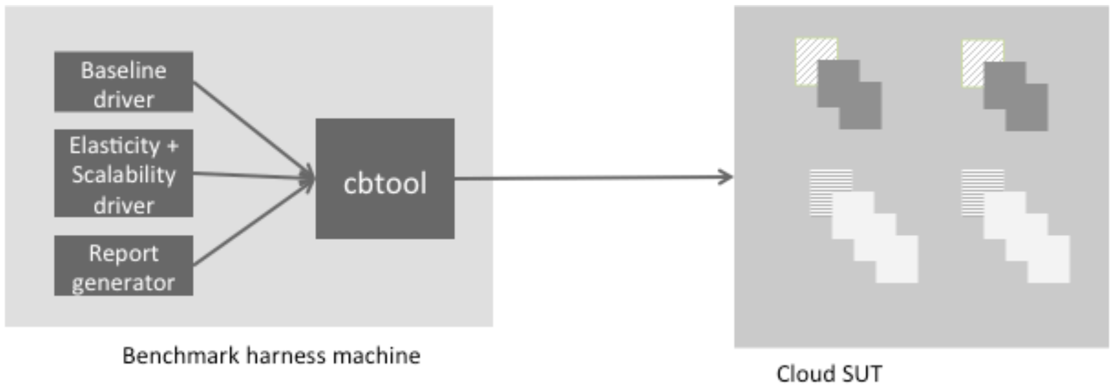
SPEC Cloud™ IaaS 2016 Benchmark Design Overview



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

Cloud SUT
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.

1. Overview of SPEC Cloud™ IaaS 2016 Benchmark

The SPEC Cloud™ IaaS 2016 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 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 names and terms used in this benchmark were established in the original SPEC OSG Cloud White Paper, published in 2012. It uses most of the definitions established in the US Government NIST white paper on cloud computing [Reference: NISTPub145].

Cloud	Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. [Reference: NISTPub145]
Cloud Provider	An organization that provides cloud services to customers who pay for only the computing time and services actually used. [Reference: NISTPub145]
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. [Reference: NISTPub145]
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 *Cloud-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). [Reference: CloudWhitePaper]
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.

	[Reference: CloudWhitePaper]
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. This will typically be the case if the entity benchmarking the cloud is different from a cloud provider (e.g., for public clouds). [Reference: CloudWhitePaper]
Hybrid Cloud	The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds). [Reference: NISTPub145]
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. [Reference: NISTPub145]
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. [Reference: NISTPub145]

The SPEC Cloud IaaS 2016 Benchmark uses these terms through out its documents.

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, AMI (Amazon machine image), or a Softlayer Flex image.
Physical machine	A typical physical machine has at least 4 GB of memory, 40 GB of local or remote (network attached, block storage) disk space, 1 Gb/s of network connectivity, and processing power equivalent to Intel Xeon processors. The physical machine can have its own physical packaging or be installed as a blade in a blade chassis.
Application Instance	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	The measured time needed to bring up a new instance, or add more resources (like CPU or storage) to an existing instance. [Reference: CloudWhitePaper] Within this document, this will be divided into two measurements: ***Instance***: The time from request to create a new instance until that instance responds to a *netcat* probe on port 22. ***Application instance***: The time from request to create a new instance until the associated cluster reports readiness to accept client 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.
SUT	One or more cloud services under test. This includes all hardware, network, base software and management systems used for the cloud service. [Reference: CloudWhitePaper]

Response Time	The time between when the work item request is issued until the corresponding completion condition. This definition is identical to the 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., 95%) 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 [Reference: Cbtool]
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 +	In the Elasticity + Scalability phase, new application instances

scalability phase	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.

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 2016 Benchmark utilizes a subset of the workloads that represent real-world use cases found on public, private, or hybrid an IaaS clouds. For this release, the benchmark comprises two workloads, the Yahoo! Cloud Serving Benchmark (YCSB) [Reference: YCSBWhitePaper] and the K-Means implementation from HiBench [References: KMeansClustering and HiBenchIntro].

SPEC Cloud IaaS 2016 Benchmark workloads are managed by a benchmark harness, Cloud Bench a.k.a ***cbtool*** [Reference: cbtool] that 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 submission.

The architecture of the benchmark comprises two execution phases, Baseline and Elasticity + Scalability. In the baseline phase, peak performance for each workload is determined in separate test runs. Data from the baseline phase is used to establish parameters for the Elasticity + Scalability phase. In the Elasticity + Scalability phase, both workloads are run concurrently to determine elasticity and 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:

- Limited requirements are placed on the internal architecture of the test-bed.
- A hypervisor or virtualization layer is not required.
- No requirements are placed on the instance configuration. A cloud provider is free to choose CPU (virtual CPU or core pinning), memory, disk (ephemeral disk or block storage), and network configuration for VMs.

- 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.
- Supports easy addition of future workloads.
- Supports optional multi-tenancy.

The cloud SUT 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 2016 Benchmark. These metrics include Durability, Isolation, 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.

This version also does not include benchmark phases that explicitly use geographically isolated cloud configurations. However, the benchmark does not preclude a submission 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 SPEC Cloud IaaS 2016 Benchmark.

2. Benchmark Details

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 2016 Benchmark is divided into two groups of hosts. These machines may be physically co-located in the same facility (data center) or company campus. The group of machines on the right side of Figure 1 comprises the system under test (SUT). The machine on the left side of Figure 1 comprises the benchmark harness.

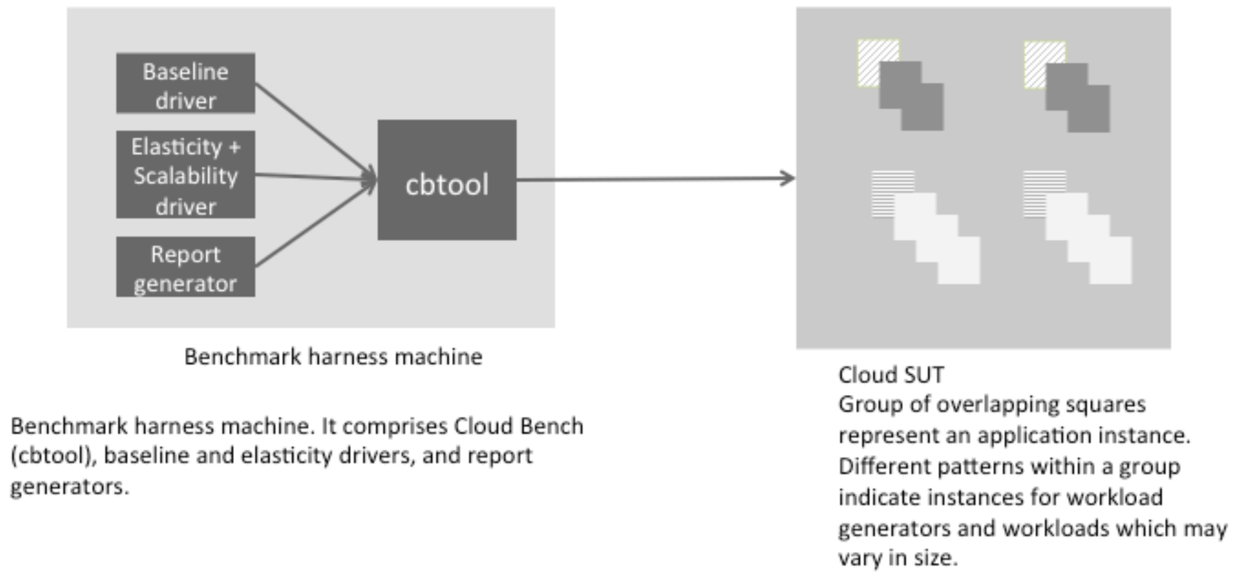

Baseline driver
Elasticity + Scalability driver
Report generator
cbtool
Benchmark harness machine
Benchmark harness machine. It comprises Cloud Bench (cbtool), baseline and elasticity drivers, and report generators.
Cloud SUT
Group of overlapping squares represent an application instance. Different patterns within a group indicate instances for workload generators and workloads which may vary in size.

Figure 1 Logical architecture of SPEC Cloud IaaS 2016 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 both public and 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 2016 Benchmark uses Cloud Bench, referred to as ***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 ***elasticity + scalability*** drivers for executing the two phases of the benchmark. Finally, the ***report generator*** is used to generate the report for the ***baseline*** and ***elasticity + scalability*** phases of the benchmark.

Cbtool provides connectors 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 appropriate scripts 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., one to five minutes). As such it is possible for the generated data to fit within an instance memory; however, within each workload, certain 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.

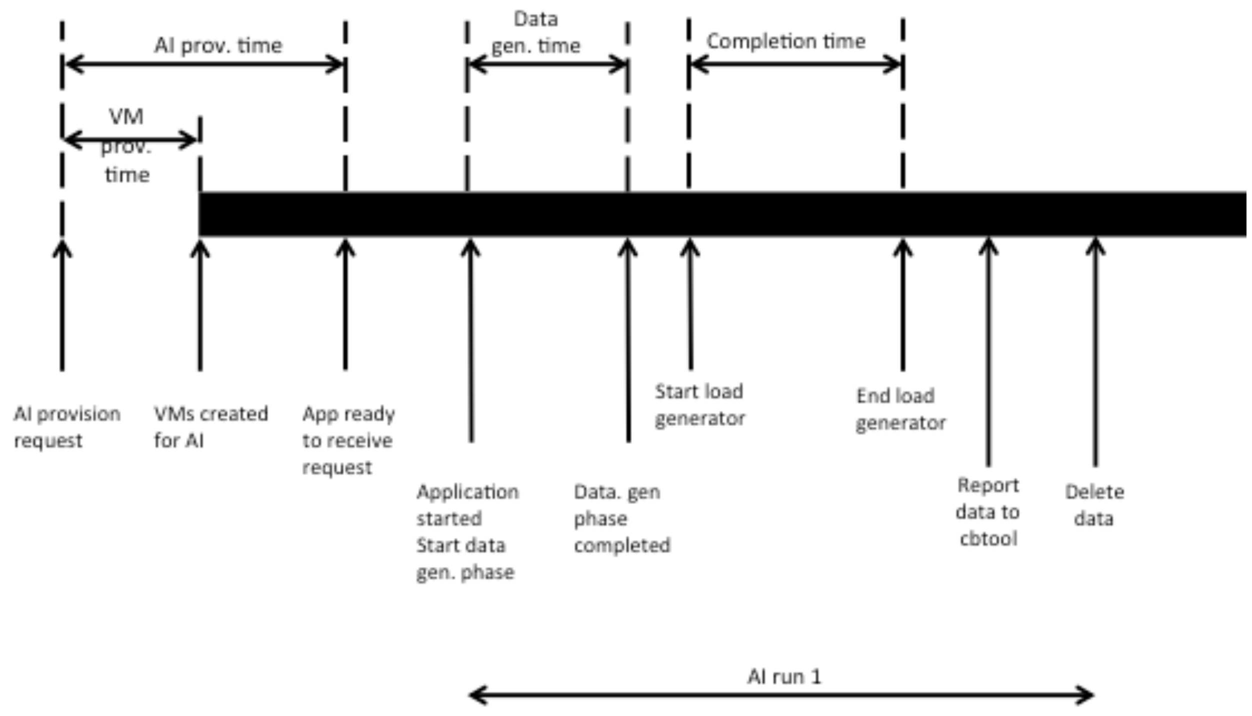


Figure 2 Life cycle of an application instance

The ***baseline driver*** generates load for the baseline phase of the benchmark. The ***baseline driver*** creates an application instance, generates the data set, and runs the load phase. As soon as the baseline driver receives the metrics by the workload driver, it terminates the application instance. It then repeats the same procedure for a total of five times. Thus, each iteration within the baseline phase comprises an AI provisioning and a single AI run.

The ***elasticity + scalability driver*** generates the load for elasticity + scalability phase of the benchmark. The ***elasticity + scalability*** driver instructs ***cbtool*** to create application instances for each workload with the interval between creations using 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 the application instances have been created, they go through one or more AI runs (data generation, 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 prevent any caching that may result due to the use of same data set across multiple application instances. During elasticity + scalability phase, application instances are not deleted. Once the ***elasticity + scalability driver*** determines that one or more QoS thresholds have been breached, the driver instructs ***cbtool*** to stop creating application instances. This marks the end of elasticity + scalability phase of the benchmark.

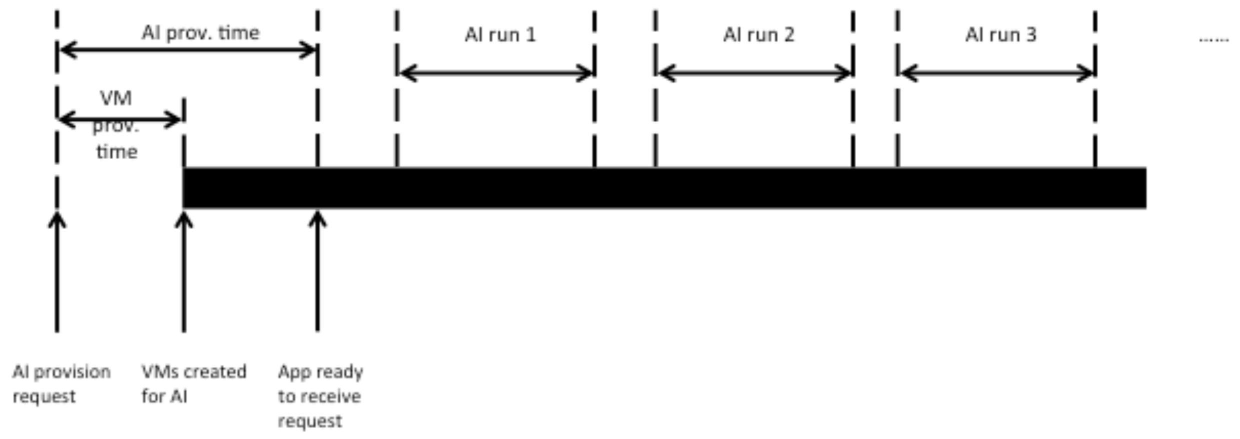


Figure 3 Life cycle of an application instance - multiple runs

2.2 SUT (IaaS Cloud)

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

- The host system(s) (including hardware and software) required to support the workloads.
- 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 that is required to build, deploys, and run the specific benchmark workload.

The SUT will support multiple instances of either physical or virtual existence.

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 initial 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 two of the most typical types 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. SPEC Cloud IaaS 2016 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 2016 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 2016 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.

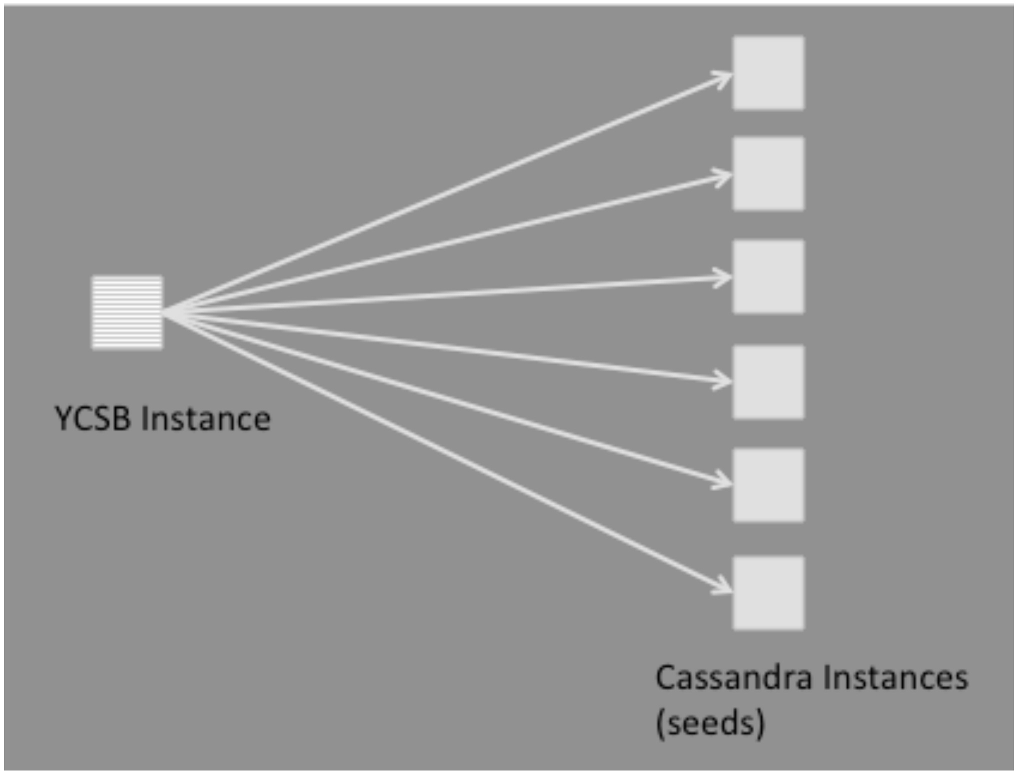


Figure 4 YCSB / Cassandra application instance in SPEC Cloud IaaS 2016 Benchmark

Cassandra supports two types of nodes in its cluster configuration, namely, seeds and data nodes. Seeds are used during startup to discover the cluster [Reference: CassandraSeedsOne]. One design option was to use three seeds and three data nodes. However, during experimentation, it was discovered that data nodes take a non-deterministic time to join the cluster. Moreover, multiple data nodes joining at the same time is potentially problematic. Per Cassandra documentation at the time of benchmark development, a gap of two minutes is recommended 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. YCSB driver instance can be configured with a variable number of threads to generate load on the Cassandra cluster. It is up to the cloud provider to determine the number of threads for YCSB driver instance (the default is 8 threads). 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 limits within YCSB driver instance are reached, 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 high throughput using a large number of threads may be susceptible to larger performance degradation as load on the cloud increases.

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 elasticity + scalability phase, the data is generated for each ***AI run***. The experimentation revealed that on average, generating a database with 1 million records took more than ten minutes. Since an ***AI run*** includes data set generation as well as the load phases, 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 into disk I/O during load generation. Moreover, the data set generation before an AI run also results into 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 2016 Benchmark

Description	YCSB parameter	YCSB parameter value
Total records inserted in DB	recordcount	1,000,000

Total operations during a YCSB run	operationcount	1,000,000
Workload uses	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

2.3.1.2 YCSB metrics

Following metrics from YCSB are used for elasticity + scalability score calculations:

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

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 2016 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 2016 Benchmark. Figure 5 shows the logical architecture of K-Means application instance in SPEC Cloud IaaS 2016 Benchmark.

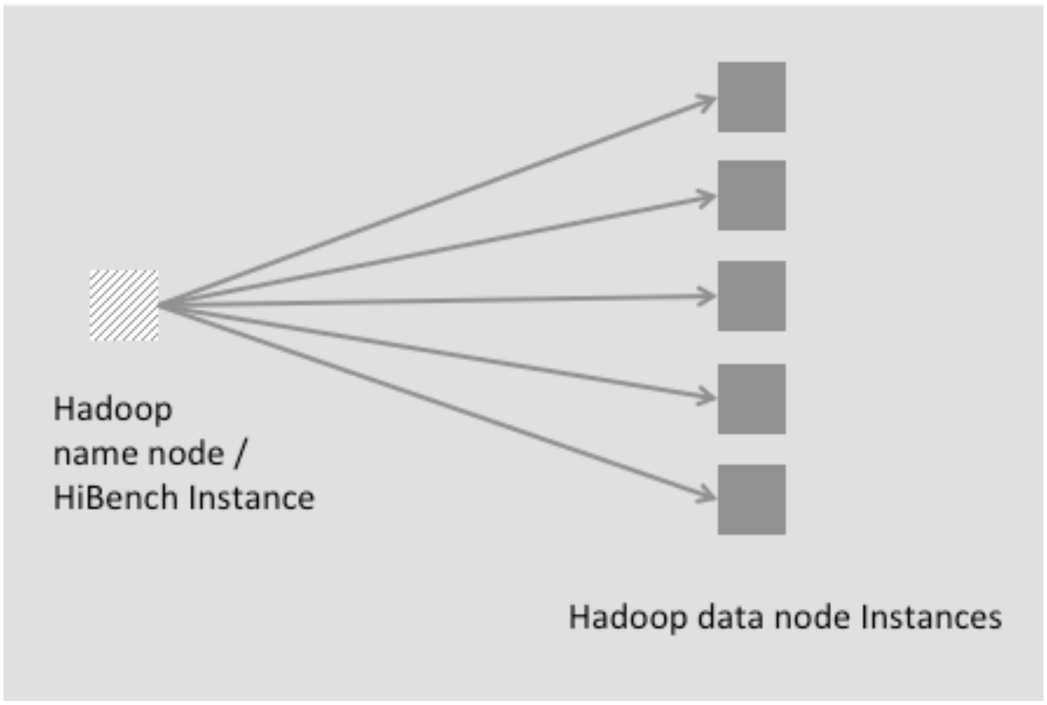


Figure 5 K-Means application instance

SPEC Cloud IaaS 2016 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 2016 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. This was considered an acceptable tradeoff. 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.

Overall, the choice of these parameters dictated that on a 2 VCPU, 4GB virtual machine, it approximately takes 10 minutes to generate the data set and to run the K-Means algorithm.

Similar to YCSB, the choice of parameters for data generation for K-Means was a tradeoff between the time it takes to generate the data and run the K-Means algorithm. 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 900MB. With Hadoop's three-way replication, the size on disk is approximately 2.8 GB.

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)

2.4 Reference Platform

The benchmark uses results from a reference platform for Scalability metric computation purposes. Member companies participating in the benchmark development collected these results.

Specifically, each company participating in the benchmark development submitted baseline results for YCSB (throughput) and K-Means (completion time) workloads for two types of instance sizes. These results were obtained without applying any performance tuning. The results are then averaged to compute a reference platform throughput and completion time for YCSB and K-Means, respectively. The average removes the bias towards one particular cloud platform.

The details for reference platform computation are in Appendix 1: SPEC Cloud IaaS 2016 Benchmark Reference Platform Metrics.

3. Running the Benchmark

cbtool exposes an API which is used by baseline and elasticity + scalability drivers to drive the workloads as per the requirements of baseline and elasticity + scalability phase (see Figure 1)

Delete

3.1 Setup (Manual)

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 system under test (SUT), or on an external network to the SUT. The tester ensures that ***cbtool*** can connect to its cloud.

The tester then installs and configures two ***instances*** for YCSB/Cassandra and HiBench/Hadoop by following the instructions in the user. The tester must take a snapshot of these instances 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 is ready to start the benchmark phases. A tester may run the baseline and elasticity + scalability phases couple of times as a trial run before starting a compliant run. Please refer to the user guide for details on how to run the baseline phase.

The tester is expected to ensure that the ***cbtool*** can connect to the cloud under test. If a cloud is not supported, the tester must write a ***cbtool*** adapter for the cloud under test.

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 tenants in the final score.

3.2 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 supporting evidence. If there is no error in the five runs of an application instance as reported by ***cbtool*** and the results meet the bounds defined for Quality-of-Service thresholds, the baseline result is considered valid.

cbtool records the settings and results associated with the most optimal performance, i.e. average of throughput, completion time and other similar criteria, as well as instance sizes. These settings and measurements are the ***baseline*** measurements and configurations and must be used by the tester in later steps.

3.3 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.

SPEC Cloud IaaS 2016 Benchmark elasticity + scalability 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.

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

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 2016 Benchmark reports eight metrics, namely,

- Elasticity
- Scalability
- Mean Instance Provisioning Time
- AI Provisioning Success
- AI Run Success
- Total Instances
- Elasticity Start Time
- Elasticity End Time

These metrics are defined below. The details for computing these metrics are included in Run and Reporting Rules document.

4.1 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).

The guideline for interpreting elasticity results is follows:

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

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 result is, the better.

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

4.1.1 Discussion

In SPEC OSG Cloud report [Reference: CloudWhitePaper], Elasticity has been defined as a function of provisioning time, agility, scale up/down and elastic speedup. Provisioning time is defined as the duration between requesting a resource and when that resource is ready to serve the request. This metric is captured by the ***AI_prov._time*** for each AI.

The agility metric was defined as the ability of the system provisioned to be as close to the needs of the workload as possible while maintaining a specified QoS. Similar to this definition, Herbst et al. [Reference: ElasticityICAC] define elasticity as the degree to which a system is able to adapt to workload changes by provisioning and deprovisioning resources in an autonomic manner, such that at each point in time the available resources match the current demand as closely as possible. It is difficult to precisely compute the agility metric or the elasticity metric defined by Herbst et al, which will measure elasticity of the cloud and not applications, and allow meaningful comparisons across clouds.

Scale up/down is defined as a measurement of the system's ability to maintain a consistent unit completion time when solving increasingly larger problems, by adding a proportional amount of storage and compute resources. Elastic speedup is defined as whether adding SUT resources as the workload is running results in a corresponding decrease in response time. As such both scale up/down and elastic speed up measure the application performance. The focus of the SPEC Cloud IaaS 2016 Benchmark is to measure the elasticity of the cloud and not of specific applications.

YCSB and K-Means application instances are configured to perform a fixed amount of work in SPEC Cloud IaaS 2016 Benchmark. As load on a cloud increases by adding more application instances, the workload specific metrics may get affected. That is, the throughput of YCSB across AI runs of application instances may decrease or insert/read response time of YCSB may increase, or the completion time of K-Means may increase, or the time to provision an application instance may increase relative to the metrics computed during the baseline phase.

The definition of elasticity for SPEC Cloud IaaS 2016 Benchmark measures this decrease in throughput or increase in response time or completion or provisioning time relative to baseline metrics for each application instance. The degradation shows that the amount of compute, storage, and network resources available to each application instance of a workload performing the same amount of work, decrease, thereby, degrading the application metrics. The degradation is computed by averaging metrics over all AI runs for an application instance, and then further averaging over all AIs. In a perfectly elastic cloud, the degradation will be zero.

Recall that each a new data set is generated within each AI run. The data set was generated for each AI run so that any caching affects due to same data set across AIs are minimized. This procedure implies that the amount of work to be performed by an AI within every AI run is statistically similar but not exactly similar.

To counter the affects of statistically similar data sets, the baseline results are average over five 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.

4.2 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. It answers the question: ***How much more gets done if N instead of one (1) application instances***? Ideally, each additional application instance will contribute 100% of its individual capability, but this may not be the case as the cloud gets loaded with multiple application instances or other workloads (for public clouds).

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

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

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

4.2.1 Discussion

The Scalability formula captures the work done by a cloud. If a cloud does more work than the other cloud for the same amount of resources while meeting QoS thresholds, than it has a higher scalability score than the other cloud.

SPEC Cloud IaaS 2016 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 scalability score. While the benchmark full disclosure report includes individual metrics of each workload, it was considered necessary to devise workload independent metrics that are reported in the final score.

The solution in coming up with a single metric for scalability was to normalize the raw throughput, and completion time metrics computed for a cloud by 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 2016 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 scalability score.

The Scalability formula may somewhat be 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 2016 Benchmark puts a lower bound on the percentage of AIs for a workload used for scalability score 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 scalability metric may somewhat be skewed towards one workload. This is considered acceptable for this release of the benchmark. In future, the reference platform metrics may be updated to reflect more modern hardware configurations.

In the Scalability metric, a reference provisioning time for an AI is not used. The reason is that provisioning time can vary widely for instances. For a cloud comprising of VMs, they are typically order of 10s of seconds, for a cloud comprising of bare metal machines, order of minutes, and for a cloud comprising of containers, order of few seconds. It is difficult to derive a reference provisioning time as a function of three different technologies. Therefore, SPEC Cloud IaaS 2016 Benchmark does not use it for reference platform computations. Instead, it relies on the workload performance metrics for computing the reference platform metrics.

4.3 SPEC Cloud IaaS 2016 Benchmark Mean Instance Provisioning Time Metric

Means Instance Provisioning Time represents an average of provisioning time of all instances in valid application instances.

4.4 SPEC Cloud IaaS 2016 Benchmark AI Provisioning Success Metric

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

4.5 SPEC Cloud IaaS 2016 Benchmark AI Run Success Metric

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

4.6 SPEC Cloud IaaS 2016 Benchmark Elasticity Start time Metric

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

4.7 SPEC Cloud IaaS 2016 Benchmark Elasticity End time Metric

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

4.8 SPEC Cloud IaaS 2016 Benchmark Total Instances Metric

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

4.9 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 Phase Start time
	97%	95%	2015-01-15_16:15:23_UTC
	Total Instances	Test Region	Elasticity Phase End Time
	33 Instances	Oregon data center	2015-01-15_19:47:53_UTC

4.9.1 Discussion on the Metrics Reported in the Final Score

Average Instance provision time is subsumed under AI provisioning time, which is part of the elasticity score. Since raw instance provisioning time has been a key metric reported in the literature and is easy to compare, it is reported separately in the final score.

4.9.2 Setting the Limit for Maximum AIs

A cloud provider may set the limit on the maximum AIs that will be run in its environment. If this limit is set to a small value, the scalability score may be small while there may be a small degradation for the elasticity metric. Similarly, average instance provisioning time may not be impacted.

In general, a cloud with higher scalability and elasticity score, lower provisioning times and errors is better than a cloud with low scalability and elasticity scores, and higher provisioning times or errors.

4.10 Benchmark Calculation Example

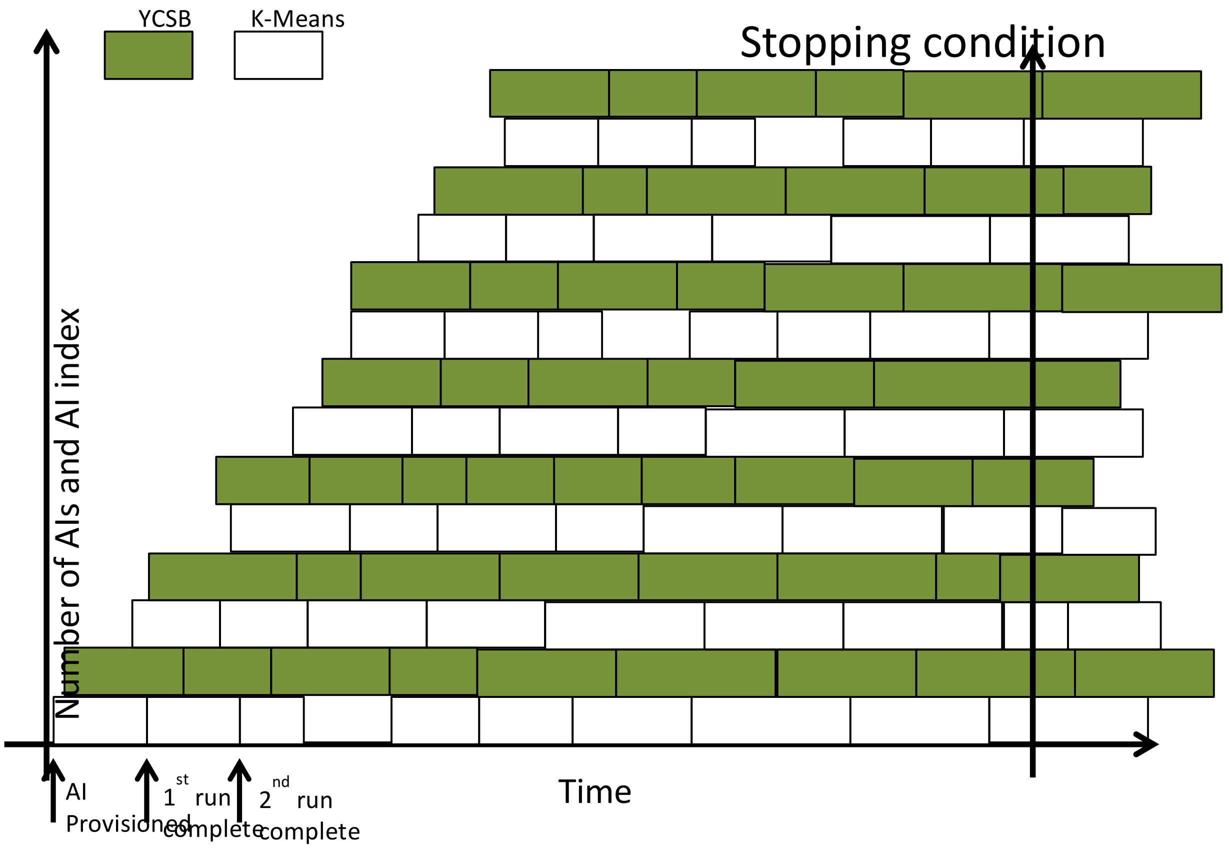
The following is an example of how the SPEC Cloud IaaS 2016 Benchmark metrics will be calculated.

RefPlatThr$_{\textbf{YCSB}}$: 8171.23 ops/s

RefPlatComTim$_{\textbf{KMEANS}}$: 189.1s

YCSB	K-Means
Baseline Throughput: 6200 ops/s Baseline Insert 99% time: 9ms Baseline Read 99% time: 5ms Baseline YCSB Deploy time: 220s	Baseline Completion Time: 400 s Baseline K-Means Deploy time: 180s

Valid_AIs: 14



YCSB AIs

Total YCSB AIs: 7

AI #	Total runs	Failed runs	AI Prov. Time (s)	Throughput of last run before stop.	Throughput average over runs	Insert 99% resp. time (ms)	Read 99% resp time (ms)
2	7	0	240	4900	6000	10	5
4	6	0	220	5300	6100	10	4
6	8	0	245	5200	5900	14	6
8	6	0	260	5100	5500	15	5
10	5	0	280	5000	5700	20	7
12	4	0	295	4800	5200	25	8
14	5	0	315	4900	5100	25	9

Average YCSB AI Prov. Time	(240 + 220 + 245 + 260 + 280 + 295 + 315) / 7	265
Average Throughput	(6000 + 6100 + 5900 + 5500 + 5700 + 5200 + 5100) / 7	5643
Average Insert 99 Response Time	(10 + 10 + 14 + 15 + 20 + 25 + 25) / 7	17
Average Read 99 Response Time	(5 + 4 + 6 + 5 + 7 + 8 + 9) / 7	6.3

$\boldsymbol{Elasticity}_{YCSB} =$

$$(0.375 \text{ x Min}\left(1, \frac{Average\ Throughput}{Base\ Throughput}\right) + 0.1875\ x\ Min\left(1, \frac{Base\ Insert\ 99\ RespTime}{Average\ Insert\ 99\ RespTime}\right) + 0.1875\ x\ Min\left(1, \frac{Base\ Read\ 99\ RespTime}{Average\ Read\ 99\ RespTime}\right) + 0.25 \text{ x Min}\left(1, \frac{Base\ YCSB\ AI\ Prov.\ Time}{Average\ Prov.Time}\right)) \text{ x } 100$$

$$= (0.375 \text{ x } (5643 / 6200) + 0.1875 \text{ x } (9 / 17) + 0.1875 \text{ x } (5 / 6.3) + 0.25 \text{ x } (220 / 265)) * 100 = 79.7\%$$

Scalability$_{YCSB}$ = Sum (Last_Throughput_AI$_{(i)}$ **/ RefPlatThr$_{YCSB}$) where (i) is from 1 to Valid_AIs for YCSB**

= (4900 + 5300 + 5200 + 5100 + 5000 + 4800 + 4900) / 8171.23 = 4.307 @ 7 Application Instances

KMeans AIs

Total K-Means AIs: 7

AI #	Total runs	Failed runs	AI. Prov. Time (s)	Average of Completion Time over all runs
1	9	0	190	390
3	7	0	170	420
5	6	0	160	350
7	6	0	200	450
9	7	0	210	440
11	5	0	220	480
13	6	0	200	490

Average KMeans AI Prov. Time	(190 + 170 + 160 + 200 + 210 + 220 + 200) / 7	193
Average Completion Time	(390 + 420 + 350 + 450 + 440 + 480 + 490) / 7	431

Elasticity$_{KMeans}$ =

$$(0.75 \text{ x Min}(1, \frac{Base\ CompltTime}{Average\ CompltTime}) + 0.25\ x \text{ Min}(1, \frac{Base\ KMeans\ AI\ Prov.\ Time}{Average\ KMeans\ AI\ Prov.Time})) \text{ x } 100$$

= (0.75 x (400 / 431) + 0.25 *(180 / 190)) * 100 = 93.3%

Scalability$_{KMEANS}$ = Sum (RefPlatComTim$_{KMEANS}$ / Avg_Compl._Time_AI$_{(i)}$ **)**

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

= 189.1 x (1 / 390 + 1 / 420 + 1 / 350 + 1 / 450 + 1 / 440 + 1 / 480 + 1 / 490) = 3.105 @ 7 Application Instances

Elasticity score = (79.7% + 93.3%) / 2 = 86.5%

Scalability score = 4.307 + 3.105 = 7.4 @ 14 Application Instances

Final score:

Scalability: 7.4 for 14 AIs

Elasticity: 86.5%

Means Instance Provisioning Time: 229s (calculations not shown)

with:

AI Provisioning Success=100%
AI Run Success=100%

5 Limitations of the benchmark

SPEC Cloud IaaS 2016 Benchmark has the following limitations.

1. SPEC Cloud IaaS 2016 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 cloud provider 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 elasticity + scalability 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 2016 Benchmark.
6. Client-server workloads (REST HTTP) (e.g., DayTrader or SPEC Web benchmark) are an important class of workloads that run on cloud. For these types of workloads, the clients typically run outside the cloud. To mimic these workloads on a cloud, the workload generators may have to be outside cloud. Such workloads will be incorporated in the next release of the benchmark.
7. SPEC Cloud IaaS 2016 Benchmark supports one or more tenants. However, it does not enforce the use of multiple tenants. The reason is that black-box clouds are typically multi-tenant, in the sense that the tenants may share the compute, storage, or network infrastructure. However, since white-box clouds are under the full-control of tester and may not have any background load, a cloud provider may desire to run workloads under more than on tenants. Moreover, a cloud may solely focus on scalability and not multi-tenancy.

6 Full Disclosure Reports

SPEC Cloud IaaS 2016 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 sufficient 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 2016 Benchmark Reference Platform Metrics

The benchmark uses results from a reference platform for Scalability metric computation purposes. The results were collected from cloud platforms of participating member companies.

Specifically, the baseline results were collected for YCSB (throughput) and K-Means (completion time) workloads for ”medium” instance sizes. These results were obtained without applying any performance tuning. The results are then averaged to compute a reference platform throughput and completion time for YCSB and K-Means, respectively. The average reduces the bias towards one particular cloud platform.

Reference metrics for “medium” instance size, which is 2 VCPUs, 4-8GB RAM, 30-40GB disk

A1.1 YCSB Reference Metrics

YCSB throughput (ops/s) reference metrics

YCSB throughput	Participant A, Cloud 1 (whitebox)	Participant B, Cloud 1 (blackbox)	Participant C, Cloud 1 (whitebox)
Iteration			
1	8695.50	7298.26	8643.34
2	8769.01	7156.66	8399.97
3	8839.31	7841.91	8468.48
4	8461.31	7237.51	8604.67
5	8910.04	6689.86	8552.64
Average	8735.03	7244.84	8533.82

Reference platform throughput (medium) = 8171.23 ops/s

YCSB 99% Insert response time (ms) reference metrics

YCSB Insert Resp. time (99%)	Participant A, Cloud 1 (whitebox)	Participant B, Cloud 1 (blackbox)	Participant C, Cloud 1 (whitebox)
Iteration			
1	1.48	1.95	1.54
2	1.37	2.40	1.48
3	1.59	1.90	1.63
4	1.68	1.79	1.42
5	1.54	2.08	1.36
Average	1.53	2.02	1.49

Reference platform insert response time (medium) = 1.68 ms

YCSB 99% Read response time (ms)

YCSB Read Resp. time (99%)	Participant A, Cloud 1 (whitebox)	Participant B, Cloud 1 (blackbox)	Participant C, Cloud 1 (whitebox)
Iteration			
1	1.87	2.52	1.93
2	1.84	2.64	2.03
3	1.87	2.23	2.02
4	1.93	2.4	1.84
5	1.95	3.13	1.90
Average	1.89	2.58	1.94

Reference platform read response time (medium) = 2.14 ms

A1.2 K-Means Reference Metrics

K-Means completion time (s)

K-Means compl. Time (s)	Participant A, Cloud 1 (whitebox)	Participant B, Cloud 1 (blackbox)	Participant C, Cloud 1 (whitebox)
Iteration			
1	230	233	132
2	117	329	100
3	128	178	165
4	131	291	185
5	102	285	230
Average	141.6	263.3	162.4

Reference platform K-Means completion time (medium) = 189.1s

Appendix 2: References

Keyword	Bibliography Information
CloudWhitePaper	SPEC OSG Cloud Working Group whitepaper https://www.spec.org/osgcloud/docs/osgcloudwgreport20120410.pdf
Cbtool	IBM Cloud Bench 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: July 28, 2016