

PyTorch Lightning Collective Discussion

Status: reviewed and agreed by lightning folks ([link](#)) and FB folks

Problem

The use of collective functions are scattered everywhere in lightning, which introduced quite a few issues:

- a) Device specific code is mixed with core APIs (supposed to be device agnostic)
- b) Usage is quite inconsistent
- c) Redundancies: similar API exists in many places.

Thanks to Ananth for the great background:

<https://github.com/PyTorchLightning/pytorch-lightning/issues/7534>

Current Status (As of Aug 2021)

1. accelerator.py

Accelerator encapsulates a `training_type_plugin` which implements collective functions. The `broadcast`, `barrier`, and `all_gather` functions inside this class are just pass through.

```
class Accelerator
-   Init: training_type_plugin : TrainingTypePlugin
-   Methods: wrapper function delegates to training_type_plugin
            - broadcast --
            - barrier
            - all_gather
```

2. training_type_plugin.py

TrainingTypePlugin defines the interface of ~5 collective functions

```
class TrainingTypePlugin
-   Method
            - abstract reduce
            - abstract barrier
            - abstract broadcast
            - abstract all_gather
            - abstract reduce_boolean_decision
```

3. Subclass of TrainingTypePlugin

DDPPlugin ← ParallelPlugin ← TrainingTypePlugin

Leverage `distributed/dist.py` - `LightningDistributed` to support `broadcast` and `barrier`. Note that the location of `LightningDistributed` should be moved from a generic to a device specific place.

- Init

- Instantiate [LightningDistributed](#)
- Method
 - override barrier → PT specific
 - override broadcast → use LightningDistributed.broadcast
 - not implemented reduce, all_gather, reduce_boolean_decision

HorovodPlugin ← ParallelPlugin ← TrainingTypePlugin

class HorovodPlugin

- Method
 - override barrier → horovod specific
 - override broadcast → horovod specific
 - override reduce → horovod specific
 - override all_gather → horovod specific

TPUSpawnPlugin ← DDPSpawnPlugin ← ... ← TrainingTypePlugin

class TPUSpawnPlugin

- Method
 - override barrier → TPU specific
 - override broadcast → TPU specific
 - override reduce_boolean_decision → TPU specific
 - override reduce → TPU specific

Proposal

Device specific collectives are coupled with existing TrainingTypePlugin subclasses which seems fine. The main issues: 1) No standard API; 2) Inconsistency and Redundancy; 3) Unorganized code, distributed codes go into many places; and 4) Lack of modularization - core APIs, utils are mingled which makes existing code quite long and hard to understand and maintain. Our proposal is about API standardization, modularization as follows:

1. Introduce distributed/collective.py


```
class Collective
    - Method
        - abstract broadcast
        - abstract barrier
        - abstract all_reduce
        - abstract all_gather
        - more from here when needed
        -
        - setup(rank, world_size) {...}
        - teardown() {...}
```
2. Collective subclasses maintain device specific impls
 - a. New three subclasses:

- i. class SingleDevice ← collective : needed for API unification only, impl is pass only
 - ii. class PyTorchCollectiveImpl ← Collective
 - iii. class HorovodCollectiveImpl ← Collective
 - iv. class TpuCollectiveImpl ← Collective
- b. impl of base Collective.{collective_function} comes from the corresponding plugin (below)
- 3. Revise TrainingTypePlugin to use Collective interface
 - class TrainingTypePlugin
 - Method
 - Init(...):
 - self.collective = None
 - @property collective { return self.collective }
 - Deprecate abstract reduce
 - Deprecate abstract barrier
 - Deprecate abstract broadcast
 - Deprecate abstract all_gather
 - Deprecate abstract reduce_boolean_decision
- 4. class Accelerator
 - a. **Keep** those wrapper functions: reduce, barrier, broadcast, all_gather, and reduce_boolean_decisions since they are just proxy.
 - b. Modify impl from training_type_plugin.{collective_func} → training_type_plugin.collective.{collective_func} e.g., training_type_plugin.collective.broadcast(...)
- 5. Instantiate platform specific collectives in platform specific plugin
 - a. DDPPlugin:
 - i. Init: self.collective = PyTorchCollectiveImpl(...)
 - b. HorovodPlugin
 - i. Init: self.collective = HorovodCollectiveImpl(..)
 - c. TPUSpawnPlugin
 - i. Init: self.collective = TPUSpawnPlugin(...)
- 6. Deprecate existing collective impl for all Plugin subclasses, e.g.,
 - a. DDPPlugin
 - i. @deprecate broadcast
- 7. All distributed related utils moving to distributed/
 - a. utilities/distributed.py -- only broadcast is supported
 - b. core/lightning.py (all_gather) -- move impl to accelerator. Change references to accelerator directly
 - c. ?
- 8. Update ChangeLog