SPEC Cloud™ IaaS 2016 Benchmark Run and Reporting Rules

Table of Contents

1. Introduction

The SPEC Cloud™ IaaS 2016 Benchmark is targeted towards measuring the performance of a public and private infrastructure-as-a-service (IaaS) clouds. In some cases, the IaaS cloud under test may be under the complete control of tester, including all hardware and software. In other cases, the cloud under test may not be under the complete control of tester. The underlying design and choices are documented in the accompanying design document.

1.1 Trademark

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

1.2 Benchmark Philosophy

The general philosophy behind the SPEC Cloud IaaS 2016 Benchmark is to provide metrics that are reproducible by independent parties. The following attributes are expected:

- Proper use of the SPEC benchmark tools as provided.
- Availability of an appropriate full disclosure report.
- Support for all of the appropriate hardware and software.

Furthermore, SPEC expects that any public use of results from this benchmark suite shall be for the SUT and configurations that are appropriate for public consumption and comparison. Thus, it is also expected that:

- The SUT used to run this benchmark must provide a suitable environment for running a private or public IaaS cloud.
- Optimizations utilized must improve performance for a larger class of workloads than those defined by this benchmark suite.
- The SUT and configuration is generally available, documented, supported, and encouraged by the vendor(s) or provider(s). See the SPEC Open Systems Group Policies and Procedures Document. (http://www.spec.org/osg/policy.html)

1.3 Fair Use of SPEC Cloud IaaS 2016 Benchmark Results

Consistency and fairness are guiding principles for SPEC. To help ensure that these principles are met, any organization or individual who makes public use of SPEC benchmark results must do so in accordance with the SPEC Fair Use Rule, as posted at http://www.spec.org/fairuse.html. Fair Use clauses specific to SPEC Cloud IaaS 2016 Benchmark are covered in http://www.spec.org/fairuse.html#cloud_iaas2016.

In the case where it appears that these rules have not been adhered to, SPEC may investigate and request that the published material be corrected.

1.4 Research and Academic Usage

Please consult the SPEC Fair Use Rule on Research and Academic Usage at http://www.spec.org/fairuse.html#Academic.

1.5 Caveat

SPEC reserves the right to adapt the benchmark codes, workloads, and rules of SPEC Cloud IaaS 2016 Benchmark as deemed necessary to preserve the goal of fair benchmarking. SPEC notifies members and licensees whenever it makes changes to this document and renames the metrics if the results are no longer comparable.

Relevant standards are cited in these run rules as URL references, and are current as of the date of publication. Changes or updates to these referenced documents or URLs may necessitate repairs to the links and/or amendment of the run rules. The most current run rules are available at the SPEC Cloud IaaS 2016 Benchmark web site. SPEC notifies members and licensees whenever it makes changes to the suite.

1.6 Definitions

The following table establishes the terms and definitions used by this document. The design documentation provides the definition for these terms. They are copied here for easy reference.

Term	Definition
Cloud Provider	An organization that provides cloud services to customers.
Cloud Consumer	A person or organization that is a customer of a cloud; note that a cloud customer may itself be a cloud and that clouds may offer services to one another.
Infrastructure as a Service (IaaS)	The *Cloud Provider* gives the *Cloud-Consumer* the capability to the provision processing, storage, network, and basic computing resources. They can also deploy and run arbitrary operating systems. The *End-Consumer* does not manage or control the underlying physical cloud infrastructure, but has control over the operating system, assigned storage, deployed applications, and limited control of select networking components (e.g., host firewalls).
SUT	Infrastructure-as-a-service cloud under test. This includes all hardware, network, base software and management systems used for the IaaS cloud.

Private Cloud	The cloud infrastructure is provisioned for exclusive use by a single organization comprising single or multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises.
Public Cloud	The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider.
Whitebox Cloud	The SUT's exact engineering specifications including all hardware and software are known and under the control of the tester. This will typically be the case for private clouds.
Blackbox Cloud	A cloud-provider provides a general specification of the SUT, usually in terms of how the cloud consumer may be billed. The exact hardware details corresponding to these compute units may not be known.
Instance	An instance is an abstracted execution environment which presents an operating system (either discrete or virtualized). The abstracted execution environment presents the appearance of a dedicated computer with CPU, memory and I/O resources available to the operating system. In SPEC Cloud, an instance consists of a single OS and the application software stack that supports a single SPEC Cloud component workload. There are several methods of implementing an instance, including physical machines, virtual machines, or containers. An instance is created or destroyed using an API provided by an IaaS cloud.
Instance Image	An image on the disk from which an instance is provisioned. Common formats for instance image include QCOW2 (Qemu copy on write 2), RAW, or AMI (Amazon machine image)
Physical machine	A set of connected components consisting on one or more general purpose processors (CPUs) with dedicated memory, network connectivity, and mass storage either local (disk) or remote (network attached, block storage). An example would be a multi-core server with 4 GB of memory, 250 GB disk, and 1 Gb/s network adapter.
Application Instance (AI)	A group of instances created to run a single workload together. An application instance comprises a workload driver instance and set of instances, which are stressed by the workload driver. SPEC Cloud IaaS 2016 benchmark uses multiple application instances during specific phases to determine elasticity and scaling.

AI_run	AI_run indicates the creation of dataset, running of load generator, and collection of results. A valid application instance created during elasticity + scalability phase will have one or more runs.
Provisioning Time	Within this document, this will be divided into two measurements: ***Instance***: The time interval between request to create a new instance until that instance responds to a *netcat* probe on port 22. ***Application instance***: The time interval betweenrequest to create a new instances as part of an application instance creation until the associated instances are to ready accept workload requests. YCSB (Yahoo! Cloud Serving Benchmark) is ready when all nodes in the underlying database are part of the Cassandra cluster. K-Means is ready when the all nodes in HDFS are ready and part of the HDFS cluster.
Response Time	The time between when the work item request is issued until the corresponding completion condition. An example is YCSB "*Latency*" metric.
Variability	The difference in workload and benchmark metrics between different runs of the benchmark. Typically, variability arises due to factors such as multi-tenancy and time of day execution, and may be more pronounced on public clouds.
Quality of Service (QoS)	The minimum percent (e.g., 90%) of collected values that complete within a predefined threshold in each measurement category. The Run and Reporting Rules document contains specific QoS limitations for each workload and benchmark.
CBTOOL	An open-source benchmark harness that instantiates and destroys application instances and collects metrics.
Cloud API	An application programming interface exposed by a cloud to perform certain operations such as instantiate or delete resources (e.g., instance, storage, network).
Cloud Adapter	Cloud adapter is a software component invoked by CBTOOL to perform cloud-specific operations for running the benchmark. Examples of these operations invoked by CBTOOL include creating or deleting instances or storage, listing instances, images, and networks. These operations are typically exposed by a cloud API, and can vary from one cloud to the other.
Benchmark phases	Benchmark has two phases, namely baseline and elasticity + scalability.

Baseline phase	In baseline, peak performance for each workload is determined in five separate test runs. During each workload run, instances are provisioned, data set is generated, load generator is started, results are accumulated, and the instances are destroyed. The workloads are run in a sequential fashion. Data from the baseline phase is used to establish parameters for the Elasticity + Scalability phase.
Baseline driver	Baseline driver runs the baseline phase of the benchmark.
Elasticity + scalability phase	In the Elasticity + Scalability phase, new application instances are created, and they run the workloads concurrently to determine the elasticity and scalability metrics. The benchmark reports are generated at the end of elasticity + scalability phase.
Elasticity driver	Elasticity driver runs the elasticity + scalability phase of the benchmark.

For clarity, the following abbreviations are used throughout this document.

Acronym	Definition
IaaS	Infrastructure as a Service
SUT	System Under Test (Infrastructure-as-a-service cloud under test)
FDR	Full Disclosure Report
CBTOOL	An open-source benchmark harness that instantiates and destroys application instances and collects metrics.
OSG	SPEC's Open Systems Group
YCSB	Yahoo! Cloud Serving Benchmark
K-Means	K-Means clustering algorithm benchmark
HiBench	Intel's benchmark suite used for K-Means benchmark

2.0 Running the SPEC Cloud IaaS 2016 Benchmark

The following run rules apply to all services, hardware and software components used to produce a compliant benchmark result, including the SUT, network, and harness.

2.1 Benchmark Environment

Figure 1 shows the logical architecture of SPEC Cloud IaaS 2016 Benchmark.

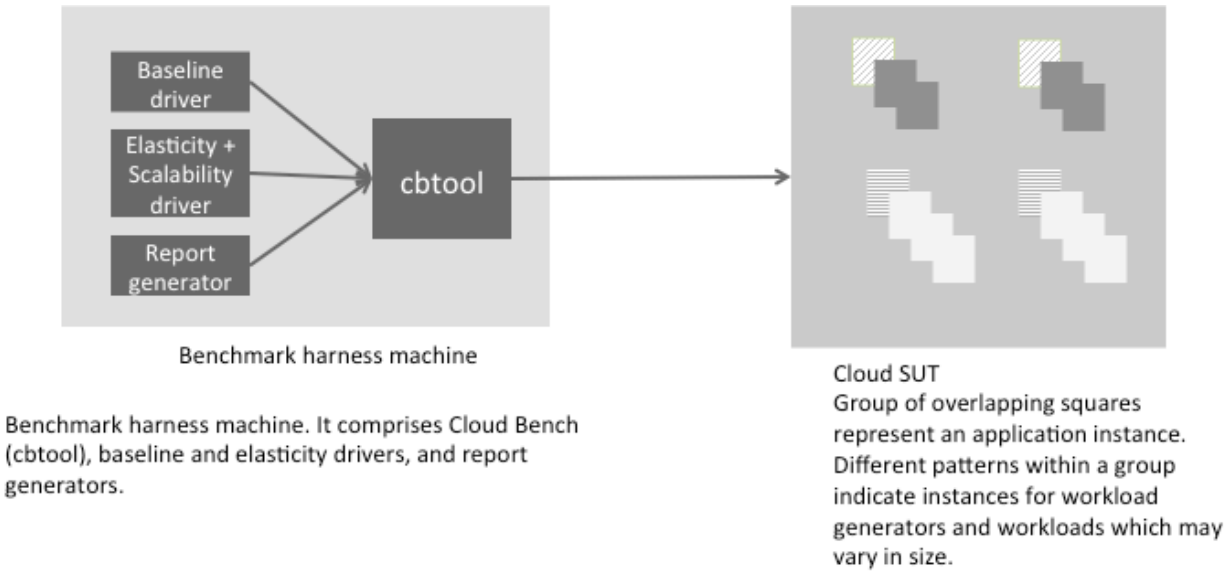

Figure 1 Logical architecture of SPEC Cloud IaaS 2016 Benchmark

The following run rules apply to all services, hardware used to produce a compliant benchmark result, including the SUT, network, and clients.

2.1.1 General Cloud Requirements

a. The defined SUT must provide a suitable environment for an IaaS cloud or provide a publicly accessible IaaS Service.
b. The SUT must have the ability to import an instance image or store a snapshot of an instance as an instance image, and provision from that instance image.
c. The SUT must have the ability to launch an instance through API.
d. The SUT must provide to the Cloud Consumer the ability to establish access for the tenant/user and associated API's using secure transport protocols.

e. Open Source Applications that are outside of a commercial distribution or support contract must adhere to the Rules on the Use of Open Source Application covered in section 3.2.3 below.

2.1.2 Additional White Box Cloud Requirements

In addition to the general cloud requirements above, white box clouds must also meet the following requirements:

a. The SUT must be composed of a minimum of 3 physical machines accessible by network to the benchmark harness machine.
b. The SUT must consist of sufficient processors, memory, network connectivity and storage (local or remote) to support running multiples of the two required Application Instances (AIs) concurrently, in compliance with these run rules. The User Guide provides guidance on the instance sizing information.
c. No components that are included in the base configuration of the SUT, as delivered to customers, may be removed from the SUT.
d. No instances can be running inside SUT other than those for administrative operations such as "monitoring" before the start of a compliant run. Having any instances running may adversely affect performance and reproducibility. It is the tester's responsibility to ensure that such instances will not improve performance of workloads. Such instances configuration and the processes running inside these instances must be appropriately reported in the FDR report (e.g., in the Notes section).

2.1.3 Additional Black Box Cloud Requirements

In addition to the general cloud requirements above, black box clouds must also meet the following requirements:

a. No instances must be running inside the ‘account’ used to run the benchmark, except for benchmark harness (***cbtool***) instance.

2.1.4 Instance Requirements

a. Because timing data will be measured on the SUT, the system clocks of all instances listed in the FDR and ***cbtool*** must be mutually synchronized to the same master NTP server to avoid drift. This could either be an NTP server running on benchmark harness machine or another NTP server. The ***cbtool*** will measure the time drift at the end of each phase. ***cbtool*** machine and instances may be configured with UTC time zone. For white-box clouds, it is recommended that the timezone of the physical servers be in UTC and they configured with a common NTP server. If a time drift is between benchmark harness machine and instances is detected, the results will follow the guidelines for handling errors as documented in Section 3.1.2 SPEC Cloud IaaS 2016 Benchmark Handling Provisioning and AI Run Errors during Elasticity + Scalability Phase.

b. The instance configuration selected for workloads during the baseline phase cannot be changed during the elasticity and scalability phase.
c. The load generators for YCSB and K-Means will be run as part of the SUT.
d. The SUT will not employ any tools for caching results returned by the workloads (e.g., memcache)
e. Instances can be booted from a block-storage device or from an ephemeral disk.

2.1.5 Network Requirements

a. The SUT must conform to the appropriate networking standards, and must utilize variations of these protocols to satisfy requests made during the benchmark.
b. The value of TCP TIME_WAIT must be at least 60 seconds (i.e., When a connection between the SUT and ***cbtool*** enters TIME_WAIT it must stay in TIME_WAIT for at least 60 seconds).
c. Private networks used to manage instances may use custom or non-compliant protocols and capacity settings when instantiating or decommissioning instances. However, *benchmark workload traffic is not permitted* on these administrative networks using custom or non-compliant protocols.)
d. The connections between a SPEC Cloud IaaS 2016 Benchmark ***cbtool*** and the SUT must not use a TCP Maximum Segment Size (MSS) greater than 1460 bytes. This needs to be accomplished by platform-specific means outside the benchmark code itself. The method used to set the TCP MSS must be disclosed. MSS is the largest "chunk" of data that TCP sends to the other end. The resulting IP datagram is normally 40 bytes larger: 20 bytes for the TCP header and 20 bytes for the IP header resulting in an MTU (Maximum Transmission Unit) of 1500 bytes.
e. Inter-instance communication has no restriction for MSS.

2.1.6 Storage

a. There is no restriction on the type of storage that instances can use. However, the instantiation of a storage object used by one instance must be separate from the storage objects used by other instances. For example, a virtual disk device used inside an instance (e.g., /volume1) cannot be used inside any other instance for storing data.

2.1.7 Instance Images

a. There is no restriction on the instance image format. QCOW2 and RAW are examples of image formats.

2.1.8 Benchmark Harness and Load Generators

a. SPEC Cloud IaaS 2016 Benchmark harness machine must run, outside of the SUT for whitebox clouds. For blackbox clouds, the benchmark harness machine can run in the same data center as the one under test.

2.1.9 Log Files

The SUT logs from the workloads are required per the individual workload run rules:

a. Until the review process concludes, the submitter must keep the required benchmark log files for the elasticity + scalability phase for the SUT from the run and make them available upon request so that the reviewers can validate the results if needed. The benchmark kit does not collect logs from instances at the end of elasticity + scalability phase.
b. The logs must be saved from all instances created during the elasticity + scalability phase for the following components as applicable:
    - Cassandra logs (/var/log/Cassandra/* - for YCSB application instances only)
    - YCSB logs (under /tmp)
    - Hadoop logs (/usr/local/hadoop/logs - for K-Means application instances only)
    - Syslog (for all instances)

2.2 Software Environment

The software environment for the SPEC Cloud IaaS 2016 Benchmark consists of several layers that span the cloud infrastructure, from the cloud management stack down through the individual workloads instances and their application stack.

For white-box clouds, the tester is responsible for ensuring that higher layers of software such as the cloud management stack and virtualization stack (if used) are compliant with these run and reporting rules (as described in Section 2.1.2 Additional White Box Cloud Requirements) including all availability and support requirements.

The tester is responsible for building a single loadable instance image for each of the workloads.; The tester is responsible for ensuring that the software versions in use for instance images are compliant with these run and reporting rules including all support and availability requirements such as the Rules on the Use of Open Source.

This operating system image for the benchmark harness machine or instances must have the SPEC Cloud IaaS 2016 Benchmark kit installed which contains the following software required to produce complaint test results:

- Benchmark harness (***cbtool*** + drivers and reporters)
- HiBench (K-Means)
- YCSB

The table below contains the additional software that must also be installed on the instances. The versions specified below for Hadoop and Cassandra must be used for SPEC Cloud IaaS 2016 Benchmark. These versions were tested on CentOS / Red Hat 7.1 and Ubuntu 14.04 operating systems. The source code and appropriate operating system packages for these versions ship with the kit.

If these versions are incompatible with a new operating system, hardware architecture, or kernel, the tester must check if there are any updates available to these versions which will allow the use of these versions on a new operating system, hardware architecture, or kernel. In the event, that no such updates are available, a tester may update to new version as follows:

- The specific version used must meet the availability requirements as defined in Section 3.3 General Availability.
- The baseline performance of a new version must be no more than 110% for YCSB throughput and greater than 90% of YCSB Insert and Read 99% latency and K-Means completion time.
- The tester must submit baseline results to the subcommittee for old and new versions of Hadoop and Cassandra for review, prior to a submission. These tests must be run in an environment compatible with both versions and under equivalent conditions.
- Any changes to YCSB or KMeans are not acceptable to accommodate version changes in Cassandra or Hadoop.

Software Component	Minimum Version
Java engine	Java 7

Apache Hadoop	2.7.1 (July 2015) Note: Hadoop can be downloaded and compiled to match the local hardware, if needed.
Apache Cassandra (NoSQL database)	2.1.11 (released Oct 2015).
Python	v. 2.7 or higher (needed for ***cbtool*** and other scripts) (released on July 2010; Python 2.7.10 May 2015)

2.3 Workloads

SPEC Cloud IaaS 2016 Benchmark uses two workloads; YCSB and K-Means.

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

The SPEC Cloud IaaS 2016 Benchmark runs the YCSB load generator in one instance and the Apache Cassandra seeds in six instances. Together, these seven instances comprise the YCSB application instance for the benchmark.

Figure 2 shows the architecture of YCSB application instance in the SPEC Cloud IaaS 2016 Benchmark. The YCSB driver instance generates load on the Cassandra cluster. Both YCSB load generator instance and Cassandra instances are part of the SUT.

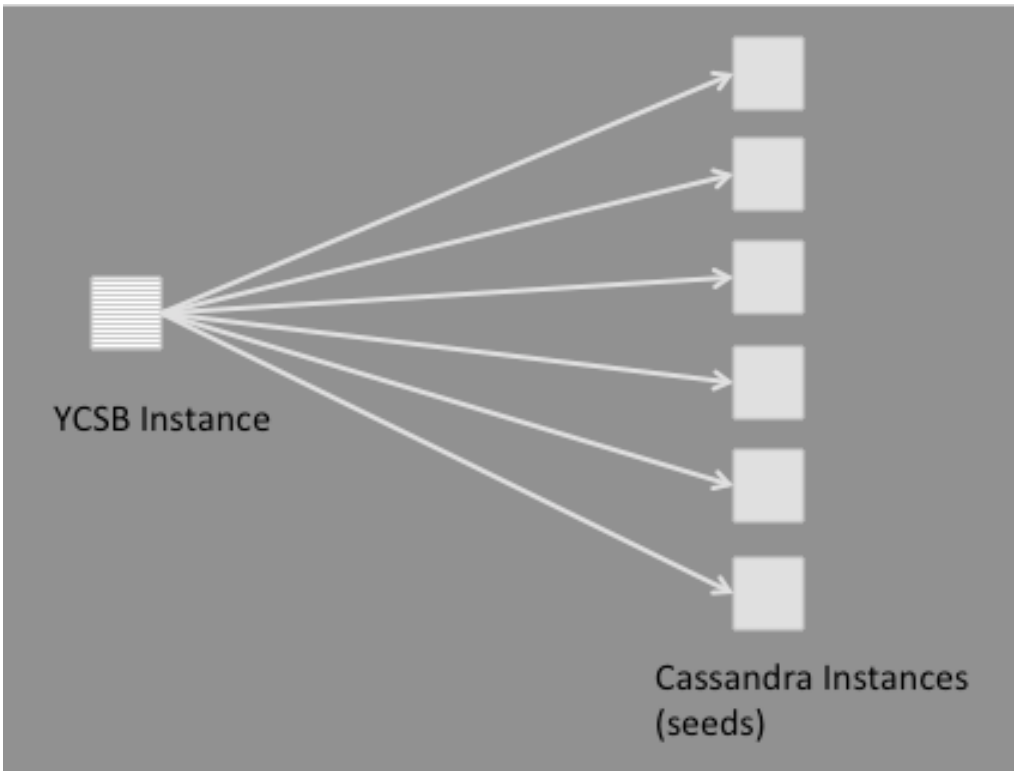

Figure 2 YCSB / Cassandra application instances in SPEC Cloud IaaS 2016 Benchmark

2.3.1.1 Workload driver

YCSB driver instance generates load on the Cassandra cluster. The tester configures the YCSB driver instance with a certain number of threads for generating load on the Cassandra cluster.

The following table shows the parameters used for YCSB application instance. The symbol CB indicates that the parameter is automatically set by ***cbtool***.

Description	Modification Condition	Setting values / Rule
Cassandra template file: ***cassandra.yaml***	Fixed	Instances (virtual or physical machines) cannot share a common logical storage device.
	Configurable	Consider using different disks for: *CommitLogDirectory* *DataFileDirectories* A compliant run may use storage not located in the same disk as the operating system. Cassandra log or data files cannot be shared between instances. There is no limit on the number of disks per instance and how Cassandra directories are laid out on these disks.
Database replication factor	Fixed (CB)	**Replication factor** must be 3 across the Cassandra cluster and workload database and must be in use.
YCSB workload file	Fixed (CB)	recordcount=1000000 (size of each record 1KB) operationcount=1000000 workload=com.yahoo.ycsb.workloads.CoreWorkload readallfields=true readproportion=0.95 updateproportion=0 scanproportion=0 insertproportion=0.05 requestdistribution=latest
	As set by the tester.	threadcount=8 (default in osgcloud_rules.yaml file)

Description	Modification Condition	Setting values / Rule
YCSB: Num of instances	Fixed (CB)	1 For every application instance, YCSB must be a distinct instance from Cassandra instances. It must be configured to send requests to all six instances of Cassandra cluster.
Cassandra: Number of instances	Fixed (CB)	6. All six instances must be configured as Cassandra seeds.
Cassandra partitioner	Fixed (CB)	`org.apache.cassandra.dht.RandomPartitioner`
Database placement strategy	Fixed (CB)	`org.apache.cassandra.locator.SimpleStrategy`
YCSB plug-ins	Fixed (0.4.0)	Must use the version distributed in the benchmark kit.

2.3.1.2 YCSB metrics

Following metrics are reported:

- Throughput (unit: ops/sec)
- 99th percentile of insert response time (unit: milliseconds)
- 99th percentile of read response time (unit: milliseconds)

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

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 2016 Benchmark. Figure 3 shows the logical architecture of K-Means application instance in SPEC Cloud IaaS 2016 Benchmark.

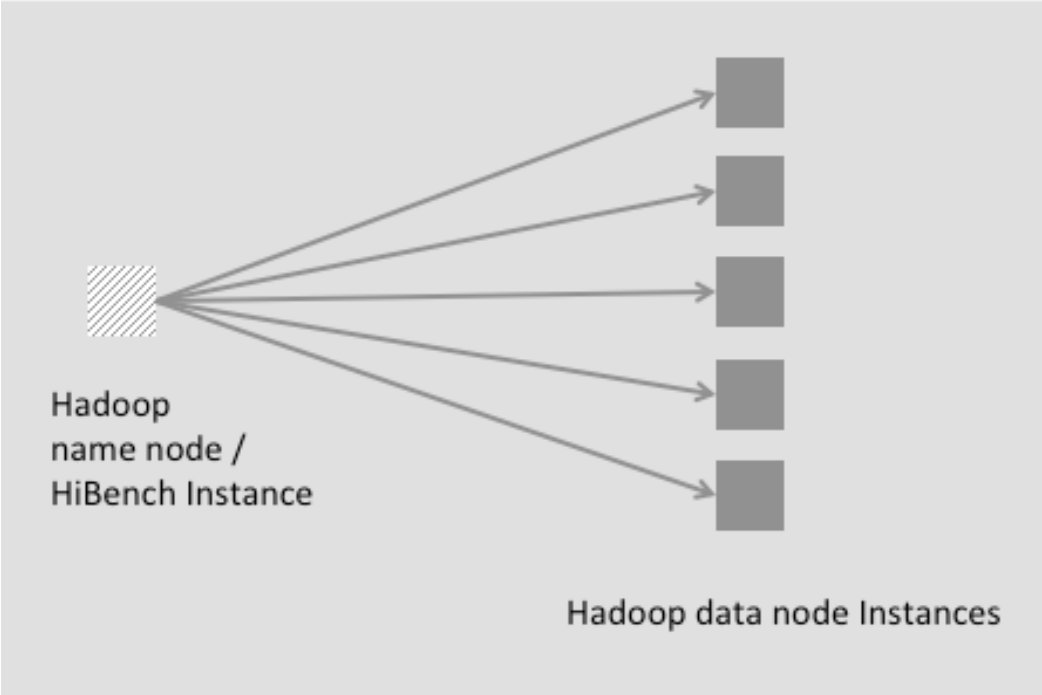

Figure 3 K-Means application instance

2.3.2.1 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.

The following table shows the parameters used for K-Means application instance. The symbol CB indicates that the parameter is set automatically by ***cbtool***.

Description	Condition	Setting values / Rule
Hi-Bench K-Means parameters	Fixed (CB)	dimensions (d) = 20 num_of_samples (s) = 1,000,000 num_of_clusters (k) = 5 max_iteration (i) = 5 samples_per_inputfile= 500,000 convergence_delta = 0.5 canopy clustering (-cl)

Description	Condition	Setting values / Rule
Hadoop: Number of Instances	Fixed (CB)	• Master instance = 1; Must run ***namenode*** and ***resourcemanager (yarn), jobhistory server, and nodemanager*** services • Slave instance = 5; Each must run ***datanode*** and ***nodemanager*** services
Hadoop: Configuration	Fixed (CB)	• DFS replication must be 3.
	Configurable	• Data compression is permitted, but must be configured solely using parameters in the ***hdfs-site.xml*** and ***mapred-site.xml*** configuration files.

The NUM_CLUSTER parameter indicates that the maximum number of clusters (or K) to be found is set to five. The DIMENSIONS parameter value 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.

By default, no compression is enabled for Hadoop.

2.3.2.2 K-Means Metrics

Following metrics are reported:

- Completion time (unit: seconds)

2.4 Measurements

The following measurements will be taken or extracted from each application instance.

Measured Value	K-Means	YCSB	Description and Conditions
Instance Provisioning Time	x	x	Time between issuing an instance creation until the instance responds on port 22 (via the *netcat* command).
Application Instance Provisioning Time	x	x	Time between issuing an instance creation until the application instance is ready to accept workload assignment
Response Time		x	Time between submitting request and time results were returned to the client.
Database generation time	x	x	Time needed to generate the database for an AI run.
Completion time	x	x	Time needed to complete assigned workload
Throughput		x	As reported by YCSB

2.5 Benchmark Setup

The following description assumes that the tester has already configured the cloud, its physical and/or virtual networks, and associated storage.

The tester installs the SPEC Cloud IaaS 2016 Benchmark software onto one or more hosts that can be co-located within the same cloud as the SUT (black-box cloud), or on an external network to the SUT (white-box cloud). The tester ensures that ***cbtool*** can connect to its cloud through a cloud adapter. ***cbtool*** ships with adapters for various clouds. The guidelines for submitting results with a cloud adapter that is not shipped with the benchmark are covered in Section 4.8 Adding a New Cloud Adapter for Cloud Bench.

The tester then either installs and configures two ***instances*** for YCSB/Cassandra and HiBench/Hadoop by following the instructions in the user guide. If a tester configures the workload instances, it must take a snapshot of these in an ***instance image***. The tester then ensures that it can create an ***application instance*** for YCSB and K-Means workloads, destroy them, and see their results in the ***cbtool*** UI. Once a tester sees the results in ***cbtool*** UI, it can start executing the benchmark phases. A tester may run the baseline phases couple of times as a trial run before starting a compliant baseline run. Please refer to the User Guide for details on how to run the baseline phase.

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 2016 Benchmark does not place any limitation on how many tenants are defined for white-box cloud. It is up to the tester to configure the number of tenants. The metrics are aggregated across all application instances across all tenants.

2.6 Baseline (Automated)

SPEC Cloud IaaS 2016 Benchmark baseline driver instructs the ***cbtool*** through its API to create and destroy a single application instance for each workload five times. ***cbtool*** instantiates the data generation in each application instance and then starts the load generators. At the end of each of the five runs of a workload, the baseline driver collects the instance and cloud supporting evidence. If there is no error in the five runs of an application instance as reported by ***cbtool***, the baseline result is considered valid.

cbtool records the settings and results associated with the most optimal performance, i.e. for YCSB, average of throughput, 99th percentile of insert and read response time; and for K-Means, completion time. In addition, for both workloads, ***cbtool*** records the average of application instance provisioning time per workload, average time for an application instance run across five runs (including sum of data generation time and completion time), and instance sizes, and other similar criteria. These settings and measurements are the ***baseline*** measurements and configurations and must be used by the tester during elasticity + scalability phase and final report generation.

2.7 Elasticity + Scalability (Automated)

SPEC Cloud IaaS 2016 Benchmark elasticity + scalability 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 each workload randomly between five and 10 minutes.
2. Asynchronously, wait until each application instance is ready to accept work, and repeat the following sequence.
    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 elasticity + scalability phase and collect supporting evidence.

2.8 Benchmark Stopping Conditions

SPEC Cloud IaaS 2016 Benchmark elasticity + scalability driver uses ***cbtool*** to detect the following stopping conditions. The benchmark stops if any of the following conditions are met:

- 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 configured in the maximum_ais parameter of osgcloud_rules.yaml have been issued.
- Results have been received from the number of reported AIs as configured in the reported_ais parameter of osgcloud_rules.yaml.

The following conditions specify various QoS thresholds that are tracked per AI run. If any of these thresholds are violated for a particular AI run, a flag is set per application instance.

Measurement	*K-Means* Threshold	*YCSB* Threshold
Completion Time	<= 3.333 x BaselineKMeansCompletionTime	
Throughput	N/A	>= BaselineYCSBThroughput / 3.333
Insert Response Time	N/A	<= 20 x BaselineYCSB99%InsertRespTime
Read Response Time	N/A	<= 20 x BaselineYCSB99%ReadRespTime

2.9 Benchmark Compliant Run Conditions

The following conditions must be met before a submission is considered compliant and the review process initiated. Non-compliant benchmark results will not be accepted for review.

Test Condition	Benchmark Compliance Conditions
Measurements within respective QoS thresholds.	The benchmark tolerates up to 20% AI provisioning failures and up to 10% AI run failures before the benchmark runis stopped.
Minimum AI provisioned	Two (2) or more of each SPEC workload AI's provisioned successfully.
Minimum AIs with valid runs	One each of SPEC workload AIs with one or more valid runs. E.g., one YCSB AI with one valid run, and one K-Means AI with one valid run is considered a valid submission assuming other stopping conditions are met.
Baseline and elasticity + scalability phase are run one after the other.	If baseline or elasticity + scalability phase fails, the whole run is invalid.
Time for AI provisioning failure.	For the elasticity phase, the number of update_attempts is controlled by the harness such that update_frequency x update_attempts **does not** exceeds three times the maximum of the YCSB and KMeans AI provisioning time measured during the baseline phase. These parameters can be set by the user prior to the start of a compliant run such that they restrict this time to a value less than the calculated three time value if desired.
Instances other than those started by benchmark harness.	No instances can be started during a compliant run using the cloud credentials setup in the benchmark harness, except for those started by the benchmark harness.
Termination of instances during baseline phase	The instances are automatically terminated at the end of each run in the baseline phase. There are five runs per workload during baseline phase. The instances during baseline phase must be successfully terminated for a compliant run.

3.0 Reporting Results

3.1 Metrics

3.1.1 SPEC Cloud IaaS 2016 Benchmark Baseline Measurements

The optimal measurements for an application instance are determined during the Baseline Phase. Each workload application instance is provisioned and a single run is executed for that application instance in the cloud under test. This process is repeated five times for a workload application instance. ***cbtool*** collects each measurements from these five runs and averages the key metrics as reported by the workload.

Following baseline metrics are computed for YCSB:

Baseline YCSBhroughput	= Average (Throughput AI (i))
BaselineYCSBInsert99%ResponseTime	= Average (Insert 99% Response Time AI (i))
BaselineYCSBRead99%ResponseTime	= Average (Read 99% Response Time AI (i))
BaselineYCSBCompletionTime	= Average (Completion Time AI (i))
BaselineYCSBAIProvisioningTime	= Average (Provisioning Time AI (i))

Following baseline metrics are computed for K-Means:

BaselineKMeansCompletionTime	= Average (Completion Time AI (i))
BaseKMeansThroughput	= Average (Throughput AI (i))
BaseKMeansAIProvisioningTime	= Average (Provisioning Time AI (i))

The AI provisioning time for YCSB and K-Means are different because the underlying applications, (Cassandra and Hadoop) are different. These applications may take a different amount of time before they are ready to start loading data.

These ***baseline values*** are used in computing the elasticity and scalability metrics as well as the stopping conditions.

3.1.2 SPEC Cloud IaaS 2016 Benchmark Handling Provisioning and AI Run Errors during Elasticity + Scalability Phase

There are three types of errors that can occur during the life cycle of an application instance:

1. An application instance fails to provision.
2. An application instance reports any error during a run.
3. An application instance reports a time synchronization error.

If an AI fails to provision, it is not considered for elasticity and scalability metrics calculations. If a percentage of AIs fails to provision during elasticity + scalability phase (as defined in 2.8 Benchmark Stopping Conditions), the elasticity + scalability phase is stopped.

If an AI reports errors in one or more runs including a time synchronization issue, then results reported for workload metrics, if any, during that run are discarded and are replaced with values using the methodology described below:

For YCSB, there are four metrics used in the elasticity and scalability score computation, namely, throughput, 99% insert time, 99% read time, and deployment time. Only the first three metrics are applicable for an AI_run.

For a failed AI_run for YCSB, assign the following values to the metrics reported in that run:

throughput	0
insert 99% response time	2 x max insert 99% response time across AI run across all AIs
read 99% response time	2 x max read 99% response time across AI runs across all AIs

For K-Means, there are two metrics used in the elasticity score computation, namely, completion time and deployment time. Only completion time is applicable for an AI_run.

completion time	2 x max completion time across AI run across all AIs

A time synchronization error occurs for an AI run, if the absolute time difference between the benchmark harness machine and the load manager instance of an application instance is more than 10 seconds at the time of reporting of results to ***cbtool***.

Application instances with reported results that have time synchronization issues are not counted towards the 10% AI run errors as defined in 2.8 Benchmark Stopping Conditions. The time synchronization is checked at the end of the elasticity + scalability phase, and the above formula is applied.

3.1.3 Calculating Sub-Metrics for Elasticity and Scalability Metrics

This section describes how to calculate sub-metrics for elasticity and scalability after the completion of elasticity + scalability phase.

Starting_time Time at which the elasticity + scalability phase started.

Stopping_time	Time at which the elasticity + scalability phase stopped.
$AI_{provisioned/inprogress}$	List of AIs that have been provisioned or failed to provision or are in progress when the stopping condition is reached.
$AI_{one\_or\_more\_successful\_run}$	AIs with one or more successful runs and no failed runs until the stopping_time is reached. Its value is $<=AI_{provisioned/inprogress}$.
$AI_{one\_or\_more\_failed\_runs}$	AIs with one or more failed runs and may have zero or more successful runs until the stopping time is reached. Its value is $<=AI_{provisioned/inprogress}$
$AI_{failed\_to\_provision}$	AIs that failed to provision before stopping_time is reached.
AI_{no_run}	AIs that provisioned successfully but had not completed any run by the time the stopping conditions were reached.

$$AI_{provisioned/inprogress} = AI_{one\_or\_more\_successful\_run} + AI_{one\_or\_more\_failed\_runs} + AI_{failed\_to\_provision} + AI_{no\_run}$$

For the purposes of elasticity and scalability metrics calculations, only runs for $AI_{one\_or\_more\_successful\_run}$ and $AI_{one\_or\_more\_failed\_runs}$ are counted. These are referred to as List_AIs. The failed runs in $AI_{one\ or\ more\ failed\ runs}$ are assigned values as defined in Section 3.1.2 SPEC Cloud IaaS 2016 Benchmark Handling Provisioning and AI Run Errors during Elasticity + Scalability Phase.

$$List\_AIs = AI_{one\_or\_more\_successful\_run} + AI_{one\_or\_more\_failed\_runs}$$

List_AIs is then sequentially pruned as follows:

(1) Prune all AIs that provisioned after stopping_time
(2) Prune all AI runs that completed after stopping_time
(3) Prune all AIs that had zero completed runs after stopping_time
(4) Count the number of AIs for a workload. If number of AIs for a workload are less than 40 percent of total AIs in List_AIs, remove the last provisioned AI one-by-one, until ratio of the two workloads is at least 60-40%

The result from pruning is **Valid_AIs**. All calculations are based on Valid_AIs.

Avg_Inst_prov._time	For Valid_AIs, average of instance provisioning time across all instances in Valid_AIs
$AI_{(i)}$_prov._time	Time to provision an application instance (i)

$Total\_Runs\_AI_{(i)}$	Total runs in AI (i) including failed runs before the stopping conditions were reached
$AI_{(i)}\_Run_{(j)}\_Compl.\_Time$	Completion time for Run (j) of AI (i)
$Avg\_AI_{(i)}\_Compl.\_Time$	Average of completion time for $Total\_Runs\_AI_{(i)}$ of $AI_{(i)}$
$Last\_Compl.\_Time\_AI_{(i)}$	Completion time of last run of $AI_{(i)}$ before stopping condition
$AI_{(i)}\_Throughput_{(j)}$	Throughput for Run (j) of AI (i)
$Avg\_AI_{(i)}\_Throughput$	Average of throughput for $Total\_Runs\_AI_{(i)}$ of $AI_{(i)}$
$Last\_Throughput\_AI_{(i)}$	Throughput of last run of $AI_{(i)}$ before stopping condition
$AI_{(i)}\_Ins99Resp_{(j)}$	99% Insert response time for Run (j) of AI (i)
$Avg\_AI_{(i)}\_Ins99Resp$	Average of 99% Insert response time for $Total\_Runs\_AI_{(i)}$ of $AI_{(i)}$
$Last\_Ins99Resp\_AI_{(i)}$ 99%	Insert Response time of last run of $AI_{(i)}$ before stopping condition
$AI_{(i)}\_Rd99Resp_{(j)}$	99% Read response time for Run (j) of AI (i)
$Avg\_AI_{(i)}\_Rd99Resp$	Average of 99% Read response time for $Total\_Runs\_AI_{(i)}$ of $AI_{(i)}$
$Last\_Rd99Resp\_AI_{(i)}$ 99%	Read Response time of last run of $AI_{(i)}$ before stopping condition
Average Complt. Time	Average ($Avg\_AI_{(i)}\_Compl.\_Time$) for i in (1 to Valid_AIs)
Average Throughput	Average ($Avg\_AI_{(i)}\_Throughput$) for i in (1 to Valid_AIs)
Average Insert 99 Resp Time	Average ($Avg\_AI_{(i)}\_Ins99Resp$) for i in (1 to Valid_AIs)
Average Read 99 Resp Time	Average ($Avg\_AI_{(i)}\_Rd99Resp$) for i in (1 to Valid_AIs)
Average YCSB AI Prov. Time	Average ($Avg\_AI_{(i)}\_Prov.\_Time$) for i in (1 to Valid_AIs for YCSB AIs)
Average KMeans AI Prov. Time	Average ($Avg\_AI_{(i)}\_Prov.\_Time$) for i in (1 to Valid_AIs for KMeans AIs)

3.1.4 SPEC Cloud IaaS 2016 Benchmark Elasticity Metric

Elasticity measures whether the work performed by application instances scales linearly in a cloud. That is, for statistically similar work, the performance of N application instances in a cloud must be the same as the performance of application instances during baseline phase when no other load is introduced by the tester. Elasticity is expressed as a percentage (out of 100).

K-Means elasticity metric is defined as follows:

$$\textbf{Elasticity}_{\text{KMeans}} = \left(0.75 \times \text{Min}\left(1, \frac{Baseline\ KMeans\ Completion\ Time}{Average\ Complt.Time}\right) + 0.25 \times \text{Min}\left(1, \frac{Baseline\ KMeans\ AI\ Prov.\ Time}{Average\ KMeans\ AI\ Prov.Time}\right)\right) \times 100$$

YCSB elasticity metric is defined as follows:

$$\textbf{Elasticity}_{\text{YCSB}} = \left(0.375 \times \text{Min}\left(1, \frac{Average\ Throughput}{Baseline\ Throughput}\right) + \left(0.1875 \times \text{Min}\left(1, \frac{Baseline\ Insert\ 99\ RespTime}{Average\ Insert\ 99\ RespTime}\right) + 0.1875 \times \text{Min}\left(1, \frac{Baseline\ Read\ 99\ RespTime}{Average\ Read\ 99\ RespTime}\right) + 0.25 \times \text{Min}\left(1, \frac{Baseline\ YCSB\ AI\ Prov.\ Time}{Average\ Prov.Time}\right)\right)\right) \times 100$$

The aggregate elasticity metric is an average of elasticity metrics for two workloads. It is expressed as a percentage out of one hundred. The higher, the better.

$$\textbf{Elasticity} = (\textbf{Elasticity}_{\text{KMeans}} + \textbf{Elasticity}_{\text{YCSB}}) / 2$$

3.1.5 SPEC Cloud IaaS 2016 Benchmark Scalability Metric

Scalability measures the total amount of work performed by application instances running in a cloud. The aggregate work performed by one or more application instances should linearly scale in an ideal cloud.

Scalability is reported as a unit-less number @ the number of valid application instances. The number is an aggregate of workloads metrics across all application instances running in a cloud normalized by workload metrics in a reference cloud platform. The reference platform metrics are an average of workload metrics taken across multiple cloud platforms during the benchmark development.

The Scalability of YCSB is calculated as follows:

$$\textbf{Scalability}_{\textbf{YCSB}} = \textbf{Sum (}\text{Avg\_Throughput\_AI}_{(i)} \textbf{ / RefPlatThr}_{\textbf{YCSB}}\textbf{) @ YCSB\_Application\_Instances}$$

where (i) is from 1 to Valid_AIs for YCSB AIs

The Scalability of K-Means is calculated as follows:

$Scalability_{KMEANS}$ = Sum (RefPlatComTim$_{KMEANS}$ / Avg_Compl._Time_$AI_{(i)}$ **) @ KMeans Application Instances**

where (i) is from 1 to Valid_AIs for K-MEANS AIs

The benchmark Scalability metric is the sum of all workload scalability scores.

Scalability = **Sum** (***Scalability***$_{KMeans}$, ***Scalability***$_{YCSB}$) @ Sum(YCSB_Application_Instances,KMeans_Application_Instances)

Scalability metric is a unit-less number. A higher value is better.

3.1.6 SPEC Cloud IaaS 2016 Benchmark Mean Instance Provisioning Time Metric

Means Instance Provisioning Time metric represents an average of provisioning time of all instances in Valid_AIs.

3.1.7 SPEC Cloud IaaS 2016 Benchmark AI Provisioning Success Metric

This metric indicates the percentage of AIs that were successfully provisioned. This number is calculated as follows:

$AI_{provisioned/inprogress}$ = $AI_{one\_or\_more\_successful\_run}$ + $AI_{one\_or\_more\_failed\_runs}$ + $AI_{failed\_to\_provision}$ + AI_{no_run}

AI Provisioning Success = 100 x (1 – ($AI_{failed\_to\_provision}$ / $AI_{provisioned/inprogress}$))

3.1.8 SPEC Cloud IaaS 2016 Benchmark AI Run Success Metric

This metric indicates the percentage of AIs that had all successful runs. It is calculated as:

Valid_AIs = $AI_{one\_or\_more\_successful\_run}$ + $AI_{one\_or\_more\_failed\_runs}$ (given that percentage of AIs for a workload is not less than 40 percent)

AI Run Success = 100 x (1 - $AI_{one\_or\_more\_failed\_runs}$ / $AI_{one\_or\_more\_successful\_run}$)

3.1.9 SPEC Cloud IaaS 2016 Benchmark Elasticity Start time Metric

This metric indicates the time at which each the elasticity + scalability phase of the benchmark was started by the harness.

3.1.10 SPEC Cloud IaaS 2016 Benchmark Elasticity End time Metric

This metric indicates the time at which each the elasticity + scalability phase of the benchmark was stopped by the harness.

3.1.11 SPEC Cloud IaaS 2016 Benchmark Total Instances Metric

This metric indicates the total instances in Valid_AIs.

3.1.12 Metric Reporting

An example of the metrics is shown below.

SPEC Cloud IaaS 2016 Metrics	Scalability	Elasticity	Mean Instance Provisioning Time
	10.349 @ 5 Application Instances	86.88%	229s
	AI Provisioning Success	AI Run Success	Elasticity + Scalability Phase Start time
	97%	95%	2015-01-15_16:15:23_UTC
	Total Instances	Test Region	Elasticity + Scalability Phase End Time
	33 Instances	Oregon data center	2015-01-15_19:47:53_UTC

3.2 Testbed Reporting Requirements

The tester is responsible for producing a Full Disclosure Report (FDR) that documents the testbed configuration. The goal of providing this documentation is that an independent 3rd party can reproduce the SUT and replicate the results without further information.

3.2.1 General Cloud Requirements

a. Any deviations from the standard default configuration for the SUT configuration components must be documented in the FDR so that an independent party would be able to reproduce the configuration and the result without further information. These deviations must meet general availability and support requirements (Section 3.3 General Availability).
b. An independent party should be able to reproduce the performance achieved to at least 90% or 110% for time measurements, of that originally reported for a tested cloud based on the contents of the FDR. The 90% rule applies to the following metrics: Scalability, Elasticity, and Total Instances. The 110% rule applies to Mean Instance Provisioning Time. It is recommended that for Blackbox tests, the benchmark be run on the same weekday and time as per the original submission, and at least two attempts for reproducibility are made. The results on the SPEC website will be marked non-compliant if the results from two runs attempted for reproducibility do not meet the 90% and 110% guidelines described above.
c. Details about instances such as instance identifiers, IP addresses, network connect connectivity, security groups, users/projects they belong to, information about their image, instance types, block storage devices used by instances, quotas must be disclosed during different phases of a compliant run.
d. Information about cloud API endpoints configured in the benchmark harness or otherwise must also be disclosed.

3.2.2 Additional White Box Cloud Requirements

a. White-box clouds are under the complete control of the tester. All components, including hardware, software, configurations, and/or data centers must be completely documented in the FDR so that an independent third party can reproduce the results.
b. The results must include a system diagram in PNG format that shows how different components are connected (network and storage) to deliver the IaaS cloud.
c. For the cloud management or automation software used in white box clouds, the FDR must disclose the details on that software and its configuration. For example, the software details will include any software (or package) version, and configuration will include a dump of configurations files that contain information such as number of workers for a database that store persistent state of an IaaS cloud.
d. The FDR must disclose the details on the physical machine / hypervisor and its configuration on which the workloads run.

e. The type of network cards and network isolation mechanism used must be disclosed for white-box clouds, for example, the use of VLAN/VxLAN/GRE/VxLAN with NIC offloading.
f. For white box clouds, the physical storage connectivity among SUT components such as iSCSI or Fiber Channel must be disclosed.

3.2.3 Additional Black Box Cloud Requirements

a. Since Black-box clouds may not be under the complete control of the tester and may not expose visibility into the underlying hardware configuration, the tester must submit all details about the black-box cloud (e.g., location of physical data center, the type of instances, storage, or network used as will be ordered by a cloud consumer) and the workloads running on it as exposed by the cloud provider to the cloud consumer.

3.2.4 Instance Requirements

a. The FDR must disclose the nature of an "instance" as either a virtual machine (VM) or a physical machine or a container.
b. The type of storage the instances are booted or started from (a block-storage device or an ephemeral disk must be disclosed as part of the FDR.

3.2.5 Network Requirements

a. The FDR must describe the virtual networks used (e.g., VLAN, GRE, VxLAN, IP subnet for instances, virtual routers), if any, for communication among instances, and communication between instances and ***cbtool***.
b. The MTU size of the virtual networks must also be disclosed as part of FDR.
c. The use of cloud security groups that restrict communication among instances must be disclosed.
d. The network connectivity among SUT components must be shown where applicable.

3.2.6 Storage Requirements

a. The type of instance storage such as block storage or ephemeral storage must be disclosed. For example, a block storage device can be attached to an instance after it has been created. That block storage device can be configured for Hadoop or Cassandra to store data.

3.2.7 Instance Images Requirements

a. The test sponsor will configure their own instances in accordance with the run rules defined in this document. The test sponsor will submit all relevant commands for creating and uploading these images in the cloud. The user guide gives examples of commands for

creating workload images and uploading these images in the cloud. These can be used as reference.

b. The FDR must indicate whether images are cached, i.e., if there were any previous runs in the cloud or some special image distribution mechanism is used that caches or streams images for faster instance creation. This information may not be exposed in a black box cloud.

3.2.8 Benchmark Harness Requirements

a. The physical, virtual, and/or cloud configuration details configured in the benchmark harness for a compliant run must be disclosed. For example, this will include cloud adapter, API, and accounts or users configured in ***cbtool*** for running the benchmark.

3.3 General Availability

The hardware and software components of the SUT must meet certain general availability requirements stated in the overall governing SPEC General Availability policy found at http://www.spec.org/osg/policy.html#AppendixC. The following sections highlight the considerations important to this benchmark.

3.3.1 Blackbox General Availability Requirements

All blackbox clouds are required to be generally available on or before the date of publication.

Blackbox cloud is considered generally available if ordinary customers can order and use the service(s) within a reasonable time frame, typically measured in minutes or hours (not days). The availability of support and documentation for the products must coincide with the release of the products. The general availability of a blackbox cloud service, even if it uses open source software, is set by the cloud provider as described above.

Restrictions on the Blackbox cloud's geographic availability outside of ordinary and necessary business requirements must be disclosed For example, when a BlackBox cloud provider only offers the service in their "*home*" country, that service restriction must be noted.

3.3.2 Whitebox General Availability Requirements

All system, hardware and software features are required to be generally available on or before date of publication, or within 3 months of the date of publication (except where precluded by these rules). The dates of general customer availability must be listed for the major components: hardware, software (cloud management stack, hypervisor, operating systems, and applications), as month and year. When multiple components have different availability dates, use the latest availability date.

Products are considered generally available if they are orderable by ordinary customers and ship within a reasonable time frame. This time frame is a function of the product size and classification, and common practice. The availability of support and documentation for the products must coincide with the release of the products.

Hardware products that are still supported by their original or primary vendor may be used if their original general availability date was within the last five years. The five-year limit is waived for hardware used in clients.

For ease of benchmarking, storage and networking hardware external to the server such as disks, storage enclosures, storage controllers and network switches, which were generally available within the last five years but are no longer available from the original vendor, may be used. If such end-of-life (and possibly unsupported) hardware is used, then the test sponsor represents that the performance measured is no better than 110% of the performance on hardware available as of the date of publication. The product(s) and their end-of-life date(s) must be noted in the disclosure. If subsequent attempts, to replicate a submitted result, determine that the performance using available hardware to be worse by more than 10%, the result shall be marked non-compliant (NC). See 3.2.1 General Cloud Requirements for details.

Software products that are still supported by their original or primary vendor may be used if their original general availability date was within the last three years.

In the disclosure, the benchmark tester must identify any component that is no longer orderable by ordinary customers.

If pre-release hardware or software is tested, then the test sponsor represents that the performance measured is generally representative of the performance to be expected on the same configuration of the release system. If subsequent attempts, to replicate a submitted result, using the same available hardware or software determine that the performance to be lower than 90% of that reported, the result shall be marked non-compliant (NC).

3.3.3 Rules on the Use of Open Source Applications

SPEC Cloud IaaS 2016 Benchmark does permit Open Source Applications outside of a commercial distribution or support contract with some limitations. The following are the rules that govern the admissibility of the Open Source Application in the context of a benchmark run or implementation. Open Source Applications do not include shareware and freeware, where the source is not part of the distribution.

1 Open Source Application rules do not apply to Open Source used for White-box cloud infrastructure software used to create the cloud, which would still require a commercial distribution and support. This includes software such as the cloud management stack, any virtualization or operating system software required.

2 Open Source Application rules do not apply to Open Source operating systems used in instance images, which would still require a commercial distribution and support.

3 Only a "stable" release can be used in the benchmark environment; non-"stable" releases (alpha, beta, or release candidates) cannot be used.

Reason: An open source project is not contractually bound and volunteer resources make predictable future release dates unlikely (i.e. may be more likely to miss SPEC's 3 month General Availability window). A "stable" release is one that is clearly denoted as a stable release or a release that is available and recommended for general use. It must be a release that is not on the development fork, not designated as an alpha, beta, test, preliminary, pre-released, prototype, release-candidate, or any other terms that indicate that it may not be suitable for general use.

4 The initial "stable" release of the application must be a minimum of 12 months old.

Reason: This helps ensure that the software has real application to the intended user base and is not a benchmark special that's put out with a benchmark result and only available for the 1st three months to meet SPEC's forward availability window.

5 At least two additional stable releases (major, minor, or bug fix) must have been completed, announced and shipped beyond the initial stable release.

Reason: This helps establish a track record for the project and shows that it is actively maintained.

6 An established online support forum must be in place and clearly active, "usable", and "useful". It's required that there be at least one posting within the last three months. Postings from the benchmarkers or their representatives, or members of the OSG Cloud Subcommittee, are not included in the count.

Reason: Another aspect that establishes that support is available for the software. However, benchmarkers must not cause the forum to appear active when it otherwise would not be. A "useful" support forum is defined as one that provides useful responses to users' questions, such that if a previously unreported problem is reported with sufficient detail, it is responded to by a project developer or community member with sufficient information that the user ends up with a solution, a workaround, or has been notified that the issue will be addressed in a future release, or that it is outside the scope of the project. The archive of the problem-reporting tool must have examples of this level of conversation. A "usable" support forum is defined as one where the problem reporting tool was available without restriction, had a simple user-interface, and users can access old reports.

7 The project must have at least two identified developers contributing and maintaining the application.

Reason: To help ensure that this is a real application with real developers and not a fly-by-night benchmark special.

8 The application must use a standard open source license such as one of those listed at http://www.opensource.org/licenses/.

9 The "stable" release used in the actual test run must have been a latest "stable" release within the prior six months at the time the result is submitted for review. The exact beginning of this time window has to be determined starting from the date of the submission then going back six months and keeping the day number the same. If benchmarkers are notified on the product web page, developer forums, or other public support mechanisms that they must migrate off the version due to a critical issue, then this subclause does not apply (that is, the use of this product version is not allowed). Note: Residual cases are treated as described as in http://www/spec.org/osg/policy.html#s2.3.4 substituting the six month window for three month availability window. Examples:

Submission date	Beginning of time window
Aug 20, 2019	Feb 20, 2019
Jul 20, 2019	Jan 20, 2019
Jun 20, 2019	Dec 20 2018

Reason: Benchmarkers should keep up to date with the recent releases; however they are not required to move to a release that would be fewer than six months old at the time of their submission.

Please note, an Open Source Application project may support several parallel development branches and so there may be multiple latest stable releases that meet these rules. For example, a project may have releases such as 10.0, 9.5.1, 8.3.12, and 8.2.29 that are all current supported and stable releases.

10 The "stable" release used in the actual test run must be no older than 18 months. If there has not been a "stable" release within 18 months, then the open source project may no longer be active and as such may no longer meet these requirements. An exception may be made for “mature” projects (see below).

In rare cases, open source projects may reach “maturity” where the software requires little or no maintenance and there may no longer be active development. If it can be demonstrated that the software is still in general use and recommended either by commercial organizations or active open source projects or user forums and the source code for the software is fewer than 20,000 lines, then a request can be made to the subcommittee to grant this software “mature” status. In general, it is expected that the final stable release for the "mature" project continues to work "as is" for the majority of users but that over time some users may need to make portability changes. This status may be reviewed semi-annually. Any projects granted "mature" status by the subcommittee will be posted the benchmark’s technical FAQ.

3.4 SUT Bill of Materials

The SPEC Cloud IaaS 2016 Benchmark submitter must provide a detailed Bill of Materials (**BOM**) describing the actual components used and how they were connected during the benchmark run. The intent of the BOM is to provide sufficient details such that a reviewer can confirm that the tested configuration satisfies the run rule requirements. The SUT description or BOM must reflect the level of detail a customer would see on an itemized bill. It must list individual items in the SUT that are not part of a standard package. For each item, the BOM must include the item's supplier, description, the item's ID (the code used by the vendor when ordering the item), and the quantity of that item in the SUT.

Additional details (such as configuration details) required to reproduce the test will need to be reported elsewhere in the full disclosure report, but not necessarily in the BOM.

Requirements for reporting information about the system under test are different for Blackbox and Whitebox clouds.

3.4.1 BOM Reporting Rules for Blackbox

The BOM for a **Blackbox service** must reflect the level of detail a customer sees on an itemized bill if that customer were to run SPEC Cloud IaaS 2016 Benchmark. The submission must provide a Bill of Materials (BOM) describing the services consumed when running SPEC Cloud IaaS 2016 Benchmark. This BOM must reflect the level of detail a customer would see on an itemized bill if that customer were to run SPEC Cloud IaaS 2016 Benchmark.

If an itemized bill provides all of the necessary information listed below, then the itemized bill is sufficient to meet the BOM reporting requirements.

3.4.1.1 Cloud Service Vendor

The following information must be included in the reported BOM:

Cloud Service	Required Information
Cloud Vendor Name(s)	The public name(s) of the company(s) of all services used in the SUT.
Product/Service Name(s)	is the itemized list of the official names of all services used during the SPEC Cloud IaaS 2016 Benchmark run. This can also be the descriptions with ordering identification description or codes used by the **Cloud Vendor** on their website or as listed in a billing invoice. If the service is offered in multiple classes that impact the functionality of that service, then sufficient details must be listed to uniquely identify the service. For example, Windows usage or Linux usage, and their respective versions may need to be specified for a computing service type.
Quantity Used	of each service used during the SPEC Cloud IaaS 2016 Benchmark run, specified in the same billing units the Vendor charged for those services (e.g. hours, GB, etc.) and at the same granularity billed (e.g. 2.5 hours, 3.1 GB).
Storage	must be specified as either Local or Remote types (block storage, SAN storage), include all non-default setup configuration settings (e.g. block size matching database blocks), and specific assignments (e.g. database files, server logs, OS files, or working areas.
Geographical or Logical Zones	used, if the **Cloud Vendor** provides such service options and was used during the submitted SPEC Cloud IaaS 2016 Benchmark run.

Start / End Date Time	when the baseline and elasticity + scalability phase were executed.
Website Link(s)	to the Cloud Vendor(s) service price lists, or a description of how to obtain the prices for all services listed in the **BOM**.

3.4.1.2 Software

The following software components must be reported:

Software Components	Required Information
All Instance Image(s)	The assigned role, description, and provenance of all OS images used during the SPEC Cloud IaaS 2016 Benchmark run. It must include: • Workload assignment (ycsb, seed, hadoopmaster, hadoopslave) – (part of ***cbtool*** configuration file) • The Operating System name, distribution name, and version information. • Source of the image, and instructions for creating the images. • Any OS configuration and tuning parameters • A list of packages and applications included in the operating system image.
Notes Section	Any additional information required to reproduce the benchmark results (e.g. number of daemons, disk configuration, non-default kernel parameters, etc.) and logging mode must be given in this section.

3.4.2 BOM Reporting Rules for Whitebox

3.4.2.1 Hardware

The following SUT hardware components must be reported:

Hardware Component	Required Information
Vendor's Name(s)	The public name(s) of the company(s) of all hardware and software used in the SUT.
System Model Name(s)	The itemized list of the official names all hardware and software used in the SUT, during the SPEC Cloud IaaS 2016 Benchmark run.
Quantity Used	The assigned role and counts used.
System Hardware Details	System firmware version(s) (e.g. BIOS) Processor model, clock rate, number of processors (#cores, #chips, #cores/chip, on-chip threading enabled/disabled) Size and organization of primary, secondary, and other cache, per processor (if a level of cache is shared among processor cores in a system that must be stated in the "notes" section) Main memory size and memory configuration if this is an end-user option which may affect performance, e.g. interleaving and access time Other hardware, e.g. write caches, or other accelerators, or GPUs
Network Interfaces	The type of NIC used (100BaseT, GigaBit, Fibre, 10GFibre) and the quantity and network assignments of each interface (management, instance, or public for communication with benchmark harness machine).
Storage	Number, type, model, and capacity of storage subsystem Storage-level redundancy used (such as RAID level), and how it is connected to the cloud machines.
File System(s)	The types used for various purposes (e.g. Cassandra logs directory, Cassandra database directory, Hadoop filesystem directory, Hadoop temporary files directory, logs, work area) and any non-standard settings used in creating and/or mounting.

Hardware Component	Required Information
Availability Date(s)	When the hardware or software are generally available for customer orders and delivery.

3.4.2.2 Software

The following SUT software components must be reported:

Software Component	Required information
Cloud Type	Information about the type of Cloud system: • Cloud management stack (e.g., OpenStack), • Software and API version • All configuration files used by the cloud management stack. • Any cloud API rate limitation being used. • Indication whether TLS is used for cloud API. • The deployment type of cloud management stack (e.g., High availability for all or some components, non-HA). Supply a deployment diagram for the cloud system that depicts the overall architecture, including network, HA, DR in PNG format.
All Instance Image(s)	The assigned role, description, and provenance of all OS images used during the SPEC Cloud IaaS 2016 Benchmark run. It must include: • Workload assignment (ycsb, seed, hadoopmaster, hadoopslave) – (part of ***cbtool*** configuration file) • The Operating System name, distribution name, and version information. • Source of the image, and instructions for creating the image. • Any OS configuration and tuning parameters • A list of packages and applications included in the image.

Software Component	Required information
Virtualization Details	If cloud instances run within virtualization environments, then the following details must be documented: • Virtualization software (hypervisor) and all hypervisor-level tunings • Virtual machine details (number of virtual processors, memory, network adapters, disks, etc.) • Any caching mechanisms for fast provisioning of Workload instances - specific details (operating system, application name and version, tunings) for each workload (NOTE: these must be identical across all instances)
Notes Section	Any clarifying information as required to reproduce benchmark results (e.g. number of daemons, BIOS parameters, disk configuration, non-default kernel parameters, etc.), and logging mode must be stated in this section.

3.4.2.3 Networking

A brief description of the network configuration used to achieve the benchmark results is required. The minimum information to be supplied is:

Network Component	Required information
Network Devices	Information about all of the physical or virtual networks connected to all SUT and testbed components: • Number and type of networks used. • Functional purpose of network (e.g. cloud management, instance, public, high-availability and disaster recovery network (can be part of management) etc) • Base speed of each network. • Number, type, model, and relationship of external network components to support SUT (e.g., any external routers, hubs, switches, etc.) The networks must be included in the diagram for the cloud system that depicts the overall architecture.

Network Component	Required information
Notes Section	Relationship of machines, machine type, and networks (including routers, etc. if applicable) -- in short: which machines, e.g., cloud controller, compute nodes, are connected with which network." Specify if any SDN or virtual networks (e.g., VLAN, VxLAN, GRE) are configured and used.

3.4.3 Cloud Configuration

The cloud configuration (as-is typically seen by a cloud consumer) must be disclosed during different phases of the benchmark. This includes:

- The configured cloud users to deploy the workloads.
- List of all instances (names and ids), the networks they are connected to (ids), the images they are provisioned from, the instance size, creation time, SSH keys, the ids of the user that created them, and (if applicable) the tenants the users belong to.
- List of all virtual networks, their subnet information, and ids
- List of all physical or virtual routers and the networks they are connected to.
- List of all images provisioned in the cloud.
- List of all instance types in the cloud.
- List of all configured users, and how they are grouped (projects/tenants) in the cloud, and their relevant quotas for instances, block storage devices.
- List of all block storage devices if any in the cloud.
- Secure transport protocol used for Cloud APIs (e.g., TLS).

Any other peculiar configuration that will make it possible for an independent third party to reproduce the results.

3.4.4 Harness

The following information about the system(s) used to run the SPEC Cloud IaaS 2016 Benchmark harness:

- System model number(s), processor type and clock rate, number of processors
- Main memory size
- Network Controller(s)
- Operating System and/or Hypervisor and Version
- Type and quantity of the storage system used by the harness during the benchmark run.
- Specify in notes section, how the harness machine was connected to the cloud under test.

4.0 Submission Requirements for SPEC Cloud IaaS 2016 Benchmark

Compliant runs need to be submitted to SPEC for review and must be accepted prior to public disclosure. All public statements using SPEC Cloud IaaS 2016 Benchmark must follow the SPEC Fair Use Rule (http://www.spec.org/fairuse.html). In order to publicly disclose SPEC Cloud IaaS 2016 Benchmark results, the submitter must adhere to these reporting rules in addition to having followed the run rules described in this document. The goal of the reporting rules is to ensure the SUT is sufficiently documented such that someone could reproduce the test and its results.

To submit results to SPEC for review and acceptance, the following information must be included in the submission package:

- Use the SPEC Cloud IaaS 2016 Benchmark harness features for collecting supporting evidence during baseline and elasticity + scalability phases. The support evidence collection from instances during baseline phase must be enabled before the start of a compliant run. The supporting evidence collection from instances must be collected at the end of elasticity + scalability phase. Example scripts for supporting evidence collection are provided with the kit, but will need tailoring for a specific operating system distribution or a cloud under test.

Licensees of the benchmark wishing to submit results for acceptance may be required to pay a fee.

4.1 Performance Results Collections

Performance results are automatically collected and processed by the SPEC Cloud IaaS 2016 Benchmark kit as part of a compliant run.

4.2 SUT Configuration Collection

The submitter must run script(s) that collects available configuration details of the SUT and all the instances used for the benchmark, including:

For whitebox clouds, the script must gather the following information from SUT:

- SUT configuration / configuration files and tuning
- SUT storage configuration and tuning
- SUT network configuration and tuning
- Hardware configuration (physical and if applicable virtual and or cloud instance)
- Host operating system, filesystem, and network configuration and tuning (e.g. non-default registry or /etc/sysctl.conf)

- Cloud management stack configuration and tuning
- Hypervisor configuration and tuning (if applicable)
- Instance configuration (# of virtual/physical CPUs, memory, ephemeral or block device size, network adapters)
- Virtual networking configuration (if applicable)
- Network and storage connectivity diagrams for SUT
- API rate limitation
- Any quota
- Any non-automated aspect of the data collection (such as accessing a SAN manager to get details on the storage configuration) must be noted.

For both black box clouds, the following information must be collected:

- Instance configuration (# of virtual/physical CPUs, memory, disk, network adapters, tuning)
- Virtual networking configuration (if applicable)
- API rate limitation
- The name of cloud account used to test
- Any quota

The primary reason for this step is to ensure that the tester may not miss any subtle differences.

The configuration gathering scripts must be included as part of the submission.

Additional files or the output of commands run on the SUT to help document details relevant to questions that may arise during the review.

Log files during elasticity + scalability phase are not collected by the benchmark harness. The benchmark harness does not terminate the instances at the end of the elasticity + scalability phase which allows a tester to collect log files post elasticity + scalability phase.

During a review of the result, the submitter may be required to provide, upon request, additional details of the harness configuration that may not be captured in the above script to help document details relevant to questions that may arise during the review.

4.3 Instance Configuration Collection

The submitter must run a script which provides the details of each instance, its operating system and application tunings. This information includes:

- Guest operating system, file system type(s), and network configuration and tuning (e.g. non-default registry or /etc/sysctl.conf)
- Application-specific configuration and tuning files

- Command outputs to document details related to any specific requirements for a workload such as software versions used.

During a review of the result, the submitter may be required to provide additional details of the instance, operating system and application tunings, and log files that may not be captured in the above script. These may include, but are not limited to:

- Application-specific log files (the disk paths for Cassandra and Hadoop log files are specified in the User Guide.
- Additional files or the output of commands run on the instances to help document details relevant to questions that may arise during the review.

The primary reason for this step is to ensure that the vendor has disclosed all non-default tunings.

4.4 Harness Configuration Collection

The submitter must submit the machine configuration of the benchmark harness machine, the cloud configuration in the harness, as well as all the parameters of CBTOOL. The submitter must also document how the benchmark harness machine is connected to the cloud (directly, via VPN, or a jump box)

4.5 Code and Script Collection

If the submitter develops or modifies any scripts for supporting evidence collection, these must be submitted as part of the results. These include:

- ***cbtool*** scripts (cbtool/scripts)
- scripts for gathering instance, quota, user, instance type, network, image type information from a cloud.
- scripts for gathering configuration and supporting evidence from instances and/or white-box cloud machines

4.6 Configuration and Results Collection Archive Format

The submitter must submit the configuration and results collection archive containing the data (files and command output) described in Sections 4.0, 4.1, 4.2, 4.3, 4.4, and 4.5 using the high level directory structure described below as the foundation. The items in bold and underline and all items under them indicate that they are automatically collected as part of a compliant run.

- PERF (automatically collected)
  - `baseline_SPECRUNID.yaml`
    `(summarizes the output of baseline phase. Used in elasticity + scalability phase for stopping conditions)`
  - `elasticity_SPECRUNID.yaml`

(summarizes the output of elasticity + scalability phase)
    - fdr_ai_SPECRUNID.yaml
      (summarizes the output of report generation)
    - **SPECRUNIDELASTICITYTIMESTAMP**
      (elasticity + scalability phase directory containing performance data in CSV files. There must only be a single instance of directory within PERF directory)
    - **SPECRUNIDKMEANSBASELINE0TIMESTAMP**
    - **SPECRUNIDKMEANSBASELINE1TIMESTAMP**
    - **SPECRUNIDKMEANSBASELINE2TIMESTAMP**
    - **SPECRUNIDKMEANSBASELINE3TIMESTAMP**
    - **SPECRUNIDKMEANSBASELINE4TIMESTAMP**
      (there are five directories for kmeans baseline corresponding to five runs. Each directory contains performance data in CSV files)
    - **SPECRUNIDYCSBBASELINE0TIMESTAMP**
    - **SPECRUNIDYCSBBASELINE1TIMESTAMP**
    - **SPECRUNIDYCSBBASELINE2TIMESTAMP**
    - **SPECRUNIDYCSBBASELINE3TIMESTAMP**
    - **SPECRUNIDYCSBBASELINE4TIMESTAMP**
      (there are five directories for kmeans baseline corresponding to five runs. Each directory contains performance data in CSV files)
    - osgcloud_elasticity_SPECRUNID-TIMESTAMP.log
      (log of the elasticity + scalability phase)
    - osgcloud_fdr_SPECRUNID-TIMESTAMP.log
      (log of the FDR generation. If FDR generation is run multiple times, please submit only the last log file)
    - osgcloud_kmeans_baseline_SPECRUNID-TIMESTAMP.log
      (the log of the baseline phase. There will be 5 log files for KMeans baseline)
    - osgcloud_rules.yaml
      (the rules file that contains parameters and configurations for the benchmark)
    - osgcloud_supporting_evidence_SPECRUNID-TIMESTAMP.log
      (the log file generated by instance supporting evidence collection scripts)
    - osgcloud_ycsb_baseline_SPECRUNID-TIMESTAMP.log
      (the log of the baseline phase. There will be 5 log files for YCSB baseline)
    - run_SPECRUNID.log
      (log file that is generated by all_run.sh script that triggers various phases of the benchmark)
    - sub_file_SPECRUNID.txt
      (the submission file as generated by the FDR HTML output generator)

- CLOUD_CONFIG (semi-automated)
    - notes.txt (general notes about SUT)

- osgcloud_environment.yaml (A YAML file describing various aspects of the cloud that go into the final report).
- IMAGE_INSTRUCTIONS
  - One file per YCSB/Cassandra and KMeans: A text or PDF file describing instructions for creating instance images, source of the image, and screen shots where applicable.
  - One file per YCSB/Cassandra and KMeans: List of installed packages inside the image (e.g., debian, rpm, pip)
- INSTANCE_ARCH_DIAG
  - Network and storage architecture for instances for black box and white-box clouds in PNG format.
- BLACKBOX (remove when submitting results for whitebox clouds)
  - notes.txt (general notes about blackbox)
- WHITEBOX (remove when submitting results for blackbox clouds)
  - notes.txt (general notes about whitebox)
  - COMPUTE_NODES
    - COMPUTE_NODE (1 to N)
      - SW
        - CONFIG_FILES_DIR
          - Mgmt stack config files
          - version.txt
          - description.txt
          - hypervisor.txt
      - MACHINE_SUP_EVID_FILES
        - `date.txt  dpkg/rm.txt  hostname.txt  lspci.txt  netstat.txt  proc  var  df.txt  etc  ifconfig.txt  mount.txt  ntp.conf  route.txt`
      - network_configuration.txt (a brief description of network configuration of this machine)
      - storage_configuration.txt (a brief description of storage configuration of this machine)
      -
  - CONTROLLER_NODES
    - CONTROLLER_NODE (1 to N)
      - purpose_version_description.txt (describes the purpose, version, and a brief description of this controller component).
      - SW
        - CONFIG_FILES
      - MACHINE_SUP_EVID_FILES
        - `date.txt  dpkg/rm.txt  hostname.txt  lspci.txt  netstat.txt  proc  var  df.txt  etc  ifconfig.txt`

```
mount.txt  ntp.conf
route.txt
```

- network_configuration.txt (a brief description of network configuration of this machine)
- storage_configuration.txt (a brief description of storage configuration of this machine)

- CLOUD_MGMT_SOFTWARE
  - description.txt (a brief overview of Cloud management software)
- ARCH_DIAGRAM
  - Network and storage architecture diagram for white-box cloud in PNG format. For HTML report generation, combine the instance and white-box images into a single file. However, include them separately in the respective directories.

- HARNESS
  - MACHINE_INFO
    - machine.txt (physical or VM)
    - sut_connectivity.txt (how harness is connected to SUT)
    - MACHINE_SUP_EVID_FILES

      ```
      date.txt  dpkg.txt  hostname.txt  lspci.txt
      netstat.txt  proc       var df.txt    etc
      ifconfig.txt  mount.txt  ntp.conf     route.txt
      ```
  - SOFTWARE
  - CONFIGFILES
  - **harness_config.yaml**

- INSTANCE_EVIDENCE_DIR (automatically collected)
  - BASELINE_PRE
  - KMEANS_BASELINE_POST
  - YCSB_BASELINE_POST
  - ELASTICITY_POST

    (for above 4 directories, results from running the scripts below in the directories. Implementation to be provided by the cloud provider. Reference OpenStack scripts are included in the kit. For black-box clouds, gethypervisors.sh can be empty).
    - `getapiendpoint.sh, getimages.sh, getnetworks.sh, getusers.sh, getblockstorage.list, getinstances.sh, getquotas.sh, gethypervisors.sh getinstancestype.sh, gettenant.sh`
  - BASELINE
    - CBTOOL_BASELINE_EXPERIMENT_ID (1 to 10)
      - AI_ID (e.g., ai 1 to 10)
        - INSTANCE_NAME

                - INSTANCE
                    - `date.txt  dpkg.txt  hostname.txt  lspci.txt  netstat.txt  proc  var  df.txt  etc  ifconfig.txt  mount.txt  ntp.conf  route.txt`
                - SW
                    - Workload software configuration
    - ELASTICITY
        - CBTOOL_ELASTICITY_EXPERIMENT_ID (1)
            - AI_ID (ais with one or more successful or failed runs)
                - INSTANCE_NAME
                - INSTANCE
                    - `date.txt  dpkg.txt  hostname.txt  lspci.txt  netstat.txt  proc  var  df.txt  etc  ifconfig.txt  mount.txt  ntp.conf  route.txt`
                - SW
                    - Workload software configuration

- CODE (manual)
    - HARNESS_SCRIPTS
        - ***cbtool*** scripts from cbtool/scripts
        - CHANGES file describing any changes in the scripts. ***Any changes must be reviewed and deemed "performance neutral" by the OSG Cloud subcommittee.***
    - CLOUD_CONFIG_SCRIPTS
        - cloud_scripts used for gathering cloud configuration (e.g., getinstances.sh, getimages.sh, getquota.sh)
    - INSTANCE_SUPPORT_EVIDENCE_SCRIPTS
        - scripts for gathering supporting evidence from instances after each iteration of workload baseline phase, and elasticity + scalability phase.
        - CHANGES file describing any changes in the scripts
    - WHITEBOX_SUPPORT_EVIDENCE_SCRIPTS
        - scripts for gathering supporting evidence from a whitebox cloud
        - README file describing the script

4.7 Submitting Results

- Create a zip file containing the result submission .txt file (under perf directory) and the cloud architecture diagram in PNG format.
- Email that zip file to subcloudiaas2016@spec.org.
- To upload supporting documents, wait until you receive your confirmation that your result has been received so you have the result number to attach the supporting docs to. Create a package (.tgz, .zip, whichever format you have standardized on) with the documents and give it the name of your associated result, e.g. cloudiaas2016-20160107-00016.tgz.
- Upload the file using the FTP information found here: https://pro.spec.org/private/osg/cloud/ftpuser.txt

4.8 Adding a New Cloud Adapter for Cloud Bench

The tester is expected to ensure that the ***cbtool*** can connect to the cloud under test. If a cloud is not supported or if the cloud API in the existing ***cbtool*** adapter has changed, the tester must follow the guidelines below on the writing or updating a new adapter, and submitting results for that new adapter. API documentation supporting the implementation of the ***cbtool*** adapter for the cloud to be tested must be publicly available.

- The tester must complete a SPEC Permission to Use (P2U) form for the Adapter source code and submit it to the SPEC office for review.
- Once the P2U is signed off, the Adapter source code can be sent to the subcommittee for review with a link to the API document.
- The subcommittee has at least four weeks to review the code, and if the review completes without issues, it is considered accepted.
- If accepted, the Adapter may be used in an actual submission.
- The source code for the Adapter must be included in the FDR for the submission.
- Accepted CBTOOL Adapters may be included in future benchmark kits.
- This process is independent of any adapter submissions to the Cloudbench project on github.

Please refer to the User Guide for the details on writing a ***cbtool*** cloud adapter.
(https://www.spec.org/cloud_iaas2016/docs/userguide/cloudbench_setup.html#adding-a-new-cloud-driver)

5.0 The SPEC Cloud IaaS 2016 Benchmark Kit

SPEC Cloud IaaS 2016 Benchmark comprises the test harness and workloads for the benchmark. This release includes the benchmark harness (***cbtool***, baseline and elasticity drivers, and relevant configuration files) along with the YCSB and Hibench/KMeans workloads, and scripts to produce benchmark reports and submission files.

This software implements various checks for conformance with these run and reporting rules; therefore the SPEC software must be used as provided. Source code modifications are not allowed, unless prior approval from SPEC is given. Any such substitution must be reviewed and deemed "performance neutral" by the OSG Cloud Subcommittee and the OSSC.

The kit also includes example scripts to facilitate testing and data collection.

Copyright © 1998-2016 Standard Performance Evaluation Corporation (SPEC). All rights reserved.

Revision Date: March 16, 2016