SPEC Cloud IaaS 2018 Benchmark
Design Overview

1. Overview of SPEC Cloud® IaaS 2018 Benchmark

The SPEC Cloud® IaaS 2018 Benchmark is a software benchmark product developed by the Standard Performance Evaluation Corporation (SPEC), a non-profit group of computer vendors, system integrators, universities, research organizations, publishers, and consultants. It is designed to evaluate a computer system's ability to act as an **Infrastructure-as-a-Service** (**IaaS**) cloud.

This document describes the design of the **IaaS** benchmark, its design principles, goals, structure and components, the selected workloads, and the reasons for the design choices.

1.1 Trademark

SPEC and the name SPEC Cloud are registered trademarks of the Standard Performance Evaluation Corporation. Additional product and service names mentioned herein may be the trademarks of their respective owners.

1.2 Definitions

The definitions for the names and terms used in the benchmark are available in the benchmark's glossary: https://dev-www.spec.org/cloud_iaas2018/docs/glossary.html.

1.3 Design Principles

The guiding principle that underlies all SPEC benchmarks is to measure the performance using representative real-world workloads. SPEC Cloud IaaS 2018 Benchmark utilizes a subset of the workloads that represent real-world use cases found on public, private, or hybrid IaaS clouds. The benchmark utilizes two workloads, the Yahoo! Cloud Serving Benchmark (YCSB) [Reference: YCSBWhitePaper] and the K-Means implementation from HiBench [References: KMeansClustering and HiBenchIntro].

SPEC Cloud IaaS 2018 Benchmark workloads are managed by a benchmark harness, the Cloud Rapid Experimentation and Analysis Tool [Reference: CBTOOL].
CBTOOL is responsible for correct test execution across different clouds. The harness interoperates with the benchmark drivers. The **harness** creates and destroys *instances*, instantiates application instances, collects various measurements and data points, and computes various scores for each test.

The benchmark uses two execution phases, Baseline and Scale-out. In the baseline phase, peak performance for each workload is determined in separate test runs. Data from the baseline phase establishes parameters for the Scale-out phase. In the Scale-out phase, both workloads are run concurrently and new workload instances are injected every few minutes to increase the load on the cloud to determine performance and relative scalability metrics.

1.4 Requirements and Goals

The primary requirement and goal is to provide metrics that not only quantify the relative performance and capacities of an IaaS cloud, but also how typical cloud application workloads behave as the underlying cloud resources are stretched and may approach full capacity. The

benchmark envisions its main audience as hardware and software vendors, cloud providers, and cloud consumers. The cloud systems may be a private or a public cloud.
The main features of the benchmark are:

- Uses workloads consistent with popular social media applications using a NoSQL database and big data analytics using Hadoop.
- Stresses the provisioning and run-time of a cloud with multiple multi-instance workloads, subject to strict Quality of Service (QoS) metrics.
- No requirements are placed on the instance configuration. The tester is free to choose CPU (virtual CPU or core pinning), memory, disk (ephemeral disk or block storage), and network configuration for instances used to run the benchmark's workloads.
- Limited requirements are placed on the internal architecture of the test-bed.
- A hypervisor or virtualization layer is not required.
- Supports optional multi-tenancy.

The cloud being tested must have the following attributes:

- Consist of three (3) or more physical servers connected by a network.
- Have the ability to import an instance image or store a snapshot of an instance as an instance image, and provision one or more instances from that instance image.
- Have the ability to launch an instance without manual intervention.
- *CBTOOL* must be able to SSH into all instances over the network. If instances have a private IP address, *CBTOOL* can use a jump box or equivalent to SSH into the instances.

1.5 Excluded Goals

Some metrics that are potentially relevant to cloud will not be explicitly measured in SPEC Cloud IaaS 2018 Benchmark. These metrics include Durability, Isolation, Elasticity, Reliability, Power, Price, and Density. The subjective nature of these metrics makes it difficult to quantify as engineering metrics in the context of this benchmark.

The benchmark does not explicitly require the use of geographically isolated cloud configuration so it does not preclude tests from using a geographically distributed configuration, as long as the location information is included in the Full Disclosure Report.

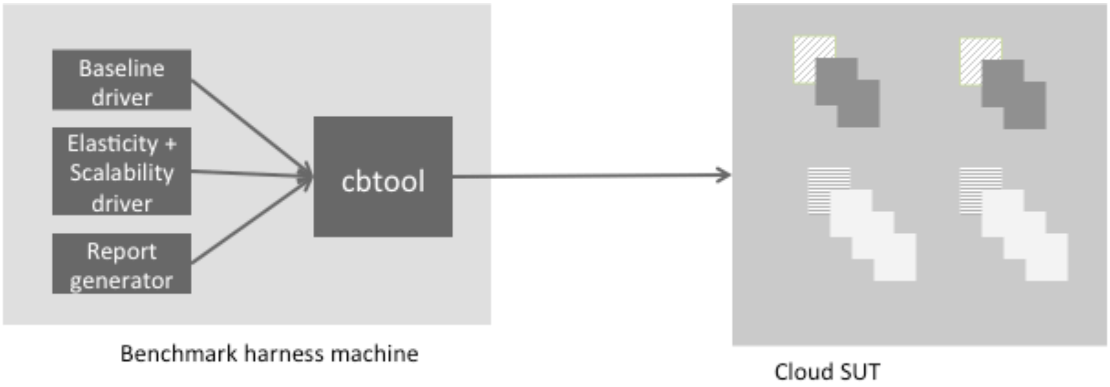
The benchmark is designed to measure performance of workloads typically run on cloud. Micro-benchmarks that measure only one aspect of IaaS cloud such as CPU, network, or memory are not part of the SPEC Cloud IaaS 2018 Benchmark.

2. SPEC Cloud IaaS 2018 Benchmark Architecture

The benchmark consists of several logical components. This section provides a high-level architecture of how these components fit together and interact with each other.

2.1 Logical Architecture

The entire physical configuration needed for SPEC Cloud IaaS 2018 Benchmark can be described using two groups of hosts. These machines may be physically co-located in the same facility (data center/region) or company campus. The machine on the left side of Figure 1 is the benchmark harness. The group of machines on the right side of Figure 1 represents the IaaS Cloud system under test (SUT).



Benchmark harness machine. It comprises Cloud Bench (cbtool), baseline and elasticity drivers, and report generators.

Group of overlapping squares represent an application instance. Different patterns within a group indicate instances for workload generators and workloads which may vary in size.

Figure 1 Logical architecture of SPEC Cloud IaaS 2018 Benchmark

2.1.1 Benchmark Harness

The harness is an infrastructure that automates the benchmarking process. It provides an interface for scheduling and launching benchmark runs. It also offers extensive functionality for viewing, comparing and charting results. Specifically, the harness:

- Starts and stops application instances and workload generators;
- Collects and aggregate the results;
- Determines if a run was successful; and
- Generates a full disclosure report

A Cloud benchmark has additional requirements not found in other SPEC benchmarks, such as instantiating new instances on command and decommissioning instances at the conclusion of the benchmark run. Usually, there are public or proprietary cloud management systems in use. Therefore, the harness must be extensible to support the ability to add or modify the distributed workloads or write custom modules that allow the benchmark to interface with the SUT's management system.

SPEC Cloud IaaS 2018 Benchmark uses Cloud Rapid Experimentation and Analysis Tool (CBTOOL), in this document. ***CBTOOL*** is an Apache 2.0 licensed cloud benchmark harness that meets the properties of the benchmark harness described above. ***CBTOOL*** exposes an API that is used by the ***baseline*** and ***scale-out driver***s for executing the two phases of the benchmark. Finally, the ***report generator*** is used to generate the report for the ***baseline*** and ***scale-out phase***s of the benchmark.

CBTOOL provides adapters for creating or deleting instances on clouds such as Amazon EC2, Digital Ocean public cloud, Google Compute Engine, IBM Softlayer, and OpenStack. Within each cloud, the API versions can vary over time. It is the responsibility of the tester to write or update an adapter for connecting ***CBTOOL*** to the cloud under test.

2.1.2 Life Cycle of An Application Instance

Figure 2 shows the life cycle of an application instance. Broadly, the life cycle of an application instance can be classified into ***provisioning***, ***data generation***, and ***load*** phases. Data generation and load phases constitute an application instance run (referred to as ***AI run***). Once CBTOOL receives a request to provision an application instance, it instructs the cloud under test to create the instances. Once the instances have been provisioned, and the application running in these instances is ready, the AI is considered to have been provisioned. This duration is indicated by the "AI prov. time" in Figure 2.

As soon as *CBTOOL* detects that the application is ready, it invokes a workload-specific script to generate the data set. The workload driver uses this data set during the run. The parameters of the workload were chosen such that data set is generated within a reasonable amount of time (i.e., under five minutes). As such it is possible for the generated data to fit within an instance memory; however, within each workload, certain data must be written to disk. Moreover, a different data set is generated for each ***AI run*** and stored in appropriate workload database.

Once the required data set has been generated, the workload driver starts the load phase. Upon completion of the load phase, the workload driver reports the collected metrics into CBTOOL. Then, any data that was generated before or during the run is discarded. The time duration between the start of data generation to the end of data deletion comprises an AI run.

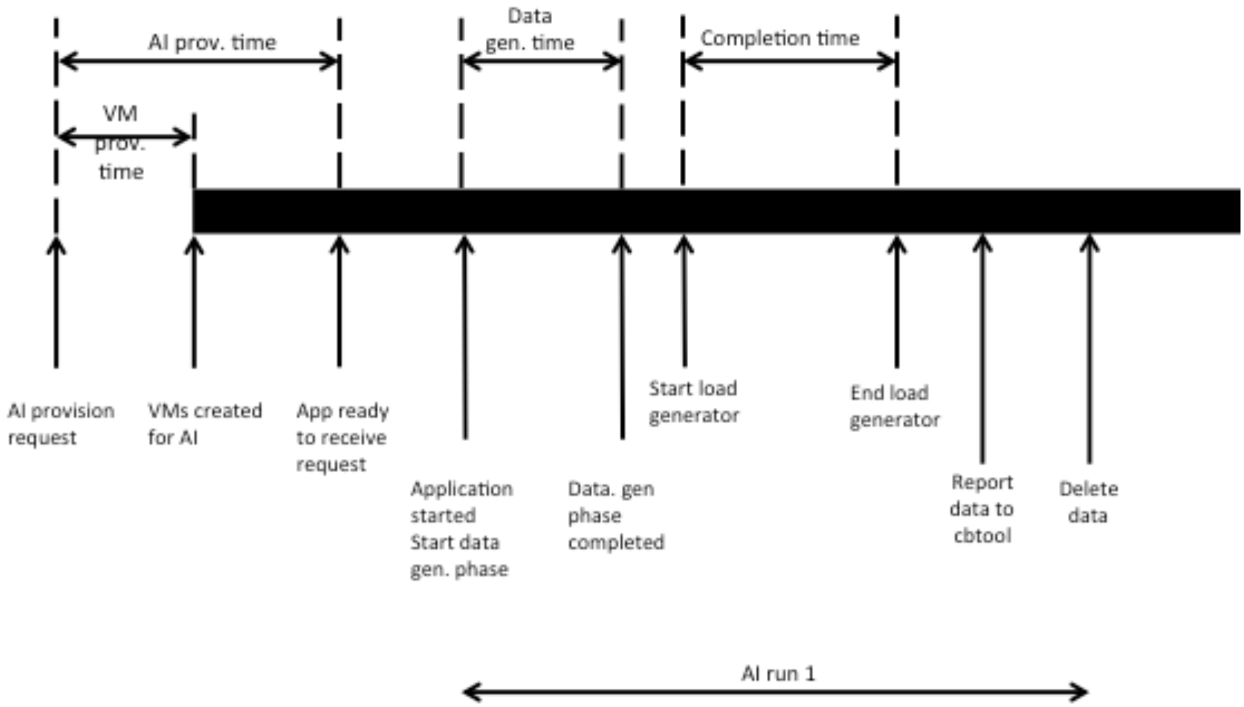


Figure 2 Life cycle of an application instance

In the baseline phase of the benchmark, the ***baseline driver*** creates an application instance, generates the data set, and runs the load phase a minimum of five times. As soon as the baseline driver receives the metrics from the workload driver, it terminates the application instance. It then repeats the same procedure for at least five times. By default, a total of 25 (5x5) AI runs for each workload are completed during the baseline phase.

In the scale-out phase of the benchmark, the ***scale-out driver*** instructs CBTOOL to create application instances for each workload with an interval between creations based on a uniform distribution between 5 and 10 minutes. The creation of each application instance results into a burst of instance creation requests at the cloud. The creation of application instance for each workload is independent. Once created, the application instances go through one or more AI runs (a data generation and a load phase).

During each AI run, CBTOOL invokes the workload-centric data generation drivers to generate the data set. The workload-centric data generation drivers generate the data according to appropriate probability distributions. The data set is generated during each ***AI run*** to reduce the caching that might result if the same data set is used across AI run or across multiple application instances. During scale-out phase, application instances are not deleted. Once the ***scale-out driver*** determines that a stopping condition for the benchmark has occurred, the driver instructs ***CBTOOL*** to stop creating application instances. A stopping condition includes events such as reaching a set maximum for number of AIs or when one or more QoS thresholds have been breached. This marks the end of scale-out phase of the benchmark.

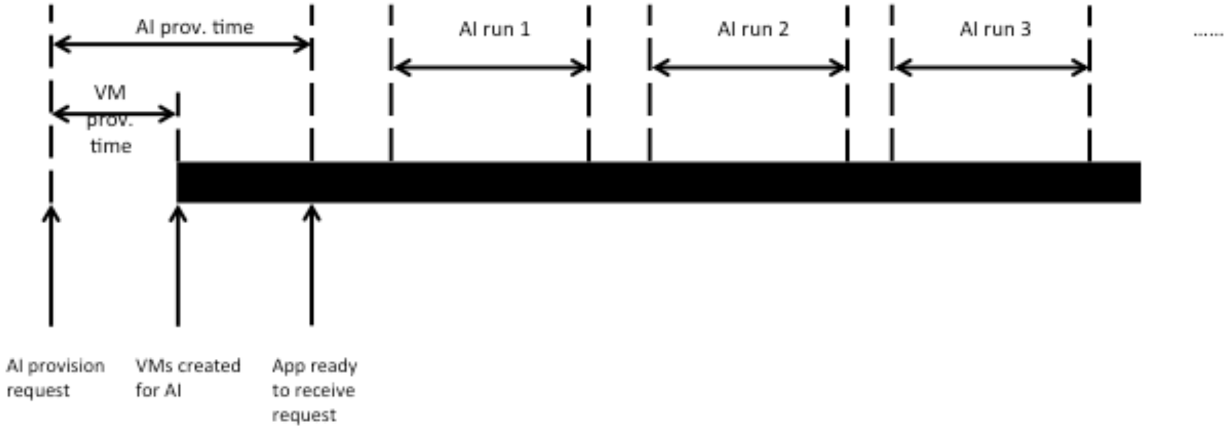


Figure 3 Life cycle of an application instance - multiple runs

2.2 IaaS Cloud – The System Under Test

The "System Under Test" (SUT) consists of:

- The host system(s) (including hardware and software) required to supply the "Infrastructure as a Service" that can support the multi-instance workload used by the benchmark.
- All network components (hardware and software) that connect the external clients to the cloud, and all network interfaces between host machines, which are part of the SUT.
- Network components between the workload generator instances/hosts and the SUT, which are not basic TCP/IP switches, routers, bridges or MAU (media adapter units). Some examples include: firewalls, round-robin DNS load balancers, load balancers, and anti-abuse filters.
- All software required to build, deploy, and run the specific benchmark workload.

  The SUT may offer infrastructure services for instances on bare metal servers, virtual machines, or containers.

2.3 Workloads

SPEC has identified multiple workload classifications already used in current cloud computing services. From this, SPEC has selected I/O and CPU intensive workloads for the benchmark. Within the wide range of I/O and CPU intensive workloads, SPEC selected social media ***NoSQL database transaction*** workload and ***K-Means clustering using map/reduce*** as representative of popular distributed workloads within cloud computing. The details of these workloads are described below.

2.3.1 I/O Intensive Workload: Yahoo! Cloud Serving Benchmark (YCSB) with Apache Cassandra

Social network sites are one of the more popular uses for large cloud computing. Social network sites contain many types of computing services, of which NoSQL database is a critical component and is I/O intensive. Yahoo! Cloud Serving Benchmark (YCSB) available under Apache 2.0 license simulates many types of database transactions, including a *read* dominated transaction mixture typical of most social media database activities. The SPEC Cloud IaaS 2018 Benchmark uses YCSB workload D (95% read, 5% insert) as the one that simulates simple social network user activities.

For NoSQL database, SPEC Cloud IaaS 2018 Benchmark uses the Apache Cassandra database as the underlying NoSQL database because during benchmark development, an Apache Cassandra database proved to be more sensitive to I/O and CPU resource constraints. Apache Cassandra is available under Apache 2.0 license.

Figure 4 shows the architecture of YCSB application instance in the SPEC Cloud IaaS 2018 Benchmark. The YCSB driver instance generates load on the Cassandra cluster. The Cassandra cluster comprises six instances. Together, these seven instances comprise the YCSB application instance for the benchmark. The choice of ***six*** instances for Cassandra represents a tradeoff between a trivial cluster size (e.g., two) and large cluster sizes (e.g., twenty), and having more than one workload generators to saturate the cluster, which will be required for large cluster sizes.

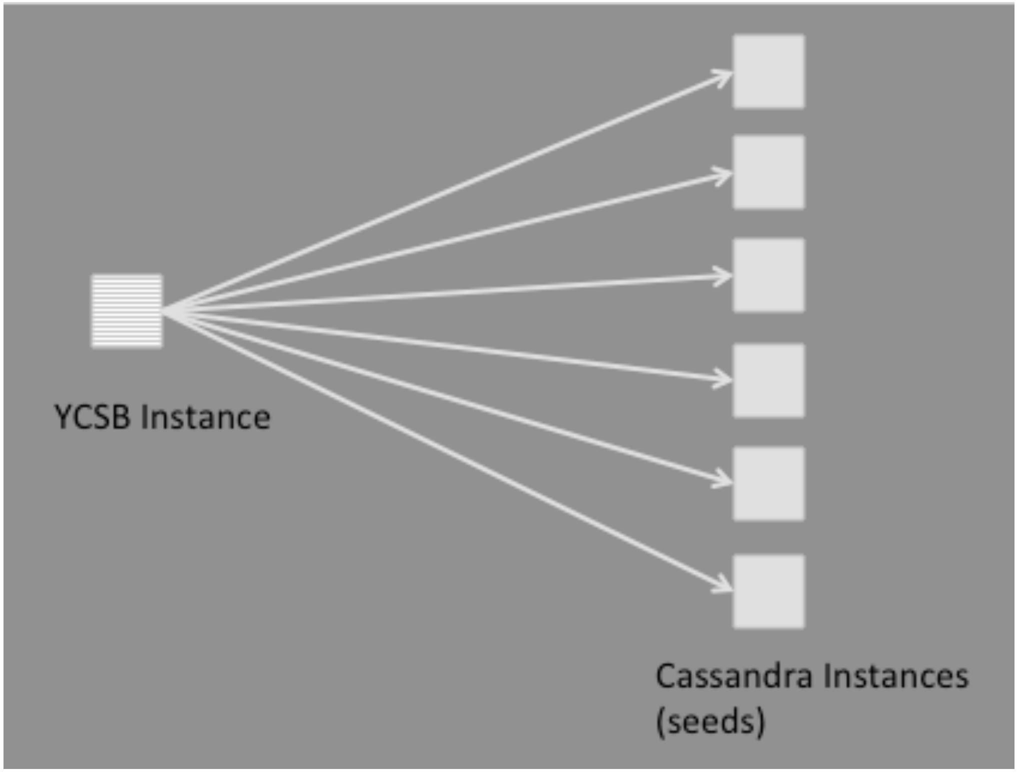


Figure 4 YCSB / Cassandra application instance in SPEC Cloud IaaS 2018 Benchmark

Cassandra supports two types of nodes in its cluster configuration, namely, seeds and data nodes. Seeds during startup work to discover the other seeds/data nodes that make up the cluster [Reference: CassandraSeedsOne]. One design option was to use three seeds and three data nodes. The data nodes take a non-deterministic time to join the cluster. Moreover, multiple data nodes joining at the same time is potentially problematic. The Cassandra documentation at the time of benchmark development, recommended a gap of two minutes between multiple

Cassandra data nodes that join an existing cluster [Reference: CassandraAddDataNodes]. Therefore, YCSB application instance uses six seeds as the six Cassandra instances.

2.3.1.1 Workload driver

YCSB driver instance generates load on the Cassandra cluster. Several configuration paramters for the YCSB driver instance can affect the load applied the Cassandra cluster. The thread_count parameter controls throughput level. In general, a higher number of threads will yield a higher throughput for Cassandra cluster until limits within the YCSB driver instance or Cassandra cluster are reached. If YCSB driver instance reaches the limits, adding more threads will not result in the YCSB driver instance sending more requests per second. If limits within the Cassandra cluster are reached, adding more threads in YCSB driver instance will not result into an increased throughput. Moreover, generating very high throughput using a very large number of threads may be susceptible to larger performance degradation as load on the cloud increases.

Another parameter is the operation_count which controls the time spent generating the load level enabled by the thread_count. After evaluating a number of options for setting these parameters, a thread_count of 40 and an operation_count of 4 million were selected for the 2018 benchmark. The goal was to allow improvements in I/O performance to be reflected by the benchmark and to the keep the YCSB workloads AI run cycle time in balance with the Kmeans workload for the "medium" flavor instances in current clouds such as those used in the reference platforms.

Table 1 shows the parameters used for YCSB driver. These parameters cannot be changed in any phase of the benchmark. The choice of total records inserted in DB is a careful design decision. During scale-out phase, the data is generated for each ***AI run***, the total records inserted were kept to 1,000,000. Given the default record length of 1KB, the data size is one GB. The effective data size with three-way replication is at least three gigabytes across six Cassandra seeds. As such this data is small enough to fit within the memory. However, 5% of the total operations are writes, which result in disk I/O during load generation. Moreover, the data set generation before an AI run also results in disk and network I/O.

The choice of request distribution governs which records become the most popular. The 'latest' distribution implies that the recently inserted records will become the most popular.

Table 1 YCSB configuration parameters for SPEC Cloud IaaS 2018 Benchmark

2.3.1.2 YCSB metrics

Description	YCSB parameter	YCSB parameter value

Total records inserted in DB	recordcount	1,000,000
Total operations during a YCSB run	operationcount	4,000,000
Number of Threads used by YCSB load generator	threadcount	40
Workload used	workload	com.yahoo.ycsb.workloads.CoreWorkload
Read all fields in the records returned	readallfields	true
Proportion of read operations	readproportion	0.95
Proportion of update operations	updateproportion	0
Proportion of scan operations	scanproportion	0
Proportion of insert operations	insertproportion	0.05
Request distribution	requestdistribution	Latest
Default data size of each record	1KB	10 fields, 100 bytes each, plus key

Following metrics from YCSB are used for Performance Score and Relative Scalability calculations:

- Throughput (ops/sec)
- 99th percentile of insert response time (ms)
- 99th percentile of read response time (ms)
- Average AI provisioning Time

2.3.2 Compute-intensive workload - K-Means with Apache Hadoop

The K-Means algorithm is a popular clustering algorithm used in machine learning. SPEC Cloud IaaS 2018 Benchmark uses Intel HiBench K-Means implementation [Reference: HiBenchIntro]. K-Means is one of the nine Hadoop workloads that are part of the HiBench suite. HiBench was selected as the benchmark suite as it provides multiple Hadoop workloads and has a uniform interface for running these workloads. HiBench uses Apache Mahout [Reference: ApacheMahout] for K-Means implementation. The HiBench K-Means workload was selected based on its range of workload models, and built-in data generator to drive the load.

The workload comprises a Hadoop name node instance, which also runs the Intel HiBench workload driver. The data is processed on five Hadoop data nodes. Together, these six instances comprise the K-Means application instance in SPEC Cloud IaaS 2018 Benchmark. Figure 5 shows the logical architecture of K-Means application instance in SPEC Cloud IaaS 2018 Benchmark.

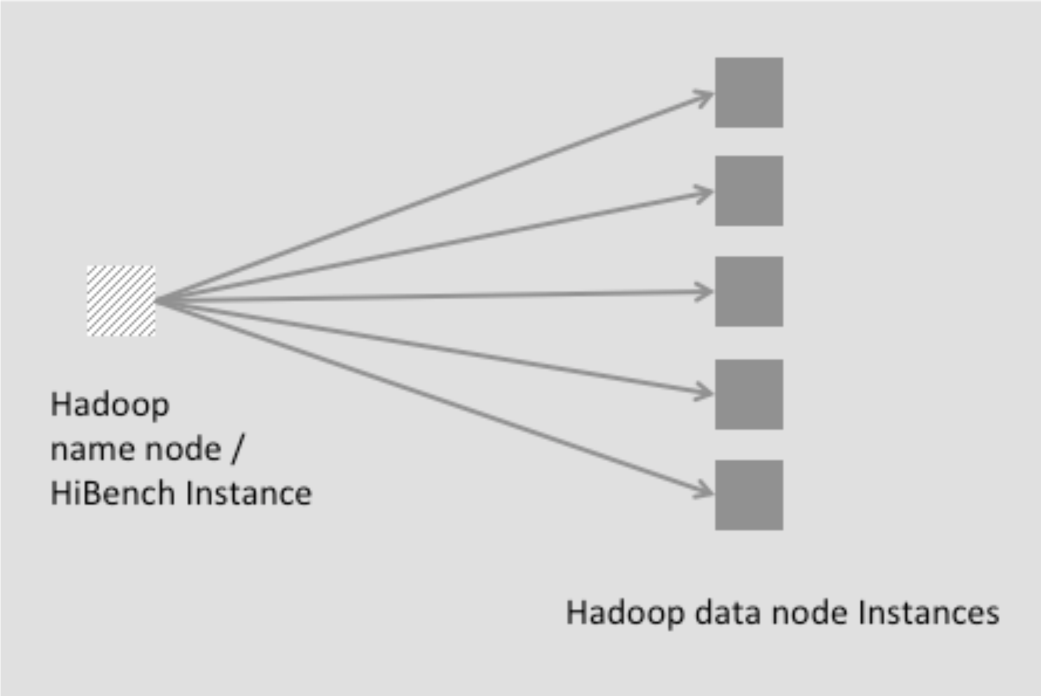


Figure 5 K-Means application instance

SPEC Cloud IaaS 2018 Benchmark uses Apache Hadoop (v2.7.1 or higher).

2.3.2.1 K-Means description

(The description in this section is copied verbatim from https://mahout.apache.org/users/clustering/k-means-clustering.html)

K-Means is a simple but well-known algorithm for grouping objects and clustering. All objects need to be represented as a set of numerical features. In addition, the user has to specify the number of groups (referred to as k) or clusters.

Each object can be thought of as being represented by some feature vector in an n-dimensional space, where n is the number of all features used to describe the objects in a cluster. The algorithm then randomly chooses k points in that vector space, and these points serve as the initial centers of the clusters. Afterwards, all objects are each assigned to the center they are closest to. Usually the distance measure is chosen by the user and determined by the learning task.

After that, for each cluster a new center is computed by averaging the feature vectors of all objects assigned to it. The process of assigning objects and recomputing centers is repeated until the process converges. The algorithm can be proven to converge after a finite number of iterations.

2.3.2.2 Workload driver

HiBench driver runs on the Hadoop namenode. It generates the dataset to be used by K-Means. It uses uniform distribution to generate centers for K-Means and uses Gaussian distribution to generate samples around these centers. Following attributes are used in data generation and for K-Means clustering.

Table 2 HiBench configuration parameters for SPEC Cloud IaaS 2018 Benchmark

Parameter	Value
NUM_OF_SAMPLES	1,000,000
SAMPLES_PER_INPUTFILE	500,000
NUM_CLUSTERS	5
DIMENSIONS	20
MAX_ITERATION	5
CONVERGENCE_DELTA (-cd option)	0.5
CANOPY_CLUSTERING (-cl option)	Used

The NUM_CLUSTER parameter indicates that the maximum number of clusters (or K) to be found is set to five. The DIMENSIONS parameter indicates that the number of features in a sample vector is set to twenty. The HiBench driver uses the EuclideanDistance to compute the distance between the sample object and the chosen centroid.

To bound the time it takes to run K-Means, MAXIMUM_ITERATION of five is specified. In theory, it implies that the algorithm can terminate before the CONVERGENCE_DELTA (cd) value of 0.5 is reached. The CANOPY_CLUSTERING option indicates that input vector clustering is done after computing canopies. Canopy clustering is used to compute the initial k vectors for K-Means. No compression is enabled for Hadoop.

With the MAXIMUM_ITERATION set to 5, the convergence can take 1, 2, 3, 4, or 5 Hadoop Iterations (HI); the probability distribution range runs from 1.56%, 19.88%, 23.39%, 21.44%, and 33.72%, respectively. It should be noted that during baseline runs using the default settings, it is very likely that a sample of 1 hadoop iteration may not occur.   If any convergence iteration count does not occur in baseline but does occur during the scale-out phase that sample will be ignored. Since it is most likely to be the rare 1-iteration that gets dropped there should be minimal impact on the relative scalability calculation.   The tester can increase the baseline iteration_count to get to more samples and increase the likelihood of having all 5 samples in their baseline run if desired.

Another effect of the range of convergence iterations in any given AI run, is the associated completion times are variable as well; for the reference platform the completion time ranges from 76 for (HI=1) to 180 seconds (HI=5). The reference completion time can calculated using the formula $CT = 26\ x\ HI + 50$. For Kmeans, both the performance score and the relative scalability calculations use the iteration specific completion times from the reference platform and the baseline run respectively. This ensures that each Kmeans AI run measurement is scaled fairly, so overweighting or underweight is avoided when a frequency of higher vs lower counts occur.

The size of the generated data set is approximately 415 MB. The total size of the data at the end of a run is approximately 900 MB. With Hadoop's three-way replication, the size on disk is approximately 2.8 GB.

Using a medium instance size (2 vCPUs and 4GB memory), the KMeans completion and data generations time have roughly the same duration as the YCSB completion time and data generation per AI run for the reference platform.

The commands to run K-Means load driver are copied from the code for reference:

```
OPTION="$COMPRESS_OPT -i ${INPUT_SAMPLE} -c ${INPUT_CLUSTER} -o
${OUTPUT_HDFS} -x ${MAX_ITERATION} -ow -cl -cd 0.5 -dm
org.apache.mahout.common.distance.EuclideanDistanceMeasure -xm mapreduce"

${MAHOUT_HOME}/bin/mahout kmeans ${OPTION}
```

2.3.2.3 K-Means Metrics

Following metrics are reported:

- Completion time (seconds)
- Hadoop iteration count for each AI run
- Average AI provisioning time

2.4 Reference Platform

The benchmark uses results from a reference platform in the Performance Score calculation. Several member companies participating in the benchmark development collected results from baseline runs on private and public clouds without applying any performance tuning. These results were composited to create the set of reference platform values for the 2018 release consisting of a YCSB throughput (with corresponding latencies for reference) and a set of KMeans completion times for each Hadoop iteration count (1-5). The reference platform values are recorded in the osgcloud_rules.yaml and can not be modified without invalidating the test results.

3. Running the Benchmark

This section provides a high-level overview of the testing process. To setup and install the benchmark please read the SPEC Cloud IaaS 2018 benchmark's User Guide and Run and Reporting Rules documents.

3.1 Setup (Manual)

The tester first needs to have access to a private or public cloud to test and to determine if there is an existing CBTOOL adapter that may support that cloud. If an adapter is not available for that cloud or the cloud API has changed since the adapter was submitted, then the adapter must be written or modified. It is up to the tester to provide a functioning adapter in order to get the SPEC Cloud IaaS 2018 benchmark running. Please see the User Guide for more details.

With a cloud available to test and a CBTOOL adapter for that cloud, the tester can proceed with setting up the benchmark.

- The tester needs an environment on which to install the SPEC Cloud IaaS 2018 Benchmark software which includes the CBTOOL harness and SPEC Cloud scripts and configuration files. For private clouds the benchmark harness must be installed on a separate system from the cloud but it must have network access to instances on the cloud. For public clouds, the harness may be run on a separate instance in the cloud.
- Instance images for the YCSB and KMeans workloads must be created. The application software for each workload is included with the benchmark kit.
- With the benchmark installed on the harness and instance images for YCSB and KMeans in the cloud, the tester can now try connecting to CBTOOL to the cloud. This can be done manually by using CBTOOL's API directly or by using the benchmark's all_run.sh script (--help) to test each workload by running a baseline (single iteration).
- With the ability to run baseline for both workloads, a full run of baseline and scale-out phases for a few AIs can be attempted. If successful, the test will generate an html report with the results of the test.
- Once a full run has been successful, the tester can proceed with additional testing and tuning of the cloud. Testers of public cloud should make sure they monitor the costs ($$$) associated with the number of instances created and their running time and ensure that the instances are deprovisioned at the end of all tests.
- Testers interested in publishing the results from compliant tests may submit these results and supporting data by following the process in the Run and Reporting Rules document.

3.2 Baseline (Automated)

SPEC Cloud IaaS 2018 Benchmark baseline driver instructs the ***CBTOOL*** through its API to create a single application instance for KMeans. ***CBTOOL*** then starts an AI run by instantiating the data generation in the application instance and then starts the load generators. After set of 5 AI runs complete, the baseline driver collects the supporting evidence and then the AI is deprovisioned. This is process is repeated until 5 KMeans AIs have been created and destroyed.

The same process is followed for the YCSB workload.

If there are no errors in the five runs of either the KMeans or YCSB AIs as reported by ***CBTOOL*** and the results meet the bounds defined for Quality-of-Service thresholds, the baseline result is considered valid.

These settings and measurements are the ***baseline*** configurations and measurements and must be used by the scale-out phase that follows.

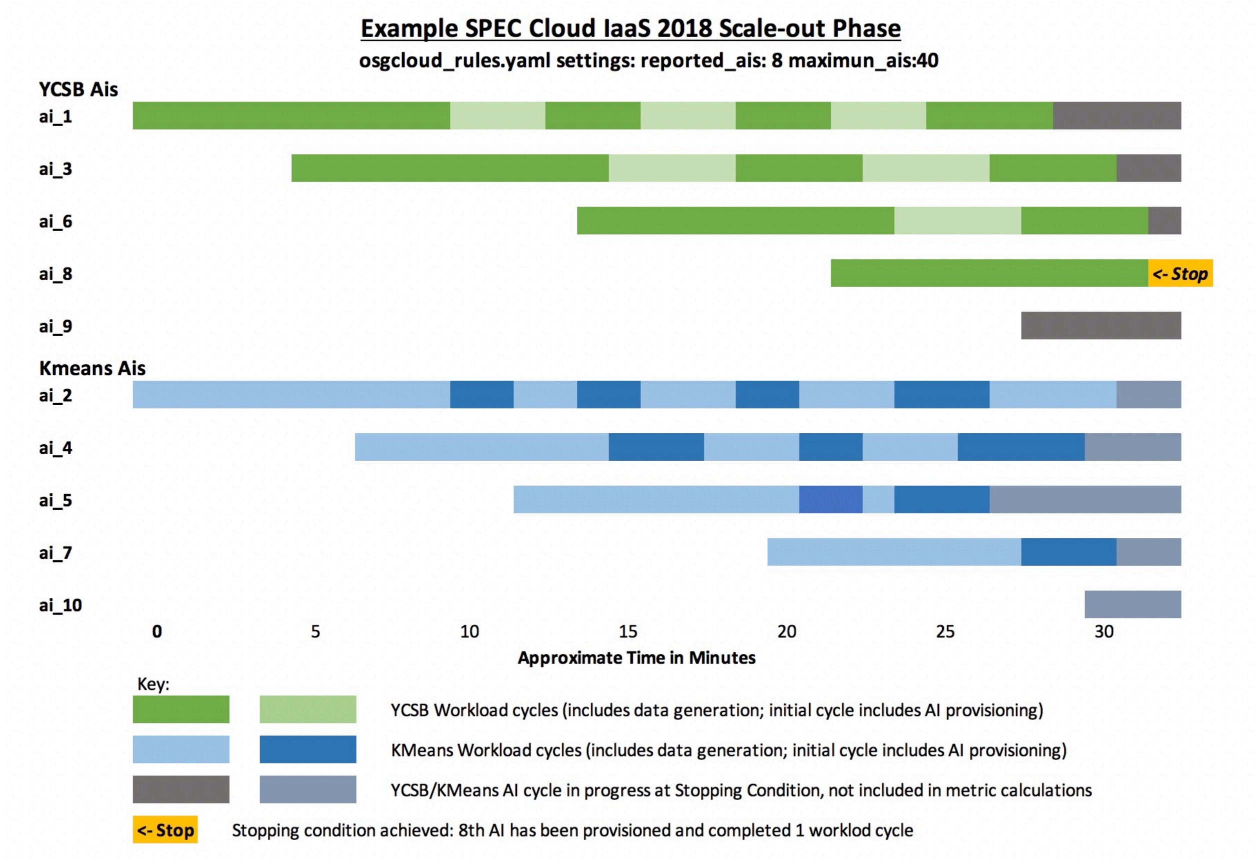
3.3 Scale-out (Automated)

SPEC Cloud IaaS 2018 Benchmark scale-out driver instructs the ***CBTOOL*** via its API to connect to the cloud and repeat the following cycle until one or more stopping conditions exist.

1. Start one application instance for the YCSB workload randomly between five and 10 minutes and start one application instance for the KMeans workload randomly between five and 10 minutes (Note: the initial workload AI starts within a few seconds).
2. Asynchronously, wait until each application instance is ready to accept work, and repeat the following sequence (an AI run).
    a. Start the data generation for each application instance
    b. Start the configured workload driver and wait until it completes
    c. Record results and verify that are within QoS thresholds and increment associated counters.
    d. Destroy any generated data, and repeat step a-c.
3. On every new instance creation or when results for an AI run are received:
    a. Check for application instance related stopping conditions.
    b. If within acceptable conditions, go to Step 2.
    c. If outside acceptable conditions or maximum AIs as set by the tester, stop the execution of the scale-out phase and collect supporting evidence.

SPEC Cloud IaaS 2018 Benchmark scale-out driver uses ***CBTOOL*** to detect the following stopping conditions. The details for stopping conditions are in the Run and Reporting Rules document:

- 20% or more of the AIs fail to provision
- 10% or more of the AIs have any errors reported in any of the AI runs. This includes AIs that fail to report metrics after 4 x the completion time of baseline phase.
- 50% of the AIs or more have QoS condition violated across any run
- Maximum number of AIs as set by the tester is reached.
- Reported AIs as set by the tester is reached.



3.3.1 Arrival Rate of AIs

The arrival rate of AIs for each workload is uniformly distributed between 5 and 10 minutes throughout the benchmark run. Each AI results into a burst of instance creation requests; seven instance creation requests for YCSB and six instance creation requests for K-Means, respectively.

3.4 White-box vs. black-box cloud considerations

Black-box clouds (public clouds) are typically multi-tenant. Multi-tenancy implies that one or most tenants (Cloud consumers) share the underlying cloud infrastructure such as compute, network, and storage with each other. In white-box clouds, both hardware and software are under the control of the tester. White-box cloud can be run as a single-tenant or multi-tenant. SPEC Cloud IaaS 2018 Benchmark does not place any limitation on how many tenants are defined for the cloud. It is up to the tester to configure the number of tenants. The metrics are aggregated across all tenants in the final score.

Blackbox (public) cloud variations may have performance variations due to multi-tenancy, use of different hardware, or time of day [Reference: EC2PerfVariations]. It is important to take criteria such as time of day and geographies in to consideration when evaluating cloud performance metrics. The benchmark can be run across multiple times of day to measure variation in performance of blackbox clouds.

4. Metrics and Computations

The SPEC Cloud IaaS 2018 Benchmark reports four primary metrics namely,

- Replicated Application Instances
- Performance Score
- Relative Scalability
- Mean Instance Provisioning Time

These metrics are
Secondary metrics reported include:

- AI Provisioning Success
- AI Run Success
- Scale-out Start Time
- Scale-out End Time
- Total Instances

The workload specific metrics and measurements are reported in the summary and detailed sections of the report. These metrics are defined below. Example calculations are provided using the data from a test using the CBtool simulator mode, so values are not representative of any real cloud. The full disclosure report for the example test is included in Appendix 1 for reference.

4.1 Metric: Replicated Application Instances

The Replicated Application Instances metric reports the total number of valid AIs that have completed at least one application iteration at the point the test ends. The total copies reported is the sum of the Valid AIs for each workload (KMeans and YCSB) where the number of Valid AIs for either workload cannot exceed 60% of the total. The other primary metrics are calculated based on conditions when this number of valid AIs is achieved so are subordinate to this metric. The unit of measure is *copies* since the benchmark scales out by adding new copies of AIs to increase the load on the cloud. The formula used to calculate the metric:

`Replicated Application Instances = (YCSB_Valid_AIs + KMeans_Valid_AIs) copies`

`= ( 4 + 4 ) = 8 copies`

4.1.1 Discussion

A tester may set the limit on the maximum AIs that will be run in the cloud environment (SUT) which limits the Replicated Application Instances metric to no more than the limit set. If this limit is set to a small value, the Performance Score may be small while the Relative Scalability metric may be high. Average instance provisioning time may not be impacted.

In general, a cloud with higher Performance Score and Relative Scalability along with, lower provisioning times and fewer errors is better than a cloud with low Performance Score and Relative Scalability metric and higher provisioning times or errors for the same number of Replicated Application Instances. Tests with different Replicated Application Instances can be compared using the other primary metrics to focus on the aspects that are of most interest to the reader (e.g., most work done, most consistency across AIs, fastest provisioning).

4.2 Metric: Performance Score

The Performance Score is an aggregate of the workload scores for all valid AIs represents the total work done at the reported number of Replicated Application Instances. It is the sum of the KMeans and YCSB workload performance scores normalized using the reference platform. The reference platform values used are a composite of baseline metrics from several different white-box and black-box clouds. Since the Performance Score is normalized, it is a unit-less metric.

The benchmark's overall Performance Score is the sum of the individual workload performance scores. The formula used to calculate the metric:

```
Performance_Score = Sum(YCSB_PerfScore,KMeans_PerfScore)
= ( 2.52 + 3.76 ) = 6.3 copies

YCSB_PerfScore =
Sum(YCSB_Avg_Throughput[AI=1..YCSB_Valid_AIs])/YCSB_Ref_Throughput
where YCSB_Ref_Throughput= 33448.21

= (20220.5 + 20360.8 + 21809.7 + 21737.4)/ 33448.21 = 2.52


KMeans_PerfScore =
Sum(Normalized_KMeans_Avg_ComplTime[AI=1..KMeans_Valid_AIs])
where Normalized_KMeans_Avg_ComplTime[AI]=
Average(KMeans_Ref_ComplTime[HI]/KMeans_ComplTime[AI][HI])
and KMeans_Ref_ComplTime[HI=1..5]={76,102,128,154,180}

Sum(
Normalized_KMeans_Avg_ComplTime[1] = Average(154/133, 128/130, 180/130)
Normalized_KMeans_Avg_ComplTime[2] = Average(154/186, 180/177, 128/145)
Normalized_KMeans_Avg_ComplTime[3] = Average(180/138, 128/181)
Normalized_KMeans_Avg_ComplTime[4] = Average(102/147)) = 3.76
```

4.2.1 Discussion

The Performance Score formula captures the work done by a cloud. If a cloud does more work than the other cloud while meeting QoS thresholds, then it would produce a higher performance score than the other cloud.

SPEC Cloud IaaS 2018 Benchmark uses YCSB and K-Means workloads. The metrics used from these workloads are throughput for YCSB; and completion time for K-Means. In general, higher throughput is preferred and lower completion time is preferred. Since the metrics of two workloads have different units (operations per second for throughput, and seconds for completion time), these metrics cannot directly be combined in a single performance score. To produce a workload independent performance metric the raw throughput, and completion time metrics must be normalized using a reference cloud. Since it is difficult to come up with one definition of a reference cloud, the participating companies in the design of SPEC Cloud IaaS 2018 Benchmark ran baseline phase for YCSB and K-Means workloads in their clouds without applying any tunings and reported the results. The baseline results reported by the participating companies are then averaged to compute the reference platform throughput and completion time metrics. The throughput and completion results for a cloud are then normalized with the reference platform throughput and completion time results to compute a workload agnostic performance score.

The Performance Score formula may become dominated by one workload. This can happen due to two reasons. One, the number of application instances for one workload is higher than the other workload. The number of application instances for one workload may be higher, because each work runs independently. SPEC Cloud IaaS 2018 Benchmark puts a lower bound on the percentage of AIs for a workload used for metric computation, that is, the percentage of AIs from a workload must be greater than or equal to 40%. The other reason is that depending on the underlying cloud configuration (hardware and software), one workload may perform much better (e.g., higher throughout for YCSB, lower completion time for K-Means) than the reference platform. As a result, the performance score may be skewed towards one workload. This allows the benchmark to reflect benefits of improved cloud hardware and software over time.

4.3 Metric: Relative Scalability

Relative Scalability measures whether the work performed by application instances scales linearly in a cloud. When multiple AIs run concurrently, each AI should offer the same level of performance as that measured for an AI running similar work during the baseline phase when the tester introduces no other load. Relative Scalability is expressed as a percentage (out of 100).

A rough guideline for interpreting Relative Scalability results is shown below:

- Fair: 50-70%
- Good: 70%-80%
- Excellent: 80-100%

The aggregate Relative Scalability metric is an average of Relative Scalability metrics for the two workloads. It is expressed as a percentage out of one hundred. The higher the result is, the better. The formula used to calculate the metric:

```
Relative_Scalability = Average(YCSB_RelScalability, KMeans_RelScalability)

= Average(96.19, 95.50) = 95.8%


YCSB_RelScalability =
((.25 * Min(1,YCSB_Base_AI_ProvTime/YCSB_Avg_AI_ProvTime)) +
(.375 * Min(1,YCSB_Avg_Throughput/YCSB_Base_Throughput)) +
(.1875 * Min(1,YCSB_Base_99ReadLatency/YCSB_Avg_99ReadLatency)) +
(.1875 * Min(1,YCSB_Base_99InsertLatency/YCSB_Avg_99InsertLatency))
) * 100 =

((.25 * Min(1, 23.2/18.2)) +
(.375 * Min(1, 21032.1/21195.5)) +
(.375 * Min(1, 11.968/13.532)) +
(.375 * Min(1, 5.755/6.204))
) * 100 = 96.19


KMeans_RelScalability =
(.25 * Min(1,KMeans_Base_AI_ProvTime/KMeans_Avg_AI_ProvTime) +
 .75 * Average(
     Min(1,KMeans_Base_ComplTime[HI=1]/KMeans_Avg_ComplTime[HI=1]),
         Min(1,KMeans_Base_ComplTime[HI=2]/KMeans_Avg_ComplTime[HI=2]),
     Min(1,KMeans_Base_ComplTime[HI=3]/KMeans_Avg_ComplTime[HI=3]),
     Min(1,KMeans_Base_ComplTime[HI=4]/KMeans_Avg_ComplTime[HI=4]),
     Min(1,KMeans_Base_ComplTime[HI=5]/KMeans_Avg_ComplTime[HI=5]))
) * 100
{Where Min term is dropped if any KMeans_*_ComplTime[HI=n] is empty}

= (.25 * Min(1, 21.0/15.8) +
   .75 * Average(
        Min(1, 141/147)
        Min(1, 159/152)
        Min(1, 127/159.5)
        Min(1, 169/150)) * 100 = (.25 * 1 + .75 * .94 ) = 95.5%
```

4.3.1 Discussion

YCSB and K-Means application instances are configured to perform a set amount of work in SPEC Cloud IaaS 2018 Benchmark based on the parameter settings in the osgcloud_rule.yaml. As load on a cloud increases by adding more application instances, the workload specific metrics may be affected. In the case of YCSB AIs, the throughput across AI runs may decrease or the read and insert latencies may increase. For KMeans AIs the completion time of K-Means may increase. The time to provision a new application instance may increase relative to the metrics computed during the baseline phase.

The Relative Scalability metric for SPEC Cloud IaaS 2018 Benchmark measures the changes in throughput or response time or completion or provisioning time relative to baseline metrics for each application instance. In a perfectly scalable cloud, each AI during the scale-out phase would have the same amount of compute, storage, and network resources available and would perform at the same levels measured during the baseline phase. Since the degradation in this perfect cloud would be zero, it would report a Relative Scalability of 100%.

A new data set is generated within each AI run so that any caching affects due to same data set across AIs are minimized. The amount of work to be performed by an AI within every AI run is statistically similar but not exactly similar. To counter the effects of statistically similar data sets, the baseline results are average over a minimum of 25 runs. Similarly, throughput, completion time, and insert/read response time are averaged over all AI runs for all AIs. This averaging of results will reduce the degree of variability that arises due to dissimilarity in statistically similar data sets.

4.4 Metric: Mean Instance Provisioning Time

The Mean Instance Provisioning Time represents an average of provisioning time of all instances in all valid application instances. Since raw instance provisioning time has been a key metric reported in the literature and is easy to compare, it is reported as a separate metric. Each instance provisioning time measurement is the time from the initial provisioning request until a connection to port 22 (ssh) can be made by the harness and is tracked by the harness.

The average instance provision time is also subsumed under the AI provisioning time for each valid AI. The AI provisioning time includes the time to start the distributed application once the deployment of its individual instances has completed.

4.5 AI Provisioning Success Metric

This metric indicates the percentage of AIs that were successfully provisioned.

4.6 AI Run Success Metric

This metric indicates the percentage of AIs that had all successful runs.

4.7 Scale-out Start time Metric

This metric indicates the time at which the Scale-out phase of the benchmark was started by the harness.

4.8 Scale-out End time Metric

This metric indicates the time at which the Scale-out phase of the benchmark was stopped by the harness.

4.9 SPEC Cloud IaaS 2018 Benchmark Total Instances Metric

This metric indicates the total instances provisioned during the benchmark that belonged to application instances with one or more runs.

5 Limitations of the benchmark

SPEC Cloud IaaS 2018 Benchmark has the following limitations.

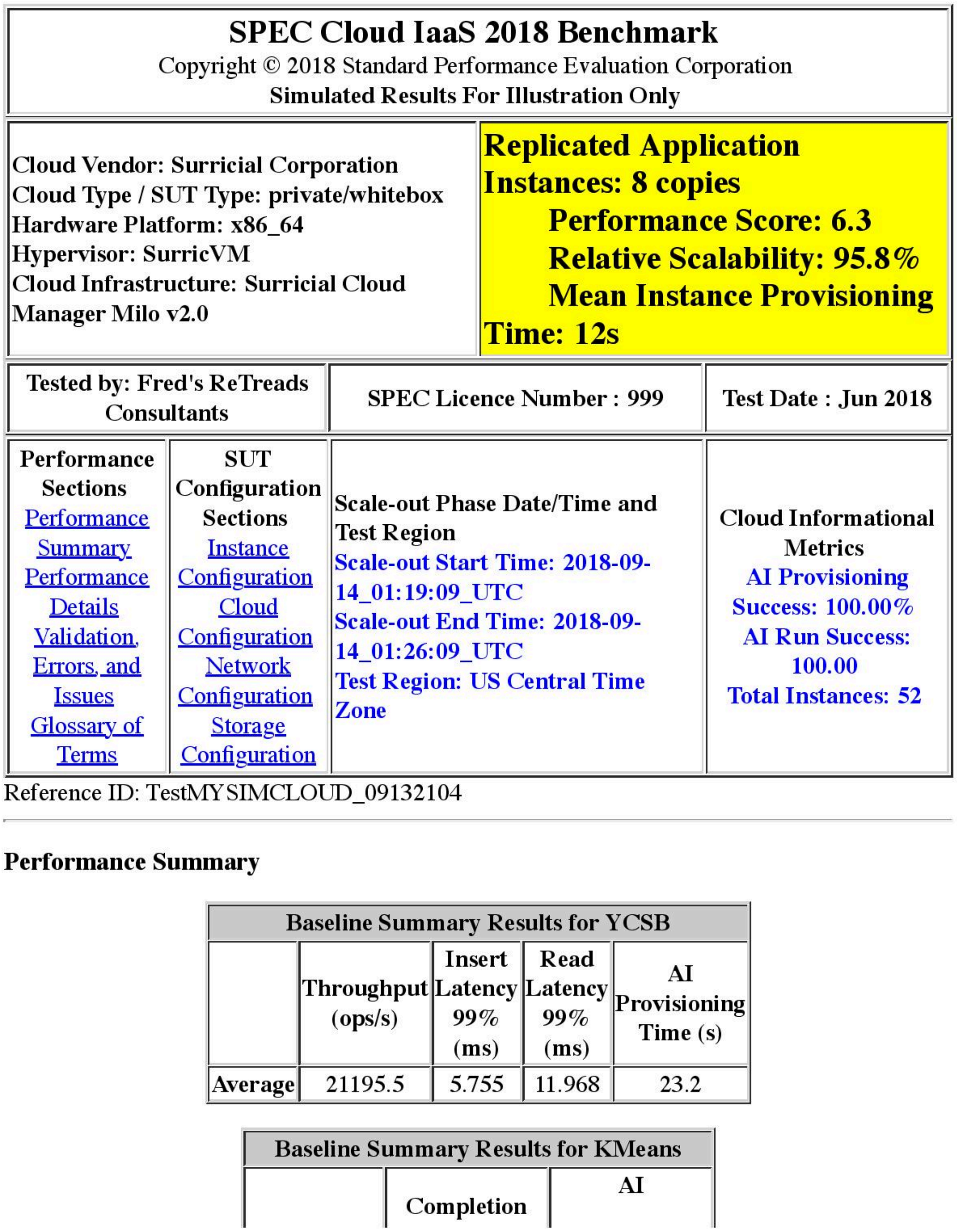
1. SPEC Cloud IaaS 2018 Benchmark is a benchmark for infrastructure-as-a-service clouds. It does not measure the performance of platform-as-a-service clouds or software-as-a-service clouds.
2. The benchmark does not explicitly measure CPU, memory, network or storage performance of an instance. The performance of these components is indirectly measured through YCSB and K-Means workloads that utilize Apache Cassandra and Apache Hadoop, respectively. A tester is free to choose instance configuration.
3. The arrival time of application instances is uniformly distributed between five and 10 minutes. Within a single AI, a burst of seven or six instances arrives for YCSB and K-Means workload, respectively. The scale-out driver does not adjust the arrival time of AIs during the benchmark run. One reason for not changing the arrival time of AIs is that a cloud may rate limit the number of instances that can be created within a unit time.
4. The size of the data set generated for YCSB and K-Means workloads may fit within the memory of the instances. Since each application instance of YCSB or K-Means generates a new data set from probability distributions, any caching across AIs due to the use of same data set is minimized. Nevertheless, data caching within the memory of an instance of AI can occur.
5. The work performed by each run across different application instances of the same workload is statistically similar but not exactly similar. This was a deliberate design decision to minimize any performance enhancement, which may result from performing an exactly similar work across application instances. Variable work for different workloads is not part of the SPEC Cloud IaaS 2018 Benchmark.
6. Client-server workloads (REST HTTP) (e.g., DayTrader or SPEC Web benchmark) require workload generators that are outside of the cloud and are not represented in this benchmark release.
7. SPEC Cloud IaaS 2018 Benchmark supports one or more tenants and does not require the use of multiple tenants. The number of tenants used is left to the tester since a cloud may solely focus on scalability and not multi-tenancy.

6 Full Disclosure Reports

SPEC Cloud IaaS 2018 Benchmark will generate the data set used to create the Full Disclosure Report (FDR) for each run. Part of the FDR will report statistics from the collected data set and the computed scores. The FDR will also provide enough detailed information on the SUT configuration to qualify as a *'Bill of Materials'* (BOM). The intent of the BOM is to enable a reviewer to confirm that the tested configuration satisfies the run rule requirements and to document the components used with enough detail to enable a customer to reproduce the tested configuration and obtain pricing information from the supplying vendors for each component of the SUT.

Appendix 1: SPEC Cloud IaaS 2018 Benchmark Example FDR

The following full disclosure report, generated using the benchmark's simulation mode, is for illustration only and is not representative of any specific cloud environment. The example metric calculations included in this document are based on the values taken from this simulated test run unless otherwise noted.



SPEC Cloud IaaS 2018 Benchmark

Copyright © 2018 Standard Performance Evaluation Corporation

Simulated Results For Illustration Only

Cloud Vendor: Surricial Corporation **Cloud Type / SUT Type: private/whitebox** **Hardware Platform: x86_64** **Hypervisor: SurricVM** **Cloud Infrastructure: Surricial Cloud Manager Milo v2.0**	**Replicated Application Instances: 8 copies** **Performance Score: 6.3** **Relative Scalability: 95.8%** **Mean Instance Provisioning Time: 12s**

Tested by: Fred's ReTreads Consultants	**SPEC Licence Number : 999**	**Test Date : Jun 2018**

Performance Sections Performance Summary Performance Details Validation, Errors, and Issues Glossary of Terms	**SUT Configuration Sections** Instance Configuration Cloud Configuration Network Configuration Storage Configuration	**Scale-out Phase Date/Time and Test Region** **Scale-out Start Time: 2018-09-14_01:19:09_UTC** **Scale-out End Time: 2018-09-14_01:26:09_UTC** **Test Region: US Central Time Zone**	**Cloud Informational Metrics** **AI Provisioning Success: 100.00%** **AI Run Success: 100.00** **Total Instances: 52**

Reference ID: TestMYSIMCLOUD_09132104

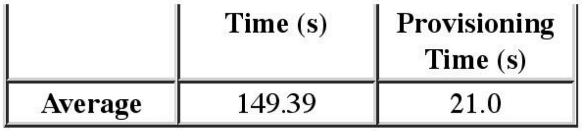
Performance Summary

Baseline Summary Results for YCSB

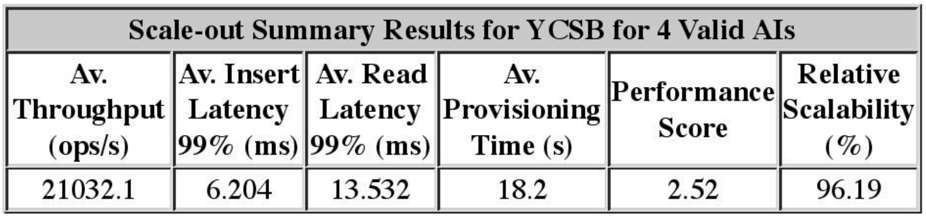
	Throughput (ops/s)	Insert Latency 99% (ms)	Read Latency 99% (ms)	AI Provisioning Time (s)
Average	21195.5	5.755	11.968	23.2

Baseline Summary Results for KMeans

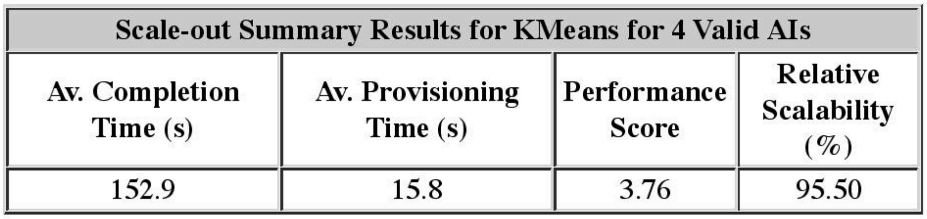
	Completion	AI



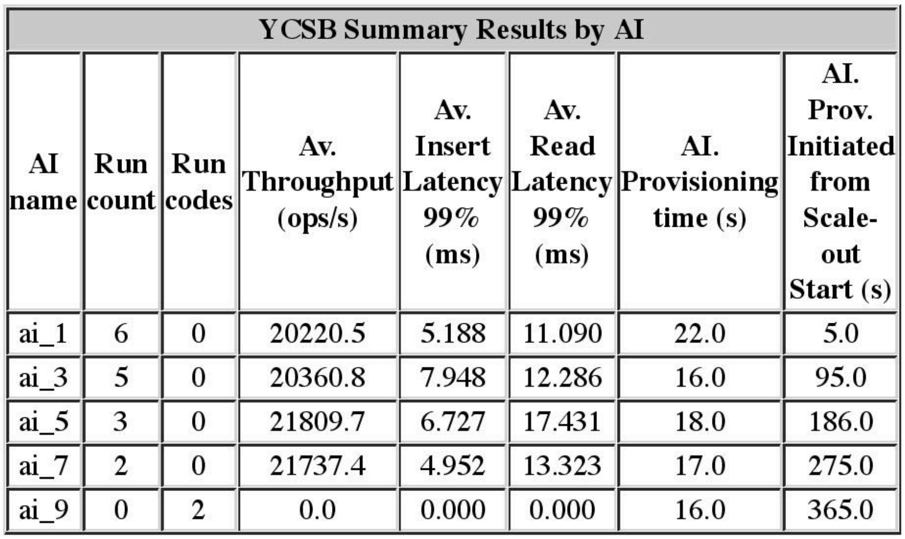
	Time (s)	Provisioning Time (s)
Average	149.39	21.0



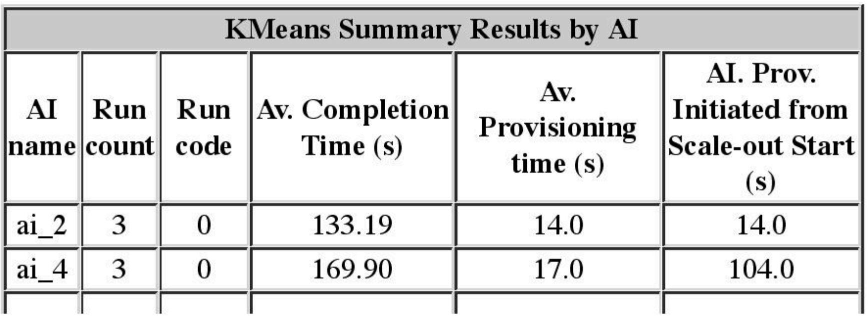
Scale-out Summary Results for YCSB for 4 Valid AIs					
Av. Throughput (ops/s)	Av. Insert Latency 99% (ms)	Av. Read Latency 99% (ms)	Av. Provisioning Time (s)	Performance Score	Relative Scalability (%)
21032.1	6.204	13.532	18.2	2.52	96.19



Scale-out Summary Results for KMeans for 4 Valid AIs			
Av. Completion Time (s)	Av. Provisioning Time (s)	Performance Score	Relative Scalability (%)
152.9	15.8	3.76	95.50



YCSB Summary Results by AI							
AI name	Run count	Run codes	Av. Throughput (ops/s)	Av. Insert Latency 99% (ms)	Av. Read Latency 99% (ms)	AI. Provisioning time (s)	AI. Prov. Initiated from Scale-out Start (s)
ai_1	6	0	20220.5	5.188	11.090	22.0	5.0
ai_3	5	0	20360.8	7.948	12.286	16.0	95.0
ai_5	3	0	21809.7	6.727	17.431	18.0	186.0
ai_7	2	0	21737.4	4.952	13.323	17.0	275.0
ai_9	0	2	0.0	0.000	0.000	16.0	365.0



KMeans Summary Results by AI					
AI name	Run count	Run code	Av. Completion Time (s)	Av. Provisioning time (s)	AI. Prov. Initiated from Scale-out Start (s)
ai_2	3	0	133.19	14.0	14.0
ai_4	3	0	169.90	17.0	104.0

ai_6	2	0	159.75	15.0	195.0
ai_8	1	0	147.32	17.0	284.0
ai_10	0	2	0	17.0	374.0

Validation, Errors, and Issues

- Stopping condition reason: Benchmark stopped because number of AIs reporting results reached the configured value. Result received from AIs:8, _rules_reported_ais:8, _ais_issued:10:NOK
- 0 YCSB AIs removed due to 60/40 rule
- 0 KMeans AIs removed due to 60/40 rule
- 1 YCSB AIs with zero runs
- 1 KMeans AIs with zero runs
- 0 YCSB AIs that failed to provision
- 0 KMeans AIs that failed to provision
- 0 YCSB AIs with provisioning qos violation
- 0 KMeans AIs with provisioning qos violation

Glossary of "Run code":

- 0 no error.
- 1 for all errors including AIs that were provisioned but did not report any result or reported results after QoS limits.
- 2 if an AI that was provisioned but did not report any runs when the benchmark terminated - the AIs that are provisioning towards the end of scale-out phase may have zero runs.
- 3 if an AI failed to provision.
- 4 if an AI provisioning was in progress when the benchmark terminated - the AIs issued towards the end of scale-out phase may not be fully provisioned.
- 5 if an AI provisioning QoS was violated.
- 6 if an AI was excluded due to reported_ais being reached.

Instance Configuration

Instance Configuration Hadoop Master	
Parameter	**Value**
instance_type_name	m1.large
cpu	2
memory	4GB

disk_size	40GB
disk_backed	block storage
image_name	cb_speccloud_kmeans
Notes	some text

Instance Configuration Hadoop Slave	
Parameter	**Value**
instance_type_name	m1.large
cpu	2
memory	4GB
disk_size	40GB
disk_backed	block storage
image_name	cb_speccloud_kmeans
Notes	some text

Instance Configuration YCSB	
Parameter	**Value**
instance_type_name	m1.large
cpu	2
memory	4GB
disk_size	40GB
disk_backed	block storage
image_name	cb_speccloud_cassandra
Notes	some text

Instance Configuration Cassandra	
Parameter	**Value**
instance_type_name	m1.large
cpu	2
memory	4GB
disk_size	40GB
disk_backed	block storage
image_name	cb_speccloud_cassandra
Notes	some text

Cloud Configuration

Cloud High Level Information and Availability	
Parameter	**Value**
Name	MYCLOUD
Type	private
Sut_Type	whitebox
Virtualized	SurricVM
Manager_Protocol	Surricial Cloud Manager
Manager_Version	Milo v2.0
Hardware_Availability	Apr-2018
Software/Service_Availability	Apr-2018
Geographic_Distribution	1
Notes	Cloud management deputy nodes reside at all data centers SUT exists in 3 different buildings within same campus Campus network backbone is infinet10000.

Cloud Under Test Software Levels	
Parameter	**Value**
Python_version	2.7.9
JVM_version	1.8.0_45
Hadoop_version	2.5.7
Cassandra_version	2.1.20
Notes	All software compiled locally from git source archives Compiler options - debug, 128bits, optimized Internal database set to emulate Cassandra API.

Benchmark Harness Node	
Parameter	**Value**
Vendor	Turbo Technologies
Platform	Turbo-2
CPU_Description	Eltin Xera J7-8409

CPU_Frequency_MHz	2800
RAM_Memory	128 GB, ECC, DDR3, 1600MHz RDIMM
Local_Storage	2x 1TB, SATA3, RAID0, 7200 RPM
Quantity	3
Availability_Date	May-2018
OS_Version	Debium 19
OS_Availability_Date	June-2017
Network_Adapter_1	1 x SulumUCom Corp. Model GCMX v2.4 Gigabit Ethernet.quad-port
Network_Adapter_2	1 x Turbo Ethernet 10GBb Model 17-04
Network_port_type_1	1000BASE-T
Network_port_type_2	10GBASE-SR
Network_speed_port_1	1 Gb/s
Network_speed_port_2	10 Gb/s
Notes	Turbo XL10000 256GB cache Some more info And even more.

Cloud Manager Node	
Parameter	**Value**
Hw_Vendor	Turbo Technologies
Hw_Platform	Turbo-2
CPU_Type	Eltin Xera J7-8409
CPU_Count	2
CPU_Cores_Per	4
CPU_Frequency_MHz	2700
RAM_Memory_GB	192
Primary_Cache_KB	64 KB I + 32 KB D per core, on chip
Secondary_Cache_KB	256 KB I+D per core, on chip
RAM_Memory_Type	ECC DDR3 1800 MHz
Hw_Availability_Date	May-2018
Local_Storage	2x2 TB HDD, RAID 0, 7200 RPM
Quantity	1
Virtualized	baremetal
	1 x SulumUCom Corp. Model GCMX v2.4 Gigabit

Network_Adapter_1	Ethernet.quad-port
Network_Adapter_2	1 x Turbo Ethernet 10GBb Model 17-04
Network_port_type_1	1000BASE-T
Network_port_type_2	10GBASE-SR
Network_speed_port_1	1 Gb/s
Network_speed_port_2	10 Gb/s
OS_Version	Debium 19
OS_Availability_Date	Dec-2017
Notes	Turbo XL10000 256GB cache Some more info And even more.

Cloud Computes Nodes	
Parameter	**Value**
type 1_Compute_nodes	5
type 1_Hw_Vendor	Lightspeed Tech., Inc.
type 1_Hw_Platform	Swisher
type 1_Quantity	10
type 1_CPU_Type	Eltin Zera
type 1_CPU_Count	24
type 1_CPU_Cores_Per	16
type 1_CPU_Frequency_MHz	3800
type 1_RAM_Memory_GB	192
type 1_Primary_Cache_KB	64 KB I + 32 KB D per core, on chip
type 1_Secondary_Cache_KB	256 KB I+D per core, on chip
type 1_RAM_Memory_Type	ECC DDR3 1800 MHz
type 1_Hw_Availability_Date	May-2018
type 1_Local_Storage	2x 2 TB SSD, SATA3, RAID 0, 7200 RPM
type 1_Virtualized	baremetal

type 1_Network_Adapter_1	1 x SulumUCom Corp. Model GCMX v2.4 Gigabit Ethernet.quad-port
type 1_Network_Adapter_2	1 x Turbo Ethernet 10GBb Model 17-04
type 1_Network_port_type_1	1000BASE-T
type 1_Network_port_type_2	10GBASE-SR
type 1_Network_speed_port_1	1 Gb/s
type 1_Network_speed_port_2	10 Gb/s
type 1_OS_Version	Debium 19
type 1_OS_Availability_Date	Dec-2017
type 1_Notes	Lightspeed TransWarp 4TB local SSD Some more info And even more.

Cloud Network Information

Cloud Network Information	
Parameter	**Value**
network 1_Technology	ethernet
network 1_Protocol	TCP/IP
network 1_Protocol_version	IPV4
network 1_Speed_Mbps	10000
network 1_Function	data
network 1_Notes	network_1 related notes
network 2_Technology	ethernet
network 2_Protocol	TCP/IP
network 2_Protocol_version	IPV4
network	1000

2_Speed_Mbps	
network 2_Function	management
network 2_Notes	network_2 related notes

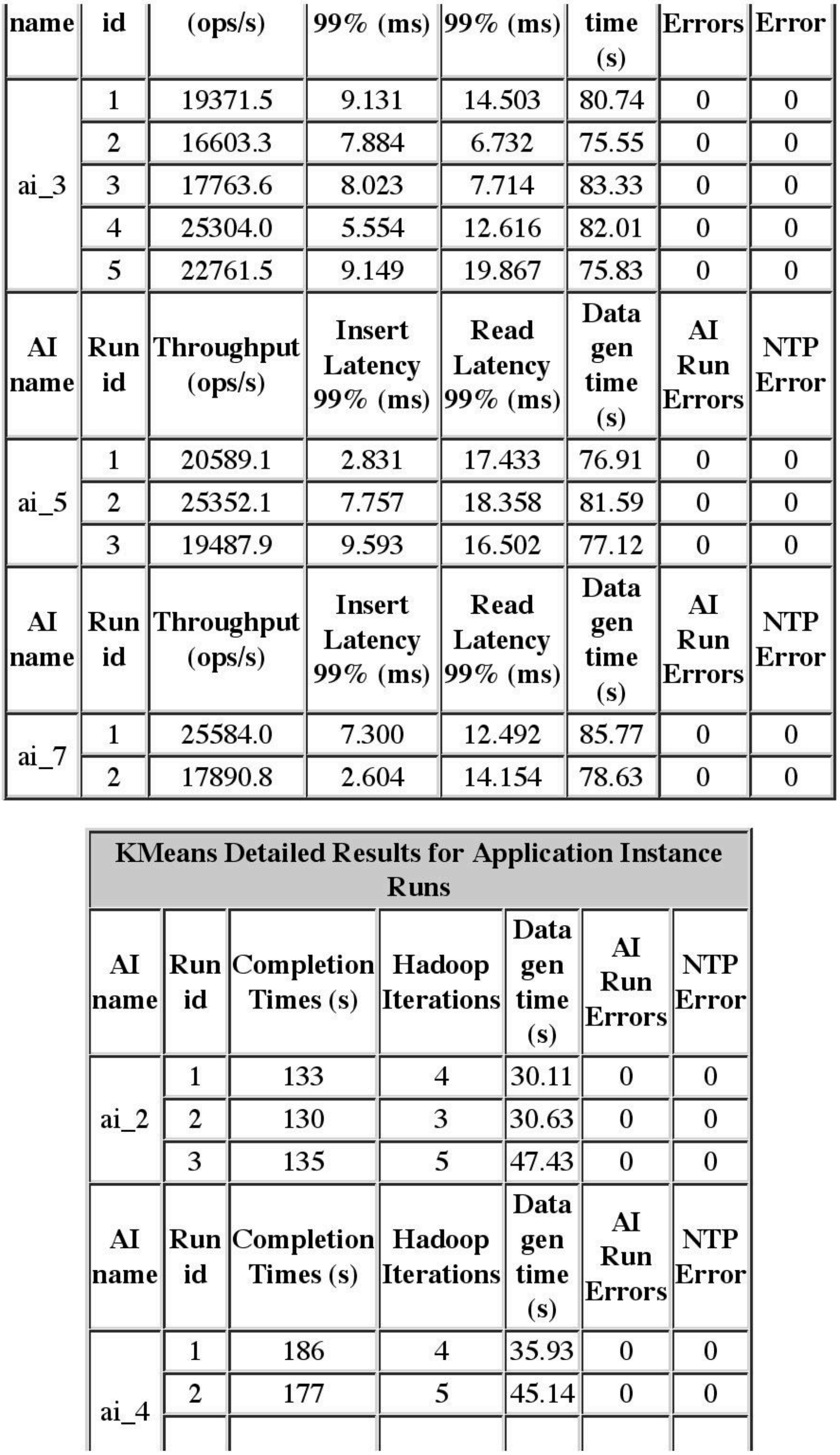
Cloud Schematic

Storage Configuration

Cloud Storage	
Parameter	**Value**
Storage 1_Attach_Type	network
Storage 1_Capacity	20 TB
Storage 1_Technology	HDD
Storage 1_Notes	Array has 1028 drive slots on SAS backplane Array has 12 redundant infinent network interfaces Space management integrated into Cloud Manager.

Performance Details

YCSB Detailed Results for Application Instance Runs							
AI name	**Run id**	**Throughput (ops/s)**	**Insert Latency 99% (ms)**	**Read Latency 99% (ms)**	**Data gen time (s)**	**AI Run Errors**	**NTP Error**
ai_1	1	24974.0	4.258	5.803	75.08	0	0
	2	17090.5	1.217	6.914	74.88	0	0
	3	23242.5	9.119	6.956	79.52	0	0
	4	18263.1	4.694	14.022	80.72	0	0
	5	16257.7	9.226	13.024	76.28	0	0
	6	21495.0	2.613	19.820	80.96	0	0
AI	**Run**	**Throughput**	**Insert Latency**	**Read Latency**	**Data gen**	**AI Run**	**NTP**



name	id	(ops/s)	99% (ms)	99% (ms)	time (s)	Errors	Error
ai_3	1	19371.5	9.131	14.503	80.74	0	0
	2	16603.3	7.884	6.732	75.55	0	0
	3	17763.6	8.023	7.714	83.33	0	0
	4	25304.0	5.554	12.616	82.01	0	0
	5	22761.5	9.149	19.867	75.83	0	0
AI name	**Run id**	**Throughput** (ops/s)	**Insert Latency** 99% (ms)	**Read Latency** 99% (ms)	**Data gen time** (s)	**AI Run Errors**	**NTP Error**
ai_5	1	20589.1	2.831	17.433	76.91	0	0
	2	25352.1	7.757	18.358	81.59	0	0
	3	19487.9	9.593	16.502	77.12	0	0
AI name	**Run id**	**Throughput** (ops/s)	**Insert Latency** 99% (ms)	**Read Latency** 99% (ms)	**Data gen time** (s)	**AI Run Errors**	**NTP Error**
ai_7	1	25584.0	7.300	12.492	85.77	0	0
	2	17890.8	2.604	14.154	78.63	0	0

KMeans Detailed Results for Application Instance Runs

AI name	Run id	Completion Times (s)	Hadoop Iterations	Data gen time (s)	AI Run Errors	NTP Error
ai_2	1	133	4	30.11	0	0
	2	130	3	30.63	0	0
	3	135	5	47.43	0	0
AI name	**Run id**	**Completion Times (s)**	**Hadoop Iterations**	**Data gen time (s)**	**AI Run Errors**	**NTP Error**
ai_4	1	186	4	35.93	0	0
	2	177	5	45.14	0	0

	3	145	3	30.18	0	0
AI name	**Run id**	**Completion Times (s)**	**Hadoop Iterations**	**Data gen time (s)**	**AI Run Errors**	**NTP Error**
ai_6	1	138	5	39.20	0	0
	2	181	3	46.90	0	0
AI name	**Run id**	**Completion Times (s)**	**Hadoop Iterations**	**Data gen time (s)**	**AI Run Errors**	**NTP Error**
ai_8	1	147	2	40.91	0	0

Glossary

Glossary of Terms

For questions about this result, please contact the tester. For other inquiries, please contact info@spec.org

Simulated Results For Illustration Only

Appendix 2: References

Keyword	**Bibliography Information**
CloudWhitePaper	SPEC OSG Cloud Working Group whitepaper https://www.spec.org/osgcloud/docs/osgcloudwgreport20120410.pdf
CBTOOL	Cloud Rapid Experimentation and Analysis Tool (CBTOOL) https://github.com/ibmcb/CBTOOL
	Dumitras, T., & Shou, D. (2011). *Toward a Standard Benchmark for Computer Security Research*. Carnegie Mellon University https://www.umiacs.umd.edu/~tdumitra/papers/BADGERS-2011.pdf
NISTPub145	Mell, P., & Grance, T.; *NIST Definition of Cloud Computing*, Publication No. 145, 2011 http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf
ElasticityICAC	Herbst, N., Kounev, S., Reussner, R. Elasticity in Cloud Computing: What it is, and What it is Not. In Proceedings of the 10th International Conference on Autonomic Computing (ICAC 2013), San Jose, CA, June 24-28 https://sdqweb.ipd.kit.edu/publications/pdfs/HeKoRe2013-ICAC-Elasticity.pdf
HiBenchIntro	**Hadoop Benchmark Suite (HiBench)** documentation https://github.com/intel-hadoop/hibench/#overview
KMeansClustering	http://en.wikipedia.org/wiki/K-means_clustering
ApacheCassandra	http://cassandra.apache.org/
ApacheHadoop	https://hadoop.apache.org/
ApacheMahout	http://mahout.apache.org/
YCSBWhitePaper	**Yahoo! Cloud Serving Benchmark (YCSB) Results Report**. Cooper, Brian; version 4, 2010 http://www.brianfrankcooper.net/home/publications/ycsb.pdf
CassandraSeeds	http://wiki.apache.org/cassandra/FAQ#seed

CassandraAddDataNodes	http://docs.datastax.com/en/cassandra/2.0/cassandra/operations/ops_add_node_to_cluster_t.html
EC2PerfVariations	http://www.infoworld.com/article/2613784/cloud-computing/benchmarking-amazon-ec2--the-wacky-world-of-cloud-performance.html

Revision Date: Jan. 25, 2018