Comparative Performance Evaluation of Hadoop on PaaS Proposals by Leveraging HiBench

Elsevier[1]

Radarweg 29, Amsterdam

Elsevier Inc[a,b], *Global Customer Service*[b,*]

[a]*1600 John F Kennedy Boulevard, Philadelphia*
[b]*360 Park Avenue South, New York*

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 beneficiants 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

☆Fully documented templates are available in the elsarticle package on CTAN.
*Corresponding author
Email address: `support@elsevier.com` (Global Customer Service)
URL: `www.elsevier.com` (Elsevier Inc)
[1]Since 1880.

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 performance dynamics, hence the benefits by leveraging them. In this 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 has become an inevitable aspect for enterprises and academia of the information era to deal with. As the global internet access rate covers a weighing majority of the global human population, 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 splitting into diverse 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. Distributed frameworks for storage and computation sparked up first by search engines were inherited and furtherly developed 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.

In parallel, the commercialization of Cloud Computing in the early 2000's 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 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 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 simply determining some settings like hardware specs and node numbers prior the 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. CSP apply behind-the-scenes tweaks in terms of reaching better performance results on selected approaches like memory intensive or compute intensive applications.

In this study we put three CSP providers' managed Hadoop services in focus in terms of performance evaluation comparison: GCP Dataproc, Azure HDInsight, and Alibaba Cloud e-MapReduce, each recognized in Gartner's 2020 report in leading or niche section. Bound by availability of their offered hardware and software options we selected by providers' promise apparently same or close settings. Without any performance tweak operation with their settings for any performance optimization on the respective managed systems after installation we immediately executed several workloads from HiBench's micro, sql, ml, and websearch categories. For a more clear understanding of the benchmark outputs, during the benchmark execution we collected system utilization records on each worker node of the cluster. The results yield that Hadoop PaaS offerings by vendor's promise side perform and system utilizations may highly vary among CSPs.

Functionality. The Elsevier article class is based on the standard article class and supports almost all of the functionality of that class. In addition, it features commands and options to format the

- document style
- baselineskip
- front matter

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. [52] Characterizing BigBench

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

Samadi et al. 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. [54] 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. [55] 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 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 conducters as Spark show a 5-10%, and 15% maximum improvement in processing time.

Ivanov et al. [56] 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 Clouderas Distribution of Hadoop (CDH) comprising core Hadoop elements HDFS and YARN integrated with elements belonging to 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 infrastructure configuration. For both installations, default system parameters have been left with their defaults. HiBenchs three chosen workloads (CPU-bound 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 this study differs in approach related to other studies using YCSB benchmark suite. HiBenchs results confirm the latters output as well.

3. Method

CSP configurations in Table 1

4. Results

Table 2 and Table 3 summarize HiBench benchmark execution outputs.

Analysis. 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. During benchmark runtime resource utilization on worker nodes have been captured. The resulting multiplots are suggested to be read as follows: Top-left, top-right, and bottom-left plots represent CPU (user%), Memory, and IO utilization 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. Even though the coloring convention might sound confusing, it gives a clear overview in terms of resource utilization 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 or IO-write transfers in byte per second. The bottom-right plot represents the comparative benchmark performance outputs of the respective CSP. 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".

USE CASE 1:

Sort - Huge. Figure1; CPU utilization in GCP and Alibaba condence 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. Figure2; switching the data scale for Sort benchmark to gigantic, GCP's processor load condences around 80% - 95% where its memory load 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. Figure3; 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 falls back in embarking Throughput and resulting in 1054 seconds response time.

Terasort - Gigantic. Figure4; 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 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, 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. Figure5; GCP performing processor utilization of 75% to 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 hance the 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. Figure6; 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 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 varyimg between 40%s and close to 100%s, 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. Figure7; 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 condences 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%s.

Dfsioe-read - Gigantic. Figure8;

Dfsioe-write - Huge. Figure9;

Dfsioe-write - Gigantic. Figure10;

Scan - Huge. Figure11;

Scan - Gigantic. Figure12;

Join - Huge. Figure13;

Join - Gigantic. Figure14;

Aggregation - Huge. Figure15;

Aggregation - Gigantic. Figure16;

Bayes - Huge. Figure17;

Bayes - Gigantic. Figure18;

Kmeans - Huge. Figure19;

Kmeans - Gigantic. Figure20;

Pagerank - Huge. Figure21;

Pagerank - Gigantic. Figure22;

USE CASE 2:

Sort - Tiny. Figure23;

Sort - Small. Figure24;

Sort - Large. Figure25;

Sort - Huge. Figure26;

Sort - Gigantic. Figure27;

Wordcount - Tiny. Figure28;

Wordcount - Small. Figure29;

Wordcount - Large. Figure30;

Wordcount - Huge. Figure31;

Wordcount - Gigantic. Figure32;

Wordcount results in scale

5. Discussion

6. Front matter

The author names and affiliations could be formatted in two ways:

(1) Group the authors per affiliation.

(2) Use footnotes to indicate the affiliations.

See the front matter of this document for examples. You are recommended to conform your choice to the journal you are submitting to.

7. Bibliography styles

There are various bibliography styles available. You can select the style of your choice in the preamble of this document. These styles are Elsevier styles based on standard styles like Harvard and Vancouver. Please use BibTEX to generate your bibliography and include DOIs whenever available.

Here are two sample references: [1, 2].

[1] R. Feynman, F. Vernon Jr., The theory of a general quantum system interacting with a linear dissipative system, Annals of Physics 24 (1963) 118–173. `doi:10.1016/0003-4916(63)90068-X`.

[2] P. Dirac, The lorentz transformation and absolute time, Physica 19 (1-12) (1953) 888–896. `doi:10.1016/S0031-8914(53)80099-6`.

Table 1: 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	(Azure	2
Block size	128 MB	storage)	128 MB

Table 2: Use Case 1 benchmark outputs

Data Scale: Huge

		Dataproc		HDInsight		e-MapReduce	
Benchmark	IDS	$D_{(s)}$	$T_{(b/s)}$	$D_{(s)}$	$T_{(b/s)}$	$D_{(s)}$	$T_{(b/s)}$
Sort	3.28 GB	70	47110942	131	25076597	111	29419903
Terasort	32.00 GB	667	47988314	858	37277268	1054	30374710
Wordcount	32.85 GB	978	33594975	1470	22340906	889	36949719
Dfsioe-r	26.99 GB	294	91772636	662	40787869	245	110206431
Dfsioe-w	27.16 GB	379	71733304	658	41296567	281	96488258
Scan	2.01 GB	73	27629405	157	12830581	74	27193804
Join	1.92 GB	181	10614143	356	5390513	175	10950553
Aggregation	372.38 MB	97	3857328	215	1728400	97	3849353
Bayes	1.88 GB	2604	722498	6120	307708	3017	623692
Kmeans	20.08 GB	2321	8652662	2313	8683526	2070	9703348
Pagerank	2.99 GB	1544	1938541	3334	897807	2458	1217764

Data Scale: Gigantic

		Dataproc		HDInsight		e-MapReduce	
Benchmark	IDS	$D_{(s)}$	$T_{(b/s)}$	$D_{(s)}$	$T_{(b/s)}$	$D_{(s)}$	$T_{(b/s)}$
Sort	32.85 GB	715	45939699	787	41721261	896	36680251
Terasort	320.00 GB	9821	32582430	—(*)	—(*)	9660	33126513
Wordcount	328.49 GB	10131	32423997	13596	24159989	8671	37882290
Dfsioe-r	216.03 GB	915	236107099	1886	114538844	660	327294273
Dfsioe-w	217.33 GB	1347	161394226	1914	113574850	1060	205123746
Scan	20.10 GB	457	43964927	514	39085260	407	49378781
Join	19.19 GB	595	32268936	761	25240939	594	32319846
Aggregation	3.69 GB	523	7051835	594	6203608	565	6523381
Bayes	3.77 GB	5350	703332	12589	299143	6363	591341
Kmeans	40.16 GB	4541	8844971	4042	9935665	4034	9956003
Pagerank	19.93 GB	8371	2381276	11779	1692241	13893	1434711

(*) Incomplete benchmark execution due to insufficient HDFS disk space on HDInsight

Table 3: Use Case 2 benchmark outputs

		Dataproc		**HDInsight**		**e-MapReduce**	
Benchmark	IDS	$D_{(s)}$	$T_{(b/s)}$	$D_{(s)}$	$T_{(b/s)}$	$D_{(s)}$	$T_{(b/s)}$
Sort (t)	39.30 KB	36	1077	69	563	32	1173
Wordcount(t)	38.65 KB	38	1005	68	551	31	1242
Sort (s)	3.28 MB	36	90155	70	47090	31	104865
Wordcount (s)	348.29 MB	50	6508975	98	3337158	47	7055759
Sort (l)	328.50 MB	42	7860401	81	4065391	42	7741448
Wordcount (l)	3.28 GB	129	25448586	269	12195273	120	27269866
Sort (h)	3.28 GB	70	47077081	141	23355371	107	30693426
Wordcount (h)	32.85 GB	952	34512554	1487	22093900	888	36991074
Sort (g)	32.85 GB	694	47300549	699	47000056	883	37192377
Wordcount (g)	328.49 GB	9749	33696143	13286	24725537	8622	38100208

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

Figure 1: UC1 - Sort (Huge; 3.2 GB)

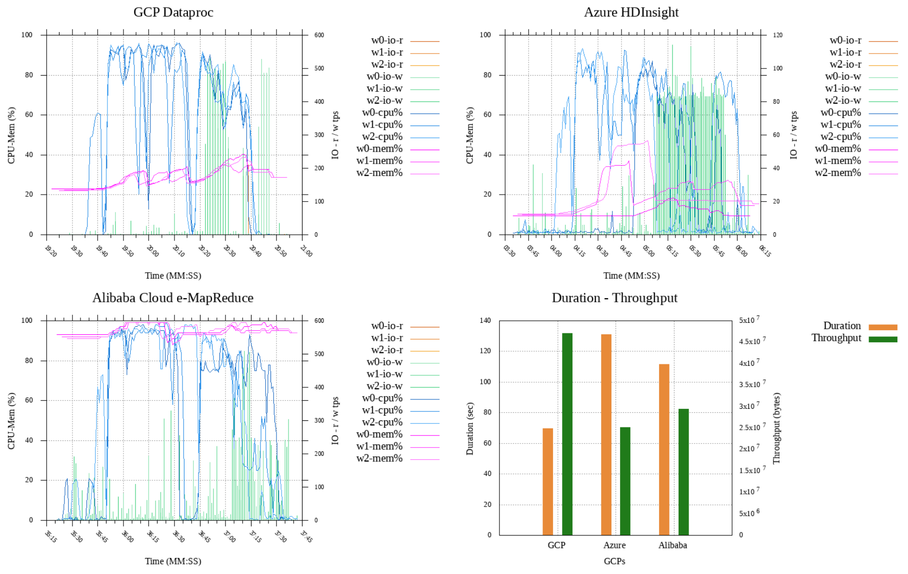


Figure 2: UC1 - Sort (Gigantic; 32 GB)

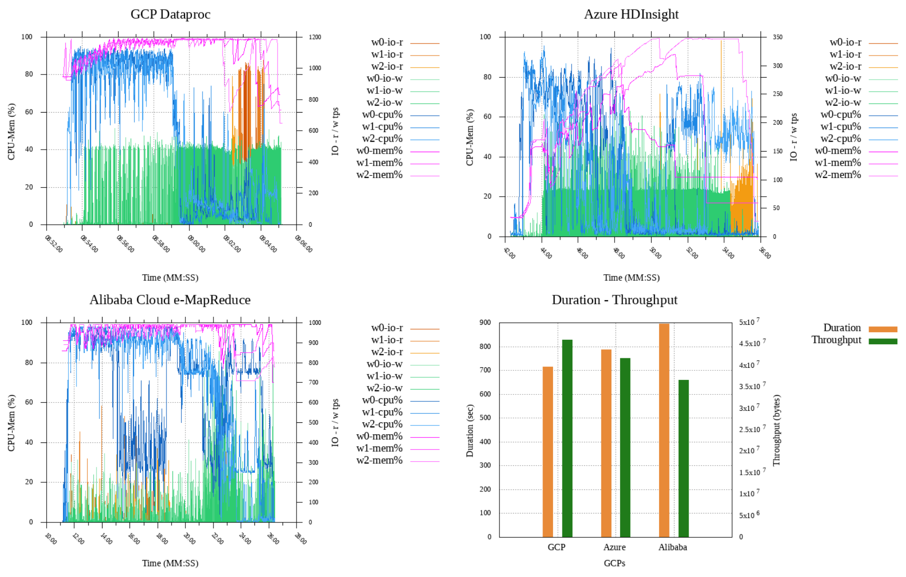


Figure 3: UC1 - Terasort (Huge; 320 MB)

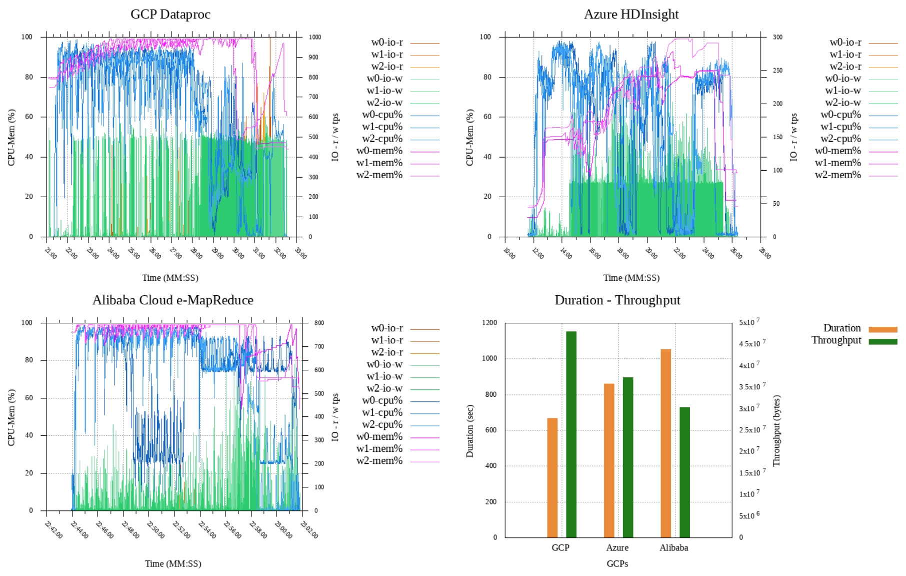


Figure 4: UC1 - Terasort (Gigantic; 3.2 GB)

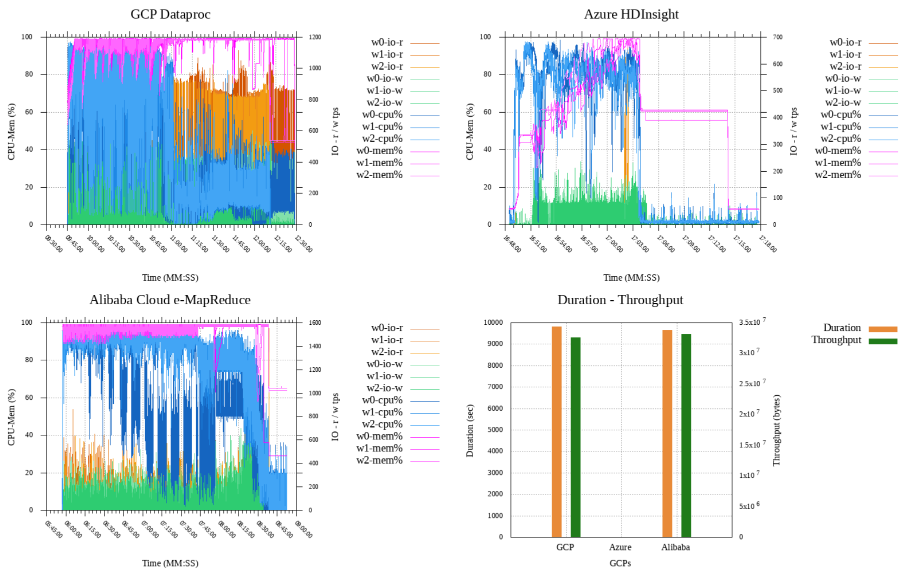


Figure 5: UC1 - Wordcount (Huge; 32 GB)

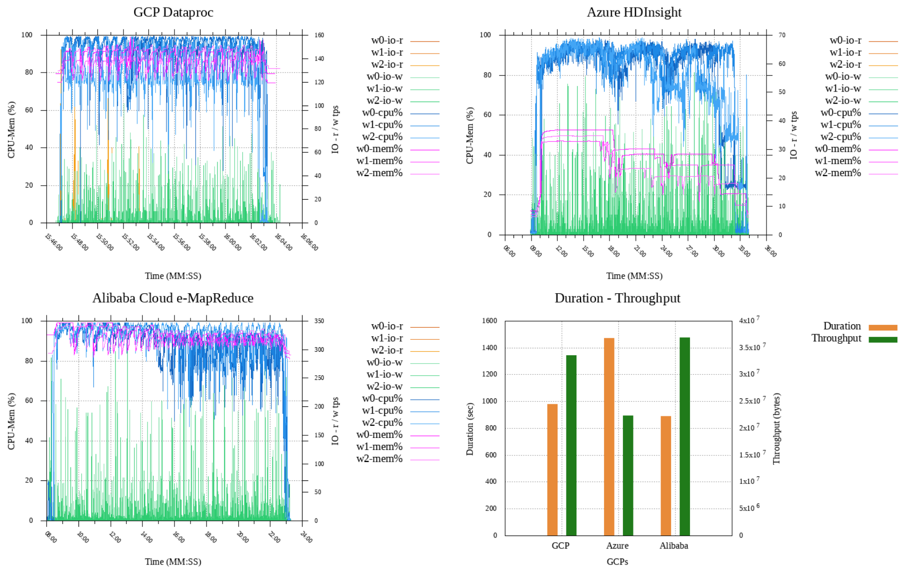


Figure 6: UC1 - Wordcount (Gigantic; 320 GB)

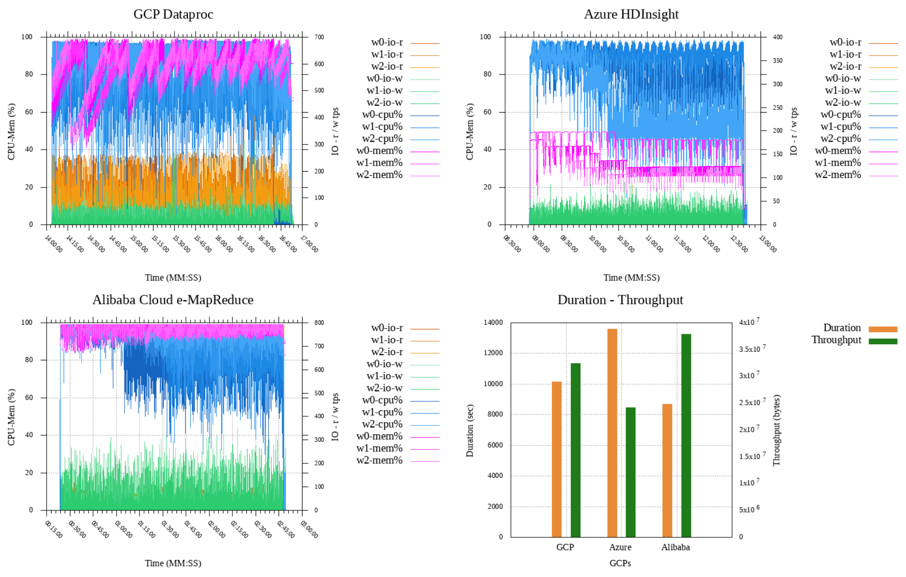


Figure 7: UC1 - Dfsioe-read (Huge; No of Files: 256, File size: 100 MB)

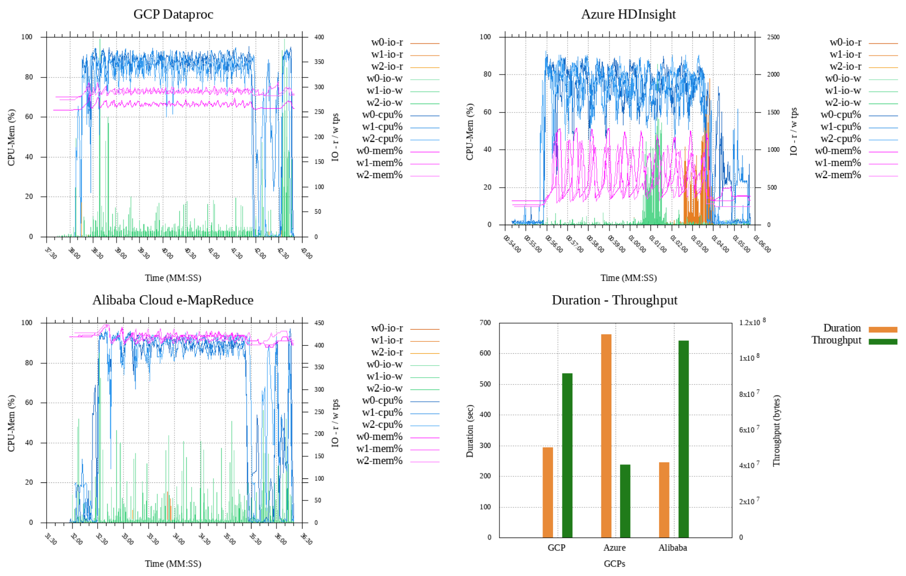


Figure 8: UC1 - Dfsioe-read (Gigantic; No of Files: 512, File size: 400 MB)

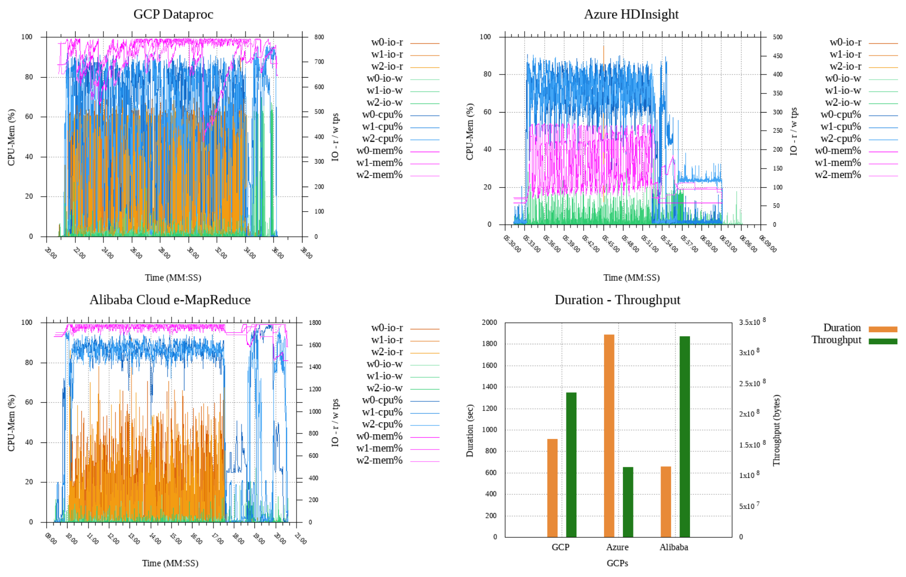


Figure 9: UC1 - Dfsioe-write (Huge; No of Files: 256, File size: 100 MB)

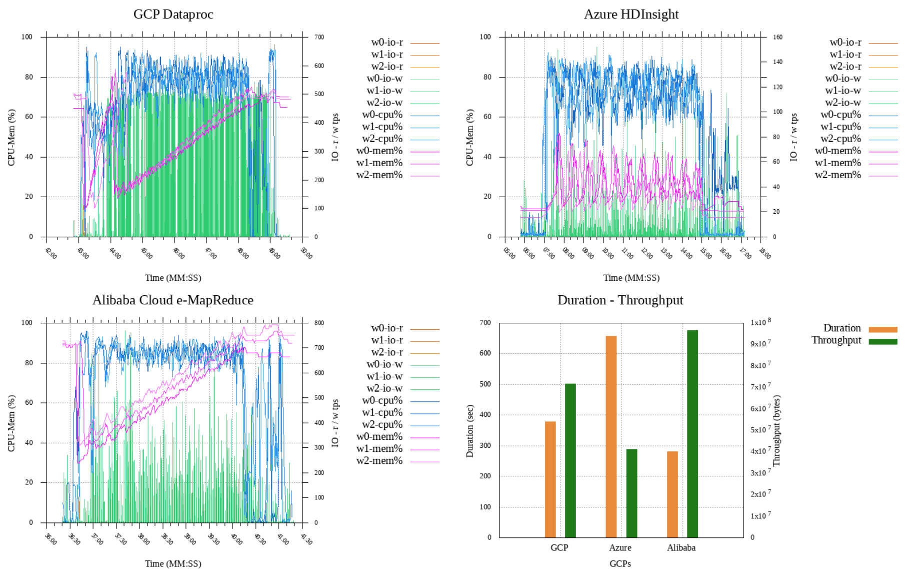


Figure 10: UC1 - Dfsioe-write (Gigantic; No of Files: 512, File size: 400 MB)

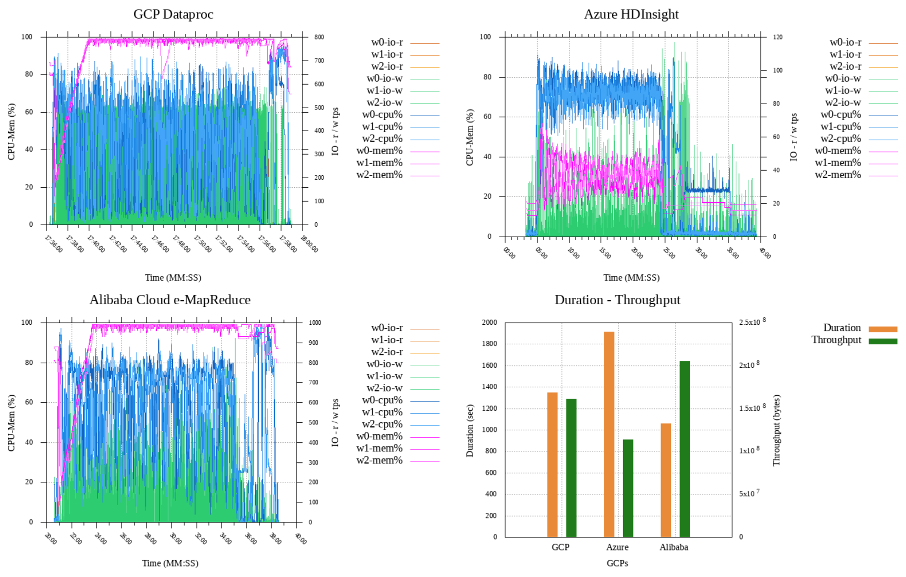


Figure 11: UC1 - Scan (Huge; USERVISITS: 10,000,000 PAGES: 1,200,000)

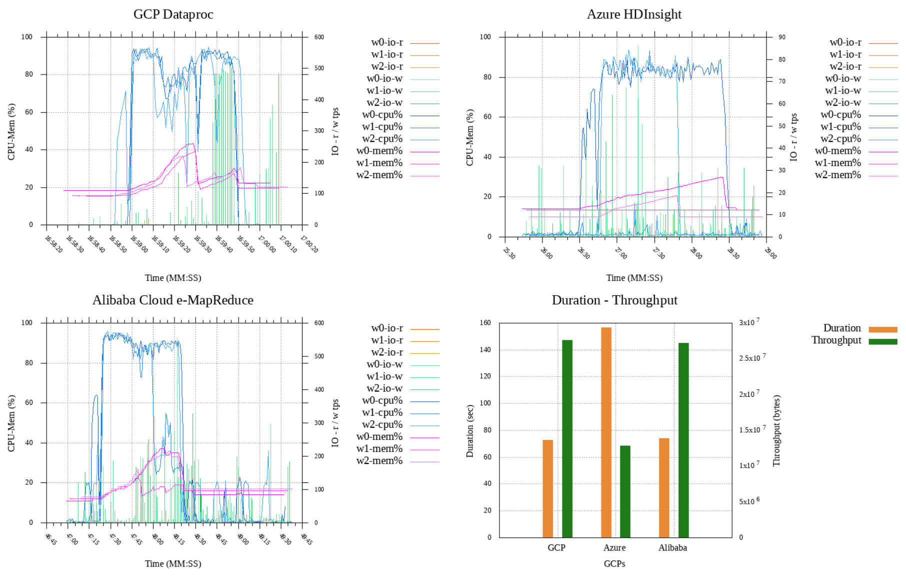


Figure 12: UC1 - Scan (Gigantic; USERVISITS: 100,000,000 PAGES: 12,000,000)

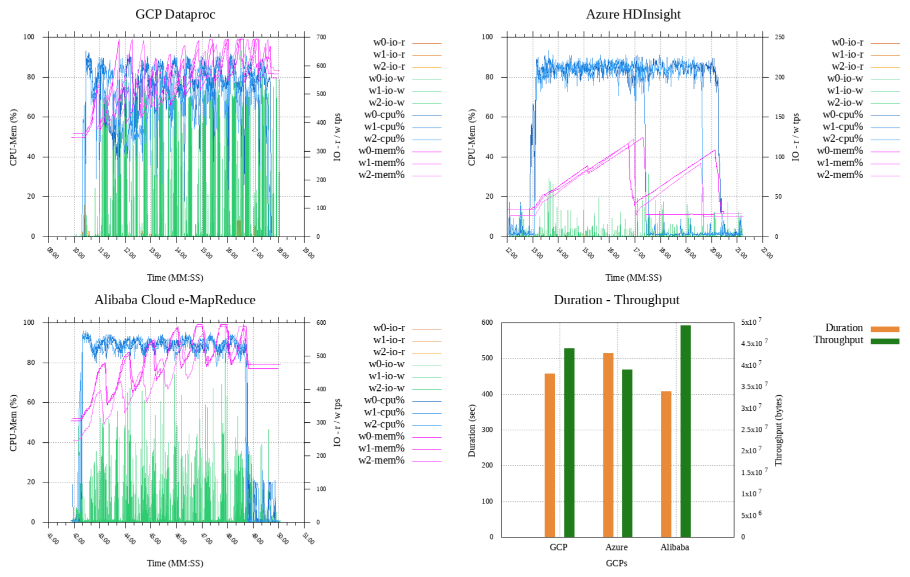


Figure 13: UC1 - Join (Huge; USERVISITS: 10,000,000 PAGES: 1,200,000)

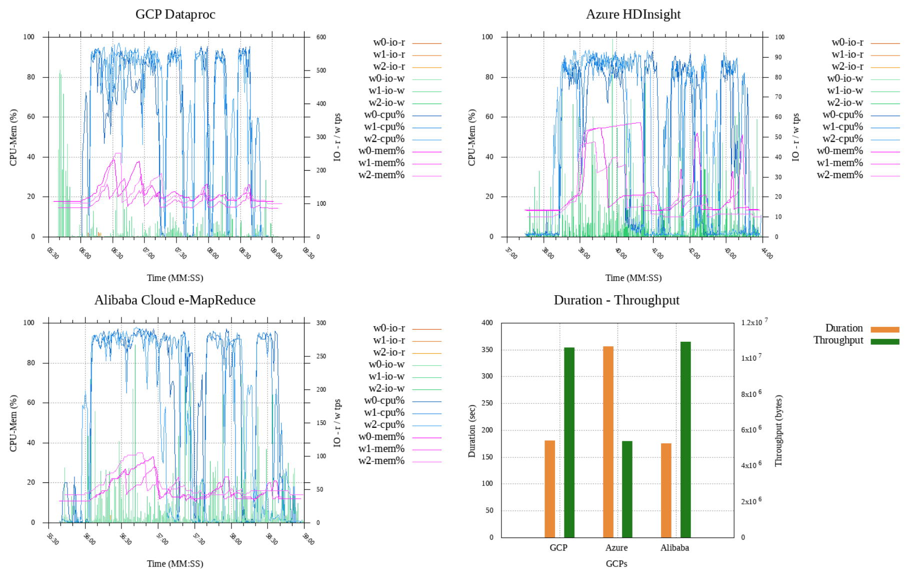


Figure 14: UC1 - Join (Gigantic; USERVISITS: 100,000,000 PAGES: 12,000,000)

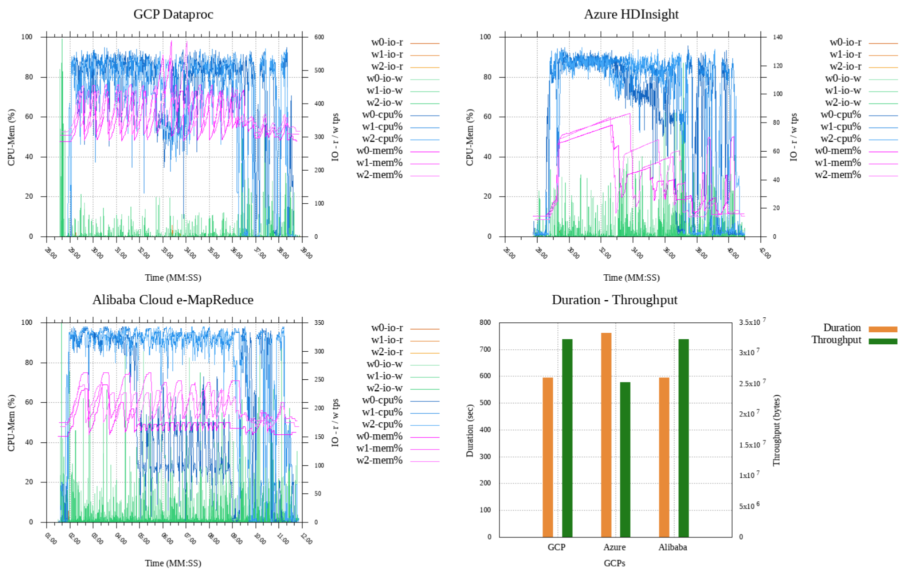


Figure 15: UC1 - Aggregation (Huge; USERVISITS: 10,000,000 PAGES: 1,200,000)

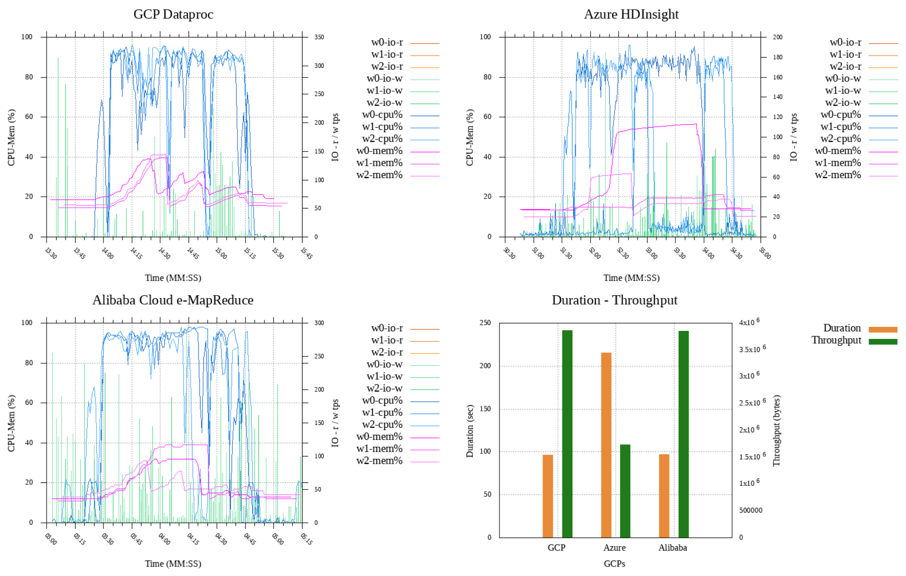


Figure 16: UC1 - Aggregation (Gigantic; USERVISITS: 100,000,000 PAGES: 12,000,000)

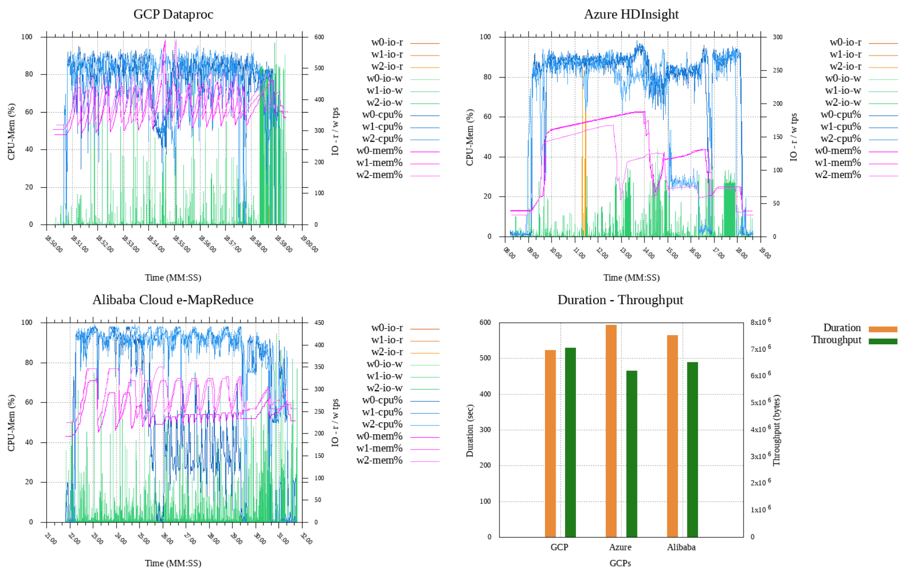


Figure 17: UC1 - Bayes (Huge; PAGES: 500,000 CLASSES: 100 NGRAMS: 2)

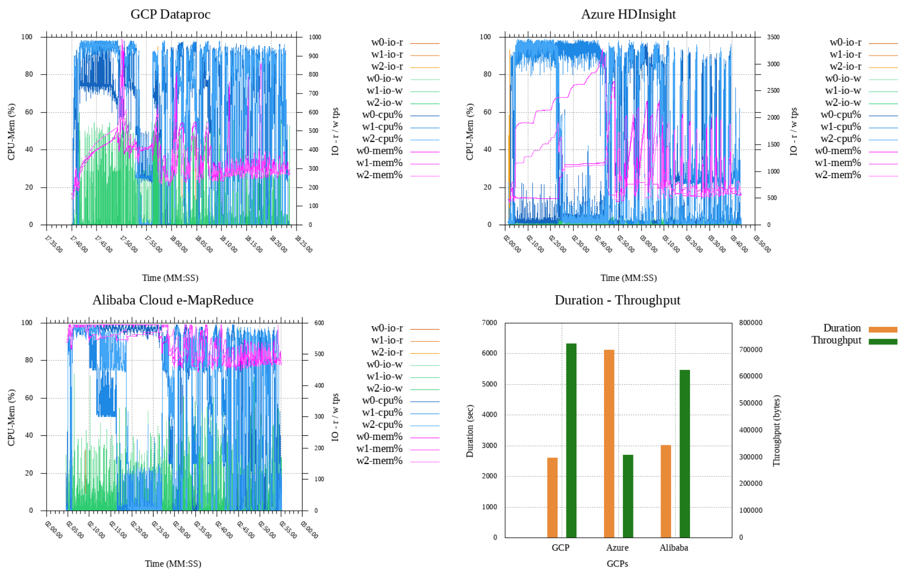


Figure 18: UC1 - Bayes (Gigantic; PAGES: 1,000,000 CLASSES: 100 NGRAMS: 2)

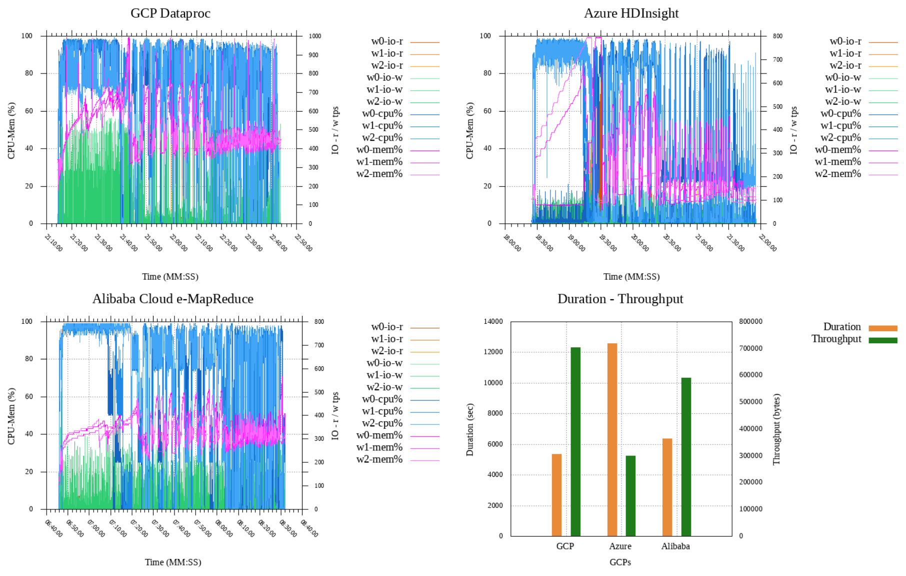


Figure 19: UC1 - Kmeans (Huge; CLUSTERS: 5 DIMENSIONS: 20 SAMPLES: 100,000,000 SAMP PER INPUT: 20,000,000 MAX IT: 5 K: 10 CONVERGEDIST: 0.5)

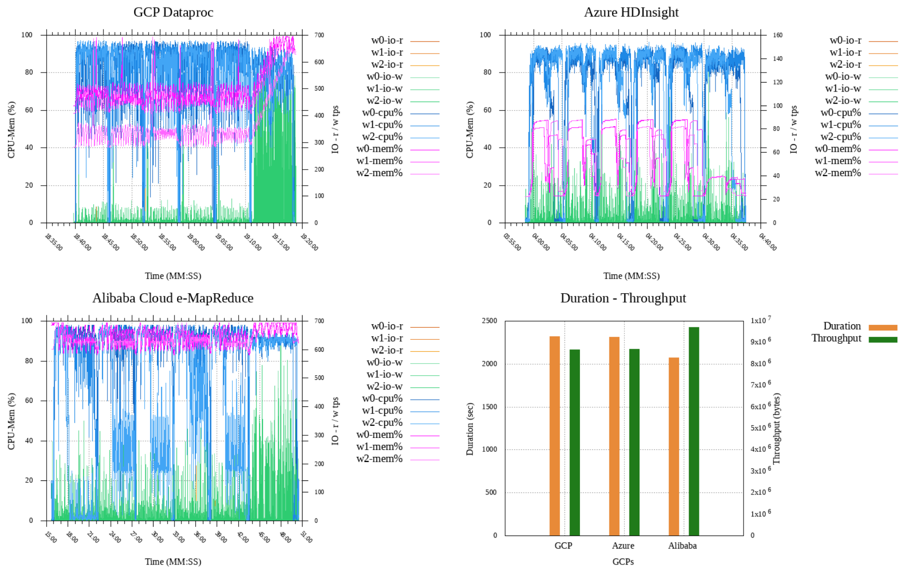


Figure 20: UC1 - Kmeans (Gigantic; CLUSTERS: 5 DIMENSIONS: 20 SAMPLES: 200,000,000 SAMP PER INPUT: 40,000,000 MAX IT: 5 K: 10 CONVERGEDIST: 0.5)

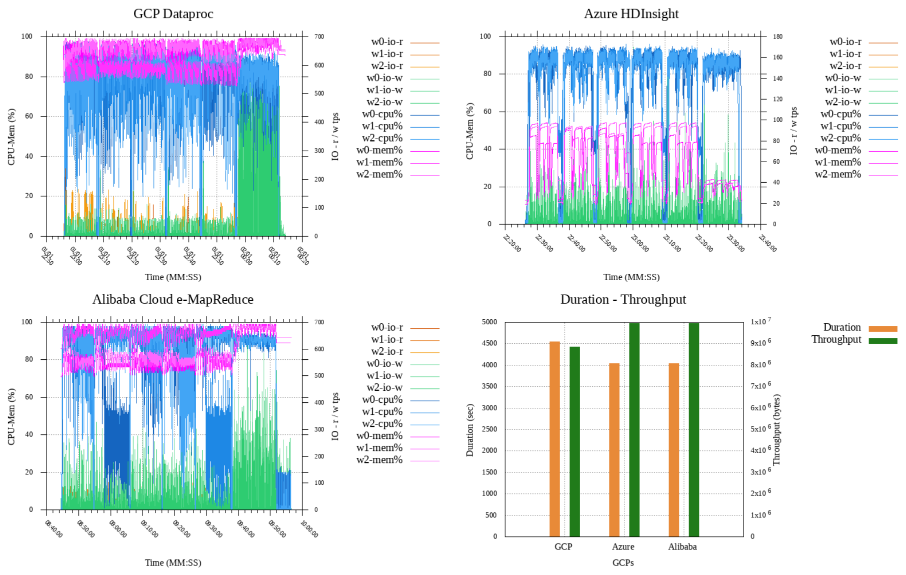


Figure 21: UC1 - Pagerank (Huge; PAGES: 5,000,000 NUM ITERATIONS: 3 BLOCK: 0 BLOCK WIDTH: 16)

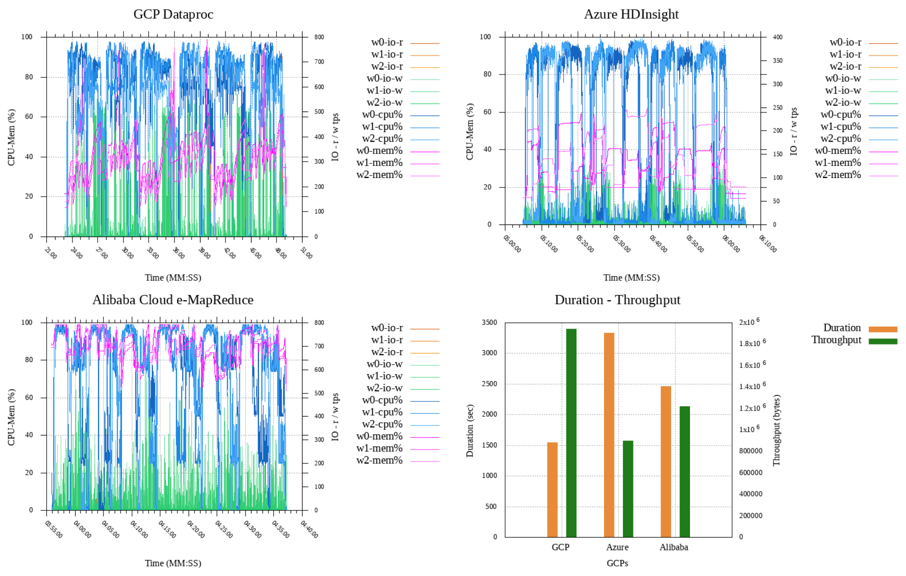


Figure 22: UC1 - Pagerank (Gigantic; PAGES: 30,000,000 NUM ITERATIONS: 3 BLOCK: 0 BLOCK WIDTH: 16)

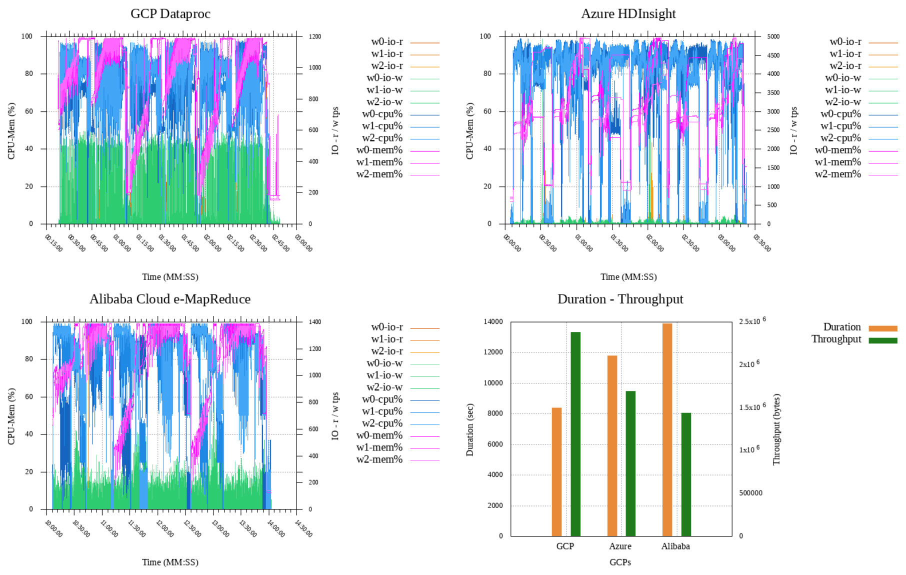


Figure 23: UC2 - Sort (Tiny; 32 KB)

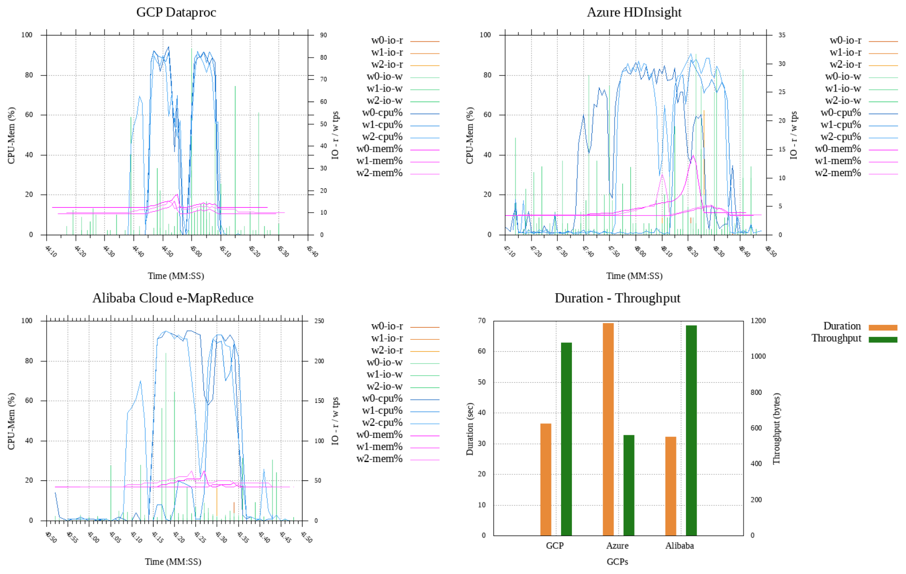


Figure 24: UC2 - Sort (Small; 3.2 MB)

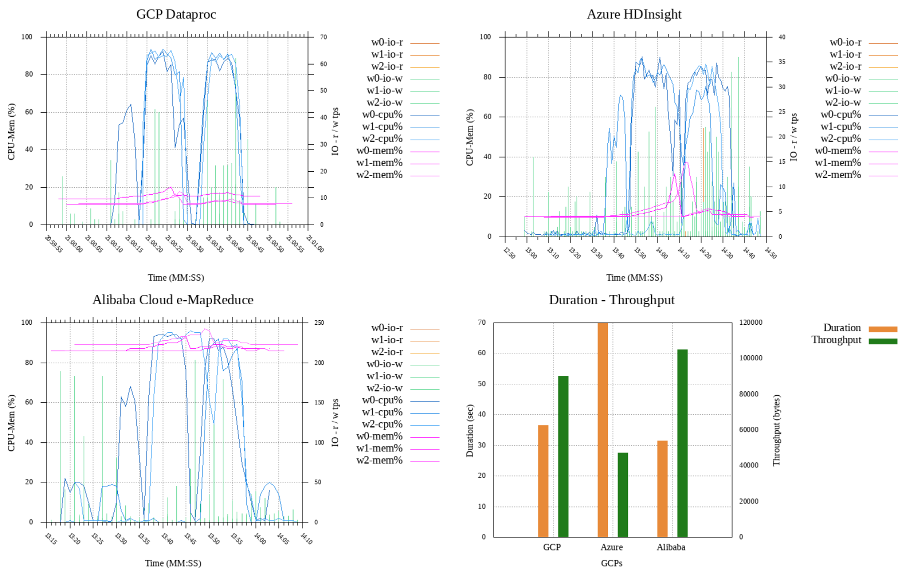


Figure 25: UC2 - Sort (Large; 320 MB)

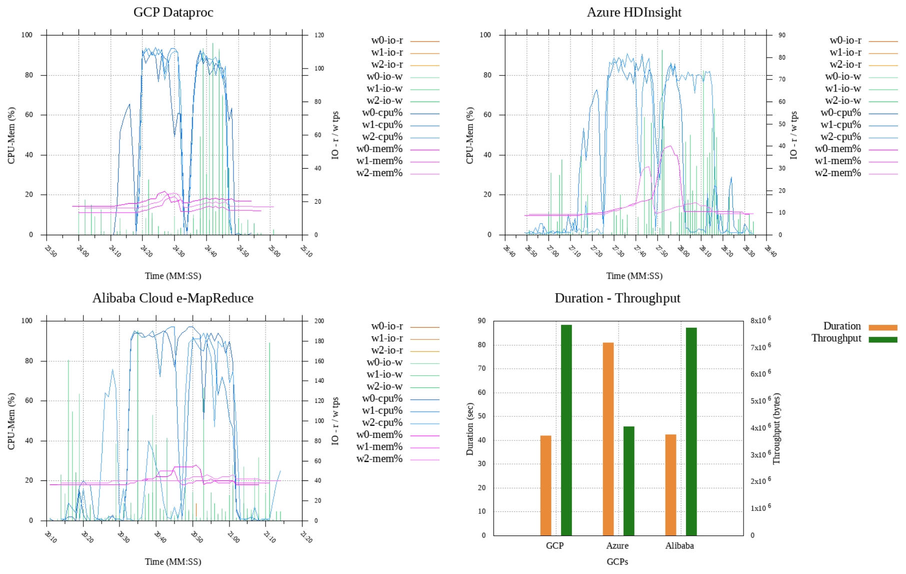


Figure 26: UC2 - Sort (Huge; 3.2 GB)

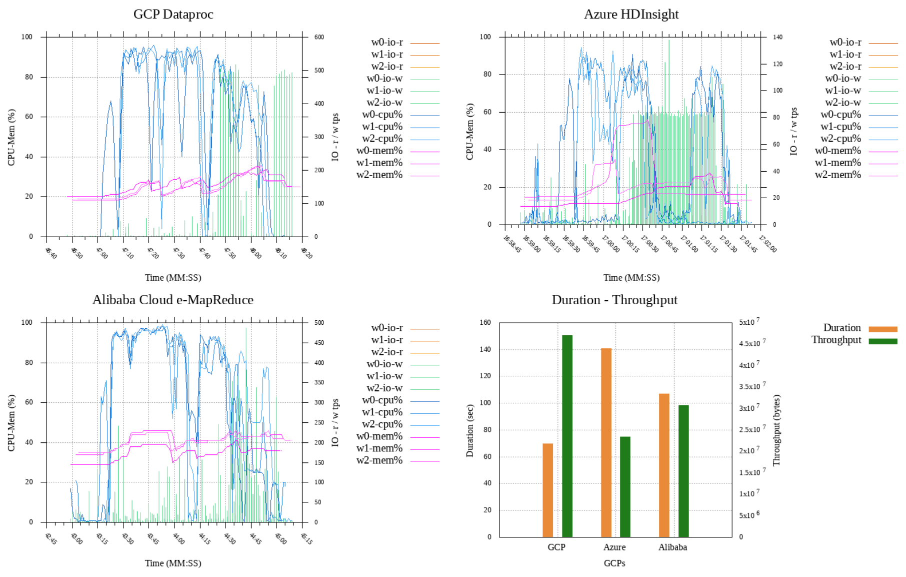


Figure 27: UC2 - Sort (Gigantic; 32 GB)

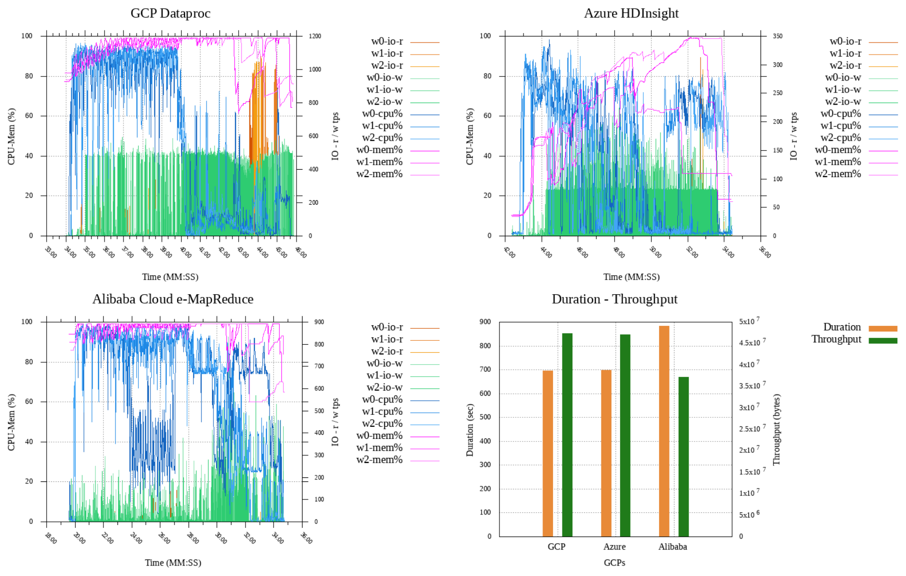


Figure 28: UC2 - Wordcount (Tiny; 32 KB)

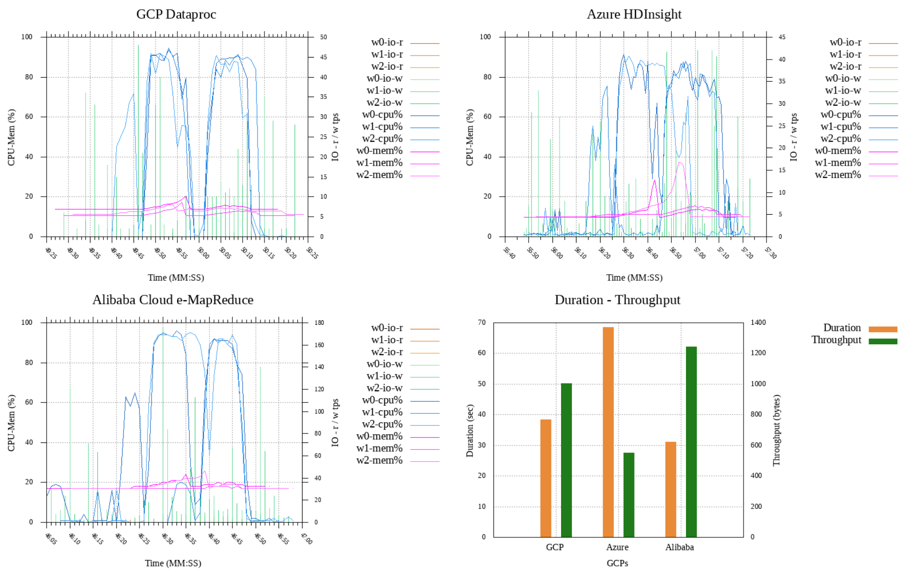


Figure 29: UC2 - Wordcount (Small; 320 MB)

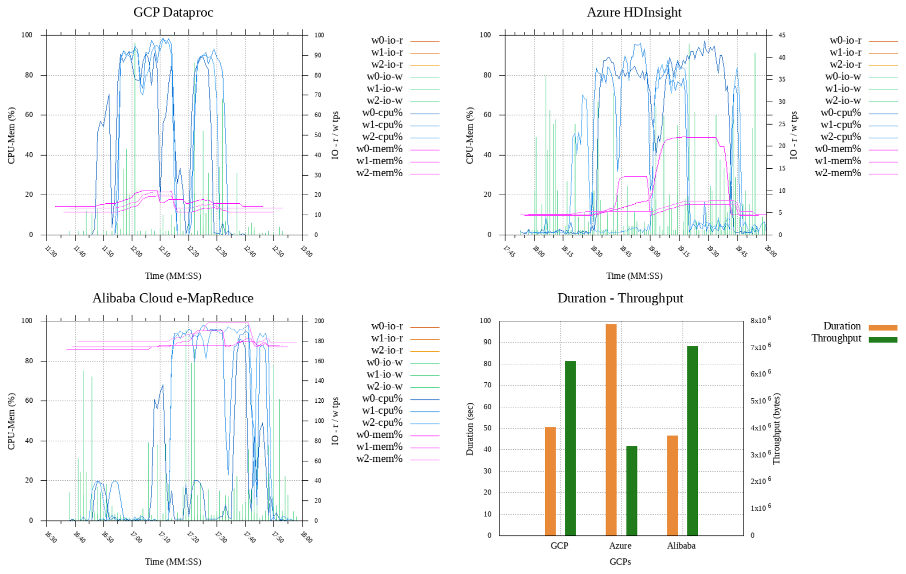


Figure 30: UC2 - Wordcount (Large; 3.2 GB)

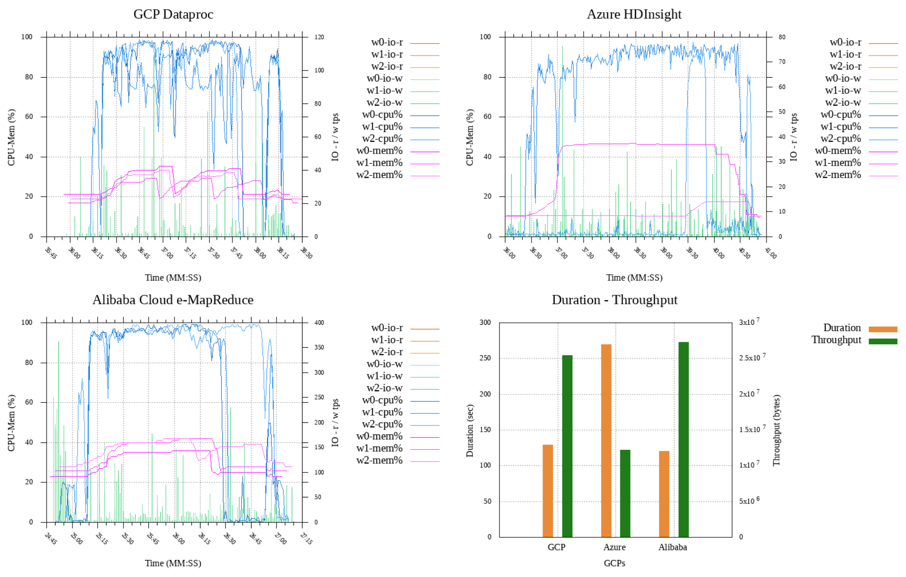


Figure 31: UC2 - Wordcount (Huge; 32 GB)

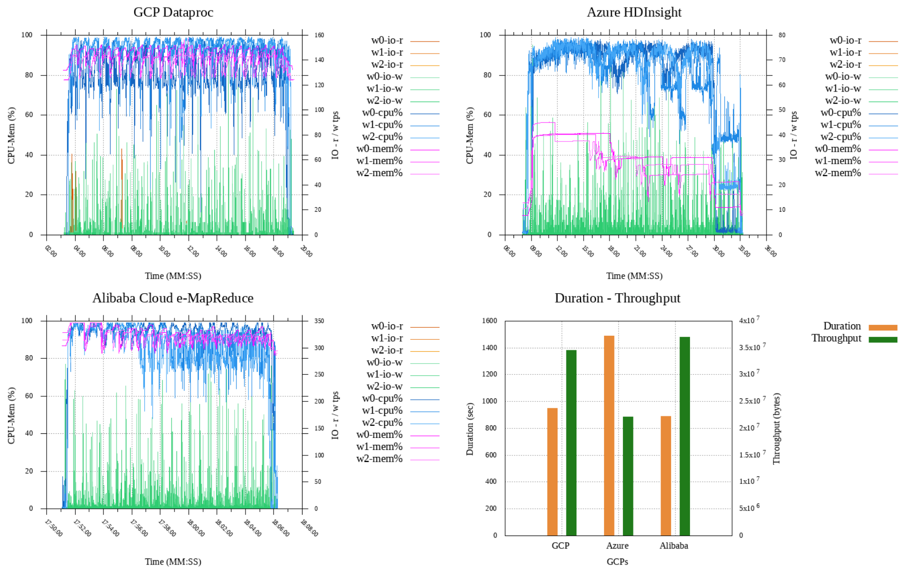
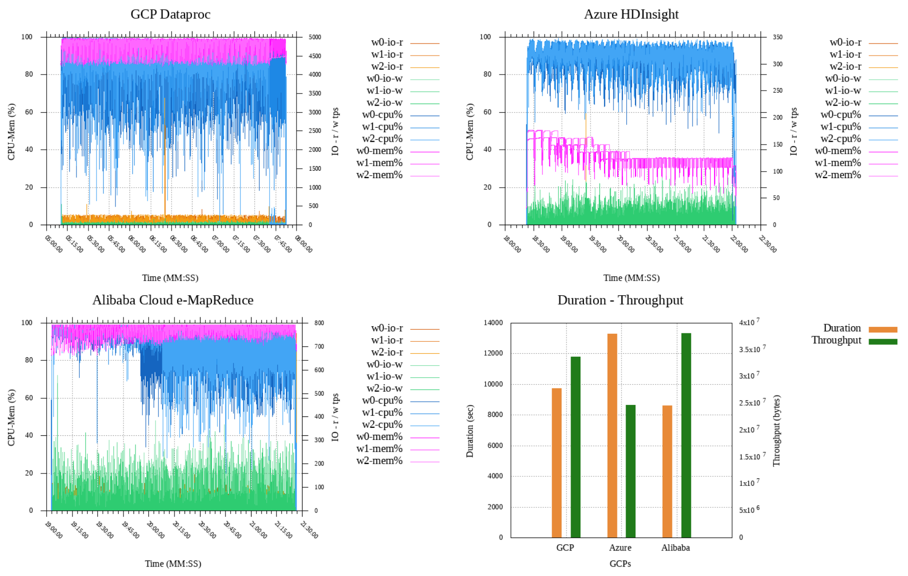


Figure 32: UC2 - Wordcount (Gigantic; 320 GB)


GCP Dataproc
Azure HDInsight
Alibaba Cloud e-MapReduce
Duration - Throughput
CPU-Mem (%)
IO - r / w tps
Time (MM:SS)
w0-io-r
w1-io-r
w2-io-r
w0-io-w
w1-io-w
w2-io-w
w0-cpu%
w1-cpu%
w2-cpu%
w0-mem%
w1-mem%
w2-mem%
Duration (sec)
Throughput (bytes)
GCPs
GCP
Azure
Alibaba
Duration
Throughput