Salus

Fine-grained GPU Sharing Primitives for Deep Learning Applications

Advisor: Mosharaf Chowdhury

2020-03-03 By Peifeng Yu

University of Michigan

Deep Learning Becomes Ubiquitous

- Computer vision
- Natural language processing
- Speech
- Robotics

"Ok Google"

Hey Cortana

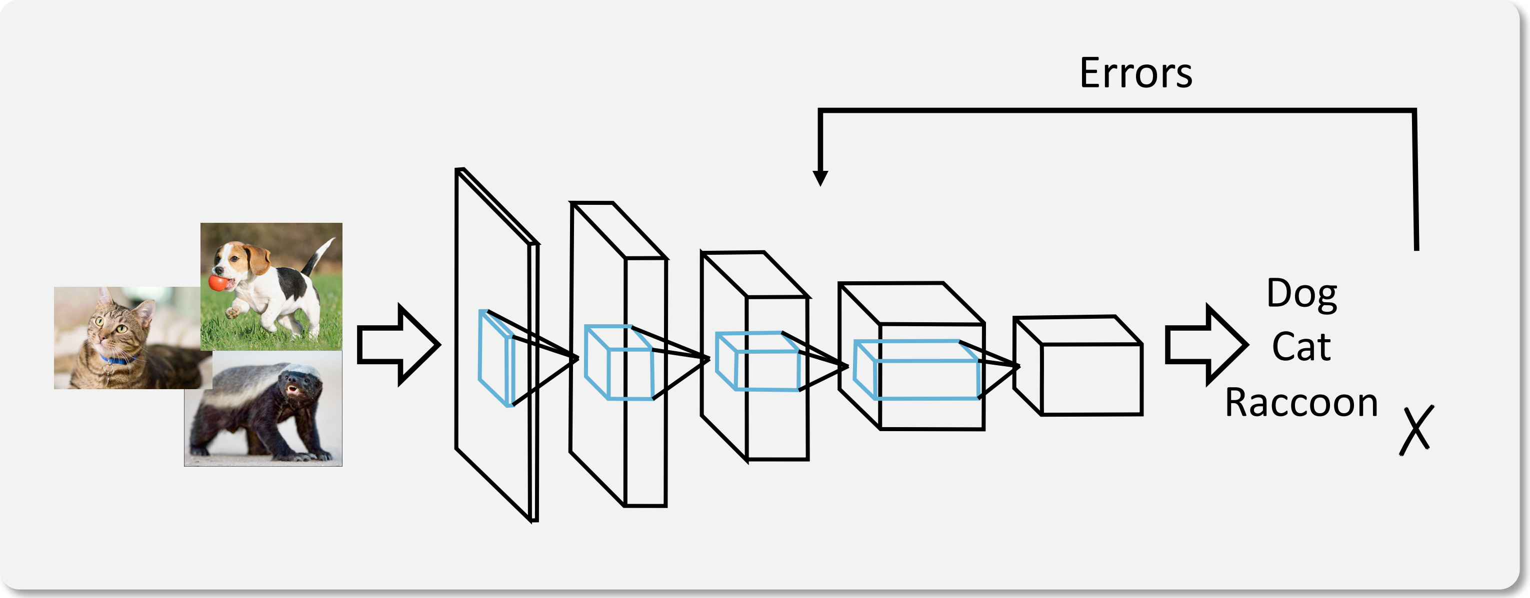
SIRI

Applications

- Intelligent assistant: Google Now, Siri, Cortana
- Face recognition
- Video content understanding

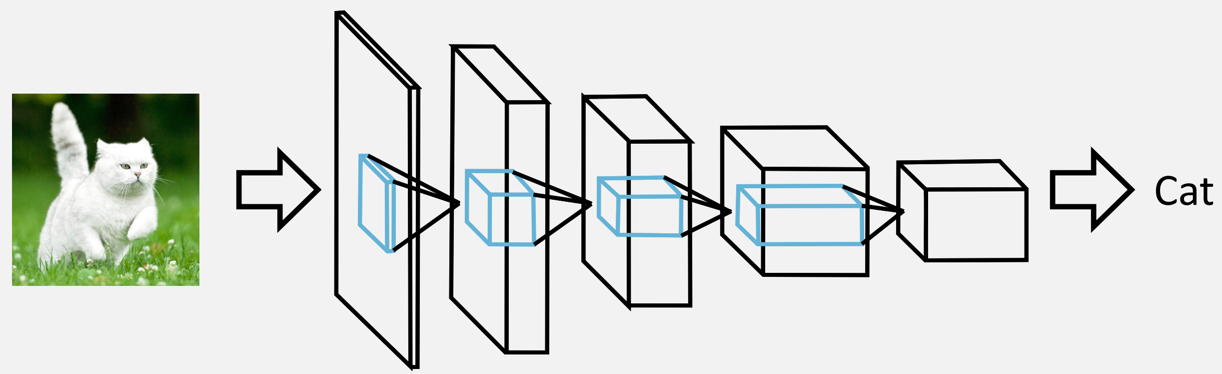
A Brief Introduction to Deep Learning

- Training:
  - Forward & backward pass
  - Iterative



A Brief Introduction to Deep Learning

- Training:
  - Forward & backward pass
  - Iterative
- Inference:
  - Forward pass



Accelerate Deep Learning with GPUs

	Neural Networks	GPUs
Inherently Parallel		
Matrix Operations		
FLOPS		

Exclusive Access to GPU

An application can have multiple GPUs, but each GPU usually belongs to exactly one application at a time.

Advantages

- Simplifies hardware design
- Efficiency

Disadvantages

- Lack of flexibility

Exclusive Access: Lack of Flexibility

- Hinders the scheduling ability of GPU cluster managers
- Underutilization
  - Hyper-parameter tuning (AutoML)
  - Model serving (inference)

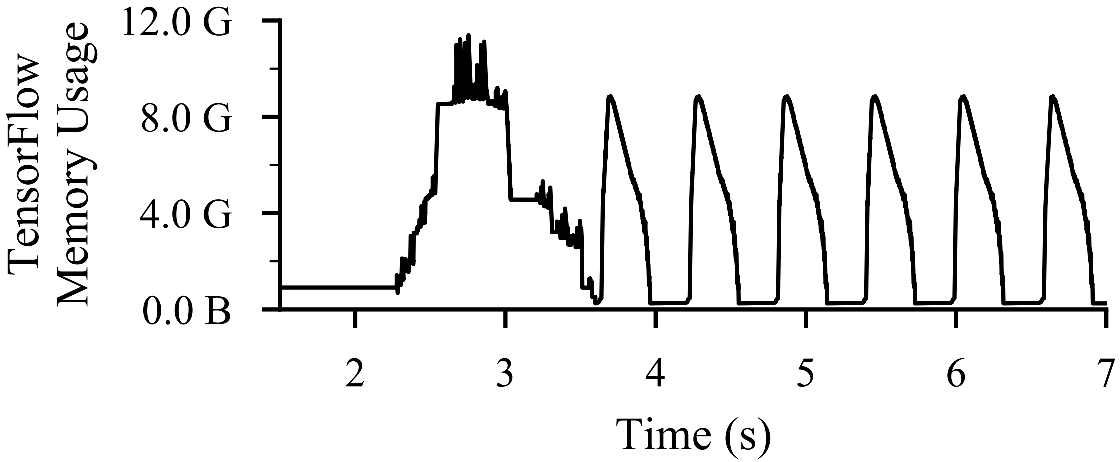
Exclusive Access: Lack of Flexibility

- Hinders the scheduling ability of GPU cluster managers
  - Starting or suspending job is expensive
  - Often easier to just do non-preemptive scheduling → FIFO
    - Head-of-line blocking

Exclusive Access: Lack of Flexibility

- Underutilization
  - Variance in memory usage → Overprovision

Model	Peak Memory Usage
VAE	28M
Super Resolution	529M
Deep Speech	3993M
Inception4	11355M



How Can We Efficiently Share a GPU for Deep Learning Applications?

GPU Sharing

- Existing sharing solutions

Approach	Efficiency	Dynamic Memory	Flexible Scheduling
Static Partitioning (SP)	No	No	Yes
Multi-Process Service (MPS)	Yes	No	No

Design Goals

Approach	Efficiency	Dynamic Memory	Flexible Scheduling
Static Partitioning (SP)	No	No	Yes
Multi-Process Service (MPS)	Yes	No	No
Ideal	Yes	Yes	Yes

Minimize deployment overhead

- No new hardware
- No modification from user side

Salus

Fine-grained GPU Sharing Primitives for Deep Learning

A consolidated execution service enabling sharing primitives

- Fast job switching,
- Memory sharing

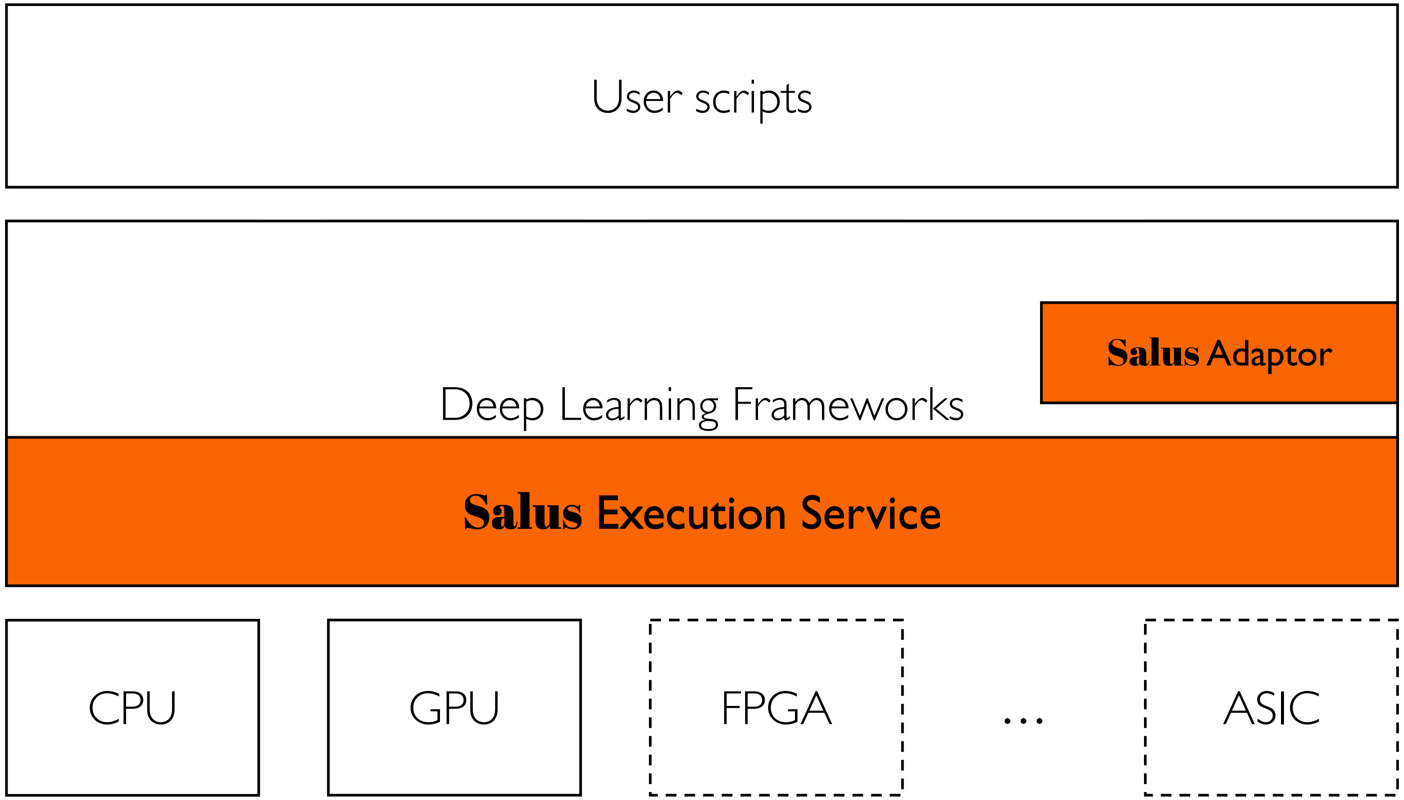
without modifying any

- User scripts,
- Operating systems, or
- Hardware

with the goal to

- Support new scheduler for GPU,
- Improve GPU utilization

Salus in DL Stack

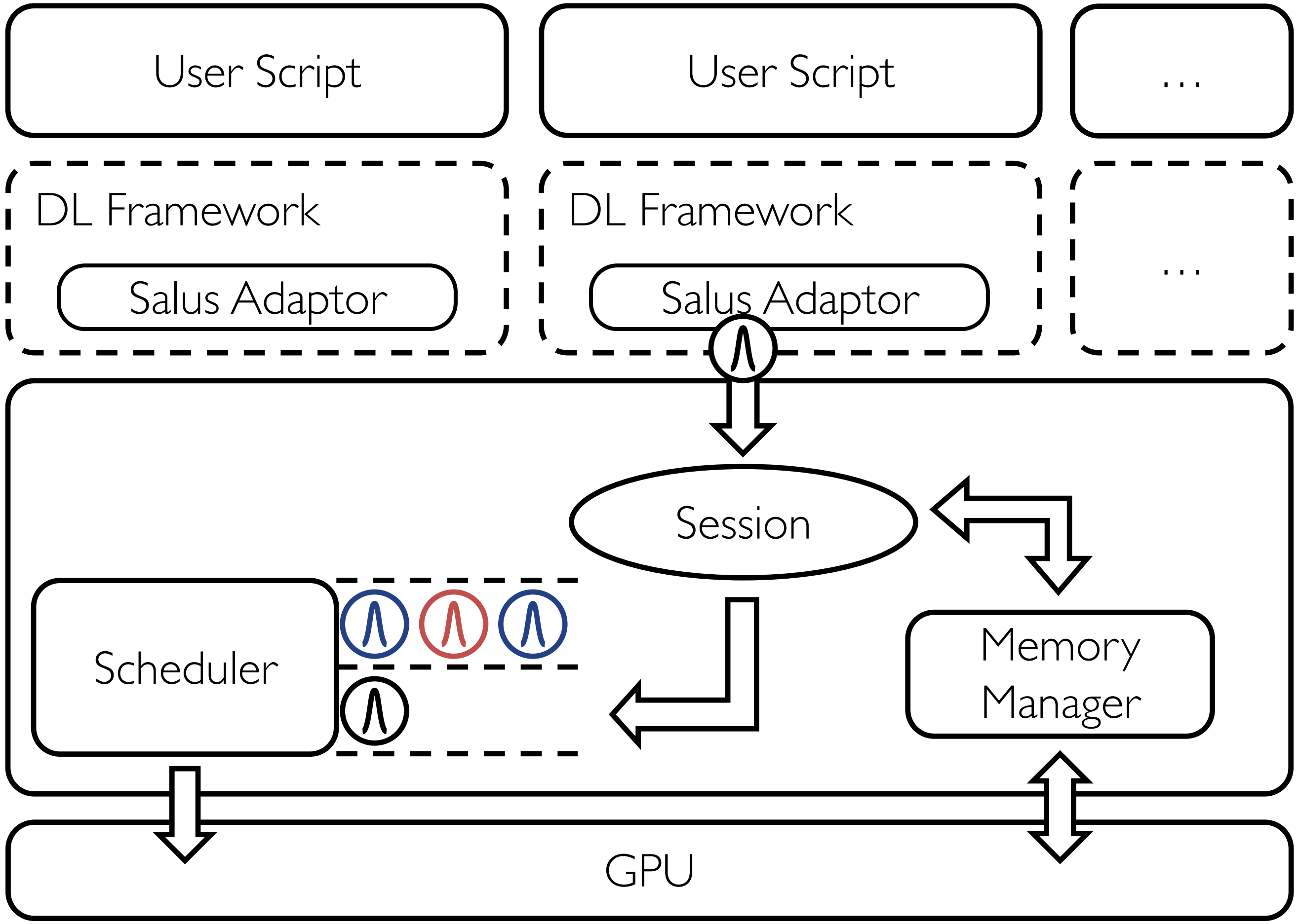


Salus *Components*

1. **Salus** Adaptor — *Transfer computation graph*
2. **Salus** Execution Service — *Consolidates all GPU accesses*

Salus in One Slide

- Create session
- Send computation graph
- For each iteration:
  - Send input
  - Check memory
  - Queue in scheduler



Sharing Primitives

- Efficient job switching
- Memory sharing: GPU lane abstraction

Sharing Primitives: Efficient Job Switching

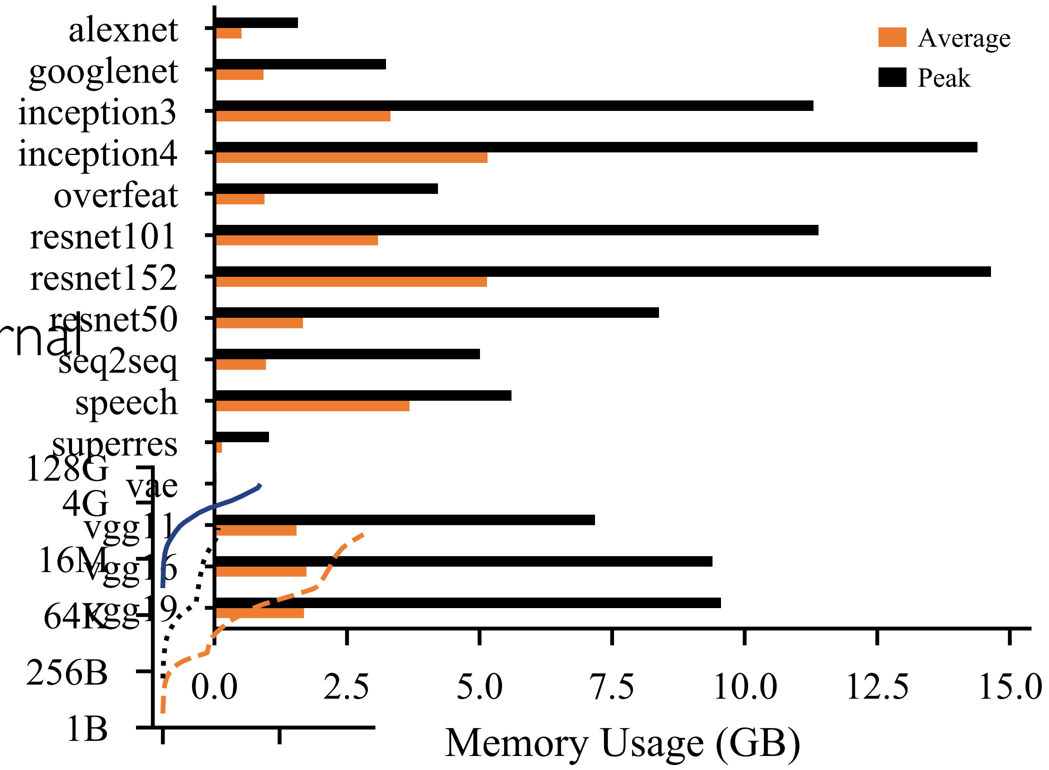
Existing Approaches	Time Scale
Stop and restart (checkpointing)	10~100s
Generate snapshot[1]	~1s

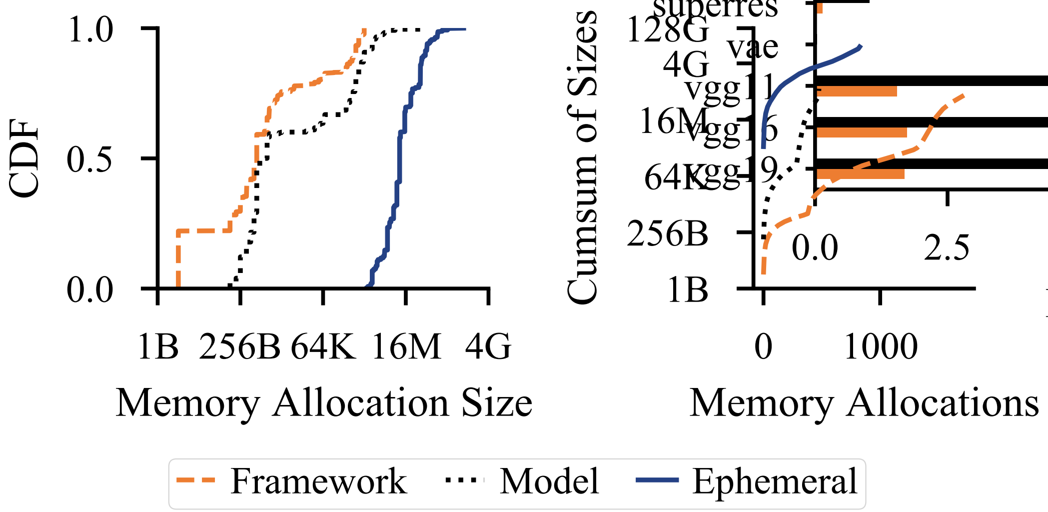
Bottleneck: data (memory) transfer

[1]: W. Xiao et al. "Gandiva: Introspective Cluster Scheduling for Deep Learning". In: OSDI. 2018.

Understand DL Job Memory

- 3 types of memory:
  - Model
  - Ephemeral
  - Framework-internal





Understand DL Job Memory

- 3 types of memory:
  - Model
  - Ephemeral
  - Framework-internal
- Data transfer time is non-negligible
  - Can be over 2X of corresponding inference latency
- Model memory << GPU memory capacity

Why not keep multiple jobs' model in memory for fast switching?

Sharing Primitives: Efficient Job Switching

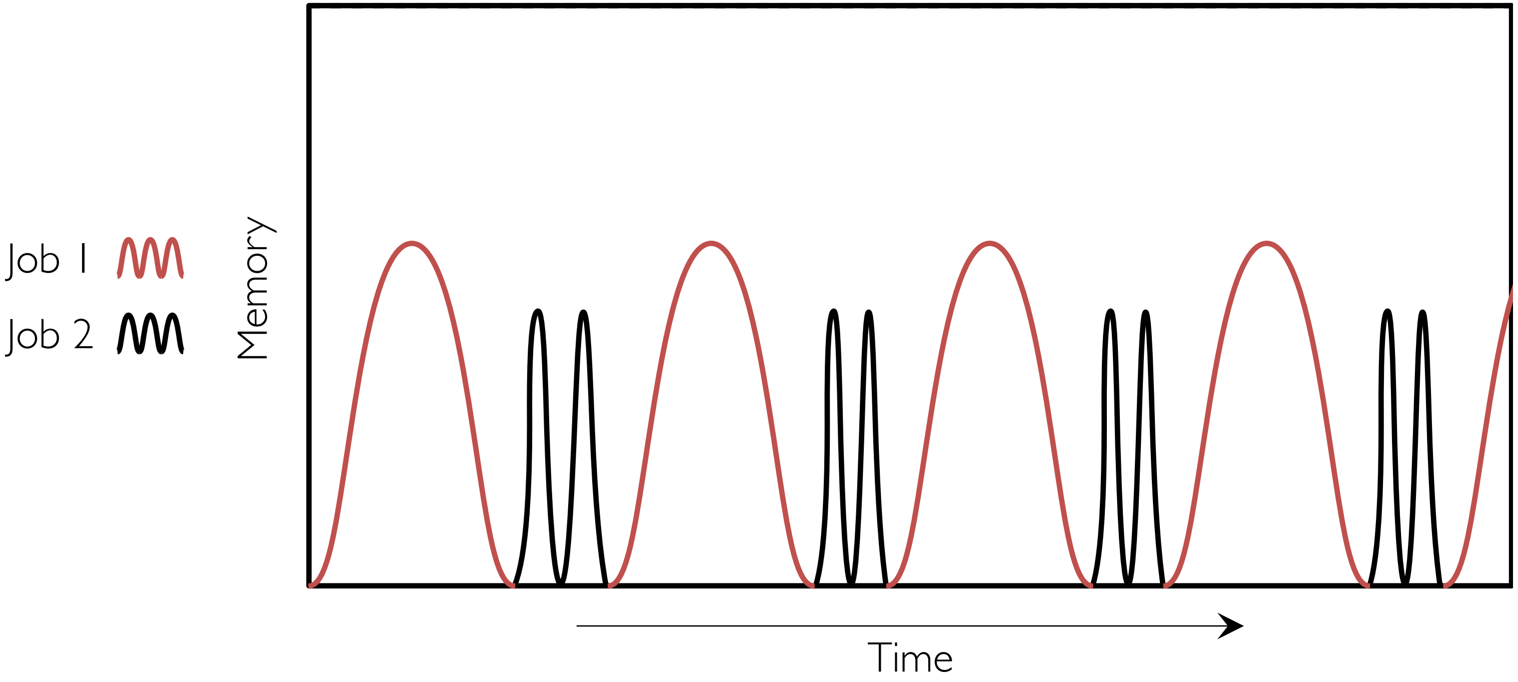
Job switching is done by determine which job's iteration to run next.

- Minimal switching overhead
- Flexible scheduling policies

A trade-off between maximum utilization and execution performance

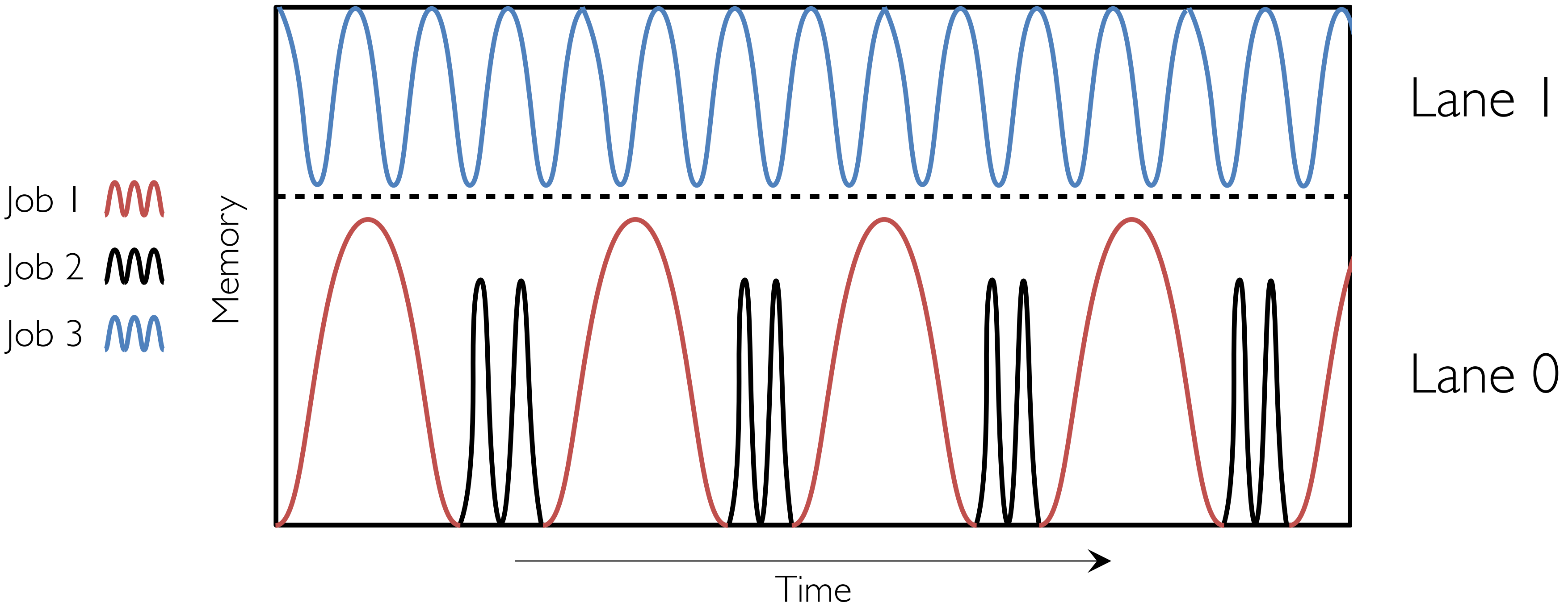
Sharing Primitives

- Efficient job switching



Sharing Primitives

- Efficient job switching
- Memory sharing: GPU lane



Sharing Primitives: Memory Sharing

- Efficient job switching
- Memory sharing: GPU lane

= Continuous physical memory + GPU stream

- *Time-slicing* within lane, *parallel* across lanes
- Dynamic re-partitioning (lane assignment)
- Avoid in-lane fragmentation

GPU Lane: Best Fit & Safety Condition

- A lane cannot accept arbitrary number of jobs
- The *Safety Condition* determines whether a job can go in a lane

$$\sum_i P_i + \max_i T_i \leq C_l$$

P_i: Model and framework-internal memory for job i
T_i: Ephemeral memory for job i
C_l: Memory capacity of lane l

GPU Lane: Best Fit & Safety Condition

- A lane cannot accept arbitrary number of jobs
- The *Safety Condition* determines whether a job can go in a lane

Static Partitioning: $$\sum_i P_i + \sum_i T_i \leq C_l$$

P_i: Model and framework-internal memory for job i
T_i: Ephemeral memory for job i
C_l: Memory capacity of lane l

Salus Scheduling Polices

FIFO is suboptimal

- HOL blocking
- Underutilization

With Salus

- Packing: achieves higher utilization
- Preemption: enables prioritization
- Fairness: equalizes the resource usage
- …
- What's more? Still a huge design space!

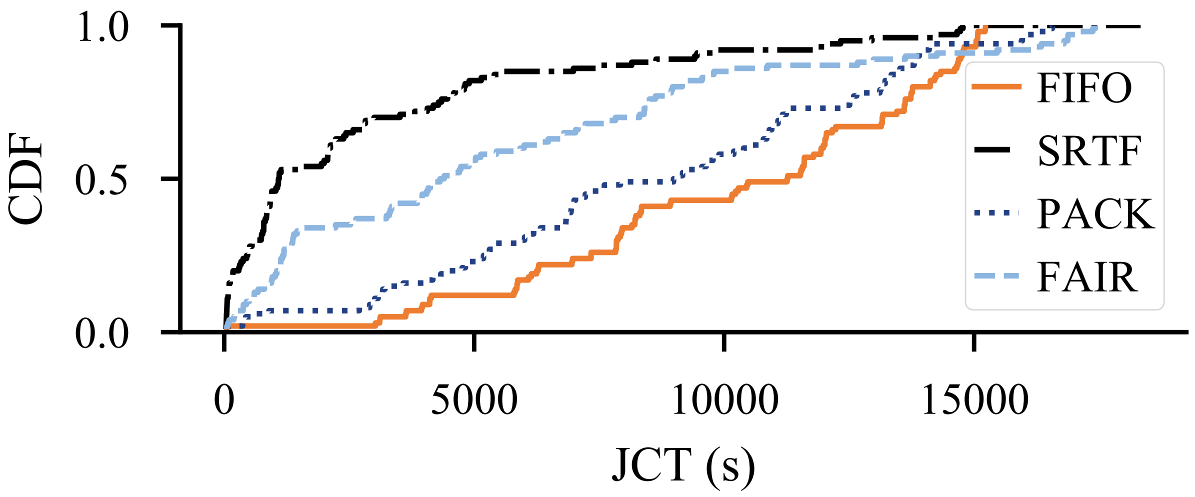
Evaluation

Deployment and evaluation on Intel E5-2670 with 2x NVIDIA Tesla P100 with 15 workloads

1. Flexible scheduler
2. Faster hyper-parameter tuning
3. High GPU utilization for inference

A Production Trace

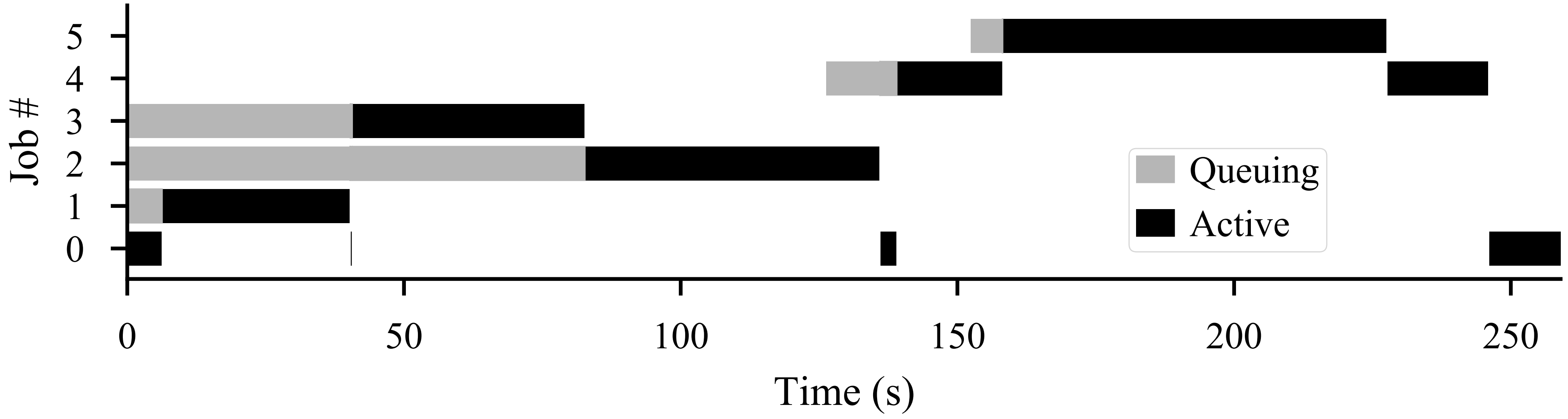
- 100 jobs from a production trace[1]
- 4 schedulers implemented as demo
- SRTF vs FIFO: 3.19x improvement in Avg. JCT



[1]: G. Juncheng et al. "Tiresias: A GPU Cluster Manager for Distributed Deep Learning". In: NSDI. 2019.

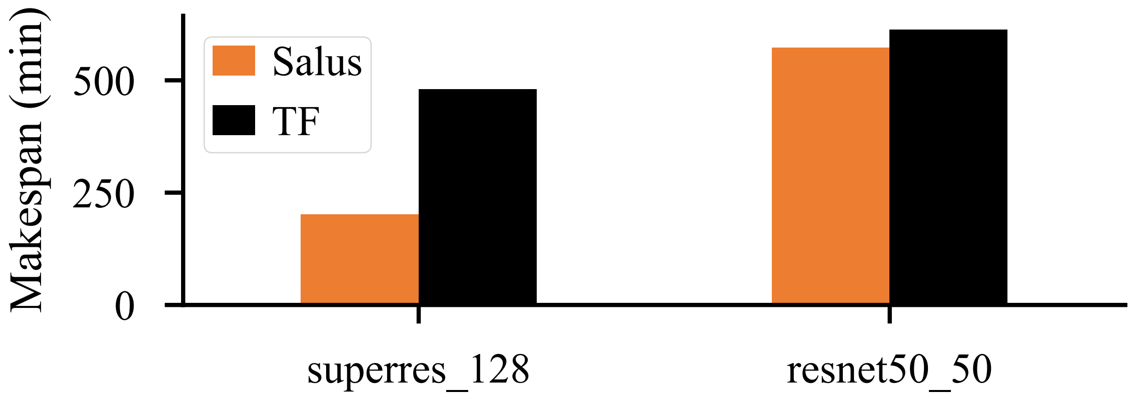
Sub-second Level Switching

- Slice of the 100 job trace, time is normalized
- Sub-second switching



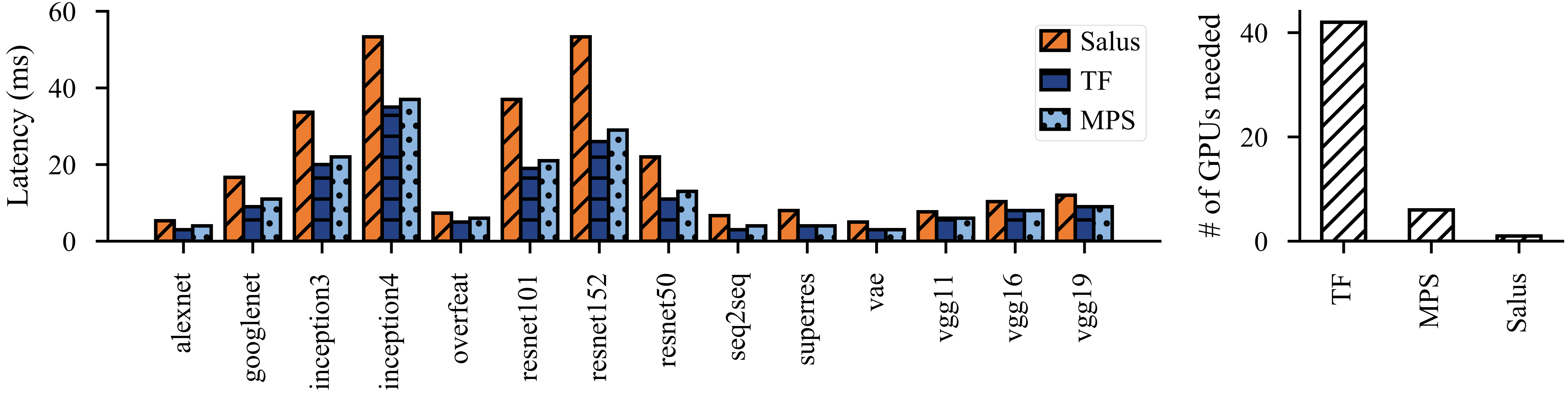
Hyper-parameter Exploration

- 2 sets of hyper-parameter exploration
- 300 exploration jobs in each set
- Makespan is important



Pack Inference Applications

- 42 DL inference applications in **1** GPU
- User facing services: latency



Salus

Fine-grained GPU Sharing Primitives for Deep Learning

Open sourced at: https://github.com/SymbioticLab/Salus

- Prebuilt docker image available