Performance-Power Exploration of Software-Defined Big Data Analytics: The AEGLE Cloud Backend

Georgios Zervakis, Sotirios Xydis and Dimitrios Soudris
National Technical University of Athens, 9 Heroon Polytechneiou, Athens Greece
{zervakis, sxydis, dsoudris}@microlab.ntua.gr

***Abstract*—In this paper, we present the design and analyze the performance-energy characteristics of a software-defined infrastructure targeting Big Data analytics workloads. This software-defined Big Data framework forms the data analytic platform adopted in AEGLE[1], an European H2020 funded project for healthcare analytics. The developed framework utilizes state-of-art open source solutions and it is very flexible to enable the definition and automatic deployment of differing SPARK over Hadoop cluster configurations as analytics engines. In this paper, we exploit this flexibility of our software defined infrastructure to explore the performance-energy trade-offs of Big Data analytics under variable resource allocation scenarios. Specifically, we show that with respect to our local infrastructure, i.e., two Intel Xeon E5-2658A servers with 128GB RAM each, virtual cluster configurations with many nodes achieve the highest performance, while virtual cluster with high available RAM memory are more power efficient, exhibiting higher instructions per cycle (IPC) per kilojoule values.**

I. Introduction

Modelling biological phenomena is typically very complex and has always been understood to be a computationally intensive process. However, the applicability of Big Data techniques on biological and health-based data, naturally quite complicated and difficult to collect, is still limited. In order to draw meaning from the exponentially increasing quantity of healthcare data, a shift towards a big data perspective is proposed, utilizing technologies capable of processing massive amounts of data efficiently and securely. Collecting and aggregating anonymous data from geographically dispersed locations makes it possible to construct statistically meaningful databases, based on which macroscopic reasoning can be made, rather than solely focusing on the individual and associated pathology.

Nowadays, there is an obvious gap in the area of big data analytics for Health Bio-data. Data-driven services are still needed to cater for the data versatility, volume, velocity and veracity within the whole data value chain of healthcare analytics. A true opportunity exists to produce value out of big data in healthcare with the goal to revolutionize integrated and personalised healthcare services. Although Information and Communication Technology (ICT) makes advancements to give solutions in big data volume and velocity issues, the healthcare industry has been hesitant in embracing Big Data. AEGLE is a European funded project under the H2020 programme, targeting to address the aforementioned open issues by implementing a full data value chain to create new value out of rich, multi-diverse, big health data. It aims to generate value from healthcare data with the vision to improve translational medicine and facilitate personalized and integrated care services overall improving healthcare at all levels, to promote data-driven research across Europe and to serve as an enabler technology platform.

Although AEGLE's infrastructure is composed of several software components, described in Section V, in this paper we focus our attention on the software infrastructure implementing the Big Data analytics engine, on top of which the specialized analytics and workflows will be executed. The Big Data analytics engine of AEGLE has been developed as a software-defined infrastructure, concerning both the computing and networking components, and it targets to cloud based virtual clusters. It eases its relocation to alternative cloud provider or virtual clusters in general , as well as it enables exploration studies to be performed over the infrastructure characteristics. In this paper, we present the design of the aforementioned software-defined Big Data infrastructure and we analyze its performance-energy characteristics.

More specifically, the developed framework utilizes state-of-art open source solutions and it is very flexible to enable the definition and automatic deployment of differing SPARK [1] over Hadoop [2] cluster configurations as analytics engines. We exploit this flexibility to explore the performance-energy trade-offs of Big Data analytics under variable resource allocation scenarios. Through extensive experimentation over a set of scaled machine learning analytics, originated from the well known SPARK bench suite [3], we show that for the specific type of workloads, allocating virtual clusters with a lot but relatively small sized servers forms efficient configuration in comparison to utilizing few but bigger in size resources.

[1]This research is partially supported by the E.C. funded program AEGLE under H2020 Grant Agreement No: 644906, http://www.aegle-uhealth.eu

Furthermore, by analyzing the behavior of the infrastructure under scaled data-size configuration, we show the high sensitivity of performance not only on the allocation of the computational resources but also on the available network bandwidth, which has to be taken into account during system design and optimization phase.

The rest of the manuscript is organized as follows: Section II presents the vision of AEGE, its goals and the expected impact on the scientific and health-care domain. Section III briefly describes the uses case scenarios from the medical domain used in AEGLE, while section IV gives a detailed description of the AEGLE's system architecture and its sub-components. Section V presents the software components defining AEGLE's Big Data Framework and the automatic deployment flows developed in AEGLE implementing a Software Defined Big Data Platform. Finally, in Section VI we experimentally evaluate and characterize the Spark execution with respect to our physical local infrastructure using the AEGLE's virtualization and deployment flows and Section VII concludes our work.

II. The vision of AEGLE

AEGLE's mission is to realize an European business ecosystem for health-care stakeholders, industry and researchers for creating out-of-box knowledge in order to provide cloud and HPC data services and support new products that will improve health. The project builds upon the synergy of heterogeneous High Performance Computing (HPC), Cloud and Big Data computing technologies for delivering optimized analytic services on Big-Bio Data application use cases from the medical and health-care domain. In this paper, we describe in depth the three target Big-Bio Data applications as well as the key technologies to be utilized within AEGLE for delivering accelerated health-care analytics.

Data driven generation of new medical knowledge premises the support of personalized medicine, thus effective treatments for each individual instead of the average patient, forms one of the missions of big data in health. Several European initiatives [4] have already pinpointed the importance and usefulness of health-care big data, e.g. to predict the outbreak of an epidemic etc. The scope of AEGLE project is to develop and provide a Big Data analytics infrastructure that will deliver integrated ICT services for health-care, to enable and promote research and innovation activities, as well as to serve as a strategic pillar for business development in the field for big data analytics for health-care. AEGLE solution targets to address the whole data value chain for health based on: cloud computing and Big Data technologies for scalable ICT services, HPC infrastructures for computational acceleration and advanced visualization techniques and contribute in the area of analytics for Health Bio-data.

Specifically, AEGLE's analytics services will offer an experimental big data research platform to data scientists, workers and data professionals across Europe. The platform consists of a large pool of semantically-annotated healthcare data, a set of libraries implementing state-of-the-art big data analytics methods including the local level big data analytics AEGLE services and APIs for federating with public and private data sets. Advanced visualization tools will be implemented by AEGLE as an instrument for gaining new knowledge and expertise, advancing the European know-how in health-care big data analytics, by allowing data scientists to steer the cloud level analytics mechanisms with their own insights. Large industries and SMEs across Europe will be given the ability to use AEGLE in order to deploy and assess the validity of their innovative data analytics solutions which aim at creating new value in the field of healthcare.

III. Brief description of AEGLE use cases

AEGLE system will target to and be validated over the following use case scenarios:

- **Chronic Lymphocytic Leukemia (CLL):** CLL is a chronic, incurable disease, leading to great distress for patients and their families as well as huge costs for the health care system. Integrative analysis will be performed to address complex clinical questions and scenarios associating phenotypic data with personal genetic profiles derived from both conventional but also, critical to the mandate and objectives of AEGLE, high-throughput approaches. In addition, AEGLE will offer the possibility of proposing and evaluating health interventions towards the goal of personalized medicine e.g. identifying groups with specific profiles that will be considered as eligible or ineligible for certain treatments and, at the same time, evaluating the cost of this intervention.
- **Intensive Care Unit (ICU):** In an ICU context, patient biosignals are continuously monitored and displayed towards recognizing alerting events. The recordings of clinical, laboratory data and physiologic waveforms could be analysed and displayed in an easy-to understand manner for clinicians. AEGLE aims to provide a set of scalable and automated analysis of the fast changing multi-dimensional functions of variables for the detection of unusual, unstable or deteriorating states in patients. In this respect, early and personalized treatment will be feasible using AEGLE technology for higher survival in ICUs around European Hospitals.
- **Type 2 Diabetes (T2D):** The risk of developing T2D can be increased by various factors; usually a mixture of modifiable and non-modifiable elements of age, weight, genetics and ethnicity. The AEGLE system will focus on analysing the inter-dependences of the factors including medication that are known to have a detrimental effect in type 2 diabetes to give a prediction on the potential deterioration. This would enable intervention to enable reduction of mortality, complications and hospitalization that would all lead to reduction in overall health costs.

IV. AEGLE ARCHITECTURE

AEGLE starts from real life conditions in the three medical cases, which include clinical and biomedical research. It is thus important to recognize not only the data that are produced at each organization, but also the procedures and analysis pipelines that are in place. Therefore AEGLE infrastructure defines the local and the cloud domain, as well as a loose coupling between them. The local domain refers to the organizational private space, where routine use of data takes place, as well as analysis without privacy concerns. The latter refers to the cloud space, where data from multiple organizations are uploaded to common storage and analysis infrastructure that can be shared, combined and analyzed, providing the necessary access and privacy-related pre-processing. The cloud domain will provide more sophisticated data management, applying robust and scalable data management techniques that are potentially lacking at the local domain, due to organizational issues among others. More specifically: The **AEGLE's local level** implements the data generation, the anonymization for data privacy, the interfaces for data uploading to cloud, as well as a set of analytics services/applications. The latter are either specialized for local use or exhibiting strict real-time constraints, processing of large volumes of fast generated and multiple-formatted raw data originating from patient monitoring services deployed within a healthcare unit, complemented with dedicated medical databases. The stakeholders of the local level analytics are healthcare units/systems and of course the patients are ultimate the beneficiaries that will benefit from the advanced treatment modalities enabled by adopting the local analytics services. For example, in the ICU scenario, a prompt reaction to detected instabilities or abnormal behaviour of the patient's status could significantly help to save the lives of patients being treated within the ICU. The **AEGLE's cloud level analytics** services will offer an experimental big data research platform to data scientists, workers and data professionals across Europe. The cloud platform consists of a large pool of anonymized healthcare data, a set of libraries implementing state-of-the-art big data analytics methods including the local level big data analytics AEGLE services and APIs and the corresponding visualization tools.

A. AEGLE subsystem description

The cloud sub-system (CSB) components of AEGLE are the following:

- **CSB1. Identity management services:** This subsystem implements all the user authentication and authorization services for accessing the AEGLE cloud services. It operates at the front-end of AEGLE cloud portal and enables users to link with the requested cloud resources, data and analytics in a secure and authorized manner.
- **CSB2. User interaction services:** This subsystem implements all the logic and user interfaces for users to navigate to AEGLE cloud services and setup their data, analysis and workflows. It is in a close cooperation with the rest of AEGLE cloud subsystems acting as a controller of the requested functionalities.
- **CSB3. Anonymization service and cloud data storage:** This cloud service includes all the functionality and interfaces for enabling the setup of local anonymization services to be installed to the local sites and configure the data mapping according to the structure of the local databases. It also implements the cloud data storage mechanisms of the anonymized data uploaded to the AEGLE cloud platform
- **CSB4. Analytics libraries** Analytics libraries subsystem implements all the tools and specialized algorithms for Big Data analytics at the cloud level. All the analytics will be available both as customized pipelines and in the next version of AEGLE cloud platform also as standalone operators able to be combined for structuring more complex analytic pipelines on demand. Specialized analytics fitting to the requirements of each of the AEGLE's use case, i.e. CLL. ICU and DII, will be included as modular components in this subsystem. The analytics will be integrated and work in close cooperation with both the data visualization libraries, the acceleration engines as well as the underlying Big Data framework in order to exploit the performance and scalability advancements and features of the latter subsystems.
- **CSB5. Data visualization libraries:** This subsystem includes the visualization functions for the analytic algorithms defined in the previous subsystem. The user will be able to setup and configure the desired visualizations on a dashboard manner. Except from specialized visualization formats, it will also implement some basic data visualization primitives usually employed in data analytics frameworks.
- **CSB6. Accelerated analytics:** This subsystem includes all the compiler, system level tools and drivers as well as the computing platforms for supporting a cluster installation based on Maxeler acceleration machines [5]. On top of this infrastructure, it will implement the accelerated versions of algorithms found in the cloud analytics libraries.
- **CSB7. Cloud and Big Data infrastructure:** This subsystem implements all the mechanisms for the AEGLE cloud virtual cluster configuration and deployment. The deployed virtual clusters will be automatically configured to support and instantiate AEGLE's Big Data framework for supporting scalable analytics and efficient resource management of the underlying infrastructure. The specific component will expose a rich set of interfaces to enable application programmers, i.e. analytics, data visualization and acceleration engineers to efficiently map their applications on the cloud infrastructure exploiting its build-in scalability primitives. It focuses mainly on the storage and

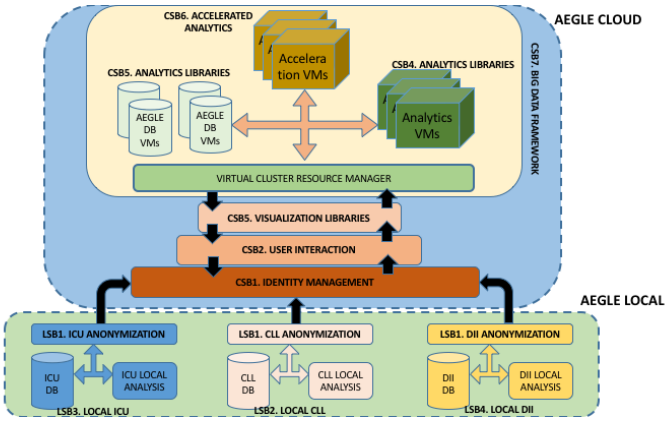


Fig. 1. AEGLE architecture and subsystems.

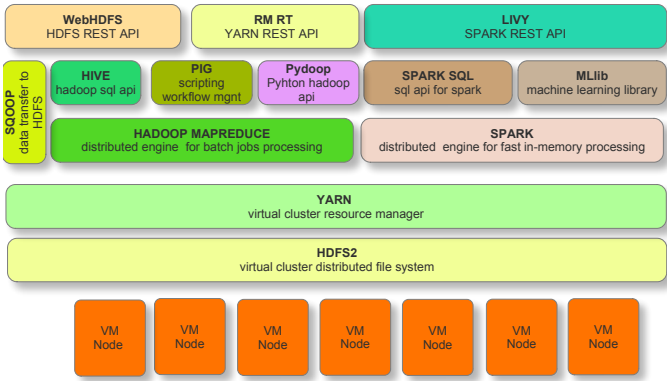


Fig. 2. AEGLE's Big Data framework: SW components and hierarchy.

computation resources of the AEGLE cloud platform and their effective interaction

B. AEGLE's Information Flow

Fig. 1 shows the overall AEGLE system architecture aggregated with the local and cloud domains and annotated with the previously described subsystems. A typical information flow is also depicted starting from the local AEGLE infrastructure, passing to AEGLE cloud and vice versa. The local user is able either to upload data to AEGLE cloud, passing through the anonymization modules or to directly perform cloud level analyses on data already available online. Each time a local user is connecting to AEGLE's portal his/her credentials are authenticated in order to acquire access to AEGLE online services. Then through the user interaction subsystem, the user can select through a set of options, e.g. uploading data, perform analytics on existing cloud data etc. In case of data upload, a configurable anonymizer will be installed at the local infrastructure, enabling the generation of a local anonymized copy of the data that will be safe for upload in the AEGLE cloud. Data upload will then instruct the AEGLE cloud storage to accommodate the new data. Cloud level workflows (analytics and visualization) are invoked on the AEGLE's Big Data infrastructure, which consists of the analytics libraries, the data storage nodes, the computation and acceleration nodes and software (SW) layer to enable an automatic deployable cluster with scalable execution engines. According to the requested analytic, e.g. accelerated on Maxeler nodes or pure SW implementation, SQL based or file based etc., a set of appropriate cluster nodes and mechanisms will be allocated to perform the analytic workflow. After completion, the data to be visualized are returned to the corresponding subsystem that depicts them according to the requested format and the analysis results are delivered to the user.

V. Software Defined Big Data Infrastructure

Fig. 2 depicts the main SW modules/components defining the AEGLE's Big Data framework. Specifically, it consists of the following components:

- HDFS2: It implements the Hadoop distributed file system efficient data storage, access and resilience in AEGLE's cloud.
- YARN: It implements the actual resource manager of the HDFS cluster [6] .
- Hadoop MapReduce: It implements the Hadoop framework implementing the map-reduce programming model for parallel and scalable data processing [7] .
- Pig: It implements a scripting interface for describing workflow utilizing the scalable Hadoop MapReduce framework [8].
- Hive: It implements the SQL API for querying HDFS stored data exploiting the parallelization primitives exposed by the Hadoop MapReduce [9].
- Sqoop: It implements an API for efficiently transferring data in/out HDFS from third party data sources, e.g. external SQL databases [10] .
- SPARK: It implements a scalable execution engine supporting data caching, also implementing the map-reduce programming model, for data intensive workloads, more specialized for iterative applications [1].
- SPARK SQL: Same as HIVE, it implements the SQL API for querying HDFS stored data exploiting SPARK's execution engine [1].
- MLlib: SPARK's library exposing efficient and scalable implementations of machine learning algorithms [1].
- WebHDFS REST API : The HTTP REST API supports the complete FileSystem interface for HDFS [11].
- ResourceManager REST API: It allows the user to submit jobs to YARN and get information about the cluster - status on the cluster, metrics on the cluster, scheduler information, information about nodes in the cluster, and information about applications on the cluster [12].
- Livy : Livy is an open source REST interface for interacting with Spark from anywhere. It supports executing snippets of code or programs in a Spark context that runs locally or in YARN. It supports i) Interactive Scala,

Python and R shells, ii) batch submissions in Scala, Java, Python, iii) multi users can share the same server (impersonation support), iv) job submission from anywhere with REST [13].

The implemented Big Data framework supports two scalable execution engines, i.e. Hadoop MapReduce and SPARK. While Hadoop MapReduce exhibits good scalability w.r.t. data volume and underlying resources, it is more efficient for batch workloads that perform large analyses defined by several chained map-reduce phases. On the other hand, SPARK engine exhibits higher performance figures, up to 100x for iterative workloads [1]. Deploying SPARK over HDFS enables most of the beneficial scalability features present in Hadoop to be utilized also in SPARK-based applications. In addition, it provides a richer API with respect to the supported programming languages. The existence of both execution frameworks in the deployed Big Data framework enables high flexibility for AEGLE applications to be customized according to their needs to the API provided by both frameworks. In the next sections, we describe in more details the cloud infrastructure along with its automation primitives as well as the components defining the Big Data framework.

A. AEGLE's Automatic Deployment Flows

Two automatic deployment flows are implemented within AEGLE targeting software defined big data infrastructures. The fist flow creates a virtual cluster of Virtual Machines (VMs) on a given number of physical host machines. The AEGLE's Virtual Cluster Flow takes as input arguments the desired cluster size n, the numbers of cores and RAM memory of the VMs, the number of the host servers k, and the IPs of the latter ones. Moreover, the selection of different disk sizes is provided for generated the VMs. For the generation of the VMs, VirtualBox [14] is used. The VMs run Linux flavor Ubuntu server and their network adapter is set to bridged, enabling the communication of guest VMs running in different host computers. The Virtual Cluster Flow detects the network interfaces (Ethernet network adapters) of each host and creates n/k VMs in each host. Assuming that a host has e network interfaces, $n/k/e$ VMs are connected to each network interfaces. Detecting the network interfaces and binding the VMs to different host Ethernet ports, leverages the hardware capabilities of the host machines as it enables balancing the network traffic passing from each interface and reduce the communication latency of the VMs. Note that e may not be the same in the different hosts. If only one network adapter is available, then all the VMs are connected to that adapter. Fig. 3 depicts an example of a n-size virtual cluster creation on k host machines.

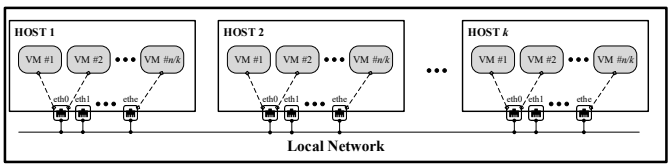


Fig. 3. Example of automatic virtual cluster creation using the Virtual Cluster Flow. n VMs are created in k host servers with e network interfaces each.

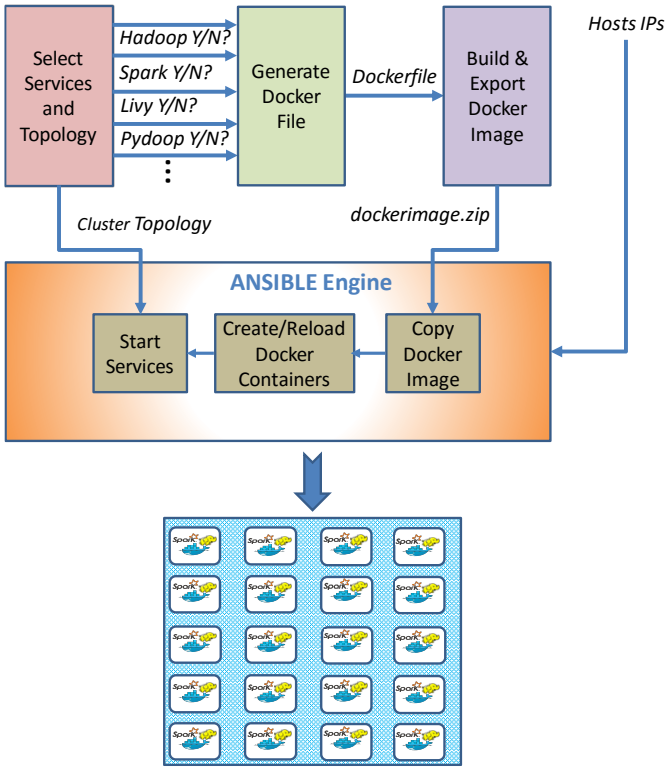
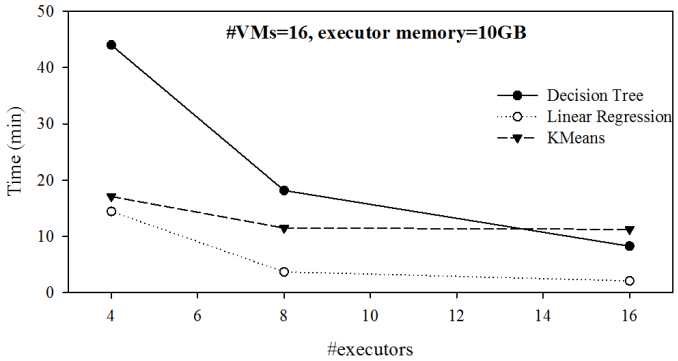
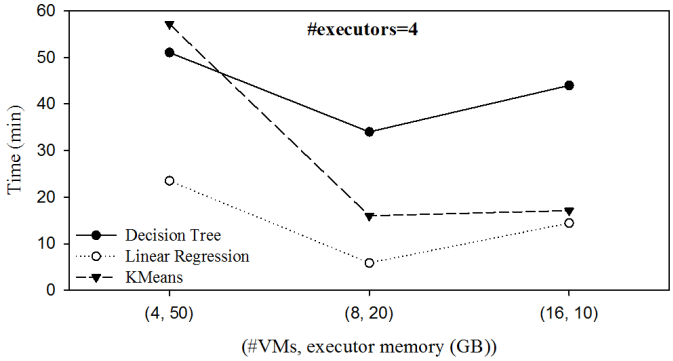


Fig. 4. AEGLE's Cloud Deployment Flow

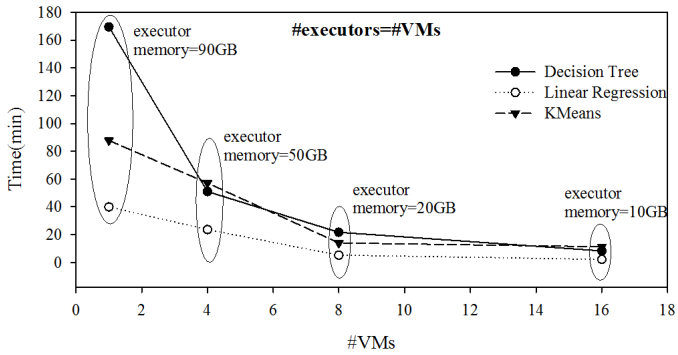
The second of the two flows developed in AEGLE is the Cloud Deployment Flow that automatically deploys the AEGLE's Big Data Framework on a given number of host machines. The hosts may be either physical machines or VMs and more than one Hadoop cluster nodes can be deployed in every host. Ansible [15] and Docker [16] are used for the cluster deployment. Ansible is used to setup accordingly the host machines, transfer, and start the Docker images. Docker is used to create user specific images containing the desired SW modules. Ansible enables the automation of the cluster deployment and the configuration of the host machines. Docker offers lightweight sand-boxed virtualization, portability through platform independent deployment, efficient host resource usage, and fast infrastructure reconfiguration. The Cloud Deployment Flow is depicted in Fig. 4. The user selects the desired SW modules, i.e., Hadoop, Spark, Hive, Sqoop, REST APIs etc., and the corresponding Docker image is generated. Then the user defines the desired cluster topology, i.e., cluster size, number of docker images per host, and the hosts of the Hadoop name node and resource manager, sets some cluster specific parameters, i.e., Hadoop, YARN, and Spark specific configurations, and provides the IPs of the host



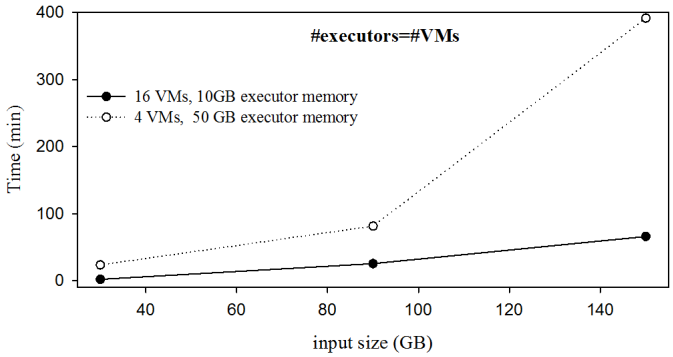
(a) Performance variation with respect to the number of executors for a fixed cluster configuration.



(b) Performance variation with respect to different cluster sizes for fixed number of executors.



(c) Performance variation with respect to different cluster configurations. In every virtual cluster the number of executors is set to be equal the number of the cluster nodes.



(d) Performance comparison of two different cluster configurations for increasing input sizes. The first cluster features many nodes and executors, while the second high executor memory.

Fig. 5. Exploration of Spark performance for different virtual cluster configurations.

machines for the cluster deployment. Finally, using Ansible the flow, sets up the hosts accordingly, transfers the docker images to the hosts and starts their services. For the networking between the Docker containers running the AEGLE's Big Data Framework Weave [17] is used. Weave provides a software defined network, creating a virtual network that connects Docker containers across multiple hosts and enables their automatic discovery.

VI. Experimental analysis

In this section, through extensive experimentation, we examine the Spark's sensitivity on differing cluster configurations and execution parameters in order to optimal tune the AEGLE's Big Data platform. Given a physical platform (server, cluster, etc.) we explore which is the optimal virtual cluster configuration, i.e., number of VMs, RAM and cores per VM, for running Spark applications. In this analysis, Spark runs over HDFS in YARN-cluster mode, as in the AEGLE system. Virtual Clusters with different configurations are created using AEGLE's Virtual Cluster Flow and then using AEGLE's Cloud Deployment Flow, a Hadoop cluster with the Spark execution engine is deployed on the created virtual cluster. Our experimental setup consists of two Supermicro servers featuring two Intel Xeon E5-2658A v3 processors (12 cores, 24 threads per processor), 128 GB of RAM memory and two network controllers each. The virtual clusters are deployed on these two servers and in total 40 virtual cores and 240 GB of RAM can be assigned to the VMs (we reserve some resources for the host machine). Three benchmarks are considered for our evaluation: the Decision Tree, the Linear Regression and the KMeans benchmarks from the Spark Bench suite [3]. The evaluation metrics, used in this analysis, are the time (in minutes) required to execute the benchmarks and the energy consumption (in kJ) of their execution. These metrics depend on the virtual cluster topology and are used to define the optimal virtual cluster configuration with respect to the physical platform that is deployed on. For the energy consumption measurements, the energy consumed in each server is measured and then these measurements are summed to obtain the total energy consumption of the virtual cluster for the benchmark's execution. Finally, for every benchmark an input size of 30GB is used, unless is specified differently.

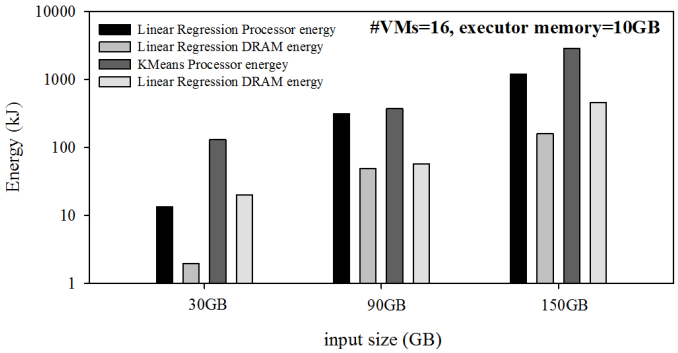
Table I summarizes all the examined configurations. For every configuration it is specified the number of the VMs, i.e., the number of the Hadoop cluster data nodes. In every configuration an additional VM serving as Name Node and Resource Manager is also created. Moreover, the cores and the RAM memory of the VMs are presented, as well as the

TABLE I
THE DIFFERENT CONFIGURATIONS USED FOR OUR EVALUATION

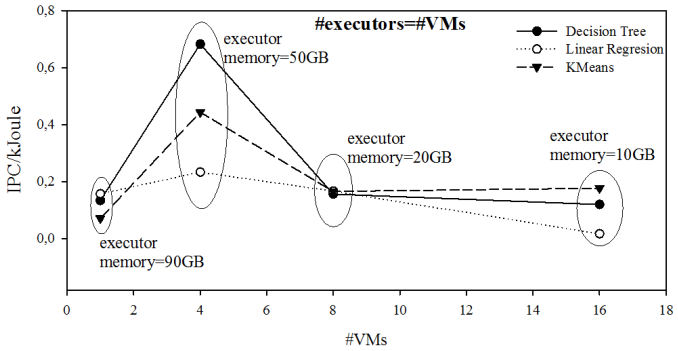
#VMs	VM Cores	VM RAM (GB)	#Executors (SPARK)	Executor Memory (GB) (SPARK)
1	32	120	1	90
4	8	60	4	50
8	8	30	4,8	20
16	4	15	4,8,16	10

Spark configuration values: number of executors and executor memory that are used for the execution of the benchmarks. The Spark's driver memory is set to 4GB for all the configurations and the number of executors is set to be less or equal to the number of the available VMs. The Spark executor cores are set equal to the number of the VM cores in every configuration. Finally, note that the memory that can be assigned to every Spark executor must be (considerably) smaller than the RAM memory of the VMs.

In Fig. 5a the sensitivity of Spark's performance with respect to the allocated executors is examined. The virtual cluster configuration of Fig. 5a consists of 16VMs with 15GB RAM and 10GB memory allocated for every Spark executor. It is shown that as the number of the executors increase the Spark performance increases. More specifically, the Decision Tree and the Linear Regression are highly benefited from the Spark executors increase. Doubling the executors speeds ups the execution of the Decision Tree by 2.4x and that of the Linear Regression by 3.9x and quadrupling the executors offers 5.3x and 6.9x higher performance, respectively. On the other hand, for the KMeans benchmark, doubling the executors speeds up its execution only by 1.5x, while increasing the executors from 8 to 16 does not offer any significant speed up. The Kmeans speed up is limited due to its interaction with the HDFS as the input data are clustered and then they are written back to the HDFS. In Fig. 5b the number of executors is fixed (equal to 4) and the influence of the cluster size on the Spark performance is examined. In this figure different cluster sizes are examined. However, as we are limited from our physical platform, increasing the number of the VMs decreases their RAM memory and as a result the Spark's executors memory. Fig. 5b shows that increasing the cluster nodes from 4 to 8 speeds up the execution of the benchmarks by 3x on average. On the other hand, increasing the nodes from 8 to 16 decreases their performance. As the number of executors is fixed to 4, in the 16 nodes cluster the memory allocated for the Spark executors is only 40GB. This value is significantly smaller than the 80GB available in the 8 nodes cluster. As a result, having large clusters but limited number of executors is not performance efficient. From Fig. 5a-5b the Spark performance benefits from the executors and cluster size increase but is limited by the available memory. In Fig. 5c, we examine virtual clusters with similar memory capacity but different cluster-size. In the cluster configurations examined in



(a) Processor and DRAM energy consumption for increasing input sizes.



(b) IPC per kilojoule for different different cluster configurations

Fig. 6. Exploration of Spark energy consumption for different virtual cluster configurations and input sizes.

Fig. 5c, the number of Spark executors is set to be equal to the cluster size (one executor per VM). Therefore, increasing the cluster size increases the Spark executors. As presented in Fig. 5c, the bigger the cluster size (and thus the available executors), the bigger the performance increase. For example, moving from a 4 node cluster size with 200GB memory (in total) allocated for the Spark executors, to a 16 nodes cluster with 160GB total executors memory speeds up the benchmarks execution by 7.5x on average. In order to verify the behavior exhibited in Fig. 5c, we examine if it is valid for different input sizes. The Linear Regression benchmark is used and two cluster configuration are tested. The optimal one resulted in Fig. 5c (16 node virtual cluster) and a 4 node cluster with high executor memory (50GB per executor). As depicted in Fig. 5c, even for large inputs, the Spark's performance is influenced more by the number of the executors and the cluster size increase rather than the available memory increase.

Concerning the energy consumption, for the optimal configuration of Fig. 5c and the data sizes examined in Fig. 5d, we examine in Fig. 6a the energy consumption of the processors and the RAM memory for the execution of Linear Regression and KMeans benchmarks. From Fig. 5c, the energy spent for computations highly dominates the energy spent for memory accesses. On average, the energy consumption of the processors is 6.7x higher than the respective of the RAM memory. Finally, Fig. 6b depicts the instructions per cycle

(IPC) per kilojoule of the host servers for the configurations examined in Fig. 5c. It is shown that the configuration with 4 nodes and 50GB executor memory achieves the highest IPC/kJ values. This is also confirmed from Fig. 6a where we deduced that the Spark energy consumption is little affected from the energy consumption due to memory accesses. The configuration with 4 nodes uses less executors and more memory comparing with the other presented configurations leading to higher IPC per kilojoule ratio. The configuration with one node and 90GB of executor memory exhibit the worst IPC/kJ. Although this configuration features a lot of memory, the existence of only one executor significantly slows down the Spark execution, increasing, thus, the execution time and the consumed energy.

VII. Conclusion

In this paper, we presented the design and analysis of a software-defined Big Data analytics framework that forms the cloud backend infrastructure of AEGLE system. In addition, we present an integrated tool that enables the automatic configuration and deployment of the developed Big Data analytics framework, thus exploiting the internal configurability of the proposed software-defined infrastructure. Through a rich set of experimental results, we explore and analyse the performance and energy trade-offs of real-life workload scenarios on Big Data analytics platforms, showing that optimal system configurations are amenable to complex interactions between resource and memory allocations.

References

[1] "Apache spark." [Online]. Available: http://spark.apache.org
[2] "Apache hadoop." [Online]. Available: http://hadoop.apache.org
[3] M. Li, J. Tan, Y. Wang, L. Zhang, and V. Salapura, "Sparkbench: A comprehensive benchmarking suite for in memory data analytic platform spark," in *Proceedings of the 12th ACM International Conference on Computing Frontiers*, 2015, pp. 53:1–53:8.
[4] "Big data: What is it and why is it important?" [Online]. Available: http://ec.europa.eu/digital-agenda/en/news/ big-data-what-it-and-why-it-important
[5] "Maxeler." [Online]. Available: http://www.maxeler.com/
[6] "Yarn." [Online]. Available: https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html
[7] "Hadoop mapreduce." [Online]. Available: https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html
[8] "Apach hive." [Online]. Available: https://hive.apache.org
[9] "Apache sqoop." [Online]. Available: http://sqoop.apache.org
[10] "Apache pig." [Online]. Available: https://pig.apache.org
[11] "Webhdfs rest api." [Online]. Available: https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/WebHDFS.html
[12] "Resourcemanager rest api." [Online]. Available: https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/ResourceManagerRest.html
[13] "Livy rest api." [Online]. Available: https://github.com/cloudera/hue/tree/master/apps/spark/java
[14] "Virtual box." [Online]. Available: https://www.virtualbox.org
[15] "Ansible." [Online]. Available: https://www.ansible.com
[16] "Docker." [Online]. Available: https://www.docker.com
[17] "Weave." [Online]. Available: https://github.com/weaveworks/weave