Modular Architecture for Recommender Systems Applied in a Brazilian e-Commerce

Allan Vidotti Prando[1*], Solange Nice Alves de Souza[2]

[1] Instituto de Pesquisas Tecnológicas do Estado de SP, Cidade Universitária, Butantã, São Paulo, Brasil.
[2] Escola Politécnica da Universidade de São Paulo, Cidade Universitária, Butantã, São Paulo, Brasil.

* Corresponding author. Tel.: +551137674000; email: alan.prando@gmail.com
Manuscript submitted February 20, 2016; accepted March 9, 2016.
doi: 10.17706/jsw.11.9.912-923

Abstract: Over the last decade, recommender systems have been widely applied by major e-commerce websites for personalized user experience. However, few efforts have been focused so far on recommender systems architecture. In addition, Big Data technologies present opportunities to create unprecedented business advantage and better service delivery at low cost.

The recommender system architecture may vary according to the context in which e-commerce is inserted and with the adopted business settings. Consequently, from smaller to bigger companies, each recommendation system has his individual architecture with distinct implementations, but sharing similar issues. Therefore, providing a software architecture which can be easily understood, implemented and extended if necessary, would help any companies to build their own efficient recommender system, contributing to maintaining and expanding their business. In this case, is also important indicates what the technology is better tailored to each point of the architecture, considering that expertise might not exists.

Modular and extensible recommender system architecture for e-commerce is proposed here. This architecture is prepared to handle a large volume of data, responding to user actions in real time and enabling the development and testing of new approaches and recommendation technologies. All layers and components of the proposed architecture are described, including technologies to fit in these components, considering the advantages of big data and open-source possibilities. Finally, as an example, the architecture implementation in a real case scenario is shown in a Brazilian e-commerce.

Key words: Big data, data mining, machine learning, recommender systems, software architecture.

1. Introduction

The explosion of data has generated an unprecedented number of data and a number of new applications to use these data, resulting in a new reality, called Big Data [1]. These data come from blogs, photos, videos, posts in social networks and log messages from servers containing the navigation and any relevant actions of users who are visiting the web [2].

The increase in the volume of information provided the world with a wide range of products and services at different quality levels, making the decision-making process about which product to buy or which service to use an arduous task for humans. In response to this problem, Recommender Systems (RS) have emerged as a way to help people in their decision making process by providing suggestions of items that most meet a particular user [3]. Moreover, RS are a prime example of the mainstream applicability of big data, with applications such as e-commerce, social medias and music/video streaming services make use of similar techniques to mine and to process large volumes of data to better match their users' needs in a personalized fashion [4].

Considering that different items could be recommended (e.g.: films, news, retail products, advertisements) in different niches (e.g.: retail, content sales, sales of services), RS has been employed as a differential in e-commerce, such as Amazon, Ebay and Netflix. For this, RS encompass personalized algorithms that use machine learning and data mining techniques to identify the preference of each user individually [5].

Predicting the user's preference is a challenge. Typically, the data used in recommendation consist of user data (e.g.: name, age, marital status, sex), items (e.g.: name, category, details) and relationships between users and items (e.g.: click on an item, item evaluation, item purchase), forming a sparse and voluminous database [6]. Using scarce and sparse data added computational complexity to recommendation algorithms [7]. A challenge for RS is the recovery and normalization of such data (e.g.: navigation events that capture clicks within the e-commerce) to include them in the recommendation life cycle. Similarly, learning from such evidence and recommending at a response time that adds value to the e-commerce (e.g.: recommend similar items to those on which the user clicked) is also a complex task.

Currently, a common practice in an e-commerce is the construction of its own RS. This is because in the e-commerce vision, especially for large companies, recommendation is considered a differential that should not be shared with the competitors. Thus, the software architecture may vary according to the context in which e-commerce is inserted and with the adopted business settings. Consequently, although each recommendation system has his individual architecture with distinct implementations, all architectures share similar issues [3], [8], [9], [10]. In contrast, smaller companies do not have expertise and resources as larger companies do to build their own RS, increasing this disadvantage. Therefore, providing a software architecture for RS which can be easily understood, implemented and extended if necessary, would help any companies to build their own efficient RS, contributing to maintaining and expanding their business. In this case, is also important indicates what the technology is better tailored to each point of the architecture, considering that expertise might not exists.

Architecture is a representation that allows analyzing the project ability to meet its requirements, as well as reducing costs associated with software construction [11]. Reference [12] define the software architecture of a computer program or system as the system structure that covers the software components, the externally visible properties of these components and the relationships between them.

As a result, a modular RS architecture for e-commerce that could be applied in different areas is proposed. This architecture is prepared to handle a large volume of data, responding to user actions in real time and enabling the development and testing of new approaches and recommendation technologies.

This paper presents the main aspects of the RS architecture proposed, which is organized as follows. Section II introduces some RS concepts and the main aspects of related works, including techniques that should be processed by a recommender system. Section III shows the proposed architecture, detailing the components and technologies employed. Section IV introduces the results of the architecture proposed customized to a real e-commerce. Finally, section V concludes and suggests future works.

2. Concepts and Related Work

There are three main approaches to the RS area [3], [6]. [13]:

- Content-based recommendations: similar items to those that showed user preference in the past are recommended.
- Collaborative Recommendations (or collaborative filtering): items are recommended because users with similar preferences to the user, who receives the recommendation liked in the past.

- Hybrid Recommendations: combines collaborative filtering and content-based approaches.

Methods and recommendation algorithms are used in different approaches depending on the scenario or strategy adopted. In this sense, the architecture of the RS must be able to: (1) process and combine different algorithms in time of variable response and (2) try and measure new methods and algorithms quickly, meeting new demands and time-to-market [10].

2.1. Data Mining Methods in Recommender Systems

Data mining and machine learning techniques (ML) are the basis for recommendation [3], [6]. ML algorithms are used in data mining as techniques to develop hypotheses for solving problems. Based on data representing instances of the problem to be solved, ML algorithms learn to induce a hypothesis that is the solution to this problem [14].

In reference [10] Amatriain offers an overview of the main techniques and algorithms within the data mining process applied to RS, divided into three stages: (1) pre-processing, which is related to data cleaning, filtering or transformation, preparing the algorithms to run at the data analysis stage. In particular, the author addresses the sampling techniques and dimensionality reduction because of their importance and their role in the RS area; (2) Data Analysis, considered the main stage, because at this stage algorithms (mainly ML classifiers and clusters) are used to find items to recommend; and (3) Interpretation of results, using data obtained at the data analysis stage to deliver business value. The article also presents the main algorithms used in RS.

The data mining field has considerably advanced in recent years, due to technological advances that provided the processing and storage of large volume and variety of data. In particular, social networks are considered essential to this change, driving the creation of such data generated by different users. Thus, individual efforts regarding techniques for extracting information on specific data types, such as text mining, become relevant for achieving results [15], [16].

According references [10], [17], RS architecture should allow several ways to improve the understanding about users and items in the recommendation process, using mostly data mining and ML techniques. As a result, the RS should provide an evaluation method to identify the best approach to achieve more comprehensive models and add better recommendation capabilities to the e-commerce.

2.2. Overview of Recommender Systems Architecture in Real Case Scenarios

Despite larger companies usually do not share their RS architecture, Netflix, Ebay and Amazon has already contributed their experience, as highlighted bellow.

2.2.1. Netflix

The architecture is divided into three layers: Online, Nearline and Offline. Each layer has a processing module, with different responsibilities and characteristics, focusing on the delivery of the recommendation, always taking into consideration the actions of the logged user.

The online layer addresses recent user events, delivering real-time response, restricting the computational complexity of the algorithms and the volume of data that can be processed. The layer offline has no limitation on the amount of data and computational complexity, but has a flexible response time and the results can become out-dated constantly due to the data frequent renewal. The Nearline layer is the middle layer, providing a computation similar to online, but with a longer response time.

In reference [17] Amatriain points out as one of the architectural challenges is how to combine and to manage online and offline computing with cohesion. All data collected is stored for offline processing in the layer, but some are also used for real-time processing for online layer. Most computer processing is consumed by ML algorithms that can be processed by the offline layer with training algorithms scheduled for consumption of the other layers. In this sense, some components and technologies play important roles.

- Event Distribution: handles all e-commerce events, such as clicks, views, evaluations, etc.
- Model Training: computing that uses existing data to generate models (ML algorithms trained) to be used in other computational processing layers. Although the models are trained on the offline layer, there are ML techniques online for incremental models in real time.
- Models: are models generated that will be used in other layers.
- Query Results: refines the computed data using Apache Pig and Apache Hive inside Apache Hadoop. Once the queries are executed, Apache Kafka is used to trigger the mechanisms for using the data.
- Batch computing of intermediate or end results: offline computing of the results defines what would be used for subsequent online processing or for presenting the user.
- Databases: Hybrid Storage with MySQL for structured data and Apache Cassandra for solutions that require reading and writing on a large scale.

Fig. 1 illustrates the architecture of the Netflix recommendation system [3]. This architecture is prepared to meet the needs of Netflix, to the particularities of a streaming video system that has thousands of users round the world. For the volume it serves, the system is extremely complex, impractical for small e-commerce.

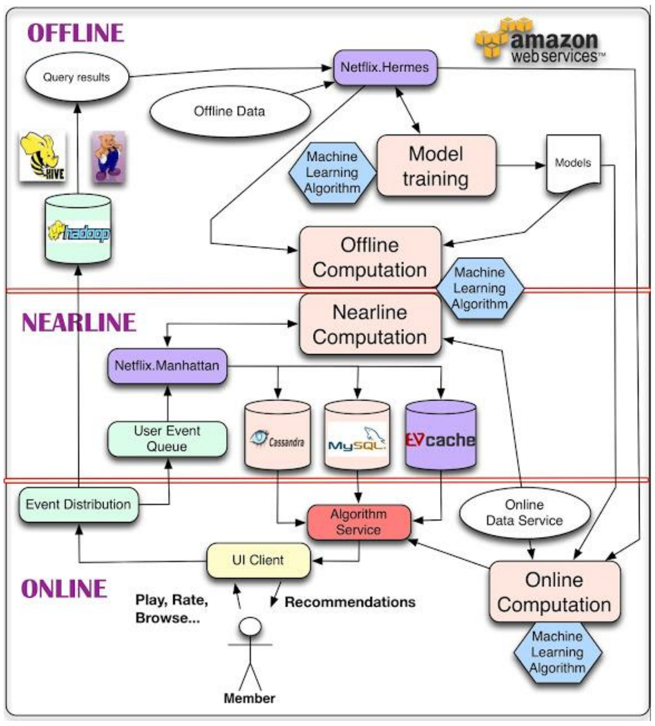

Fig. 1. Netflix RS architecture.

2.2.2. Ebay

The architecture of the RS used by Ebay, shows a scalable system that recommends items with a short life span (e.g.: auctions) and provides control over the trade-off between relevance and quality [18].

The solution has two layers: Offline Model Generation and Real-time Performance System. The offline layer processes models using clustering ML algorithms. The real-time layer combines the cluster model with dynamic characteristics obtained by the user section in e-commerce.

The authors emphasize two main approach scenarios: pre-order recommendation and after-purchase recommendation. In the pre-purchase, RS recommends good alternatives to recently viewed items. In the post-purchase, RS recommends complementary items and/or related to an item that the user has recently acquired.

The database also acts as a bond for the layers, since both use the stored data. In addition to basic information such as users, items, and user actions (navigation, access to auctions, etc.), the base also stores the outputs of the clusters, such as which group a set of similar items belong.

The Real-time layer has two components: Similar Item Recommender (SIR) and Related Item Recommender (RIR). Both receive an item as input and return a set of similar or related items in return. As the response must occur in real time, all the computational complexity is in the offline layer, consisting of Apache Hadoop running map reduce jobs, queries and K-means algorithm.

To achieve eBay requirements, the architecture in Figure 3 uses only clustering ML methods, limiting itself to present better performance to short-time life items. Therefore, it is difficult to extend this architecture to ordinary e-commerce.

2.2.3. Amazon

Reference [19] present the item-to-item collaborative filtering used by Amazon in 2003. At that time, Amazon.com extensively uses recommendation algorithms to personalize its Web site to each customer's interests. Amazon developed their own architecture focused on a core algorithm, named item-to-item collaborative filtering, scaling to massive data sets and producing high-quality recommendations in real time.

The algorithm matches each of the user's purchased and rated items to similar items, then combines those similar items into a recommendation list. To determine the most similar match for a given item, the algorithm builds a similar items table by finding items that customers tend to purchase together. Given a similar items table, the algorithm finds items similar to each of the user's purchases and ratings, aggregates those items, and then recommends the most popular or correlated items.

The key to architecture scalability and performance is that it produces the similar items table offline. The algorithm of online component scales based on how many titles the user has purchased or rated, regardless of the catalogue size or the total number of customers. Therefore it is fast even for extremely large data sets. Reference [19] claims that recommendation quality is excellent because the algorithm recommends highly correlated similar items.

Since there are no recent papers about Amazon RS architecture, it is difficult to analyze how extensible it is. However, Amazon.com e-commerce is known to be one of the most personalized websites on the world.

3. Proposed System Architecture

As described in Section 2, a common practice in e-commerce is the construction of its own RS focused on his own business [17], [18], [19]. Consequently, although each RS has his individual architecture with distinct implementations, all architectures share similar issues. Also it increases the disadvantage between smaller companies and e-commerce leaders, since smaller companies do not have expertise and resources to build their own RS. Therefore, a modular architecture which can be easily implemented and extended depending on e-commerce needs would help any company to build their own efficient RS, contributing to maintaining and expanding their business.

As a result, the proposed architecture is modular, extensible and intends to be easily adaptable to e-commerce niche. The RS proposed architecture is shown in Fig. 2.

To begin with, this architecture is ready to handle a large volume of data, respond to user interactions, process and deliver recommendations in real time. This architecture is modular, allowing the use of different technologies and platforms on each component, thus facilitating the use of technology that has the best implementation to solve the problem regardless of e-commerce size.

One of the key points of the architecture is how to combine and manage online and offline computation in a seamless manner. As a consequence, the architecture is divided into three main layers. Each layer has its responsibility, with components to perform different roles. Each component can have one or more technologies, allowing the architecture to be extended according to the complexity of the problem.

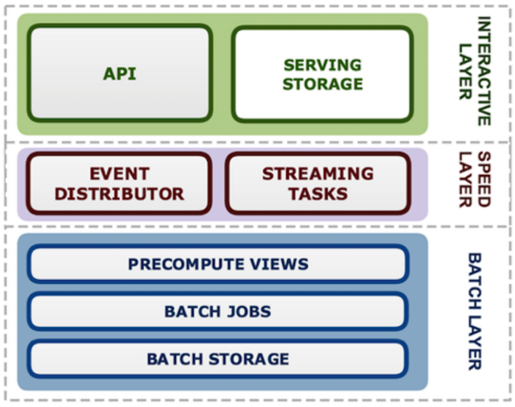

Fig. 2. Proposed architecture.

3.1. Layers and Components

Interactive Layer is responsible for receiving, interpreting and forwarding e-commerce requests. It has API (Application Programming Interface) and Serving Store module.

- API: is the interface between the e-commerce and the RS. All the requests are made in this module, which has high availability as its main requirement.
- Serving Storage: stores all the recommendations processed by Streaming Tasks and Batch Jobs module.

Speed layer processes real time recommendation. Therefore, it does not perform ML training algorithms, but it uses the trained models and pre-processed data to perform the recommendation. It has Event Distributor and Streaming Tasks modules.

- Event Distributor: is a service able to decide whether the request should be handled by speed or batch layers, forwarding the message to the correct recipient.
- Streaming Tasks: processes a task in real time. The main task of this module is to process the recommendation trained using templates and pre-processed data by batch layer.

Batch layer: tasks are performed with a long response time. It has the precomputed Views, Batch Jobs and Batch Storage modules.

- Precompute Views: provides data to be consumed by the speed layer. Among these data, there are trained ML models and the prepared information used for recommendation. The storage is optimized for reading and searching.
- Batch Jobs: processes demands on tasks such as ML training algorithms, calculates product similarity and pre-processing data (products and users). The result is stored in precomputed views module.

Batch Storage: stores the recovered raw data of e-commerce (user, product and tracking). Storage is optimized for writing.

3.2. Technologies

Big Data technologies have been created to handle massive data sets and provide scalability for data analysis. Also most of these technologies are open-source and can be used for a low cost. To evaluate this scenario, the proposed architecture investigates the performance of some Big Data technologies in each layer as following (Fig. 3).

The technology mapped to the architecture proposed:

- API: Java EE (Servlets or frameworks such as Spring and Playframework), Spray (http://spray.io/), NodeJS and .NET.
- Serving store: Redis and MongoDB.
- Event distributor: Apache Kafka and RabbitMQ.
- Streaming tasks and batch jobs: Apache Hadoop (YARN / HDFS), Apache Mahout, Apache Spark, Apache Hive.
- Batch storage and precompute views: Apache Cassandra and Apache HBase.
- Possible programming languages: Java, Scala, Python, R and C#.

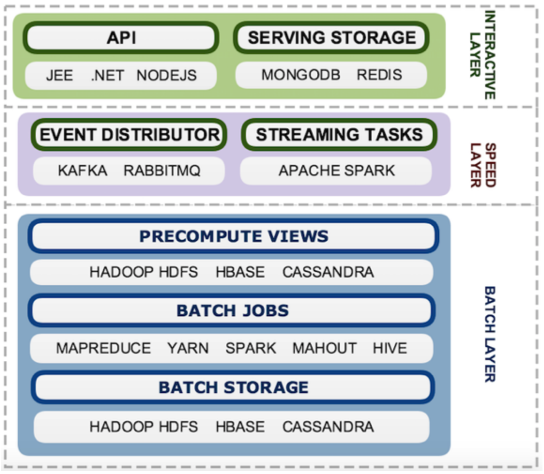

Fig. 3. Map of suggested technologies.

The technologies of streaming tasks, batch jobs, batch storage and precompute views are the RS architecture key points. The Apache Hadoop, open-source technology reported by several authors as the main piece of a big data environment that can be employed in these modules [4], [20], [7].

Apache Hadoop was created by the Yahoo Company in 2006 and maintained by the Apache Software Foundation, the main objective behind its development was to provide a fault tolerant and highly available scalable parallel computing environment. Hadoop allows 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 of them offering local computation and storage [7].

Apache Hadoop is divided into 4 modules:

- Hadoop Common: Bundle of utilities, and programs that support the other modules.
- Hadoop Distributed File System (HDFS): Distributed File System for data storage.
- Hadoop YARN: Platform that manages cluster resources and performs processing tasks.
- Hadoop MapReduce: implements the MapReduce paradigm parallel programming proposed by Google [21].

Many projects have been and are being developed around the Hadoop, causing it to be considered a complete ecosystem for data processing and storage [4], [20], [7]:

- Apache Hive enables queries through a similar SQL (Structured Query Language) to the HDFS.
- Apache HBase, which provides a database structure NoSQL (not only SQL) using HDFS as storage, facilitating reading and writing in real time.
- Apache Mahout, a scalable machine learning API and data mining that has algorithms implemented in the MapReduce paradigm prepared to run at YARN.

In the proposed architecture, Hadoop can be used for the recommendation processing by the ML algorithm implementations using MapReduce paradigm (Apache Mahout) and other paradigms (Apache Spark) executed in YARN. The architecture also enables the development of specific jobs using Java and R, as both are compatible and can be run on a Hadoop cluster (Java is the native Hadoop language).

Apache Kafka and RabbitMQ technologies stand out as a distributed messaging system based on the publish-subscribe model capable of playing as an event distributor.

The NoSQL databases MongoDB and Redis can store the recommendation in serving store. These technologies are highlighted because they have fast reading and writing. The API can be developed with any technology, language or platform that meets the HTTP requests. The JavaEE and .NET technologies stand out as the most used while Spray and NodeJS stands out for its performance.

The languages listed can be used to develop jobs inside Hadoop (Java), Mahout (Java), Spark (Scala, Java, R) and implement statistical functions and ML (R, Python).

Although RS meet the requirements proposed, all items can be replaced without affecting the other layers of the architecture, reinforcing the modularity importance

3.3. Data

An important point about the RS architecture is the logical structure of the data. This work is intended to build a RS that fits an e-commerce of many niches (movies, sports, music, etc.). Fig. 4 shows the RS Entity-Relationship diagram, comprising a data set with seven main entities: (1) user, storing information such as name, sex, date of birth and social networking access keys; (2) department, used to group products in a hierarchical fashion (3) product, including basic and detailed features, which vary by category; (4) rating, performed explicitly by a user on a product; (5) action, performed implicitly by a user on a product; (6) Store and (7) Recommendation, the list of personalized products for an user.

Serving storage stores entity "recommendation" in order to provide information to the e-commerce in real time. All other entities are stored in batch storage, providing information to all processing of recommendation flow through the module precompute views.

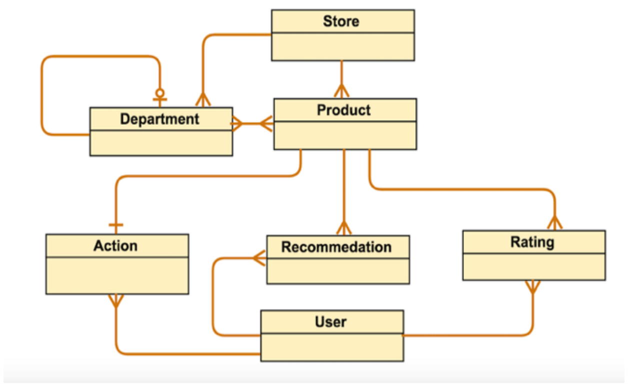

Fig. 4. Entity relationship diagram.

3.4. Activity Diagram

In the simplest formulation, the RS have to perform two major tasks: (1) get user data to achieve the recommendation and (2) process and deliver recommendations to an e-commerce user (Fig. 5a). As a consequence, the RS architecture must be able to save all user interactions as well as to provide recommendations in real time (Fig. 5b).

4. Experimental Results

To demonstrate how the proposed architecture works, it was extended and implemented on Brazilian e-commerce, which is the largest retail store of sporting items in Latin America [3]. The company aims to bring the new concept of sports megastore to Brazil. As with nearly 180 physical stores throughout the country, the Company Online has a huge variety of products such as shoes, accessories and sports equipment and casual fashion for men, women and children.

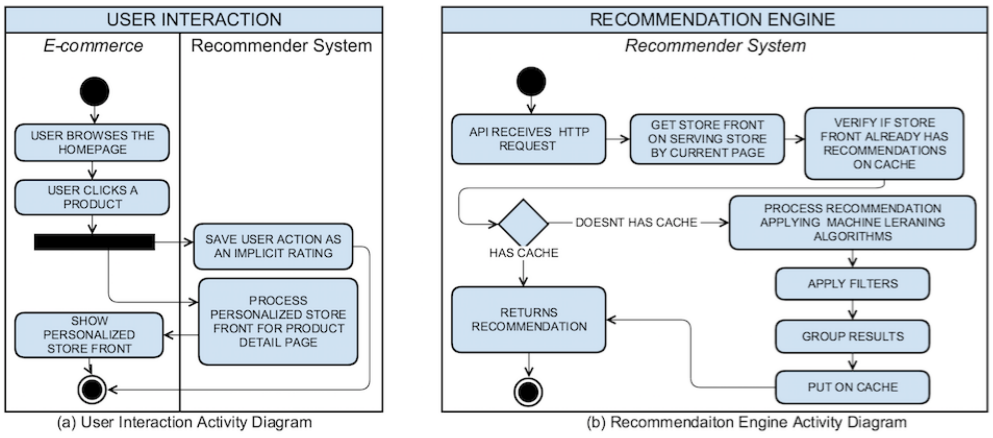
USER INTERACTION
E-commerce
Recommender System
USER BROWSES THE HOMEPAGE
USER CLICKS A PRODUCT
SAVE USER ACTION AS AN IMPLICIT RATING
PROCESS PERSONALIZED STORE FRONT FOR PRODUCT DETAIL PAGE
SHOW PERSONALIZED STORE FRONT
(a) User Interaction Activity Diagram
RECOMMENDATION ENGINE
Recommender System
API RECEIVES HTTP REQUEST
GET STORE FRONT ON SERVING STORE BY CURRENT PAGE
VERIFY IF STORE FRONT ALREADY HAS RECOMMENDATIONS ON CACHE
DOESNT HAS CACHE
PROCESS RECOMMENDATION APPLYING MACHINE LERANING ALGORITHMS
HAS CACHE
APPLY FILTERS
RETURNS RECOMMENDATION
GROUP RESULTS
PUT ON CACHE
(b) Recommendaiton Engine Activity Diagram

Fig. 5. User interaction and recommendation engine activity diagrams.

The Company Online has 15.5 million user sessions per month (7 million unique users) and 3.5 million pageviews per day (70 pageviews per second in the rush hour). To deal with it, as mentioned in section 3, the architecture must use technologies capable of storing and processing large volume of data.

The Company chose Playframework to be the API and Event Distributor. Redis and Mongo are the serving storage and Apache Hadoop is the Batch Jobs and Streaming Tasks. Hbase is the Batch Storage, storing all user actions in its e-commerce. These technologies were selected because each of them supplies a demand of RS architecture on Brazilian Company's scenario. Besides as Big Data technologies, all of them are open-source and can scale as much as needed.

In four months, 2.8TB was stored inside HBase with all user actions (clicks, carts, orders, etc.). This data, tracked by API, is used to generate the recommendation mainly using Apache Mahout and Apache Spark - frameworks running on Apache Hadoop – in addition to jobs map reduce.   These recommendations are generated using all sorts of machine learning approaches, from unsupervised methods such as k-means to supervised classifiers such as k-Nearest Neighborhood, Support Vector Machines and so on.

Initially, the Hadoop cluster was configured carefully, analyzing how the data had to be compressed inside HBase and also tuning YARN slaves to compute as fast as possible. After that, Redis database was modeled to store all the recommendations processed by Spark and Mahout, ensuring that when the API requested it, the answer would be in real time.

The first obstacle was Hbase schema modeling. Since it is a wide column NoSQL database with some particularities, it needed extra efforts and tries to achieve an acceptable performance. Also, prepare the tracked data to be used on ML algorithms was a challenge, considering the variety of products and the cold-start problem, caused by missing of ratings in the first days.

Secondly, the development of map reduce and spark jobs to process data in real time (e.g.: recommend products with discount to users who recently showed interest on similar products) was a challenge. The data must be available few seconds after user click, and the job must discover knowledge from data as much as possible to reach the user on the right time.

Finally, the establishment of a workflow to reuse and combine algorithms was important to ensure RS performance. Sometimes, to fit a recommendation strategy the architecture must incorporate results to deliver the best suggestion. Apache Hadoop was capable of manage this request without shake RS performance.

Table 1. Experimental Results

Type	Description	Score
Open Rate	Rate of total open mail divided by total mail sent	42%
CTR	Rate of total clicks divided by total mail sent	31%
CTOR	Rate of total unique clicks divided by total mail sent	19%
Conversion	Rate of total of clicks with orders after 30 minutes	4%

The Brazilian Company RS still in a test phase and so far is performing as expected. Table 1 presents results from the recommendation by mail, with 42% of open rate, 19% of CTR (click through rate - measure the effectiveness of both the subject and email copy), 31% of CTOR (click to open rate - measure focused only on email success in driving the user to click) and 4% of conversion (orders made 30 minutes after a click on an email). Using Hadoop Ecosystem as the core component and Redis storing all recommendations, the RS was able to generate and deliver all recommendations. It was not possible to compare with previews results because the recommendation by email is offering for the first time.

5. Conclusion

This paper showed how RS architectures of Ebay, Amazon and Netflix e-commerce are mature and since each one creates its own architecture with some particularities, it is difficult to extend their architecture to create a RS.

Therefore, a modular and extensible architecture for RS was proposed that could be used for recommendation in e-commerce of different areas and sizes. The architecture presented here is part of a research that is being conducted, in which social networks is applied as a source of data to help recommendation process. As a consequence, it is ready to handle a large volume of data, responding to user actions in real time using Big Data technologies. The issues presented herein are expected to contribute to advancing the discussion about the next generation of RS architecture in the RS community.

There are many future important challenges in RS architecture that arise from the nature of personalization: discovering user preferences by processing a sparse, diverse and large dataset using techniques that demand a high computational capacity [4]. Researchers and developers will have to deal with some challenges in the next years.

- Social Recommendation: Social networks can obviously be a dataset to RS. However, how add this dataset to a recommendation life cycle is the challenge [22]. To do so, the RS have to have permission to obtain user social data and use these data to personalize one's life [23].
- Early Adopter Architecture: as far as hardware capacity continues to grow, so do the possibilities of new recommendation approaches. The software architecture has to use such capacity to support rapid innovation, applying all data and algorithms available to create the best possible experience.
- Dealing with images, videos and audios: users and companies are using all kinds of data to communicate, such as photos uploaded on users social networks or videos on any website, such as YouTube. RS architecture must be able to read and to extract value from this data to better understand what the user preferences are.

Big Data is going to continue growing over the next years, and more and larger sources of data will appear. As personalization algorithms keep improving and data keep growing, the RS architecture must improve together, with opportunities and lessons to be learned.

References

[1] Singh, S., & Singh, N. (2012). Big data analytics. *Proceedings of International Conference on Communication, Information and Computing Techonolgy* (pp. 1-4).

[2] Landim, T., Souza, S. N. A., & De Souza, L. S. (2013). Improving a WebSite with web analytics – A case study. *Proceedings of Iberian Conference on Information Systems and Technologies* (pp. 423-427). Lisbon, Portugal.

[3] Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, *17(6)*, 734-749.

[4] Fan, W., & Bifet, A. (2012). Mining big data: current status, and forecast to the future. *ACM SIGKDD Explorations Newsletter*, *14(2)*, 1-5. New York, USA.

[5] Yand, S. H., Long, B., Smola, A. J., Zha, H., & Zheng, Z. (2011). Collaborative competitive filtering: Learning recommender using context of user choice. *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 295-304). New York, USA.

[6] Ricci, F., Rokach, L., & Bracha, S. (2011). Introduction to recommender systems handbook. *Recommender Systems Handbook*.

[7] Rios, L. G., & Diguez, J. A. I. (2014). Big data infraestructure for analyzing data generated by Wireless Sensor Networks. *Proceedings of IEEE International Congress in Big Data* (pp. 816-823).

[8] Zhang, Y., & Pennacchiotti, M. (2013). Predicting purchase behaviors from social media. *Proceedings of the 22nd International Conference on World Wide Web* (pp.1521-2532).

[9] Zhang, Y., & Pennacchiotti, M. (2013). Recommending branded products from social media. *Proceedings of the 7th ACM Conference on Recommender Systems* (pp. 77-84).

[10] Amatriain, X. (2012). Mining large streams of user data for personalized recommendations. *ACM SIGKDD Explorations Newsletter*, *14(2)*, 37-48. New York, USA.

[11] Pressman, R. S. (2010). *Software Engineering: A Practitioner's Approach*. Palgrave Macmillan.

[12] Bass, L. (2007). *Software Architecture in Practice.* Pearson Education India.

[13] Burke, R (2007). Hybrid web recommender systems, *The Adaptive Web: Methods and Strategies of Web Personalization.* (pp.377-408), Springer Berlin Heidelberg.

[14] Kononenko, I., & Kukar, M. (2007). *Machine Learning and Data Mining: Introduction to Principles and algorithms*. Horwood Publishing Limited.

[15] Aggarwal, C., & Zhai, C. (2012). An Introduction to Text Mining. In: Aggarwal, C. & Zhai, C. (Eds.), *Mining Text Data*. (pp.1-10). Springer.

[16] Hu, X., & Liu, H. (2012). Text Analytics in Social Media. In: Aggarwal, C. & Zhai, C. (Eds.), *Mining Text Data*. (pp.385-414), Springer.

[17] Amatriain, X. (2013). Big & Personal: data and models behind Netflix recommendations. In ACM New York (Eds.), *Proceedings of the 2 International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications* (pp. 1-6). New York, USA.

[18] Katukuri, J., Mukherjee, R., & Konik T. (2013). Large-scale recommendations in a dynamic marketplace. *Proceedings of Workshop on Large Scale Recommendation Systems at RecSys*. Hong Kong, China.

[19] Linden, G., Smith, B., & York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing* (pp. 76-80).

[20] Lin, J., & Ryaboy, D. (2012). Scaling big data mining infrastructure: The twitter experience. *ACM SIGKDD Explorations Newsletter*, *14(2)*, 6-19.

[21] Dean, J., & Ghemawat, S. (2008). MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, *51(1)*, 107-113. New York, USA.

[22] Oliveira, A. D. R., Bessa, L. N., Andrade, T. R., Filgueiras, L. V. L., & Sichman, J. S. (2012). Trust-based

recommendation for the social Web. *IEEE Latin America Transactions*, *10(2)*, 1661-1666.
[23] Quirino, G. Z., Mals, N. P., Groterhorst, V. M., Souza, S. N. Al., & DeSouza, L. S. (2015). Meneduca – Social school network to support the educational environment. *Proceedings of XLI Latin American Computing Conference* (pp. 1-6).

Alan Vidotti Prando is graduated in information systems at Centro Universitário Fundação Santo André. He is a final year master of software engineering student at Instituto de Pesquisas Tecnológicas do Estado de SP (IPT), São Paulo, Brazil. His research interests are recommender systems, big data, data mining, machine learning and social media. He has several years of working experience in software architecture and big data.

Solange Alves Nice de Souza is graduated in physics at Fluminense Federal University. She holds a M.Sc. in nuclear engineering from Federal University of Rio de Janeiro and Ph.D. in electric engineering, database concentration area from University of São Paulo (USP). Dr. Souza is a professor of Polytechnic School of USP since 2003. Her research interest includes big data, data management, object-relational databases and database modeling.