A multi-layer software architecture framework for adaptive real-time analytics

Athena Vakali
Informatics Department
Aristotle University, Greece
avakali@csd.auth.gr

Paschalis Korosoglou and Pavlos Daoglou
IT Center
Aristotle University, Greece
{pkoro,pdaog}@it.auth.gr

***Abstract*—Highly distributed applications dominate today's software industry posing new challenges for novel software architectures capable of supporting real time processing and analytics. The proposed framework, so called REAλICS, is motivated by the fact that the demand for aggregating current and past big data streams requires new software methodologies, platforms and services. The proposed framework is designed to tackle with data intensive problems in real time environments, via services built dynamically under a fully scalable and elastic Lambda based architecture. REAλICS proposes a multi-layer software platform, based on the lambda architecture paradigm, for aggregating and synchronizing real time and batch processing. The proposed software layers and adaptive components support quality of experience, along with community driven software development. Flexibility and elasticity are targeted by hiding the complexity of bootstrapping and maintaining a multi level architecture, upon which the end user can drive queries over input data streams. REAλICS proposes a flexible and extensible software architecture that can capture users preference at the front-end and adapt the appropriate distributed technologies and processes at the back-end. Such a model enables real time analytics in large-scale data driven cloud-based systems.**

Keywords—software architectures; real time data management; big data analytics; cloud based services.

I. INTRODUCTION

The emerging big data era demands sophisticated models for data summarization, mining and analytics. Nowadays, such analytics have become the drivers for substantially improved decision-making and for the next generation of products and services. Both real-time and near real-time responses in many multi domain use cases (network fault prediction, security threat prediction, fraud analytics etc) require data intensive and data driven analytics [1].

In this context, it is important to design and deploy data-driven software architectures, while introducing new abstractions for data pipelining in distributed and heterogeneous cloud environments. Moreover, real-time execution of big data analytics requires addressing and overcoming network performance constraints when large amounts of data are transferred over cloud and distributed architectures [2], [3]. Big Data management tools have progressed and reached adequate maturity levels by now, but they do exhibit performance issues, especially in real time scenarios. The current abundance of big data software releases and features, have highlighted popular software solutions, such as the Apache Hadoop[1] and its associated components. These technologies have proven their value and have become dominant in the recent years for handling big data streams [4]. Despite their capabilities, such big data software releases, are mostly suited for batch processing needs and they are often challenged with slow performance on real-time analytics [5]. It is now needed to offer novel software solutions adaptive, tunable and elastic. Such solutions should be able to aggregate past data knowledge with real time data generated data streams since several use cases from different domains require such tasks.

The proposed framework so called *REAλICS* (*REA*l time *λ* architecture analytICS), is highly motivated by the fact that in today's highly distributed applications, processing of real time (multi-streamed) big data has become a bottleneck. REAλICS is inspired by such challenges and the need to integrate data intensive software solutions, towards supporting *continuous and real-time data and information processing*. Different use cases can leverage such integration and platform's capabilities as the ones of ENCASE project[2] which targets a user-centric architecture of distinct data integration services. Data processing architectures designed to handle such latencies and handle massive quantities of data by taking advantage of both batch and stream processing methods have already appeared. A popular case is the Lambda[3] architecture designed to balance performance, throughput, and fault-tolerance. Both batch and real-time stream processing are utilized on a layered approach to provide comprehensive and accurate views of batch and online data [6].

In REAλICS emphasis is placed on the real time execution of big data analytics, under a flexible software architecture, which aims at facilitating big data analytics methods and tools implementations. In enabling such analytics, REAλICS deals with both current and past data streams, which are aggregated, analyzed and visualized to provide qualitative information in response to users' queries. Quality of results is due to the aggregation of live along with past data sources, such as

[1] Apache Hadoop http://hadoop.apache.org/

[2] "Enhancing security and privacy in the Social Web" http://encase.socialcomputing.eu/

[3] Lambda Architecture : A repository dedicated to the Lambda Architecture http://lambda-architecture.net/

historical data, which are expected to reveal hidden inter-relationships, phenomena, and reciprocities on the queried distributed data sets.

REAλICS is currently under a prototyping phase. An assessment of the underlying key technologies to be used has been completed and the development of the necessary workflows (to dynamically provide real time based services upon the IaaS layers) is under way and being constantly validated through software testing mechanisms (i.e. Unit tests, integration tests etc). Meanwhile, an Intrusion Detection use case (of significant importance for the IT Center of the Aristotle University of Thessaloniki), is being developed upon the tools that shall be made available through REAλICS, so as to assist in the results validation and services dissemination phases.

REAλICS aims at improving entities, system components and user experience at all data and software layers. It anticipates to encapsulate building and enabling end-user dynamic delivery of content under a fully scalable multi-level software architecture for tackling data intensive problems and optimize system layer components performance by dynamic provisioning of services and by hiding the complexity of bootstrapping and maintaining an underlying multi level Lambda capable architecture. Farther beyond REAλICS aims to enrich user quality of experience, by hiding complexity layers and by being adaptive and flexible in allowing users to: define their input data streams (to be collected, streamed and queried), outline the features of the underlying resources and identify the connectors to be implemented depending on their computational needs. Live and past data streams are to be processed by hiding all the infrastructure's internal complexity and configuration from the user. Finally, REAλICS will advance scalability and elasticity restrictions specially in the infrastructural and software layers since the elasticity of the REAλICS architecture is foreseen from a service representation of the architecture.

The rest of the paper is structured as follows: Section II discusses related work in the proposed work mostly relevant areas, Section III has all the details for the proposed framework's principles and methodology, while its implementation and readiness level are presented in Section IV. Section V sums up the frameworks current statis and discusses its undergoing and future development.

II. STATE OF THE ART AND BEYOND

The real time feature of the REAλICS platform requires certain different optimization methodologies. Specifically, measures of latencies and delays become highly relevant. A general control mechanism to guaranty specified bounds on latencies is to: 1) observe the current application's workload and predict the trend of its change; and 2) based on the latter, to devise a strategy when and how to elastically scale resources (e.g., the number of virtual cluster nodes). There have been several approaches to optimize the usage of cloud resources according to the current workloads of the running applications, e.g., [6, 7, 8]. For example, reference [7] develops the controller for the applications run on the Amazon EC2 cluster; the controller enables high percentage quantile guarantees on the latencies (of type, e.g., 99% of all requests must be answered within 100ms).

The REAλICS platform will dynamically adapt the used resources to the running application, e.g., through elastic scaling of the size of the underlying clustered resources. For example, in the time intervals of high workloads of the application, the REAλICS platform will detect the need to add more cluster (i.e. worker) nodes; conversely, in the time intervals of low workloads, certain number of cluster nodes will be released. The elastic scaling will be performed such that the application performance, as measured, e.g., by the latency, is stable and is targeted to be independent of the current application workload.

There have been previous projects on the optimization of the PaaS cloud systems. At the BigFoot project, for example, the cross-layer system optimization of the PaaS cloud infrastructure is performed [10]. This approach, in contrast with the REAλICS, is dedicated to batch analysis and is not primarily concerned with the real time aspects of the platform. While certain aspects of the two frameworks are similar, like interactivity of the platform with the user, the major differences are that: 1) the REAλICS main focus is on the real time data analytics; and 2) the REAλICS major goal is to enable cloud federation. Moreover, multilevel software architectures are dvekl, Lambda architecture embodies a set of principles tailored for effective design of Big Data management systems. The greatest argument supporting its realization and use is the ability to run ad hoc queries involving computations against the whole range of stored data while data continues to expand from real time streams.

REAλICS beyond the State of the Art : REAλICS introduces novel design methodologies that have not been used before in the context of PaaS clouds and in overcoming problems raised on earlier work (such as in [7, 8, 9]). Specifically, REAλICS project advances the state of the art through the following:

1. *Distributed controller*. Existing work [7,8,9] utilizes centralized controllers to control the amount of application's resources used. We propose to use distributed controller, i.e., to introduce multiple controllers that work in parallel and reduce the delays of the control operation. We will optimally balance the tradeoff between the amount of resources allocated to the control part of the system (e.g., number of controllers) and the achieved performance of the control loop (e.g., achieved latency bound guarantees).

2. *Distributed bin packing*. An important aspect of the resource usage optimization is the problem of bin packing: how to store the data that corresponds to an application using the minimal amount of storage resources, e.g., the number of cluster nodes. The bin packing problem is NP hard problem but polynomial-time algorithms to approximately solve it are available [11]. Existing work, e.g., [11] proposes greedy algorithms for bin packing. While they are cost-effective, greedy algorithms may be highly sub-optimal in certain problems. We propose instead to use convex relaxation, primal-dual Lagrangian methods to solve the bin packing problem [12]. The advantages are guaranteed sub-optimality measures, and amenability for distributed implementation (see, e.g., [14]).

A key feature in REAλICS is the components optimization at the backend. While the data analytics and/or machine learning algorithms are mostly employed from the existing state of the art solutions (mostly relevant with streams mining and big data analytics), the optimization focus of the REAλICS project is on the elastic management of cloud computational and storage resources (at the top of IaaS layers such as OpenStack or OpenNebula) [25]. Specifically, our aim is to enable for the users' applications running on the REAλICS platform guaranteed Service Level Objectives (SLOs), with the minimized amount of cloud computational and storage resources used. Hence, the main objective of the REAλICS platform optimization is centered on its users and aims at optimizing their quality of experience. With respect to the SLOs, our major focus is on the real time aspect and the temporal SLOs, e.g., bounds on latencies, as this is highly relevant to satisfy our users' real time data mining needs.

Employing the above new methodologies can significantly improve the resource usage over the existing solutions, in the context of predictive analytics and mining solutions (ref [19-[24]). A major reason for the latter is that our methodology enables distributed solutions where existing work usually employ centralized solutions. The distributed solutions will allow to optimally strike the balance between the latency and the amount of used overall resources (including, e.g., the controller part), while this feature is not present in the existing work.

III. REAʌICS PRINCIPLES AND METHODOLOGY

New modeling and abstractions for data to schemas models, as well as for query and custom querying modeling, are required for supporting real time big data demands. In this context, the main addressed problems by the proposed framework refer to the need for the real-time execution of big data analytics. The principles governing the proposed design originate from the fact that data-centric distributed applications, are characterized by data intensive streams generation, requiring cloud-based solutions with emphasis on processing scalability and elasticity. The main flexibility of REAλICS originates from its capability to encapsulate different models and software tools which are re-usable for the various real time analytics tasks (such as data modeling, querying, streams monitoring, etc). Maximizing Quality of Experience (QoE) by adapting users requirements and allowing REAλICS backend to serve as platform for the user, is another main principle of REAλICS.

One novelty of REAλICS is that it sets the user at the core of the platform, aiming at QoE beyond what is offered by conventional platforms. Experimentation and definitions of requirements on both the infrastructural and the algorithmic levels are primarily user driven. The principles of the proposed framework are to enable users to select the tools for dynamically formulating a Lambda capable Big Data architecture. REAλICS framework design provisions a comprehensive set of software tools to tackle data intensive problems with real time demands. The proposed design addresses the non-expert users needs, since they can implement and further customize their preferences.

REAλICS proposes a fully elastic architectural stack to dynamically develop and execute use cases and experiments. In addition, REAλICS prioritizes the development of an open source uniquely tailored software toolbox, for easing use cases realization. Through the proposed design, the planned testing and implementation cycles are designed to optimize results. The planned toolbox software and algorithms for achieving better performance results, are taking into consideration both infrastructural (i.e. elasticity) and scalability demands.

REAλICS operates through interoperable Cloud APIs and open source infrastructural tools. At the core of a provided set of resources user owned elastic virtual cluster, based on the Lambda architectural prototype, are configured and delivered. These virtualized clusters are dynamically reconfigurable either upon a users' request (i.e. add/remove cluster nodes) or through heuristic approaches controlled via the REAλICS platform itself aiming at achieving best performance results with the minimal (at any given time) consumption of physical resource

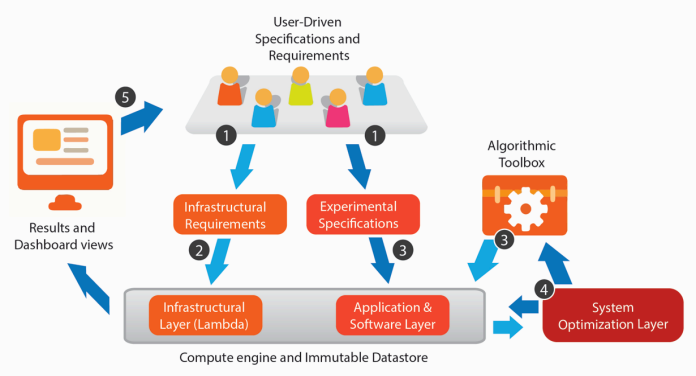

Fig. 1. REAλICS overview

The proposed methodology to support such a framework is outlined in Fig. 1, as a step-by-step systematic process. The overall REAλICS process utilizes a wizard based front-end interface (1st step), at which requirements for the infrastructural level and the specifications of the experiment/use case are set. Such specifications refer to: the data streams to collect, any additional historical datasets that may be related, data flow patterns to be followed, queries that should be addressed, tools to be implemented etc. At this *1st step*, the user is given the option to use and implement predefined algorithms provided via the open source and community driven toolbox. Then, the REAλICS service collects these requirements and as a *2nd step* it bootstraps a user owned Lambda capable infrastructure. Once the "pre-flight" testing of the underlying resources and services is completed as a *3rd step* the experimental specifications and the required algorithms from the toolbox are passed onto the compute and storage layers and hence the data collection and querying on the Lambda architectural level commence. Dynamic reconfiguration and adaptation takes place as a *4th step*. During this cycle the infrastructural layers scale up or down and adapt to an equilibrium based on user restrictions, quotas and computing demands, while the software is re-designed and integrated through development cycles to reach optimal levels of maturity and performance. As a *5th step* the results are delivered to the user who can judge the initial

use case setup to decide whether reconfiguring and reinitiating of a given experiment is required. Moreover, data collected in previous iterations of the experiment may be kept for future use. Therefore, as an experiment grows and evolves it follows a perpetual cycle enabling the data collected for the experiment to be constantly accumulated.

In summary, REAλICS proposes a flexible and extensible software architecture which can capture users preference at the front-end and adapt the appropriate distributed technologies and processes at the back-end. Such a model enables real time analytics in large-scale data driven cloud-based systems.

IV. IMPLEMENTATION AND READINESS LEVEL

REAλICS perpetual cycle strategy enables the realization of a given experiment on a software stack (Fig. 2). The software stack covers all of the processes from the user-driven initiated requirements to the systems and services. REAλICS Web front-end user interface, handles user interaction and a back-end system includes subsystems and services to interact with the distributed persistent storage and the virtualized YARN/Hadoop stacks. To proceed with an adaptive and flexible architecture REAλICS empowers a tailored API, which addresses the front-end requirements, and appropriately manages the stack, the network and the persistent storage.

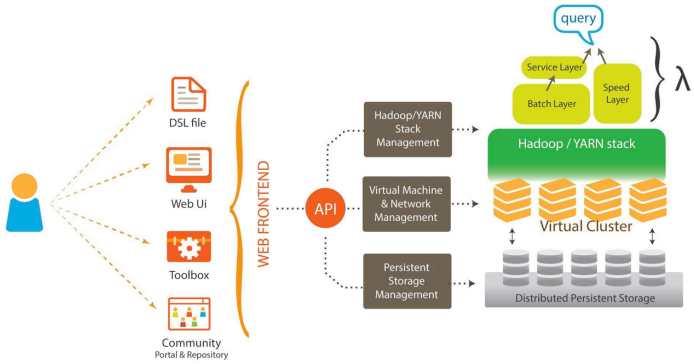

Fig. 2. REAλICS software stack overview

REAλICS iterative and adaptive procedure enables both: *monitoring* of an experiment's cycle in the course of its execution (i.e. data to be collected, together with the corresponding queries, which may vary and evolve as more insight is delivered via the REAλICS platform) and the *offering* of an option to withhold data, from the previous versions of the experiment, as the latter one evolves (thus not losing historical data which have already been collected in previous versions of the experiment).

A. Components and Functionality

To support adaptiveness efficiently, REAλICS builds upon and utilizes a Domain Specific Language (DSL), which describes an evolving experiment conclusively (the YAML markup language specification is being used syntactically). DSL files are forwarded from the Frontend UI to the custom service programmatic interface for the realization/configuration of the resources to take place.

To support the functionality of cloud federation and real time data analytics, a user-driven cloud management platform at the IaaS layer is proposed. Since such cloud management platforms are evolving, REAλICS adapts its capabilities to enable outsourcing of computing power to external cloud infrastructures. Monitoring of the IaaS layer can be extended to account for the real time data analysis requirements. Next, the main components and their functionalities are summarized.

Front-end: REAλICS proposes a rich-feature Web User (wizard enabling) Interface (UI) via which the user is able to set project requirements and access the computed results, the toolbox contents and the repository. More specifically, user options are given to: provide an initial set of requirements via the UI, define the data streams to collect, specify any additional related historical datasets, data flow patterns to be followed, as well as queries which should be executed on the collected data sets. The outcome of this step-by-step options declaration is stored at a DSL file, which the user may further archive, edit or share with other users of the platform. Additionally, REAλICS user can dynamically update the whole work and data flow via the Web UI wizard enabled procedure. As mentioned above, the carried out experiment undergoes of an iterative and adaptive procedure, i.e. in the course of it the data streams, along with the corresponding queries may be updated. An extra feature enabled through REAλICS is that the user is given the option to keep the data from the previous versions of the experiment (thus not losing already collected historical data).

Software Toolbox: REAλICS provides a software toolbox that includes predefined, easily configurable and adaptable algorithms and data views, which the user may implement (per use case). The toolbox is extensible and can include flow algorithms for tackling common data intensive issues. Alongside this algorithmic toolbox, predefined views of the results are available upon request (again for easing and improving users QoE). REAλICS toolbox heavily builds upon Apache Spark and Apache Flink with further development and implementations for state of the art machine learning and data mining algorithms, which can be executed in parallel on computer clusters. Among the specific categories of algorithms we primarily target collaborative filtering, classification, clustering, dimensional reduction, topic modeling and other algorithms, while new algorithms can also be inspired and implemented in the expansion of REAλICS. The fact that such platforms are executed upon frameworks that compose basic elements of the REAλICS Lambda driven architecture (Hadoop, Spark, Flink) allows the exploitation of their capabilities from the REAλICS framework for the construction of its open source toolkit of algorithms. Implemented algorithms will be offered from the REAλICS toolkit such that users can choose among them for their specific data analytics requirements.

B. Lambda architecture Implementation

REAλICS enables big data analytics projects to be implemented by using virtual clusters which implement a Lambda architecture approach. As discussed in Section II, the

most significant benefit of the proposed Lambda architecture is that the user query is executed over the whole range of data which are accumulated up to the point in time the query was submitted, thus both (past and live) data streams are taken into account.

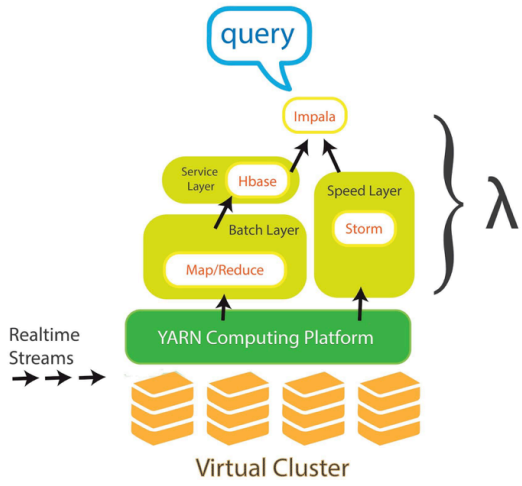

Fig. 3. REAλICS Lambda based configuration. Tools from the Cloudera Hadoop ecosystem are placed in the corresponding layers of Lambda. The tools outlined in this figure are only indicative

λ at REAλICS is proposed to exploit its three layer design for user defined queries addressed at the unified Lambda service layer, the data accumulated on the storage (batch) layer and the data streams processed dynamically on the speed layer.

C. Testing and Implementation

REAλICS perpetual methodology cycle enables several use cases implementation due to its multi-phase design (Fig. 4.):

Phase 1: Use case definition and set-up of the initial set of requirements. These requirements translate into requirements for the : (1) infrastructure (i.e. expected size of Lambda infrastructure), (2) software tools and Big Data ecosystem components and (3) data sources and streams, data flows, data partitioning etc.

Phase 2: The user (i.e. use case owner) is supported by the Web frontend wizard to feed these requirements into the REAλICS platform. A DSL file is properly formatted and delivered to the use case owner

Phase 3: The initial set of requirements (in the form of a DSL file) are fed into REAλICS back-end components and Tasks queue handles bootstrapping a virtualized Lambda infrastructure in accordance to the requirements. Unit tests (prebuilt and/or built dynamically based on the requirements) are executed to ensure all components are functional.

Phase 4: Data collection from defined sources and streams commences. Data flow patterns are also applied. Data are stored immutably on the persistent storage layer while computations based on the user defined queries are utilized in predefined or ad-hoc intervals driven by the user requirements.

Phase 5: The experiment's execution is examined and assessed. The use case owner updates requirements. The use case owner has the option to either drop the data already collected in Phase 4 or maintain them to be used in the next version (iteration) of the experiment). At this point the use case owner thus returns to Phase 2 to update requirements (either via the wizard steps or by updating the DSL file) and continues within this perpetual use case cycle.

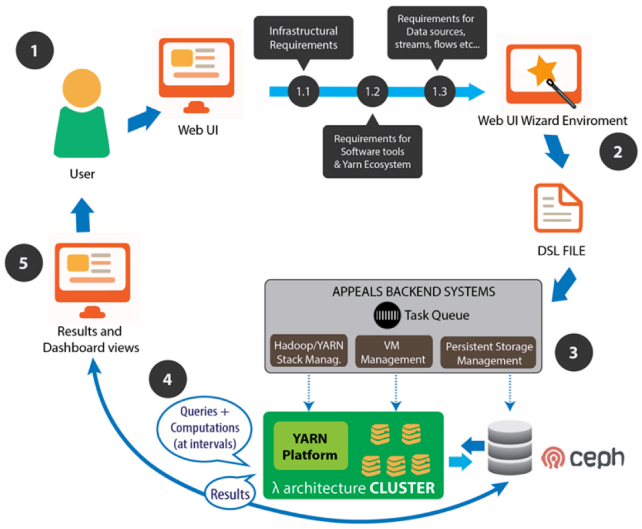

Fig. 4. REAλICS Use case experimentation perpetual lifecycle

Intrusion detection for distributed resources: The Aristotle University of Thessaloniki hosts four (4) Data centers distributed across the University Campus. The infrastructure overall is comprised of approximately more than 1000 physical and virtual servers, more than 200 physical networking devices and more than 500 virtual subnets, most of which are monitored at the systems level via appropriate monitoring appliances. Currently, the systems are monitored for intrusions on the physical level; However an overall view of the campus resources is not achievable with standalone monitoring appliances due to the volume and variety of the data needed for such a purpose estimated to be in the order of tens of TBs. Moreover, as industries and academia turn towards "Bring Your Own Device" (BYOD) and IoT strategies bringing forth the need for a really "distributed resources" targeted IDS monitoring is imperative. The use case, currently being developed shall be used during the validation phase of the REAλICS, which we envision will be used to build upon it a real-time IDS monitoring service specifically targeted on detecting intrusions and break-in attempts not only focusing on data derived from the systems logging mechanisms (as is most commonly done on several IDS implementations) but also on data aggregated from different sources (environmental monitoring sensors, door traps, network taps IoT settings etc).

V. CONCLUSIONS AND FUTURE WORK

REAλICS proposed framework addresses the highly distributed systems inherent demand to gain knowledge from multiple data sources and streams, such as live data as well as historical data sources. Challenges in this problem are due to the fact that appropriate insight on data's summarization should be delivered to engineers, decision makers and interested stakeholders in real time demanding environments where even a fraction of a second counts. REAλICS embeds innovation potential at various levels of its approach, in the context of offering an innovation-friendly industrial and competitive software stack. REAλICS is innovating since its user-centered framework entities, prioritize users QoE and promote community driven open source software distributions..

Adaptiveness of REAλICS is heavily based on the user defined custom DSL implementation (currently in prototype status). Several contributions that are already available in the PaaS landscape will be taken into account in the final DSL specification. Moreover, elasticity at the moment can be handled on the batch and service layers but to truly address elasticity on the stream layer the latest available advances from Apache Flink and Apache Spark will need to be endorsed and adopted within REAλICS.

REAλICS seeks to advance the state of the art in IaaS management by extending functionality and feature-rich solutions for the comprehensive management of virtualized data centers. Such an approach will be validated and tested under the ENCASE project use cases which stress test data integration demands under security and privacy preserving mechanisms. The future goal is to validate the proposed framework a) as an enterprise-ready product and not a group of components that need integration to be built upon the cloud, b) to give support for services provided by the people who developed and manage the software and c) to provide easiness to fit into any existing data platform. An example that would fit the design goals of REAλICS is the platform, currently being developed and used, in the context of the ENCASE project, where fast adaptiveness and integration of new filtering tools is needed in real time.

ACKNOWLEDGMENTS

The authors thank Efrosini Kaltimiridou and Konstantinos Kaggelidis, staff of the IT Center of Aristotle University, for their contribution in the framework's design and presentation. Part of this work is funded from the European Union's Horizon 2020 research and innovation programme "ENCASE" under the Marie Skłodowska-Curie grant agreement No 691025.

REFERENCES

[1] Brosig, Fabian, et al. "Quantitative evaluation of model-driven performance analysis and simulation of component-based architectures." *IEEE Transactions on Software Engineering* 41.2 (2015): 157-175.

[2] Bahrami, Mehdi, and Mukesh Singhal. "The role of cloud computing architecture in big data." *Information granularity, big data, and computational intelligence*. Springer International Publishing, 2015. 275-295.

[3] Schutt, Kyle, and Osman Balci. "Cloud software development platforms: A comparative overview." *Software Engineering Research, Management and Applications (SERA), 2016 IEEE 14th International Conference on*. IEEE, 2016.

[4] D. McCreary and A. Kelly, Making Sense of NoSQL: A Guide for Managers and the Rest of Us. Manning Publications Company, 2013.

[5] Buyya, Rajkumar, et al. "Big Data Analytics-Enhanced Cloud Computing: Challenges, Architectural Elements, and Future Directions." *Parallel and Distributed Systems (ICPADS), 2015 IEEE 21st International Conference on*. IEEE, 2015.

[6] Kiran, Mariam, et al. "Lambda architecture for cost-effective batch and speed big data processing." *Big Data (Big Data), 2015 IEEE International Conference on*. IEEE, 2015.

[7] B. Jennings, R. Stadler, "Resource management in clouds: Survey and research challenges," J. Netw. Syst. Manage., Springer, New York, 2014

[8] B. Trushkowsky, P. Bodik, A. Fox, M. J. Franklin, M. I. Jordan, D. A. Patterson, "The SCADS director: Scaling a distributed storage system under stringent performance requirements,"in proc. 9th USENIX Conference on Filing Storage Technologies (FAST), 2011

[9] G. Toffetti, A. Gambi, M. Pezze, C. Pautasso, "Engineering autonomic controllers for virtualized WEB applications," in: Benetallah, B. Casati F., Kappael, G., Rossi, G., (eds.), Web Engineering no. 6189, in LNCS pp-66-80, Springer, Berlin, 2010

[10] Shen, Yun; Thonnard, Olivier; Vervier, Pierre-Antoine; Dacier, Marc, *Scalable Multi-Criteria Data Clustering for Big Data Security Intelligence Analytics,* submitted to IEEE Transactions on Big Data, 2015.

[11] V. Gupta, A.Radovanovic, "Online Stochastic Bin Packing," arxiv preprint, available at: http://arxiv.org/pdf/1211.2687.pdf, 2012

[12] M. Chiang, S. H. Low, A. R. Calderbank, J. C. Doyle, "Layering as Optimization Decomposition: A Mathematical Theory of Network Architectures,"Proceedings of the IEEE, vol. 95, no.1, Jan. 2007

[13] Bohlouli, M., Schulz, F., Angelis, L., Pahor, D., Brandic, I., Atlan, D., & Tate, R. (2013). Towards an integrated platform for big data analysis. In Integration of Practice-Oriented Knowledge Technology: Trends and Prospectives (pp. 47-56). Springer Berlin Heidelberg.

[14] X. Amatriain. Mining large streams of user data for personalized recommendations. SIGKDD Explorations, 14(2), 2012.

[15] U. Kang and C. Faloutsos. Big graph mining: Algorithms and discoveries. SIGKDD Explorations, 14(2), 2012.

[16] J. Lin and D. Ryaboy. Scaling big data mining infrastructure: The twitter experience. SIGKDD Explorations, 14(2), 2012.

[17] Y. Sun and J. Han. Mining heterogeneous information networks: A structural analysis approach. SIGKDD Explorations, 14(2), 2012.

[18] [12] Dhar, V. (2013)."Data science and prediction". *Communications of the ACM* 56 (12): 64.

[19] Predictive Model Markup Language (PMML) http://www.dmg.org/

[20] The shiny R tool, http://www.rstudio.com/shiny/

[21] Revolution analytics, http://www.revolution-computing.com

[22] Eirini Giannakidou, Athena Vakali, Nikolaos Mavridis: Towards a Framework for Social Semiotic Mining. WIMS 2014: 21:1-21:6

[23] González Alonso, I., Rodríguez Fernández, M., Jacobo Peralta, J., & Cortés García, A. (2013). A Holistic Approach to Energy Efficiency Systems through Consumption Management and Big Data Analytics. International Journal On Advances in Software, 6(3 and 4), 261-271.

[24] Domingos, Pedro. "A few useful things to know about machine learning." *Communications of the ACM* 55.10 (2012): 78-87.

[25] Ignacio M. Llorente OpenNebula vs. OpenStack: User Needs vs. Vendor Driven, Technical Report http://opennebula.org/opennebula-vs-openstack-user-needs-vs-vendor-driven/]