OLF 2023

FlexFlow: unveiling the optimal parallel strategy for your LLMs automatically

Liang Yan
09/09/2023

Self-Introduction

Liang Yan

Sr. Software Engineer II, DigitalOcean, Louisville KY

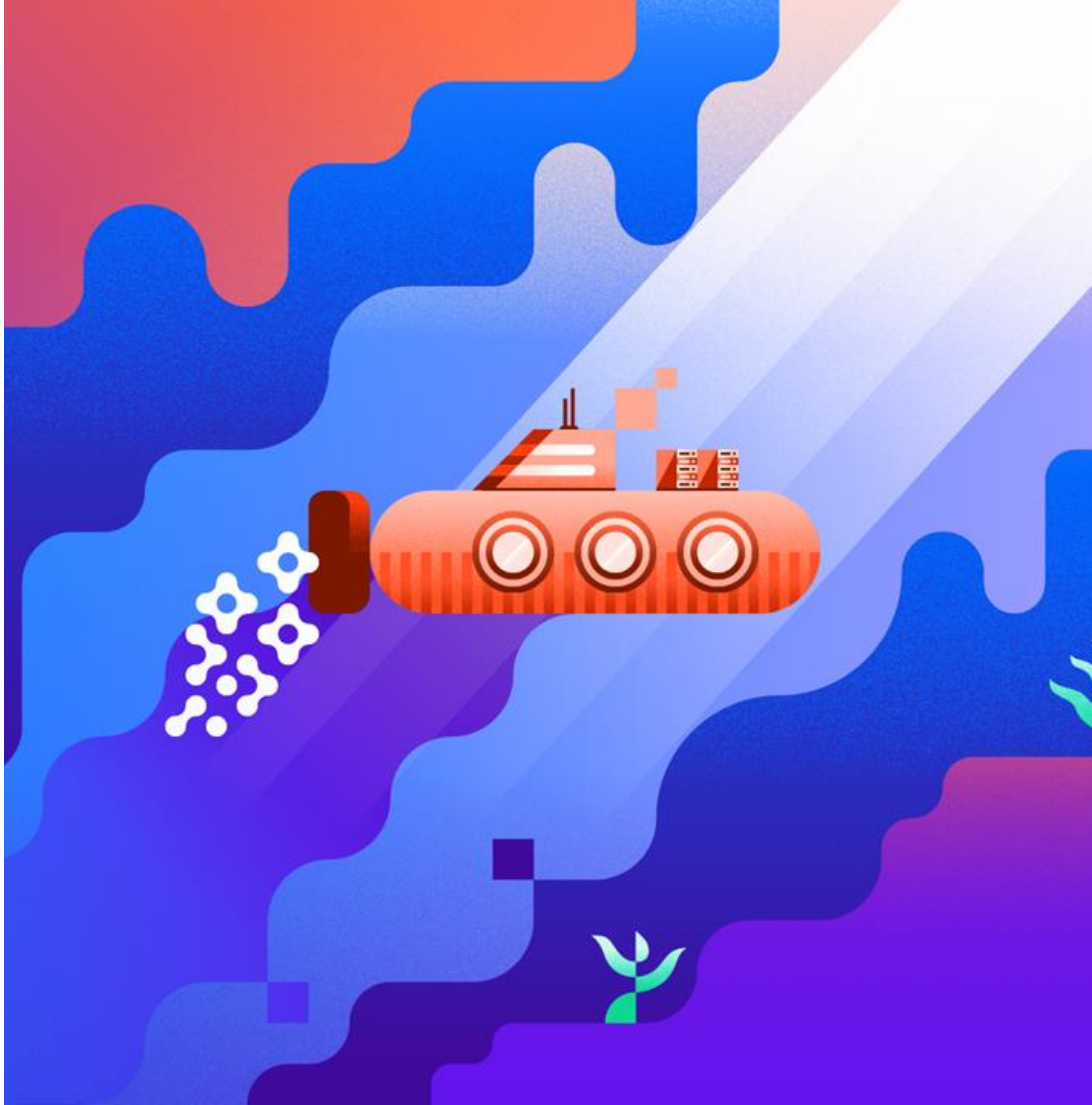
- Over ten years industry experience on Virtualization, mostly on GPU, Network and Live Migration.
- Current focus on Distributed Machine Learning Infra for Large Model
- Opensource and Arm64 Board Enthusiast

---

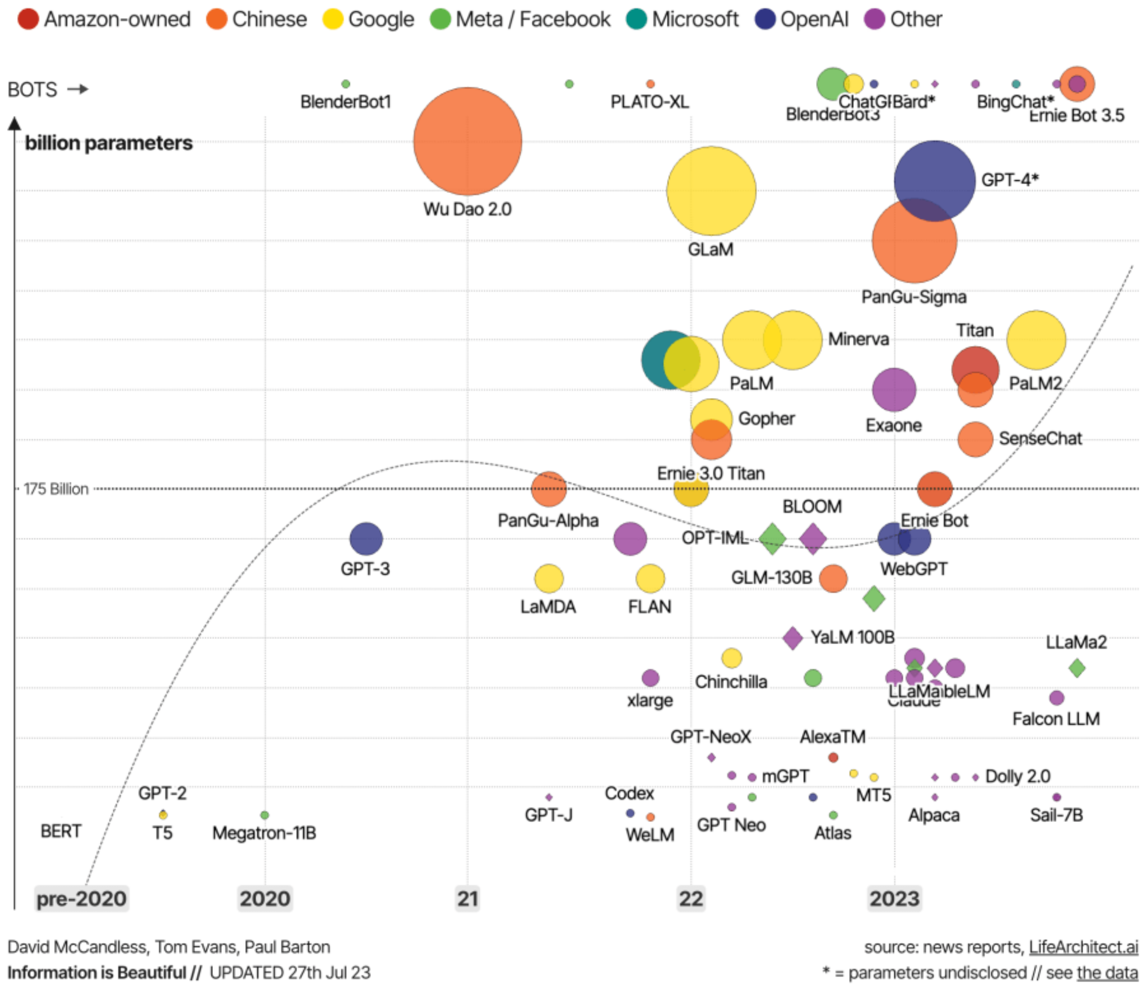
https://www.linkedin.com/in/lyantech

Agenda

- Where the story begins
- Distributed ML
- FlexFlow and More
- A practice to AlexNet in FlexFlow
- Conclusion

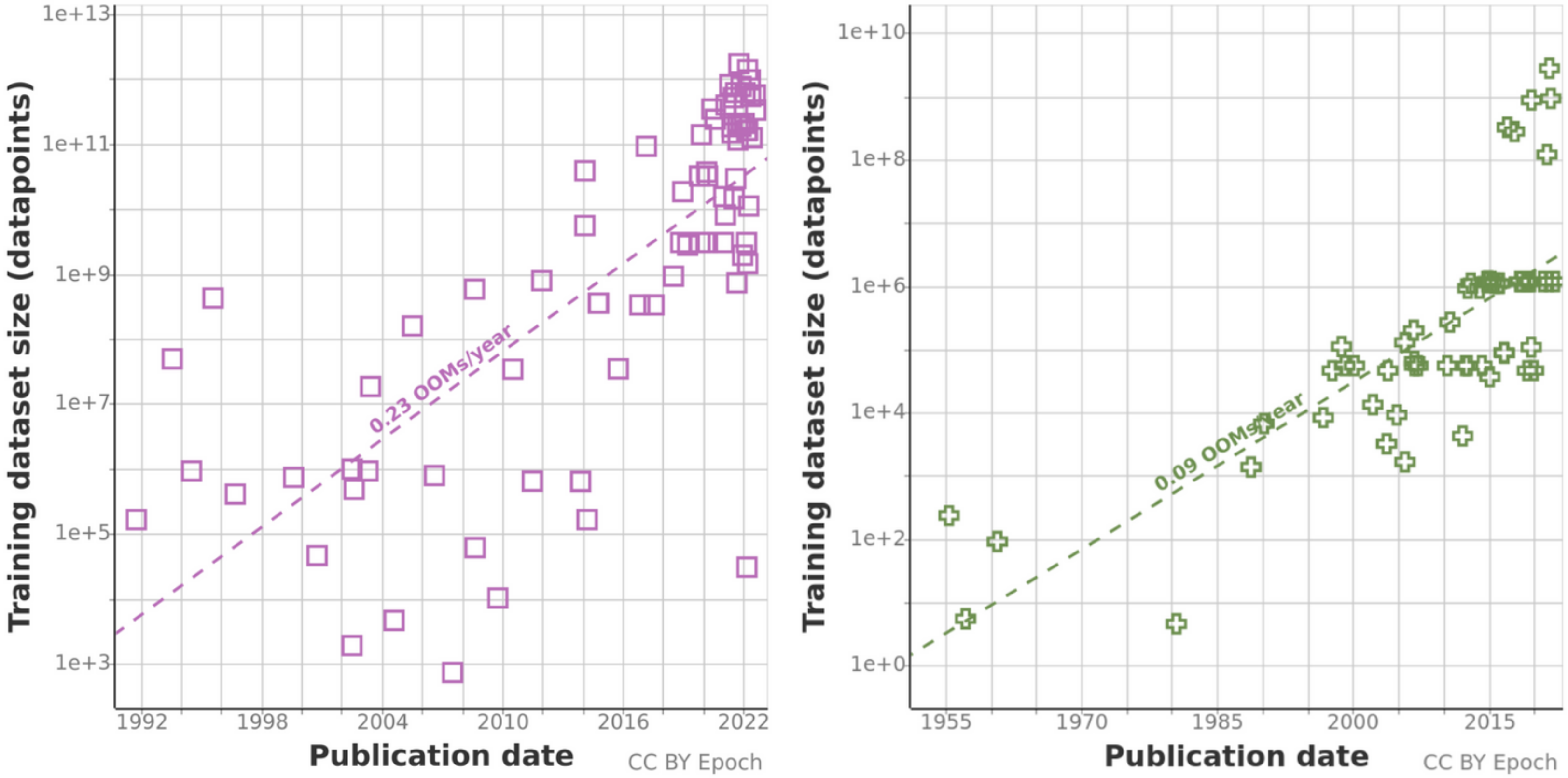


Large Models



https://informationisbeautiful.net/

DataSet Size



Training datasets for language (left) and vision (right).

https://epochai.org/blog/trends-in-training-dataset-sizes

Distributed machine learning

- Data Parallelism: different node different data same model

  Horovod、Tensorflow Estimator、PyTorch DDP

- Model Parallelism: different node different part of mode same data

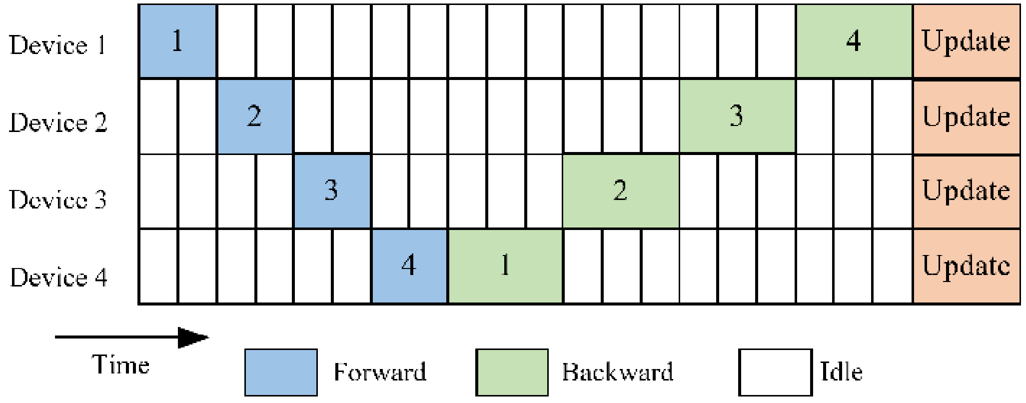
- PipeLine Parallelism:
  Gpipe、PipeDream、PipeMare

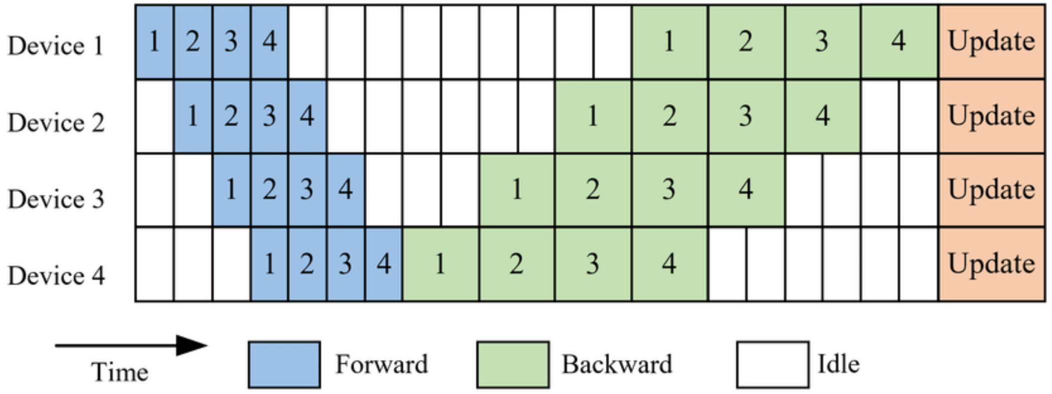
- Operator Parallelism:
  Mesh Tensorflow、FlexFlow、OneFlow、MindSpore

---

https://www.researchgate.net/publication/362249737_Dive_into_Big_Model_Training

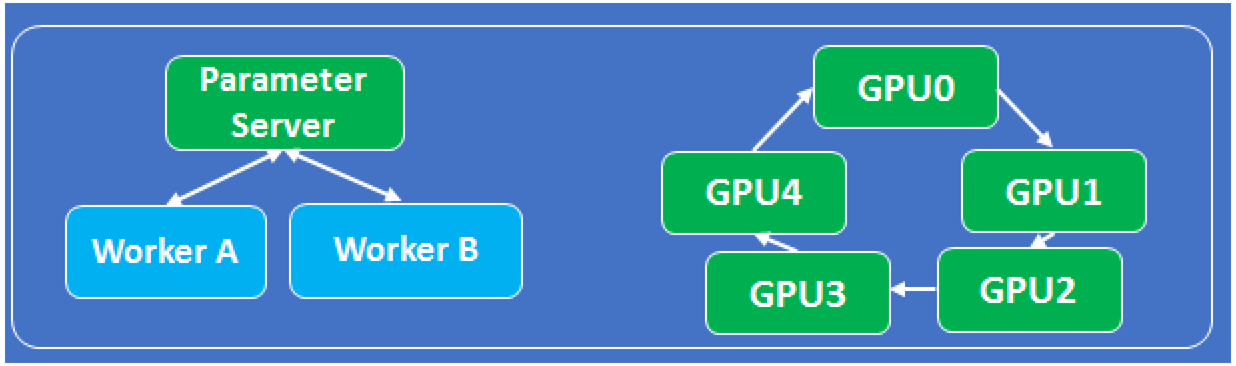
Distributed machine learning





https://www.researchgate.net/publication/362249737_Dive_into_Big_Model_Training

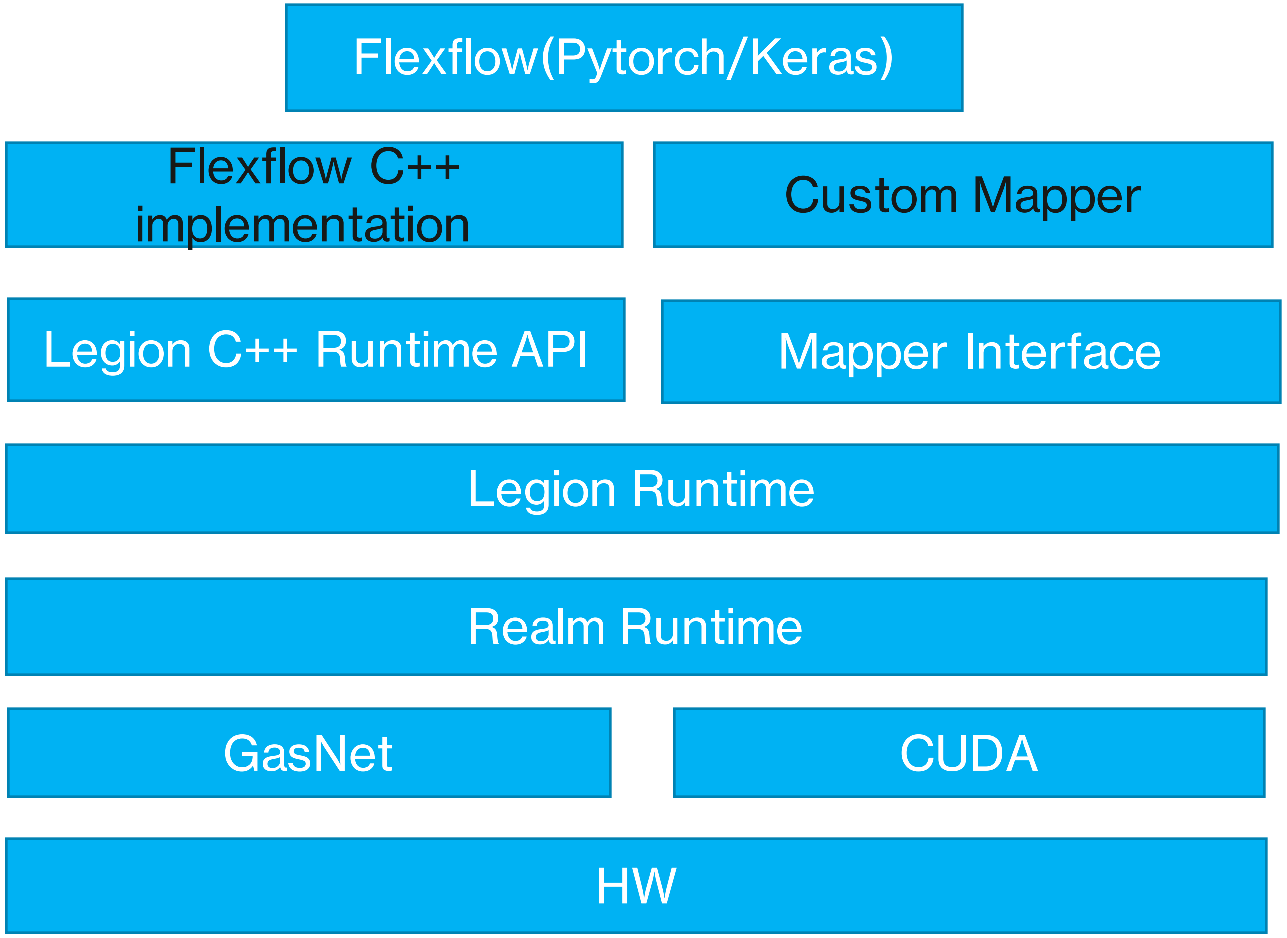
Distributed machine learning



In general, parameter server works better if you have a large number of unreliable and not so powerful machine. Ring-AllReduce works better if you have a small amount of fast devices(variance of step time between each device is small) run in a controlled environment with strong connected links.

PS: performance bottleneck from communication
Ring - AllReduce: linear relationship with GPUs

FlexFlow Architecture



FlexFlow

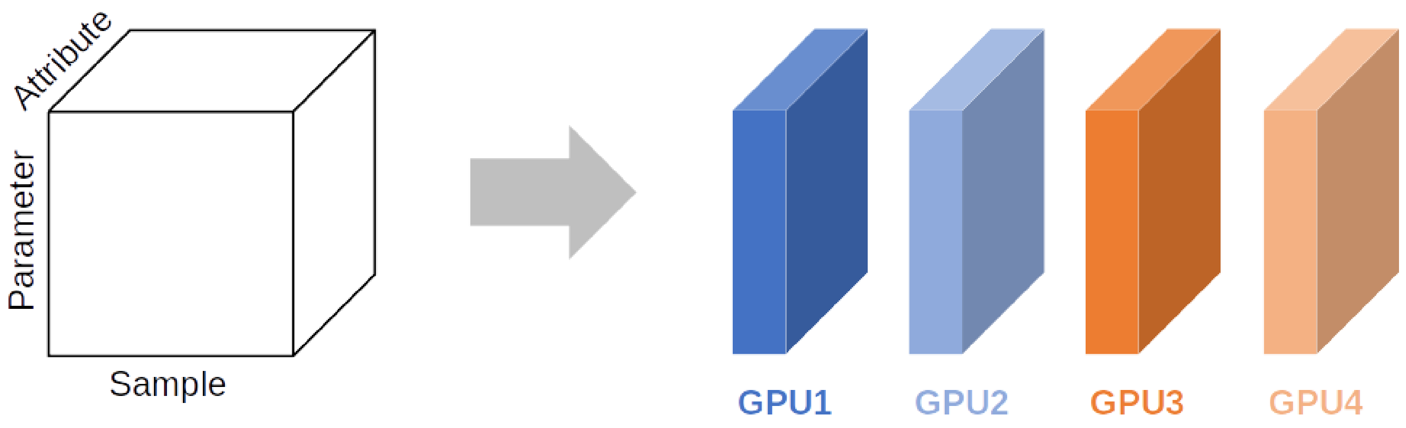
Define a search space of possible parallelization strategies

A cost model and a search algorithm

Optimized Parallelization strategies

SOAP

Samples: partitioning training samples (Data Parallelism)
Operators
Attributes
Parameters

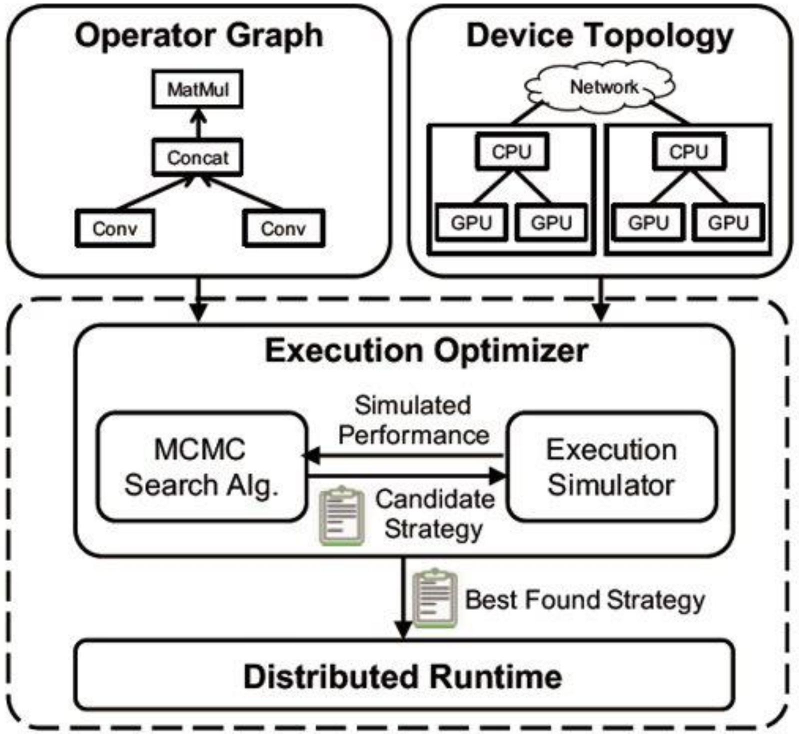


FlexFlow Simulator

•Input

1. Operator graph G: node as operator and edge as tensor
2. Device Topology D: node as device and edge as connection like NVLink, PCI-e, IB, RDMA...

Execution Optimizer:
Found the efficient strategy from SOAP space by MCMC, then feed into simulator to find the best strategy.



Operator Graph
DotFile<SimTask *> taskGraph;

Device Topology:
MachineModel

std::map<Op*, ParallelConfig> strategies;

MCMC => FFModel::optimize

Start from a random strategy(data parallelism by default)

For iter :=1 to budget:

    Generate a new strategy S* from s by updating one layer

    If cost(S*) < cost(S):

        Replace S with S*

    Else:

        Replace S with S* with probability exp(a *(cost(S) - cost(S*)))

Return the best discovered S

get_random_parallel_config

```
for (int i = 1; i <= ff.config.workersPerNode; i++)
    if (channel % i == 0)
      for (int j = 1; i * j <= total_devices; j++)
        if (batch % j == 0) {
          batch_candidates.push_back(j);
          channel_candidates.push_back(i);
```

when run the func get_random_parallel_config of current ops(Linear), the product of the new rewrite ParallelStrategyConfig is allowed to be less than the number of GPUs (i*j <= total_devices).

FlexFlow allows using a subset of GPUs if that's beneficial (e.g., communication cost outweighs performance gains). Image data and model parallelism as two dimensions FlexFlow considers for parallelizing Linear operators. In this case, the product of the degree of data and model parallelism is the overall number of GPUs used for training a Linear op.

Simulate_Cost

Step 1: register forward and backward tasks

model layer => op(partition ndims) => task_manager

Step 2: insert dependencies and comm. tasks before compute tasks

add_task_dependencies_with_xfer, iterate inputOp,

Step 2.5: add finals tasks for each compute device to capture the returning comm tasks from parameter servers

new_barrier_task

Step 3a: consider backpropagation and weight update are overlapped

add a compute task for parameter update

Step 3b: Bulk Synchronous Model

add a per-device barrier before weight update

Step 4: add ready tasks into ready_queue

std::priority_queue<SimTask*, std::vector<SimTask*>, SimTaskCompare> ready_queue;

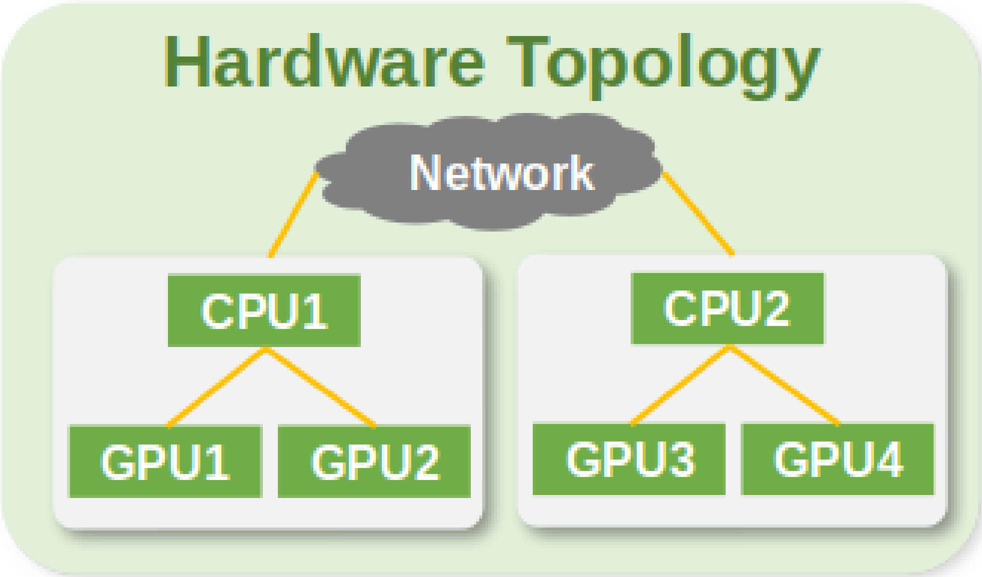
Step 5: perform simulation *

Step 5.5: update nccl_time

Step 6: add penalty to strategies that exceed the memory limits on devices

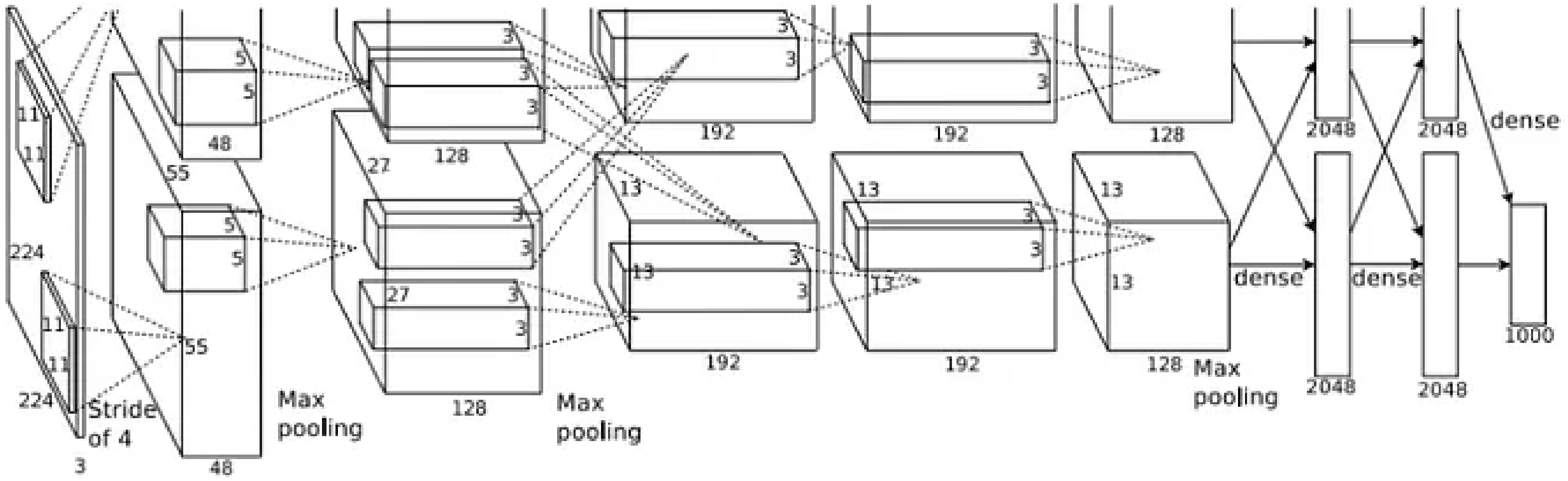
Penalize the total runtime by 1ms if we exceed the memory budget by 1MB

What we do



A customized Hardware Topology which is closer to the cluster in datacenter today!

Practice in ~~Transformer~~ AlexNet

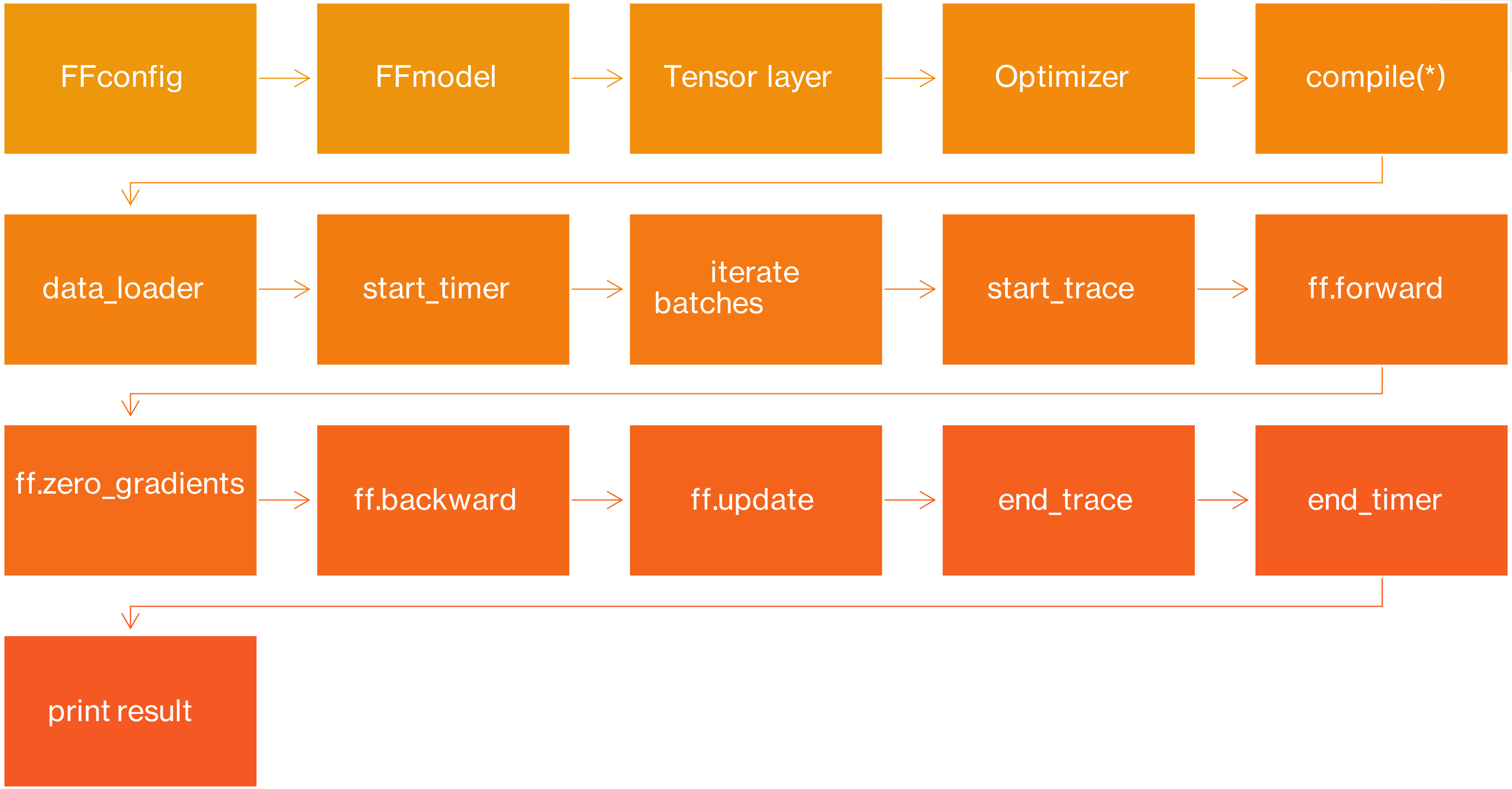


https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

AlexNet Operator & Tensor Layer

Ops	channel	kernelH	KernelW	strideH	strideW	paddingH	paddingW	Activate
input	3	299	299					
conv2d	64	11	11	4	4	2	2	relu
pool2d		3	3	2	2	0	0	
conv2d	192	5	5	1	1	2	2	relu
pool2d		3	3	2	2	0	0	
conv2d	384	3	3	1	1	1	1	relu
conv2d	256	3	3	1	1	1	1	relu
conv2d	256	3	3	1	1	1	1	relu
pool2d		3	3	2	2	0	0	
flat								
dense	4096							relu
dense	4096							relu
dense	10							
softmax								

AlexNet Process in FlexFlow

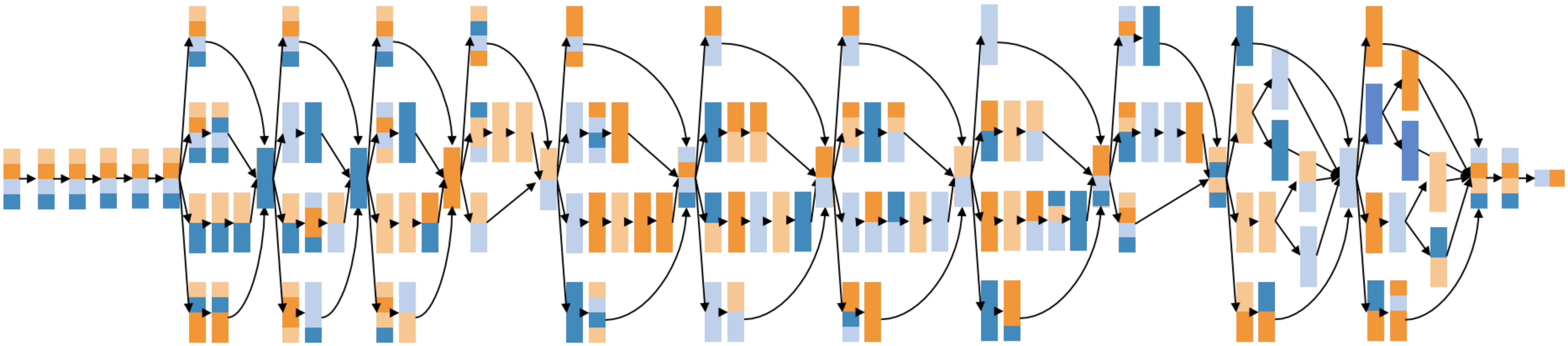


Strategy => FFModel::optimize

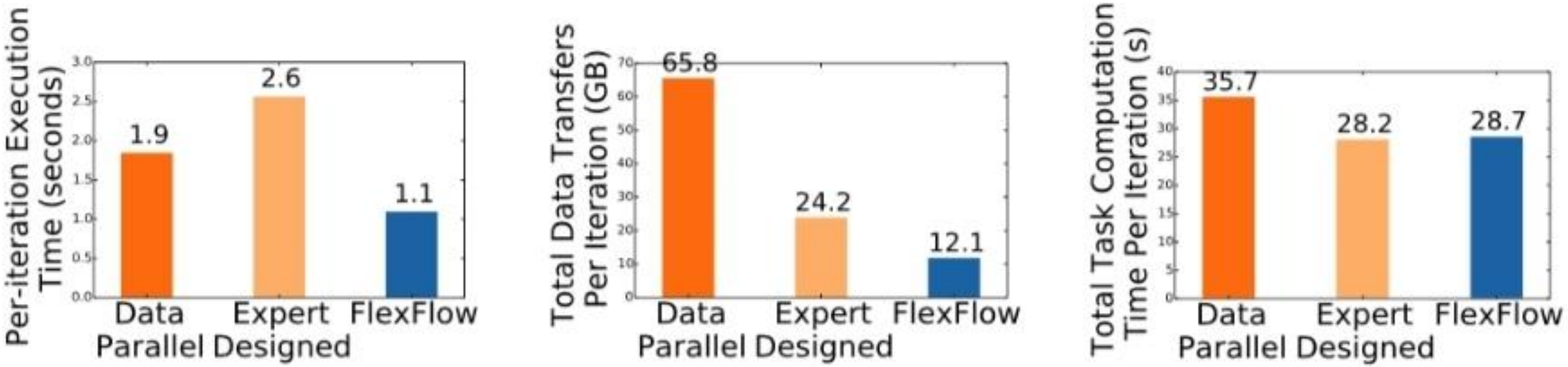
=========== Best Discovered Strategy ==========
[Conv2D_100] num_dims(4) dims[1,1,1,2] device_ids[0,1]
[Conv2D_101] num_dims(4) dims[1,1,1,2] device_ids[0,1]
[Pool2D_102] num_dims(4) dims[1,1,1,2] device_ids[0,1]
[Flat_103] num_dims(2) dims[1,2] device_ids[0,1]
[Dense_104] num_dims(2) dims[1,1] device_ids[1]
[Dense_105] num_dims(2) dims[1,2] device_ids[0,1]
[Softmax_106] num_dims(2) dims[1,2] device_ids[0,1]

Conv2D_100 -> name of the operator
0 -> the device that the operator can be executed on, 0 means GPU, 1 means CPU
4 -> the dimensions of the input tensor of the operator
1 1 1 2 -> how each parallelizable dimension is parallelized, 1 means no parallelization, n means this dimension is divided into n pieces for parallelization
2 -> number of devices the operator is executed on
0 1 -> device id the operator is executed on

Final look



Evaluation

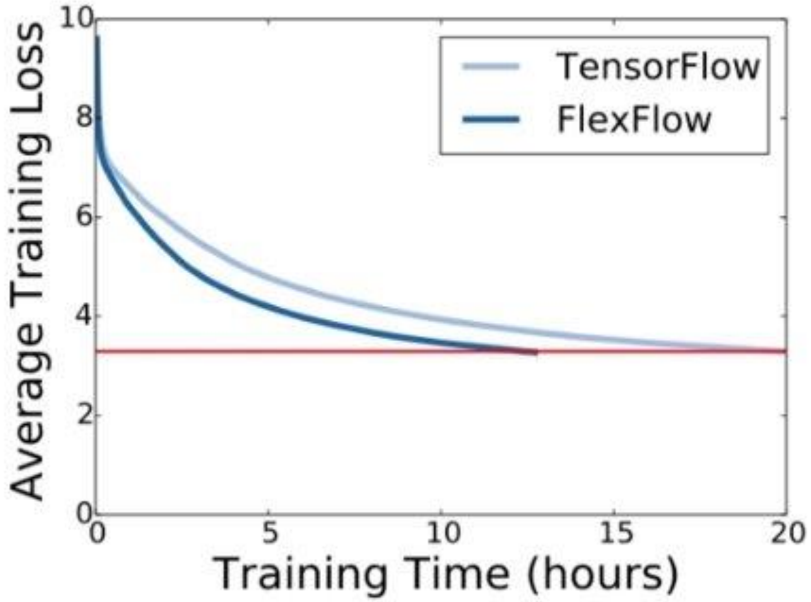


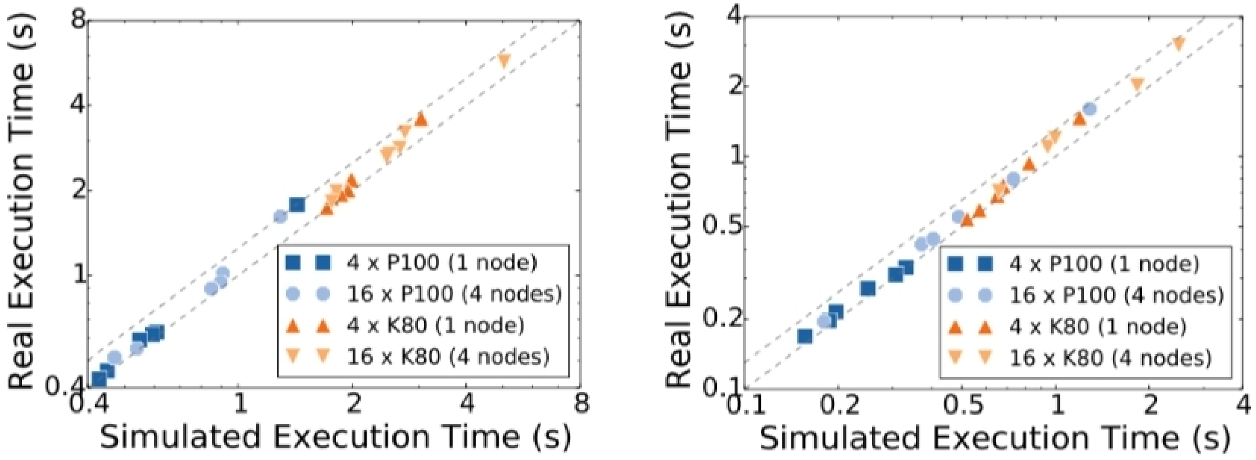
(a) Per-iteration execution time.

(b) Overall data transfers per iteration.

(c) Overall task run time per iteration.

Evaluation





(a) Inception-v3

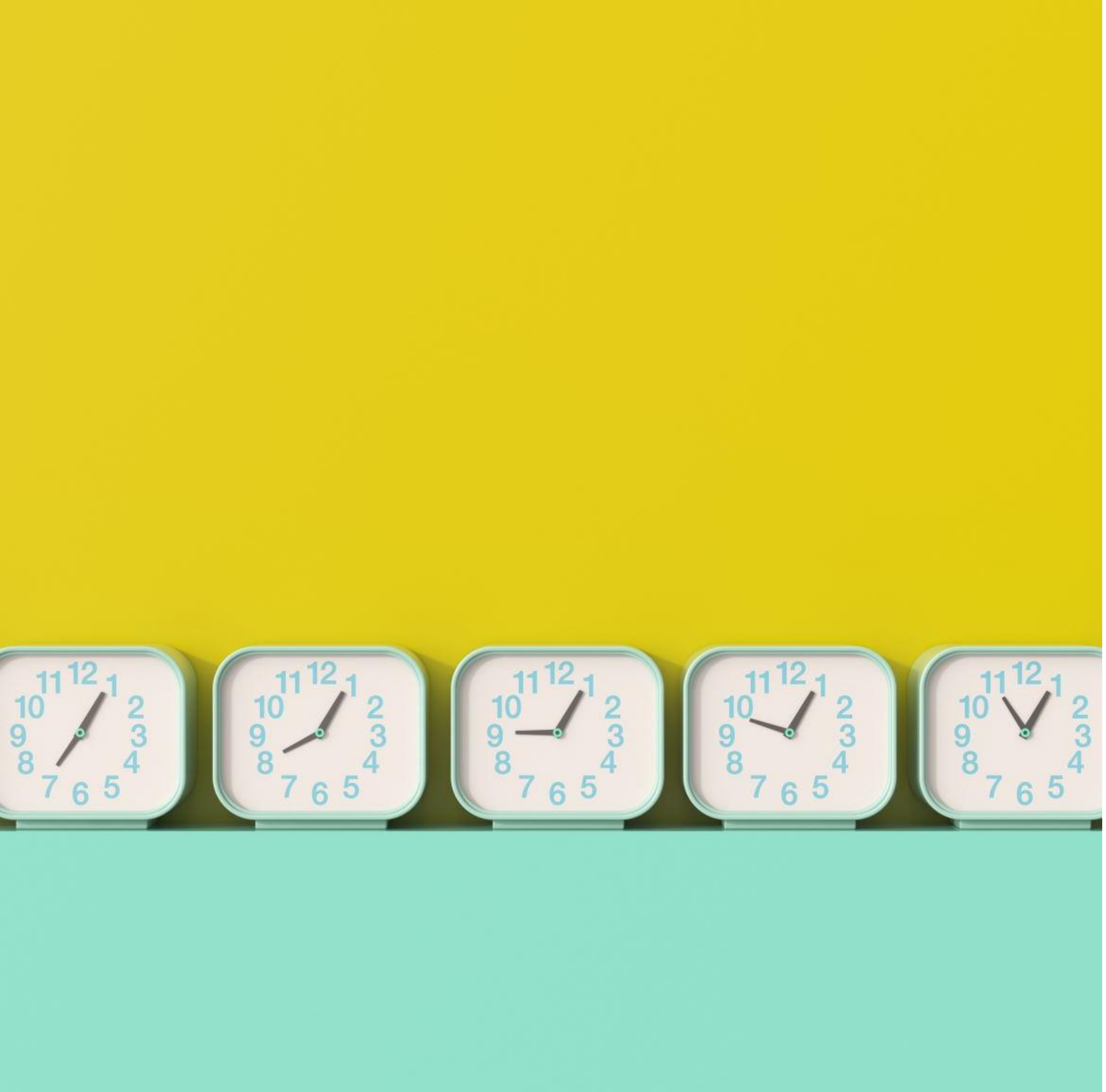
(b) NMT

Conclusion

1. We focus on the simulator and search algorithm
2. We are extending the machine topology to heterogeneous GPU capability

Reference

1. https://flexflow.ai/
2. https://informationisbeautiful.net/
3. https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf
4. https://arxiv.org/abs/1807.05358
5. https://epochai.org/blog/trends-in-training-dataset-sizes



Q & A

Thanks!