Quantum factorization simulation as a benchmark for HPC

Santiago I Betelu

UNT Mathematics Department (adjunct)

Data Vortex Technologies (chief scientist)

Why quantum factorization as a benchmark

- Quantum factorization a widely known & relevant problem, it is known to be hard to simulate
- Good to quantify the ability of a computer to simulate ideal quantum circuits
- The output is reproducible, easy to validate and understand
- Each additional qubit doubles RAM usage, CPU power and internode communication: excellent to test large machines
- Runs in a reasonable time: 1/2-3 hours
- Minimal portable code with ~300 lines of C and MPI
- It just runs: no input or special knowledge from user

Summary of the benchmark

- Simulates a quantum computer with state $|\psi\rangle = \sum_{x=0}^{(2^Q-1)} c_x\, |x\rangle$
- The test consists on running a simplified Shor's algorithm* with increasing number of Q qubits until resources are exhausted.
- Only timing of Fourier Transform AQFT, not the modular exponentiation
- For each Q run the simplified Shor's algorithm to factorize an integer n

$$n \;=\; p \cdot q$$

- p and q are chosen to maximize the period r of $2^x \; mod \; n$

  with the constraints $n^2 \leq 2^Q < 2n^2$ with r even and $2^{r/2} \neq -1 \; mod \; n$
- Then verify that the total probability under peaks is larger than $1/2$

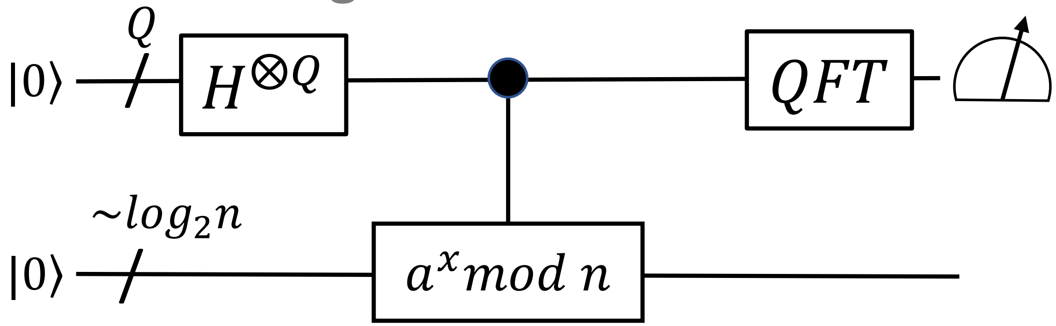
$$P = \sum_{peaks} |c_x|^2 \quad > \frac{1}{2}$$

- Peaks located at $x = \frac{2^Q}{(p-1)(q-1)} \times \text{integer}$

* PW Shor, "Polynomial-Time algorithms for prime Factorization and Discrete Logarithms on a Quantum Computer", SIAM J. Comp 1997

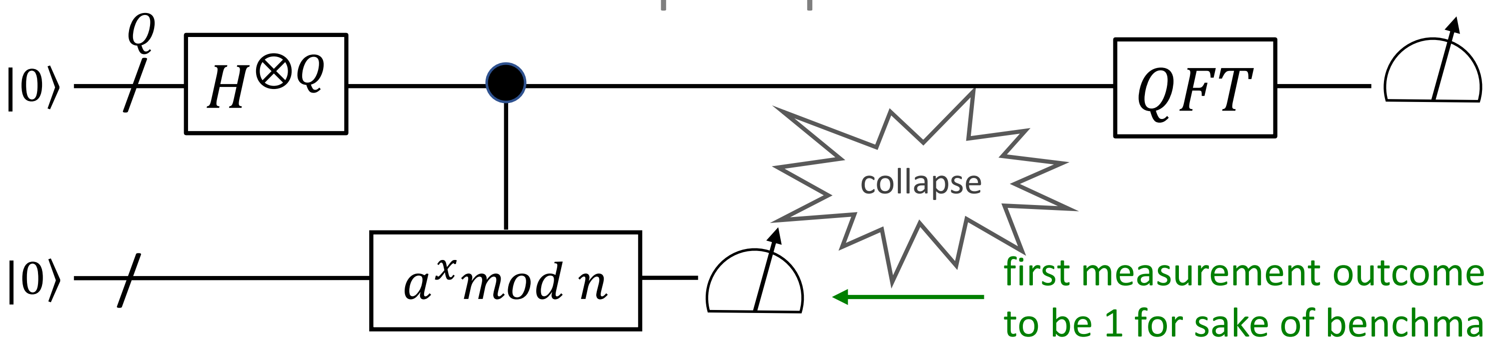
Use deferred measurement principle to save qubits

Original Shor’s algorithm



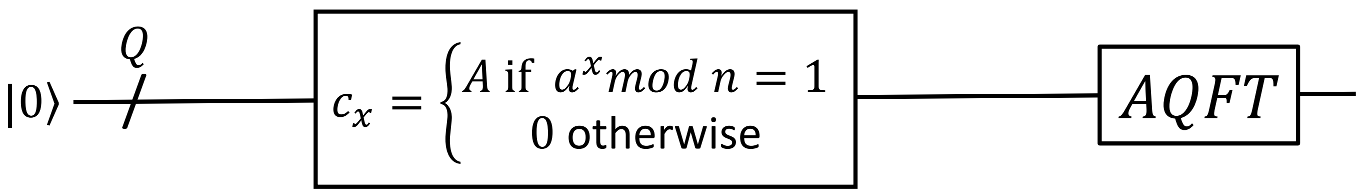
This measurement is inconvenient in a benchmark

With deferred measurement principle



first measurement outcome chosen to be 1 for sake of benchmarking

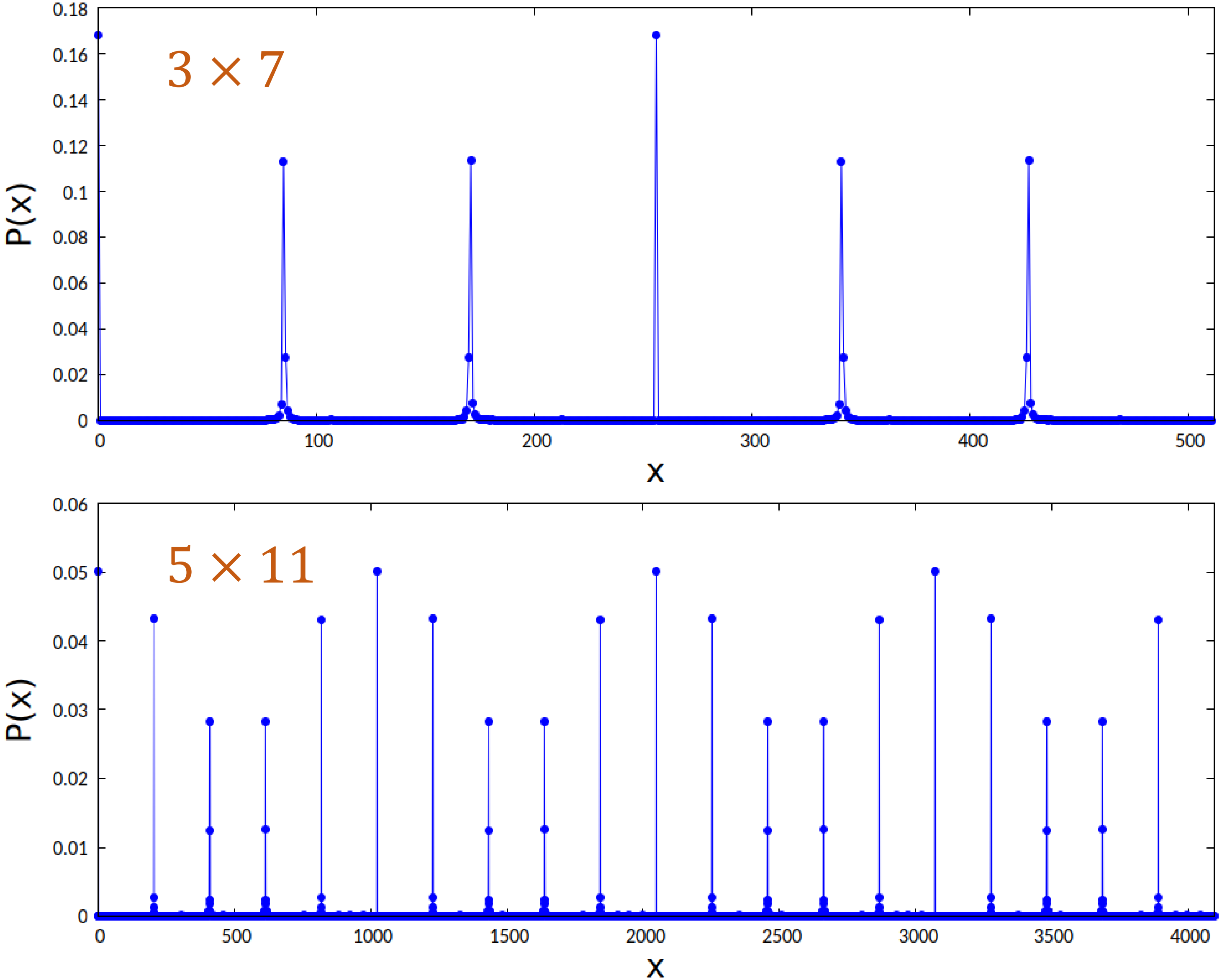
This benchmark saves $log_2 n$ qubits, only AQFT is timed



Do not measure, just check peak locations

A is a normalization const.

Verify location of peaks for each $n = p \cdot q$



List of factorizations used in the test

$n = p \cdot q$ chosen to maximize period r of $2^x \ mod \ n,\ n^2 \leq 2^Q < 2n^2$ with r even and $2^{r/2} \neq -1 \ mod \ n$

Q	$p \times q$
9	3×7
10	3×7
11	3×13
12	5×11
13	7×11
14	5×23
15	11×13
16	11×23
17	5×71
18	19×23
19	5×139
20	19×53
21	23×61

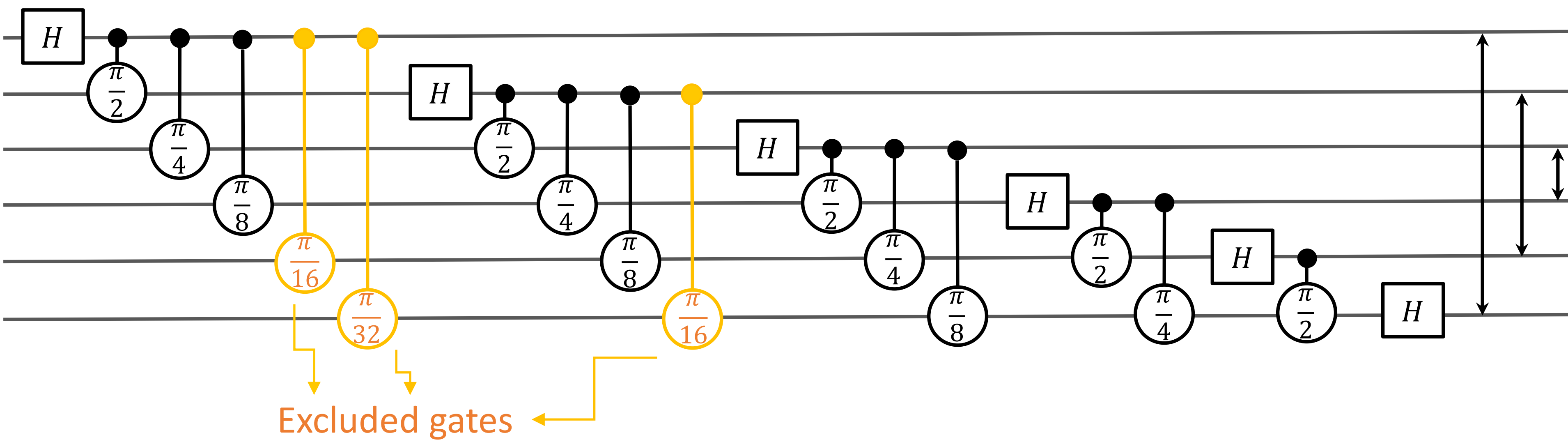
Q	$p \times q$
22	29×67
23	47×61
24	29×139
25	29×199
26	47×173
27	71×163
28	83×197
29	79×293
30	103×317
31	149×311
32	101×647
33	149×619
34	269×487

Q	$p \times q$
35	167×1109
36	479×547
37	479×773
38	367×1427
39	859×863
40	563×1861
41	1039×1427
42	947×2213
43	1307×2269
44	2027×2069
45	2039×2909
46	2357×3559
47	2237×5303

Q	$p \times q$
48	3917×4283
49	4127×5749
50	4813×6971
51	6173×7687
52	6029×11131
53	7243×13103
54	10357×12959
55	12757×14879
56	11399×23549
57	19427×19541
58	20771×25847
59	24847×30557
60	27779×38653

Use AQFT* because it is faster than QFT

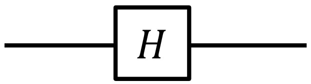
- For each H gate we compute $1 + log_2 Q$ controlled phase gates
- Phase gates computed with one memory access per state
- Complexity AQFT is $\mathrm{O}(\mathrm{Q} \log \mathrm{Q})$ versus $\mathrm{O}(Q^2)$ for QFT



Results in the approximation $f_y \cong \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} c_x e^{2\pi i x y/N}$

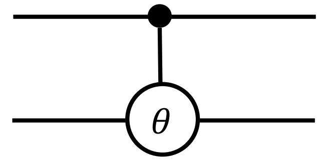
* D Coppersmith, An Approximate Fourier Transform Useful in Quantum Factoring, IBM report 1994
A Barenco et al, Approximate Quantum Fourier Transform and Decoherence, 1996

AQFT uses 3 types of gates



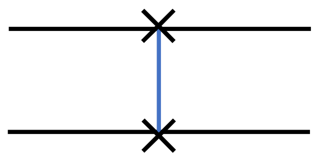
$$= \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Hadamard



$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\theta} \end{bmatrix}$$

Controlled phase



$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Swap qubits

- The essentially heavy part of computation is the Approximate Fourier Transform, computed simulating gates
- Modular exponentiation $a^x \bmod n$ is not implemented gate-by-gate and it is not timed. We compute directly in C, and the computation time is negligible.
- Exponentiation could be implemented with stabilizer-group gates, thus simulated efficiently, not in this benchmark

How are the coefficients distributed between nodes

- The state of the quantum registers $|\psi\rangle = \sum_{x=0}^{(2^Q-1)} c_x\, |x\rangle$ is distributed among nodes
- A basis state is described in binary as

$$|x\rangle = |b_Q b_{Q-1} \ldots b_{Q-M+1}\rangle \otimes |b_L b_{L-1} \ldots\ b_2\ b_1\rangle$$

- The black digits denote indices of amplitudes within each node.
- The blue digits denote node index.
- The number of nodes is $p = 2^M \quad Q = L + M$
- For example the Hadamard gate at qubit q combines the amplitudes c_x and c_y where x and y differ only in bit q.
- Depending on the value of q, we may be combining data between nodes or within node

Example H gate with *Q=5, L=3, M=2*

H(3): same node

c_0 00 000	c_8 01 000	c_{16} 10 000	c_{24} 11 000
c_1 00 001	c_9 01 001	c_{17} 10 001	c_{25} 11 001
c_2 00 010	c_{10} 01 010	c_{18} 10 010	c_{26} 11 010
c_3 00 011	c_{11} 01 011	c_{19} 10 011	c_{27} 11 011
c_4 00 100	c_{12} 01 100	c_{20} 10 100	c_{28} 11 100
c_5 00 101	c_{13} 01 101	c_{21} 10 101	c_{29} 11 101
c_6 00 110	c_{14} 01 110	c_{22} 10 110	c_{30} 11 110
c_7 00 111	c_{15} 01 111	c_{23} 10 111	c_{31} 11 111
Node 0	Node 1	Node 2	Node 3

H(3): same node

H(4): Inter node

How to use the benchmark (MPI version)

This is system dependent. In computers with mpicc and slurm, first prepare a batch file with number of nodes and number of tasks equal to a power of 2 and then launch

```
#SBATCH –o output-8nodesx64cores
#SBATCH --nodes=8     # 8 nodes
#SBATCH –n 256        # 64 cores per node
#SBATCH –p normal     # for KNL processors
#SBATCH –t 02:00:00   # usually less than 2 hours
ibrun ./quansimbench
```

```
> mpicc -Ofast quansimbench.c -o quansimbench -lm -Wall
> sbatch quansimbench.batch
```

May need to add processor specific options such as

–xCORE-AVX512 for SKX

–xMIC-AVX512   for KNL

and may use OpenMP as well.

Typical output

- 32768 KNL cores
- 512 nodes
- memory limited
- less than 1h run

Memory exhausted
Qubits=42
States/s=1.2124e12

Main metrics

Quansimbench version 1.0
Ranks: 32768

Qubits	Factors	Probability	Time	States/s	States/s/rank	Pass
15	7*23	0.727522	1.5009e-01	1.5719e+07	4.7971e+02	yes
16	11*23	0.760696	2.3128e-03	2.5219e+09	7.6962e+04	yes
17	11*31	0.675652	1.1153e-03	1.1165e+10	3.4072e+05	yes
18	17*29	0.756086	4.6819e-04	5.7110e+10	1.7429e+06	yes
19	23*31	0.749676	1.3307e-03	4.2551e+10	1.2986e+06	yes
20	19*53	0.765709	5.7523e-04	2.0963e+11	6.3974e+06	yes
21	23*61	0.765677	7.7942e-04	3.2557e+11	9.9355e+06	yes
22	23*89	0.678047	1.3810e-03	3.8876e+11	1.1864e+07	yes
23	43*67	0.763136	1.3978e-03	8.0418e+11	2.4541e+07	yes
24	61*67	0.763811	2.5894e-03	9.1358e+11	2.7880e+07	yes
25	53*109	0.763169	4.9684e-03	9.9277e+11	3.0297e+07	yes
26	79*103	0.761923	9.3256e-03	1.1082e+12	3.3820e+07	yes
27	71*163	0.762637	1.8255e-02	1.1764e+12	3.5900e+07	yes
28	83*197	0.762689	4.9026e-02	9.1439e+11	2.7905e+07	yes
29	101*229	0.762295	8.3367e-02	1.1141e+12	3.4000e+07	yes
30	137*239	0.760210	1.5135e-01	1.2770e+12	3.8971e+07	yes
31	149*311	0.760909	3.1232e-01	1.2789e+12	3.9029e+07	yes
32	233*281	0.770465	6.9549e-01	1.3524e+12	4.1273e+07	yes
33	211*439	0.770119	1.5432e+00	1.2580e+12	3.8392e+07	yes
34	283*463	0.769982	2.9051e+00	1.3838e+12	4.2230e+07	yes
35	241*769	0.785465	5.8528e+00	1.4148e+12	4.3177e+07	yes
36	503*521	0.770013	1.3677e+01	1.2511e+12	3.8180e+07	yes
37	389*953	0.769878	2.9071e+01	1.2103e+12	3.6935e+07	yes
38	557*941	0.769711	5.1814e+01	1.4005e+12	4.2741e+07	yes
39	859*863	0.769253	1.0714e+02	1.3905e+12	4.2436e+07	yes
40	911*1151	0.769082	2.3201e+02	1.3222e+12	4.0350e+07	yes
41	1039*1427	0.768945	4.7427e+02	1.3261e+12	4.0469e+07	yes
42	1399*1499	0.768789	1.0665e+03	1.2124e+12	3.7000e+07	yes

Ending due to allocation error

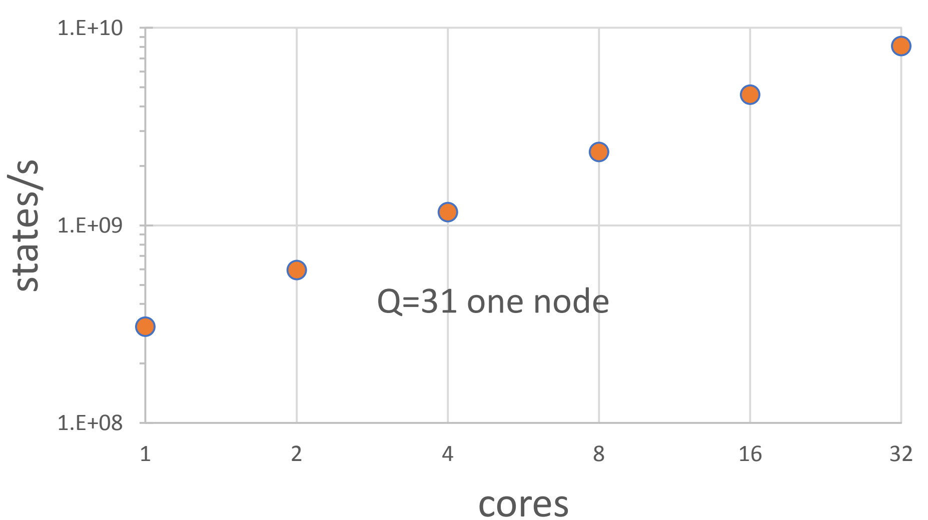
Normalized performance

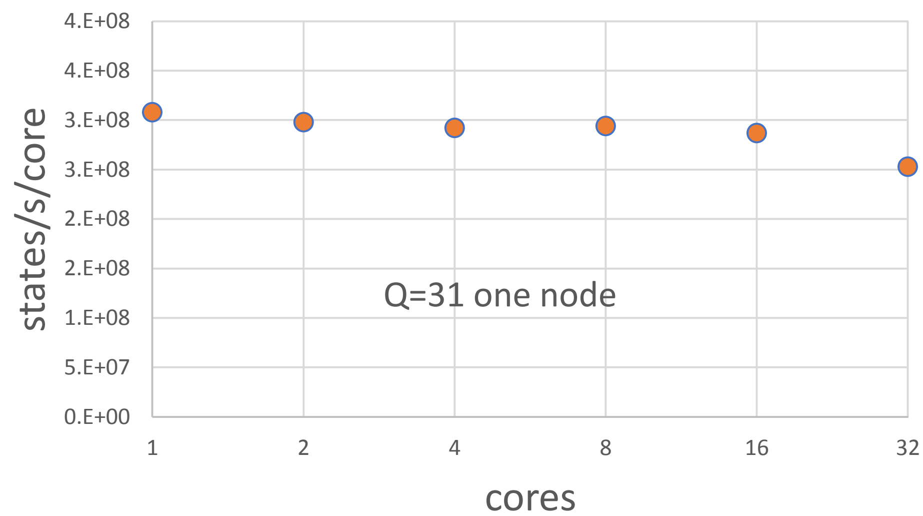
- Number of basis states each core can process in a second

$$States/(s\ core) = \frac{gates \times 2^Q}{Rawtime \times cores}$$

- In an ideal computer that should be a constant
- In reality it will vary due to the network and memory access
- Useful to compare cores
- Raw time useful to compare whole systems

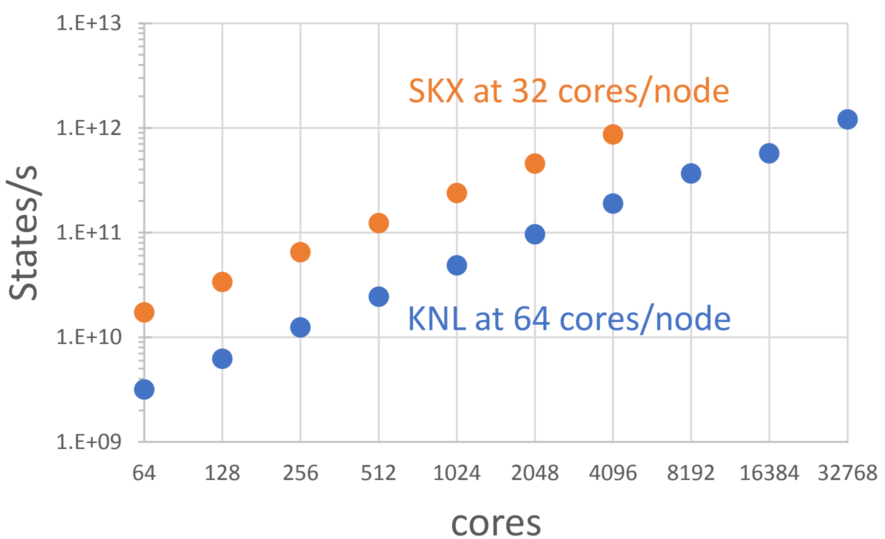
One node SKX Xeon 8160 and different core numbers, the metric is nearly constant.

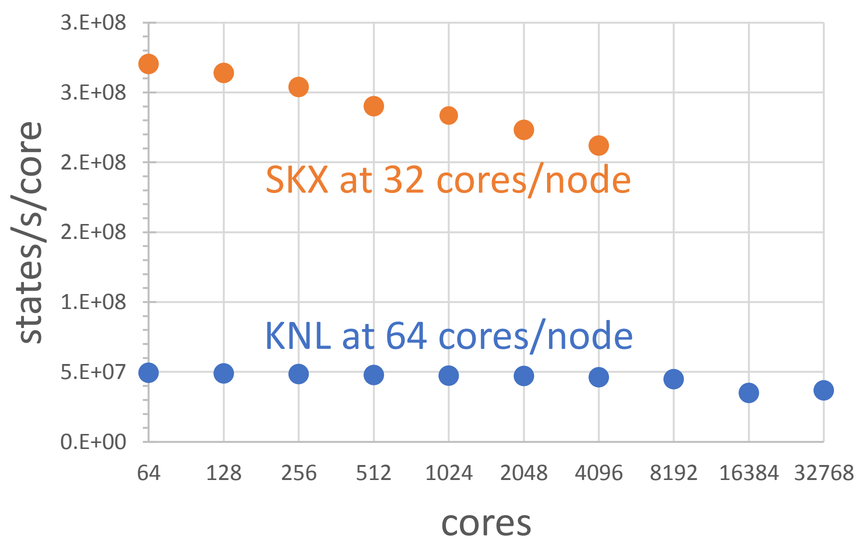




A typical comparison in TACC's Stampede 2

- Compare KNL Xeon Phi 7250 vs. SKX Xeon 8160 nodes
- From 1 to 512 nodes
- Left graph shows overall performance
- Right graph shows memory/switch slowdown in per core basis





Acknowledgments

- Scott Pakin (LANL)
- Texas Advanced Computing Center (TACC)
- UNT High Performance Computing
- Coke Reed
- John Lockman
- Thanks Data Vortex Technologies for providing the hardware and research support for creating this benchmark