GPU Traveling

Enabling Efficient Confidential Collaborative Training with TEE-Enabled GPUs

Shixuan Zhao

A joint work with Zhongshu Gu, Salman Ahmed, Enriquillo Valdez, Hani Jamjoom, Zhiqiang Lin

THE OHIO STATE UNIVERSITY

IBM

Backgrounds
What are TEEs?

Trust Execution Environments are hardware features that can protect the software in it from the outside world

Things can go quite wrong in software

- Software bugs
- Operating system bugs

You runs things in the cloud

- … via VMs in the cloud

You can't really trust things in the cloud

- What if Amazon peeks your super duper extremely valuable secret?
- What if Amazon got hacked and the attacker peeks your super duper extremely valuable secret?

Backgrounds
What are TEEs?

Confidential VMs are here to help

Hypervisor can still manage your VMs

- Create/delete
- Pause
- Limit resource

But they can’t do anything else

- VM is encrypted inside the memory
- Limited interface

You know if you are safe

- *Remote attestation* to verify
- Intel/AMD will tell you if the CPU is good
- CPU will tell you if your VM is protected

Backgrounds
TEEs in the AI age

Introducing Confidential GPUs

Your models/data worth tons of money

But GPUs are merely PCIe devices

- Not in CVM’s trust boundary
- Can be hijacked by the hypervisor

It can’t be protected by CVMs alone!

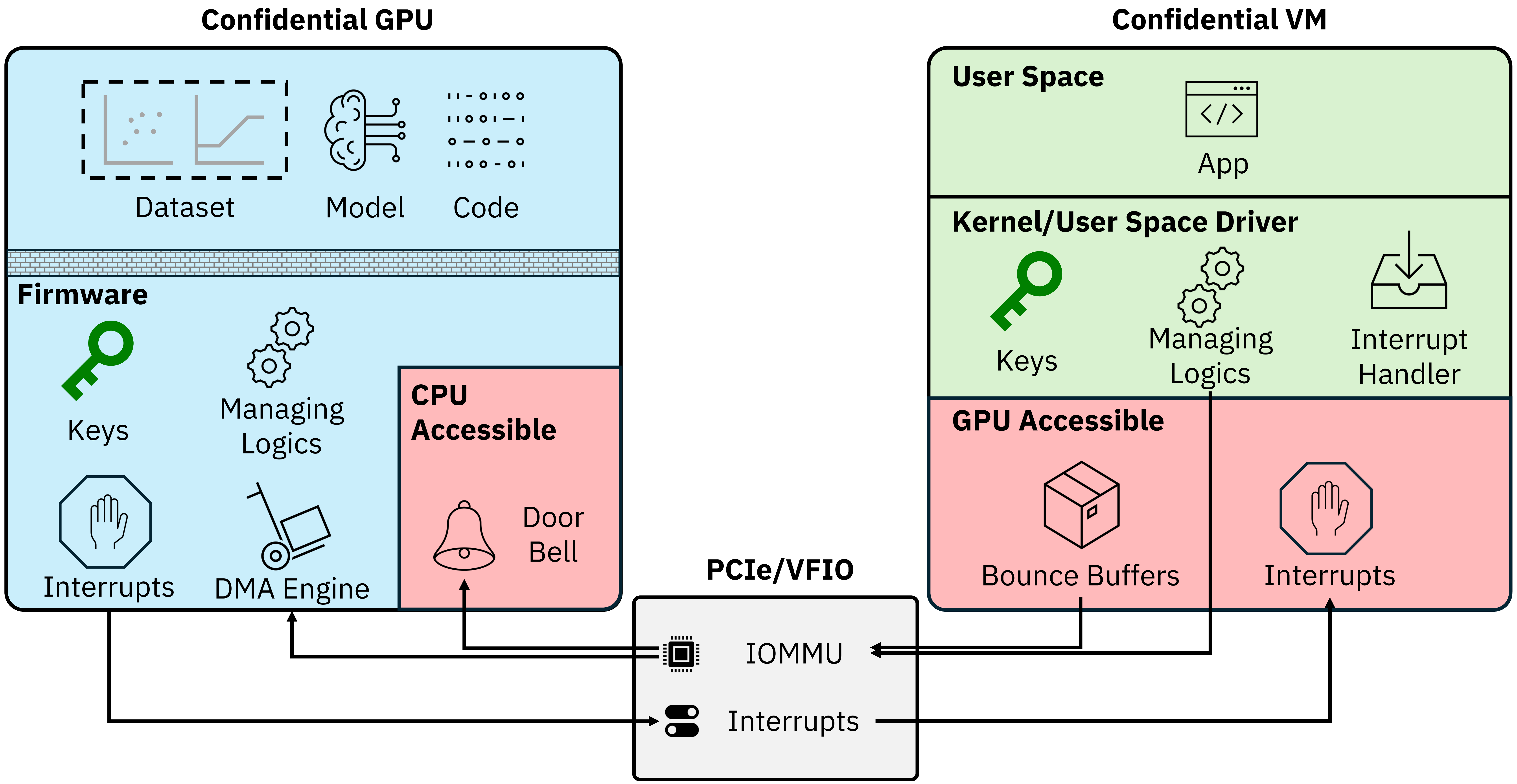
Backgrounds
TEEs in the AI age

Introducing Confidential GPUs

NVIDIA H100: The first confidential GPU
- It’s likely more expensive than your car
- Still not in CVM’s trust boundary
- Encryption via driver
- Communicate with CVM only with encryption
- No hypervisor access until reset

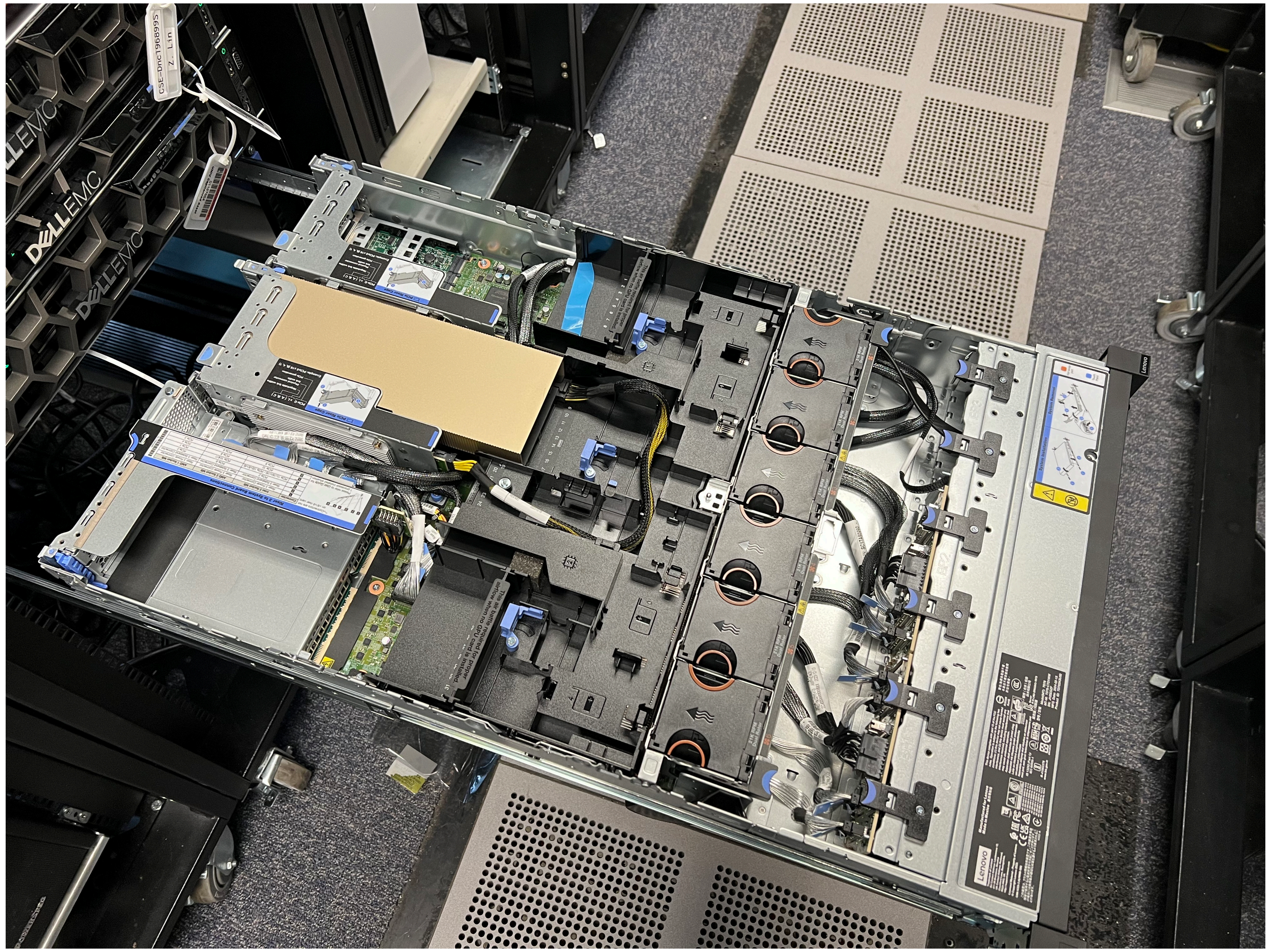
Same verifiable trust
- The CVM can now attest the GPU
- You attest the CVM
- So now you trust the GPU

Backgrounds
TEEs in the AI age



Backgrounds
TEEs in the AI age

In case you are wondering what it looks like...



Research Problem

Data sharing in confidential collaborative learning is expensive & prone to attacks

Data sharing alone is expensive

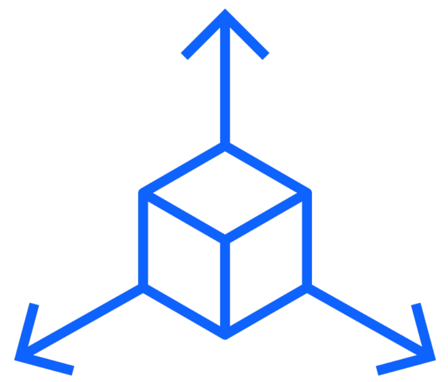
- Datasets can be huge
- Models can be large
- Transmission cost across nodes is high
- GPU has limited bandwidth

Confidential data sharing is even more tricky

- Sensitive data sharing is prohibited
- Memory sharing across different confidential domains has extra high cost for encryption
- Longer data path can increase the attack surface

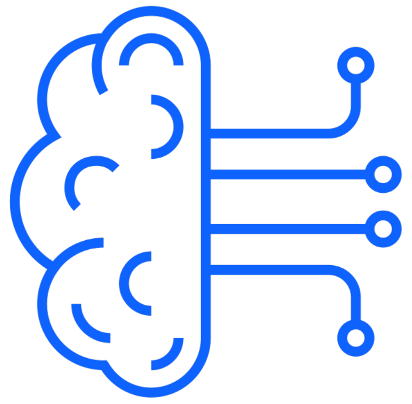
Existing Solutions

Either share the dataset or model



Sharing Dataset

- **Performance:**
  Impacted due to huge datasets
- **Security:**
  Can be vulnerable due to broaden attack surface



Sharing Model/Gradients

- **Performance:**
  Can be impacted due to huge models
- **Security:**
  Can be vulnerable due to individual model inversion

Proposal

Minimising data sharing via GPU Travelling

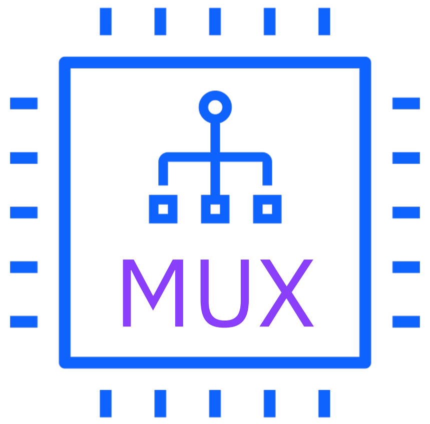
Confidential Training Data
Distrusting data holders own their *confidential VMs* and controls private training data

GPU Travelling
Physical GPU rotates to different confidential VMs at runtime

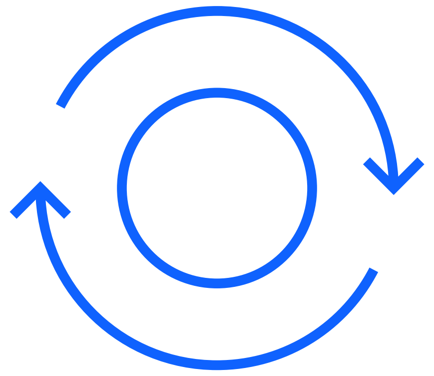
Low Cost of Data Sharing
Model stays in GPU memory and every datasets is only copied once from main memory to GPU memory

Enabler

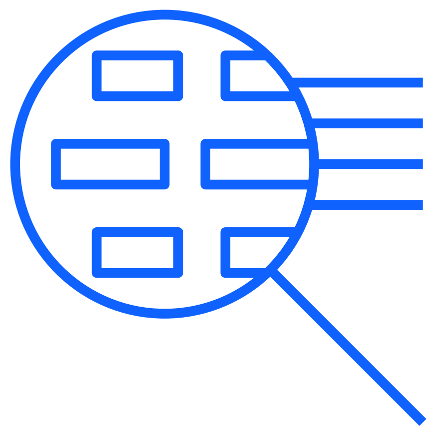
NVIDIA H100 CC uses encrypted data path



PCIe/SXM MUX: Routing a physical GPU to multiple VMs



Encrypted transfer via PCIe: Sharing the key = sharing the GPU



Single-way attestation: Only a CVM attests a GPU but not vice versa, allowing us to share the GPU to another CVM

System Architecture

A Central Orchestrator

Multiple Data Holders

A Travelling GPU

A Central Orchestrator

- Orchestrates the learning process
- Manages GPU sharing to data-holders with the data buffer address
- Model training and private data scrubbing before handing the GPU to another data holder
- Verification of the integrity of code and model in GPU’s memory

Multiple Data Holders

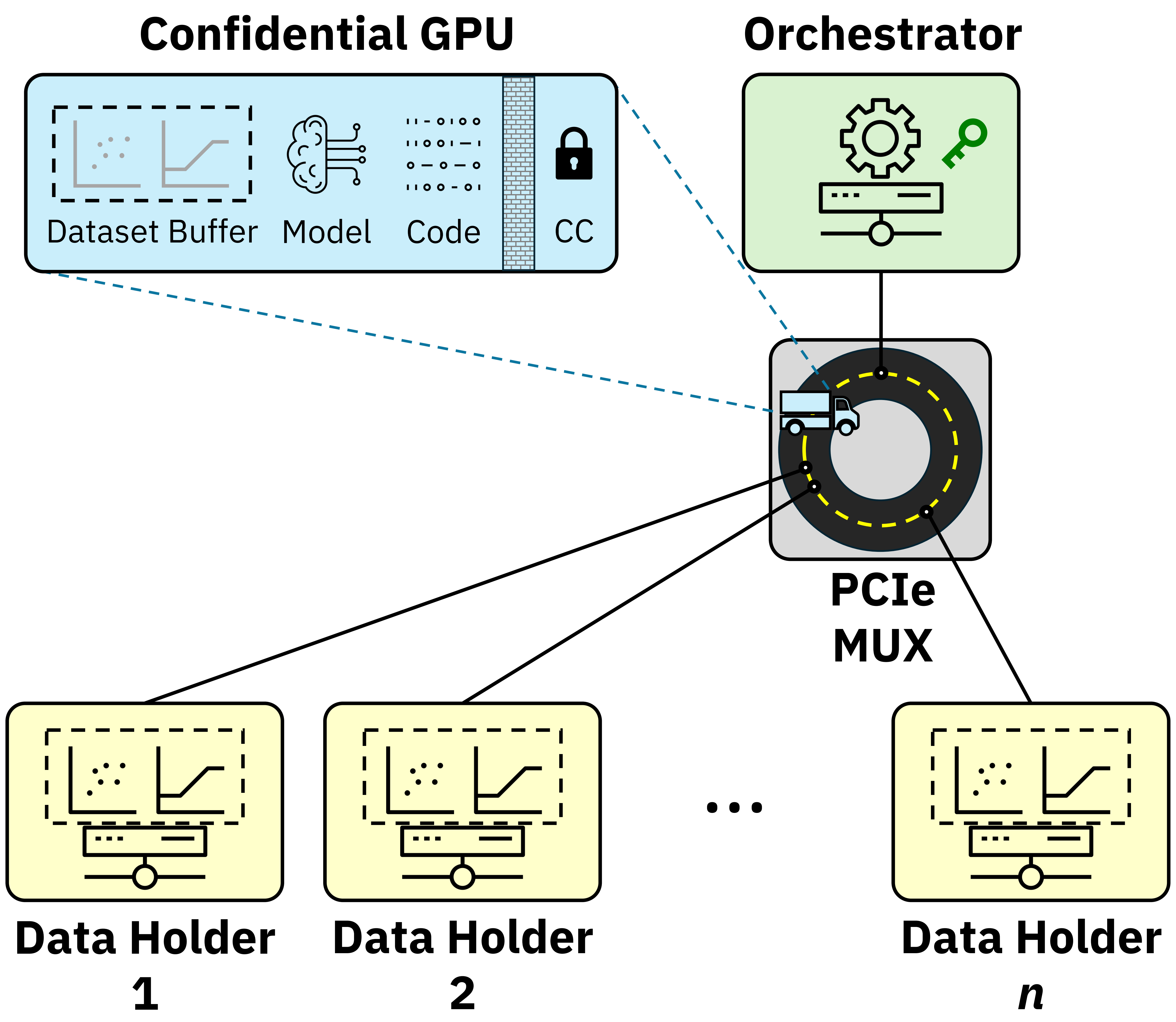
- Provisioning private training data to designated buffer and returning the GPU to orchestrator

A Travelling GPU

- Keeping the model in device memory
- Executing commands from the orchestrator
- Obtaining data from data holders

System Architecture

GPU works like a *truck*, travels to data holders and then come back for training



Threat Model: Who trust whom?

Each data holder only trusts itself, orchestrator and hardware

Orchestrator

- Only trusts the hardware (CPU & GPU)

Data Holders

- Trusts itself
- Trusts the orchestrator
- Trusts the hardware

Who can be malicious?

- Data holders
- Hypervisors (platform provider)
- Outside attackers

Threat Model

Guarantee that each data holder’s dataset is confidential

What is protected?

- Each data holder’s dataset

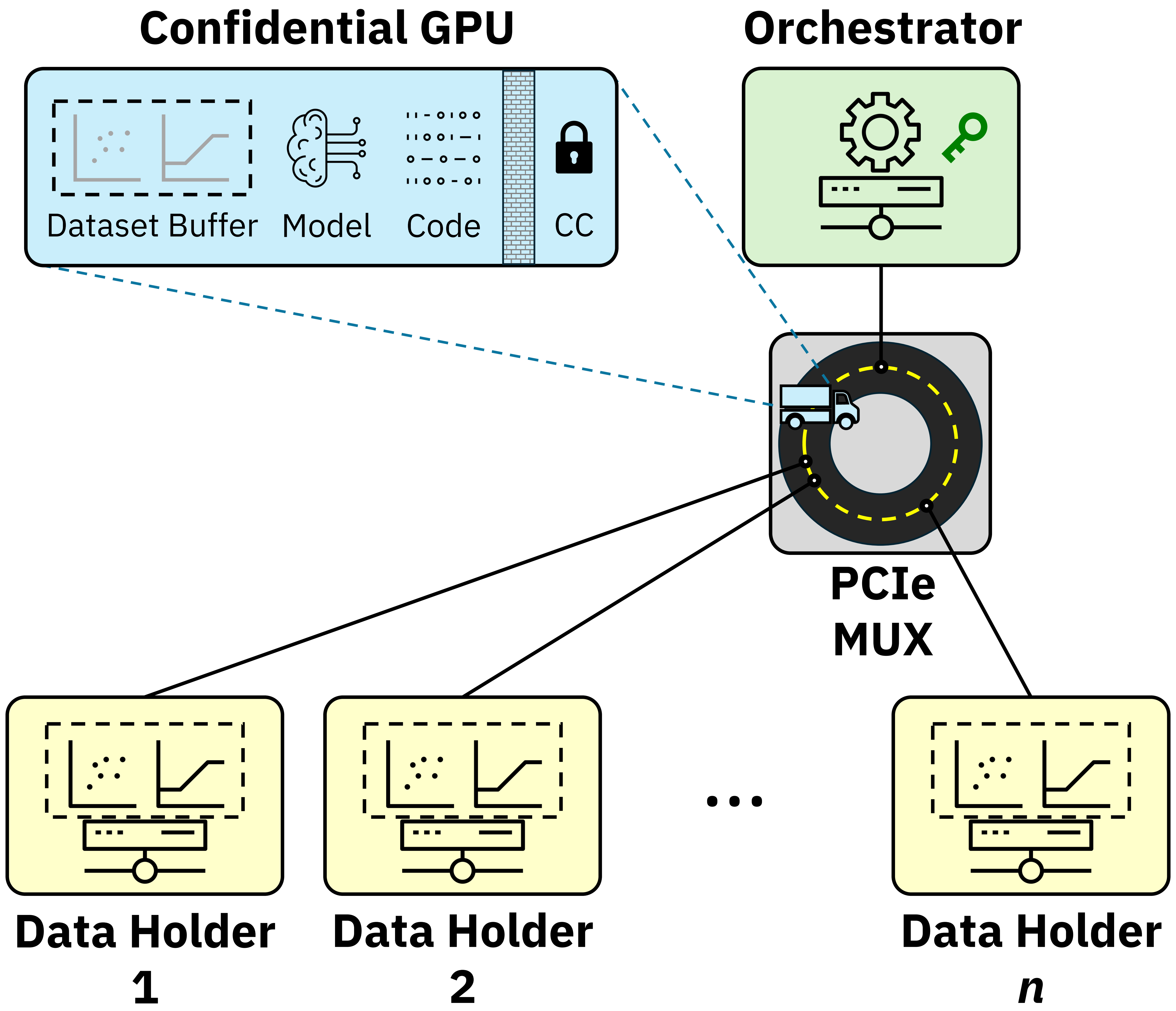
What is not protected?

- The model (not against a malicious data holder)

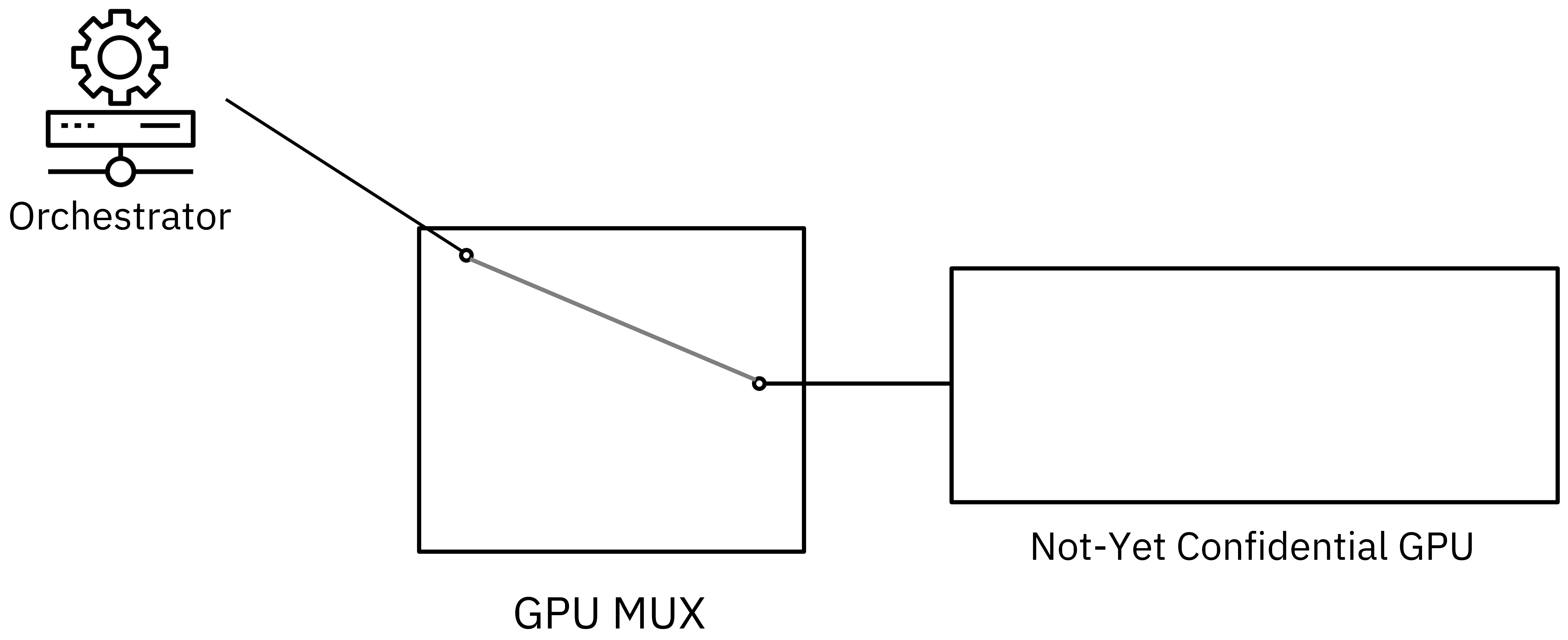
What is out of scope?

- Physical attacks
- Side channels

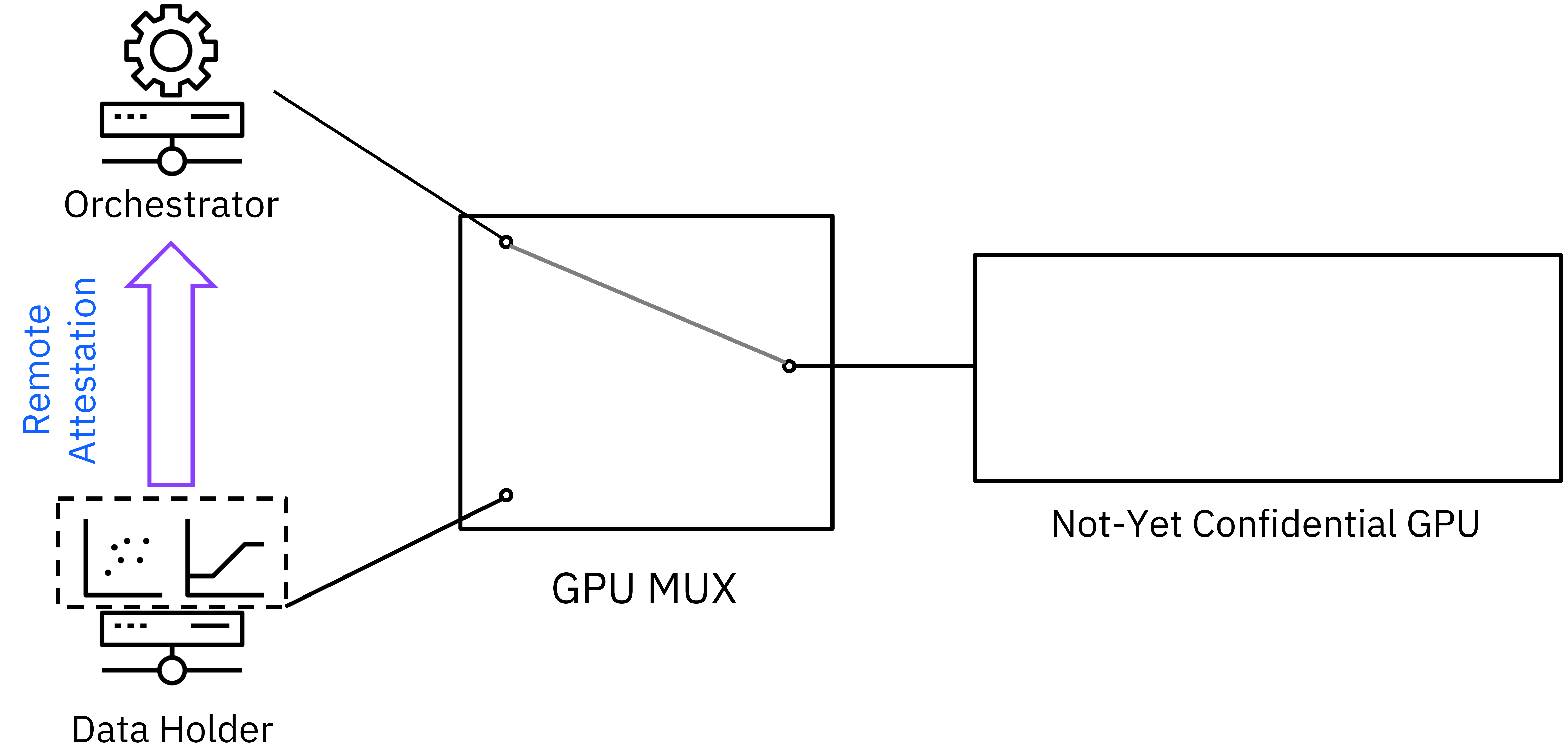
System Architecture



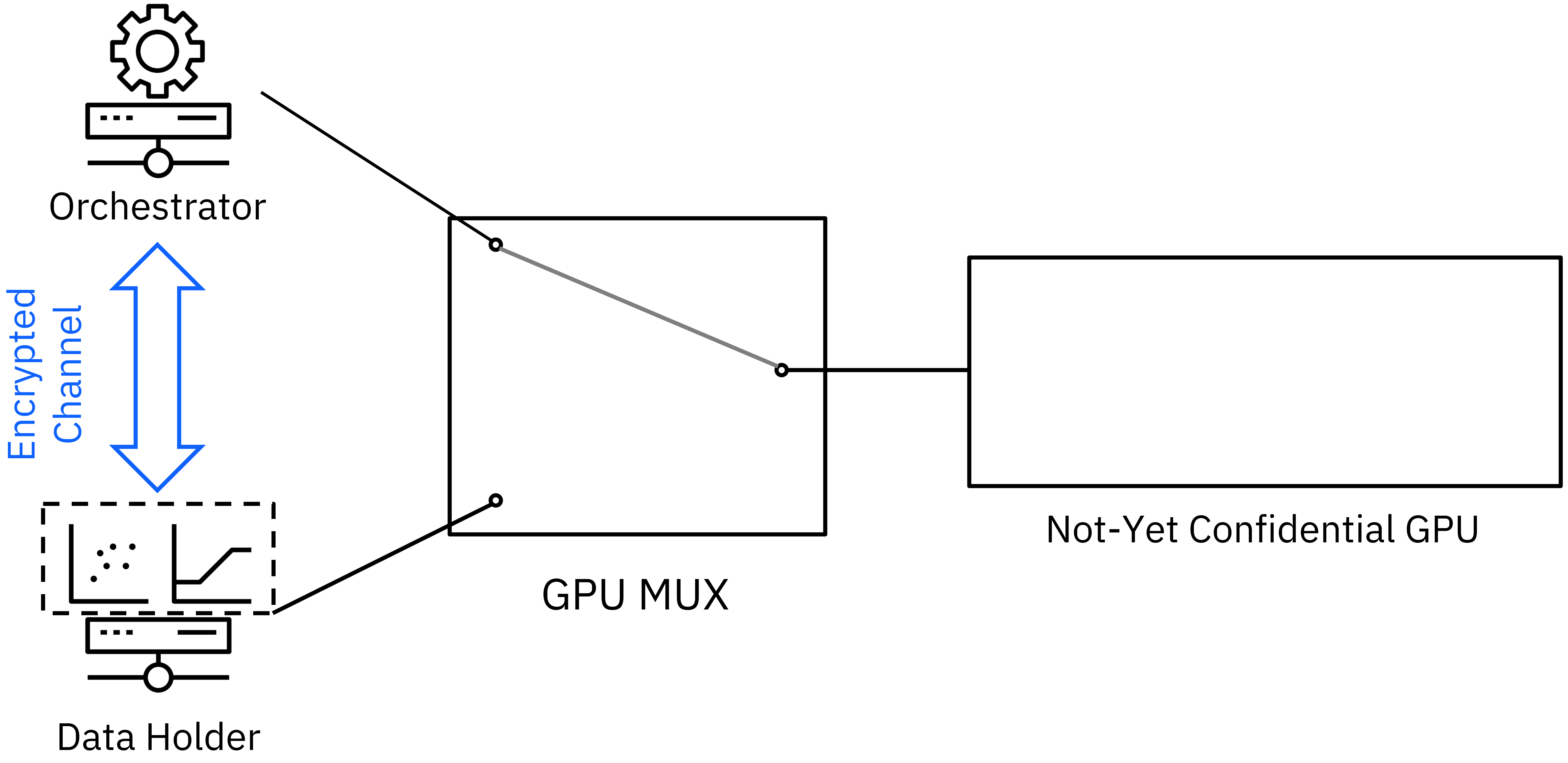
How it works
Booting



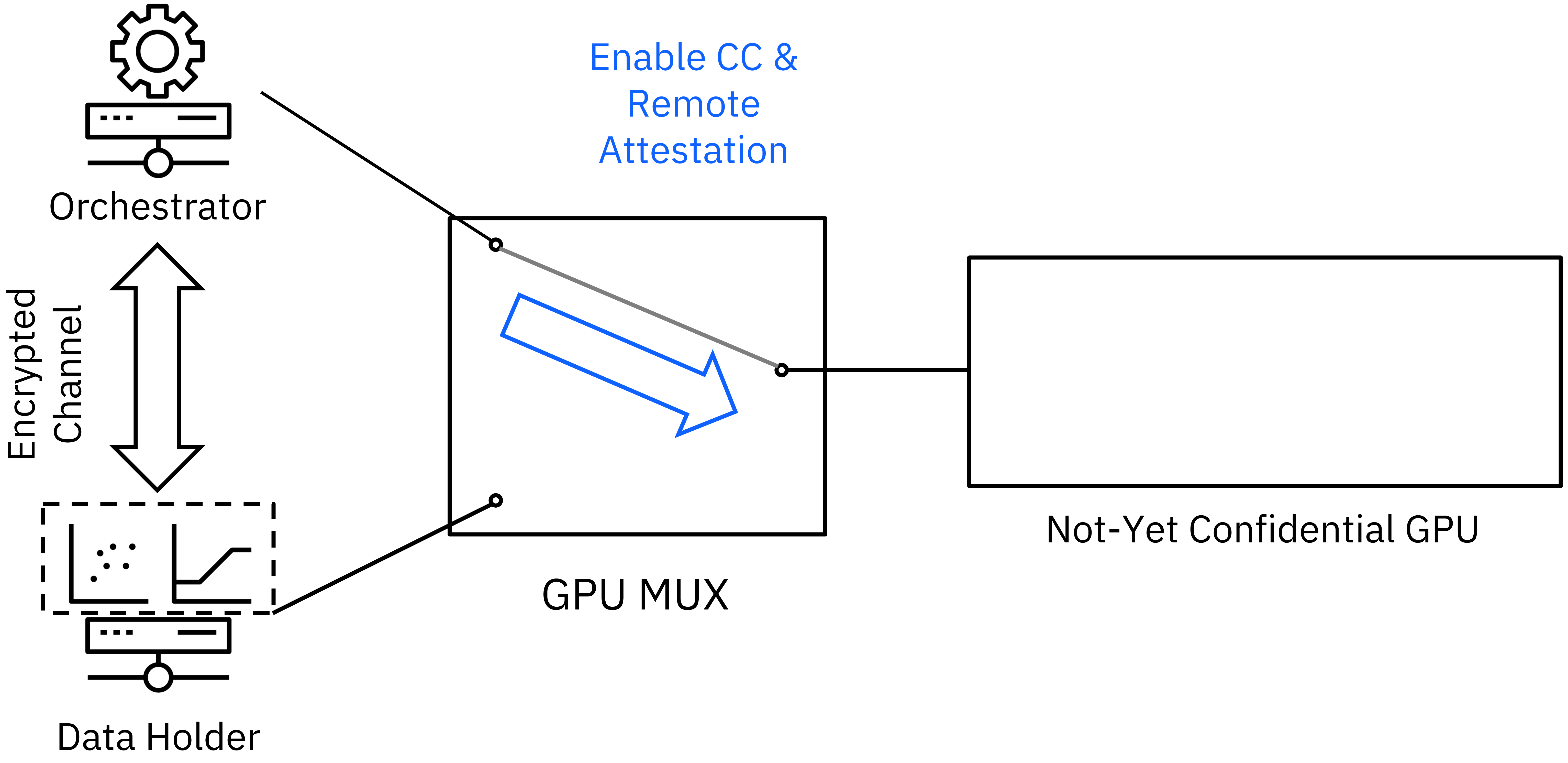
How it works
Booting



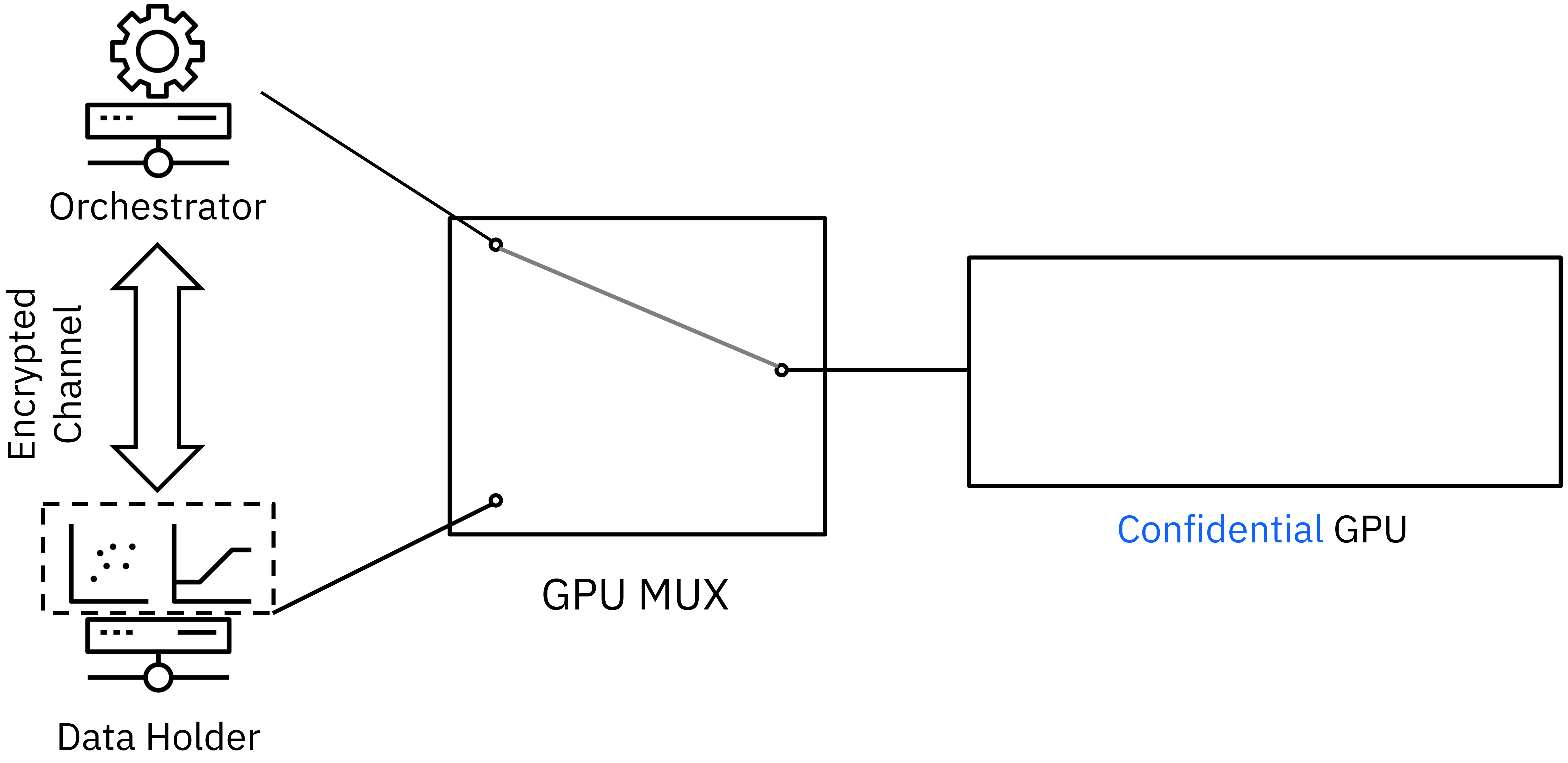
How it works
Booting



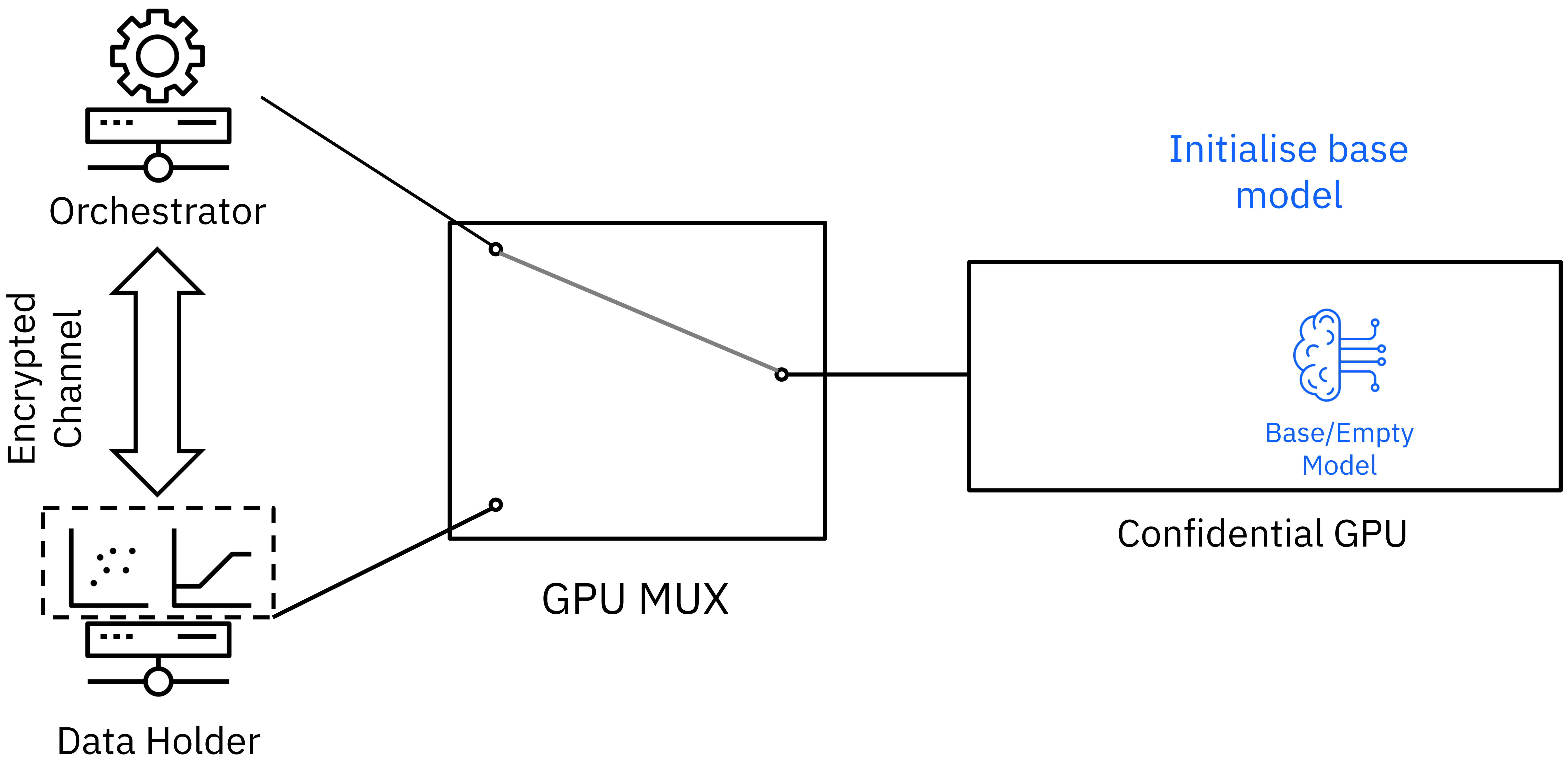
How it works
Booting



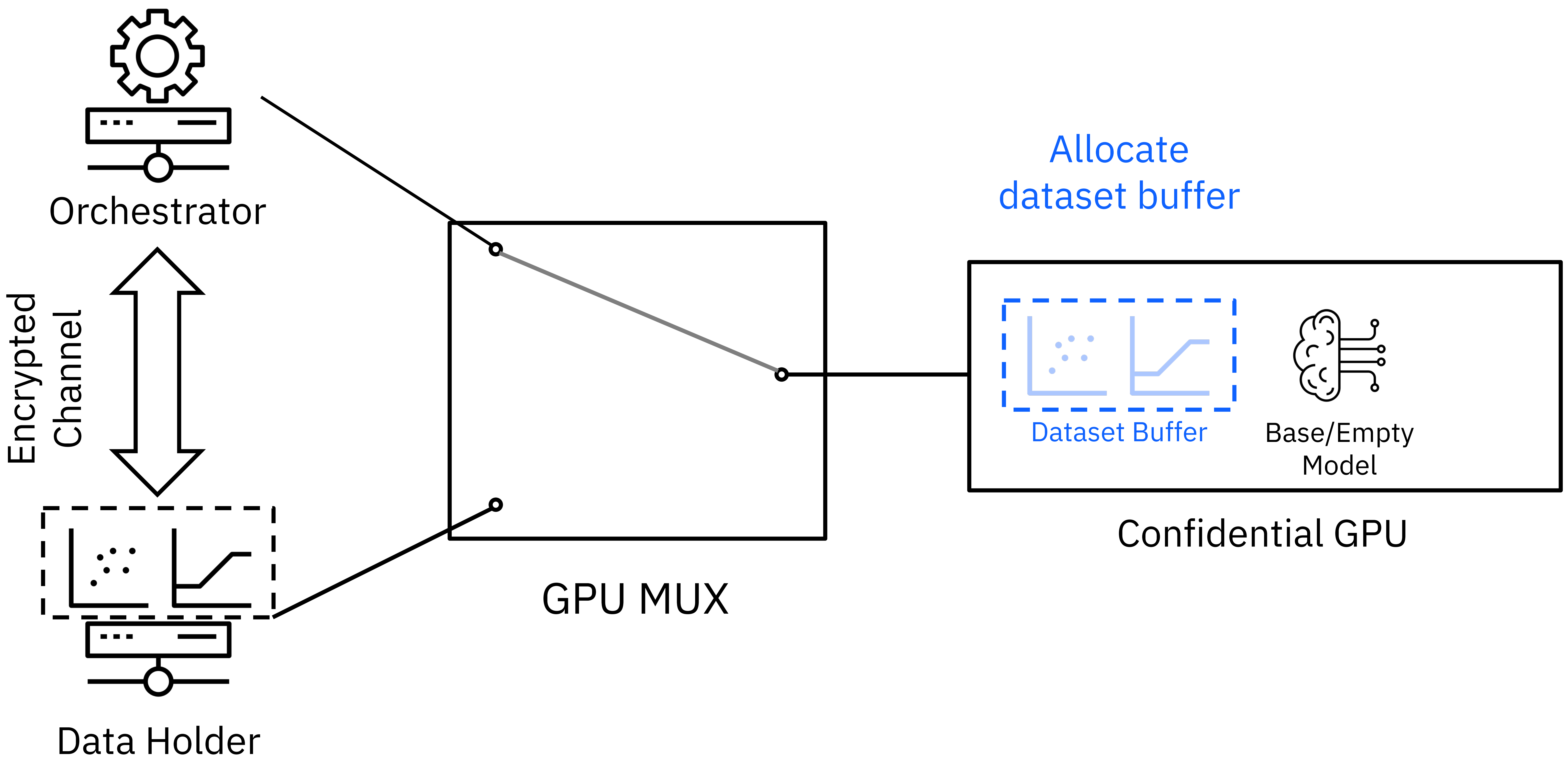
How it works
Booting



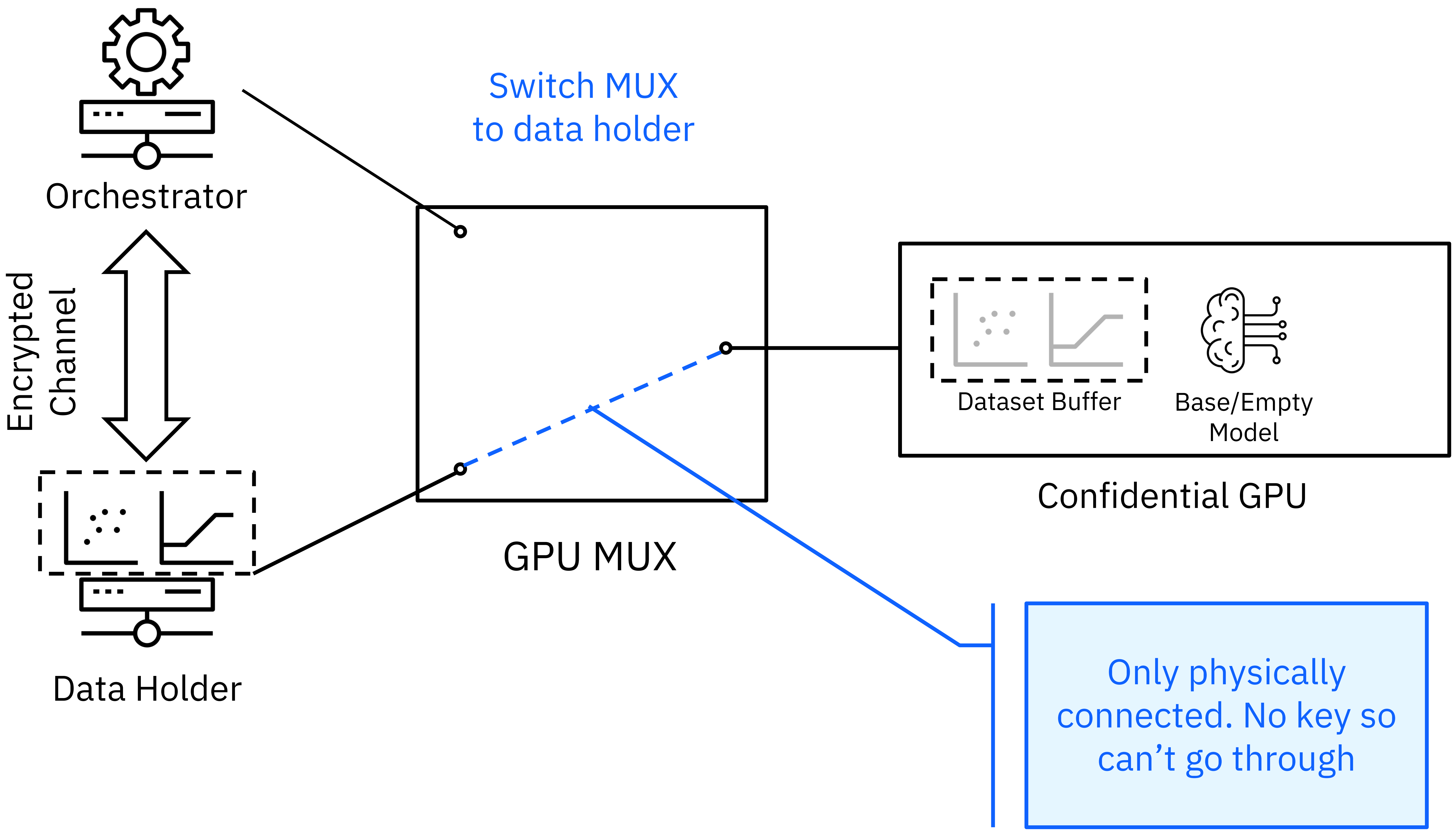
How it works
Setup



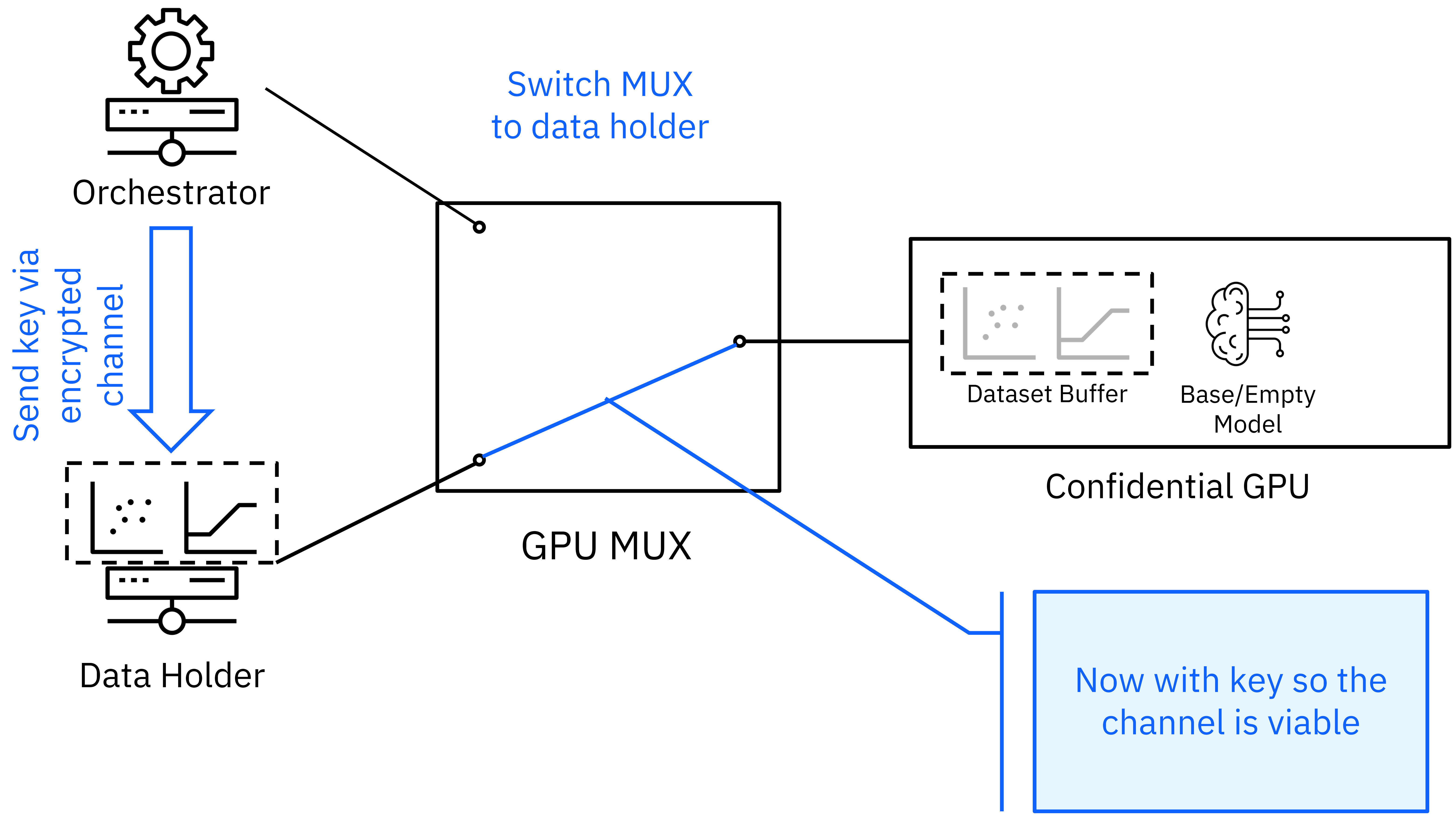
How it works
GPU Switching



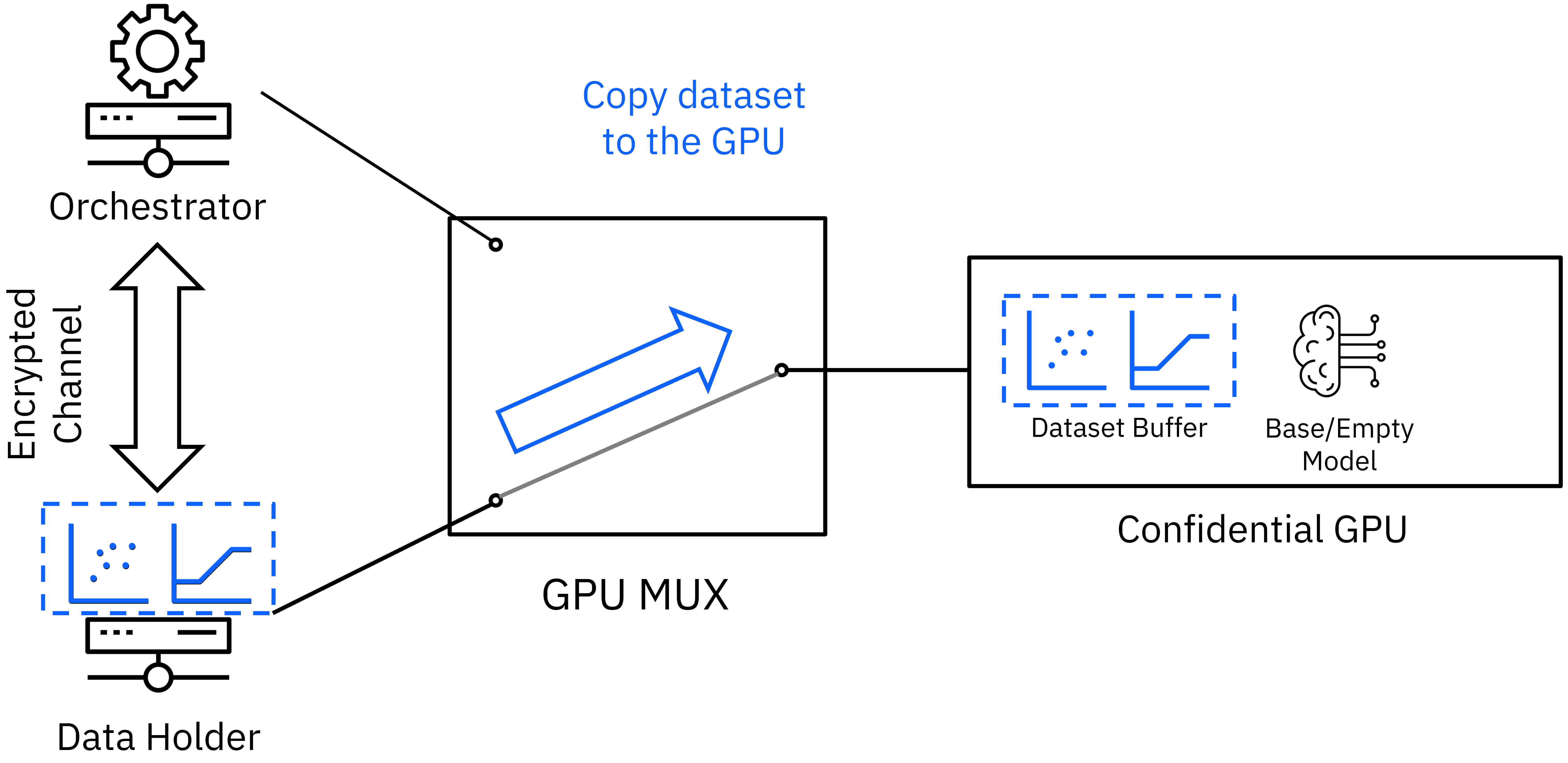
How it works
GPU Switching



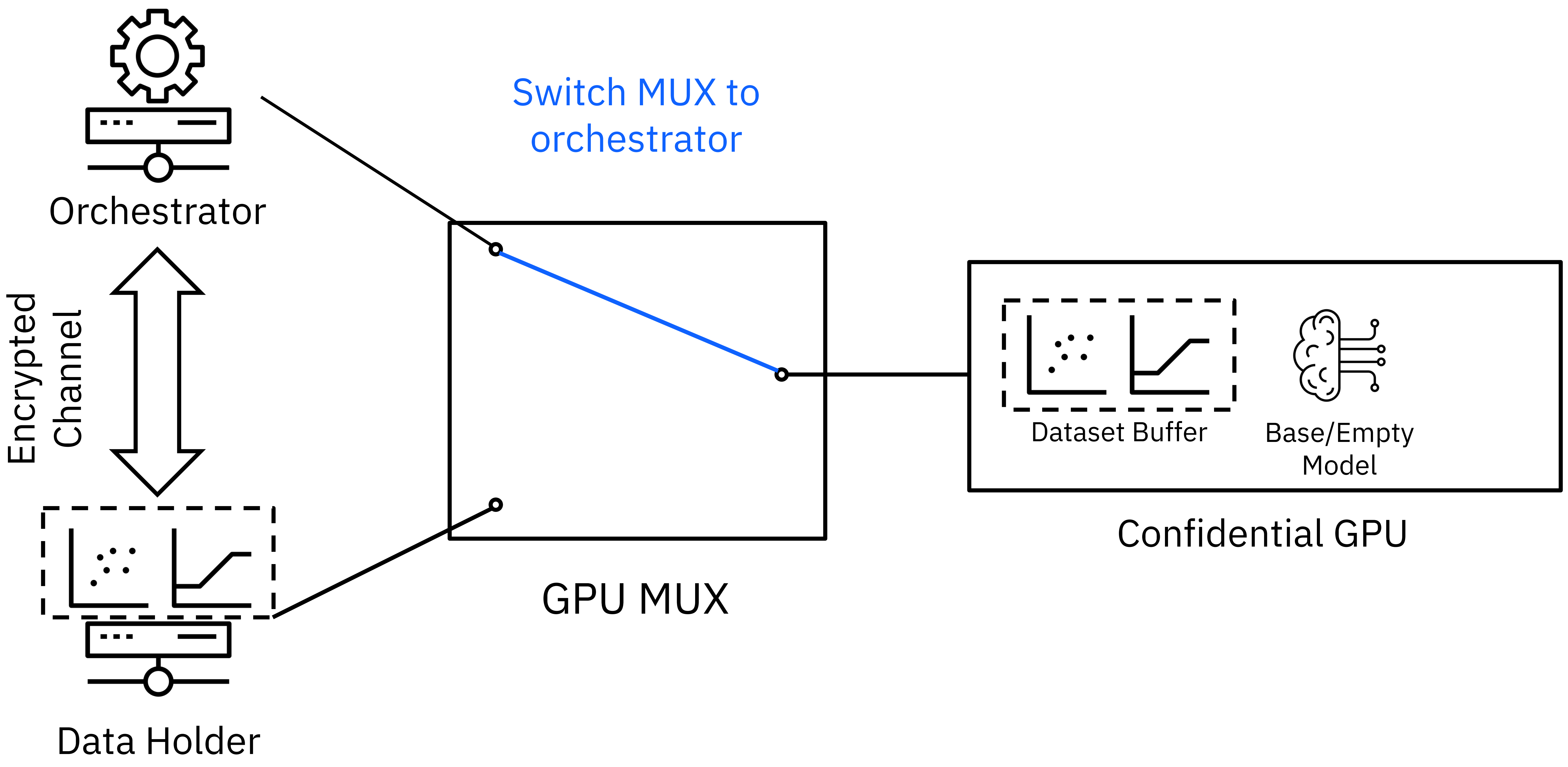
How it works
GPU Switching



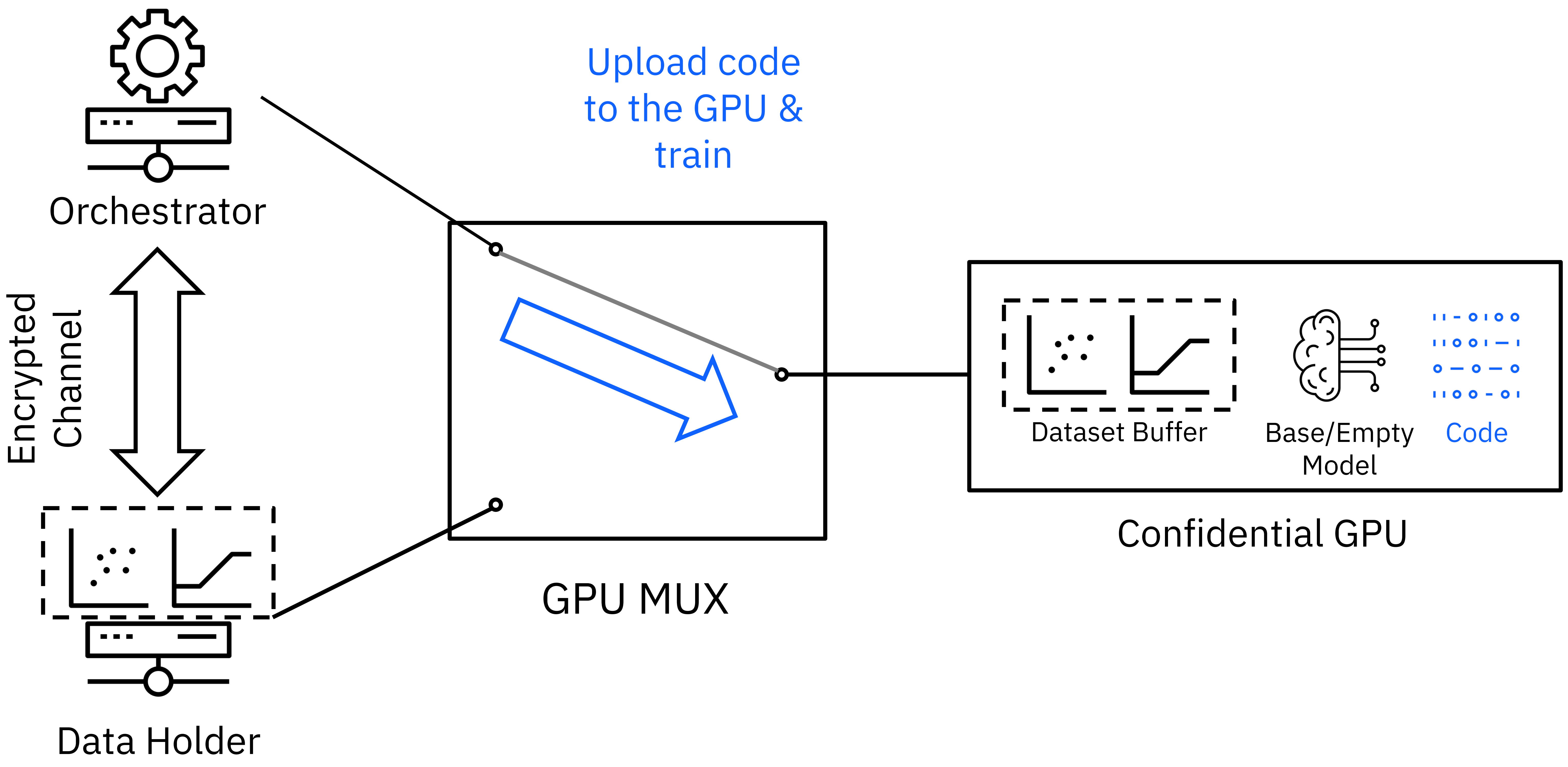
How it works
Data Provisioning



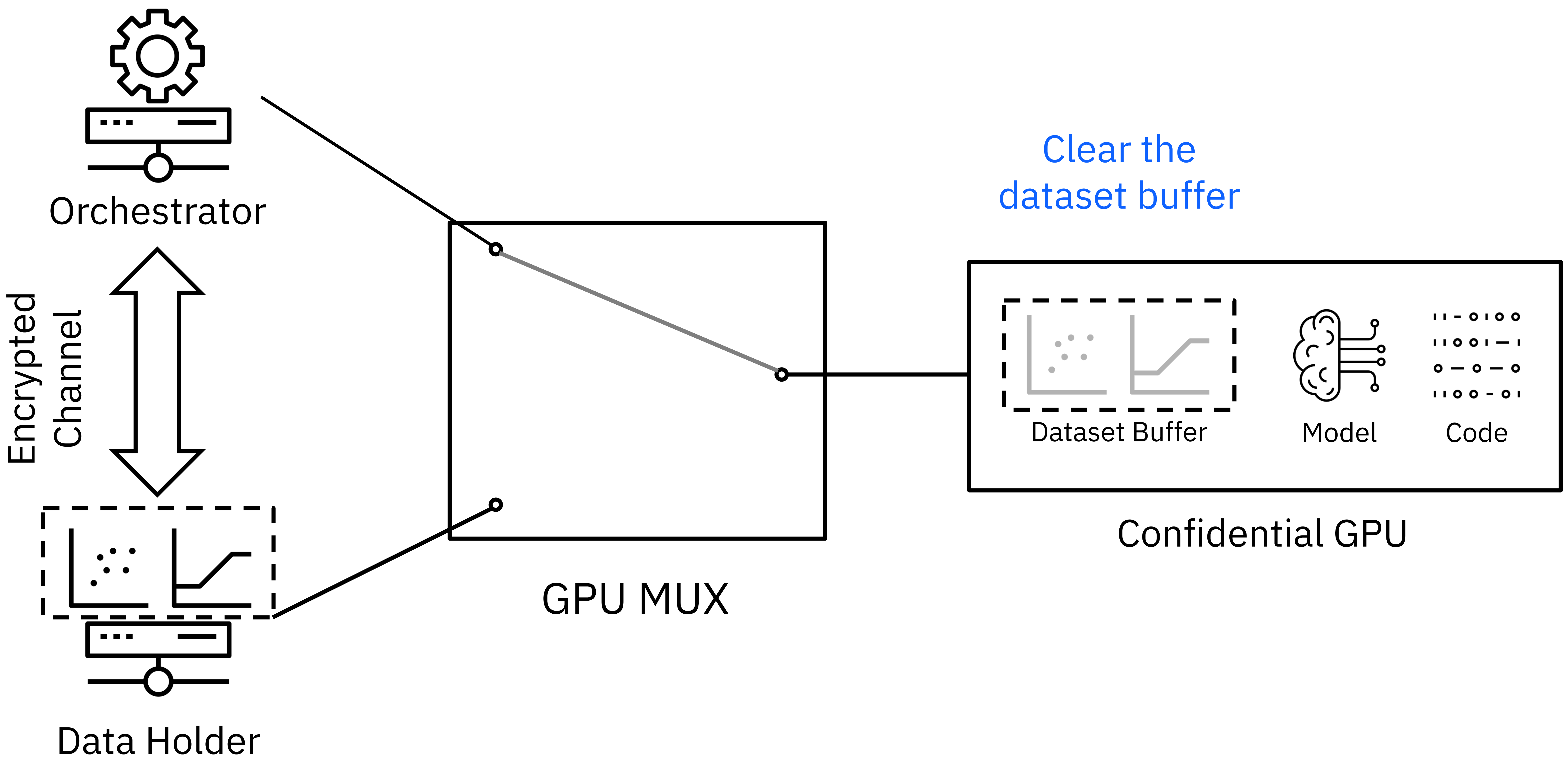
How it works
GPU Switching



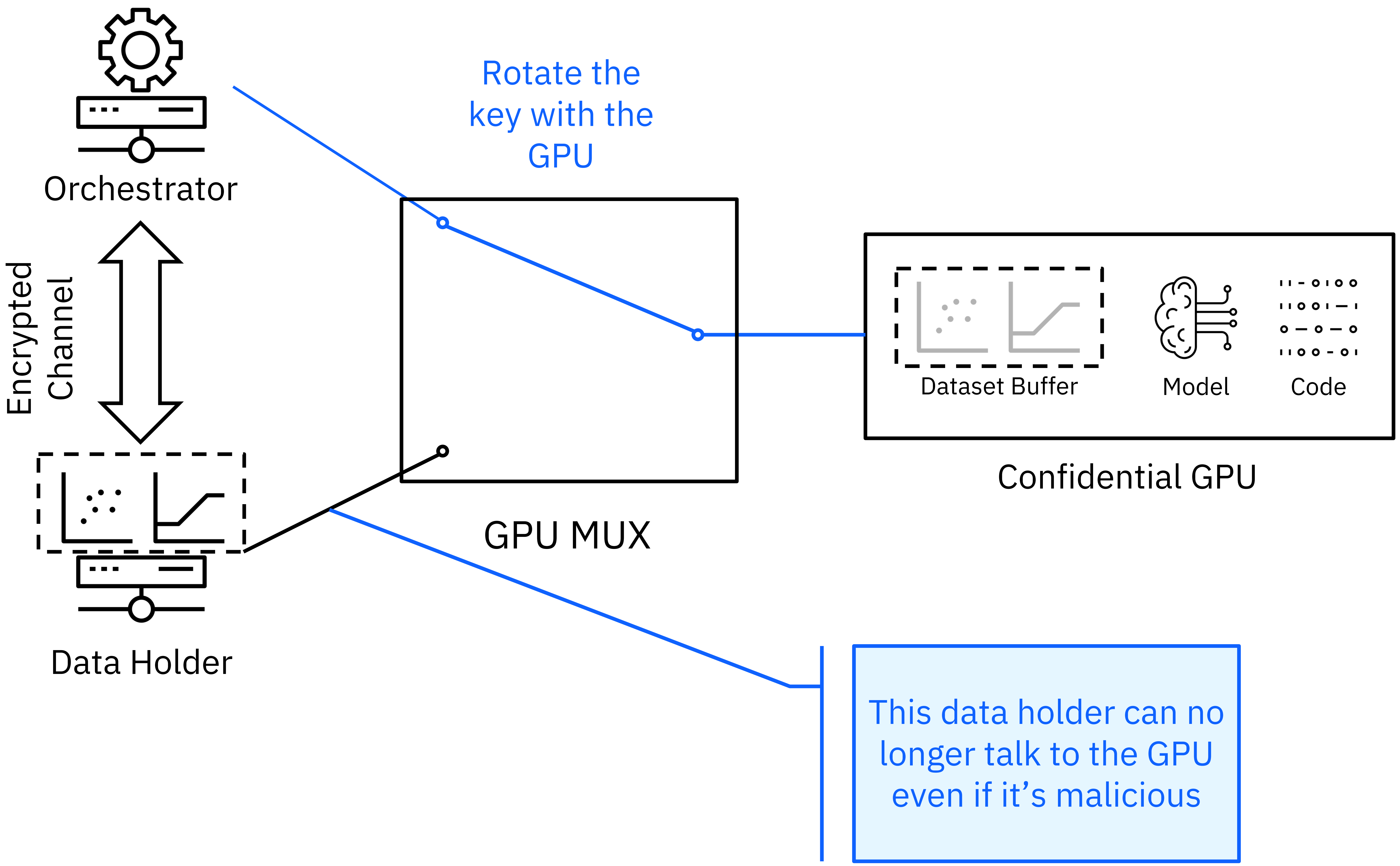
How it works
Training



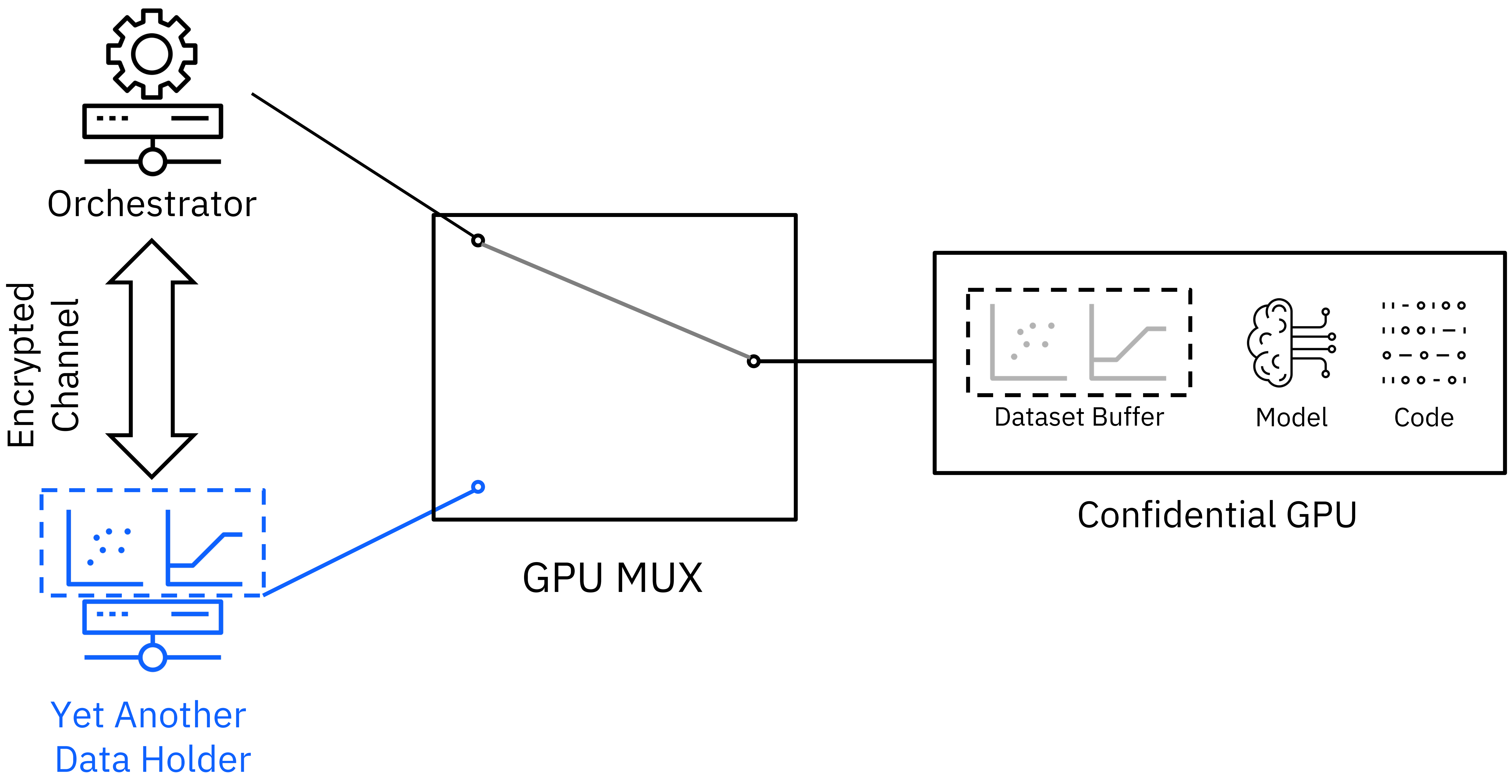
How it works
Epilogue



How it works
Epilogue

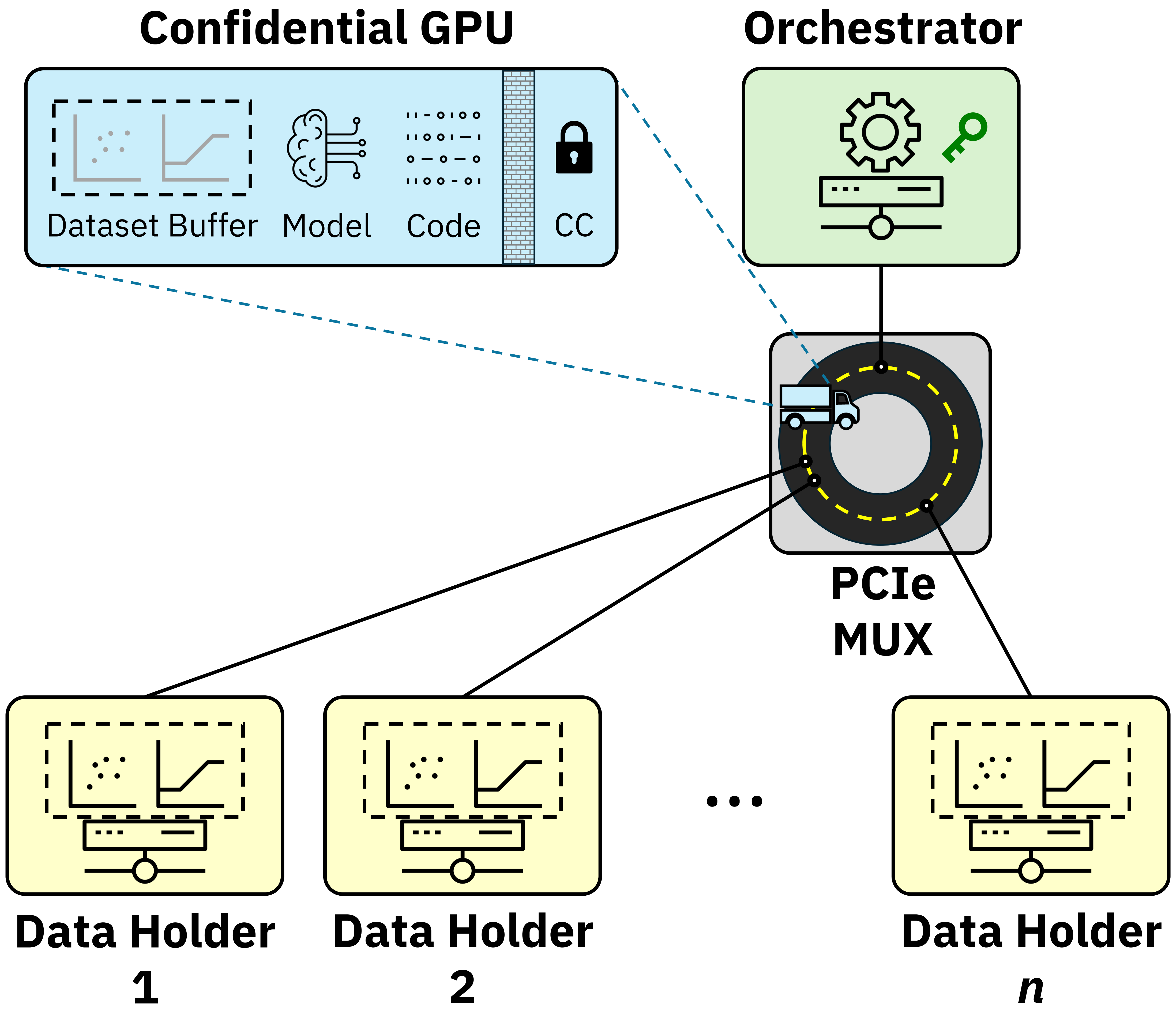


How it works
Next one, please



Ready to be passed to another data holder

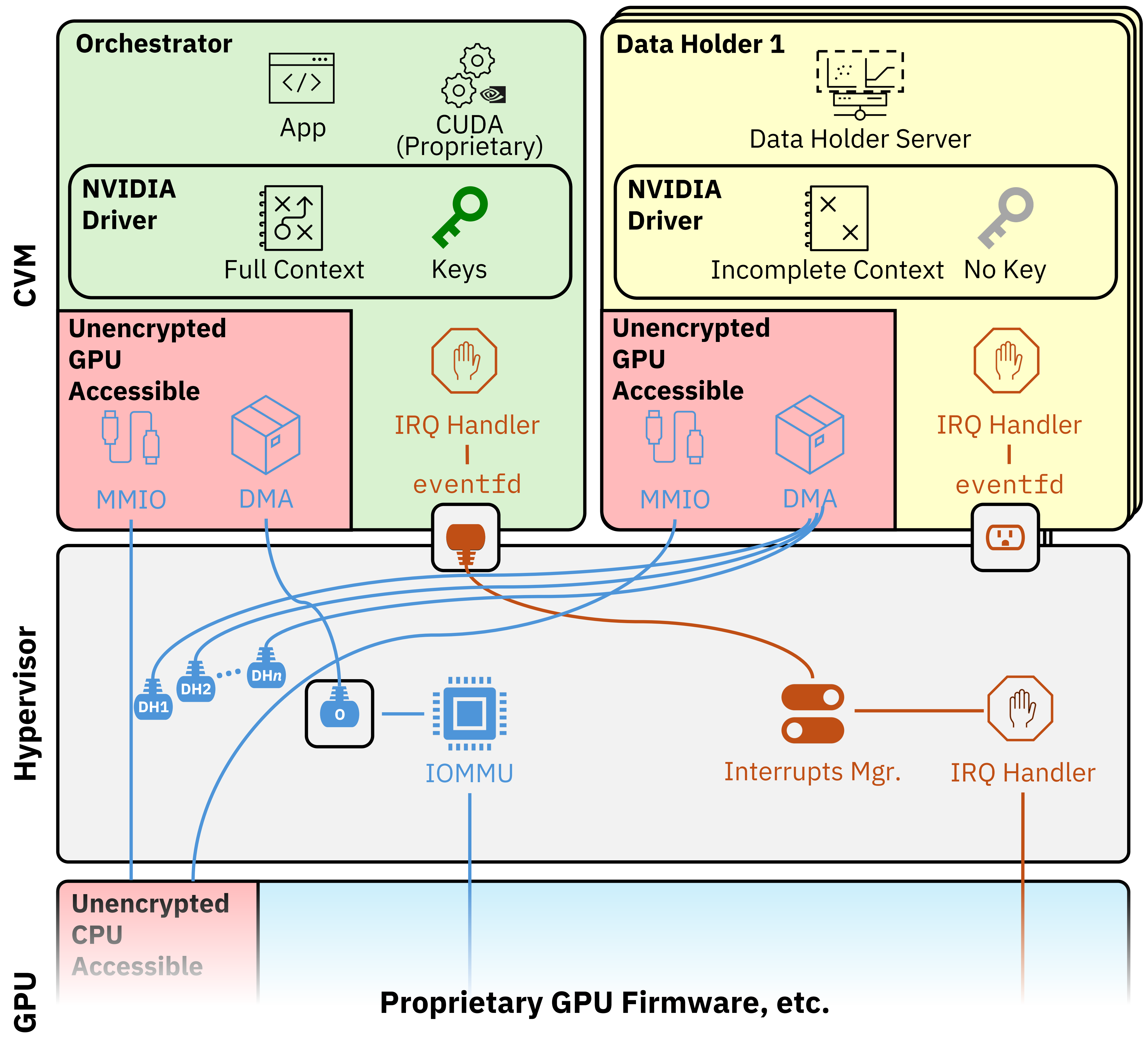
System Architecture



Implementation

Intel TDX

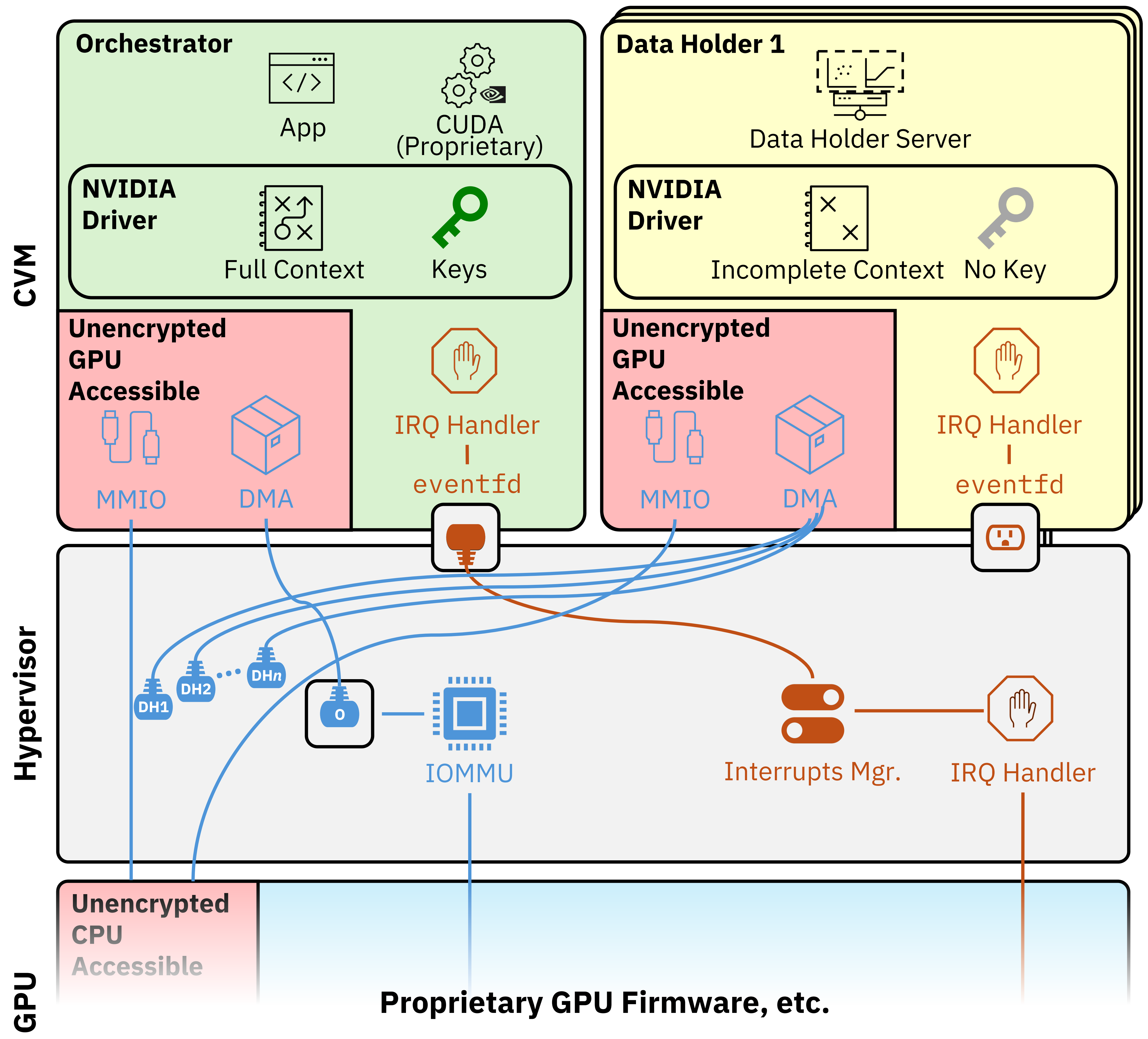
NVIDIA H100



Implementation

Challenge: No changes in proprietary stuff

But man, it's NVIDIA...



Security Analysis

Guarantee that each data holder's dataset is confidential

Malicious Data Holders:

- The dataset buffer is cleared
- Model's address is unknown
- Code is reuploaded each time after GPU returns to the orchestrator

Malicious Hypervisor:

- GPU-CVM communication is encrypted with cryptographical integrity protection
- Fake/malicious GPU can't decrypt the communication

Collusion:

- GPU communication key is rotated each time before travelling to a new data holder
- No way for a data holder to intercept the new data holder's traffic

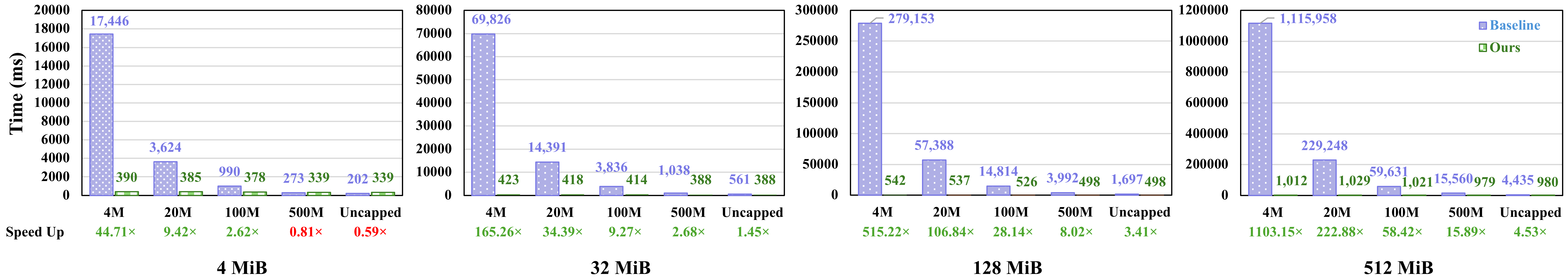
Evaluations

We’ve tried the real deal

llm.c-based demo

- Yes it runs LLM
- Yes we tested on LLM
- Yes it performs like crazy

Evaluations



Data transfer overhead

The bigger the dataset buffer, the faster we are

Evaluations

	Baseline			Ours		
	Training (s)	Tx (s)	Tx Percentage	Training (s)	Tx (s)	Tx Percentage
4M	1230.00	1115.88	47.568%	1230.26	1.01	0.082%
20M	1231.39	230.84	15.787%	1229.39	1.03	0.084%
100M	1231.17	60.36	4.674%	1230.57	1.02	0.083%
500M	1230.73	15.86	1.272%	1229.67	0.98	0.080%
Uncapped	1229.36	7.34	0.594%	1231.46	0.98	0.079%

llm.c comparison w/ GPT-2

You save at least 7 seconds per 256 MiB buffer

Fineweb is 44 TiB size

One epoch saves you 1261568 s (14+ days)

Outlook

Can we do more if…

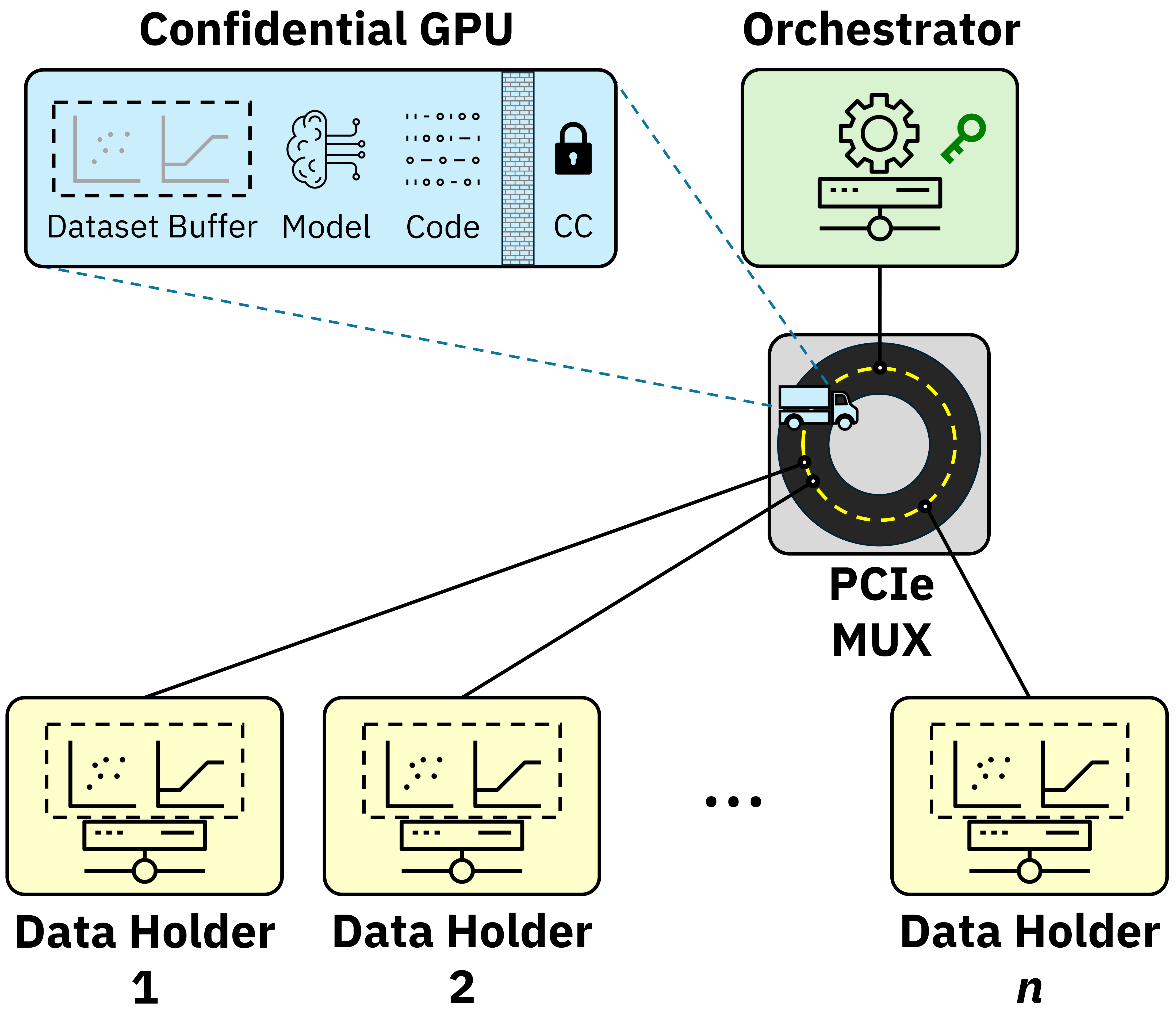
Hardware PCIe switches

- Go beyond one server into the whole data centre

Change proprietary firmware

- Limit data holder’s access to only the dataset buffer
- CUDA context migration: Can achieve backup orchestrator

Q&A


Confidential GPU
Dataset Buffer
Model
Code
CC
Orchestrator
PCIe
MUX
...
Data Holder
1
Data Holder
2
Data Holder
n