Automated Benchmarking of Container Applications

Paulius Dilkas

10th July 2019

1 Introduction

2 Architecture

Two simple Dockerfiles were created. The first one is used for TaskManager and JobManager pods and extends the original Flink Docker image by enabling Prometheus support on port 9250. Prometheus can then query that port in order to retrieve and record performance metrics.

The second Dockerfile extends the first one by adding a JAR file with Java code for both the control server and the Flink app. This image also contains a custom `ENTRYPOINT` that sends the Flink app to the JobManager (via port 8081) as a background process, while executing the control server in the foreground. This is the optimal arrangement of the two processes since the control server always waits for the Flink app to finish in order to take its running time into account when requesting data from Prometheus.

In both cases, one needs to change the permissions and group ownership of the `/opt/flink` directory (to `775` and `root`, respectively) so that the containers can be successfully executed by any user belonging to the group `root`. Both images were put on Docker Hub, as this was the simplest way to make them available to MiniShift.

A Docker Compose file was written with three services: Control, JobManager, and TaskManager, establishing open ports as pictured in Figure 2). This file was then converted to OpenShift manifestos using Kompose[1]. The generated manifestos, relevant network connections, and other dependencies are displayed in Figure 2.

Prometheus add-on for MiniShift[2] was modified to disable OAuth-based authentication by replacing `-skip-auth-regex=^/metrics` with `-skip-auth-regex=^/`. Furthermore, its configuration file was updated to set both scrape and evaluation intervals to 1 second and the list of targets to JobManager and TaskManager, both on port 9250.

Configuration Files component represents a ConfigMap created using the `oc` command that contains two configuration files, `global.yaml` and `components.yaml`. The former has the following...

3 Local Performance Tuning

3.1 Experiments

3.2 Adjusted Performance

4 Experimental Results

5 Conclusion

[1] `http://kompose.io/`

[2] `https://github.com/minishift/minishift-addons/tree/master/add-ons/prometheus`

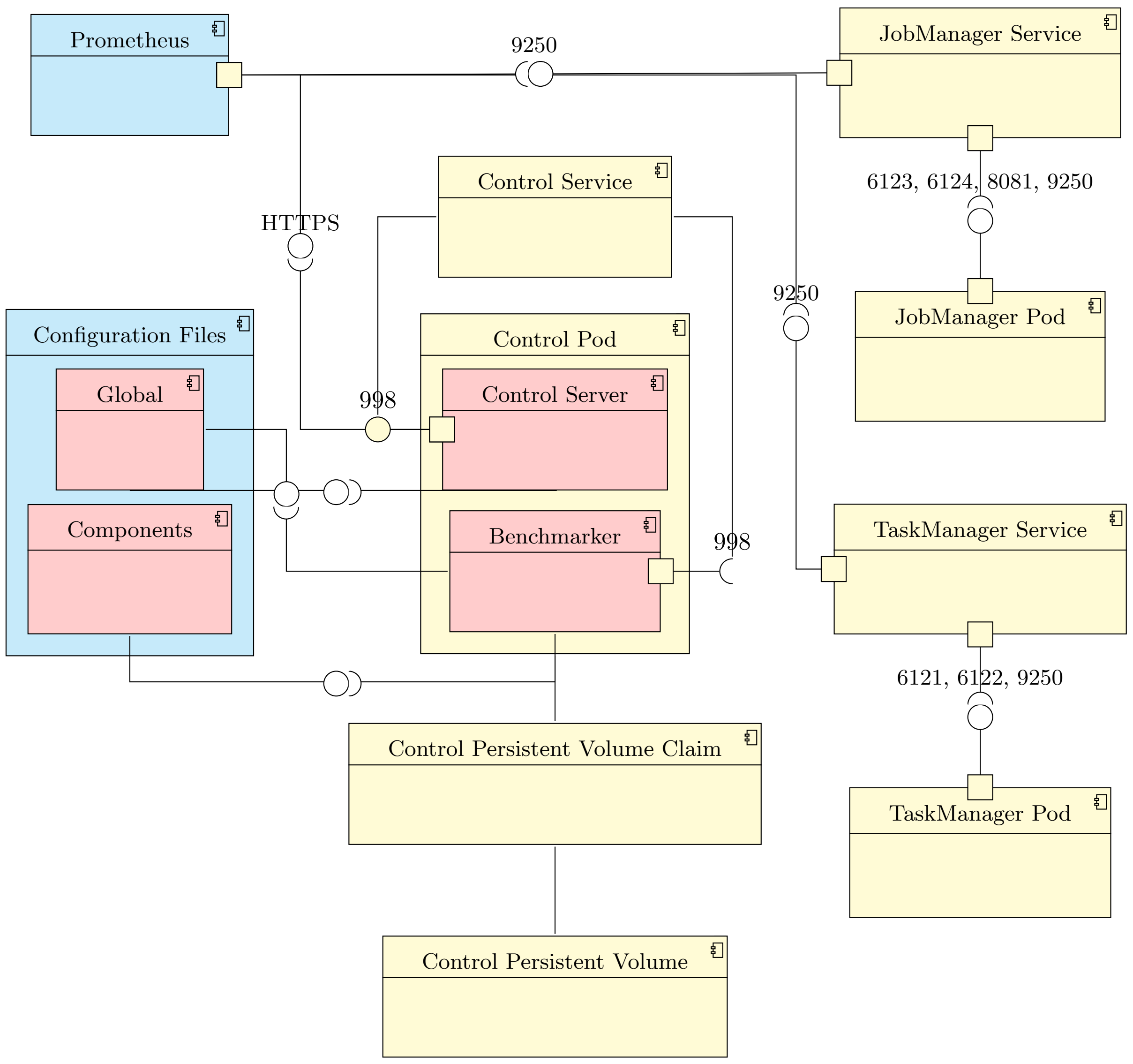

Figure 1: UML component diagram of the system, as deployed on MiniShift. Yellow components are OpenShift manifestos, while red components represent files (either Java classes or YAML configuration files). Network connections are shown with ports and have port numbers (or application-layer protocol names) displayed.

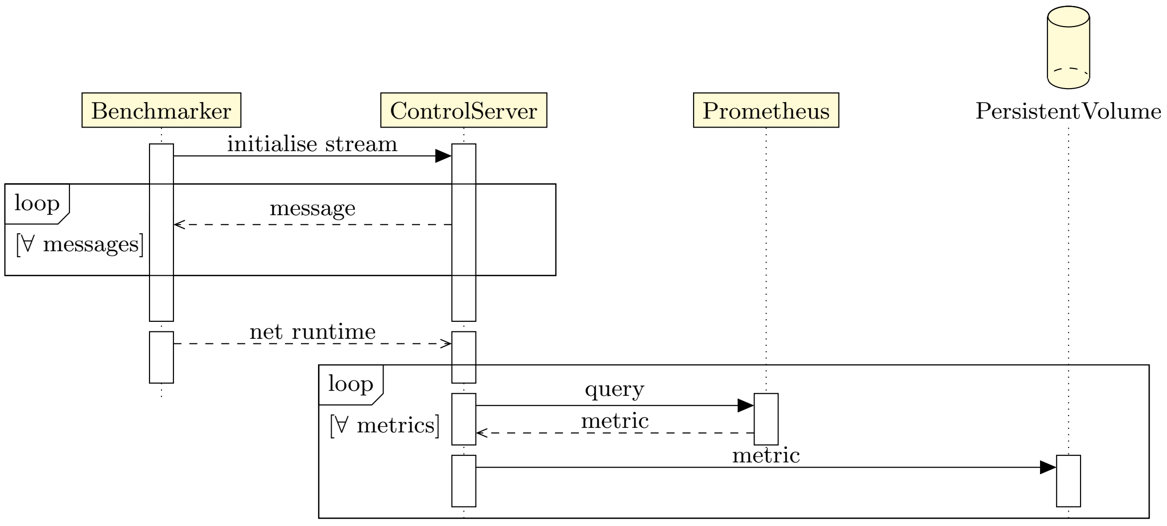

Figure 2: Communication between different parts of the system visualised as a UML sequence diagram

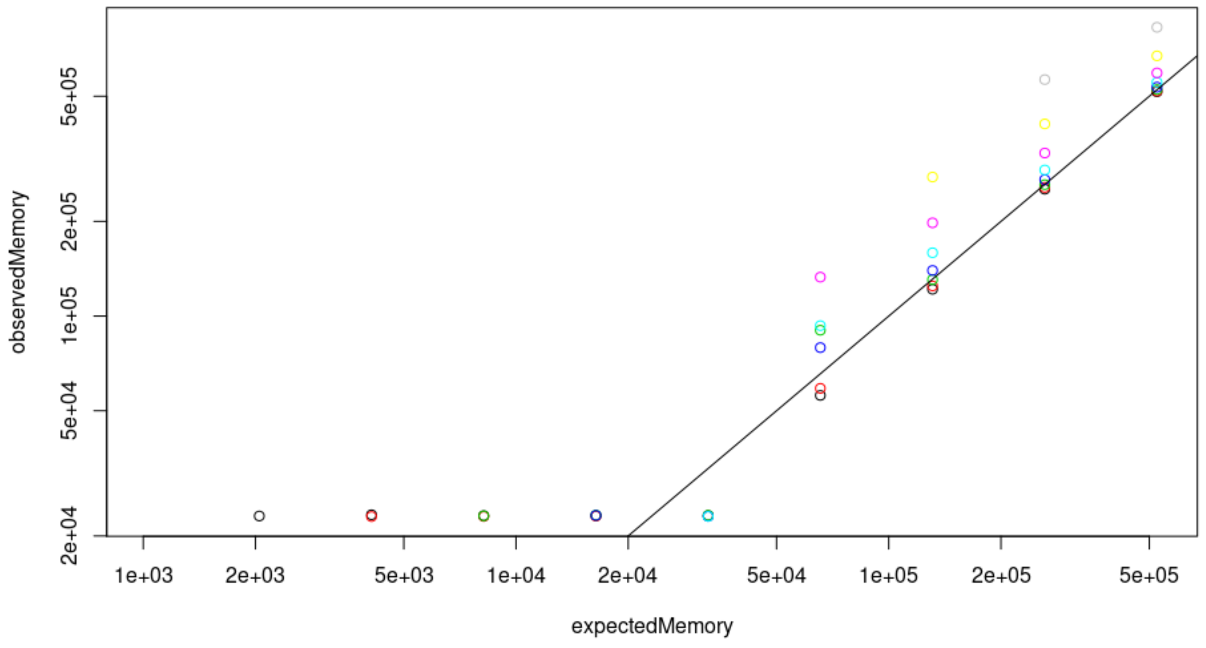

Figure 3:

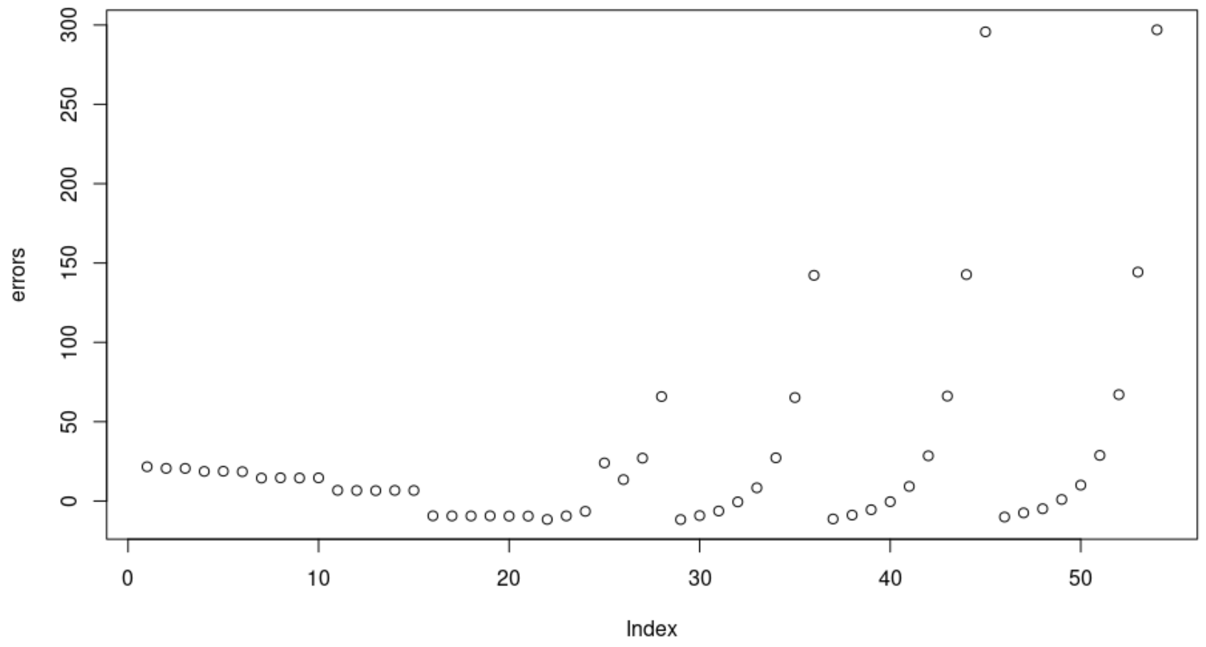

Figure 4:

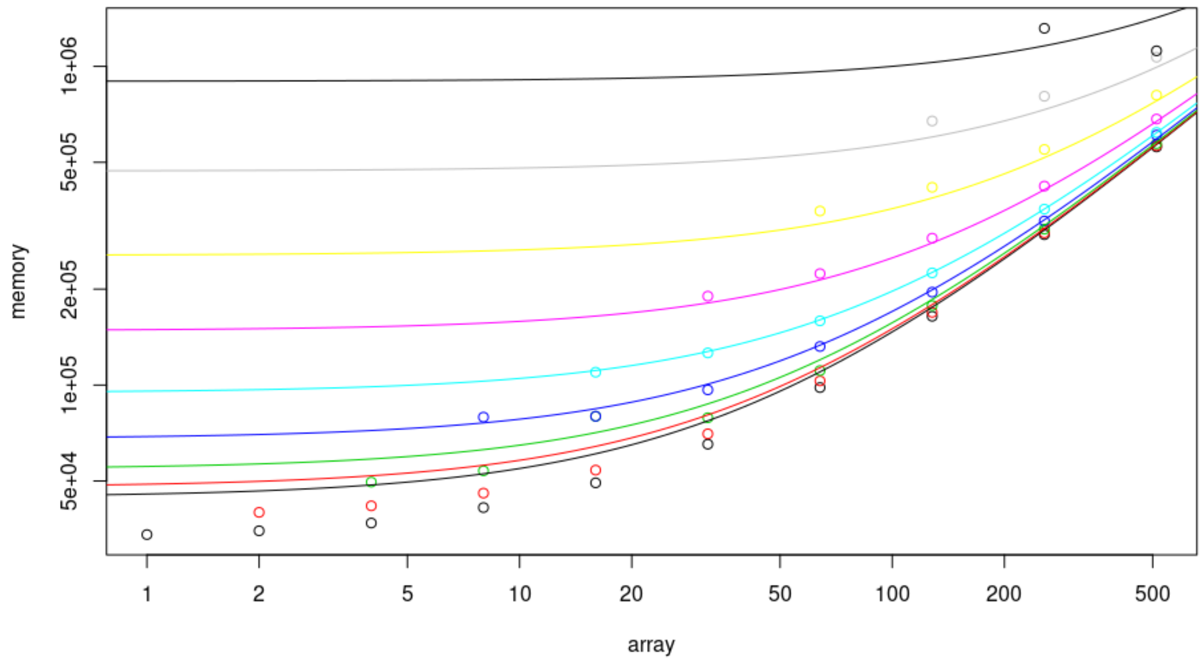

Figure 5:

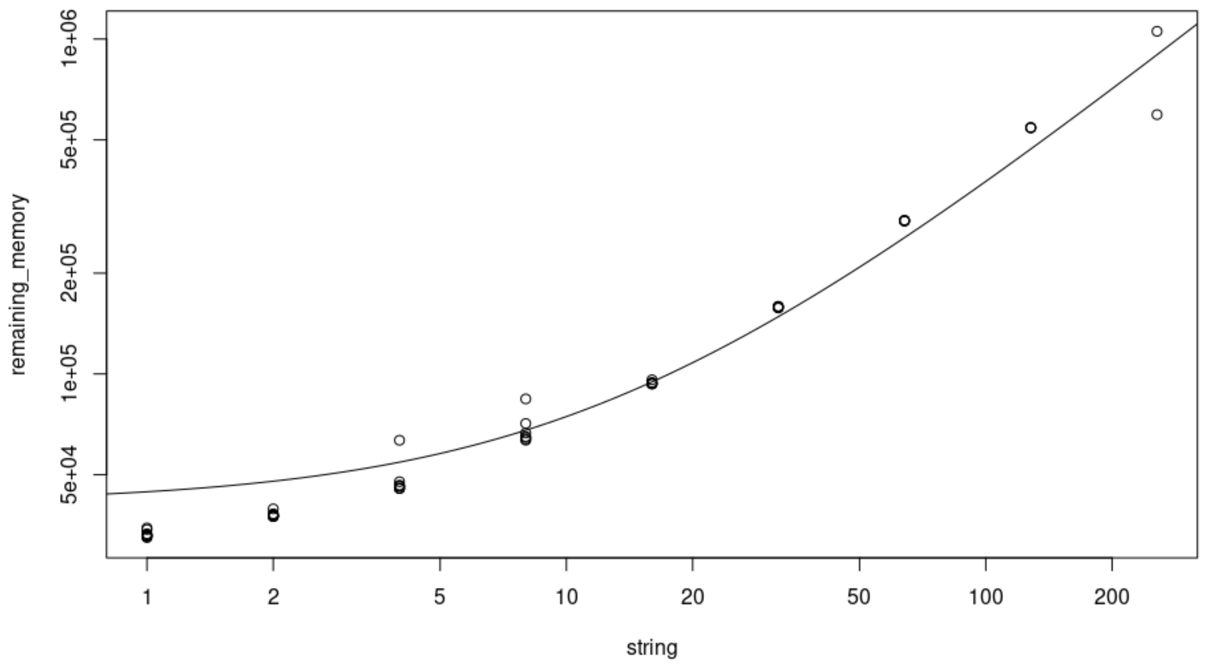
remaining_memory
1e+06
5e+05
2e+05
1e+05
5e+04
1
2
5
10
20
50
100
200
string

Figure 6: