Gosbench User Guide

Table of Contents

Table of Contents **1**

Introduction **1**

Architecture **2**

Configuration **4**

S3 Configuration 7

Grafana Configuration 7

Test Configuration 8

Command Line Instructions **10**

Server 10

Driver 10

Output **10**

Logging 10

Performance Results 11

Capturing Time Series Data **11**

Introduction

Gosbench is a S3 object store performance testing tool, written in Golang, that is similar to the Java based Cosbench testing tool. Gosbench is written only for AWS S3 testing. Gosbench was originally written by members of the CEPH team to do S3 testing on CEPH. You read more about Gosbench from the following link: https://github.com/mulbc/gosbench

Architecture

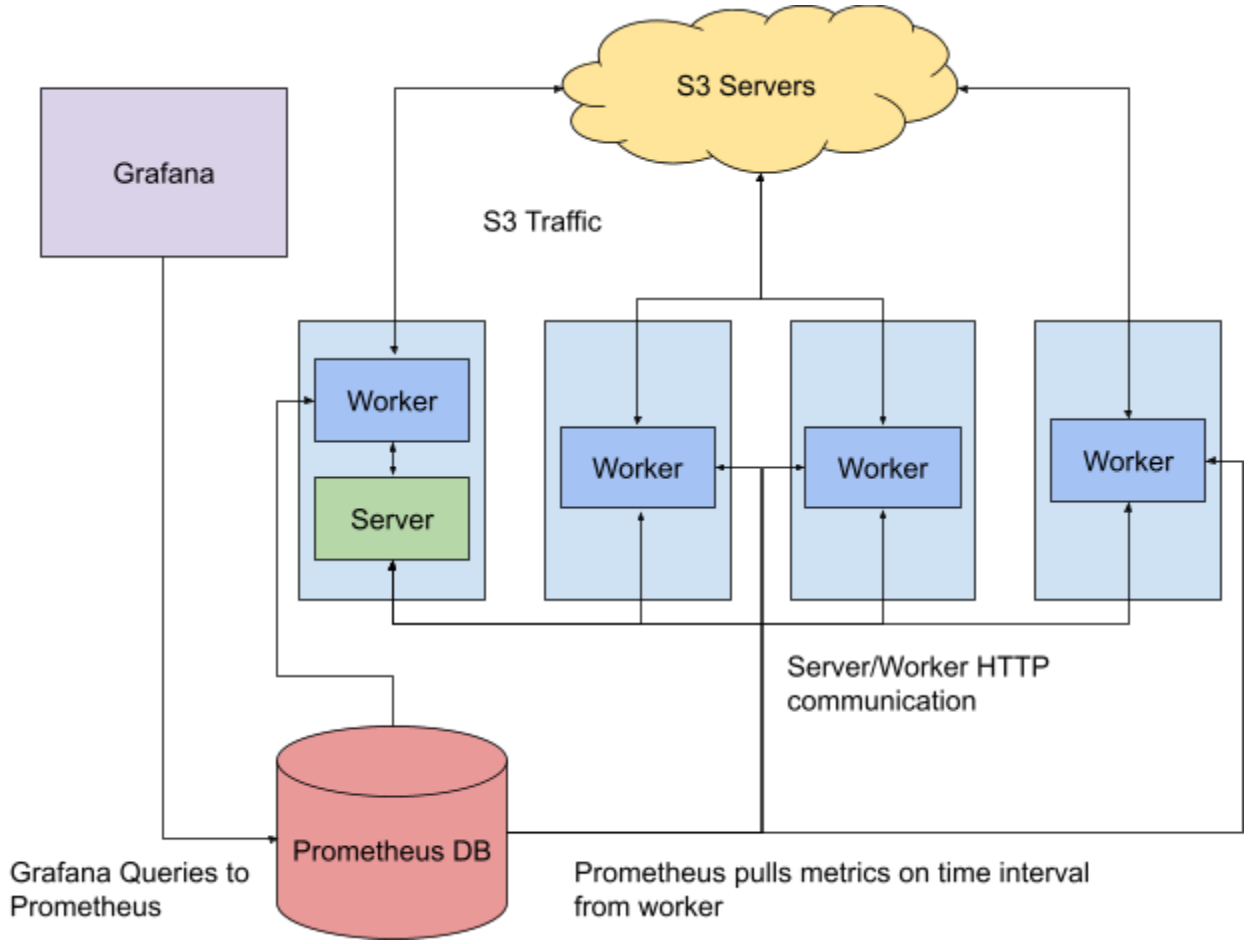
Gosbench is designed to run distributed tests against one or more S3 compliant storage servers. A basic Gosbench deployment consists of 3 files:

- Server - Coordinates Workers and general test queue
- Driver - Connects to S3 and perform reading, writing, deleting and listing of objects
- S3 Configuration File - Defines the S3 servers to be testing
- Workload Configuration File - Defines the tests to be run

To set up Gosbench you would first need to copy the server binary and configuration files to a host that has networking connectivity to the hosts that will run the drivers. The server and drivers both communicate with each other over TCP, so traffic must be open in both directions. By default the server will listen on port 2000 for responses from the driver. The listening port for the server can be changed as a command line option, when starting the server (See the Command Line Instructions section). Once the server is in place and your configuration files are complete, you can start the server process. The server will wait until the configured number of drivers connects to the server, before it starts any tests. Now that the server is set up and running, you can copy the driver to the hosts that will be used to make S3 requests to the S3 server or cluster. The driver can also be run on the same host as the server. Make sure you start the number of drivers that you specified in the configuration file, otherwise the test will not start. Once all the drivers are started and connected to the server, the server will initiate the tests for each driver.

Gosbench also supports the Prometheus time series database, for collecting metrics from the Gosbench drivers. Each driver will listen on a port for Prometheus scrape requests. By default the driver will listen on port 9995, but this default can be overwritten with command line options for the driver (See the Command Line Instructions section). These metrics can be viewed using Grafana dashboards. Gosbench also sends annotations to the Grafana server to annotate the start and stop of test jobs to help with viewing these jobs in the Grafana dashboards. The Prometheus and Grafana services are completely optional, and are not required to run or collect performance data. These tools can provide better visibility into what is happening on each driver node while the test is running.

Sample Gosbench Setup:



Configuration

The configuration files for gosbench can be formatted as a JSON file or as a YAML file. There are two configuration files needed to run a gosbench test. The first configuration file is the S3 configuration file. This file will specify the S3 endpoint, access credentials, and proxy configuration for one or more endpoints. The second configuration file is the workload configuration. This configuration file has 2 parts, the Grafana section and the tests section.

Example S3 JSON Configuration:

```
[
  {
    "access_key": "abc", "secret_key": "as",
    "region": "eu-central-1", "endpoint": "https://my.rgw.endpoint:8080",
    "skipSSLverify": false, "proxyHost": "http://localhost:1234"
  },
  {
    "access_key": "def", "secret_key": "as",
    "region": "eu-central-2", "endpoint": "https://my.rgw.endpoint:8080",
    "skipSSLverify": false, "proxyHost": "http://localhost:1234"
  },
  {
    "access_key": "ghi", "secret_key": "as",
    "region": "eu-central-3", "endpoint": "https://my.rgw.endpoint:8080",
    "skipSSLverify": false, "proxyHost": "http://localhost:1234"
  }
]
```

Example Test JSON Configuration:

```
{
    "grafana_config": { "endpoint": "http://grafana", "username": "admin",
"password": "grafana" },
    "tests": [
        { "name": "My first example test", "read_weight": 20,
"existing_read_weight": 0, "write_weight": 80, "delete_weight": 0,
"existing_delete_weight": 0, "list_weight": 0, "bucket_prefix":
"1255gosbench-", "object_prefix": "obj", "stop_with_runtime": "1h30m",
"stop_with_ops": 10,
          "drivers": 6, "workers_share_buckets": true, "workers": 30,
"clean_after": true,
          "objects": {"size_min": 5, "size_max": 100, "size_distribution":
"random", "unit": "KB",
                      "number_min": 10, "number_max": 10,
"number_distribution": "constant" },
          "buckets": { "number_min": 1, "number_max": 10,
"number_distribution": "constant" },
          "multipart": { "write_mpu_enabled": true, "write_part_size": 5,
"write_unit": "MB", "write_concurrency": 5,
                         "read_mpu_enabled": true, "read_part_size": 5,
"read_unit": "MB", "read_concurrency": 5 }
        }
    ]
}
```

Example S3 YAML configuration:

```
---

- access_key: abc
  secret_key: as
  region: eu-central-1
  endpoint: https://my.rgw.endpoint:8080
  skipSSLverify: false
  proxyHost: http://localhost:1234"
- access_key: def
  secret_key: as
  region: eu-central-2
  endpoint: https://my.rgw.endpoint:8080
  skipSSLverify: false
  proxyHost: http://localhost:1234"
- access_key: ghi
  secret_key: as
  region: eu-central-3
  endpoint: https://my.rgw.endpoint:8080
  skipSSLverify: false
  proxyHost: http://localhost:1234"

...
```

Example Test YAML configuration:

```
---

# For generating annotations when we start/stop test cases
# https://grafana.com/docs/http_api/annotations/#create-annotation
grafana_config:
  endpoint: http://grafana
  username: admin
  password: grafana

tests:
  - name: My first example test
    read_weight: 20
    existing_read_weight: 0
    write_weight: 80
    delete_weight: 0
    existing_delete_weight: 0
    list_weight: 0
    objects:
      size_min: 5
      size_max: 100
      # distribution: constant, random, sequential
      size_distribution: random
      unit: KB
      number_min: 10
      number_max: 10
      # distribution: constant, random, sequential
      number_distribution: constant
    buckets:
      number_min: 1
      number_max: 10
      # distribution: constant, random, sequential
      number_distribution: constant
    multipart:
      write_mpu_enabled: true
      write_part_size: 5
      write_unit: MB
      write_concurrency: 5
      read_mpu_enabled: true
      read_part_size: 5
      read_unit: MB
      read_concurrency: 5
    # Name prefix for buckets and objects
    bucket_prefix: 1255gosbench-
    object_prefix: obj
    # End after a set amount of time
    # Runtime in time.Duration - do not forget the unit please
    # stop_with_runtime: 60s # Example with 60 seconds runtime
```

```
    stop_with_runtime:
    # End after a set amount of operations (per driver)
    stop_with_ops: 10
    # Number of s3 performance test servers to run in parallel
    drivers: 2
    # Set whether drivers share the same buckets or not
    # If set to True - bucket names will have the driver # appended
    drivers_share_buckets: True
    # Number of requests processed in parallel by each driver
    workers: 3
    # Remove all generated buckets and its content after run
    clean_after: True

...
```

S3 Configuration

The S3 configuration section allows for a list of S3 servers to be configured for testing. If load balancer will be used, you may only need a single S3 configuration. If no load balancer is available you can specify multiple individual S3 servers, and Gosbench will assign the servers out evenly.

Configuration Options:

- **access_key** - Access key for S3 credentials
- **secret_key** - Secret key for S3 credentials
- **region** - Region to use for testing
- **endpoint** - The full HTTP(S) URL to use for S3 request. This URI should include a port if needed. Example: http://s3-region1.servicenow.cloudian.tme:80
- **skipSSLverify** - Should be set to true or false. True does not enforce strict validation of server certificate, false does enforce strict validation.
- **proxyHost** - The full HTTP(S) URL to use for proxy request. This URI should include a port if needed. Example: http://localhost:1234

Grafana Configuration

The Grafana configuration is used by Gosbench to send annotations to the Grafana DB when test jobs start and stop.

Configuration Options:

- **endpoint** - The full HTTP(S) URL to the Grafana server http://grafana:3000
- **username** - Grafana admin username
- **password** - Password for username

Test Configuration

The test configuration specifies the details of the test to be performed, including which operations to run, bucket/object names, object size, etc. The test configuration section has several top level parameters as well as parameters that contain subsections, such as “objects”, “buckets” and “multipart”.

Configuration Options:
Top Level Options:

- **name** - Name of the test
- **read_weight** - The priority to give to read requests
- **existing_read_weight** - The priority to give to existing_read requests
- **write_weight** - The priority to give to existing_read requests
- **delete_weight** - The priority to give to delete requests
- **existing_delete_weight** - The priority to give to existing_deleterequests
- **list_weight** - The priority to give to existing_read requests
- **bucket_prefix** - String to use as a prefix for bucket names
- **object_prefix** - String to use as a prefix for bucket names
- **stop_with_runtime** - If this option is set to any value greater than 0 the test will run for the specified amount of time, then it will stop. The “stop_with_runtime” takes precedence over the “stop_with_ops” parameter. If both are set, only the “stop_with_runtime” will be used. Be sure that a unit suffix is provided, such as “60s”, "300m", "1.5h" or "2h45m". Valid time units are "ns", "us" (or "µs"), "ms", "s", "m", "h".
- **stop_with_ops** - Specifies the number of operations to run before ending the test.
- **drivers** - The number of drivers that the server should expect to connect before starting the tests
- **workers** - The number of workers (or threads) that each driver should start up to run S3 commands
- **workers_share_buckets** - If true, all workers will use the same buckers to read, write, lisy, and delete objects from.
- **clean_after** - If true, Gosbench will delete all buckets and objects created during the test until number max is reached, then only number)max will be used.

Objects Options:

- **size_min** - Minimum size of object to use
- **size_max** - Maximum size object to use
- **size_distribution** - This parameter defines how object sizes are distributed. The valid values for this parameter are “constant”, “random”, “sequential”. If “constant” is set then only the size_min value is used for the object size. If “random” is set, then any value >= size_min and <= size_max may be used. If “sequential” is set the object size will start at size_min and the size will increment by 1 on each test.

- **unit** - The unit to use for size_min and size_max. Valid values are: B, K or KB, M or MB, G or GB, and T or TB. Either upper or lower case characters can be used.
- **number_min** - The minimum number value to use when generating a number suffix for object names.
- **number_max** - The maximum number value to use when generating a number suffix for object names.
- **number_distribution** - This parameter defines how object numbers are distributed. The valid values for this parameter are “constant”, “random”, “sequential”. If “constant” is set then only the number_min value is used for the object size. If “random” is set, then any value >= number_min and <= number_max may be used. If “sequential” is set the object size will start at number_min and the size will increment by 1 on each test until number max is reached, then only number)max will be used.

Buckets Options:

- **number_min** - The minimum number value to use when generating a number suffix for bucket names.
- **number_max** - The maximum number value to use when generating a number suffix for bucket names.
- **number_distribution** - This parameter defines how object numbers are distributed. The valid values for this parameter are “constant”, “random”, “sequential”. If “constant” is set then only the number_min value is used for the object size. If “random” is set, then any value >= number_min and <= number_max may be used. If “sequential” is set the object size will start at number_min and the size will increment by 1 on each test until number max is reached, then only number)max will be used.

Multipart Options:

- **write_mpu_enabled** - If true, this enables multipart writes using AWS’s upload manager. False, will use the putObject() function for uploading objects in a single request
- **write_part_size** - Specifies the size each part should be for multipart requests
- write_unit - The unit to use for write_part_size. Valid values are: B, K or KB, M or MB, G or GB, and T or TB. Either upper or lower case characters can be used.
- **write_concurrency** - The number of threads used by the upload manager to send parts simultaneously.
- **read_mpu_enabled** - If true, this enables multipart reads using AWS’s down manager. False, will use the getObject() function for downloading objects in a single request
- **read_part_size** - Specifies the size each part should be for multipart requests
- **read_unit** - The unit to use for read_part_size. Valid values are: B, K or KB, M or MB, G or GB, and T or TB. Either upper or lower case characters can be used.
- **read_concurrency** - The number of threads used by the download manager to receive parts simultaneously.

Command Line Instructions

Server

Starting the server:

```
gosserver -s <S3 Config File> -c <Test Config File> [-p <port number>] [-d] [-t]
```

Options:
-c Config file describing test run (Mandatory)
-p Port on which the server will be available for clients. Default is 2000
-d Enable debug log output
-t Enable trace log output

Driver

Starting the driver:

```
gosdriver -s <Address:Port to server> [-p <Prometheus port>] [-d] [-t]
```

Options:
-s Address to server. Example 192.168.1.1:2000 (Mandatory)
-p Port on which the worker will listen for Prometheus scrape requests. Default is 9995
-d Enable debug log output
-t Enable trace log output

Output

Logging

All logging (including debug and trace) for the server and the driver will be done to stdout and stderr. The server log output will include the performance results from each driver as well as a total of the results. The driver will log it's individual results.

Performance Results

The performance results for each test will be written in CSV format to “gosbench_results.csv”. If the file exists the results are appended to the current file. If the file does not exist then the file is created and results are appended. The fields written to the CSV file are:

- **TestName** - The name of the test
- **Operation Name** - The operations that were performed.
- **Workers** - The total number of workers across all drivers
- **Object Size** - Average object size
- **Completed Operations** - Successful operations
- **Failed Operations** - Failed Operations
- **Ops/Second** - Operations per second
- **Total Bytes** - Total bytes read/written
- **Bandwidth in Bytes/s** - Transfer rate in bytes per second
- **Average RT Latency in ms** - Average round trip latency in milliseconds. This is the average time to complete the operation
- **Average TTFB Latency in ms** - Average time to first byte latency in milliseconds. This is the average time to receive the first header byte
- **Success Ratio** - Success percentage
- **Start Time** - Time (Unix Time) the server started the tests on the drivers
- **Stop Time** - The time (Unix Time) when all drivers reported that their work was complete
- **Test duration seen by server in seconds** - The time it took for all driver to report finished
- **Test Options** - A list of options used during the test.

Capturing Time Series Data

Coming soon...