Cloud Based Sensor Big Data Management: A Literature Review

Anthony J. Christe

January 23, 2017

Abstract

1 Introduction (abstract?)

The exponential increase in volume, variety, velocity, veracity, and value of data has caused us to rethink traditional server architectures when it comes to data acquisition, storage, analysis, quality of data, and governance of data. With the emergence of Internet of Things (IoT) and increasing numbers of ubiquitous mobile sensors such as mobile phones, distributed sensor networks are growing at an unprecedented pace and producing an unprecedented amount of streaming data. It's predicted by the European Comission that IoT devices will number between 50 to 100 billion devices by 2020[13].

The size of sensor networks is quickly growing. BBC Research provides figures that the market share for sensor networks in 2010 was $56 billion and was predicted to be closer to $91 billion by the end of 2016 [20]. Data generated from the IoT are surpassing the compute and memory resources of existing IT infrastructures. [3]. Not only is the size of data rapidly exploding, but data is also becoming more complex. Data from sensor networks is often semi-structured on unstructured with data quality issues.

Sensor networks can benefit from the generally "unlimited resources" of the cloud, namely processing, storage, and network resources. We believe that by leveraging cloud computing, distributed persistence models, and distributed anayltics, its possible to provide a platform that is able to meet the demands of the increasing distributed sensor market and the increasing volume, velocity, variety, and value of data that comes along with that.

This review hopes to summarize the current state of the art surrounding distributed sensor networks and the use of cloud computing as a means for big sensor data acquisition and analysis. We will also briefly discuss privacy concerns relating to sensor data in the cloud.

The rest of this review is as follows: The rest of chapter 1 will review the concepts cloud computing, big data, and sensor networks with motivation as to why these technologies are intertwined. Chapter 2 will compare and contrast various Big Data persistence models. Chapter 3 will provide a review of the current state of the art analytics for Big Data. Chapter 4 will discuss privacy in relation to sensor networks and the cloud. Chapter 5 compares reference implementations of sensor frameworks in the cloud. Chapter 6 discusses future directions and open research questions.

1.1 Big Sensor Data

Sensor data is generally different from Big Data in several ways [3]. One of the most common characteristics of sensor data is the amount of noise in the data. Environmental sensing will always include noise because the data is born analog[10]. Often sensor networks provide data in a large variety of unstructured formats with missing, partial, or conflicting meta-data. Sensor data can also contain a large amount of data redundency from multiple sensors in similar locations. The problem very quickly becomes a needle-in-the-haystack problem or more aptly put, finding the signal in the noise.

1.2 Cloud Computing

NIST[9] defines cloud computing as "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction".

The major five tennants of cloud computing as defined by NIST are as follows:

On-demand self-service where the user can provision network, storage, and compute capactity automatically without the need for human intervention. In essense, this becomes a virtual shopping mart where to the comsumer it appears that virtually unlimeted cloud resources are avaiable to chose from and the user (or algorithm) can increase or decrease the utilization of cloud resources at any time.

Broad network acess where computation capabilities are performed over a network and results are delivered to clients such as mobile devices.

Resource pooling where resources within a cloud such as storage, network, or compute capacity are shared among multiple tenants. This allows for efficient utilization of hardware when generally virtual services are provided to clients. Clients don't nesisarily know where their physical hardware is located or provisioned.

Rapid elasticty is the ability to provision or remove cloud resources (i.e. storage, network, or compute resources) at any time from a system as demand on that system either increases or shrinks. Often times a human may not even be involed in making these descisions and this scaling will take place autimatically using a set of predefined usage thresholds.

Measure service where cloud providers provide a means of metering the compute resources that are used by clients. This provides a transparent means of selling cloud computing resources to clients and clients can always know how much capacity they have consumed.

Even though the NIST definition is starting to show its age, it's major tenants are still the underlying foundation of cloud software even today. Many additional service and deployment models have been developed since NIST defined cloud computing, but an understanding of the basic underpinnings is required before exploring the rest of this vast filed.

Cloud computing frameworks can provide on-demand availability and scaling of virtual computing resources for storage, processing, and analyzing of very large data sets in real-time or near real-time. This model makes it possible to build applications in the cloud for dealing with Big Data sets such as those produced from large distributed sensor networks.

By using the cloud as a central sink of data for our devices within a sensor network, it's possible to take advantage of central repositories of information, localized dynamic computing resources, and parallel computations. With the advent of cheap and ubiquitous network connections, it's becoming easier to do less processing within sensor networks and to offload the work to a distributed set of servers and processes in the cloud[6].

Cloud computing includes both technical and economical advantages as discussed in [1].

On the economical side, computing resources are pay-per-use. Businesses can dynamically increase or decrease the computing resources they are currently leasing. This makes it possible to utilize massive amounts of computing power for short amounts of time and then scale back resources when demand isn't at its peak. Before cloud computing these same businesses would be required to manage and maintain their own hardware for peak load without the ability to dynamically scale their hardware if the peak load were to increase.

On the technical side, the localization of computing resources provides for a wide variety of benefits including energy efficiency, hardware optimizations, software optimizations, and performance isolation.

1.2.1 Cloud Computing Service Models

When discussing cloud computing, it's useful to understand the service models that traditional cloud computing provide. The three major service models as defined by NIST[9] are **Infrastructure as a Service** (IaaS), **Platform as a Service** (PaaS) and **Software as a Service** (SaaS).

At the lowest level is the Infrastructure as a Service

(IaaS) model which provides virtual machines that users have the ability to deploy and manage. Users can install operating systems on these virtual machines and interact with deployed virtual machines as if they were local servers. Consumers using IaaS have the ability to manage and provision virtual hardware and network resources, but do not need to worry about the underlying hardware or network infrastructures. Other than providing virtual resources, consuming utilizing IaaS still require a decent amount of systems administration knowledge develop, deploy, and secure applications into the cloud using IaaS.

Sitting in the middle of the traditional cloud service models is the Platform as a Service (PaaS) model. In this service model consumers don't have the ability to interact or provision individual cloud resources such as virtual machines, storage, networking, or compute capacity. Instead, users have the ability to deploy their application to the cloud via custom cloud provides tools or via a cloud provided application programming interfaces (APIs).

At the highest level is the Software as a Service (SaaS) layer. Generally speaking, applications in a SaaS environment are generally provided by the cloud provider. In a SaaS model, users do not have the ability to control their own cloud resources and users do not have the ability to upload their own applications to the cloud. Users do sometimes have the ability to alter the configuration of the software they are interacting with in this model.

This review aims to look at software and architecture somewhere between and including the PaaS and SaaS layers. There are many examples in the literature at optimizing data centers for cloud architecture and the IaaS service level (i.e. [7], [18], [15], [11], [8]). These discussions are outside of the scope of this review as we want to focus on software framework for sensor data collection in the cloud.

1.2.2 Cloud Deployment Models

NIST[9] provides four types of deployment models in its cloud computing definition: **private cloud**, **community cloud**, **public cloud**, and **hybrid cloud**.

A private cloud is is a cloud where resources are provisioned to a single organization (or multiple parties within a single organization). In this model the organization may deploy their own cloud hardware or use cloud resources provided by a third party.

In a community cloud, resources are provided to a group of organizations that have similar requirements and may be owned and managed by a single organization, multiple organizations, or third parties.

Public clouds provide computing resources to anyone willing to purchase said cloud resources. Public clouds can be owned and manged by anyone, a government, or any third-party. Generally all hardware in public clouds are managed by the cloud provider.

Hybrid clouds use and provide a combination of the previously mentioned deployment models and can be configured, split-up, and manged in many ways.

Virtual private clouds as described in Botta et al.[1] provide aspects from both public and private clouds using virtual private network (VPN) technology to allow users to manage specfific and often times complicated network technologies.

1.2.3 Issues with Cloud Computing

Being a relatively new research field cloud computing is showing quite a few growing pains.

The biggest issues facing cloud computing deal with security and privacy. Subashini et al.[16] go into the specific security and privacy risks associated with the three major cloud deployment models.

The deployment model that exibits the most risks in the PaaS layer since this model requires that consumers manage their own virtual machines, deployment of cloud applications, and configuration. Within this model, the following items are of conern.

Data security is a conern on all deployment models, but especially at the PaaS layer where sensitive data will be stored on remote servers not owned by a client. Since clients manage their own servers in the PaaS layer, they are also required to manage their own data security. Data security issues include cross-site scripting, access control weaknesses, injection attacks, cross-site request forgery, cookie manipulation, hidden field manipulation, insecure storage, and insecure configuration.

Network security becomes an issue when sensitive data is transfered from clients to the cloud backend

and visa-versa. If encryption is not used, users could be vulnerable to port scans, ip spoofing, man-in-the-middle attacks, packet sniffing, and more. Even if encryption is used, users can still be vulnerable to packet analysis, insecure SSL configurations, session management weaknesses.

At some level, comsumers are required to trust that the cloud provider they choose will implement security and privacy best practices within their cloud architecture. This does now however resolve the larger issues of security vulnerablities within cloud software and their communication componenents.

1.3 Mobile Cloud Computing

Mobile devices such as smartphones make fantastic distributed sensors for temporalspatial data. Not only do they carry a wide array of sensors on-board (microphones, barometers, accelerometers, GPS, compasses, cameras, clocks), but they generally have multiple modes of offloading data (WiFi, bluetooth, cellular, SD cards), and support some preprocessing on-board.

1.4 Scientific Data Infrastructure

1.5 Applications of Distributed Sensor Networks

Zaslaveky et al. [20] cites several examples of distributed sensor networks in-the-wild including: a real-time greenhouse gas detection network deployed across California, real-time structural monitoring such as the St. Anthony Falls Bridge sensor network in Minneapolis, distributed radiation detection in Fukushima, real-time parking space inventory in San Francisco

Software for cloud environments include distributed fault-tolerant databases and distributed parallel algorithms for computer clusters.

One area that shows a lot of promise for distributed sensor networks with centralized management is smart grids. The smart grid is an collection of technologies aiming to advance the electrical grid into the future with respect to intelligent energy distribution and integration of renewables. Electrical grids can benefit by using a large distributed sensor network to collect power consumption, production, and quality information and use that information to control power production and consumption in real-time.

Many times, the sensor nodes in smart grids lack powerful local computation abilities, but generally have network connections and sensing capabilities. This makes the cloud a perfect sink of information for analyzing complex power trends from a large scale distributed sensor network for smart grids[1].

2 Big Data Persistence Models

Traditional storage methods for meta-data and related products has traditionally made use of the filesystem and relational database systems (RDMS).

Big Sensor Data by its nature can be structured, unstructured, large, diverse, noisy, etc. Many of the properties of BSD do not fit nicely into the structured world of traditional RDMSs.

In-order to meet the needs of Big Sensor Data and distributed sensor networks, we look to the ever growing field of NoSQL (not only SQL) and related Big Data storage models. There are multiple types of data models with different use cases. We will review the current players in this field and with a focus on how these technologies could aid in managing Big Sensor Data in the cloud.

According to Song et al.[5] an ideal NoSQL data model strives for "high concurrency, low latency, efficient storage, high scalability, high availability, reduced management and operation costs." The challenges of realizing an ideal NoSQL data model however lie in three main areas[3]: consistency, availability, and partition tolerance.

Several of the persistence models we review do not support ACID (Atomicty, Consistency, Isolation, Durability). A consequence of this is less than perfect consistency. Consistency issues occur when data is stored in a distributed manner with multiple copies. In situations of server failure (or with systems that support different consistency models), situations can arise where multiple copies of the same resource contain different contents.

Vogels and Wener[17] explain the main forms of consistency. Assume a record is being updated across multiple servers. With "strong consistency", any access of that resource after the update will return the updated result. With "weak consistency", subsequent access of that resource is not guaranteed to return the updated result if that access is within a certain "inconsistency window". With eventual consistency, the only guarantee you get is that access to the resource will be show up "eventually" where eventually can depend on many factors.

As the amount of hardware (servers, switches, *etc*) increases in a distributed system so does the amount of hardware errors. Availability refers to the ability to remain operational even as parts of a distributed system drop in and drop out[3]. Gilbert[4] defines availability as "every request received by a non-failing node in the system must result in a response." He goes on further to point out that this definition does allow for unbounded computation since it's possible to wait for a result that never returns.

As the amount of hardware increases in a distributed system, the number of communication packets that drops also increases. The ability to maintain service in the face of some amount of drops refers to partition tolerance[3].

The above ideas are all tied together into the CAP theorem proposed by Brewer[2] which states that in any shared data system, you can only achieve two of the three following properties: Consistency, Availability, or Partition (tolerance). As we review different Big Data architectures, we will examine how they fit into the CAP theorem and what guarantees they provide for these three major areas of distributed data management.

We also believe, ease of use, maturity of the product, and community (or commercial) support should also factor into the comparisons between data models.

With the above factors in mind, we can begin categorizing and analyzing several major Big Data model solutions.

2.1 Data Models

2.1.1 Key-Value

The simplest data model for distributed storage is likely the Key-Value (KV) data model [19]. In this model, every piece of data stored is indexed by a unique primary key. Queries to that item all happen via its key to access the value. Values in a KV system can be treated as blobs and the content of the value is irrelevant to the semantics of KV stores. KV systems are popular due to their simplicity and ease of scaling.

Keys in KV systems are the unit parallelism that provide the main means of concurrency. If you want to guarentee transactions, then keys can be naively sharded across servers. This does now however provide safety of data loss in which case a system will strive to provide replication at the cost if ACID compliance. Stores and requests can usually be achieved in $O(1)$ even in distributed systems[12].

If the major advantages are simplicity and query response time[3], the major disadvantage to KV stores is the fact that they lack advanced query capabilities. The only way to query a database is by its unique key. Range based queries, secondary, and tertiary indexes are only supported by a third party systems.

Popular KV based solutions include Dynamo, Riak, Voldemort, Redis, MemcacheDB, Ignite and others.

Key-Value stores can be useful tools for simple distributed sensor networks that keep track of a handful of non-complex data such as current temperature, current voltage, etc. Issues arise when you want to start recording timestamped data. If timestamped data arrives at irregular intervals, KV stores to not provide range based queries to query on timestamps. Thus, KV stores may be poorly suited for time based sensor data. Finally, KV stores do not provide queries for location based sensor data making it potentially a poor choice for geo-spatial data.

2.1.2 Document

Document based data models allow data to be stored in structured documents including KV pairs. The underlying document structure can be anything as long

as its structure is something that the store can understand. Common document formats include XML, JSON, YAML, BSON, RavenDB, and others[3]. Documents can also store other documents recursively.

Even though documents are restricted to their underlying structure (i.e. JSON, YAML, etc), document databases do not impose a schema like traditional RDMS databases. This allows for painless transitions of data storage when the underlying data change[14]. This is of special concern with heterogenous sensor networks which can produce large varieties of data in different and changing formats.

Document based data stores often present less of an impendance mismatch between data structures in a programming language and their underlying representation in the data store. Compared to the way data is structured in a RDMS, it's often easier to understand the underlying data structure in a documented based store. There are many common libraries for converting between documents in a document store and a data structure in a particular programming language. Compared to RDBMS, fields in document stores generally do not normalize their data which also enhances the readability of the underlying data structure[12].

An advantage that document based stores have over KV stores is that its possible to create more complex queries. Not only can you query on the primary key, but its possible to create queries over any keys in the document including in sub-documents. Indexes can be created on key and sub-keys. Many document based stores provide range and geospatial based queries. This advantage alone makes document based stores a decent choice for distributed sensor data.

Common document based stores include MongoDB, CouchDB, OrientDB, RavenDB, SimpleDB,

2.1.3 Graph

2.1.4 Wide Column Store

2.2 Data Guarentees

2.2.1 title

2.3 Indexing

3 Big Data Analytics

4 Privacy

5 Reference Implementations

6 Open Research Question

References

[1] Alessio Botta, Walter de Donato, Valerio Persico, and Antonio Pescap. Integration of cloud computing and internet of things: A survey. 56:684–700.

[2] Eric A Brewer. Towards robust distributed systems. In *PODC*, volume 7, 2000.

[3] Min Chen, Shiwen Mao, and Yunhao Liu. Big data: A survey. 19(2):171–209.

[4] Seth Gilbert and Nancy Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *ACM SIGACT News*, 33(2):51–59, 2002.

[5] A. Jin, C. Cheng, F. Ren, and S. Song. An index model of global subdivision in cloud computing environment. In *2011 19th International Conference on Geoinformatics*, pages 1–5, June 2011.

[6] Supun Kamburugamuve, Leif Christiansen, and Geoffrey Fox. A framework for real time processing of sensor data in the cloud. 2015:1–11.

[7] Niloofar Khanghahi, Ramin Nasiri, and Mahsa Razavi Davoudi. A new approach towards integrated cloud computing architecture. 4(1):24–34.

[8] Praveenkumar Khethavath, Johnson P. Thomas, and Eric Chan-tin. Towards an efficient distributed cloud computing architecture.

[9] Peter Mell and Tim Grance. The nist definition of cloud computing. 2011.

[10] President's Council of Advisors on Science and author Technology (U.S.). *Report to the President, big data and privacy : a technology perspective.* Washington, District of Columbia : Executive Office of the President, President's Council of Advisors on Science and Technology, 2014. Includes bibliographical references.

[11] Han Qi, Muhammad Shiraz, Abdullah Gani, Md Whaiduzzaman, and Suleman Khan. Sierpinski triangle based data center architecture in cloud computing. 69(2):887–907.

[12] A. Rahien and O. Eini. *RavenDB Mythology Documentation.*

[13] D. Reed, J. R. Larus, and D. Gannon. Imagining the future: Thoughts on computing. *Computer*, 45(1):25–30, Jan 2012.

[14] Sugam Sharma. An Extended Classification and Comparison of NoSQL Big Data Models. *arXiv preprint arXiv:1509.08035*, 2015.

[15] Zhiming Shen, Sethuraman Subbiah, Xiaohui Gu, and John Wilkes. Cloudscale: elastic resource scaling for multi-tenant cloud systems. In *Proceedings of the 2nd ACM Symposium on Cloud Computing*, page 5. ACM.

[16] S. Subashini and V. Kavitha. A survey on security issues in service delivery models of cloud computing. *Journal of Network and Computer Applications*, 34(1):1–11, January 2011.

[17] Werner Vogels. Eventually consistent. *Queue*, 6(6):14–19, 2008.

[18] Bin Wang, Zhengwei Qi, Ruhui Ma, Haibing Guan, and Athanasios V. Vasilakos. A survey on data center networking for cloud computing. 91:528–547.

[19] Silvan Weber. Nosql databases. *University of Applied Sciences HTW Chur, Switzerland*, 2010.

[20] Arkady Zaslavsky, Charith Perera, and Dimitrios Georgakopoulos. Sensing as a service and big data.