

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 Artificial Intelligence Engineering*

Abstract

Big Data's advent emerged in any data-driven domain and scaled up the extent and depth of data and its handling. New approaches leaning on enhanced distributed storage and computing paradigms are invented in an ongoing maturing process to overcome management and running analytics challenges. Within the given context, Hadoop is embraced on a wide scale by beneficiaries from industry and academia since its first release in 2005. Cloud Computing's commercialization started a grand migration movement towards the cloud, impacting Hadoop to transfer its presence from on-premises to virtual machines stored and tamed in extensive data center facilities by global Cloud Service Providers. The CSPs' response to result-focused analytics purposes emerged a service called managed systems where the contractor overtakes the demanding workload of multi-node cluster implementation by 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 reduced costs commercially presented as pay-as-you-go or pay-per-use. However, there is a payoff; managed Hadoop systems do present a black-box behavior to the end-user who cannot be clear on the inner performance dynamics, hence the benefits by leveraging

*Corresponding authors

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

them. In the study, we selected three global providers (Google Cloud Platform, Azure, and Alibaba Cloud), activated their Hadoop PaaS services (Dataproc, HDInsight, and e-MapReduce, respectively) within the same geographical region and by promise same computing specifications, and executed several Hadoop workloads of the HiBench Benchmark Suite. The results yield that the same computation specs among CSPs' services as they come out-of-the-box do not guarantee close performance outputs to each other, nor consistent performances within themselves. We assume that the pre-configuration work of managed systems done by the contractor plays a weighing role in their performance.

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

2010 MSC: 00-01, 99-00

1. Introduction

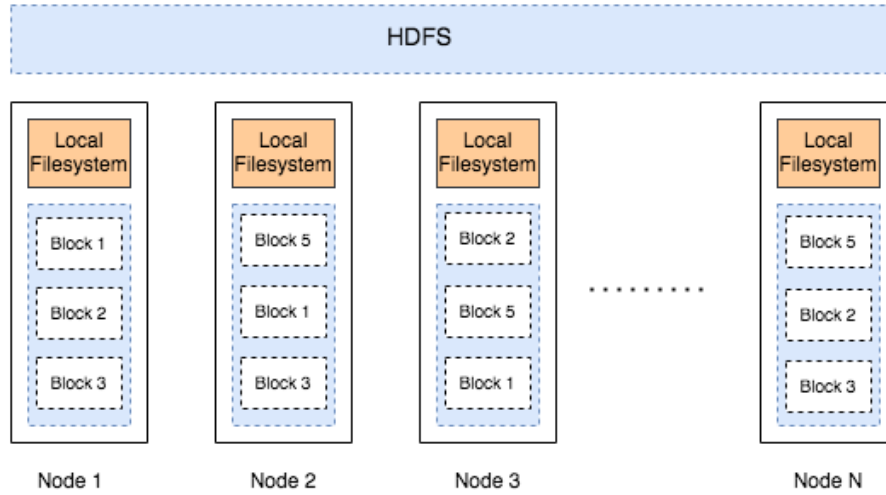
Big Data. An indispensable aspect for enterprises and academia of the information era is Big Data. As the estimated global internet access rate covers a weighing majority of roughly 63% of the global human population by 2020
5 [1], mobile technology devices become more democratized, sensors and IoT devices the more occupy daily life, and ongoing scientific researches produce vast amounts of data outputs, the Big Data phenomenon gathered itself utilizing overwhelming size with Volume, ever-accelerating growth rate with Velocity, and diverse data structures with Variety, new approaches were forced to mature
10 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.
15 Considering the complexity of dealing with big data, Hadoop represents a modern analytics framework decreasing management efforts and analytics opera-

tions' duration to an acceptable level employing affordable commodity computers. For its core functionality, Hadoop comprises three:

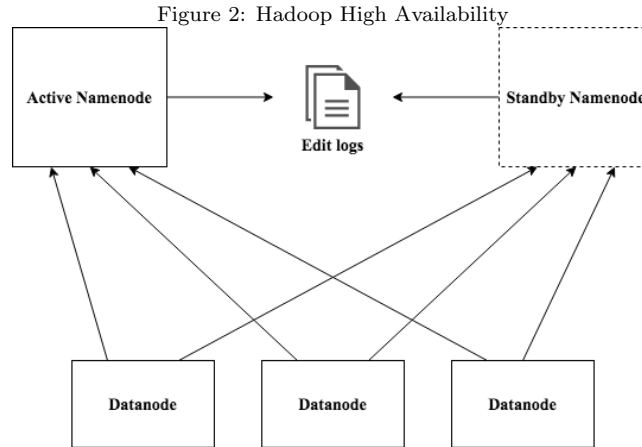
- HDFS filesystem for storing extensive data across a cluster of nodes,
- the MapReduce framework developed for distributed computation, and
- YARN for allocating available resources for the requested tasks.

Figure 1: Hadoop Distributed File System



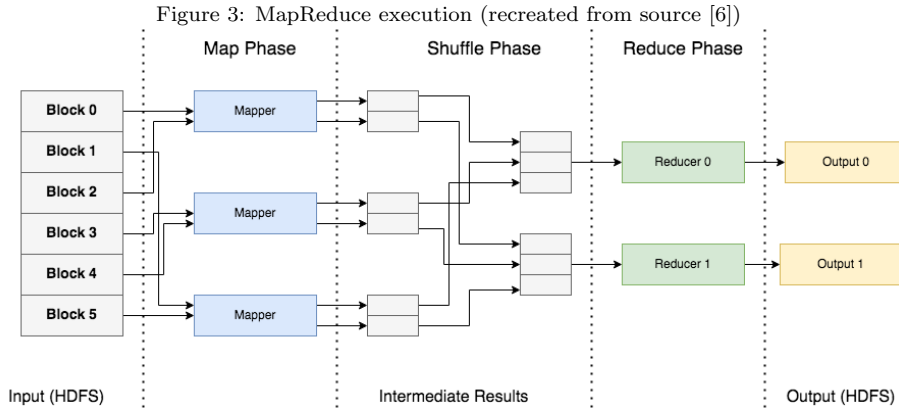
HDFS. Hadoop Distributed File System, which stands for HDFS, is developed with inspiration from the guidelines described in a whitepaper about Google's Filesystem (GFS), a distributed storage paradigm to handle petabytes and larger-scale data volumes within a cluster robust to machine failures, published in 2003 [3]. Significant data volumes are chunked and stored within 128 MB blocks, and 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; if 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 of HDFS. The architecture of HDFS comprises Namenode, a dedicated machine to keep track of the files and folders and respective metadata like block locations across the cluster, and many data nodes 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 exist two namenodes, one active namenode, and one standby namenode communicating with the data nodes and storing edit logs in a shared folder. As the naming refers, the active namenode is in charge, whereas the standby node behaves more like a shadow system. Whenever the active namenode breaks down, the standby namenode gets activated, resulting in no service breakage for the end-user.



MapReduce. Like HDFS, Hadoop's MapReduce is the open-source implementation of the MapReduce framework described in a paper [5] published in 2004.

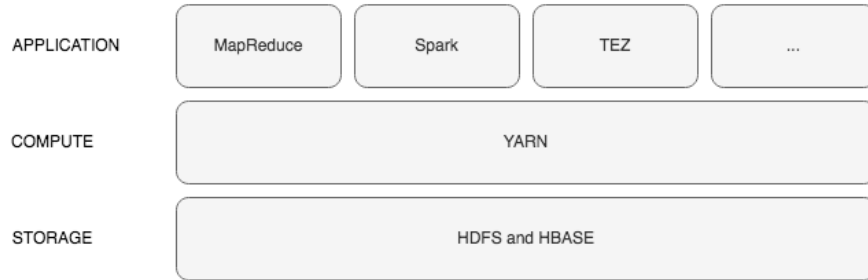
An MR flow, as depicted in Figure 3, starts with assigning blocks from HDFS as input splits to mappers, computed intermediate results are then shuffled and passed to reducers where outputs are sent back to the client. Nodes running mappers and reducers provide parallel processing, scaled by simply adding new commodity computers. The computation is made on nodes where related data blocks reside, which explains the term data locality; executing the computation by leveraging local resources of the node where the respective data block is also stored eliminates the need for moving data across nodes for computation.



YARN. Hadoop version 2.x includes significant architectural improvements in terms of resource management. 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", emerged as an intermediate layer between HDFS and MapReduce by taking over some of the load that was previously carried out by MRv1, has become a gate to 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 the previous version hence more efficient with processing oriented focus. YARN innovates Hadoop by bringing the architectural elements resource manager, which is one

node dedicated to tracking resources across the cluster by availability, and node managers who reside inside each worker node and monitor the containers.

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



Cloud Computing. The commercialization of Cloud Computing in the midst
 2000s [7] delivered utilization of storage and computing resources to the end-
 75 users saving them high investments in hardware technology that is soon going to
 be obsolete and is expensive to maintain. As the cloud migration 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 embraced the need
 80 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 by merely determining some set-
 tings like hardware specs and node numbers prior the installation. The Managed
 85 Hadoop system is both a blessing and a curse; by leaving the hard implemen-
 tation part, which is not necessarily related to the primary 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 to reach better performance
 90 results on selected approaches like memory-intensive or compute-intensive ap-
 plications. However, from the end-user's point of view, the performance of a

managed system’s user-defined specifications remains uncertain.

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

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 to its conjugates by not being a Hadoop cluster installed on the cloud VM service layer, namely Azure Virtual Machines; instead, it is an HDP platform optimized 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 study’s date, Azure was obligating Hadoop to use in High Availability mode with two master nodes hence leaving hardware 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-

land China are not as numerous as its conjugates hence available in the United States, Europe, Middle East, and the Asia Pacific. Alibaba Cloud’s managed service distinct in its pre-installed Operating System, Aliyun Linux 2, is developed by Alibaba Cloud, a Linux distribution based on CentOS, the open-source
125 version of RedHat Linux. AliyunOS proclaims to provide a stable and reliable environment optimized for the Alibaba Cloud infrastructure and be open source at 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 Alibaba Cloud e-MapReduce, the type of
130 storage, whether it shall be HDFS or leverage Object Storage Service (OSS) [16], Alibaba’s cloud storage, can be specified by the end-user.

HiBench [17]. The open-source HiBench implementation, introduced to the community in 2010 [18], is motivated by the aim of providing a comprehensive and representative benchmark for Hadoop, helping evaluate different big
135 data frameworks in terms of speed, throughput, and system utilization. At the study’s date, the current version of HiBench is 7.1, comprising 29 benchmarks categorized in 6 sections (Micro, Machine Learning, SQL, Websearch, Graph, and Streaming). Table 1 lists the Hadoop related workloads in HiBench 7.1 [19] executed in the study. The benchmarks in the category Micro are taken from
140 Hadoop’s native benchmark tools, where Dfsioe is the enhanced version of its originator, which calculates the average I/O rate, and average throughput of each map task, and aggregat throughput of the cluster. The column ”Engine” points out HiBench’s dependencies, which are downloaded when Apache Maven compiles the benchmark suite and leveraged during benchmark executions; Hive
145 engine is leveraged to run HiveQL queries for Scan, Join, and Aggregation benchmarks. The Hadoop based ML benchmarks are driven by Mahout; Nutch Engine is leveraged for Nutchindexing benchmark; Pegasus, a peta-scale graph mining system, calculates PageRank algorithm. The dependencies execute benchmark tasks by translating respective jobs into MapReduce. HiBench’s predefined default data scales, namely, tiny, small, large, huge, gigantic, and bigdata, provide
150

Table 1: HiBench 7.1 - Hadoop-related Workloads

Category	Workload	Engine
Micro	Sort	M/R
	Terasort	
	Wordcount	
	Dfsioe	
SQL	Scan	Hive
	Join	
	Aggregation	
ML	Bayes	Mahout
	Kmeans	
Websearch	Nutchindexing	Nutch Engine
	Pagerank	Pegasus

a scale varying from 32 KB up to 300 GB for Sort, 6 GB for Terasort, and 1.6 TB for Wordcount. User-defined data scales are also applicable within HiBench configurations.

2. Related Work

Our search for related work to the study, which focuses on benchmarking Hadoop on PaaS, i.e., managed Hadoop systems, yielded one paper with a similar approach to our intention, and is referred to first below. Other works we have found present differing approaches in benchmarking, such as stressing Hadoop on IaaS with manually installed clusters, testing the before and after tweak performances, comparing different Hadoop environments, or comparing Hadoop vs. Spark frameworks.

Poggi et al. [20] conduct a comprehensive study benchmarking Hadoop PaaS services provided by Amazon (EMR), Google (Dataproc), and Azure (HDInsight). The benchmark tool they use, namely BigBench (TPCx-BB), is an industry-standard benchmarking framework developed by Transaction Processing Performance Council (TPC). TPCx-BB comprises 30 business use cases and

differentiates from SQL-only benchmarks by also requiring other frameworks like MapReduce, user code (UDF), Natural Language Processing, and Machine Learning. The researchers’ study objective is twofold: Firstly, it is characterizing BigBench queries and out-of-the-box performances of Hive and Spark in the cloud, and secondly, it compares the cloud vendors in terms of reliability within a data scale ranging from 1GB to 10TB. The medium-size clusters put under test consist of 16 cores and 60+GB memory for the master node and 16 data nodes with 128 total cores. The executions of TCPx-BB along with the data scales show that Hive performs in most cases better in up to 1TB, whereas, in a 10TB data scale, Spark hits a significantly better performance, up to two times faster than Hive.

Samadi et al. [21] 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 to 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. [22] put Spark on YARN’s performance on the test with HiBench to handle a deluge of data generated by IoT devices. The experiment is run on a cluster with one master and three worker nodes, each node possessing an Intel Xeon processor with 20 cores and 128GB main memory meaning 60 cores and 384GB memory in total. HiBench 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) is leveraged by a chosen data scale of 30 GB. Spark occupies memory during the whole job execution, which reduces I/Os’ negative impact on processor performance. The conductors modified YARN’s minimum memory allocation and Spark executor settings to optimize resource usage so that the Spark executors’ overall loads remain below total system memory. Alongside with HiBench’s duration and throughput report, CPU / memory

utilization and disk throughput are profiled as well. This paper’s finding points out that Spark guarantees performance when provided with enough memory.

200 Han et al. [23] study the impact of memory size on big data processing through Hadoop and Spark performance comparison leveraging HiBench’s k-Means workload as the only benchmark. For memory sizes 4, 8, and 12 GB, across a data scale from 1 to 8 GB, k-Means benchmark for Hadoop and Spark is executed. The results depict Spark’s overperforming Hadoop unless the total
205 input data size is smaller than 33.5% of worker nodes’ total memory size. After reaching that ratio, Spark suffers from 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 determine if Spark’s performance can
210 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 HiBench’s k-means benchmark outputs a report interpreted by the conductors as Spark show a 5-10% and 15% maximum improvement in processing time.

215 Ivanov et al. [24] compare the performances of two enterprise-grade applications, DataStax Enterprise (DSE), a production-level implementation of Apache Cassandra with extended features like in-memory computing and advanced security, to name but two, and Cloudera’s Distribution of Hadoop (CDH) comprising core Hadoop elements HDFS and YARN integrated with elements
220 belonging to the Hadoop ecosystem. DSE’s HDFS compatible file system CSF lets Hadoop applications run without any modification. The conductors installed the latest stable releases of both software on equal CPU, memory, and network infrastructure configuration; for both installations, default system parameters have been left with their defaults. HiBench’s three chosen workloads
225 (CPU-bound wordcount, I/O-bound dfsioe, and mixed HiveBench) are executed three times; furthermore, the average values have been taken for representativeness. Their study’s conclusions 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
230 the study differs to other studies using the YCSB benchmark suite. HiBench’s
results confirm the latter’s 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-
235 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 [25] either in leading or in niche
section.

We constructed the benchmark study in two use cases: Use Case 1 aims
240 to map the overall performance behaviors of respective CSP’s service 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).

245 Use Case 2. We picked one benchmark of I/O-bound character (Sort), one
benchmark of CPU bound character (Wordcount), and executed these in increas-
ing data scales delivered by HiBench: Tiny, Small, Large, Huge, and Gigantic.
The largest predefined HiBench data scale, namely Bigdata, would cross the
installed clusters’ storage limits; hence we left it off the execution.

250 Bound by the availability of their respective hardware and software options,
we selected the same or close settings within providers’ promises. Concerning
our aim to benchmark managed systems as they come out-of-the-box, we put
three basic principles to stick with for each provider in order to align initial
circumstances before benchmark execution: Among the CSPs, the respective
255 data center’s geographic location shall be the same or close (1), processor num-
ber and memory capacity (by providers’ promise) shall be same or as close as
possible (2), selected Hadoop versions among the providers’ offerings shall be

close to each other and remain within version limits supported by HiBench7.1, which corresponds to Hadoop 2.x. Thus Frankfurt is selected as the location for all providers’ data centers; within the given options, we selected 8 CPUs/64GB RAM for the master node and 4 CPU’s/32 GB RAM for each worker node totaling 12 CPUs and 96GB worker compute power for the cluster; within proposed images, Hadoop versions 2.7.3, 2.8.5, and 2.9 have been selected. Specified cluster installation options and details are given in Table 2. Without applying any performance tweak operation to the respective configurations, we immediately executed several Hadoop benchmarks from HiBench’s Micro, SQL, ML, and Websearch categories. We only modified the default cluster’s configuration in cases where the benchmark was detained from running, reported in the Discussion section.

We collected system utilization records addressing CPU, Memory, and I/O behavior on each worker node of the cluster during the benchmark execution. This approach enabled us to visualize the system utilization on each worker node within the cluster over the respective benchmark’s 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 transfers the respective block to the node with a free slot requiring a network activity, a possible decreasing impact over the performance. However, due to the small cluster size consisting of 3 nodes, we excluded the network activity track with the assumption data locality condition is provided during the benchmark’s executions and is likely to have the 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), and Websearch (Pagerank) have been executed on all three CSPs’ managed Hadoop services. Table 3 lists HiBench 7.1 benchmark reports for Use Case 1, Table 4 lists reports for Use Case 2.

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
Extras	–	1 Master Node clone for HA, 3 extra nodes for Zookeeper	–
WORKER NODES			
# of 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

Table 3: Use Case 1 benchmark results

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

Plots. During benchmark runtime, resource utilization on worker nodes has been captured. The plots are suggested to be perceived as follows: Top-left, top-right, and bottom-left plots represent CPU (user%), memory (%memused), and I/O 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, I/O-read and I/O-write tps' represented with orange tones and green tones. Even though the coloring convention might sound confusing, it gives a clear overview of the entire benchmark process's utilization behavior over time. The left-hand side x-axis measures CPU/Memory usage in percent; the right-hand side x-axis measures I/O-read and I/O-write transfers per second. The bottom-right plot represents the comparative benchmark performance outputs of the respective CSPs. Duration measure in seconds is expected to be perceived as "lower is better" while throughput, which is the amount of processed data per second in bytes, is expected to be perceived as "higher is better".

For each use case, we provided spider charts, i.e., radar plots, to give a comparative overview for system utilization vs. benchmark performances of relevant CSPs. In these plots, normalized average values of the captured CPU, Memory, and I/O data on the worker nodes have been averaged a second time in order to get scalar CPU, Memory, and I/O values representing the cluster’s average performance. For the sake of better visualization of I/O utilization, we removed 0 valued instances before taking the I/O-read and I/O-write data average. Combining the values mentioned above with normalized duration and throughput report values of HiBench outputs the study’s spider plots. Nevertheless, for the spider plots, both throughput and duration shall be perceived as ”higher is better”. Even though the duration is lower-is-better, for the sake of proper visualization, we display the shortest duration as the largest value, making it the most visible on the plot.

4.1. Use Case 1 Results

Sort - Huge (Figure 5). CPU utilization in GCP and Alibaba condense around 80% to 98%, whereas in Azure, the range spans between 50% and 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 varies between 10% and 50%. In the second half of the benchmark execution, I/O write transfers in GCP and Alibaba show peaks at about 500 tps, wherein Azure I/O transfer s limited with 100 tps. 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 rises to range 80% - 100%; I/O write transfers behave at about 500 tps where I/O 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 I/O-read and I/O-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 I/O write reaches the peak at 800 tps., which results in GCP embarking highest and
335 Azure the second highest throughput, respective durations output in 715 and 758 seconds.

Terasort - Huge (Figure 7). GCP depicts a high utilization of processors at a range of 80% - 100%, memory moving from 80% to 100%, and I/O behavior 500 tps in overall and peaking at 1000 tps in I/O-read, resulting in the highest
340 throughput in 667 seconds response time among other CSPs. Azure's processor and memory utilization fluctuate in overall execution where memory performance incrementally reaches 100% in one node, a stable I/O-write in overall process at about 70 tps peaking at 250 tps reaches the second highest throughput in 858 seconds. With high memory and processor utilization and varying
345 I/O tps' in the overall process, Alibaba falls back in embarking throughput and resulting in 1054 seconds response time.

Terasort - Gigantic (Figure 8). Switching the scale to gigantic causes dramatic resource utilization changes on all CSPs. During all benchmark processes, GCP depicts very condensed high utilization on the processor at a broad range of
350 30% to about 95%, memory consumption between 95% and 100%, and I/O read transfer varying between 800 to 1100 tps. On the other hand, Azure fails to complete the benchmark on three attempts, suffering from YARN insufficient HDFS allocation requested for bringing the job further. I/O scores drop to null, and task raises failure. I/O scores drop to null, and task raises failure. Alibaba
355 Cloud's resource utilization goes within maximum levels where processor utilization depicts a consumption of 80% to 90% overall, memory utilization at it highest during all process, and I/O reads and writes moving along the 600 tps' and peaking around 1600 tps. With a slightly higher value in throughput, Alibaba performs best with 9660 seconds, followed by GCP with 9821 seconds.
360 Azure disqualifies this session.

Wordcount - Huge (Figure 9). GCP performing processor utilization of 75% to 100% with a memory consumption between 80% to 95% where I/O 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 I/O transfers peaking at about 60 tps reaches the lowest throughput hence the most extended duration of 1470 seconds. Alibaba depicting high processor and memory load moving in ranges 70% to 100% and 80% to slightly over 90% with I/O 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 at their maximum during the overall process, for CPU within 60% up to pushing 100%, memory utilization varying between 60% to 100%, and I/O-read and I/O-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 the range of 20% up to 50%, I/O behavior seldomly reacts over 60 tps resulting in the lowest throughput and most extended duration of 13596 seconds. Alibaba depicting high processor utilization within range 85%-100%, relatively more stable memory consumption in the range 85% to 100%, and I/O transfers mostly about 300 tps, thus performing the highest throughput hence shortest response time of 8671 seconds.

Dfsioe-read - Huge (Figure 11). GCP's processor utilization is moving roughly between 80% to 95%, accompanied by memory utilization within range slightly below and over 70%, I/O transfers peaking at 400 tps mostly go along 100 tps to 250 tps performs second highest throughput resulting in 294 seconds of duration. Azure's processor utilization is somewhat limited to between 60% and 90%, whereas the memory utilization behaves between 10% to 50%, generally low I/O transfers with two markable I/O-read and I/O-write condenses reaching

peaks at 2000 tps and 1500 tps, respectively, results in the lowest throughput and 662 seconds of response time. With 245 seconds duration, Alibaba displays a similar resource utilization pattern with GCP where it differentiates in memory utilization, moving around 90%.

395 *Dfsioe-read - Gigantic (Figure 12).* Processor utilization in the GCP concentrates within the range of 80% to 90%; memory utilization moves around 95% to 100%, whereas I/O read transfers moving along about 500 tps bring its performance to the second highest throughput by 915 seconds duration. Azure performs within a range of 55% up to 90% in CPU utilization, 20% to 55%
400 in memory utilization, and I/O-write activity moving along 70 tps, I/O-read showing a peak at about 470 tps, which results in throughput outputting 1886 seconds response time. Alibaba system utilization moves along 80%-90% range for processing performance, 90%-100% for memory load, and behavior up to 1700 tps I/O-write placing its performance to the highest throughput with 660
405 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 I/O-write transfers move along 500 tps resulting in the second shortest response time 379 seconds. Azure's CPU utilization around 60% to 90%, memory load moves
410 between 20% and 40%, whereas I/O-write transfers vary about 20 tps to 80 tps peaking at about 120 tps by a duration output of 658 seconds. Alibaba, with processor utilization 80% and 95%, memory load increments from about 40% up to 90%-100%, and I/O-write transfers at about 100 tps to 600 tps peaking at about 800 tps performs the heaviest throughput, thus reaching the shortest
415 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, I/O-writes move along 500 tps to 600 tps placing GCP to the second-best performance with 1347 seconds

420 response time. Azure shows a CPU utilization in a range of 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 I/O-write transfers peak at about 120 tps resulting
in 1914 seconds response time. Alibaba performing the shortest response time
425 with 1060 seconds displays a similar memory and processor utilization behavior
with GCP where it differs in I/O-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 I/O-write transfer reaching over 500
430 tps, which brings GCP to the top 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
I/O-write transfer peaking at about 85 tps results in lowest throughput and
157 seconds duration. Alibaba follows a similar pattern like GCP in resource
435 utilization with about 90% CPU load, 15% to below 40% memory utilization
but a less dense I/O-write transfer peaking at about 550 tps and so reaching a
close throughput to GCP resulting in 74 seconds response time.

Scan - Gigantic (Figure 16). GCP's CPU utilization varying between 50% and
90% with convergence at about 80%, memory load showing similar range with
440 utilization reaching up 100% and I/O-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
load incrementing two times from 20% to 40%, I/O-write transfer moving
along 50 tps to 60 tps peaking at about 200 tps results in 514 seconds duration.
445 Alibaba's processor utilization moves in a relatively higher range between 80%
and 95%, incrementing memory load range from 40%-50% to 80%-100%
during runtime and I/O-write transfers peaking at about over 500 tps results in
the highest throughput, thus the shortest duration of 407 seconds.

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

Join - Gigantic (Figure 18). CPU load in GCP moves along 70% to 90% over-
all whereas RAM utilization among worker nodes takes a path within 50% to
460 90% range, I/O-write transfers observed in behavior about 100 tps resulting
in a throughput outputting 595 seconds of response time. Azure's performance
dynamics show a load of 80% to 90% with decreasing utilization on some worker
nodes in a later stage, memory utilization moving along the range of 20% up to
60%, and I/O-write transfer observations peaking at 120 tps provide the lowest
465 throughput hence the longest response time of 761 seconds. Alibaba is perform-
ing a processor utilization of about below 90% to below 100%, memory load
behavior moving along 50% to below 70%, and I/O-write observation peaking
at 350 tps performs a slightly higher throughput than GCP hence a slightly
shorter duration of 594 seconds.

470 *Aggregation - Huge (Figure 19).* GCP processor utilization moving around 80%
to about 95%, memory load behaving between 15% to 40%, and I/O-write
transfer observed at about 150 tps result in 97 seconds. Azure's CPU utilization
behaves between 80% and 90%, whereas memory load moves along from 10%
to varying utilization among worker nodes up to 30% and below 60%, and I/O-
475 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 memory, and I/O-write

transfer average of 26 tps show comparable performance to GCP resulting in the same completion time of 97 seconds.

480 *Aggregation - Gigantic (Figure 20).* Processor utilization moving between 80% and 90% in overall execution, memory load between 50% and 70% peaking to below 100%, I/O-write transfers reaching a dense behavior at the end reaching 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
485 moving along 30% to 60%, I/O-write transfers behaving about 100 tps resulting in relatively low throughput and 594 seconds of duration to complete. Alibaba's processor load moves along 80% to below 100%, I/O-write transfer behavior scaling from about 50 tps to 400 tps results in 565 seconds response time.

490 *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, and I/O-write transfer behavior about 500 tps resulting in highest throughput thus fastest response time of 2604 seconds. Azure displays a split behavior in CPU utilization among worker nodes, where one load moves
495 at about 80% to lower 100% and one node depicts a movement on the minimum at about 20%; memory load on worker nodes also vary from each other by about 20% load difference at the beginning however meeting the similar range starting at the second half, I/O observations depict a low range with the exception at the high I/O-write transfer at about 3200 tps at the initial state of the execution
500 result in a relatively low throughput hence the most prolonged duration of 6120 seconds. Alibaba CPU utilization shows varieties among worker nodes in load ranges; RAM utilization condenses at higher levels of 80%, I/O transfers moving along 100tps to observable 450 tps resulting in 3017 seconds response time.

Bayes - Gigantic (Figure 22). GCP's processor utilization moving along 70%
505 to below 100%, accompanied by memory load incrementing from 40% to 70% becoming stable at 50%, and I/O-write transfers moving along 500 tps results

in the highest throughput and 5350 seconds of response time. Azure’s resource utilization shows the distinction in CPU and memory among worker nodes while two 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 I/O-write transfer move along 100 tps where I/O-read transfer peaks at about 650 tps resulting in the lowest performance of 12589 seconds duration. Alibaba depicting different utilization ranges among worker nodes for processing, the memory consumption follows a more balanced load among worker nodes with I/O-write transfer moving along 200 tps a relatively lower throughput hence the relatively long duration of 6363 seconds, with respect to GCP.

Kmeans - Huge (Figure 23). GCP’s CPU performance behaves in the range of about 60% to below 100%, whereas memory utilization moves about 60%, increasing at the reduce phase where I/O-write activity also condenses 500 tps. The utilization in Azure depicts itself in a high CPU behavior of about 80% and 90%, a moderate memory consumption moving along 20% up to below 60%, and I/O transfers moving along 40 tps results in the second shortest duration. Alibaba utilizing processor and memory in high levels, including drops in processors on particular worker nodes, with an increased I/O-write transfer in reduce phase reaches the highest throughput hence shortest duration.

Kmeans - Gigantic (Figure 24). In the gigantic data scale, all three services remain their previous utilization behavior in a more condensed manner in I/O transfers. Alibaba and Azure competing in throughputs, the resulting outcome is 4034 seconds for Alibaba and 4042 seconds for Azure, GCP’s duration does not stand far apart by 4541 seconds.

Pagerank - Huge (Figure 25). GCP’s processor utilization behavior moves in a higher range at about 70% and below 100%; RAM utilization follows a midlevel path moving along to 50%, including peak points at below 100%, I/O transfers vary in the heights at about 500 tps resulting in the best performance by 1544

seconds. Azure performs a high utilization behavior in the processor at about 80% to over 90%, memory load changes between about 20% up to 60%, a lower level range of 100 tps to 150 tps characterizes its I/O transfer utilization resulting in 3334 seconds duration. Alibaba's both CPU and memory utilization
540 across worker nodes move between 80% and lower 100%, and I/O transfer behavior pushing upper limits at about 600 tps reaches second-highest throughput resulting in 2458 seconds.

Pagerank - Gigantic (Figure 26). Switching to a gigantic scale, the CPU load of GCP becomes more condense in a range of about 60% up to below 100%,
545 a fluctuating RAM utilization behavior incrementing three times to 100% over the execution time, and an I/O transfer at about 500 tps performs the highest throughput and shortest duration of 8,371 seconds. Azure's processor load varies between 70% up to below 100%, showing fluctuating memory utilization incrementing from 40% to below 100%, general I/O transfers are limited to
550 about 200 tps with peak exceptions at about 5000 tps performing the second shortest duration of 11,779 seconds. Alibaba's resource utilization draws condense load for CPU and memory, pushing the limits in a fluctuating style; I/O transfers mostly stay below about 300 tps, peaking at 600 tps to 1200 tps several times during execution, performing the lowest throughput result in the longest
555 duration of 13,893 seconds.

Use Case 1 Overview. Figure 27 and Figure 28 project a summarized version of the benchmark results in relation to system utilization averages of respective CSPs. In most cases, systems, which can afford more utilization, are closer to relatively better performance. The spider plot also depicts the main character
560 of a benchmark, whether it is I/O-bound like in Sort or CPU-bound as in Word-count, or both, distributed in the map and reduce phases as in SQL benchmarks Scan, Join, and Aggregation.

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

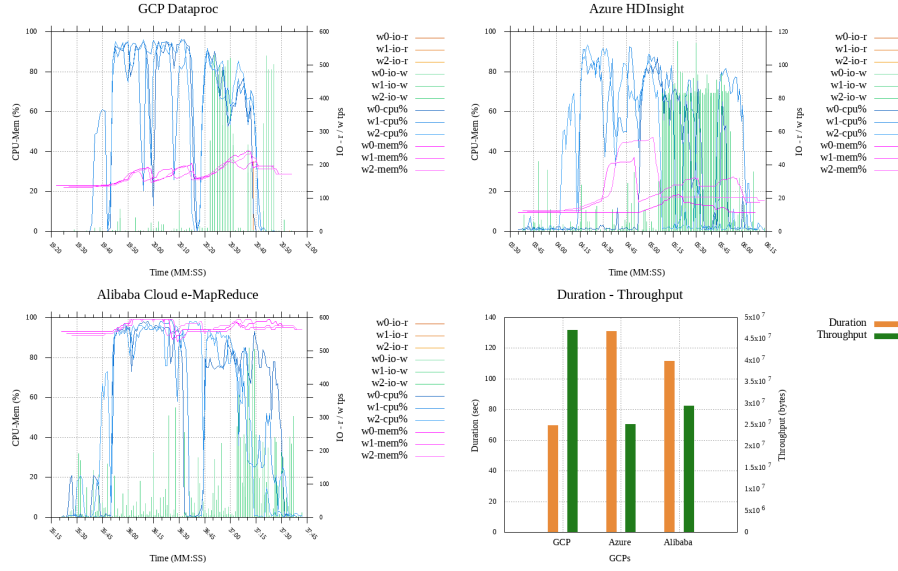


Figure 6: UC1 - Sort (Gigantic; 32 GB)

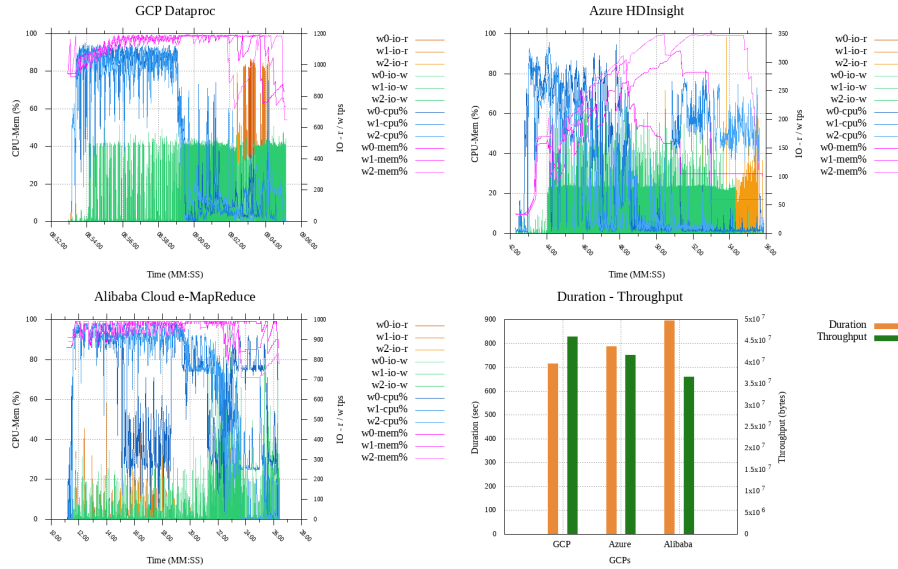


Figure 7: UC1 - Terasort (Huge; 320 MB)

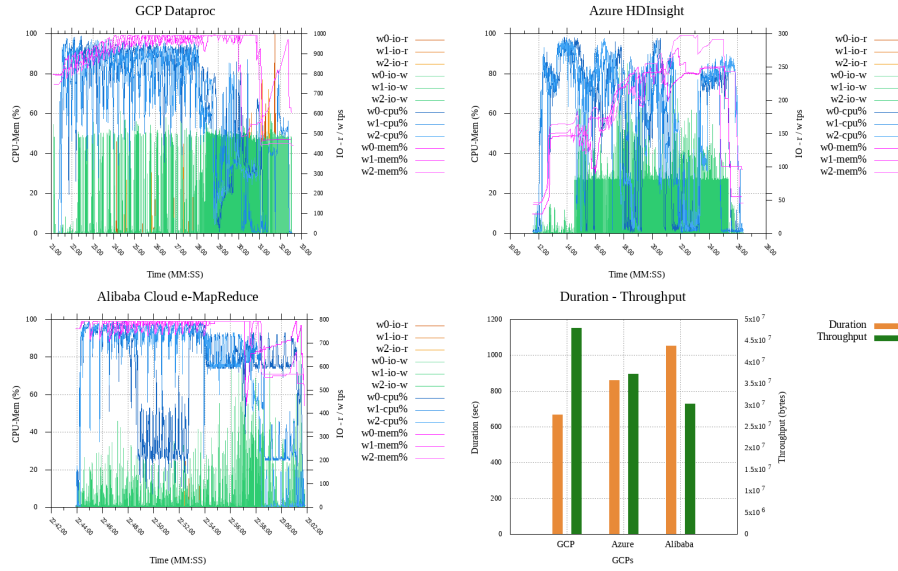


Figure 8: UC1 - Terasort (Gigantic; 3.2 GB)

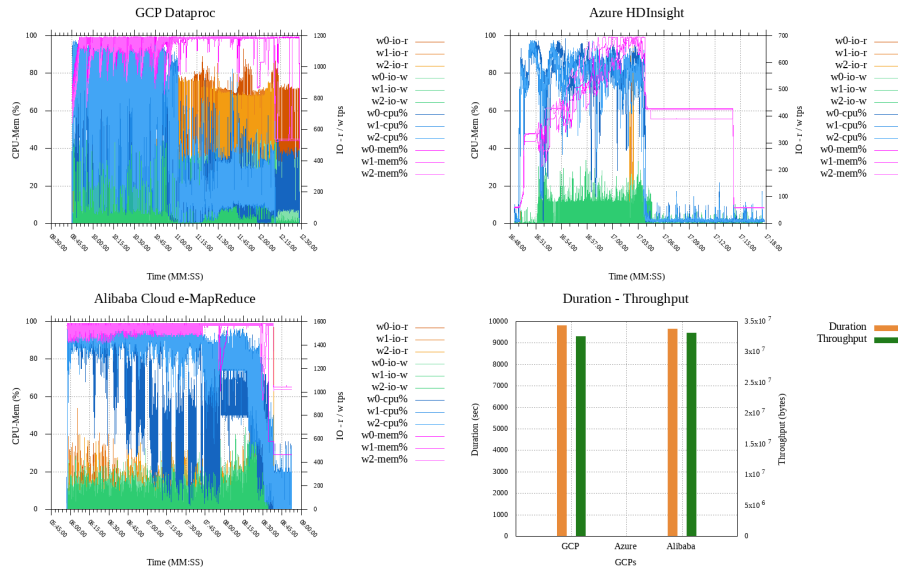


Figure 9: UC1 - Wordcount (Huge; 32 GB)

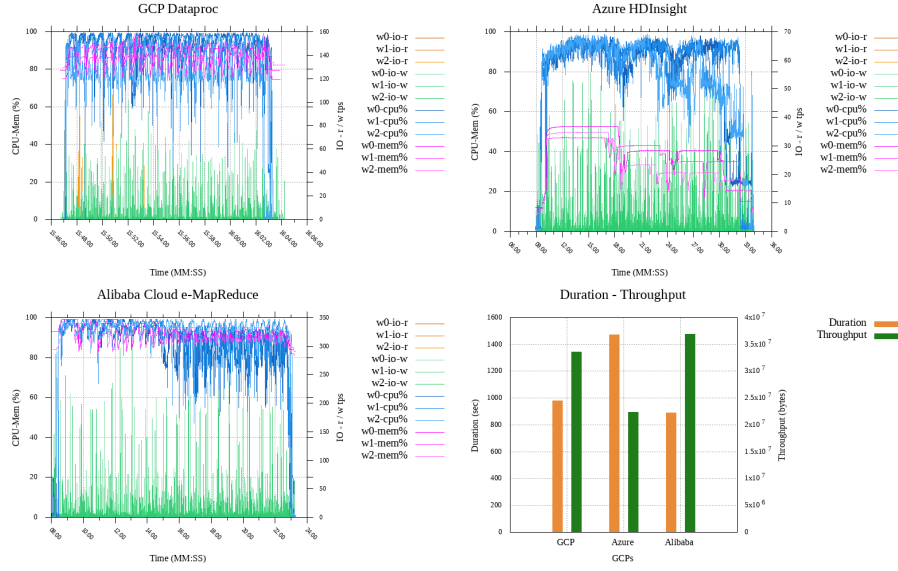


Figure 10: UC1 - Wordcount (Gigantic; 320 GB)

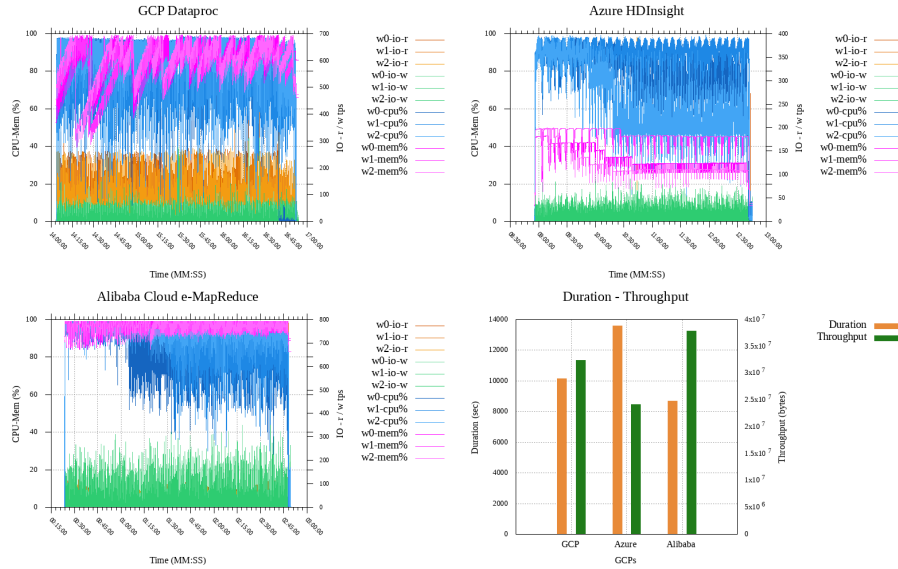


Figure 11: UC1 - Dfsioe-read (Huge; No of Files: 256, File size: 100 MB)

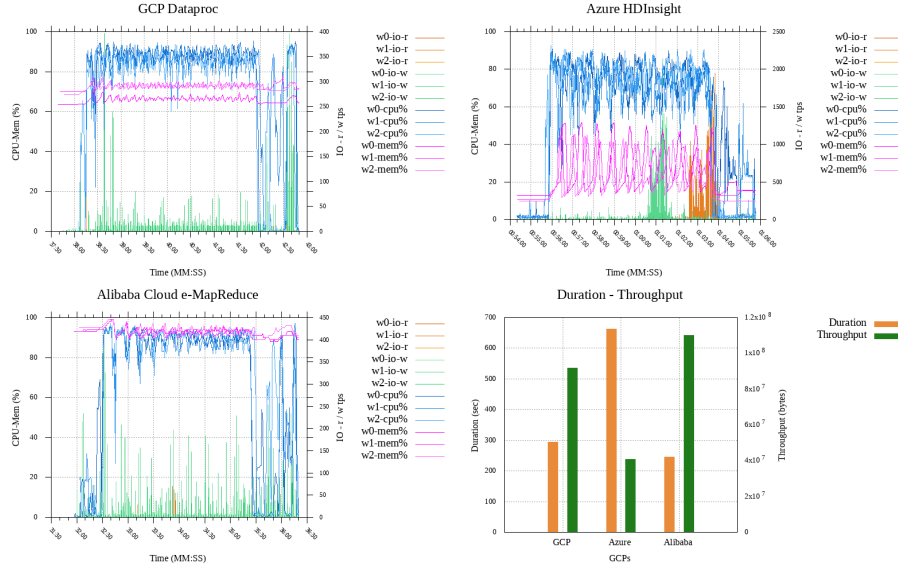


Figure 12: UC1 - Dfsioe-read (Gigantic; No of Files: 512, File size: 400 MB)

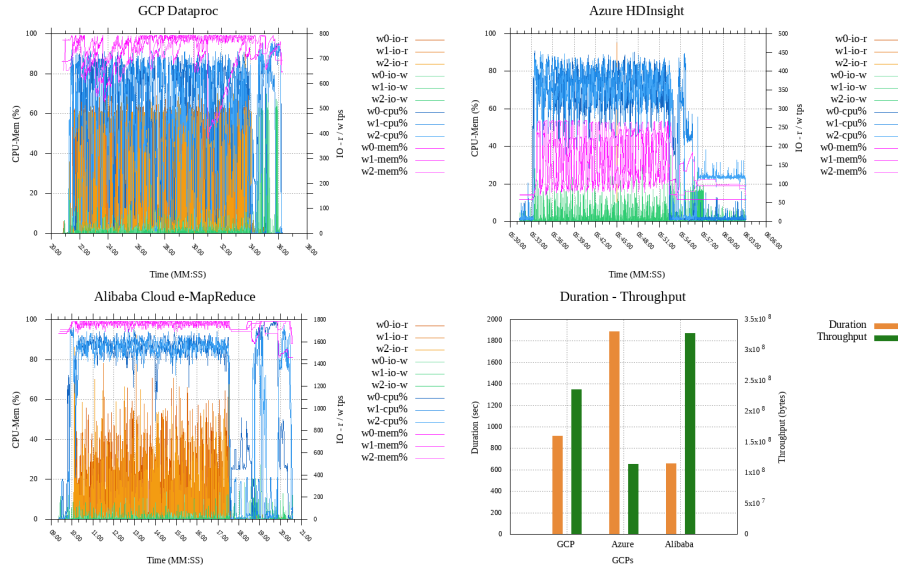


Figure 13: UC1 - Dfsioe-write (Huge; No of Files: 256, File size: 100 MB)

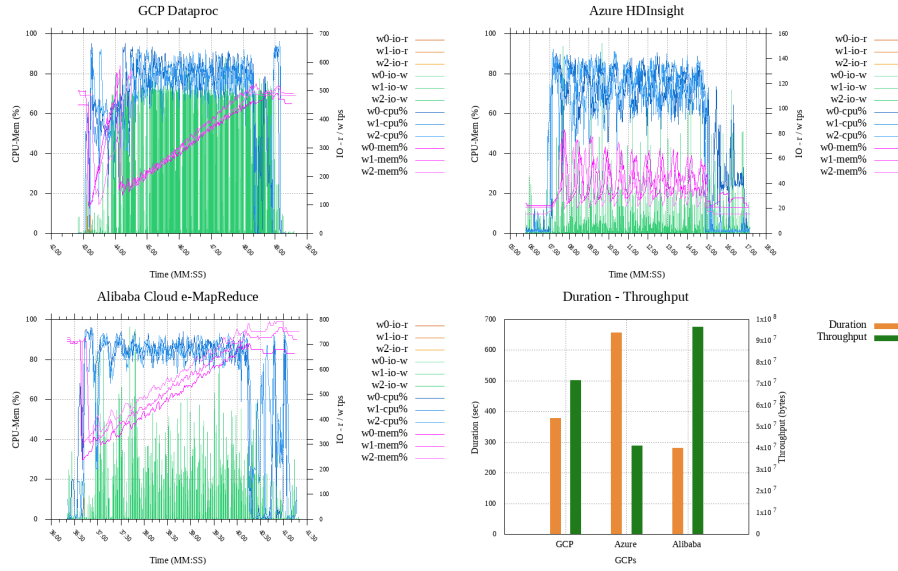


Figure 14: UC1 - Dfsioe-write (Gigantic; No of Files: 512, File size: 400 MB)

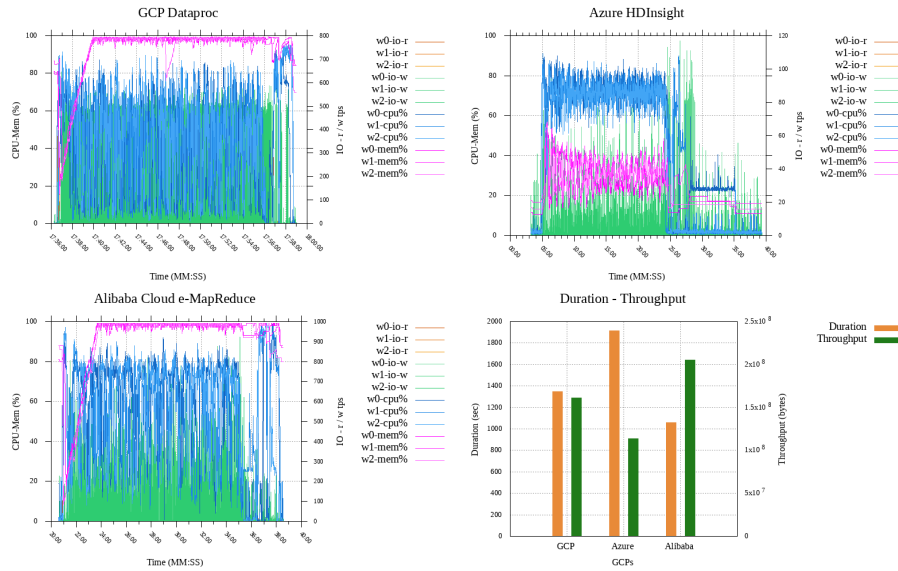


Figure 15: UC1 - Scan (Huge; USERVISITS: 10,000,000 PAGES: 1,200,000)

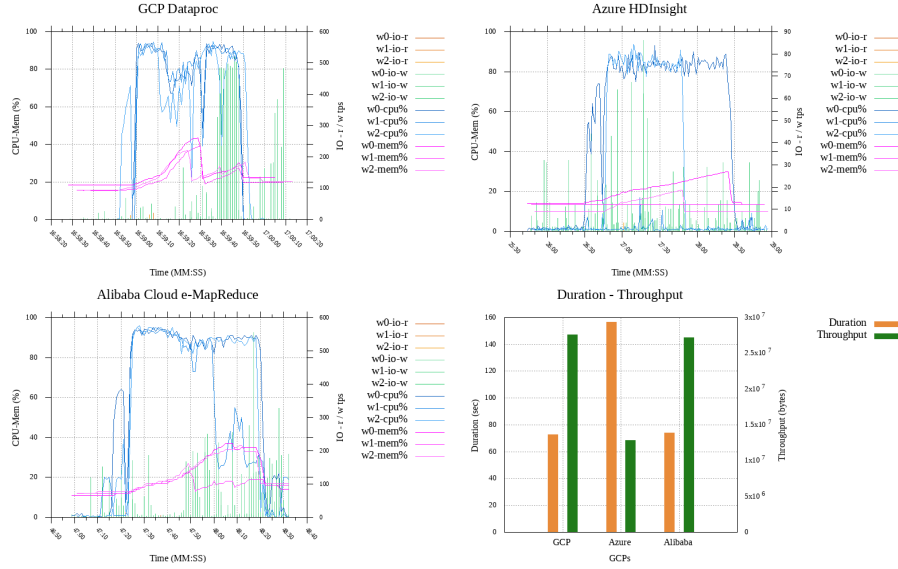


Figure 16: UC1 - Scan (Gigantic; USERVISITS: 100,000,000 PAGES: 12,000,000)

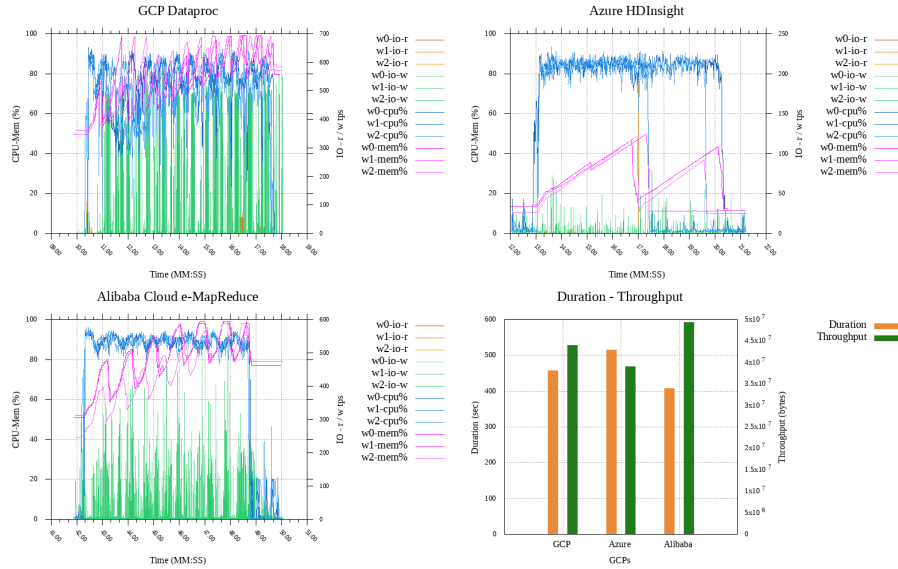


Figure 17: UC1 - Join (Huge; USERVISITS: 10,000,000 PAGES: 1,200,000)

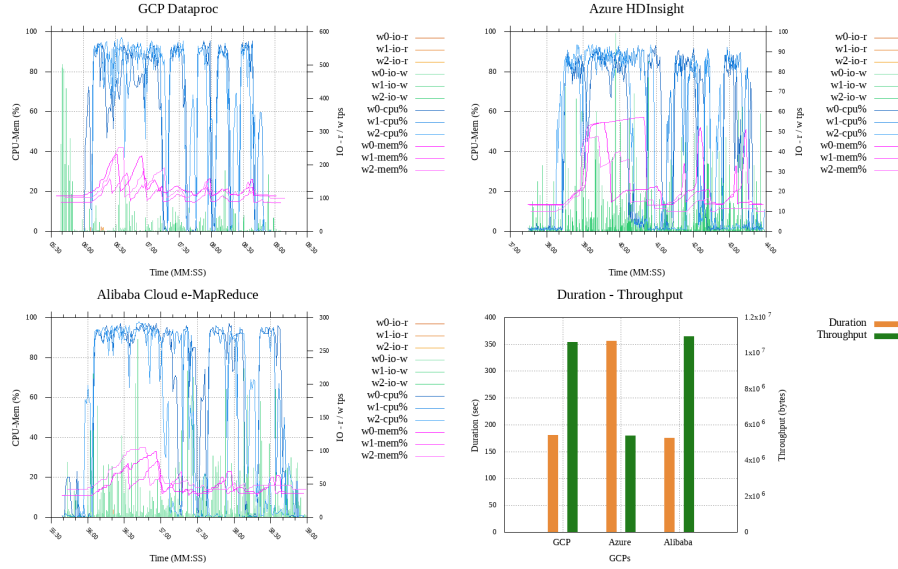


Figure 18: UC1 - Join (Gigantic; USERVISITS: 100,000,000 PAGES: 12,000,000)

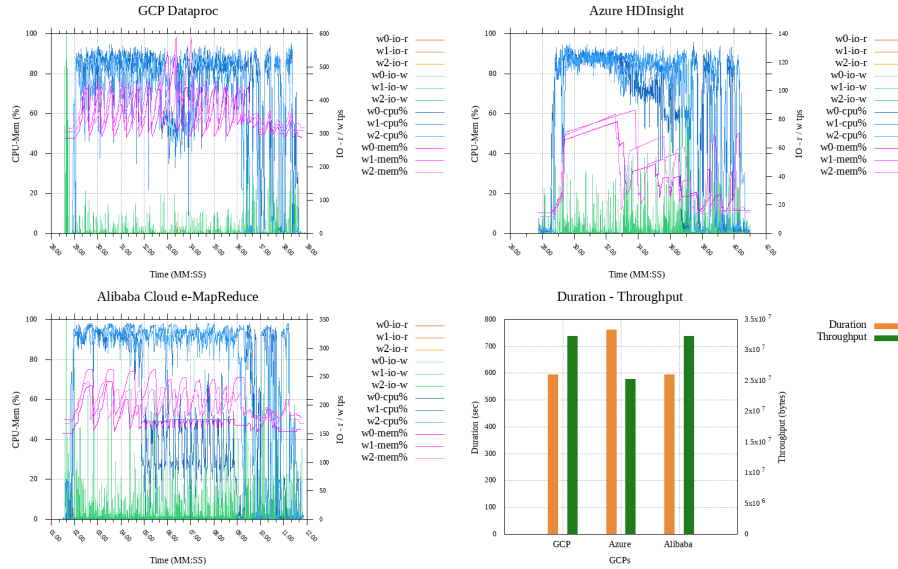


Figure 19: UC1 - Aggregation (Huge; USERVISITS: 10,000,000 PAGES: 1,200,000)

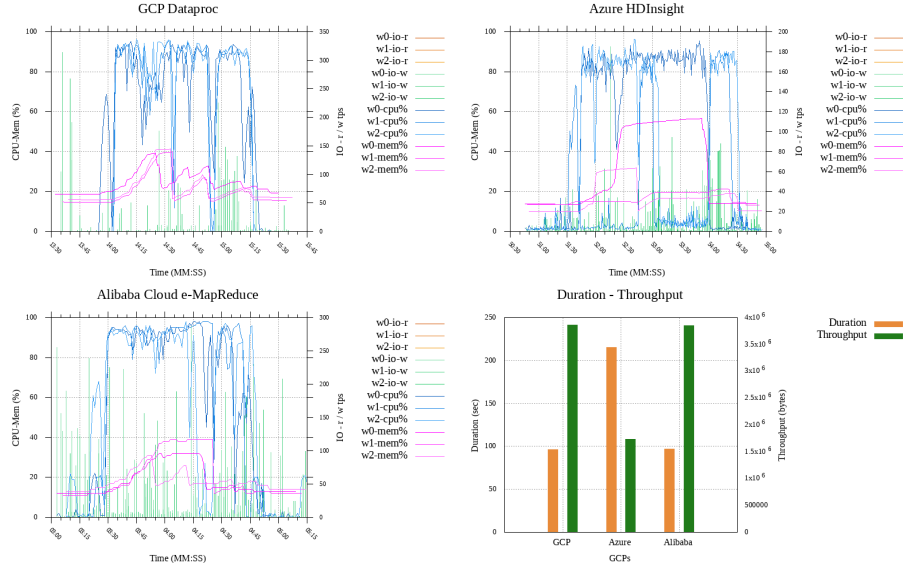


Figure 20: UC1 - Aggregation (Gigantic; USERVISITS: 100,000,000 PAGES: 12,000,000)

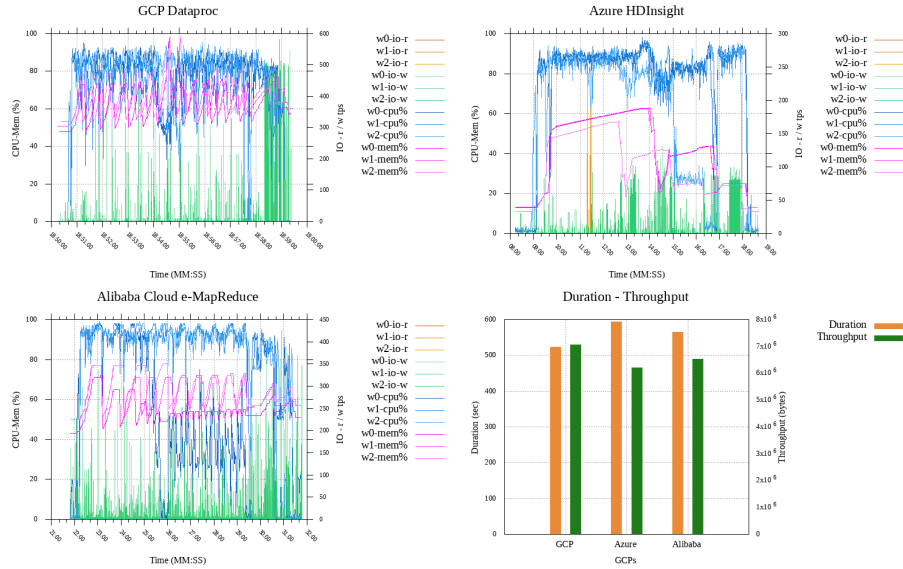


Figure 21: UC1 - Bayes (Huge; PAGES: 500,000 CLASSES: 100 NGRAMS: 2)

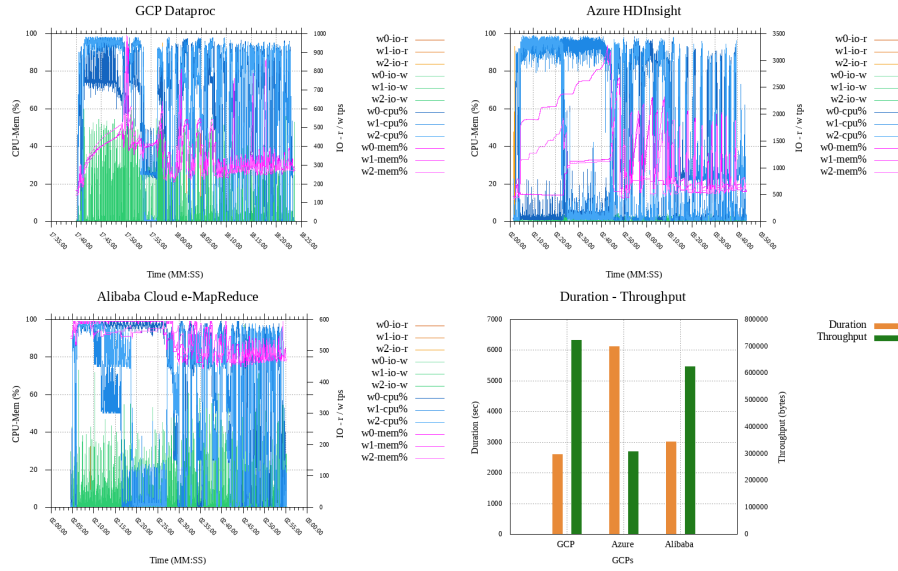


Figure 22: UC1 - Bayes (Gigantic; PAGES: 1,000,000 CLASSES: 100 NGRAMS: 2)

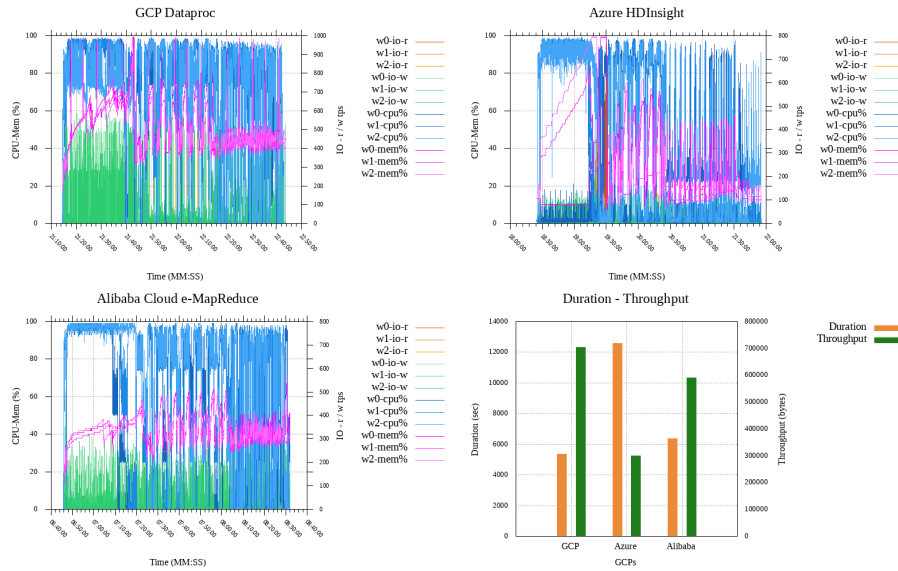


Figure 23: UC1 - Kmeans (Huge; CLUSTERS: 5 DIMENSIONS: 20 SAMPLES: 100,000,000
SAMP PER INPUT: 20,000,000 MAX IT: 5 K: 10 CONVERGEDIST: 0.5)

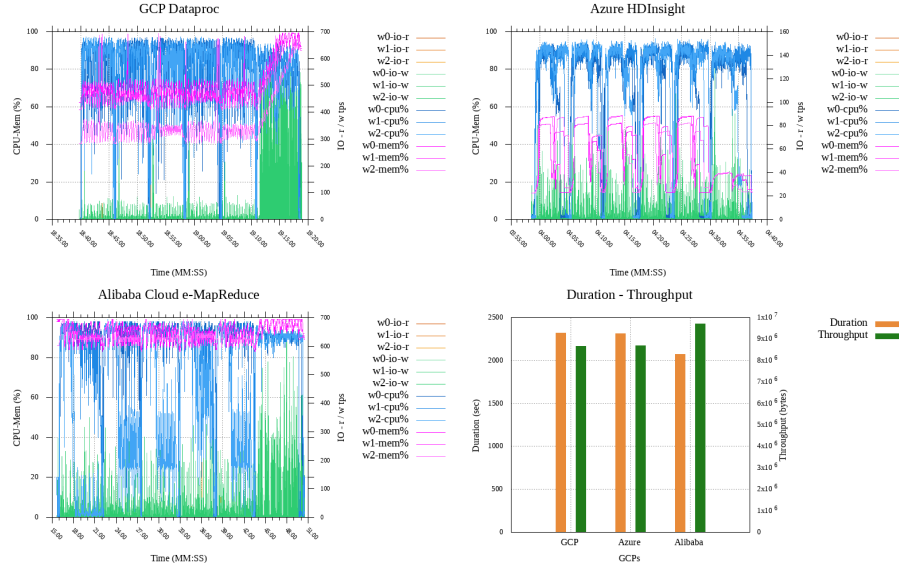


Figure 24: UC1 - Kmeans (Gigantic; CLUSTERS: 5 DIMENSIONS: 20 SAMPLES: 200,000,000 SAMP PER INPUT: 40,000,000 MAX IT: 5 K: 10 CONVERGEDIST: 0.5)

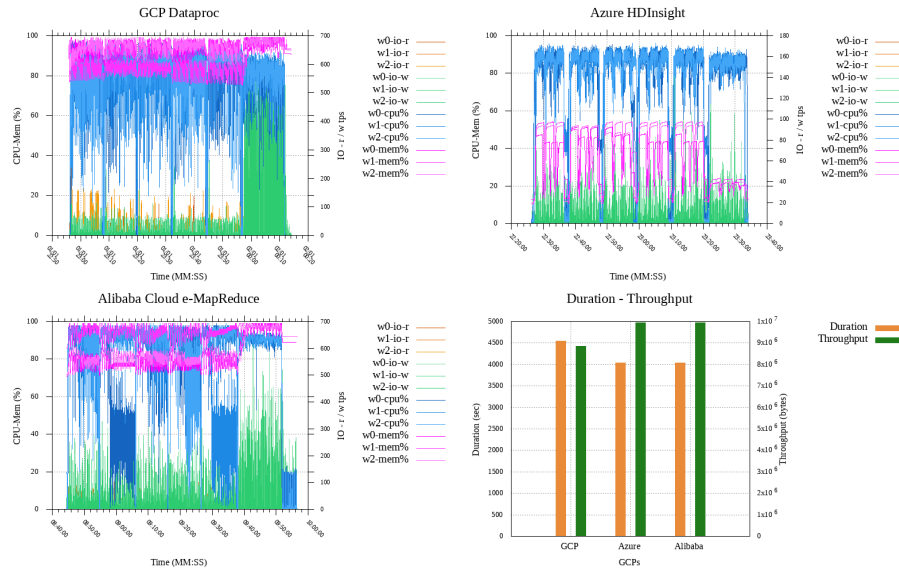


Figure 25: UC1 - Pagerank (Huge; PAGES: 5,000,000 NUM ITERATIONS: 3 BLOCK: 0
BLOCK WIDTH: 16)

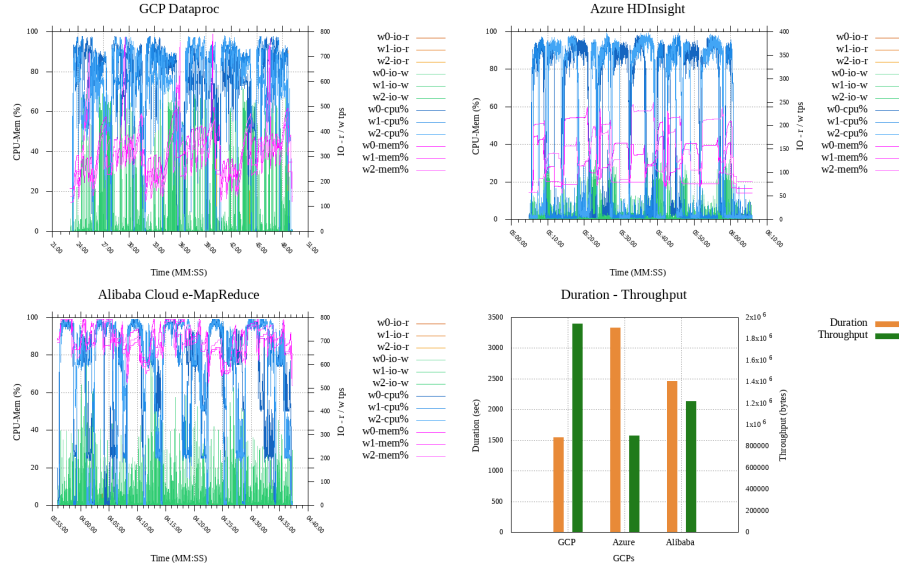


Figure 26: UC1 - Pagerank (Gigantic; PAGES: 30,000,000 NUM ITERATIONS: 3 BLOCK:
0 BLOCK WIDTH: 16)

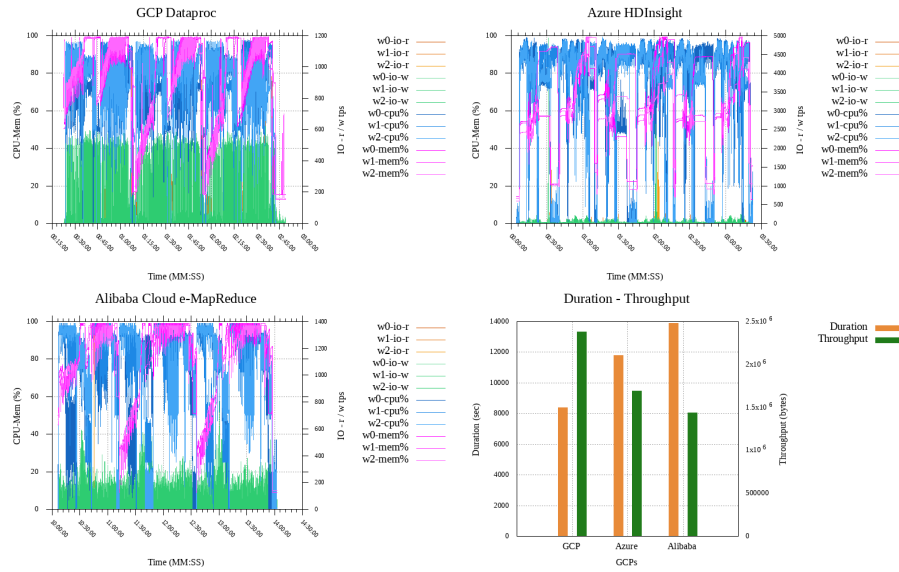


Figure 27: Use Case 1 - Overview to system utilization in data scale huge

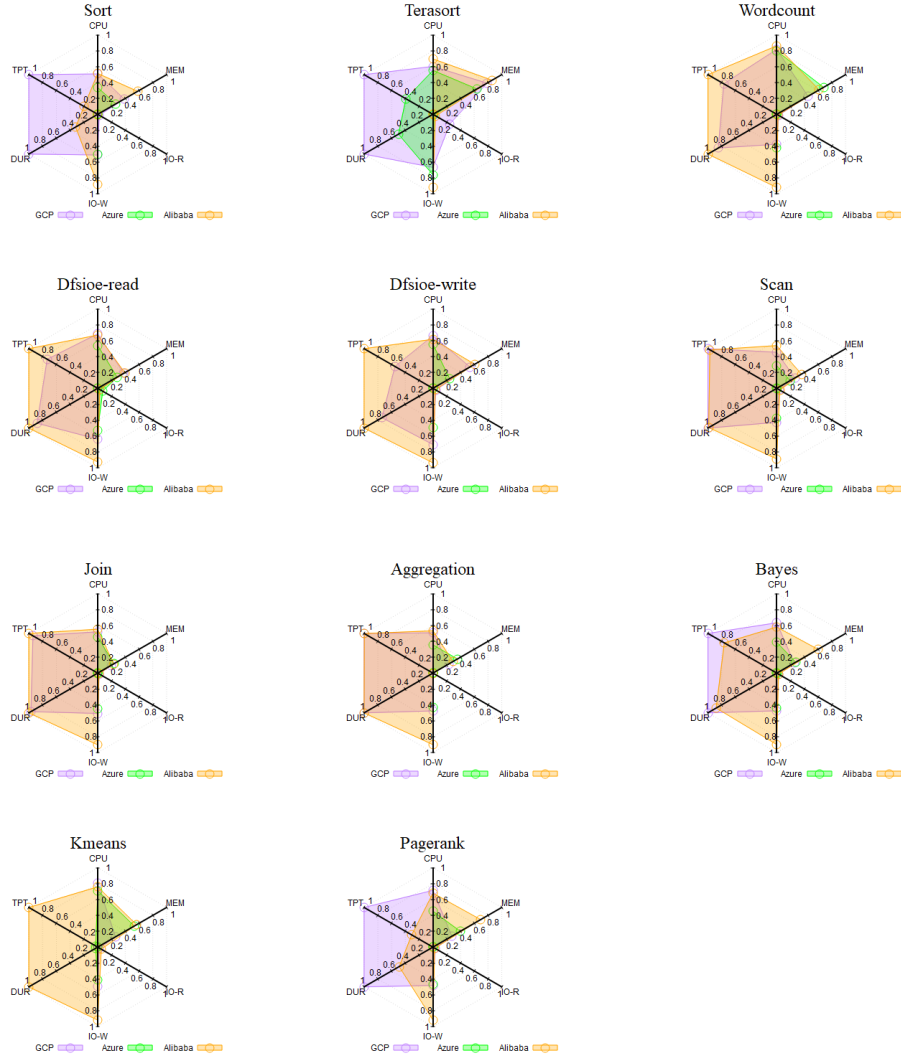
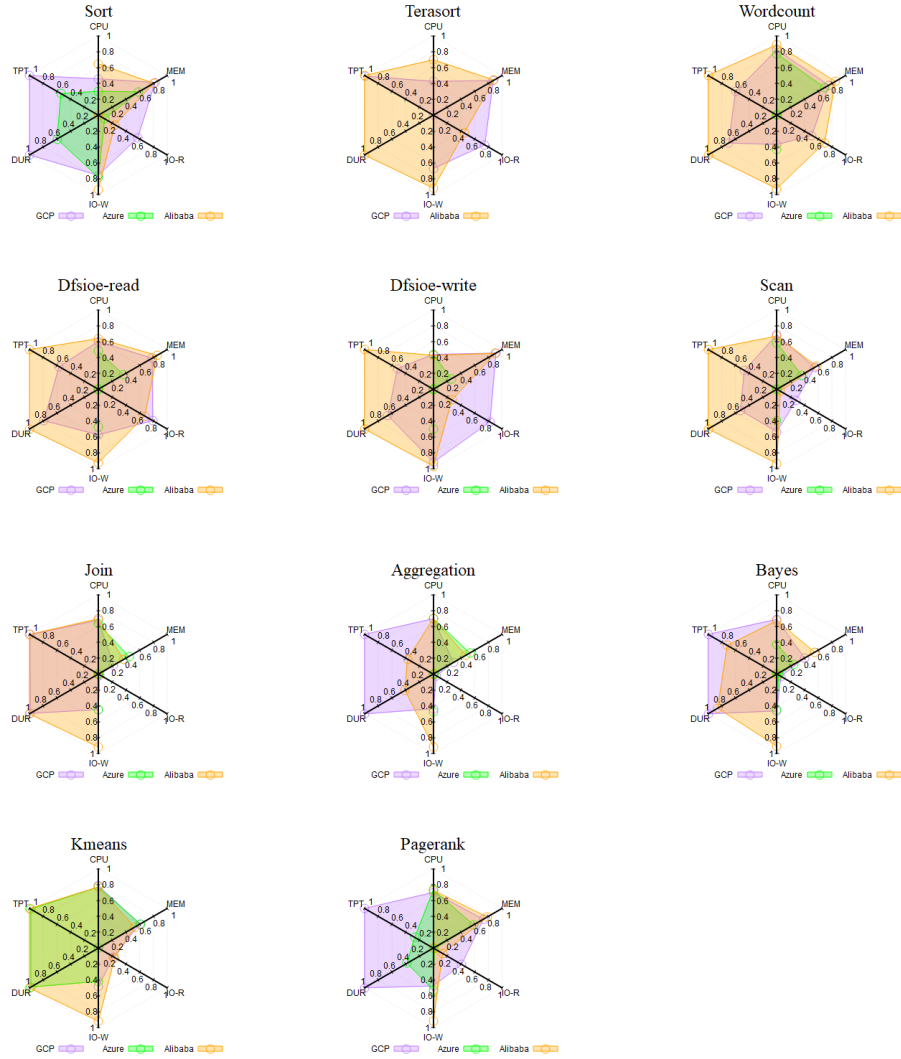


Figure 28: Use Case 1 - Overview to system utilization in data scale gigantic



4.2. Use Case 2 Results

In this section, we inspect the utilization behavior of I/O transfers per second
565 for Sort and CPU utilization behavior for Wordcount.

Sort - Tiny, Small, Large, Huge, Gigantic (Figures 29, 30, 31, 32, 33). In the data scales tiny and small, I/O transfers of the managed systems remain within similar boundaries, reaching peak points at about 80 tps for GCP, 30 tps to 35 tps for Azure, and about 200 tps for Alibaba, reaching the highest throughput
570 thus shortest duration in lighter data scales. At 328.5 MB input data size, which corresponds to the large data scale, GCP and Alibaba perform with an equal duration of 42 seconds, GCP utilizes I/O peaking at about 110 tps, whereas Alibaba scores peak at about 189 tps. Azure’s I/O utilization peaks at about 80 tps and a duration of 81 seconds. In the huge data scale with 3.28 GB
575 data, GCP starts utilizing I/O at instances with about 500 tps, thus performing the highest throughput projected in 70 seconds. Azure shows a condensed I/O transfer load at around 80 tps, peaking at 140 tps, and results in 140 seconds. Switching to the huge data scale with 3.2 GB, GCP’s I/O transfers push the upper limits at about 500 tps in the reduce phase, thus performing 47MB/s
580 throughput and a duration of 70 seconds; Azure’s I/O load draws condense 80 tps behavior peaking at about 140 tps, and results in 23MB/s duration and 141 seconds duration. Alibaba’s I/O load varies at about 100 tps to 450 tps, an approx. 31MB/s throughput brings 107 seconds response time. In the gigantic data scale, an input size of 32GB, GCP remains the level of 500 tps in a more
585 condensed way, peaking at about 1000 tps in I/O-reads, an approx. 47 MB/s throughput and 694 seconds of response time. At this scale, Azure improves its performance greatly, even though I/O utilization remains between about 75 tps to 150 tps, carrying out a throughput of 47MB/s, slightly less than GCP, results in a close follower of the best performance duration, by 699 seconds.
590 Alibaba’s I/O activity hits a peak at about 800 tps and approx. 400 to 500 tps in reduce phase, 37MB/s throughput results in 883 seconds response time.

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

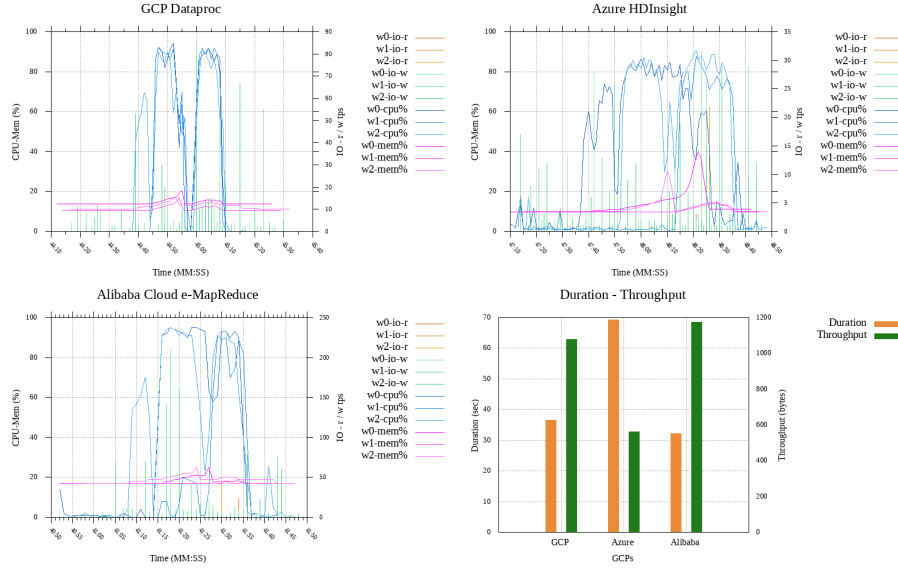


Figure 30: UC2 - Sort (Small; 3.2 MB)

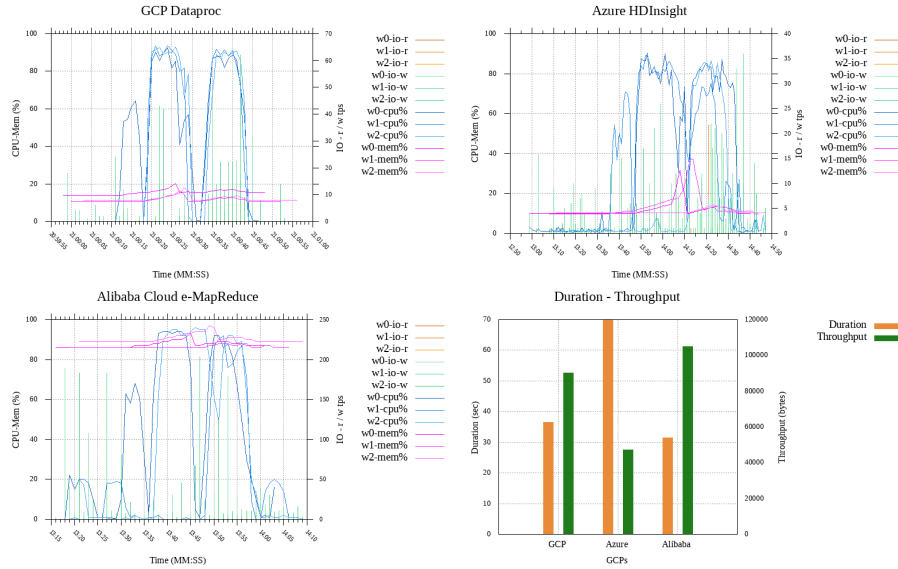


Figure 31: UC2 - Sort (Large; 320 MB)

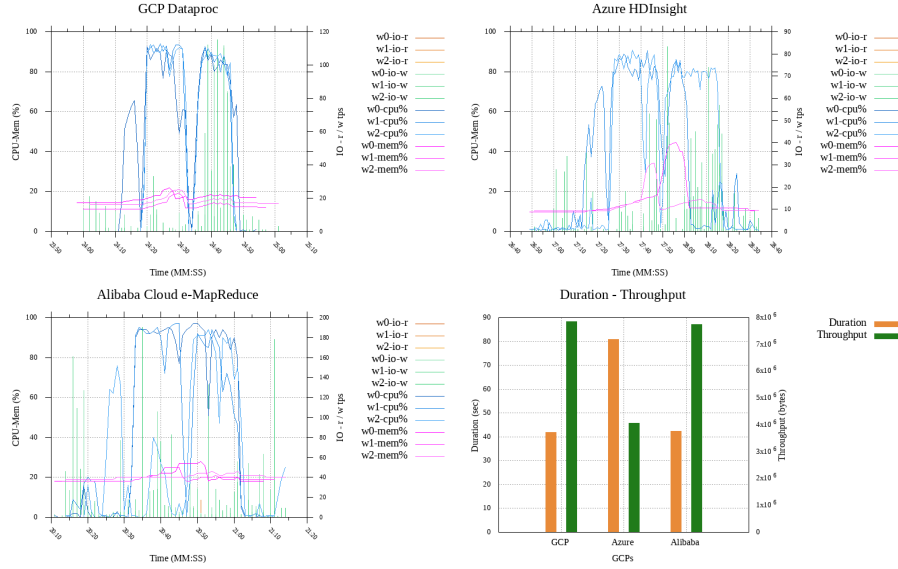
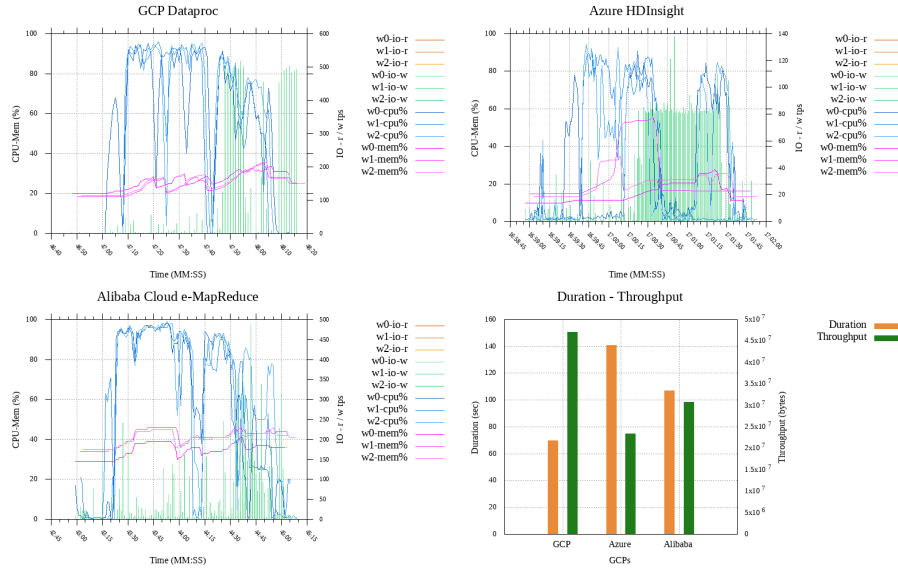


Figure 32: UC2 - Sort (Huge; 3.2 GB)



Wordcount - Tiny, Small, Large, Huge, Gigantic (Figures 34, 35, 36, 37, 38).

Following the system utilization of the managed systems along with the plot sequence, the high CPU load of Wordcount is noticeable. The longer the duration, the condenser its high load visualization gets. The CPU utilization behavior across different managed services moves within 80% and below 100%. However, in Azure, one worker node’s CPU load somewhat stays in a very low range in the data scales Tiny, Small, and Large. Low and mid-level memory consumption across CSPs remains similar between CSPs. In I/O, Alibaba depicts a higher level of transfers per second than its conjugates, which probably yields the highest throughput hence shortest response times for e-MapReduce in all given data scales.

Overviews of Sort and Wordcount utilizations along with data scales Tiny, Small, Large, Huge, and Gigantic (Figure 39 and Figure 40). The spider plot visualizations withdraw the time dimension and place utilization averages per cluster into the axes. The performances depicted in Sort benchmark show Azure’s back falling performances in Tiny, Small, Large, and Huge data scales and its enhanced performance in the gigantic data scale. In the Wordcount benchmark, where likely CPU utilization among the services occurs, the compressed I/O average load becomes the determining aspect of higher throughput and better performance for Alibaba.

5. Discussion

Limitations, workarounds, failures. One major limitation of the study has been the high-cost of benchmark executions’ total running time charged in a pay-per-use manner, which led the conductors to keep benchmarks at only one successful execution for each workload deviating from the best practice; executing each benchmark thrice and taking the average would have provided more stable outcomes, especially in cases where there occurs a very tiny difference between successive performances of respective providers. Another result of the limitation mentioned above is that we also left off the largest predefined data scale

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

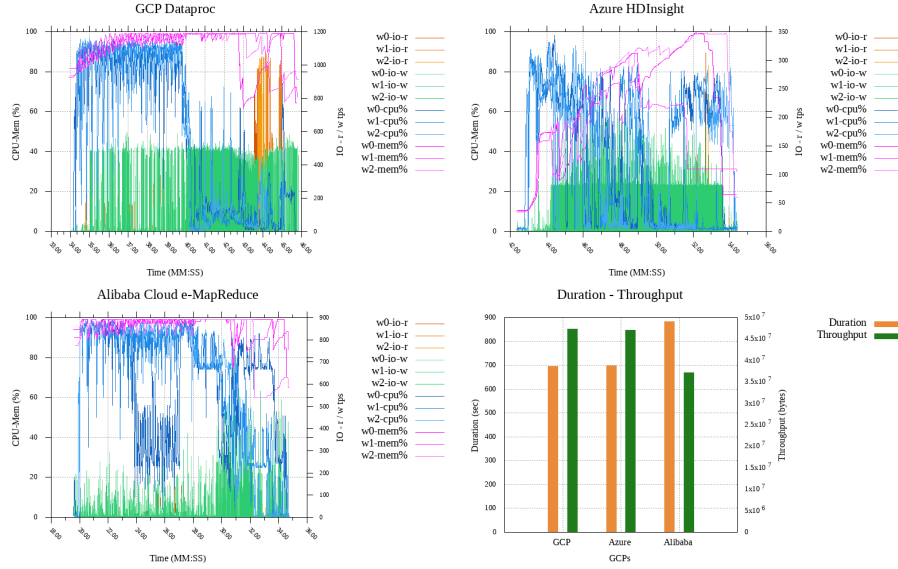


Figure 34: UC2 - Wordcount (Tiny; 32 KB)

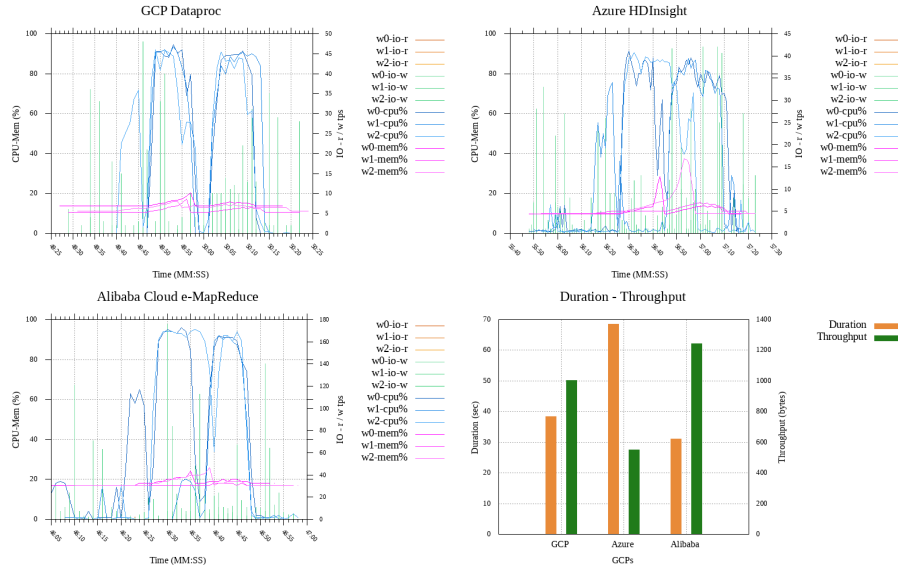


Figure 35: UC2 - Wordcount (Small; 320 MB)

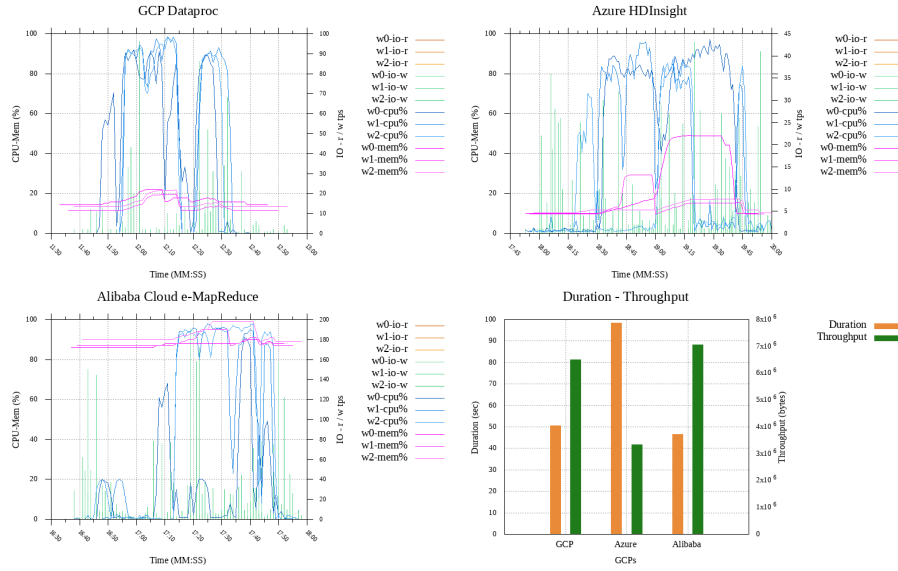


Figure 36: UC2 - Wordcount (Large; 3.2 GB)

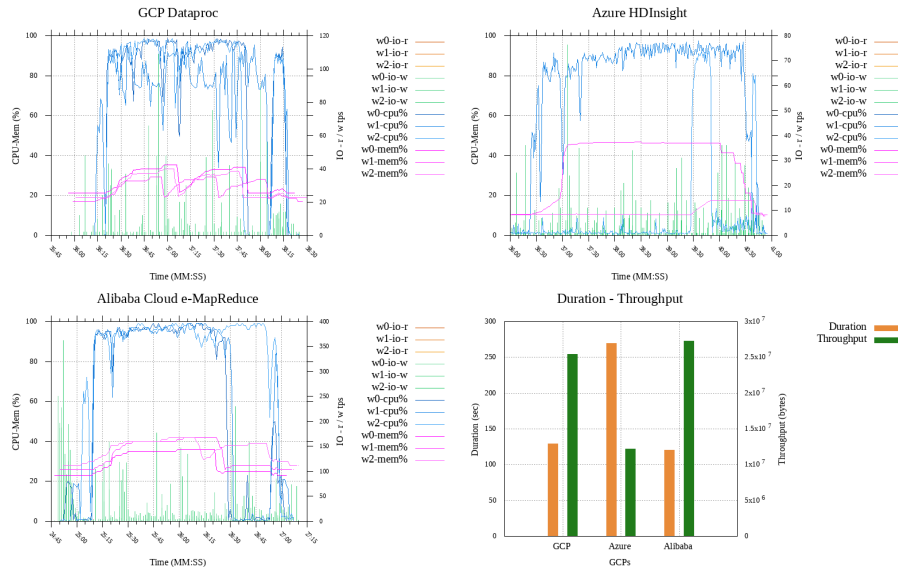


Figure 37: UC2 - Wordcount (Huge; 32 GB)

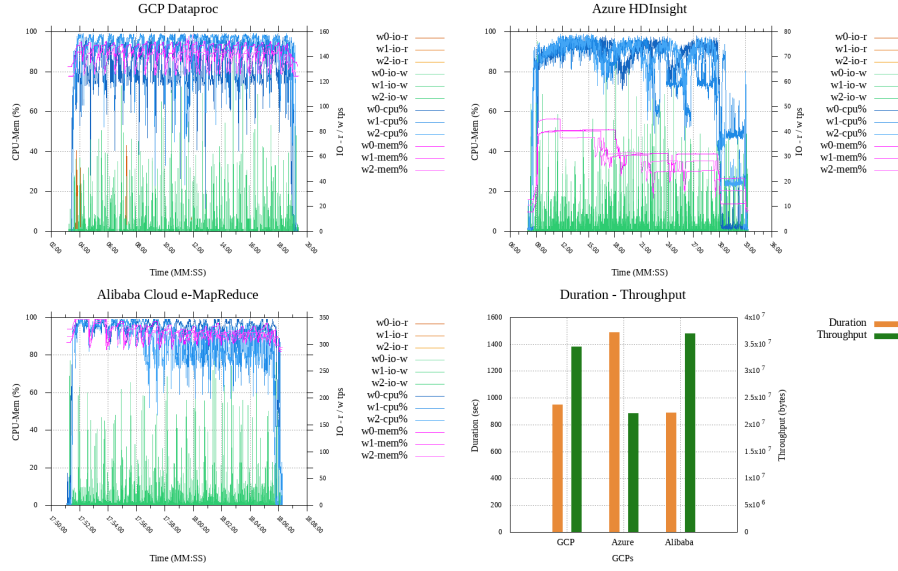


Figure 38: UC2 - Wordcount (Gigantic; 320 GB)

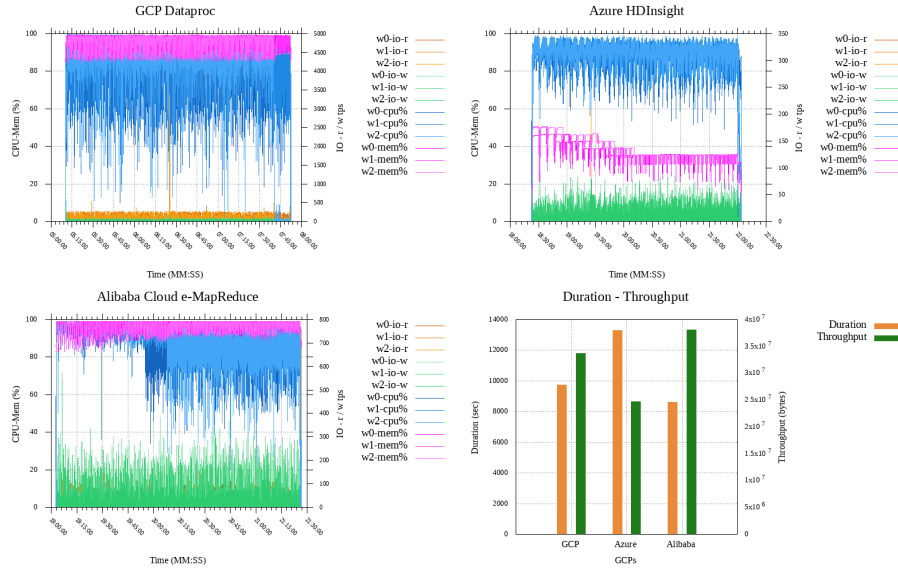


Figure 39: Use Case 2 - Sort performances along with data scales

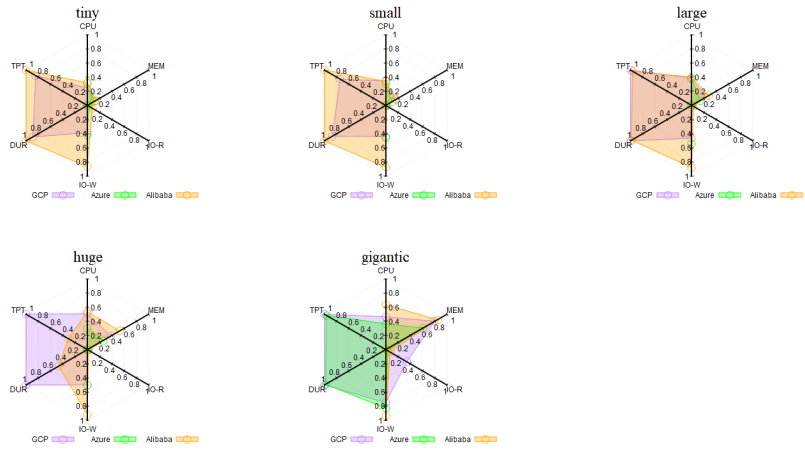
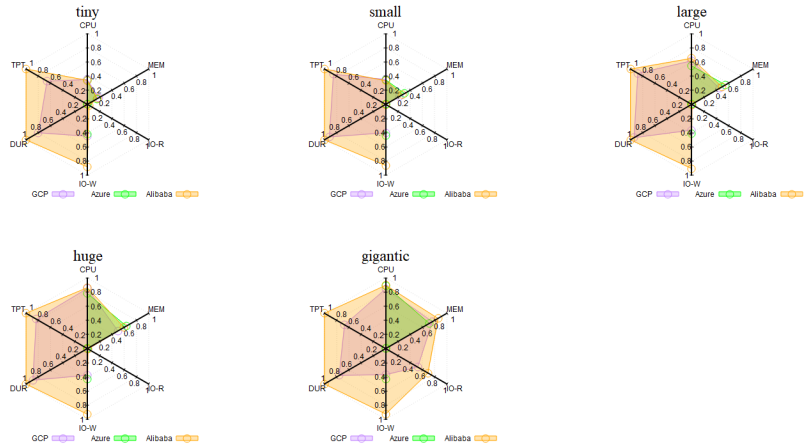


Figure 40: Use Case 2 - Wordcount performances along with data scales



of HiBench, namely Bigdata. This benchmark’s execution would have provided performance outputs in a vast data scale like 300 GB for Sort, 1.6 TB for Word-count, and 12.000.000 pages by 1.000.000.000 user visits for Aggregation, to name but three. Hence, we recommend the study to be understood in terms
625 of an attempt to bring more clarity to black-box nature managed Hadoop proposals’ performance behavior using resource utilization dynamics and as the managed services come out-of-the-box without any performance tweaking configurations applied. We do not recommend the study to be understood as a grading board for respective managed Hadoop systems’ business values.

630 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 e-Mapreduce comprises a ready-made Hive hook triggering a Java file to run post-execution transactions for other services within the package. However, this
635 configuration prevented HiBench from starting with respective benchmarks’ execution since the HiBench based Hive engine does not include the jar file mentioned above defined for e-MapReduce’s specific environment. A workaround attempt, copying the related jar file to an appropriate directory within HiBench, made the jar file available; however, this time, HiBench’s Hive engine
640 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 Alibaba e-MapReduce’s UI management console apparently solved this issue and enabled HiBench’s SQL workloads to run, but its impact on the respective performance values remains unknown, hence the need to annotate it
645 here. With GCP and Azure, issues of this kind did not occur.

In the Azure environment, the Terasort benchmark running in data scale gigantic failed to complete in all three attempts we conducted where about 20% of map operations were reached. Showing distinction to GCP and Alibaba where the end-user is liberated in choosing HDFS or the respective provider’s cloud
650 storage service as the cluster’s file system, Azure obligates the user to go with WASB blob storage among other Azure storage services with a promise to Peta-

scale. However, after inspecting the failure, it turned out that the exception is not due to the WASB. 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 to store intermediary results and fail to allocate free space HDFS reaching specific levels of maps operators. The conductors marked this failure as a structural bottleneck since all end users running Terasort operation 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 it as incomplete. Figure 8 displays system resource utilization on Azure, including the point where the failure occurs.

Along with the resource utilization vs. performance plots both for Use Case 1 and Use Case 2, it is easily recognizable how likely architectures, namely GCP Dataproc and Alibaba Cloud e-MapReduce, both leveraging a rather conventional Hadoop installation and infrastructure, follow a similar pattern in utilization over time whereas a different architectural approach distinct in its performance behavior as in Azure’s HDP based HDInsight. To better understand the relative performance dynamics, we gathered and processed the CSPs’ HiBench performances in duration (seconds) in Table 5 and Table 6. Column ”Benchmark” refers to the respective benchmark executed, column ”First” contains the best-performing provider, column ”Second” points out the provider service with the second shortest duration, column ”Third” lists the provider of the longest duration for the respective benchmark. Columns ”-Perf%” display the relatively low performance of the respective provider’s service listed to their left with respect to the best performing service’s value. *Relative Performance Rate = (Shortest duration - Current duration) / Shortest duration*. Relative Performance Rates indicate that in overall, alike architectures are close in their results and competitive like performing the same duration in UC1-Aggregation or as close as 0.17% time difference. The farthest distance between GCP and Alibaba is 65.97% in UC2-PageRank.

On the other hand, the Relative Performance Rates in Table 5 and Table 6 depict numerous samples where Azure’s performance falls back the best duration

Table 5: Use Case 1 Relative Performance Rates
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%

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 Relative Performance Rates

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

by more than 100%, meaning that performing at least two times slower in the respective benchmarks. A possible reason for the -170.20% and -134.16% low performances in UC1’s Dfsioe’s and -185.76% and -80.57% low performances in UC2’s Dfsioe’s might be the replacement of the native Hadoop filesystem by Azure blob. HDI performances for SQL benchmarks (Scan, Join, and Aggregation) move between -103% and -121% in UC1, whereas in UC2, a better performance moving along -13% and -28% occurs. Another contradictory case for Azure is in its ML performances, especially in gigantic data scale where for Bayes it low performs at -135.31% whereas for Kmeans it almost scores the best performance, getting the second place by a performance slightly less than -0.20% with respect to the shortest duration. Why is it so? In our attempt to understand this situation, we delved into the execution plans, generated initially by Hadoop during execution and further captured and stored by HiBench inside the Reports folder. Table 7 and Table 8 comprise the allocated map and reduce slots by their respective Hadoop cluster for the relevant benchmarks. It turns out that GCP and Alibaba Cloud’s managed services show themselves more capable in resource allocation to respective applications, hence competi-

Table 7: Allocated map and reduce slots in SQL benchmarks of Use Case 1

Benchmark	GCP		Azure		Alibaba		Ref.
	Maps	Reduces	Maps	Reduces	Maps	Reduces	
Scan * (h)	24	–	12	–	24	–	<i>Fig.15</i>
Scan * (g)	144	–	36	–	144	–	<i>Fig.16</i>
Join (h)	60	25	48	25	60	25	<i>Fig.17</i>
Join (g)	180	25	72	25	180	25	<i>Fig.18</i>
Aggregation (h)	24	12	12	12	24	12	<i>Fig.19</i>
Aggregation (g)	144	12	36	12	144	12	<i>Fig.20</i>

(h): huge, (g): gigantic
 * No reduce operation

700 tive to each other. Another instance of the relatively low-performance Azure’s,
 for Bayes benchmark in Use Case 1, might point to a structural shortcoming
 where the execution logs point Java heap space errors, which cause the job
 to restart nine times, thus prolonging the overall duration of the benchmark,
 and resulting in termination at the end. Azure’s default configurations some-
 705 what lack to provide free resources enough to stay in the competitionn and this
 might be resolved with performance tuning, though. Having sufficient resources,
 Azure almost outbeats its conjugates in K-means, leveraging the very same Ma-
 hout engine with Bayes benchmark. Scoring its highest (-0.20%) and lowest
 (-135.31%) relative performances in the same category (ML) is somewhat con-
 710 tradictory but reasonable when considering the low performance as a structural
 pre-configuration issue.

6. Conclusion and Future Work

Managed Hadoop systems are by CSPs provided cloud service proposals com-
 prising pre-installed and pre-configured Hadoop clusters in order to eliminate
 715 the hard work on planning and installing Hadoop clusters manually on VM in-
 stances, 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

Table 8: Allocated map and reduce slots in Use Case 2

Benchmark	GCP		Azure		Alibaba		Ref.
	Maps	Reduces	Maps	Reduces	Maps	Reduces	
Sort (t)	11	12	12	12	11	12	<i>Fig.29</i>
Sort (s)	11	12	12	12	11	12	<i>Fig.30</i>
Sort (l)	12	11	12	12	11	11	<i>Fig.31</i>
Sort (h)	24	12	12	12	23	12	<i>Fig.32</i>
Sort (g)	251	12	60	12	252	12	<i>Fig.33</i>
Wordcount (t)	11	12	12	12	11	12	<i>Fig.34</i>
Wordcount (s)	12	11	12	12	12	11	<i>Fig.35</i>
Wordcount (l)	24	11	12	12	23	12	<i>Fig.36</i>
Wordcount (h)	250	11	60	12	252	12	<i>Fig.37</i>
Wordcount (g)	2447	12	612	12	2447	11	<i>Fig.38</i>

(h): huge, (g): gigantic

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 the same geographical location as the region and by promise apparently the 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 line chart and spider plot visualizations leveraging aforementioned outputs spitted out high-resolution scans of respective executions over time and a comparison of cluster-based performance averages. Based on the performance results, we calculated relative performance rates for each CSP's managed system related to the best performing one to get a better view of how far the respective performances are distinct from each other. In the final stage,

for selected benchmark performances, we gathered the numbers of the allocated
735 map and reduce slots by vendors' managed Hadoop systems. There is a strong
correlation between sufficiently allocated map slots and the performance out-
put; hence it appears that default pre-configuration settings somewhat do not
let sufficient resources to utilize, resulting in low performance for a vendor.

The results yield that Hadoop PaaS offerings by vendors' promise do not
740 necessarily utilize system resources in similar ways hence may highly distinct
in some cases. We assume that CSP's pre-configuration choices impact per-
formance results even with the same hardware settings. The end-users shall
be aware of their use cases' requirements, truly understand the default config-
uration proposals and their respective outcome, and possess the know-how to
745 improve respective vendors' default configurations.

Based on the gained experience, the study might be 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, also putting
750 other cloud vendors into the scope. Micro cases like benchmarking before and
after configuration tweaks could be possible approaches to go.

[1] World Internet Users Statistics and 2020 World Population Stats.

URL <https://www.internetworldstats.com/stats.htm>

[2] Apache Hadoop.

755 URL <https://hadoop.apache.org/>

[3] S. Ghemawat, H. Gobioff, S.-T. Leung, The Google file system, in: Proceed-
ings 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,
760 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.
765 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/>
- 770 [8] Dataproc.
URL <https://cloud.google.com/dataproc>
- [9] Compute Engine: Virtual Machines (VMs).
URL <https://cloud.google.com/compute>
- [10] Cloud Storage.
775 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/>
780
- [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>
- 785 [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>

- 790 [16] OSS (Object Storage Service): Secure Cloud Storage.
URL `product/oss`
- [17] Intel-bigdata/HiBench, original-date: 2012-06-12T07:56:57Z (Jan. 2021).
URL `https://github.com/Intel-bigdata/HiBench`
- [18] 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 (ICDEW 2010), 2010, pp. 41–51. doi:10.1109/ICDEW.2010.5452747.
- 795 [19] Release HiBench-7.1.1 Intel-bigdata/HiBench.
URL `/Intel-bigdata/HiBench/releases/tag/v7.1.1`
- 800 [20] N. Poggi, A. Montero, D. Carrera, Characterizing bigbench queries, hive, and spark in multi-cloud environments, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 10661 LNCS (2018) 55–74. doi:10.1007/978-3-319-72401-0_5.
- 805 [21] Y. Samadi, M. Zbakh, C. Taddonki, Performance comparison between hadoop and spark frameworks using HiBench benchmarks, Concurrency Computation 30 (12). doi:10.1002/cpe.4367.
- [22] H. Ahn, H. Kim, W. You, Performance Study of Spark on YARN Cluster Using HiBench, 2018. doi:10.1109/ICCE-ASIA.2018.8552137.
- 810 [23] S. Han, W. Choi, R. Muwafiq, Y. Nah, Impact of memory size on bigdata processing based on hadoop and spark, Vol. 2017-January, 2017, pp. 275–280. doi:10.1145/3129676.3129688.
- [24] 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.
- 815

[25] Gartner Reprint.

URL <https://www.gartner.com/doc/reprints?id=1-1ZDZDMTF&ct=200703&st=sb>