BigDataStack: A holistic data-driven stack for big data applications and operations

Dimosthenis Kyriazis[a], Christos Doulkeridis[a], Panagiotis Gouvas[b], Ricardo Jimenez-Peris[c], Ana Juan Ferrer[d], Leonidas Kallipolitis[e], Pavlos Kranas[c], George Kousiouris[a], Craig Macdonald[f], Richard McCreadie[f], Yosef Moatti[g], Apostolos Papageorgiou[h], Marta Patino-Martinez[i], Stathis Plitsos[j], Dimitris Poulopoulos[a], Antonio Paradell[d], Amaryllis Raouzaiou[e], Paula Ta-Shma[g], Valerio Vianello[i]

[a] University of Piraeus, Piraeus, Greece
[b] Ubitech, Athens, Greece
[c] LeanXcale, Madrid, Spain
[d] Atos Research and Innovation, Madrid, Spain
[e] Athens Technology Center, Athens, Greece
[f] University of Glasgow, Scotland, United Kingdom
[g] IBM Research, Haifa, Israel
[h] NEC Laboratories Europe, Heidelberg, Germany
[i] Universidad Politecnica de Madrid, Madrid, Spain
[j] Danaos Shipping, Athens, Greece

dimos@unipi.gr, cdoulk@unipi.gr, pgouvas@ubitech.eu, rjimenez@leanxcale.com, ana.juanf@atos.net, l.kallipolitis@atc.gr, pavlos@leanxcale.com, gkousiou@gmail.com, craig.macdonald@glasgow.ac.uk, richard.mccreadie@glasgow.ac.uk, moatti@il.ibm.com, apostolos.papageorgiou@neclab.eu, mpatino@fi.upm.es, splitsos@danaos.gr, james@unipi.gr, antonio.paradell@worldline.com, a.raouzaiou@atc.gr, paula@il.ibm.com, vvianello@fi.upm.es

***Abstract*—The new data-driven industrial revolution highlights the need for big data technologies to unlock the potential in various application domains. In this context, emerging innovative solutions exploit several underlying infrastructure and cluster management systems. However, these systems have not been designed and implemented in a "big data context", and they rather emphasize and address the computational needs and aspects of applications and services to be deployed. In this paper we present the architecture of a complete stack (namely BigDataStack), based on a frontrunner infrastructure management system that drives decisions according to data aspects, thus being fully scalable, runtime adaptable and high-performant to address the needs of big data operations and data-intensive applications. Furthermore, the stack goes beyond purely infrastructure elements by introducing techniques for dimensioning big data applications, modelling and analyzing of processes as well as provisioning data-as-a-service by exploiting a seamless analytics framework.**

Big data architectures; infrastructure management; big data as a service;

I. Introduction

It is a fact that we experience the data-driven society. By 2020, the world will generate 50 times today's information, creating the "Digital Universe" [1] that grows exponentially since the data creation growth rate is between 40% and 60% according to OECD [2]. The result of this data explosion is apparent in all domains of everyday life, ranging from user-generated content of around 2.5 quintillion bytes every day [3] to applications in healthcare, transportation, logistics and retail. In all domains, data is the key, not only by adding value and increasing the efficiency of existing solutions but also by opening new opportunities and facilitating new functionalities in these domains. What is more, the value of data goes beyond their utilization in data-driven applications since data per se have value. In this data-driven world, infrastructures play a critical role. It is expected that more data will be uploaded and downloaded from various infrastructure in the next years [4]. In this context, "enhanced infrastructure capabilities are a must in data centres" for most customers going forward in 2017 and beyond [5]. The goal for infrastructures is clear: go beyond storing, processing and offering data to enabling optimum data service provisioning by turning the underlying infrastructures to enhanced data-driven and data-oriented environments. These environments should also include mechanisms for runtime adaptations across the complete data path and service lifecycle, since as the data evolve (including sources, formats, rates, etc), the needs will also change (likely increase), triggering infrastructure adjustments [6]. These items have also been characterised as of key priorities in the BDVA Strategic Research and Innovation Agenda [7].

Additionally, given the data value for applications in different domains, the need is for business and process analytics frameworks. Such frameworks will exploit analytics and processing frameworks for predictive and perspective analytics facilitating event and pattern discovery as well as deep learning for business intelligence.

Based on the above, we propose a data-driven architecture, which aims at ensuring that infrastructure management is fully efficient and optimized for data operations and data-intensive applications. A representative data-driven example would be that a service-defined decision (current approach) may result to VMs deployment in the same physical host for time-critical operations (e.g. real-time stream processing). The BigDataStack approach instead will base the decision according to data aspects (e.g. generation rates, transfer bottlenecks, etc), which in this example could be that the data to be aggregated and processed emerge from a big number of distributed sources. Thus, the bottleneck is on retrieving the data from the sources. To this end, BigDataStack infrastructure management would propose a data-driven decision resulting to containers/VMs deployed in geographically distributed physical hosts.

The remainder of the paper is structured as follows: Section II reviews related work and highlights the advancements of our approach. Section III describes the phases of the overall architecture, which is presented thereafter in Section IV. The paper concludes with a summary of the presented architecture and a discussion on future work and potentials.

II. Related Work

A. *Application performance analysis and dimensioning*

Application analysis is a key step for enabling any type of optimization on the infrastructure level. In order to meet scaling requirements, an analysis model is presented [8], which considers an application as a set of parallel or pipelined tasks that constructs a set of jobs, while in [9], a probability-theory model is based mostly on the application deployment architecture working together with a queueing model for handling the incoming requests. Probabilistic models are helpful in cases of predicting behaviours, requests or performance-related metrics [10]. In [11] different execution logs and footprints from the developers are analyzed to get an application description. [12] is solving the problem of allocating scientific data-intensive workloads with respect to data transfer and execution times, by considering a workload to be a set of tasks that construct a directed acyclic graph. Authors in [13] present an optimization approach on the basis of runtime adaptive MapReduce, using feedback and stochastic learning controls for parameterizing mappers and reducers. Automatically parameterizing MapReduce parameters is also followed in [14], where regression models are used for forecasting performance under different Hadoop configurations.

In this paper, we propose an application dimensioning workbench to identify dependencies between application components, between such components and data services, as well as between data services. The workbench includes a load generator / injector to facilitate benchmarking with different set of input parameters and according to these identify and capture the dependencies and their impact. The workbench also incorporates performance prediction techniques for the required infrastructure resources related to the application and the data services (e.g. cleaning). For the dimensioning of the latter, analysers of data generation patterns and query executions are also part of the workbench.

B. *Infrastructure and cluster management*

Containers as a Service [15] is becoming the new Platform as a Service. Containers are a more lightweight virtualization concept [16], which can be seen as more flexible tools for packaging, delivering and orchestrating both software services and applications. However, controlling a vast deployment of containers presents some complications, such as the need to match the resources required by containers [17]. Several solutions emerge to manage containers including Docker Swarm [18], CoreOS [19] and Kubernetes [20] among others. Apache Mesos [21] focuses on the effective isolation of resources and sharing of applications across distributed networks or frameworks. Google's Borg [22] combines admission control, efficient task-packing, over-commitment, and machine sharing with process-level performance isolation. Quasar [23] is able to increase resource utilization while providing consistently high application performance. In [24], the authors introduce an architecture that increases persistence and reliability of infrastructure management building upon Chef configuration management system and IaaS resources from Amazon. A container-based cluster management platform, where the virtualization technology is assimilated with resource and job management is discussed in [25].

In the proposed infrastructure management system all decisions including resource allocation, admission control, scaling, orchestration, runtime migration and adaptations, will be data-driven. This data-driven approach will incorporate the interdependencies between compute, storage and networking resources to drive resource management decisions. For different target resource characteristics, a flexible infrastructure deployment approach for containers, small-footprint or full-fledged VMs will be triggered.

C. *Dynamic orchestration of resources & data operations*

Orchestration can be linked to optimizations regarding placement of tasks to resources, decisions regarding parallelization degrees of parallelizable tasks / services, load balancing, and more. These are handled in a “service-driven” way, meaning that the optimization functions, the criteria, and the system setup are built around basic features such as CPU, bandwidth, and task / service requirements in order to optimize certain metrics (e.g. latency). Transform latency- and load-related metrics into distances in a “cost space” and then apply a placement algorithm which minimizes the cost in this cost space is proposed in [26], while [27] optimizes placement in stream processing based on topology traffic and node capabilities. Various other works for such optimizations can be found in the survey of [28], which focuses on stream processing. A very interesting approach is also presented in [29] with an emphasis on elasticity in the case of stream processing. Such concrete optimizations have rarely been handled homogeneously or investigated in a common context, there are also gaps towards making them data-driven rather than service-driven.

To support data-driven overall orchestration and data-driven solutions of specific optimization problems (e.g. placement), we propose techniques to identify the synergies between characteristics of data analytics and system KPIs, e.g. functions that represent how data I/O volumes affect the CPU-intensity of certain tasks. BigDataStack provides the basis for using such data-related synergies for all system orchestration aspects by defining specifications for profiling homogenously all involved entities (e.g. nodes, network links, application tasks) in data-driven orchestration. A rule-based approach for triggering runtime optimizations based on the monitored data will also be applied.

III. Main phases of the Stack

The concept of BigDataStack is reflected in three main phases: Dimensioning, Deployment and Operation. The goal of dimensioning is to estimate the required infrastructure resources for the data services and application components. Deployment phase delivers optimum deployment for these data and application services, by considering the resources and the interdependencies between application components and data services (based on the dimensioning phase outcomes). The operation phase enables the delivery of services including technologies for resource management, monitoring and evaluation towards runtime adaptations.

A. *Dimensioning phase*

The dimensioning approach facilitates the provisioning of data services and applications by understanding their data-related requirements (e.g. data sources, storage needs, etc) and their needs across the data path (e.g. required services for data representation, aggregation, etc). Thus, the approach includes a 2-step phase. In the first step the composite application is analysed to identify the required data services. The example illustrated in Fig. 1a shows that 3 out of 5 application components require specific data services for aggregation and analytics. The second step dimensions these identified data services and the application components, in terms of their infrastructure resource needs exploiting a load injector generating different loads to benchmark the services and analyze their resources and data requirements (e.g. volume, generation rate, legal constraints, etc).

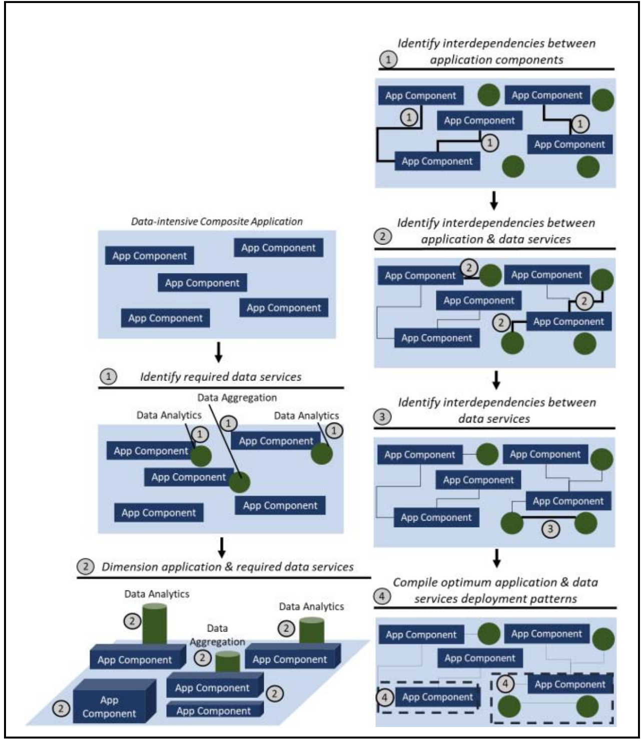
Data-intensive Composite Application
App Component
1 Identify required data services
Data Aggregation
Data Analytics
2 Dimension application & required data services
Identify interdependencies between application components
Identify interdependencies between application & data services
Identify interdependencies between data services
Compile optimum application & data services deployment patterns

Figure 1. (a) Dimensioning of application and data services. (b) Deployment patterns compilation

B. *Deployment phase*

The deployment approach aims at identifying the optimum deployment patterns and configurations for the data services and applications. The optimization need emerges from the fact that the services have interdependencies that need to be considered to increase the performance of all data-related operations. To this end, BigDataStack deployment approach includes a four-step phase. In the first three steps different interdependencies are identified between: application components, application components and data services, and between data services (e.g. between storage and aggregation). Following the identification and analysis of these interrelations and their impact in terms of computation, storage and networking resources, optimum deployment patterns (considering data characteristics such as volumes, application components and data services I/O rates) will be compiled as shown in the previous figure (Fig. 1b).

C. *Operations phase*

The operations phase aims at the optimized data-driven management of the infrastructure resources. The approach includes a 7-step process as depicted in Fig. 2: (i) compute and storage resources are identified based on the outcomes of the dimensioning phase, (ii) distributed storage resources are allocated considering the computing resources allocated in the previous step, (iii) data-driven networking functions are compiled and deployed to facilitate networking between different computing and storage resources, (iv) application and data services are deployed and orchestrated based on "combined" data- and application- aware orchestration templates, (v) data analytics tasks are distributed across different data stores, (vi) monitoring data is collected and evaluated for the resources (computing, storage, network), the data services (e.g. query execution status) and the application components, and (vii) runtime adaptations take place for all elements of the environment including resource re-allocation, storage and analytics re-distribution, and re-compilation of network functions and deployment patterns.

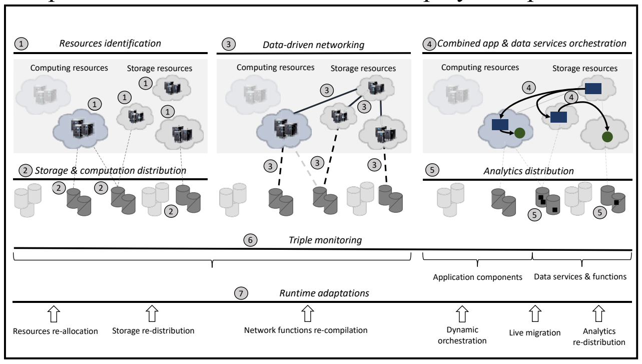

Figure 2. Operation and runtime adaptation of the complete environment.

IV. Proposed architecture

The phases described above and realized through a set of mechanisms, which are presented in Fig. 3. The raw data are ingested through the *Gateway & Unified API* component to the *Storage* engine of BigDataStack, which enables storage and data migration across different resources. The engine includes both stores for relational and non-relational data, as well as an object store and a CEP engine for streaming data processing. The raw data are obtained by the *Data Cleaning* component to enhance their quality in terms of completeness, accuracy and volatility, while thereafter as cleaned data are obtained by the *Data Layout Optimization* framework in order to be modelled (using flexible schemas to describe streaming and stored data) and annotated with metadata.

Given the stored data, decision makers are able to model their business workflows through the *Process modelling framework* that incorporates two main components. The first of them is the *Process modelling*, which provides an interface for business process modelling and specification of optimization goals for the overall process. The outcome of the component is a model in a structural representation (e.g. YAML or JSON). *Process mapping* identifies the specific mining and analytics tasks to realize the corresponding business processes (obtaining candidates from a *Catalogue* of analytics). The outcome of this component is a specific list of analytics tasks (in the form of a graph) that is passed to the *Application dimensioning workbench* to estimate the resource requirements by also considering interdependencies between services. Moreover, this graph is obtained by the *Data toolkit,* which includes a component that exploits the preferences and requirements of the users and along with the mapped list / graph of processes creates a "playbook" as a deployable concrete graph of services. The playbook also includes pre-deployment configuration information for the services (e.g. the "k" for k-means clustering).

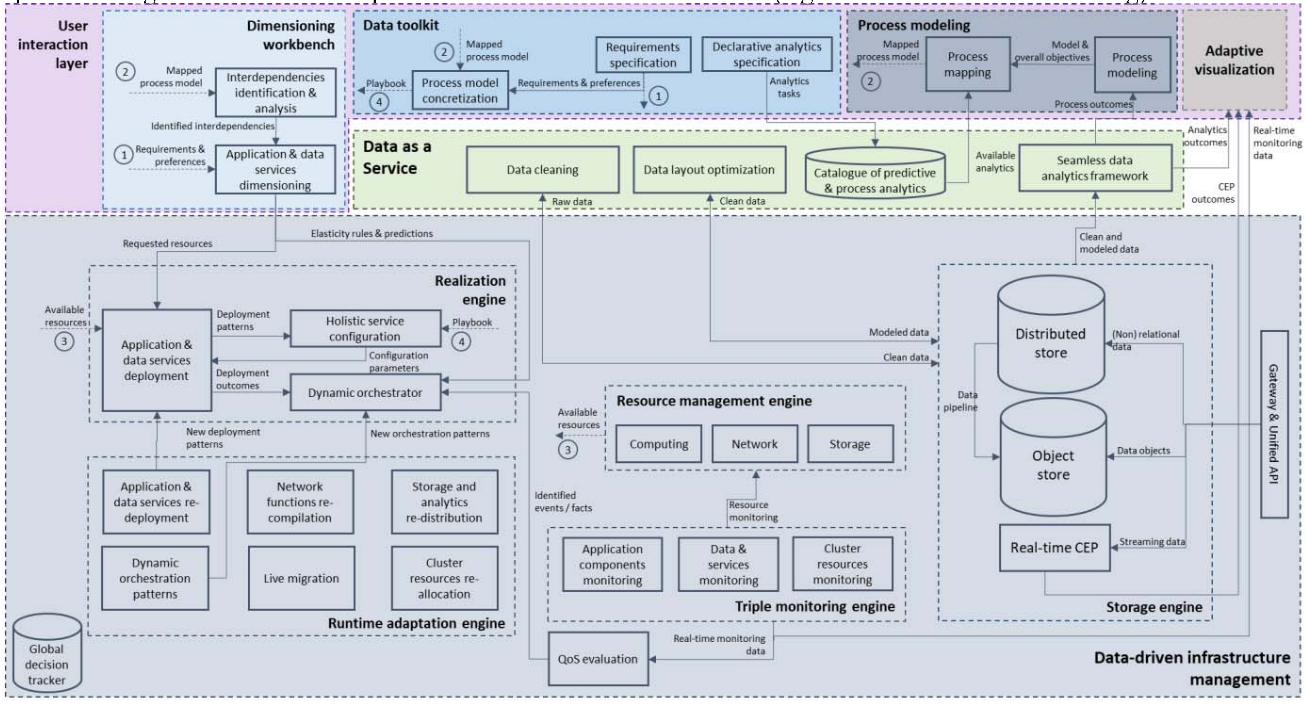
User interaction layer
Dimensioning workbench
Mapped process model
Interdependencies identification & analysis
Identified interdependencies
Requirements & preferences
Application & data services dimensioning
Data toolkit
Mapped process model
Requirements specification
Declarative analytics specification
Playbook
Process model concretization
Requirements & preferences
Analytics tasks
Process modeling
Mapped process model
Process mapping
Model & overall objectives
Process modeling
Process outcomes
Adaptive visualization
Data as a Service
Data cleaning
Data layout optimization
Catalogue of predictive & process analytics
Available analytics
Seamless data analytics framework
Analytics outcomes
Real-time monitoring data
CEP outcomes
Raw data
Clean data
Requested resources
Elasticity rules & predictions
Clean and modeled data
Realization engine
Available resources
Deployment patterns
Holistic service configuration
Playbook
Application & data services deployment
Configuration parameters
Deployment outcomes
Dynamic orchestrator
Modeled data
Clean data
Distributed store
(Non) relational data
Data pipeline
Object store
Data objects
Real-time CEP
Streaming data
Storage engine
Gateway & Unified API
New deployment patterns
New orchestration patterns
Available resources
Resource management engine
Computing
Network
Storage
Application & data services re-deployment
Network functions re-compilation
Storage and analytics re-distribution
Dynamic orchestration patterns
Live migration
Cluster resources re-allocation
Runtime adaptation engine
Identified events / facts
Resource monitoring
Application components monitoring
Data & services monitoring
Cluster resources monitoring
Triple monitoring engine
Global decision tracker
QoS evaluation
Real-time monitoring data
Data-driven infrastructure management

Figure 3. BigDataStack overall concpetual architecture.

The outcome of the application dimensioning workbench is relayed to the *Realization engine*, which includes a set of sub-components for deployment and orchestration. Based on the obtained dimensioning outcomes (i.e. resource needs and dependencies) and the availability of resources (information received from the *Resource management engine*), deployment patterns are compiled and used for deployment on the selected resources (through the *Application & data services deployment* mechanism), along with potential configuration parameters obtained by the *Holistic services configuration* component. The *Dynamic* o*rchestrator* performs orchestrations of the application and data services on the allocated resources. The execution of the data analytics tasks (triggered by the *Dynamic orchestrator*), is performed on the *Seamless data analytics framework* that facilitates analytics across multiple resources and locations for both data in flight and at rest.

During runtime, the *Triple monitoring engine* collects data regarding: resources, application components (e.g. application metrics, data flows across application components, etc) and data operations (e.g. analytics / queries progress, storage distribution, etc.). The collected data are evaluated through a *QoS Evaluation* component to identify events / facts that affect the overall quality of service. These are utilized by the *Runtime adaptation engine*, that includes a set of components to trigger the corresponding runtime adaptations for all infrastructure elements.

Finally, the architecture includes the so called *Global decision tracker*, which aims at storing the information from all components regarding the decisions taken for future optimizations. The key rationale for the introduction of this component is the fact that decisions have a cascading effect in the proposed architecture. For example a dimensioning decision affects the deployment patterns compilation, the distribution of storage and analytics, etc. The information whether these decisions are altered during runtime will be exploited for optimized future decisions across all components through the decision tracker. Thus, the tracker provides the ground for cross-component optimization

V. Conclusions

The emerging requirements and wide exploitation of data operations and data-intensive applications highlight the need for innovative offerings across the complete data lifecycle. In this paper we have introduced BigDataStack as a complete stack based on an infrastructure management system that drives decisions according to data aspects thus being fully scalable, runtime adaptable and performant for big data operations and data-intensive applications. BigDataStack promotes automation and quality and ensures that the provided data are meaningful, and fit-for-purpose through its Data as a Service offering that addresses the complete data path with approaches for data cleaning, modelling, data layout optimization, and distributed storage. The architecture also incorporates approaches for data-focused application analysis and dimensioning, and process modelling towards increased performance, agility and efficiency, while a toolkit allows the specification of analytics tasks and their integration in the data path. It is within our next steps to provide an implementation of the proposed architecture based on Kubernets and validate it through three different scenarios from maritime, retail and banking.

Acknowledgment

The research leading to the results presented in this paper has received funding from the European Union's funded Project BigDataStack under grant agreement no 779747.

References

[1] IDC, "The Digital Universe of Opportunities: Rich Data and the Increasing Value of the Internet of Things", https://www.emc.com/collateral/analyst-reports/idc-digital-universe-2014.pdf

[2] OECD, "New sources of growth- knowledge based capital", http://www.oecd.org/sti/inno/newsourcesofgrowthknowledge-basedcapital.htm

[3] IBM, "Bringing big data to the enterprise", http://www-01.ibm.com/software/data/bigdata/what-is-big-data.html

[4] Where Did The 'Data Explosion' Come From?, http://blog.bimeanalytics.com/english/where-did-the-data-explosion-come-from

[5] Data Center Knowledge, "2016: The Year of The Data Center", http://www.datacenterknowledge.com/archives/2015/11/23/2016-year-data-center/

[6] J. Baer, "The 4 keys to running a data-driven business", http://venturebeat.com/2013/12/04/the-4-keys-to-running-a-data-driven-business/

[7] Big Data Value Association, "Strategic Research and Innovation Agenda", http://www.bdva.eu/node/123

[8] M. Mao, and M. Humphrey, "Auto-scaling to minimize cost and meet application deadlines in cloud workflows," Proc. International Conference for High Performance Computing, Networking, Storage and Analysis, Seattle, Washington, pp. 12-18, 2011.

[9] J. Jiang, J. Lu, G. Zhang, and G. Long, "Optimal Cloud Resource Auto-Scaling for Web Applications," 13th IEEE/ACM International Symp. Cluster, Cloud and Grid Computing (CCGrid), May 2013.

[10] Y. Xie, and S-Z. Yu, "A large-scale hidden semi-Markov model for anomaly detection on user browsing behaviors", IEEE/ACM Transaction on Networks, vol. 17, Feb. 2009, pp. 54-65.

[11] C. Szabo, Q.Z. Sheng, T. Kroeger, Y. Zhang, J. Yu, "Science in the cloud: Allocation and execution of data-intensive scientific workflows", Journal of Grid Computing, vol. 12, 2014, pp. 245–264.

[12] M. Koehler and S. Benkner, "Design of an Adaptive Framework for Utility-Based Optimization of Scientific Applications in the Cloud", Proc. IEEE/ACM Fifth International Conference on Utility and Cloud Computing (UCC '12), IEEE Computer Society, Washington, DC, USA, 303-308, 2012.

[13] F. Zhang, J. Cao, X. Song, H. Cai, and C. Wu, "AMREF: An Adaptive MapReduce Framework for Real Time Applications," Proc. International Conference on Grid and Cloud Computing (GCC '10), IEEE Computer Society, Washington, DC, USA, pp. 157-162, 2010.

[14] H. Yang, Z. Luan, W. Li, D. Qian, G. Guan, "Statistics-based Workload Modelling for MapReduce," Proc. IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW '12), IEEE Computer Society, Washington, DC, USA, pp. 2043-2051, 2012.

[15] Computing, "The rise of containers-as-a-service platform", https://www.computing.co.uk/ctg/opinion/3016229/the-rise-of-the-containers-as-a-service-platform

[16] P. Claus, and B. Lee, "Containers and clusters for edge cloud architectures--a technology review," 3rd International Conference on Future Internet of Things and Cloud (FiCloud), IEEE, 2015.

[17] P. René, F. Holzschuher, and F. Pfitzer, "Docker cluster management for the cloud-survey results and own solution," Journal of Grid Computing, vol. 14, 2016, pp. 265-282.

[18] Swarm mode overview, https://docs.docker.com/engine/swarm/

[19] CoreOS, https://coreos.com

[20] Kubernetes, https://kubernetes.io

[21] Apache Mesos, http://mesos.apache.org

[22] A. Verma, et al. "Large-scale cluster management at Google with Borg." Proceedings of the Tenth European Conference on Computer Systems. ACM, 2015.

[23] C. Delimitrou, and C. Kozyrakis, "Quasar: resource-efficient and QoS-aware cluster management,", ACM SIGPLAN, vol. 49, 2014.

[24] D. Dmitry, M. Haney, and H. Tufo, "Highly Available Cloud-Based Cluster Management," 15th IEEE/ACM International Symp. Cluster, Cloud and Grid Computing (CCGrid), IEEE, 2015.

[25] J. Park, and J. Hahm, "Container-based Cluster Management Platform for Distributed Computing," Proc. International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA), 2015.

[26] P. Pietzuch, J. Ledlie, J. Shneidman, M. Roussopoulos, M. Welsh, and M. Seltzer, "Network-Aware Operator Placement for Stream-Processing Systems", 22nd International Conference on Data Engineering (ICDE '06), pp. 49–53, IEEE Computer Society, 2006.

[27] V. Cardellini, V. Grassi, F. Lo Presti, and M. Nardelli, "Distributed QoS-aware Scheduling in Storm", 9th ACM International Conference on Distributed Event-Based Systems, pp. 344-347, ACM, 2015.

[28] M. Hirzel, R. Soule, S. Schneider, B. Gedik, and R. Grimm, "A Catalog of Stream Processing Optimizations", ACM Computing Surveys, vol. 46, Mar. 2014, pp 1–34.

[29] T. De Matteis, G. Mencagli, "Proactive elasticity and energy awareness in data stream processing", Journal of Systems and Software, vol. 127, 2017, pp 302-319.