TensorFlow 实战

黄文坚　唐源　著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

本书介绍了 TensorFlow 的基础原理和应用，并侧重于结合实际例子讲解使用 TensorFlow 的方法。TensorFlow 目前最主要的应用是在机器学习和深度学习领域，本书讲解了全连接神经网络、卷积神经网络、循环神经网络、深度强化学习等常见的深度学习模型，还介绍了 TensorBoard、单机多 GPU 并行、分布式并行，TF Learn 和其他 TensorFlow 辅助组件。

希望快速上手 TensorFlow、了解深度学习技术及其应用实践的人士，以及机器学习、分布式计算领域的学生、从业者。

图书在版编目（CIP）数据

TensorFlow 实战 / 黄文坚，唐源著. —北京：电子工业出版社，2017.2
ISBN 978-7-121-30912-0

Ⅰ. ①T… Ⅱ. ①黄… ②唐… Ⅲ. ①人工智能—算法—研究 Ⅳ. ①TP18

中国版本图书馆 CIP 数据核字(2017)第 014533 号

策划编辑：郑柳洁
责任编辑：郑柳洁
印　　刷：三河市鑫金马印装有限公司
装　　订：三河市鑫金马印装有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：720×1000　1/16　印张：19.5　字数：335 千字
版　　次：2017 年 2 月第 1 版
印　　次：2017 年 2 月第 2 次印刷
定　　价：79.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819，faq@phei.com.cn。

好评袭来

"AI and Machine Learning are going to be a key part of our future. We made TensorFlow open source to bring these technologies to everyone and help move the world forward. This book is a great example of the TensorFlow community giving back to multiply everyone's efforts."

Engineering Director of TensorFlow，Rajat Monga

TensorFlow 的开源对整个学术界及工业界都产生了巨大的影响，可以比做机器学习的 Hadoop。本书涵盖了从多层感知机、CNN、RNN 到强化学习等一系列模型的 TensorFlow 实现；在详尽地介绍算法和模型的细节的同时穿插实际的代码，对帮助读者快速建立算法和代码的联系大有助益；对入门 TensorFlow 和深度学习的研究者来说是一份非常好的学习材料。

360 首席科学家，颜水成

TensorFlow 是基于 Computation Graph 的机器学习框架，支持 GPU 和分布式，是目前最有影响力的开源深度学习系统。TensorFlow 的工程实现非常优秀，拓展也非常灵活，对机器学习尤其是深度学习的推广大有裨益。本书结合了大量的实际例子，清晰地讲解了

如何使用 TensorFlow 构筑常见的深度学习模型，可通读也可作为工具书查阅。在本书上市前，国内还没有介绍 TensorFlow 的技术书籍，推荐对 TensorFlow 或深度学习感兴趣的人士阅读此书。

北京大学计算机系教授 网络与信息系统研究所所长，崔斌

深度学习乃至人工智能正逐渐在 FinTech 领域发挥巨大的作用，其应用包括自动报告生成、金融智能搜索、量化交易和智能投顾。而 TensorFlow 为金融业方便地使用深度学习提供了可能。本书介绍了通过 TensorFlow 实现各类神经网络的案例，非常适合初学者快速入门。

PPmoney CTO，康德胜

TensorFlow 是 Google 开源的一套深度学习框架，已发展成为最主流的深度学习框架，目前在市面上没有看到关于 TensorFlow 的中文书籍出版。本书一方面一步步地介绍了 TensorFlow 的使用方法，使得没有使用过的人可以很快上手使用；另一方面，讲解了诸如卷积神经网络、循环神经网络、强化学习、自编码器等深度学习知识，使得不懂深度学习的人也可以入门。本书在介绍基本知识和原理的同时，用实例进行讲解，比较适合初学者学习使用 TensorFlow 及深度学习知识。

格灵深瞳 CTO，邓亚峰

《TensorFlow 实战》由浅入深，透过大量的代码实例，为读者揭开深度学习的层层面纱，加深理论理解的同时，也更好地联系了实际应用。

小米图像算法资深工程师，万韶华

前言

AlphaGo 在 2017 年年初化身 Master，在弈城和野狐等平台上连胜中日韩围棋高手，其中包括围棋世界冠军井山裕太、朴廷桓、柯洁等，还有棋圣聂卫平，总计取得 60 连胜，未尝败绩。设想 2016 年 3 月，当时 AlphaGo 挑战李世石还一度不被看好，到今日已经可以完胜各位高手。AlphaGo 背后神秘的推动力就是 TensorFlow——Google 于 2015 年 11 月开源的机器学习及深度学习框架。DeepMind 宣布全面迁移到 TensorFlow 后，AlphaGo 的算法训练任务就全部放在了 TensorFlow 这套分布式框架上。

TensorFlow 在 2015 年年底一出现就受到了极大的关注，在一个月内获得了 GitHub 上超过一万颗星的关注，目前在所有的机器学习、深度学习项目中排名第一，甚至在所有的 Python 项目中也排名第一。本书将重点从实用的层面，为读者讲解如何使用 TensorFlow 实现全连接神经网络、卷积神经网络、循环神经网络，乃至 Deep Q-Network。同时结合 TensorFlow 原理，以及深度学习的部分知识，尽可能让读者通过学习本书做出实际项目和成果。

本书各章节间没有太强的依赖关系，如果读者对某一章感兴趣，可以直接阅读。本书使用 TensorFlow 1.0.0-rc0 作为示例讲解，应该与最新版的 TensorFlow 兼容绝大部分代码，可能存在少数接口的更新，读者可参阅提示信息。书中大部分代码是 Python 代码，这也是 TensorFlow 支持的最全、最完整的接口语言。

本书的前两章介绍了 TensorFlow 的基础知识和概念。第 3 章和第 4 章介绍了简单的

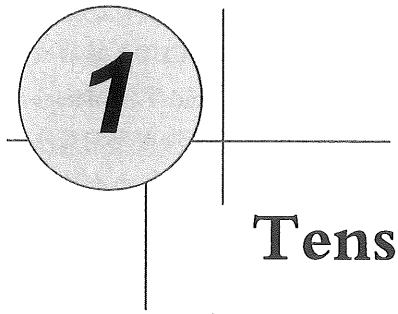
示例及全连接神经网络。第 5 章和第 6 章介绍了基础的卷积神经网络，以及目前比较经典的 AlexNet、VGGNet、Inception Net 和 ResNet。第 7 章介绍了 Word2Vec、RNN 和 LSTM。第 8 章介绍了强化学习，以及基于深度学习的策略网络和估值网络。第 9 章介绍了 TensorBoard、单机多 GPU 并行，以及分布式并行。

第 10 章介绍了 TensorFlow 里面的 contrib.learn 模块，包含许多类型的深度学习及流行的机器学习算法的使用方法，也解析了这个模块的分布式 Estimator 的基本架构，以及如何使用 Estimator 快速搭建自己的分布式机器学习模型架构，进行模型的训练和评估，也介绍了如何使用监督器更好地监测和跟踪模型的训练及使用 DataFrame 读取不同的数据格式。第 11 章介绍了 Contrib 模块，这个模块里提供了许多机器学习需要的功能，包括统计分布、机器学习层、优化函数、指标，等等。本章将简单介绍其中的一些功能让大家了解 TensorFlow 的涵盖范围，并感受到社区的积极参与和贡献度。第 10 章和第 11 章使用了 TensorFlow 0.11.0-rc0 版本作为示例讲解。

作者在写作本书时，获得了亲人、同事、好友的帮助，在此非常感谢你们的支持。

作 者

目录



TensorFlow 基础

1.1 TensorFlow 概要

Google 第一代分布式机器学习框架 DistBelief[1]，在内部大规模使用后并没有选择开源。而后第二代分布式机器学习系统 TensorFlow[2]终于选择于 2015 年 11 月在 GitHub 上开源，且在 2016 年 4 月补充了分布式版本，并于 2017 年 1 月发布了 1.0 版本的预览，API 接口趋于稳定。目前 TensorFlow 仍处于快速开发迭代中，有大量新功能及性能优化在持续研发。TensorFlow 最早由 Google Brain 的研究员和工程师开发，设计初衷是加速机器学习的研究，并快速地将研究原型转化为产品。Google 选择开源 TensorFlow 的原因也非常简单：第一是希望通过社区的力量，让大家一起完善 TensorFlow。之前 Google 内部 DistBelief 及 TensorFlow 的用户就贡献了非常多的意见和反馈，使得产品质量得到了快速提升；第二是回馈社区，Google 希望让这个优秀的工具得到更多的应用，从整体上提高学术界乃至工业界使用深度学习的效率。除了 TensorFlow，Google 也开源过大量成功的项目，包括大名鼎鼎的移动操作系统 Android、浏览器 Chromium、编程语言 Go、JavaScript 引擎 V8、数据交换框架 Protobuf、编译工具 Bazel、OCR 工具 Tesseract 等共计数百个高质量的项目。

TensorFlow 的官方网址：www.tensorflow.org

GitHub 网址：github.com/tensorflow/tensorflow

模型仓库网址：github.com/tensorflow/models

TensorFlow 既是一个实现机器学习算法的接口，同时也是执行机器学习算法的框架。它前端支持 Python、C++、Go、Java 等多种开发语言，后端使用 C++、CUDA 等写成。TensorFlow 实现的算法可以在众多异构的系统上方便地移植，比如 Android 手机、iPhone、普通的 CPU 服务器，乃至大规模 GPU 集群，如图 1-1 所示。除了执行深度学习算法，TensorFlow 还可以用来实现很多其他算法，包括线性回归、逻辑回归、随机森林等。TensorFlow 建立的大规模深度学习模型的应用场景也非常广，包括语音识别、自然语言处理、计算机视觉、机器人控制、信息抽取、药物研发、分子活动预测等，使用 TensorFlow 开发的模型也在这些领域获得了最前沿的成果。

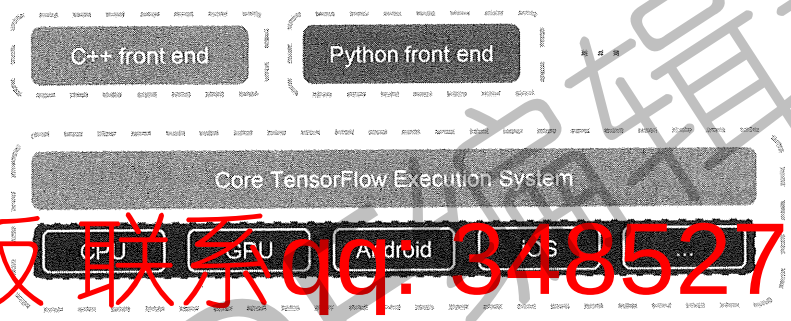


图 1-1　TensorFlow 基础架构

为了研究超大规模的深度神经网络，Google 在 2011 年启动了 Google Brain 项目，同时开发了第一代的分布式机器学习框架 DistBelief。有超过 50 个 Google 的团队在他们的产品中使用了 DistBelief，比如 Google Search 中的搜索结果排序、Google Photos 中的图片标注、Google Translate 中的自然语言处理等，都依赖于 DistBelief 建立的深度学习模型。Google 基于使用 DistBelief 时的经验及训练大规模分布式神经网络的需求，开发了 TensorFlow——第二代分布式机器学习算法实现框架和部署系统。Google 将著名的 Inception Net 从 DistBelief 移植到 TensorFlow 后，获得了 6 倍的训练速度提升。目前，在 Google 内部使用 TensorFlow 的项目呈爆炸性的增长趋势，在 2016 年已经有超过 2000 个项目使用了 TensorFlow 建立的深度学习模型，而且这个数字还在高速增长中，如图 1-2 所示。

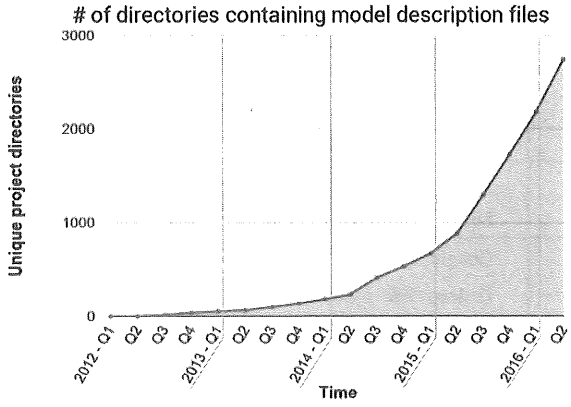


图 1-2 TensorFlow 在 Google 的使用趋势

TensorFlow 使用数据流式图来规划计算流程，它可以将计算映射到不同的硬件和操作系统平台。凭借着统一的架构，TensorFlow 可以方便地部署到各种平台，大大简化了真实场景中应用机器学习的难度。使用 TensorFlow 我们不需要给大规模的模型训练和小规模的应用部署开发两套不同的系统，避免了同时维护两套程序的成本，TensorFlow 给训练和预测的共同部分提供了一个恰当的抽象。TensorFlow 的计算可以表示为有状态的数据流式图，对于大规模的神经网络训练，TensorFlow 可以让用户简单地实现并行计算，同时使用不同的硬件资源进行训练，同步或异步地更新全局共享的模型参数和状态。将一个串行的 TensorFlow 算法改造成并行的成本也是非常低的，通常只需要对小部分代码进行改写。相比于 DistBelief，TensorFlow 的计算模型更简洁灵活，计算性能显著提升，同时支持更多的异构计算系统。大量 Google 内部的 DistBelief 用户转向了 TensorFlow，他们使用 TensorFlow 进行各种研究和产品开发，包括在手机上跑计算机视觉模型，或是训练有数百亿参数、数千亿数据的神经网络模型。虽然绝大多数的 TensorFlow 应用都在机器学习及深度学习领域，但 TensorFlow 抽象出的数据流式图也可以应用在通用数值计算和符号计算上，比如分形图计算或者偏微分方程数值求解。表 1-1 所示为 TensorFlow 的主要技术特性。

表 1-1 TensorFlow 的主要技术特性

编程模型	Dataflow-like model（数据流模型）
语言	Python、C++、Go、Rust、Haskell、Java（还有非官方的 Julia、JavaScript、R 的支持）
部署	Code once, run everywhere（一次编写，各处运行）

续表

计算资源	CPU（Linux、Mac、Windows、Android、iOS）
	GPU（Linux、Mac、Windows）
	TPU（Tensor Processing Unit，张量计算单元，主要用作推断）
实现方式	Local Implementation（单机实现）
	Distributed Implementation（分布式实现）
平台支持	Google Cloud Platform（谷歌云平台）
	Hadoop File System（Hadoop 分布式文件系统）
数学表达	Math Graph Expression（数学计算图表达）
	Auto Differentiation（自动微分）
优化	Common Subexpression Elimination（共同子图消除）
	Asynchronous Kernel Optimization（异步核优化）
	Communication Optimization（通信优化）
	Model Parallelism（模型并行）
	Data Parallelism（数据并行）
	Pipeline（流水线）

1.2 TensorFlow 编程模型简介

1.2.1 核心概念

TensorFlow 中的计算可以表示为一个有向图（directed graph），或称计算图（computation graph），其中每一个运算操作（operation）将作为一个节点（node），节点与节点之间的连接称为边（edge）。这个计算图描述了数据的计算流程，它也负责维护和更新状态，用户可以对计算图的分支进行条件控制或循环操作。用户可以使用 Python、C++、Go、Java 等几种语言设计这个数据计算的有向图。计算图中每一个节点可以有任意多个输入和任意多个输出，每一个节点描述了一种运算操作，节点可以算是运算操作的实例化（instance）。在计算图的边中流动（flow）的数据被称为张量（tensor），故得名 TensorFlow。而 tensor 的数据类型，可以是事先定义的，也可以根据计算图的结构推断得到。有一类特殊的边中没有数据流动，这种边是依赖控制（control dependencies），作用是让它的起始节点执行完之后再执行目标节点，用户可以使用这样的边进行灵活的条件控制，比如限制内存使用的最高峰值。下面是用 Python 设计并执行计算图的示例。计算图示例如图 1-3 所示。

```
import tensorflow as tf
b=tf.Variable(tf.zeros([100]))          # 生成 100 维的向量，初始化为 0
W=tf.Variable(tf.random_uniform([784,100],-1,1)) # 生成 784x100 的随机矩阵 W
x=tf.placeholder(name="x")          # 输入的 Placeholder
relu=tf.nn.relu(tf.matmul(W, x)+b)  # ReLU(Wx+b)
C=[...]                             # 根据 ReLU 函数的结果计算 Cost
s=tf.Session()
for step in range(0, 10):
    input=...construct 100-D input array... # 为输入创建一个 100 维的向量
    result=s.run(C, feed_dict={x: input})     # 获取 Cost，供给输入 x
    print(step, result)
```

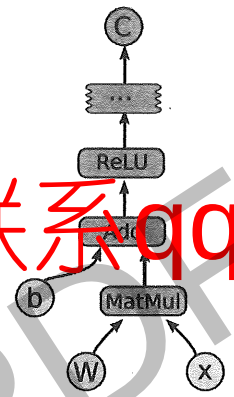


图 1-3 计算图示例

一个运算操作代表了一种类型的抽象运算，比如矩阵乘法或者向量加法。运算操作可以有自己的属性，但是所有属性必须被预先设置，或者能在创建计算图时被推断出来。通过设置运算操作的属性可以用来支持不同的 tensor 元素类型，比如让向量加法支持浮点数（float）或者整数（int）。运算核（kernel）是一个运算操作在某个具体硬件（比如在 CPU 或者 GPU 中）的实现。在 TensorFlow 中，可以通过注册机制加入新的运算操作或者运算核。表 1-2 所示为部分 TensorFlow 内建的运算操作。

表 1-2 TensorFlow 内建的运算操作

类　　型	示　　例
标量运算	Add、Sub、Mul、Div、Exp、Log、Greater、Less、Equal

续表

类　　型	示　　例
向量运算	Concat、Slice、Split、Constant、Rank、Shape、Shuffle
矩阵运算	MatMul、MatrixInverse、MatrixDeterminant
带状态的运算	Variable、Assign、AssignAdd
神经网络组件	SoftMax、Sigmoid、ReLU、Convolution2D、MaxPooling
储存、恢复	Save、Restore
队列及同步运算	Enqueue、Dequeue、MutexAcquire、MutexRelease
控制流	Merge、Switch、Enter、Leave、NextIteration

Session 是用户使用 TensorFlow 时的交互式接口。用户可以通过 Session 的 Extend 方法添加新的节点和边，用以创建计算图，然后就可以通过 Session 的 Run 方法执行计算图：用户给出需要计算的节点，同时提供输入数据，TensorFlow 就会自动寻找所有需要计算的节点并按依赖顺序执行它们。对绝大多数的用户来说，他们只会创建一次计算图，然后反复地执行整个计算图或是其中的一部分子图（sub-graph）。

在大多数运算中，计算图会被反复执行多次，而数据也就是 tensor 并不会被持续保留，只是在计算图中过一遍。Variable 是一类特殊的运算操作，它可以将一些需要保留的 tensor 储存在内存或显存中，比如神经网络模型中的系数。每一次执行计算图后，Variable 中的数据 tensor 将会被保存，同时在计算过程中这些 tensor 也可以被更新，比如神经网络每一次 mini-batch 训练时，神经网络的系数将会被更新并保存。使用 Variable，可以在计算图中实现一些特殊的操作，比如 Assign、AssignAdd（+=）或 AssignMul（*=）。

1.2.2　实现原理

TensorFlow 有一个重要组件 client，顾名思义，就是客户端，它通过 Session 的接口与 master 及多个 worker 相连。其中每一个 worker 可以与多个硬件设备（device）相连，比如 CPU 或 GPU，并负责管理这些硬件。而 master 则负责指导所有 worker 按流程执行计算图。TensorFlow 有单机模式和分布式模式两种实现，其中单机指 client、master、worker 全部在一台机器上的同一个进程中；分布式的版本允许 client、master、worker 在不同机器的不同进程中，同时由集群调度系统统一管理各项任务。图 1-4 所示为单机版和分布式版本的示例图。

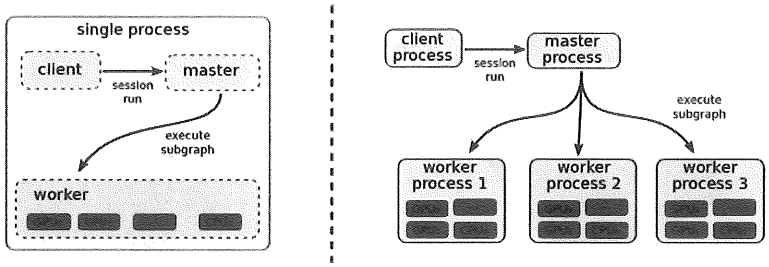


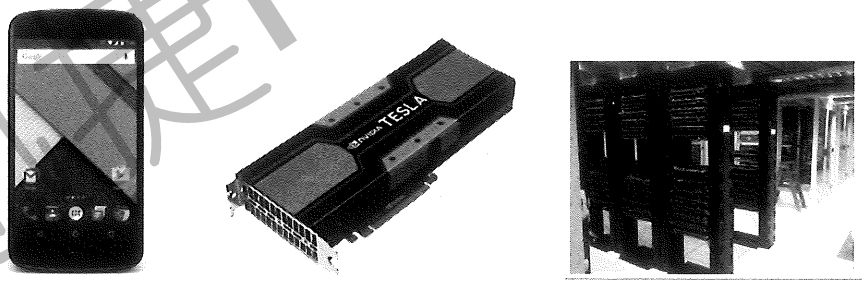
图 1-4 TensorFlow 单机版本和分布式版本的示例图

TensorFlow 中每一个 worker 可以管理多个设备，每一个设备的 name 包含硬件类别、编号、任务号（单机版本没有），示例如下。

单机模式：/job:localhost/device:cpu:0

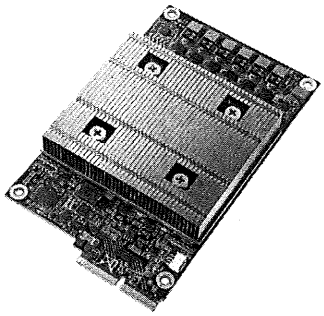
分布式模式：/job:worker/task:17/device:gpu:3

TensorFlow 为 CPU 和 GPU 提供了管理设备的对象接口，每一个对象负责分配、释放设备的内存，以及执行节点的运算核。TensorFlow 中的 tensor 是多维数组，数据类型支持 8 位至 64 位的 int，以及 IEEE 标准的 float、double 和复数型，同时还支持任意字符串。每一个设备有单独的 allocator 负责储存各种数据类型的 tensor，同时 tensor 的引用次数也会被记录，当引用数为 0 时，内存将被释放。如图 1-5 所示，TensorFlow 支持的设备包括 x86 架构 CPU、手机上的 ARM CPU、GPU、TPU（Tensor Processing Unit，Google 专门为大规模深度学习计算定制的芯片，但目前还没有公开发布的计划），例如 AlphaGo 在与李世石比赛时就大量使用了 TPU 集群的计算资源。

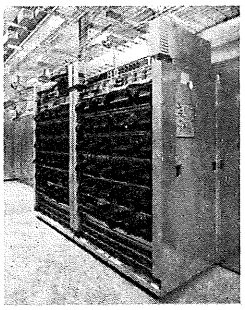


手机　　GPU　　GPU 集群

图 1-5 TensorFlow 支持的设备



TPU



TPU 集群

图 1-5　TensorFlow 支持的设备（续）

在只有一个硬件设备的情况下，计算图会按依赖关系被顺序执行。当一个节点的所有上游依赖都被执行完时（依赖数为 0），这个节点就会被加入 ready queue 以等待执行。同时，它下游所有节点的依赖数减 1，实际上这就是标准的计算拓扑序的方式。当有多个设备时，情况就变得复杂了，难点有二：

（1）每一个节点该让什么硬件设备执行。

（2）如何管理节点间的数据通信。

对第 1 个问题，TensorFlow 设计了一套为节点分配设备的策略。这个策略首先需要计算一个代价模型，这个代价模型估算每一个节点的输入、输出 tensor 的大小，以及所需要的计算时间。代价模型一部分由人工经验制定的启发式规则得到，另一部分则是由对一小部分数据进行实际运算而测量得到的。接下来，分配策略会模拟执行整个计算图，首先会从起点开始，按拓扑序执行。在模拟执行一个节点时，会把每一个能执行这个节点的设备都测试一遍，这个测试会考虑代价模型对这个节点的计算时间的估算，加上数据传到这个设备上所需要的通信时间，最后选择一个综合时间最短的设备作为这个节点的运算设备。可以看到这个策略是一个简单的贪婪策略，它不能确保找到全局最优解，但是可以用较快的速度找到一个不错的节点运算分配方案。同时除了运行时间，内存的最高使用峰值也会被考虑进来。目前 TensorFlow 的节点分配策略仍在不断研发、优化，将来可能使用一个强化学习（Reinforcement　Learning）的神经网络来辅助决策。此外，TensorFlow 还允许用户对节点的分配设置限制条件，比如“只给这个节点分配 GPU 类型的设备”，“只给这个节点分配/job:worker/task:17 上的设备”，“这个节点分配的设备必须和 variable13 一致”。对于这些限制条件，TensorFlow 会先计算每个节点可以使用的设备，再使用并查集

（union-find）算法找到必须使用同一个设备的节点。

我们再来看第 2 个问题，当给节点分配设备的方案被确定，整个计算图就会被划分为许多子图，使用同一个设备并且相邻的节点会被划分到同一个子图。然后计算图中从 x 到 y 的边，会被取代为一个发送端的发送节点（send node）、一个接收端的接收节点（receive node），以及从发送节点到接收节点的边，如图 1-6 所示。

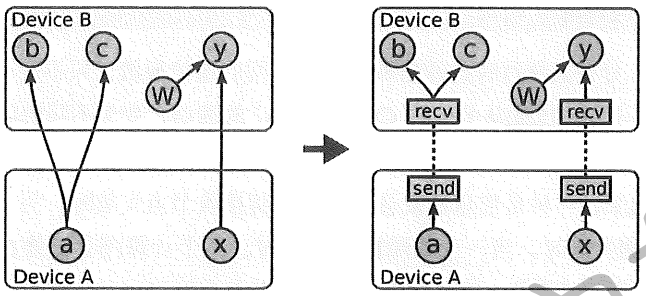


图 1-6 TensorFlow 的通信机制

这样就把数据通信的问题转变为发送节点和接收节点的实现问题，用户不需要为不同的硬件环境实现通信方法。同时，两个子图之间可能会有多个接收节点，如果这些接收节点接收的都是同一个 tensor，那么所有这些接收节点会被自动合并为一个，避免了数据的反复传输或者重复占用设备内存。总结一下，TensorFlow 的通信机制很优秀，发送节点和接收节点的设计简化了底层的通信模式，用户无须设计节点之间的通信流程，可以让同一套代码，自动扩展到不同硬件环境并处理复杂的通信流程。图 1-7 所示为 CPU 与 GPU 之间通信的流程。

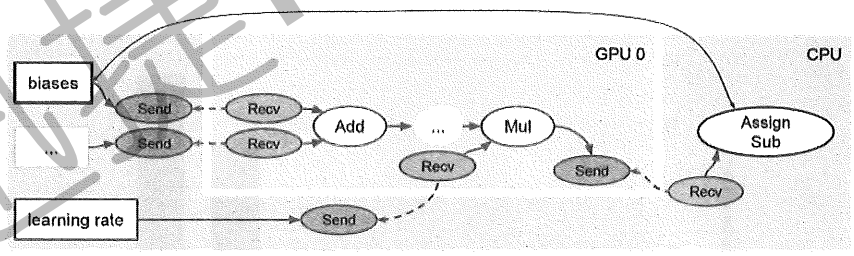


图 1-7 CPU 与 GPU 的通信过程

同时，从单机单设备的版本改造为单机多设备的版本也非常容易，下面的代码只添加了加粗的这一行，就实现了从一块 GPU 训练到多块 GPU 训练的改造。