Deploying Ray cluster of HAL (Beta)

Notice

This is a beta version of deployment. If you meet any problems, please go to HAL Slack channel for help.

Always remember to tear down your Ray cluster after using!

Introduction

This is the instruction for setting up a Ray cluster with autoscaling capability on HAL cluster. With the autoscaling functionality, the Ray cluster tries to add more "nodes" to the existing Ray cluster by submitting new SLURM jobs, and disconnect "nodes" by cancelling SLURM jobs when idle.

Deployment Instructions

The deployment process includes getting the ray library, modify the ray library to support autoscaler on HAL, and configure launch-specific parameters.

Get the Ray library into your private environment

Since Ray is not directly available on HAL and cannot be installed directly from pip, we need to do modification to the ray library based on the one provided by openCE.

To do so first load opence/1.6.1, using:

```
module load opence/1.6.1
```

Then clone this module to your own environment using this command:

```
conda create --name <env_name> --clone opence/1.6.1
```

You can use any environment name you like (make sure you remember it). This step can take about 30 minutes.

The path to ray after you've cloned the environment looks something like: /home/<username>/.conda/envs/<env_name>/lib/python3.9/site-packages/ray

To activate the environment, use

```
conda activate <env_name>
```

Configure bash to load modules automatically

Ray autoscaler requires conda to be accessible when a shell is opened. To do so, you need to modify the file ~/.bashrc, which is automatically executed every time a bash shell is opened.

1. Run

```
vi ~/.bashrc
```

This will open the bashrc file using vim.

2. Hit the 'i' key in order to edit the file. Add these lines under "#User specific environment"

```
module load opence/1.6.1
```

3. Hit the escape key to get out of edit mode. Enter ':qw' to save and quit

Deploy the Ray-SLURM autoscaler module into Ray library

Download the autoscaler code and deployment script from https://github.com/TingkaiLiu/Ray-SLURM-autoscaler by running:

```
git clone https://github.com/TingkaiLiu/Ray-SLURM-autoscaler.git
```

Several changes on the deployment script is needed specifically for the HAL cluster

1. Open deploy.py and change the following

- Set the RAY_PATH to be the path to your Ray library: /home/<username>/.conda/envs/<env_name>/lib/python3.9/site-packages/ray

- Set the SLURM_IP_LOOKUP table to be

```
SLURM_IP_LOOKUP = """{
    "hal01" : "192.168.20.1",
    "hal02" : "192.168.20.2",
    "hal03" : "192.168.20.3",
    "hal04" : "192.168.20.4",
    "hal05" : "192.168.20.5",
    "hal06" : "192.168.20.6",
    "hal07" : "192.168.20.7",
    "hal08" : "192.168.20.8",
    "hal09" : "192.168.20.9",
    "hal10" : "192.168.20.10",
    "hal11" : "192.168.20.11",
    "hal12" : "192.168.20.12",
    "hal13" : "192.168.20.13",
    "hal14" : "192.168.20.14",
    "hal15" : "192.168.20.15",
    "hal16" : "192.168.20.16",
}"""
```

- Change HEAD and WORKER CPUS/GPUS according to the partition you want to use. See the "Available queues" under "Native SLURM Style" section at Job management with SLURM. If you want to run the Ray head node outside SLURM, set the CPUS/GPUS of the head node to be 0.

2. Change slurm/worker.slurm

- Change line 4 "#SBATCH --gpus-per-task=[_DEPLOY_WORKER_GPUS_]" to "#SBATCH --gres=gpu:[_DEPLOY_WORKER_GPUS_]"
- Under line "set -x" add "SLURM_GPUS_PER_TASK="[_DEPLOY_WORKER_GPUS_]"

3. Change slurm/head.slurm

- Change line 4 "#SBATCH --gpus-per-task=[_DEPLOY_HEAD_GPUS_]" to "#SBATCH --gres=gpu:[_DEPLOY_HEAD_GPUS_]"
- Under line "set -x" add "SLURM_GPUS_PER_TASK="[_DEPLOY_HEAD_GPUS_]"

4. Run deploy.py

```
python3 deploy.py
```

This should generate the ray-slurm.yaml file for cluster launching.

Up to this point. The Ray autoscaler should be already enabled.

Configuration for specific cluster launch

After the module is deployment, you may want different configuration every time you launch a Ray cluster. For example, you may want to change the maximum number of nodes in your cluster for different workloads. Those launch-specific changes requires only changes to the cluster config yaml file.

For now:

- Change "init_command" on line 43 and 60 to be activating your own environment
- Set "under_slurm:" on line 37 to be 0 or 1 based on your need. (See the Github Doc for explanation)
- If you don't have a reservation, comment out lines 45 and 62 -> lines that say - "#SBATCH --reservation=username" (make sure the comment lines up correctly with rest of code)
- Change line 46 and 63 according to the partition you try to use

To start the Ray cluster, run

```
ray up ray-slurm.yaml --no-config-cache
```

If under_slurm is set to be 1, It is required that there is at least one idle node. Otherwise, the launching process keeps retrying until there is an idle node.

If you forced-terminated the launching process, run "ray down ray-slurm.yaml" to perform garbage collection.

If the "ray up" command runs successfully, the Ray cluster with autoscaling functionality should be started at this point. To connect to the Ray cluster in your Python code,

- If you launch head outside SLURM (under_slurm = 0), use

```
ray.init(address="192.168.20.203:<gcs_port>", redis_password="<The password generated at start time>")
```

- If you launch head inside SLURM (under_slurm = 1), you need to find the head node ip from the print out message and use Ray client to connect to it

```
ray.init(address="ray://<head_ip>:<gcs_port>", redis_password="<The password generated at start time>")
```

Always remember to tear down your Ray cluster after using!

To tear down the Ray cluster, run

```
ray down ray-slurm.yaml
```

Testing

You can check whether the Ray cluster is autoscaling when you launch heavy workload with Ray by the following:

- Check whether there are output message starting with "(scheduler"
- Run "squeue -u <username>" to see whether new SLURM jobs are submitted automatically under your username
- Check the .out file produced by SLURM

Acknowledgment

This document is contributed by Tingkai Liu, Will Tegge, and Arnav Mehta