Lecture 24 CS695.

① - processes & threads ~ via server.

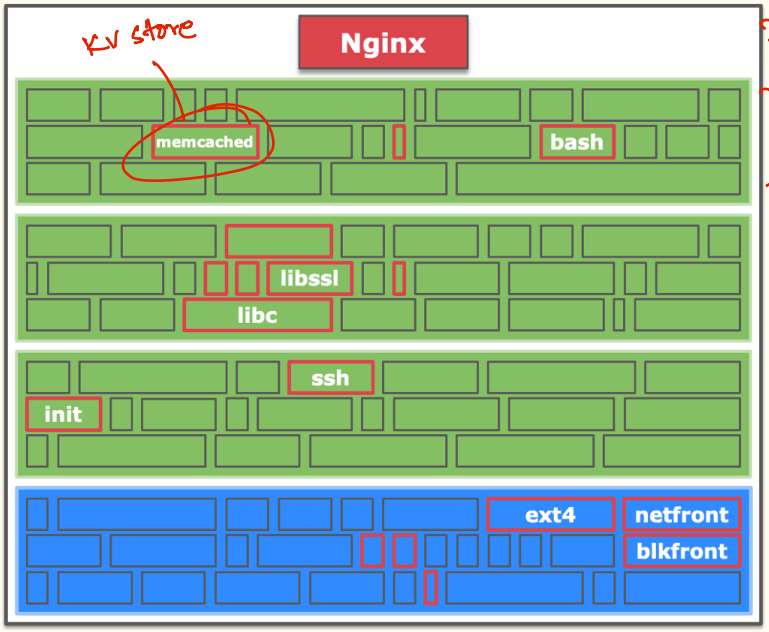
~ VMs

~ containers

~ unikernels ~ optimized Kernel + library which is stack tuned for a single application.

② [LightVM] SOSP 2017

My VM is lighter (and safer) than your containers.



application

system tools / middleware

std. libraries

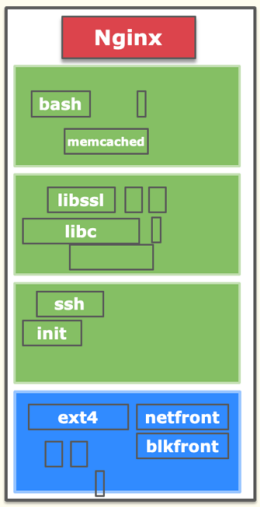
system binaries

Kernel components

Amazon

↳ Fire Cracker VM

↳ light VM + container (lambda fns).



Tinyx VMs are also lightweight:

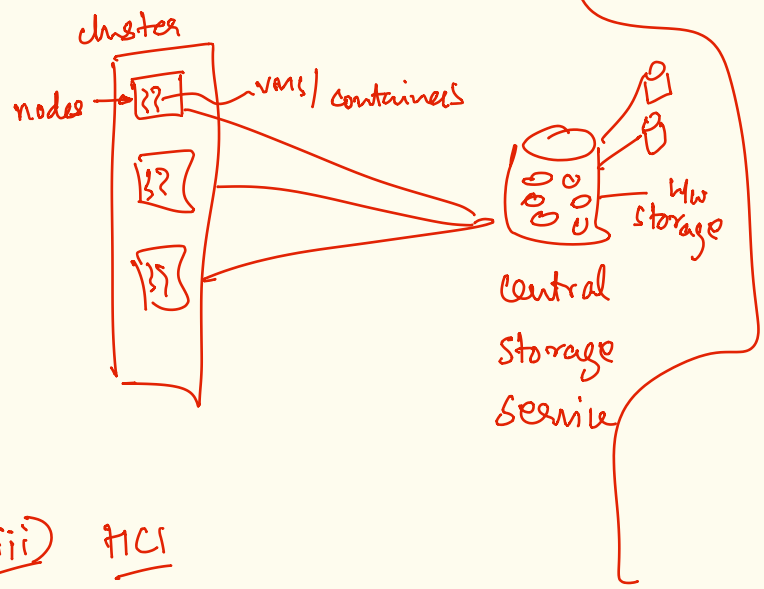
- Kernel: 1.5MB (compared to 8MB)
- Image size – 10-30MB (compared to 1GB).
- Boot: 200ms instead of 2s.

← how to determine e2e dependencies?

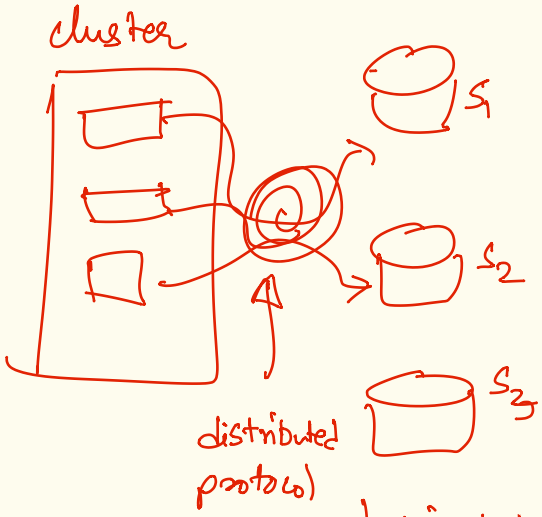
- how to build the e2e runtime env?

(2) data center architectures ← service hosting

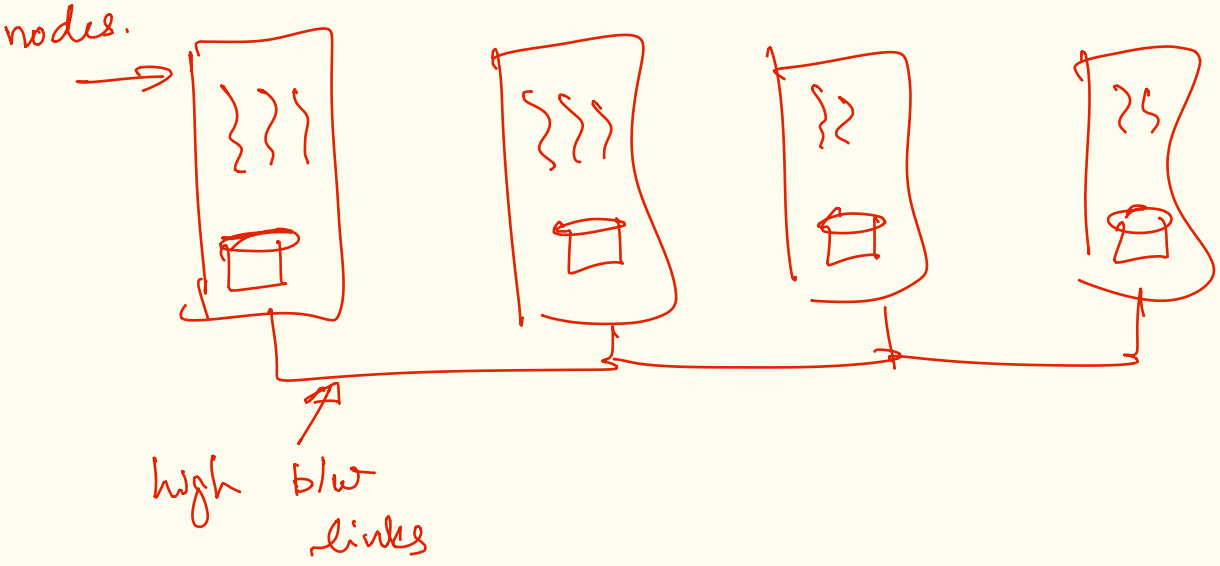
(i) clustered compute
centralized storage



(ii) clustered compute
clustered / networked storage

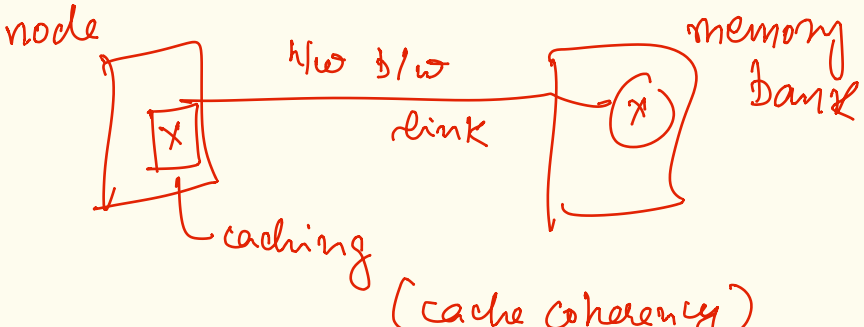


(iii) HCI
hyper-converged infra.
- all nodes contribute to compute & storage



capacity of cluster = sum of capacities of each node.

(iv) Rack-scale computing / Disaggregated infra.



CXL

(3) network processing for services / cloud appl^n's.

~ n/w traffic / processing reqs. ~~are~~ are non-trivial (in the kernel).

~ n/w pkt arrives → interrupt → IRQ handler → pkt. handling n/w stack processing → park at at a socket (sock fd)

(a) payload is in appl^n. buffer ← read on socket ←

(b) Split pkt processing.

IRQ handler + packet processing softirqd (kernel threads)

- interrupt handler
- copies pkt. to memory region DMAs
- updates state of recv. pkts
- returns from handler.

- process/read pkts for protocol processing

(c) polling

combining pkts. for processing

reusing sockets.

- socket
- bind
- listen

⇒ accept() ← new socket ← read / write

bottleneck at high connection rates.

multi-threaded

SOCK_REUSEPORT

---

multi-threading of accept on the same socket.

(d) Bypass the kernel itself.

— similar to the unikernel idea

└ perform custom n/w processing in user space.

└ esp. for custom headers.

└ NFV

network fn. virtualization!

- hw & sw techniques

└ DPDK

└ SRIOV

└ xdp — sw soln.

enabled via eBPF

extended Berkeley Packet filter.

④ storage

- distributed key-value stores ~ memcache, redis, memcached
- distributed FS ~ VMFS, GFS.

⑤ multi-cloud deployments

- services across providers.
- serverless seems promising!

⑥ acceleration-as-a-service

- GPUs!