International Journal of *Geo-Information*

Article

A NoSQL–SQL Hybrid Organization and Management Approach for Real-Time Geospatial Data: A Case Study of Public Security Video Surveillance

Chen Wu [1,2], Qing Zhu [1,2,3,*], Yeting Zhang [1,2,*], Zhiqiang Du [1,2], Xinyue Ye [4,*], Han Qin [5] and Yan Zhou [6]

1 State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, Wuhan 430079, China; wuc_oct17@126.com (C.W.); duzhiqiang@whu.edu.cn (Z.D.)
2 Collaborative Innovation Center for Geospatial Technology, 129 Luoyu Road, Wuhan 430079, China
3 Faculty of Geosciences and Environmental Engineering of Southwest Jiaotong University, Chengdu 611756, China
4 Department of Geography, Kent State University, Kent, OH 44240, USA
5 Department of Geography and Geoinformation Science, George Mason University, Fairfax, VA 22030, USA; hqin1021@live.com
6 School of Resource and Environment, University of Electric Science and Technology, Chengdu 611731, China; zhouyan_gis@uestc.edu.cn
* Correspondence: zhuq66@263.net (Q.Z.); zhangyeting@263.net (Y.Z.); xye5@kent.edu (X.Y.)

Academic Editors: Yichun Xie and Wolfgang Kainz
Received: 16 October 2016; Accepted: 16 January 2017; Published: 19 January 2017

Abstract: With the widespread deployment of ground, air and space sensor sources (internet of things or IoT, social networks, sensor networks), the integrated applications of real-time geospatial data from ubiquitous sensors, especially in public security and smart city domains, are becoming challenging issues. The traditional geographic information system (GIS) mostly manages time-discretized geospatial data by means of the Structured Query Language (SQL) database management system (DBMS) and emphasizes query and retrieval of massive historical geospatial data on disk. This limits its capability for on-the-fly access of real-time geospatial data for online analysis in real time. This paper proposes a hybrid database organization and management approach with SQL relational databases (RDB) and not only SQL (NoSQL) databases (including the main memory database, MMDB, and distributed files system, DFS). This hybrid approach makes full use of the advantages of NoSQL and SQL DBMS for the real-time access of input data and structured on-the-fly analysis results which can meet the requirements of increased spatio-temporal big data linking analysis. The MMDB facilitates real-time access of the latest input data such as the sensor web and IoT, and supports the real-time query for online geospatial analysis. The RDB stores change information such as multi-modal features and abnormal events extracted from real-time input data. The DFS on disk manages the massive geospatial data, and the extensible storage architecture and distributed scheduling of a NoSQL database satisfy the performance requirements of incremental storage and multi-user concurrent access. A case study of geographic video (GeoVideo) surveillance of public security is presented to prove the feasibility of this hybrid organization and management approach.

Keywords: real-time geospatial data; NoSQL; RDBMS; data management; public security; hybrid databases; GeoVideo

1. Introduction

With the widespread deployment of ground, air and space sensor sources (internet of things or IoT, social networks and sensor networks), the integrated applications of real-time geospatial data from ubiquitous sensors have already become challenging issues, especially in the case of public security management and the facility management of smart city and present characteristics of 4V categories (volume, velocity, variety and value) [1,2]. Due to the rapid progression of data acquisition tools and Internet techniques, a new generation named the "big data era" has appeared and has exerted a significant influence on spatial science research. Great challenges remain in archiving, retrieving, and mining the massive unstructured sensor data and user-generated datasets efficiently for instant perception and understanding [3,4].

The traditional geographic information system (GIS) aims to map the "snapshot" of the geographical world in a moment in an structured format into commercial relational databases (RDBs) such as Oracle and MySQL, using geospatial data persistence followed by further development and integration of on-demand application functions operated on "outdated" database records [5]. The "current" snapshot in GIS databases still falls out of sync with the real-time input data from the fast-paced constantly changing world due to the input/output (I/O) bottleneck and the high-latency for consistency maintenance in RDB [6,7]. This limits its capability of on-the-fly access of real-time geospatial data for online analysis in real time. Meanwhile, RDB can hardly provide sufficient storage capability in handling the fast-growing big geospatial data because of the "scale-up" system expansion scheme, which requires repeating upgrades of storage devices [8]. In addition, the "store first, then compute" mode stores the incremental unstructured input big data tuple by tuple with large amount of invalid or duplicated data, which not only puts considerable pressure on incremental storage and efficient scheduling, but also cannot meet the requirements of increased spatio-temporal big data linking analysis in big-data-driven GIS studies [9,10]. Therefore, traditional organization and management approaches of geospatial data using RDB-based GIS databases cannot support online analysis in real time.

To address above issues, not only SQL (NoSQL) database management systems (DBMS) are emerging as a new solution. The mainstream NoSQL databases can be classified into four categories in terms of the data storage model: key-value store, document store, column store, and graph store. The emergence of NoSQL DBMS was accompanied by the urgent demand for handling continuous generation, large volumes and unstructured formats of real-time data, with the main characteristics of low-latency accessing, extensibility and low cost of hardware/software/labor [11–13]. For example, Hao et al. used the Hadoop File System (HDFS) to store real-time multi-sensor stream data from IoT to track objects, such as tracking people in indoor spaces using radio frequency identification (RFID) [14]. Kang et al. proposed a sensor-integrated data repository model using MongoDB to integrate heterogeneous IoT data sources such as RFID, sensor and global positioning systems (GPS), and optimize a shard key to maximize query speed and uniform data distribution over data servers [15]. Van et al. compared the read/write performance of sensor data between Cassandra and MongoDB, and concluded that Cassandra is a good choice for relatively larger amounts of sensor data, while MongoDB is good for smaller amounts of sensor data [16]. Kim et al. utilized Redis to solve the high traffics of web services in concurrent access [17]. Although these NoSQL DBMS offer the benefits of high-performance writing/querying for large volume real-time input data, a novel approach for organization of real-time geospatial data is needed to cope with both fast-growing real-time input data and on-the-fly analysis results for real-time geo-processing.

Compared with real-time input data, for which the relevant problems are predominantly focused on the low-latency of large volume data writing and querying as well as fast-growing storage, the problems that must be solved to handle on-the-fly analysis results are predominantly related to the ability of flexible queries and transaction processing. Although RDB are prominent for structured data storage and transaction processing, they can hardly provide sufficient performance in handling the fast-growing big data due to the "scale-up" system expansion scheme [15]. As complements to RDB, NoSQL databases provide a series of features that RDB cannot provide, such as horizontal scalability, memory and distributed index, dynamically modifying data schema, etc. They can store unstructured data efficiently because of easy data schema-modification capability, and require lower server expansion cost than RDB because of the "scale-out" scheme. However, the NoSQL database is lacking in completed atomicity, consistency, isolation and durability (ACID) constraint and support for complicated queries, transaction processing and join operations [18]. Therefore, there is feasibility in synthetically using features of NoSQL and SQL databases to support real-time storage, query, and computation for real-time geospatial data.

To lay the foundations for online GIS analysis in real time, this manuscript presents a hybrid database organization and management approach of SQL database RDBs and NoSQL databases (including the main memory database, MMDB, and distributed files systems, DFS). This hybrid approach makes full use of the advantages of NoSQL and SQL DBMS for the real-time access of input data and structured on-the-fly analysis results. The MMDB facilitates real-time access of the latest input data such as the sensor web and IoT and supports the real-time query for online geospatial analysis. The RDB stores change information like multi-modal features and abnormal events extracted from real-time input data; the DFS on disk manages the massive geospatial data, and the extensible storage architecture and distributed scheduling of a NoSQL database satisfy the performance requirements of incremental storage and multi-user concurrent access. The proposed approach offers the capability of managing large-volume real-time geospatial data and meets the requirements of increased spatio-temporal big data linking analysis. This research will help the research community to conduct big-data-driven GIS studies in a more efficient and productive way. A case analysis of geographic video (GeoVideo) surveillance of public security is presented to prove the feasibility of this hybrid organization and management approach.

The remainder of the paper is organized as follows. Section 2 describes the NoSQL–SQL hybrid organization and management approach. Section 3 takes real-time GeoVideo data as a case study to demonstrate the dataflow and key algorithms in the proposed approach. Section 4 reports several sets of comparative experiment conducted to demonstrate the advantages of the proposed approach. Section 5 describes the conclusion of the study and applications of this research.

2. NoSQL–SQL Hybrid Organization and Management Approach for Real-Time Geospatial Data

2.1. NoSQL–SQL DBMS Hybrid Storage Architecture

This section describes the storage architecture of the proposed approach, which makes full use of the advantages of NoSQL and SQL DBMS for the real-time access of input data and structured on-the-fly analysis results. The main innovative concept of the proposed approach is managing the dataflow of spatio-temporal change information, and uses the change element to connect the three databases. Figure 1 shows the framework of the three types of NoSQL–SQL DBMS. Three sub-structures perform different roles, as follows:

(1) The MMDB is the structure for access. It supports real-time writing and query of the latest input data and synchronizes historical data into the DFS in batches after preprocessing.
(2) The RDB is the secondary part. It stores structured on-the-fly analysis results and change information extracted from input data, uses a trigger mechanism to detect abnormal events in real-time, and instantaneously delivers events to correlative geographic objects and modules by the "subscribe/publish" message mechanism for dynamic geo-processing.
(3) The DFS is the main part. It stores the incremental large volume historical geospatial data with even distribution and a scalable storage environment.

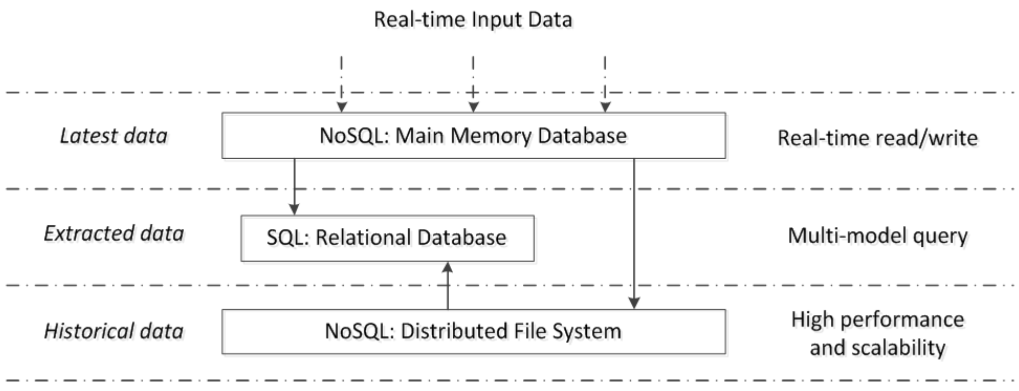


Figure 1. Framework of hybrid storage architecture.

2.1.1. MMDB for Real-Time Access

With the accumulation of massive real-time geospatial data, the access efficiency for real-time geo-processing will decline sharply due to I/O bottlenecks and high consumption for spatio-temporal index maintenance. Besides, original spatio-temporal data has an uneven distributed data value with a large amount of invalid or duplicated data, which need preprocessing, for example information extraction, data cleaning and removal of duplicates. In other words, only non-repeated valid real-time spatio-temporal data is worth serialization. The traditional "store first, then compute" mode on disk-resident SQL DBMS stores the original input big data tuple by tuple first, then queries the current data to implement preprocessing and update preprocessed data finally, which not only poses great pressure on disk throughput, but also influences the real-time performance of accessing and preprocessing for real-time input data.

The read/write speed of memory is significantly faster than that of the disk, so the main memory is a good choice for managing the real-time input data for query and preprocessing within a certain time limit. Owing to limited memory resources and an infinite data stream, storing the entire data stream in bounded memory space is impossible. Therefore, a sliding window is opened to store the most recent sequence data and synchronize the preprocessed data into permanent storage in batches. Compared with the memory cache, MMDB has the advantages of strong stability, high concurrency and scalability. However, the existing MMDB (such as Redis and Cassandra) belong to the NoSQL database, characterized by weak consistency, which means that it can only query recent data, not real-time inserting data. In sum, we can use a fixed-length, singly-linked list in MMDB for each input data channel to support concurrent access and query the list head to obtain the latest data with the shortest latency, which can support real-time storage and query for geo-processing.

2.1.2. DFS for Incremental Historical Data

The large-volume "outdated" spatio-temporal data need to be stored on disk to support further data mining in the future. With the accumulation of incremental data, traditional centralized storage environments of RDB have the problems of I/O bottlenecks and un-scalability, which are unfit for large-scale storage of spatio-temporal data. Being a typical type of NoSQL databases, widely used DFS (such as Hadoop File System, HDFS, and MongoDB) have scalable system architecture consisting of multiple storage servers, which can balance the load pressure of storing and scheduling of massive spatio-temporal data. Many experiments have proven that DFS have greater performance advantages over RDB.

The shard key is the essential factor to determine the even distribution of data storage between multiple storage servers in DFS. Taking MongoDB cluster Mongos as an example, choosing the appropriate shard key with a big number range such as a timestamp can provide better distribution ability. Establishing an ascending index on the timestamp can maintain the latest data in the memory cache, improving the query efficiency of real-time data. In addition, aggregating the spatio-temporal related geospatial data in a region in the same storage server can improve the pre-scheduling efficiency of spatial nearest neighbor queries. Choosing the compound shard key composed of the ascending key (such as a timestamp) and search key (such as spatial grid identity, ID) can aggregate a certain spatio-temporal range data in the same storage server and keep the latest data in the memory cache. Therefore, utilizing DFS with a large cardinal compound shard key can efficiently balance the pressure of read/write operations and index maintenance cost using the distributed storage and distributed index.

2.1.3. RDB for On-The-Fly Extracted Data

For the geospatial data, it is essential to extract specific information in advance before serialization. The specific information includes the interest value, the abnormal change value and the digest value. These three kinds of values are lightweight, and dig out the value of multi-source heterogeneous input data, reduce the amount of storage in a record and increase the speed of queries.

(a) The interest value is the most significant information of the geospatial data in the form of number and characters. For instance, we usually pay attention to some key information that lies in the sensor data rather the sensor data itself, such as the position of a person in the surveillance video who breaks into the enclosed alert area.

(b) The abnormal change value is the specific interest value out of range. This change information often includes the "blasting fuse" to drive the evolution process of geographic environment and what the people care about the most. For example, the value of interest may record the state of a car, but only when the car is speeding, stopped for a long period of time, or moving away from a predefined route, the abnormal change value can trigger the traffic GIS platform to implement emergency disposal.

(c) The digest value is a uniform way to briefly describe the multi-media data like videos, images, audios and signals which is beneficial for de-duplicating and retrieving [19]. For example, if two video frames are the same, the digest values generated by Message Digest 5 Algorithm (MD5) will be the same. Therefore, when retrieving a video stream we just need to compare the digest values and ignore those duplicated video frames, which can greatly reduce scheduling pressure.

As we know, the specific information extracted from real-time input data has multi-modal features, a predefined structure and flexible query requirements, which are suitable to be stored tuple by tuple in RDB. Real-time abnormal change detection is suitable for implementation by the embedded computing function "trigger mechanism" in RDB without external I/O operations, and we can integrate the "subscribe/publish" message mechanism within the trigger to actively dispatch the detected abnormal change information to the correlative module for further processing in real-time. Aiming at the multi-model features of the digest value, we can establish an inverted index, which contains a hash index on all feature semantic elements extracted from geospatial data to support semantic queries in memory with time complexity O(1), and a group of data ID linked list which maps to the entity data.

2.2. Multi-Granularity Organization Method

Existing organization methods mainly concentrate on recording the state of geographic entities or phenomena without considering associations between time-discretized input data. This remarkably decreases the value of the data and the availability of the online GIS analysis based on these data. According to the theory of real-time GIS [20,21], we defined three types of geographic elements: geographic objects, events, and processes. When the interest value of an object, computed from real-time geospatial data, exceeds the predefined threshold, an event is generated and dispatched to related geographic objects to call modules to access and analyze subsequent process data.

(a) The geographic object represents the monitoring entity, like person, vehicle, area or virtual object. For example, the GPS data at a given time reflect the position of a taxi; the monitoring video frame displays the public security situation of a community. The geographic object can be instantiated by the structure of object ID, name, type, lifetime, and description.

(b) The geographic event represents the abnormal state change of a geographic object at a certain moment. For example, if car "A" drives over the speed limit at time "t1"; we can create the event "OverSpeed" visualized by the trajectory point with speed parameter at time "t1" for object "A". A geographic event can be instantiated by the structure of event ID, name, type, timestamp, object ID and a pointer to the abnormal changed input data.

(c) The geographic process represents the activity of a geographic object triggered by event, materialized by clustering the data sequence during one or several time periods around the same theme. For example, the trajectory of a taxi can be divided into several process segments described with the semantics "move" and "stop"; the flood sensor data stream can be segmented into at least three process categories "below normal level", "normal level" and "surpass normal level". The geographic process can be instantiated by the structure of process ID, name, type, time period, object ID, event ID and a pointer to the data segment.

2.3. Unified Scheduling by Uniform ID Structure Design

The division of the geographic elements separates the relevance between each other. It is difficult when designing the ID structure to implicitly present its relationship and logically maintain its integrity to support unified scheduling between hybrid storage environments. The typical ID structure in NoSQL database MongoDB is one such example. The unique identification code of MongoDB comprises 12 bytes that support distributed storage. Among them, 0–3 bytes store the time reference, the absolute number of seconds since 1 January 1970 00:00:00 UTC; 4–6 bytes store the server ID, which is usually a hash value of the machine name; 7–8 bytes store the process identifier of the MongoDB instance; 9–11 bytes store the incremental number. Although the ID structure of MongoDB ensures the uniqueness of each object identifier, there are two problems that must be solved: (1) the geographic object ID, event ID and process ID are generated separately and have no relevance. Therefore, the mapping relationship must be preserved completely; (2) ID duplication cannot be detected. Although the ID structure largely maintains production uniqueness, there inevitably remains a low probability of duplication because of the hash algorithm defect. Such ID structure design must traverse all object IDs to check duplication. Thus, the ID structure is time-consuming and inefficient.

Based on the analysis above, this section defines a novel ID structure as illustrated in Figure 2 that associates multi-granularity geographic elements. The advantage of this novel ID structure is rapid duplication detection and support to distributed production. This ID structure occupies 12 bytes, and is scalable. Among them, byte 0 stores the types of geographic process, object and event. The first byte is the management byte that distinguishes the storing position of these three elements and element type. Bytes 2–5 store the mapping relationship between three elements. For example, bytes 2–3 of the geographic process ID store the number of associated geographic objects, bytes 4–5 store the number of associated geographic events; if the value of this area is "i", it means this process contains "i" geographic elements; changing the value of these 2 bytes to the range of [0, i − 1], the geographic process ID is changed to the relevant geographic element logical ID; then using a mapping table to transform the logical ID to the real geographic element ID. Bytes 6–7 store an incremental number and identify the distributed working space. The working space is the minimum administrative unit for managing geographic processes, objects and events. Each working space has different identifiers in these three bytes, but the processes, objects and events in the same working space have the same identifier in those bytes. It is easier to check duplication based on the working space and support distributed production. Bytes 8–11 store the incremental number. Every ID in each type has a different value in those 4 bytes. In general, bytes 6–11 ensure that the ID is globally unique in a monitored area and are the key part to execute the query operation.

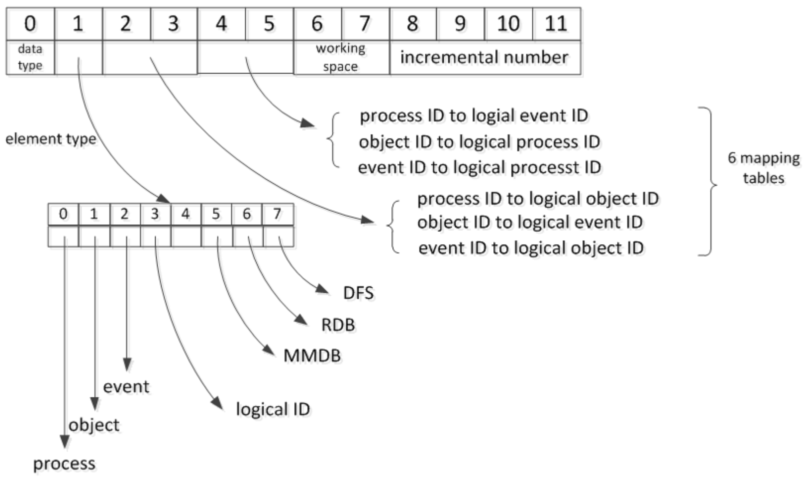

0 1 2 3 4 5 6 7 8 9 10 11
data type
working space
incremental number
element type
process ID to logial event ID
object ID to logical process ID
event ID to logical processt ID
process ID to logical object ID
object ID to logical event ID
event ID to logical object ID
6 mapping tables
0 1 2 3 4 5 6 7
DFS
RDB
MMDB
event
logical ID
object
process

Figure 2. Uniform geographic element identity (ID) structure design for unified scheduling.

3. Case Study of Real-Time GeoVideo Data in Hybrid NoSQL–SQL DBMS

GeoVideo is typical real-time geospatial data, which maps video frames into a geographic space. In this section, we take public security video surveillance as an example to illustrate the dataflow of massive GeoVideo data between typical NoSQL databases (MMDB Redis and DFS MongoDB) and the SQL database (RDB MySQL).

3.1. Dataflow in Hybrid Databases

Real-time motion-area detection and foreground extraction are critical procedures for public security GeoVideo surveillance, which transform the original video frames into geographic elements including geo-referenced vehicles, human bodies and static background for higher-level visual processing and GIS analysis [22–24]. Therefore, linking storage and computation of real-time GeoVideo stream is an effective way to reduce the response time in emergency. Figure 3 shows the dataflow of GeoVideo data between Redis, MySQL and MongoDB.

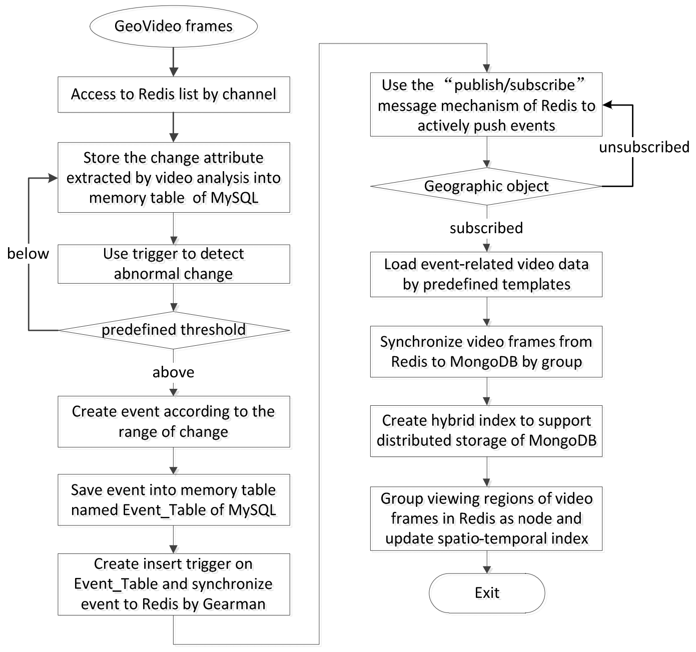

GeoVideo frames
Access to Redis list by channel
Store the change attribute extracted by video analysis into memory table of MySQL
Use trigger to detect abnormal change
below
predefined threshold
above
Create event according to the range of change
Save event into memory table named Event_Table of MySQL
Create insert trigger on Event_Table and synchronize event to Redis by Gearman
Use the "publish/subscribe" message mechanism of Redis to actively push events
unsubscribed
Geographic object
subscribed
Load event-related video data by predefined templates
Synchronize video frames from Redis to MongoDB by group
Create hybrid index to support distributed storage of MongoDB
Group viewing regions of video frames in Redis as node and update spatio-temporal index
Exit

Figure 3. Dataflow of geographic video (GeoVideo) data in hybrid storage environment.

Step 1: Access the GeoVideo stream and convert the video stream into video frames according to frame rate. Calculate the field of view (FOV) based on the attitude angle and focal length of the camera.

Step 2: The sequence of GeoVideo frames is stored in Redis, and each video channel corresponds to a Redis list, which supports queries of the recent period of video data. The video analysis system obtains the latest video frames from the head of the list for real-time motion region detection and foreground extraction, and maintains a constant list length by a first-in-first-out (FIFO) replacement algorithm.

Step 3: Store the change attribute extracted by a comparative analysis of video frames in the memory table named Change_Attribute_Table of MySQL, and create an insert trigger to perform real-time monitoring of abnormal changes. If the change value exceeds predefined threshold values, the trigger creates a corresponding event according to the range of change and inserts the event into the memory table named Event_Table of MySQL.

Step 4: Create an insert trigger in Event_Table and use the distributed scheduling system Gearman to synchronize the event in Redis as a cache. The event tuple in MySQL maps to the data structure of hash on Redis and takes the globally unique event ID as the hash key.

Step 5: Use the "subscribe/publish" message mechanism of Redis to actively push events to the relevant geographic objects. Take the event category as "channel" to execute event push operations.

Step 6: Geographic objects receive subscribed events, analyze properties of the events, and automatically load spatio-temporal related GeoVideo data using predefined event templates.

Step 7: Group the preprocessed video frames in Redis with geographic process semantics and write to the MongoDB cluster Mongos. Choose the compound shard key—"day+cameraID" and create hybrid index on these two attributes to support distributed storage of massive GeoVideo data.

Step 8: Group the FOV of video frames in Redis, as the motion trajectory of the camera. Calculate the 3D minimum boundary rectangle of these FOVs and update the spatio-temporal index while synchronizing video frames from Redis to MongoDB.

3.2. Structure and Algorithm of Interoperability

3.2.1. Change Detection and Event Trigger

The abnormal change value extracted from the GeoVideo stream in Redis is the trigger to execute video analysis. Using the "trigger mechanism" in RDB to link storage and computation of the GeoVideo data can actively detect abnormal changes and create events in real time without external I/O operations. In video surveillance of a museum, the distance between a visitor and an exhibit, stay duration in an exhibition hall, exhibit movement and fire hazard are all important factors in the determination of exhibit safety. For example, in monitoring the distance between a visitor and an exhibit, varying distances will trigger different levels of security events. Table 1 describes the structure of the table Safe_Distance, which records the distance between an exhibit and visitor extracted from a video frame. Table 2 describes the structure of the table Event_Safe_Distance, which records events detected from the table Safe_Distance by checking the distance value between an exhibit and visitor. The trigger Trigger_Safe_Distance (Algorithm 1) checks abnormal distances in the table Safe_Distance, and inserts the events into the table Event_Safe_Distance. The table Safe_Distance, table Event_Safe_Distance, and the trigger Trigger_Safe_Distance in MySQL constitute an integrated procedure of real-time change detection and event trigger.

Table 1. Description of the Safe_Distance table.

Element	Type	Description
_id	RowID	Auto-adding identity of change element
exhibit_id	int	Unique identity of exhibit in museum
visitor_id	int	Unique identity of visitor in museum
distance	float	Distance between visitor and exhibit
camera_id	int	Unique identity of camera that captures the scene

Table 2. Description of the Event_Safe_Distance table.

Element	Type	Description
eid	string	Unique identity of event
event_type	int	Type of event
exhibit_id	int	Identity of exhibit related to this event
visitor_id	int	Identity of visitor related to this event
distance	float	The distance between visitor and exhibit
camera_id	int	Unique identity of camera that captures the scene

Algorithm 1: Trigger_SafeDistance (Tuple DistanceValue)

```
1. create trigger Trigger_Safe_Distance
2. after insert on Table Safe_Distance
3. foreach new row
4.   begin
5.     if newTuple.distance < safe_distance_level1 then
6.       insert into Event_Safe_Distance values(event_type1, newTuple.attributes)
7.     elseif newTuple.distance < safe_distance_level2 then
8.       insert into Event_Safe_Distance values(event_type2, newTuple. attributes)
9.     elseif newTuple.distance < safe_distance_level3 then
10.      insert into Event_Safe_Distance values(event_type3, newTuple. attributes)
11.    else
12.      insert into Event_Safe_Distance values(event_type4, newTuple. attributes)
13.    end if-then
14.  end-begin
15. end-foreach
```

3.2.2. Event Subscribing and Publishing

Use of the message mechanism of "subscribe/publish" can actively dispatch events to related geographic objects in the first time. We define the insert trigger Dispatch_Event (Algorithm 2) in the table Event_Safe_Distance of MySQL. In the stored procedure of trigger Dispatch_Event, we use the distributed scheduling system Gearman to synchronize events from MySQL to Redis, by editing the Gearman Worker named "SyncAndDispatchEvent" (Algorithm 3) and using the embedded "subscribe/publish" message mechanism of Redis to push the real-time generated events to related geographic objects. In the video surveillance application, we separate each event category as a channel, and various safety departments are subscribed to different types of events. For example, a museum safety department receives all levels of security incidents, but the local police station only receives those at a higher level. Through real-time publishing of security events, subscribers receive event notifications and query event-related real-time and historical GeoVideo data from Redis and MongoDB for comprehensive analysis.

Algorithm 2: Trigger_DispatchEvent (Tuple SafeDistanceEvent)

```
create trigger Dispatch_Event
after insert on Table Event_Safe_Distance
foreach new row
  begin
    set @ret=gman_do_background (GearmanWorker 'SyncAndDispatchEvent',
    json_object(newTupleInMySQL.attributes as newKeyValuePairInRedis.attributes))
  end-begin
end-foreach
```

Algorithm 3: GearmanWorker_SyncAndDispatchEvent (Tuple SafeDistanceEvent)

```
$worker = new GearmanWorker()
$worker->addFunction('SyncAndDispatchEvent')
$redis = new Redis()
$redis->connect(ip, port)
while($worker->work())
begin
  function SyncAndDispatchEvent (Tuple $job)
    global $redis
    $workString = $job->workload()
    $work = json_decode($workString)
    $redis->hmset(key: "attributeName", value: $work->attribute)
    $redis->publish(channel: $work->event_type, $work->eid)
  end-function
end-begin
```

4. Experimental Study

In this section, we presented experiments using the organization and management approach described above and analyze performance results in terms of real-time GeoVideo data accessing, abnormal event detection and dispatch and massive historical data-distributed storage. In Section 4.1, we describe the software and hardware environment, the dataset and other preparations involved in the experiments. Detailed experiments were presented in Section 4.2.

4.1. Experimental Setting

All experiments were performed on two Dell OPTIPLEX 9020 workstations, each of which created three virtual machines as work nodes. Each node had a 4-Core Intel I7-4790M 3.60 GHz CPU with eight hardware threads and 4 GB of RAM. The two workstations communicated via a gigabit network. All six nodes used a 64-bit Linux operating system (CentOS Enterprise Server). A 64-bit MongoDB, 64-bit Redis, 64-bit MySQL, task distribution system Gearman and 64-bit OpenCV were used to implement the hybrid organization method described above on the two work nodes for real-time GeoVideo data.

In the experiments, we chose 58 randomly distributed cameras both inside and outside the office building. Each one of the cameras recorded 48 h at the sampling rate of 25 frames per second, with one geo-referenced frame per second due to the sampling rate of GPS and the compass (the digital compass can achieve 30 or 40 readings per second but GPS can only get one sample per second). These were key parameters for calculating spatial information of GeoVideo, such as FOV and position of the monitoring object. Therefore, we had a dataset with about 250 million video frames and 10 million geo-referenced key frames; each frame size is about 100 K. The change detection of video frames, including motion-area detection and foreground extraction were implemented by an open-source feature extraction algorithm library OpenCV.

In order to validate the proposed method, we carried out a series of experiments using a sample GeoVideo data set. The first experiment compared updating and query performance between the Redis-MongoDB-based repository, MongoDB-based repository and MySQL-based repository in order to check whether the hybrid method outperformed independent representative RDB and DFS in real-time accessing. The second experiment compared abnormal event detection and dispatch performance of active detect-push and regular scan in order to validate linking of storage and computation by close visit, greatly reducing response time. The third experiment compared data distribution and query performance of different shard key selection in order to validate the choice of compound shard key and checked whether an increase in the number of MongoDB shards improved access performance.

4.2. Experimental Results

4.2.1. Real-Time Accessing

This test compared the real-time accessing performance of the suggested Redis-MongoDB-based repository with an independent MongoDB-based repository and MySQL-based repository. For this test, we configured a MongoDB-based repository which has one config server, three shard servers and one mongs router, and a Redis-based repository and a MySQL-based repository on a single node. We used three update/query proportions (75% updating and 25% querying, 50% updating and 50% querying, and 25% updating and 75% querying) and got the average value to measure the update/query throughput and response latency of querying latest data in the same way.

From Figure 4a–c, we can conclude that the accessing efficiency and the latency for querying latest data of Redis-MongoDB-based repository was significantly better than the other two. The Redis-MongoDB-based repository outperformed the MongoDB-based repository, while the MongoDB-based repository outperformed the MySQL-based repository, which was because Redis solved the problems of I/O bottleneck and high consumption for global index maintenance and MongoDB simplified the complex operations for maintaining strong consistency for real-time updating by keeping eventual consistency. In general, the accessing throughput of the three repositories kept stable initially and then declined gradually with the increasing amount of data stored in the database. The number of GeoVideo data volume had a small effect on Redis-MongoDB-based repository and a great effect on the MySQL-based repository. Therefore, separating real-time and historical video data, using the MMDB to manage real-time data and utilizing DFS to manage massive historical data was an efficient way to satisfy the requirements of real-time access.

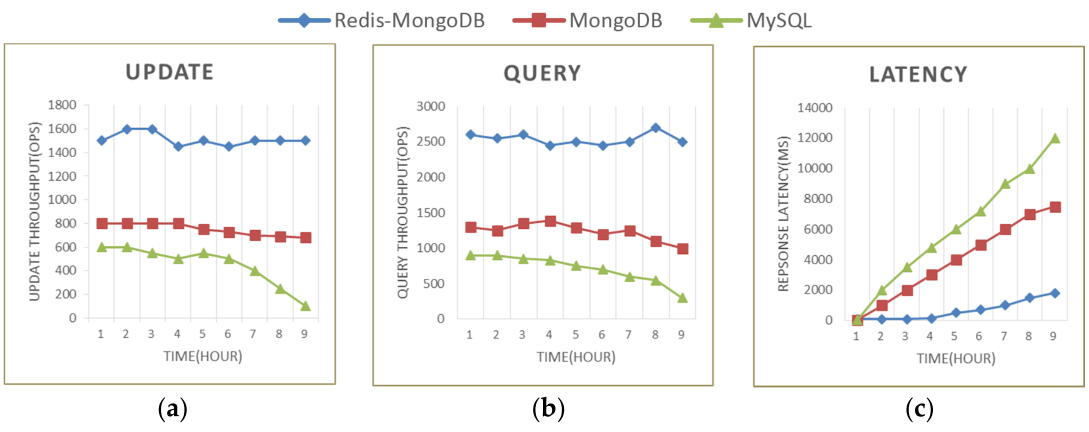
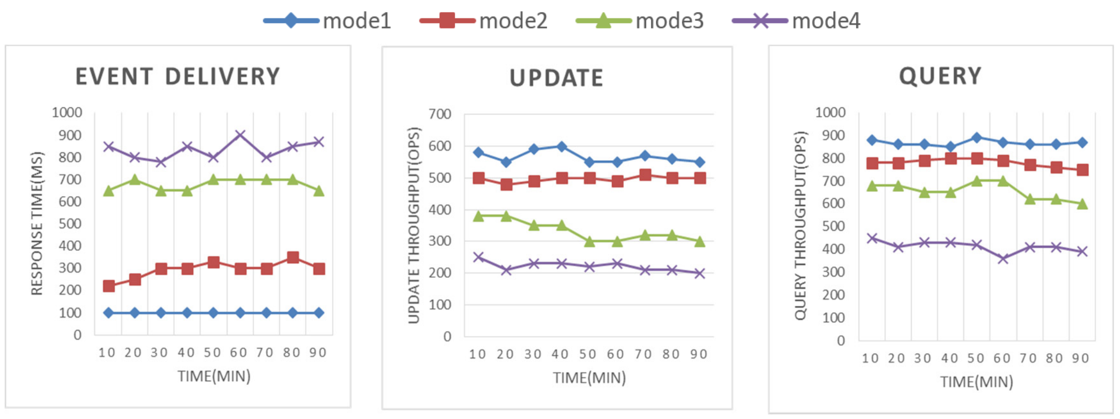


Figure 4. Real-time accessing performance between Redis-MongoDB-based, MongoDB-based and MySQL-based repositories. (**a**) Update throughput of three repositories; (**b**) Query throughput of three repositories; (**c**) Latency for getting latest data of three repositories.

4.2.2. Event Detection and Dispatch

To demonstrate the performance of real-time abnormal change detection and event dispatch, a comparative trial of time cost of event detection and dispatch was designed in four modes as follows. Mode 1: Use of the trigger mechanism of MySQL in the memory table Safe_Distance to detect abnormal change, and utilization of an embedded Gearman worker in the trigger of memory table Event_Safe_Distance to push events to related geographic objects by "subscribe/publish" message mechanism without external I/O operations. Mode 2: Use of the trigger mechanism of MySQL to detect abnormal change and utilization of the embedded Gearman worker to push events to related geographic objects, however table Safe_Distance and table Event_Safe_Distance were stored on disk. Mode 3: Use of the trigger mechanism of MySQL to detect abnormal change in memory tables Safe_Distance and storage of different types of events in different memory tables. Then, applications periodically (500 ms) scanned event tables to get the latest events. Mode 4: Use of the procedure functions of MySQL to periodically (500 ms) detect abnormal change and storage of calculated events in the memory table Event_Safe_Distance. Then, the trigger in memory table Event_Safe_Distance dispatched the events to related geographic objects and modules using the embedded Gearman worker.

From Figure 5a, we can conclude that the event detection and dispatch efficiency of mode 1 was significantly greater than the other three. The comparison between mode 1 and 2 demonstrated that memory tables can greatly decrease the time cost of event trigger and event delivery. The comparison between mode 1 and 3 as well as mode 1 and mode 4 revealed that active operations of event detection and event dispatch were more efficient. Figure 5b,c show that mode 1 consumed the least resources of MySQL, and it had little influence on the accessing performance of MySQL when compared to the other three modes.



(**a**) (**b**) (**c**)

Figure 5. Comparison between four modes of abnormal change detection and event dispatch. (**a**) Time cost of event detection and delivery of four modes; (**b**) Update throughput of MySQL under the running of event detection and delivery of four modes; (**c**) Query throughput of MySQL under the running of event detection and delivery of four modes.

4.2.3. Historical Data Distribution

This experiment aimed to validate whether the MongoDB cluster Mongos guaranteed an even distribution of GeoVideo data, better query performance and scalable storage. We configured a MongoDB cluster Mongos which had one config server, three shard servers, one mongs router and an alternative extendible shard server. A comparison between three shard keys: compound key "day+cameraID", hash key "cameraID" and incremental key "day" was made.

Figure 6a,b showed that compound shard key had better accessing performance when compared to the other two types of shard key. Although the hash key had a better even distribution and updating efficiency of GeoVideo data, it had a bottleneck for queries due to random distribution of GeoVideo data without clustering and an invalid distributed index. The incremental key made the updating operation focus on one shard server that recorded the latest data, which resulted in an update bottleneck that did not allow scalable storage to fully come into play. Figure 6c showed that an increase in the number of shards improved update performance which means MongoDB had a strong capability to scale out. The shard number had a small performance improvement on the incremental key and a great performance improvement on the compound key and hash key.

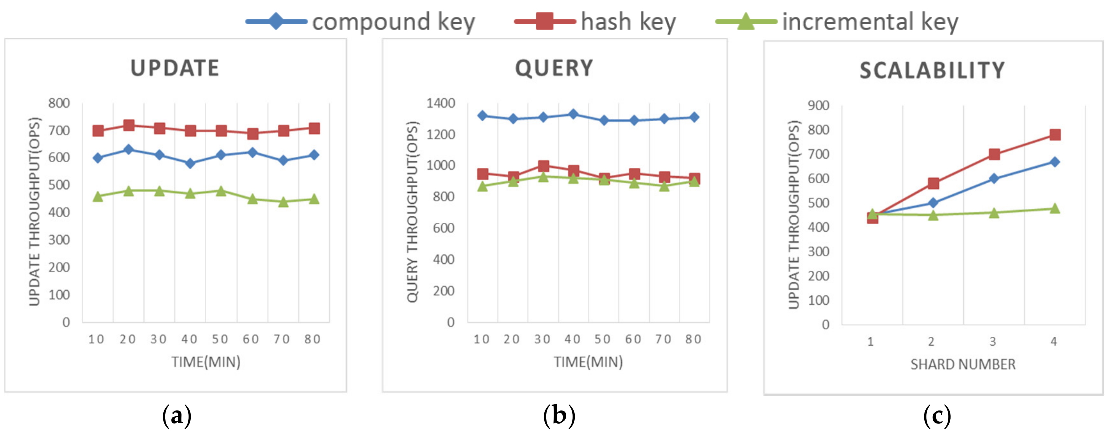
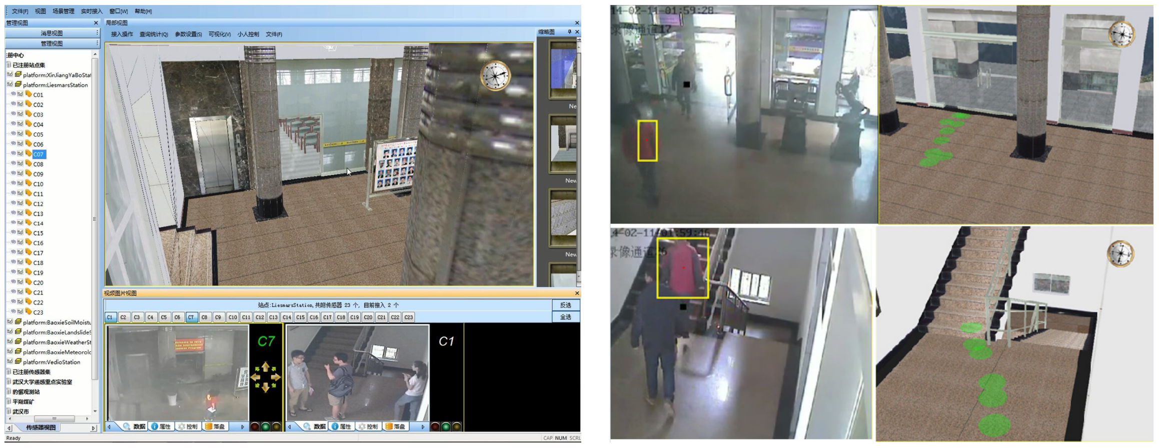


Figure 6. Accessing performance under a different shard key. (**a**) Update throughput of MongoDB using a different shard key; (**b**) Query throughput of MongoDB using a different shard key; (**c**) Performance improvement with a different shard number.

5. Conclusions

The complex structures of geospatial systems have a pressing need for appropriate management [25,26]. Recent developments in information technology commonly referred to as "big data", along with the related fields of data science and analytics are needed to process, analyze, and determine the value of the overwhelming amounts of geospatial data [3,27]. The existing geospatial analysis methods have been developed primarily in the context of small data. Yet, all of the processes of interest to the general public and decision makers operate in the real time and heterogeneous context. The imbalance between rich data products and poor data utilization calls attention to the techniques of real-time data access and management. The hybrid NoSQL–SQL organization and management approach of real-time geospatial data makes full use of the advantages of the real-time access operation of NoSQL MMDB for various heterogeneous input data, flexible queries and transaction processing of SQL RDBMS to support the access of on-the-fly analysis results, and scalable ability of NoSQL DFS for massive data. This approach is considered effective for supporting real-time storage, query, and computation in real-time GIS. Aimed at difficulties in the linkage between storage and computation within GIS online analysis, this paper presented an internal and external collaborative storage strategy by separately managing real-time and historical geospatial data, and designs a workflow integrating real-time change detection and active event delivery for driving geographic process evolution. A novel ID structure was also designed to associate the multi-granularity geographic elements for unified scheduling in hybrid NoSQL–SQL DBMS. Additionally, using the subscribe/publish message mechanism to map the relationships between geographic objects, events and processes, the method can efficiently decrease the latency of collaborative scheduling of real-time and historical spatio-temporal data. Experimental results from concrete application of GeoVideo based on Redis, MongoDB and MySQL demonstrate the practicality and reliability of this method in supporting real-time GIS applications.

This NoSQL–SQL hybrid organization approach is an important foundation in the real-time GIS platform for environment monitoring. This system has been successfully applied for GeoVideo monitoring and trajectory tracking and provides public security decision support for the security department (see Figure 7). The data management approach has facilitated the real-time access and abnormal change detection of big GeoVideo data. The developed organization and management approach of real-time geospatial data will enable advancements in a broad spectrum of applications by assisting researchers in tackling the challenges posed by big data.



(**a**) (**b**)

Figure 7. The interface of the real-time geographic information system (GIS) monitoring platform, which shows a visualization of the real-time geo-referenced video data that maps its field of view (FOV) to the 3D scene for managing massive video data from spatial perspective. (**a**) Choosing real-time GeoVideo data that covers lobby in the first floor; (**b**) Real-time detection of moving objects and tracking trajectory.

Acknowledgments: This work was supported by National Natural Science Foundation of China (Grant No.41571392, 41471320, 41471332) and US National Science Foundation (Grant No.1637242).

Author Contributions: Chen Wu wrote the first draft of this article and made the key experiments. Qing Zhu contributed to the concept and framework of the geospatial data organization and management approach. Yeting Zhang, Zhiqiang Du and Yan Zhou contributed a lot to the modification and optimization of the procedures and case analysis. Xinyue Ye updated the manuscript and concepts. Han Qin contributed to the innovative idea of the experiments and language polishing.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kwan, M.P.; Lee, J. Emergency response after 9/11: The potential of real-time 3D GIS for quick emergency response in micro-spatial environments. *Comput. Environ. Urban* **2005**, *29*, 93–113. [CrossRef]
2. Zerger, A.; Smith, D.I. Impediments to using GIS for real-time disaster decision support. *Comput. Environ. Urban* **2003**, *27*, 123–141. [CrossRef]
3. Zhang, F.; Zheng, Y.; Xu, D.; Du, Z.; Wang, Y.; Liu, R.; Ye, X. Real-time spatial queries for moving objects using storm topology. *ISPRS Int. J. Geo-Inf.* **2016**, *5*. [CrossRef]
4. Looking Forward: Five Thoughts on the Future of GIS. Available online: http://www.esri.com/news/arcwatch/0211/future-of-gis.html (accessed on 7 August 2016).
5. Xu, W.; Zhu, Q.; Zhang, Y.; Ding, Y.; Hu, M. Real-Time GIS and its application in indoor fire disaster. *ISPRS Arch.* **2013**, *XL-2/W2*, 121–127. [CrossRef]
6. Pelekis, N.; Theodoulidis, B.; Kopanakis, I.; Theodoridis, Y. Literature review of spatio-temporal database models. *Knowl. Eng. Rev.* **2004**, *19*, 235–274. [CrossRef]
7. Breunig, M.; Türker, C.; Böhlen, M.H.; Dieker, S.; Güting, R.H.; Jensen, C.S.; Relly, L.; Rigaux, P.; Schek, H.J.; Scholl, M. Architectures and implementations of spatio-temporal database management systems. In *Spatio-Temporal Databases: The CHOROCHRONOS Approach*, 1st ed.; Frank, A.U., Sellis, T., Koubarakis, M., Eds.; Springer: Berlin, Germany, 2003; pp. 264–314.

8. Shvachko, K.; Kuang, H.; Radia, S.; Chansler, R. The hadoop distributed file system. In Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies, Washington, DC, USA, 3–7 May 2010.
9. Abadi, D.J.; Ahmad, Y.; Balazinska, M.; Çetintemel, U.; Cherniack, M.; Hwang, J.H.; Lindner, W.; Maskey, A.S.; Rasin, A.; Ryvkina, E.; et al. The design of the borealis stream processing engine. In Proceedings of the 2005 CIDR Conference, Asilomar, CA, USA, 4–7 January 2005.
10. Arasu, A.; Babu, S.; Widom, J. The CQL continuous query language: Semantic foundations and query execution. *VLDB J.* **2006**, *15*, 121–142. [CrossRef]
11. Gyllstrom, D.; Wu, E.; Chae, H.J.; Diao, Y.; Stahlberg, P.; Anderson, G. SASE: Complex event processing over streams. In Proceedings of the 3rd Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, 7–10 January 2007.
12. Demers, A.J.; Gehrke, J.; Panda, B.; Riedewald, M.; Sharma, V.; White, W.M. Cayuga: A general purpose event monitoring system. In Proceedings of the 3rd Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, 7–10 January 2007.
13. Gedik, B.; Andrade, H.; Wu, K.L.; Yu, P.S.; Doo, M. SPADE: The system s declarative stream processing engine. In Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, Vancouver, BC, Canada, 9–12 June 2008.
14. Hao, X.; Jin, P.; Yue, L. Efficient storage of multi-sensor object-tracking data. *IEEE Trans. Parall. Distr.* **2015**, *27*, 2881–2894. [CrossRef]
15. Kang, Y.; Park, I.; Rhee, J.; Lee, Y. MongoDB-based repository design for IoT-generated RFID/Sensor big data. *IEEE Sens J.* **2015**, *16*, 485–497. [CrossRef]
16. Sipke, V.D.V.J.; Bram, V.D.W.; Meijer, R.J. Sensor data storage performance: SQL or NoSQL, physical or virtual. In Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing, Honolulu, HI, USA, 24–29 June 2012.
17. Kim, C.H.; Park, K.W.; Choi, Y.L. Web service performance improvement with the Redis. *J. Korea Inst. Inf. Commun. Eng.* **2015**, *19*, 2064–2072. [CrossRef]
18. Jiang, L.; Xu, L.D.; Cai, H.; Jiang, Z. An IoT-Oriented data storage framework in cloud computing platform. *IEEE Trans. Ind. Inform.* **2014**, *10*, 1443–1451. [CrossRef]
19. Li, T.; Liu, Y.; Tian, Y.; Shen, S.; Mao, W. A storage solution for massive IoT data based on NoSQL. In Proceedings of the 2012 IEEE International Conference on Green Computing and Communications, Besancon, France, 20–23 November 2012.
20. Gong, J.; Li, X.; Wu, H. Spatio-temporal data model for Real-Time GIS. *AGCS* **2014**, *43*, 226–232.
21. Li, X.; Yang, J.; Guan, X.; Wu, H. An event-driven spatiotemporal data model (e-st) supporting dynamic expression and simulation of geographic processes. *Trans. GIS* **2014**, *18*, 76–96. [CrossRef]
22. Lu, Y.; Shahabi, C.; Kim, S.H. Efficient indexing and retrieval of large-scale geo-tagged video databases. *Geoinformatica* **2016**, *20*, 829–857. [CrossRef]
23. Milosavljević, A.; Rančić, D.; Dimitrijević, A.; Predić, B.; Mihajlović, V. Integration of GIS and video surveillance. *Int. J. Geogr. Inf. Sci.* **2016**, *30*, 2089–2107. [CrossRef]
24. Lewis, P.; Fotheringham, S.; Winstanley, A. Spatial video and GIS. *Int. J. Geogr. Inf. Sci.* **2011**, *25*, 697–716. [CrossRef]
25. Al-Dohuki, S.; Kamw, F.; Zhao, Y.; Ma, C.; Wu, Y.; Yang, J.; Ye, X.; Wang, F.; Li, X.; Chen, W. SemanticTraj: A New Approach to Interacting with Massive Taxi Trajectories. *IEEE Trans. Vis. Comput. Graph.* **2017**, *23*, 11–20. [CrossRef] [PubMed]
26. Huang, X.; Zhao, Y.; Yang, J.; Zhang, C.; Ma, C.; Ye, X. TrajGraph: A Graph-Based Visual Analytics Approach to Studying Urban Network Centralities Using Taxi Trajectory Data. *IEEE Trans. Vis. Comput. Graph.* **2016**, *22*, 160–169. [CrossRef] [PubMed]
27. Ye, X.; Huang, Q.; Li, W. Integrating Big Social Data, Computing, and Modeling for Spatial Social Science. *Cartogr. Geogr. Inf. Sci.* **2016**. [CrossRef]