ASC Student Supercomputer Challenge 2020

Preliminary Round Notification

Dear ASC Teams,

Welcome to the 2020 edition of ASC Student Supercomputer Challenge (ASC20)! Now in its 9th year, ASC challenge is the world's largest supercomputing hackathon, striving to foster the next generation of young talents, and inspires exploration, innovation and collaboration in the field supercomputing and AI. In this notification, we will be detailing the submission guidelines, basic requirements and application tasks in the preliminary round of ASC20.

About the Preliminary Round

In the preliminary round, each team is required to submit a set of documents that include a proposal, optimized source code files and output files (detailed requirements are specified in Appendix A). The proposal should be written in English, and will be reviewed by the ASC evaluation committee.

Submission Guidelines

All teams should submit their full set of documents to info@asc-events.org before 0:00 AM, February 29, 2020 (UTC/GMT +8:00). The confirmation email will be sent out shortly upon receipt of the documents, which should include the following items:

a) The proposal in .docx or .pdf if you'd like. The file name should include the university or college's name and the contact person's name (e.g. AAAUniversity_BBB.docx).

b) The additional files should be compressed into one file (e.g. AAAUniversity_BBB.zip, other compression formats are allowed). The compressed file should at least include 4 folders (detailed requirements are specified in Appendix A):

- Output files of HPL
- Output files of HPCG
- Required files of Language Exam (LE) Challenge
- Required files of QuEST Challenge

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 ASC20 journey!
Sincerely,
ASC20 Committee

Appendix A: Proposal Requirements

I. Brief introduction of the university's or the department's supercomputing activities (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 how you set up the team
2. Brief introduction of each team member (including group photos of the team)
3. Team slogan.

III. Technical proposal requirements (90 points)

1. Design of HPC system (15 points)

a) Within the 3,000-watt power budget, the system should be designed to achieve the best computing performance.

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

c) Your system should be based on the Inspur NF5280M5 server. The components and the specific power consumption listed in the table below are for your reference when design the system. The NF5280M5 server can support up to 4x GPUs.

Item	Name	Configuration
Server	Inspur NF5280M5	CPU: Intel Xeon Gold 6230 x 2，2.1GHz，20 cores Memory: 32G x 12，DDR4，2933Mhz Hard disk: 480G SSD SATA x 1 *Power consumption estimation:* *6230 TDP 125W, memory 7.5W, hard disk 10W*
HCA card	EDR	InfiniBand Mellanox ConnectX®-5 HCA card, single port QSFP, EDR IB *Power consumption estimation: 9W*
Switch	GbE switch	10/100/1000Mb/s，24 ports Ethernet switch

		Power consumption estimation: 30W
	EDR-IB switch	Switc-IB™ EDR InfiniBand switch, 36 QSFP port *Power consumption estimation: 130W*
Cable	Gigabit CAT6 cables	CAT6 copper cable, blue, 3m
	InfiniBand cable	InfiniBand EDR copper cable, QSFP port, cooperating with the InfiniBand switch for use

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 testing method, performance optimization methods, performance estimation, problem and solution analysis, etc. In-depth analysis on HPL, HPCG algorithm and the source code 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 on x86 Xeon CPU or Tesla GPU platforms. Furthermore, teams that have to use other hardware platforms are welcome to submit their analysis and results if there is a reasonable performance.

3. Language Exam (LE) Challenge (30 points)

The Task

Teaching machines to understand human language documents is one of the most elusive and long-standing challenges in artificial intelligence [1]. In order to do so, different tasks in various aspects should be solved like part-of speech tagging, named entity recognition, syntactic parsing, coreference resolution *et al*.

Since the success of deep learning in other areas like computer vision, designing a deep neural network to understand human language is also a long been concerned approach. Bert, a transformer [3] style architecture deep neural network released in 2018, is the first language model that successfully applied to variety of language tasks such as sentimental analysis, question answering, named entity recognition *et al*. Since then, some other transformer style models, like XLNet[4], transformer-xl[5], RoBERTa[6], GPT2[7] also published in the recent year.

There are variety of tasks and datasets designed to evaluate how well the deep neural network understand the human language. Most of the tasks, like SQuAD [8], are either crowd-sourced or automatically-generated, bringing a significant amount of noises in the datasets and limits the ceiling performance by domain experts. English language exam, is a comprehensive task designed by teacher to assess the level of the student of the understanding about the English. Commonly used tasks in the exam including listening comprehension, cloze, reading comprehension and writing. Using these teacher-designed human-oriented tasks to evaluate the performance of neural networks is straightforward and also challengeable.

In the preliminary round of ASC20, a cloze style dataset is provided. The dataset is collected from internet and contains multi-level English language exams in China including high school level and college entrance exams, CET4 (College English Test 4), CET6 (College English Test 6) and NETEM (National Entrance Test of English for MA/MS Candidates). Part of the data comes from public CLOTH dataset [9]. Among the dataset, there are 4603 passages and 83395 questions in the training set, 400 passages and 7798 questions in the dev set and 400 passages and 7829 questions in the test set. The participants should design and train a neural network to achieve the best performance on the test set.

Dataset

Download dataset at Baidu: https://pan.baidu.com/s/1t7miv2h2MyEmY191y6lXOA (password: fhd4) or Microsoft OneDrive https://1drv.ms/u/s!Ar_0HIDyftZTsBiXPJUtGfiMmTom?e=w7VPAx. Below is a sample of the training set. Each data is organized into a json file, the json file contain 3 elements: “article”, “options” and “answers”. “_” is used as the placeholder in the article.

{"article": "At age 86, Millie Garfield is one of the world's oldest elderly bloggers . _ reading a newspaper article in 2003 and then asking her son for _ in getting online, Millie has been blogging ever since. We usually associate blogging with the _ : our children, grandchildren, nieces or nephews. While the blogging landscape was once _ almost entirely by teens, it has opened to different age groups now. After 38 years of marriage, Millie _ her husband in 1994. She has no siblings and has only one son. She has to live alone. Like many elderly people, her social network was beginning to _ in size as many of her friends were in assisted living. Blogging has _ Millie's universe. \"I have to blog once a week,\" she says. \"If I don't, they start _ about me.\" When I ask who \"they\" are, Millie says they are the 70 or 80 _ who visit her blog each day. When she was three days _ in posting one week, she began getting _ from them to see if she was okay. She has also got to _ other bloggers from around the country. Not only has blogging helped Millie make new _ , but it has also helped her learn about herself. \"I write about everyday living in a _ fashion, so I try to find interesting things in a TV show, a movie, or a(n) _ to the dentist, she says. \"I never knew I was funny but now people _ me I am. It is a big discovery.\" Millie _ loves blogging. \"My life would be _ and empty without it. I'm able to learn from people all over the world,\" she says. Then she adds, \"When you're older, you don't have many _ . The wonderful thing about blogging is that you can have many people hear what you think and no one _ you when you are speaking.\"",
"options": [["While", "Until", "After", "As"], ["help", "apology", "excuse", "permission"], ["old", "young", "rich", "sick"], ["damaged", "occupied", "prepared", "designed"], ["missed", "followed", "recognized", "lost"], ["grow", "develop", "decrease", "remain"], ["expanded", "concluded", "found", "ruined"], ["complaining", "thinking", "arguing", "worrying"], ["workers", "readers", "passengers", "speakers"], ["late", "away", "fast", "ready"], ["warnings", "suggestions", "emails", "books"], ["know", "see", "change", "ask"], ["comments", "connections", "contributions", "combinations"], ["popular", "famous", "similar", "humorous"], ["gift", "visit", "wave", "award"], ["warn", "prove", "order", "tell"], ["probably", "fortunately", "hardly", "clearly"], ["poor", "slow", "dull", "simple"], ["listeners", "managers", "interpreters", "lecturers"], ["fears", "interrupts", "controls", "treats"]],
"answers": ["C", "A", "B", "B", "D", "C", "A", "D", "B", "D", "C", "A", "B", "D", "B", "D", "D", "C", "A", "B"]}

The only difference between train/evaluation datasets and test dataset is that the test dataset only contains 2 elements: article and options. The answers of the test dataset are reserved by the committee for scoring.

Result Submission

The participant should organize all its results into a single json file and follow the format below:

```
{
"test0001", ["A","A","A","A","A","A","A","A","A","A","A","A","A",……],
"test0002", ["B","B","B","B","B","B","B","B","B","B","B","B","B",……],
……
}
```

For both preliminary and final rounds, each team should also submit a folder that contains source code and model that could reproduce the test results. The folder structure should be like:

Folder Name	Contents
LE	Root directory
test	A single json file contains answers for test set.
script	PyTorch source code here
model	PyTorch model here

Evaluation

In the preliminary, the score is calculated based on the formula below:

$$S = 30 \cdot \left(\frac{s_{test} - 60}{s_{max} - 60}\right)^2$$

- s_{test} is the score based on the test set, and s_{max} is the highest score on the test set among all valid submissions.
- 60 is used as the baseline score as it is the most commonly used score to measure whether the student passes the exam.
- s_{test} should be higher than 60, otherwise the score S will be 0.

Training Framework and baseline code

The participants must use PyTorch framework (https://pytorch.org/) for this task. The submission using or depending on any other deep learning framework will be forfeited.

The committee do not supply any baseline code for this task, the participants should design their deep learning network based on public resources. The participants should also consider the training performance of their neural network. Using distributed training strategies like data-parallelism and model-parallelism to accelerate the training process is encouraged, as the training performance on the participant-designed HPC cluster will be used as one of the metrics for scoring in the final.

Hardware requirement

It is highly recommended to run the training code on a GPU platform.

Reference

1. Danqi Chen, Neural reading comprehension and beyond. https://purl.stanford.edu/gd576xb1833
2. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding https://arxiv.org/abs/1810.04805
3. Attention Is All You Need https://arxiv.org/abs/1706.03762
4. XLNet: Generalized Autoregressive Pretraining for Language Understanding https://arxiv.org/abs/1906.08237
5. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context https://arxiv.org/abs/1901.02860
6. RoBERTa: A Robustly Optimized BERT Pretraining Approach https://arxiv.org/abs/1907.11692
7. Better Language Models and Their Implications https://openai.com/blog/better-language-models/
8. https://rajpurkar.github.io/SQuAD-explorer/
9. Large-scale Cloze Test Dataset Created by Teachers https://arxiv.org/abs/1711.03225

4. The QuEST Challenge (30 points)

The Task

Because of the huge cost of computational resource and elapsed time, numerous problems, such as cryptology, many body quantum mechanics and quantum machine learning, still can't be solved effectively even utilizing those powerful supercomputers. However, the quantum computer, which is based on the quantum mechanics, shows its great advantage in these fields benefiting from the suitable quantum algorithms. Among those quantum algorithms, the Shor large number decomposition algorithm as well as the Grover quantum search algorithm are the two most famous algorithms, which gain intense interest from the scientists and inspire the development of quantum computing and quantum information. Unfortunately, the achievement of the quantum computer hardware is so slow that there is no actual quantum computer which can be used for solving the science or cryptology issues. Therefore, so far, the quantum system is still costly simulated by using the classical computers.

QuEST is the first open source, hybrid multithreaded and distributed, GPU accelerated simulator of universal quantum circuits. QuEST is capable of simulating generic quantum circuits of general one and two-qubits gates and multi-qubit controlled gates, on pure and mixed states, represented as state-vectors and density matrices, and under the presence of decoherence.

Describing the state of an n-bit classical register requires n bits, but describing the state of an n-qubit quantum register requires 2^n complex numbers. Consequently, simulating a quantum computer using a classical machine is believed to be exponentially costly with respect to the number of qubits. Despite this, classical simulation of quantum computation is vital for the study of new algorithms and architectures.

The mathematical principle and algorithm for QuEST have been fully described in Reference [1]. We strongly recommend using the stable version of QuEST_2.1.0 and the corresponding source code is available in Reference [2]. More information about installation and usage of QuEST can be found from Reference [3].

In the preliminary round of ASC20, all teams are encouraged to complete the simulation of quantum circuits composed of 30 qubits by using the provided quantum random circuit (random.c) and the quantum Fourier transform circuits (GHZ_QFT.c). The memory requirement is at least 16GB. Actually, such circuits are intended for cracking the RSA encryption algorithm. In this challenge, you should obtain the right results and make efforts to reduce the computational needs, both in time and resources. The proposal document should include descriptions of the software environment (operating system, complier, math library, MPI software, and QuEST version, etc.), the testing method, performance optimization methods, performance estimation, problem and solution analysis, etc. In-depth analysis into QuEST's algorithm and source code is highly encouraged. The detailed tasks and requirements of this challenge are listed below.

Compile and install QuEST and run the program against the given data according to the instructions.

In order to compile and install QuEST, you may refer to the following steps:

The source code of QuEST can be obtained by:
The QuEST is most easily downloaded by using apt, git and GNU make, which can first be obtained with "*sudo apt install git make* (Ubuntu) or *yum install git make* (Redhat)"; then QuEST can be downloaded to the current directory (path/ to/QuEST) with "*git clone https://github.com/QuEST-Kit/QuEST.git*".
Or you can directly download the compressed file of QuEST (.tar.gz) at https://github.com/QuEST-Kit/QuEST/releases.

Install QuEST and run the challenge tests
Three necessary files, including the mytimer.hpp, random.c and GHZ_QFT.c, are provided and can be downloaded through Baidu SkyDrive or Microsoft OneDrive, same as the Language exam (LE) challenge listed above. In this challenge, two workloads should be completed: for the first workload named random circuit, mytimer.hpp, random.c and the source code of QuEST will be used, while for the other workload named GHZ_QFT, the mytimer.hpp, GHZ_QFT.c and the source code of QuEST will be used. Due to the independence of the two workloads, QuEST should be installed separately to run each workload.

For the workload named random circuit:
“cp random.c tutorial_example.c”
“cd build”
“cmake ..”
“make –j4”
“./demo”
Note this is an example to show how to install and run the tests. The participants can compile the code by using their optimizing strategies or run the test with MPI or OpenMP.
The other workload (GHZ_QFT.c) can be installed the same way mentioned above, except for “*cp GHC_QHF.c tutorial_example.c*” (instead of “*cp random.c tutorial_example.c*”).

Result Submission

Please submit all the requested files of each workload in following formats:

Workload name	Compressed file name	Contents
Case1: random circuit	random.tar.gz	Probs.dat stateVector.dat Command line file(*.sh) Screen output(*.log)
Case2: GHZ_QFT	GHZ_QFT.tar.gz	Probs.dat stateVector.dat Command line file(*.sh) Screen output(*.log)

In the proposal, please describe test platform include hardware configuration and architecture, and run time for each step (submission of a log file is needed). Also the compiling process of the package and your modifications of the source code, how and why, should be described. Describe the strategies used to optimize the performance in the test process. The modified code should be submitted along with the proposal, which gives strong support for verifying the correctness of optimizing strategies.

Evaluation

In each workload, two output files will be generated: probs.dat and stateVector.dat. The former describes the probability of every qubit that equals to 1, while the latter provides the amplitude of the first ten state vectors, which are complex numbers including real and imagine parts. The evaluation follows the rules as below:
1. The output files in each workload should be the exactly same as the given references of probs.dat and stateVector.dat. The corresponding references can be obtained through Baidu SkyDrive or Microsoft OneDrive as listed above;
2. Then the execution time will be evaluated by the number listed in the screen output file *.log on the ground that 1st condition is fulfilled. However, the computing platforms that used by participants have great influence on the execution time, such influence will be considered when giving the score.

3. The score of two workloads is equal because of the independence of them.
4. Proposals with clarity as well as rigorous descriptions are highly appreciated and beneficial for higher scores.

Hardware requirement

In these two contest cases, **if GPU accelerating option is used, the results may be not correct.** If teams decide to accelerate the task using GPU, please check the output files and be sure it is the exactly same as the given references of probs.dat and stateVector.dat.

Reference

[1]. QuEST description:
Quest and high performance simulation of quantum computers. Jones T, Brown A, Bush I, et al. Scientific reports, 2019, 9(1): 1-11.
[2]. QuEST_v2.1.0 source code:
https://github.com/QuEST-Kit/QuEST/releases
[3]. QuEST userguide:
https://quest.qtechtheory.org/docs/

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