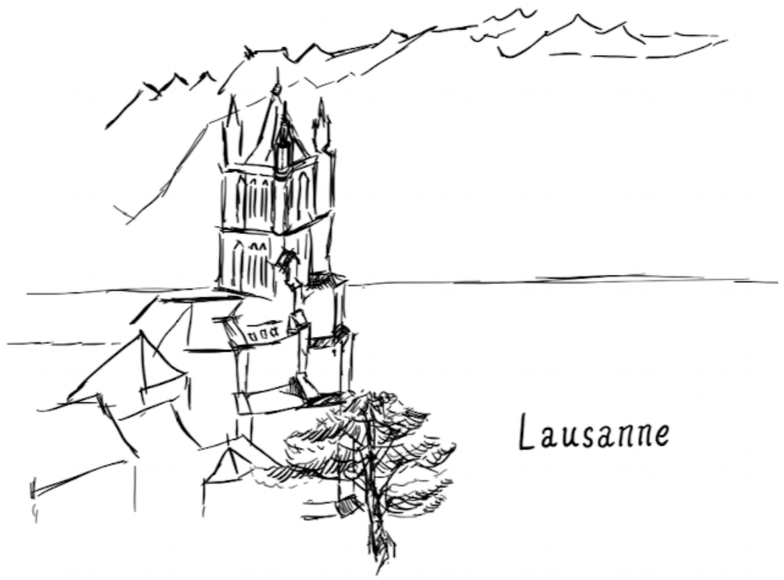
TMO: Transparent Memory Offloading

IOCost: Block IO for Containers

] in Datacenters



Two papers in ASPLOS 2022

All Drawings by Jing Liu

Memory Offloading

DRAM

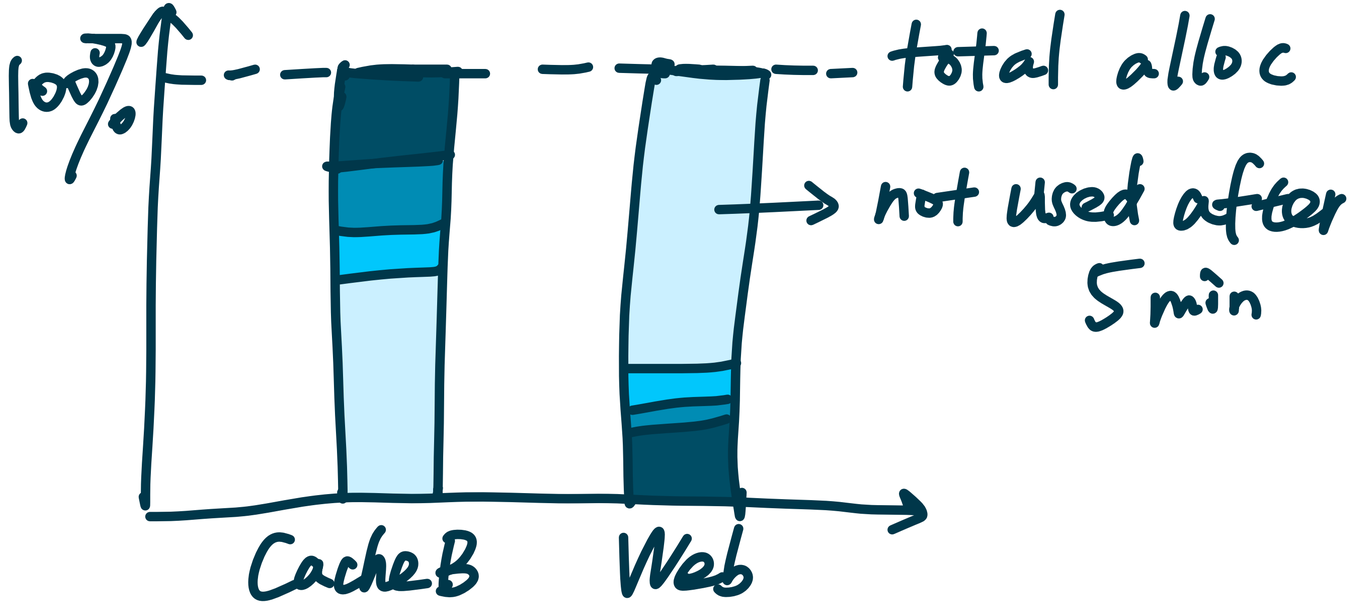
Apps consume a lot of memory

offload

⊕ Memory tiering

compressed mem

NUMA SSD



State-of-the-art & Issues

zswap (Google, ASPLOS 19)

② single slow memory tier (compressed mem)

② offline application profiling
metric = page-promotion rate

State-of-the-art & Issues

zswap (Google, ASPLOS 19)

② single slow memory tier (compressed mem)

⇒ add NVMe SSD → cheaper, power

⇒ Compression not applicable (Sometimes)

② offline application profiling

metric = page-promotion rate

⇒ Handle dynamic + diverse apps

App's sensitivity differs ⇒ e2e perf

⇒ Handle device heterogeneity (diff SSDs)

slow device implies low PPR

TMO

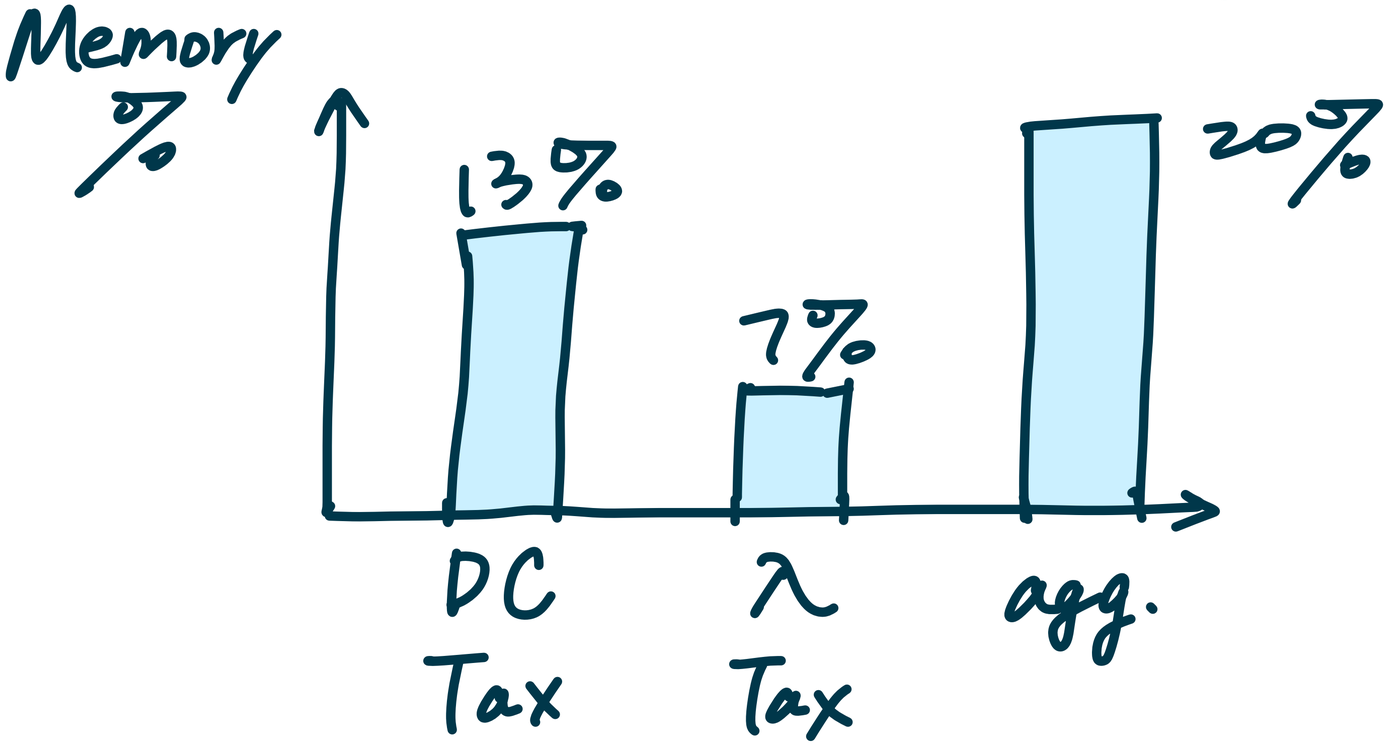
- Transparent memory offloading
- containerized environment ⇒ interesting setting

key questions:
- How much memory to offload?
- What memory to offload?

⇒ mechanism for better information
⇒ perf model captures: app-centric slowdown ⊕ diff devices
⇒ framework + algorithm to adjust at runtime

Data center Setting / Deployment ⊕ Challenges

Memory Tax = { infrastructure-level functions
packaging, logging profiling ....
microservices
routing, proxy...



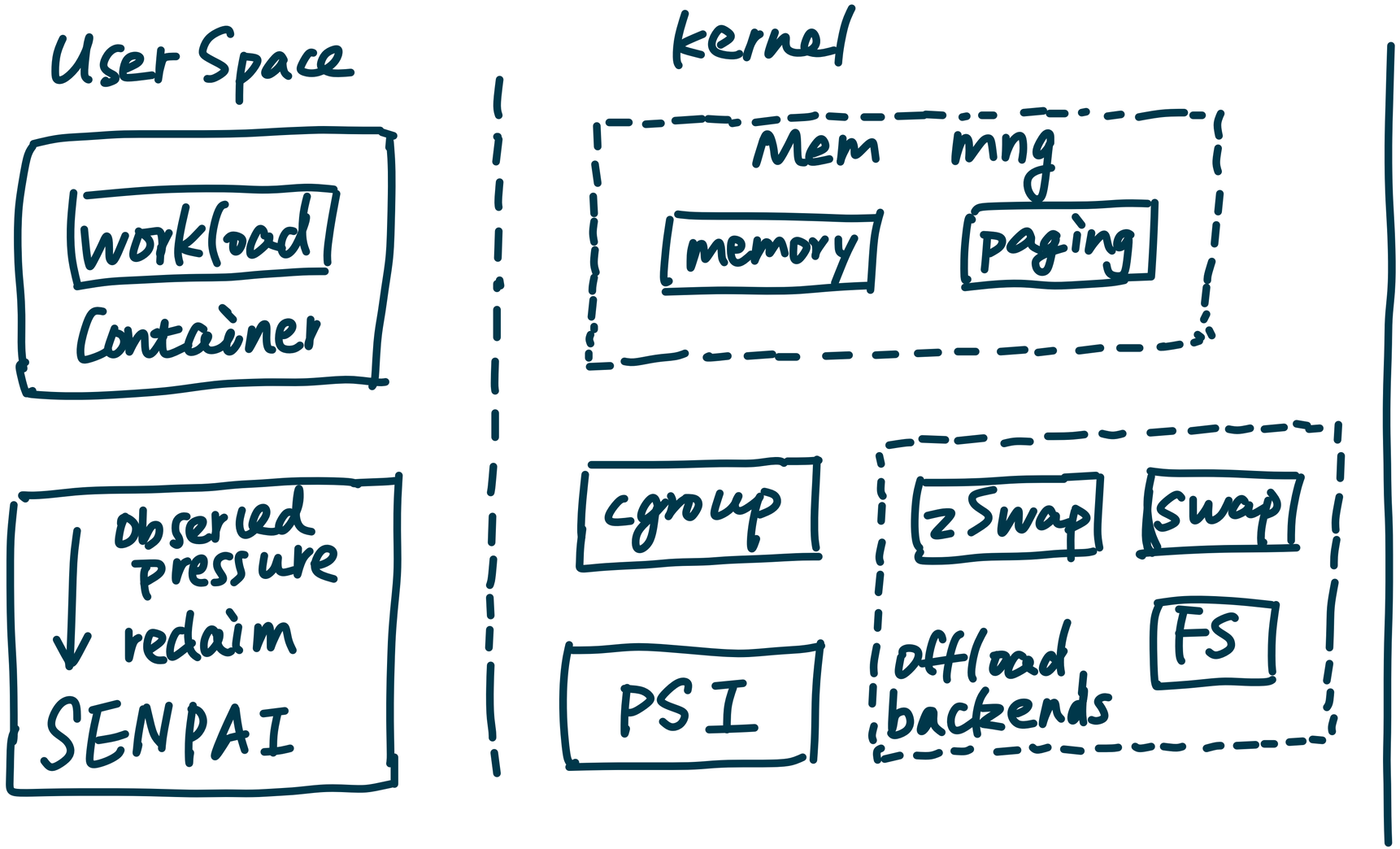
memory tax SLA = relaxed
⇓
primary target of offloading (first version)

Data center Setting / Deployment ⊕ Challenges

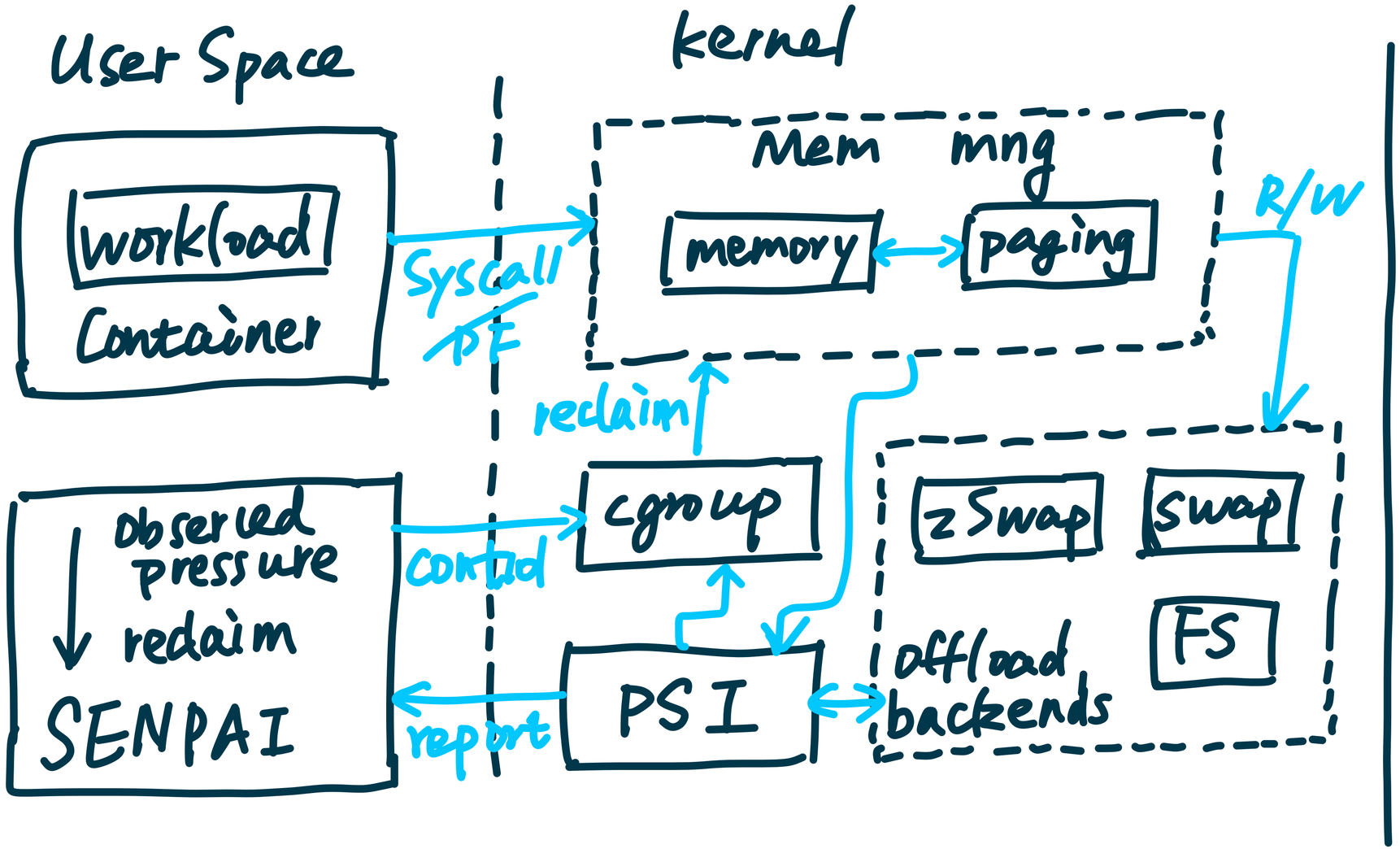
HW heterogeneity of offload backend

- Different types of NVMe SSDs
  - types: generations
  - { perf, endurance ★ important }
  - eg. R/W latency: 9.3 ms → 470 us
- Compressed memory: faster!

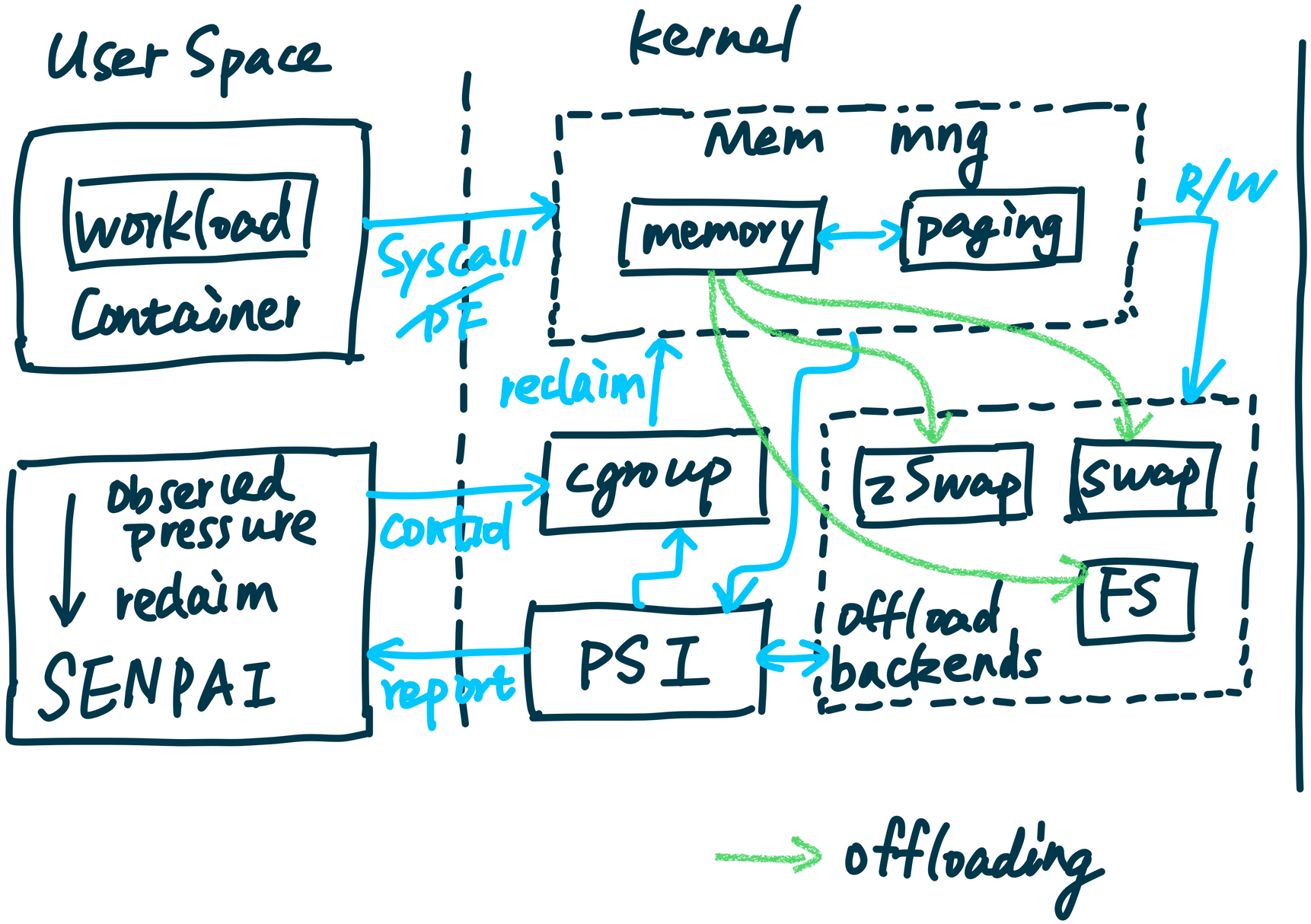
TMO Design


User Space
kernel
workload
Container
Mem mng
memory
paging
observed
pressure
reclaim
SENPAI
cgroup
zSwap
swap
FS
offload
backends
PSI

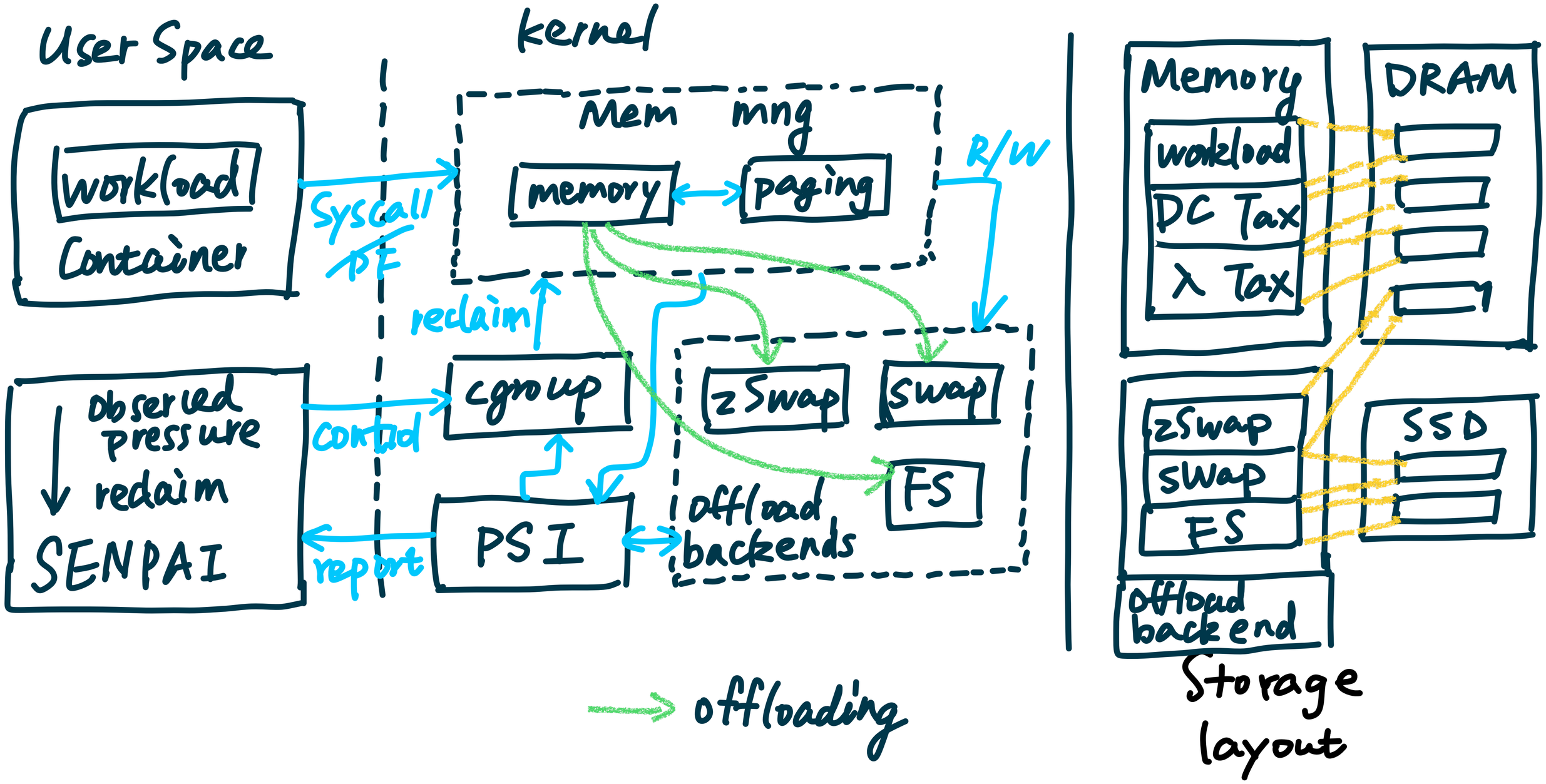
TMO Design


User Space
kernel
workload
Container
Syscall
PF
Mem mng
memory
paging
R/W
reclaim
observed pressure
reclaim
SENPAI
control
report
cgroup
PSI
zSwap
swap
FS
offload backends

TMO Design


User Space
kernel
Mem mng
workload
Container
Syscall
PF
memory
paging
R/W
reclaim
observed pressure
reclaim
SENPAI
config
cgroup
zSwap
swap
FS
offload backends
PSI
report
offloading

TMO Design


User Space
kernel
workload
Container
Syscall
PF
Mem mng
memory
paging
R/W
reclaim
observed pressure
reclaim
SENPAI
control
cgroup
zSwap
swap
FS
offload backends
report
PSI
offloading
Memory
workload
DC Tax
λ Tax
DRAM
zSwap
swap
FS
SSD
offload backend
Storage layout

Resource Pressure: Pressure Stall Info (PSI)

How much memory to offload?

metric: predict the resulting performance after offloading

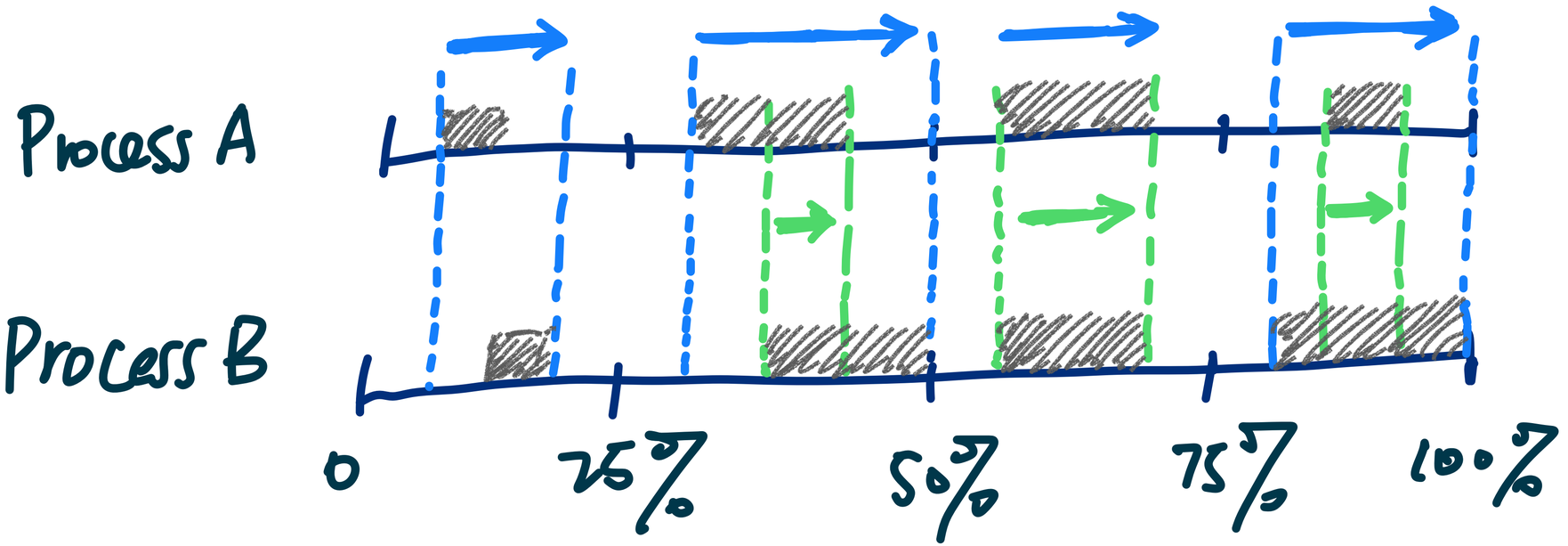
why hard?
- kernel's information is indirect ⇒ mechanism support ⊕
- single metric is hard
- accurately attribute performance penalty to offloading
  - eg. workload start up
  - workload working set transition
  - ....

mechanism to provide information && accounting to container (in kernel)

Resource Pressure: Pressure Stall Info (PSI)

How much memory to offload?

metric: predict the resulting perfomance after offloading



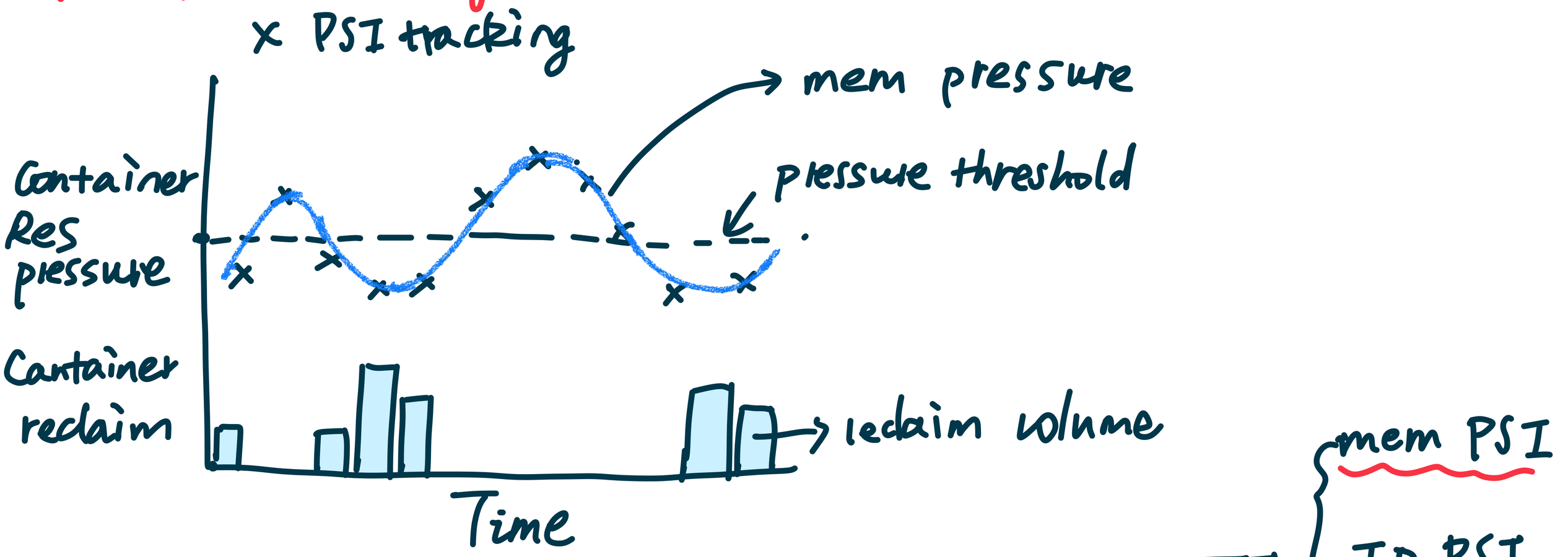
→ Some: One process stall for resource

→ Full: All processes stall for resource

▨ Stall

Determine Memory Requirement of Containers

How much memory to offload?



Senpai: userspace process PSI as feedback

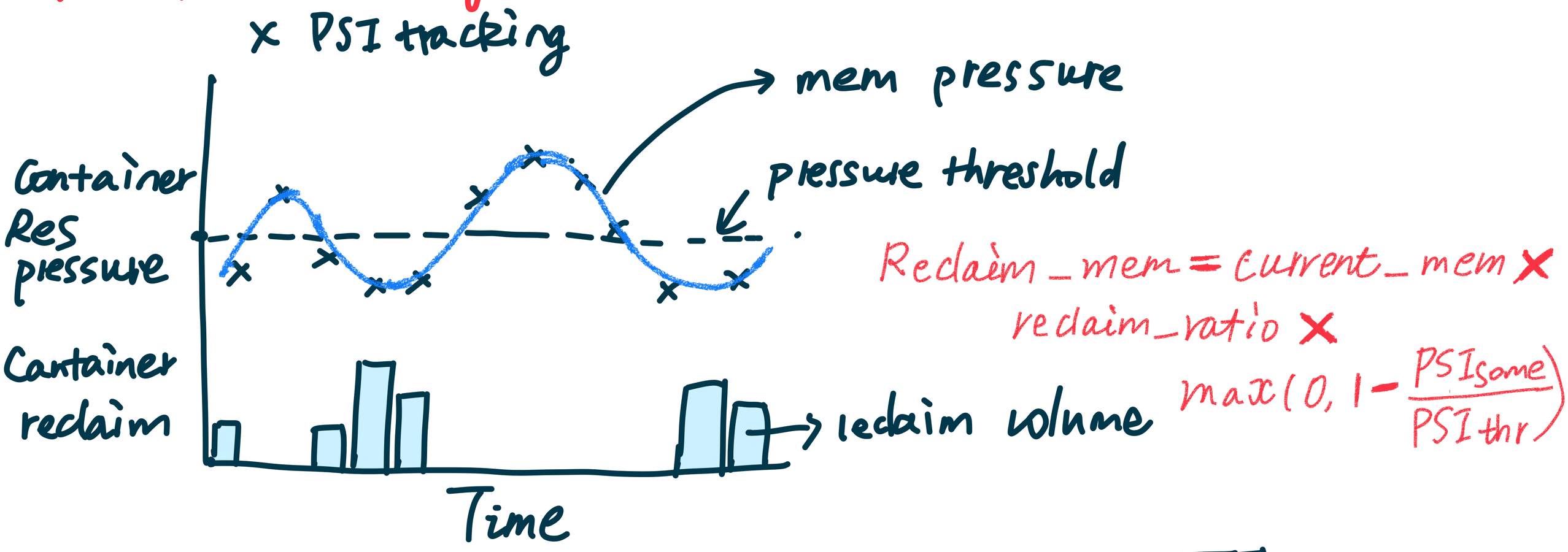
cgroup config

kernel

- mem PSI
- IO PSI
  - SSD endurance
- CPU PSI
  - not significant

Determine Memory Requirement of Containers

How much memory to offload?



$$Reclaim\_mem = current\_mem \times reclaim\_ratio \times \max\left(0, 1 - \frac{PSI_{some}}{PSI_{thr}}\right)$$

Senpai: userspace process PSI as feedback

cgroup config

kernel

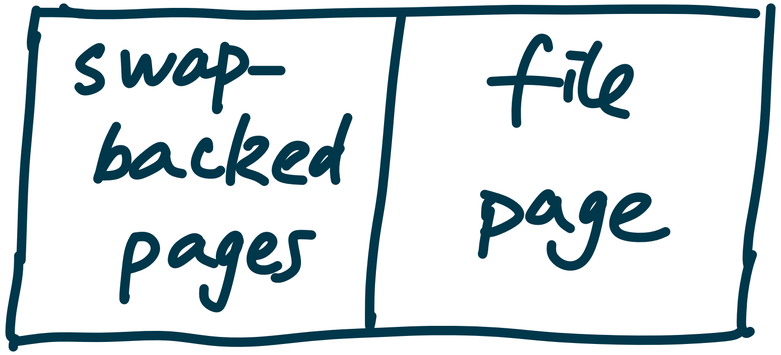
Kernel Optimization for memory offloading

What memory to offload?

mechanism: kernel's reclaim

issue: kernel's swaping is very conservative
skewed heavily towards file cache

+ refault-detect mechanism
  non-resident cache tracking
  ⇒ reuse-distance
  ⇒ exclude first-time-accessed file cache



+ if refault happens { refault rate, swap-in rate } balance

TMO - Eval: Performance + memory saving

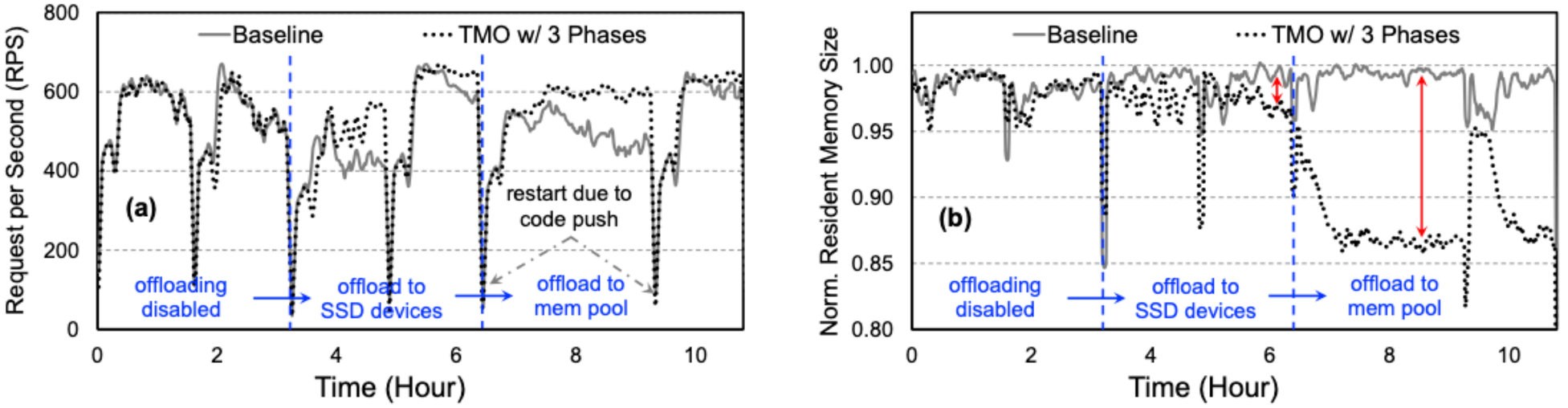


Figure 11: Performance improvement and memory saving for the Web application on memory-bound hosts. We set up two tiers: started with identical configuration without any swap enabled and later enabled SSD & then compressed memory offloading on the 2nd tier.

TMO - Eval: PSI

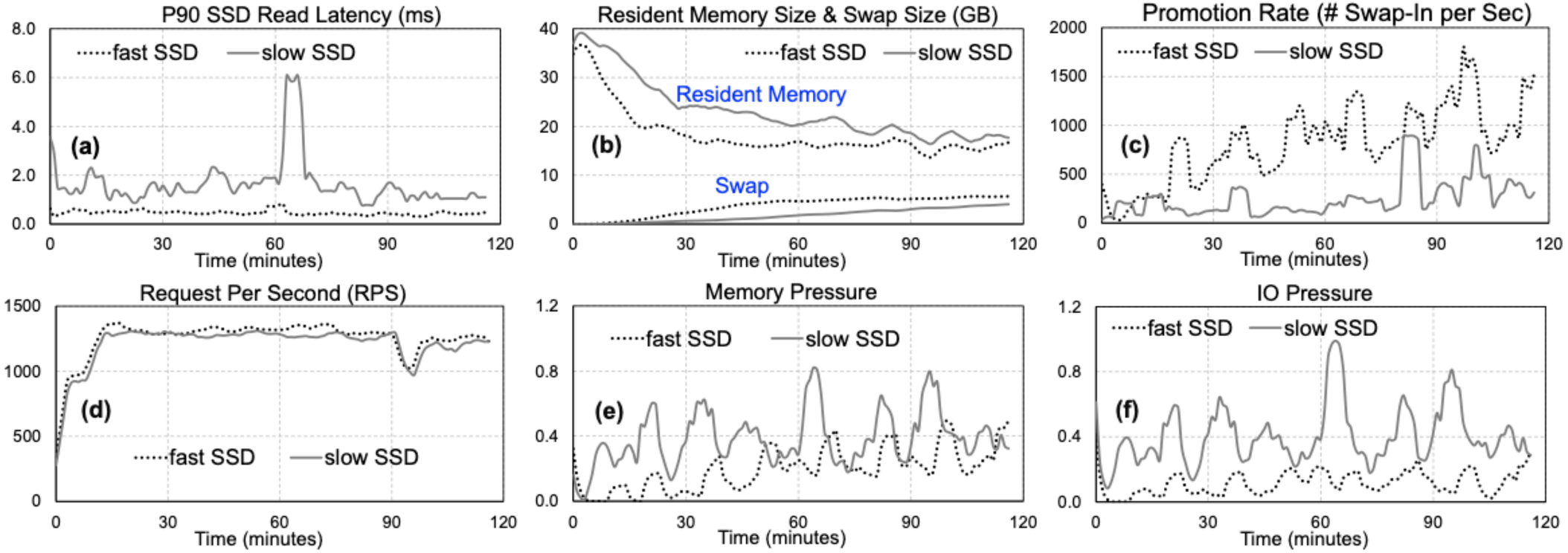


Figure 12: The Web application under the control of TMO with fast and slow SSDs. The solid-gray configuration has lower RPS/performance but actually lower promotion/swap-in rate; while its PSI metrics are indeed higher.

TMO - Eval: Device Endurancement

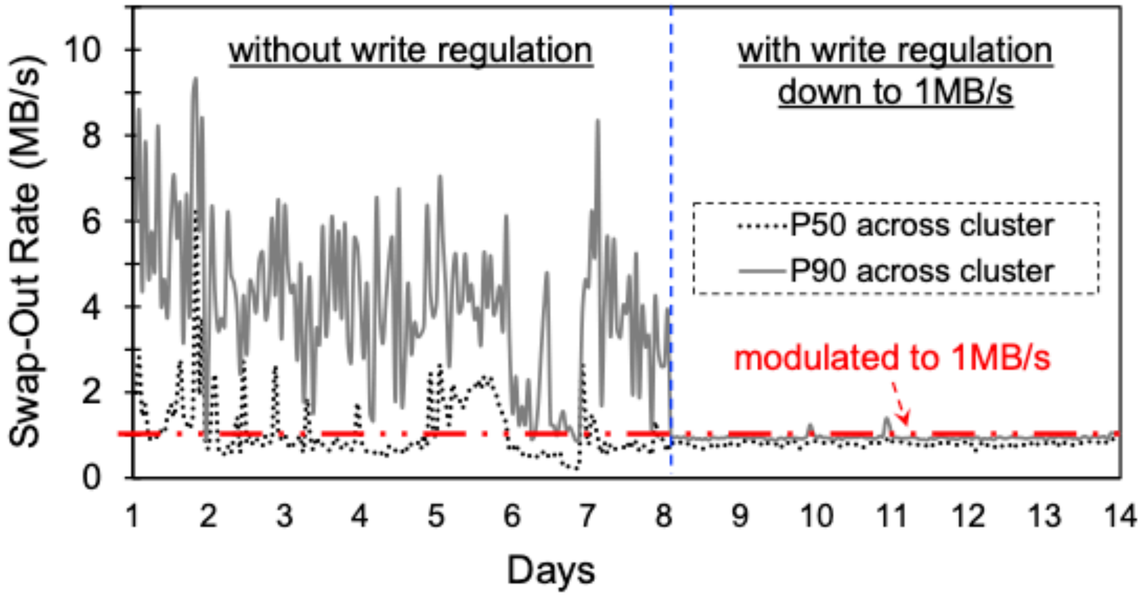
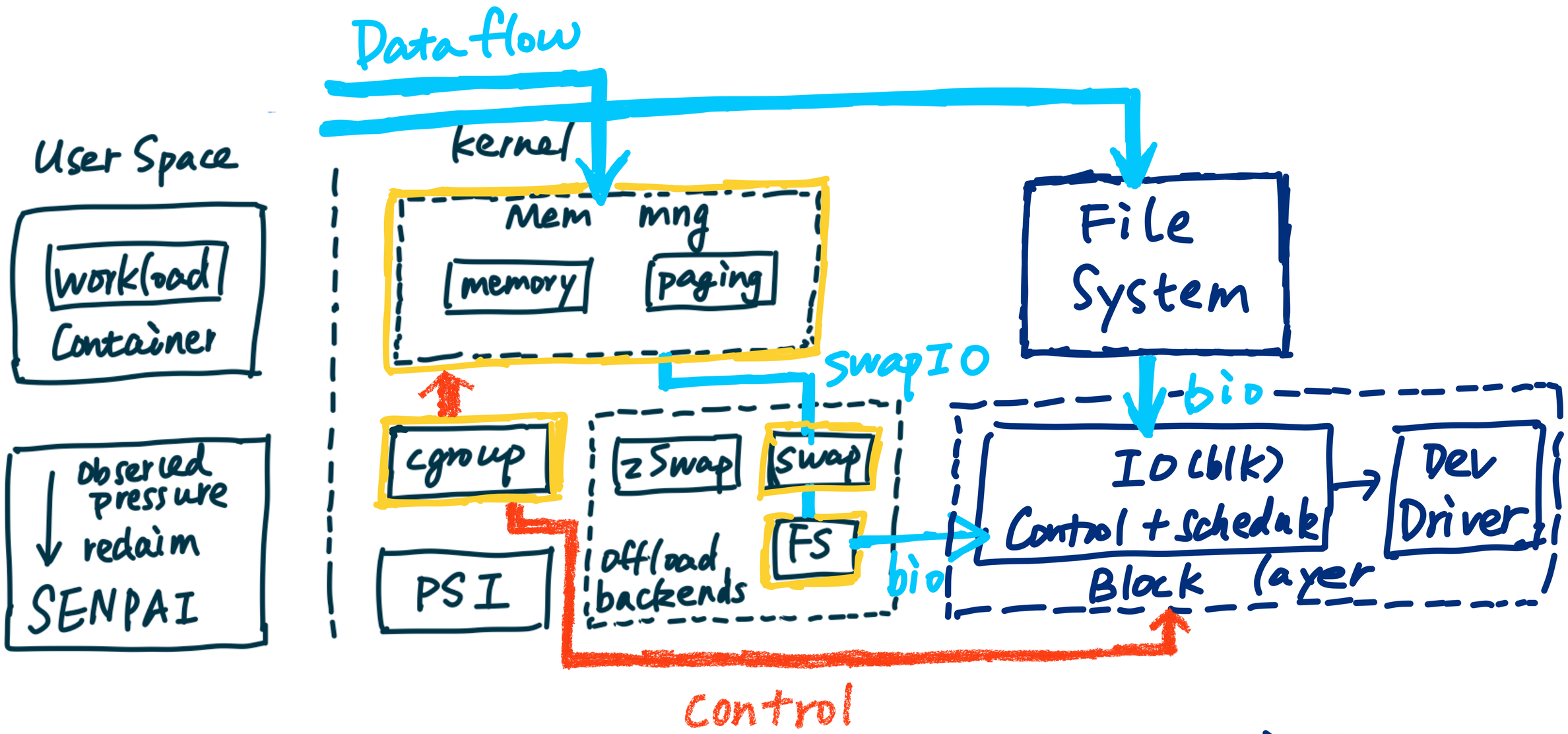


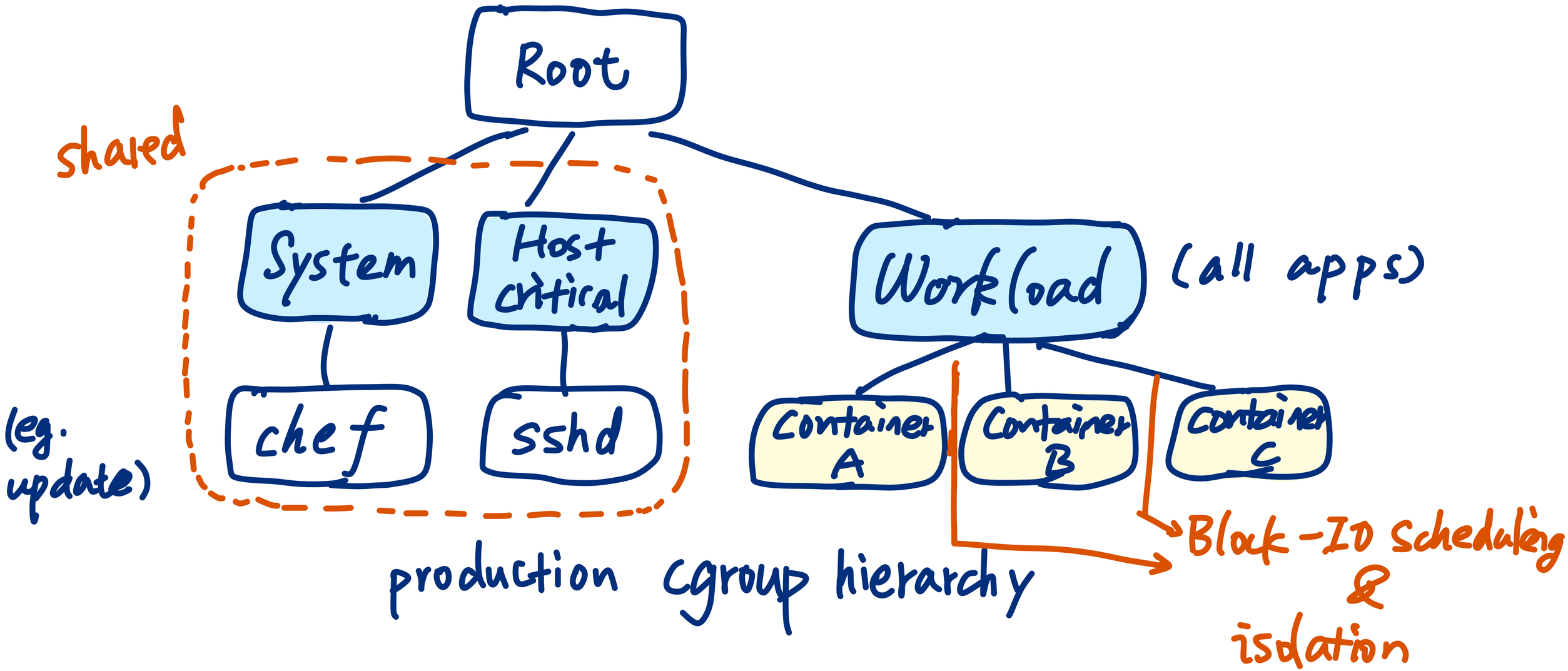
Figure 14: Swap rate with and without write regulation.



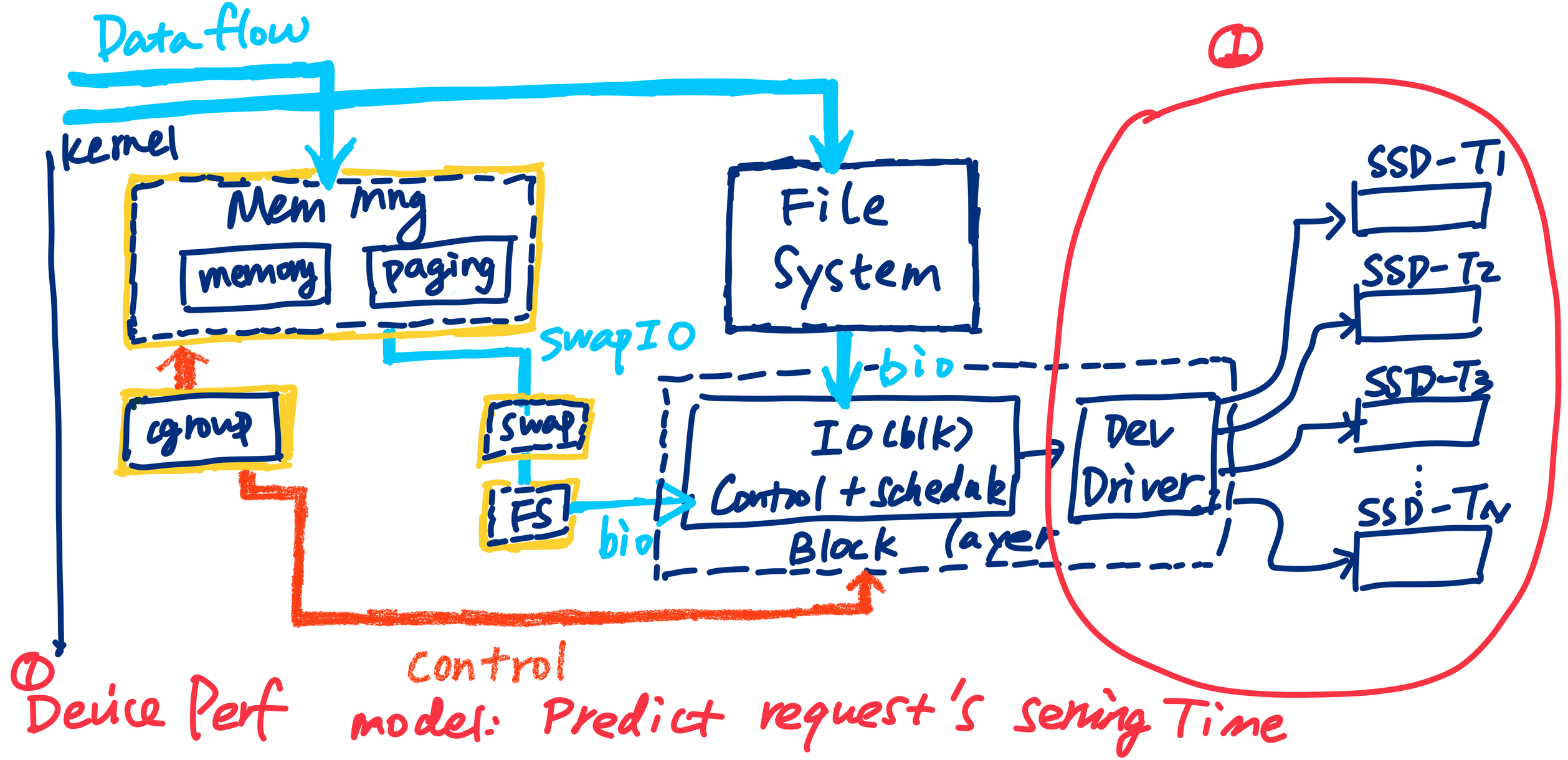
IO cost: Block IO control

- PC containers
- heterogeneity in Devices

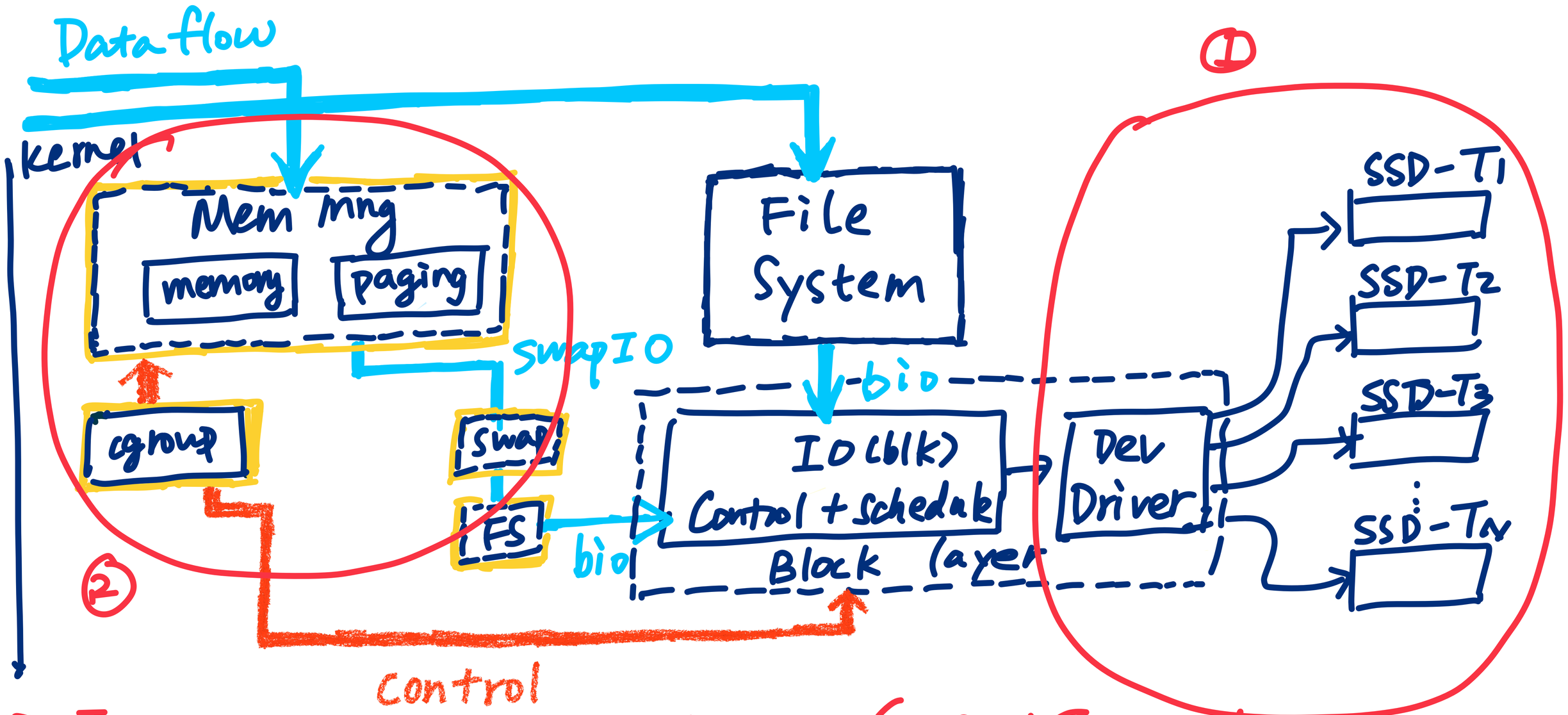
Data center Setting / Deployment ⊕ Challenges



Data center Setting / Deployment ⊕ Challenges

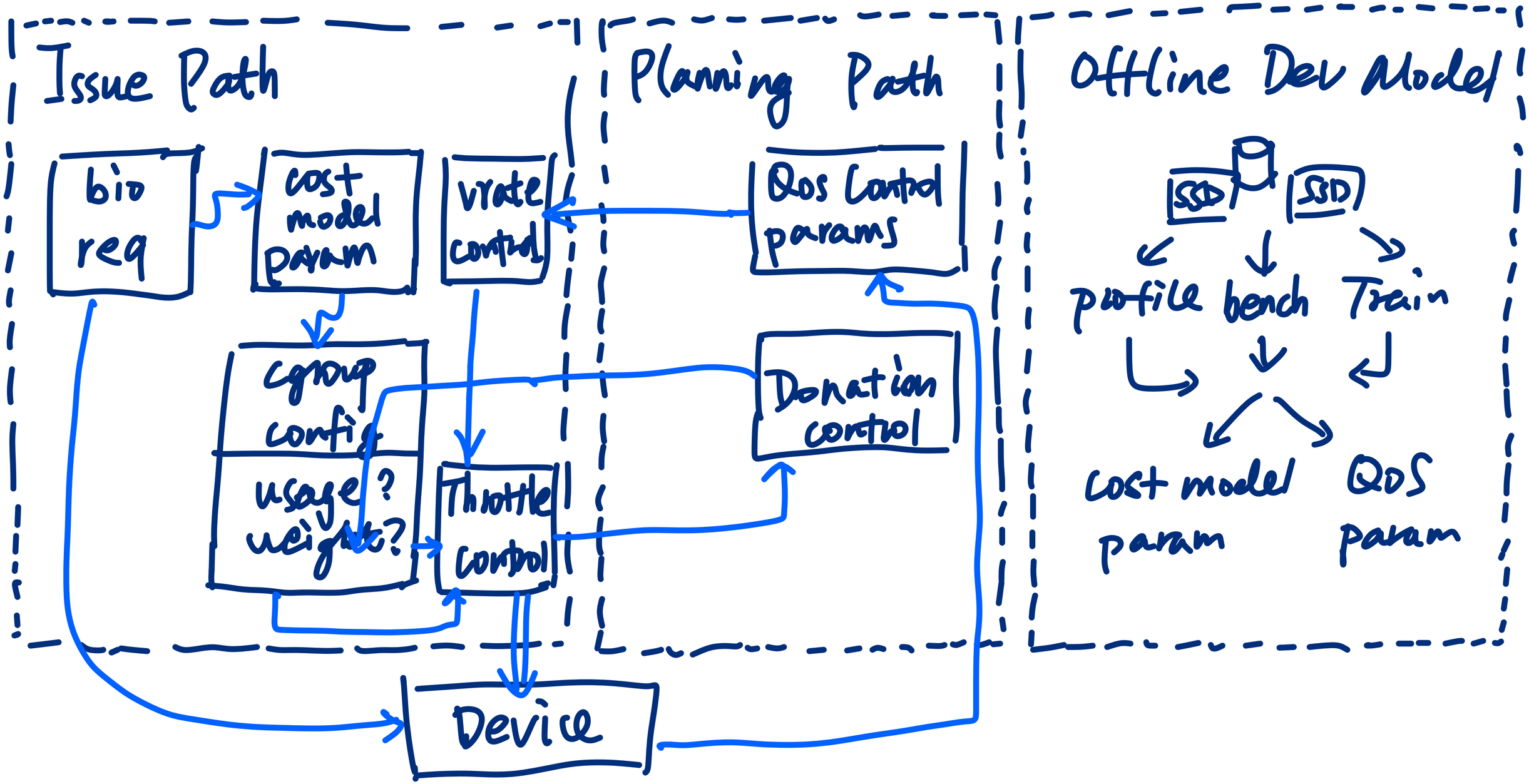


Data center Setting / Deployment ⊕ Challenges



② Interation with memory subsystem (mem + IO control make coherent decision)

IO Cost Overview



Summary of IO Cost

⇒ Offline performance model of device (heavyweight model)

⊕ runtime block IO control under dynamic workload (live model) ⊕

- fast path : io operation / policy enforcements
- slow path : policy decison

Summary: TMO + IOcost

⇒ Mechanism support for more control in kernel
+ cross component coordination

⇒ userspace process for policy level decision

⇒ Modeling the device from different aspects
+ online approximate + query

⇒ more direct app-performance metrics after certain resource control: $\Delta res \Longrightarrow \Delta perf_{APP}$

levels, a lot of components, shared infra
eg. sidecar container