When Directory Design Meets Data Explosion: Rethinking Query Performance for IoT

Luoyao Hao and Henning Schulzrinne

Department of Computer Science, Columbia University, New York, NY, USA
Email: {lyhao, hgs}@cs.columbia.edu

***Abstract*—As IoT services scale up from single homes to smart cities, directories and mapping services are needed to manage potentially millions of devices. However, directory service providers will likely struggle to accommodate the increasing number of IoT devices, made more challenging by their heterogeneous metadata and the large volume of queries. One of the critical challenges, the high heterogeneity of IoT, is being addressed by a working standard of W3C, which formalizes a physical or virtual device as a formatted Thing Description (TD).**

We propose a local directory service architecture with a series of design requirements. With a focus on query performance, we build a proof-of-concept system to store metadata of IoT devices as TDs in terms of the working standard. A Raspberry Pi is configured to investigate the query performance of relational database and non-relational database as the classic choices for internal directories. Evaluation results demonstrate that compared with relational database, non-relational database can achieve 2.9 times higher resilience on property query and 2.35 times faster processing on spatial query, with mild loss on aggregation query.

I. Introduction

Internet of Things (IoT) consists of physical objects and devices connecting, communicating, and interacting with one another through the network. The arrival of 5G era would foster IoT into a new stage. With the phenomenal growth in IoT technologies and devices, future mobile network is foreseen to accommodate additional billions of IoT connections by 2024 [1], making computing and connectivity more pervasive in our day-to-day lives.

In line with this trend, IoT will continue to bring huge business opportunities to vertical industries and social sectors in a bunch of fields such as smart home, smart city, healthcare, and public safety. For further supporting countless access requests from massive user base, performance-guaranteed directory services are required to store IoT metadata including identifier, name, address, security, type, etc [2]. IoT directory services provide look-up functions for users as well as applications to retrieve these metadata.

While the development of IoT has brought tremendous convenience and business opportunities, the untoward effects gradually appear for directory services. Both storage efficiency and query performance are facing challenges [2], [3]. Since metadata of IoT devices are various and have diverse entries depending on their manufacturers, storing large amounts of heterogeneous IoT metadata to the same directory would either waste lots of spaces (with relational database) or lead to a peril facing malicious query (with non-relational database). Besides, pre-configuring a device to a directory is infeasible due to the mobility of IoT. Similar factors also restrain directories from expeditious responding to users' queries. Thus, if service providers fail to stem the heterogeneity and focus on query performance soon, IoT directory will come under enormous pressure, regardless of how popular the service look.

There have been a few pioneering works that discuss the metadata standardization and directory design of IoT. W3C [4] is working on a standard Thing Description (TD) of IoT. TD contains metadata and interaction affordances about the device itself and is an entry point to be mounted upon the internet. Despite of remarkable standardization work, describing metadata is fundamental but not enough for directory design. Scalable directory architecture is designed to dynamically assign network resources and updates replicas in [2], [5]. While in addition to the standardization issue, more attention on database and data format, with the analysis in our work, would potentially improve the query performance. Query performance for IoT application is discussed in [6], [7], but their targeted scenarios are not directory services.

Correspondingly, we focus on query performance especially for IoT directory, given insufficient work and unavoidable challenges in this area. We investigate the initial concerns for IoT directory design from the perspective of service providers. To get rid of the burdensome structure and concentrate on query performance, we propose the local directory architecture, which is an abstraction of the huge global directory model. For the standardization concern, we adopt W3C TD format with slight modification to describe our metadata. We then build a proof-of-concept system that consists of data layer, setting layer, and test layer. Based on it, we evaluate the performance of property query, aggregation query, and spatial query for both MongoDB and MySQL databases that are selected as the representatives of relational and non-relational databases. We also measure CPU utilization and the impacts of TD's flexibility and polygon size. Additionally, we compare the efficiency of the metadata description frameworks, JSON-LD and RDF formats, using SPARQL queries. Although most cases support the benefits of non-relational database, we do measure a boundary beyond which using relational database achieves better results. Thus, we argue that service providers should estimate and measure the possible queries based on different categories before making a wise choice.

Our contributions in this paper are summarized as follows.

978-1-7281-5628-6/20/$31.00 © 2020 IEEE

- We analyze the design issues for IoT directory services.
- We propose a local directory architecture for future IoT related services and implement a proof-of-concept system adopting a W3C working standard.
- Extensive performance evaluation is conducted based on different types of queries, databases, and data formats.

The rest of this paper is organized as follows: Section II provides the considered system model and related issues of IoT directory services. Section III introduces the design and implementation of our prototype. Section IV analyzes the query performance based on different databases and date formats. Section V gives the related work. Finally, Section VI concludes the paper.

II. System Model

In this section, we introduce the local directory service architecture and discuss two key points that affect query performance: data format and database selection.

A. Directory Architecture

Future IoT directory service would allow users or IoT applications to register the profiles of devices and look them up. The profiles are considered as standard metadata descriptions. Directory services provide functions to process data items using database-like CRUD operations with a focus on query performance. Since directory services are highly divergent, optimization of the overall query performance should make the most of different query types.

Fig. 1 depicts a local directory architecture diagram on top of standard IoT description. In a local network, a discovery agent runs on an edge computation node or an IoT gateway, which collects metadata of IoT devices as well as external environment information (e.g. location, temperature, etc.) and compiles them as TDs. Those TDs are then transmitted to a nearby directory consisting of a database and three modules named indexer, publisher, and resolver. The indexer receives TDs, indexes them in real-time, and stores the indexed data to the Database. The publisher is responsible for pushing data to other related directories or synchronizing data among other directories in the same hierarchy. As all the validated data stored in the database, the resolver can interact with users or applications by resolving their various queries and responding to them after extracting the targeted results from the database.

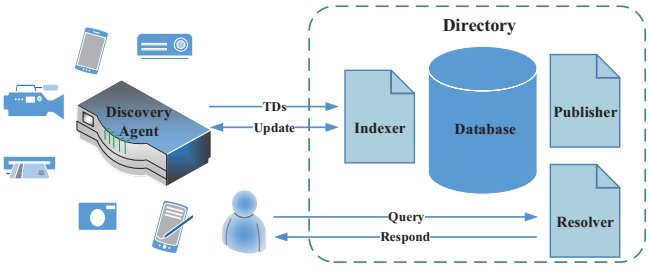


Fig. 1. Local directory service architecture.

Needless to say, a directory architecture can be fairly complicated generalized upon the local service architecture. Scaling the local directory architecture derives scalable and hierarchical geo-distributed directory architecture. Imagine a global directory service architecture with hierarchical directories, relevant challenges [2], [8], [9] such as ACID transactions, data aggregation, name services, cross-domain access control, are essential to address, but beyond the scope of this paper. Local directory model works very well for our purpose of analyzing query performance.

B. Data Description Format

A TD can be either presented in JavaScript Object Notation for Linked Data (JSON-LD) or in Resource Description Framework (RDF) formats.

JSON-LD is initially designed to present linked data of interoperable web resources over web-based programming models. JSON-LD stores linked data in JSON-based storage engines, which enables existing JSON files to be easily interpreted to JSON-LD. In addition, since JSON-LD is completely compatible with JSON, the large number of well-configured data, methods, and libraries for JSON can be reused without any extra efforts. A sample thing description is shown in Listing 1. This TD describes an IoT object titled "smart_light" with unique id ending with "1234". It defines basic security scheme for the object and one of the properties shows the status of object. Note that for space constraints, it is just an example with other properties, actions, events, and some default values intentionally omitted.

```
{
  "@context": "https://.../td/v1",
  "id": "urn:dev:wot:com:...:1234",
  "title": "smart_light",
  "securityDefinitions": {
    "basic_sc": {"scheme": "basic"}
  },
  "security": ["basic_sc"],
  "properties": {
    "status": {
      "type": "string",
      "forms": [{"href": "http:...."}]
    }
  },
  "actions": {...},
  "events": {...}
}
```

Listing 1. An example TD file with simplified fields and values.

Similarly, RDF is also a description language for representing metadata about web resources based on URI and XML [10]. The metadata of web resource could be title, author, redirection, etc. With the development of internet, the concept of web resource has been soon generalized to nearly all kinds of information identified on the web. RDF is based on the idea of identifying things using URIs and describing resources in terms of simple properties and corresponding values. RDF is represented in triplets: *(subject, predicate, object)*. For example, $< https : //xxx//1234 >< https :$

$//www.w3.org/2019/wot/td\#title > \text{“}smart\_light\text{”}$ indicates that a smart light with $id = 1234$, whose title is “smart_light”. Apparently, a RDF file can be easily converted to JSON-LD format, and vice versa.

Based on the compatibility and additional performance tests in Section IV-B, we set JSON-LD as our default data format in the following sections.

C. Database Selection

Database selection is another key concern for establishing efficient directory services. The debates between SQL and NoSQL databases remain a critical issue in IoT environment. Both SQL and NoSQL databases have shown their fortes in particular aspects. However, neither of them would completely supersede the counterpart considering scalability, flexibility, and database maturity.

The number of devices in a single local directory is anticipated not larger than several million, since IoT directories are often distributed based on geographical location. A single directory may store metadata of IoT devices from a building size to a city size. Thus, both vertical scalability favorable by SQL database and horizontal scalability favorable by NoSQL database are potentially needed in specific situations.

The speed of data retrieval and the flexibility of data storage often considered as an inevitable trade-off. NoSQL beats SQL in flexibility, whereas SQL generally reveals higher data processing capability. Moreover, the standardization of IoT description injects new opportunity to utilize SQL database, since the fields of data will be formalized. For directories with standard metadata especially, although NoSQL seems more flexible to store and arrange heterogenous IoT data, rethinking query performance of database deserves more consideration in order to provide fast look-up functions.

In addition, SQL is an experienced technology and multiple functional suites are developed to address potential issues. Some security issues such as authentication and data confidentiality are incorporated in SQL. On other hand, such security features are yet to be addressed in NoSQL [7]. Similarly in spatial query, only a few NoSQL functions are generated compared with that provided by SQL. Security and geographic features are critical in IoT applications. A secured communication channel or a suite of well-defined interfaces are occasionally deterministic to deliver services.

III. Prototype Implementation

In this section, we introduce the prototype implementation and related configuration.

To realize the local directory architecture and investigate query performance upon it, we build a proof-of-concept system where we use Raspberry Pi to connect to the directory server remotely. Tab. I and Tab. II show the configuration of the directory server and Raspberry Pi respectively. The core component of the directory is implemented as a database server storing 10,000 records, which is expected to be a regular size for a local directory. Each record is a TD for describing a physical or virtual IoT device. The directory enables manual registration for devices and metadata retrieval with queries by property or location. Since current devices do not hold standard description files, we generate mock metadata as the source of records based on the working standard. The generated TDs are validated in [11]. Required fields inside a record are: @context, id, title, securityDefinitions, security, properties, actions, and events.

TABLE I
Configuration of Directory Server

Category	Specification
CPU	Intel CORE i7-8750, 6 cores, 2.2GHz
System	Ubuntu 18.04.2 LTS
Storage	8GB RAM, 256GB Hard Disk
Database	MySQL 5.7.27 [12], MongoDB 3.6.3 [13]

TABLE II
Configuration of Raspberry Pi

Category	Specification
CPU	ARMv7 Processor, 4 cores, 1.2GHz
System	Raspbian GNU/Linux 9
Storage	1GB RAM, 32GB microSD

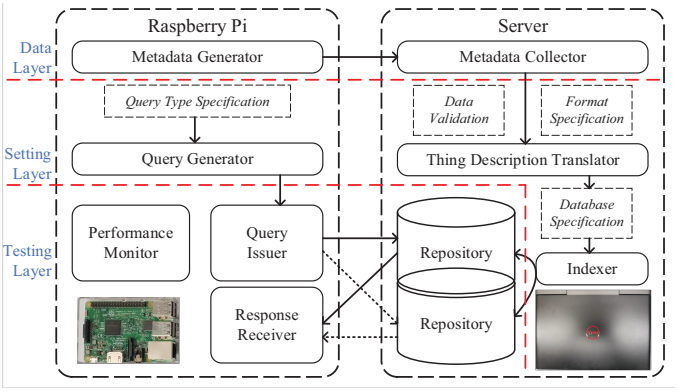
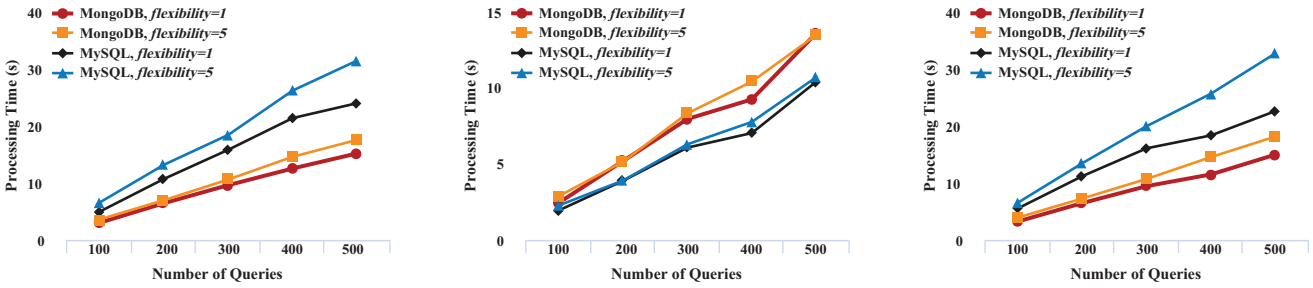


Fig. 2. The system consists of the directory server running in a laptop and a Raspberry Pi querying metadata remotely.

As shown in Fig. 2, the system is composed of three layers: data layer, setting layer, and testing layer. In the data layer, metadata are generated in IoT devices where we use the Raspberry Pi to represent the general IoT devices. TDs are produced and updated by the generator running inside the data layer. The data are delivered to the collector running on the server. In the setting layer, devices generate a bunch of queries for testing based on three specific types of queries: property query, aggregation query, and polygon query. On the server side, metadata are validated as all necessary fields are properly provided. Thing description translator then transforms metadata to either JSON-LD format or RDF/XML format. Then based on user-identified preference of SQL or NoSQL database, thing descriptions with JSON-LD format are pushed to MySQL or MongoDB respectively. Incidentally, JSON-LD format data are stored in SQL table by reserving the space to conquer the flexibility issues, which will be discussed in Section IV. In our tests, we conduct an additional experiment to compare query performance for different data formats bypassing configuring a RDF database. Querying and responding are running in the testing layer. The query issuer takes the generated queries and performs look-up query to



(a). Property Query (b). Aggregation Query (c). Spatial Query

Fig. 3. Query performance on property query, aggregation query, spatial query for local directory with different $flexibility$ of TDs.

MySQL or MongoDB databases. The receiver waits for the response, parses it, and outputs the results. The performance is monitored during the former process. Since lookup functions in directory services require high throughput and low response time, we monitor the response time per query as the performance indicator from time to time in our tests.

IV. Query Performance Analysis

In this section, we present and analyze our evaluation regarding database specification and data description format. We compare SQL database and NoSQL database according to different query types, features, and CPU utilization. Two kinds of thing description format, JSON-LD and RDF, are also evaluated as a supplement.

A. Database Specification

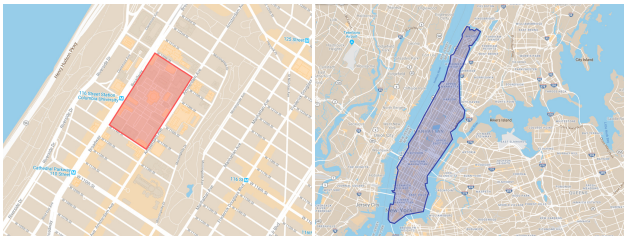
Since IoT devices are highly variable and each device could be configured with distinct fields, formatting metadata into SQL tables requires some sort of unification. In order to reserve spaces for SQL table, we define the flexibility of thing description in Def. 1. The flexibility of TD gives an upper bound of the number of properties, actions, or events. In practice, if a TD exceeds the flexibility limit, the directory drops the exceeding fields in the validation process.

Definition 1 (Flexibility of Thing Description)**.** *The flexibility of a thing description is defined as the largest number of properties, actions, or events allowed by directory databases, i.e.* $flexibility = \max\{\#properties, \#actions, \#events\}$.

Location information is a key feature in IoT scenarios given the high mobility and location-sensitive services in IoT. Substantial location-based queries would impose a heavy load to IoT directory. The performance of polygon query plays an important role in directory services. Correspondingly, our metadata principle requires each TD has an explicitly defined geographic location field with position coordinates (i.e. longitude and latitude). For easy understanding, we formalize the size of a polygon as Def. 2.

Definition 2 (Size of a Polygon)**.** *The number of vertices of a polygon required to accurately represent a geographic area is referred as the size of the polygon.*

Note that it is impossible to achieve 100% accuracy when describing a geographic area. In practice, a polygon in small size usually presents a regular area such as a building or a campus, while a polygon in large size usually defines a irregular area such as a district or a city. As an example in Fig.4, a rectangular polygon shapes Columbia University with $size = 5$ and we quite roughly shapes Manhattan with the polygon whose $size = 55$.



(a). Polygon for Columbia University (b). Polygon for Manhattan

Fig. 4. Sample polygons with small size and large size.

In our experiments, we first evaluate the base case with $flexibility = 1$, where each device has just one property, one action, and one event respectively. Then we extend the experiments to $flexibility = 5$, where each record randomly has up to five properties, five actions, and five events. Therefore, if these data are stored in MySQL, some blank fields are left inside a SQL table. In both cases, the directory loads ten thousands of TDs and we independently test property query, aggregation query, and spatial query. After each round of query testing, we clear the query cache in the database. The density of spatial queries is set to 8 objects per query and the polygon size is set to 5. The queries we used to compare query performance are given in Tab. III. The performance is represented by processing time which is measured as the timespan from issuing a query to fetching the entire response as a tuple. Thus, the processing time can be expressed by the following formula:

$$processing\ time = round\ trip\ time + retrieval\ time$$

TABLE III
Queries Selected in Our Tests

Type of Query	Description
Property Query	Find the devices with a specific title.
Aggregation Query	Count the devices supporting a specific protocol.
Spatial Query	Find the devices inside a specific polygon.

Fig. 3 shows the comparisons for processing time of property query, aggregation query, and spatial query. Although MySQL performs always better on aggregation query, MongoDB responds much faster on property query and spatial query than MySQL. When the flexibility increases to 5, Mon-

goDB outperforms MySQL even obviously on property query and spatial query. The impact of processing time caused by the different flexibility reflects the resilience of databases. The less processing time increases, the better resilience database achieves. For aggregation query, MySQL is always around 1.3 times outperforms than MongoDB, which might be considered as a mild performance loss of MongoDB as the flexibility of TD increases. Note that $flexibility = 1$ means the fields inside tables are mandated, which is the worst case for MongoDB in the comparisons. Thus, we can conclude that MongoDB achieves at least 1.9 times better performance than MySQL for property query and spatial query. Additionally, MongoDB achieves 2.9 and 2.7 times better resilience for property query and spatial query on average, without any resilience loss in aggregation query.

In Fig. 5, we measure CPU utilization rates in the server side where each type of query are tested across eight minutes. It turns out that MongoDB exerts slight more CPU pressure on aggregation query while saves around 60% CPU utilization on other queries.

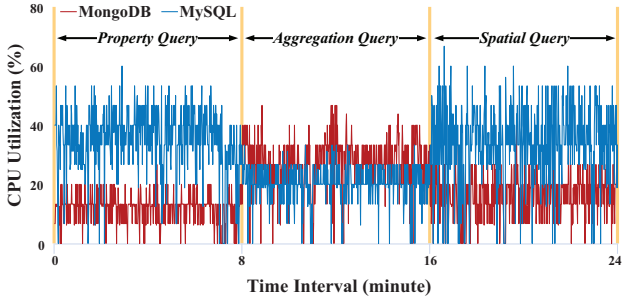
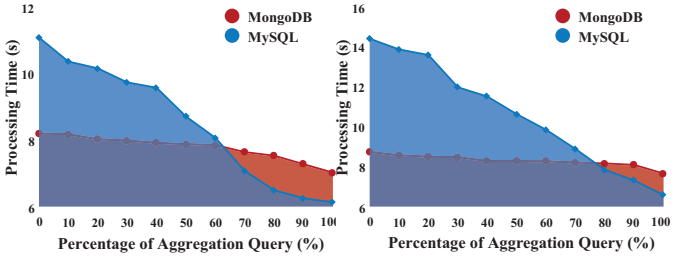


Fig. 5. CPU utilization ratio for three types of query measured in the server.

According to the analysis above, most of the results are in favor of MongoDB. However, MySQL truly handles aggregation query very well. To make the best of that, Fig. 6 measures the processing time against the percentage of aggregation query where property query and spatial query evenly take the rest proportion. The non-overlapping area intuitively presents the benefit of different database. As the proportion of aggregation queries increases over the boundary (63% in Fig 6(a) and 77% in Fig 6(b)), MySQL potentially becomes a better choice.



(a). $flexibility = 1$ (b). $flexibility = 5$

Fig. 6. Impact of proportion of aggregation query.

With the performance testing above, neither of them could evidently perform better than the other one in all aspects, which holds our claim that selection of databases should consider the dedicated IoT behaviour for directory services. Thus, a smart directory provider would estimate the proportions of queries in different types based on their history statistics, rather than recklessly start building directories. That said, if 30% performance loss on aggregation query is acceptable, non-relational database would be a preferable choice for query performance concern.

In addition, polygon size is also a crucial factor mattering the performance of spatial query. During our tests, we find the performance is not much differentiated for polygon sizes inside a small interval, which implies that query performance is not highly sensitive to polygon size if it is restrained to a moderate interval. For space constrains, we only show the performance of two databases against large interval choices in Fig. 7. We present performance as the number of queries per second versus variable polygon sizes. The density of spatial queries is set as around 30 objects per query. The query performance goes down when the polygon size increases at a large step. As the polygon size reaches 365, MongoDB is 2.35 times faster than MySQL.

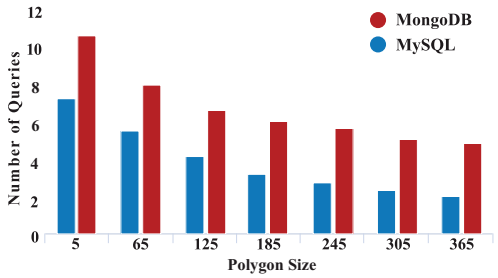


Fig. 7. Spatial query performance versus polygon size.

B. Description Scheme Specification

JSON-LD and RDF are suggested by W3C to be the standard data formats for thing descriptions. As a supplement, we show an additional reason that we adopt JSON-LD as our default description format in the directory system besides its compatibility with databases.

Since both two formats can be queried by SPARQL, it raises our interest to see the differences on query performance. Using Apache Jena framework [14], we perform SPARQL query to the read same TD record, visualized in Fig. 8 stored in JSON-LD and RDF/XML formats.

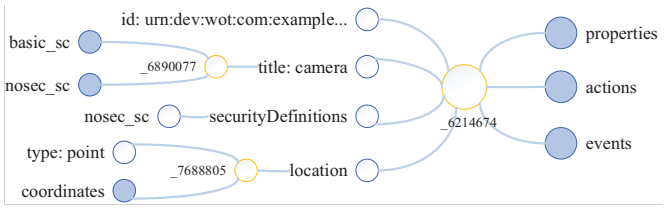


Fig. 8. Visualization of the RDF file queried in our testing.

In Fig. 9, we periodically call the look up in the testing interval. The processing time of JSON-LD is steadily less than that of RDF/XML with only a few exceptions. On average, accessing JSON-LD files is 24% faster than accessing the same information presented in RDF format. For directories using SPARQL query, JSON-LD likely becomes an efficient

choice based on our test. Note that although this test indicates that JSON-LD performs better in simple reading, we make no claims about complicated queries and other aspects.

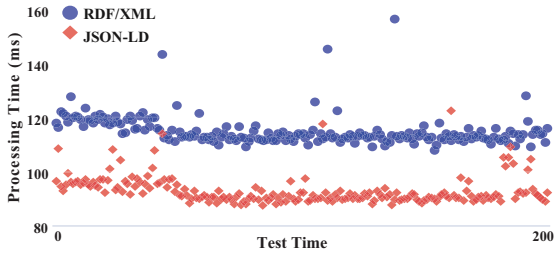


Fig. 9. Read performance for JSON-LD versus RDF by SPARQL queries.

V. Related Work

Directory Service for IoT: A directory service is a service that provides read-optimized access to general data about entities, such as people, corporations, and computers [15]. Traditional directory service [16] achieves satisfied read speed, while does not support variable queries needed by IoT, such as polygon query. Thingweb [11] is an ongoing project which aims to build standard directory services based on both HTTP and COAP, yet it is not necessarily represents the needs of the mass market given various IoT scenarios and diverse databases used in practice. Scalable directory service is designed in [2], [5], which aims to manage massive records and support fast lookup for IoT applications. Instead of scalability, comparing query performance is the major concern in our work and we consider location-based query as a key feature for mobile IoT.

IoT Metadata Description: W3C is drafting the standardization for thing description (TD) [4]. A thing description describes the metadata and interfaces of a physical or virtual entity. Thing description is recommended to be encoded in JSON-LD or RDF formats for machine-understandability. Given the potential limitations of IoT, the thing description file can be hosted by the thing itself or hosted in an external gateway or even cloud. A TD instance has four main components: textual metadata about the thing itself, a set of interaction affordances (i.e. properties, actions, and events) that indicate how the thing can be used, schemas for the data exchanged with the thing, and web links to express any formal or informal relation to other things or documents on the internet.

Database Comparison: There are a few existing works comparing SQL and NoSQL database [6], [7], [17]. Although fundamental achievements are delivered, their works either target some typical sensing scenarios or cannot be applied to directory services. Read, write, and delete performances for sensors with fixed data are evaluated in [6]. In [7], different databases inside the IoT applications are compared. Although MongoDB shows some sort of deficiency, it outperforms Cassandra and Riak when measuring read latency [18]. Geographic query performance under concurrent users is analyzed in [17]. Our work mainly focuses on the future directory analysis, which stores standard IoT description files and dedicates query performance.

VI. Conclusion

The rapid growth of IoT brings substantial related metadata, which renders providers carefully designing their directories. With a focus on query performance, we discuss IoT directory design issues including local directory architecture, database selection, data format selection. In order to achieve higher quality of service, directory designers are supposed to estimate query performance based on the proportions of different query types and utilize proper database to store standard metadata. Our analysis and evaluation based on the proof-of-concept system are the initial practice for this principle. While additional work is required to build the IoT directory in industrial grade, we believe that our measurement on the prototype paves the path for widespread approaches in practice.

In the future, we will extend our directory prototype and build the scalable geo-distributed system.

References

[1] P. Cerwall, P. Jonsson, R. Möller, S. Bävertoft, S. Carson, I. Godor, P. Kersch, A. Kälvemark, G. Lemne, and P. Lindberg, "Ericsson mobility report," *On the Pulse of the Networked Society*, 2015.

[2] V. P. Kafle, Y. Fukushima, P. Martinez-Julia, and H. Harai, "Directory service for mobile IoT applications," in *IEEE Conference on Computer Communications Workshops*, 2017, pp. 24–29.

[3] X. Gao, A. Song, L. Hao, J. Zou, G. Chen, and S. Tang, "Towards efficient multi-channel data broadcast for multimedia streams," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 10, pp. 2370–2383, 2019.

[4] "W3C web of things, thing description editors' draft," https://www.w3.org/TR/wot-thing-description/.

[5] V. P. Kafle, Y. Fukushima, P. Martinez-Julia, and H. Harai, "Design of scalable directory service for future IoT applications," in *IEEE ITU Kaleidoscope: ICTs for a Sustainable World*, 2016, pp. 1–7.

[6] H. Fatima and K. Wasnik, "Comparison of SQL, NoSQL and NewSQL databases for internet of things," in *IEEE Bombay Section Symposium*, 2016, pp. 1–6.

[7] S. Rautmare and D. Bhalerao, "MySQL and NoSQL database comparison for IoT application," in *IEEE International Conference on Advances in Computer Applications*, 2016, pp. 235–238.

[8] A. Sharma, X. Tie, H. Uppal, A. Venkataramani, D. Westbrook, and A. Yadav, "A global name service for a highly mobile internetwork," in *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, 2014, pp. 247–258.

[9] L. Hao, C. Jin, X. Gao, L. Kong, and G. Chen, "QoE-aware optimization for SVC-based adaptive streaming in D2D communications," in *IEEE International Performance Computing and Communications Conference*, 2017, pp. 1–8.

[10] K. Arabshian and H. Schulzrinne, "Gloserv: Global service discovery architecture," in *IEEE Annual International Conference on Mobile and Ubiquitous Systems*, 2004, pp. 319–325.

[11] "Thingweb," http://www.thingweb.io/.

[12] "Mysql," https://www.mysql.com/.

[13] "Mongodb," https://www.mongodb.com/.

[14] "Apache jena," https://jena.apache.org.

[15] S. Fitzgerald, I. Foster, C. Kesselman, G. Von Laszewski, W. Smith, and S. Tuecke, "A directory service for configuring high-performance distributed computations," in *IEEE International Symposium on High Performance Distributed Computing*, 1997, pp. 365–375.

[16] X. Wang, H. Schulzrinne, D. Kandlur, and D. Verma, "Measurement and analysis of LDAP performance," *IEEE/ACM Transactions On Networking*, vol. 16, no. 1, pp. 232–243, 2008.

[17] E. Baralis, A. Dalla Valle, P. Garza, C. Rossi, and F. Scullino, "SQL versus NoSQL databases for geospatial applications," in *IEEE International Conference on Big Data*, 2017, pp. 3388–3397.

[18] J. Klein, I. Gorton, N. Ernst, P. Donohoe, K. Pham, and C. Matser, "Performance evaluation of NoSQL databases: a case study," in *Proceedings of the 1st Workshop on Performance Analysis of Big Data Systems*, 2015, pp. 5–10.