ASC Student Supercomputer Challenge 2022

Preliminary Round Notification

Dear ASC22 Teams:

Welcome to the 2022 ASC Student Supercomputer Challenge: ASC22!

ASC Student Supercomputer Challenge now on its 10th year of continuous success. It has become the world's largest supercomputing hackathon striving to foster the next generation of young talents, to inspire exploration, innovation and collaboration in Supercomputing and AI. After the ASC22 registration was kicked off at SC21 on November 15, 2021, we received enormous interest from hundreds of registered teams, and now the competition is moving on the next phase of Preliminary!

Preliminary Round

In the preliminary round, the ASC22 Committee is looking forward for each team to do their best in accomplishing all of the tasks approved for this competition, and submit the proposal documentation, which include the cluster design details, source code optimization approaches, and output files. The ASC22 evaluation committee will review the proposals in English.

Submission Guidelines

All ASC22 teams are expected to upload before 24:00, of March 4, 2022 (UTC/GMT +8:00), the following items to ASC22 official website: http://asc-events.org/ASC22/.

a) The proposal document named by combining the university or college name and the contact person: i.e: ABCUniversity_John_Doe. The document should be in either .docx or .pdf file format.

b) All the additional information should be compressed using ZIP, or and other tools, into one file using the same naming convention: ABC University_John_Doe. The compressed file should include at least four folders, per the requirements detailed in Appendix A.

- Output files of HPL
- Output files of HPCG
- Required files of Yuan Large Language Model Challenge
- Required files of DeePMD Challenge

The confirmation email will be sent back shortly after all the above information is received. For any further inquiries, please contact the ASC committee via:

Technical Support: techsupport@asc-events.org

General Information: info@asc-events.org

Press: media@asc-events.org

Wish you all the best of luck in your ASC22 journey!

ASC22 Committee

Appendix A:

Proposal Requirements

I. Introduction of the university department activities in supercomputing (5 points)

1. Supercomputing-related hardware and software platforms.
2. Supercomputing-related courses, trainings, and interest groups.
3. Supercomputing-related research and applications.
4. A brief description of the key achievements on supercomputing research, no more than 2 items.

II. Team introduction (5 points)

1. Brief description of the team setup.
2. Introduction and the photo of each member, including group photos of the team.
3. Team's motto or catch-phrase.

III. Technical proposal requirements (90 points)

b) Specify the system's software and hardware configuration and interconnection. Describe the power consumption, evaluate the performance, analyze the advantages and disadvantages of your proposed architecture.

c) The components listed in the table below are provided for reference only. They are based on the Inspur NF5280M6 server, which supports up to 2 GPUs.

Item	Name	Configuration
Server	Inspur NF5280M6	CPU: Intel IceLake 8358 *2 Memory: 32G x 16，DDR4，3200Mhz Hard disk: 480G SSD SATA x 1
HCA card	HDR	InfiniBand Mellanox ConnectX®-6 HDR
Switch	GbE switch	10/100/1000Mb/s，24 ports Ethernet switch
	HDR-IB switch	Mellanox Quantum (TM) HDR InfiniBand Switch, 40 QSFP56 ports, 2 Power Supplies (AC), unmanaged, standard depth, P2C airflow, Rail Kit, RoHS6
Cable	Gigabit CAT6 cables	CAT6 copper cable, blue, 3m
	InfiniBand cable	InfiniBand HDR copper cable, QSFP port, compatible with the InfiniBand switch in use.

◆ The hardware configuration in the ASC22 competition finals may be different from the table above.

2. HPL and HPCG (15 points)

The proposal should include descriptions of the software environment, (operating system, complier, math library, MPI software, software version, etc.), the performance optimization and testing methods, performance measurement, problem and solution analysis, etc. In-depth analysis on HPL, HPCG algorithms and the respective source codes would be a plus.

Download the HPL software at: http://www.netlib.org/benchmark/hpl/.
Download the HPCG software at: https://github.com/hpcg-benchmark/hpcg

It is recommended to run verification and optimization of HPL and HPCG benchmarks on x86 Xeon CPU and Tesla GPU platforms. If other hardware platforms are used, you are welcomed to submit the related analysis and results that demonstrate adequate performance.

3. Yuan Large Language Model Challenge (30 points)

Task Description:

GPT-3, a large Language Model (LM) with 175 billion parameters for Natural Language Processing (NLP), triggered a whole new trend in AI. GPT-3 can be widely used for multiple NLP applications, such as reading comprehension, question answering, textual deduction and so on. Since then, lots of large LM were released.

Training large Language Models with billions or trillions of parameters is difficult, because not only require a lot of computing resources, but also require sophisticated training approaches to handle such massive model parameters, which may exceed the memory limitation of modern processors. So some special training methodologies, like model parallel and pipeline parallel, should be used.
Yuan 1.0 is one of the largest singleton Chinese Language Model. It was trained on a new Chinese dataset of 5TB high-quality text that was extracted from 850TB raw data from Internet. The architecture of Yuan 1.0 was designed by integrating its intrinsic model structure with crucial factors that drastically affects the performance of large-scale distributed training. Yuan 1.0 was trained on a cluster with 2128 GPUs for about 16 days. During training the actual stable performance achieved 45% of the theoretical peak. The source code of Yuan 1.0 can be download from https://github.com/Shawn-Inspur/Yuan-1.0

A 100GB dataset of Yuan will be provided for the preliminary round of ASC22. Its raw dataset was collected from Common Crawl, and was processed with a Massive Data Filtering System (MDFS). The ASC22 teams should note that this dataset is made available based on the license agreement with Inspur, which ASC Committee deems to be read and accepted from the participants by default.

In the preliminary, the participants should train a massive LM like Yuan 1.0 using the dataset provided by ASC22 Committee, with model structure parameters specified in the following table:

The objective of Yuan is to minimize the loss value shown below:

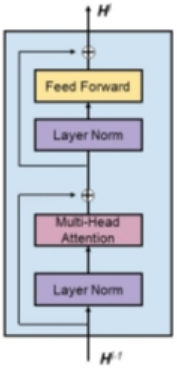
$$Loss = \sum_{k=1}^{n} -\log P(x_k | x_1, x_2, ..., x_{k-1})$$

$$H^0 = E + P$$

$$H^l = Transformer\_block(H^{l-1}), 1 \ll l \ll L$$

$$P(x_k | x_1, x_2, ..., x_{k-1}) = softmax(W_v H_k^L)|_{x_k}$$

Loss is loss function, *n* is the training sequence length, *E* is word embedding matrix and ***P*** is position embedding matrix, L is the number of Transformer blocks. The architecture of Transformer_block is illustrated in the following figure.



There are two layers with weights $W_1 \in R^{d \times 4d}$ and $W_2 \in R^{4d \times d}$ in the FFN block.

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

The model must be built with PyTorch. It is not allowed to use other DL frameworks. It's encouraged to train the model with HuggingFace, DeepSpeed or Megatron-LM. Other distributed training frameworks based on PyTorch are also allowed. We encourage teams to optimize training performance based on open source frameworks. The participating teams need to train the model for 1 billion tokens, achieve the final loss converge of less than 7.0, and the reach the fastest training speed possible.

Dataset

The dataset could be downloaded from:

Baidu Disk: https://pan.baidu.com/s/1zQgKjf1tafzw__COaufyPA (extract code: 1jjz)

Microsoft OneDrive: https://1drv.ms/f/s!Ar_0HIDyftZTsFkbM8eQFtquk4ZH

The script for the pre-processing of raw data and vocab file can be downloaded from the following link: Github: https://github.com/Shawn-Inspur/Yuan-1.0

Result Submission

For both preliminary and final rounds, each team should also submit a folder containing training log, tensorboard log files, the loss curve and source code that could be used to reproduce the training process.

Folder Name	Contents
Yuan	Root directory
Yuan/Log	Pretraining log file
Yuan/tensorboard	Tensorboard log file
Yuan/model	Language model pretraining script source code. If you don't use the script for the pre-processing given by ASC22 Committee, then the pre-process script also needs to be submitted in this folder

Evaluation

During the scoring process, the ASC22 Committee will first pay attention to the training speed realized and optimization strategy.

2. The participants must submit all the files required in the above Yuan directory, otherwise the score of this part is 0.
3. The submitted pre training log file needs to include the following contents:
   a) All hyper parameters, which including model structural parameters, random seed, batch size, learning rate and so on.
   b) The start and end time of model training
   c) Information of each iteration step, including batch_size, learning_rate, loss and time cost during each iteration
4. The participants should provide all the details of model implementation process and optimization method, which will be considered by ASC22 committee as the main basis for scoring.

Training Framework and Baseline Code

The ASC22 committee will not supply any baseline code for this task. The participants should build their deep learning network based on public resources.

Hardware Requirement

It is highly recommended to run the training code on a multi-GPU system with NVLink- NVSwitch.

4. The DeePMD Challenge (30 points)

Task Description:

Molecular dynamics (MD) is a computer simulation method for analyzing the physical movements of atoms and molecules. The atoms and molecules interaction are studied for a given period, creating a picture of the dynamic "evolution" of the system in time.

For a successful MD computer simulation, must compute the potential energy and interatomic forces of given systems, followed by other methods for dynamic "evolution" update. However, representing the inter-atomic potential energy surface (PES), both accurately and efficiently, is one of the most challenging problems in molecular modeling. Traditional approaches have either resorted in direct application of quantum mechanics models, - such as density functional theory (DFT) models, or empirical construction of potential atomic models, - such as empirical force fields (EFF) based. The former approach is severely limited by the size of the system that can be handled, while the latter class of methods are limited by the accuracy and the transferability of the model. This dilemma has confronted the molecular modeling community for long time.

In recent years, machine learning (ML) methods tackled this classical problem and a large body of work has been published in this area. One such model, Deep Potential Molecular Dynamics (DeePMD), has demonstrated achieving an accuracy comparable to AIMD, and an efficiency close to EFF based MD, - which pushes the limit of molecular dynamics with ab-initio to 100 million atoms.

DeePMD-kit is a realization of DeePMD written in Python/C++, designed to minimize the effort required to build deep learning based model of interatomic potential energy and force field and to perform molecular dynamics simulation. This brings new hope to addressing the accuracy-versus-efficiency dilemma in molecular simulations. Applications of DeePMD-kit span from finite molecules to extended systems, and from metallic systems to chemically bonded systems.

Reference

[1] Zhang, L., et al. (2018). "End-to-end Symmetry Preserving Inter-Atomic Potential Energy Model for Finite and Extended Systems."

[2] Wang, H., et al. (2018). "DeePMD-kit: A deep learning package for many-body potential energy representation and molecular dynamics." Computer Physics Communications 228: 178-184.

[3] Jia, W., et al. (2020). "Pushing the limit of molecular dynamics with ab initio accuracy to 100 million atoms with machine learning."

For this task, the participants need to train DeePMD models with DeePMD-kit package on given datasets, and then make improvements and analysis on the training procedure. More specifically, the task can be split into 3 steps:

1. Download the source code of DeePMD-kit, complete the compilation and installation. Then train two models on the given systems, making their baselines optimized respectively.
2. Dive into the implementation of DeePMD-kit code for training, and make improvement to speed up the training procedure on GPUs.
3. Provide analysis on the improvement and submit the modified code with a detailed report, containing the understanding of DeePMD-kit, the specific optimization method and improvement.

More details and requirements for this task are listed below：

Compile and install DeePMD-kit.

In order to compile and install the training code of DeePMD-kit, you may refer to the following manual: https://github.com/deepmodeling/deepmd-kit/blob/devel/doc/install/install-from-source.md#install-the-python-interface

Start training on example dataset.

Once you have installed the deepmd-kit python interface, you can easily start the training by simply running the following command:

```
Bash
cd $deepmd_source_dir/examples/water/se_e2_a
dp train input.json
```

After a few seconds, you should get a result similar to:

```
Bash
DEEPMD INFO    batch     100 training time 3.61 s, testing time 0.02 s
DEEPMD INFO    batch     200 training time 1.83 s, testing time 0.02 s
```

Note that training time represents the total time of 100 training steps.

Brief introduction to data format

The example water dataset in DeePMD-kit format can be found in $deepmd_source_dir/examples/water/data/data_0, which contains several necessary inputs for training models. The model takes coordinates of each atoms (**set.000/coord.npy**, in numpy format, shape : **[num_of_samples, num_of_atoms_per_sample * 3]**), atomic type (**type_map.raw** mapping the index to specific element, **type.raw** mapping atomic indexes in each sample to corresponding element) and box size of each sample (set.000/box.npy, in numpy format, shape : [**num_of_samples, 3 * 3**]) as inputs, and predicts the corresponding energy of samples (**set.000/energy.npy**, in numpy format, shape : **[num_of_samples,]**) and atomic forces (**set.000/force.npy**, in numpy format, shape : **[num_of_samples, num_of_atoms_per_sample * 3]**).

Evaluation

1. Program

All teams are required to use the ASC22 branch of DeePMD-kit:

```
Nginx
git clone --recursive https://github.com/deepmodeling/deepmd-kit.git
```

2. Dataset

Our community can use DeePMD-kit for various simulations, including the material simulation, phase changing, chemical reactions and so many more. For the sake of simplicity, we have selected the following three most representative simulations as benchmark systems, for Water, Copper and Al-Cu-Mg ternary alloy. **(No other datasets are allowed).**

The datasets could be downloaded from

Baidu Disk: https://pan.baidu.com/s/1_Qs5T1--VhshB1rnv_1COg;

password:nsi0 (asc-water.zip)

https://pan.baidu.com/s/1M2OqalfrRtLBJp2cPlmhBg;

password:wrq0 (asc-mgalcu.zip)

https://pan.baidu.com/s/1l1YSnUlSkY4ZCI6resGBPA;

password:bp9a (asc-copper.zip)

Google drive:

https://drive.google.com/file/d/16QKrvw4Fylq7dvwa0t9hI_SW4bKeHPjy/view?usp=sharing

3. Training script

Taking water as an example, the following contents can be found after opening the data set:

```
Bash
data_sets input.json
```

JSON


```
{
  "model": {
    "type_map": [
      "O",
      "H"
    ],
    "descriptor": {
      "type": "se_e2_a",
      "sel": [
        46,
        92
      ],
      "rcut_smth": 0.5,
      "rcut": 6.0,
      "neuron": [
        25,
        50,
        100
      ],
      "resnet_dt": false,
      "axis_neuron": 16,
      "seed": 1,
      "activation_function": "tanh",
      "type_one_side": false,
      "precision": "float64",
      "trainable": true,
      "exclude_types": [],
      "set_davg_zero": false
    },
    "fitting_net": {
      "neuron": [
        240,
        240,
        240
      ],
      "resnet_dt": true,
      "seed": 1,
      "type": "ener",
      "numb_fparam": 0,
      "numb_aparam": 0,
      "activation_function": "tanh",
      "precision": "float64",
      "trainable": true,
      "rcond": 0.001,
      "atom_ener": []
    },
    "data_stat_nbatch": 10,
    "data_stat_protect": 0.01
  },
  "learning_rate": {
    "type": "exp",
```

```
    "scale_by_worker": "linear"
  },
  "loss": {
    "type": "ener",
    "start_pref_e": 0.02,
    "limit_pref_e": 1,
    "start_pref_f": 1000,
    "limit_pref_f": 1,
    "start_pref_v": 0,
    "limit_pref_v": 0,
    "start_pref_ae": 0.0,
    "limit_pref_ae": 0.0,
    "start_pref_pf": 0.0,
    "limit_pref_pf": 0.0
  },
  "training": {
    "training_data": {
      "systems": [
        "../data/data_0/",
        "../data/data_1/",
        "../data/data_2/"
      ],
      "batch_size": "auto",
      "set_prefix": "set",
      "auto_prob": "prob_sys_size",
      "sys_probs": null
    },
    "validation_data": {
      "systems": [
        "../data/data_3"
      ],
      "batch_size": 1,
      "numb_btch": 3,
      "set_prefix": "set",
      "auto_prob": "prob_sys_size",
      "sys_probs": null
    },
    "numb_steps": 1000000,
    "seed": 10,
    "disp_file": "lcurve.out",
    "disp_freq": 100,
    "save_freq": 1000,
    "save_ckpt": "model.ckpt",
    "disp_training": true,
    "time_training": true,
    "profiling": false,
    "profiling_file": "timeline.json",
    "tensorboard": false,
    "tensorboard_log_dir": "log",
    "tensorboard_freq": 1
  }
}
```

Other limitations

No devices limitations as long as you can run deepmd-kit.

Correctness verification

As a scientific computing software, DeePMD-kit requires high algorithm accuracy. Normally, we use dp test method of DeePMD-kit to verify the correctness of accelerated algorithm. As an example, one can easily use the dp test method by simply running the following command:

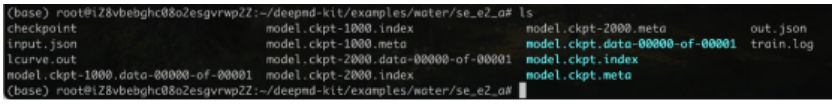
```
Bash
dp test -m frozen_model.pb -s data
```

Any optimization method must ensure the following verification of correctness: **The dp test results between DeePMD-kit asc-2022 branch and team's accelerated DeePMD-kit must be consistent.**

When you have finished your optimization strategies, please submit all the requested files of each case, per the following format:

Compressed file name	**Contents**	
deepmd.tar.gz	deepmd-kit	Directory of edited source code, including all the files downloaded from github.
	ase_results/water	Each case directory includes: 1. Input file (input.json) 2. Screen output during `dp train` procedure (train.log) 3. Other files automatically generated during `dp train` procedure (lcurve.out, model.ckpt*, checkpoint, out.json).
	ase_results/cu	
	ase_results/alloy	

For example, in ase_results/water directory, all files which are required for submitting are shown in the picture below:



Notes:

1. In the proposal Submit the description of computational resources, configuration and architecture used, and how long it took for each step. Submission of a log file is encouraged. May also describe the package compilation details and whether some modifications of the code were made, how and why.
2. Describe the strategies implemented to speed up the training procedure or reduce the computational needs for this task.
3. The modified code also should be submitted in your proposal, which will help to verify the correctness.

For any further questions, please contact techsupport@asc-events.org

Contact Us

Technical Support	Yu Liu techsupport@asc-events.org
Media	Jie He media@asc-events.org
Collaboration	Vangel Bojaxhi executive.director@asc-events.org
General Information	info@asc-events.org

TOP

Partners

Follow us