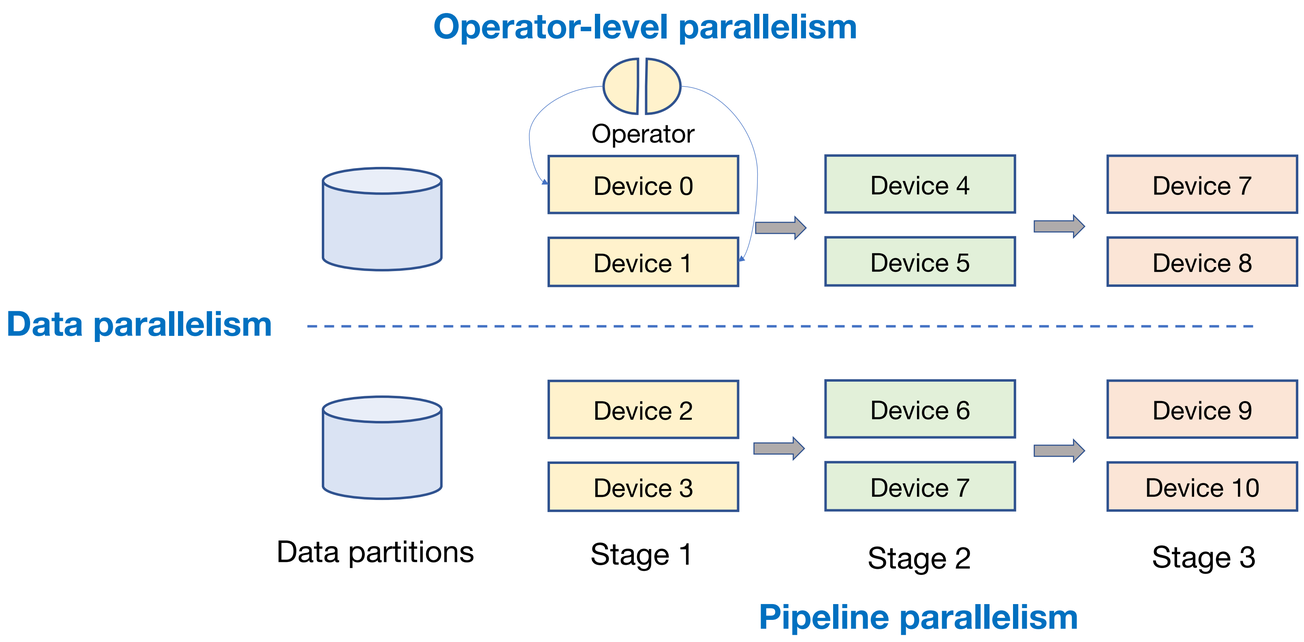
Efficient Training of Giant Neural Networks

Zeyuan Tan, Man-Kit Sit, Luo Mai
School of Informatics, University of Edinburgh

Many strategies are available to parallelise the training of giant neural networks



Giant neural networks: Trillions parameters (e.g., GPT-3, SimCLR)

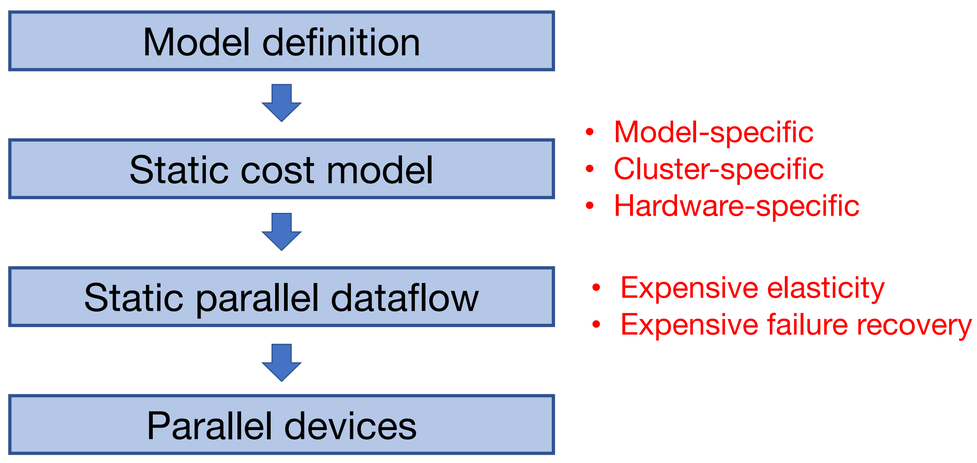
Parallel training: 100s parallel CPU/GPU devices must be coordinated to overcome memory limits

Research question: How to find best parallelism strategies that can minimise model training time?

State-of-the-art parallel training systems rely on **manual** & **static configuration**

```
context.set_auto_parallel_context(device_num=4)
class FeedForward(nn.Cell):
    def __init__(self, in, h, out):
        self.matmul1 = ops.MatMul().shard(((2, 1), (1, 2)))
        self.gelu = ops.GeLU().shard((2, 2))
        self.matmul2 = ops.MatMul().shard(((2, 2), (2, 1)))
        self.dropout = ops.Dropout().shard((2, 1))
        self.w_i = Parameters([in, h])
        self.w_o = Parameters([h, out])
    def construct(self, x):
        y = self.matmul1(x, self.w_i)
        gelu_out = self.gelu(y)
        mm_out = self.matmul2(gelu_out, self.w_o)
        z = self.dropout(mm_out)
        return z
```

Manual configuration
(GSPMD [1], MindSpore [2])



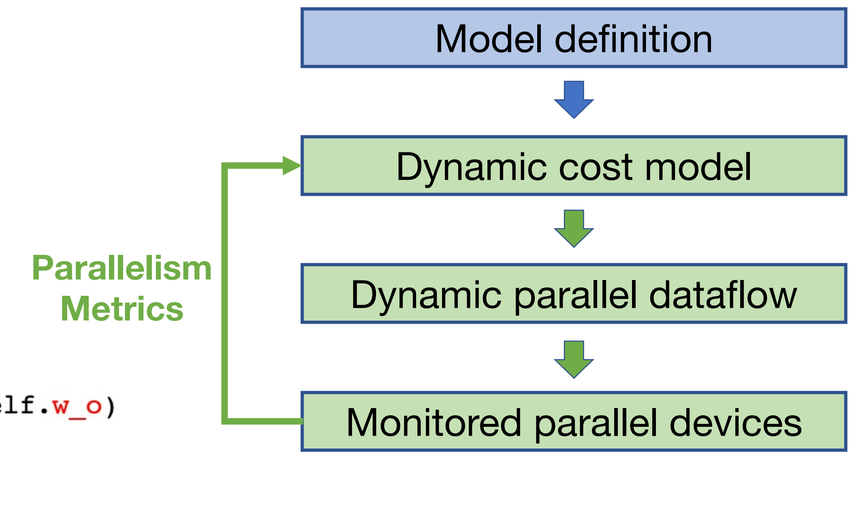
Static parallelism
(DeepSpeed [3], Auto-Parallel [4])

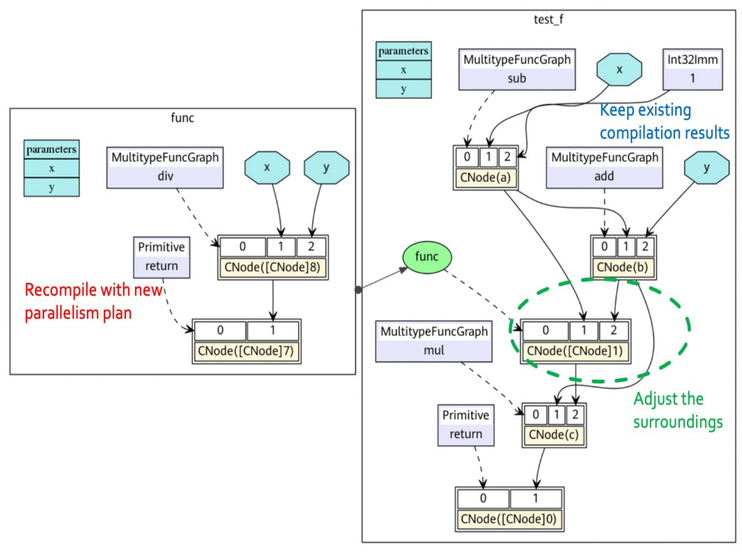
Issues:

- Manual configuration requires expert knowledge about parallel training, which is not available on ML developers
- Static parallelism configuration suffer from inaccurate static cost model & model diversity

Designing an **efficient dynamic parallelism system** for giant neural networks

```
@auto_parallel
class MyModel(nn.Cell):
    def __init__(self, in, h, out):
      self.matmul1 = ops.MatMul()
      self.gelu = ops.GeLU()
      self.matmul2 = ops.MatMul()
      self.dropout = ops.Dropout()
      self.w_i = Parameters([in, h])
      self.w_o = Parameters([h, out])
    def construct(self, x):
      y = self.matmul1(x, self.w_i)
      gelu_out = self.gelu(y)
      mm_out = self.matmul2(gelu_out, self.w_o)
      z = self.dropout(mm_out)
    return z
```





#1: Programming giant DNNs as **on a single device**

#2: Augmenting cost models with **runtime parallelism metrics**

#3: **Dynamic compilation** for parallelism plans

[1] GSPMD: General and Scalable Parallelization for ML Computation Graphs, 2021
[2] PANGU-α: Large-Scale Autoregressive Pretrained Chinese Language Models with Auto-parallel Computation, 2021
[3] ZeRO: Memory Optimizations Toward Training Trillion Parameter Models, 2020
[4] TensorOpt: Exploring the Tradeoffs in Distributed DNN Training with Auto-Parallelism, 2021