt{B_{size}} + k)$ where k is the number of answer points.

Given a bounding radius r_b, and a centering point q the ball-point query for each $B_{header} \in \mathcal{M}_\beta$ starts at the root and recursively traverses the tree whilst pruning a bounding box that does not intersect the hypersphere of radius $= r_b$ centered at q.

6. Experimental Evaluation

We now present a detailed evaluation of spatio-temporal queries over blockchain. We first present the random graph model for block-DAG generation. Then, we describe the experimental setup that is followed by the evaluation results and discussion.

Algorithm 3 An outline of the steps for GenBlockDAG routine

Require: $\mathcal{T}_n, \mathcal{T}_r, \alpha, \mathcal{D}, B_{size}$

$\mathcal{T}_{rlist} \leftarrow \text{gen}(\mathcal{N}\left(\mathcal{T}_r, \alpha^2\right), \mathcal{T}_n)$ ▷ list of various transaction rates per second

$(V, E) \leftarrow (\text{genesis block}, \emptyset)$ ▷ initialize vertices and block references

$\omega \leftarrow 0$ ▷ remainder of transactions from previous chunk

for each chunk c of size $\mathcal{D}$ from $\mathcal{T}_{rlist}$ **do**

  $\mathcal{B} \leftarrow \texttt{round}((\texttt{sum}(c) + \omega)/B_{size})$ ▷ number of unconnected blocks while $\mathcal{D}$

  $E \leftarrow$ `(connect all` $\mathcal{B}$ `to orphans of` V`)`

  $V \leftarrow \mathcal{B}$

  $\omega \leftarrow (\texttt{sum}(c) + \omega) \texttt{ mod } B_{size}$

return (V, E)

6.1. Random Graph Model for block-DAG

The synthetic topology structure of block-DAG is generated by Algorithm 3. The following parameters are considered for block-DAG generation: network delay $\mathcal{D}$, number of transactions per second $\mathcal{T}_r$, total number of transactions $\mathcal{T}_n$, block size B_{size}, and a standard deviation α for a normal distribution centered at $\mathcal{T}_r$. The generation model is based on the following assumptions: (i) two honest blocks created at the same time are not mutually reachable, and (ii) no more than n honest unconnected blocks can be created at the same time where $n = \mathcal{D} * B_{rate}$ and B_{rate} is block creation rate.

6.2. Setup

For the experiments, we consider a spatio-temmporal dataset, 'Pokeman Go'. The dataset includes 18732 records where each record contains latitude, longitude, timestamp and a Pokemon type. We assume a constant and intensive transaction flow. In order to simulate a computing extensive environment, we replicate the dataset 300 times. Thereafter, we generate a block-DAG with the assumption that the network delay is typically 3 seconds, $\mathcal{T}_r = 60$, $B_{size} = 50$, $\alpha^2 = 3$, and $\mathcal{T}_n$ is 300 times the Pokemon Go dataset size. As the records are duplicated, the timestamps are updated in accordance with the original dataset.

For query performance measurement, we repeat each experiment ten times at each testing point to get a descriptive and robust median point. The implementations are in Python and the results are obtained on a Linux machine with Intel Core i7-6700 CPU 3.40GHz processor, and 16 GB RAM. The query times reported in the following text are in 000's milliseconds, e.g. we say '3 units' when we mean 3000 milliseconds.

6.3. Spatio-temporal Query Performance Analysis

We evaluate the performance of spatio-temporal queries (Section 5): (i) for temporal range upto 2 weeks, (ii) for increasing number of hours over a block-DAG with $B_{size} = 40$, and (iii) $B_{size} = 100$, (iv) performance change for a fixed temporal range, 2 hours, for increasing number of transactions included in the block-DAG. In another set of experiments, we evaluate query performance of spatial queries: (i) for varying values of k for k-NN query, (ii) varying bounding radius r_b for bounded k-NN query, and (iii) ball-point query.

In another set of experiments, we evaluate the queries under study for spatio-temporal data handling. As the scan operation for a blockchain is a brute-force iteration over every block, a temporal search must go through all the entries of a block-header due to the absence of time-stamp information within. For performance comparison, we consider the following: (a) scan operation that firstly filters transactions by time and then by space, SCAN time-space, and vice versa, SCAN space-time, and (b) our proposed time graph search, TGS, and blockchain space index, BSI, for spatio-temporal querying, TGS-BSI.

6.3.1. Block size

Block size B_{size} is the number of transactions in a block and this information is stored in the block header. We report median query time in units (000′s milliseconds) as we increase B_{size} from 30 to 110. The observations of Figure 4 state that an increase in block-size improves query performance for TGS-BSI as more pruning occurs in each block that reduces the total number of observations to be considered. As the synthetic block-DAG is for a two week period, the spatio-temporal query for two weeks temporal range leads to a spatial query.

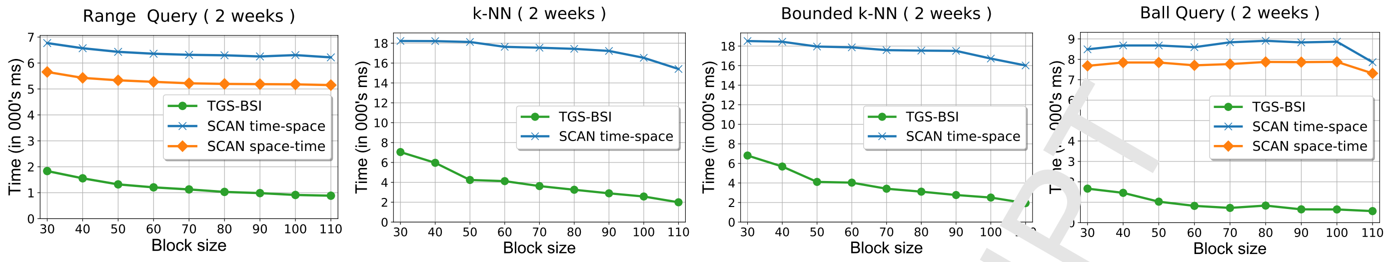
We observe in Figure 4 that the query time decreases when the block size B_{size} increases. The running times for k-NN and bounded k-NN queries are shown in Figure 4 (ii)–(iii). For $B_{size} = 30$, the TGS-BSI query time is 7.5 units and it decreases to 1 unit for $B_{size} = 100$, while the k-NN scan queries require 17.5 units. The overall query performance remains stable for range and ball queries, e.g. for TGS-BSI queries, running time remains below 2 units whereas for scan queries the running time is 5 units and upwards.

It can be observed that the block size is an important parameter for k-NN query performance and as the block size increases, pruning is more effective for TGS-BSI. Nonetheless, in real applications, we expect that the average B_{size} will typically remain below 100.

6.3.2. Temporal range

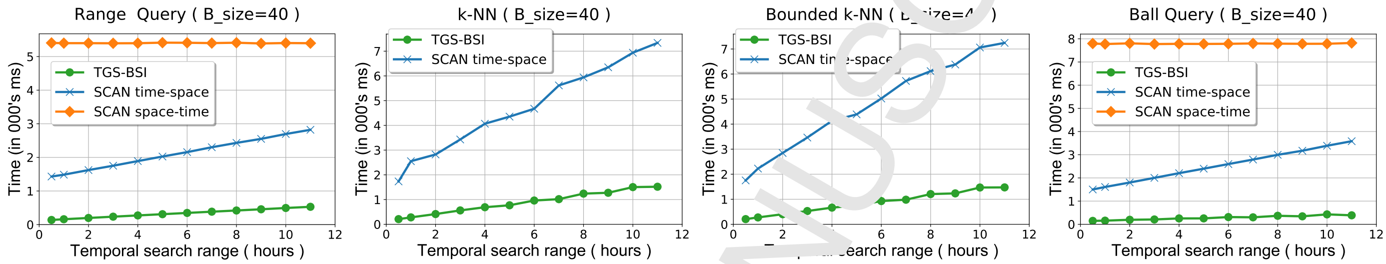
We now report time range β results for spatio-temporal queries. Scan operation handles temporal queries by scanning timestamps for each transaction whereas TGS-BSI is a breadth-first search over the block-DAG that stores timestamp attribute on each node. We also consider more realistic short temporal ranges from 0.5 hours to 12 hours. The performance evaluations are shown for two cases: (i) $B_{size} = 40$, Figure 5, and (ii) $B_{size} = 100$, Figure 6.

For the experiments, we increase β from 0.5 hours to 12 hours and observe that the running time increases linearly with β, however, time-space scan operation is noticeably time consuming. The results of TGS-BSI for k-NN and bounded k-NN queries for two variants of experiments presented in Figure 5 (ii)–(iii), and Figure 6 (ii)–(iii), show that for $B_{size} = 100$ the running time is almost half compared to the running time for $B_{size} = 40$. However, the overall time for TGS-BSI remains stable for range and ball-point queries.



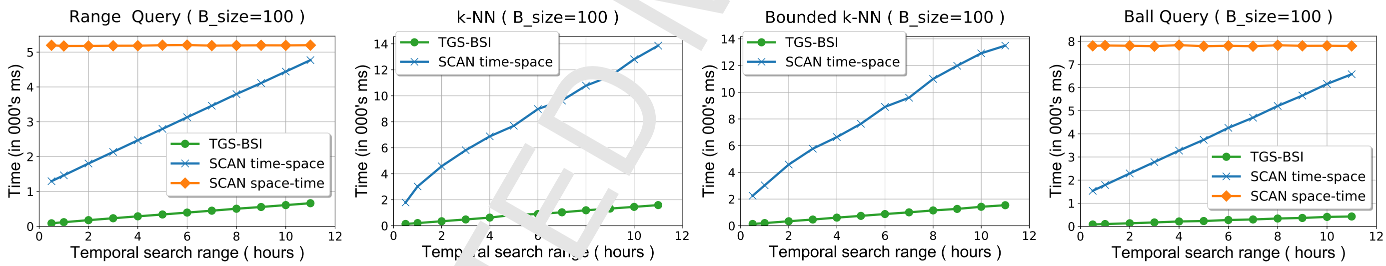
(i) Block size increase (σ_q). (ii) Block size increase ($\sigma_{k,q}$). (iii) Block size increase (σ_{k,q,r_b}). (iv) Block size increase (σ_{q,r_b}).

Figure 4: Performance evaluation for increasing B_{size} from 30 to 110 over a 2 week synthetic block-DAG.
Query parameters: β =2 weeks, $q = (\phi, \lambda) = (22.6, 114)$, $k = 15$, $r_b = 15$.



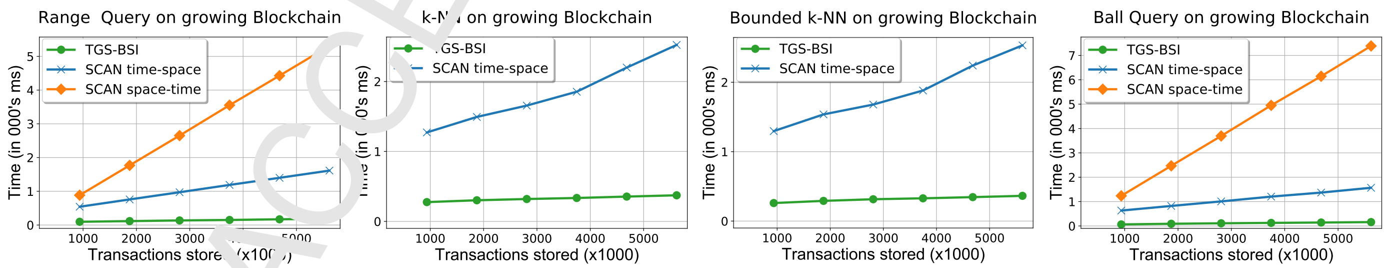
(i) Time range extend (σ_q). (ii) Time range extend ($\sigma_{k,q}$). (iii) Time range extend (σ_{k,q,r_b}). (iv) Time range extend (σ_{q,r_b}).

Figure 5: Performance evaluation for increasing time range β from 0.5 hours to 12 hours over a 2 week synthetic block-DAG with $B_{size} = 40$.
Query parameters: $q = (\phi, \lambda) = (22.6, 114)$, $k = 15$, $r_b = 15$.



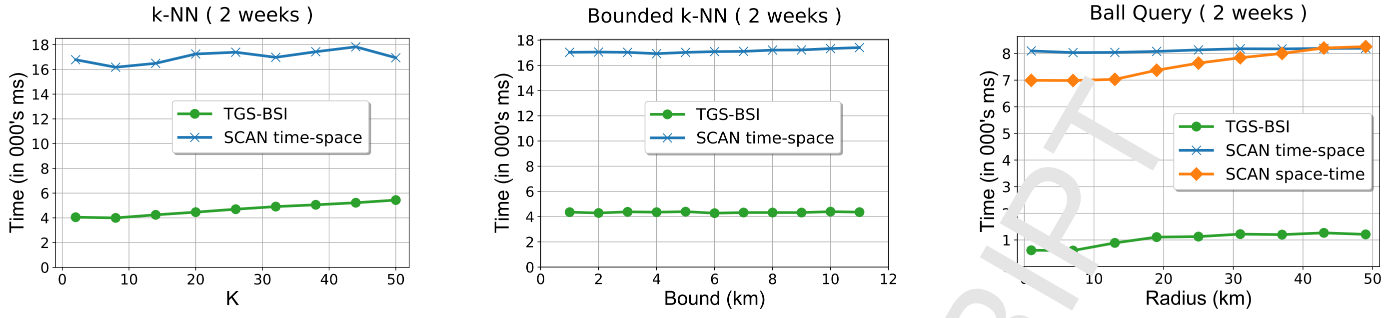
(i) Time range extend (σ_q). (ii) Time range extend ($\sigma_{k,q}$). (iii) Time range extend (σ_{k,q,r_b}). (iv) Time range extend (σ_{q,r_b}).

Figure 6: Performance evaluation for increasing time range β from 0.5 hours to 12 hours over a 2 week synthetic block-DAG with $B_{size} = 100$.
Query parameters: $q = (\phi, \lambda) = (22.6, 114)$, $k = 15$, $r_b = 15$.



(i) Growing $\mathcal{T}_n$ (σ_q). (ii) Growing $\mathcal{T}_n$ ($\sigma_{k,q}$). (iii) Growing $\mathcal{T}_n$ (σ_{k,q,r_b}). (iv) Growing $\mathcal{T}_n$ (σ_{q,r_b}).

Figure 7: Performance evaluation for increasing $\mathcal{T}_n$ from 10^6 to $6 * 10^6$ transactions over a 2 week synthetic block-DAG.
Query parameters: β =2 weeks, $q = (\phi, \lambda) = (22.6, 114)$, $k = 15$, $r_b = 15$.



(i) Increase K neighbours ($\sigma_{k,q}$). (ii) Increase bound in r_b (σ_{k,q,r_b}). (iii) Increase radius r_b (σ_{q,r_b}).

Figure 8: Performance evaluation over a 2 week synthetic block-DAG. We vary number of neighbors k from 1 to 50 for k-NN search, bounding radius r_b from 1 to 11 km for bounded k-NN search and bounding radius from 1 to 50 for ball-point query. Query parameters: β =2 weeks, $q = (\phi, \lambda) = (22.6, 114)$, $k = 15$, $r_b = 15$.

6.3.3. *Number of transactions stored*

We also test the scalability of the algorithms we study for increasing number of transactions $\mathcal{T}_n$ stored in a block-DAG. We report the running times as we increase $\mathcal{T}_n$ from 10^6 to $6*10^6$ in a synthetic block-DAG. The size of block-DAG and the running time results are shown in Figure 7 (i)–(iv). It can be observed that TGS-BSI performs significantly better than scan-range operations. As growing number of transactions are accepted in a decentralized ledger, it becomes clear that TGS-BSI query performance is encouraging as it outperforms scan-range search by orders of magnitude.

6.3.4. *Query parameters*

In addition to the above experiments, we study query performance in a 2 week temporal range, Figure 8, for the following three parameters: number of nearest neighbors k for k-NN queries, Figure 8(i), bounding radius r_b for bounded k-NN, Figure 8(ii), and increasing bounding radius for ball-point query, Figure 8(iii).

We report running times as we increase k from 1 up to 50 for the k-NN query in Figure 8(i). It can be observed that the query time increases linearly with the increasing values of k. TGS-BSI outperforms scan-range for k-NN query. Similar trends can be observed for increasing r_b for bounded k-NN query, Figure 8(ii). We also observe that the overall time for TGS-BSI remains stable even for harder instances.

In summary, block-DAG based TGS-BSI is a promising approach that provides significant speedup for spatio-temporal queries on blockchain. As a block-DAG is a significantly compact data structure and TGS-BSI is customized to answer queries under study, we conjecture that the performance of TGS-BSI is bound to improve for more realistic large datasets. Further, block-DAG is customizable for specific applications and TGS-BSI can be enhanced to work with additional query types.

7. Conclusion and Future Work

We have presented efficient spatio-temporal data storage and query processing in public decentralized ledgers that maintain integrity through cryptographically signed history in block-DAG and enable efficient spatio-temporal queries without additional local indexing. We also presented a protocol for authenticated tracking of a set of entities based on the public key or hashed account identifier. We considered four types of queries in this work and reported on performance evaluation of *range query*, *k-NN query*, *bounded k-NN query*, and *ball-point query*. The experimental results demonstrated the effectiveness and applicability of the solution for real-life applications.

As this is one of the initial studies on the topic, a number of open directions remain. For example, a study on client-peer communication seems an interesting idea as it is crucial to enable lightweight clients at the edge of the network to access the data from block-DAG peers in an effective way. Further, due to the presence of malicious peers, *query authentication* should also be an important consideration. Additionally, the extension of spatio-temporal queries to more sophisticated queries, for instance, 'find all pairs of points whose distance is at most k within a time bounds', skyline k-NN query, reverse k-NN, etc., is also an interesting direction to explore. Another interesting idea is to extend query computation to decentralized Map-Reduce framework and support OLAP like queries [39, 40] on a large number of blocks.

Acknowledgments. This work was partially supported by the CAS Pioneer Hundred Talents Program, China [grant number Y84402, 2017], and CAS President's International Fellowship Initiative, China [grant number 2018VTB0005, 2018].

References

[1] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, 2008.
[2] M. Muzammal, Q. Qu, B. Nasrulin, Renovating blockchain with distributed databases: An open source system, Future Generation Computer Systems 90 (2019) 105–117.
[3] T. M. Fernández-Caramés, P. Fraga-Lamas, A review on the use of blockchain for the internet of things, IEEE Access 6 (2018) 32979–33001.
[4] Y. Lewenberg, Y. Sompolinsky, A. Zohar, Inclusive block chain protocols, in: FCDS, pp. 528–547.
[5] Q. Qu, C. Chen, C. S. Jensen, A. Skovsgaard, Space-time aware behavioral topic modeling for microblog posts, IEEE Data Eng. Bull. 38 (2015) 58–67.
[6] G. Becker, Merkle signature schemes, merkle trees and their cryptanalysis, 2008.
[7] A. Trouw, M. Levin, S. Scheper, The XY Oracle Network: The Proof-of-Origin Based Cryptographic Location Network, Technical Report, 2018.

[8] E. Farris, F. M. I. William, System and method for controlling drone delivery or pick up during a delivery or pick up phase of drone operation, 2016. US Patent App. 14/814,501.

[9] K. Dorling, J. Heinrichs, G. G. Messier, S. Magierowski, Vehicle routing problems for drone delivery, IEEE Trans. Systems, Man, and Cybernetics: Systems 47 (2017) 70–85.

[10] I. Nurgaliev, M. Muzammal, Q. Qu, Enabling blockchain for efficient spatio-temporal query processing, in: WISE, pp. 36–51.

[11] M. A. Azad, S. Bag, F. Hao, Privbox: Verifiable decentralized reputation system for the on-line marketplaces, Future Generation Computer Systems 89 (2018) 44–57.

[12] Z. Ma, M. Jiang, H. Gao, Z. Wang, Blockchain for digital rights management, Future Generation Computer Systems 89 (2018) 746–764.

[13] K. Leng, Y. Bi, L. Jing, H.-C. Fu, I. Van Nieuwenhuyse, Research on agricultural supply chain system with double chain architecture based on blockchain technology, Future Generation Computer Systems 86 (2018) 641–649.

[14] A. Reyna, C. Martín, J. Chen, E. Soler, M. Díaz, On blockchain and its integration with iot. challenges and opportunities, Future Generation Computer Systems 88 (2018) 173–190.

[15] B. Nasrulin, M. Muzammal, Q. Qu, A robust spatio-temporal verification protocol for blockchain, in: WISE, pp. 52–67.

[16] B. Nastrulin, M. Muzammal, Q. Qu, ChainMOB: Mobility analytics on blockchain, in: MDM, pp. 556–557.

[17] A. Karpi, R. Sternfeld, et al., Spatio-temporal data processing systems and methods, 2017. US Patent 9,734,220.

[18] G. Niemeyer, Geohash, 2008.

[19] X. Cao, L. Chen, G. Cong, C. S. Jensen, Q. Qu, A. Skovsgaard, D. Wu, M. L. Yiu, Spatial keyword querying, in: ER, pp. 16–29.

[20] Q. Qu, S. Liu, B. Yang, C. S. Jensen, Integrating non-spatial preferences into spatial location queries, in: SSDBM, pp. 8:1–8:12.

[21] Q. Qu, S. Liu, B. Yang, C. S. Jensen, Efficient top-k spatial locality search for co-located spatial web objects, in: MDM, pp. 269–278.

[22] A. Guttman, R-trees: A dynamic index structure for spatial searching, in: SIGMOD, pp. 47–57.

[23] X. Xu, Rt-tree: An improved r-tree index structure for spatiotemporal databases, in: Proc. of the 4th Intl. Symposium on Spatial Data Handling.

[24] Y. Theodoridis, M. Vazirgiannis, T. K. Sellis, Spatio-temporal indexing for large multimedia applications, in: ICMCS, pp. 441–448.

[25] R. P. Mahapatra, P. S. Chakraborty, Comparative analysis of nearest neighbor query processing techniques, Procedia Computer Science 57 (2015) 1289–1298.

[26] D. Papadias, Y. Tao, G. Fu, B. Seeger, Progressive skyline computation in database systems, ACM Trans. Database Syst. 30 (2005) 41–82.

[27] F. Li, K. Yi, M. Hadjieleftheriou, G. Kollios, Proof-infused streams: Enabling authentication of sliding window queries on streams, in: VLDB, pp. 147–158.

[28] K. Mouratidis, D. Sacharidis, H. Pang, Partially materialized digest scheme: an efficient verification method for outsourced databases, VLDB J. 18 (2009) 363–381.

[29] L. Hu, W. Ku, S. Bakiras, C. Shahabi, Spatial query integrity with voronoi neighbors, IEEE Trans. Knowl. Data Eng. 25 (2013) 863–876.

[30] A. D. Fox, C. N. Eichelberger, J. N. Hughes, S. Lyon, Spatio-temporal indexing in non-relational distributed databases, in: ICBD, pp. 291–299.

[31] O. Procopiuc, P. K. Agarwal, L. Arge, J. S. Vitter, Bkd-tree: A dynamic scalable kd-tree, in: SSTD, pp. 46–65.

[32] J. N. Hughes, A. Annex, C. N. Eichelberger, A. Fox, A. Hulbert, M. Ronquest, Geomesa: a distributed architecture for spatio-temporal fusion, in: Geospatial Informatics, Fusion, and Motion Video Analytics V, volume 9473, International Society for Optics and Photonics, p. 94730F.

[33] I. Komargodski, M. Naor, E. Yogev, Collision resistant hashing for paranoids: Dealing with multiple collisions, in: Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 162–194.

[34] C. U. Martel, G. Nuckolls, P. T. Devanbu, M. Gertz, A. Kwong, S. G. Stubblebine, A general model for authenticated data structures, Algorithmica 39 (2004) 21–41.

[35] J. Xu, L. Wei, Y. Zhang, A. Wang, F. Zhou, C. Gao, Dynamic fully homomorphic encryption-based merkle tree for lightweight streaming authenticated data structures, J. Network and Computer Applications 107 (2018) 113–124.

[36] G. Wood, Ethereum: A secure decentralised generalised transaction ledger, Ethereum Project Yellow Paper 151 (2014) 1–32.

[37] F. Corp., Foam: The consensus driven map of the world, 2018.

[38] G. E. Mackey, Efficient Nearest Neighbor Searches in N-ABLE TM, Technical Report, Sandia National Laboratories, 2010.

[39] Q. Qu, F. Zhu, X. Yan, J. Han, P. S. Yu, H. Li, Efficient topological OLAP on information networks, in: DASFAA, pp. 389–403.

[40] Q. Qu, S. Liu, F. Zhu, C. S. Jensen, Efficient online summarization of large-scale dynamic networks, IEEE Trans. Knowl. Data Eng. 28 (2016) 3231–3245.

QIANG QU is an associate professor at Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences (CAS), and the director of Guangdong Provincial R&D Center of Blockchain and Distributed IoT Security. He is a candidate of the CAS Poineer Hundred Talents Program. He received the MSc degree in computer science from Peking University and the PhD degree from Aarhus University. His current research interests are in data-intensive applications and systems, focusing on efficient and scalable algorithm design, blockchain, data sense-making, and mobility intelligence. His recent research has been published in leading journals and international conferences, including ACM SIGMOD, VLDB, AAAI, the IEEE transactions on Data Engineering, the IEEE Transactions on Intelligent Transportation Systems, and Information Sciences. He was TPC member of several prestige conferences, and he chaired workshops in VLDB 2018, VLDB 2017, ICDM 2015, and APWEB-WAIM 2017 on mobility analysis.

ILDAR NURGALIEV received the M.Sc. degree in Data Science from the Innopolis university in 2017, Tatarstan, Russia. Currently, he is a PhD student in Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China. He was a student at CERN OpenLab in 2016, Geneva, Switzerland. His research interests include blockchain and social network analysis.

MUHAMMAD MUZAMMAL is an associate professor at the Department of Computer Science, Bahria University, Islamabad, Pakistan, a visiting associate professor under CAS President's International Fellowship Initiative (PIFI) at the Centre of Big Mobile Intelligence, Shenzhen Institutes of Advance Technology (SIAT), Chinese Academy of Sciences (CAS), Shenzhen, and a Vice Director of Guangdong Provincial R&D Centre of Blockchain and Distributed IoT Security, Guangdong, China. He received the PhD degree from University of Leicester, UK, in 2012. Before, that he was a software analyst at LMKR. He received the master's and bachelor's degrees from FAST-NU, Pakistan, and IIUI, Pakistan, in 2007 and 2005, respectively. His research interests are in large scale data mining including algorithm design and mobility data mining. Recently, he is interested in blockchain technology with a focus on decentralized systems and mining.

CHRISTIAN S. JENSEN is a Professor of Computer Science at Aarhus University, Denmark, and he was previously at Aalborg University for two decades. He recently spent a 1-year sabbatical at Google Inc., Mountain View. His research concerns data management and data-intensive systems, and its focus is on temporal and spatio-temporal data management. Christian is an ACM and an IEEE fellow, and he is a member of the Royal Danish Academy of Sciences and Letters and the Danish Academy of Technical Sciences. He has received several national and international awards for his research. He is currently vice-chair of ACM SIGMOD and an editor-in-chief of The VLDB Journal.

JIANPING FAN, since 2006, served as Director of Shenzhen Institutes of Advanced Technology, CAS. His research interests include high performance computing, Grid computing, and computer architecture. He designed and developed a series supercomputer including Dawning I, Dawning 1000, Dawning 3000 and Dawning 4000. He has been a Principal Investigator of 21 projects funded by both government and industries. Based on his research work, he has published more than 118 papers and acquired 23 pending or issued patents. He has received many awards, such as outstanding award of CAS science and technology progress, first price of national science and technology progress, first price of Beijing science and technology progress, the outstanding young scientist of CAS, and the national talent of New Century talent program. He has actively participated in various professional activities. He served as editor of journal of integration technology, joint professor in Beijing Jiaotong University, council member of china digital library consultant committee, Advisor of 11th five years science and technology development plan of ministry of information industry, general chair of HPC China2007,GCC2008,PAKDD 2011, CNCC 2011, the The mayor award of Shenzhen 2013, Fellow of CCF 2014. Vice president of the Institute of talent research 2016.

Highlights

The highlights of this work are as follows:

- One of the first attempts to formalize the spatio-temporal blockchain query processing
- A note on the limitations in current blockchain systems and a novel solution for spatio-temporal query processing on blockchain
- Introduction of a block-DAG based novel index traversal algorithm, TGS-BSI, to handle spatio-temporal queries on a block-DAG.
- A detailed experimental evaluation to demonstrate the effectiveness of the solution