vmgo

mechiel lukkien

go amsterdam - 17 oct 2019

github.com/mjl-/vmgo

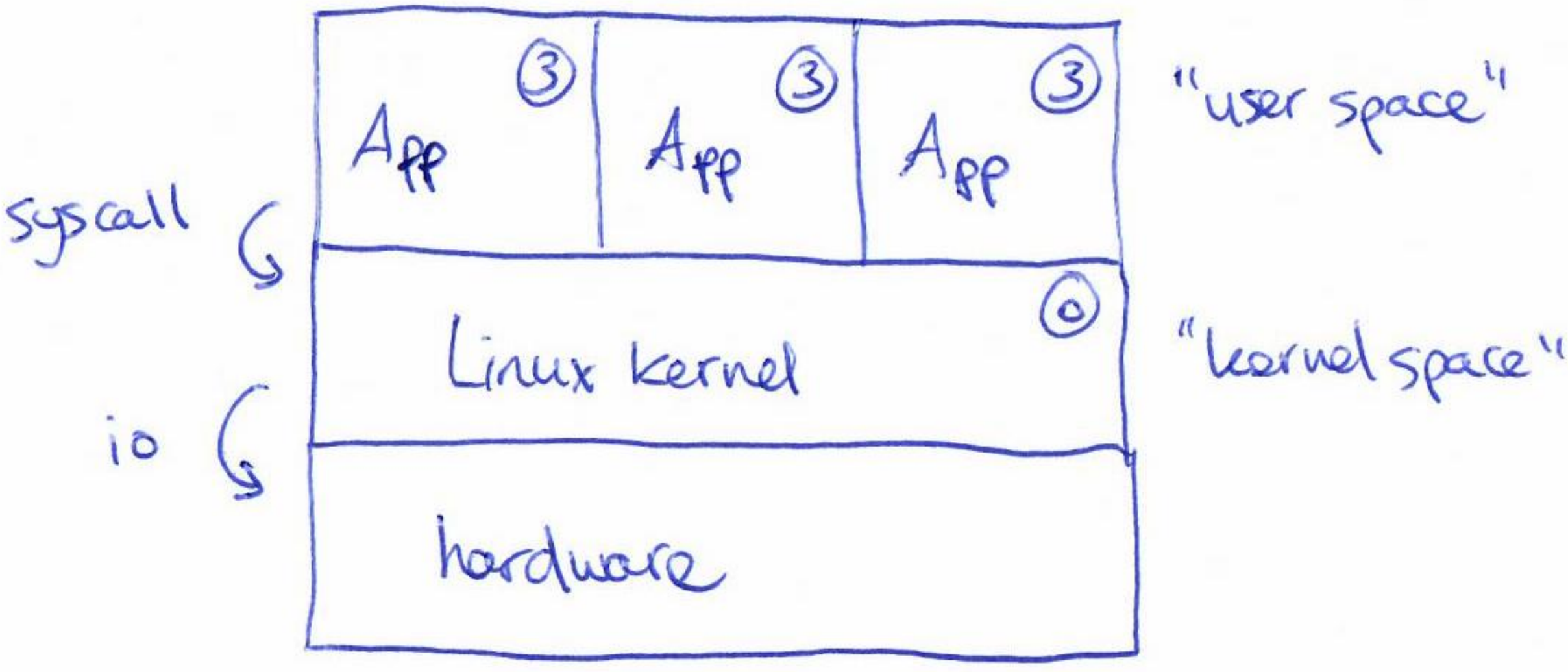
table of contents

goals

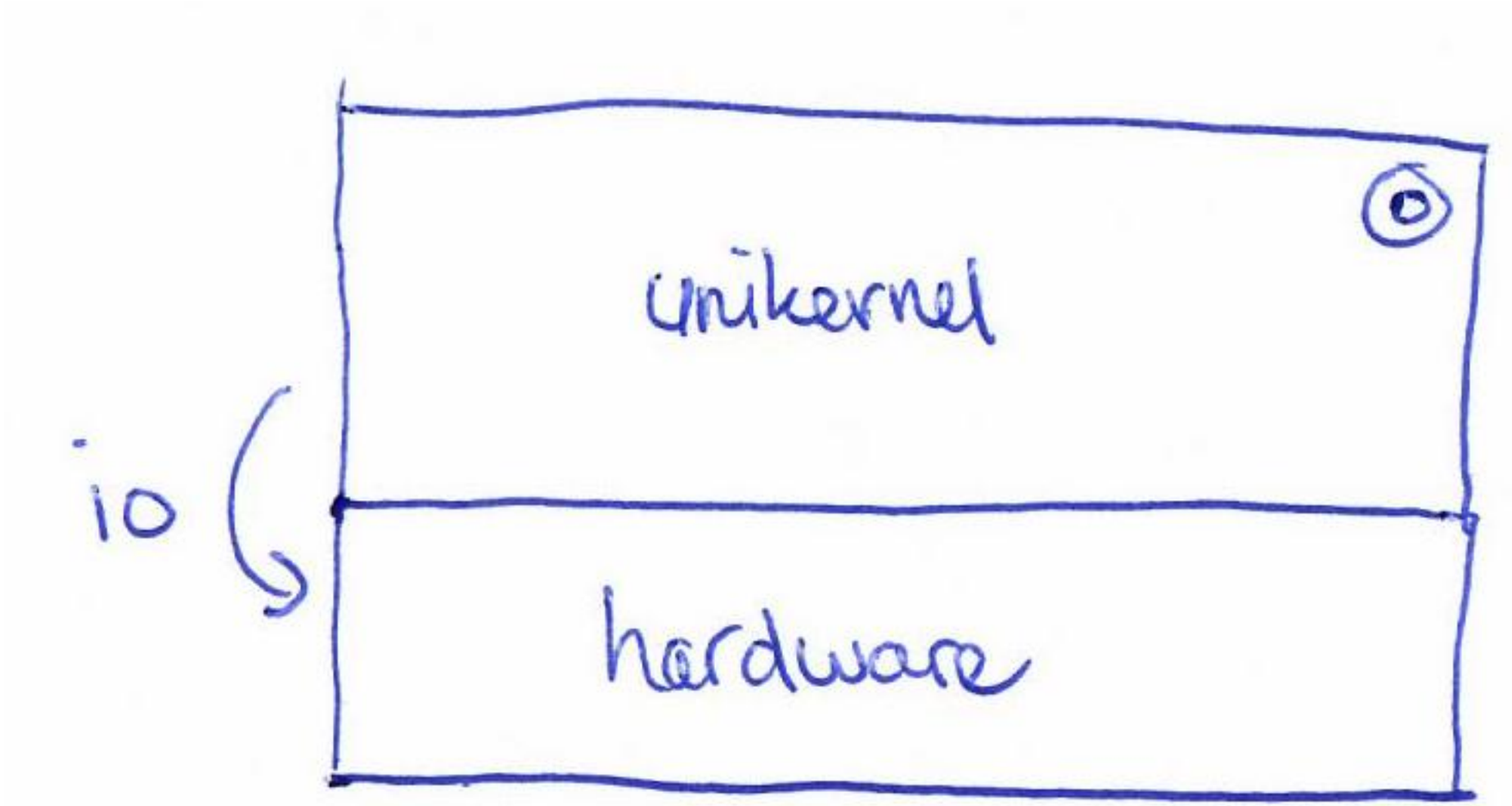
goals

- run go programs
  - secure
  - isolated
    - minimal software stack
- how?
  - unikernels
  - smaller hypervisor

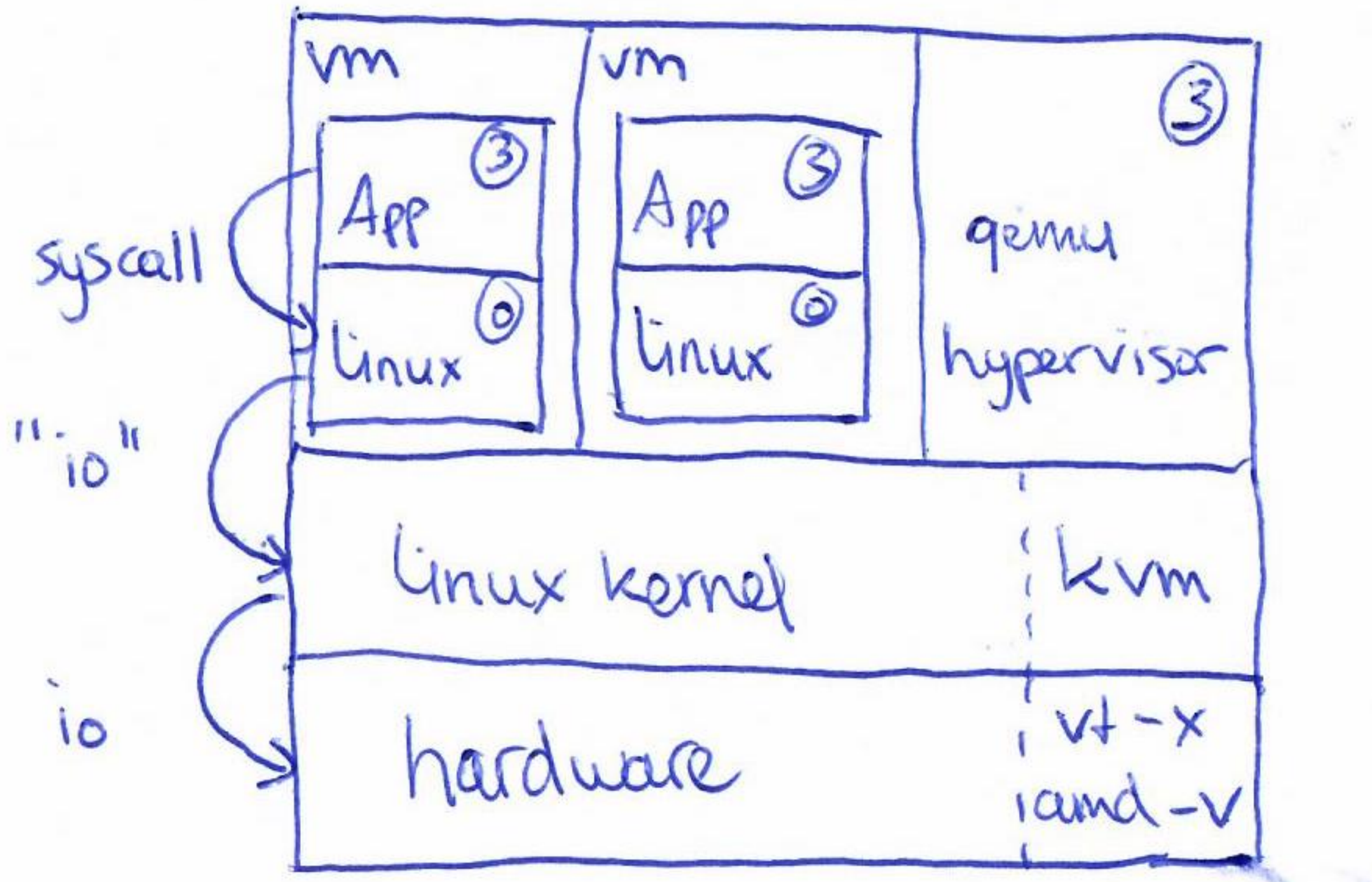
kernel


App
3
App
3
App
3
"user space"
syscall
Linux kernel
0
"kernel space"
io
hardware

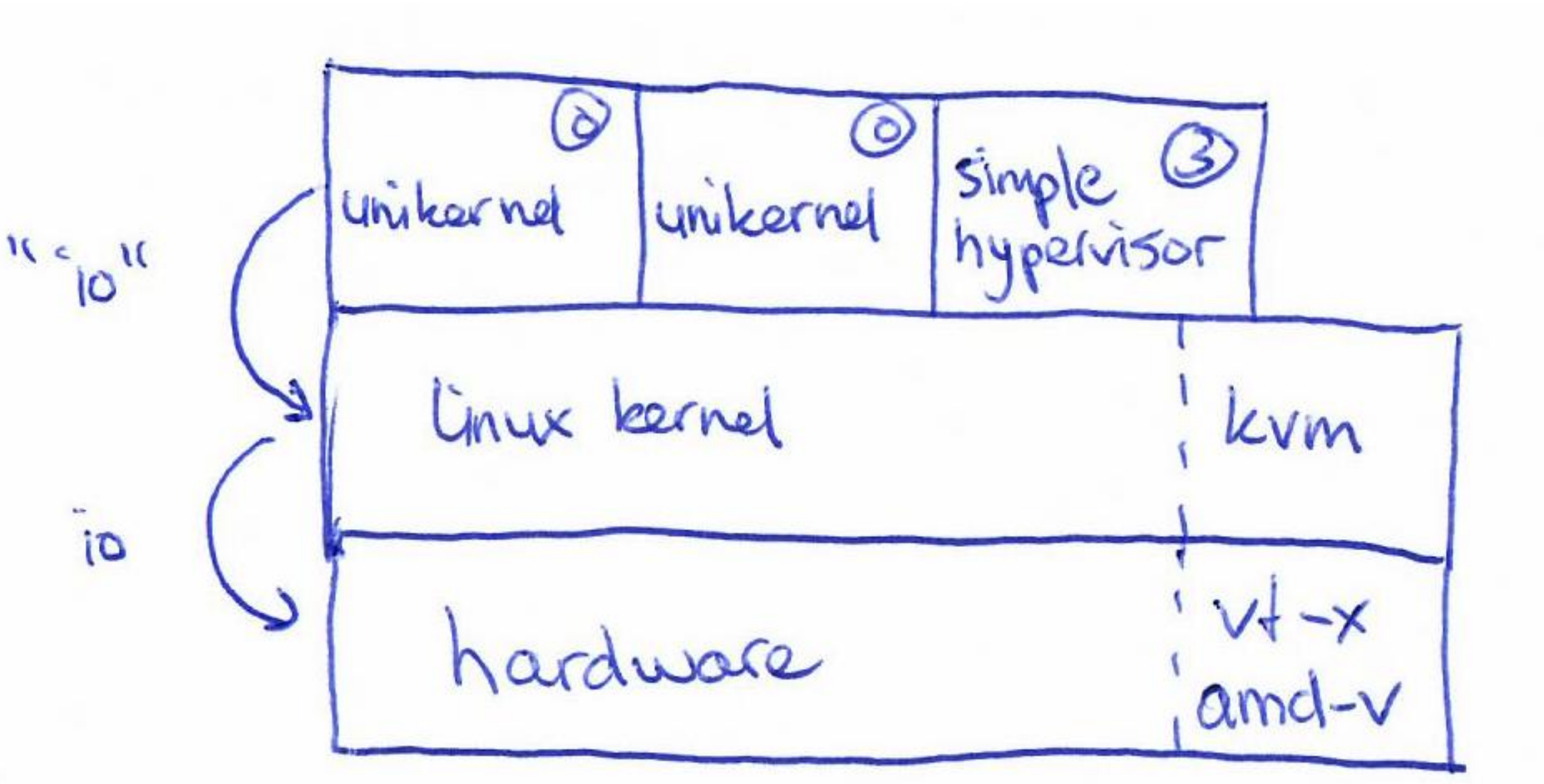
unikernel



virtualisation



simpler virtualisation



benefits

- security
  - less code
    - replace C with Go
- performance (?)
  - less context switching, boot times
- understanding the go toolchain & runtime

small hypervisor - solo5

- simple boot
- single virtual cpu, fixed memory
- no interrupts
- simple hypercall mechanism
- few hypercall functions:
    - walltime, puts, netread, netwrite, blockread, blockwrite, poll, halt
- tsc for time

workflow

$ cd existing-go-code

$ GOOS=solo5hvt go build -o unikernel

$ solo5-hvt --net:net0=tap0 --block:blk0=disk.img ./unikernel

work

replace

- files, sockets, processes

+ raw disk, raw network, virtual cpu

need to modify go toolchain & runtime!

go runtime

- src/runtime
- part of your go binary
- goroutines, channels, memory management, network fd's
- uses system calls

approach

approach

start with GOOS=openbsd

strip system calls until there are none left

1. files
2. sockets
3. processes

1. files

do we need files? os.Open

standard library needs surprisingly few files

- timezone database
- /etc/... resolv.conf hosts services protocols
- /etc/... passwd group
- /etc/ssl/cert.pem
- /etc/mime.types

add support for "fake files"

- os.AddFile(path string, data []byte)

later: add full file system support

2. network

- abstracted away in package "net": Dial, Listen
- implement TCP/IP stack in Go
- lots of work!

netstack

- github.com/google/netstack
- github.com/google/netstack/tcpip/adapters/gonet
- modify "net" package to use netstack

netstack sidebar

github.com/mjl-/vmgo/tree/netstack

"net" implemented with netstack (linux/amd64 only)

$ cd webapp

$ go build -tags netstack

$ export GONET='verbose; \
nic id=1 ether=fe:e1:ba:d0:33:33 mtu=1500 dev=tap0 sniff=true; \
ip nic=1 addr=192.168.1.100/24; \
route nic=1 ipnet=0.0.0.0/0 gw=192.168.1.1; \
dns ip=8.8.8.8'

$ ./webapp

3. processes

- remove support for "os/exec"
- hard: remove processes from runtime
  - looks like js/wasm
- quite some process-related code in runtime

new GOOS=solo5hvt

- add GOOS definitions to files
- copy & add add code to runtime/stdlib for solo5

- run empty go program!

    package main

    func main() {}

state

- compile simple programs & launch with solo5-hvt
    - create goroutines
    - write to stdout
    - time.Now, time.Sleep
    - read/write network packets from user go program

next

- network i/o with runtime
- integrate netstack
- support files
- more solid, more packages
- preemption...

future

- different target?
  - multiple vcpu's, preemption
  - firecracker
  - xen
  - virtio
  - other?

- UI

- look at other projects: AtmanOS, others?

info

github.com/mjl-/vmgo

mechiel@ueber.net

github.com/Solo5/solo5

gopher images by Renee French, CC BY 3.0

better than processes?

- still whole kernel running

- removed ring0 protection from vm

+ possibly in future: encrypted vm with amd sev-es

+ potential for live migration

runtime & processes

- "G" - a goroutine
- "M" - for machine: host OS process
  - one for each GOMAXPROC ("P")
  - for system calls (eg file i/o, network i/o)
  - preemption