Storage Solution of Spatial-Temporal Data for Water Monitoring Infrastructures used in Smart Cities

Catalin Negru[1], Florin Pop[1,2], Mariana Mocanu[1] and Valentin Cristea[1]
[1]Computer Science Department, University Politehnica of Bucharest, Romania
[2]National Institute for Research and Development in Informatics (ICI), Bucharest, Romania
[3] Department of Computer Science, University of Salerno, Italy
Emails: catalin.negru@cs.pub.ro, florin.pop@cs.pub.ro,
mariana.mocanu@cs.pub.ro, valentin.cristea@cs.pub.ro

***Abstract*—Smart Cities make use of complex utility infrastructure systems such as water distribution. These infrastructures have raised management costs due to improper resource utilization and maintenance or aging infrastructure. New innovative information systems are essential to overcome these issues. Due to the continuous growth of these infrastructure systems in scope and scale, lead to the usage of spatial-temporal networks. The data generated have to be stored in an efficient and scalable way in order to be used by applications such as spatial and temporal analytics or monitoring applications.**

I. Introduction

Due to the fact that smart-cities become more interconnected and instrumented we face with rich spatial-temporal data. Environment monitoring (e.g.river water, air, noise), trajectories of taxis recorded by GPS devices, social events with location and timestamps represent spatio-temporal data. Performing data analytics on these data pose new challenges such as large data volume, data uncertainty, complex relationship, and system dynamics and bring new opportunities such as discovering predictive patterns related to time and space.

There are four sectors that need to be integrated in order to achieve the vision of a smart city: smart water, smart public services, smart energy and smart buildings. The water distribution system represents a critical infrastructure in a smart city and management systems and applications are designed to collect data about the flow, pressure and distribution of a city's water. Further, it is critical that the consumption and forecasting of water use is accurate [1].

Water management infrastructures can manage different components such as supervisory control and data acquisition (SCADA) systems, sensors and meters that produce data in diverse formats and scales. Furthermore, data can come from geographic information systems, or third party sources such as stakeholders and industrial water users.

The systems for water resource management need fast, scalable, reliable and efficient storage systems, in order to achieve cost reduction, faster and better decision making in case of accidents of failures in the system.

Moreover, the monitoring of water distribution systems implies large volume of heterogeneous information (e.g. spatial, sensor and multimedia data) with temporal dimensions. It is very important to acquire, store, transmit and analyze data in order to respond in real-time and to alert possible affected population in case of pollution accidents.

With the explosion of collected data in smart cities it is necessary to provide an integrated and interoperable system for storage and data processing in order to provide support in crisis management situations. Mobile and Cloud based geographical information systems technologies can improve disaster management. We have to combine mobile applications, existing electronic services and data repositories, in an architecture based on Cloud solutions and existing Big Data approaches;

It is essential to have a scalable environment with flexible information access, easy communication and real time collaboration from all types of computing devices, including mobile handheld devices (e.g. smart phones, PDAs and tablets (iPads)). Also, it is mandatory that the system must be accessible, scalable, and transparent from location, migration and resource perspectives.

This paper is organized as follows: In Section I is presented a short introduction related to the need of efficient storage systems that have to support processing operations on spatial-temporal data. In Section II is presented related work. In Section III is presented the proposed architecture. Finally in Section IV are presented conclusions of the paper.

II. Related Work

In order to integrate heterogeneous data sources, we need a context-aware approach and an appropriate model to aggregate, semantically organize and access large amounts of data. Data handling methods have been applied in several areas including water network data analysis and modeling, water quality analysis, energy-water relationship, efficiency modeling of regional water [2].

Applications for spatio-temporal data processing can have demanding requirements, such as moving large complex data objects such as digital elevation models, moving points in space or time series data. Moreover, different datasources must be combined in order to provide relevant results.

A research direction is to modify relational database management systems in order to store efficiently spatial-temporal data. Another approach is to design new engines for database systems that can handle spatial-temporal data processing challenges. For instance, the authors of [3] propose a prototype database system and a framework of a generic index tree for spatial-temporal data. The main idea is that the index is based on the conceptual behavior of data and make use of R-trees and can be defined without any predefined hard-wired spatial or temporal data types such as intervals or rectangles. Results show that there are necessary only two important properties, OVERLAP and SPLIT. First, it checks for spatial or temporal overlap between objects and second provides hierarchical decomposition of the data space into subspaces.

In [4] the authors propose HadoopDB a hybrid system that combine parallel database management systems and MapReduce systems. In this way the running applications can benefit from the performance and efficiency of parallel databases and the scalability, fault tolerance, and flexibility of the MapReduce systems. The main idea is connection of multiple database systems using Hadoop as the task coordinator and to communicate through network layer. In this way queries can be parallelized with MapReduce. The drawback of this approach is that HadoopDB does not match the performance of parallel database systems.

Another research direction is the novel definition of declarative languages able to map an ontology into queries for a set of data sources This method is mainly designed for the case of integration of multiple heterogeneous relational database systems [5]. The authors introduce the concepts of Semantic Identifier and Semantic Join. First, represent a solution for the problem of entity resolution and the second one is designed to help in the problem of record linkage. Although, is an interesting approach, this have to be modified in order to be used for multiple heterogeneous data sources in the context of Big Data. Still, we cannot know for sure how accurate the mapping phase can be when dealing with these type of data sources.

In [6] the authors propose a spatial-temporal big-data storage system called "Pyro" tailored for high resolution geometry queries and dynamic hotspots. It understands geometries internally, permitting in this way to perform optimized aggregated geometry queries. The results show that Pyro reduces the response time by 60X on 1km1km rectangle geometries, when compared to other similar solutions.

Spatial data mining research area is concerned with the identification of interesting spatial patterns from data stored in spatial databases and geographic information systems (GIS). In [7] the authors present an analysis of a database with spatial and time stamped data of Slovenian traffic accidents. They used descriptive statistical methods, spatial clustering and vizualizations using geographical information systems facilities.

Teradata[1] propose a design for a data management system with three components storage engine, processing layer and functions library [8]. Storage engine keep relational data in databases and non-relational data as deserialized objects, similar to Blobs. Processing layer is an SQL engine extended with MapReduce functions. Functions library layer represents the core element, permitting users to write functions in order to manipulate and query data that are stored in a library and results in database tables.

Our approach, in order to overcome the data heterogeneity in Big Data platforms and to provide a unified and unique view of heterogeneous data is to add a layer on top of the different data management systems with aggregation and integration functions.

III. Cloud Storage Architecture for Water Management

Due to the continuous growth of cyber-infrastructures systems in scope and scale of the provided applications and data sources, lead to the concept of "data lake". From practical point of view it is characterized by three key attributes:

- **collect everything** - a data lake contains all data, both raw sources over extended periods of time as well as any processed data;
- **dive in anywhere** - a data lake enables users across multiple business units to refine, explore and enrich data on their terms;
- **flexible access** - a data lake enables multiple data access patterns across a shared infrastructure: batch, interactive, on-line, search, in-memory and other processing engines; as a result, a data lake delivers maximum scale and insight with the lowest possible friction and cost.

The following characteristics are important to take into consideration when design and analyze the storage system:

- **file size distribution** - is important for I/O optimization; the file size in workloads depends on the program styles of specific applications;
- **data commonality/data compressibility** - determine the data similarity of files;
- **data lifetime** - can help to choose an optimum storage device (e.g., workflows intermediate files and checkpoint images) are only temporary;
- **data open pattern** - represents the access modes for file by specifying parameters in file open call;
- **data request frequency** - the frequency of access to the file;
- **directory pattern** - directory structure of file;
- **data locality** - some applications exhibit high access locality, that is, the working sets of multiple application instances running on different nodes significantly overlap, while in other cases application instances running on different nodes have disjoint working sets;
- **data consistency requirements**.

These characteristics can be exploited through different optimization techniques to enhance storage system performance and to reduce the operation cost, with regard to different metrics such as latency, throughput, disk utilization, CPU load. For

[1]http://www.teradata.com/

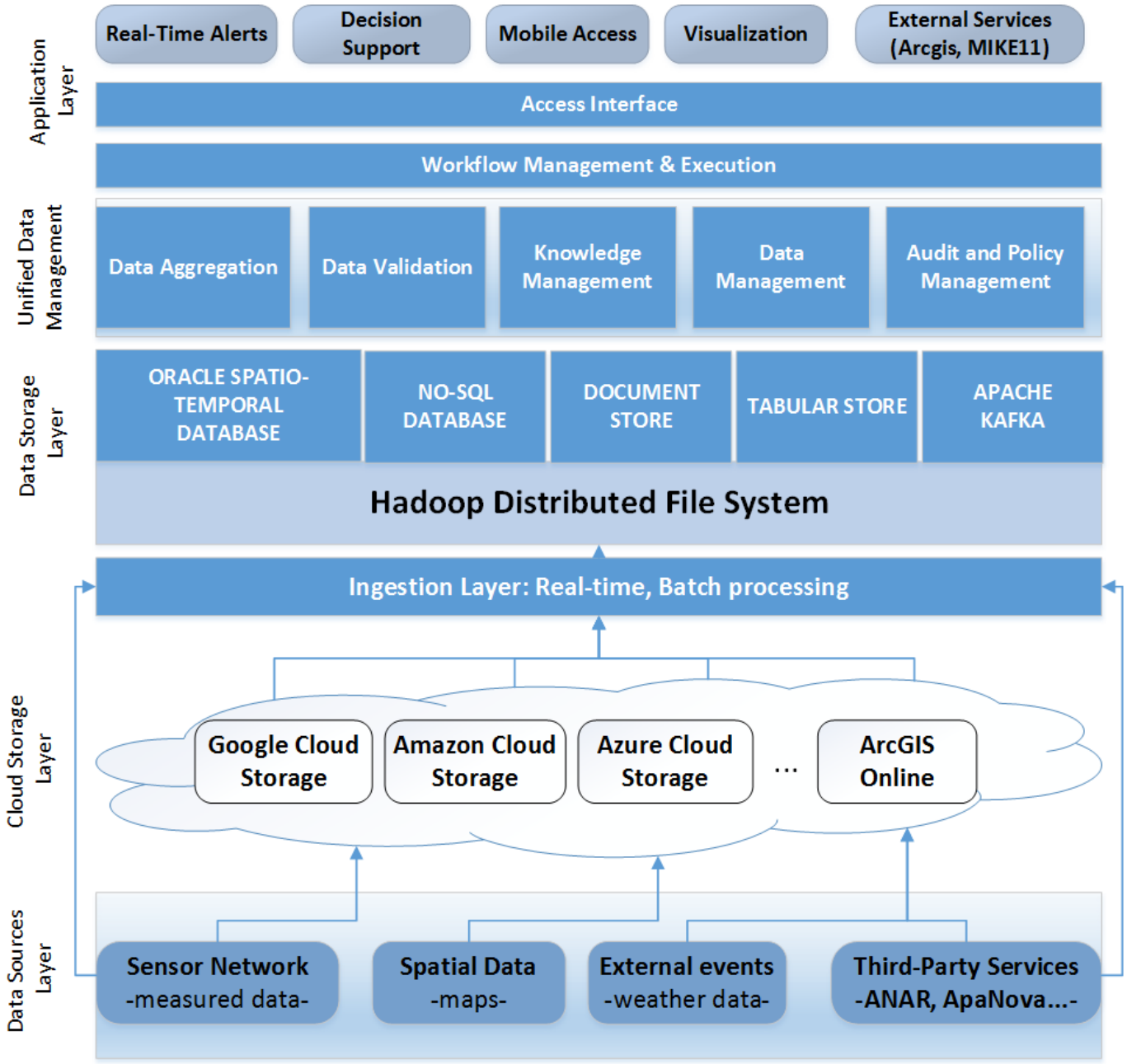

Fig. 1. CyberWater Cloud storage architecture

example, buffering can dramatically enhance throughput for write only workloads. Furthermore, deduplication technique can save considerable storage space for highly compressible workloads, but consume more when data need to be accessed.

It is important to note that different optimization techniques can have a negative impact on the operational cost. For example consistency mechanisms might have a negative impact on write throughput. Consequently, these optimizations do not generally coexist on the same data pipeline.

A. *Proposed architecture*

In Figure 1 is presented the architecture of the storage system for CyberWater project. Data sources layer consist of various heterogeneous data sources geographically distributed such as sensor network, spatial data, data suppliers (e.g. GIS, water treatments plants etc.), and third-party services data (e.g. ANAR, ApaNova, and other Romanian institutions). In order to process data first we store it at different Cloud storage providers, near to data sources. Second, depending on data needs these data are transfered for processing in a private datacenter. Moreover, we can buy also cloud processing

services if in a certain use case is more cost efficient to process data with a cloud service or if the processing capabilities of our private datacenter are exceeded.

Ingestion layer deals with the process of obtaining and importing data for storage or immediate use for the case of real-time alerts service. For the first use case data can be ingested in batches (e.g. discrete chunks at periodic intervals of time). In the second use case ingestion happens in real-time, data being ingested as it is emitted by the source. Data ingestion process supposes two phases data import and the data routing to the correct storage engine. For instance, if we deal with sensor data then we route them to he oracle spatial-temporal database.

Data storage layer consist of Hadoop Distributed File System. Hadoop provides HDFS and a framework for the analysis and transformation of very large data sets using the MapReduce paradigm. An important characteristic of Hadoop is the partitioning of data and computation across many (thousands) of hosts, and executing application computations in parallel close to their data. A Hadoop cluster scales computation and storage capacity and I/O bandwidth by simply adding commodity servers. Moreover, MapReduce and HDFS may run on the same set of nodes, which means that the compute nodes and the storage nodes are hosted on the same machines. The framework achieves higher bandwidth across the cluster by scheduling tasks on the same nodes where data is stored. The Map tasks process input records and write to the local disk a set of intermediate records.

HDFS stores file system metadata and application data separately. As in other distributed file systems, like PVFS, Lustre and GFS, HDFS stores metadata on a dedicated server, called the NameNode. Application data are stored on other servers called DataNodes. All servers are fully connected and communicate with each other using TCP-based protocols. Unlike Lustre and PVFS, the DataNodes in HDFS do not use data protection mechanisms such as RAID to make the data durable. Instead, like GFS, the file content is replicated on multiple DataNodes for reliability. While ensuring data durability, this strategy has the added advantage that data transfer bandwidth is multiplied, and there are more opportunities for locating computation near the needed data.

On top of HDFS we put different storage engines such as Oracle spatial-temporal database for sensor and GIS data, NoSql database for key-value data, document and tabular store for semi-structured data. Relational databases can handle various types of data for example sensor data or GIS data. Every query has the same kind of data location, water parameters, map and so on. These are all stored in a table with one column for each piece of data. On the other side we have multimedia files such as images or videos that can be attached to an event reporting action. Multimedia files cannot be represented in the same way as a series of columns. What can be stored in a relational way is represented by the data about the files, which is in fact metadata. Alongside with multimedia files we have social media objects such as blogs, tweets, and emails, which can be categorized in the category of non-relational data.

B. Cloud storage systems performance evaluation

In order to test the performance of different cloud storage providers we upload and download data collected form sensors. Each test was performed 8 times and took the average value.

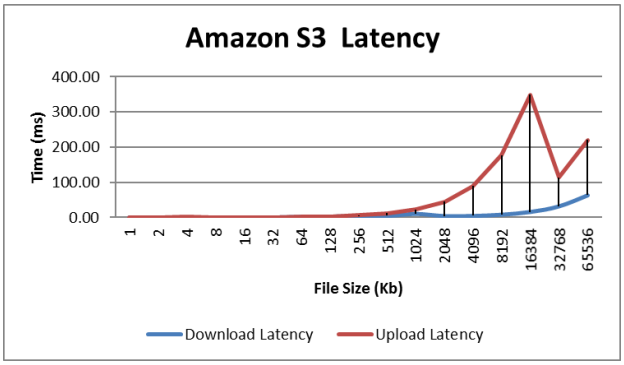

Fig. 2. Amazon S3 Latency

Figure 2 presents average upload/download time (ms) for different file size (Kb) for Amazon S3. It can be seen that after 512 KB, the upload time grows proportional to the size. There seem to be little deviations from this rule. The downloading speed grows slowly for files us to 1024KB and after that it approximately doubles for each file increase. It can be observed that upload speeds are approximately five times slower that download speeds.

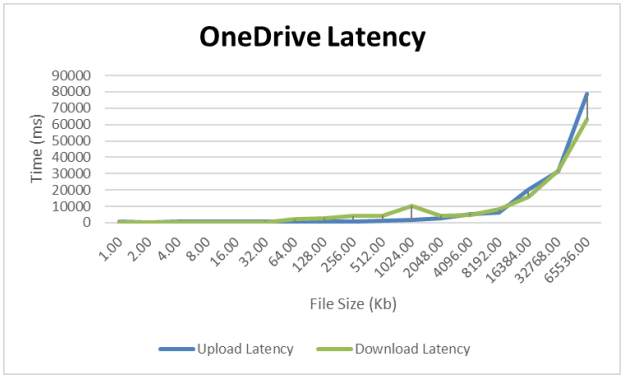

Fig. 3. OneDrive Latency

Figure 3 presents average upload/download Time (ms) for different file size (Kb) for OneDrive. Upload and download times are aproximately equal with a slight difference for 1024Kb file sizes. Values less than 8Kb may be excluded from the results, which were inconclusive

Figure 4 shows the uploading and downloading latency for files of different sizes, where we can observe that copying files from local to Hadoop file system (HDFS) is much faster than vice-versa.

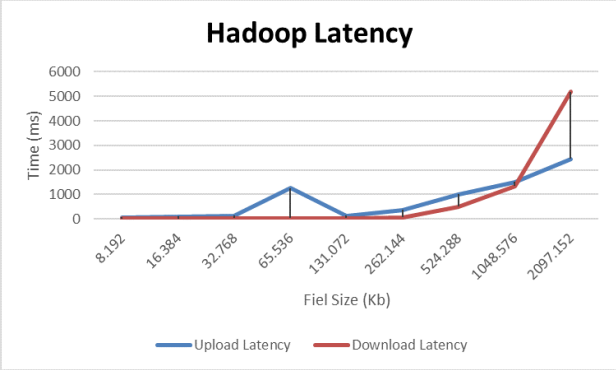

Fig. 4. Hadoop Latency

IV. Conclusion

In the case of geographically distributed applications Cloud Computing model may not always be the best choice. In order to improve offered service, the processing and data storage needs to be performed closer to the data source. Real-time applications have small response time in the order of milliseconds (25ms to 50ms). Processing in cloud will introduce a significant latency as we have seen between device and cloud infrastructure. Some alternatives are edge nodes with computation and storage capabilities (e.g. set-top-box machines, routers, etc.) which are one-hop away from data source and can be used to reduce network latency.

Acknowledgment

The research presented in this paper is supported by projects: ; Data4Water - H2020-TWINN-2015, CSA-690900; *DataWay*: Real-time Data Processing Platform for Smart Cities: Making sense of Big Data - PN-II-RU-TE-2014-4-2731.

We would like to thank the reviewers for their time and expertise, constructive comments and valuable insight.

References

[1] T. Nam and T. A. Pardo, "Conceptualizing smart city with dimensions of technology, people, and institutions," in *Proceedings of the 12th annual international digital government research conference: digital government innovation in challenging times*. ACM, 2011, pp. 282–291.

[2] C. Negru, F. Pop, M. Mocanu, and V. Cristea, "A unified approach to data modeling and management in big data era," in *Data Science and Big Data Computing*. Springer, 2016, pp. 95–116.

[3] L. Relly, A. Kuckelberg, and H.-J. Schek, "A framework ofa generic index for spatio-temporal data in concert," in *Spatio-Temporal Database Management*. Springer, 1999, pp. 135–151.

[4] A. Abouzeid, K. Bajda-Pawlikowski, D. Abadi, A. Silberschatz, and A. Rasin, "Hadoopdb: an architectural hybrid of mapreduce and dbms technologies for analytical workloads," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 922–933, 2009.

[5] M. Leida, A. Gusmini, and J. Davies, "Semantics-aware data integration for heterogeneous data sources," *Journal of Ambient Intelligence and Humanized Computing*, vol. 4, no. 4, pp. 471–491, 2013.

[6] S. Li, S. Hu, R. K. Ganti, M. Srivatsa, and T. F. Abdelzaher, "Pyro: A spatial-temporal big-data storage system." in *USENIX Annual Technical Conference*, 2015, pp. 97–109.

[7] N. Lavrac, D. Jesenovec, N. Trdin, and N. M. Kosta, "Mining spatio-temporal data of traffic accidents and spatial pattern visualization," *Metodoloski zvezki*, vol. 5, no. 1, p. 45, 2008.

[8] M. Whitehorn, "Aster data founders explain unified approach to data big and small," 2015. [Online]. Available: http://www.computerweekly.com/feature/Aster-Data-founders-explain-unified-approach-to-data-big-and-small