



PaddlePaddle

Towards a Deep Learning Compiler





成为软件技术专家 全球软件开发大会的必经之路

[北京站] 2018

2018年4月20-22日 北京·国际会议中心

十二购票中,每张立减2040元

团购享受更多优惠



识别二维码了解更多





下载极客时间App 获取有声IT新闻、技术产品专栏,每日更新



扫一扫下载极客时间App



ACON

全球人工智能与机器学习技术大会

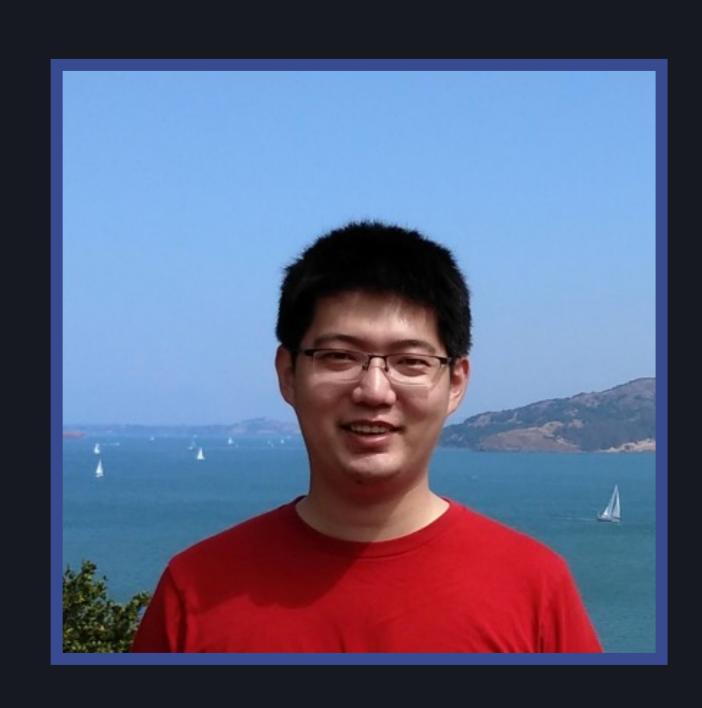
助力人工智能落地

2018.1.13 - 1.14 北京国际会议中心



扫描关注大会官网

SPEAKER INTRODUCE



于洋

百度资深工程师

百度资深工程师,从事百度深度学习平台PaddlePaddle开发工作。2015年天津大学硕士毕业。之后加入百度后一直从事深度学习系统的研发,主要负责深度学习系统的性能优化和功能开发工作。是PaddlePaddle的主要开发人员之一,参与开发了PaddlePaddle的Fluid模块。

PaddlePaddle历史

开源前

- 百度内部项目
- 四年前由徐伟老师发起
- 50+百度内部产品使用
- 获得两次百度百万美元最高奖

开源后

- 2016年9月开源
- 开源后TechLeader ---- 王益
- 新的PythonAPI
- 支持浏览器编程,云端运行
- · 深度整合Kubernetes
- 发布PaddlePaddle Fluid



深度学习框架历史

基于层的深度学习框架

用Sequence Of Layer表示神经网络



基于Op的深度学习框 架

用DAG表示神经网络



命令式编程的深度 学习框架

使用编程语句表示神经网络

四年前

两年前

目前



深度学习框架历史

- 四年时间,发展出三代深度学习系统
- 深度学习系统表达能力越来越强
 - Sequence of Layer
 - 适应CNN
 - Symbolic Programming paradigm
 - DAG
 - 可以适应RNN,并减少框架核心的代码量
 - Symbolic Programming paradigm
 - Imperative programming
 - 不区分神经网络的配置和执行。
 - 神经网络边配置边执行。



Symbolic 神经网络框架的问题

- 缺乏像编程语言般的灵活性
- 举例:
 - RNN会按照展开成多份静态网络
 - · 每份静态的网络只能处理同样形状(Shape)的数据。

PyTorch的命令式编程范式

- 借用了其他编程语言的控制流
 - · 好处: 框架本身不用关心If/Else, While如何实现
- 问题:
 - 无法序列化网络。不能很好的部署、集群并行。
 - 性能优化空间小。



编译器的主要模块

Source Code

Frontend

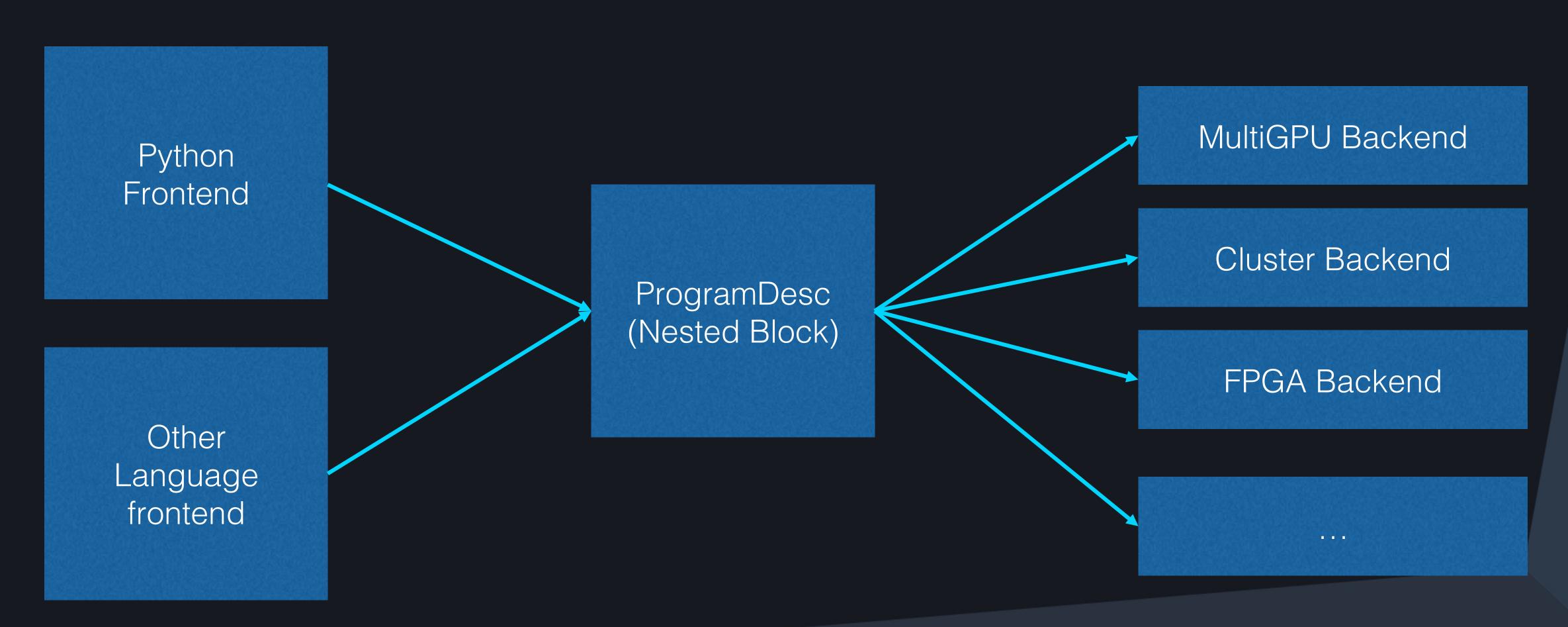
Common Optimizer

Backend

Machine Code



PaddlePaddle-Fluid编译器设计思路





编译阶段与运行阶段分离

- 编译阶段描述计算流程
 - ·编译结果是ProgramDesc
 - · 类似编程语言中的AST(抽象语法树)
 - 设备无关优化在编译阶段完成
- ·不同执行器后端根据设备特性对Program进行优化、运行



Block

· PaddlePaddle-Fluid将计算描述成嵌套的Block,而不是DAG。

编程语言	PaddlePaddle
For, while	WhileOp
If-Else, Switch	IfElseOp
顺序执行	指令(Op)序列



RNN / Loop

```
x = sequence([10, 20, 30]) # shape=[None, 1]
m = var(0) # shape=[1]
W = var(0.314, param=true) # shape=[1]
U = var(0.375, param=true) # shape=[1]
rnn = pd.rnn()
with rnn.step():
 x_ = rnn.step_input(x)
  h = rnn.memory(init = m)
  hh = rnn.previous_memory(h)
  a = layer.fc(W, x_{-})
  b = layer.fc(U, hh)
  s = pd.add(a, b)
  act = pd.sigmoid(s)
  rnn.update_memory(h, act)
  rnn.output(a, b)
o1, o2 = rnn()
```

```
int* x = \{10, 20, 30\};
int* m = \{0\};
int* W = \{0.314\};
int* U = \{0.375\};
int mem[sizeof(x) / sizeof(x[0]) + 1];
int o1[sizeof(x) / sizeof(x[0]) + 1];
int o2[sizeof(x) / sizeof(x[0]) + 1];
for (int i = 1; i \le sizeof(x)/sizeof(x[0]); ++i) {
 int x = x[i-1];
  if (i == 1) mem[0] = m;
 int a = W * x;
 int b = Y * mem[i-1];
  int s = fc_out + hidden_out;
  int act = sigmoid(sum);
  mem[i] = act;
  o1[i] = act;
  o2[i] = hidden_out;
```



If-else / IfElseOp

```
import paddle as pd
x = minibatch([10, 20, 30]) # shape=[None, 1]
y = var(1) # shape=[1], value=1
z = minibatch([10, 20, 30]) # shape=[None, 1]
cond = larger_than(x, 15) # [false, true, true]
ie = pd.ifelse()
with ie.true_block():
    d = pd.layer.add_scalar(x, y)
    ie.output(d, pd.layer.softmax(d))
with ie.false_block():
    d = pd.layer.fc(z)
    ie.output(d, d+1)
o1, o2 = ie(cond)
```

```
namespace pd = paddle;
int x = 10;
int y = 1;
int z = 10;
bool cond = false;
int o1, o2;
if (cond) {
  int d = x + y;
  o1 = z;
  o2 = pd::layer::softmax(z);
} else {
  int d = pd::layer::fc(z);
  o1 = d;
  o2 = d+1;
```

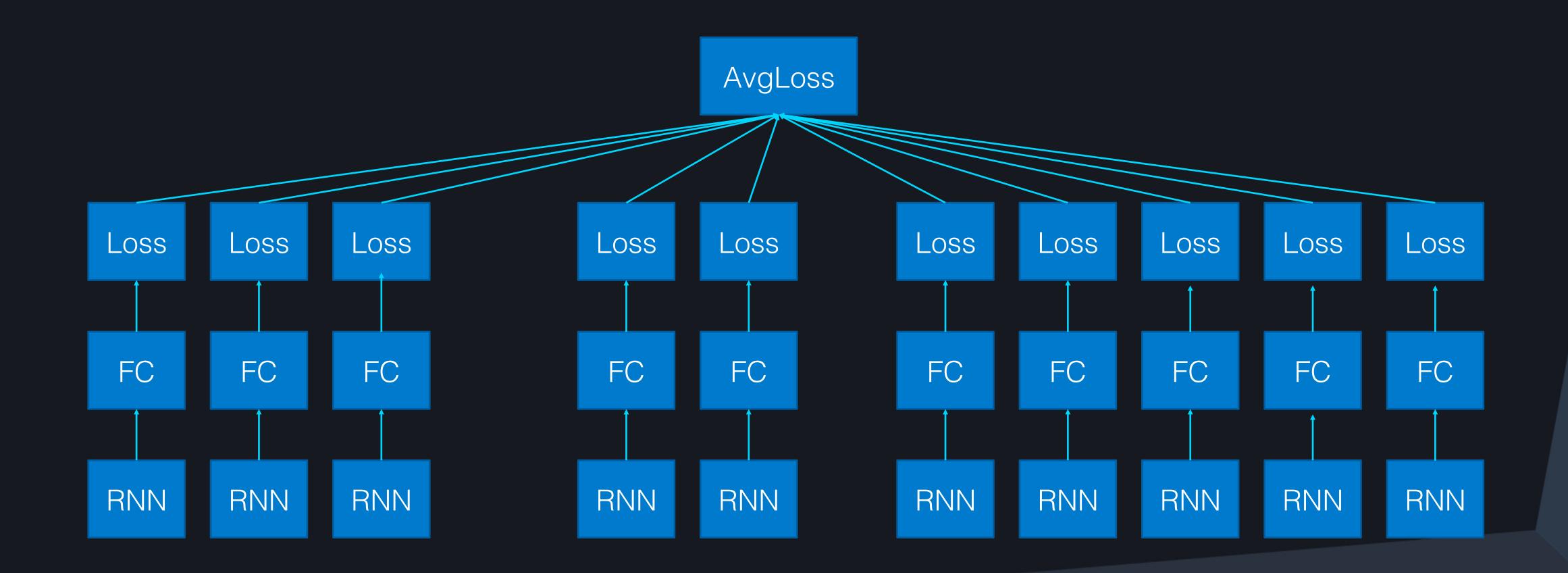


更强的IfElse/While

- 与PyTorch/DyNet实现动态网络的机制不同,Fluid实现动态的IfElse/ While操作
- · 『动态』:每一个mini-batch中的每条数据的分支都可以不同
- ·『高效』:数据自动基于Batch运行,无补零

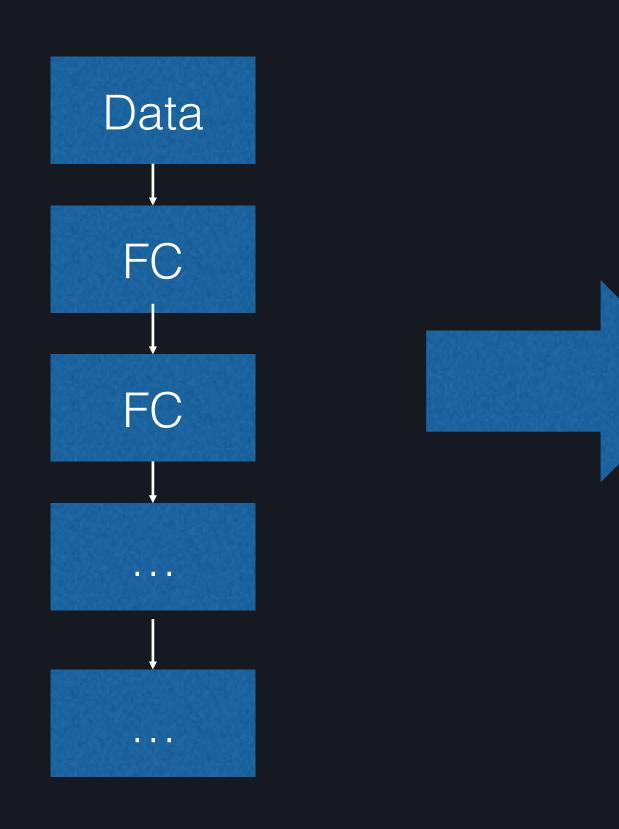


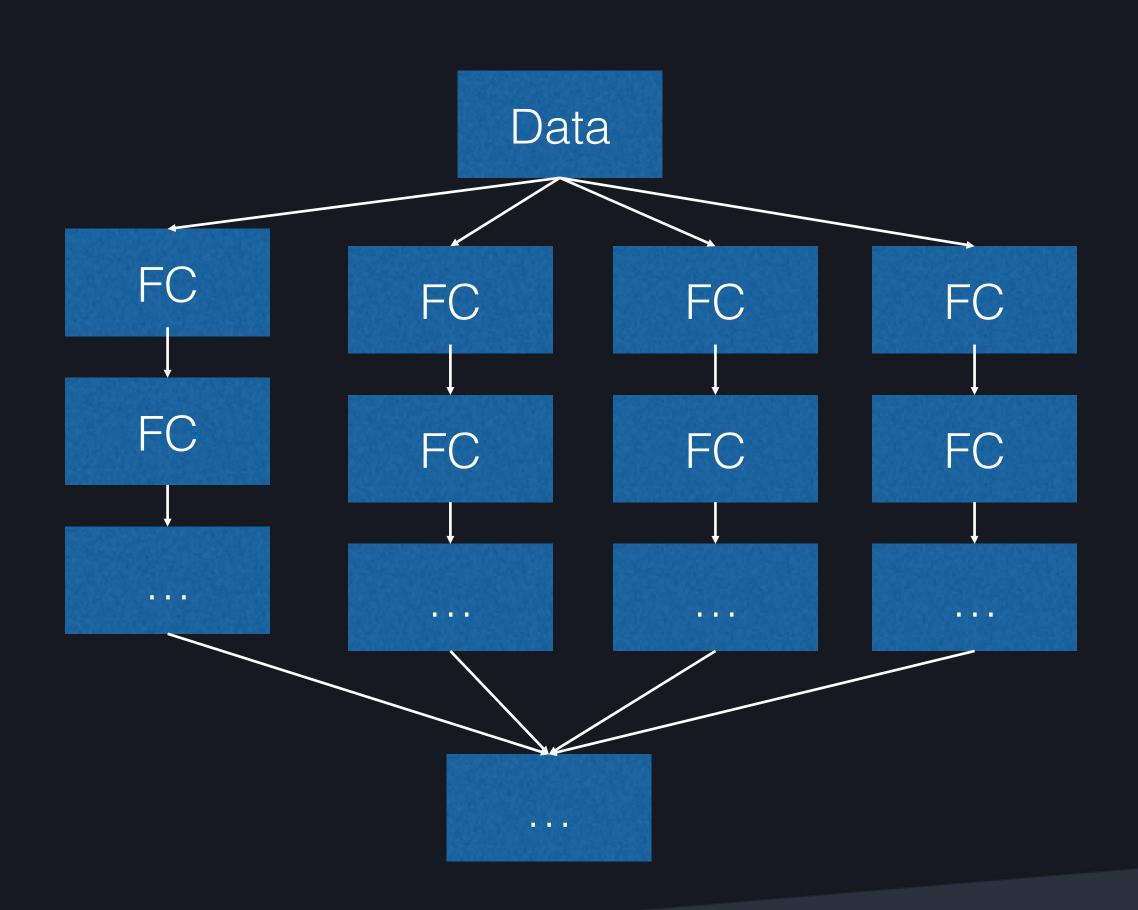
PaddlePaddle-Fluid RNN执行流程





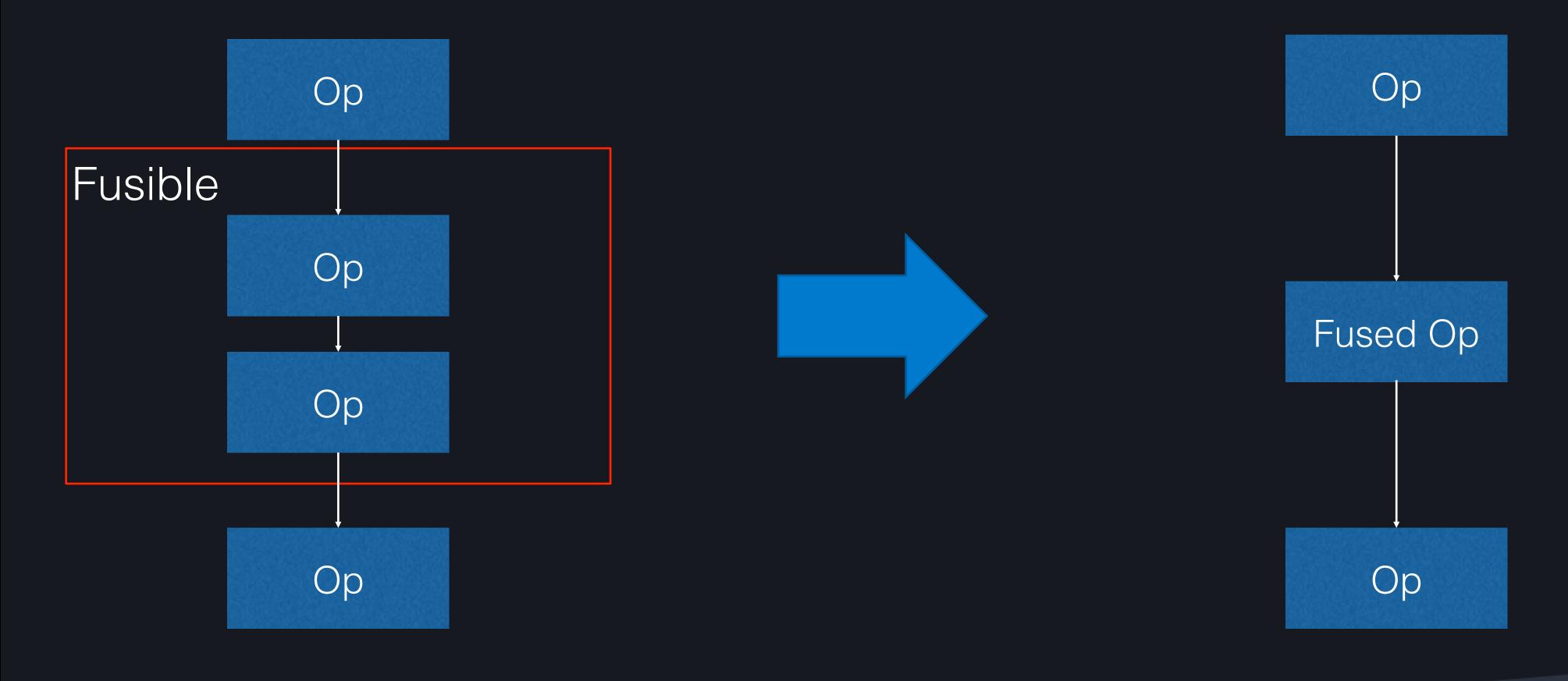
执行器优化:单设备一多设备







执行器优化: Kernel Fusion



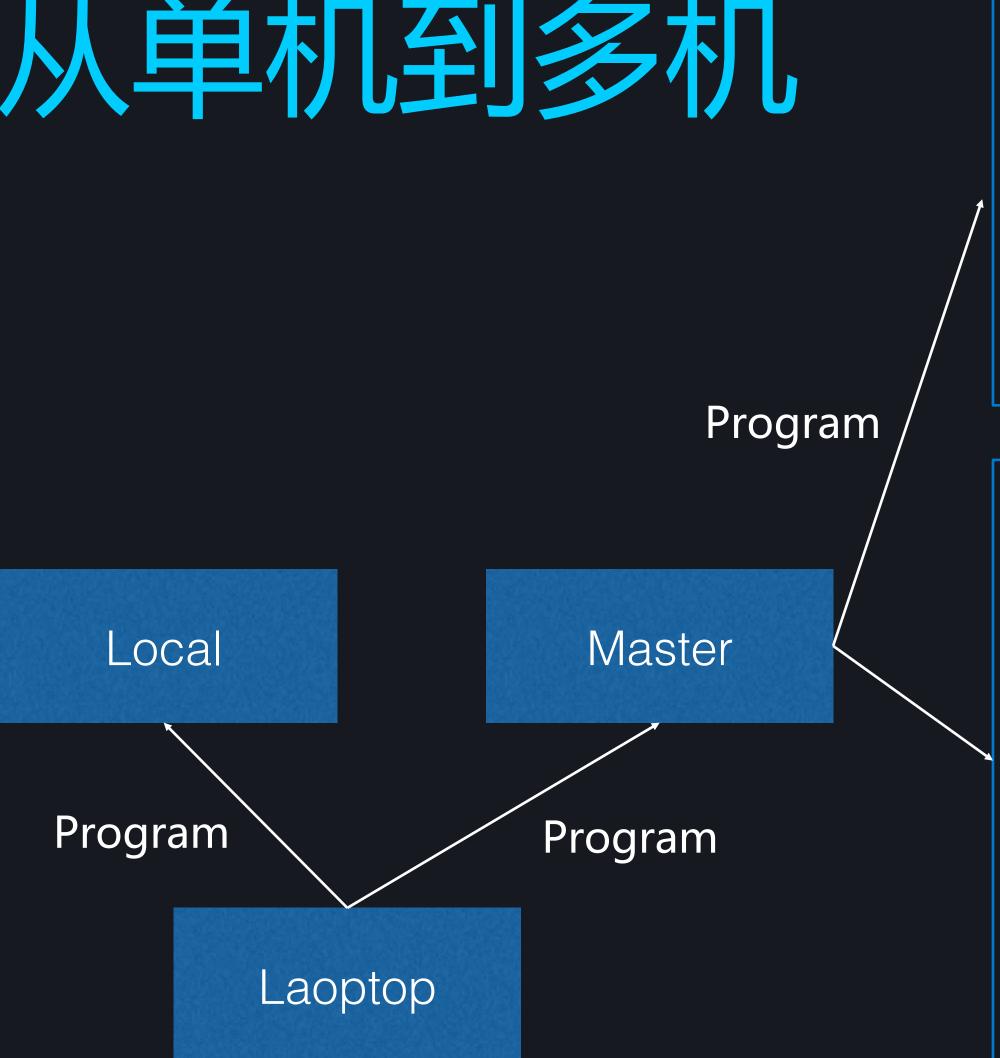


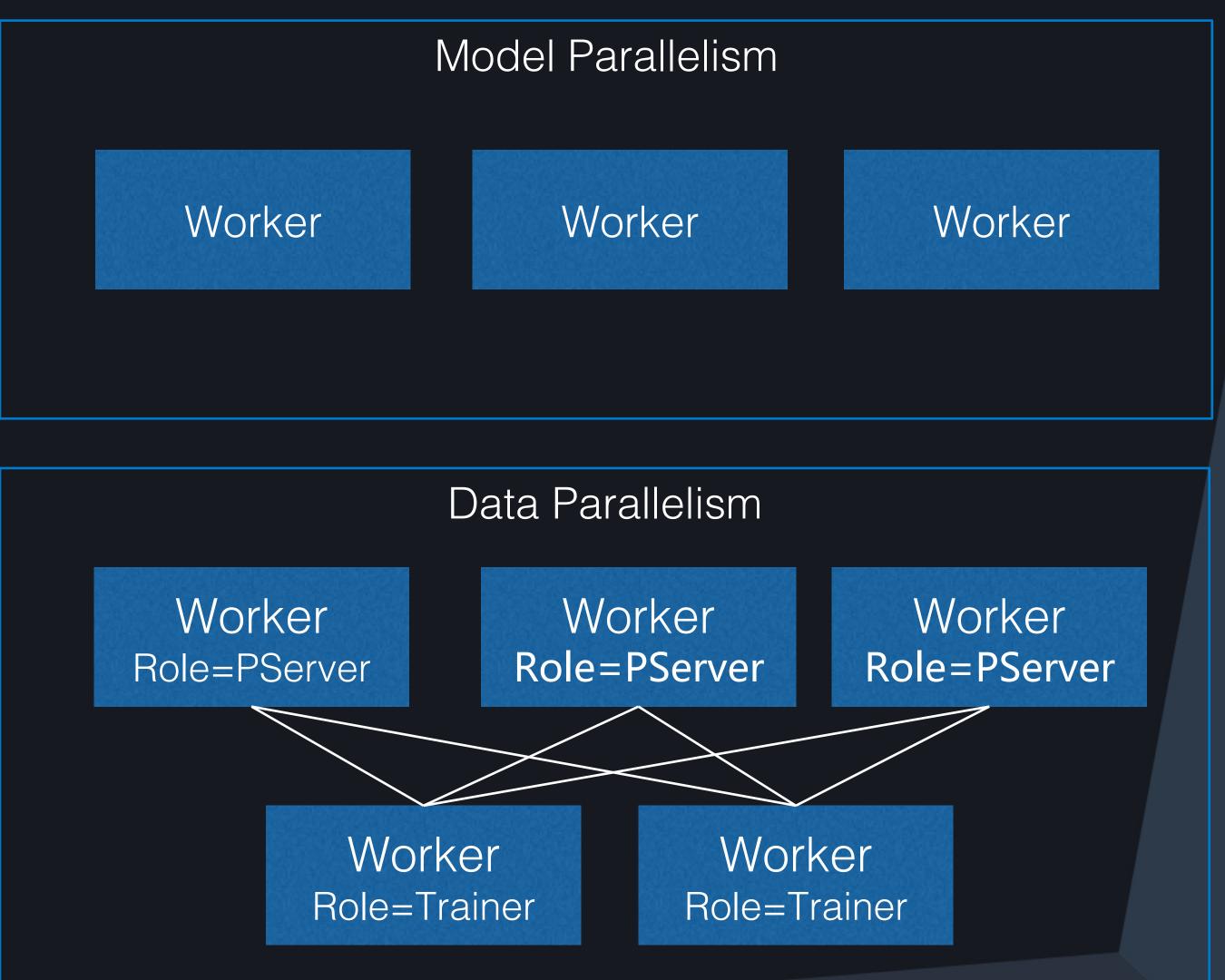
从单机到多机

- · 本地训练程序编译出中间结果 `Program`
- ·本地训练程序将中间结果上传给集群Master节点
- 集群Master节点将单机`Program`分解变换成集群每个节点需要执行的程序



从单机到多机







项目信息

- Main Repo
 - github.com/PaddlePaddle/Paddle
- Model Bank
 - github.com/PaddlePaddle/models

- Book
 - github.com/PaddlePaddle/book
- Cloud
 - github.com/PaddlePaddle/cloud



THANKYOU

如有需求,欢迎至[讲师交流会议室]与我们的讲师进一步交流

