ALOJA: a Systematic Study of Hadoop Deployment Variables to Enable Automated Characterization of Cost-Effectiveness

Nicolas Poggi, David Carrera, Aaron Call,
Sergio Mendoza, Yolanda Becerra, Jordi Torres,
Eduard Ayguadé, Fabrizio Gagliardi and Jesús Labarta
Barcelona Supercomputing Center (BSC)
Universitat Politècnica de Catlalunya (BarcelonaTech)
Barcelona, Spain

Rob Reinauer, Nikola Vujic[1],
Daron Green and José Blakeley
Microsoft Corporation, Microsoft Research (MSR)
Redmond, USA
Microsoft Development Center Serbia (MDCS)[1]
Belgrade, Serbia

***Abstract*—This article presents the ALOJA project, an initiative to produce mechanisms for an automated characterization of cost-effectiveness of Hadoop deployments and reports its initial results. ALOJA is the latest phase of a long-term collaborative engagement between BSC and Microsoft which, over the past 6 years has explored a range of different aspects of computing systems, software technologies and performance profiling. While during the last 5 years, Hadoop has become the *de-facto* platform for Big Data deployments, still little is understood of how the different layers of the software and hardware deployment options affects its performance. Early ALOJA results show that Hadoop's runtime performance, and therefore its price, are critically affected by relatively simple software and hardware configuration choices e.g., number of mappers, compression, or volume configuration. Project ALOJA presents a vendor-neutral repository featuring over 5000 Hadoop runs, a test bed, and tools to evaluate the cost-effectiveness of different hardware, parameter tuning, and Cloud services for Hadoop. As few organizations have the time or performance profiling expertise, we expect our growing repository will benefit Hadoop customers to meet their Big Data application needs. ALOJA seeks to provide both knowledge and an online service to with which users make better informed configuration choices for their Hadoop compute infrastructure whether this be on-premise or cloud-based.**

The initial version of ALOJA's Web application and sources are available at http://hadoop.bsc.es

I. Introduction

During the last years, Hadoop has established itself as the *de facto* framework for Big Data processing deployments and continues to grow with over a 58% compound annual growth rate [11]. However, despite this rapid growth, the Hadoop architecture implements an extremely complex distributed execution environment. Different HW components impact per node performance in a variety of ways which also differ in their impact by workload type. Numerous software parameters exposed by both Hadoop and the Java runtime can have pronounced impacts and the different deployment patterns of on premise servers vs cloud based deployments add yet another layer of complexity. Therefore Hadoop often requires manual, iterative and time consuming benchmarking and fine tuning over a myriad of possible configuration and deployment options. To illustrate this complexity Figure 1 presents the search space for evaluating the cost-effectiveness of a particular setup.

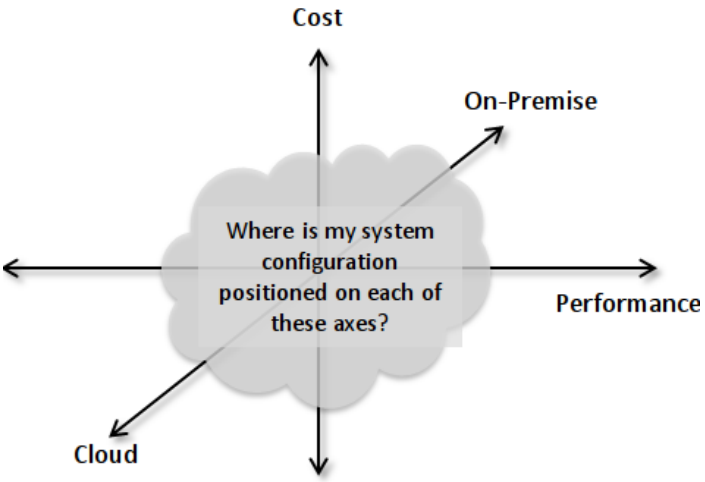

Fig. 1. A cloud of points for Cost vs. Performance vs. Cloud or On-premise

This article presents the ALOJA project, its goals, platform, tools and initial results while performing a systematic study of Hadoop deployment variables by benchmarking and tuning Hadoop in different architectures. ALOJA is an initiative of the Barcelona Supercomputing Center (BSC) in which both Microsoft Product groups and Microsoft Research (MSR) are participating to explore upcoming hardware architectures and building automated mechanism for deploying cost-effective Hadoop clusters. ALOJA's initial approach is to create the most comprehensive (to date) open public Hadoop benchmarking repository. Comparing not only software configuration parameters, but also *current*, to new available hardware including SSDs, InfiniBand networks, and Cloud services. The ALOJA project also determines the cost of each possible setup along with the run time performance with a view to ultimately offering an automated configuration recommendation for a given workload. In this way, serving not only as a reference guide for designing new Hadoop clusters. Exploring parameter relationship, but to ultimately take informed decisions

for reducing the TCO and improving new and existing data processing infrastructures.

A. Motivation

Most of the industry's efforts during past years have focused initially into building scalable data processing frameworks i.e., Hadoop and its derived services. Later, the efforts have focused in the adoption of Hadoop by companies, and the development of Map/Reduce applications. Lately, as more Hadoop clusters are deployed and running, tuning their performance and managing data is gaining importance in focus. Some studies show that Hadoop's execution performance can be improved at least by 3x from the default configuration for most deployments [7–9]. The large number of configuration options and parallel data processing in Hadoop, creates a complex runtime and a challenge to characterize the resource consumption, performance, and costs.

Moreover, as it happened over the last decade with open-source relational databases i.e., MySQL, the Hadoop ecosystem is quite large and spread. It is currently dominated by several vendors that offer their custom distributions of Hadoop, who usually offer custom patches and configuration changes to the default Apache Hadoop open-source distribution. Vendor changes and optimizations rarely get pushed to the main distribution, which by default runs at sub-optimal condition [7–9]. There is also evidence that Hadoop performs poorly with newer and scale-up hardware [1]. Scaling out in number of servers can usually improve performance, but at increased cost, power and space usage [1]. These situations and available services make a case to reconsider scale-up hardware and new Cloud services from both a research and an industry perspective.

II. The ALOJA Project

Project ALOJA borns as an attempt to provide solutions to an every time more important problem for the Hadoop community, which is the lack of understanding of what parameters, either software or hardware, determine the performance of Hadoop workloads, therefore its costs. Additionally, as Hadoop deployments become more common, they can be found in a diversity of operational environments, comprising from low-end commodity clusters, to high-end data appliances and including all types of Cloud-based solutions at scale. This fact results in a growing need for the community to understand how different operational parameters, such as the VM size used in Cloud deployments, affect the cost-effectiveness of Hadoop workloads.

The project is structured in different phases: an initial phase, almost complete by the time of writing this paper, aims to create a benchmarking platform for continuous execution of Hadoop workloads across different software, hardware and operational variables: the ALOJA platform. Besides the benchmarking components, an online Web application (see Section III-B2) is envisioned as the entry point for data scientists to explore the results collected from the executions, providing insights on the obtained results through continuously evolving data views. The following phase of the project is the development of performance models that derive from the collected data, both following supervised learning strategies and formal analytical performance modeling. Finally, the project aims to leverage the developed models and the benchmarking information to provide automatic means to characterize the cost-effectiveness of a Hadoop workload across several configuration options. This information should guide users to device what deployment options are the most adequate to maximize the cost-effectiveness of their workloads.

Project ALOJA is an initiative of the Barcelona Supercomputing Center (BSC), a center of excellence with over 6 years of Hadoop research experience [3]. The project is supported, in part, by Microsoft Corporation and includes technical contribution from product teams, financial support and awards for resources as part of the Azure4Research programme. In addition, this project serves as a base for experimentation in several research lines for research within BSC, as well as to other research and industry groups that might leverage the repository and tools.

A. Methodology and Road-Map

The effort for ALOJA has been partitioned into 3 distinct phases of execution:

1) Phase 1: In the initial phase of the project —which has been underway for approximately 8 months—, we are executing a systematic study of performance results across a range of hardware components, software parameters values and solution deployment patterns.

Hardware components under examination include:

- Number and type of Storage volumes: 7200 RPM hard drives; enterprise SSDs; Cloud based storage volumes, including both Remote volumes (over the network), and locally attached for temporary and intermediate data.
- Types of network connections: 1Gb, 4Gb (via bonding), 10Gb Ethernet, 40Gb IP over InfiniBand (IPoIB).
- On-premise vs. Cloud servers: Physical vs. virtual nodes, virtualization overhead, and multi-tenancy performance.

Main initial Hadoop parameters under examination include:

- Ideal number of mappers and reducers per Job type according to the underlying hardware.
- Data compression, both to speed-up execution time and to reduce workload size.
- I/O buffer sizes, block sizes, and data replication factor.

Some of the different deployment patterns include:

- On-premise scale-up physical servers including: multi-cores/sockets, SSDs, InfiniBand, and large RAM memory.
- On-premise commodity hardware including: low core density, limited RAM, SATA array of disks.
- Varied number of Virtual Machines (VMs) in Cloud (Microsoft Azure initially)

- Varied type of VMs available in Microsoft Azure e.g., small A2, medium A7, large A9 VMs.

The resulting performance execution metrics from the ongoing first phase are already accessible at the online application [2]. Hadoop logs, job/task history counters, and abstracted metrics can be accessed from it as described in Section III-B2. In the next phase, the online application will allow requesting executions, and uploading executions to it.

2) Phase 2: During the second phase of project ALOJA, analytical models of Hadoop executions will be introduced. The accumulated performance data in the ALOJA online repository is a corpus of results; modeling then allows price/performance predictions to be made given an input set of workload execution characteristics, hardware components and a solution deployment pattern. These models will allow predictions of likely performance and efficiency outcomes given a set of workload execution characteristics, a set of specified hardware components and a solution deployment pattern by abstracting and characterizing Map/Reduce behavior.

During the second phase, we plan to expand the capabilities of the online application described later in Section III-B2. Part of the extension will include benchmarks from different clusters both for on-premise as well as from different Cloud providers, to continue enriching the online repository. In order to compare benchmark executions from different deployments and environments, a new set of abstracted metrics will be provided during this phase of ALOJA. To reduce the number of executions to run, sampling will be improved by doing an statistical analysis of parameters. Also a new set of fabric configuration and hardware parameters will be evaluated to expand the study of cost-effectiveness:

- Different Hadoop versions e.g, 1.0, 1.3, 2.0, 2.1 versions and vendor distributions.
- Different versions of the Java Virtual Machine (JVM) and its settings.
- Different operating system options, including Windows versions and Linux distributions e.g., general purpose vs. vendor optimized.
- Operating system configurations including JBOD vs. mirroring or striped volumes, buffer caching, paging, and file system types.
- Hadoop executions under the Windows operating system both on-premise and in the cloud as IaaS and PaaS (HDInsight).
- Add different type of benchmarks i.e., SWIM, TPC-H, TPC-DS, or custom Hadoop jobs to the automated execution platform.

In addition to the new set of fabric parameters, we plan to add visualization tools have a detailed and deep understanding during the execution of benchmarks. For instance, manually looking data we found one case where map tasks related to a concrete job started reading about 255 MB, and suddenly it dropped to 100 MB for half of the job's tasks. Therefore, one of the new improvements expected in this phase will be the addition of new visualization tools in the ALOJA Web application. In order to rapidly —and visually— find such variations to filter them out, or examine them in more detail to find the underlying cause of the problem.

Also during this phase, we plan to offer an answer the question of: what is the the best software and hardware configuration for my Hadoop jobs? Having into account not only job characteristics, but also the budget or hardware limitations of the user. To answer this question, we will develop automated mechanisms into ALOJA to determine the cost-effectiveness of each configuration and be able to filter it by the different supported hardware configurations.

3) Phase 3: On the third planned phase of ALOJA, the intention is to build automation around the analytical cost vs. performance models generated in the previous phase so that when provided with a set of workload characteristics, the tools from ALOJA can calculate a prioritized list of hardware, software and deployments pattern options which will be most cost effective for that workload. It will also provide means to predict possible Cloud deployment configurations that will guarantee cost-effective scalability of the deployment. The following section describes the components of the platform, most of which can be used online.

III. The ALOJA Platform

The ALOJA platform is composed of a set of open-source benchmarking and configuration management tools, high-level system performance metric collection, and Web-based data analytics tools. As an example, Figure 2 presents the main screen of the online application. Where the list of over 5000 benchmarks under different hardware and Hadoop configuration parameters have been executed. From the main page interface, the user can filter, select, and compare different Hadoop executions according to their running time, execution cost, and contrast system resource allocations. The following subsections describe the implemented components and partial roadmap. Their source-code and initial documentation can be found in [2].

A. Benchmarking Components

1) Configuration Management: The configuration management scripts are in charge of setting up first the servers, either on-premise, in the Cloud IaaS, or Cloud PaaS. Second, the OS, system metrics gathering tools, and JVM configuration. Third, the Hadoop installation and configuration files according to the *parameter selector* component described in the next sub-section. Followed by the selected benchmark/s (see Subsection III-A3). Then, executing the benchmark. And finally, saving execution logs and performance metrics, cleaning up, and destroying the deployment if required. Optimizations can be applied to reduce the execution time of setting up clusters for situations where more than one test is going to be executed in the same hardware. The same applies for different job types under the same configuration.

2) Parameter Selection and Queuing: The *parameter selector* creates a list of individual tests to run from the defined cluster and capabilities configurations in ALOJA. Each containing the Hadoop job to execute, the Hadoop configuration, and the system settings. Including hardware capabilities of the clusters e.g., to use InfiniBand or Gigabit Ethernet. As well as software options, including JVM version and settings, and Hadoop

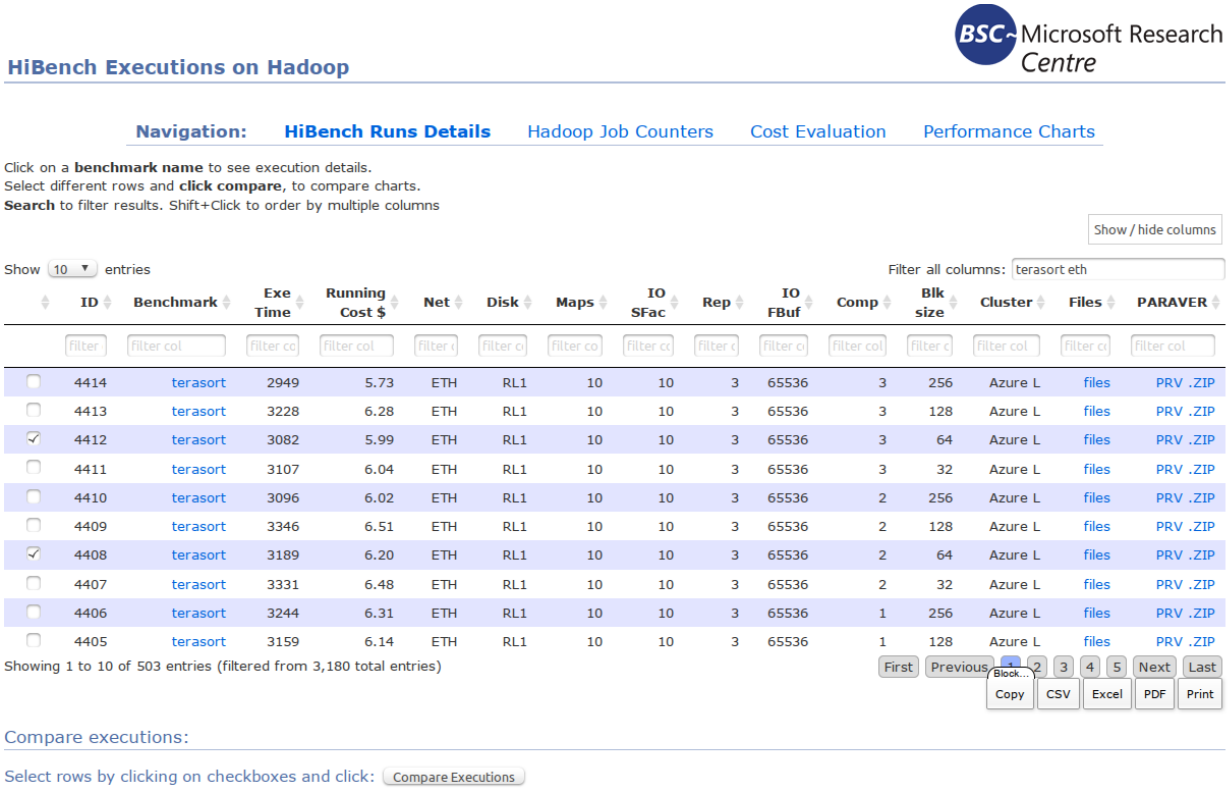

Fig. 2. ALOJA main page: benchmarks execution list by configuration. http://hadoop.bsc.es/

configurations e.g., the number of mappers and reducers to use, or the HDFS block size. For the defined clusters in the configuration of ALOJA, the *parameter selector* iterates through all the possible configuration options. Optionally, sampling can be selected to reduce the number of tests to execute. Sampling will be expanded in the second phase of the project.

ALOJA features queuing services to schedule, prioritize and execute defined jobs. After jobs are created, they are submitted to the corresponding execution queues that control the priorities and execution order. In the second phase of ALOJA we plan to allow users of the application to request executions in the available clusters using their selection of Hadoop configuration options.

3) Benchmarking: Due to the large number of configuration options that have an effect on Hadoop's performance, it is necessary to characterize Hadoop using extensive benchmarking. Hadoop's distribution includes jobs that can be used to benchmark it's performance, usually referred as *micro benchmarks*, however these type of benchmarks usually have limitation on their representativeness and variety. ALOJA currently features the HiBench open-source benchmark from Intel [10], which can be more realistic and comprehensive than the supplied example jobs in Hadoop. HiBench features several ready to use benchmarks from 4 categories: micro benchmarks, Web search, Machine Learning, HDFS benchmarks. The following list briefly describes the benchmarks currently implemented, for a complete description please refer to [10].

- *Terasort*, sorts 1TB of data generated by the *TeraGen* program distributed with Hadoop. *Terasort* is widely used as reference in research papers as well as in Big Data competitions. *Terasort* is I/O and CPU intensive.
- *Wordcount*, counts number of word occurrences in a large text files. It is distributed with Hadoop and used in many Map/Reduce learning books. It is CPU bound.
- *Sort*, uses the Map/Reduce framework to sort the input directory into the output directory, being predominately I/O intensive.
- *Pagerank*, an implementation of Google's Web page ranking algorithm. It crawls Wikipedia sample pages.
- *Bayes*, Bayesian Machine Learning classification using the Mahout library. The input of this benchmark is extracted from a subset of the Wikipedia dump.
- *K-means*, Mahout's implementation of the k-means algorithm for knowledge discovery and data mining.
- DFSIOE or EnhancedDFSIO, an I/O intensive benchmark to measure throughput in HDFS using map reduce. It features separate read and a write benchmarks.

A characterization of the performance details for the benchmarks can be obtained in the *Performance Charts* section of ALOJA's online application [2]. During the second phase of ALOJA, new benchmarks will be added (See Section II-A2).

4) Captured metrics: Benchmark execution captures system and per process performance using a combination of unix system tools i.e., the *sysstat* package, *iostat*, and *bwm-ng*. Featuring: CPU (aggregated and per core), RAM memory state detail usage, system paging, networking, and I/O metrics.

Support for Microsoft performance metrics and tools in will be added in future releases. ALOJA also includes a Hadoop job history log parser, to obtain *Job counters* and also to be able to match Job and tasks execution states to the system performance. All of the collected performance metrics and Hadoop logs and counters can be downloaded from the online application [2].

B. Data Analysis and Flow

After the execution of a benchmark in ALOJA, the resulting data is compressed into a folder to be sent to the file repository. In the master server, the data is uncompressed and performance metrics are extracted from their respective formats, Hadoop history logs and configurations are parsed and the data is imported into a relational database.

1) Database: On import, each benchmark is first assigned a unique identifier, the metrics are time stamped and dumped into a relational database (DB). The DB structure correlates and matches data coming from different source to be able to relate e.g., the moment a particular running task was running in a server host, to the IO wait time of the CPU, or the effect of *shuffling* in Hadoop to the network. At the time of writing the DB engine in use is MySQL, but as the data grows there are plans to move it to a Hadoop based query engine for scalability. The DB enables simple SQL querying to the execution sources, and it is leveraged in the online data analysis application described next.

2) Online Analytics and Sharing Tool: ALOJA features an online Web application (*ALOJA-WEB*) publicly accessible [2] and serves several purposes. First, it is used as the main platform for diffusion of the data generated by the project, containing an indexed, searchable, repository of Hadoop benchmarks executions. The main screen can be seen in Figure 2.

Second, it is the main means of sharing data and findings with other researchers involved in the project. The source files can be downloaded directly for the offline analysis with other tools. For example, *ALOJA-WEB* also features export functionality to Paraver trace format, part of the low-level performance instrumentation employed in BSC for High Performance Computing (HPC), detailed in Section III-C0a.

Third, *ALOJA-WEB* allows browsing performance metrics of different executions and a comparison between different runs. Including both system performance metrics and Hadoop execution details for the different hosts, aggregations, and visualizations. This feature allows a rapid comparison and sharing of findings just by sharing the resulting URLs after using any of the filters.

Fourth, *ALOJA-WEB* features a *Configuration Speedup* evaluation section, where the user in a single view can filter the over 5000 execution variations to quickly obtain insights from the aggregated metrics (see Sections IV-A, , and IV-B1 for examples). Also, the *Cost Evaluation* section presents a way to quickly gain an insight on the cost-effectiveness for each of the different hardware deployments and software configurations. Examples of its outputs are described in Section IV-C.

C. Deep instrumentation

ALOJA Online provides a simple mechanism for direct comparison of results from different Hadoop runs, however,

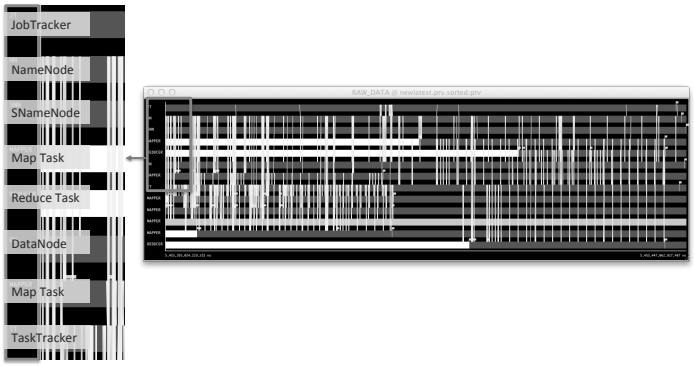

Fig. 3. Packet level communication trace between Hadoop Daemons and Tasks. X-axis represents time. Y-axis are the captured daemons. Vertical lines represent communications between pairs of daemons and/or tasks.

in some cases it will be necessary to conduct more detailed forensics on the runtime performance. To do this, different software tools developed by BSC will be used to create more fine-grained Hadoop performance traces that will be afterwards analyzed to extract knowledge about the root causes of performance degradation.

a) Paraver: Trace analysis and visualization: Paraver [4] is a flexible performance visualization and analysis tool based on an easy-to-use Motif GUI. Paraver was developed to respond to the need to have a qualitative global perception of the application behavior by visual inspection and then to be able to focus on the detailed quantitative analysis of the problems. Paraver provides a large amount of information useful to improve the decisions on whether and where to invest programming effort to optimize an application. While the tool has been widely used in the past for the performance analysis of HPC applications, it has also been leveraged for Web Application analysis and now for Hadoop workloads. All data logs available through the Web application are already downloadable in Paraver trace format and can be analyzed and visualized in detail using its analysis tools.

b) Hadoop Analysis Toolkit: Hadoop deep analysis will leverage different previously developed [5] components. Java instrumentation Suite (JIS), a tracing environment oriented to the study of Java applications is being extended to collect information about Hadoop execution in detail, as shown in Figure 3. Four levels are considered by JIS when tracing a system: operating system, JVM, middleware (HDFS and MapReduce) and application. Information collected by all levels is finally correlated and merged to produce an execution trace file. The level of detail of the information produced by each JIS level can be dynamically configured. To enable JIS hooks in the code of Hadoop runtimes, a dynamic code interposition tool is under development, leveraging Aspect Oriented Programming concepts to avoid the need of code recompilation.

c) Advanced Analysis tools: In previous work [6], it was proven the suitability of DBSCAN [12] based on performance counters to characterize the internal structure of message-passing applications. The clustering technique is able to correctly detect different algorithm phases as well as regions of different subroutines with similar behavior. ALOJA plans to take such work and migrate it to the Hadoop ecosystem,

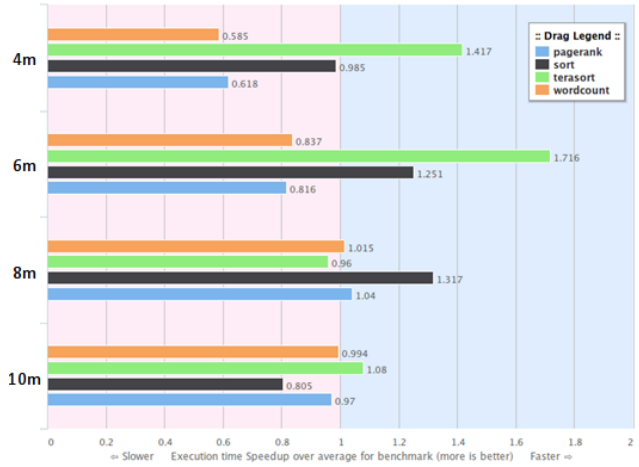

Fig. 4. Speedup of different number of maximum maps for different benchmarks

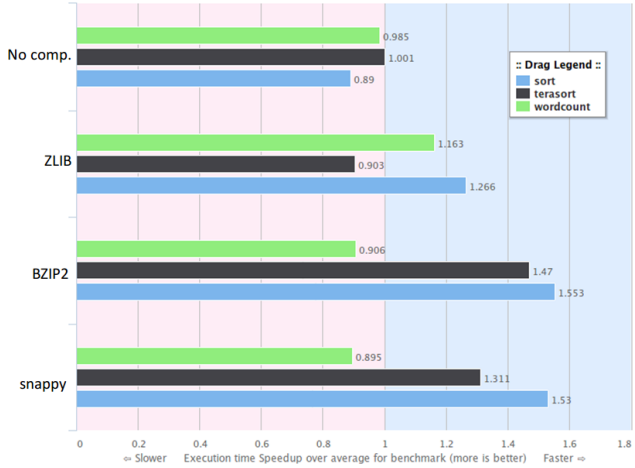

Fig. 5. Speedup of different compression options for different benchmarks

to allow for automatic analysis of performance traces. The analysis tools will be hooked to the portal backend so that deep data processing can be integrated into the search capabilities of the web application for better understanding of the collected performance data.

D. Initial Testing Infrastructure

The ALOJA project aims to create a vendor-neutral workbench, and for this reason the execution environment of the tests is not a closed specification. The ALOJA platform is designed to support multiple execution environments and configuration. However, the initial development and testing conducted with the platform, used the following hardware configuration.

1) High-End Cluster: The on-premise cluster is composed of machines enabled with two 6-core sandy-bridge Intel processors and 64GB of RAM. The I/O subsystem is composed of 6 SATA2 SSD drives configured as a RAID-0 volume. The measured performance of the SSD array is 1.6GB/s for read operations, and almost 1GB/s for write operations. The nodes are also equipped with a boot drive and a 3TB SATA HDD. The network interfaces configured for each node include four GbE ports, configured either as single ports or as Link Aggregation group through the use of LACP. Each node is also enabled with two FDR InfiniBand ports, each of them providing a peak bandwidth of 56Gpbs. The nodes are interconnected through FDR InfiniBand switching fabric as well as non-blocking GbE fabric.

2) Cloud IaaS: The IaaS tests were conducted on Microsoft Azure A7 instances, each of them containing 8 cores and 56 GB or RAM. Each Azure instance can mount up to 16 remotes volumes, each of them limited to a maximum of 500 IOPS. For the initial experiments, different configurations of remote volumes and local storage were explored. In some cases intermediate data were stored on the local disks while input and output data were stored in remote volumes. In other cases all data, including intermediate results, were stored in remote volumes.

IV. Early Findings

This section presents the impact of different Hadoop configuration parameters, as well as hardware configurations to both performance and costs. The following experiments can be obtained in ALOJA's online application [2].

Figures 4, 6, and 7 present the *speedup* of different software and hardware parameters from the benchmark repositories. The *speedup* is defined as the he relative performance improvement, in this case, to the average execution time for the selected benchmark. Where the average execution time is at 1 on the X axis. Any value below 1 represents a *speed-down* and a value above 1 represents a *speed-up* over the average. The figures are presented next.

A. Impact of Software Configurations

Figure 4 shows the speedup according to the maximum number of maps configured in Hadoop for *pagerank*, *sort*, *terasort*, and *wordcount* benchmarks in the Azure cluster. While the A7 virtual machines used in Microsoft Azure have 8 CPU cores available, and the maximum number of mappers are directly related to the available cores, it can be seen that for the different benchmarks, there is differentiated best performing setting. While *pagerank*, *sort*, and *wordcount* achieve best performance with 8 mappers (1 mapper per core), *terasort* achieves a 1.7x speedup when set to 6 mappers, mainly due to increased CPU I/O wait time caused by the concurrency while reading data. It can also be seen that a setting of 4 and 10 mappers yield sub-optimal performance. The default Hadoop configuration sets only 2 mappers.

Figure 5 shows the compression *speed-up* by selecting different compression schemes for *sort*, *terasort*, and *wordcount*. The first group of bars shows the effect of having no compression. It can be seen that for the *sort* benchmark using no compression *speeds-down* the execution, while the rest benchmarks are not affected in comparison to their averages. The ZLIB algorithm *speeds up* both *sort* and *wordcount*, but not *terasort*. BZIP2, a high compression algorithm, but at the expense of more CPU time, *speeds up* both *sort* and *terasort* but not *wordcount*. BZIP2 results in Figure 5 in this example

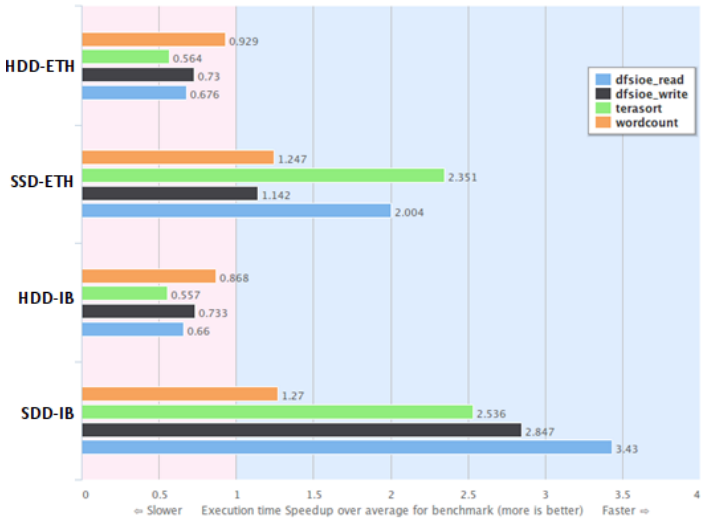

Fig. 6. Speedup of using SSDs and InfiniBand for different benchmarks

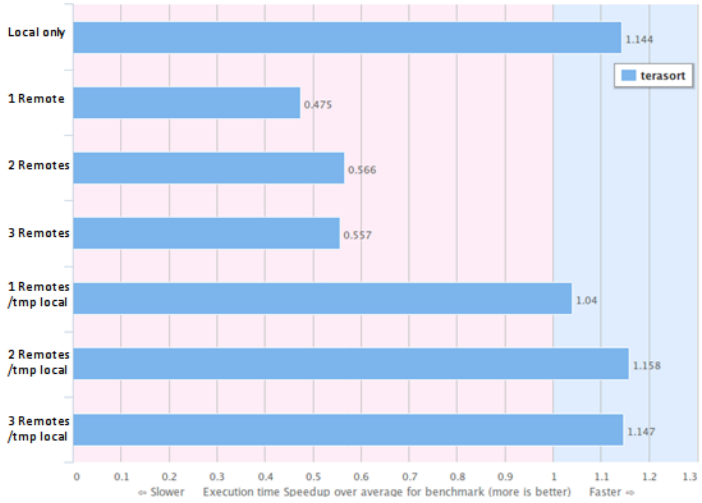

Fig. 7. Speedup of different Cloud deployment options for disks

are very similar to the *snappy* compression algorithm that features fast compression but a low compression ratio. For this experiment, the recommendation for *sort* and *terasort* will be to use BZIP2 compression, as it features better compression ratio, making data smaller, while achieving comparable or faster *speedups* to *snappy*.

B. Impact of Hardware Configurations

The effect of different hardware configuration options can also be studied directly in ALOJA-WEB. Figure 6 presents the speedup of using SSDs and InfiniBand networks over rotational disks (SATA) and Gigabit Ethernet network from our on-premise cluster (see Section III-D). Results are presented for 4 benchmarks: *dfsioe_read*, *dfsioe_write*, *terasort*, and *wordcount*. The first group of bars *HDD-ETH* represent the baseline comparison. The second group *SSD-ETH* represent the improvement of using SSDs. It can be seen that both *terasort* and *dfsioe_read* improve their performance by at least 2x. While *wordcount* and *dfsioe_write* have a minimal speedup. The next group of bars *HDD-IB* present the improvement of InfiniBand alone. It can be seen that the improvement is minimal in comparison with the baseline. The last group of bars *SDD-IB* presents the added improvements of both SSDs and InfiniBand networks. It can be seen that the *dfsioe* benchmarks improve their performance up to 3x using InfiniBand networks. While for *terasort* and *wordcount* the improvement is negligible. *Wordcount* has a minimal improvement by using SSDs, since *wordcount* is CPU bound rather that I/O intensive. These early findings show that in order to benefit from InfiniBand networks on the default Hadoop distribution, it requires a faster I/O subsystem i.e., SSD drives. By using SSDs only, some benchmarks can *speed up* their execution up to 2x, while if combined with InfiniBand networks up to 3.5x.

1) Remote volumes in the Cloud: In cloud environments, large price-competitive storage for Big Data usually comes in the form of remote volumes —network attached storage—, as VMs might be migrated to different hardware and contents on local disk lost. Figure 7 compares the speedup obtained by using different disk configurations in Azure. The fist bar, *Local* refers to the baseline measurement by having Hadoop configured to use only local disk, both for the HDFS data and *temp* directory. This configuration is the baseline, as would rarely represent a real deployment, as data on the local disk would be lost if migrating VMs. Local disks are also faster than remotes, as we found that on A7 VMs the underlying physical host is using SSD disks, at the expense of less storage space —600GB for Local. The next 3 bars represent the *speed-down* when setting up Hadoop to use remote storage to 1, 2, and 3 remote volumes respectively. The last 3 bars represent a configuration of 1, 2, and 3 remote volumes for HDFS, while Hadoop's *temp* folder is configured to use the local drive. It can be seen that while using only 1 remote the speedup is just below the baseline comparison of just local disk, it performs almost 2x compared to setting the *temp* dir to the remote. The next bar —2 remotes, temp local—, achieves the best performance, and improves the baseline comparison. The last bar shows the effect of adding a third remote, and that it does speedup the execution; however, it adds extra capacity to HDFS at the expense of some cost. The next section presents a cost-performance comparison that also reports results for distinct remote volumes configuration.

C. Performance vs. Cost Analysis

This subsection presents a performance vs. cost analysis feature of the ALOJA online application [2]. It is used to evaluate the impact in both running costs and total execution time under different hardware and software configurations. Figures 8 and 9 presents the normalized (standardized score) cost vs. performance evaluation of *terasort* vs. *wordcount* benchmarks respectively. Results for other benchmarks can be obtained online. Both figures are divided into 4 quadrants to represent the execution cost vs. performance result; point (0,0) represents the best cost-effective execution. The quadrants being: Fast-Economical, Fast-Expensive, Slow-Expensive, and Slow-Economical.

Each point in Figures 8 and 9 represents a different benchmark execution with a different hardware and software configuration. It can be seen from both figures, that results tend to cluster naturally according to their hardware characteristics.

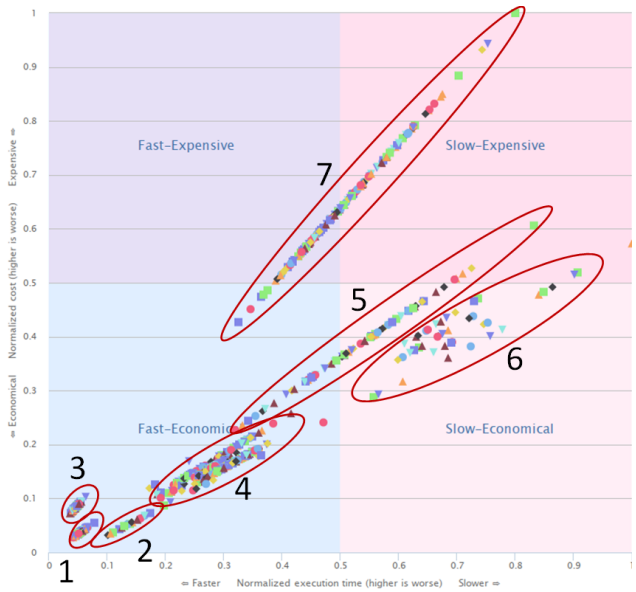

Fig. 8. Cost vs. Performance for Terasort for different SW and HW

Fig. 9. Cost vs. Performance for Wordcount for different SW and HW

In total we have 7 different hardware configurations. The 4 for our on-premise cluster of 2 distinct disks and network configurations, can also be seen in Figure 6 from the previous sub-section. As well as 3 for the Azure IaaS cluster, by using different types of disk, representing up to 4 distinct disk configurations. Each cluster is grouped for identification using an oval and a number matching hardware configurations listed below:

1) On-premise cluster: SSD disks + Gigabit Ethernet network.
2) Azure IaaS: Using only the local disk, virtualized SSD and Gigabit network Used ad baseline comparison, as it will rarely be a used on a real deployment.
3) On-premise cluster: SSD disks + InfiniBand over IP network.
4) Azure IaaS: 1-3 remote volumes (Blob storage) and Hadoop *temp* dir to local disk virtualized SSD and Gigabit network.
5) On-premise cluster:1 SATA disk + Gigabit Ethernet.
6) Azure IaaS: 1-3 remote volumes (Blob storage).
7) On-premise cluster:1 SATA disk + InfiniBand over IP network.

The costs for each hardware configuration can be found under the *Cost Performance* evaluation section of ALOJA's online application [2]. Where prices can also be edited and the chart recalculated to evalute different pricing.

From each cluster in Figures 8 and 9, it can be noted that the different software configurations also vary the cost vs. performance result, as they influence in the total running time. There are several insights that can be obtained by evaluating both figures. It can be seen that the clusters for each benchmark are in different positions of each quadrant. Recall that the charts are normalized to the standard score between 0 and 1 so that the results can be comparable. This means that for different benchmarks, hardware and software configurations have a different impact in their total execution time, and finally in their running costs. The best cost vs. performance in both figures are for cluster 1. The distance between the rest of the clusters and their positions express their difference in cost vs. performance. In both figures cluster 1 and 3 are in the same X position, this means that they had a similar execution time. However, since cluster 3 uses an InfiniBand network which is more expensive and in both cases does not improves performance, it is placed higher in the Y axis (costs).

Both the on-premise and the Microsoft Azure cluster have similar underlying hardware in order to compare between the two. However, there is a difference in number of cores —12 to 8 respectively—, and since *wordcount* is CPU intensive, this difference in number of cores yields to better results in the on-premise cluster. For *terasort*, which is more I/O intensive Azure configuration score better. While cluster 2, that uses only the local disks and does not represent a realistic deployment, we can see that cluster 4 is very close behind for *terasort*. For *wordcount*, we can see that both clusters 2 (baseline) and 4 achieves same performance, but cluster 4 executions are marginally more expensive according to the number of external drives employed.

Another insight from Figures 8 and 9 is the cost vs. performance improvement of tuning Hadoop software parameters. For each of the clusters, the lowest point to the left was the best configuration for that cluster. Since the faster a benchmark executes, its running costs are reduced proportionally to execution time. A remarkable feature that is shown in both clusters 5 and 7 —that features SATA drives— on both figures, is that software parameters can reduce the execution time by half to the worst case scenario. This means that for SATA drives, the evaluated software parameters have higher impact in performance that on SSDs, seen on clusters 1 and 3.

This section has presented the current value of the ALOJA project and online application for evaluating the cost-effectiveness of different hardware and software configurations for Hadoop.

V. Conclusions

This article presented the ALOJA project, a joint research effort with the goal of providing automated optimization for the performance of Hadoop infrastructure deployments. With an undergoing initial focus to carry out a systematic study of Hadoop execution performance across a broad range of different hardware and software configurations. Deployed to both on-premise physical servers as well as cloud based infrastructures including both IaaS and PaaS. The performance results from these tests, as well as the testing methodology and test infrastructure is made available to the public through an online Web application [2] featuring more than 5000 runs already.

The early findings of the project show significant value in understanding Hadoop's runtime and the cost-effectiveness of different configuration and deployment options. With the available online performance evaluations, ALOJA can currently derive the expected *speed up* of hardware configurations i.e., SSD disks and InfiniBand networks. As an example, we have shown that for InfiniBand networks to be cost-effective, they need to be combined with SSDs or other fast disk options, if not the improvement they provide is negligible. The scalability and cost vs. performance evaluation of remote volumes in the cloud was also evaluated and found that 2 remote volumes with the temporary data on local disk is the most cost-effective configuration. Also, best Hadoop configuration options such as the number of mappers to run in parallel according to the available CPU cores and job types, or the best cost effective compression factor to use according to the different workloads was presented. ALOJA already shows value to the Hadoop community by producing more knowledge and understanding of the underlying Hadoop runtime while it is executing. Our intent is that researchers and organizations evaluating or deploying the Hadoop solution stack will benefit from this growing database of performance results and configuration guidance.

Acknowledgements

This work is partially supported by the Ministry of Science and Technology of Spain under contracts TIN2012-34557 and 2014SGR1051.

References

[1] R. Appuswamy, C. Gkantsidis, D. Narayanan, O. Hodson, and A. Rowstron. Scale-up vs scale-out for hadoop: Time to rethink? In *Proceedings of the 4th Annual Symposium on Cloud Computing*, SOCC '13, pages 20:1–20:13, 2013.

[2] BSC. Aloja home page: http://aloja.bsc.es/, 2014.

[3] BSC. Autonomic systems and big data research group page: http://www.bsc.es/computer-sciences/autonomic-systems-and-e-business-platforms, 2014.

[4] BSC. Performance tools research group page: http://www.bsc.es/computer-sciences/performance-tools, 2014.

[5] D. Carrera, D. Garcia, J. Torres, E. Ayguade, and J. Labarta. Was control center: an autonomic performance-triggered tracing environment for websphere. In *Parallel, Distributed and Network-Based Processing, 2005. PDP 2005. 13th Euromicro Conference on*, pages 26–32, Feb 2005.

[6] J. Gonzalez, J. Gimenez, and J. Labarta. Automatic detection of parallel applications computation phases. In *Parallel Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pages 1–11, May 2009.

[7] D. Heger. *Hadoop Performance Tuning https://hadoop-toolkit.googlecode.com/files/White paper-HadoopPerformanceTuning.pdf*. Impetus, 2009.

[8] D. Heger. *Hadoop Performance Tuning - A Pragmatic & Iterative Approach*. DH Technologies, 2013.

[9] G. L. N. B. L. D. F. B. C. S. B. Herodotos Herodotou, Harold Lim. Starfish: A self-tuning system for big data analytics. In *In CIDR*, pages 261–272, 2011.

[10] S. Huang, J. Huang, J. Dai, T. Xie, and B. Huang. The HiBench benchmark suite: Characterization of the MapReduce-based data analysis. *Data Engineering Workshops, 22nd International Conference on*, 0:41–51, 2010.

[11] L. Person. *Global Hadoop Market*. Allied Market Research, March, 2014.

[12] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu. Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data Min. Knowl. Discov.*, 2(2):169–194, June 1998.