Behaviour & Information Technology

ISSN: 0144-929X (Print) 1362-3001 (Online) Journal homepage: https://www.tandfonline.com/loi/tbit20

Applying big data and stream processing to the real estate domain

Herminio García-González, Daniel Fernández-Álvarez, José Emilio Labra-Gayo & Patricia Ordóñez de Pablos

To cite this article: Herminio García-González, Daniel Fernández-Álvarez, José Emilio Labra-Gayo & Patricia Ordóñez de Pablos (2019): Applying big data and stream processing to the real estate domain, Behaviour & Information Technology, DOI: 10.1080/0144929X.2019.1620858

To link to this article: https://doi.org/10.1080/0144929X.2019.1620858

Published online: 24 May 2019.

BEHAVIOUR & INFORMATION TECHNOLOGY
https://doi.org/10.1080/0144929X.2019.1620858

Applying big data and stream processing to the real estate domain

Herminio García-González[a], Daniel Fernández-Álvarez[a], José Emilio Labra-Gayo[a] and Patricia Ordóñez de Pablos[b]

[a]Department of Computer Science, University of Oviedo, Oviedo, Spain; [b]Department of Business Administration, University of Oviedo, Oviedo, Spain

ABSTRACT
In this paper, we propose an architecture that combines Big Data and Stream Processing which can be applied to the Real Estate Domain. Our approach consists of a specialisation of Lambda architecture and it is inspired by some aspects of Kappa architecture. As a proof of this solution, we show a prototype developed following it and a comparison of the three architecture quality models. Finally, we highlight the differences between the proposed architecture and similar ones and draw some future lines following the present approach.

ARTICLE HISTORY
Received 28 February 2018
Accepted 11 May 2019

KEYWORDS
Big data; stream processing; real estate; software architecture

1. Introduction

Big data has supposed the appearance of a new set of applications that could handle the new challenge of big and growing datasets and offer new visualisations, analytics and, definitely, useful information to users among different fields. These techniques combined with new advances in hardware could open new possibilities that in the past could not been reached due to state-of-the-art limitations.

But new challenges, linked to the big amount of data, have emerged. One of them is the real-time data. Real-time data has the characteristic of changing in short periods of time, i.e. it is data that quantifies something as it happens. This kind of data is also growing due to advances in Internet of Things (IoT) (Wingerath et al. 2016) where a lot of sensors are capturing environmental data and publishing it through the Internet.

One of the fields that are taking benefit of this development on big data technologies and techniques is real estate business. Real estate business produces a lot of information, including transactions, prices, new constructions, etc. Exploitation of this data is crucial to make a difference in real estate domain. Furthermore, being capable of predicting some indicators based on existing data could be the next big step in this field and a current research interest among the community.

In order to achieve these purposes, there is the need of solutions that can handle the existing and upcoming real estate data. In order to serve to this purpose, in this paper we present a specialisation of Lambda architecture specifically designed for real estate domain called Adaptation of Lambda Architecture for Real Estate Applications (Alarea).

Thus, the structure of this paper is as follows: Section 2 describes related work. In Section 3, we describe the proposed architecture and in Section 4, we make a comparison of the architecture quality models. We describe our prototype implementation in Section 5. In Section 6, we establish a discussion about the benefits of the proposed architecture compared with similar architectures. Finally, in Section 7 we present some conclusions and future work.

2. Related work

2.1. Big data and streaming

Over the past 25 years, data has dramatically increased in various fields (Chen, Mao, and Liu 2014). Activity in social networks or sensors lectures linked to IoT have contributed to this data explosion (Marz and Warren 2015; Rao and Ali 2015). The term big data emerges as a way to describe those huge datasets which cannot be handled by just using classic batch approaches. There are some distinguishing features that define the universe of big data applications. These features were originally three and they were known as the *3Vs model* (Laney 2001). Nowadays, this model has evolved to a *5Vs* one (Demchenko et al. 2013), which is composed by:

- *Volume*: It refers to the huge amount of data that they handle.

CONTACT Herminio García-González herminiogg@gmail.com

- *Velocity*: Systems that work with big data usually should produce an output in a certain time frame, which may be a real-time answer.
- *Variety*: Many applications have to deal not just with enormous volumes of bytes but also to process information of different formats or types (messages, videos, structured data, …) produced by different sources (sensors, social networks, RSS of news, etc.).
- *Veracity*: False or corrupted information may lead to wrong conclusions and failed applications. Veracity is incorporated to the *3V* to emphasise the necessity of developing applications that consider security aspects in order to ensure the quality and confidentiality of the data.
- *Value*: This *V* was incorporated to the original model of *3Vs* in order to highlight the necessity of obtaining valuable pieces of information among the entire dataset. In Marz and Warren (2015), the difference between raw data (pre-analysis) and valuable information (post-analysis) is established.

Hardware systems have evolved in order to satisfy those needs, being specially significant the apparition of Solid State Disk (SSD) technology to improve the capacity and performance of memory-level data analysis (Chen, Mao, and Liu 2014). At the same time, software architectures specially designed to be used in big data scenarios have been purposed, looking for high levels of scale-out, elasticity and availability (Katal, Wazid, and Goudar 2013). Probably one of the most significant contributions in this field is MapReduce model. Initially proposed by Google (Dean and Ghemawat 2008), MapReduce was widely adopted by the community once it became an open source project with the implementation of Apache Hadoop Foundation (2011). The model proposes a simple scheme based on two functions (Map and Reduce, which should be provided by the developer) which allows for an easy horizontal scaling.

Real-time or pseudo-real-time systems raise extra technological challenges that cannot be handled by purely batch approaches. These scenarios require systems able to update their algorithms with the arrival of new data, to work within the required time constraints and to deal with memory limitations (Ramírez-Gallego et al. 2017). In order to fulfil these requirements, stream processing emerges as an alternative to batch processing.

Although stream processing is frequently associated to big data applications, the idea of computing data in motion is not new. We can find early examples of similar approaches in some Database Management Systems (DBMS) (Terry et al. 1992) or Complex Event Processing (CEP) engines (Abadi et al. 2003, 2005). In big data scenarios, streams are frequent when processing user-generated content or lectures from sensors, wearables or any device associated to IoT (Wingerath et al. 2016).

Real-time systems in big data should produce nearly immediate responses based on huge amounts of information, which require maximum performance and minimum latency. In order to reach such low latencies, stream processing methods need to reduce the complexity of the raw data (García, Luengo, and Herrera 2016), with the cost of just being able to compute small pieces of data each time. A purely stream-based approach cannot use the potential knowledge reachable by computing the whole dataset (Vanhove et al. 2016). In order to take advantage of both approaches (batch and stream), Lambda architecture is proposed (Marz and Warren 2015). Lambda describes two layers to make parallel and different computations of the same input data. By that, an application which implements Lambda is able to offer real-time feedback as well as pieces of information not so up-to-date obtained with batch processes of higher latencies.

Despite the general acceptance of Lambda architecture (Perera and Suhothayan 2015), this solution has been criticised due to the need of maintaining two code bases for the separate processing layers (Kreps 2014). Kappa architecture is suggested as an alternative, consisting in a simplification of Lambda by removing the batch processing layer. Nevertheless, while this approach effectively tackles the problem of maintaining two code bases, it also restricts some analytic results that just can be reached by processing the source dataset as a whole.

Several stream processing systems have been provided in the last years, including Storm (Foundation 2016b), Trident (Foundation 2016c), Samza (Foundation 2016a) and Spark Streaming Foundation (2014). Spark streaming is linked to the Spark framework (Zaharia et al. 2012), which consists of a batch-processing framework, sometimes considered the successor of Hadoop. This combination of batch and streaming processing makes Spark a great option for hybrid approaches such as Lambda or Alarea. A comparison of all these systems is provided in Wingerath et al. (2016).

2.2. Real Estate applications

Big data analytics have been successfully applied in several Real Estate applications to support decisions of both buyers and sellers. In general, these systems have two main aims:

- Decision support, such as Dujmović et al. (2013) and Montes et al. (2015). Based on soft computing models, they assist users' choices based on their preferences, profile and diverse available information. They are more oriented to buyers.

- Predictions, such as Trawinski (2013) and Rafiei and Adeli (2015). They compute different types of data with machine-learning algorithms in order to develop predictive models about the evolution of supply, demand and pricing. They are more oriented to sellers.

Some current Real Estate applications require the use of techniques related to big data due to the massive amount of data with potential to be exploited in this kind of market (Du, Li, and Zhang 2014). It is not just about computing supply and demand and users preferences, but many other types of information may be used as parameters to develop accurate models. In Du, Li, and Zhang (2014), a list of Chinese companies which have analysed different types of data for the real Estate market is shown, including but not limited to land resources and owners personal information (Vanke), owners health condition (Shimao Group), or even drivers' paths obtained via Global Positioning System (Windermere).

Some of the potential information processed in real estate models may be exposed and computed via streams. In Trawinski (2013), a predictive model based on correction of aged models via stream processing is presented. In this case, the authors do not look for a system with real-time responses, but they propose a model based in the computation of real-time events. An insightful example of a relevant stream-like notion considered in some real estate models is sentiment analysis (Marcato and Nanda 2016). Several works have successfully employed sentiment indexes to improve predictive models in house pricing, such as the NAHB/Wells Fargo Housing Market Index (HMI), based on a survey of home builders, or the Reuters/University of Michigan Index of Consumer Sentiment (Nanda 2007; Dua 2008; Jin, Soydemir, and Tidwell 2014; Ling, Naranjo, and Scheick 2014). While those works use sentiment measures based on surveys, current state-of-art sentiment analysis allows for the development of price prediction models using real-time opinion mining in more stream-like sources, such as search engines or social media (Smailović, Grčar, and Lavrač 2014; Zhang, Zhao, and Xu 2016). In Wu and Brynjolfsson (2015), the authors find evidence of correlation between search terms and house pricing variation. They also argue that real estate market is a good context to apply predictive models based on on-line sentiment analysis. This is because events viralised through social media with short-term consequences (a few days) tend to have a softer impact over consumers' criteria, since the transactions in real estate market use to require long decision times.

The potential of utilising this data in real estate models motivates the development of architectures such as Alarea, useful for handling big data sources and streams of information.

3. Architecture proposal

3.1. Lambda architecture

Lambda architecture (Marz and Warren 2015) is an architecture proposed by Nathan Marz that aims to process big data sources and perform the commended tasks by minimising the latency of this kind of works.

For this purpose, lambda architecture is splitted into three layers that are as follows:

- Batch layer: This layer is responsible for making transformations, calculations and aggregations that are required through the existing data on the master dataset. Likewise of integrating the new data that arrives to the architecture.
- Serving layer: This layer should generate different views for the desired queries. These views can be precomputed and cached in order to speed the process of querying.
- Speed layer: This layer arise as a solution when batch processes suffer from latency or when new data is coming in short periods of time. As we mentioned, batch processes could take long time to refresh data on batch views. Therefore, speed layer makes use of real-time technologies in order to complete queries with the most up-to-date data.

Combining these two paradigms: batch layer and speed layer, lambda architecture offers a solid big data approach with the combination of no latency and up-to-date data (Figure 1).

3.2. Kappa architecture

Kappa architecture (Kreps 2014) was proposed by Jay Kreps as a simplification of Lambda architecture. The proposal of this architecture aimed to remove the batch layer used on Lambda architecture. Instead of using a master dataset, all operations will be based on the speed layer and, therefore, in an append-only immutable log.

Serving layer is maintained but instead of making migrations of databases each time that new data come, new versions of the database are made by taking the desired information from the immutable log.

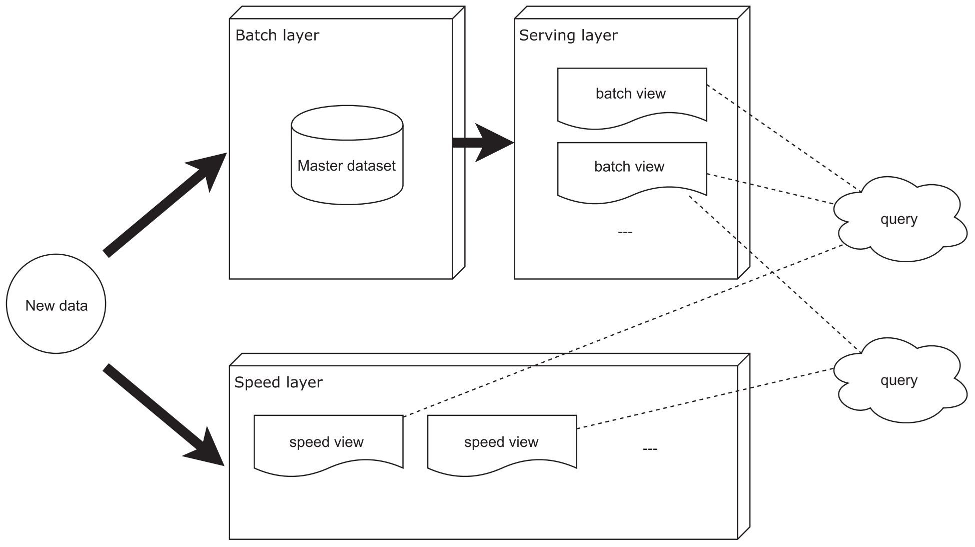


Figure 1. Diagram of lambda architecture. Source: http://lambda-architecture.net

3.3. Alarea architecture

Our proposal of architecture materialise in the Alarea architecture. This architecture is a specialisation of Lambda architecture and takes some ideas from the Kappa architecture. Alarea architecture aims to bring the big amount of data stored in public, governmental and social media platforms to join them together in a unique source of information in order to offer them to possible consumers in a usable and more convenient form. Alarea architecture gathers two different forms of data: real-time data and non-real-time data. Non-real-time data includes sources of information that are not variable along short periods of time and therefore could be managed using batch processing techniques. Meanwhile, real-time data includes sources of information that are variable through a little time window portion, i.e. they are changing continually.

Therefore, to fulfil the requirements proposed in the previous paragraph, we propose an architecture and a set of technologies that can handle these two types of data. Moreover, this architecture not only integrates heterogeneous data, it also integrates heterogeneous timing data.

Alarea architecture combines two styles of processing, as we introduced earlier: batch processing and real-time processing. With batch processing, this architecture is capable of handling big amount of data without the problem of getting stuck or showing a high latency effect. Using the advances made in big data technologies, it can offer a fast, reliable and parallelised processing of non-real-time data. However, real-time data is a new paradigm of data that architectures should be aware of processing. Alarea architecture uses advances in real-time processing to handle and gather also this kind of data.

Alarea presents two main layers: batch layer and real-time layer. Batch layer is aware of processing data that could be stored in all kinds of non-real-time data formats (e.g. JSON, XML, Excel files, CSV, JDBC, etc.). This layer is responsible for parsing, transforming and storing the data. In the other hand, real-time layer is aware of processing data that is served as real-time data and therefore is changing in a little window time. As well as with batch layer, real-time data is also responsible for transformations that could be delegated as part of this process.

Batch layer is composed by non-real-time data sources, the batch processor and the distributed persistence. Therefore, the batch processor will make the necessary transformations from the non-real-time sources to the distributed persistence. The main goal of batch processor is to reach this conversion in a distributed way, that is, it should increase horizontally with a cluster of computers instead of vertically, with more hardware in a single machine.

Real-time layer is composed by real-time sources, the real-time processor and the distributed event queue. Therefore, real-time processor will make the necessary transformations from the sources to the distributed event queue. By its side, distributed event queue presents a publish subscribe architecture that must handle publication of new events of changed data and subscription and notifications of these events to all the subscribers into the topic. This layer should be also parallelised in order to make it horizontally growable as we stated with batch layer.

Finally, the integrator should consume data from both layers and create new views from the received data. It

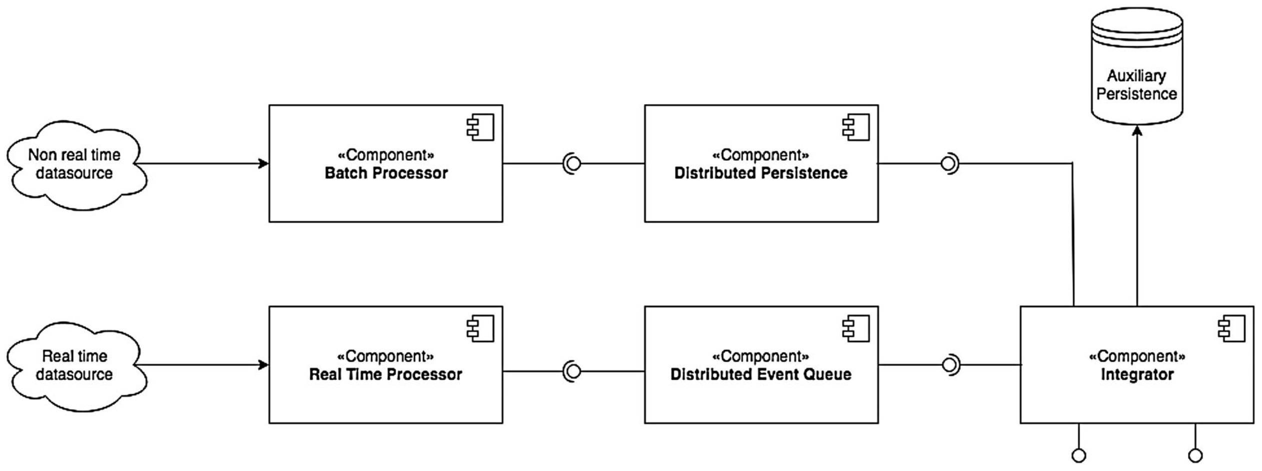

Auxiliary Persistence
Non real time datasource
«Component» Batch Processor
«Component» Distributed Persistence
Real time datasource
«Component» Real Time Processor
«Component» Distributed Event Queue
«Component» Integrator

Figure 2. Proposed architecture diagram.

could use an auxiliary repository in order to cache the generated views. A diagram of this architecture can be seen in Figure 2.

4. Quality model

In this section, we describe the quality model of the three in-study architectures by means of a quality attributes comparison. For simplicity, the serving layer has not been taking into account as the three architectures have this layer in common. The selected quality attributes are:

- Recoverability: Ability of a system to recover to a stable point once a failure has been produced.
- Fault tolerance: Property from which a system can continue working when a component has failed.
- New data gap: Measurement of the waiting time from when a new data is received until it is completely available.
- Hardware consumption: Amount of required hardware resources needed by an architecture to operate correctly.
- Modifiability: Property of an architecture that allows the modification of a component using the minimum time possible.

Table 1. Quality attributes comparison.

Quality attributes	Lambda architecture	Kappa architecture	Alarea
Recoverability	H	M	M
Fault tolerance	M	M	M
New data gap	L	H	M
Hardware consumption	H	M	M
Modifiability	L	H	H

H, High; M, Medium; L, Low.

In Table 1, we have detailed our evaluation for each quality attribute in each architecture which was based in the following criteria.

4.1. Recoverability

Recoverability can be defined as the ability that a system has to recover from a failure and to restore to a stable point. From this point of view, Lambda architecture presents high recoverability due to its double layer, batch layer and speed layer, which provides two sources of redundant information. In the case of Kappa architecture, the only existence of a speed layer makes that all the knowledge is hosted in the so-called 'source of truth' which, if it is not replicated, can deal to the loss of information. However, the append-only system can also excel the reconstructions of the produced events. Alarea, from its side, suffers from the same problem as Kappa architecture which makes that, if not being replicated by default, it could lead to the loss of information. Nonetheless, it can take benefit from the append-only log in the streaming layer and also batch layer can be recomputed from existing sources. Therefore, recoverability is rated as high in Lambda architecture whereas in Kappa architecture and Alarea is rated as medium.

4.2. Fault tolerance

This is the ability of a system to continue working when one of its components has failed. Lambda architecture, thanks to its double layer design, can deal with failures in one of its components. However, if this is produced in the stream layer the latency will be increased. Therefore, it can be labelled as medium. Kappa architecture, due to its append-only log design, can deal with the reconstruction of the events produced in the system.

Although, this kind of reconstruction can increase the latency as well. Hence, it can be labelled as medium. The case of Alarea is similar to the Kappa architecture where the data can be reconstructed from the append-only log in the streaming layer and from the recomputation in the batch layer. Therefore, it can be labelled as medium.

4.3. New data gap

New data gap is the scenario when new data arrives to the architecture until it is fully queryable. Lambda architecture is designed with this quality attribute in mind, due to its double layer new data is aggregated to the views by means of the speed layer while it is processed by the batch layer. Therefore, the latency is very low. In the case of Kappa architecture, the latency will be derived from the recomputation of the views which, depending on its complexity, could take more or less time. In Alarea, the recomputation is needed for both layers and it could take some time as in Kappa architecture. However, in this architecture batch layer is only meant for data with low frequency change rate and recomputation will be only made continuously in the streaming layer. Therefore, the fastest recomputation is made on Lambda architecture whereas Kappa architecture and Alarea suffer from similar latencies.

4.4. Hardware consumption

Hardware consumption is a measure that gives an impression of which hardware is required for a concrete architecture. In Lambda architecture, every time new data is received the batch and the streaming layer must do a recomputation. Therefore, the hardware consumption in Lambda architecture can be labelled as high. In Kappa architecture, the only existence of the stream layer oblies to make a recomputation of the views each time new data is received. In contrast to the Lambda architecture, only one layer has the computation. Therefore, the hardware consumption can be labelled as medium. In Alarea, the recomputation is made only in the layer affected depending on the kind of new data. This is more or less the same scenario as in the Kappa architecture where only one layer recomputation is needed. Therefore, it can be labelled as medium.

4.5. Modifiability

Modifiability is the property of an architecture to be modified within the lowest time possible. In Lambda architecture, a modification in a batch layer component would require to modify its counterpart component in the speed layer. Therefore, it can be labelled as low. In Kappa architecture, a modification is isolated to the only existing speed layer. Therefore, it can be labelled as high. In the case of Alarea architecture, the modification will be isolated to one of the layers depending on the kind of data. Therefore, it can be labelled as high.

5. Prototype implementation

For the sake of architecture validation, we have developed a small prototype that is based on the proposed one. This prototype is intended for real estate data processing and viewing, and it uses both real-time data and non-real-time data.

Batch layer uses data from Spanish Ministry of Development (2004-2016) which represents the values of real estate transactions on different categories. These files are in Excel format and they are processed, transformed and persisted by the batch layer.

The main components of the batch layer are:

- Spark + Hive: Spark is responsible for doing the transformation and persistence of data into Hive technology. Spark takes the data in Excel format files, transform and aggregate it into tables to persist as

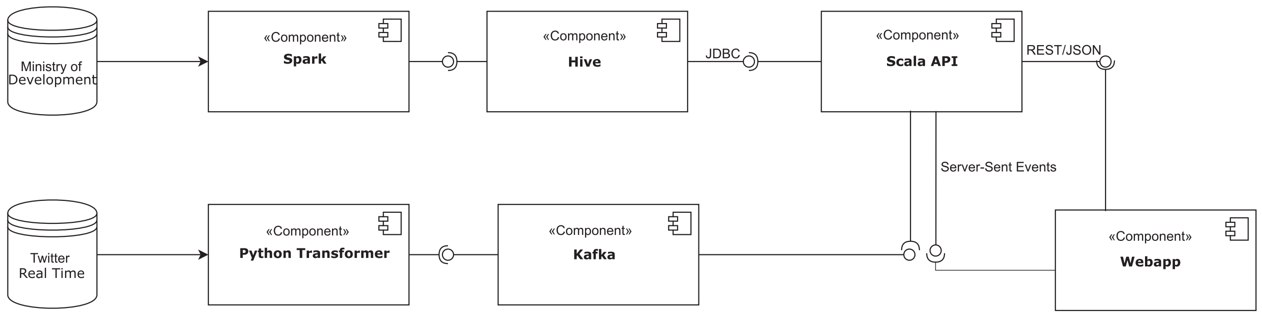
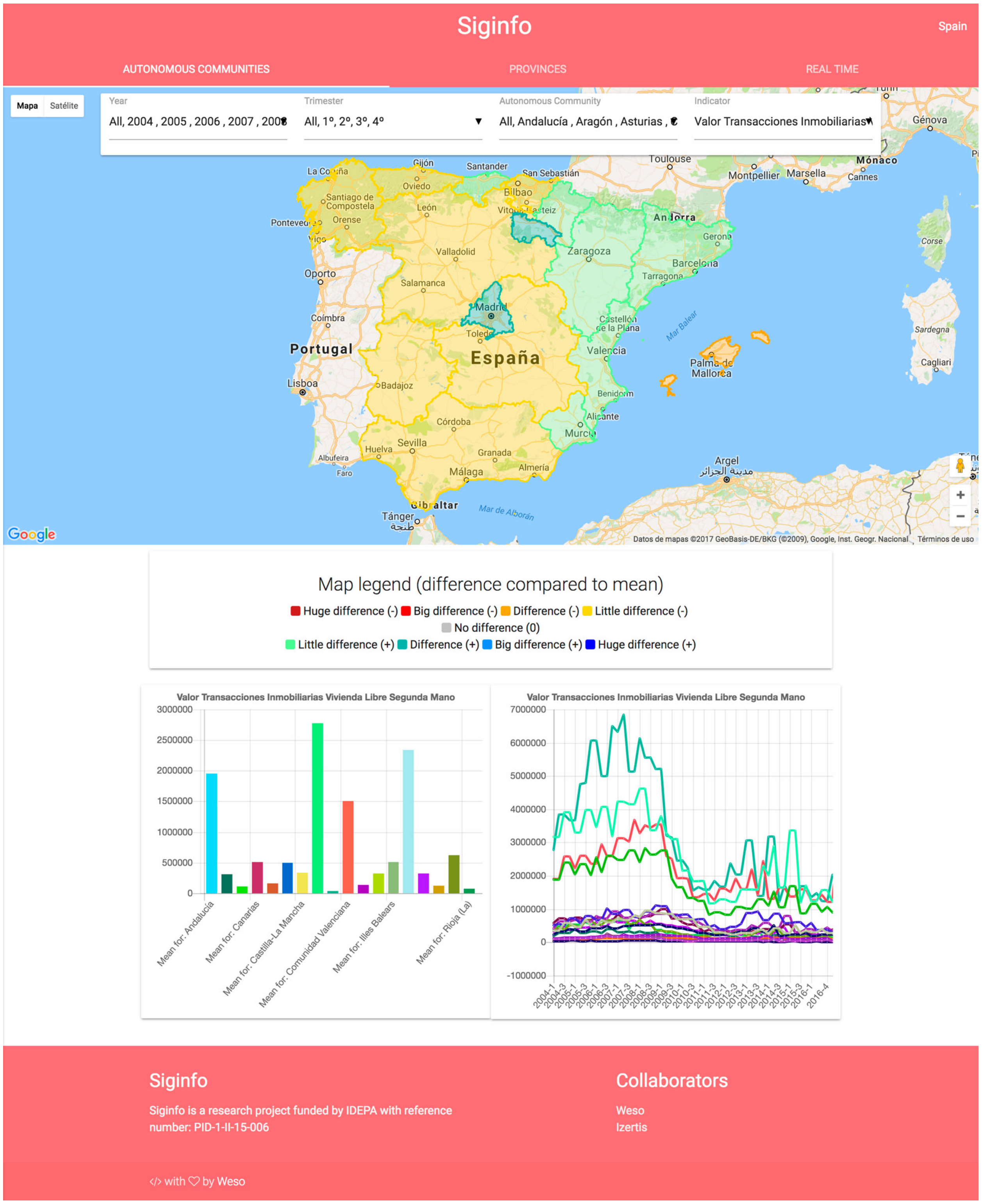


Figure 3. Diagram of the architecture with the proposed solution.

Hive data tables. Once they are persisted into Hive they can be queried using JDBC connection and SQL as the query language.

- Scala API: This API is responsible for taking the data from the Hive's JDBC interface and expose different queries in REST API format. It is also responsible for receiving the streams from the real-time layer as we will explain ahead.

Real-time layer uses data from Twitter real-time API in order to enrich the batch processed data with users interaction in this social media platform. Tweets are


Siginfo
Spain
AUTONOMOUS COMMUNITIES
PROVINCES
REAL TIME
Mapa
Satélite
Portugal
España
Google
Datos de mapas ©2017 GeoBasis-DE/BKG (©2009), Google, Inst. Geogr. Nacional
Términos de uso
Valor Transacciones Inmobiliarias Vivienda Libre Segunda Mano
Mean for: Andalucia
Mean for: Canarias
Mean for: Castilla-La Mancha
Mean for: Comunidad Valenciana
Mean for: Illes Balears
Mean for: Rioja (La)

Year: All, 2004 , 2005 , 2006 , 2007 , 2008

Trimester: All, 1º, 2º, 3º, 4º

Autonomous Community: All, Andalucía , Aragón , Asturias ,

Indicator: Valor Transacciones Inmobiliarias

Map legend (difference compared to mean)

🟥 Huge difference (-) 🟥 Big difference (-) 🟧 Difference (-) 🟨 Little difference (-)
⬜ No difference (0)
🟩 Little difference (+) 🟩 Difference (+) 🟦 Big difference (+) 🟦 Huge difference (+)

Siginfo

Siginfo is a research project funded by IDEPA with reference number: PID-1-II-15-006

</> with ♡ by Weso

Collaborators

Weso
Izertis

Figure 4. Prototype web page. Available at: http://hazelnut.weso.es/siginfo.

processed and delivered to different clients in real time as they occur.

The main components of the real time layer are:

- Python Transformer: This is responsible for bringing the data from Twitter real-time API to the event queue middleware. Transformations are made here and then transferred to the event queue.
- Apache Kafka: Kafka is a distributed event queue that aims to connect different sources and consumers of information through a distributed and fault tolerant system.
- Scala API: As we mentioned with the batch layer, Scala API will collect real-time events and expose different aggregated views to clients.

In order to show all this information to clients through the Internet, we implemented a web page built on top of HTML + CSS + JS, using Google Maps for geographical information, AJAX to dynamically change and query data from the Scala API and Server-Sent Events for sending the information from the API to the client web page (Figure 3). Figure 4 shows how mentioned web page looks like.

6. Discussion

In the previous sections, we presented the proposed architecture and a short description of Lambda architecture and Kappa architecture.

Our proposed architecture is a specialisation of Lambda architecture in the terms of: it is intended for big data, it has a batch layer and a streaming layer, and it combines real-time data with non-real-time data. However, Lambda architecture is meant, mainly, for getting information from the same source. While our one is intended, from the very beginning, to mix and integrate heterogeneous data sources. This is present not only in the existence of various data sources (see Figure 2) but also in the existence of an integrator component.

Inspiration from Kappa architecture comes in the real-time layer – or streaming layer – where data is transferred and persisted into an append-only immutable log and data is taken from there.

Kappa architecture mainly focus on real-time technologies as a way of dealing with all kind of data. This approach makes the architecture more simple but it could suffer from complexity in the data management due to the dependency on the append-only immutable log.

Moreover, while Lambda architecture excels in combining two paradigms for data processing of the same sources, Alarea brings the same foundation for heterogeneous data sources, giving developers the opportunity to decide which layer is better for their purposes. In the other hand, Kappa architecture, and its only streaming layer, could restrict some of the analytics that can be done by the batch layer. Instead, Alarea gives the option for one type of processing or for using both of them at the same time.

With this combination of both layers, the architecture copes with two kind of data and is capable of treating it no matter the timing that they present. This feature excels the flexibility and adaptability of the presented architecture.

7. Conclusion and future work

In this paper, we have presented an architecture for dealing with big data and real-time data in the real estate domain. The solution combines batch processing and real-time processing in two different layers. Although there are similar architectures to solve these problems, we stated the main differences and advantages of our solution and demonstrated with a quality model study and a prototype the usefulness of Alarea architecture.

However, future work should be encouraged to refine and improve this work. We identify the development of a more generic prototype, the industrial exploitation of the presented prototype, machine learning approaches and validation of the architecture as future lines that could be followed.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work has been funded by a collaboration with the Izertis company funded by IDEPA with reference number PID-1-II-15-006 and dossier number IDE/2015/000694.

ORCID

Herminio García-González http://orcid.org/0000-0001-5590-4857
José Emilio Labra-Gayo http://orcid.org/0000-0001-8907-5348

References

Abadi, D. J., Y. Ahmad, M. Balazinska, U. Cetintemel, M. Cherniack, J.-H. Hwang, W. Lindner, A. Maskey, A. Rasin, and E. Ryvkina. 2005. “The Design of the Borealis Stream Processing Engine.” In *Cidr*. Vol. 5, 277–289.

Abadi, D. J., D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik. 2003. "Aurora: A New Model and Architecture for Data Stream Management." *International Journal on Very Large Data Bases* 12: 120–139.

Chen, M., S. Mao, and Y. Liu. 2014. "Big Data: A Survey." *Mobile Networks and Applications* 19: 171–209.

Dean, J., and S. Ghemawat. 2008. "MapReduce: Simplified Data Processing on Large Clusters." *Communications of the ACM* 51: 107–113.

Demchenko, Y., P. Grosso, C. De Laat, and P. Membrey. 2013. "Addressing Big Data Issues in Scientific Data Infrastructure." In *2013 International Conference on Collaboration Technologies and Systems (CTS)*, 48–55. IEEE.

Dua, P. 2008. "Analysis of Consumers—Perceptions of Buying Conditions for Houses." *The Journal of Real Estate Finance and Economics* 37: 335–350.

Du, D., A. Li, and L. Zhang. 2014. "Survey on the Applications of Big Data in Chinese Real Estate Enterprise." *Procedia Computer Science* 30: 24–33.

Dujmović, J., G. De Tré, N. Singh, D. Tomasevich, and R. Yokoohji. 2013. "Soft Computing Models in Online Real Estate." *Advance Trends in Soft Computing, WCSC*, 77–91.

Foundation, A. S. 2011. Apache Hadoop. Accessed 11 May 2017. http://hadoop.apache.org/

Foundation, A. S. 2014. Spark Streaming. Accessed 11 May 2017. http://spark.apache.org/streaming/.

Foundation, A. S. 2016a. Samza. Accessed 11 May 2017. http://samza.apache.org/.

Foundation, A. S. 2016b. Storm. Accessed 11 May 2017. http://storm.apache.org/.

Foundation, A. S. 2016c. Trident. Accessed 11 May 2017. http://storm.apache.org/releases/1.0.0/Trident-API-Overview.html.

García, S., J. Luengo, and F. Herrera. 2016. "Tutorial on Practical Tips of the Most Influential Data Preprocessing Algorithms in Data Mining." *Knowledge-Based Systems* 98: 1–29.

Jin, C., G. Soydemir, and A. Tidwell. 2014. "The US Housing Market and the Pricing of Risk: Fundamental Analysis and Market Sentiment." *Journal of Real Estate Research* 36: 187–220.

Katal, A., M. Wazid, and R. Goudar. 2013. "Big Data: Issues, Challenges, Tools and Good Practices." In *2013 Sixth International Conference on Contemporary Computing (IC3)*, 404–409. IEEE.

Kreps, J. 2014. "Questioning the Lambda Architecture." Online article, July.

Laney, D. 2001. "3d Data Management: Controlling Data Volume, Velocity and Variety." *META Group Research Note* 6: 70.

Ling, D. C., A. Naranjo, and B. Scheick. 2014. "Investor Sentiment, Limits to Arbitrage and Private Market Returns." *Real Estate Economics* 42: 531–577.

Marcato, G., and A. Nanda. 2016. "Information Content and Forecasting Ability of Sentiment Indicators: Case of Real Estate Market." *Journal of Real Estate Research* 38: 165–203.

Marz, N., and J. Warren. 2015. "Big Data: Principles and Best Practices of Scalable Realtime Data Systems." Manning.

Montes, R., A. M. Sánchez, P. Villar, and F. Herrera. 2015. "A Web Tool to Support Decision Making in the Housing Market using Hesitant Fuzzy Linguistic Term Sets." *Applied Soft Computing* 35: 949–957.

Nanda, A. 2007. "Examining the NAHB/Wells Fargo Housing Market Index (HMI)." *Housing Economics*.

Perera, S., and S. Suhothayan. 2015. "Solution Patterns for Realtime Streaming Analytics." In *Proceedings of the 9th ACM International Conference on Distributed Event-Based Systems*, 247–255. ACM.

Rafiei, M. H., and H. Adeli. 2015. "A Novel Machine Learning Model for Estimation of Sale Prices of Real Estate Units." *Journal of Construction Engineering and Management* 142: 04015066.

Ramírez-Gallego, S., B. Krawczyk, S. García, M. Woźniak, and F. Herrera. 2017. "A Survey on Data Preprocessing for Data Stream Mining: Current Status and Future Directions." *Neurocomputing* 239: 39–57.

Rao, K. V., and M. A. Ali. 2015. Survey on big data and applications of real time big data analytics.

Smailović, J., M. Grčar, and N. Lavrač. 2014. "Stream-based Active Learning for Sentiment Analysis in the Financial Domain." *Information Sciences* 285: 181–203.

Spanish Ministry of Development. 2004-2016. Real Estate Transactions in Spain. Accessed 11 May 2017. http://www.fomento.gob.es/BE2/?nivel=2andorden=34000000

Terry, D., D. Goldberg, D. Nichols, and B. Oki. 1992. *Continuous Queries Over Append-only Databases*. Vol. 21. ACM.

Trawinski, B. 2013. "Evolutionary Fuzzy System Ensemble Approach to Model Real Estate Market based on Data Stream Exploration." *Journal of Universal Computer Science* 19: 539–562.

Vanhove, T., G. Van Seghbroeck, T. Wauters, B. Volckaert, and F. De Turck. 2016. "Managing the Synchronization in the Lambda Architecture for Optimized Big Data Analysis." *IEICE Transactions on Communications* 99: 297–306.

Wingerath, W., F. Gessert, S. Friedrich, and N. Ritter. 2016. "Real-time Stream Processing for Big Data." *Information Technology* 58: 186–194.

Wu, L., and E. Brynjolfsson. 2015. "The Future of Prediction: How Google Searches Foreshadow Housing Prices and Sales." In *Economic Analysis of the Digital Economy*, 89–118. University of Chicago Press.

Zaharia, M., M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica. 2012. "Resilient Distributed Datasets: A Fault-tolerant Abstraction for In-memory Cluster Computing." In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, 2–2. USENIX Association.

Zhang, L., J. Zhao, and K. Xu. 2016. "Emotion-based Social Computing Platform for Streaming Big-data: Architecture and Application." In *2016 13th International Conference on Service Systems and Service Management (ICSSSM)*, 1–6. IEEE.