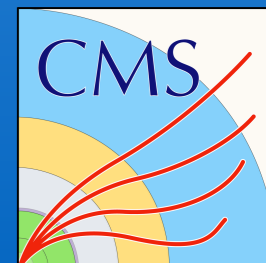


TensorFlow 2.0 with GPU Support in CMSSW

Marcel Rieger

7th Patatrack Hackathon

3.10.2019





- **Project objective**

- Provide **all** the tools to perform fast inference of TF (2.0) models on GPUs in CMSSW

- **Tasks to accomplish**

 Build TF 2.0 and all requirements in [cmsdist](#)


- ▷ **Done:** updated Bazel, Protobuf, CUDA, added CUPTI & cuDNN,

- ▷ **Done:** fix CUBLAS, added Nvidia NCCL, added and compiled **TF 2.0** (C++ & Py)

 Deal with Bazel's fear of / inability to handle non-standard environments

- ▷ **Done:** patched some java source files to respect CMSSW environment

 Adapt customizations in CMSSW (mostly interference due to different threading approach)

 Add a “resource scheduler”, assigning resources (CPU/GPU*/GPU RAM) to CMSSW modules

 SOON Handle GPU memory growth on demand and simultaneously prevent too large batch sizes

 SOON Add PR to cmsdist repository

yesterday--

yesterday

today++



- “Scheduling” of resources should happen in 3 steps:
 1. Modules claim resources in their constructors (or before, i.e., in some static method?)
 - ▷ Number of GPUs **or** even preferred GPU device numbers
 - ▷ Amount of total memory **or** memory per GPU
 - ▷ Whether or not the module can fallback to CPUs
 - ▷ Priority and other soft information to be used in step 2
 2. Scheduler evaluates all claims and divides resources according to policies (🕒)
 3. Modules receive the response encoded in a simple struct (where? constructor?)
 - ▷ “devices” (list): GPU device numbers **or** empty list (== CPU)
 - ▷ “memory” (list): Memory assigned per GPU **or** empty list

```
DeviceOpts opts = getMyDevice(1, 2048.);  
int gpuNum = opts.devices[0];  
string device = "/GPU:" + std::to_string(gpuNum);  
  
tensorflow::GraphDef* graphDef = tensorflow::loadGraphDef("somegraph.pb");  
tensorflow::Session* session = tensorflow::createSession(graphDef, device);
```