ASC Student Supercomputer Challenge 2020-2021

Preliminary Round Notification

Dear ASC Teams:

Welcome to the 2020-2021 edition of ASC Student Supercomputer Challenge ASC20-21!
ASC Student Supercomputer Challenge, now in its 9th year, 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. This document details the submission guidelines, basic requirements, and application tasks for the preliminary round of ASC20-21.

Preliminary Round

In the preliminary round, the ASC Committee asks each team to do their best in accomplishing all of the tasks for this competition and finalizing the document set which include the proposal, optimized source code, and output files. The ASC evaluation committee will review the proposals written in English.

Submission Guidelines

All teams are expected to submit the following items to info@asc-events.org before 24:00, of January 8, 2021 (UTC/GMT +8:00).

a) The proposal document, with the name of the university or college 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: ABCUniversity_John Doe. The compressed file should include at least five folders per the detailed requirements in Appendix A.

- Output files of HPL
- Output files of HPCG
- Required files of Language Exam (LE) Challenge
- Required files of QuEST Challenge
- Required files of PRESTO 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 ASC20-21 journey!
ASC20-21 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)

1. Design of HPC system (11.25 points)

a) The system should be designed for best computing performance, within the limitation of 3KW power consumption,

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) The components and the specific power consumption listed in the table below are for reference, based on the Inspur NF5280M5 server. 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 (11.25 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 algorithms and the 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 or Tesla GPU platforms. In addition, if other hardware platforms are used, you are welcomed to submit the related analysis and results of demonstrating reasonable performance.

3. Language Exam (LE) Challenge (22.5 points)

Task Description:

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

The successful use of deep learning in other areas like computer vision by designing deep neural networks has long been a concerned approach to AI understanding human language. 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, etc. Since then other transformer style models, like XLNet[4], transformer-xl[5], RoBERTa[6], GPT2[7] also published in the recent years.

There are variety of tasks and datasets designed to evaluate how well the deep neural networks can 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 limit the ceiling performance of the domain experts. The English language exam is a comprehensive test designed to assess the level of proficiency of the students in English language. Commonly used tasks in the English exam including listening comprehension, cloze style dataset, reading comprehension, and writing. Using a teacher-designed and human-oriented tasks to evaluate the performance of neural networks is straightforward but also challenging.

In the preliminary round of ASC20-21, a cloze style dataset is provided. The dataset is collected from internet and contains multi-level English language exams used in China for high school, 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]. There are 4603 passages and 83395 questions in the training set, 400 passages and 7798 questions in the developer set, and 400 passages and 7829 questions in the test set. The participants should design and train their neural network to achieve the best performance on the test set.

Dataset

Download dataset from 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 participants should organize all results into a single json file in 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. A submission using or depending on any other deep learning frameworks will be forfeited.

The ASC 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. It's encouraged to use distributed training strategies like data-parallelism and model-parallelism to accelerate the training process as the training performance on the participant-designed HPC cluster will be used as one of the scoring metrics 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 (22.5 points)

Task Description:

Because of the huge cost of computational resources and the processing time needed, numerous problems such as cryptology, many body quantum mechanics, and quantum machine learning still can't be solved effectively even utilizing the most powerful supercomputers today. However, quantum computers, which are based on the quantum mechanics, have demonstrated huge advantages in these fields by benefiting from the advanced quantum algorithms. Among those quantum algorithms, the Shor large number decomposition algorithm as well as the Grover quantum search algorithm are the two most renowned. They have raised interest from the scientists and is inspiring further development of quantum computing and quantum information. Unfortunately, the development curve of the quantum computer hardware is still not very step, and there is no actual quantum computer which can be used for solving practical science or cryptology issues. Thus, the quantum systems are still 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-qubit and two-qubits and multi-qubit controlled gates, on pure and mixed states, represented as state-vectors and density matrices under the presence of decoherence.

Describing the state of an n-bit classical register requires just n bits while describing the state of an n-qubit quantum register requires 2^n complex numbers. As a result, simulating a quantum computer using a classical machine is believed to be exponentially expensive with respect to the number of qubits. But despite this, classical simulation of quantum computation is crucial for the study of new algorithms and architectures.

The mathematical principle and algorithm for QuEST have been fully described in Reference [1]. In this competition, you must use 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 in Reference [3].

In the preliminary round of ASC20-21, all teams are encouraged to complete the simulation of quantum circuits 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. Such circuits are actually intended for cracking the RSA encryption algorithm. In this challenge, you should strive for the best results by reducing the computational needs 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.), testing methods, 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 for this challenge are listed below.

Please compile, install the QuEST and run the program against the given data according to the instructions below:

First, should compile and install QuEST, referring to the following steps:

1. Download the source code of QuEST

- ➢ The QuEST is most easily downloaded by using apt, git and GNU make, which can be obtained with "*sudo apt install git make* (Ubuntu) or *yum install git make* (Redhat)".
- ➢ Downloaded it 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.*

2. Download the necessary QuEST associated files

   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**,** similar with 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, and 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 for each workload.

3. Install QuEST and run the challenge tests

For the workload named random circuit:

"cp random.c tutorial_example.c"
"cd build"
"cmake .."
"make –j4"
"./demo"

Note that this is just 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 second workload (GHZ_QFT.c) can be installed in 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 using the 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 the test platform, including hardware configuration, architecture, and run time for each step (submission of a log file is needed). Also, explain the compiling process of the package and your modifications of the source code. Describe the strategies used to optimize the performance in the test process. The modified code should be submitted along with the proposal, to substantiate the verification of the correctness for 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 will follow the rules below:

1. The output files in each workload should be the exact same as the given references of probs.dat and stateVector.dat. The corresponding references can be obtained through Baidu SkyDrive or Microsoft OneDrive listed above.
2. 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, since the computing platform used by participants will impact the execution time, its specifics will be considered in scoring process.
3. Proposals written with clarity and rigorous description will benefit for higher score.
3. Each workload is accounted for 50% of the scores.
4. You can optimize the source code of QuEST. But please note that the provided circuit GHZ_QFT.c and random.c should not be modified. Any changes to them will void the score.

Hardware Requirement

In both two cases, if GPU accelerating option is used, the results may be not correct**.** If you decide to accelerate the task using GPUs, please check the output files and make sure that they are the exact same as of probs.dat and stateVector.dat, given for references.

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/

5. The PRESTO Challenge (22.5 points)

Task Description:

Radio pulsars are fast-spinning neutron stars that, like stellar lighthouses, emit strong beams of radiation. Some pulsars are very stable cosmic clocks that could help to practically test General Relativity based on time and to help detect gravitational waves which are triggered by the merging of supermassive black holes in distant galaxies.

In the past decades almost 3000 pulsars were discovered by using the pulsar data observed by radio telescopes. The pulsar search data are organized in array of very quickly sampled spectra containing two dimensions: time and frequency. The lower frequency part of the data is delayed in comparison to higher frequency due to the interstellar dispersion. The time delay is described by the following equation: $\Delta t = 4.15 \times 10^{-9}\text{s}\ (f_1^{-2} - f_2^{-2}) \times \text{DM}$ [1], where f_1 and f_2 are the two frequencies in units of MHz, and DM is the measured level of dispersion. The pulsars could be found by two significant features: 1. their signals are dispersed by the interstellar medium; 2. their signals are periodic with periods ranging from milliseconds to seconds. Most pulsar searching algorithms take at least two steps: 1. Testing of many (often thousands) possible dispersion measures (DM) and correcting them for the interstellar delays, - so called "de-dispersion"; 2. Searching for a period of time using fast Fourier transformation.

.

PRESTO (**P**ulsa**R** **E**xploration and **S**earch **TO**olkit) [2] is large suite of pulsar search and analysis software. It is written primarily in ANSI C, with many of the recent routines in Python. PRESTO was primarily designed to efficiently search for binary millisecond pulsars from long observations of globular clusters. The software is composed of numerous routines designed to handle three main areas of pulsar analysis:

1. Data Preparation: Interference detection (rfifind) and removal (zapbirds) , de-dispersion (prepdata, prepsubband, and mpiprepsubband), barycentering (via TEMPO).
2. Searching: Fourier-domain acceleration (accelsearch), single-pulse (single_pulse_search.py), and phase-modulation or sideband searches (search_bin).
3. Folding: Candidate optimization (prepfold) and Time-of-Arrival (TOA) generation (get_TOAs.py).

More than 700 pulsars have been discovered by PRESTO, including almost 300 recycled and/or binary pulsars.[2]

The mathematical principle and algorithm for PRESTO have been fully described in Reference [2]. We strongly recommend using the stable version of PRESTO and the corresponding source code is available in Reference [3]. More information about installation and usage of PRESTO can be found from Reference [4].

In the preliminary round of ASC20-21, all teams are required to parallelize the pulsar search process. We provide two set of data and a python script "pipeline.py" to demonstrate the process. The teams are encouraged to develop their parallel program or python script to optimize the use of multi-core CPU hardware. In this challenge, you should obtain the correct results and make efforts to reduce the computational time and resources. The proposal document should include descriptions of the software environment (operating system, complier, python library, and PRESTO version, etc.), the testing method, parallelization strategy, performance optimization methods, performance estimation, problem and solution analysis, etc. In-depth analysis into PRESTO algorithm and source code [3] is highly encouraged. The detailed tasks and requirements of this challenge are listed below.

Compile and install PRESTO and its depended libraries, and run the program against the given data according to the instructions.

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

The source code of PRESTO can most easily downloaded by git with command of "*git clone https://github.com/scottransom/presto.git*". Or you can download the compressed file of PRESTO (.tar.gz) at "*https://github.com/scottransom/presto/releases*".

Install PRESTO and run the challenge tests:

For the installation steps of PRESTO and its depended libraries could refer to "*https://github.com/scottransom/presto/blob/master/INSTALL*". Three necessary files, including the example python script "pipeline.py" for pulsar searching, and two sets of data file "GBT_Lband_PSR.fil" and "Dec+1554_arcdrift+23.4-M12_0194.fil", can be download through Baidu SkyDrive or Microsoft OneDrive. In this challenge, two workloads should be completed with the participants' parallelized version program or script: for the first workload named "PRESTO Dedispersion", the data file "GBT_Lband_PSR.fil" and the source code of PRESTO will be used, while for the other workload named "FAST PulsarSearch", the data file "Dec+1554_arcdrift+23.4-M12_0194.fil" and the source code of PRESTO will be used.

Please use the following commands to run two workloads with the example python script for pulsar searching:

Workload 1:

"*mkdir TestData1 && cd TestData1*"

"(*time python ./pipeline.py GBT_Lband_PSR.fil*) > *log.pulsar_search 2>&1*"

"*cd -*"

Workload 2:

"*mkdir TestData2 && cd TestData2*"

"(*time python ./pipeline.py Dec+1554_arcdrift+23.4-M12_0194.fil*) > *log.pulsar_search 2>&1*"

"*cd -*"

Note that this is an example to show how to run the example python script for pulsar searching. The python file "*./pipeline.py*" should be replaced by the parallelized version program or script proposed by the participants. The source code of PRESTO could also be modified with parallelization or optimizing strategies. However, the original parameters and output information in "pipeline.py" should not be changed.

Result Submission

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

Workload name	Compressed file name	Contents
Case1: PRESTO_Dedispersion	PRESTO_Dedispersion.tar.gz	Command line file(*.sh) Screen output(log.pulsar_search) Result files (subbands/*.log, subbands/*.inf, subbands/*.pfd, subbands/*.bestprof, subbands/*.ps)
Case2: FAST_PulsarSearch	FAST_PulsarSearch.tar.gz	Command line file(*.sh) Screen output(log.pulsar_search) Result files (subbands/*.log, subbands/*.inf, subbands/*.pfd, subbands/*.bestprof, subbands/*.ps)

In the proposal, please describe the 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 parallelize and optimize the performance in the pulsar search process. The modified code should be submitted along with the proposal, to substantiate the support for verifying the correctness of optimizing strategies.

Evaluation

For each workload, a result folder "subbands" will be generated. The log files of each process and the search results are placed in the folder. The evaluation follows the rules below:

1. The output file name in each workload should be the exactly same as the given example. The example python script can be obtained through Baidu SkyDrive or Microsoft OneDrive as listed above.
2. The execution time will be evaluated by the number listed in the screen output file "log.pulsar_search" in each work directory. However, since the computing platform used by participants will impact the execution time, its specifics will be considered in scoring process.
3. Like the previous workload, the results from this workload will count for 50% of the scores.
4. Proposals written with clarity and rigorous description will benefit for higher score.

Hardware Requirement

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

Reference

[1]. Pulsar searching algorithm description:
Manchester R N, Taylor J H. Parameters of 61 pulsars[J]. Astrophysical Letters, 1972, 10: 67.
[2]. PRESTO Home Page:
https://www.cv.nrao.edu/~sransom/presto/
[3]. PRESTO source code:
https://github.com/scottransom/presto
[4]. PRESTO install guide:
https://github.com/scottransom/presto/blob/master/INSTALL

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