一步步在 **Raspberry Pi** 上构建 **TensorFlow**

Phodal Huang

September 8, 2017

目录

步骤 0：材料准备 3
步骤 1：安装依赖 3
步骤 2：安装内存驱动器用于作编译的交换空间 4
步骤 3：构建 Bazel 5
步骤 4：编译 TensorFlow 7
步骤 5：清理 9

原文链接：https://www.wandianshenme.com/play/build-tensorflow-on-raspberry-pi-step-by-step-g

本文介绍了如何一步步在 Raspberry Pi 及 Raspbian 上构建 Tensorflow 环境，并使用 bazel 进行编译。

步骤 0：材料准备

- Raspberry Pi 2 或 3 B 型
- 一个运行 Raspbian 的 SD 卡，拥有几 GB 的可用空间
  - 建议使用最低 16GB 的 SD 卡。
  - 这些说明可能适用于除 Raspbian 以外的 Linux 发行版
- Raspberry Pi 连接到互联网
- 可以安装为交换内存的 USB 存储驱动器（如果是闪存驱动器，请确保不关心驱动器）。超过 1GB 的都行
- 相当长的时间

这些说明是为 Raspberry Pi 3 B 型号制作的，运行 Raspbian 8.0（jessie）的 vanilla 版本。它似乎能在 Raspberry Pi 2 上工作，但有一些扭结正在制定中。如果这些说明适用于不同的发行版，请告诉我们!

以下是基本计划：构建一个适用于 RPi 的 Bazel 版本，并使用它来构建 TensorFlow。

- 安装基本依赖
- 安装 USB 存储器作为交换空间
- 构建 Bazel
- 编译 TensorFlow
- 收尾

步骤 1：安装依赖

使用 apt-get 更新软件源到最新

```
sudo apt-get update
```

下一步，安装我们所需要的依赖和工具。

Bazel 所需要的依赖：

```
sudo apt-get install pkg-config zip g++ zlib1g-dev unzip
```

TensorFlow 所需要的依赖：

```
# For Python 2.7
sudo apt-get install python-pip python-numpy swig python-devsudo pip
    install wheel

# For Python 3.3+
sudo apt-get install python3-pip python3-numpy swig python3-dev
sudo pip3 install wheel
```

为了能够利用某些优化标志，执行：

```
sudo apt-get install gcc-4.8 g++-4.8
sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.8 100
sudo update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-4.8 100
```

最后，为了清洁，制作一个目录，用于保存 Protobuf、Bazel 和 TensorFlow 库。

```
mkdir tf
cd tf
```

步骤 2：安装内存驱动器用于作编译的交换空间

为了成功构建 TensorFlow，您的 Raspberry Pi 需要更多的内存才能进行。幸运的是，这个过程非常简单——找到一个至少有 1GB 内存的 USB 存储驱动器。我使用的闪存驱动器，上面没有重要的数据。也就是说，我们只会在编译时将驱动器用作交换，因此此过程应该不会对相对较新的 USB 驱动器造成太大的损害。

首先，插入您的 USB 驱动器，并在 /dev/XXX 路径下找到相应的设备。

```
sudo blkid
```

例如，我的驱动器的路径是 `/dev/sda1`。

找到您的设备后，使用 `umount` 命令卸载它。

```
sudo umount /dev/XXX
```

然后将您的设备格式化为交换分区（swap）：

```
sudo mkswap /dev/XXX
```

如果上一个命令输出了一个字母数字的 UUID，请复制它。不然的话，只能再次运行 blkid 才能找到 UUID。复制与 /dev/XXX 关联的 UUID

```
sudo blkid
```

现在编辑您的 /etc/fstab 文件以注册您的交换文件系统。（我是使用 Vim 的人，但 Nano 是默认安装的）

```
sudo nano /etc/fstab
```

在另一行上输入以下信息。用 UUID 替换 X（不带引号）：

```
UUID=XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX none swap sw,pri=5 0 0
```

保存 /etc/fstab，退出文本编辑器，然后运行以下命令：

```
sudo swapon -a
```

如果您收到错误声明无法找到您的 UUID，请返回并编辑 /etc/fstab。用原始 /dev/XXX 信息替换 UUID=XXX。

```
sudo nano /etc/fstab
```

```
# Replace the UUID with /dev/XXX
/dev/XXX none swap sw,pri=5 0 0
```

好的！现在，你有交换空间了！不要忘了 /dev/XXX 的信息 - 您将需要它来安全地删除设备。

步骤 3：构建 Bazel

要构建 Bazel，我们需要下载一个包含分发存档的 zip 文件。我们现在就把它解压到一个名为 bazel 的新目录中：

```
wget
   https://github.com/bazelbuild/bazel/releases/download/0.4.5/bazel-0.4.5-dist.zip
unzip -d bazel bazel-0.4.5-dist.zip
```

完成下载和提取后，我们可以进入目录进行一些修改：

```
cd bazel
```

在构建 Bazel 之前，我们需要为此作业设置 javac 的最大堆大小，否则我们将得到一个 OutOfMemoryError 的错误。为了做到这一点，我们需要对 `bazel/scripts/bootstrap/compile.sh` 做一个小的修改。（感觉 @SangMan-LINUX 指出这一点..

```
nano scripts/bootstrap/compile.sh
```

往下移到第 117 行，您将看到以下代码块：

```
run "${JAVAC}" -classpath "${classpath}" -sourcepath "${sourcepath}" \
      -d "${output}/classes" -source "$JAVA_VERSION" -target
          "$JAVA_VERSION" \
      -encoding UTF-8 "@${paramfile}"
```

在此块的末尾，添加 -J-Xmx500M 标志，该标志将 Java 堆的最大大小设置为 500MB：

```
run "${JAVAC}" -classpath "${classpath}" -sourcepath "${sourcepath}" \
      -d "${output}/classes" -source "$JAVA_VERSION" -target
          "$JAVA_VERSION" \
      -encoding UTF-8 "@${paramfile}" -J-Xmx500M
```

最后，我们必须添加一个工具到 /cpp/cc_configure.bzl - 打开它进行编辑：

```
nano tools/cpp/cc_configure.bzl
```

将 return “arm” 放到 133 行（在 _get_cpu_value 函数的开头）：

```
...
"""Compute the cpu_value based on the OS name."""
return "arm"
...
```

现在我们可以建造 Bazel！注意：这也需要一些时间。

```
sudo ./compile.sh
```

当构建完成后，您将得到一个新的二进制位于 output/bazel。将其复制到 /usr/local/bin 目录。

```
sudo cp output/bazel /usr/local/bin/bazel
```

为了确保它正常工作，请在命令行上运行 bazel，并验证它是否打印帮助文本。注意：这可能需要 15-30 秒运行，所以要耐心等待！

```
bazel

Usage: bazel <command> <options> ...

```

```
Available commands:
  analyze-profile     Analyzes build profile data.
build               Builds the specified targets.  canonicalize-flags
    Canonicalizes a list of bazel options.
  clean               Removes output files and optionally stops the server.
  dump                Dumps the internal state of the bazel server process.
  fetch               Fetches external repositories that are prerequisites
    to the targets.
  help                Prints help for commands, or the index.
  info                Displays runtime info about the bazel server.
  mobile-install      Installs targets to mobile devices.
  query               Executes a dependency graph query.
  run                 Runs the specified target.
  shutdown            Stops the bazel server.
  test                Builds and runs the specified test targets.
  version             Prints version information for bazel.

Getting more help:
  bazel help <command>
                  Prints help and options for <command>.
  bazel help startup_options
                  Options for the JVM hosting bazel.
  bazel help target-syntax
                  Explains the syntax for specifying targets.
  bazel help info-keys
                  Displays a list of keys used by the info command.
```

跳出 bazel 目录，我们将进入下一步：

```
cd ..
```

步骤 4：编译 TensorFlow

首先，克隆 TensorFlow 的代码库，并移动到新创建的目录中。

```
git clone --recurse-submodules https://github.com/tensorflow/tensorflow
cd tensorflow
```

注意：如果你想建立一个特定的版本或提交 TensorFlow（而不是 master 的 HEAD），你现在应该执行 `git checkout`。

一旦在目录中，我们必须编写一个非常重要的、漂亮的单行代码。下一行将介绍 64 位程序实现（我们无法访问）到 32 位实现的所有文件和更改引用。整齐！

```
grep -Rl 'lib64' | xargs sed -i 's/lib64/lib/g'
```

接下来，我们需要删除 `tensorflow/core/platform/platform.h` 中的特定行。在您喜欢的文本编辑器中打开文件：

```
sudo nano tensorflow/core/platform/platform.h
```

现在，向下滚动到底部，删除包含 `#define IS_MOBILE_PLATFORM`（在第 48 行附近）的以下行：

```
#elif defined(__arm__)
#define PLATFORM_POSIX
...
#define IS_MOBILE_PLATFORM   <----- DELETE THIS LINE
```

这使得我们的 Raspberry Pi 设备能（其使用的中 ARM CPU）被识别为移动设备。

最后，我们必须调整协议以访问 Numeric JS 库 - 由于某些原因 Cloudflare 安全证书在 https 上无法正常工作。我们需要在 Bazel `WORKSPACE` 文件中解决这个问题：

```
sudo nano WORKSPACE
```

跑到 283 行，将 https 更改为 http：

```
http_file(
  name = "numericjs_numeric_min_js",
  url =
      "http://cdnjs.cloudflare.com/ajax/libs/numeric/1.2.6/numeric.min.js",
)
```

现在，让我们来配置构建：

```
./configure

Please specify the location of python. [Default is /usr/bin/python]:
    /usr/bin/python
Please specify optimization flags to use during compilation when bazel
    option "--config=opt" is specified [Default is -march=native]:
```

```
Do you wish to use jemalloc as the malloc implementation? [Y/n] Y
Do you wish to build TensorFlow with Google Cloud Platform support? [y/N]
    NDo you wish to build TensorFlow with Hadoop File System support? [y/N] N
Do you wish to build TensorFlow with the XLA just-in-time compiler
    (experimental)? [y/N] N
Please input the desired Python library path to use. Default is
    [/usr/local/lib/python2.7/dist-packages]
Do you wish to build TensorFlow with OpenCL support? [y/N] N
Do you wish to build TensorFlow with CUDA support? [y/N] N
```

注意：如果要为 Python 3 构建，请将 Python 的位置指定为 `/usr/bin/python3`，为 Python 库路径指定 `/usr/local/lib/python3.4/dist-packages`。

Bazel 现在会尝试清理。这需要很长时间（通常最终会出错），所以你可以发送一些键盘中断（CTRL-C）来跳过这个，并节省一些时间。

现在我们可以用它构建 TensorFlow！警告：这需要很长的时间。几个小时。

```
bazel build -c opt --copt="-mfpu=neon-vfpv4"
    --copt="-funsafe-math-optimizations" --copt="-ftree-vectorize"
    --copt="-fomit-frame-pointer" --local_resources 1024,1.0,1.0
    --verbose_failures tensorflow/tools/pip_package:build_pip_package
```

注意：我玩弄了，告诉 Bazel 使用 Raspberry Pi 中的所有四个核心，但这似乎使编译更容易完全锁定。这个过程需要很长时间，所以我坚持使用更可靠的选项。如果要加粗，请尝试使用 **–local_resources 1024,2.0,1.0** 或 **–local_resources 1024,4.0,1.0**

当你第二天早上醒来，完成编译后，你就在家里！使用内置的二进制文件，创建一个 Python wheel。

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

然后安装吧！

```
sudo pip install
    /tmp/tensorflow_pkg/tensorflow-1.1.0-cp27-none-linux_armv7l.whl
```

步骤 5：清理

完成之前，我们需要做最后一点的清理：将我们以前使用过的 USB 驱动器移除。

首先，关闭驱动器作为交换：

```
sudo swapoff /dev/XXX
```

最后，删除您在 /etc/fstab 中引用该设备的行：

```
sudo nano /etc/fstab
```

然后重新启动你的 Raspberry Pi。

Enjoy it!

原文链接：https://github.com/samjabrahams/tensorflow-on-raspberry-pi/blob/master/GUIDE.md

原文链接：https://www.wandianshenme.com/play/build-tensorflow-on-raspberry-pi-step-by-step-g