# 今晚 Hackathon — 构建你的目标检测器

我们只提供测试数据，要求大家自行采集训练数据

- 如何采集并标注数据
- 如何处理上线后用户数据跟训练数据不一致 (covariate shift)

要求

- 组队参加 3—5人一对
- 我们提供需要检测的物体，大家可以用手机或者相机拍摄
- 需要自带笔记本（我们提供GPU实例）

http://1day-zh.d2l.ai

# 动手一天学深度学习

1 基础 · 2 卷积网络 · 3 计算 · 4 计算机视觉

深度学习实训营 2019

何通，李沐

http://1day-zh.d2l.ai

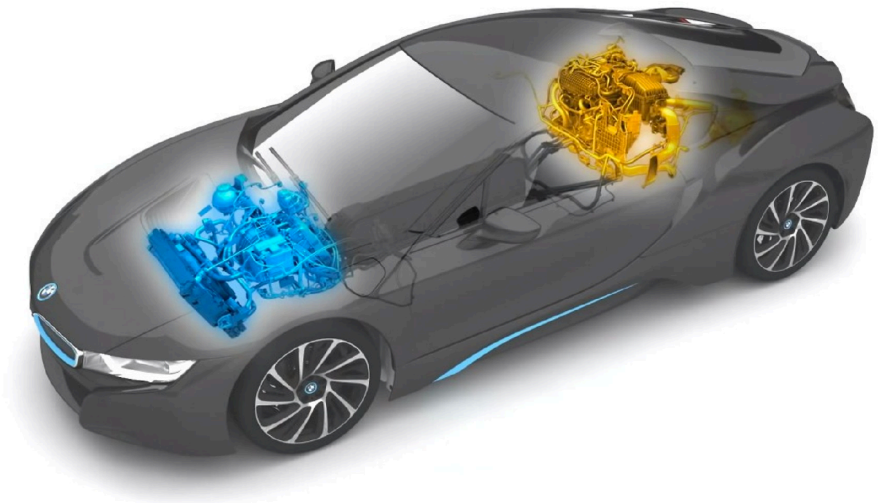


aws

# 大纲

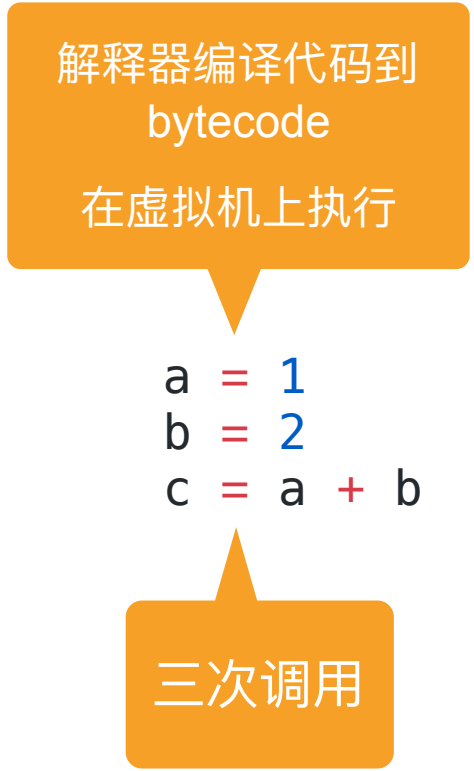
- 混合计算
- 异步计算
- 多GPU、多机训练

# 命令式和符号式混合编程



# 命令式编程

- Python, Java, C/C++的正常打开方式, …
- 直观，容易调试
- 需要Python解释器
  - 部署模型会困难（手机，浏览器，嵌入式设备）
  - 性能问题
    - 10 微秒额外开销



# 符号式编程

- 先定义计算，然后提供数据运行
- 数学, SQL, …
- 容易优化，极小前端额外代价，可移植
- 难用

知道了整个程序，容易优化

```
expr = "c = a + b"
exec = compile(expr)
exec(a=1, b=2)
```

可以不在Python上执行

一次调用

# Gluon的混合编程

- 通过 nn.HybridSequential 或者 nn.HybridBlock 定义模型
- 调用 .hybridize() 来从命令式转到到符号式执行

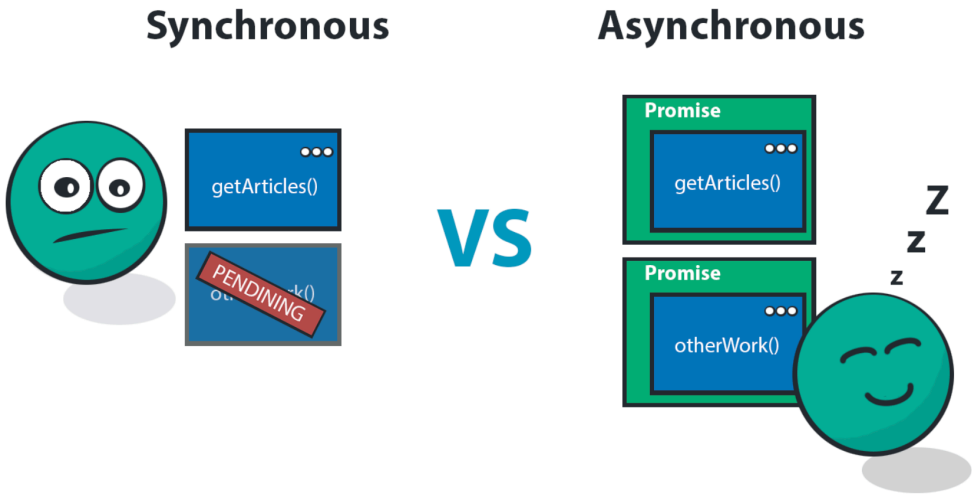
```
net = nn.HybridSequential()
net.add(nn.Dense(256, activation='relu'),
        nn.Dense(10))
net.hybridize()
```

http://1day-zh.d2l.ai

# Hybridize Notebook

http://1day-zh.d2l.ai

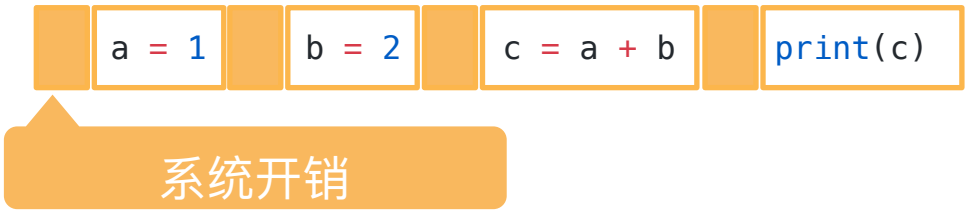
# 異步计算




http://1day-zh.d2l.ai

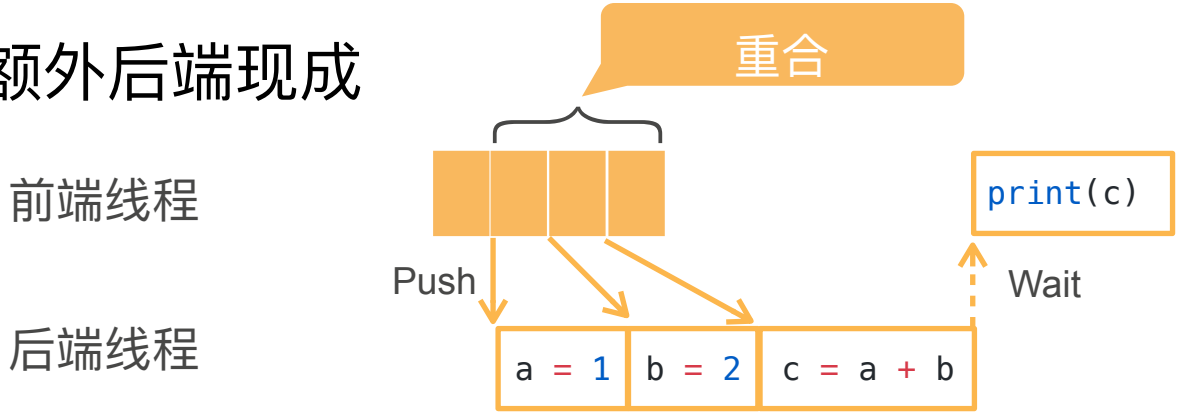
# 异步计算

- 逐个同步执行




```
a = 1
b = 2
c = a + b
print(c)
```

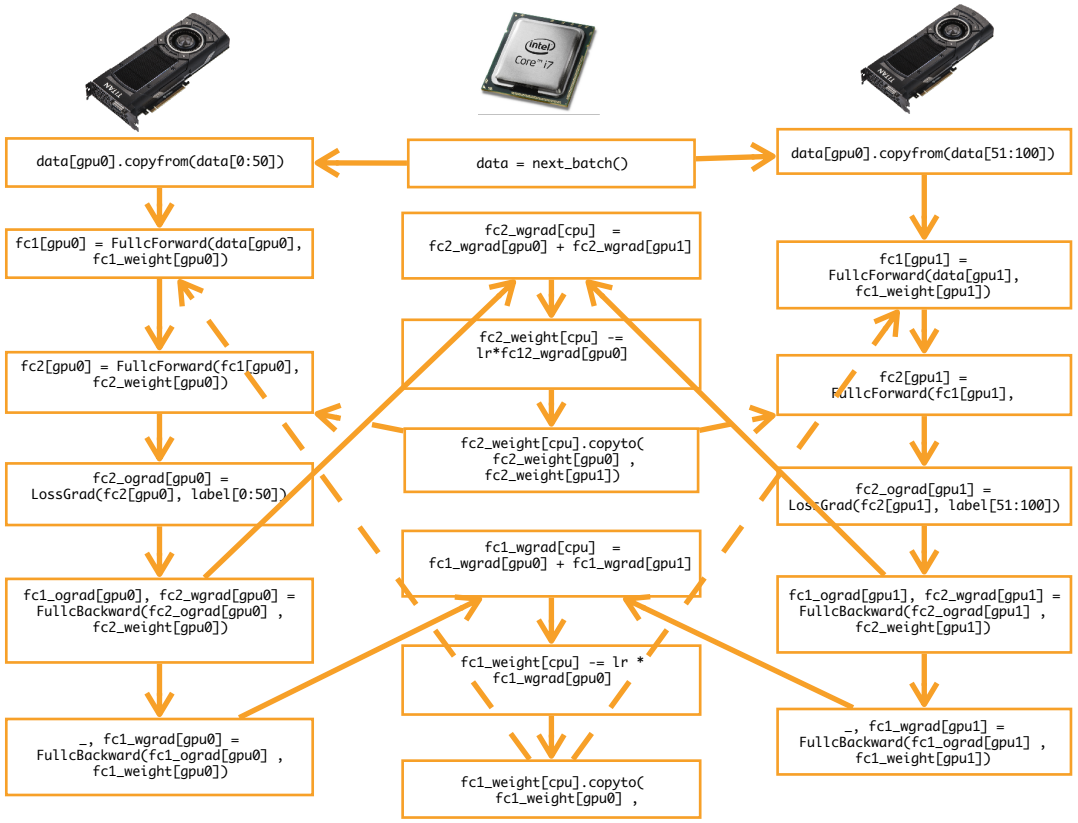
- 使用额外后端现成




http://1day-zh.d2l.ai

# 自动并行

# 并行编程很难




- 单隐藏层MLP在两个GPU上的依赖图
- 需要拓展到上百层和上十个设备上

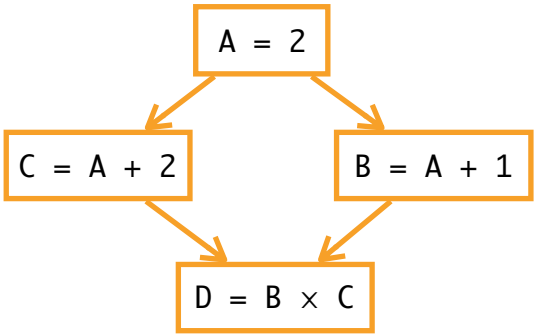
http://1day-zh.d2l.ai

# 自动并行

编写顺序程序

```
A = nd.ones((2,2)) * 2
C = A + 2
B = A + 1
D = B * C
```

平行执行



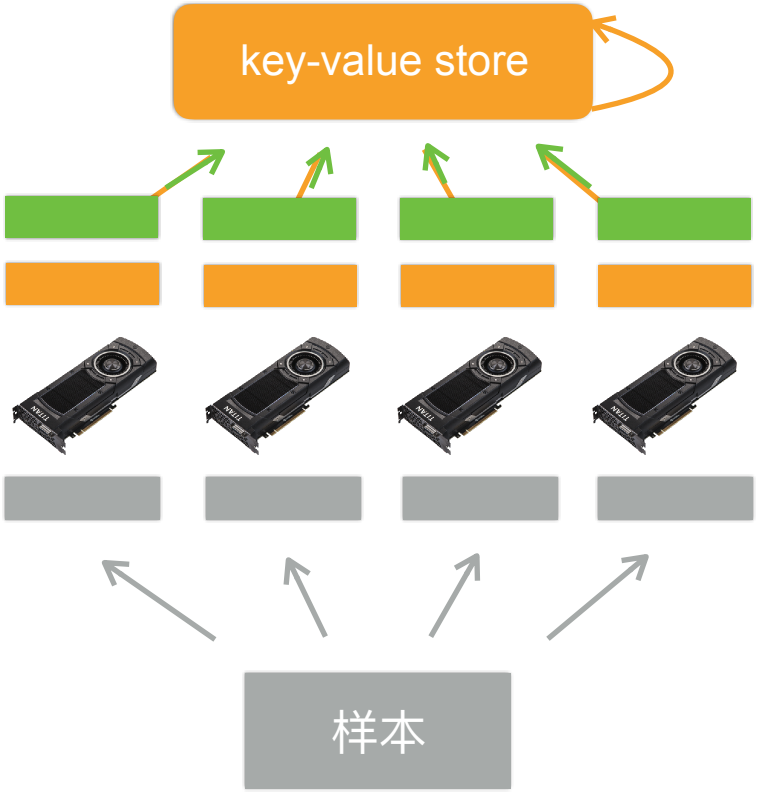
# 多GPU训练

2014年农历新年

http://1day-zh.d2l.ai

aws

# 数据并行



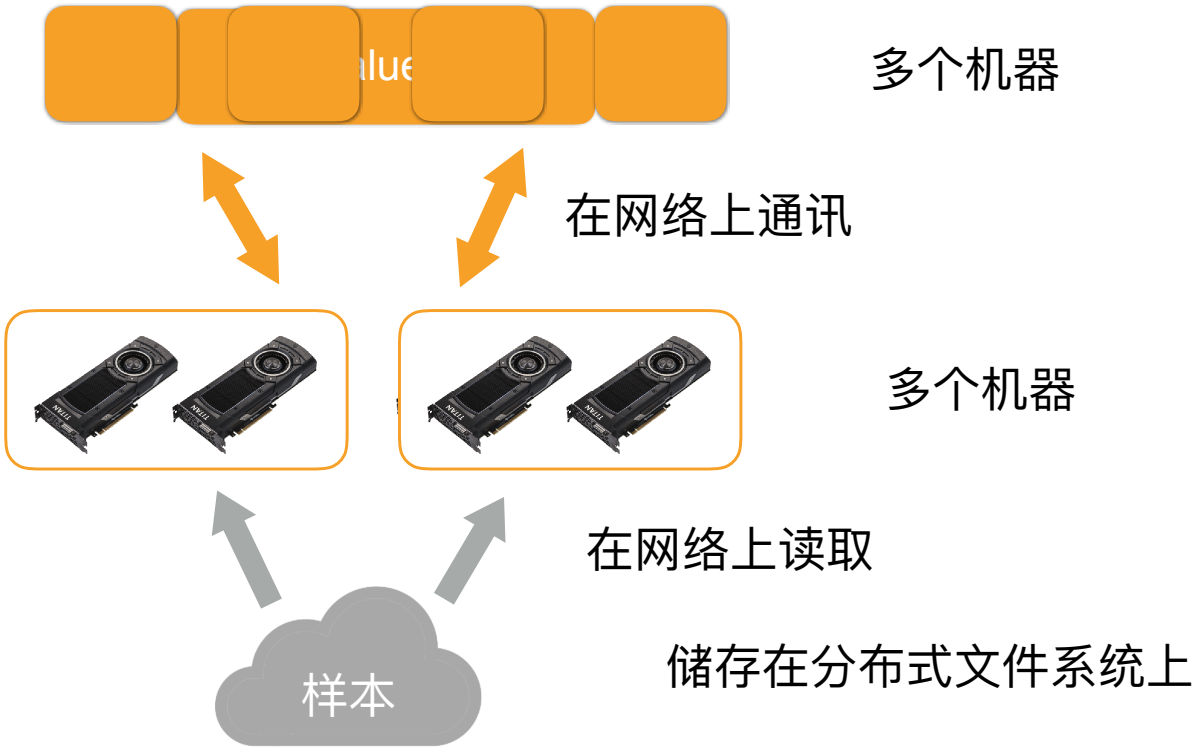

1. 读取一个数据批量
2. 读取模型参数
3. 计算梯度
4. 发送梯度
5. 更新模型参数

http://1day-zh.d2l.ai

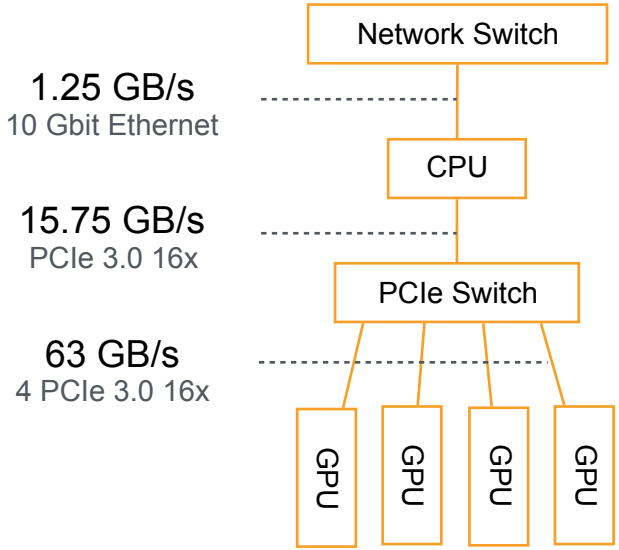
# 分布式计算

(Alex 在 CMU 组装的 GPU 集群，2015

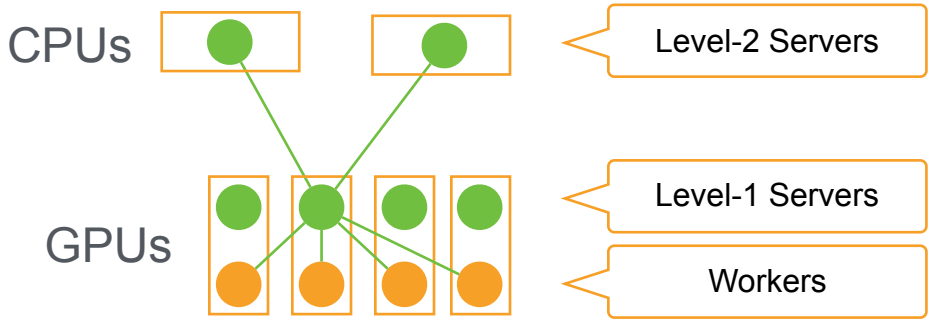
# 分布式计算



# GPU 机器的结构




层次化的 key-value store

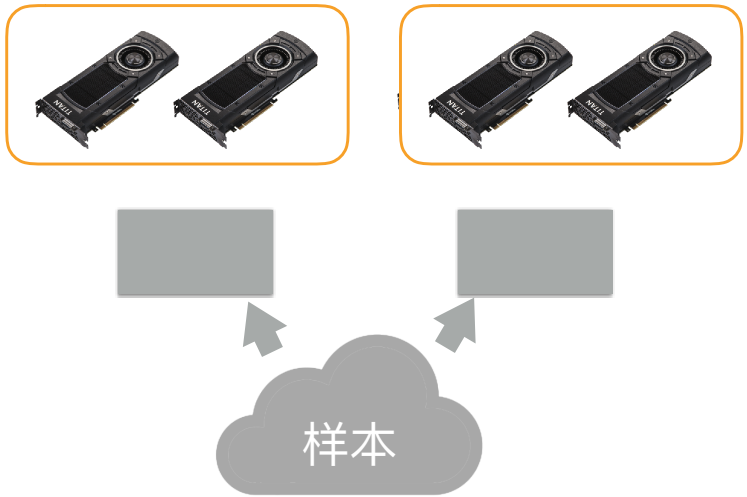



http://1day-zh.d2l.ai

aws

# 一个批量的计算

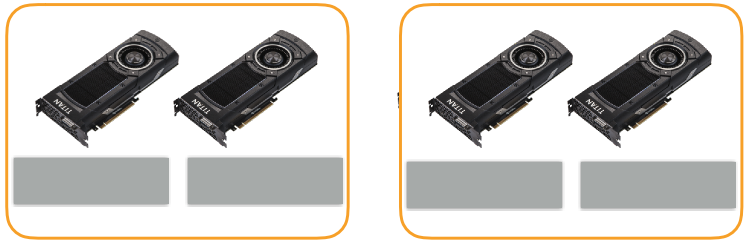
- 每个 worker 机器读取一个数据批量



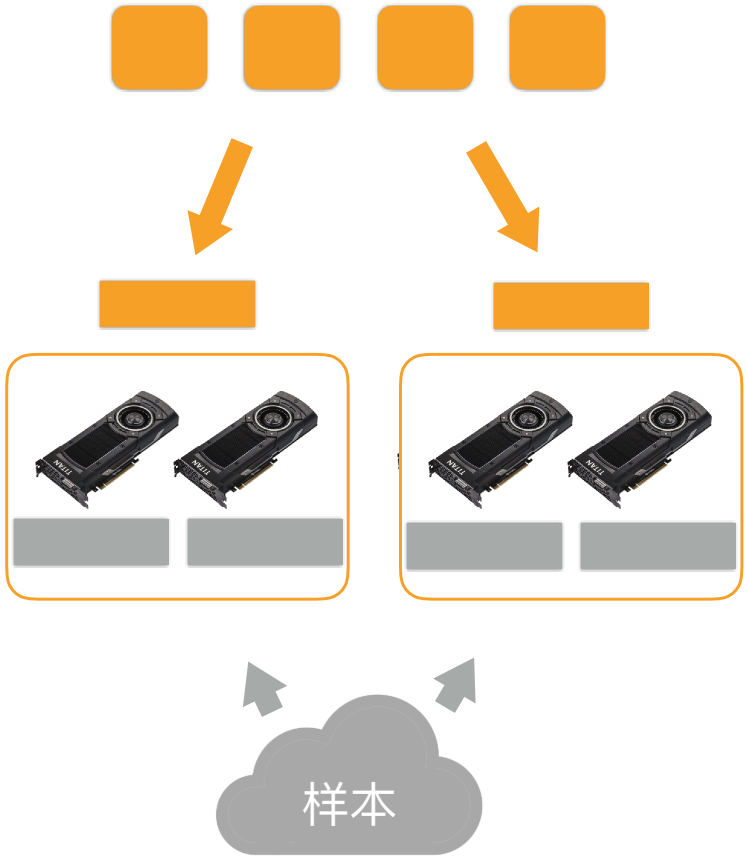

http://1day-zh.d2l.ai

# 一个批量的计算

- 将每个批量分到 GPU 上



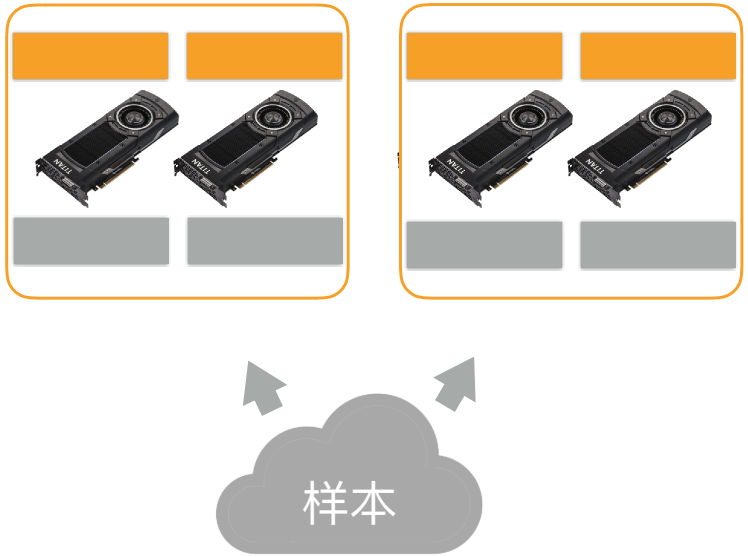
# 一个批量的计算




- 每个 server 维护一部分模型参数
- 每个 worker 读取所有的模型参数

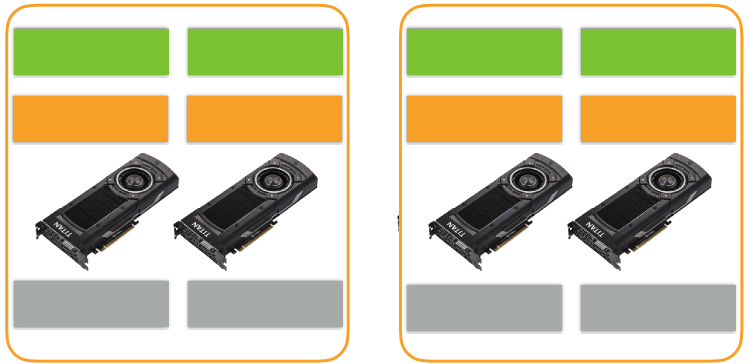
# 一个批量的计算

- 复制参数到 GPU

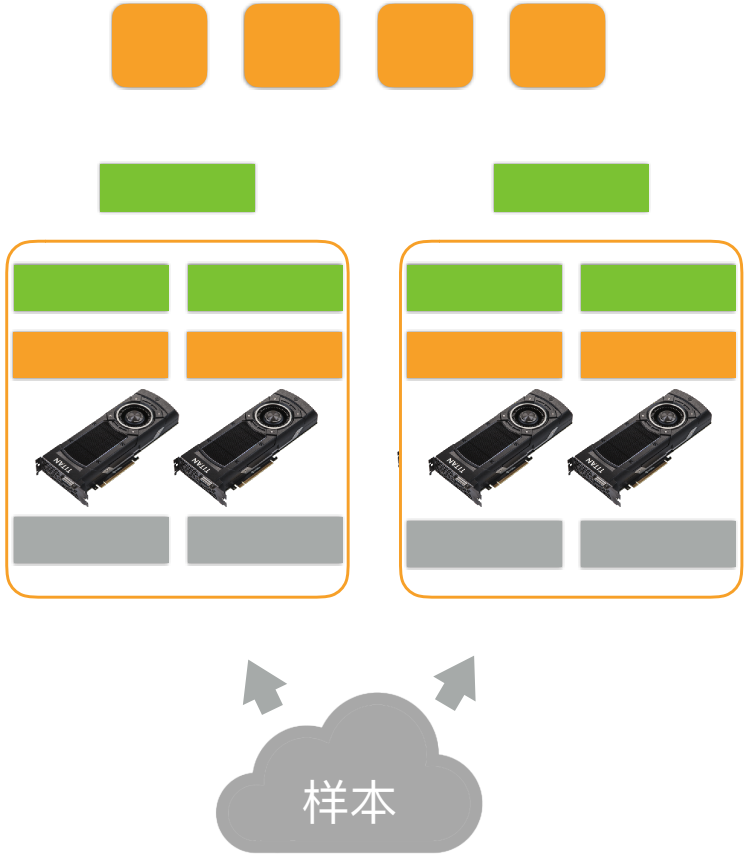


# 一个批量的计算

- 每个 GPU 计算梯度

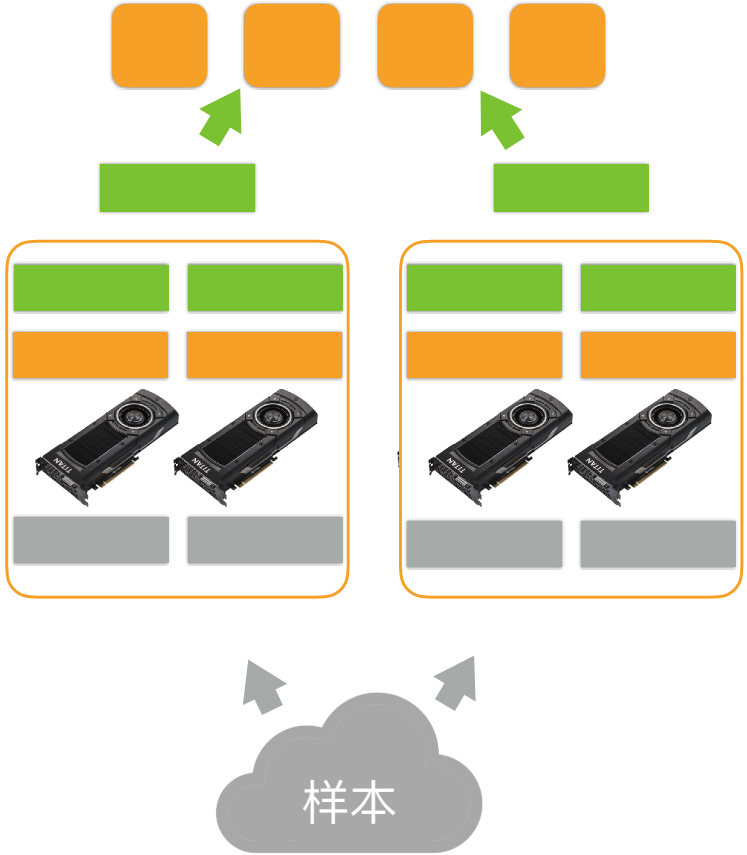


# 一个批量的计算




- 本地梯度求和

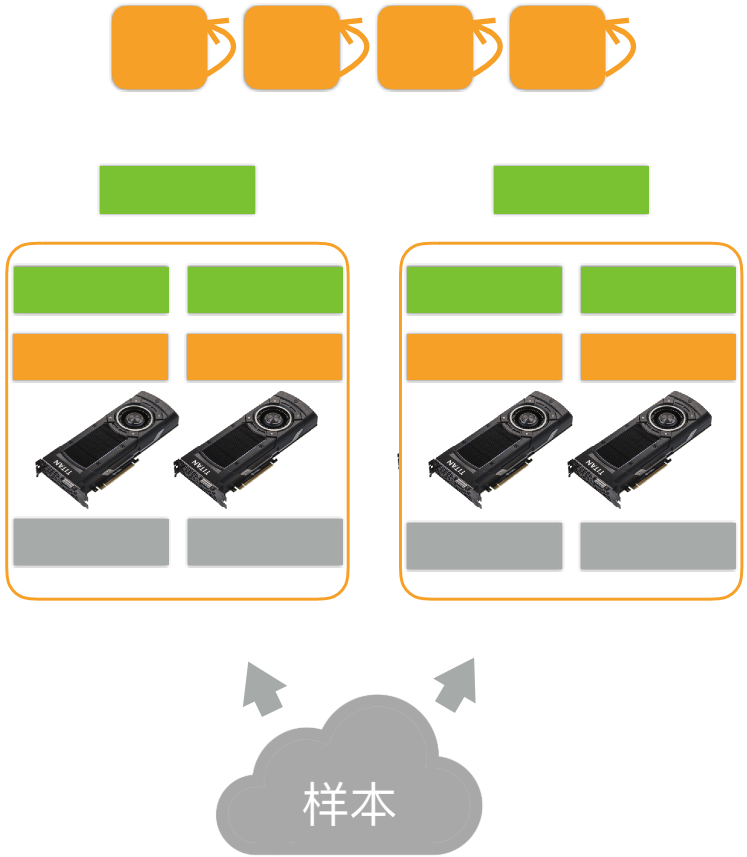
# 一个批量的计算




- 发送梯度到 server

http://1day-zh.d2l.ai

# 一个批量的计算




- 每个 server 对梯度求和，并更新模型参数

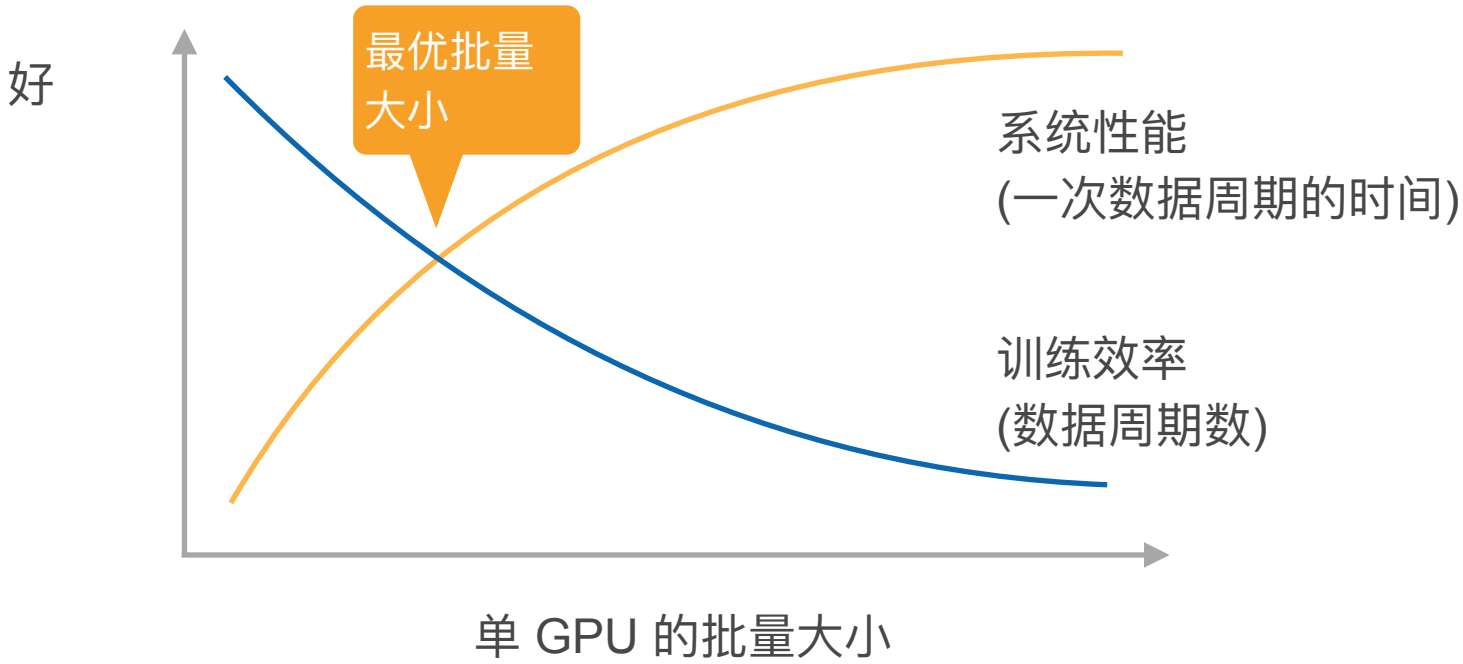
# 同步 SGD

- 每个 worker 同步执行
- 假设 $n$ 个 GPU，每个 GPU 每次处理 $b$ 个样本
  - 同步 SGD 等价于在单 GPU 上执行批量大小是 $nb$ 的 SGD
- 理想情况下，使用 $n$ GPU 会相对于单 GPU 有 $n$ 倍加速

# 性能

- T1 = O($b$): 单 GPU 计算 $b$ 个样本梯度的时间
- T2 = O($m$): 一个 worker 收发 $m$ 个参数的时间
- 每个批量的总时间是 max(T1, T2)
  - 理想情况: T1 > T2, 就是用足够大的 $b$
- 但过大的 $b$ 导致需要计算能多数据来达到理想的模型质量

# 性能的权衡



# 对实际情况的建议

- 使用一个大的数据集
- 很好的 GPU-GPU 和机器-机器带宽
- 高效的数据读取和预处理
- 模型需要有好的计算 (FLOP) vs 通讯 (模型大小) 比例
  - ResNet > AlexNet
- 用一个足够大的批量大小来保证系统性能
- 使用优化技巧来提升大批量下的训练效率

# Multi-GPU Notebooks

http://1day-zh.d2l.ai

# 总结

- 混合计算
- 异步计算
- 多GPU、多机训练

http://1day-zh.d2l.ai