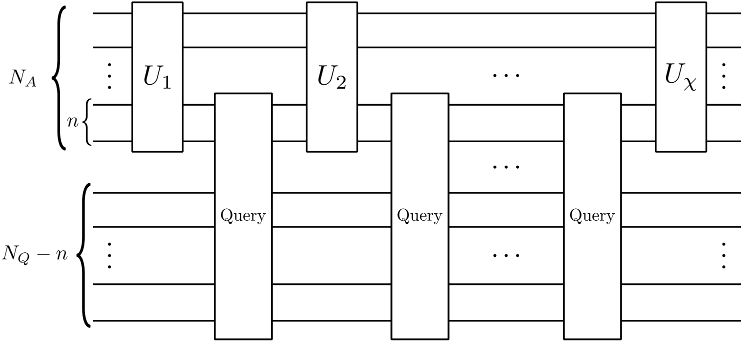
Fault-tolerant resource estimation of quantum random-access memories

Olivia Di Matteo, Vlad Gheorghiu, and Michele Mosca

22 June 2019 – QRE 2019 – Phoenix, AZ

Quantum random-access memory (qRAM)

Many quantum algorithms require routines to store and query classical/quantum data *in superposition*.



Quantum random-access memory (qRAM)

We will think of the qRAM as an *oracle*.

The query can take many different forms:

Quantum random-access memory (qRAM)

We will think of the qRAM as an *oracle*.

The query can take many different forms:

- Read in a bit b_x as a phase (e.g. Grover's algorithm)

$$|x\rangle \rightarrow (-1)^{b_x}|x\rangle$$

Quantum random-access memory (qRAM)

We will think of the qRAM as an *oracle*.

The query can take many different forms:

- Read in a bit b_x as a phase (e.g. Grover's algorithm)

$$|x\rangle \rightarrow (-1)^{b_x}|x\rangle$$

- Read in a bit b_x as a qubit (e.g. element distinctness)

$$|x\rangle|0\rangle \rightarrow |x\rangle|b_x\rangle$$

Quantum random-access memory (qRAM)

We will think of the qRAM as an *oracle*.

The query can take many different forms:

- Read in a bit b_x as a phase (e.g. Grover's algorithm)

$$|x\rangle \to (-1)^{b_x}|x\rangle$$

- Read in a bit b_x as a qubit (e.g. element distinctness)

$$|x\rangle|0\rangle \to |x\rangle|b_x\rangle$$

- Read in a complex vector as amplitudes (e.g. HHL algorithm)

$$\mathbf{b} = (b_1 \ldots b_n) \to \sum_j b_j|j\rangle$$

The trouble with qRAM

Problem

"We made this really cool algorithm. It runs really fast!
...Assuming that I have this very specifically crafted input state and can query a qRAM efficiently."

— many recent quantum algorithm papers

The trouble with qRAM

Problem

"We made this really cool algorithm. It runs really fast!
...Assuming that I have this very specifically crafted input state and can query a qRAM efficiently."

— many recent quantum algorithm papers

Why is this a problem?

To run algorithms 'at scale' with a **large number of queries** we need to run them fault-tolerantly. But to run something fault-tolerantly typically incurs a massive overhead.

Can we really assume that querying a qRAM can be done efficiently in a fault-tolerant setting?

No.

But this isn't the end of the story.

Our goal is to analyze the tradeoffs and resources required for fault-tolerant qRAM.

There are two contexts to consider:

1. Small number of queries
2. **Large number of queries (our work)**
   - qRAM circuit families
   - Runtime complexities
   - Surface code costs and resource estimates

Our qRAM model

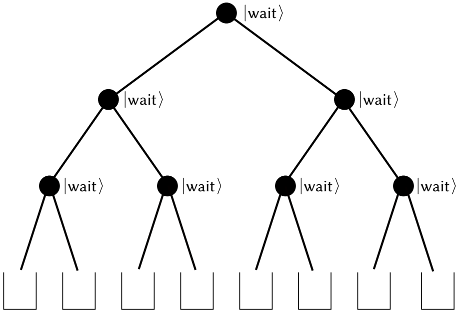
We want to query **classical data** in superposition:

$$\sum_j \alpha_j |j\rangle|0\rangle \longrightarrow \sum_j \alpha_j |j\rangle|b_j\rangle$$

where $|b_j\rangle$ is either $|0\rangle$ or $|1\rangle$ (not a superposition).

1. Small number of queries

Bucket-brigade qRAM (Giovannetti, Lloyd, Maccone 2008).



Polynomial number of operations $\rightarrow$ can get away with inverse-polynomial gate error rates.

Our result: Our main result shows that there is essentially no quantum advantage when searching with a faulty oracle.

Theorem 1. *Any algorithm that solves the p-faulty Grover problem must use* $T > \frac{p}{10(1-p)} N$ *queries.*

– Regev and Schiff (2008)

In algorithms with a *superpolynomial* number of queries, it appears that we must use fault-tolerant error correction to suppress the error rates to be superpolynomially small (Arunachalam et. al, 2015).

Resource estimation of fault-tolerant qRAM

What is the cost of fault-tolerantly implementing a large qRAM performing a large number of queries?

Resource estimation of fault-tolerant qRAM

What is the cost of fault-tolerantly implementing a large qRAM performing a large number of queries?

Logical-layer cost model

$$\text{Cost} = \text{Logical qubits} \times T\text{-depth}$$

What is the cost of fault-tolerantly implementing a large qRAM performing a large number of queries?

Logical-layer cost model

$$\text{Cost} = \text{Logical qubits} \times T\text{-depth}$$

Physical-layer cost model

$$\text{Cost} = \text{Physical qubits} \times \text{Surface code cycles}$$

How to build a qRAM

There are two ways to store data in a qRAM:

1. **Explicitly:** data is stored in actual qubit states in hardware and are queried by coupling to them with CNOTs

   Advantages: Circuit needs to be compiled only once; independent of the contents of the memory.

   Disadvantages: Space overhead - need as many qubits as you have memory slots used only for storage. Need to initialize them and ensure they maintain their state.

2. **Implicitly:** data is encoded into a circuit. Can be considered as a qROM or lookup table.

   Advantages: Can in principle design more “compact” circuits. Can optimize based on structure and content of the memory.

   Disadvantages: Can only do this if you know the contents of the memory in advance. Requires recompilation of the query circuit when memory contents are updated.

How to build a qRAM

We focus mostly on designing implicit circuits, but also analyze an explicit circuit (bucket brigade) for comparison.

We focus mostly on designing implicit circuits, but also analyze an explicit circuit (bucket brigade) for comparison.

Key parameters are:

- n, the number of address bits (memory can hold 2^n bits)
- q, denotes that 2^q memory locations contain a 1

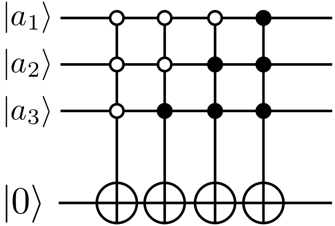
A simple qRAM circuit

Let $n = 3, q = 2$. Suppose we know the locations of the 1s: $|000\rangle, |001\rangle, |011\rangle, |111\rangle$.

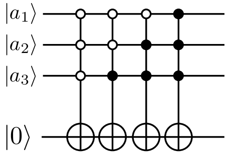
A simple qRAM circuit

Let $n = 3, q = 2$. Suppose we know the locations of the 1s: $|000\rangle, |001\rangle, |011\rangle, |111\rangle$.

Design a circuit such that 'valid' addresses flip output bit to $|1\rangle$.



A simple qRAM circuit



The good: This circuit uses very few qubits.
The bad: It will be slow.

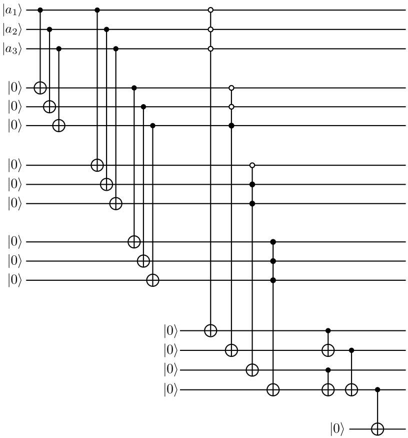
Calculate[1] logical qubits and T-depth:

$$\begin{aligned} N_Q &= 2n \\ T_d &= 4 \cdot 2^q (n-2) \end{aligned}$$

Then **total cost** is $O(n^2 2^q)$.

[1] We add some ancillae to implement the mixed-polarity gates.

Parallelize everything as much as we can.



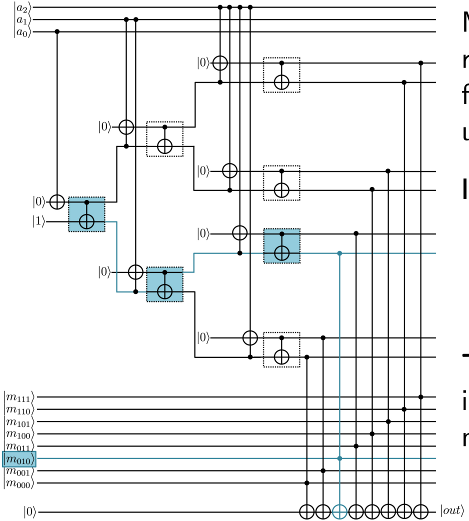
The good: It will be fast.
The bad: Huge amount of qubits.

Resources:

$$N_Q = 2n \cdot 2^q + 1$$
$$T_d = 4(n-2)$$

Total cost is still $O(n^2 2^q)$, but we have traded space for time.

Explicit version: bucket-brigade qRAM



Memory contents stored in a register of qubits. Log-depth fanout of address bits, readout using Toffolis.

If we fully parallelize Toffolis:

$$N_Q = 8 \cdot 2^n$$
$$T_d = 2n - 1$$

Total cost is $O(n2^n)$; independent of contents of memory/number of 1s.

Image credit: S. Arunachalam, V. Gheorghiu, T. Jochym-OConnor, M. Mosca, P. Srinivasan, New Journal of Physics, **17** (12) 123010 (2015)

Complexity cost of a qRAM

Circuit	Large depth	Large width	Bucket brigade parallel
N_Q	$2n$	$n2^{q+1}+1$	$8 \cdot 2^n$
T_d	$2^{q+2}(n-2)$	$4(n-2)$	$2n-1$
Cost	$O(n^2 \cdot 2^q)$	$O(n^2 \cdot 2^q)$	$O(n \cdot 2^n)$

Large depth/width circuits become advantageous for sparser memories, when $q \approx n - \log n$.

What about *actual* resource costs?

The numbers...

We perform circuit synthesis over Clifford + T.

We perform estimates using **defect-based surface codes**[2].

All our circuits, data, and estimation routines are available at:
`https://github.com/glassnotes/FT_qRAM_Circuits`

[2] Estimates using lattice surgery are in progress.

Important note:

1. We have estimated the physical resources making very optimistic assumptions about the surface code:
   - Cycle time of 200ns
   - Gate error rate of 10^{-5}
   - Input state injection failure probability 10^{-4}

The numbers...

Important note:

1. We have estimated the physical resources making very optimistic assumptions about the surface code:
   - Cycle time of 200ns
   - Gate error rate of 10^{-5}
   - Input state injection failure probability 10^{-4}
2. State distillation is the limiting step in the runtime, but we always have as many distilleries as is needed for a layer of T-depth

The numbers...

Important note:

1. We have estimated the physical resources making very optimistic assumptions about the surface code:
   - Cycle time of 200ns
   - Gate error rate of 10^{-5}
   - Input state injection failure probability 10^{-4}
2. State distillation is the limiting step in the runtime, but we always have as many distilleries as is needed for a layer of T-depth
3. We assume a random memory, with the 'worst' possible situation in each case, and therefore don't perform any extensive circuit optimization.

The numbers...

Important note:

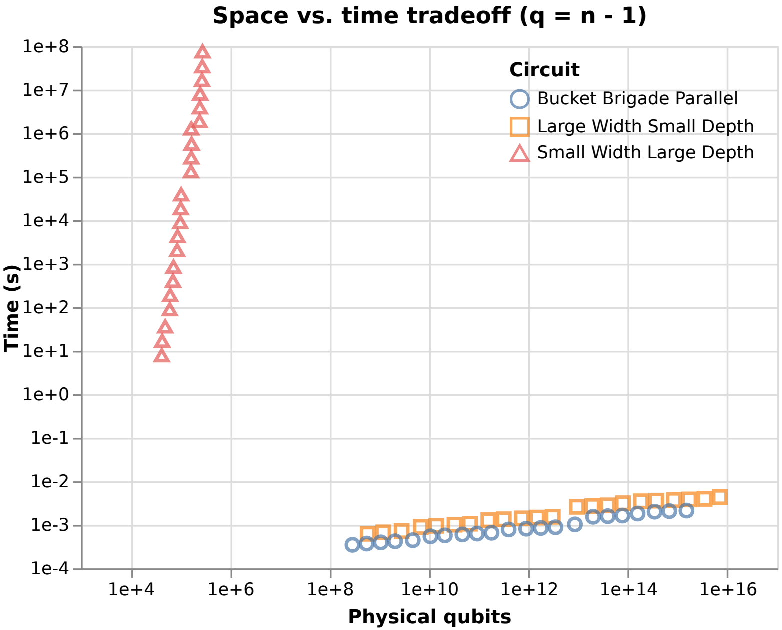
1. We have estimated the physical resources making very optimistic assumptions about the surface code:
   - Cycle time of 200ns
   - Gate error rate of 10^{-5}
   - Input state injection failure probability 10^{-4}
2. State distillation is the limiting step in the runtime, but we always have as many distilleries as is needed for a layer of T-depth
3. We assume a random memory, with the 'worst' possible situation in each case, and therefore don't perform any extensive circuit optimization.

Important note:

1. We have estimated the physical resources making very optimistic assumptions about the surface code:
   - Cycle time of 200ns
   - Gate error rate of 10^{-5}
   - Input state injection failure probability 10^{-4}
2. State distillation is the limiting step in the runtime, but we always have as many distilleries as is needed for a layer of T-depth
3. We assume a random memory, with the 'worst' possible situation in each case, and therefore don't perform any extensive circuit optimization.

So, *don't take the numbers too literally*. They are meant to be representative of the sheer scale of the problem, and to highlight the different tradeoffs between our circuits.

Space vs. time for $n = 15$ to $n = 36$ ($q = n - 1$)



Space vs. time

Circuit	n	q	Total time (s)	Physical qubits
Bucket brigade parallel	15	-	$3.48 \cdot 10^{-4}$	$2.89 \cdot 10^{8}$
Large width small depth	15	14	$6.24 \cdot 10^{-4}$	$5.84 \cdot 10^{8}$
Small width large depth	15	14	7.86	$4.23 \cdot 10^{4}$
Bucket brigade parallel	36	-	$2.13 \cdot 10^{-3}$	$1.50 \cdot 10^{15}$
Large width small depth	36	35	$4.35 \cdot 10^{-3}$	$7.06 \cdot 10^{15}$
Small width large depth	36	35	$7.55 \cdot 10^{7}$	$2.80 \cdot 10^{5}$

Space vs. time

Circuit	n	q	Total time (s)	Physical qubits
Bucket brigade parallel	15	-	$3.48 \cdot 10^{-4}$	$2.89 \cdot 10^{8}$
Large width small depth	15	14	$6.24 \cdot 10^{-4}$	$5.84 \cdot 10^{8}$
Small width large depth	15	14	7.86	$4.23 \cdot 10^{4}$
Bucket brigade parallel	36	-	$2.13 \cdot 10^{-3}$	$1.50 \cdot 10^{15}$
Large width small depth	36	35	$4.35 \cdot 10^{-3}$	$7.06 \cdot 10^{15}$
Small width large depth	36	35	$7.55 \cdot 10^{7}$	$2.80 \cdot 10^{5}$

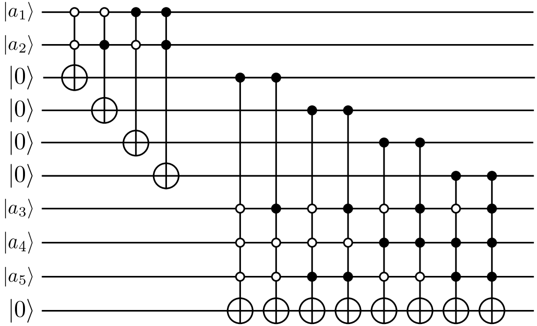
Why these n?

- $n = 15 \Rightarrow$ 4KB - Apple I shipped with this much RAM in 1976
- $n = 36 \Rightarrow$ 8GB - what machines ship with today

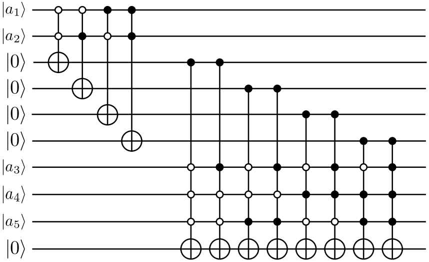
Hybrid circuits

Can we make our space-time tradeoffs more flexible?

Idea: Control on the first k address bits, then use the outputs to control on the remaining $n - k$.



Hybrid circuits

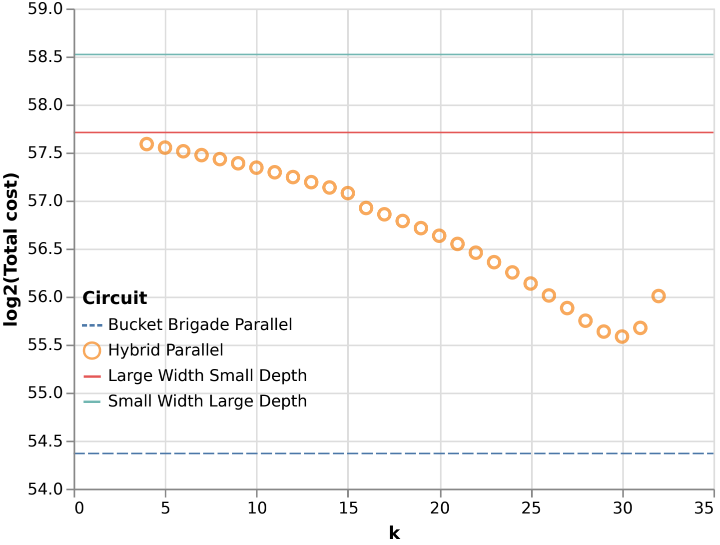


We can then:

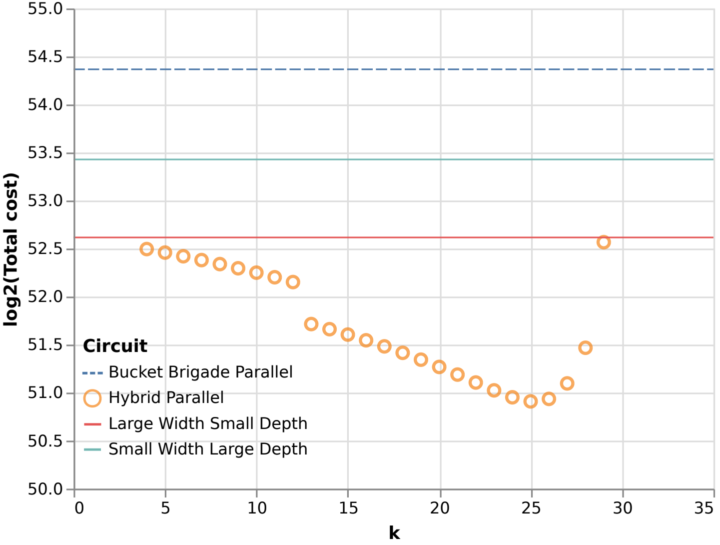
- Run this as-is
- Parallelize only the top 'tier', or only the bottom 'tier'
- Parallelize everything

In the *worst case*, there will be 2^k unique outputs from the 'top tier'. Is this really better? How does it depend on k?

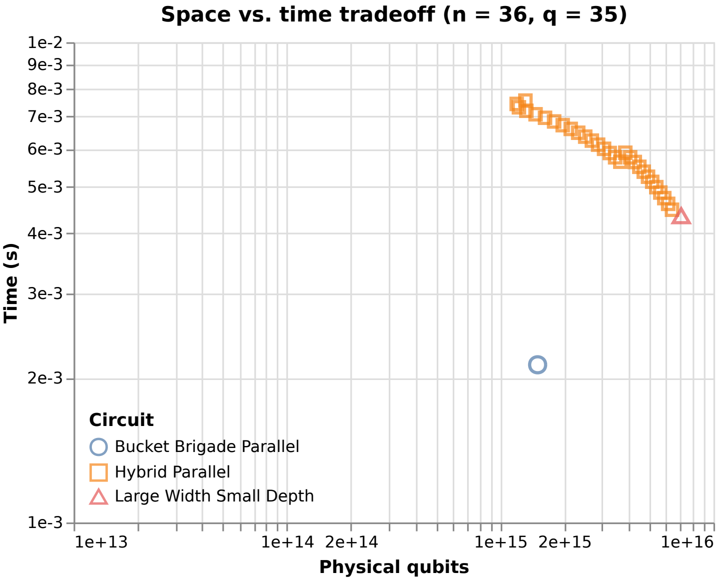
Surface code cost estimates for $n = 36, q = 35$



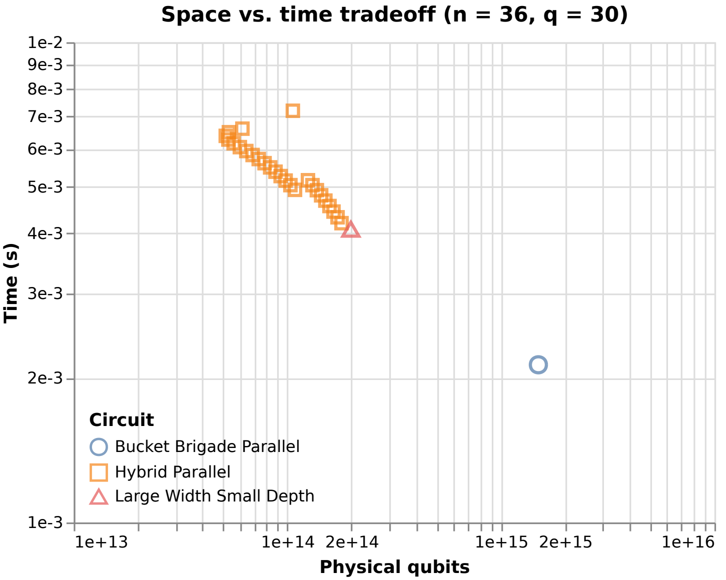
Surface code cost estimates for $n = 36, q = 30$



Space vs. time tradeoff for $n = 36, q = 35$

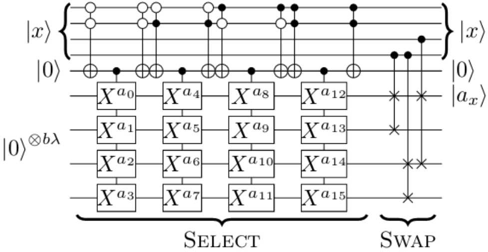


Space vs. time tradeoff for $n = 36, q = 30$



Low, Kliuchnikov, Schaeffer. *Trading T-gates for dirty qubits in state preparation and unitary synthesis* (arXiv:1812.00954)

A different way of making a depth/width tradeoff, with additional flexibility in terms of the input qubits.



Duplicate lower register λ times - larger λ means more qubits and more SWAPs, but smaller MPMCTs in the upper register.

SelectSwap circuits

Logical-level runtime complexity; $\lambda = O(\sqrt{N})$ is optimal.

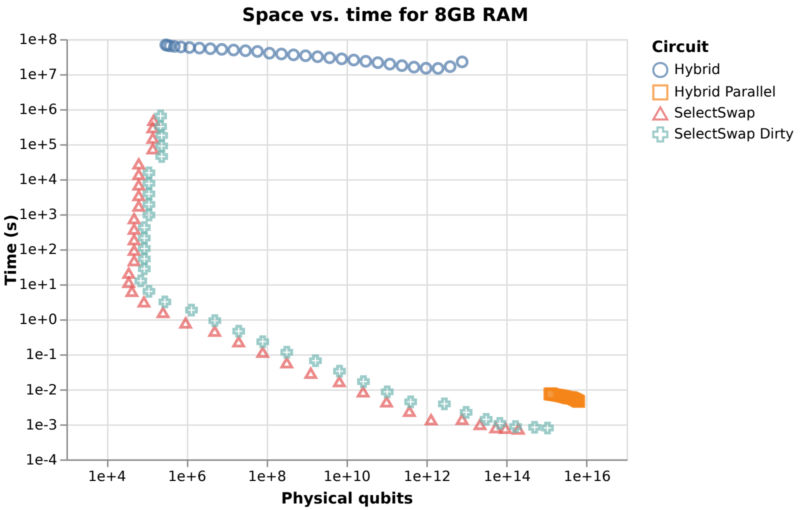
Ancillae	Qubits	T-depth	T-count
Clean	$b\lambda + 2\lceil \log_2 N \rceil$	$\frac{N}{\lambda} + \log \lambda$	$4\left\lceil \frac{N}{\lambda} \right\rceil + 8b\lambda$
Dirty	$(b+1)\lambda + 2\lceil \log_2 N \rceil$	$\frac{N}{\lambda} + \log \lambda$	$8\left\lceil \frac{N}{\lambda} \right\rceil + 32b\lambda$

Let's compare with our hybrid circuits: set $b = 1, N = 2^n$.

Ancillae	Qubits	T-depth	T-count
Clean	$\lambda + 2n$	$\frac{2^n}{\lambda} + \log \lambda$	$4\left\lceil \frac{2^n}{\lambda} \right\rceil + 8\lambda$
Dirty	$2\lambda + 2n$	$\frac{2^n}{\lambda} + \log \lambda$	$8\left\lceil \frac{2^n}{\lambda} \right\rceil + 32\lambda$

Space vs. time for SelectSwap circuits for $n = 36$

For simplicity, choose $\lambda = 2^m, \quad m = 1, 2, ..., n - 1$.



Outlook

Bad news: Unlikely we'll be querying large-scale qRAMs efficiently any time soon.

Outlook

Bad news: Unlikely we'll be querying large-scale qRAMs efficiently any time soon.

Good news: Tons of room for improvement!

Outlook

Bad news: Unlikely we'll be querying large-scale qRAMs efficiently any time soon.

Good news: Tons of room for improvement!

... and lots left to do:

- estimates for other models of qRAM
- exploring a wider range of hybrid circuits
- costing 'real-world' examples that have been heavily optimized
- better surface code techniques (lattice surgery)
- see where we can take advantage of special address structures

Thank you for your attention!

Thanks to Vlad and Michele, Matt Amy, Dominic Berry, Austin Fowler, Craig Gidney, and Matthias Soeken for useful discussions, and my thesis committee (Richard Cleve, Seth Lloyd, Roger Melko, Kevin Resch) for a critical reading of an early version of this work.

TRIUMF UNIVERSITY OF WATERLOO | IQC Institute for Quantum Computing NSERC CRSNG