Jojodru

在Ubuntu 16.04上使用bazel交叉编译tensorflow

鸽了这么久，正式开工

Author: carbon

email: ecras_y@163.com

参考资料:

https://github.com/tensorflow/tensorflow

https://github.com/snipsco/tensorflow-build

年中6月份的时候被抽到AI项目组，有幸接触目前最火的深度学习神经网络，

从开始到现在，一直坚守在google的tensorflow

目前行业的趋势是在PC或者服务器集群上进行训练，然后将训练好的模型export到mobile设备上，

由mobile设备加载使用这些训练好的模型。

我早几年前做过很多嵌入式的项目，所以将tensorflow移植到mobile嵌入式平台上的任务就落到了我的身上，我也当仁不让，

一撸到底。

目前仅支持交叉编译tensorflow的c和c++模块。

python模块由于链接目标库问题，在mobile设备上也没有训练模型的需求，

兼具性能的考虑，暂时没做python这一块的交叉编译支持。感兴趣的同学可以自己试试看。

测试板子使用树莓派，

树莓派可以跑完整的Linux操作系统，如果需要在树莓派上运行基于python的tensorflow，

需要在树莓派上进行编译，但是可能需要编译几天时间

基于以后可能需要将项目export到其它的嵌入式系统上，需要考虑到通用性

Android的编译比较特殊，直接使用官方的预编译好的组件即可

这里只考虑非Android和iOS的嵌入式平台

tensorflow编译使用bazel进行自动化编译管理，涉及到:

1. 子模块网络下载

2. 预编译期代码生成(generate files)

3. 模块之间依赖判定

4. 代码编译

如果将工程更改为Makefile编译，工作量巨大，难以操作

最好的做法还是使用bazel，加入交叉编译支持

bazel交叉编译官方有一些指导文档，有很好的参考意义

> https://github.com/bazelbuild/bazel/wiki/Building-with-a-custom-toolchain

下载源码

2012年12月 (2)
2012年10月 (7)
2012年9月 (7)
2012年8月 (3)
2012年7月 (2)
2012年6月 (2)
2012年5月 (7)

最新评论

1. Re:在Ubuntu 16.04上使用bazel交叉编译tensorflow
为了在新版本绕过交叉编译引用python库的bug
需要将编译目标直接更改为tensorflow:libtensorflow_framework.so
编译测试通过
--Jojodru
2. Re:在Ubuntu 16.04上使用bazel交叉编译tensorflow
v.1.6.0版本编译前需要在configure之前需要export PYTHON_BIN_PATH=/path/to/your/python看git上的issue，这好像是新版本的一个bug，就算编.....
--Jojodru
3. Re:在Ubuntu 16.04上使用bazel交叉编译tensorflow
你好，我现在是使用改patch的方法编译tensorflow。但是编译到第3000个左右遇到了问题。提示很多的的

使用git命令下载tensorflow源码

```
git clone git@github.com:tensorflow/tensorflow.git
```

google在最近发布了tensorflow lite，用于支持mobile设备，

更小的编译目标文件，更好的性能

但是还没有export到最新的tag中，所以这里我们直接基于master分支来操作

```
git checkout master

git checkout -b cross-compile
```

文章最后加入对交叉编译tensorflow lite的说明

编译之前需要先修正数据对齐bug

x86和x64平台无对齐问题

但是在某些arm平台上，需要地址对齐，

否则程序会在运行时，访问内存奇数地址或者未对齐地址，导致crash

```
vim tensorflow/core/lib/gtl/inlined_vector.h +288
```

将 T* unused_aligner 替换为 uint64_t unused_aligner

强制为8字节对齐

准备交叉编译工具链

这里以树莓派为例

```
git clone https://github.com/raspberrypi/tools.git
```

函数没定义。都是tensorflow的函数，例如protobuff，oplist等相当多.....

--302792317

4. Re:在Ubuntu 16.04上使用bazel交叉编译tensorflow

@风过无痕sama不清楚你用的嵌入式平台和你编译的参数设置我在全志A33的R16和树莓派3上都测试通过tensorflow版本选的是1.4.1tensorflow lite直接用的master分支.....

--Jojodru

5. Re:在Ubuntu 16.04上使用bazel交叉编译tensorflow

@Jojodru那段gpu的代码我给注释了，没有用到gpu，这个溢出问题可能是因为我交叉编译的某些选项或者参数不正确导致的吗？ 因为我有些demo可以编译过去，比如说官网的示例demo：#includ.....

--风过无痕sama

阅读排行榜

1. 字符编码之UCS-2与Utf-8(10951)
2. gcc链接g++编译生成的静态库和动态库的makefile示例(6968)
3. libcurl的封装，支持同步异步请求，支持多线程下载，支持https(4555)

下载完成之后，将工具链路径添加到PATH中

脚本编写

目录结构

```
.
└── tensorflow
    ├── armv6-compiler
    │   ├── BUILD
    │   ├── CROSSTOOL
    │   └── cross_toolchain_target_armv6.BUILD
    ├── build_armv6.sh
    ├── build_tflite.sh

    └── WORKSPACE
```

1. 修改WORKSPACE文件，添加交叉编译工具链仓库描述

打开WORKSPACE文件，按照bazel的语法在文件末尾添加以下内容

```
new_local_repository(
        name='toolchain_target_armv6',
        path='/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-gnueabihf',
```

4. Mac OSX 读写usb composite设备(3035)
5. Linux Makefile多目录的编写(2935)

评论排行榜

1. Mac OSX 读写usb composite设备(12)
2. 在Ubuntu 16.04上使用bazel交叉编译tensorflow(9)
3. adb设备，根据serial获取vid pid(4)
4. adb and zlib for vs2008(2)
5. 不规则按钮，支持普通Button，Radio Button， Check Button(1)

推荐排行榜

1. 字符编码之UCS-2与Utf-8(1)
2. Mac OSX 读写usb composite设备(1)
3. 不规则按钮Button修正版(1)
4. libcurl的封装，支持同步异步请求，支持多线程下载，支持https(1)
5. libcurl 下载上传(1)

```
    build_file = 'armv6-compiler/cross_toolchain_target_armv6.BUILD'


)
```

参数说明:

交叉编译工具链别名: toolchain_target_armv6

交叉编译工具链路经: /path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-gnueabihf

交叉编译工具链描述文件: armv6-compiler/cross_toolchain_target_armv6.BUILD

2. 编写WORKSPACE仓库引用的交叉编译工具链的BUILD描述文件

```
mkdir armv6-compiler && cd armv6-compiler

touch cross_toolchain_target_armv6.BUILD


vi cross_toolchain_target_armv6.BUILD
```

具体语法规则参考详见

```
https://github.com/bazelbuild/bazel/wiki/Building-with-a-custom-toolchain
```

脚本内容如下:

```
1 package(default_visibility = ['//visibility:public'])
2
```

```
filegroup(
  name = 'gcc',
  srcs = [
    'bin/arm-linux-gnueabihf-gcc',
  ],
)

filegroup(
  name = 'ar',
  srcs = [
    'bin/arm-linux-gnueabihf-ar',
  ],
)

filegroup(
  name = 'ld',
  srcs = [
    'bin/arm-linux-gnueabihf-ld',
  ],
)

filegroup(
  name = 'nm',
  srcs = [
    'bin/arm-linux-gnueabihf-nm',
  ],
)

filegroup(
  name = 'objcopy',
  srcs = [
    'bin/arm-linux-gnueabihf-objcopy',
  ],
)
```

```
37
38 filegroup(
39   name = 'objdump',
40   srcs = [
41     'bin/arm-linux-gnueabihf-objdump',
42   ],
43 )
44
45 filegroup(
46   name = 'strip',
47   srcs = [
48     'bin/arm-linux-gnueabihf-strip',
49   ],
50 )
51
52 filegroup(
53   name = 'as',
54   srcs = [
55     'bin/arm-linux-gnueabihf-as',
56   ],
57 )
58
59 filegroup(
60   name = 'compiler_pieces',
61   srcs = glob([
62     'arm-linux-gnueabihf/**',
63     'libexec/**',
64     'lib/gcc/arm-linux-gnueabihf/**',
65     'include/**',
66   ]),
67 )
68
69 filegroup(
70   name = 'compiler_components',
```

```
71   srcs = [
72     ':gcc',
73     ':ar',
74     ':ld',
75     ':nm',
76     ':objcopy',
77     ':objdump',
78     ':strip',
79     ':as',
80   ],
81 )
```

3. 编写CROSSTOOL文件

CROSSTOOL文件负责描述交叉编译工具链，从编译选项到链接选项，非常繁杂

很多选项是放屁脱裤子，吃力不讨好

愚以为这bazel有点反人类阿

文件内容以下:

```
1 major_version: "local"
2 minor_version: ""
3 default_target_cpu: "armv6"
4
5 default_toolchain {
6   cpu: "armv6"
7   toolchain_identifier: "arm-linux-gnueabihf"
8 }
```

```
 9
10 default_toolchain {
11   cpu: "k8"
12   toolchain_identifier: "local"
13 }
14
15 toolchain {
16   abi_version: "gcc"
17   abi_libc_version: "glibc_2.23"
18   builtin_sysroot: ""
19   compiler: "compiler"
20   host_system_name: "raspberrypi"
21   needsPic: true
22   supports_gold_linker: false
23   supports_incremental_linker: false
24   supports_fission: false
25   supports_interface_shared_objects: false
26   supports_normalizing_ar: true
27   supports_start_end_lib: false
28   supports_thin_archives: true
29   target_libc: "glibc_2.23"
30   target_cpu: "armv6"
31   target_system_name: "raspberrypi"
32   toolchain_identifier: "arm-linux-gnueabihf"
33
34   tool_path { name: "ar" path: "/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-
gnueabihf/bin/arm-linux-gnueabihf-ar" }
35   tool_path { name: "compat-ld" path: "/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-
gnueabihf/bin/arm-linux-gnueabihf-ld" }
36   tool_path { name: "cpp" path: "/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-
gnueabihf/bin/arm-linux-gnueabihf-cpp" }
37   tool_path { name: "dwp" path: "/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-
gnueabihf/bin/arm-linux-gnueabihf-dwp" }
38   tool_path { name: "gcc" path: "/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-
```

```
gnueabihf/bin/arm-linux-gnueabihf-gcc" }
 39   tool_path { name: "gcov" path: "/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-
gnueabihf/bin/arm-linux-gnueabihf-gcov" }
 40   tool_path { name: "ld" path: "/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-
gnueabihf/bin/arm-linux-gnueabihf-ld" }
 41   tool_path { name: "nm" path: "/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-
gnueabihf/bin/arm-linux-gnueabihf-nm" }
 42   tool_path { name: "objcopy" path: "/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-
gnueabihf/bin/arm-linux-gnueabihf-objcopy" }
 43   objcopy_embed_flag: "-I"
 44   objcopy_embed_flag: "binary"
 45   tool_path { name: "objdump" path: "/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-
gnueabihf/bin/arm-linux-gnueabihf-objdump" }
 46   tool_path { name: "strip" path: "/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-
gnueabihf/bin/arm-linux-gnueabihf-strip" }
 47
 48   compiler_flag: "-nostdinc"
 49   compiler_flag: "-mfloat-abi=hard"
 50   compiler_flag: "-mfpu=neon-vfpv4"
 51   compiler_flag: "-funsafe-math-optimizations"
 52
 53   compiler_flag: "-isystem"
 54   compiler_flag: "/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-gnueabihf/include"
 55   compiler_flag: "-isystem"
 56   compiler_flag: "/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-gnueabihf/arm-linux-
gnueabihf/usr/include"
 57   compiler_flag: "-isystem"
 58   compiler_flag: "/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-gnueabihf/arm-linux-
gnueabihf/sysroot/usr/include"
 59
 60   compiler_flag: "-isystem"
 61   compiler_flag: "/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-gnueabihf/lib/gcc/arm-linux-
gnueabihf/4.9.3/include"
 62   compiler_flag: "-isystem"
```

```
 63   compiler_flag: "/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-gnueabihf/lib/gcc/arm-linux-gnueabihf/4.9.3/include-fixed"
 64
 65   compiler_flag: "-isystem"
 66   compiler_flag: "/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-gnueabihf/arm-linux-gnueabihf/include/c++/4.9.3"
 67   compiler_flag: "-isystem"
 68   compiler_flag: "/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-gnueabihf/arm-linux-gnueabihf/include/c++/4.9.3/arm-linux-gnueabihf"
 69
 70   cxx_flag: "-isystem"
 71   cxx_flag: "/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-gnueabihf/include"
 72   cxx_flag: "-isystem"
 73   cxx_flag: "/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-gnueabihf/arm-linux-gnueabihf/usr/include"
 74   cxx_flag: "-isystem"
 75   cxx_flag: "/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-gnueabihf/arm-linux-gnueabihf/sysroot/usr/include"
 76   cxx_flag: "-isystem"
 77   cxx_flag: "/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-gnueabihf/arm-linux-gnueabihf/include/c++/4.9.3"
 78   cxx_flag: "-isystem"
 79   cxx_flag: "/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-gnueabihf/arm-linux-gnueabihf/include/c++/4.9.3/arm-linux-gnueabihf"
 80   cxx_flag: "-std=c++11"
 81
 82   cxx_builtin_include_directory: "/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-gnueabihf/include"
 83   cxx_builtin_include_directory: "/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-gnueabihf/arm-linux-gnueabihf/usr/include"
 84   cxx_builtin_include_directory: "/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-gnueabihf/arm-linux-gnueabihf/sysroot/usr/include"
 85   cxx_builtin_include_directory: "/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-gnueabihf/lib/gcc/arm-linux-gnueabihf/4.9.3/include"
```

```
 86   cxx_builtin_include_directory: "/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-gnueabihf/lib/gcc/arm-linux-gnueabihf/4.9.3/include-fixed"
 87   cxx_builtin_include_directory: "/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-gnueabihf/arm-linux-gnueabihf/include/c++/4.9.3"
 88   cxx_builtin_include_directory: "/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-gnueabihf/arm-linux-gnueabihf/include/c++/4.9.3/arm-linux-gnueabihf"
 89
 90   linker_flag: "-lstdc++"
 91   linker_flag: "-L/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-gnueabihf/arm-linux-gnueabihf/lib"
 92   linker_flag: "-L/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-gnueabihf/arm-linux-gnueabihf/arm-linux-gnueabihf/lib"
 93   linker_flag: "-L/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-gnueabihf/arm-linux-gnueabihf/arm-linux-gnueabihf/sysroot/lib"
 94   linker_flag: "-L/path/to/toolchain/arm-bcm2708/arm-rpi-4.9.3-linux-gnueabihf/arm-linux-gnueabihf/arm-linux-gnueabihf/sysroot/usr/lib"
 95   linker_flag: "-Wl,--dynamic-linker=/lib/ld-linux-armhf.so.3"
 96
 97   # Anticipated future default.
 98   # This makes GCC and Clang do what we want when called through symlinks.
 99   unfiltered_cxx_flag: "-no-canonical-prefixes"
100   linker_flag: "-no-canonical-prefixes"
101
102   # Make C++ compilation deterministic. Use linkstamping instead of these
103   # compiler symbols.
104   unfiltered_cxx_flag: "-Wno-builtin-macro-redefined"
105   unfiltered_cxx_flag: "-D__DATE__=\"redacted\""
106   unfiltered_cxx_flag: "-D__TIMESTAMP__=\"redacted\""
107   unfiltered_cxx_flag: "-D__TIME__=\"redacted\""
108
109   # Security hardening on by default.
110   # Conservative choice; -D_FORTIFY_SOURCE=2 may be unsafe in some cases.
111   # We need to undef it before redefining it as some distributions now have
112   # it enabled by default.
```

```
113   compiler_flag: "-U_FORTIFY_SOURCE"
114   compiler_flag: "-fstack-protector"
115   compiler_flag: "-fPIE"
116   linker_flag: "-pie"
117   linker_flag: "-Wl,-z,relro,-z,now"
118
119   # Enable coloring even if there's no attached terminal. Bazel removes the
120   # escape sequences if --nocolor is specified.
121   compiler_flag: "-fdiagnostics-color=always"
122
123   # All warnings are enabled. Maybe enable -Werror as well?
124   compiler_flag: "-Wall"
125   # Enable a few more warnings that aren't part of -Wall.
126   compiler_flag: "-Wunused-but-set-parameter"
127   # But disable some that are problematic.
128   compiler_flag: "-Wno-free-nonheap-object" # has false positives
129
130   # Keep stack frames for debugging, even in opt mode.
131   compiler_flag: "-fno-omit-frame-pointer"
132
133   # Stamp the binary with a unique identifier.
134   linker_flag: "-Wl,--build-id=md5"
135   linker_flag: "-Wl,--hash-style=gnu"
136
137   compilation_mode_flags {
138     mode: DBG
139     # Enable debug symbols.
140     compiler_flag: "-g"
141   }
142   compilation_mode_flags {
143     mode: OPT
144
145     # No debug symbols.
146     # Maybe we should enable https://gcc.gnu.org/wiki/DebugFission for opt or
```

```
147     # even generally? However, that can't happen here, as it requires special
148     # handling in Bazel.
149     compiler_flag: "-g0"
150
151     # Conservative choice for -O
152     # -O3 can increase binary size and even slow down the resulting binaries.
153     # Profile first and / or use FDO if you need better performance than this.
154     compiler_flag: "-O3"
155
156     # Disable assertions
157     compiler_flag: "-DNDEBUG"
158
159     # Removal of unused code and data at link time (can this increase binary size in some cases?).
160     compiler_flag: "-ffunction-sections"
161     compiler_flag: "-fdata-sections"
162     linker_flag: "-Wl,--gc-sections"
163   }
164 }
165
166 toolchain {
167   toolchain_identifier: "local"
168   abi_libc_version: "local"
169   abi_version: "local"
170   builtin_sysroot: ""
171   compiler: "compiler"
172   compiler_flag: "-U_FORTIFY_SOURCE"
173   compiler_flag: "-D_FORTIFY_SOURCE=2"
174   compiler_flag: "-fstack-protector"
175   compiler_flag: "-Wall"
176   compiler_flag: "-Wl,-z,-relro,-z,now"
177   compiler_flag: "-B/usr/bin"
178   compiler_flag: "-B/usr/bin"
179   compiler_flag: "-Wunused-but-set-parameter"
180   compiler_flag: "-Wno-free-nonheap-object"
```

```
  compiler_flag: "-fno-omit-frame-pointer"
  compiler_flag: "-isystem"
  compiler_flag: "/usr/include"
  cxx_builtin_include_directory: "/usr/include/c++/5.4.0"
  cxx_builtin_include_directory: "/usr/include/c++/5"
  cxx_builtin_include_directory: "/usr/lib/gcc/x86_64-linux-gnu/5/include"
  cxx_builtin_include_directory: "/usr/include/x86_64-linux-gnu/c++/5.4.0"
  cxx_builtin_include_directory: "/usr/include/c++/5.4.0/backward"
  cxx_builtin_include_directory: "/usr/lib/gcc/x86_64-linux-gnu/5.4.0/include"
  cxx_builtin_include_directory: "/usr/local/include"
  cxx_builtin_include_directory: "/usr/lib/gcc/x86_64-linux-gnu/5.4.0/include-fixed"
  cxx_builtin_include_directory: "/usr/lib/gcc/x86_64-linux-gnu/5/include-fixed"
  cxx_builtin_include_directory: "/usr/include/x86_64-linux-gnu"
  cxx_builtin_include_directory: "/usr/include"
  cxx_flag: "-std=c++11"
  host_system_name: "local"
  linker_flag: "-lstdc++"
  linker_flag: "-lm"
  linker_flag: "-Wl,-no-as-needed"
  linker_flag: "-B/usr/bin"
  linker_flag: "-B/usr/bin"
  linker_flag: "-pass-exit-codes"
  needsPic: true
  objcopy_embed_flag: "-I"
  objcopy_embed_flag: "binary"
  supports_fission: false
  supports_gold_linker: false
  supports_incremental_linker: false
  supports_interface_shared_objects: false
  supports_normalizing_ar: false
  supports_start_end_lib: false
  supports_thin_archives: false
  target_cpu: "k8"
  target_libc: "local"
```

```
  target_system_name: "local"
  unfiltered_cxx_flag: "-fno-canonical-system-headers"
  unfiltered_cxx_flag: "-Wno-builtin-macro-redefined"
  unfiltered_cxx_flag: "-D__DATE__=\"redacted\""
  unfiltered_cxx_flag: "-D__TIMESTAMP__=\"redacted\""
  unfiltered_cxx_flag: "-D__TIME__=\"redacted\""
  tool_path {name: "ar" path: "/usr/bin/ar" }
  tool_path {name: "cpp" path: "/usr/bin/cpp" }
  tool_path {name: "dwp" path: "/usr/bin/dwp" }
  tool_path {name: "gcc" path: "/usr/bin/gcc" }
  tool_path {name: "gcov" path: "/usr/bin/gcov" }
  tool_path {name: "ld" path: "/usr/bin/ld" }
  tool_path {name: "nm" path: "/usr/bin/nm" }
  tool_path {name: "objcopy" path: "/usr/bin/objcopy" }
  tool_path {name: "objdump" path: "/usr/bin/objdump" }
  tool_path {name: "strip" path: "/usr/bin/strip" }

  compilation_mode_flags {
    mode: DBG
    compiler_flag: "-g"
  }
  compilation_mode_flags {
    mode: OPT
    compiler_flag: "-g0"
    compiler_flag: "-O3"
    compiler_flag: "-DNDEBUG"
    compiler_flag: "-ffunction-sections"
    compiler_flag: "-fdata-sections"
    linker_flag: "-Wl,--gc-sections"
  }
  linking_mode_flags { mode: DYNAMIC }
}
```

toolchain:

有两个描述

一个用于描述交叉编译工具链

一个用于描述本地编译环境

tensorflow编译时会同时用到本地编译环境和交叉编译环境

本地编译描述使用bazel build官网的描述即可

以下为交叉编译描述的一些事项:

abi_libc_version:

target_libc:

可以通过以下代码提前获取

```
printf("GNU libc version: %s\n", gnu_get_libc_version());
```

host_system_name:

树莓派3的值是armeabi-v6

Android的一般是armeabi-v7a

全志A33的也是armeabi-v7a

target_cpu:

树莓派3的值是armv6

Android的一般是armeabi-v7a

这个值需要和BUILD文件里面的描述一致

同时编译时--cpu来指定该用哪个交叉编译工具链

tensorflow里面有些依赖的模块BUILD文件描述里会用到这个值，稍后提到

tool_path:

涉及到的变量，替换为交叉编译工具链的绝对路径即可

compiler_flag:

编译选项，由于嵌入式平台，所以-nostdinc是必须的，否则会提示缺失头文件导致编译失败

-isystem 用于设置include的系统头文件目录

可以在交叉编译的工具链目录使用find命令查找

```
find /path/to/toolchain -name 'include*' -type d
```

一般找到的目录包括include和include-fixed，而且不会遗漏sysroot中的目录

其实这一步令人非常不解，因为工具链知道自己的头文件目录，根本不需指定

使用这个方法，导致可能会遗漏导致编译失败

cxx_flag:

和compiler_flag大同小异，需要额外指定c++的头文件目录

同时多了一个-std=c++11的值，用于开启c++11的特性

如果工具链不支持c++11，可以关掉

cxx_builtin_include_directory:

我觉得这个值和cxx_flag描述重复了

但是为了避免出错，依葫芦画瓢，把cxx_flag的值重新赋给它即可

linker_flag:

编译链接选项

主要是指定链接库

因为使用了c++11特性，第一个值就是-lstdc++

其它的值可以使用find命令查找

```
find /path/to/toolchain -name 'lib*' -type d
```

将查找结果赋值给-L即可

最后一个值是指定linker的加载器

典型值是-Wl,--dynamic-linker=/lib/ld-linux-armhf.so.3

可以在嵌入式设备上使用find命令找到具体的值

```
find / -name 'ld*' -type f
```

4. 编写BUILD文件

The BUILD file next to the CROSSTOOL is what ties everything together.

这个BUILD文件负责将以上的东西都串联起来，组成一个交叉编译工具链的完整描述

cc_toolchain_suite:

负责描述所有的编译工具链

里面的值必须和之前的cpu值定义相一致

bazel根据这个文件查找相对应的工具链

其它的选项并不复杂，不做赘述

脚本内容如下:

```
package(default_visibility = ["//visibility:public"])

cc_toolchain_suite(
    name = "toolchain",
    toolchains = {
        "armv6|compiler": ":cc-compiler-armv6",
        "k8|compiler": ":cc-compiler-local",
    },
)

filegroup(
    name = "empty",
    srcs = [],
)

filegroup(
    name = "arm_linux_all_files",
    srcs = [
        "@toolchain_target_armv6//:compiler_pieces",
    ],
```

```
)

cc_toolchain(
    name = "cc-compiler-local",
    all_files = ":empty",
    compiler_files = ":empty",
    cpu = "local",
    dwp_files = ":empty",
    dynamic_runtime_libs = [":empty"],
    linker_files = ":empty",
    objcopy_files = ":empty",
    static_runtime_libs = [":empty"],
    strip_files = ":empty",
    supports_param_files = 1,
)

cc_toolchain(
    name = "cc-compiler-armv6",
    all_files = ":arm_linux_all_files",
    compiler_files = ":arm_linux_all_files",
    cpu = "armv6",
    dwp_files = ":empty",
    dynamic_runtime_libs = [":empty"],
    linker_files = ":arm_linux_all_files",
    objcopy_files = "arm_linux_all_files",
    static_runtime_libs = [":empty"],
    strip_files = "arm_linux_all_files",
    supports_param_files = 1,
)
```

5. 编写build_armv6.sh文件

目的是避免每次重复输入命令

使用-march=armv6指定树莓派的架构

查看树莓派官网，树莓派3支持硬件浮点运算，并且支持到neon vfp v4

使用-mfloat-abi=hard -mfpu=neon-vfpv4打开浮点运算硬件支持

gcc需要-funsafe-math-optimizations额外打开浮点运算硬件支持

使用--crosstool_top=//armv6-compiler:toolchain --cpu=armv6 --config=opt

指定编译工具链为树莓派交叉编译工具链

```
#!/bin/bash
bazel build --copt="-fPIC" --copt="-march=armv6" \
    --copt="-mfloat-abi=hard" \
    --copt="-mfpu=neon-vfpv4" \
    --copt="-funsafe-math-optimizations" \
    --copt="-Wno-unused-function" \
    --copt="-Wno-sign-compare" \
    --copt="-ftree-vectorize" --copt="-fomit-frame-pointer" \
    --cxxopt="-Wno-maybe-uninitialized" \
    --cxxopt="-Wno-narrowing" \
    --cxxopt="-Wno-unused" \
    --cxxopt="-Wno-comment" \
    --cxxopt="-Wno-unused-function" \
    --cxxopt="-Wno-sign-compare" \
    --cxxopt="-funsafe-math-optimizations" \
    --linkopt="-mfloat-abi=hard" \
    --linkopt="-mfpu=neon-vfpv4" \
    --linkopt="-funsafe-math-optimizations" \
    --verbose_failures \
```

```
20      --crosstool_top=//armv6-compiler:toolchain --cpu=armv6 --config=opt \
21      tensorflow/examples/label_image/...
```

6. 在所有编译之前仅而且必须运行一遍configure

jmalloc可以选Yes，但是这里还是建议选No，使用C库的malloc

除了最后一个使用-march=来指定参数，其它一律选No

新版本的代码加入了一个Android交叉编译交互的配置

和本文不相干，选No

7. 编译

运行build_armv6.sh即可开始编译

编译开始时bazel会自动下载依赖组件

同时很快将因为错误停止编译

修正编译错误

1. 编译时bazel会从网络上下载依赖的子模块，需要网络支持，并且需要梯子FQ

否则会出现如下错误:

```
java.io.IOException: Error downloading
```

2. 为nsync添加交叉编译支持

缺乏配置描述，会在编译时出现类似以下错误

```
external/nsync/BUILD:401:13: Configurable attribute "copts" doesn't match this configuration
```

根据错误提示，打开nsync对应工程的BUILD文件

```
vi /path/to/bazel-cache/external/nsync/BUILD
```

参考android_arm的描述增加树莓派的配置描述

```
config_setting(
    name = "armv6",
    values = {"cpu": "armv6"},
)
```

本来想做个patch贴上来，太懒了，这里先将就

具体代码内容如下:

```
# -*- mode: python; -*-

# nsync is a C library that exports synchronization primitives, such as reader
# writer locks with conditional critical sections, designed to be open sourced
# in portable C.  See https://github.com/google/nsync
#
# See public/*.h for API.  When compiled with C++11 rather than C, it's in the
```

```
 8 # "nsync" name space.
 9 #
10 # BUILD file usage:
11 #   deps = "@nsync://nsync" for C version
12 #   deps = "@nsync://nsync_cpp" for C++11 version.
13 # The latter uses no OS-specific system calls or architecture-specific atomic
14 # operations.
15
16 package(default_visibility = ["//visibility:public"])
17
18 licenses(["notice"])  # Apache 2.0
19
20 exports_files(["LICENSE"])
21
22 # ------------------------------------------
23 # Parameters to the compilation:  compiler (e.g., for atomics), architecture
24 # (e.g., for load and store barrier behaviour), and OS.
25 # Bazel merges these into one, somewhat slippery, "cpu" string.
26 # Bazel uses a rather verbose mechanism for choosing which implementations
27 # are needed on given platforms; hence all the config_setting() rules below.
28
29 config_setting(
30     name = "gcc_linux_x86_32_1",
31     values = {"cpu": "piii"},
32 )
33
34 config_setting(
35     name = "gcc_linux_x86_64_1",
36     values = {"cpu": "k8"},
37 )
38
39 config_setting(
40     name = "gcc_linux_x86_64_2",
41     values = {"cpu": "haswell"},
```

```
)

config_setting(
    name = "gcc_linux_aarch64",
    values = {"cpu": "arm"},
)

config_setting(
    name = "gcc_linux_ppc64",
    values = {"cpu": "ppc"},
)

config_setting(
    name = "gcc_linux_s390x",
    values = {"cpu": "s390x"},
)

config_setting(
    name = "clang_macos_x86_64",
    values = {"cpu": "darwin"},
)

config_setting(
    name = "android_x86_32",
    values = {"cpu": "x86"},
)

config_setting(
    name = "android_x86_64",
    values = {"cpu": "x86_64"},
)

config_setting(
    name = "android_armeabi",
```

```
    values = {"cpu": "armeabi"},
)

config_setting(
    name = "android_arm",
    values = {"cpu": "armeabi-v7a"},
)

config_setting(
    name = "android_arm64",
    values = {"cpu": "arm64-v8a"},
)

config_setting(
    name = "msvc_windows_x86_64",
    values = {"cpu": "x64_windows"},
)

config_setting(
    name = "freebsd",
    values = {"cpu": "freebsd"},
)

config_setting(
    name = "ios_x86_64",
    values = {"cpu": "ios_x86_64"},
)

config_setting(
    name = "raspberry",
    values = {"cpu": "armv6"},
)

# ------------------------------------------
```

```
110 # Compilation options.
111
112 load(":bazel/pkg_path_name.bzl", "pkg_path_name")
113
114 # Compilation options that apply to both C++11 and C.
115 NSYNC_OPTS_GENERIC = select({
116     # Select the CPU architecture include directory.
117     # This select() has no real effect in the C++11 build, but satisfies a
118     # #include that would otherwise need a #if.
119     ":gcc_linux_x86_32_1": ["-I" + pkg_path_name() + "/platform/x86_32"],
120     ":gcc_linux_x86_64_1": ["-I" + pkg_path_name() + "/platform/x86_64"],
121     ":gcc_linux_x86_64_2": ["-I" + pkg_path_name() + "/platform/x86_64"],
122     ":gcc_linux_aarch64": ["-I" + pkg_path_name() + "/platform/aarch64"],
123     ":gcc_linux_ppc64": ["-I" + pkg_path_name() + "/platform/ppc64"],
124     ":gcc_linux_s390x": ["-I" + pkg_path_name() + "/platform/s390x"],
125     ":clang_macos_x86_64": ["-I" + pkg_path_name() + "/platform/x86_64"],
126     ":freebsd": ["-I" + pkg_path_name() + "/platform/x86_64"],
127     ":ios_x86_64": ["-I" + pkg_path_name() + "/platform/x86_64"],
128     ":android_x86_32": ["-I" + pkg_path_name() + "/platform/x86_32"],
129     ":android_x86_64": ["-I" + pkg_path_name() + "/platform/x86_64"],
130     ":android_armeabi": ["-I" + pkg_path_name() + "/platform/arm"],
131     ":android_arm": ["-I" + pkg_path_name() + "/platform/arm"],
132     ":raspberry": ["-I" + pkg_path_name() + "/platform/arm"],
133     ":android_arm64": ["-I" + pkg_path_name() + "/platform/aarch64"],
134     ":msvc_windows_x86_64": ["-I" + pkg_path_name() + "/platform/x86_64"],
135 }) + [
136     "-I" + pkg_path_name() + "/public",
137     "-I" + pkg_path_name() + "/internal",
138     "-I" + pkg_path_name() + "/platform/posix",
139 ] + select({
140     ":msvc_windows_x86_64": [
141     ],
142     ":freebsd": ["-pthread"],
143     "//conditions:default": [
```

```
        "-D_POSIX_C_SOURCE=200809L",
        "-pthread",
    ],
})

# Options for C build, rather then C++11 build.
NSYNC_OPTS = select({
    # Select the OS include directory.
    ":gcc_linux_x86_32_1": ["-I" + pkg_path_name() + "/platform/linux"],
    ":gcc_linux_x86_64_1": ["-I" + pkg_path_name() + "/platform/linux"],
    ":gcc_linux_x86_64_2": ["-I" + pkg_path_name() + "/platform/linux"],
    ":gcc_linux_aarch64": ["-I" + pkg_path_name() + "/platform/linux"],
    ":gcc_linux_ppc64": ["-I" + pkg_path_name() + "/platform/linux"],
    ":gcc_linux_s390x": ["-I" + pkg_path_name() + "/platform/linux"],
    ":clang_macos_x86_64": ["-I" + pkg_path_name() + "/platform/macos"],
    ":freebsd": ["-I" + pkg_path_name() + "/platform/freebsd"],
    ":ios_x86_64": ["-I" + pkg_path_name() + "/platform/macos"],
    ":android_x86_32": ["-I" + pkg_path_name() + "/platform/linux"],
    ":android_x86_64": ["-I" + pkg_path_name() + "/platform/linux"],
    ":android_armeabi": ["-I" + pkg_path_name() + "/platform/linux"],
    ":android_arm": ["-I" + pkg_path_name() + "/platform/linux"],
    ":raspberry": ["-I" + pkg_path_name() + "/platform/linux"],
    ":android_arm64": ["-I" + pkg_path_name() + "/platform/linux"],
    ":msvc_windows_x86_64": ["-I" + pkg_path_name() + "/platform/win32"],
    "//conditions:default": [],
}) + select({
    # Select the compiler include directory.
    ":gcc_linux_x86_32_1": ["-I" + pkg_path_name() + "/platform/gcc"],
    ":gcc_linux_x86_64_1": ["-I" + pkg_path_name() + "/platform/gcc"],
    ":gcc_linux_x86_64_2": ["-I" + pkg_path_name() + "/platform/gcc"],
    ":gcc_linux_aarch64": ["-I" + pkg_path_name() + "/platform/gcc"],
    ":gcc_linux_ppc64": ["-I" + pkg_path_name() + "/platform/gcc"],
    ":gcc_linux_s390x": ["-I" + pkg_path_name() + "/platform/gcc"],
    ":clang_macos_x86_64": ["-I" + pkg_path_name() + "/platform/clang"],
```

```
178     ":freebsd": ["-I" + pkg_path_name() + "/platform/clang"],
179     ":ios_x86_64": ["-I" + pkg_path_name() + "/platform/clang"],
180     ":android_x86_32": ["-I" + pkg_path_name() + "/platform/gcc"],
181     ":android_x86_64": ["-I" + pkg_path_name() + "/platform/gcc"],
182     ":android_armeabi": ["-I" + pkg_path_name() + "/platform/gcc"],
183     ":android_arm": ["-I" + pkg_path_name() + "/platform/gcc"],
184     ":raspberry": ["-I" + pkg_path_name() + "/platform/gcc"],
185     ":android_arm64": ["-I" + pkg_path_name() + "/platform/gcc"],
186     ":msvc_windows_x86_64": ["-I" + pkg_path_name() + "/platform/msvc"],
187 }) + select({
188     # Apple deprecated their atomics library, yet recent versions have no
189     # working version of stdatomic.h; so some recent versions need one, and
190     # other versions prefer the other.  For the moment, just ignore the
191     # depreaction.
192     ":clang_macos_x86_64": ["-Wno-deprecated-declarations"],
193     "//conditions:default": [],
194 }) + NSYNC_OPTS_GENERIC
195
196 # Options for C++11 build, rather then C build.
197 NSYNC_OPTS_CPP = select({
198     ":msvc_windows_x86_64": [
199         "/TP",
200     ],
201     "//conditions:default": [
202         "-x",
203         "c++",
204         "-std=c++11",
205     ],
206 }) + select({
207     # Some versions of MacOS (notably Sierra) require -D_DARWIN_C_SOURCE
208     # to include some standard C++11 headers, like <mutex>.
209     ":clang_macos_x86_64": ["-D_DARWIN_C_SOURCE"],
210     "//conditions:default": [],
211 }) + [
```

```
212     "-DNSYNC_ATOMIC_CPP11",
213     "-DNSYNC_USE_CPP11_TIMEPOINT",
214     "-I" + pkg_path_name() + "/platform/c++11",
215 ] + select({
216     # must follow the -I...platform/c++11
217     ":ios_x86_64": ["-I" + pkg_path_name() + "/platform/gcc_no_tls"],
218     ":msvc_windows_x86_64": [
219         "-I" + pkg_path_name() + "/platform/win32",
220         "-I" + pkg_path_name() + "/platform/msvc",
221     ],
222     "//conditions:default": ["-I" + pkg_path_name() + "/platform/gcc"],
223 }) + NSYNC_OPTS_GENERIC
224
225 # Link options (for tests) built in C (rather than C++11).
226 NSYNC_LINK_OPTS = select({
227     ":msvc_windows_x86_64": [],
228     "//conditions:default": ["-pthread"],
229 })
230
231 # Link options (for tests) built in C++11 (rather than C).
232 NSYNC_LINK_OPTS_CPP = select({
233     ":msvc_windows_x86_64": [],
234     "//conditions:default": ["-pthread"],
235 })
236
237 # ------------------------------------------------
238 # Header files the source may include.
239
240 # Internal library headers.
241 NSYNC_INTERNAL_HEADERS = [
242     "internal/common.h",
243     "internal/dll.h",
244     "internal/headers.h",
245     "internal/sem.h",
```

```
246     "internal/wait_internal.h",
247 ]
248
249 # Internal test headers.
250 NSYNC_TEST_HEADERS = NSYNC_INTERNAL_HEADERS + [
251     "testing/array.h",
252     "testing/atm_log.h",
253     "testing/closure.h",
254     "testing/heap.h",
255     "testing/smprintf.h",
256     "testing/testing.h",
257     "testing/time_extra.h",
258 ]
259
260 # Platform specific headers.
261 # This declares headers for all platforms, not just the one
262 # we're building for, to avoid a more complex build file.
263 NSYNC_INTERNAL_HEADERS_PLATFORM = [
264     "platform/aarch64/cputype.h",
265     "platform/alpha/cputype.h",
266     "platform/arm/cputype.h",
267     "platform/atomic_ind/atomic.h",
268     "platform/c++11/atomic.h",
269     "platform/c++11/platform.h",
270     "platform/c11/atomic.h",
271     "platform/clang/atomic.h",
272     "platform/clang/compiler.h",
273     "platform/cygwin/platform.h",
274     "platform/decc/compiler.h",
275     "platform/freebsd/platform.h",
276     "platform/gcc/atomic.h",
277     "platform/gcc/compiler.h",
278     "platform/gcc_new/atomic.h",
279     "platform/gcc_new_debug/atomic.h",
```

```
280     "platform/gcc_no_tls/compiler.h",
281     "platform/gcc_old/atomic.h",
282     "platform/lcc/compiler.h",
283     "platform/lcc/nsync_time_init.h",
284     "platform/linux/platform.h",
285     "platform/win32/atomic.h",
286     "platform/macos/platform_c++11_os.h",
287     "platform/msvc/compiler.h",
288     "platform/netbsd/atomic.h",
289     "platform/netbsd/platform.h",
290     "platform/openbsd/platform.h",
291     "platform/osf1/platform.h",
292     "platform/macos/atomic.h",
293     "platform/macos/platform.h",
294     "platform/pmax/cputype.h",
295     "platform/posix/cputype.h",
296     "platform/posix/nsync_time_init.h",
297     "platform/posix/platform_c++11_os.h",
298     "platform/ppc32/cputype.h",
299     "platform/ppc64/cputype.h",
300     "platform/s390x/cputype.h",
301     "platform/shark/cputype.h",
302     "platform/tcc/compiler.h",
303     "platform/win32/platform.h",
304     "platform/win32/platform_c++11_os.h",
305     "platform/x86_32/cputype.h",
306     "platform/x86_64/cputype.h",
307 ]
308
309 # ------------------------------------------
310 # The nsync library.
311
312 # Linux-specific library source.
313 NSYNC_SRC_LINUX = [
```

```
314     "platform/linux/src/nsync_semaphore_futex.c",
315     "platform/posix/src/per_thread_waiter.c",
316     "platform/posix/src/yield.c",
317     "platform/posix/src/time_rep.c",
318     "platform/posix/src/nsync_panic.c",
319 ]
320
321 # Android-specific library source.
322 NSYNC_SRC_ANDROID = [
323     "platform/posix/src/nsync_semaphore_sem_t.c",
324     "platform/posix/src/per_thread_waiter.c",
325     "platform/posix/src/yield.c",
326     "platform/posix/src/time_rep.c",
327     "platform/posix/src/nsync_panic.c",
328 ]
329
330 # MacOS-specific library source.
331 NSYNC_SRC_MACOS = [
332     "platform/posix/src/clock_gettime.c",
333     "platform/posix/src/nsync_semaphore_mutex.c",
334     "platform/posix/src/per_thread_waiter.c",
335     "platform/posix/src/yield.c",
336     "platform/posix/src/time_rep.c",
337     "platform/posix/src/nsync_panic.c",
338 ]
339
340 # Windows-specific library source.
341 NSYNC_SRC_WINDOWS = [
342     "platform/posix/src/nsync_panic.c",
343     "platform/posix/src/per_thread_waiter.c",
344     "platform/posix/src/time_rep.c",
345     "platform/posix/src/yield.c",
346     "platform/win32/src/clock_gettime.c",
347     "platform/win32/src/init_callback_win32.c",
```

```
348     "platform/win32/src/nanosleep.c",
349     "platform/win32/src/nsync_semaphore_win32.c",
350     "platform/win32/src/pthread_cond_timedwait_win32.c",
351     "platform/win32/src/pthread_key_win32.cc",
352 ]
353
354 # FreeBSD-specific library source.
355 NSYNC_SRC_FREEBSD = [
356     "platform/posix/src/nsync_semaphore_sem_t.c",
357     "platform/posix/src/per_thread_waiter.c",
358     "platform/posix/src/yield.c",
359     "platform/posix/src/time_rep.c",
360     "platform/posix/src/nsync_panic.c",
361 ]
362
363 # OS-specific library source.
364 NSYNC_SRC_PLATFORM = select({
365     ":gcc_linux_x86_32_1": NSYNC_SRC_LINUX,
366     ":gcc_linux_x86_64_1": NSYNC_SRC_LINUX,
367     ":gcc_linux_x86_64_2": NSYNC_SRC_LINUX,
368     ":gcc_linux_aarch64": NSYNC_SRC_LINUX,
369     ":gcc_linux_ppc64": NSYNC_SRC_LINUX,
370     ":gcc_linux_s390x": NSYNC_SRC_LINUX,
371     ":clang_macos_x86_64": NSYNC_SRC_MACOS,
372     ":freebsd": NSYNC_SRC_FREEBSD,
373     ":ios_x86_64": NSYNC_SRC_MACOS,
374     ":android_x86_32": NSYNC_SRC_ANDROID,
375     ":android_x86_64": NSYNC_SRC_ANDROID,
376     ":android_armeabi": NSYNC_SRC_ANDROID,
377     ":android_arm": NSYNC_SRC_ANDROID,
378     ":android_arm64": NSYNC_SRC_ANDROID,
379     ":msvc_windows_x86_64": NSYNC_SRC_WINDOWS,
380     ":raspberry": NSYNC_SRC_LINUX,
381 })
```

```
382
383 # C++11-specific (OS and architecture independent) library source.
384 NSYNC_SRC_PLATFORM_CPP = [
385     "platform/c++11/src/nsync_semaphore_mutex.cc",
386     "platform/c++11/src/time_rep_timespec.cc",
387     "platform/c++11/src/nsync_panic.cc",
388     "platform/c++11/src/yield.cc",
389 ] + select({
390     # MacOS and Android don't have working C++11 thread local storage.
391     ":clang_macos_x86_64": ["platform/posix/src/per_thread_waiter.c"],
392     ":android_x86_32": ["platform/posix/src/per_thread_waiter.c"],
393     ":android_x86_64": ["platform/posix/src/per_thread_waiter.c"],
394     ":android_armeabi": ["platform/posix/src/per_thread_waiter.c"],
395     ":android_arm": ["platform/posix/src/per_thread_waiter.c"],
396     ":android_arm64": ["platform/posix/src/per_thread_waiter.c"],
397     ":ios_x86_64": ["platform/posix/src/per_thread_waiter.c"],
398     ":msvc_windows_x86_64": [
399         "platform/win32/src/clock_gettime.c",
400         "platform/win32/src/pthread_key_win32.cc",
401         "platform/c++11/src/per_thread_waiter.cc",
402     ],
403     "//conditions:default": ["platform/c++11/src/per_thread_waiter.cc"],
404 })
405
406 # Generic library source.
407 NSYNC_SRC_GENERIC = [
408     "internal/common.c",
409     "internal/counter.c",
410     "internal/cv.c",
411     "internal/debug.c",
412     "internal/dll.c",
413     "internal/mu.c",
414     "internal/mu_wait.c",
415     "internal/note.c",
```

```
416     "internal/once.c",
417     "internal/sem_wait.c",
418     "internal/time_internal.c",
419     "internal/wait.c",
420 ]
421
422 # Generic library header files.
423 NSYNC_HDR_GENERIC = [
424     "public/nsync.h",
425     "public/nsync_atomic.h",
426     "public/nsync_counter.h",
427     "public/nsync_cpp.h",
428     "public/nsync_cv.h",
429     "public/nsync_debug.h",
430     "public/nsync_mu.h",
431     "public/nsync_mu_wait.h",
432     "public/nsync_note.h",
433     "public/nsync_once.h",
434     "public/nsync_time.h",
435     "public/nsync_time_internal.h",
436     "public/nsync_waiter.h",
437 ]
438
439 # The library compiled in C, rather than C++11.
440 cc_library(
441     name = "nsync",
442     srcs = NSYNC_SRC_GENERIC + NSYNC_SRC_PLATFORM,
443     hdrs = NSYNC_HDR_GENERIC,
444     copts = NSYNC_OPTS,
445     includes = ["public"],
446     textual_hdrs = NSYNC_INTERNAL_HEADERS + NSYNC_INTERNAL_HEADERS_PLATFORM,
447 )
448
449 # The library compiled in C++11, rather than C.
```

```
450 cc_library(
451     name = "nsync_cpp",
452     srcs = NSYNC_SRC_GENERIC + NSYNC_SRC_PLATFORM_CPP,
453     hdrs = NSYNC_HDR_GENERIC,
454     copts = NSYNC_OPTS_CPP,
455     includes = ["public"],
456     textual_hdrs = NSYNC_INTERNAL_HEADERS + NSYNC_INTERNAL_HEADERS_PLATFORM,
457 )
458
459 # nsync_headers provides just the header files for use in projects that need to
460 # build shared libraries for dynamic loading.  Bazel seems unable to cope
461 # otherwise.
462 cc_library(
463     name = "nsync_headers",
464     hdrs = glob(["public/*.h"]),
465     includes = ["public"],
466 )
467
468 # ----------------------------------------------
469 # Test code.
470
471 # Linux-specific test library source.
472 NSYNC_TEST_SRC_LINUX = [
473     "platform/posix/src/start_thread.c",
474 ]
475
476 # Android-specific test library source.
477 NSYNC_TEST_SRC_ANDROID = [
478     "platform/posix/src/start_thread.c",
479 ]
480
481 # MacOS-specific test library source.
482 NSYNC_TEST_SRC_MACOS = [
483     "platform/posix/src/start_thread.c",
```

```
484 ]
485
486 # Windows-specific test library source.
487 NSYNC_TEST_SRC_WINDOWS = [
488     "platform/win32/src/start_thread.c",
489 ]
490
491 # FreeBSD-specific test library source.
492 NSYNC_TEST_SRC_FREEBSD = [
493     "platform/posix/src/start_thread.c",
494 ]
495
496 # OS-specific test library source.
497 NSYNC_TEST_SRC_PLATFORM = select({
498     ":gcc_linux_x86_32_1": NSYNC_TEST_SRC_LINUX,
499     ":gcc_linux_x86_64_1": NSYNC_TEST_SRC_LINUX,
500     ":gcc_linux_x86_64_2": NSYNC_TEST_SRC_LINUX,
501     ":gcc_linux_aarch64": NSYNC_TEST_SRC_LINUX,
502     ":gcc_linux_ppc64": NSYNC_TEST_SRC_LINUX,
503     ":gcc_linux_s390x": NSYNC_TEST_SRC_LINUX,
504     ":clang_macos_x86_64": NSYNC_TEST_SRC_MACOS,
505     ":freebsd": NSYNC_TEST_SRC_FREEBSD,
506     ":ios_x86_64": NSYNC_TEST_SRC_MACOS,
507     ":android_x86_32": NSYNC_TEST_SRC_ANDROID,
508     ":android_x86_64": NSYNC_TEST_SRC_ANDROID,
509     ":android_armeabi": NSYNC_TEST_SRC_ANDROID,
510     ":android_arm": NSYNC_TEST_SRC_ANDROID,
511     ":android_arm64": NSYNC_TEST_SRC_ANDROID,
512     ":msvc_windows_x86_64": NSYNC_TEST_SRC_WINDOWS,
513     ":raspberry": NSYNC_TEST_SRC_LINUX,
514 })
515
516 # C++11-specific (OS and architecture independent) test library source.
517 NSYNC_TEST_SRC_PLATFORM_CPP = [
```

```
518     "platform/c++11/src/start_thread.cc",
519 ]
520
521 # Generic test library source.
522 NSYNC_TEST_SRC_GENERIC = [
523     "testing/array.c",
524     "testing/atm_log.c",
525     "testing/closure.c",
526     "testing/smprintf.c",
527     "testing/testing.c",
528     "testing/time_extra.c",
529 ]
530
531 # The test library compiled in C, rather than C++11.
532 cc_library(
533     name = "nsync_test_lib",
534     testonly = 1,
535     srcs = NSYNC_TEST_SRC_GENERIC + NSYNC_TEST_SRC_PLATFORM,
536     hdrs = ["testing/testing.h"],
537     copts = NSYNC_OPTS,
538     textual_hdrs = NSYNC_TEST_HEADERS + NSYNC_INTERNAL_HEADERS_PLATFORM,
539     deps = [":nsync"],
540 )
541
542 # The test library compiled in C++11, rather than C.
543 cc_library(
544     name = "nsync_test_lib_cpp",
545     testonly = 1,
546     srcs = NSYNC_TEST_SRC_GENERIC + NSYNC_TEST_SRC_PLATFORM_CPP,
547     hdrs = ["testing/testing.h"],
548     copts = NSYNC_OPTS_CPP,
549     textual_hdrs = NSYNC_TEST_HEADERS + NSYNC_INTERNAL_HEADERS_PLATFORM,
550     deps = [":nsync_cpp"],
551 )
```

```
552
553 # ----------------------------------------------
554 # The tests, compiled in C rather than C++11.
555
556 cc_test(
557     name = "counter_test",
558     size = "small",
559     srcs = ["testing/counter_test.c"],
560     copts = NSYNC_OPTS,
561     linkopts = NSYNC_LINK_OPTS,
562     deps = [
563         ":nsync",
564         ":nsync_test_lib",
565     ],
566 )
567
568 cc_test(
569     name = "cv_mu_timeout_stress_test",
570     size = "small",
571     srcs = ["testing/cv_mu_timeout_stress_test.c"],
572     copts = NSYNC_OPTS,
573     linkopts = NSYNC_LINK_OPTS,
574     deps = [
575         ":nsync",
576         ":nsync_test_lib",
577     ],
578 )
579
580 cc_test(
581     name = "cv_test",
582     size = "small",
583     srcs = ["testing/cv_test.c"],
584     copts = NSYNC_OPTS,
585     linkopts = NSYNC_LINK_OPTS,
```

```
586     deps = [
587         ":nsync",
588         ":nsync_test_lib",
589     ],
590 )
591
592 cc_test(
593     name = "cv_wait_example_test",
594     size = "small",
595     srcs = ["testing/cv_wait_example_test.c"],
596     copts = NSYNC_OPTS,
597     linkopts = NSYNC_LINK_OPTS,
598     deps = [
599         ":nsync",
600         ":nsync_test_lib",
601     ],
602 )
603
604 cc_test(
605     name = "dll_test",
606     size = "small",
607     srcs = ["testing/dll_test.c"],
608     copts = NSYNC_OPTS,
609     linkopts = NSYNC_LINK_OPTS,
610     deps = [
611         ":nsync",
612         ":nsync_test_lib",
613     ],
614 )
615
616 cc_test(
617     name = "mu_starvation_test",
618     size = "small",
619     srcs = ["testing/mu_starvation_test.c"],
```

```
620     copts = NSYNC_OPTS,
621     linkopts = NSYNC_LINK_OPTS,
622     deps = [
623         ":nsync",
624         ":nsync_test_lib",
625     ],
626 )
627
628 cc_test(
629     name = "mu_test",
630     size = "small",
631     srcs = ["testing/mu_test.c"],
632     copts = NSYNC_OPTS,
633     linkopts = NSYNC_LINK_OPTS,
634     deps = [
635         ":nsync",
636         ":nsync_test_lib",
637     ],
638 )
639
640 cc_test(
641     name = "mu_wait_example_test",
642     size = "small",
643     srcs = ["testing/mu_wait_example_test.c"],
644     copts = NSYNC_OPTS,
645     linkopts = NSYNC_LINK_OPTS,
646     deps = [
647         ":nsync",
648         ":nsync_test_lib",
649     ],
650 )
651
652 cc_test(
653     name = "mu_wait_test",
```

```
654     size = "small",
655     srcs = ["testing/mu_wait_test.c"],
656     copts = NSYNC_OPTS,
657     linkopts = NSYNC_LINK_OPTS,
658     deps = [
659         ":nsync",
660         ":nsync_test_lib",
661     ],
662 )
663
664 cc_test(
665     name = "note_test",
666     size = "small",
667     srcs = ["testing/note_test.c"],
668     copts = NSYNC_OPTS,
669     linkopts = NSYNC_LINK_OPTS,
670     deps = [
671         ":nsync",
672         ":nsync_test_lib",
673     ],
674 )
675
676 cc_test(
677     name = "once_test",
678     size = "small",
679     srcs = ["testing/once_test.c"],
680     copts = NSYNC_OPTS,
681     linkopts = NSYNC_LINK_OPTS,
682     deps = [
683         ":nsync",
684         ":nsync_test_lib",
685     ],
686 )
687
```

```
688 cc_test(
689     name = "pingpong_test",
690     size = "small",
691     srcs = ["testing/pingpong_test.c"],
692     copts = NSYNC_OPTS,
693     linkopts = NSYNC_LINK_OPTS,
694     deps = [
695         ":nsync",
696         ":nsync_test_lib",
697     ],
698 )
699
700 cc_test(
701     name = "wait_test",
702     size = "small",
703     srcs = ["testing/wait_test.c"],
704     copts = NSYNC_OPTS,
705     linkopts = NSYNC_LINK_OPTS,
706     deps = [
707         ":nsync",
708         ":nsync_test_lib",
709     ],
710 )
711
712 # --------------------------------------------
713 # The tests, compiled in C++11, rather than C.
714
715 cc_test(
716     name = "counter_cpp_test",
717     size = "small",
718     srcs = ["testing/counter_test.c"],
719     copts = NSYNC_OPTS_CPP,
720     linkopts = NSYNC_LINK_OPTS_CPP,
721     deps = [
```

```
722         ":nsync_cpp",
723         ":nsync_test_lib_cpp",
724     ],
725 )
726
727 cc_test(
728     name = "cv_mu_timeout_stress_cpp_test",
729     size = "small",
730     srcs = ["testing/cv_mu_timeout_stress_test.c"],
731     copts = NSYNC_OPTS_CPP,
732     linkopts = NSYNC_LINK_OPTS_CPP,
733     deps = [
734         ":nsync_cpp",
735         ":nsync_test_lib_cpp",
736     ],
737 )
738
739 cc_test(
740     name = "cv_cpp_test",
741     size = "small",
742     srcs = ["testing/cv_test.c"],
743     copts = NSYNC_OPTS_CPP,
744     linkopts = NSYNC_LINK_OPTS_CPP,
745     deps = [
746         ":nsync_cpp",
747         ":nsync_test_lib_cpp",
748     ],
749 )
750
751 cc_test(
752     name = "cv_wait_example_cpp_test",
753     size = "small",
754     srcs = ["testing/cv_wait_example_test.c"],
755     copts = NSYNC_OPTS_CPP,
```

```
    linkopts = NSYNC_LINK_OPTS_CPP,
    deps = [
        ":nsync_cpp",
        ":nsync_test_lib_cpp",
    ],
)

cc_test(
    name = "dll_cpp_test",
    size = "small",
    srcs = ["testing/dll_test.c"],
    copts = NSYNC_OPTS_CPP,
    linkopts = NSYNC_LINK_OPTS_CPP,
    deps = [
        ":nsync_cpp",
        ":nsync_test_lib_cpp",
    ],
)

cc_test(
    name = "mu_starvation_cpp_test",
    size = "small",
    srcs = ["testing/mu_starvation_test.c"],
    copts = NSYNC_OPTS_CPP,
    linkopts = NSYNC_LINK_OPTS_CPP,
    deps = [
        ":nsync_cpp",
        ":nsync_test_lib_cpp",
    ],
)

cc_test(
    name = "mu_cpp_test",
    size = "small",
```

```
790     srcs = ["testing/mu_test.c"],
791     copts = NSYNC_OPTS_CPP,
792     linkopts = NSYNC_LINK_OPTS_CPP,
793     deps = [
794         ":nsync_cpp",
795         ":nsync_test_lib_cpp",
796     ],
797 )
798
799 cc_test(
800     name = "mu_wait_example_cpp_test",
801     size = "small",
802     srcs = ["testing/mu_wait_example_test.c"],
803     copts = NSYNC_OPTS_CPP,
804     linkopts = NSYNC_LINK_OPTS_CPP,
805     deps = [
806         ":nsync_cpp",
807         ":nsync_test_lib_cpp",
808     ],
809 )
810
811 cc_test(
812     name = "mu_wait_cpp_test",
813     size = "small",
814     srcs = ["testing/mu_wait_test.c"],
815     copts = NSYNC_OPTS_CPP,
816     linkopts = NSYNC_LINK_OPTS_CPP,
817     deps = [
818         ":nsync_cpp",
819         ":nsync_test_lib_cpp",
820     ],
821 )
822
823 cc_test(
```

```
824     name = "note_cpp_test",
825     size = "small",
826     srcs = ["testing/note_test.c"],
827     copts = NSYNC_OPTS_CPP,
828     linkopts = NSYNC_LINK_OPTS_CPP,
829     deps = [
830         ":nsync_cpp",
831         ":nsync_test_lib_cpp",
832     ],
833 )
834
835 cc_test(
836     name = "once_cpp_test",
837     size = "small",
838     srcs = ["testing/once_test.c"],
839     copts = NSYNC_OPTS_CPP,
840     linkopts = NSYNC_LINK_OPTS_CPP,
841     deps = [
842         ":nsync_cpp",
843         ":nsync_test_lib_cpp",
844     ],
845 )
846
847 cc_test(
848     name = "pingpong_cpp_test",
849     size = "small",
850     srcs = ["testing/pingpong_test.c"],
851     copts = NSYNC_OPTS_CPP,
852     linkopts = NSYNC_LINK_OPTS_CPP,
853     deps = [
854         ":nsync_cpp",
855         ":nsync_test_lib_cpp",
856     ],
857 )
```

```
858
859 cc_test(
860     name = "wait_cpp_test",
861     size = "small",
862     srcs = ["testing/wait_test.c"],
863     copts = NSYNC_OPTS_CPP,
864     linkopts = NSYNC_LINK_OPTS_CPP,
865     deps = [
866         ":nsync_cpp",
867         ":nsync_test_lib_cpp",
868     ],
869 )
```

3. 再次运行build_armv6.sh编译即可

如果还出现错误，可以继续返回步骤2，修复错误

也可以给我邮件，但回复可能很慢

编译成功，目标文件生成目录:

```
bazel-bin/tensorflow/examples/label_image


bazel-bin/tensorflow/libtensorflow_framework.so
```

bazel并没有执行strip，如果有必要，需要手动执行strip

4. 刚发现最新的代码，使用-O2会触发树莓派交叉编译工具链崩溃

解决办法是，修改CROSSTOOL，更改为-O3

交叉编译tensorflow lite

1. Activate fpu support

```
vi tensorflow/contrib/lite/kernels/internal/BUILD
```

Add

```
config_setting(
        name = "armv6",
        values = { "cpu": "armv6", },


)
```

Add value

```
"armv6": [
        "-O2",
        "-mfpu=neon-vfpv4",
        "-mfloat-abi=hard",


    ],
```

to NEON_FLAGS_IF_APPLICABLE

and replace armv7a value with armv6

Add

```
":armv6": [
        ":neon_tensor_utils",
    ],
    ":armv7a": [
        ":neon_tensor_utils",


    ],
```

to cc_library tensor_utils select options

2. 编译编译脚本sh

```
#!/bin/bash

tflite_model="tensorflow/contrib/lite/kernels/internal:tensor_utils_test"
if [ $@"z" == "z" ]
then
    echo "$0 ${tflite_model}"
else
    tflite_model=$@
fi
bazel build --copt="-fPIC" --copt="-march=armv6" \
```

```
    --copt="-mfloat-abi=hard" \
    --copt="-mfpu=neon-vfpv4" \
    --copt="-funsafe-math-optimizations" \
    --copt="-Wno-unused-function" \
    --copt="-Wno-sign-compare" \
    --copt="-ftree-vectorize" \
    --copt="-fomit-frame-pointer" \
    --cxxopt='--std=c++11' \
    --cxxopt="-Wno-maybe-uninitialized" \
    --cxxopt="-Wno-narrowing" \
    --cxxopt="-Wno-unused" \
    --cxxopt="-Wno-comment" \
    --cxxopt="-Wno-unused-function" \
    --cxxopt="-Wno-sign-compare" \
    --cxxopt="-funsafe-math-optimizations" \
    --linkopt="-lstdc++" \
    --linkopt="-mfloat-abi=hard" \
    --linkopt="-mfpu=neon-vfpv4" \
    --linkopt="-funsafe-math-optimizations" \
    --verbose_failures \
    --strip=always \
    --crosstool_top=//armv6-compiler:toolchain --cpu=armv6 --config=opt \
    $@
```

3. 输出库和头文件

libbuiltin_ops.a libframework.a libneon_tensor_utils.a libquantization_util.a libtensor_utils.a

libcontext.a libfarmhash.a libgemm_support.a libportable_tensor_utils.a libstring_util.a

使用find命令和cp命令组合，将它们拷贝出来备用

头文件目录

tensorflow/contrib/lite

bazel-tensorflow/external/flatbuffers/include

4. Build toco

tensorflow lite加载的model文件需要预先使用toco转换格式

5. Configure and Build

configure with -march=native before building toco

```
bazel build tensorflow/contrib/lite/toco:toco
```

restore configure to cross compile after toco built

6. Run toco

具体用法参考tensorflow lite官网说明

```
bazel run --config=opt tensorflow/contrib/lite/toco:toco -- \
  --input_file=(pwd)/mobilenet_v1_1.0_224/frozen_graph.pb \
  --input_format=TENSORFLOW_GRAPHDEF  --output_format=TFLITE \
  --output_file=/tmp/mobilenet_v1_1.0_224.lite --inference_type=FLOAT \
  --input_type=FLOAT --input_arrays=input \
```

```
--output_arrays=MobilenetV1/Predictions/Reshape_1 --input_shapes=1,224,224,3
```

7. Done

enjoy

https://github.com/tensorflow/tensorflow

分类: C/C++,Linux

Jojodru
关注 - 0
粉丝 - 11
+加关注

« 上一篇：基于C++11实现的线程池

posted @ 2017-10-27 18:04 Jojodru 阅读(819) 评论(9) 编辑 收藏

评论列表

#1楼 2017-10-30 15:23 风过无痕sama

哥们你好，我最近也在做tensorflow的交叉编译工作，但一直没成功，能分享下你的过程和代码吗，非常感谢！

支持(0) 反对(0)

#2楼[楼主] 2017-12-15 14:05 Jojodru

@ 风过无痕sama

已更新

支持(0) 反对(0)

#3楼 2017-12-15 16:02 风过无痕sama

@ Jojodru

非常感谢大神！ 我按照大神的博客修正了数据对齐，在Arm下运行一些基本的tensorflow加减是没问题的，但是我按照 https://tebesu.github.io/posts/Training-a-TensorFlow-graph-in-C++-API 这篇博客中的例子运行时就会出现一个问题，涉及矩阵操作的时候，在X86环境下运行就没有问题，但是在arm平台下出现了下面的错误 ：

```
1970-01-16 01:22:04.392695: W tensorflow/core/framework/op_kernel.cc:1192] Resou
rce exhausted: OOM when allocating tensor with shape[536870912,268435456]
1970-01-16 01:22:04.392697: W tensorflow/core/framework/op_kernel.cc:1192] Resou
rce exhausted: OOM when allocating tensor with shape[268435456,134217728]
1970-01-16 01:22:04.392801: F tensorflow/examples/cpptrain/main.cc:26] Non-OK-st
atus: session->Run({}, {}, {"init_all_vars_op"}, nullptr) status: Resource exhau
sted: OOM when allocating tensor with shape[536870912,268435456]
[[Node: truncated_normal/TruncatedNormal = TruncatedNormal[T=DT_INT32,
dtype=DT_FLOAT, seed=0, seed2=0, _device="/job:localhost/replica:0/task:0/cpu:0"
](truncated_normal/shape)]]
Aborted
```

我初始化的矩阵大小：

```
Tensor x(DT_FLOAT, TensorShape({100, 32}));
Tensor y(DT_FLOAT, TensorShape({100, 8}));
```

我非常疑惑在Arm上运行矩阵变的很大，请问可能是什么原因？非常感谢大神指导！！！

支持(0) 反对(0)

#4楼[楼主] 2017-12-15 16:32 Jojodru

@ 风过无痕sama

我没做在arm上训练的测试

你的log看起来像是溢出了

我看了你发的例子，里面用到了gpu

但是arm上现在并不支持gpu，arm公司还在为mali的gpu努力
可能以后会支持gpu加速

支持(0) 反对(0)

#5楼 2017-12-15 16:41 风过无痕sama

@ Jojodru
那段gpu的代码我给注释了，没有用到gpu，这个溢出问题可能是因为我交叉编译的某些选项或者参数不正确导致的吗？ 因为我有些demo可以编译过去，比如说官网的示例demo：

```
#include "tensorflow/cc/client/client_session.h"
#include "tensorflow/cc/ops/standard_ops.h"
#include "tensorflow/core/framework/tensor.h"

int main() {
using namespace tensorflow;
using namespace tensorflow::ops;
Scope root = Scope::NewRootScope();
// Matrix A = [3 2; -1 0]
auto A = Const(root, { {3.f, 2.f}, {-1.f, 0.f} });
// Vector b = [3 5]
auto b = Const(root, { {3.f, 5.f} });
// v = Ab^T
auto v = MatMul(root.WithOpName("v"), A, b, MatMul::TransposeB(true));
std::vector<Tensor> outputs;
ClientSession session(root);
// Run and fetch v
TF_CHECK_OK(session.Run({v}, &outputs));
// Expect outputs[0] == [19; -3]
LOG(INFO) << outputs[0].matrix<float>();
return 0;
}
```

我刚刚跑了下label_image程序：

/tmp/wangcong # ./label_image

Call trace:

Out of memory: Kill process 3079 (label_image) score 30 or sacrifice child

Killed process 3079 (label_image) total-vm:147164kB, anon-rss:99092kB, file-rss:

19384kB

Call trace:

Killed

直接超出内存限制了，因为我是嵌入式新手，再次感谢大神百忙中抽出时间回答我的问题，感激不尽！

支持(0) 反对(0)

#6楼[楼主] 2017-12-15 16:48 Jojodru

@ 风过无痕sama

不清楚你用的嵌入式平台和你编译的参数设置

我在全志A33的R16和树莓派3上都测试通过

tensorflow版本选的是1.4.1

tensorflow lite直接用的master分支

支持(0) 反对(0)

#7楼 2018-01-09 11:33 302792317

你好，我现在是使用改patch的方法编译tensorflow。但是编译到第3000个左右遇到了问题。提示很多的的函数没定义。都是tensorflow的函数，例如protobuff，oplist等相当多的未定义。请问我是哪里的参数设置不正确呢。使用的是v1.4.1的版本，已经注释掉了platform.h中的

#if !defined(RASPBERRY_PI)

//#define IS_MOBILE_PLATFORM

#endif // !defined(RASPBERRY_PI)

支持(0) 反对(0)

#8楼[楼主] 2018-03-08 16:00 Jojodru

v.1.6.0版本编译前需要在configure之前需要

export PYTHON_BIN_PATH=/path/to/your/python

看git上的issue，
这好像是新版本的一个bug，
就算编译模块没有涉及到python
仍然需要设置一遍这个值

支持(0) 反对(0)

#9楼[楼主] 2018-03-08 16:21 Jojodru
为了在新版本绕过交叉编译引用python库的bug
需要将编译目标直接更改为
tensorflow:libtensorflow_framework.so
编译测试通过

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。