

Comparative Performance Evaluation of Hadoop on PaaS Proposals by Leveraging HiBench

Uluer Emre Özdil^{a,b,*}, Serkan Ayvaz^{a,c,*}

^a*Bahçeşehir Üniversitesi, Beşiktaş - İstanbul, Turkey*

^b*School of Engineering, Big Data Analytics and Management*

^c*Department of Software Engineering*

Abstract

The advent of Big Data emerged in any data-driven domain and scaled up the extent as well as the depth of data and its handling. In an ongoing maturing process, new approaches leaning on enhanced distributed storage and computing paradigms are invented helping overcome management and running analytics challenges. In this context Hadoop is embraced in a wide scale by beneficiaries both from industry and academia since its first release in 2005. The commercialization of Cloud Computing started a grand migration movement towards cloud, applying also for Hadoop transferring its presence from on-premises to virtual machines stored and tamed in large data center facilities by global Cloud Service Providers. The CSPs' response to result-focused analytics purposes emerged a service called managed systems where the hard workload of multi node cluster implementation is overtaken by the contractor providing a pre-configured Hadoop package simplifying the installation process to a matter of property selection thus eliminating technical know-how requirements on such an implementation. Converting the concept of cloud based Hadoop from IaaS to PaaS apparently reduced costs commercially presented as pay-as-you-go or pay-per-use. There is a payoff, though, managed Hadoop systems do present a black-box behavior to the end user who cannot be clear on the inner perfor-

*Corresponding authors

Email addresses: ulueremre@gmail.com (Uluer Emre Özdil),
serkan.ayvaz@eng.bau.edu.tr (Serkan Ayvaz)

mance dynamics, hence the benefits by leveraging them. In the study we selected three global providers (GCP, Azure, and Alibaba Cloud), activated their Hadoop PaaS services (Dataproc, HDInsight, and e-MapReduce, respectively) within same geographical region and by promise apparently same computing specifications, and executed several Hadoop workloads of the HiBench Benchmark Suite. The results yield that apparently same computation specs among CSPs' services do not necessarily guarantee equal or close performance outputs to each other. Our assumption is that the pre-configuration work of managed systems done by the contractor play a weighing role on their performance.

Keywords: `elsarticle.cls`, Benchmark Hadoop PaaS, HiBench, Performance evaluation

2010 MSC: 00-01, 99-00

1. Introduction

Big Data. An indispensable aspect for enterprises and academia of the information era to deal with is the term Big Data. As the estimated global internet access rate covers a weighing majority of roughly 63% of the global human population by 2020 [1], mobile technology devices become democratized, sensors and IoT devices the more occupy daily life, ongoing scientific researches produce vast amounts of data outputs the Big Data phenomenon gathered itself by means of overwhelming size with Volume, ever accelerating growth rate with Velocity, and diverse data structures with Variety, new approaches were forced to mature in order to ease the maintenance of Big Data and enable extracting valuable insights from it leveraging complex statistical formulae.

Hadoop [2]. Frameworks for distributed storage and computation sparked up first by web indexing engines were inherited and developed further by the open source community yielding what is known as Hadoop and its ecosystem today. Considering the complexity of dealing with big data Hadoop represents a modern analytics framework decreasing management efforts and duration of analytics operations to an acceptable level by means of affordable commodity computers.

Hadoop comprises three for its core functionality: HDFS filesystem for storing very large data across a cluster of nodes, the MapReduce framework developed for distributed computation, and YARN for allocating available resources for the requested tasks.

HDFS. Hadoop Distributed File System which stands for HDFS is developed with inspiration from the guidelines described in a whitepaper about Google's Filesystem (GFS), a distributed storage paradigm to handle petabyte and larger scale data volumes within a cluster robust to machine failures, published in 2003 [3]. Big data volumes are chunked and stored within 128 MB blocks, each block is replicated to different nodes by a factor of 3, these values are the defaults and can be changed. When the data file is requested, related blocks are constructed from the nodes across the cluster. The redundancy of the blocks guarantees availability, in a case where one or more nodes become out of service, requested data blocks are gathered from the available redundant copies stored on other nodes. HDFS is a co-existing file system on the nodes it is installed, it provides a global distributed view to the files across the cluster, thus listing an HDFS directory is possible from within all nodes, the files on HDFS are listed as they exist on a local filesystem, but the physical parts of the files reside on other physical locations. Figure 1 depicts an overview to HDFS. The architecture of HDFS comprises Namenode which is a dedicated machine to keep track of the files and folders and respective metadata like block locations across the cluster, and a number of datanodes on which the data blocks are residing [4]. Namenode is a single-point-of-failure meaning if the namenode is down, the whole Hadoop system is down. To overcome this issue starting with version 2 Hadoop matured to a High Availability configuration depicted in Figure 2 where there exists two namenodes, one active namenode and one standby namenode communicating with the data nodes and storing edit logs in a shared folder. As their naming convention refers, active namenode is in charge whereas the standby node behaves more like a shadow system. Whenever the active namenode breaks down, the shadow node gets activated resulting in no

service breakage for the end user.

MapReduce. Similar to HDFS, Hadoop's MapReduce is the open source implementation of the MapReduce framework described in a paper [5] published in 2004. Depicted in Figure 3, a MR flow starts with assigning blocks from HDFS as input splits to mappers, computed intermediate results are then shuffled and passed to reducers where outputs are send back to the client. Nodes running mappers and reducers provide parallel processing which can be scaled by simply adding new commodity computers. The computation is made on nodes where related data blocks reside, which explains the term data locality bringing computation to the data thus eliminating the need for moving data across nodes for computation.

YARN. Hadoop version 2.x includes major architectural improvements in terms of resource managing. Version 1.x of Hadoop suffered shortcomings due to an overload of its resource management duties which was handled within MRv1 where job tracker node and task tracker nodes were running the organizational load of MapReduce executions. YARN standing for "Yet Another Resource Negotiator" emerging as an intermediate layer between HDFS and MapReduce taking over some of the load that was previosly carried out by MRv1 also become a gate to other batch, stream, interactive and graph processing engines to leverage HDFS file system (Figure 4). After YARN, MRv1 has become MRv2 isolated from resource managerial duties regarding previous version hence more efficient with processing oriented focus. YARN innovates Hadoop by brining the architectural elements resource manager, which is one node dedicated to track resources across the cluster by availability, and node managers which reside inside each worker node and monitor containers.

Cloud Computing. The commercialization of Cloud Computing in the midst 2000's [7] delivered utilization of storage and computing resources to the end users saving them high investments on hardware technology that is soon going to be obsolete and is expensive to maintain. As the migration to the cloud

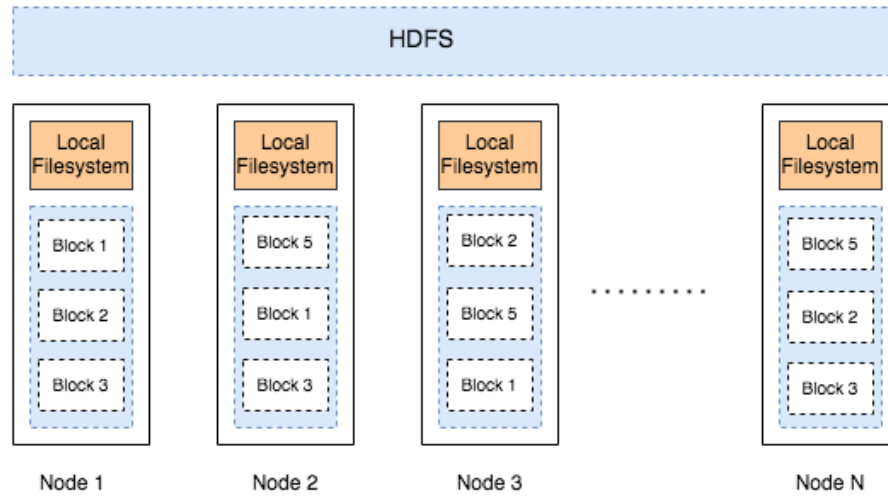


Figure 1: Hadoop Distributed File System

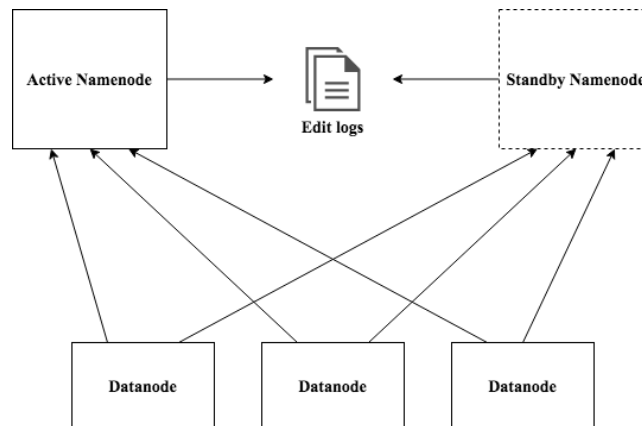


Figure 2: Hadoop High Availability

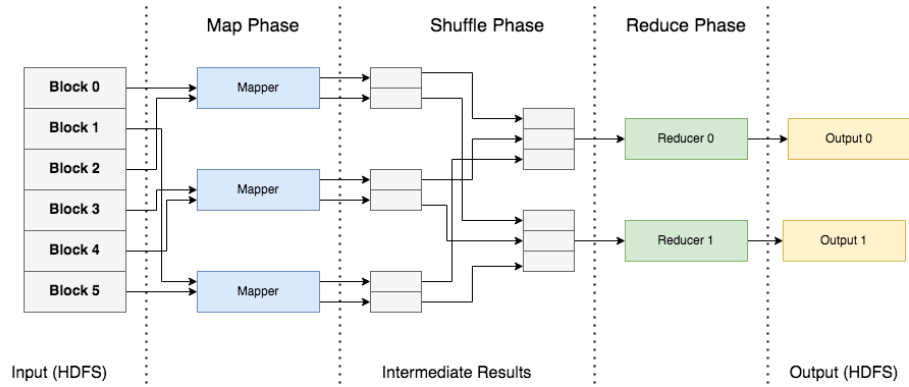


Figure 3: MapReduce execution (recreated from source [6])

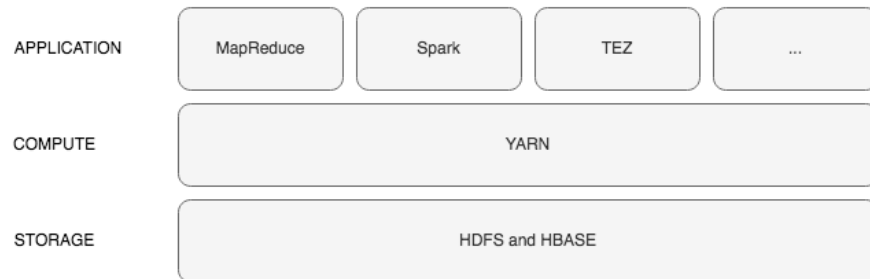


Figure 4: YARN (recreated from source [5])

is an ongoing process, Hadoop also slips out from its residence on on-prem infrastructure to the cloud by being implemented on virtual machine instances provided as IaaS platforms by many providers. The Cloud Service Providers
80 embraced the need of eliminating Hadoop’s complex implementation process on multi-node VMs by providing managed Hadoop systems commercially packaged as PaaS, which are pre-installed and pre-configured Hadoop clusters allowing the installation of tens to hundreds of nodes in a matter of minutes simply by determining some settings like hardware specs and node numbers prior the
85 installation. The Managed Hadoop system is both a blessing and a curse, by leaving the hard implementation part which is not necessarily related with the main analysis objective to a contractor the end user saves time and efforts including a payoff, though: By definition, managed systems are prepackaged solutions provided in black-box nature. CSPs apply behind-the-scenes tweaks in
90 terms of reaching better performance results on selected approaches like memory intensive or compute intensive applications.

GCP Dataproc [8]. Google’s managed Spark and Hadoop solution, namely Dataproc, is a preconfigured Hadoop on PaaS service built upon preinstalled VM instances on Compute Engine [9] which is another service of GCP. An OS spec-
95 trum comprising Debian and/or ubuntu was offered at the date of the study which are proposed as preinstalled images during Dataproc installation process. While Hadoop core elements HDFS, YARN, and MapReduce reside as default frameworks, a variety of elements belonging to Hadoop ecosystem are offered as well. The end user is also offered to leverage Cloud Storage [10], Google’s pro-
100 prietary cloud storage service to keep data in long term where Dataproc cluster is meant to be terminated after usage. By means of Web UI or local CLI via API the end user can access the Dataproc cluster as well as single VM instances within Compute Engine. Google offers a large number of data centers around the globe.

105 *Azure HDInsight* [11]. Azure’s solution to managed Hadoop and Spark platform is a product of the collaboration with Hortonworks bringing Hortonworks

Data Platform (HDP) to a cloud platform [12]. As being so, HDInsight differs in architecture with respect to its conjugates, by not being a Hadoop cluster installed on the cloud VM service layer, namely Azure Virtual Machines, instead
110 it is an HDP platform optimised for the cloud. Another difference is that HDInsight obligates WASB, the Azure blob system as the storage system including an option to leverage Data Lake Storage, excluding Hadoop’s native file system HDFS from the options. At the date of the study Azure was obligating Hadoop to use in in High Availability mode with 2 master nodes hence leaving hardware
115 selection to the end user.

Alibaba Cloud e-MapReduce [13]. Alibaba Cloud’s managed Hadoop service e-MapReduce leveraging Apache Hadoop and Apache Spark is placed as a service layer on its Elastic Compute Service (ECS) [14], a similar approach to the one of GCP’s. Alibaba’s number of provided data center locations beyond Main-
120 land China are not as numerous as its conjugates hence available in the United States, Europe, Middle East and Asia Pacific. Alibaba Cloud’s managed service distincts in that its preinstalled Operating System Aliyun Linux 2 by the date of the study is developed by Alibaba Cloud, a linux distribution based on CentOS, the open source version of RedHat Linux. AliyunOS is claimed to
125 provide a stable and reliable environment functioning deeply optimized with the Alibaba Cloud infrastructure, and be open source on its GitHub repository [15]. e-MapReduce offers a wide scale of machine types with different specifications for specific purposes like CPU intensive or memory intensive tasks. As with GCP, on Alibabacloud e-MapReduce the type of storage whether it shall be
130 HDFS or leverage Alibaba cloud storage can be specified by the end user.

HiBench [16]. Introduced to the community in 2010 [17], the open source HiBench implementation is motivated by the aim providing a comprehensive and representative benchmark for Hadoop helping evaluate different big data frameworks in terms of speed, throughput, and system utilization. At the date of the
135 study current version of HiBench is 7.1, comprising 29 benchmarks categorized in 6 sections (Micro, Machine Learning, SQL, Websearch, Graph, and Stream-

Table 1: HiBench 7.1 - Hadoop-related Workloads

Category	Workload	Dominant Character
Micro	Sort	IO bound
	Terasort	IO bound
	Wordcount	CPU (map), IO (reduce)
	Dfsioe	IO bound
SQL	Scan	IO bound
	Join	IO bound
	Aggregation	IO bound
ML	Bayes	IO bound (CPU bound in map)
	Kmeans	CPU (iterations), IO (clustering)
Websearch	Nutchindexing	IO bound
	Pagerank	CPU bound

ing). Table 1 lists the Hadoop related workloads in HiBench 7.1 [18] executed in the study.

Here is another HiBench citation [19]

2. Related Work

HiBench is a tool to measure a specific systems performing behaviour during execution. The conceptualization of conducting a benchmark may arise from different soils. Based on the conductors motivation; a benchmarks use case could be an inner evaluation of a systems performance before and after some configuration tweaks are set, comparison of rival / complementary systems, or putting CSPs cloud infrastructure services on scale. Following literature has been searched with finding different use cases of HiBench benchmarking suite in mind.

Poggi et al. [20] Characterizing BigBench

Poggi et al. [21] the state of SQL on Hadoop [53]

Samadi et al. [22] conduct an experimental comparison between Spark and Hadoop installed on virtual machines on Amazon EC2 by leveraging nine among

the provided HiBench workloads. Accuracy reasons led the conductors run the workloads three times concluding input data scales of 1, 3, and 5 GB respectively. Based on the outputs comprising duration, throughput, speed up, and CPU/memory consumption, the conclusion draws Spark consuming less CPU and performing better on all workload results over Hadoop.

Ahn et al. [23] put Spark on YARNs performance on test with HiBench in terms of handling a deluge of data generated by IoT devices. The experiment is run on a cluster with one master and 3 worker nodes each node possessing Intel Xeon processor with 20 cores and 128GB main memory meaning 60 cores and 384GB memory in total. HiBenchs workloads Micro (comprising Sort, TeraSort, and Wordcount), SQL (comprising Aggregation, Join, and Scan), and Machine Learning (comprising Bayes, Logistic Regression, Gradient Boosting Tree, Random Forest, and Linear Regression) are leveraged by a chosen data scale of 30 GB. Spark occupies memory during the whole job execution which in result reduces IOs negative impact on processor performance. For optimizing resource usage the conductors modified YARNs minimum memory allocation and Spark executor settings so that the Spark executors overall loads remain below total system memory. Alongside with HiBenchs duration and throughput report, CPU / memory utilization and disk throughput are profiled as well. Finding of this paper points out that Spark guarantees performance when provided with enough memory.

Han et al. [24] study the impact of memory size on big data processing by means of Hadoop and Spark performance comparison leveraging HiBenchs k-Means workload as the only benchmark. For each of the specified memory sizes of 4, 8, and 12 GB, iterating through a data scale of 1 to 8 GB, with 1GB increment inbetween, k-Means benchmark for Hadoop and Spark is executed. The results depict Sparks overperforming Hadoop unless the total input data size is smaller than 33.5% of the total memory size assigned to worker nodes. After reaching that ratio Spark suffers with insufficient memory resources and is led to interoperate with HDFS causing a sharp decrease in its performance and brings Hadoop in throughput and duration performance to the front. The

conductors make a second experiment to find out if Sparks performance can
185 be improved by tweaking the allocation setting for storage memory and shuffle
memory while remaining within the specified memory limitations of 4, 8 and
12 GB. Executing HiBenchs k-means benchmark outputs a report interpreted
by the conductors as Spark show a 5-10%, and 15% maximum improvement in
processing time.

190 Ivanov et al. [25] compare the performances of two enterprise grade applica-
tions, DataStax Enterprise (DSE), a production level implementation of Apache
Cassandra with extended features like in memory computing and advanced secu-
rity to name but two, and Clouderas Distribution of Hadoop (CDH) comprising
core Hadoop elements HDFS and YARN integrated with elements belonging to
195 the Hadoop ecosystem. DSEs HDFS compatible file system CSF lets Hadoop
applications run without any modification. The conductors installed the latest
stable releases of both softwares on equal CPU, memory and network infras-
tructure configuration. For both installations, default system parameters have
been left with their defaults. HiBenchs three chosen workloads (CPU-bound
200 wordcount, IO-bound dfsioe, and mixed HiveBench) are executed three times,
the average values have been taken for representativeness. Several conclusions of
their study proclaim linearly scaling of both systems by the increase of data size,
while CDH outperforms DSE in read intensive workloads, DSE performs better
in write intensive workloads. Leveraging HiBench is where the study differs
205 in approach related to other studies using YCSB benchmark suite. HiBenchs
results confirm the latters output as well.

3. Method

In the study we benchmarked Hadoop on PaaS services of three CSPs in
terms of comparing their performances accompanied by respective resource uti-
210 lization across the worker nodes: GCP Dataproc, Azure HDInsight, and Al-
ibaba Cloud e-MapReduce, each recognized in Gartner’s 2020 Magic Quadrant
for Cloud Infrastructure and Platform Services [26] either in leading or in niche

section.

We constructed the benchmark study in two use cases: Use Case 1 aims
215 to map the overall performance behaviors of respective CSPs within HiBench’s
predefined data scales huge and gigantic. The following benchmarks have been
executed on the clusters of the respective providers: Micro (Sort, Terasort,
Wordcount, and Dfsioe), Sql (Scan, Join, and Aggregation), ML (Bayes and
Kmeans), and Websearch (Pagerank).

220 Use Case 2. We picked one benchmark of IO bound character (Sort), one
benchmark of CPU bound character (Wordcount), and executed these in in-
creasing data scales HiBench’s: Tiny, Small, Large, Huge, and Gigantic. The
largest predefined HiBench data scale namely Bigdata would cross the storage
limits of the installed clusters, hence we left it off the execution.

225 Bound by availability of their respective hardware and software options we
selected by providers’ promise apparently same or close settings. With respect
to our aim benchmarking managed systems as they come out-of-the-box, we put
two basic rules to stick for each provider in order to align initial circumstances
prior to benchmark execution: Among the CSPs, respective data center’s ge-
230 ographic location shall be same or close (1), processor number and memory
capacity (by providers’ promise) shall be same or as close as possible. Thus
Frankfurt is selected as location for all providers data center, within given op-
tions we selected 8 CPUs/64GB RAM for master node and 4 CPU’s/32GB RAM
for each worker node totaling in 12 CPUs and 96GB worker compute power for
235 the cluster. Specified cluster installation options and details are given in Ta-
ble 2. Without applying any performance tweak operation to the respective
preconfigurations we immediately executed several Hadoop benchmarks from
HiBench’s Micro, Sql, ML, and Websearch categories. We only modified default
cluster’s configuration in cases where the benchmark was detained from running
240 which is reported in Discussion section.

During the benchmark execution we collected system utilization records ad-
dressing CPU, Memory, and IO behavior on each worker node of the cluster.
This approach enabled us the visualization of an overview to the system utiliza-

Table 2: Selected configurations on CSPs’ managed Hadoop services

	GCP	Azure	Alibaba Cloud
Service	Dataproc	HDInsight	e-MapReduce
Region	europe-west3-a	Germany West Central	eu-central-1
Location	Frankfurt	Frankfurt	Frankfurt
Image	1.4-ubuntu18	HDI 3.6	EMR-3.32.0
OS	ubuntu18.04	ubuntu 16.04	Aliyun Linux 2
Hadoop v.	2.9	2.7.3	2.8.5
Java	1.8.0_275	1.8.0_275	1.8.0_252
MASTER NODE			
Machine Type	e2-highmem-8	A8m v2	ecs.se1.2xlarge
Processors	8 vCPU	8 cores	8 vCPU
Memory	64 GB RAM	64 GB RAM	64 GB RAM
WORKER NODES			
# Nodes	3	3	3
Machine Type	e2-highmem-4	A4m v2	ecs.se1.xlarge
Processors	4 vCPU	4 cores	4 vCPU
Memory	32 GB RAM	32 GB RAM	32 GB RAM
Storage	HDFS 1000 GB	WASB	HDFS 1000 GB
Replication	2	<i>Azure blob storage</i>	2
Block size	128 MB		128 MB

tions on each worker node within the cluster over the respective benchmark’s
 245 execution time. To provide data locality, if a specific data block to process does
 not reside on the same node where there is an available map slot YARN trans-
 250 fers the respective block to the node with free slot requiring a network activity
 which would have a decreasing impact over the performance. However, due to
 the small cluster size consisting of 3 nodes we excluded network activity track
 with the assumption data locality condition is provided during the executions
 of the benchmark and likely to have least impact on the performance.

4. Results

HiBench’s Hadoop related benchmarks in groups micro (Sort, Terasort, Dfsioe, and Wordcount), sql (Scan, Join, and Aggregation), ml (Bayes and Kmeans),
255 and websearch (Pagerank) have been executed on all three CSPs managed Hadoop services. HiBench 7.1 benchmark execution reports for Use Case 1 are listed in Table 3, execution reports for Use Case 2 are listed in Table 4.

Plots. During benchmark runtime resource utilization on worker nodes have been captured. The plots are suggested to perceived read as follows: Top-left,
260 top-right, and bottom-left plots represent CPU (user%), Memory (%memused), and IO utilization (read transfers and write transfers per second) on each worker node of the respective cluster over time. CPU utilization lines are given in blue tones, Memory utilization lines are given in fuchsia tones, IO-read and IO-write tps’ are represented with orange tones and green tones, respectively.
265 Even though the coloring convention might sound confusing, it gives a clear overview in terms of utilization behavior of the total benchmark process over time. The left hand side x-axis measures CPU/Memory usage in percent, the right hand side x-axis measures IO-read and IO-write transfers per second. The bottom-right plot represents the comparative benchmark performance outputs
270 of the respective CSPs. Duration measure in seconds is expected to perceived as ”lower is better” while Throughput which is the amount of processed data per second in bytes is expected to perceived as ”higher is better”.

For each use case we provided spider charts, a.k.a. radar plots, to give a comparative overview for system utilization vs. benchmark performances of
275 relevant CSPs. In these plots normalized average values of the captured CPU, Memory, and IO data on the worker nodes have been taken and averaged a second time in order to get scalar CPU, Memory and IO values representing the cluster’s average performance. Combining the aforementioned values with normalized Duration and Throughput report values of HiBench resulting the
280 spider plots of the paper. Nevertheless, for the spider plots our recommendation for both perceptions of Throughput and Duration shall be ”Higher is better”.

Table 3: Use Case 1 benchmark outputs

Data Scale: Huge

Benchmark	IDS	Dataproc		HDInsight		e-MapReduce	
		$D_{(s)}$	$T_{(MB/s)}$	$D_{(s)}$	$T_{(MB/s)}$	$D_{(s)}$	$T_{(MB/s)}$
Sort	3.28	70	47.11	131	25.08	111	29.42
Terasort	32.00	667	47.99	858	37.28	1054	30.37
Wordcount	32.85	978	33.60	1470	22.34	889	36.95
Dfsioe-r	26.99	294	91.77	662	40.79	245	110.21
Dfsioe-w	27.16	379	71.73	658	41.30	281	96.49
Scan	2.01	73	27.63	157	12.83	74 (*)	27.19 (*)
Join	1.92	181	10.61	356	5.39	175 (*)	10.95 (*)
Aggregation	0.37	97	3.86	215	1.73	97 (*)	3.85 (*)
Bayes	1.88	2604	0.72	6120	0.31	3017	0.62
Kmeans	20.08	2321	8.65	2313	8.68	2070	9.70
Pagerank	2.99	1544	1.94	3334	0.90	2458	1.22

Data Scale: Gigantic

Benchmark	IDS	Dataproc		HDInsight		e-MapReduce	
		$D_{(s)}$	$T_{(MB/s)}$	$D_{(s)}$	$T_{(MB/s)}$	$D_{(s)}$	$T_{(MB/s)}$
Sort	32.85	715	45.94	787	41.72	896	36.68
Terasort	320.00	9821	32.58	—(**)	—(**)	9660	33.13
Wordcount	328.49	10131	32.42	13596	24.16	8671	37.88
Dfsioe-r	216.03	915	236.11	1886	114.54	660	327.29
Dfsioe-w	217.33	1347	161.39	1914	113.57	1060	205.12
Scan	20.10	457	43.96	514	39.09	407 (*)	49.38 (*)
Join	19.19	595	32.27	761	25.24	594 (*)	32.32 (*)
Aggregation	3.69	523	7.05	594	6.20	565 (*)	6.52 (*)
Bayes	3.77	5350	0.70	12589	0.30	6363	0.60
Kmeans	40.16	4541	8.84	4042	9.94	4034	9.96
Pagerank	19.93	8371	2.38	11779	1.70	13893	1.43

IDS: Input Data Size (GB); $D_{(s)}$: Duration (sec); $T_{(MB/s)}$: Throughput (MB/sec)

(*) Benchmark succeeds after modifying preconfiguration, more on this in Discussion

(**) System failure due to insufficient space on HDFS, more on this in Discussion

Table 4: Use Case 2 benchmark outputs

Benchmark	Dataproc			HDInsight		e-MapReduce	
	IDS	$D_{(s)}$	$T_{(MB/s)}$	$D_{(s)}$	$T_{(MB/s)}$	$D_{(s)}$	$T_{(MB/s)}$
Sort (t)	39.30 KB	36	0.0012	69	0.0006	32	0.0012
Sort (s)	3.28 MB	36	0.09	70	0.0471	31	0.105
Sort (l)	328.50 MB	42	7.86	81	4.07	42	7.74
Sort (h)	3.28 GB	70	47.08	141	23.36	107	30.69
Sort (g)	32.85 GB	694	47.30	699	47.00	883	37.20
Wordcount(t)	38.65 KB	38	0.001	68	0.0006	31	0.0012
Wordcount (s)	348.29 MB	50	6.51	98	3.34	47	7.06
Wordcount (l)	3.28 GB	129	25.45	269	12.20	120	27.27
Wordcount (h)	32.85 GB	952	34.51	1487	22.10	888	37.00
Wordcount (g)	328.49 GB	9749	33.70	13286	24.73	8622	38.10

IDS: Input Data Size; $D_{(s)}$: Duration (sec); $T_{(MB/s)}$: Throughput (MB/sec)
(t): tiny, (s): small, (l): large, (h): huge, (g): gigantic

Even though the Duration is of lower is better nature, for the sake of proper visualization we needed to display the shortest duration as the most visible among other CSPs.

USE CASE 1:

Sort - Huge. Figure 5; CPU utilization in GCP and Alibaba condense around 80% to 98% whereas in Azure the range widens up between 50% to 90%. Memory loads in GCP and Alibaba among the worker nodes display a harmonic behavior between 20% to 40% and 90% to 100% respectively, whereas the memory load in Azure's worker nodes vary between 10% and 50%. In the second half of the benchmark execution IO write transfers in GCP and Alibaba show peaks at about 500 tps where in Azure it is limited with 100 tps. Resulting in GCP carrying out the highest Throughput, thus reaching the shortest Duration of 70 seconds.

Sort - Gigantic. Figure 6; switching the data scale for Sort benchmark to gigantic, GCP's processor load condenses around 80% - 95% where its memory load

Table 5: Use Case 1 Comparative Performances

Data Scale: Huge					
Benchmark	First	Second	-Perf.%	Third	-Perf.%
Sort	GCP	Alibaba	-58.57%	Azure	-87.14%
Terasort	GCP	Azure	-28.64%	Alibaba	-58.02%
Wordcount	Alibaba	GCP	-10.01%	Azure	-65.35%
Dfsioe-r	Alibaba	GCP	-20.00%	Azure	-170.20%
Dfsioe-w	Alibaba	GCP	-34.88%	Azure	-134.16%
Scan	GCP	Alibaba	-1.37%	Azure	-115.07%
Join	Alibaba	GCP	-3.43%	Azure	-103.43%
Aggregation	GCP-Alibaba	—	—	Azure	-121.65%
Bayes	GCP	Alibaba	-15.86%	Azure	-135.02%
Kmeans	Alibaba	Azure	-11.74%	GCP	-12.13%
Pagerank	GCP	Alibaba	-59.20%	Azure	-115.93%
Data Scale: Gigantic					
Benchmark	First	Second	-Perf.%	Third	-Perf.%
Sort	GCP	Azure	-10.07%	Alibaba	-25.31%
Terasort	Alibaba	GCP	-1.67%	Azure	—
Wordcount	Alibaba	GCP	-16.84%	Azure	-56.80%
Dfsioe-r	Alibaba	GCP	-38.64%	Azure	-185.76%
Dfsioe-w	Alibaba	GCP	-27.08%	Azure	-80.57%
Scan	Alibaba	GCP	-12.29%	Azure	-26.29%
Join	Alibaba	GCP	-0.17%	Azure	-28.11%
Aggregation	GCP	Alibaba	-8.03%	Azure	-13.58%
Bayes	GCP	Alibaba	-18.93%	Azure	-135.31%
Kmeans	Alibaba	Azure	-0.20%	GCP	-12.57%
Pagerank	GCP	Azure	-40.71%	Alibaba	-65.97%

Table 6: Use Case 2 Comparative Performances

Benchmark	First	Second	-Perf.%	Third	-Perf.%
Sort (t)	Alibaba	GCP	-12.50%	Azure	-115.63%
Sort (s)	Alibaba	GCP	-16.13%	Azure	-125.81%
Sort (l)	GCP-Alibaba	—	—	Azure	-92.86%
Sort (h)	GCP	Alibaba	-52.86%	Azure	-101.43%
Sort (g)	GCP	Azure	-0.72%	Alibaba	-27.23%
Wordcount (t)	Alibaba	GCP	-22.58%	Azure	-119.35%
Wordcount (s)	Alibaba	GCP	-6.38%	Azure	-108.51%
Wordcount (l)	Alibaba	GCP	-7.50%	Azure	-124.17%
Wordcount (h)	Alibaba	GCP	-7.21%	Azure	-67.45%
Wordcount (g)	Alibaba	GCP	-13.07%	Azure	-54.09%

(t): tiny, (s): small, (l): large, (h): huge, (g): gigantic

rises to range 80% - 100%; IO write transfers behave at about 500 tps where
 IO reads reach 1000 tps in the second half of the benchmark process. Azure's
 processor and memory performances depict a looser behavior not utilizing the
 maximum potential whereas IO-read and IO-write tps' reach their maximum at
 200 and 350, respectively. Alibaba's resource utilization depicts a high memory
 load of 90% - 100% dropping to 70% and about 83% on partial nodes in the
 second half. As so with the processor load behaving between 80% and 100% in
 the first half dropping to about 50% in the second half where IO write reaches
 peak at 800 tps. Resulting in GCP embarking highest and Azure the second
 highest Throughput, respective Durations output in 715 and 758 seconds.

Terasort - Huge. Figure 7; GCP depicts a high utilization on processors at a
 range of 80% - 100%, memory moving from 80% to 100%, and IO behavior 500
 tps in overall and peaking at 1000 tps in IO-read resulting in highest Throughput
 in 667 seconds response time among other CSPs. Azure's processor and memory
 utilization fluctuate in overall where memory performance incrementally reach
 100% in one node, a stable IO-write in overall process at about 70 tps peaking
 at 250 tps reaches the second highest Throughput in 858 seconds. Alibaba with

high memory and processor utilization, and varying IO tps's in overall process
315 falls back in embarking Throughput and resulting in 1054 seconds response time.

Terasort - Gigantic. Figure 8; switching the scale to gigantic causes dramatic changes on resource utilization on all CSPs. During all benchmark process GCP depicts very condensed high utilization on processor at a broad range of 30% to about 95%, memory consumption between 95% and 100% and IO
320 read transfer varying between 800 to 1100 tps. Azure on the other side, fails to complete the benchmark on 3 attempts; suffering from YARN insufficient HDFS allocation requested for bringing the job further. IO scores drop to null and task raises failure. Alibaba Cloud's resource utilization goes within maximum levels where processor utilization depicts a consumption of 80% to 90% in the overall,
325 memory utilization at it highest during all process, IO reads and writes moving along the 600 tps' and peaking around 1600 tps. With a slicely higher value in Throughput, Alibaba performs best with 9660 seconds, followed by GCP with 9821 seconds. Azure disqualifies this session.

Wordcount - Huge. Figure 9; GCP performing processor utilization of 75% to
330 100% with a memory consumption between 80% to 95% where IO transfers move along 60 tps peaking at about 150 tps reaches second highest Throughput with 978 seconds response time. Azure's processor utilization moving along 70% to about 100% where memory depicts 30% to 50% utilization and lower IO transfers peaking at about 60 tps reaches the lowest throughput hence the
335 longest duration of 1470 seconds. Alibaba depicting high processor and memory load moving in ranges 70% to 100% and 80% to slightly over 90% with IO transfers moving along 200 tps peaking at 300tps to 350tps reaches the highest Throughput hence shortest execution time of 889 seconds.

Wordcount - Gigantic. Figure 10; GCP utilizing processor and memory sources
340 at their maximum during the overall process, for CPU within 60% up to pushing 100%, memory utilization varying between 60% to 100%, and IO-read and IO-write transfers moving along 90 tps and 250 tps respectively, reaches second

highest Throughput load resulting in 10131 seconds response time. Azure shows a dens processor utilization varying between 40% and close to 100%, memory consumption is somehow oppressed to stay within range 20% up to 50%, IO behavior reacts seldomly over 60 tps resulting in the lowest Throughput and longest Duration of 13596 seconds. Alibaba depicting a high processor utilization within range 85%-100%, a relatively more stable memory consumption in range 85% to 100% and IO transfers mostly about 300 tps thus performing highest Throughput hence shortest response time of 8671 seconds.

Dfsioe-read - Huge. Figure 11; GCP's processor utilization moving roughly between 80% to 95% accompanied by memory utilization within range slightly below and over 70%, IO transfers peaking at 400 tps mostly go along 100 tps and 250 tps performs second highest Throughput resulting in 294 seconds of Duration. Azure's processor utilization is rather limited to between 60% and 90% whereas the memory utilization behaves between 10% to 50%, generally low IO transfers with two markable IO-read and IO-write condensces reaching peaks at 2000 tps and 1500 tps respectively results in the lowest Throughput and 662 seconds of response time. Alibaba with 245 seconds Duration displays a similar resoure utilization pattern with GCP where it differentiates in memory utilization moving around 90%.

Dfsioe-read - Gigantic. Figure 12; processor utilization in GCP cloud concentrates within range of 80% to 90%, memory utilization moves around 95% to 100% whereas IO read transfers moving along about 500 tps brings its performance to second highest Throughput with 915 seconds Duration. Azure displays a performance within a range of 55% up to 90% in CPU utilization, 20% to 55% in memory utilization, and IO-write activity moving along 70 tps, IO-read showing peak at about 470 tps reaching the lowest performance in Throughput outputting 1886 seconds response time. Alibaba system utilization moves along 80%-90% range for processing performance, 90%-100% for memory load and a behavior up to 1700 tps IO-write placing its performance to the highest Throughput with 660 seconds duration.

Dfsioe-write - Huge. Figure 13; in GCP processor utilization moves between 70% and 90%, memory load increments from 20% up to 70% whereas IO-write transfers move along 500 tps resulting in second shortest response time 379 seconds. Azure's CPU utilization around 60% to 90%, memory load moves between 20% and 40% whereas IO-write transfers vary about 20 tps to 80 tp peaking at about 120 tps by a Duration output of 658 seconds. Alibaba with processor utilization 80% and 95%, incrementing memory load from about 40% up to 90%-100%, and IO-write transfers at about 100 tps to 600 tps peaking at about 800 tps performs the heaviest Throughput thus reaching the shortest response time with 281 seconds.

Dfsioe-write - Gigantic. Figure 14; GCP processor utilization moves along a broad range hitting up to 90% load, memory load quickly rises to 95%-100% range and keeps its position in overall execution, IO-writes move along 500 tps to 600 tps placing GCP to the second best performance with 1347 seconds response time. Azure shows a CPU utilization in a range 60% up to 90% until it drops to about 25% in the 2/3th of the overall process, memory load moves along lower 20% to upper 50% until its sudden drop to upper 15% simultaneously with CPU utilization where IO-write transfers peak at about 120 tps resulting in 1914 seconds response time. Alibaba performing the shortest response time with 1060 seconds displays a similar memory and processor utilization behaviour with GCP where it differs in IO-write transfers peaking at about 900 tps.

Scan - Huge. Figure 15; GCP processor utilization condenses between about 80% and 90% during benchmark execution, memory load depicts movement between 20% and 40% accompanied by an IO-write transfer reaching over 500 tps which brings GCP to the leading performance within this benchmark with 73 seconds. On Azure's side processor utilization moves along 80%-90% range whereas memory load increments from the 15% to the 30%, relatively low IO-write transfer peaking at about 85 tps results in lowest Throughput and 157 seconds Duration. Alibaba follows a similar pattern like GCP in resource utilization with about 90% CPU load, 15% to below 40% memory utilization

but a less dense IO-write transfer peaking at about 550 tps and so reaching a close Throughput to GCP resulting in 74 seconds response time.

405 *Scan - Gigantic.* Figure 16; GCP's CPU utilization varying between 50% and 90% with a convergence at about 80%, memory load showing similar range with utilization reaching up 100% and IO-write transfers moving between 500 tps and 600 tps results in second highest Throughput by a 457 seconds response time. Azure's CPU utilization condenses between 80% and 90% and memory
410 load incrementing two times from 20% to 40%, IO-write transfer moving along 50 tps to 60 tps peaking at about 200 tps results in 514 seconds Duration. Alibaba's processor utilization moves in a relatively higher range between 80% and 95%, an incrementing memory load range from 40%-50% to 80%-100% during runtime, and IO-write transfers peaking at about over 500 tps results in
415 highest Throughput thus shortest Duration of 407 seconds.

Join - Huge. Figure 17; GCP processor load moves mainly between 80% and over 90%, memory utilization moves along about 20% and 30%, IO-writes behaving at about 150 tps brings outputs 181 seconds response time. Azures resource utilization shows range between 80% and 90% for CPU, varying mem-
420 ory loads moving along below 20% and below 60%, IO-write transfers behave below 60 tps with a peak of 100 tps resulting in a lower throughput and response time of 356 seconds. Resource utilization at Alibaba behaves similar to GCP in processor and memory load whereas IO-write peaks at about 260 tps, reaching highest Throughput and a Duration of 175 seconds.

425 *Join - Gigantic.* Figure 18; CPU load in GCP moves along 70% to 90% in overall whereas RAM utilization among worker nodes takes a path within 50% to 90% range, IO-write transfers observed in behaviour about 100 tps resulting in a Throughput outputting 595 seconds of response time. Azure's resource utilization dynamics show a utilization of 80% to 90% with decreasing utilization
430 on some worker nodes in later stage, memory utilization moving along the range 20% up to 60% and IO-write transfer observations peaking at 120 tps provide

the lowest Throughput hence the longest response time of 761 seconds. Alibaba performing a processor utilization of about below 90% to below 100%, memory load behavior moving along 50% to below 70%, and IO-write observation
435 peaking at 350 tps performs a slightly higher Throughput than GCP hence slightly shorter Duration of 594 seconds.

Aggregation - Huge. Figure 19; GCP processor utilization moving around 80% to about 95%, memory load behaving between 15% to 40%, and IO-write transfer observed at about 150 tps results in 97 seconds Duration. Azure's CPU
440 utilization behaves between 80% and 90% whereas memory load moves along from 10% to varying utilizations among worker nodes up to 30% and below 60%, and IO-write transfers peaking at about 180 tps performs a relatively small Throughput resulting in 215 seconds completion time. Alibaba resource utilization moving along between 80% and 90% for CPU, 10% to 40% in mem-
445 ort, and IO-write transfer average of 26 tps show close performance to GCP resulting in same completion time of 97 seconds.

Aggregation - Gigantic. Figure 20; processor utilization moving between 80% and 90% in overall execution, memory load between 50% and 70% peaking to below 100%, IO-write transfers reaching a dense behavior at the end reaching
450 500 tps results in highest Throughput and 523 seconds response time. Azure displays a CPU utilization of below 80% to above 90%, mild RAM utilization moving along 30% to 60%, IO-write transfers behaving about 100 tps result in a relative low Throughput and 594 seconds of Duration to complete. Alibaba's processor load moves along 80% to below 100%, IO-write transfer behavior
455 scaling from about 50 tps to 400 tps results in 565 seconds response time.

Bayes - Huge. Figure 21; GCP in processor behavior moving along about 70% to below 100%, memory load at about 30% with a short peak to below 100% at the first half, an IO-write transfer behavior about 500 tps resulting in highest Throughput thus fastest response time of 2604 seconds. Azure showing a
460 splitted behavior in CPU utilization among worker nodes where one load moves

along about 80% to lower 100% and one node depict a movement along minimum to about 20%, memory loads also varying from each other by about 20% load difference at the beginning however meeting the similar range starting at the second half, IO observations depict a low range with exception at the high
 465 IO-write transfer at about 3200 tps at the initial state of the execution result in a relatively low Throughput hence longest Duration of 6120 seconds. Alibaba CPU utilization shows varieties among worker nodes in load ranges, RAM utilization condenses at higher levels over 80%, IO transfers moving along 100tps to observable 450 tps resulting in 3017 seconds response time.

470 *Bayes - Gigantic*. Figure 22; GCP's processor utilization moving along 70% to below 100%, accompanied by memory load incrementing from 40% to 70% becoming stable at 50%, and IO-write transfers moving along 500 tps results in the highest Throughput and 5350 seconds of response time. Azure's resource utilization shows distinction in CPU and memory among worker nodes while two
 475 nodes depict higher utilization at about 70% to below 100% for processing and 40%-50% for memory, one node staying at lower levels for respective resources, the observed IO-write transfer move along 100 tps where IO-read transfer peaks at about 650 tps resulting in a lowest performance of 12589 seconds Duration. Alibaba depicting different utilization ranges among worker nodes for processing,
 480 ing, the memory consumption follows a more balanced load among worker nodes with IO-write transfer moving along 200 tps a relatively lower Throughput hence relatively longer Duration of 6363 seconds, with respect to GCP.

Kmeans - Huge. Figure 23;

Kmeans - Gigantic. Figure 24;

485 *Pagerank - Huge*. Figure 25;

Pagerank - Gigantic. Figure 26;

Overview to benchmarks executed in data scale Huge/. Figure 27;

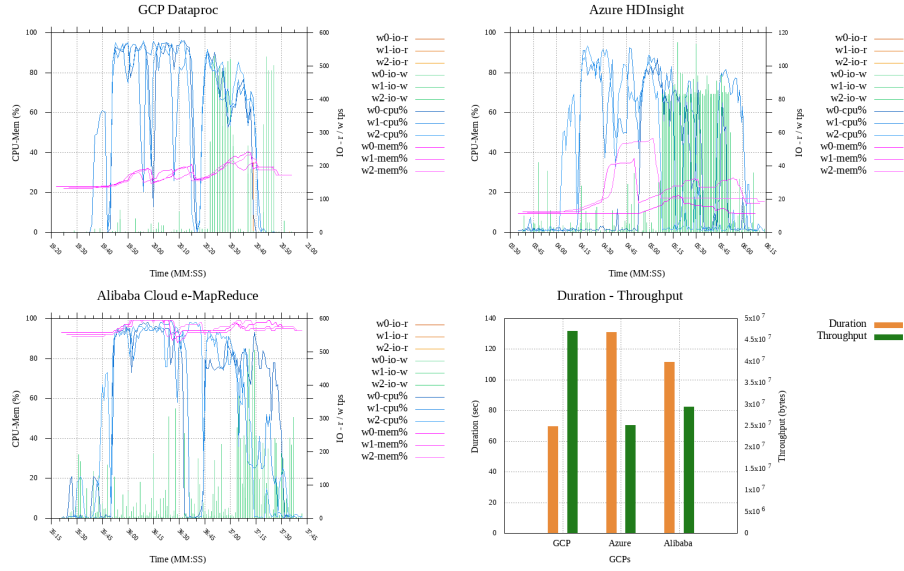


Figure 5: UC1 - Sort (Huge; 3.2 GB)

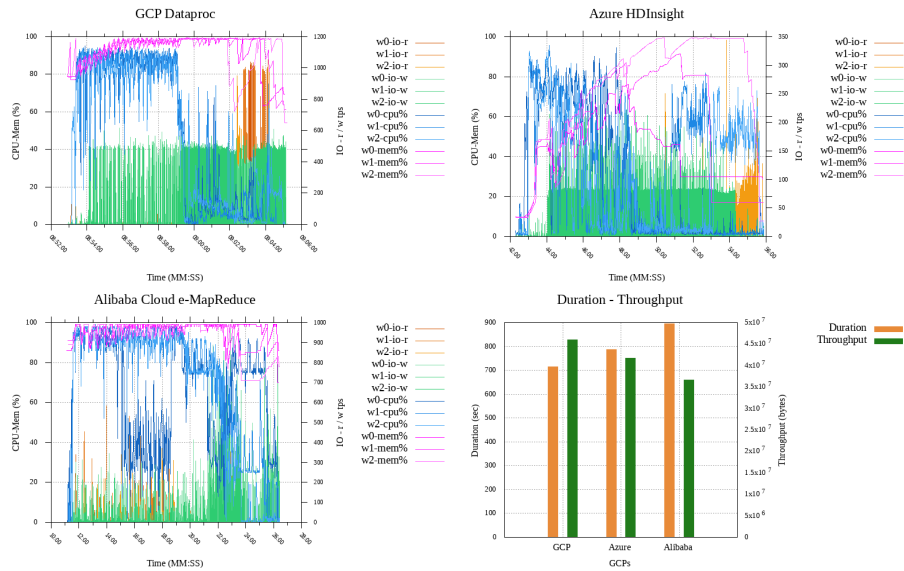


Figure 6: UC1 - Sort (Gigantic; 32 GB)

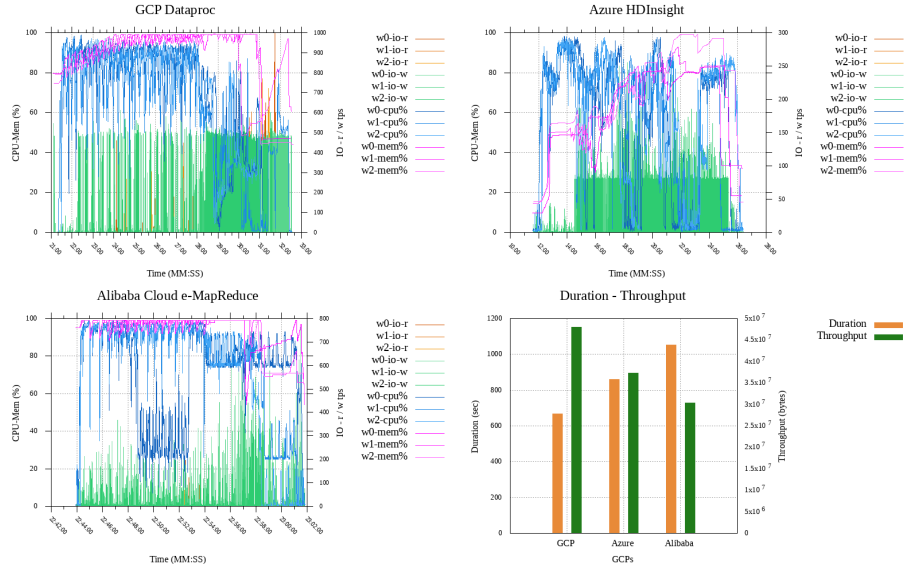


Figure 7: UC1 - Terasort (Huge; 320 MB)

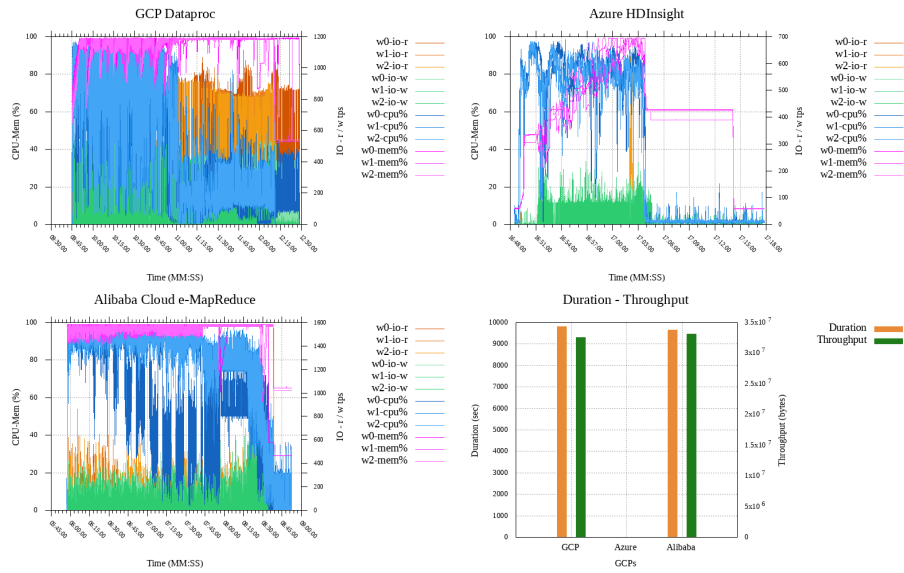


Figure 8: UC1 - Terasort (Gigantic; 3.2 GB)

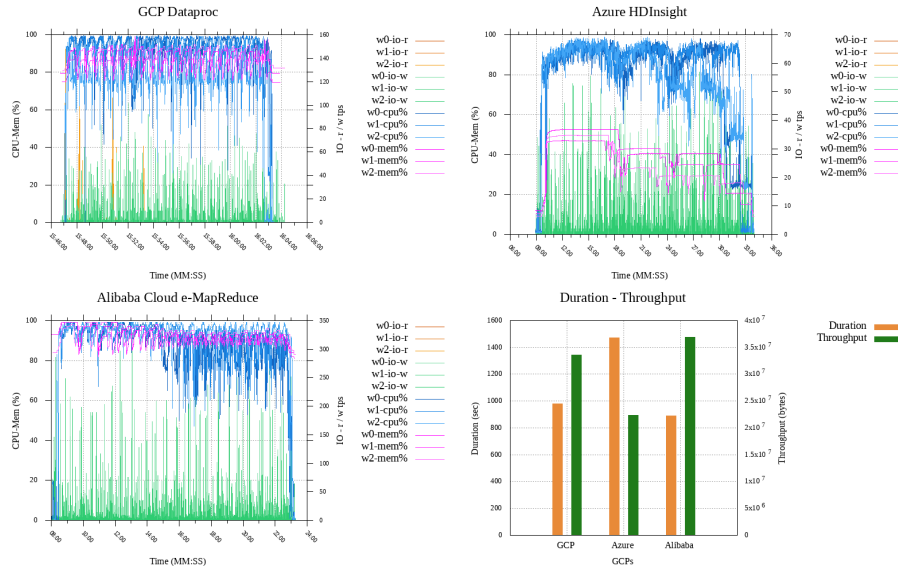


Figure 9: UC1 - Wordcount (Huge; 32 GB)

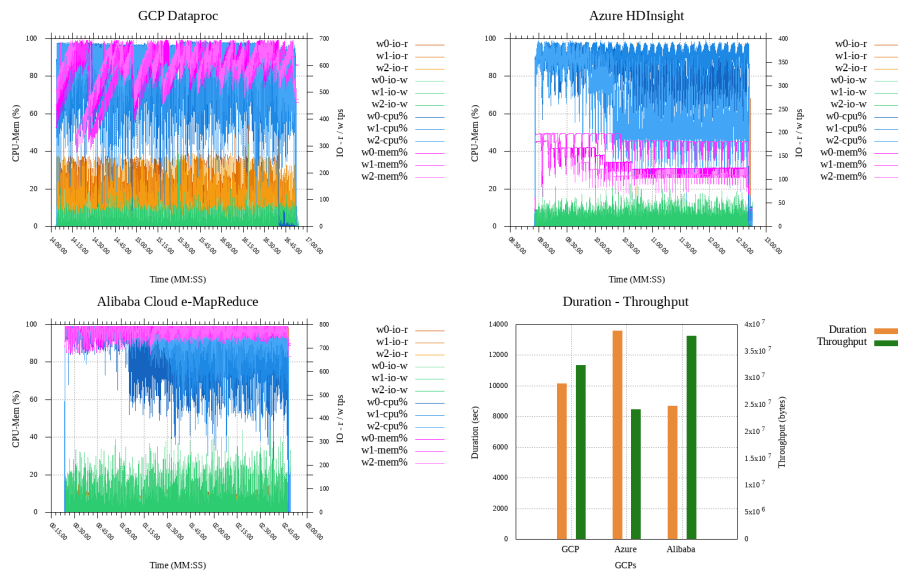


Figure 10: UC1 - Wordcount (Gigantic; 320 GB)

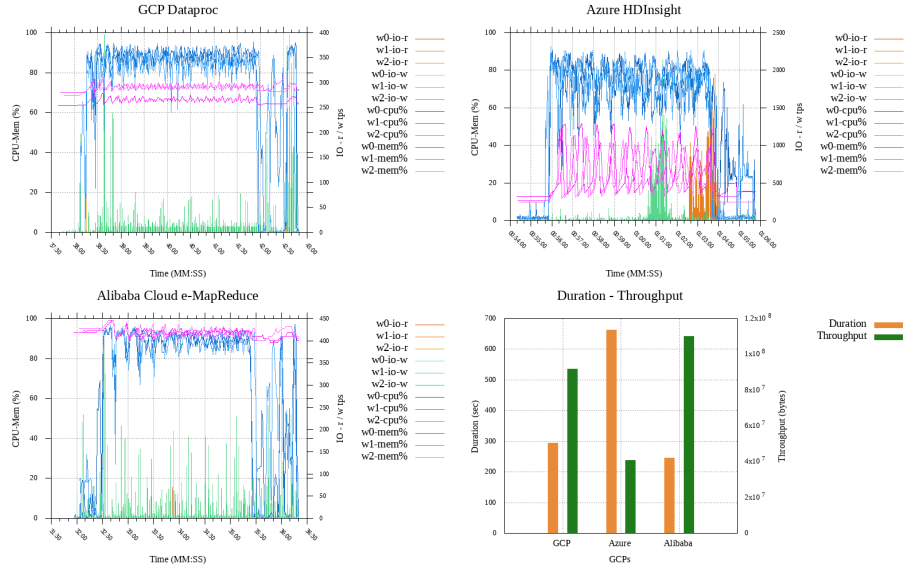


Figure 11: UC1 - Dfsioe-read (Huge; No of Files: 256, File size: 100 MB)

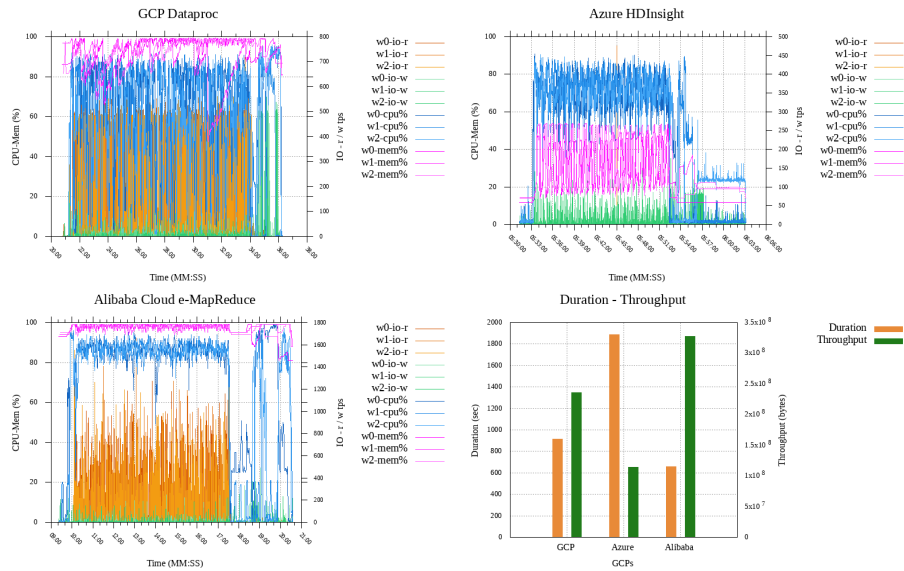


Figure 12: UC1 - Dfsioe-read (Gigantic; No of Files: 512, File size: 400 MB)

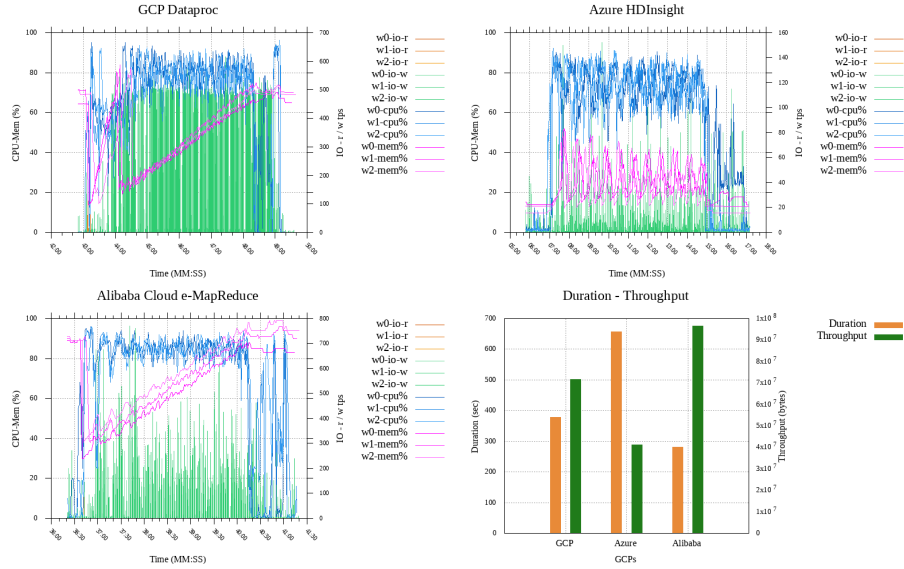


Figure 13: UC1 - Dfsioe-write (Huge; No of Files: 256, File size: 100 MB)

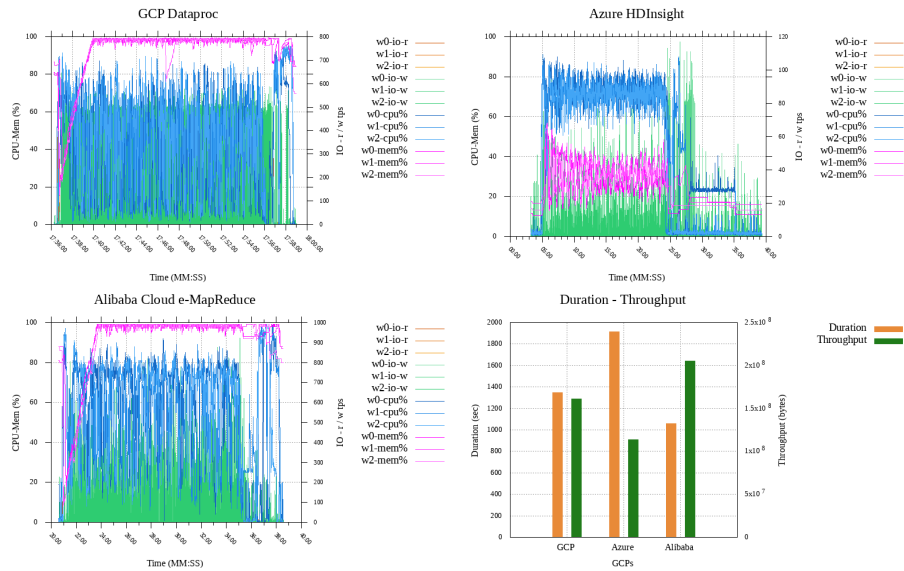


Figure 14: UC1 - Dfsioe-write (Gigantic; No of Files: 512, File size: 400 MB)

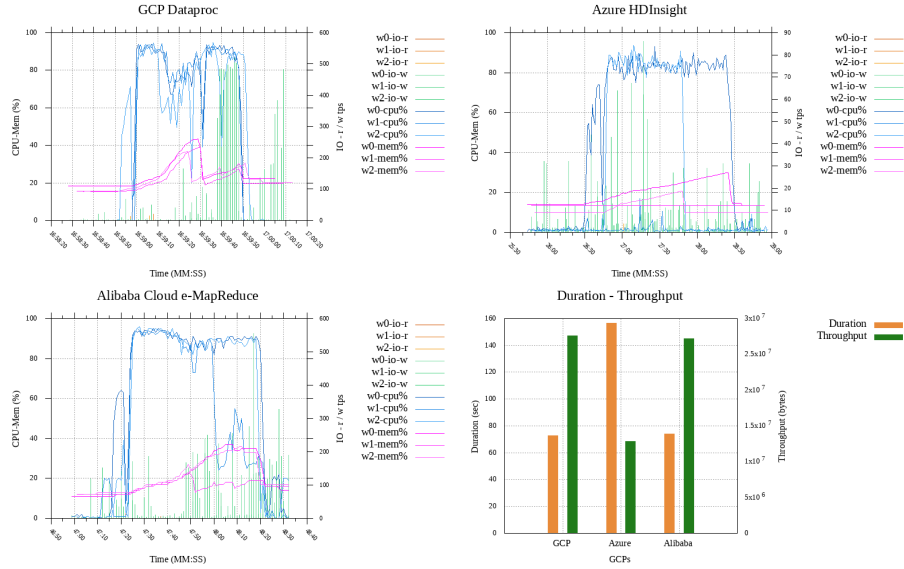


Figure 15: UC1 - Scan (Huge; USERVISITS: 10,000,000 PAGES: 1,200,000)

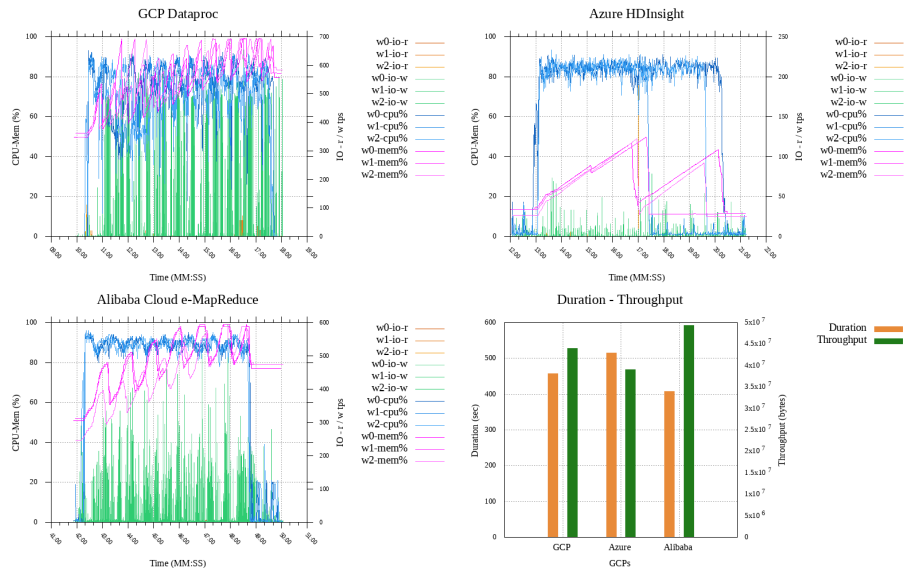


Figure 16: UC1 - Scan (Gigantic; USERVISITS: 100,000,000 PAGES: 12,000,000)

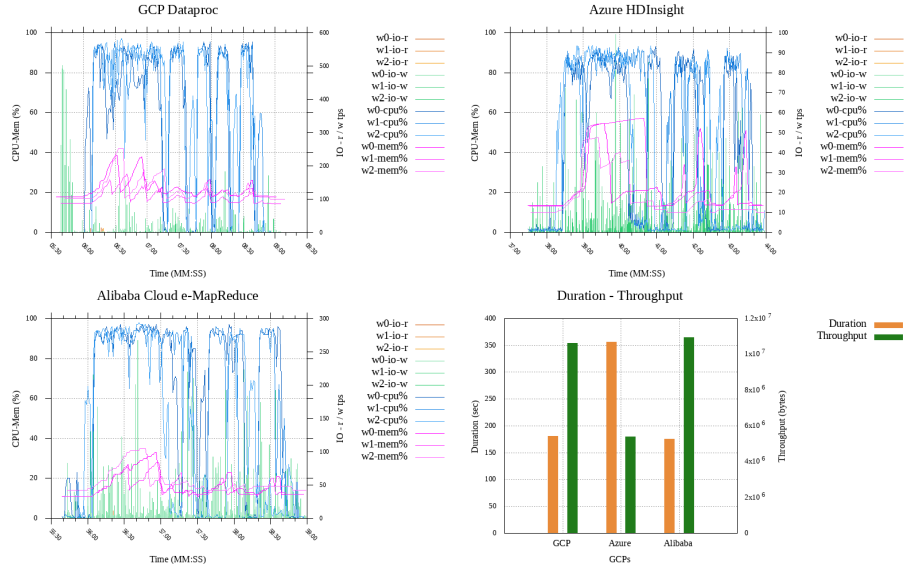


Figure 17: UC1 - Join (Huge; USERVISITS: 10,000,000 PAGES: 1,200,000)

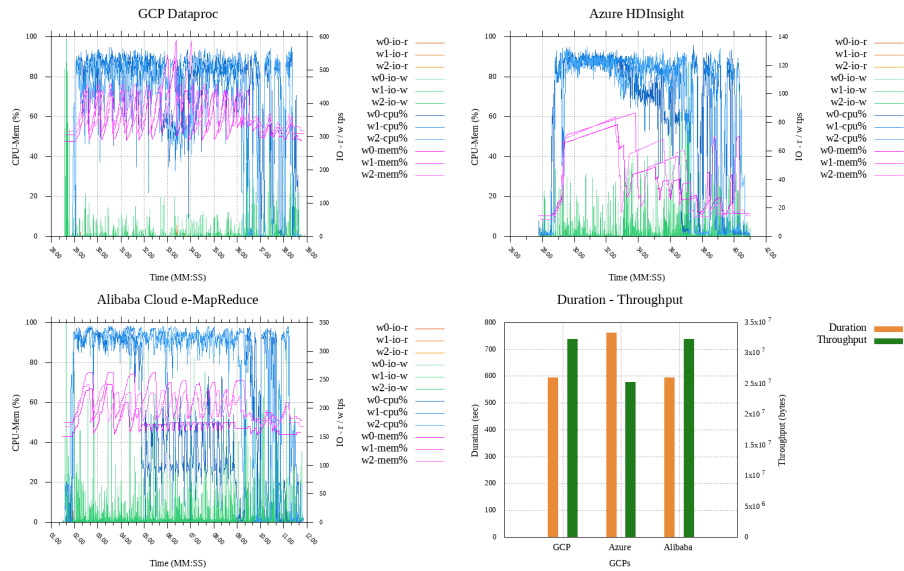


Figure 18: UC1 - Join (Gigantic; USERVISITS: 100,000,000 PAGES: 12,000,000)

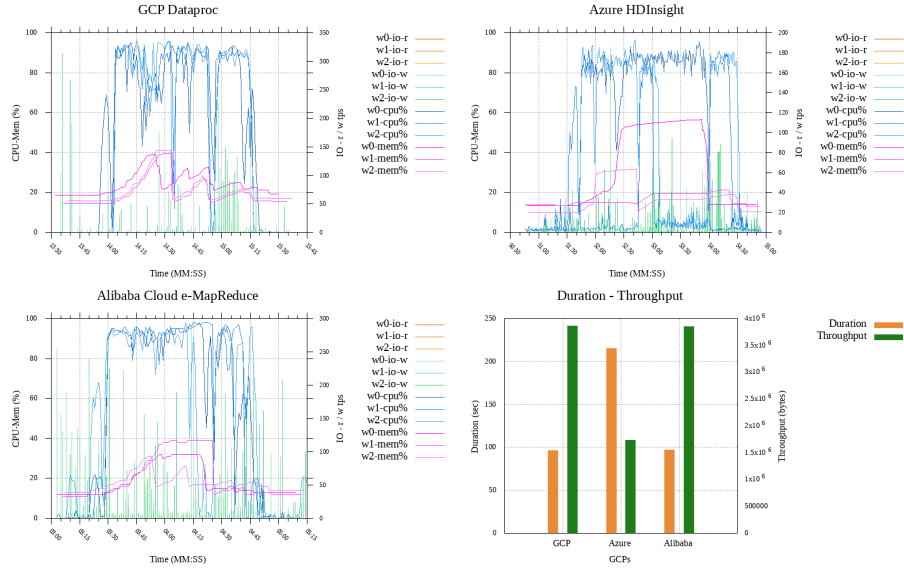


Figure 19: UC1 - Aggregation (Huge; USERVISITS: 10,000,000 PAGES: 1,200,000)

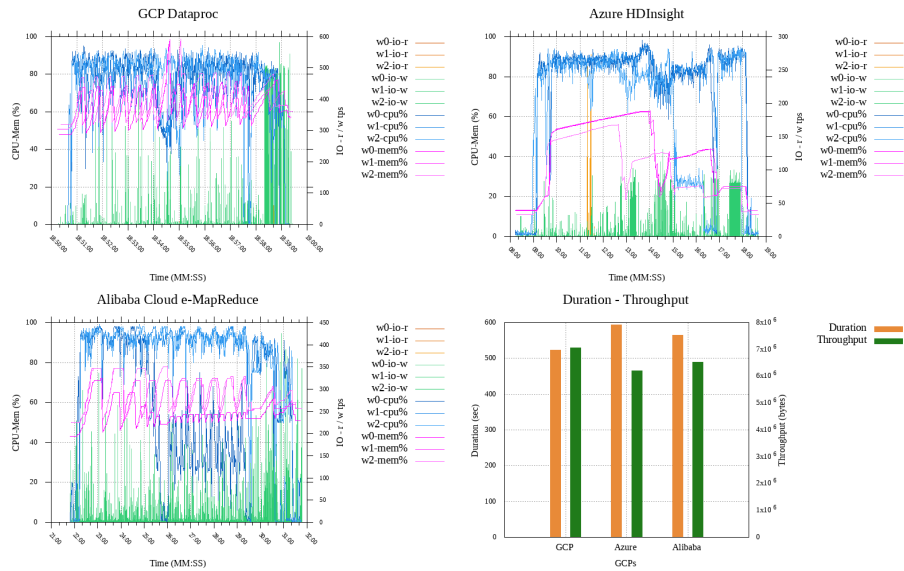


Figure 20: UC1 - Aggregation (Gigantic; USERVISITS: 100,000,000 PAGES: 12,000,000)

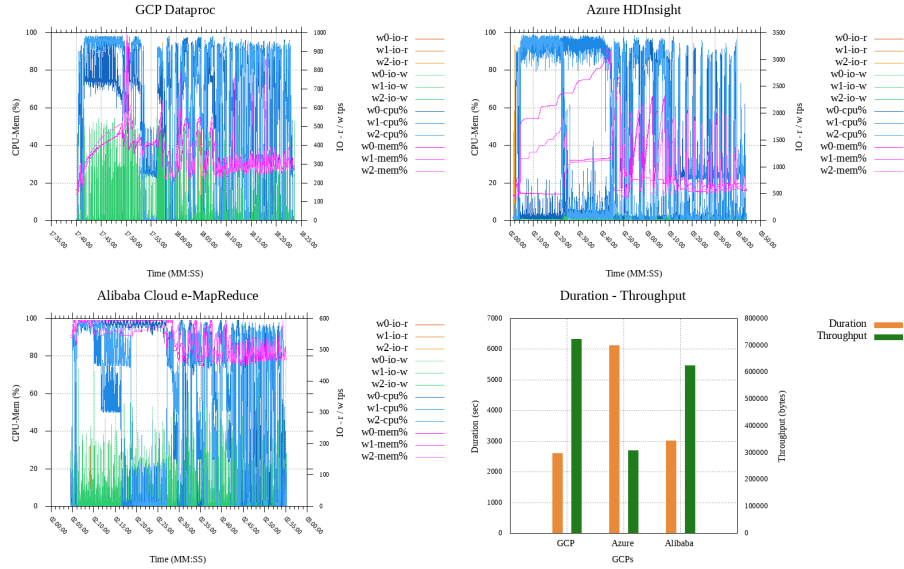


Figure 21: UC1 - Bayes (Huge; PAGES: 500,000 CLASSES: 100 NGRAMS: 2)

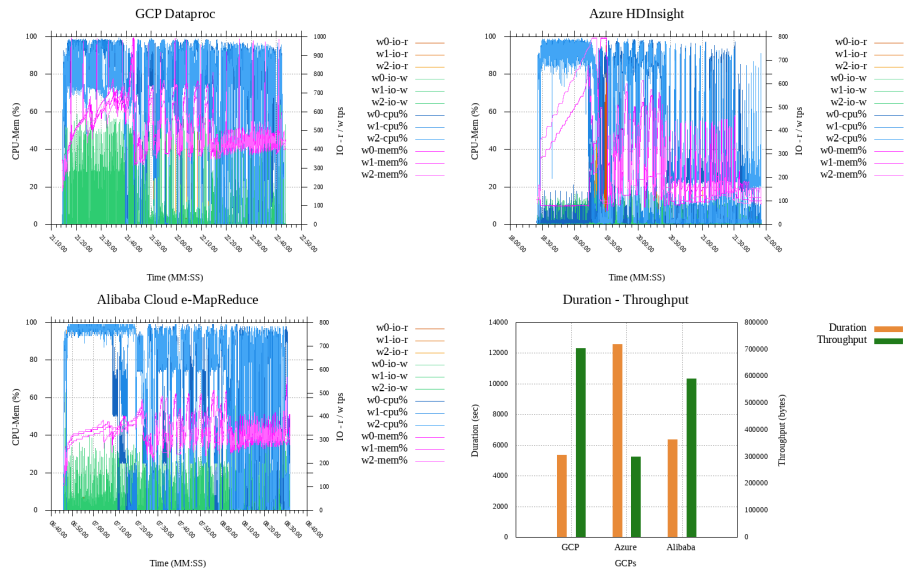


Figure 22: UC1 - Bayes (Gigantic; PAGES: 1,000,000 CLASSES: 100 NGRAMS: 2)

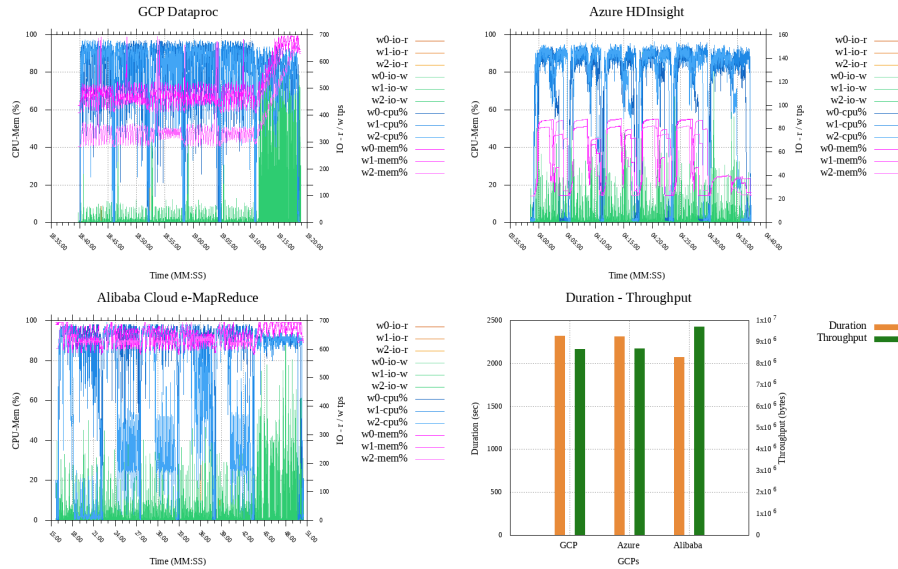


Figure 23: UC1 - Kmeans (Huge; CLUSTERS: 5 DIMENSIONS: 20 SAMPLES: 100,000,000 SAMP PER INPUT: 20,000,000 MAX IT: 5 K: 10 CONVERGEDIST: 0.5)

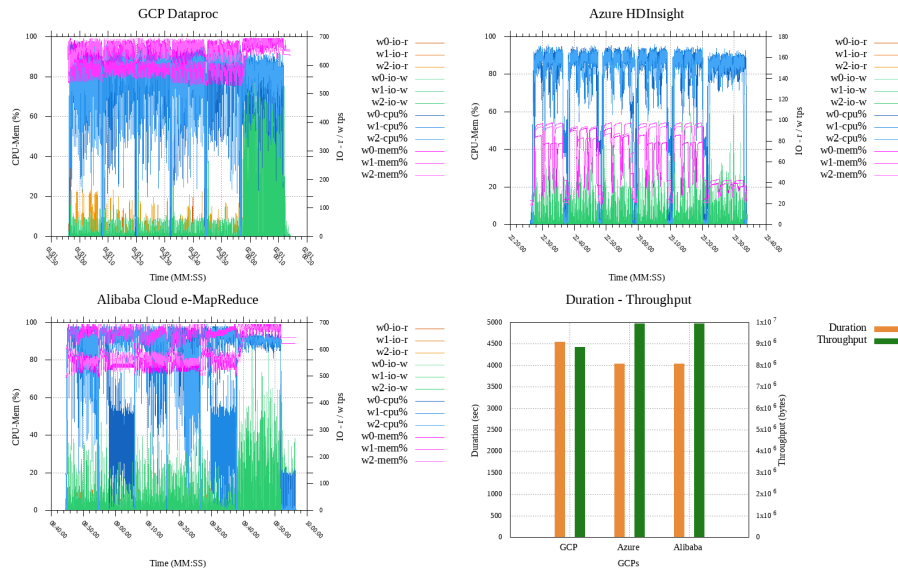


Figure 24: UC1 - Kmeans (Gigantic; CLUSTERS: 5 DIMENSIONS: 20 SAMPLES: 200,000,000 SAMP PER INPUT: 40,000,000 MAX IT: 5 K: 10 CONVERGEDIST: 0.5)

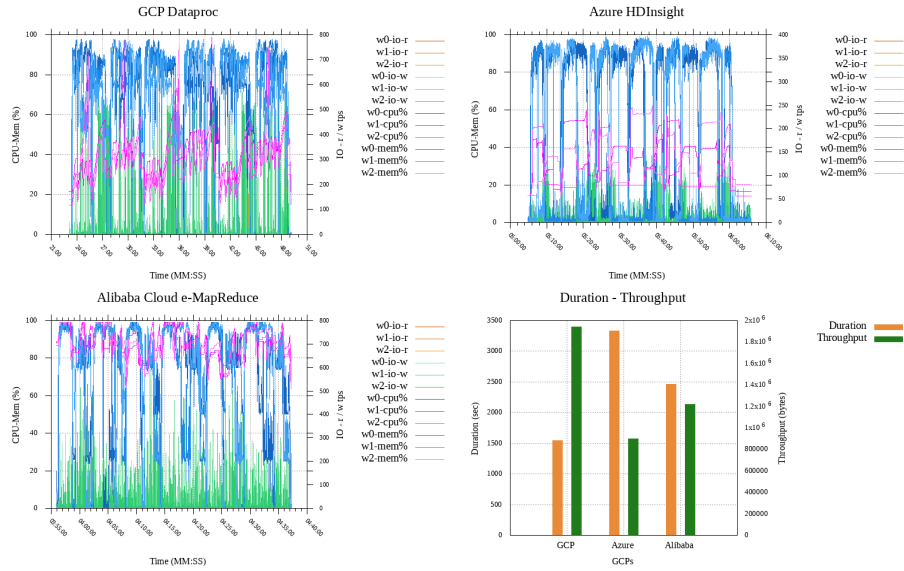


Figure 25: UC1 - Pagerank (Huge; PAGES: 5,000,000 NUM ITERATIONS: 3 BLOCK: 0 BLOCK WIDTH: 16)

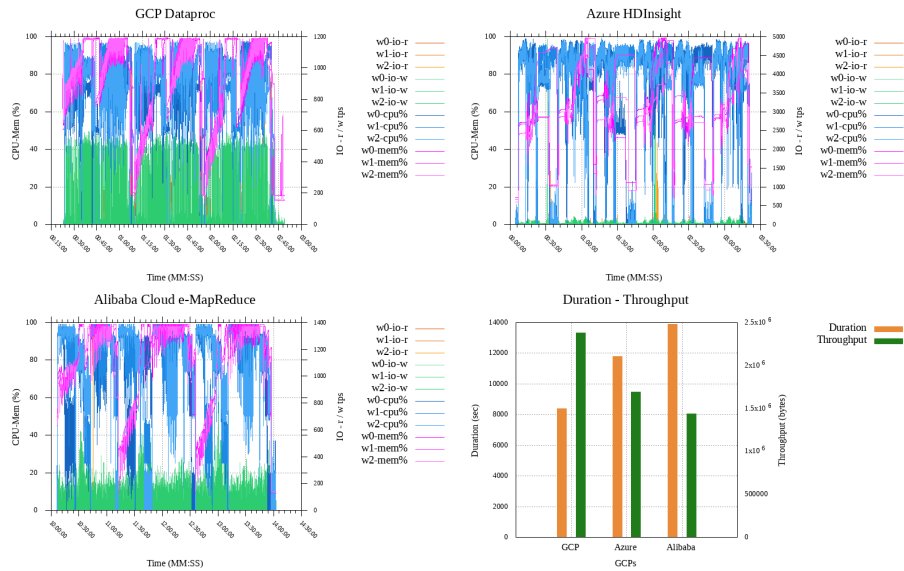


Figure 26: UC1 - Pagerank (Gigantic; PAGES: 30,000,000 NUM ITERATIONS: 3 BLOCK: 0 BLOCK WIDTH: 16)

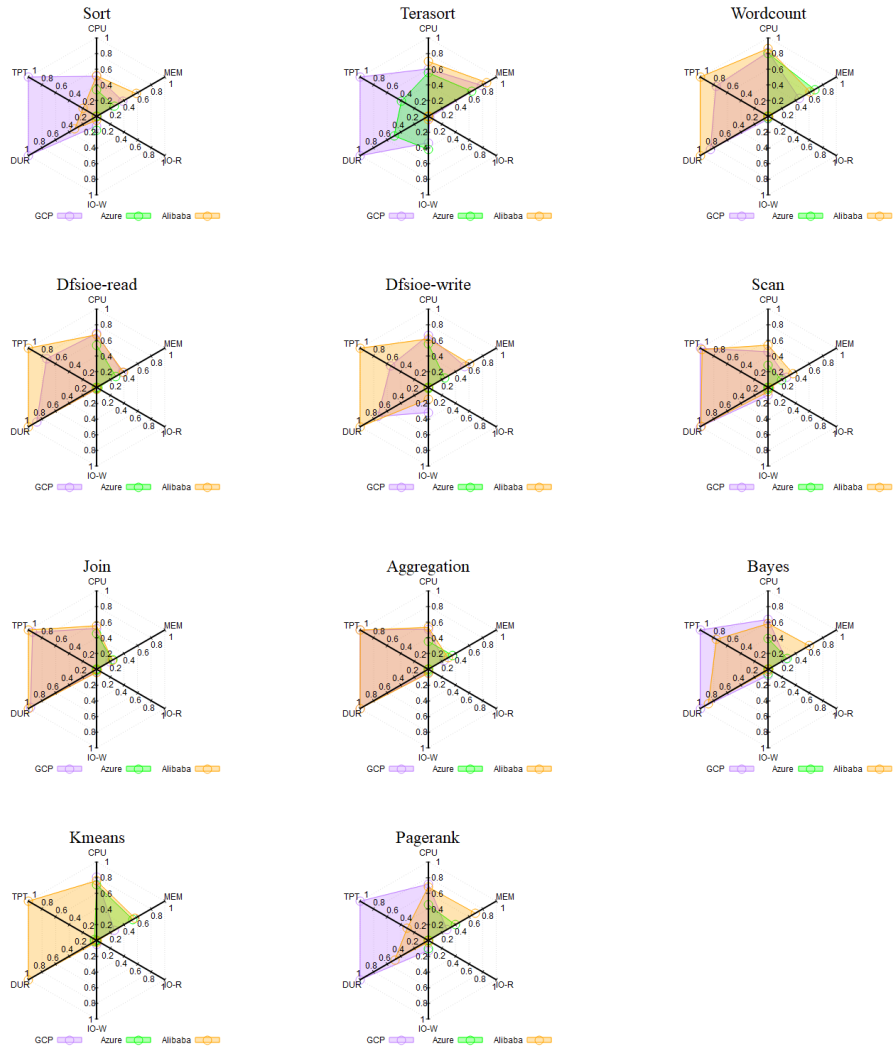


Figure 27: Use Case 1 - Hadoop benchmark comparisons in huge data scale

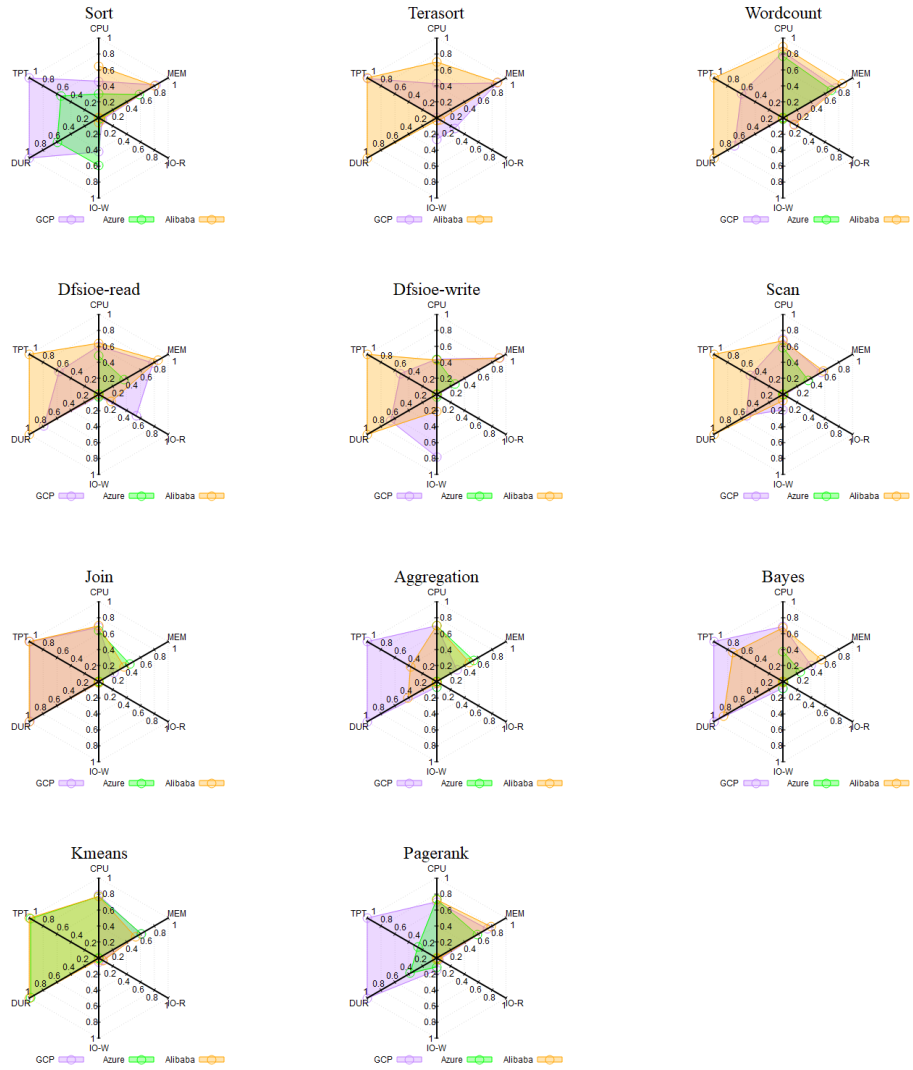


Figure 28: Use Case 1 - Hadoop benchmark comparisons in gigantic data scale

Overview to benchmarks executed in data scale Gigantic. Figure 28;

USE CASE 2:

490 *Sort.* Figure 29;

Sort - Small. Figure 30;

Sort - Large. Figure 31;

Sort - Huge. Figure 32;

Sort - Gigantic. Figure 33;

495 *Wordcount.* Figure 34;

Wordcount - Small. Figure 35;

Wordcount - Large. Figure 36;

Wordcount - Huge. Figure 37;

Wordcount - Gigantic. Figure 38;

500 Wordcount results in scale

Overview of Sort benchmark varying data scales. Figure 39;

Overview of Wordcount benchmark varying data scales. Figure 40;

5. Discussion

Limitations, workarounds, failures. One major limitation of the study has been
505 the high-cost of benchmark executions' total running time charged in a pay-per-
use manner which led the conductors keeping benchmarks at only one successful
execution for each workload. This is not fitting the best practice, though, where
a benchmark would have been executed thrice for each workload and the average
performance results were taken. Following the best practise would have provided
510 more stable outcomes especially in cases where there occurs very tiny difference

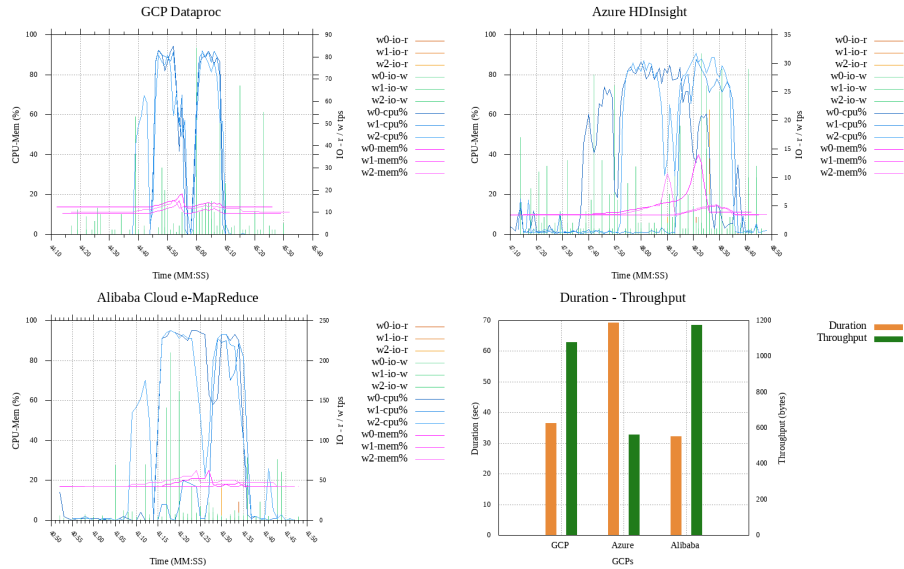


Figure 29: UC2 - Sort (Tiny; 32 KB)

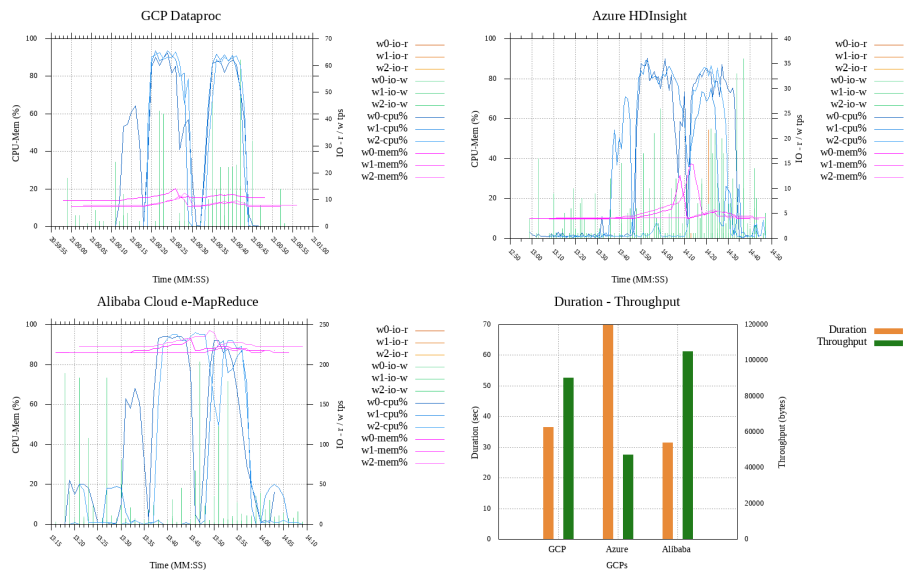


Figure 30: UC2 - Sort (Small; 3.2 MB)

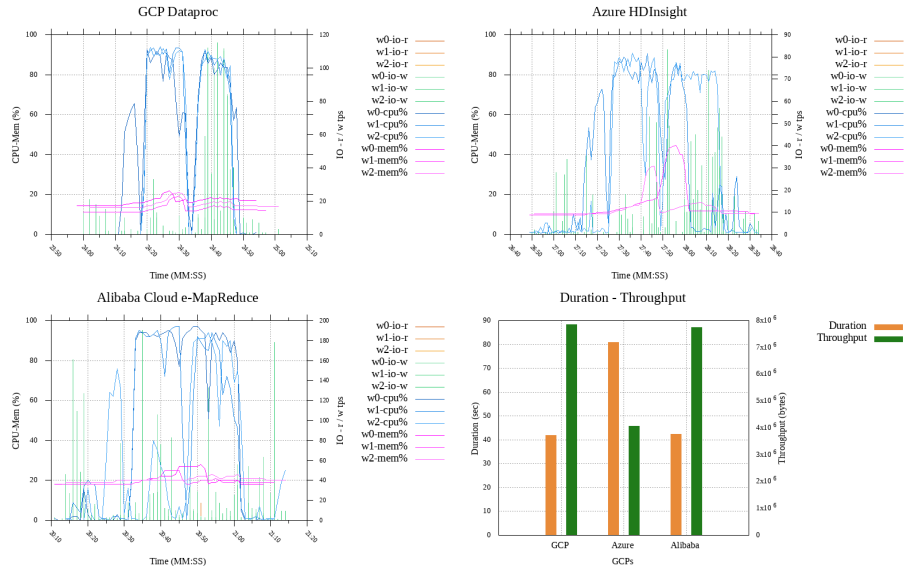


Figure 31: UC2 - Sort (Large; 320 MB)

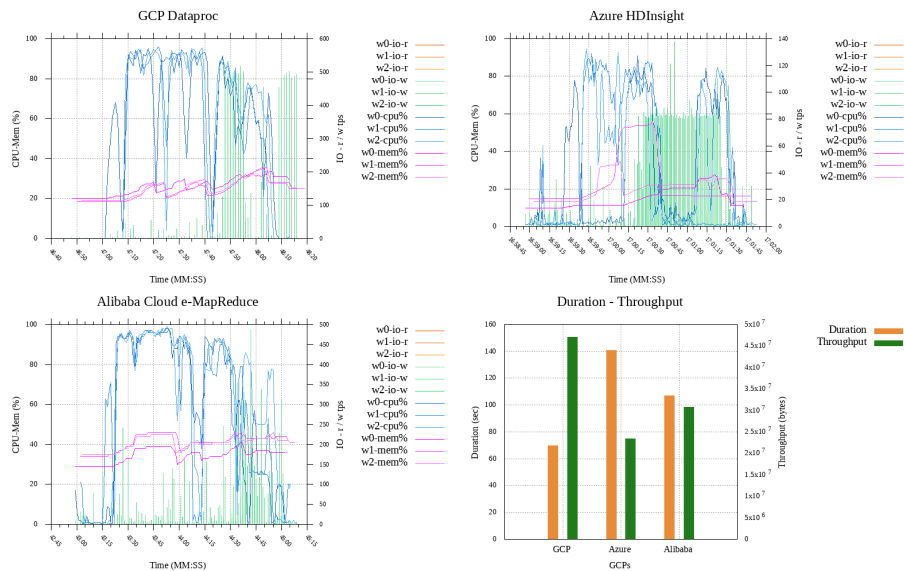


Figure 32: UC2 - Sort (Huge; 3.2 GB)

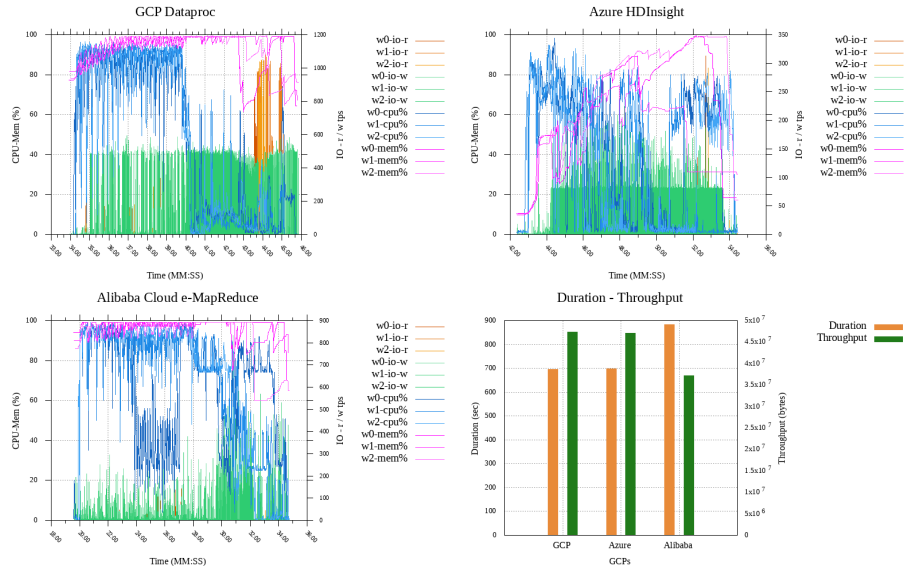


Figure 33: UC2 - Sort (Gigantic; 32 GB)

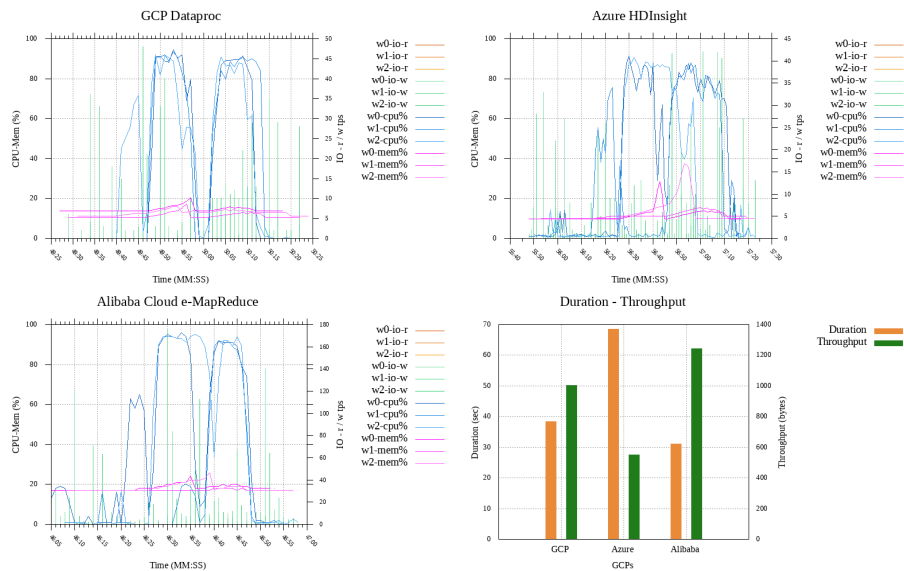


Figure 34: UC2 - Wordcount (Tiny; 32 KB)

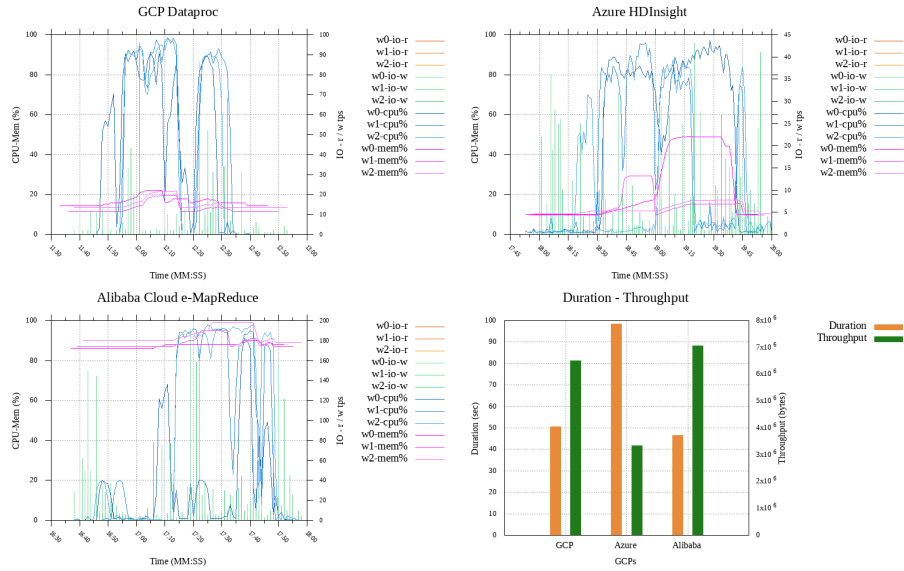


Figure 35: UC2 - Wordcount (Small; 320 MB)

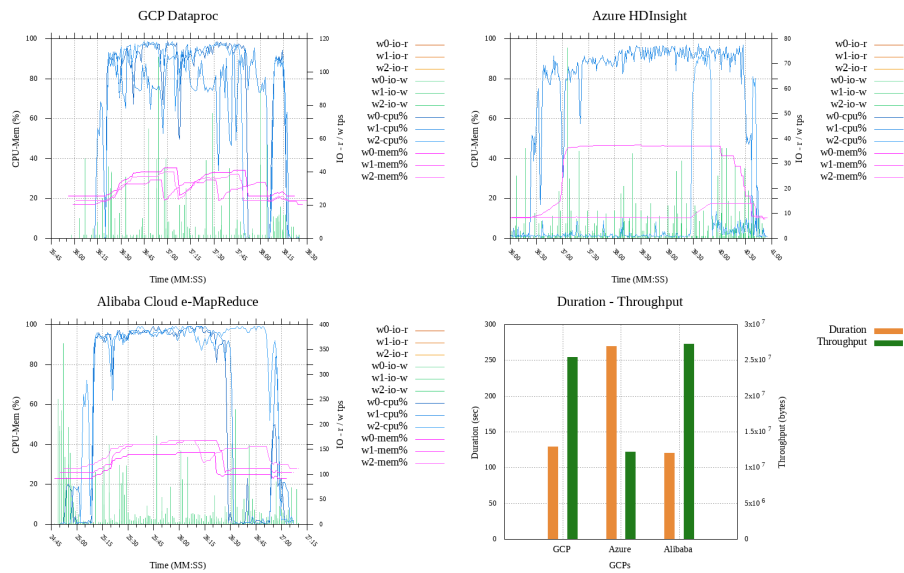


Figure 36: UC2 - Wordcount (Large; 3.2 GB)

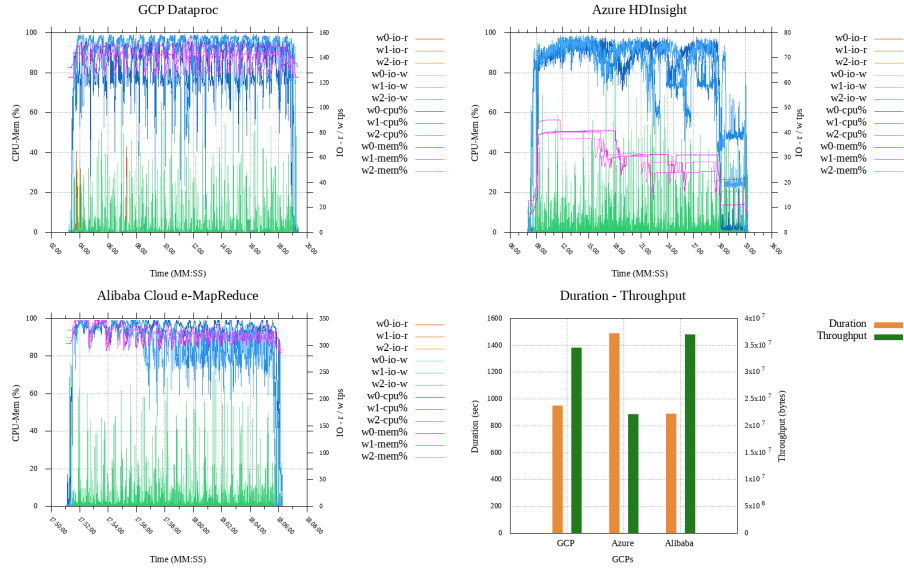


Figure 37: UC2 - Wordcount (Huge; 32 GB)

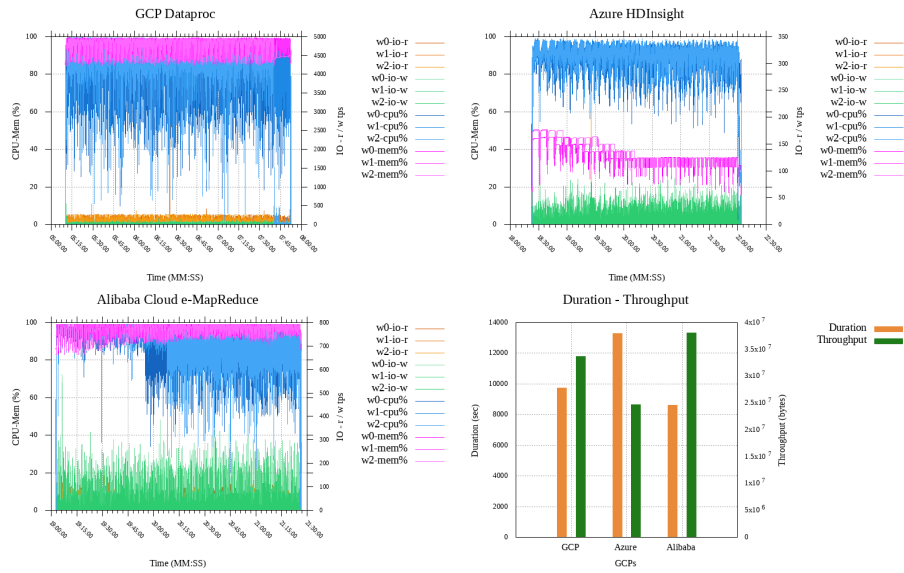


Figure 38: UC2 - Wordcount (Gigantic; 320 GB)

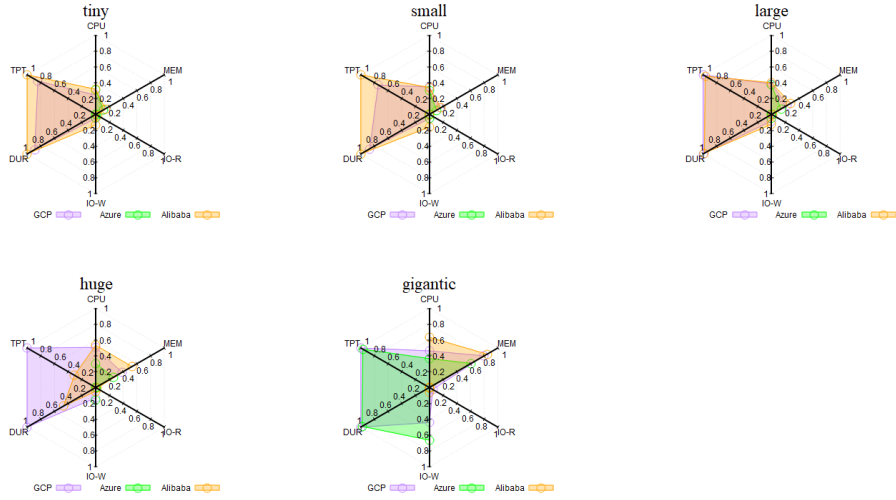


Figure 39: Use Case 2 - Sort performances along data scales

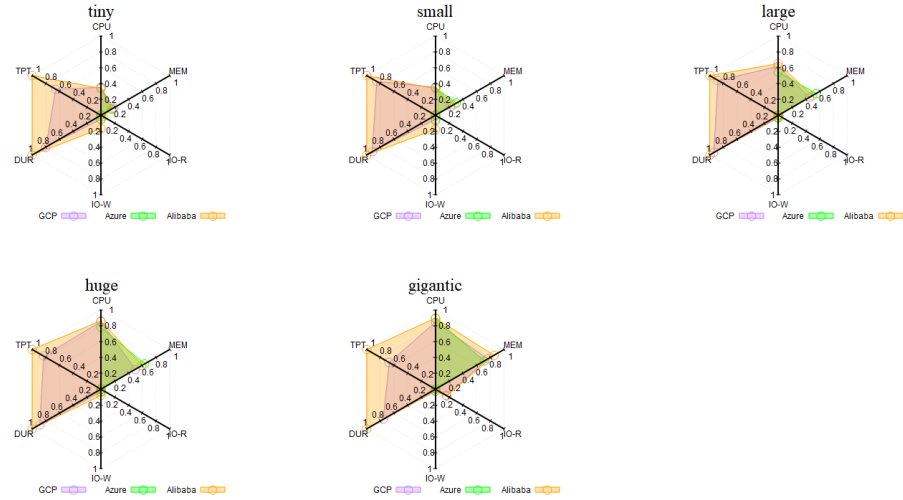


Figure 40: Use Case 2 - Wordcount performances along data scales

between successive performances of respective providers. Another result of the
aforementioned limitation is that we also left off the largest predefined data scale
of HiBench, namely Bigdata. Execution of this benchmark would have provide
us performance outputs in a very large data scale like 300 GB for Sort, 1.6 TB for
515 Wordcount, and 12.000.000 Pages by 1.000.000.000 Uservisits for Aggregation,
to name but three. Hence, we recommend the study to be understood in terms
of an attempt trying to bring more clarity to black-box nature managed Hadoop
proposals' performance behavior by means of resource utilization dynamics, and
as the managed services come out-of-the-box without any performance tweaking
520 configurations applied. We do not recommend the study to be understood as a
grading board for the business values provided by respective CSPs.

HiBench comes with dependencies downloaded during its compilation process
by Apache Maven. The Hive engine is one of those dependencies leveraged
by HiBench for running SQL workloads Scan, Join, and Aggregation. Alibaba's
525 e-Mapreduce comprises a ready made Hive hook trigerring a Java file to run post
executional transactions for other services within the package. However this
preconfiguration prevented HiBench from starting with respective benchmarks'
execution since the HiBench based Hive engine does not include the aforemen-
tioned jar file defined for e-MapReduce's specific environment. A workaround
530 attempt, copying the related jar file to an appropriate directory within HiBench
made the jar file available, however this time HiBench's Hive engine of an older
version did not support the hook "hive.exec.post.hooks" defined in Alibaba's
Hive configuration. At this point disabling the respective Hive hook from Al-
ibaba e-MapReduce's UI management console apparently solved this issue and
535 enabled HiBench's SQL workloads to run, but the impact of this modification on
the respective performance values remains unknown, hence the need to annotate
it here. With GCP and Azure issues of this kind did not occur.

In Azure environment the Terasort benchmark running in data scale gigantic
failed to complete in all three attempts we conducted where about 20% of maps
540 operations were reached. Showing distinction to GCP and Alibaba where the
end user is liberate in chosing HDFS or respective provider's cloud storage

service as the cluster’s file system, Azure obligates the user to go with WASB file system among other Azure storage services with a promise to Peta-scale. However, after inspecting the failure it turned out that the exception is not
545 due the WASB file system. Even though WASB is predefined as the cluster’s file system, during the application’s run time YARN still leverages the cluster’s HDFS file system for storing intermediary results failing to allocate free space on HDFS reaching specific levels of maps operators. The conductors marked this failure as a structural bottleneck, since all end users running Terasort operation
550 at this scale would face the same error, and since resource utilization can be tracked up to the failure point and after, we kept this benchmark by marking as incomplete. Figure 8 displays system resource utilization on Azure including the point where the failure occurs.

6. Conclusion and Future Work

555 Managed Hadoop systems are by CSPs provided cloud service proposals comprising pre-installed and pre-configured Hadoop clusters in order to eliminate the hard work on planning and installing Hadoop clusters manually on VM instances hence reducing the cluster installation to a matter of property selection and letting end users focus only to applications that will run on the cluster. As
560 managed systems come in a black-box nature respective pre-configurations and their impacts on the cluster performance remains unclear to the end user. In the study we conducted HiBench benchmarks against Hadoop clusters on PaaS in their proposed form as they come out-of-the-box, provided by respective CSPs. For each CSP, among the proposed installation specifications, we selected same
565 geographical location as region and by-promise apparently same CPU and Memory properties. Our aim is to compare those systems’ performance behaviors by leveraging HiBench Benchmark Suite. We executed Hadoop related benchmarks on all three Hadoop on PaaS services and captured system resource utilization data in the meanwhile. The results yield that Hadoop PaaS offerings by vendor’s
570 promise do not necessarily utilize system resources in similar ways hence

may highly distinct in some cases. Our assumption is that the pre-configuration choices applied by CSPs highly impact performance results even with apparently same hardware settings. The end users shall be aware of their use cases' requirements and truly understand what CSPs configuration proposals and their
575 respective outcome.

Based on the gained experience the study might varied in an approach more sensitive in balancing small performance differences among CSPs. As HiBench offers benchmarks for Spark, graph and streaming frameworks as well, the comparison can be made from within the respective perspective, putting also other
580 global CSP players into the scope.

References

- [1] World Internet Users Statistics and 2020 World Population Stats.
URL <https://www.internetworldstats.com/stats.htm>
- [2] Apache Hadoop.
585 URL <https://hadoop.apache.org/>
- [3] S. Ghemawat, H. Gobioff, S.-T. Leung, The Google file system, in: Proceedings of the nineteenth ACM symposium on Operating systems principles, Vol. 37, 2003, pp. 29–43, issue: 5. doi:10.1145/1165389.945450.
- [4] T. White, Hadoop: the definitive guide, fourth edition Edition, O'Reilly,
590 Beijing, 2015, oCLC: ocn904818464.
- [5] J. Dean, S. Ghemawat, MapReduce: Simplified data processing on large clusters, 2004, pp. 137–149.
- [6] a. A. T. H. Schtzle, Martin Przyjacił-Zablocki, Giant Data: MapReduce and Hadoop ADMIN Magazine.
595 URL <http://www.admin-magazine.com/HPC/Articles/MapReduce-and-Hadoop>

- [7] Announcing Amazon Elastic Compute Cloud (Amazon EC2) - beta.
URL <https://aws.amazon.com/about-aws/whats-new/2006/08/24/announcing-amazon-elastic-compute-cloud-amazon-ec2---beta/>
- 600 [8] Dataproc.
URL <https://cloud.google.com/dataproc>
- [9] Compute Engine: Virtual Machines (VMs).
URL <https://cloud.google.com/compute>
- [10] Cloud Storage.
605 URL <https://cloud.google.com/storage>
- [11] Azure HDInsight - Hadoop, Spark, & Kafka Service | Microsoft Azure.
URL <https://azure.microsoft.com/en-us/services/hdinsight/>
- [12] Announcing general availability of Azure HDInsight 3.6.
URL <https://azure.microsoft.com/en-us/blog/announcing-general-availability-of-azure-hdinsight-3-6/>
610
- [13] What is E-MapReduce? - Product Introduction | Alibaba Cloud Documentation Center.
URL <https://www.alibabacloud.com/help/doc-detail/28068.htm?spm=a2c63.128256.b99.4.65e270b2YXyKDV>
- 615 [14] Elastic Compute Service (ECS): Elastic & Secure Cloud Servers - Alibaba Cloud.
URL <https://www.alibabacloud.com/product/ecs>
- [15] Alibaba Cloud Linux OS.
URL <https://alibaba.github.io/cloud-kernel/os.html>
- 620 [16] Intel-bigdata/HiBench, original-date: 2012-06-12T07:56:57Z (Jan. 2021).
URL <https://github.com/Intel-bigdata/HiBench>
- [17] S. Huang, J. Huang, J. Dai, T. Xie, B. Huang, The HiBench benchmark suite: Characterization of the MapReduce-based data analysis, in:

- 2010 IEEE 26th International Conference on Data Engineering Workshops
 625 (ICDEW 2010), 2010, pp. 41–51. doi:10.1109/ICDEW.2010.5452747.
- [18] Release HiBench-7.1.1 Intel-bigdata/HiBench.
 URL /Intel-bigdata/HiBench/releases/tag/v7.1.1
- [19] L. Yi, J. Dai, Experience from Hadoop Benchmarking with HiBench: From
 Micro-Benchmarks Toward End-to-End Pipelines, in: T. Rabl, N. Raghu-
 630 nath, M. Poess, M. Bhandarkar, H.-A. Jacobsen, C. Baru (Eds.), *Advanc-
 ing Big Data Benchmarks*, Vol. 8585, Springer International Publishing,
 Cham, 2014, pp. 43–48, series Title: *Lecture Notes in Computer Science*.
 doi:10.1007/978-3-319-10596-3_4.
 URL http://link.springer.com/10.1007/978-3-319-10596-3_4
- 635 [20] N. Poggi, A. Montero, D. Carrera, Characterizing bigbench queries, hive,
 and spark in multi-cloud environments, *Lecture Notes in Computer Sci-
 ence (including subseries Lecture Notes in Artificial Intelligence and Lec-
 ture Notes in Bioinformatics)* 10661 LNCS (2018) 55–74. doi:10.1007/
 978-3-319-72401-0_5.
- 640 [21] N. Poggi, J. L. Berral, T. Fenech, D. Carrera, J. Blakeley, U. F. Min-
 has, N. Vujic, The state of SQL-on-Hadoop in the cloud, in: *2016 IEEE
 International Conference on Big Data (Big Data)*, 2016, pp. 1432–1443.
 doi:10.1109/BigData.2016.7840751.
- [22] Y. Samadi, M. Zbakh, C. Taddonki, Performance comparison between
 645 hadoop and spark frameworks using HiBench benchmarks, *Concurrency
 Computation* 30 (12). doi:10.1002/cpe.4367.
- [23] H. Ahn, H. Kim, W. You, Performance Study of Spark on YARN Cluster
 Using HiBench, 2018. doi:10.1109/ICCE-ASIA.2018.8552137.
- [24] S. Han, W. Choi, R. Muwafiq, Y. Nah, Impact of memory size on bigdata
 650 processing based on hadoop and spark, Vol. 2017-January, 2017, pp. 275–
 280. doi:10.1145/3129676.3129688.

- [25] T. Ivanov, R. Niemann, S. Izberovic, M. Rosselli, K. Tolle, R. V. Zicari, Performance Evaluation of Enterprise Big Data Platforms with HiBench, in: 2015 IEEE Trustcom/BigDataSE/ISPA, Vol. 2, 2015, pp. 120–127. doi: 10.1109/Trustcom.2015.570.
- [26] Gartner Reprint.
URL <https://www.gartner.com/doc/reprints?id=1-1ZDZDMTF&ct=200703&st=sb>