Data Management for Distributed Sensor Networks: A Literature Review

Anthony J. Christe

January 30, 2017

Abstract

Sensor networks can benefit from the generally "unlimited resources" of the cloud, namely processing, storage, and network resources. This literature review surveys the major components of distributed data management, namely, cloud computing, distributed persistence models, and distributed analytics.

Contents

List of Figures

1 Introduction

The exponential increase in volume, variety, velocity, veracity, and value of data has caused us to rethink traditional client-server architectures with respect 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 Commission that IoT devices will number between 50 to 100 billion devices by 2020[25].

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 [30]. Data generated from the IoT are surpassing the compute and memory resources of existing IT infrastructures. [8]. 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 analytics, it's now 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 summarizes 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. In particular, we will define Big Data and review it in the context of sensor networks, review cloud computing and service models related to distributed sensing, discuss modern distributed persistence for Big Data, and modern distributed analytics for Big Data all with an emphasis on acquiring and managing Big Sensor Data.

1.1 Applications of Distributed Sensor Networks

Zaslaveky et al. [30] 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.

Perera et al. in their paper on sensing as a service[22] provide three examples of areas distributed sensor networks would accelerate at.

First, distributed sensors could be used by cities to optimize waste management which consumes a significant amount of time, money, and labor. Waste management also has many processes including collection, transport, processing, disposal, and monitoring. By collecting and storing sensor data in the cloud from these processes, various interested parties could access sensor data in

order to optimize for the current state of the system. As an example, Perera mentions that city council members could optimize garbage routes and collection rates based on the amount of trash available and recycling centers could forecast what to expect based off of the same sensor data. Basically interested parties at all points of the management process could benefit by analyzing data points from IoT devices in a smart city.

Second, Perera mentions that smart agriculture can take advantage of distributed sensor networks and cites the *Phenonet* project as an example of distributed agricultural sensing which has the ability to monitor plant growth, soil composition, air composition, and pests. A major advantage of this system is that it can supplement traditional research by allowing multiple researchers access to the same data in near real-time.

Third, Perera postulates that environmental management could utilize existing distributed environmental sensors upgraded to communicate with the cloud allowing for data sharing and data fusion among interested parties.

Gerla et al.[12] propose an internet of vehicles as a means to autonomous vehicles. By treating vehicles as platforms of thousands of sensors each and by creating dynamic distributed clouds, they hope to allow fleets of vehicles to make autonomous decisions. This model uses distributed clouds based on proximity and peer-to-peer to technologies rather than sending data to a centralized cloud. The real-time nature and the size and amount of sensors makes this an interesting case study.

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 renewable. 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.

In some cases, 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[6].

1.2 Rest of this Review

The rest of this review is structured as follows: Section 2 provides an overview of Big Sensor Data2. Section 3 will focus on cloud computing and how its concepts can be utilized to manage distributed sensor data. Section 4 will examine the current state of the art distributed persistence models with an emphasis on how NoSQL and distributed persistence models can aid in managing distributed sensor data. Section 5 will examine the current state of big data analytics options in the cloud and how these can be utilized for performing analytics on distributed sensor data.

2 Big Data

Big Data is described using many definitions. Cox, in 1997[9], provides us with one of the earliest definitions where Big Data is “too large to be processed by standard algorithms and software on the hardware one has available to them”. He also mentions that sources for big data collections include data from remote sensors and satellite imaging in the fields of atmospheric sciences, geophysics, and healthcare.

Cox separates Big Data into big data into two categories; namely, *big data collections* and *big data objects*.

Big data objects are single, very large data sets such as computational models computed from physical phenomena. Big data objects often do not fit in memory or local disks. Big data objects also have adverse affects on bandwidth and latency. Cox looks to moving computation to the data and more advanced segmentation and paging techniques at the OS level to deal with big data objects.

Big data collections contain many smaller objects or even many big objects. Big data collections present their own set of issues including: distributed data, heterogeneous data formats, no platform independent definition, non-local meta-data, large storage requirements, poor locality, and insufficient network resources.

Cox provides us a useful definition to build on. He also advocates for the development and advancement of operating system constructs for moving data that is too large for memory in and out of memory using stenciling, segmentation, paging, and application controlled segmentation. It’s interesting to note that this was before cloud computing and distributed systems, but we are now facing similar problems at the distributed level rather than a local level.

The Apache Hadoop project, in 2010, defined big data as “datasets which could not be captured, managed, and processed by general computers within an acceptable scope”[8].

Manyika et al[18] in 2011 define big data as “the amount of data just beyond technology’s capability to store, manage, and process efficiently” essentially making the definition of big data a moving target that is constantly evolving as technology becomes updated.

Hashem et al.[14] build on these previous definition in their 2015 review on Big Data providing the definition by attempting to create a definition that encompasses the spirit of many of the previous definitions. They define big data as “a set of techniques and technologies that require new forms of integration to uncover large hidden values from large datasets that are diverse, complex, and of a massive scale”.

NIST, in 2015, [20] provide multiple definitions relating to big data. NIST defines big data as “extensive datasets–primarily in the characteristics of volume, variety, velocuty, and/or variability–that require a scalable architecture for efficient storage, manipulation, and analyst. To my knowledge, NIST is the only organization to specify the need of a scalable architecture alongside its defini-

tion of big data. NIST next defines the big data paradigm as "the distribution of data systems across horizontally coupled, independent resources to achieve the scalability needed for the efficient processing of extensive datasets".

Perhaps one of the most popular definitions of Big Data is characterizing data by "the four Vs"[14], *volume*, *variety*, *velocity*, and more recently, *value*.

2.1 The Four V's

The first mention of the three V's was in Laney's 2001 article *3-d data management: controlling data volume, velocity, and variety*[17]. Laney describes the challenges of managing e-commerce data by categorizing the data challenges into three dimensions. First, we will review Laney's definition of the 3 V's and then we will examine updated, expanded, and more modern interpretations of the three V's and also look at the more recent "fourth V".

Volume is the amount of data flowing into a system at any one time. Laney argues that the increased availability of the internet to anyone as an e-commerce platform greatly increases the amount of transactional data stored on server backends. Since data is a tangible asset, organizations may be reluctant to discard the data. At the same time, as the amount of data increases, each individual data point becomes less important. Laney mentions that if organizations are not willing to simply buy more online storage, that they can take the following steps to limit volume growth: implement tiered storage systems, limit data collected to only data required for current organization processes, limit analytics to statistically sampled data, eliminate redundancy in data sources, offload "cold spots" to cheaper storage (i.e. tape), and outsorce data management.

Velocity is defined by Laney as "increased point-of-interaction (POI) speed and, consequently, the pace data used to support interactions and generated by interactions", or more generally, the pace of data arriving and how long it takes to analyze, store, and act on that data. Laney offers several solutions to data velocity in using operational data stored that prioritizes production data, front-end caching, point-to-point data routing protocols, and architecting software in such a way that balances data analysis latency with stated real-time requirements.

Laney describes data variety as data that is in "incompatible formats, non-aligned data structures, and inconsistent data semantics". Laney's proposed solutions to variety include profiling data to find inconsistencies, a standardized XML data format, interprocess application communication, middlewares on top of "dumb data" to provide meaning and intelligent, metadata management, and more advanced indexing techniques.

The definition of the 4 V's have been updated over the years in the following ways. The volume is data is expanding faster than ever before. According to IBM[2], "90% of the world's data has been created in the past two years" and the current data volume is 2.7 zettabytes and is expected to double every two years[1].

2.2 Features of Big Data

NIST[20] provides us with a list common features of big data. One common feature of big data is associated metadata. Metadata is data about data that includes information about how/when/where the data was collected and processed. Metadata describing the history of data provides for data provenance. This becomes more important as data is transfered between many processes with multiple transformations. Provenance provides a means to track how data was transferred and how it was transformed. Semantic metadata is an attempt at providing metadata with the ability of describing itself. Examples of semantic metadata include the Semantic Web[5] and NIST's Big Data Paradigm.

Another common feature of big data is that is can often be unstructured, semi-structured, or non-relational data.

2.3 Examples of Big Data

2.4 Sensor Data

Sensor data is generally different from Big Data in several ways [8]. 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[21]. 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 redundancy 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.

Perera et al.[22] expect with IoT we could see as many as 1 billion sensors online by the year 2020.

TODO: this section needs to be expanded, but the definition of cloud computing has been done over and over again to death. This will continue to focus on the challenges associated with data specifically from distributed sensor networks.

3 Cloud Computing

NIST[19] defines cloud computing as "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 tenants of cloud computing as defined by NIST are as follows:

On-demand self-service where the user can provision network, storage, and compute capacity automatically without the need for human intervention. In essence, this becomes a virtual shopping mart where to the consumer it

appears that virtually unlimited cloud resources are available to chose from and the user (or algorithm) can increase or decrease the utilization of cloud resources at any time.

Broad network access 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 necessarily know where their physical hardware is located or provisioned.

Rapid elasticity 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 involved in making these decisions and this scaling will take place automatically 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, its 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[16].

Cloud computing includes both technical and economical advantages as discussed in [6].

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.

3.1 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[19] 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.

Since the original service models were penned, there have been many other types services models introduced. Several of these focus on IoT service layers as noted by Botta et al's[6]. These include *Sensing as a Service* (S^2aaS), *Sensing and Actuation as a Service* (SAaaS), *Sensor Event as a Service* (SEaaS), *Sensor as a Service* (SenaaS), *Data Base as a Service* (DBaaS), *Data as a Service* (DaaS), *Ethernet as a Service* (EaaS), *Idendity and Policy Management as a Service* (IPMaaS), and *Video Surveillance as a Service* (VSaaS).

Some of the above mentioned service models are of particular interest for a survey examining cloud computing and sensor networks. We will examine these in more detail in sections 3.2.

3.2 Sensing as a Service

Sensing as a Service (SaaS or S^2aaS) describes "the process of making the sensor data and event of interests available to the clients respectively over the cloud infrastructure"[10].

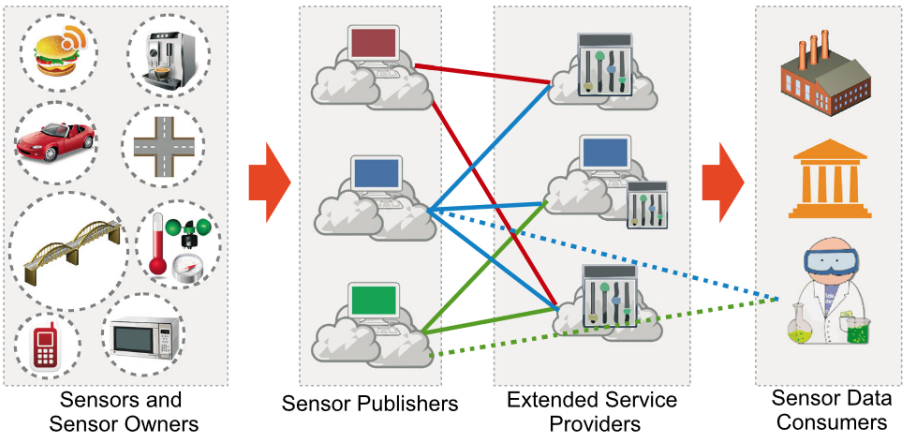

Figure 1: Sensing as a service layers.[22]

The sensing as a service model includes 4 layers[22]. Figure 1 shows these 4 layers in more detail.

The *sensor and sensor owners* layer includes physical sensors which can sense an increasingly broad variety of natural phenomena and sensor owners which can be personal, household, private, public, or commercial. Sensor owners have the final say in what data gets to the cloud and who can access the data once it is in the cloud using conditions and restrictions.

The *sensor publishers* (SP) layer manages the detection of online sensors and acts as a middle-man between sensor consumers and sensors and sensor owners. Sensors register with the publisher layer. Sensor data consumers make requests to sensor publishers for specified types of data over specified amounts of time.

The *extended service providers* (ESP) layer builds abstraction on top of sensor publishers. A single ESP can interact with mulitple SPs. ESPs can be used to automatically request data from multiple sensors depending on criteria provided to the ESP. This can be useful if the sensor consumer does not care about the underlying individual sensors but instead queries data at a higher level (i.e. all temperature data within a given polygon).

Finally, the *sensor data consumers* layer consist of data consumers who must register with the ESPs and provide valid digital certificates. Consumers can either deal with SPs directly or deal with ESPs. The benefit to dealing with SPs is reduced cost of communications with the ESP. The benefit of dealing with ESPs is higher level querying to data and the ability to query data across multiple SPs.

I find sensing as a service appealing, but lacking in actual implementation details. To Perera's credit, he does mention quite a few open technological challenges that need filled including architectural designs, sensor configuration, sensor management, data fusion, filtering, processing, storage, and energy

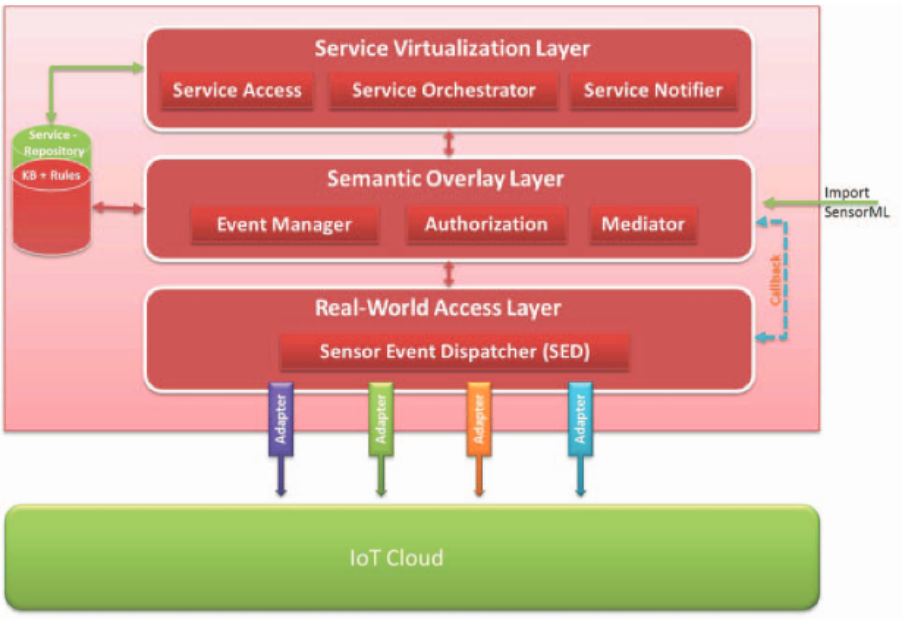

Figure 2: Sensor as a service layers.[4]

consumption.

Rao et al.[24] mention several other research challenges for SaaS including the need for a standard distributed computing framework for distributed big sensor data as well as a framework for the real-time monitoring of sensor events.

3.3 Sensor as a Service

In a Sensor as a Service (SenaaS) [4] service model, virtual and physical sensors are combined according to a Service Oriented Architecture. This type of model concerns itself more with the management of distributed sensors than it does with the access, transfer, and governance of data as the sensing as a service model[30]. Figure 2 shows the three layers that make up the architecture in the SenaaS model. Sensors and events can be defined and standardized in XML and other serialization formats such as SensorML[3] and OWL[11].

The *Real-World Access Layer* interfaces to the sensors using adapters which need to be designed for each sensor. Messages from this layer are asynchronously forwarded to the *Semantic Overlay Layer* via callbacks.

The *Semantic Overlay Layer* is responsible for persisting data either in-memory or on disk. This layer also provides for in-memory caching capabilities. Policy based authorization can be implemented in this layer to provide some control over data access.

The *Service Virtualization Layer* provides an abstraction on top of the semantic overlay layer by performing queries based on the access rights of consumers. This layer transforms the results of queries into something that can be consumed by the clients.

3.4 Cloud Deployment Models

NIST[19] 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.[6] provide aspects from both public and private clouds using virtual private network (VPN) technology to allow users to manage specific and often times complicated network technologies.

3.5 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 pre-processing on-board.

TODO: Expand on this more with an emphasis on the broad range of sensors being used by cellular networks..

3.6 Issues with Cloud Computing

The biggest issues facing cloud computing deal with security and privacy. Subashini et al.[27] go into the specific security and privacy risks associated with the three major cloud deployment models. The following is largely a review of their work.

The deployment model that exhibits 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 concern.

Data security is a concern 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 transferred from clients to the cloud backed 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.

Laws and regulations often require that data not leave or enter certain jurisdictions giving rise to issues of data locality. Users generally do not get to decide where data is stored within a cloud environment as most of those decisions are handled by the cloud provider.

The introduction of distributed systems means that we can no longer make guarantees about data persistence such as ACID (Atomicity, Consistency, Isolation, Durability). Without these guarantees a large amount of data integrity issues surface. We will look at these issues in greater detail when discussion big data persistence models in later chapters.

Data segregation issues occur with multiple virtual resources sharing the same physical resources. Data can be unintentionally leaked or stolen either by attacking the cloud provider multi-tenancy framework or by attacking the virtual servers directly through SQL injection, data validation, and insecure storage.

Large organizations with multiple employees having access to the cloud can create data access issues. Clear policies must be defined, enforced, and updated as to which virtual resources employees have access to.

At some level, consumers 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 vulnerabilities within cloud software and their communication components.

4 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.[15] 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[8]: 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[28] 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[8]. Gilbert[13] 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[8].

The above ideas are all tied together into the CAP theorem proposed by Brewer[7] 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.

4.1 Data Models

4.1.1 Key-Value

The simplest data model for distributed storage is likely the Key-Value (KV) data model [29]. 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 guarantee 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[23].

If the major advantages are simplicity and query response time[8], 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.

4.1.2 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[8]. 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[26]. This is of special concern with heterogeneous sensor networks which can produce large varieties of data in different and changing formats.

Document based data stores often present less of an impedance mismatch between data structures in a programming language and their underlying repre-

sentation 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[23].

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,

4.1.3 Graph

4.1.4 Wide Column Store

4.2 Data Guarentees

4.2.1 title

4.3 Indexing

5 Big Data Analytics

References

[1] Data Science university of wisconsin. `http://datasciencedegree.wisconsin.edu/data-science/what-is-big-data/`.

[2] IBM. `http://www-01.ibm.com/software/data/bigdata/what-is-big-data.html`.

[3] Sensorml. `http://www.opengeospatial.org/standards/sensorml`.

[4] Sarfraz Alam, Mohammad MR Chowdhury, and Josef Noll. Senaas: An event-driven sensor virtualization approach for internet of things cloud. In *Networked Embedded Systems for Enterprise Applications (NESEA), 2010 IEEE International Conference on*, pages 1–6. IEEE, 2010.

[5] Tim Berners-Lee, James Hendler, Ora Lassila, and others. The semantic web. *Scientific american*, 284(5):28–37, 2001.

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

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

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

[9] Michael Cox and David Ellsworth. Managing big data for scientific visualization. In *ACM Siggraph*, volume 97, pages 146–162, 1997.

[10] Sanjit Kumar Dash, Subasish Mohapatra, and Prasant Kumar Pattnaik. A survey on applications of wireless sensor network using cloud computing. *International Journal of Computer science & Engineering Technologies (E-ISSN: 2044-6004)*, 1(4):50–55, 2010.

[11] Mike Dean, Guus Schreiber, Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. OWL web ontology language reference. *W3C Recommendation February*, 10, 2004.

[12] Mario Gerla, Eun-Kyu Lee, Giovanni Pau, and Uichin Lee. Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds. In *Internet of Things (WF-IoT), 2014 IEEE World Forum on*, pages 241–246. IEEE, 2014.

[13] 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.

[14] Ibrahim Abaker Targio Hashem, Ibrar Yaqoob, Nor Badrul Anuar, Salimah Mokhtar, Abdullah Gani, and Samee Ullah Khan. The rise of big data on cloud computing: Review and open research issues. *Information Systems*, 47:98–115, January 2015.

[15] 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.

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

[17] Doug Laney. 3d data management: Controlling data volume, velocity and variety. *META Group Research Note*, 6:70, 2001.

[18] James Manyika, Michael Chui, Brad Brown, Jacques Bughin, Richard Dobbs, Charles Roxburgh, and Angela Hung Byers. Big data: The next frontier for innovation, competition, and productivity. *Big Data: The Next Frontier for Innovation, Competition and Productivity*, pages 1 – 143, 2011.

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

[20] NIST Big Data Public Working Group Definitions and Taxonomies Subgroup. NIST Big Data Interoperability Framework: Volume 1, Definitions. Technical Report NIST SP 1500-1, National Institute of Standards and Technology, October 2015. DOI: 10.6028/NIST.SP.1500-1.

[21] 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.

[22] Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. Sensing as a service model for smart cities supported by Internet of Things. *Transactions on Emerging Telecommunications Technologies*, 25(1):81–93, January 2014.

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

[24] BB Prahlada Rao, Paval Saluia, Neetu Sharma, Ankit Mittal, and Shivay Veer Sharma. Cloud computing for Internet of Things & sensing based applications. In *Sensing Technology (ICST), 2012 Sixth International Conference on*, pages 374–380. IEEE, 2012.

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

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

[27] 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.

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

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

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