Enhancing PyTorch Performance on Frontier with the RCCL OFI-Plugin

Mengshiou Wu, Mark Stock
April 17th, 2024
Introduction

- RCCL (ROCm Communication Collectives Library) is used as the underlying communication library for major machine learning frameworks such as PyTorch or TensorFlow/Horovod.
- The default setting of RCCL uses TCP/IP for inter-node communications and does not utilize Slingshot/Libfabric.
- The Slurm scripts of OLCF’s MLDL training usually include a few lines that enables RCCL to use a plugin to improve performance of MLDL codes:

```
#export NCCL_DEBUG=INFO
export LD_LIBRARY_PATH=${PATH TO THE PLUGIN}/libs/
#export FI_LOG_LEVEL=info
export NCCL_NET_GDR_LEVEL=3
```
• The focus of this talk is on the communication libraries used by PyTorch.
  • For PyTorch-DDP, PyTorch-FSDP, or introduction of different model parallelisms, OLCF has an excellent training talk (AI for Science at Scale – Part 2: [https://github.com/olcf/ai-training-series](https://github.com/olcf/ai-training-series)).

• **Part I:**
  • The basic of RCCL, RCCL tester, and variables that may impact performance.
  • The basic of the OFI-Plugin.

• **Part II:**
  • PyTorch examples with and without using the Plugin.
  • Profiling and analyzing performance data using PyTorch Profiler.
Part I: RCCL, RCCL Tester and the Plugin
RCCL Basics

- MLDL User Code (python, C++)
- Framework (PyTorch, Tensorflow, Horovod)
- Communication Library (MPI, RCCL, Gloo)
- Transport layer (libfabric, IB, TCP/IP)
- Other Network-y stuff (Socket)
- Physical layers (xGMI, PCIe, Cat5, fiber)
RCCL Basics

- RCCL versions & library location
  - `/opt/rocm-${version}/rccl/lib` or `/opt/rocm-${version}/rccl/include` ➔ before 6.0.0
  - `/opt/rocm-${version}/lib` or `/opt/rocm-${version}/include` ➔ after 6.0.0
  - **Load the rocm module should be all you need to do.**

- Build RCCL from source
  - If plan to build RCCL from source: [https://github.com/ROCm/rccl](https://github.com/ROCm/rccl)

- Most useful environment variables
  - `NCCL_NET_GDR_LEVEL=3` or `NCCL_NET_GDR_LEVEL=PHB` - This variable enables RDMA between GPUs.
  - `NCCL_ALGO` - Specify the algorithms for collective.
  - `NCCL_CROSS_NIC=1` - On large systems, this NCCL setting has been found to improve performance.
  - `NCCL_DEBUG` - For verification and debugging.

- RCCL is a port of NCCL and thus RCCL keeps some notations.
RCCL Basics

- Default intra-node communications
  - Utilize xGMI for collectives.

- Default inter-node communications on Frontier
  - TCP/IP: needs to use handshaking protocol, persistent connections.

- Collective Algorithms
  - Tree or Ring, depending on number of nodes/ranks and topology.
  - It uses complex tuning strategies rather than a simple threshold of message size.
RCCL Performance

- https://github.com/ROCm/rccl-tests

  The tester uses MPI only for summarized numbers; it does NOT use MPI for collectives.

  The tester can be used to test most collectives; we will focus on all_reduce here.
RCCL Tester Build Script

#!/usr/bin/bash -i
rocm_version=5.7.1

export LD_LIBRARY_PATH="${CRAY_LD_LIBRARY_PATH}:${LD_LIBRARY_PATH}"
git clone --recursive https://github.com/ROCmSoftwarePlatform/rccl-tests
cd rccl-tests
grep -RiIl 'opt\</rocm' | xargs sed -i "s/opt\</rocm/opt\</rocm-"rocm_version"/g"

module load libtool
module swap PrgEnv-crAy PrgEnv-gnu
module load rocM/$rocm_version
module load craype-accel-amd-gfx90a
module load gcc/12.2.0
module load cray-mpich/8.1.27

echo $MPICh_DIR
make MPI=1 MPI_HOME=$MPICh_DIR ROCm_HOME=/opt/rocm-"rocm_version"
export LD_LIBRARY_PATH=/opt/rocm-"rocm_version"/rccl/lib:${LD_LIBRARY_PATH}

ldd ./build/all_reduce_perf

Check if the correct version of ROCm is selected.
RCCL Tester Definitions

• https://github.com/ROCm/rccl-tests/blob/develop/doc/PERFORMANCE.md

• **Algorithm bandwidth**
  - Algorithm bandwidth is using the most commonly used formula for bandwidth: \( \frac{\text{size (S)}}{\text{time (t)}} \). It is useful to compute how much time any large operation would take by simply dividing the size of the operation by the algorithm bandwidth.
  - \( \text{algbw} = \frac{S}{t} \)

• **Bus bandwidth**
  - This number is obtained **applying a multiple to the algorithm bandwidth to reflect the speed of the inter-GPU communication**. Using this bus bandwidth, we can compare it with the hardware peak bandwidth, independently of the number of ranks used.
    - AllReduce: \( \text{algbw} \times \frac{2(n-1)}{n} \)
    - ReduceScatter: \( \text{algbw} \times \frac{n-1}{n} \)
    - AllGather: \( \text{algbw} \times \frac{n-1}{n} \)
    - Broadcast: \( \frac{S}{t} \)
    - Reduce: \( \frac{S}{t} \)

• **In-Place**: use the same buffer for input/output.
• **Out-of-place**: use different buffers for input/output.
RCCL Test Data - Single Node

srun -n8 -N1 --tasks-per-node=8 --gpu-bind=closest all_reduce_perf -b 64K -e 4G -f 2 -g 1
(run setting: test message size from 64 K to 4G, double message size, use one GPU per rank)

<table>
<thead>
<tr>
<th>#</th>
<th>size</th>
<th>count</th>
<th>type</th>
<th>redop</th>
<th>root</th>
<th>time (us)</th>
<th>algbw (GB/s)</th>
<th>busbw (GB/s)</th>
<th>#wrong</th>
<th>time (us)</th>
<th>algbw (GB/s)</th>
<th>busbw (GB/s)</th>
<th>#wrong</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>65536</td>
<td>16384</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>37.54</td>
<td>1.75</td>
<td>3.06</td>
<td>0</td>
<td>34.53</td>
<td>1.90</td>
<td>3.32</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>131072</td>
<td>32768</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>34.76</td>
<td>3.77</td>
<td>6.60</td>
<td>0</td>
<td>34.43</td>
<td>3.81</td>
<td>6.66</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>262144</td>
<td>65536</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>67.83</td>
<td>3.86</td>
<td>6.76</td>
<td>0</td>
<td>66.68</td>
<td>3.93</td>
<td>6.88</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>524288</td>
<td>131072</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>90.54</td>
<td>5.79</td>
<td>10.13</td>
<td>0</td>
<td>90.54</td>
<td>5.79</td>
<td>10.13</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1048576</td>
<td>262144</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>101.7</td>
<td>10.31</td>
<td>18.05</td>
<td>0</td>
<td>101.4</td>
<td>10.34</td>
<td>18.10</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>2097152</td>
<td>524288</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>125.1</td>
<td>16.77</td>
<td>29.34</td>
<td>0</td>
<td>125.0</td>
<td>16.78</td>
<td>29.37</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>4194304</td>
<td>1048576</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>170.8</td>
<td>24.56</td>
<td>42.99</td>
<td>0</td>
<td>170.0</td>
<td>24.68</td>
<td>43.19</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>8388608</td>
<td>2097152</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>271.6</td>
<td>30.89</td>
<td>54.06</td>
<td>0</td>
<td>271.8</td>
<td>30.86</td>
<td>54.01</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>16777216</td>
<td>4194304</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>286.7</td>
<td>58.52</td>
<td>102.41</td>
<td>0</td>
<td>284.6</td>
<td>58.96</td>
<td>103.18</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>33554432</td>
<td>8388608</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>498.5</td>
<td>67.31</td>
<td>117.80</td>
<td>0</td>
<td>498.3</td>
<td>67.34</td>
<td>117.84</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>67108864</td>
<td>16777216</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>931.1</td>
<td>72.08</td>
<td>126.13</td>
<td>0</td>
<td>932.9</td>
<td>71.93</td>
<td>125.89</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>134217728</td>
<td>33554432</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>1804.6</td>
<td>74.38</td>
<td>130.16</td>
<td>0</td>
<td>1804.2</td>
<td>74.39</td>
<td>130.19</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>268435456</td>
<td>67108864</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>3552.8</td>
<td>75.56</td>
<td>132.22</td>
<td>0</td>
<td>3551.5</td>
<td>75.58</td>
<td>132.27</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>536870912</td>
<td>134217728</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>7059.0</td>
<td>76.05</td>
<td>133.10</td>
<td>0</td>
<td>7047.4</td>
<td>76.18</td>
<td>133.31</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>1073741824</td>
<td>268435456</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>13993</td>
<td>76.73</td>
<td>134.28</td>
<td>0</td>
<td>14009</td>
<td>76.64</td>
<td>134.13</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>2147483648</td>
<td>536870912</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>27847</td>
<td>77.12</td>
<td>134.96</td>
<td>0</td>
<td>27750</td>
<td>77.39</td>
<td>135.43</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>4294967296</td>
<td>1073741824</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>55157</td>
<td>77.87</td>
<td>136.27</td>
<td>0</td>
<td>55313</td>
<td>77.65</td>
<td>135.88</td>
<td>0</td>
</tr>
</tbody>
</table>
RCCL Test Data - 2 Nodes With TCP/IP for Inter Node Communications

```
srun -n16 -N2 --tasks-per-node=8 --gpu-bind=closest all_reduce_perf -b 64K -e 4G -f 2 -g 1
```

### out-of-place

<table>
<thead>
<tr>
<th>#</th>
<th>size (B)</th>
<th>count (elements)</th>
<th>type</th>
<th>redop</th>
<th>root</th>
<th>time (us)</th>
<th>algbw (GB/s)</th>
<th>busbw (GB/s)</th>
<th>#wrong</th>
<th>time (us)</th>
<th>algbw (GB/s)</th>
<th>busbw (GB/s)</th>
<th>#wrong</th>
</tr>
</thead>
<tbody>
<tr>
<td>65536</td>
<td>16384</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td></td>
<td>138.8</td>
<td>0.47</td>
<td>0.89</td>
<td>0</td>
<td>132.1</td>
<td>0.50</td>
<td>0.93</td>
<td>0</td>
</tr>
<tr>
<td>131072</td>
<td>32768</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td></td>
<td>143.0</td>
<td>0.92</td>
<td>1.72</td>
<td>0</td>
<td>151.3</td>
<td>0.87</td>
<td>1.62</td>
<td>0</td>
</tr>
<tr>
<td>262144</td>
<td>65536</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td></td>
<td>206.9</td>
<td>1.27</td>
<td>2.38</td>
<td>0</td>
<td>206.7</td>
<td>1.27</td>
<td>2.38</td>
<td>0</td>
</tr>
<tr>
<td>524288</td>
<td>131072</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td></td>
<td>288.8</td>
<td>1.82</td>
<td>3.40</td>
<td>0</td>
<td>282.3</td>
<td>1.86</td>
<td>3.48</td>
<td>0</td>
</tr>
<tr>
<td>1048576</td>
<td>262144</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td></td>
<td>438.2</td>
<td>2.39</td>
<td>4.49</td>
<td>0</td>
<td>447.1</td>
<td>2.35</td>
<td>4.40</td>
<td>0</td>
</tr>
<tr>
<td>2097152</td>
<td>524288</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td></td>
<td>732.4</td>
<td>2.86</td>
<td>5.37</td>
<td>0</td>
<td>693.3</td>
<td>3.02</td>
<td>5.67</td>
<td>0</td>
</tr>
<tr>
<td>4194304</td>
<td>1048576</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td></td>
<td>1235.0</td>
<td>3.40</td>
<td>6.37</td>
<td>0</td>
<td>1243.1</td>
<td>3.37</td>
<td>6.33</td>
<td>0</td>
</tr>
<tr>
<td>8388608</td>
<td>2097152</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td></td>
<td>2086.3</td>
<td>4.02</td>
<td>7.54</td>
<td>0</td>
<td>2070.2</td>
<td>4.05</td>
<td>7.60</td>
<td>0</td>
</tr>
<tr>
<td>16777216</td>
<td>4194304</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td></td>
<td>4069.6</td>
<td>4.12</td>
<td>7.73</td>
<td>0</td>
<td>3981.8</td>
<td>4.21</td>
<td>7.90</td>
<td>0</td>
</tr>
<tr>
<td>33554432</td>
<td>8388608</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td></td>
<td>7602.9</td>
<td>4.41</td>
<td>8.28</td>
<td>0</td>
<td>7622.4</td>
<td>4.40</td>
<td>8.25</td>
<td>0</td>
</tr>
<tr>
<td>67108864</td>
<td>16777216</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td></td>
<td>15715</td>
<td>4.27</td>
<td>8.01</td>
<td>0</td>
<td>15658</td>
<td>4.29</td>
<td>8.04</td>
<td>0</td>
</tr>
<tr>
<td>134217728</td>
<td>33554432</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td></td>
<td>32255</td>
<td>4.16</td>
<td>7.80</td>
<td>0</td>
<td>32296</td>
<td>4.16</td>
<td>7.79</td>
<td>0</td>
</tr>
<tr>
<td>268435456</td>
<td>67108864</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td></td>
<td>64577</td>
<td>4.16</td>
<td>7.79</td>
<td>0</td>
<td>64669</td>
<td>4.15</td>
<td>7.78</td>
<td>0</td>
</tr>
<tr>
<td>536870912</td>
<td>134217728</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td></td>
<td>128849</td>
<td>4.17</td>
<td>7.81</td>
<td>0</td>
<td>128647</td>
<td>4.17</td>
<td>7.82</td>
<td>0</td>
</tr>
<tr>
<td>1073741824</td>
<td>268435456</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td></td>
<td>252600</td>
<td>4.25</td>
<td>7.97</td>
<td>0</td>
<td>252861</td>
<td>4.25</td>
<td>7.96</td>
<td>0</td>
</tr>
<tr>
<td>2147483648</td>
<td>536870912</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td></td>
<td>503080</td>
<td>4.27</td>
<td>8.00</td>
<td>0</td>
<td>503154</td>
<td>4.27</td>
<td>8.00</td>
<td>0</td>
</tr>
<tr>
<td>4294967296</td>
<td>1073741824</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td></td>
<td>1002721</td>
<td>4.28</td>
<td>8.03</td>
<td>0</td>
<td>1003000</td>
<td>4.28</td>
<td>8.03</td>
<td>0</td>
</tr>
</tbody>
</table>

### in-place

```
135.88 GB/s to 8.03 GB/s. No one is happy.
```
AWS-OFI-RCCL Basic

- From the plugin README: "This project implements a plug-in which maps RCCLs connection-oriented transport APIs to libfabric's connection-less reliable interface."

- This allows RCCL applications to take benefit of libfabric's transport layer services like reliable message support and operating system bypass.

- The current version of AWS-OFI-RCCL plugin implements NCCL network API (ncclNet_v5) to utilize Libfabric. For example, an isend in RCCL will use ofi_isend in the plugin instead.

- The current version of AWS-OFI-RCCL plugin does not implement collNet (for inter-node allReduce):
  - Collective algorithms are implemented in RCCL. No Libfabric/CXI involved.

- Read more about NCCL/RCCL Net Plugin in: https://github.com/ROCm/rccl/tree/develop/ext-net.
AWS-OFI-RCCL Plugin Build Script

#!/usr/bin/bash -i
rocm_version=5.7.1
git clone --recursive --depth=1 https://github.com/ROCmSoftwarePlatform/aws-ofi-rccl
cd aws-ofi-rccl
module load libtool
module swap PrgEnv-cray PrgEnv-gnu
module load rocm/$rocm_version
module load craype-accel-amd-gfx90a
module load gcc/12.2.0
module load cray-mpich/8.1.27
libfabric_path=/opt/cray/libfabric/1.15.2.0
./autogen.sh
export LD_LIBRARY_PATH=/opt/rocm-$rocm_version/hip/lib:$LD_LIBRARY_PATH
CC=cc CFLAGS=-I/opt/rocm-$rocm_version/rccl/include ./configure \
   --with-libfabric=$libfabric_path --with-rccl=/opt/rocm-$rocm_version --enable-trace \
   --prefix=$PWD --with-hip=/opt/rocm-$rocm_version/hip --with-mpi=$MPICH_DIR
make
make install
**RCCL Test Data - 2 Nodes with the Plugin**

```
LD_LIBRARY_PATH=${PATH TO THE PLUGIN DIRECTORY}:${LD_LIBRARY_PATH}
srun -n16 -N2 --tasks-per-node=8 --gpu-bind=closest all_reduce_perf -b 64K -e 4G -f 2 -g 1
```

<table>
<thead>
<tr>
<th>#</th>
<th>size</th>
<th>count (elements)</th>
<th>type</th>
<th>redop</th>
<th>root</th>
<th>time (us)</th>
<th>out-of-place algbw (GB/s)</th>
<th>busbw (GB/s)</th>
<th>#wrong</th>
<th>in-place time (us)</th>
<th>algbw (GB/s)</th>
<th>busbw (GB/s)</th>
<th>#wrong</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>65536</td>
<td>16384</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>48.15</td>
<td>1.36</td>
<td>2.55</td>
<td>0</td>
<td>44.81</td>
<td>1.46</td>
<td>2.74</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>131072</td>
<td>32768</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>66.72</td>
<td>1.96</td>
<td>3.68</td>
<td>0</td>
<td>59.44</td>
<td>2.21</td>
<td>4.13</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>262144</td>
<td>65536</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>94.59</td>
<td>2.77</td>
<td>5.20</td>
<td>0</td>
<td>83.99</td>
<td>3.12</td>
<td>5.85</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>524288</td>
<td>131072</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>131.9</td>
<td>3.98</td>
<td>7.45</td>
<td>0</td>
<td>129.7</td>
<td>4.04</td>
<td>7.58</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1048576</td>
<td>262144</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>151.5</td>
<td>6.92</td>
<td>12.98</td>
<td>0</td>
<td>151.0</td>
<td>6.94</td>
<td>13.02</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>2097152</td>
<td>524288</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>191.4</td>
<td>10.96</td>
<td>20.54</td>
<td>0</td>
<td>191.6</td>
<td>10.95</td>
<td>20.52</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>4194304</td>
<td>1048576</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>282.6</td>
<td>14.84</td>
<td>27.83</td>
<td>0</td>
<td>282.9</td>
<td>14.83</td>
<td>27.80</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>8388608</td>
<td>2097152</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>441.6</td>
<td>19.00</td>
<td>35.62</td>
<td>0</td>
<td>442.1</td>
<td>18.97</td>
<td>35.58</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>16777216</td>
<td>4194304</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>718.8</td>
<td>23.34</td>
<td>43.77</td>
<td>0</td>
<td>719.8</td>
<td>23.31</td>
<td>43.70</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>33554432</td>
<td>8388608</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>1502.7</td>
<td>22.33</td>
<td>41.87</td>
<td>0</td>
<td>1273.5</td>
<td>26.35</td>
<td>49.40</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>67108864</td>
<td>16777216</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>2436.5</td>
<td>27.54</td>
<td>51.64</td>
<td>0</td>
<td>2437.3</td>
<td>27.53</td>
<td>51.63</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>134217728</td>
<td>33554432</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>4617.0</td>
<td>29.07</td>
<td>54.51</td>
<td>0</td>
<td>4623.6</td>
<td>29.03</td>
<td>54.43</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>268435456</td>
<td>67108864</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>9119.3</td>
<td>29.44</td>
<td>55.19</td>
<td>0</td>
<td>9146.5</td>
<td>29.35</td>
<td>55.03</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>536870912</td>
<td>134217728</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>17938</td>
<td>29.93</td>
<td>56.12</td>
<td>0</td>
<td>17882</td>
<td>30.02</td>
<td>56.29</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>1073741824</td>
<td>268435456</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>35112</td>
<td>30.58</td>
<td>57.34</td>
<td>0</td>
<td>35057</td>
<td>30.63</td>
<td>57.43</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>2147483648</td>
<td>536870912</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>70499</td>
<td>30.46</td>
<td>57.11</td>
<td>0</td>
<td>70247</td>
<td>30.57</td>
<td>57.32</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>4294967296</td>
<td>1073741824</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>140733</td>
<td>30.52</td>
<td>57.22</td>
<td>0</td>
<td>141643</td>
<td>30.32</td>
<td>56.85</td>
<td>0</td>
</tr>
</tbody>
</table>

No need to rebuild your binaries.
**LD_LIBRARY_PATH=${PATH TO THE PLUGIN DIRECTORY}:${LD_LIBRARY_PATH}
NCCL_NET_GDR_LEVEL=3**
srun -n16 -N2 --tasks-per-node=8 --gpu-bind=closest all_reduce_perf -b 64K -e 4G -f 2 -g 1

## RCCL Test Data - 2 Nodes with the Plugin + NCCL_NET_GDR_LEVEL=3

<table>
<thead>
<tr>
<th>#</th>
<th>size (B)</th>
<th>count (elements)</th>
<th>type</th>
<th>redop</th>
<th>root</th>
<th>time (us)</th>
<th>algbw (GB/s)</th>
<th>busy (GB/s)</th>
<th>#wrong</th>
<th>time (us)</th>
<th>algbw (GB/s)</th>
<th>busy (GB/s)</th>
<th>#wrong</th>
</tr>
</thead>
<tbody>
<tr>
<td>65536</td>
<td>16384</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>47.80</td>
<td>1.37</td>
<td>2.57</td>
<td>0</td>
<td>45.96</td>
<td>1.43</td>
<td>2.67</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>131072</td>
<td>32768</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>66.79</td>
<td>1.96</td>
<td>3.68</td>
<td>0</td>
<td>57.10</td>
<td>2.30</td>
<td>4.30</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>262144</td>
<td>65536</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>87.64</td>
<td>2.99</td>
<td>5.61</td>
<td>0</td>
<td>81.01</td>
<td>3.24</td>
<td>6.07</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>524288</td>
<td>131072</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>125.5</td>
<td>4.18</td>
<td>7.83</td>
<td>0</td>
<td>125.2</td>
<td>4.19</td>
<td>7.85</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1048576</td>
<td>262144</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>145.1</td>
<td>7.23</td>
<td>13.55</td>
<td>0</td>
<td>145.4</td>
<td>7.21</td>
<td>13.52</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2097152</td>
<td>524288</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>181.7</td>
<td>11.54</td>
<td>21.64</td>
<td>0</td>
<td>183.1</td>
<td>11.46</td>
<td>21.48</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4194304</td>
<td>1048576</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>266.1</td>
<td>15.76</td>
<td>29.56</td>
<td>0</td>
<td>263.6</td>
<td>15.91</td>
<td>29.83</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8388608</td>
<td>2097152</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>341.0</td>
<td>24.60</td>
<td>46.13</td>
<td>0</td>
<td>342.0</td>
<td>24.53</td>
<td>45.99</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>16777216</td>
<td>4194304</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>549.4</td>
<td>30.54</td>
<td>57.26</td>
<td>0</td>
<td>534.5</td>
<td>31.39</td>
<td>58.85</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>33554432</td>
<td>8388608</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>1810.3</td>
<td>18.53</td>
<td>34.75</td>
<td>0</td>
<td>886.4</td>
<td>37.85</td>
<td>70.98</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>67108864</td>
<td>16777216</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>1508.5</td>
<td>44.49</td>
<td>83.41</td>
<td>0</td>
<td>1507.6</td>
<td>44.51</td>
<td>83.46</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>134217728</td>
<td>33554432</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>2747.7</td>
<td>48.85</td>
<td>91.59</td>
<td>0</td>
<td>2746.8</td>
<td>48.86</td>
<td>91.62</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>268435456</td>
<td>67108864</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>5165.4</td>
<td>51.97</td>
<td>97.44</td>
<td>0</td>
<td>5164.1</td>
<td>51.98</td>
<td>97.46</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>536870912</td>
<td>134217728</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>9971.1</td>
<td>53.84</td>
<td>100.95</td>
<td>0</td>
<td>9967.0</td>
<td>53.87</td>
<td>101.00</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1073741824</td>
<td>268435456</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>19099</td>
<td>56.22</td>
<td>105.41</td>
<td>0</td>
<td>19098</td>
<td>56.22</td>
<td>105.42</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2147483648</td>
<td>536870912</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>37328</td>
<td>57.53</td>
<td>107.87</td>
<td>0</td>
<td>37323</td>
<td>57.54</td>
<td>107.88</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4294967296</td>
<td>1073741824</td>
<td>float</td>
<td>sum</td>
<td>-1</td>
<td>73869</td>
<td>58.14</td>
<td>109.02</td>
<td>0</td>
<td>73877</td>
<td>58.14</td>
<td>109.01</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
### RCCL Test Data - All_reduce on 2 nodes, 16 ranks (n16N2)

<table>
<thead>
<tr>
<th>Message Size (bytes)</th>
<th>No Plugin</th>
<th>With Plugin</th>
<th>With Plugin + GDR = 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>65536</td>
<td>0.93</td>
<td>2.74</td>
<td>2.67</td>
</tr>
<tr>
<td>131072</td>
<td>1.62</td>
<td>4.13</td>
<td>4.3</td>
</tr>
<tr>
<td>262144</td>
<td>2.38</td>
<td>5.85</td>
<td>6.07</td>
</tr>
<tr>
<td>524288</td>
<td>3.48</td>
<td>7.58</td>
<td>7.85</td>
</tr>
<tr>
<td>1048576</td>
<td>4.4</td>
<td>13.02</td>
<td>13.52</td>
</tr>
<tr>
<td>2097152</td>
<td>5.67</td>
<td>20.52</td>
<td>21.48</td>
</tr>
<tr>
<td>4194304</td>
<td>6.33</td>
<td>27.8</td>
<td>29.83</td>
</tr>
<tr>
<td>8388608</td>
<td>7.6</td>
<td>35.58</td>
<td>45.99</td>
</tr>
<tr>
<td>16777216</td>
<td>7.9</td>
<td>43.7</td>
<td>58.85</td>
</tr>
<tr>
<td>33554432</td>
<td>8.25</td>
<td>49.4</td>
<td>70.98</td>
</tr>
<tr>
<td>67108864</td>
<td>8.04</td>
<td>51.63</td>
<td>83.46</td>
</tr>
<tr>
<td>134217728</td>
<td>7.79</td>
<td>54.43</td>
<td>91.62</td>
</tr>
<tr>
<td>268435456</td>
<td>7.78</td>
<td>55.03</td>
<td>97.46</td>
</tr>
<tr>
<td>536870912</td>
<td>7.82</td>
<td>56.29</td>
<td>101</td>
</tr>
<tr>
<td>1073741824</td>
<td>7.96</td>
<td>57.43</td>
<td>105.42</td>
</tr>
<tr>
<td>2147483648</td>
<td>8</td>
<td>57.32</td>
<td>107.88</td>
</tr>
<tr>
<td>4294967296</td>
<td>8.03</td>
<td>56.85</td>
<td>109.01</td>
</tr>
</tbody>
</table>
Use NCCL_DEBUG=info to Verify Use of the Plugin

- Checking if the plugin exists:
  
  frontier03329:113050:113050 [0] NCCL INFO NET/Plugin: Failed to find ncclNetPlugin_v6 symbol.
  frontier03329:113050:113050 [0] NCCL INFO NET/Plugin: Loaded net plugin AWS Libfabric (v5)
  frontier03329:113050:113050 [0] NCCL INFO NET/Plugin: Failed to find ncclCollNetPlugin_v6 symbol.
  frontier03329:113050:113050 [0] NCCL INFO NET/Plugin: Failed to find ncclCollNetPlugin symbol (v4 or v5).

- Without the plugin
  
  frontier01466:86382:86734 [0] NCCL INFO Channel 02/0 : 36[d9000] -> 45[de000] [send] via NET/Socket/4 comm 0x7ffbec000d70 nRanks 80

- With the plugin
  
  frontier08155:21778:21989 [0] NCCL INFO Channel 01/0 : 70[c1000] -> 74[c9000] [send] via NET/AWS Libfabric/0 comm 0x7ffbec000d70 nRanks 80

- With the plugin + GDR3
  
  frontier08431:43731:43989 [0] NCCL INFO Channel 01/0 : 14[c1000] -> 18[c9000] [send] via NET/AWS Libfabric/0/GDRDMA comm 0x7ffbec000d70 nRanks 80

The plugin implements NCCL Net API v5

The plugin does not support ncclCollNet

Use NCCL_DEBUG=info to Verify Use of the Plugin

The plugin implements NCCL Net API v5

The plugin does not support ncclCollNet
Comparison of NCCL_ALGO, 64 nodes, 512 ranks

<table>
<thead>
<tr>
<th>Message Size</th>
<th>Default</th>
<th>ALGO=Tree</th>
<th>ALGO=Ring</th>
</tr>
</thead>
<tbody>
<tr>
<td>65536</td>
<td>0.85</td>
<td>0.85</td>
<td>0</td>
</tr>
<tr>
<td>131072</td>
<td>1.17</td>
<td>1.18</td>
<td>0</td>
</tr>
<tr>
<td>262144</td>
<td>1.41</td>
<td>1.42</td>
<td>0</td>
</tr>
<tr>
<td>524288</td>
<td>2.59</td>
<td>2.43</td>
<td>0.01</td>
</tr>
<tr>
<td>1048576</td>
<td>4.18</td>
<td>4.46</td>
<td>0.77</td>
</tr>
<tr>
<td>2097152</td>
<td>5.75</td>
<td>8.6</td>
<td>0.67</td>
</tr>
<tr>
<td>4194304</td>
<td>8.65</td>
<td>7.67</td>
<td>1.32</td>
</tr>
<tr>
<td>8388608</td>
<td>12.15</td>
<td>10</td>
<td>1.24</td>
</tr>
<tr>
<td>16777216</td>
<td>17.59</td>
<td>13.14</td>
<td>1.17</td>
</tr>
<tr>
<td>33554432</td>
<td>16.28</td>
<td>20.14</td>
<td>2.16</td>
</tr>
<tr>
<td>67108864</td>
<td>31.34</td>
<td>29.49</td>
<td>0.19</td>
</tr>
<tr>
<td>134217728</td>
<td>54.02</td>
<td>45.43</td>
<td>2.01</td>
</tr>
<tr>
<td>268435456</td>
<td>56.66</td>
<td>56.08</td>
<td>17.78</td>
</tr>
<tr>
<td>536870912</td>
<td>68.37</td>
<td>68.99</td>
<td>30.09</td>
</tr>
<tr>
<td>1073741824</td>
<td>70.81</td>
<td>70.77</td>
<td>57.92</td>
</tr>
<tr>
<td>2147483648</td>
<td>90.02</td>
<td>73.43</td>
<td>90.3</td>
</tr>
<tr>
<td>4294967296</td>
<td>90.31</td>
<td>75</td>
<td>90.45</td>
</tr>
</tbody>
</table>
CPU-GPU Binding

- Goal: match CPU cores to the GCD in the same L3 region
- RCCL self-checks the *GPU* and NIC topology
- Result: it doesn't need to bind to CPU cores if there's not much CPU-GPU transfer

Nevertheless, it won't hurt to bind, so use
--gpus-per-task=1 --gpu-bind=closest and not
--tasks-per-node=8
Known Issues

- Does not perform well at larger scale (>= 256 nodes).
- Hang or crash when using more than 256 nodes and with NCCL_NET_GDR_LEVEL=3 set.
- Current plugin version (based on aws-ofi-nccl 1.4.0) is behind the latest aws-ofi-nccl version (1.8.1)
Part II: PyTorch Examples
A Harmless Message

• [W socket.cpp:697] [c10d] The client socket cannot be initialized to connect to [crusher002.crusher.olcf.ornl.gov]:3442 (errno: 97 – Address family not supported by protocol).

• After PyTorch v1.x, when using tcp to initialize PyTorch DDP, the default is to use IPv6 address; PyTorch falls back to use IPv4 if IPv6 does not work.

• The message is harmless, and it does not affect PyTorch-DDP uses RCCL as backend.
PyTorch Examples from AI-AT-SCALE-PART 2

- https://github.com/olcf/ai-training-series
- The training provides is a set of well-organized PyTorch-DDP examples. We use two of them in this talk:
  - launch_bigbird.crusher
  - launch_bigbird_srun.frontier
  - launch_gptJ_srun.frontier
  - launch_bigbird.frontier
  - launch_gptJ_fsdp.frontier
  - launch_gpt_srun.frontier

- Follow the instruction to prepare and download data:
  - git clone --recursive https://github.com/olcf/ai-training-series
  - In ai_at_scale_part_2/ddp_examples directory:
    - bash download_oscar.sh
    - bash dl_models.sh
Minor Modifications for Running the PyTorch Scripts

- **Account**
  
  #SBATCH -A `{your project account}`

- **Environment**

  ```bash
  source /lustre/orion/world-shared/stf218/sajal/miniconda3/bin/activate
  conda activate /lustre/orion/world-shared/stf218/sajal/TORCH2/env-py310-rccl
  ```

  This environment uses rocm/5.4.0 and pytorch-rocm/2.1.0, deepspeed/0.9.1.

- **Add path to the plugin and the GDR variable**

  ```bash
  #export NCCL_DEBUG=INFO
  export LD_LIBRARY_PATH=${PATH TO THE PLUGIN}/libs/
  #export FI_LOG_LEVEL=info
  export NCCL_NET_GDR_LEVEL=3
  ```

- **Or use Sajal’s pre-build library**

  ```bash
  export LD_LIBRARY_PATH=/lustre/orion/world-shared/stf218/sajal/aws-ofi-rcc1/lib/:${LD_LIBRARY_PATH}
  ```
Experiment Data - GPT-J

- “sbatch launch_gptJ_srun.frontier”
- Experiment setting: 16 nodes, 8 GPUs per node, max_steps=10, per_device_train_batch_size=4

- Without the plugin
  
  ```json
  {
  'train_runtime': 1126.1257,
  'train_samples_per_second': 36.372,
  'train_steps_per_second': 0.009,
  'train_loss': 80.29856567382812,
  'epoch': 0.09
  }
  ```

- With the plugin
  
  ```json
  {
  'train_runtime': 671.7561,
  'train_samples_per_second': 60.975,
  'train_steps_per_second': 0.015,
  'train_loss': 79.96430053710938,
  'epoch': 0.09
  }
  ```

- With the plugin + NCCL_NET_GDR_LEVEL=3
  
  ```json
  {
  'train_runtime': 568.0539,
  'train_samples_per_second': 72.106,
  'train_steps_per_second': 0.018,
  'train_loss': 79.44228515625,
  'epoch': 0.09
  }
  
  Good performance improvement by using the plugin.
Experiment Data - BigBird Oscar

• “sbatch launch_bigbird_srun.frontier”

• Experiment setting: 10 nodes, 8 GPUs per node, 100 steps, per_device_train_batch_size =24
  • ”train_micro_batch_size_per_gpu” in ds_config.json is 4; change it to 24 to match per_device_train_batch_size.

• Without the plugin
  
  `{train_runtime': 208.6629, 'train_samples_per_second': 920.145, 'train_steps_per_second': 0.479, 'train_loss': 6.68152099609375, 'epoch': 0.16}

• With the plugin
  
  `{train_runtime': 204.6703, 'train_samples_per_second': 938.094, 'train_steps_per_second': 0.489, 'train_loss': 6.672891845703125, 'epoch': 0.16}

• With the plugin + NCCL_NET_GDR_LEVEL=3
  
  `{train_runtime': 203.6554, 'train_samples_per_second': 942.769, 'train_steps_per_second': 0.491, 'train_loss': 6.687650756835938, 'epoch': 0.16}

Almost no performance improvement by using the plugin.
PyTorch Profiler Basic

- [https://pytorch.org/tutorials/recipes/recipes/profiler_recipe.html](https://pytorch.org/tutorials/recipes/recipes/profiler_recipe.html)

```python
with torch.profiler.profile(activities=[torch.profiler.ProfilerActivity.CPU, torch.profiler.ProfilerActivity.CUDA]) as profiler:

    for epoch in range(epochs_trained, num_train_epochs):
        .
        training codes
        .

    print(profiler.key_averages().table(sort_by="cuda_time_total", row_limit=20))
    print(profiler.key_averages().table(sort_by="cpu_time_total", row_limit=20))
```
with torch.profiler.profile(
    activities=[torch.profiler.ProfilerActivity.CPU,
                torch.profiler.ProfilerActivity.CUDA],
    schedule=torch.profiler.schedule(
        wait=1,
        warmup=1,
        active=2,
        repeat=1),
    on_trace_ready=torch.profiler.tensorboard_trace_handler('./logs/gpt'),
) as profiler:

    for epoch in range(epochs_trained, num_train_epochs):
        .
        training codes
        .
        profiler.step()

print(profiler.key_averages().table(sort_by="cuda_time_total", row_limit=20))
print(profiler.key_averages().table(sort_by="cpu_time_total", row_limit=20))
Modifications for Using PyTorch Profiler

• The BigBird Oscar script (bigbird_oscar_srun.py) uses custom_trainer.py, which has PyTorch Profiler enabled but does not print the performance data table.
  • Add the following two lines after the training loop (line 446):

```
print(profiler.key_averages().table(sort_by="cuda_time_total", row_limit=20))
print(profiler.key_averages().table(sort_by="cpu_time_total", row_limit=20))
```

• The GPT-J case uses the trainer provided by HuggingFace’s Transformers; adding PyTorch Profiler codes requires more work.

• To keep things simple – modify the gptJ_oscar_srun.py script to use customer_trainer.py instead. The results show similar performance characteristic to using HuggingFace’s Transformers’s trainer.
  • In the file gptJ_oscar_srun.py, change the line:

```
trainer = Trainer
to
trainer = MyTrainer
```
Examining PyTorch Profiler Outputs

• There are three ways to view PyTorch Profiler outputs:
  • **Text output**
    – Basic PyTorch statistics shown at the end of a run.
  
  • **Viewing tracing output use Perfetto**
    – Need to download tracing files to local systems and then use tools such as Perfetto (https://ui.perfetto.dev/) to view the files.

  • **Tensorboard-tb-plugin**
    – Need to start a Tensorboard server on a Frontier login node, make an ssh connection, and then use a browser to connect to the login to view the data.

• PyTorch Profiler generates many types of performance data; the focus here is to evaluate information related to communications from the three types of output.
c10d Call Stack from a PyTorch Tracing File

- c10d is PyTorch’s collective communication layer. When RCCL is used as the backend, PyTorch c10d uses RCCL collectives.
- This shows the call stack of c10d allreduce:
  c10d::allreduce ➔ record_param_comms ➔ nccl::all_reduce ➔ ncclKernel_SendRecv_RING_SIMPLE_Sum_Int8_t().
A table generated by PyTorch Profiler can have (too) many columns. This is the minimum number of columns when setting to profile with ProfilerActivity.CPU and ProfilerActivity.CUDA.

From the profiling output, c10d::reduce time is significant.
# PyTorch Profiler Test Output

<table>
<thead>
<tr>
<th>Name</th>
<th>Self CPU %</th>
<th>Self CPU</th>
<th>CPU total %</th>
<th>CPU total</th>
<th>CPU time avg</th>
<th>Self CUDA</th>
<th>Self CUDA %</th>
<th>CUDA total</th>
<th>CUDA time avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>autograd::engine::evaluate_function: SliceBackward0</td>
<td>0.06%</td>
<td>3.862ms</td>
<td>1.19%</td>
<td>80.099ms</td>
<td>32.403us</td>
<td>0.000us</td>
<td>0.00%</td>
<td>1.425s</td>
<td>576.324us</td>
</tr>
<tr>
<td>ProfilerStep*</td>
<td>44.69%</td>
<td>3.007s</td>
<td>57.13%</td>
<td>3.845s</td>
<td>1.922s</td>
<td>0.000us</td>
<td>0.00%</td>
<td>1.403s</td>
<td>701.365ms</td>
</tr>
<tr>
<td>aten::slice_backward</td>
<td>0.13%</td>
<td>8.933ms</td>
<td>0.89%</td>
<td>59.795ms</td>
<td>24.190ms</td>
<td>0.000us</td>
<td>0.00%</td>
<td>1.117s</td>
<td>451.718us</td>
</tr>
<tr>
<td>SliceBackward0</td>
<td>0.15%</td>
<td>10.059ms</td>
<td>0.95%</td>
<td>64.199ms</td>
<td>25.970us</td>
<td>0.000us</td>
<td>0.00%</td>
<td>1.032s</td>
<td>417.547us</td>
</tr>
<tr>
<td>aten::copy_</td>
<td>0.20%</td>
<td>13.452ms</td>
<td>18.30%</td>
<td>1.232s</td>
<td>265.930ms</td>
<td>32.403us</td>
<td>0.00%</td>
<td>944.254ms</td>
<td>203.854us</td>
</tr>
<tr>
<td>autograd::engine::evaluate_function: AddmmBackward0</td>
<td>0.02%</td>
<td>1.599ms</td>
<td>0.23%</td>
<td>15.242ms</td>
<td>102.986us</td>
<td>0.000us</td>
<td>0.00%</td>
<td>816.697ms</td>
<td>5.518ms</td>
</tr>
<tr>
<td>AddmmBackward0</td>
<td>0.03%</td>
<td>1.893ms</td>
<td>0.15%</td>
<td>9.883ms</td>
<td>66.777us</td>
<td>0.000us</td>
<td>0.00%</td>
<td>761.703ms</td>
<td>5.147ms</td>
</tr>
<tr>
<td>aten::mm</td>
<td>0.07%</td>
<td>4.745ms</td>
<td>0.10%</td>
<td>6.544ms</td>
<td>22.108us</td>
<td>0.000us</td>
<td>0.00%</td>
<td>761.703ms</td>
<td>2.573ms</td>
</tr>
<tr>
<td>CopyToDeviceToDevice</td>
<td>0.00%</td>
<td>0.000us</td>
<td>0.00%</td>
<td>0.000us</td>
<td>0.000us</td>
<td>0.000us</td>
<td>0.00%</td>
<td>369.570ms</td>
<td>2.464ms</td>
</tr>
<tr>
<td>aten::fill_</td>
<td>0.13%</td>
<td>8.684ms</td>
<td>0.34%</td>
<td>22.976ms</td>
<td>6.896us</td>
<td>0.000us</td>
<td>9.13%</td>
<td>494.488ms</td>
<td>148.406ms</td>
</tr>
<tr>
<td>hipLaunchKernel</td>
<td>0.68%</td>
<td>45.624ms</td>
<td>0.69%</td>
<td>45.624ms</td>
<td>4.351us</td>
<td>11.91%</td>
<td>471.582ms</td>
<td>44.973ms</td>
<td></td>
</tr>
<tr>
<td>aten::zeros</td>
<td>0.13%</td>
<td>9.015ms</td>
<td>0.58%</td>
<td>39.004ms</td>
<td>12.367us</td>
<td>0.000us</td>
<td>9.52%</td>
<td>437.015ms</td>
<td>312.154us</td>
</tr>
<tr>
<td>aten::addmm</td>
<td>0.29%</td>
<td>19.395ms</td>
<td>7.86%</td>
<td>529.127ms</td>
<td>339.261ms</td>
<td>8.57%</td>
<td>385.911ms</td>
<td>334.992us</td>
<td></td>
</tr>
<tr>
<td>aten::add</td>
<td>0.10%</td>
<td>6.835ms</td>
<td>0.18%</td>
<td>11.806ms</td>
<td>8.433us</td>
<td>9.52%</td>
<td>437.015ms</td>
<td>312.154us</td>
<td></td>
</tr>
<tr>
<td>aten::zeros</td>
<td>0.13%</td>
<td>9.015ms</td>
<td>0.58%</td>
<td>39.004ms</td>
<td>12.367us</td>
<td>0.000us</td>
<td>9.52%</td>
<td>437.015ms</td>
<td>312.154us</td>
</tr>
<tr>
<td>aten::addmm</td>
<td>0.06%</td>
<td>4.281ms</td>
<td>0.09%</td>
<td>6.102ms</td>
<td>40.680us</td>
<td>0.000us</td>
<td>9.07%</td>
<td>360.402ms</td>
<td>109.946us</td>
</tr>
<tr>
<td>aten::mul</td>
<td>0.13%</td>
<td>8.472ms</td>
<td>0.19%</td>
<td>13.113ms</td>
<td>11.878us</td>
<td>6.49%</td>
<td>316.466ms</td>
<td>286.654us</td>
<td></td>
</tr>
</tbody>
</table>

- c10d time is nowhere to be found in the profiling output.
• This is the view of one epoch (one step).
• All the green blocks are ncclKernel_SendRecv_RING_SIMPLE calls.
This is the view of one epoch (one step).

The BigBird Oscar case only has one `ncclKernel_SendRecv_RING_SIMPLE` call.
The “Marker” Issue in PyTorch-rocm Tracing Files

- Pytorch-rocm prior to 2.1.2 generate many “Marker” blocks in tracing files.
- PyTorch-rocm/2.1.2 and later releases have the issue fixed.
PyTorch Profiler with Tensorboard

- https://pytorch.org/tutorials/intermediate/tensorboard_profiler_tutorial.html

1. Install the Tensorboard Plugin package with
   `pip install torch_tb_profiler`
2. Start tensorboard on a Frontier login node with
   `tensorboard --logdir ${PATH_TO_LOG_DIRECTORY} --host localhost`
3. ssh from local system to the Frontier node, for example:
   `ssh -L 6006:localhost:6006 $USER@login04.frontier.olcf.ornl.gov`

- The “Distributed View” is supposed to provide information about communication vs. computation. However, this view does not always show up.
The Distributed View is not available from tracing output of the GPT-J example.
PyTorch Profiler and the Communication Cost

• Both profiling text output and tracing output show the GPT-J case is communication intensive while BigBird Oscar case is computationally intensive.
  ➔ The plugin improves communication time and thus only the GPT-J case can benefit from using the plugin.

• The TensorBoard PyTorch plugin could be useful in the future if the Distributed View could be available for every DDP run.

• PyTorch Profiler does not provide statistics of communication such as how many times a certain message size was used during a training run.
Revisit RCCL All_reduce on 2 nodes, 16 ranks (-n16 -N2)

- Small message sizes = less performance improvement.
- Message sizes < 4MB, GDR=3 does not help much.
The steps to use Perftools to profile Python codes:

- "module load perftools-base perftools-preload"
- Instead of “srun ... python ${your_script}”:
  - `export PAT_RT_TRACE_PYTHON_GROUPS="pytorch"
  - `srun ... pat_run -w -g rccl python ${your_script}`
A Warning About Using Profiler or Enabling Debugging Messages

- Enabling debugging variables, such as NCCL_DEBUG or FI_LOG_LEVEL, can generate many messages and create large files.

- When using PyTorch Profiler with tracing on, each rank may generate a large file; at large scale the outputs from PyTorch Profiler may use a large amount of disk space.

- The overhead of using a profiler sometimes can be significant and can distort the collected performance data – always double check with the original performance data.
Some Environment Variables for the RCCL Layer

- **On the PyTorch layer**
  - There is no environment variables for the c10d layer.
  - At the moment, setting NCCL as the c10d backend is the only option to run PyTorch at scale on Frontier.
  - If you're not seeing performance improvement with the plugin – try a profiler and evaluate communication cost.

- **On the RCCL layer**
  - NCCL_NET_GDR_LEVEL=3 - Remove this setting if you encounter a hang/crash.
  - NCCL_ALGO=TREE or RING – May see performance difference with this setting.
  - NCCL_CROSS_NIC=1 - On large systems, this NCCL setting has been found to improve performance.
  - NCCL_DEBUG=info for debugging.
Some Environment Variables for the Plugin Layer

• On the OFI-plugin layer
  • Libfabric/CXI variables are mostly for debugging or workaround purpose.
    – `FI_LOG_LEVEL=warn FI_LOG_PROV=cxi` – print only CXI related messages.
    – `FI_MR_CACHE_MONITOR=userfaultfd` – The memory cache monitor is responsible for detecting system memory changes made between the virtual addresses used by an application and the underlying physical pages. Userfaultfd is a Linux kernel feature used to report virtual to physical address mapping changes to user space.
    – `FI_CXI_DISABLE_HOST_REGISTER=1` – Disable registration of host buffers (overflow and request) with GPU
    – `FI_CXI_DEFAULT_CQ_SIZE` – the default is 131072.
      – `FI_CXI_DEFAULT_TX_SIZE` – the default is 256.

• Encounter a hang or crash when using the plugin:
  • Frontier Office Hours.
  • File an OLCFHELP ticket. If possible, a reproducer would be very helpful.
Summary

• Topics we talked about:

  • In Part I:
    – Basic about RCCL, how to build and use the RCCL Tester to evaluate RCCL performance.
    – The OFI-Plugin, and how it improve performance of the RCCL Tester.

  • In Part II:
    – Examples showing performance with and without using the plugin.
    – How to use PyTorch Profiler, and different methods to examine performance data generated by PyTorch Profiler.
Thank you

Mengshiou Wu – meng-shiou.wu@hpe.com
Mark Stock – mark.stock@hpe.com
# Message Histograms from Perftools

## Table 1: Time and Bytes Transferred for Accelerator Regions

<table>
<thead>
<tr>
<th>Time Data</th>
<th>Host-To-Accelerator Transfer Data</th>
<th>Accelerator-to-Host Transfer Data</th>
<th>Accelerator-to-Accelerator Transfer Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time%</td>
<td>Time%</td>
<td>Time%</td>
<td>Time%</td>
</tr>
<tr>
<td>Acc Copy</td>
<td>In</td>
<td>Don't Know</td>
<td>Out</td>
</tr>
<tr>
<td>HtoA</td>
<td>CpySz</td>
<td>CpySz</td>
<td>AtoH</td>
</tr>
<tr>
<td>E&lt;16</td>
<td>CpySz</td>
<td>CpySz</td>
<td>AtoH</td>
</tr>
<tr>
<td>E&lt;256</td>
<td>CpySz</td>
<td>CpySz</td>
<td>AtoH</td>
</tr>
<tr>
<td>E&lt;4K</td>
<td>CpySz</td>
<td>CpySz</td>
<td>AtoH</td>
</tr>
<tr>
<td>E&lt;64K</td>
<td>CpySz</td>
<td>CpySz</td>
<td>AtoH</td>
</tr>
<tr>
<td>E&lt;1M</td>
<td>CpySz</td>
<td>CpySz</td>
<td>AtoH</td>
</tr>
<tr>
<td>E&lt;8M</td>
<td>Count</td>
<td>Count</td>
<td>Count</td>
</tr>
<tr>
<td>E&lt;16M</td>
<td>Count</td>
<td>Count</td>
<td>Count</td>
</tr>
<tr>
<td>E&lt;64M</td>
<td>Count</td>
<td>Count</td>
<td>Count</td>
</tr>
<tr>
<td>E&lt;1G</td>
<td>Count</td>
<td>Count</td>
<td>Count</td>
</tr>
</tbody>
</table>

### Note

- **User-defined Region**

  - python_train
  - examples/distributed/FSDP/train_train.py
  - site-packages/torch/distributed/distributed_c10d.py
TensorFlow Example

- Other MLDL frameworks or codes that use RCCL may benefit from using the plugin.

- TensorFlow tf_cnn_benchmark.py code:
  - Use TensorFlow 2.9 + Horovod/RCCL backend built with ROCm/5.6.0, batch size = 128

<table>
<thead>
<tr>
<th></th>
<th>No Plugin</th>
<th>With the Plugin</th>
<th>With the Plugin + GDR=3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>16 nodes, 128 ranks</strong></td>
<td>118874.19</td>
<td>129670.74</td>
<td>133778.7</td>
</tr>
<tr>
<td><strong>32 nodes, 256 ranks</strong></td>
<td>219602.73</td>
<td>252821.21</td>
<td>261120.75</td>
</tr>
<tr>
<td><strong>64 nodes, 512 ranks</strong></td>
<td>388282.87</td>
<td>477924.70</td>
<td>508689.71</td>
</tr>
</tbody>
</table>

- Numbers = Images per second, higher is better.
• A zoom in view to a torch::autograd::AccumulateGrad step.
• There are many c10d calls.
• Zoom out further. The line shows a c10d/nccl reduce call from an early stage that later invokes ncclKernel_SendRecv_RING_SIMPLE
• The red box is the viewing area in the previous slide.
• A zoom out view of the figure in the previous slide.
• The red box is the viewing area in the previous slide.