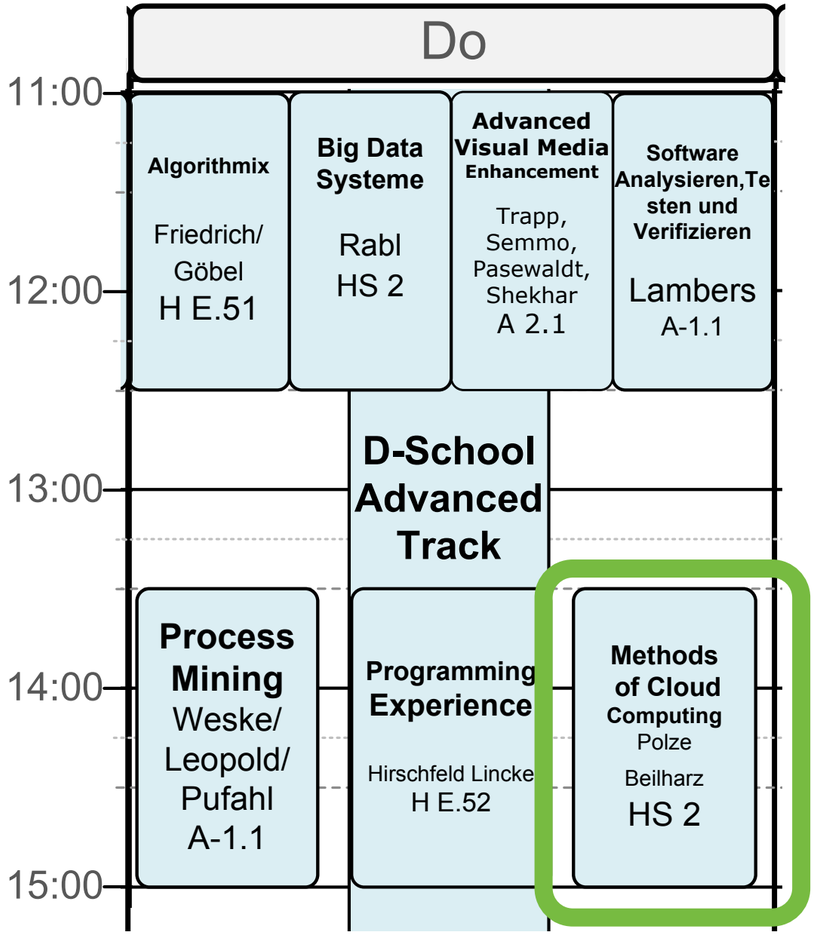
MoCC

Assignment 2

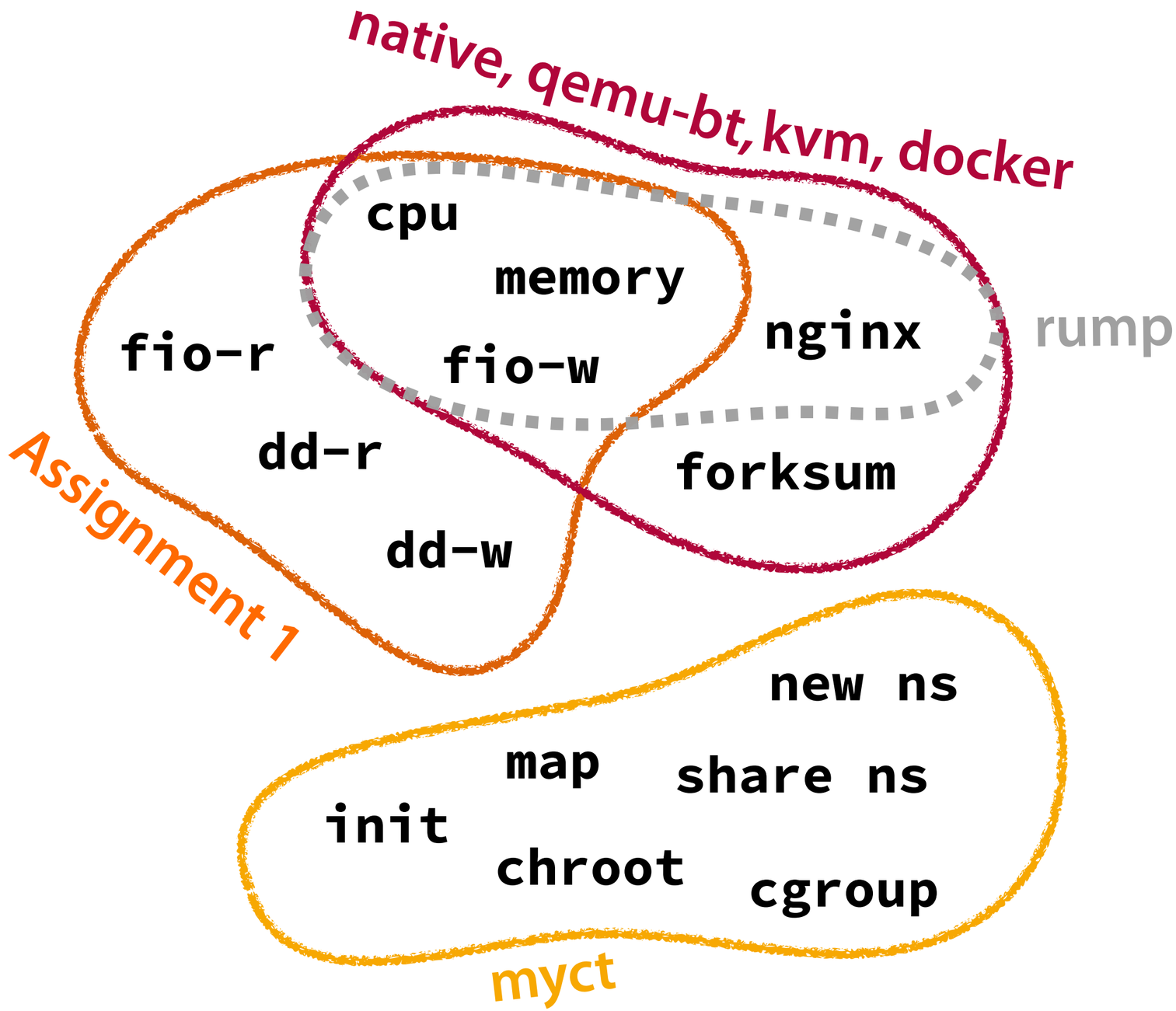
Sven Köhler
Hasso Plattner Institute
2018-12-06



> 50 % of points per assignment for exam submission

Organization

1


native, qemu-bt, kvm, docker
cpu
memory
fio-r
fio-w
nginx
rump
dd-r
forksum
dd-w
Assignment 1
new ns
map
share ns
init
chroot
cgroup
myct

Tasks

Run all experiments on the **same underlying host** hardware!
Document your hardware features and software versions.

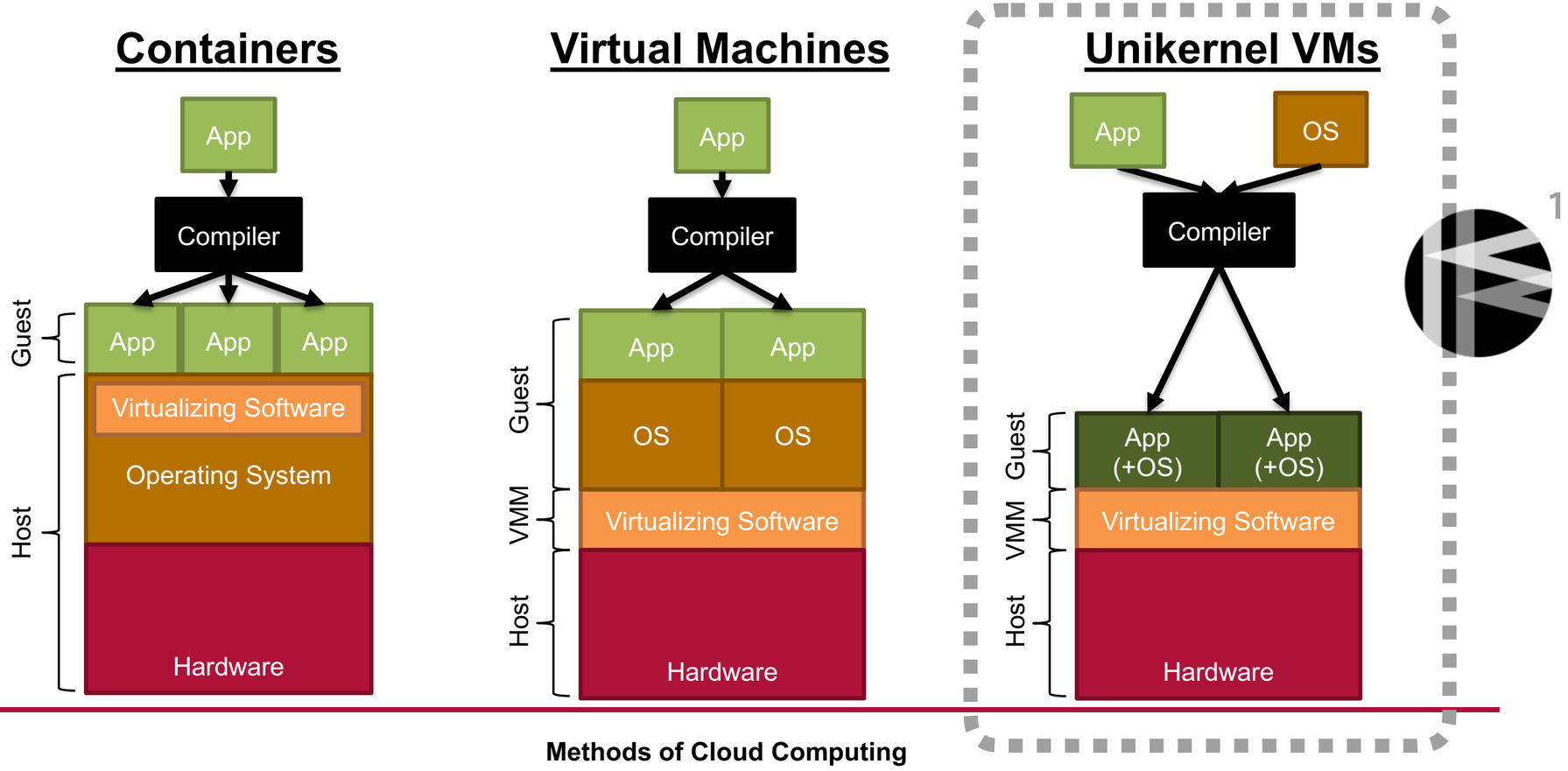
Four (+1) platforms:

- Your Linux laptop/workstation/…
- A virtual machine with binary translation (QEMU)
- A virtual machine with hardware support (QEMU-KVM)
- A container (Docker)
- A virtual machine without guest OS (Rumpkernel)

optional (+10 %)

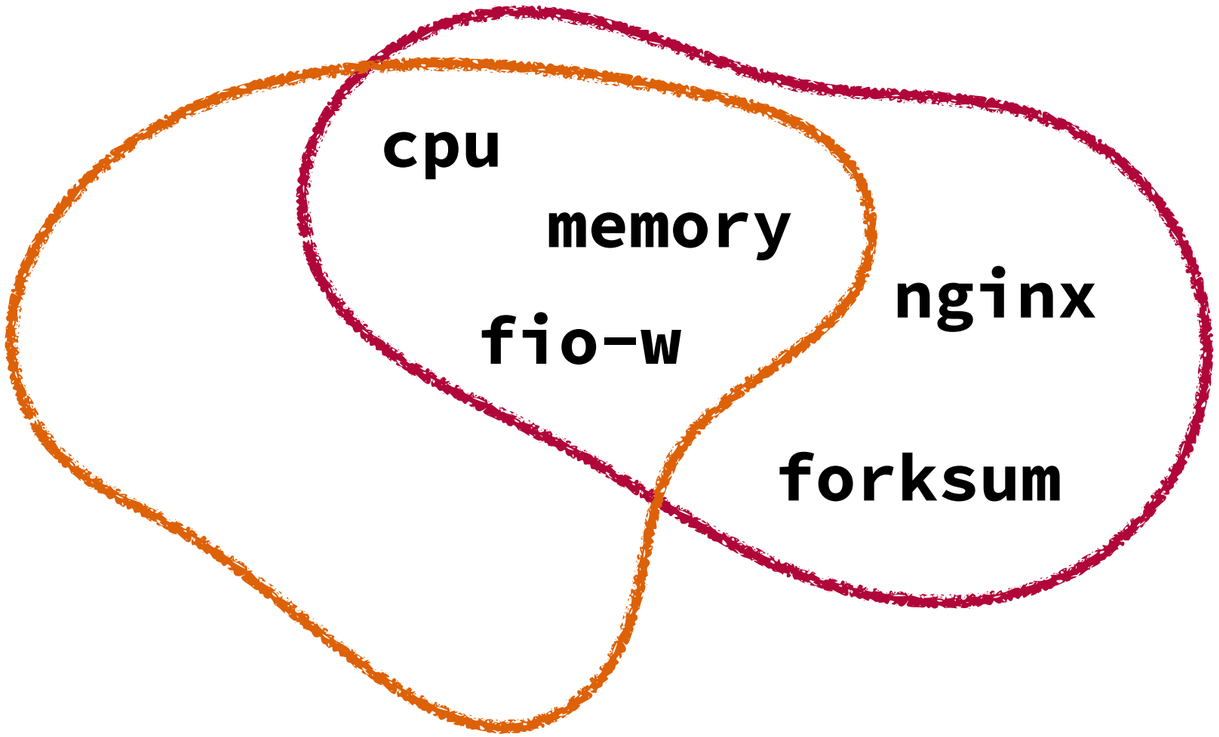
Sidenote: Unikernel VM Images

- Compile and link application code directly with all required OS functionality



Methods of Cloud Computing

[1] Kantee, A., & Cormack, J. (2014). *Rump Kernels: No OS? No Problem!.* ; login: The USENIX magazine.

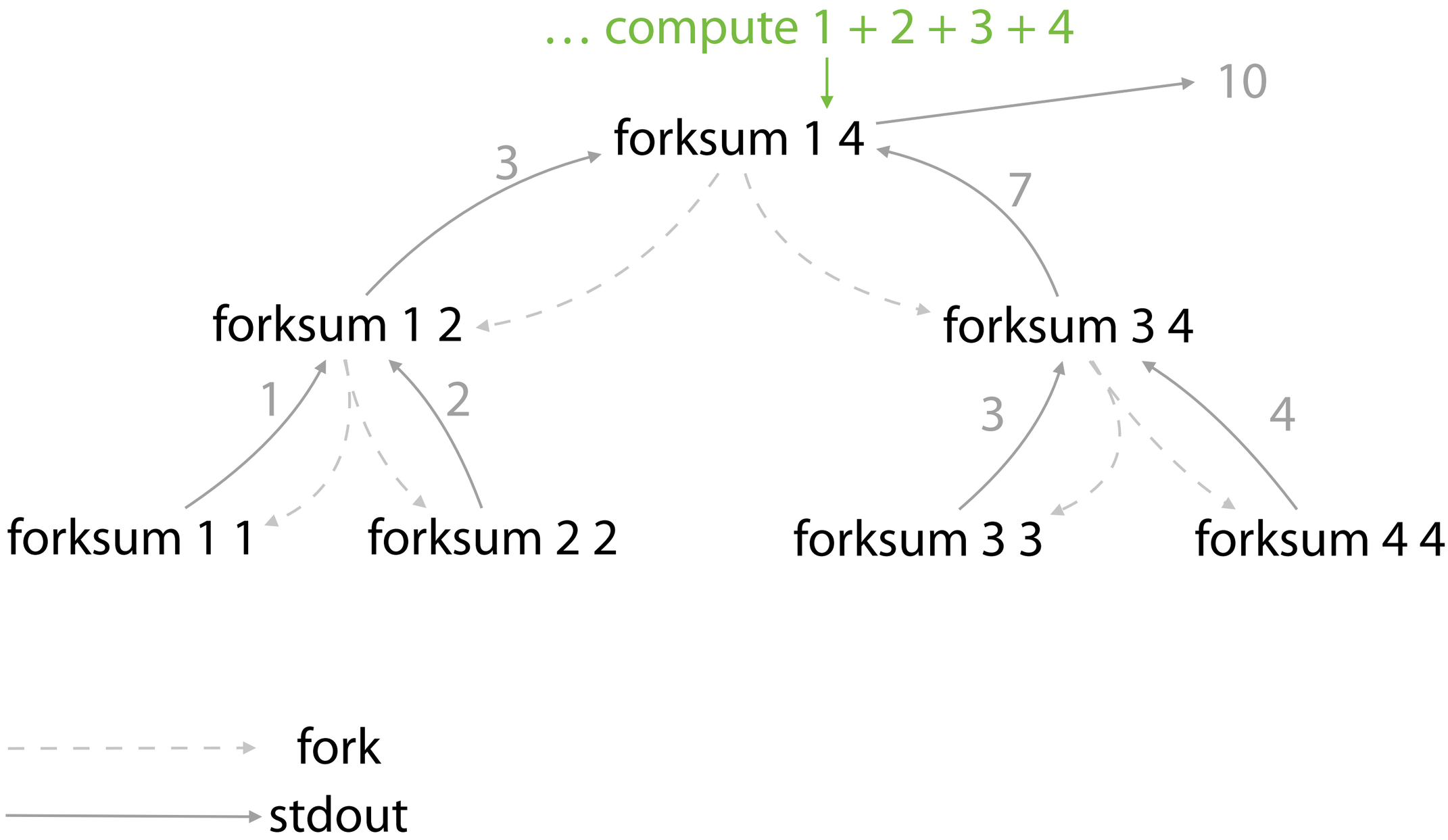

cpu
memory
fio-w
nginx
forksum

forksum receives 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 print their results, parse their outputs and print the sum


… compute 1 + 2 + 3 + 4
forksum 1 4
10
3
7
forksum 1 2
forksum 3 4
1
2
3
4
forksum 1 1
forksum 2 2
forksum 3 3
forksum 4 4
fork
stdout

SYSCALLS!

```
./${EXECUTABLE}   |   tail -1
```

fork, pipe, dup/dup2, close, write, read, wait
[execve]

`fork`	continue program as two separate processes Return value tells if in the child (0) or parent process (child pid)
`pipe`	create a pipe to transfer data between two processes
`read`	can be used to read from a pipe
`dup/dup2`	assign the standard output of a child to a pipe (and vice versa)
`wait`	wait for child processes to exit

Also useful: `atoi, printf, fprintf, close, exit`

Hint: the `execve` system call is not necessary

<problems?>

Check results, wait+retry, …

- 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) [default arguments are also okay]
- 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

How to container

<why container?>

initially™: isolation

<demo>

open questions

SHUT DOWN YOUR VMS

(in case you used any cloud platforms)

end