

LArSoft - Support #25571

need new version of Nvidia Triton (formerly TensorRT) client in ups/products to support Tensorflow 2.3.1

03/01/2021 11:46 AM - Michael Wang

Status:	Assigned	Start date:	03/01/2021
Priority:	Normal	Due date:	03/31/2021
Assignee:	Lynn Garren	% Done:	0%
Category:		Estimated time:	0.00 hour
Target version:		Spent time:	0.00 hour
Experiment:	-	Co-Assignees:	Patrick Gartung
Description			
<p>In moving from Tensorflow 1.12.x in larsoft to Tensorflow 2.3.1, the Triton servers (e.g. ailab01.fnal.gov, which "serves" out the ML models) will need to be upgraded to a new version to support Tensorflow 2.x. This new version uses a new v2 API (we have been using the v1 API so far) which will require larsoft to use a new client supporting the v2 API. The clients we are currently using are:</p> <pre>"trtis_clients" "v19_11a" "Linux64bit+3.10-2.17" "e19:prof" "" "trtis_clients" "v19_11b" "Linux64bit+3.10-2.17" "e19:prof" "" "trtis_clients" "v19_11c" "Linux64bit+3.10-2.17" "e19:prof" ""</pre> <p>I believe the actual version of these products is v1.8. We should upgrade these to v2.x. CMS is currently using v2.3. We may want to use the same version. Some details of the CMS version from Kevin Pedro:</p> <p>"CMS is using 2.3 right now: https://github.com/triton-inference-server/server/releases/tag/v2.3.0 the way we build it is: https://github.com/cms-sw/cmsdist/blob/IB/CMSSW_11_3_X/master/triton-inference-server.spec"</p>			

History

#1 - 03/08/2021 10:30 AM - Kyle Knoepfel

- Assignee set to Lynn Garren
- Status changed from New to Assigned

#2 - 03/29/2021 10:53 AM - Lynn Garren

- Co-Assignees Patrick Gartung added

The cmake configuration commans is failing with

```
CMake Error at /scratch/gartung/products/triton/v2_3_0/source/server-2.3.0/src/core/CMakeLists.txt:33 (protobu  
f_generate_cpp):  
  Unknown CMake command "protobuf_generate_cpp".
```

Looking in protobuf build area I have from tensorflow building tests I see that this macro is defined in an example file in the source directory

```
[gartung@scisoftbuild01 v3_12_3]$ grep -R protobuf_generate_cpp .  
...  
./source/examples/CMakeLists.txt:   protobuf_generate_cpp(${example}_PROTO_SRCS ${example}_PROTO_HDRS ${${exa  
mple}_PROTOS})  
./source/cmake/protobuf-module.cmake.in:   cmake_parse_arguments(protobuf_generate_cpp "" "EXPORT_MACRO" "" ${A  
RGN})  
./source/cmake/protobuf-module.cmake.in:   set(_proto_files "${protobuf_generate_cpp_UNPARSED_ARGUMENTS}")  
./source/cmake/protobuf-module.cmake.in:   protobuf_generate(${_append_arg} LANGUAGE cpp EXPORT_MACRO ${protobu  
f_generate_cpp_EXPORT_MACRO} OUT_VAR _outvar ${_import_arg} PROTOS ${_proto_files})
```

No file named protobuf-module.cmake appears in the install directory.
The first step is to understand how to generate this file in the protobuf build and install it.

#3 - 03/29/2021 12:15 PM - Patrick Gartung

Seems that we need to build protobuf with cmake as CMS does
https://github.com/cms-sw/cmsdist/blob/IB/CMSSW_11_3_X/master/protobuf.spec
to generate the config file that the triton server build expects.

#4 - 03/29/2021 04:26 PM - Patrick Gartung

To enable the client build to find the missing `protobuf_generate_cpp` macro I needed to set `-DTRITON_ENABLE_GRPC=ON`. This then added a `find_package(gRPC)` which breaks because gRPC is not installed. CMS builds standalone packages for gprc, protobuf and curl all of which the top level triton-server CMakeList builds as external projects when you run 'make client'

So the solution might be to use the top level triton-server CMakeList to configure the job and run 'make client'. However the build of gprc breaks when building its python bindings.

I think the solution might be to build gprc and libcurl as CMS does.

#5 - 03/29/2021 06:50 PM - Patrick Gartung

See the `cmake-build` branch in the `protobuf-ssi-build` repo. It is the only way to generate the file `protobuf/lib/cmake/protobuf/protobuf-module.cmake` which defines the function `protobuf_generate_cpp`

#6 - 03/29/2021 06:57 PM - Patrick Gartung

See the `gartung-build` branch in the `triton-ssi-build` repo. This builds the correct library which is `libgrpcclient.so` in v2.3.0. This library is linked against the static version of `libgrpc` so it does not have the same dependencies as the CMS version which links against dynamic libraries from gprc.

We can always add a gprc package if we want to do the dynamic linking instead.

#7 - 03/30/2021 05:37 PM - Patrick Gartung

A build script for gprc has been added. Gprc depends on protobuf and the build worked fine with the cmake build of protobuf. The `triton-inference-server` client depends on both protobuf and gprc along with c-ares libraries which are built by gprc.

The version of protobuf was bumped to `v3_12_3a` for the cmake build.

The cmake build of protobuf was tested with the tensorflow build and the libtorch build. Both succeeded.

All of the branches on the ssi-build repos used for testing the cmake build of protobuf have been merged to local master. I will push those to redmine once the changes are approved.

#8 - 04/01/2021 05:23 PM - Lynn Garren

Unfortunately, I am unable to build triton using the head of the master branch. I am not certain, but based on the log file, it appears that there is an attempt to download a cached protobuf. Then the cmake build tries to use our ups product and cmake reports a failure.

```
Collecting grpcio-tools
  Using cached grpcio_tools-1.36.1-cp38-cp38-manylinux2014_x86_64.whl (2.5 MB)
Collecting grpcio>=1.36.1
  Using cached grpcio-1.36.1-cp38-cp38-manylinux2014_x86_64.whl (4.1 MB)
Collecting protobuf<4.0dev,>=3.5.0.post1
  Using cached protobuf-3.15.6-cp38-cp38-manylinux1_x86_64.whl (1.0 MB)

+ cd /scratch/garren/products/triton/v2_3_0/build/Linux64bit+3.10-2.17-e20-prof
+ cmake /scratch/garren/products/triton/v2_3_0/source/server-2.3.0/build/client -DCMAKE_INSTALL_PREFIX=/scratch/garren/products/triton/v2_3_0/Linux64bit+3.10-2.17-e20-prof -DCMAKE_INSTALL_LIBDIR=lib -DCMAKE_BUILD_TYPE=MinSizeRel -DCMAKE_C_COMPILER=gcc -DCMAKE_CXX_COMPILER=g++ '-DCMAKE_CXX_FLAGS=-Wno-deprecated-declarations -Wno-error=deprecated-declarations' -DTRITON_CLIENT_SKIP_EXAMPLES=ON -DTRITON_ENABLE_HTTP=OFF -DTRITON_ENABLE_GRPC=ON -DTRITON_ENABLE_GPU=OFF -DProtobuf_DIR=/scratch/garren/products/protobuf/v3_12_3a/Linux64bit+3.10-2.17-e20/lib/cmake/protobuf -DTRITON_VERSION=2.3.0

CMake Error at /scratch/products/cmake/v3_20_0/Linux64bit+3.10-2.17/share/cmake-3.20/Modules/FindPackageHandleStandardArgs.cmake:230 (message):
  Could NOT find Protobuf (missing: Protobuf_PROTOC_EXECUTABLE) (found version "3.12.3.0")
Call Stack (most recent call first):
  /scratch/products/cmake/v3_20_0/Linux64bit+3.10-2.17/share/cmake-3.20/Modules/FindPackageHandleStandardArgs.cmake:594 (_FPHSA_FAILURE_MESSAGE)
  /scratch/garren/products/protobuf/v3_12_3a/Linux64bit+3.10-2.17-e20/lib/cmake/protobuf/protobuf-module.cmake:162 (FIND_PACKAGE_HANDLE_STANDARD_ARGS)
  /scratch/garren/products/protobuf/v3_12_3a/Linux64bit+3.10-2.17-e20/lib/cmake/protobuf/protobuf-config.cmake:149 (include)
  CMakeLists.txt:81 (find_package)
```

#9 - 04/01/2021 08:32 PM - Patrick Gartung

I am seeing the same thing. I think the last change I made to the protobuf build script may have caused the problem. I did not retry the triton build after I rebuilt protobuf.

#10 - 04/01/2021 09:03 PM - Patrick Gartung

So the build where CMAKE_BUILD_TYPE was accidentally not set was the one that worked. The CMS buildscript does not set CMAKE_BUILD_TYPE either.

This generated protobuf/lib/cmake/protobuf/protobuf-targets-noconfig.cmake. The generated protobuf/lib/cmake/protobuf/protobuf-module.cmake looks for the protoc executable in one of three variables IMPORTED_LOCATION_{RELEASE,DEBUG,NOCONFIG} (see below). The CMAKE_RELEASE_TYPE was set to MinSizeRel for which there was not corresponding IMPORTED_LOCATION variable.

The head of the master branch for protobuf can now be used to create a build that the triton build script will work with.

```
# Set the protoc Executable
get_target_property(Protobuf_PROTOC_EXECUTABLE protobuf::protoc
  IMPORTED_LOCATION_RELEASE)
if(NOT EXISTS "${Protobuf_PROTOC_EXECUTABLE}")
  get_target_property(Protobuf_PROTOC_EXECUTABLE protobuf::protoc
    IMPORTED_LOCATION_DEBUG)
endif()
if(NOT EXISTS "${Protobuf_PROTOC_EXECUTABLE}")
  get_target_property(Protobuf_PROTOC_EXECUTABLE protobuf::protoc
    IMPORTED_LOCATION_NOCONFIG)
endif()
```