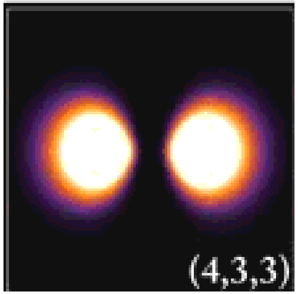
EAI ICISML 2022

Quantum Data Management and Quantum Machine Learning for Data Management: State-of-the-Art and Open Challenges

Professor Dr. rer. nat. habil. Sven Groppe
https://www.ifis.uni-luebeck.de/index.php?id=groppe

Quantum Mechanics

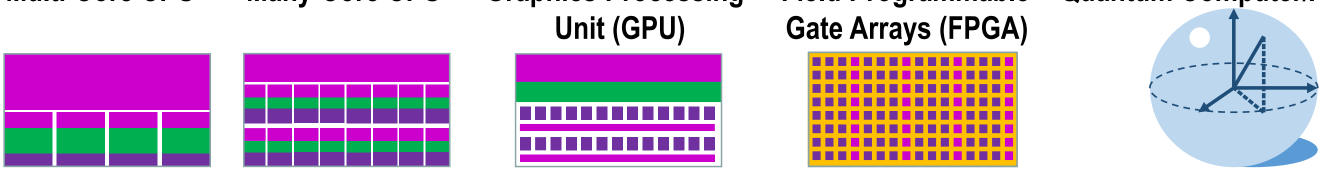
- Very small particles and light behave differently from objects in normal life
- Mechanics of light and matter at the atomic and subatomic scale are described by quantum theory
  - forming the underlying principles of chemistry and most of physics
- Quantum theory has brought us the information age 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.

Architectures of Emergent Hardware

	Multi-Core CPU	Many-Core CPU	Graphics Processing Unit (GPU)	Field Programmable Gate Arrays (FPGA)	Quantum Computer/Annealer

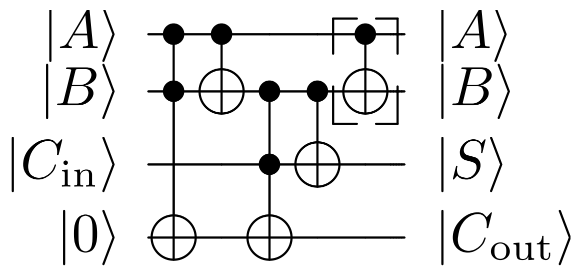


- **Cores**: ~10 → ~100 → ~1000 → ~100 000 | ~100/~5000 qubits
- **Core Complexity**: Complex (optimized for single thread performance) ← Simple | Complex
- **Computational Model**: MIMD + SIMD (Multi-Core CPU, Many-Core CPU) | SIMD (GPU) | Data-Flow (FPGA) | Universal/Adiabatic Quantum computing
- **Parallelism**: Thread and Data Parallel (Multi-Core CPU, Many-Core CPU) | Data Parallel (GPU) | Arbitrary (FPGA) | Quantum Logic Gates/ Fluctuation (glob. Min.)
- **Memory Model**: Shared (Multi-Core CPU) | Distributed (Many-Core CPU, GPU, FPGA) | Quantum Superposition
- **Power**: 150 W | 200 W | 250 W | 50 W | 25 KW
- **Database Op.**:
  - Query Optimization (Enumeration of Plans), Concurrency Control (all)
  - Efficient Processing Of (all)
  - Query Processing (Multi-Core CPU to FPGA)

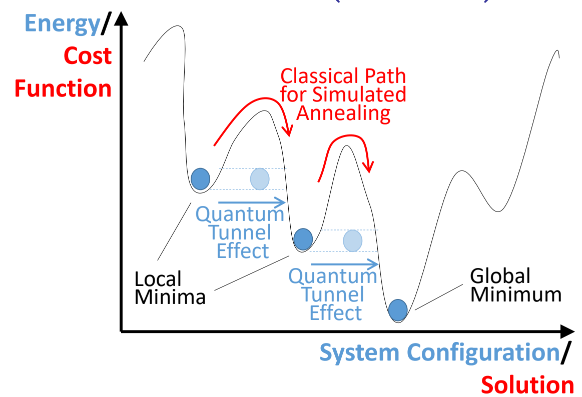
Legend: Computational Unit · Execution Controller · Interconnection Network · On-Chip Memory

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



Quantum Circuit (Full Adder)



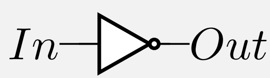
Simulated versus Quantum Annealing

Classical versus Quantum Computing

	Classical	Quantum
Information Unit	**Bi**nary Digi**t** (Bit): • basis of a 2-level system • can be in state 0 or 1	**Qu**antum **Bit** (Qubit): • 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: • performs on 1 or more bits to produce a single bit output	Quantum Logic Gate: • performs on 1 or more qubits to change the quantum state of a single qubit
Example Operation	NOT/Inverter:	NOT/Pauli$_x$-Gate:

Example Operation — Classical:

NOT/Inverter:

Digital Circuit: 

In	Out
0	1
1	0

Example Operation — Quantum:

NOT/Pauli$_x$-Gate:

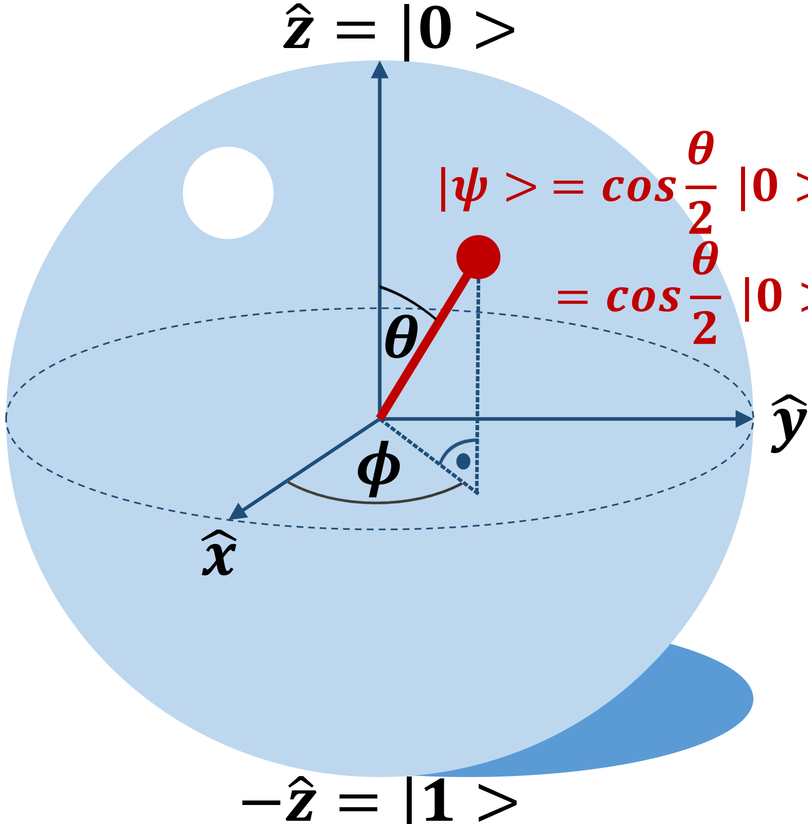
Quantum Circuit: $|In\rangle \;-\!\oplus\!-\; |Out\rangle$

Alternatively: $|In\rangle \;-\boxed{X}-\; |Out\rangle$

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

Representation of a Qubit in Bloch-Sphere

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

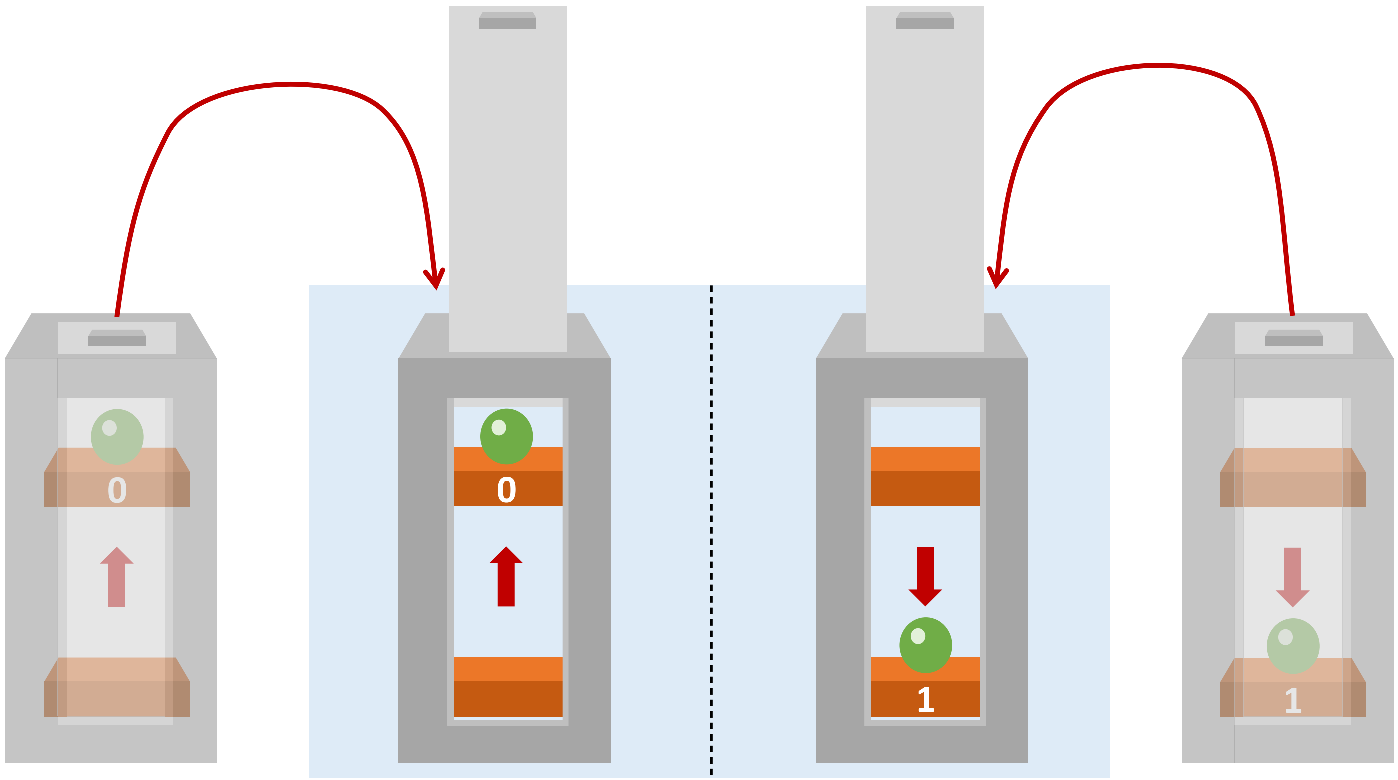


$$|\psi> = cos\frac{\theta}{2}\,|0> + e^{i\phi} sin\frac{\theta}{2}\,|1>$$

$$= cos\frac{\theta}{2}\,|0> + (cos\,\phi + i\,sin\,\phi) sin\frac{\theta}{2}\,|1>$$

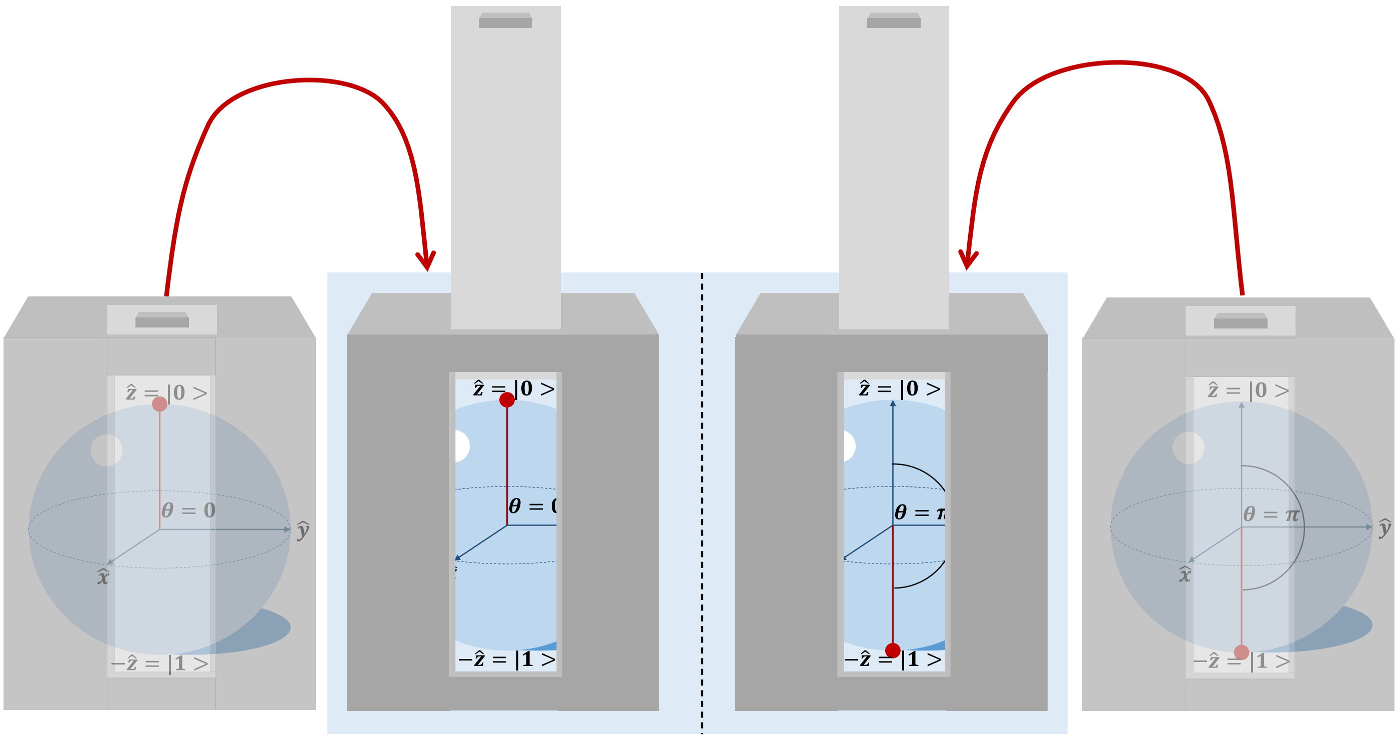
Classical Measurement/Observation

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



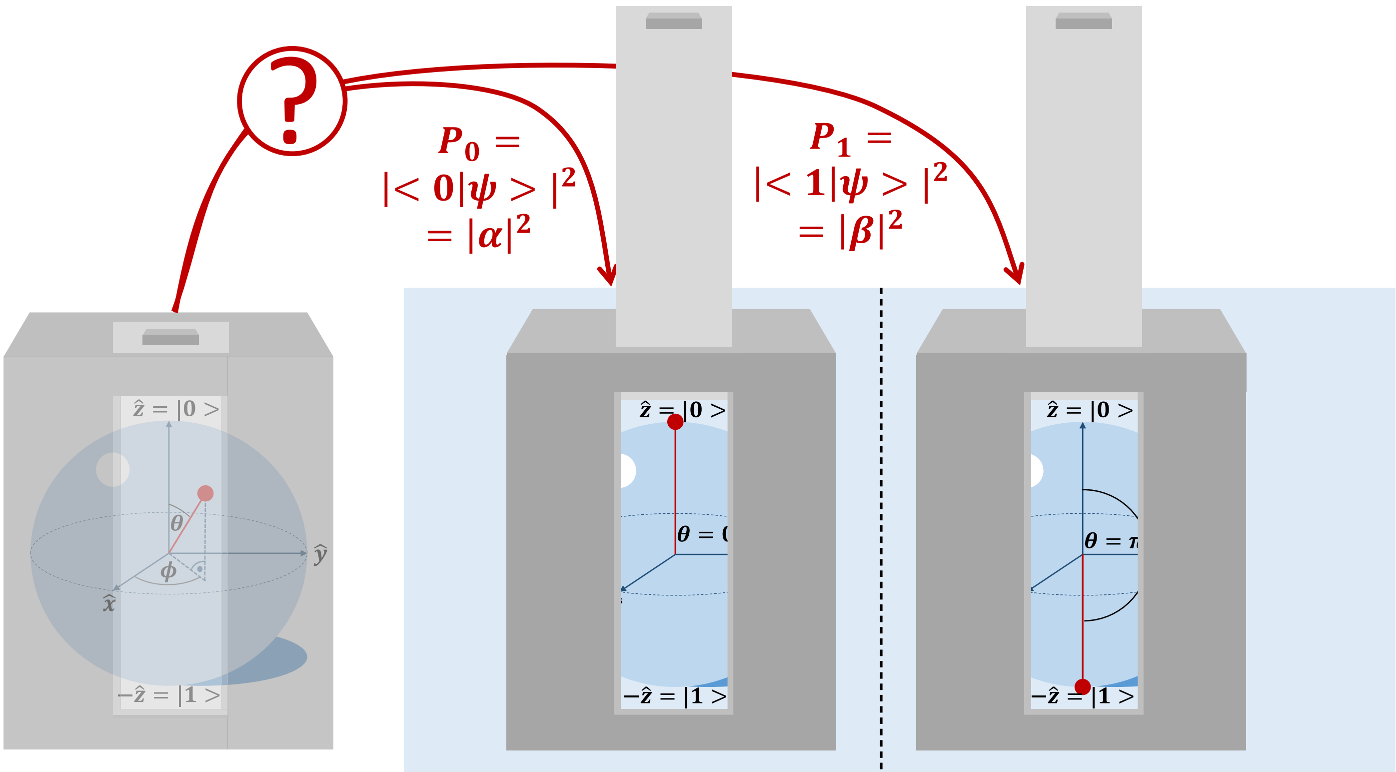
Quantum Measurement/Observation 1/2

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



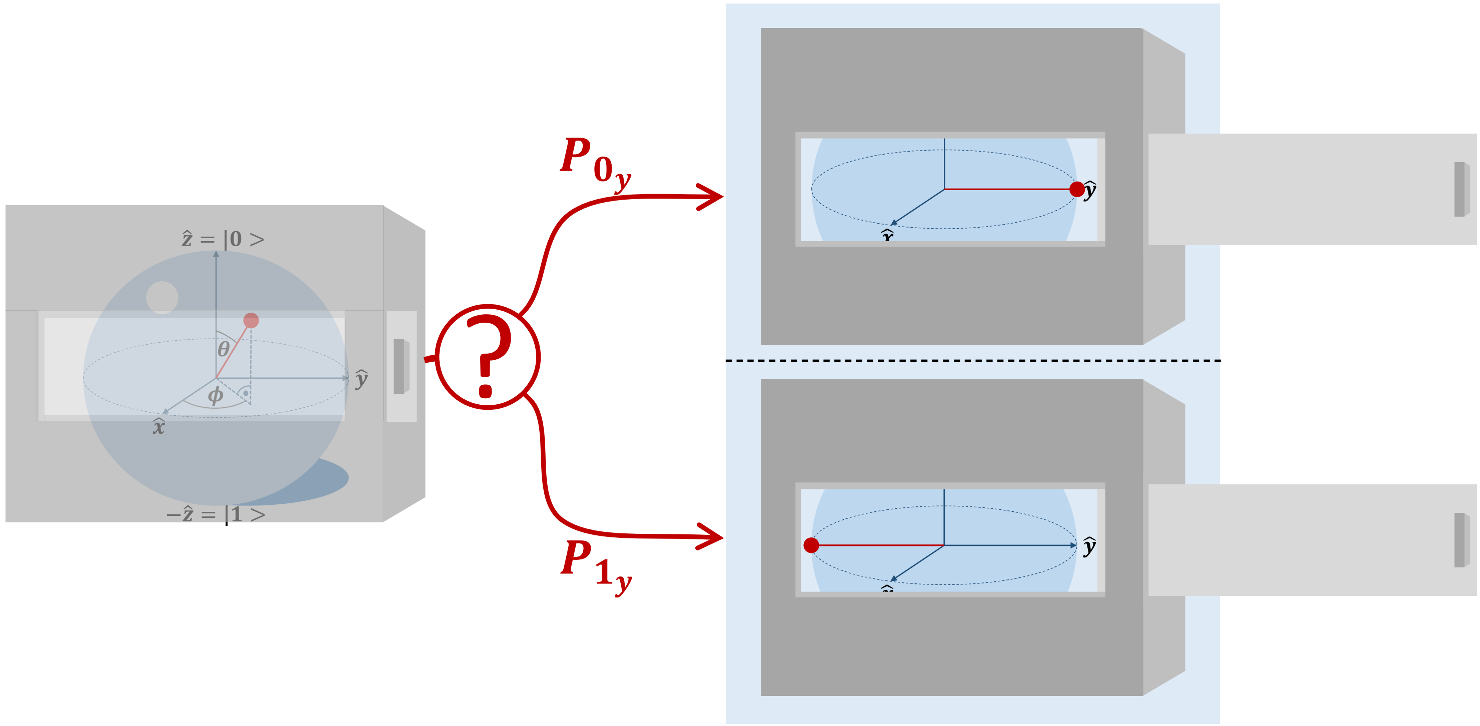
Quantum Measurement/Observation 2/2

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



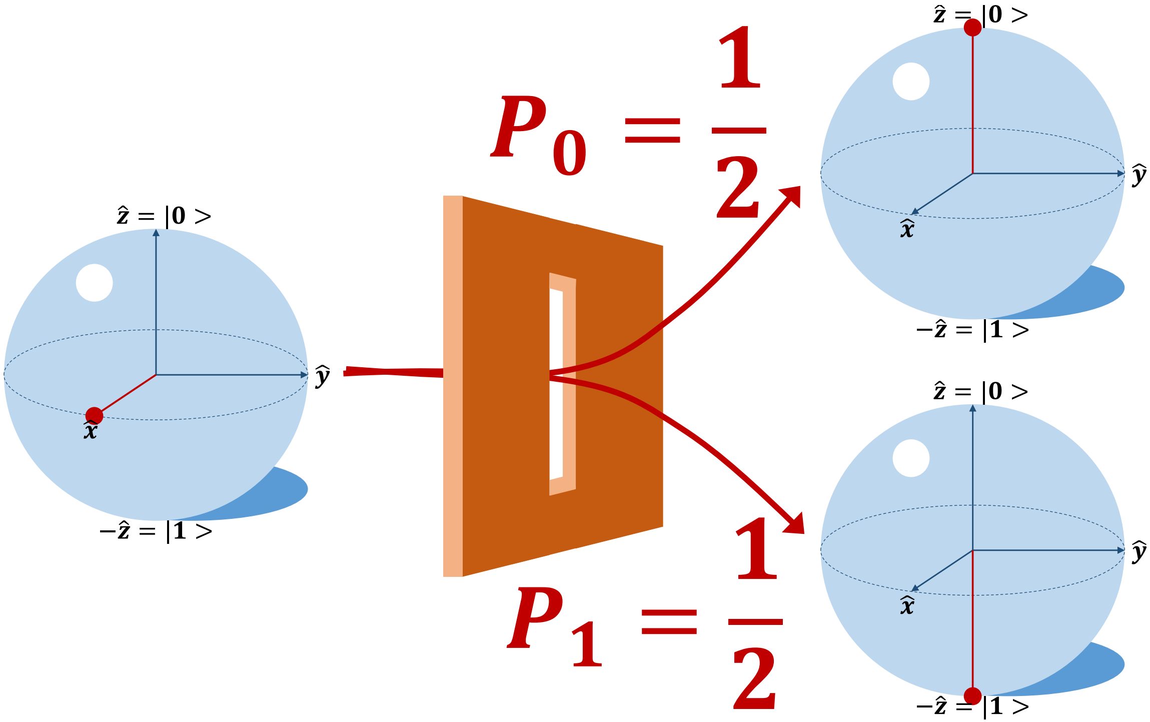
Measurement/Observation along other axis (here y-axis)

- However, observation typically according to z-axis



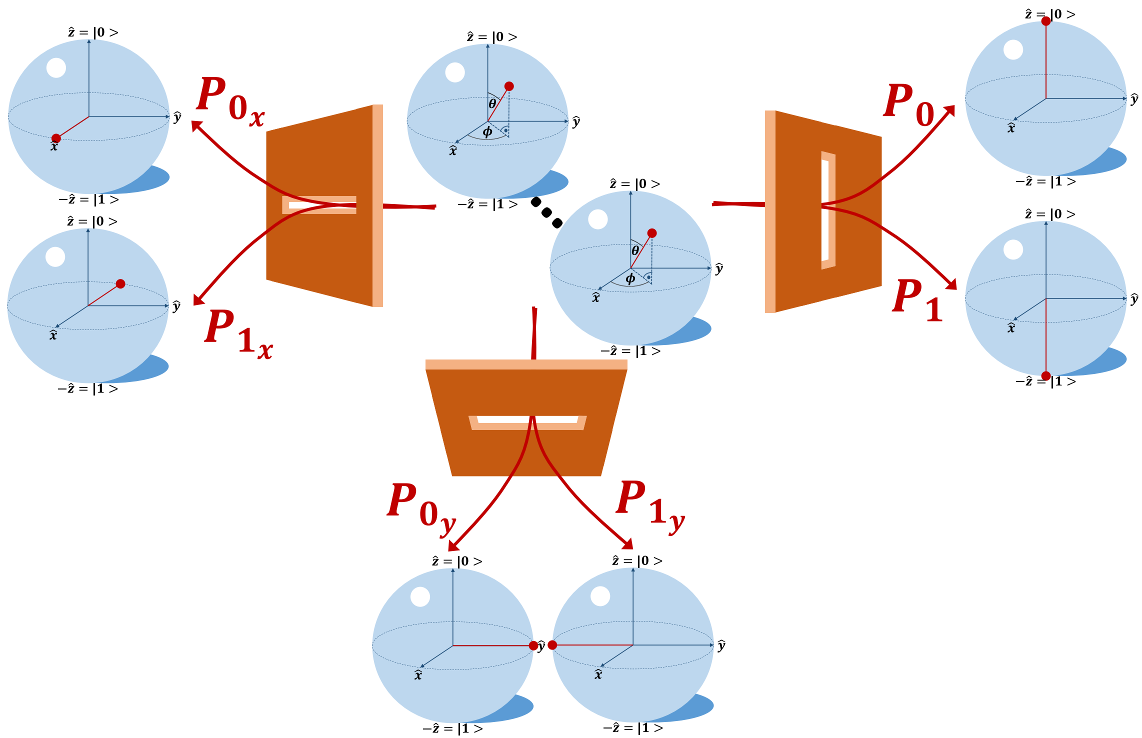
Generator for True Random Numbers

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



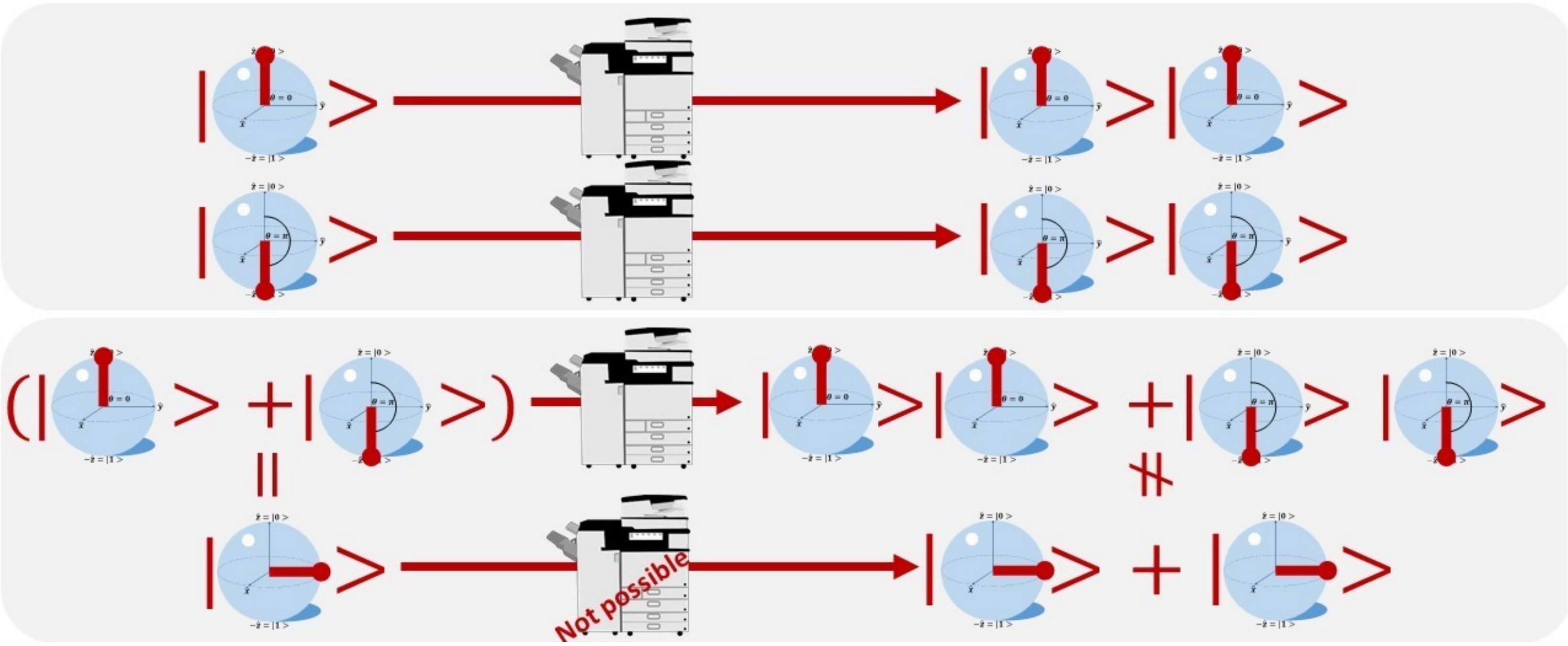
Determining the states (θ, ϕ) of identical prepared Qubits

- 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



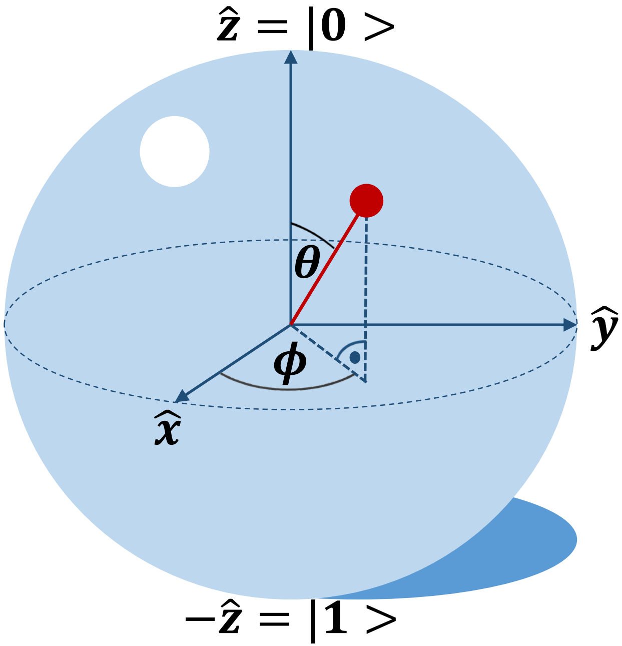
No-Cloning-Theorem of 1 Qubit

- 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	$NOT = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	
Quantum Circuit	$\|In\rangle$ $\oplus$ $\|Out\rangle$ Alternatively: $\|In\rangle$ $\boxed{X}$ $\|Out\rangle$	
Table of in- & outputs:	In	Out
	$\|0\rangle$	$\|1\rangle$
	$\|1\rangle$	$\|0\rangle$
	$\frac{1}{\sqrt{2}}(\|0\rangle + \|1\rangle)$	$\frac{1}{\sqrt{2}}(\|0\rangle + \|1\rangle)$
	$\frac{3\cdot i}{5}\|0\rangle + \frac{4}{5}\|1\rangle$	$\frac{4}{5}\|0\rangle + \frac{3\cdot i}{5}\|1\rangle$

Controlled NOT (CNOT)-Gate

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

Quantum Circuit

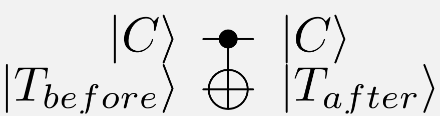


Table of in- & outputs

Inputs		Output
Control C	Target T_{before}	Target T_{after}
$\|0\rangle$	$\|0\rangle$	$\|0\rangle$
$\|0\rangle$	$\|1\rangle$	$\|1\rangle$
$\|1\rangle$	$\|0\rangle$	$\|1\rangle$
$\|1\rangle$	$\|1\rangle$	$\|0\rangle$

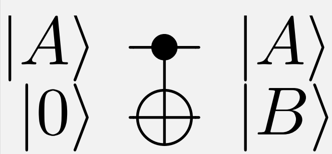
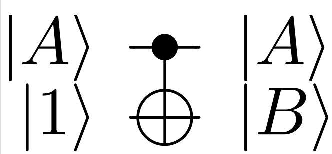
$R \cdot \frac{1}{\sqrt{2}}(|01\rangle + |11\rangle) = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$

Rotation Matrix R

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

- 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	B
		$\lvert 0\rangle$	$\lvert 0\rangle$
		$\lvert 1\rangle$	$\lvert 1\rangle$
		$\frac{1}{\sqrt{2}}(\lvert 00\rangle + \lvert 11\rangle)$	
Anti-Correlated		A	B
		$\lvert 0\rangle$	$\lvert 1\rangle$
		$\lvert 1\rangle$	$\lvert 0\rangle$
		$\frac{1}{\sqrt{2}}(\lvert 01\rangle + \lvert 10\rangle)$	

- Entangled qubits at different locations are still entangled (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

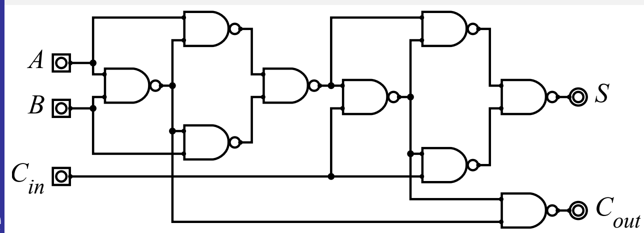
Yin et al., "Satellite-based entanglement distribution over 1200 kilometers". Quantum Optics. 356: 1140–1144, 2017

Digital versus Quantum Circuits

	Digital Circuit	Quantum Circuit
Building Blocks	Logic Gates	Quantum Logic Gates

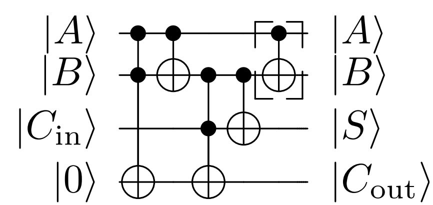
Full Adder Example

Digital Circuit:



consists of NAND gates

Quantum Circuit:



consists of Toffoli and CNOT gates[1]

In- and Output Full Adder

Digital Circuit:

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

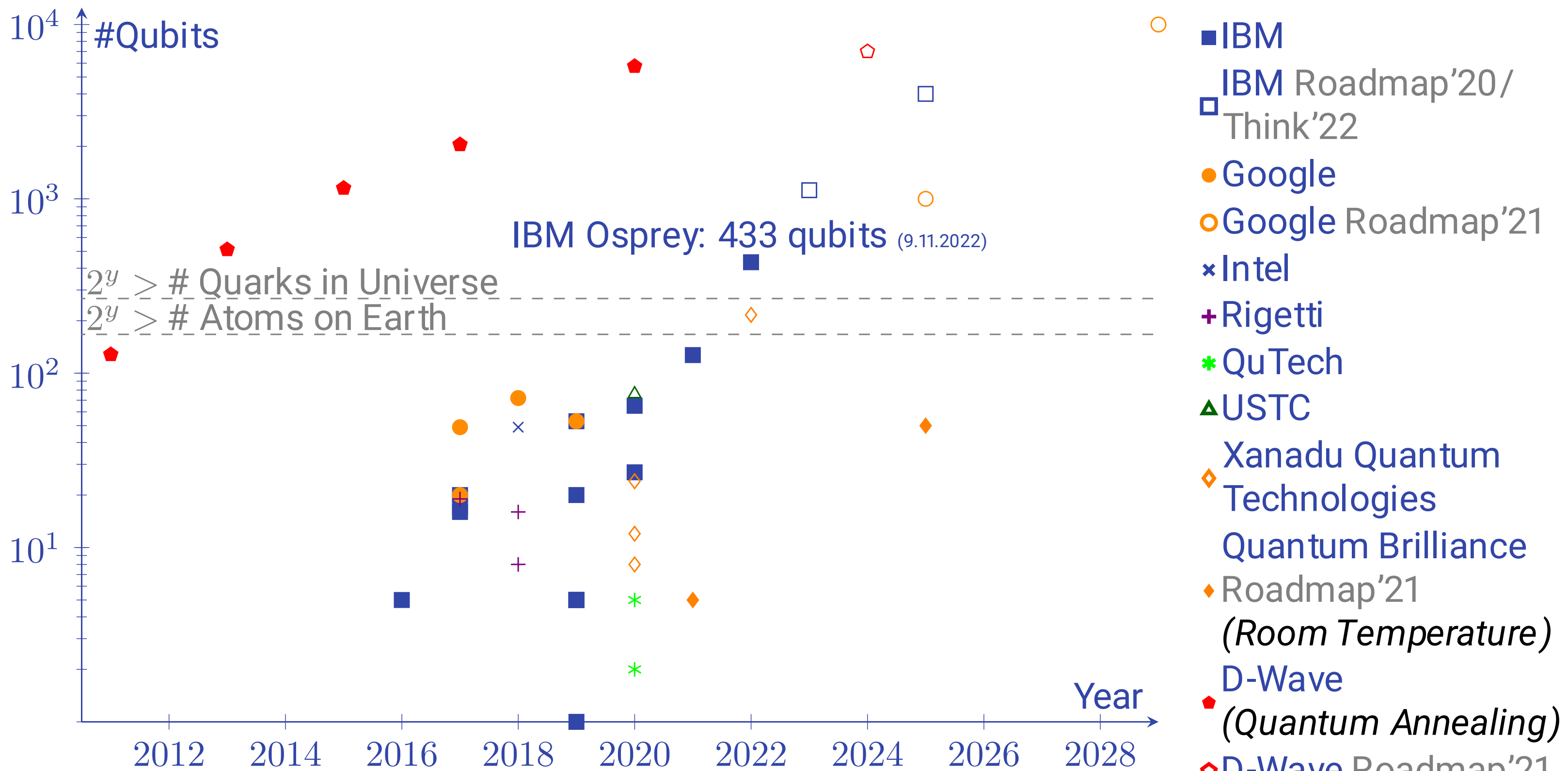
Quantum Circuit:

$|0\rangle$ **and** $|1\rangle$ **as input:** Output is $|0\rangle$ and $|1\rangle$ analogous to digital circuit.
Superpositions as input: Superpositions as output with corresponding probabilities for basic quantum states, e.g.:

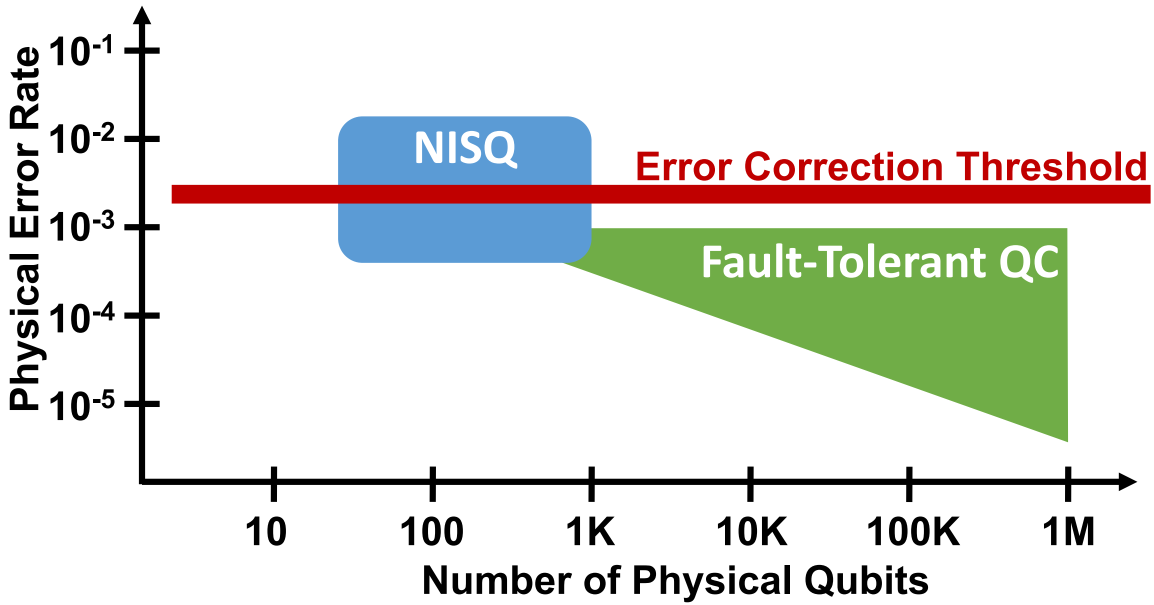
A	B	C_{in}	C_{out}	S
$\frac{1}{\sqrt{2}}(\|0\rangle + \|1\rangle)$	$\|0\rangle$	$\|0\rangle$	**$ABC_{out}S$:** $\frac{1}{\sqrt{2}}(\|0000\rangle + \|1001\rangle)$	
$\frac{1}{\sqrt{2}}(\|0\rangle + \|1\rangle)$	$\|A\rangle$ $\|1\rangle$ $\|A\rangle$ $\|B\rangle$	$\|0\rangle$	$\|0\rangle$	$\|1\rangle$

[1] The 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

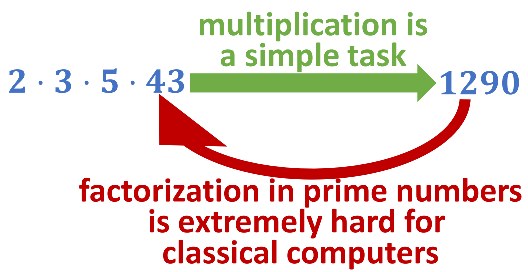
Potential of Quantum Algorithms

- **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 Algorithm[1]



- factoring integers in polynomial time
  - Depth of quantum circuit[2] 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 $\longrightarrow$ 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) $f : \{0, ..., 2^b - 1\} \mapsto \{true, false\}$
- Grover's search algorithm finds one $x \in \{0, ..., 2^b - 1\}$, such that $f(x) = true$
  - 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

Quantum Data Management

- What can data management do for quantum computing?
  - vision of a database management system
    - offering easy access to quantum computing applications
    - by integrating high-level functionalities for quantum computing applications
- What can quantum computing do for data management?
  - Speeding up some of the tasks of data management with high runtime complexity

Quantum Data Management - **DM4QC**

- What can data management do for quantum computing?
  - vision of a database management system
    - offering easy access to quantum computing applications
    - by integrating high-level functionalities for quantum computing applications
      - extension of query language to call quantum computing subroutines
      - easy storing and accessing the input and output of quantum computing applications in DBMS
      - automatically choosing of the needed and available quantum resources as well as apply hybrid algorithms
        - tailoring in a way that missing resources and support of qubits and circuit depths in quantum computers are overcome by taking over more computations by classical algorithms or simulations of quantum computing on classical hardware

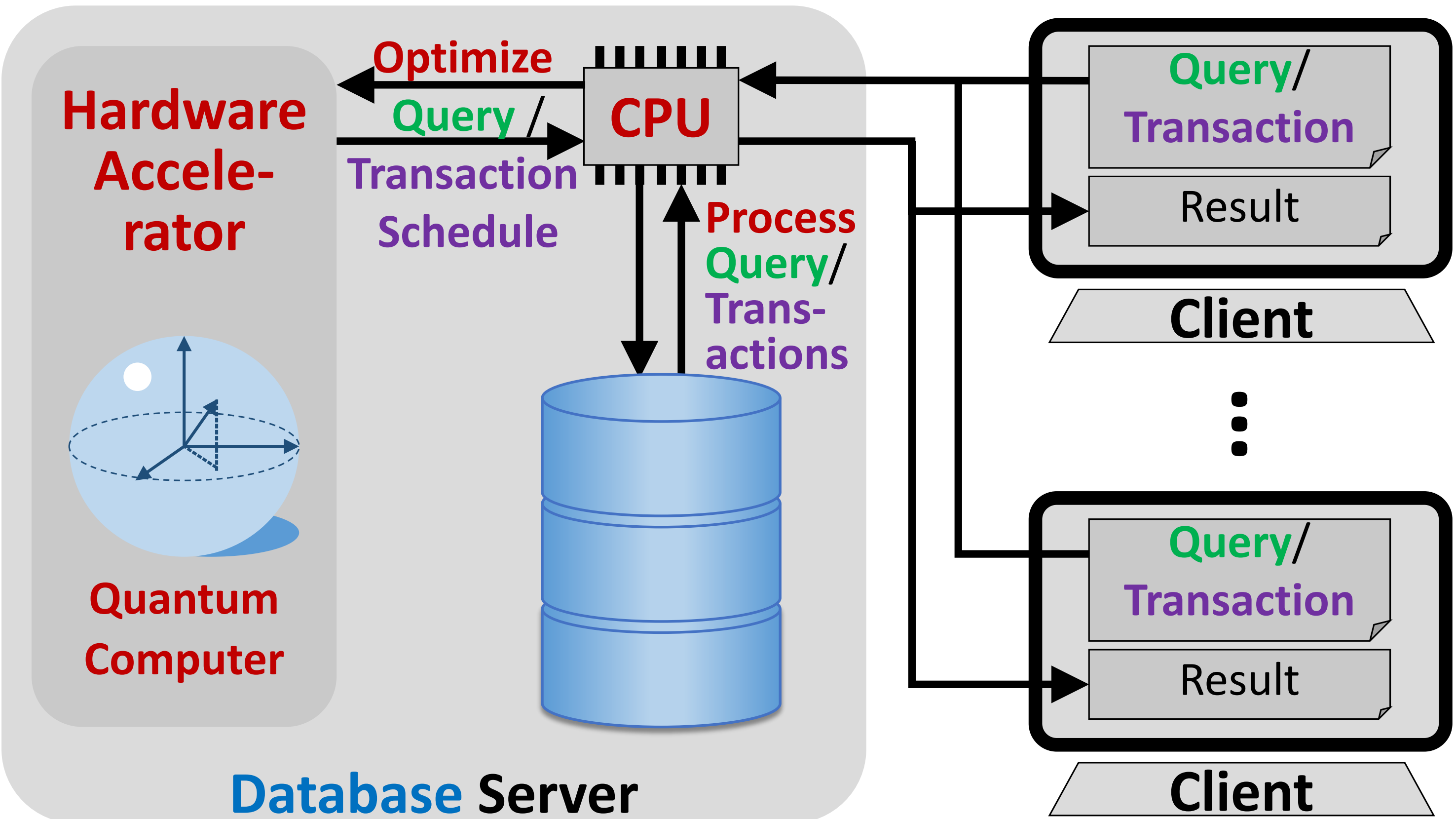
Quantum Data Management - **DM4QML**

- What can data management do for quantum computing and especially **quantum machine learning (QML)**?
  - trend for DBMS to offer machine learning functionalities
  - machine learning tasks pose additional requirements for
    - scalable training, where often large-scale data sets are processed in comparison to the (often simple and efficient to perform) application/prediction phase
    - storing models and their parameters after training for their use in the application/prediction phase
    - language constructs for expressing complex machine learning tasks in a simple way while combining these language constructs with query languages
    - automatic choosing and optimizing models and machine learning tasks
  - Open challenges: integration of QML into DBMS and adapting the requirements and solutions to QML and available quantum computers with an additional focus on hybrid algorithms
    - Hybrid algorithms combine a quantum algorithm with a classical algorithm to flexibly adapt the available system configuration of classical and quantum hardware

Quantum Data Management - **QC4DM**

- What can quantum computing do for data management?
  - Speeding up some of the tasks of data management with high runtime complexity
    - Examples: **Optimizing queries and transaction schedules**

Using Hardware Accelerator for optimizing Queries / Transaction Schedules



Approaches for Query/Transaction Schedule Optimization

Query Optimization:

$\bowtie_{i=1}^{n} R_i$?

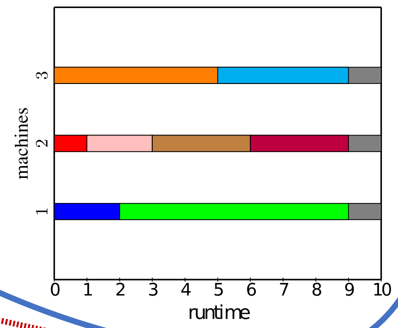
$(R_1 \bowtie R_2) \cdots \bowtie R_n$

$\vdots$

$(R_1 \bowtie R_n) \bowtie (\cdots)$

Transaction Schedule Optimization:

$\{T_1, \ldots, T_m\}$?



Open Source Relational Database Management System (RDBMS),
e.g. PostgreSQL, MySQL

Dynamic Programming
Random Walk
Linear Program-ming
Machine Learning
...
Simulated Annealing
Genetic Algorithm

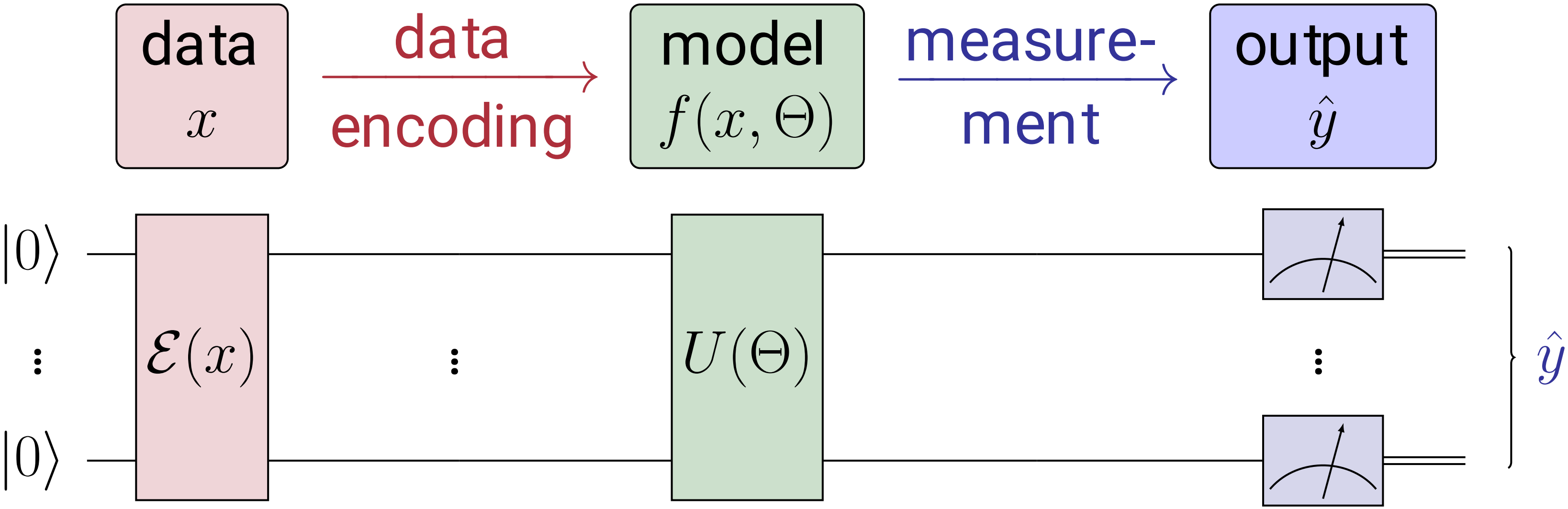
Algorithms (used e.g. in Query Optimization) and their Quantum Counterparts

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]
[TK17]	Mixed Integer Linear [BGG+71] Programming [D02]	[HHL09] [A12] [CKS17] [SSO19] [AL22] [AL22]

This list is not complete...

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

Quantum Machine Learning - Data encoding and Quantum Model

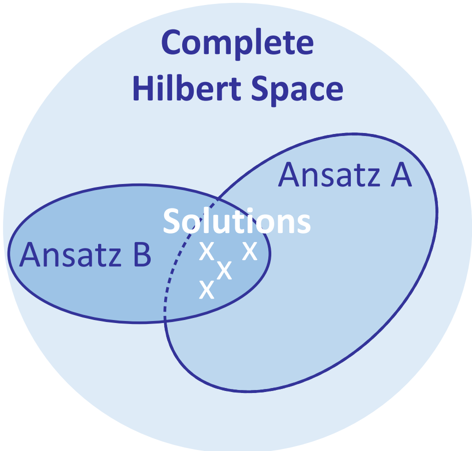


Relations to Join

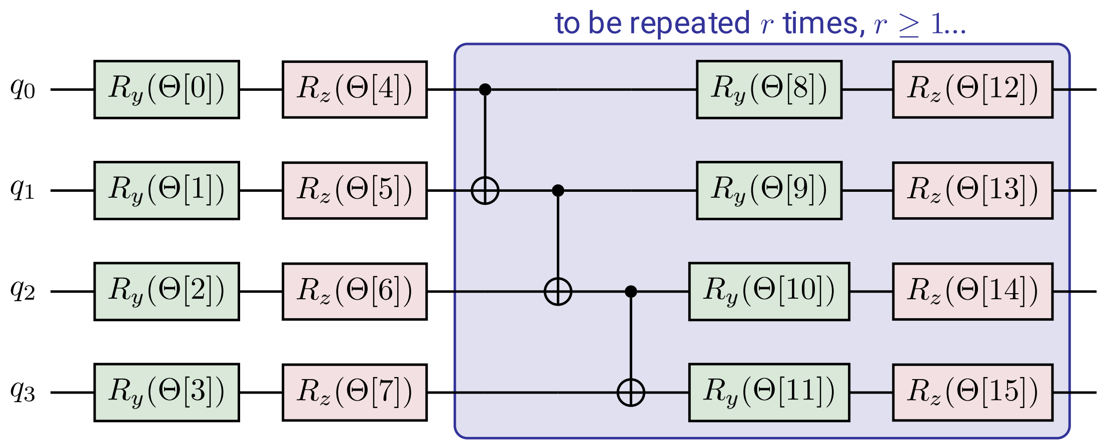
Join Order

How can we build a quantum model?

Solution Space



Hardware-efficient Ansatz



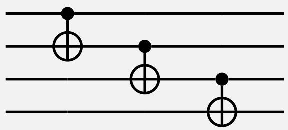
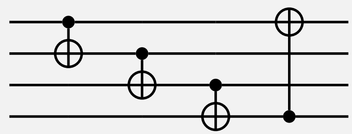
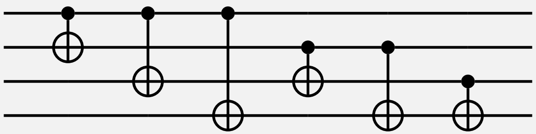
Problem-specific ansatz

Use knowledge about problem to choose ansatz!

Example: Does the wave function of the quantum states only have real amplitudes?

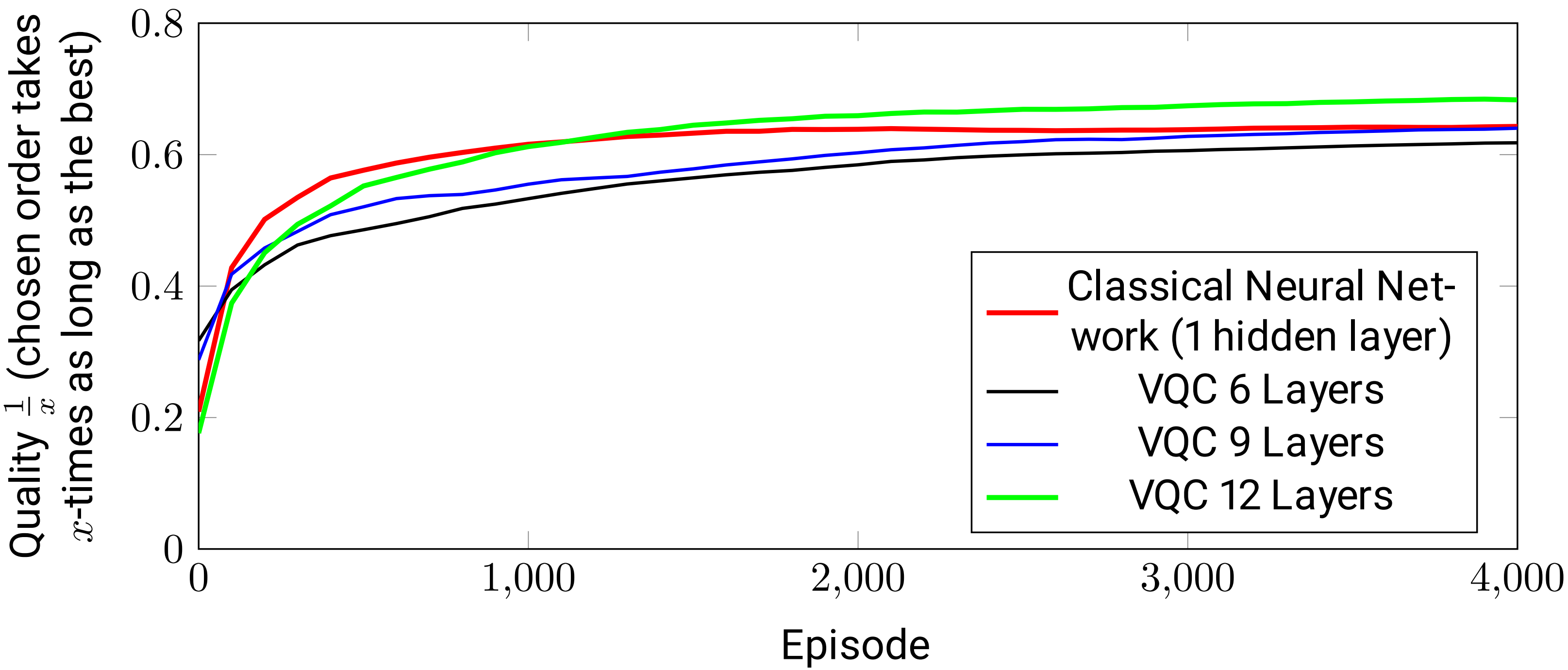
Many different quantum circuits for machine learning models, see [B+'19]

Variants

Entanglement Layer	Linear	Circular	Fully Connected
Circuit			
#Gates for n qubits	$n - 1$	n	$\binom{n}{2}$

- Different combinations of x ,y and z rotation gates in the rotation layer
- In variants rotation layer and entanglement layer are combined into one by using controlled rotation gates

QML 4 Join Order Optimization



- Variational quantum circuits (VQCs) beat classical neural networks for join order optimization
- Contributions in other domains report learning on fewer data points/faster learning [C+22], which is not observed here

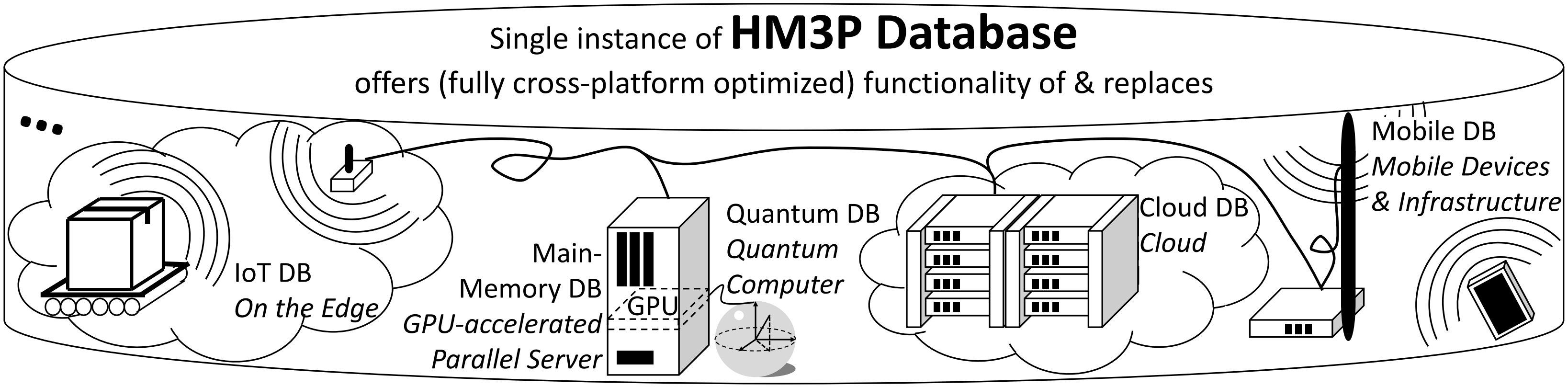
Open Challenges for QC for Databases

- Replacing basic algorithms with their QC counterparts in query optimizations 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 and transaction schedule optimizations benefit from quantum computers?
  (In short: those based on mathematical optimization problems, but also other...?)

QC4DB: Accelerating Relational Database Management Systems via Quantum Computing

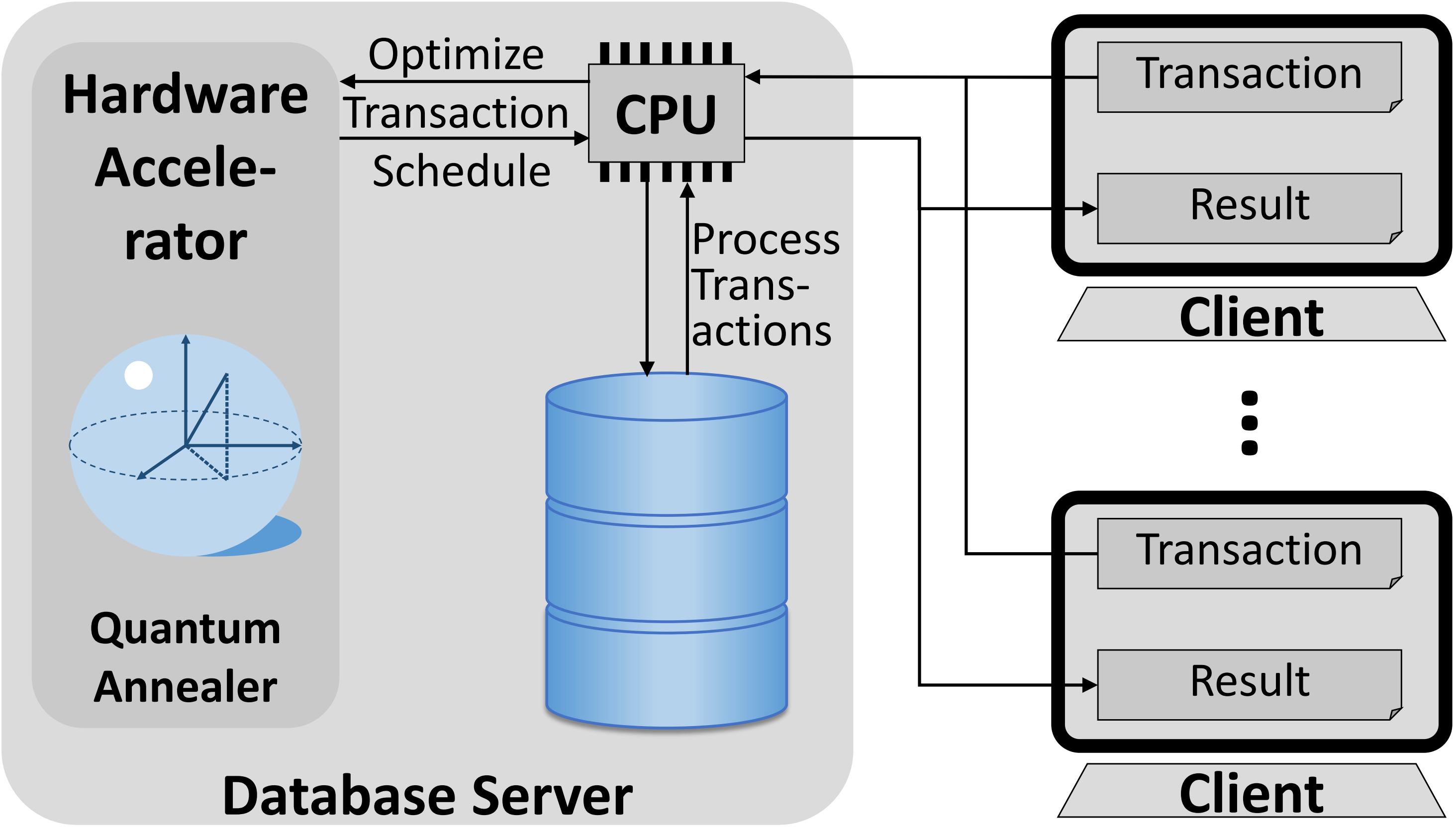
Name:	QC4DB: Accelerating Relational Database Management Systems via Quantum Computing	
Proj. Web:	**Project Website@Quantentechnologien**	
Funded by:	BMBF, Fördermaßnahme Anwendungsnetzwerk für das Quantencomputing	
Duration:	3 years	
Volume:	1.8M Euros	
Topics:	Optimizing an open source relational database management system • Queries • Transaction Schedules	
Partners:	UNIVERSITÄT ZU LÜBECK INSTITUT FÜR INFORMATIONSSYSTEME (Coord.)	QUANTUM BRILLIANCE
Expertises:	Hardware-Acceleration of Databases	Room Temperature Diamond Quantum Accelerators/qbOS
Website:	**https://www.ifis.uni-luebeck.de/~groppe/**	**https://quantumbrilliance.com/**

Hybrid Multi-Model Multi-Platform (HM3P) Database

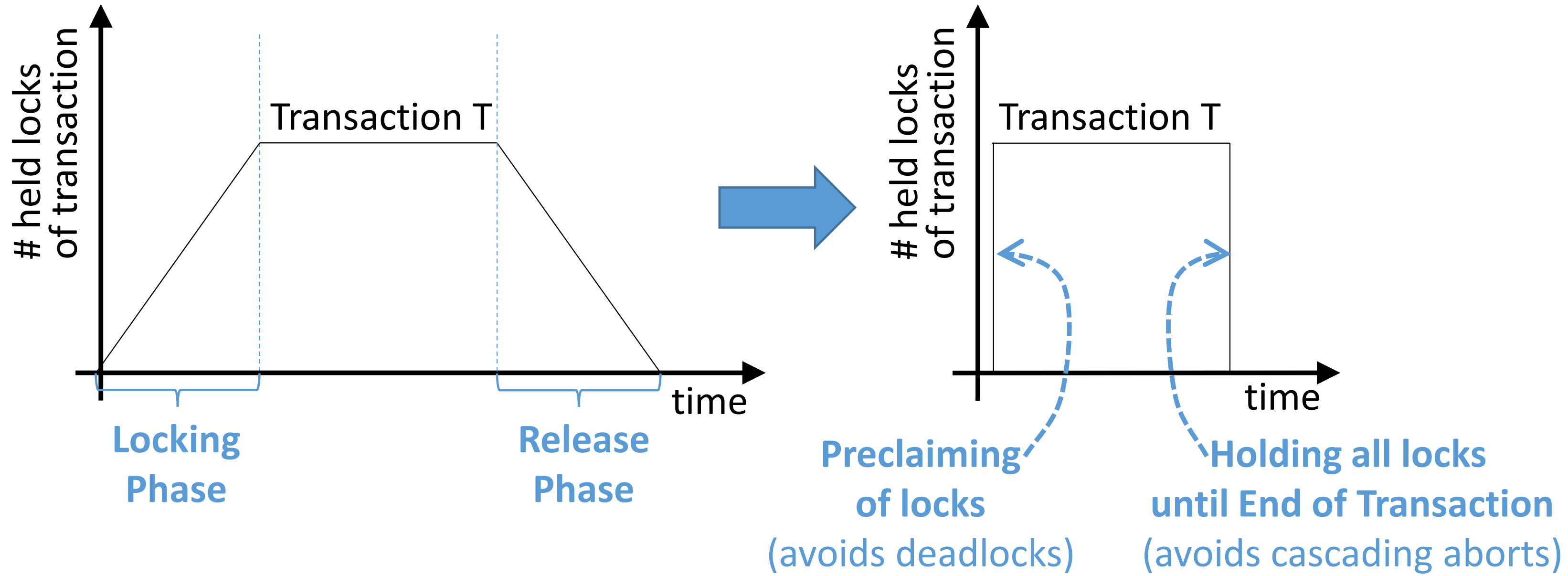


- full and uniform data integration at database level
- performance: fully optimized across different data models
- transparent fault-tolerance
- SQL standards: relational ('87), XML ('03), temporal ('11), JSON ('16), Multi-dimensional Arrays ('19), schemaless ('19), streams ('20?), property graphs ('21?)
- **features of different types of databases running on different platforms can be used**

Using Hardware Accelerator for optimizing Transaction Schedules



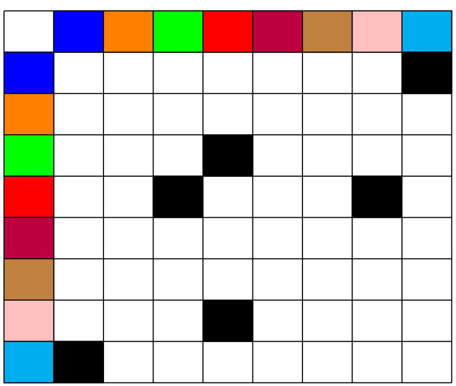
2 Phase Locking (2PL) versus Strict Conservative 2PL



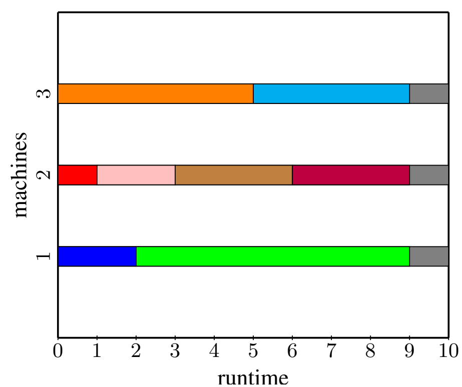
- required locks to be determined by
  - static analysis of transaction, or if static analysis is not possible:
  - an additional phase at runtime before transaction processing
    - A. Thomson et al., "Calvin: Fast distributed transactions for partitioned database systems", SIGMOD 2012.

Optimizing Transaction Schedules

- Variant of job shop schedule problem (JSSP):
  - Multi-Core CPU
    - Process whole job (here transaction) on core X
  - Schedule: ∀ 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

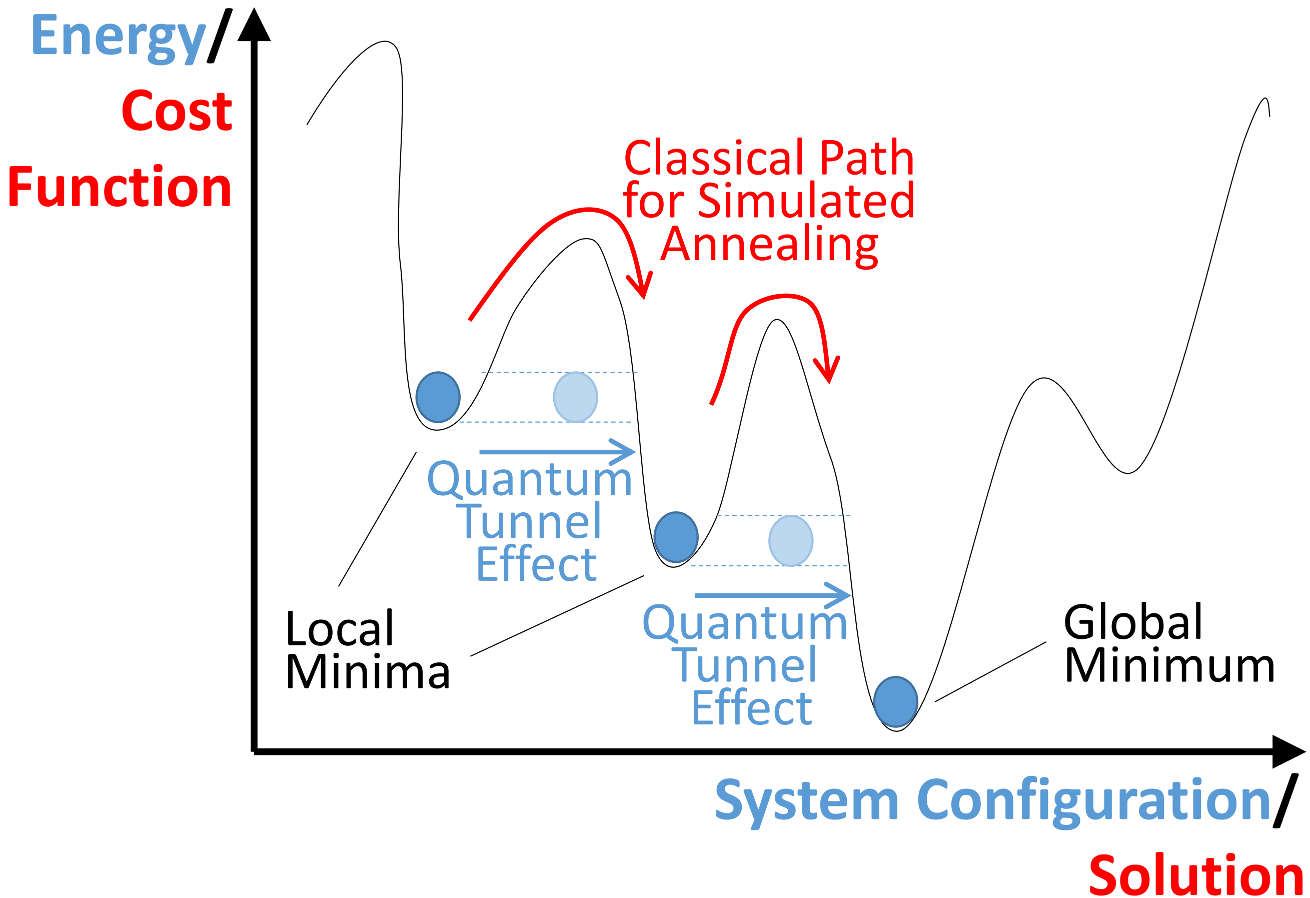


Transaction schedule

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

* M.R. Garey, D.S. Johnson and R. Sethi, The Complexity of Flowshop and Jobshop Scheduling, 1(2):117-129, 1976

Quantum versus Simulated Annealing

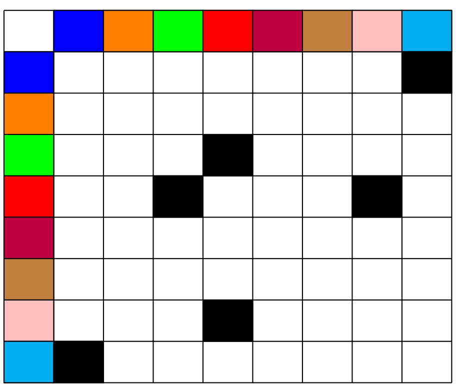


Optimizing Transaction Schedules via Quantum Annealing

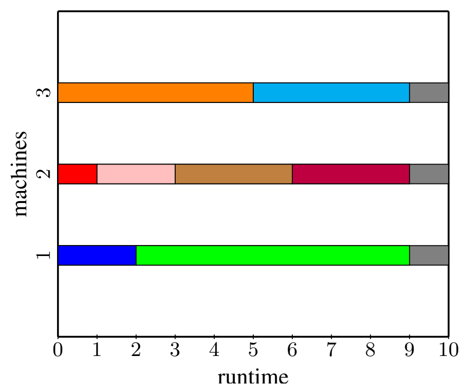
- Transaction Model
  - T: set of transactions with |T| = n
  - M: set of machines with |M| = k
  - $O \subseteq T \times T$: set of blocking transactions
  - l_i: length of transaction i
  - R: maximum execution time
  - upper bound $r_i = R - l_i$ for start time of transaction i
- Example
  - T = $\{t_1, t_2, t_3\}$, n=3
  - M = $\{m_1, m_2\}$, k=2
  - $O = \{(t_2, t_3)\}$
  - $l_1 = 2, l_2 = 1, l_3 = 1$
  - R = 2
  - $r_1 = 0, r_2 = 1, r_3 = 1$
- Quadratic unconstrained binary optimization (QUBO) problems (solving is NP-hard)
  - A QUBO-problem is defined by N weighted binary variables $X_1, ..., X_N \in 0, 1$, 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:



Black: Blocking transactions



Transaction schedule

- 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

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

Example:

$$R = 2$$
$$T = \{t_1, t_2, t_3\} \text{ with } |T| = n = 3$$
$$M = \{m_1, m_2\} \text{ with } |M| = k = 2$$
$$O = \{(t_2, t_3)\}$$
$$l_1 = 2,\ l_2 = 1,\ l_3 = 1$$
$$r_1 = 0,\ r_2 = 1,\ r_3 = 1$$

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

Optimizing Transaction Schedules via Quantum Annealing

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

$$B = \underbrace{\sum_{j=1}^{k}}_{\text{machines}} \overbrace{\sum_{i_1=1}^{n-1}}^{\text{transactions without } t_n} \underbrace{\sum_{s_1=0}^{r_{i_1}}}_{\text{start times}} \overbrace{\sum_{i_2=i_1+1}^{n}}^{\text{remaining transactions}} \underbrace{\sum_{s_2=q}^{p}}_{\text{invalid start times}} X_{i_1,j,s_1} X_{i_2,j,s_2} \text{ for } q = \max\{0, s_1 - l_{i_2} + 1\}, p = \min\{s_1 + l_{i_1}, r_{i_2}\}$$

Example: $R = 2$
$T = \{t_1, t_2, t_3\}$ with $|T| = n = 3$
$M = \{m_1, m_2\}$ with $|M| = k = 2$
$O = \{(t_2, t_3)\}$
$l_1 = 2,\ l_2 = 1,\ l_3 = 1$
$r_1 = 0,\ r_2 = 1,\ r_3 = 1$

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

Optimizing Transaction Schedules via Quantum Annealing

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

$$C = \underbrace{\sum_{\{t_{i_1}, t_{i_2}\} \in O}}_{\text{blocking transactions}} \overbrace{\sum_{j_1=1}^{k}}^{\text{machines}} \underbrace{\sum_{s_1=0}^{r_{i_1}}}_{\text{start times}} \overbrace{\sum_{j_2 \in J}}^{\substack{\text{remaining}\\ \text{machines}}} \underbrace{\sum_{s_2=q}^{p}}_{\text{invalid start times}} X_{i_1,j_1,s_1} X_{i_2,j_2,s_2} \quad \text{for } J = \{1, ..., k\} \setminus \{j_1\}, q = \max\{0, s_1 - l_{i_2} + 1\}, p = \min\{s_1 + l_{i_1}, r_{i_2}\}$$

Example:

$$R = 2$$
$$T = \{t_1, t_2, t_3\} \text{ with } |T| = n = 3$$
$$M = \{m_1, m_2\} \text{ with } |M| = k = 2$$
$$O = \{(t_2, t_3)\}$$
$$l_1 = 2,\ l_2 = 1,\ l_3 = 1$$
$$r_1 = 0,\ r_2 = 1,\ r_3 = 1$$

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

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
  - Weigths in A, B and C ≥ 1
    $\Rightarrow$ first priority is validity, second priority is optimality

Example:

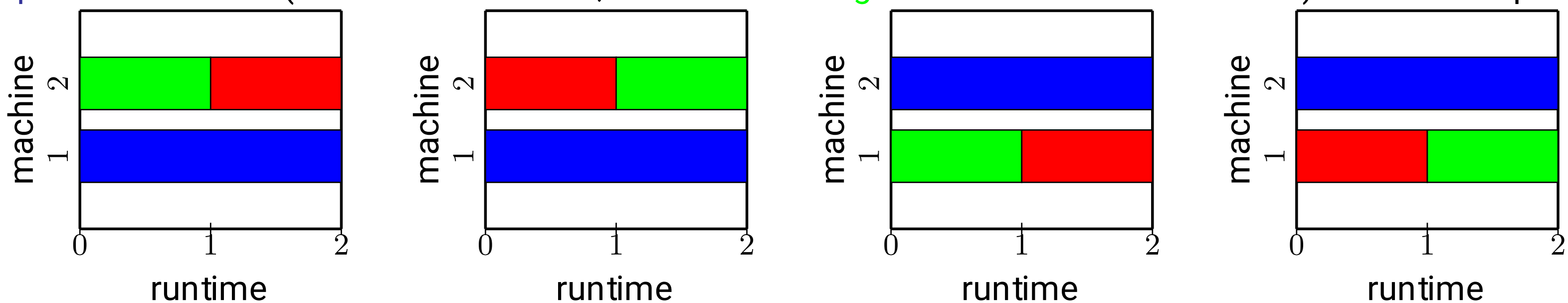
$$\begin{aligned} R &= 2 \\ T &= \{t_1, t_2, t_3\} \text{ with } |T| = n = 3 \\ M &= \{m_1, m_2\} \text{ with } |M| = k = 2 \\ O &= \{(t_2, t_3)\} \\ l_1 &= 2,\ l_2 = 1,\ l_3 = 1 \\ r_1 &= 0,\ r_2 = 1,\ r_3 = 1 \end{aligned}$$

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

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: $O((n \cdot k \cdot R)^2)$

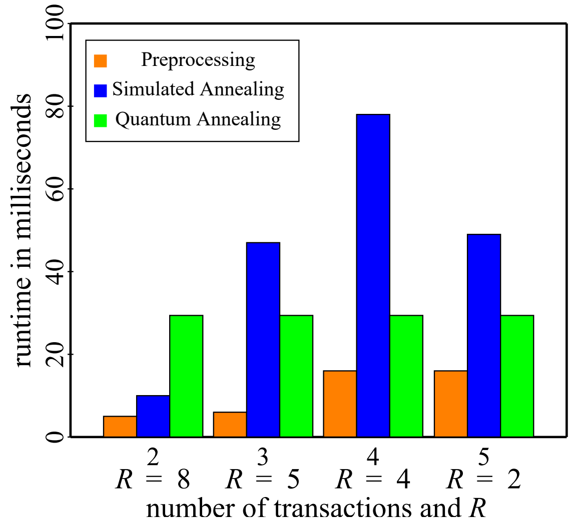
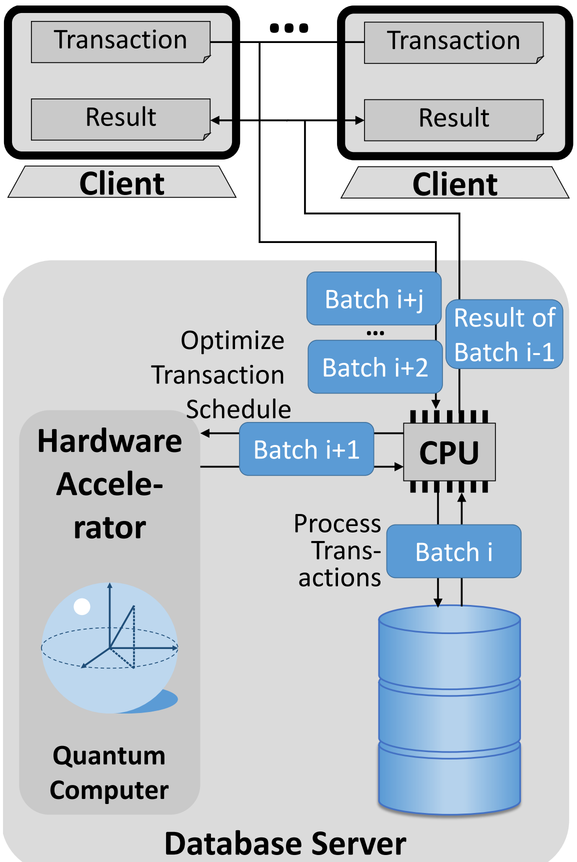


Fig.	k	n	R	O	$l_1, \ldots, l_n$	$r_1, \ldots, r_n$	**req. var.**
11	2	2	8	$\{\}$	8, 4	0, 4	8
		3	5	$\{(t_1, t_3)\}$	4, 5, 1	1, 0, 4	10
		4	4	$\{(t_2, t_4)\}$	3, 2, 1, 2	1, 2, 3, 2	16
		5	2	$\{(t_1, t_2), (t_4, t_5)\}$	1, 1, 1, 1, 1	1, 1, 1, 1, 1	10

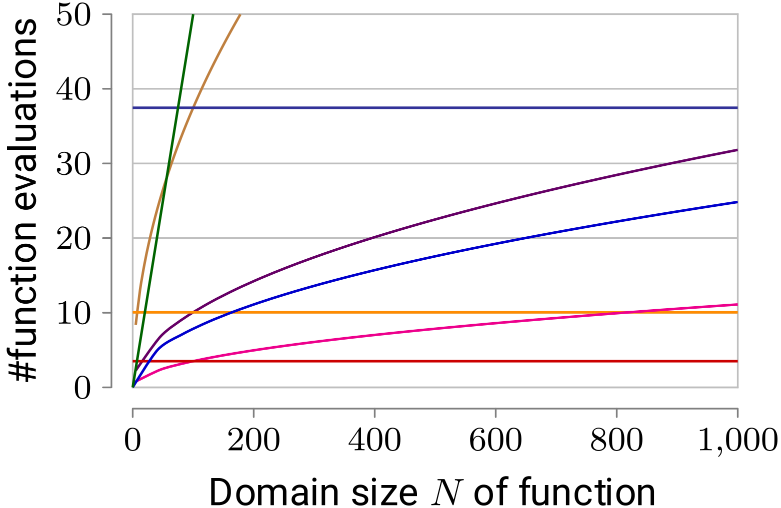
Optimizing Transaction Schedules via Quantum Computing



Grover's Search Algorithm

- Black box function $f: \{0, ..., 2^b - 1\} \mapsto \{true, false\}$
- Grover's search algorithm finds one $x \in \{0, ..., 2^b - 1\}$, such that $f(x) = true$
  - 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



- $\frac{N}{2}$ Linear Search

k known in advance:

- $\frac{\pi}{4} \cdot \sqrt{N}$ Grover's Search for $k = 1$
- $\frac{\pi}{4} \cdot \sqrt{\frac{N}{k}}$ Grover's Search for $k = \frac{5}{100} \cdot N$
- $\frac{\pi}{4} \cdot \sqrt{\frac{N}{k}}$ Grover's Search for $k = 5$

$k = \frac{5}{100} \cdot N$ unknown in advance:

- $\frac{9}{4} \cdot \sqrt{\frac{N}{k}}$ randomized Grover's Search
- $\frac{8 \cdot \pi}{3} \cdot \sqrt{\frac{N}{k}}$ deterministic Grover's Search

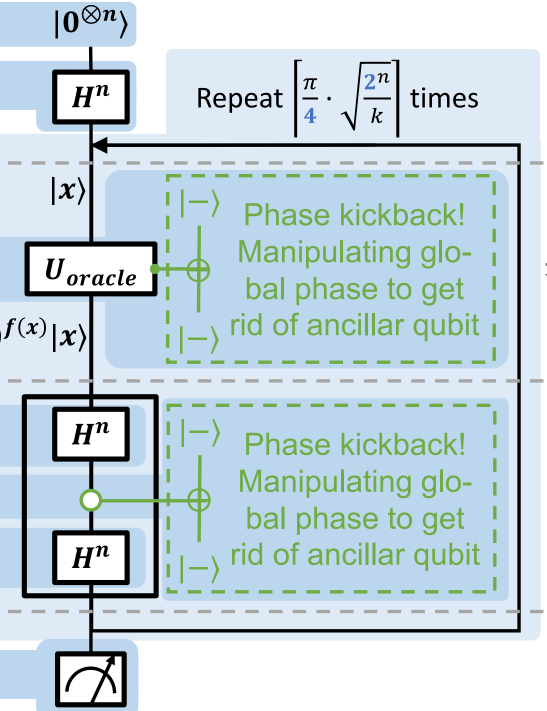
$k = 5$ unknown in advance:

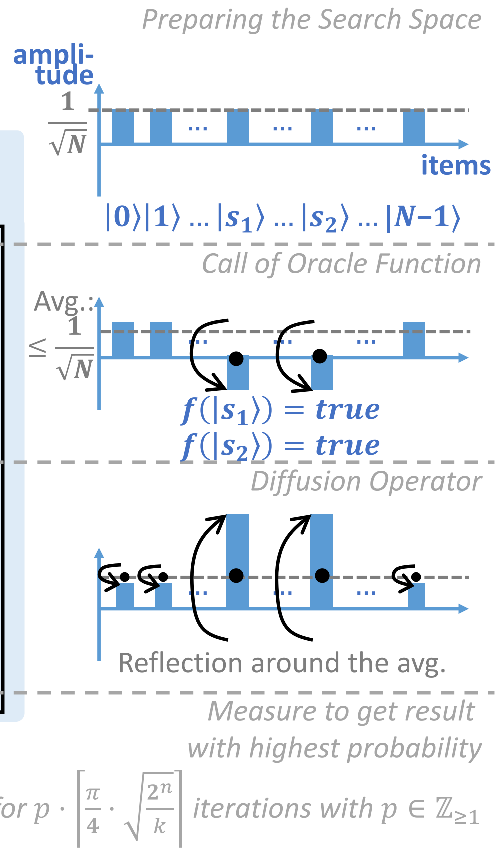
- $\frac{9}{4} \cdot \sqrt{\frac{N}{k}}$ randomized Grover's Search
- $\frac{8 \cdot \pi}{3} \cdot \sqrt{\frac{N}{k}}$ deterministic Grover's Search

N	Linear Search $k = 1$	Grover $k = 1$	$k = \frac{5}{100} \cdot N$			$k = 5$		
			known k	unknown k		known k	unknown k	
				randomized	deterministic		randomized	deterministic
10	5	2.48	3.51	10.06	37.46	1.11	3.81	11.85
100	50	7.85				3.51	10.06	37.47
1000	500	24.84				11.11	31.82	118.48
1000 000	500 000	785.40				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 := ⌊π/4 · √(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

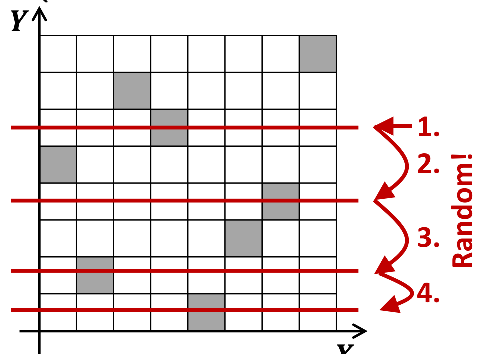
- If you apply Grover's search with $k \approx$ number of solutions k', then there is a high probability for success
- Several ways for unknown k:
  - Successively applying Grover's search with $k = N, \frac{N}{2}, \frac{N}{4}, \ldots, \frac{N}{2^t}$ until solution is found
    - **Runtime** complexity: With sufficiently high probability, a marked entry will be found by iteration $t = \log_2(\frac{N}{k'}) + c$ for some constant c
      $\Rightarrow$ #iterations $\leq \frac{\pi}{4}\left(1 + \sqrt{2} + \sqrt{4} + \ldots + \sqrt{\frac{N}{k' \cdot 2^c}}\right) = O\left(\sqrt{\frac{N}{k'}}\right)$
  - 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 𝔹, t: !ℕ !→ !𝔹, 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));
    }
}
```

Applying Grover - Finding Minimum

- Given function $f : \{0, ..., N-1\} \mapsto \mathbb{R}$
- Algorithm to find minimal $f(x)$:
  1. Choose randomly $x' \in N$
  2. Threshold $t := f(x')$
  3. while(true)
     a) $r :=$ Grover's Search for unknown k with oracle $f(x) < t$
     b) 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

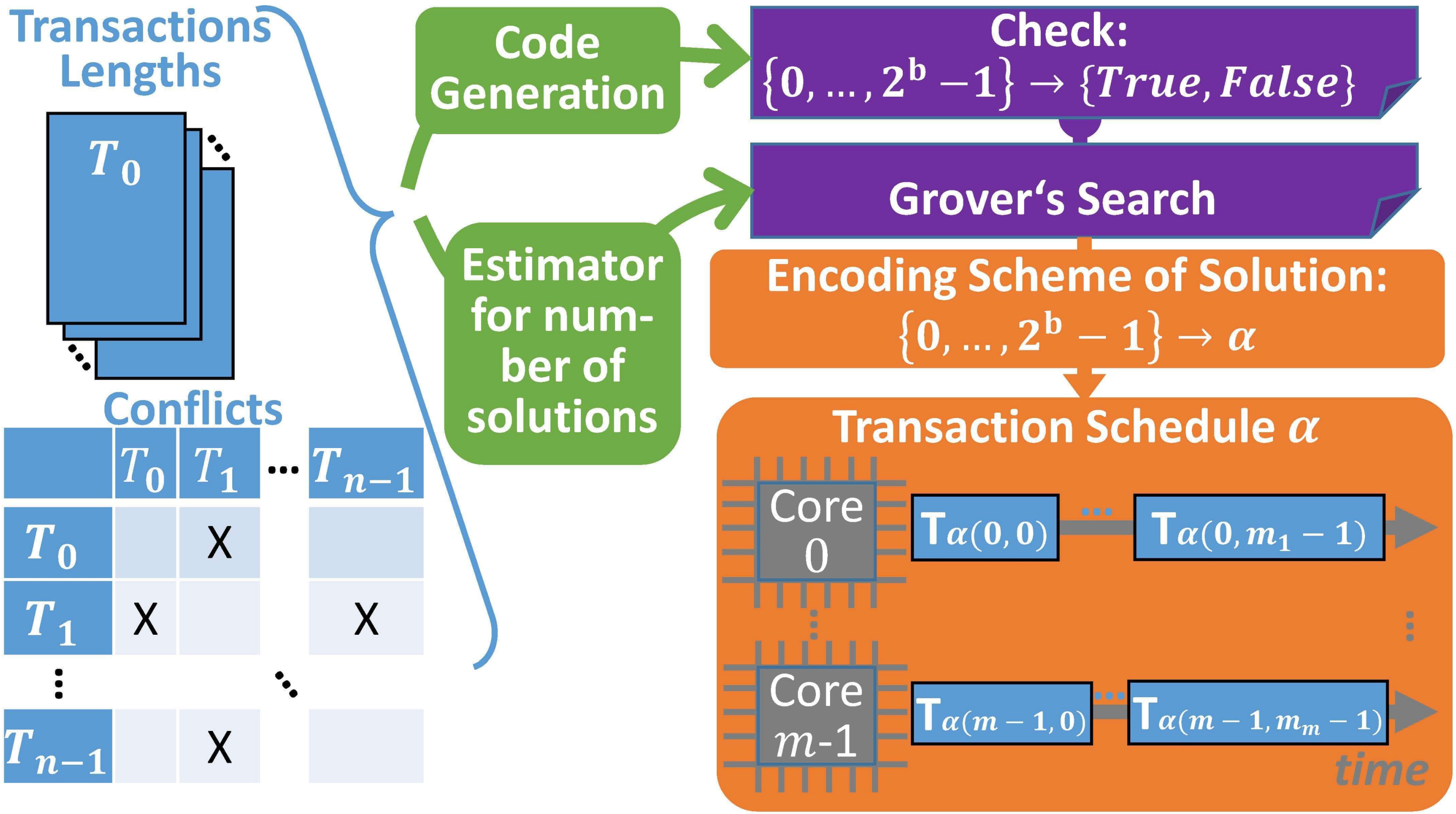
- Variants:
  - Application-oriented:
    - Estimate good initial threshold t, then continue original algorithm at 3. with threshold t
    - Example (see also next lecture unit):
      - Transactions $T_1, ..., T_p$ to be distributed to m cores of a CPU
      - Total runtime $l = \sum_{i=1}^{p} |T_i|$ 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, …, 2^b-1}
Output: $\{0, \ldots, n{-}1\}^{m-1} \times \{0, \ldots, n{-}1\}^{n-1}$
for(x **in** 1..m-1)
  $\mu_x = p$ **mod** n
  $p = p$ **div** n
a = [0, …, n-1]
for(i **in** 0..n-1)
  $j = p$ **mod** (n-i)
  $p = p$ **div** (n-i)
  $\pi[i] = a[j]$
  a[j] = $a[n\text{-}i\text{-}1]$
return ($\mu_1, \ldots, \mu_{m-1}, \pi$)

Example (n=4, m=2):
29

x = 1:
μ_1 = 1
p = 7
a = [0, 1, 2, 3]

i = 0:	i = 1:	i = 2:	i = 3:
j = 3	j = 1	j = 0	j = 0
p = 1	p = 0	p = 0	p = 0
π[0] = 3	π[1] = 1	π[2] = 0	π[3] = 2
a[3]=a[3]	$a[1]=a[2]$	$a[0]=a[1]$	$a[0]=a[0]$

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
  $\Rightarrow$ circuit is generated dependent on the concrete problem instance
- Sketch of algo:

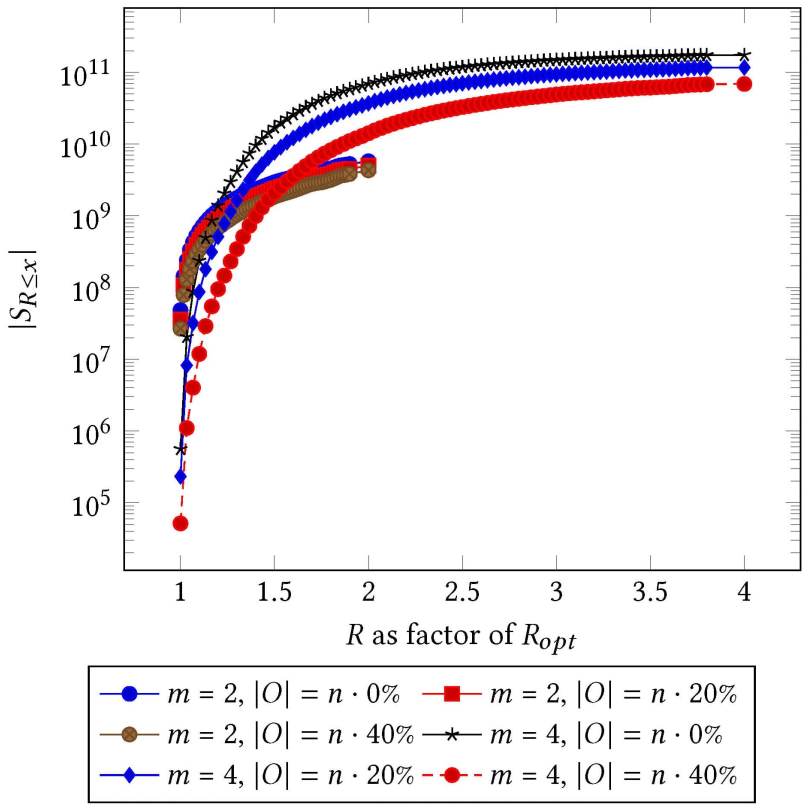
1. Determine Separators and Permutation	$O(m+n)$
2. Check Validity of Separators	$O(m)$
3. $\forall i$: Determine lengths of i-th transaction in permutation	$O(n \cdot log_2\ (n))$ with decision tree over transaction number
4. Check: Which separator configuration? For current case:	$O(n)$
4a. determine total runtime of core and check if it's below given limit	$O(n)$
4a. determine start and end times of conflicting transactions	$O(n \cdot log_2\ (min(n,c)) + c)$ with decision tree over conflicting transactions (for $n >> c$) or transaction numbers (for $c >> n$)
5. Check: Do conflicting transactions overlap?	$O(c)$
	$\sum : O(n \cdot log_2\ (n) + c)$

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