



Quantum Computing and Its Applications (QCA'22)

Quantum Algorithms

Lecture in Summer School organized by University
of Delhi and Jaypee Institute of Information
Technology, Noida

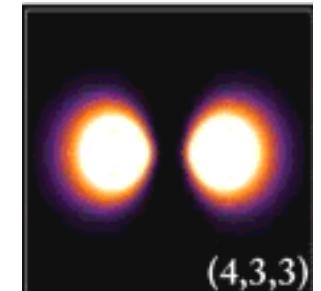
Quantum Computing for Databases - Optimizing Transaction Schedules

Professor Dr. rer. nat. habil. Sven Groppe

<https://www.ifis.uni-luebeck.de/index.php?id=groppe>

Quantum Mechanics

- Very small particles and light behave differently from objects in normal life
 - Mechanics of light and matter at the atomic and subatomic scale are described by quantum theory
 - forming the underlying principles of chemistry and most of physics
 - Quantum theory has brought us the information age with its disruptive technologies of
 - transistors,
 - lasers,
 - nuclear power, and
 - superconductivity...
- ...and now also quantum computers!**



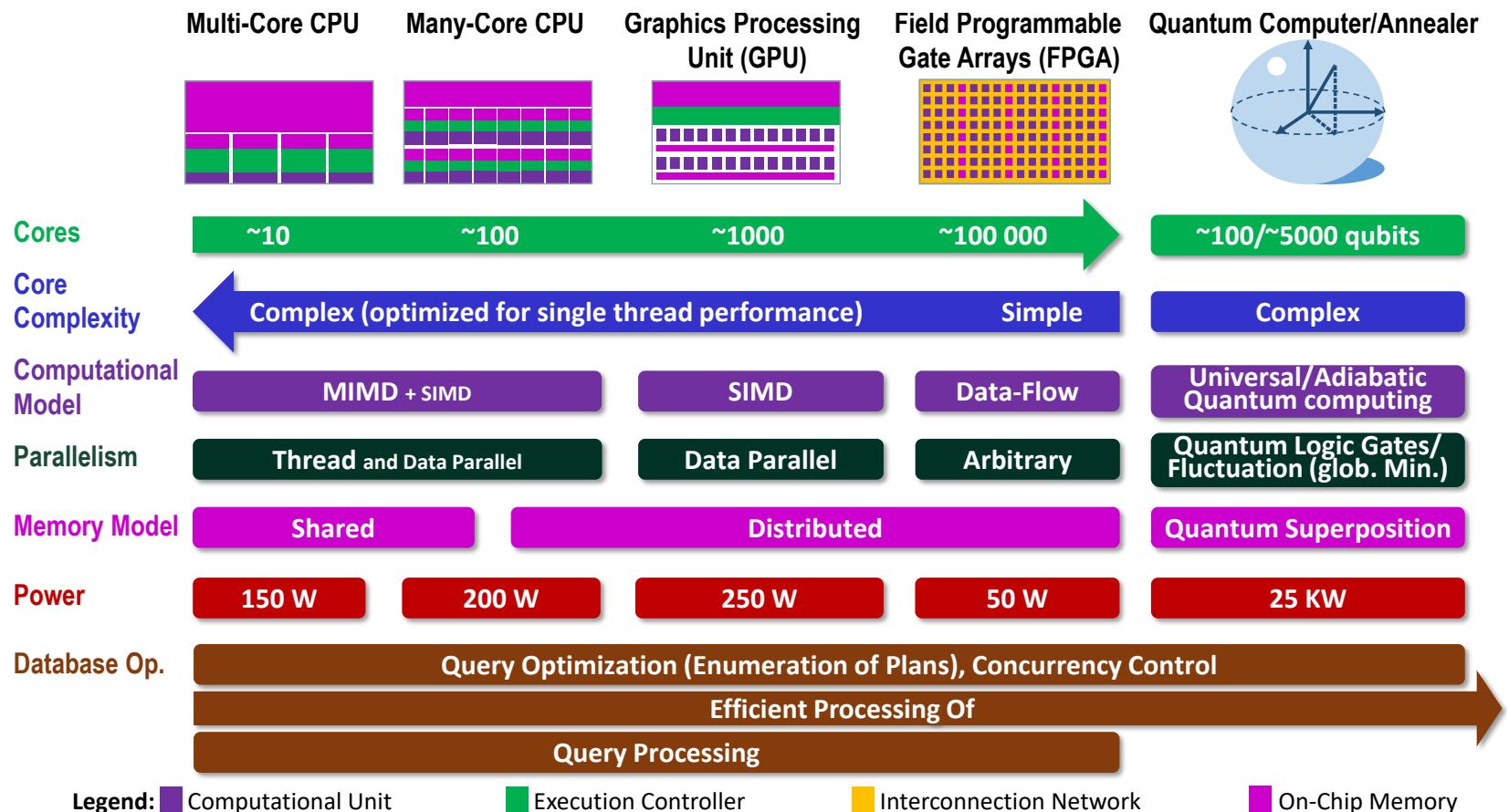
Wavefunctions of the electron in a hydrogen atom at different energy levels. Brighter areas represent a higher probability of finding the electron.

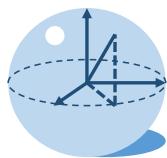


Timeline of important events

- **1982:** Feynman first proposes the concept of quantum computers [F'82]
- **2019:** Google announces "Quantum Supremacy" by its 53-bit chip "Sycamore" [A+'19]
 - 200 seconds on Sycamore versus 10,000 years on the world's fastest supercomputer IBM Summit
 - IBM [P+'19]: only 2.5 days on classical supercomputer after deduction of the problem (i.e., using a better classical algorithm) with more disk space
 - Pan, Chen and Zhang [PCZ'21]: only 15 hours on 512 GPU-cluster using another classical algorithm for obtaining a large number of uncorrelated samples
 - Estimation: a few dozens of seconds on a modern supercomputer with ExaFLOPS performance
 - Discussion intensified the excessive hype about the current state of quantum technology

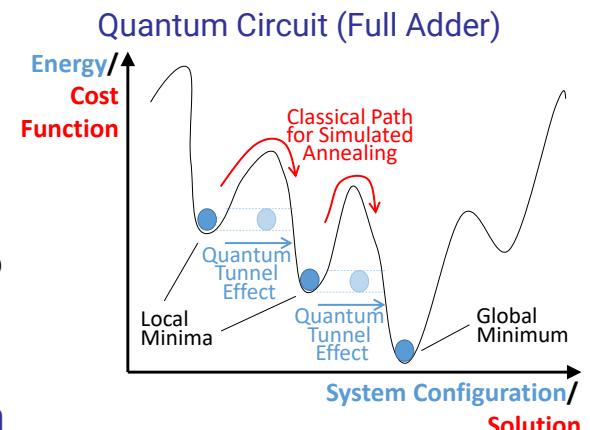
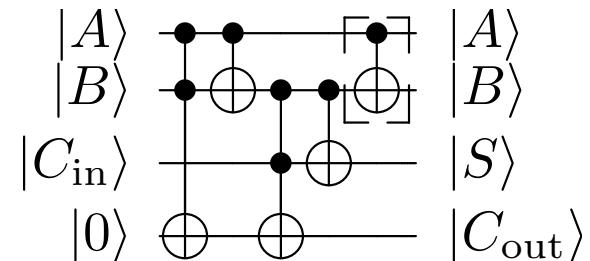
Architectures of Emergent Hardware





Quantum Computer

- use of quantum-mechanical phenomena such as superposition and entanglement to perform computation
- Different types of quantum computer, e.g.
 - Universal Quantum Computer
 - uses quantum logic gates arranged in a circuit to do computation
 - measurement (sometimes called observation) assigns the observed variable to a single value
 - Quantum Annealing
 - metaheuristic for finding the global minimum of a given objective function over a given set of candidate solutions
 - i.e., some way to solve a special type of mathematical optimization problem



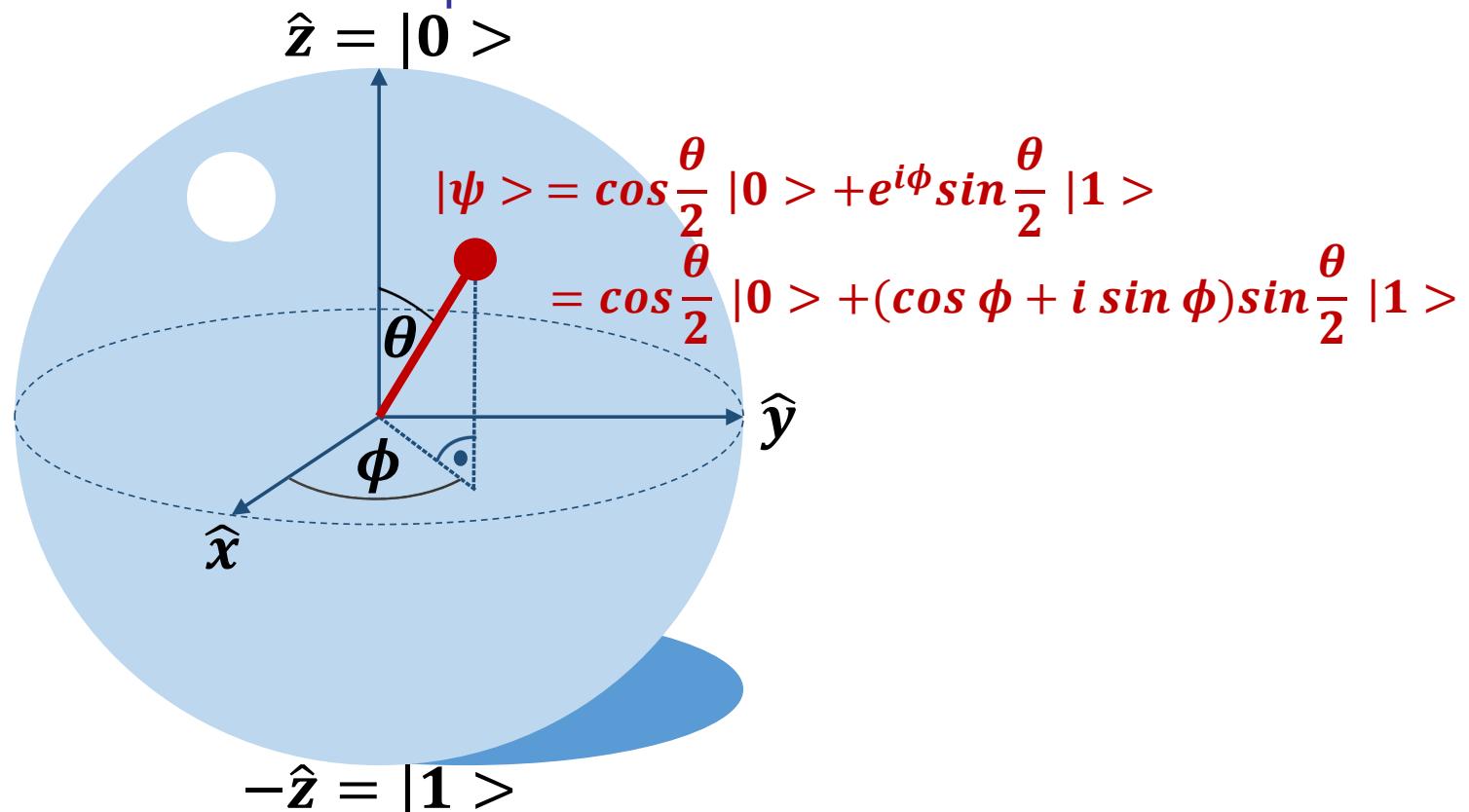
Simulated versus Quantum Annealing

Classical versus Quantum Computing

	Classical	Quantum												
Information Unit	Binary Digit (Bit): <ul style="list-style-type: none"> basis of a 2-level system can be in state 0 or 1 	Quantum Bit (Qubit): <ul style="list-style-type: none"> basis of a 2-level quantum system can be in state $0\rangle$, $1\rangle$ or in a linear combination of both states 												
Operation	Logic Gate: <ul style="list-style-type: none"> performs on 1 or more bits to produce a single bit output 	Quantum Logic Gate: <ul style="list-style-type: none"> performs on 1 or more qubits to change the quantum state of a single qubit 												
Example Operation	NOT/Inverter: Digital Circuit: $In \rightarrowtail Out$ <table border="1"> <thead> <tr> <th>In</th> <th>Out</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> </tr> </tbody> </table>	In	Out	0	1	1	0	NOT/Pauli_x-Gate: Quantum Circuit: $ In\rangle \xrightarrow{\oplus} Out\rangle$ Alternatively: $ In\rangle \xrightarrow{[X]} Out\rangle$ <table border="1"> <thead> <tr> <th>In</th> <th>Out</th> </tr> </thead> <tbody> <tr> <td>$0\rangle$</td> <td>$1\rangle$</td> </tr> <tr> <td>$1\rangle$</td> <td>$0\rangle$</td> </tr> </tbody> </table> $\frac{1}{\sqrt{2}} (0\rangle + 1\rangle) \quad \frac{1}{\sqrt{2}} (0\rangle + 1\rangle)$ $\frac{3 \cdot i}{5} 0\rangle + \frac{4}{5} 1\rangle \quad \frac{4}{5} 0\rangle + \frac{3 \cdot i}{5} 1\rangle$	In	Out	$ 0\rangle$	$ 1\rangle$	$ 1\rangle$	$ 0\rangle$
In	Out													
0	1													
1	0													
In	Out													
$ 0\rangle$	$ 1\rangle$													
$ 1\rangle$	$ 0\rangle$													

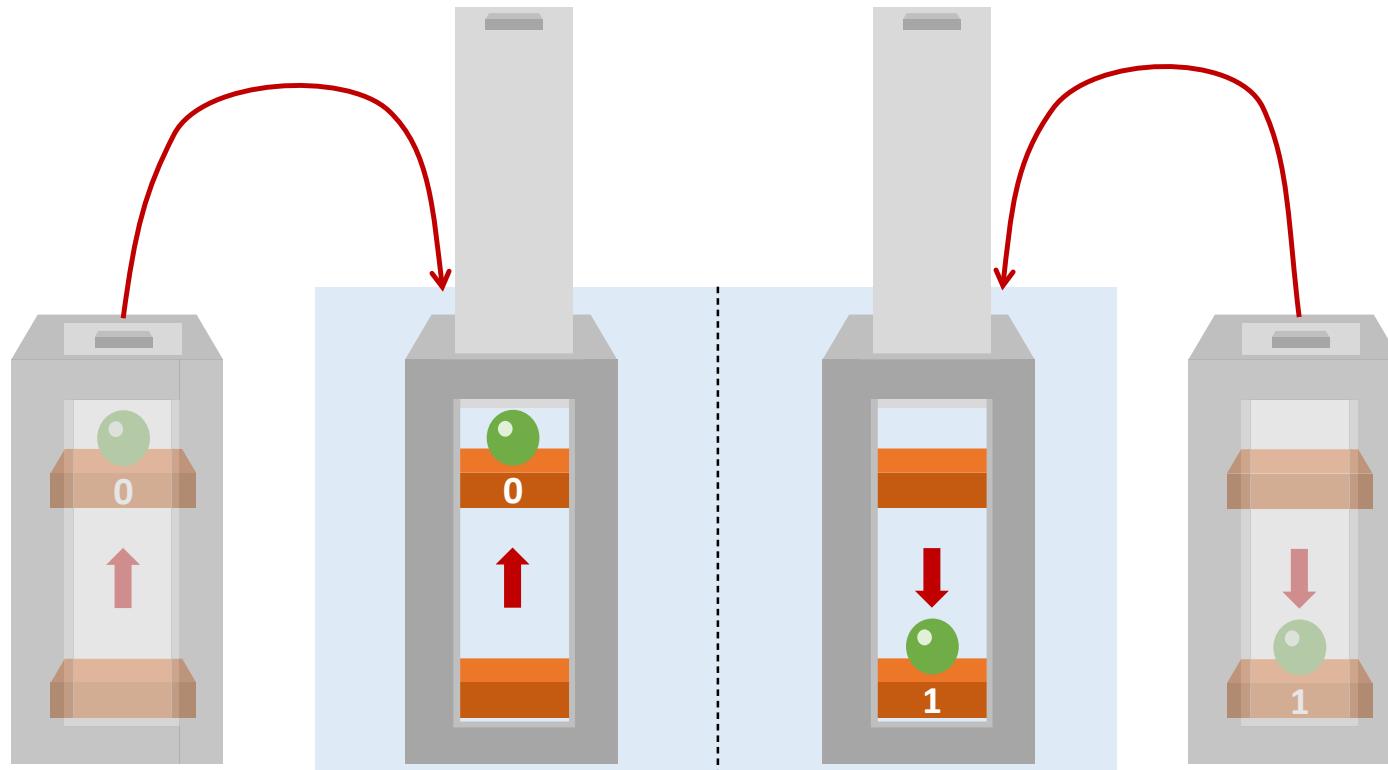
Representation of a Qubit in Bloch-Sphere

- Angels θ and ϕ can be associated with spherical coordinates on the so-called Bloch-sphere:



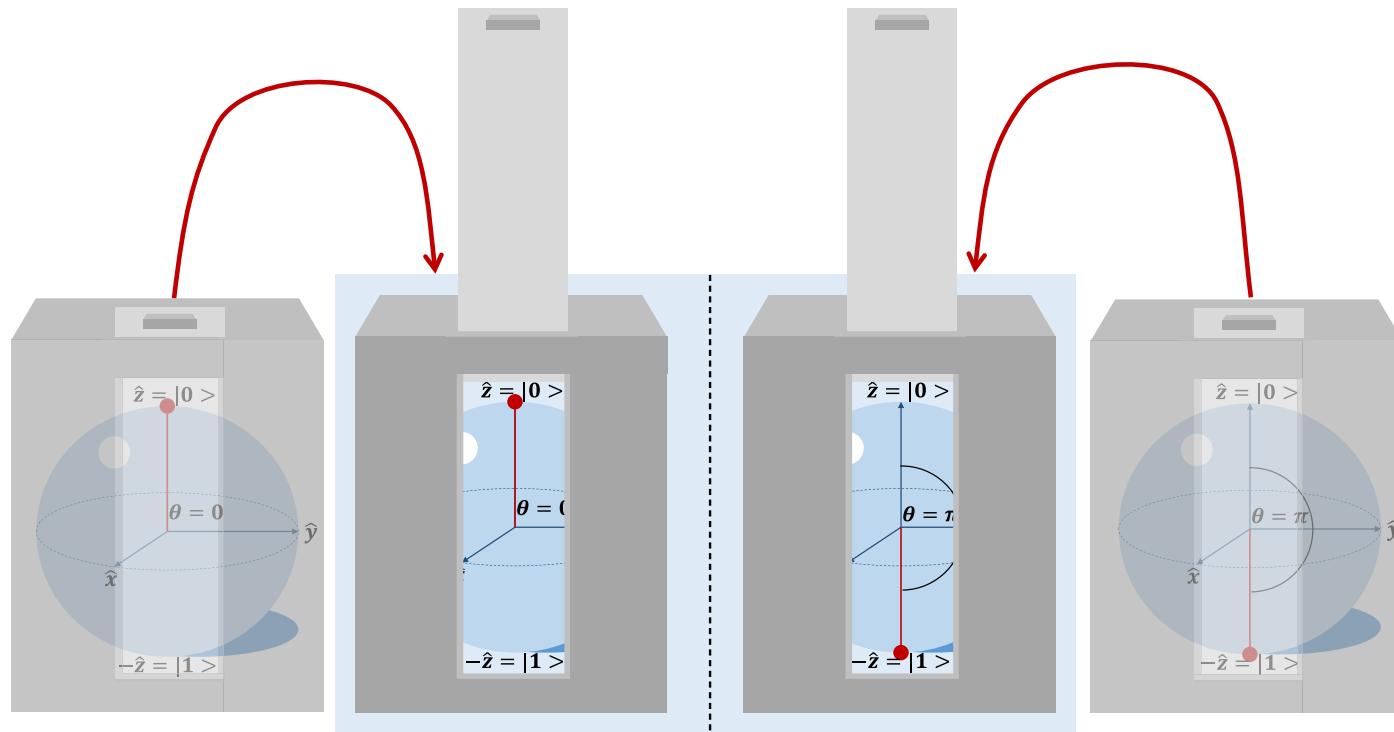
Classical Measurement/Observation

- The state is not destroyed by a measurement/observation in classical systems:



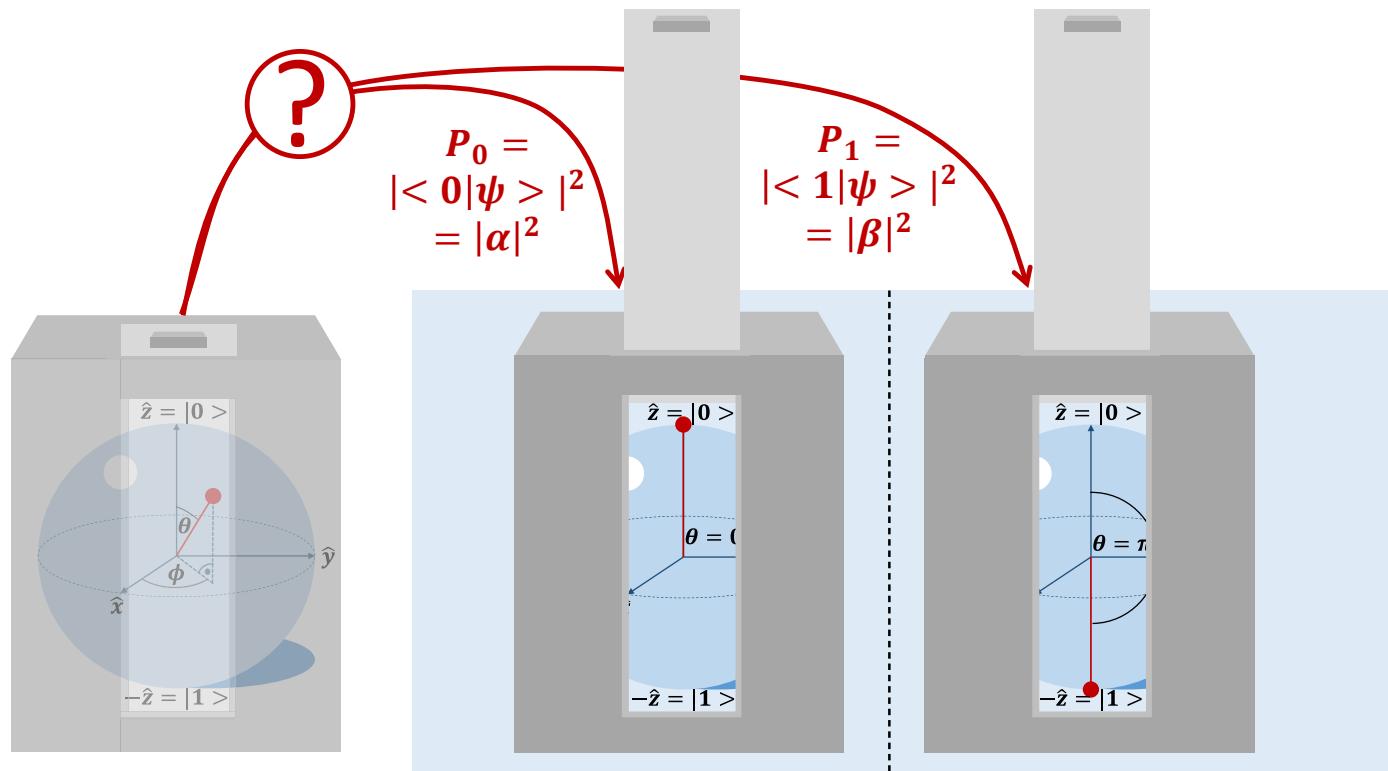
Quantum Measurement/Observation 1/2

- The state is not destroyed by a measurement/observation in quantum mechanical systems for state $|0\rangle$ and $|1\rangle$:



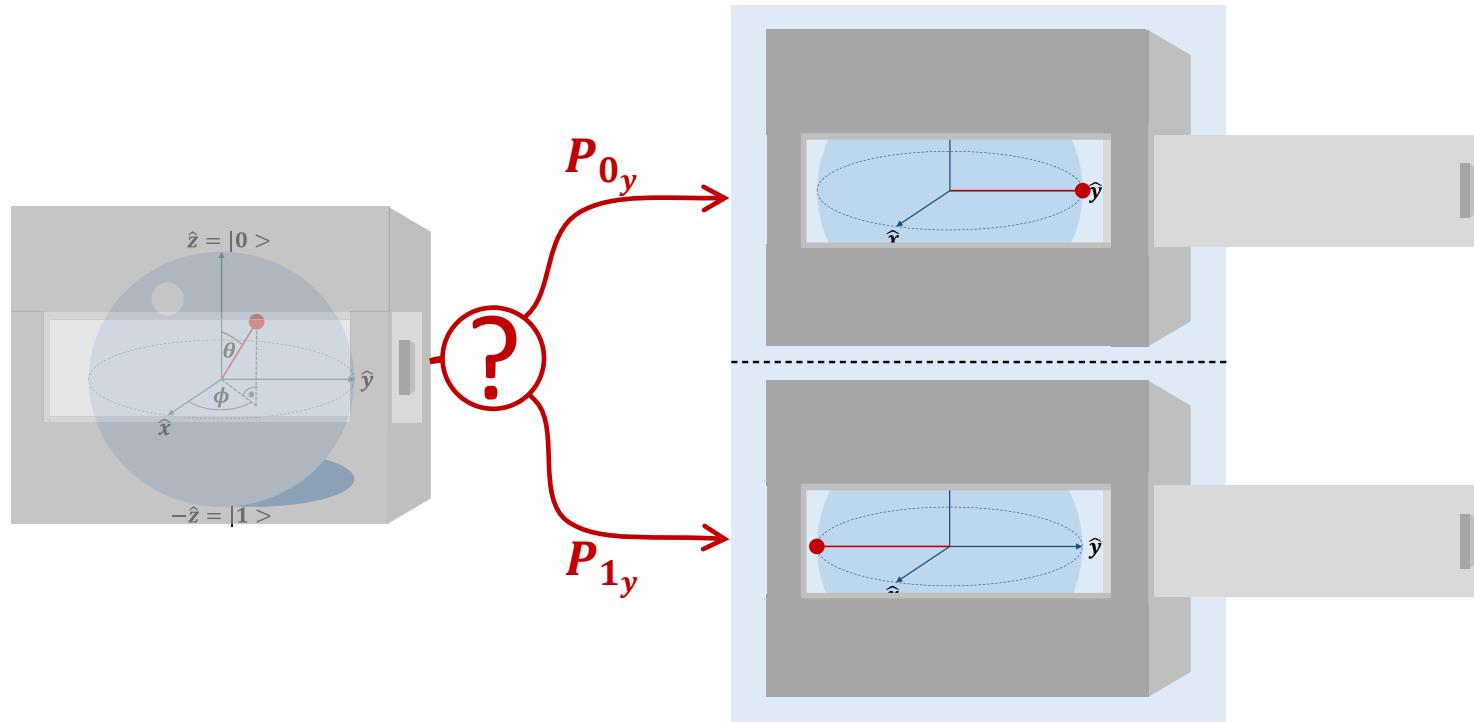
Quantum Measurement/Observation 2/2

- During observation a superposition state collapses to $|0\rangle$ or $|1\rangle$ according to corresponding probabilities:



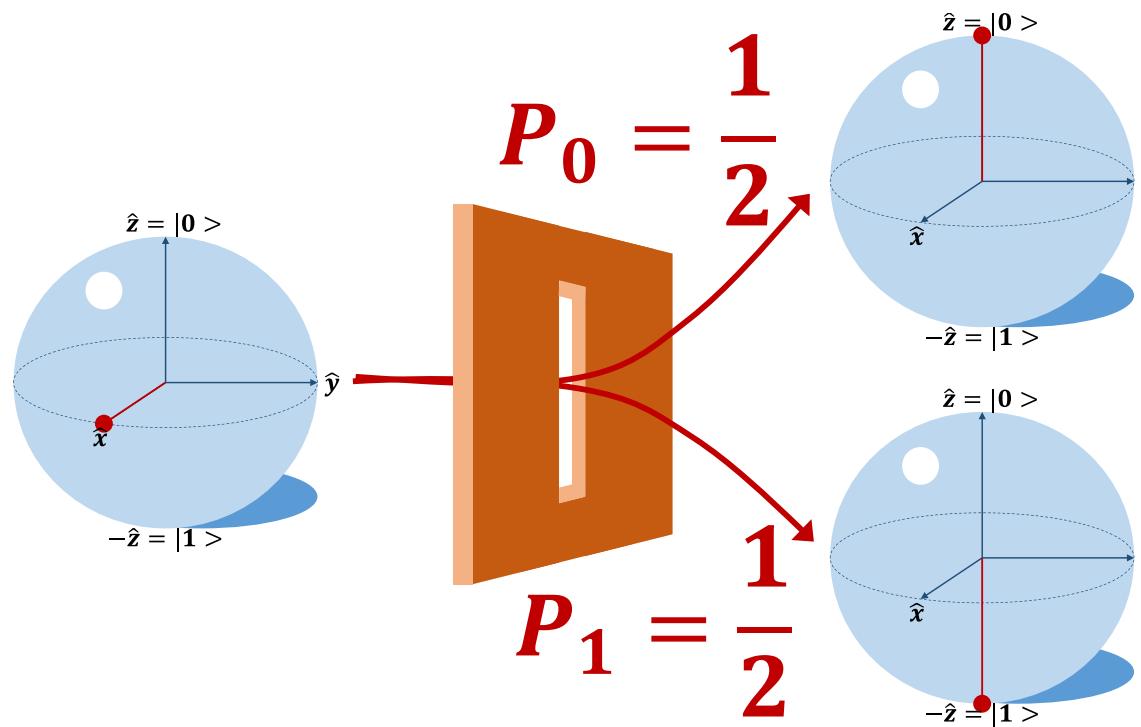
Measurement/Observation along other axis (here y-axis)

- However, observation typically according to z-axis



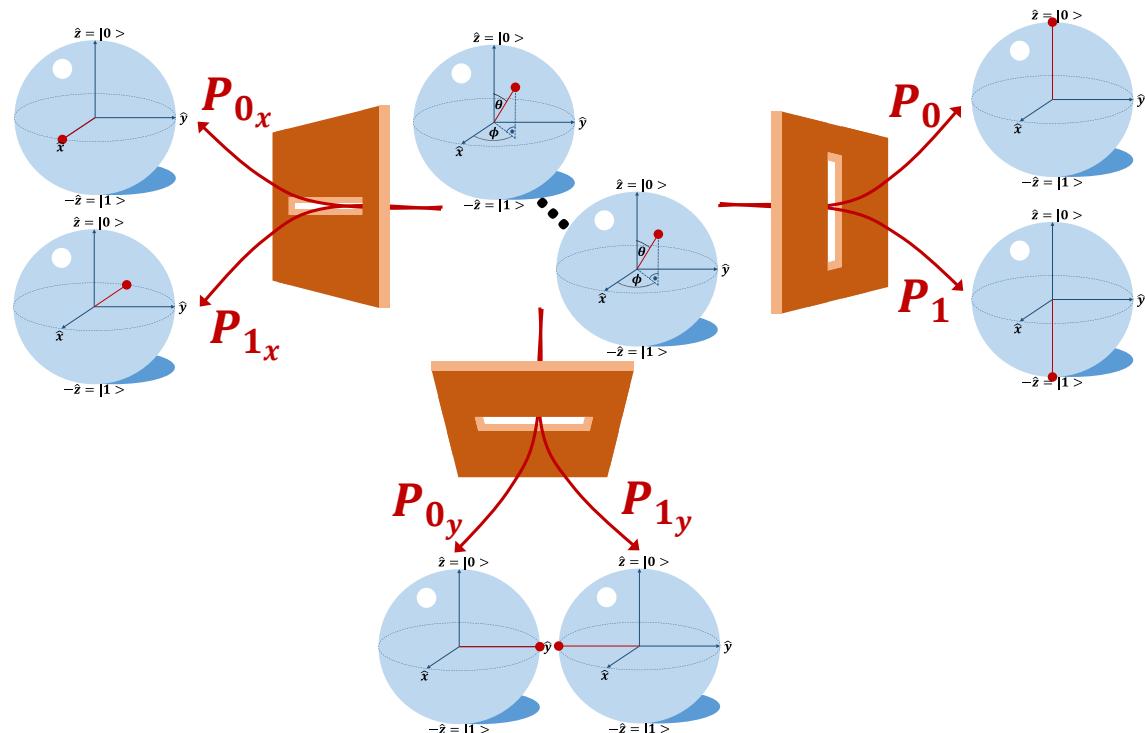
Generator for True Random Numbers

- Commercially available, see e.g.
 - <https://www.magiqtech.com/solutions/network-security/>
 - <https://www.idquantique.com/random-number-generation/>



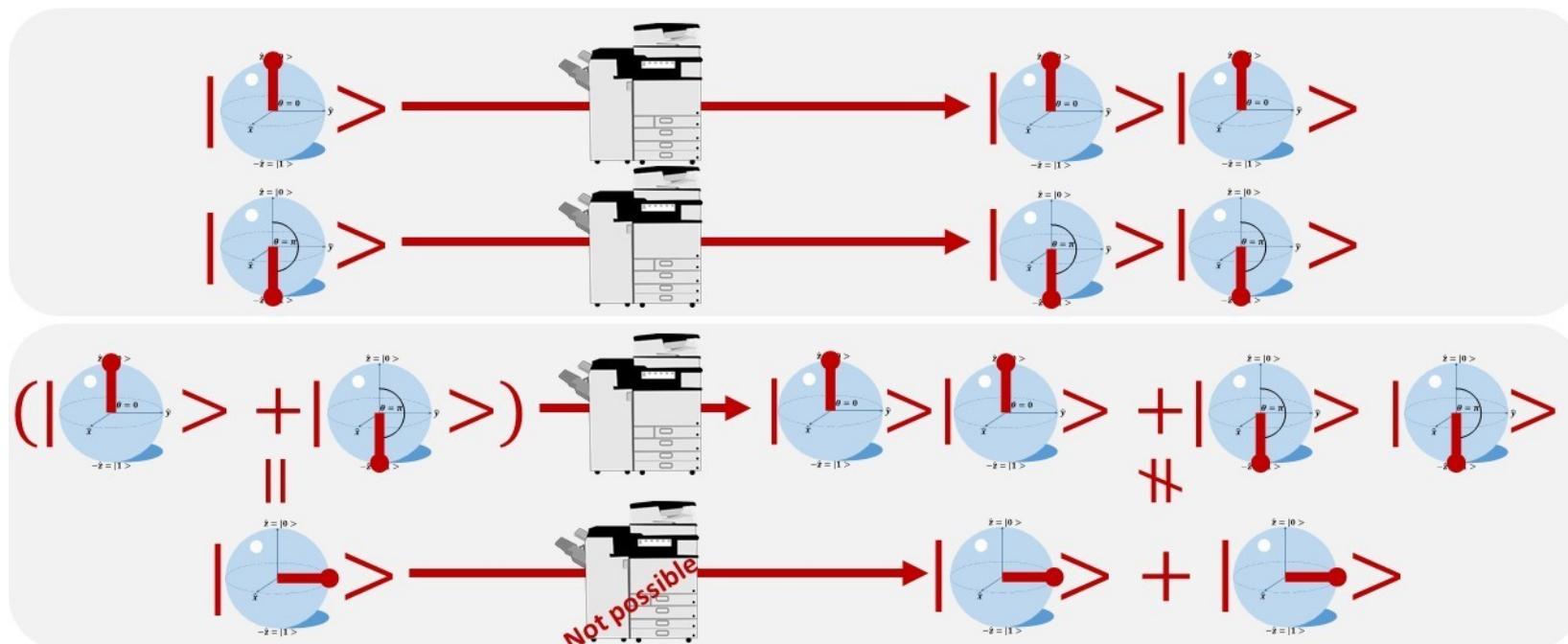
Determining the states (θ, ϕ) of identical prepared Qubits

- After one measurement in one of the axis (x, y, z), the qubit collapses to $|0_a\rangle$ or $|1_a\rangle$ with $a \in \{x, y, z\}$
- As more identical prepared qubits are measured in a axis, as more the measured distribution of $|0_a\rangle$ and $|1_a\rangle$ is getting closer to P_{0_a} and $P_{1_a} \Rightarrow \theta, \phi$ can be determined



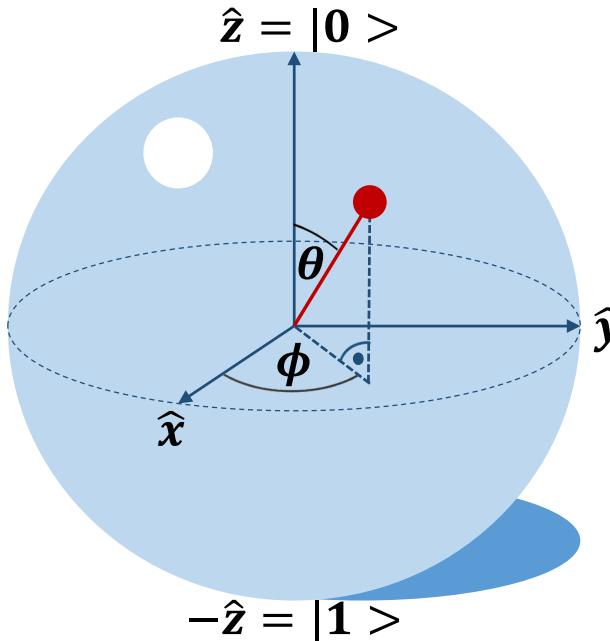
No-Cloning-Theorem of 1 Qubit

- Only not perfect **copying** possible of information in one of the (x, y, z) axis, **other information of superposition gets lost**



Operations via Quantum Logic Gates

- quantum logic gates for 1 qubit: often rotation around one axis
 - Relatively general quantum logic gate: rotation around a specified angle θ :



Rotation operator for rotation around x -axis:

$$R_X(\theta) = e^{-iX\frac{\theta}{2}} = \begin{bmatrix} \cos\frac{\theta}{2} & -i \sin\frac{\theta}{2} \\ -i \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}$$

e.g. Pauli _{x} -Gate:

Rotation Matrix

Quantum Circuit

Table of in- & outputs:

$$NOT = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$|In\rangle \oplus |Out\rangle$$

Alternatively: $|In\rangle \xrightarrow[X]{} |Out\rangle$

In	Out
$ 0\rangle$	$ 1\rangle$
$ 1\rangle$	$ 0\rangle$
$\frac{1}{\sqrt{2}} (0\rangle + 1\rangle)$	$\frac{1}{\sqrt{2}} (0\rangle + 1\rangle)$
$\frac{3i}{5} 0\rangle + \frac{4}{5} 1\rangle$	$\frac{4}{5} 0\rangle + \frac{3i}{5} 1\rangle$

Controlled NOT (CNOT)-Gate

- "*If the control bit is set, then it flips the target bit.*"

Quantum Circuit	Table of in- & outputs			Rotation Matrix R
	Inputs	Output		
Control C	Target T_{before}	Target T_{after}		
$ C\rangle T_{before}\rangle \xrightarrow{\oplus} C\rangle T_{after}\rangle$	$ 0\rangle$	$ 0\rangle$	$ 0\rangle$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
	$ 0\rangle$	$ 1\rangle$	$ 1\rangle$	
	$ 1\rangle$	$ 0\rangle$	$ 1\rangle$	
	$ 1\rangle$	$ 1\rangle$	$ 0\rangle$	

$R \cdot \frac{1}{\sqrt{2}} (|01\rangle + |11\rangle) = \frac{1}{\sqrt{2}} (|01\rangle + |10\rangle)$

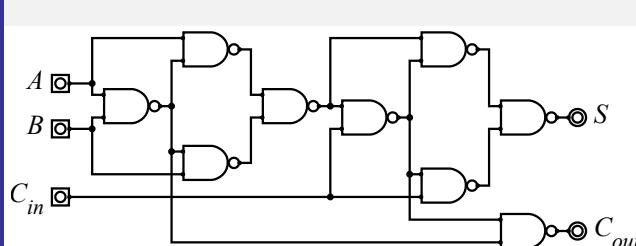
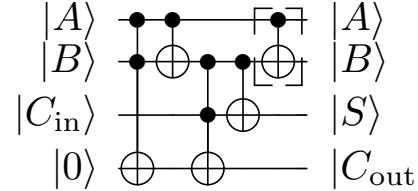
- reversible gate: 2 applications of CNOT retrieves the original input
- Classical analog of the CNOT gate is a reversible XOR gate (i.e., they have analogous in- & and outputs for $\{|0\rangle, |1\rangle\}$ inputs)
- $|a, b\rangle \mapsto |a, a \oplus b\rangle$, where \oplus is XOR

Bell States via Entanglement

Entanglement	Quantum Circuit	Table of in- & outputs								
Correlated	$ A\rangle$ $ A\rangle$ $ 0\rangle$ $ B\rangle$	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> </tr> </thead> <tbody> <tr> <td>$0\rangle$</td> <td>$0\rangle$</td> </tr> <tr> <td>$1\rangle$</td> <td>$1\rangle$</td> </tr> <tr> <td colspan="2">$\frac{1}{\sqrt{2}} (00\rangle + 11\rangle)$</td> </tr> </tbody> </table>	A	B	$ 0\rangle$	$ 0\rangle$	$ 1\rangle$	$ 1\rangle$	$\frac{1}{\sqrt{2}} (00\rangle + 11\rangle)$	
A	B									
$ 0\rangle$	$ 0\rangle$									
$ 1\rangle$	$ 1\rangle$									
$\frac{1}{\sqrt{2}} (00\rangle + 11\rangle)$										
Anti-Correlated	$ A\rangle$ $ A\rangle$ $ 1\rangle$ $ B\rangle$	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> </tr> </thead> <tbody> <tr> <td>$0\rangle$</td> <td>$1\rangle$</td> </tr> <tr> <td>$1\rangle$</td> <td>$0\rangle$</td> </tr> <tr> <td colspan="2">$\frac{1}{\sqrt{2}} (01\rangle + 10\rangle)$</td> </tr> </tbody> </table>	A	B	$ 0\rangle$	$ 1\rangle$	$ 1\rangle$	$ 0\rangle$	$\frac{1}{\sqrt{2}} (01\rangle + 10\rangle)$	
A	B									
$ 0\rangle$	$ 1\rangle$									
$ 1\rangle$	$ 0\rangle$									
$\frac{1}{\sqrt{2}} (01\rangle + 10\rangle)$										

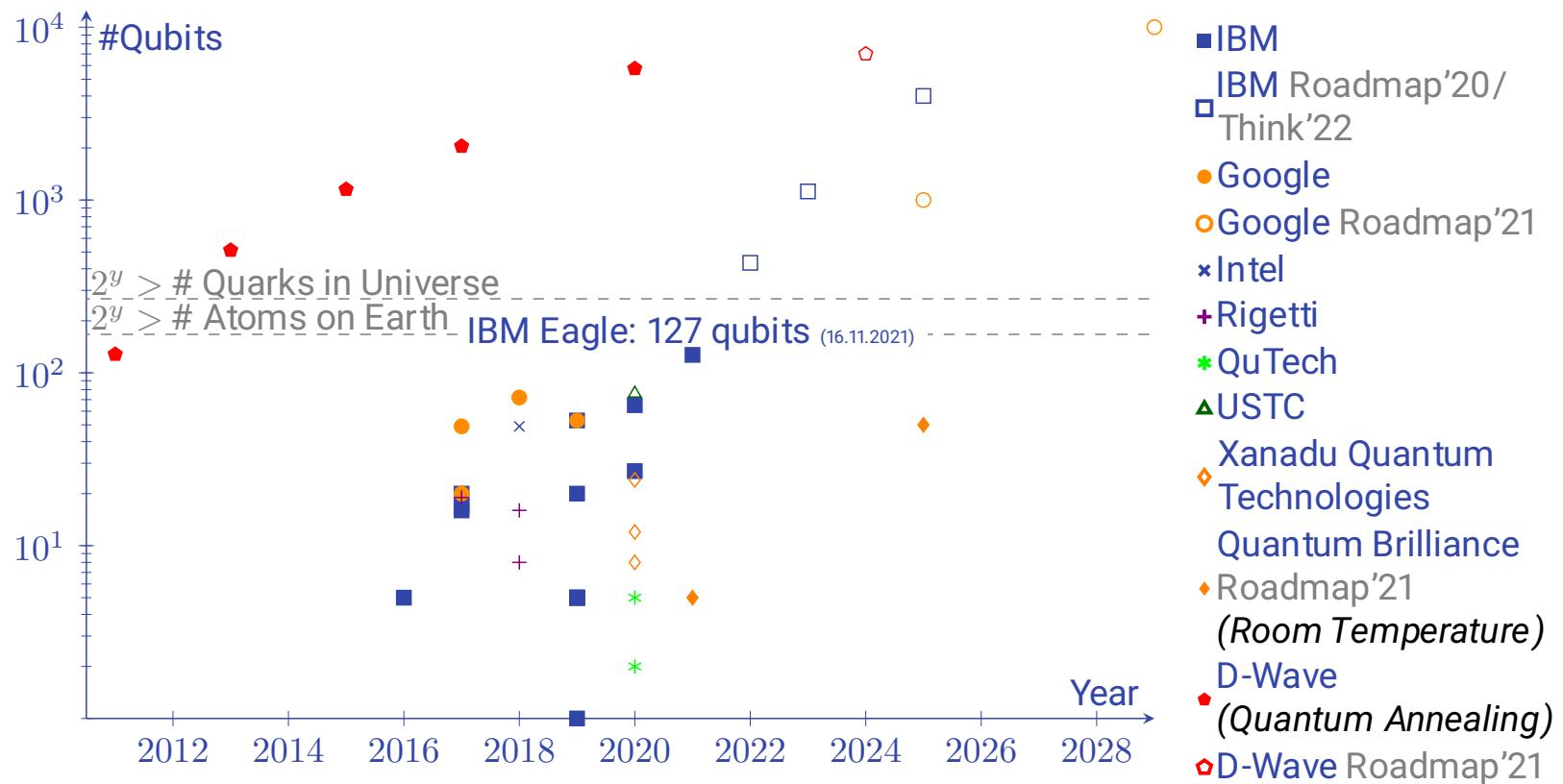
- Even if the entangled qubits are at different locations, they are still entangled \Rightarrow quantum teleportation (phenomena appears to happen instantaneously ignoring speed of light: open question in physics)
 - As of 2017 experimentally verified for distances of up to 1200 kilometers
- Succeeding operations on entangled qubits do not change the state of the other entangled qubits if the operation is not a measurement

Digital versus Quantum Circuits

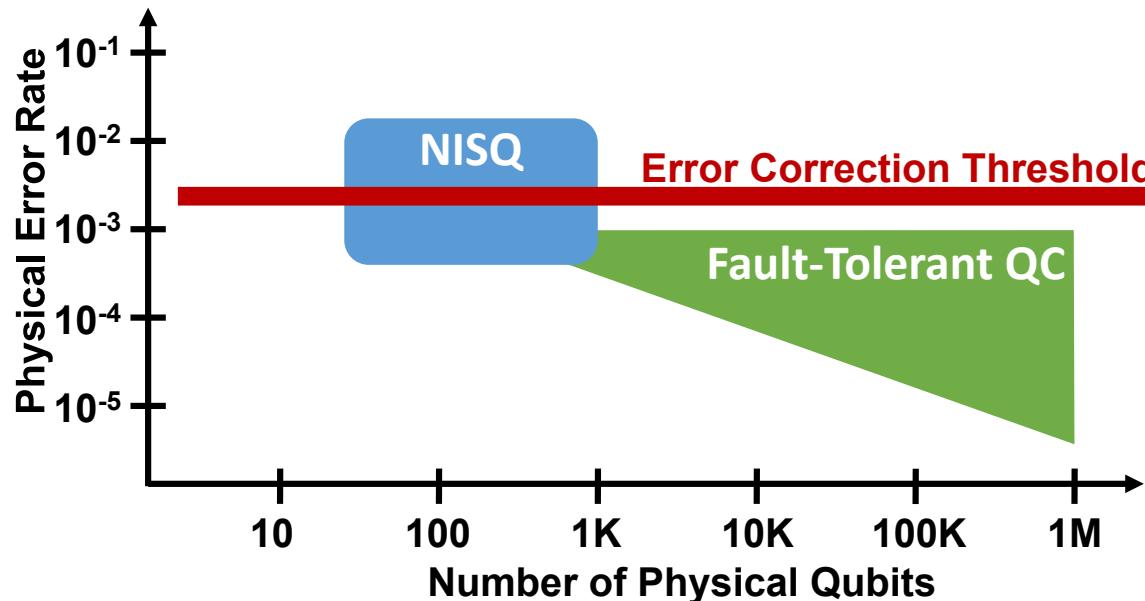
	Digital Circuit	Quantum Circuit																																																																	
Building Blocks	<p>Logic Gates</p> 	<p>Quantum Logic Gates</p> 																																																																	
Full Adder Example	<p>consists of NAND gates</p> <table border="1" data-bbox="245 899 772 1260"> <thead> <tr> <th colspan="3">Inputs</th> <th colspan="2">Outputs</th> </tr> <tr> <th>A</th> <th>B</th> <th>C_{in}</th> <th>C_{out}</th> <th>S</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	Inputs			Outputs		A	B	C _{in}	C _{out}	S	0	0	0	0	0	0	0	1	0	1	0	1	0	0	1	0	1	1	1	0	1	0	0	0	1	1	0	1	1	0	1	1	0	1	0	1	1	1	1	1	<p>consists of Toffoli and CNOT gates¹</p> <p> 0> and 1> as input: Output is 0> and 1> analogous to digital circuit.</p> <p>Superpositions as input: Superpositions as output with corresponding probabilities for basic quantum states, e.g.:</p> <table border="1" data-bbox="881 1076 1726 1308"> <thead> <tr> <th>A</th><th>B</th><th>C_{in}</th><th>C_{out}</th><th>S</th></tr> </thead> <tbody> <tr> <td>$\frac{1}{\sqrt{2}}(0\rangle + 1\rangle)$</td><td> 0></td><td> 0></td><td>$\frac{1}{\sqrt{2}}(0000\rangle + 1001\rangle)$</td><td>$\text{ABC}_{\text{out}}\text{S}:$</td></tr> <tr> <td>$\frac{1}{\sqrt{2}}(0\rangle + 1\rangle)$</td><td>$\begin{array}{c} A\rangle \\ 1\rangle \end{array}$</td><td>$\begin{array}{c} A\rangle \\ \oplus \\ B\rangle \end{array}$</td><td> 0></td><td> 0></td></tr> </tbody> </table>	A	B	C _{in}	C _{out}	S	$\frac{1}{\sqrt{2}}(0\rangle + 1\rangle)$	0>	0>	$\frac{1}{\sqrt{2}}(0000\rangle + 1001\rangle)$	$\text{ABC}_{\text{out}}\text{S}:$	$\frac{1}{\sqrt{2}}(0\rangle + 1\rangle)$	$\begin{array}{c} A\rangle \\ 1\rangle \end{array}$	$\begin{array}{c} A\rangle \\ \oplus \\ B\rangle \end{array}$	0>	0>
Inputs			Outputs																																																																
A	B	C _{in}	C _{out}	S																																																															
0	0	0	0	0																																																															
0	0	1	0	1																																																															
0	1	0	0	1																																																															
0	1	1	1	0																																																															
1	0	0	0	1																																																															
1	0	1	1	0																																																															
1	1	0	1	0																																																															
1	1	1	1	1																																																															
A	B	C _{in}	C _{out}	S																																																															
$\frac{1}{\sqrt{2}}(0\rangle + 1\rangle)$	0>	0>	$\frac{1}{\sqrt{2}}(0000\rangle + 1001\rangle)$	$\text{ABC}_{\text{out}}\text{S}:$																																																															
$\frac{1}{\sqrt{2}}(0\rangle + 1\rangle)$	$\begin{array}{c} A\rangle \\ 1\rangle \end{array}$	$\begin{array}{c} A\rangle \\ \oplus \\ B\rangle \end{array}$	0>	0>																																																															

¹The dotted square marks a superfluous gate if uncomputation to restore the B output is not required. [Feynman, 1986]

Timeline of Quantum Computers

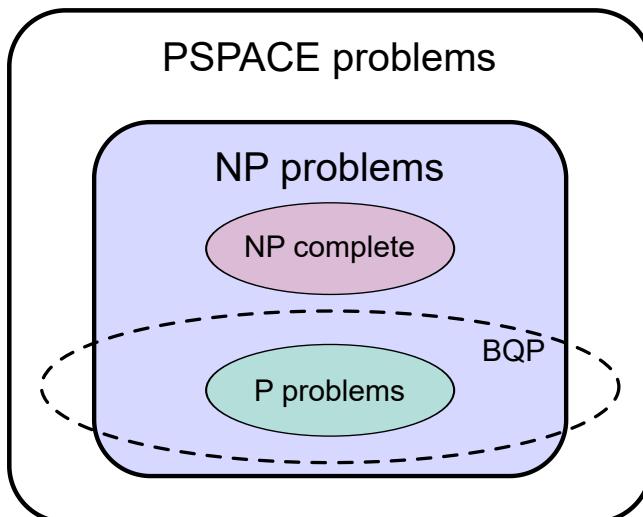


Noisy Intermediate-Scale Quantum (NISQ)



- quantum computers with 50-100 qubits: noise in quantum gates limits the size of quantum circuits that can be executed reliably
- Such NISQ devices may be able to perform tasks which surpass the capabilities of today's classical digital computers with application areas like quantum chemistry, optimization & machine learning
- 100-qubit quantum computers are (only) intermediate technologies

Suspected Shape of BQP



- **PSPACE** is the set of all decision problems that can be solved by a Turing machine using a polynomial amount of space
- **NP: nondeterministic polynomial time** is the set of problems that can be solved in polynomial time by a nondeterministic Turing machine
- A problem is said to be **NP-hard** if everything in NP can be transformed in polynomial time into it even though it may not be in NP
- A problem is **NP-complete** if it is both in NP and NP-hard
- **P: PTIME** contains all decision problems that can be solved by a deterministic Turing machine in polynomial time
- **BQP: bounded-error quantum polynomial time** is the class of decision problems solvable by a quantum computer in polynomial time, with an error probability of at most 1/3 for all instances
- Note: **Neither $P \neq NP$ nor $P \neq PSPACE$ has been proven**, and if either of these are equal, the shape of BQP would be different



Potential of Quantum Algorithms

- **Quantum Algorithm Zoo ↗** as example of collection of *important* quantum algorithms

Covered Years	1974-today
#Investigated References	430 (visited on October 2021)
#Algorithms	64
Speedups	Superpolynomial: 31, Polynomial: 27, Constant factor: 1, Varies: 3, Various: 1, Unknown: 1

Terminology:

α : positive constant

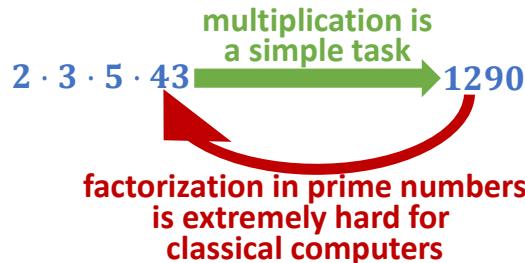
$C(n)$: runtime of the best known classical algorithm

$Q(n)$: runtime of the quantum algorithm

Superpolynomial Speedup: $C = 2^{\Omega(Q^\alpha)}$

Polynomial Speedup: otherwise

Very Important Quantum Algorithms 1/2: Shor's Algorithm¹



- factoring integers in polynomial time
 - Depth of quantum circuit² to factor integer N :
 $O((\log N)^2 (\log \log N) (\log \log \log N))$
 - superpolynomial speedup, i.e., almost exponentially faster than the most efficient known classical factoring algorithm (general number field sieve):
 $O(e^{1.9(\log N)^{\frac{1}{3}} (\log \log N)^{\frac{2}{3}}})$
- Important for cryptography → Post-Quantum Cryptography
- Most quantum algorithms with superpolynomial speedup like Shor's algorithm are based on quantum Fourier transforms (quantum analogue of inverse discrete Fourier transform)



Very Important Quantum Algorithms 2/2: Grover's Search Algorithm

- Black box function (oracle) $f : \{0, \dots, 2^b - 1\} \mapsto \{\text{true}, \text{false}\}$
- Grover's search algorithm finds one $x \in \{0, \dots, 2^b - 1\}$, such that $f(x) = \text{true}$
 - if there is only one solution: $\frac{\pi}{4} \cdot \sqrt{2^b}$ basic steps each of which calls f
Let $f'(b)$ be runtime complexity of f for testing x to be true:
 $\Rightarrow O(\sqrt{2^b} \cdot f'(b))$
 - if there are k possible solutions: $O(\sqrt{\frac{2^b}{k}} \cdot f'(b))$
- Basis of many other quantum algorithms and applications

Algorithms (used e.g. in Query Optimization) and their Quantum Counterparts

Query Optimization Approach	Basic Algorithm	Quantum Computing Counterpart
[S+79] ↗	Dynamic Programming [E04] ↗	[R19] ↗ [A+19] ↗
[IW87] ↗, QA: [TK16] ↗	Simulated Annealing [KGV83] ↗	[J+11] ↗
[MP18] ↗ [Y+20] ↗ [W+19] ↗ [O+19] ↗	Reinforcement Learning [BSB81] ↗	[S+21] ↗ [DCC05] ↗
[GPK94] ↗	Random Walk [BN70] ↗	[ADZ93] ↗ [A+01] ↗
[BFI91] ↗	Genetic Algorithm [H92] ↗	[W+13] ↗
[TC19] ↗	Ant Colony Optimization [CDM91] ↗ [DBS06] ↗	[WNF07] ↗ [G+20] ↗

This list is not complete...

- Please check my lecture about quantum computing:
[↗https://www.ifis.uni-luebeck.de/~groppe/lectures/qc](https://www.ifis.uni-luebeck.de/~groppe/lectures/qc)



Open Challenges for QC for Databases

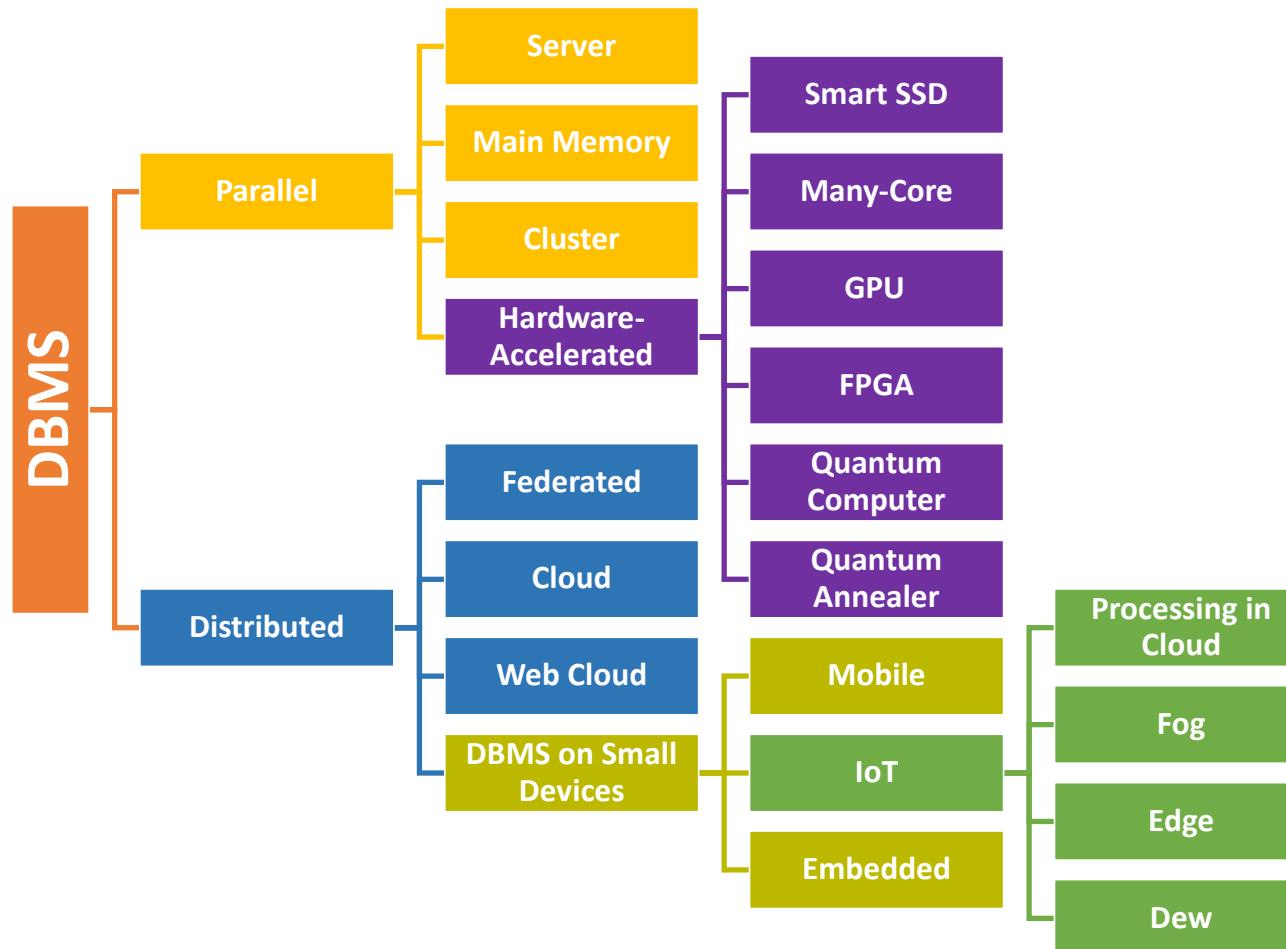
- Are QC counterparts of basic algorithms used in query optimizations suitable for speeding up databases?
- What should be the properties of a quantum computer (e.g. #qubits, latencies of gates) to achieve certain speedups?
- How to combine classical and quantum computing algorithms to achieve good speedups with few qubits?
(...for running database optimizations on current available quantum computers...)
- What other database domains besides query optimization benefit from quantum computers?
(In short: those based on mathematical optimization problems, but also other...?)



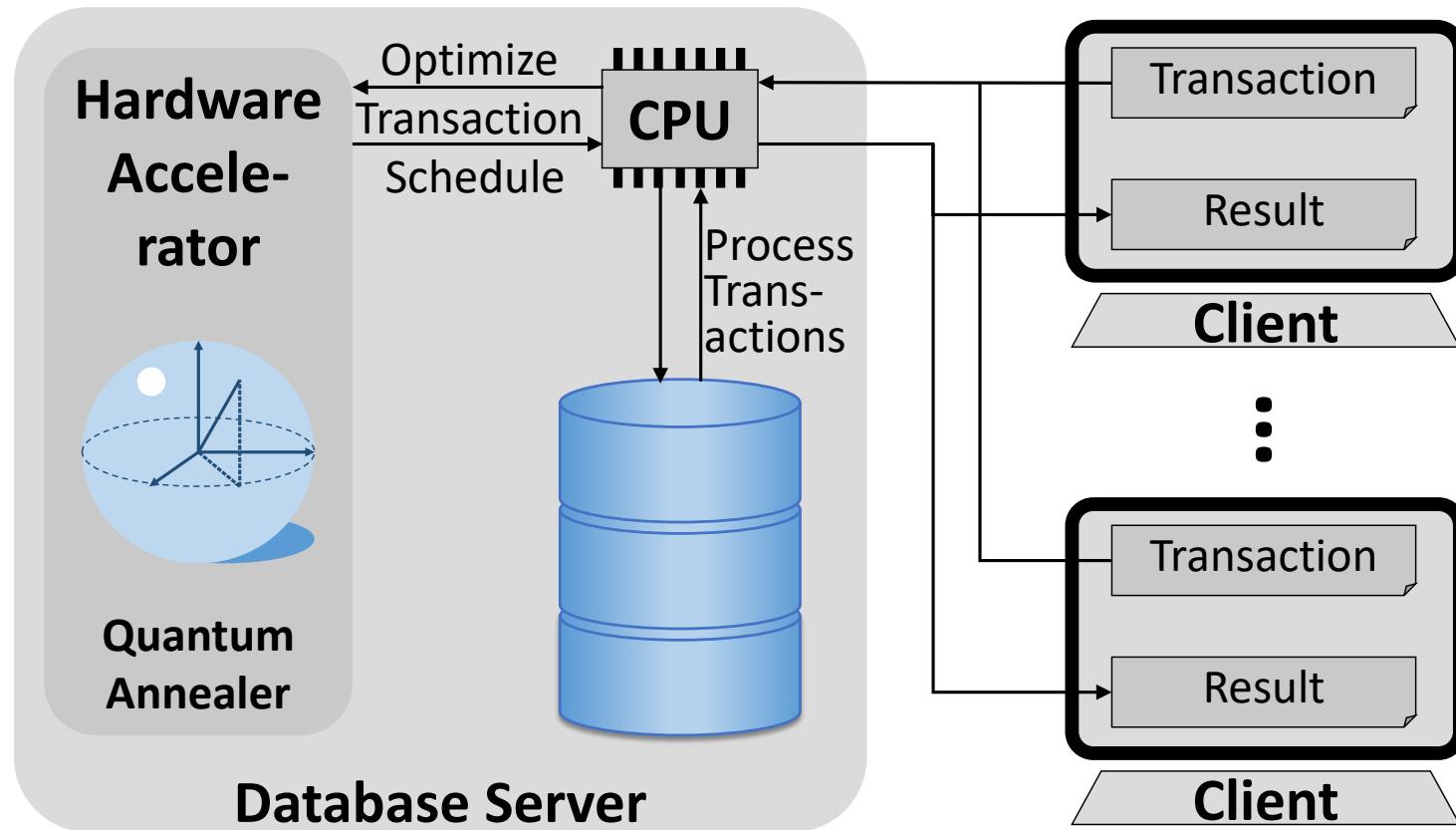
QC4DB: Accelerating Relational Database Management Systems via Quantum Computing

- Project Website@Quantentechnologien ↗
- Project funded by BMBF
 - Duration 3 years, 1.8M Euros
- Topics
 - Query Optimization
 - Optimizing Transaction Schedules
- of an open source relational database management system
- Partners
 - University of Lübeck (Coordinator Sven Groppe)
 - Hardware-Acceleration of Databases
 - Website: ↗<https://www.ifis.uni-luebeck.de/~groppe/>
 - Quantum Brilliance GmbH
 - Room Temperature Diamond Quantum Accelerators
 - Website: ↗<https://quantumbrilliance.com/>
- Collaborations are welcome!

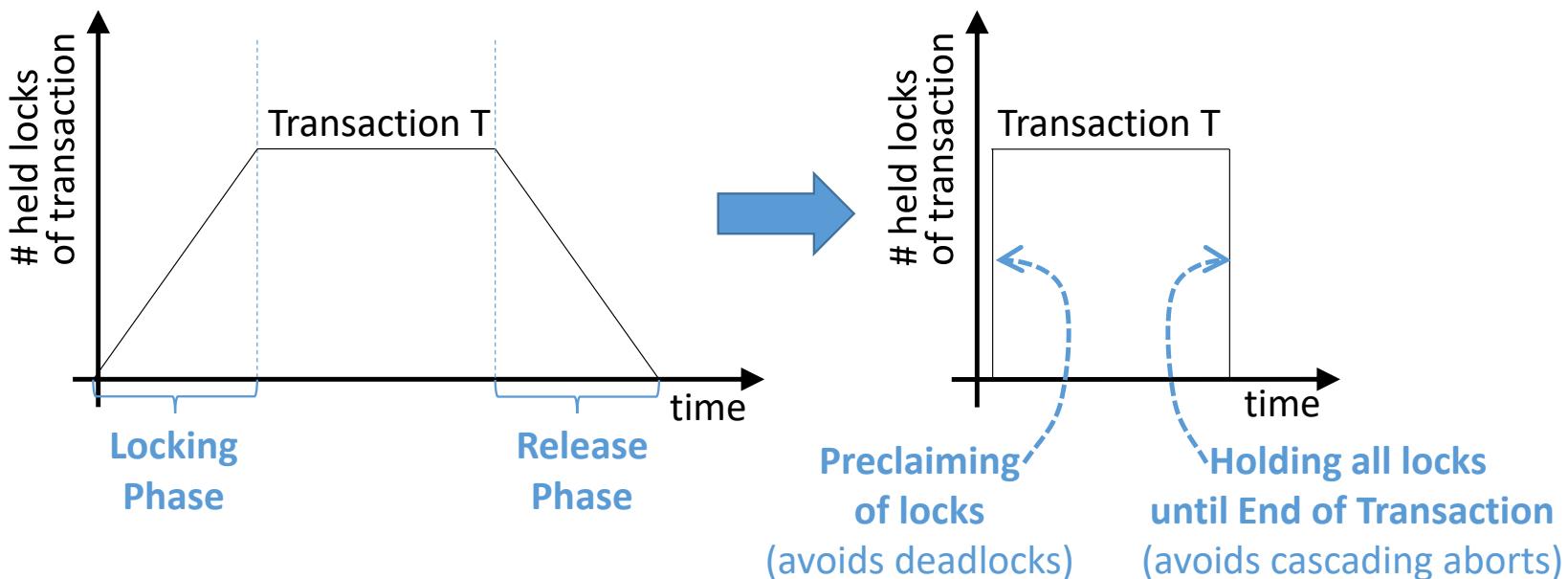
Platform-specific types of DBMS



Using Hardware Accelerator for optimizing Transaction Schedules



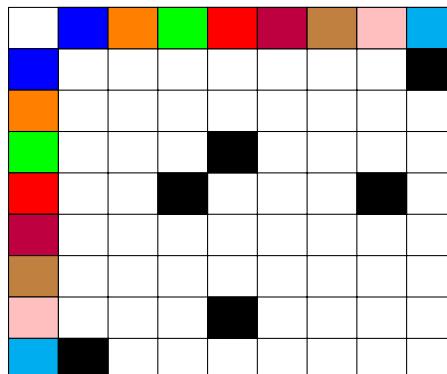
2 Phase Locking (2PL) versus Strict Conservative 2PL



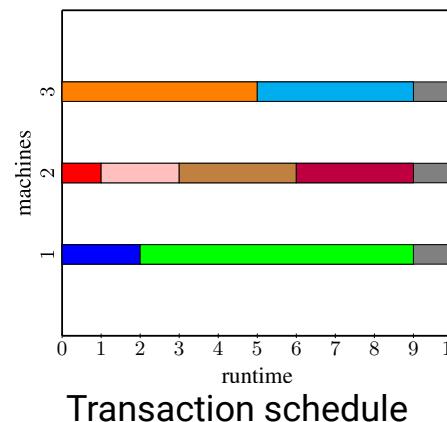
- required locks to be determined by
 - static analysis of transaction, or if static analysis is not possible:
 - an additional phase at runtime before transaction processing
 - A. Thomson et al., "Calvin: Fast distributed transactions for partitioned database systems", SIGMOD 2012.

Optimizing Transaction Schedules

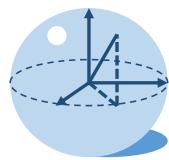
- Variant of job shop schedule problem (JSSP):
 - Multi-Core CPU
 - Process whole job (here transaction) on core X
 - Schedule: \forall cores: Sequence of jobs to be processed
 - What is the optimal schedule for minimal overall processing time?
- Additionally to JSSP:
Blocking transactions not to be processed in parallel
- Example:



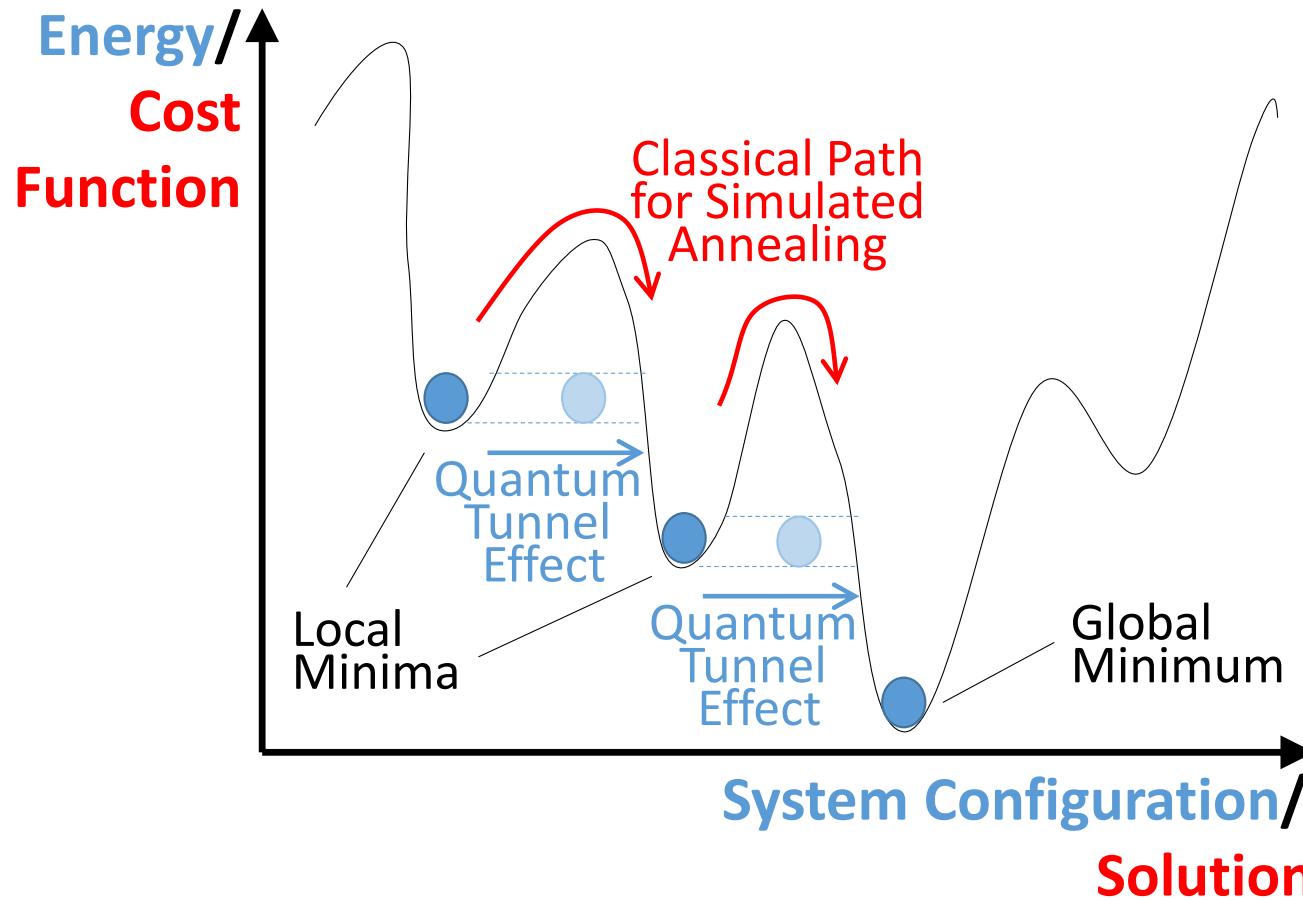
Black: Blocking transactions



- JSSP is among the hardest combinatorial optimizing problems *
- \Rightarrow Hardware accelerating the optimization of transaction schedules



Quantum versus Simulated Annealing





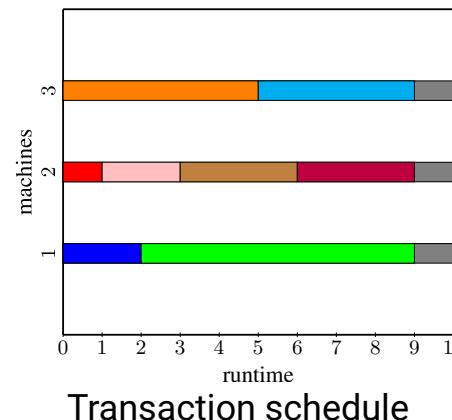
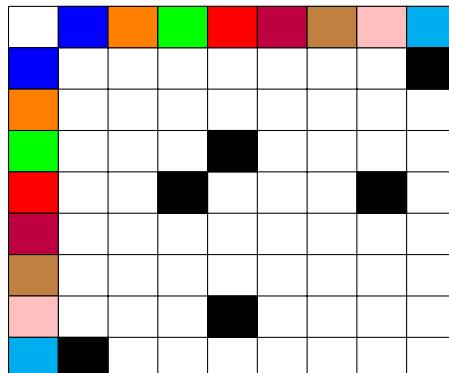
Optimizing Transaction Schedules via Quantum Annealing

- Transaction Model
 - T: set of transactions with $|T| = n$
 - M: set of machines with $|M| = k$
 - $O \subseteq T \times T$: set of **blocking** transactions
 - l_i : length of transaction i
 - R: maximum execution time
 - upper bound $r_i = R - l_i$ for start time of transaction i
- Quadratic unconstrained binary optimization (QUBO) problems (solving is NP-hard)
 - A QUBO-problem is defined by N weighted binary variables $X_1, \dots, X_N \in \{0, 1\}$, either as linear or quadratic term to be minimized:

$$\sum_{0 < i \leq N} w_i X_i + \sum_{i \leq j \leq N} w_{ij} X_i X_j, \text{ where } w_i, w_{ij} \in \mathbb{R}$$

Optimizing Transaction Schedules via Quantum Annealing

- Multi-Core CPU
 - Process whole transaction on core X
- Solution formulated as set of binary variables
 - $X_{i,j,s}$ is 1 iff transaction t_i is started at time s on machine m_j , otherwise 0
- Example:



- Solution:
 $X_{1,1,0}, X_{3,1,2}, X_{4,2,0},$
 $X_{7,2,1}, X_{6,2,3}, X_{5,2,6},$
 $X_{2,3,0}, X_{8,3,5}$



Optimizing Transaction Schedules via Quantum Annealing

- **Valid Solution**
 - A: each transaction starts exactly once

$$A = \sum_{i=1}^n \left(\underbrace{\sum_{j=1}^k}_{\text{transactions}} \underbrace{\sum_{s=0}^{r_i}}_{\text{machines start times}} X_{i,j,s} - 1 \right)^2$$

Example: $R = 2$

$T = \{t_1, t_2, t_3\}$ with $|T| = n = 3$

$M = \{m_1, m_2\}$ with $|M| = k = 2$

$O = \{(t_2, t_3)\}$

$l_1 = 2, l_2 = 1, l_3 = 1$

$r_1 = 0, r_2 = 1, r_3 = 1$

$$\begin{aligned} A = & (X_{1,1,0} + X_{1,2,0} - 1)^2 \\ & + (X_{2,1,0} + X_{2,1,1} + X_{2,2,0} + X_{2,2,1} - 1)^2 \\ & + (X_{3,1,0} + X_{3,1,1} + X_{3,2,0} + X_{3,2,1} - 1)^2 \end{aligned}$$

Optimizing Transaction Schedules via Quantum Annealing

- Valid Solution

- B: transactions cannot be executed at the same time on the same machine

transactions without t_n remaining transactions

$$B = \underbrace{\sum_{j=1}^k}_{\text{machines}} \underbrace{\sum_{i_1=1}^{n-1}}_{\text{start times}} \underbrace{\sum_{s_1=0}^{r_{i_1}}}_{\text{invalid start times}} \underbrace{\sum_{i_2=i_1+1}^n}_{\text{start times}} \underbrace{\sum_{s_2=q}^p}_{\text{invalid start times}} X_{i_1,j,s_1} X_{i_2,j,s_2} \text{ for } q = \max\{0, s_1 - l_{i_2} + 1\}, p = \min\{s_1 + l_{i_1}, r_{i_2}\}$$

Example: $R = 2$

$T = \{t_1, t_2, t_3\}$ with $|T| = n = 3$

$M = \{m_1, m_2\}$ with $|M| = k = 2$

$O = \{(t_2, t_3)\}$

$l_1 = 2, l_2 = 1, l_3 = 1$

$r_1 = 0, r_2 = 1, r_3 = 1$

$$\begin{aligned} B = & X_{1,1,0}X_{2,1,0} + X_{1,1,0}X_{2,1,1} + X_{1,1,0}X_{3,1,0} \\ & + X_{1,1,0}X_{3,1,1} + X_{2,1,0}X_{3,1,0} + X_{2,1,1}X_{3,1,1} \\ & + X_{1,2,0}X_{2,2,0} + X_{1,2,0}X_{2,2,1} + X_{1,2,0}X_{3,2,0} \\ & + X_{1,2,0}X_{3,2,1} + X_{2,2,0}X_{3,2,0} + X_{2,2,1}X_{3,2,1} \end{aligned}$$

Optimizing Transaction Schedules via Quantum Annealing

- Valid Solution
 - C: transactions that block each other cannot be executed at the same time

$$C = \sum_{\{(t_{i_1}, t_{i_2})\} \in O} \underbrace{\sum_{j_1=1}^k \sum_{s_1=0}^{r_{i_1}}}_{\text{machines}} \sum_{j_2 \in J} \underbrace{\sum_{s_2=q}^p}_{\text{remaining machines}} X_{i_1, j_1, s_1} X_{i_2, j_2, s_2} \quad \text{for } J = \{1, \dots, k\} \setminus \{j_1\}, q = \max\{0, s_1 - l_{i_2} + 1\}, p = \min\{s_1 + l_{i_1}, r_{i_2}\}$$

blocking transactions start times invalid start times

Example: $R = 2$

$T = \{t_1, t_2, t_3\}$ with $|T| = n = 3$

$M = \{m_1, m_2\}$ with $|M| = k = 2$

$O = \{(t_2, t_3)\}$

$l_1 = 2, l_2 = 1, l_3 = 1$

$r_1 = 0, r_2 = 1, r_3 = 1$

$$C = X_{2,1,0}X_{3,2,0} + X_{2,1,1}X_{3,2,1} \\ + X_{2,2,0}X_{3,1,0} + X_{2,2,1}X_{3,1,1}$$

Optimizing Transaction Schedules via Quantum Annealing

- Optimal Solution

- D: minimizing the maximum execution time

$$D = \sum_{i=1}^n \sum_{j=1}^k \sum_{s=0}^{r_i} w_{s+l_i} X_{i,j,s}, \text{ where } w_{s+l_i} = \frac{(k+1)^{s+l_i-1}}{(k+1)^R} < 1$$

- Increasing weights: Weight of step n is larger than of all preceding steps 1 to n-1 \Rightarrow preferring transactions ending earlier
 - Weights in A, B and C ≥ 1
 \Rightarrow first priority is validity, second priority is optimality

Example: $R = 2$

$T = \{t_1, t_2, t_3\}$ with $|T| = n = 3$

$M = \{m_1, m_2\}$ with $|M| = k = 2$

$O = \{(t_2, t_3)\}$

$l_1 = 2, l_2 = 1, l_3 = 1$

$r_1 = 0, r_2 = 1, r_3 = 1$

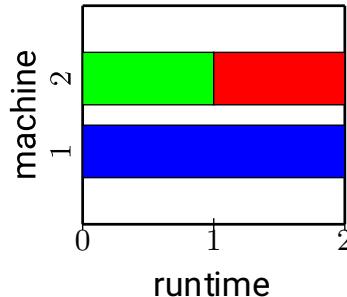
$$\begin{aligned} D = & \frac{3}{9}X_{1,1,0} + \frac{3}{9}X_{1,2,0} \\ & + \frac{1}{9}X_{2,1,0} + \frac{3}{9}X_{2,1,1} + \frac{1}{9}X_{2,2,0} + \frac{3}{9}X_{2,2,1} \\ & + \frac{1}{9}X_{3,1,0} + \frac{3}{9}X_{3,1,1} + \frac{1}{9}X_{3,2,0} + \frac{3}{9}X_{3,2,1} \end{aligned}$$

Optimizing Transaction Schedules via Quantum Annealing

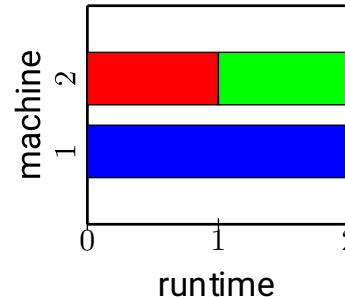
- Overall Solution

- Minimize $P = A + B + C + D$

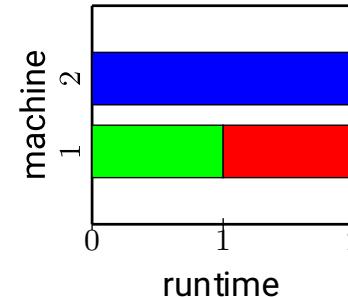
Optimal schedules (transaction 1 in blue, transaction 2 in green and transaction 3 in red) for our example:



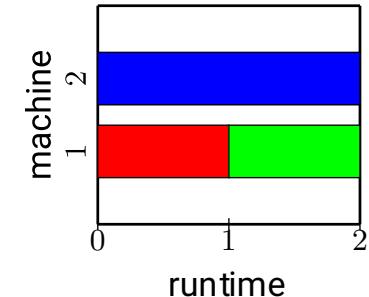
$$X_{1,1,0}, X_{2,2,0}, X_{3,2,1}$$



$$X_{1,1,0}, X_{2,2,1}, X_{3,2,0}$$



$$X_{1,2,0}, X_{2,1,0}, X_{3,1,1}$$



$$X_{1,2,0}, X_{2,1,1}, X_{3,1,0}$$

The result of P is the following value for all 4 different (optimal) schedules:

$$P = A + B + C + D = -3 + 0 + 0 + \frac{7}{9} = -2\frac{2}{9}$$

If the offset is added (optional), then the result is:

$$P = A + B + C + D = -3 + 0 + 0 + \frac{7}{9} + 3 = \frac{7}{9}$$

Optimizing Transaction Schedules via Quantum Annealing

- Experiments on real Quantum Annealer (D-Wave 2000Q cloud service)
 - first minute free (afterwards too much for our budget)
- Versus Simulated Annealing on CPU
- Preprocessing time/Number of QuBits: $O((n \cdot k \cdot R)^2)$

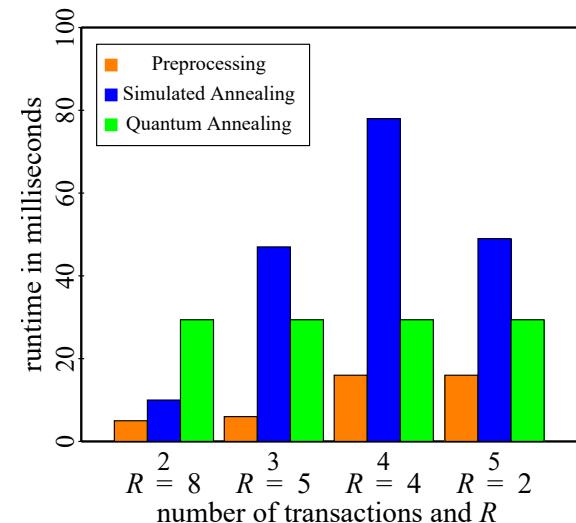
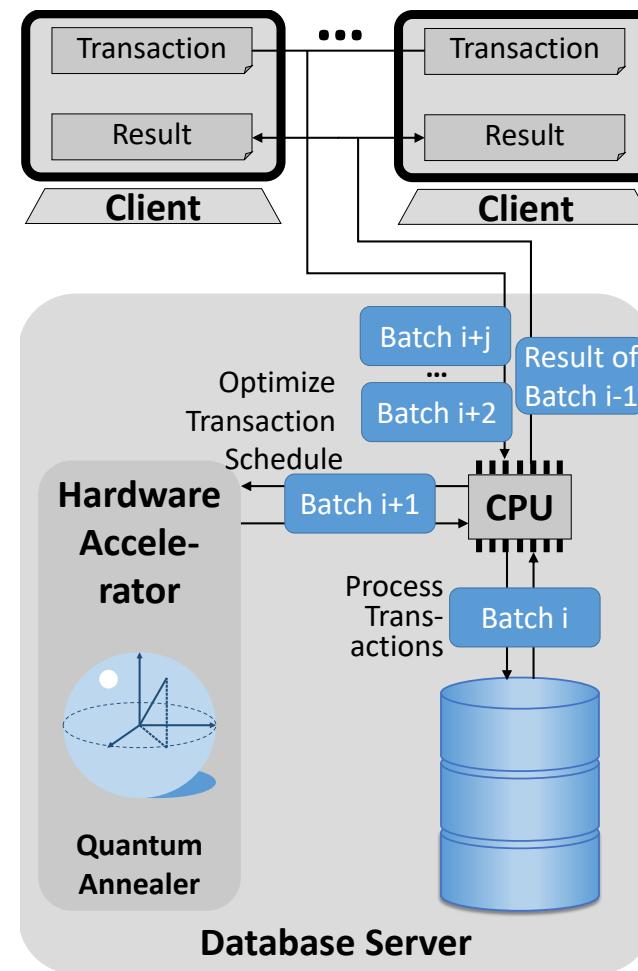


Fig.	k	n	R	O	l_1, \dots, l_n	r_1, \dots, r_n	req. var.
11	2	2	8	$\{\}$	8, 4	0, 4	8
		3	5	$\{(t_1, t_3)\}$	4, 5, 1	1, 0, 4	10
		4	4	$\{(t_2, t_4)\}$	3, 2, 1, 2	1, 2, 3, 2	16
		5	2	$\{(t_1, t_2), (t_4, t_5)\}$	1, 1, 1, 1, 1	1, 1, 1, 1, 1	10

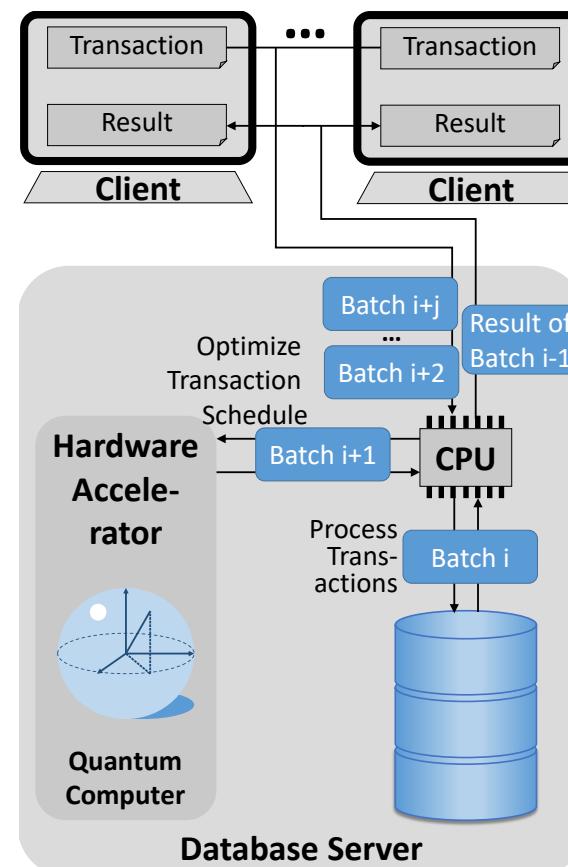
Pipelining for further Speedup



Caching of & Reusing generated formulas to minimize preprocessing time

- Following parameters are fixed:
 - the number k of machines (system does not change during runtime)
 - the number n of transactions (for batches of the same size)
- We observe:
 - The (maximal) execution time R and hence upper bounds of start times r_i, \dots, r_n depend on the lengths of the transactions,
 - the formulas A, B and D depend on the fixed parameters k and n , and on the lengths of the transactions, and
 - the formula C is a sum of sub-formulas depending on k and the lengths of blocking transactions as well as the identifiers of blocking transactions.
⇒ Caching A, B and D , and sub-formulas of C with the key of the lengths of the transactions (orderd by lengths) (Example: using $(1, 1, 2)$ as key)
Further information in paper!

Optimizing Transaction Schedules via Quantum Computing

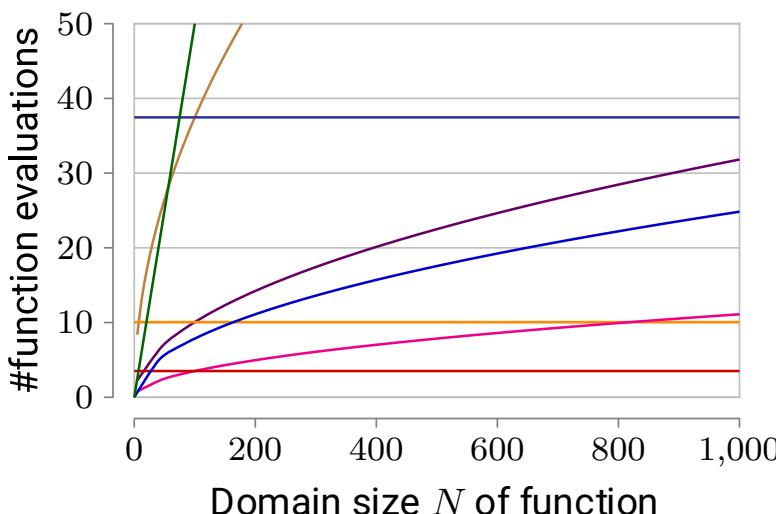




Grover's Search Algorithm

- Black box function $f : \{0, \dots, 2^b - 1\} \mapsto \{\text{true}, \text{false}\}$
- Grover's search algorithm finds one $x \in \{0, \dots, 2^b - 1\}$, such that $f(x) = \text{true}$
 - if there is only one solution: $\frac{\pi}{4} \cdot \sqrt{2^b}$ basic steps each of which calls f
Let $f'(b)$ be runtime complexity of f for testing x to be true:
 $\Rightarrow O(\sqrt{2^b} \cdot f'(b))$
 - if there are k possible solutions: $O(\sqrt{\frac{2^b}{k}} \cdot f'(b))$

Motivation - Quadratic Speedup



- $\sim \frac{N}{2}$ Linear Search
- k known in advance:
- $\sim \frac{\pi}{4} \cdot \sqrt{N}$ Grover's Search for $k = 1$
- $\sim \frac{\pi}{4} \cdot \sqrt{\frac{N}{k}}$ Grover's Search for $k = \frac{5}{100} \cdot N$
- $\sim \frac{\pi}{4} \cdot \sqrt{\frac{N}{k}}$ Grover's Search for $k = 5$
- $k = \frac{5}{100} \cdot N$ unknown in advance:
- $\sim \frac{9}{4} \cdot \sqrt{\frac{N}{k}}$ randomized Grover's Search
- $\sim \frac{8 \cdot \pi}{3} \cdot \sqrt{\frac{N}{k}}$ deterministic Grover's Search
- $k = 5$ unknown in advance:
- $\sim \frac{9}{4} \cdot \sqrt{\frac{N}{k}}$ randomized Grover's Search
- $\sim \frac{8 \cdot \pi}{3} \cdot \sqrt{\frac{N}{k}}$ deterministic Grover's Search

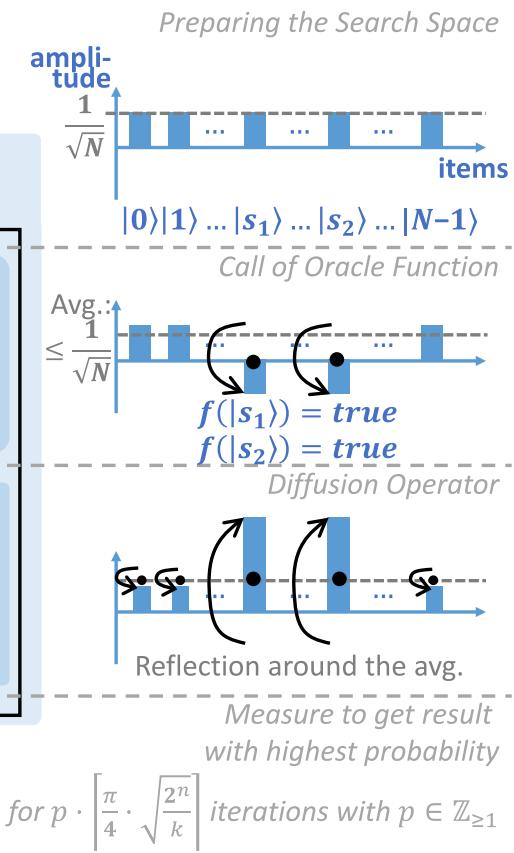
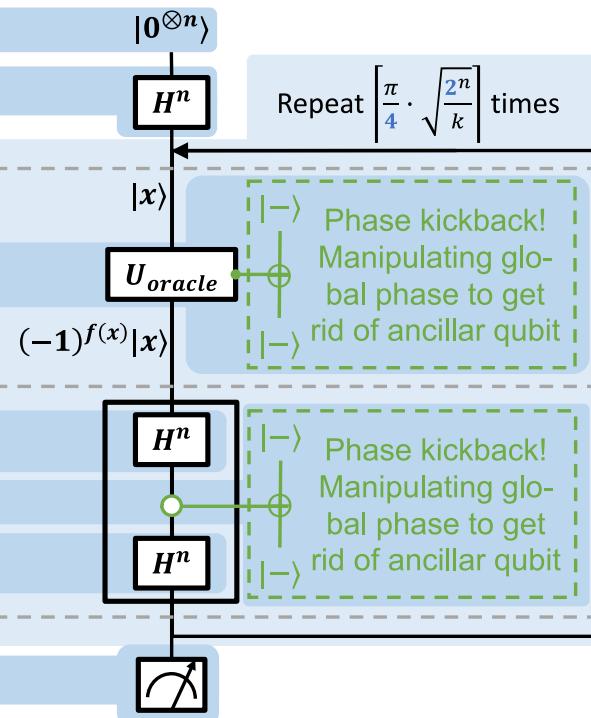
N	Linear Search $k = 1$	Grover $k = 1$	known k	$k = \frac{5}{100} \cdot N$		known k	$k = 5$	
				unknown k	randomized		unknown k	randomized
10	5	2.48				1.11	3.81	11.85
100	50	7.85		3.51		3.51	10.06	37.47
1000	500	24.84		10.06		11.11	31.82	118.48
1000 000	500 000	785.40		37.46		351.24	1006.23	3746.57

Grover's Search for k Solutions

```

def grover_k [n:!N, k:!N] (oracle: const uint[n]!→ lifted B)!N {
    nIterations := ⌈ $\frac{\pi}{4} \cdot \sqrt{\frac{2^n}{k}}$ ⌉;
    cand := 0:uint[n];
    for k in [0..n){ cand[k] := H(cand[k]); }
    for k in [0..nIterations) {
        if oracle(cand) { phase(π); }
        for m in [0..n) { cand[m] := H(cand[m]); }
        if cand!=0 { phase(π); }
        for m in [0..n) { cand[m] := H(cand[m]); }
    }
    return measure(cand) as !N;
}

```



Grover's Search for Unknown Number of Solutions

- If you apply Grover's search with $k \approx$ number of solutions k' , then there is a high probability for success
- Several ways for unknown k :
 - Successively applying Grover's search with $k = N, \frac{N}{2}, \frac{N}{4}, \dots, \frac{N}{2^t}$ until solution is found
 - **Runtime** complexity: With sufficiently high probability, a marked entry will be found by iteration $t = \log_2(\frac{N}{k'}) + c$ for some constant c
 $\Rightarrow \text{\#iterations} \leq \frac{\pi}{4} \left(1 + \sqrt{2} + \sqrt{4} + \dots + \sqrt{\frac{N}{k' \cdot 2^c}} \right) = O\left(\sqrt{\frac{N}{k'}}\right)$
 - Next slide: **Randomized** version with $\frac{9}{4} \sqrt{\frac{N}{k'}}$ iterations...
- These methods need to be called several times in case of not finding a solution in preceding calls
 - The probability for success is high, but never 100%

Grover's Search for Unknown Number of Solutions - Silq code

```
// Popular Lehmer generator that uses the prime modulus 2^32-5
def random(state:!N):(!N)×(!Q) {
    upperlimit := (2^32 - 5);
    newstate := (state · 279470273) % upperlimit;
    newstateQ := newstate as !Q; // for the following division to get a rational number
    return (newstate, newstateQ / upperlimit);
}
// f oracle function, t oracle function as classical function for testing the result!
def grover_unknown_k[n:!N](f: const uint[n] !→ lifted ℂ, t: !N !→ !ℂ, seed:!N):(!Z)×(!N) {
    l := 6/5; // Any value of l strictly between 1 and 4/3 would do...
    m := 1 as !ℝ;
    state := seed; // work with seed to allow for repeated function calls with other results
    while(true){
        (zstate, z) := random(state);
        state = zstate;
        k := floor(z·m) coerce !N;
        result := grover_k[n,k](f);
        if(t(result)){ // call of f would also work, but in this way hybrid approach is more visible
            return (result, state) as (!Z)×(!N);
        }
        if(m>=sqrt(2^n)){ return (-1, state) as (!Z)×(!N); } // different from cited paper: restart!
        m = min(l·m, sqrt(n));
    }
}
```



Optimal Runtime Complexity of Grover's Search

- Grover's algorithm for $k = 1$ and any k : optimal up to sub-constant factors
 - Any algorithm that accesses the database only by using the oracle must apply the oracle at least a $1 - o(1)$ fraction as many times as Grover's algorithm

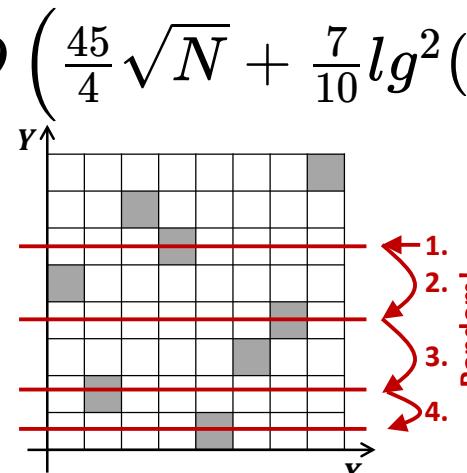


Grover's Search as Basic Algorithm Used in other Approaches

- Solving the collision problem
- Finding Minimum
 - Dynamic Programming
- Solving NP-complete problems by performing exhaustive searches over the set of possible solutions
 - results in quadratic speedup over classical solutions
 - but no polynomial-time solution for NP-complete problems because exponential speedup is **not** reached
 - Example: k -SAT
- ...

Applying Grover - Finding Minimum

- Given function $f : \{0, \dots, N - 1\} \mapsto \mathbb{R}$
- Algorithm to find minimal $f(x)$:
 - Choose randomly $x' \in N$
 - Threshold $t := f(x')$
 - while(true)
 - $r :=$ Grover's Search for unknown k with oracle $f(x) < t$
 - if($f(r) < t$) $t := f(r)$ else return t
- Runtime Complexity: $O\left(\frac{45}{4}\sqrt{N} + \frac{7}{10}\lg^2(N)\right)$



Applying Grover - Finding Minimum

- Variants:

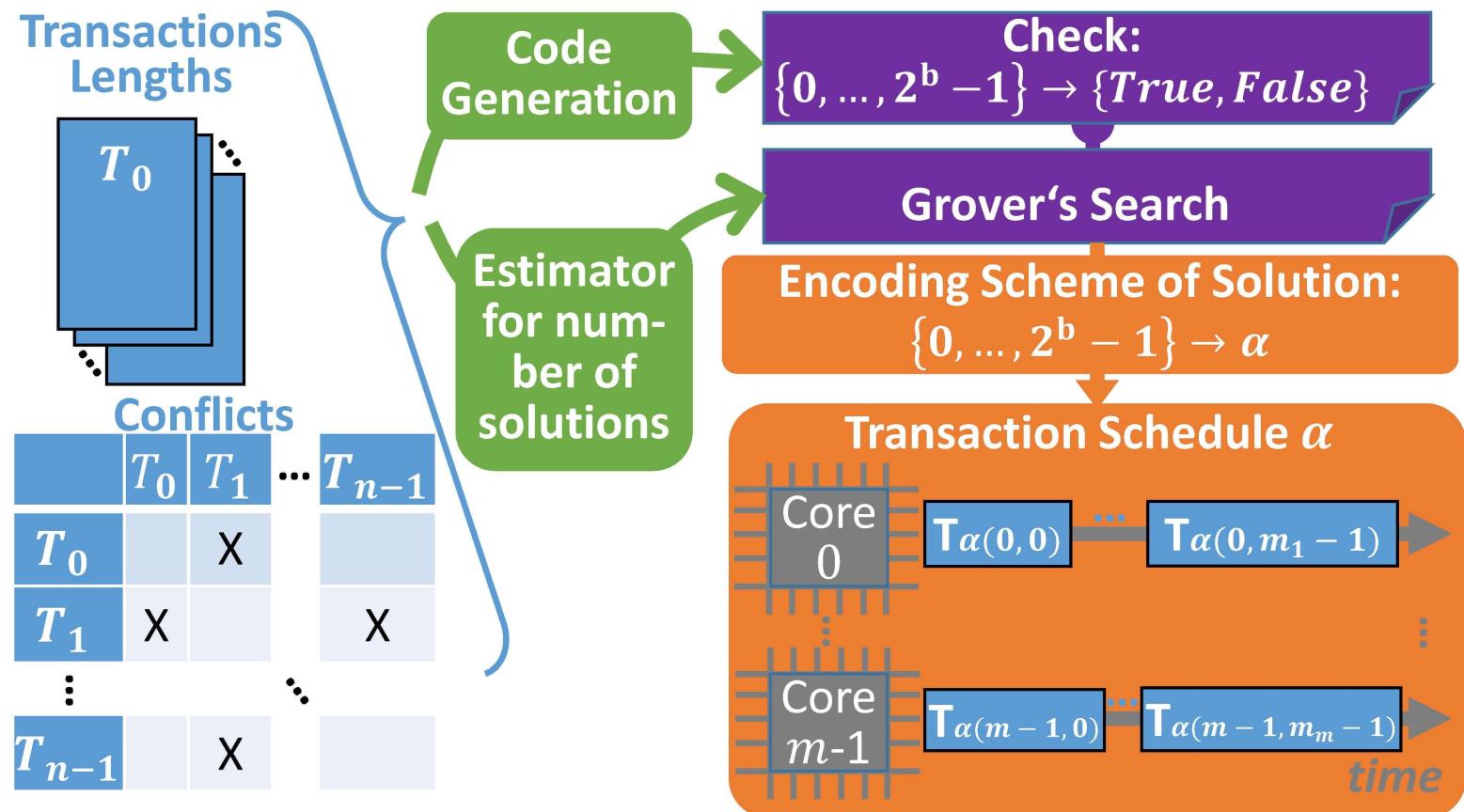
- Application-oriented:

- Estimate good initial threshold t , then continue original algorithm at 3. with threshold t
 - Example (see also next lecture unit):
 - Transactions T_1, \dots, T_p to be distributed to m cores of a CPU
 - Total runtime $l = \sum_{i=1}^p |T_i|$ of transactions
 - Threshold $t \geq \frac{l}{m}$ for max. runtime on all cores
 - In some applications, the number k of solutions of the oracle can be approximately estimated and used to speed up Grover's search

- Domain-oriented:

- I) If given domain $[start, end]$ is "small", then continue original algorithm at 3. with $t := end$
 - II) $m := \frac{end - start}{2}$
 - III) If Grover's search with oracle $f(x) < m$ finds a solution r , then continue at I) with domain $[start, r]$ else with $[m + 1, end]$

Overview of Optimizing Transaction Schedules via Quantum Computing



Encoding Scheme of Transaction Schedules

Algo determineSchedule

Input: $p:\{0, \dots, 2^b-1\}$

Output: $\{0, \dots, n-1\}^{m-1} \times \{0, \dots, n-1\}^{n-1}$

for(x **in** $1..m-1$)

$\mu_x = p \bmod n$

$p = p \bmod n$

$a = [0, \dots, n-1]$

for(i **in** $0..n-1$)

$j = p \bmod (n-i)$

$p = p \bmod (n-i)$

$\pi[i] = a[j]$

$a[j] = a[n-i-1]$

return ($\mu_1, \dots, \mu_{m-1}, \pi$)

Example (n=4, m=2):

29

$x = 1:$

$\mu_1 = 1$

$p = 7$

$a = [0, 1, 2, 3]$

$i = 0: \quad i = 1: \quad i = 2: \quad i = 3:$

$j = 3 \quad j = 1 \quad j = 0 \quad j = 0$

$p = 1 \quad p = 0 \quad p = 0 \quad p = 0$

$\pi[0] = 3 \quad \pi[1] = 1 \quad \pi[2] = 0 \quad \pi[3] = 2$

$a[3]=a[3] \quad a[1]=a[2] \quad a[0]=a[1] \quad a[0]=a[0]$

return (1,[3,1,0,2])

- $29 = 000111\ 01$ *binary* \equiv Core 0 [3, 1] $\mu_1 = 1$ [0, 2] Core 1,
some bits for permutation and some for separators
- $b = (m - 1) \cdot \lceil \log_2(n - 1) \rceil + \lceil \log_2(n! - 1) \rceil$



Generated Black Box Function

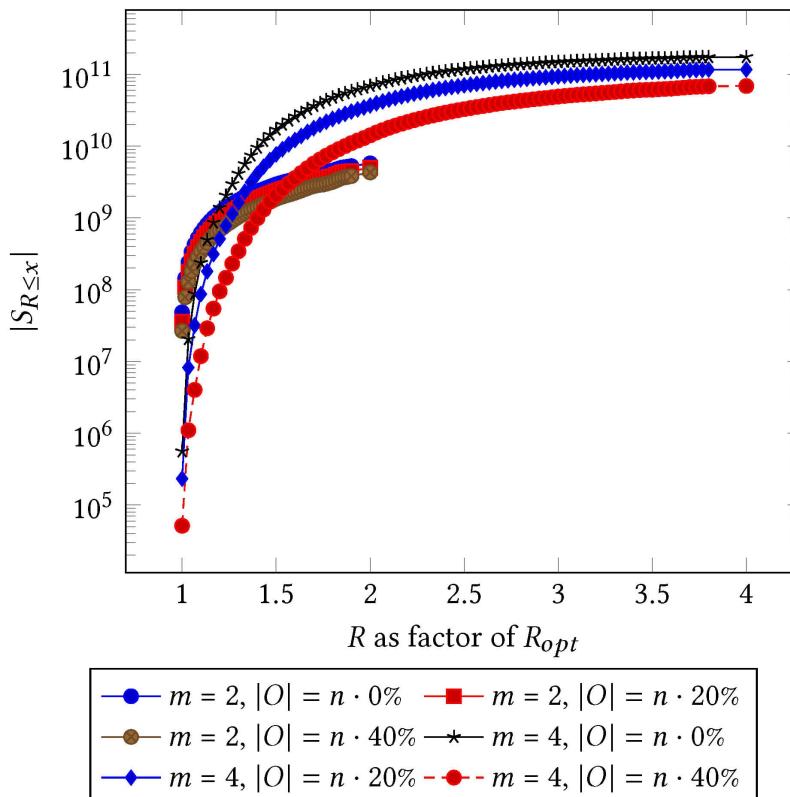
- Quantum computation: circuit of quantum logic gates
⇒ circuit is generated dependent on the concrete problem instance
 - Sketch of algo:
 1. Determine Separators and Permutation
 2. Check Validity of Separators
 3. $\forall i$: Determine lengths of i -th transaction in permutation
 4. Check: Which separator configuration? For current case:
 - 4a. determine total runtime of core and check if it's below given limit
 - 4a. determine start and end times of conflicting transactions
 5. Check: Do conflicting transactions overlap?
- $O(m + n)$
 $O(m)$
 $O(n \cdot \log_2 (n))$ with decision tree over transaction number
 $O(n)$
 $O(n)$
 $O(n \cdot \log_2 (\min(n, c)) + c)$ with decision tree over conflicting transactions (for $n \gg c$) or transaction numbers (for $c \gg n$)
 $O(c)$
 $\sum : O(n \cdot \log_2 (n) + c)$

Complexity Analysis

Approach	CPU	Quantum Computer	Quantum Annealing
Preprocessing	$O(1)$	$O(n^2 \cdot c)$	$O(m \cdot R^2 \cdot (c \cdot m + n^2))$
Execution	$O(\frac{(m+n-1)!}{(m-1)!} \cdot (n + c))$	$O(\sqrt{\frac{n! \cdot n^m}{k}} \cdot (n \cdot \log_2(n) + c))$	$O(1)$
Space	$O(n + m + c)$	$O((n + m) \cdot \log_2(n))$	$O(m \cdot R^2 \cdot (c \cdot m + n^2))$
Code	$O(1)$	$O(n^2 \cdot c)$	$O(m \cdot R^2 \cdot (c \cdot m + n^2))$

m : number of machines n : number of transactions c : number of conflicts R : max. runtime k : number of solutions

Number of Solutions



	$m = 2$	$m = 4$
N	8,589,934,592	2,199,023,255,552
k	48,384,000	559,872
k for $\leq 1.25 \cdot R_{opt}$	1,472,567,040	2,047,306,752



Summary & Conclusions

- Scheduling transactions as variant of jobshop problem with additionally considering blocking transactions
 - Hard combinatorial optimization problem \Rightarrow hardware acceleration
- Enumeration of all possible transaction schedules for finding an optimal one
 - Hardware acceleration via quantum annealing
 - Formulating transaction schedule problem as quadratic unconstrained binary optimization (QUBO) problem
 - Constant execution time in contrast to simulated annealing on classical computers
 - Preprocessing time increasing with larger problem sizes
 - Grover's search: \approx quadratic speedup on Universal Quantum Computers
 - Estimation of number of solutions for a further speedup
 - Estimation of speedup for suboptimal solutions being a guaranteed factor away from optimal solution
 - Code Generator available at [↗
https://github.com/luposdate/OptimizingTransactionSchedulesWithSilq](https://github.com/luposdate/OptimizingTransactionSchedulesWithSilq)