Nvshare

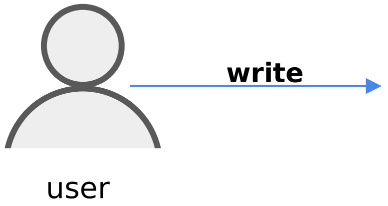
A Kubernetes Use Case

George Alexopoulos <giorgosalexo0@gmail.com>

Interactive environment

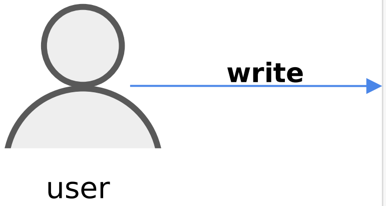
Interactive environment

Interactive environment



Interactive environment

Browser

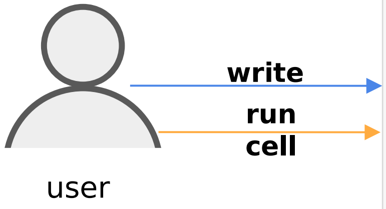



```
In [ ]: # Initialize denominator
        k = 1

        # Initialize sum
        s = 0

        for i in range(1000000):

            # even index elements are positive
            if i % 2 == 0:
                s += 4/k
            else:
                # odd index elements are negative
                s -= 4/k

            # denominator is odd
            k += 2

        print("pi =",s)
```

```
In [ ]: print("Do something else.")
```

Interactive environment

Browse
r

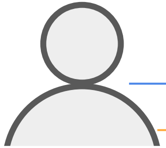


In [ ]:
```python
# Initialize denominator
k = 1

# Initialize sum
s = 0

for i in range(1000000):

    # even index elements are positive
    if i % 2 == 0:
        s += 4/k
    else:
        # odd index elements are negative
        s -= 4/k

    # denominator is odd
    k += 2

print("pi =",s)
```

In [ ]:
```python
print("Do something else.")
```

Interactive environment

Browser



user

write

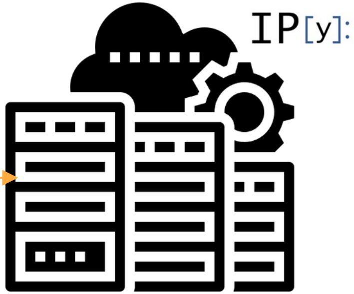
run cell

```
In [*]: # Initialize denominator
        k = 1

        # Initialize sum
        s = 0

        for i in range(1000000):

            # even index elements are positive
            if i % 2 == 0:
                s += 4/k
            else:
                # odd index elements are negative
                s -= 4/k

            # denominator is odd
            k += 2

        print("pi =",s)
```

```
In [ ]: print("Do something else.")
```

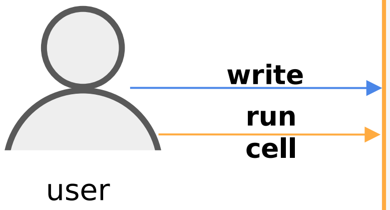
send cell code

backend-IPython



Interactive environment

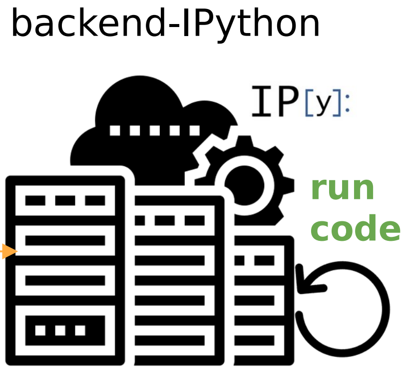
Browser




```
In [*]: # Initialize denominator
        k = 1

        # Initialize sum
        s = 0

        for i in range(1000000):

            # even index elements are positive
            if i % 2 == 0:
                s += 4/k
            else:
                # odd index elements are negative
                s -= 4/k

            # denominator is odd
            k += 2

        print("pi =",s)
```

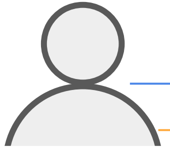
```
In [ ]: print("Do something else.")
```

send cell code



Interactive environment

Browser



write

run cell

In [*]:
```python
# Initialize denominator
k = 1

# Initialize sum
s = 0

for i in range(1000000):

    # even index elements are positive
    if i % 2 == 0:
        s += 4/k
    else:
        # odd index elements are negative
        s -= 4/k

    # denominator is odd
    k += 2

print("pi =",s)
```

In [ ]:
```python
print("Do something else.")
```

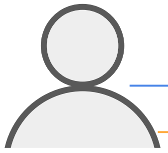
send cell code

receive result

backend-IPython

Interactive environment

Browse



write

run cell

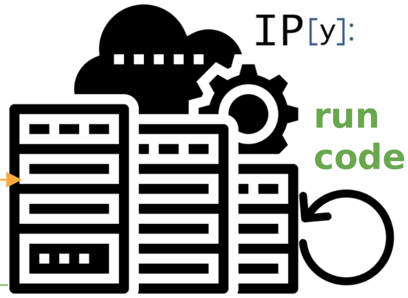
user

```
In [1]: # Initialize denominator
        k = 1

        # Initialize sum
        s = 0

        for i in range(1000000):

            # even index elements are positive
            if i % 2 == 0:
                s += 4/k
            else:
                # odd index elements are negative
                s -= 4/k

            # denominator is odd
            k += 2

        print("pi =",s)

pi = 3.141591653589774З

In [ ]: print("Do something else.")
```

send cell code

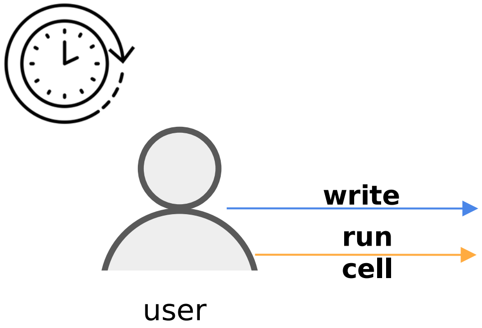
receive result

backend-IPython



Interactive environment

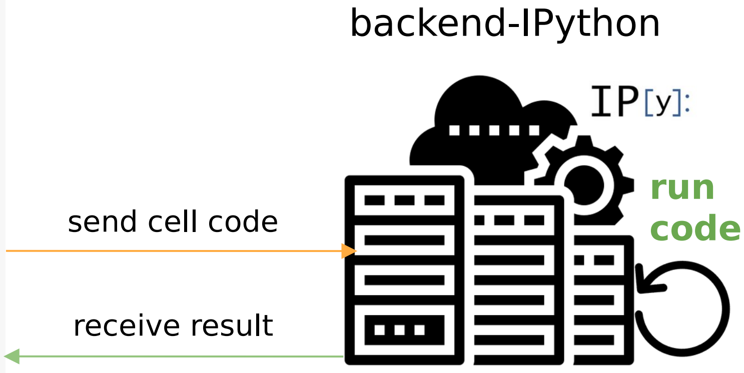
Browse




```
In [1]: # Initialize denominator
k = 1

# Initialize sum
s = 0

for i in range(1000000):

    # even index elements are positive
    if i % 2 == 0:
        s += 4/k
    else:
        # odd index elements are negative
        s -= 4/k

    # denominator is odd
    k += 2

print("pi =",s)
```

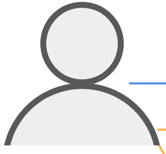
```
pi = 3.1415916535897743
```

```
In [ ]: print("Do something else.")
```



Interactive environment

Browse



user

write

run cell

In [1]:

```
# Initialize denominator
k = 1

# Initialize sum
s = 0

for i in range(1000000):

    # even index elements are positive
    if i % 2 == 0:
        s += 4/k
    else:
        # odd index elements are negative
        s -= 4/k

    # denominator is odd
    k += 2

print("pi =",s)
```

```
pi = 3.1415916535897743
```

run cell

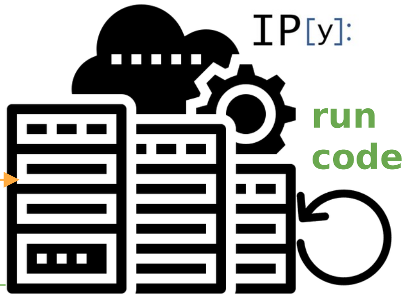
In [ ]:

```
print("Do something else.")
```

send cell code

receive result

backend-IPython



Interactive environment

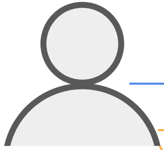
Browse

```
In [1]: # Initialize denominator
        k = 1

        # Initialize sum
        s = 0

        for i in range(1000000):

            # even index elements are positive
            if i % 2 == 0:
                s += 4/k
            else:
                # odd index elements are negative
                s -= 4/k

            # denominator is odd
            k += 2

        print("pi =",s)
```

```
pi = 3.141591653589774З
```

```
In [ ]: print("Do something else.")
```

⋮



user

write

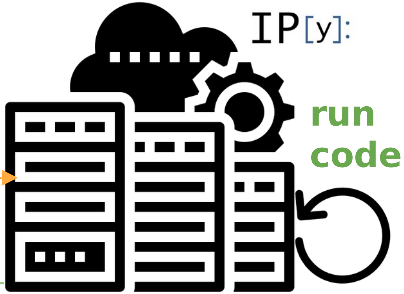
run cell

run cell

send cell code

receive result

backend-IPython



What is Kubeflow?

What is Kubeflow?

Kubeflow

What is Kubeflow?

Kubeflow =

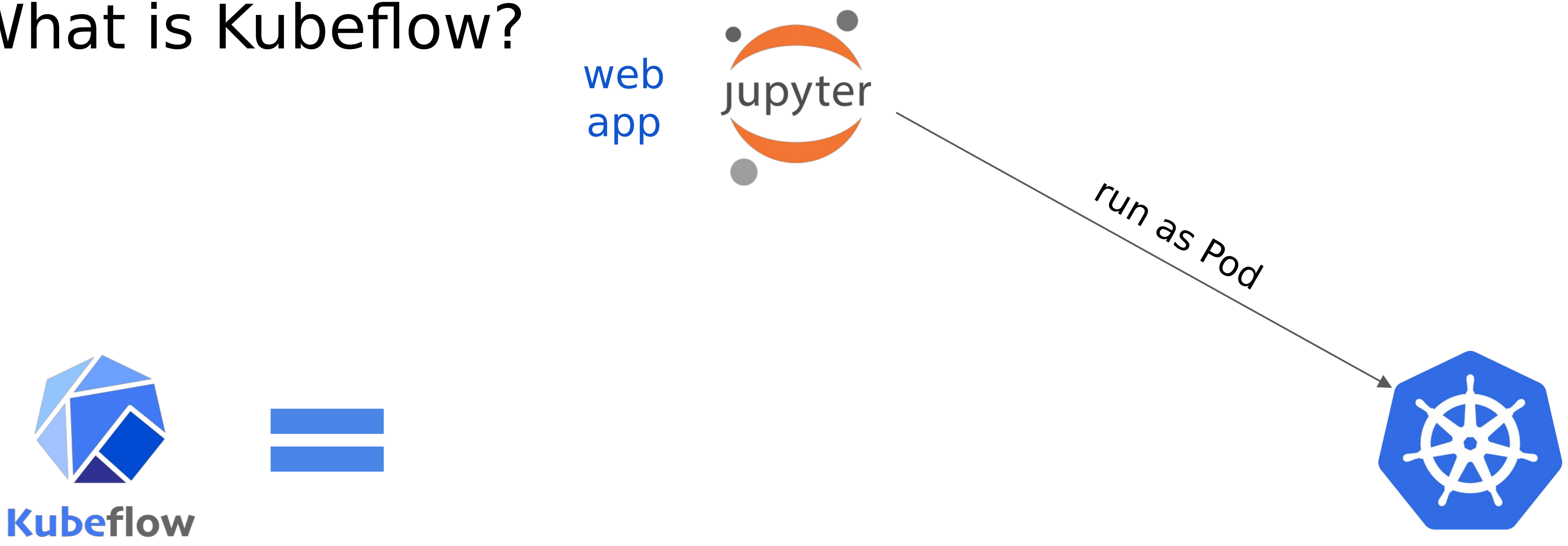
What is Kubeflow?

Kubeflow

What is Kubeflow?

web
app

What is Kubeflow?



What is Kubeflow?

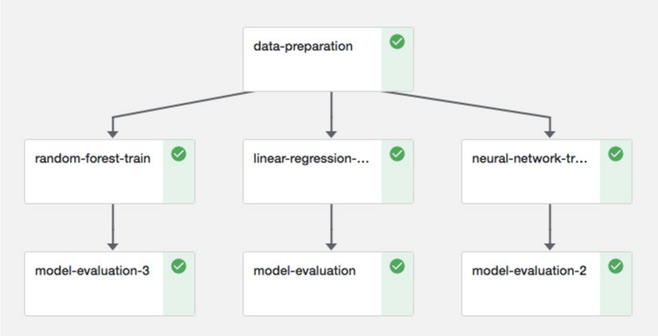
web
app

jupyter

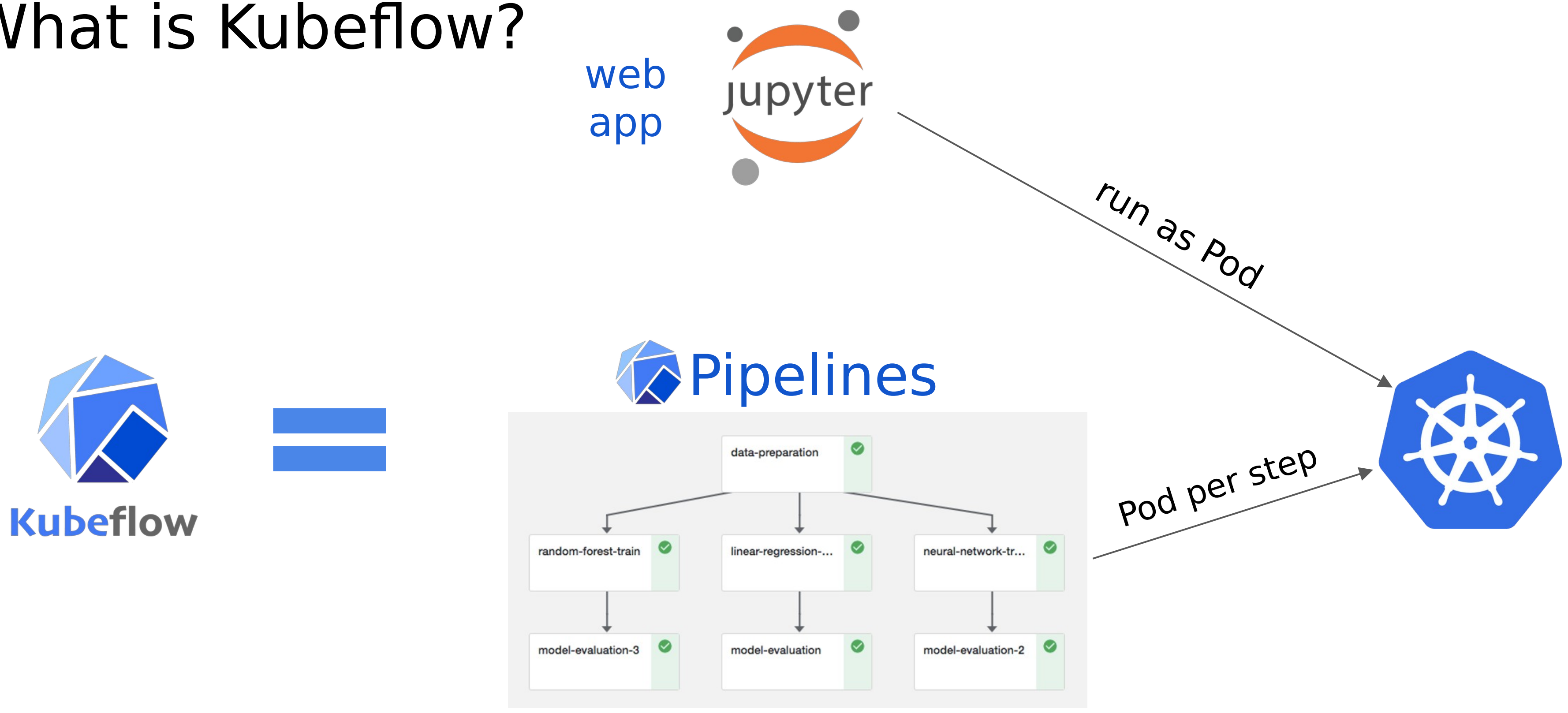
run as Pod

Kubeflow

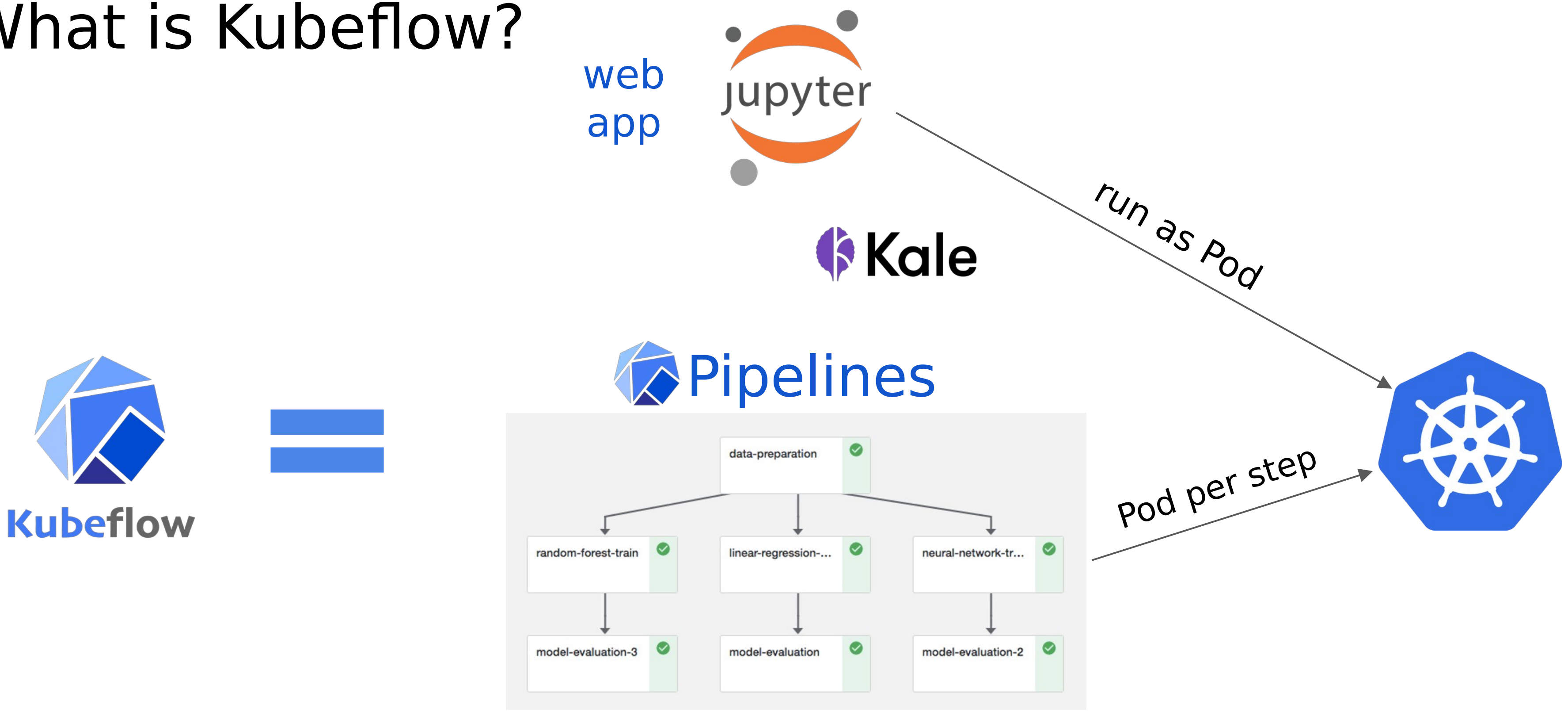
Pipelines



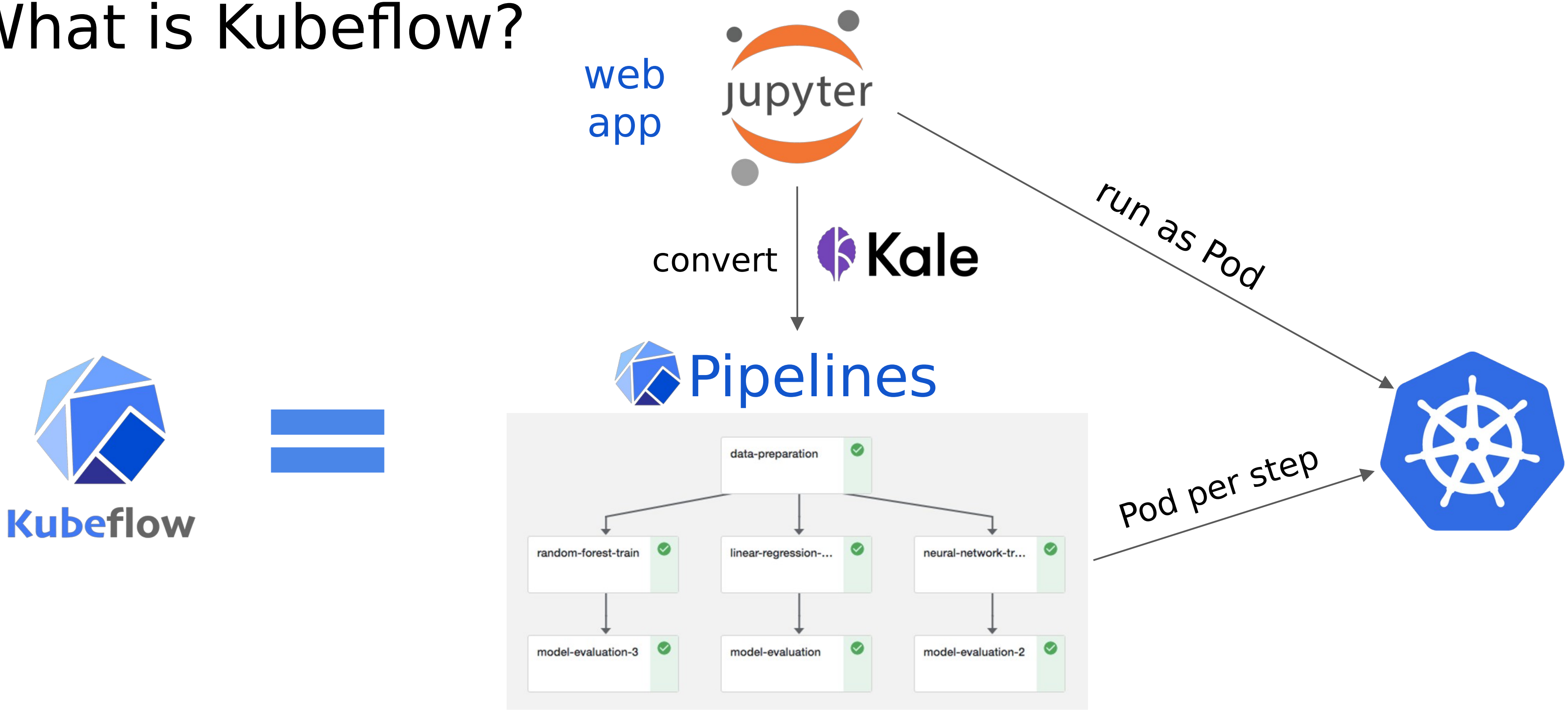
What is Kubeflow?



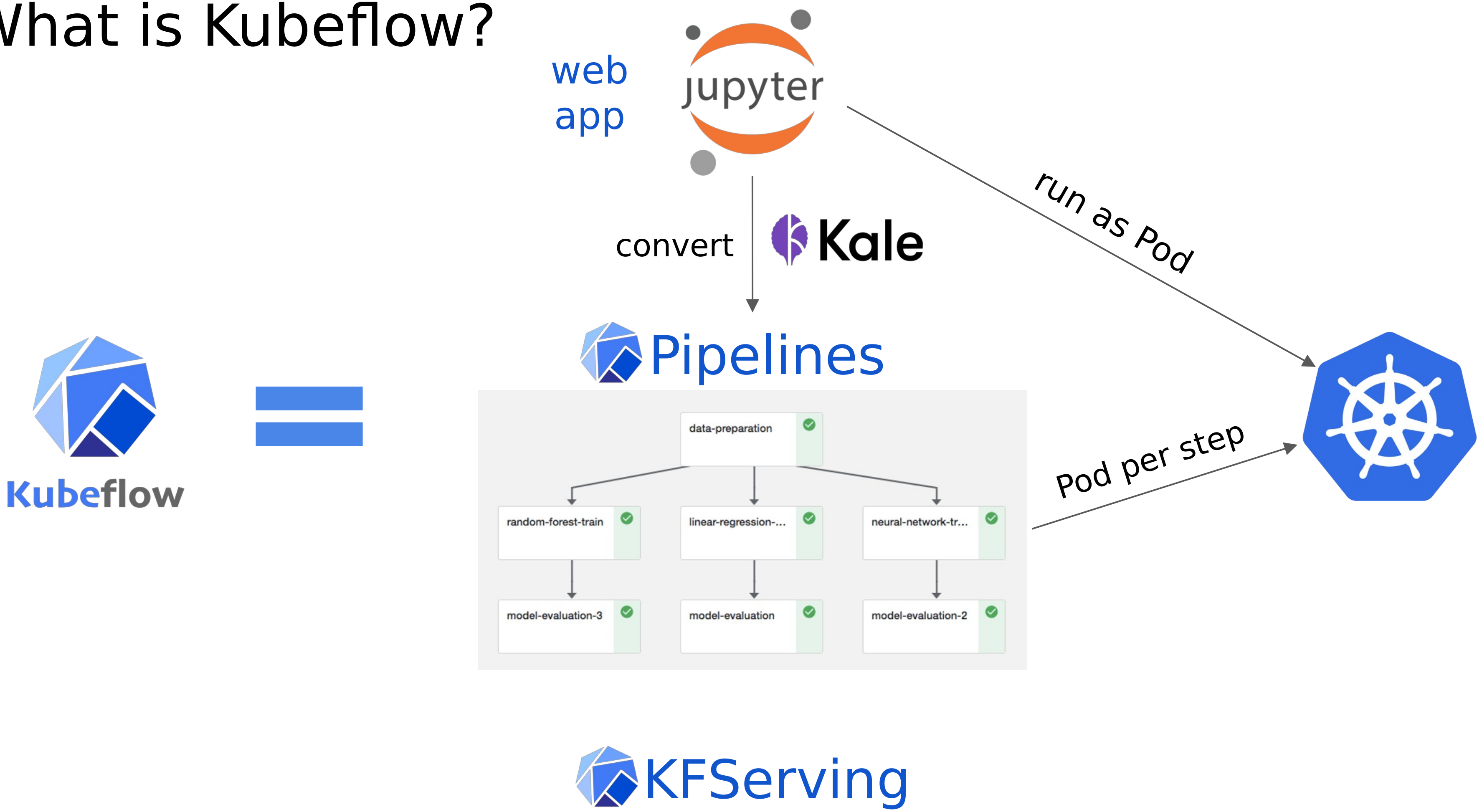
What is Kubeflow?



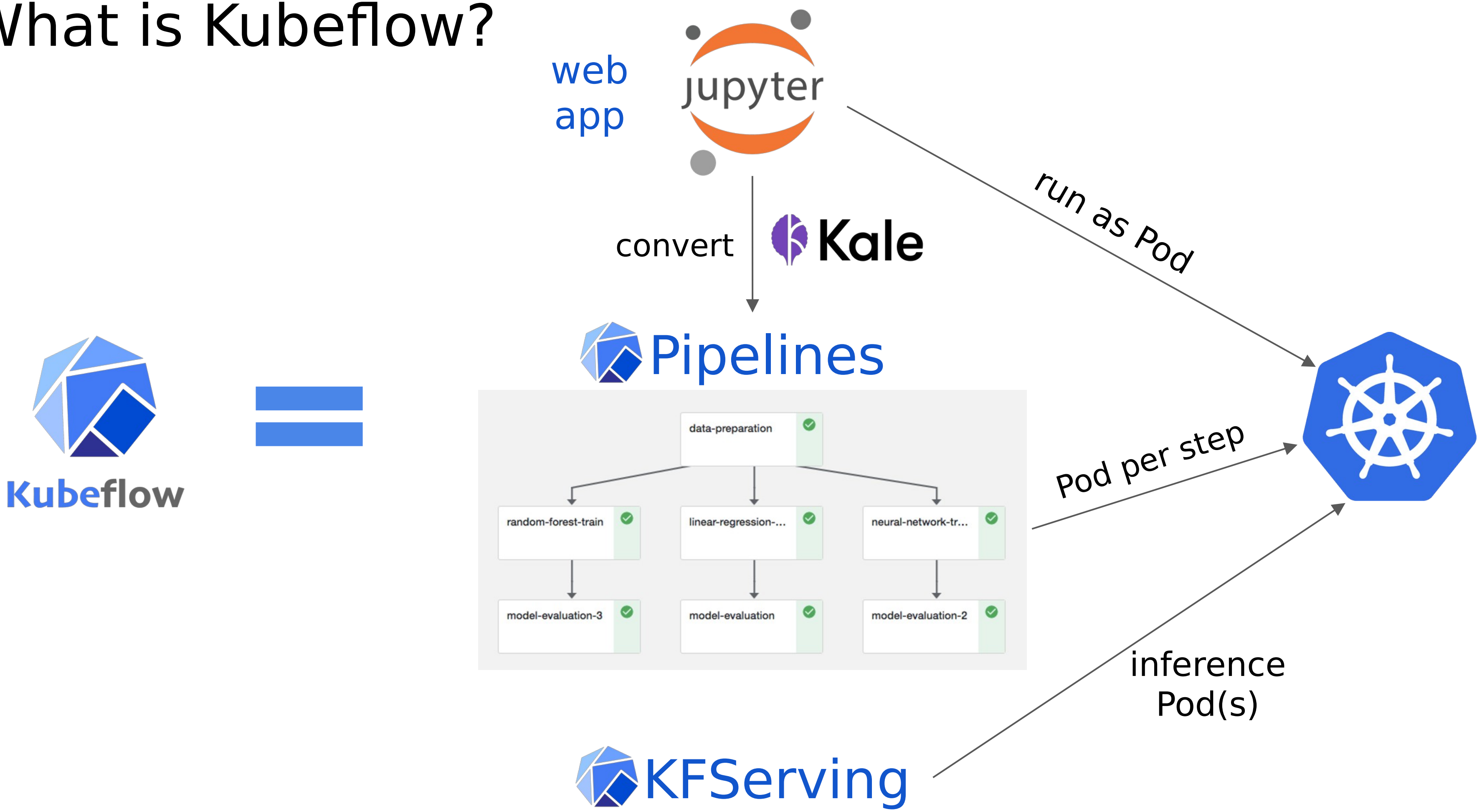
What is Kubeflow?



What is Kubeflow?



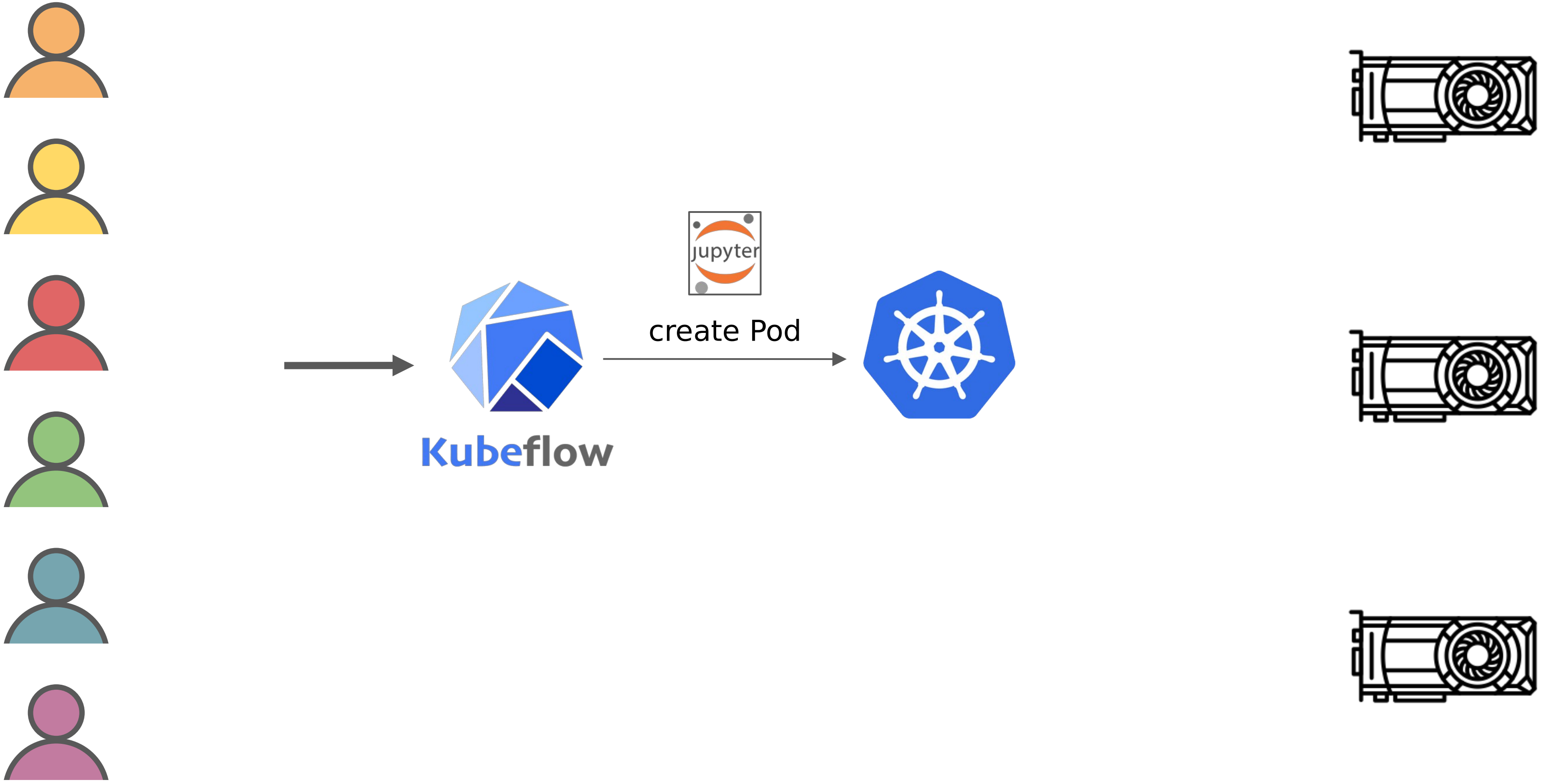
What is Kubeflow?



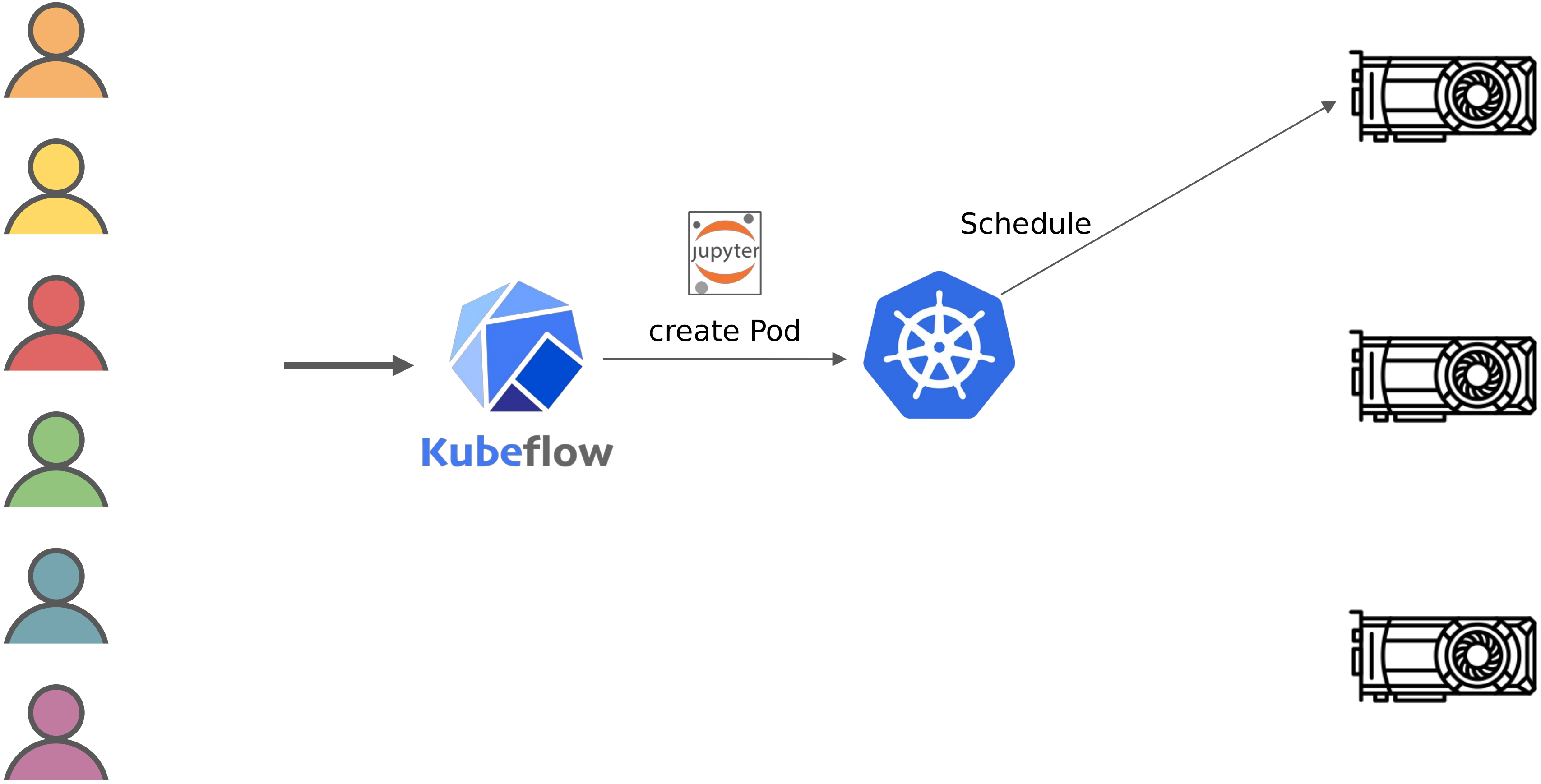
The Problem

The Problem

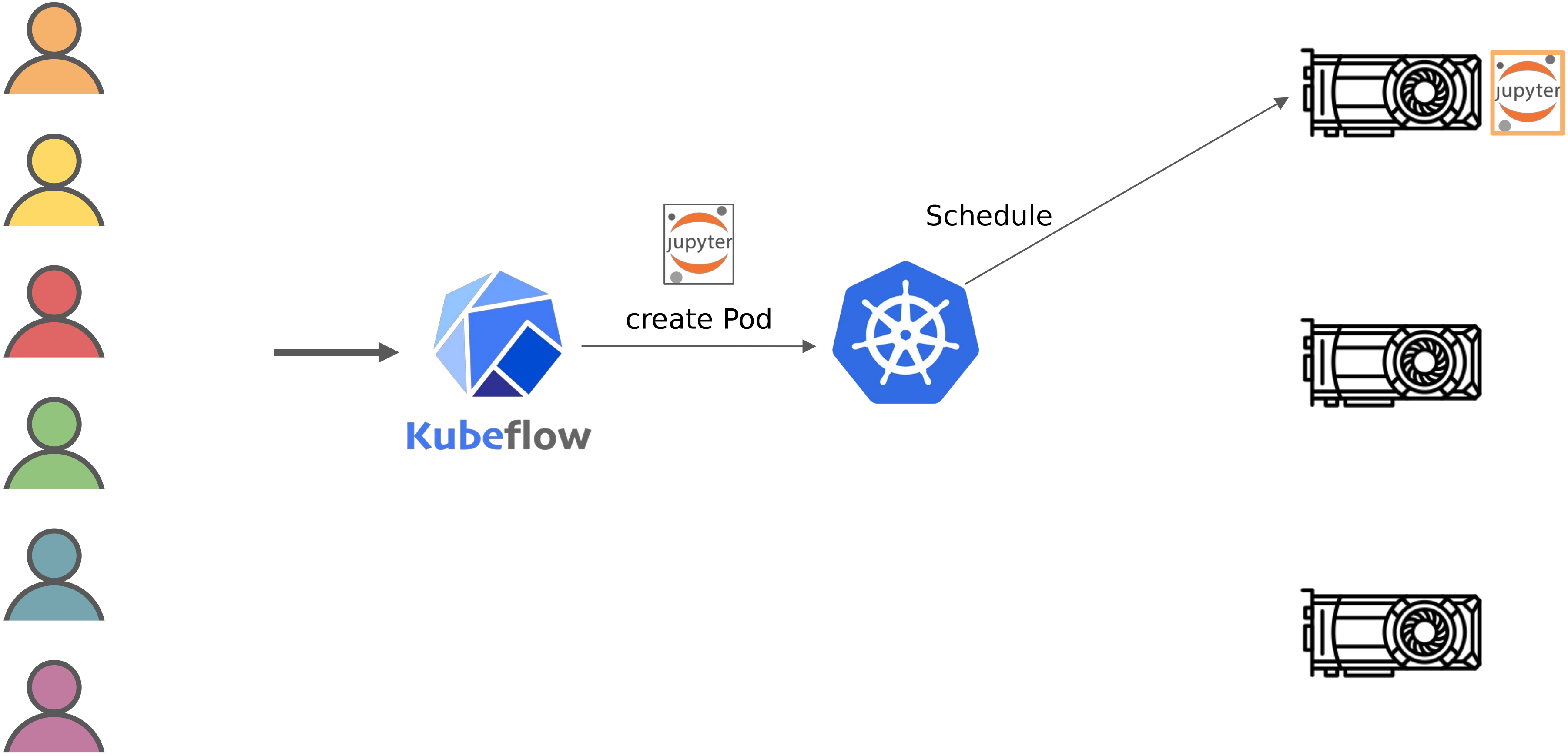
The Problem



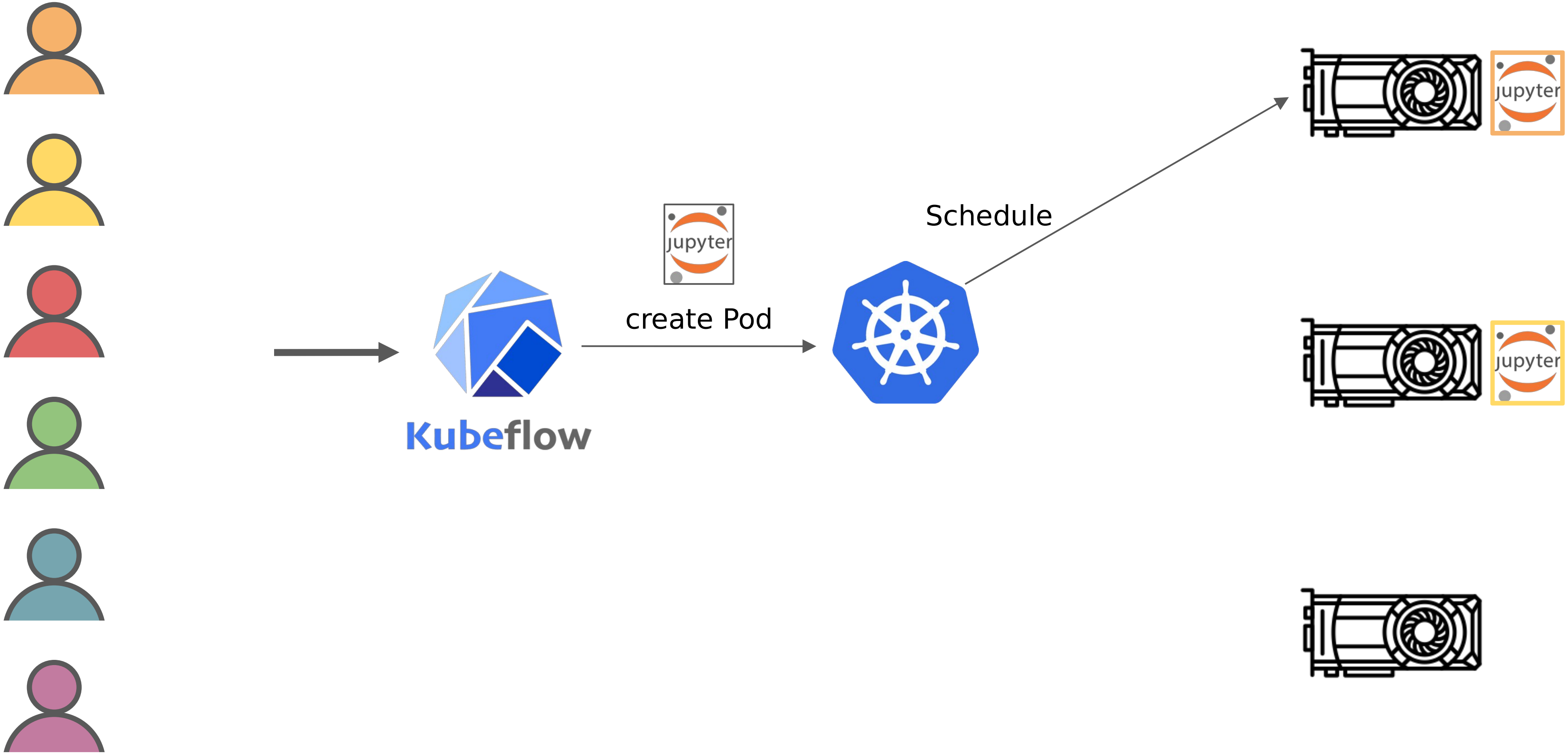
The Problem



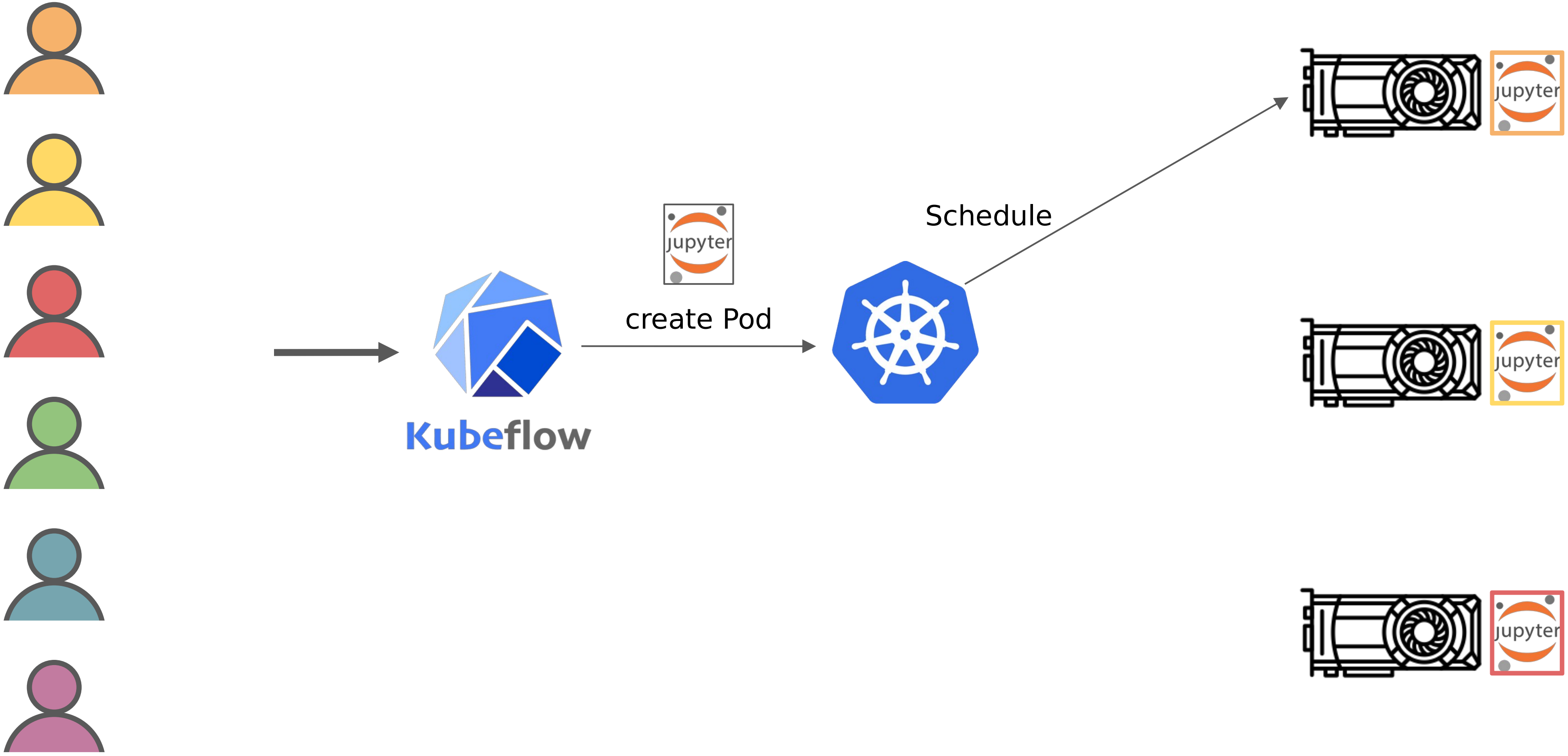
The Problem



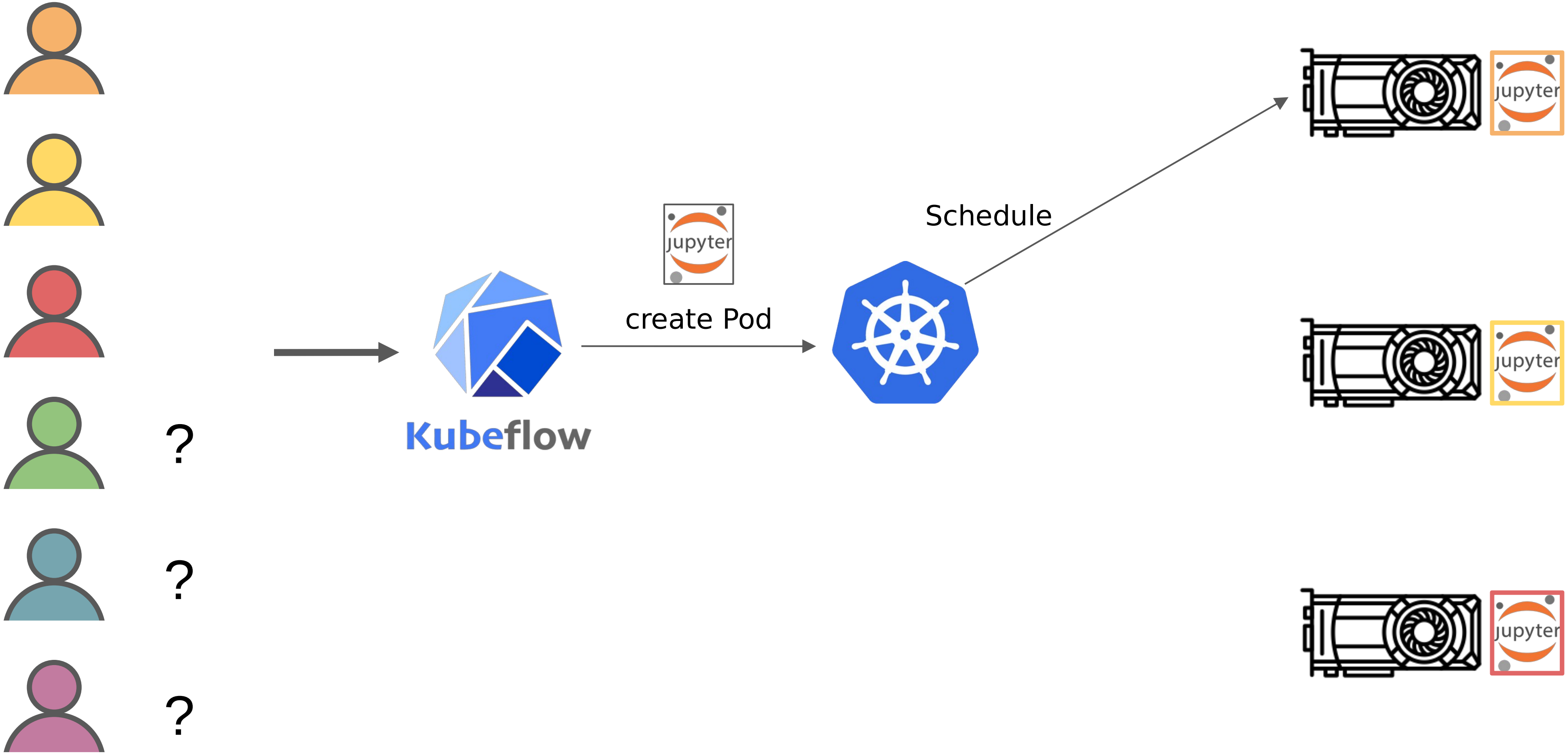
The Problem



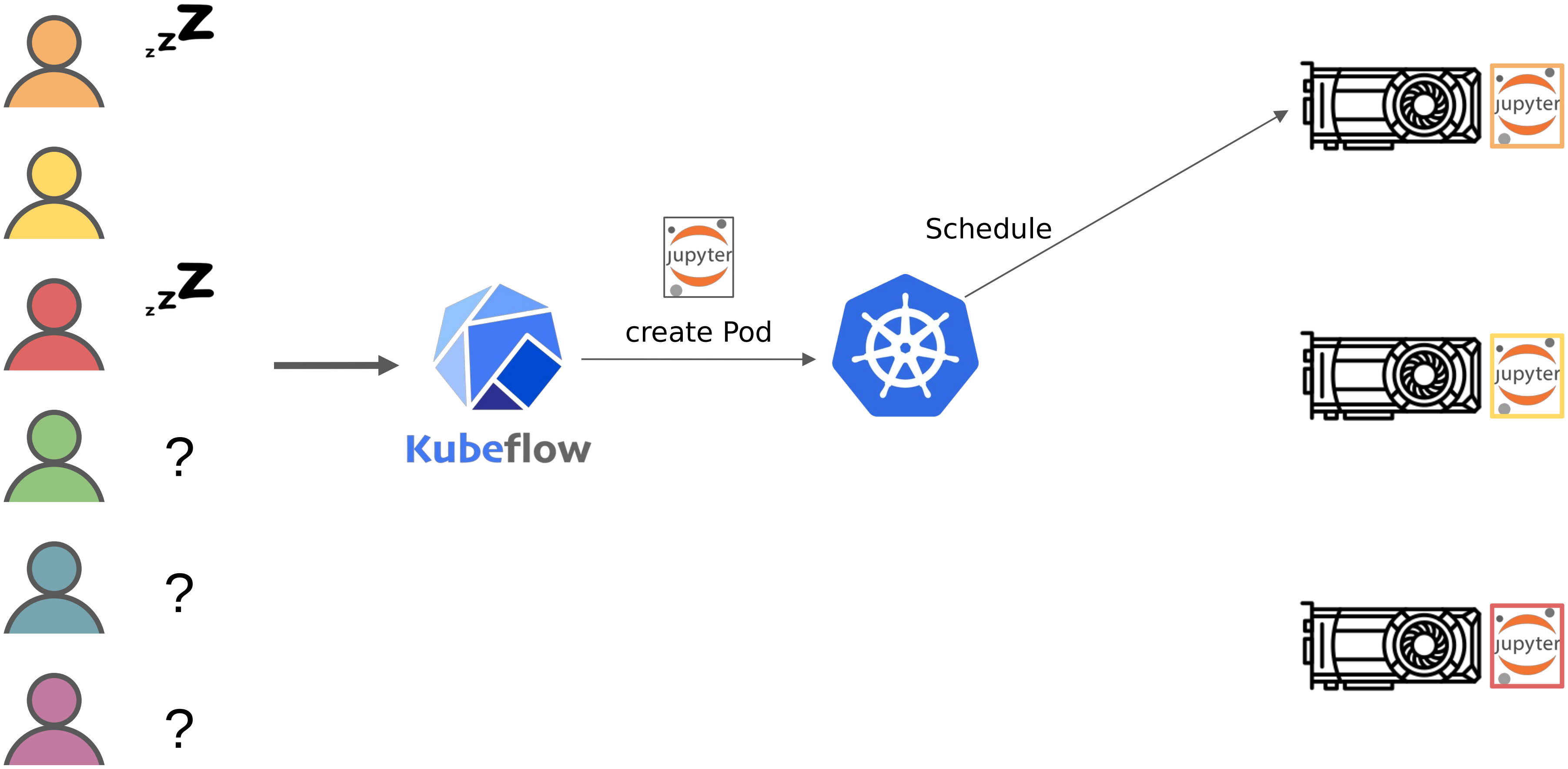
The Problem


jupyter
create Pod
Kubeflow
Schedule
jupyter
jupyter
jupyter

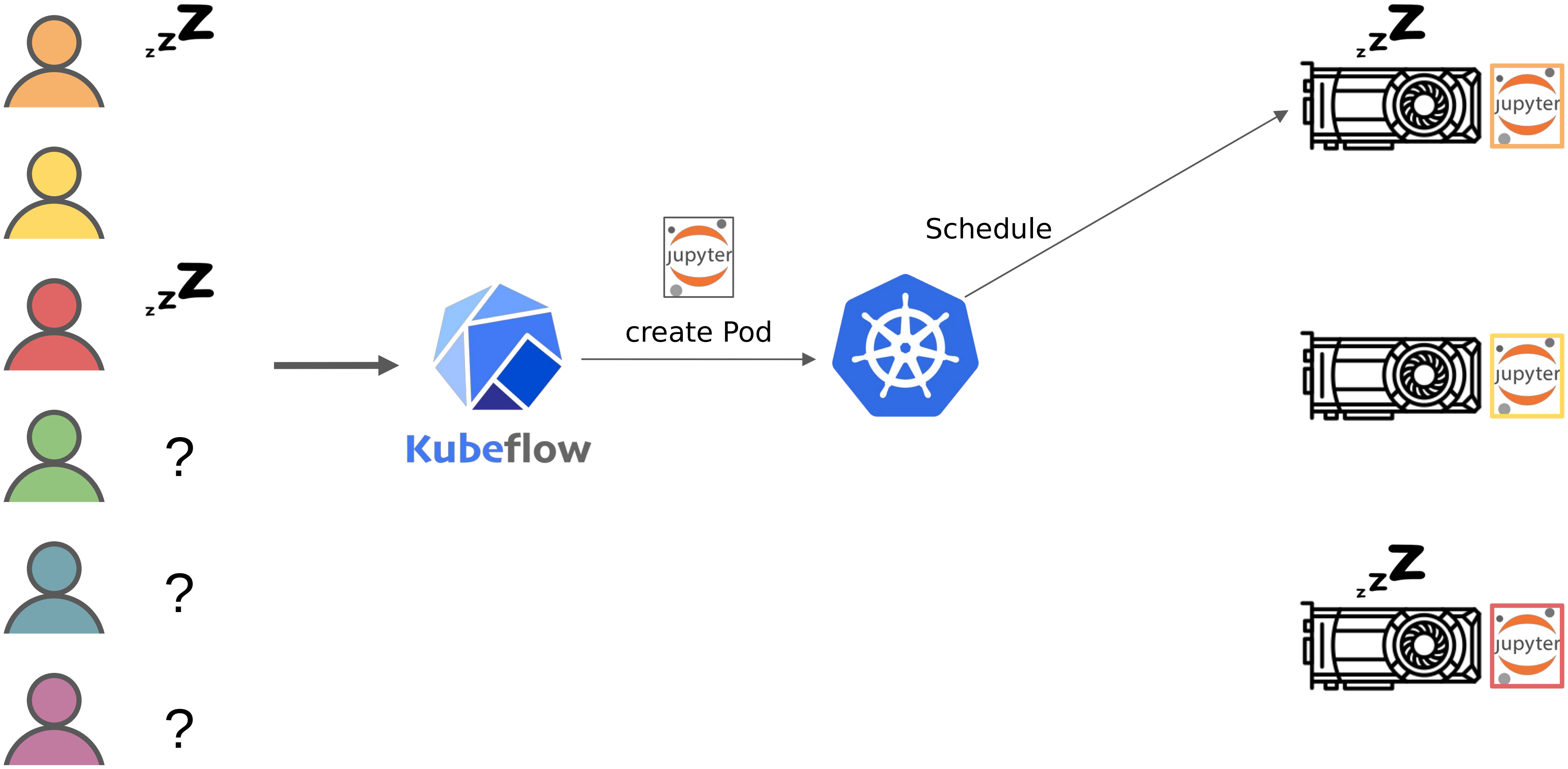
The Problem


?
?
?
Kubeflow
jupyter
create Pod
Schedule
jupyter
jupyter
jupyter

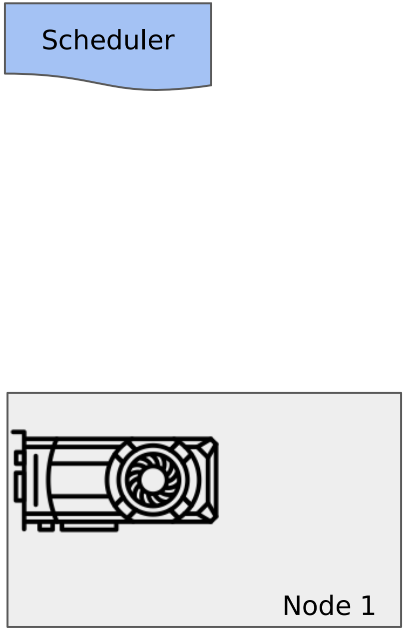
The Problem

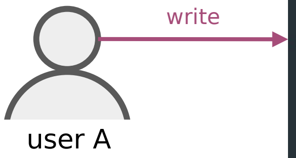

zzZ
zzZ
?
?
?
Kubeflow
jupyter
create Pod
Schedule
jupyter
jupyter
jupyter

The Problem



zzZ
zzZ
?
?
?
Kubeflow
jupyter
create Pod
Schedule
zzZ
jupyter
jupyter
zzZ
jupyter

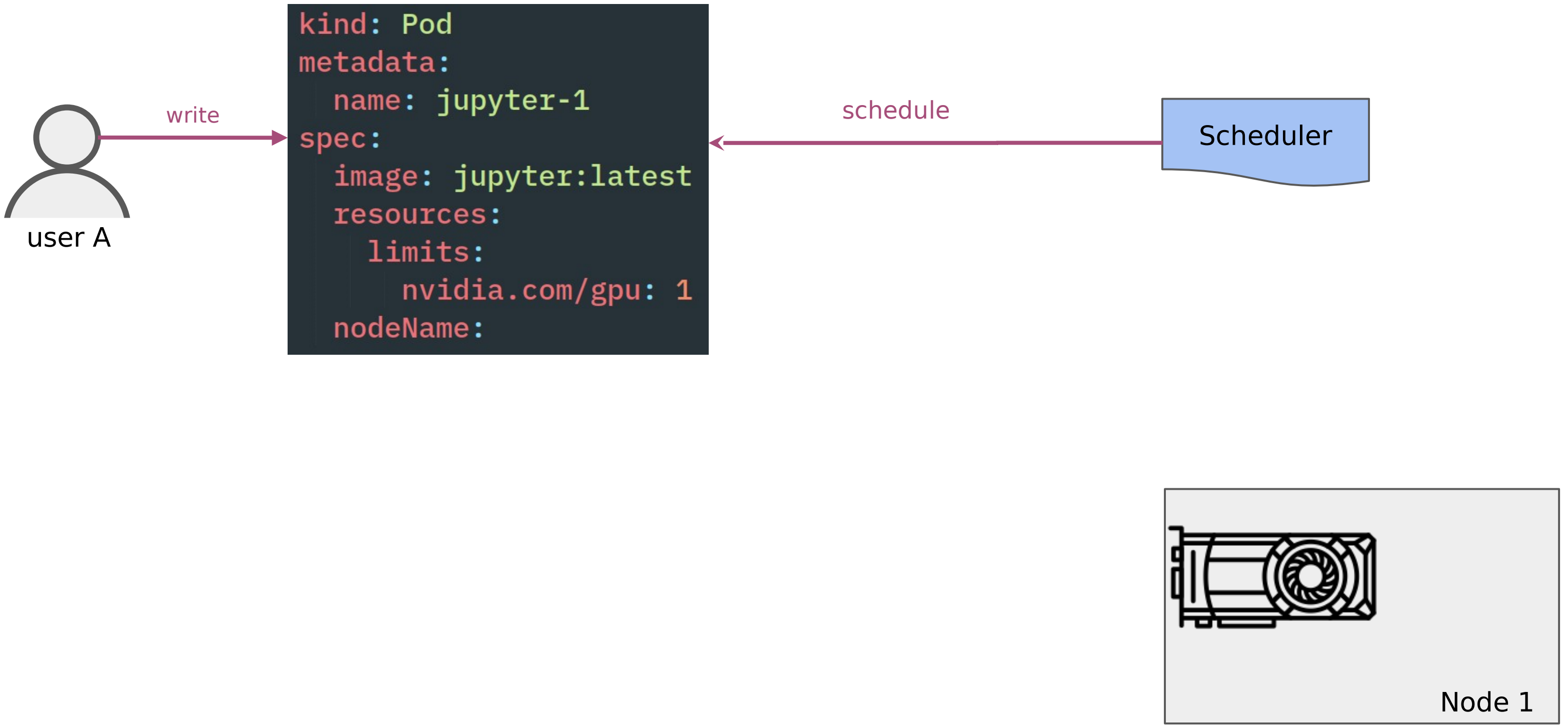

Scheduler
Node 1

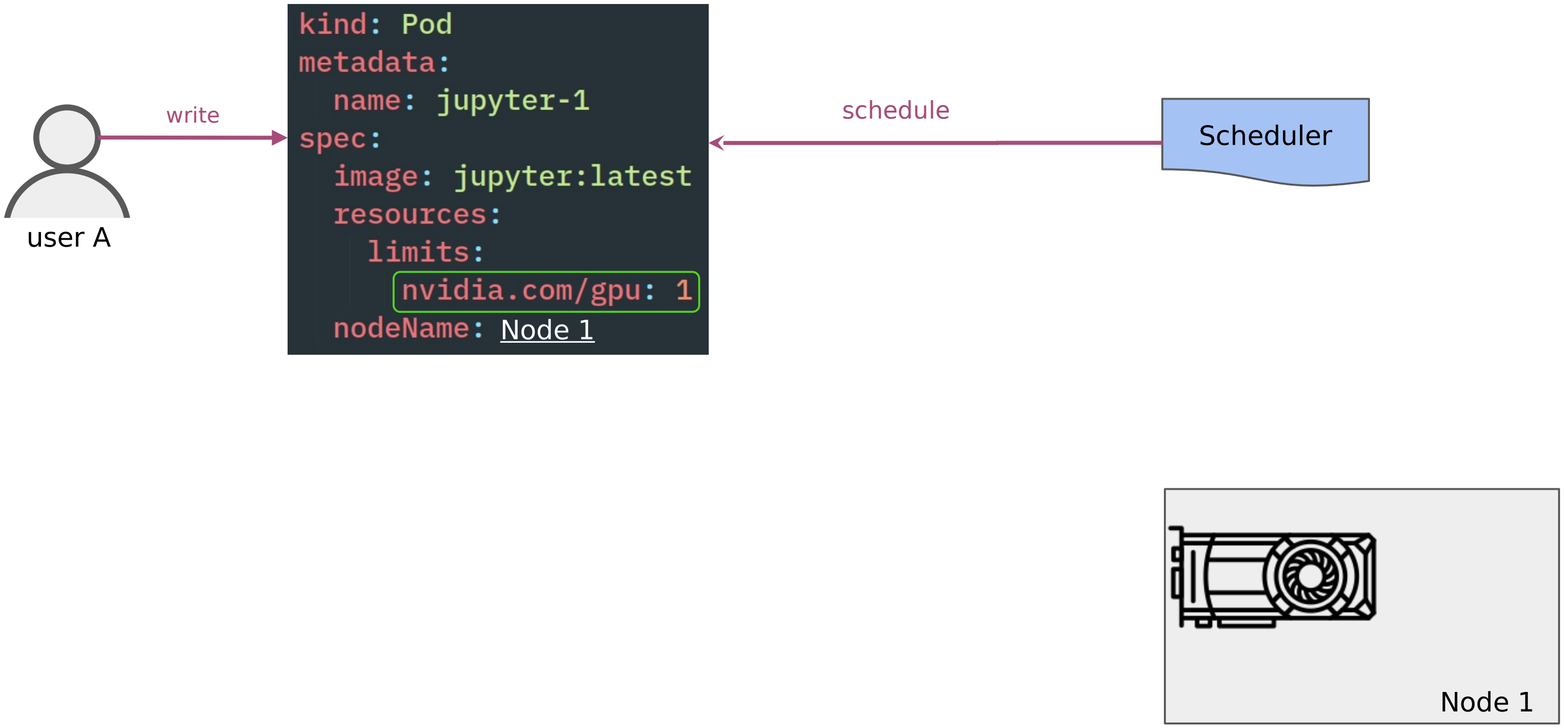


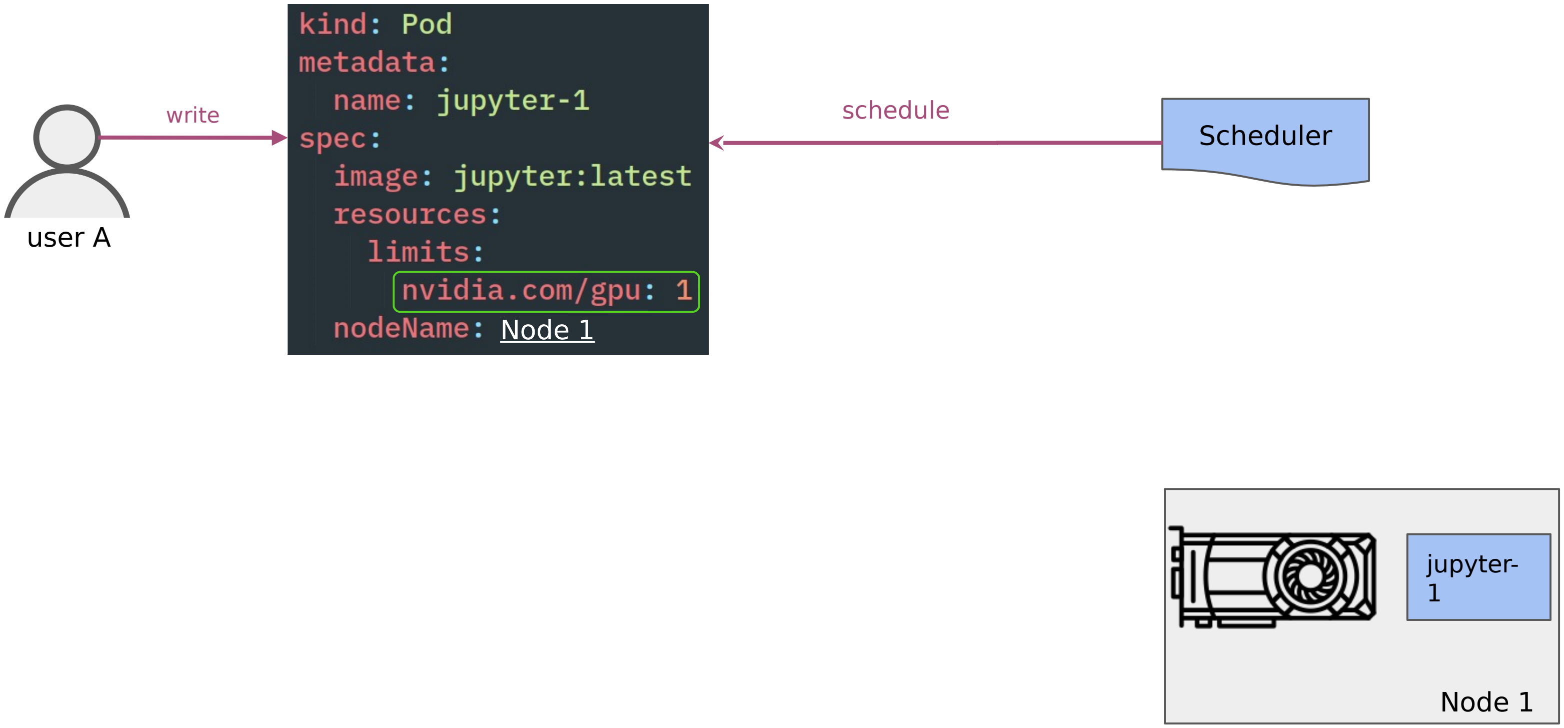

```yaml
kind: Pod
metadata:
  name: jupyter-1
spec:
  image: jupyter:latest
  resources:
    limits:
      nvidia.com/gpu: 1
  nodeName:
```

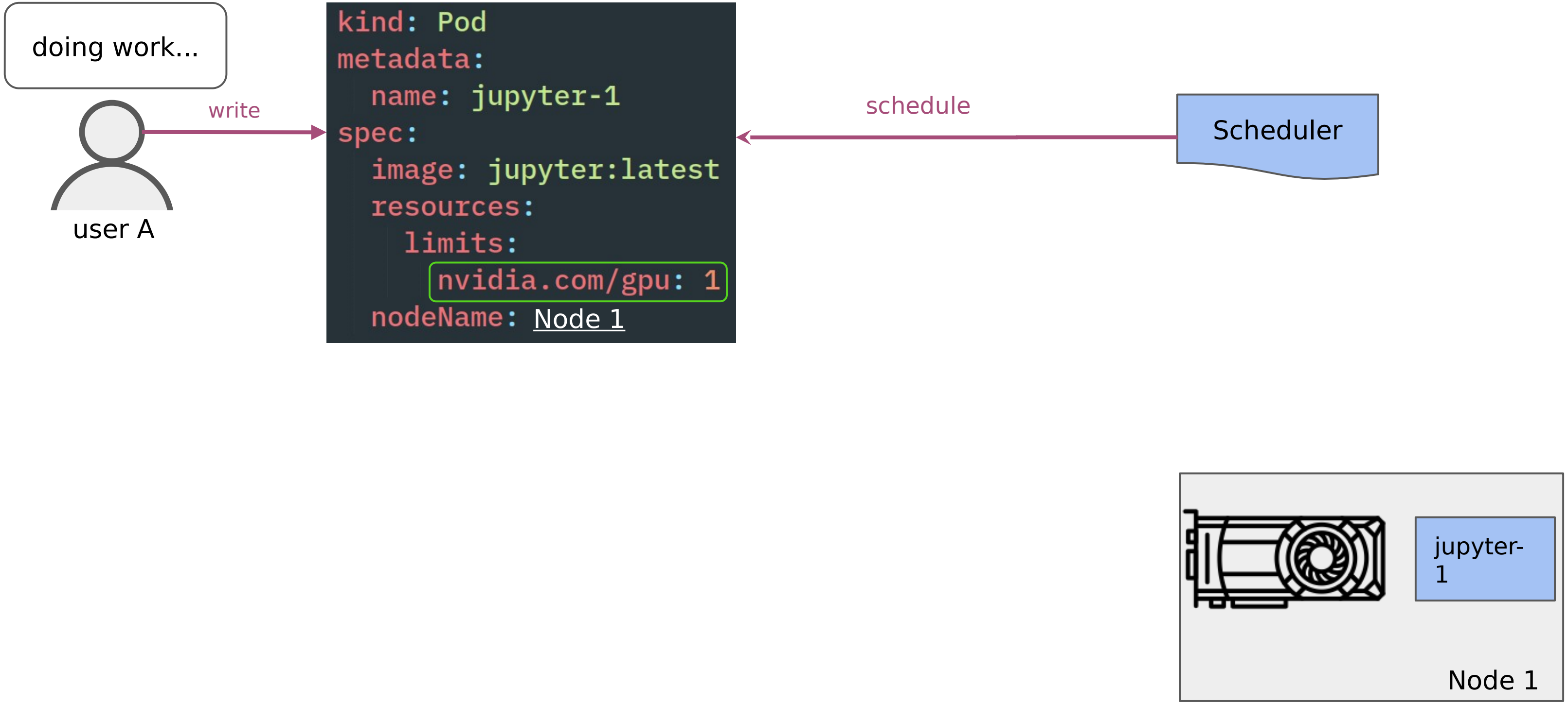
Scheduler

Node 1


write
user A
kind: Pod
metadata:
name: jupyter-1
spec:
image: jupyter:latest
resources:
limits:
nvidia.com/gpu: 1
nodeName:
schedule
Scheduler
Node 1


write
user A
kind: Pod
metadata:
name: jupyter-1
spec:
image: jupyter:latest
resources:
limits:
nvidia.com/gpu: 1
nodeName: Node 1
schedule
Scheduler
Node 1


write
user A
kind: Pod
metadata:
name: jupyter-1
spec:
image: jupyter:latest
resources:
limits:
nvidia.com/gpu: 1
nodeName: Node 1
schedule
Scheduler
jupyter-1
Node 1


doing work...
user A
write
kind: Pod
metadata:
name: jupyter-1
spec:
image: jupyter:latest
resources:
limits:
nvidia.com/gpu: 1
nodeName: Node 1
schedule
Scheduler
jupyter-1
Node 1

doing work...

user A

write

```yaml
kind: Pod
metadata:
  name: jupyter-1
spec:
  image: jupyter:latest
  resources:
    limits:
      nvidia.com/gpu: 1
  nodeName: Node 1
```

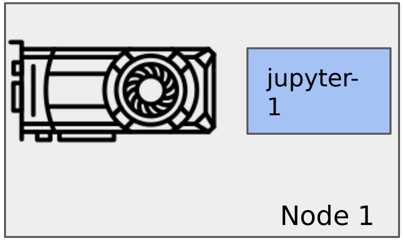
schedule

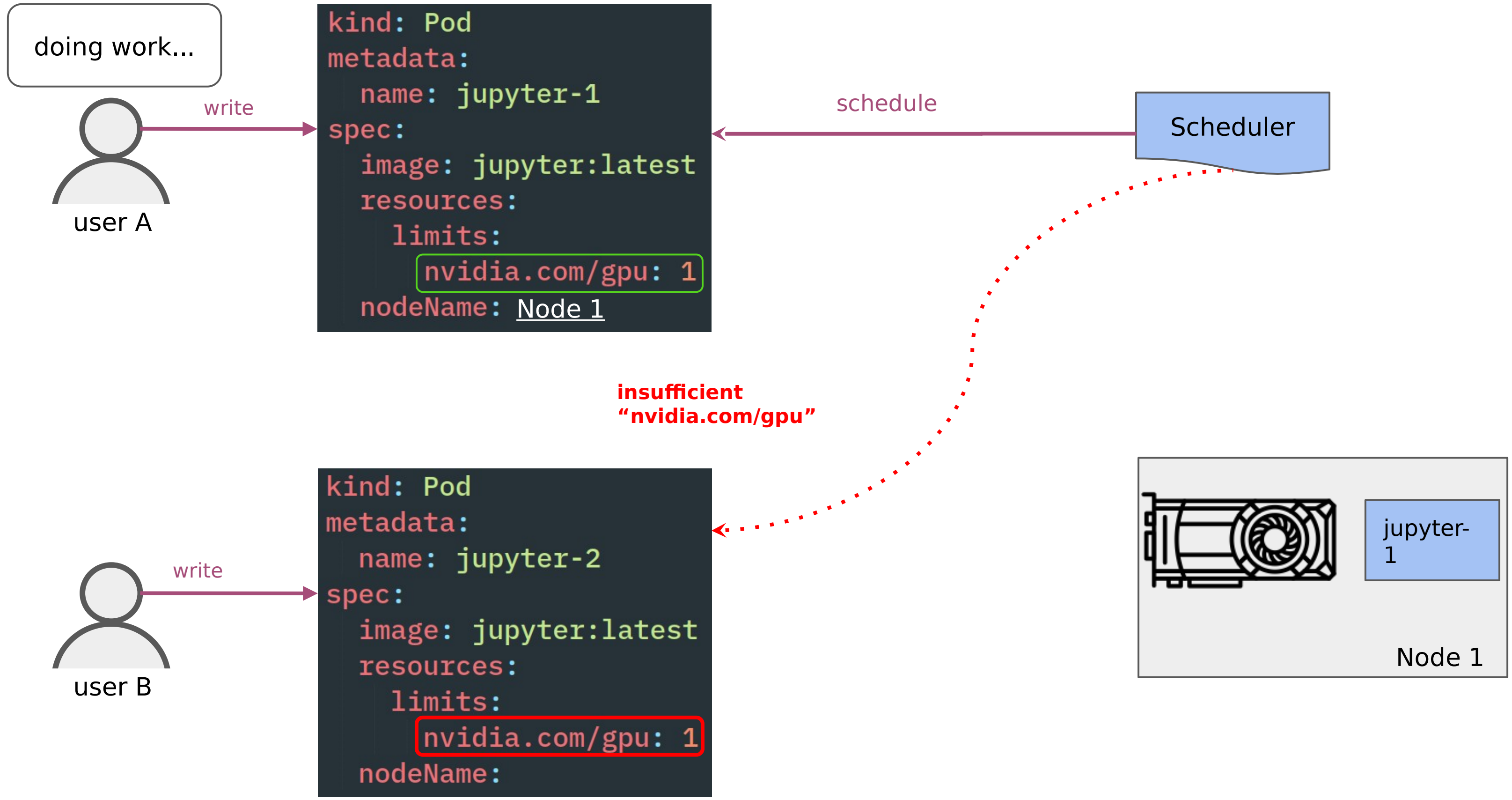
Scheduler

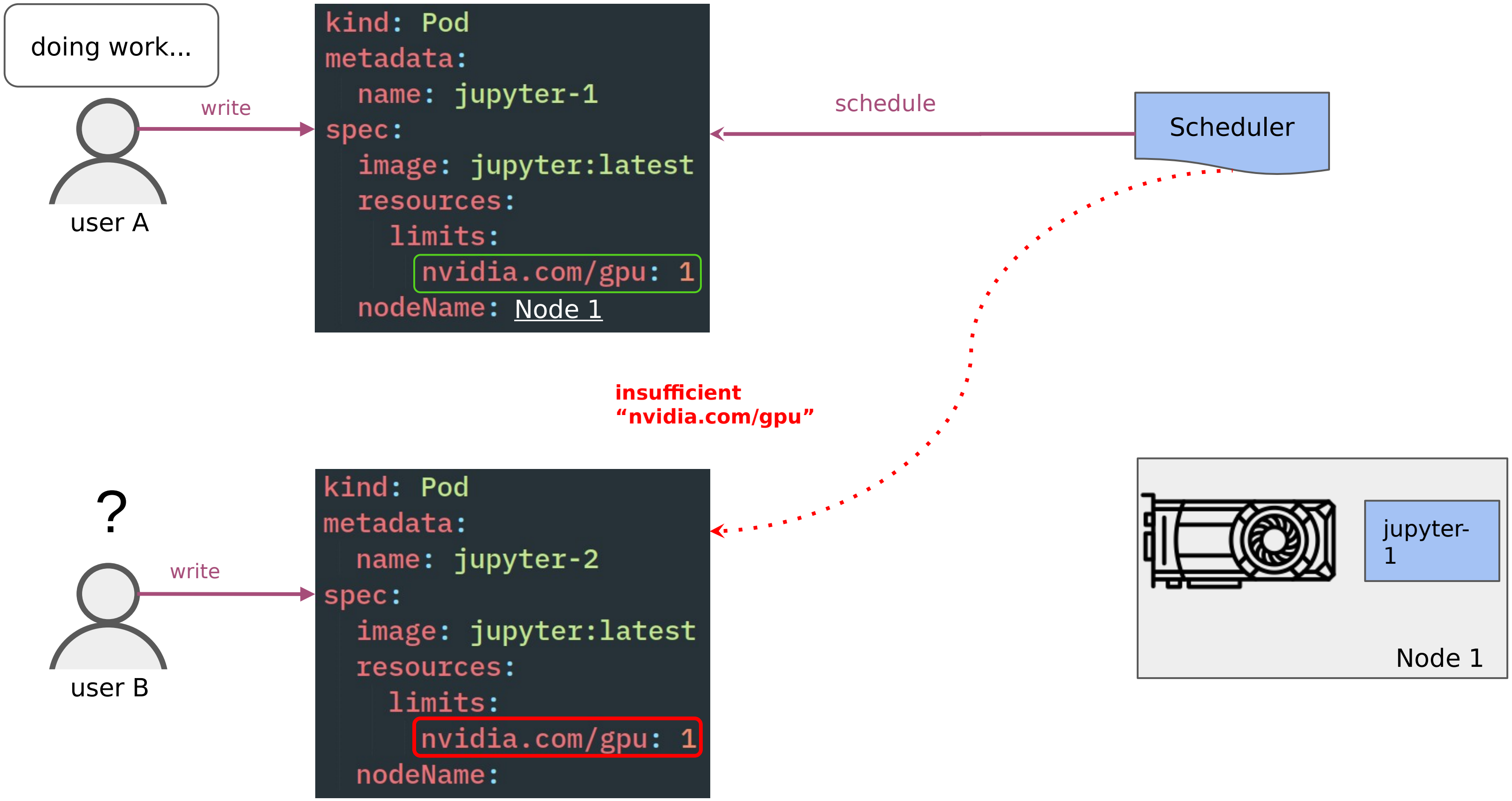
user B

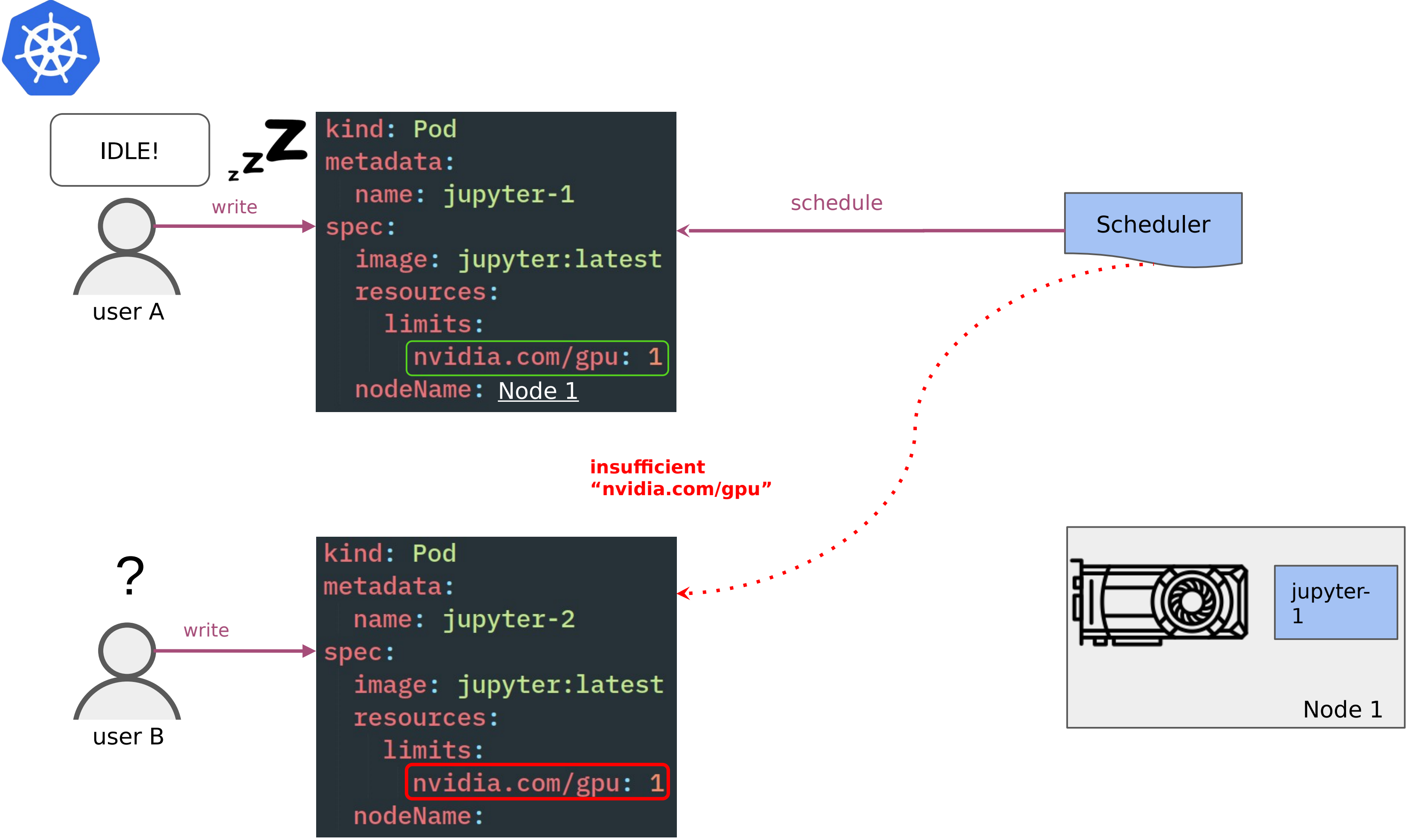
write

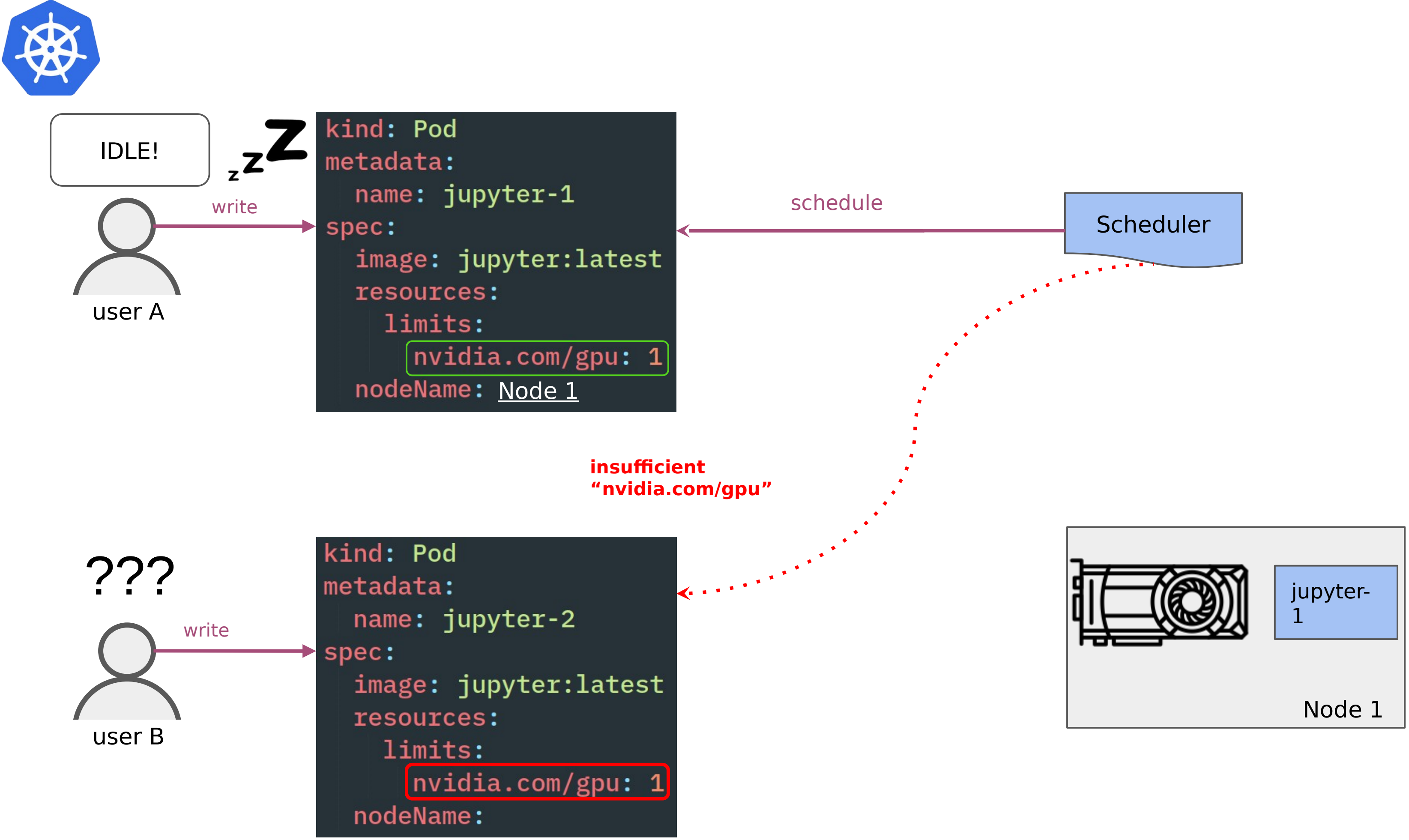
```yaml
kind: Pod
metadata:
  name: jupyter-2
spec:
  image: jupyter:latest
  resources:
    limits:
      nvidia.com/gpu: 1
  nodeName:
```



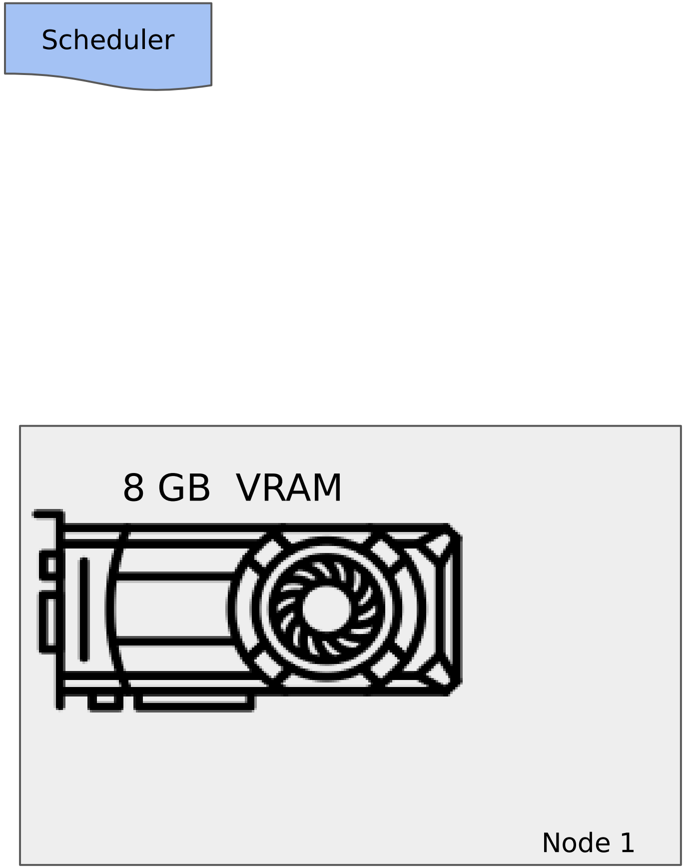

doing work...
write
user A
kind: Pod
metadata:
name: jupyter-1
spec:
image: jupyter:latest
resources:
limits:
nvidia.com/gpu: 1
nodeName: Node 1
schedule
Scheduler
insufficient
“nvidia.com/gpu”
write
user B
kind: Pod
metadata:
name: jupyter-2
spec:
image: jupyter:latest
resources:
limits:
nvidia.com/gpu: 1
nodeName:
jupyter-
1
Node 1


doing work...
write
user A
kind: Pod
metadata:
name: jupyter-1
spec:
image: jupyter:latest
resources:
limits:
nvidia.com/gpu: 1
nodeName: Node 1
schedule
Scheduler
insufficient
“nvidia.com/gpu”
?
write
user B
kind: Pod
metadata:
name: jupyter-2
spec:
image: jupyter:latest
resources:
limits:
nvidia.com/gpu: 1
nodeName:
jupyter-1
Node 1

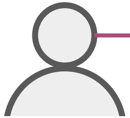

IDLE!
zzZ
write
user A
kind: Pod
metadata:
name: jupyter-1
spec:
image: jupyter:latest
resources:
limits:
nvidia.com/gpu: 1
nodeName: Node 1
schedule
Scheduler
insufficient
“nvidia.com/gpu”
?
write
user B
kind: Pod
metadata:
name: jupyter-2
spec:
image: jupyter:latest
resources:
limits:
nvidia.com/gpu: 1
nodeName:
jupyter-
1
Node 1


IDLE!
zzZ
write
user A
kind: Pod
metadata:
name: jupyter-1
spec:
image: jupyter:latest
resources:
limits:
nvidia.com/gpu: 1
nodeName: Node 1
schedule
Scheduler
insufficient
“nvidia.com/gpu”
???
write
user B
kind: Pod
metadata:
name: jupyter-2
spec:
image: jupyter:latest
resources:
limits:
nvidia.com/gpu: 1
nodeName:
jupyter-
1
Node 1

Existing Approaches (Aliyun, Kubeshare)



Existing Approaches (Aliyun, Kubeshare)



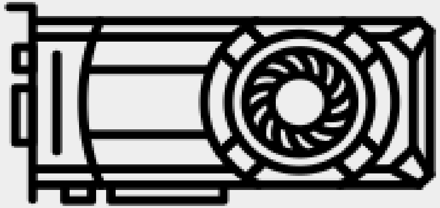
write

user
A

```yaml
kind: Pod
metadata:
  name: jupyter-1
spec:
  image: jupyter:latest
  resources:
    limits:
      aliyun.com/gpu-mem: 4
  nodeName:
```

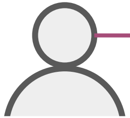
Scheduler

8 GB VRAM



Node 1

Existing Approaches (Aliyun, Kubeshare)



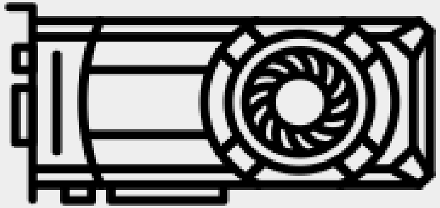
write

```yaml
kind: Pod
metadata:
  name: jupyter-1
spec:
  image: jupyter:latest
  resources:
    limits:
      aliyun.com/gpu-mem: 4
  nodeName:
```

schedule

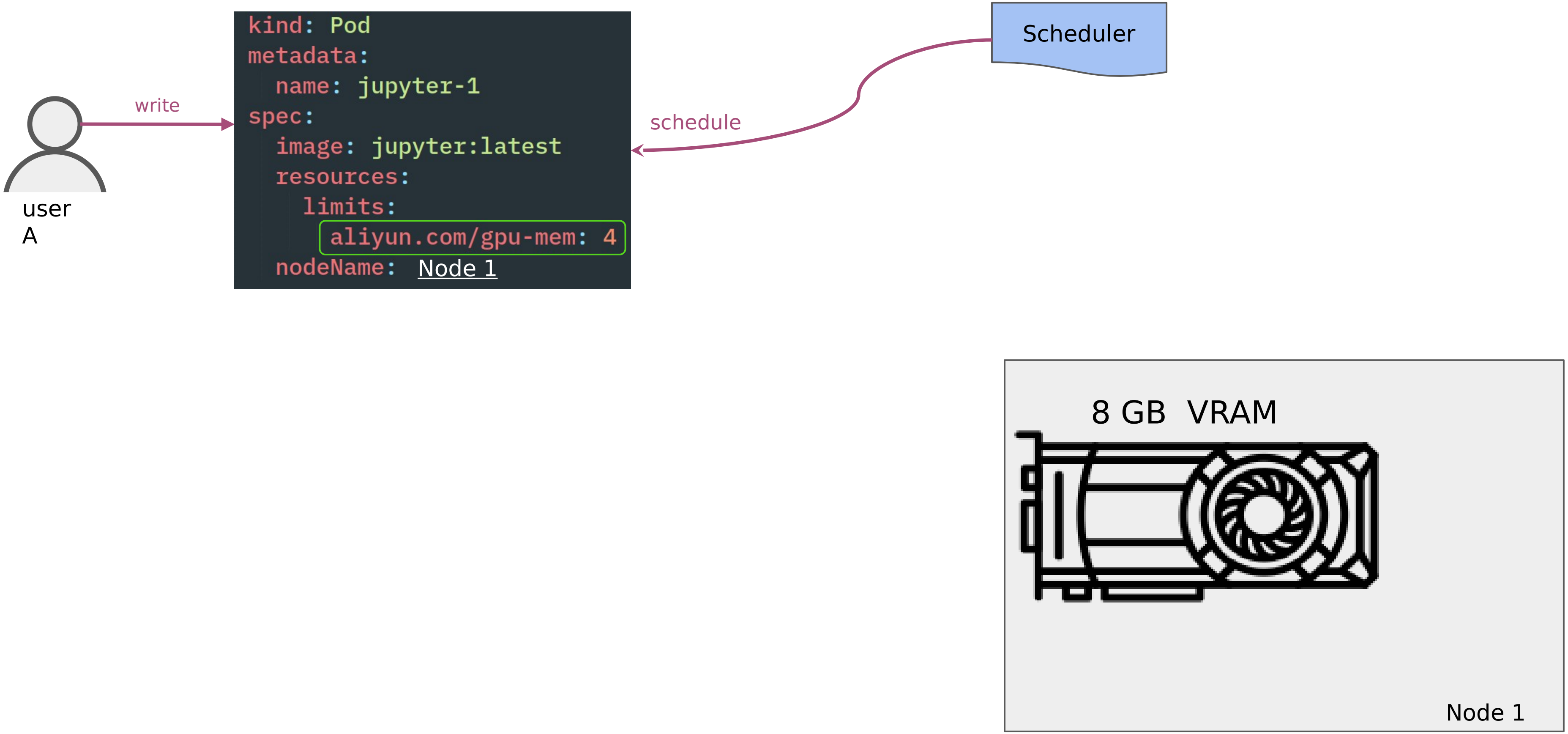
Scheduler

8 GB VRAM



Node 1

Existing Approaches (Aliyun, Kubeshare)



Existing Approaches (Aliyun, Kubeshare)

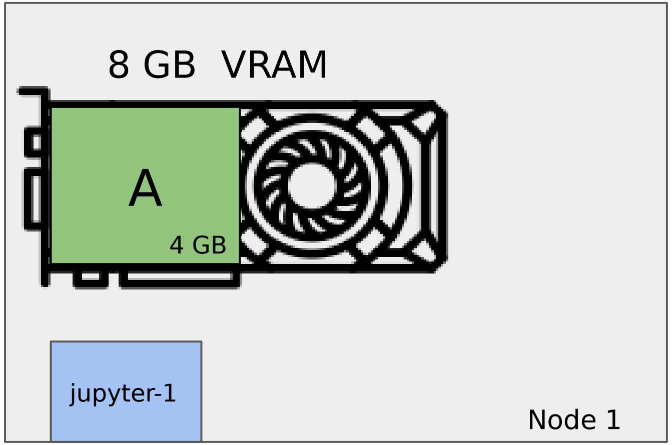
user A

write

```yaml
kind: Pod
metadata:
  name: jupyter-1
spec:
  image: jupyter:latest
  resources:
    limits:
      aliyun.com/gpu-mem: 4
  nodeName: Node 1
```

schedule

Scheduler



Existing Approaches (Aliyun, Kubeshare)

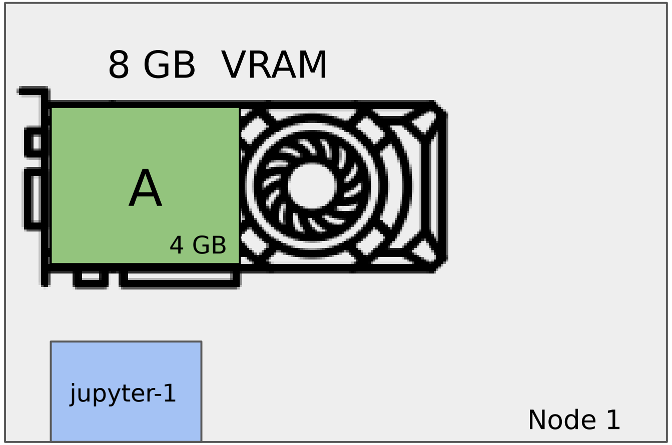
user A —write→

```yaml
kind: Pod
metadata:
  name: jupyter-1
spec:
  image: jupyter:latest
  resources:
    limits:
      aliyun.com/gpu-mem: 4
  nodeName: Node 1
```

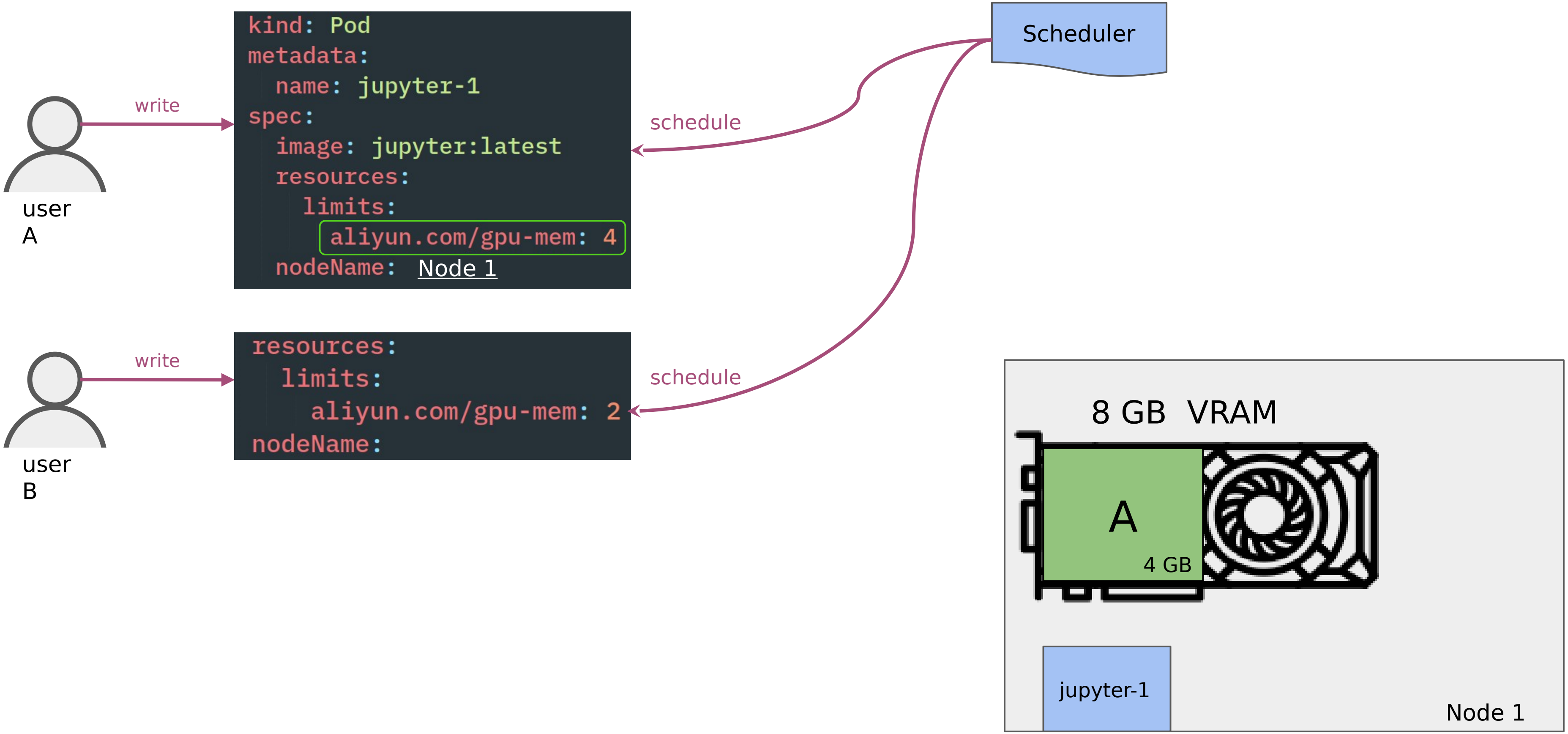
←schedule— Scheduler

user B —write→

```yaml
resources:
  limits:
    aliyun.com/gpu-mem: 2
nodeName:
```



Existing Approaches (Aliyun, Kubeshare)



Existing Approaches (Aliyun, Kubeshare)

Scheduler

user A — write

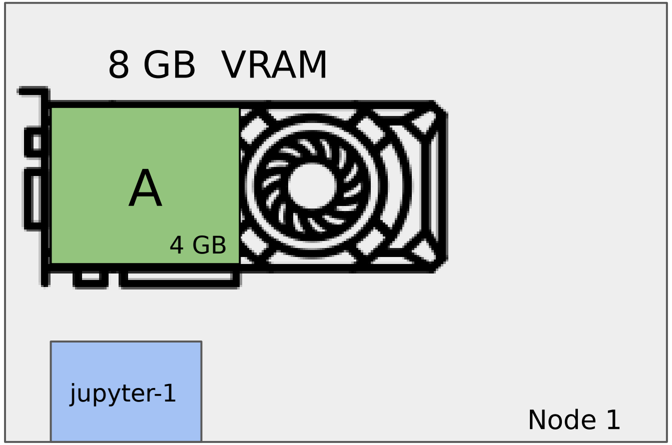
```yaml
kind: Pod
metadata:
  name: jupyter-1
spec:
  image: jupyter:latest
  resources:
    limits:
      aliyun.com/gpu-mem: 4
  nodeName: Node 1
```

schedule

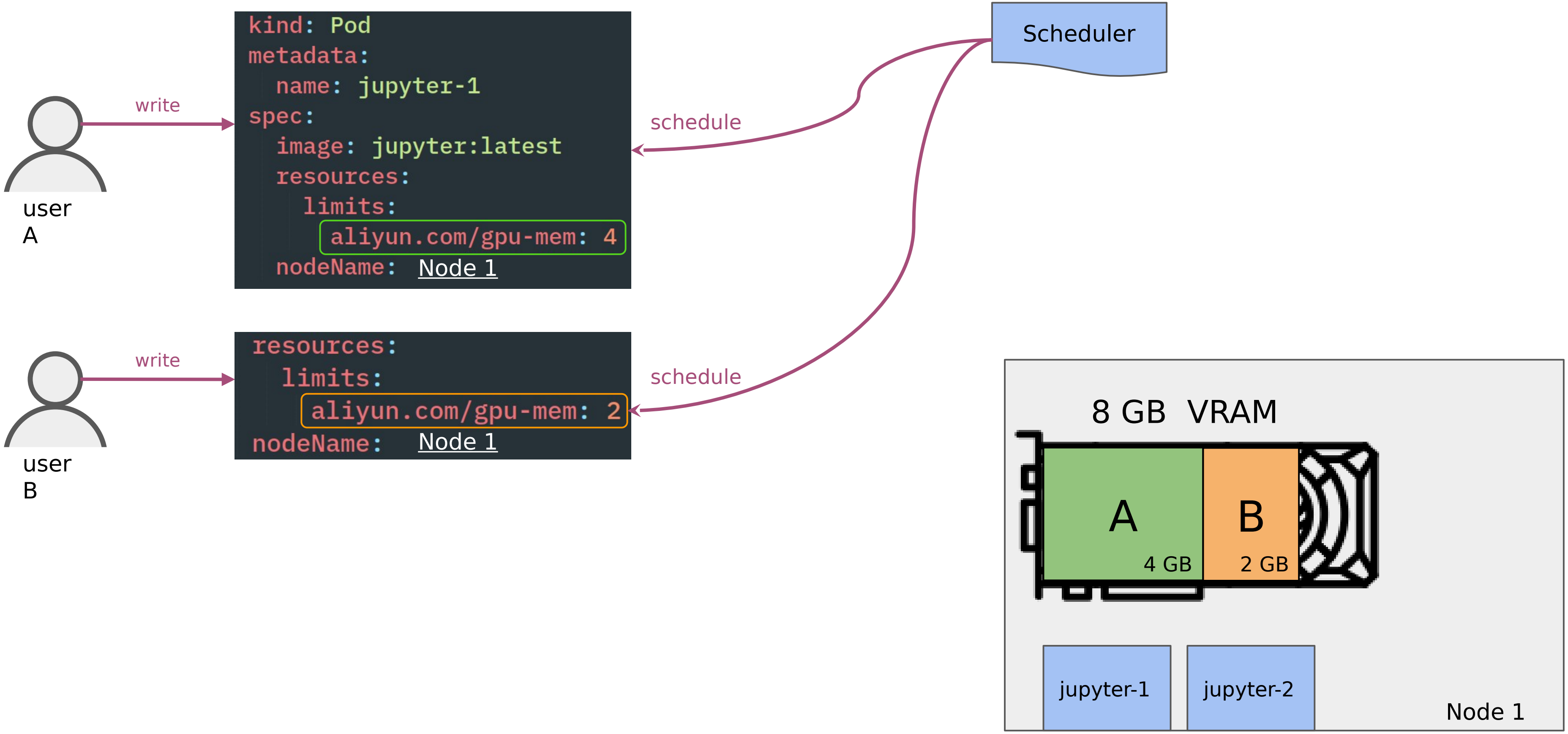
user B — write

```yaml
resources:
  limits:
    aliyun.com/gpu-mem: 2
nodeName: Node 1
```

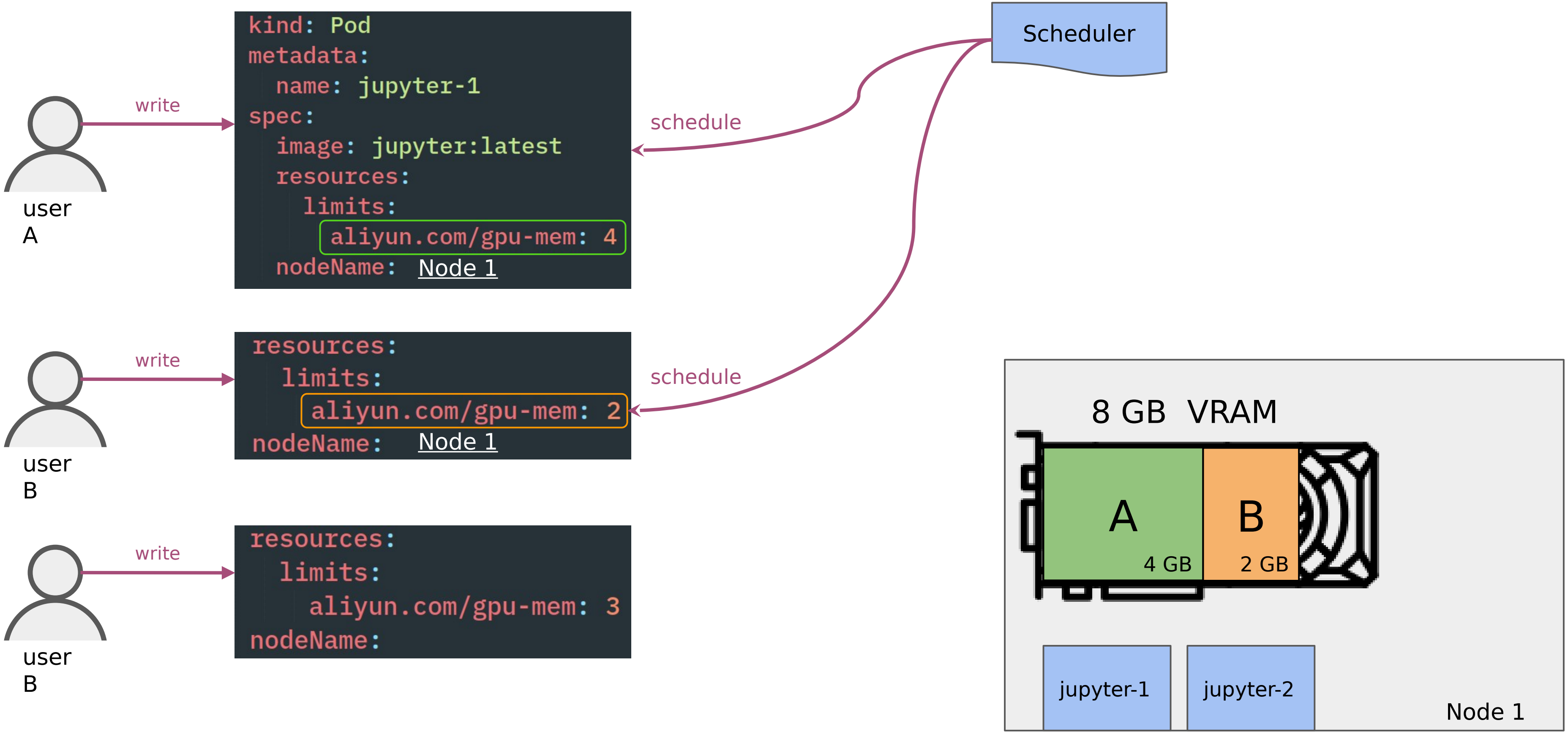
schedule



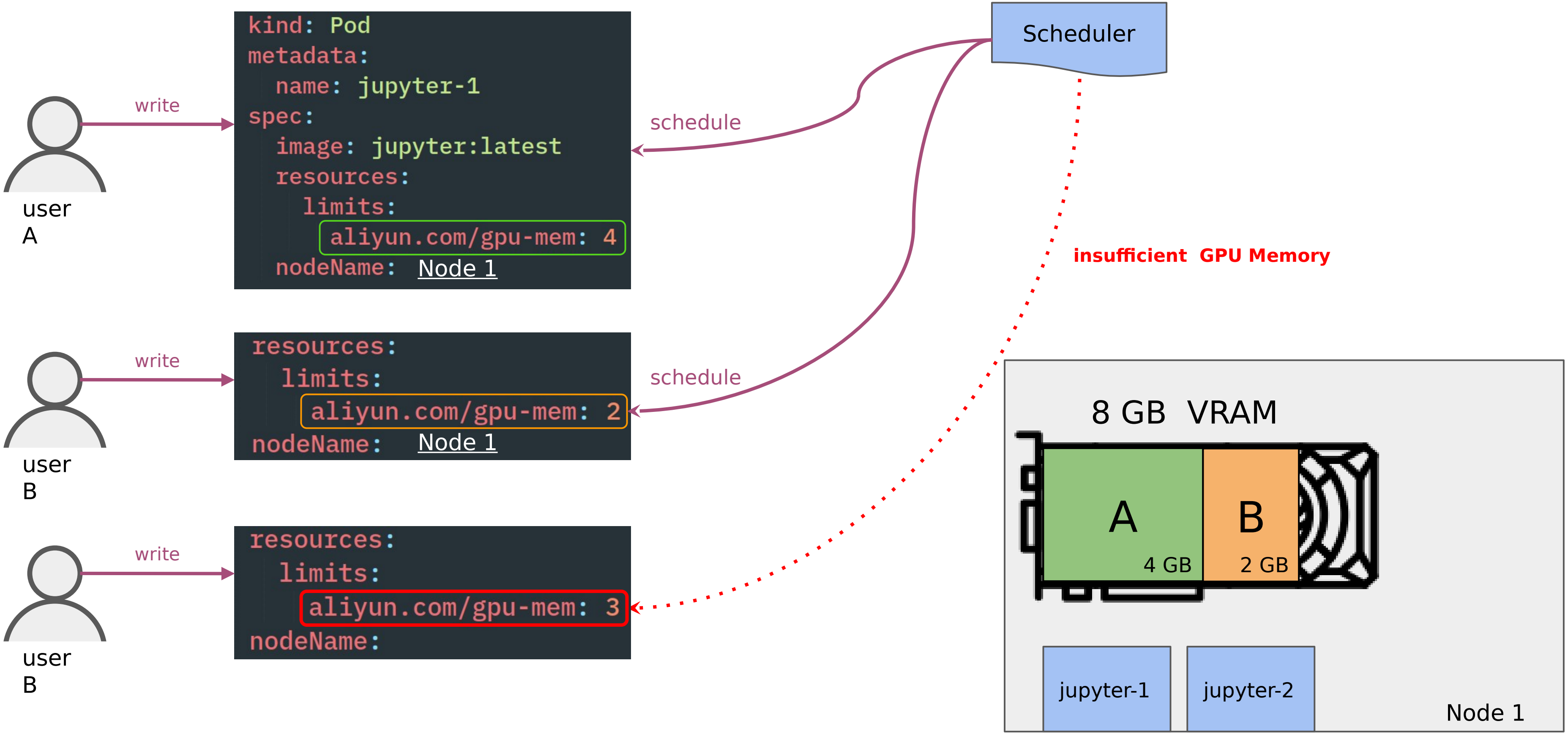
Existing Approaches (Aliyun, Kubeshare)



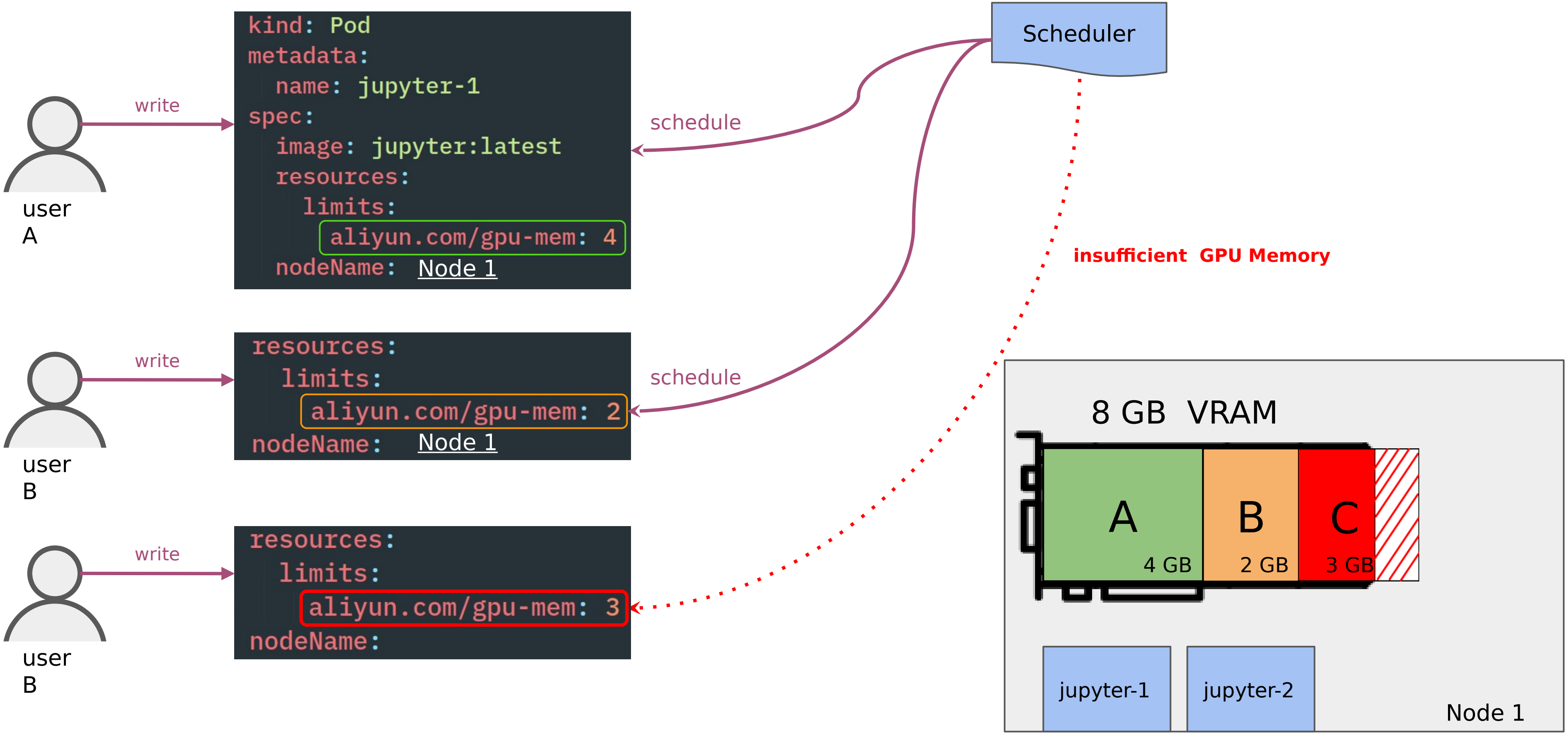
Existing Approaches (Aliyun, Kubeshare)



Existing Approaches (Aliyun, Kubeshare)

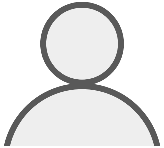


Existing Approaches (Aliyun, Kubeshare)

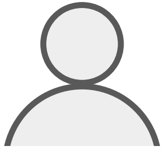


Key Idea

Key Idea

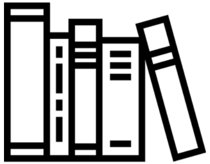


Key Idea

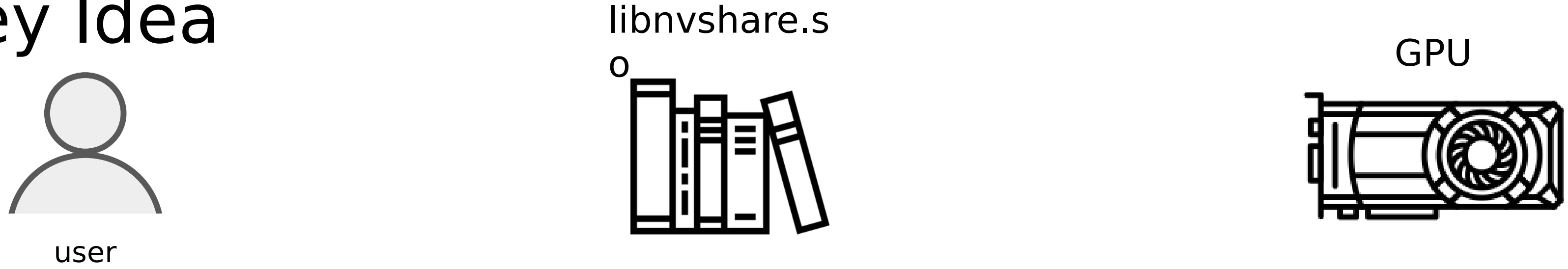


user

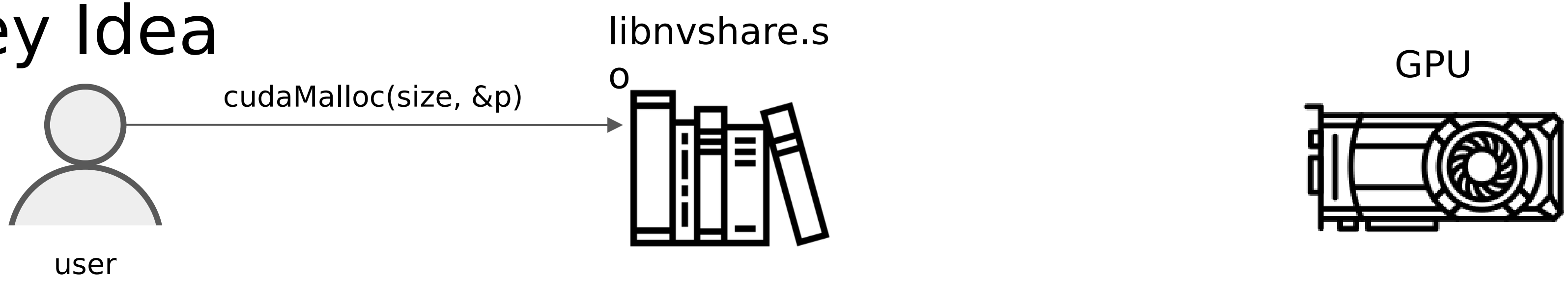
libnvshare.s
o



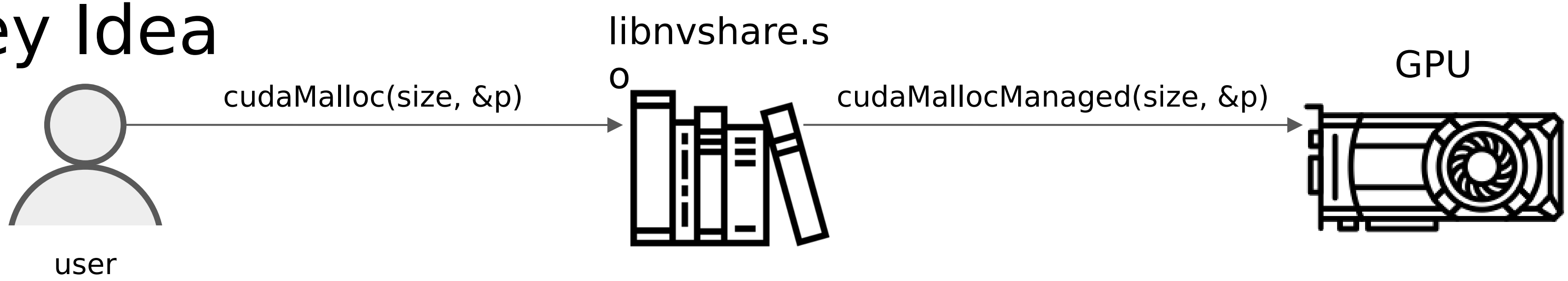
Key Idea



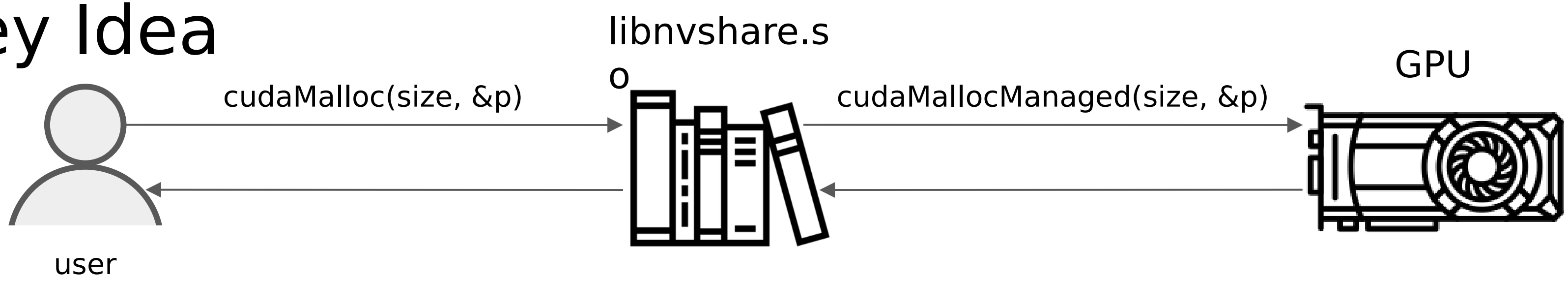
Key Idea



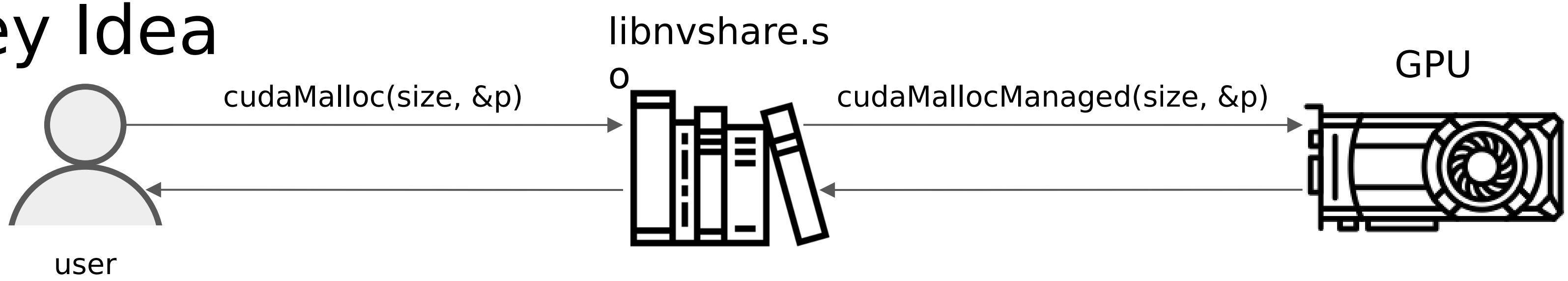
Key Idea


libnvshare.so
GPU
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
user

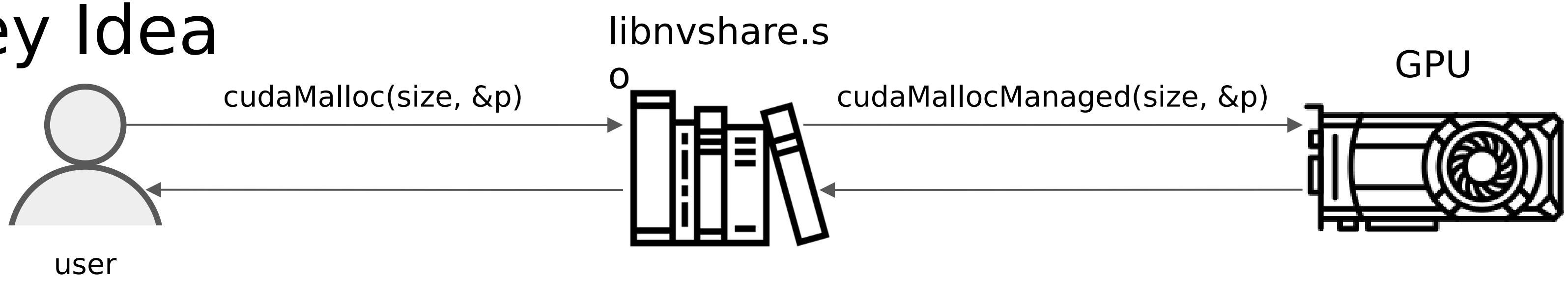
Key Idea


libnvshare.s
o
GPU
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
user

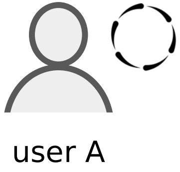
Key Idea


libnvshare.so
GPU
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
user

Key Idea

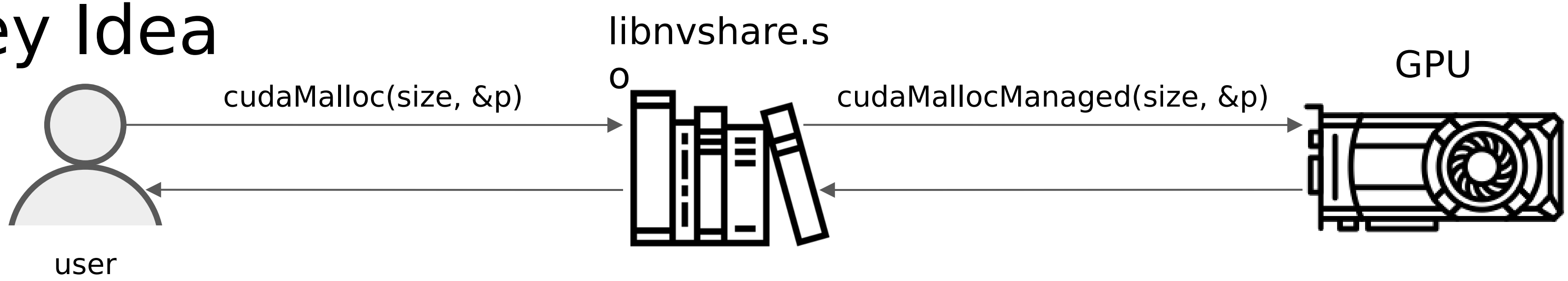


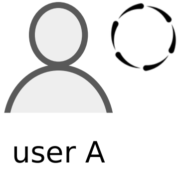
libnvshare.s
o
GPU
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
user

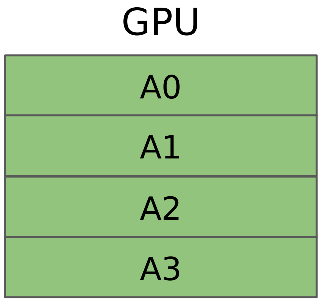


user A

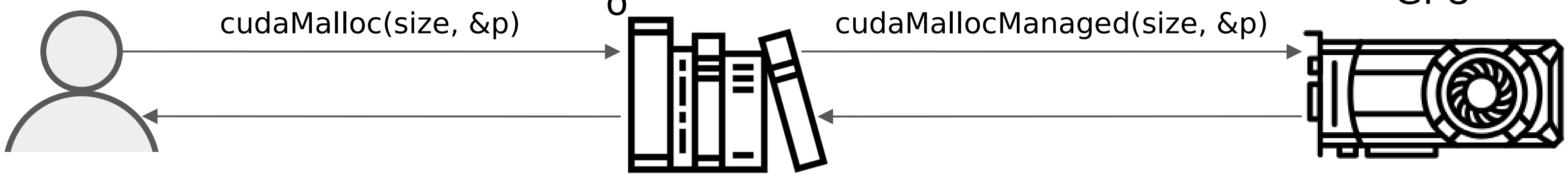
Key Idea

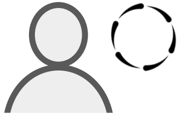

libnvshare.so
GPU
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
user


user A


GPU
A0
A1
A2
A3

Key Idea

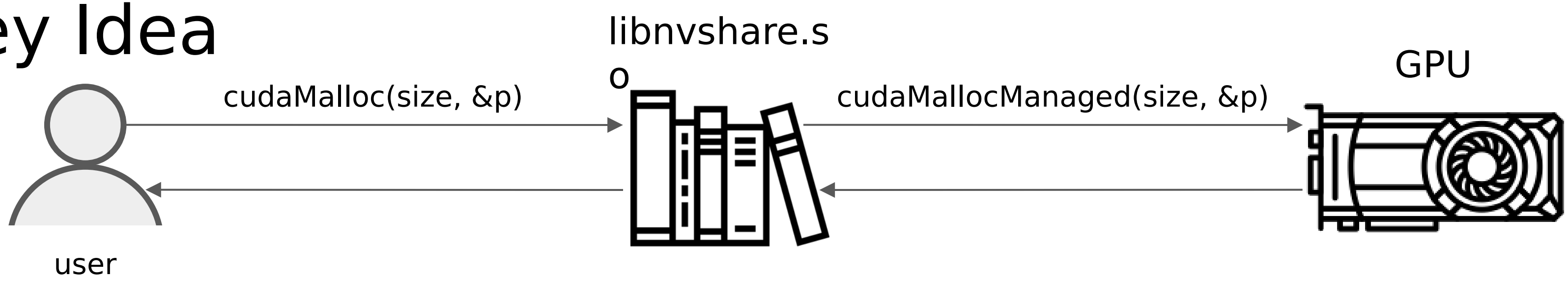


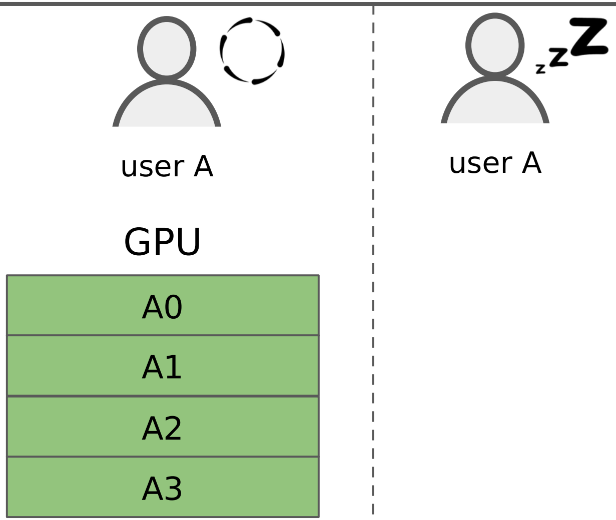


GPU

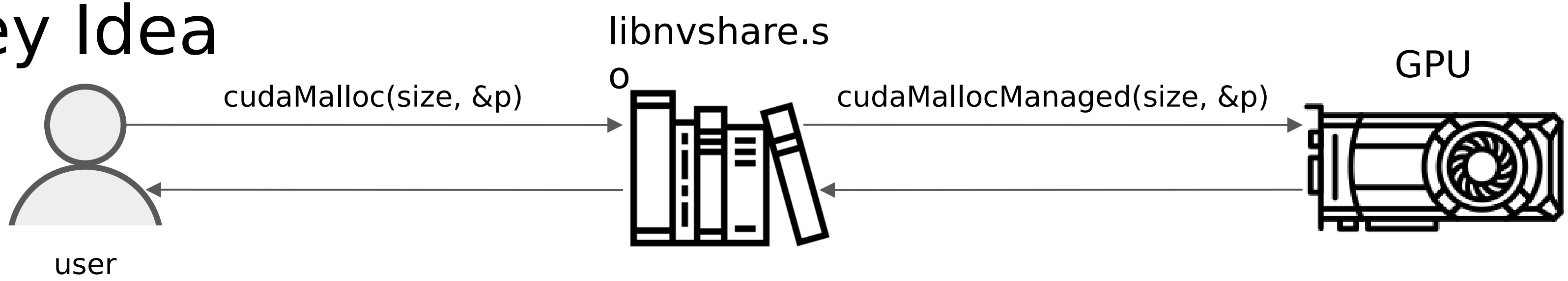
A0
A1
A2
A3

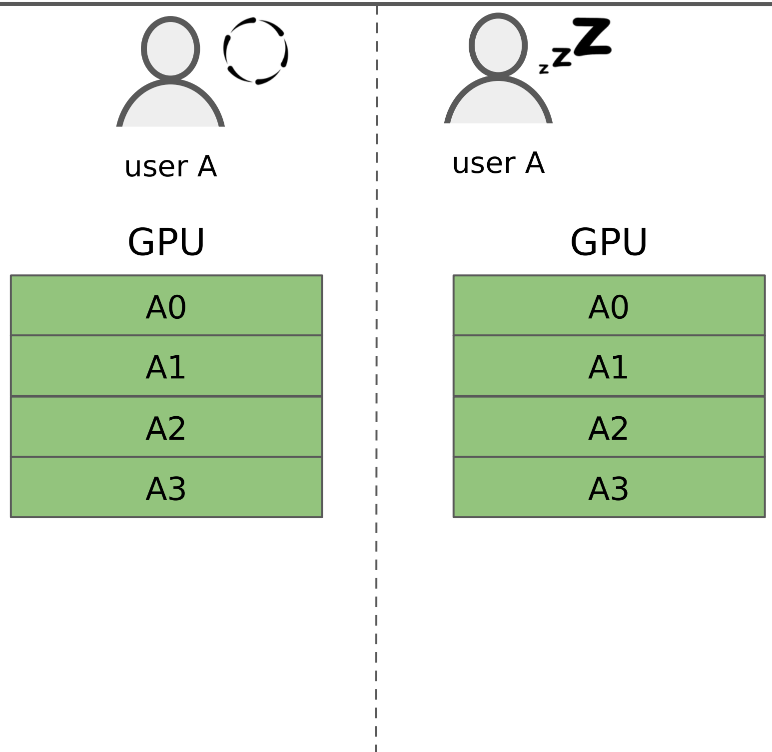
Key Idea


libnvshare.so
GPU
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
user

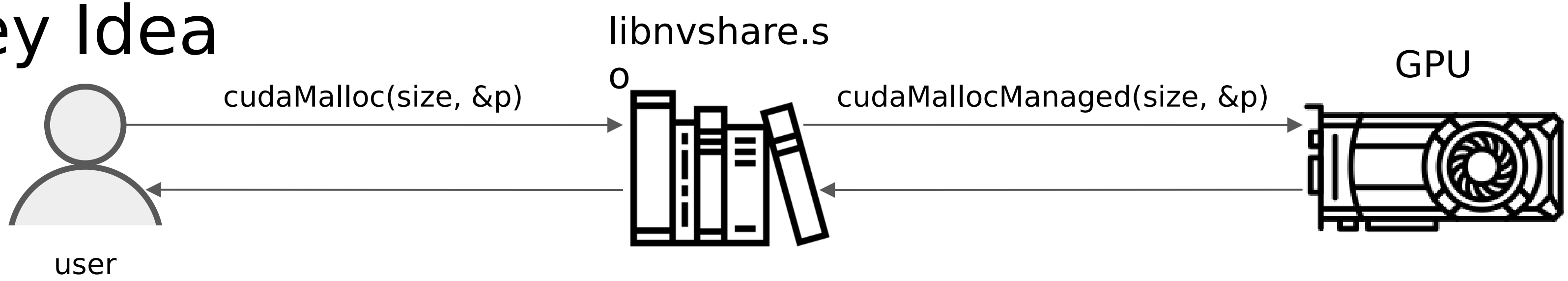

user A
user A
GPU
A0
A1
A2
A3

Key Idea

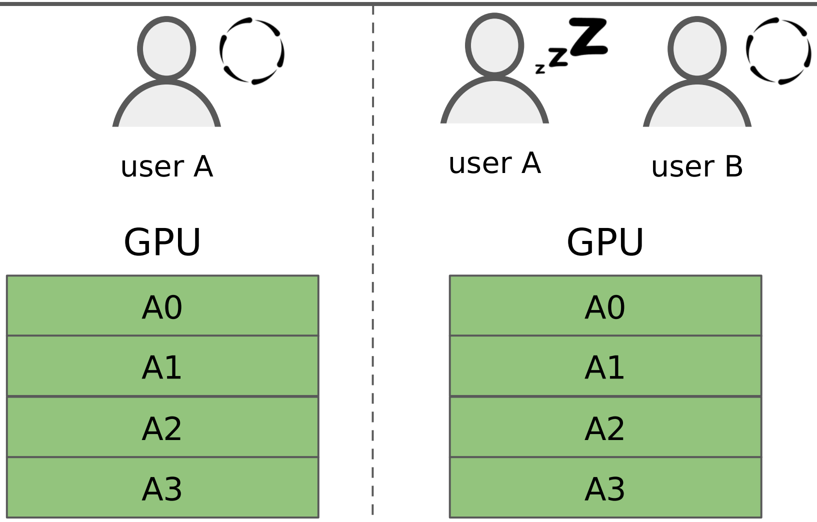

libnvshare.so
GPU
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
user


user A
GPU
A0
A1
A2
A3
user A
GPU
A0
A1
A2
A3

Key Idea

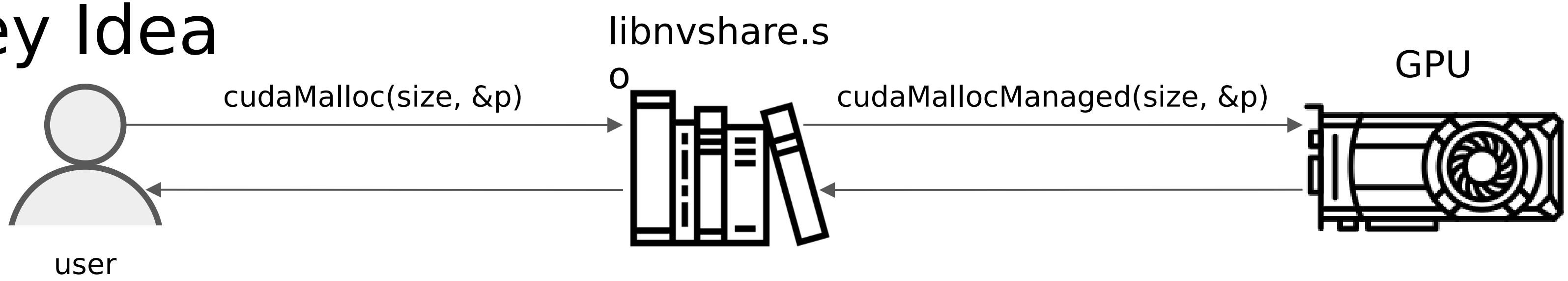


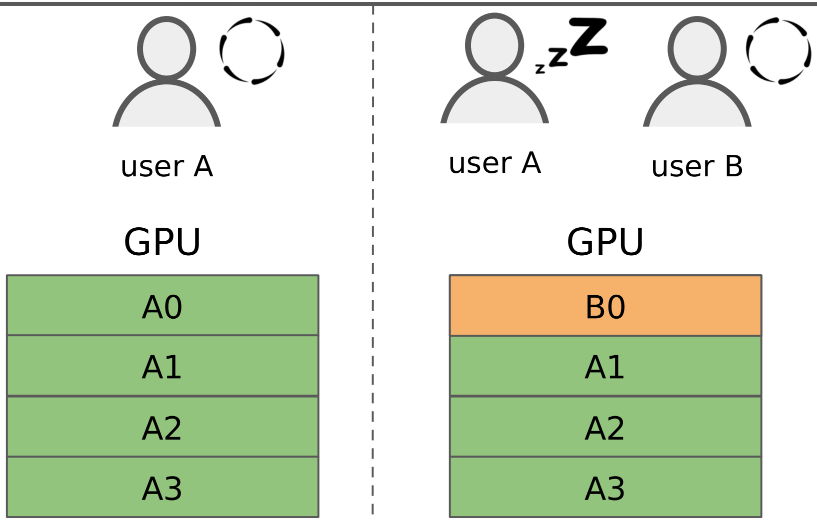
libnvshare.so
GPU
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
user



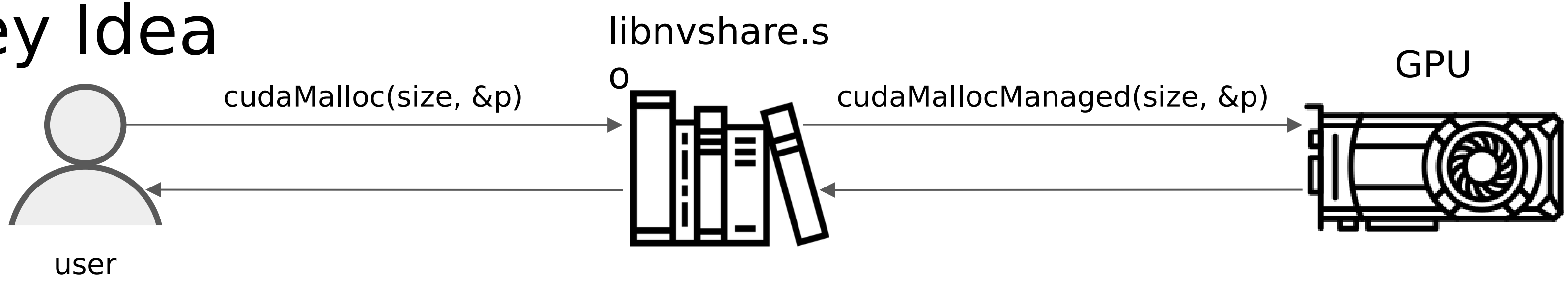
user A
GPU
A0
A1
A2
A3
user A
user B
GPU
A0
A1
A2
A3

Key Idea

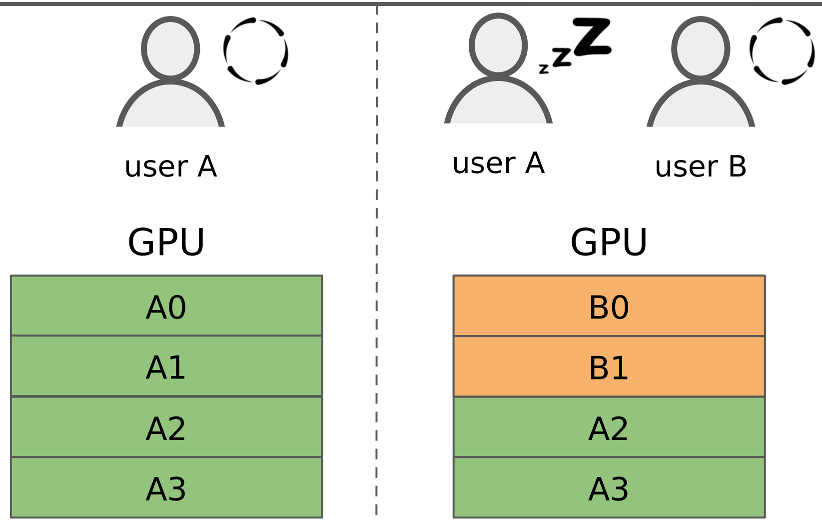

libnvshare.so
GPU
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
user


user A
user A
user B
GPU
A0
A1
A2
A3
GPU
B0
A1
A2
A3

Key Idea

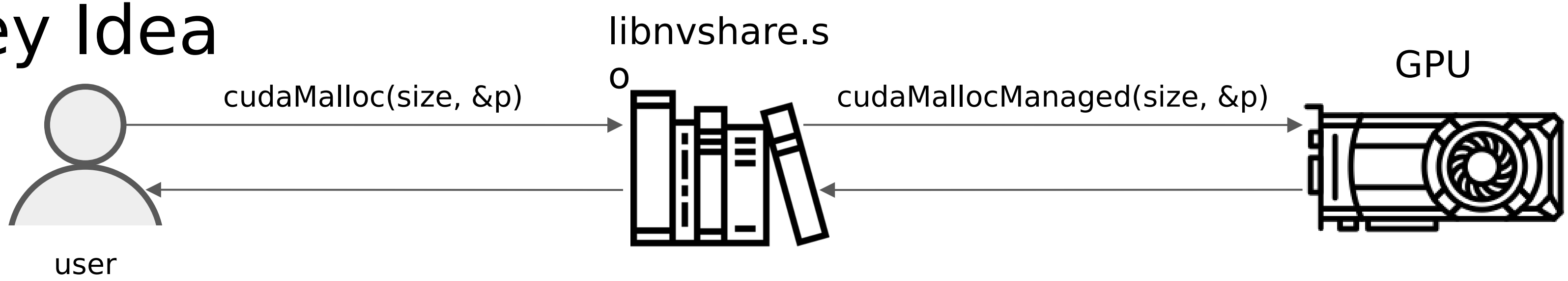


libnvshare.s
o
GPU
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
user

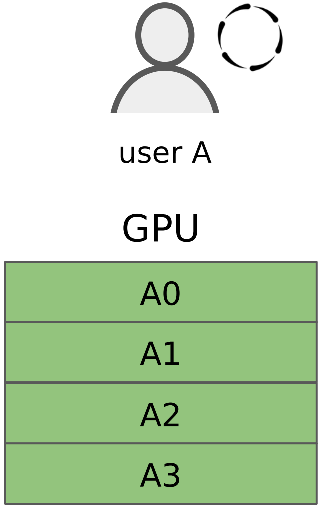


user A
GPU
A0
A1
A2
A3
user A
user B
GPU
B0
B1
A2
A3

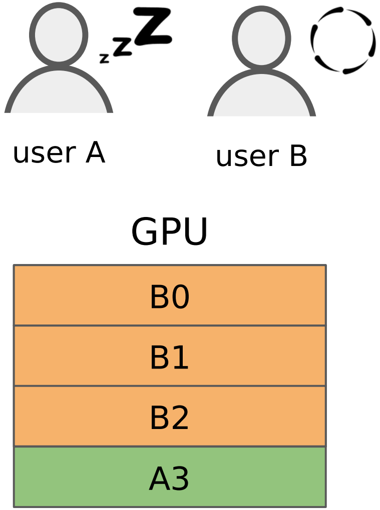
Key Idea



libnvshare.so
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
GPU
user

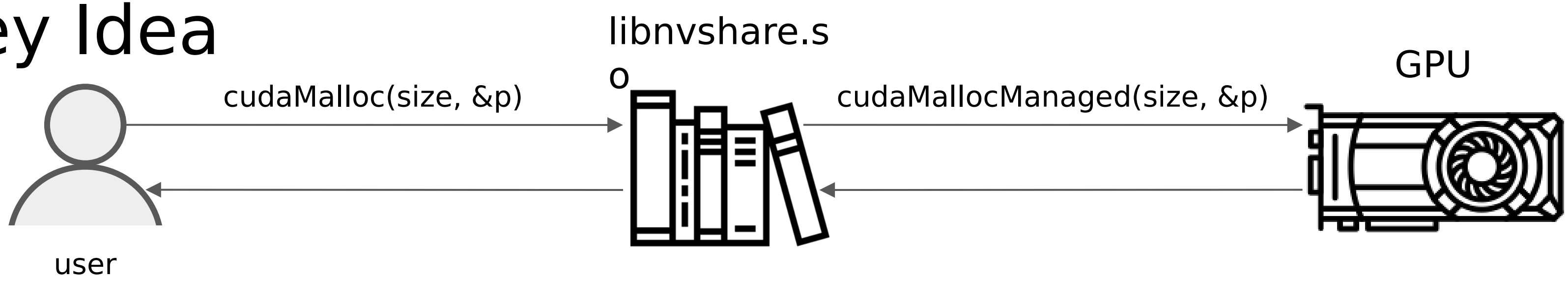


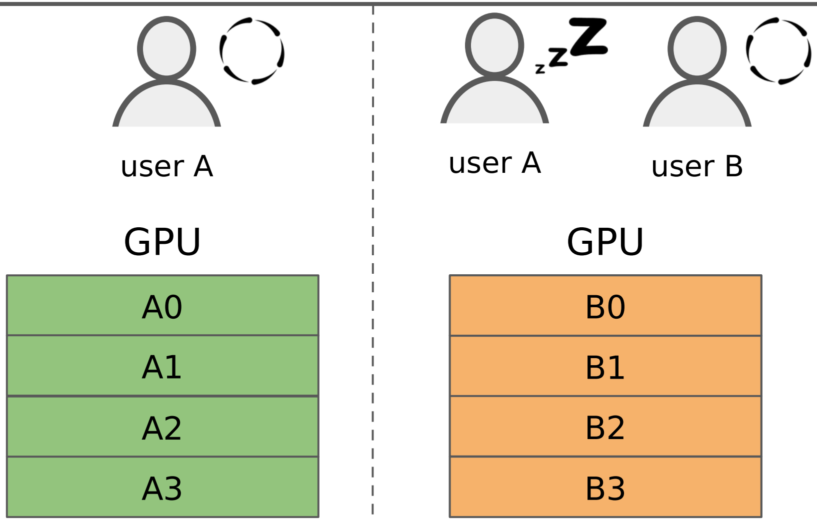
user A
GPU
A0
A1
A2
A3



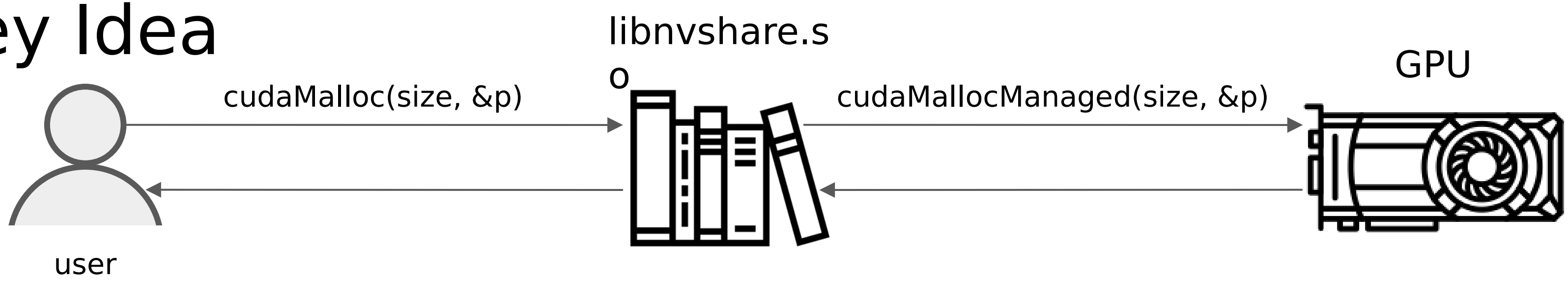
user A
user B
GPU
B0
B1
B2
A3

Key Idea

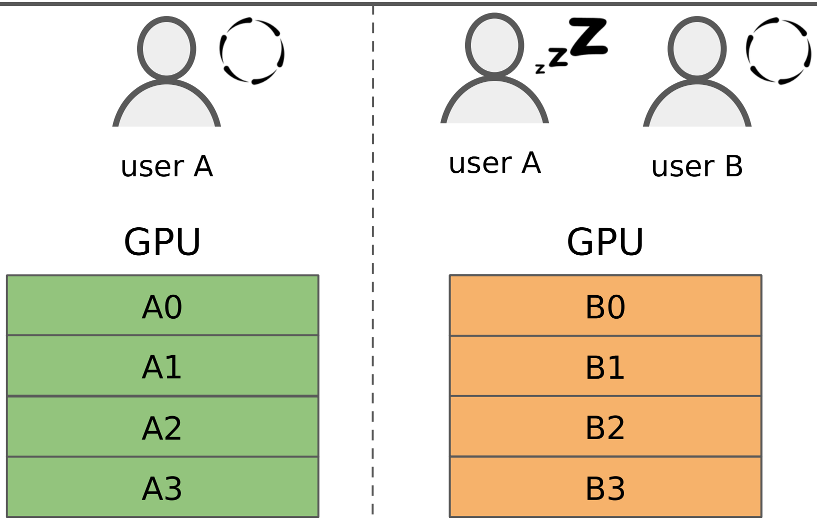

libnvshare.s
o
GPU
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
user


user A
user A
user B
GPU
A0
A1
A2
A3
GPU
B0
B1
B2
B3

Key Idea

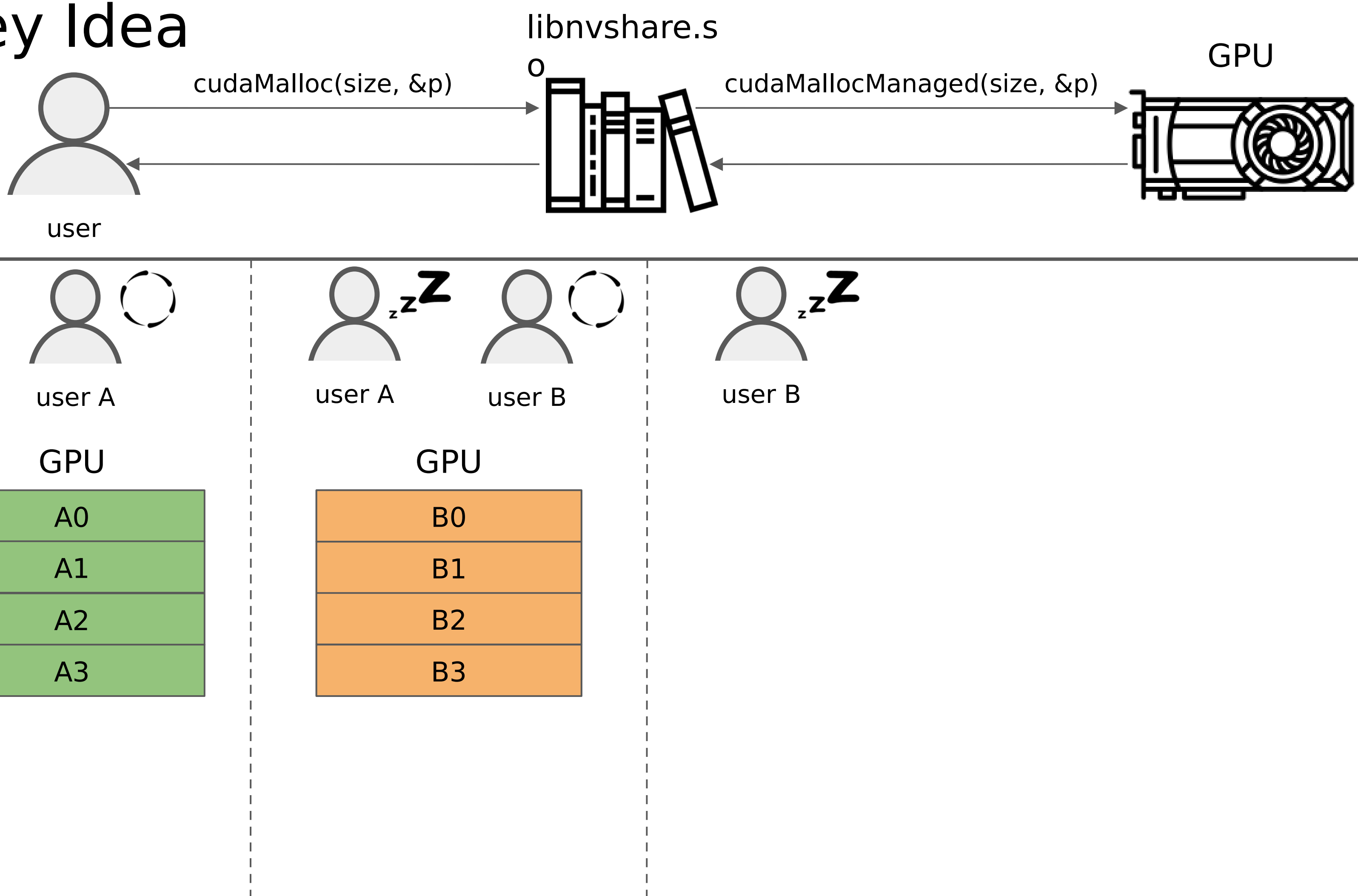


libnvshare.so
GPU
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
user



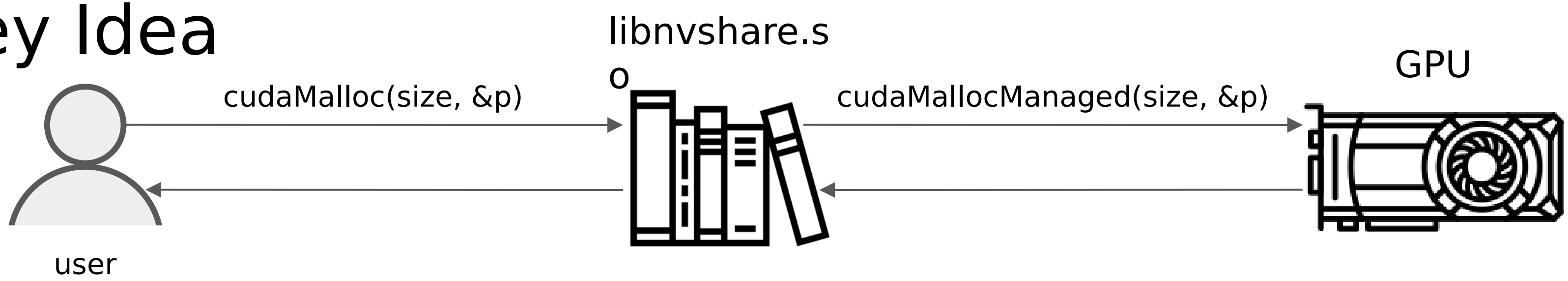
user A
user A
user B
GPU
A0
A1
A2
A3
GPU
B0
B1
B2
B3

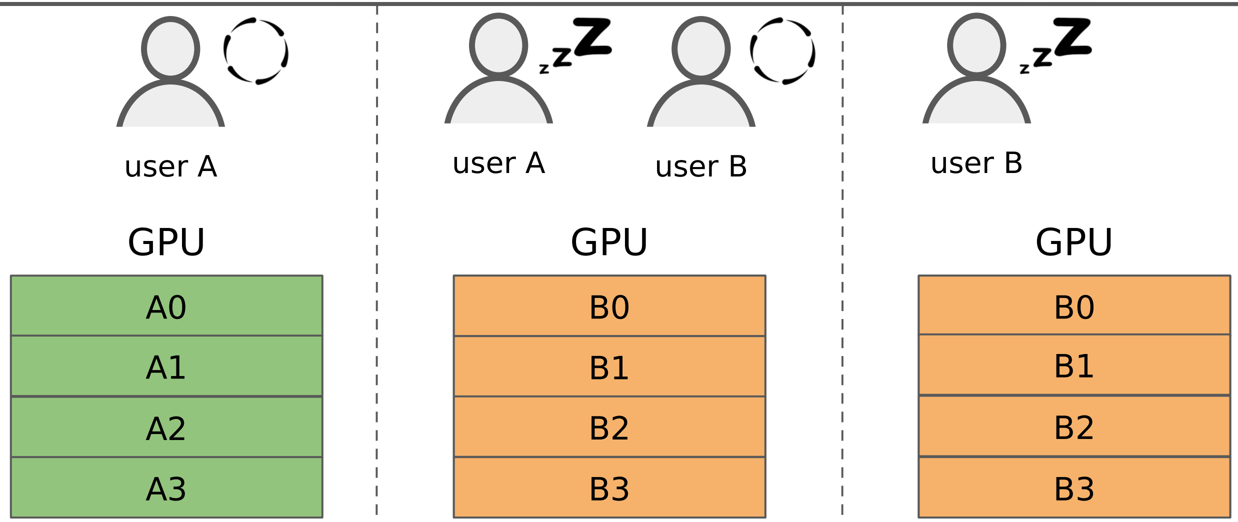
Key Idea



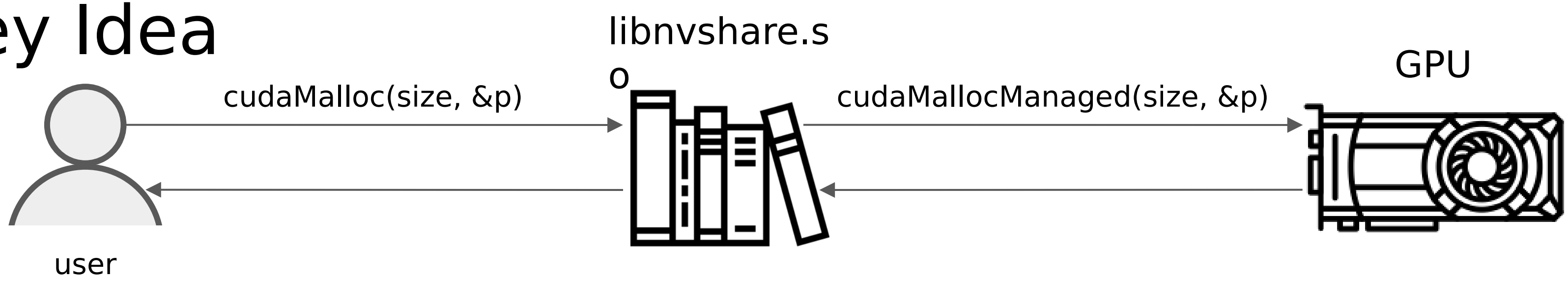
libnvshare.so
GPU
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
user
user A
user A
user B
user B
GPU
A0
A1
A2
A3
GPU
B0
B1
B2
B3

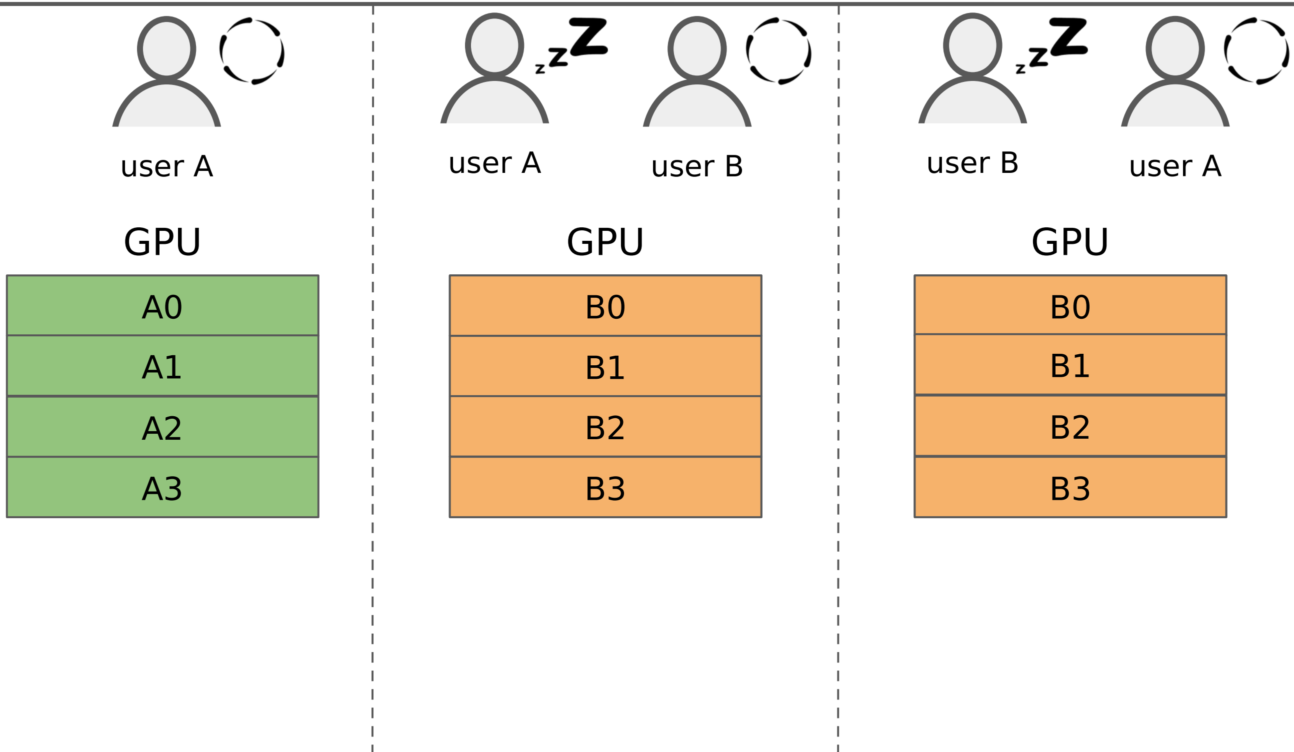
Key Idea


libnvshare.s
o
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
GPU
user

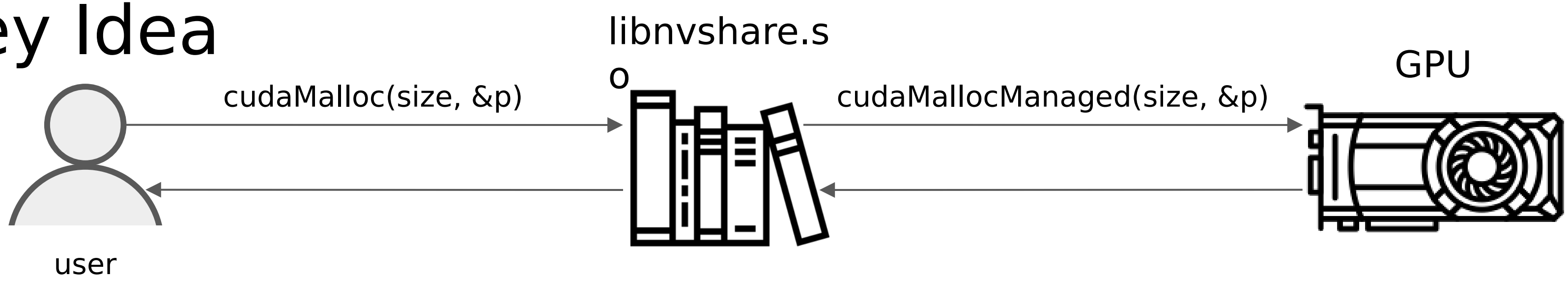

user A
GPU
A0
A1
A2
A3
user A
user B
GPU
B0
B1
B2
B3
user B
GPU
B0
B1
B2
B3

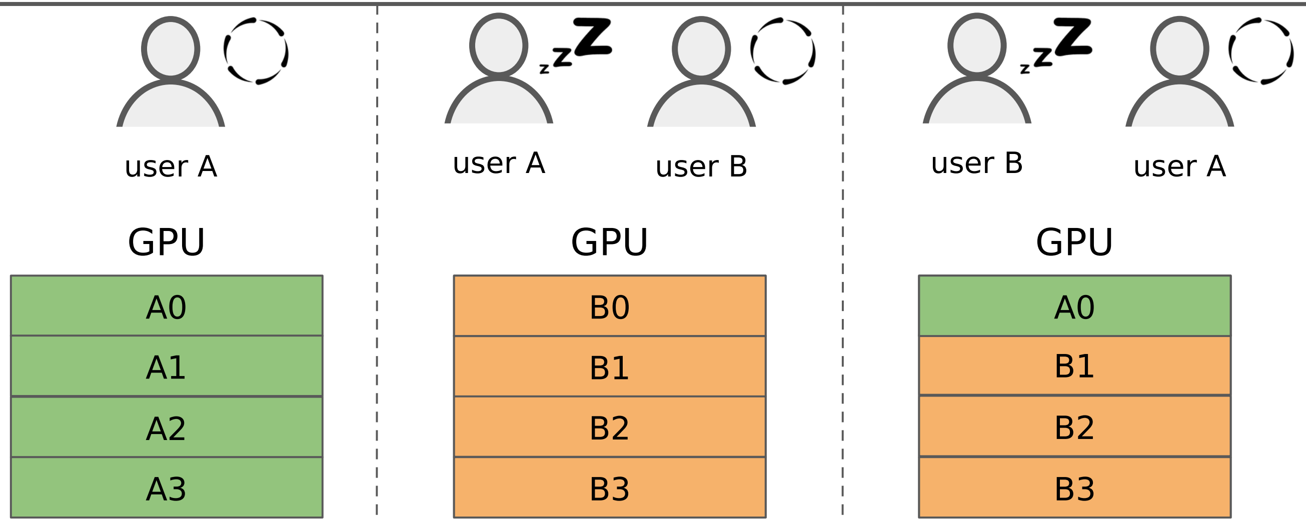
Key Idea


libnvshare.so
GPU
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
user

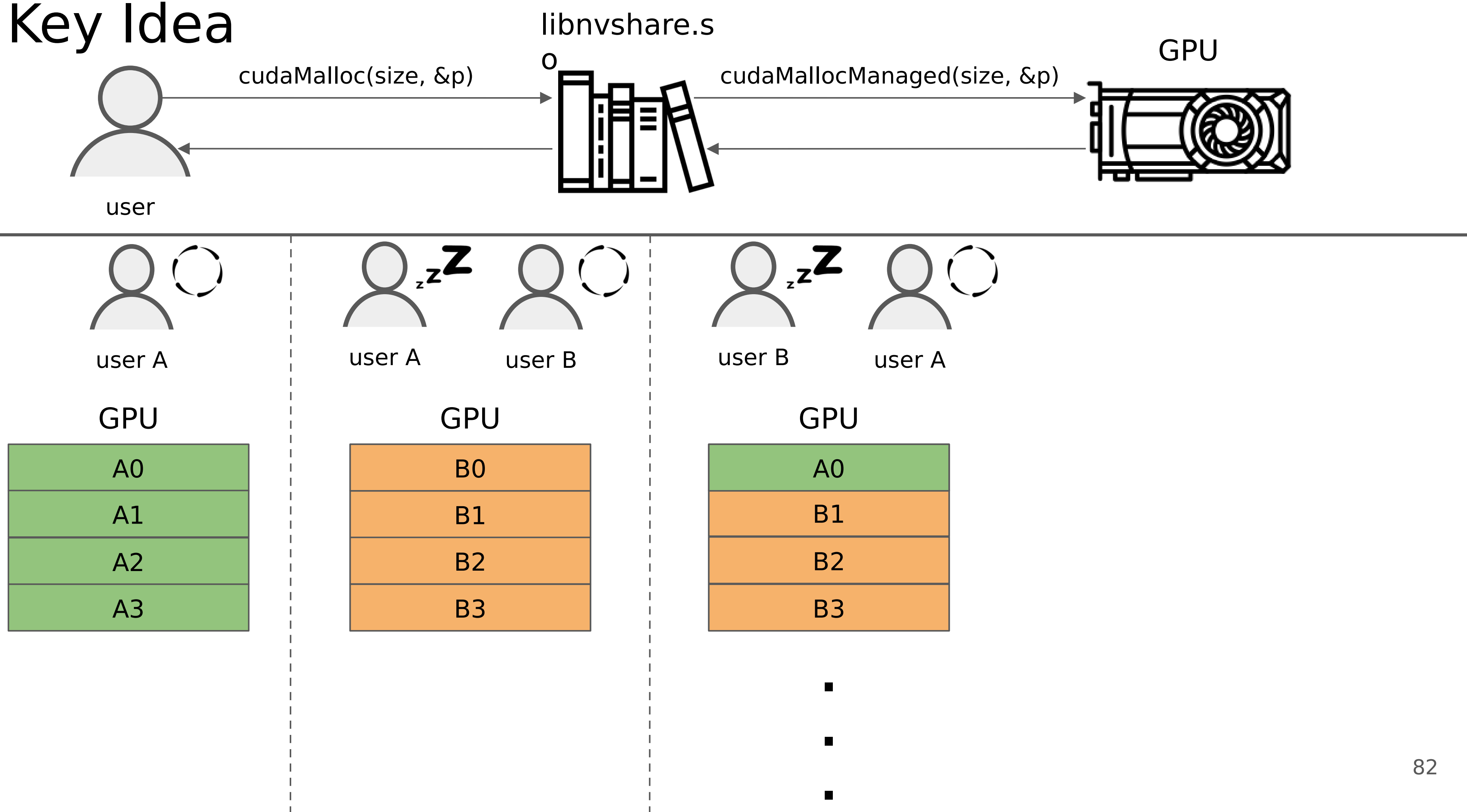

user A
GPU
A0
A1
A2
A3
user A
user B
GPU
B0
B1
B2
B3
user B
user A
GPU
B0
B1
B2
B3

Key Idea


libnvshare.so
GPU
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
user

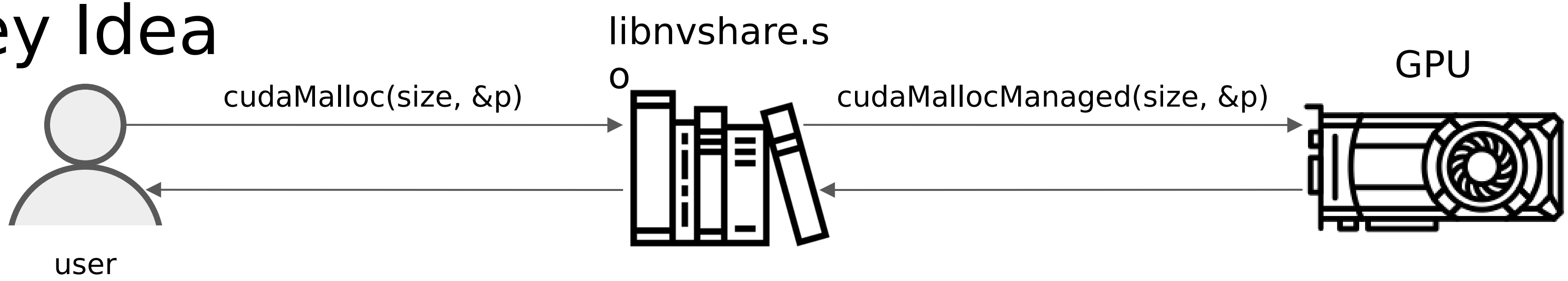

user A
GPU
A0
A1
A2
A3
user A
user B
GPU
B0
B1
B2
B3
user B
user A
GPU
A0
B1
B2
B3

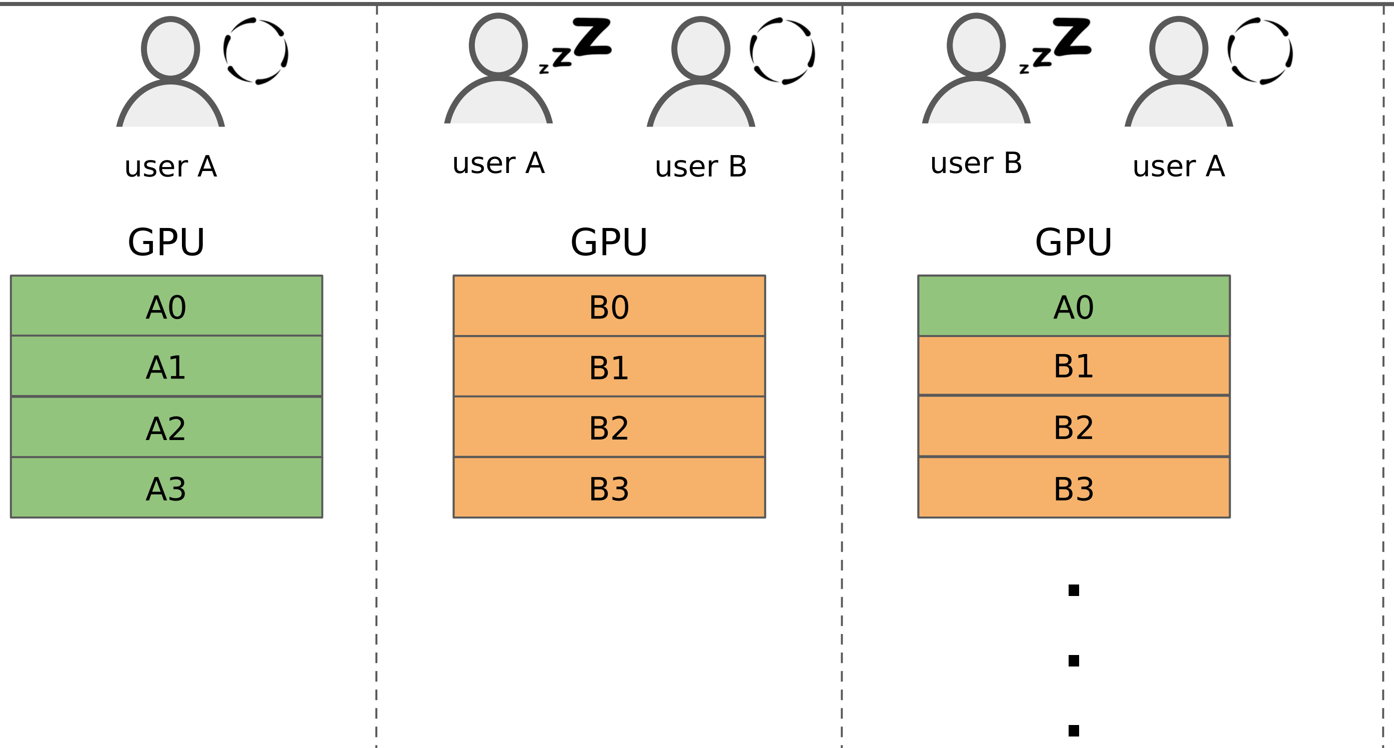
Key Idea



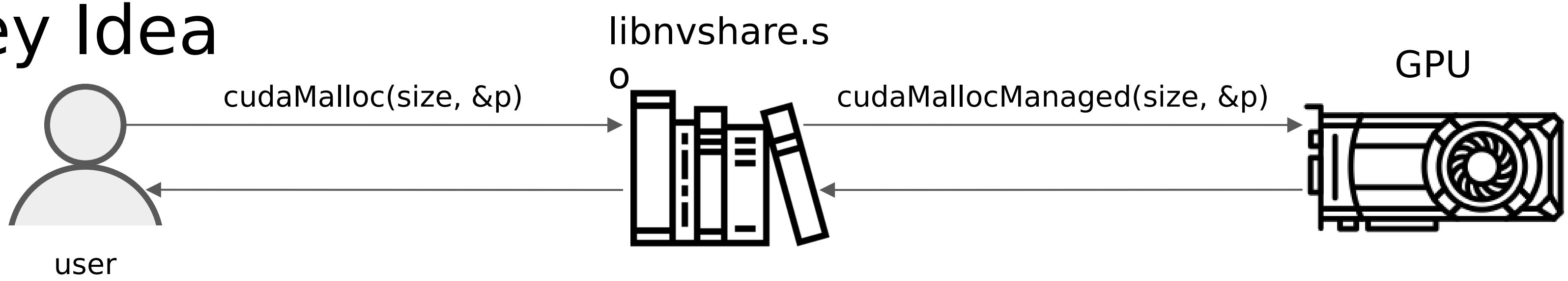
libnvshare.so
GPU
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
user
user A
user A
user B
user B
user A
GPU
A0
A1
A2
A3
GPU
B0
B1
B2
B3
GPU
A0
B1
B2
B3

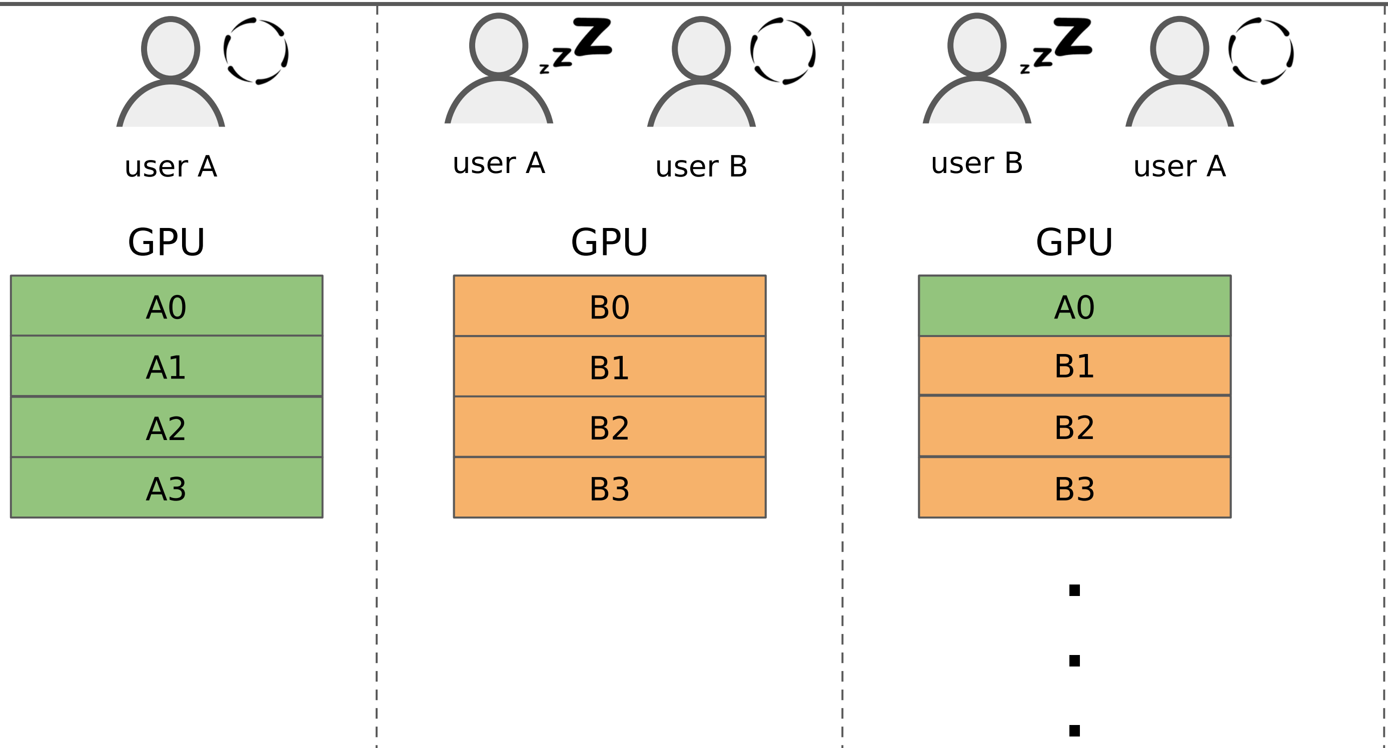
Key Idea


libnvshare.so
GPU
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
user


user A
GPU
A0
A1
A2
A3
user A
user B
GPU
B0
B1
B2
B3
user B
user A
GPU
A0
B1
B2
B3

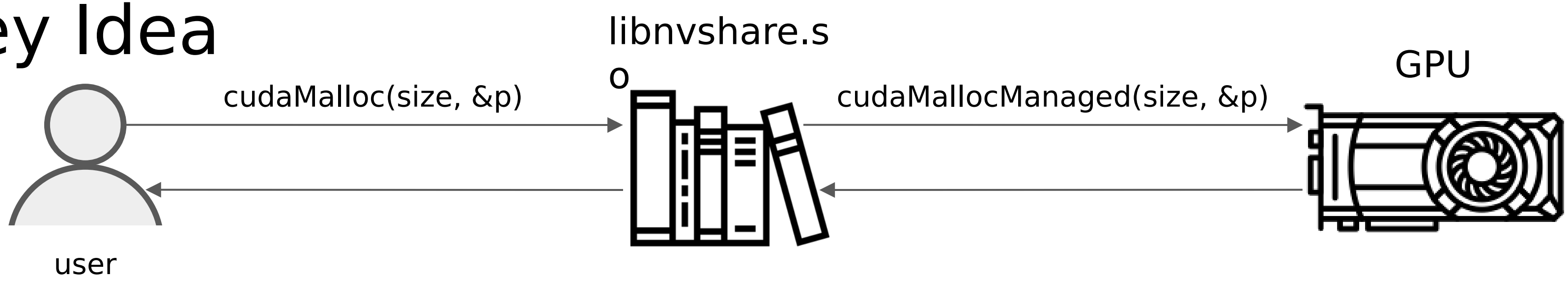
Key Idea

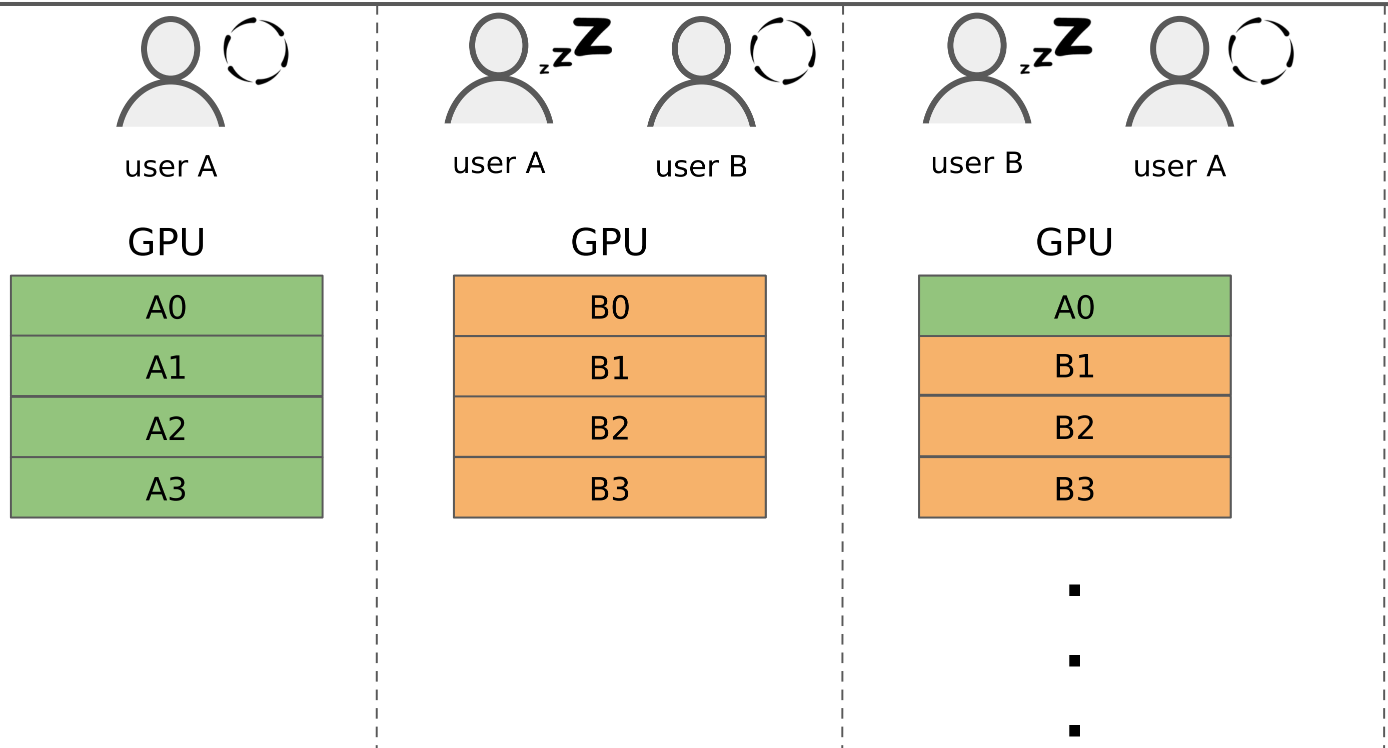




Result

Key Idea

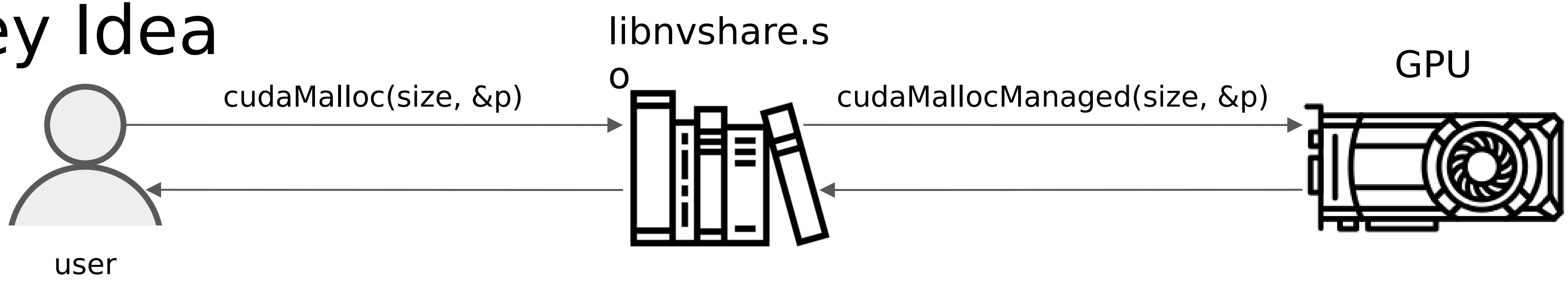


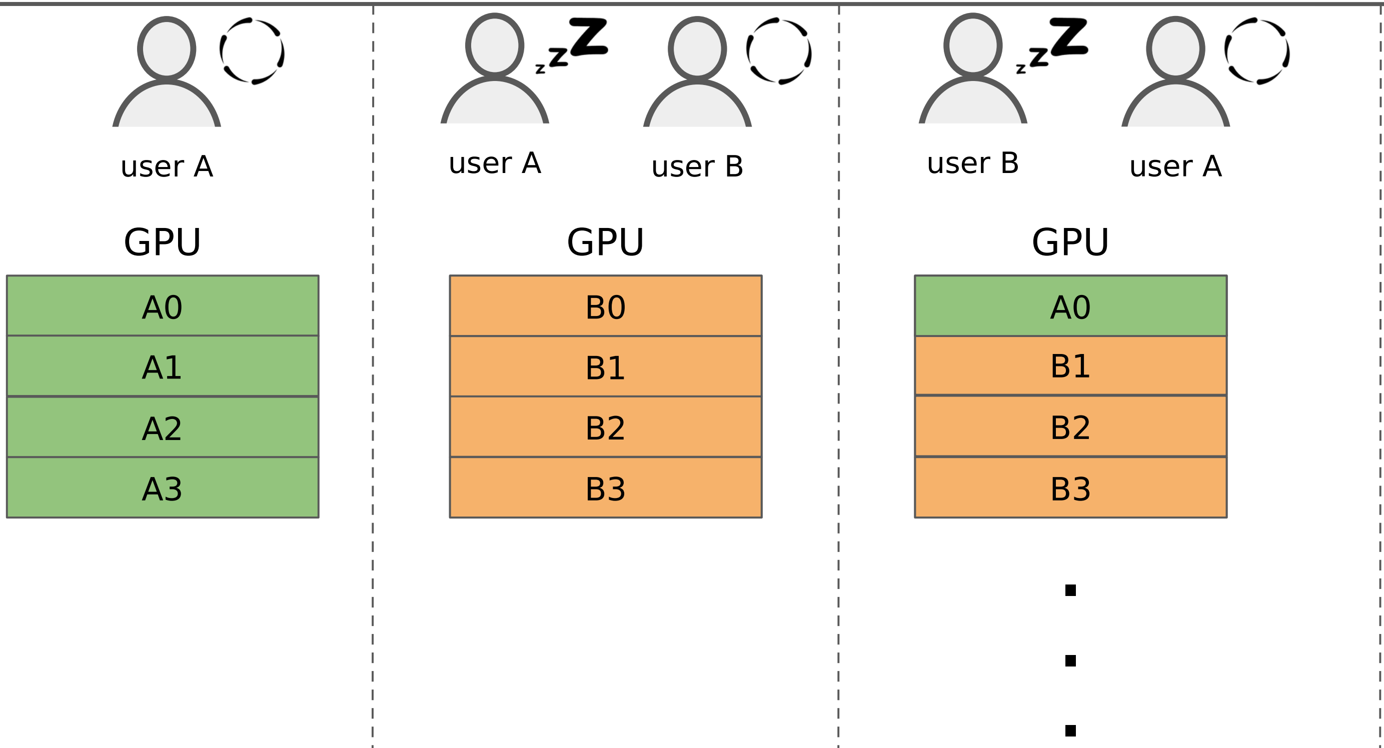


Result

- # of co-located processes limited by system RAM

Key Idea

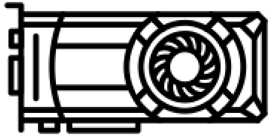


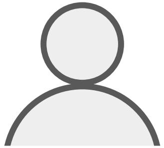


Result

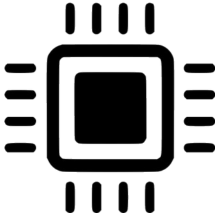
- # of co-located processes limited by system RAM
- each process can use the whole GPU memory

CUDA programming model

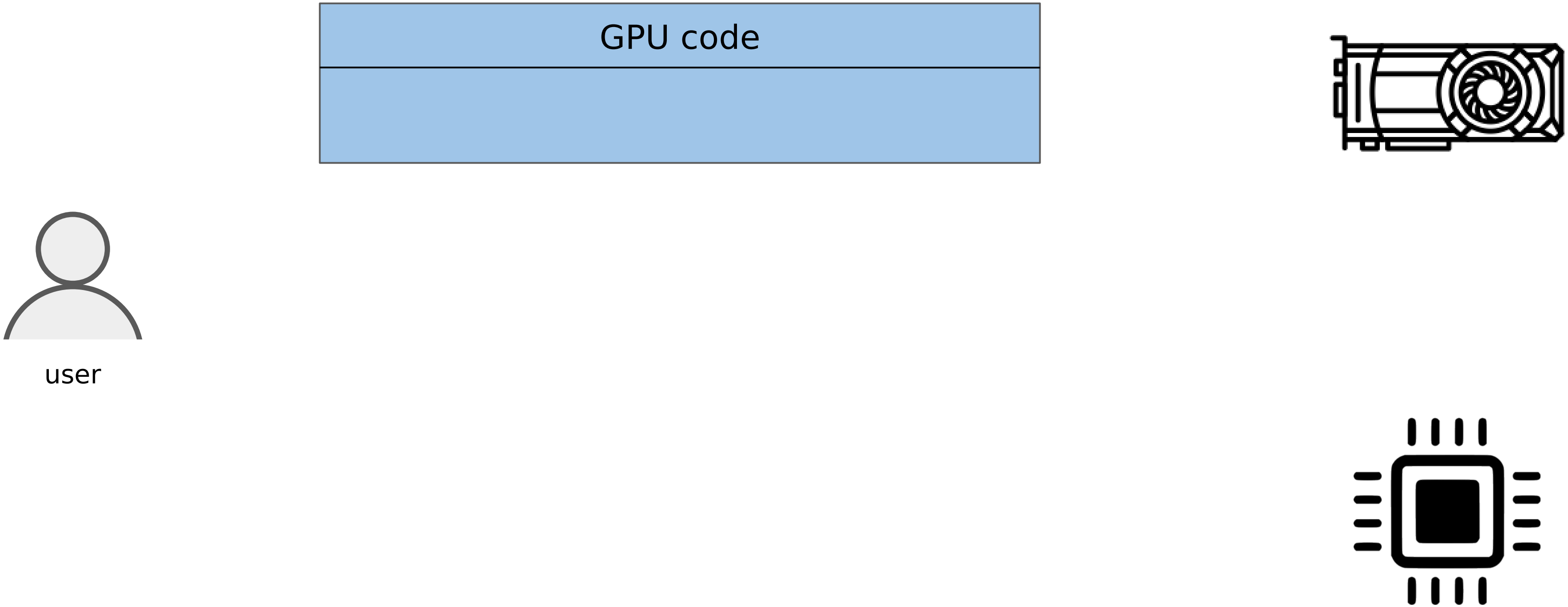




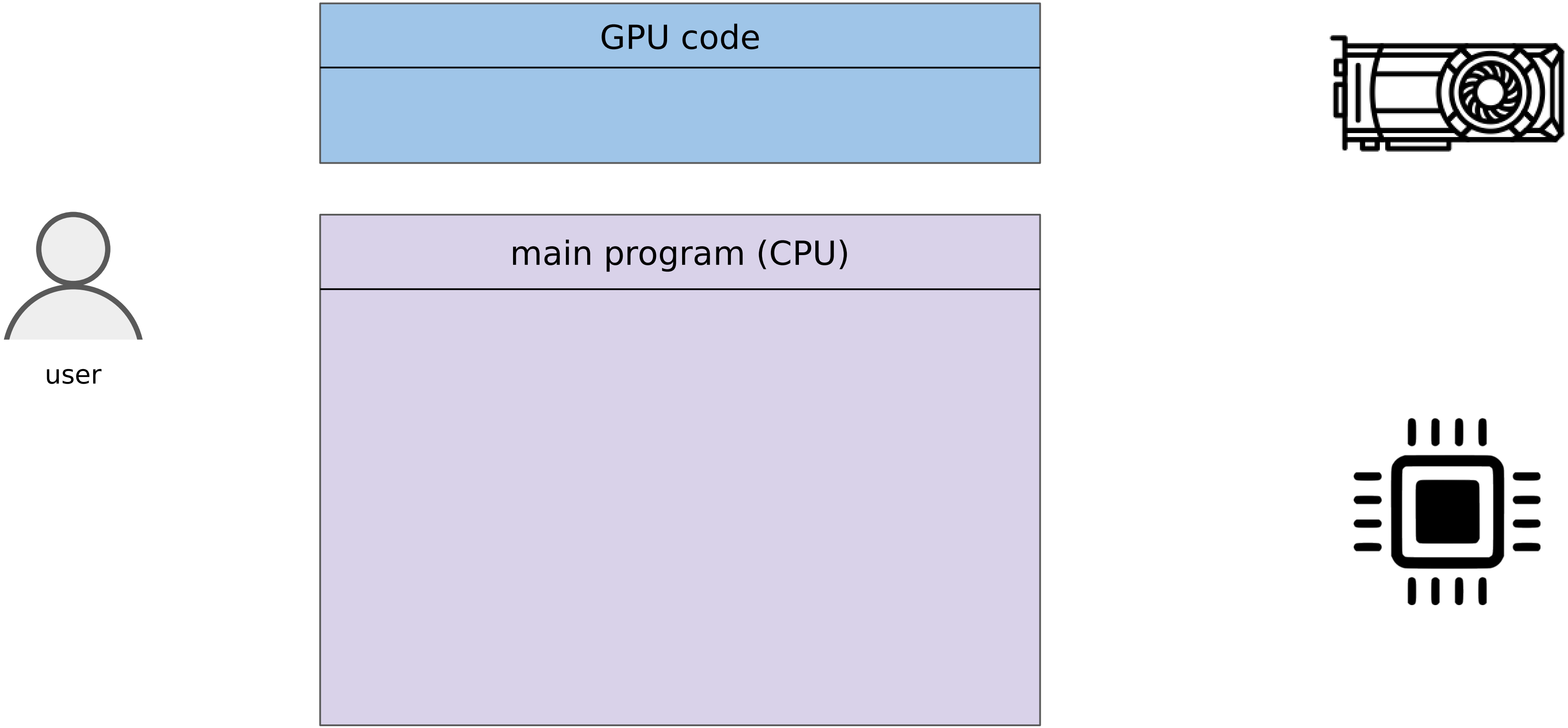
user



CUDA programming model



CUDA programming model

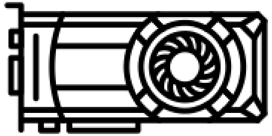


CUDA programming model

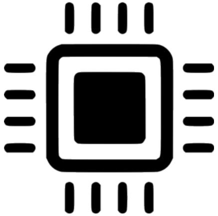
user

GPU code

```
__global__
void saxpy(int n, float a, float *x, float *y)
```



main program (CPU)

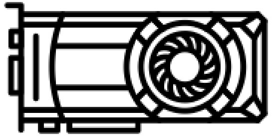


CUDA programming model

user

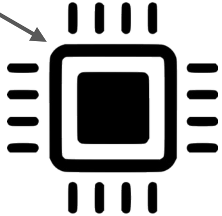
GPU code

```
__global__
void saxpy(int n, float a, float *x, float *y)
```



main program (CPU)

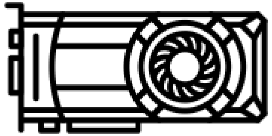
run

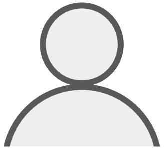


CUDA programming model

GPU code

```
__global__
void saxpy(int n, float a, float *x, float *y)
```

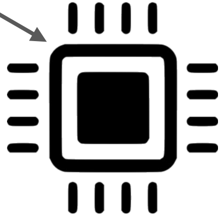




main program (CPU)

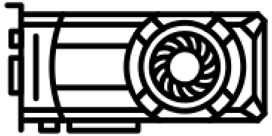
1.
```
malloc();
cudaMalloc();
```

run



CUDA programming model

user



GPU code

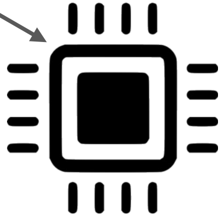
```
__global__
void saxpy(int n, float a, float *x, float *y)
```

main program (CPU)

1.
```
malloc();
cudaMalloc();
```

2.
```
cudaMemcpy(HostToDevice);
```

run

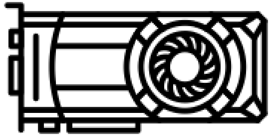


CUDA programming model

GPU code

```
__global__
void saxpy(int n, float a, float *x, float *y)
```

run



user

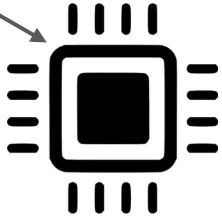
main program (CPU)

1.
```
malloc();
cudaMalloc();
```

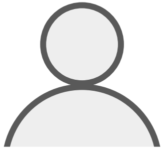
2.
```
cudaMemcpy(HostToDevice);
```

3.
```
saxpy<<<>>>();
```

run



CUDA programming model

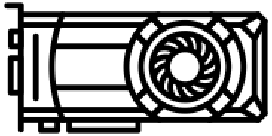


user

GPU code

```
__global__
void saxpy(int n, float a, float *x, float *y)
```

run



main program (CPU)

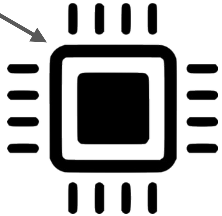
1.
```
malloc();
cudaMalloc();
```

2.
```
cudaMemcpy(HostToDevice);
```

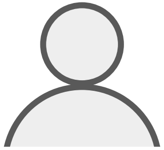
3.
```
saxpy<<<>>>();
```

asynchronous!

run



CUDA programming model

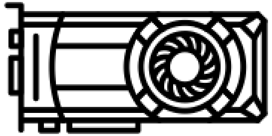


user

GPU code

```
__global__
void saxpy(int n, float a, float *x, float *y)
```

run



main program (CPU)

1.
```
malloc();
cudaMalloc();
```

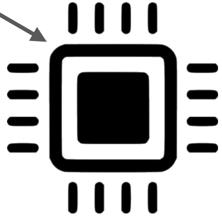
2.
```
cudaMemcpy(HostToDevice);
```

3.
```
saxpy<<<>>>();
```

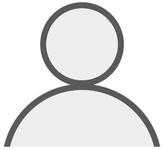
asynchronous!

```
cudaDeviceSynchronize();
```

run



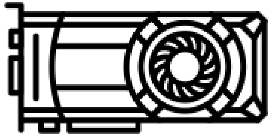
CUDA programming model



GPU code

```
__global__
void saxpy(int n, float a, float *x, float *y)
```

run



main program (CPU)

1.
```
malloc();
cudaMalloc();
```

2.
```
cudaMemcpy(HostToDevice);
```

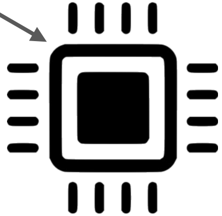
3.
```
saxpy<<<>>>();
```

asynchronous!

```
cudaDeviceSynchronize();
```

4.
```
cudaMemcpy(DeviceToHost);
```

run



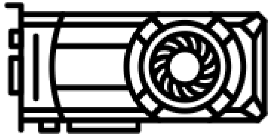
CUDA programming model

user

GPU code

```
__global__
void saxpy(int n, float a, float *x, float *y)
```

run



main program (CPU)

1.
```
malloc();
cudaMalloc();
```

2.
```
cudaMemcpy(HostToDevice);
```

3.
```
saxpy<<<>>>();
```

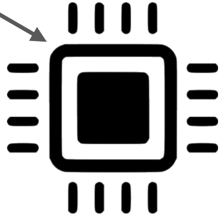
asynchronous!

```
cudaDeviceSynchronize();
```

4.
```
cudaMemcpy(DeviceToHost);
```

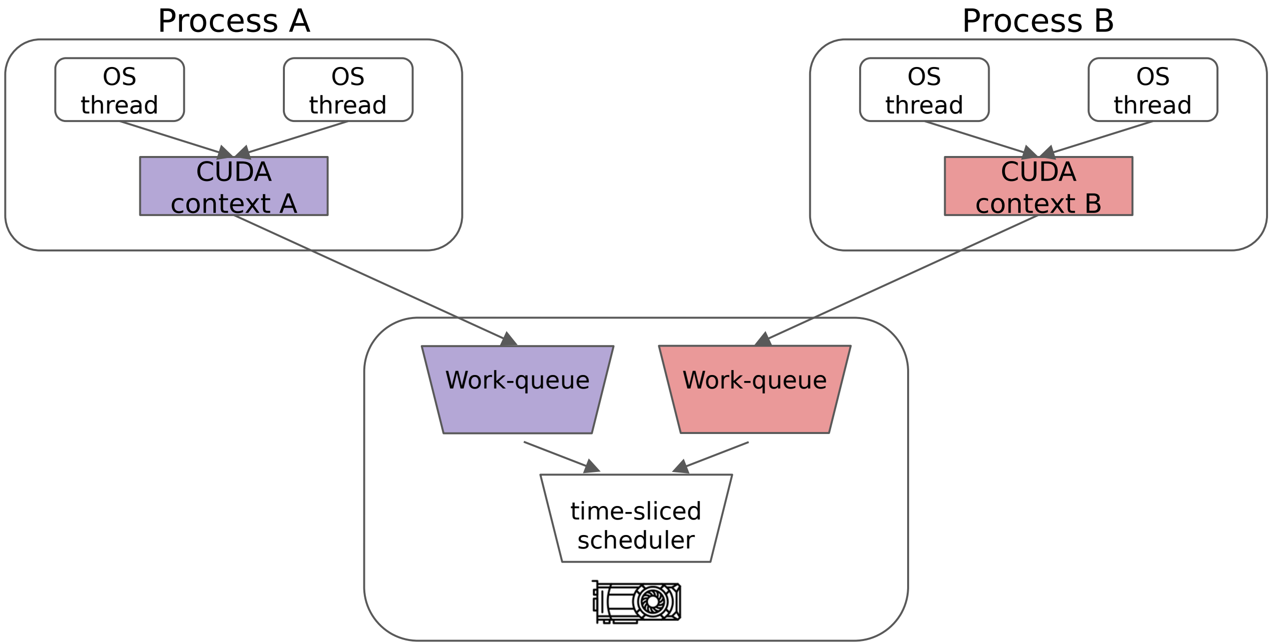
5.
```
cudaFree();
free();
```

run

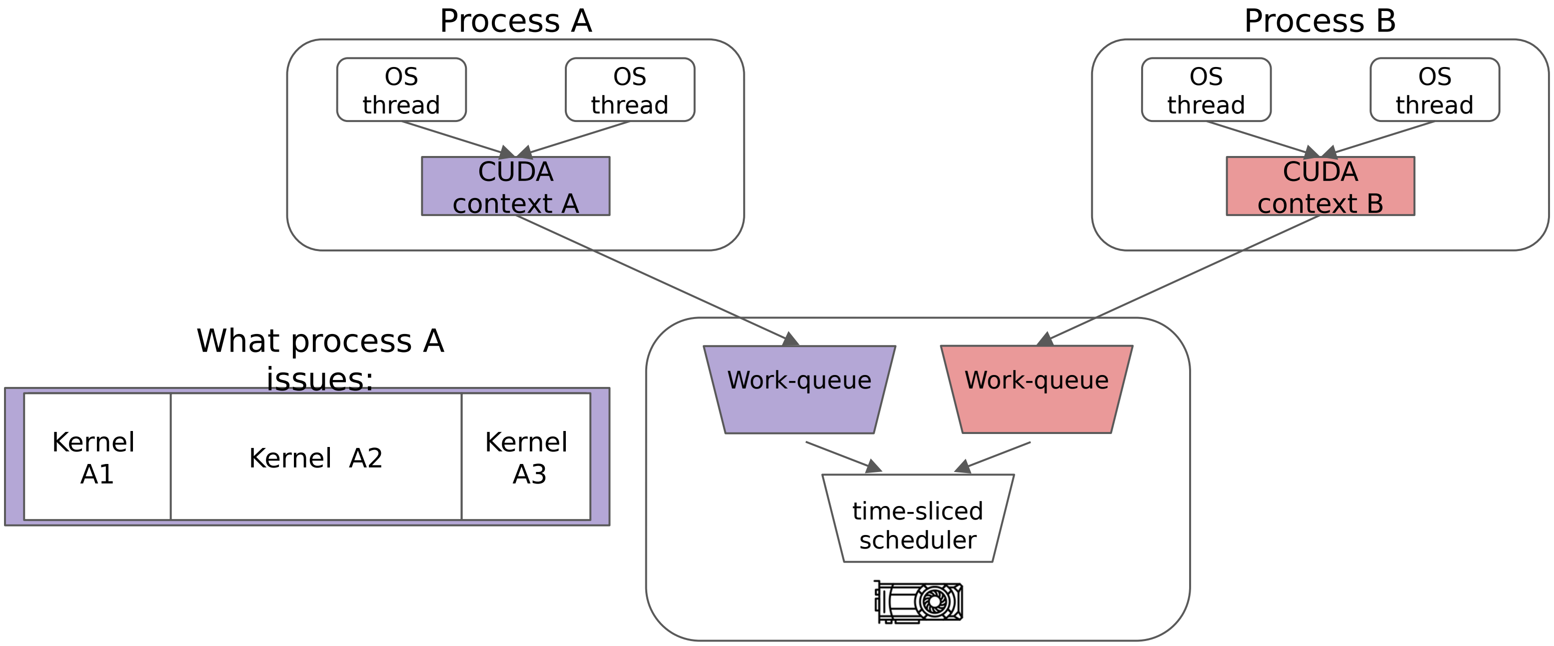


CUDA multi-processing

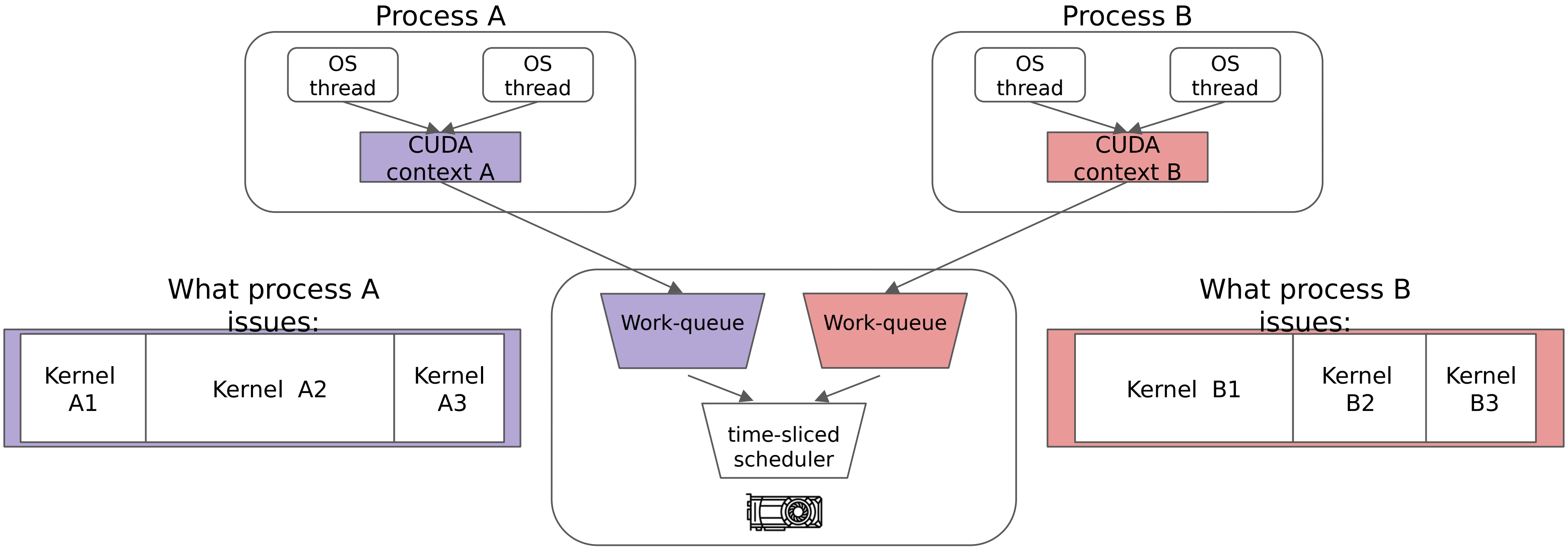
CUDA multi-processing



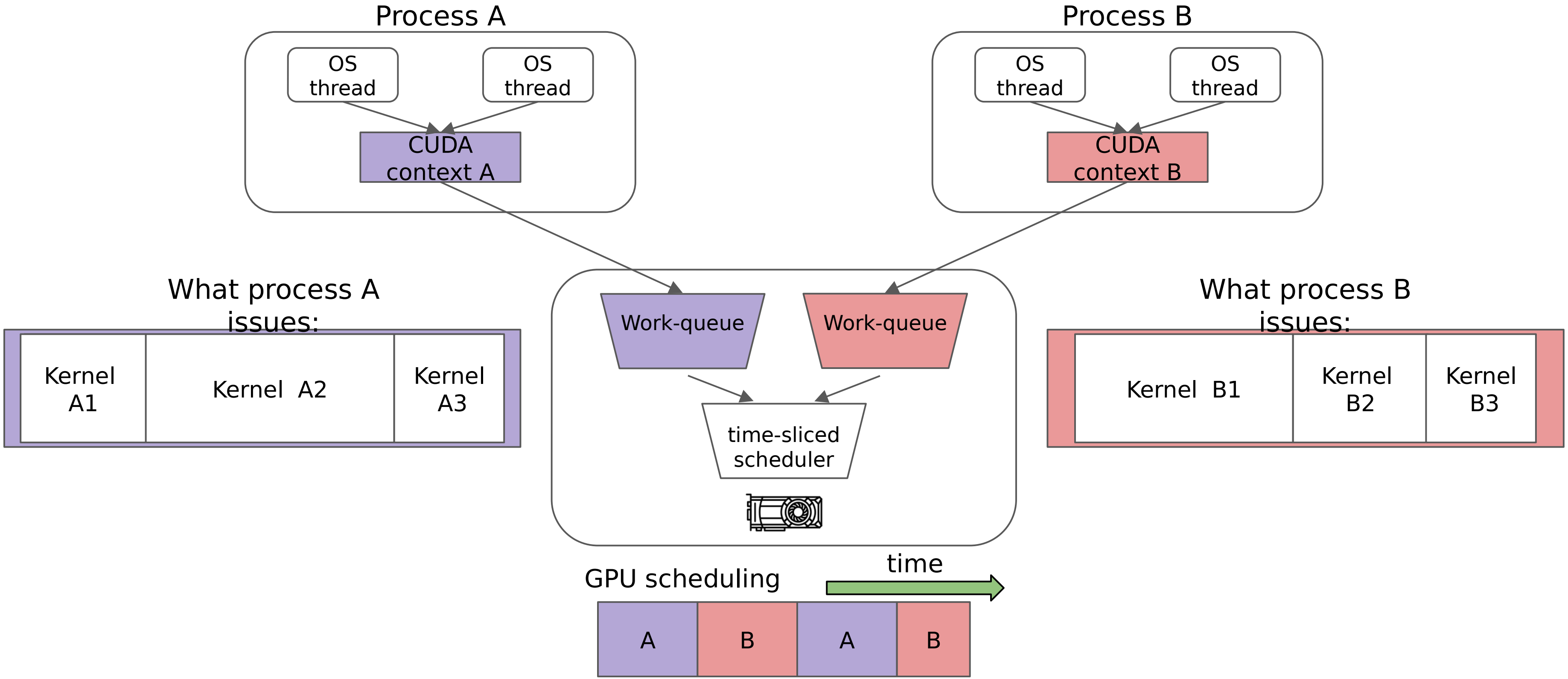
CUDA multi-processing



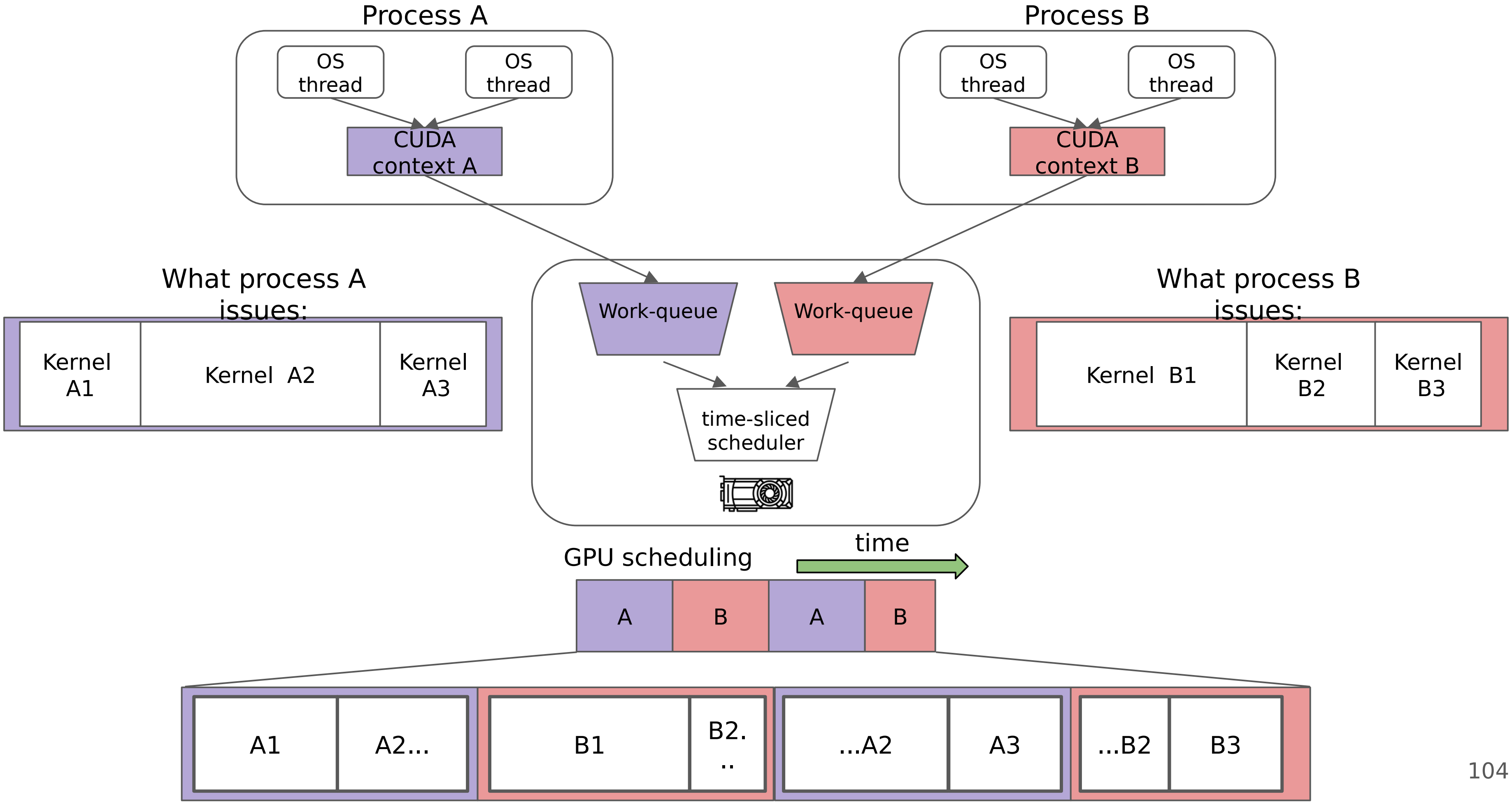
CUDA multi-processing



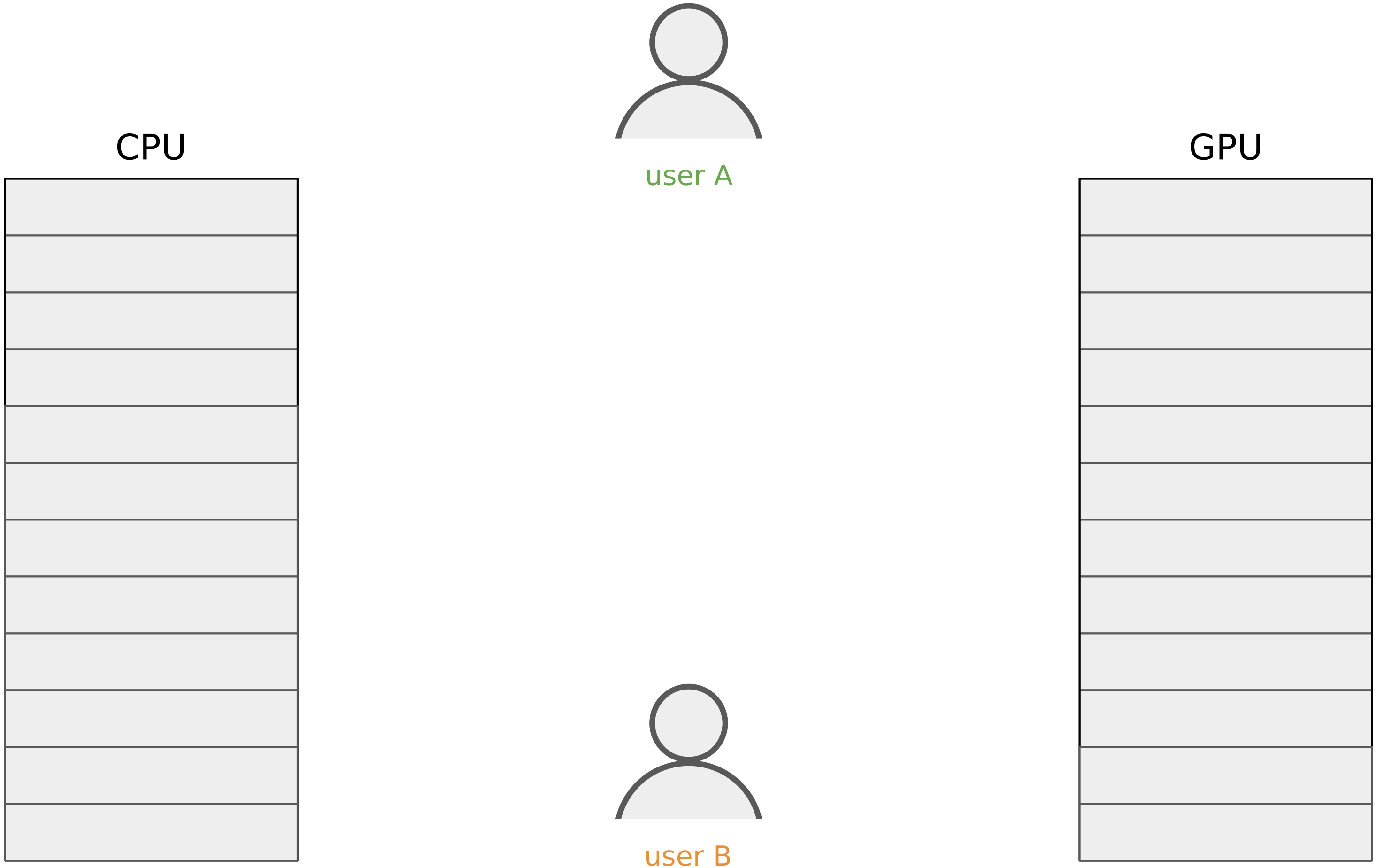
CUDA multi-processing



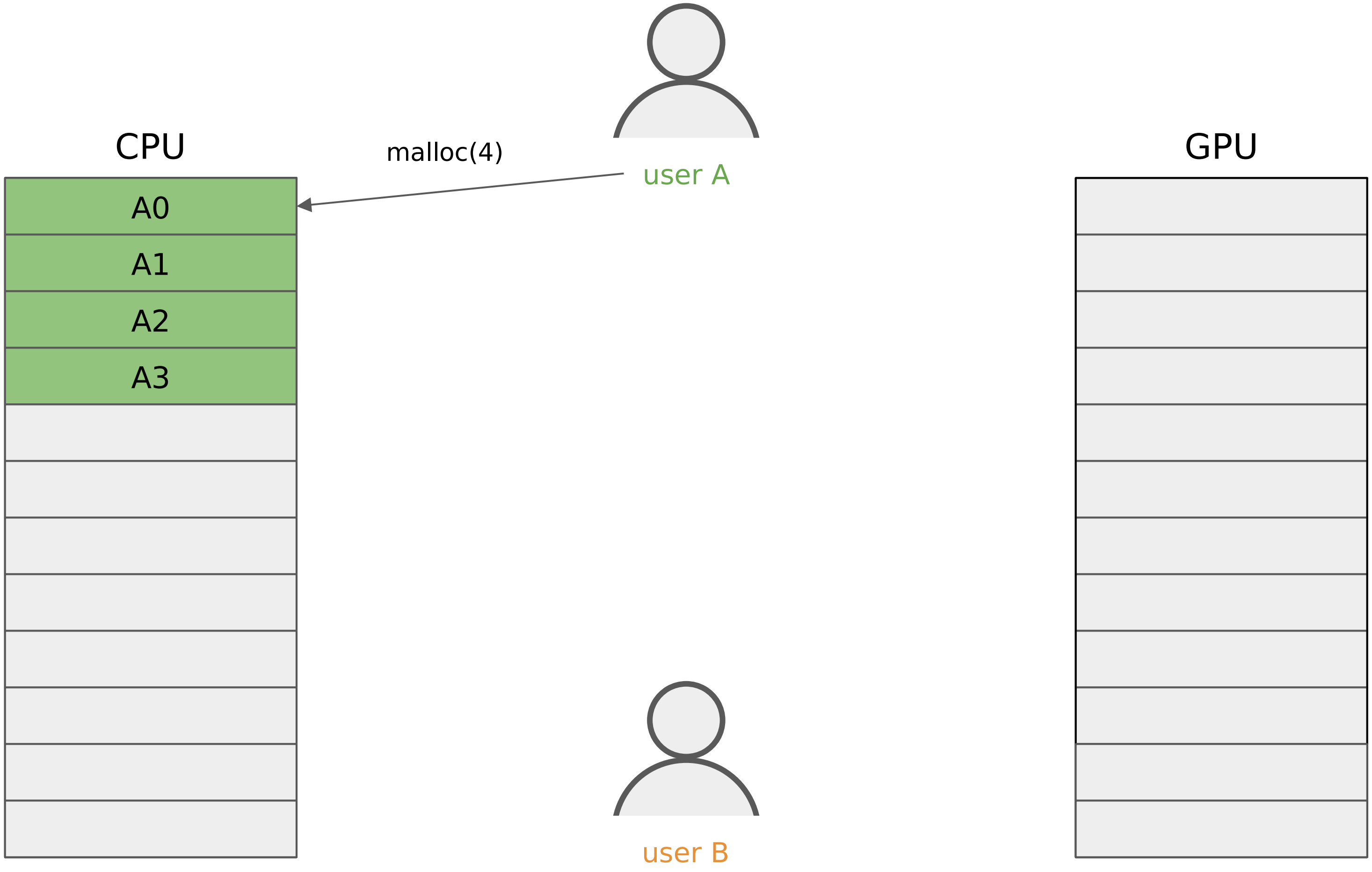
CUDA multi-processing



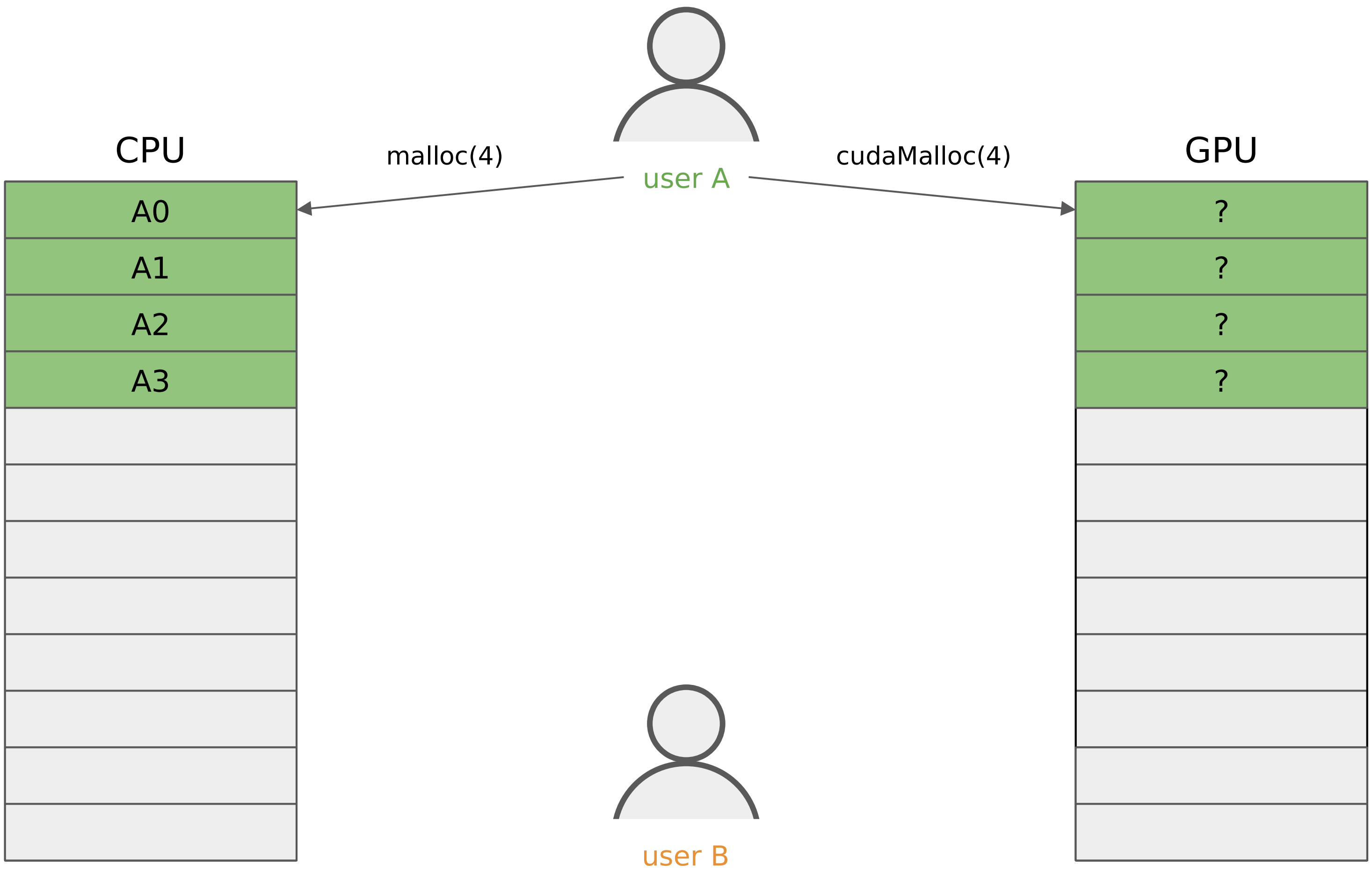
Default CUDA Memory Model



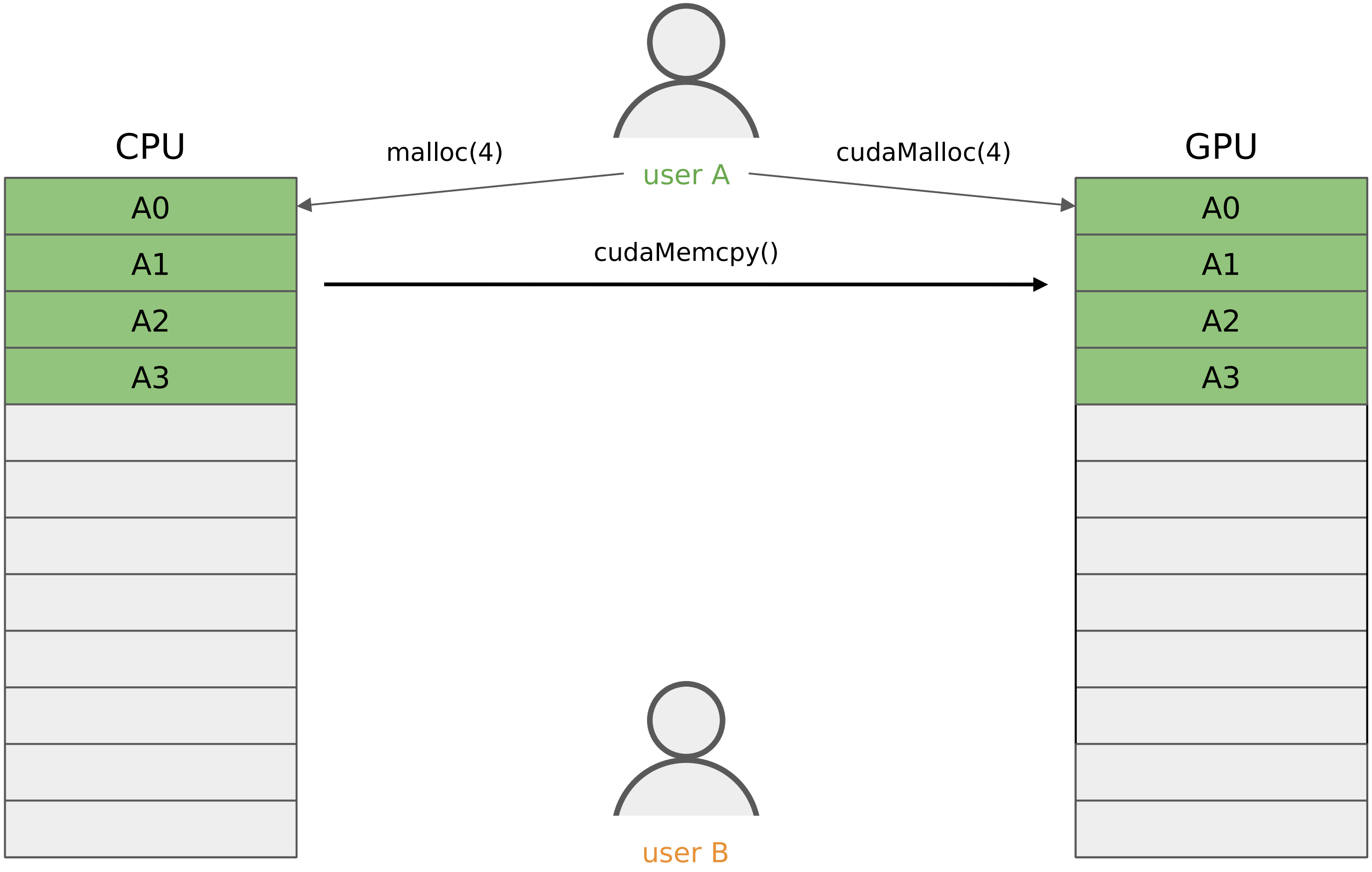
Default CUDA Memory Model



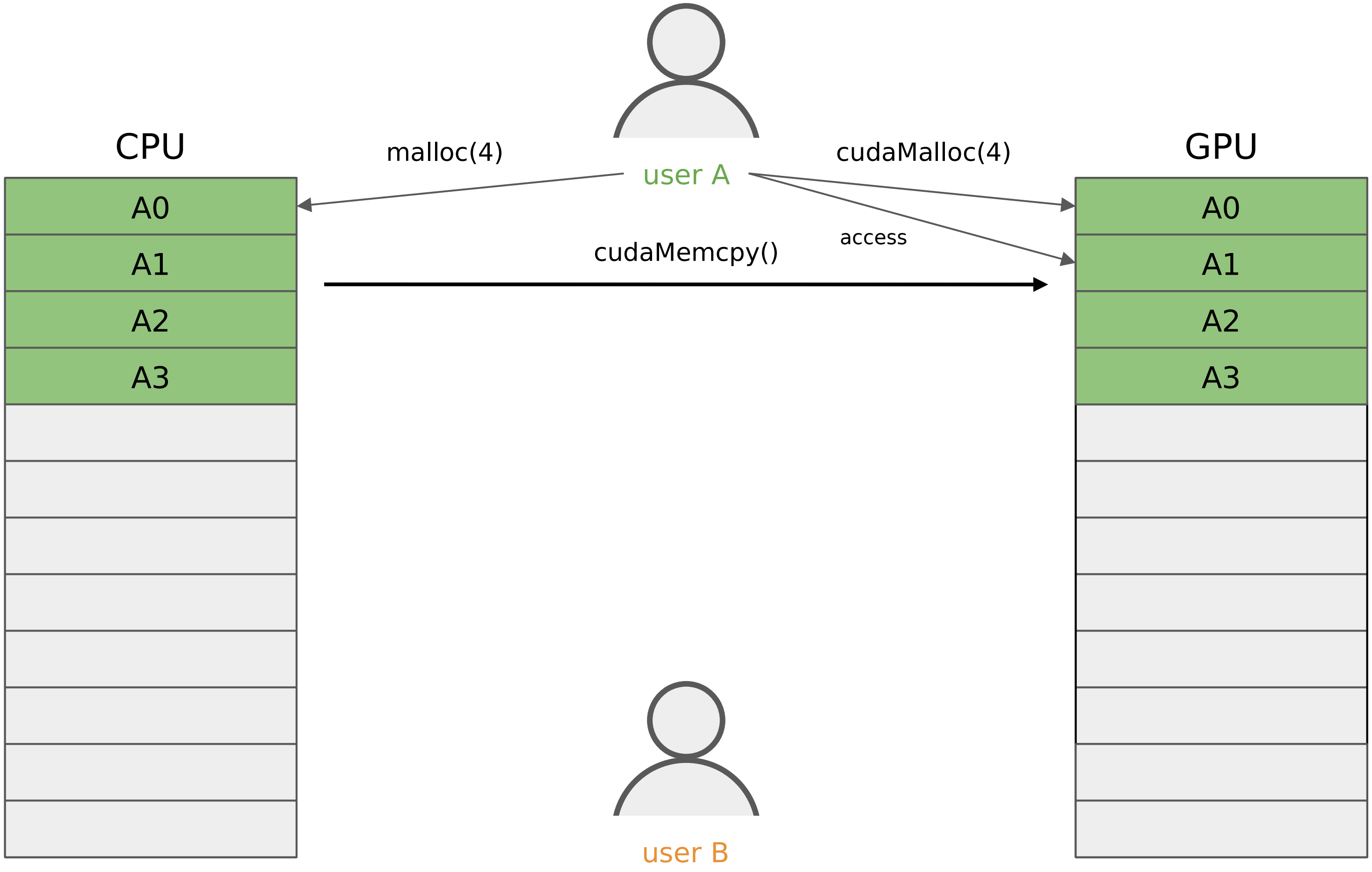
Default CUDA Memory Model



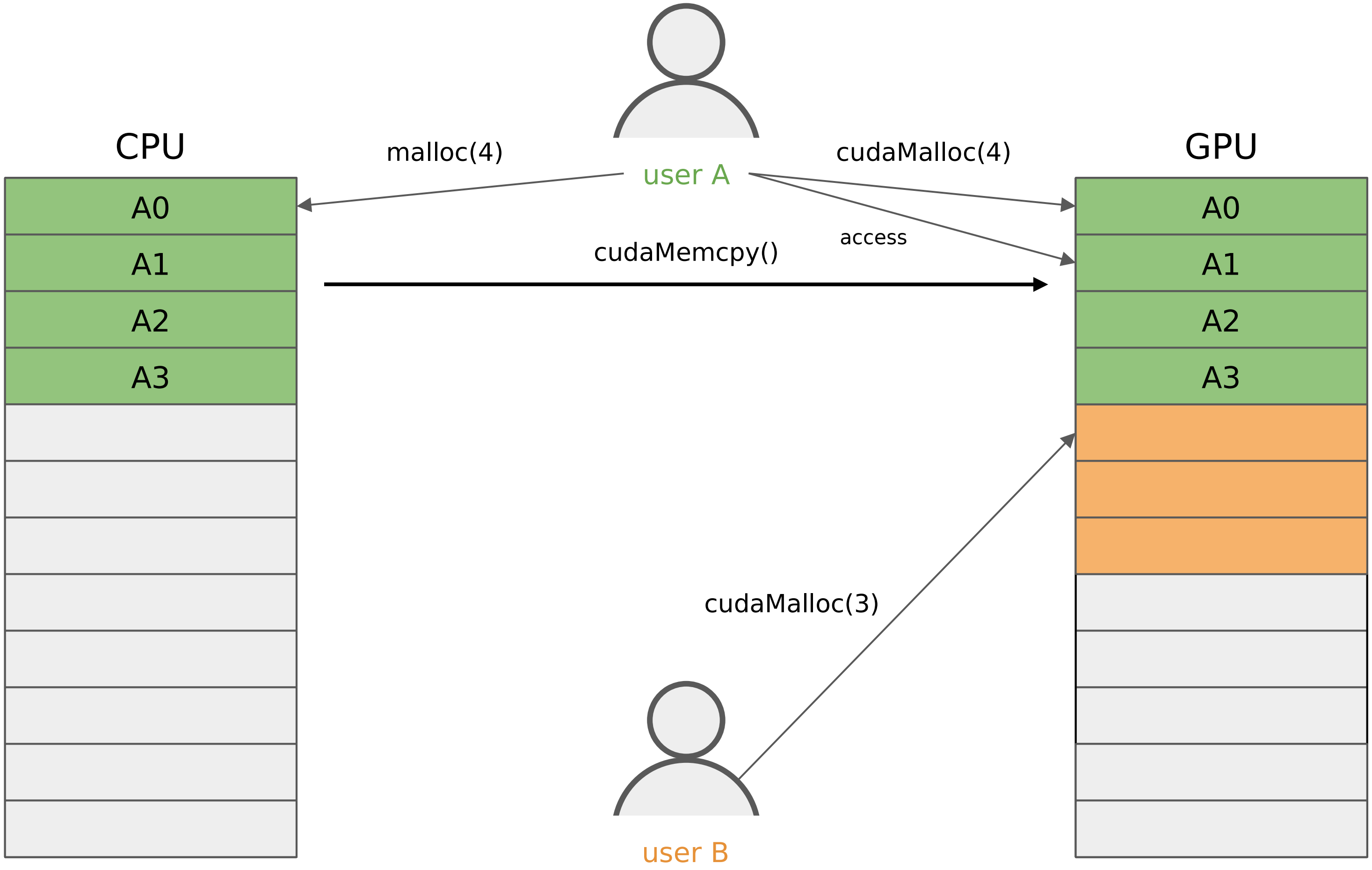
Default CUDA Memory Model



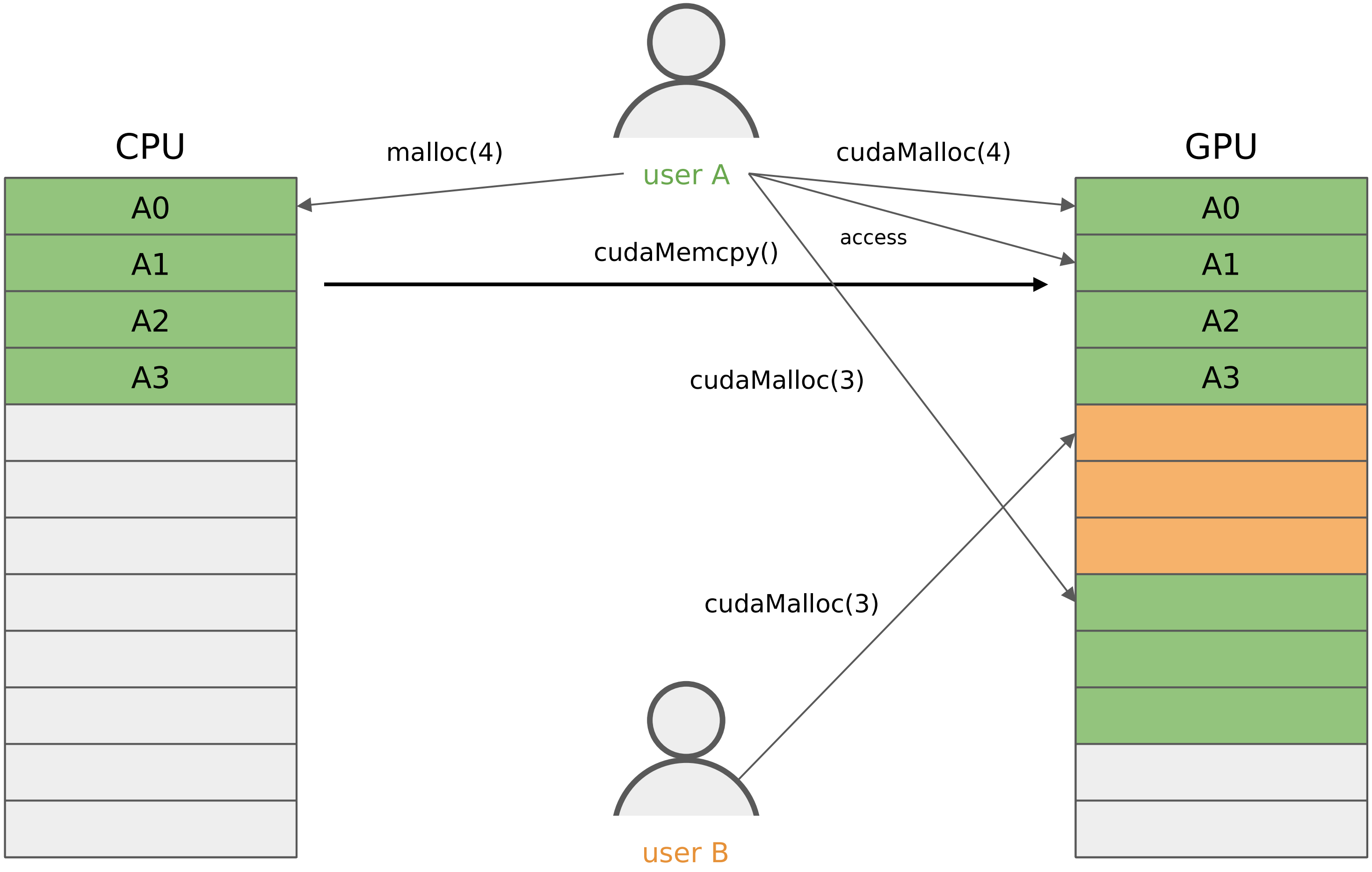
Default CUDA Memory Model



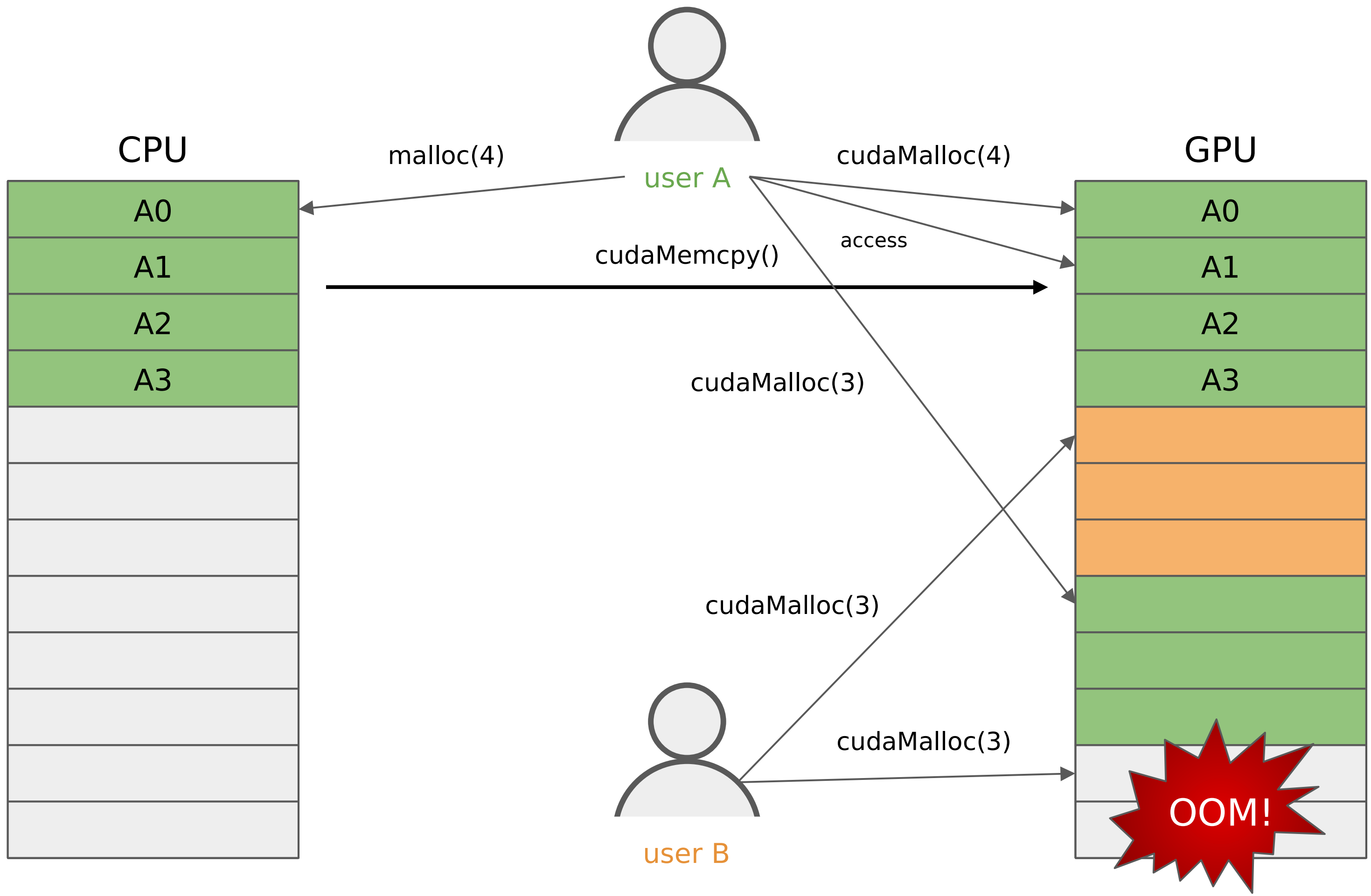
Default CUDA Memory Model



Default CUDA Memory Model

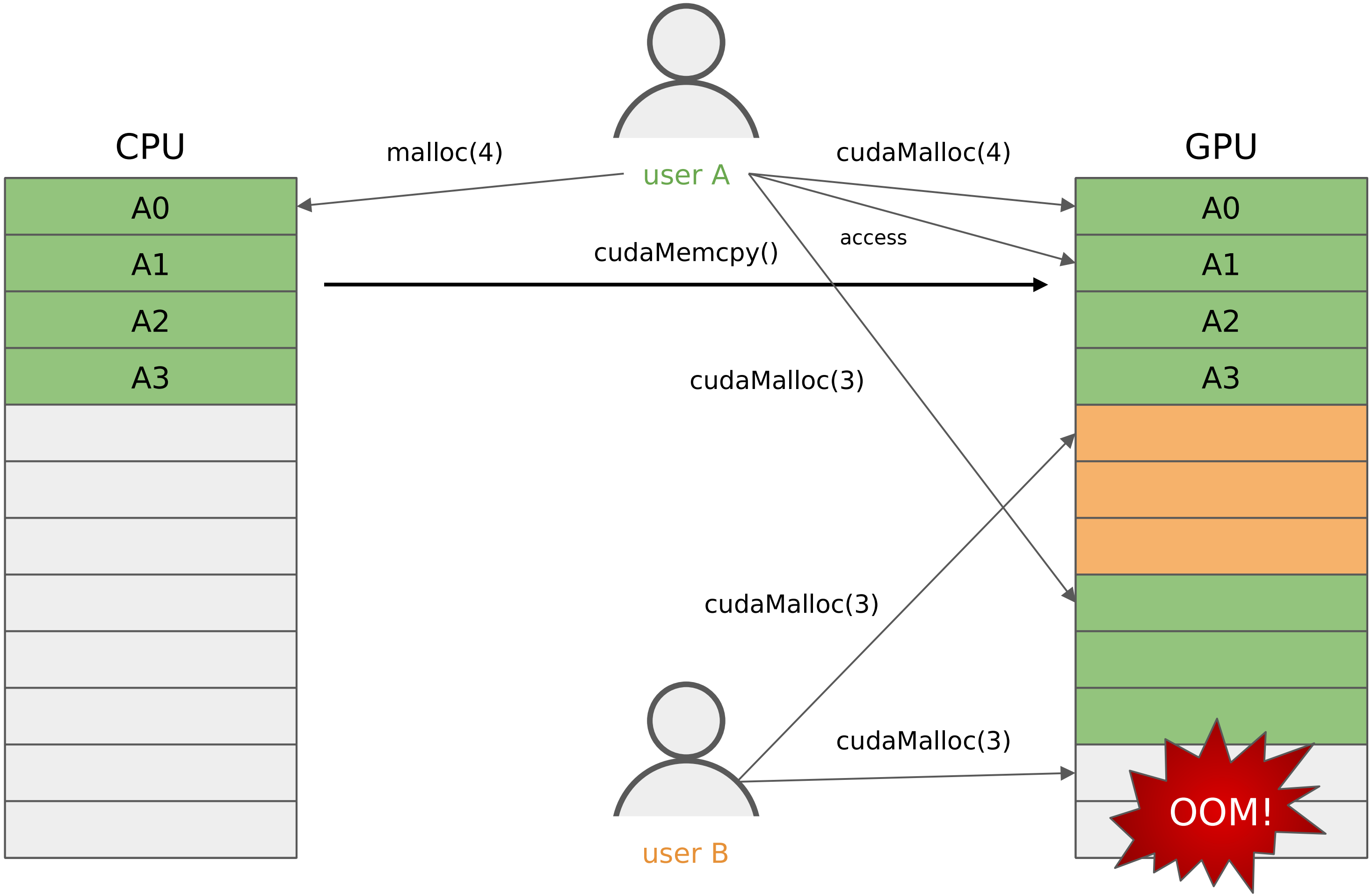


Default CUDA Memory Model



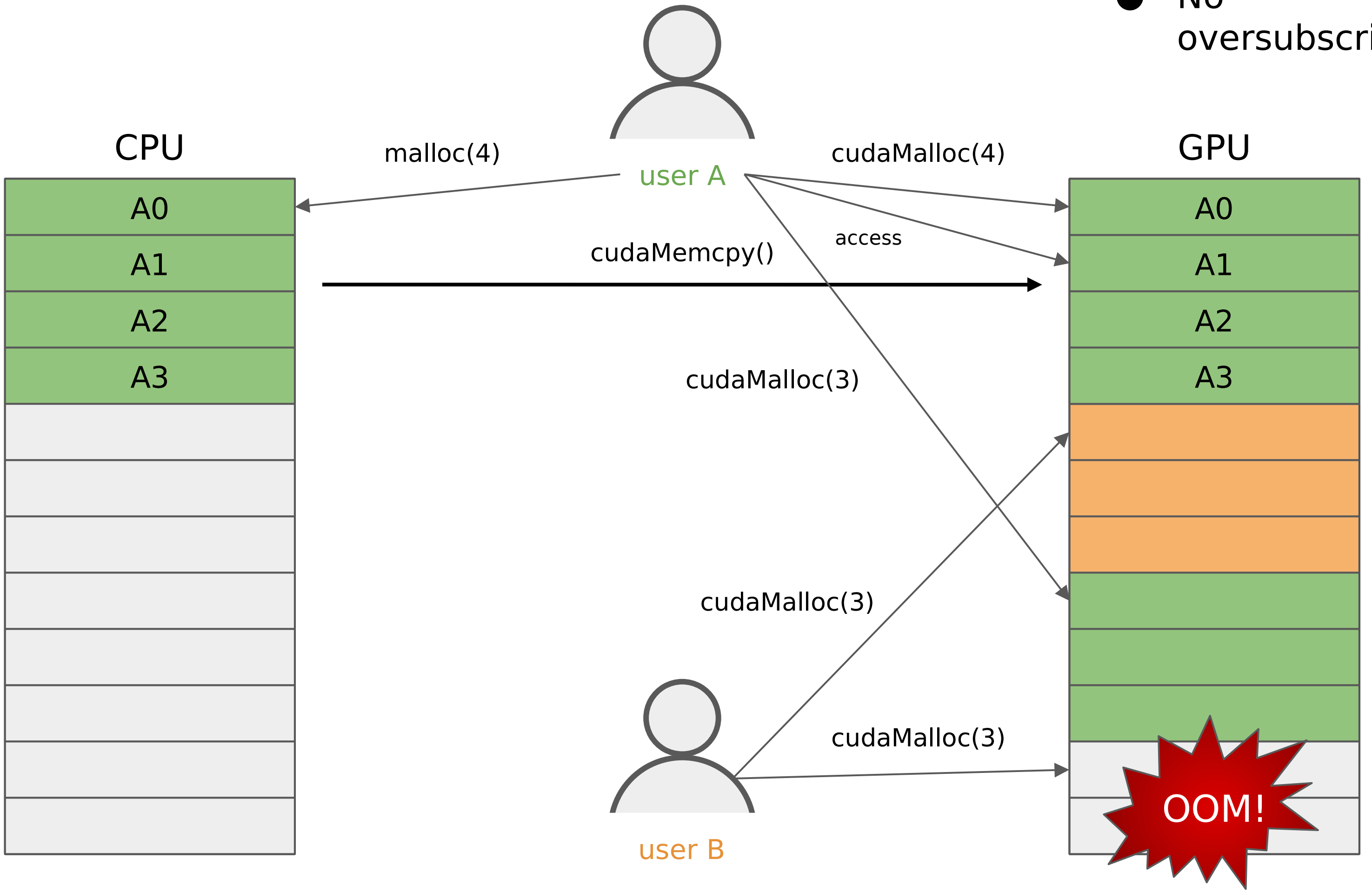
Default CUDA Memory Model

- FCFS



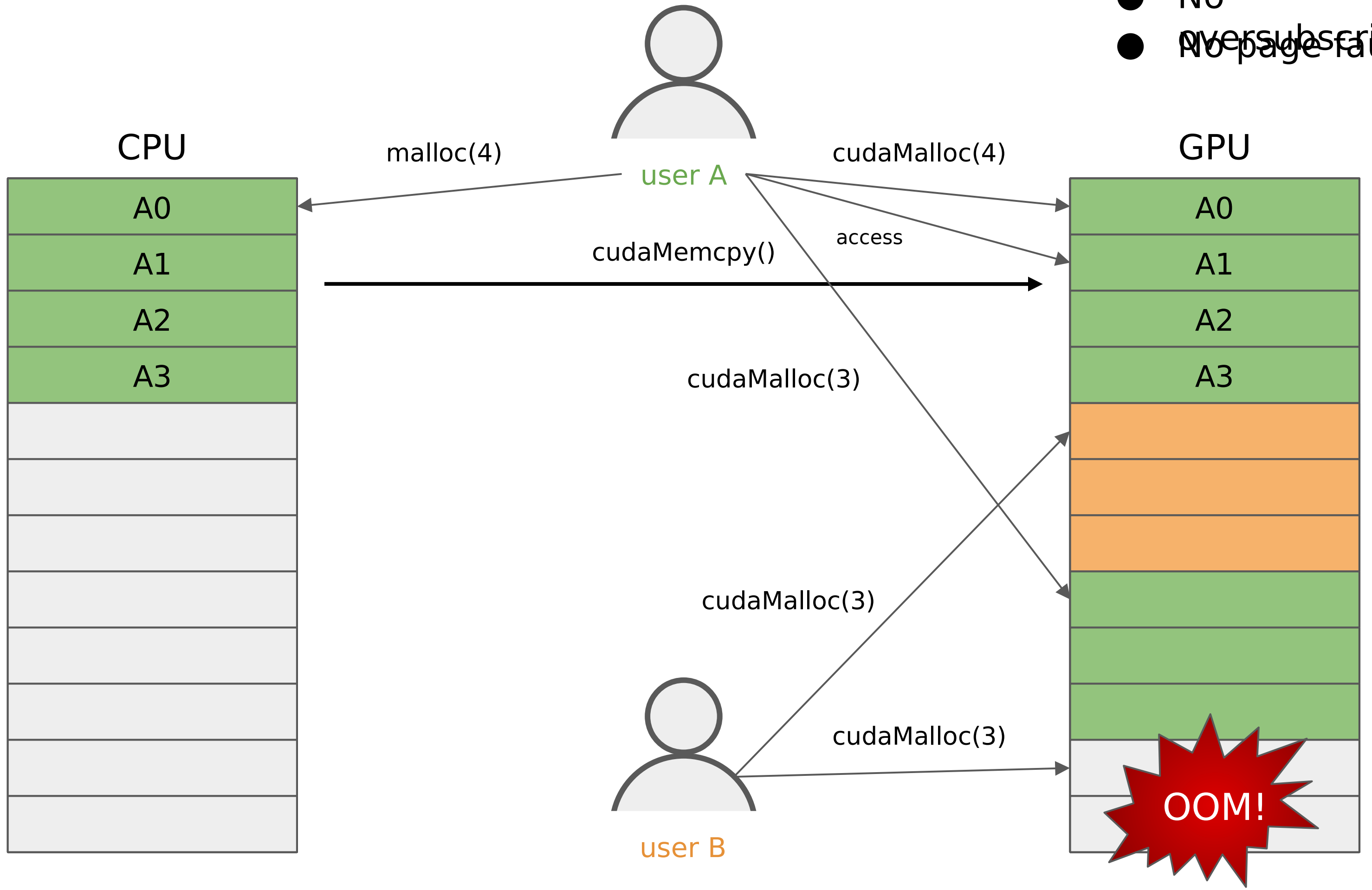
Default CUDA Memory Model

- FCFS
- No oversubscription



Default CUDA Memory Model

- FCFS
- No oversubscription
- No page faults

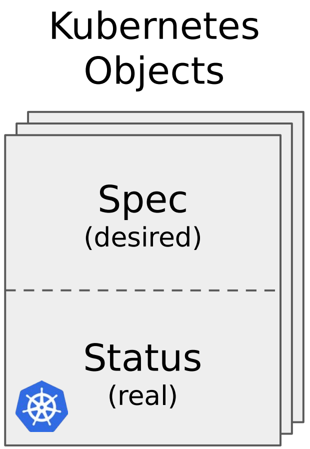


Controller Pattern

Used ***everywhere*** in Kubernetes

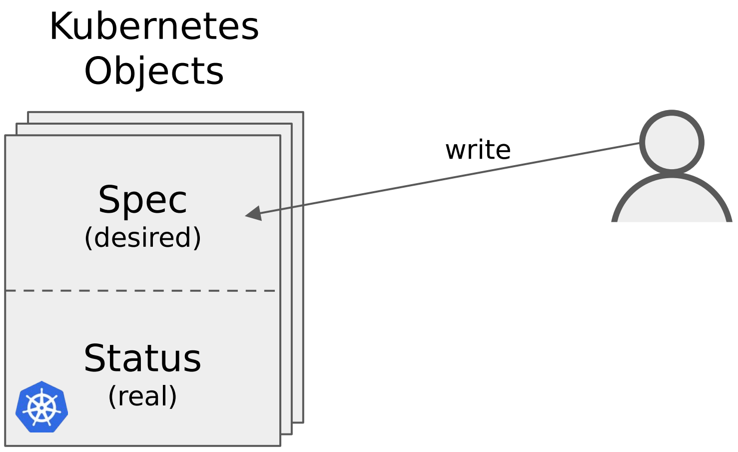
Controller Pattern

Used ***everywhere*** in Kubernetes



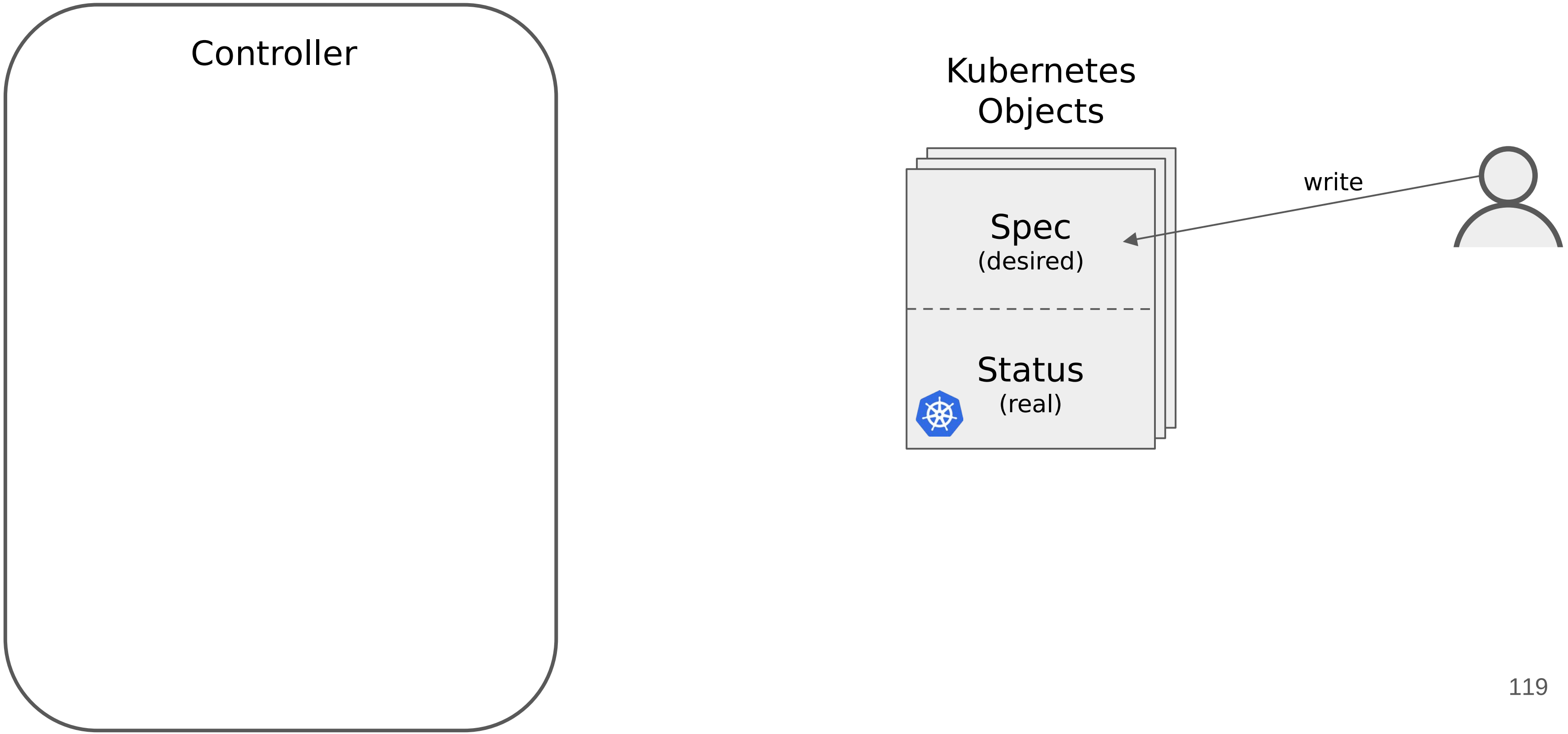
Controller Pattern

Used ***everywhere*** in Kubernetes



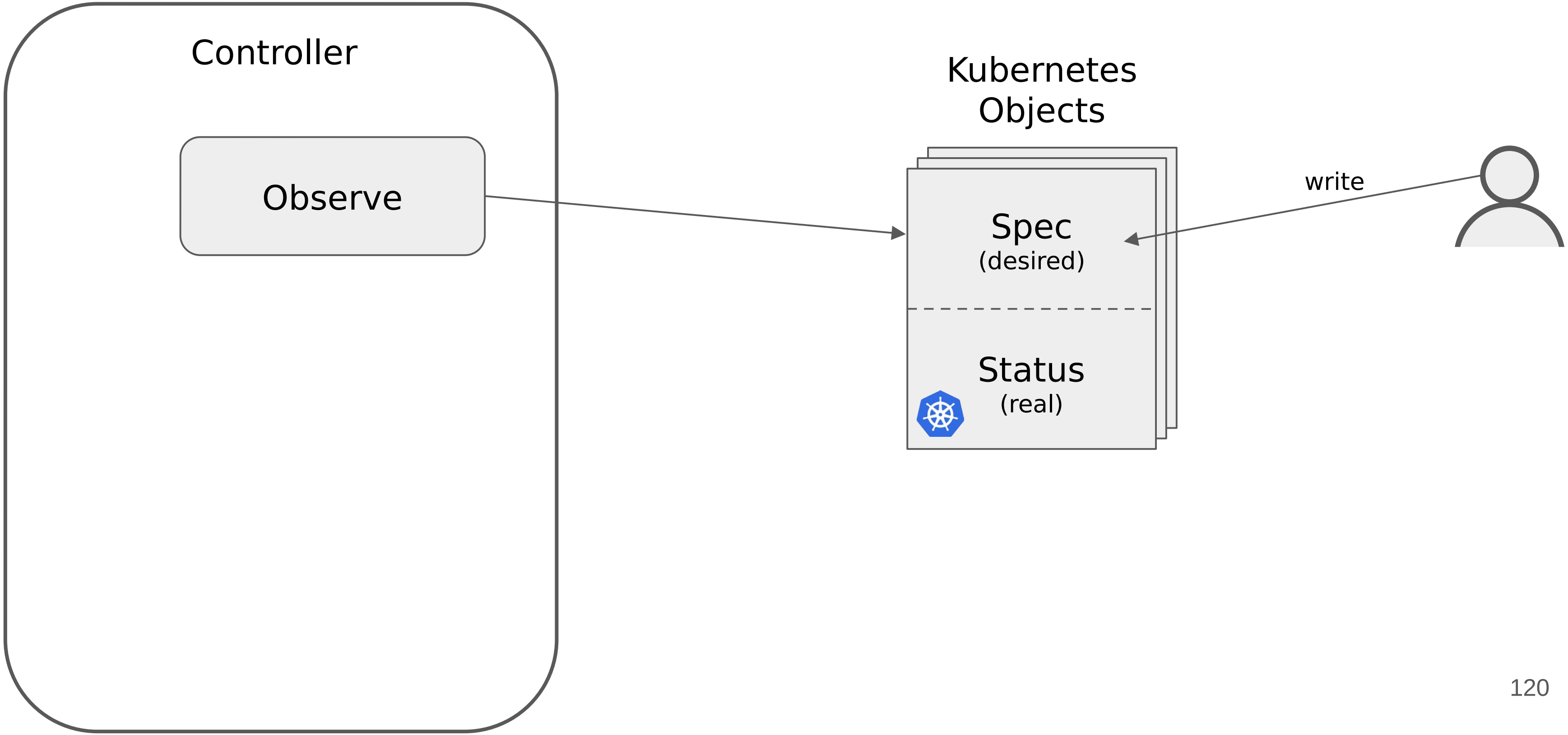
Controller Pattern

Used ***everywhere*** in Kubernetes



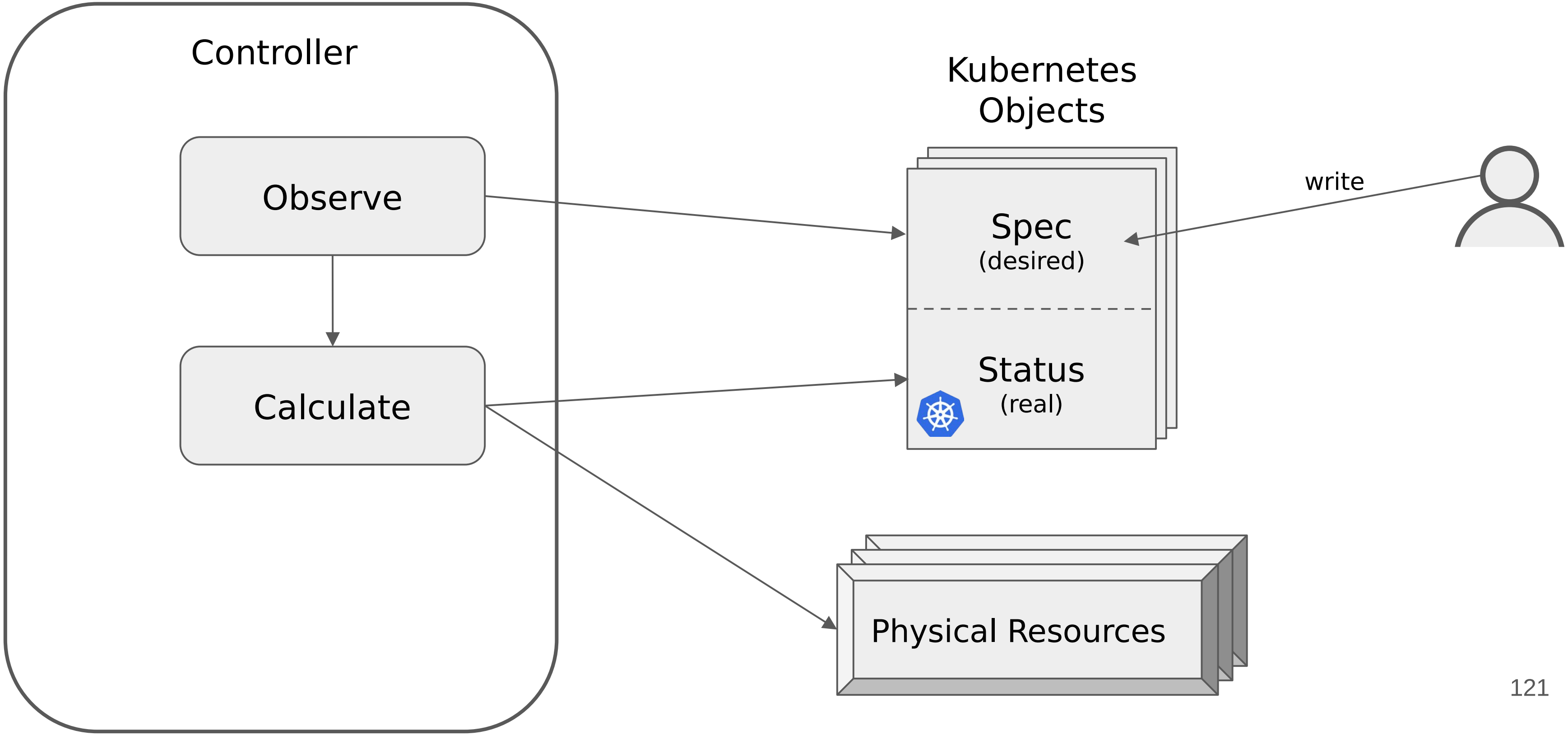
Controller Pattern

Used ***everywhere*** in Kubernetes



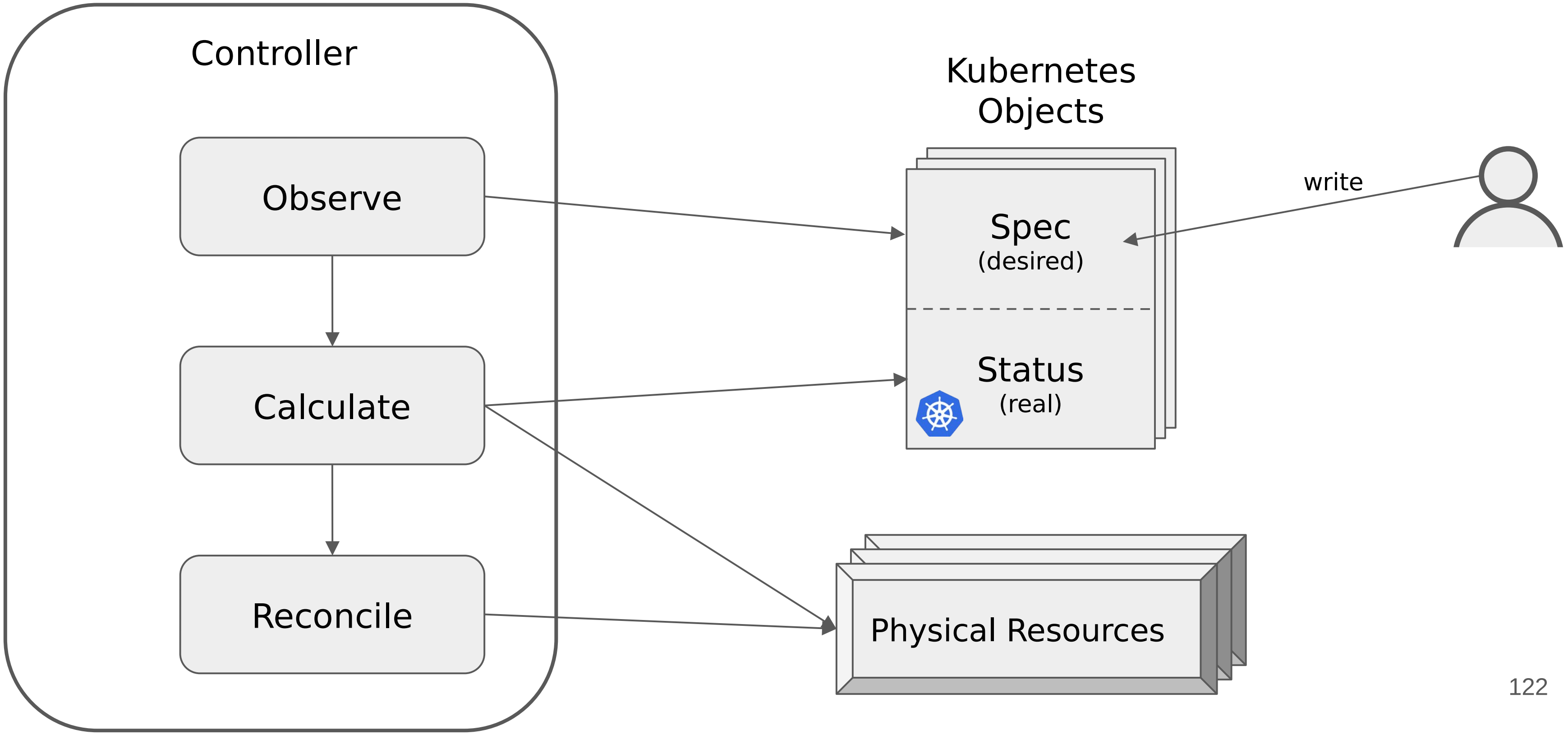
Controller Pattern

Used ***everywhere*** in Kubernetes



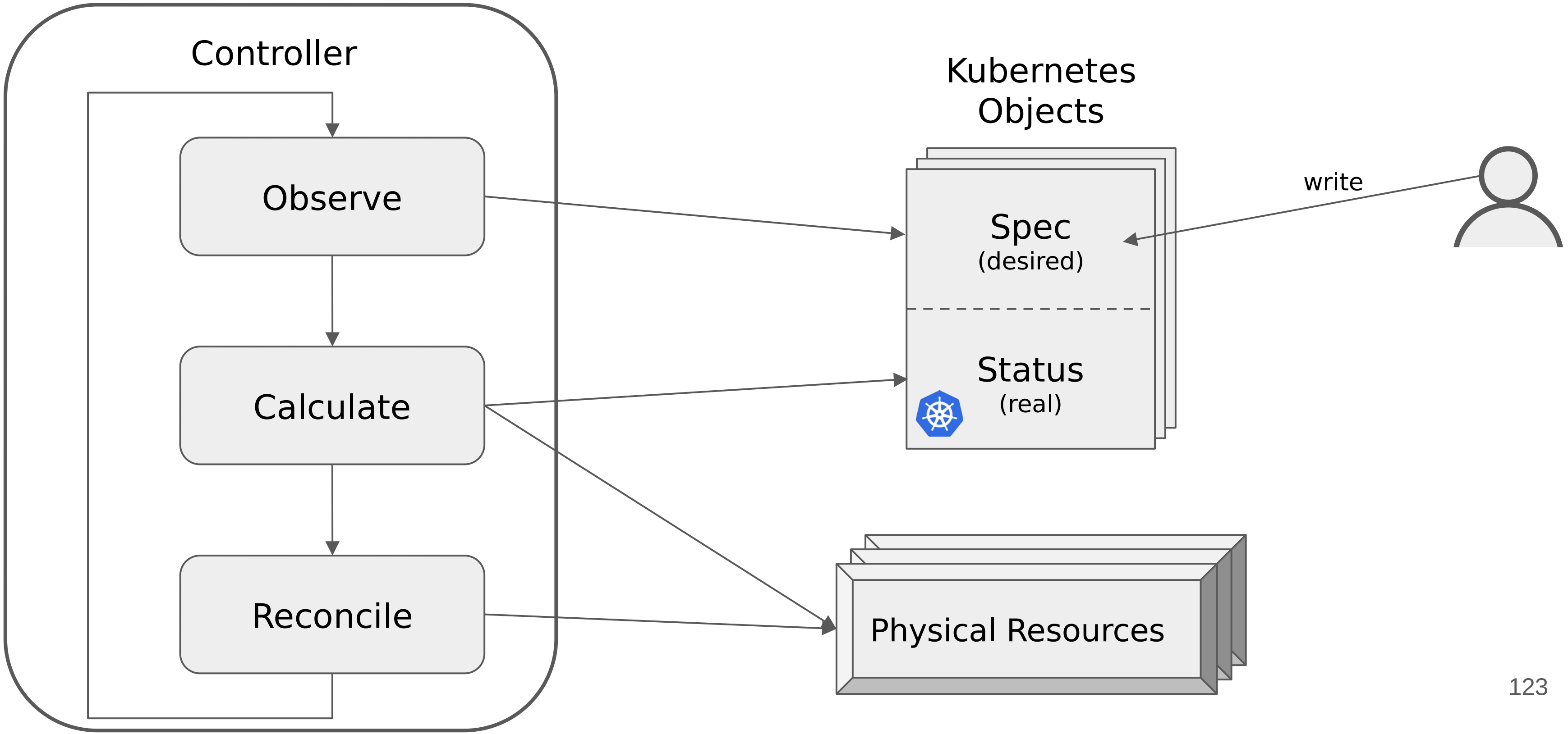
Controller Pattern

Used ***everywhere*** in Kubernetes

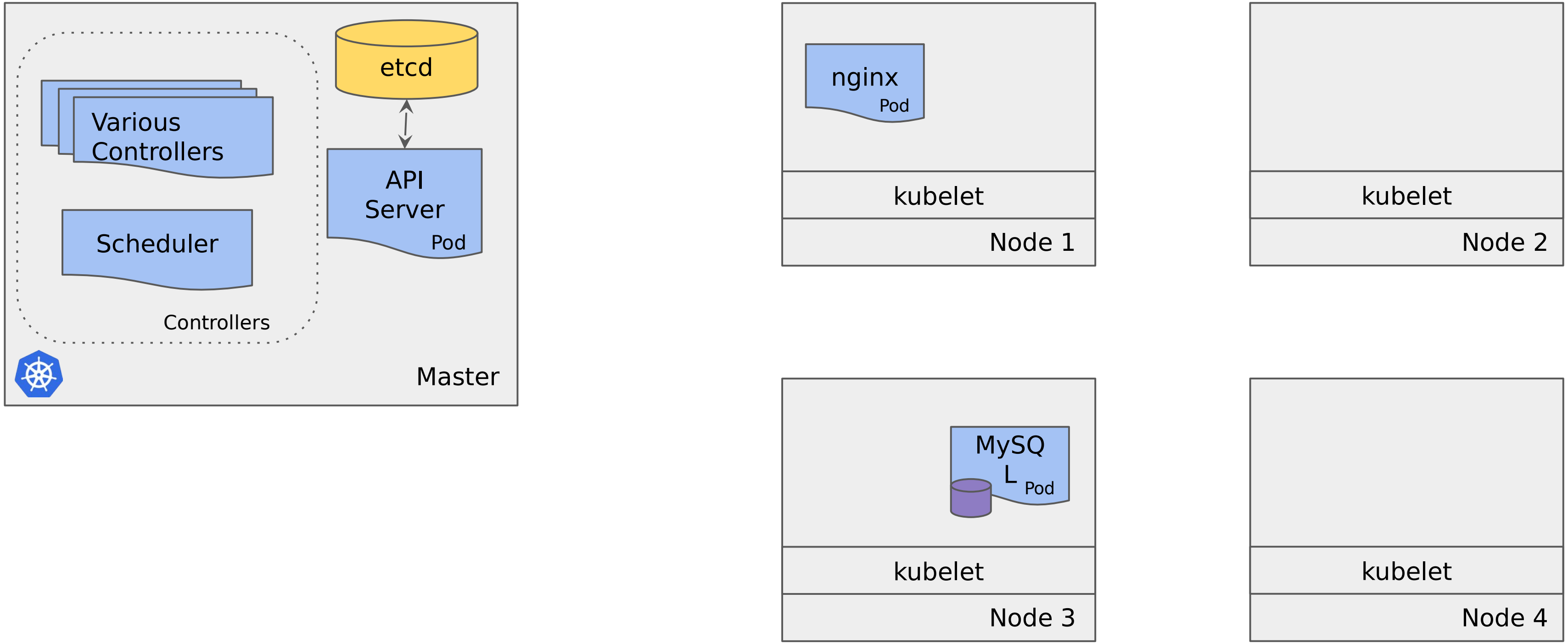


Controller Pattern

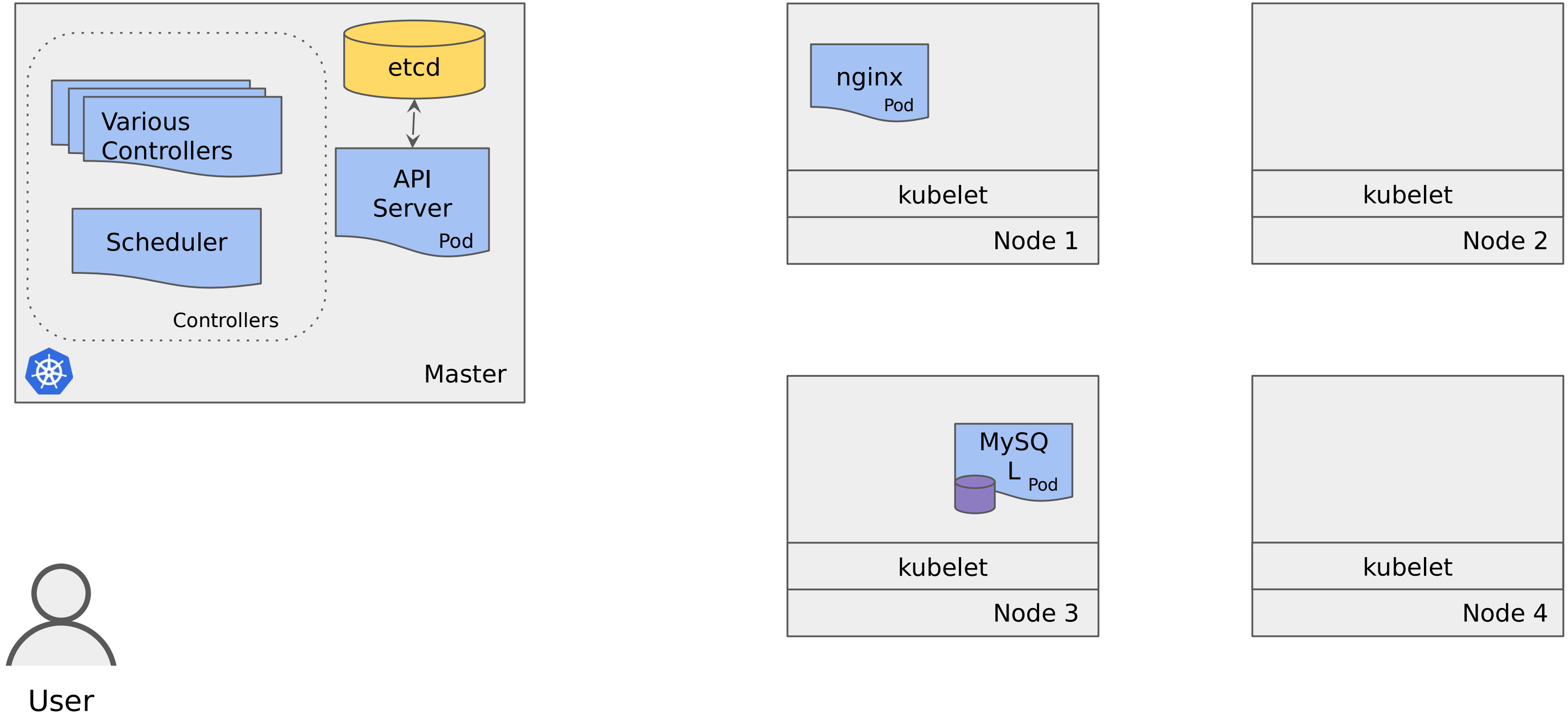
Used ***everywhere*** in Kubernetes



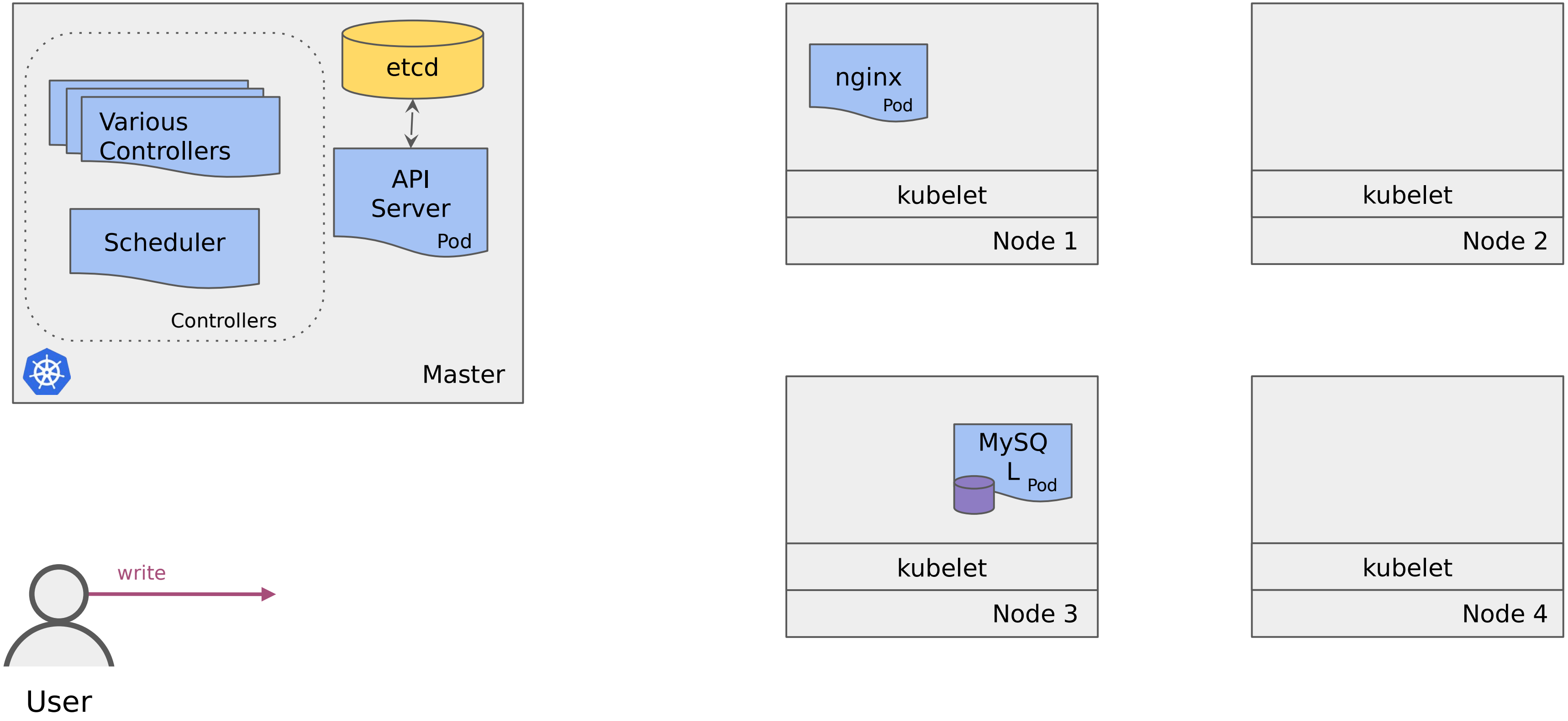
Kubernetes Scheduling



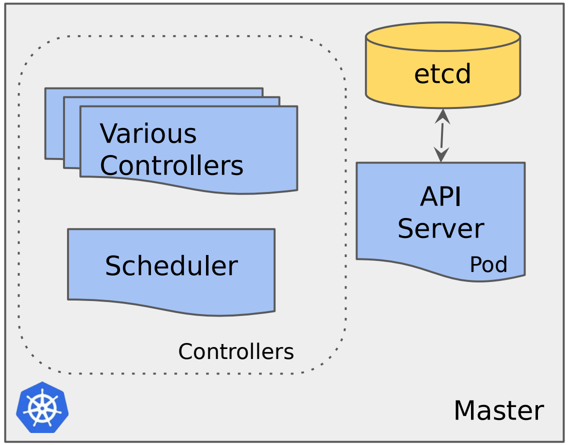
Kubernetes Scheduling

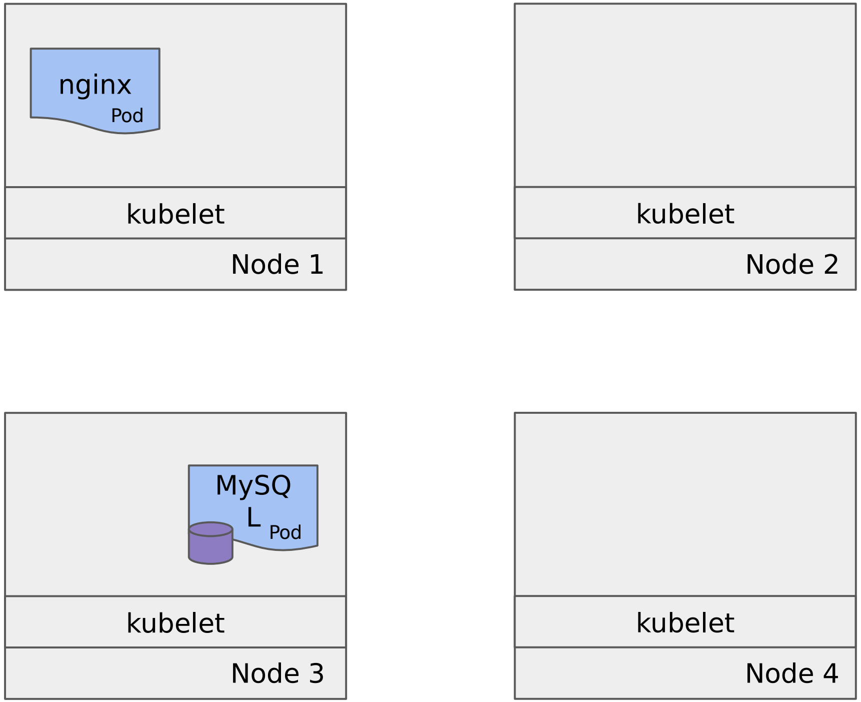


Kubernetes Scheduling



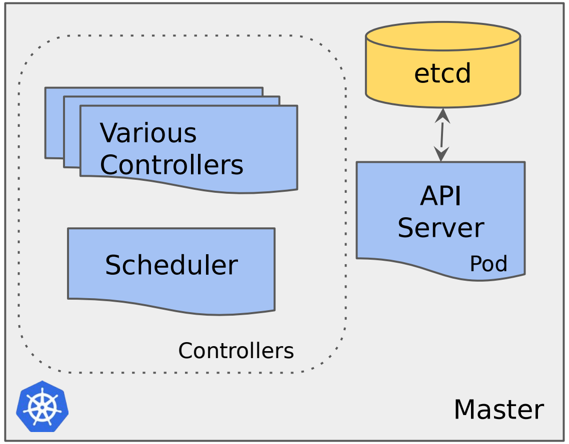
Kubernetes Scheduling

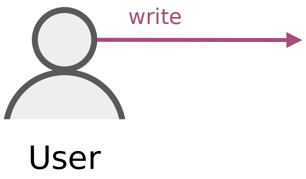





```yaml
kind: Pod
metadata:
  name: busybox
spec:
  image: busybox
  command: ['sh', '-c', 'sleep 1000']
  resources:
    requests:
      memory: "64Mi"
      cpu: "250m"
    limits:
      memory: "128Mi"
      cpu: "500m"
  nodeName:
```

Kubernetes Scheduling

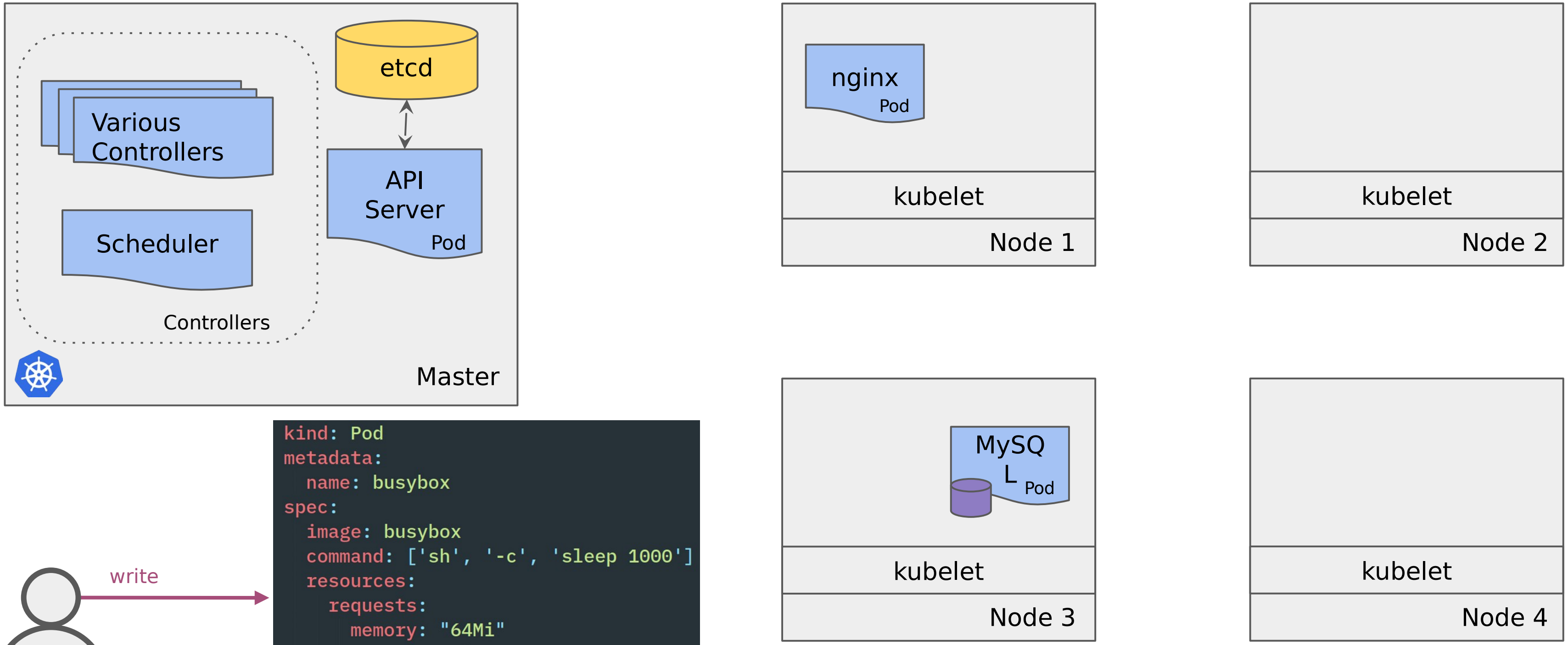





```yaml
kind: Pod
metadata:
  name: busybox
spec:
  image: busybox
  command: ['sh', '-c', 'sleep 1000']
  resources:
    requests:
      memory: "64Mi"
      cpu: "250m"
    limits:
      memory: "128Mi"
      cpu: "500m"
  nodeName:
```

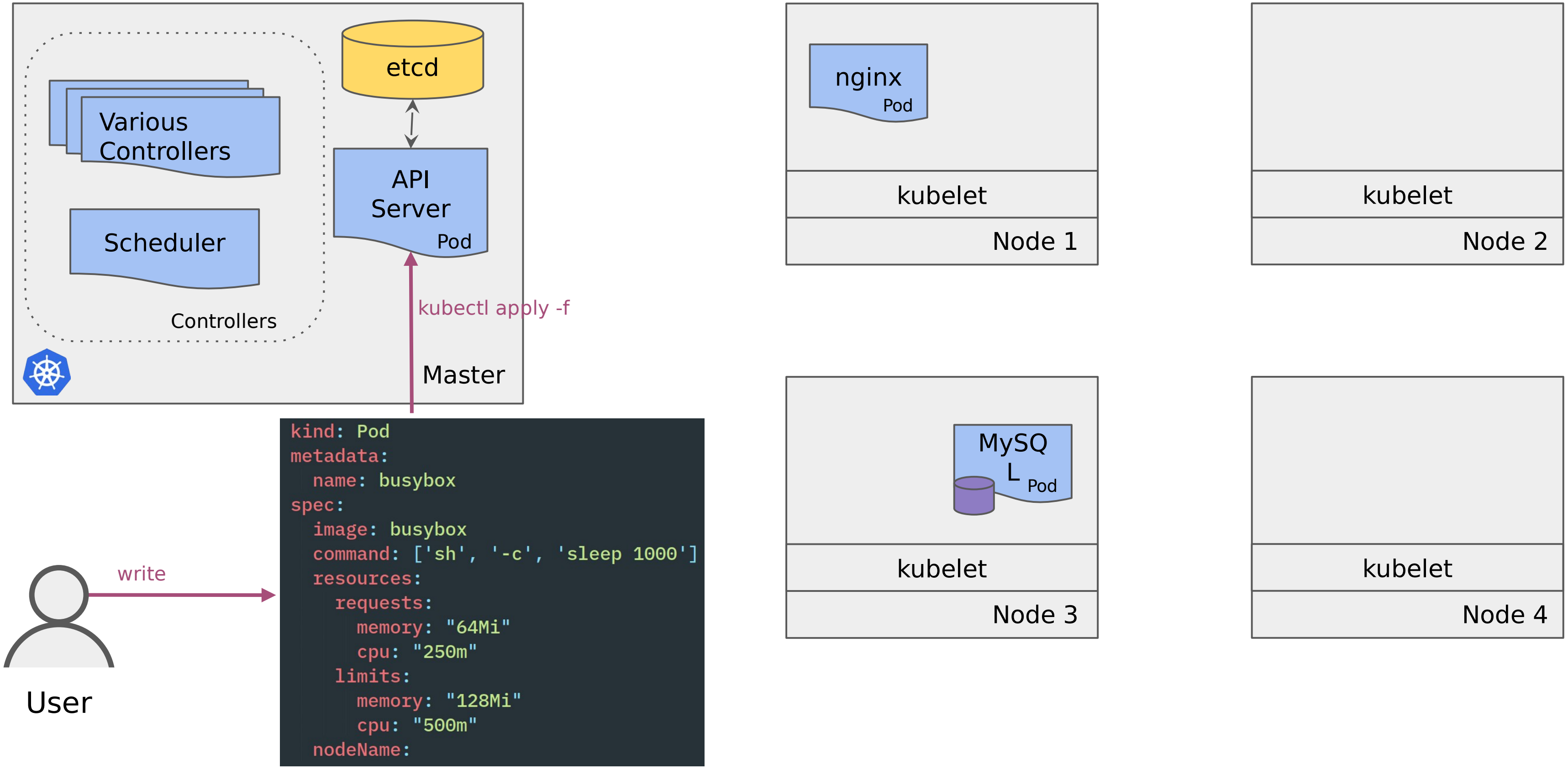

```yaml
kind: Pod
metadata:
  name: busybox
spec:
  image: busybox
  command: ['sh', '-c', 'sleep 1000']
  resources:
    requests:
      memory: "64Mi"
      cpu: "250m"
    limits:
      memory: "128Mi"
      cpu: "500m"
  nodeName:
```

Kubernetes Scheduling



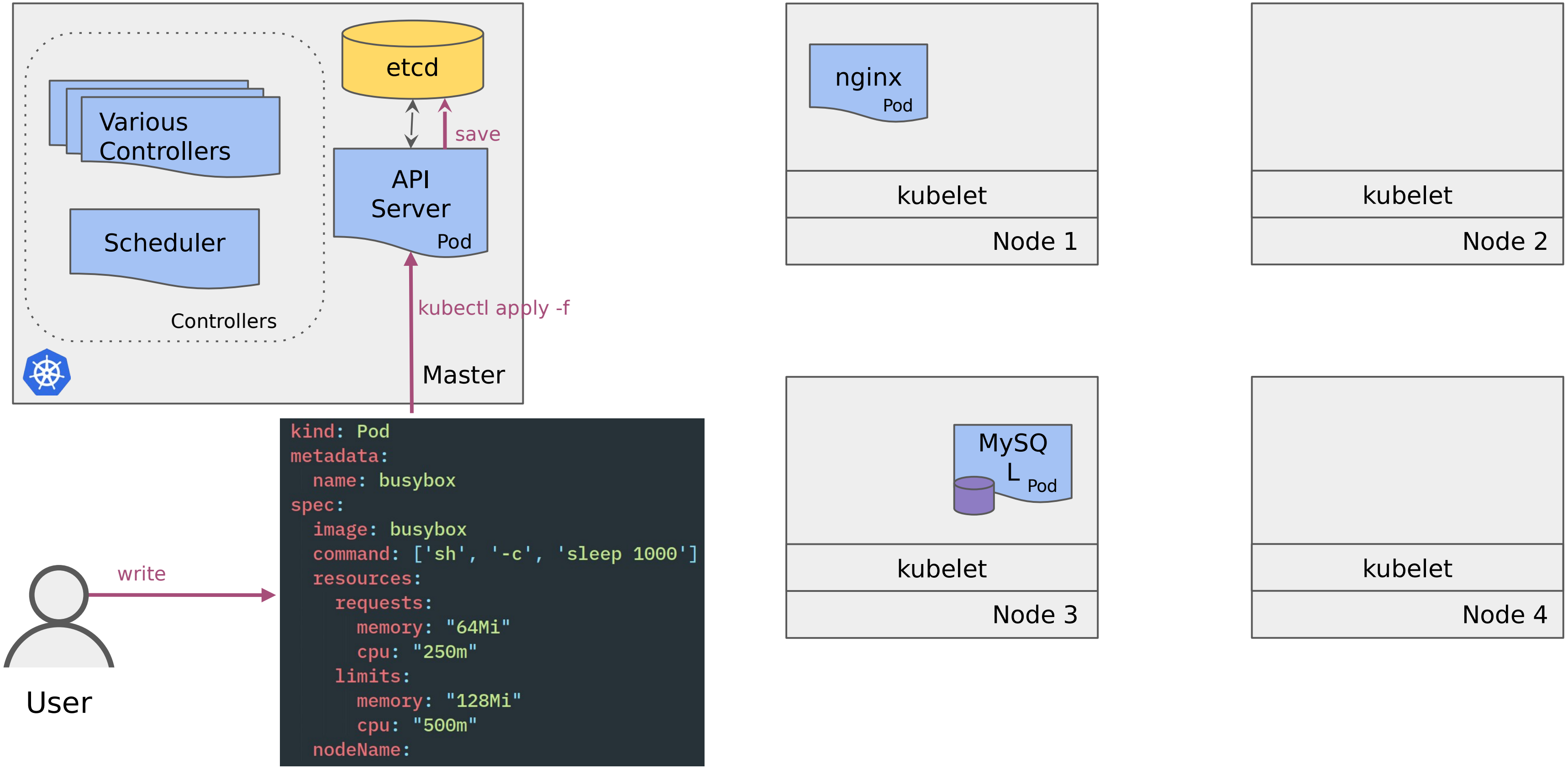

```yaml
kind: Pod
metadata:
  name: busybox
spec:
  image: busybox
  command: ['sh', '-c', 'sleep 1000']
  resources:
    requests:
      memory: "64Mi"
      cpu: "250m"
    limits:
      memory: "128Mi"
      cpu: "500m"
  nodeName:
```

Kubernetes Scheduling



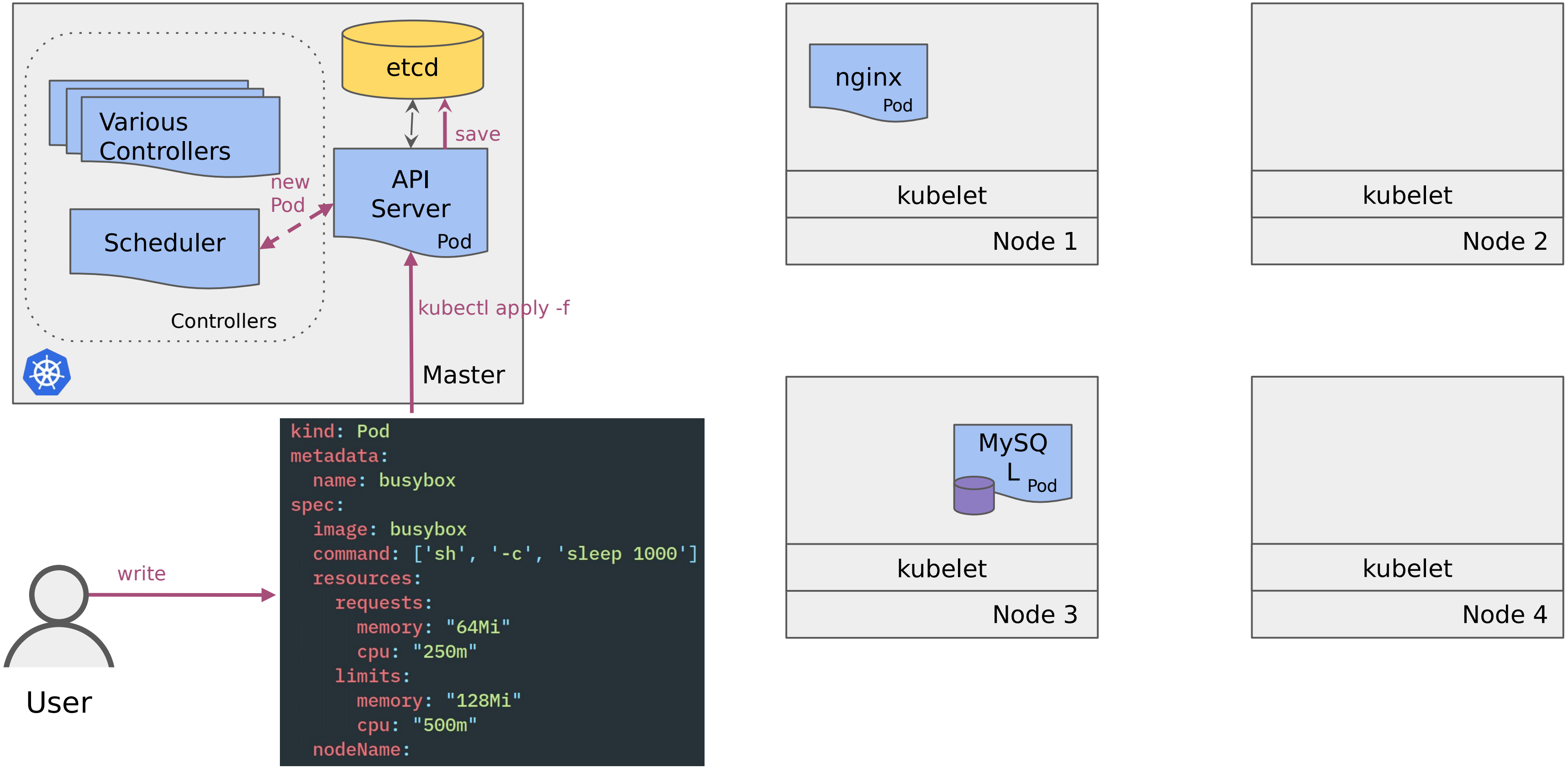

```yaml
kind: Pod
metadata:
  name: busybox
spec:
  image: busybox
  command: ['sh', '-c', 'sleep 1000']
  resources:
    requests:
      memory: "64Mi"
      cpu: "250m"
    limits:
      memory: "128Mi"
      cpu: "500m"
  nodeName:
```

Kubernetes Scheduling



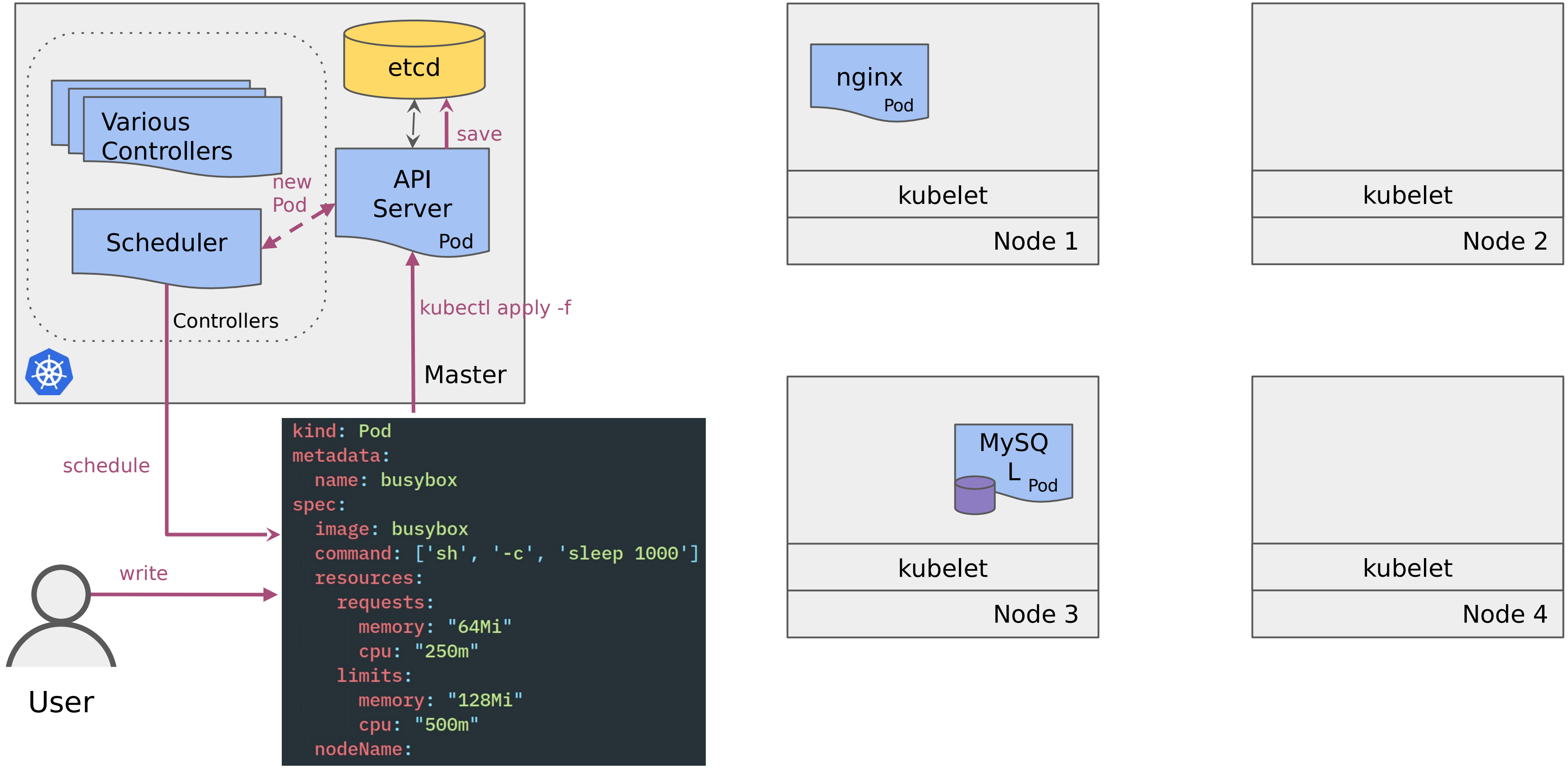

```yaml
kind: Pod
metadata:
  name: busybox
spec:
  image: busybox
  command: ['sh', '-c', 'sleep 1000']
  resources:
    requests:
      memory: "64Mi"
      cpu: "250m"
    limits:
      memory: "128Mi"
      cpu: "500m"
  nodeName:
```

Kubernetes Scheduling

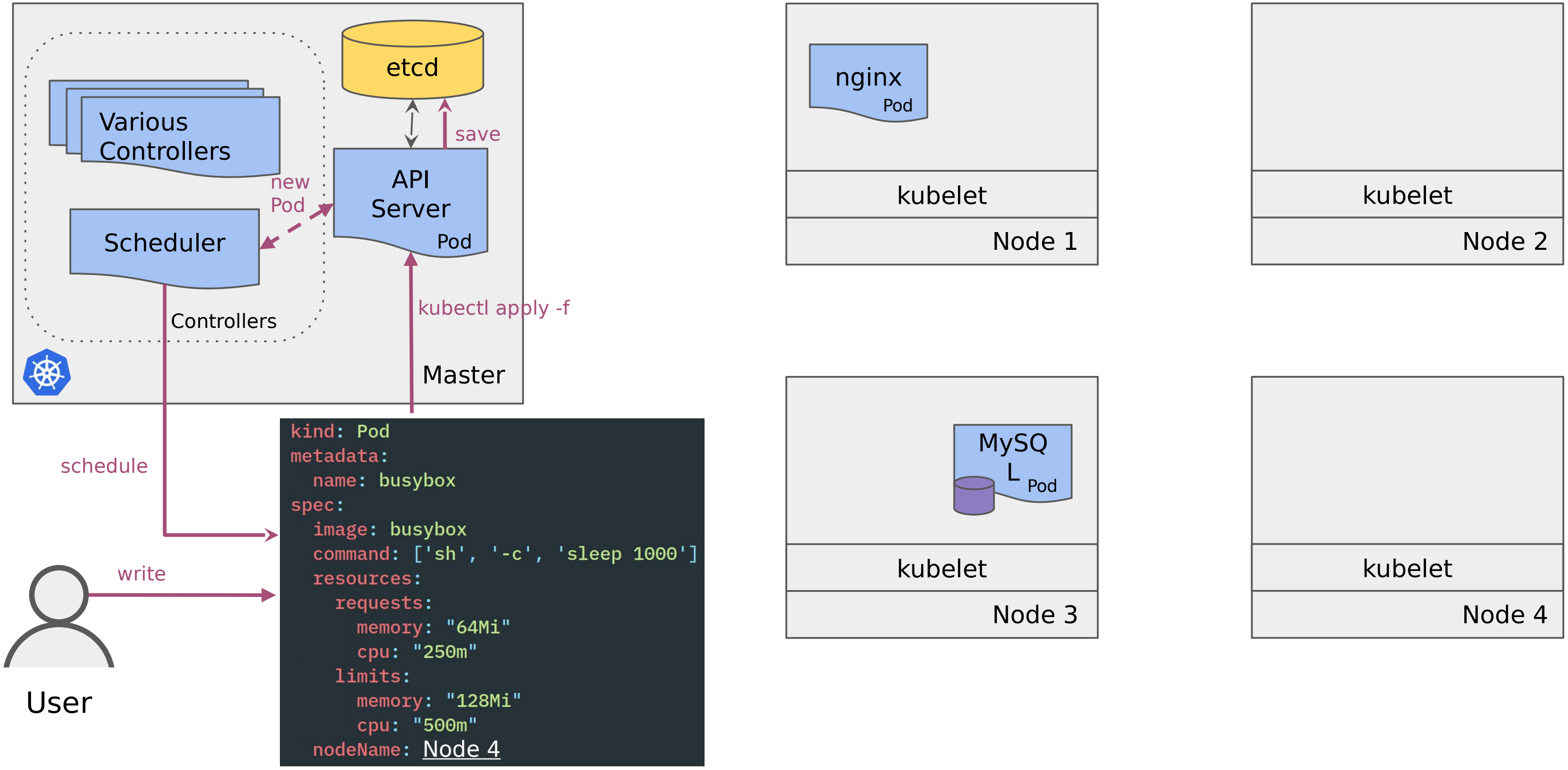



```yaml
kind: Pod
metadata:
  name: busybox
spec:
  image: busybox
  command: ['sh', '-c', 'sleep 1000']
  resources:
    requests:
      memory: "64Mi"
      cpu: "250m"
    limits:
      memory: "128Mi"
      cpu: "500m"
  nodeName:
```

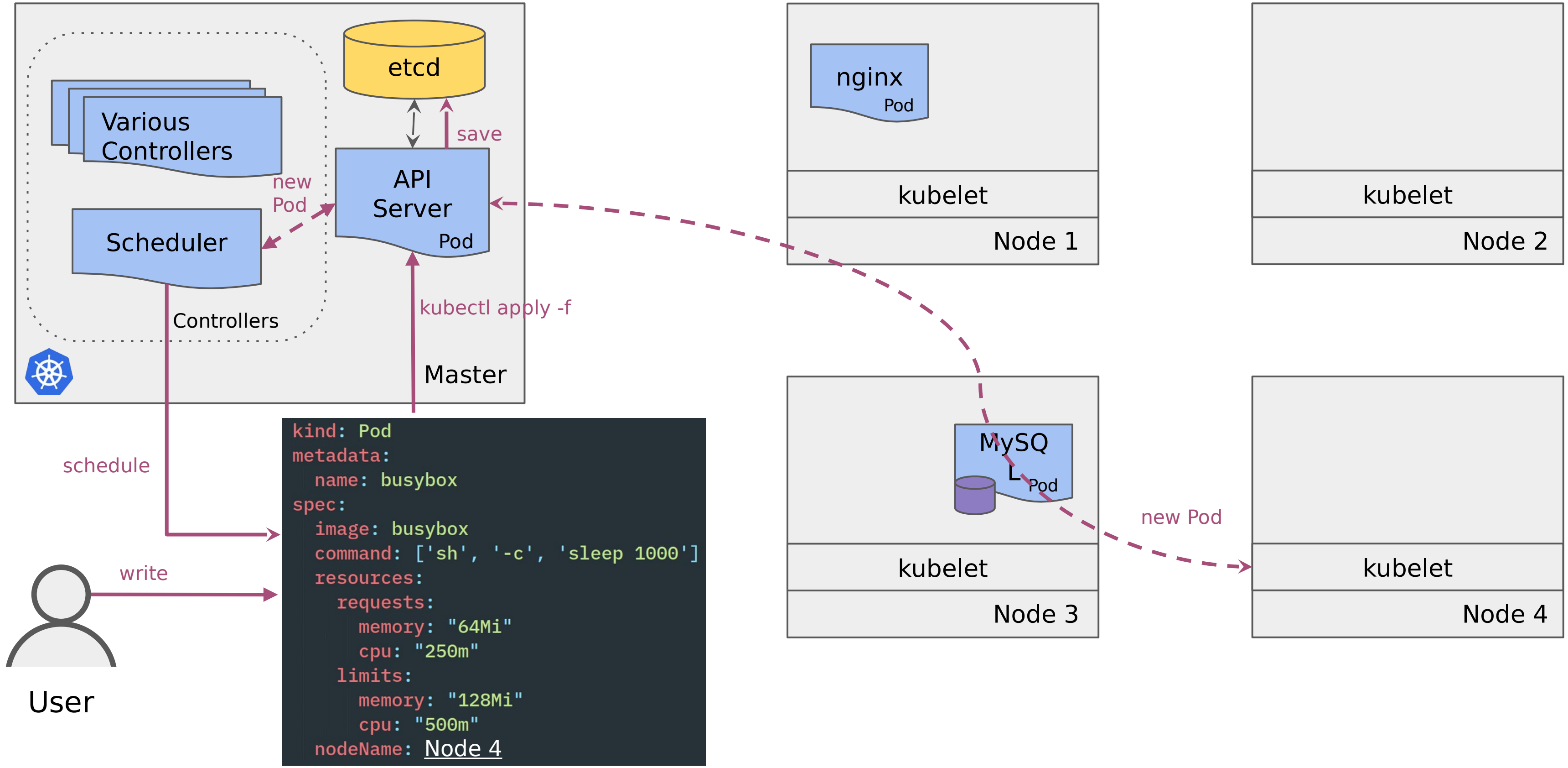
Kubernetes Scheduling



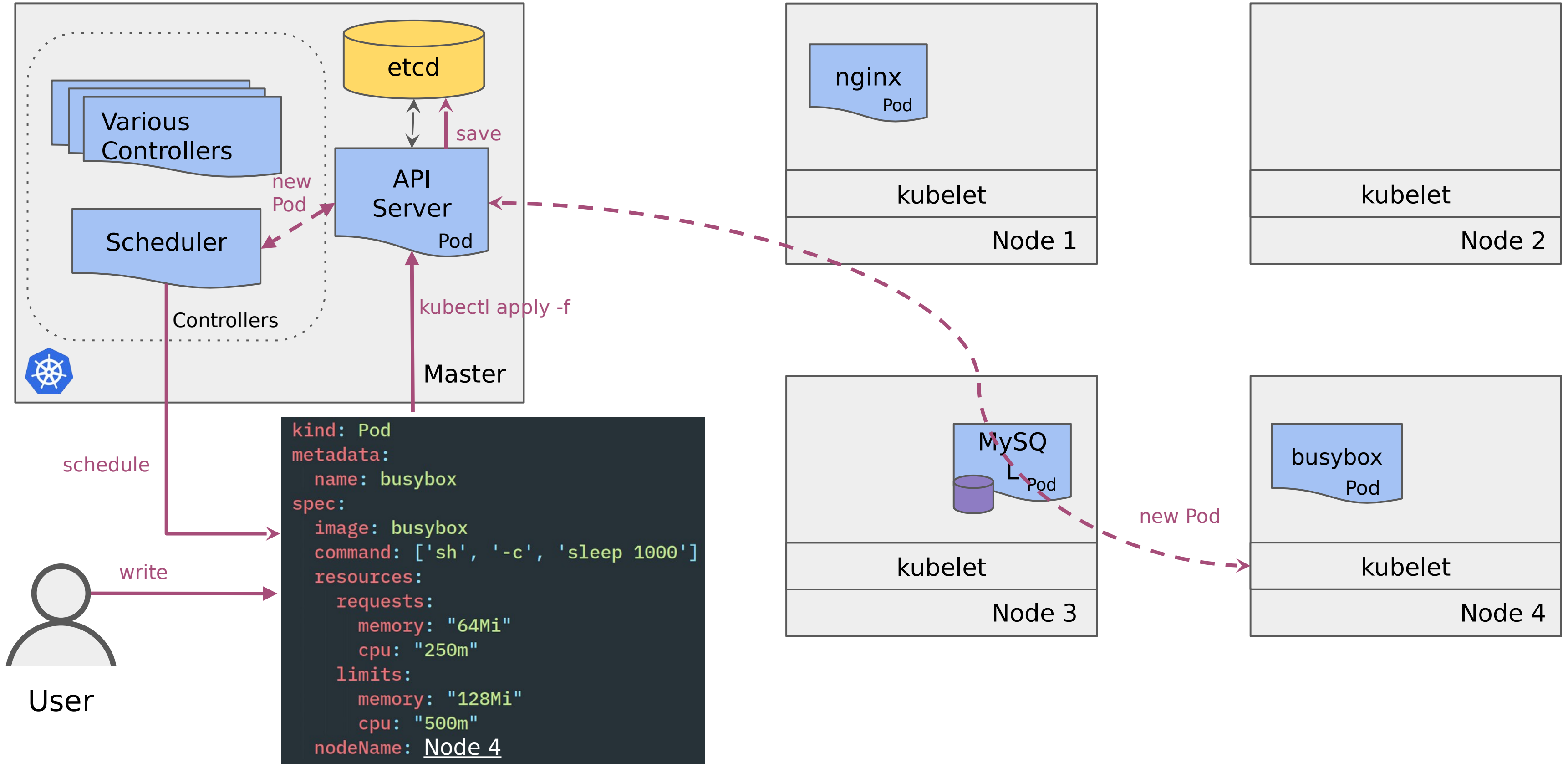
Kubernetes Scheduling



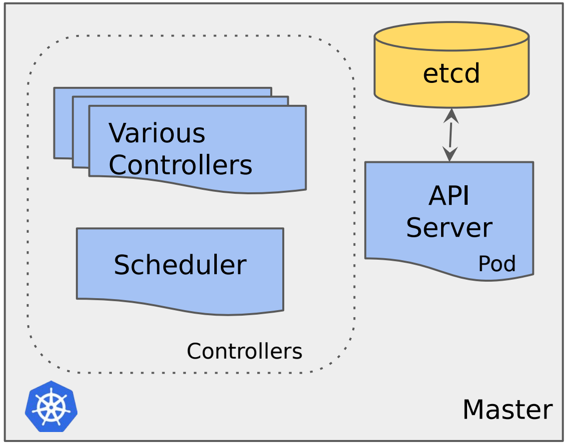
Kubernetes Scheduling

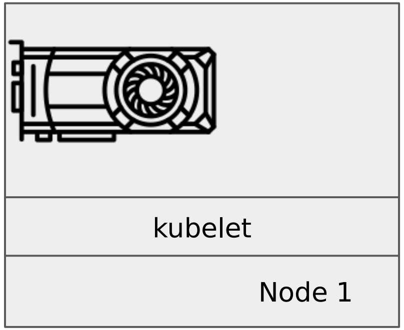


Kubernetes Scheduling

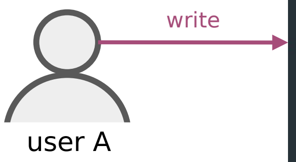


The Problem (once more)

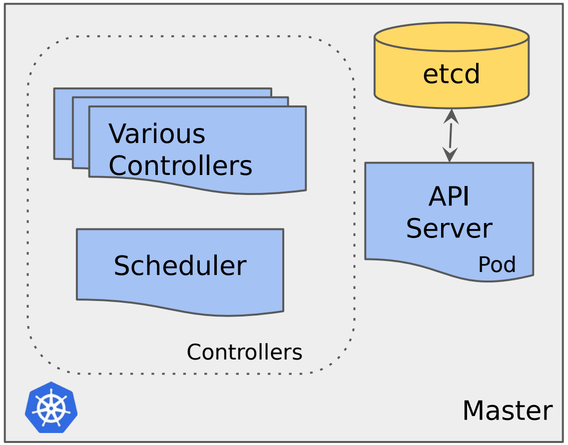


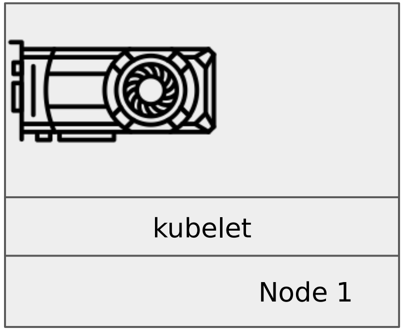


The Problem (once more)

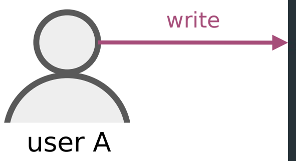



```yaml
kind: Pod
metadata:
  name: jupyter-1
spec:
  image: jupyter:latest
  resources:
    limits:
      nvidia.com/gpu: 1
  nodeName:
```

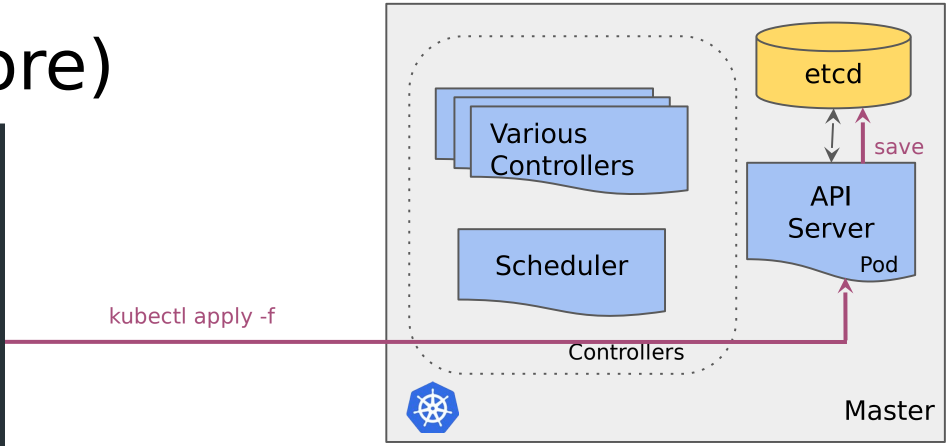


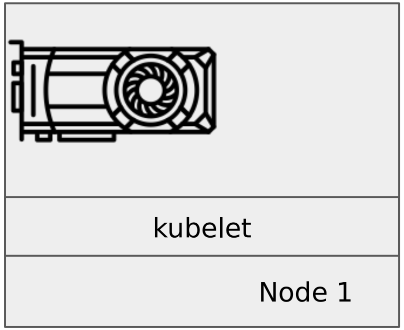


The Problem (once more)



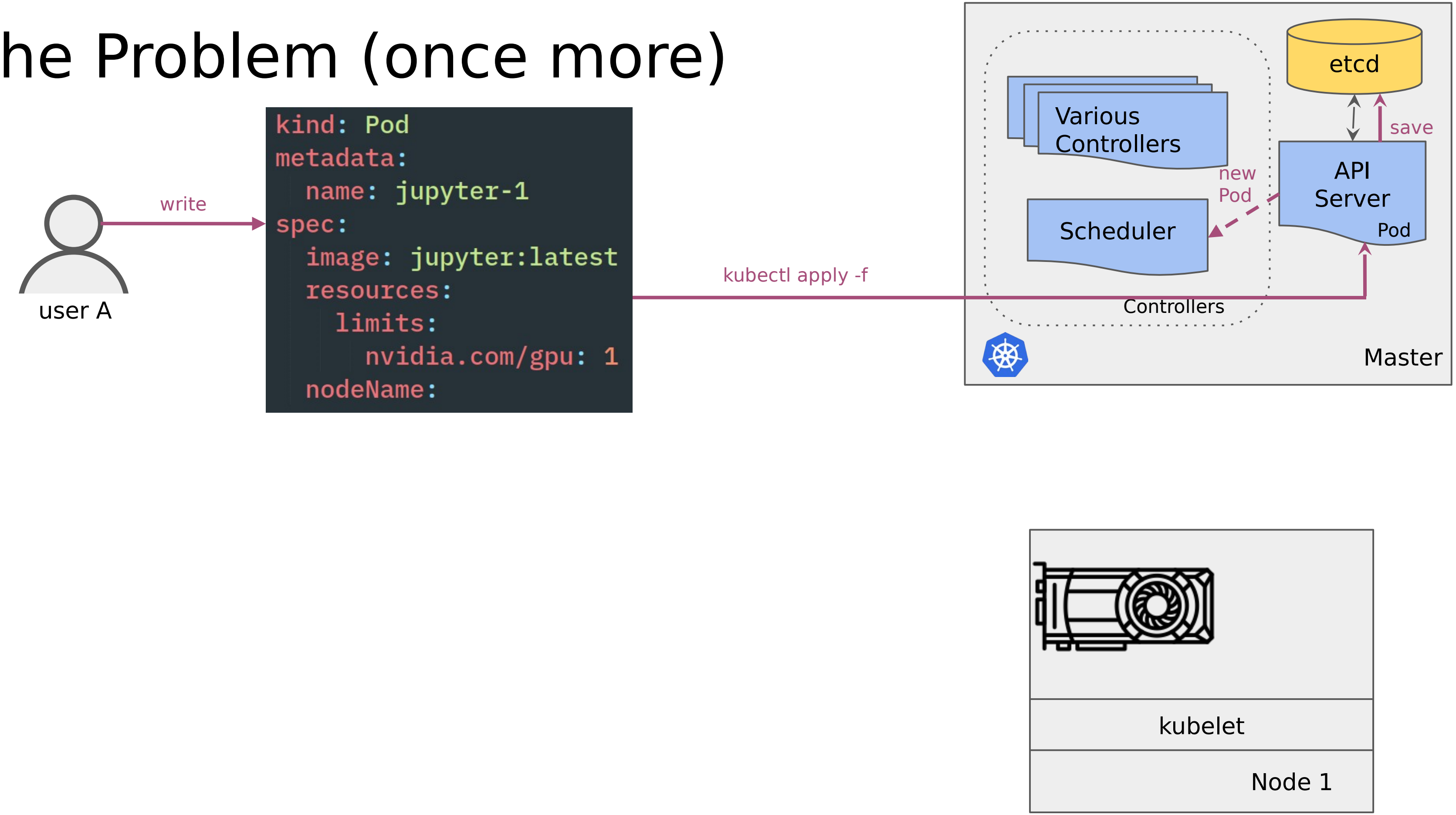

```yaml
kind: Pod
metadata:
  name: jupyter-1
spec:
  image: jupyter:latest
  resources:
    limits:
      nvidia.com/gpu: 1
  nodeName:
```





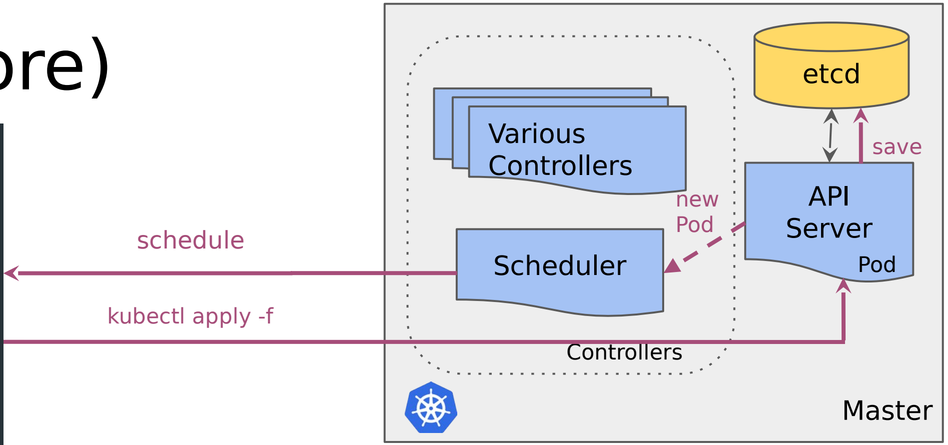
The Problem (once more)

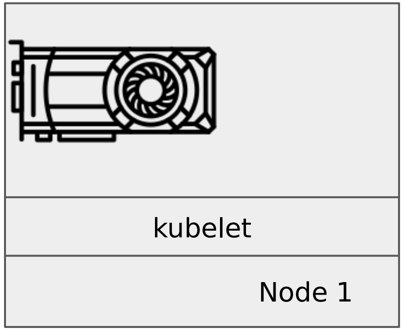
```yaml
kind: Pod
metadata:
  name: jupyter-1
spec:
  image: jupyter:latest
  resources:
    limits:
      nvidia.com/gpu: 1
  nodeName:
```



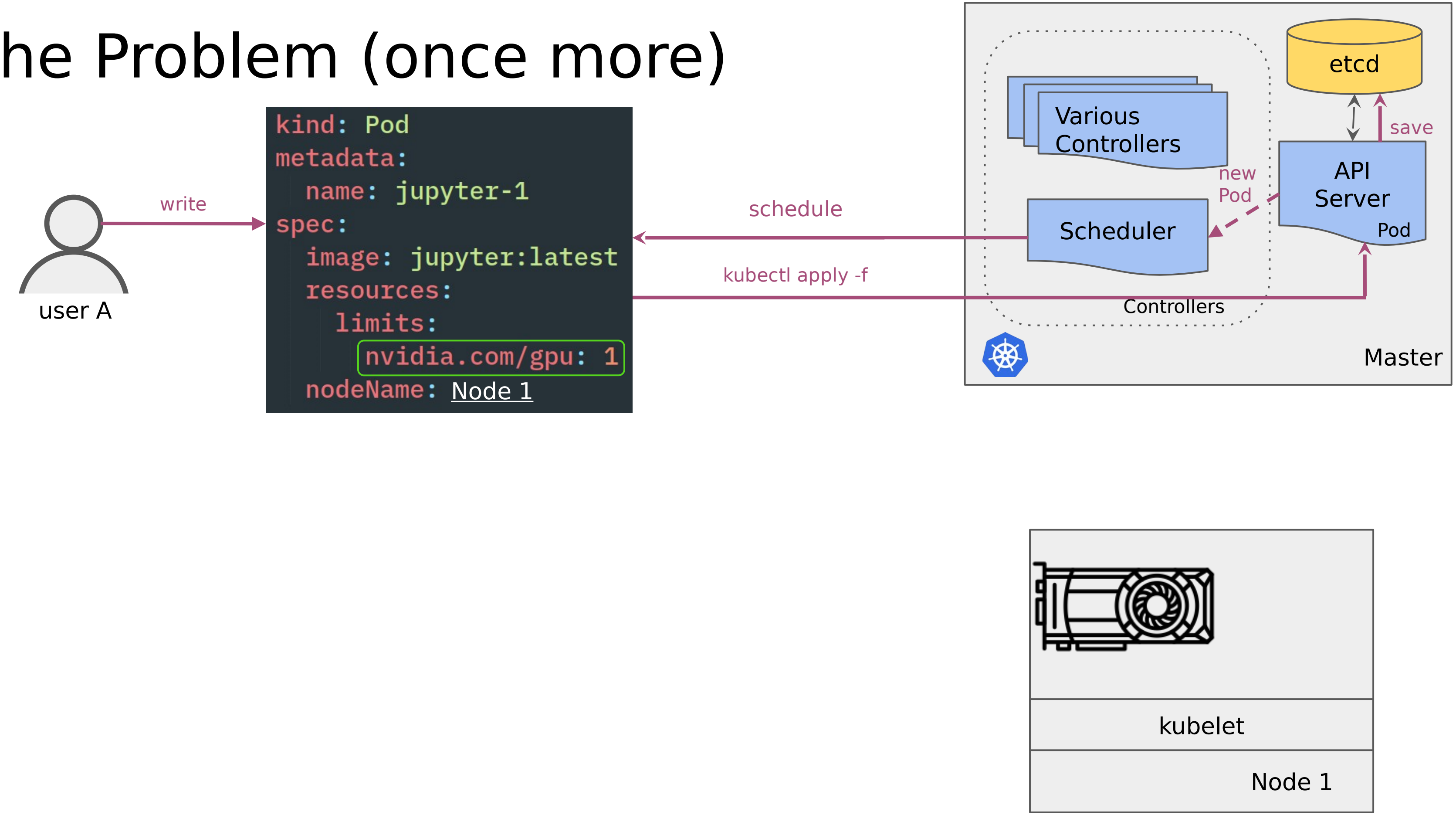
The Problem (once more)


```yaml
kind: Pod
metadata:
  name: jupyter-1
spec:
  image: jupyter:latest
  resources:
    limits:
      nvidia.com/gpu: 1
  nodeName:
```

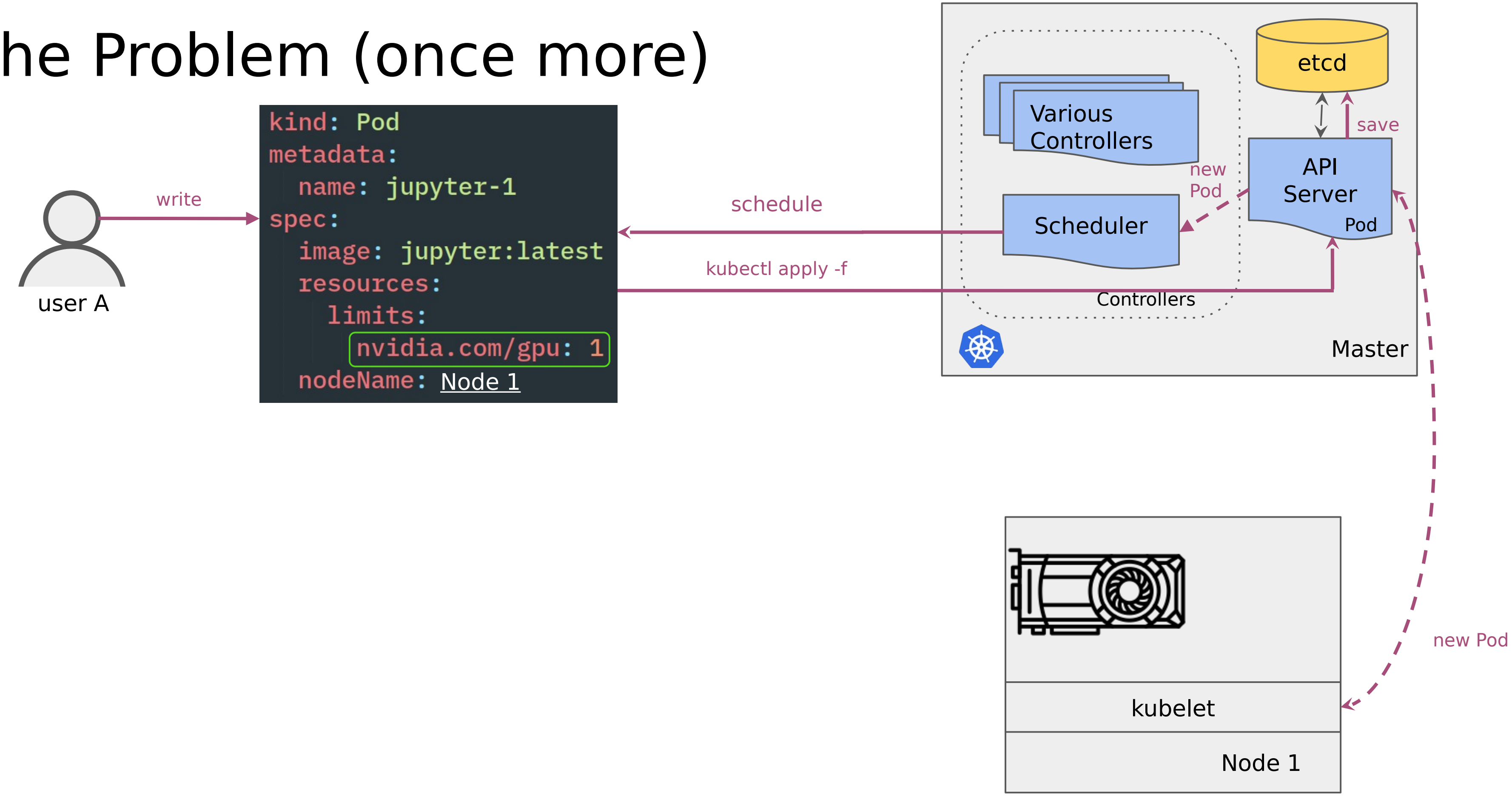




The Problem (once more)

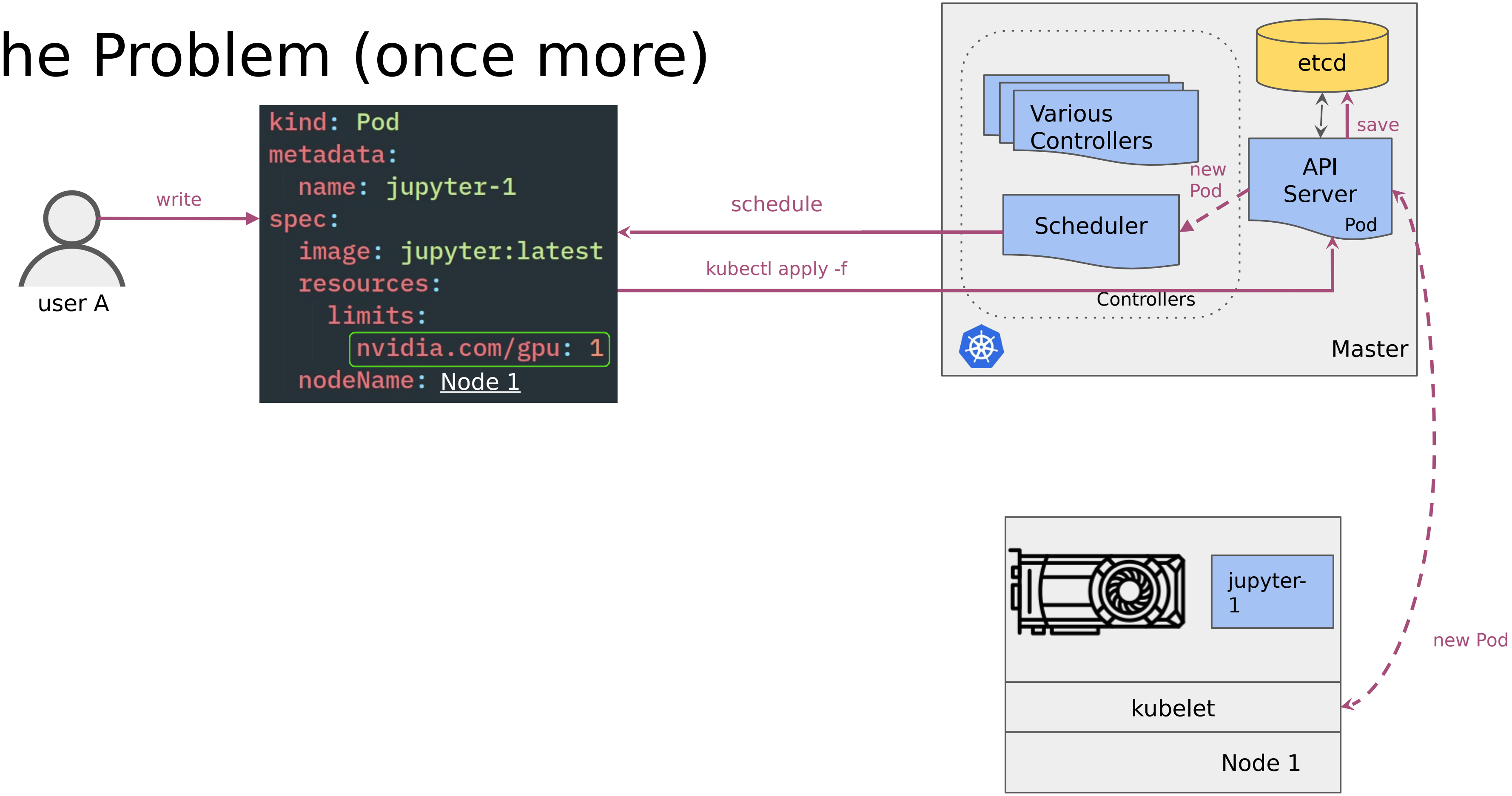


The Problem (once more)

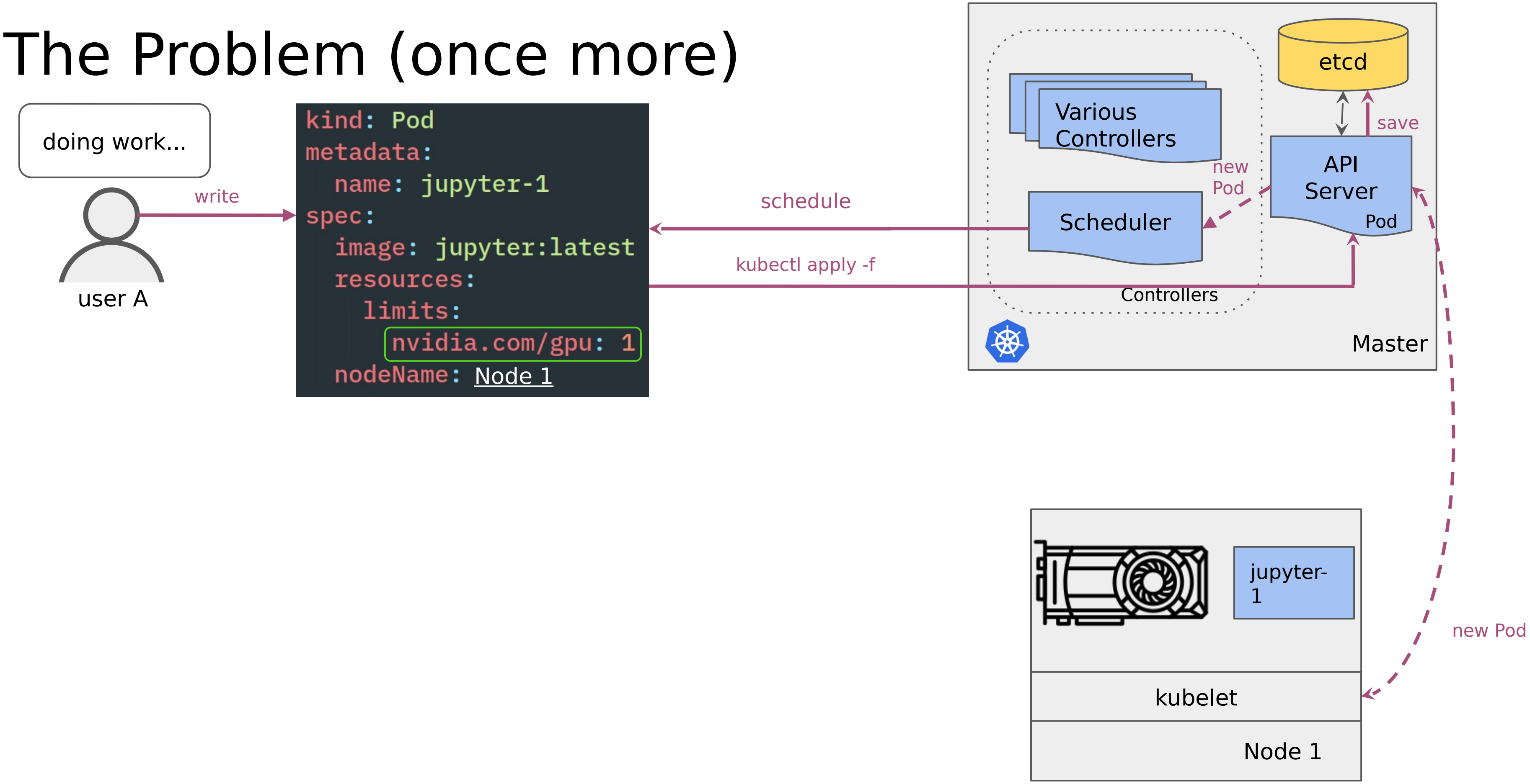


The Problem (once more)

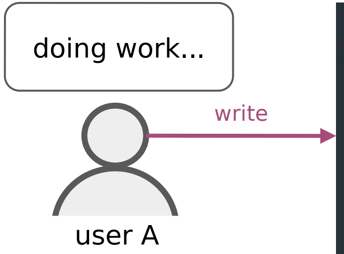
```yaml
kind: Pod
metadata:
  name: jupyter-1
spec:
  image: jupyter:latest
  resources:
    limits:
      nvidia.com/gpu: 1
  nodeName: Node 1
```



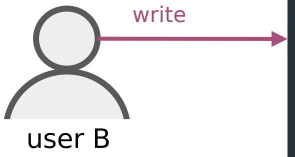
The Problem (once more)



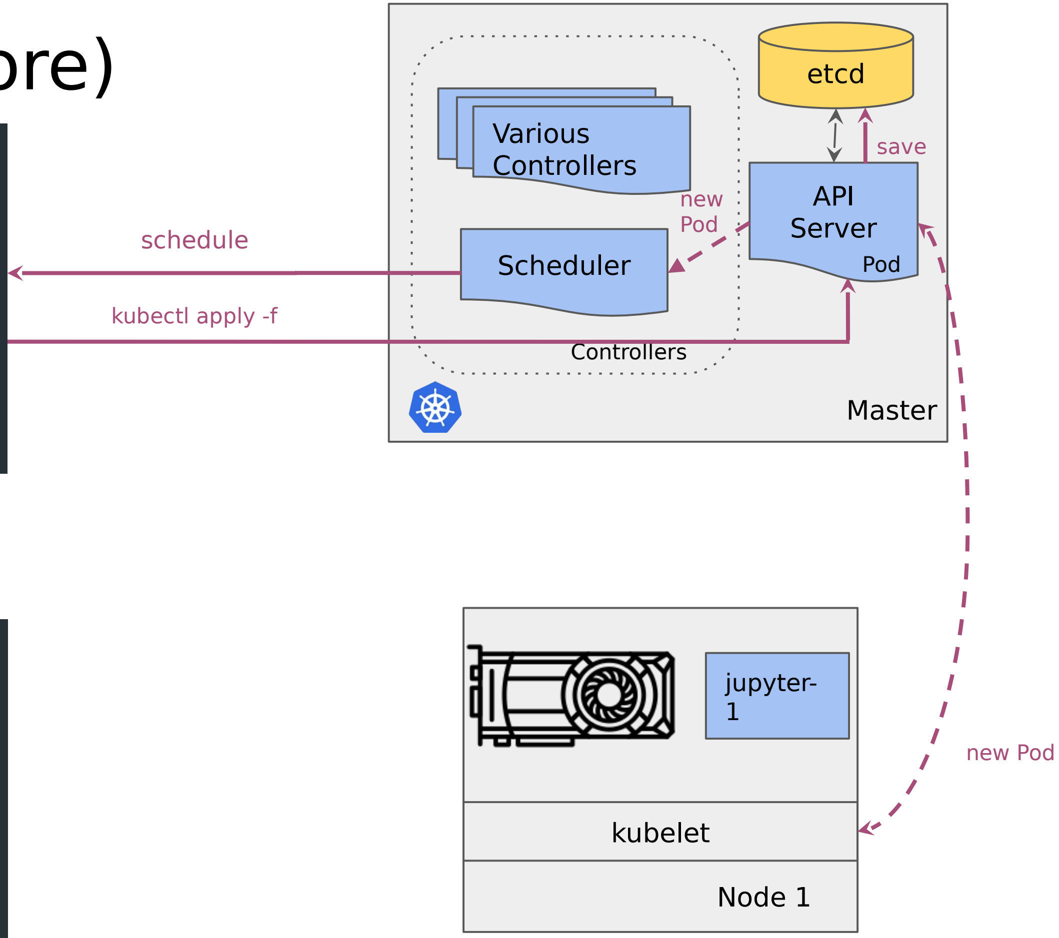
The Problem (once more)



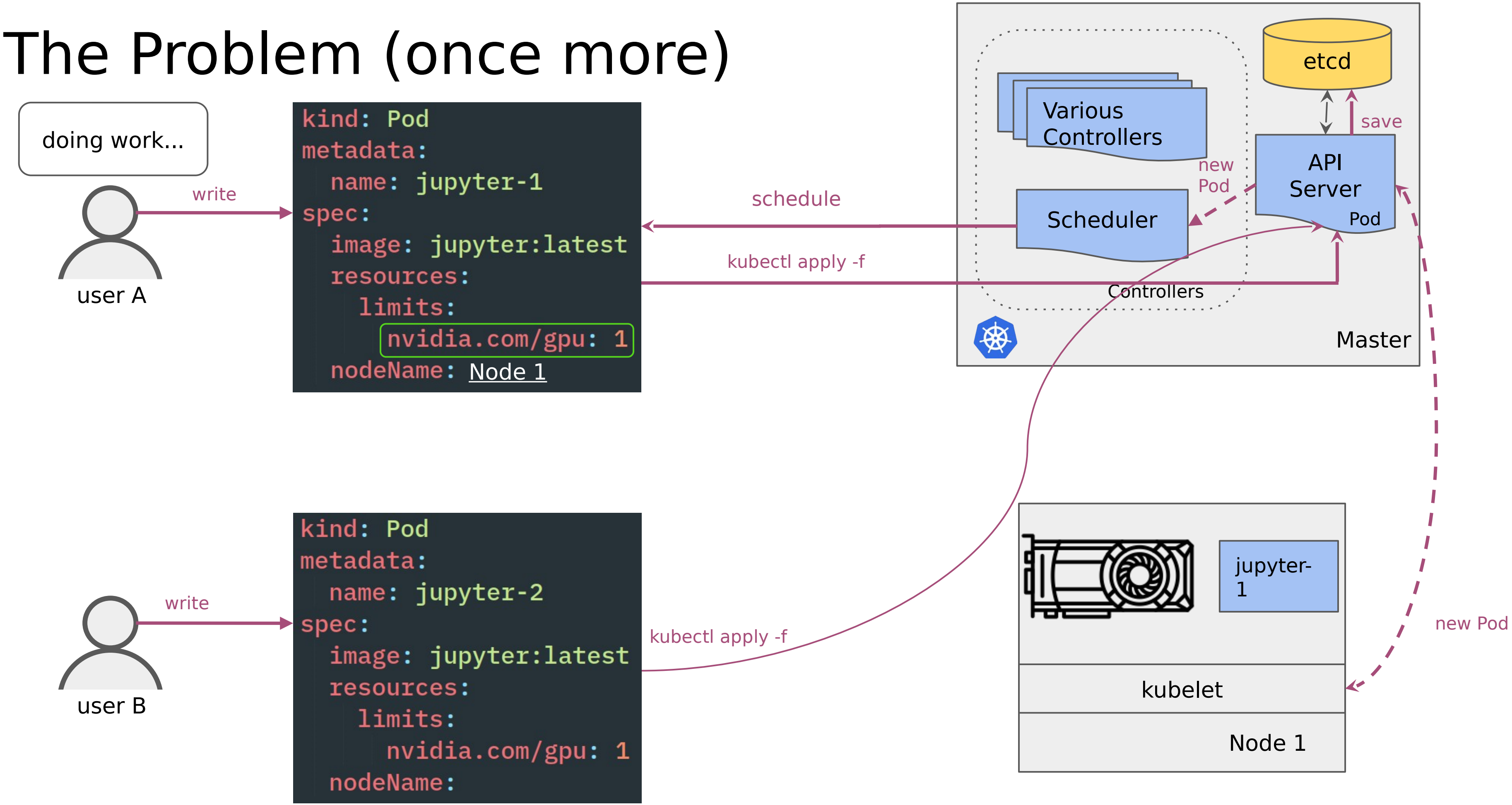

```yaml
kind: Pod
metadata:
  name: jupyter-1
spec:
  image: jupyter:latest
  resources:
    limits:
      nvidia.com/gpu: 1
  nodeName: Node 1
```



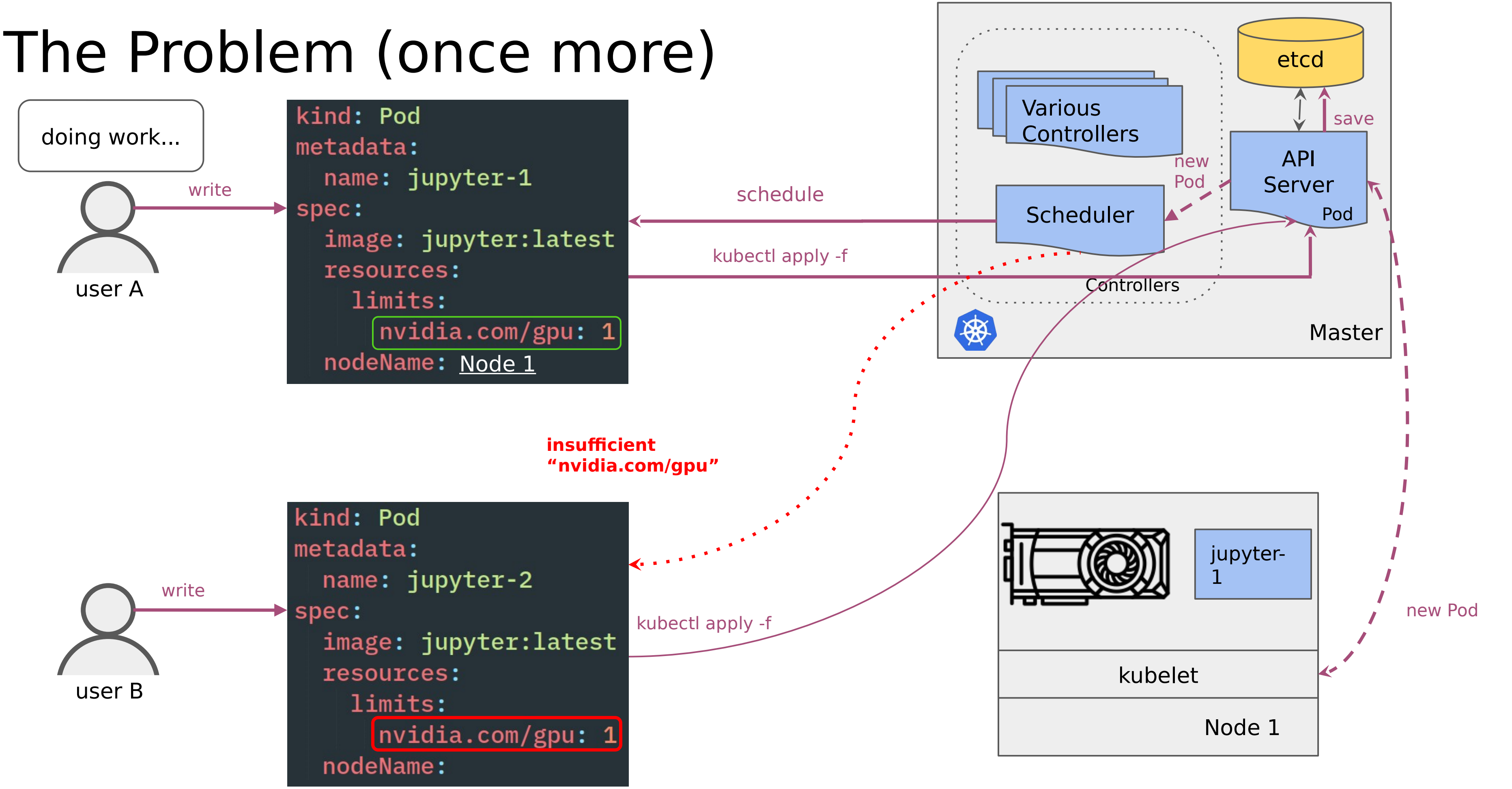

```yaml
kind: Pod
metadata:
  name: jupyter-2
spec:
  image: jupyter:latest
  resources:
    limits:
      nvidia.com/gpu: 1
  nodeName:
```



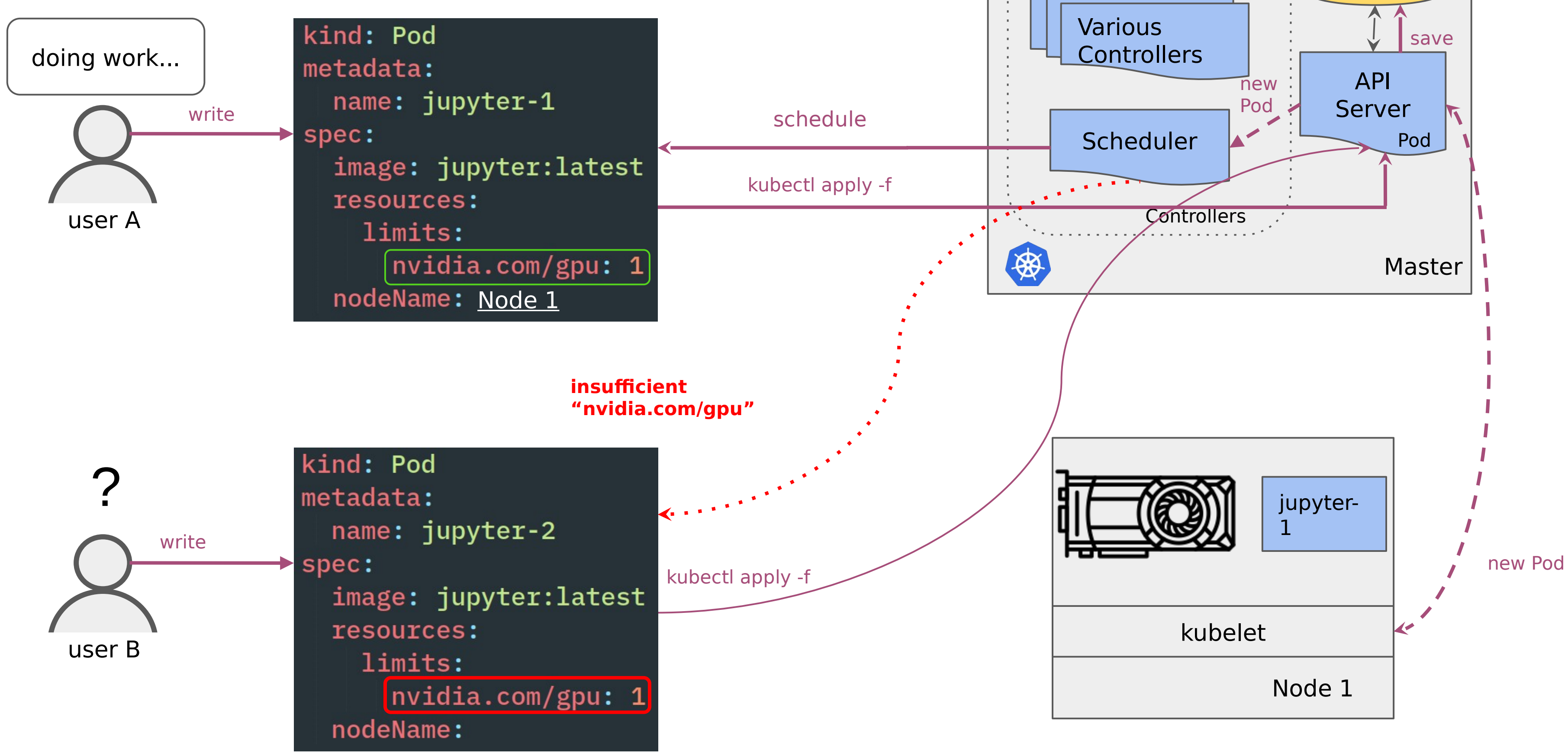
The Problem (once more)



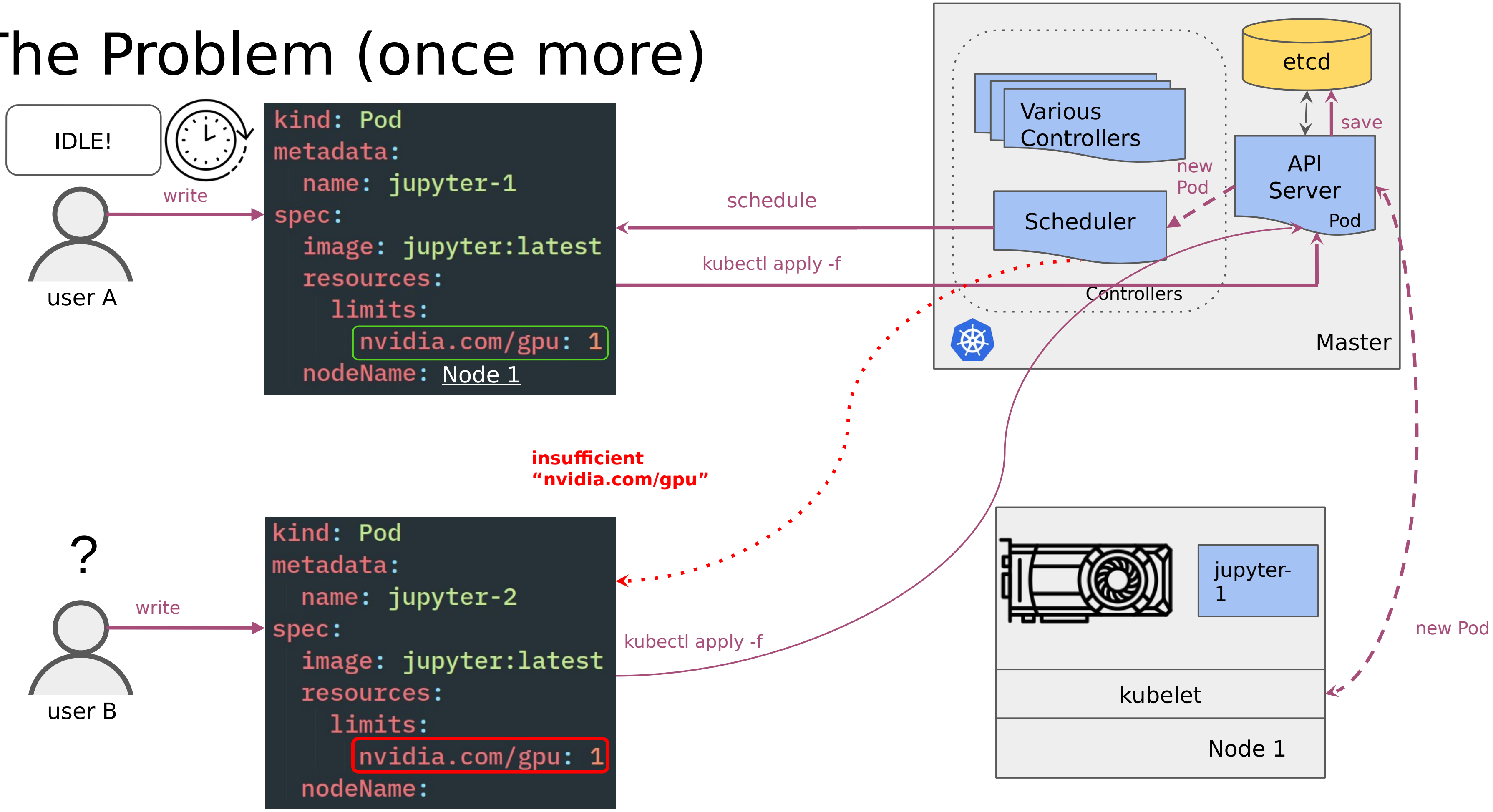
The Problem (once more)



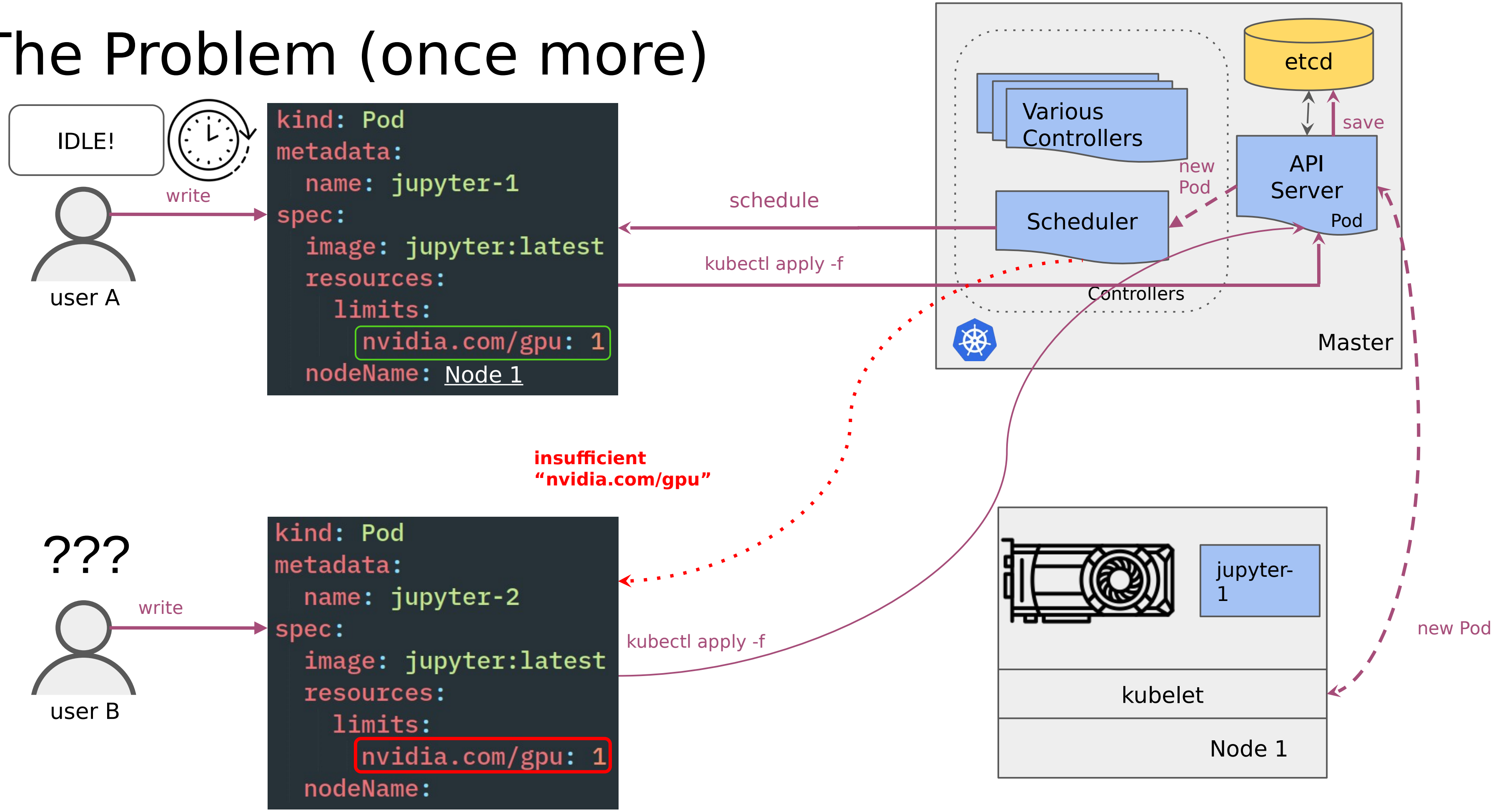
The Problem (once more)



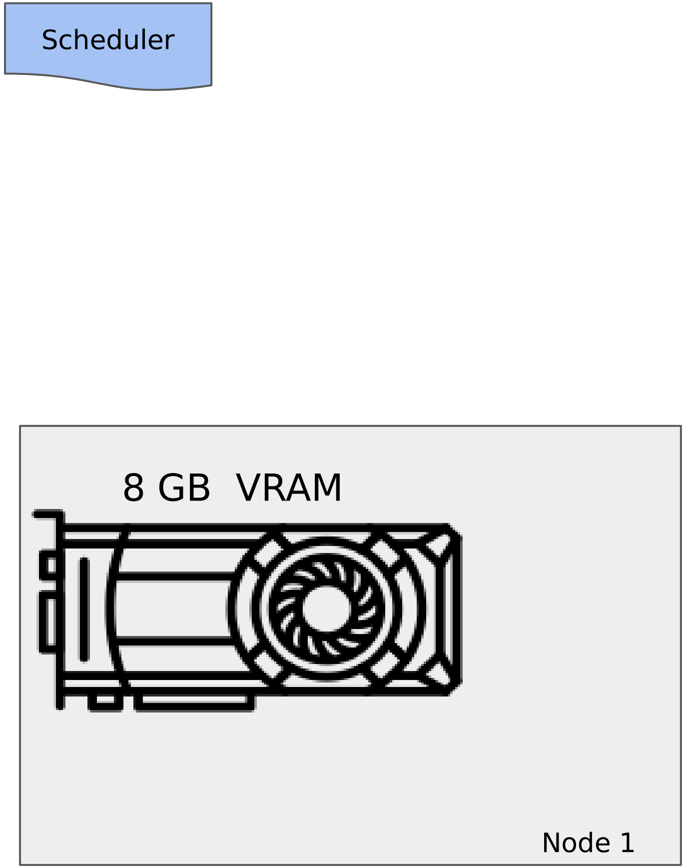
The Problem (once more)



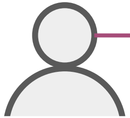
The Problem (once more)



Existing Approaches



Existing Approaches



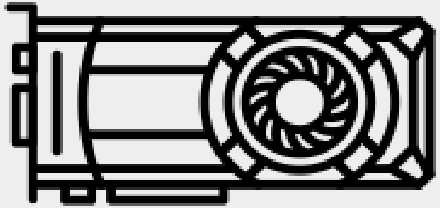
write

user
A

```yaml
kind: Pod
metadata:
  name: jupyter-1
spec:
  image: jupyter:latest
  resources:
    limits:
      aliyun.com/gpu-mem: 4
  nodeName:
```

Scheduler

8 GB VRAM



Node 1

Existing Approaches

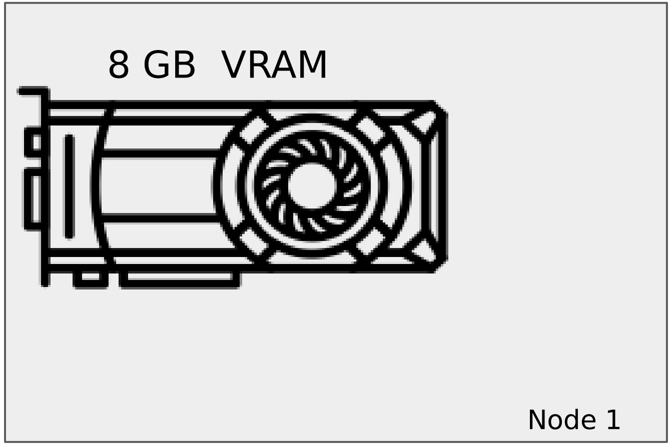
user A

write

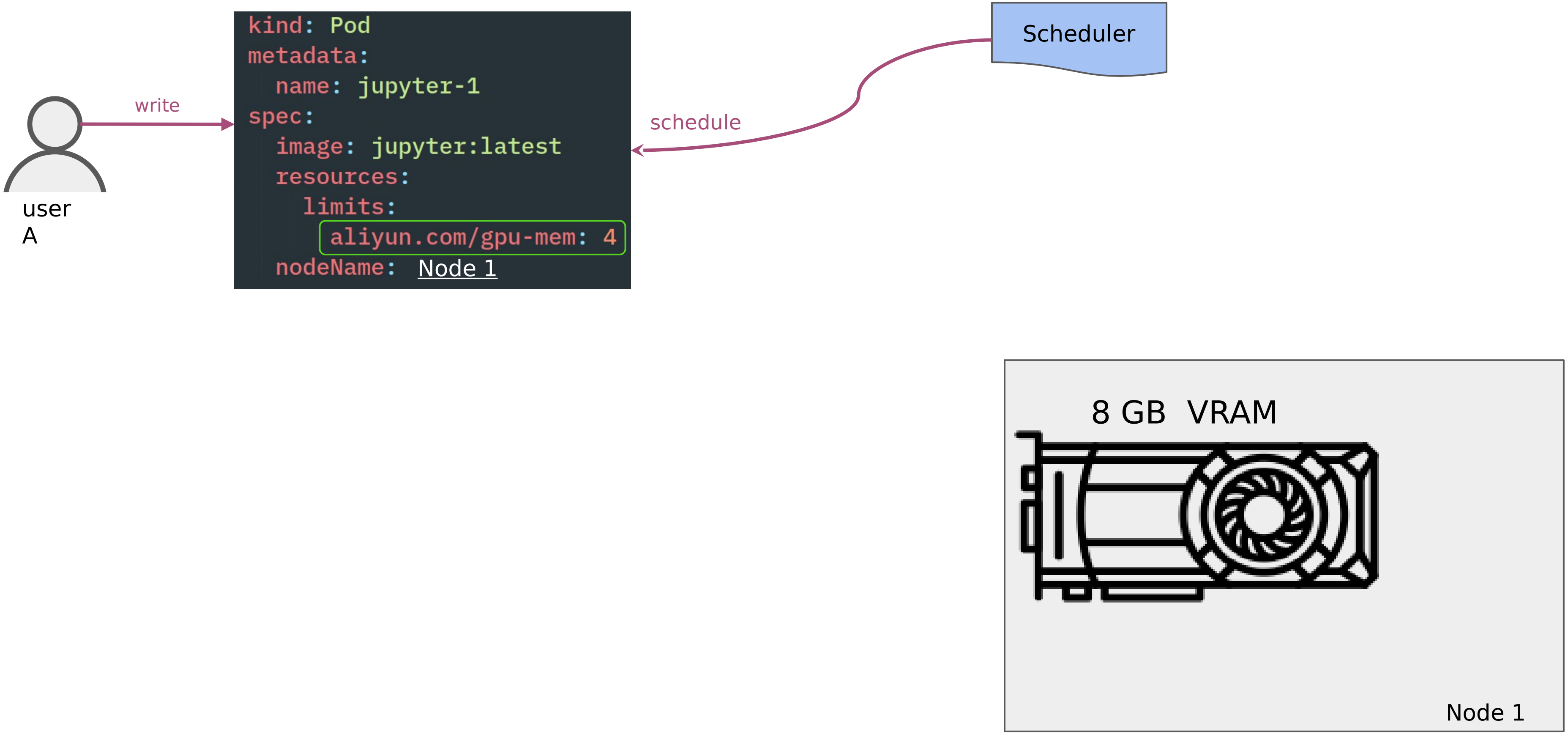
```yaml
kind: Pod
metadata:
  name: jupyter-1
spec:
  image: jupyter:latest
  resources:
    limits:
      aliyun.com/gpu-mem: 4
  nodeName:
```

schedule

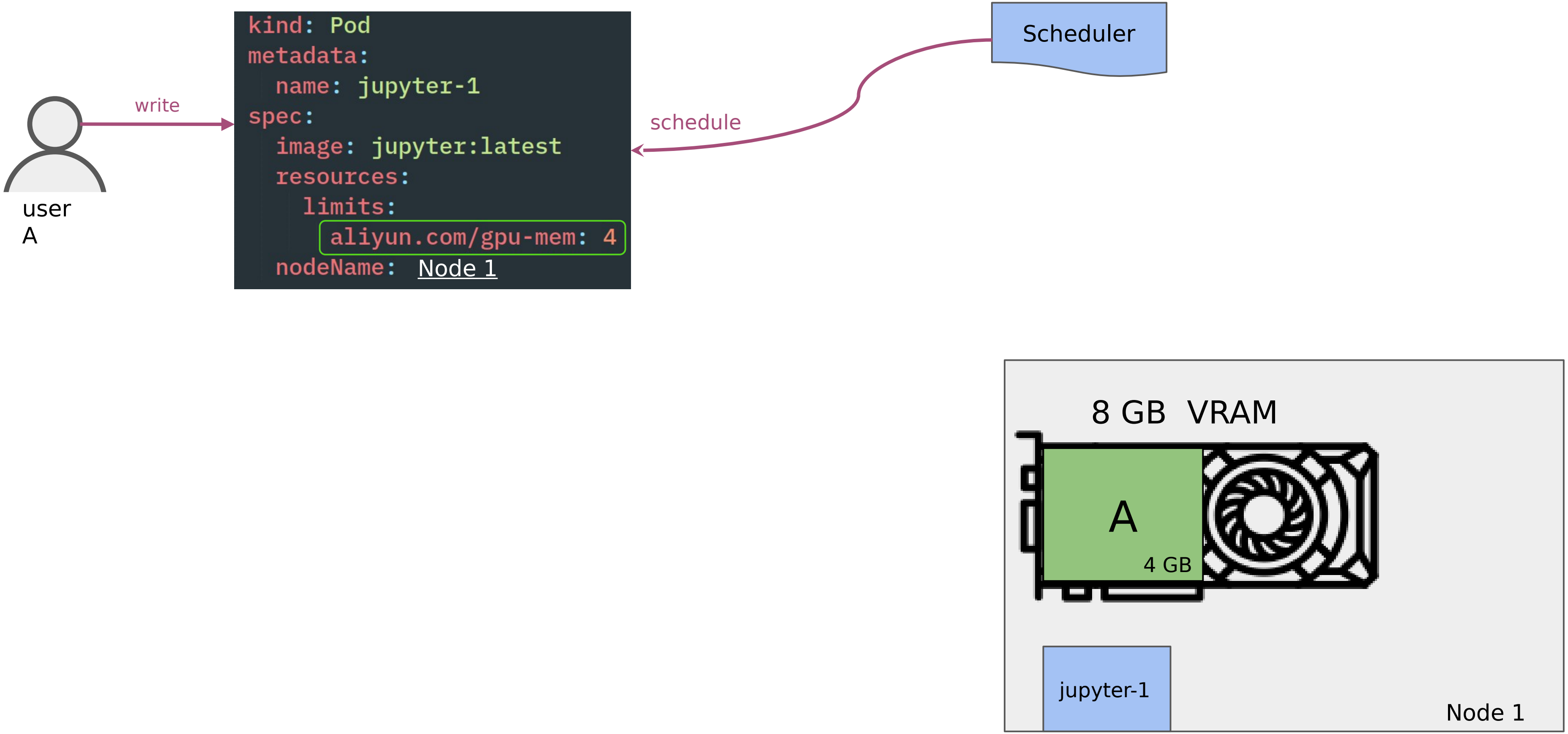
Scheduler



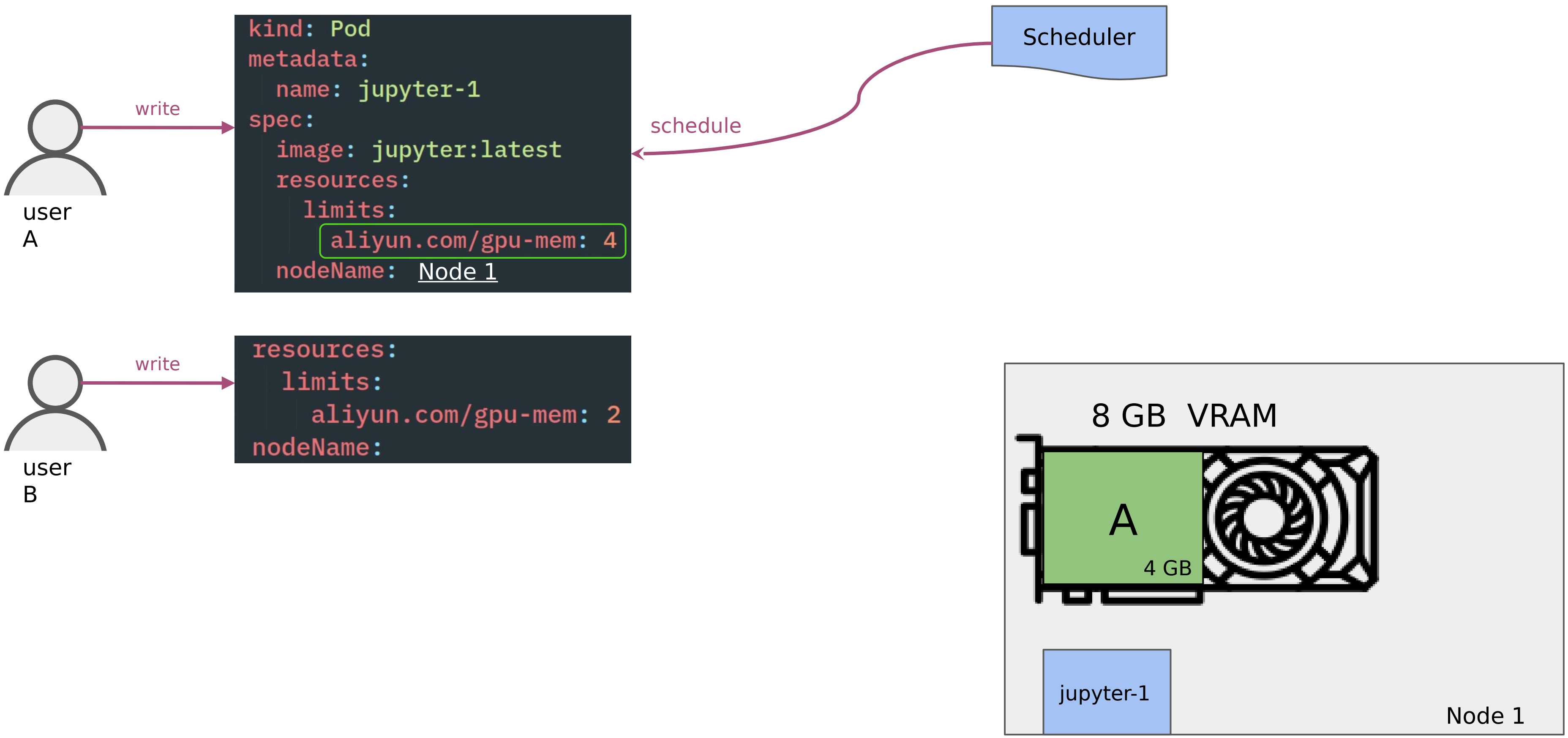
Existing Approaches



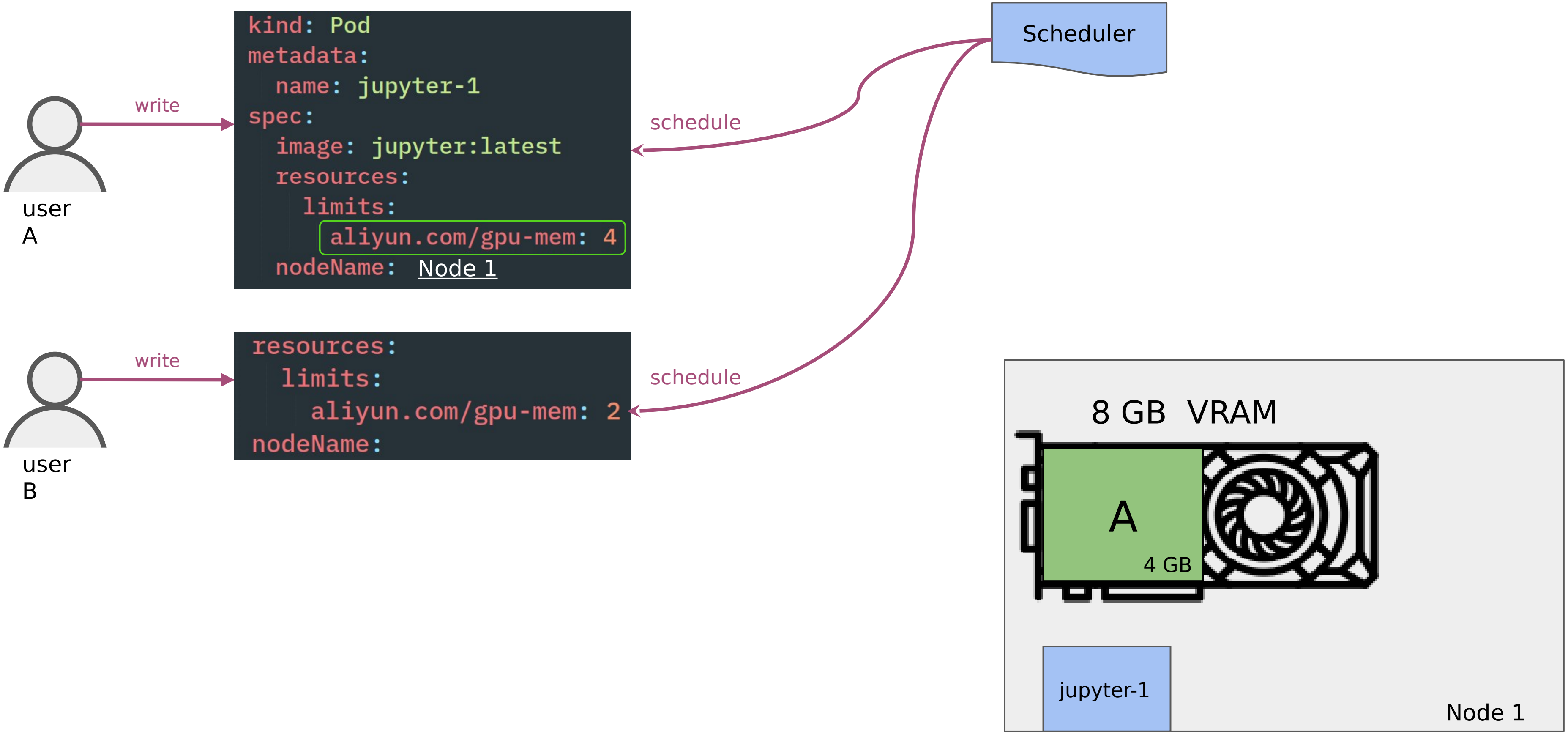
Existing Approaches



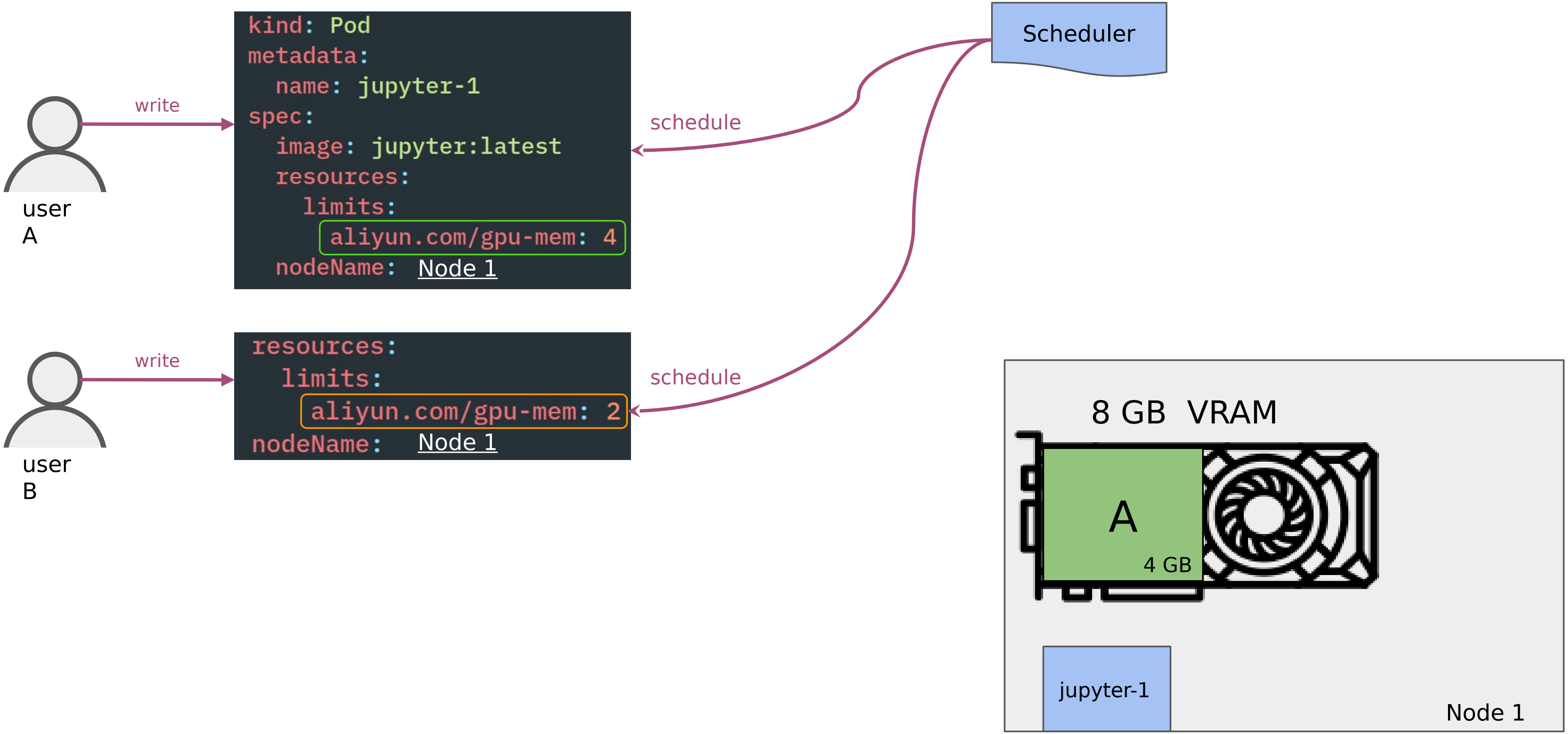
Existing Approaches



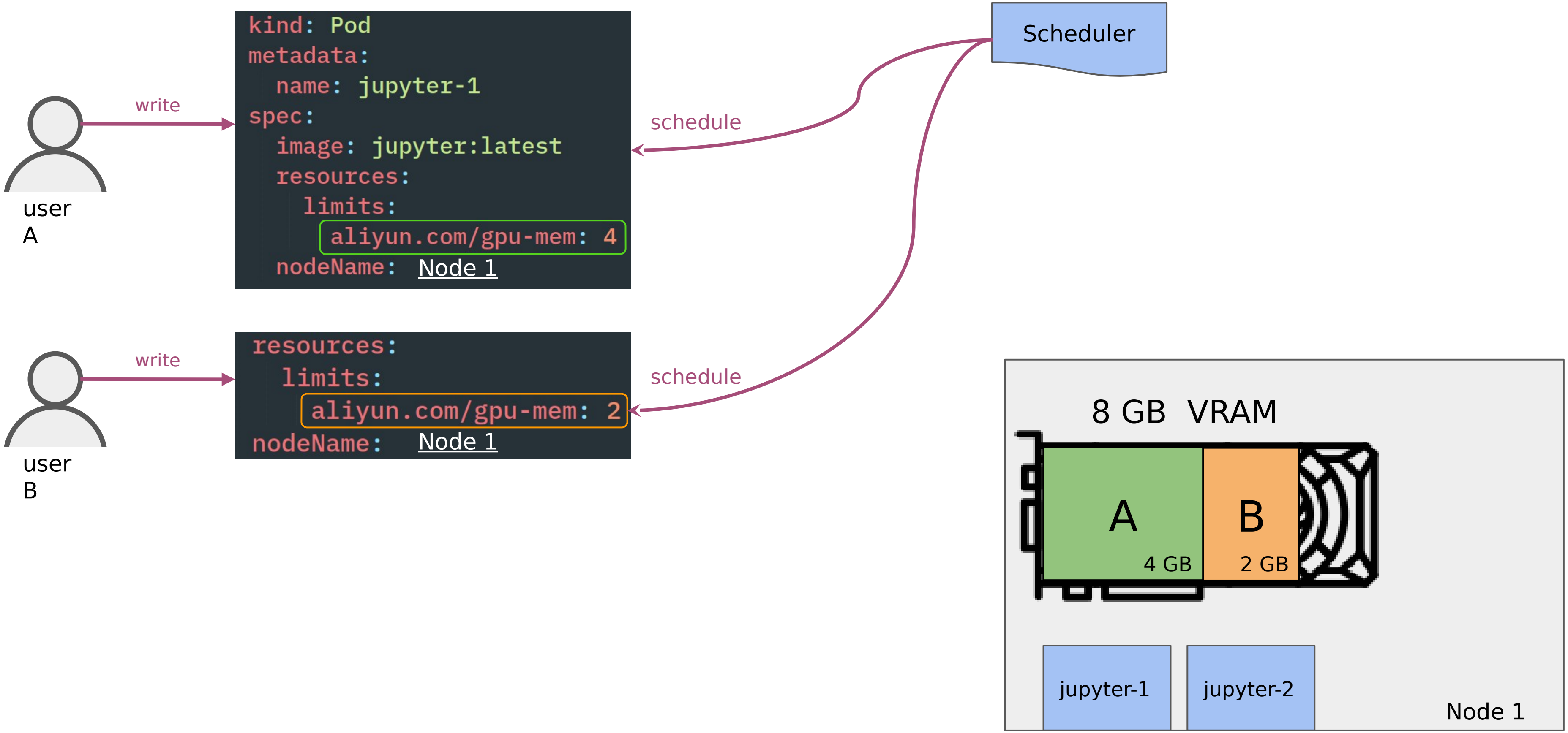
Existing Approaches



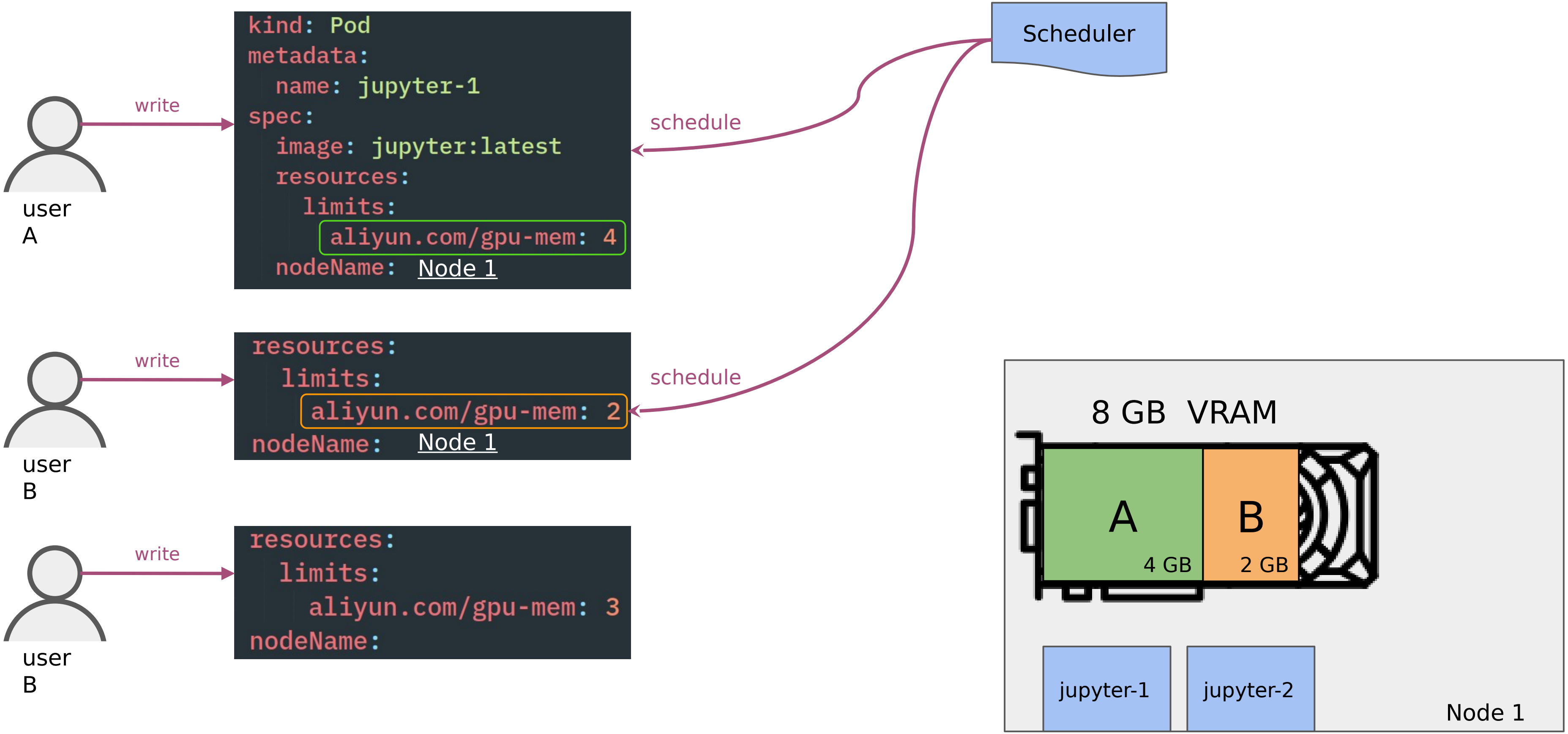
Existing Approaches



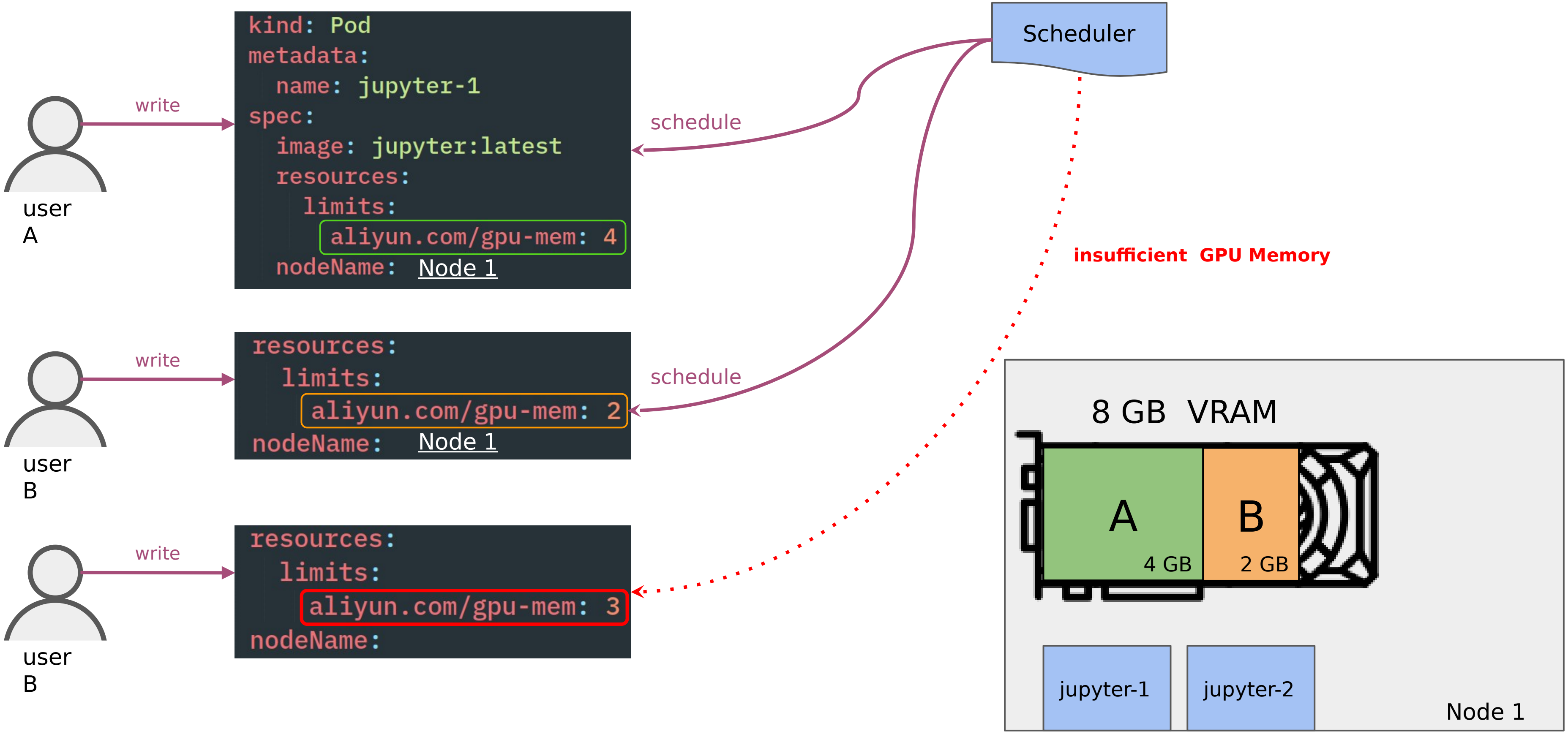
Existing Approaches



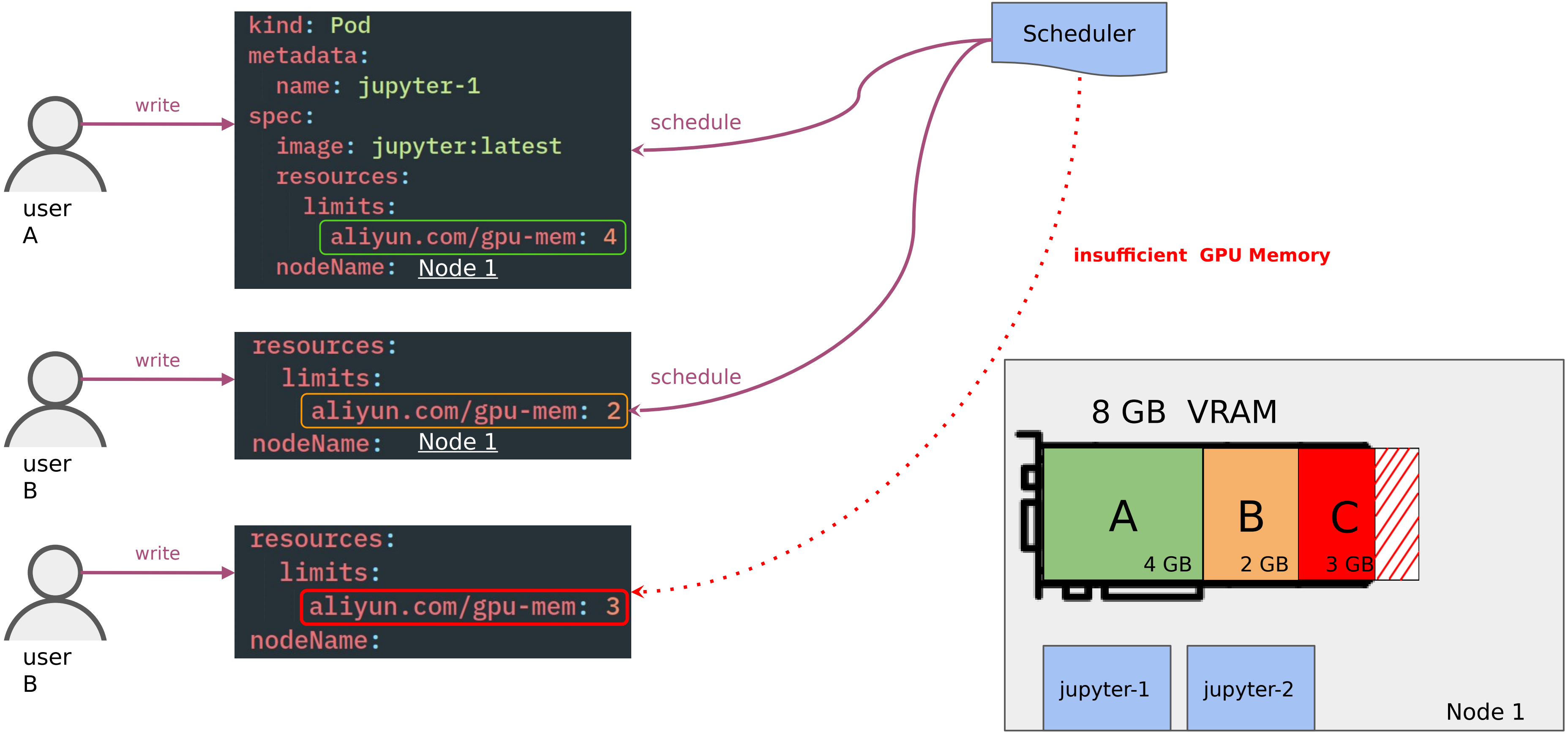
Existing Approaches



Existing Approaches



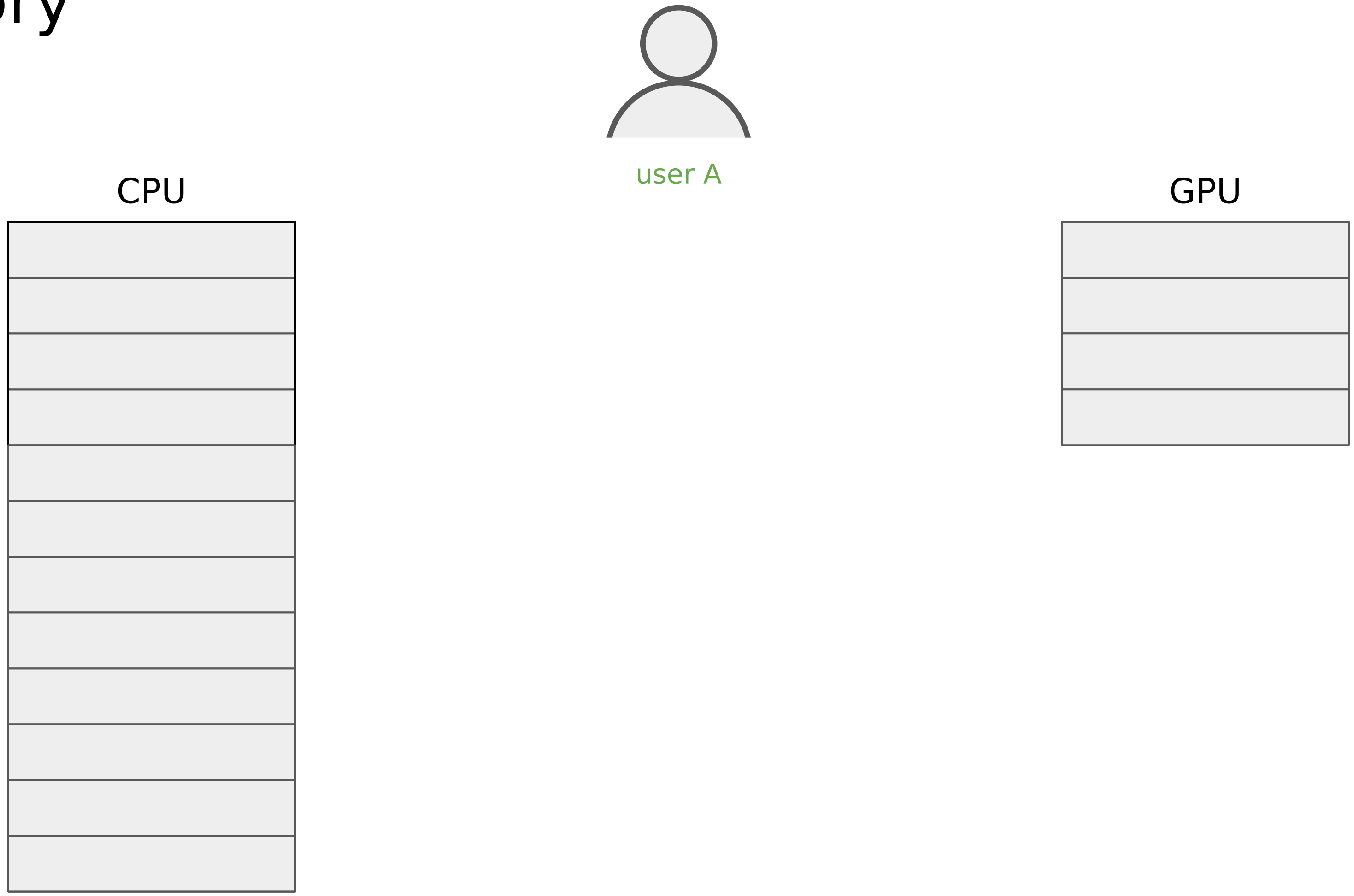
Existing Approaches



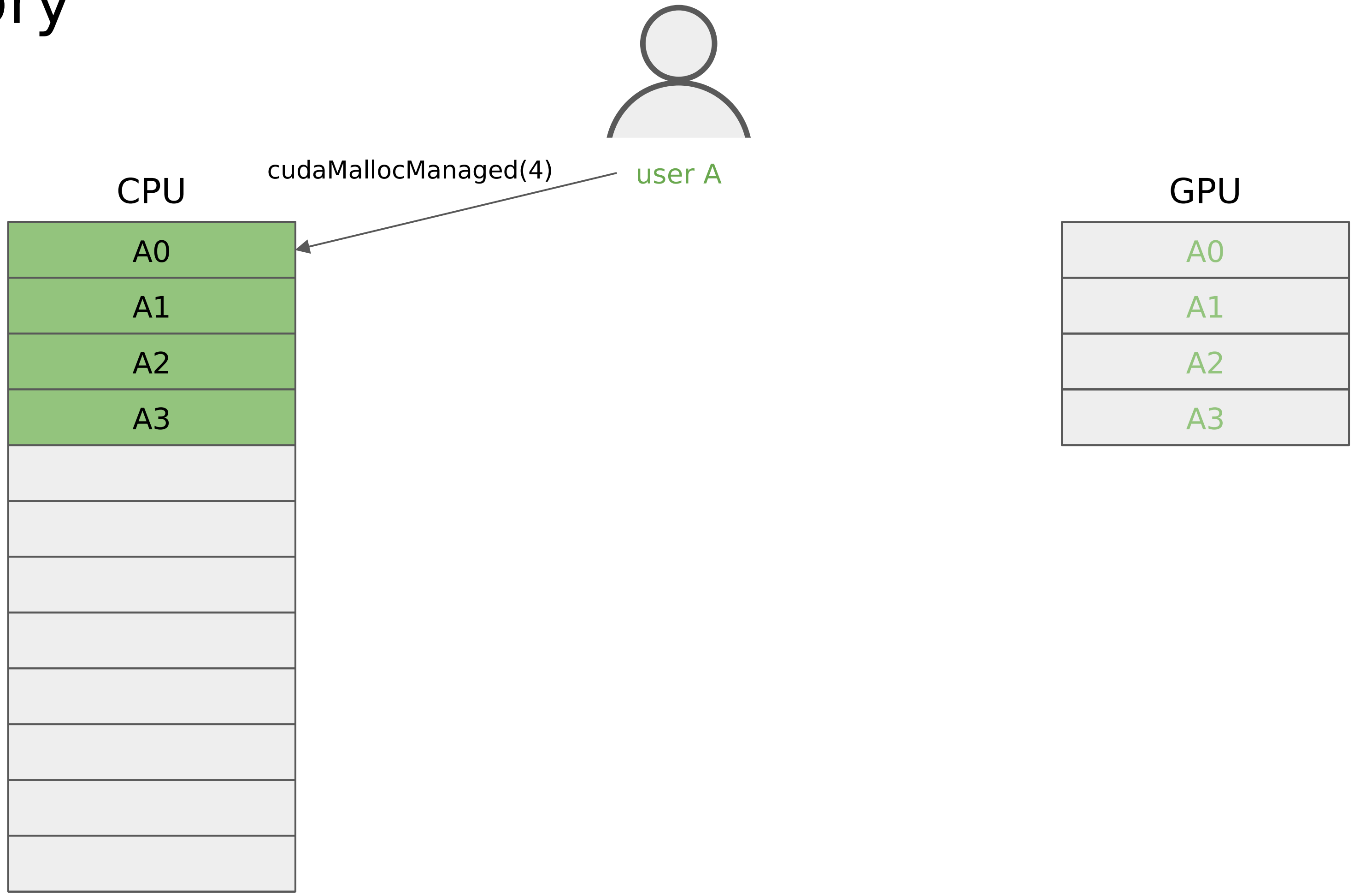
Existing Approaches

- Aliyun Scheduler Extender
  - GPUs no longer exclusively assigned to Pods
  - users specify GPU memory requests
  - uses GPU memory to bin pack Pods in Kubernetes
  - does not enforce memory limits (only used for scheduling)
- Kubeshare (HPDC '20)
  - similar to Aliyun w.r.t. the scheduling part
  - enforces GPU memory limits (assigns a memory slice to each process)
  - wraps CUDA API to keep track of how much memory each process allocates

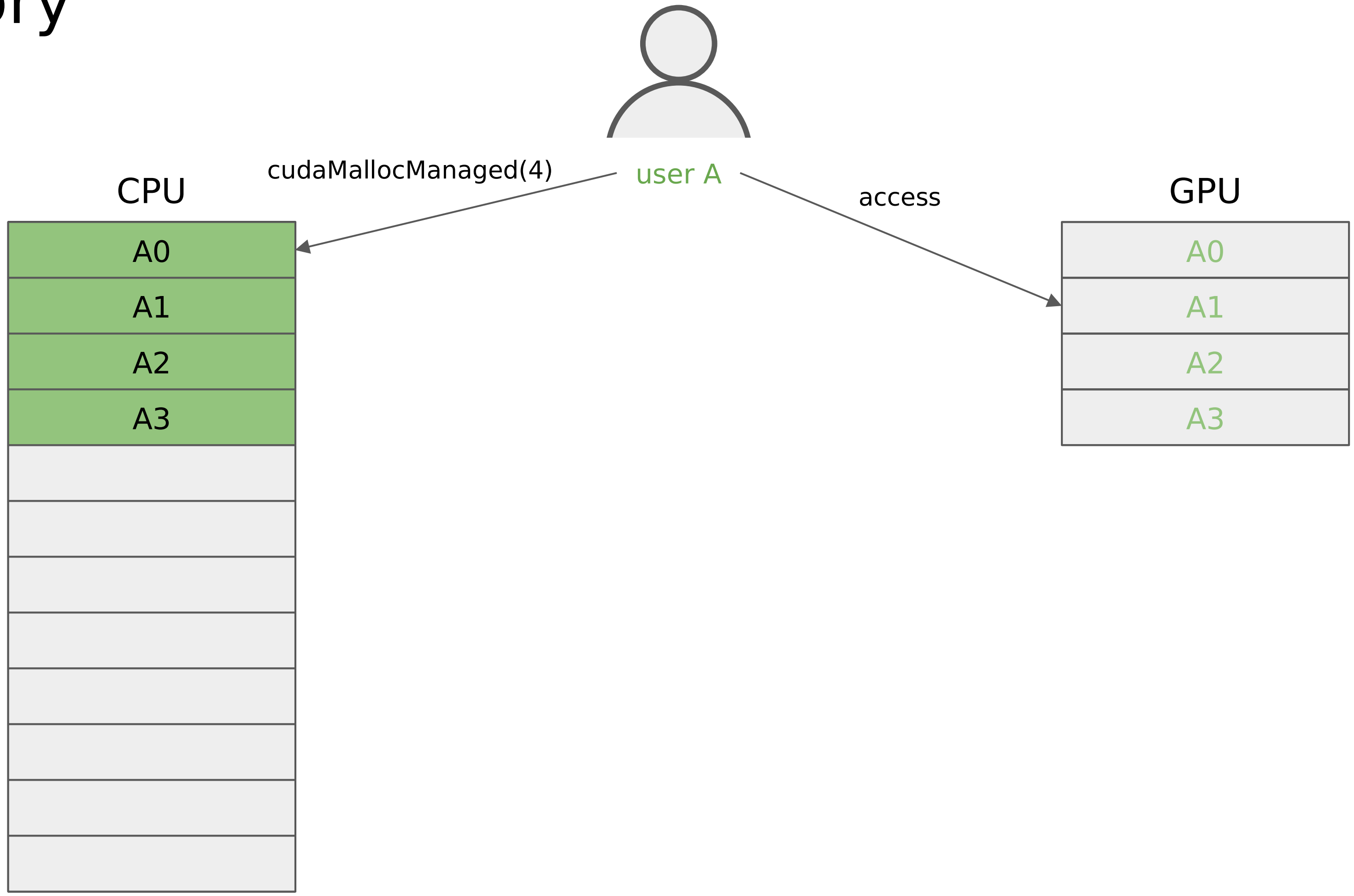
Memory



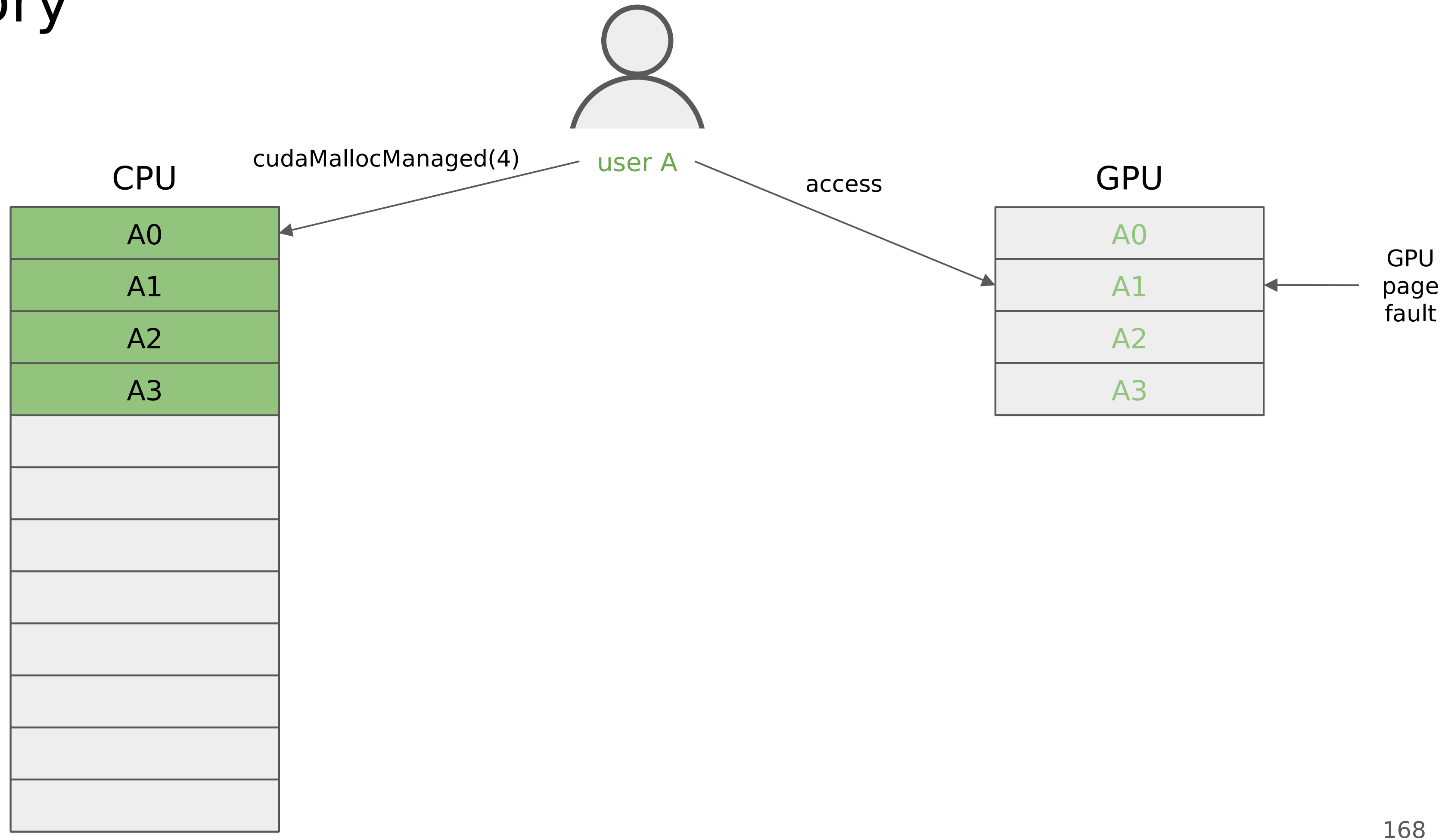
Memory



Memory

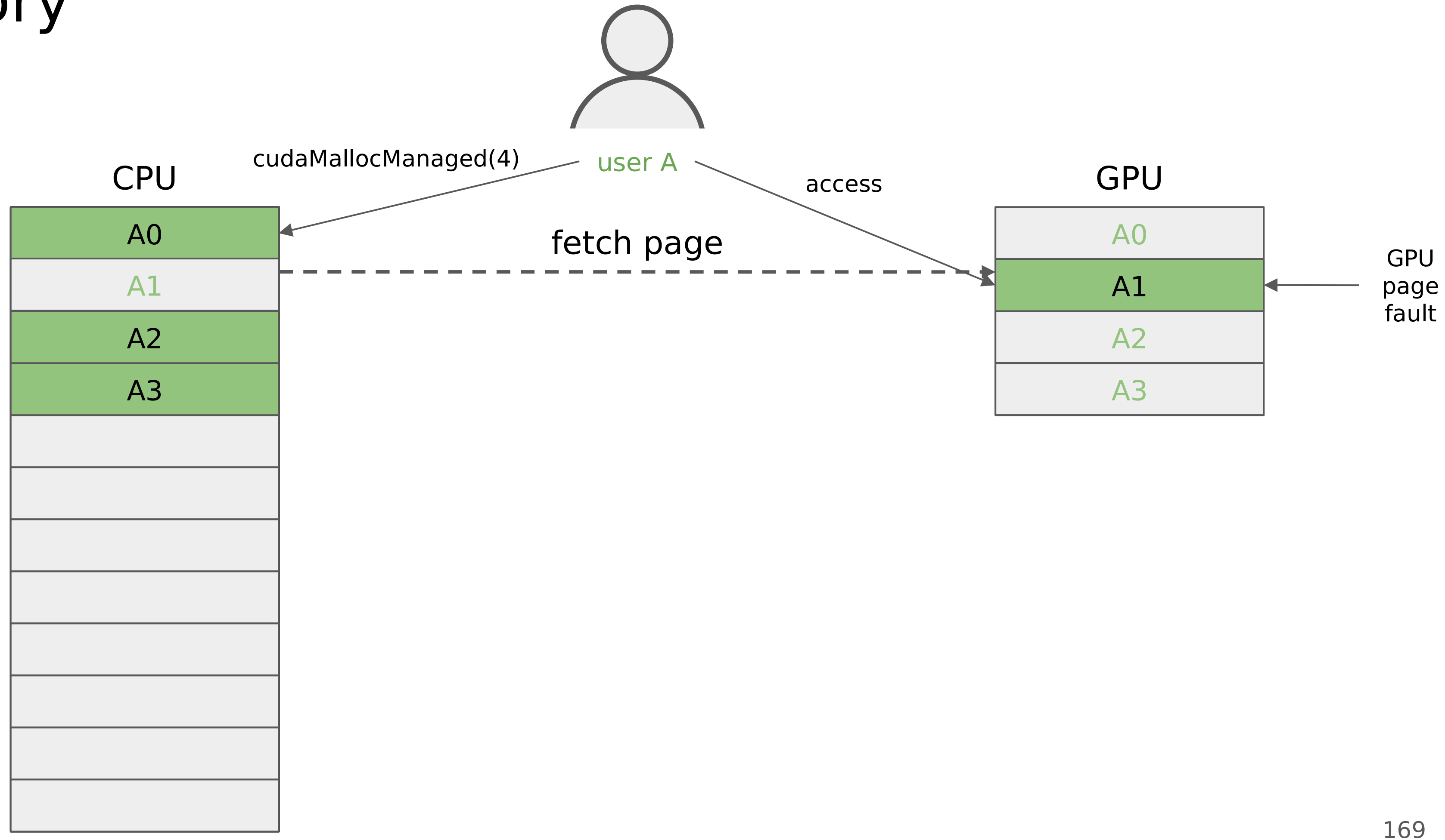

user A
cudaMallocManaged(4)
access
CPU
A0
A1
A2
A3
GPU
A0
A1
A2
A3

Memory



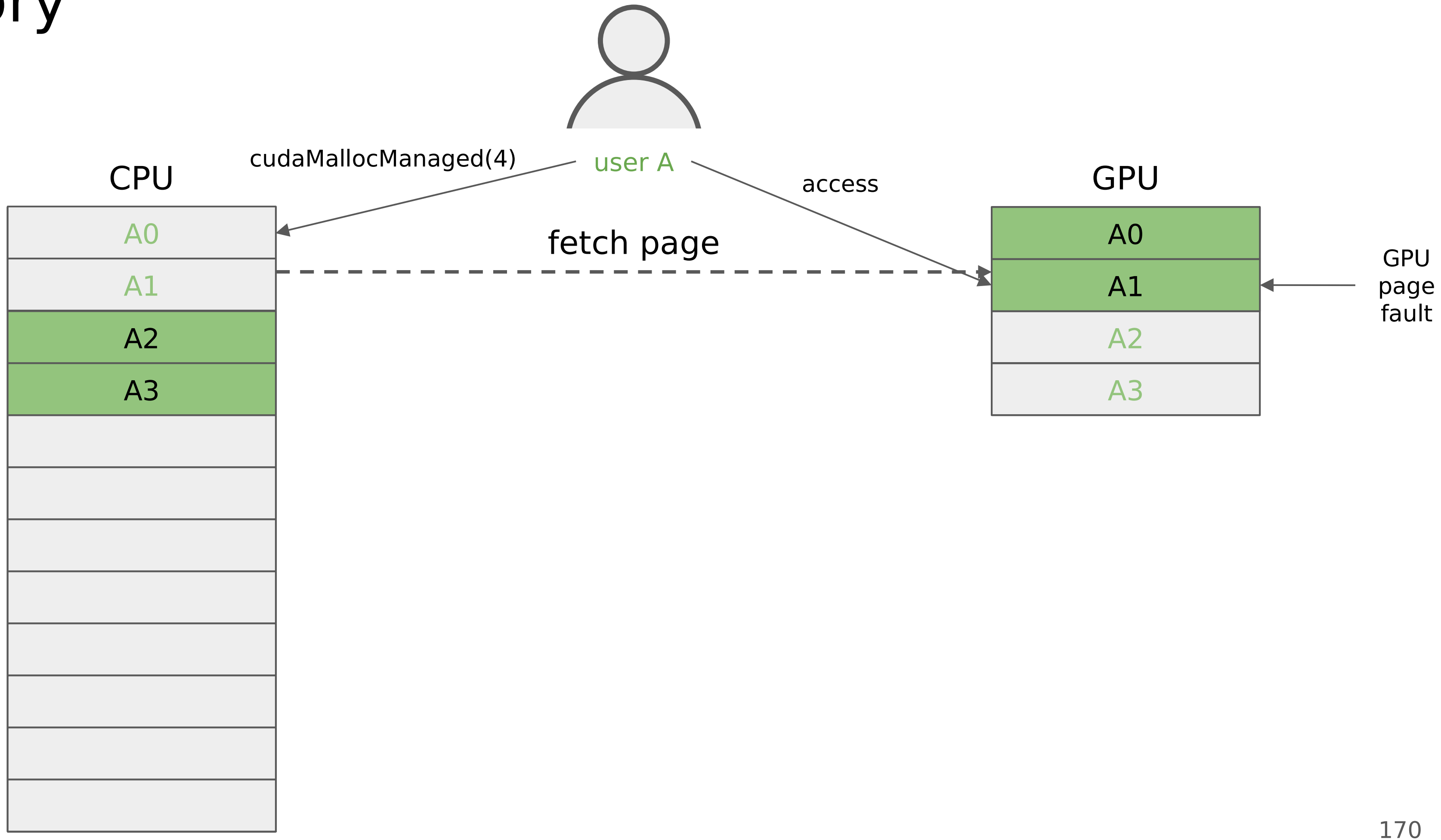
user A
cudaMallocManaged(4)
access
CPU
A0
A1
A2
A3
GPU
A0
A1
A2
A3
GPU
page
fault

Memory

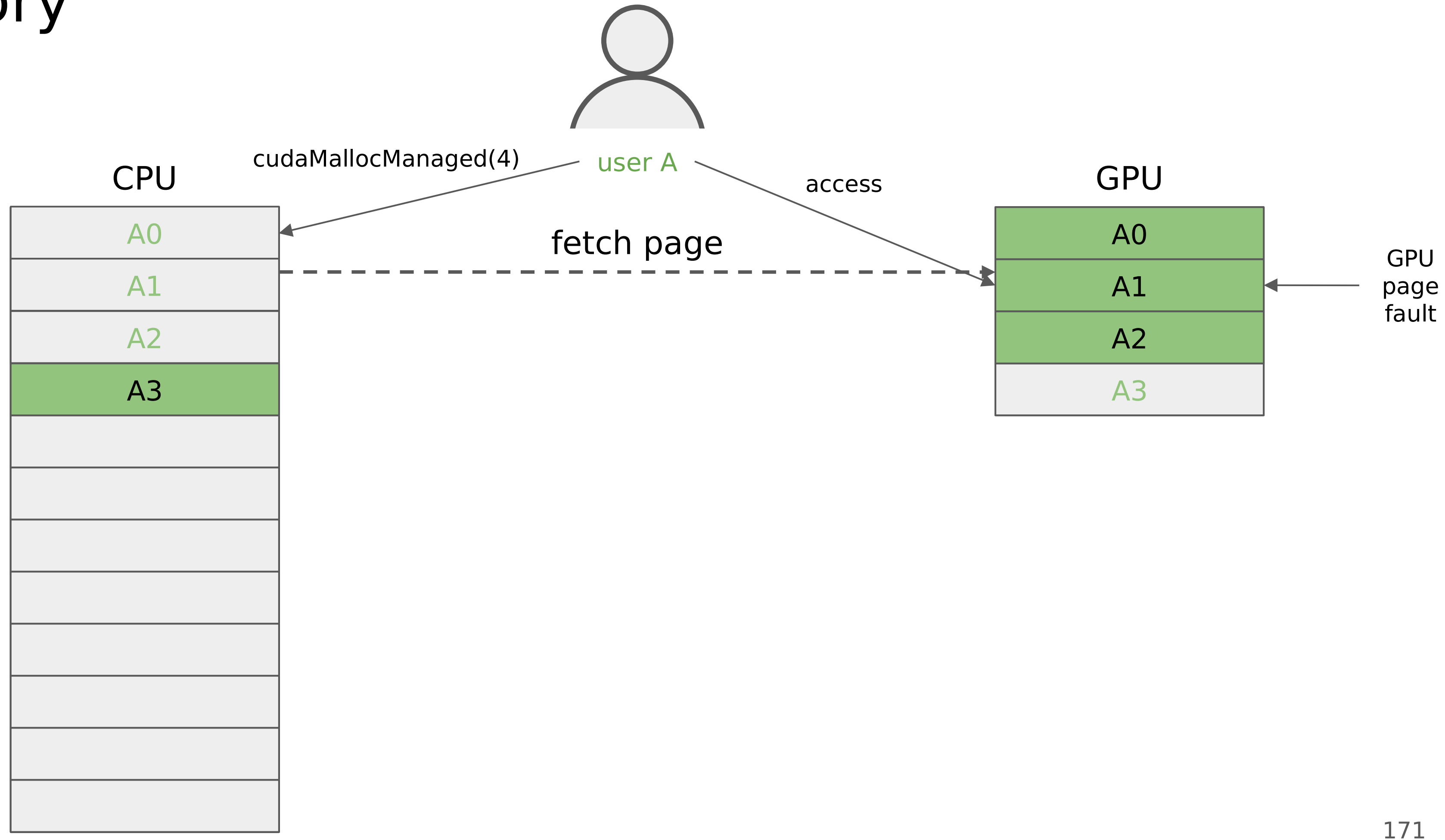


user A
cudaMallocManaged(4)
access
CPU
GPU
A0
A1
A2
A3
fetch page
A0
A1
A2
A3
GPU
page
fault

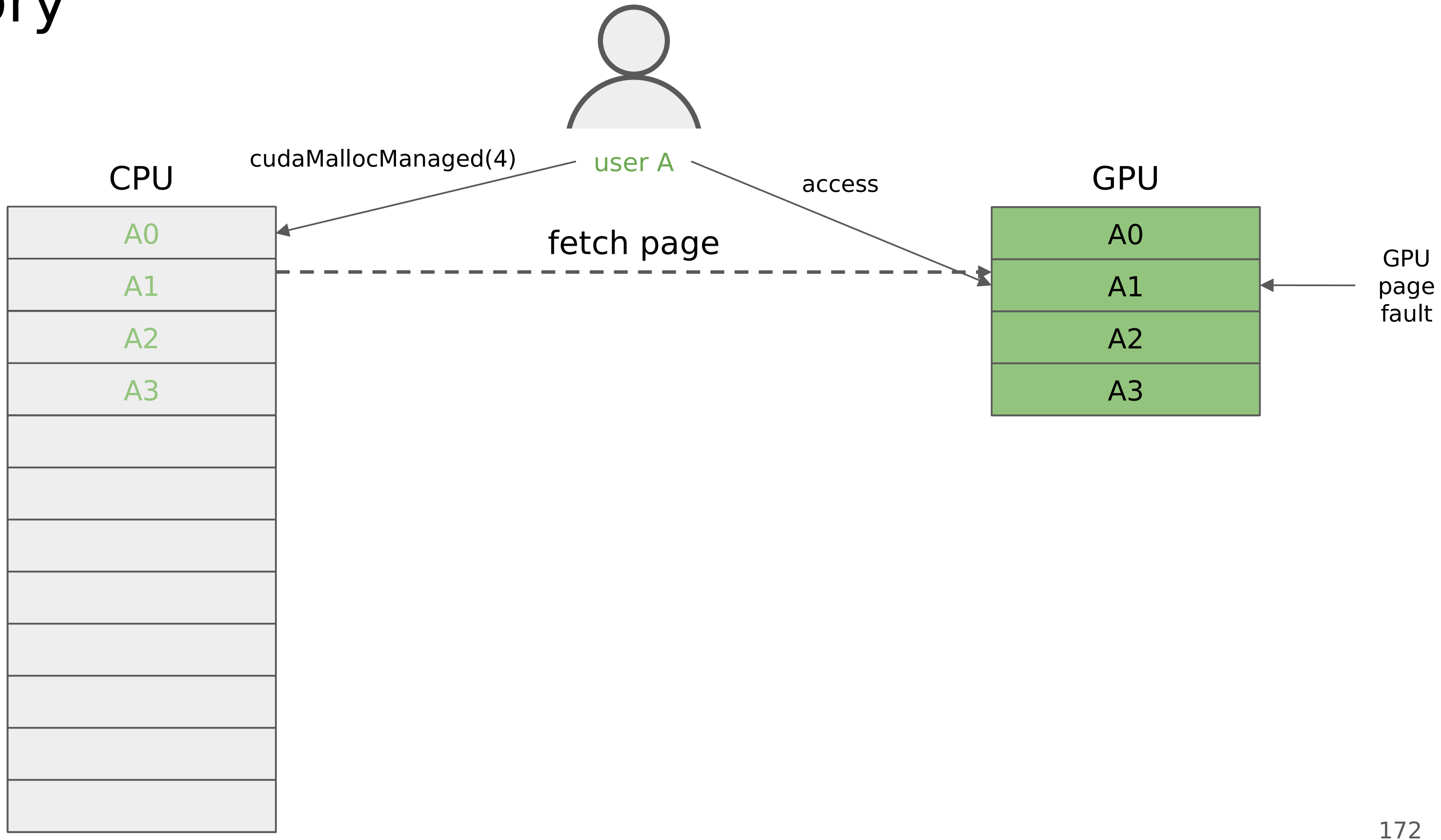
Memory


user A
cudaMallocManaged(4)
access
CPU
GPU
fetch page
A0
A1
A2
A3
A0
A1
A2
A3
GPU page fault

Memory

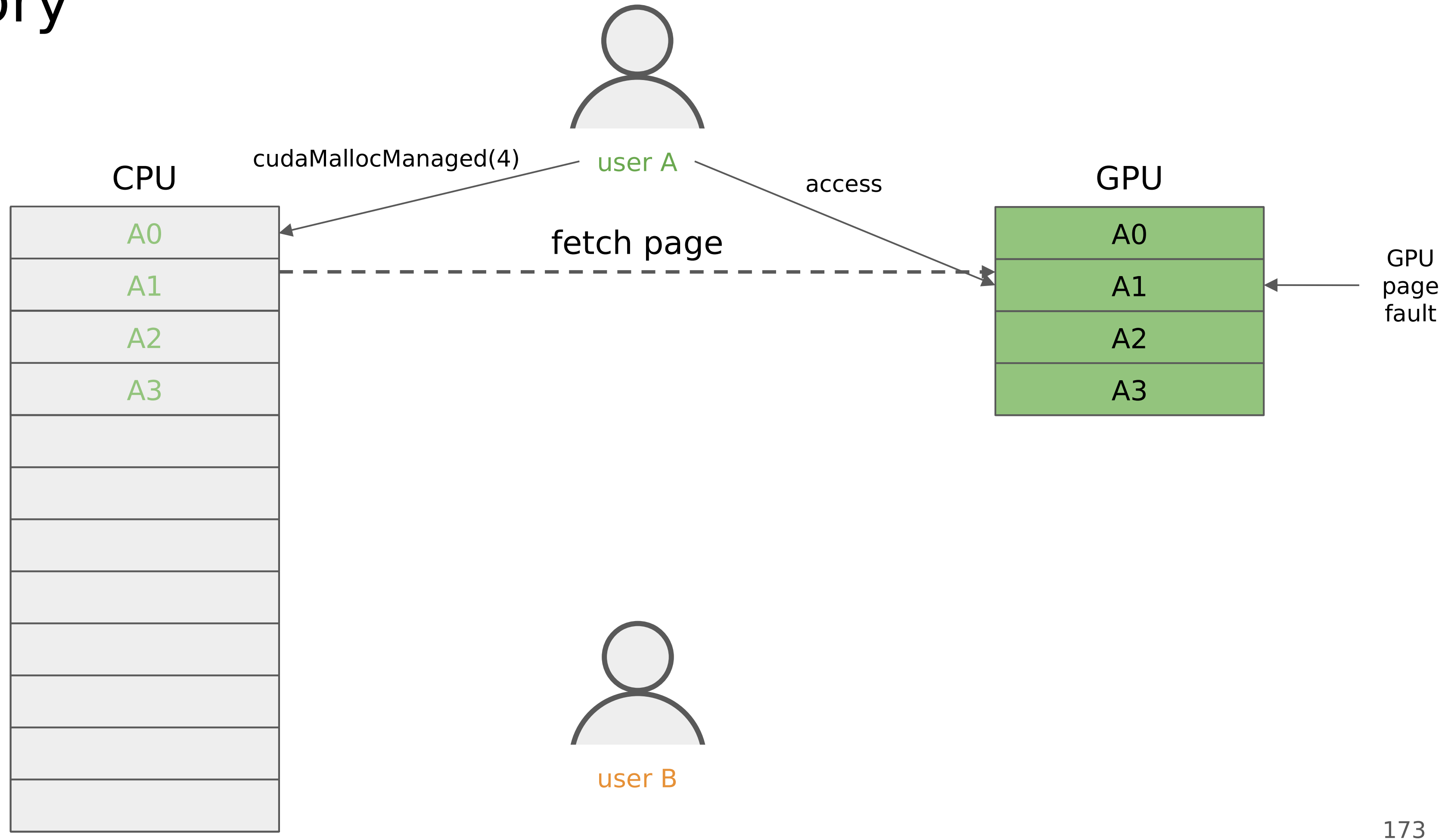

user A
cudaMallocManaged(4)
access
CPU
GPU
A0
A1
A2
A3
fetch page
A0
A1
A2
A3
GPU
page
fault

Memory



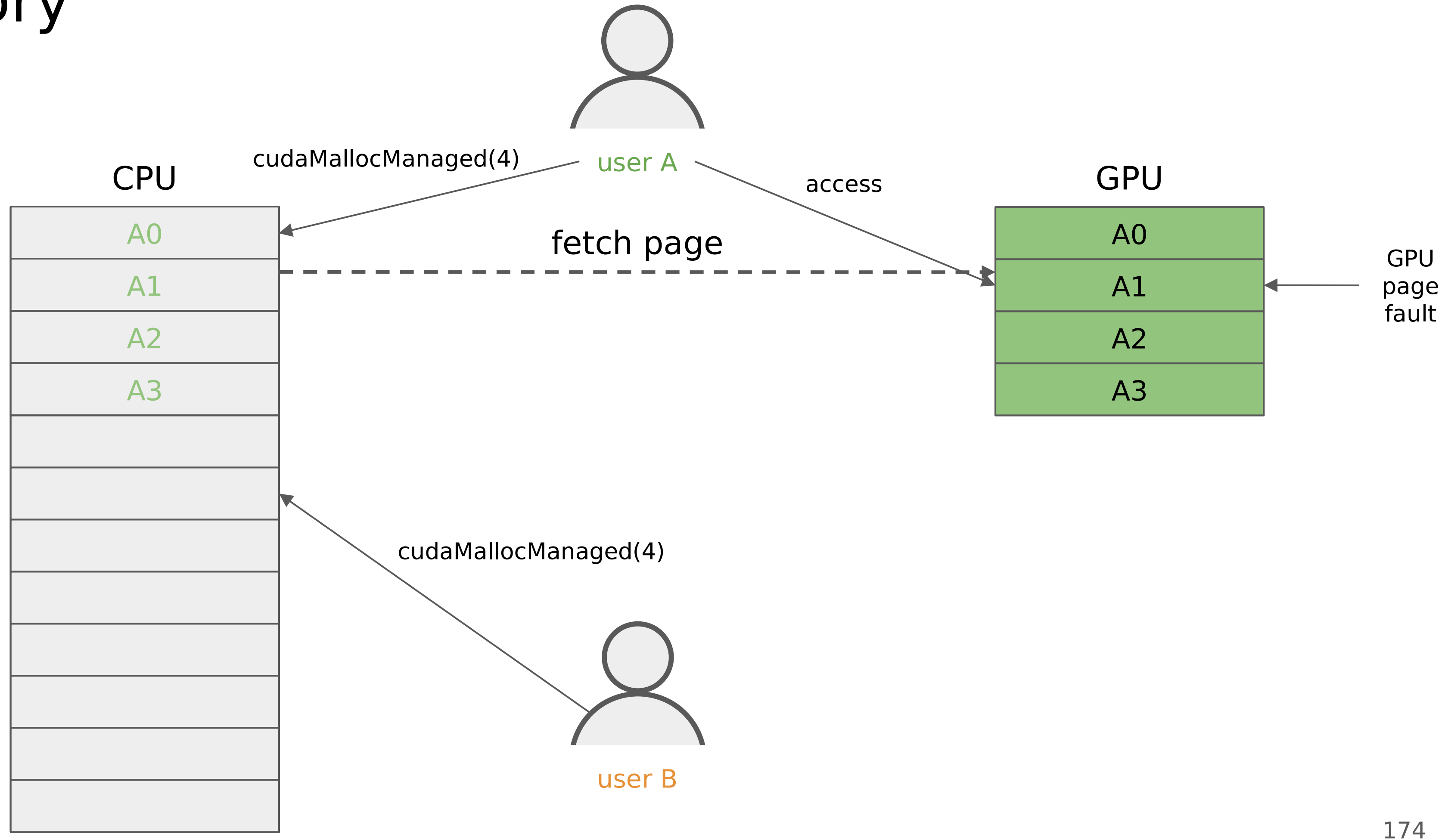
user A
cudaMallocManaged(4)
access
CPU
GPU
A0
A1
A2
A3
fetch page
A0
A1
A2
A3
GPU
page
fault

Memory



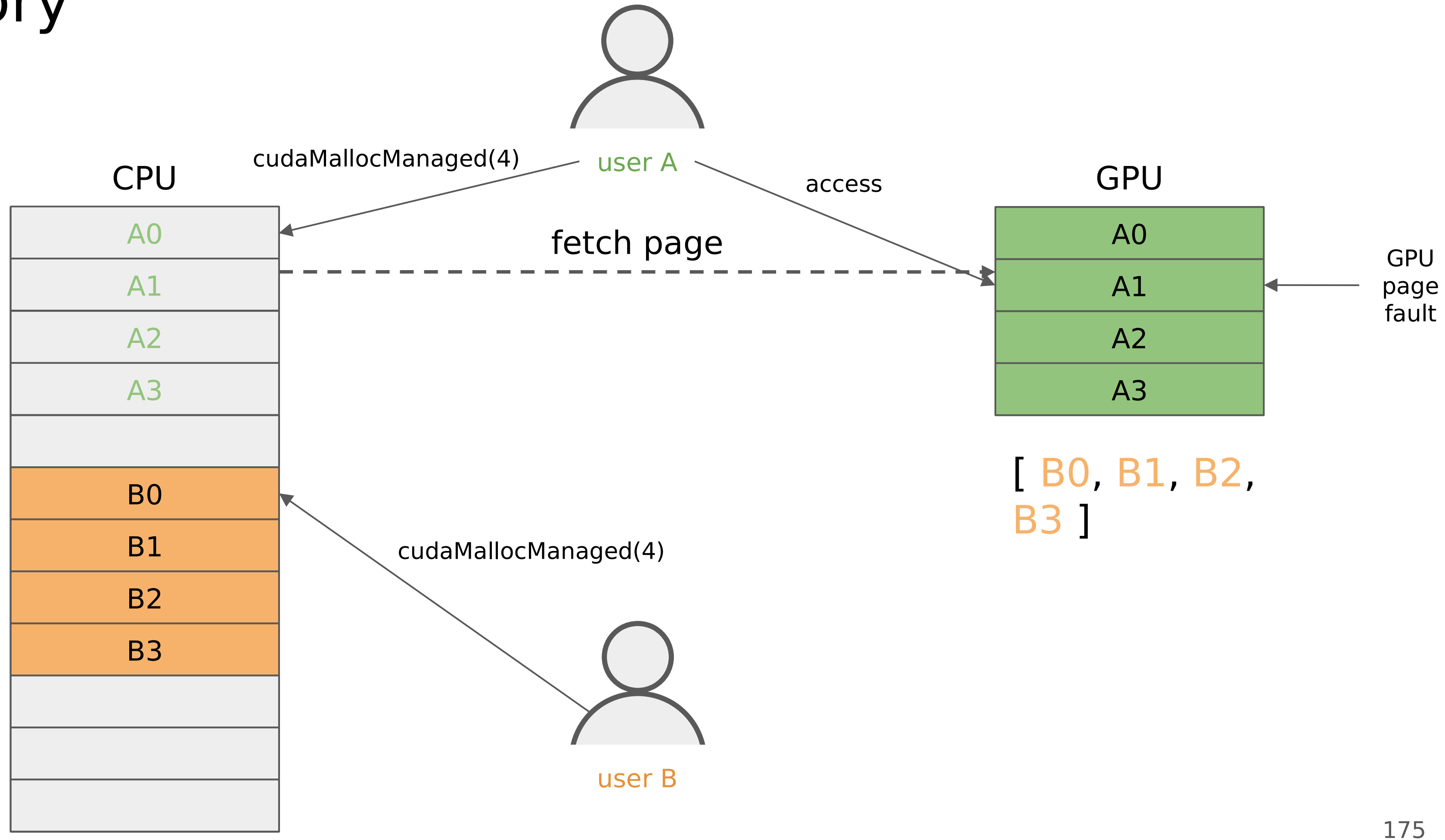
user A
cudaMallocManaged(4)
access
CPU
GPU
A0
A1
A2
A3
fetch page
A0
A1
A2
A3
GPU
page
fault
user B

Memory



user A
cudaMallocManaged(4)
access
CPU
GPU
A0
A1
A2
A3
fetch page
A0
A1
A2
A3
GPU page fault
cudaMallocManaged(4)
user B

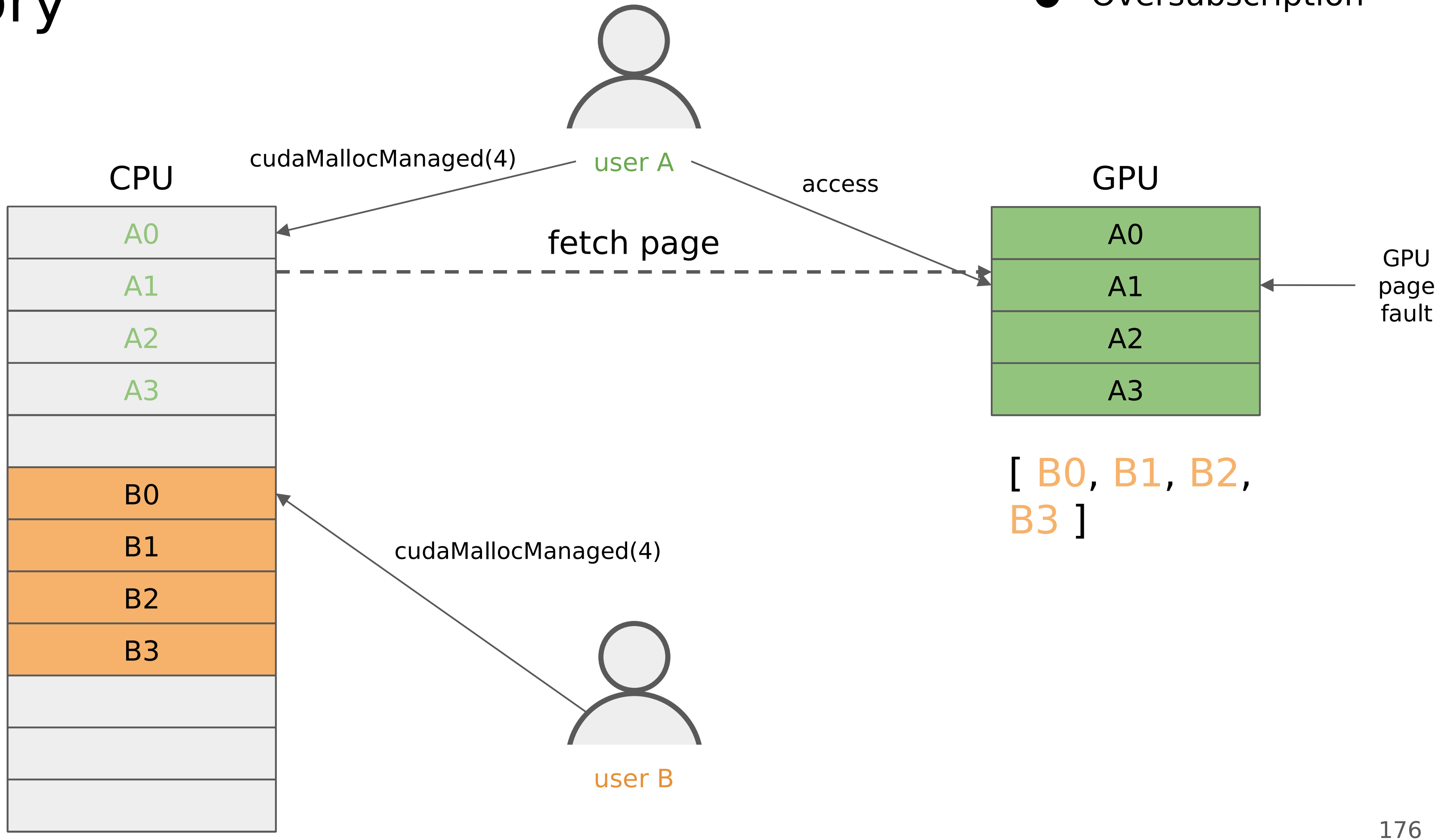
Memory



CPU
cudaMallocManaged(4)
user A
access
GPU
A0
A1
A2
A3
fetch page
A0
A1
A2
A3
GPU page fault
B0
B1
B2
B3
[B0, B1, B2, B3]
cudaMallocManaged(4)
user B

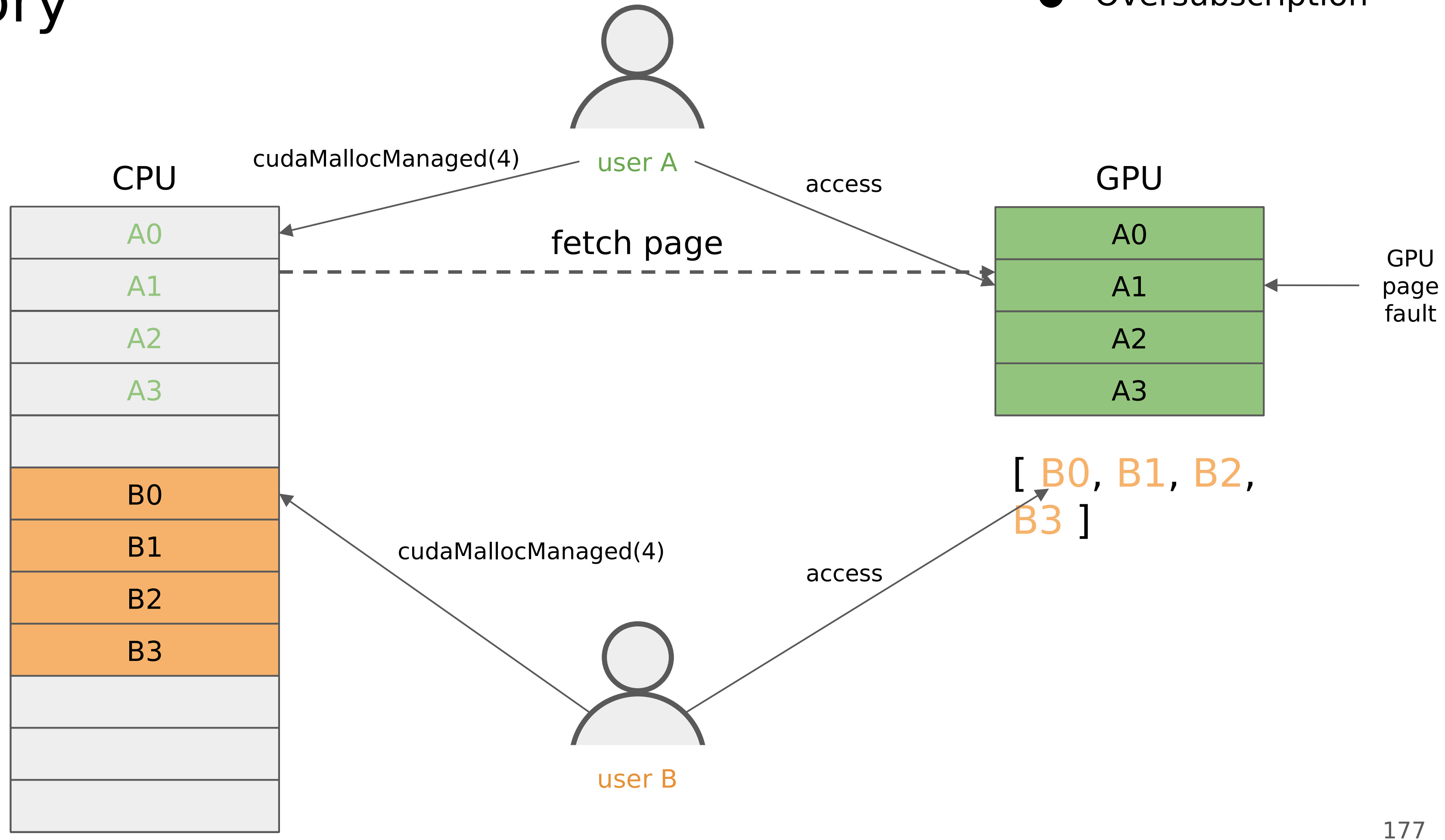
Memory

- Oversubscription



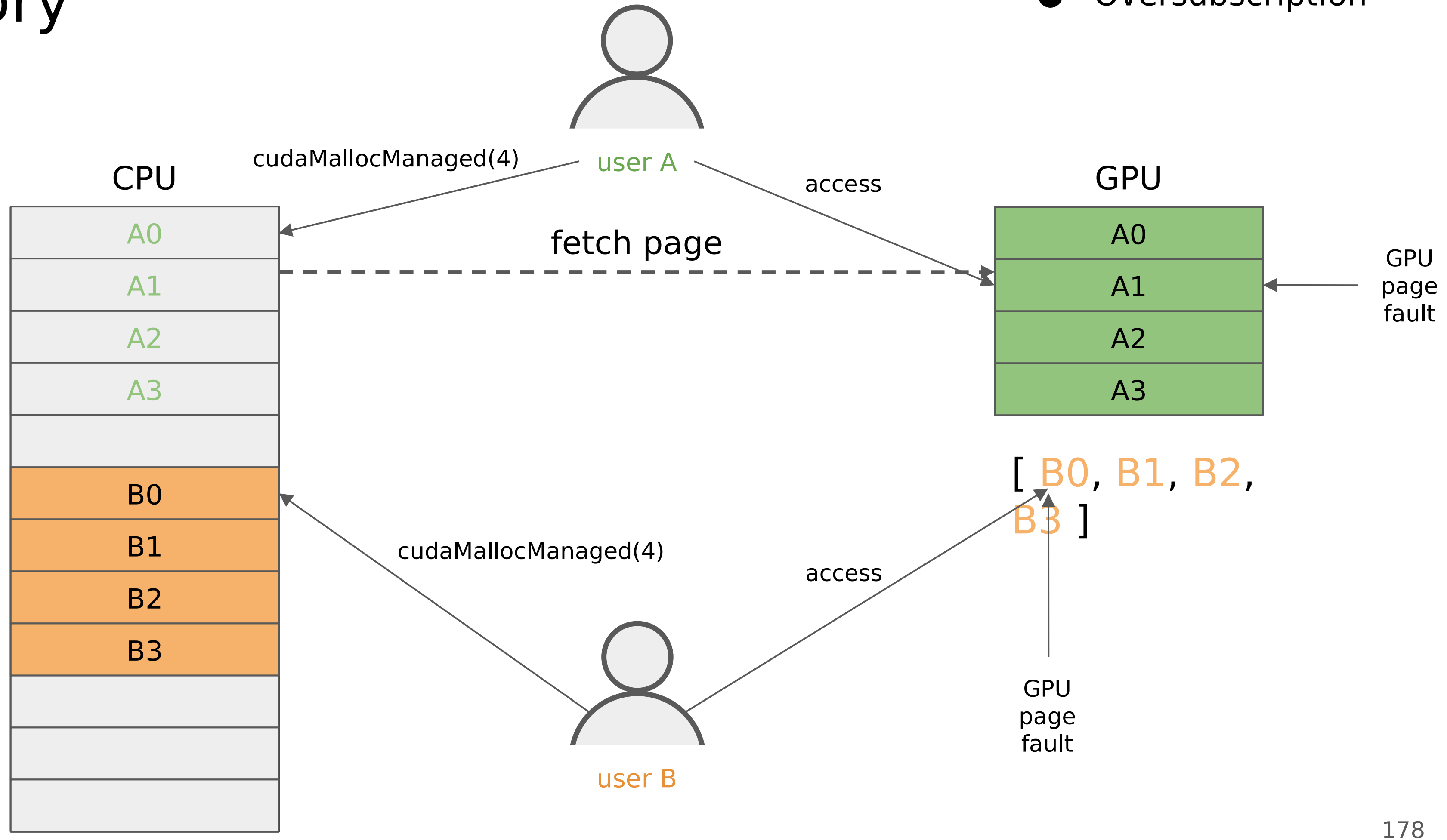
Memory

- Oversubscription



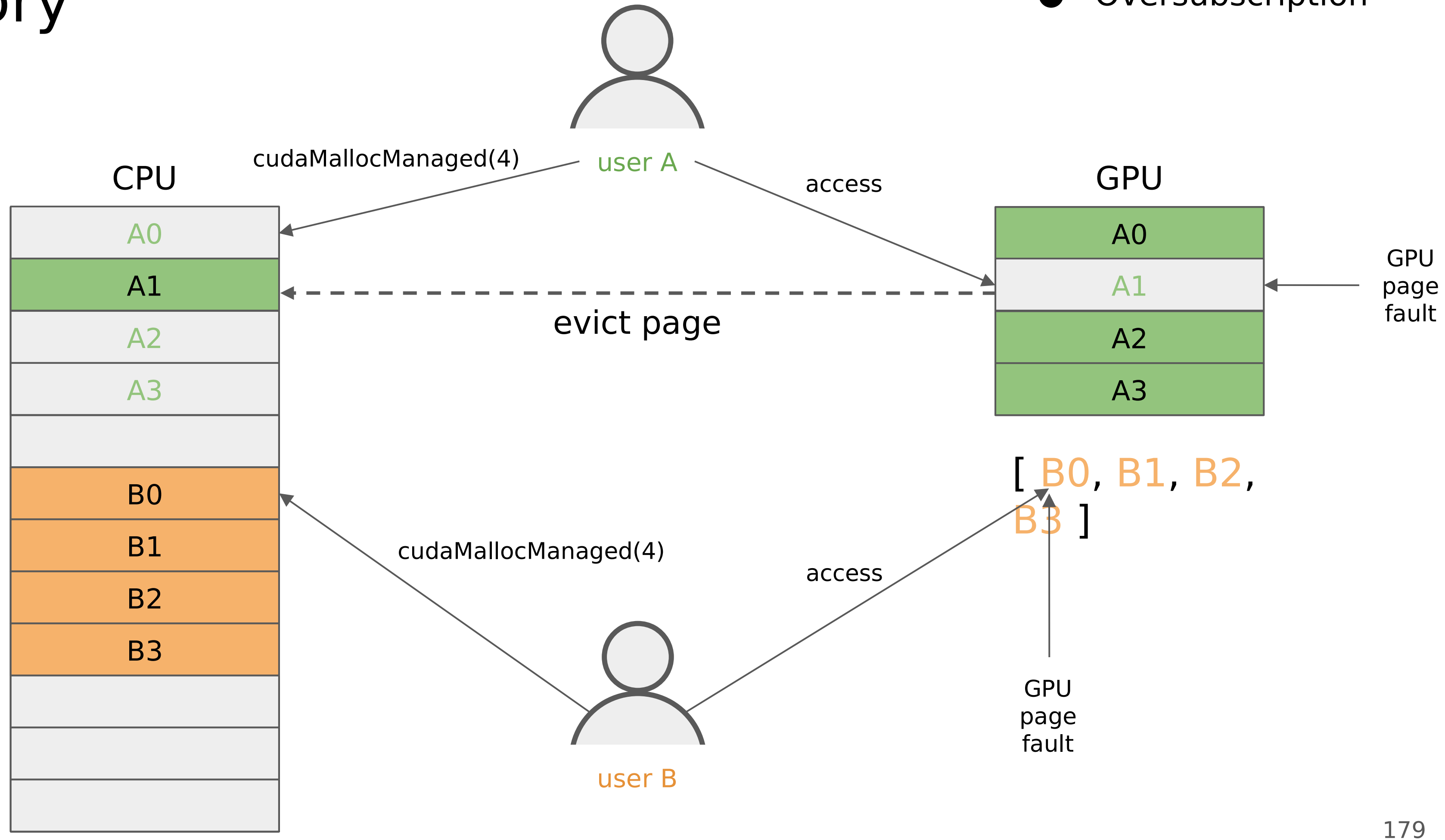
Memory

- Oversubscription



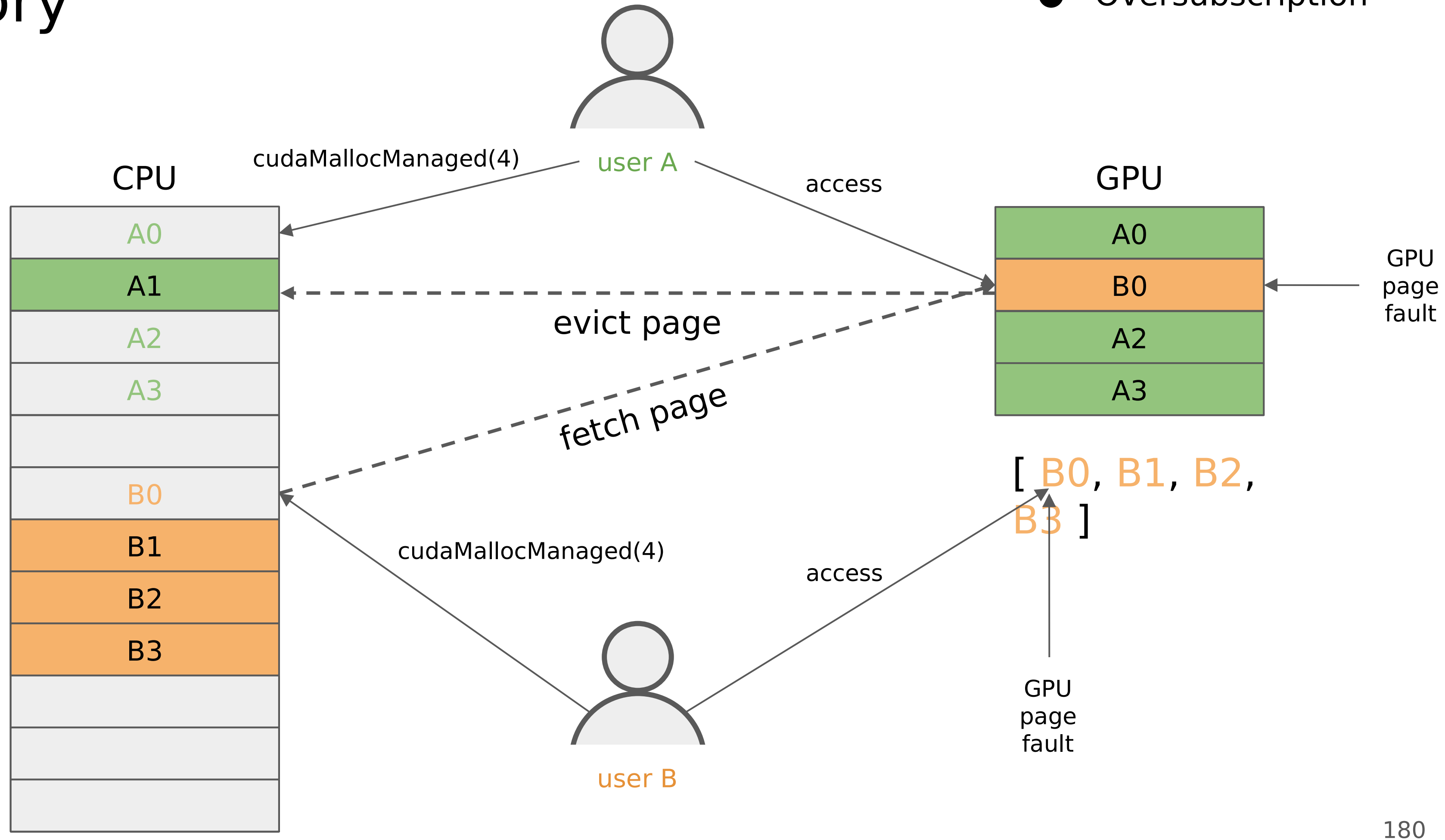
Memory

- Oversubscription



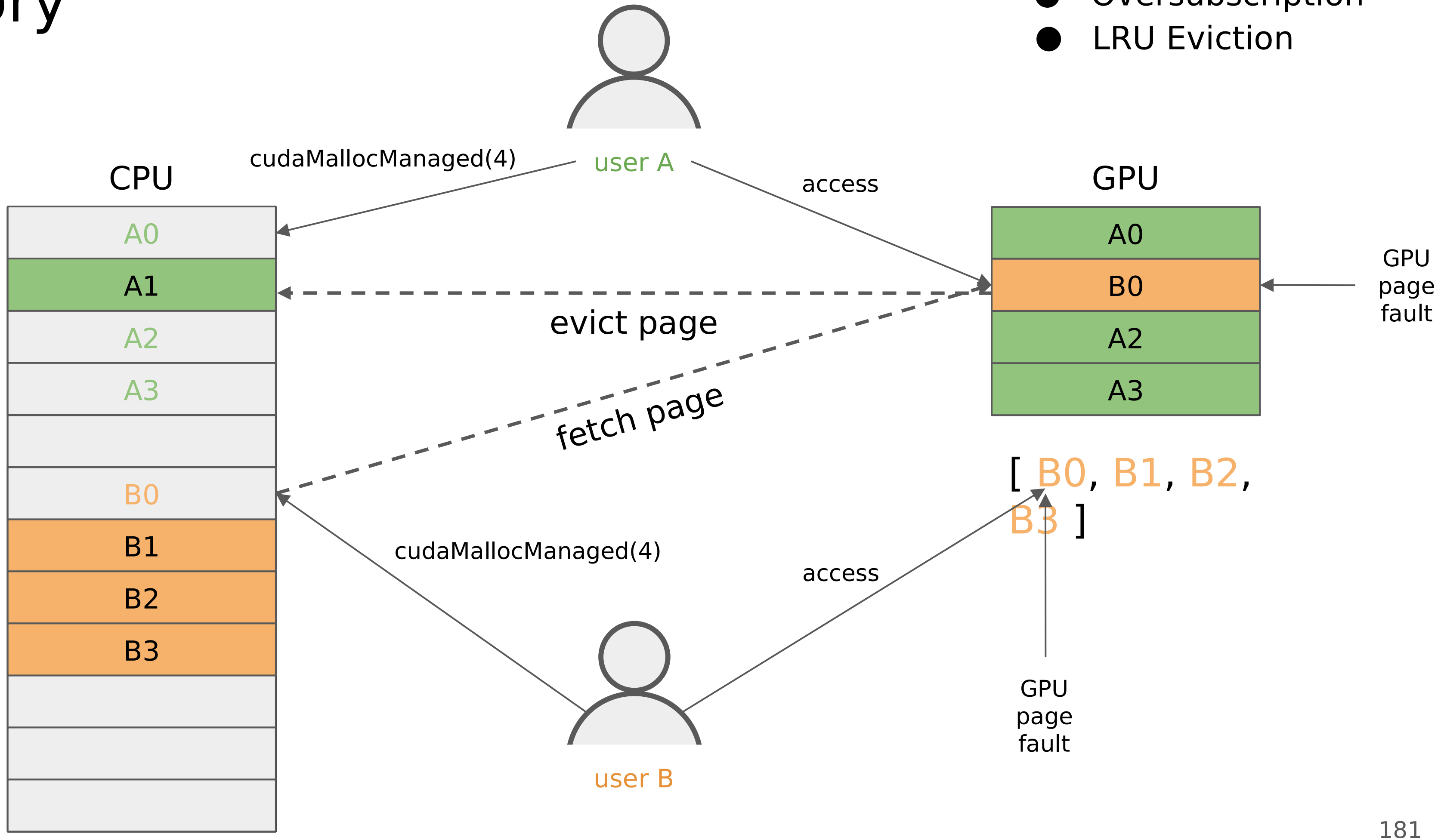
Memory

- Oversubscription



Memory

- Oversubscription
- LRU Eviction



Function wrappers

Function wrappers

- *execve()* loads executable code into memory and passes control to *ld.so*

Function wrappers

- *execve()* loads executable code into memory and passes control to *ld.so*
- *ld.so* maps shared objects to program address space and **resolves symbols**

Function wrappers

- *execve()* loads executable code into memory and passes control to *ld.so*
- *ld.so* maps shared objects to program address space and **resolves symbols**
- application begins execution

Function wrappers

- *execve()* loads executable code into memory and passes control to *ld.so*
- *ld.so* maps shared objects to program address space and **resolves symbols**
- application begins execution

```
$ nm -D ./matrixMul

  U cudaLaunchKernel
  U cudaMalloc
  U cudaMemcpy
```

Function wrappers

- *execve()* loads executable code into memory and passes control to *ld.so*
- *ld.so* maps shared objects to program address space and **resolves symbols**
- application begins execution

print dynamic symbol table entries

```
$ nm -D ./matrixMul

  U cudaLaunchKernel
  U cudaMalloc
  U cudaMemcpy
```

Function wrappers

- *execve()* loads executable code into memory and passes control to *ld.so*
- *ld.so* maps shared objects to program address space and **resolves symbols**
- application begins execution

print dynamic symbol table entries

```
$ nm -D ./matrixMul

  U cudaLaunchKernel
  U cudaMalloc
  U cudaMemcpy
```

Undefined Symbol

Function wrappers

- *execve()* loads executable code into memory and passes control to *ld.so*
- *ld.so* maps shared objects to program address space and **resolves symbols**
- application begins execution

print dynamic symbol table entries

```
$ nm -D ./matrixMul

  U cudaLaunchKernel
  U cudaMalloc
  U cudaMemcpy
```

Undefined Symbol

```
$ nm -D libcudart.so

..00002510 T cudaMalloc

$ nm -D libnvshare.so

..00002510 T cudaMalloc
```

Function wrappers

- *execve()* loads executable code into memory and passes control to *ld.so*
- *ld.so* maps shared objects to program address space and **resolves symbols**
- application begins execution

print dynamic symbol table entries

```
$ nm -D ./matrixMul

  U cudaLaunchKernel
  U cudaMalloc
  U cudaMemcpy
```

Undefined Symbol

```
$ nm -D libcudart.so

..00002510 T cudaMalloc

$ nm -D libnvshare.so

..00002510 T cudaMalloc
```

Symbol defined in .text

Function wrappers

- *execve()* loads executable code into memory and passes control to *ld.so*
- *ld.so* maps shared objects to program address space and **resolves symbols**
- application begins execution

print dynamic symbol table entries

```
$ nm -D ./matrixMul

  U cudaLaunchKernel
  U cudaMalloc
  U cudaMemcpy
```

Undefined Symbol

```
$ nm -D libcudart.so

..00002510 T cudaMalloc

$ nm -D libnvshare.so

..00002510 T cudaMalloc
```

Symbol defined in .text

our custom version

Function wrappers

- *execve()* loads executable code into memory and passes control to *ld.so*
- *ld.so* maps shared objects to program address space and **resolves symbols**
- application begins execution

print dynamic symbol table entries

```
$ nm -D ./matrixMul

  U cudaLaunchKernel
  U cudaMalloc
  U cudaMemcpy
```

Undefined Symbol

```
$ nm -D libcudart.so

..00002510 T cudaMalloc

$ nm -D libnvshare.so

..00002510 T cudaMalloc
```

Symbol defined in .text

our custom version

Function wrappers

- *execve()* loads executable code into memory and passes control to *ld.so*
- *ld.so* maps shared objects to program address space and **resolves symbols**
- application begins execution

print dynamic symbol table entries

```
$ nm -D ./matrixMul

  U cudaLaunchKernel
  U cudaMalloc
  U cudaMemcpy
```

```
$ nm -D libcudart.so

..00002510 T cudaMalloc

$ nm -D libnvshare.so

..00002510 T cudaMalloc
```

Undefined Symbol

Symbol defined in .text

our custom version

Normally:

Function wrappers

- *execve()* loads executable code into memory and passes control to *ld.so*
- *ld.so* maps shared objects to program address space and **resolves symbols**
- application begins execution

print dynamic symbol table entries

```
$ nm -D ./matrixMul

  U cudaLaunchKernel
  U cudaMalloc
  U cudaMemcpy
```

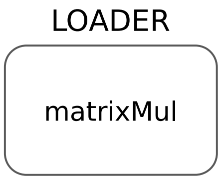
Undefined Symbol

```
$ nm -D libcudart.so

..00002510 T cudaMalloc

$ nm -D libnvshare.so

..00002510 T cudaMalloc
```

Symbol defined in .text

our custom version

Normally:



Function wrappers

- *execve()* loads executable code into memory and passes control to *ld.so*
- *ld.so* maps shared objects to program address space and **resolves symbols**
- application begins execution

print dynamic symbol table entries

```
$ nm -D ./matrixMul

  U cudaLaunchKernel
  U cudaMalloc
  U cudaMemcpy
```

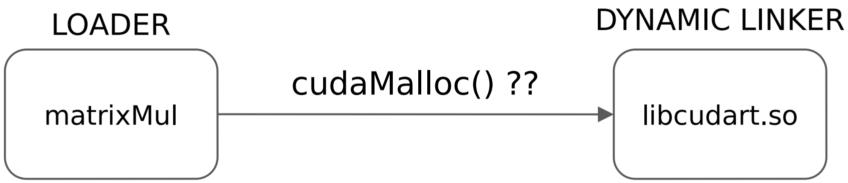
Undefined Symbol

```
$ nm -D libcudart.so

..00002510 T cudaMalloc

$ nm -D libnvshare.so

..00002510 T cudaMalloc
```

Symbol defined in .text

our custom version

Normally:



Function wrappers

- *execve()* loads executable code into memory and passes control to *ld.so*
- *ld.so* maps shared objects to program address space and **resolves symbols**
- application begins execution

print dynamic symbol table entries

```
$ nm -D ./matrixMul

  U cudaLaunchKernel
  U cudaMalloc
  U cudaMemcpy
```

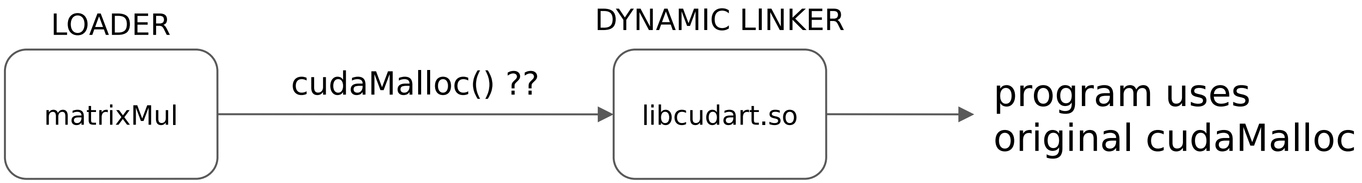
Undefined Symbol

```
$ nm -D libcudart.so

..00002510 T cudaMalloc

$ nm -D libnvshare.so

..00002510 T cudaMalloc
```

Symbol defined in .text

our custom version

Normally:



Function wrappers

- *execve()* loads executable code into memory and passes control to *ld.so*
- *ld.so* maps shared objects to program address space and **resolves symbols**
- application begins execution

print dynamic symbol table entries

```
$ nm -D ./matrixMul

  U cudaLaunchKernel
  U cudaMalloc
  U cudaMemcpy
```

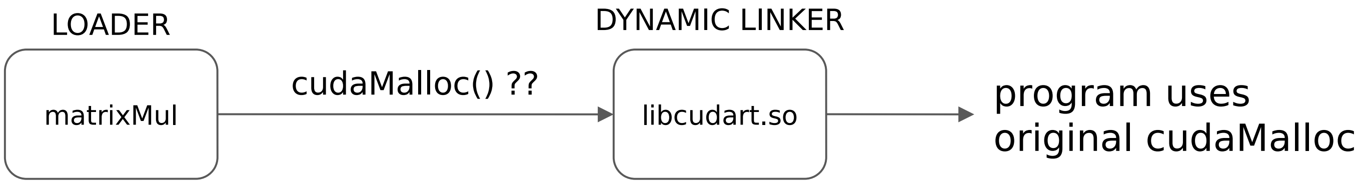
```
$ nm -D libcudart.so

..00002510 T cudaMalloc

$ nm -D libnvshare.so

..00002510 T cudaMalloc
```

Undefined Symbol

Symbol defined in .text

our custom version

Normally:



Function wrappers

- *execve()* loads executable code into memory and passes control to *ld.so*
- *ld.so* maps shared objects to program address space and **resolves symbols**
- application begins execution

print dynamic symbol table entries

```
$ nm -D ./matrixMul

  U cudaLaunchKernel
  U cudaMalloc
  U cudaMemcpy
```

Undefined Symbol

```
$ nm -D libcudart.so

..00002510 T cudaMalloc

$ nm -D libnvshare.so

..00002510 T cudaMalloc
```

Symbol defined in .text

our custom version

Normally:

LOADER

matrixMul

cudaMalloc() ??

DYNAMIC LINKER

libcudart.so

program uses original cudaMalloc

LD_PRELOAD=libnvshare.so

Function wrappers

- *execve()* loads executable code into memory and passes control to *ld.so*
- *ld.so* maps shared objects to program address space and **resolves symbols**
- application begins execution

print dynamic symbol table entries

```
$ nm -D ./matrixMul

  U cudaLaunchKernel
  U cudaMalloc
  U cudaMemcpy
```

Undefined Symbol

```
$ nm -D libcudart.so

..00002510 T cudaMalloc

$ nm -D libnvshare.so

..00002510 T cudaMalloc
```

Symbol defined in .text

our custom version

Normally:

LOADER

matrixMul

cudaMalloc() ??

DYNAMIC LINKER

libcudart.so

program uses original cudaMalloc

LD_PRELOAD=libnvshare.so

LOADER

matrixMul

Function wrappers

- *execve()* loads executable code into memory and passes control to *ld.so*
- *ld.so* maps shared objects to program address space and **resolves symbols**
- application begins execution

print dynamic symbol table entries

```
$ nm -D ./matrixMul

  U cudaLaunchKernel
  U cudaMalloc
  U cudaMemcpy
```

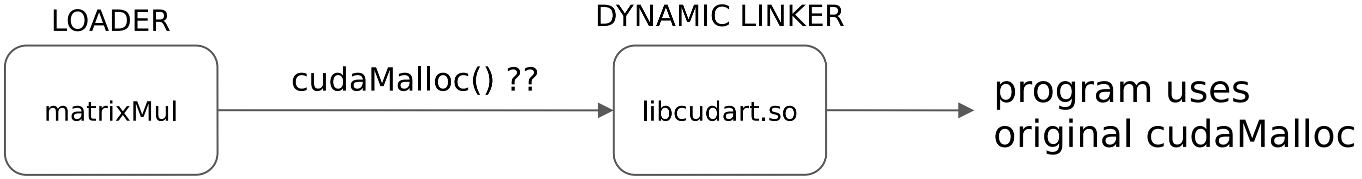
Undefined Symbol

```
$ nm -D libcudart.so

..00002510 T cudaMalloc

$ nm -D libnvshare.so

..00002510 T cudaMalloc
```

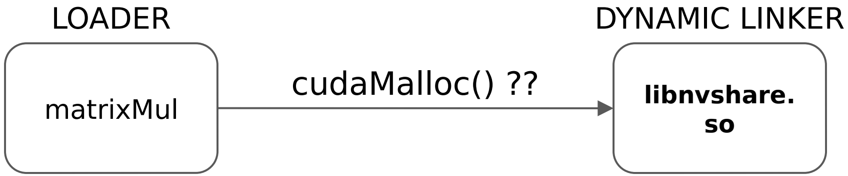
Symbol defined in .text

our custom version

Normally:



LD_PRELOAD=libnvshare.so



Function wrappers

- *execve()* loads executable code into memory and passes control to *ld.so*
- *ld.so* maps shared objects to program address space and **resolves symbols**
- application begins execution

print dynamic symbol table entries

```
$ nm -D ./matrixMul

  U cudaLaunchKernel
  U cudaMalloc
  U cudaMemcpy
```

Undefined Symbol

```
$ nm -D libcudart.so

..00002510 T cudaMalloc

$ nm -D libnvshare.so

..00002510 T cudaMalloc
```

Symbol defined in .text

our custom version

Normally:

LOADER: matrixMul → cudaMalloc() ?? → DYNAMIC LINKER: libcudart.so → program uses original cudaMalloc

LD_PRELOAD=libnvshare.so

LOADER: matrixMul → cudaMalloc() ?? → DYNAMIC LINKER: **libnvshare.so** → program uses our version

Function wrappers

- *execve()* loads executable code into memory and passes control to *ld.so*
- *ld.so* maps shared objects to program address space and **resolves symbols**
- application begins execution

print dynamic symbol table entries

```
$ nm -D ./matrixMul

  U cudaLaunchKernel
  U cudaMalloc
  U cudaMemcpy
```

Undefined Symbol

```
$ nm -D libcudart.so

..00002510 T cudaMalloc

$ nm -D libnvshare.so

..00002510 T cudaMalloc
```

Symbol defined in .text

our custom version

Normally:

LOADER: matrixMul → cudaMalloc() ?? → DYNAMIC LINKER: libcudart.so → program uses original cudaMalloc

LD_PRELOAD=libnvshare.so

LOADER: matrixMul → cudaMalloc() ?? → DYNAMIC LINKER: **libnvshare.so** → program uses our version

- which can **internally** call the original

Function wrappers

- *execve()* loads executable code into memory and passes control to *ld.so*
- *ld.so* maps shared objects to program address space and **resolves symbols**
- application begins execution

print dynamic symbol table entries

```
$ nm -D ./matrixMul

  U cudaLaunchKernel
  U cudaMalloc
  U cudaMemcpy
```

Undefined Symbol

```
$ nm -D libcudart.so

..00002510 T cudaMalloc

$ nm -D libnvshare.so

..00002510 T cudaMalloc
```

Symbol defined in .text

our custom version

Normally:

LOADER: matrixMul → cudaMalloc() ?? → DYNAMIC LINKER: libcudart.so → program uses original cudaMalloc

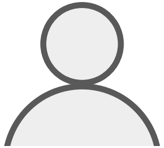
LD_PRELOAD=libnvshare.so

LOADER: matrixMul → cudaMalloc() ?? → DYNAMIC LINKER: **libnvshare.so** → program uses our version

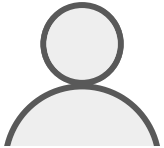
- which can **internally** call the original
- … if we will it

Key Idea

Key Idea

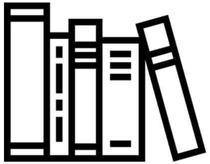


Key Idea



user

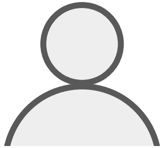
libnvshare.so



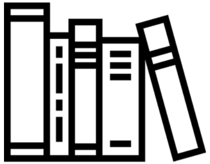
Key Idea

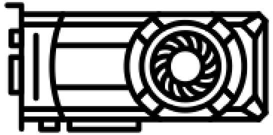
libnvshare.so

GPU

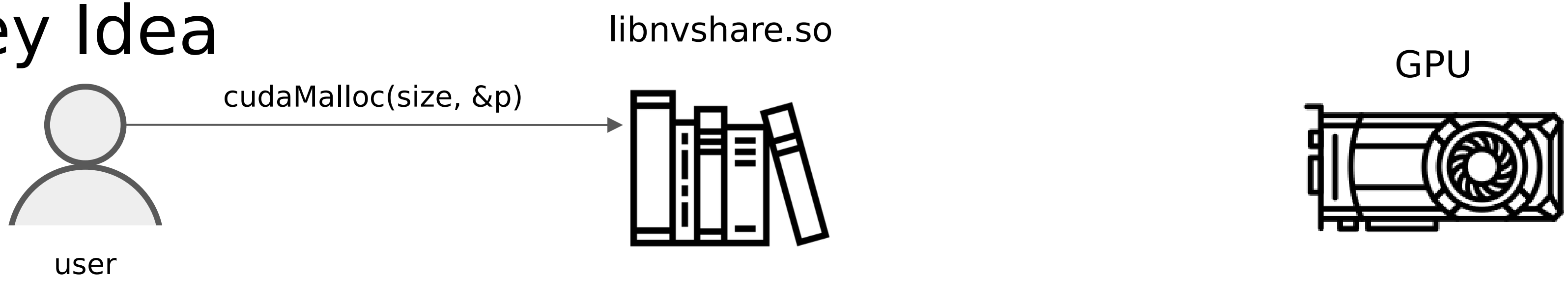


user

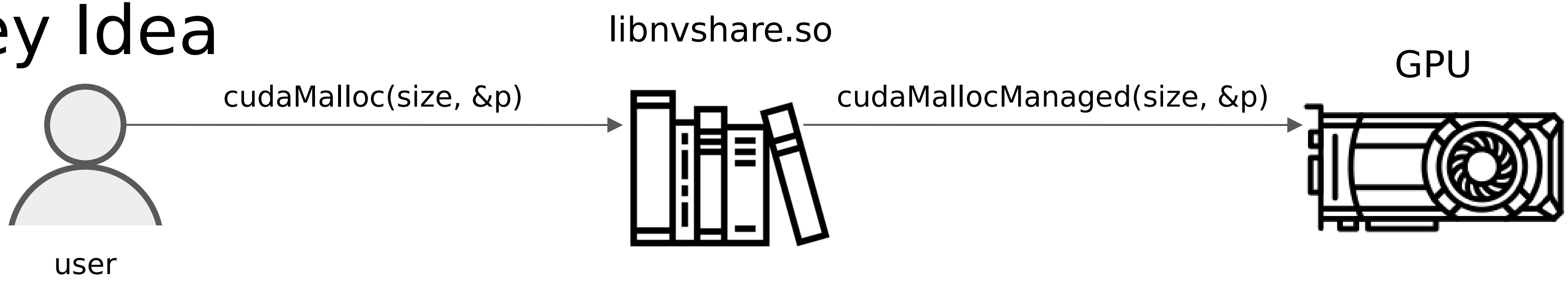




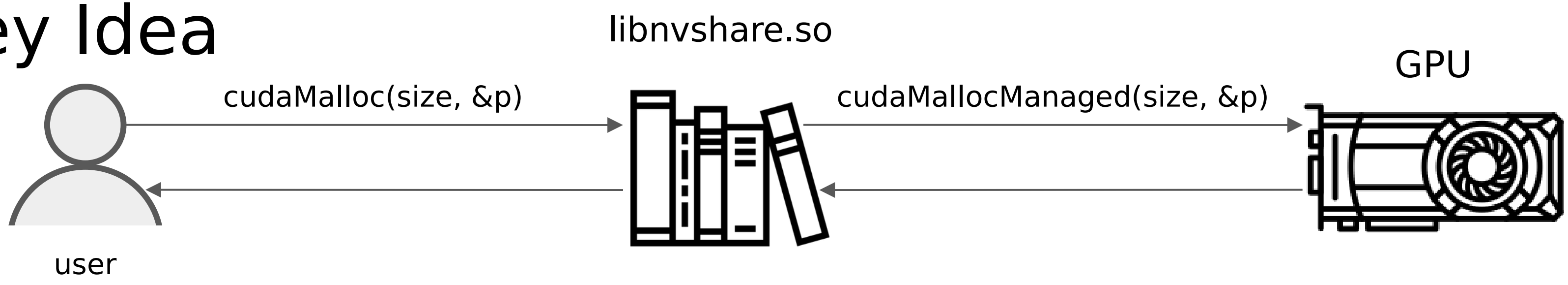
Key Idea



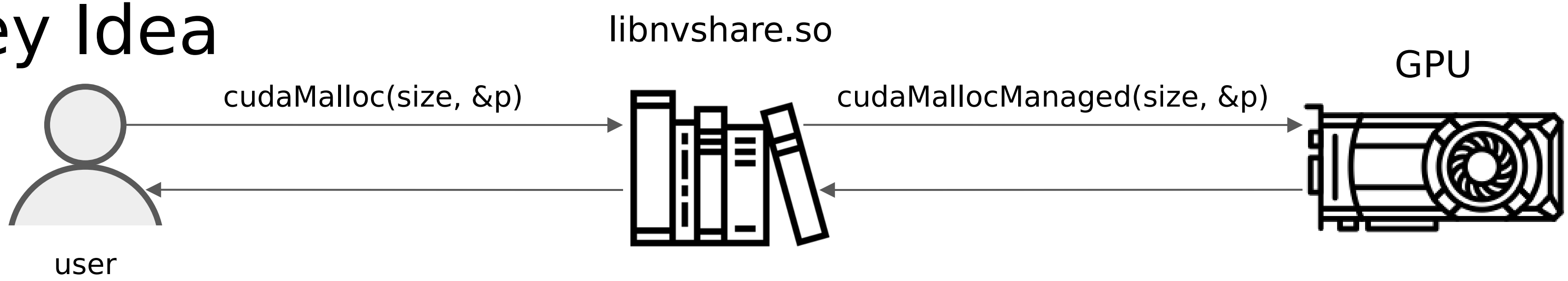
Key Idea


libnvshare.so
GPU
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
user

Key Idea

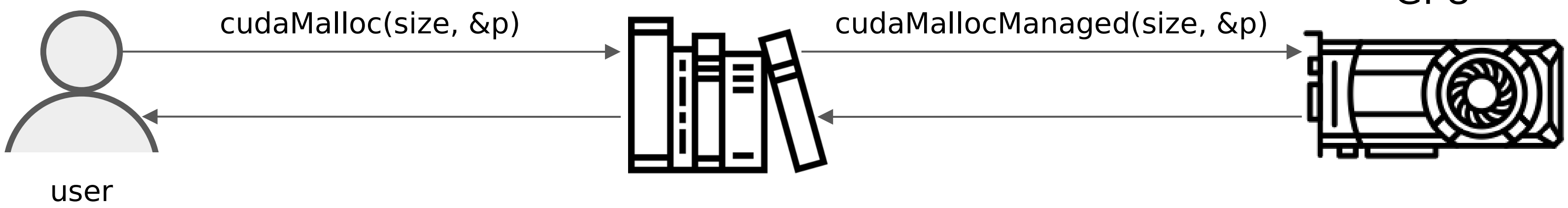

libnvshare.so
GPU
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
user

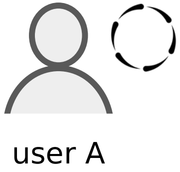
Key Idea



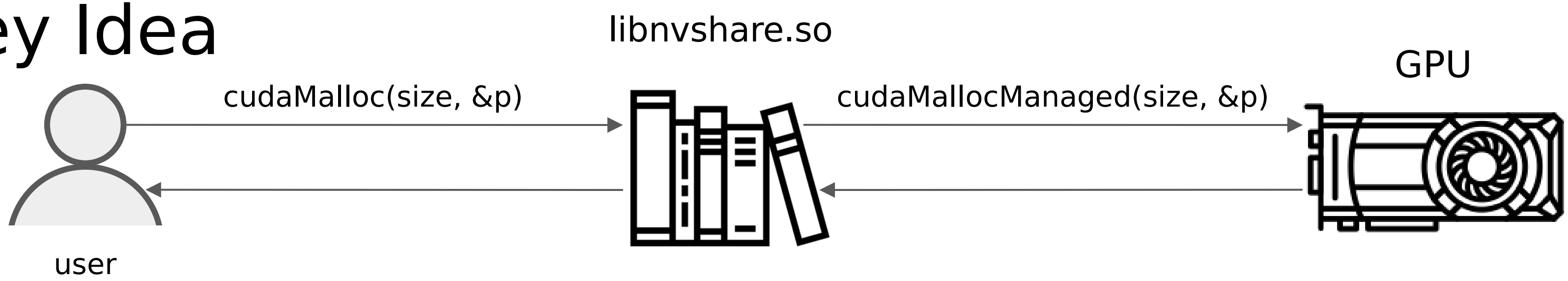
libnvshare.so
GPU
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
user

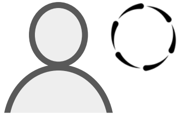
Key Idea


libnvshare.so
GPU
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
user


user A

Key Idea



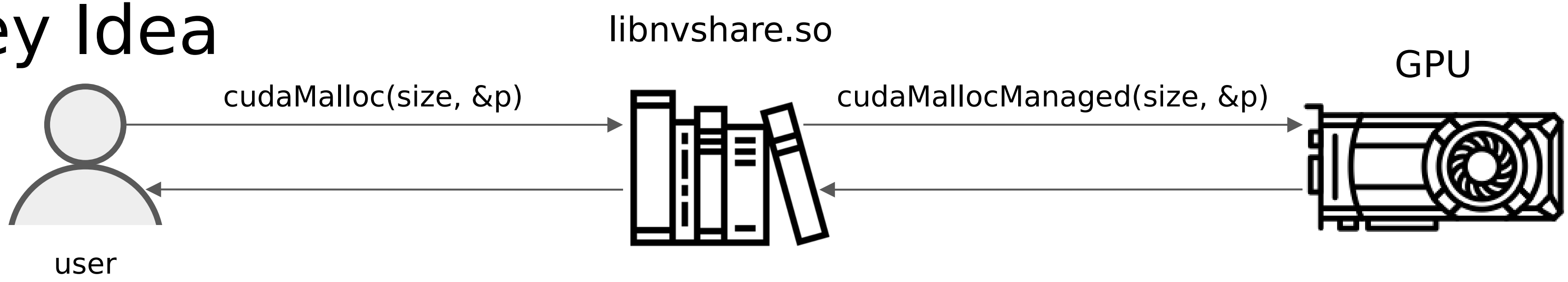


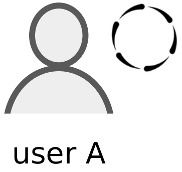
user A

GPU

A0
A1
A2
A3

Key Idea

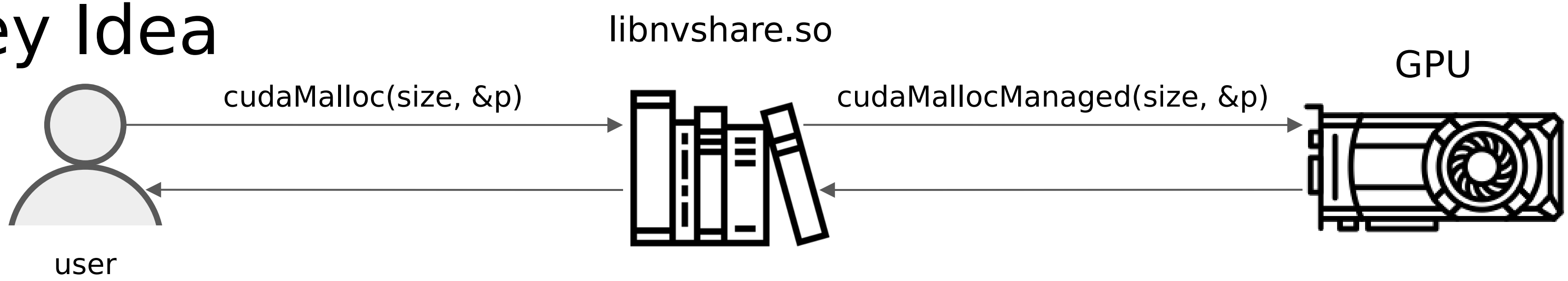


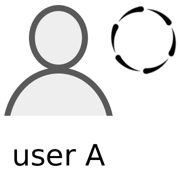


GPU

A0
A1
A2
A3

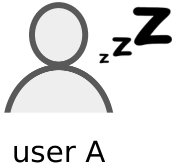
Key Idea



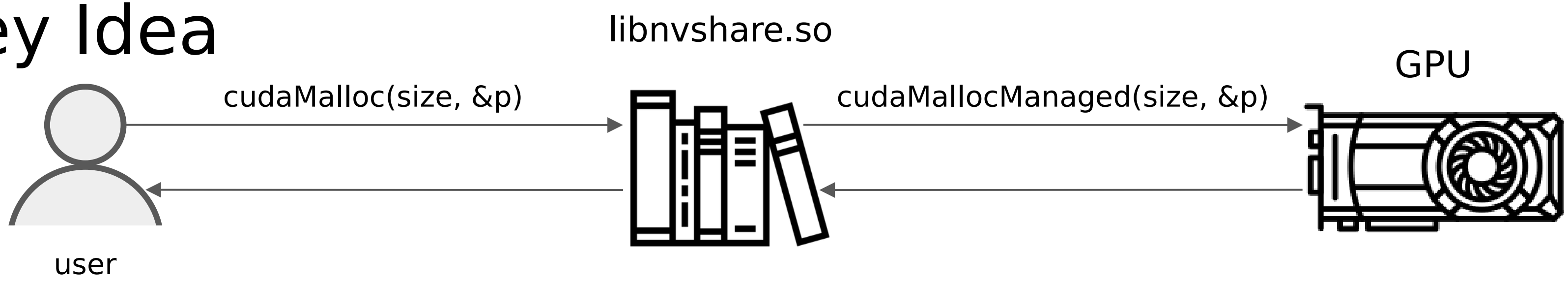


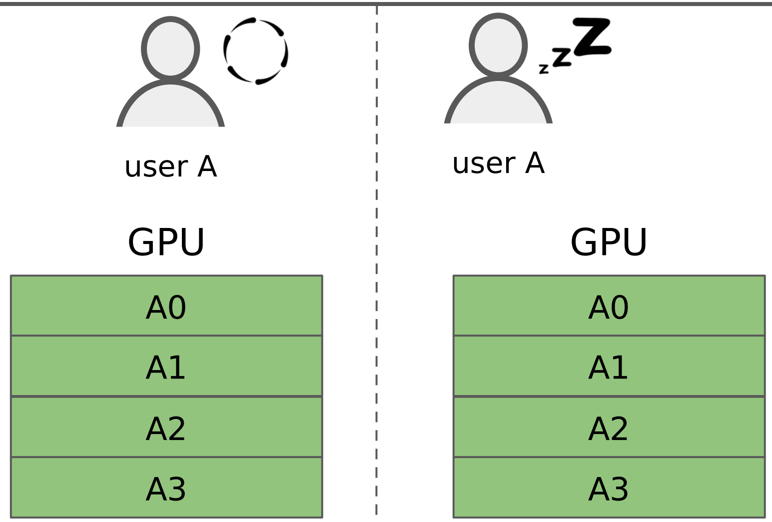
GPU

A0
A1
A2
A3

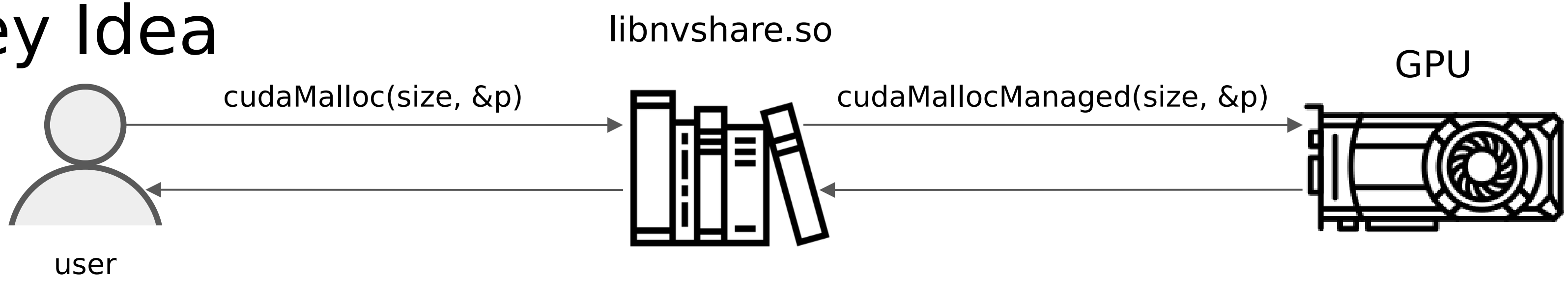


Key Idea

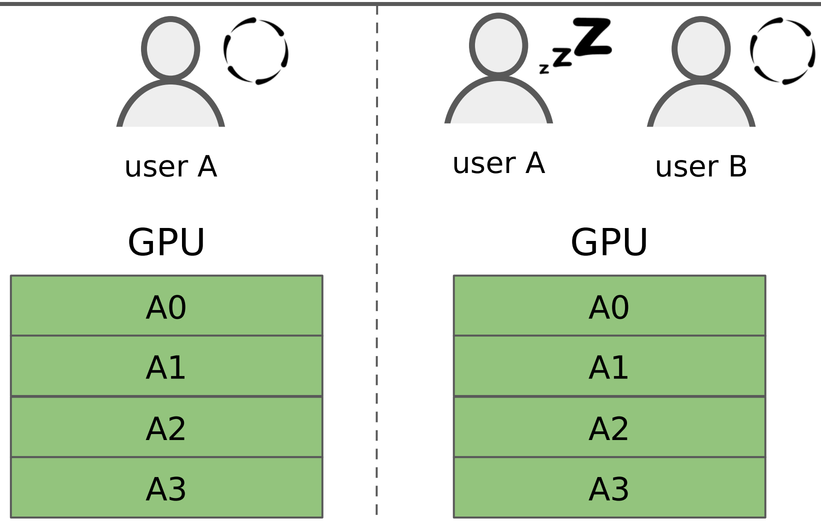

libnvshare.so
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
GPU
user


user A
GPU
A0
A1
A2
A3
user A
GPU
A0
A1
A2
A3

Key Idea

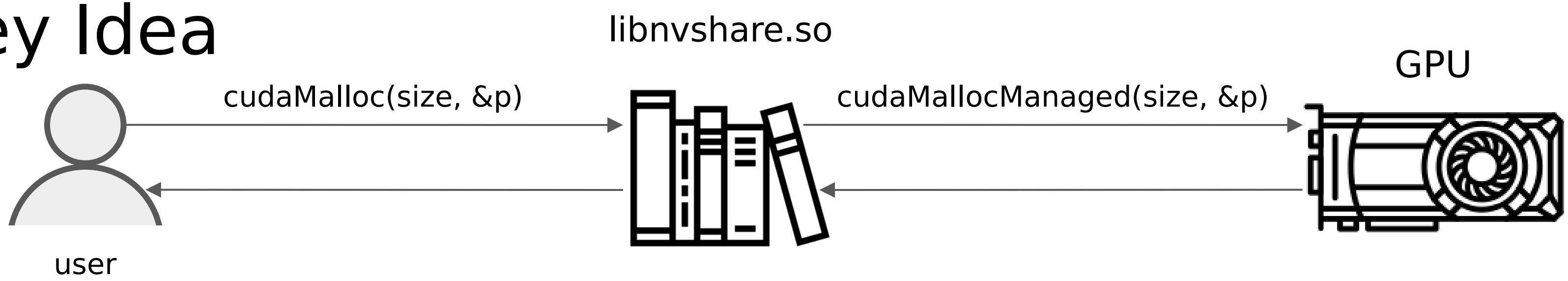


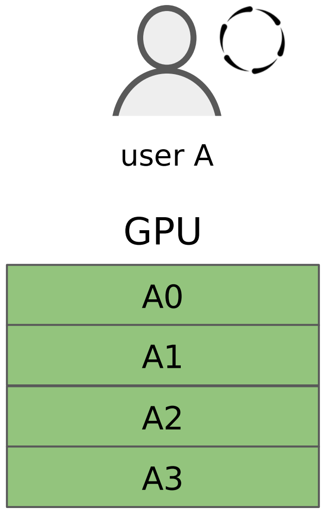
libnvshare.so
GPU
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
user

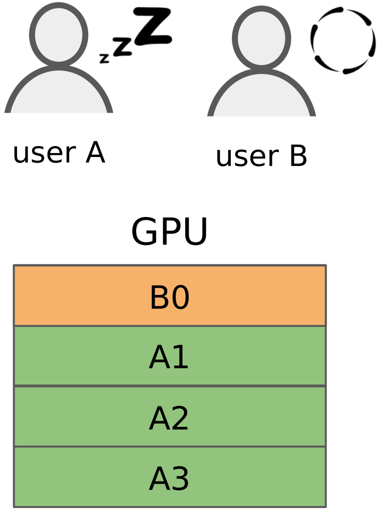


user A
GPU
A0
A1
A2
A3
user A
user B
GPU
A0
A1
A2
A3

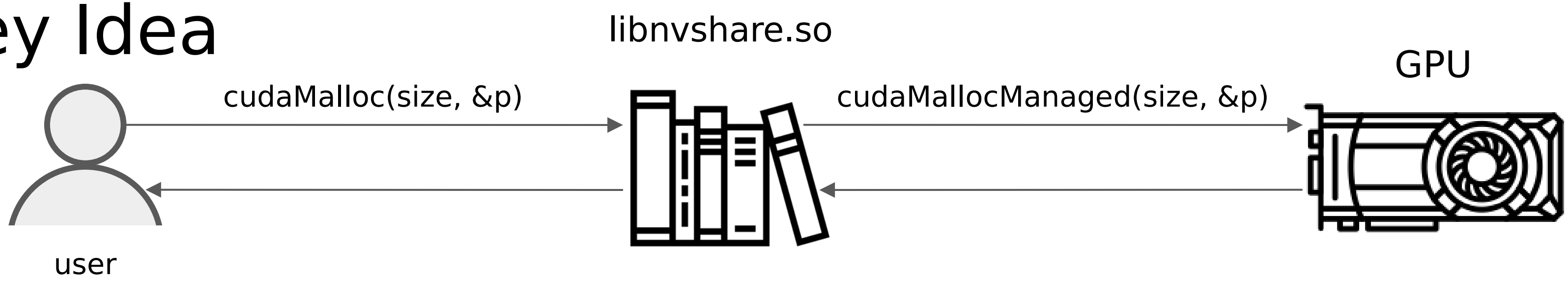
Key Idea


libnvshare.so
GPU
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
user

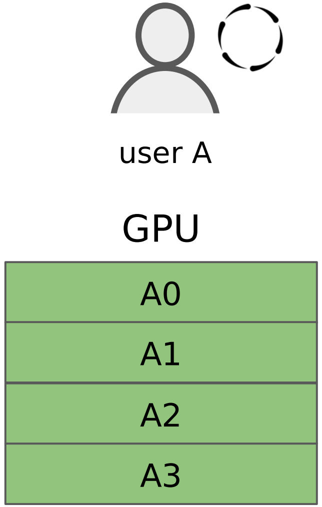

user A
GPU
A0
A1
A2
A3


user A
user B
GPU
B0
A1
A2
A3

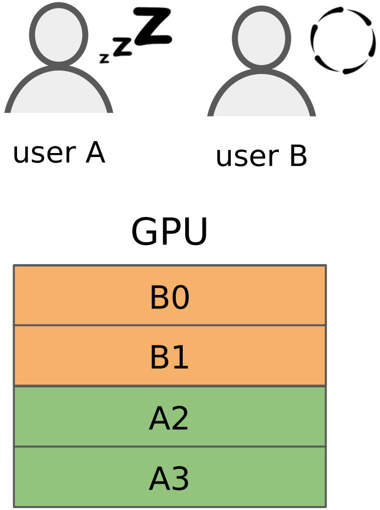
Key Idea



libnvshare.so
GPU
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
user

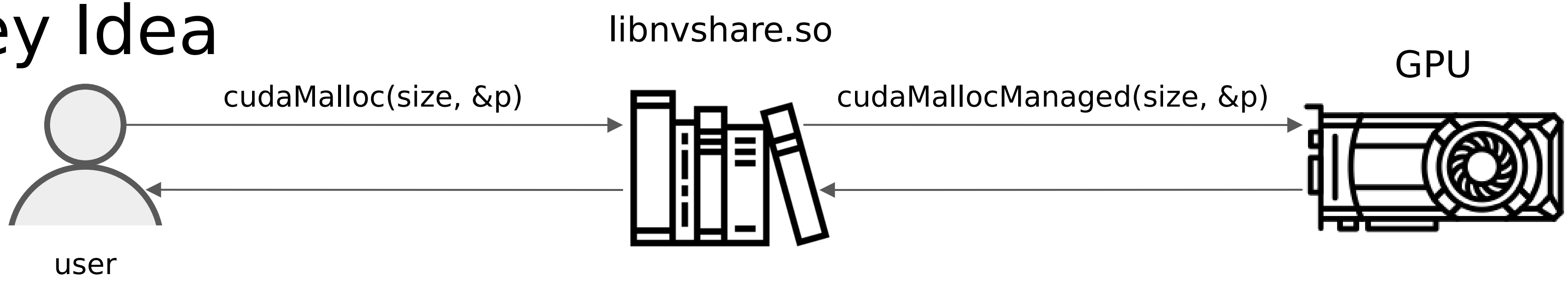


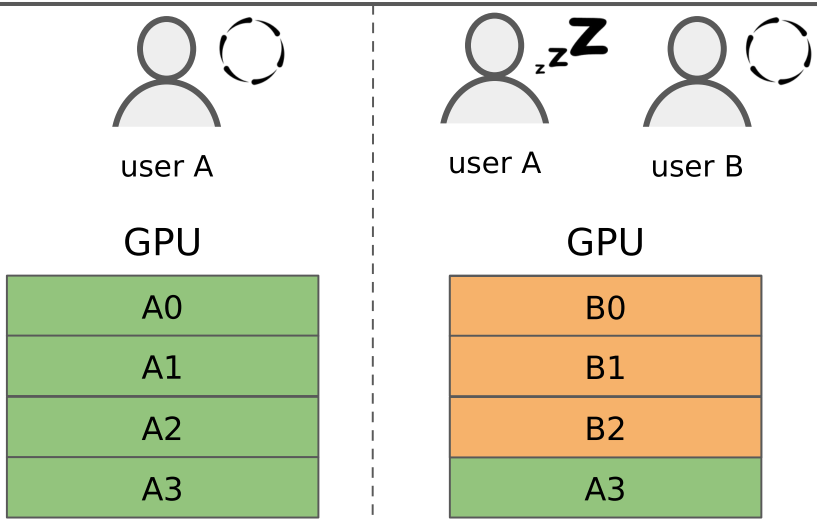
user A
GPU
A0
A1
A2
A3



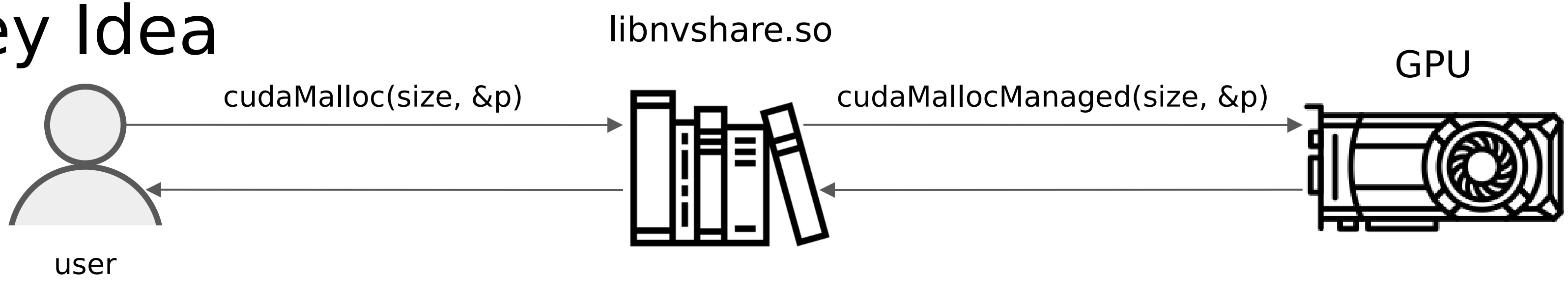
user A
user B
GPU
B0
B1
A2
A3

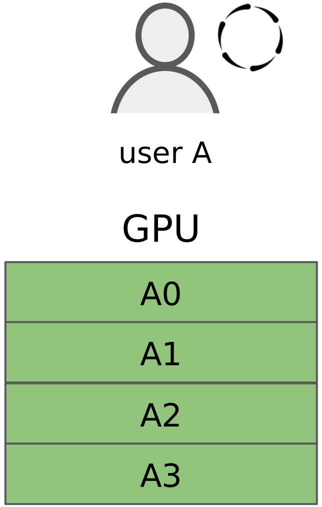
Key Idea

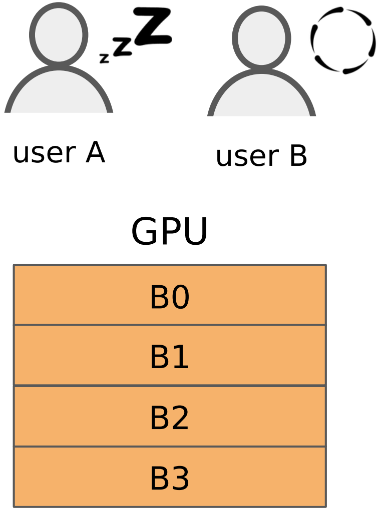

libnvshare.so
GPU
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
user


user A
GPU
A0
A1
A2
A3
user A
user B
GPU
B0
B1
B2
A3

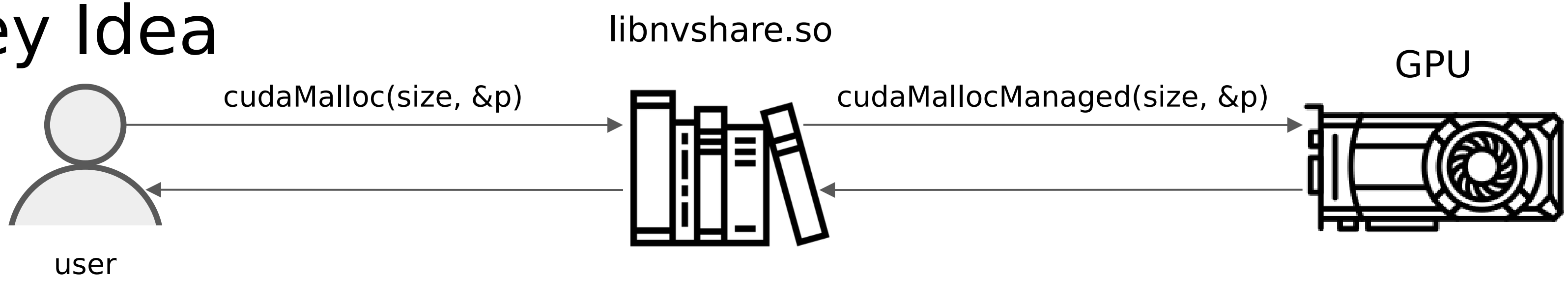
Key Idea

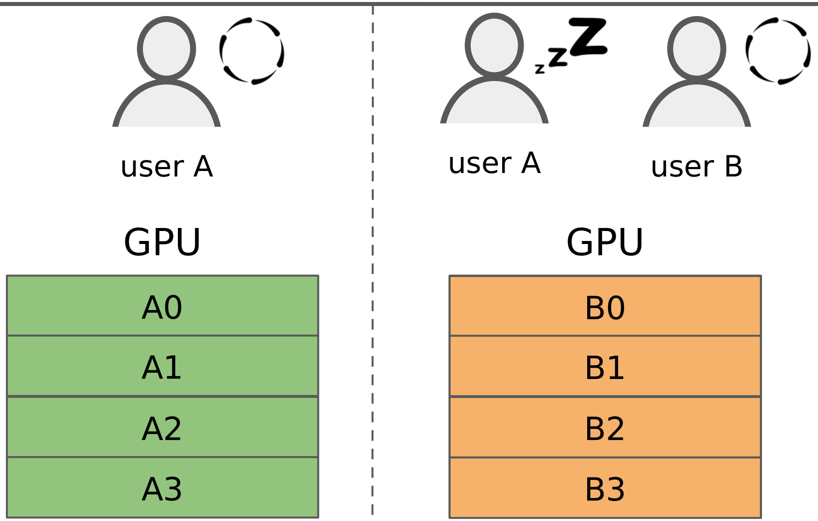

libnvshare.so
GPU
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
user


user A
GPU
A0
A1
A2
A3

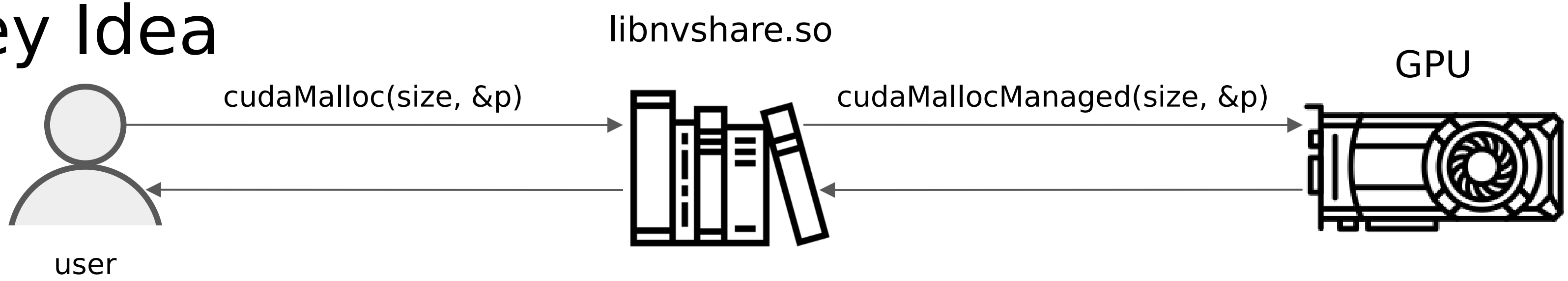

user A
user B
GPU
B0
B1
B2
B3

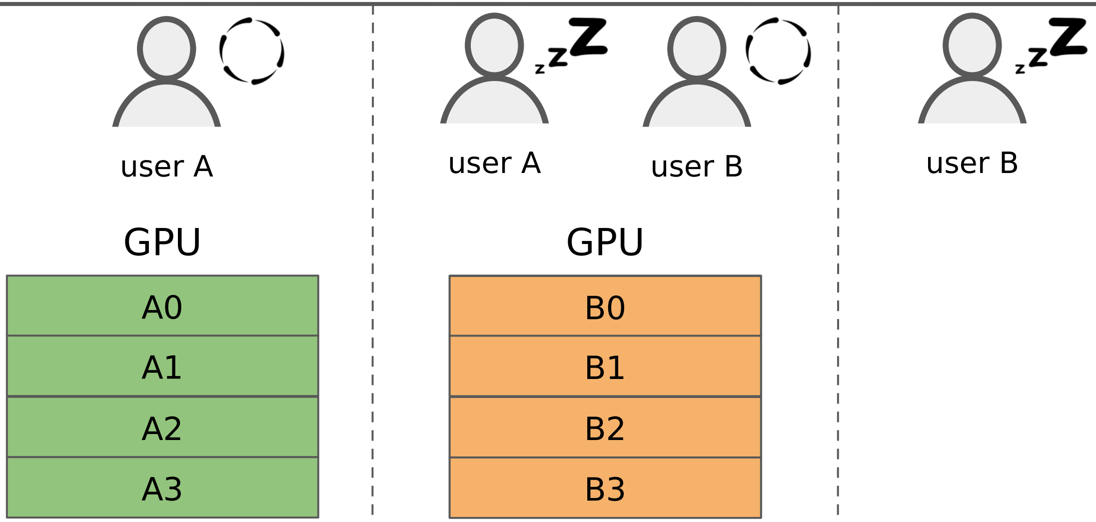
Key Idea


libnvshare.so
GPU
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
user

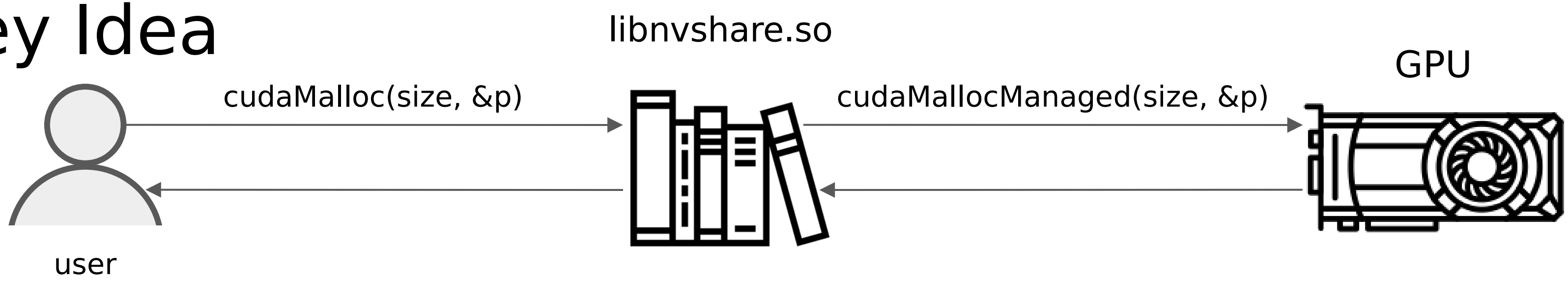

user A
GPU
A0
A1
A2
A3
user A
user B
GPU
B0
B1
B2
B3

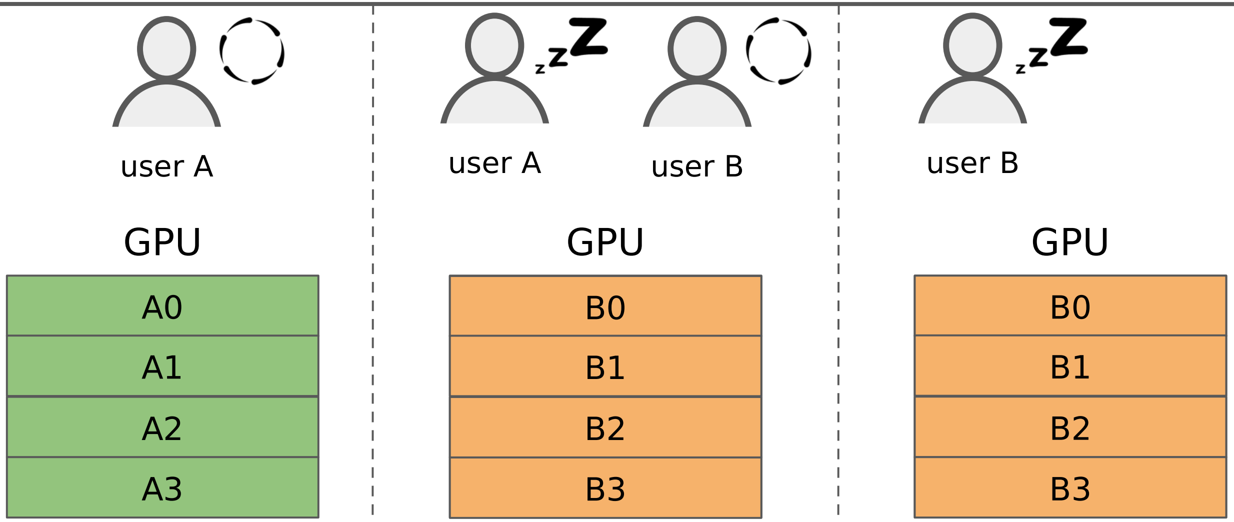
Key Idea


libnvshare.so
GPU
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
user

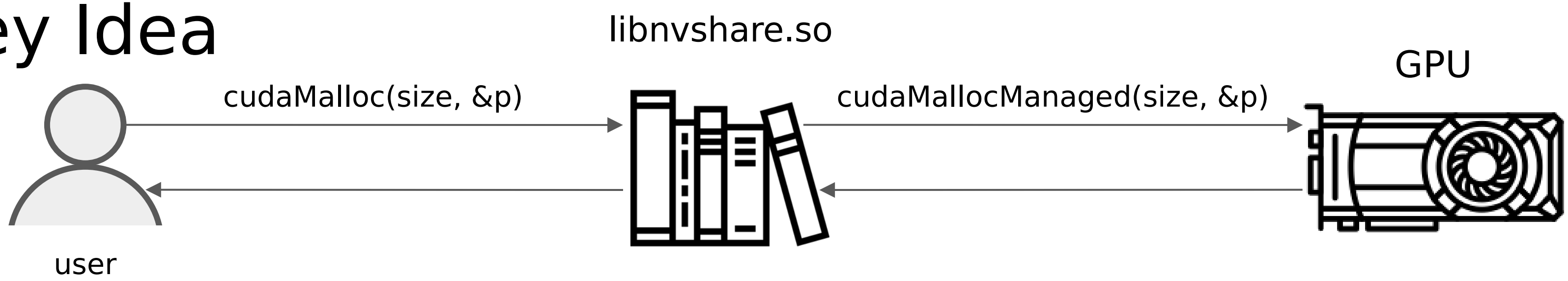

user A
user A
user B
user B
GPU
A0
A1
A2
A3
GPU
B0
B1
B2
B3

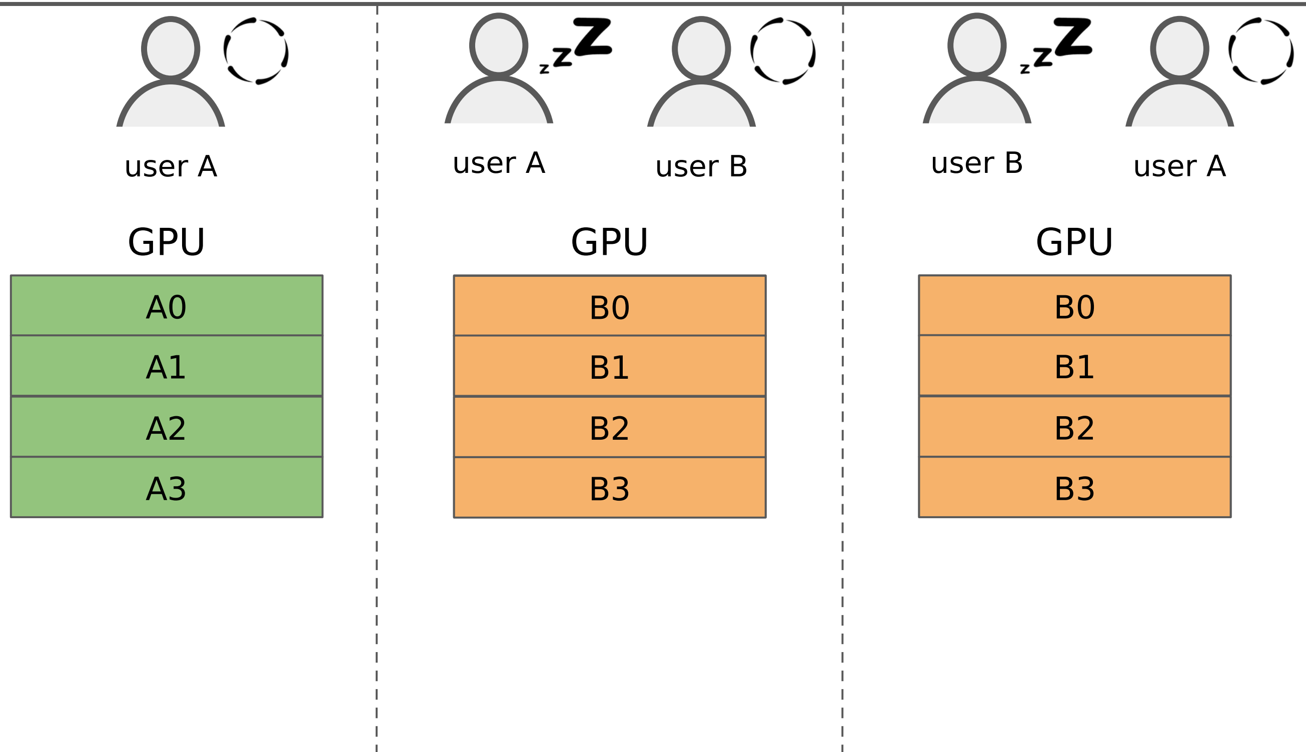
Key Idea


libnvshare.so
GPU
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
user

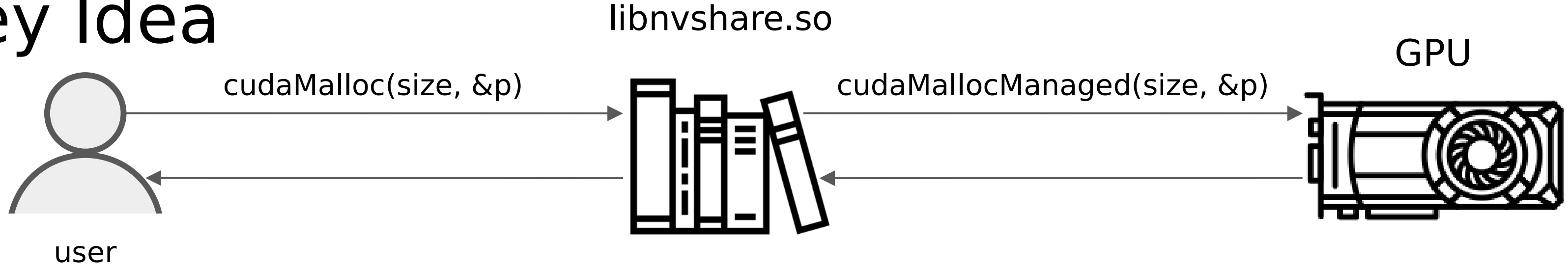

user A
user A
user B
user B
GPU
A0
A1
A2
A3
GPU
B0
B1
B2
B3
GPU
B0
B1
B2
B3

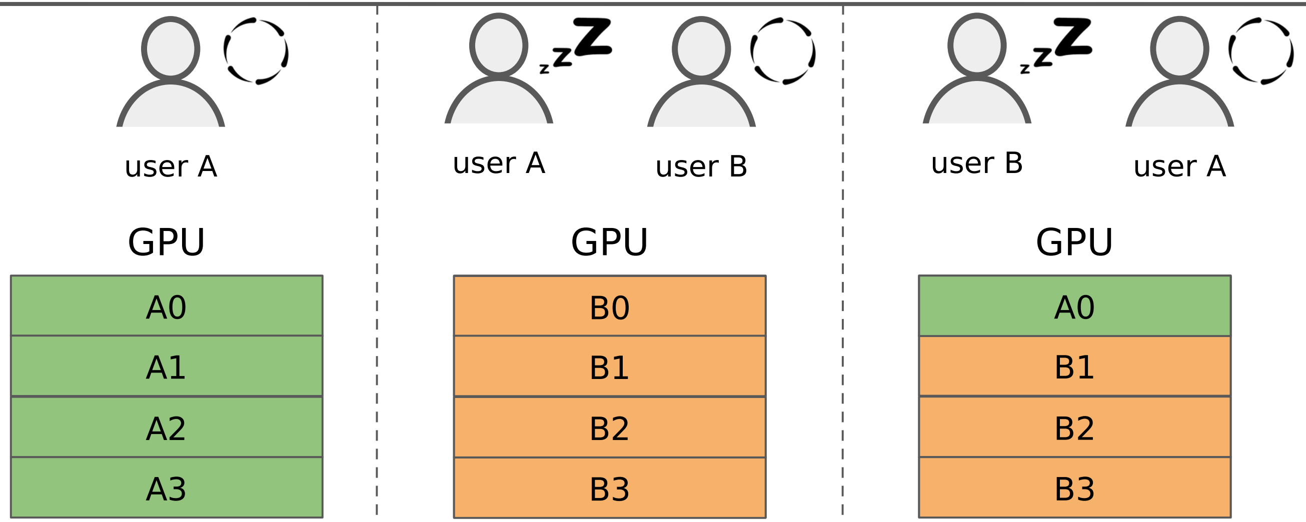
Key Idea


libnvshare.so
GPU
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
user

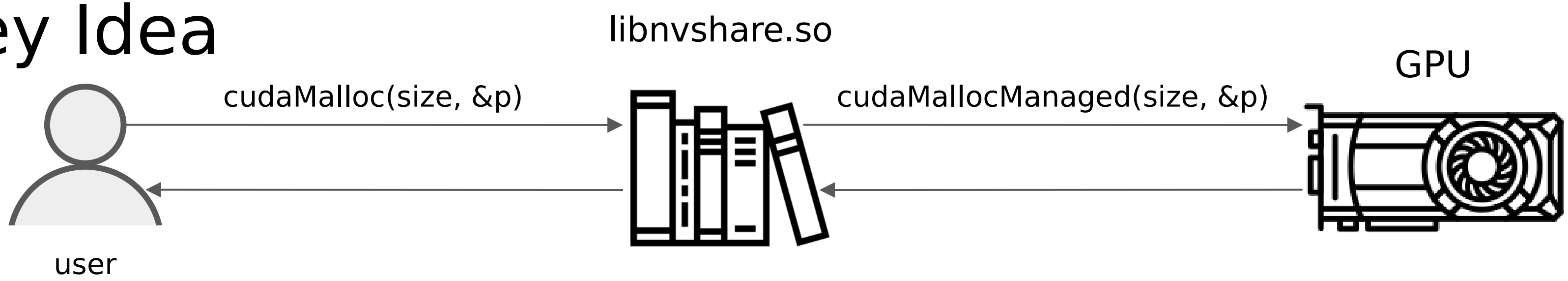

user A
GPU
A0
A1
A2
A3
user A
user B
GPU
B0
B1
B2
B3
user B
user A
GPU
B0
B1
B2
B3

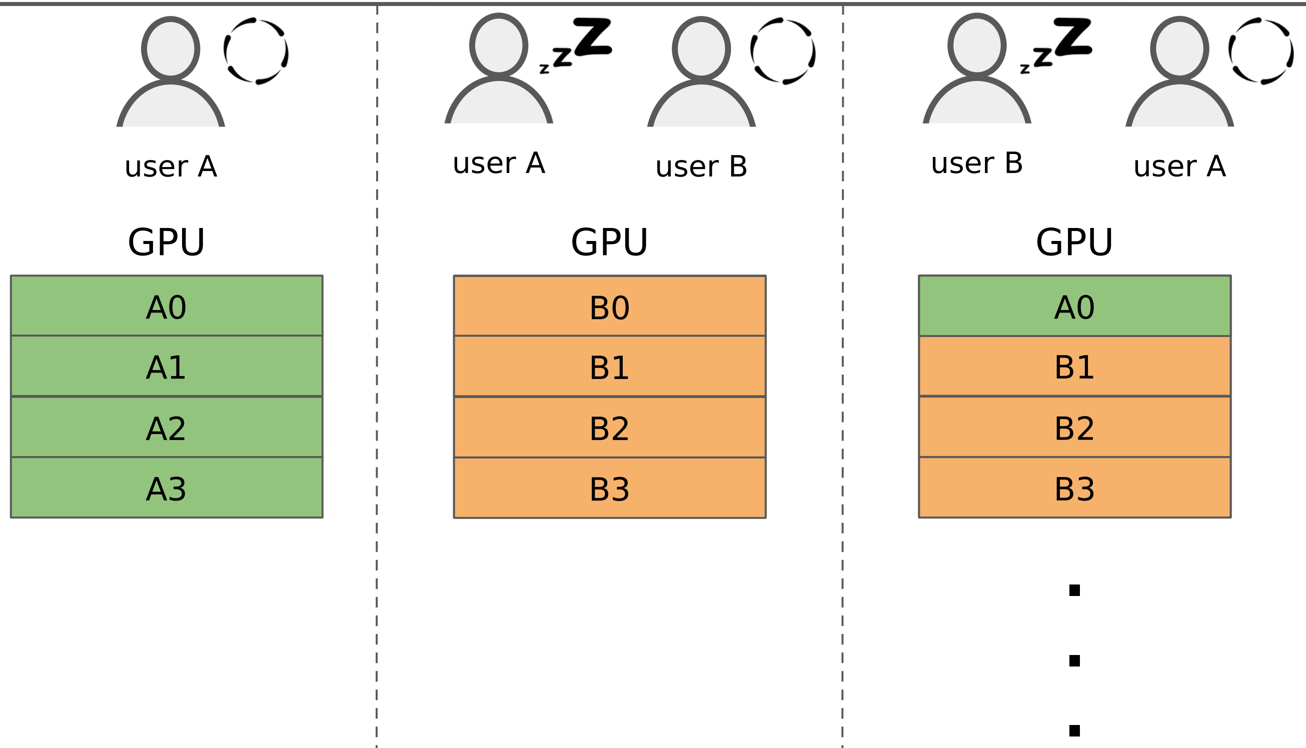
Key Idea


libnvshare.so
GPU
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
user

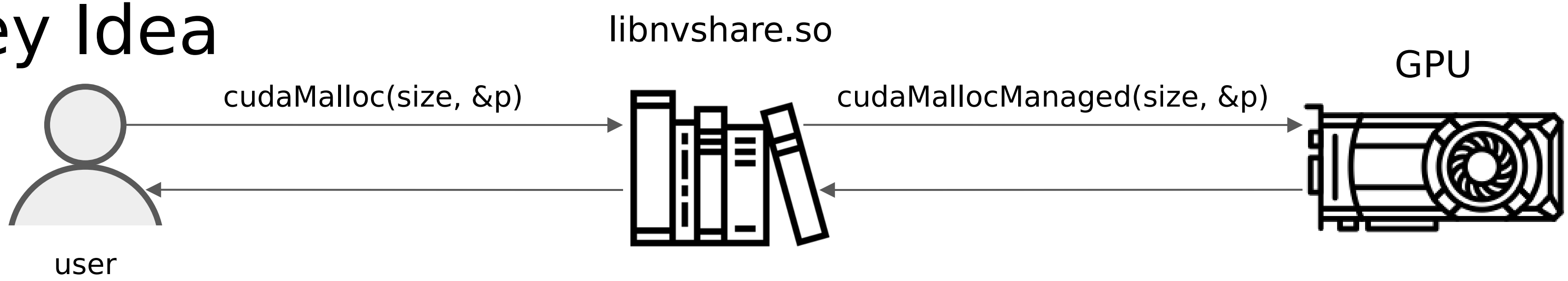

user A
GPU
A0
A1
A2
A3
user A
user B
GPU
B0
B1
B2
B3
user B
user A
GPU
A0
B1
B2
B3

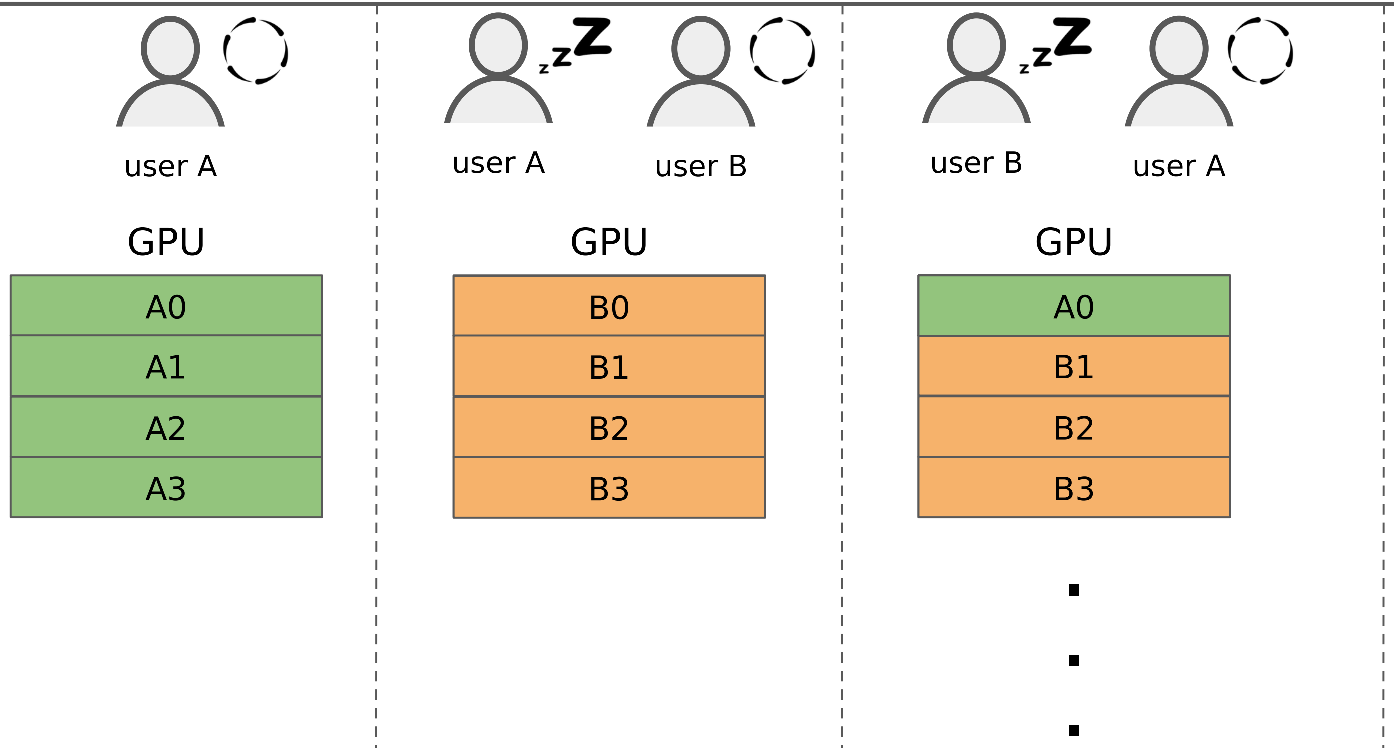
Key Idea


libnvshare.so
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
GPU
user

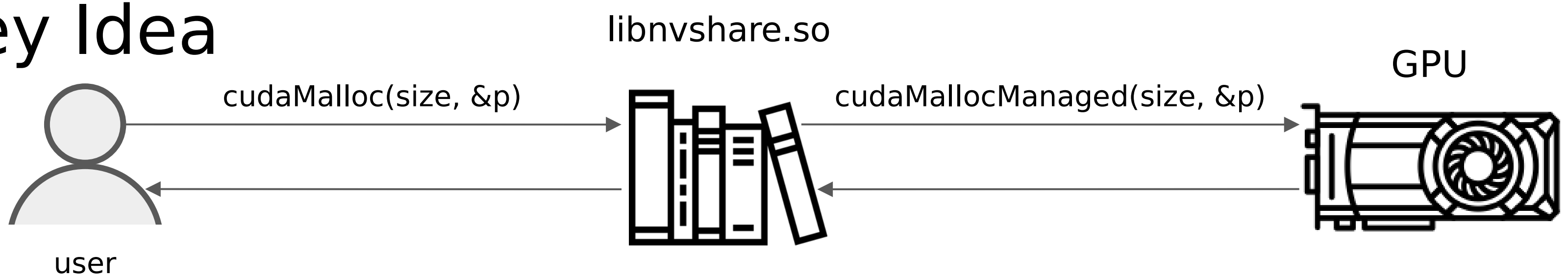

user A
GPU
A0
A1
A2
A3
user A
user B
GPU
B0
B1
B2
B3
user B
user A
GPU
A0
B1
B2
B3

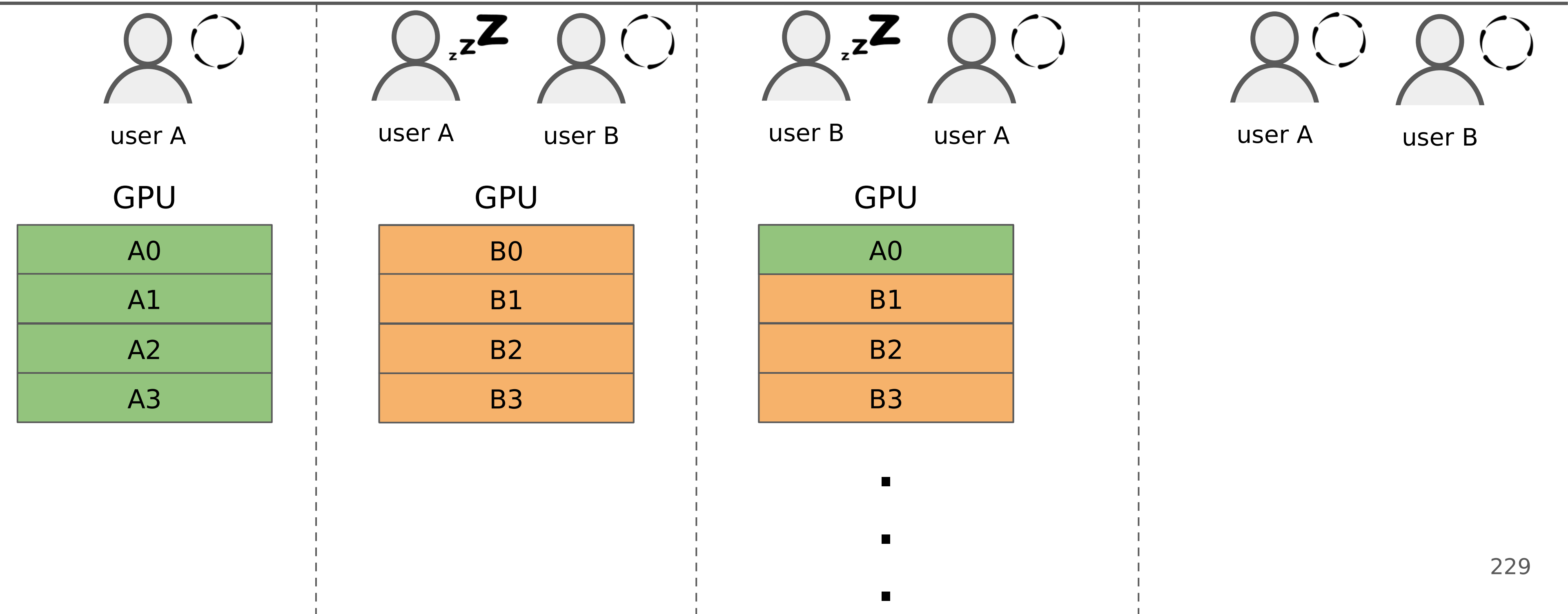
Key Idea


libnvshare.so
GPU
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
user

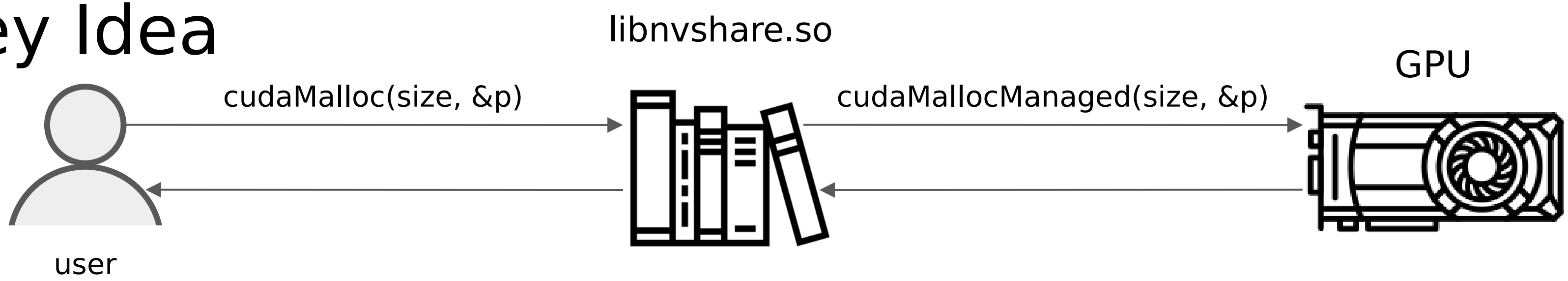

user A
user A
user B
user B
user A
GPU
A0
A1
A2
A3
GPU
B0
B1
B2
B3
GPU
A0
B1
B2
B3

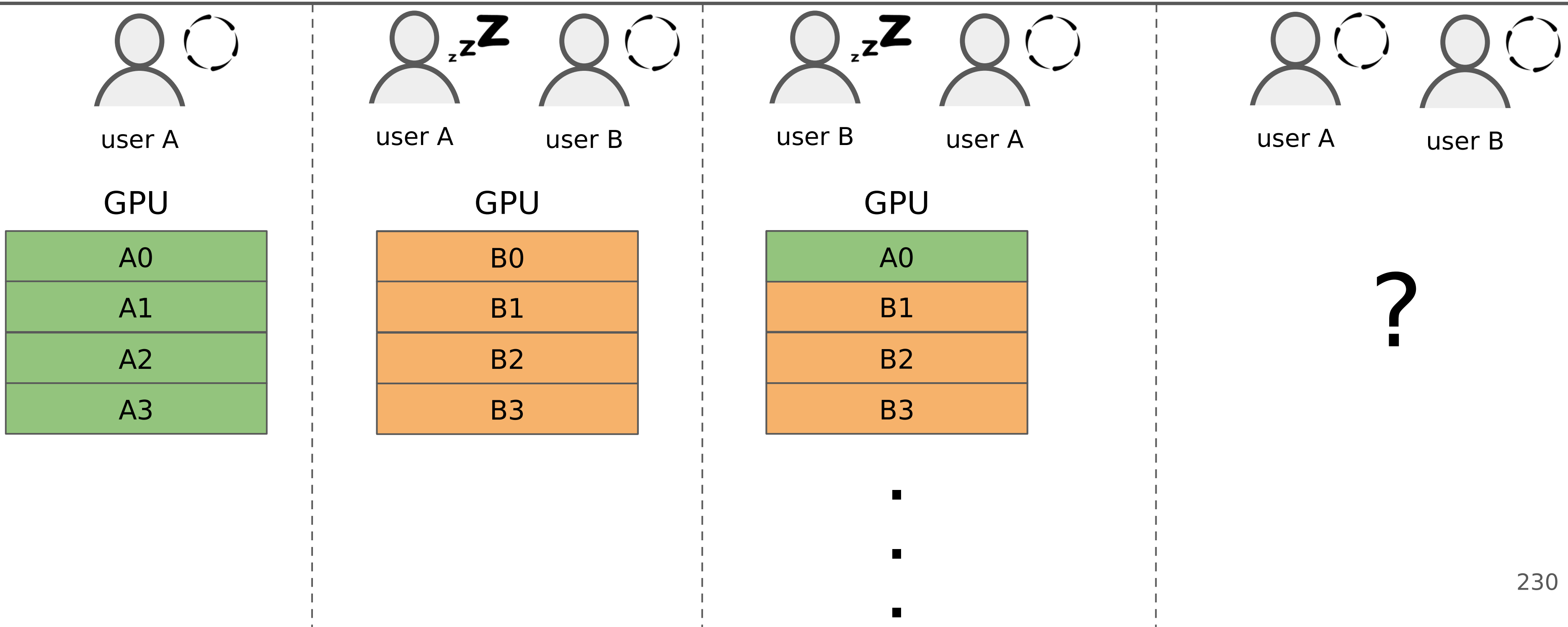
Key Idea


libnvshare.so
GPU
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
user


user A
user A
user B
user B
user A
user A
user B
GPU
A0
A1
A2
A3
GPU
B0
B1
B2
B3
GPU
A0
B1
B2
B3

Key Idea


libnvshare.so
GPU
cudaMalloc(size, &p)
cudaMallocManaged(size, &p)
user


user A
user A
user B
user B
user A
user A
user B
GPU
A0
A1
A2
A3
GPU
B0
B1
B2
B3
GPU
A0
B1
B2
B3
?

When GPU bursts overlap

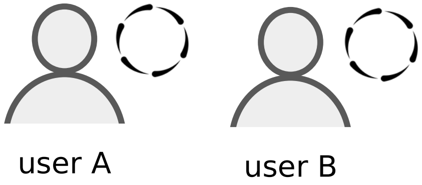
When GPU bursts overlap

When GPU bursts overlap

Sometimes

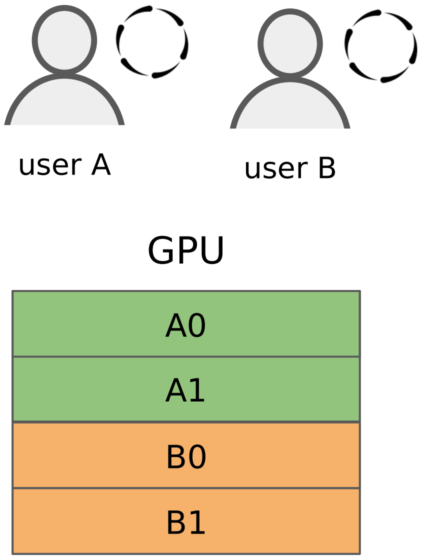
When GPU bursts overlap

Sometimes



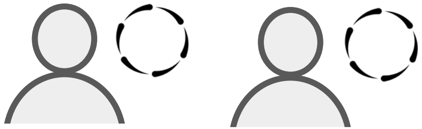
When GPU bursts overlap

Sometimes



When GPU bursts overlap

Sometimes



user A user B

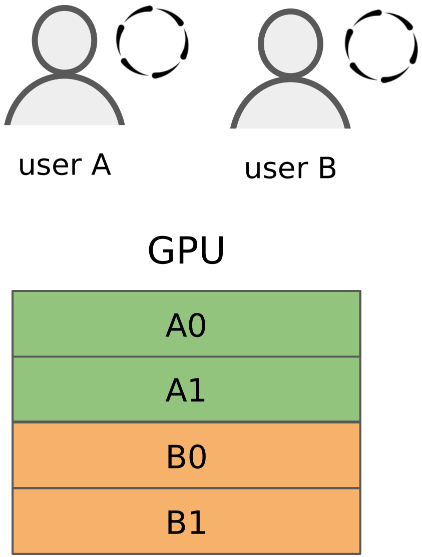
GPU

A0
A1
B0
B1

$\Sigma Wss_i <$ GPU memory

When GPU bursts overlap

Sometimes

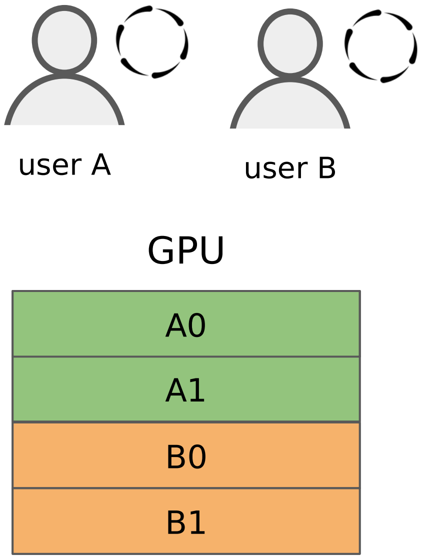


$\Sigma wss_i <$ GPU memory

No page faults

When GPU bursts overlap

Sometimes

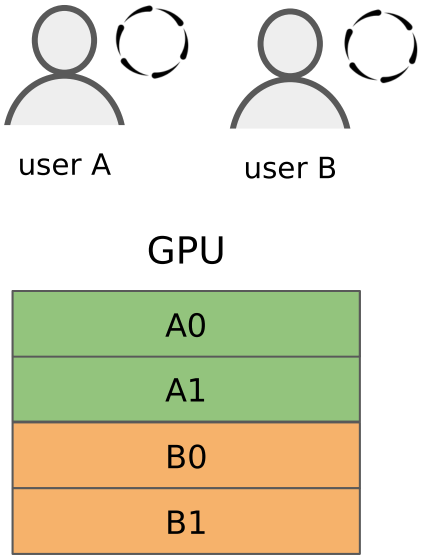


$\Sigma Wss_i <$ GPU memory

No page faults

When GPU bursts overlap

Sometimes | Other times

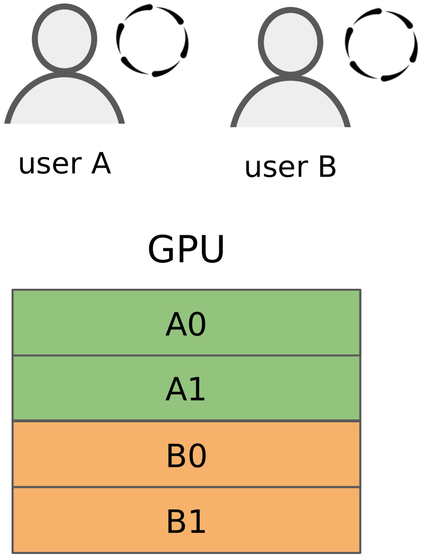


$\Sigma Wss_i <$ GPU memory

No page faults

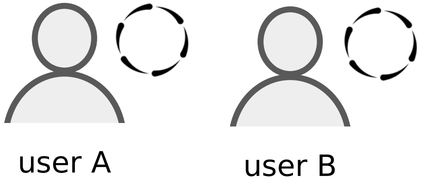
When GPU bursts overlap

Sometimes | **Other times**



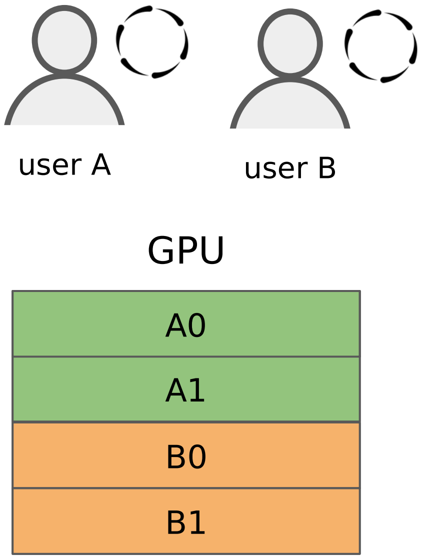
$\Sigma Wss_i <$ GPU memory

No page faults



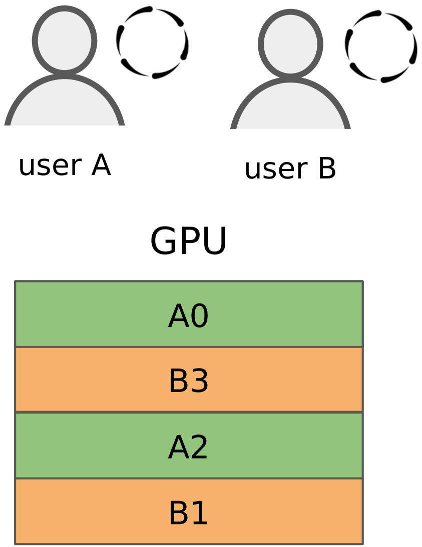
When GPU bursts overlap

Sometimes	Other times



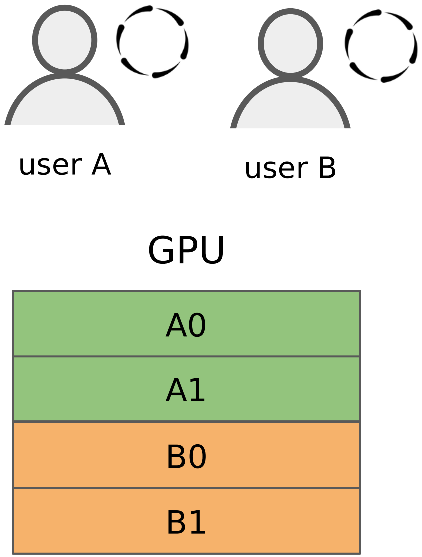
$\Sigma wss_i <$ GPU memory

No page faults



When GPU bursts overlap

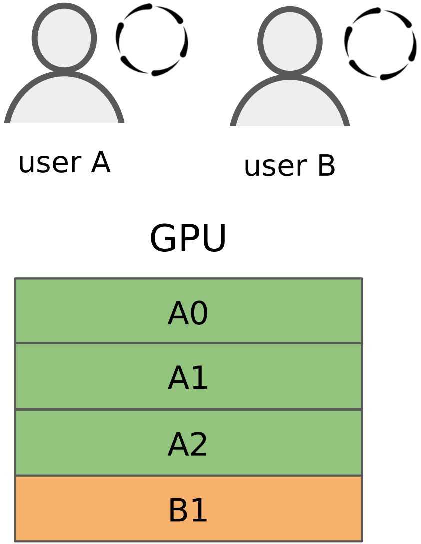
Sometimes



$\Sigma \text{Wss}_i <$ GPU memory

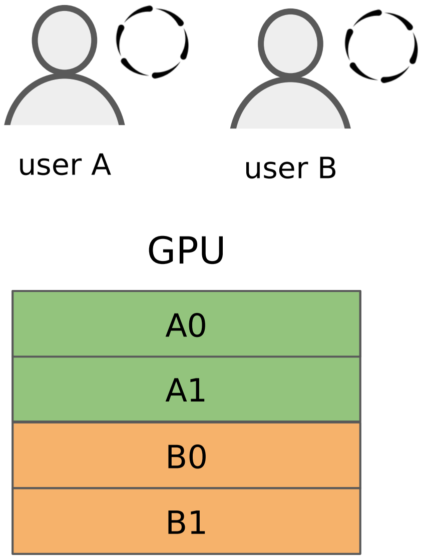
No page faults

Other times



When GPU bursts overlap

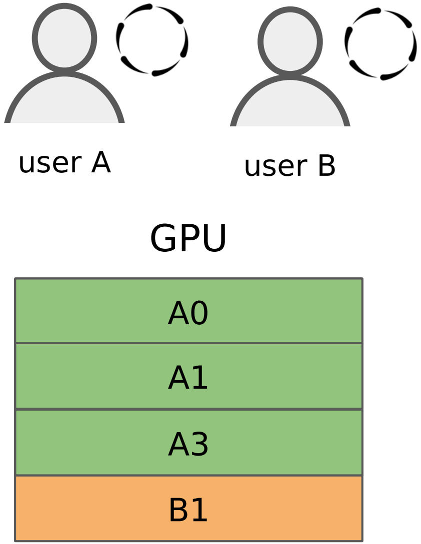
Sometimes



$\Sigma Wss_i <$ GPU memory

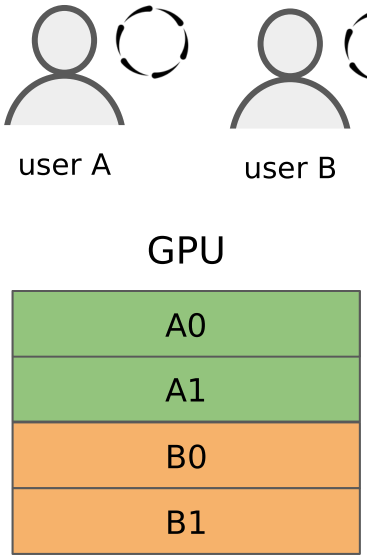
No page faults

Other times



When GPU bursts overlap

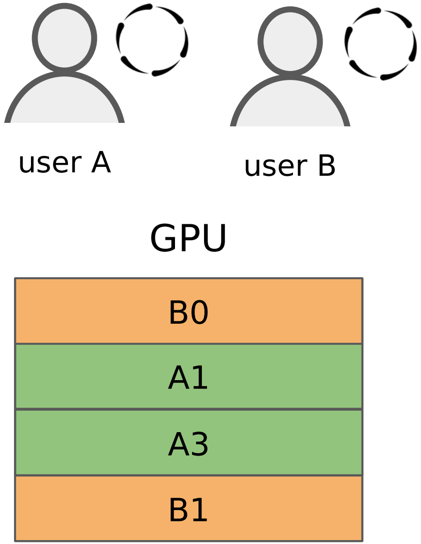
Sometimes



$\Sigma Wss_i < $ GPU memory

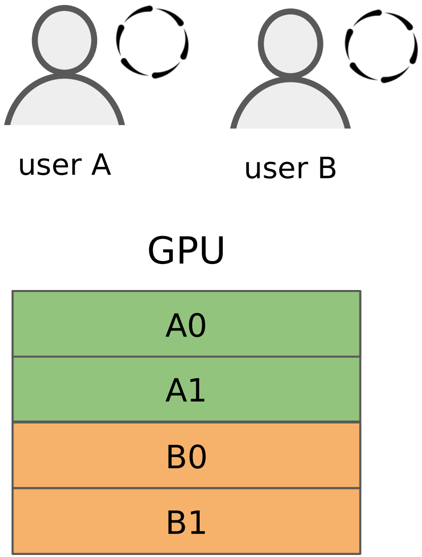
No page faults

Other times



When GPU bursts overlap

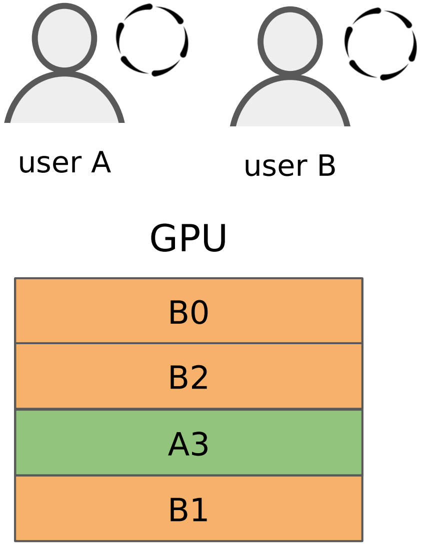
Sometimes



$\Sigma Wss_i <$ GPU memory

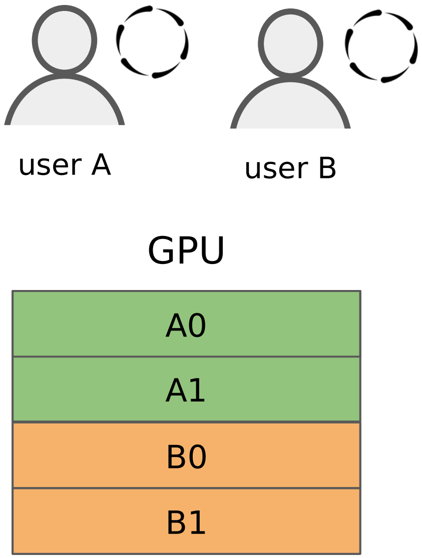
No page faults

Other times



When GPU bursts overlap

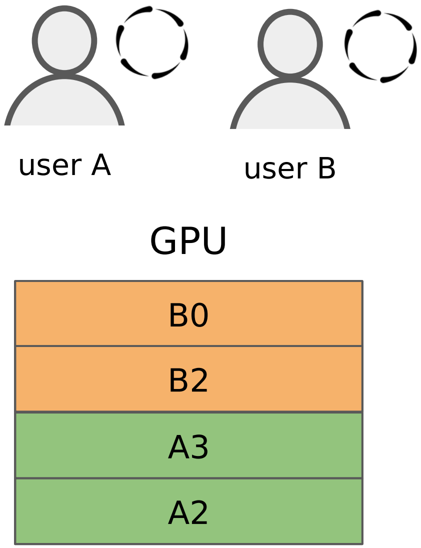
Sometimes



$\Sigma \text{Wss}_i <$ GPU memory

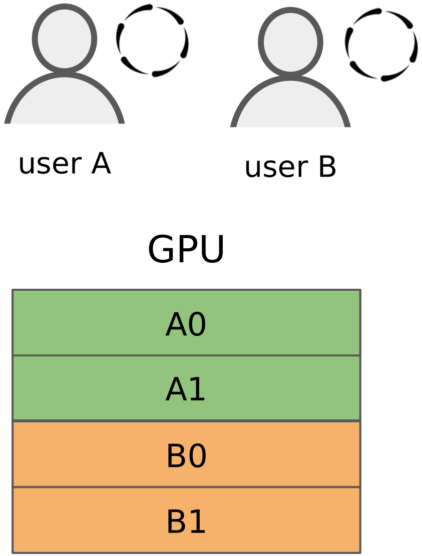
No page faults

Other times



When GPU bursts overlap

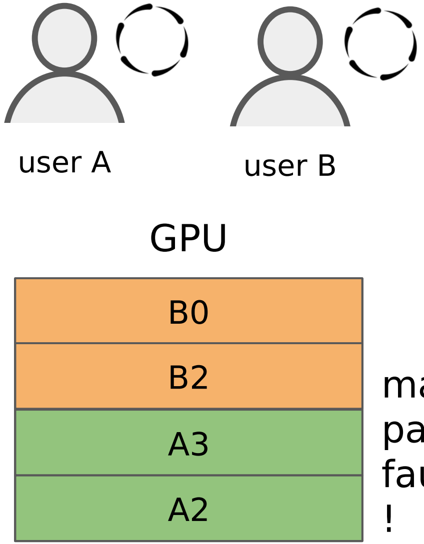
Sometimes



$\Sigma Wss_i <$ GPU memory

No page faults

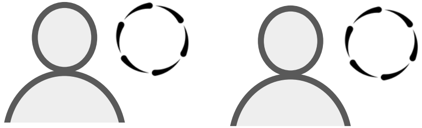
Other times



many page faults !

When GPU bursts overlap

Sometimes	Other times



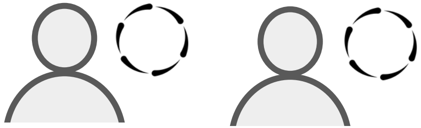
user A user B

GPU

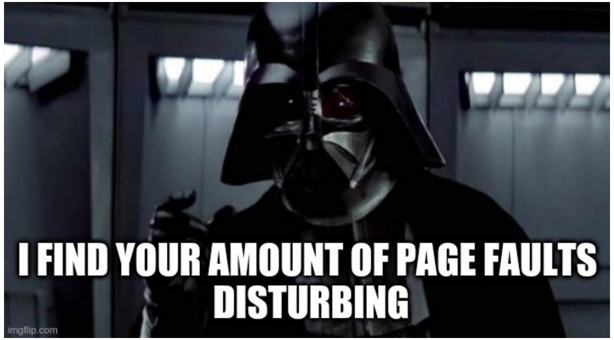
A0
A1
B0
B1

$\Sigma Wss_i <$ GPU memory

No page faults

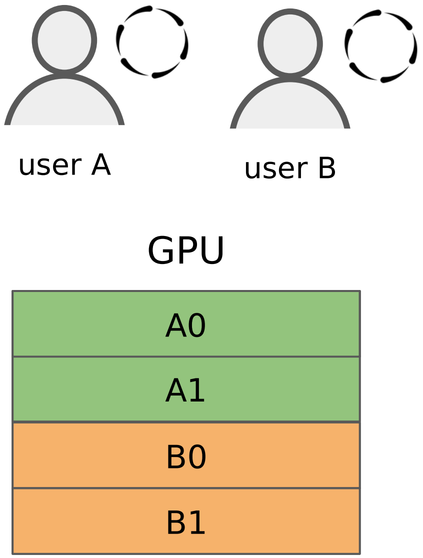


user A user B



When GPU bursts overlap

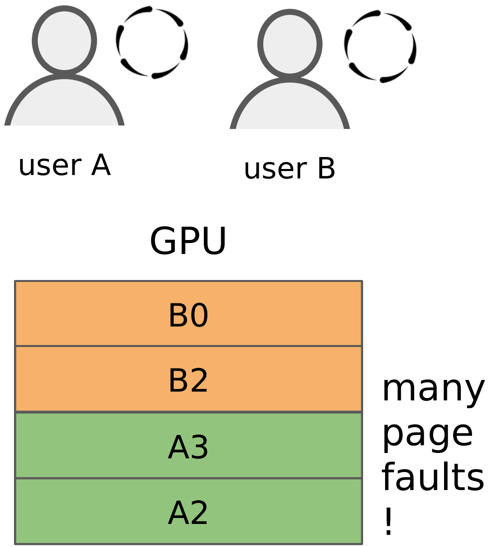
Sometimes



$\Sigma Wss_i <$ GPU memory

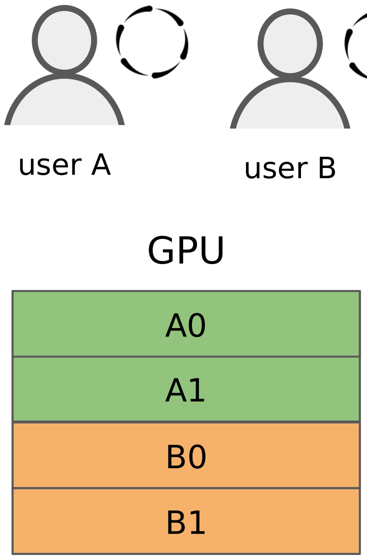
No page faults

Other times



When GPU bursts overlap

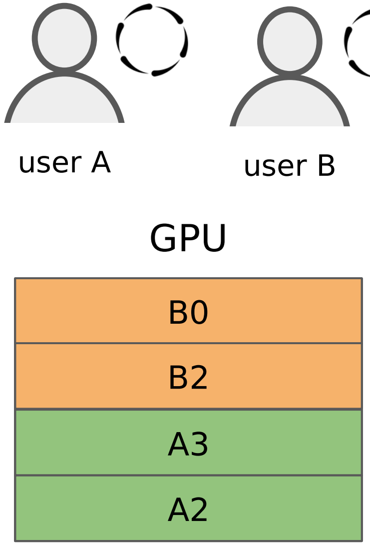
Sometimes



$\Sigma Wss_i <$ GPU memory

No page faults

Other times

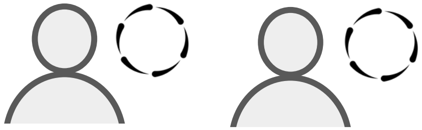


many page faults !

$\Sigma Wss_i >$ GPU memory

When GPU bursts overlap

Sometimes



user A

user B

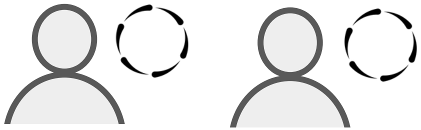
GPU

A0
A1
B0
B1

$\Sigma Wss_i <$ GPU memory

No page faults

Other times



user A

user B

GPU

B0
B2
A3
A2

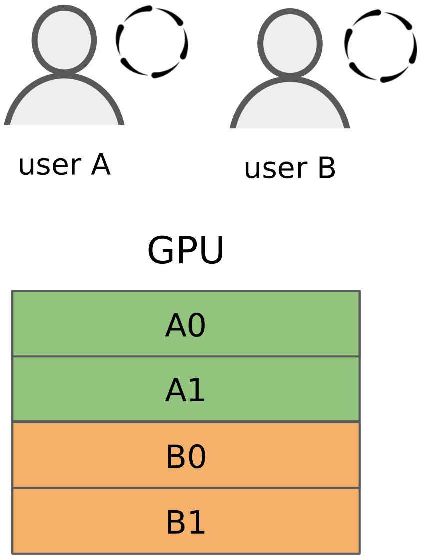
many page faults !

$\Sigma Wss_i >$ GPU memory

Thrashing!

When GPU bursts overlap

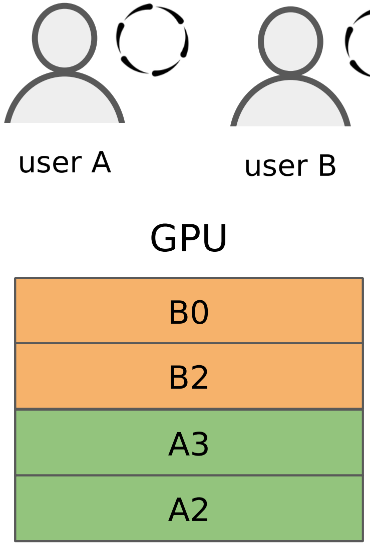
Sometimes



$\Sigma Wss_i <$ GPU memory

No page faults

Other times



many page faults !

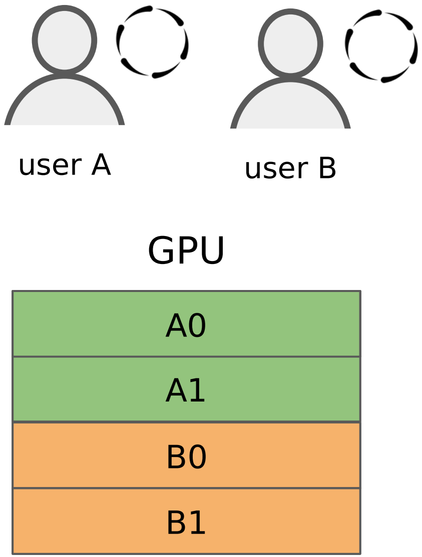
$\Sigma Wss_i >$ GPU memory

Thrashing!

When GPU bursts overlap

Sometimes	Other times	Solution

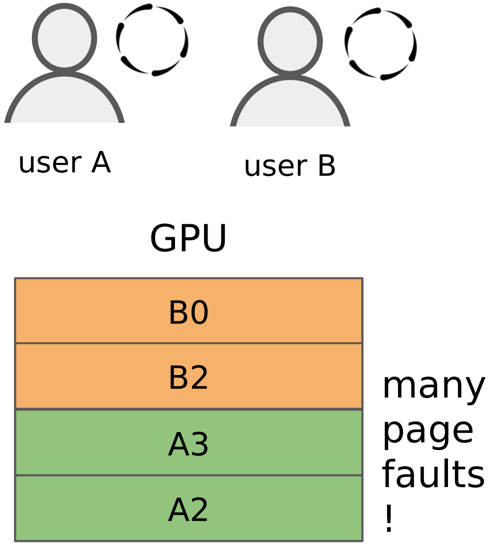
Sometimes



$\Sigma Wss_i < \text{GPU memory}$

No page faults

Other times



$\Sigma Wss_i > \text{GPU memory}$

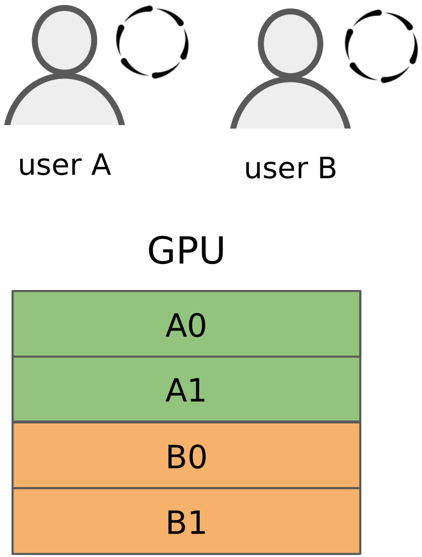
Thrashing!

Solution

When GPU bursts overlap

Sometimes	Other times	Solution

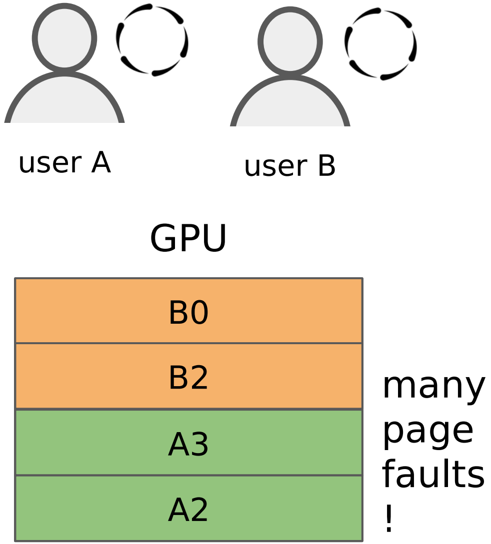
Sometimes



$\Sigma Wss_i <$ GPU memory

No page faults

Other times

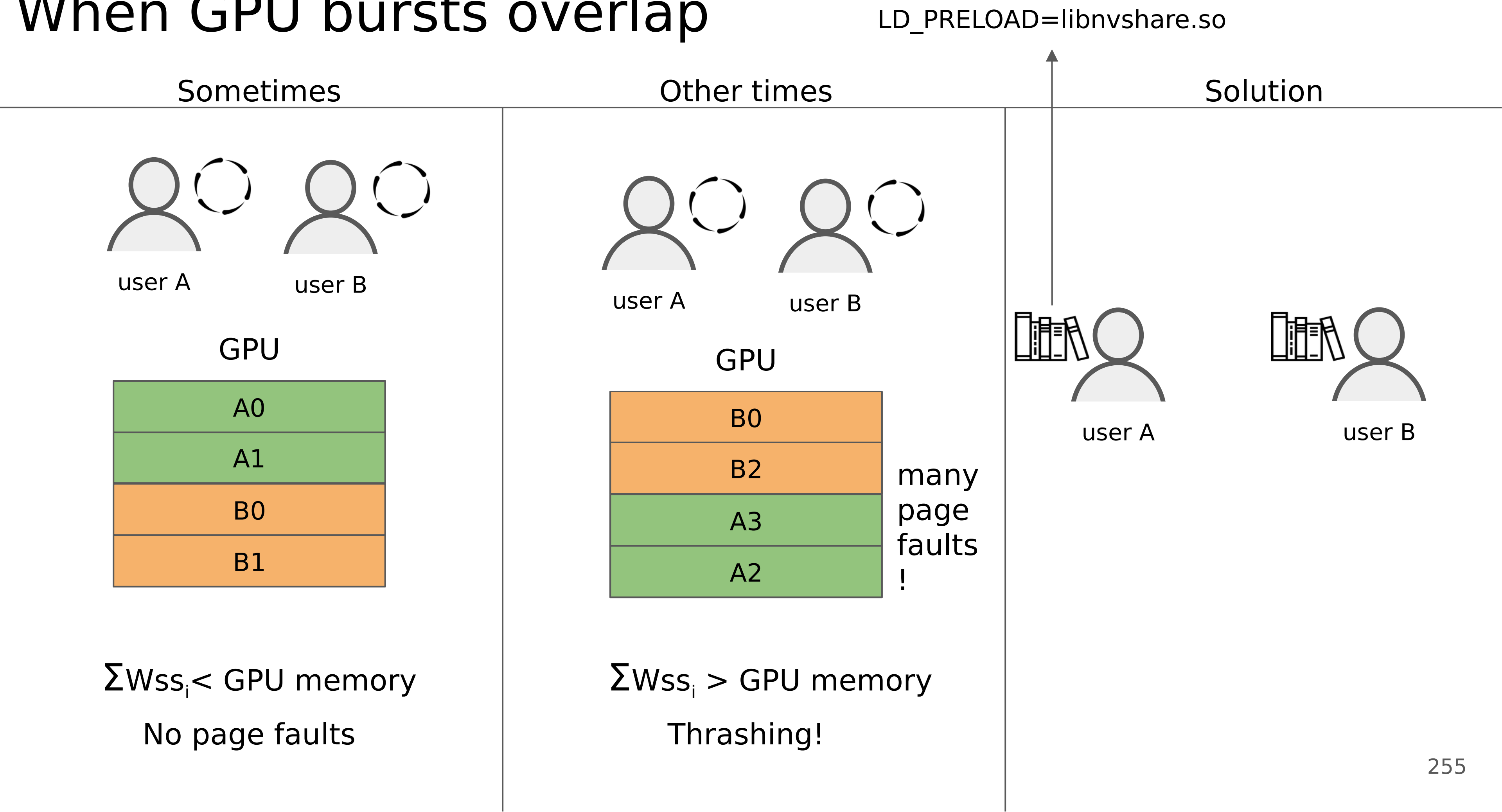


$\Sigma Wss_i >$ GPU memory

Thrashing!

Solution

When GPU bursts overlap



When GPU bursts overlap

LD_PRELOAD=libnvshare.so

Sometimes

user A

user B

GPU

A0
A1
B0
B1

$\Sigma Wss_i <$ GPU memory

No page faults

Other times

user A

user B

GPU

B0
B2
A3
A2

many page faults !

$\Sigma Wss_i >$ GPU memory

Thrashing!

Solution

user A

user B

When GPU bursts overlap

LD_PRELOAD=libnvshare.so

Sometimes

user A

user B

GPU

A0
A1
B0
B1

$\Sigma Wss_i <$ GPU memory

No page faults

Other times

user A

user B

GPU

B0
B2
A3
A2

many page faults !

$\Sigma Wss_i >$ GPU memory

Thrashing!

Solution

user A

user B

When GPU bursts overlap

LD_PRELOAD=libnvshare.so

Sometimes

user A

user B

GPU

A0
A1
B0
B1

$\Sigma Wss_i <$ GPU memory

No page faults

Other times

user A

user B

GPU

B0
B2
A3
A2

many page faults !

$\Sigma Wss_i >$ GPU memory

Thrashing!

Solution

user A

user B

GPU

B0
B2
A3
A2

When GPU bursts overlap

LD_PRELOAD=libnvshare.so

Sometimes

user A

user B

GPU

A0
A1
B0
B1

$\Sigma Wss_i <$ GPU memory

No page faults

Other times

user A

user B

GPU

B0
B2
A3
A2

many page faults !

$\Sigma Wss_i >$ GPU memory

Thrashing!

Solution

user A

user B

GPU

B0
B2
A3
A2

- pause only GPU computations

When GPU bursts overlap

LD_PRELOAD=libnvshare.so

Sometimes

user A

user B

GPU

A0
A1
B0
B1

$\Sigma Wss_i <$ GPU memory

No page faults

Other times

user A

user B

GPU

B0
B2
A3
A2

many page faults !

$\Sigma Wss_i >$ GPU memory

Thrashing!

Solution

GO

user A

user B

GPU

B0
B2
A3
A2

- pause only GPU computations

When GPU bursts overlap

LD_PRELOAD=libnvshare.so

Sometimes

user A   user B

GPU

A0
A1
B0
B1

$\Sigma Wss_i <$ GPU memory

No page faults

Other times

user A   user B

GPU

B0
B2
A3
A2

many page faults !

$\Sigma Wss_i >$ GPU memory

Thrashing!

Solution

GO

user A   user B

GPU

B0
B2
A3
A2

- pause only GPU computations

When GPU bursts overlap

LD_PRELOAD=libnvshare.so

Sometimes

user A

user B

GPU

A0
A1
B0
B1

$\Sigma Wss_i <$ GPU memory

No page faults

Other times

user A

user B

GPU

B0
B2
A3
A2

many page faults !

$\Sigma Wss_i >$ GPU memory

Thrashing!

Solution

GO

user A

user B

GPU

A0
B2
A3
A2

- pause only GPU computations

When GPU bursts overlap

LD_PRELOAD=libnvshare.so

Sometimes

user A

user B

GPU

A0
A1
B0
B1

$\Sigma Wss_i <$ GPU memory

No page faults

Other times

user A

user B

GPU

B0
B2
A3
A2

many page faults !

$\Sigma Wss_i >$ GPU memory

Thrashing!

Solution

GO

user A

user B

GPU

A0
A1
A3
A2

- pause only GPU computations

When GPU bursts overlap

LD_PRELOAD=libnvshare.so

Sometimes

user A

user B

GPU

A0
A1
B0
B1

$\Sigma Wss_i <$ GPU memory

No page faults

Other times

user A

user B

GPU

B0
B2
A3
A2

many page faults !

$\Sigma Wss_i >$ GPU memory

Thrashing!

Solution

GO

user A

user B

GPU

A0
A1
A3
A2

- pause only GPU computations

When GPU bursts overlap

LD_PRELOAD=libnvshare.so

Sometimes

user A

user B

GPU

A0
A1
B0
B1

$\Sigma Wss_i <$ GPU memory

No page faults

Other times

user A

user B

GPU

B0
B2
A3
A2

many page faults !

$\Sigma Wss_i >$ GPU memory

Thrashing!

Solution

GO

WAIT

user A

user B

GPU

A0
A1
A3
A2

- pause only GPU computations

When GPU bursts overlap

LD_PRELOAD=libnvshare.so

Sometimes

user A user B

GPU

A0
A1
B0
B1

$\Sigma Wss_i <$ GPU memory

No page faults

Other times

user A user B

GPU

B0
B2
A3
A2

many page faults !

$\Sigma Wss_i >$ GPU memory

Thrashing!

Solution

GO

WAIT

user A user B

GPU

A0
A1
A3
A2

- pause only GPU computations

When GPU bursts overlap

LD_PRELOAD=libnvshare.so

Sometimes

user A   user B

GPU

A0
A1
B0
B1

$\Sigma Wss_i <$ GPU memory

No page faults

Other times

user A   user B

GPU

B0
B2
A3
A2

many page faults !

$\Sigma Wss_i >$ GPU memory

Thrashing!

Solution

GO

WAIT

user A   user B

GPU

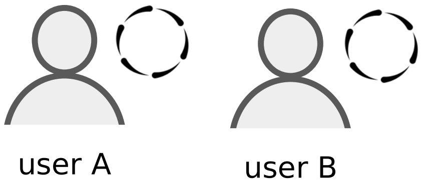
A0
A1
A3
A2

- pause only GPU computations

When GPU bursts overlap

LD_PRELOAD=libnvshare.so

Sometimes



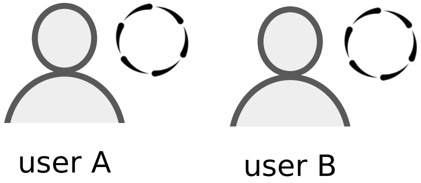
GPU

A0
A1
B0
B1

$\Sigma Wss_i <$ GPU memory

No page faults

Other times



GPU

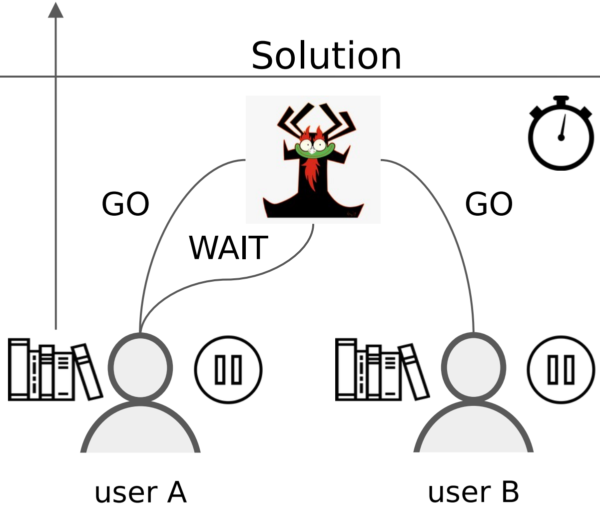
B0
B2
A3
A2

many page faults !

$\Sigma Wss_i >$ GPU memory

Thrashing!

Solution



GPU

A0
A1
A3
A2

- pause only GPU computations

When GPU bursts overlap

LD_PRELOAD=libnvshare.so

Sometimes

user A

user B

GPU

A0
A1
B0
B1

$\Sigma Wss_i <$ GPU memory

No page faults

Other times

user A

user B

GPU

B0
B2
A3
A2

many page faults !

$\Sigma Wss_i >$ GPU memory

Thrashing!

Solution

GO

WAIT

GO

user A

user B

GPU

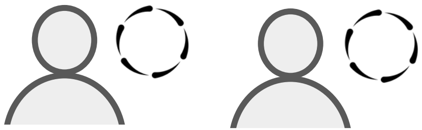
A0
A1
A3
A2

- pause only GPU computations

When GPU bursts overlap

LD_PRELOAD=libnvshare.so

Sometimes



user A user B

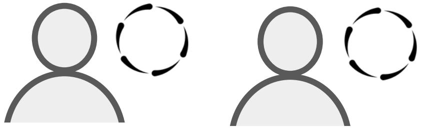
GPU

A0
A1
B0
B1

$\Sigma Wss_i <$ GPU memory

No page faults

Other times



user A user B

GPU

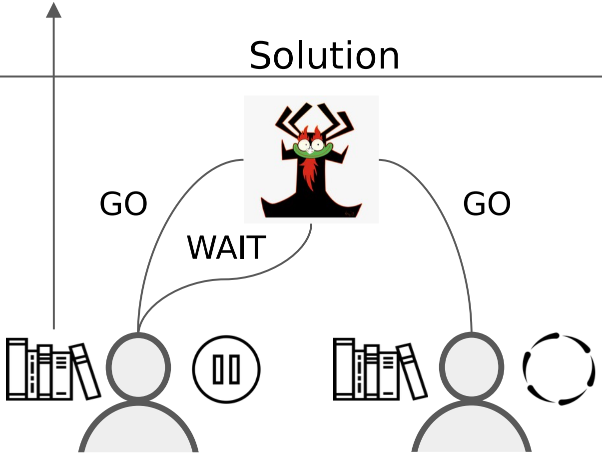
B0
B2
A3
A2

many page faults !

$\Sigma Wss_i >$ GPU memory

Thrashing!

Solution



user A user B

GPU

A0
A1
B0
A2

- pause only GPU computations

When GPU bursts overlap

LD_PRELOAD=libnvshare.so

Sometimes

user A user B

GPU

A0
A1
B0
B1

$\Sigma Wss_i <$ GPU memory

No page faults

Other times

user A user B

GPU

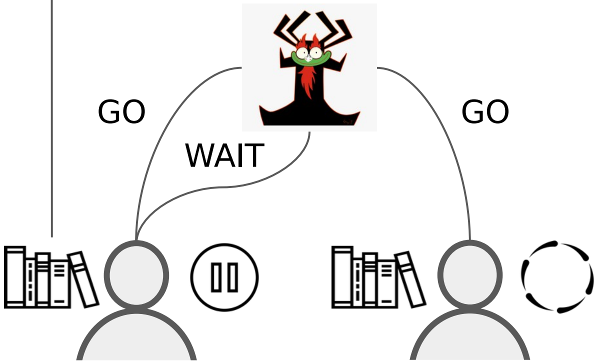
B0
B2
A3
A2

many page faults !

$\Sigma Wss_i >$ GPU memory

Thrashing!

Solution



user A user B

GPU

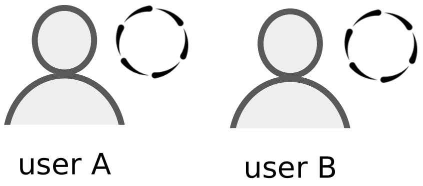
A0
A1
B0
B1

- pause only GPU computations

When GPU bursts overlap

LD_PRELOAD=libnvshare.so

Sometimes



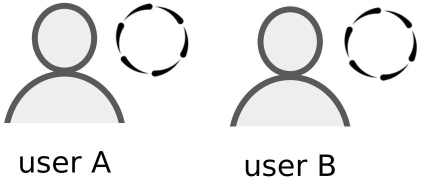
GPU

A0
A1
B0
B1

$\Sigma Wss_i <$ GPU memory

No page faults

Other times



GPU

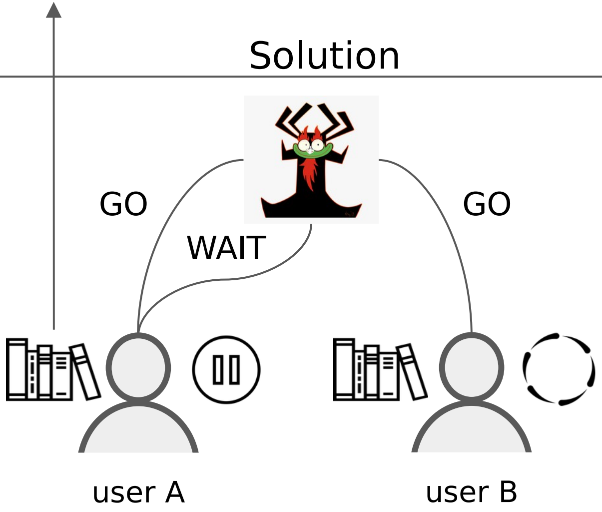
B0
B2
A3
A2

many page faults !

$\Sigma Wss_i >$ GPU memory

Thrashing!

Solution



GPU

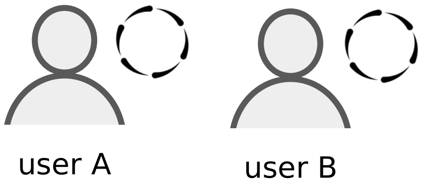
B2
A1
B0
B1

- pause only GPU computations

When GPU bursts overlap

LD_PRELOAD=libnvshare.so

Sometimes



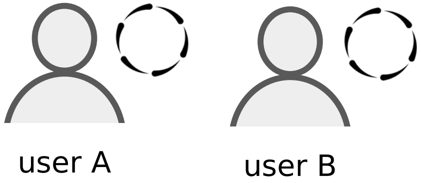
GPU

A0
A1
B0
B1

$\Sigma Wss_i <$ GPU memory

No page faults

Other times



GPU

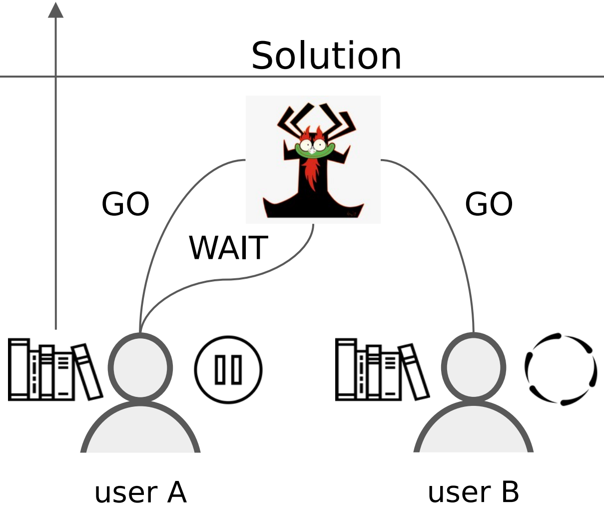
B0
B2
A3
A2

many page faults !

$\Sigma Wss_i >$ GPU memory

Thrashing!

Solution



GPU

B2
B3
B0
B1

- pause only GPU computations

When GPU bursts overlap

LD_PRELOAD=libnvshare.so

Sometimes

user A

user B

GPU

A0
A1
B0
B1

$\Sigma Wss_i <$ GPU memory

No page faults

Other times

user A

user B

GPU

B0
B2
A3
A2

many page faults !

$\Sigma Wss_i >$ GPU memory

Thrashing!

Solution

GO

WAIT

GO

user A

user B

GPU

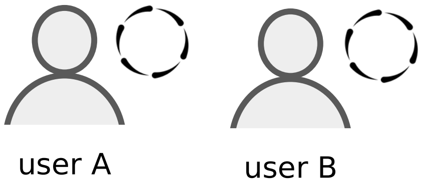
B2
B3
B0
B1

- pause only GPU computations

When GPU bursts overlap

LD_PRELOAD=libnvshare.so

Sometimes



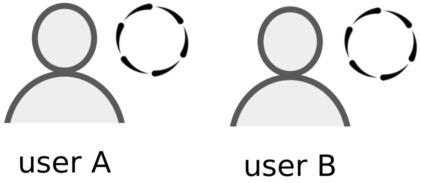
GPU

A0
A1
B0
B1

$\Sigma Wss_i <$ GPU memory

No page faults

Other times



GPU

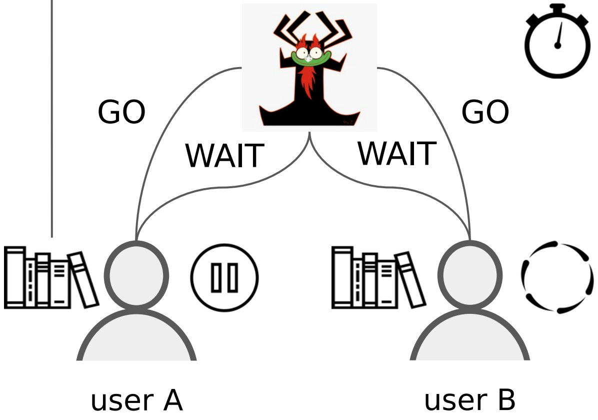
B0
B2
A3
A2

many page faults !

$\Sigma Wss_i >$ GPU memory

Thrashing!

Solution



GPU

B2
B3
B0
B1

- pause only GPU computations

Anti-thrashing idea

Anti-thrashing idea

- WSS > GPU Memory

Anti-thrashing idea

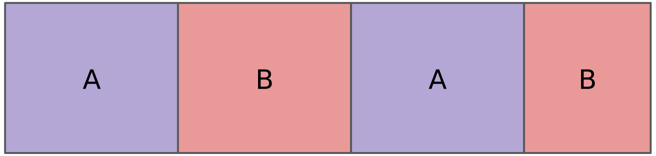
- WSS > GPU Memory

Default case:

Anti-thrashing idea

- WSS > GPU Memory

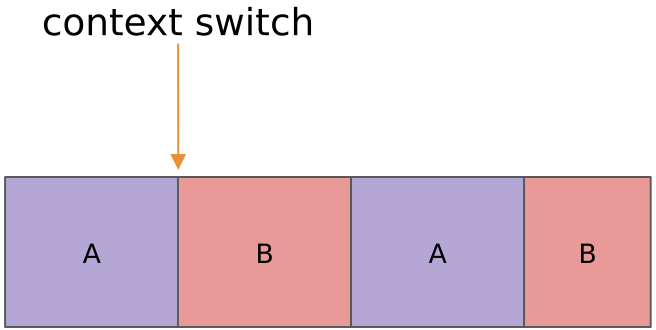
Default case:



Anti-thrashing idea

- WSS > GPU Memory

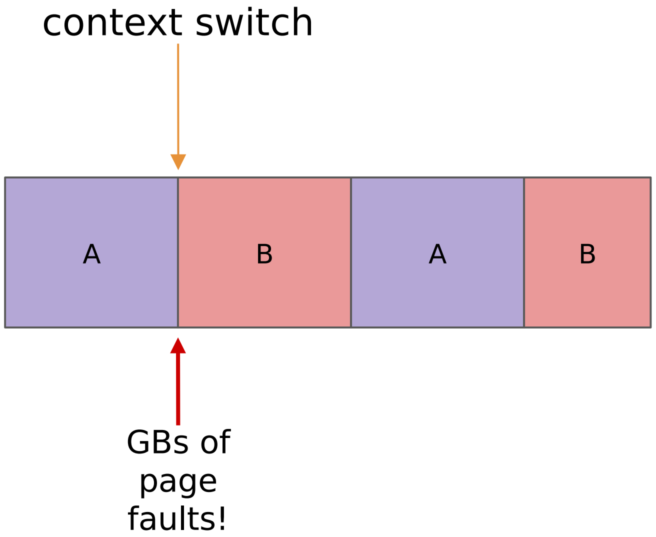
<u>Default case:</u>



Anti-thrashing idea

- WSS > GPU Memory

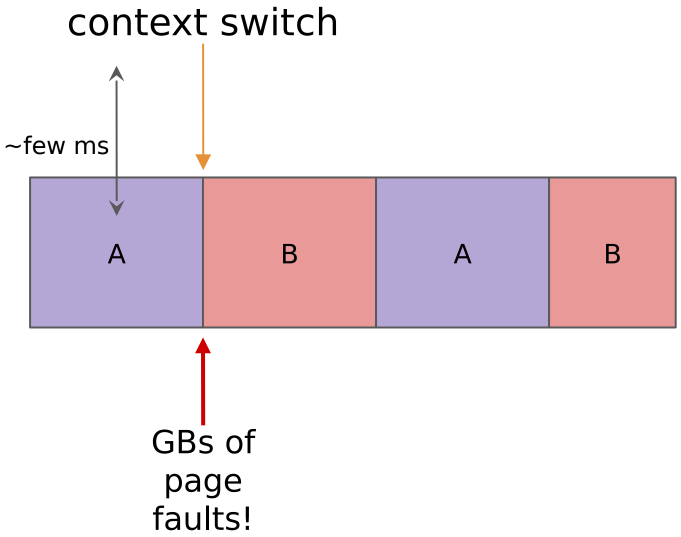
Default case:



Anti-thrashing idea

- WSS > GPU Memory

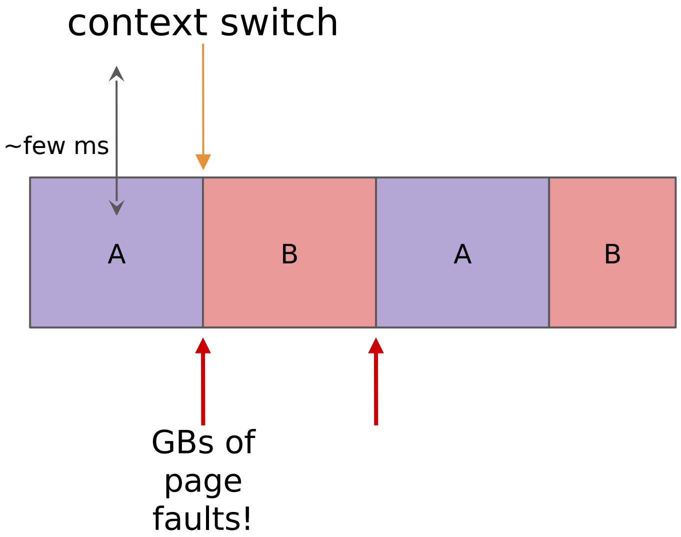
<u>Default case:</u>



Anti-thrashing idea

- WSS > GPU Memory

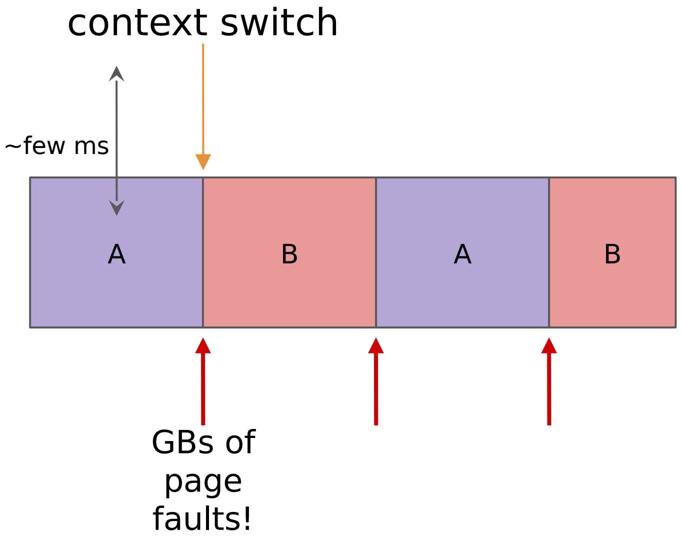
Default case:



Anti-thrashing idea

- WSS > GPU Memory

Default case:

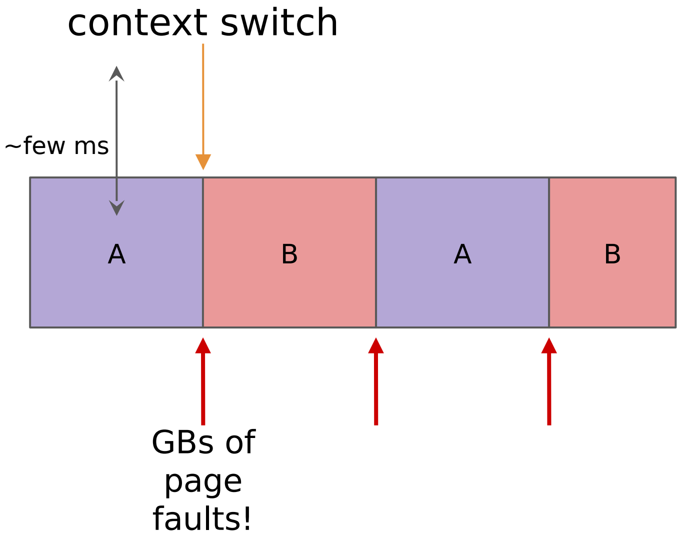


Anti-thrashing idea

- WSS > GPU Memory

<u>Default case:</u>

- memory swaps happen too frequently

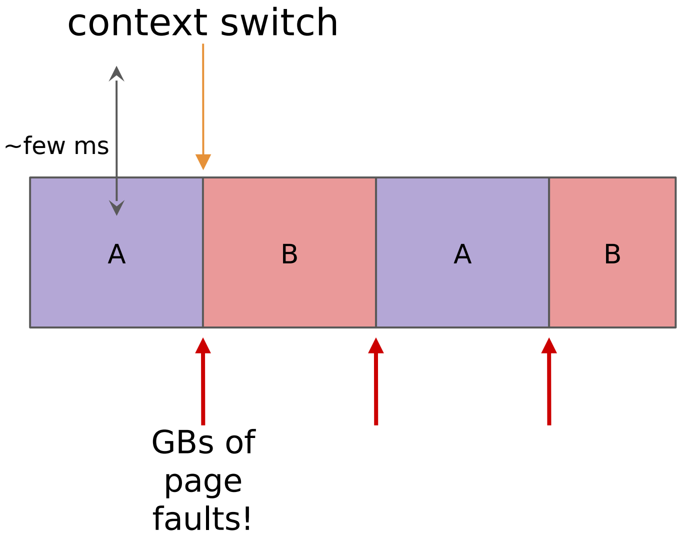


Anti-thrashing idea

- WSS > GPU Memory

Default case:

- memory swaps happen too frequently
- Thrashing!

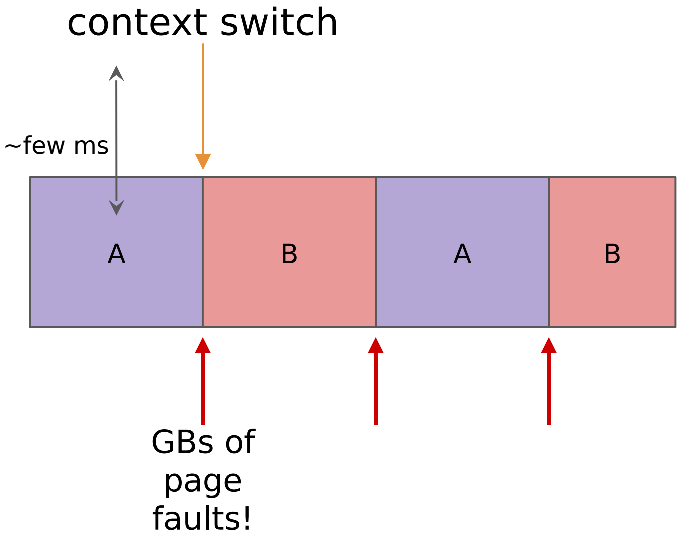


Anti-thrashing idea

- WSS > GPU Memory

Default case:

- memory swaps happen too frequently
- Thrashing!



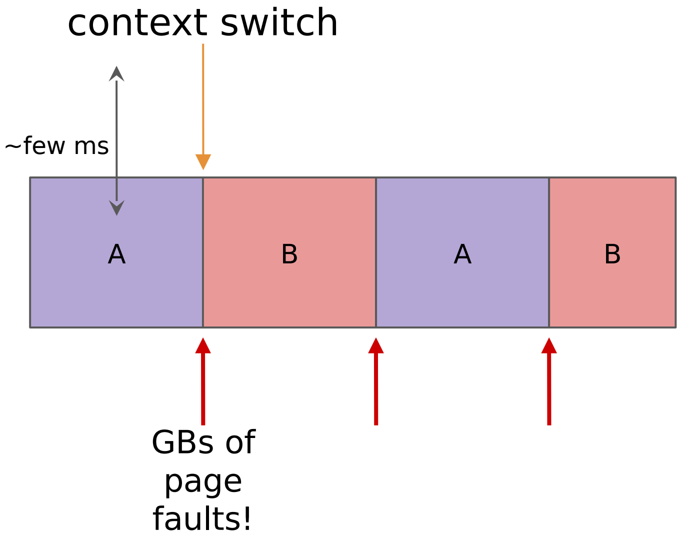
anti-thrashing mechanism:

Anti-thrashing idea

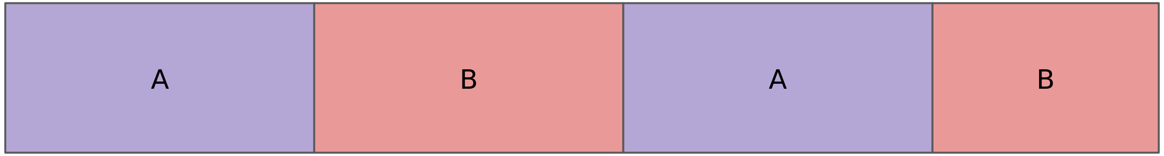
- WSS > GPU Memory

<u>Default case:</u>

- memory swaps happen too frequently
- Thrashing!

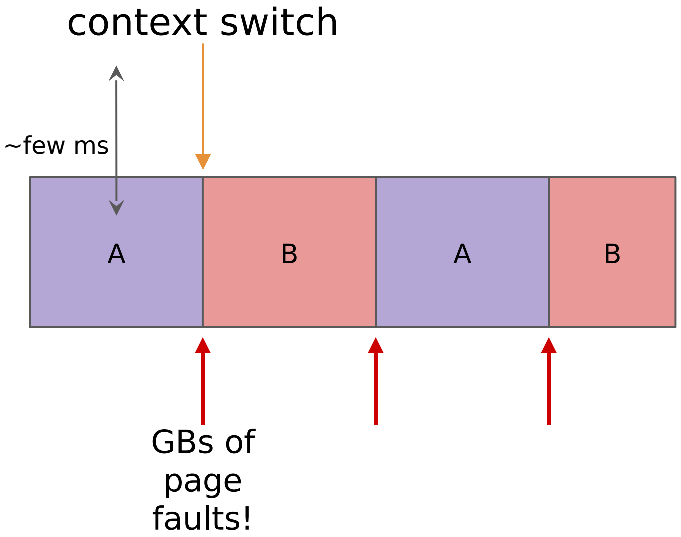


<u>anti-thrashing mechanism:</u>



Anti-thrashing idea

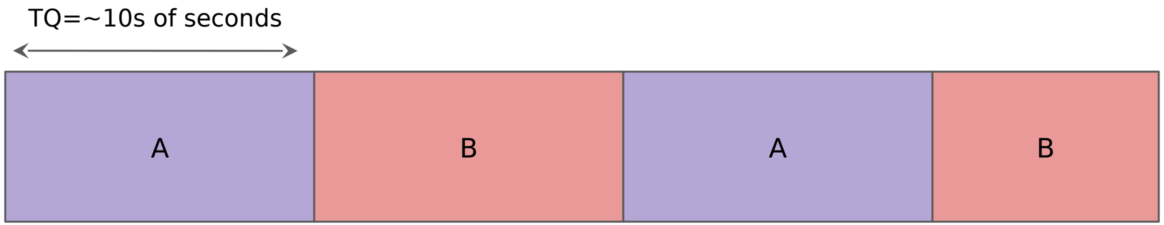
- WSS > GPU Memory



Default case:

- memory swaps happen too frequently
- Thrashing!

anti-thrashing mechanism:



Anti-thrashing idea

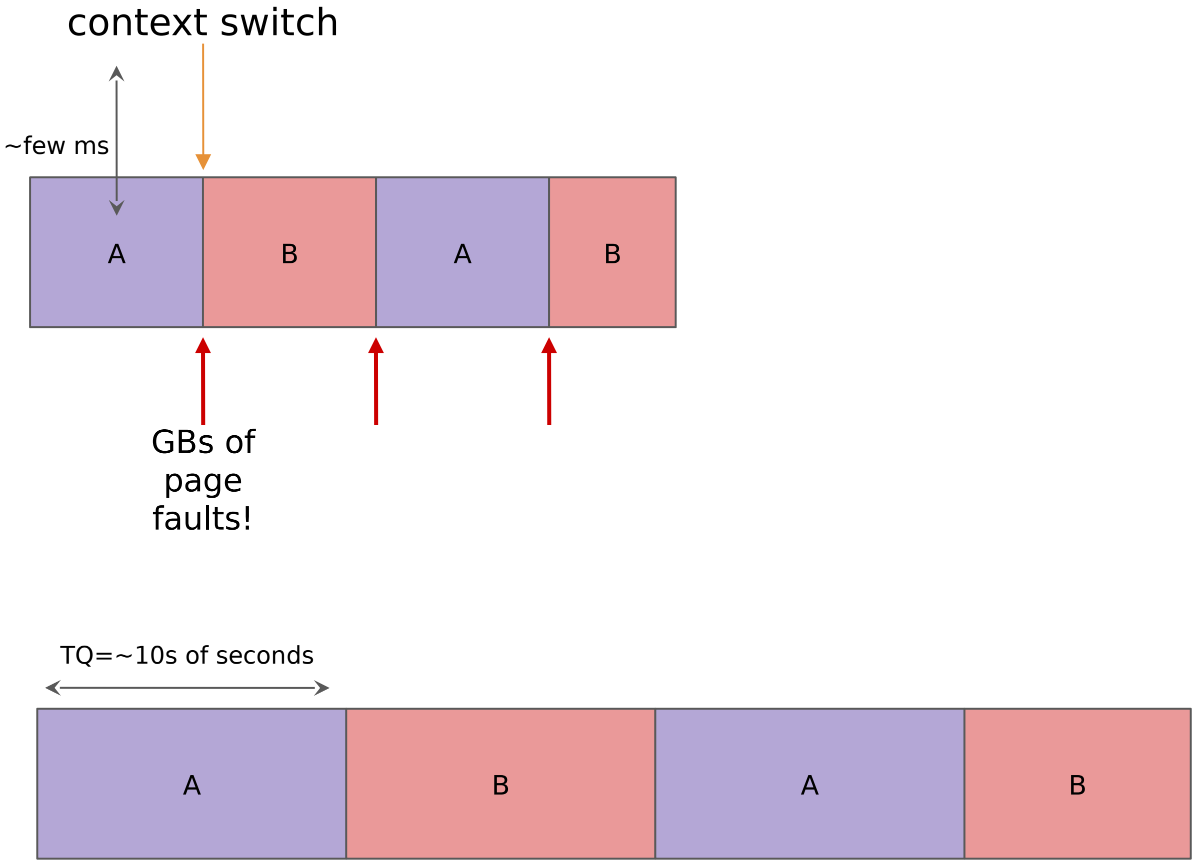
- WSS > GPU Memory

Default case:

- memory swaps happen too frequently
- Thrashing!

anti-thrashing mechanism:

- orders of magnitude fewer memory swaps

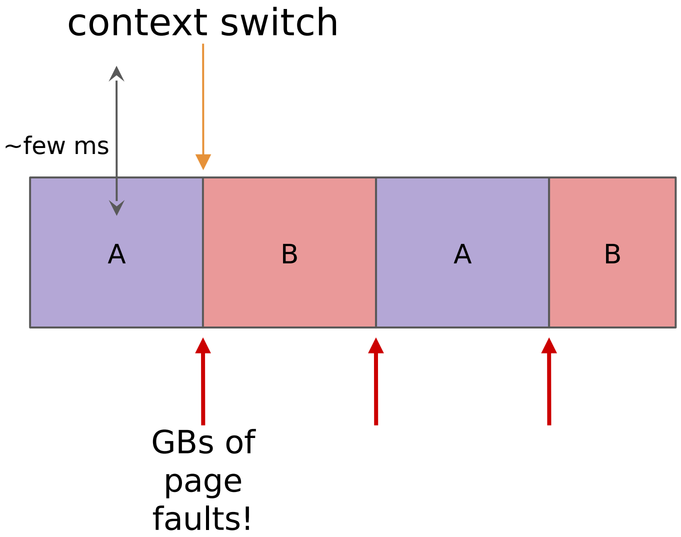


Anti-thrashing idea

- WSS > GPU Memory

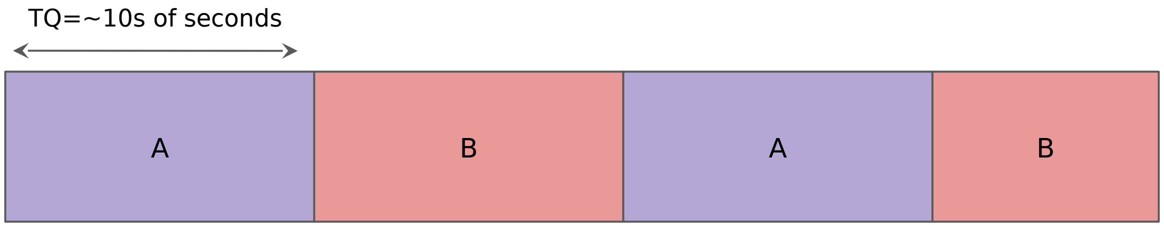
Default case:

- memory swaps happen too frequently
- Thrashing!



anti-thrashing mechanism:

- orders of magnitude fewer memory swaps
- Cooperative Scheduling

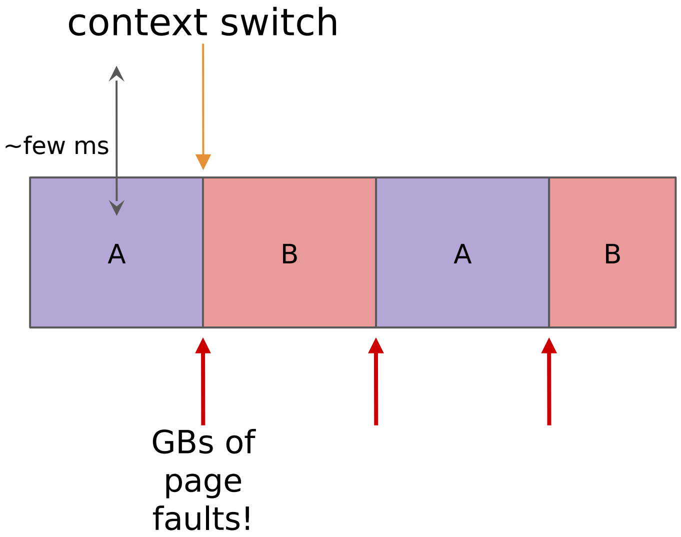


Anti-thrashing idea

- WSS > GPU Memory

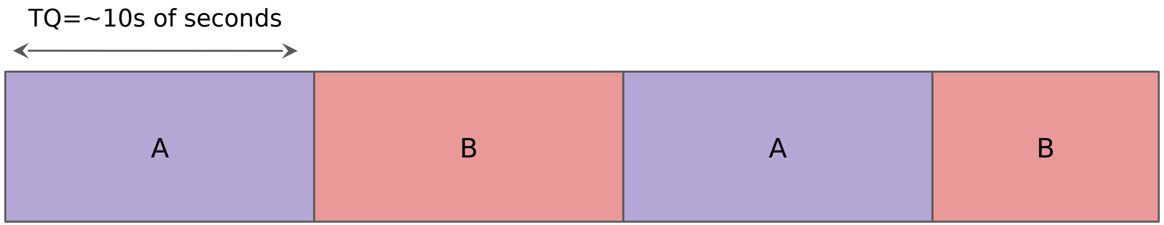
Default case:

- memory swaps happen too frequently
- Thrashing!



anti-thrashing mechanism:

- orders of magnitude fewer memory swaps
- Cooperative Scheduling
- No Thrashing!



Demo

Evaluation

Evaluation

- 16-core CPU (Intel Xeon 2.3 GHz - Haswell), 104 GB of RAM and an NVIDIA Tesla P100 GPU (16 GB).

Evaluation

- 16-core CPU (Intel Xeon 2.3 GHz - Haswell), 104 GB of RAM and an NVIDIA Tesla P100 GPU (16 GB).

- non-interactive (conventional) tasks

Evaluation

- 16-core CPU (Intel Xeon 2.3 GHz - Haswell), 104 GB of RAM and an NVIDIA Tesla P100 GPU (16 GB).
- non-interactive (conventional) tasks
- measure **T**otal **C**ompletion **T**ime

Evaluation

- 16-core CPU (Intel Xeon 2.3 GHz - Haswell), 104 GB of RAM and an NVIDIA Tesla P100 GPU (16 GB).
- non-interactive (conventional) tasks
- measure **T**otal **C**ompletion **T**ime

Programs

Name	Working Set Size	GPU/CPU ratio
small_90	7.2 GB	90/10
small_50	7.2 GB	50/50
big_90	15.3 GB	90/10
big_50	15.3 GB	50/50

Evaluation

- 16-core CPU (Intel Xeon 2.3 GHz - Haswell), 104 GB of RAM and an NVIDIA Tesla P100 GPU (16 GB).
- non-interactive (conventional) tasks
- measure **T**otal **C**ompletion **T**ime

Programs

Name	**Working Set Size**	**GPU/CPU ratio**
small_90	7.2 GB	90/10
small_50	7.2 GB	50/50
big_90	15.3 GB	90/10
big_50	15.3 GB	50/50

Evaluation

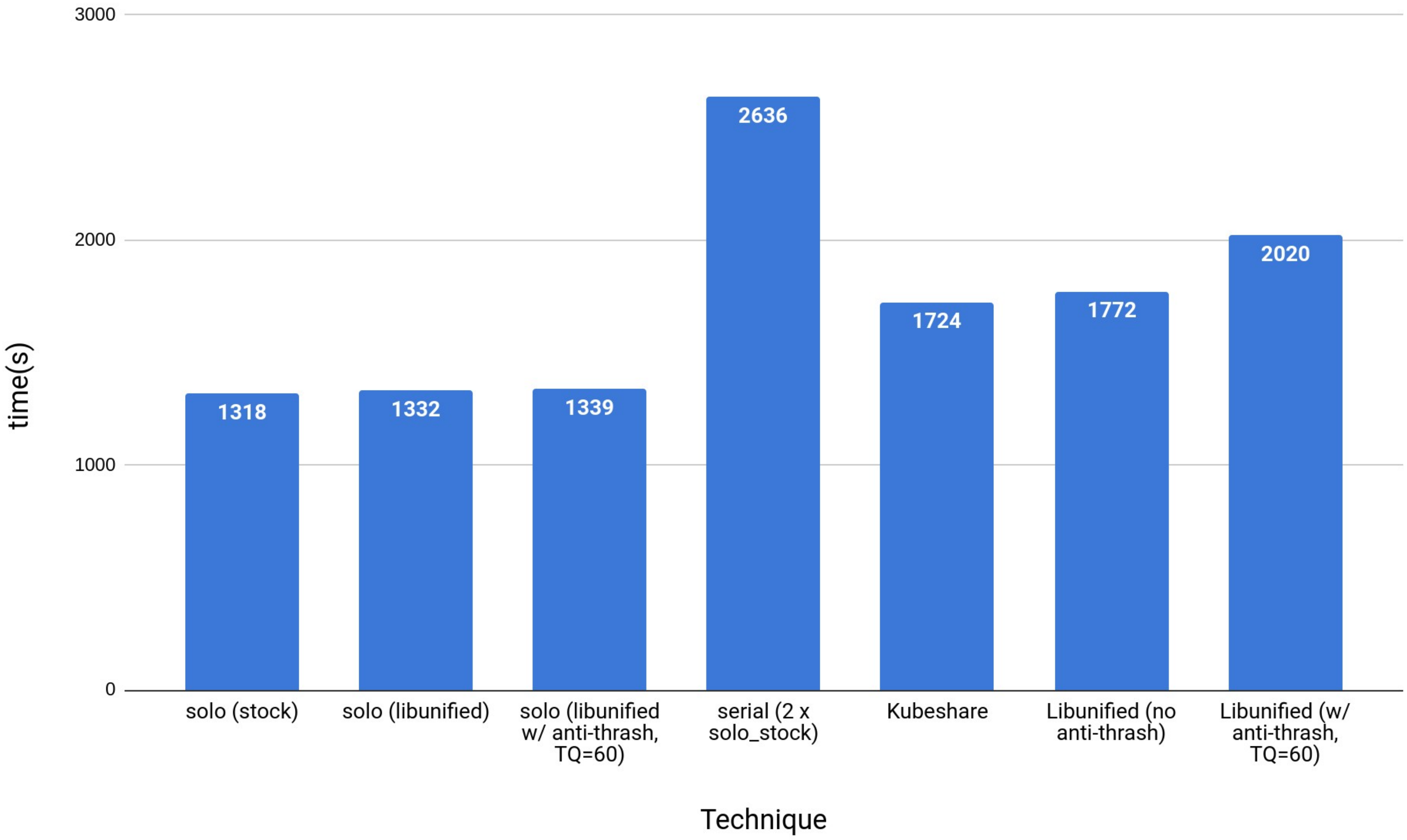
- 16-core CPU (Intel Xeon 2.3 GHz - Haswell), 104 GB of RAM and an NVIDIA Tesla P100 GPU (16 GB).
- non-interactive (conventional) tasks
- measure **T**otal **C**ompletion **T**ime

Programs

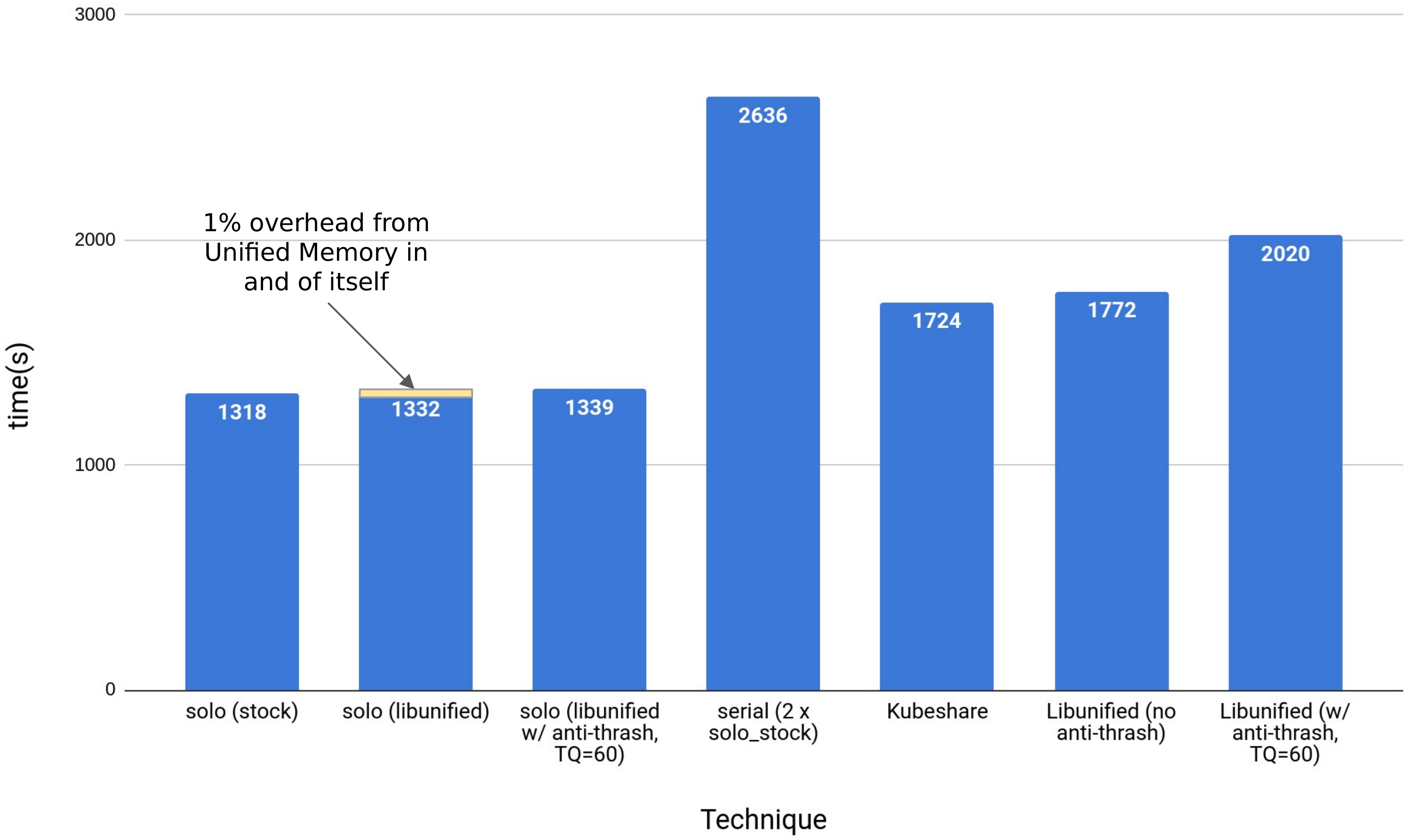
Name	Working Set Size	GPU/CPU ratio
small_90	7.2 GB	90/10
small_50	7.2 GB	50/50
big_90	15.3 GB	90/10
big_50	15.3 GB	50/50

- Running 2 x big_{50,90} in parallel causes **thrashing**!

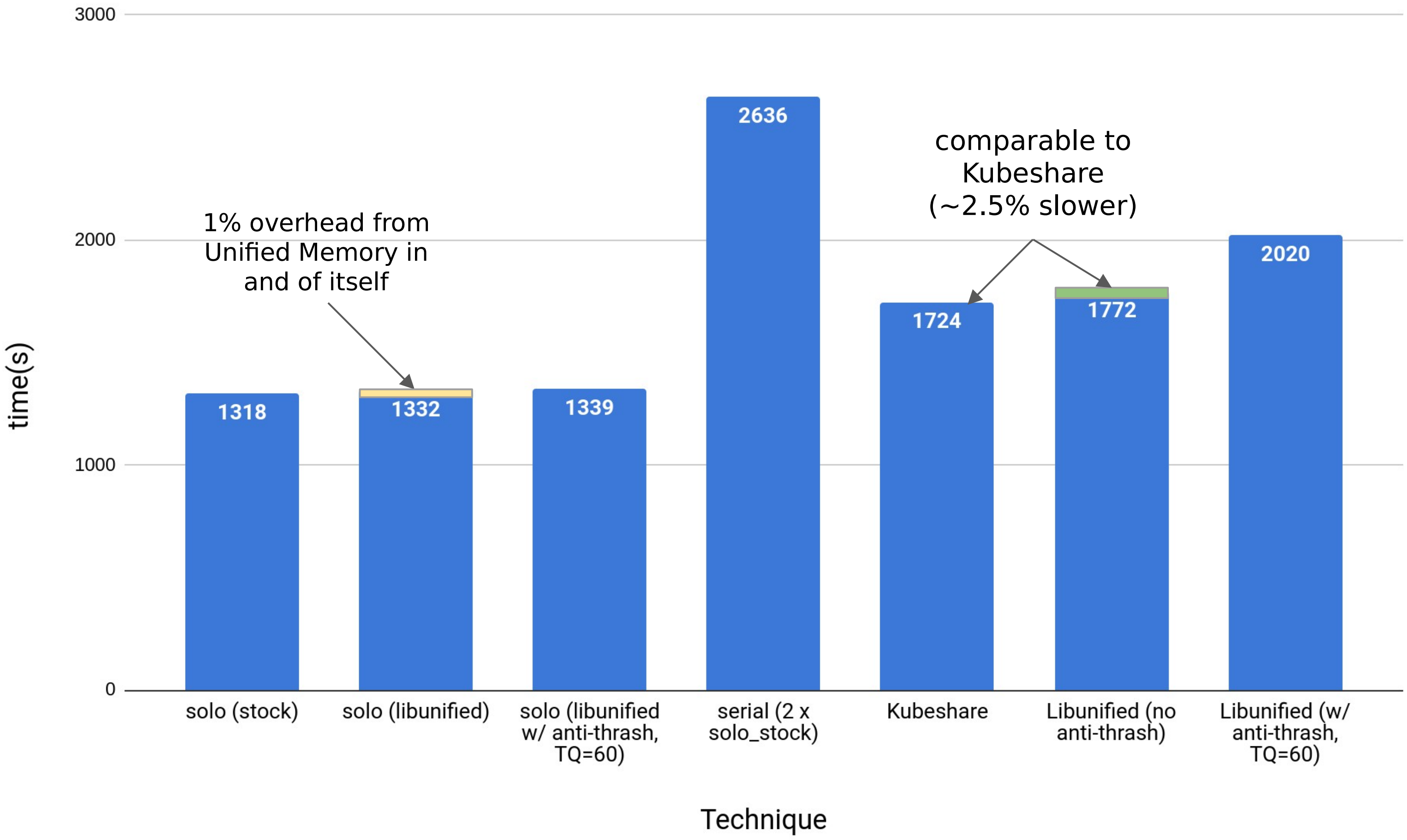
Results for small_50



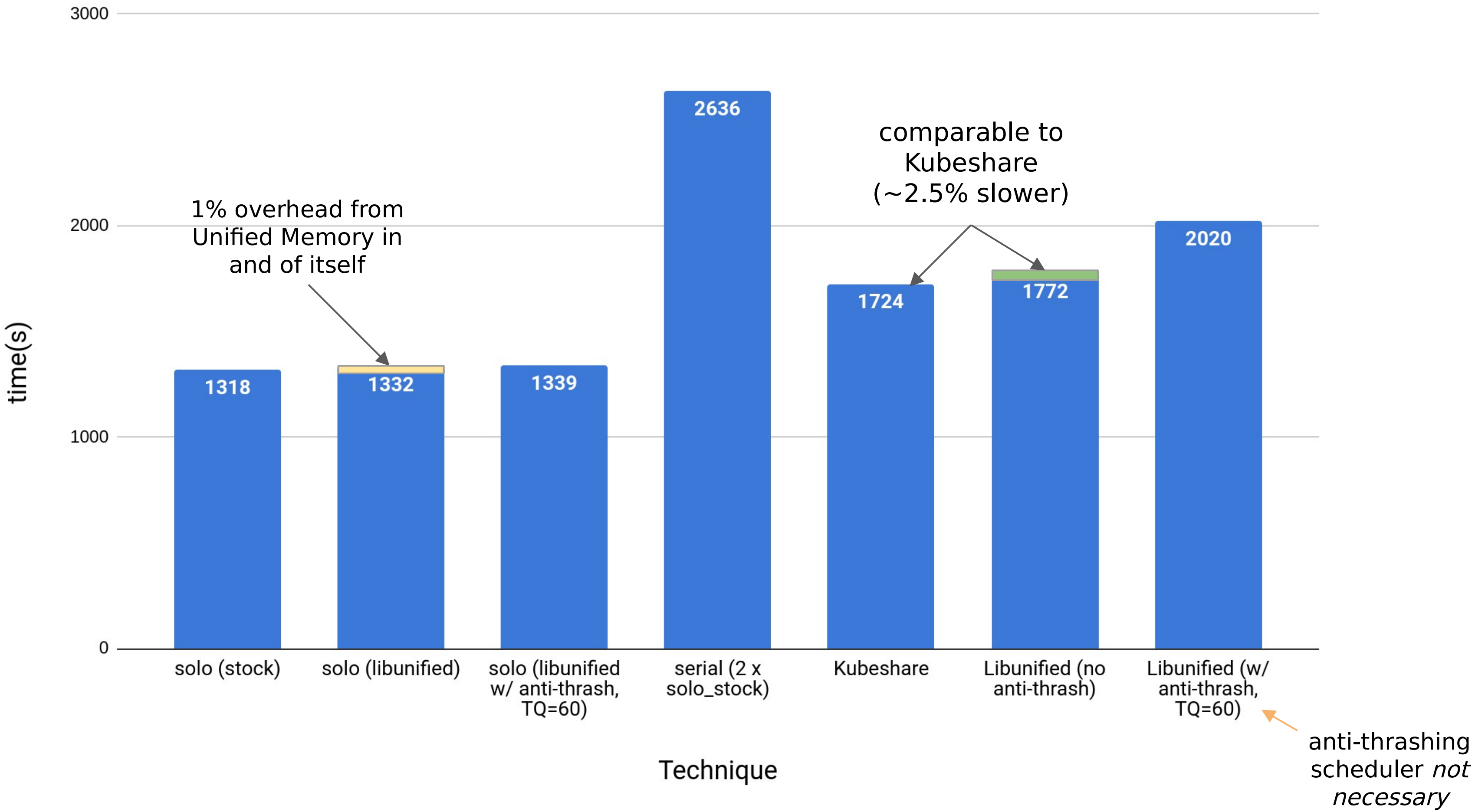
Results for small_50



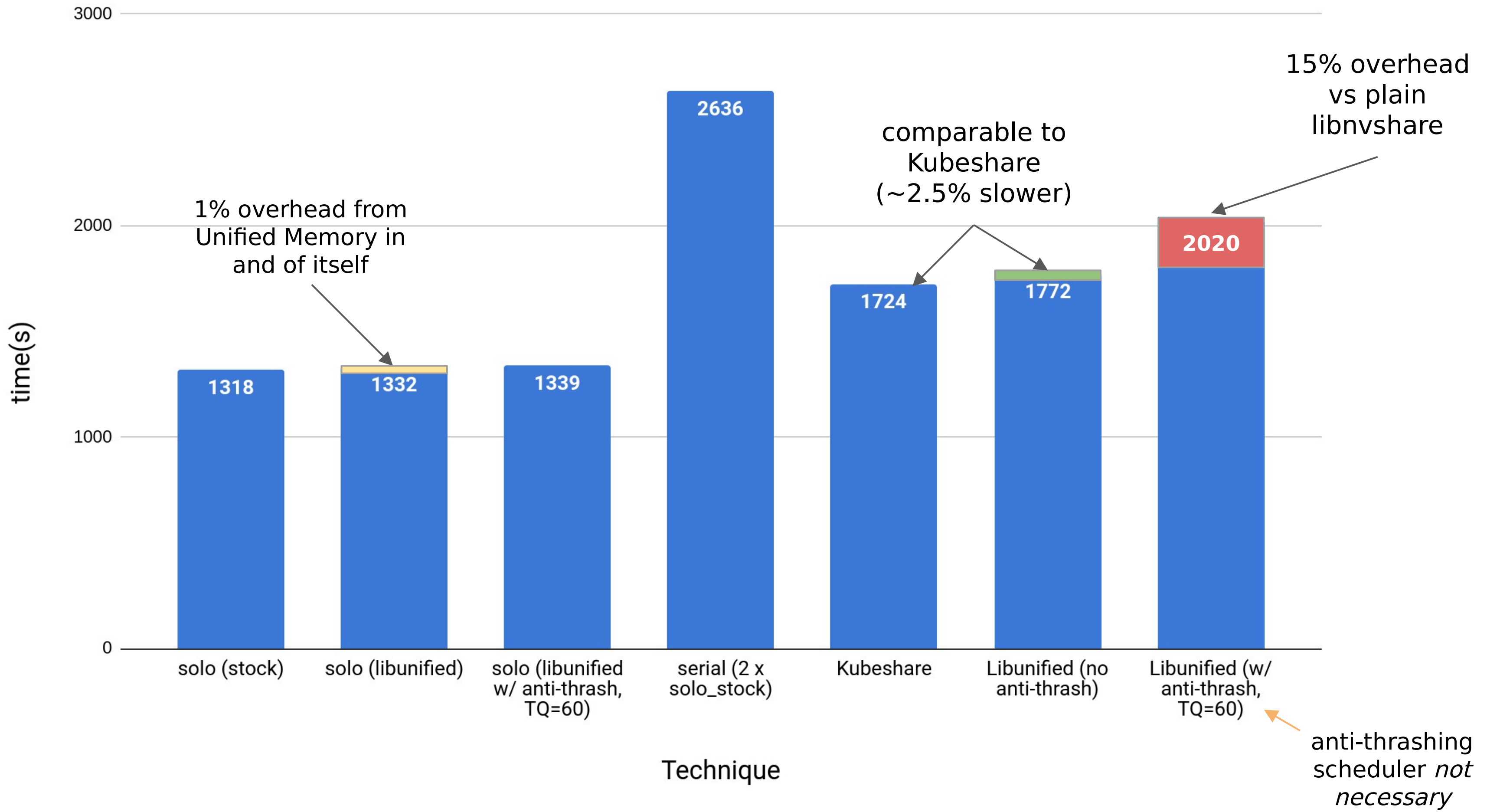
Results for small_50



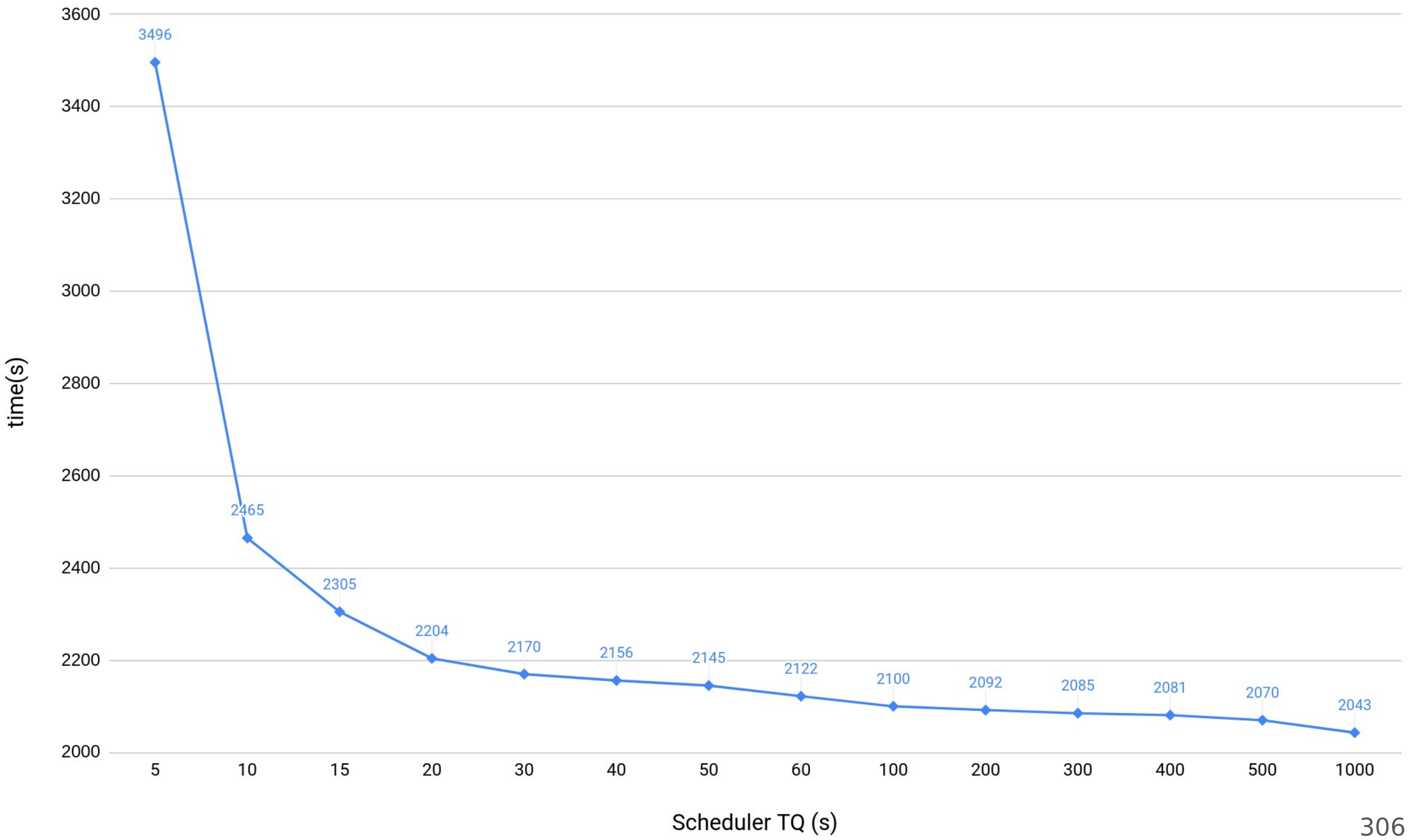
Results for small_50



Results for small_50

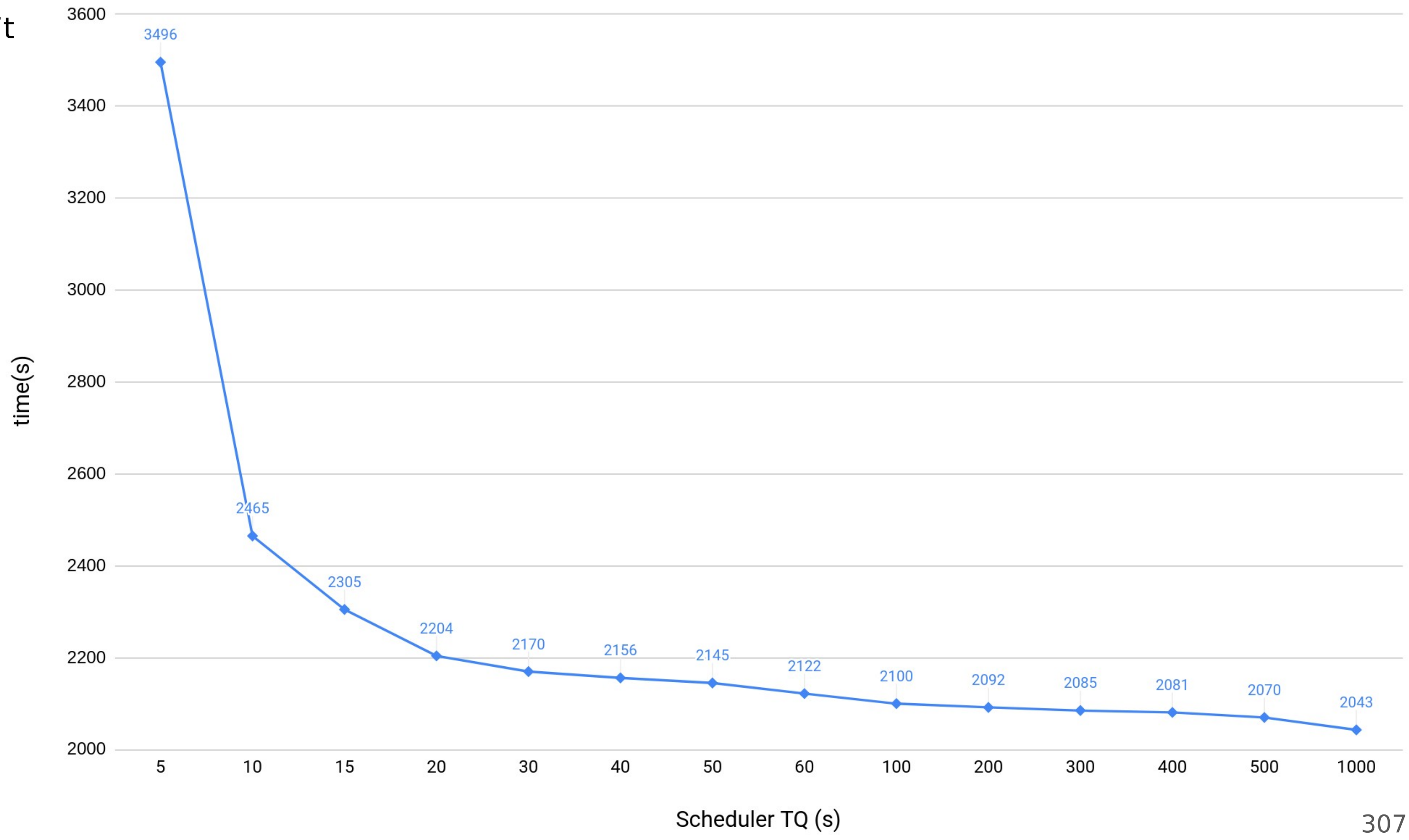


Results for big_50



Results for big_50

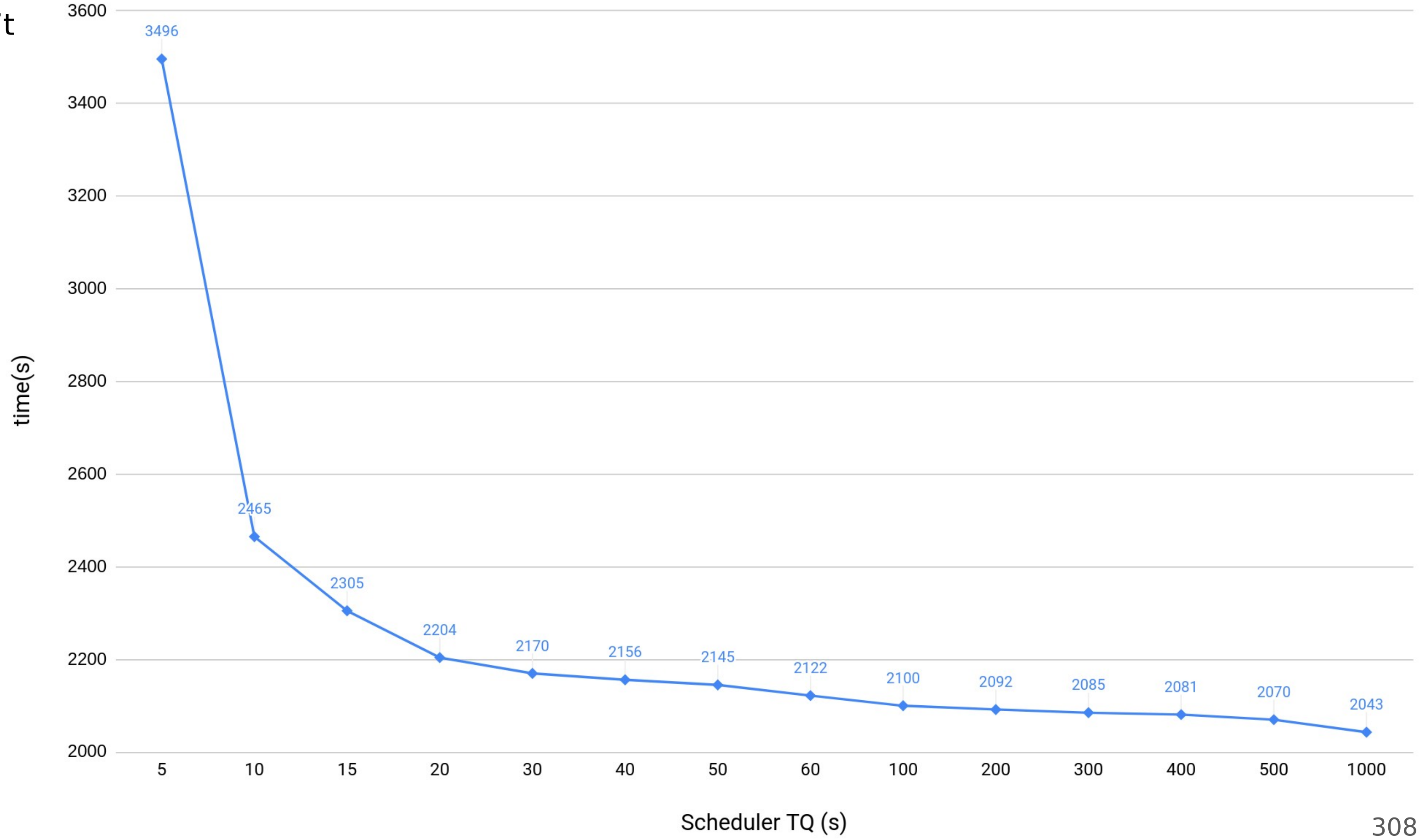
- Kubeshare can't run 2 x big !



Results for big_50

T(libnvshare, no anti-thrash) = 11757 s

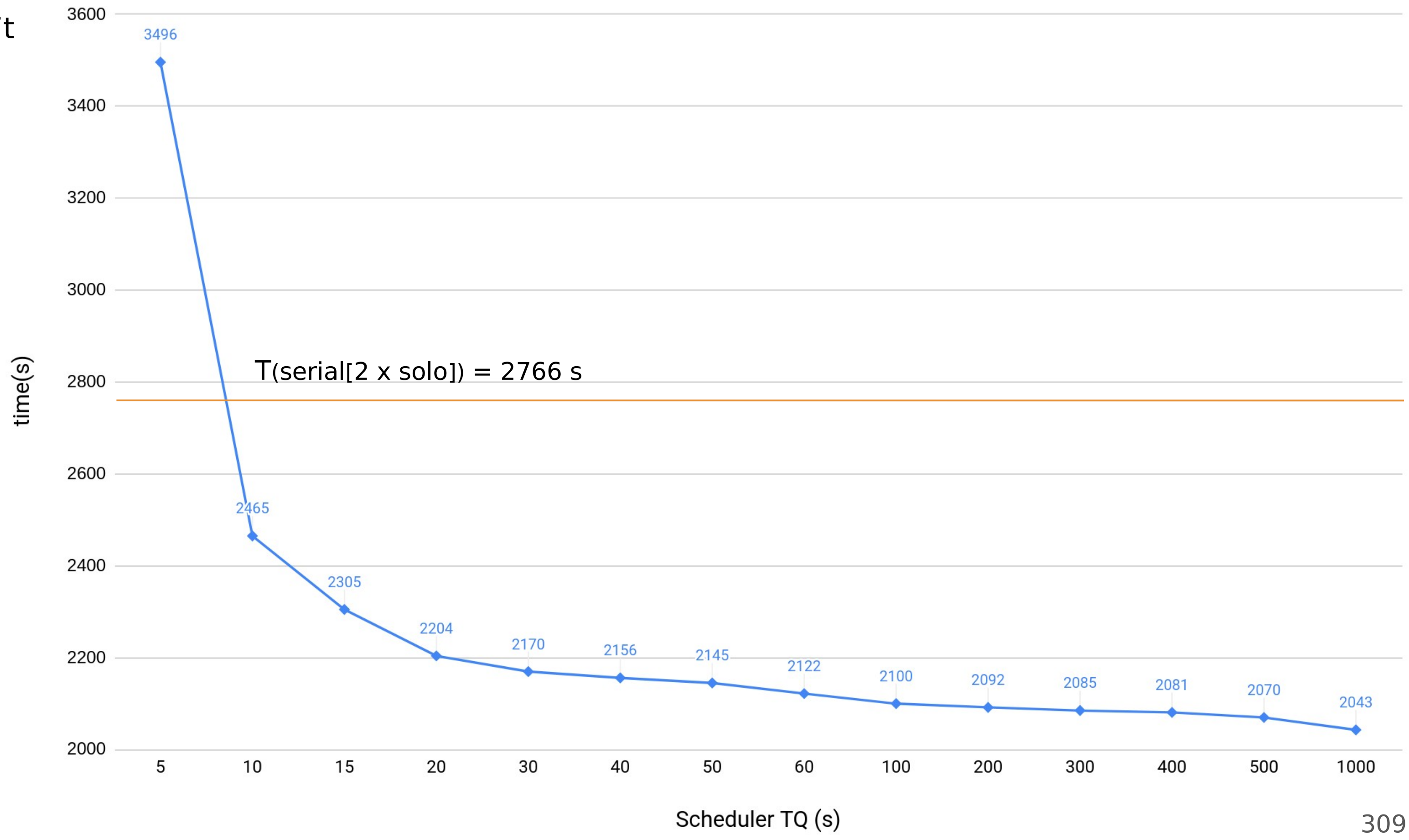
- Kubeshare can’t run 2 x big !



Results for big_50

T(libnvshare, no anti-thrash) = 11757 s

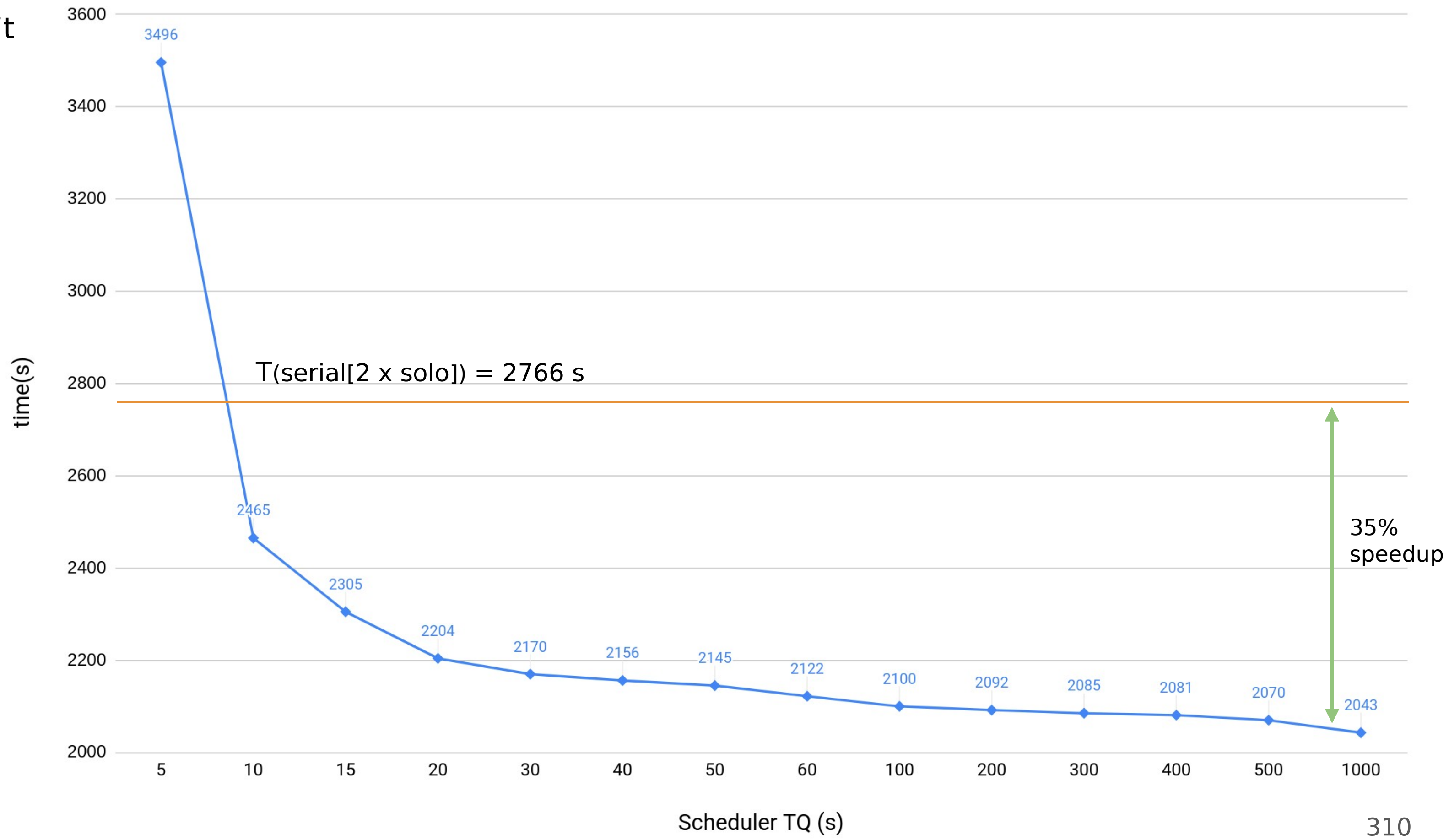
- Kubeshare can’t run 2 x big !



Results for big_50

T(libnvshare, no anti-thrash) = 11757 s

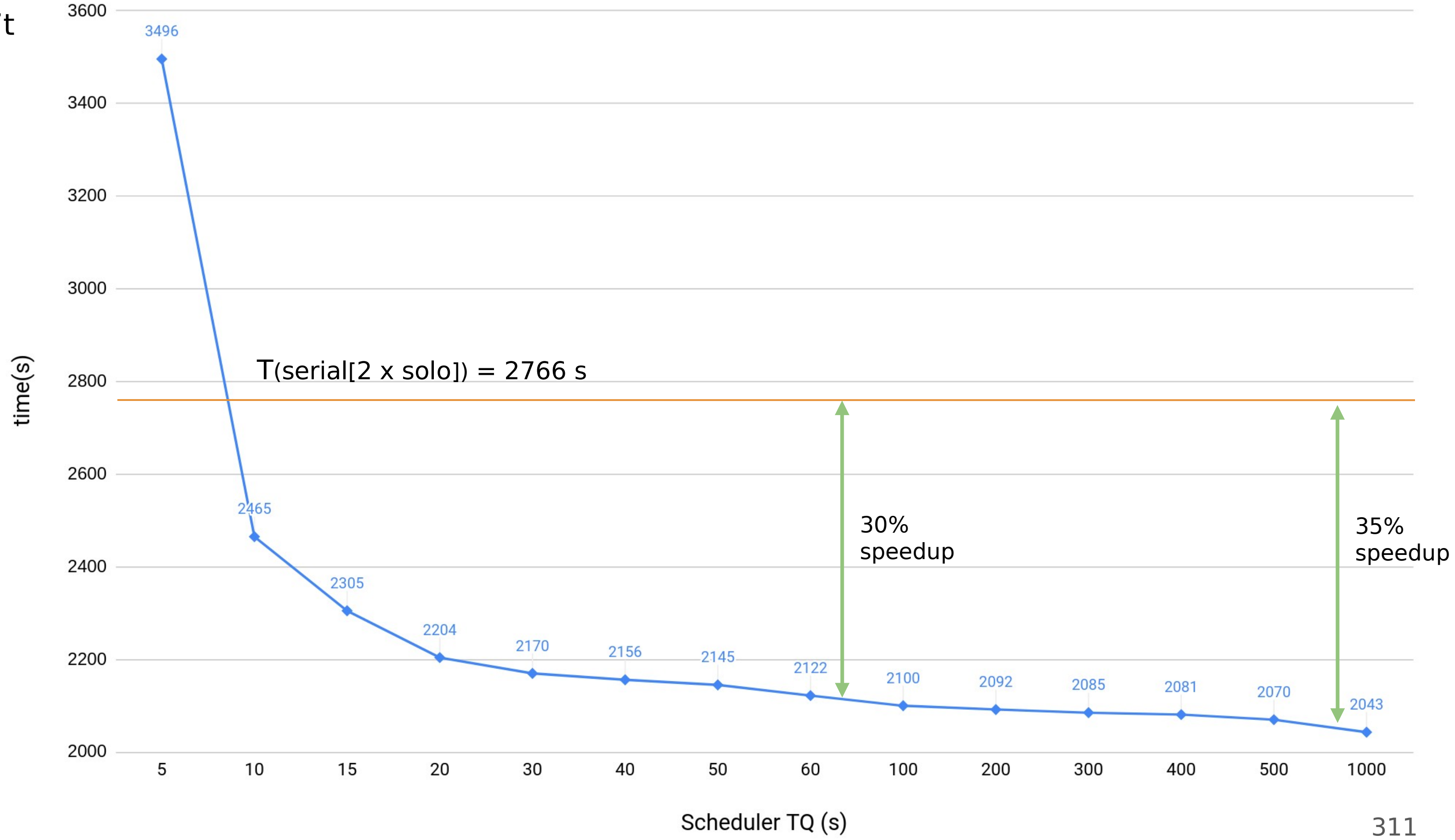
- Kubeshare can’t run 2 x big !



Results for big_50

T(libnvshare, no anti-thrash) = 11757 s

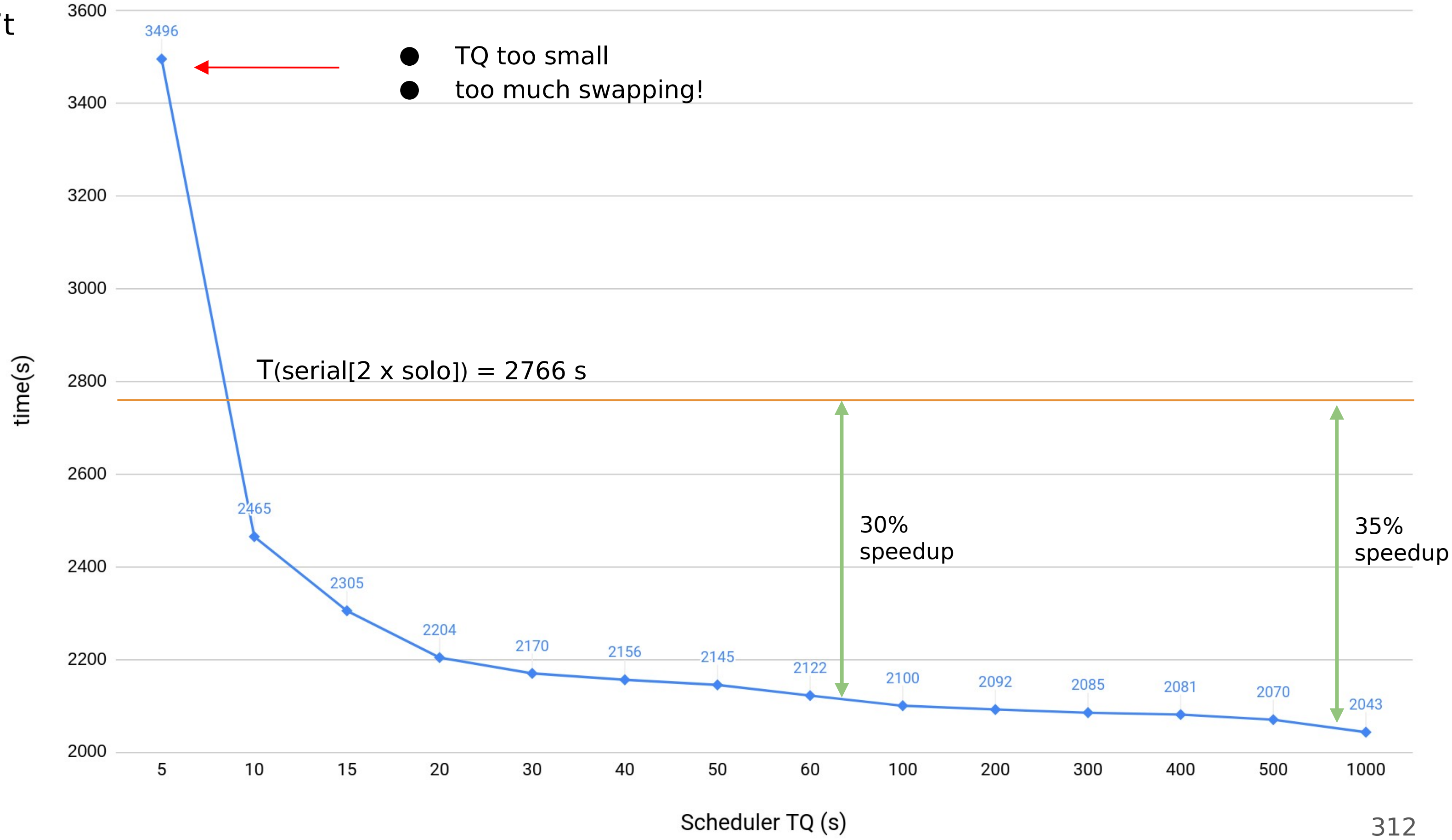
- Kubeshare can’t run 2 x big !



Results for big_50

T(libnvshare, no anti-thrash) = 11757 s

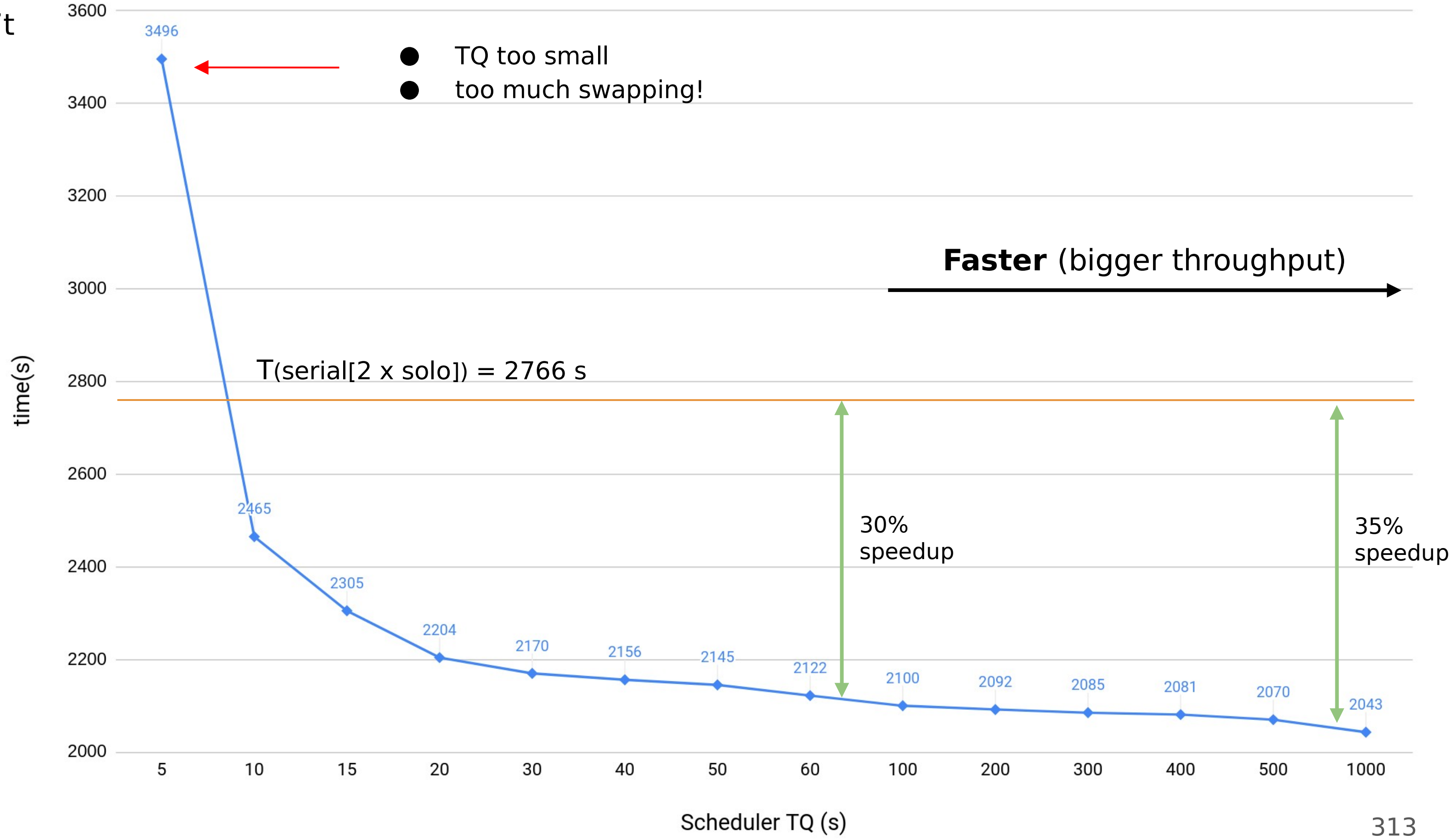
- Kubeshare can't run 2 x big !



Results for big_50

T(libnvshare, no anti-thrash) = 11757 s

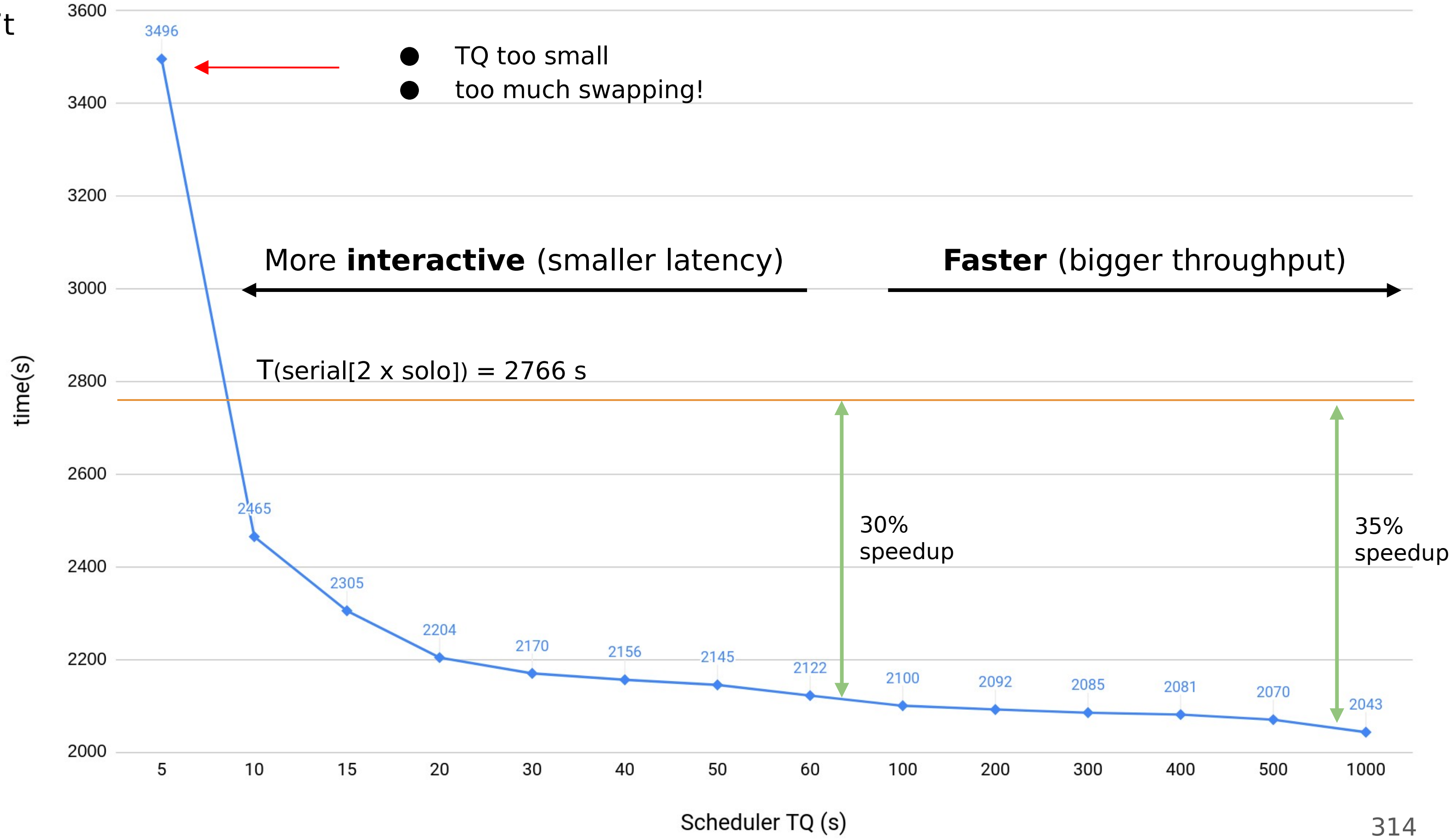
- Kubeshare can't run 2 x big !



Results for big_50

T(libnvshare, no anti-thrash) = 11757 s

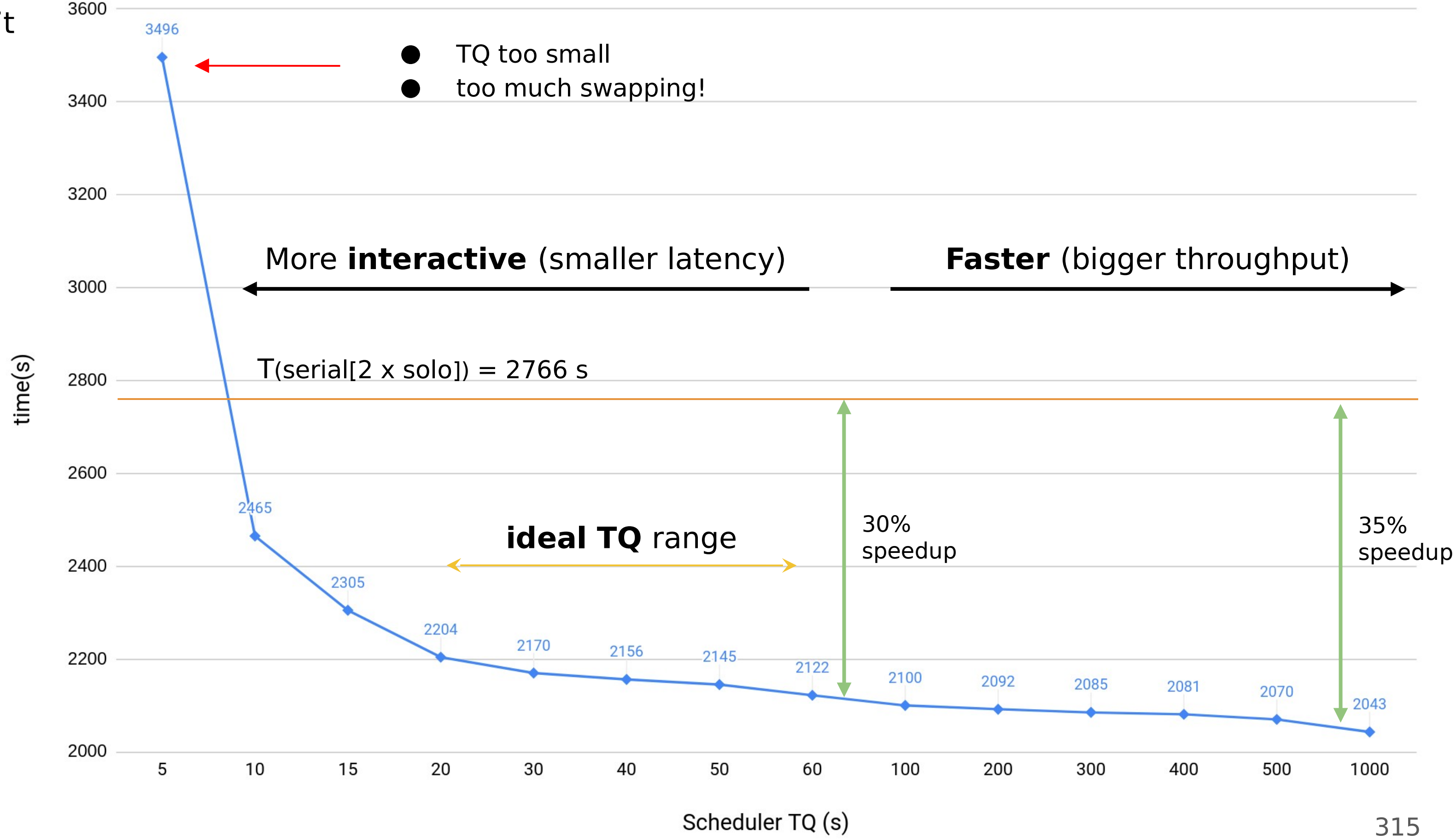
- Kubeshare can't run 2 x big !



Results for big_50

T(libnvshare, no anti-thrash) = 11757 s

- Kubeshare can't run 2 x big !



A new state of the art

A new state of the art

- # of co-located processes limited by system RAM

A new state of the art

- # of co-located processes limited by system RAM
- each process can use the whole GPU memory

A new state of the art

- # of co-located processes limited by system RAM
- each process can use the whole GPU memory
- GPU utilization increases even for non-interactive jobs

A new state of the art

- # of co-located processes limited by system RAM
- each process can use the whole GPU memory
- GPU utilization increases even for non-interactive jobs
- No code changes to user programs

A new state of the art

- # of co-located processes limited by system RAM
- each process can use the whole GPU memory
- GPU utilization increases even for non-interactive jobs
- No code changes to user programs
- Sensible default policies:
  - anti-thrashing scheduler **always on**
  - default TQ = 20 sec

Future Work

- Deploy to production environments
- Support multiple GPUs per node
- Create heuristics (e.g. PCIe traffic) to automatically enable/disable the anti-thrashing mechanism
- Publish our work

- Later on: Intra-node job migration from one GPU to another
- Later on: Inter-node GPU job migration (Checkpoint-restart)

Thank you!

Questions?