银河麒麟服务器操作系统 V4

TensorFlow 软件适配手册

天津麒麟信息技术有限公司

2019 年 5 月

目 录

1 概述

1.1 系统概述

银河麒麟服务器操作系统主要面向军队综合电子信息系统、金融系统以及电力系统等国家关键行业的服务器应用领域，突出高安全性、高可用性、高效数据处理、虚拟化等关键技术优势，针对关键业务构建的丰富高效、安全可靠的功能特性，兼容适配联想、浪潮、华为、曙光等国内主流厂商的服务器整机产品，以及达梦、金仓、神通等主要国产数据库和中创、金蝶、东方通等国产中间件，满足虚拟化、云计算和大数据时代，服务器业务对操作系统在性能、安全性及可扩展性等方面的需求，是一款具有高安全、高可用、高可靠、高性能的自主可控服务器操作系统。

1.2 环境概述

服务器型号	长城信安擎天 DF720 服务器
CPU 类型	飞腾 2000+处理器
操作系统版本	Kylin-4.0.2-server-sp2-2000-19050910.Z1
内核版本	4.4.131
TensorFlow 版本	1.10

1.3 TensorFlow 软件概述

TensorFlow 是谷歌基于 DistBelief 进行研发的第二代人工智能学习系统，其命名来源于本身的运行原理。TensorFlow 最初由 Google 大脑小组（隶属于 Google 机器智能研究机构）的研究员和工程师们开发出来，用于机器学习和深度神经网络方面的研究，但这个系统的通用性使其也可广泛用于其他计算领域。TensorFlow 是一个采用数据流图（data flow graphs），用于数值计算的开源软件库。节点（Nodes）在图中表示数学操作，图中的线（edges）则表示在节点间相互联系的多维数据数组，即张量（tensor）。它灵活的架构让你可以在多种平台上展开计算，例如台式计算机中的一个或多个 CPU（或 GPU），服务器，移动设备等等。TensorFlow 支持 Python 和 C++，也允许在 CPU 和 GPU 上的计算分布，甚至支持使用 gRPC 进行水平扩展。任何人都可以使用 Tensorflow，学生、研究员、爱好者、极客、工程师、开发者、发明家、创业者等等都可以在 Apache 2.0 开源协议下使用 Tensorflow。

1.4 TensorFlow 特点

1. 高度的灵活性

TensorFlow 不是一个严格的“神经网络”库。只要你可以将你的计算表示为一个数据流图，你就可以使用 Tensorflow。你来构建图，描写驱动计算的内部循环。我们提供了有用的工具来帮助你组装“子图”（常用于神经网络），当然用户也可以自己在 Tensorflow 基础上写自己的“上层库”。定义顺手好用的新复合操作和写一个 python 函数一样容易，而且也不用担心性能损耗。当然万一你发现找不到想要的底层数据操作，你也可以自己写一点 c++代码来丰富底层的操作。

2. 真正的可移植性（Portability）

Tensorflow 在 CPU 和 GPU 上运行，比如说可以运行在台式机、服务器、手机移动设备等等。想要在没有特殊硬件的前提下，在你的笔记本上跑一下机器学习的新想法？Tensorflow 可以办到这点。准备将你的训练模型在多个 CPU 上规模化运算，又不想修改代码？Tensorflow 可以办到这点。想要将你的训练好的模型作为产品的一部分用到手机 app 里？Tensorflow 可以办到这点。你改变主意了，想要将你的模型作为云端服务运行在自己的服务器上，或者运行在 Docker 容器里？Tensorfow 也能办到。

3. 将科研和产品联系在一起

过去如果要将科研中的机器学习想法用到产品中，需要大量的代码重写工作。那样的日子一去不复返了！在 Google，科学家用 Tensorflow 尝试新的算法，产品团队则用 Tensorflow 来训练和使用计算模型，并直接提供给在线用户。使用 Tensorflow 可以让应用型研究者将想法迅速运用到产品中，也可以让学术性研究者更直接地彼此分享代码，从而提高科研产出率。

4. 自动求微分

基于梯度的机器学习算法会受益于 Tensorflow 自动求微分的能力。作为 Tensorflow 用户，你只需要定义预测模型的结构，将这个结构和目标函数（objective function）结合在一起，并添加数据，Tensorflow 将自动为你计算相关的微分导数。计算某个变量相对于其他变量的导数仅仅是通过扩展你的图来完成的，所以你能一直清楚看到究竟在发生什么。

5. 多语言支持

Tensorflow 有一个合理的 c++使用界面，也有一个易用的 python 使用界面来构建和执行你的 graphs。你可以直接写 python/c++程序，也可以用交互式的 ipython 界面来用 Tensorflow 尝试些想法，它可以帮你将笔记、代码、可视化等有条理地归置好。当然这仅仅是个起点——我们希望能鼓励你创造自己最喜欢的语言界面，比如 Go，Java，Lua，Javascript，或者是 R。

6. 性能最优化

比如说你有一个 32 个 CPU 内核、4 个 GPU 显卡的工作站，想要将你工作站的计算潜能全发挥出来？由于 Tensorflow 给予了线程、队列、异步操作等以最佳的支持，Tensorflow 让你可以将你手边硬件的计算潜能全部发挥出来。你可以自由地将 Tensorflow 图中的计算元素分配到不同设备上，Tensorflow 可以帮你管理好这些不同副本。

1.5 TensorFlow 原理介绍

TensorFlow 主要是由计算图、张量以及模型会话三个部分组成。

1. 计算图

在编写程序时，我们都是一步一步计算的，每计算完一步就可以得到一个执行结果。在 TensorFlow 中，首先需要构建一个计算图，然后按照计算图启动一个会话，在会话中完成变量赋值，计算，得到最终结果等操作。因此，可以说 TensorFlow 是一个按照计算图设计的逻辑进行计算的编程系统。

TensorFlow 的计算图可以分为两个部分：

- 构造部分，包含计算流图；
- 执行部分，通过 session 执行图中的计算。

构造部分又分为两部分：

- 创建源节点；
- 源节点输出传递给其他节点做运算。

2. 张量

在 TensorFlow 中，张量是对运算结果的引用，运算结果多以数组的形式存储，与 numpy 中数组不同的是张量还包含三个重要属性名字、维度、类型。张量的名字，是张量的唯一标识符，通过名字可以发现张量是如何计算出来的。比如“add:0”代表的是计算节点"add"的第一个输出结果。维度和类型与数组类似。

3. 模型会话

用来执行构造好的计算图，同时会话拥有和管理程序运行时的所有资源。当计算完成之后，需要通过关闭会话来帮助系统回收资源。

2 TensorFlow 软件适配

2.1 安装编译需要的依赖包

```
$ apt-get install python-dev
$ pip install setuptools six numpy wheel mock
$ pip uninstall enum
$ pip install keras_preprocessing
$ apt-get install python-enum34
```

2.2 下载指定分支源码

```
$ git clone -b r1.10 https://github.com/tensorflow/tensorflow
```

2.3 编译选项和以及平台相关修改

```
$ ./configure
```

Python 版本路径选择填写：/usr/bin/python3

其余选项均选择 N 或 n。

针对飞腾平台，需要对部分源码进行少量修改，具体如下:

a. tensorflow/BUILD 文件：

添加：

```
config_setting(
        name = "linux_aarch64",
        values = {"cpu": "aarch64"},
        visibility = ["//visibility:public"],
)
```

b. tensorflow/contrib/lite/kernels/internal/BUILD 文件：

注释掉所有 mfpu 相关的编译选项。

2.4 编译及安装

进行编译：

```
$ bazel build -c opt --copt="-funsafe-math-optimizations"
--copt="-ftree-vectorize" --copt="-fomit-frame-pointer" --verbose_failures
tensorflow/tools/pip_package:build_pip_package

$ bazel-bin/tensorflow/tools/pip_package/build_pip_package
/tmp/tensorflow_pkg
```

编译完成后，tensorflow 包位于：/tmp/tensorflow_pkg/下。

安装 TensorFlow：

```
$ pip install /opt/tensorflow_pkg/tensorflow-1.10.1-cp35-cp35m-linux_aarch64.whl
```

3 TensorFlow 软件功能验证

3.1 验证 TensorFlow 安装

在 python 交互式环境下验证 TensorFlow 是否正确安装。

```
root@sp2-arm64-hargrove-01:/opt# python3
Python 3.5.2 (default, Nov 19 2018, 06:26:54)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
>>> hello = tf.constant('Hello, TensorFlow!')
>>> sess = tf.Session()
2019-05-24 16:16:53.027218: I tensorflow/core/platform/profile_utils/cpu_utils.cc:94] CPU Frequency: 1797119999 Hz
2019-05-24 16:16:53.031794: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x940a2b0 executing computations on platform Host. Devices:
2019-05-24 16:16:53.032063: I tensorflow/compiler/xla/service/service.cc:175]   StreamExecutor device (0): <undefined>, <undefined>
>>> print(sess.run(hello))
b'Hello, TensorFlow!'
>>>
```

看到以上输出，说明 TensorFlow 已经安装完成。

3.2 尝试 convolutional model

通过 tensorflow 训练机器学习 hello world 模型：

```
$ git clone https://github.com/tensorflow/models.git
$ cd models/tutorials/image/mnist/
$ python convolutional.py
```

结果如下：

```
Step 0 (epoch 0.00), 28.0 ms
Minibatch loss: 8.334, learning rate: 0.010000
Minibatch error: 85.9%
Validation error: 84.6%
Step 100 (epoch 0.12), 1393.0 ms
Minibatch loss: 3.240, learning rate: 0.010000
Minibatch error: 3.1%
Validation error: 7.4%
Step 200 (epoch 0.23), 1379.4 ms
Minibatch loss: 3.351, learning rate: 0.010000
Minibatch error: 7.8%
Validation error: 4.5%
Step 300 (epoch 0.35), 1368.6 ms
Minibatch loss: 3.149, learning rate: 0.010000
Minibatch error: 4.7%
Validation error: 3.2%
Step 400 (epoch 0.47), 1363.2 ms
Minibatch loss: 3.228, learning rate: 0.010000
Minibatch error: 7.8%
Validation error: 2.8%
Step 500 (epoch 0.58), 1360.3 ms
Minibatch loss: 3.176, learning rate: 0.010000
Minibatch error: 6.2%
Validation error: 2.5%
Step 600 (epoch 0.70), 1355.2 ms
Minibatch loss: 3.131, learning rate: 0.010000
Minibatch error: 4.7%
Validation error: 2.0%
```