Efficient Approximate Range Aggregation over Large-scale Spatial Data Federation

Yexuan Shi, Yongxin Tong, *Member, IEEE,* Yuxiang Zeng, Zimu Zhou, *Member, IEEE,* Bolin Ding, and Lei Chen, *Fellow, IEEE*

Abstract— Range aggregation is a primitive operation in spatial data applications and there is a growing demand to support such operations over a data federation, where the entire spatial data are separately held by multiple data providers (*a.k.a.*, data silos). Data federations notably increase the amount of data available for data-intensive applications such as smart mobility planning and public health emergency responses. Yet they also challenge the conventional implementation of range aggregation queries because the raw data cannot be shared within the federation and the data partition at each data silo is fixed during query processing. These constraints limit the design space of distributed range aggregation query processing and render existing solutions inefficient on large-scale data. In this work, we propose the first-of-its-kind approximate algorithms for efficient range aggregation over spatial data federation. We devise novel single-silo sampling algorithms that process queries in parallel and design a level sampling based algorithm which reduces the time complexity of local queries at each data silo to $O(\log \frac{1}{\epsilon})$, where ϵ is the approximation ratio of the accuracy guarantee. Extensive evaluations with real-world data show that compared with state-of-the-arts, our solutions reduce the time cost and communication cost by up to $85.1\times$ and $5.5\times$ respectively, with average approximate errors of below 2.8%. In addition, our solutions yield a throughput of over 250 queries per second, achieving real-time responses for real-world bike-sharing applications.

Index Terms—Spatial Data Federation, Range Aggregation, Sampling.

1 Introduction

A range aggregation query over spatial data returns summarized information about the spatial objects falling within a spatial range specified as either a circle or a rectangle [1], [2]. Such queries are crucial for various big spatial data applications such as smart mobility planning [3], public health emergency response [4], urban environment monitoring [5], location-based services [6], etc.

There is a growing trend for the service provider of these applications to operate on a *data federation*, where the data from multiple data providers (*a.k.a.*, data silos) collaborate to improve the quality of services. Each data silo in the federation holds part of the entire data (*i.e.*, rows) under the same schema, and interact with the service provider without revealing its own raw data partition. For example, a bike sharing service provider may frequently process queries such as "*how many shared-bikes are there within 2 kilometers of a subway station*" over data from multiple bike sharing companies. This is exactly the case for real-world applications such as 9-Bike [7], which provides real-time bike sharing services over a federation of bike sharing companies including DiDi bike [8], Hello bike [9], mobike [10], etc.

It is challenging to offer real-time response to frequent range aggregation queries in these applications. *(i)* Traditional distributed range aggregation techniques [6], [11], [12], [13] improve query processing throughput by optimizing data partitioning. However, the data partition is fixed in the federated setting. *(ii)* Prior spatial query processing schemes [2] fail to deliver real-time response in case of high-frequency queries. For example, real-world bike sharing applications may receive around 150 queries per second [14], whereas existing exact range aggregation solutions can only process 50 queries per second (see Sec. 8).

In this paper, we define the Federated Range Aggregation (FRA) problem and investigate efficient solutions to range aggregation queries over large-scale spatial data federation. Observing that the underlying applications demand real-time response of high-frequency queries while a small error in the result is acceptable, we focus on solutions that offer *high-throughput* and high-quality *approximate* query results. At a high level, we optimize FRA query processing from two aspects. *(i)* We avoid enumerating all data silos to answer a single FRA query. We argue that the service provider only needs to communicate with one data silo to offer high-quality query result. This allows data silos to process queries in parallel, which notably improves the throughput. *(ii)* We propose a novel index to accelerate local range aggregation queries at each data silo to further reduce the time complexity to query on large-scale data.

Our main contributions and results are as follows.

- We devise a novel single-silo sampling scheme that radically reduces the communications with data si-

- *Y. Shi and Y. Tong are with the State Key Laboratory of Software Development Environment and Advanced Innovation Center for Big Data and Brain Computing, School of Computer Science and Engineering, Beihang University, PR China. E-mail: {skyxuan, yxtong}@buaa.edu.cn.*
- *Y. Zeng and L. Chen are with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong SAR, China. E-mail: {yzengal,leichen}@cse.ust.hk.*
- *Z. Zhou is with Singapore Management University, Singapore. E-mail: zimuzhou@smu.edu.sg.*
- *B. Ding is with Alibaba Group, China. E-mail: bolin.ding@alibaba-inc.com.*
- *Yongxin Tong is the corresponding author in this paper.*

los to one round of interaction with a single silo, achieving a communication cost of $O(1)$ for IID case and $O(\sum_{j=0}^{|g|} |g_j|)$ for Non-IID case. Such a reduction in communication cost facilitates parallel processing of FRA queries for high throughput.

We propose a new level sampling based index called LSR-Forest for each data silo to accelerate the local range aggregation query at each data silo, which has a time complexity of $O(\log \frac{1}{\varepsilon})$.

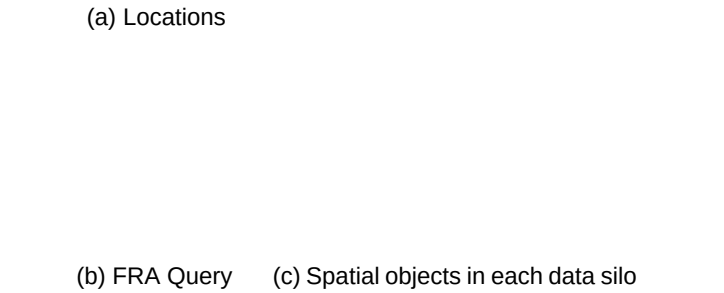
We prove that both the single-silo sampling scheme and the level sampling based local query offer theoretical guarantees on query accuracy. We also show that when the two techniques are used together, the guarantee of query accuracy is still bounded.

Extensive experiments on real-world data show that compared with exact solutions, our approximate algorithms reduce the time cost of a single FRA query by up to 85.1 and communication cost by up to 5.5 with average approximate errors of below 2.8%. In addition, our solutions yield a throughput of over 250 queries per second, achieving real-time responses for real-world bike-sharing applications.

In the rest of this paper, we formulate the FRA problem in Sec. 2, present a solution overview in Sec. 3, and elaborate on detailed techniques in Sec. 4 and Sec. 5. We prove the accuracy guarantees of our solutions in Sec. 6 and discuss their extensions in Sec. 7. Sec. 8 presents the evaluations, Sec. 9 reviews the related work and we conclude in Sec. 10.

TABLE 1: Summary of major notations

Notation	Description
$l_o; a_o$	location and measure attribute of a spatial object o
$s_i; P_{s_i}$	a data silo s_i and its spatial objects P_{s_i}
n_{s_i}	the number of spatial objects in P_{s_i}
$S; P$	a federation S and the whole spatial objects P
m	the number of data silos in the federation S
$R; F$	a query range R and a aggregation function F
$Q(S; R; F)$	our proposed FRA query



(a) Locations

(b) FRA Query (c) Spatial objects in each data silo

Fig. 1: An example of the FRA query

2 Problem Statement

This section defines the Federated Range Aggregation (FRA) query and specifies the performance metrics.

Definition 1 (Spatial Object). A spatial object is denoted by $o = \langle l_o; a_o \rangle$, where l_o is the location of the spatial object and a_o is the corresponding measure attribute.

Here the location is defined in the two-dimensional Euclidean space. The measure attribute is application-specific. For example, l_o can be the GPS of a taxi and a_o can be its speed. For ease of presentation, $P_{s_i} = \{o_1; o_2; \ldots; o_{n_{s_i}}\}$ represents the set of spatial objects stored at the data silo s_i, where n_{s_i} is the number of spatial objects in the set P_{s_i}.

As in previous research [15], [16], [17], we assume multiple data silos are united as a federation for querying over a collection of spatial objects. Specifically:

- The spatial data federation ("federation" for short) S consists of m data silos, i.e., $S = \{s_1; \ldots; s_m\}$.
- Each data silo s_i follows a common schema and owns a horizontal partition P_{s_i} of the entire collection of spatial objects $P = \bigcup_{i=1}^{m} P_{s_i}$.
- A service provider makes queries over the federation S but can only access the spatial objects in P_{s_i} via the query interface of data silo s_i.

With the above setting, the Federated Range Aggregation (FRA) query is defined as below:

Definition 2 (FRA Query). Given a federation S possessing a collection of spatial objects P, a query range R and an aggregation function F, a federated range aggregation (FRA) query from the service provider aims to aggregate the measure attributes of the spatial objects within R:

$$Q(S; R; F) = F(\{a_o \mid o \in P; o \text{ is within } R\}); \quad (1)$$

where each data silo s_i can only access its own data partition P_{s_i}, i.e., s_i can only answer the range aggregation query of $Q(s_i; R; F) = F(\{a_o \mid o \in P_{s_i}; o \text{ is within } R\})$, and R can be either circular or rectangular.

We mainly explain our solution to the aggregation function F of COUNT or SUM and discuss the extensions to other aggregation functions in Sec. 7. Tab. 1 summarizes the major notations that will be used throughout this paper.

Example 1. Assume a federation S of two data silos (Fig. 1a). The first data silo has 10 spatial objects (marked in blue) and the second data silo holds 8 spatial objects (marked in red). The locations and measure attributes of these spatial objects are in Fig. 1c. An FRA query is shown in Fig. 1b, which asks the SUM of the measure attributes of those spatial objects within a circular range (marked in green) centered at $(4; 6)$ with a radius of 3.

As mentioned in Sec. 1, we aim to develop high-throughput, high-quality approximate algorithms to process frequent FRA queries over large-scale data federation. In addition to throughput, we also use time complexity and communication cost as efficiency metrics. The accuracy of approximate algorithms is quantified by ε-approximation.

Definition 3 (ε-approximation). For an FRA query with an exact result ans, an ε-approximation solution should always report a result ans' such that $(1 - \varepsilon)\, ans \le ans' \le (1 + \varepsilon)\, ans$.

Fig. 2: An overview of our approximate solutions

3 Solution Overview

We optimize FRA query processing from two aspects.

- Avoid enumerating all data silos to answer an FRA query. A naive solution would exchange information with every data silo to answer a range aggregation query, allowing only sequential processing. Proper sampling strategies can enable parallel processing, which improves the throughput on query streams.
- Accelerate local range aggregation queries at each data silo. Although spatial indices such as R-trees enable $O(\log n_{s_k})$-time range aggregation queries for a data silo s_k, the time to obtain a partial aggregation answer is still a bottleneck. We argue that the local range aggregation queries can be further sped up via approximate solutions.

Fig. 2 shows an overview of our solution. Central in our solution are two techniques:

- Single-Silo Sampling (Sec. 4). We reduce the number of silos for partial aggregation result retrieval from m to 1 to allow parallel query processing. This is achieved with a grid index to track the distribution of the spatial data partitioning and the corresponding query result estimation algorithms.
- Level Sampling based Local Query (Sec. 5). We devise a novel level sampling based index (LSR-Forest) for fast approximate range aggregation queries at each silo. It reduces the average time complexity of local range aggregation queries to $O(\log \frac{1}{\varepsilon})$.

4 Single-Silo Sampling

In this section, we present our strategies to avoid enumerating all data silos when processing an FRA query. We first introduce the indices (Sec. 4.1), then explain our silo sampling and result estimation methods (Sec. 4.2), and finally present the framework to process multiple FRA queries (Sec. 4.3).

4.1 Grid Index Construction

As a prerequisite for silo sampling, the service provider constructs grid indices to track the distributions of the spatial objects, where each grid aggregates the measure attributes of its covered spatial objects.

Algorithm 1: Construct grid index

Input: federation $S = \{s_1, \ldots, s_m\}$
Output: a set of grid indices G

1 foreach data silo $s_i \in S$ do
2 $g_i \leftarrow$ construct a grid index for the data in s_i;
3 s_i sends the grid index g_i to service provider;
4 $g_0 \leftarrow$ merge the received grid indices $g_1, \ldots, g_m$;
5 return $G = \{g_0, g_1, \ldots, g_m\}$;

Alg. 1 shows how to construct the grid indices. Denote $G = \{g_0, g_1, \ldots, g_m\}$ as a set of grid indices, where $g_1, \ldots, g_m$ are the grid indices of the spatial objects owned by data silos $s_1, \ldots, s_m$ and g_0 is merged from the other m grid indices. In lines 1-3, a request of constructing the grid index is sent to each data silo s_i and each silo sends its grid index g_i back to the service provider. These grid indices are then merged as g_0 for all the spatial objects.

Example 2. Back to Example 1, we build a grid index, whose grid length is 2.5. The two data silos can be spilt into 16 grids, as shown in Fig. 1b. In these grid indices, we store the number (COUNT) of the covered spatial objects and SUM of their measure attributes. For example, in the bottom-left corner of the grid index, the first data silo has no spatial object while the second data silo has one spatial object. Thus, for the first data silo, both COUNT and SUM values of this grid are 0. For the second data silo, the COUNT value is 1, and the SUM value is 7 since the measure attribute of the spatial object $(2, 2)$ is 7. Finally, we can build the grid index g_0 by summing up the COUNT/SUM values of the corresponding grid in grid indices g_1 and g_2.

Complexity Analysis. In Alg. 1, each silo s_i takes $O(n_{s_i})$ time and $O(|g_i|)$ space to construct a grid index, and the service provider takes $O(\sum_{i=1}^{m} |g_i|)$ time, $O(|g_0|)$ space and $O(\sum_{i=1}^{m} |g_i|)$ communication cost to construct the grid index g_0, where $|g_i|$ is the number of grids in g_i.

4.2 Silo Sampling & Result Estimation

Now we explain how to sample only one data silo for a partial answer and estimate the result for the FRA query based on the grid indices. Since the spatial objects can distribute identically or non-identically, we devise different strategies for these two cases[1].

4.2.1 Identically Distributed Spatial Data

If the spatial objects are identically (and independently) distributed (i.e., IID) across data silos, it is intuitive to only retrieve a partial answer from one data silo and estimate the result from its partial answer. Note that our algorithm is applicable for any specific distribution of the IID case.

Alg. 2 illustrates single-silo sampling and query result estimates for a given FRA query in the IID case. In line 1, a data silo s_k is picked at random from the federation S. In lines 2-3, a request of local range aggregation query is sent to s_k and the result res_k is sent back to the service provider. In lines 4-8, the service provider estimates the approximate

1. Usually the spatial objects are independent.

Algorithm 2: IID-est

Input: federation $S = \{s_1, \ldots, s_m\}$, query range R, aggregation function F and grid indices G
Output: approximate result of FRA query

1 s_k randomly sample a data silo from S;
2 send a request of range aggregation query (R, F) to the data silo s_k;
3 res_k receive the query answer sent by the data silo s_k;
4 $sum_0 \leftarrow 0$; $sum_k \leftarrow 0$;
5 **foreach** grid i in the grid index g_0 such that grid i intersects with the query range R **do**
6 $sum_0 \leftarrow sum_0 +$ aggregation of grid i in g_0;
7 $sum_k \leftarrow sum_k +$ aggregation of grid i in g_k;
8 $ans^0 \leftarrow sum_0 \cdot (res_k / sum_k)$;
9 **return** ans^0;

Algorithm 3: NonIID-est

Input: federation $S = \{s_1, \ldots, s_m\}$, query range R, aggregation function F and grid indices G
Output: approximate result of FRA query

1 s_k randomly sample a data silo from S;
2 send a request of range aggregation query (R, F) to data silo s_k;
3 $res_k^1, \ldots, res_k^{|g_k|}$ receive the query results sent by data silo s_k, where res_k^i denotes the contribution of the spatial objects in grid i;
4 $ans^0 \leftarrow 0$;
5 **foreach** grid i in the grid index g_0 such that grid i intersects with the query range R **do**
6 $est_0^i \leftarrow res_k^i \cdot \frac{\text{aggregation of grid } i \text{ in } g_0}{\text{aggregation of grid } i \text{ in } g_k}$;
7 $ans^0 \leftarrow ans^0 + est_0^i$;
8 **return** ans^0;

result based on res_k. Specifically, it first iterates each grid i that intersects with the query range R (line 5). Then, it accumulates the SUM/COUNT values of i in the grid index g_0 as sum_0 and accumulates the SUM/COUNT values of i in the grid index g_k as sum_k (lines 6-7). The approximate result ans^0 is calculated as $sum_0 \cdot (res_k / sum_k)$ (line 8).

Example 3. Back to Example 1. Assume silo s_2 is sampled to process the local range aggregation query. From Fig. 1, its partial answer res_k is 4 (line 3). Next, we estimate the aggregation results in lines 5-7. We only need to enumerate the 3 grids in the top-left corner. Accordingly, we can calculate $sum_0 = 4+0+0$ (first row) $+2+2+4$ (second row) $+4+1+4$ (third row) $= 21$. and $sum_k = 3+0+0+0+1+2+0+1+4 = 11$. Finally, we get the approximate result as $21 \cdot (4/11) = 7.6$.

Complexity Analysis. In Alg. 2, s_k is sampled uniformly at random. Thus, line 3 takes $O(avg\{\log n_{s_i}\})$ time on average. In lines 4-7, it takes $O(|g_0|)$ time and extra $O(1)$ space (other than the usage of grid indices) to calculate the estimated result. Hence Alg. 2 takes $O(|g_0| + avg\{\log n_{s_i}\})$ time. The communicate cost is $O(1)$ since the service provider only interacts with one silo s_k.

Remarks. The time complexity of Alg. 2 can be reduced to $O(avg\{\log n_{s_i}\})$ by efficiently calculating the aggregation results of the grids which intersect with R. Specifically, we maintain a cumulative array $arr_k[i][j]$ to store the sum of SUM/COUNT of grids $(0,0)$ to (i,j), which can be pre-calculated during the construction of g_k. By the inclusion-exclusion principle, it takes $O(1)$ time to calculate sum_0 and sum_k. In Example 1, we can calculate $(0,0)$ in Fig. 1b and $(2,2)$ in Fig. 1b among the grids intersecting with R. Then $sum_0 = arr_0[2][3] - arr_0[2][0] = 32 - 11 = 21$ and $sum_k = arr_k[2][3] - arr_k[2][0] = 18 - 7 = 11$, which takes $O(1)$ time. This way, lines 4-7 take $O(1)$ time and the time complexity of Alg. 2 is $O(avg\{\log n_{s_i}\})$.

4.2.2 Non-Identically Distributed Spatial Data

If the spatial objects are not identically distributed (Non-IID), Alg. 2 can be biased (see Theorem 1). Hence we need a new solution to the Non-IID spatial data. The key idea is to leverage a commonly used assumption [18], [19], [20] that the spatial objects within a small area (e.g., a grid) often follow the same distribution. This inspires us to ask the sampled data silo to send not only the partial answer to the service provider, but also the contribution of the spatial objects by each grid in the aggregation answer. Such information will enable an unbiased estimation of the result for a given FRA query (detailed proof in Sec. 6).

Alg. 3 illustrates the silo sampling and result estimation algorithm for Non-IID spatial data. In lines 1-2, a data silo s_k is selected at random to process the local range aggregation query. In line 3, s_k sends back to the service provider a vector $res_k^1, \ldots, res_k^{|g_k|}$ from s_k, where res_k^i is the contribution of the spatial objects of grid i in the partial answer and $|g_k|$ is the number of grids in the grid index of s_k. In lines 4-7, the service provider estimates the approximate result based on the received vector. Specifically, it first iterates each grid i that intersects with the query range R (line 5). Then, it estimates the contribution of the spatial objects of all the data silos in grid i, which is denoted by est_0^i (line 6). Finally, it accumulates the estimated contribution est_0^i into the approximate result ans^0 (line 7).

Here we assume that the coverage of data silos is overlapping, but may not be uniform across space. This is reasonable because for applications such as federated bike sharing, companies provide services across the entire city, although each company may have a different strategical focus. Our method can also be extended to the case of non-overlapping coverage. Specifically, in line 1, we sample s_k from silos who have data in the query range. This way, we can filter irrelevant silos and find a representative silo s_k.

Example 4. Back to Example 1 and still assume data silo s_2 is sampled to answer the local range aggregation query. According to Alg. 3, s_2 will send a vector $\{0,0,0,0$ (first row), $0,1,2,0$ (second row), $0,1,0,0$ (third row), $0,0,0,0$ (fourth row)$\}$ back (line 3). To estimate the approximate result of the FRA query, we first enumerate the 3×3 grids in the top-left corner. Then, for each grid, we estimate the contribution of all the spatial objects in the final result. For example, when the grid in the second row and second column is iterated, we have $est_0^i = 1 \cdot \frac{2}{1} = 2$ and then accumulate est_0^i into the final result ans^0. In the end, we get the estimated result $ans^0 = 0+0+0+0+2+4+0+1+0 = 7$.

Complexity Analysis. In Alg. 3, s_k is sampled uniformly at

Algorithm 4: Single-Silo Sampling Framework

Input: federation $S = \{s_1, \ldots, s_m\}$, a set of query $Q = \{q_1, \ldots, q_{|Q|}\}$ and grid indices G
Output: approximate result of FRA queries

1 foreach $q \in Q$ do
2 $\quad s_q$ randomly sample a data silo from S;
3 $\quad$ send a request of range aggregation query q to data silo s_q according to corresponding distribution (IID or Non-IID);
4 foreach received query results of query q from silo s_q do
5 $\quad$ estimate the final result ans_q according to corresponding distribution (IID or Non-IID);
6 return $ans_1, \ldots, ans_{|Q|}$;

random. Thus line 3 takes $O(\text{avg}\{\log n_{s_i}\})$ time. In lines 4-7, the service provider needs $O(|g_0|)$ time and extra $O(|g_0|)$ space (other than the usage of grid indices) to calculate the estimated result. Thus, the total time complexity of Alg. 3 is $O(|g_0| + \text{avg}\{\log n_{s_i}\})$. The communicate cost is also $O(|g_0|)$ since it only receives a vector of $|g_k|$ ($< |g_0|$) elements from one data silo s_k.

Remarks. As with in Sec. 4.2.1, the time complexity of Alg. 3 can also be further reduced by enumerating only the grids which intersect with the boundary of R. For those grids covered in R, we can apply the technique in Sec. 4.2.1 to directly calculate their actual contributions. For instance, in Example 1, only grid $(1, 1)$ is covered in R. Its actual contribution can be calculated by $res_0^i = 2$. This way, line 3 only transmits $O(\sqrt{|g_0|})$ query results and lines 4-7 only takes $O(\sqrt{|g_0|})$ time. Thus, the time complexity and communication cost of Alg. 3 is reduced to $O(\sqrt{|g_0|} + \text{avg}\{\log n_{s_i}\})$ and $O(\sqrt{|g_0|})$.

4.3 Framework to Process Multiple FRA Queries

In this subsection, we explain how the single-silo sampling algorithm in Sec. 4.2 enables parallel processing of multiple FRA queries for high-throughput.

Alg. 4 shows the framework to process $|Q|$ FRA queries. In lines 1-3, the service provider sends a query to a random sampled silo s_k. s_k will answer the query and return the query results to service provider (line 4). After the service provider receives the query results of q from silo s_k, it estimates the final result ans_q in line 5. Finally, the answers of all the $|Q|$ queries are returned in line 6.

Complexity Analysis. In Alg. 4, s_q is sampled uniformly at random. Thus, different queries are executed on different silos in parallel. It means that one silo only needs to execute $|Q|/m$ queries in expectation. As a result, the expected time complexity of line 4 is $O(\frac{|Q|}{m}\text{avg}\{\log s_q\})$. The rest of the framework will take $O(|Q|)$ time for IID-est and $O(|Q|\sqrt{|g_0|})$ for NonIID-est. The total communication cost is also $O(|Q|)$ for IID-est and $O(|Q|\sqrt{|g_0|})$ for NonIID-est.

Remarks. For high-throughput parallel processing of multiple queries, it is important to reduce the workload of each silos. In our solution, each silo will only receive $|Q|/m$ queries for execution in expectation. Compared to the naive solution in which each silo will process all $|Q|$ queries, our framework can reduce the workload by $1/m$.

5 Level Sampling Based Local Query

This section presents our algorithm to reduce the average time complexity of local range aggregation query from $O(\log n_{s_k})$ to $O(\log \frac{1}{\varepsilon})$. It is achieved by a novel level sampling based index (Sec. 5.1) and the corresponding local range aggregation query algorithm (Sec. 5.2).

5.1 LSR-Forest Index Construction

For data silo s_k, the LSR-Forest index consists of multiple R-trees $T^0, T^1, \ldots, T^{\log n_{s_k}}$. Each R-tree T^i indexes a set of sampled spatial objects (denoted by $P_{s_k}^i$), where i is the identifier of R-tree and each spatial object is sampled with probability $1/2^i$. In other words, the tree of a higher identifier will have fewer spatial objects.

Alg. 5 illustrates how to construct the LSR-Forest for data silo s_k. Specifically, in lines 1-2, s_k constructs an R-tree T^0 based on all the spatial objects $P_{s_k}^0$ (i.e., $P_{s_k}^0 = P_{s_k}$). In lines 3-5, as the level i increases, fewer spatial objects are sampled. Concretely, the spatial objects $P_{s_k}^i$ in current level are generated by sampling the spatial objects in $P_{s_k}^{i-1}$ with probability $1/2$. Then data silo s_k constructs R-tree T^i based on the newly sampled spatial objects. Finally, all the constructed R-trees $T^0, T^1, \ldots, T^{\log n_{s_k}}$ form the LSR-Forest index for data silo s_k.

Example 5. Back to Example 1. We now construct the LSR-Forest index for silo s_2. As shown in Fig. 1c, there are eight spatial objects o_1-o_8 in the second data silo. In other words, $P_{s_2}^0 = \{o_1, \ldots, o_8\}$ and $n_{s_2} = 8$ in line 1, and hence we can construct an R-tree T^0 of P^0. In lines 3-5, there are $\log n_{s_2} = 3$ iterations and we sample a subset $P_{s_2}^i$ in each iteration. For example, in the first iteration, we assume the first four spatial objects in $P_{s_2}^0$ are sampled, i.e., $P_{s_2}^1 = \{o_1, \ldots, o_4\}$, since the sampling probability is $1/2$. In the second iteration, we assume the first two spatial objects in P^1 are sampled, i.e., $P_{s_2}^2 = \{o_1, o_2\}$. In the last iteration, we assume the first one spatial object in $P_{s_2}^2$ is sampled, i.e., $P_{s_2}^3 = \{o_1\}$. Then we can construct an R-tree for the sampled spatial objects in each iteration.

Complexity Analysis. Assume n_{s_k} is the number of spatial objects in silo s_k. In lines 2-5, the construction of R-tree on $P_{s_k}^i$ takes $O(\frac{1}{2^i} n_{s_k} \log n_{s_k})$ time and space. Thus, the entire construction takes $\sum_{i=0}^{\log n_{s_k}} O(\frac{1}{2^i} n_{s_k} \log n_{s_k}) = O(n_{s_k} \log n_{s_k})$ time and space. Note that the LSR-Forest index construction is a one-off effort. Its time complexity is negligible when processing large amounts of FRA queries.

5.2 Local Range Aggregation Query

With the LSR-Forest indices, we can conduct approximate local range aggregation query at a given silo as follows: first select a level l, then conduct the local range aggregation query on the R-tree T^l, and finally use the answer on the R-tree T^l multiplying 2^l as the result from this silo. Note that the selection of level l is determined by the required accuracy guarantee. Given an approximate ratio ε, a least upper bound τ and a rough estimate of the query result sum_0 (the aggregation result of grids that intersect with the query range in our work), l is obtained by $\lfloor \log_2 \frac{\varepsilon^2 sum_0}{3 \ln \frac{2}{\tau}} \rfloor$. Detailed derivation on determining l is deferred to Lemma 1.

Algorithm 5: Construct LSR-Forest for data silo s_k

Input: spatial objects P_{s_k}
Output: index LSR-Forest
1 $P^0_{s_k} \quad P_{s_k}$; $n_{s_k} \quad |P_{s_k}|$;
2 $T^0 \quad$ construct an R-tree based on $P^0_{s_k}$;
3 for $i \quad 1$ to $\log n_{s_k}$ do
4 $\quad P^i_{s_k} \quad$ sample each spatial object in $P^{i-1}_{s_k}$ with probability $1/2$;
5 $\quad T^i \quad$ construct an R-tree based on $P^i_{s_k}$;
6 return $T^0, T^1, \ldots, T^{\log n_{s_k}}$;

Algorithm 6: Local range aggregation query by LSR-Forest

Input: query range R, aggregation function F, index LSR-Forest, approximate ratio ε, least upper bound τ and the aggregation result on grids sum_0
Output: approximate result of local range aggregation query
1 $T_l \quad$ selects a proper R-tree based on Lemma 1, where $l = \lfloor \log_2 \frac{2 sum_0}{3 \ln \frac{2}{\tau}} \rfloor$;
2 $res_l \quad$ answer the range aggregation query F by the R-tree T_l;
3 $res^0 \quad res_l \quad 2^l$;
4 return res^0;

Alg. 6 presents the procedure to answer a range aggregation query with our index LSR-Forest. In line 1, we pick a proper R-tree T^l from LSR-Forest based on ε using Lemma 1. In line 2, we use the picked R-tree T^l to conduct the range aggregation query and obtain the answer res^l. Accordingly, we can estimate the answer of all the spatial objects owned by data silo s_k as $res_l \quad 2^l$ in line 3.

Example 6. Back to Example 1. Assume silo s_2 is sampled for local range aggregation query and l is calculated as 1 in line 1 of Alg. 6. Since T^1 is an R-tree index of the spatial objects o_3 o_4, only o_3 and o_4 are within the range R, i.e., $res_l = 1 + 1 = 2$ in line 2. Finally, the approximate result of the local range aggregation query is $res^0 = 2 \quad 2^1 = 4$.

Complexity Analysis. Assume n_{s_k} is the number of spatial objects in data silo s_k. The time cost of Alg. 6 is related to the chosen level l. Since the expected number of spatial objects in T^l is $\frac{n_{s_k}}{2^l}$, the time cost for a query on T^l (line 2) is $O(\log \frac{n_{s_k}}{2^l})$. In line 1 we have $2^l = O(\varepsilon^2 sum_0)$ and sum_0 is the number of spatial objects in the grids which intersect with the query range. Assume sum_0 varies from 0 to n_{s_k} uniformly. Then we have $2^l = O(\varepsilon^2 n_{s_k})$. Thus, the time cost of line 2 is $O(\log \frac{n_{s_k}}{\varepsilon^2 n_{s_k}}) = O(\log \frac{1}{\varepsilon})$.

Remark. With Alg. 6, the time complexity of local query is reduced from $O(\log n_{s_k})$ to $O(\log \frac{1}{\varepsilon})$ in average, which is independent of the number of spatial objects in data silo s_k. Thus, the time cost for local queries is almost the same across data silos, which contributes to better load balancing when combined with Alg. 2 or Alg. 3 to answer FRA queries.

6 Accuracy Guarantees

Main Theoretical Results. Our approximate solution to an FRA query achieves ε-approximation with probability $1 \quad 4\exp\{\frac{\varepsilon^2 ans}{2 sum_0}\}$, where ans and sum_0 are the exact answer and a rough estimate of the query result, respectively. The guarantee holds for both the IID and Non-IID cases, whether with or without level sampling based local query.

We prove the above claims by first showing the accuracy bound of our level sampling based local query (Alg. 6), followed by the IID case without / with level sampling (Alg. 2 / Alg. 2 + Alg. 6), and finally the Non-IID case without / with level sampling (Alg. 3 / Alg. 3 + Alg. 6).

Accuracy of Level Sampling Based Local Query. The accuracy of our level sampling based local query is ensured by the following lemma.

Lemma 1. Alg. 6 achieves ε-approximation with probability $1 \quad \tau$.

Proof. Let X_{o_i} indicate whether the spatial object o_i locates within R. In Alg. 6, res_l is the answer of the range aggregation query by the R-tree T^l. We have

$$E[res_l] = \frac{res}{2^l}.$$

In Alg. 6, $res^0 = res_l \quad 2^l$. Then we have

$$E[res^0] = 2^l \; E[res_l] = 2^l \quad \frac{res}{2^l} = res.$$

Based on the Chernoff's inequality [21], we have

$$P[|res_l \quad \frac{res}{2^l}| > \quad \frac{res}{2^l}] \quad 2\exp\{\quad \frac{\varepsilon^2 res}{3 \quad 2^l}\}.$$

Multiplying 2^l to both sides of the inequality inside $P[\ ]$,

$$P[|res^0 \quad res| > \quad res\ ] \quad 2\exp\{\quad \frac{\varepsilon^2 res}{3 \quad 2^l}\}.$$

Since $l = \lfloor \log_2 \frac{2 sum_0}{3 \ln \frac{2}{\tau}} \rfloor$, where sum_0 is a rough estimate result of res by grids. In practice, the ratio of sum_0 and res is usually less than 2. Thus, $l = \lfloor \log_2 \frac{2 res}{3 \ln \frac{2}{\tau}} \rfloor$. Finally, $P[|res^0 \quad res| > \quad res] \quad \tau$.

□

Accuracy of Approximate Solutions (IID Case). Our solutions in the IID case include (i) single-silo sampling (Alg. 2) and (ii) single-silo sampling combined with level sampling (Alg. 2 + Alg. 6). Their accuracy is guaranteed by Theorem 1 and Theorem 2.

Theorem 1. Alg. 2 achieves ε-approximation with probability $1 \quad 4\exp\{\frac{\varepsilon^2 ans}{2 sum_0}\}$.

Proof. In the IID case, the spatial objects of all data silos follow the same distribution. Without loss of generality, assume the probability density function (a.k.a. PDF) of the distribution is $f(o)$. Let X_{o_i} indicate whether the spatial object o_i in P locates in the query range R. Then, X_{o_i} follows a Bernoulli distribution and the exact result is $ans = \sum_{o_i \in P} X_{o_i}$. In Alg. 2, res_k is the result of local range aggregation on the sampled data silo s_k, and sum_k is the aggregation result of all the grids on s_k which intersect with R. Let G' be such a set of grids and denote $R \quad G'$ as grids G' that intersect with R. Then, we have

$$E[ans] = E\Big[\sum_{o_i \in P} X_{o_i}\Big] = E\Big[\sum_{o_i \in P; o_i \text{ locates in } G'} X_{o_i}\Big] = sum_0 \cdot \frac{\int_R f(o)do}{\int_{G'} f(o)do}.$$

Similarly,

$$E[res_k] = sum_k \frac{\int_R f(o)do}{\int_{G'} f(o)do}.$$

From Alg. 2, we know

$$ans' = sum_0 \frac{res_k}{sum_k}.$$

Thus,

$$E[ans'] = sum_0 \frac{E[res_k]}{sum_k} = sum_0 \frac{\int_R f(o)do}{\int_{G'} f(o)do} = E[ans].$$

Applying Hoeffding's inequality [21], we have

$$P\{|ans' - E[ans']| \geq \frac{\varepsilon}{2} ans\} \leq 2\exp\{-\frac{2\varepsilon^2 ans^2}{2sum_0^2}\}.$$

Similarly,

$$P\{E[ans'] - ans \geq \frac{\varepsilon}{2} ans\} = P\{ans - E[ans] \geq \frac{\varepsilon}{2} ans\} \leq 2\exp\{-\frac{2\varepsilon^2 ans^2}{2sum_0^2}\}.$$

Finally, the probability to achieve ε-approximation is

$$\begin{aligned} &P[|ans' - ans| \geq \varepsilon ans] \\ &\leq P[|ans' - E[ans']| + |E[ans'] - ans| \geq \varepsilon ans] \\ &\leq P\{|ans' - E[ans']| \geq \frac{\varepsilon}{2} ans\} + P\{|E[ans'] - ans| \geq \frac{\varepsilon}{2} ans\} \\ &\leq 4\exp\{-\frac{2\varepsilon^2 ans^2}{2sum_0^2}\}. \end{aligned}$$

□

Theorem 2. If local range aggregation query is performed by Alg. 6, Alg. 2 achieves ε-approximation with probability $1 - 4\exp\{-\frac{2\varepsilon^2 ans^2}{2sum_0^2}\}$.

Proof. Denote the query result of Alg. 6 as res'_k. From Alg. 2, we know $ans' = sum_0 \frac{res'_k}{sum_k}$.

According to Lemma 1, we have

$$E[res'_k] = res_k.$$

Thus, based on the proof of Theorem 1, we know

$$E[ans'] = sum_0 \frac{E[res'_k]}{sum_k} = E[ans].$$

$$P[|ans' - ans| \geq \varepsilon ans] \leq 4\exp\{-\frac{2\varepsilon^2 ans^2}{2sum_0^2}\}.$$

□

Accuracy of Approximate Solutions (Non-IID Case). Our solutions in the Non-IID case include (i) single-silo sampling (Alg. 3) and (ii) single-silo sampling combined with level sampling (Alg. 3 + Alg. 6). Their accuracy is guaranteed by Theorem 3 and Theorem 4.

Theorem 3. Alg. 3 achieves ε-approximation with probability $1 - 4\exp\{-\frac{2\varepsilon^2 ans^2}{2sum_0^2}\}$.

Proof. In the Non-IID case, assume the spatial objects in different grids follow different distributions, while the spatial objects in the same grid follow the same distribution. Denote $f_i(o)$ as the PDF of the distribution in the grid i. Further denote P^i as the set of spatial objects in the federation which are located in grid i, and P^i_k as the set of spatial objects in the sampled data silo s_k which are located in grid i. Assume $R \setminus i$ is the area where grid i intersects with query range R. Let X_{o_j} indicates whether a spatial object is located within R. Then, X_{o_j} follows a Bernoulli distribution and $ans = \sum_{o_j \in P} X_{o_j}$. Accordingly, we have

$$E[ans] = E[\sum_{o_j \in P} X_{o_j}].$$

From Alg. 3, we have

$$\begin{aligned} res^i_k &= \sum_{o_j \in P^i_k} X_{o_j}; \\ est^i_0 &= res^i_k \frac{\text{aggregation of grid } i \text{ in } g_0}{\text{aggregation of grid } i \text{ in } g_k}; \\ ans' &= \sum_i est^i_0. \end{aligned}$$

Thus, it can be inferred that

$$\begin{aligned} E[est^i_0] &= E[res^i_k] \frac{\text{aggregation of grid } i \text{ in } g_0}{\text{aggregation of grid } i \text{ in } g_k} \\ &= E[\sum_{o_j \in P^i_k} X_{o_j}] \frac{\text{aggregation of grid } i \text{ in } g_0}{\text{aggregation of grid } i \text{ in } g_k} \\ &= E[\sum_{o_j \in P^i} X_{o_j}]. \end{aligned}$$

By the linearity of expectation, we have

$$E[ans'] = \sum_i E[est^0_i] = \sum_i E[\sum_{o_j \in P^i} X_{o_j}] = E[\sum_{o_j \in P} X_{o_j}] = E[ans].$$

By applying Hoeffding's inequality [21], we have

$$P\{|ans' - E[ans']| \geq \frac{\varepsilon ans}{2}\} \leq 2\exp\{-\frac{2\varepsilon^2 ans^2}{2sum_0^2}\}.$$

Similarly,

$$P\{ans - E[ans'] \geq \frac{\varepsilon ans}{2}\} = P\{ans - E[ans] \geq \frac{\varepsilon ans}{2}\} \leq 2\exp\{-\frac{2\varepsilon^2 ans^2}{2sum_0^2}\}.$$

Finally, the probability to achieve ε-approximation is

$$\begin{aligned} &P[|ans' - ans| \geq \varepsilon ans] \\ &\leq P[|ans' - E[ans']| + |E[ans'] - ans| \geq \varepsilon ans] \\ &\leq P\{|ans' - E[ans']| \geq \frac{\varepsilon ans}{2}\} + P\{|E[ans'] - ans| \geq \frac{\varepsilon ans}{2}\} \\ &\leq 4\exp\{-\frac{2\varepsilon^2 ans^2}{2sum_0^2}\}. \end{aligned}$$

□

Theorem 4. If local range aggregation query is performed by Alg. 6, Alg. 3 achieves ε-approximation with probability $1 - 4\exp\{-\frac{2\varepsilon^2 ans^2}{2sum_0^2}\}$.

Proof. Denote the query result of Alg. 6 as res'^i_k. From Alg. 3, we know

$$est^i_0 = res'^i_k \frac{\text{aggregation of grid } i \text{ in } g_0}{\text{aggregation of grid } i \text{ in } g_k}.$$

According to Lemma 1, we have

$$E[res'^i_k] = res^i_k.$$

Thus, based on the proof of Theorem 3, we know

$$E[ans'] = \sum_i E[est_i'] = \sum_i E[res_k^i] \frac{\text{aggregation of grid } i \text{ in } g_0}{\text{aggregation of grid } i \text{ in } g_k}$$
$$= \sum_i E[\sum_{o_j \in P_i} X_{o_j}] = E[\sum_{o_j \in P} X_{o_j}] = E[ans].$$

Similar to the proof of Theorem 3, by Hoeffding's inequality [21], the probability to achieve α-approximation is

$$P[|ans' - ans| \geq \alpha ans] \leq 4\exp\{-\frac{2\alpha^2 ans^2}{2sum_0}\}.$$

□

7 Extensions

We briefly discuss extensions of our proposed solutions to other aggregation functions such as AVG and STDEV. Note that our level sampling based local query is agnostic to the underlying aggregation function. Hence we mainly discuss the changes in single-silo sampling and result estimation to support AVG and STDEV.

Extension to AVG. Since Alg. 2 and Alg. 3 support SUM and COUNT, AVG can be implemented as the ratio between a SUM query and a COUNT query.

Accordingly, there will be two rounds of local aggregation queries in lines 2-3 of Alg. 2 and Alg. 3 to query COUNT and SUM. Similarly, lines 5-7 of these two algorithms are changed to estimate the results of COUNT and SUM for the entire federation, count and sum. Finally, the result of AVG is estimated as $\frac{sum}{count}$.

Extension to STDEV. To support STDEV, we first implement a user-defined aggregation function SUM_SQR, where SUM_SQR(P) is the sum of the squares of the measure attributes, i.e., $\sum_{o \in P} a_o^2$. To support SUM_SQR, we maintain an extra value a_o^2 in each index and process it in the same way as SUM. Then STDEV is supported because

$$STDEV(P) = \sqrt{\frac{SUM\_SQR}{|P|} - AVG(P)^2}.$$

Accordingly, there will be three rounds of local aggregation queries in lines 2-3 of Alg. 2 and Alg. 3. The first round queries COUNT, the second round queries SUM and the last round queries SUM_SQR. The same process in lines 5-7 follows to estimate the results of COUNT, SUM and SUM_SQR for the entire federation, denoted as count, sum and ssqr. Finally, the result of STDEV is estimated as

$$STDEV(P) = \sqrt{\frac{ssqr}{count} - \frac{sum}{count}^2}.$$

Remark. The time complexity and communication cost of the extensions to AVG and STDEV are the same as COUNT and SUM, but may have a larger constant factor. The extensions also have bounded accuracy guarantees. This is because the results of AVG and STDEV are derived from the results of COUNT, SUM and SUM_SQR, whose accuracy guarantees are bounded. Note that the accuracy guarantee of SUM_SQR is the same as that of SUM. We omit the proof due to limited space.

8 Experiment Study

We introduce the experimental setup in Sec. 8.1 and present the results on real-world dataset in Sec. 8.2.

TABLE 2: Parameter settings (defaults are marked in bold)

Parameters	Settings
size of data federation $\|P\|$	1, 2, 3, 4, 5 (10^6)
number of data silos m	3, 6, 9, 12, 15
radius of query range r (km)	1, 1.5, 2, 2.5, 3
number of queries nQ	50, 100, 150, 200, 250
approximate ratio of LSR-Forest	0.05, 0.10, 0.15, 0.20, 0.25
least upper bound of LSR-Forest	0.01, 0.02, 0.03, 0.04, 0.05
grid length L (km)	0.5, 1, 1.5, 2, 2.5

8.1 Experiment Setup

Dataset. We use the real-word data collected by three shard mobility companies in Beijing in June 2013. The total size of records in this dataset is over 1TB. Each record has a collected time, the vehicle's location and affiliated company, and the number of carried passengers (as measure attribute). The proportion of the records owned by the three companies is 1 : 1 : 2 The locations of these records fall into an area from 39.5 N 42.0 N and 115.5 E 117.2 E. To simulate real-world bike sharing applications, we vary the size of data federation $|P|$ (i.e., the number of total spatial objects) from 1 million to 5 million by random sampling, because there are already 2.35 million bikes in Beijing in 2017 [22]. To vary the number of data silos (m), we equally split the records of each company to form more data silos, which is a common used way to evaluate the performance of multiple participants [16], [17]. For instance, when m is 6, the records of each company are separated into two data silos, where each data silo has half of the dataset owned by this company. Here we choose $m = 6$ as the default setting, because the federation of real-world applications such as 9-Bike [7] often consists of at least 5 companies. Tab. 2 summarizes the major parameters. The default values are marked in bold.

Queries. We use COUNT and SUM as the aggregation function F, and synthetically generate the circular range R. Specifically, we randomly select a location from the dataset as the center of the circle and vary the radius r from 1km to 3km. To simulate the query frequency in real-world applications, for each radius, we generate a set of nQ independent range aggregation queries, where nQ varies from 50 to 250. These queries arrive in one second. Note that applications such as bike sharing expect over 150 range aggregation queries per second in rush hour [14].

Compared Algorithms. We experiment with the following algorithms: EXACT [2], an exact solution; OPTA, an optimal approximate histogram-based solution with provable guarantees [23]; IID-est (Alg. 2); IID-est+LSR (Alg. 2 with level sampling based local query Alg. 6); NonIID-est (Alg. 3) and NonIID-est+LSR (Alg. 3 with level sampling based local query Alg. 6). The default α and δ of the approximate algorithms are set as $\alpha = 0.1$ and $\delta = 0.01$. The grid size of the grid indices varies from 0.5km to 2.5km. We use multi-threading for communication between the service provider and silos, where the number of threads equals to the number of silos.

Environment. All the experiments are conducted on multiple (4-16) machines. Each machine runs on Ubuntu 18.04LTS with Intel(R) Xeon(R) 2.50GHz processor and 32GB memory. We select one machine as the service provider and the rest as silos. The algorithms are programmed by python 3.6 and

(a) Mean relative error (b) Running time (c) Communication cost (d) Memory of indices

Fig. 3: Performance of COUNT query varying radius of query range r

(a) Mean relative error (b) Running time (c) Communication cost (d) Memory of indices

Fig. 4: Performance of COUNT query varying number of data silos m

(a) Mean relative error (b) Running time (c) Communication cost (d) Memory of indices

Fig. 5: Performance of COUNT query varying grid length L

(a) Mean relative error (b) Running time (c) Communication cost (d) Memory of indices

Fig. 6: Performance of COUNT query varying

tested on a dedicated 1Gb/s network.

Metrics. We focus on the following performance metrics.

Mean relative error . Relative error (RE) is a standard metric to measure the approximation error of one query Q, as is defined below.

$$RE(Q) = \frac{|\text{exact result} - \text{approximate result}|}{\text{exact result}}. \quad (2)$$

To measure the average approximation error of multiple (e.g, nQ) queries, Mean relative error (MRE) is widely applied [24], [25], [26].

$$MRE = \frac{\sum_{Q} RE(Q)}{nQ}. \quad (3)$$

Since the MRE of any exact solution is zero, we omit EXACT in the results on MRE.

(Total) running time . It refers to the total time to answer all the FRA queries (the time to construct the static indices excluded). Since throughput is often defined as the number of queries (nQ) divided by the the running time [11], [27] and nQ is fixed for each compared algorithm, a shorter running time indicates a higher throughput.

(Total) communication cost . It includes the communication cost of sending local query from service provider to silos and the communication cost of sending partial answer from silos to service provider.

(Total) memory of indices . It includes the memory of grid index for service provider and the memory of R-tree/LSR-Forest for silos. The memory cost of OPTA is tiny (less than 0.2M). Thus we omitted it in our experiment.

8.2 Experiment Results

In this section, we show the experiment results for COUNT query. The results for SUM query have the same trend as

(a) Mean relative error (b) Running time (c) Communication cost (d) Memory of indices

Fig. 7: Performance of COUNT query varying

(a) Mean relative error (b) Running time (c) Communication cost (d) Memory of indices

Fig. 8: Performance of COUNT query varying number of queries nQ

(a) Mean relative error (b) Running time (c) Communication cost (d) Memory of indices

Fig. 9: Performance of COUNT query varying size of data federation $|P|$

those on COUNT and we omit them due to limited space.

Impact of radius r. Fig. 3 presents the results of varying the radius of the circular query range for COUNT query. The MRE of all approximate algorithms decreases with the increase of query radius (see Fig. 3a). This is because a large radius involves more samples in the results and decreases the bias of the proposed methods. NonIID-est achieves the smallest MRE followed by NonIID-est+LSR. The MREs of NonIID-est and NonIID-est+LSR are 2.77% on average, and the MREs of IID-est and IID-est+LSR are 5.30% on average. OPTA performs the worst with an average MRE of 8.57%. The use of LSR-Forest does not notably increase MRE. For example, the accuracy gap between NonIID-est and NonIID-est+LSR is below 1%. In terms of running time, our NonIID-est, NonIID-est+LSR, IID-est and IID-est+LSR are up to 3.2×, 5.7×, 6.7× and 85.1× faster than EXACT. The results also show that the running time can be improved by up to 12.7× with our level sampling based local query. As for the communication cost, the results of NonIID-est and NonIID-est+LSR increase slightly as the radius becomes longer, since more grids will intersect with larger query range. Compare with EXACT, all the approximate algorithms significantly reduce the communication cost. As for the memory of indices, IID-est+LSR and NonIID-est+LSR consume 1× more memory because they apply an extra index (LSR-Forest).

Impact of number of data silos m. Fig. 4 illustrates the results of varying the number of data silos m for COUNT query. From Fig. 4a, OPTA also performs the worst. NonIID-est is the most accurate and NonIID-est+LSR is the runner-up. The MREs of all approximate algorithms except OPTA are below 4.08%. In terms of running time, all our approximate algorithms take less time with the increase of m. For instance, when $m = 15$, NonIID-est is 6.3× faster than EXACT. The communication cost of EXACT and OPTA can be up to 15.0× higher than IID-est+LSR and 8.3× higher than NonIID-est+LSR. The memory of indices are relative stable when m increases (Fig. 4d), since the total amount of data is fixed.

Impact of grid length L. Fig. 5 shows the results of varying gird length L for COUNT query. As the grid length increases, the MREs of all our algorithms increase. This is because as the grid length increases, the area of grids intersecting with the query range expands, which increase approximation errors. Among the approximate algorithms, NonIID-est has the smallest MRE, which is up to 0.23%, 3.4%, 3.8% and 5.0% lower than NonIID-est+LSR, IID-est, IID-est+LSR and OPTA, respectively. The gap between IID-est and IID-est+LSR is within 0.5%. In Fig. 5b, NonIID-est+LSR is up to 2.7× faster than NonIID-est and IID-est+LSR is up to 7.8× faster than IID-est with the help of the LSR-Forest index. In terms of communication cost, EXACT and OPTA needs up to 4.3× and 5.8× higher cost than NonIID-est+LSR and IID-est+LSR, respectively. All the algorithms need no more than 1GB memory for indices.

Impact of and . Fig. 6-7 show the results of varying

and for COUNT. Since only local queries by LSR-Forest are affected by and , the MREs of IID-est, OPTA and NonIID-est are stable. Conversely, the MREs ofIID-est+LSR and NonIID-est+LSR increase with the increase of and . In terms of running time, IID-est+LSR and NonIID-est+LSR become faster with a larger . However, the changes in the time cost are marginal with the increase of . The communication cost (Fig. 6c,7c) and memory of indices (Fig. 6d,7d) of all the compared algorithms are stable.

Impact of the number of queries nQ. Fig. 8 shows the results of varying the number of queries for COUNT query. The MREs of all the algorithms are relatively stable, e.g, the changes are lower than 2%. The MRE of NonIID-est+LSR is 4:7% lower than OPTA and 0:27% higher than NonIID-est. As for the running time, IID-est+LSR and OPTA is the fastest. NonIID-est+LSR is 56:1 faster than EXACT. Our IID-est+LSR and NonIID-est+LSR can process more than 250 queries per second, achieving real-time responses ride-hailing services. All the approximate algorithms are at least 2:8 faster than EXACT. In terms of communication cost, our approximate algorithms are still more ef cient. For instance, the communication cost of NonIID-est and NonIID-est+LSR is up to 4:3 lower than the communication cost of EXACT and OPTA. In Fig. 8d, the memory cost of our algorithms is less than 1GB, which is relatively ef cient.

Impact of size of data federation jPj. Fig. 9 presents the results of varying the size of federation jPj for COUNT. The performances of all the algorithms are relatively stable. In Fig. 9a, the MREs of all our approximate algorithms are below 5:41%. But the MRE of OPTA is 7:13% on average. NonIID-est is still the most accurate and LSR-Forest sacri- ces no more than 0:74% of MRE. In terms of running time and communication cost, our approximate algorithms are more ef cient than EXACT and OPTA. For example, IID-est+LSR and NonIID-est+LSR are up to 41:6 and 7:2 faster than EXACT, 5:5 and 4:1 lower communication cost than OPTA. The memory of indices show similar patterns as previous experiments.

Summary of Results. (i) The MREs of our approximate algorithms are small, e.g, the MREs of IID-est and IID-est+LSR are no more than 9:86% and the MREs of NonIID-est and NonIID-est+LSR are no more than 5:29%. (ii) The running time and communication cost of our approximate algorithms are signi cantly more ef cient than the exact solution. IID-est+LSR and NonIID-est+LSR can be up to 85:1 and 7:2 faster than EXACT with 5:5 and 4:1 lower communication cost. Our solutions can process over 250 queries per second, which is t for real-time applications like federated ride-hailing services. (iii) Memory of indices is acceptable for all the algorithms. (iv) Our level sampling based local query notably improves the ef ciency with only a small increase of MRE and memory consumption. It can reduce up to 114:7 running time, with an increase of MRE below 1% and an increase of space below 1GB.

9 Related Work

Our work is related to range aggregate queries in spatial data and query processing in federated data

9.1 Range Aggregate Queries in Spatial Data

Range aggregate processing is extensively studied in spatial databases [2], [5], [28], [29], [30], [31], [32], [33]. In the past few years, distributed systems have been developed to support ef cient range aggregation queries on massive data. For example, Hadoop-GIS [34] and SpatialHadoop [27] extend Hadoop to support the query processing on spatial data in the way of MapReduce. GeoSpark [12], SpatialSpark [35] and Simba [11] extend Spark [36] for in-memory spatial analytics. Others (e.g, DITA [37] and UlTraMan [38]) focus on processing trajectory data. Range aggregate processing is well supported in all these systems.

There are many approximate solutions for querying on streaming data [39], [40], [41]. However, they cannot be easily extended to our problem because we deal with frequent but independent queries on data federation. Parallel/distributed online aggregation is also relevant, which is an interactive solution where a series of estimators with improving accuracy are generated [32], [42]. In our work, however, one-time estimation is needed.

With the growth of data analysis, many industries desire more volumes of data than they have in order to improve their services. As a result, some companies will join in a federation to share their data with others. Since each company owns a horizontal partition of the whole data in the federation, the whole data is called as data federation [16], [17]. Unlike distributed range aggregation [11], [12], [27], [34], [35], range aggregation over data federation is more challenging. Since companies are not likely to exchange their owned data, the techniques of data partitioning in these distributed systems cannot be applied, and new solutions are needed to improve the ef ciency or communication cost.

There are also some works on approximate range aggregation queries with privacy preservation, e.g, [43], [44]. These studies usually assume some data is sensitive and should be protected. However, we assume that data cannot be directly accessed (e.g, based on GDPR). How to preserve privacy on a spatial data federation is a future direction.

9.2 Querying and Analysis on Federated Data

As a type of classical data sharing technique, the concept of federated database system was proposed since the 1980s [45], [46]. A federated database system is virtually integrated from multiple autonomous databases, which provides some collaborative querying interfaces rather than directly sharing raw data. However, traditional federated database systems mainly focus on relational data.

Recently, data federation has emerged as a new solution for data sharing. Speci cally, [16] and [17] study the privacy preservation problem in the relational data federation. [16] applies secure multiparty computation and [17] adopts differential privacy. [47] and [48] propose good demonstrations on query processing over data federation. However, these methods are not suitable for query processing and optimization on large-scale spatial data. Other papers [15], [49], [50] focus on machine learning over data federation, which is out of the scope of this paper.

Overall, all the aforementioned studies focus on query processing and optimization for relational data federation rather than spatial data federation.

10 Conclusion

In this paper, we formulate the FRA problem, which aims at range aggregation queries over large-scale spatial data federation. To enable efficient processing of such queries on large-scale data, we propose to solve the FRA approximately. Specifically, we devise a set of novel indexing and sampling techniques that facilitate parallel processing of FRA queries for high throughput and decrease the average time complexity of local range aggregation query at each data silo to $O(\log \frac{1}{\varepsilon})$. Furthermore, the accuracy guarantee of FRA queries via our solutions is theoretically bounded. We validate the efficiency and effectiveness of our solutions on large-scale real-world datasets. Experiments show that compared with the exact solution, our approximate algorithms can reduce the time cost and communication cost by up to 85.1 and 6.3 , respectively, with average approximate errors below 2.8%. Our solutions can also process more than 250 queries per second, achieving real-time responses for real-world bike-sharing applications.

Acknowledgment

We are grateful to anonymous reviewers for their constructive comments. Yexuan Shi and Yongxin Tong's work are partially supported by the National Key Research and Development Program of China under Grant No. 2018AAA0101100, the National Science Foundation of China (NSFC) under Grant Nos. 61822201, U1811463 and 62076017, the CCF-Huawei Database System Innovation Research Plan No. CCF-HuaweiDBIR2020008B, and the State Key Laboratory of Software Development Environment Open Funding No. SKLSDE-2020ZX-07. Lei Chen's work is partially supported by the Hong Kong RGC GRF Project 16209519, CRF Project C6030-18G, C1031-18G, C5026-18G, AOE Project AoE/E-603/18, Theme-based project TRS T41-603/20R, China NSFC No. 61729201, Guangdong Basic and Applied Basic Research Foundation 2019B151530001, Hong Kong ITC ITF grants ITS/044/18FX and ITS/470/18FX, Microsoft Research Asia Collaborative Research Grant, HKUST-NAVER/LINE AI Lab, Didi-HKUST joint research lab, HKUST-Webank joint research lab grants.

References

[1] R. H. Güting, "An introduction to spatial database systems," *The VLDB Journal*, vol. 3, no. 4, pp. 357–399, 1994.
[2] Y. Tao and D. Papadias, "Range aggregate processing in spatial databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 12, pp. 1555–1570, 2004.
[3] E. T. Zacharatou, H. Doraiswamy, A. Ailamaki, C. T. Silva, and J. Freire, "GPU rasterization for real-time spatial aggregation over arbitrary polygons," *PVLDB*, vol. 11, no. 3, pp. 352–365, 2017.
[4] D. Guo, Y. Zhu, and W. Yin, "OSCAR: a framework to integrate spatial computing ability and data aggregation for emergency management of public health," *GeoInformatica*, vol. 22, no. 2, pp. 383–410, 2018.
[5] D. Zhang and V. J. Tsotras, "Optimizing spatial min/max aggregations," *The VLDB Journal*, vol. 14, no. 2, pp. 170–181, 2005.
[6] M. Tang, Y. Yu, W. G. Aref, A. R. Mahmood, Q. M. Malluhi, and M. Ouzzani, "Locationspark: In-memory distributed spatial query processing and optimization," *CoRR*, vol. abs/1907.03736, 2019.
[7] "9 Bike," *http://www.9bikelm.com*, 2020.
[8] "DiDi Chuxing," *http://www.didiglobal.com/*, 2020.
[9] "Hello Bike," *https://www.helloglobal.com/*, 2020.
[10] "Mobike," *https://mobike.com/cn/*, 2020.
[11] D. Xie, F. Li, B. Yao, G. Li, L. Zhou, and M. Guo, "Simba: Efficient in-memory spatial analytics," in *SIGMOD*, 2016, pp. 1071–1085.
[12] J. Yu, J. Wu, and M. Sarwat, "Geospark: a cluster computing framework for processing large-scale spatial data," in *SIGSPATIAL*, 2015, pp. 70:1–70:4.
[13] J. Yu, Z. Zhang, and M. Sarwat, "Spatial data management in apache spark: the geospark perspective and beyond," *GeoInformatica*, vol. 23, no. 1, pp. 37–78, 2019.
[14] Y. Zhang and Z. Mi, "Environmental benefits of bike sharing: A big data-based analysis," *Applied Energy*, vol. 220, pp. 296 – 301, 2018.
[15] P. Kairouz, H. B. McMahan, B. Avent, and et. al., "Advances and open problems in federated learning," *CoRR*, vol. abs/1912.04977, 2019.
[16] J. Bater, G. Elliott, C. Eggen, S. Goel, A. Kho, and J. Rogers, "Smcql: secure querying for federated databases," *PVLDB*, vol. 10, no. 6, pp. 673–684, 2017.
[17] J. Bater, X. He, W. Ehrich, A. Machanavajjhala, and J. Rogers, "Shrinkwrap: Efficient SQL query processing in differentially private data federations," *PVLDB*, vol. 12, no. 3, pp. 307–320, 2018.
[18] W. H. Qardaji, W. Yang, and N. Li, "Differentially private grids for geospatial data," in *ICDE*, 2013, pp. 757–768.
[19] J. P. Dickerson, K. A. Sankararaman, A. Srinivasan, and P. Xu, "Assigning tasks to workers based on historical data: Online task assignment with two-sided arrivals," in *AAMAS*, 2018, pp. 318–326.
[20] B. Zhao, P. Xu, Y. Shi, Y. Tong, Z. Zhou, and Y. Zeng, "Preference-aware task assignment in on-demand taxi dispatching: An online stable matching approach," in *AAAI*, 2019, pp. 2245–2252.
[21] R. Vershynin, *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge University Press, 2018.
[22] Sun and Yiyun, "Sharing and riding: How the dockless bike sharing scheme in china shapes the city," *Urban Science*, vol. 2, no. 3, p. 68, 2018.
[23] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss, "Optimal and approximate computation of summary statistics for range aggregates," in *PODS*, 2001.
[24] K. Chakrabarti, M. N. Garofalakis, R. Rastogi, and K. Shim, "Approximate query processing using wavelets," in *VLDB*, 2000, pp. 111–122.
[25] S. Chaudhuri, G. Das, and V. R. Narasayya, "A robust, optimization-based approach for approximate answering of aggregate queries," in *SIGMOD*, 2001, pp. 295–306.
[26] Y. Chen and K. Yi, "Two-level sampling for join size estimation," in *SIGMOD*, 2017, pp. 759–774.
[27] A. Eldawy and M. F. Mokbel, "Spatialhadoop: A mapreduce framework for spatial data," in *ICDE*, 2015, pp. 1352–1363.
[28] Y. Tao, C. Sheng, C.-W. Chung, and J.-R. Lee, "Range aggregation with set selection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 5, pp. 1240–1252, 2013.
[29] Y. Tao, G. Kollios, J. Considine, F. Li, and D. Papadias, "Spatio-temporal aggregation using sketches," in *ICDE*, 2004, pp. 214–225.
[30] I. F. V. López, R. T. Snodgrass, and B. Moon, "Spatiotemporal aggregate computation: a survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 2, pp. 271–286, 2005.
[31] D. Papadias, P. Kalnis, J. Zhang, and Y. Tao, "Efficient olap operations in spatial data warehouses," in *SSTD*, 2001, pp. 443–459.
[32] L. Wang, R. Christensen, F. Li, and K. Yi, "Spatial online sampling and aggregation," *PVLDB*, vol. 9, no. 3, pp. 84–95, 2015.
[33] M. Sharifzadeh and C. Shahabi, "Supporting spatial aggregation in sensor network databases," in *SIGSPATIAL*, 2004, pp. 166–175.
[34] A. Aji, F. Wang, H. Vo, R. Lee, Q. Liu, X. Zhang, and J. H. Saltz, "Hadoop-gis: A high performance spatial data warehousing system over mapreduce," *PVLDB*, vol. 6, no. 11, pp. 1009–1020, 2013.
[35] S. You, J. Zhang, and L. Gruenwald, "Large-scale spatial join query processing in cloud," in *ICDE Workshops*, 2015, pp. 34–41.
[36] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, I. Stoica *et al.*, "Spark: Cluster computing with working sets." *HotCloud*, vol. 10, no. 10-10, p. 95, 2010.
[37] Z. Shang, G. Li, and Z. Bao, "DITA: distributed in-memory trajectory analytics," in *SIGMOD*, 2018, pp. 725–740.
[38] X. Ding, L. Chen, Y. Gao, C. S. Jensen, and H. Bao, "Ultraman: A unified platform for big trajectory data management and analytics," *PVLDB*, vol. 11, no. 7, pp. 787–799, 2018.
[39] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and issues in data stream systems," in *PODS*. ACM, 2002, pp. 1–16.