Tengu: an Experimentation Platform for Big data Applications

Thomas Vanhove, Gregory Van Seghbroeck, Tim Wauters, Filip De Turck, Brecht Vermeulen, Piet Demeester
Department of Information Technology, Ghent University - iMinds
Gaston Crommenlaan 8/201, 9050 Gent, Belgium
Email: thomas.vanhove@intec.ugent.be

***Abstract*—Big data applications have stringent service requirements for scalability and fault-tolerance and involve high volumes of data, high processing speeds and large varieties of database technologies. In order to test big data management solutions, large experimentation facilities are needed, which are expensive in terms of both resource cost and configuration time. This paper presents Tengu, an experimentation platform for big data applications that can automatically be instantiated on GENI (US federation of testbeds) and Fed4FIRE (EU federation of testbeds) compatible testbeds. Tengu allows for automatic deployments of several data processing, storage and cloud technologies, including Hadoop, Storm and OpenStack. The paper discusses the Tengu architecture, the Tengu-as-a-service approach and a demonstration of an automated instantiation of the Tengu experimentation suite on the Virtual Wall, a large-scale Emulab testbed at the iMinds research institute in Europe.**

I. Introduction

Big data applications require scalable and fault-tolerant frameworks to handle the high volumes of data and guarantee high processing speeds. A plethora of technologies have been invented to support efficient analysis, processing and storage of big data sets for many different scenarios [1], [2], [3], [4]. Nevertheless, these technologies are often cluster-based in order to meet the scalability and fault-tolerance requirements of the applications. Testing these applications therefore requires large experimentation facilities. These setups are expensive in resource cost, but the clustered setup of the majority of these technologies complicates and increases the configuration time.

We propose Tengu, a new experimentation platform deployed on the Virtual Wall. The iLab.t Virtual Wall facility[1], based on the Emulab[2]/GENI platform, is a generic test environment for advanced network, distributed software and service evaluation, and supports scalability research. Tengu provides an automatic setup and deployment of big data analysis frameworks (e.g. Hadoop and Storm), SQL data bases (e.g. MySQL), data stores (e.g. Cassandra and ElasticSearch) and other cloud technologies, such as OpenStack. Furthermore, the platform offers several unique features: the Lambda architecture [5], which combines batch and stream big data analysis frameworks, and live data store transformations [6].

Many ground-breaking novel applications and services cover multiple innovation areas. Therefore, the need for these solutions to be tested on cross-domain experimentation facilities with both novel infrastructure technologies and newly emerging service platforms is rising. The Fed4FIRE project[3] aims at federating otherwise isolated experimentation facilities from the FIRE[4] community in order to foster synergies between research areas [7]. As the Virtual Wall facility is part of this federation, the Tengu platform is interconnected with other testbeds offering wired, wireless and sensor networks, SDN and OpenFlow technologies, cloud computing and smart city services, which ensures that researchers can perform experiments across the boundaries of the big data area.

Fed4FIRE offers various forms of federation. While testbeds can be merely *associated* with the federation (i.e. listed on the website with links to contact information, documentation and tutorials), the primary options are to be integrated through *advanced* or *light* federation.

- Light: access to the testbeds services is realized by exposing a Web-based API. This option does not allow full control over the individual testbed resources, but ensures unified access to experimenters.
- Advanced: the testbed is fully integrated in the federation so that experimenters can interact with their experiment during all stages of the experiments life cycle (resource selection, instantiation, control, monitoring, etc.). This option requires the implementation of the Federation Aggregate Manager (AM) API on top of the testbed.

Tengu is federated through the *light* option. Its RESTful API for example allows for the instantiation of the platform through simple HTTP POST commands. Tengu in its turn relies on the same client tools offered by Fed4FIRE. The POST command to instantiate the platform creates an RSpec (a description of the requested resources), which is then deployed using the jFed client tool. As a consequence, the same tools can be used to for example visualize the setup through the jFed GUI [5] and interact with the underlying resources, if required.

The remainder of this paper is organized as follows: Section II illustrates the architecture of the Tengu platform, consisting of the core platform and its front end. The service platform, containing the RESTful API, is detailed in Section III. Section IV lays out a use case of the Tengu platform

[1]http://ilabt.iminds.be
[2]https://www.emulab.net
[3]http://www.fed4fire.eu
[4]http://www.ict-fire.eu
[5]http://jfed.iminds.be/

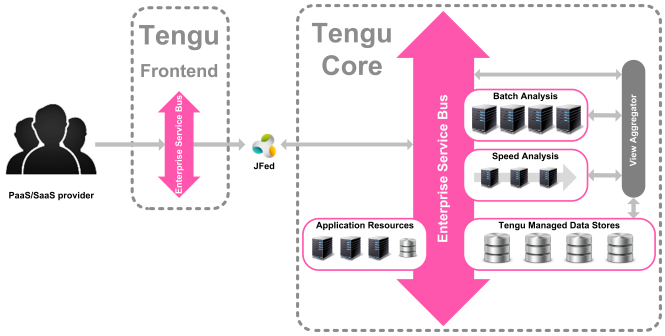

Fig. 1. General overview of the Tengu architecture

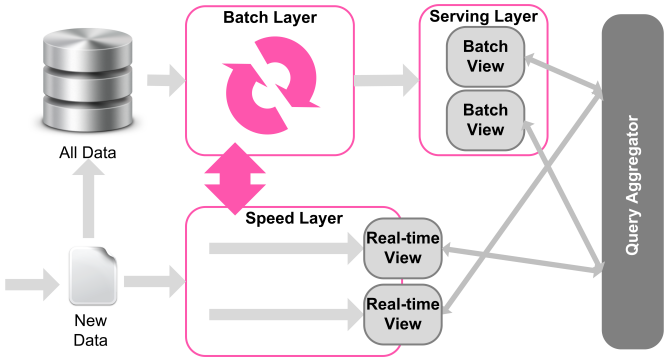

Fig. 2. Conceptual overview of the Lambda architecture

related to social network monitoring, followed by a description of the demo in Section V. Finally, related work is discussed in Section VI, while Section VII concludes this paper and offers several insights towards future work.

II. Architecture

The overall Tengu architecture has two distinct stages as illustrated in Figure 1. The first stage consists of the front end which translates RESTful method calls from the experimenters into an RSpec of a new Tengu core instantiation. Depending on the types of nodes defined in the RSpec, it can then be deployed by jFed on several testbeds of the Fed4FIRE federation. Once the deployment is finished, the second stage starts. The configuration management software makes sure all different components are correctly configured and interconnected, resulting in a new full-featured Tengu core setup.

In the following subsections we will dive deeper into the different stages and the involved architectural components.

A. Tengu core setup

The core setup of Tengu can again be divided into three main parts: a computational unit, data stores managed by Tengu and the application specific resource pool. An Enterprise Service Bus manages the communication between these different parts. It acts as a middleware software shielding the different components from each others specific implementation and routes data and messages between them.

1) Lambda architecture: The computational unit that is provided to all users of the Tengu platform is based on the concept of the Lambda architecture [5]. This concept, coined by Nathan Marz, combines two current approaches to big data analysis: batch and real-time or stream data processing. Batch data analysis frameworks analyze entire big data sets and create a view on this data set based on the implemented algorithms. However, specific types of applications, processing and analyzing data from sensor networks, social media, and network monitoring applications, generate data streams, causing results provided by a batch analysis to be always out of sync with the real-life application as new data is continuously created during the batch analysis run. This cultivated the need for (near) real-time or stream processing [3]. These stream processing technologies provide incremental updates to their results whenever new data is received, but in doing so they lack a general overview of the entire data set.

The Lambda architecture, depicted in Figure 2, thus is a specific hybrid approach for big data analysis leveraging the computing power of batch processing in a batch layer with the responsiveness of real-time computing system in a speed layer. The batch layer provides a batch view on the entire data set, while new data is instantly analyzed by the speed layer, yielding a speed view on the most recent data. Additionally, new data is also added to the data set so it can be analyzed by the batch layer in a subsequent run. Once this run is completed, the batch view is updated while any redundant information is removed from the speed view. Querying the views of an applications big data set will therefore always include the aggregation of information in both the batch and speed view.

This entire computational unit is provided as a service to the applications on the Tengu platform. However, it is important to note that while the entire Lambda architecture can be provided, both batch and speed layer can also be used separate of each other. In this context the ESB manages the coordination between both layers, routing messages to the correct layer and data stores.

2) Tengu managed data stores: In the big data domain, technologies for storing data are designed aimed to scale horizontally, providing read/write operations distributed over many servers. This yields a new category of storage systems called NoSQL data stores [4]. As many different data stores exist today, each with their own (dis)advantages in specific scenarios, the Tengu platform offers many different data store solutions. This allows applications to use the optimal data store or even multiple data stores for their data.

Long-term experimental applications tend to evolve with frequent updates and changing user numbers, rendering the once optimal data store no longer optimal. Hence Tengu provides additional features for data stores managed by the platform, such as a live transformation between two data stores [6]. This transformation is executed using the same Lambda architecture provided to the applications. A snapshot (schema and data) of the original data store is transformed in the batch layer, while the speed layer transforms any new

queries that arrive after the snapshot is taken. Once the batch layer is finished, a new data store is set up with the transformed schema and data, then updated with the queries that were transformed by the speed layer, after which a turnover is initiated to use the new transformed data store. During this entire process, the application still queries the original data store as to eliminate any downtime.

Normally this process would also require some form of change in the application code, using the new query language. However, the ESB shields the application from its data store and through the continuous transformation of queries in the speed layer, an application can still query in the language of the original data store, even though it has been transformed [6].

3) Application specific resource pool: This resource pool contains several servers for the deployment of the applications on the Tengu platform. For example, while an application takes advantage of the computational unit of the Tengu platform, it can still manage its own data store outside of the Tengu environment. Additionally, applications might require specific resources such as a Tomcat server. These can also be provided in this pool. To better utilize the available infrastructure, the application specific resource pool is set up as a private cloud.

B. Tengu front end

The Tengu front end consists of a RESTful API component and jFed. The RESTful API component transforms incoming user requests for platform instantiations to an RSpec that can be deployed by jFed on testbeds in the Fed4FIRE federation. A user in this context is an experimenter who either wants to experiment with Tengu as a Platform-as-a-Service for big data applications or who wants to use Tengu to experiment with an application for big data analysis. The RESTful API has POST methods to create and deploy new Tengu core instances and GET methods to receive important information about a particular instance. It is discussed in more detail in the next section.

Deployment of the Tengu core instances is handled by jFed, as is retrieving the state information of the deployment (both general state as for example the used nodes). Using jFed has many advantages over using the underlying Fed4FIRE APIs (Aggregate Manager, User and Slice API). jFed validates the provided RSpec, not only in formatting errors, but for example also the used slice identifiers. The tool also combines many individual API calls into a single operation, e.g. requesting user information, access control checks, allocating resources and state handling. The only drawback of working with jFed, is that it does not come with an easy RESTful interface (or any other remote interface). To make the interaction with the Tengu RESTful API easier a RESTful wrapper was created around the different jFed commands.

III. Service Platform

A. RESTful API

RESTful API 1: `POST /tengu/core`

This API call allows to asynchronously create and deploy a new Tengu core instance. It has three mandatory query parameters: `testbed`, `snodes` and `hnodes`. The `testbed` parameter is to specify on which Fed4FIRE testbed the Tengu core instance has to be deployed. The `snodes` and `hnodes` parameters define the size of the Storm cluster and Hadoop cluster, respectively. The response of this POST includes a unique identifier, more specifically a UUID. This identifier can be used to retrieve the information about the Tengu core instance that is being deployed.

Listing 1. The message format of a response to a POST call

```
<ten:tengu ... >
 <ten:platform>
  <ten:id>{uuid}</ten:id>
  <lnk:link method="get" href="/tengu/{uuid}" />
 </ten:platform>
</ten:tengu>
```

Listing 1 shows response's message format. Notice the usage of XHTML links, this to easily show how an experimenter can proceed next.

RESTful API 2: `GET /tengu/{uuid}`

To retrieve information about an individual Tengu core instance, a user can perform this RESTful method call. The only thing that has to be provided, is the unique identifier (`uuid`) of the instance. The content of the response depends on the current state of the instance. At the moment we have three states: unknown (the deployment is not finished yet), ready (deployment is done and all components are correctly configured) and failed. If the Tengu core instance is fully deployed, the response also includes links to the important Tengu components (e.g. the Hadoop Job history, the HDFS namenode, the Storm web UI, the OpenStack horizon web front end, etc.)

Listing 2. The message format of the response for a GET call

```
<ten:tengu ... >
   <ten:platform>
   <ten:id>{uuid}</ten:id>
   <ten:id>{UNKNOWN|READY|FAILED}</ten:id>
   <lnk:link method="..." rel="..." href="..." /> *
   </ten:platform>
</ten:tengu>
```

The Tengu core setup already includes a wide variety of technologies (e.g. Hadoop, Storm and OpenStack). By using the already available RSpec generation process, deployment with jFed and configuration with Chef, it is very straightforward to also provide separate API calls to set up these individual environments. This allows the user to deploy for example only an OpenStack environment or a ready to use Hadoop cluster. The extra API calls are presented in Table I together with the necessary query parameters. The response to such a call is the same as shown in Listing 2. Information requests for these specialized environments also use the GET request (RESTful API 2), the responses will of course show a different set of links, depending on the deployed environment.

B. RSpec generation and deployment

The RESTful API is implemented as part of an Enterprise Service Bus (ESB), more particularly the WSO2 ESB [6]. The

[6]http://wso2.com/products/enterprise-service-bus

RESTful API calls	Description
`POST /tengu/hadoop`	deploy a Hadoop cluster
`POST /tengu/storm`	deploy a Storm cluster
`POST /tengu/openstack`	deploy an OpenStack cluster

TABLE I
SPECIAL RESTFUL API CALLS TO SET UP A SPECIFIC CLUSTER. ALL CALLS HAVE THE SAME SET OF QUERY PARAMETERS: NODES, TESTBED. THE CALLS WILL DEPLOY A CLUSTER OF SIZE {NODES} ON THE SPECIFIED {TESTBED}

main reason for choosing this software component is not only its straightforward manner to define RESTful APIs, but its routing capabilities. It is easy to configure the ESB so it, provided certain parameters, execute a particular workflow. During this workflow it is also possible to change the incoming and outgoing messages. It is exactly this process that is used to construct the RSpec.

The first step in the RSpec creation process, starts with a templated version of the RSpec. The parameters provided with the POST RESTful call are integrated in this template. For the variable clusters (currently Hadoop, Storm, OpenStack), the template has included a placeholder. Using XSLT transformations this placeholder is changed into correct RSpec node information.

C. Deployment scripts

The RSpec refers to a script that will install a Chef server and workstation on a separate node. Chef is a configuration management software automating the build, deployment and management of the Tengu infrastructure. All technologies in the Tengu platform are defined through cookbooks and recipes, which are basically idempotent step-by-step installation scripts. Multiple executions of these scripts will therefore never change the outcome; in most cases a running framework or service.

Based on the nodes defined in the RSpec, the script deploys the cookbooks for the corresponding technologies and executes them on the correct nodes. For example, if a master node is requested in the setup, together with one or several Hadoop slave nodes (hnode), a Hadoop cluster will be deployed on these nodes. If these nodes are not present, the Hadoop cookbook will not be deployed. This modular approach to building and deploying the Tengu platform, together with Chef, allows for a flexible setup of the platform, tailored to the requirements of the experimenter.

Current available technologies include Hadoop and Storm for batch and speed layer respectively. Three data stores are already supported: MySQL, Cassandra and ElasticSearch. Other supported technologies include Tomcat, Zookeeper and Kafka. Nonetheless, the chosen approach with Chef and the ESB allows an easy integration of new technologies for the batch/speed layer and data stores.

D. Application deployment

Applications for the Tengu platform typically exist of several combinations of batch jobs and speed jobs, more specifically in the current setup of the Tengu platform, these are Hadoop MapReduce jobs and Storm topologies. For the application's UI, Tengu currently provides an Apache Tomcat Server and the application's specific meta data can be stored in a separate data store. Exactly which components are required is application specific, but to ease the setup of the actual application components, Tengu uses Chef to assist the experimenter. Consequently, the platform can be easily extended with all software components already provided through Chef's Supermarket [7].

The actual resources that are used by Chef to deploy the necessary application specific software components are part of the OpenStack private cloud set up in Tengu. This means that Chef will create a new virtual machine on the OpenStack cluster and also deploys the correct cookbooks and recipes on this virtual machine. The method of using a virtual environment for application specific resources opens up a lot of possibilities towards multi-tenancy, resource management and optimization, resource isolation, and automation.

IV. USE CASE

The AMiCA (Automatic Monitoring for Cyberspace Applications) project aims to mine relevant social media (blogs, chat rooms, and social networking sites) and collect, analyse, and integrate large amounts of information using text and image analysis. The ultimate goal is to trace harmful content, contact, or conduct in an automatic way. Essentially, a cross-media mining approach is taken that allows to detect risks "on-the-fly". When critical situations are detected (e.g. a very violent communication), alerts can be issued to moderators of the social networking sites.

The AMiCA project leverages the Lambda architecture using the speed layer to get near real-time feedback on developing situations on the Social Network Site (SNS), while the batch layer provides specific views on the entire history of the site. In this use case an example chat conversation between two generic users of a generic SNS is used.

The big data set of the SNS contains the entire history of the relationship of these two users, including their chat conversations. During the execution of the batch layer, the speed layer provides an analysis of the most recent incoming chat messages since the batch layer started its execution. It is clear that the speed layer does not have access to any other messages of the conversation other than the message provided at that specific moment in time. This limits the analytical power of the speed layer, but it can still provide some valuable feedback in terms of language used, picture or video/audio analysis [8], [9]. The information retrieved from this limited analysis is used in two ways. Firstly, querying the results requires an intelligent aggregation of the information in both the batch and speed view. For example, a single aggressive comment might not mean anything if the relationship has not shown any or limited signs of aggression in the past. However, when this fits into a relationship of repeated aggression, additional steps need to be taken (e.g. blocking the account of the aggressor). Secondly, in extreme cases (e.g. very violent

[7] https://supermarket.chef.io/

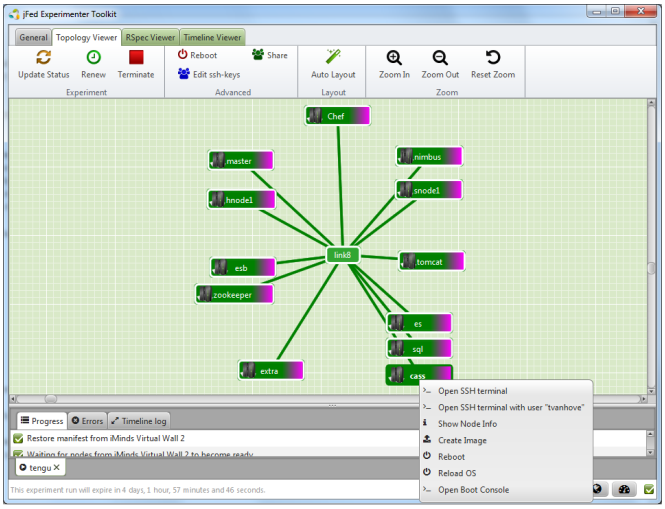

Fig. 3. Screenshot of the JFed GUI showing an instantiation of Tengu

Fig. 4. The iLab.t Virtual Wall facility

language or graphics) the information immediately triggers an alert for the moderators of the SNS.

The other partners in this project provide components for text, image or video analysis and do not need a thorough knowledge of the setup and deployment of big data analysis frameworks. The Tengu service platform allows them to set up these frameworks for their short and long term experimenting needs.

V. Demo

The demo of the Tengu platform illustrates the ease of use for the setup of the platform and a comparison of the use case application running on a Tengu setup with varying cluster sizes. Figure 3 shows the JFed GUI which allows for an easy interaction with all the resources of the Tengu platform once it has been instantiated. JFed is also responsible for the authentication and of experimenters on the different testbeds in the Fed4FIRE federation. The Tengu platform deployed in Figure 3 utilizes resources from the Virtual Wall, depicted in Figure 4.

In the demo Tengu setups with different cluster sizes of both batch and speed layer are deployed on the Virtual Wall. With experiments running on these different setups, several key points of interest are highlighted, such as number of messages processed per minute in the batch and speed layer, and number of queries handled per minute. A clear seperation is made between queries on batch views and speed views. The demo also clearly shows the possibility of experiment repeatability and reproducibility even though in between tests the nodes of the Virtual Wall are released for other experiments.

The next step is to dynamically increase the cluster size of any depending technology. While some preliminary tests have already been conducted dynamically increasing the size of the Hadoop cluster, this functionality has yet to be integrated into the Tengu platform and is therefore part of future work. This requires a monitoring framework within the platform, which could also provide statistics about application performance to the experimenters. Additionally, these performance statistics could also contain valuable information regarding the views and/or data stores: if a certain view/data store is no longer performing well relative to certain performance constraints, a transformed as described in Section II-A2 could be automatically initiated.

VI. Related work

This Section discusses related work, similar setups and their main differences with the Tengu platform.

Many big data analysis frameworks are already offered as a service by the large cloud providers such as Amazon [8], Google [9] and Microsoft [10]. These frameworks are tightly integrated within their platform environment giving customers access to many other services as well for storage, networking, and elastic scaling among others. They do however often require application developers to adopt the in-house technologies, creating a vendor lock-in. Once embedded in the ecosystem, moving to another provider would require a large investment of time and resources. Tengu eliminates this vendor lock-in by using open source technologies that are already available in both research and industry. Moreover, the ESB middleware, shielding an application from its data store(s), and the live data store transformation limit application changes altogether.

HPPC Systems [10] offers a massive open source parallel computing platform that is also built around the principles of the Lambda architecture. They have a dedicated batch and serving layer, called Thor and Roxie respectively. Their speed layer is built up from several components from Thor and Roxie combined with an Apache Kafka consumer plugin. They furthermore allow for incremental updates of views through a concept of superfiles and superkeys. While HPCC Systems provides an interesting approach to big data analysis

[8] http://aws.amazon.com/elasticmapreduce/
[9] https://cloud.google.com/appengine/docs/python/dataprocessing/
[10] http://azure.microsoft.com/en-us/documentation/services/hdinsight/

through the Lambda architecture, the Tengu platform aims to bundle existing technologies, integrated in research and industry, while every the specific combination of technologies is tailored to every application. The WSO2 ESB is the core of the Tengu platform and a necessary part of every setup, but in doing so technologies for batch/speed layer, data stores or other cloud technologies are not fixed. This also preserves the platform-agnostic idea of the Lambda architecture.

With the large amount of SQL and NoSQL data stores, persistence frameworks are trying to eliminate the complexity of these different technologies by creating an abstract layer on top of the data stores. Examples like Hibernate ORM/OGM [11], PlayORM [12] and Kundera [13] have already found their way into many projects. Through a unified querying language and schema all supported data stores can be queried. The application is shielded from the complexity of the different data stores but in most cases the querying and schema language are newly introduced languages by the developers of the persistence framework. Tengu again introduces no new querying language for the communication with the data stores as many developers are already familiar with one or more data stores. The application can use the querying language and schema representation it is most familiar with, the continuous transformation in the speed layer will transform these queries into the querying language of the actual data store. This also allows for experimenters to easily try out new data stores and evaluate what this could mean for their applications.

VII. Conclusion and Future work

This paper presented the Tengu platform, a new experimentation platform part of the Fed4FIRE federation. It allows for the automatic setup and deployment of different big data analysis frameworks, SQL/NoSQL data stores and other cloud technologies.

As mentioned in Section IV, the Tengu platform still requires a service that can autonomously decide to rescale the clusters of the different technologies (analysis frameworks and data stores). This requires a monitoring system tracking the performance of all the clusters. The monitoring information could also prove valuable for deciding when to transform between data stores: when the query response time in a certain data store no longer meets the requirements, a transformation to a more appropriate data store. Finally, this monitoring information could be relayed to the users directly as well, providing them with detailed information about the performance of their entire application.

While the RESTful API currently returns all important links to the different technologies, in the future it would be interesting to include an automated application deployment. Users would then be able to pass a bundle of their entire application to the service, which would then be deployed on the various parts of the Tengu core platform.

[11] http://hibernate.org/

[12] http://buffalosw.com/wiki/playorm-documentation/

[13] https://github.com/impetus-opensource/Kundera

Acknowledgment

This work was partly carried out with the support of the Fed4FIRE project (”Federation for FIRE”), an integrated project funded by the European Commission through the 7th ICT Framework Programme (318389), and the AMiCA (Automatic Monitoring for Cyberspace Applications) project, funded by IWT (Institute for the Promotion of Innovation through Science and Technology in Flanders) (120007).

References

[1] J. Dean and S. Ghemawat, “Mapreduce: Simplified data processing on large clusters,” *Commun. ACM*, vol. 51, pp. 107–113, Jan. 2008.

[2] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, “Spark: Cluster computing with working sets,” in *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud’10, (Berkeley, CA, USA), pp. 10–10, USENIX Association, 2010.

[3] J. Gama, *Knowledge Discovery from Data Streams*. Chapman & Hall/CRC, 2010.

[4] R. Cattell, “Scalable SQL and NoSQL Data Stores,” *SIGMOD Rec.*, vol. 39, pp. 12–27, May 2011.

[5] N. Marz and J. Warren, *Big Data: Principles and best practices of scalable realtime data systems*. Greenwich, CT, USA: Manning Publications Co., 2014. (Early Access Program).

[6] T. Vanhove, G. Van Seghbroeck, T. Wauters, and F. De Turck, “Live Datastore Transformation for optimizing Big Data applications in Cloud Environments,” in *Proceedings of the 2015 IEEE/IFIP International Symposium on Integrated Network Management (IM 2015)*, may 2015.

[7] T. Wauters, B. Vermeulen, W. Vandenberghe, P. Demeester, S. Taylor, L. Baron, M. Smirnov, Y. Al-Hazmi, A. Willner, M. Sawyer, *et al.*, “Federation of internet experimentation facilities: architecture and implementation,” in *European Conference on Networks and Communications (EuCNC 2014)*, pp. 1–5, 2014.

[8] B. Desmet and V. Hoste, “Recognising suicidal messages in dutch social media,” in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC’14)*, (Reykjavik, Iceland), European Language Resources Association (ELRA), may 2014.

[9] B. Verhoeven, J. Soler Company, and W. Daelemans, “Evaluating content-independent features for personality recognition,” in *Proceedings of the 2014 ACM Multi Media on Workshop on Computational Personality Recognition*, WCPR ’14, (New York, NY, USA), pp. 7–10, ACM, 2014.

[10] A. M. Middleton, “Introduction to HPCC (High-Performance Computing Cluster).” White Paper, may 2011.