# Appraising SPARK on Large-Scale Social Media Analysis

**Chapter** · February 2018

DOI: 10.1007/978-3-319-75178-8_39

**4 authors**, including:

L. Belcastro
Università della Calabria
**18** PUBLICATIONS **56** CITATIONS
SEE PROFILE

Fabrizio Marozzo
Università della Calabria
**60** PUBLICATIONS **483** CITATIONS
SEE PROFILE

Domenico Talia
Università della Calabria
**407** PUBLICATIONS **5,113** CITATIONS
SEE PROFILE

# Appraising SPARK on Large-Scale Social Media Analysis

Loris Belcastro, Fabrizio Marozzo✉, Domenico Talia, and Paolo Trunfio

DIMES, University of Calabria, Italy,
[lbelcastro, fmarozzo, talia, trunfio]@dimes.unical.it

**Abstract.** Software systems for social media analysis provide algorithms and tools for extracting useful knowledge from user-generated social media data. ParSoDA (Parallel Social Data Analytics) is a Java library for developing parallel data analysis applications based on the extraction of useful knowledge from social media data. This library aims at reducing the programming skills necessary to implement scalable social data analysis applications. This work describes how the ParSoDA library has been extended to execute applications on Apache Spark. Using a cluster of 12 workers, the Spark version of the library reduces the execution time of two case study applications exploiting social media data up to 42%, compared to the Hadoop version of the library.

**Keywords:** Social Data analysis, Scalability, Spark, Cloud computing, Parallel library, Big Data.

## 1 Introduction

Every day, huge volumes of data are generated by users of social networks like Facebook, Twitter, Instagram and Flickr. Social media analysis aims at extracting useful knowledge from this big amount of data [3]. Social media analysis tools and algorithms have been used for the analysis of collective sentiments [15], for understanding the behavior of groups of people [6][5] or the dynamics of public opinion [2]. The use of parallel and distributed data analysis techniques and frameworks (e.g. MapReduce [10]) is essential to cope with the size and complexity of social media data. However, it is hard for many users to use such frameworks, mainly due to the programming skills necessary to implement the desired data analysis methods on top of them [18].

ParSoDA (Parallel Social Data Analytics) is a Java library for building parallel social media analysis applications, designed for simplifying the programming task necessary to implement these class of applications on parallel computing systems. To reach this goal, ParSoDA includes functions that are widely used for processing and analyzing data gathered from social media for finding different types of information (e.g., user mobility, user sentiments, topics trends). ParSoDA defines a general framework for a social data analysis application that includes a number of steps (data acquisition, filtering, mapping, partitioning, reduction, analysis, and visualization), and provides a predefined (but extensible) set of functions for each step. Thus, an application developed with ParSoDA is expressed by a concise code that specifies the functions invoked at each step. The library includes algorithms that are widely used on social media data for extracting different types of information. In a previous work [4], we presented the main features of ParSoDA and described how it can be used to execute parallel social data analysis on a Cloud system exploiting Apache Hadoop [19]. In this work we describe how the ParSoDA library has been extended to execute applications on Apache Spark [22]. Spark is one of the most popular framework for Big Data processing. Differently from Hadoop, in which intermediate data are always stored in distributed file systems, Spark stores data in main memory and processes it repeatedly so as to obtain better performance for some classes of applications (e.g., iterative machine learning algorithms and queries on data [20]).

We experimentally evaluated the scalability of the Spark version of ParSoDA proposed in this paper, compared to the previous Hadoop version of the library that has been presented in [4]. The experimental evaluation is based on two case study applications on social media data published in Flickr and Twitter. The first application aims at discovering sequential patterns from user movements, so as to find the common routes followed by users. The second application discovers the frequent sets of places visited by users. The ParSoDA library performance has been evaluated carrying out the data analysis applications both on a Hadoop and a Spark cluster deployed on the Microsoft Azure cloud platform. On a cluster using 12 workers, the Spark version of ParSoDA reduced the execution time up to 42% compared to the Hadoop version of the library.

The remainder of the paper is organized as follows. Section 2 discusses related work. Section 3 describes the ParSoDA library and the proposed integration with Spark. Section 4 presents the experimental evaluation of two case studies. Finally, Section 5 concludes the paper.

## 2 Related work

Many professionals and researches are working on the design and implementation of tools and algorithms for extracting useful information from data gathered from social networks. In most cases the amount of data to be analyzed is so big that high performance computers, such as many and multi-core systems, Clouds, and multi-clusters, paired with parallel and distributed algorithms, are used by data analysts to reduce response time to a reasonable value [3].

Several research activities consider not only data analysis, but also providing solutions for building social data applications, with the aim of helping scientists to develop the different steps that compose social data mining applications without the need to implement common operations from scratch.

SOCLE [1] is a framework for expressing and optimizing data preparation in social applications. It is composed by a general-purpose three-layers architecture, an algebra, and a language to define operations for data preparation in social applications. As an example, SOCLE provides operators to remove all unnecessary information from data (data pruning), to add information by using external sources (data enrichment), to transform data values (data normalization). The authors examined the use of SOCLE for manipulating social data in two families of social applications, recommendation and analytics, but no studies have been performed to assess its scalability, and no details about framework requirements have been provided.

Cuesta et al. [9] proposed a framework for easing Twitter data extraction and analysis. In the proposed architecture the tweets, mined by the application through the Twitter APIs, are cleaned and then stored in a MongoDB database [7]. In addition to basic database operations (i.e. selection, projection, insertion, updating and deletion), the framework can be extended creating more complex aggregation MapReduce tasks in Python. By default, the framework provides researchers modules for executing sentiment analysis and generating reports.

SODATO (SOcial Data Analytics Tool) [12] is an on-line tool for helping researches on social data. It utilizes the APIs provided by social networks (i.e., currently, it supports only Facebook and Twitter) for collecting data; then, it provides a combination of web as well as console applications that run in batches for preprocessing and aggregating data for analysis. At the end of the analytics process, the results can be displayed using the integrated visualization module. SODATO provides methods for several kinds of analysis, such as sentiments analysis, keyword analysis, content performance analysis, social influencer analysis, etc.

You et al. [21] presented a framework, running on Clouds, for developing social data analysis applications for smarter cities, especially designed to support smart mobility. In particular, the framework is composed by five components (i.e., data collector, data preprocessor, data analyzer, data presenter, and data storage) that cover the whole data analysis lifecycle. The framework supports data collection from social networks (e.g., Twitter, Foursquare), by exploiting their public APIs, and from other Internet sources (e.g. website, blog, files). A component devoted to data preprocessing provides functions for data cleansing, filtering and normalization. Afterwards, the data analyzer component provides needed analysis methods (e.g. K-means, DBScan, and Self-organizing Map) to make some data analysis.

The main differences between ParSoDA and the systems described above (but the one by You et al. [21]), is that our system was specifically designed to build Cloud-based data analytics applications. To this end, it provides scalability mechanisms based on two of the most popular parallel processing frameworks (Hadoop and Spark), which are fundamental to provide satisfactory services as the amount of data to be managed grows.

## 3 The ParSoDA library

ParSoDA (Parallel Social Data Analytics) is a Java library that includes algorithms that are widely used to process and analyze data gathered from social networks for extracting different kinds of information (e.g., user mobility, user sentiments, topic trends).

ParSoDA defines a general structure for a social data analysis application that is formed by the following steps:

– *Data acquisition*: during this step, it is possible to run multiple crawlers in parallel; the collected social media items are stored on a distributed file system (HDFS [17]).
– *Data filtering*: this step filters the social media items according to a set of filtering functions.
– *Data mapping*: this step transforms the information contained in each social media item by applying a set of map functions.
– *Data partitioning*: during this step, data is partitioned into shards by a primary key and then sorted by a secondary key.
– *Data reduction*: this step aggregates all the data contained in a shard according to the provided reduce function.
– *Data analysis*: this step analyzes data using a given data analysis function to extract the knowledge of interest.
– *Data visualization*: at this final step, a visualization function is applied on the data analysis results to present them in the desired format.

For each of these steps ParSoDA provides a predefined set of functions. Users are free to extend this set with their own functions. For example, for the data acquisition step, ParSoDA provides crawling functions for gathering data from some of the most popular social networks (Twitter and Flickr), while for the data filtering step, ParSoDA provides functions for filtering geotagged items based on their position, time of publication, and contained keywords.

Figure 1 presents the reference architectures describing how user applications based on the ParSoDA library are executed on the Hadoop and Spark frameworks, which allows implementing parallel and distributed applications with high level of scalability for several data mining tasks [8] [22]. As shown in the figure, user applications can make use of ParSoDA and other libraries. Applications can be executed on a Hadoop or a Spark cluster, using YARN as resource manager and HDFS as distributed file system.

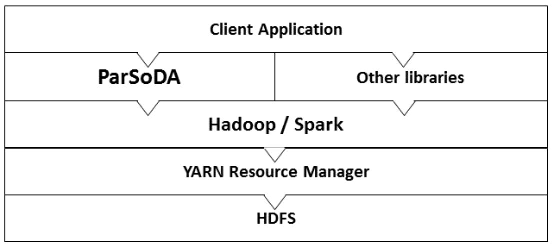
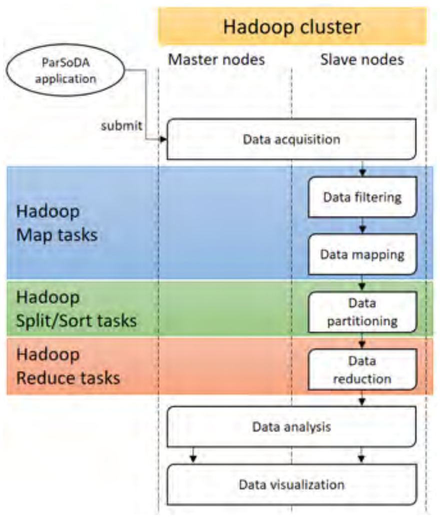


**Fig. 1.** Reference architecture

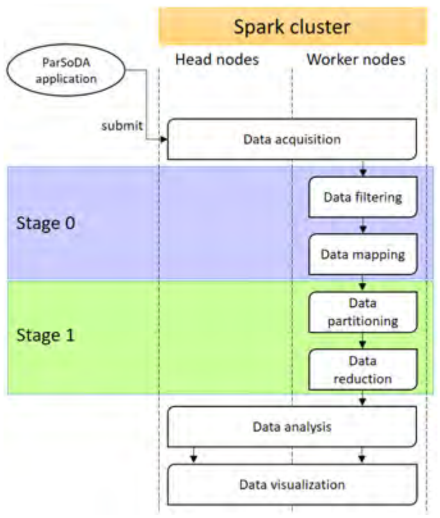
Figure 2 provides details on how applications are executed on a Hadoop or a Spark cluster. The cluster is formed by one or more master nodes, and multiple slave nodes. Once a user application is submitted to the cluster, its steps are executed according to their order (i.e., data acquisition, data filtering, etc.).

On a Hadoop cluster (see Figure 2(a)), some steps are inherently MapReduce-based, namely: data filtering, data mapping, data partitioning and data reduction. This means that all the functions used to perform these steps are executed within a MapReduce job that runs on a set of slave nodes. Specifically: the data filtering and data mapping steps are wrapped within Hadoop Map tasks; the data partitioning step corresponds to Hadoop Split and Sort tasks; the data reduction step is executed as a Hadoop Reduce task. The remaining steps (data acquisition, data analysis, and data visualization) are not necessarily MapReduce-based. This means that the functions associated to these steps could be executed in parallel on multiple slave nodes, or alternatively they could be executed locally by the master node(s). The latter case does not imply that execution is sequential, because a master node could make use of some other parallel runtime (e.g., MPI).

On a Spark cluster (see Figure 2(b)), the main steps are executed within two Spark stages that run on a set of worker nodes. A stage is a set of independent tasks executing functions that do not need to perform data shuffling (e.g., transformation and action functions). Specifically: data filtering and mapping are executed within the first stage (*Stage 0*), while data partitioning and reduction are executed within the second stage (*Stage 1*). Concerning the remaining steps (data acquisition, data analysis, and data visualization), the same considerations made for Hadoop apply to Spark.



(a) Hadoop execution flow.



(b) Spark execution flow.

**Fig. 2.** Hadoop and Spark execution flows.

## 4 Case studies

We ran experiments to evaluate the scalability of the Spark version of ParSoDA proposed in this paper, in comparison with the previous Hadoop version of the library that has been presented in [4]. The experimental evaluation is based on two case studies based on the analysis of social media data published in Flickr and Twitter. The first application aims at discovering sequential patterns from user movements, so as to find the common routes followed by users. The second one aims at discovering the frequent sets of places visited by users. The analysis was carried out by analyzing 325 GB of social media data published in Flickr and Twitter from November 2014 to July 2016 that refer to the center of Rome.

### 4.1 Application code

Listing 1.1 shows the code of the application for executing the sequential pattern mining. First, an instance of the *SocialDataApp* class must be created (*line 1*). Then a file containing the boundaries of the regions of interest (*RomeRoIs.kml*) is distributed to the processing nodes (*lines 2-3*). Afterwards, the different steps of the application are configured as described here:

1. *Data collection.* The names of two crawling classes (*FlickrCrawler* and *TwitterCrawler*) are defined in the *cFunctions* array (*line 4*). The parameters used to configure the instances of the two crawling classes are defined in the *cParames* array (*line 5*). The two arrays are then passed to the *setCrawlers* method (*line 6*).
2. *Data filtering.* Two filtering classes are specified: *IsGeotagged* and *IsInPlace* (*line 7*). The former filters data by keeping only geotagged items. The latter filters out data that are not in the center of Rome, which is defined by its geographical coordinates. The parameters of the two filtering functions are specified in the *fParams* array (*line 8*). The names of the filtering classes and associated parameters are then passed to the *setFilters* method (*line 9*).
3. *Data mapping.* The map class *FindPoI* (*line 10*), which does not require parameters to be instantiated (*line 11*), is specified. The mapping function defined in *FindPoI* assigns to each social media item the name of the place it refers to. To do this, it refers to the boundaries specified in the file defined at *line 2*. The name of the map class is then passed to the *setMapFunctions* method (*line 12*).
4. *Data partitioning.* The id of the user who posted a social media item is used as the *groupKey* (*line 13*), while the date and time when the social media item was posted is used as the *sortKey* (*line 14*). The two keys are then passed to the *setPartitioningKeys* method (*line 15*).
5. *Data reduction.* A reduce class, named *ReduceByTrajectories* (*line 16*), is specified to aggregate all the social media items posted by a single user, into a list of individual trajectories across places. The parameters of the reduce class are specified in the *rParams* string (*line 17*). In particular, it receives only a parameter $t$, which is the maximum time gap in hours that can be

taken for consecutive places in the same trajectory. The name of the reduce class and its parameters are then passed to the *setReduceFunction* method (*line 18*).

6. *Data analysis.* A data analysis class, named *PrefixSpan*, is specified (*line 19*). The class implements PrefixSpan [16], a scalable frequent sequence mining algorithm, built for Spark and included in the Spark Machine Learning library (MLlib), which takes as input a collection of sequences and mines frequent sequences. The parameters of data analysis class are specified in the *aParams* string (*line 20*). The name of the data analysis class and its parameters are then passed to the *setAnalysisFunction* method (*line 21*). In the Hadoop version of the application presented in [4], as data analysis class we used *MGFSM* [14], a scalable frequent sequence mining algorithm built for MapReduce.
7. *Data visualization.* The *SortResults* class is specified to perform the data visualization function (*line 22*). A configuration string *vParams*, containing the parameters of the data visualization class, is specified at *line 23*. The class receives two parameters: the key used to sort results (the sequence support) and the sort direction (descending order). The name of the data visualization class and its parameters are then passed to the *setVisualizationFunction* method (*line 24*).

Finally, the execution of the application is obtained by invoking the *execute* method (*line 25*).

```
1  SocialDataApp app = new SocialDataApp("SPM - City of Rome");
2  String[] cFiles = {"RomeRoIs.kml"};
3  app.setDistributedCacheFiles(cacheFiles);
4  String[] cFunctions = {"FlickrCrawler","TwitterCrawler"};
5  String[] cParams = {"-lat 12.492 -lng 41.890 -radius 10 -startDate
       2016-07-31 -endDate 2014-11-01","-lat 12.492 -lng 41.890 -radius 10 -
       startDate 2016-07-31 -endDate 2014-11-01"};
6  app.setCrawlers(cFunctions,cParams);
7  String[] fFunctions = {"IsGeotagged","IsInPlace"};
8  String[] fParams = {"true","-lat 12.492 -lng 41.890 -radius 10"};
9  app.setFilters(fFunctions, fParams);
10 String[] mFunctions = {"FindPoI"};
11 String[] mParams = null;
12 app.setMapFunctions(mFunctions, mParams);
13 String groupKey = "USER.USERID";
14 String sortKey = "DATETIME";
15 app.setPartitioningKeys(groupKey,sortKey);
16 String rFunction = "ReduceByTrajectories";
17 String rParams = "-t 5";
18 app.setReduceFunction(rFunction,rParams);
19 String aFunction = "PrefixSpan";
20 String aParams = "-maxPatternLength 5 -minSupport 0.01";
21 app.setAnalysisFunction(aFunction,aParams);
22 String vFunction = "SortBy";
23 String vParams = "-k support -d DESC";
```

```
24 app.setVisualizationFunction(vFunction,vParams);
25 app.execute();
```

**Listing 1.1.** An example of sequential pattern mining (SPM) application on Flickr and Twitter data from the City of Rome, written using the ParSoDA library.

The code for executing the frequent itemset analysis differs from that described above only for the used data analysis algorithm (*lines 19-21*). In particular, for extracting frequent sets of places from social media data, a parallel implementation of FP-Growth [11] called PFP [13], has been used both in the Spark- and in the Hadoop-version of the application.

## 4.2 Applications results

A set of 24 popular places in the center of Rome have been considered to run the sequential pattern mining task and the frequent itemset discovery task, both implemented as ParSoDA applications. In the following, we discuss some of the most interesting results that have been obtained. Table 1 shows the top 5 places visited in Rome, with the corresponding support in the data. The Colosseum is the most visited place, followed by the St. Peter's Basilica.

**Table 1.** Top 5 places visited in Rome

| Place | Support |
|---|---|
| Colosseum | 21.7% |
| St Peter's Basilica | 13.9% |
| Trastevere | 8.7% |
| Pantheon | 6.5% |
| Trevi Fountain | 5.3% |

**Table 2.** Top 5 frequent sets of places visited in Rome

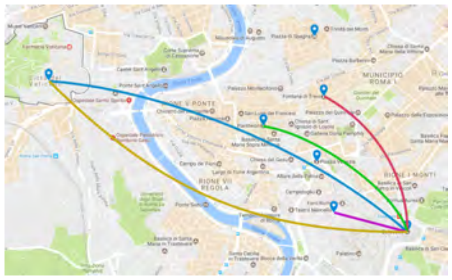
| Set of places | Support |
|---|---|
| Pantheon, St. Peter's Basilica, Colosseum | 5.3% |
| Trevi Fountain, St. Peter's Basilica, Colosseum | 4.5% |
| Roman Forum, St. Peter's Basilica, Colosseum | 4.4% |
| Vatican Museums, St. Peter's Basilica, Colosseum | 4.4% |
| Trevi Fountain, Pantheon, Colosseum | 4.0% |

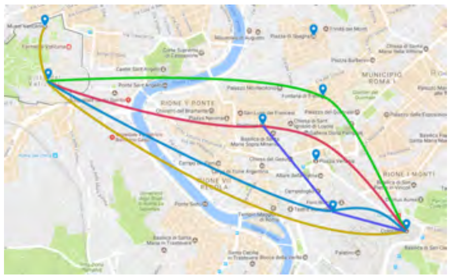
Table 2 shows the most frequent itemsets of length 3 that have been discovered by the PFP algorithm. Set {*Pantheon, St. Peter's Basilica, Colosseum*} is the most frequent set of places visited by social users in Rome, with a support of 5.3%. Combining the information contained in Tables 1 and 2, an interesting result is that *Trastevere*, a popular district of Rome, is the third most visited place, but it is not present in any frequent itemset. This could happen because Trastevere is visited by people during the evening, for having a dinner in one of its many restaurants or pubs, but it is not part of common tourist routes during the daylight.

The sequential pattern analysis has been carried out for discovering the most frequent routes in Rome. In this experiment, it has been set a maximum time duration (gap) to move from a place to another of 5 hours. This means that if the time distance between two contiguous places in sequence is greater than 5 hours, they will belong to different sequences.

Figure 3(a) shows the top five visited places in Rome that have been found by the PFP algorithm. Figures 3(b), 3(c), 3(d) show respectively the top five interesting patterns of length 3, 4, and 5, which have been found by the PrefixSpan algorithm. More detailed information about the most frequent patterns and the corresponding supports are reported in Table 3. Considering the sequential patterns of length 2, the sequence $\{Colosseum \rightarrow St.\ Peter's\ Basilica\}$ is the most frequent route among places in Rome, followed by 9.07% of users. The sequence $\{Colosseum \rightarrow Roman\ Forum \rightarrow St.\ Peter's\ Basilica\}$ is the most frequent route of length 3, which is followed by 4.4% of users. Finally, the sequence $\{Colosseum \rightarrow Trevi\ Fountain \rightarrow Pantheon \rightarrow St.\ Peter's\ Basilica\}$ is the most frequent route of length 4 with a quite low support of 0.64%.

(a) Top 5 places of interest in Rome.


(b) Top 5 sequential patterns of length 2.


(c) Top 5 sequential patterns of length 3.

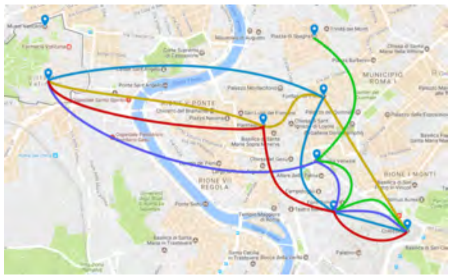

(d) Top 5 sequential patterns of length 4.

**Fig. 3.** Sequential pattern mining application.

## 4.3 Scalability evaluation

As mentioned before, we experimentally evaluated the scalability of the Spark version of ParSoDA proposed in this paper, compared to the previous Hadoop version of the library. The scalability was evaluated running the data analysis applications on the Microsoft Azure cloud. Specifically, we used one cluster equipped with 2 head nodes (each one having four 2.2 GHz CPU cores and 14 GB of memory), and 12 worker nodes (each one equipped with four 2.2 GHz CPU cores and 14 GB of memory). Here we present the results obtained with

**Table 3.** Top 5 sequential patterns of length 2, 3 and 4 across places in Rome

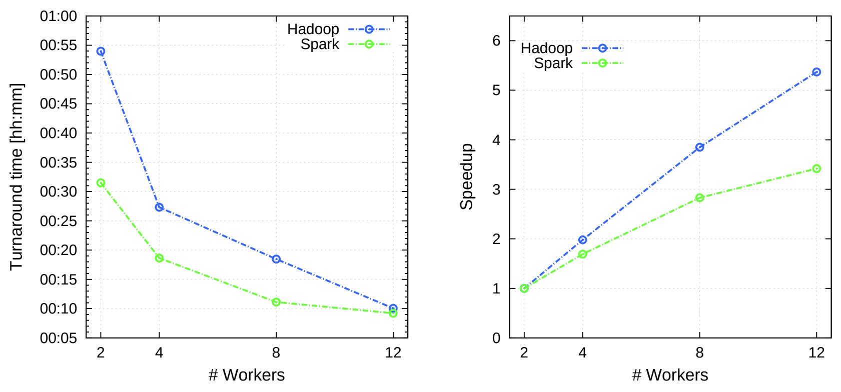
| **Sequential pattern** | **Support** |
|---|---|
| Colosseum → St. Peter's Basilica | 9.07% |
| St. Peter's Basilica → Colosseum | 7.72% |
| Colosseum → Roman Forum | 5.28% |
| Colosseum → Pantheon | 4.44% |
| Colosseum → Trevi Fountain | 4.19% |
| Colosseum → Roman Forum → St. Peter's Basilica | 4.4% |
| Vatican Museums → St. Peter's Basilica → Colosseum | 3.9% |
| Colosseum → Trevi Fountain → St. Peter's Basilica | 3.7% |
| Colosseum → Roman Forum → Pantheon | 3.6% |
| Colosseum → Pantheon → St. Peter's Basilica | 3.6% |
| Colosseum → Trevi Fountain → Pantheon → St. Peter's Basilica | 0.64% |
| Colosseum → Roman Forum → Trevi Fountain → San St. Peter's Basilica | 0.61% |
| Colosseum → Roman Forum → Piazza Venezia → Piazza di Spagna | 0.58% |
| Colosseum → Roman Forum → Piazza Venezia → St. Peter's Basilica | 0.58% |
| Colosseum → Roman Forum → Pantheon → St. Peter's Basilica | 0.58% |

the sequential pattern mining application. The performance obtained with the frequent itemset applications are almost identical.

As shown in Figure 4(a), the turnaround time of the Hadoop-based application decreases from about 54 minutes using two workers, to 10 minutes using 12 workers. The turnaround time of the Spark-based application decreases from about 32 minutes using two workers, to 9 minutes using 12 workers. Thus, using the same computing resources, the Spark version of ParSoDA results to be 8% (12 workers) to 42% (2 workers) faster than the Hadoop version. In terms of speedup (see Figure 4(b)), Hadoop obtains a speedup ranging from 1.98 using 4 workers, to 5.37 using 12 workers. On the other hand, the Spark version achieves a lower relative speedup than Hadoop, as it passes from 1.74 using 2 workers, to 3.53 using 12 workers. This is due to the fact that the Spark version spends most of the time to load data in memory and to distribute it across the worker nodes. Thus, for such application, increasing the number of nodes beyond a certain number seems not have significant benefits. However, the advantage of Spark over Hadoop is significant in terms of absolute times reduction, as shown by the results presented in Figure 4(a).

## 5 Conclusions

Social media analysis is an important research area aimed at extracting useful information from the big amount of data gathered from social networks. To cope with the size and complexity of social media data, the use of parallel and distributed data analysis techniques is fundamental. ParSoDA is a Java library that can be used for building parallel social data analysis applications. ParSoDA defines a general structure for a social data analysis application that includes a number of steps (data acquisition, filtering, mapping, partitioning, reduction, analysis, and visualization), and provides a predefined (but extensible) set of



(a) Turnaround time. (b) Relative speedup.

**Fig. 4.** Turnaround time and relative speedup of the sequential pattern mining application using Hadoop and Spark.

functions for each step. In a previous work [4], we described how ParSoDA can be used to run parallel social data analysis on the cloud using Hadoop.

In the present work we presented an extension of ParSoDA to execute applications on Spark. We experimentally evaluated the scalability of the Spark version of ParSoDA compared to the previous Hadoop version of the library. The experimental evaluation is based on two case study applications on social media data published in Flickr and Twitter. The ParSoDA library performance has been evaluated carrying out the data analysis applications both on a Hadoop and a Spark cluster deployed on the Microsoft Azure cloud platform. The results obtained on a cluster with 12 workers, showed that the Spark version of ParSoDA was able to reduce the execution time up to 42% compared to the Hadoop version of the library.

## References

1. Amer-Yahia, S., Ibrahim, N., Kengne, C.K., Ulliana, F., Rousset, M.C.: Socle: Towards a framework for data preparation in social applications. Ingénierie des Systèmes d'Information 19(3), 49–72 (2014)
2. Anstead, N., O'Loughlin, B.: Social media analysis and public opinion: The 2010 uk general election. Journal of Computer-Mediated Communication 20(2), 204–220 (2015)
3. Belcastro, L., Marozzo, F., Talia, D., Trunfio, P.: Big data analysis on clouds. In: Sakr, S., Zomaya, A. (eds.) Handbook of Big Data Technologies, pp. 101–142. Springer (December 2017), iSBN: 978-3-319-49339-8
4. Belcastro, L., Marozzo, F., Talia, D., Trunfio, P.: A parallel library for social media analytics. In: The 2017 International Conference on High Performance Computing & Simulation (HPCS 2017). Genoa, Italy (17-21 July 2017)

5. Cesario, E., Congedo, C., Marozzo, F., Riotta, G., Spada, A., Talia, D., Trunfio, P., Turri, C.: Following soccer fans from geotagged tweets at fifa world cup 2014. In: Proc. of the 2nd IEEE Conference on Spatial Data Mining and Geographical Knowledge Services. pp. 33–38. Fuzhou, China (July 2015), iSBN 978-1- 4799-7748-2
6. Cesario, E., Iannazzo, A.R., Marozzo, F., Morello, F., Riotta, G., Spada, A., Talia, D., Trunfio, P.: Analyzing social media data to discover mobility patterns at expo 2015: Methodology and results. In: The 2016 International Conference on High Performance Computing and Simulation (HPCS 2016). Innsbruck, Austria (18-22 July 2016)
7. Chodorow, K.: MongoDB: the definitive guide. " O'Reilly Media, Inc." (2013)
8. Chu, C., Kim, S.K., Lin, Y.A., Yu, Y., Bradski, G., Ng, A.Y., Olukotun, K.: Map-reduce for machine learning on multicore. Advances in neural information processing systems 19, 281 (2007)
9. Cuesta, Á., Barrero, D.F., R-Moreno, M.D.: A framework for massive twitter data extraction and analysis. Malaysian J. of Computer Science 27, 1 (2014)
10. Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. In: Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation. pp. 10–10. OSDI'04, Berkeley, USA (2004)
11. Han, J., Pei, J., Yin, Y., Mao, R.: Mining frequent patterns without candidate generation: A frequent-pattern tree approach. Data mining and knowledge discovery 8(1), 53–87 (2004)
12. Hussain, A., Vatrapu, R.: Social Data Analytics Tool (SODATO), pp. 368–372. Springer International Publishing, Cham (2014)
13. Li, H., Wang, Y., Zhang, D., Zhang, M., Chang, E.Y.: Pfp: Parallel fp-growth for query recommendation. In: Proceedings of the 2008 ACM Conference on Recommender Systems. pp. 107–114. New York, NY, USA (2008)
14. Miliaraki, I., Berberich, K., Gemulla, R., Zoupanos, S.: Mind the gap: Large-scale frequent sequence mining. In: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. pp. 797–808 (2013)
15. Pang, B., Lee, L.: Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval 2(12), 1–135 (2008)
16. Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., Hsu, M.C.: Mining sequential patterns by pattern-growth: the prefixspan approach. IEEE Transactions on Knowledge and Data Engineering 16(11), 1424–1440 (Nov 2004)
17. Shvachko, K., Kuang, H., Radia, S., Chansler, R.: The hadoop distributed file system. In: Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on. pp. 1–10. IEEE (2010)
18. Talia, D., Trunfio, P., Marozzo, F.: Data Analysis in the Cloud. Elsevier (October 2015)
19. White, T.: Hadoop: The Definitive Guide. O'Reilly Media, Inc., 1st edn. (2009)
20. Xin, R.S., Rosen, J., Zaharia, M., Franklin, M.J., Shenker, S., Stoica, I.: Shark: Sql and rich analytics at scale. In: Proceedings of the 2013 ACM SIGMOD Conference on Management of data. pp. 13–24. ACM (2013)
21. You, L., Motta, G., Sacco, D., Ma, T.: Social data analysis framework in cloud and mobility analyzer for smarter cities. In: IEEE Int. Conference on Service Operations and Logistics, and Informatics. pp. 96–101 (Oct 2014)
22. Zaharia, M., Xin, R.S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M.J., et al.: Apache spark: A unified engine for big data processing. Communications of the ACM 59(11), 56–65 (2016)