|  |  |  |  |
| --- | --- | --- | --- |
| Grifo UA preto | **University of Aveiro**  **2015.** | Department of Electronics, Telecommunications and Informatics | |
| Vedran  Semenski | SMARTIE – Secure and Smarter Cities Data Management | | |
|  | Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em (designação do Mestrado), realizada sob a orientação científica do Doutor (nome do orientador), Professor (categoria do orientador) do Departamento de (designação do departamento) da Universidade de Aveiro | | |
|  | texto Apoio financeiro do  (se aplicável) | |  |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  | texto Dedico este trabalho à minha esposa e filho pelo incansável apoio.  (opcional) |

|  |  |
| --- | --- |
|  |  |
|  |  |
| o júri |  |
| presidente | Prof. Doutor João Antunes da Silva  professor associado da Faculdade de Engenharia da Universidade do Porto |
|  |  |
|  | Prof. Doutor João Antunes da Silva  professor associado da Faculdade de Engenharia da Universidade do Porto |
|  |  |
|  | Prof. Doutor João Antunes da Silva  professor associado da Faculdade de Engenharia da Universidade do Porto |
|  |  |
|  | Prof. Doutor João Antunes da Silva  professor associado da Faculdade de Engenharia da Universidade do Porto |
|  |  |
|  | Prof. Doutor João Antunes da Silva  professor associado da Faculdade de Engenharia da Universidade do Porto |
|  |  |
|  | Prof. Doutor João Antunes da Silva  professor associado da Faculdade de Engenharia da Universidade do Porto |
|  |  |
|  | Prof. Doutor João Antunes da Silva  professor associado da Faculdade de Engenharia da Universidade do Porto |

|  |  |
| --- | --- |
|  |  |
|  |  |
| agradecimentos | texto O presente trabalho propõe-se divulgar as mais significativas técnicas de construção existentes em Portugal continental. O livro é composto por uma apresentação dos materiais tradicionais de construção (suas principais características), uma compilação de fichas técnicas (de carácter prático, uma vasta bibliografia comentada e um glossário de termos técnicos.  A importante colaboração de diversas personalidades ligadas à área da História da Arquitectura, bem como o levantamento fotográfico realizado contribuem para o conhecimento e valorização de um saber tradicional.  (opcional) |

|  |  |
| --- | --- |
|  |  |
|  |  |
| palavras-chave | texto livro, arquitectura, história, construção, materiais de construção, saber tradicional. |
| resumo | texto O presente trabalho propõe-se divulgar as mais significativas técnicas de construção existentes em Portugal continental. O livro é composto por uma apresentação dos materiais tradicionais de construção (suas principais características), uma compilação de fichas técnicas (de carácter prático, uma vasta bibliografia comentada e um glossário de termos técnicos.  A importante colaboração de diversas personalidades ligadas à área da História da Arquitectura, bem como o levantamento fotográfico realizado contribuem para o conhecimento e valorização de um saber tradicional. |

|  |  |
| --- | --- |
|  |  |
|  |  |
| keywords | IoT, M2M, SMARTIE, ABAC, XACML, JSON, Access Control, NoSQL |
| abstract | texto O presente trabalho propõe-se divulgar as mais significativas técnicas de construção existentes em Portugal continental. O livro é composto por uma apresentação dos materiais tradicionais de construção (suas principais características), uma compilação de fichas técnicas (de carácter prático, uma vasta bibliografia comentada e um glossário de termos técnicos.  A importante colaboração de diversas personalidades ligadas à área da História da Arquitectura, bem como o levantamento fotográfico realizado contribuem para o conhecimento e valorização de um saber tradicional. |

# Table of content

[Table of content I](#_Toc411864359)

[List of Acronyms I](#_Toc411864360)

[List of figures III](#_Toc411864361)

[List of tables IV](#_Toc411864362)

[1 Introduction 1](#_Toc411864363)

[1.1 Brief description 1](#_Toc411864364)

[1.2 Defining the goal/concept 1](#_Toc411864365)

[1.3 Basic concept 1](#_Toc411864366)

[2 State of the art 2](#_Toc411864367)

[2.1 The Internet of Things (IoT) 3](#_Toc411864368)

[2.2 Machine to Machine communication (M2M) 5](#_Toc411864369)

[2.3 NoSQL databases and Big Data 6](#_Toc411864370)

[2.3.1 Types of NoSQL databases 6](#_Toc411864371)

[2.3.2 BigData 6](#_Toc411864372)

[2.3.3 General overview of the technologies available at the moment 6](#_Toc411864373)

[2.3.4 Studies and Comparisons 10](#_Toc411864374)

[2.3.5 Conclusion 10](#_Toc411864375)

[2.4 Access Control 11](#_Toc411864376)

[2.4.1 General overview 11](#_Toc411864377)

[2.4.2 Types of access controll 11](#_Toc411864378)

[2.4.3 Comparisons 11](#_Toc411864379)

[2.4.4 ABAC in more detail 11](#_Toc411864380)

[2.4.5 XACML and JSON 11](#_Toc411864381)

[2.4.6 Current Technologies 12](#_Toc411864382)

[2.4.7 Conclusion 12](#_Toc411864383)

[2.5 SMARTIE 12](#_Toc411864384)

[2.5.1 Brief Overview 12](#_Toc411864385)

[2.5.2 General Architecture 12](#_Toc411864386)

[2.5.3 Goals 12](#_Toc411864387)

[2.5.4 Current state 12](#_Toc411864388)

[2.5.5 Future plans 12](#_Toc411864389)

[2.6 Other related work 12](#_Toc411864390)

[3 Temporary 3 13](#_Toc411864391)

[3.1 Temporary 31 13](#_Toc411864392)

[3.1.1 Temporary 311 13](#_Toc411864393)

[4 References 14](#_Toc411864394)

[5 References (other) 16](#_Toc411864395)

# List of Acronyms

|  |  |
| --- | --- |
| ABAC | Attribute-Based Access Control |
| ACL | Access Control List |
| CC | Portuguese Citizen Card |
| CLI | Call Level Interfaces |
| DAC | Discretionary Access Control |
| DACA | Dynamic Access Control Architecture |
| DBMS | Database Management System |
| EKE | Encrypted Key Exchange |
| HDFS | Hadoop Distributed File System |
| IDAC | Direct Access Mode Interface |
| IDE | Integrated Development Environment |
| IIAM | Indirect Access Mode Interface |
| IoT | Internet of Things |
| JDBC | Java Database Connectivity |
| JVM | Java Virtual Machine |
| LDS | Local Data Set |
| LINQ | Language Integrated Query |
| M2m | Machine to machine |
| MAC | Mandatory Access Control / Message Authentication Code |
| NoSQL | Not Only SQL |
| OCSP | Online Certificate Status Protocol |
| PDP | Policy Decision Point |
| PEP | Policy Enforcement Point |
| PSK | Pre-Shared Key |
| RBAC | Role Based Access Control |
| RDBMS | Relational Database Management System |
| SAAM | Secondary and Approximate Authorization Model |
| SDP | Secondary Decision Point |
| S-DRACA | Secure, Dynamic and Distributed Role-based Access Control Architecture |
| SPEKE | Simple Password Exponential Key Exchange |
| SQL | Structured Query Language |
| SRP | Secure Remote Password |
| SSL | Secure Sockets Layer |
| TFA | Two Factor Authentication |
| TLS | Transport Layer Security |

# List of figures

[Figure 1. Random Image 2](#_Toc411506059)

# List of tables

[Table 1. Temporary table 3](#_Toc411506264)

# Introduction

making the introduction and describing how the document is organized into parts with brief descriptions of each part

## Brief description

description of the thesis

## Defining the goal/concept

defining which part was determined as most important and which goals were set

## Basic concept

describing the basic concept of the solution (very briefly)

# State of the art

In this sections the madadnan will be explained and ...

This section is organized in this and this way...

## The Internet of Things (IoT)

In this section the current state of IoT will be presented. The main areas that will be briefly presented and analyzed are: current state and overview, general concerns, recent work and studies, technologies used, implementation examples and expected future developments.

The IoT is a recent paradigm in the area of networks and communication that has had a lot of growth and new developments in recent years. Although there are a lot of definitions of the IoT and the somewhat changing and branching nature of the development trends in this area the basic and simplified idea of this concept is that nowadays everyday objects can be equipped with cheap microcontrollers, sensors, means of connecting to one another and the Internet. The devices can generally be divided into two categories: sensors and actuators, meaning that their purpose is to provide and share data or some kind of readings or to receive commands and react/complete actions accordingly. The overall implementation and use of this could be used in a number of applications including: home automation, industrial automation, medical aids, mobile health care, elderly assistance, intelligent energy management and smart grids, automotive, traffic management and many others [1] [2]. All of these offer beneficial and significant impact on almost all areas of everyday life and have potential for providing advancements in research areas unrelated to networking or computer science. The definition of IoT is not exact as the authors of [1] wrote. They explain that the two main views come from the name "Internet of Things". One vision views the concept from a network perspective concentrating on communication and connection problems while the other is oriented on the "Things" or object perspective concentrating on sensor technologies, new communication technologies like RFID and NFC ,and integration into other devices in a seamless and affordable way. A third view is also present which the authors defined as a "Semantic oriented vision" which is concentrated more on the use, implementation and processing of data.

A number of challenges are in the way of successfully building and utilizing the IoT. Scalability is one of the obvious challenges. Any kind of IoT application that requires a large number of devices commonly face problems with response time, memory, processing and energy constraints. Other challenges include security issues. The data being generated by sensor networks is huge and could over saturate the network, the data could be personal and as such has to be protected from unwanted access. Attacks by hackers, malicious software, viruses and other sources can also disturb the flow and integrity of data/information. As the authors of [3] [4] [5] describe in their work and stress the importance of security for a widely acceptable solution. In their work they describes the IoT divided into 3 or 4 layers and define security requirements for every layer providing the current state of technologies used in these areas. They stress the need for an uniformed and standardised open architecture and solution. Solutions for many of these problems are already conceptually known and are in some examples implemented but the lack of a open and standardized solution is certainly an issue that would proved to be beneficial if/when solved.

Technological advancements in various sensor modules and cheap and energy efficient microcontrollers along with recent communication technologies like RFID, NFC and network protocols are the main factors that enabled the fast development and spreading the IoT. Because of these advancements the concept became a reality and is gaining importance and more uses and applications. As the means of connection and communication are open and utilizes a number of older and newer technologies the networks and number of devices is fast growing and also brings up the problem of standardisation and implementation of standards in the purpose of uniformly solving known problems with for instance security, integrity and scalability.

A more future oriented view and analysis is provided by [6]. It describes a more human centric rather than thing centric future of the IoT. Devices being linked to people and monitoring their state and condition. Others are used for monitoring or controlling things in the environment but always in close relation to human needs and/or wants. It describes a need for IoT applications to provide better quality of service and seamless integration into areas of life. The more relevant and faster growing application domains are: Environmental Monitoring, Smart Retail, Smart Agriculture, Smart Energy and Power Grids and Smart Healthcare. The architecture proposed is separated into 5 layers stacked one on top another in this order:

1. Things - containing devices or elements that are data generators and/or consumers of information. The devices could range from small and unintelligent embedded systems to complex devices or virtual entities. The communication would be done different communication technologies and a wide variety of protocols.
2. Network - this layer contains functionality and means of managing a sensor network. Discovery of new devices, maintenance, scalability, universal abstractions of the Inputs and Outputs and general abstraction for the upper and lower layers and enabling plug-n-play like use using already known models like push/pull or publish/subscribe models and REST based protocols.
3. Data Management - it is defined as "Big-Little" Data Management referring to the data usually generated by sensor networks meaning that the data generated by for instance temperature sensors is small in individual reading size large considering a large number of sensors over a period of time. This layer is responsible for categorizing and aggregating data retrieved from the Network layer in order for it to be used and/or by the Analytics layer. Data generated by sensors is often slow changing so this fact should be taken advantage of for more efficient data storage.
4. Analytics - this layer mines/retrieves data from the Data Management layer and performs data processing and analysis depending on the type of data and end user of the result. Is provides the applications (which would be located on top) with useful data and information for subsequent use.

There a lot of commercial implementations of IoT on a smaller scale which are mainly focused on personal use in home energy consumption, health monitoring and environment monitoring applications. These solution usually utilize custom solutions along with custom hardware (regarding the sensor devices). Introducing standards that would be accepted and the creation of publicly available frameworks that utilize those standards would be very beneficial and would allow easier further developments. The lack of said standards and frameworks mean during initial developments of applications many of them face the same problems and implement a custom solution. This leads to incompatibility issues and stand in the way of a truly global IoT. The surveys and proposals given in [5] [3] give a good overview of the current situation in IoT and a give proposals for more uniformed future research and developments.

## Machine to Machine communication (M2M)

----------------------------------------------------

## Big Data

## NoSQL

NoSQL, or otherwise known as "Non only SQL" refers to databases or data management systems which are designed to handle data management problems where conventional RDBMS solutions cannot cope for various reasons [7] [8]. These problems are usually related to: handling large amounts of data and the processing of it, high number of operations, specific types of operations or needs and others. Commonly, NoSQL systems are designed for large scale data storage and parallel processing over a large number of servers. Some systems provide APIs that support SQL or SQL-like languages and convert them into native non-SQL languages and use mechanisms different to ones in RDBMS. Because they provide the ability to handle large amounts of data it usually comes at the price of fully ACID (Atomicity, Consistency, Isolation, Durability) characteristics. This can be explained by the CAP (strong Consistency, high Availability, Partition tolerance) theorem which can be seen in more detail in [8], and because of this most systems can be described as BASE (Basically Available, Soft-state, Eventually consistent).

### Types of NoSQL databases

NoSQL databases can be classified in four basic categories as done in [8].

* Key-Value stores
* Document databases (or stores)
* Wide-Column (or Column-Family) stores
* Graph databases

Key-Value stores

These storage systems are organized in simple, standalone tables organized like "hash tables". The items stored in tables are key-value pairs where the key is a alpha-numeric identifier and the value is one or a set of values associated with that identifier. As the organisation of table is a "hash table" the limitation that is present is that they are usually limited to only "exact match" type of queries or allowing "<,>" type of operations with a significant reduction in speed. On the other hand read operations are very fast, as to be expected from a hash table data set, and because the keys can also be viewed as the addresses of the value wanted to be retrieved, even data from the same table can be distributed over several locations so these storage systems linear linear characteristics regarding scalability.

Document databases

Document based storage systems, as their name implies, are designed to store data in documents. They use standard data exchange formats such as XML, JSON or BSON to store the data in documents and as can distribute these documents on multiple locations. These are considered scheme-less databases because the storage format or storage data structure can be loosely defined. Single columns or single data entries can house hundreds of values and the number or type of values stored can vary from row to row. These are good for storing and managing big collections of documents containing significant amounts of data like text documents, emails, XML documents or objects containing large amounts of values and data. Along with that they are also convenient for storing sparse data collections because of their schema-less data structure. This means that the usual filling out with null values (that is traditionally done in RDBMS) is not necessary and means that the overall amount of space used is correlated to the amount of data stored inside the database. These solutions offer great scalability, unlike key-value based stores allow multiple "< >" types of comparators and both keys and values are fully searchable. Although they can offer MapReduce [9] features they tend to have slow response times to queries. This reason is because fetching data with multiple set parameters for values means reading and parsing data from whole documents.

Wide-Column Stores

Wide-Column (or Column-Family) stores are somewhat in between document based and key-value based storage systems. They have a structure similar to a key-value structure but allow multiple values and require at least one identifier column which fills the role of a primary key. They can form multiple indexes upon other values and allow "equal type" comparisons over the value attributes. Because of the similarity to key-value based systems they share the same faults regarding "< >" types of operations. The similarity to document based systems comes because they are distributed. The key value can be used to distribute data from a table to multiple locations. This offers good characteristics regarding scalability. Reading and writing operations are fast so they are specially suited for MapReduce operations and parallel processing of large amounts of data which is their main purpose and use.

Graph Databases

Graph databases, by basic concept are relational databases but still are very different from RDBMS in the sense that they also have relations. The relations themselves can be considered as more important because these are used when we mainly need to store date regarding the relationships and dependencies between objects rather than information about the objects themselves. They store data similar to object-oriented databases as use objects as network nodes which have relationships (edges) and properties or object attributes stored as key-value pairs. The relationships can also have different attributes or properties attached to them. They are suited for storing and visualizing data regarding graphs, networks etc.

### General overview of the technologies available at the moment

This section contains an overview of the current NoSQL data storage and management systems, a brief comparison and conclusion.

Hadoop

The Apache Hadoop project develops open-source software for reliable, scalable, distributed computing. The software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures [10].

The project includes these modules:

* Hadoop Common: The common utilities that support the other Hadoop modules.
* Hadoop Distributed File System (HDFS™): A distributed file system that provides high-throughput access to application data.
* Hadoop YARN: A framework for job scheduling and cluster resource management.
* Hadoop MapReduce: A YARN-based system for parallel processing of large data sets.

The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware. HDFS provides high throughput access to application data and is suitable for applications that have large data sets. HDFS relaxes a few POSIX requirements to enable streaming access to file system data. HDFS was originally built as infrastructure for the Apache Nutch web search engine project. HDFS is part of the Apache Hadoop Core project [11].

MapReduce has undergone a complete overhaul in hadoop-0.23 and we now have, what we call, MapReduce 2.0 (MRv2) or YARN. The fundamental idea of MRv2 is to split up the two major functionalities of the JobTracker, resource management and job scheduling/monitoring, into separate daemons. The idea is to have a global ResourceManager (RM) and per-application ApplicationMaster (AM). An application is either a single job in the classical sense of Map-Reduce jobs or a DAG of jobs. The ResourceManager and per-node slave, the NodeManager (NM), form the data-computation framework. The ResourceManager is the ultimate authority that arbitrates resources among all the applications in the system. The per-application ApplicationMaster is, in effect, a framework specific library and is tasked with negotiating resources from the ResourceManager and working with the NodeManager(s) to execute and monitor the tasks [12].

Hadoop MapReduce is a software framework for easily writing applications which process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner. A MapReduce job usually splits the input data-set into independent chunks which are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks. Typically both the input and the output of the job are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks. Typically the compute nodes and the storage nodes are the same, that is, the MapReduce framework and the Hadoop Distributed File System (see HDFS Architecture Guide) are running on the same set of nodes. This configuration allows the framework to effectively schedule tasks on the nodes where data is already present, resulting in very high aggregate bandwidth across the cluster. The MapReduce framework consists of a single master ResourceManager, one slave NodeManager per cluster-node, and MRAppMaster per application (see YARN Architecture Guide). Minimally, applications specify the input/output locations and supply map and reduce functions via implementations of appropriate interfaces and/or abstract-classes. These, and other job parameters, comprise the job configuration. The Hadoop job client then submits the job (jar/executable etc.) and configuration to the ResourceManager which then assumes the responsibility of distributing the software/configuration to the slaves, scheduling tasks and monitoring them, providing status and diagnostic information to the job-client [13].

Apache Hive

Apache Hive is a data warehouse system for Apache Hadoop. It has been widely used in organizations to manage and process large volumes of data, such as eBay, Facebook, LinkedIn, Spotify, Taobao, Tencent, and Yahoo!. As an open source project, Hive has a strong technical development community working with widely located and diverse users and organizations. Hive was originally designed as a translation layer on top of Hadoop MapReduce. It exposes its own dialect of SQL to users and translates data manipulation statements (queries) to a directed acyclic graph (DAG) of MapReduce jobs [14].

The Apache HiveTM data warehouse software facilitates querying and managing large datasets residing in distributed storage. Built on top of Apache HadoopTM, it provides: Tools to enable easy data extract/transform/load (ETL), a mechanism to impose structure on a variety of data formats, access to files stored either directly in Apache HDFSTM or in other data storage systems such as Apache HBaseTM, query execution via MapReduce. Hive defines a simple SQL-like query language, called QL, that enables users familiar with SQL to query the data. At the same time, this language also allows programmers who are familiar with the MapReduce framework to be able to plug in their custom mappers and reducers to perform more sophisticated analysis that may not be supported by the built-in capabilities of the language. QL can also be extended with custom scalar functions (UDF's), aggregations (UDAF's), and table functions (UDTF's). Hive does not mandate read or written data be in the "Hive format" — there is no such thing. Hive works equally well on Thrift, control delimited, or your specialized data formats. Please see File Formats and Hive SerDe in the Developer Guide for details. Hive is not designed for OLTP workloads and does not offer real-time queries or row-level updates. It is best used for batch jobs over large sets of append-only data (like web logs). What Hive values most are scalability (scale out with more machines added dynamically to the Hadoop cluster), extensibility (with MapReduce framework and UDF/UDAF/UDTF), fault-tolerance, and loose-coupling with its input formats. Components of Hive include HCatalog and WebHCat. HCatalog is a component of Hive. It is a table and storage management layer for Hadoop that enables users with different data processing tools — including Pig and MapReduce — to more easily read and write data on the grid. WebHCat provides a service that you can use to run Hadoop MapReduce (or YARN), Pig, Hive jobs or perform Hive metadata operations using an HTTP (REST style) interface [15].

Apache Spark

Spark is also an Apache open source (since 2010) project. It is a fast and general processing engine compatible with Hadoop data. It can run in Hadoop clusters through YARN or Spark's standalone mode, and it can process data in HDFS, HBase, Cassandra, Hive, and any Hadoop InputFormat. It is designed to perform both batch processing (similar to MapReduce) and new workloads like streaming, interactive queries, and machine learning [16]. It is built around speed, ease of use, and sophisticated analytics. It was originally developed at UC Berkeley in 2009. Databricks was founded by the creators of Spark in 2013. The engine runs in a variety of environments, from cloud services to Hadoop or Mesos clusters. It is used to perform ETL, interactive queries (SQL), advanced analytics (e.g. machine learning) and streaming over large datasets in a wide range of data stores (e.g. HDFS, Cassandra, HBase, S3). Spark supports a variety of popular development languages including Java, Python and Scala. Since its release, it has seen rapid adoption by enterprises across a wide range of industries. It has quickly become the largest open source community in big data, with over 400 contributors from 100+ organizations. It provides easy-to-use APIs for operating on large datasets. This includes a collection over 80 operators for transforming data and familiar data frame APIs for manipulating semi-structured data. It is packaged with higher level libraries, including support for SQL queries, streaming data, machine learning and graph processing. These standard libraries increase developer productivity and can be seamlessly combined to create complex workflows.

Apache HBase

HBase is a type of "NoSQL" database. "NoSQL" is a general term meaning that the database isn’t an RDBMS which supports SQL as its primary access language, but there are many types of NoSQL databases: BerkeleyDB is an example of a local NoSQL database, whereas HBase is very much a distributed database. Technically speaking, HBase is really more a "Data Store" than "Data Base" because it lacks many of the features you find in an RDBMS, such as typed columns, secondary indexes, triggers, and advanced query languages, etc.

However, HBase has many features which supports both linear and modular scaling. HBase clusters expand by adding RegionServers that are hosted on commodity class servers. If a cluster expands from 10 to 20 RegionServers, for example, it doubles both in terms of storage and as well as processing capacity. RDBMS can scale well, but only up to a point - specifically, the size of a single database server - and for the best performance requires specialized hardware and storage devices. HBase features of note are:

* Strongly consistent reads/writes: HBase is not an "eventually consistent" DataStore. This makes it very suitable for tasks such as high-speed counter aggregation.
* Automatic sharding: HBase tables are distributed on the cluster via regions, and regions are automatically split and re-distributed as your data grows.
* Automatic RegionServer failover
* Hadoop/HDFS Integration: HBase supports HDFS out of the box as its distributed file system.
* MapReduce: HBase supports massively parallelized processing via MapReduce for using HBase as both source and sink.
* Java Client API: HBase supports an easy to use Java API for programmatic access.
* Thrift/REST API: HBase also supports Thrift and REST for non-Java front-ends.
* Block Cache and Bloom Filters: HBase supports a Block Cache and Bloom Filters for high volume query optimization.
* Operational Management: HBase provides build-in web-pages for operational insight as well as JMX metrics.

[HDFS](http://hadoop.apache.org/hdfs/) is a distributed file system that is well suited for the storage of large files. Its documentation states that it is not, however, a general purpose file system, and does not provide fast individual record lookups in files. HBase, on the other hand, is built on top of HDFS and provides fast record lookups (and updates) for large tables. This can sometimes be a point of conceptual confusion. HBase internally puts your data in indexed "StoreFiles" that exist on HDFS for high-speed lookups.

Apache Casandra

MongoDB

Arango DB

Oracle NoSQL Database

OrientDB

Redis

### Studies and Comparisons

### Conclusion

## Access Control

### General overview

### Types of access controll

IBAC

RBAC

Diogo variant

ABAC

Other

### Comparisons

### ABAC in more detail

### XACML and JSON

### Current Technologies

### Conclusion

## SMARTIE

### Brief Overview

### General Architecture

### Goals of SMARTIE

### Current state

use cases and current accomplishments

### Future plans

## Other related work

# Temporary 3

## Temporary 31

### Temporary 311

# References

|  |  |
| --- | --- |
| [1] | A. I. G. M. Luigi Atzori, “The Internet of Things: A survey,” *Computer Networks 54,* p. 2787–2805, 14 June 2010. |
| [2] | N. B. A. C. V. M. Z. Andrea Zanella, “Internet of Things for Smart Cities,” 2014 April 2014. [Online]. Available: http://eprints.networks.imdea.org/id/eprint/740. [Accessed 12 February 2015]. |
| [3] | A. V. V. J. W. J. L. D. Q. Qi Jing, “Security of the Internet of Things: perspectives and challenges,” *Wireless Netw,* p. 20:2481–2501, 17 June 2014). |
| [4] | J. W. C. Z. J. L. Hui Suo, “Security in the Internet of Things: A Review,” in *International Conference on Computer Science and Electronics Engineering*, 2012. |
| [5] | A. R. L. G. A. C.-P. S. Sicari, “Security, privacy and trust in Internet of Things: The road ahead,” *Computer Networks 76,* p. 146–164, 15 July 2014. |
| [6] | Y. S. J. W. Prasant Misra, “Towards a Practical Architecture for the Next Generation Internet of Things,” 3 February 2015. [Online]. Available: http://arxiv.org/pdf/1502.00797v1.pdf. [Accessed 10 Februry 2015]. |
| [7] | “Alternatives to relational database: Comparison of NoSQL and XML approaches for clinical data storage,” *computer methods and programs in biomedicine 110,* p. 99–109, 2013. |
| [8] | S. A. H. A B M Moniruzzaman, “NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison,” *International Journal of Database Theory and Application,* 2013. |
| [9] | “MongoDB HomePage,” MongoDB, Inc., [Online]. Available: http://www.mongodb.org/. [Accessed 7 February 2015]. |
| [10] | “Apache Hadoop,” The Apache Software Foundation, 12 12 2014. [Online]. Available: http://hadoop.apache.org/. [Accessed 7 February 2015]. |
| [11] | “Apache Hadoop HDFS,” Apache Software Foundation, 13 11 2014. [Online]. Available: http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html. [Accessed 6 February 2015]. |
| [12] | “Apache Hadoop YARN,” Apache Software Foundation, 11 13 2014. [Online]. Available: http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html. [Accessed 7 February 2015]. |
| [13] | “Apache Hadoop MapReduce,” Apache Software Foundation, 13 11 2014. [Online]. Available: http://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html. [Accessed 7 February 2015]. |
| [14] | Y. H. A. C. A. G. G. H. E. N. Hanson, “Major Technical Advancements in Apache Hive,” in *SIGMOD '14 Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, New York, NY, USA, 2014 . |
| [15] | “Apache Hive,” The Apache Software Foundation, 12 January 2015. [Online]. Available: https://cwiki.apache.org/confluence/display/Hive/Home. [Accessed 7 February 2015]. |
| [16] | “Apache Spark,” The Apache Software Foundation, [Online]. Available: http://spark.apache.org/. [Accessed 7 February 2015]. |
| [17] | R. B. S. M. M. P. Jayavardhana Gubbi, “Internet of Things (IoT): A vision, architectural elements, and future directions,” *Future Generation Computer Systems 29,* p. 1645–1660, 24 February 2013.. |
| [18] | M. C. M. J. F. S. S. I. S. Matei Zaharia, “Spark: Cluster Computing with Working Sets,” 2010. |
| [19] | “Apache Hbase,” The Apache Software Foundation, 11 Februaray 2015. [Online]. Available: http://hbase.apache.org/book.html. [Accessed 15 February 2015]. |
| [20] | “A comparison between several NoSQL databases with comments and notes”. |

# References (other)

|  |  |
| --- | --- |
| [1] | A. I. G. M. Luigi Atzori, “The Internet of Things: A survey,” *Computer Networks 54,* p. 2787–2805, 14 June 2010. |
| [2] | N. B. A. C. V. M. Z. Andrea Zanella, “Internet of Things for Smart Cities,” 2014 April 2014. [Online]. Available: http://eprints.networks.imdea.org/id/eprint/740. [Accessed 12 February 2015]. |
| [3] | A. V. V. J. W. J. L. D. Q. Qi Jing, “Security of the Internet of Things: perspectives and challenges,” *Wireless Netw,* p. 20:2481–2501, 17 June 2014). |
| [4] | J. W. C. Z. J. L. Hui Suo, “Security in the Internet of Things: A Review,” in *International Conference on Computer Science and Electronics Engineering*, 2012. |
| [5] | A. R. L. G. A. C.-P. S. Sicari, “Security, privacy and trust in Internet of Things: The road ahead,” *Computer Networks 76,* p. 146–164, 15 July 2014. |
| [6] | Y. S. J. W. Prasant Misra, “Towards a Practical Architecture for the Next Generation Internet of Things,” 3 February 2015. [Online]. Available: http://arxiv.org/pdf/1502.00797v1.pdf. [Accessed 10 Februry 2015]. |
| [7] | “Alternatives to relational database: Comparison of NoSQL and XML approaches for clinical data storage,” *computer methods and programs in biomedicine 110,* p. 99–109, 2013. |
| [8] | S. A. H. A B M Moniruzzaman, “NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison,” *International Journal of Database Theory and Application,* 2013. |
| [9] | “MongoDB HomePage,” MongoDB, Inc., [Online]. Available: http://www.mongodb.org/. [Accessed 7 February 2015]. |
| [10] | “Apache Hadoop,” The Apache Software Foundation, 12 12 2014. [Online]. Available: http://hadoop.apache.org/. [Accessed 7 February 2015]. |
| [11] | “Apache Hadoop HDFS,” Apache Software Foundation, 13 11 2014. [Online]. Available: http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html. [Accessed 6 February 2015]. |
| [12] | “Apache Hadoop YARN,” Apache Software Foundation, 11 13 2014. [Online]. Available: http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html. [Accessed 7 February 2015]. |
| [13] | “Apache Hadoop MapReduce,” Apache Software Foundation, 13 11 2014. [Online]. Available: http://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html. [Accessed 7 February 2015]. |
| [14] | Y. H. A. C. A. G. G. H. E. N. Hanson, “Major Technical Advancements in Apache Hive,” in *SIGMOD '14 Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, New York, NY, USA, 2014 . |
| [15] | “Apache Hive,” The Apache Software Foundation, 12 January 2015. [Online]. Available: https://cwiki.apache.org/confluence/display/Hive/Home. [Accessed 7 February 2015]. |
| [16] | “Apache Spark,” The Apache Software Foundation, [Online]. Available: http://spark.apache.org/. [Accessed 7 February 2015]. |
| [17] | R. B. S. M. M. P. Jayavardhana Gubbi, “Internet of Things (IoT): A vision, architectural elements, and future directions,” *Future Generation Computer Systems 29,* p. 1645–1660, 24 February 2013.. |
| [18] | M. C. M. J. F. S. S. I. S. Matei Zaharia, “Spark: Cluster Computing with Working Sets,” 2010. |
| [19] | “Apache Hbase,” The Apache Software Foundation, 11 Februaray 2015. [Online]. Available: http://hbase.apache.org/book.html. [Accessed 15 February 2015]. |
| [20] | “A comparison between several NoSQL databases with comments and notes”. |