ACM HPDC 2025, Notre Dame, IN, USA

FluidFaaS: A Dynamic Pipelined Solution for Serverless Computing with Strong Isolation-based GPU Sharing

Xinning Hui[1], Yuanchao Xu[2], Xipeng Shen[1]

North Carolina State University[1], University of California, Santa Cruz[2]

NC STATE UC SANTA CRUZ

What is Serverless Computing?

- Serverless computing popular cloud paradigm
  - Users deploy apps, providers provision resources
- Many benefits
  - Simple and modular programming
  - Automatic resource scaling
  - Pay-as-you-go model
- AWS lambda, Microsoft Azure, IBM Cloud, Google cloud functions

aws Lambda

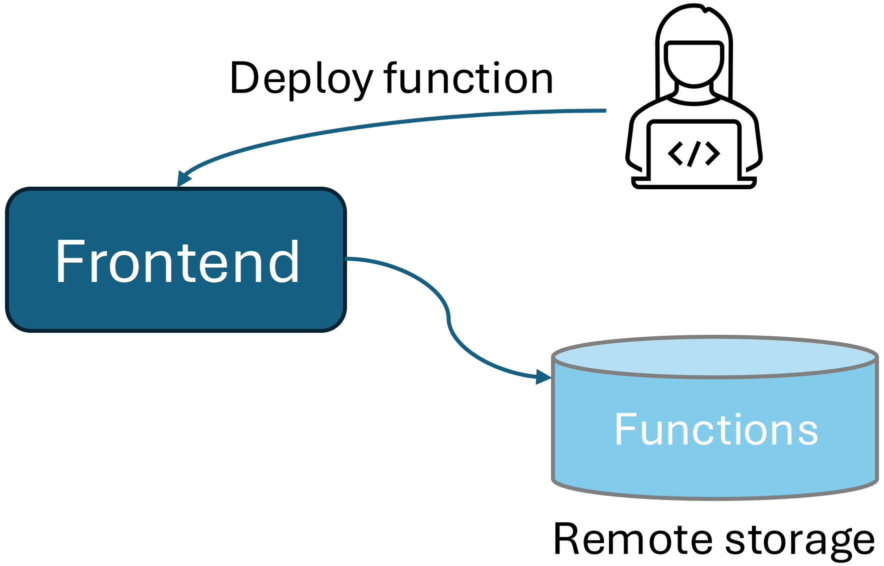
Google Cloud Functions

Azure Functions

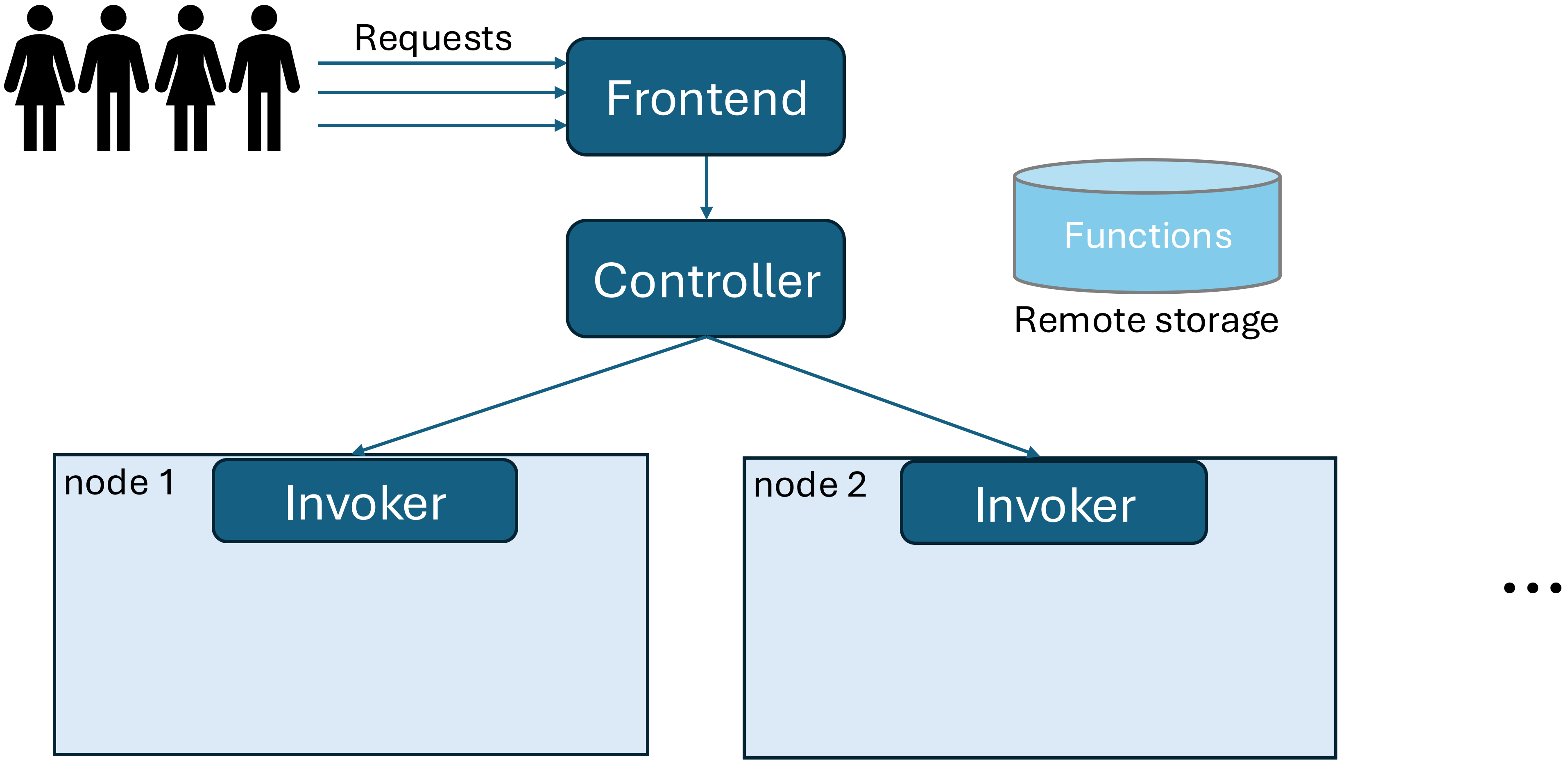
APACHE OpenWhisk

…

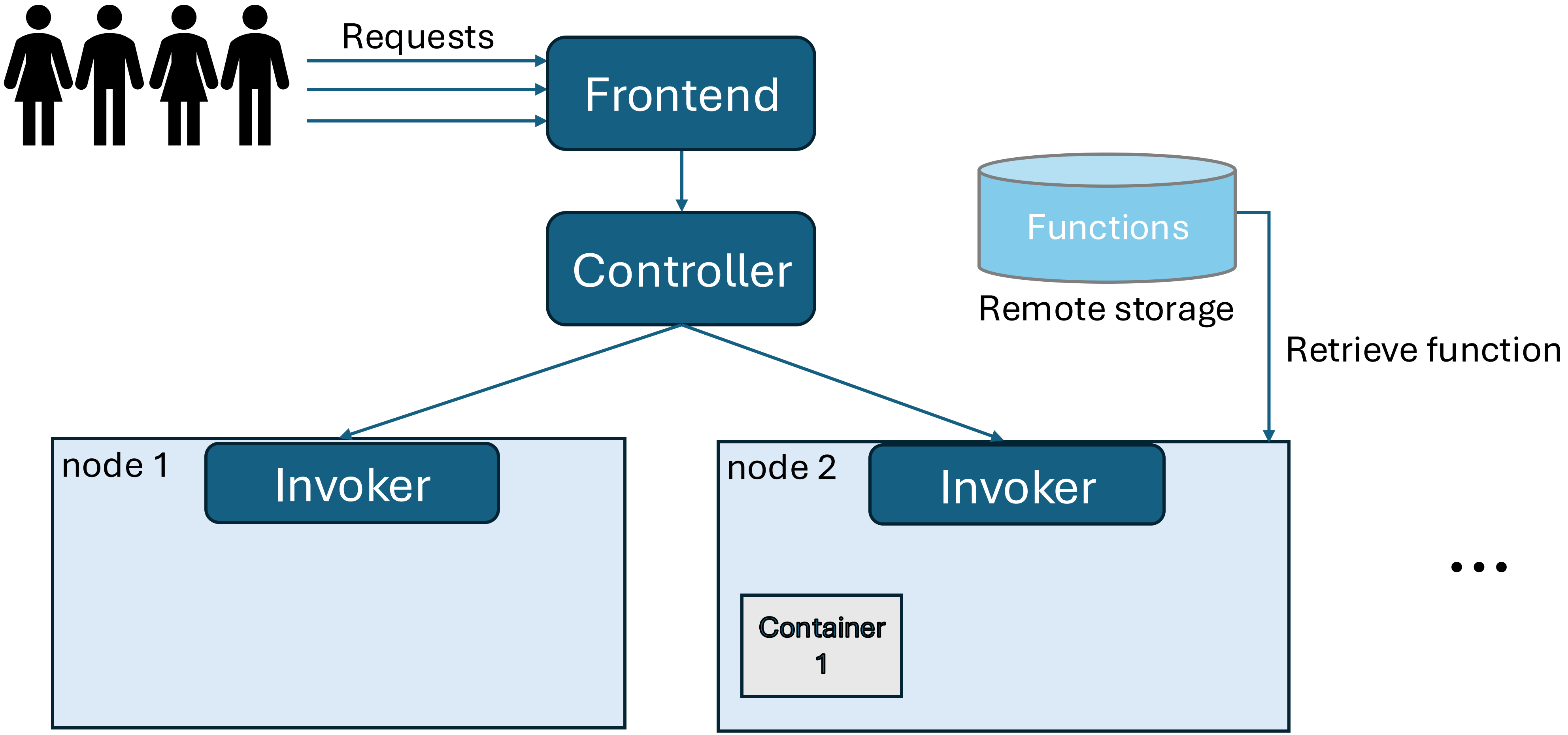
How Does Serverless Computing Work?



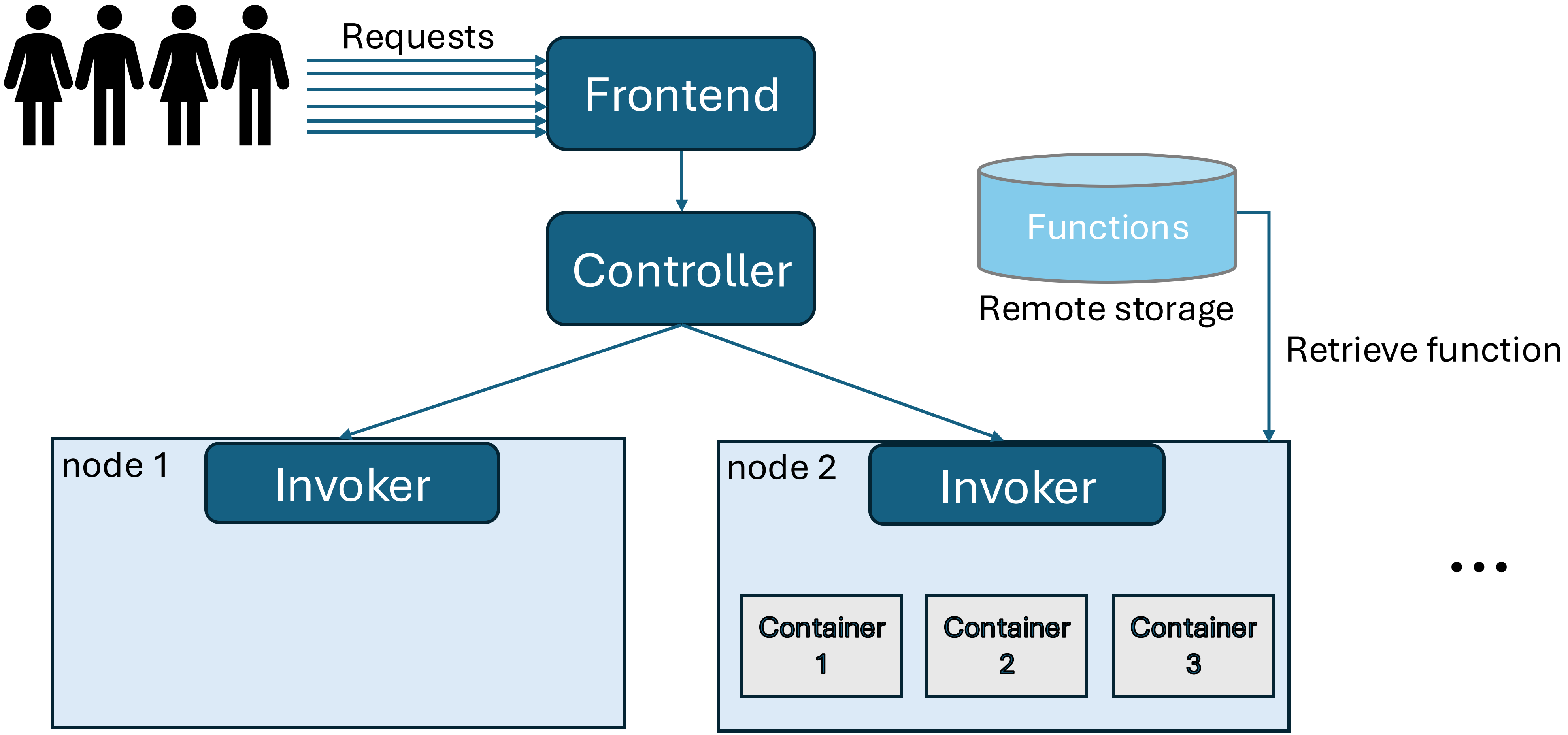
How Does Serverless Computing Work?



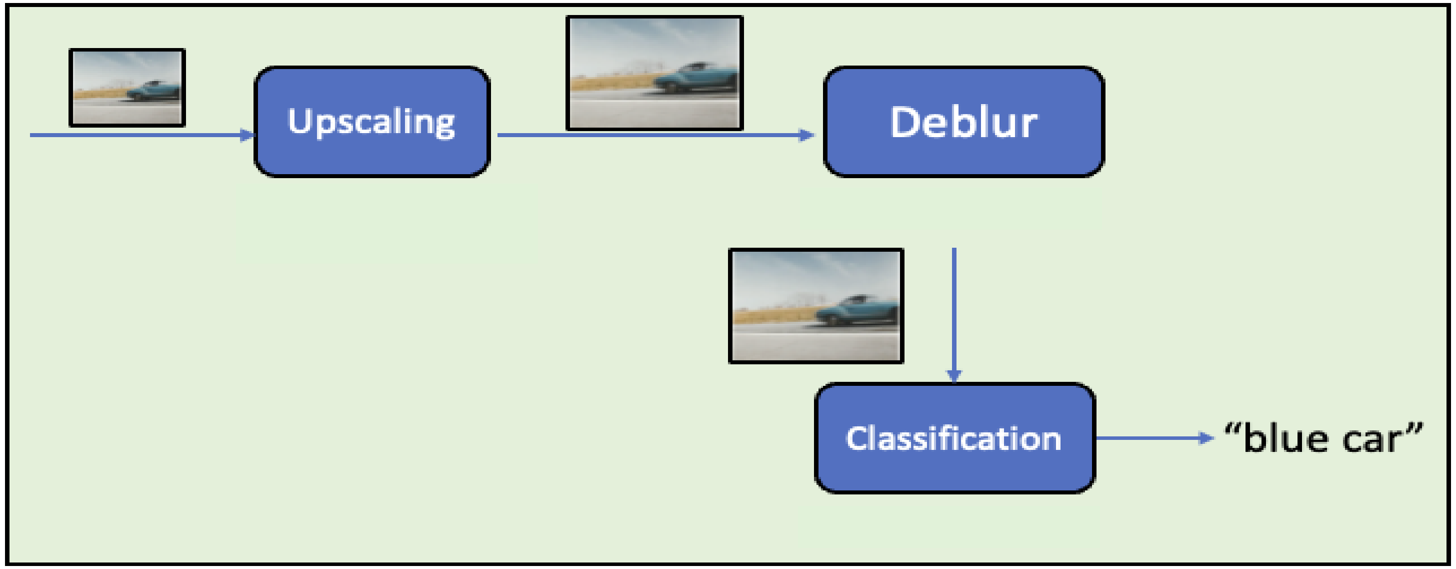
How Does Serverless Computing Work?



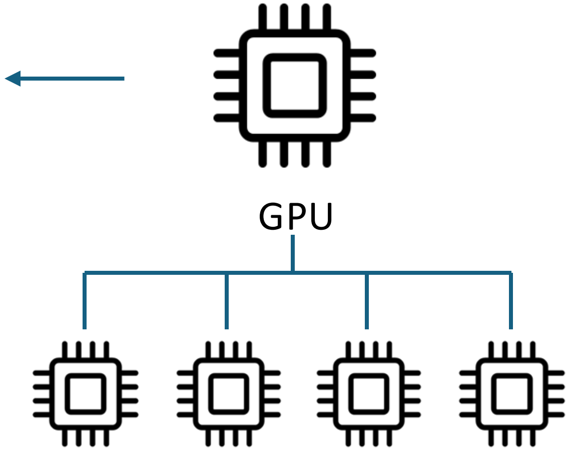
How Does Serverless Computing Work?



ML on Serverless Computing

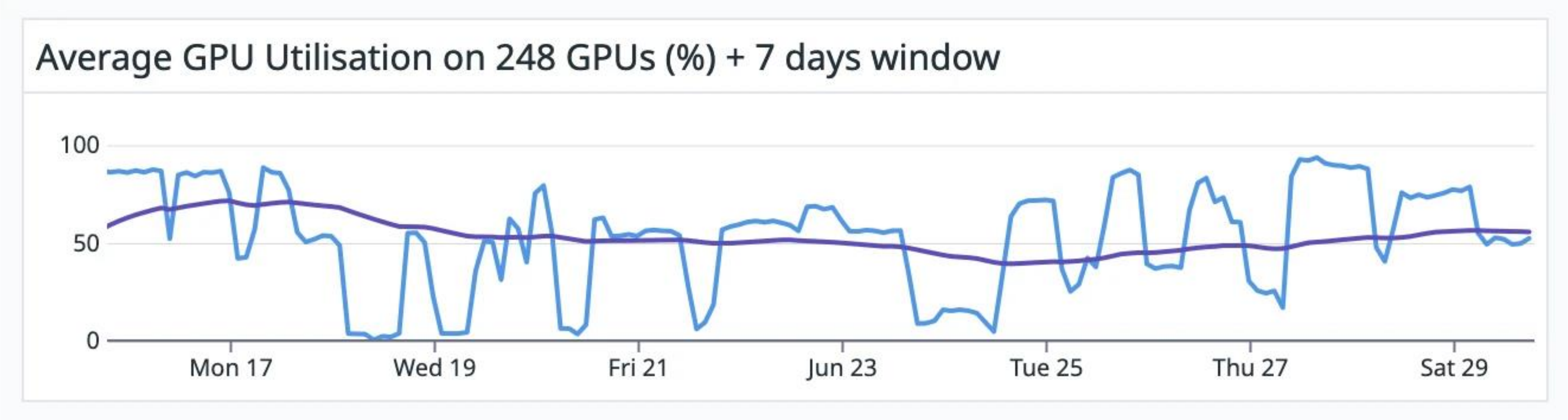


Deep learning inference application



GPU sharing

GPU in Serverless Computing

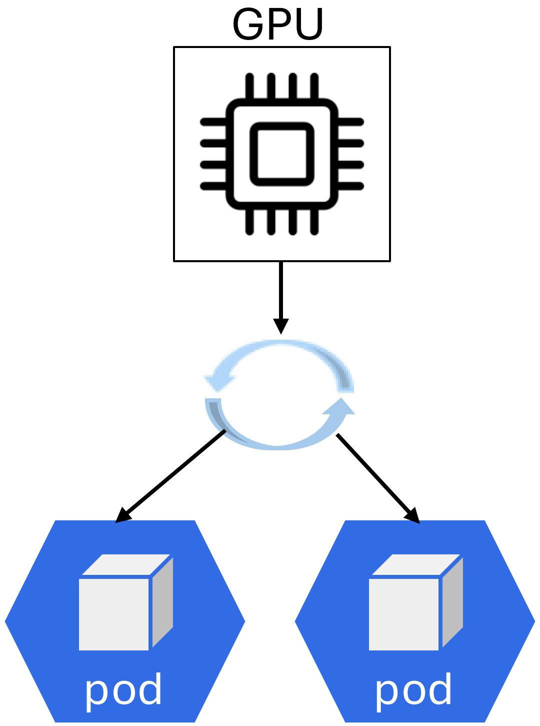


Low GPU utilization

[1] source: https://www.photoroom.com/inside-photoroom/so-you-want-to-rent-an-nvidia-h100-cluster-2024-consumer-guide

GPU Sharing

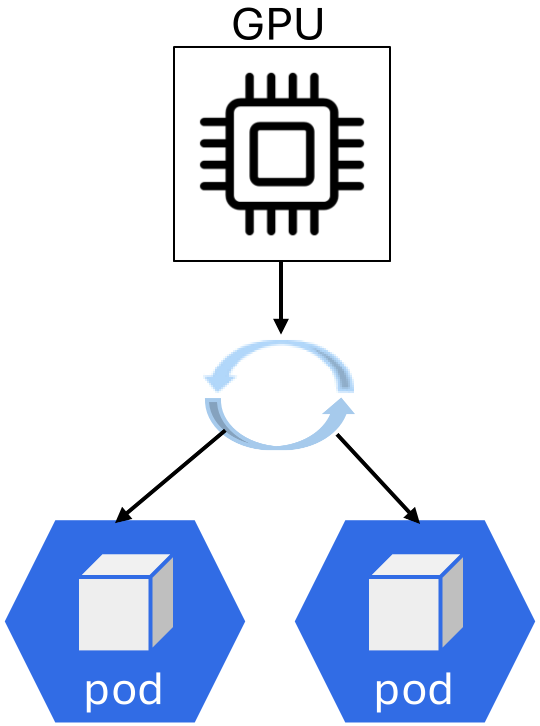
Time Sharing



Not solve underutilization

GPU Sharing

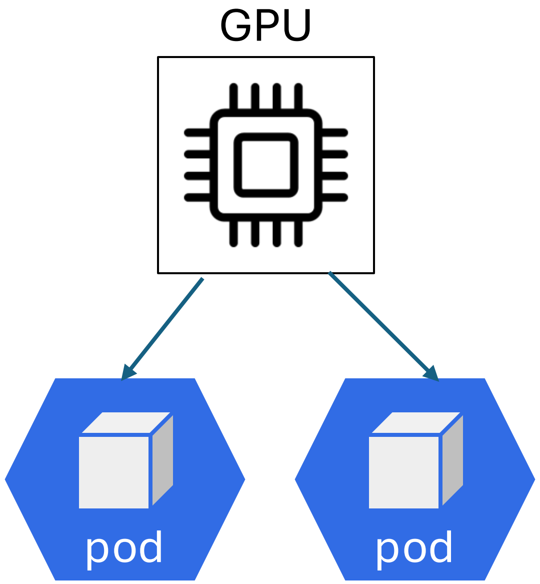
Time Sharing



Not solve underutilization

Spatial Sharing

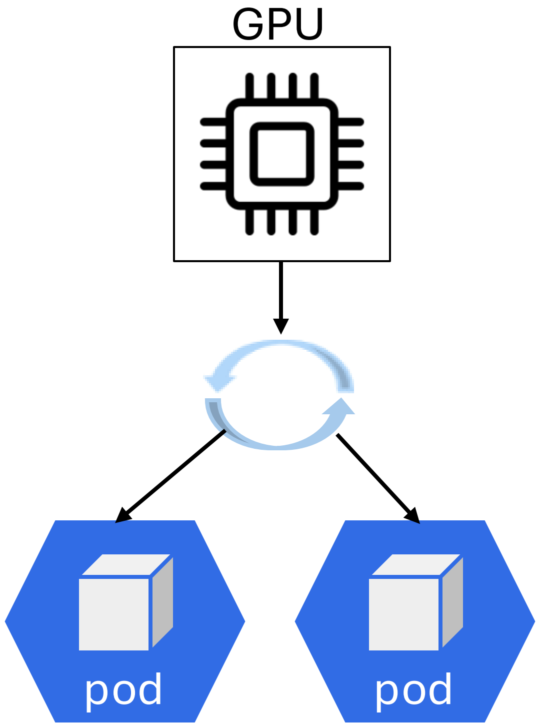
Multi-Process Service (MPS)



Performance interference & Security concern

GPU Sharing

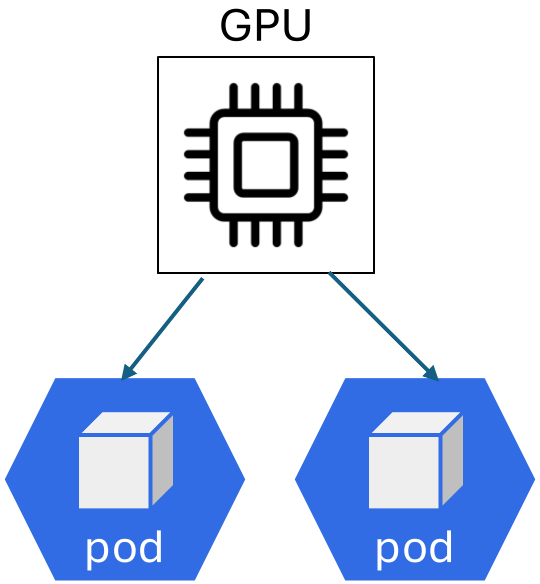
Time Sharing



Not solve underutilization

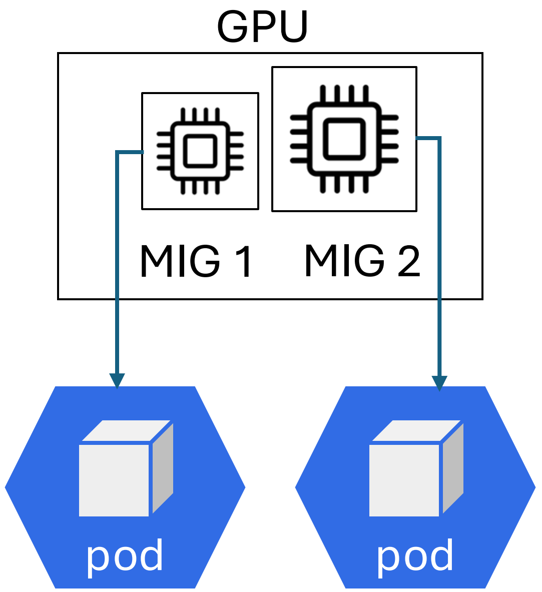
Spatial Sharing

Multi-Process Service (MPS)



Performance interference & Security concern

Multi-Instance GPU (MIG)



Hardware isolated & performance and security guarantee

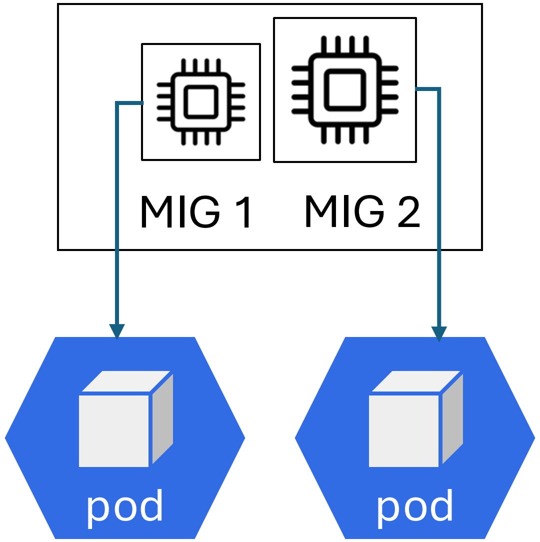
GPU Sharing

- MIG is preconfigured and reconfigure takes time
- MIG partitions example

Profile Name	Fraction of Memory	Fraction of SMs	Number of instance
MIG 1g.10gb	1/8	1/7	7
MIG 2g.20gb	2/8	2/7	3
MIG 3g.40gb	4/8	3/7	2
MIG 4g.40gb	4/8	4/7	1
MIG 7g.80gb	Full	7/7	1

Good for Serverless

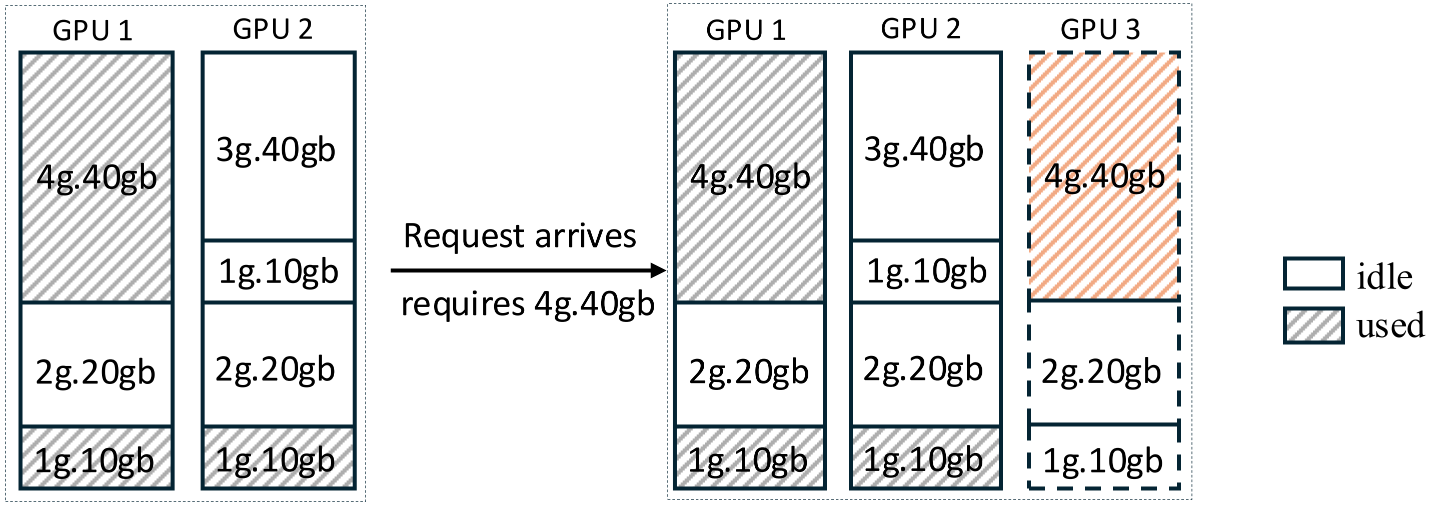
Multi-Instance GPU (MIG)



Hardware isolated & performance and security guarantee

Gaps for using MIG in Serverless ML

- **Gap #1: MIG underutilization caused by resource fragmentation**
  - Rigid MIG partition cause the MIG underutilized
  - Dynamic reconfiguration during runtime is impractical

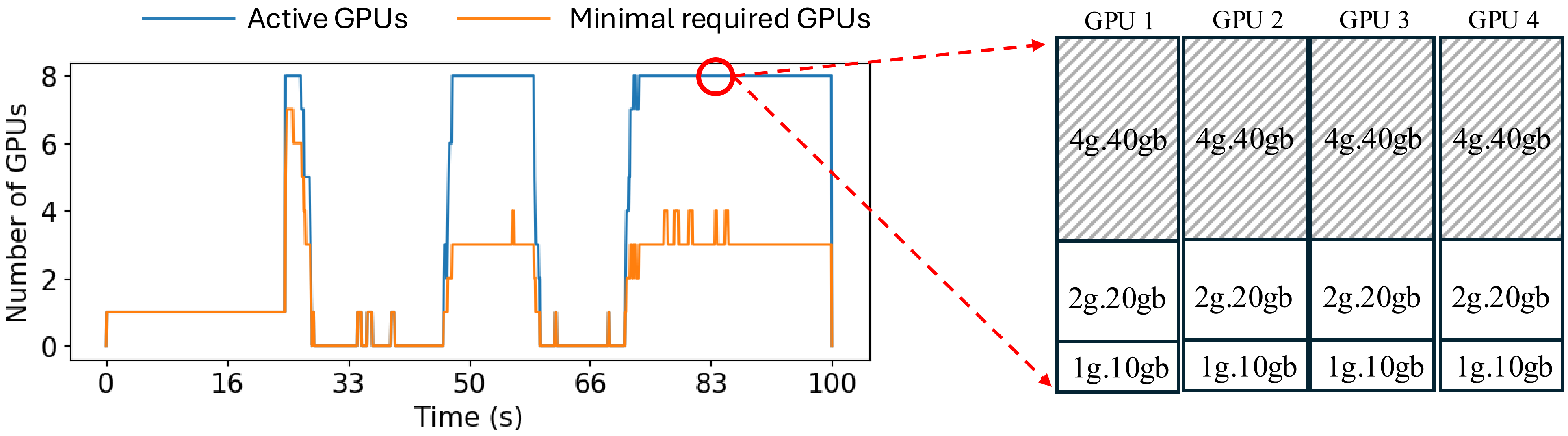


(a) System status

(b) Default assignment

Gaps for using MIG in Serverless ML

❖ Gap #1: MIG underutilization caused by resource fragmentation

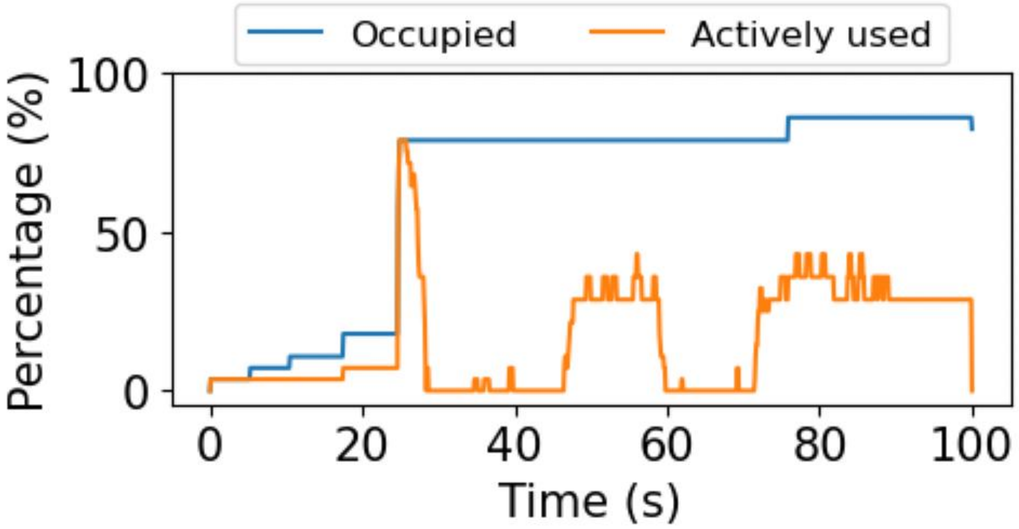


GPU 1	GPU 2	GPU 3	GPU 4
4g.40gb	4g.40gb	4g.40gb	4g.40gb
2g.20gb	2g.20gb	2g.20gb	2g.20gb
1g.10gb	1g.10gb	1g.10gb	1g.10gb

Only 4g.40gb is utilized

Gaps for using MIG in Serverless ML

- **Gap #2: MIG underutilization caused by exclusivity in warm state**
  - **Keeping a model active and precludes its resources from being used.**



Average active percentage is 16.1%, MIGs operate at less than 35% for 90% of time.

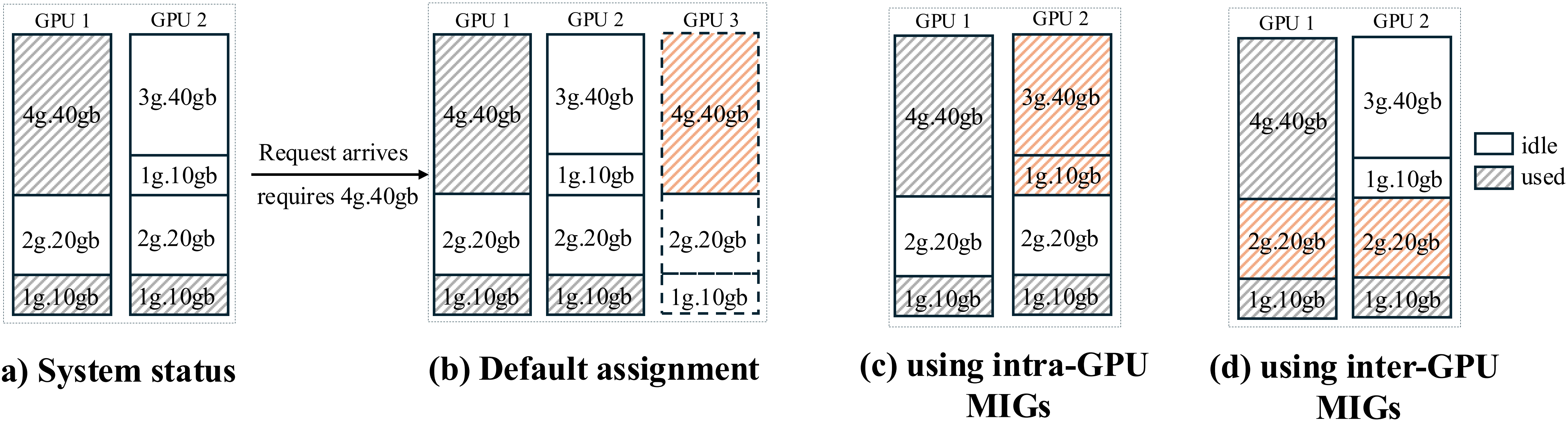
FluidFaaS

- **Underutilization caused by resource fragmentation**
  - **Automatic pipeline construction on-the-fly**

- Underutilization caused by **exclusivity in warm state**.
  - Hotness-aware eviction-based time sharing

FluidFaaS

❖ Design point #1: Automatic pipeline construction on-the-fly



(a) System status

(b) Default assignment

(c) using intra-GPU MIGs

(d) using inter-GPU MIGs

Decompose the workflow into multi MIGs

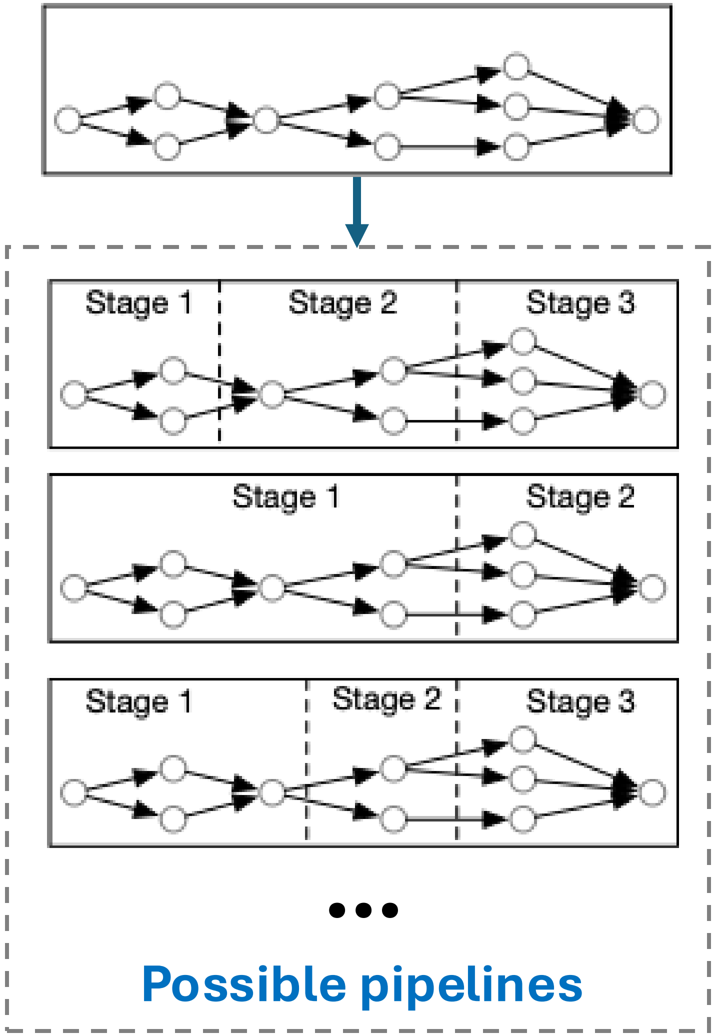
Programming support

One process one MIG (flexibility & throughput)

Runtime support

FluidFaaS – Automatic Pipeline Construction on-the-fly

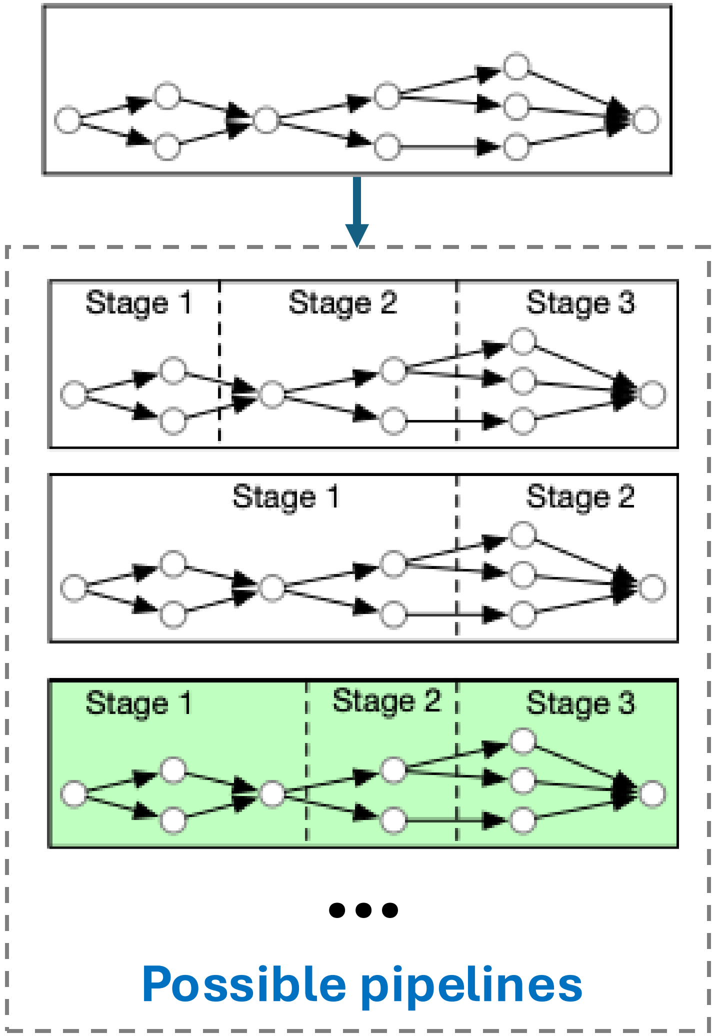
❖ **Programming Support:** transparent to users and adaptable at runtime.



Program support defines the minimal block.

FluidFaaS – Automatic Pipeline Construction on-the-fly

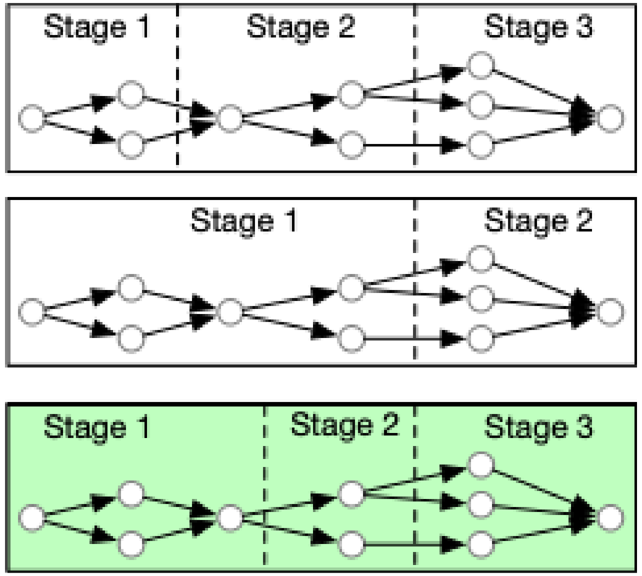
❖ **Runtime Support:** balanced pipeline and adaptive to resource availability

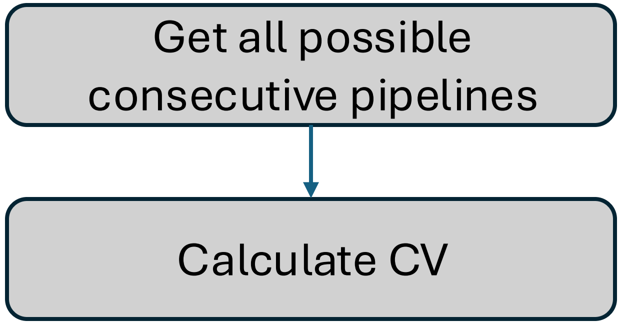


Runtime support construct the pipeline and provide the interface to run the pipeline.

FluidFaaS – Automatic Pipeline Construction on-the-fly

❖ **Runtime Support: balanced pipeline and adaptive to resource availability**



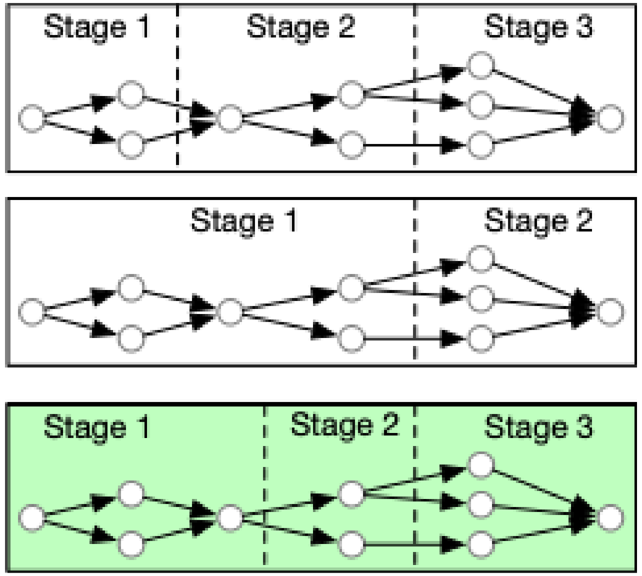


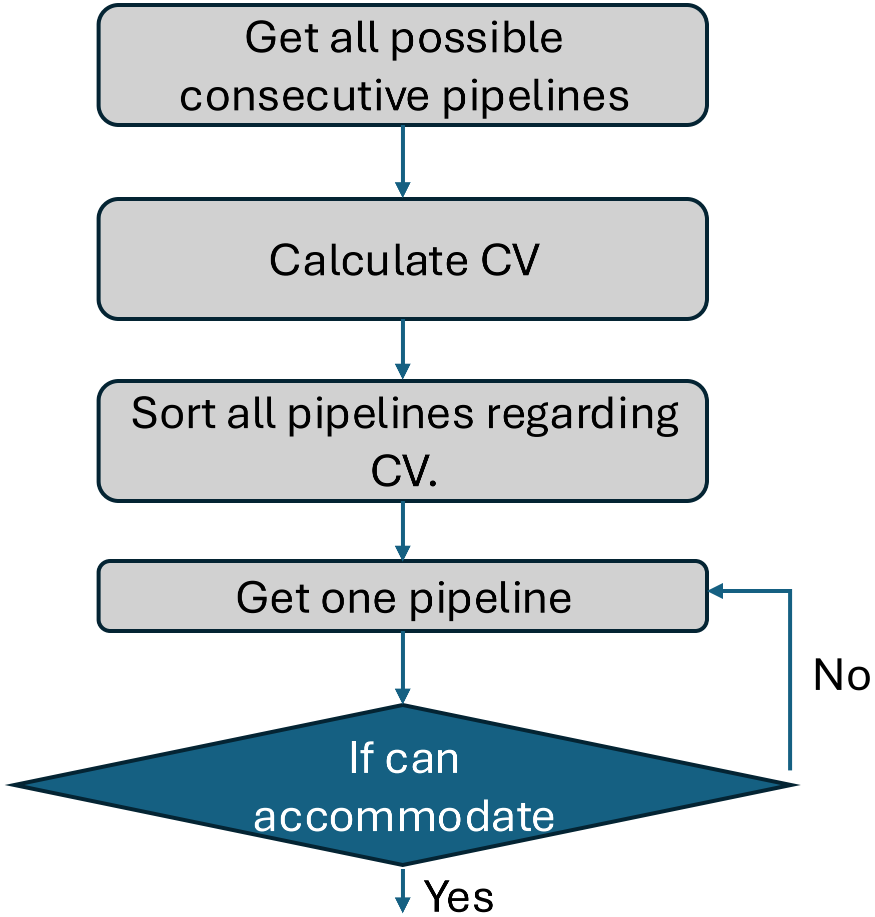
Get all possible consecutive pipelines

Calculate CV

FluidFaaS – Automatic Pipeline Construction on-the-fly

- **Runtime Support: balanced pipeline and adaptive to resource availability**
  - **Coefficient of variation (CV)**





Get all possible consecutive pipelines

Calculate CV

Sort all pipelines regarding CV.

Get one pipeline

If can accommodate

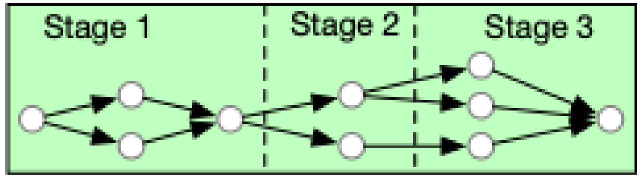
No

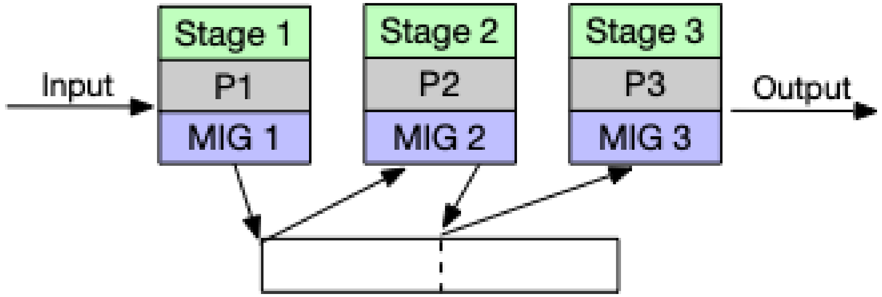
Yes

$CV = std(t_1, t_2, \dots, t_n)/\text{mean}(t_1, t_2, \dots, t_n),$ lower is better.

FluidFaaS – Automatic Pipeline Construction on-the-fly

- **Runtime support:** transparent to users and adaptable at runtime.
  - Create a separate process for each MIG
  - Communicate via shared memory.
  - Reside in Invoker





Shared memory in host

FluidFaaS

- Underutilization caused by **resource fragmentation**
  - Automatic pipeline construction on-the-fly

- **Underutilization caused by exclusivity in warm state.**
  - **Hotness-aware eviction-based time sharing**

FluidFaaS - Hotness-aware Eviction-based Time Sharing

- Design principle: improve MIG utilization
  - interleaved usage of MIG slice through eviction.

FluidFaaS - Hotness-aware Eviction-based Time Sharing

- Design principle: improve MIG utilization
  - interleaved usage of MIG slice through eviction.
    - Exclusive hot state: high request load instance.
    - Time sharing state: not actively busy instance (i.e., utilization below 30%).

FluidFaaS - Hotness-aware Eviction-based Time Sharing

- Design principle: improve MIG utilization
  - interleaved usage of MIG slice through eviction.
    - Exclusive hot state: high request load instance.
    - Time sharing state: not actively busy instance (i.e., utilization below 30%).
  - Clod state vs. warm state (in the CPU)

FluidFaaS - Hotness-aware Eviction-based Time Sharing

- Design principle: improve MIG utilization
  - interleaved usage of MIG slice through eviction.
    - Exclusive hot state: high request load instance.
    - Time sharing state: not actively busy instance (i.e., utilization below 30%).
  - Clod state vs. warm state (in the CPU)
  - Heterogeneity-aware request routing.
    - urgent to exclusive and non-urgent to the time sharing instance.

FluidFaaS - Hotness-aware Eviction-based Time Sharing

- Design principle: improve MIG utilization
  - interleaved usage of MIG slice through eviction.
    - Exclusive hot state: high request load instance.
    - Time sharing state: not actively busy instance (i.e., utilization below 30%).
  - Clod state vs. warm state (in the CPU)
  - Heterogeneity-aware request routing.
    - urgent to exclusive and non-urgent to the time sharing instance.
  - Pipeline migration
    - migrate pipeline instance to non-pipeline when large MIG slices are available.

Evaluation

- **Methodology:**
  - Hardware
    - 8 * A100 (80 GB)
    - 1g.10gb + 2g.20gb + 4g.40gb

Evaluation

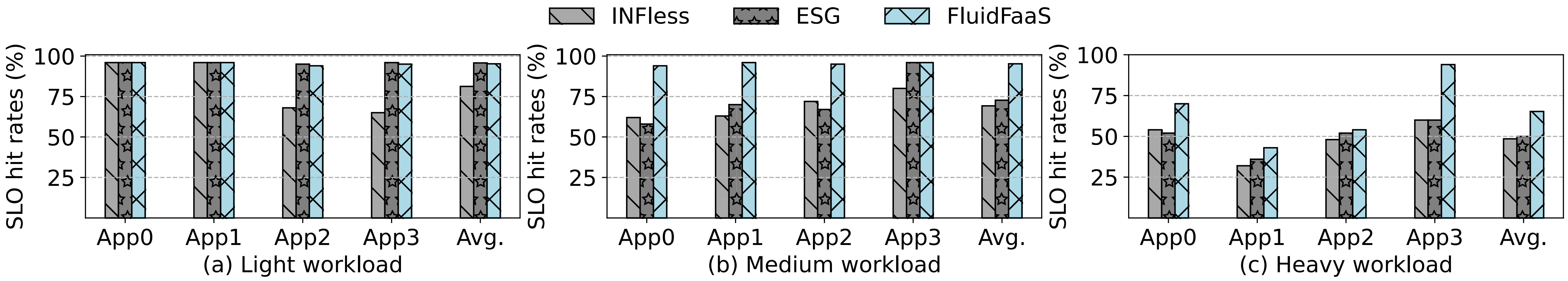
- **Methodology:**
  - **Hardware**
    - **8 * A100 (80 GB)**
    - **1g.10gb + 2g.20gb + 4g.40gb**
  - **Application**
    - **ML inference applications**

Evaluation

- Methodology:
  - Hardware
    - 8 * A100 (80 GB)
    - 1g.10gb + 2g.20gb + 4g.40gb
  - Application
    - ML inference applications
  - Baseline.
    - ESG (HPDC24) – most resource efficient MIG, no pipeline.
    - INFless (ASPLOS22) – MPS based Serverless platform, no pipeline.

Evaluation

- **Evaluated metrics: SLO hit rates.**



(a) Light workload

(b) Medium workload

(c) Heavy workload

91% high in medium workloads.

61% high in heavy workload.

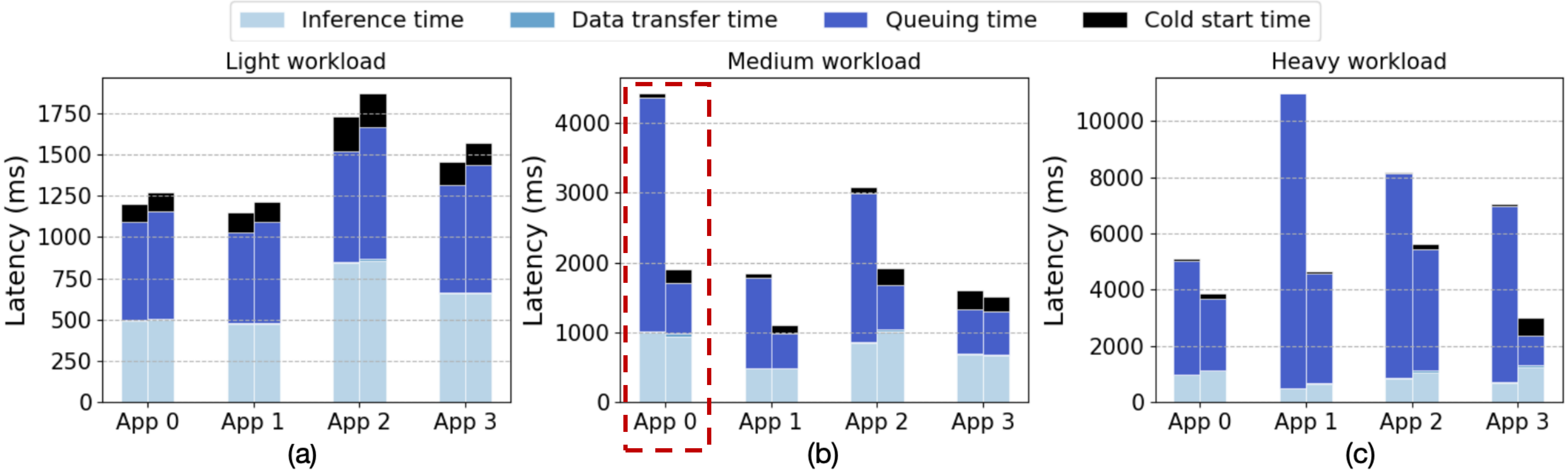
INFless and ESG similar performance due to non-pipeline execution model.

[1] Xinning Hui, etc. ESG: Pipeline-Conscious Efficient Scheduling of DNN Workflows on Serverless Platforms with Shareable GPUs , 2024, HPDC.

[2] Yanan Yang, etc. INFless: a native serverless system for lowlatency, high-throughput inference. 2022, ASPLOS.

Evaluation

- **Evaluated metrics:** End-to-end latency breakdown.



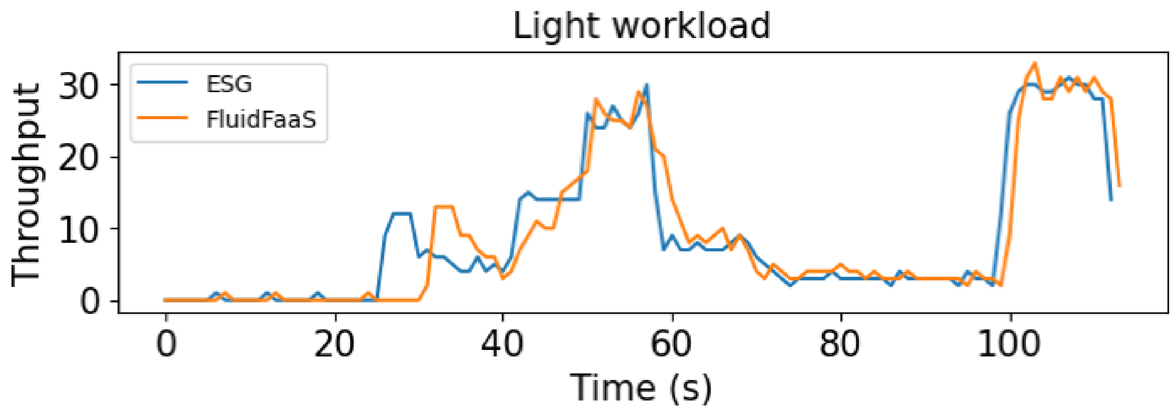
ESG (left) vs. FluidFaaS (right)

2.36x lower latency than ESG.

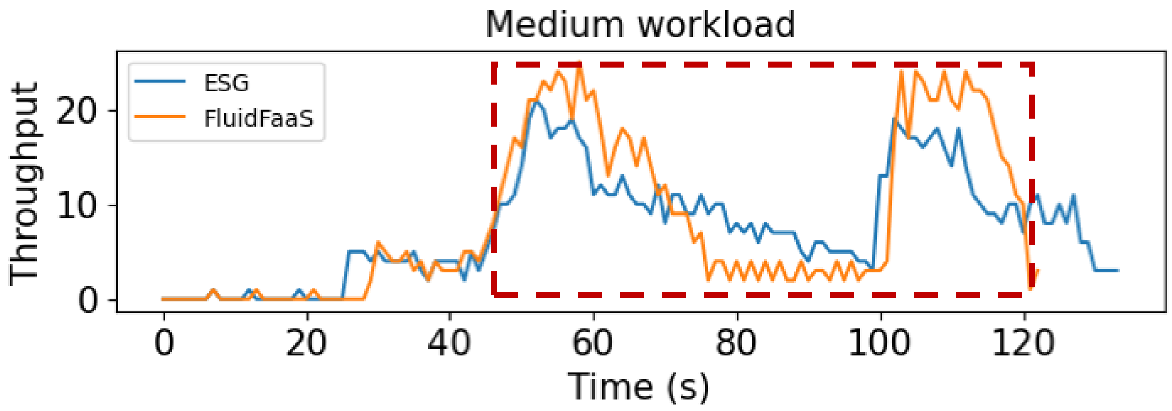
Data transfer time is negligible.

Evaluation

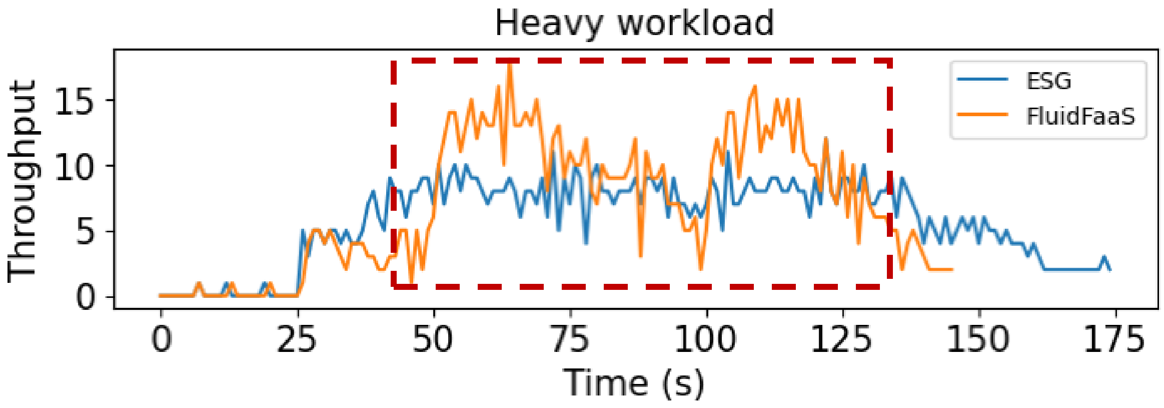
❖ **Evaluated metrics: Throughput**



Light workload



Medium workload



Heavy workload

75% higher in heavy workload.

25% higher in medium workload.

Conclusion

- Identifies the fundamental reason for severe GPU under-utilization, **MIG resource fragmentation** and **exclusive keep-alive policy**.
- Give **programming support** and **runtime support** to enable **on-the-fly pipeline construction.**
- Propose **Hotness-aware eviction-based time sharing**.
- Empirically show **25%-75%** improvement in system throughput and improving up to **90%** SLO hit rates.

Thanks for your time!

Questions are more than welcomed!