reprobench

Rakha Kanz Kautsar

Jul 17, 2019

CONTENTS:

CHAPTER

ONE

REPROBENCH PACKAGE

1.1 Subpackages

1.1.1 reprobench.executors package

Submodules

reprobench.executors.base module

class reprobench.executors.base.**Executor**(**args*, ***kwargs*)
Bases: reprobench.core.base.Step

classmethod execute(*context*, *config=None*)

classmethod register(*config=None*)

run(*cmdline*, *out_path=None*, *err_path=None*, *input_str=None*, *directory=None*, ***kwargs*)

class reprobench.executors.base.**RunStatisticObserver**
Bases: reprobench.core.base.Observer

SUBSCRIBED_EVENTS = (b'executor:store_runstats',)

classmethod handle_event(*event_type*, *payload*, ***kwargs*)

reprobench.executors.db module

class reprobench.executors.db.**RunStatistic**(**args*, ***kwargs*)
Bases: reprobench.core.db.BaseModel

DoesNotExist
alias of RunStatisticDoesNotExist

MEMOUT = 'MEM'

OUTPUT_LIMIT = 'OLE'

RUNTIME_ERR = 'RTE'

SUCCESS = 'OK'

TIMEOUT = 'TLE'

VERDICT_CHOICES = (('TLE', 'Time Limit Exceeded'), ('MEM', 'Memory Limit Exceeded'), (

cpu_time = <FloatField: RunStatistic.cpu_time>

```
created_at = <DateTimeField:  RunStatistic.created_at>
max_memory = <FloatField:  RunStatistic.max_memory>
return_code = <IntegerField:  RunStatistic.return_code>
run = <ForeignKeyField:  RunStatistic.run>
run_id = <ForeignKeyField:  RunStatistic.run>
verdict = <CharField:  RunStatistic.verdict>
wall_time = <FloatField:  RunStatistic.wall_time>
```

reprobench.executors.events module

reprobench.executors.psmon module

class reprobench.executors.psmon.**PsmonExecutor**(*context*, *config*)

Bases: *reprobench.executors.base.Executor*

compile_stats(*stats*)

run(*cmdline*, *out_path=None*, *err_path=None*, *input_str=None*, *directory=None*, ***kwargs*)

Module contents

1.1.2 reprobench.managers package

Subpackages

reprobench.managers.local package

Submodules

reprobench.managers.local.manager module

Module contents

reprobench.managers.slurm package

Submodules

reprobench.managers.slurm.manager module

reprobench.managers.slurm.utils module

Module contents

Submodules

reprobench.managers.base module

Module contents

1.2 Submodules

1.3 reprobench.utils module

Various utilities

reprobench.utils.**check_valid_config_space**(*config_space*, *parameters*)
Check if the parameters is valid based on a configuration space

Parameters

- **config_space** (*ConfigSpace*) – configuration space
- **parameters** (*dict*) – parameters dictionary

Raises **ValueError** – If there is invalid values

reprobench.utils.**decode_message**(*msg*)
Decode an encoded object

This method deserialize the encoded object from *encode_message(obj)*.

Parameters **bin** – binary string of the encoded object

Returns decoded object

Return type obj

reprobench.utils.**download_file**(*url*, *dest*)
Download a file by the specified URL

Parameters

- **url** (*str*) – URL for the file to download
- **dest** (*str*) – Destination path for saving the file

reprobench.utils.**encode_message**(*obj*)
Encode an object for transport

This method serialize the object with msgpack for network transportation.

Parameters **obj** – serializable object

Returns binary string of the encoded object

Return type bin

reprobench.utils.**extract_archives**(*path*)
Extract archives based on its extension

Parameters **path** (*str*) – Path to the archive file

reprobench.utils.**extract_tar**(*path*, *dest*)
Extract a TAR file

Parameters

- **path** (*str*) – Path to TAR file
- **dest** (*str*) – Destination for extraction

reprobench.utils.**extract_zip**(*path*, *dest*)
Extract a ZIP file

Parameters

- **path** (*str*) – Path to ZIP file
- **dest** (*str*) – Destination for extraction

reprobench.utils.**find_executable**(*executable*)
Find an executable path from its name

Similar to */usr/bin/which*, this function find the path of an executable by its name, for example by finding it in the PATH environment variable.

Parameters **executable** (*str*) – The executable name

Returns Path of the executable

Return type str

Raises **ExecutableNotFoundError** – If no path for *executable* is found.

reprobench.utils.**get_db_path**(*output_dir*)
Get the database path from the given output directory

Parameters **output_dir** (*str*) – path to the output directory

Returns database path

Return type str

reprobench.utils.**get_pcs_parameter_range**(*parameter_str*, *is_categorical*)
Generate a range from specified pcs range notation

Parameters

- **parameter_str** (*str*) – specified pcs parameter
- **is_categorical** (*bool*) – is the range categorical

Raises **NotSupportedError** – If there is no function for resolving the range

Returns Generated range

Return type range

reprobench.utils.**import_class**(*path*)
Import a class by its path

Parameters **path** (*str*) – the path to the class, in similar notation as modules

Returns the specified class

Return type class

Examples

```
>>> import_class("reprobench.core.server.BenchmarkServer")
<class 'reprobench.core.server.BenchmarkServer'>
```

reprobench.utils.**init_db**(*db_path*)
Initialize the given database

Parameters **db_path** (*str*) – path to the database

reprobench.utils.**is_range_str**(*range_str*)
Check if a string is in range notation

Parameters **range_str** (*str*) – The string to check

Returns if the string is in range notation

Return type bool

Examples

```
>>> is_range_str("1..2")
True
>>> is_range_str("1..5..2")
True
>>> is_range_str("1")
False
```

reprobench.utils.**parse_pcs_parameters**(*lines*)
Parse parameters from a pcs file content

Parameters **lines** (*[str]*) – pcs file content

Returns generated parameters

Return type dict

reprobench.utils.**read_config**(*config_path*, *resolve_files=False*)
Read a YAML configuration from a path

Parameters

- **config_path** (*str*) – Configuration file path (YAML)
- **resolve_files** (*bool, optional*) – Should files be resolved to its content? Defaults to False.

Returns Configuration

Return type dict

reprobench.utils.**recv_event**(*socket*)
Receive published event for the observers

Parameters **socket** (*zmq.Socket*) – SUB socket for receiving the event

Returns Tuple for received events

Return type (event_type, payload, address)

reprobench.utils.**resolve_files_uri**(*root*)
Resolve all *file://* URIs in a dictionary to its content

Parameters **root** (*dict*) – Root dictionary of the configuration

Examples

```
>>> d = dict(test="file://./test.txt")
>>> resolve_files_uri(d)
>>> d
{'a': 'this is the content of test.txt\n'}
```

reprobench.utils.**send_event**(*socket*, *event_type*, *payload=None*)
Used in the worker with a DEALER socket to send events to the server.

Parameters

- **socket** (*zmq.Socket*) – the socket for sending the event
- **event_type** (*str*) – event type agreed between the parties
- **payload** (*any, optional*) – the payload for the event

reprobench.utils.**str_to_range**(*range_str*)
Generate range from a string with range notation

Parameters **range_str** (*str*) – The string with range notation

Returns The generated range

Return type range

Examples

```
>>> str_to_range("1..3")
range(1, 4)
>>> str_to_range("1..5..2")
range(1, 6, 2)
>>> [*str_to_range("1..3")]
[1, 2, 3]
```

1.4 Module contents

CHAPTER

TWO

TEST

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

r

INDEX

S

T

V

W