nuro

March 2025

End-to-end optimization of onboard autonomy workloads

Sebastian Jodłowski

Intro: Nuro.ai

01

Autonomy for all.
All roads, all rides.

02

Scalable Level 4 autonomous driving system, vehicle-platform agnostic.

03

Build on top of NVIDIA Thor automotive SoC and NVIDIA DriveOS 7 software platform.

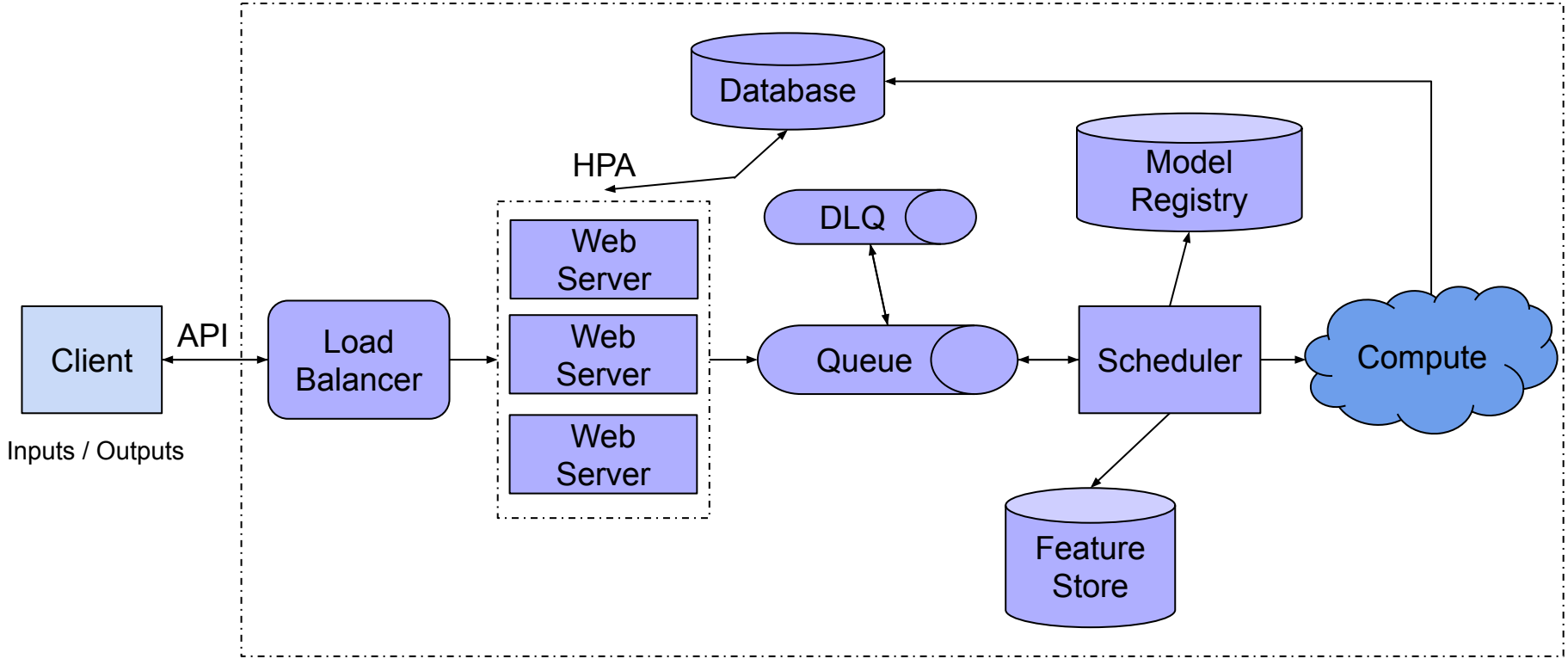
Content

Inference: Cloud vs Car

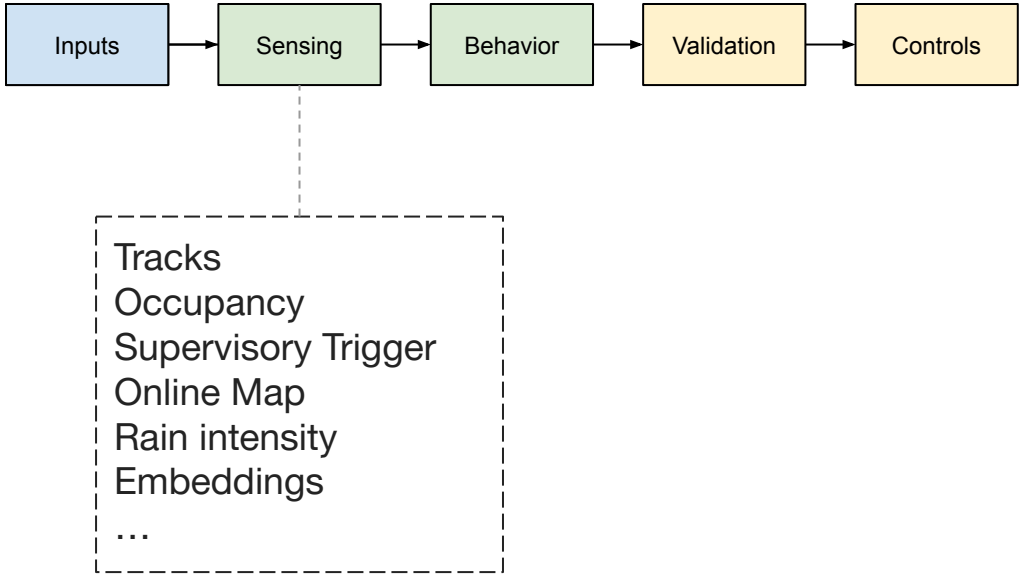
01

Inference in the cloud: strawman service

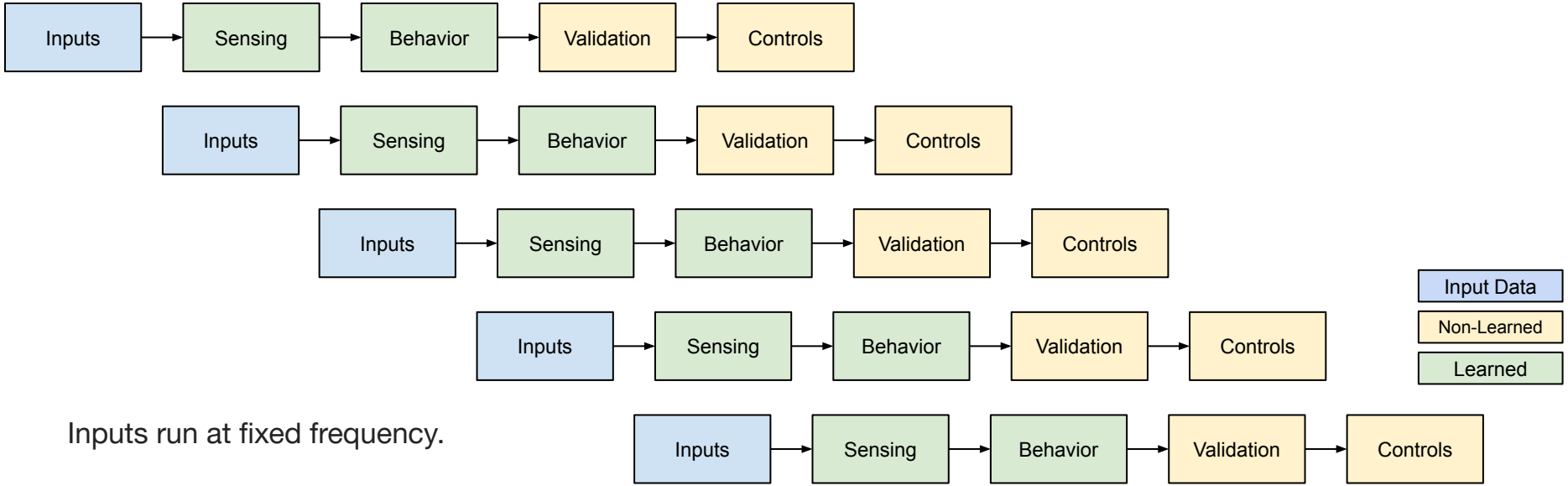
Let's focus on the online inference - closer to the AV use case.



Inference in the car: Autonomy invocation



Inference in the car: Autonomy pipeline



Inputs run at fixed frequency.

Cloud versus Car

Missing the latency gate - for 7x9s, @10Hz:
- 1 in 10m invocations
- once per ~227h
- @15mph: once per ~4155k miles

Category	Cloud	Car
COMPUTE	Flexible	Fixed
WORKLOAD	Flexible	Fixed
INPUT SIZE	Flexible	Fixed
FREQUENCY	Flexible	Fixed
KEY METRIC	Latency: P99, but P50 as well	Latency: P99.99999…

Determinism

02

Determinism

Fighting system contention

02

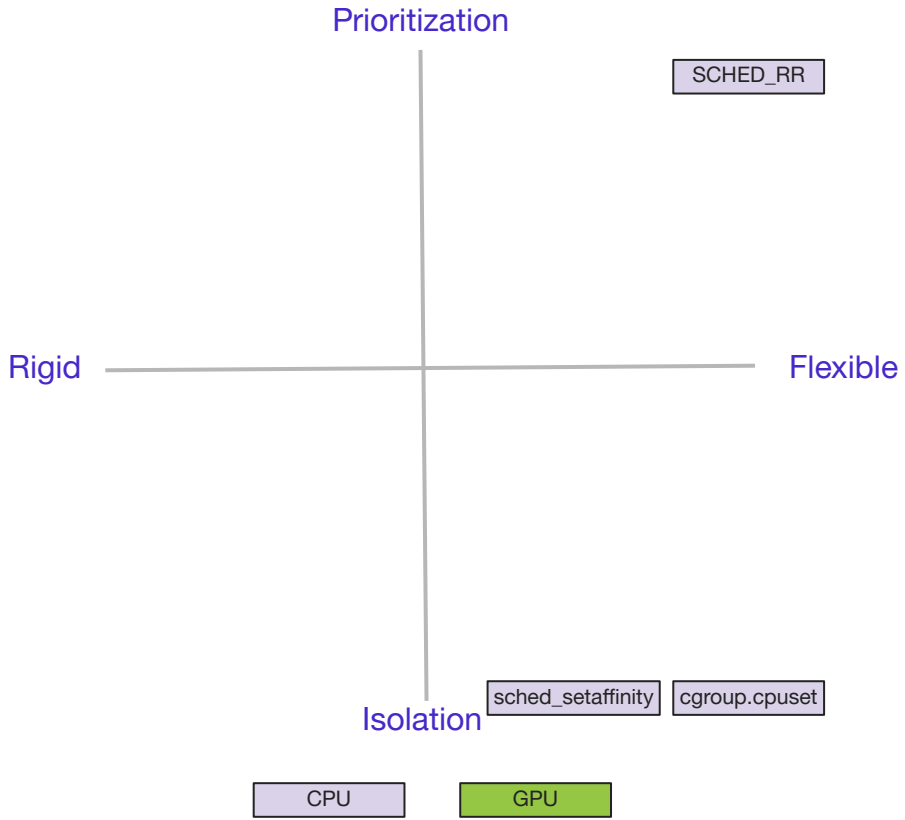
CPU mechanisms

Very strong scheduling support:

- SCHED_RR
  - Allows for real-time scheduling policy
  - Tasks of the same priority execute in cyclic order
    - 100 levels allow for fine-grained configuration
  - Fixed time-slice per task

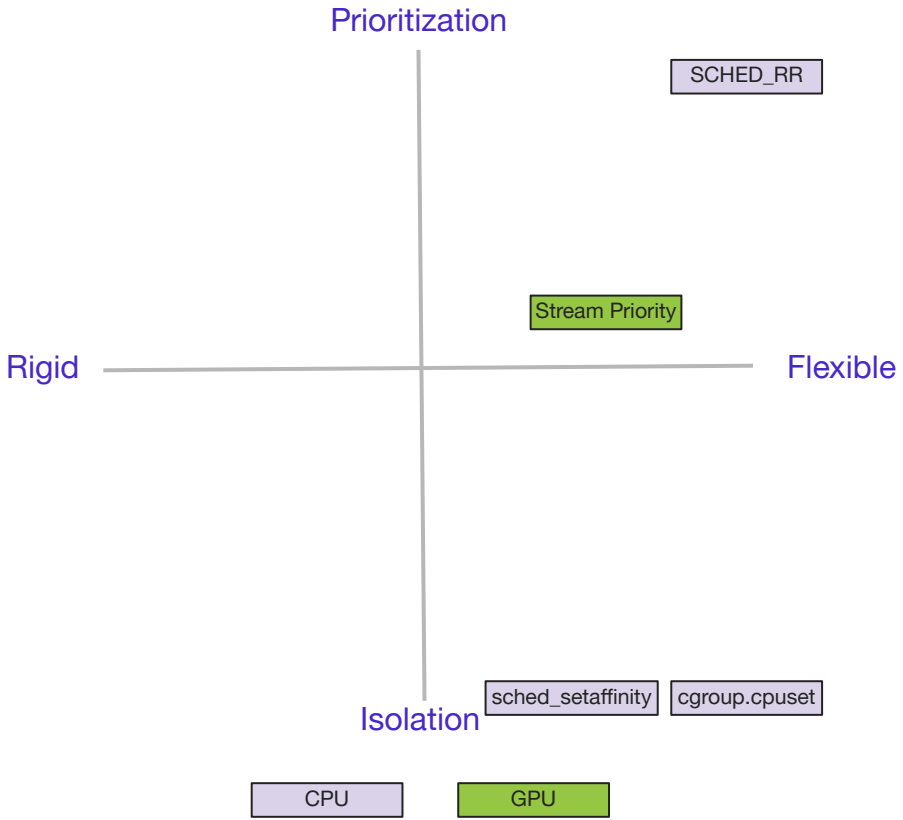
Very strong resource isolation support:

- cgroup.cpuset
  - Allows for core isolation between processes
- sched_setaffinity
  - Allows for core isolation between threads within a process



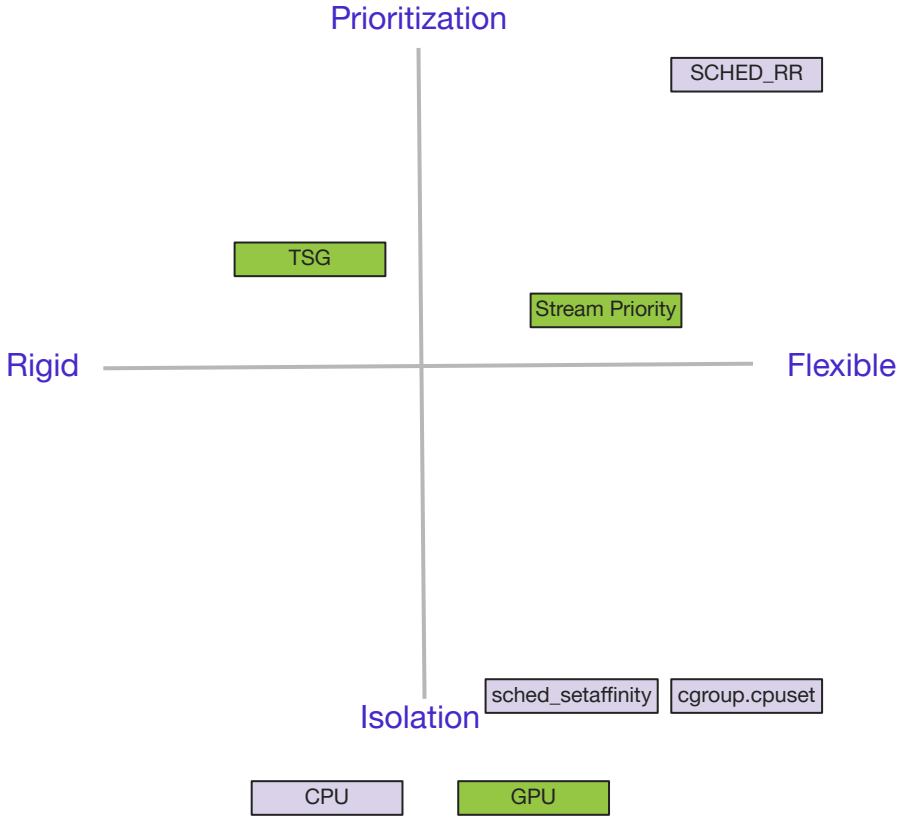
GPU mechanisms: Stream Priority

- Allows to control kernel priority at a stream granularity only within the same context
- Only 8 levels
- Extremely difficult to benefit from due to priority selection being applied at the block scheduler level
  - The block must be ready to fire!



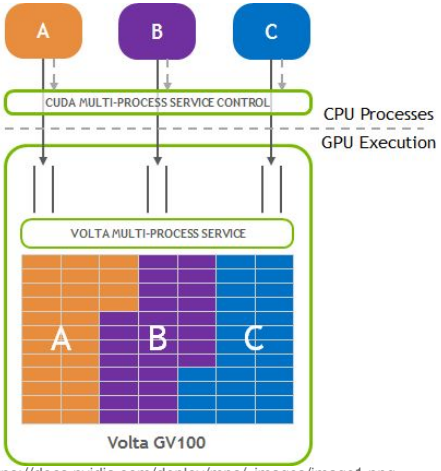
GPU mechanism: TSG (Time-Slice Group)

- Allows to logically group contexts that share a common scheduling policy
- Controls the priority within the group and the time slice length
  - Only 3 priority levels (L/M/H)
  - Fixed time slice per group
- Feels like poor's man SCHED_RR
  - Because tasks cannot share the GPU

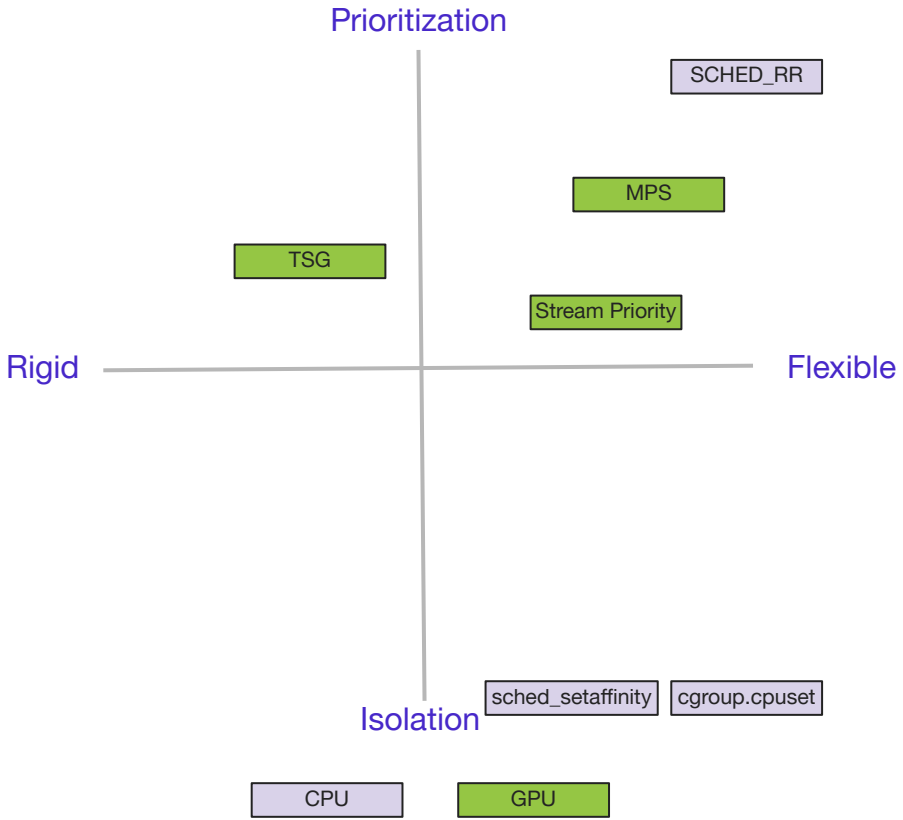


GPU mechanisms: MPS (Multi-Process Service)

- Allows for multiple process to share a single GPU context.
- Allows for limiting the number of SMs process can utilize.
  - `CUDA_MPS_ACTIVE_THREAD_PERCENTAGE`
- Introduced only recently for Tegra! 💚

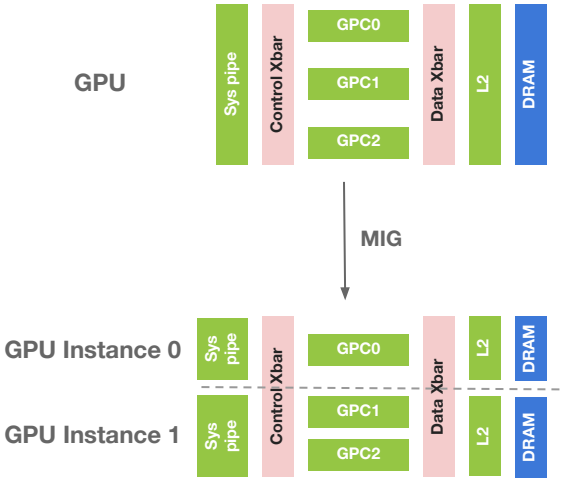


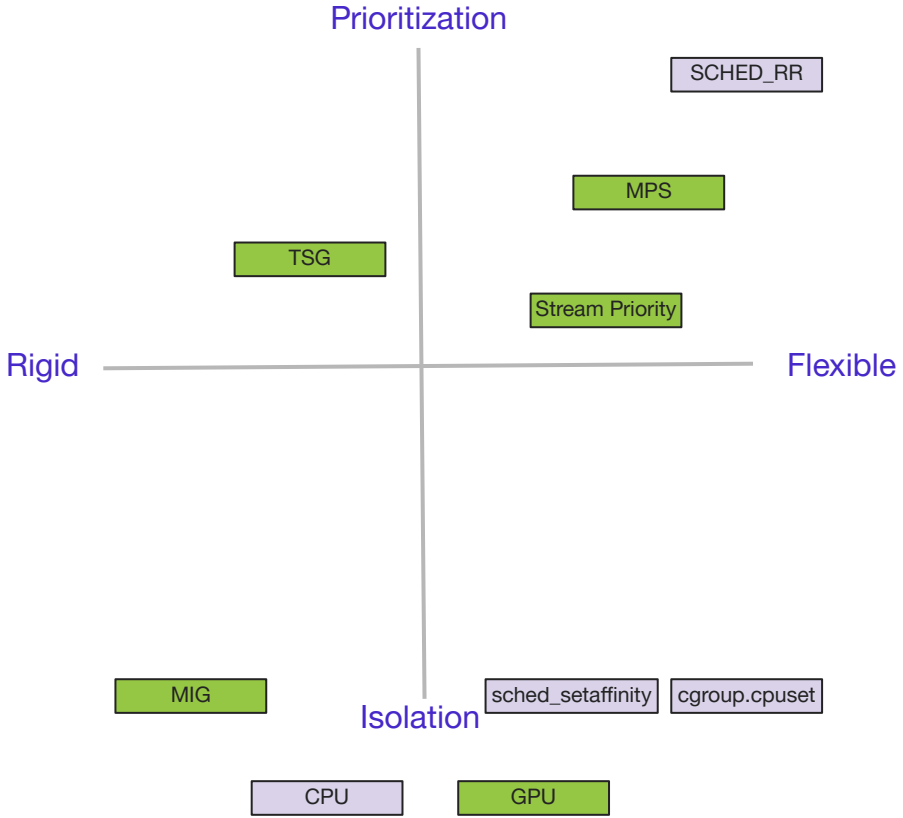
https://docs.nvidia.com/deploy/mps/_images/image1.png



GPU mechanisms: MIG (Multi-Instance GPU)

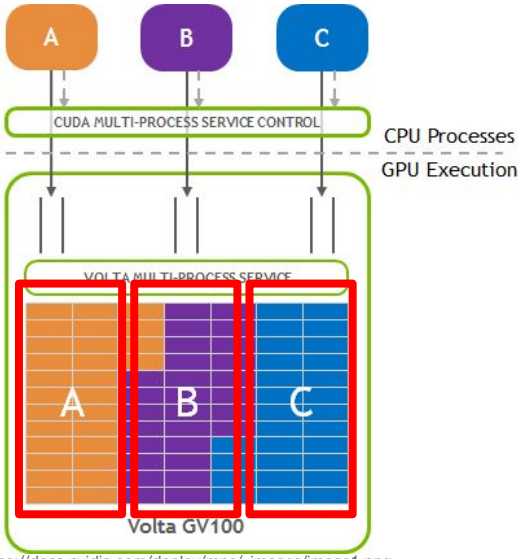
- Allows for a single physical GPU to be partitioned into multiple, smaller, independent GPU instances
- Complete isolation at the hardware level
  - But only single configuration: 1:2 GPCs



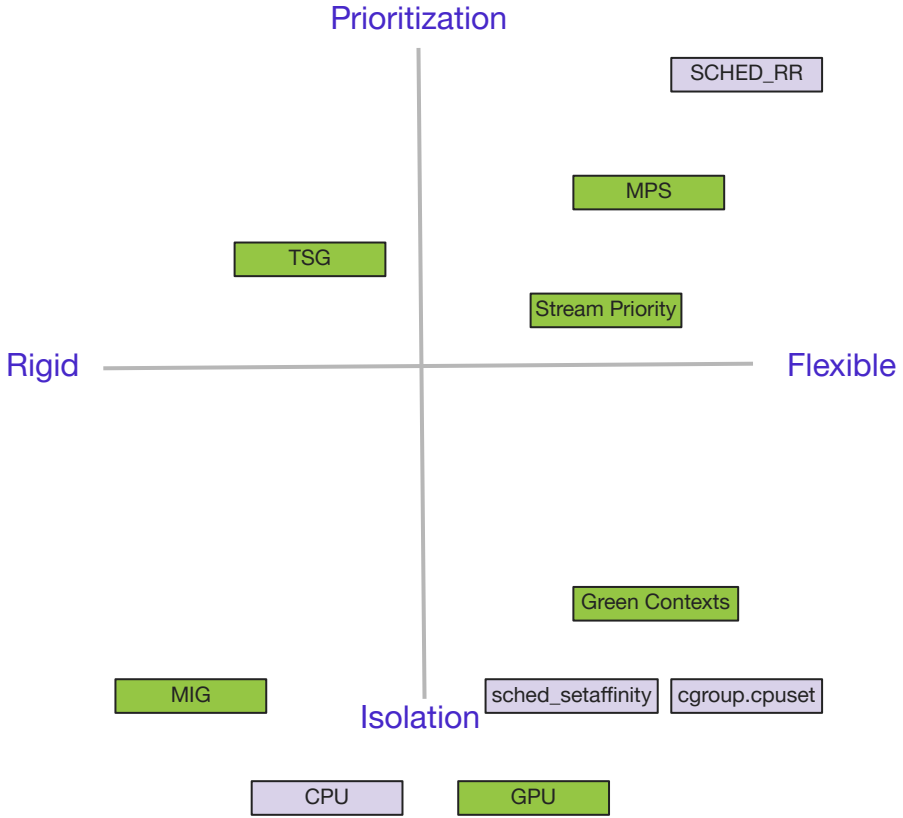


GPU mechanisms: Green Contexts

- Extension to MPS that allows for finer-grain resource allocation
- Carve out a dedicated subset of SMs
  - Not a full static partitioning (yet!)



https://docs.nvidia.com/deploy/mps/_images/image1.png

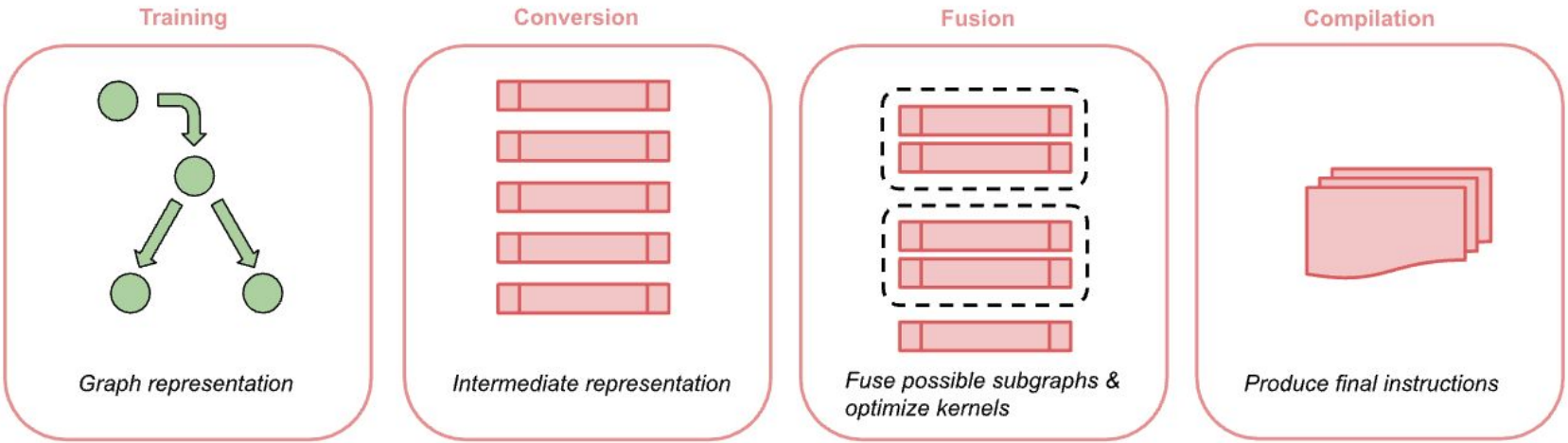


Faster Than Light

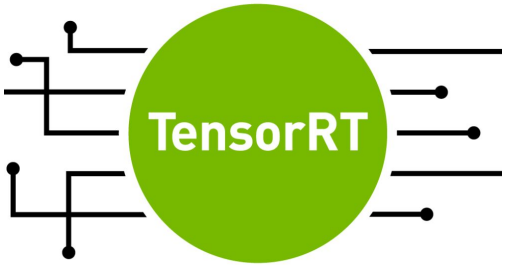
Model Compiler and Interpreter

03

Machine Learning Compilers



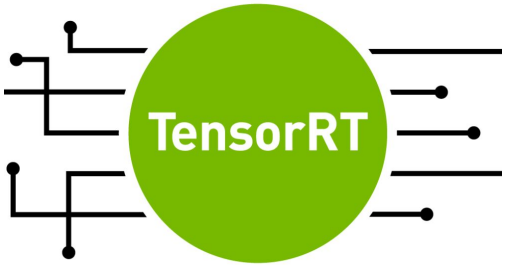
Machine Learning Compilers



Two challenges:

- No clear winner (especially for non-mainstream models)
  - Inference latency
  - Accuracy
  - Unsupported layers
- Unaware of scheduling, prioritization and isolation requirements

Machine Learning Compilers



Two challenges:

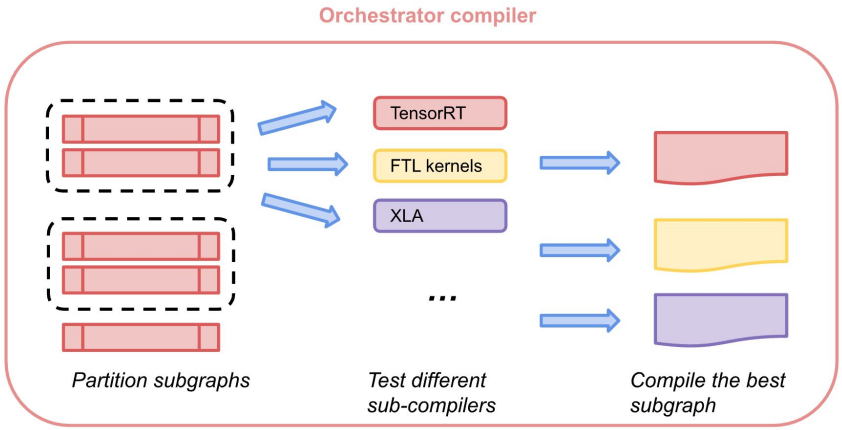
- No clear winner (especially for non-mainstream models)
  - Inference latency
  - Accuracy
  - Unsupported layers
- Unaware of scheduling, prioritization and isolation requirements

FTL Orchestrator Compiler

FTL Model Interpreter

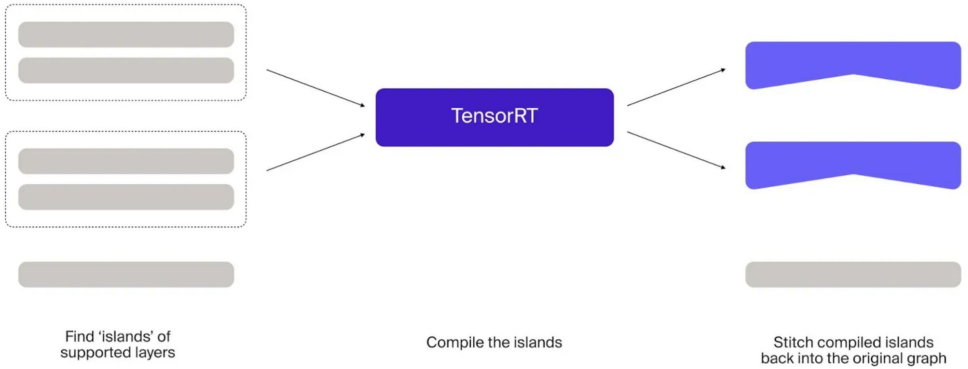
FTL Orchestrator Compiler

- Leverages different industry leading compilers in tandem
- Produces a highly optimized binary that is faster than any single industry compiler
- Based on MLIR
- Highly configurable: programmatically select subset of the model to compile

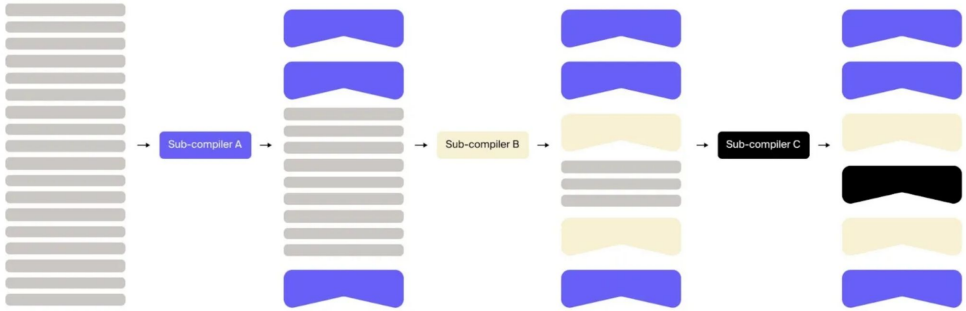


FTL Orchestrator Compiler

- Compile islands: greedily collect layers in topological order, stopping if a certain layer fails a certain predefined criteria

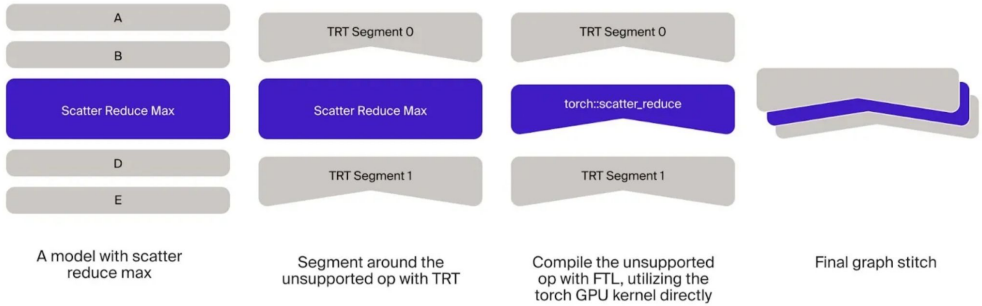


- Stitch multiple sub-compiler islands



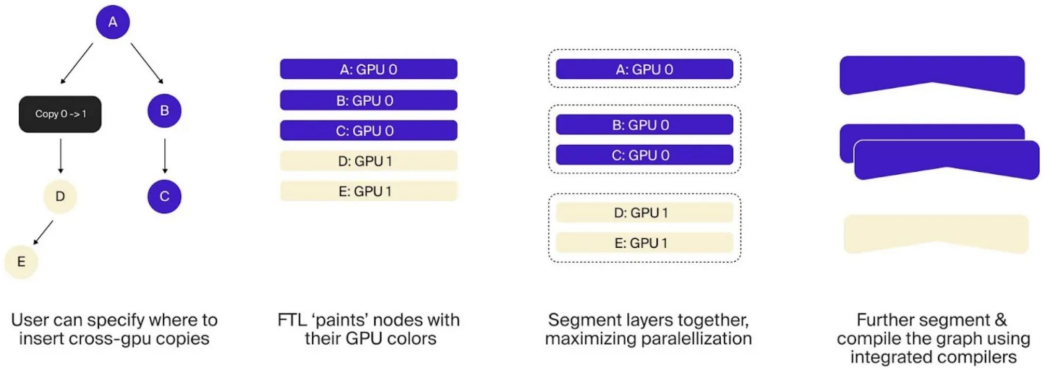
FTL Orchestrator Compiler

- Custom kernel injection for problematic ops
- Supporting CUDA, Triton and Pallas



FTL Model Interpreter

- Allows to specify both pipeline and model parallelism
- Allows for scheduler to prioritize critical model outputs (Stream Priority)
- Allows for scheduler to target isolated compute resources (MPS + Green Context)
- Unlocks flexible frequency control, i.e. running one part of the model less frequently



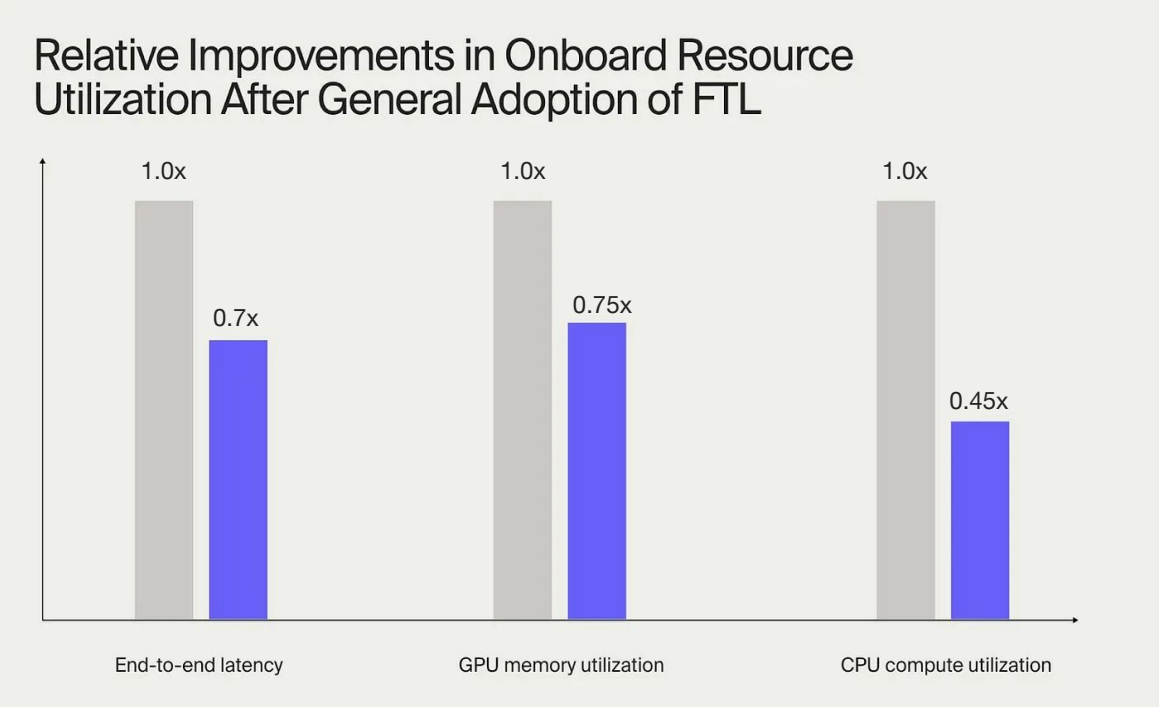
User can specify where to insert cross-gpu copies

FTL 'paints' nodes with their GPU colors

Segment layers together, maximizing paralellization

Further segment & compile the graph using integrated compilers

FTL: In action



Wrap up

- Autonomy workloads are a complex mix of many cooperating models deployed on limited compute resources
- The very-long-tail events are all that matter
- Deterministic latency = resource isolation + scheduling
- GPU mechanisms are a bit lagging behind, but getting there
- Resource-aware scheduling on top of classical model compilers is necessary

Thank you.