# Cloud Computing

**Winter Term 2020/2021**

*Practical Assignment No. 3*

**Due: 20.12.2020 23:55**

The primary goal of this assignment is to gain insight into performance characteristics of virtualization by benchmarking different virtualization and isolation techniques and comparing and evaluating the benchmark results. The secondary goal is to gain an understanding of isolation features provided by modern Linux kernels.

# 1. Prepare Virtualization

Prepare a Unix experiment host. It can be a physical machine you own (e.g. your Laptop) or a virtual machine on one of the public clouds. If you use the public cloud, use a medium sized VM. Run all experiments on the same underlying host.

Install QEMU (and the KVM kernel module) on your experiment host. Study the QEMU documentation on how to work with disk images. Create or obtain a VM image in the `qcow2` format with Ubuntu 18.04 installed. You can either manually install Ubuntu, or obtain an Ubuntu cloud image. When working with a cloud image, you have to configure cloud-init (see for example this guide and these examples). The guest OS needs to be able to access the internet and you need SSH access into the guest to exchange files and run commands. You are free to use QEMU manually or use a management tool such as Libvirt. Give the VMs access to all CPU cores of the experiment host and ensure sufficient memory.

Install Docker on your experiment host. Study the Docker documentation and get to know the Docker command line tool.

Write an `setup.txt` file containing basic information about your setup. What experiment host do you use? What are the versions of all relevant software? How did you prepare/obtain your VM image? Also add info about the CPU, RAM and disk of your experiment host.

**Outputs:**

- File **`setup.txt`**
  - Describing your system (see above)

# 2. Prepare Performance Benchmarks

Extend the `benchmark.sh` from the first assignment. You are free to use the solution script uploaded to ISIS as a basis. The script takes no parameters and performs the 4 benchmarks from the first assignment (CPU, RAM, sequential and random access to disk), plus the additional benchmarks described below. The script outputs the results as a single CSV line, such as the following:

**`1575076473,877.37,470446.80,308.78,116.17,7572.76,7.78`**

As in the first assignment, every benchmark should run for 60 seconds. The first four benchmarks have exactly the same requirements as before. The requirements for the two new benchmarks are as follows.

**Fork Benchmark**

This benchmark will measure how fast new processes can be spawned. In addition to the system resources, this measures overhead introduced through system calls in the Linux kernel.

Finish the **`forkbench.c`** program provided on ISIS. The program performs a parallel computation of an integer range sum. `forkbench.c` receives a start and end value of an integer range and computes the sum of all numbers in that range. The range is split in two halves and two child processes are forked to compute the respective sub-problem, until the recursion ends when start and end of the range are the same. In addition to the sum, the program counts the number of executed fork operations, measures the total execution time, and outputs the number of forks per second to stdout. Your finished program must use the `fork` and `pipe` system calls. Other useful functions include fdopen and getline. Further details and hints can be found in the comments of the provided forkbench.c file and in the assignment slides. Make sure that forkbench.c can be compiled on your test systems through the following command (install required packages):

```
$ make forkbench
```

In this simple case `make` does not need a `Makefile`. After finishing and testing `forkbench.c`, find a number range that runs without errors on all benchmark targets and takes at least 1-2 seconds. Extend the `benchmark.sh` script:

- Compile `forkbench` with the command given above.
- Run forkbench with the chosen input values. Repeat the program until 60 seconds have passed and record the standard output of each execution. The output is the number of forks performed per second.
- Compute an average forks-per-second value of all executions of forkbench, and output that value in the end of the script, as part of the resulting CSV line. Hint: the bc program can do floating point calculations in shell scripts.

**Network Benchmark**

Use iperf3 to benchmark the network upload speed in Mbit/s. Use IPv4 and 5 connections in parallel. Use an iperf3 server hosted on the experiment host. Don't forget to limit the runtime, and at the end to extract the required value from the output.

**Outputs:**

- Finished program **`forkbench.c`**
- Extended script **`benchmark.sh`**
    - Contains new benchmarks based on forkbench and iperf3

# 3. Execute Performance Benchmarks

The final step is to execute the benchmark script 10 times on the following platforms:

- Native on your host
- In a Docker container
- Virtualization with hardware support (Qemu with KVM)
- Virtualization with dynamic binary translation (Qemu)

**Preparation:**

1. Write a `Dockerfile` that builds a container image that is able to execute your benchmark script without any further files or other changes. When starting, the container should automatically execute the benchmark script, output the results, and exit.
2. Make sure you have the two VMs (KVM and Qemu) running idle in the background, ready to accept SSH connections (no need to document how you prepare and start the VMs).
3. Write a script **`exec_benchmark.sh`** for the following tasks:
    - Prepare the 4 result files by writing a CSV header into them (file names and CSV header see below)
    - Run the benchmark on the native system 10 times and collect the results
    - Build the benchmarking Docker container image
    - Run the Docker image 10 times and collect the results
        - Hint: do not use the --tty/-t option, since it merges the stdout and stderr streams
    - Copy the required files via scp to your KVM VM
    - Run the KVM benchmark 10 times via ssh and collect the results
    - Copy the required files via scp to your Qemu VM
    - Run the Qemu benchmark 10 times via ssh and collect the results
        - For the ssh/scp commands, simply hardcode the IP addresses and other information. Delete private or sensitive parts before submitting.

Example of a resulting CSV file:

```
time,cpu,mem,diskRand,diskSeq,fork,uplink
1575075020,878.48,468847.55,302.73,143.00,7363.26,7.84
1575075386,876.73,470509.97,307.32,114.33,7503.29,7.80
1575075748,877.57,470365.19,308.76,116.44,7591.76,7.89
...
```

Execute `exec_benchmark.sh`. This should take a few hours (4 platforms * 7 benchmarks * 10 repetitions * 1 minute), make sure to have no background tasks running on your experiment host during that time. Afterwards, plot the results with the provided `plot.py` Python script. Answer the questions below in a well-formatted, readable text. Copy the text with your answers into the submission text field on ISIS.

**Outputs:**

- **`Dockerfile`**
- Script **`exec_benchmark.sh`**
- 4 CSV-files:
  - **`[native|docker|kvm|qemu]-results.csv`**
- 6 plots generated by `plot.py`:
  - **`[cpu|mem|diskRand|diskSeq|fork|uplink]-results.png`**
- Text with answers to questions
  - Paste the text into the submission text field on ISIS
  - Please use max. 200 words per question

**Benchmark questions:**

1. Look at your benchmark results. Are they consistent with your expectations, regarding the different virtualization platforms? Explain your answer. What are the main reasons for the differences between the platforms? Answer these questions for all benchmarks:
   a. CPU
   b. Memory
   c. Random disk access
   d. Sequential disk access
   e. Fork
   f. iperf uplink

2. Choose the guest system with the highest uplink deviation compared to the native execution and rerun the iperf3 benchmark using a public iperf3 server. Run the same experiment using the host as a client (and the same public iperf3 server). Could the results from Task 3 be reproduced? Why? Why not?

# 5. Submission Deliverables

Submit your solution on the ISIS platform as individual files. Please submit ONLY the necessary files and use exactly the given file names!
Submit the text with answers to the questions directly in the submission text field.

Expected submission files:

- `setup.txt`
  - **4 points**
- `forkbench.c`
  - **7 points**
- `benchmark.sh`
  - **10 points**
- `Dockerfile`
  - **10 points**
- `exec_benchmarks.sh`
  - **10 points**
- `[native|docker|kvm|qemu]-results.csv`
  - **4 CSV files, 2 points each**
- `[cpu|mem|diskRand|diskSeq|fork|uplink]-plot.png`
  - **6 image files, 1 point each**
- Text with answers to questions
  - **15 points**

*Total points: 70*