The state of SQL-on-Hadoop in the Cloud

Nicolas Poggi, Josep Ll. Berral, Thomas Fenech, and David Carrera,
Barcelona Supercomputing Center (BSC)
Universitat Politècnica de Catlalunya (BarcelonaTech)
Barcelona, Spain

José Blakeley, Umar Farooq Minhas, and Nikola Vujic[1]
Microsoft Corporation, Microsoft Research (MSR)
Redmond, USA
Microsoft Development Center Serbia (MDCS)[1]
Belgrade, Serbia

***Abstract*—Managed Hadoop in the cloud, especially SQL-on-Hadoop, has been gaining attention recently. On Platform-as-a-Service (PaaS), analytical services like *Hive* and *Spark* come pre-configured for general-purpose and ready to use. Thus, giving companies a quick entry and on-demand deployment of ready SQL-like solutions for their big data needs. This study evaluates cloud services from an end-user perspective, comparing providers including: Microsoft Azure, Amazon Web Services, Google Cloud, and Rackspace. The study focuses on performance, readiness, scalability, and cost-effectiveness of the different solutions at entry/test level clusters sizes. Results are based on over 15,000 Hive queries derived from the industry standard TPC-H benchmark. The study is framed within the ALOJA research project, which features an open source benchmarking and analysis platform that has been recently extended to support SQL-on-Hadoop engines. The ALOJA Project aims to lower the total cost of ownership (TCO) of big data deployments and study their performance characteristics for optimization.**

The study benchmarks cloud providers across a diverse range instance types, and uses input data scales from 1GB to 1TB, in order to survey the popular entry-level PaaS SQL-on-Hadoop solutions, thereby establishing a common results-base upon which subsequent research can be carried out by the project. Initial results already show the main performance trends to both hardware and software configuration, pricing, similarities and architectural differences of the evaluated PaaS solutions. Whereas some providers focus on decoupling storage and computing resources while offering network-based *elastic* storage, others choose to keep the local processing model from Hadoop for high performance, but reducing flexibility. Results also show the importance of application-level tuning and how keeping up-to-date hardware and software stacks can influence performance even more than replicating the on-premises model in the cloud.

I. Introduction

Enterprise migration of big data services remains among the last barriers for complete migration to the cloud [17]. The migration and operation process can represent a significant economic investment, mainly due to the cost of data movement and storage, though there are also concerns regarding data security and governance. Cloud providers are required to offer a service that meets these needs at a price and performance that is competitive with traditional offerings. Historically, on-premises local deployments have been favored, in part due to Hadoop systems being designed for commodity local hardware. The classical Hadoop-model takes advantage of application-based data replication and locality, and exclusive access to the physical resources. However, in recent years, new managed enterprise big data services have emerged in most cloud providers [17], facilitating software-defined on-demand big data deployments. These services create compelling technical reasons for migration, such as elasticity of both compute and storage, while maintaining a simplified infrastructure management i.e. via *virtualization*. Furthermore, with such services often using a *Pay-as-you-Go* or even *Pay-as-you-Process* pricing model, they are becoming increasingly economically attractive to customers.

This study focuses on Platform-as-a-Service (PaaS) deployments with managed Hadoop, in which popular services like *Hive* and *Spark* usually come pre-configured, saving enterprise time and effort when looking for ready-to-use SQL-like solutions. Cloud providers make the complex configuration and tuning process [20] transparent to their clients, while providing features such as data security and governance. On top of this, by having multiple customers, service providers can potentially improve their software-stack from user feedback, as upgrading services more often than smaller companies. As a result, the client can benefit from the immediate availability of a tested and generically-optimized platform with upgrade management. It is the intent of this work to compare out-of-the-box performance of the most popular available solutions and give an insight into the life-cycle of the services.

The recent study conducted by Forrester Market Research [17] compared performance across a range of cloud providers, highlighting the leaders in the field. The services offered by four of those providers are examined in this report, namely those of Microsoft Azure, Amazon Web Services (AWS), Google Cloud Platform, and Rackspace. Different Virtual Machine (VM) instance types are tested for each of the providers, with a focus on the provider selected defaults, and a fixed cluster size to 8 data nodes for the main results. A complementary extension is also included, showing the scalability from 4 to 32 datanodes on the Azure platform. In order to provide results reflective of an out-of-the-box experience, no fine-tuning of the execution environment is carried out in any of the PaaS instances. In order to compare the impact of fine-tuning the software stack, a local commodity Hadoop cluster is also benchmarked. Tests were carried out across a range of data scales from 1GB to 1TB (where supported) using a TCP-H derived benchmark for Hive. Hive was chosen over SparkSQL, firstly as it is the most mature and stable alternative of the two, as well as because all the selected providers have Hive better configured and offer similar versions, providing a better baseline for comparison.

Objectives and contributions:

1) Survey the popular entry level PaaS SQL-on-Hadoop solutions from main cloud providers and compare their default offerings.
2) Study the hardware architectural differences and explore elasticity and performance of the choices for analytical services using Hive.

3) Characterize the cost-effectiveness of the solutions and how the software configuration affects performance.
4) Give insights on scalability in data and compute, and the life-cycle of the services.

The initial results show main performance trends, limits and differences of the evaluated PaaS solutions. For example, while some providers prefer to decouple with network-based *elastic* storage, separated from the computing resources, others prefer to keep local processing models, maintaining Hadoop classical design. However, classical Hadoop configuration does not necessarily imply fast execution of analytical queries (i.e., Hive), where keeping up-to-date versions and fine tuning can influence the performance more than local compute. The life-cycle analysis also shows how services are updated constantly, both in hardware and software stacks, improving the performance of services and lowering costs. The provided flexibility and simplified management by current PaaS offerings can be very attractive features for enterprises when migrating or starting big data services in the cloud. As results show, users can expect comparable performance in mid-sized clusters compared to traditional setups, as well to frequently updated services. As a note, prices and even performance numbers in this report represent only a snapshot of the fast-paced cloud environment, in which cloud providers are constantly surpassing one another in terms of performance and pricing.

Organization: The rest of the article is organized as follows: Section II presents the tested systems, their prices, and main characteristics. Section III introduces state of the art on which this study is framed. Section IV explains the methodology followed in the experiments, as well as presents a summary of execution times for the different data scales, the costs per run, and the cluster size scalability tests. Section V presents a detailed analysis on how the selected hardware and software configuration impact query running times, providing examples on the update life-cycle of two providers, and compares the running time variability between different instances. Finally, Section VI provides a discussion of the lessons learned, presents the conclusions, and future work.

II. Providers and Systems-Under-Test (SUTs)

This section presents and compares the different providers and Systems-Under-Test as well as the selection criteria for the different options and software versions. The Hive-based SQL-on-Hadoop PaaS services from 4 major cloud providers are compared:

- HDInsight (HDI) from Microsoft Azure.
- Cloud Big Data (CBD) from Rackspace.
- Elastic Map Reduce (EMR) from Amazon Web services (AWS).
- Cloud Dataproc (CDP) from Google Cloud Platform.

There are different reasons justifying the selection of each provider. HDI and CBD have both been studied previously [19, 20], and as such, their APIs are already well-integrated into the ALOJA platform (see Section III-1). EMR was the first major Hadoop PaaS solution, and currently has one of the largest usage shares [17]. Both AWS and EMR are commonly used as bases of comparison in the literature [24, 26, 27]. CDP from Google Cloud has been included due to it being identified as a leading provider [17], as well as for being a new service (GA in 2016) which could potentially have a different architecture. Additionally, results are presented from a classical *commodity* on-premises local cluster, with a general vs. a fine-tuned configuration, to contrast the performance of the cloud solutions to both a general installation and an expert-managed solution.

A. Virtual Machine instances

When creating a new Hadoop cluster, one of the main decisions that has an impact on performance and pricing is the choice of the type of VM instance and the number of data nodes. In this study, default VMs from each provider are compared with the higher-performance alternatives, to evaluate the price/performance benefit. Table I summarizes the specifications of the different VMs that were tested and the prices for the 8 data nodes clusters and 5TB of HDFS capacity required for the main experiments. Default VMs are highlighted in bold.

1) HDInsight: for HDI, benchmarks were run on three generations of VM: D3v2 (current default), D3v1 (previous default), and Large (also referenced as A3 and default during 2015)—all with 4-cores, and D4v2 featuring 2x the specs of D3v2 including 8-cores. The HDFS on all HDI instances is backed by the Azure Blob store (through the WASB driver). This means that it is an object-store over the network. As a consequence, the storage on HDI is decoupled from compute and can grow elastically, as well as be used from outside the HDFS cluster on other shared services and users. Local disks, backed by SSD drives on the D-series, are *ephemeral* and used for temporary or intermediate data only. Deployment times in Azure took close to 25 minutes on most builds.

2) Cloud Big Data: in CBD on the other hand, the proposed default option is based on the OnMetal40 instance. As the name implies, the OnMetal40 (from now on referred as OnMetal) instance is dedicated hardware that is managed as a VM through their API/Web, and offers a high-end setup: 40-cores, 128GB of RAM, and 2 local SSDs. For this reason and to match a closer pricing and specs to the rest of the SUTs, we have set the OnMetal-based clusters to 4 data nodes only, while all the rest have 8, as highlighted in Table I. The OnMetal cluster is the only SUT besides our on-premises cluster that is not potentially *shared* with other cloud users. CBD also offers multi-tenant VMs like the rest of the providers; we have tested VMs: hadoop1-7, hadoop1-15 (the shared default), hadoop1-30, where the last digits represent the amount of RAM per node. In the case of CBD, HDFS is mounted on the provided local disks, offering a similar setup to on-premises deployments. CBD also offers the option of adding external stores to HDFS i.e., the Cloud Block Store (CBS) and AWS S3 compatible, but this was not tested in this work, partially due to having enough local disk space for our tests, which resulted in a saving in storage costs. Deployment times for CBD were similar to Azure, at around 25 minutes.

3) Elastic Map Reduce: for EMR, the default m3.xlarge instance was tested, which comes with 80GB local SSD disk per node (an EBS option is also available); and so was the m4.xlarge, an EBS-only storage subsystem instance. The systems, both with 4 cores, have 15GB and 16GB RAM respectively, but the m4-series contains a newer hardware generation. EBS stands for Elastic Block Store, Amazon's over-the-network storage. EBS has 4 different throughput (IOPS) plans, according to the technology backing the storage. The plans being being

TABLE I. COMPARISON OF THE DIFFERENT SUTS SPECS AND PRICES FOR 8 DATANODES AND 5TB STORAGE

Provider	Instance type	Default?	Cores/ Node	RAM/ Node	Storage/Node	Cost/hour cluster 8 data nodes	Cost/5TB/hr	Deploy time
Amazon EMR	**m3.xlarge**	Yes	4	15	2x40GB Local SSD (EBS option)	$ 3.36	$ 0.07	∼10 mins
	m4.xlarge	No	4	16	EBS size defined on deploy	$ 2.99		
Azure HDI	A3 (Large)	(old def.)	4	7	Elastic (WASB) + 285GB SATA	$ 2.70	$ 0.17	∼25 mins
	D3 v1 and v2	Yes	4	14	Elastic (WASB) + 200GB SSD	$ 5.25		
	D4 v1 and v2	No	4	14	Elastic (WASB) + 400GB SSD	$ 10.48		
Google CDP	**n1-standard-4**	Yes	4	15	GCS size defined on deploy	$ 1.81	$ 0.18	∼01 min
	n1-standard-4 1SSD	No	4	15	1x375GB SSD + GCS	$ 1.92		
	n1-standard-8	No	8	30	GCS size defined on deploy	$ 3.61		
Rackspace CBD	hadoop1-7	No	2	7	1xSATA 1.5TB (CBS option)	$ 2.72	N/A (local) CBS $ 0.07	∼25 mins
	hadoop1-15	(2nd def.)	4	15	2xSATA 2.5TB (CBS option)	$ 5.44		
	hadoop1-30	No	8	30	3xSATA 5TB (CBS option)	$ 10.88		
	OnMetal40	Yes	40	128	2xSSD 3TB (CBS option)	(4-nodes) $ 11.80		
On-premises	Intel Xeon E5-2630L	N/A	12	64	6x1TB SATA + 300GB SSD	$ 3.50	N/A (local)	N/A

high-performance or regular, for both SSDs and rotational drives; we chose the regular SSDs. Since the m3.xlarge comes with only 80GB of local store per node, we also used EBS disks to reach the 5TB capacity needed. Deployment times were faster than HDI and CBD at around 10 minutes.

4) Cloud Dataproc: for CDP, we have evaluated the n1-standard-4 default instance with 4-cores and 15GB RAM, both with Google Cloud Storage (GCS)—the network based storage—and with one additional local SSD reported as a second SUT. In CDP, up to 4 SSDs can be added per node at creation time and the volumes are not *ephemeral* as in HDI, but part of the HDFS tiered storage. The third CDP SUT is a n1-standard-8 instance, with 2x the specs of the n1-standard-4, that is 8-cores and 30GB of RAM. Deployment times were surprisingly fast for CDP, with cluster build times at around 1 minute.

It is interesting to note, that the default cloud-based VMs for each provider all have 4 virtual CPU cores each and about 16GB of RAM (highlighted in bold in Table I) providing a comparable common ground among providers—with the OnMetal exception that was adjusted to a smaller cluster to be comparable. Default settings provide roughly 4GB of RAM per CPU core on most SUTs. Software configuration is later analyzed in Section V-C. All four providers also include at least two master and auxiliary servers for High Availability (HA) and cluster management. These extra nodes, which can be used for monitoring, can be customized to different specs than the datanodes, defaulting to lower-end VMs that are included in the price. This is a difference with our on-premises cluster, where we benchmark a test-only solution without the HA and production tools available in providers. The on-premises cluster is from 2012 and composed of Intel Xeon CPU E5-2630L with two CPU sockets with 12-cores total, 64GB of RAM, 6x1TB SATA drives as JBOD, and 1x300GB SDD for temporary storage. For these tests a setup with 8 datanodes and Ubuntu Linux 14.04 is used.

B. Pricing and Elasticity

Contrasting the similarity of CPU and RAM specs for default instances, Table I also shows the diverse pricing of the complete clusters. Prices range from USD 1.81 to 11.80 per hour, with CDP being the least expensive and CBD OnMetal40 the most expensive respectively. Another important factor that affects final pricing is that not all instance types include the I/O used. Only the instances that use the local disks (as opposed to networked storage) have no extra charges in both data storage and operations (IOPS) consumed during usage and make use of the default Hadoop Distributed File System (HDFS) storage. However, networked/remote storage has the advantage of elastic scaling of data. This means that the user is not tied to the maximum amount of local disk storage, but that storage can increase as the user needs, paying only for what is used i.e., as in HDI. Furthermore, data replication, a common feature of HDFS, is handled by the storage driver for each provider, leaving each provider with the task of optimizing the networked storage for their architectures, which can lead to differentiated Quality-of-Service (QoS). The price-to-performance ratio of each approach is later analyzed in Section IV-B.

All of the providers offer a way to elastically scale the number of processing—compute—nodes through their APIs, as opposed to the on-premises cluster which is fixed. However, providers differ on the storage approach at cluster creation time. On CBD it is optional and local processing is preferred, on EMR it depends on the instance model e.g., optional for the m3.xlarge but not for the m4.large we have tested; while on HDI and CDP it is mandatory for all clusters. While both Google's GCS and Amazon's EBS are network-based stores and with user-defined sizes, in both, the user needs to specify the size when creating the cluster. This means that each subsequent datanode added in the future will be fixed to this initial size and the user pays for this capacity even if it is not used. On the other hand, on HDI the user does not have to make a decision on the cluster size and pays for only what is used, benefiting from the system being more elastic and completely disaggregated. Having disaggregated storage also means that a cluster can grow in compute independently. In this case, the user can choose to add more computing capacity depending on the type of jobs that will be run to improve execution speed, but to operate only a minimal cluster—or even shut it down—when it is not being used, as the storage is maintained (and charged) independently. In HDI, this is the only option available, as local disks are considered *ephemeral* and not permanent storage. This is to contrast with on-premises clusters, where sizes are fixed and upgrades years later result in newer, but heterogeneous hardware. Another advantage of disaggregated storage, is that different teams from the same company can deploy different clusters, and link them to the same data stores or access part of the data independently of each other.

C. Software stack and versions

While EMR, dating back to 2009, was the first major PaaS Hadoop solution, the other main providers have caught up in packaging Hadoop with other popular ecosystem services. Currently, all of the four providers tested ships with both *Hive* and *Spark* for SQL-like analysis on top of Hadoop, as well

as other services and tools i.e., HBase, Pig, etc... There are differences in security and governance features, but these are not compared in this work. In relation to the software versions, both Azure and Rackspace base their PaaS solutions on the Hortonworks Data Platform (HDP) [10], a popular Hadoop distribution that users might already be familiar with. Both HDI and CBD services were released in 2013. HDI supports both Ubuntu Linux 14.04 and Windows Server, while CBD uses CentOS 7 as its operating system. During the main experimental period—March to July 2016—HDI added support for HDP version 2.4, while continuing to support 2.3, but both shipping with Hive 1.2 and Hadoop 2.71. The HDI results presented in Section IV are only for 2.4, while Section V-D compares both versions on 4 data nodes clusters as a separate experiment. CDB still used HDP version 2.3.4. AWS uses a custom-built stack for EMR, as well as a custom Linux version called the Amazon Linux AMI. The EMR tests were run with the latest version available at the time of testing, EMR 4.7. EMR instances came with Hive 1.0.0-amzn-5 and Hadoop 2.7.2-amzn-2. During August 2016, EMR upgraded to version 5.0, and so we also present a brief comparison between versions in Section V-D as an extra experiment on 4 data nodes. Like AWS, Google's Cloud Dataproc also uses a custom distribution. Tests were run on version 1.0, which includes Hadoop 2.7.2 and Hive 1.21 on a Debian Linux 7 OS. Versions tested for the SUTs are the default ones offered at time of cluster deployment when using the command-line tools during March to July 2016. More information on the software versions can be found on the release notes of each provider. Relating to data center zones, we have tested HDI at *South Central US*, EMR at *us-east-1*, CDP at *west-europe-1*, and CBD at both Chicago and Dallas data centers. For our on-premises cluster, we have used the versions in ALOJA during the test period which include Hadoop 2.7.1, Hive 1.2.1, and Tez 7.0. This meant that EMR was the only provider with Hive 1.0, while the rest were in the 1.2.x versions. Specific Hadoop and Hive configurations are compared later in Section V-C.

III. Background and Related work

The motivation behind this work is to expand the ALOJA benchmarking and analysis platform from Hadoop Map Reduce (M/R) *jobs* [19, 20], into SQL-on-Hadoop systems. The desire to expand in this direction is driven by the current transition of the market [17, 25]. In a previous study with Ivanov et. al., collaborations with the SPEC research group in big data benchmarking [9] led to the generation of a "Big Data Benchmark Compendium" [22], which surveys and classifies the most popular big data benchmarks. This work represents an expansion into SQL-like open source systems, the results of which are publicly accessible through the ALOJA project described below.

1) The ALOJA project [20]: The ALOJA project is an open initiative from the Barcelona Supercomputing Center (BSC) to explore and automate the characterization of cost-effectiveness for big data deployments. BSC is a research center of excellence with over 8 years of research expertise in Hadoop environments. The project relies on support from both big data product and research groups within Microsoft, as well as cloud resources from the *Azure4Reseach* [7] program, and recently from Rackspace Inc [21], and collaborates with Intel Inc. ALOJA attempts to provide solutions to an increasingly important problem for the big data community, which is the lack of understanding of what parameters, either software or hardware, determine the performance of big data workloads. The selected configuration determines the speed in which data is processed, and most importantly, the hosting budget. In this work, the current state of PaaS SQL-on-Hadoop is surveyed, in terms of both price and performance of the different offerings, in order to later optimize and make recommendations to both the providers of these platforms and to the end-users.

2) Hive [11]: has become the *de facto* data-warehousing engine on top of Hadoop, providing data summarization, analysis, and most importantly, support for SQL-like queries. The Hive engine can be controlled using HiveQL, an SQL-like language designed to abstract the Map/Reduce (M/R) jobs involved in such queries for analysts. As an example, Hive queries have been gradually replacing the use of M/R in large companies such as Facebook and Yahoo! [25]. Likewise, SparkSQL, a library for Spark including SQL-like queries, is also gaining momentum. While SparkSQL is offered by the surveyed providers, this work focuses only on Hive, as it is considered to be more mature and providers share similar versions and configurations. In Hive the default engine to manage task executions is Hadoop's M/R. However, in the latest versions Hive added support for different execution engines. Namely Tez [25] (from the Stinger project) is a popular drop-in replacement, which improves on the M/R model to provide lower latency and performance. The Hive configuration employed by the different SUTs is described further in Section V-C.

3) TPC-H [23]: In the data-warehousing context, the benchmark of choice for this study is TPC-H [23], as it is the *de facto* standard for testing the data warehouse capability of a system. TPC-H is a well-understood benchmark used by both industry and academia [22]. TPC-H also offered this study a common base upon which to compare the proposed Hadoop-based systems, between themselves and with traditional engines from disclosed reports. We have extended a derived version of the benchmark adapted for Hive for the purpose of these experiments, described in Section IV. In TPC-H, instead of representing the activity of any particular business segment, TPC-H models any industry that sells products, making it applicable to different settings. The benchmark is technology-agnostic, which makes it also suitable to test big data systems, that might not support all the expected features of traditional OLAP databases. The benchmark is composed of 22 queries designed to stress system functionality representative of complex decision support applications. The definition of the workload can be found in the latest specification [23]. TPC-H was adapted and derived very early in the development of *Hive* [12]. However, in order for TPC-H results to be published for a given system, it needs to support full ACID transactions, and be fully audited. In this sense, this study only reports running times of queries as developed and as such, is not comparable directly to the published TPC-H results. Technical details of the different queries and their *choke points* can be found In "TPC-H Analyzed" [3].

4) Related work: Most recent evaluations of SQL-on-Hadoop systems measure the power run performance of on-premises SUTs. This particular work extends the "SQL-on-Hadoop: Full Circle Back to Shared-nothing Database Architectures" [6] by Floratou, Minhas and Ozcan, where authors compare both Hive with M/R and Tez with Impala on on-premises servers. This is the first attempt to measure price-performance of cloud-based SUTs and make comparisons between the main providers. There are already several tests of Amazon's EMR services in the literature: Sruthi [24] presents

the performance and costs models of EMR (PaaS) vs. AWS (IaaS) using the Intel HiBench [13] benchmark suite, but only includes a minimal Hive-test benchmark based on Pavlo's CALDA[18] benchmark, concluding that PaaS solutions benefit from being provider-optimized. Zhang et. al. [26, 27] focuses on scheduling of jobs in EMR and on Hadoop configuration using micro-benchmarks similar to our previous work [20]. In addition, Malik et. al. [16] and [15] both compare AWS Redshift to Google's Big Query services, also with a TPC-H derived benchmark, but not including the Hadoop stack. In [5] Floratou et. al. describe the current problems associated with benchmarking SQL-on-Hadoop systems, and advocate the standardization of the process. Until a consensus is reached, we believe that the TPC-H and TPC-DS are the current standards. However, BigBench [8]—the new TPCx-BB—benchmark might replace them in the near future, as it is more representative of big data workloads. In ALOJA, BigBench is currently supported, and it is a future extension of this work to combine SQL queries with user code, alongside Machine Learning algorithms.

IV. Methodology and Results

The configuration of the different deployed PaaS clusters is left as the provider pre-configures them, so that the out-of-the-box performance can be measured. This means that the HDFS configuration, default execution engine i.e. M/R or Tez, Hive, and OS are all left unmodified. Specific configurations are later discussed in Section V-C. Our intentions are not to produce the best results for the hardware as in our previous works, but instead to survey different instance combinations, and test the clusters as the provider intended for a managed general purpose usage. In doing this, the aim is to provide an indication of experience that an entry-level user might be expected to have on a given system without having to invest in additional fine-tuning. To contrast a general purpose configuration to a domain expert fine-tuned setup, for our on-premises cluster, two configurations are presented as different SUTs: while the first has Hadoop (M/R, HDFS, YARN) properly configured and using the default configuration of the [12] repository with M/R; results are also shown for a second fine-tuned configuration with Tez [25] as the execution engine. Specific configurations are compared later in Section V-C.

As test methodology, the 22 TPC-H queries are run sequentially and at least 3 times for each SUT and scale factor, capturing running times. Another query—query ALL—is created as a post-processing step, which accounts for the total time of the full run, when all the previous 22 queries ran successfully. According to the data scale factor used, some SUTs fail to run either some or all queries; this is analyzed in a later section. We test each SUT with scale factor 1, 10, 100, 500 (a nonstandard scale factor, used only for the scalability subsection), and 1000 (1TB). Results are reported only for the scale factors in which the SUT was able to run the 22 queries. Besides *query ALL*, *query 16* is used in later sections when drilling down into both resource usage metrics and variability analysis. Query 16 has been identified [3, 15] as being the most representative query as having filtering, aggregation, expressions, joins, and sub-joins. All of the runs are *power runs*, which means that they run in isolation (within a public VM) and with no concurrency. The metrics we present are the execution times of each individual query and the cost of running each query.

The tests were run between March and July 2016, and the reported pricing and specs correspond to this period. All of the instances are using the on-demand (non-discounted) pricing, which usually is the highest. To calculate the execution cost, we multiply the run time by the cost per hour normalized to seconds of the particular setup previously presented in Table I. The reported costs do not take into account idle times or deployment and are always calculated based on the time per second, regardless of whether the service is usually charged for based on different units of time. This approach facilitates comparison between the providers, who currently measure usage time in different intervals. It appears that the industry is shifting to measuring usage to a higher precision, as there are already providers doing so by the second [1]. As a side note, prices should only be regarded as indicative; during this 4-month test period, prices were changed at least 2 times for some providers, while also new versions of the software stack were released e.g., HDI rolling HDP v2.4 from v2.3 and ERM from v4.7 to v5.0, a major version upgrade, discussed later in Section V-D. Pricing for the on-premises cluster was calculated by amortizing the HW to 3 years (at the time of purchase) and adding maintenance and upgrade costs, as described in more detail in our previous work [20].

The *driver* (query executor) for the tests is the ALOJA benchmarking engine. The code of the implementation can be reviewed on the project's page [4]. The DDLs and queries have been adapted for Hive and HiveQL based on the Hortonworks *hive-testbench* repository [12], which is the most popular adaptation of TPC-H for Hive, and used frequently to test regressions and improvements in Hive versions. It also includes a M/R based distributed *datagen* tool, to generate the data distributively using across the whole cluster, speeding up the data generation process. Due to the fact that the minimum scale factor supported by the M/R generator is 2GB, the official TPC-H *datagen* is used for 1GB scale. The source code, DDLs, and SQL queries used in the experiments are available in the D2F *git* repository [14]. The D2F repository is based on the *hive-testbench* [12] from Hortonworks, but with improved query syntax, that besides fixing some errors, makes the code HDP-agnostic to work with generic Hive. Configuration settings for Hive bucketing—13 buckets—and the use of ORC compressed columnar format has been left as in the original repository when generating the data files and used in all the reported tests.

A. Results for SUTs and scale factors

Table II shows the average TCP-H full run (query ALL) results from the benchmark runs on the different SUTs. Results are composed of more than 260 successful full TPC-H runs—this is the sum of the sequential run time of 22 queries for the different data scales of each SUT. It is organized by provider, instance type following the same order as the previous Table I, so that columns i.e. the cost per hour of the cluster specs can be compared. Default instance types are highlighted in bold. The next group of columns presents the average execution times from 1GB to 1TB. Following this is the corresponding execution cost for each run, also for the same scale factors. The hadoop1-7 instance was not able to process the 1TB data scale, denoted by *Failed* for both the time and cost fields. Main errors found are later discussed in Section V-C.

1) Execution times: The scalability of the different SUTs with respect to data size can be seen in Table II. Running times range from 20 minutes to over 1 hour for 1GB, and from 4 hours to a full day for the fastest to the slowest SUT, respectively. Relating to the best average response times, the HDI D4v2 gets

TABLE II. SUMMARY OF TOTAL RUNNING TIME AND COST BY SUT AND SCALE FACTOR

Provider	Instance type	AVG execution time in secs				Average execution cost in USD			
		1GB	10GB	100GB	1000GB	1GB	10GB	100GB	1000GB
Amazon EMR	**m3.xlarge**	2,703 s.	4,472 s.	11,329 s.	73,381 s.	$ 2.52	$ 4.17	$ 10.57	$ 68.49
	m4.xlarge	2,350 s.	3,929 s.	9,624 s.	63,841 s.	$ 1.95	$ 3.26	$ 7.99	$ 53.02
Azure HDI	A3 (Large)	2,147 s.	3,446 s.	12,071 s.	79,208 s.	$ 1.61	$ 2.58	$ 9.05	$ 59.41
	D3v1	1,665 s.	2,658 s.	9,536 s.	36,946 s.	$ 2.43	$ 3.88	$ 13.91	$ 53.88
	D3v2	1,171 s.	2,076 s.	5,442 s.	26,857 s.	$ 1.71	$ 3.03	$ 7.94	$ **39.17**
	D4v2	701 s.	1,655 s.	3,449 s.	14,205 s.	$ 2.04	$ 4.82	$ 10.04	$ 41.35
Google CDP	**n1-standard-4**	2,304 s.	4,437 s.	11,517 s.	84,065 s.	$ 1.16	$ **2.23**	$ **5.79**	$ 42.27
	n1-standard-4 1SSD	2,272 s.	4,353 s.	11,204 s.	86,452 s.	$ **1.21**	$ 2.33	$ 5.98	$ 46.18
	n1-standard-8	1,538 s.	4,406 s.	8,877 s.	42,646 s.	$ 1.54	$ 4.42	$ 8.90	$ 42.76
Rackspace CBD	hadoop1-7	3,739 s.	4,111 s.	25,353 s.	*Failed*	$ 2.83	$ 3.11	$ 19.16	*Failed*
	hadoop1-15	4,044 s.	6,357 s.	17,501 s.	84,997 s.	$ 6.11	$ 9.61	$ 26.45	$ 128.44
	hadoop1-30	3,490 s.	5,294 s.	11,853 s.	84,414 s.	$ 10.55	$ 16.00	$ 35.82	$ 255.12
	OnMetal40	2,981 s.	4,104 s.	5,892 s.	20,676 s.	$ 9.77	$ 13.45	$ 19.31	$ 67.77
on-premises	12cores-M/R	4,080 s.	6,374 s.	12,749 s.	41,677 s.	$ 3.97	$ 6.20	$ 12.39	$ 40.52
	12cores-TEZ	1,386 s.	1,998 s.	4,505 s.	17,931 s.	$ 1.35	$ 1.94	$ 4.38	$ 17.43

the best times for all of the tested data sizes. While the D4v2, with 8 cores, has a higher spec than most of its competitors by default, it is still surprising that it outperforms both the on-premises system in either configuration (M/R and Tez) and the OnMetal40 cluster, which both have higher specs than the D4v2. The HDI D4v2 also performs over twice as fast as the CDP n1-standard-8 and the CDB hadoop1-30, which both have 8 cores and similar RAM. After the D4v2, the on-premises 12-cores-TEZ gets the second-best times at all data sizes except for 1GB.

From the SUTs with 4 cores—the defaults and in bold, the HDI D3v2 gets the best average running times, followed by the D3v1 (previous generation CPU) at scales from 1GB to 10GB. Between 100GB and 1TB, the OnMetal40 performs better than the D3v1, as well as the n1-standard-8 at 100GB. Continuing to look at the 4-core VMs, the CDP n1-standard-4 performs better than both the EMR m3.xlarge and the m4.xlarge. The EMR m4.xlarge is faster than the previous generation m3.xlarge, even though it is EBS-only and the m3.xlarge has part of the data on a local SSD. Similarly, on CDP, while the n1-standard-4 with one SSD gets slightly better results than without (except at the 1TB scale), the difference is minimal. So with the current configuration of CDP, for Hive, adding SSDs leads to marginal reductions in execution times while increasing the cost. Besides the OnMetal40, the rest of the CBD SUTs have high running times, as well as scaling poorly to large data sizes. Details on system-level benchmarks and query performance is later analyzed in Section V, and scalability is later analyzed in Section IV-C. The on-premises cluster was run with the M/R and Tez execution engines; one can quickly notice the run time improvement of about 3X from 1GB to 100GB, and 2X for 1TB with Tez.

As an additional test, an specialized data-warehouse in the cloud, namely the Azure DW system, with similar resources as the 4-core SUTs has been benchmarked. The 1TB TPC-H workload executes in 1 hour and 15 mins on the Azure DW system, as compared to 4 hours for the fastest Hive system. Of course, the Hadoop-based SUTs are general purpose and open-source data engine, but it was interesting to note the current gap at this data scale with specialized stores.

B. Price-to-performance ratio

While raw performance is important, especially when optimizing a system, one of the most important features of cloud-based systems is the Pay-as-you-Go model. For illustration purposes of the price-to-performance snapshot of cloud providers at time of writing, Table II shows the total cost in USD of the average full run of each SUT. Prices range from just above one to ten USD at 1GB, and from below forty to over two hundred fifty USD at 1TB. The HDI D4v2 got the best running times, however, it also has higher specs than the default multi-tenant SUTs (highlighted in bold in the table). Price-to-Performance ratios give an insight into the actual gain relative to the cost and are easily estimated for the compute part of cloud systems. Of course, one could choose to pay more and finish an analytical job earlier, but if there is no tight deadline, cost-efficiency might be desirable.

As we can see from Table II, the most cost-performing SUTs are the CDP n1-standard-4 between 1 to 100 GB, and from 100GB the HDI D3v2. At 1GB, CDP with the SSD version is the most cost-effective, while from 10 to 100 GB, without the SSD would be cheaper to run. At 1TB scales, however, the HDI D3v2 is the most cost-effective system, with a couple of USDs difference from CDP n1-standard-4. The main reason for this shift is the decrease in scalability of the CDP SUTs when increasing the input data, compared to the D3v2 and D4v2 systems. In the same vein, the OnMetal40 scales well with data size, while its cost is higher than the CDP and HDI results—at 1TB, it has a similar price to the EMR m3.xlarge. It is likely that at higher data scales e.g., 10TB, price-to-performance rankings will keep changing since cost and performance are proportional for a given SUT. An example of this relation can be seen with the D3 series instances: while both the v1 and the newer-gen v2 have the same price, the v2 is faster, so the cost for the run is lower.

The cost-effectiveness bubble diagram is shown in Figure 1. It places each SUT at an (x,y) location of a bi-dimensional space, where the x- and y-axis represent the execution time in seconds and running cost in USD respectively. The closest to point $(0, 0)$ would be the most cost-effective and the bubble size relates to the number of repeat runs for each system. CBD hadoop1-N SUTs were omitted from the chart, to make it easier to differentiate the best performing SUTs. The diagram shows visually how fast—closer to the y-axis—and how cheap—closer to the x-axis—each cluster is. One can quickly see that since the n1-standard with 1 SSD has similar time to the GCS-only version, due to the extra cost of the SSD, it is placed higher. One can also see how the n1-standard-8 moves along the X-axis in a linear fashion reducing the running time while keeping total cost similar, representing a good scalability of the instance at this data scale. On the other hand, the D3v1 which has the same

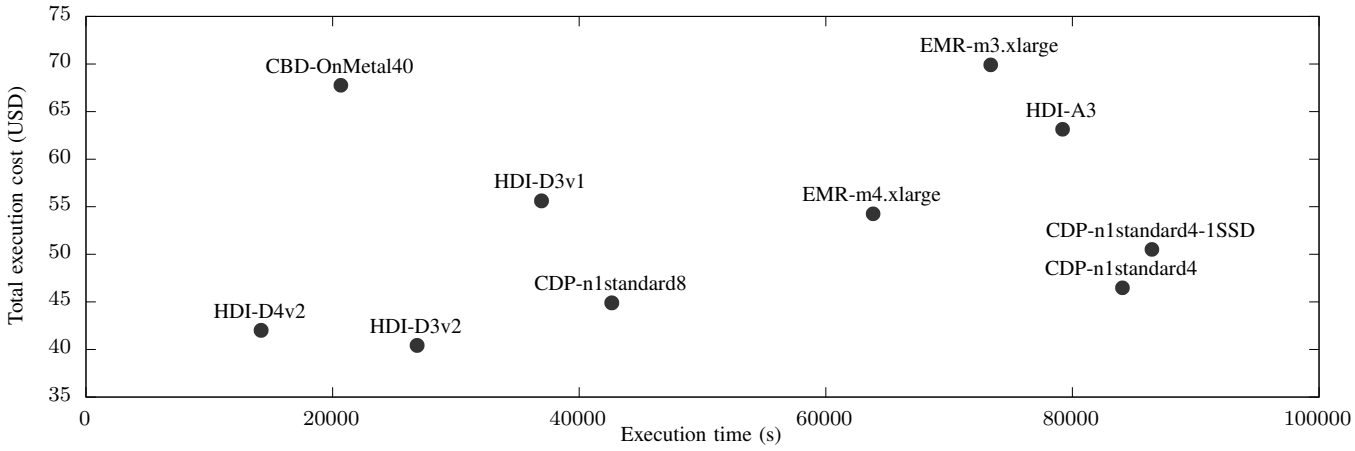

Fig. 1. Price-to-Performance ratio of SUTs at 1000GB (1TB) scale

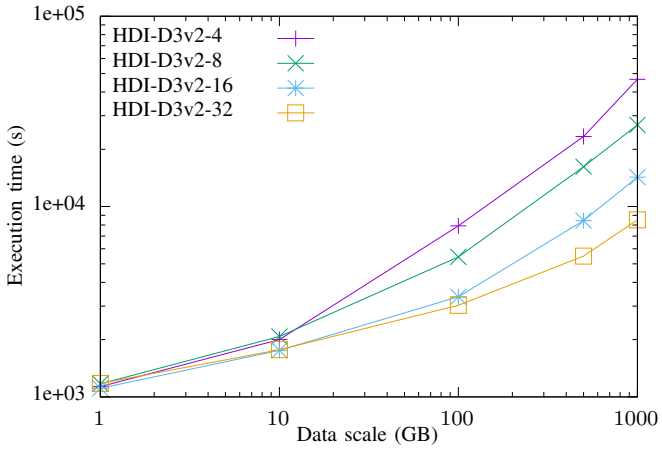

Fig. 2. Scalability of Query ALL (full run) as scale factor increases for HDI clusters from 4 to 32 datanodes

price as the newer gen D3v2 due to taking more time is placed higher in the chart.

C. Scalability to data size

Table II presented the data size scalability of each SUT on 8 data nodes. As a different experiment of scalability to cluster sizes in the cloud, Figure 2 compares the scalability of the HDI D3v2 only between 4- to 32 data nodes. At 1GB scale factor all four cluster sizes obtain similar numbers. At 100GB, the 4- and 8-node systems perform similarly, while the 16 and 32 nodes are just 14% faster. Figure 2 shows the scalability to data sizes of 4 clusters having 4 to 32 datanodes with the X-axis in logarithmic scale. It can be seen that by increasing the number of nodes in the cluster, one can obtain lower execution times for the full TPC-H run as expected. At 100GB, 32 nodes is 2.6X faster than having a 4 datanode cluster size, while at 500GB and 1TB the improvement increases to 4.3X and 5.5X respectively. Scalability in datanodes is less than linear (ideal), but one can see that the larger the data size, the more efficient the larger clusters are. At 1TB data scales the full benefit of larger cluster i.e., 16- to 32 data nodes is still not reached, and larger data scaled would be needed to compare them more fairly. The same can be said of the OnMetal instance, where more tests and larger data and cluster scales would be needed.

V. Impact of Hardware and Software on runtimes

This section compares the Hardware (HW) and Software (SW) differences between the providers and SUTs, primarily to rationalize the performance differences that have been observed, while also providing an insight into the architectural variations between each service. Additionally, a complementary set of TPC-H experiments are included, to provide an insight into the life-cycle of the PaaS services.

A. System-level benchmarks

To further understand and verify HW performance limits of the SUTs, micro-level benchmarks of CPU, memory, network, and disk I/O bandwidth (BW) have been run, utilizing standard Linux tools already implemented in ALOJA [4]. The intention of these tests is to provide additional information about the application TPC-H based benchmarks and validate assumptions. However, the following results should only be considered indicative. In order to get further insights, access to the underlying HW would be required, but these additional tests are considered to be out of the scope of this initial survey. For example, low-level HW architectures are generally abstracted by the *hypervisor* of the VMs in the cloud. Furthermore, as cloud VMs are usually very different among providers, a common approach for comparing cross-provider VMs is to compare servers with similar CPU core counts and RAM sizes. However, as not all CPUs and memory buses are from the same generation and frequency, we have run CPU and RAM memory (MEM) BW benchmarks using the *sysbench* package in ALOJA.

1) CPU-MEM: the *sysbench* tests were run using 1-thread configuration to get a comparable measure between CPU-MEM generations, as the SUTs had varied core counts. We classified results from the SUTs into 3 groups of CPU/MEM performance. Firstly, the OnMetal nodes scored at least 2x higher than the rest of the systems followed by the on-premises cluster, as expected. Secondly, for the shared VMs, the EMR m4.xlarge got the best numbers, followed closely by both the CDP n1-standard series and the HDI D-series which shared almost the same scores both for CPUs and memory BW. Finally, the HDI A3s, the EMR m3.xlarge, and the CBD hadoop1-series had the lowest performance. In conclusion, even though EMR, CPD, and HDI default to 4-core nodes, they belong to at least two different generations, and a third if the OnMetal SUT is included. All

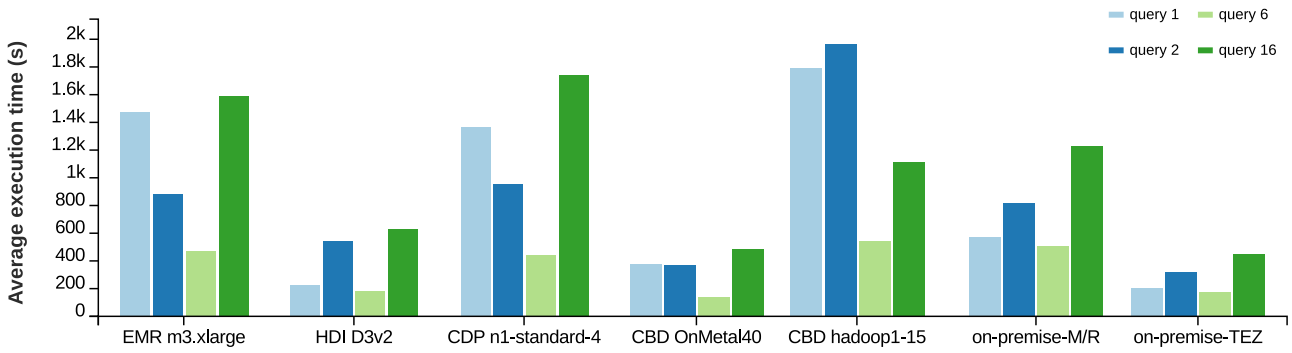

Fig. 3. Average execution times of *scan* queries (1 and 6) vs. *join* queries (2 and 16) at 1TB for default SUTS and on-premises M/R and Tez

of the CPUs were reported as being Intel Xeons, except for the CDB shared VMs, which also featured AMD Opterons.

2) NET: to measure network BW between two nodes in the SUT clusters we used the *iperf* network utility with 100GB of total payload and with varying numbers of threads, including 1 and the number of cores of the node. In this case, EMR had the best scores with numbers close to 10Gbp/s on both SUTs. EMR was followed by CDP, with numbers that varied between 5–8Gbp/s between runs, then CBD, which had a maximum throughput of about 5Gbp/s. On HDI, the network bandwidth varied depending on the SUT, meaning that the network is throttled according to the VM size: for the D4v2-series, bandwidths of up to 6Gbp/s were obtained, whereas for D3v2 the maximum was 3Gbp/s, for D3v1, 2Gbp/s, and for the A3s, a limit of 1Gbp/s was observed. The on-premises cluster had a close to 1Gbp/s result, which matches the physical network bandwidth. It is noteworthy that besides the on-premises results, there was great variation between the *iperf* runs e.g., the 5–8Gbp/s on CDP, indicating that the network is shared with other cloud tenants and not as reliable.

3) Disk I/O: for big data systems, traditionally local disk Input/Output (I/O) has been the main bottleneck. To account for disk Read/Write (R/W) BW, we have run simple tests using the *hdparm* and *dd* linux commands also with at 100GB payload on the mount points used for the HDFS. The OnMetal SUT again had the best numbers, having high performing SSDs with 2000/1000 MB/s R/W each; disk specs can be reviewed in Table I. HDI again has different numbers by SUT series: The D4s 500/250—getting the best shared VM numbers, D3s 250/100, and the A3s 120/100 MB/s R/W respectively. The rest of the SUTs had similar numbers to those of the A3s (and in general to SATA drives) around 120/100 MB/s including our on-premises cluster, except for the VMs with extra SSD and the CBD shared VMs. The CDP SUT with 1 SSD that achieved 400MB/s symmetrically for R/W, and the EMR m3.xlarge achieved 280/125 MB/s R/W. On CBD, all shared VMs showed a very low write throughput, around 20 MB/s, and needs to be investigated further.

4) DFSIO [20]: opposed to classical Hadoop, most of the benchmarked cloud systems store the HDFS data on networked disks through custom drivers and vendor specific settings, using local disks for temporary data only. Furthermore, the disk I/O tests measure single disk performance, the total aggregated BW of combining multiple disks was not measured. To better understand the total BW available to the Hadoop applications, we have also run tests of the DFSIO benchmark, again using ALOJA [4]. DFSIO are simple test benchmarks that either *write* or *read* data using M/R, but without making use of the data. Results Read/Write MB/s results for DFSIO are : CDB OnMetal 615/315; CDP n1.standard series 50/29, with 1 SSD 321/91; HDI D4v2 57/50, D3v2 46/38, D3v1 32/35, A3 30/16; EMR m3.xlarge 30/16 and the m4.xlarge 35/31; CBD cloud based hadoop1-7 30/3, hadoop1-15 29/4, hadoop1-30 73/5; and the on-premises 50/228 MB/s R/W respectively. Analyzing results, again the OnMetal high-end dedicated HW scored best in this test, the combined use of SATA drives in the on-premises cluster gave good BW write results. HDI scored again by VM type and the CBD cloud-based had a poor write BW and in the disk I/O tests and has been reported. CBD also had an older version of HDP (HDP 2.3) which contains a *buggy* version of DFSIO which might also have contributed to the low results. Interestingly, the CDP n1.standard system with 1 SSD gets the best BW number for raw HDFS I/O—after the OnMetal, this benefit is not translated to good scalability on the Section IV-A1 TPC-H results. Another remark is that the EMR m4 gets higher BW than the m3 SUT which has 2 local SSD drives indicating also newer generation of HW on the m4, as well as the general good performance of EBS.

B. TPC-H performance metrics

Going back to the TPC application-level benchmark, queries can be classified by the SQL predicates and operations they perform and in general to their system resource bottlenecks [3, 15]. For example, queries 1 and 6 are mainly *scan* type queries, which implies that they should be highly parallelizable and their main bottleneck is I/O. Query 1 also requires some CPU for aggregates. On the other hand, queries 2 and 16, contain multiple joins, creating the need to *shuffle* data often between nodes requiring fast networks, high-CPU for calculations, and being more complex to parallelize.

In this context at 1TB scale, we found that the CBD OnMetal and HDI D4 and D3 series performed the best respectively, on both queries 1 and 6; while CDP, only with the n1.standard.8 for scan-type queries. Both EMR SUTs, CDP n1.standard.4 with and without local SSD—in contrast to DFSIO, and the shared CBD SUTs performed poorly on I/O type queries. For queries 2 and 16—join-types, HDI D4s got the best runs, followed by the CBD OnMetal, then the HDI D3s. It was interesting to see, that the D4s with only 8-cores and using remote disks were able to obtain better numbers than the CBD high-end OnMetal cluster for join-type queries. Figure 3 shows the average execution times for queries 1, 2, 6, and 16 for the defaults SUTs and both on-premises configurations. The

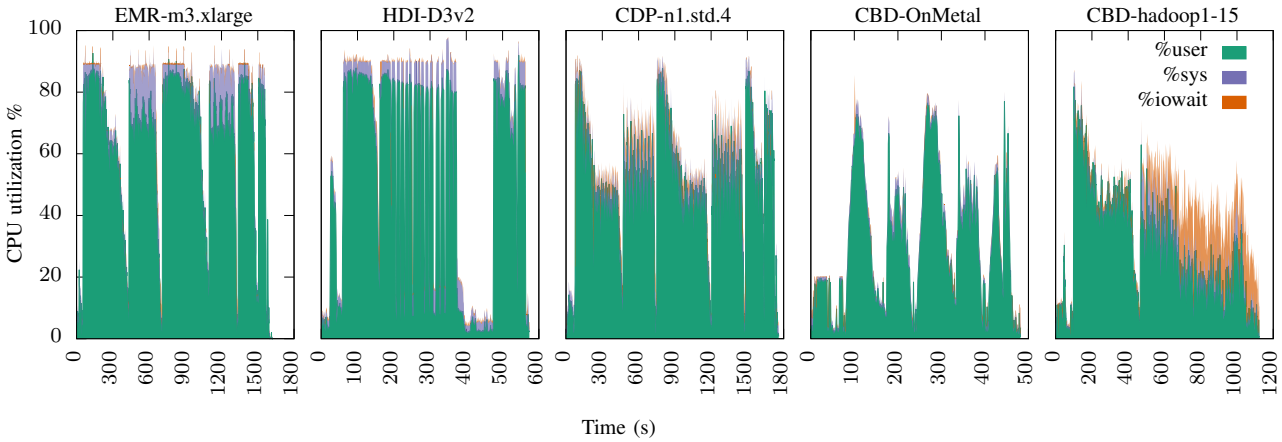

Fig. 4. CPU peformance details of Query 16 on default SUTs

on-premises configurations shows the difference of fine-tuning Hive, over a baseline configuration.

As the ALOJA platform is used to drive the benchmarks, detailed performance metrics and logs are captured for each query. Figure 4 shows the CPU percentage usage when running TPC-H query 16 (the most representative query) with 1TB data scale, averaged across data-nodes, but excluding the master node as it has a different behavior. To highlight main differences of default SUTs and for space limitations, only CPU% of default systems are presented here. CPU% charts gives an indication on how busy the cluster was during the different phases of the query execution (usually seen as humps) of the *map* and *reduce* and sub-jobs. CPU% charts show the different modes (system, user, I/O wait) on all the node CPUs (Y-axis) over the duration of the query run (X-axis) in seconds; a high I/O wait indicates either disk or network bottlenecks. From Figure 4, we can see that while the CDP, EMR, and CBD OnMetal SUTs follow a similar trend each having 5 main stages—humps, the HDI SUT has a very distinct pattern containing *spikes*. The CBD hadoop1-15 shows a very high I/O wait, leaving the processor under-utilized and impacting the total running time. Similar, but less pronounced, the CDP instance also presented a with I/O wait and under utilization. Other interesting differences are that on the CBD OnMetal SUT, the CPU is the least busy and likely has over-dimensioned HW for the tested data scales; the EMR m3.xlarge is the most used, also with the highest percentage of system mode. The distinctive pattern of the HDI SUTs and in general its good results over the rest of the providers for similar HW, led us to analyze further the SW configuration SUTs detailed in the next section; as well to replicate a similar configuration of HDI on our on-premises cluster as shown in Figure 3.

C. Software configuration differences

While this work focuses on the out-of-the-box performance of PaaS services, the difference in execution times from Section IV-A1 for SUTs with similar HW led us to compare configuration choices in detail as summarized by Table III. Note that while the Java versions varied from 1.7 to 1.8, all providers used the OpenJDK versions, as opposed to our on-premises cluster, which used Oracle's JDK 1.7 as traditionally recommended. At the HDFS level, all providers used their object networked-based storage, except for CBD and our on-premises cluster. As object stores are typically replicated besides by Hadoop, and in the case of OnMetal using HW RAID, it was interesting to see that only CDP and CBD lowered the HDFS replication to 2 copies. Most block sizes were the default at 128MB, while only EMR used the default *file buffer sizes*. EMR and CBD both compressed the map outputs by default, and each tuned the I/O factor and the I/O MBs.

While there were some differences at the Java/Hadoop level, tuning Hive had the most significant impact on performance. It was remarkable that only HDI had the Tez execution engine enabled by default. Tez is a project closely related to the HDP platform used by HDI and CBD, and HDI having a more recent version of HDP might be part of the reason it is enabled by default. To verify the impact of Tez, for the on-premises cluster we have also tested with the Tez engine. As can be seen from the main results in Table II, Tez reduces the total TPC-H running time by 2-3x. However, to implement Tez support in ALOJA it took over three weeks of effort to have it installed outside the HDP distribution and especially to fine tune it by a domain expert to achieve the performance improvements presented. There were also differences in other Hive parameters among providers, such as using the *cost-based optimizer*, *vectorized execution* and bucket settings. The provider that enabled most performance improvements in Hive i.e., HDI, got the best results for similar HW. As a note on an EMR release after the tests were performed, it has been verified that EMR also enables Tez by default (among other upgrades), which improves the performance on their platform significantly.

Errors found during tests: while results presented are only for runs that have finished successfully—except for the CBD hadoop1-7 SUT at 1TB scale, there were errors on our first approach to the systems and on some runs later on. None of the problems that we experienced arose from HiveQL syntax errors, as we found that the smallest factor (1GB) always ran successfully on all SUTs. Initially the intention was to use 4 data nodes clusters for this study, however, not all of the tested SUTs were able to scale up to 1TB which was our target data size. This was mainly due to the Hadoop containers per node surpassing the memory capacity using the provider's default configurations. This fact was confirmed by looking at the specific errors in the Hadoop/Hive logs, as well running the tests at larger node counts successfully without changing the configurations. Another common error that we had was surpassing the HDFS

TABLE III. MOST RELEVANT HADOOP-STACK CONFIGURATIONS FOR PROVIDERS

Category	Config	EMR	HDI	CDP	CBD (On Metal)	on-premises
System	OS	Linux AMI 4.4	Ubuntu 14.04	Debian 7.0	CentOS 7.0	Ubuntu 14.04
	Java version	OpenJDK 1.7.0_111	OpenJDK 1.7.0_101	OpenJDK 1.8.0_91	OpenJDK 1.8.0_71	JDK 1.7
HDFS	File system	Local + EBS	WASB	GCS	Local	Local
	Replication	3	3	2	2	3
	Block size	128MB	128MB	128MB	256MB	128MB
	File buffer size	4KB	128KB	64KB	256KB	64KB
M/R	Output compression	SNAPPY	FALSE	FALSE	SNAPPY	FALSE
	IO Factor / MB	48 / 200	100 / 614	10 /100	100 / 358	10 / 100
	Memory MB	1536	1536	3072	2048	1024 | 4096
Hive	Hive version	1.0.0-amzn-5	1.2.1000.2.4.2.4	1.2.1	1.2.1.2.3.4.0	1.2.1
	Engine	M/R	Tez	M/R	M/R	M/R | Tez
	ORC config	Defaults	Defaults	Defaults	Defaults	Defaults
	Vectorized exec	FALSE	Enabled	FALSE	FALSE	Enabled
	Cost Based Op	FALSE	Enabled	Enabled	Enabled	Enabled
	Enforce Bucketing	FALSE	TRUE	FALSE	FALSE	TRUE
	Optimize bucket map join	FALSE	TRUE	FALSE	FALSE	TRUE

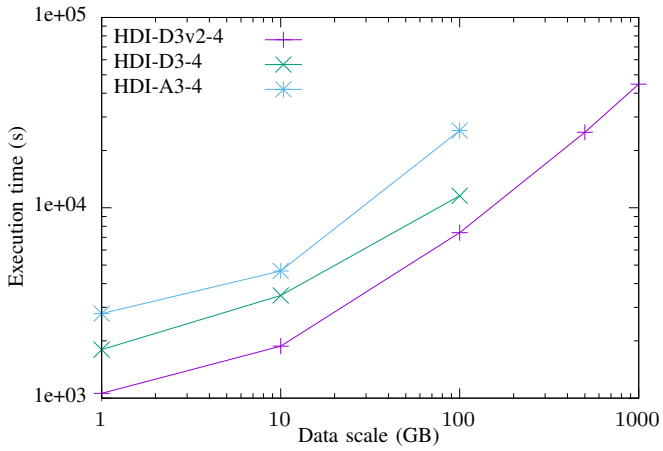

Fig. 5. Log–log plot showing scalability over HDI default VM generations

capacity when running initial tests, specifically for EMR and CDP. When dimensioning storage capacity one has to account for HDFS replication and it differs by provider in Table III, as well as the amount of intermediate data depends on the specific configuration of the SUT. In EMR and CDP you have to select the size of each data disk on cluster creation, we had to recreate clusters to increase the capacity on the same node-count. On EMR there was a setting present in the YARN configuration that would shutdown all the nodes when reaching 90% disk capacity, and was difficult to spot. After some experimentation we found that we needed 5TB of data to fit all of our experiments, sizing the SUTs to at least this capacity, but having to pay for the extra capacity not used. In HDI, storage is completely elastic, so the problems we found have been capacity based on system resources i.e., machines were swapping (out of RAM). We found that this was not only because of Hadoop YARN tasks, but due to auxiliary services, e.g. Ambari metrics collector, having a high RAM utilization. The errors found along the way have been reported and the metrics collector error was fixed in HDP 2.4.

D. Software and Hardware improvement over time

Cloud providers constantly update and upgrade their services with new features, software versions, security patches, as well as new hardware. In this section, an overview is presented of the changes observed while working with HDI and EMR, to give an insight into the life-cycle and improvements over time of Hadoop-based PaaS services. During the last 2 years spent running benchmarks on Azure systems (see [19, 20]), the HDI service has continued to upgrade the underlying hardware of its VMs, from the A3 (Large) default during 2014, to the D3v1 during 2015, and now the D3v2. Both machines in the D-series have local SSD drives and faster networking (10Gbp/s), as previously noted. In an earlier test, response times for the 3 systems of full TPC-H runs on HDP 2.3 and 4 datanodes were benchmarked, the data size scalability improvement can be seen in Figure 5. Over this time, not only has the A3 system halved in price but the D3v1 when launched offered a 2.2x performance improvement over its predecessor. In 2016, the D3v2 is the current default, at the same pricing as the D3v1, while offering a newer gen CPU. The D3v2 is 3.4x faster than the original A3 and 0.55x faster than the D3v1, resulting in faster performance for the same price in less than a year's time. The improvement is not only in execution times on all the scale factors, but also in reliability. In Figure 5, one can see 100GB is the maximum data scale for both the A3 and D3v1 (on HDP 2.3). This was due to capacity problems of the SUTs to process more data; the D3v2 however was able to process up to 1TB.

In relation to software updates, the HDI service has also been improving over time in SW versions; when upgraded from HDP version 2.3 to 2.4, the 4 data nodes D3v1 obtained a 35% execution time improvement at 100GB scale. It was also interesting to note that with the HDP 2.4 upgrade, the D3v1 was able to process up to 1TB of data scale. On the EMR side, after our main experiments period presented in Section IV, the EMR service had a major release upgrade to their software stack in August 2016. The change was from EMR 4.7 to 5.0. EMR 5.0 provides both the new major releases of Hive and Spark to v2, as well as other services such as having Tez by default, and this has proved to be a major differentiator of HDI, as shown in Section V-C. In the EMR case, the execution time was reduced by more than a factor of 2 when upgrading from version 4.7 to 5.0 on a 4 data nodes cluster of the m4.xlarge SUT. In both providers, previous versions are still available to users for a time—in HDI the last 3 updates are still available, largely to maintain compatibility with other software that users might depend on. To benefit from both the hardware and the software upgrades in the cloud in general, it is necessary to re-deploy one's clusters, as the upgrade is not automatic. When storage is decoupled from compute, redeployment does not require data migration, simplifying the process.

E. Analysis on Service Time Variability

To illustrate the current variability in the cloud, Figure 6 presents the box plot of the HDI A3 and D4v1 (newer gen, higher specs) SUTs, comparing it to the on-premises cluster (here labeled M100). The plot has two entries for each SUT at 4 and 8 data nodes cluster sizes and there are at least 3 runs for the full TPC-H at 100GB data scale. One can quickly see that there is a great variation in time (whisker distance) on the A3s SUTs on both clusters sizes. However, on the D4v1 SUTs, this variation is minimal, even lower in this case than on the on-premises cluster. While further analysis would be required to quantify the predictability, as many factors including multi-tenancy and time of the day might influence service times in the cloud, there is still a clear trend that the newer generation of VMs with faster processors, network, and especially with local SSDs, can provide a reliability closer to fully dedicated, on-premises hardware.

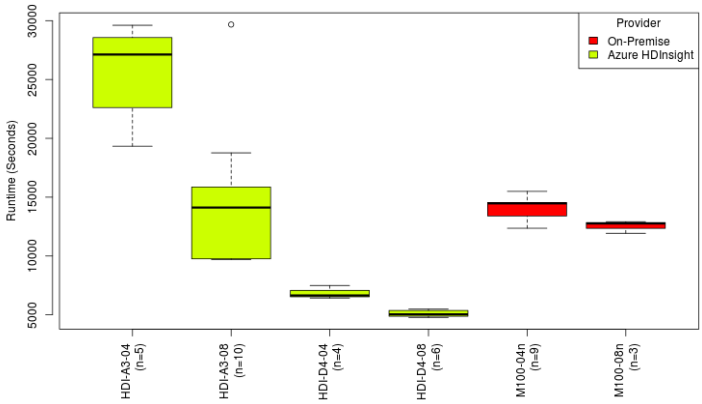

Fig. 6. Comparison of executions of full TPC-H on 100GB on HDI-A3 vs. HDI-D3v2 vs. on-premises deployments using the default configuration

VI. Discussion and Conclusions

This article presented a snapshot of the out-of-the-box performance and prices as of March-July 2016 of popular PaaS solutions, as well as providing an insight into the scalability and product upgrade life-cycle. It was shown that cloud providers share many similarities in hardware offerings. For example, most providers on their multi-tenant VMs are defaulting to 4-core nodes and about 16GB of RAM. This leaves Hadoop containers with roughly 4GB of RAM per CPU-core, which is enough to process databases up to 1TB in size from 8 data nodes clusters with Hive v1. However, this configuration might not be sufficient to run other in-memory frameworks efficiently in the future i.e., Spark or Giraph, or even Hive 2 with in-memory cache. While entry-level clusters standardize at 4-cores, there is a significant difference in CPU generations, which impacts performance greatly. The hardware generation a user gets is related to how often providers upgrade their offerings. In some cases, upgrades can be frequent e.g., for HDI, more than once per year, but customers need to redeploy their clusters after each upgrade in order to benefit from the improvements. All providers are also defaulting to VM instances with local SSD disks and faster networks (currently at 10Gbp/s), which can now rival the speed of on-premises commodity clusters, while also reducing the variability between repeat runs of the same workload. Relating to the older generation VM instances that were tested, it can be concluded from the performance and variability in service time that *commodity VMs* (cheap, but older generation) in the cloud are not the same as the *classical* commodity hardware found in Hadoop literature, and should be avoided when possible.

The providers tested support elastic storage by default, each adopting a different approach, while managing and optimizing transparently for their users. Implementations vary between local disk and over-the-network storage. However, there is a divergence in the way that storage and compute are coupled. In HDI—and Azure in general—permanent storage is fully disaggregated from compute, allowing the two to scale asymmetrically, while also permitting the sharing of resources. Constrastingly, in EMR and CDP the storage size per node is user-defined, but specified at cluster creation time, so more compute nodes must be added in order to increase storage. EMR still supports a mixed mode with local storage on older gen VMs i.e., the m3.xlarge, but EBS-only on newer instances, which also showed better performance. CDP also allows adding local SSDs as tiered-storage for Hadoop, and this improved the speed of the DFSIO benchmark, but not the Hive TPC-H benchmark, due to the configuration not being fine-tuned. CBD, on the other hand, still favors local processing; while Openstack-based and AWS S3-compatible object stores are supported, they were not tested.

Performance results show that networked storage can provide adequate query run times, while simplifying decision making in how to dimension storage on the cluster from the very start. Instances with network storage in HDI, CDP, and EMR obtained comparable results to both the CBD OnMetal and our on-premises SUTs. While the OnMetal and the CDP n1.standard.4 with 1 SSD obtained better micro-benchmark results i.e., DFSIO and disk tests, application-level Hive configuration has a high impact on the final performance, as was the case with the tuned HDI instances. From the results for our on-premises cluster, it has been verified that a fine-tuned Hive configuration can improve the performance by 2 to 3 times, especially when the Tez execution engine was enabled as in the second configuration of our cluster. In this sense, there is room for improvement on all the providers for fine-tuning the default configuration, at the cost of domain expert support and time. This is particularly the case for the CBD OnMetal cluster, where system utilization was low, but run times were not always the best. This indicates that more tests are needed and different instance and cluster sizes must be evaluated for this platform, implying, of course, more test costs and time to be dedicated in future work. In addition, it has been observed how updates of the software stack to newer versions improved performance for both the HDI and EMR cases, and would likely benefit the rest.

While there is similarity in the instance's hardware offerings, pricing and billing can be quite different. Per-hour prices of tested SUTs varied from around 2 to 12 USD dollars for the whole cluster. For network-based storage, the used capacity is billed separately from the compute cluster so it also incurs extra costs, but these costs were found to be minimal in comparison—especially for AWS. The main cost was the compute, which elastically scales to the number of nodes to satisfy the current processing speed and budget limitations e.g., leaving a minimal cluster or even turning it off and keeping the data in the external storage.

CDP's billing by the minute can be very attractive, combined with very short deployment times, making them ideal for short-lived clusters. It is expected that other providers will follow their

lead, billing in shorter increments and improving deployment times in the near future [1]. Relating to price-performance, results show that the price-to-performance ratio for the best SUTs is within a 30% cost difference for the 1TB scale. It is also shown that the best price-performing SUT varied on the data scale used at the moment; while CDP was more cost-effective from 1 to 100 GB, HDI excelled at scales exceeding 100GB using provider default instances.

Cloud providers have come a long way during the last couple of years to provide fully-elastic on-demand SQL-on-Hadoop based solutions. These improvements facilitate planning-free infrastructures by separating compute from storage, as well as providing newer generation hardware improving on previous service times and reliability. At the same time, providers offer managed software and hardware updates for new deployments, fine-tuned for general purpose use with comparable performance results to commodity clusters for the tested medium-size clusters. The market is fast-paced, with cloud vendors upgrading their services as frequently as daily in some cases, resulting in them overtaking one another in performance frequently. The choice of provider depends as usual, on the use case, and performance requirements, as well as the final budget. The results of this study are published with logs and performance, and are publicly available on the ALOJA website [4].

The expansion of this survey of the marketplace remains the subject of future work, a process which will require the usage of more resources and time for experiments. Particular subjects of interest are the new versions of Hive and Spark, which will be tested once they are production-ready and have comparable setups across the different providers. Finally, it is planned to continue modelling the results-bed, and to characterize the different providers and configurations using the ALOJA-ML Predictive Analytics features [2].

Acknowledgements

This work is partially supported by the Microsoft Azure for Research program, the European Research Council (ERC) under the EUs Horizon 2020 programme (GA 639595), the Spanish Ministry of Education (TIN2015-65316-P), and the Generalitat de Catalunya (2014-SGR-1051).

References

[1] Outscale introduces per second billing boost cloud: http://www.businesswire.com/news/home/20160915005885/en/world-outscale-introduces-per-second-billing-boost-cloud, Sept 15 2012.

[2] Josep Lluis Berral, Nicolas Poggi, David Carrera, et al. Aloja-ml: A framework for automating characterization and knowledge discovery in hadoop deployments. In *21st ACM SIGKDD Conf. on Knowledge Discovery and DataMining, 2015, Australia*.

[3] Peter Boncz, Thomas Neumann, and Orri Erling. Tpc-h analyzed: Hidden messages and lessons learned from an influential benchmark. *Performance Characterization and Benchmarking, TPCTC 2013*.

[4] BSC. Aloja home page: http://aloja.bsc.es/, 2016.

[5] Avrilia Floratou et al. Benchmarking sql-on-hadoop systems: Tpc or not tpc? *Big Data Benchmarking: 5th International Workshop, WBDB 2014, Germany, 2014*.

[6] Avrilia Floratou, Umar Farooq Minhas, and Fatma Özcan. Sql-on-hadoop: Full circle back to shared-nothing database architectures. *Proc. VLDB Endow.*, 2014.

[7] Microsoft Azure for Research. Project home page: http://research.microsoft.com/en-us/projects/azure/, 2015.

[8] Ahmad Ghazal, Tilmann Rabl, Minqing Hu, Francois Raab, et al. Bigbench: Towards an industry standard benchmark for big data analytics. In *ACM SIGMOD 2013*.

[9] SPEC RG Big Data Working Group. https://research.spec.org/working-groups/big-data-working-group.html, 2016.

[10] Hortonworks Data Platform (HDP). http://hortonworks.com/products/hdp/, 2016.

[11] Apache Hive. https://hive.apache.org/, 2016.

[12] Hortonworks Hive-testbench. Tpc-h benchmark repository: https://github.com/hortonworks/hive-testbench, 2016.

[13] Shengsheng Huang et al. The HiBench benchmark suite: Characterization of the MapReduce-based data analysis. *Data Engineering Workshops, 22nd Int. Conf. on*, 2010.

[14] Todor Ivanov. D2f tpc-h benchmark repository: https://github.com/t-ivanov/d2f-bench, 2016.

[15] Jialin Li et al. Performance analysis of cloud relational database services, June 2013.

[16] Saif Ur Malik, Samee U. Khan, et al. Performance analysis of data intensive cloud systems based on data management and replication: A survey. *Dist. Parallel Databases*, 2016.

[17] Mike Gualtieri Noel Yuhanna. Elasticity, automation, and pay-as-you-go compel enterprise adoption of hadoop in the cloud. *The Forrester Wave: Big Data Hadoop Cloud Solutions*, Q2, 2016.

[18] Andrew Pavlo, Erik Paulson, Alexander Rasin, Daniel J. Abadi, David J. DeWitt, Samuel Madden, and Michael Stonebraker. A Comparison of Approaches to Large-Scale Data Analysis. In *SIGMOD*, pages 165–178, 2009.

[19] Nicolas Poggi, J. L. Berral, D. Carrera, N. Vujic, D. Green, J. Blakeley, et al. From performance profiling to predictive analytics while evaluating hadoop cost-efficiency in aloja. In *Big Data (Big Data), 2015 IEEE International Conference on*, 2015.

[20] Nicolas Poggi, David Carrera, Nikola Vujic, Jose Blakeley, et al. ALOJA: A systematic study of hadoop deployment variables to enable automated characterization of cost-effectiveness. In *2014 IEEE Intl. Conf. on Big Data, Big Data 2014, Washington, DC, USA, October 27-30, 2014*.

[21] Inc Rackspace US. Company home page: http://www.rackspace.com/, 2015.

[22] Meikel Poess Anna Queralt John Poelman Nicolas Poggi Jeffrey Buell Todor Ivanov, Tilmann Rabl. Big data benchmark compendium. *7th TPC Technology Conference on Performance Evaluation and Benchmarking (TPCTC 2015) USA*.

[23] Transaction Processing Performance Council. TPC Benchmark H - Standard Specification, 2014. Version 2.17.1.

[24] Sruthi Vijayakumar. Hadoop based data intensive computation on iaas cloud platforms. *UNF Theses and Dissertations*, page Paper 567, 2015.

[25] Tez Yahoo Betting on Apache Hive and YARN. https://yahoodevelopers.tumblr.com/post/85930551108/yahoo-betting-on-apache-hive-tez-and-yarn, 2014.

[26] Zhuoyao Zhang, Ludmila Cherkasova, and Boon Thau Loo. Exploiting cloud heterogeneity for optimized cost/performance mapreduce processing. In *CloudDP 2014*.

[27] Zhuoyao Zhang et al. Optimizing cost and performance trade-offs for mapreduce job processing in the cloud. In *NOMS 2014*.