KISTI-6 BMT program list and rules[1)]

2023.04.14. KISTI

<Table 1> BMT program list for KISTI-6 procurement

Category	Program	Requirement	Target system
Component	Stream	Mandatory[2)]	CPU partition and GPU partition
	OSU Micro-Benchmarks	Mandatory	GPU partition
	IOR	Mandatory	GPU partition
	MDTEST	Mandatory	GPU partition
	Stride	Reference[3)]	CPU partition and GPU partition
	HPCC RandomAccess	Reference	CPU partition and GPU partition
	CLOMP	Reference	CPU partition and GPU partition
Comprehensive (Standard)	HPL	Mandatory	CPU partition and GPU partition
	HPL-AI	Mandatory	GPU partition
	HPCG	Mandatory	CPU partition and GPU partition
	GREEN500	Mandatory	GPU partition
	GRAPH500	Mandatory	CPU partition or GPU partition
	MLPerf	Mandatory	GPU partition
	P3DFFT	Mandatory	CPU partition or GPU partition
Application	LAMMPS	Mandatory	GPU partition
	PELE-LM	Mandatory	CPU partition and GPU partition[4)]
	PySCF	Mandatory	CPU partition
	Quantum Espresso	Mandatory	GPU partition
	QUEST	Mandatory	CPU partition or GPU partition or memory partition

1) All contents of this document except the BMT programs are tentative and subject to change until RFP is finalized.
2) The result of BMT programs marked by "Mandatory" should be submitted, and they will be scored by relative evaluation.
3) The result of BMT programs marked by "Reference" should be submitted, but they will not be scored.
4) Results on CPU and GPU partitions must be submitted, while the better one will be evaluated.

1. General rules for source code modification and optimization

1.1. Modification and optimization of the source codes of BMT programs should be done under the following general conditions. Additional conditions stated in Appendix 3 have precedence.

1) If no special conditions are given, floating-point operations should use FP64 precision and integer operations should use INT32 precision.

2) Options for compile and link may be used if they are officially supported by suppliers. (Proof of support must be submitted with the BMT results.)

3) Any optimized libraries can be used if they are officially supported by suppliers. (Commitment of support should be submitted with the BMT results.)

4) If any reformulations of mathematical operations are made, the results should be identical within an acceptable range of accuracy.

5) Modifications and optimizations involving solution estimation (i.e., interpolation or extrapolation) are not acceptable.

6) Modifications of arguments in functions or subroutines are not acceptable.

7) Modifications and optimizations related to parallelization are permitted if the logic of arithmetic operations is unchanged and the accuracy of results is ensured.

1.2. Modifications and optimizations that do not belong to the general rules and detailed rules for each BMT program are acceptable only if they are agreed upon by KISTI staffs in case modifications are inevitable for porting to the proposed system and running on the proposed system.

2. Individual Component Performance Test

2.1. STREAM

2.1.1. Overview

1) STREAM is a tool to measure bandwidth between processor and memory in a single node.

2.1.2. Installation

1) Download BabelStream source code (version 4.0) from the official website, https://github.com/UoB-HPC/BabelStream

2.1.3. Execution

1) Build and execute model according to the following guide

```
$ cmake −Bbuild −H . -DMODEL=(model) (additional options depending on model selection)
$ cmake --build build
$ ./build/(model)-stream (additional execution options)
```

2) Change the value of OMP_NUM_THREADS to find the number of threads for optimal performance results.

```
$ export OMP_NUM_THREADS = (the number of threads)
```

2.1.4. Submission

1) CMake build options (with additional options depending on model selection)
2) Execution options (e.g., environment variable settings, option settings)
3) Peak value of Triad Rate (MB/s) and the number of optimal threads
4) All processor information and memory of the proposed system must be described.
5) Results can be submitted using its own Stream benchmark tool if BabelStream is not available on the processor. (In this case, the source code for benchmark should be submitted)

<Table 2> Benchmark Test Results: STREAM

Target System		Processor/ Memory Specifications	Common	
			Optimal Thread Count	Triad Rate (MB/s)
BMT System	CPU Node	Host specification		
	GPU Node	Host specification		
		Accelerator specifications		
Proposed System	CPU Node	Host specification		$P_{stream,\ 1}$
	GPU Node	Host specification		$P_{stream,\ 2}$
		Accelerator specifications		$P_{stream,\ 3}$

※ Scale-up projection is permitted in the STREAM test, and the tenderer must submit results from both CPU and GPU nodes.

※ Results in shaded marks (▨) are used for evaluation

2.1.5. Run rules

1) A STREAM test should be carried out according to benchmark rules.

2) The tenderer should carry out a test of the bandwidth between the CPU and the DDR main memory (off-package) and between GPU and GPU memory (on-package) for single compute node and The tenderer should submit the maximum bandwidth of single node using MPI and OpenMP based on the measured and estimated results.

3) Algorithmic modification is not admitted.

2.2. OSU Micro-Benchmarks (OMB)

2.2.1. Overview

1) The OSU Micro-Benchmarks(OMB) is a collection of independent micro-benchmarks for MPI message passing performance developed and written at Ohio State University. It includes benchmarks for performance and scalability such as latency, bandwidth and host overhead.

2.2.2. Installation

1) Download OSU Micro-Benchmarks 7.0 from the official website (http://mvapich.cse.ohio-state.edu/benchmarks/)

2.2.3. Execution

1) Point-to-Point, Collective MPI, Non-Blocking Collective MPI Benchmarks should be conducted

```
$ mpirun −np #ps executable
```

※ #ps: number of processes

※ Executable file

Point-to-Point MPI Benchmarks: osu_latency, osu_multi_lat, osu_bw, osu_bibw,

Collective MPI: osu_bcast, osu_alltoall, osu_allgather, osu_scatter, osu_allreduce

Non-Blocking Collective MPI: osu_ibcast, osu_ialltoall, osu_iallgather, osu_iscatter

2) A benchmark test must be carried out using all computing resources.

2.2.4. Submission

1) Compile options and link settings (makefile)

2) Execution options (e.g., environment variable settings, option settings)

<Table 3> Point-to-Point MPI Benchmarks Test Results: OMB

Target System		Case	Latency (usec) (message size = 0)		Bandwidth (GB/s) (message size = 4MB)	
			osu_latency	osu_multi_lat	osu_bw	osu_bibw
BMT system	GPU Partition	best case				
Proposed System	GPU Partition	best case	$T_{OMB,\ P2P,\ 1}$	$T_{OMB,\ P2P,\ 2}$	$B_{OMB,\ P2P,\ 1}$	$B_{OMB,\ P2P,\ 2}$
		3-hops case	$T_{OMB,\ P2P,\ 3}$	$T_{OMB,\ P2P,\ 4}$	$B_{OMB,\ P2P,\ 3}$	$B_{OMB,\ P2P,\ 4}$
		worst case	$T_{OMB,\ P2P,\ 5}$	$T_{OMB,\ P2P,\ 6}$	$B_{OMB,\ P2P,\ 5}$	$B_{OMB,\ P2P,\ 6}$

※ Scale-up prediction is only allowed for OMB Point-to-Point (P2P) MPI Benchmarks and the tenderer should submit the results from the GPU partition.

※ Results in shaded cells (▨) are used for evaluation.

※ Tests should be carried out by executing one MPI process on each node after choosing two random nodes. The osu_multi_lat case should be executed 24 MPI processes on each node.

※ For the 3-hops case, there must be three different switches between any two nodes.

※ For the best case, the shortest path should be tested between two nodes.

※ For the worst case, the longest path should be tested between two nodes.

<Table 4> OSU Collective MPI Benchmarks Test Results: OMB

Target System		Process	Message Size (byte)	Common				
				Execution Time (usec)				
				osu_bcast	osu_alltoall	osu_allgather	osu_scatter	osu_allreduce
BMT System	GPU Partition	Nmin	64					
			1M					
		...	64					
			1M					
		Nmax	64					
			1M					
Proposed System	GPU Partition	256	64					
			1M					
		512	64					
			1M					
		1024	64	$T_{OMB,Col,1}$	$T_{OMB,Col,2}$	$T_{OMB,Col,3}$	$T_{OMB,Col,4}$	$T_{OMB,Col,5}$
			1M	$T_{OMB,Col,6}$	$T_{OMB,Col,7}$	$T_{OMB,Col,8}$	$T_{OMB,Col,9}$	$T_{OMB,Col,10}$

※ Scale-up and scale-out predictions are allowed for OMB Collective MPI Benchmarks and the tenderer submits the results obtained from the GPU partition

※ Execute one MPI process on each node.

※ Results in shaded cells (▨) are used for evaluation.

<Table 5> OSU Non-Blocking Collective MPI Benchmarks Test Results: OMB

Target System		Process	Message size (byte)	Common			
				Execution time (usec)			
				osu_ibcast	osu_ialltoall	osu_iallgather	osu_iscatter
BMT System	GPU Partition	Nmin	64				
			1M				
		...	64				
			1M				
		Nmax	64				
			1M				
Proposed System	GPU Partition	256	64				
			1M				
		512	64				
			1M				
		1024	64	$T_{OMB,NB,\ 1}$	$T_{OMB,NB,\ 2}$	$T_{OMB,NB,\ 3}$	$T_{OMB,NB,\ 4}$
			1M	$T_{OMB,NB,\ 5}$	$T_{OMB,NB,\ 6}$	$T_{OMB,NB,\ 7}$	$T_{OMB,NB,\ 8}$

※ Scale-up and scale-out prediction are allowed for for OMB non-blocking Collective MPI

Benchmarks and the tenderer must submit the results in the GPU partition

※ Results in shaded cells (▨) are used for evaluation.

※ Execute one MPI process on each node.

2.2.5. Run rules

1) The OMB test should be carried out according to benchmark rules.

2) Modification of the source code for the optimization of the performance is not permitted.

2.3. IOR

2.3.1. Overview

1) IOR test is used for benchmarking parallel file systems.
2) IOR is used for testing the performance of parallel file systems using various interfaces and access patterns.

2.3.2. Installation

1) Download BMT code (v3.3 or higher) from GitHUB hpc/ior (https://github.com/hpc/ior)
2) Test after building the IOR executable file using the compiler and MPI library

2.3.3. Execution

1) The performance of read and write should be measured using IOR
2) Each result includes‘Max Write(MB/s)’and‘Max Read(MB/s)’
3) The benchmark test must be carried out by use of the GPU partition.
4) The benchmark test must be carried out separately in flash (NVMe SSD) storage and HDD storage.

   ※ Tests should be conducted under the conditions of 3.2.10.1.3 (RAID or parity configurations)

   ※ Tests should be conducted under the configurations of the distributed metadata settings in 3.2.9.1.11 similar to those in the production service environment of the proposed system as much as possible.

5) Tests should be conducted without applying the client cache of the compute nodes.
6) Execution command

```
$ mpirun –np #ps ior -a MPIIO –b #bs –o [PFS]/IOR_MPIIO –t #ts –s #sc -d 10 -C -Q 25
  -e -w -r -k -F –i 3
```

   ※ #ps: number of processes, [PFS]: parallel file system directory, #ts: transfer size

   ※ Tests should be conducted by adjusting the file size through the block size (-b) and number of segments (-s) options so that the cache effect (on-package and off-package memory) is excluded.

   Ex.) FileSize = BlockSize(#bs)×NumTasks(#ps)×SegmentCount(#sc)

2.3.4. Submission

1) Maximum bandwidth performance (MiB/s) of Flash (NVMe SSD) storage and HDD storage using MPI-IO
2) Target node information: BMT system and proposed system

<Table 6> NVMe File System Benchmark MPI-IO File-Per-Process Test Results: IOR

Target System			Num. of Nodes	Process	Common							
					Transfer Size							
					10 KB		100 KB		1M		10M	
					Max Write	Max Read	Max Write	Max Read	Max Write	Max Read	Max Write	Max Read
BMT system	GPU partition	Single Node	1									
		Optimal Node										
		Entire Node										
Proposed system	GPU partition	Single Node	1		$B_{IORw,NV,1}$	$B_{IORR,NV,1}$	$B_{IORw,NV,2}$	$B_{IORr,NV,2}$	$B_{IORw,NV,3}$	$B_{IORr,NV,3}$	$B_{IORw,NV,4}$	$B_{IORr,NV,4}$
		Optimal Node			$B_{IORw,NV,5}$	$B_{IORr,NV,5}$	$B_{IORw,NV,6}$	$B_{IORr,NV,6}$	$B_{IORw,NV,7}$	$B_{IORr,NV,7}$	$B_{IORw,NV,8}$	$B_{IORRr,NV,8}$

<Table 7> HDD File System Benchmark MPI-IO File-Per-Process Test Results: IOR

Target System			Num. of Nodes	Process	Common							
					Transfer Size							
					10 KB		100 KB		1M		10M	
					Max Write	Max Read	Max Write	Max Read	Max Write	Max Read	Max Write	Max Read
BMT system	GPU partition	Single Node	1									
		Optimal Node										
		Entire Node										
Proposed system	GPU partition	Single Node	1		$B_{IORw,HD,1}$	$B_{IORr,HD,1}$	$B_{IORw,HD,2}$	$B_{IORr,HD,2}$	$B_{IORw,HD,3}$	$B_{IORr,HD,3}$	$B_{IORw,HD,4}$	$B_{IORr,HD,4}$
		Optimal Node			$B_{IORw,HD,5}$	$B_{IORr,HD,5}$	$B_{IORw,HD,6}$	$B_{IORr,HD,6}$	$B_{IORw,HD,7}$	$B_{IORr,HD,7}$	$B_{IORw,HD,8}$	$B_{IORr,HD,8}$

※ Optimal nodes means the number of nodes that leads to peak performance.

※ Scale-up and scale-out projections are both permitted in the IOR test, and the tenderer submits the results obtained from the GPU partition

※ Results in shaded cells (▨) are used for evaluation.

2.3.5. Run rules

1) IOR test should be carried out according to benchmark rules.

2) Modification of the source code for the optimization of the performance is not permitted.

2.4. MDTEST

2.4.1. Overview

1) MDTEST is a program that measures the performance of various metadata operations (open/stat/close).
2) The benchmark test should be carried out under conditions similar to those of the proposal file system if possible.

2.4.2. Installation

1) Download BMT code from GitHUB IOR (https://github.com/hpc/ior)
2) Build the executable files using the compiler and MPI library

2.4.3. Execution

1) Describe the number of generated files for each node. (This is not the total number of generated files.)
2) Each benchmark test must be carried out by using the specified numbers of nodes and numbers of processes mentioned in the table below.
3) BMT should be performed without applying the client cache of the compute node.
4) Execution command and option

① Generate 1,000,000 files using a single processor in a single directory

```
$ mpirun -np 1 ./mdtest -n 1,000,000 -d [PFS]/[SDIR]/ -F -i 3
```

② Generate 1,000,000 files in multiple directories

```
$ mpirun -np #ps ./mdtest -n <1,000,000/#ps> -d [PFS]/[UDIR]/ -F -u -i 3
```

③ Generate 1,000,000 files in a single directory

```
$ mpirun -np #ps ./mdtest -n 1,000,000 -d [PFS]/[SDIR]/ -F -i 3
```

④ Generate 1 file by using multiple processes

```
$ mpirun -np #ps ./mdtest -n 1 -d [PSF]/[SDIR]/ -F -S -i 3
```

※ [PFS]: PFS directory, [SDIR]: Shared directory, [UDIR]: Individual directory

※ Directory parameter must be full path.

※ #ps: the number of processes

※ -u option: generating individual directory for each job

※ -S option: shared file access

2.4.4. Submission

1) The number of processes for testing are as follows in the table below, and tests are conducted on a single node and multiple nodes.
2) Target system: BMT system and proposed system

<Table 8> File System Benchmark Single Node Test Results: MDTEST

Target System		Test Target	Process/Node	Common			
				Creation Rate (files/sec)		Removal Time (sec)	
				NVMe SSDs	HDD	NVMe SSDs	HDD
BMT System	GPU Partition	Generate 1,000,000 files in multiple directories	4				
			8				
			Total number of available cores				
		Generate 1,000,000 files in a single directory	One				
			4				
			8				
			Total number of available cores				
		Generate 1 file by use of multiple processes	4				
			8				
			Total number of available cores				
Proposed System	GPU Partition	Generate 1,000,000 files in multiple directories	4				
			8				
			Total number of available cores	$I_{MDs,NV,1}$	$I_{MDs,HD,1}$		
		Generate 1,000,000 files in a single directory	One				
			4				
			8				
			Total number of available cores	$I_{MDs,NV,2}$	$I_{MDs,HD,2}$		
		Generate 1 file by use of multiple processes	4				
			8				
			Total number of available cores	$I_{MDs,NV,3}$	$I_{MDs,HD,3}$		

<Table 9> File System Benchmark Multi-node Test Results: MDTEST

Target System		Test Target	Num. of Node	Process/Node	Common			
					Creation Rate (files/sec)		Removal Time (sec)	
					NVMe SSDs	HDD	NVMe SSDs	HDD
BMT System	GPU Partition	Generate 1,000,000 files in multiple directories	Nmin	4				
				8				
				Total number of available cores				
			Nmax	4				
				8				
				Total number of available cores				
		Generate 1,000,000 files in a single directory	Nmin	4				
				8				
				Total number of available cores				
			Nmax	4				
				8				
				Total number of available cores				
		Generate 1 file by use of multiple processes	Nmin	4				
				8				
				Total number of available cores				
			Nmax	4				
				8				
				Total number of available cores				
Proposed System	GPU Partition	Generate 1,000,000 files in multiple directories	8	4				
				8				
				Total number of available cores	$I_{MDm,NV,1}$	$I_{MDm,HD,1}$		
			32	4				
				8				
				Total number of available cores	$I_{MDm,NV,2}$	$I_{MDm,HD,2}$		
		Generate 1,000,000 files in a single directory	8	4				
				8				
				Total number of available cores	$I_{MDm,NV,3}$	$I_{MDm,HD,3}$		
			32	4				
				8				
				Total number of available cores	$I_{MDm,NV,4}$	$I_{MDm,HD,4}$		
		Generate 1 file by use of multiple processes	8	4				
				8				
				Total number of available cores	$I_{MDm,NV,5}$	$I_{MDm,HD,5}$		
			32	4				
				8				
				Total number of available cores	$I_{MDm,NV,6}$	$I_{MDm,HD,6}$		

※ MDTEST allows scale-up and scale-out, and the tenderer submits the results obtained from the GPU partition

※ Results in shaded cells (▨) are used for evaluation.

2.4.5. Run rules

1) The MDTEST should be carried out according to benchmark rules.

2) Modification of the source code related to performance optimization are not allowed.

2.5. STRIDE

2.5.1. Overview

1) STRIDE is designed to stress the memory subsystem on a node severely using a series of sequential kernels.
2) Benchmark test for a single node.

2.5.2. Installation

1) Download STRIDE source (https://asc.llnl.gov/sites/asc/files/2020-09/STRIDE_v1.1.zip)

2.5.3. Execution

1) Install and run using C compiler

```
$ gcc -O strid3.c −o stride.exe −lm

$ ./stride.exe
```

2.5.4. Submission

1) Compile and link settings (makefile)
2) Execution options (e.g., environment variable settings, option settings)
3) Arithmetic average of operation speed (MF/s) from single node to STRIDE 1 to 1024
4) Memory and cache structure of BMT system must be described.
5) The total processor and memory structure of the proposed system must be described.

<Table 10> Benchmark Test Results: STRIDE

Target System		Processor and Memory type	Common	
			STRIDE Size	Computation Speed (MF/s)
BMT System	CPU node		One	
			2	
			...	
			1024	
	GPU node		One	
			2	
			...	
			1024	
Proposed system	CPU node		One	
			2	
			...	
			1024	
	GPU node		One	
			2	
			...	
			1024	

※ Scale-up prediction is allowed for STRIDE, and the tenderer must submit results on both CPU and GPU nodes.

2.5.5. Run rules

1) The STRIDE should be carried out according to benchmark rules..

2.6. RandomAccess

2.6.1. Overview

1) The RandomAccess is a tool for measuring the random access performance of memory and is part of the HPCC (HPC Challenge) benchmark developed for the HPCS program.
2) Measurement of Memory access time of single process or multi process.

2.6.2. Installation

1) Download hpcc-1.5.0 (https://hpcchallenge.org/hpcc/software/index.html)
2) Using RandomAccess source code in hpcc

2.6.3. Execution

1) Install and run using C compiler

2.6.4. Submission

1) Compile and link settings (makefile)
2) Execution options (e.g., environment variable settings, option settings)
3) Uptates performance from single to multi process(GPUS, Giga Updates Per Second)
4) The total processor and memory structure of the proposed system must be described.

<Table 11> Benchmark Test Results: RandomAccess

Target System		Processor and Memory Type	Common	
			Process	Updates per sec.(GPUS)
BMT system	CPU node		1	
			32	
			1024	
	GPU node		1	
			32	
			1024	
proposed system	CPU node		1	
			32	
			1024	
	GPU node		1	
			32	
			1024	

※ Scale-up prediction is allowed for HPCC RandomAccess, and the tenderer must submit results on both CPU and GPU nodes.

2.7. CLOMP

2.7.1. Overview

1) CLOMP measures OpenMP overheads and other performance effects with multi-threading.

2) Benchmark test for a single node.

2.7.2. Installation

1) Download clomp 1.1.2 (https://openbenchmarking.org/test/pts/clomp-1.1.2)

2.7.3. Execution

1) Install and run using C compiler.

```
$ gcc --openmp -O3 clomp.c -o clomp -lm
```

① Measure default performance

```
$ ./clomp -1 1 64 100 32 1 100
```

② Measure the effect of NUMA on performance

```
$ ./clomp -1 -1 64 100 32 1 100
```

③ Measure the effect of cache on performance

```
$ ./clomp -1 1 64 1 32 100 100
```

④ Measure the effect of memory-bound on performance

```
$ ./clomp -1 1 64 10000 512 1 100
```

2.7.4. Submission

1) Compile and link settings (makefile)

2) Execution options (e.g., environment variable settings, option settings)

3) The ‘Speedup’ values of the Manual OMP for the four execution commands

(Speedup = ‘Elapsed time of parallel code/Elapsed time of serial code’ ; ideal value of speedup is the number of threads)

4) Specifications of all processors must be described.

5) If the test node has accelerator, the performance result of accelerator should be submitted.

<Table 12> Benchmark Test Results: CLOMP

Target System		Processor and Memory Type	Common			
			Default	NUMA	Cache	Mem Bound
BMT System	CPU Node					
	GPU Node					
Proposed System	CPU Node					
	GPU Node					

※ Scale-up prediction is allowed for CLOMP and results from on CPU and GPU nodes must be submitted.

2.7.5. Run rules

1) The CLOMP test should be carried out according to benchmark rules.

3. Comprehensive Performance

3.1. High Performance LINPACK (HPL)

3.1.1. Overview

1) HPL is the LINPACK TPP (Toward Peak Performance) benchmark which measures the performance of floating point operations to solve a linear system of equations.

3.1.2. Method

1) The HPL benchmark test is performed according to the guides specified on the TOP500 website (www.top500.org).

2) Refer to the TOP500 submission website.

   (https://www.top500.org/project/call-for-participation/)

3.1.3. Submission

1) Added or modified files (e.g., makefile, HPL.dat, script file, etc.)

2) Execution environment information

   ① Compiler type and version information, library (MPI, BLAS) type and version information

   ② Execution options (e.g., environment variable settings, mpirun options, etc.)

3) Execution log file with standard output (adjust device out value in HPL.dat file)

4) Benchmark test results according to the following table

<Table 13> Benchmark Test Results: HPL

Target System		Theoretical Performance (FLOPS)	Nodes	Common	
				Matrix Size	Measured Performance (PFLOPS)
BMT System	CPU partition	Nmin			
		...			
		Nmax			
	GPU partition	Nmin			
		...			
		Nmax			
Proposed System	CPU partition				
	GPU partition				▨

※ Scale-out prediction and scale-up prediction are allowed, and the tenderer must submit both CPU and GPU partition results.

※ Results in shaded cells (▨) are used for evaluation.

3.1.4. Run rules

1) The results in the proposed system must be listed on TOP500 after installation.

3.2. High Performance LINPACK-Artificial Intelligence (HPL-AI)

3.2.1. Overview

1) HPL-AI is the benchmark test for solving large linear equations using mixed-precision arithmetic.

3.2.2. Method

1) The HPL-AI benchmark test is performed according to the rules specified in https://hpl-mxp.org/rules.md

2) Allow use of own implementations as well as Reference Implementation (see https://bitbucket.org/icl/hpl-ai/src/main/)

3.2.3. Submission

1) Added or modified files (e.g., makefile, HPL.dat, script file, etc.)

2) Execution environment information

① Compiler type and version information, library (MPI, BLAS) type and version information

② Execution options (e.g., environment variable settings, mpirun options, etc.)

3) Source code in case own implementation is used.

4) Run log file with standard output

5) Benchmark test results according to the following table

<Table 14> Benchmark Test Results: HPL-AI

target system		theoretical Performance (FLOPS)	Nodes	Common	
				Matrix Size	Measured Performance (PFLOPS)
BMT System	GPU Partition	Nmin			
		...			
		Nmax			
Proposed System	GPU Partition				▨

※ Scale-out prediction and scale-up prediction are allowed, and the tenderer must submit the results in the GPU partition

※ Results in shaded cells (▨) are used for evaluation.

3.2.4. Run rules

1) The results in the proposed system must be listed on HPL-AI (https://hpl-mxp.org/results.md) after installation

3.3. GREEN500

3.3.1. Overview

1) Green500 measures LINPACK FLOPS per watt and ranks computers from the TOP500 list of supercomputers in terms of energy efficiency

3.3.2. Method

1) The GREEN500 benchmark test is conducted according to the rules specified in relation to GREEN500 on the TOP500 (www.top500.org) website
2) Refer to the TOP500 submission website (https://www.top500.org/project/call-for-participation/)
3) Perform measurements for each level 2/3 presented in the EEHPC Power Measurement Methodology (https://www.top500.org/static/media/uploads/methodology-2.0rc1.pdf)

3.3.3. Submission

1) Files added or modified when executing HPL in 3.4.1. (e.g., makefile, HPL.dat, script file, etc.)
2) Execution environment information

① HPL execution environment information in 3.4.1.

② If the system setting is different from 3.4.1 HPL Benchmark, specify the information

e.g.) CPU speed, memory settings, internal network settings, etc.

3) Results for each of Level 2/3 mentioned in EEHPC Power Measurement Methodology
4) HPL result information: GFLOPS and core phase time (HPL_pdgesv execution time) information is required
5) Energy efficiency (GFLOPS/W) results according to the following table

<Table 15> Benchmark Test Results: GREEN500

Target System		Theoretical Performance (FLOPS)	Nodes	Common	
				Level	GFLOPS per Watt
BMT System	GPU Partition	Nmin			
		...			
		Nmax			
Proposed System	GPU Partition			2	
				3	

※ Scale-out prediction and scale-up prediction are allowed, and results on the GPU partition must be submitted.

※ Results in shaded cells (▨) are used for evaluation.

3.3.4. Run rules

1) The results in the proposed system must be listed on GREEN500 after installation.

3.4. High Performance Conjugate Gradients (HPCG)

3.4.1. Overview

1) HPCG benchmark is a tool to measure the floating point operation performance through sparse matrix calculation of partial differential equations that are widely used in real applications.

3.4.2. Method

1) The HPCG benchmark test is conducted according to the regulations specified on the HPCG website (http://www.hpcg-benchmark.org)

2) Technical Specification (https://www.osti.gov/servlets/purl/1113870) of HPCG code modification and optimization are permitted under the conditions specified in "6. Permitted Transformations and Optimizations"

3.4.3. Submission

1) Added or modified files (e.g., Makefile, setup/Make.xxx, hpcg.dat, script file, etc.)

2) Execution environment information

① Compiler type and version information, library type and version information

② Execution options (e.g., environment variable settings, mpirun options, etc.)

3) Source file if code optimization is performed.

4) Run log file with standard output (log file and yaml file).

5) Benchmark test results according to the following table

<Table 16> Benchmark Test Results: HPCG

Target System		Theoretical Performance (FLOPS)	Nodes	Common
				Measured Performance (PFLOPS)
BMT System	CPU Partition	Nmin		
		...		
		Nmax		
	GPU Partition	Nmin		
		...		
		Nmax		
Proposed System	CPU Partition			
	GPU Partition			

※ Scale-out prediction and scale-up prediction are allowed, and the tenderer must submit both CPU and GPU partition results.

※ Results in shaded cells (▨) are used for evaluation.

3.4.4. Run rules

1) The results in the proposed system must be listed on the HPCG list after installation.

3.5. GRAPH500

3.5.1. Overview

1) GRAPH500 is a large-scale graph analysis benchmark.

2) It measures the performance of BFS (Breadth-First Search) and SSSP (Single-Source Shortest Path) search through massive graph traversing between randomly chosen vertices in large-scale Kronecker graphs.

3.5.2. Method

1) The GRAPH500 benchmark test is performed according to the regulations in the GRAPH500 website (http://www.graph500.org).

2) Allow optimization within GRAPH500 regulations.

3.5.3. Submission

1) Added or modified files (e.g., make.inc, Makefile, script file, etc.)

2) Details about the execution environment

   ① Compiler type and version information, library type and version information

   ② Execution options (e.g., environment variable settings, mpirun options, etc.)

3) Source files if optimization is performed.

4) Execution log file with standard output and verification result

   ① Graph500 Specifications (https://graph500.org/?page_id=12) "9.3. Information specified in the "Output" section

   ② Graph500 Specifications (https://graph500.org/?page_id=12) "8. Verification result information presented in "Validation"

5) Benchmark test results of SSSP and BFS according to the following table

<Table 17> Benchmark Test Results: GRAPH500

Target System		Field	Theoretical Performance (FLOPS)	Nodes	Common	
					Problem Scale	Edges Per Second (GTEPS)
BMT System	CPU Partition or GPU Partition (partition specified)	SSSP	Nmin			
			...			
			Nmax			
	CPU Partition or GPU Partition (partition specified)	BFS	Nmin			
			...			
			Nmax			
Proposed System	CPU Partition or GPU Partition (partition specified)	SSSP				P_{SSSP}
	CPU Partition or GPU Partition (partition specified)	BFS				P_{BFS}

※ Scale-out prediction and scale-up prediction are allowed, and the tenderer must submit the results in the CPU partition or GPU partition.

※ Results in shaded cells (▨) are used for evaluation.

3.5.4. Run rules

1) The results in the proposed system must be listed on GRAPH500 after installation.

3.6. MLPerf

3.6.1. Overview

1) MLPerf is an performance measurement tool for AI model provided by MLCommons.
2) It consists of eight workloads covering multiple use cases including vision, language, recommendation, and reinforcement learning.

3.6.2. Method

1) The MLPerf BMT is conducted with the reference implementations published in the MLCommons repository (https://github.com/mlcommons/training). In case of modifying the source code, the code must be disclosed according to MLPerf rules.
2) The BMT should be conducted according to MLPerf rules announced in The MLCommons repository (https://github.com/mlcommons/training_policies/blob/master/training_rules.adoc).
3) The BMT of BERT, CosmoFlow, and DeepCAM models should be conducted based on Closed Division.

3.6.3. Submission

1) Added or modified files
2) Execution environment information
   ① Compiler type and version information, library type and version information
   ② Execution options (e.g., environment variable settings, mpirun options, etc.)
3) Source files if they are modified
4) Run log file with standard output
5) "Processor and count, Accelerator and count, Software, Benchmark Results, Details, Code" of the "Submission Information" items on The MLCommons homepage (https://mlcommons.org/en/training-normal-21/)
6) Benchmark test results according to the following table

<Table 18> Benchmark Test Results: MLPerf

Target System		Workload	Theoretical Performance (FLOPS)	Number of Nodes	Benchmark Results (minutes)
BMT System	GPU Partition	BERT	Nmin		
			…		
			Nmax		
		CosmoFlow	Nmin		
			…		
			Nmax		
		DeepCAM	Nmin		
			…		
			Nmax		
Proposed System	GPU Partition	BERT			T_{BERT}
		CosmoFlow			$T_{CosmoFlow}$
		DeepCAM			$T_{DeepCAM}$

※ Scale-out and scale-up predictions are allowed, and the tenderer must submit the results in the GPU partition.

※ Results in shaded cells (▨) are used for evaluation.

3.6.4. Run rules

1) The results in the proposed system must be listed on the MLPerf list after installation. (refer to https://mlcommons.org/en/training-normal-21/)

3.7. P3DFFT (Parallel Three-Dimensional Fast Fourier Transform)

3.7.1. Overview

1) P3DFFT is a Fourier transform library for multi-dimensional space and is used in various application problems such as the spectral method.

2) It measures the performance of conducting FFT in a distributed memory environment for a 3D domain with many Catesian grid points.

3.7.2. Method

1) Download and install the P3DFFT program from the P3DFFT website (https://p3dfft.readthedocs.io/en/latest/), and build the samples/C/driver_sine.c or samples/FORTRAN/driver_sine.F90 file.

2) Execute using 'stdin' and 'dims' files as input files

3) 'stdin' file settings

```
16384 4096 16384 2 1
```

① The 1st, 2nd and 3rd number are the number of grid points in the x, y, z directions, and cannot be modified.

② The 4th number uses 1 or 2 as the dimension of domain decomposition and the contents of the dims file should be modified accordingly.

③ The 5th number is the number of iterations and cannot be modified.

4) 'dims' file settings

```
2 4
```

① If the 4th number of 'stdin' is 1, 'dims' is unnecessary and the 3D domain is decomposed in one direction by the number of MPI processes

② If the 4th number of 'stdin' is 2, a 'dims' file with the above two numbers are required. Each number is the number of domains divided in two directions, and the product of the two numbers must be equal to the number of MPI processes.

③ The content of 'dims' and the corresponding number of MPI processes are freely selectable, and the number of OpenMP threads is also selected freely.

3.7.3. Submission

1) Added or modified files (e.g., make.inc, Makefile, stdin, dims file, etc.)

2) Execution environment information

① Compile and link settings (makefile)

② Execution options (e.g., environment variable settings, mpirun option settings)

③ External FFT library information

3) Execution log file with standard output

4) Benchmark test results (execution time output after "Time per loop = " in the log file) according to the format of the following table

<Table 19> Benchmark Test Results: P3DFFT

Target System		Theoretical Performance (FLOPS)	Number of Nodes Used	Common
				Elapsed Time (sec)
BMT System	Specify the Partition	Nmin		
		...		
		Nmax		
Proposed System	Specify the Partition			T_{P3DFFT}

※ Scale-out and scale-up predictions are allowed, and the tenderer must submit the results in the CPU partition or GPU partition

※ Results in shaded cells (▨) are used for evaluation.

3.7.4. Run rules

1) The source code for the executable file (samples/C/driver_sine.c or samples/FORTRAN/driver_sine.F90) cannot be modified.
2) The message "Results are correct" should be displayed in the result file.
3) External FFT libraries (FFTW, MKL, etc.) used by P3DFFT can be replaced, and in this case, the FFT library must be available to KISTI and KISTI supercomputer users without restrictions.
4) External FFT library must use double precision.
5) Allow optimizations related to communication and memory access.
6) Allow OpenMP and MPI related optimizations.

4. Application software performance

4.1. LAMMPS

4.1.1. Overview

1) Description

① LAMMPS is a general-purpose molecular dynamics program freely available.

② It supports certain types of hardware, including multi-core CPUs, GPUs (using CUDA, OpenCL, HIP, SYCL), and Intel Xeon Phi coprocessors, OpenMP and etc.

③ The scalability test of a single running job and the throughput test of multiple simultaneously running jobs should be conducted.

4.1.2. Installation

1) Install the stable version (2022.06.23 or later) at http://lammps.sandia.gov/ as of the date the RFP was released .

2) Follow the official installation guide at https://docs.lammps.org/Manual.html.

3) Install essential packages (molecule, kspace, rigid) for benchmark testing.

4) Install package for CPU and GPU utilization.

4.1.3. Execution

1) Input files

① Use the data.rhodo and in.rhodo.scaled files in the "lammps-*/bench" directory.

② Modify "in.rhodo.scaled" file.

- Change "run" in "in.rhodo.scaled" to 500
- Basically, pair_style should be “lj/charmm/coul/long 8.0 10.0” and “kspace_style pppm 1e-4”. Employing suffixes for hardware accelerators, such as gpu, intel, kk, omp, opt, and etc., is permitted. If a package not presently included in LAMMPS is introduced and utilized, it is required to provide the newly added package to KISTI at no cost.
- Deactivating interpolation tables with the use of “pair_modify table 0” is permitted.
- The modification of processor mappings with the use of the “processors” command is permitted.
- Additional modification is not permitted.

③ Modification of the “data.rhodo” file is not permitted.

2) Execution

① Scalability test

- run the command below

```
$ executable -var x 40 -var y 40 -var z 40 -in in.rhodo.scaled
```

② Throughput test

- Submit 10,000 of the following jobs to the scheduler

```
$ executable -var x 4 -var y 4 -var z 4 -in in.rhodo.scaled
```

4.1.4. Submission

1) Scalability test

① Installation script or documents on installation method.

② Used input files (in.rhodo.scaled, data.rhodo)

③ Execution log file with standard output

④ Benchmark test results (elapsed time displayed at "Total wall time: " in the execution log file) according to the following table

<Table 20> Scalability Test Results: LAMMPS

Target System		Theoretical Performance (TFLOPS)	Use Number of Nodes	Common
				Elapsed Time (sec)
BMT System	GPU Partition	Nmin		
		...		
		Nmax		
Proposed System	GPU Partition			$T_{LAMMPS,scale}$

※ Scale-out and scale-up predictions are allowed for LAMMPS scalability test, and results on the GPU partition must be submitted.

※ Results in shaded cells (▨) are used for evaluation.

2) Throughput test

① Installation script or documents on installation method (can be replaced by scalability test result)

② Used input files (in.rhodo.scaled, data.rhodo) and job scripts

③ Execution log file containing log records and standard output of the scheduler

④ Benchmark test results (execution time displayed in the scheduler log file) according to the following table

<Table 21> Throughput Test Results: LAMMPS

Target System		Theoretical Performance (TFLOPS)	Number of Nodes Used	Common	
				Number of Jobs	Elapsed Time (sec)
BMT System	GPU Partition	Nmin			
		...			
		Nmax			
Proposed System	GPU Partition			10,000	$T_{LAMMPS,throughput}$

※ The execution time is the time elapsed from the start of the first task to the end of the last task on the scheduler.

※ Scale-out and scale-up predictions are allowed for LAMMPS throughput test, and the tenderer must submit the results on the GPU partition.

※ Results in shaded cells (▨) are used for evaluation.

4.1.5. Run rules

1) Essentially, source code modifications and optimizations must follow the general rules in Section 1, but additional following rules should be obeyed.

   ① Vectorization and the data structure modifications for vectorization are allowed.

   ② Optimization using other programming languages (OpenCL, CUDA, among others), and data structure modifications for a corresponding programming language are allowed.

   ③ Modifications of the subroutine and function parameters when calling an external library are allowed. In such a case, an open API (Application Program Interface) for the library must be submitted.

   ④ Modifications of the subroutine or function parameters except the above ② and ③ are not allowed.

2) The potential energy values after optimization must agree with those before the optimization within the accuracy of at least five significant figures.

3) Only double precision is permitted.

4.2. PELE-LMeX

4.2.1. Overview

1) PELE-LMeX, which is an application in energy fields of the Exascale Computing Project (ECP), is an open-source turbulent flow combustion solver that includes chemical reactions such as combustion, based on adaptive meshes.
2) It finds numerical solutions for chemical reactions and 3D Navier-Stokes equations at low Mach numbers using block-structured adaptive mesh refinement.
3) It is for the scalability tests of single job.

4.2.2. Installation

1) Refer to PELE-LMeX official website.
   (https://amrex-combustion.github.io/PeleLMeX/manual/html/index.html)
2) After cloning the PeleProduction repository (Releases v23.03), initialize and update the submodule in the installation folder.

```
$ git clone –b v23.03 https://github.com/AMReX-Combustion/PeleLMeX.git
$ cd PeleLMeX
PeleLMeX$ git submoudle init
PeleLMeX$ git submoudle update
```

4.2.3. Execution

1) Use the FlameSheet problem as a BMT problem, compile source codes in the problem folder to create an executable file (you can copy and use the problem folder).

```
PeleLMeX$ cd Exec/RegTests/FlameSheet/
```

2) Modify compile options of Make file (GNU makefile)

```
# Change the path settings
TOP = ../../..
PELELMEX_HOME       ?= ${TOP}
AMREX_HOME          ?= ${PELELMEX_HOME}/Submodules/amrex
PELE_PHYSICS_HOME  ?= ${PELELMEX_HOME}/Submodules/PelePhysics
AMREX_HYDRO_HOME    ?= ${PELELMEX_HOME}/Submodules/AMReX-Hydro
# AMReX options
DIM                 = 3
DEBUG               = FALSE
PRECISION           = DOUBLE
VERBOSE             = FALSE
TINY_PROFILE        = FALSE
# Compilation
COMP                = gnu
USE_MPI             = TRUE
USE_OMP             = FALSE
USE_CUDA            = FALSE
USE_HIP             = FALSE
# PeleLMeX
```

```
USE_EFIELD         = FALSE
# PelePhysics
Chemistry_Model  = drm19
Eos_Model          = Fuego
Transport_Model  = Simple
include $(PELELMEX_HOME)/Utils/Make.PeleLMeX
```

3) Make: Third party libraries such as sundials can be used. Thread option (-j) can be modified

```
$ make -j 4 TPL
```

4) Create PeleLMeX**.***.***.ex file in the problem folder. (An executable file name is depending on compiler and its options. PeleLM3deX.gnu.***.MPI.ex is created when using the above settings)

5) Tests must be conducted with the input file provied by KISTI.

4.2.4. Submission

1) Accuracy result

① Compare the values of BinCenter, temp, Volume, HeatRelease_Avg on 34 line in the output named “condTest000010.dat” with the reference values.

<Table 22> Acceptable Range of Results

Reference Value	Tolerance Range

BinCenter= 1.13167689e+03
temp = 1.12617797e+03
Volume = 6.40000000e-08
HeatRelease_Avg = 6.57849001e+08
	The values must match up to 8 significant figures.
	(5 significant figures for GPU)

2) Information on modified files

3) Execution environment information: compile and execution settings (GNU makefile and input.3d* files)

4) Execution log file with standard output: Confirmation of completion with the last AMRex(*****) finalized statement in the standard output.

5) Benchmark test results (Sum of elapsed time of each time step displayed at ">> PeleLM:Advance()--> Time: *****" after NEW TIME STEP lines in the execution log file)

<Table 23> Scalability Test Results: PeleLM

Target System	Partition	Theoretical Performance (FLOPS)	Number of Nodes Used		Common
			Nodes	CPUs/node	Elapsed Time (sec)
BMT System	CPU Partition	Nmin			

		…			
		Nmax			
	GPU Partition	Nmin			
		…			
		Nmax			
Proposed System	CPU Partition				$T_{PeleLM,CPU}$
	GPU Partition				$T_{PeleLM,GPU}$

※ Scale-out and scale-up predictions are allowed, and the tenderer must submit both CPU and GPU partition results.

※ Results in shaded cells (▨) are used for evaluation.

4.2.5. Run rules

1) "AMRex(*****) finalized" message should be displayed in the execution log file.
2) The input files provided by KISTI cannot be modified.
3) Essentially, source code modifications and optimizations must follow the general rules in Section 1, but additional following rules should be obeyed.
   ① MPI-related optimizations are allowed.
   ② Vectorization and the data structure modifications for vectorization are allowed.
   ③ Optimization using other programming languages (OpenCL, CUDA, among others), and data structure modifications for a corresponding programming language are allowed.
   ④ Modification of the subroutine and function parameters when calling an external library are allowed. In such a case, an open API (Application Program Interface) for the library must be submitted.
   ⑤ Modifications of the subroutine or function parameters except the above ② and ③ are not allowed.

4.3. PySCF

4.3.1. Overview

1) PySCF is an open source program for electronic structure calculation, which uses various post-Hartree Fock methodologies using Gaussian-type basis functions.
2) It is developed in Python and does not support MPI and accelerator (gpu4pyscf and mpi4pyscf plugins are not allowed.)
3) It is for the throughput test of multiple job runs.

4.3.2. Installation

1) Refer to homepage (https://pyscf.org/install.html)
2) All of the various installation methods described on the official website are allowed.

(Building from the source code and using pip, conda, and docker are all permitted)

4.3.3. Execution

1) Execute by referring to the ./BMT_PySCF directory

2) Run the individual task given in python script (run.py).

3) Submit a given number of test cases to the scheduler (individual tasks are configured to execute the Python script once)

4.3.4. Submission

1) Accuracy result: the output of SCF energy value and CASSCF energy value must show a difference of 1e-3 or less from the reference value (SCF energy = -878.1746 & CASSCF energy = -878.2462)

2) Installation script or documents on installation method

3) Input files and scheduler input scripts used

4) Execution log file containing standard outputs of the PySCF and scheduler

5) Benchmark test results according to the following table

① Elapsed time in the scheduler log file between the beginning of the first job and the end of the last job.

② It is not allowed to change the number of nodes used for performance prediction and the number of test cases.

<Table 24> Throughput Test Results: PySCF

Target System		Theoretical Performance (TFLOPS)	Number of Nodes Used	Common	
				Test Case Number of Jobs	Elapsed Time (sec)
BMT System	CPU Partition	Nmin			
		...			
		Nmax			
Proposed System	CPU Partition		1200	4800	$\boldsymbol{T_{PySCF}}$

※ Scale-up prediction is allowed, and the tenderer must submit the results on the CPU partition

※ Results in shaded cells (▨) are used for evaluation.

4.3.5. Run ruless

1) Source code modifications and optimizations must follow the general rules in Section 1, but additional following rules should be obeyed. (Select version v2.2.0 or later shown at the official PySCF website, https://github.com/pyscf/pyscf/tags)

① Performance optimization through environment variable settings is permitted, but any changes in run.py and pyscf source code are not allowed.

4.4. Quantum Espresso (QE)

4.4.1. Overview

1) Quantum Espresso is an electronic structure calculation program for plane-wave based density functional theory calculation.
2) It is developed in Fortran and supports MPI and OpenMP.
3) Version 6.4.1 or later should be used.
4) It is for a scalability test with a single task..

4.4.2. Installation

1) Install by referring to the official repository (https://github.com/QEF/qe).

4.4.3. Execution

1) Download the benchmark repository (https://github.com/QEF/benchmarks.git).
2) Run the pw.x executable file using the given input file (pw.in) in the CNT10POR8 folder.

4.4.4. Submission

1) Accuracy result: The result is accepted only when the total energy value of each SCF step in the execution log file is within 5e-3 compared to those of the baseline runs.
2) Installation script or documents on installation method
3) Input file used
4) Run log file with standard output
5) Benchmark test results according to the following table

① Wall time displayed at "PWSCF : OOO CPU time, OOO wall time" in the execution log file.

<Table 25> Scalability Test Results: QE

Target System		Theoretical Performance (TFLOPS)	Number of Nodes Used	Common
				Elapsed Time (sec)
BMT System	GPU Partition	Nmin		
		…		
		Nmax		
Proposed System	GPU Partition			T_{QE}

※ Scale-out and scale-up predictions are allowed, and the tenderer must submit the results in the GPU partition

※ Results in shaded cells (▨) are used for evaluation.

4.4.5. Run rules

1) Source code modifications and optimizations must follow the general rules in Section 1, but additional following rules should be obeyed.

① Vectorization and the data structure modifications for vectorization are allowed. ② Optimization using other programming languages (OpenCL, CUDA, among others), and data structure modifications for a corresponding programming language are allowed.

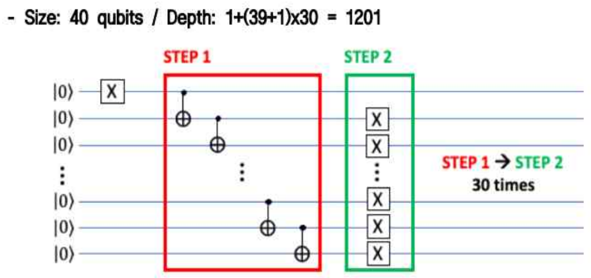
③ Mmodifications of the subroutine and function parameters when calling an external library are allowed. In such a case, an open API (Application Program Interface) for the library must be submitted.

④ Modifications of the subroutine or function parameters except the above ② and ③ are not allowed.

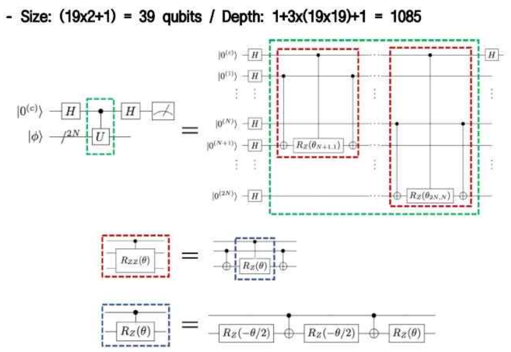
4.5. QUEST

4.5.1. Overview

1) QUEST is a quantum circuit simulation program with classical computing.
2) It supports multi-threading and MPI-based parallelization, and can run in GPU environment.
3) Problem 1 using QUEST is a 40-qubit quantum circuit that continuously performs the universal gate CNOT and X operation, and Problem 2 is a 39-qubit quantum circuit that calculates the permanent of a matrix.



(a) Problem 1 circuit



(b) Problem 2 circuit

<Figure 1> QUEST Problem Sets (a) 40-qubit Quantum Circuit (b) 39-qubit Quantum Circuit

4.5.2. Installation

1) Refer to the official website (https://quest.qtechtheory.org/). There is no input file, and the source code should be recompiled with a distributed file containing the main function for problems, modifying Makefile for test problems

4.5.3. Execution

1) Compile the source files which will be provided by KISTI for the given problem and run them

4.5.4. Submission

1) Accuracy result

① For problem 1, the ㅣ11.....11> status value in every loop should be as follows:

```
loop = 1
Probability amplitude of |1...1>: 1
Probability amplitude of |1...1>: 0
...
loop = 30
Probability amplitude of |1...1>: 1
Probability amplitude of |1...1>: 0
```

② For problem 2, the probability that the state of qubit 0 is ㅣ0〉 or ㅣ1〉 should be as follows:

```
----------------------------------
Printing Amplitudes & Finishing...
----------------------------------
Probability of qubit 0 being in state 0: 0.522802
Probability of qubit 0 being in state 1: 0.477198
```

2) Installation script or documents on installation method.

3) 'main function' source file used.

4) Run log file with standard output.

5) Benchmark test results according to the following table (Input the sum of "Allocation & Initialization", "Circuit Operations", and "Deallocation" times displayed at the bottom of "Time components" of the execution log).

<Table 26> Scalability Test Results: QUEST

Target System		Problem	Theoretical Performance (TFLOPS)	Number of Nodes Used	Common
					Elapsed Time (sec)
BMT System	Specify the Partition	40-qubit quantum circuit	Nmin		
			...		
			Nmax		
		39-qubit quantum circuit	Nmin		
			...		
			Nmax		
Proposed System	Specify the Partition	40-qubit quantum circuit			$T_{QUEST40}$
		39-qubit quantum circuit			$T_{QUEST39}$

※ Scale-out and scale-up predictions are allowed, and the tenderer can submit the results in the desired partition (CPU, GPU, large memory node, etc.).

※ Shaded marks (▨) are items used for evaluation score computation

4.5.5. Run rules

1) The distributed source files, bmt01.c and bmt02.c corresponding to the main function, cannot be modified.

2) Source code modifications and optimizations must follow the general rules in Section 1, but additional following rules should be obeyed.

① Vectorization and the data structure modifications for vectorization are allowed.

② Optimization using other programming languages (OpenCL, CUDA, among others), and data structure modifications for a corresponding programming language are allowed.

③ Modifications of the subroutine and function parameters when calling an external library are allowed. In such a case, an open API (Application Program Interface) for the library must be submitted.

④ Modifications of the subroutine or function parameters except the above ② and ③ are not allowed.