ANL/ALCF-18/2

Evaluation of the Intel-QS performance on Theta supercomputer

Computational Science Division

About Argonne National Laboratory

Argonne is a U.S. Department of Energy laboratory managed by UChicago Argonne, LLC under contract DE-AC02-06CH11357. The Laboratory's main facility is outside Chicago, at 9700 South Cass Avenue, Argonne, Illinois 60439. For information about Argonne and its pioneering science and technology programs, see www.anl.gov.

ANL/ALCF-18/2

Evaluation of the Intel-QS performance on Theta supercomputer

prepared by
Yuri Alexeev
Computational Science Division, Argonne National Laboratory

April 11, 2018

Introduction

Intel-QS is a freely available quantum simulator under Apache 2.0 license on Github (https://github.com/intel/Intel-QS). The scope of this work is to describe work done on porting, optimization, and benchmarking of this code on Cray/Intel supercomputer Theta located in Argonne Leadership Computing Facility.

We are interested in a quantum simulator because it can be used, for example, to develop new quantum algorithms and quantum computers, to study and optimize new quantum circuits, to investigate the performance of circuits in the presence of noise, and to develop new error correction schemes.

Intel-QS is a distributed high-performance C++ implementation of a quantum linear algebra simulator on a classical computer; it is formerly known as qHiPSTER [1]. It is programmed to take full advantage of multi-core and multi-node architectures. The code is capable of simulating 1 and 2 qubit quantum logical gates, which are the building blocks of quantum circuits. The parallelization of the code is done by using MPI and OpenMP.

The key feature of Intel-QS is that it stores only 2^N (N is number of qubits) state vector instead of 2^N*2^N density matrix to reduce memory footprint. The gate operations, which are represented by matrices are multiplied on state vector. As an example, a quantum single-qubit gate operation on qubit k can be represented by the following unitary transformation equivalent to the outer product of matrices:

$$U = I \otimes I \ldots \otimes Q_k \otimes \ldots I \otimes I$$

where I is an identity matrix and Q_k is a 2x2 unitary matrix

$$Q_k = \begin{pmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{pmatrix}$$

Matrix U is sparse and there is no need to construct it. A much better approach is to apply Q_k matrix operation directly on the state vector.

The key features of the algorithm are that it requires a massive amount of memory to store state vector. For example, a 45-qubit simulation will require to store 2^45 double precision state vector, which takes 0.5 PB of memory. The second key feature is that the gate operations are relatively computationally cheap, but most values of state vector need to be updated with every gate operation. Thus, quantum simulators are bound either by memory bandwidth on a single node or by network bandwidth for multi-node simulations.

Accomplished work

The following work has been accomplished for porting and optimization of Intel-QS on Theta supercomputer:

1. Enabled large simulations by interfacing code with BigMPI library and fixing overflowing integers
2. Fixed QASM input bug for multi-node jobs
3. Added new gates to be able to run chemistry circuits
4. Added Intel-QS support in ProjectQ
5. Implemented a version of the Intel-QS QASM interface that utilizes the existing Intel-QS noisy qubit class, which allows to set the noise model with a random seed, amplitude, and phase damping parameters
6. Developed Quantum Fourier and Quantum Chemistry benchmarks
7. Benchmarked Intel-QS on Theta, Atos, QLM, Gomez, and Skylake

It is not a complete list and more work will be done on Intel-QS. As a result of this work, Intel-QS is fully operational on Theta and JLSE machines for users.
The future works involves reconstruction of a density matrix from Intel-QS noisy interface state vector. We also plan to add an option for the noise to be applied selectively at specified gates, possibly through specification in the QASM.

Benchmarking results

To get a better understanding of Intel-QS performance, a number of benchmarks were ran on Theta. In the first benchmark, a single Hadamard gate was applied to every qubit to establish baseline for memory requirements.

Qubits	Intel-QS memory requirements, GB	Ideal memory requirements, GB	Theta nodes	Time, sec
33	131	128	1	270
34	394	256	2	391
35	788	512	4	458
36	1576	1024	8	526
37	3152	2048	16	604
38	6304	4096	32	662
39	12608	8049	64	728
40	25216	16098	128	796
41	50432	32196	256	889
42	100864	64392	512	960
43	201728	128784	1024	1046
44	403456	257568	2048	1278
45	806912	515136	4096	1632

Table 1. Single Hadamard gate benchmark on Theta.

As it is shown in Table 1, Intel-QS requires about ½ more memory relative to the ideal requirements to store the state vector. The additional memory is used for buffers. It is needed for an efficient update of the state vector across nodes. In particular, each local state vector on a node is logically partitioned into two halves. Nodes perform pairwise

exchange of the halves until all halves are updated. The advantage of this approach is that it is easy to implement and the load balance between nodes is almost perfect. The obvious disadvantage is the additional memory requirements to store halves from other nodes. For example, it is possible to reduce the size of the buffer to store only ¼ or less of the local state vector. Memory requirements for the buffer will decrease by a factor of two or more, but the time to solution will increase because of additional MPI traffic to update the state vector. Even in the current implementation we were able to run 45-qubit simulation, which required 0.8 PB of memory with the ideal requirement of 0.5 PB.

To understand Intel-QS performance on a single node, MPI ran was set to one (to maximize available memory) and number of threads were varied for a quantum chemistry 30 qubit simulation.

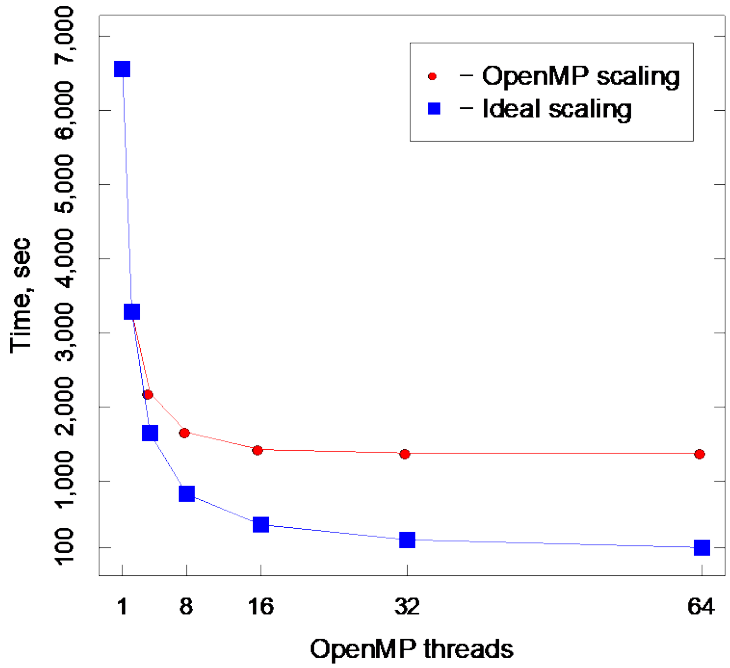

Figure 1. Time to solution for a quantum chemistry 30 gate circuit running on 30 qubits. Number of threads were varied from 1 to 64 for 1 MPI rank.

As it is shown on Figure 1, Intel-QS scales up to 32 threads for 30 gate circuit running on 30-qubits. We chose intently a relatively small simulation to be able to fit in the memory on a single Xeon Phi node. There is a minor improvement in time for solution going from 32 to 64 threads. In all following benchmarks, we used 1 MPI rank and 64 threads to run simulations.

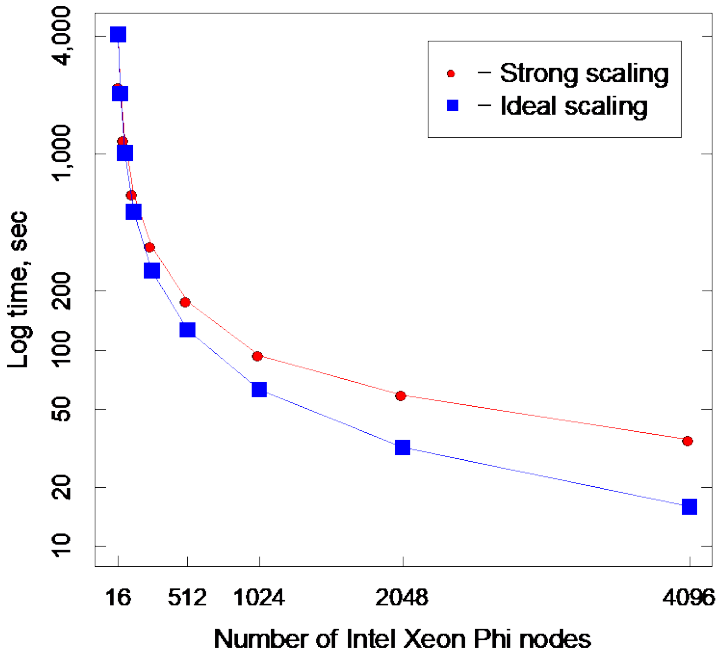

Figure 2. Time to solution for a 35-qubit simulation. A single Hadamard gate per qubit was executed and measured. The number of MPI ranks was varied from 1 to 4,096 MPI ranks with 64 OpenMP threads per rank.

To benchmark MPI performance of the code, 35-qubit simulation was ran on up to 4,096 Theta nodes. The code scales compared to OpenMP threading almost perfectly as shown on Figure 2. A relatively small qubit simulation was chosen to be able to fit in the memory for a small number of nodes. In our case, we started with 16 nodes. It is possible that scaling behavior may change with a larger number of qubits and a different type of the circuit.

Benchmarking of weak scaling is complicated for quantum simulators. The complexity of calculations scales in two directions: the number of qubits and the gate depth. For the first weak scaling, the idea is to keep ratio 2^N divided by the number of nodes fixed. The circuit is kept simple – just an execution of a single Hadarmard gate on each qubit and its measurement. A simple math shows that to keep ratio 2^N/nodes fixed means following: if 45 qubit-simulation was ran on 4,096 nodes then 44 qubit simulation needs to be run on 2,048 nodes, 43 qubit simulation requires 1,024 nodes and so on. The results are shown on Figure 3, where the number of qubits scaled from 37 on 16 nodes to 45 qubits on 4,096 nodes. It was found that this metric is not especially useful since 2^N/nodes does keep amount of work constant as the number of nodes and qubits increase. The relationship is not linear and cannot be easily quantified due to sparse nature of state vector and operations upon it.

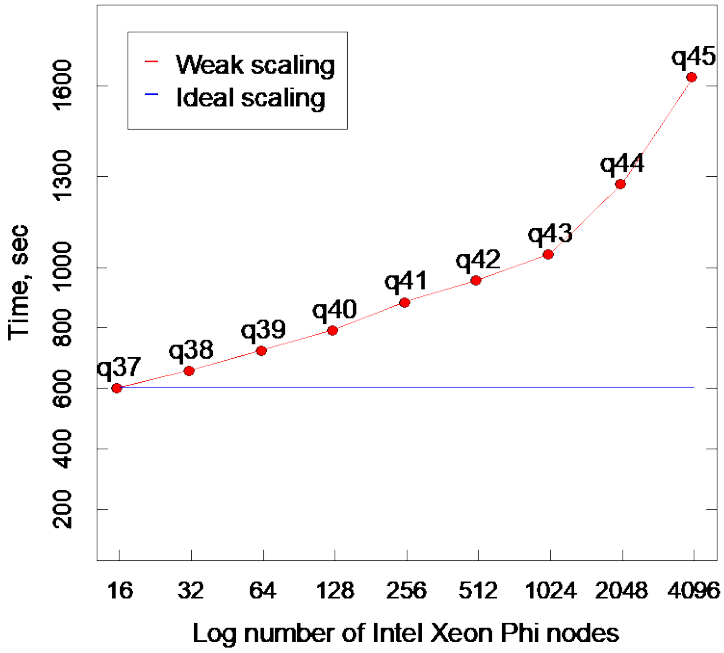

Figure 3. Weak scaling plot showing time to solution vs number of Theta nodes as the number of qubits increases.

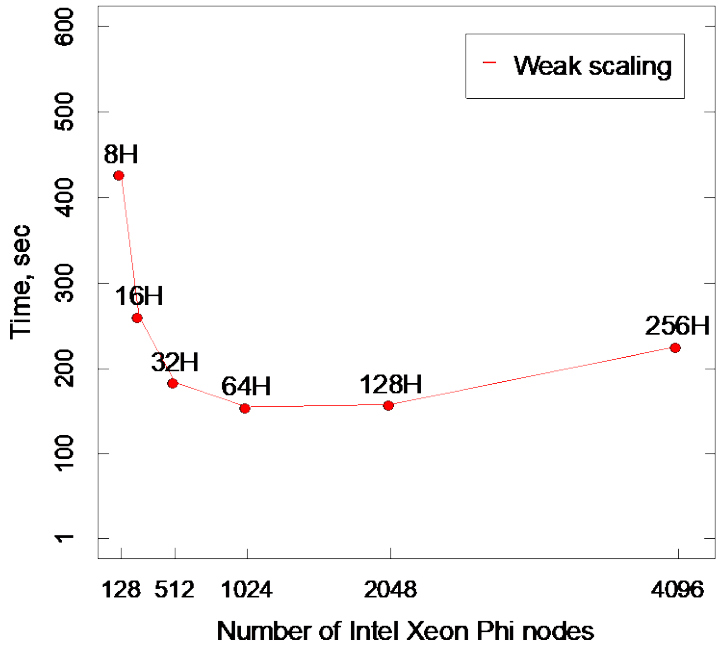

Figure 4. This weak scaling figure shows how the time to solution scales with number of Theta nodes as the number of Hadamard gates is increased.

A better weak scaling benchmark is to increase the number of Hadamard gates with the number of Theta nodes because of a linear relationship between them. It has been shown on Figure 4, where the number of Hadamard gates was scaled from 8 on 128 Theta nodes to 256 on 4,096 Theta nodes. It was found that on small and large numbers of nodes the relationship is not linear and the time to solution is taking longer than expected.

Gates	Time, sec
2	563
4	1,068
10	645
50	1,050
100	2,028
200	3,089
400	~86,400*

Table 2. Time to solution for a mix of CNOT and Hadamard gates on a single Theta node. * - time to solution is estimated.

To estimate the number of gates that could be run in 24 hours (maximum available time in any queue), a benchmark was run on a single Theta node. The results are shown in Table 2. The chosen benchmark has a mix of CNOT and Hadamard gates and it is for 35 qubit-simulation to be able to fit in the memory of a single node. The relationship between the number of gates and the time to solution is linear where there are more than 50 gates. It allows to approximate the maximum number of gates, which can be executed over 24 hours and it is equal to about 400 gates. Obviously, this number will be a lot smaller for the circuits executed across the nodes with a larger number of qubits. It also might be possible to increase performance with further optimization of the code for Xeon Phi architecture.

IBM's 'quantum volume' benchmark [2] combines both qubit and gate weak scaling benchmarks. It might be a more accurate way to study performance of a quantum simulator, which we plan to address in our future work.

Conclusions

This report presents the performance of Intel-QS quantum simulator on Cray/Intel Theta supercomputer. A number of strong and weak scaling benchmarks were performed. It was found that the code scales over MPI ranks across Theta nodes with less than optimal OpenMP scaling on a single node. It was also found that up to 400 gates can be executed on Theta. The additional code development is required to improve performance of the code.

Acknowledgements

This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357.

References

[1] Mikhail Smelyanskiy, Nicolas P. D. Sawaya, and Alán Aspuru-Guzik. 2016. qHiPSTER: The Quantum High Performance Software Testing Environment. 1–9. Retrieved from http://arxiv.org/abs/1601.07195

[2] Lev S Bishop, Sergey Bravyi, Andrew Cross, Jay M Gambetta, and John Smolin. 2017. Quantum Volume. Retrieved April 11, 2018 from https://pdfs.semanticscholar.org/650c/3fa2a231cd77cf3d882e1659ee14175c01d5.pdf

Computational Science Division
Argonne National Laboratory
9700 South Cass Avenue, Bldg. 240
Argonne, IL 60439

www.anl.gov

Argonne National Laboratory is a U.S. Department of Energy
laboratory managed by UChicago Argonne, LLC