Cloud Computing

Winter Term 2018/2019

Tutorial Session 2

Anton Gulenko
Complex and Distributed IT-Systems

anton.gulenko@tu-berlin.de

Practical Assignment 2

- Due: 20.12.2018
- Summary:
  - Work on 1 host machine
    - Preferably your laptop (physical machine). If you don’t have Linux, use a VM.
  - Prepare virtualization environments:
    - Qemu/KVM
    - Docker
  - Write 2 new benchmarks:
    - Forksum
    - Nginx
  - Execute benchmarks on different virtualization platforms

Practical Assignment 2

- Public cloud platforms are not mandatory for this assignment, but do not yet delete your accounts (shut down your VMs if not used for the assignment)

- We will continue using OpenSubmit, **BUT**:
  - Only for validating that all assignment files are there
  - If your submission fails the test, we will still grade it, but very likely your points will be reduced

Virtualization Platforms

- Docker
  - Write one Dockerfile for every benchmark
  - The containers should contain all material for the respective benchmark, execute the benchmark when started without parameters, and exit after printing the results
  - Exception: Nginx benchmark, since it is executed outside the container
- Qemu (with and without KVM)
  - Use a cloud image of any Debian based Linux distribution (e.g. Ubuntu)
  - Load all benchmark material in one image, it will not be submitted

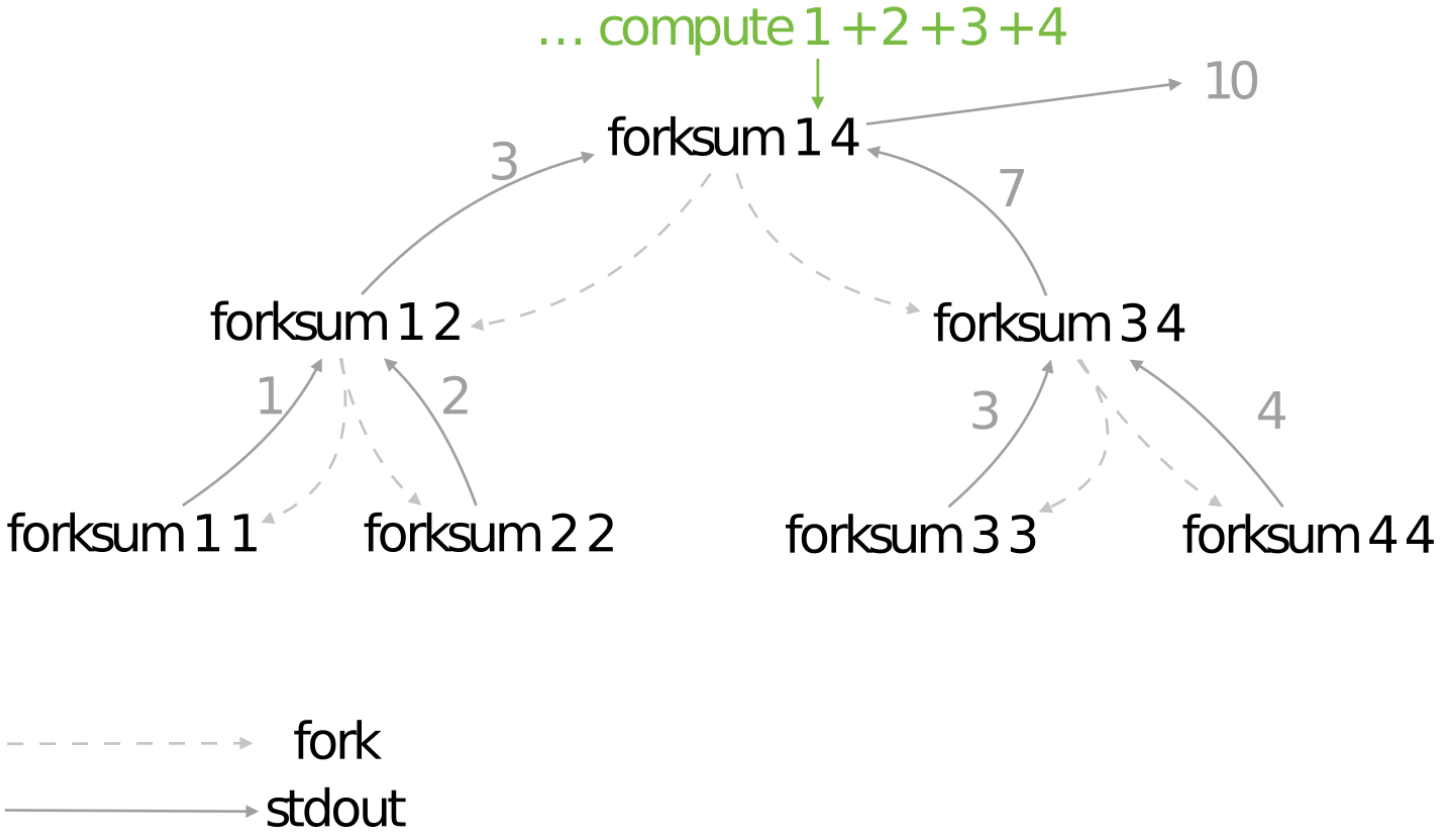
Benchmarks

- Basic resources: CPU, RAM and disk access
  - Reuse from assignment 1
  - Exception: For disk benchmark, only use fio to measure write accesses per second
- New benchmark: forksum
  - Simulates creation of many parallel processes
  - Main benchmark target: system calls
- New benchmark: Nginx web server
  - Download big file over the network
  - Main benchmark target: network performance

Forksum

- Program received 2 parameters: start and end of integer range
  - Task: compute sum of all integer within the range
  - Example: **`./forksum 100 1000`** should print **`495550`**
- Every sum is executed by a separate child process
  - If start == end: output value
  - Else: spawn 2 child processes: one for lower sub-range and one for upper sub-range
    - After child-processes return their results, parse them and output the sum

Forksum: Example



Forksum: required C system calls

- **`fork()`**: continue program as two separate processes
  - Return value of `fork()` tells the program if it is the child or parent process
- **`pipe()`**: create a pipe that can be used to read the standard output of a child process
- **`read()`**: can be used to read from a pipe
- **`dup()`**/**`dup2()`**: Used to assign the standard output of a child to a pipe
- **`wait()`**: Wait for child processes to exit
- Other useful functions: strotl(), printf(), fprintf(), close(), exit(),
- Example for how to read output from a child process:
  - http://www.microhowto.info/howto/capture_the_output_of_a_child_process_in_c.html
- Hint: the **`execl()`** system call is not necessary

Nginx Benchmark

- Install Nginx web server
- Configure Nginx to serve static files from the disk
- Add a file (> 500MB) to the Nginx server
- Your benchmark script should receive an IP address as parameter and generate requests to the file (at least 2 requests at the same time)
- After ~10 seconds, the average access duration should be printed
- You can write a benchmarking client in any programming language or use any existing HTTP load generation tool