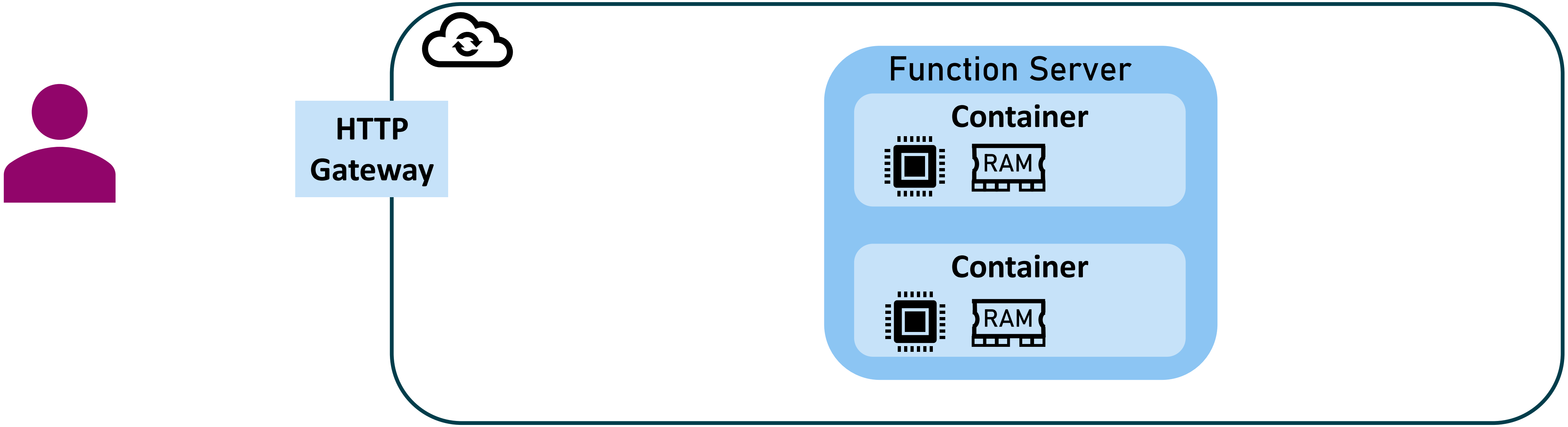
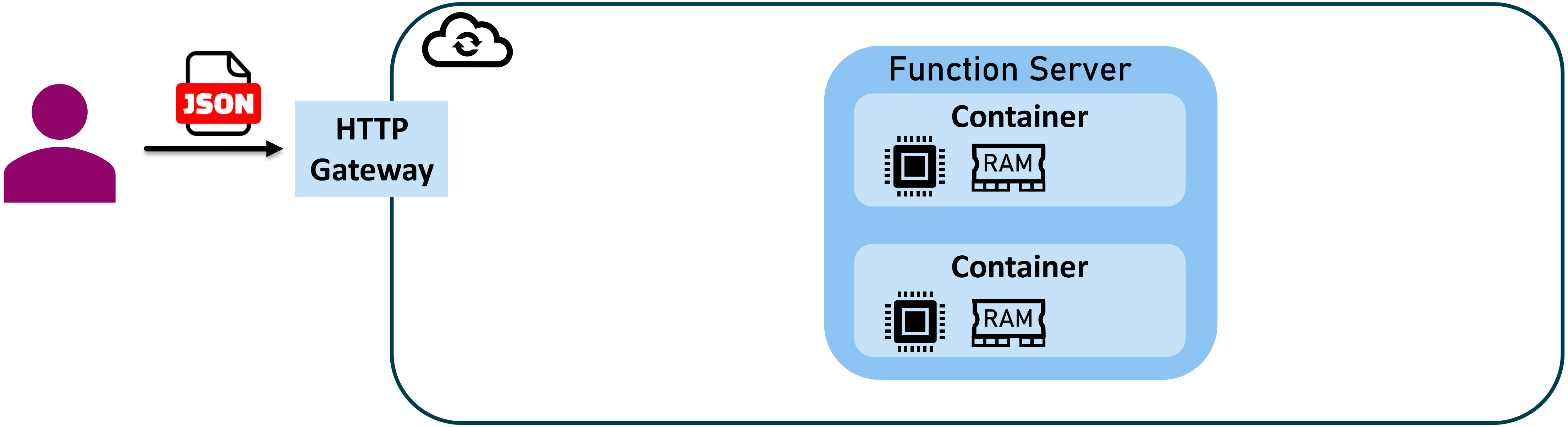
Marcin Copik, Alexandru Calotoiu, Pengyu Zhou, Lukas Tobler, Torsten Hoefler

# MIGnificient: Fast, Isolated, and GPU-Enabled Serverless Functions

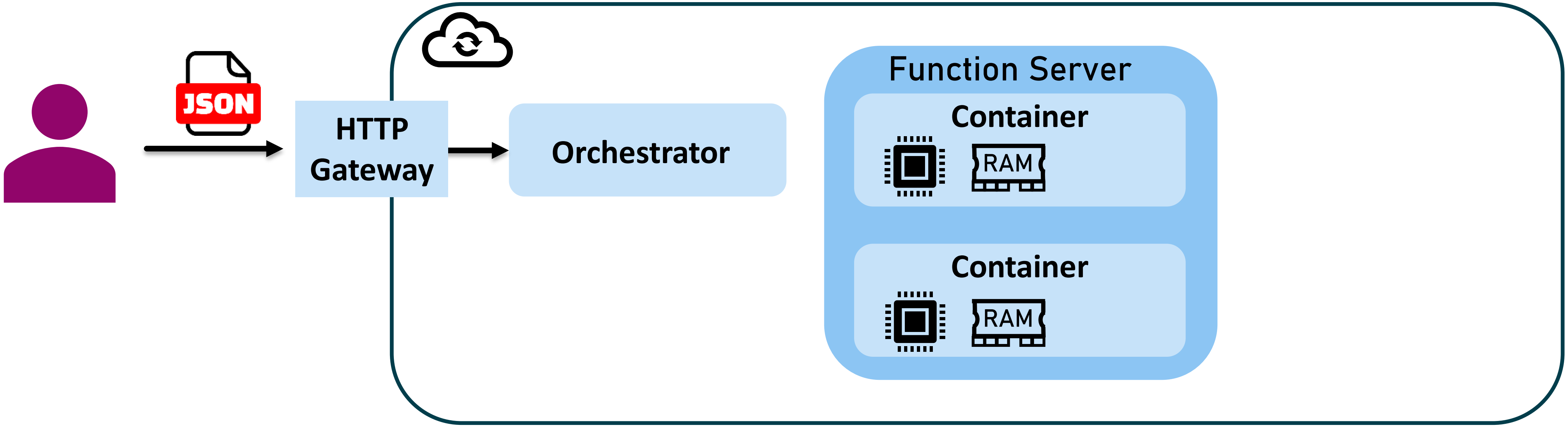
# From Simple Functions to HPC Serverless



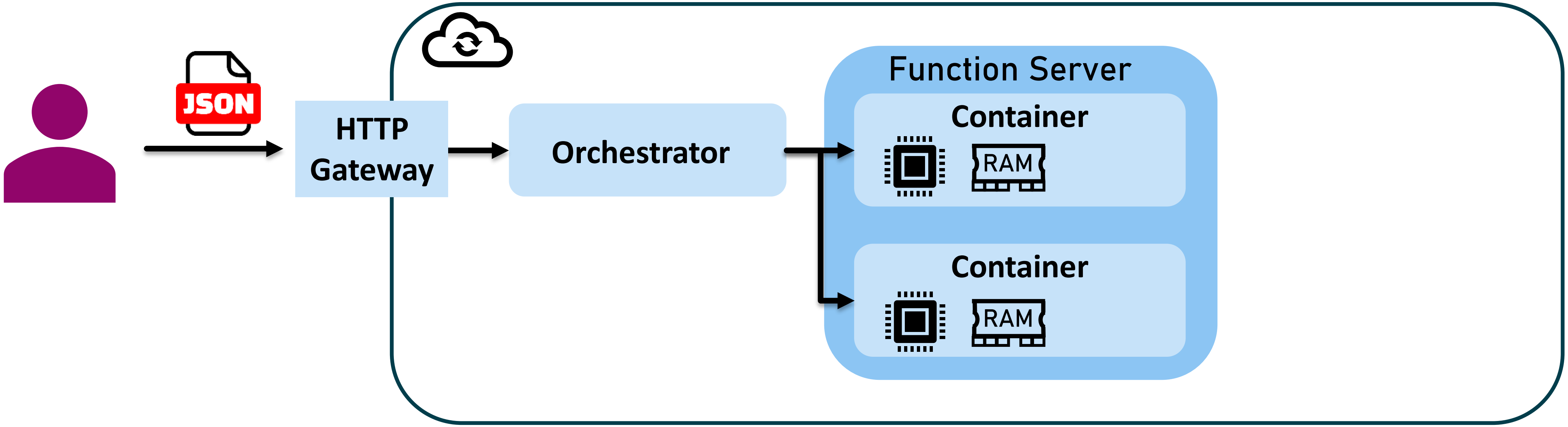
# From Simple Functions to HPC Serverless



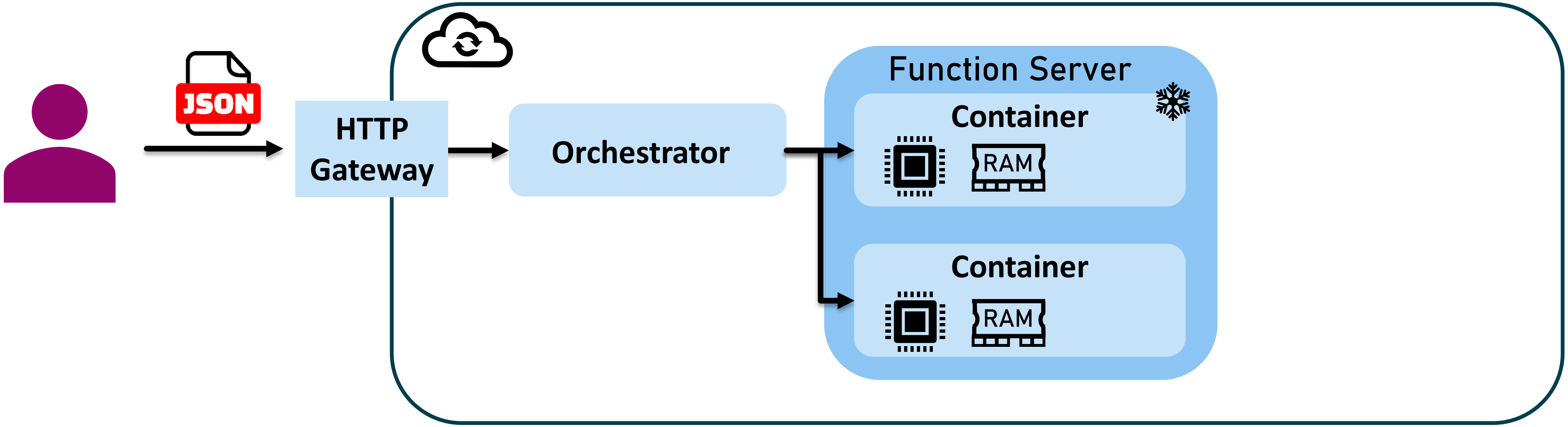
# From Simple Functions to HPC Serverless



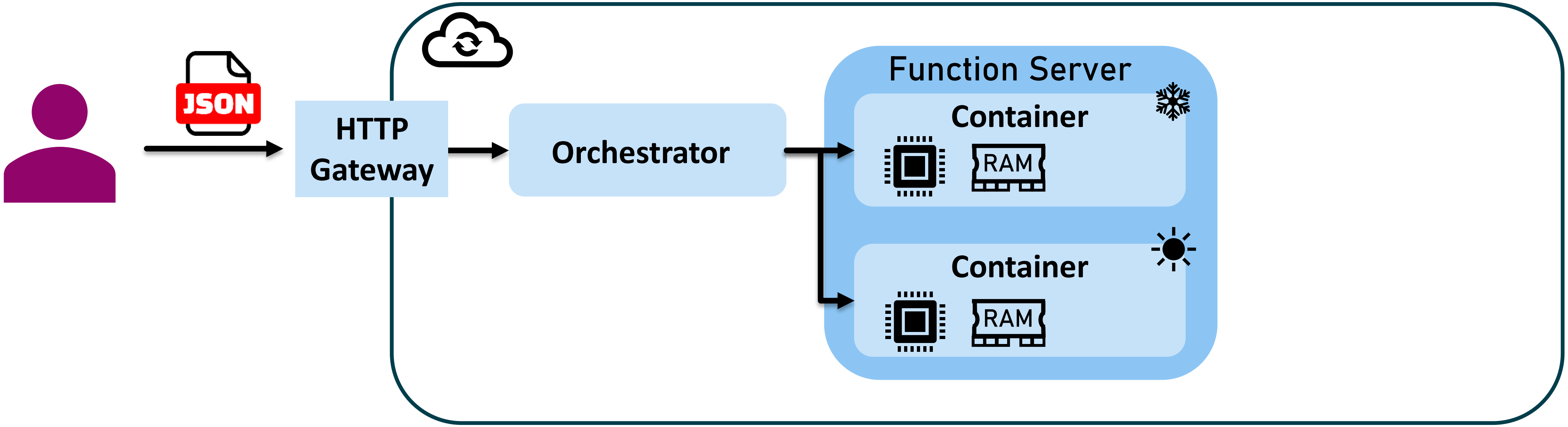
# From Simple Functions to HPC Serverless



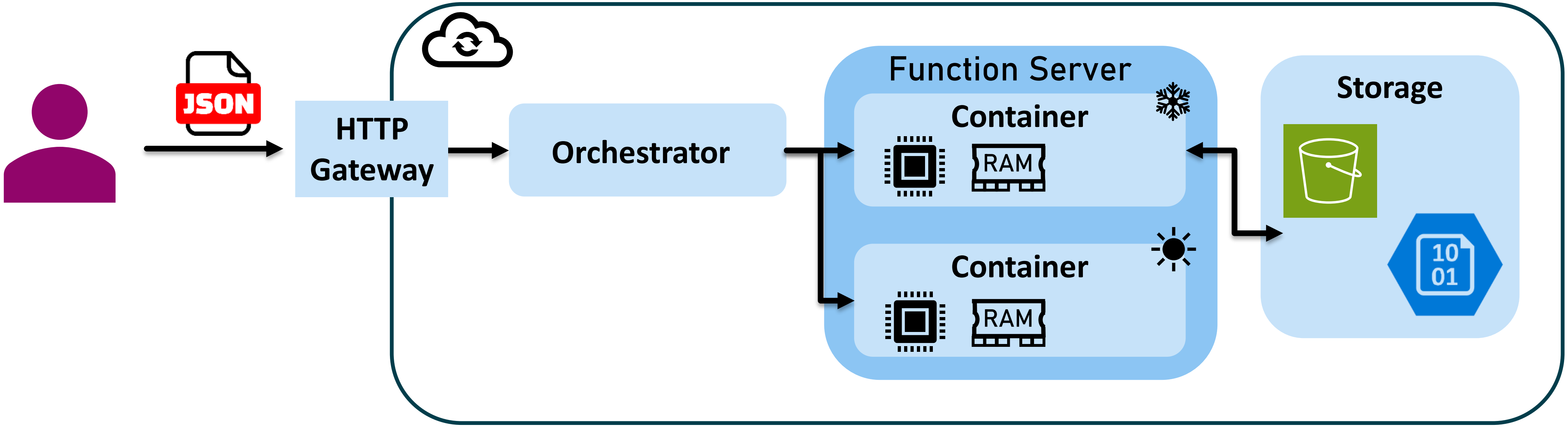
# From Simple Functions to HPC Serverless



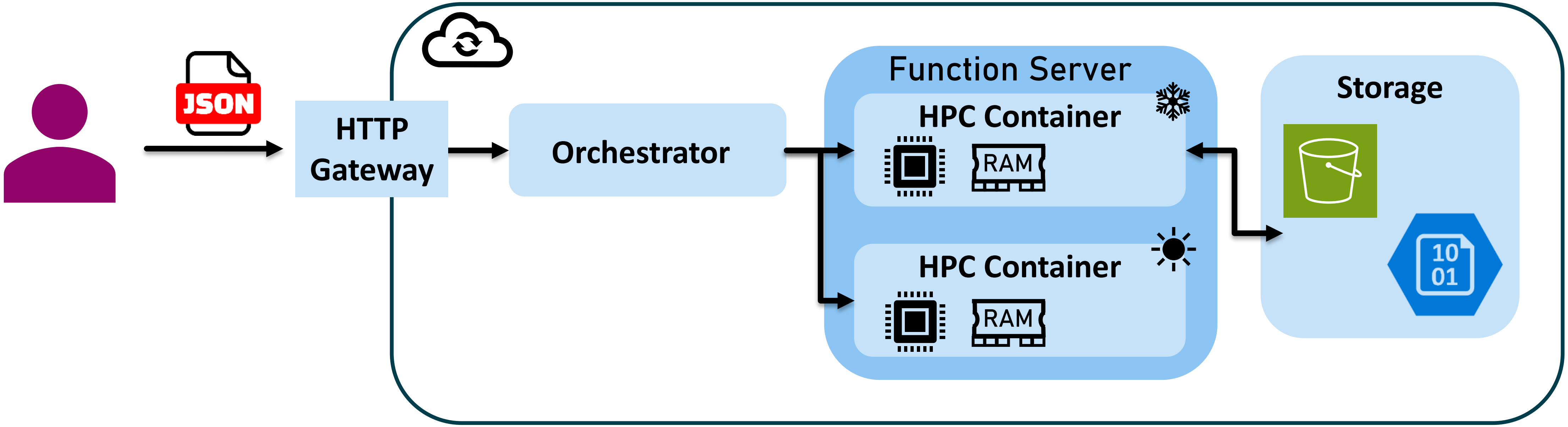
# From Simple Functions to HPC Serverless



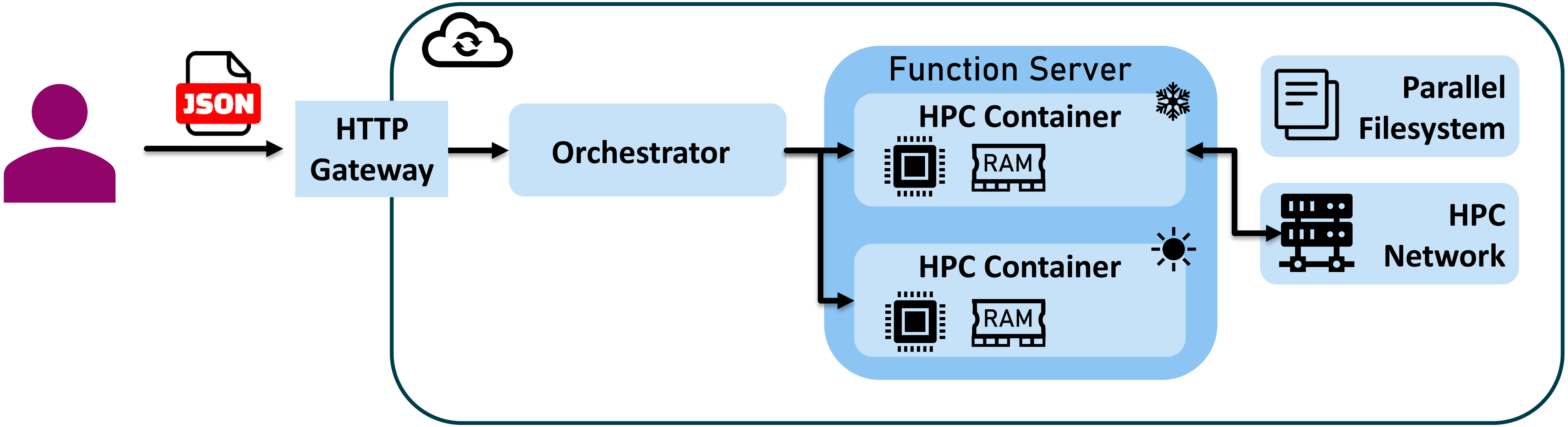
# From Simple Functions to HPC Serverless



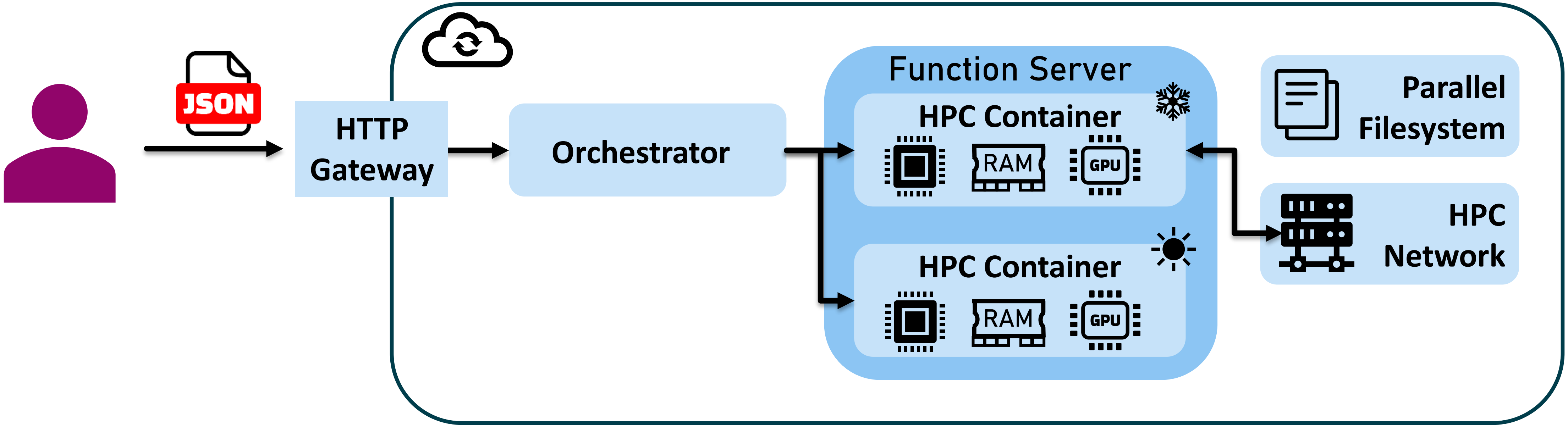
# From Simple Functions to HPC Serverless



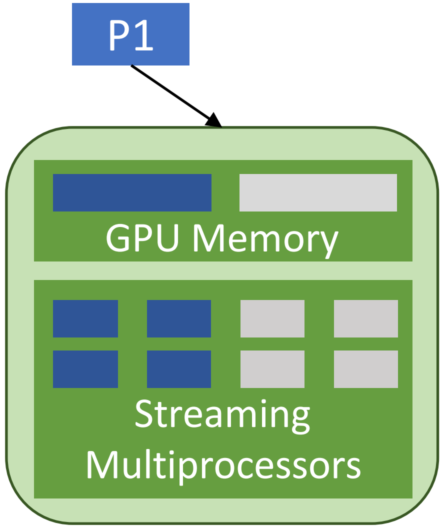
# From Simple Functions to HPC Serverless



# From Simple Functions to HPC Serverless

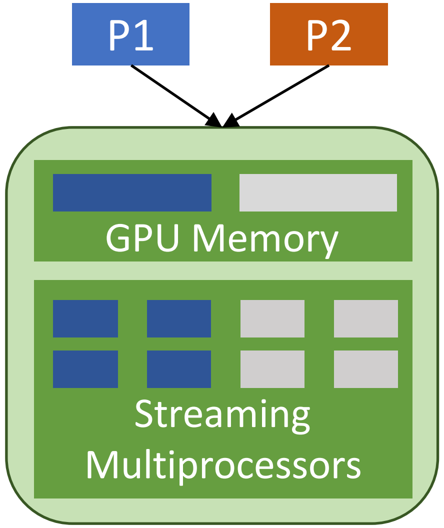


# Time Sharing GPU between Functions



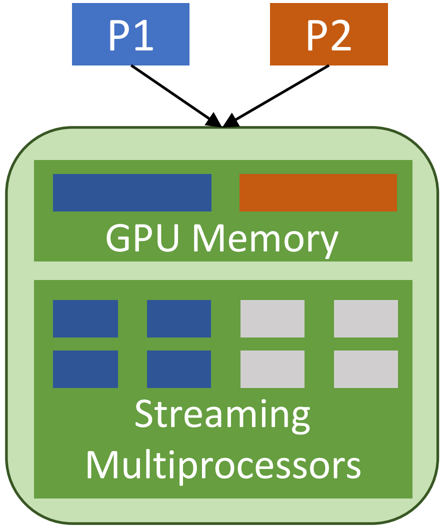

**Time slicing between processes.**

# Time Sharing GPU between Functions



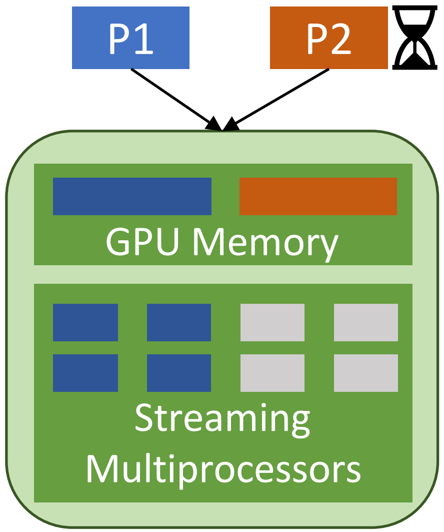

**Time slicing between processes.**

# Time Sharing GPU between Functions



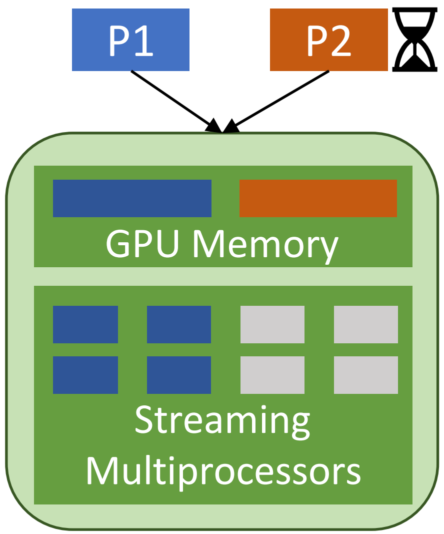

**Time slicing between processes.**

# Time Sharing GPU between Functions



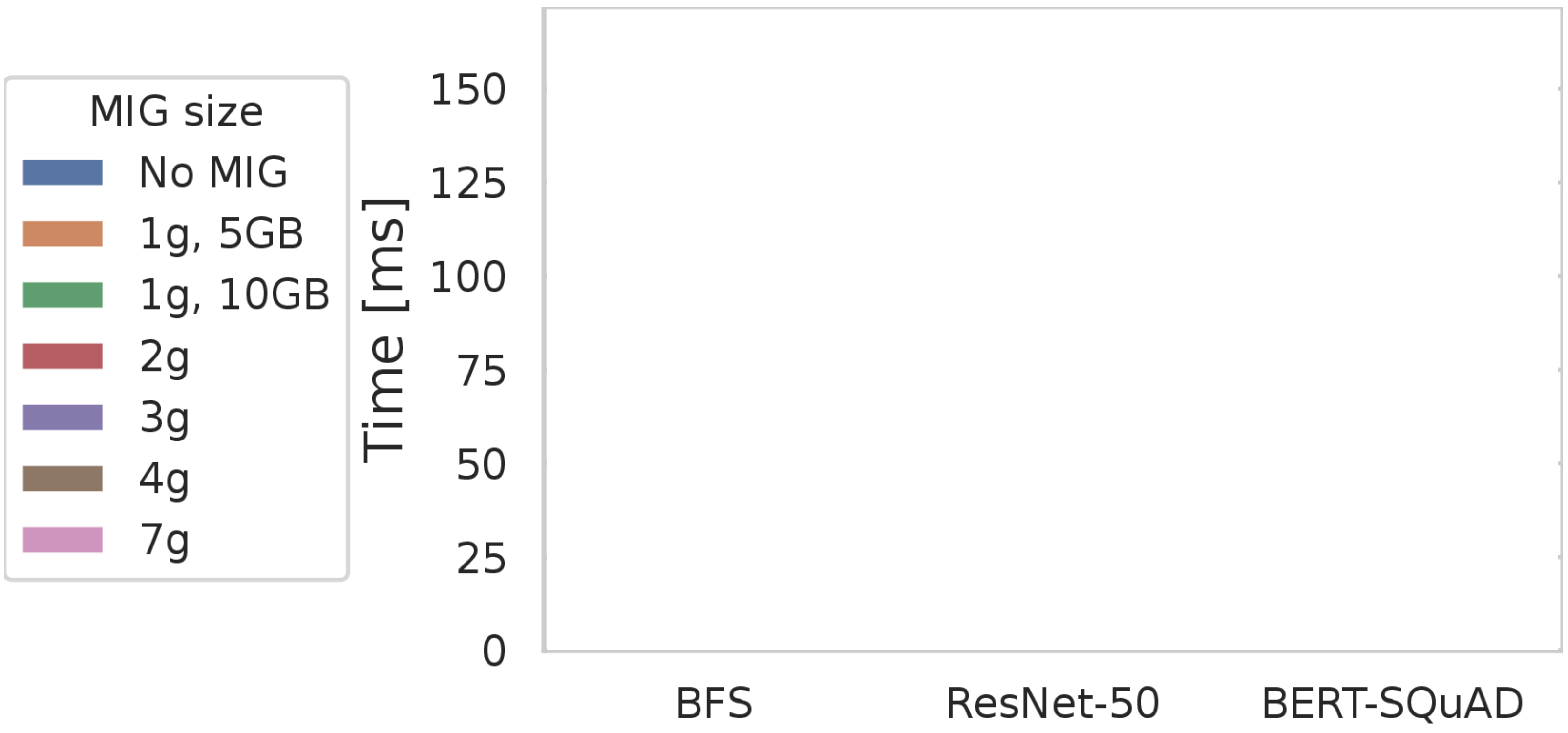

**Time slicing between processes.**

# Time Sharing GPU between Functions

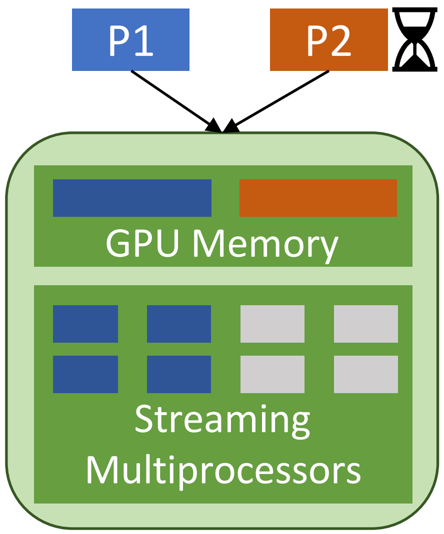



Time slicing between processes.

**Runtime on different partition sizes of A100 GPU.**

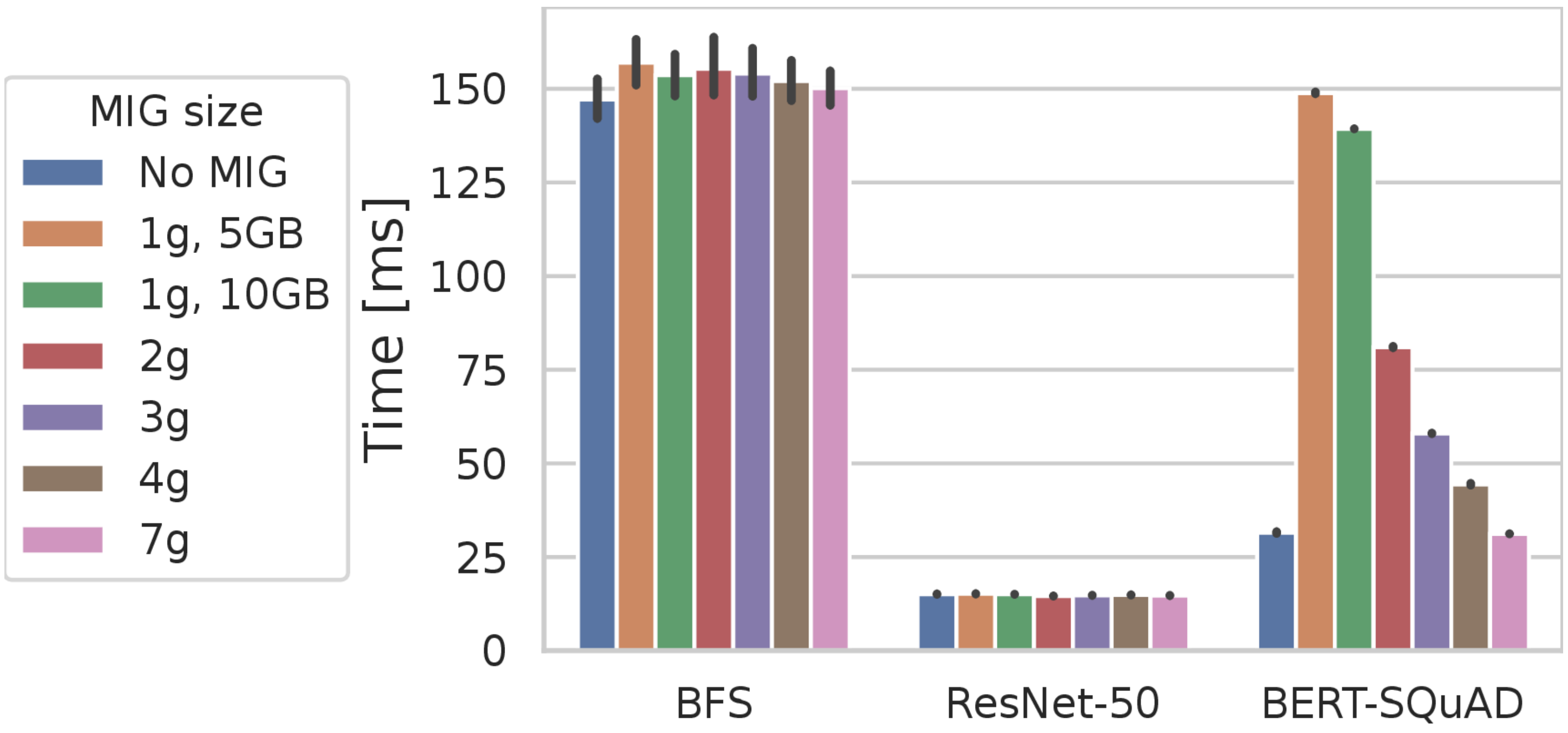


# Time Sharing GPU between Functions

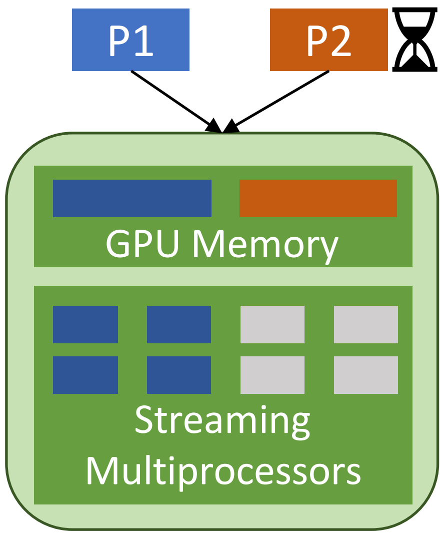



**Time slicing between processes.**

**Runtime on different partition sizes of A100 GPU.**

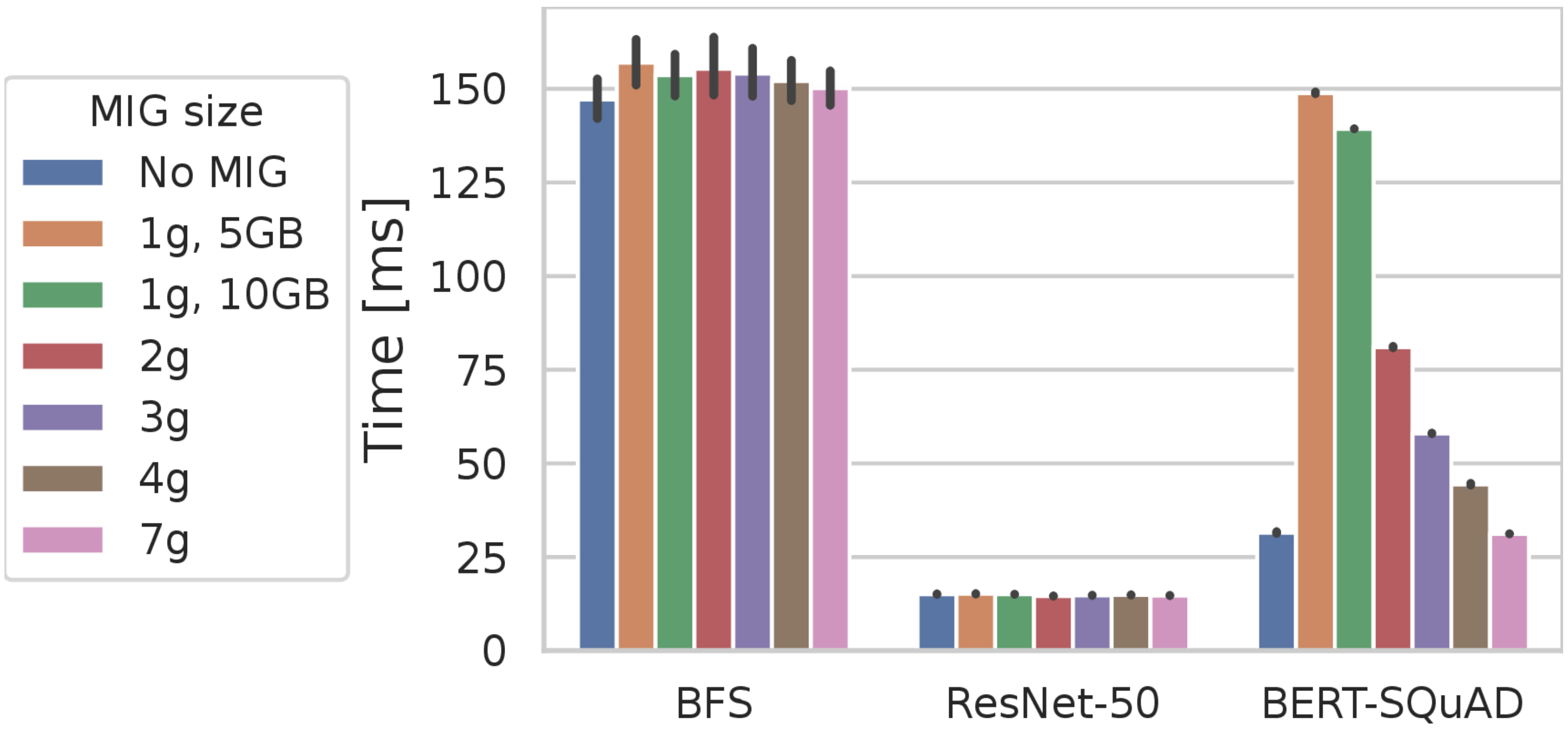


# Time Sharing GPU between Functions



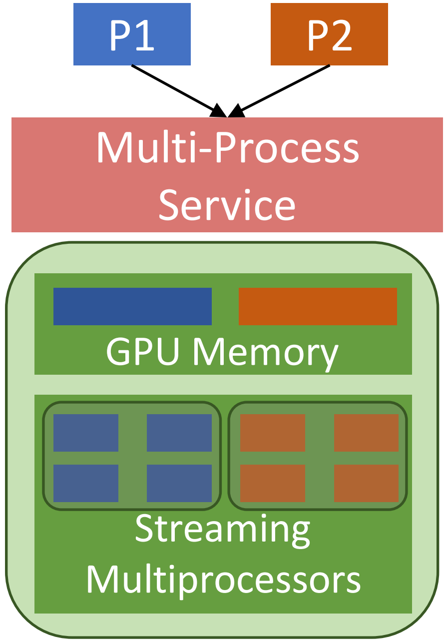

**Time slicing between processes.**

**Runtime on different partition sizes of A100 GPU.**

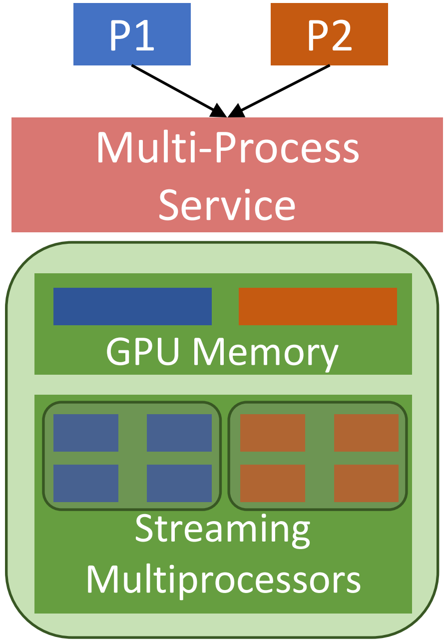



**BERT-SQuAD: 4.77x speedup, but at the cost of 7x more resources!**

# Multi-Process Service (MPS) to the Rescue?

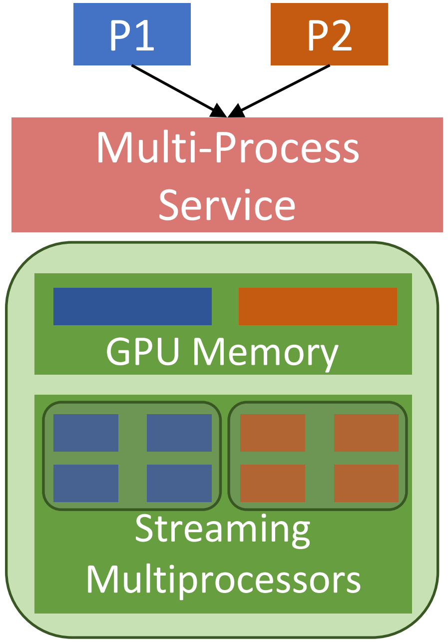


# Multi-Process Service (MPS) to the Rescue?




"MPS is only recommended for running **cooperative processes** effectively acting as a **single application**, such as multiple ranks of the same MPI job, such that the severity of the following memory protection and error containment limitations is acceptable."

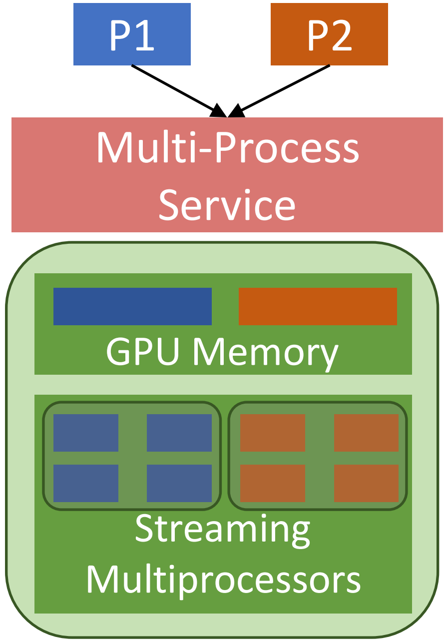
# Multi-Process Service (MPS) to the Rescue?




"MPS is only recommended for running **cooperative processes** effectively acting as a **single application**, such as multiple ranks of the same MPI job, such that the severity of the following memory protection and error containment limitations is acceptable."

⚠ Serverless is multi-tenant and executes arbitrary user code.

# Multi-Process Service (MPS) to the Rescue?

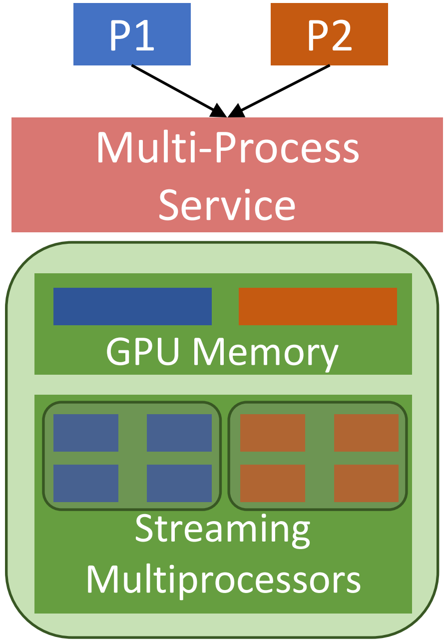



"MPS is only recommended for running **cooperative processes** effectively acting as a **single application**, such as multiple ranks of the same MPI job, such that the severity of the following memory protection and error containment limitations is acceptable."

⚠ Serverless is multi-tenant and executes arbitrary user code.

✖ **Limited security!**
Function can conduct side-channel attack.

# Multi-Process Service (MPS) to the Rescue?



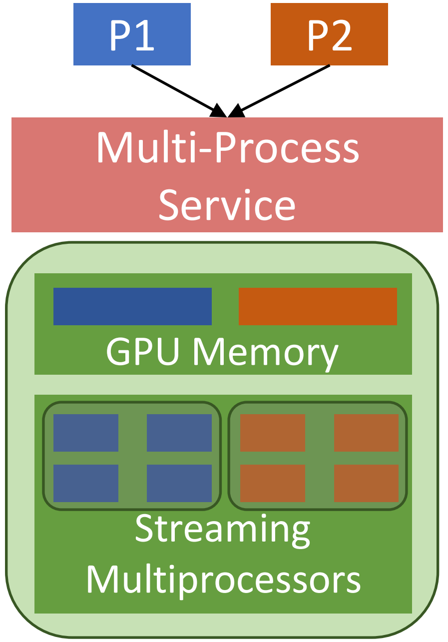

"MPS is only recommended for running **cooperative processes** effectively acting as a **single application**, such as multiple ranks of the same MPI job, such that the severity of the following memory protection and error containment limitations is acceptable."

⚠ Serverless is multi-tenant and executes arbitrary user code.

**Limited security!**
Function can conduct side-channel attack.

**No performance isolation!**
Function can hog the memory bandwidth.

# Multi-Process Service (MPS) to the Rescue?




"MPS is only recommended for running **cooperative processes** effectively acting as a **single application**, such as multiple ranks of the same MPI job, such that the severity of the following memory protection and error containment limitations is acceptable."

⚠ Serverless is multi-tenant and executes arbitrary user code.

✖ **Limited security!**
Function can conduct side-channel attack.

✖ **No performance isolation!**
Function can hog the memory bandwidth.

✖ **No error containment!**
Function can maliciously kill GPU contexts.

# Multi-Instance GPU (MIGs): Building GPU from Blocks

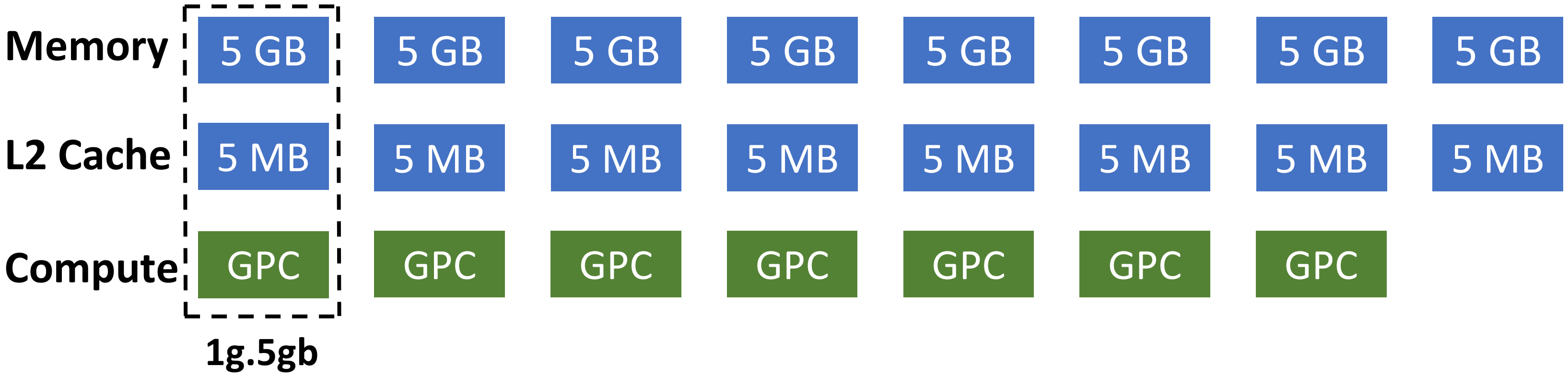
# Multi-Instance GPU (MIGs): Building GPU from Blocks

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Memory** | 5 GB | 5 GB | 5 GB | 5 GB | 5 GB | 5 GB | 5 GB | 5 GB |
| **L2 Cache** | 5 MB | 5 MB | 5 MB | 5 MB | 5 MB | 5 MB | 5 MB | 5 MB |

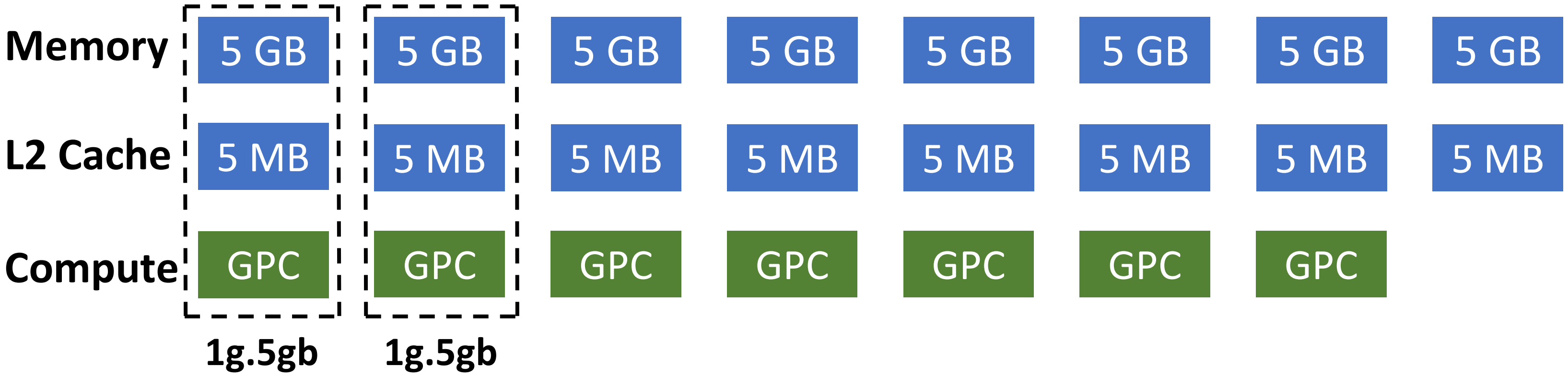
# Multi-Instance GPU (MIGs): Building GPU from Blocks

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Memory** | 5 GB | 5 GB | 5 GB | 5 GB | 5 GB | 5 GB | 5 GB | 5 GB |
| **L2 Cache** | 5 MB | 5 MB | 5 MB | 5 MB | 5 MB | 5 MB | 5 MB | 5 MB |
| **Compute** | GPC | GPC | GPC | GPC | GPC | GPC | GPC | |

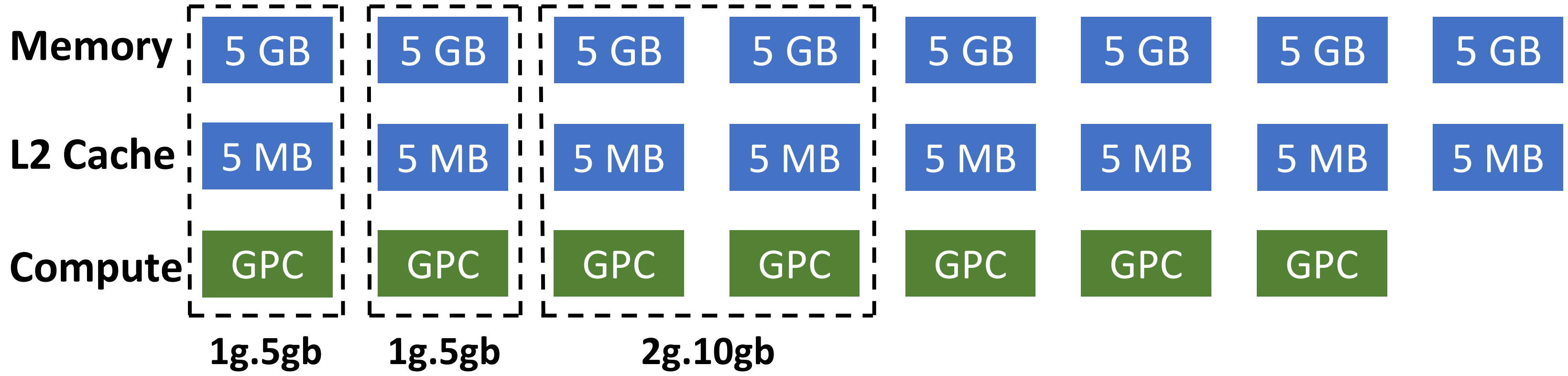
# Multi-Instance GPU (MIGs): Building GPU from Blocks

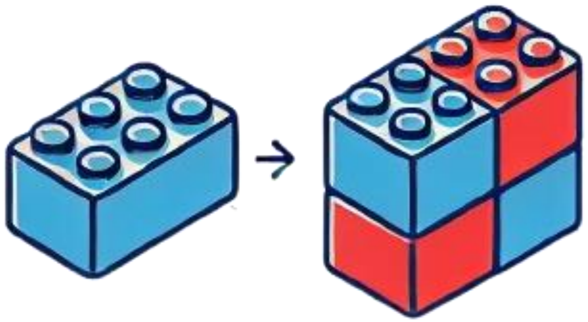


# Multi-Instance GPU (MIGs): Building GPU from Blocks

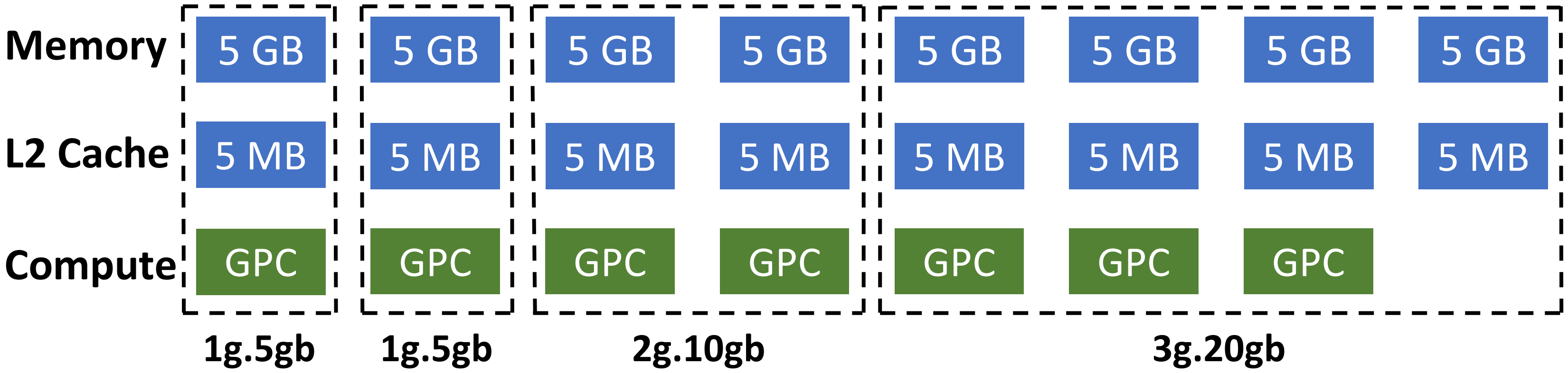


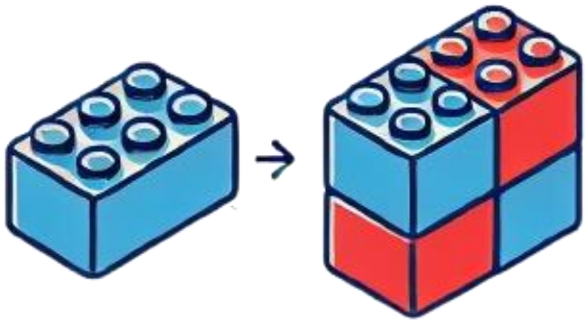

# Multi-Instance GPU (MIGs): Building GPU from Blocks



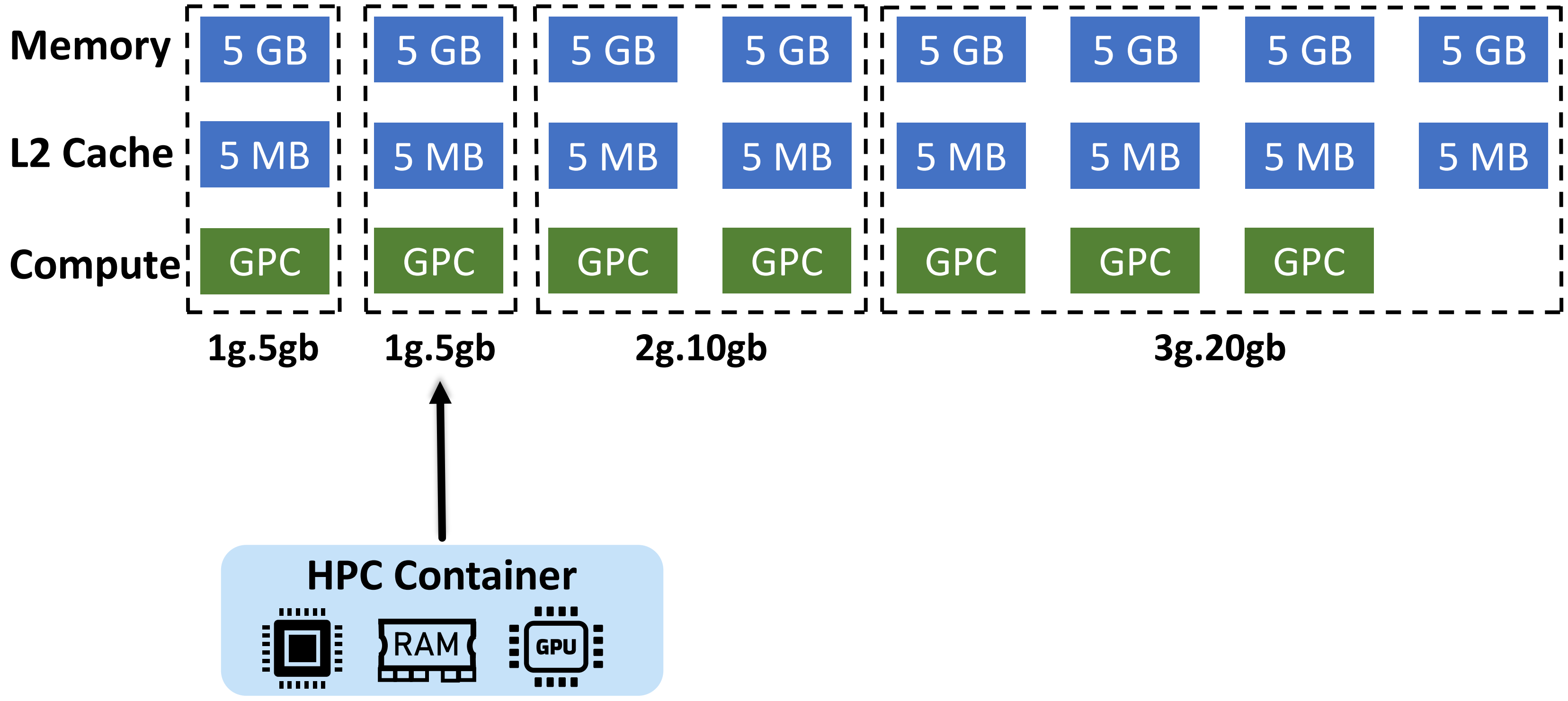



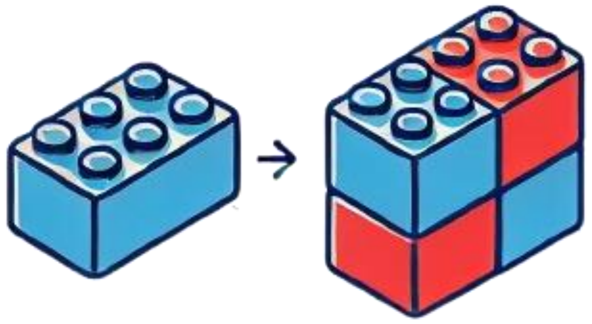
# Multi-Instance GPU (MIGs): Building GPU from Blocks



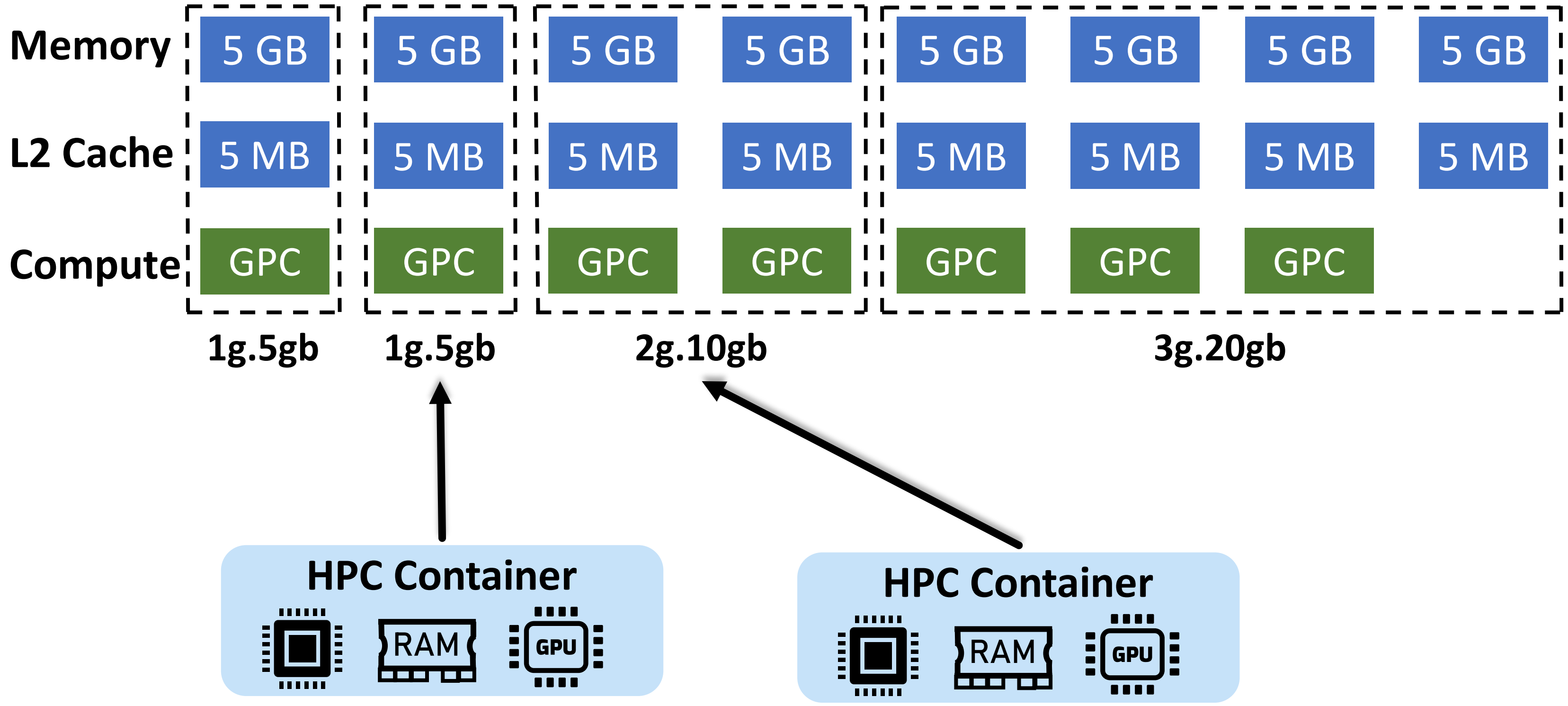



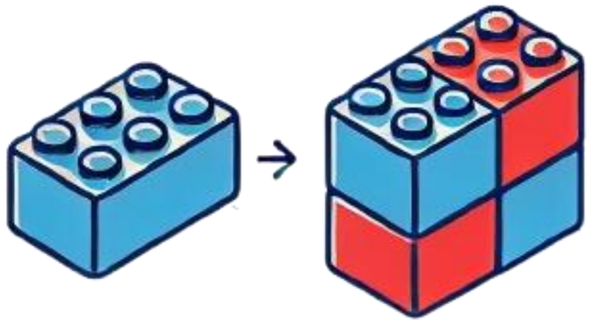
# Multi-Instance GPU (MIGs): Building GPU from Blocks



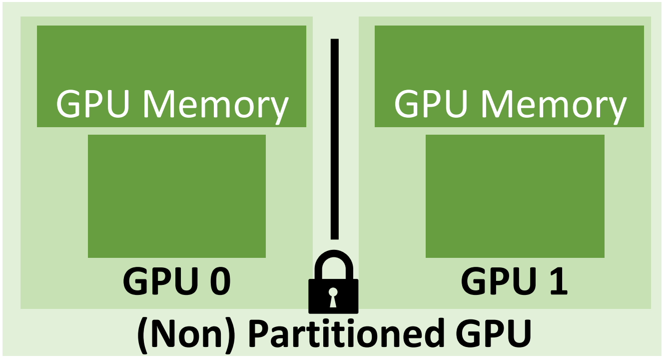
# Multi-Instance GPU (MIGs): Building GPU from Blocks



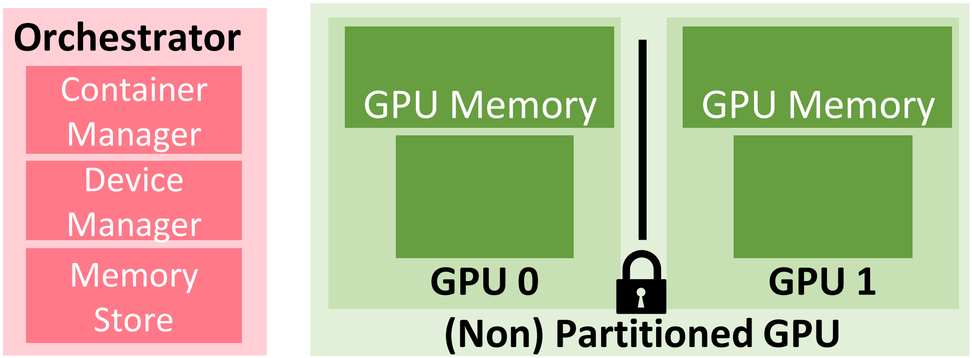



# MIGnificient: Fast and Isolated GPU Functions

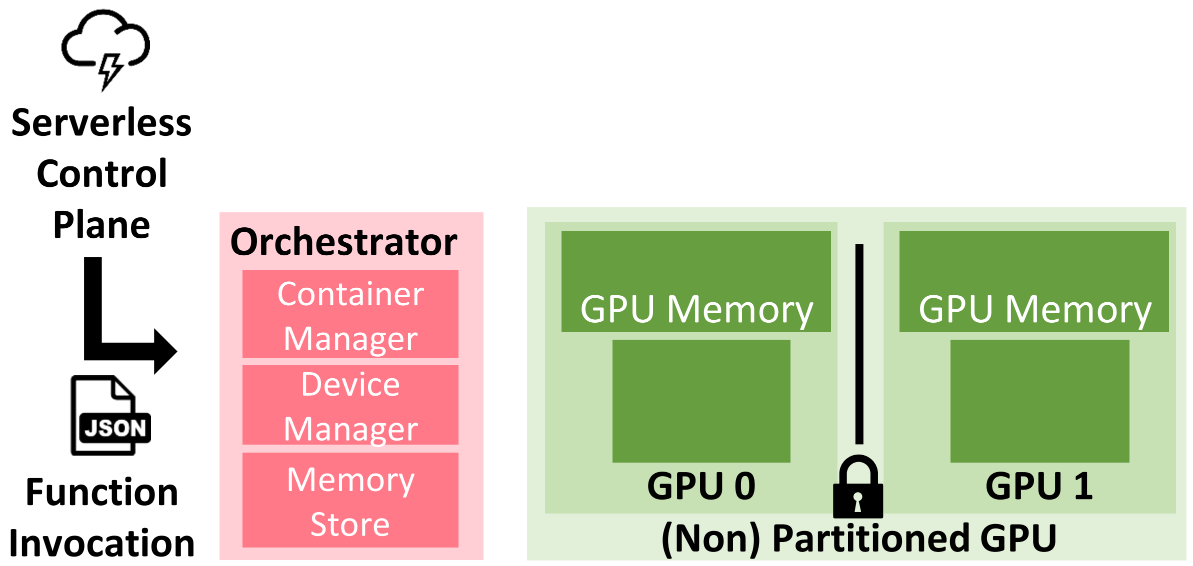
# MIGnificient: Fast and Isolated GPU Functions



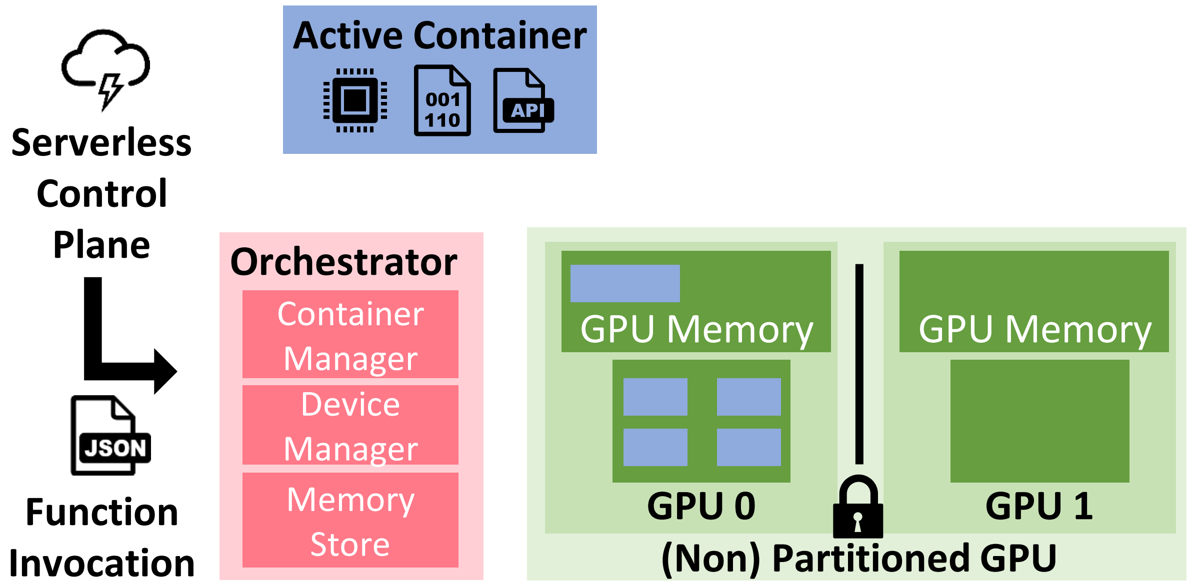
# MIGnificient: Fast and Isolated GPU Functions



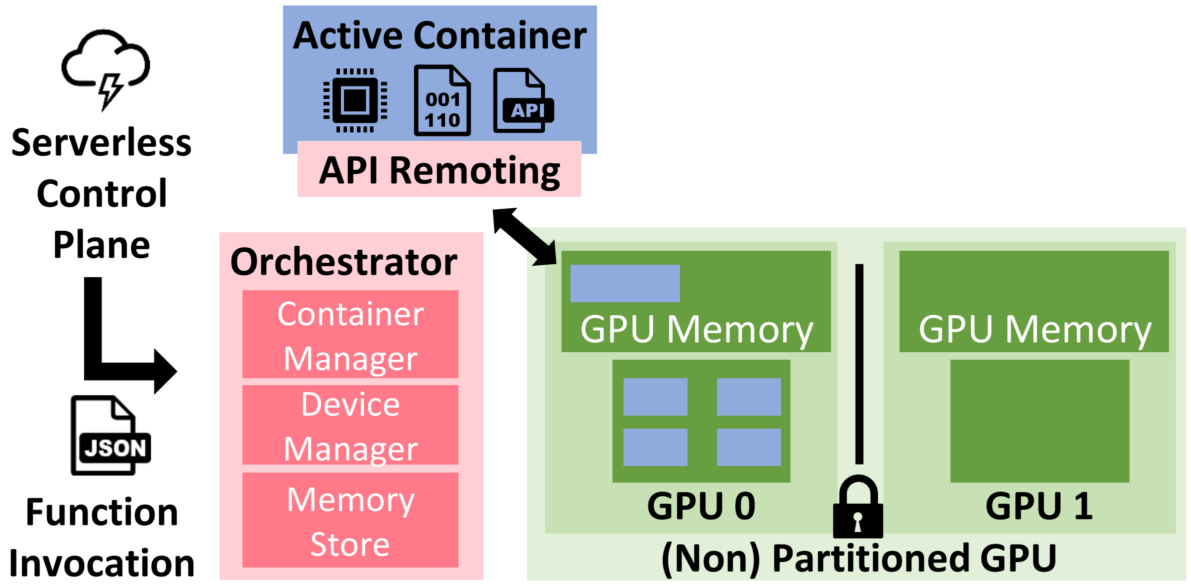
# MIGnificient: Fast and Isolated GPU Functions



# MIGnificient: Fast and Isolated GPU Functions

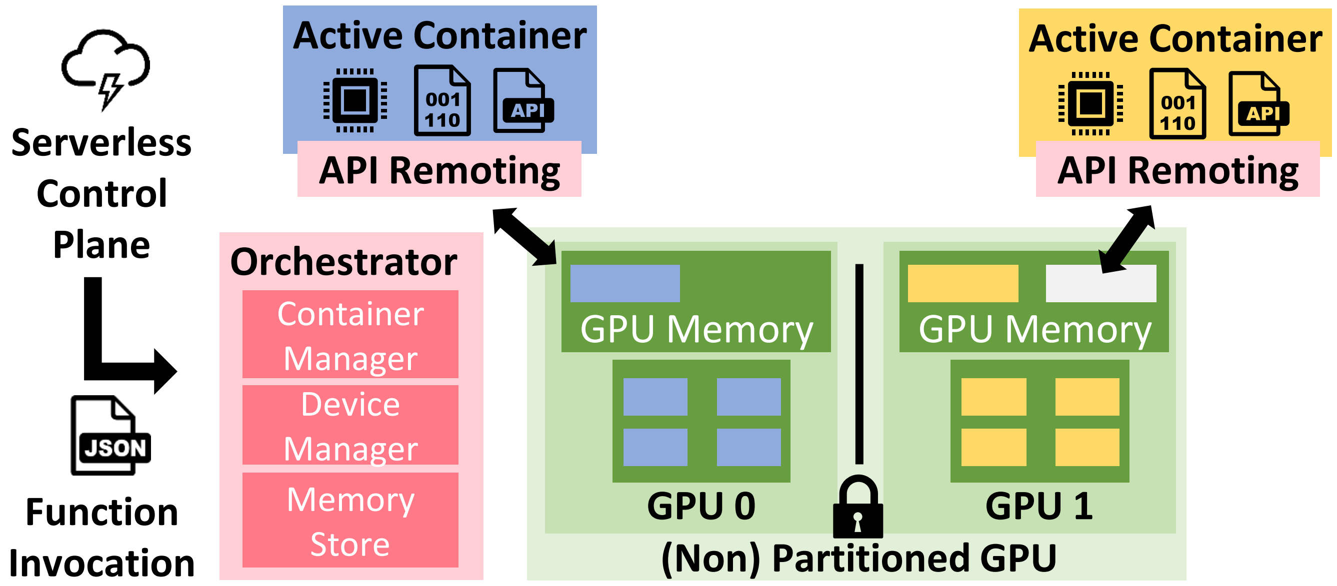


# MIGnificient: Fast and Isolated GPU Functions




API Remoting
over shared memory

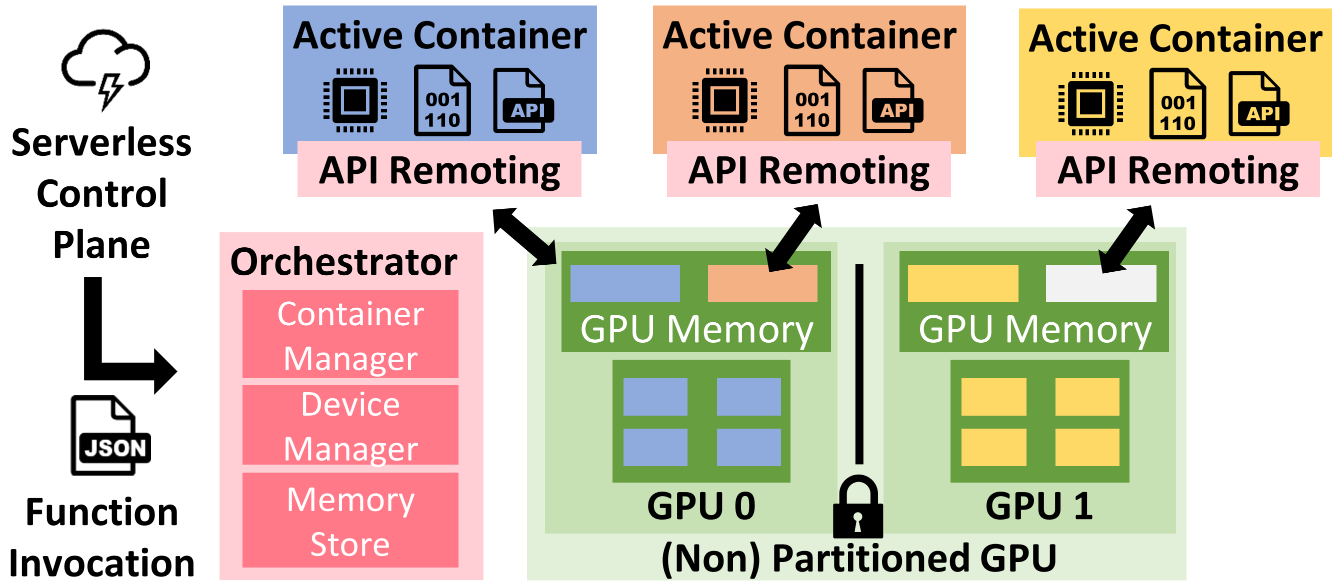
# MIGnificient: Fast and Isolated GPU Functions




API Remoting over shared memory

Concurrent containers on different partitions

# MIGnificient: Fast and Isolated GPU Functions




API Remoting over shared memory

Concurrent containers on different partitions

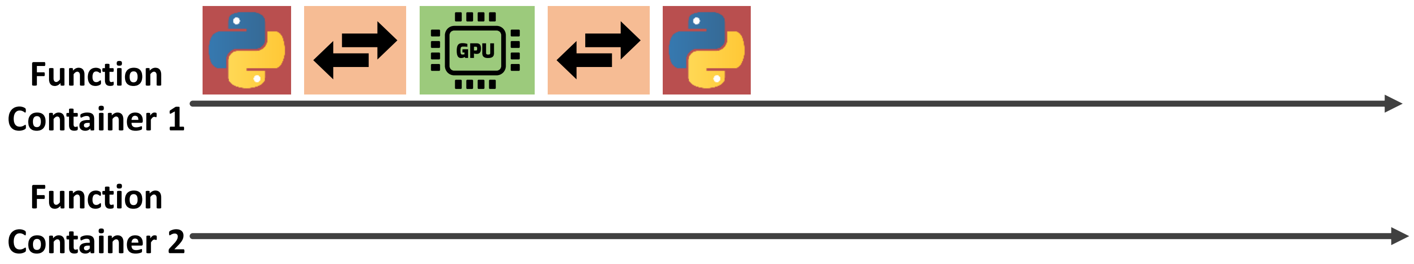
Consecutive containers on the same partition

# Fast Function Switching

## Sequential

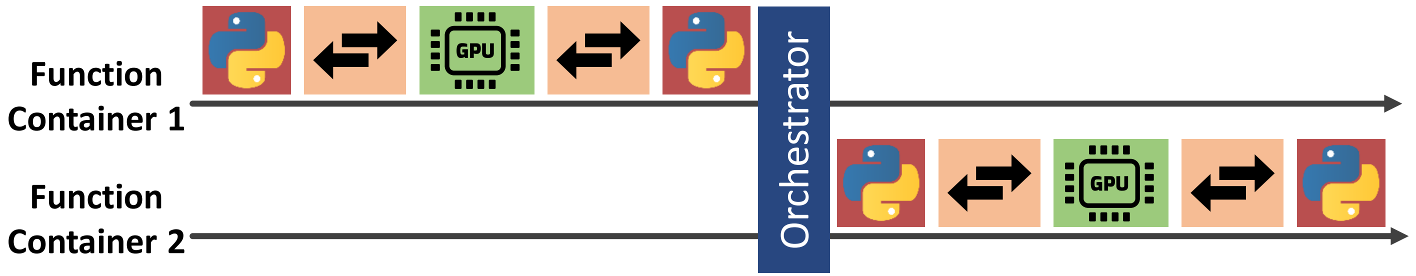
# Fast Function Switching

## Sequential



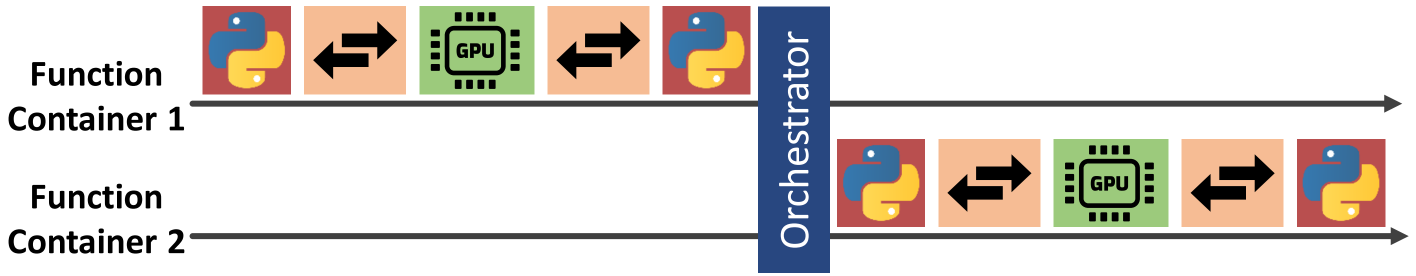
# Fast Function Switching

## Sequential

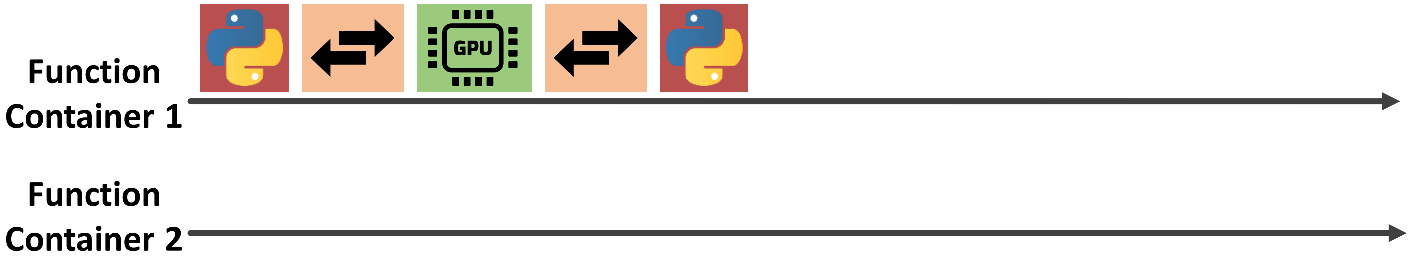


# Fast Function Switching

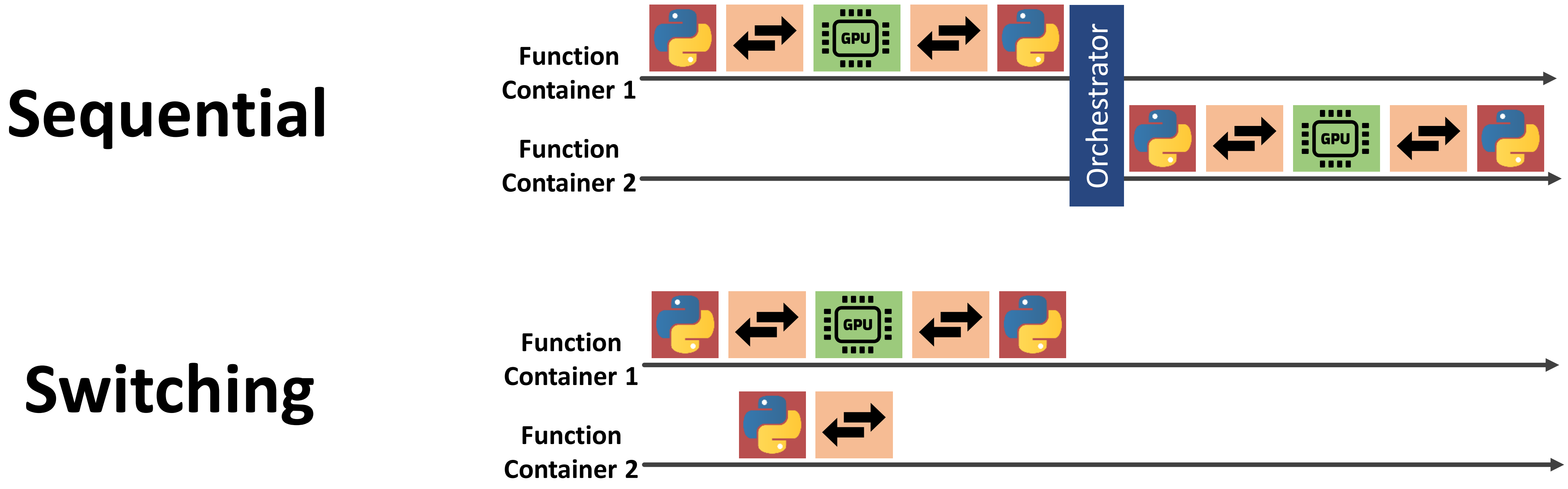
## Sequential




## Switching



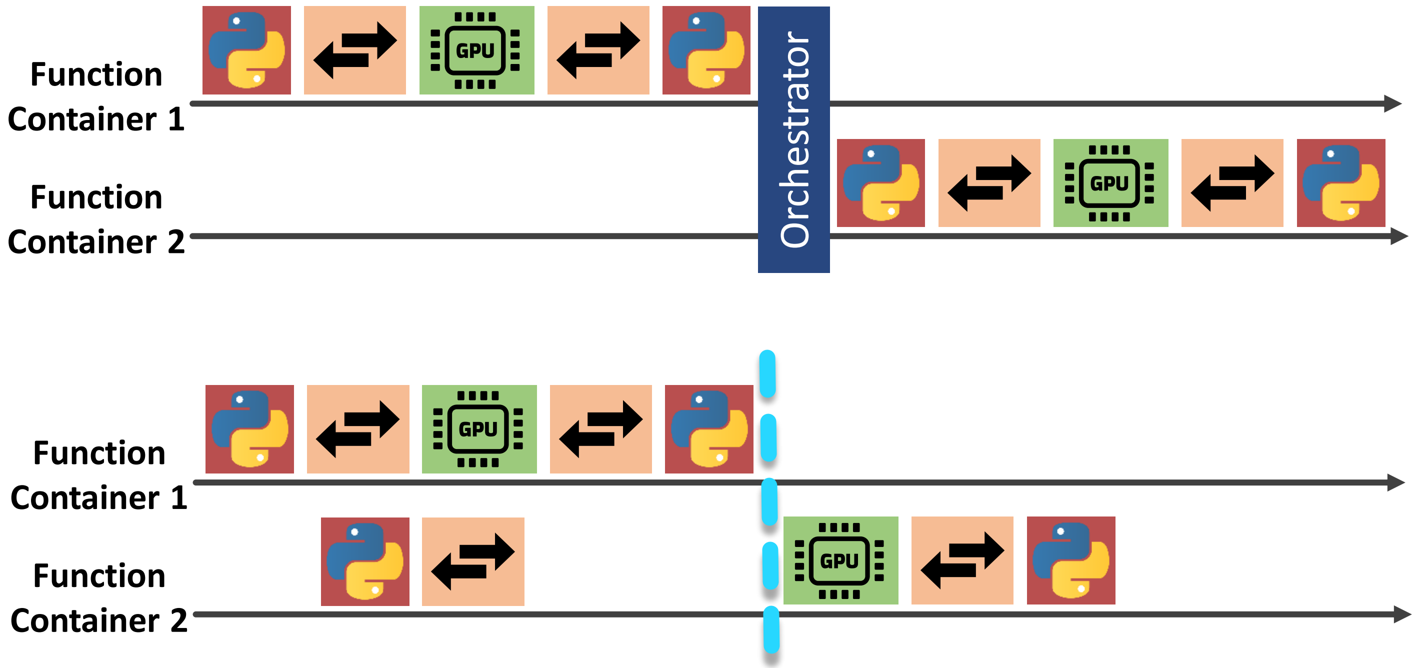
# Fast Function Switching



# Fast Function Switching

## Sequential

## Switching

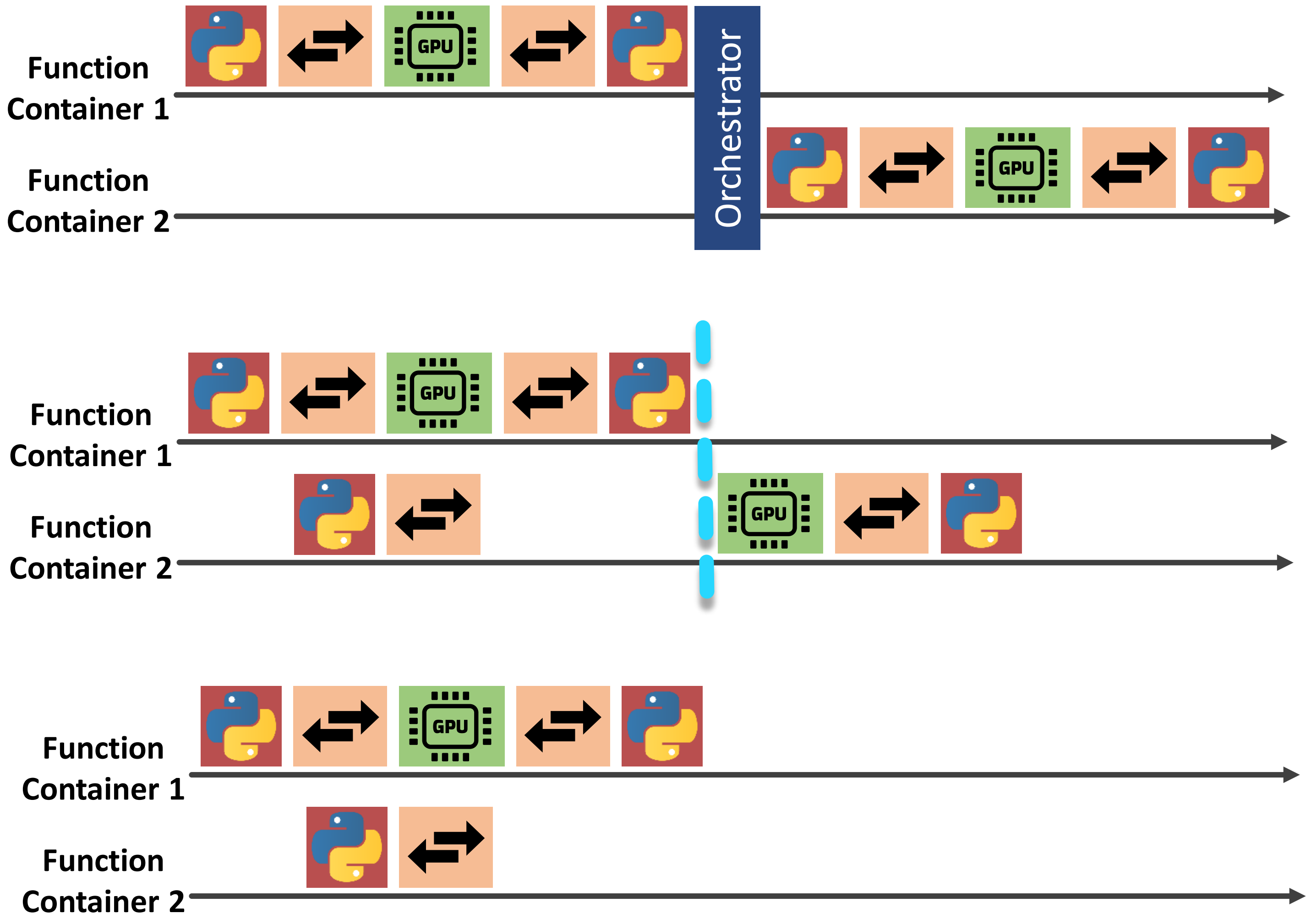


# Fast Function Switching

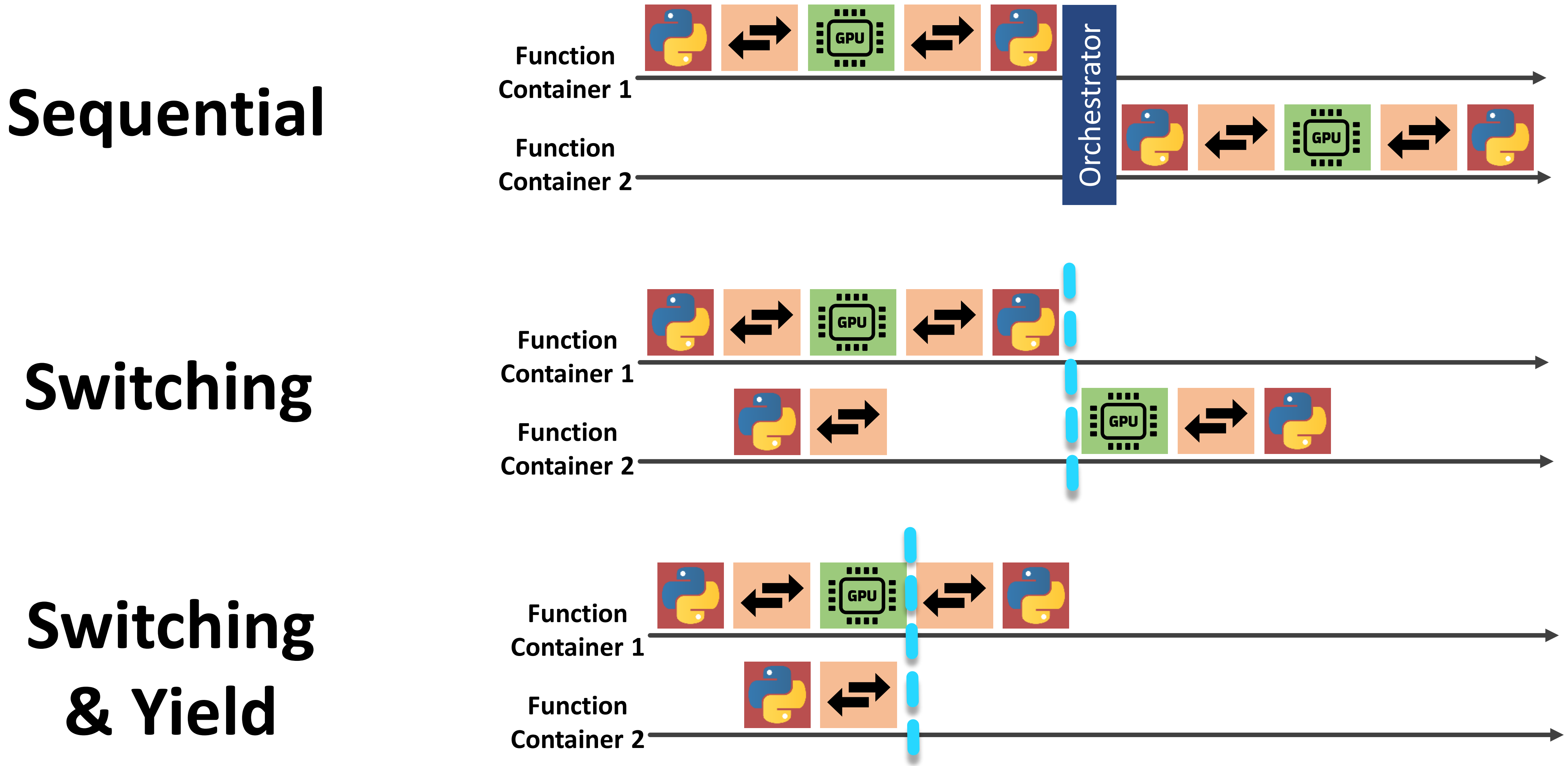
**Sequential**

**Switching**

**Switching & Yield**

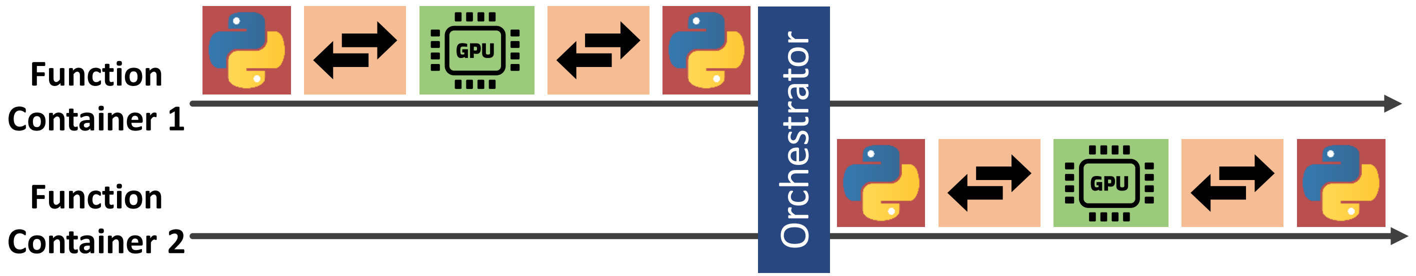


# Fast Function Switching

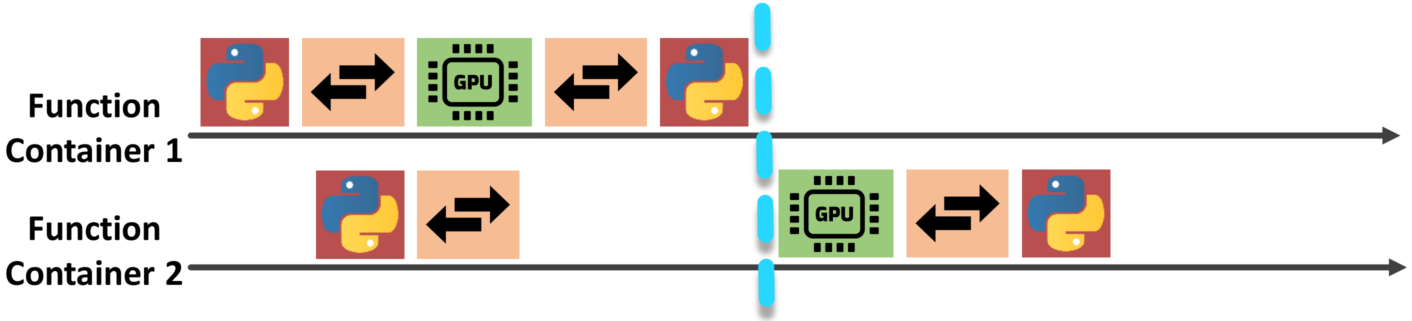


# Fast Function Switching

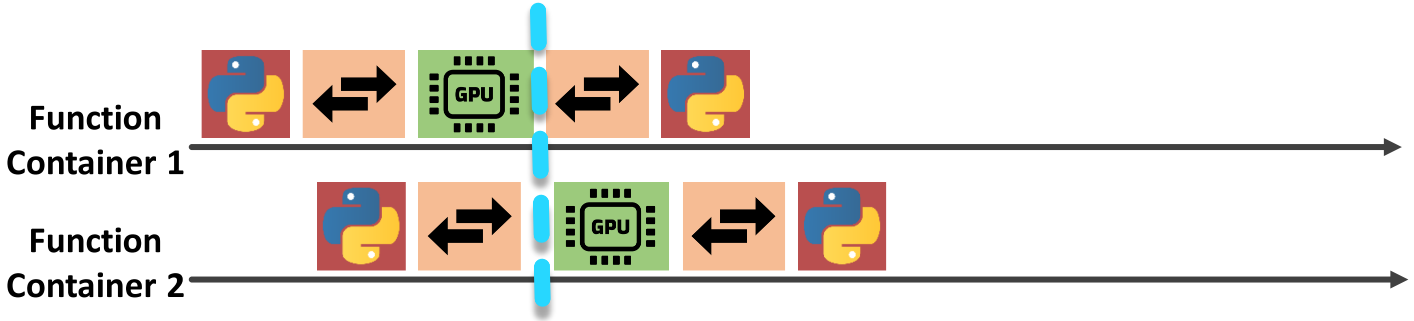
Sequential




Switching




Switching & Yield



# Fast Function Switching

| Benchmark | Native CUDA | MIGnificient | |
|---|---|---|---|
| | Time Sharing | Sequential | Fast Function Switching |
| BFS | 505.3 ± 2.5 | 990.5 ± 112 | 528.8 ± 14.6 |
| hotspot | 92.1 ± 0.8 | 195.1 ± 22.2 | 103 ± 9.9 |
| ResNet-50 | 18 ± 0.3 | 53.3 ± 6 | 27.5 ± 0.7 |
| AlexNet | 15.4 ± 0.5 | 49.2 ± 5.5 | 26.4 ± 0.9 |
| Vgg19 | 23.6 ± 1 | 54.5 ± 6.5 | 27.8 ± 1 |
| BERT-SQuaD | 40.2 ± 2.5 | 65.8 ± 7.5 | 41.4 ± 3.1 |

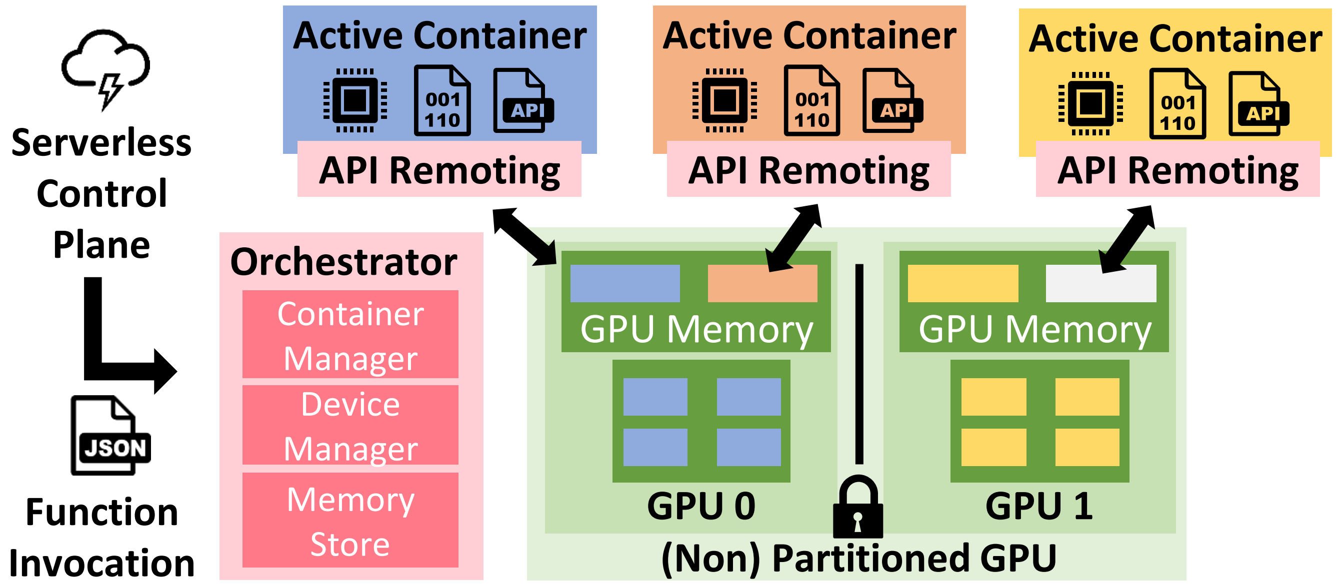
**Two clients sending concurrently in total 10 requests.**
**Unmodified benchmarks without yield.**
**Bare-metal processes on RTX 4070 GPU.**

# Fast Function Switching

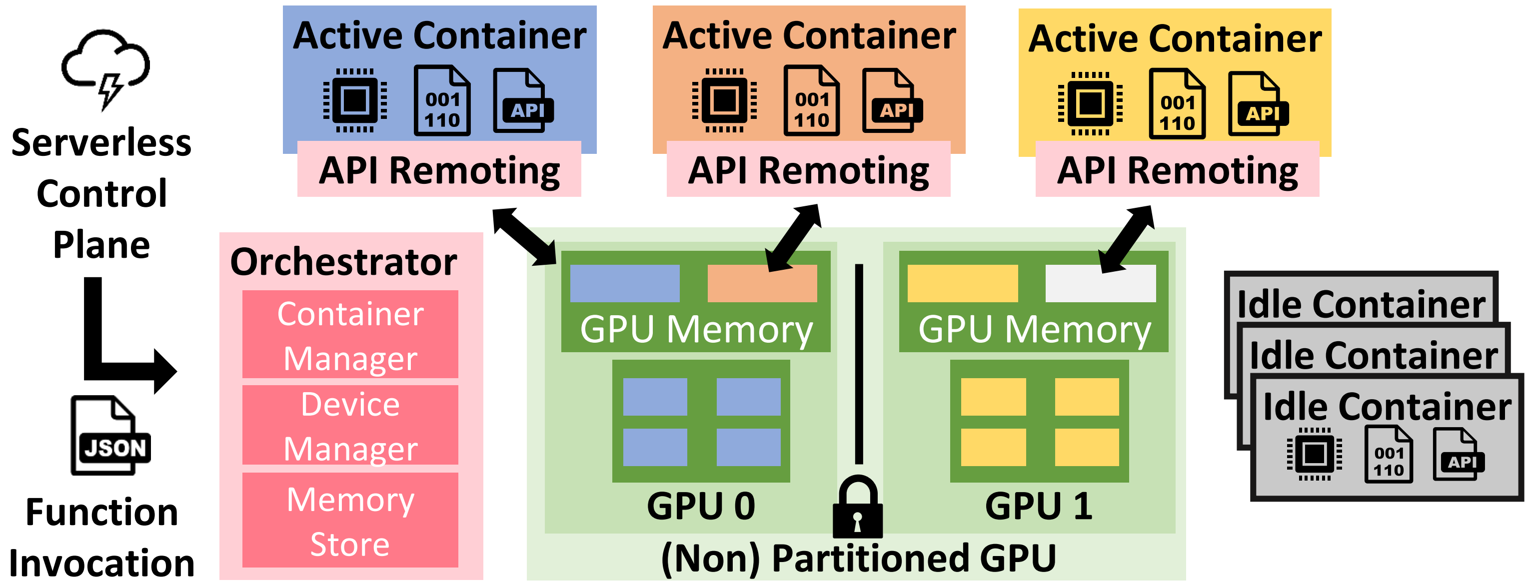
| Benchmark | Native CUDA | MIGnificient | | |
|---|---|---|---|---|
| | Time Sharing | Sequential | Fast Function Switching | |
| BFS | 505.3 ± 2.5 | 990.5 ± 112 | 528.8 ± 14.6 | **1.87x** |
| hotspot | 92.1 ± 0.8 | 195.1 ± 22.2 | 103 ± 9.9 | **1.89x** |
| ResNet-50 | 18 ± 0.3 | 53.3 ± 6 | 27.5 ± 0.7 | **1.94x** |
| AlexNet | 15.4 ± 0.5 | 49.2 ± 5.5 | 26.4 ± 0.9 | **1.86x** |
| Vgg19 | 23.6 ± 1 | 54.5 ± 6.5 | 27.8 ± 1 | **1.96x** |
| BERT-SQuaD | 40.2 ± 2.5 | 65.8 ± 7.5 | 41.4 ± 3.1 | **1.58x** |

**Two clients sending concurrently in total 10 requests.**
**Unmodified benchmarks without yield.**
**Bare-metal processes on RTX 4070 GPU.**

# MIGnificient: Isolated Functions

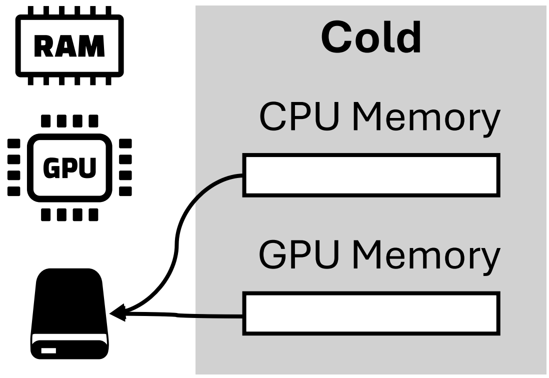


# MIGnificient: Isolated Functions

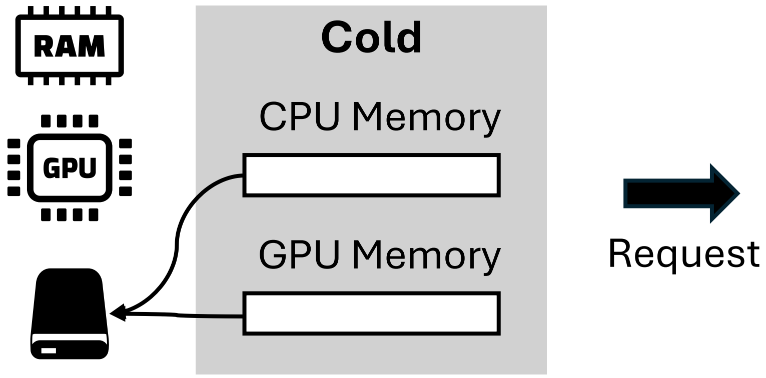



Idle container: CPU process + GPU context

# Lukewarm Functions

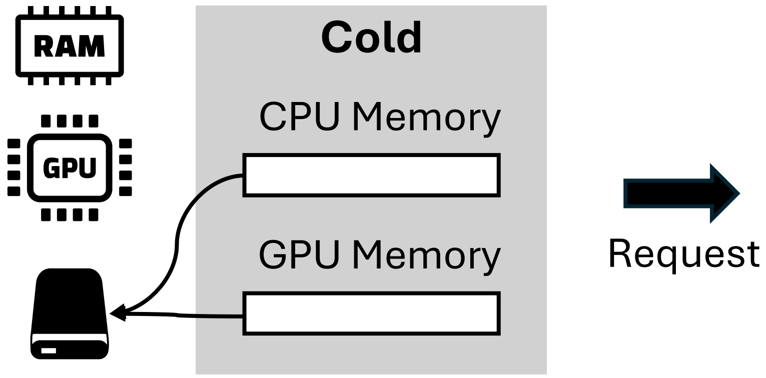


# Lukewarm Functions



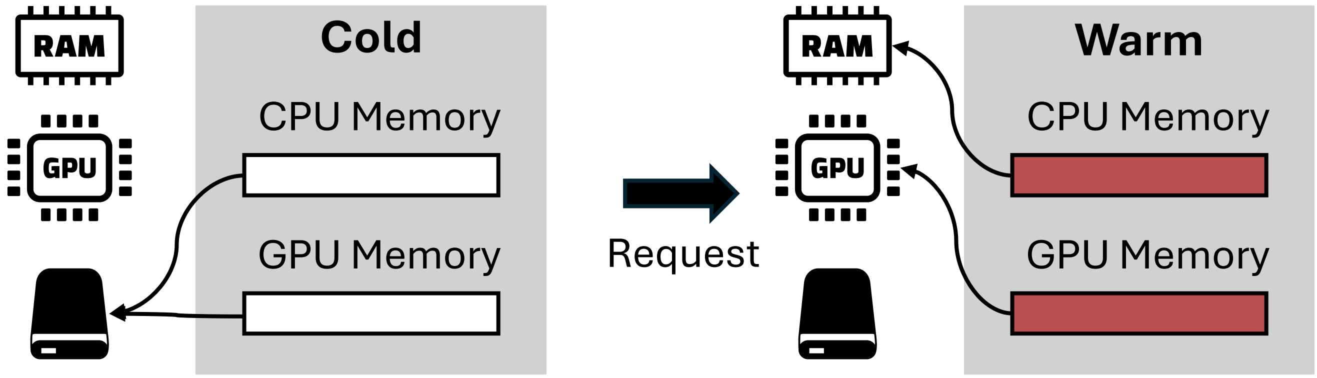
# Lukewarm Functions

Create CPU Process -> Import PyTorch -> Initialize CUDA Context -> Load Libraries -> Load Model from Disk



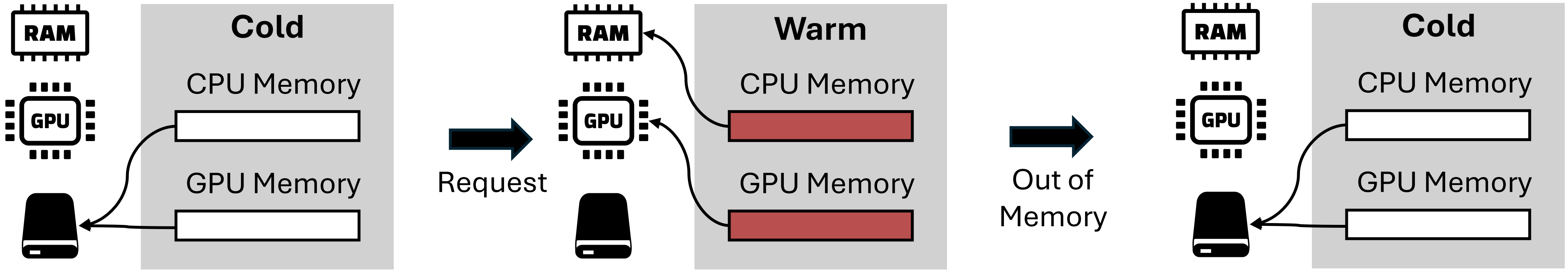
# Lukewarm Functions

Create CPU Process -> Import PyTorch -> Initialize CUDA Context -> Load Libraries -> Load Model from Disk



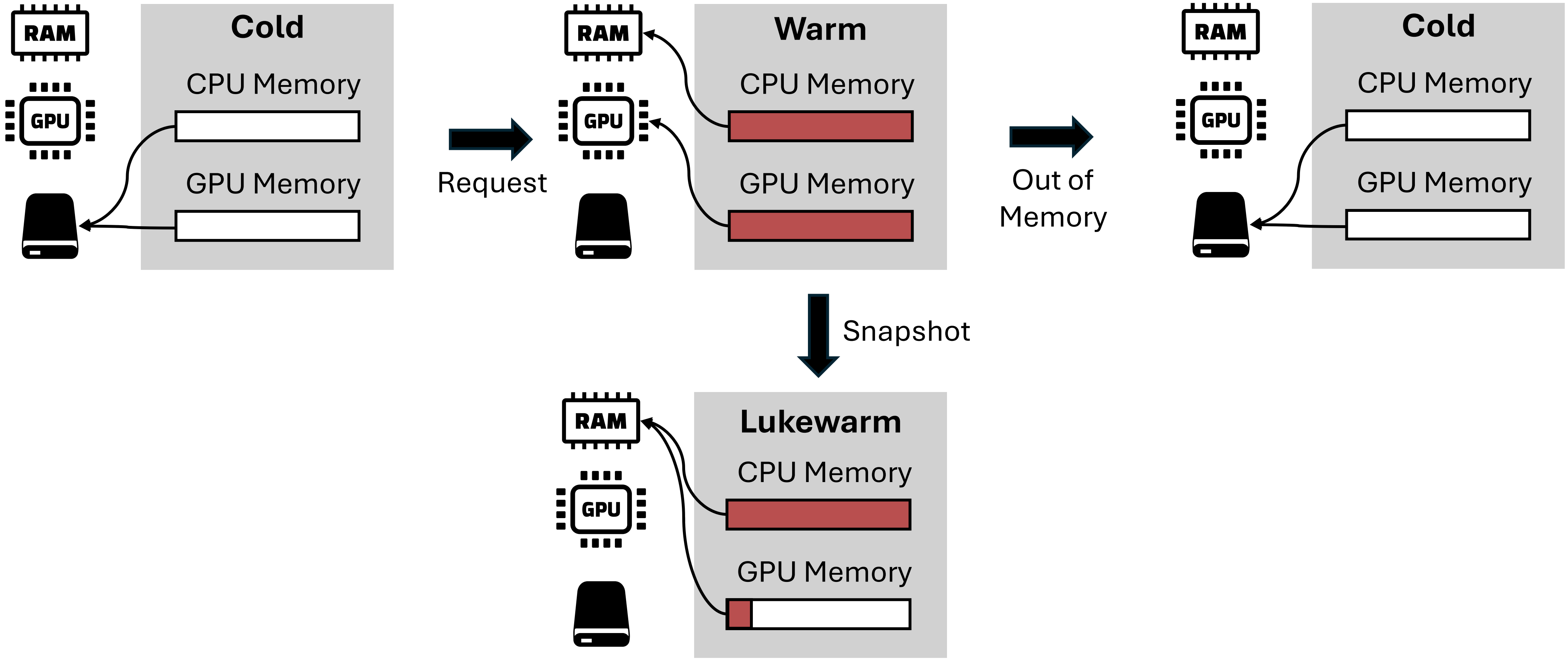
# Lukewarm Functions

Create CPU Process -> Import PyTorch -> Initialize CUDA Context -> Load Libraries -> Load Model from Disk



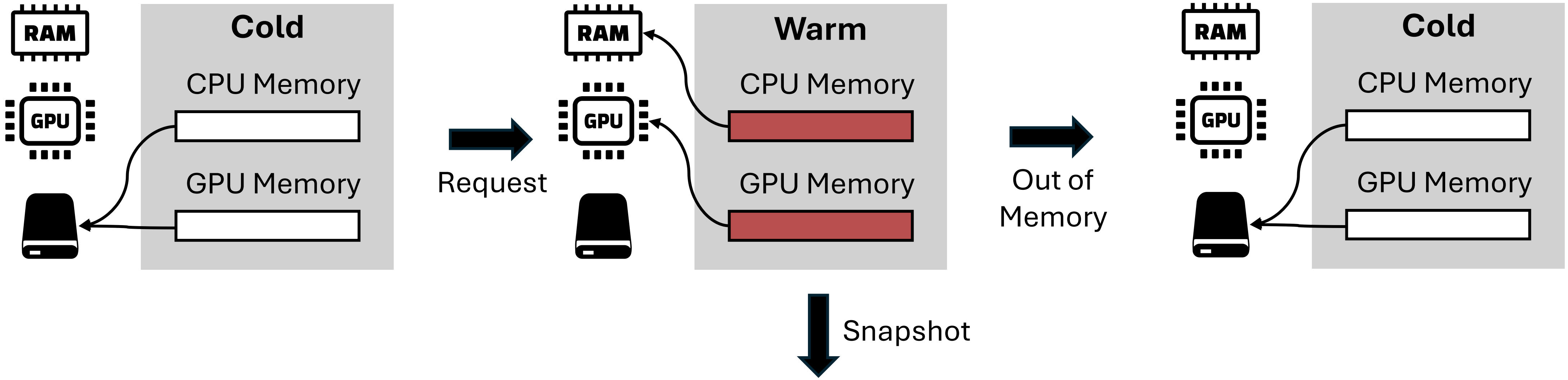
# Lukewarm Functions

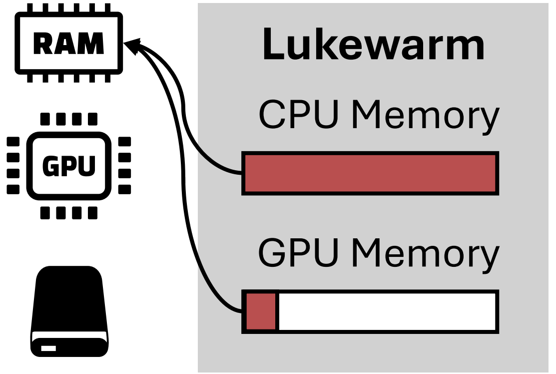
Create CPU Process -> Import PyTorch -> Initialize CUDA Context -> Load Libraries -> Load Model from Disk



# Lukewarm Functions

**Create CPU Process -> Import PyTorch -> Initialize CUDA Context -> Load Libraries -> Load Model from Disk**



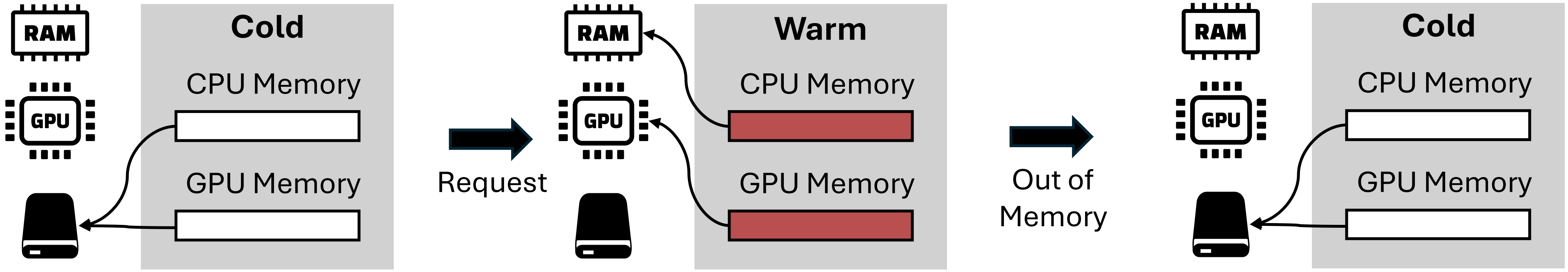



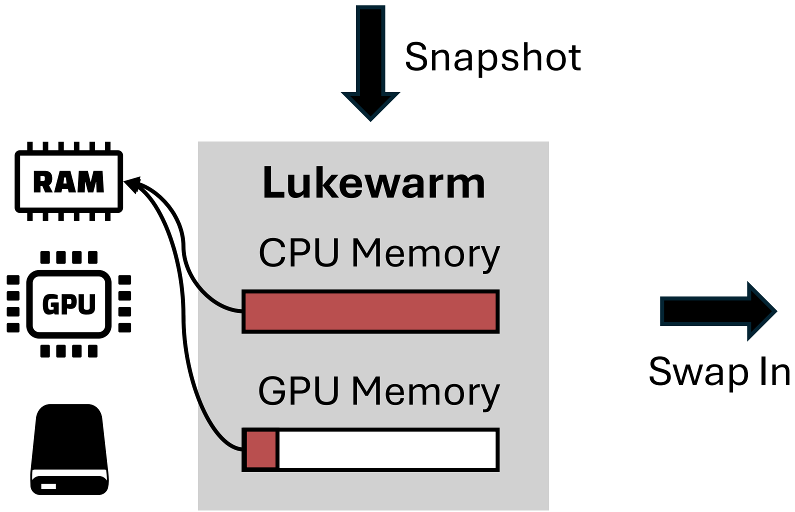

CPU Process
CUDA Context
Libraries
CUDA Virtual Addresses

CUDA Memory Allocations

# Lukewarm Functions

**Create CPU Process -> Import PyTorch -> Initialize CUDA Context -> Load Libraries -> Load Model from Disk**



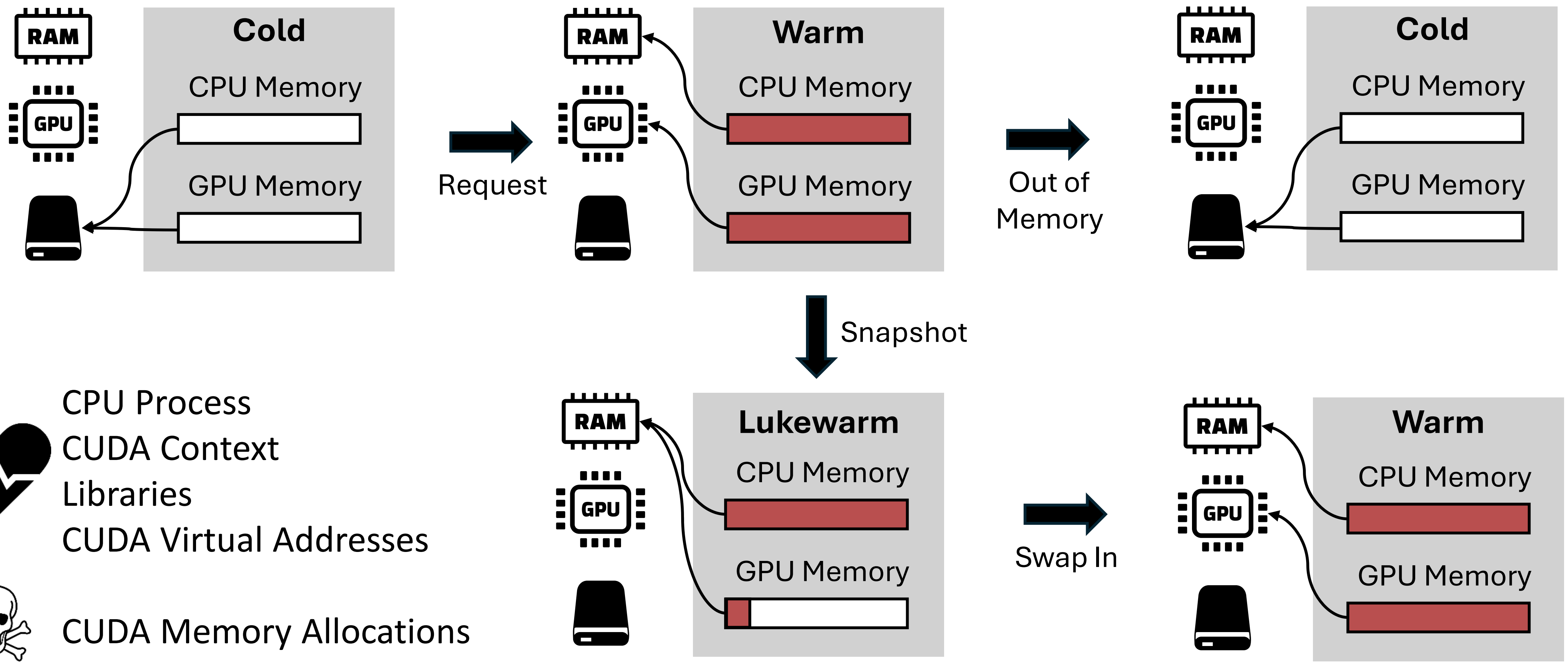




CPU Process
CUDA Context
Libraries
CUDA Virtual Addresses

CUDA Memory Allocations

# Lukewarm Functions

**Create CPU Process -> Import PyTorch -> Initialize CUDA Context -> Load Libraries -> Load Model from Disk**

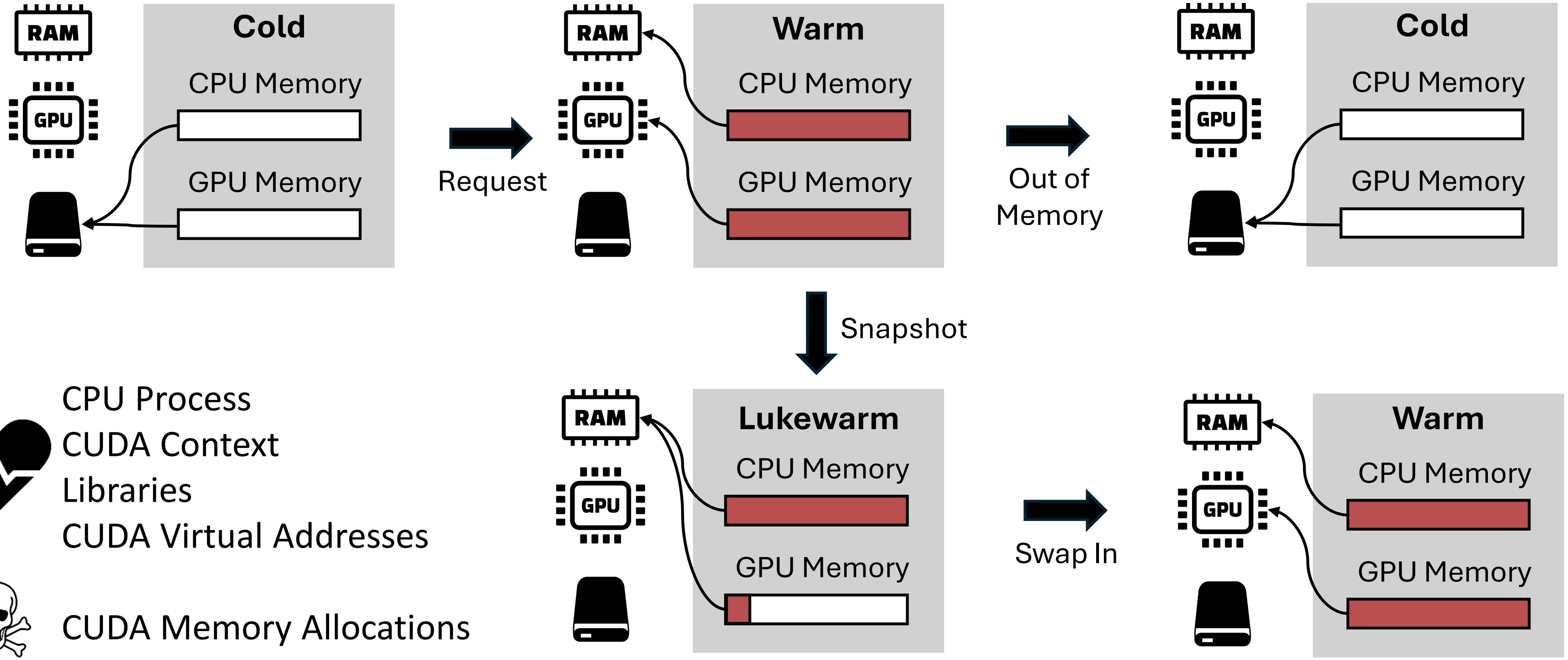



CPU Process
CUDA Context
Libraries
CUDA Virtual Addresses

CUDA Memory Allocations

# Lukewarm Functions

Create CPU Process -> Import PyTorch -> Initialize CUDA Context -> Load Libraries -> **Load Model from Disk**

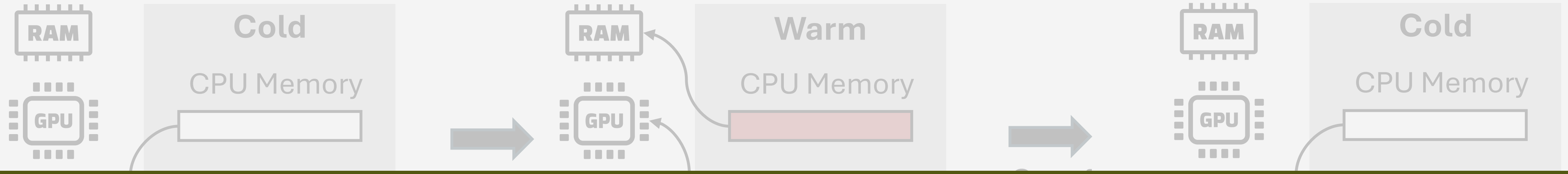



- CPU Process
- CUDA Context
- Libraries
- CUDA Virtual Addresses

- CUDA Memory Allocations

# Lukewarm Functions

Create CPU Process -> Import PyTorch -> Initialize CUDA Context -> Load Libraries -> Load Model from Disk

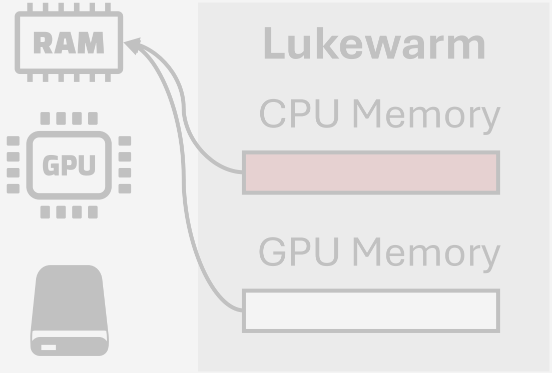



**ResNet-50: 23.45 ms of copying 142 MB snapshot versus 107 ms load time.**

**BERT: 214.47 ms of copying 1.3 GB snapshot versus 730.2 ms load time.**

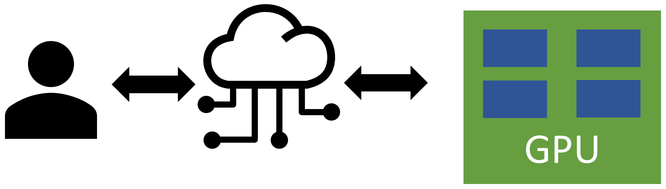
CPU Process
CUDA Context
Libraries
CUDA Virtual Addresses

CUDA Memory Allocations



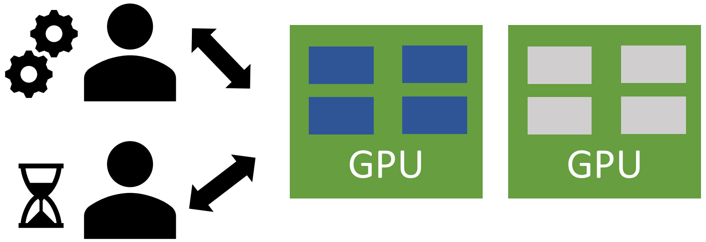

**Bare-metal processes on RTX 4070 GPU.**

# Conclusions

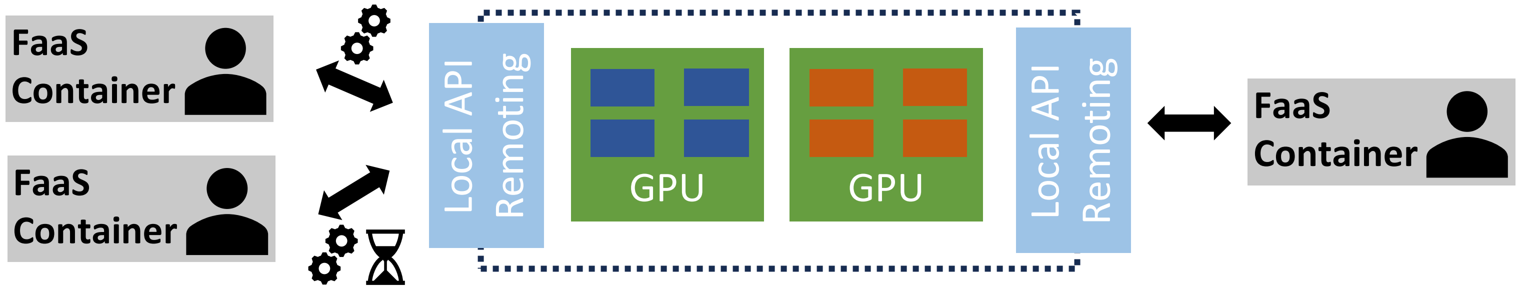

**Single tenant: underutilization.**

**API remoting: network performance penalty.**


**MPS: insufficient isolation.**

**MIG: insufficient elasticity.**




**MIGnificient: spatial isolation with optimized scheduling and overlapping.**

## More of SPCL's research:

- youtube.com/@spcl — 210+ Talks
- twitter.com/spcl_eth — 1.6K+ Followers
- github.com/spcl — 5.6K+ Stars

… or spcl.ethz.ch

GitHub **spcl/mignificient**

**Serverless HPC Projects**