# confirming installation and setup running examples in the code #60

New issue

Closed atinsood opened this issue on Oct 28, 2017 · 46 comments

**atinsood** commented on Oct 28, 2017

Guys thanks for this interesting project. Appreciate the effort that you have put in here. I am trying out a few things but have a few doubts if I am doing things right.

I have 2 machines, each with 2 P100s and 16gig mem on each GPU
I am starting with examples/ , and executing the command `~/.openmpi/bin/mpirun -np 4 -v -hostfile ~/hostfile -x NCCL_DEBUG=INFO ~/envs/horovod/bin/python tensorflow_word2vec.py`

I can confirm that this kicks off the example across both my machines in the host files and across the total 4 gpus.

But I have a few doubts if I am missing something, as I am seeing a higher time periods of exec when running `a` single GPU with horovd as compared to vanilla single core TF `b` distributed horovod vs single GPU horovod execution

So just wanted to confirm a few things:

- whether the GPUs are sufficiently being utilized

(nvidia-smi seems to point out that each of the GPU is running the code and 7% of the memory is being used).
From top:
`%Cpu(s): 3.1 us, 0.8 sy, 0.0 ni, 96.1 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st`

```
  PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
  20   0 29.199g 1.487g 421128 S 116.9  2.4   5:32.41 python
  20   0 29.199g 1.495g 421172 S 114.6  2.4   5:24.61 python
```

- whether the GPUs are efficiently communicating with each other

( as pointed out in the other ticket I should see something in the logs about it, but I don't see anything. I can see the processes started across machines, but I am not sure if all the communication is good. The training gets to completion, if I specify incorrect host then I do see errors, so I have a feeling that all the instances are communicating but `a` not sure if they are communicating properly `b` they are communicating via libnccl2

- whether anything special needs to be done from open mpi side
  I basically downloaded openmpi, did not do anything special to build it optimally for say GPUs or libnccl. Do I need to anything additional to optimize/leverage for GPUs

- training times are increasing as we add more GPUs

vanilla tensor flow for above example ~ 3 sec
horovod single GPU ~ 4 sec
horovod with 2 GPU on same machine ~9 sec

**alsrgv** commented on Oct 28, 2017 Collaborator

Hi **@atinsood**,

`tensorflow_word2vec.py` is actually very tiny example utilizing embeddings - small enough that it doesn't scale well. You won't need multiple GPUs to train word2vec model. This example is primarily there to serve two purposes - 1) show that you can do averaging of sparse tensors, used in embeddings, 2) show that you don't have to use `tf.train.MonitoredTrainingSession`.

Assignees
No one assigned

Labels
question

Projects
None yet

Milestone
No milestone

Notifications

3 participants

If you want to see scaling, you can use benchmarks here, branch `horovod_v2`, and run CNN benchmarks (ResNet, Inception, VGG):

```
$ git clone https://github.com/alsrgv/benchmarks; cd benchmarks
$ git checkout horovod_v2
$ mpirun -np 4 python scripts/tf_cnn_benchmarks/tf_cnn_benchmarks.py --model resnet101 --
batch_size 64 --variable_update horovod
```

We also found that LSTMs with embeddings scale well, too - as long as they reach sufficient complexity. E.g. seq2seq model training scales very well.

alsrgv added the question label on Oct 28, 2017

atinsood commented on Oct 28, 2017

@alsrgv appreciate the help in answering the question, seem to be running into this issue when running the above command. I want to think I did install TF correctly `pip install --upgrade ~/tensorflow_gpu-1.3.0-cp27-none-linux_x86_64.whl`

```
ImportError: No module named all_reduce.python```

may be I am missing something
```

alsrgv commented on Oct 28, 2017 Collaborator

@atinsood looks like TF benchmarks have been updated and now require TF 1.4.0-rc1. You'd have to upgrade to that and reinstall Horovod (with `--no-cache-dir` to make sure it actually does rebuild C++ code).

atinsood commented on Oct 28, 2017

@alsrgv this was perfect. thanks that did the trick and I am past the error and able to run `$ mpirun -np 4 python scripts/tf_cnn_benchmarks/tf_cnn_benchmarks.py --model resnet101 --batch_size 64 --variable_update horovod`

I am trying to understand now, if I have to run the same benchmarks that you guys ran and test these against our hardware,

`a` what would be the way to go about it
`b` how do I make sure that all the optmizations you have put in place, using libnccl etc are being leveraged.
`c` any other considerations I should keep in mind.

alsrgv commented on Oct 29, 2017 Collaborator

@atinsood great! You should be able to easily repro out benchmarks just by running that script. Make sure that NCCL is installed and is being used by adding `-x NCCL_DEBUG=INFO` to your `mpirun` command like you did before. If you have RoCE/InfiniBand, I found this `mpirun` option to improve performance, too: `-mca btl_openib_receive_queues P,128,32:P,2048,32:P,12288,32:P,131072,32`

2

atinsood commented on Oct 29, 2017 • edited

@alsrgv perfect perfect. and thank you for being so patient and answering my questions. I guess this is my relative ignore with the deep learning topic and TF, but I am trying to understand the benchmark correctly.

I am looking at this comment #21 (comment) which is also trying to replicate your benchmark

```
mpirun --allow-run-as-root \
-n 8 \
-npernode 4 \
python tf_cnn_benchmarks.py \
--model=resnet50 \
--batch_size=64 \
--horovod_device=gpu \
--num_batches=2000 \
--data_dir=${DATASET_DIR} \
--data_name="imagenet" \
--print_training_accuracy```

and I am assuming that I will also need to provide the data set for the training, though I am not sure when I just run the command that you mentioned `mpirun -np 4 python scripts/tf_cnn_benchmarks/tf_cnn_benchmarks.py --model resnet101 --batch_size 64 --variable_update horovod` . Is that just using a trained model against the test set.

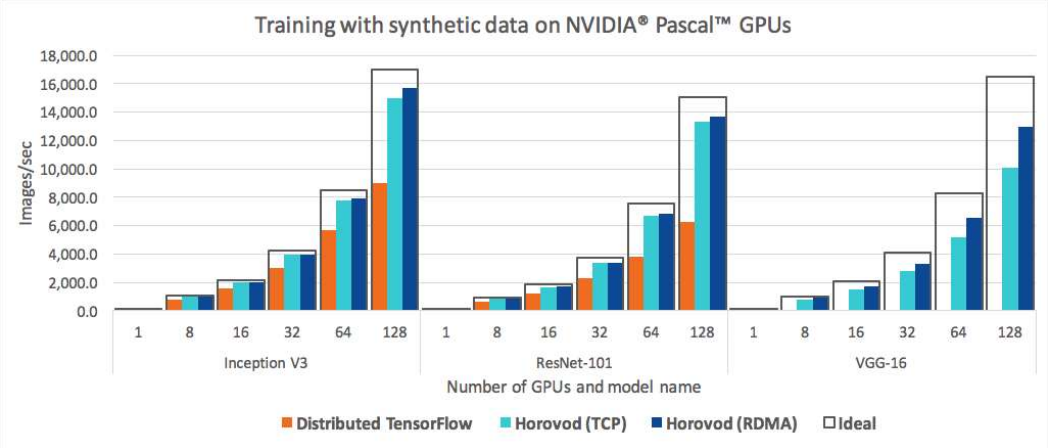
From the blog I feel that you ran the resnet benchmark against imagenet dataset `https://eng.uber.com/horovod/` but please correct me if I am wrong here and mind elaborating a little more on the benchmark.

Pardon my ignorance and really appreciate your patience here.
```

alsrgv commented on Oct 29, 2017 Collaborator

@atinsood, no worries!

The numbers below are actually using synthetic data (i.e. fake data), to purely measure GPU scaling with the assumption that CPU and disk IO speed are sufficient to support that.




`mpirun -np 4 -H host1:2,host2:2 python scripts/tf_cnn_benchmarks/tf_cnn_benchmarks.py --model resnet101 --batch_size 64 --variable_update horovod` will allow you to replicate that.

Now, if you want to throw in real data into the mix, you need to download ImageNet dataset and run the script to convert ImageNet JPEGs into TFRecords.

Then, `mpirun -np 4 -H host1:2,host2:2 python scripts/tf_cnn_benchmarks/tf_cnn_benchmarks.py --model resnet101 --batch_size 64 --variable_update horovod --data_dir /path/to/tfrecord --data_name imagenet --num_batches=2000` will allow you to measure real data performance.

I just did a quick test on 16 GPUs (4x4) on resnet101 with real data and got about 87% scaling efficiency (=14x). During the run, I was observing load average of ~80, so this seems to be overtaxing the CPUs.

I then found this line in preprocessing code and noticed that it's making 64 preprocessing threads **per worker**. After modifying the code to divide that number by 4, I got 92% scaling efficiency (=14.7x) on resnet101 which I'm pretty happy with.

alsrgv referenced this issue on Oct 31, 2017

[q] mpirun crash #62 Closed

atinsood commented on Nov 1, 2017

@alsrgv just wanted to touch base that all the above helped and we were able to get things going at our end. Really appreciate it. Once I have the numbers in and have tried out a couple of iterations going will touch base again.

thanks for all the help.

alsrgv commented on Nov 1, 2017 Collaborator

@atinsood, excellent!

alsrgv added the tracking label on Nov 1, 2017

alsrgv referenced this issue on Nov 3, 2017

**multi-gpu seems to be slower than single gpu** #71 Closed

weixsong commented on Nov 3, 2017

@alsrgv , is that a must to use **-x NCCL_DEBUG=INFO** for speedup training?

alsrgv commented on Nov 3, 2017 Collaborator

@weixsong, it's important to use NCCL, but the `-x NCCL_DEBUG=INFO` flag is just for the diagnostic output. It helps to confirm that NCCL is working correctly.

atinsood commented on Nov 4, 2017

@alsrgv so I was able to get going but have noticed an odd thing with my results on 1, 2, 4 and 6 GPU nodes. I have machines with P100s with each machine holding 2 P100s

Here are my results

```
####(Case 3)####  1 GPU on 1 node
time ~/.openmpi/bin/mpirun -np 1 ~/envs/horovod/bin/python
scripts/tf_cnn_benchmarks/tf_cnn_benchmarks.py --model resnet101 --batch_size 64 --
variable_update horovod --data_dir /home/SSO/wps/imagenet_tf/IMAGENET-FULL/ --data_name imagenet
--num_batches=2000 | tee 1GPUs1nodes.log
------------------------------------------------------------------
total images/sec: 35.31
------------------------------------------------------------------

####(Case 2)#### 2 GPUs on 1 node
time ~/.openmpi/bin/mpirun -np 2 ~/envs/horovod/bin/python
scripts/tf_cnn_benchmarks/tf_cnn_benchmarks.py --model resnet101 --batch_size 64 --
variable_update horovod --data_dir /home/SSO/wps/imagenet_tf/IMAGENET-FULL/ --data_name imagenet
--num_batches=2000 | tee 2GPUs1nodes.log
------------------------------------------------------------------
total images/sec: 74.59
------------------------------------------------------------------

####(Case 1)#### 4 GPUs on 2 nodes
 time ~/.openmpi/bin/mpirun -np 4 --hostfile /home/SSO/wps/hostfile ~/envs/horovod/bin/python
scripts/tf_cnn_benchmarks/tf_cnn_benchmarks.py --model resnet101 --batch_size 64 --
variable_update horovod --data_dir /home/SSO/wps/imagenet_tf/IMAGENET-FULL/ --data_name imagenet
--num_batches=2000 | tee 4GPUs2nodes.log
------------------------------------------------------------------
total images/sec: 436.56
------------------------------------------------------------------
```

Notice how on 1 and 2 GPU nodes within the same machine I am just getting about 35 images per second on each node, while on a 2 machine 4 GPU node setup I get about 100 images/sec on each node.

I am not sure if I am able to explain this scaling. Anything that you think I am fundamentally missing or doing wrong.

**alsrgv** commented on Nov 4, 2017 — Collaborator

**@atinsood**, let's debug this.

1. What do you get if you just run `~/envs/horovod/bin/python scripts/tf_cnn_benchmarks/tf_cnn_benchmarks.py --model resnet101 --batch_size 64 --data_dir /home/SSO/wps/imagenet_tf/IMAGENET-FULL/ --data_name imagenet --num_batches=2000` (no Horovod)?
2. Can you paste full logs of those runs? You can use https://gist.github.com/ and put the links here.
3. What is the version of MPI?
4. Do you have NCCL installed? Which version?
5. Do you have RoCE / InfiniBand in your system?
6. What's the output of `find / 2>/dev/null | grep libucs.so`?

**atinsood** commented on Nov 4, 2017

What is the version of MPI?
~/.openmpi/bin/mpirun --version
mpirun (Open MPI) 3.0.0

Do you have NCCL installed? Which version?
Yes, but the current runs are without nccl. Also when installing horovod, I did not specify the all reduce to use nccl

Do you have RoCE / InfiniBand in your system?
No

What's the output of find / 2>/dev/null | grep libucs.so?
returned empty

Let me get you the other results as well

**atinsood** commented on Nov 4, 2017

**@alsrgv** also noticing on 8GPU 4node run with data that the jitter increases significantly starting from 0 to about 8.0 after about 200 iterations taking the number of images processed per second from about 90 on each GPU node to 70

**alsrgv** commented on Nov 5, 2017 — Collaborator

**@atinsood**, the dropoff may be related to how many images your CPUs can preprocess per second. Having CPUs do the allreduce takes away additional cycles that could otherwise be spent preprocessing images. We should see those things from the logs.

**atinsood** commented on Nov 5, 2017

**@alsrgv** what makes you think that the CPUs are doing the allreduce. I thought we were using nccl2 for all reduce which in my mind was implemented in GPU.

**alsrgv** commented on Nov 5, 2017 — Collaborator

**@atinsood**, you said "the current runs are without nccl. Also when installing horovod, I did not specify the all reduce to use nccl".

**atinsood** commented on Nov 5, 2017

**@alsrgv** here are the gists for when I ran the test on the node with native TF and with horovod

https://gist.github.com/atinsood/8ffcc05e487c159efff2e8f09ee8871d
https://gist.github.com/atinsood/5ff2bf149d54d358f06b63225e24879d

Also to your above question, I am running all the latest tests with horovod installed with `HOROVOD_GPU_ALLREDUCE=NCCL pip install --no-cache-dir horovod`

And goes without saying, appreciate your patience in helping out with this.

**atinsood** commented on Nov 5, 2017

also curious if you guys ran your tests on your local bare metal boxes, AWS or google cloud and did you guys did anything to ensure that the neighbors required for ring reduce approach were optimally placed close to each other.

**alsrgv** commented on Nov 5, 2017 • edited ▾ Collaborator

**@atinsood**, thanks for the logs!

Yeah, we ran the tests on local bare metal boxes. We give out names / IP addresses in racks and between racks sequentially, so we just order addresses in the `mpirun` command by name / IP address. This ensures that each rack will have one inflow and one outflow (per training session).

Back to your performance:

```
2017-11-05 00:27:29.531500: I tensorflow/core/kernels/shuffle_dataset_op.cc:110] Filling up
shuffle buffer (this may take a while): 468 of 10000
2017-11-05 00:27:29.654469: I tensorflow/core/kernels/shuffle_dataset_op.cc:110] Filling up
shuffle buffer (this may take a while): 465 of 10000
...
2017-11-05 00:30:39.841956: I tensorflow/core/kernels/shuffle_dataset_op.cc:110] Filling up
shuffle buffer (this may take a while): 9773 of 10000
2017-11-05 00:30:40.177912: I tensorflow/core/kernels/shuffle_dataset_op.cc:110] Filling up
shuffle buffer (this may take a while): 9913 of 10000
2017-11-05 00:30:41.565690: I tensorflow/core/kernels/shuffle_dataset_op.cc:121] Shuffle buffer
filled.
2017-11-05 00:30:43.967015: I tensorflow/core/kernels/shuffle_dataset_op.cc:121] Shuffle buffer
filled.
```

Since it took 3 min and 10 sec to fill up 10k image buffer, the preprocessing rate is ~53 img/sec, which is fairly low. What kind of CPUs do you have in these servers?

One suggestion I have is to play with `num_parallel_calls` setting in this file. Right now it spins `batch_size` threads per worker (64), it may be better to reduce it to `num_parallel_calls=16` or less.

What performance are you getting if you run the benchmark on synthetic data?

**atinsood** commented on Nov 5, 2017 • edited ▾

**@alsrgv** yep I can try tweaking that number and see what we get and here's the log for 2GPUs on 1 node with synthetic data `https://gist.github.com/atinsood/136692a008645d163738c59a473b7dc3`

Though I am curious if you can elaborate more on `This ensures that each rack will have one inflow and one outflow (per training session).` . For a particular run did your topology always included more than 1 rack or only once you got past a certain number of GPUs. I am just more curious about topology, when you plan to put this in public cloud, how are you planning to solve the problem of placing these nodes correctly to optimize the nodes with the fastest networking as each others direct neighbor.

I can cross check the CPUs at my end, though I am curious once the preprocessing is done, isn't it suppsd to be all GPU numbers, and how the preprocessing numbers and the images/sec processed by GPU training related. I am assuming the numbers you are reporting in your findings are GPU training numbers , something along the lines mentioned below, or am I doing this wrong.

Also from the logs, did anything strike out as incorrect to you, I am curious if I am running this correctly.

Let me run this on synthethic data and get you the numbers, but attaching the logs for 4 GPUs across a total of 2 nodes and you will notice how images processed on each GPU jumps to about 100 images/sec on each GPU. I am trying to understand what can be the possible reasoning for this jump, ideas?

https://gist.github.com/atinsood/c096f27b149468c682d88716fed112a7

```
2000    images/sec: 106.8 +/- 0.1 (jitter = 3.3)        8.701
----------------------------------------------------------------
total images/sec: 2350.14
----------------------------------------------------------------```
```

**atinsood** commented on Nov 5, 2017

Also if you know off the top of.your head how to.run this in a distributed tensor flow native dkst across gpus across nodes I can use that to.cpate against native tf.distribution on my setup

**alsrgv** commented on Nov 5, 2017 Collaborator

**@atinsood**,

Yes, we try to tight-package jobs inside the server or a rack, but even if they spread multiple racks each rack will only have one inflow and one outflow in the ring topology, so most of the traffic is contained.

In AWS, you can request similar topology by having a placement group. I tried running Horovod on AWS P3 and found that their 25 Gbit networking is a bottleneck and requires some tricks to be fully utilized. Are you planning to run Horovod on AWS?

**Good news**, I think I figured out your CPU problem. Apparently, Open MPI v3 has a more aggressive binding policy, so if you give it `-np 1` it thinks it's entitled to use only one CPU core. Can you add `--bind-to none` to your `mpirun`? That should solve your preprocessing problem.

You can find an example of running parameter server distributed TensorFlow here. I don't have commands for benchmarks handy, but it should be just running all those workers and parameter servers with those additional parameters.

**alsrgv** added the update docs label on Nov 5, 2017

**alsrgv** referenced this issue on Nov 6, 2017

**Various README updates** #76 Merged

**alsrgv** removed the update docs label on Nov 6, 2017

**atinsood** commented on Nov 7, 2017 • edited

**@alsrgv** so yep that tweak you suggested in openmpi argumet did the trick and I was able to get the numbers which linearly scale. appreciate the help

There are a few things we are looking at:

0. I think I did pick up your comment on one of TF user mails about the desire to run this in mesos. We run a k8s cluster and been trying to investigate what's a good way to have these running on a k8s cluster and how to deploy this alongside with the application code. My mental picture for the same is to run a separate GPU cluster and a separate application cluster and simply submit the model training jobs to the GPU cluster from the application cluster, but I'd be interested in hearing how you guys are doing this in a container world.

1. Is there more debug logging we can use to collect the communication between the nodes. we are trying to pick the right topology, so would be interesting if there are stats are how long the nodes are taking to communicate, which one's being the slowest and so on, any stats around convergence and so on

2. I was able to get the ps/worker setup going with native tf distributed
`python scripts/tf_cnn_benchmarks/tf_cnn_benchmarks.py --model resnet101 --batch_size 64 --num_batches=2000 --num_gpus 2 --data_dir /home/SSO/wps/imagenet_tf/IMAGENET-FULL/ --data_name imagenet --task_index 0 --job_name ps --ps_hosts <host:ip> --worker_hosts=<4hosts:ips separated by commas>`
though I am yet to figure out how to run with other variable_updates
`https://github.com/tensorflow/benchmarks/blob/master/scripts/tf_cnn_benchmarks/benchmark_cnn.py#L272`

**alsrgv** commented on Nov 7, 2017 — Collaborator

That's great! Glad it works for you. To answer your questions:

1. We just use Mesos and a scheduler on top of it (e.g. Aurora). Aurora spins up few containers (one container per host), which have TensorFlow, MPI, Horovod, NCCL, ssh and user code installed. All the containers also have per-job ssh keypair, so they can ssh onto each other. Then, we simply execute `mpirun` on the first container and it starts the training on other containers via ssh. Here are the slides about it.

2. You can use Horovod Timeline to analyze the allreduce performance.

3. I didn't have a chance to try those other modes yet but I'm curious to hear about people's experience.

**atinsood** commented on Nov 14, 2017

**@alsrgv** have a quick question, I know little about this, but I am curious if this impacts horovod in any fashion https://research.googleblog.com/2017/02/announcing-tensorflow-fold-deep.html

I wonder if there is anything that horovod does so far to leverage the static nature of graph of TF

**alsrgv** commented on Nov 14, 2017 — Collaborator

**@atinsood**, haven't tried TensorFlow Fold, but was planning to try Eager execution.

In theory, it should just work - as long as graph nodes have the same name on all workers.

**atinsood** commented on Nov 14, 2017

argh!! yep Eager execution was what I was actually referring to, did not realize I had the wrong link in. If things work without a change then great.

**atinsood** commented on Nov 15, 2017

**@alsrgv** similar question, out of curiosity does the ordering of the code matter. I am thinking that broadly do

```
config = tf.ConfigProto()
config.gpu_options.allow_growth = True
config.gpu_options.visible_device_list = str(hvd.local_rank()```

as the first few statements, and can then add optimizer and broadcasting hooks later.
```

**alsrgv** commented on Nov 15, 2017 — Collaborator

**@atinsood**, you need to have that config whenever you're creating `tf.Session()`, `tf.train.MonitoredTrainingSession()` or any other kind of session. Apart from that, it doesn't matter.

**atinsood** commented on Nov 20, 2017 • edited

**@alsrgv** well this is embarassing and I guess pushing you too much, but have you seen issues with running `/openmpi/bin/mpirun -np 1 ~/envs/horovod/bin/python tensorflow_word2vec.py` in docker/kubernetes.

I am able to run `~/envs/horovod/bin/python tensorflow_word2vec.py` without any issues but the openmpi python just gets stuck with nothing in the logs.

I was able to run `openmpi echo hello` correctly, so slightly confused what I might be missing.

may be its the allow-run-as-root that might be messing things up. I don't see a clear error message. the program just get stuck. `ps -ef | grep python` shows the program started but I don't see any logs

**alsrgv** commented on Nov 20, 2017 — Collaborator

**@atinsood**, no worries! What's the version of `openmpi` that ships with kubernetes?

**atinsood** commented on Nov 20, 2017 • edited

**@alsrgv**
this is the openmpi that I use `https://www.open-mpi.org/software/ompi/v3.0/downloads/openmpi-3.0.0.tar.gz`

and this is the docker file https://gist.github.com/atinsood/535104d57b8e35b6fcc7a480a0a3ddb2

I don't think there's anything specific to k8s here. I'd be curious if you are more than doing using the simple docker file I pasted above to get things going

**alsrgv** commented on Nov 20, 2017 — Collaborator

**@atinsood**, thanks for details. Helps me help you :-)

1. Can you add `--enable-mpirun-prefix-by-default` to `./configure` on line https://gist.github.com/atinsood/535104d57b8e35b6fcc7a480a0a3ddb2#file-dockerfile-L89. Unless you do that, you have to add `--prefix=/openmpi` on every `mpirun`.
2. Can you replace line https://gist.github.com/atinsood/535104d57b8e35b6fcc7a480a0a3ddb2#file-dockerfile-L98 with:
`HOROVOD_GPU_ALLREDUCE=NCCL pip install --upgrade --no-cache-dir horovod`. This makes sure Horovod actually uses NCCL.
3. Can you paste the whole output of `/openmpi/bin/mpirun -np 1 ~/envs/horovod/bin/python tensorflow_word2vec.py`?

**atinsood** commented on Nov 21, 2017

ha, I think I found the issue. something is messed up in that dockerfile (I will try to clean up and debug), but the good thing is I was able to piggy back on top of the tf docker file, get nccl2 and horovod installed and that did the trick.

I am curious though, do you guys have openmpi installed per container, or on the underlying nodes . Also is the openmpi only used for configuration or more than that. To me coming from a non HPC background, it seems like there might be better ways to do configuration rather than openmpi. There's definitely convenience with openmpi but not sure how your experience has been with openmpi in the container world.

**alsrgv** commented on Nov 21, 2017 Collaborator

**@atinsood**, we have Open MPI in the container. It's used for orchestration - negotiate which tensors are ready to be reduced and orchestrate the reduction. This is necessary since TensorFlow can execute multiple Graph ops in parallel, and so the order of tensors to be reduced is non-deterministic.

**atinsood** commented on Dec 6, 2017

**@alsrgv** just curious if you have thought through other options on doing this besides openmpi or you happy with what openmpi supports. Also curious how does recovery works, when say openmpi spins process across boxes, the process fails or there is a network partition for a short duration.
Curious if these cases will make any difference on running openmpi

**alsrgv** commented on Dec 6, 2017 Collaborator

**@atinsood**, at the moment we're pretty happy with MPI, but we're open to ideas for orchestration layer that would be easier to set up. With MPI, if any process fails, MPI will clean up all the processes. In our environment, we have simple retry logic to restart failed MPI jobs if we detect failures such as host failure. We don't usually have network partitioning within the datacenter, but host failures happen due to maintenance.

1

**atinsood** commented on Dec 14, 2017

**@alsrgv** yep, looking into things here.

btw did anything change with the latest pip package. I keep running into

```
  Running setup.py install for horovod: started
    Running setup.py install for horovod: finished with status 'error'
    Complete output from command /usr/bin/python -u -c "import setuptools,
tokenize;__file__='/tmp/pip-build-l77E8v/horovod/setup.py';f=getattr(tokenize, 'open', open)
(__file__);code=f.read().replace('\r\n', '\n');f.close();exec(compile(code, __file__, 'exec'))"
install --record /tmp/pip-W6cERl-record/install-record.txt --single-version-externally-managed --
compile:
    running install
    running build
    running build_py
    creating build
    creating build/lib.linux-x86_64-2.7
    creating build/lib.linux-x86_64-2.7/horovod
    copying horovod/__init__.py -> build/lib.linux-x86_64-2.7/horovod
    creating build/lib.linux-x86_64-2.7/horovod/keras
    copying horovod/keras/__init__.py -> build/lib.linux-x86_64-2.7/horovod/keras
    copying horovod/keras/callbacks.py -> build/lib.linux-x86_64-2.7/horovod/keras
    creating build/lib.linux-x86_64-2.7/horovod/tensorflow
    copying horovod/tensorflow/__init__.py -> build/lib.linux-x86_64-2.7/horovod/tensorflow
    copying horovod/tensorflow/mpi_ops.py -> build/lib.linux-x86_64-2.7/horovod/tensorflow
    copying horovod/tensorflow/mpi_ops_test.py -> build/lib.linux-x86_64-2.7/horovod/tensorflow
    running build_ext

    ----------------------------------------
Command "/usr/bin/python -u -c "import setuptools, tokenize;__file__='/tmp/pip-build-
```

```
l77E8v/horovod/setup.py';f=getattr(tokenize, 'open', open)
(__file__);code=f.read().replace('\r\n', '\n');f.close();exec(compile(code, __file__, 'exec'))"
install --record /tmp/pip-W6cERl-record/install-record.txt --single-version-externally-managed --
compile" failed with error code -11 in /tmp/pip-build-l77E8v/horovod/```
```

**atinsood** commented on Dec 14, 2017

```
Inconsistency detected by ld.so: get-dynamic-info.h: 129: elf_get_dynamic_info: Assertion
`info[DT_RELAENT]->d_un.d_val == sizeof (ElfW(Rela))' failed!
```

**alsrgv** commented on Dec 14, 2017 Collaborator

**@atinsood**, I doubt this error is related to Horovod. After quick googling, I found issues along the lines of Linuxbrew/legacy-linuxbrew#126.

**atinsood** commented on Dec 14, 2017 • edited

**@alsrgv** interesting, let me try to take a look https://gist.github.com/atinsood/11f41bd8f7fd0aca8e8546e32e191cf3 is what I end up using for the docker file. Pretty sure this used to work, may be some of the internal dependency changed

**alsrgv** commented on Dec 14, 2017 Collaborator

**@atinsood**, looks reasonable. What if you try `tensorflow/tensorflow:1.4.0-gpu` instead of devel-gpu? Maybe the official release will be more stable.

**atinsood** commented on Dec 15, 2017

yep yep, argh!!! that was it. switching to tensorflow-gpu fixed things. thank you thank you, appreciate you taking the time and looking into it :)

1

**atinsood** commented on Apr 3

**@alsrgv** I wanted to have done this earlier, but just had a little baby boy so that's keeping me really busy. I would like to thank you for helping out with getting horovod going and your patience in answering my questions around horovod. I am a part of IBM Watson and we have been able to integrate Horovod natively as a part of IBM Watson Studio which offers you Deep Learning as a Service. I believe we see the value of all reduce algorithm in doing distributed and especially the fact how horovod made it so easy to use distributed using all reduce in existing code.

Here's a blog post covering that and here's the link to get to IBM Watson Studio

There is a guide that we are in the works to show how to run horovod with IBM Watson Studio similar to what you have here . We are still offering the service as beta, along with 2 more flavors, native distributed TF and our own DDL. [Here's] (https://datascience.ibm.com/docs/content/analyze-data/ml_dlaas_distributed.html?context=analytics) the guide to get horovod going, broadly the intent is to show you you can bring in horovod code right from your examples and be able to run it in IBM Watson Studio without the need to worry about MPI and zero code changes required. Here's a detailed example that you can run in Watson Studio that can run the mnist example that you have provided.

Also this is a ticket that some team members of mine are working on to find a good way of doing inferencing

Again, I really really want to thank the effort you had put into horovod and the help you have provided as a part of tickets and esp. this ticket in helping me understanding horovod.

I'd really love it for you to try out horovod on IBM Watson Studio and please feel free to reach out as a part of this ticket or over email asood@us.ibm.com if you have feedback and/or any way we can coordinate more on this.

1

gtbai referenced this issue on Jun 16

**Questions about Running Horovod on Aurora** #315 Open

alsrgv removed the tracking label on Aug 3

alsrgv closed this on Aug 3