Cloud-based architectures for Geo-located blogosphere dynamics detection

Athena Vakali
Department of Informatics
Aristotle University
54124 Thessaloniki, Greece
+30 2310998415
avakali@csd.auth.gr

Stefanos Antaris
Department of Informatics
Aristotle University
54124 Thessaloniki, Greece
+30 2310991865
santaris@csd.auth.gr

Maria Giatsoglou
Department of Informatics
Aristotle University
54124 Thessaloniki, Greece
+30 2310998236
mgiatsog@csd.auth.gr

ABSTRACT
Social networking data threads emerge rapidly and such crowd-driven big data streams are valuable for detecting trends and opinions. For such analytics, conventional data mining approaches are challenged by both high-dimensionality and scalability concerns. Here, we leverage on the Cloud4Trends framework, for collecting and analyzing geo-located microblogging content, partitioned into clusters under cloud-based infrastructures. Different cloud architectures are proposed to offer flexible solutions for geo-located data analytics, with emphasis on incremental trend analysis. The proposed architectures are largely based on a set of service modules which facilitate the deployment of the experimentation on Cloud infrastructures. Several experimentation remarks are highlighted to showcase the requirements and testing capabilities of different cloud computing settings.

General Terms
Cloud based models and implementations, social data analytics, go-located data experimentation

Keywords : social networks and wisdom of the crowd, geo-located blogosphere dynamics, social geo-located data clustering, cloud service deployment.

1. INTRODUCTION
Huge data threads are produced in social media constantly, with microblogging and blogging frameworks dominating people's trend to broadcast information in a real-time fashion. In Twitter, for example, tweets threads reach massive sizes, currently in the scale of 500 million per day and around 200 billion tweets per year[1]. The dynamic and unstructured nature of such information broadcasting offers an abundance of data out of which unexpected latent information can be harvested. Moreover, the current practice of declaring geo-location offers new sources of metadata (such as time, point of interest, geo-coordinates etc) which can reveal important trends in a geo-bounded area, such as in a city.

User-originating information posted on social media typically reflect topics of their actual interest, so such crowd-sourced and in particular geo-tagged content is particularly useful for geo-located trends detection. Detecting trends of an area (such as in a city), is of major importance, due to the fact that trends can be utilized to spot collective emergent or evolving behaviour and phenomena, useful for proceeding to appropriate decision and city policy making.

Raw information from Twitter has been exploited in research at several domains, such as for

[1] http://www.internetlivestats.com/twitter-statistics/

predicting revenues and stock prices, for the real-time identification of phenomena, for political standings etc. Therefore, it is well acknowledged that the microblogging "sphere" forms a valuable source of latent information relevant with the dynamics involved in the public's opinions, and views. This is further justified by the fact that such applications capture the dynamics and the co-evolution social pulse [1]. Blogosphere as well is a rich information source at which the dynamics and the "voice of the public" may be extracted and mined especially with respect to certain locations or events. Therefore, microblogging and blogging activities can serve as major social dynamics barometers. This is due to the fact that such parallel information flows embed valuable and often hidden information about trending users' interests and opinions in a geo-located area.

This paper places emphasis on both the design framework and on the leveraging of the cloud paradigm to stress-test data analytics for localized trending topics detection. Trends dynamics are harvested and may be analyzed over user-contributed content from both microblogging (Twitter) and blogosphere activities through an approach such as in incremental text clustering. To support such an approach, dealing with high processing and data management demands, different cloud-based architectures have been designed and proposed for various experimentation scales and requirements.

The proposed framework's contribution is summarized in its following objectives:

- *dealing with the large scale data production in the Web 2.0 micro-blogsphere* (with huge and rapidly evolving data) by enabling methods efficient implementation, useful for real world application settings;
- *supporting the analysis of text data from emergent web sources* which may be generated at various rates under a unified data processing cloud-leveraging manner;
- *proposing a methodology for unsupervised detection of local trends* by combining content from different sources to enrich detected geo-located related trends;
- *designing and experimenting with different Cloud-based infrastructures* to support a geo-located social data processing scenarios under parallelized computation settings.

The rest of the paper is structured as follows. Section 2 reviews the current state regarding trend detection approaches that leverage microblogging and blogging data, discussing their challenges. Section 3 outlines the idea for leveraging a cloud based trends detection framework, under which potential, with its implementation details summarized in Section 4. Different cloud based architectures are proposed in Section 5 to enable different architectures focus ranging from lightweight to fully parallelized solutions. At Section 6 some indicative experimentation results are outlined in order to highlight the cloud based solutions capabilities which can be exploited for different social media data threads. Finally, Section 6 has the conclusions and indication for the proposed work's exploitation in real city bounded use cases.

2. MICRO-BLOGOSPHERE TRENDS DETECTION : STATUS AND CHALLENGES

Localized trend detection and "public's pulse" monitoring strongly set the need for efficient scalable and/or summarizing methodologies and frameworks. Current data mining (such as clustering) approaches focus on detecting (e.g. in twitter): (i) clusters of users densely associated via follower or message links, or (ii) groups of tweets using text mining techniques, such as exploiting common word co-occurrences [17].

2.1 MINING FOR TREND ANALYSIS

A typical approach to trend analysis involves tracking users' interests in different keywords across time. Already, temporal trend analysis based on keyword frequency has appeared in several commercial blog and Web search engines

such as: Google Hot Trends[2] and BlogPulse[3]. Although Google Hot Trends analyzes millions of web searches to identify trends, it does not emphasize on social data analytics. This leaves aside a collective source of intelligence which embeds opinions, facts and sentiments. BlogPulse is an online service that discovers trends from blogs on a daily basis with statistical techniques for detecting trending phrases based on their frequency of appearance [2]. Other online services such as Twitter-related Trendistic[4], outline term frequency trends again under statistical methodologies. Twitter itself also exhibits local trends[5] (for some locations) as keywords which are popular at the current time and at a particular city.

Clustering has been widely applied on content generated in Web social media to uncover latent associations, while recently the feature of time and the temporal evolution of clusters have been researched [8], [9]. Social data hierarchical graph clustering approaches have been applied for trend detection, in cases where associations among cross-blogs are modeled with a graph structure [3], [9]. This approach operates on a static dataset, as it is not tailored for real-time online operation.

In TwitterStand, news detection from tweets, is based on data from the Twitter's GardenHose service (with a sample of Twitter's public timeline) [6]. To deal with noise, TwitterStand also filters out tweets that are unrelated to news via a classification method based on the Naïve Bayes Classifier. After that, the tweets are clustered with an online method that holds many similarities to the one followed in Cloud4Trends application [17]. In particular, the TwitterStand's algorithm extracts TF-IDF feature vectors for the tweets and the clusters and performs clustering based on their similarity, while it also incorporates the temporal dimension in the clustering process in the same way as Cloud4Trends does.
TwitterMonitor [6] is another framework for online trend detection over Twitter, following an approach similar to BlogScope [3].

[2] http://www.google.com/trends
[3] http://www.blogpulse.com/trends.html

2.2 CHALLENGES IN CROWDSOURCED TREND ANALYSIS

Data Characteristics	Challenges
Vast size: huge amount of textual content, e.g. posts, tweets, comments, etc., produced on social media is an intrinsic characteristic for social data analysis	**Scalability:** a scalable clustering methodology is suitable to process the vast amount of social text data, and social media mining requires features such as geo-location and time.
Noisy Data: Social text data are mostly written in informal style and have simple phrases, abbreviations, etc.	**Data Preprocessing:** An efficient text preprocessing stage which identifies the noisy data is of great value for knowledge extraction.
Dynamic data: Social media users produce new textual data at unexpected rates of time.	**Online Processing:** A real-time, adaptive text clustering approach is needed to process highly evolving social data
Social data are geo and time-dependent: Social users generate textual content of similar topic at a specific time period, and at different locations	**Streaming Clustering:** Social text data processed online must meet memory, performance requirements in cases of streaming clustering algorithms.

Table 1. Online social data processing challenges

Several solutions have been proposed to surpass the challenges imposed on social text clustering algorithms. These challenges are largely due to the inherent main social data characteristics (summarized in Table 1) :

[4] http://trendistic.com/
[5] https://support.twitter.com/articles/101125

- **Vast social data sizes :** which demand scalable solutions, dealing with the computational time complexities required by conventional text mining algorithms. Emerging clustering approaches should be considered to result in efficient social text data analysis, since data need to be processed at a limited amount of time [18],[19]. Current parallel and distributed infrastructures are proposed to meet the scaling demands of the clustering algorithms, and already several parallel or distributed clustering approaches have been proposed reducing both the computational cost and the execution time [16],[18],[20].
- **Noisy social networks data :** pose the need for methodologies dealing with the multiple noise states, due to the non formal and unstructured social networks expression. Several text preprocessing methodologies (e.g. emoticon identification, acronyms recognition, etc.,) are proposed as vital for the refinement of the text content and the improvement of the clustering approach [17].
- **Dynamic data threads :** demand fast and often real-time processing and monitoring so new adaptive methodologies should be considered and validated [9].
- **Social data geo and time-dependencies :** require multiple features integration since both geo-location and time are crucial in several location based social networks analytics [13].

3. LEVERAGING CLOUD4TRENDS FOR SOCIAL TEXT DYNAMICS DETECTION

This work places emphasis on dynamics detection in a geo-located and time related context. It leverages on Cloud4Trends, a framework proposed by the authors to enable the online identification of trends dynamics, using Twitter and the Blogosphere [17].

It is important to notice that some commercially available products have focused on offering trend analytics solutions (such as [22], [23]). These tools place emphasis on attracting social customers by unifying engagement, visual planning, and collective collaboration. Their focus is on performance analytics for real-time campaign decisions and they differ with Cloud4Trends in terms of their focus on customer interactions and not on the latent knowledge extraction. By using Cloud4Trends, text clustering is employed in an incremental manner for detecting and maintaining a set of dynamic clusters. This framework is based on the assumption that the analysis is implemented on a "document" level, instead of a "term" level, whereas the corresponding clustering approach follows the TwitterStand process [6]. It is important to note that with Cloud4Trends clusters which are active at a given time and locations express the so called active topics which are of users' interest. By dynamically observing the clusters' updating rate, we identify trends at their peak and detect the topics that are no more trending. This is followed instead of applying a fixed-threshold based method that sets as inactive clusters after a predefined period of time. In our approach we separately collect and clusters tweets that pertain to a desired geographical area, rather than examining the geographical scope of the resulting clusters as a post-analysis process.

In our proposed process here, we proceed to a microblogging analysis performed on a streaming fashion to capture constantly changing trending users' interests, with an analysis which further :

- exploits associations based on the broadcasting **time**, alleviating gaps in earlier efforts such as in [1], which employs a clustering method after identifying a set of trending phrases and focuses only on the latter, in an offline fashion;
- deals with the respective user's physical **location** (exploiting the tweet geo-location feature).

Such mutual multi-feature analysis is expected to produce more fine-grained high-quality clusters of tweets which will correspond to actual topics that are popular at a given location and time period. It is also expected to alleviate the generally acknowledged problem of noisy microblogging

data, since the joint consideration of location and time generally improves the clustering quality and contributes to filtering out noisy tweets.

The proposed process is outlined in Figure 1 and it actually involves a 3-tier design that deals with the: i) collection of data in a streaming manner from Twitter as well as from a pool of selected blogs focused on a number of geographic areas, ii) application of an online clustering technique on the data to detect recent trending topics, and iii) refinement and ranking of clusters such that trends are detected and visualized. These three tiers are summarized in the next subsections

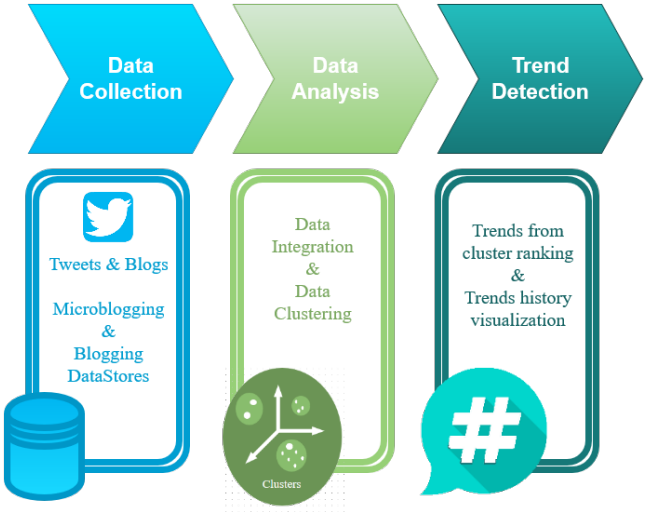

Figure 1. Microblogging Trend Detection Outline

3.1 The Data Collection Tier

The Data collection tier involves special online data aggregators for collecting recently published content from Twitter and the Blogosphere. The content corresponds to some specific geographic area (such as a city level), leveraging the Twitter Streaming API[6] and Google Blogger API[7] (other possibilities in blogging and microblogging platforms can also be considered). While the first API provides a continuous stream of recently generated posts the second one is based on REST requests. To this end, based on a collection of identifiers of blogs owned by Blogger users who have declared that they reside within the monitored geographic area, we use the API for requesting new posts for each blog at a fixed time interval (e.g. daily, which is reasonable since we do not expect blogs to be updated as frequently as content in Twitter).

3.2 The Data Analysis and Processing Tier

The retrieved posts (either tweets, blog posts, or extended tweets) are processed in order to produce clusters which contain posts pertaining to the same topic. Data are filtered to remove low quality content with typical approaches including filtering out tweets/blog posts with very few terms, etc. Text sanitization techniques are applied on the resources' text to filter out common words (defined in a stop word lists) and to perform stemming. Next, resources, are represented with a common model that includes: a unique identifier, a TF-IDF-based key-value map, a timestamp, and the resource's type (tweet, blog post, or extended tweet). For a given resource the key-value map structure includes as keys all the resource's unique terms, taken from the initial data model's *text*, *tags* and *title* (for blogs only) attributes. Using the Lucene Search Engine library[8] separate indexes are kept for each resource type and for each attribute. Though these indexes, TF-IDF key-value maps are obtained for each attribute.

3.3 The Trend Detection and Visualization Tier

This tier builds on the basis of the outcome of the data processing tier which produces three sets of clusters for the tweets', blog posts', and extended tweets' datasets. A given cluster can be characterized as *active* or *inactive* based on whether it is corresponds to topics that are popular at the given time, or it corresponds to topics that are no longer considered as trending. Clusters update rates are monitored to determine when a cluster should be made inactive due to limited activity. To this end, additional information is maintained for each cluster: *the evolution of the temporal distance between the timestamps of the last two resources assigned to the given cluster.*

[6] https://dev.twitter.com/docs/streaming-api
[7] http://code.google.com/apis/blogger/
[8] http://lucene.apache.org/core/

By taking the moving average of the aforementioned parameter to smoothen its evolution, we can identify periods of time when the cluster is increasingly rising in popularity. This is due to users' intense activity, when it is at its peak. To improve clusters' quality, the tiny sized clusters (with very few members) are considered as noise and thus are eliminated.

Active clusters are considered as representative of topics that concern web users at a given times, however, in order to identify the actual trends, clusters should be ranked in terms of an activity measure. To this end, for each type of content (and for a given monitored location) Cloud4Trends retrieves the *active* clusters and ranks them based on: i) their members' number, and ii) their mean timestamp, under the assumption that the "hottest" topics are those that are referred to in many resources, and that are additionally being created on average close to the current time.

The topics that characterize each cluster are identified as the terms with the highest scores in the cluster's mean key-value map. Cloud4Trends then generates a summary description for each cluster comprising of few member terms or phrases based on their scores and their significance (hashtags, title terms, etc), while the high-ranked clusters shape the trending topics for the given time.

In Cloud4Trends trends are therefore calculated based on the three different data sources for each location under investigation. Depending on the update rates of the resources' types (e.g. faster in Twitter while slower in blogs), one can decide on how often the clusters' "trending scale" will be recalculated.

4. IMPLEMENTATION DESIGN OF A CLOUD-BASED FRAMEWORK FOR BLOGOSPHERE DYNAMICS

Cloud4Trends is proposed to handle data intensive use cases as it involves concurrent analysis of large sizes of web social data in an online fashion. In handling for example, both tweets, with unexpectedly peaks, and blogs whose sizes may be considerably large, problems for handling large and fluctuating sizes of data arise. Feasible approaches should address parallel programming techniques required for many of our proposed operations, as for example in the cases of :

- data which should be concurrently analyzed for the different geographic areas and their analysis should be done effectively;
- blogs and Twitter data which should be collected in parallel;
- data collection module which should be constantly available for receiving new data;
- data processing which should be carried out for data already arrived and awaiting analysis under different concurrent process.

Therefore, suggested ideas can heavily utilize the Cloud computing paradigm which offers a significant ground for such social streams mining applications due to its support via scalable and powerful infrastructures [8]. Our design requirements match well with the *MapReduce* computing paradigm, which codifies a generic "recipe" for processing large datasets when this processing consists of more than one stage. The MapReduce technology matches the needs of the Clou4Trends data analysis and processing tier, given that in a cloud-based deployment the *mapping* operations can be distributed into separate computer nodes. Prior to being ported to the Cloud, Cloud4Trends ran into a multi-core computer, designed over a software architecture which posed obstacles in aggregating and analyzing data from both Twitter and blogs. We believe that parallel approaches in cloud computing infrastructures constitute viable solutions for real-time large-scale data mining applications.

Since Cloud4Trends aims is to validate the quality of the resulting clusters and observe and quantify the differences in the trends resulting from the three data sources which represent different user groups. Cloud infrastructure can be leveraged for efficiently handling both the data's high scalability, the requirement for real-time tweet processing and clusters' update, as well as for ensuring quality of service for an increasing number of end users (inline with Table 1 challenges).

4.1 The VENUS-C Infrastructure

The suggested system (as described in previous section) is currently implemented in the context of the so called "Cloud4Trends" experiment[9] entitled "*Leveraging the Cloud infrastructure for localized real-time trend detection in social media*", which runs over the VENUS-C infrastructure. VENUS-C[10] (*Virtual Multidisciplinary EnviroNments USing Cloud Infrastructures*) is a pioneering project that develops and deploys a Cloud computing service for research and industry communities in Europe by offering an industrial-quality, service-oriented platform based on virtualization technologies and taking advantage of previous experience on Grids and Supercomputing, to facilitate a range of research fields through easy deployment of end-user services.

VENUS-C offers several service components to allow a wide range of end-user applications (targeting mainly research groups and SMEs) to benefit from the advantages of a Cloud computing platform, without having to develop custom Cloud-aware solutions. It offers a selection of programming models that, combined with appropriate data access mechanisms, constitute a convenient abstraction for deploying scientific applications on top of plain virtual machines. VENUS-C programming model enactment services expose their functionality via an *Open Grid Forums Basic Execution Service* (OGF BES) [15] and *Job Submission Description Language* (JSDL) [16] compliant web service interface, and take care of the enactment of a job at a given Cloud Provider. Each enactment service deploys a specific application on a number of either Windows Azure[11] virtual machines or Unix virtual resources from open source Cloud middlewares (OpenNebula[12] & EMOTIVE Cloud[13]).

A Cloud-based data management SDK is provided by VENUS-C for handling all data transfer operations between the Cloud infrastructure and the on-premises client applications, which supports the Storage Networking Industry Association (SNIA) Cloud Data Management Interface (CDMI) specification [16]. Resource monitoring has also been taken into account in VENUS-C so that each end-user is to be able to identify its application's resource consumption via the VENUS-C Accounting Service.

4.2 The Cloud4Trends Cloud-based Architecture

The suggested design is implemented in Cloud4Trends under the previously described 3-tiered conceptual design structure. Cloud4Trends is implemented as a hybrid application based on the integration of : i) on-premises client interface and ii) multiple job execution components with different functionalities on top of the VENUS-C Cloud services infrastructure. In particular, Cloud4Trends uses VENUS-C Generic Worker programming model for job submission and application deployment on top of the Azure.

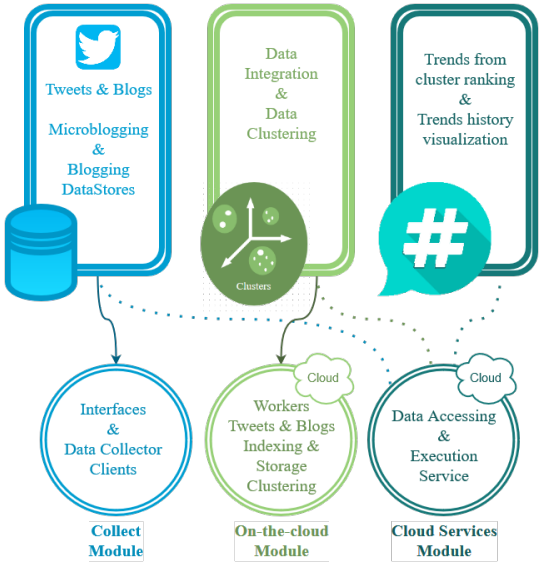

Figure 2. Trend Detection Cloud-based framework

Cloud infrastructure functionalities which enable effective data-driven and task-based job submissions are exploited. Figure 2 illustrates the three Cloud4Trends modules, namely: the Collect module, the on-the-Cloud module, and the Cloud

[9] http://oswinds.csd.auth.gr/?page_id=1320
[10] http://www.venus-c.eu/
[11] http://www.windowsazure.com/
[12] http://opennebula.org
[13] http://www.emotivecloud.net/

services module. These modules support the proposed 3-tier design (described above) such that the Client module implements the Data Collection and Visualization tiers, whereas the Cloud-based module implements the Data Processing and knowledge extraction tier.

More details on this implementation are given in [17], at which an indicative workflow is proposed to be orchestrated as follows : *Collect module*, which is responsible for collecting data from social Web (Twitter and Blogosphere) and initializing new experiments when required by the researchers, submits new Parsing jobs (executed by Twitter or Blog Parsers) to the cloud via the Job Submission Client of *Generic Worker*, which is hosted at the Execution Service, when new data are available. The required data for each given job are uploaded, as a batch, using the Data Access Service to Azure Blobs. Cloud4Trends' *Indexing Service Module* consists of three separate Azure services (for indexing tweets, blog posts, and extended tweets) that are responsible for the Full Text Indexing of the parsed data using the a Lucene library for Azure[14]. When an Indexing service completes its execution it initiates a *Splitter* Job using the Generic worker's Job Submission Client, which receives as input data the new resources. Splitter application also downloads the appropriate currently active clusters from the corresponding Azure Table. Different *Similarity* estimation Workers (Mapper jobs) are submitted by each Splitter job, via the Execution Service module, and in particular using the Generic Worker Local Job Submission service, which calculate the similarity scores between the new resource and the respective active clusters. An *Aggregation* Worker (Reducer job) is also submitted by the Splitter Job via the Local Job Submission service, to collect all similarity score combinations emitted by the corresponding Mapper jobs, and identify the best match to an existing cluster for each resource.

5. CLOUD ARCHITECTURES FOR SOCIAL DATA ANALYSIS

Cloud4Trends has exploited the *Windows Azure* infrastructure, which provides functionalities but also poses additional challenges since in cloud computing a predefined number of resources (i.e. number of CPU, RAM, etc.) is available to be utilized and shared. Each computer resource in the cloud executes a specific system's service, the *Worker Role* and Windows Azure provides several instances of resources to be defined on each Worker Roles. By initiating more than one instance of a Worker Role in order to provide parallelization, the number of reserved resources is increasing according to the defined resource's instance. As such, great emphasis should be given on the number of resources which are assigned on each *Worker Role* in order to efficiently divide the shared resources to the overall system's *Worker Roles.* An erroneous partition of the cloud resources over the *Worker Roles* may lead to non-scalable system architectures. Moreover, the *Azure Service Bus* service is provided to establish a fail-safe communication channel among the *Worker Roles,* and *Virtual Machine*s are hosted into the Windows Azure infrastructure, which allocate same shared resources with the other Worker Roles, under the same communication network to avoid network delays during their communication.

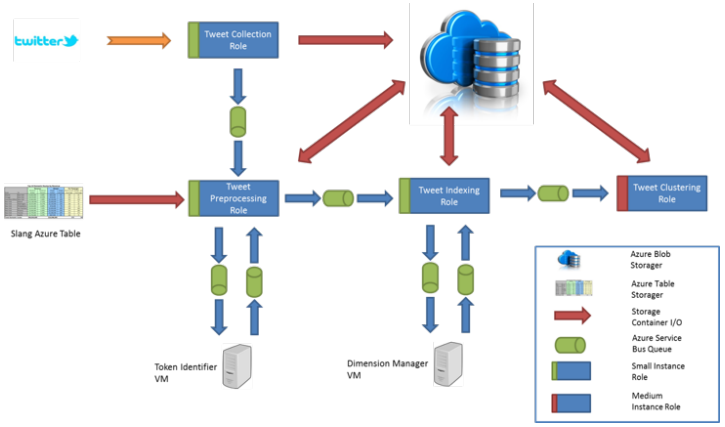

Figure 3. 1st architecture (lightweight)

[14] http://code.msdn.microsoft.com/windowsazure/Azure-Library-for-83562538

Based on the above specifications, we propose five distinct system architectures emphasizing on the differences among single-node and cloud-computing architecture :

- *1st system architecture, as a lightweight suggestion* at which a sequential flow of the social text clustering process over the cloud is accomplished, as shown in Figure 3. No parallelization is provided in any of our framework's component, and components of the same tier are included into a single Worker Role, providing though the minimum number of Worker Roles and just two Virtual Machines are to be generated for efficiently executing the component's process: i)*Token Identifier VM*, and ii) *Dimension Manager VM*
- *2nd system architecture, to focus on the text processing* and at which distinction of the tweet collection and the Json parsing process is emphasized. While the process of roles does not require high memory allocation, small instances of each role is needed and multiple instances of the Json Parser Role and the Tweet Preprocessing Roles are initialized (Figure 4) to provide components parallelization and offer a more scalable solution.
- *3rd system architecture, to support Tweet Indexing* and at which parallelization may be accomplished on the Vector Handler component, while the *Indexing* and the *Dimension Manager* components are not required to be parallelized (Figure 5). Since Windows Azure, enables programming balancing of a Worker Role's instances, flexibility and dimensionality reduction is reached.
- *4th system architecture, to place* emphasis *on the parallelization of clustering components* and at which Text Clustering Role is divided into three distinct roles, as shown Figure 6. This approach supports i) Similarity Calculator Role, ii) Results Comparator Role, and iii) Cluster Manager Role, such that Similarity Calculator Roles are processed effectively.

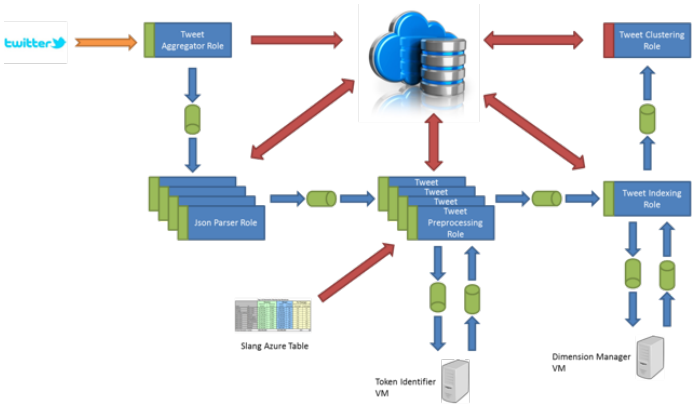

Figure 4. 2nd architecture (text processing)

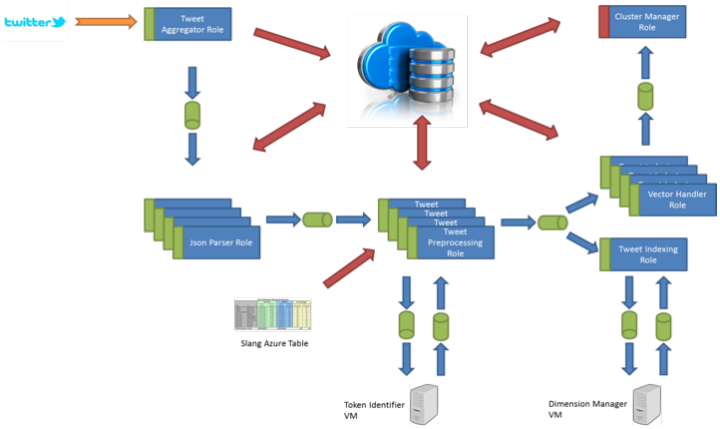

Figure 5. 3rd architecture (indexing)

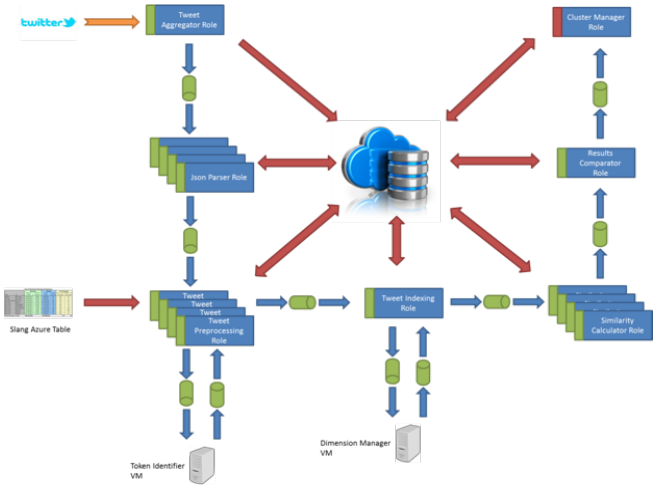

Figure 6. 4th architecture (clustering)

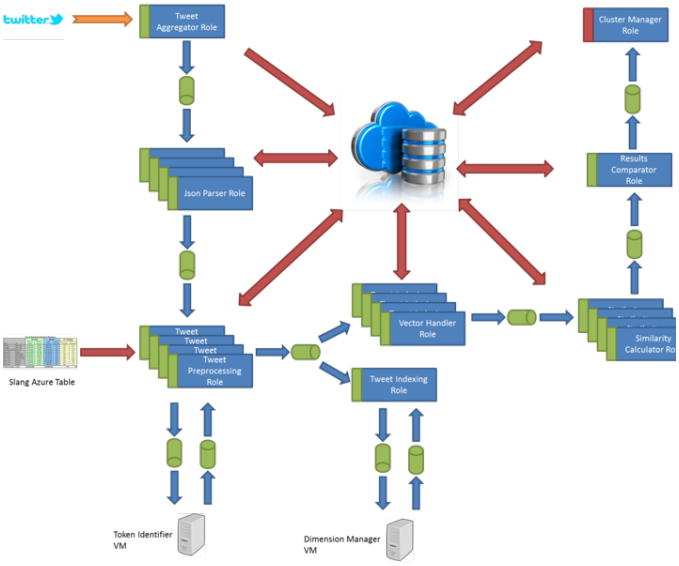

Figure 7. 5th architecture (parallelization)

- *5th system architecture, to offer parallelization in most of the framework's components* (Figure 7). Combining the 3rd and the 4th system architectures, maximum parallelization is targeted, for scalable solutions in real-time geo-located social text clustering. Medium instances of resources are assigned on *Cluster Manager Role* and *Dimension Manager VM,* while small instances are assigned to the other of the system's roles.
-

6. EXPERIMENTATION RESULTS

For testing purposes, the real-time social text clustering approach uses the FSD Corpus[15]. This dataset was collected by the Cross project, which deals with identifying real-time story detection across multiple massive streams. Dataset consists of almost 52 million tweet IDs along with their corresponding user screen names (from Twitter Streaming API). Additionally, labeling process has been applied on a subset of the dataset to categorize the tweets on a distinct topic. This subset consists of 3034 tweets which were tagged as being on-topic for one of the 27 topics defined.

Services	System architectures				
	1st	2nd	3rd	4th	5th
Tweet Preprocessing Role	1x1 Core 1x1.75GB RAM	4x1 Core 4x1.75 GB RAM	3x1 Core 3x1.75 GB RAM	2x1 Core 2x1.75 GB RAM	2x1 Core 2x1.75 GB RAM
Tweet Indexing Role	1x1 Core 1x1.75GB RAM	1x1 Core 1x1.75GB RAM	1x1 Core 1x1.75 GB RAM	1x1 Core 1x1.75 GB RAM	1x1 Core 1x1.75 GB RAM
Tweet Clustering Role	1x2 Cores 1x3.5 GB Ram	1x2 Cores 1x3.5GB Ram	1x2 Cores 1x3.5 GB Ram	-	-
Results Comparator Role	-	-	-	1x1 Core 1x1.75 GB RAM	1x1 Core 1x1.75 GB RAM
Cluster Manager Role	-	-	-	1x2 Cores 1x3.5 GB Ram	1x2 Cores 1x3.5 GB Ram
Dimension Manager VM	1x4 Cores 1x7GB Ram	1x4 Cores 1x7 GB Ram	1x4 Cores 1x7GB Ram	1x4 Cores 1x7GB Ram	1x4 Cores 1x7GB Ram
Total Number of Resources	10 Cores 17.5 GB Ram	17 Cores 29.75 GB RAM	20 Cores 35 GB Ram	20 Cores 35 GB Ram	20 Cores 35 GB Ram

Table 2. Experimentation technical details

[15] http://demeter.inf.ed.ac.uk/cross/index.html

Great emphasis is given on the computer architectures used to apply our experiments, evaluating the benefits and the limitations of each architecture. The same experiments are conducted over the different proposed (Section 5) cloud-computing architectures and also on single node solutions. Details for each architecture's services technical details are summarized in Table 2.

The initial experimentation focused on the two computing infrastructures paradigm and we stress-tested the proposed framework with a dimensionality reduction technique, for both cloud and traditional computing infrastructures.

Figure 8 showcases the fact that Windows Azure cloud computing infrastructure performed better when compared with the single-node infrastructure, and it was noticed that regarding its high computational cost and its parallelization demands, cloud computing infrastructure has provided the appropriate resources to efficiently support such a demanding and scalable solution.

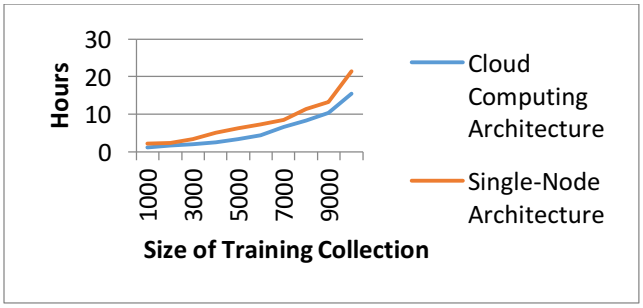

Figure 8. Computer Infrastructures Comparison

The basic workflow of the experiments and the different parameters of our proposed methodology, including the appropriate tweets' selection and the rate of the dimensionality reduction performed, have been stress-tested under several scenarios. Here, we focus on the experimentation results which are relevant in the proposed architectures emphasis and potential on geo-located time dependent knowledge extraction from social text threads.

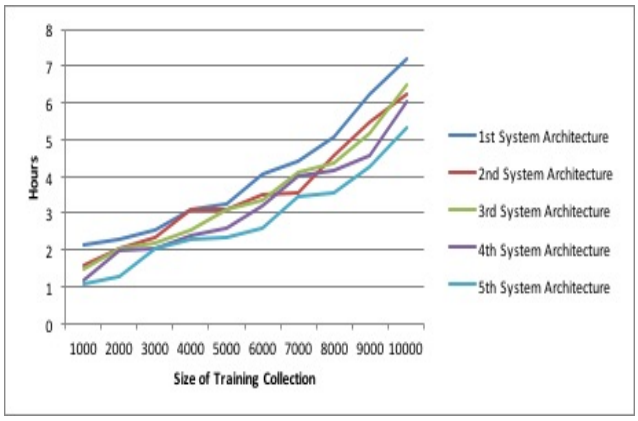

Figure 9. Azure Hours of Execution Time - No dimensionality Reduction

Figure 9 summarizes the proposed cloud computing architectures (details at Section 5), the execution time of each system architecture, when no dimensionality reduction is performed. As depicted in this figure, the 1st System Architecture demands more hours than the other system architectures to accomplish the experiment. Instead, when the parallelization is occurred on the other system architectures, the experiments' execution time is diminishing. In the 5th System architecture, where maximum parallelization is provided in most of its services, the total execution time is lower than the others. The scalability obstacle, though, is confronted with the support of the parallelization on the systems' services.

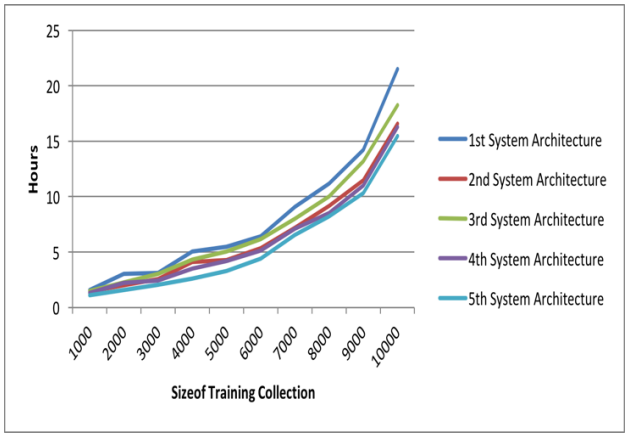

Figure 10. Azure Hours of Execution Time - 25% latent factors dimensionality reduction

The dimensionality reduction approach has been further applied on the cloud computing architectures, the time execution of each system architecture is shown in Figure 10 . Similar to the previous approach, 1st System Architecture requires the most execution time than the other architectures, while the 5th System Architecture provides a more scalable solution.

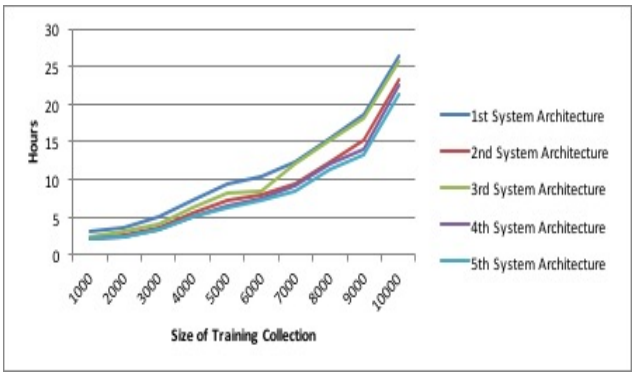

Figure 11. Single-Node Hours of Execution Time - 25% latent factors dimensionality reduction

Finally, the difference among the proposed system architectures when the dimensionality reduction technique is applied is stressed-tested, by considering that parallelization is provided on the distinct systems' services (Figure 11). The total execution time is shown to be decreased, since the tweets' vector representation is reduced, and the similarity calculation along with the existing clusters is not of high-computational cost. Such cases exhibit no bottlenecks in the clustering process and their applicability in geo-located time dependent use cases is promising.

7. CONCLUSIONS AND FUTURE WORK

This proposed work has focused on utilizing the cloud computing paradigm, by the use of VENUS-C enactment services, and under data-dependent jobs, which stress-tested five different proposed architectures. Scalability issues are addressed by dedicated components which enable dedicated Cloud Tables for monitoring clusters' "activity" and for updating their states. Dynamics can thus be detected under clusters' representations at parameterized time intervals, with geo-spotted trends identification.

Future work is planned around the following axes: i) the fine-tuning of the clustering and trend detection algorithm and the experimental evaluation of results, ii) the implementation of a shard-based distributed Indexing service since for the time being the service for each type of resource is deployed on a single instance, iii) measuring the system's performance for different design parameters, iv) creating a web-based user interface (hosted either on Cloud or on premises) for visualization of the detected real-time trends and trends' analytics, and v) stress-test the proposed framework and architectures under different cities datasets and requirements.

The benefit that Cloud4Trends and different architectures offer is that they verify that Cloud-based architectures constitute a viable solution for online social web geo-located and time-related data mining applications. In particular, the proposed work enables massive data analysis at a distributed setting, thus reducing the prerequisite for real-time applications data processing time. At the same time is allows easier testing of new scenarios, achieving high-quality results under different demands of cloud-based architectures which can improve application capability sharing. Overall, these are benefit for both researchers and entrepreneurs in actual real use cases.

In summary, the proposed framework (as realized by the Cloud4Trends experiment), demonstrates that porting trend detection into the Cloud is a very suitable solution considering the challenges posed by the data, geo-location features and time intensive processes involved in online collection and analysis of large and evolving geo-located Web 2.0 datasets.

ACKNOWLEDGMENTS

This work is partly funded and realized within the Cloud4Trends pilot project of the VENUS-C project. VENUS-C (Virtual multidisciplinary EnviroNments USing Cloud infrastructures) is co-funded by the GÉANT and e-Infrastructures Unit, DG Information Society and Media, European Commission. The authors thank the anonymous reviewers for their valuable comments and suggestions.

REFERENCES

[1] Antoniades, D. and C. Dovrolis. "Co-evolutionary dynamics in social networks: A case study of Twitter." Computational Social Networks2.1 (2015): 1-21.

[2] N. S. Glance, M. Hurst, and T. Tomokiyo: BlogPulse: Automated Trend Discovery for Weblogs. WWW Conference. 2004.

[3] M. Uchida, N. Shibata, and S. Shirayama: Identification and Visualization of Emerging Trend from Blogosphere, proceedings of International Conference on Weblogs and Social Media (ICWSM), pp. 305-306, 2007.

[4] M. Mathioudakis, N. Bansal, and N. Koudas: Identifying, attributing and describing spatial bursts. Proc. VLDB Endow. 3, 1-2 (September 2010), 1091-1102. 2010.

[5] J. Sankaranarayanan, H. Samet, B. E. Teitler, M. D. Lieberman, and J. Sperling: TwitterStand: news in tweets. In Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS '09). ACM, 42-51. 2009.

[6] M. Mathioudakis and N. Koudas: TwitterMonitor: trend detection over the twitter stream. In Proceedings of the 2010 international conference on Management of data (SIGMOD '10). ACM, 1155-1158. 2010.

[7] V. Koutsonikola, A. Vakali, E. Giannakidou, I. Kompatsiaris: Clustering Users of a Social Tagging System: A Topic and Time Based Approach. WISE 2009: 75-86. 2009.

[8] I. Livenson and E. Laure: Towards transparent integration of heterogeneous cloud storage platforms. In DIDC '11. ACM, 27-34. 2011.

[9] Giatsoglou, Maria, and Athena Vakali. "Capturing social data evolution using graph clustering." *IEEE Internet Computing* (2013): 74-79.

[10] Spärck Jones, Karen: A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation 28 (1): 11–21. 1972.

[11] D. Lezzi, R. Rafanell, F. Lordan, E. Tejedor, R.M. Badia: COMPSs in the VENUS-C Platform: enabling e-Science applications on the Cloud. Proceedings of 4th Iberian Grid Infrastructure Conference, 2011, 73-84.

[12] Yogesh Simmhan, Catharine van Ingen, Girish Subramanian, and Jie Li: Bridging the Gap between Desktop and the Cloud for eScience Applications. In Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing (CLOUD '10). IEEE Computer Society, 474-481. 2010.

[13] Bao, Jie, et al. "Recommendations in location-based social networks: a survey." GeoInformatica 19.3 (2015): 525-565.

[14] Foster I. et. al:OGSA Basic Execution Service Version 1.0. Grid Forum Document GFD-RP. 108. 8 August 2007.

[15] Savva A (Editor). Job Submission Description Language (JSDL) Specification, Version 1.0. Grid Forum Document GFD-R.056. 7 November 2005.

[16] SNIA CDMI: http://www.snia.org/cdmi

[17] A. Vakali, M. Giatsoglou, and S. Antaris. Social networking trends and dynamics detection via a cloud-based framework design. WWW (Companion Volume), pages 1213-1220. ACM, WWW 2012.

[18] Clarg E., Araki K. (2011). Text Normalization in Social Media: Progress, Problems and Applications for a Pre-Processing System of Casual English. Special issue of Computational Linguistics and Related Fields. pp 2 – 11

[19] Reuter T., Cimiano P., Drumond L., Buza K., Schmidt-Thieme L. (2011). Scalable Event-based Clustering of Social Media via Record Linkage Techniques. In Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media.

[20] Whang J.J., Sui Xin, Dhillon I.S. (2012). Scalable and Memory-Efficient Clustering of Large-Scale Social Networks. In Proceedings

of the 12th International Conference on Data Mining. pp. 705 – 714

[21] Yerva S. R., Jeung H., Aberer K. (2012). Cloud based Social and Sensor Data Fusion. Proceedings of the 15th International Conference on Information Fusion. pp. 2494 – 2501.

[22] Sysomos MAP social research engine : https://sysomos.com/

[23] Radian6 Buddy Media Social Studio : http://www.exacttarget.com/