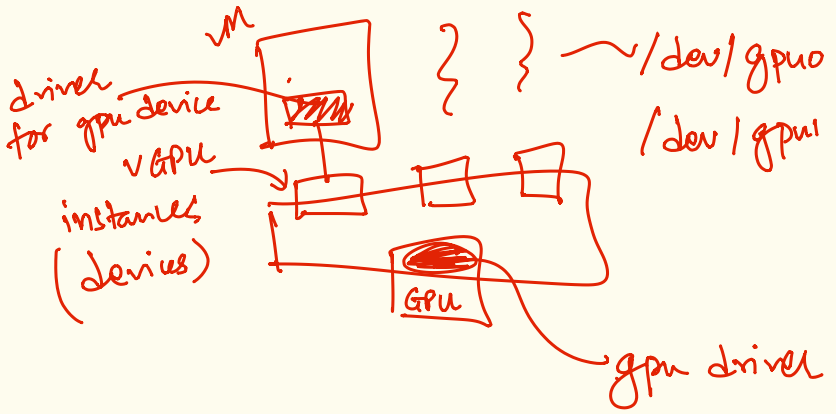
CS695

GPU virtualization

① h/w - assisted GPU virtualization

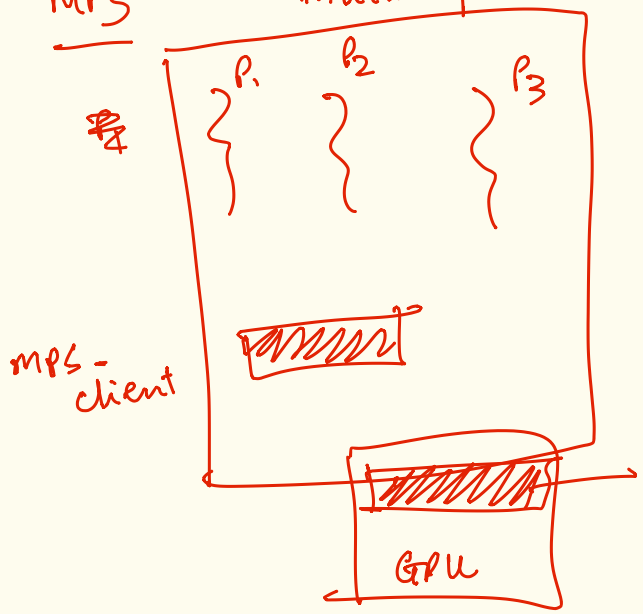


① MiG — spatial mem & cores

② vGPU — spatial mem + temporal cores

→ discrete virtual GPUs & use them for sharing / hosting.

③ MPS — multi-process service



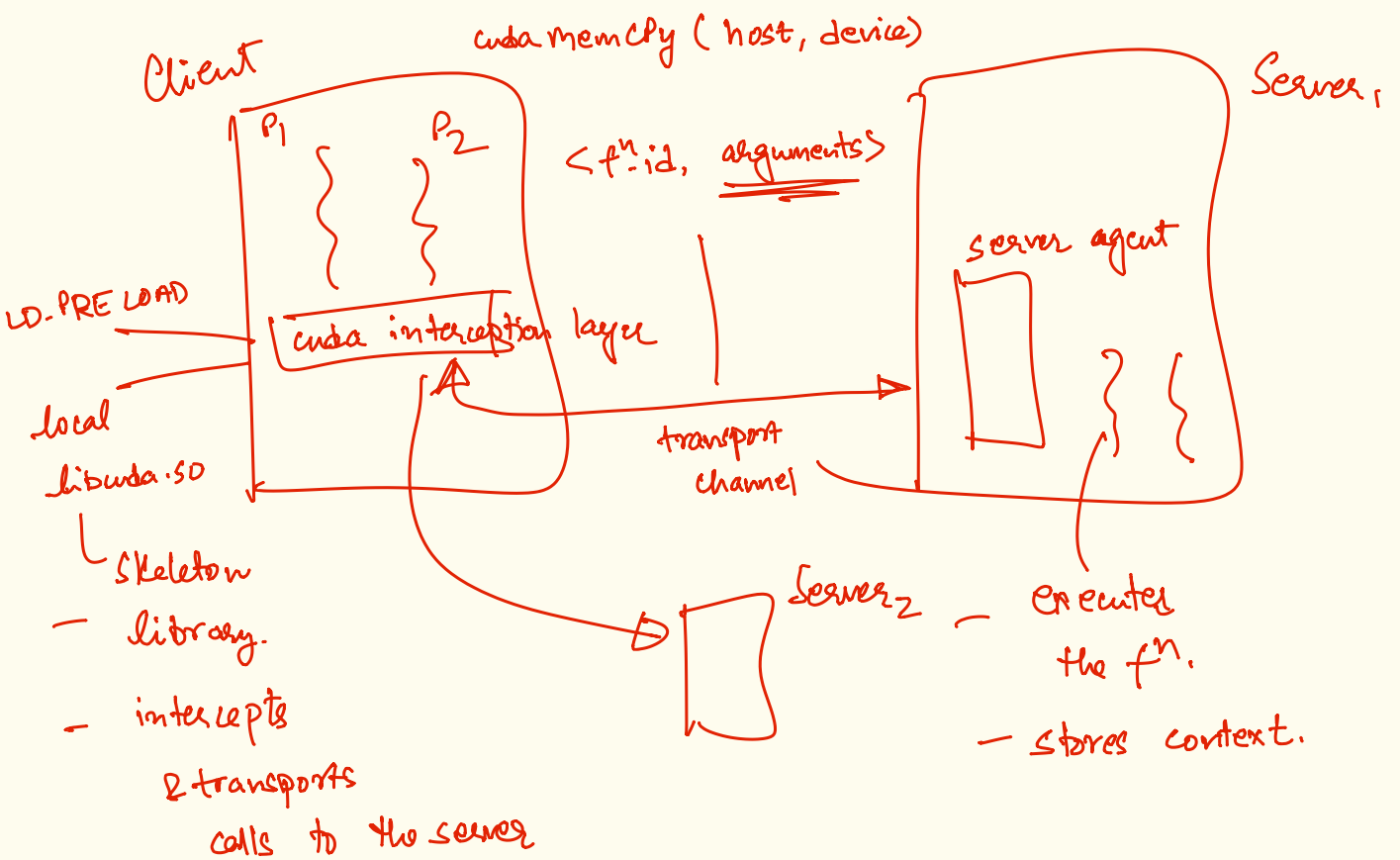
↳ per process GPU memory / cores limit / allocation.

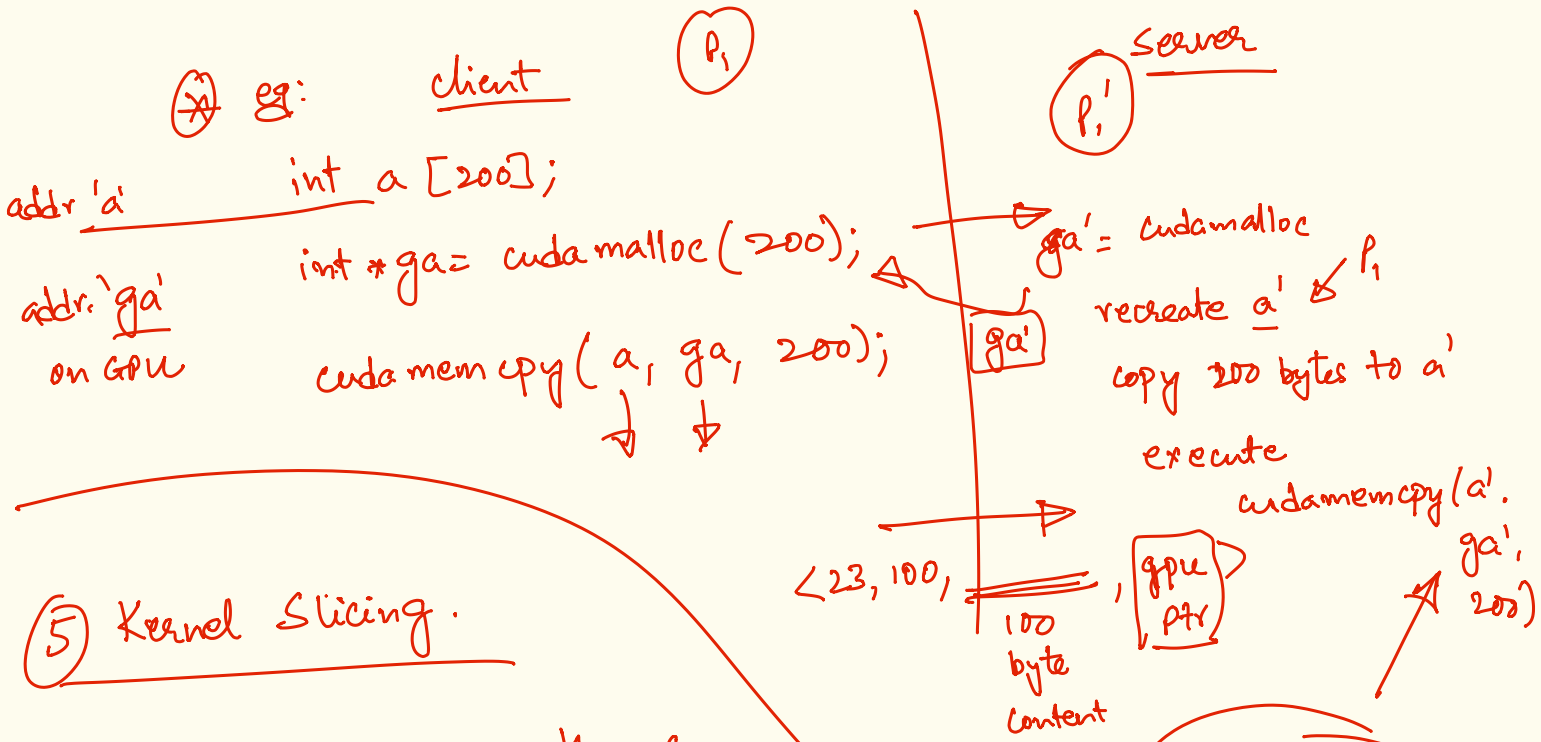
(gpu driver-assisted OS/VS virtualization)

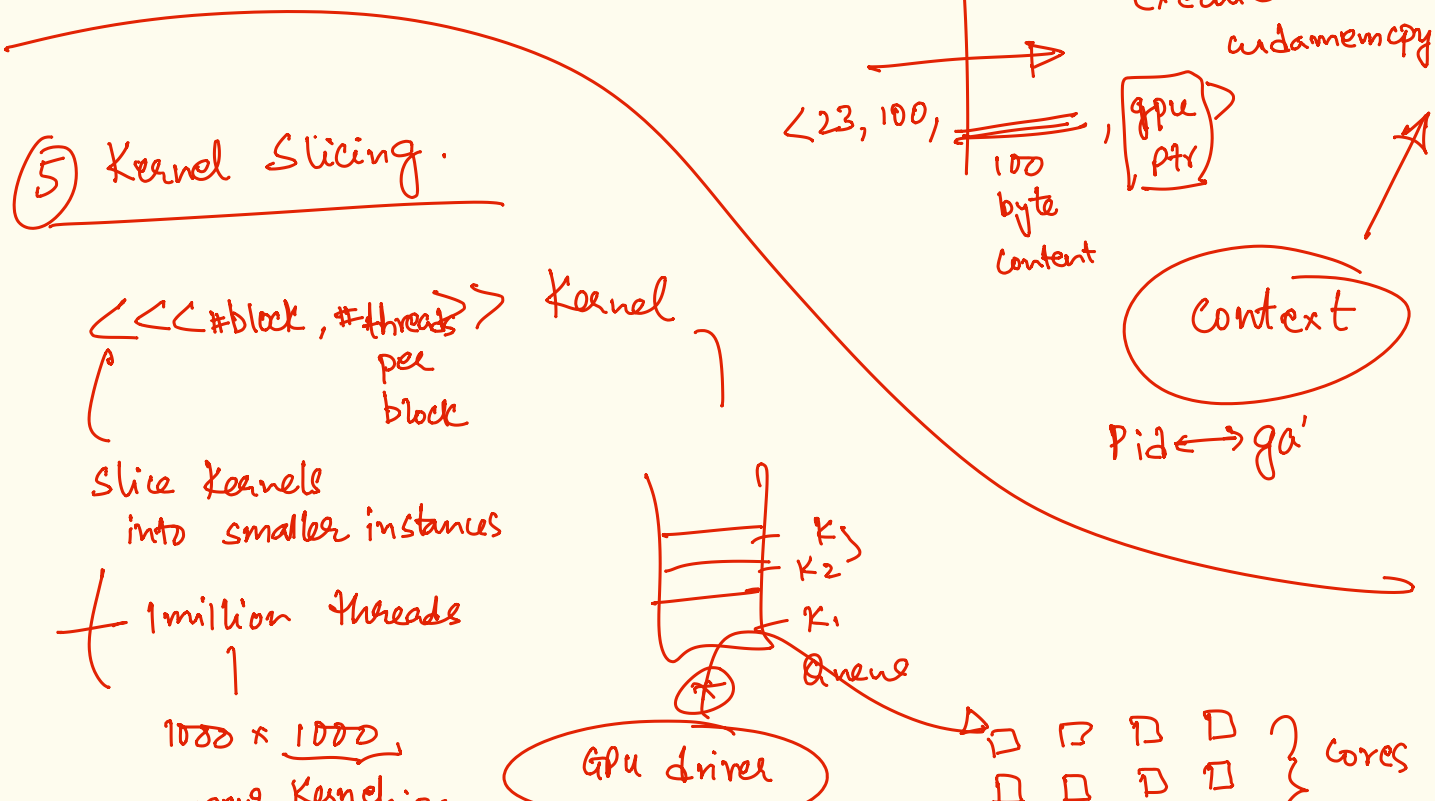
④ API virtualization / remoting

- RPC for GPUs
  - intercept CUDA calls

rcuda
vcuda
qcuda
vmcuda

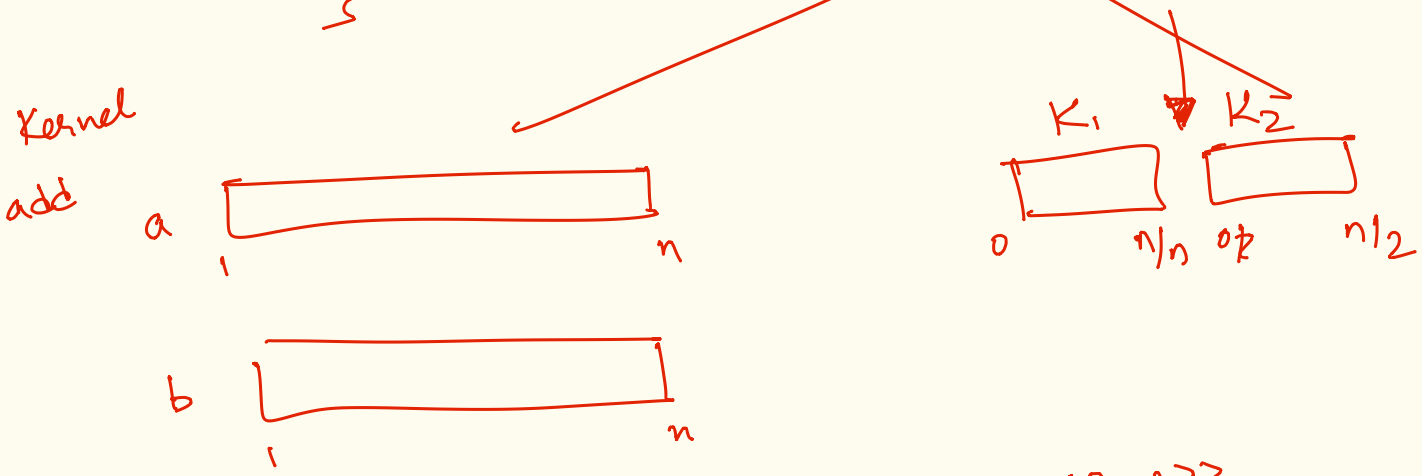

Client
cudamemcpy (host, device)
Server
P1
P2
<fn_id, arguments>
server agent
LD_PRELOAD
cuda interception layer
local
libcuda.so
transport
channel
skeleton
- library.
- intercepts & transports calls to the server
Server2
- executes the fn.
- stores context.


P1
Server
P1'
eg: client
addr 'a'
int a [200];
addr 'ga' on GPU
int *ga = cudamalloc (200);
cudamemcpy (a, ga, 200);
ga' = cudamalloc
recreate a'
P1
copy 200 bytes to a'
execute cudamemcpy(a', ga', 200)
ga'
<23, 100,
gpu ptr
100 byte content
context
Pid ↔ ga'


5 Kernel Slicing.
<<<#block, #threads per block>>> Kernel
Slice Kernels into smaller instances
1 million threads
1000 x 1000
new Kernel size
K3
K2
K1
Queue
GPU driver
cores

(5) slow down kernel threads per invocation
& inject artificial pre-emption at end of each mini-kernel.

```
for( i=0 ; i< # kernels) , i++) {
    << new # blocks, new tpb >> kernel
    slicedid ++;
}
```



<<< 1, 10^6 >> add

= a[tid] + b[tid]

= tid = blockid * tpb + threadid;

<<2, 2>>

blk=0 tid=0
1

blk=1 tid=0
1

(6) FaaS

└ decouple fn from fn execution

- GPU functions
- containers using GPUs on a server

unikernels ——— VMs ——— isolation.

performance. { containers, processes }

- Kernel, the libraries is usually very bloated.
- optimized, lean — Kernel + stock set of libraries & tools for the target appn.
- tradeoffs — performance & generalized runtime/execution.
- how to build this?
  - figure out dependencies & multiple abstractions | enforce them on the subsystems.

~ My VM is lighter & smaller than your Container. - SOSP 2017

- objdump
- Busybox / debootstrap Debian } Firecracker VM

Kata Containers

microVMs

ldd