Cloud Computing Applications and Services

University of Minho

Guide 5
Benchmarking

2024

The monitoring tools from the previous guide provide hints on your application's resource usage, which is useful to ensure you have the necessary initial resources for Swap execution. However, how can you be sure that these resources will be sufficient under increased loads (e.g., client requests)? Also, how can you ensure your application is ready to handle such increased loads?

The main goal of this guide is to use benchmarking tools (JMeter in this case), in conjunction with the monitoring tools from the previous guide, to better understand Swap's performance and resource usage under different workloads.

Recall these symbols from previous guides:

- [☁] Tasks to be performed by the *cloud provider*;
- [</>] Tasks to be performed by the *application developer*;

Apache JMeter™

JMeter is an open-source Java application designed to conduct performance, functional, and load testing of web applications.
JMeter is available at: `https://jmeter.apache.org`

1 Setup [☁]

1. Deploy the setup used in *Guide5: Monitoring*. You should use the VMs from the previous class.
   - K8s Setup

     ```
     vagrant up monitor myvm controlplane node1 node2
     ```

   - Docker Setup

     ```
     vagrant up monitor myvm node1
     ```

   Note: Make sure to boot the *monitor* VM in the first place to ensure Elasticsearch and Kibana are ready when Metricbeat (from other VMs) tries to connect to them.
2. Ensure you can access the Swap application and the Kibana dashboards. It may take a couple of minutes for Metricbeat to start sending data to Kibana.

2 Tasks [</>]

2.1 JMeter Installation

1. Install Java at your computer: `https://www.java.com/en/`
2. Download JMeter Binaries (Binaries - version 5.6.3):
   `https://dlcdn.apache.org//jmeter/binaries/apache-jmeter-5.6.3.tgz`

2.2 JMeter Configuration

1. Run JMeter in GUI mode (`./bin/jmeter`) at your computer.
2. Now let us create the testing workload:
   (a) Add a "*Thread Group*" to your "*Test Plan*".
   At the group, we will define the behavior of benchmark clients.
   (b) Add "*Constant Timer*" to the "*Thread Group*".
   The timer allows defining the delay between requests of a given client ("*Thread Delay*").
   Set this value to 300 ms.
   (c) Add an "*HTTP Request*" sampler to the "*Thread Group*".
   The sampler defines the requests being done by clients.
   Use an HTTP GET request type to the Swap login page ("*Path*": `/login`).
   The IP address and port number should be the ones exposed by your Swap setup.
   (d) Add "*Summary Report*" and "*Graph Results*" listeners to the "*HTTP Request*" sampler.
   These Listeners provide visual representations for the results being observed.
   Note: Listener results can be exported to a file.

2.3 Swap Testing

1. At the "*Thread Group*", set the number of threads to 1 and the loop count to infinite (*i.e.*, 1 benchmark client doing requests infinitely).
   Run the experiment and observe Kibana's Dashboard "*[Metricbeat System] Host overview ECS*" and JMeter's "*Summary Report*" and "*Graph Results Listeners*" to obtain different metrics.
   Note: Make sure the test is running correctly and there are no errors!
2. Stop the test and change the number of threads to 10.
   Re-run the experiment and observe the results at JMeter and Kibana.
3. Stop the test and change the number of threads to 100.
   Re-run the experiment and observe the results at JMeter and Kibana.
4. Export the template to a file and run the experiment with the Non-GUI mode.
   `./bin/jmeter -n -t [jmx file] -l [results file]`

   Important: Always run experiments for collecting measurements with the Non-GUI mode!

Extra

1. Create an Ansible Playbook to automate the execution of JMeter tests:
   - JMeter can be run on remote servers (e.g., *myvm*). Just make sure not to run JMeter on the same server that is being tested to avoid influencing the results.
   - Do not forget to automate all the steps necessary to run JMeter, including its installation.
   - Explore how you could use Ansible's template module and the *jmx* file generated in 2.2 to provide the different test configurations from 2.3 on your playbook.
2. Explore JMeter browser recording capabilities:
   `https://jmeter.apache.org/usermanual/jmeter_proxy_step_by_step.html`

Learning Outcomes

Recognize the goals and considerations that must be taken when doing a performance evaluation of real systems. Apply the JMeter benchmark to evaluate a web-based service. Complement benchmarking analysis with monitoring.