Benchmarking Suite Documentation

Release 3.1.0

Gabriele Giammatteo

Apr 01, 2021

Contents

The Benchmarking Suite is an all-in-one solution for benchmarking cloud services simulating different typical application behaviours and comparing the results on different cloud providers. It wraps a set of representative, de-facto standard and widely used third-party benchmarking tools and relies on them for the workload simulation and performance measurement.

The Benchmarking Suite automates the benchmarking process managing the allocation and de-allocation of necessary resources, the installation and execution of the benchmarking tools and the storage of data.

It has been designed to be extendible and allow an easy integration of new third-party benchmarking tools and cloud services. Data collected and stored during the tests execution is homogenized and aggregated on different higher-level metrics (e.g. average value) allowing performance comparisons among different providers and/or different dates.

The Benchmarking Suite development has been funded by two European reasearch and innovation projects: ARTIST[1] and CloudPerfect[2], and proudly supported by EIT Digital[3] within the EasyCloud innovation activity.

[1] http://www.artist-project.eu/
[2] https://cloudperfect.eu/
[3] https://www.eitdigital.eu/

CHAPTER 1

License

The Benchmarking Suite is an open source product released under the Apache License v2.0[4].

[4] https://www.apache.org/licenses/LICENSE-2.0

CHAPTER 2

Topics

2.1 Quick Start

2.1.1 Install

The Benchmarking Suite is package and distributed through PyPI[1].

Important: The Benchmarking Suite requires Python 3.5+. If it is not the default version in you system, it is recommended to create a virtualenv:

```
virtualenv -p /usr/bin/python3.5 benchmarking-suite
source benchsuite/bin/activate
```

Let's start by installing the command line tool and the standard library:

```
$ pip install benchsuite.stdlib benchsuite.cli
```

This will make available the `benchsuite` bash command and will copy the standard benchmark tests configuration into the default configuration location (located under `~/.config/benchmarking-suite/benchmarks`).

2.1.2 Configure

Before executing a benchmark, we have to configure at least one Service Provider. The `benchsuite.stdlib` provides some template (located under `~/.config/benchmarking-suite/providers`).

For instance, for Amazon EC2 we can start from the template and complete it:

```
cp ~/.config/benchmarking-suite/providers/amazon.conf.example my-amazon.conf
```

[1] https://python.org/pypi/benchsuite.core/

Open and edit `my-amazon.conf`

```
[provider]
class = benchsuite.provider.libcloud.LibcloudComputeProvider
type = ec2

access_id = <your access_id>
secret_key = <your secret_key>

[ubuntu_micro]
image = ami-73f7da13
size = t2.micro
```

In this case we will provide this file directly to the command line tool, but we can also configure our own configuration directory, put all our service providers and benchmarking tests configuration there and refer to them by name.

(Full specification of the configuration files syntax, can be found in the "Service Providers" sections).

2.1.3 Run!

Now you can execute your first benchmark test:

```
benchsuite multiexec --provider my-amazon.conf --service ubuntu_micro ycsb-
↪mongodb:workloada
```

2.1.4 Go REST

Enable the REST server is very simple:

```
pip install benchsuite.rest
benchsuite-rest start
tail -f benchsuite-rest.log
```

2.1.5 References

2.2 Installation

This section discusses the hardware and software requirements as well as the process of installing the Benchmarking Suite.

2.2.1 CLI installation

The Benchmarking Suite is package and distributed through PyPI.

Important: The Benchmarking Suite requires Python 3.5+. If it is not the default version in you system, it is recommended to create a virtualenv:

```
virtualenv -p /usr/bin/python3.5 benchmarking-suite
source benchsuite/bin/activate
```

Let's start by installing the command line tool and the standard library:

```
$ pip install benchsuite.stdlib benchsuite.cli
```

This will make available the `benchsuite` bash command and will copy the standard benchmark tests configuration into the default configuration location (located under `~/.config/benchmarking-suite/benchmarks`).

2.2.2 Service installation

The Benchmarking Suite has also been packaged as a SaaS solution to fit different usage scenarios. This way, once installed on a server, all functionalities will be available through your web browser in a multi-tenant environment.

The Benchmarking Suite is distributed as a multi-container Docker application defined through Docker Compose. Individual components are shipped as Docker images, while the whole suite can be deployed and orchestrated through Docker Swarm.

For each individual service in the 'docker-compose.yml' here is the list of the variables to configure. A description of each service can be found in the next sections.

Note that the template 'docker-compose.yml' in the code repository references a '.env' configuration file where all secrets are configured.

1) Clone the repository:

```
git clone https://gitlab.res.eng.it/benchsuite/benchsuite-compose.git
```

2) Create a *.env* file with the following content:

```
# Keycloak
DB_USER=keycloak-user
DB_PASSWORD=keycloak-pass
KEYCLOAK_USER=admin
KEYCLOAK_PASSWORD=admin

# Postgres (for Keycloak)
POSTGRES_USER=keycloak-user
POSTGRES_PASSWORD=keycloak-password

#Grafana
GF_AUTH_GENERIC_OAUTH_CLIENT_SECRET=<some temporary secret>

#Influx
INFLUXDB_ADMIN_USER=influx-user
INFLUXDB_ADMIN_PASSWORD=influx-password

#Benchsuite API
BENCHSUITE_OIDC_CLIENT_SECRET=<some temporary secret>
BENCHSUITE_DEV_KEY=aSecretDevKey
# AES_KEY must be 16 characters long!
AES_KEY=This_is_a_key123

# Syndication Client
MARKETPLACE_USER=marketplace-user
MARKETPLACE_PASSWORD=marketplace-password
MARKETPLACE_SYNDICATION_EVENT_SIGNATURE_TOKEN=marketplace-token
MARKETPLACE_BENCHSUITE_SKU=marketplace-sku
```

(continues on next page)

(continued from previous page)

```
GRAFANA_USERNAME=grafana-admin-user
GRAFANA_PASSWORD=grafana-admin-password
KEYCLOAK_ADMIN_CLIENT_SECRET_KEY=<some temporary secret>
```

Further details about the Marketplace Syndication are available at: https://docs.cloudesire.com/docs/syndication.html

3) Make sure that INFLUXDB_ADMIN_USER and INFLUXDB_ADMIN_PASSWORD match the values *username* and *password* in the storage.conf file

4) Make sure that DB_USER and DB_PASSWORD match POSTGRES_USER and POSTGRES_PASSWORD

5) Set stack name and network domain env variables:

```
$ export STACK_NAME=bes1

$ export NET_DOMAIN=benchsuite.acme.org
```

6) Create the *traefik.yaml* file for the Traefik dynamic configuration:

```
tls:
  options:
    default:
      minVersion: VersionTLS12
      cipherSuites:
        - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
        - TLS_RSA_WITH_AES_256_GCM_SHA384
```

7) Create an empty file with permission 600 called *traefik-acme.json*. It will contain the certificates generated by Traefik

8) Deploy the stack

```
sudo NET_DOMAIN=$NET_DOMAIN STACK_NAME=$STACK_NAME docker stack deploy -c docker-
↪compose.yml $STACK_NAME
```

9) Setup Keycloak

   a) Access https://\protect\T1\textdollarNET_DOMAIN/auth/

   b) Create a realm named “benchsuite”

   c) Set frontend url in realm settings to “https://\protect\T1\textdollarNET_DOMAIN/auth/”

   d) create a client named “grafana”

      1) set redirect uri to https://\protect\T1\textdollarNET_DOMAIN/charts/*

      2) make it ‘confidential’

      3) create roles admin, viewer and editor

      4) create a mapper for the role (type: “User Client Role”, token claim name: “roles”, json type: “string”)

   e) copy grafana client secret and set the env var

   f) create “benchsuite-api” client

      1) set redirect uri to “https://\protect\T1\textdollarNET_DOMAIN/*”

      2) make it ‘confidential’

   g) create a user, assign grafana role

10) Update the .env with client secrets as generated/set in Keycloak. The following variables must be set:

```
GF_AUTH_GENERIC_OAUTH_CLIENT_SECRET

BENCHSUITE_OIDC_CLIENT_SECRET

KEYCLOAK_ADMIN_CLIENT_SECRET_KEY
```

11) Redeploy the stack

```
$ sudo docker stack rm $STACK_NAME

$ sudo NET_DOMAIN=$NET_DOMAIN STACK_NAME=$STACK_NAME docker stack deploy -c docker-
↪compose.yml $STACK_NAME
```

2.3 Architecture

The Benchmarking Suite is composed by five main components summarized in the following diagram:

- **Core**: defines the main model, the extension framework for the benchmarks and the Cloud providers and the main data representation types;
- **REST**: a REST server to access all the Benchmarking Suite functionalities;
- **CLI**: a command line tool (bash-compatible) to access all the Benchmarking Suite functionalities;
- **Std Library**: a set of selected benchmark tools, including their configurations and the implementation of the required wrapping scripts for the execution and the parsing of results;
- **Backend Connectors**: a set of connectors to store the Benchmarking Suite executions results on different storage technologies (e.g. MySQL, MongoDB).
- **Scheduler**: a service that periodically executes a benchmarking session;

The *Core* component is the only required component, the other components are optional. However the Benchmarking Suite installation will miss the functionalities of not-installed modules (e.g. if the *Backend Connectors* is not installed, the execution results will not be stored).

The *User's Cloud Configuration* is the required configuration of the Cloud Providers that the Benchmarking Suite needs to be able to access the *Target Cloud Provider*. It can be specified either as configuration file or as parameter in the execution requests (through the REST or CLI components). Refer to section *Service Providers* for further details

2.3.1 Domain Model

The core concept in the Benchmarking Suite is the **BenchmarkExecution**. It represents the execution of a **Benchmark** test against an **ExecutionEnvironment** provided from a **ServiceProvider** and produces an **ExecutionResult**.

Note: For instance, following this model we can easily model the execution of YCSB.WorkloadA (the *Benchmark*) on the Virtual Machine with ip=50.1.1.1 (the *ExecutionEnvironment*) provided by Amazon EC2 (the *ServiceProvider*).

Since it is frequent to execute multiple tests against the same Service Provider, the Benchmarking Suite has also the concept of **BenchmarkingSession**. that can gropu one or more executions of the ServiceProvider, using the same ExecutionEnvironment.

2.3.2 Software Modules

In order to address all the different use cases and the installation necessities, the Benchmarking Suite is distributed in six different software modules that can be installed separately:

`benchsuite.core`	the core library (all other modules depend on it) with the definition of types and the fundamental framework for the extension of the Benchmarking Suite
`benchsuite.stdlib`	a collection of benchmark tests configuration files and support for some Cloud Providers
`benchsuite.cli`	a bash command line tool to manage tests and results
`benchsuite.rest`	an HTTP server and a REST API to interact with the Benchmarking Suite
`benchsuite.backend`	connectors for the supported storage backends
`benchsuite.scheduler`	a service to automatically execute periodic benchmarks

2.4 Benchmarks

The Benchmarking Suite comes with a set of third-party benchmarking tools, each of them with a set of different test configurations ready to be executed. The tools are:

- **CFD**: a tool realized in the CloudPerfect EU project[1] that uses OpenFOAM to run a waterbox simulation. Can be configured with different solvers, number of iterations and write to disk strategies. It is primarily a CPU intensive benchmark;
- **DaCapo**: a tool for Java benchmarking simulating real world applications with non-trivial memory loads. It is mainly a CPU and memory intensive benchmark;
- **Filebench**: a powerful and flexible tool able to generate and execute a variety of filesystem workloads to simulate applications like Web servers, File servers, Video services. It is mainly a Disk intensive benchmark;
- **Iperf**: is a tool for active measurements of the maximum achievable bandwidth on IP networks;
- **Sysbench**: a tool to test CPU, memory, file I/O, mutex performance and MySQL on Linux systems;
- **YCSB**: a tool for database benchmarking that supports several database technologies. In the Benchmarking Suite, tests for Mysql and MongoDB are provided. It is primarily a Disk intensive benchmark;
- **WebFrameworks**: tests common web frameworks workloads like fetching and inserting data in a database or create/parse json objects. It is mainly a Memory and Network intensive benchmark;

The following table summarizes the tools available and their compatibility with different operating system.

[1] CloudPerect project homepage: http://cloudperfect.eu/

Table 1: Test-OS compatibility matrix

Tool	Version	CentOS	Ubuntu 14	Ubuntu 16	Ubuntu 18	Ubuntu 20	Debian
CFD	1.0		✓				
DaCapo	9.12	✓	✓				
Filebench	1.4.9.1	✓	✓	✓			
Iperf	2.0.5		✓	✓			
Sysbench	2.1.0		✓	✓			
YCSB-MySQL	0.12.0	✓	✓				
YCSB-MongoDB	0.11.0	✓	✓				
WebFrameworks	master		✓	✓			

2.4.1 CFD

The CFD benchmarking tool has been realized in the context of the CloudPerfect EU project[1] and released open source on GitHub[2]. The tool executes a CFD simulation on a waterbox geometry allowing to customize several parameters in order to simulate different simulations.

The following combination of parameters is used in the Benchmarking Suite tests:

100iterGAMG	100 iterations using the GAMG solver
100iterWriteAtLast	100 iterations using the GAMG solver and not writing intermediate results on the disk
500iterGAMG	500 iterations using the GAMG solver
500iterGAMGWriteAtLast	500 iterations using the GAMG solver and not writing intermediate results on the disk
500iterICCG	500 iterations using the ICCG solver
500iterPCG	500 iterations using the PCG solver

All the tests uses all the CPUs available in the machine.

Metrics

Metric	Unit	Description
duration	s	The overall duration of the simulation

2.4.2 DaCapo

DaCapo[3] as a tool for Java benchmarking by the programming language, memory management and computer architecture communities. It consists of a set of open source, real world applications with non-trivial memory loads. Tests implemented by the tool are:

[2] CFD Benchmark Case code: https://github.com/benchmarking-suite/cfd-benchmark-case
[3] DaCapo homepage: http://www.dacapobench.org/

Table 2: DaCapo tests (source: http://www.dacapobench.org/)

avrora	simulates a number of programs run on a grid of AVR microcontrollers
batik	produces a number of Scalable Vector Graphics (SVG) images based on the unit tests in Apache Batik
eclipse	executes some of the (non-gui) jdt performance tests for the Eclipse IDE
fop	takes an XSL-FO file, parses it and formats it, generating a PDF file.
h2	executes a JDBCbench-like in-memory benchmark, executing a number of transactions against a model of a banking application, replacing the hsqldb benchmark
jython	inteprets a the pybench Python benchmark
luindex	Uses lucene to indexes a set of documents; the works of Shakespeare and the King James Bible
lusearch	Uses lucene to do a text search of keywords over a corpus of data comprising the works of Shakespeare and the King James Bible
pmd	analyzes a set of Java classes for a range of source code problems
sunflow	renders a set of images using ray tracing
tomcat	runs a set of queries against a Tomcat server retrieving and verifying the resulting webpages
tradebeans	runs the daytrader benchmark via a Jave Beans to a GERONIMO backend with an in memory h2 as the underlying database
tradesoap	runs the daytrader benchmark via a SOAP to a GERONIMO backend with in memory h2 as the underlying database
xalan	transforms XML documents into HTML

Each test is executed multiple times, until the exectuions duration converge (variance is <= 3.0 in the latest 3 executions).

Metrics

Metric	Unit	Description
timed_duration	ms	the duration of the latest execution
warmup_iters	num	the number of executions that were necessary to converge

2.4.3 Filebench

Filebench[4] is a very powerful tool able to generate a variety of filesystem- and storage-based workloads. It implements a set of basic primitives like *createfile*, *readfile*, *mkdir*, *fsync*, ... and provide a language (the Workload Model Language - WML) to combine these primitives in complex workloads.

In the Benchmarking Suite, a set of pre-defined workloads have been used to simulate different services:

[4] Filebench homepage: https://github.com/filebench/filebench/wiki

Table 3: Filebench workloads (source: https://github.com/filebench/filebench/wiki/Predefined-personalities)

file-server	Emulates simple file-server I/O activity. This workload performs a sequence of creates, deletes, appends, reads, writes and attribute operations on a directory tree. 50 threads are used by default. The workload generated is somewhat similar to SPECsfs.
webproxy	Emulates I/O activity of a simple web proxy server. A mix of create-write-close, open-read-close, and delete operations of multiple files in a directory tree and a file append to simulate proxy log. 100 threads are used by default.
web-server	Emulates simple web-server I/O activity. Produces a sequence of open-read-close on multiple files in a directory tree plus a log file append. 100 threads are used by default.
videoserver	This workloads emulates a video server. It has two filesets: one contains videos that are actively served, and the second one has videos that are available but currently inactive. One thread is writing new videos to replace no longer viewed videos in the passive set. Meanwhile $nthreads threads are serving up videos from the active video fileset.
var-mail	Emulates I/O activity of a simple mail server that stores each e-mail in a separate file (/var/mail/ server). The workload consists of a multi-threaded set of create-append-sync, read-append-sync, read and delete operations in a single directory. 16 threads are used by default. The workload generated is somewhat similar to Postmark but multi-threaded.

Metrics

Metric	Unit	Description
duration	s	The overall duration of the test
ops	num	The sum of all operations (of any type) executed
ops_throughput	ops/s	The average number of operations executed per second
throughput	MB/s	The average number of MBs written/read during the test
cputime	µs	The average cpu time taken by each operation
latency_avg	µs	The average duration of each operation

2.4.4 Iperf

IPerf[5] is a benchmarking tool to measure the maximum achievable bandwidth on IP networks. It provides statistics both for TCP and UDP protocols.

In the Benchmarking Suite, the following pre-defined workloads have been created:

tcp_10_1	transfer data over a single TCP connections for 10 seconds
tcp_10_10	transfer data over 10 parallel TCP connections for 10 seconds
udp_10_1_1	transfer UDP packets over a single connection with a maximum bandwidth limited at 1MBit/s
udp_10_1_10	transfer UDP packets over a single connection with a maximum bandwidth limited at 10MBit/s
udp_10_10_10	transfer UDP packets over 10 parallel connections with a maximum bandwidth limited at 1MBit/s

Metrics

For the TCP workloads:

[5] IPerf homepage: https://iperf.fr/

Metric	Unit	Description
duration	s	The overall duration of the test
transferred_x	bytes	data transferred for the connection x
bandwidth_x	bit/s	bandwidth fo the connection x
transferred_sum	bytes	sum of data transferred in all connections
bandwidth_sum	bit/s	sum of bandwidth of all connections

For the UDP workloads:

Metric	Unit	Description
duration	s	The overall duration of the test
transferred_x	bytes	data transferred over connection x
bandwidth_x	bit/s	bandwidth of connection x
total_datagrams_x	num	number of UDP packets sent over connection x
lost_datagrams_x	num	number of lost UDP packets over connection x
jitter_x	ms	latency of connection x
outoforder_x	num	number of packets received by the server in the wrong order
transferred_avg	bytes	average data transferred by each connection
bandwidth	bit/s	average bandwidth of each connection
total_datagrams_avg	num	average number of packets sent over each connection
lost_datagrams_avg	num	average number of packets lost for each connection
jitter_avg	ms	average latency
outoforder_avg	num	average number of packets received in the wrong order

2.4.5 Sysbench

SysBench[6] is a modular, cross-platform and multi-threaded benchmark tool for evaluating CPU, memory, file I/O, mutex performance, and even MySQL benchmarking. At the moment, in the Benchmarking Suite only the CPU benchmarking capabilities are integrated.

cpu_10000	Verifies prime numbers between 0 and 20000 by doing standard division of the number by all numbers between 2 and the square root of the number. This is repeated 1000 times and using 1, 2, 4, 8, 16 and 32 threads

Metrics

Metric	Unit	Description
events_rate_X	num/s	the number of times prime numbers between 0 and 20000 are verified each second with X threads
total_time_X	s	total number of seconds it took to execute the 1000 cycles with X threads
latency_min_X	ms	minimum time it took for a cycle
latency_max_X	ms	maximum time it took for a cycle
latency_avg_X	ms	average time the 1000 cycles took. It gives a good measure of the cpu speed
latency_95_X	ms	95th percentile of the latency times.

[6] Sysbench homepage: https://github.com/akopytov/sysbench

2.4.6 YCSB

YCSB[7] is a database benchmarking tool. It has the support for several database technologies and provides a configuration mechanism to simulate different usages.

In the Benchmarking Suite, YCSB is used to benchmark two of the most popular database servers: **MySQL** and **MongoDB**.

For each database, the following workloads are executed:

workloada	Simulates an application that performs read and update operations with a ratio of 50/50 (e.g. recent actions recording)
workloadb	Simulates an application that performs read and update operations with a ratio of 95/5 (e.g. photo tagging)
workloadc	Simulates a read-only databases (100% read operations)
workloadd	Simulates an application that performs read and insert operations with a ratio of 95/5 (e.g. user status update)
workloade	Simulates an application that performs scan and insert operations with a ratio of 95/5 (e.g. threaded conversations)
workloadf	Simulates an application that performs read and read-modify-write operations with a ratio of 50/50 (e.g. user database)

Metrics

Metric	Unit	Description
duration	s	The overall duration of the test
read_ops	num	THe number of read operations executed
read_latency_avg	µs	The average latency of the read operations
read_latency_min	µs	The minimum latency of the read operations
read_latency_max	µs	The maximum latency of the read operations
read_latency_95	µs	The maximum latency for the 95% of the read operations
read_latency_99	µs	The maximum latency for the 99% of the read operations
insert_ops	num	THe number of insert operations executed
insert_latency_avg	µs	The average latency of the insert operations
insert_latency_min	µs	The minimum latency of the insert operations
insert_latency_max	µs	The maximum latency of the insert operations
insert_latency_95	µs	The maximum latency for the 95% of the insert operations
insert_latency_99	µs	The maximum latency for the 99% of the insert operations
update_ops	num	THe number of update operations executed
update_latency_avg	µs	The average latency of the update operations
update_latency_min	µs	The minimum latency of the update operations
update_latency_max	µs	The maximum latency of the update operations
update_latency_95	µs	The maximum latency for the 95% of the update operations
update_latency_99	µs	The maximum latency for the 99% of the update operations

[7] YCSB homepage: https://github.com/brianfrankcooper/YCSB/wiki

2.4.7 WebFrameworks

This is an open source tool[8] used to compare many web application frameworks executing fundamental tasks such as JSON serialization, database access, and server-side template composition. The tool has been developed and it is used to run the tests that generate the results available at: https://www.techempower.com/benchmarks/.

Currently, in the Benchmarking Suite the framework supported are: **Django**, **Spring**, **CakePHP**, **Flask**, **FastHttp** and **NodeJS**.

For each framework the following tests are executed:

Table 4: Test types (source: https://www.techempower.com/benchmarks/#section=code&hw=ph)

json	This test exercises the framework fundamentals including keep-alive support, request routing, request header parsing, object instantiation, JSON serialization, response header generation, and request count throughput.
query	This test exercises the framework's object-relational mapper (ORM), random number generator, database driver, and database connection pool.
fortunes	This test exercises the ORM, database connectivity, dynamic-size collections, sorting, server-side templates, XSS countermeasures, and character encoding.
db	This test uses a testing World table. Multiple rows are fetched to more dramatically punish the database driver and connection pool. At the highest queries-per-request tested (20), this test demonstrates all frameworks' convergence toward zero requests-per-second as database activity increases.
plaintext	This test is an exercise of the request-routing fundamentals only, designed to demonstrate the capacity of high-performance platforms in particular. Requests will be sent using HTTP pipelining.
update	This test exercises the ORM's persistence of objects and the database driver's performance at running UPDATE statements or similar. The spirit of this test is to exercise a variable number of read-then-write style database operations.

For the types *json*, *query*, *fortunes* and *db* the tool executes six different burst of requests. Each burst last 15 seconds and have a different concurrency level (number of requests done concurrently): 16, 32, 64, 128, 256 and 512.

For the type *plaintext*, the tool executes four burst of 15 seconds each with the following concurrency levels: 256, 1024, 4096 and 16384.

For the type *update*, the tool executes five burst of 15 seconds each with a 512 concurrency level, but different number of queries to perform: 1, 5, 10, 15 and 20.

Metrics

Metric	Unit	Description
duration	s	The overall duration of the test
duration_N	s	The overall duration for the N concurrency level*. It is fixed to 15 seconds by default
totalRequests_N	num	The overall number of requests processed during the 15 seconds test at the N concurrency level*
timeout_N	num	The number of requests that went in timeout for the N concurrency level*
latencyAvg_N	s	the average latency between a request and its response for the N concurrency level*
latencyMax_N	s	the maximum latency between a request and its response for the N concurrency level*
latencyStdev_N	s	the standard deviation measure for the latency for the N concurrency level*

[8] Web Framewoks Benchmarking code: https://github.com/TechEmpower/FrameworkBenchmarks

2.4.8 Adding a new benchmarking tool

In addition to the benchmarking tests coming with the standard Benchmarking Suite release, it is possible to add new benchmarking tools by providing a configuration file to instruct the Benchmarking Suite how to install, configure and execute the tool.

The configuration file must contain on section `[DEFAULT]` with the commands to install and execute the benchmarking tool, plus one or more sections that define different sets of input parameters to the tool. In this way, it is possible to execute the same tool to generate multiple workloads.

```
[DEFAULT]
class = benchsuite.stdlib.benchmark.vm_benchmark.BashCommandBenchmark


#
# install, install_ubuntu, install_centos_7 are all valid keys
install_<platform> =
    echo "these are the..."
    echo "...install %(option1)s commands"

execute_<platform> =
    echo "execute commands"

cleanup =
    echo "commands to cleanup the %(option2)s environment"



[workload_1]
option1 = value1
option2 = value

[workload_n]
option1 = value1
option2 = valueN
```

For instance, a very minimal configuration file to integrate the Sysbench[6] benchmarking tool is shown below:

```
[DEFAULT]
class = benchsuite.stdlib.benchmark.vm_benchmark.BashCommandBenchmark

install =
    curl -s https://packagecloud.io/install/repositories/akopytov/sysbench/script.deb.
↪sh | sudo bash
    sudo apt-get -yq install sysbench
    sysbench %(test)s prepare %(options)s

execute =
    sysbench %(test)s run %(options)s --time=0

cleanup =
    sysbench %(test)s cleanup %(options)s

[cpu_workload1]
    test = cpu
    options = --cpu-max-prime=20000 --events=10000
```

Configuration files of the benchmarks included in the Benchmarking Suite releases can be used as starting point and

are available here[10]

2.4.9 Managing benchmarking tools through the GUI

The Benchmarking Suite comes with a set of third-party, widely-known, open source benchmarking workloads (e.g. SysBench, FileBench, DaCapo, Web Framework Benchmarking). These workloads are available to any registered user and are encouraged to be used to enable comparability of results along time and across providers and users. However, to support specific user requirements, custom workloads can be defined and, according to the user choice, shared with others or kept private. As with the CLI, workloads registered in the Benchmarking Suite are typically a wrapper around existing benchmarking applications for which the registration process should provide installation, execution and results parsing capabilities.

When the Benchmarking Suite is used through the web interface, benchmarking tools (a.k.a. workloads) can be added and edited as well.

From the 'Workload' panel, a new workload can be *added* via the 'New Workload' button which shows a form asking for metadata already presented in the previous section.

Similarly, once a workload has been selected, it can be *modified*, *cloned* into a new one or *deleted* (provided you have enough permissions on it).

Workload metadata

- **Workload name** is a name, not necessarily unique, given to the workload;
- **Tool name** is the name of the tool providing the given workload;
- **Workload ID** is a unique identifier provided by the system, and is not modifiable;
- **Description** is to tell what the workload does, what parameter is measured, and any other useful detail about the the workload;
- **Categories** allows to specify some tags to ease search;
- **Abstract** is to mark workload 'templates' not meant for execution but only for specialization;
- **Parent workload** is a base workload definition from which properties/commands are inherited. The current workload only needs to specialize some of them. A typical usage of this feature is to define multiple workloads provided by a single tool;

Workload execution

- **Install** scripts are executed just after the provisioning of the virtual machine, usually to download and install a benchmarking tool;
- **Post-create** scripts are executed after the provisioning of the virtual machine to perform some general initialization (e.g. configure the DNS);
- **Execute** scripts are executed to perform the benchmark of the environment;
- **Cleanup** scripts are executed for multi-benchmark execution to ensure a clean environment for following benchmarks;
- **User and Support** scripts and **Workload parameters** are meant to be used in above scripts for readability and better coding style;

[10] Benchmark configuartion files: https://github.com/benchmarking-suite/benchsuite-stdlib/tree/master/data/benchmarks

Install, *post-create*, *execute* and *cleanup* scripts can be specialized for a specific operating system, as well as for specific version of the operating system. This is achieved through a mechanism matching the environment on the target VM with the name of the script.

As an example, when executing the workload on a VM running Debian 10.8, the following install scripts are searched:

1. install_debian_10_8
2. install_debian_10
3. install_debian
4. install

The first matched script (i.e. the most-specific one) is executed; the others are ignored.

Sharing workloads

- **Sharing** sets the level of visibility for the workload. A workload can be private to its creator or publicly visible and, thus, executable. Note that this applies to the workload only and not to infrastructure being benchmarked nor to produced results which have their own visibility levels;

Exporting workloads

A JSON representation of the workload can be generated for offline inspection/editing. This can be done for all visible workloads (hit the 'Export All' button) or for individual workloads (hit the 'Export' button when viewing the workload).

When exported workloads are linked by some inheritance relationship, you can decide to export them so that:

1) the hierarchy is preserved;

2) inherited properties are collapsed into most-specific workloads (i.e. the hierarchy is lost).

2.5 Service Providers

2.5.1 Configuration

Each provider has its own configuration file that provides all the information to access and use the provider plus the configuration of the services offered by the provider (e.g. VM creation). The configuration is specified in a properties file (also in JSON format). The list of supported properties is shown below (see on GitHub[1]):

```
#
# PROVIDER SECTION
#
#

[provider]

#
# MANDATORY FIELDS
#

# the Benchmarking Suite class that implement the access to this provider
```

(continues on next page)

[1] https://github.com/benchmarking-suite/benchsuite-stdlib/blob/master/data/providers/openstack.conf.example

(continued from previous page)

```
class = benchsuite.stdlib.provider.libcloud.LibcloudComputeProvider

# custom human-readable name for the provider. Used only for displaying purposes
name = ote-testbed

# the driver that Libcloud should use to access the provider
driver = openstack

# Access credentials
access_id = admin
secret_key = xxxxx

# Authentication URL for the target Cloud
auth_url = http://cloudpcntlr:5000/


#
# OPTIONAL FIELDS
#

# If not specified RegionOne is used
region = RegionOne

# if not specified 2.0_password and 3.x_password will be attempted
auth_version = 3.x_password

# if not specified, a new security group "benchsuite_sg" will be created (if
# not exist)
security_group = mysg

# Automatically selected if multiple exist
network = mynet

# If not specified the access_id is used
tenant = admin

# If specified do not attemp to assign a public IP
benchsuite.openstack.no_floating_ip = true


# after a new VM is created, a connection test is executed to check that everything
# is ok and to execute the post-creation scripts.
new_vm.connection_retry_period = 30
new_vm.connection_retry_times = 10


#
# SERIVCE TYPE SECTIONS
#
#

[ubuntu_large]

#
# MANDATORY FIELDS
#
```

(continues on next page)

(continued from previous page)

```
image = base_ubuntu_14.04
size = m1.large


#
# OPTIONAL FIELDS
#

# name of the keypair to assign to the VM. If not specified, a new keypair will
# be created and then removed after the test ends.
key_name = ggiammat-key

# file that contains the private key. Alternatively the private key can be
# specified in this config file directly with the ssh_private_key property
key_path = /home/ggiammat/credentials/filab-vicenza/ggiammat-key.pem

# optional way to specify the key directly in the configuration instead
# of by filename
ssh_private_key = -----BEGIN RSA PRIVATE KEY-----
    MIIEowIBAAKCAQEAkadPr5n1NSOyHloajvovCD05M5Gz36NN4UouSWmId8QuTwXx
    Hw6m9aOXJmYHdkSYLrNs+y65EDpUkw1DXNDEJ146ZK9PxAQEdcngwPk76a4A/ybz
    [...]
    x+GRpQ9o/4EAzpBw9NVNNJ9Glbd7SSFqhpHR5pn5OBG/fdPJV8DzjUET528o8Jd9
    gynGwAYRed38UtCE7gn+u1RSvmYUveDwQ7Cf2KIohI2jlzR6YLea
    -----END RSA PRIVATE KEY-----

# If not specified, the Benchmarking Suite will try to guess them
vm_user = ubuntu
platform = ubuntu

# any command to run just after the VM has been created
post_create_script =
    sudo hostname localhost
```

2.5.2 Managing service providers through the GUI

This functionality allows the registration of cloud service accounts you own, on top of which benchmarks are to be executed.

Besides basic metadata (e.g. name and description) Cloud services are characterized by a set of connection parameters required to create ephemeral virtual machines to support benchmarking sessions.

Regardless of the type of benchmark that will be executed, VMs might need to be configured just after the provisioning process; to this end, the Cloud Services Management allows the definition of custom commands to be executed on the target VM, also supporting the differentiation among target OS (e.g. CentOS, Ubuntu, Ubuntu_18, Ubuntu_20, etc.).

Supported Cloud technologies include OpenStack, VMWare vCloud and Amazon EC2.

From the 'Providers' panel, a new service provider can be *added* via the 'New Provider' button which shows a form asking for a few provider metadata.

Similarly, once a provider has been selected, it can be *modified*, *cloned* into a new one or *deleted* (provided you have enough permissions on it).

Provider metadata

- **Provider name** is a name, not necessarily unique, given to the provider;
- **Provider ID** is a unique identifier provided by the system, and is not modifiable;

Connection metadata

This section contains coordinates needed to connect to the service provider in order to create virtual machines. The exact set of metadata depends on the selected cloud technology:

OpenStack

- **Authentication method**
- **Authentication URL**
- **Tenant name**
- **Region**
- **Domain**
- **Username**
- **Password**

VMWare

- **Host**
- **Organization**
- **VDC**
- **Username**
- **Password**

Amazon EC2

- **Region**
- **Username**
- **Password**

Service accounts

This section contains credentials to access services (i.e. VMs).

The base property names for credentials are 'vm_user' and 'vm_password'. But os-specific credentials can be set by suffixing the properties with os name/version (e.g. 'vm_user_debian', 'vm_user_debian_10', 'vm_user_debian_10_9', . . .)

Post-create scripts

Post-create scripts are executed after the provisioning of the virtual machine to perform some general initialization (e.g. configure the DNS).

As with service accounts, OS-specific scripts can be set through naming mechanism, like in the following example:

- **debian_10_8**: *post-install script to execute on Debian 10.9 VMs*
- **debian**: *post-install script to execute on any other Debian VM*
- **centos**: *post-install script to execute on any CentOS VM*
- **default**: *post-install script to execute on any VMs*

Additional scripts can also be defined to be used within the above ones (to define some action to be taken for multiple platforms, e.g. configure_dns).

Custom properties

Depending on the provider technology, additional connection properties might be needed; they can be set here.

Additionally, properties set here can be referenced within *post-create* scripts.

Sharing

Cloud services are always kept private to their owner, although some general metadata (e.g. name and description) might appear along with associated metrics if these are shared to an organization or made public.

2.6 Backends

During execution of tests, the Benchmarking Suite produces different type of data:

- metrics:
- logs
- metadata on the execution environment, the test and the provider

The Benchmarking Suite supports multiple backends technologies to store data

Multiple backends can be active at the same time

2.6.1 Configuration format

2.6.2 Supported Providers

StdOutBackend

TBD

MongoDB

TBD

InfluxDB

TBD

2.7 Executions

Execution management grants the user the capability to run benchmarks on any other cloud service the user registered in the platform.

With a single request, the user can trigger the execution of any number of workloads, on different sizes of different VM images of a given cloud service.

The backend services, then, take care of the VMs provisioning process, benchmarking installation, execution and results collection or, in case of failure, logs and exception reporting to enable troubleshooting.

If multiple VMs are expected to be created, they will be provisioned on the VDC sequentially to minimize the impact on the resource quota the user might have on the VDC.

Similarly, if multiple benchmarks have been requested, they are executed sequentially within the same VM to minimise provisioning requests and, thus, overhead times and resource utilization.

Execution management provides the user with capabilities to browse previous executions and to inspect collected metrics and logs.

At submission time, the user can also control the visibility scope of collected metrics: results can be kept private, shared within the user organization or made available to any registered user.

2.7.1 Execution a benchmark through the GUI

From the 'Executions' panel, a benchmark can be ran via the 'New Execution' button.

Provider

The first thing to do is select a Service Provider to benchmark. Here you'll find the list of providers you registered in the 'Provider' panel.

Benchmarked Resources

Once a provider has been selected, this section presents the list of services types available at that provider. Typically it consists of VM images and available flavours.

Multiple service types can be entered here.

Workloads

In this section you can select the workloads to execute on provider resources.

Multiple workloads can be entered here.

Configuration

Entries in this section can control various aspects of the benchmark execution:

Configure the workload

Some workloads allow customization via *properties* and/or *environment variables* on the host VM. Refer to the documentation of each workload to learn more about them.

Configure the benchmarking command

This section is to set additional options to the benchmarking controller command.

Configure the Docker image creation

This section is to set additional options for the benchmarking controller container.

Sharing results

By default, collected benchmarking metrics are private (i.e. visibile to you only).

However, you might want to share them with a wider audience: if made *public*, they will be visibile to any registered user; when shared with an *organization*, all members of that organization will have access to them.

Note: organization membership is managed externally to the benchmarking suite (e.g. through a LDAP, Keycloak, etc.)

2.7.2 Browsing previous benchmark executions

By selecting the 'Executions' panel, most-recent benchmark executions are shown. This list contains your own execution and those ran by others, where they shared results with you.

Both on-demand and scheduled executions appear here. You can tell what the trigger was by looking at the entry icon: *a rocket* for on on-demand and *gears* for scheduled executions.

The list shows some general information about the execution. Selecting a workloads will show full details about it. Such details contain:

- **start time**, **duration** and overall **status** of the execution.
- details about the **service provider**. If you are not the owner of this provider, you might see a few things about it (i.e. name and description).
- if the execution was triggered by a **schedule**, this section shows some basic info about it *as it was at execution time*.
- for each combination of selected workload, image and size, there's a section with **metrics**, **logs** and **error messages**.

2.7.3 Re-submitting an execution

Completed executions can be re-ran as they were submitted. You're not constrained to the original submission request: everything can be modified/removed in the new submission.

2.8 Command line tool

2.8.1 Install

The Benchmarking Suite Command line tool can be installed with:

```
pip install benchsuite.cli
```

If the installation was successful, the `benchsuite` command should be in your path.

2.8.2 Configuration

The Benchmarking Suite has a configuration folder where the providers, benchmarks and storage backends configuration files are located. If not explicitly set, the configuration folder is the located in the system default configuration folder (e.g. `/home/<user>/.config` in Linux). It can be customized setting the `BENCHSUITE_CONFIG_FOLDER` env variable.

The Benchmarking Suite stores the active benchmarking sessions data on the filesystem. By default it is located under the system default data folder (e.g. `/Home/<user>/.local/share` in Linux). It can be customized setting the `BENCHSUITE_DATA_FOLDER` env variable.

2.8.3 Usage and Examples

Create a new session

To create a new session, the Benchmarking Suite needs two information: the provider configuration and the service type. The command line tool offers multiple options to specify these parameters.

Provider Configuration

There are different alternatives:

1. specify a name with the `--provider` option (e.g. `--provider myamazon`). In this case, a provider configuration file named `<name>.conf` will be searched in the configuration path;
2. specify a filename in the `--provider` option (e.g. `--provider /path/myamazon.conf`). The configuration file specified will be used;
3. store the provider configuration in the `BENCHSUITE_PROVIDER` environment variable.

As example, the following invocations load the same provider configuration:

```
$ benchsuite new-session --provider my-amazon.conf ...
```

```
$ benchsuite new-session --provider $BENCHSUITE_CONFIG_FOLDER/providers/my-amazon.
↪conf ...
```

```
$ export BENCHSUITE_PROVIDER=```cat $BENCHSUITE_CONFIG_FOLDER/providers/my-amazon.
↪conf```
$ benchsuite new-session ...
```

Service Type

The service type can be specified using the `--service-type` option (e.g. `--service-type ubuntu_micro`). The value of the service type must be one of the ones defined in the provider configuration. Alternatively the ‘‘BENCHSUITE_SERVICE_TYPE‘ enviornment variable can be used.

If neither the `--service-type` option nor the ‘‘BENCHSUITE_SERVICE_TYPE‘ enviornment variable are specified and the provider configuration defines only ONE service type, that one will be used, otherwise the invocation will fail.

2.8.4 Command Line Tool Documentation

This is an autogenerated documentation from the Python `argparse` options.

```
usage: benchsuite [-h] [--verbose] [--quiet] [--config CONFIG]
                  {shell,new-session,new-exec,prepare-exec,cleanup-exec,run-exec,list-
↪sessions,list-providers,list-benchmarks,destroy-session,list-execs,collect-exec,
↪multiexec}
                  ...
```

Named Arguments

--verbose, -v print more information (3 levels)

--quiet, -q suppress normal output

Default: False

--config, -c foo help

Sub-commands:

shell

Starts an interactive shell

```
benchsuite shell [-h]
```

new-session

Creates a new benchmarking session

```
benchsuite new-session [-h] [--provider PROVIDER]
                       [--service-type SERVICE_TYPE] [--property PROPERTY]
                       [--user USER] [--tag TAG]
```

Named Arguments

--provider, -p The name for the service provider configuration or the filepath of the provider configuration file. Alternatively, the provider configuration can be specified in the environment variable BENCHSUITE_PROVIDER (the content of the variable must be the actual configuration not the filepath)

--service-type, -s The name of one of the service types defined in the provider configuration. Alternatively, it can be specified in the BENCHSUITE_SERVICE_TYPE environment varaible

--property, -P	Add a user defined property to the session. The property must be expressed in the format <name>=<value>
--user, -u	sets the "user" property. It is a shortcut for "–property user=<name>
--tag, -t	sets one or more session tags. Internally, tags are stored as properties

Example: benchsuite new-session -p myamazon -s centos_tiny

new-exec

Creates a new execution

```
benchsuite new-exec [-h] session tool workload
```

Positional Arguments

session	a valid session id
tool	a valid benchmarking tool
workload	a valid benchmarking tool workload

Example: benchsuite new-exec 73cff747-d31a-488c-98f5-a70b9a77a11f filebench varmail

prepare-exec

Executes the install scripts for an execution

```
benchsuite prepare-exec [-h] id
```

Positional Arguments

id	a valid id of the execution

Example: benchsuite prepare-exec 4a5a86d4-88b6-11e7-9f96-742b62857160

cleanup-exec

Executed the cleanup script for an execution

```
benchsuite cleanup-exec [-h] id
```

Positional Arguments

id	a valid id of the execution

Example: benchsuite cleanup-exec 4a5a86d4-88b6-11e7-9f96-742b62857160

run-exec

Executes the execute scripts for an execution

```
benchsuite run-exec [-h] [--storage-config STORAGE_CONFIG] [--async] id
```

Positional Arguments

id	a valid id of the execution

Named Arguments

--storage-config, -r	Specify a custom location for the storage configuration file
--async	start the execution of the scripts and return (do not wait for the execution to finish) Default: False

Example: benchsuite run-exec 4a5a86d4-88b6-11e7-9f96-742b62857160

list-sessions

a help

```
benchsuite list-sessions [-h]
```

list-providers

a help

```
benchsuite list-providers [-h]
```

list-benchmarks

a help

```
benchsuite list-benchmarks [-h]
```

destroy-session

a help

```
benchsuite destroy-session [-h] id
```

Positional Arguments

id	bar help

list-execs

lists the executions

```
benchsuite list-execs [-h]
```

collect-exec

collects the outputs of an execution

```
benchsuite collect-exec [-h] id
```

Positional Arguments

id the execution id

multiexec

Execute multiple tests in a single benchmarking session

```
benchsuite multiexec [-h] [--provider PROVIDER] [--service-type SERVICE_TYPE]
                     [--storage-config STORAGE_CONFIG] [--property PROPERTY]
                     [--user USER] [--tag TAG] [--failonerror] [--keep-env]
                     [--max-retry MAX_RETRY]
                     tests [tests ...]
```

Positional Arguments

tests one or more tests in the format <tool>[:<workload>]. If workload is omitted, all workloads defined for that tool will be executed

Named Arguments

--provider, -p The name for the service provider configuration or the filepath of the provider configuration file

--service-type, -s The name of one of the service types defined in the provider configuration. If not specified, all service types will be used

--storage-config, -r Specify a custom location for the storage configuration file

--property, -P Add a user defined property to the session. The property must be expressed in the format <name>=<value>

--user, -u sets the "user" property. It is a shortcut for "–property user=<name>

--tag, -t sets one or more session tags. Internally, tags are stored as properties

--failonerror, -e If set, exits immediately if one of the tests fail. It is false by default

Default: False

--keep-env, -k If set, doesn't destroy the session after the end of the tests

Default: False

--max-retry, -m If a test fails, retry for a maximum of max-retry times (default is 1)

Example: benchsuite multiexec -p myamazon -s centos_tiny cfd:workload1 ycsb:workloada ycsb:workloadb

2.9 REST Server

2.9.1 Quick Start

This short tutorial shows how to use the API to perform a step-by-step benchmarking test.

First, we need to create a new session. This can be done making a `POST` request to `/api/v1/sessions` providing the provider name and service type.

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/
↪json' -d '{\
    "provider": "my-provider",\
    "service": "my-service-type"\
}' http://localhost:5000/api/v1/sessions
```

Alternatively, the provider configuration can be provided directly in the request payload. For instance, a typical request to create a benchmarking session for Amazon EC2 would be:

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/
↪json' -d '{\
    "config": {\
        "provider": {\
                "class": "benchsuite.stdlib.provider.libcloud.LibcloudComputeProvider
↪",\
                    "name": "ec2-ggiammat",\
                    "driver": "ec2",\
                    "access_id": "<your_access_id>",\
                    "secret_key": "<your_key>",\
                    "region": "us-west-1"\
            },\
            "centos_micro": {\
                    "image": "ami-327f5352",\
                    "size": "t2.micro",\
                    "vm_user": "ec2-user",\
                    "platform": "centos_6",\
                    "key_name": "<keypair_name>",\
                    "ssh_private_key": "----BEGIN RSA PRIVATE KEY-----\nIIE... [...] .
↪..6alL\n-----END RSA PRIVATE KEY-----"\
            }\
    }\
}' http://localhost:5000/api/v1/sessions/
```

Important: Providing the configuration directly in the request payload, your credentials will be sent over the network unencrypted. Do it only when the server is running in a trusted environment!

Note: The ssh private key msut be provided on a sinlge line (json does not support multiline values), but the line ends must be preserved. A convenient method to generate this string in bash is:

```
sed -E ':a;N;$!ba;s/\r{0,1}\n/\\n/g' my-key.pem
```

The response will contain the `id` of the session created:

```
{
    "id": "58920c6c-c57c-4c55-a227-0ab1919e83be",
    [...]
}
```

Now we can create a new benchmarking test execution in the session (note that the id of the session is used in the request URL:

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/
↪json' -d '{ \
    "tool": "idle", \
    "workload": "idle30" \
}' http://localhost:5000/api/v1/sessions/58920c6c-c57c-4c55-a227-0ab1919e83be/
↪executions/
```

The response will contain (along with other execution details) the `id` of the execution:

```
{
    "id": "253d9544-b3db-11e7-8bc2-742b62857160",
    [...]
}
```

With this execution id we can now invoke the *prepare* step that will create the resources on the provider, install the necessary tools and load the workloads:

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/
↪json' http://localhost:5000/api/v1/executions/253d9544-b3db-11e7-8bc2-742b62857160/
↪prepare
```

Finally, we can invoke the *run* step:

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/
↪json' http://localhost:5000/api/v1/executions/253d9544-b3db-11e7-8bc2-742b62857160/
↪run
```

The response of the *prepare* and *run* steps contain the start time and the duration of the operation:

```
{
    "started": "2017-10-18 08:18:33",
    "duration": "32.28253793716431"
}
```

The same session can be used to run multiple executions. At the end, the session and the resources created (e.g. VM) can be destroyed using the DELETE operation:

```
curl -X DELETE --header 'Accept: application/json' http://localhost:5000/api/v1/
↪sessions/58920c6c-c57c-4c55-a227-0ab1919e83be
```

2.9.2 Swagger Doc

This documentation is autogenerated from the Swagger API Specification using sphinx-swaggerdoc.

A better documentation for the REST API can be found directly in the REST Server:

1. Launch the server
2. Open http://localhost:5000/api/v1/

sessions

POST /sessions/

Parameters

Name	Position	Description	Type
payload	body		
X-Fields	header	An optional fields mask	string

GET /sessions/

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string

DELETE /sessions/{session_id}

Parameters

Name	Position	Description	Type
session_id	path	The id of the session	string

GET /sessions/{session_id}

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string
session_id	path	The id of the session	string

POST /sessions/{session_id}/executions/

Parameters

Name	Position	Description	Type
payload	body		
X-Fields	header	An optional fields mask	string
session_id	path		string

GET /sessions/{session_id}/executions/

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string
session_id	path		string

---

executions

GET /executions/

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string

GET /executions/user_actions

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string

GET /executions/{execution_id}

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string
execution_id	path	The id of the execution	string

GET /executions/{execution_id}/request

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string
execution_id	path	The id of the execution	string

GET /executions/{execution_id}/runs

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string
execution_id	path	The id of the execution	string

---

providers

POST /providers/

Parameters

Name	Position	Description	Type
payload	body		
X-Fields	header	An optional fields mask	string

GET /providers/

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string

GET /providers/user_actions

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string

DELETE /providers/{provider_id}

Parameters

Name	Position	Description	Type
provider_id	path	The id of the provider	string

PUT /providers/{provider_id}

Parameters

Name	Position	Description	Type
payload	body		
X-Fields	header	An optional fields mask	string
provider_id	path	The id of the provider	string

GET /providers/{provider_id}

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string
provider_id	path	The id of the provider	string

PATCH /providers/{provider_id}

Parameters

Name	Position	Description	Type
payload	body		
X-Fields	header	An optional fields mask	string
provider_id	path	The id of the provider	string

GET /providers/{provider_id}/check

Parameters

Name	Position	Description	Type
provider_id	path	The id of the provider	string

GET /providers/{provider_id}/user_actions

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string
provider_id	path	The id of the provider	string

GET /providers/{provider_id}/vm/flavours

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string
provider_id	path	The id of the provider	string

GET /providers/{provider_id}/vm/images

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string
provider_id	path	The id of the provider	string

GET /providers/{provider_id}/vm/params

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string
provider_id	path	The id of the provider	string

GET /providers/{provider_id}/vm_images

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string
provider_id	path	The id of the provider	string

GET /providers/{provider_id}/vm_sizes

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string
provider_id	path	The id of the provider	string

benchmarks

GET /benchmarks/

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string

GET /benchmarks/{benchmark_id}

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string
benchmark_id	path		string

users

POST /users/

Parameters

Name	Position	Description	Type
payload	body		
X-Fields	header	An optional fields mask	string

GET /users/

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string

DELETE /users/{user_id}

Parameters

Name	Position	Description	Type
user_id	path	The id of the user	string

PUT /users/{user_id}

Parameters

Name	Position	Description	Type
payload	body		
X-Fields	header	An optional fields mask	string
user_id	path	The id of the user	string

GET /users/{user_id}

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string
user_id	path	The id of the user	string

GET /users/{username}/scopes

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string
username	path	The id of the user	string

---

organizations

POST /organizations/

Parameters

Name	Position	Description	Type
payload	body		
X-Fields	header	An optional fields mask	string

GET /organizations/

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string

DELETE /organizations/{organization_id}

Parameters

Name	Position	Description	Type
organization_id	path	The id of the organization	string

PUT /organizations/{organization_id}

Parameters

Name	Position	Description	Type
payload	body		
X-Fields	header	An optional fields mask	string
organization_id	path	The id of the organization	string

GET /organizations/{organization_id}

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string
organization_id	path	The id of the organization	string

POST /organizations/{organization_id}/users

Parameters

Name	Position	Description	Type
payload	body		
X-Fields	header	An optional fields mask	string
organization_id	path	The id of the organization	string

GET /organizations/{organization_id}/users

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string
organization_id	path	The id of the organization	string

DELETE /organizations/{organization_id}/users/{user_id}

Parameters

Name	Position	Description	Type
organization_id	path	The id of the organization	string
user_id	path		string

workloads

POST /workloads/

Parameters

Name	Position	Description	Type
payload	body		
X-Fields	header	An optional fields mask	string

GET /workloads/

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string

GET /workloads/download

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string

GET /workloads/user_actions

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string

DELETE /workloads/{workload_id}

Parameters

Name	Position	Description	Type
workload_id	path	The id of the workload	string

PUT /workloads/{workload_id}

Parameters

Name	Position	Description	Type
payload	body		
X-Fields	header	An optional fields mask	string
workload_id	path	The id of the workload	string

GET /workloads/{workload_id}

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string
workload_id	path	The id of the workload	string

GET /workloads/{workload_id}/download

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string
workload_id	path	The id of the workload	string

GET /workloads/{workload_id}/user_actions

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string
workload_id	path	The id of the workload	string

schedules

POST /schedules/

Parameters

Name	Position	Description	Type
payload	body		
X-Fields	header	An optional fields mask	string

GET /schedules/

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string

GET /schedules/user_actions

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string

DELETE /schedules/{schedule_id}

Parameters

Name	Position	Description	Type
schedule_id	path	The id of the schedule	string

PUT /schedules/{schedule_id}

Parameters

Name	Position	Description	Type
payload	body		
X-Fields	header	An optional fields mask	string
schedule_id	path	The id of the schedule	string

GET /schedules/{schedule_id}

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string
schedule_id	path	The id of the schedule	string

GET /schedules/{schedule_id}/user_actions

Parameters

Name	Position	Description	Type
X-Fields	header	An optional fields mask	string
schedule_id	path	The id of the schedule	string

2.10 Docker

The Benchmarking Suite is also distributed in two different Docker containers. They are available at https://cloud.docker.com/app/benchsuite/repository/list.

2.10.1 benchsuite-multiexec

This container can be used to run benchmarks in batch mode.

Get (or update) the image with:

```
docker pull benchsuite/benchsuite-multiexec
```

Run the container binding the provider and storage (optional) configuration files stored in the local machine and passing the list of tests to execute as parameters (e.g. `idle:idle5`):

```
docker run -v /home/mypc/amazon.conf:/provider.conf -v /home/mypc/storage.conf:/
↪storage.conf benchsuite/benchsuite-multiexec:dev -p provider.conf -s centos_micro␣
↪idle:idle5
```

In case the storage service is running on the local machine, it could be necessary to use the `--net=host` option to reach it.

Alternatively, provider and storage configurations can be specified through environment variables: `BENCHSUITE_PROVIDER` and `BENCHSUITE_STORAGE_CONFIG` respectively.

```
docker run -e BENCHSUITE_PROVIDER="[myconf]...." -e BENCHSUITE_SERVICE_TYPE="centos_
↪micro" -v /home/mypc/storage.conf:/storage.conf benchsuite/benchsuite-multiexec:dev␣
↪idle:idle5
```

2.10.2 benchsuite-rest-service

This image contains the Benchmarking Suite REST SERVER (see rest-server-doc section). When started, the container exposes the REST service on port 5000.

To run the container, just use the Docker CLI:

```
docker run benchsuite/benchsuite-rest-server
```

The service reads the Benchmarking Suite bs-configuration from the `/` directory of the container. For instance, to provide a configuration for the storage (to persist results in the db) mount a file in the container named `/storage.conf` or `/storage.json`:

```
docker run -v my-storage.conf:/storage.conf benchsuite/benchsuite-rest-server
```

Also providers configuration files can be mounted in the container in the same way:

```
docker run -v my-provider.conf:/providers/my-provider.conf benchsuite/benchsuite-rest-
↪server
```

2.11 Scheduler

The Benchsuite Scheduler allows to schedule the execution of benchmarking tests at pre-fixed intervals. It needs:

- a MongoDB instance to load the **schedules** (see below), keep its state, log the executions and save the results
- a Docker Swarm instance to launch the tests (the *benchsuite-multiexec* Docker image is used) and to run the scheduler itself

The scheduler works in this way:

1. loads from a MongoDB collection the schedules and creates a job for each schedule (it uses APScheduler[1] under the hood). The jobs are kept in sync and refreshed periodically
2. sets-up a timer for each job accordingly with the time interval defined in the schedule
3. when its the time to execute a job, launchs a *benchsuite-multiexec* and configure it to execute the needed tests and to store the results on another MongoDB collection

2.11.1 Schedules

The **schedules** are the main input to the scheduler and models the tests that needs to be scheduled. Each schedule contains two types of information: the parameters of the tests and the timing information. Each schedule is expected to be a MongoDB document with this structure:

```
{
    "id" : "ec2-123asd-filebench",
    "active": true,
    "provider_config_secret" : "ec2",
    "username" : "ggiammat",
    "tests" : [
        "filebench",
        "ycsb-mysql",
        "dacapo"
    ],
    "tags" : [
        "scheduled"
    ],
    "properties" : {
        "prop1" : "val1",
        "prop2" : "prop2"
    },
    "env" : {
        "MYVAR": "val"
    },
    "interval" : {
```

(continues on next page)

[1] https://apscheduler.readthedocs.io/en/latest/

(continued from previous page)

```
        "hours" : 1
    },
    "benchsuite_additional_opts": ["-v", "--another-opt],
    "docker_additional_opts": {
        "hosts": {"myhost":"10.1.0.1"}
    }
}
```

It contains:

- `id`: a unique id
- `active`: defined whether this schedule should be considered by the scheduler or not
- `provider_config_secret`: the name (or the id) of the Docker secret that contains the Cloud Provider configuration. It uses the Docker secrets because the configuration also contains the user credentials to access the Cloud Provider
- `username`: an identifier of the user that is requesting the execution. It will be saved also in the benchmarking results
- `tests`: a list of test names to execute to be passed to the benchsuite `multiexec` command (see *Command Line Tool Documentation*)
- `tags`: a list of tags to assign to the results
- `properties`: a list of key-value properties that will be assigned to the results
- `env`: key-value pairs that define enviornment variables to be available in the execution environment during the execution
- `interval`: the time interval between two executions. The accepted keys are: `weeks`, `days`, `hours`, `minutes` and `seconds`. Multiple keys can be combined and if not specified, the default value is 0
- `benchsuite_additional_opts`: a list of string that will be appended to the benchsuite-multiexec command line
- `docker_additional_opts`: a dictionary of additional options to use when creating new Docker services (see DockerCreateServiceReference[2] for a reference of available options)

New schedules are automatically loaded and they are rescheduled if a change is detected.

2.11.2 Configuration

The schduler accepts multiple parameters. Some of them are mandatory, while some other have a default value.

All the parameters can be specified in a config file in the format

```
PARAM1=val1
PARAM2=val2
...
```

or specified as environment variable (the latter overrides the former).

The list of mandatory parameters are:

- `DB_HOST`: the connection string to the MongDB (e.g. "mongodb://localhost:27017"). It can be omitted only if the `SCHEDULES_DB_HOST`, `JOBS_DB_HOST` and `EXEC_DB_HOSTS` are provided

[2] https://docker-py.readthedocs.io/en/stable/services.html

- DOCKER_STORAGE_SECRET: the name of the secret that contains the Benchsuite Storage configuration (used to store results of the tests)

The optional parameters (or the ones that have a default value) are:

- SCHEDULES_SYNC_INTERVAL (default: 60): it the number of seconds between two refresh of the schedules in the MongoDB collection
- SCHEDULES_JOBS_PRINT_INTERVAL (default: 60): interval time in seconds to print on the console a report of the scheduled and running jobs
- DB_NAME (default: "benchmarking"): the name of the MongoDB database to use
- SCHEDULES_DB_HOST: if set, overrides the DB_HOST value for the MongoDB instance used to load the schedules
- SCHEDULES_DB_NAME: if set, overrides the DB_NAME value for the database used to load the schedules
- SCHEDULES_DB_COLLECTION (default: "scheduling"): the name of the collection that contains the schedules
- JOBS_DB_HOST: if set, overrides the DB_HOST value for the MongoDB instance used to store the internal state of the scheduler
- JOBS_DB_NAME: if set, overrides the DB_NAME value for the database used to store the internal state of the scheduler
- JOBS_DB_COLLECTION (default: "_apjobs"): the name of the collection that contains the internal state of the scheduler
- EXEC_DB_HOST: if set, overrides the ‘‘DB_HOST‘ value for the MongoDB instance used to log the executions
- EXEC_DB_NAME: if set, overrides the DB_NAME value for the database used to log the executions
- EXEC_DB_COLLECTION (default: "_apexec"): the name of the collection that contains the logs of the executions
- DOCKER_HOST (default: "localhost:2375"): the host and port of the Docker Swarm instance (used to create containers though the Docker API)
- DOCKER_BENCHSUITE_IMAGE (default: "benchsuite/benchsuite-multiexec"): the name of the benchsuite-multiexec image to use
- DOCKER_GLOBAL_ENV: a comma separated list of environment variables that will be set in the benchsuite-multiexec container (e.g. "VAR1=val1,var_2=val2"). Useful to set the an http proxy if necessary. Use ‘,’ to insert a comma in the variables names or values.
- BENCHSUITE_GLOBAL_TAGS: a comma separated list of string that will be set as tags in the benchmarking results (e.g. "test1,scheduled,automatic")
- DOCKER_ADDITIONAL_OPTS: a comma separated list of options in the format ‘KEY=VAL’ that will be added to the Docker service create invocation. VAL is evaluated using json.loads() function. See DockerCreateServiceReference[2] for a reference of available options (e.g. ‘hosts={"myhost":"10.1.0.1"}’)
- BENCHSUITE_ADDITIONAL_OPTS: additional options that will be set on the benchsuite-multiexec command line (e.g. "-vvv –failonerror")

2.11.3 Benchsuite Scheduler Docker image

The simplest way to run the Benchsuite Scheduler is to run the benchsuite/benchsuite-scheduler Docker image specifying the configuration parameters as envrionment variables:

```
docker run -e DB_HOST=mongodb://172.17.0.1:27017/ -e DOCKER_STORAGE_SECRET=storage -e␣
↪DOCKER_HOST=172.17.0.1:2375 benchsuite/benchsuite-scheduler
```

Alternatively, the configuration can be specified in the `/tmp/config` file.

```
docker run -v /home/mypc/scheduler.conf:/tmp/config benchsuite/benchsuite-scheduler
```

The two approaches can be also be mixed.

2.12 API Reference

class benchsuite.core.controller.**BenchmarkingController**(*config_folder=None*, *storage_config_file=None*)

The facade to all Benchmarking Suite operations

class benchsuite.core.model.benchmark.**Benchmark**(*tool_id*, *workload_id*, *tool_name*, *workload_name*, *workload_categories*, *workload_description*)

A Benchmark

2.13 Resources

- Articles:
  - 2018 - Benchmarking Suite Blog Post
  - 2018 - CloudPerfect White Paper

- Youtube:
  - https://www.youtube.com/watch?v=hKqBJWSv19c

CHAPTER 3

Contacts

Main contact person for the Benchmarking Suite is:

Person Gabriele Giammatteo

Company Research and Development Laboratory Engineering Ingegneria Informatica S.p.A.

Address Piazzale dell'Agricoltura, 24 00144 Rome, Italy

e-mail gabriele.giammatteo@eng.it

For bugs, features and other code-related requests the issue tracker can be used at: https://github.com/benchmarking-suite/benchsuite-core/issues

CHAPTER 4

References

Python Module Index

b

Index

B