Lecture

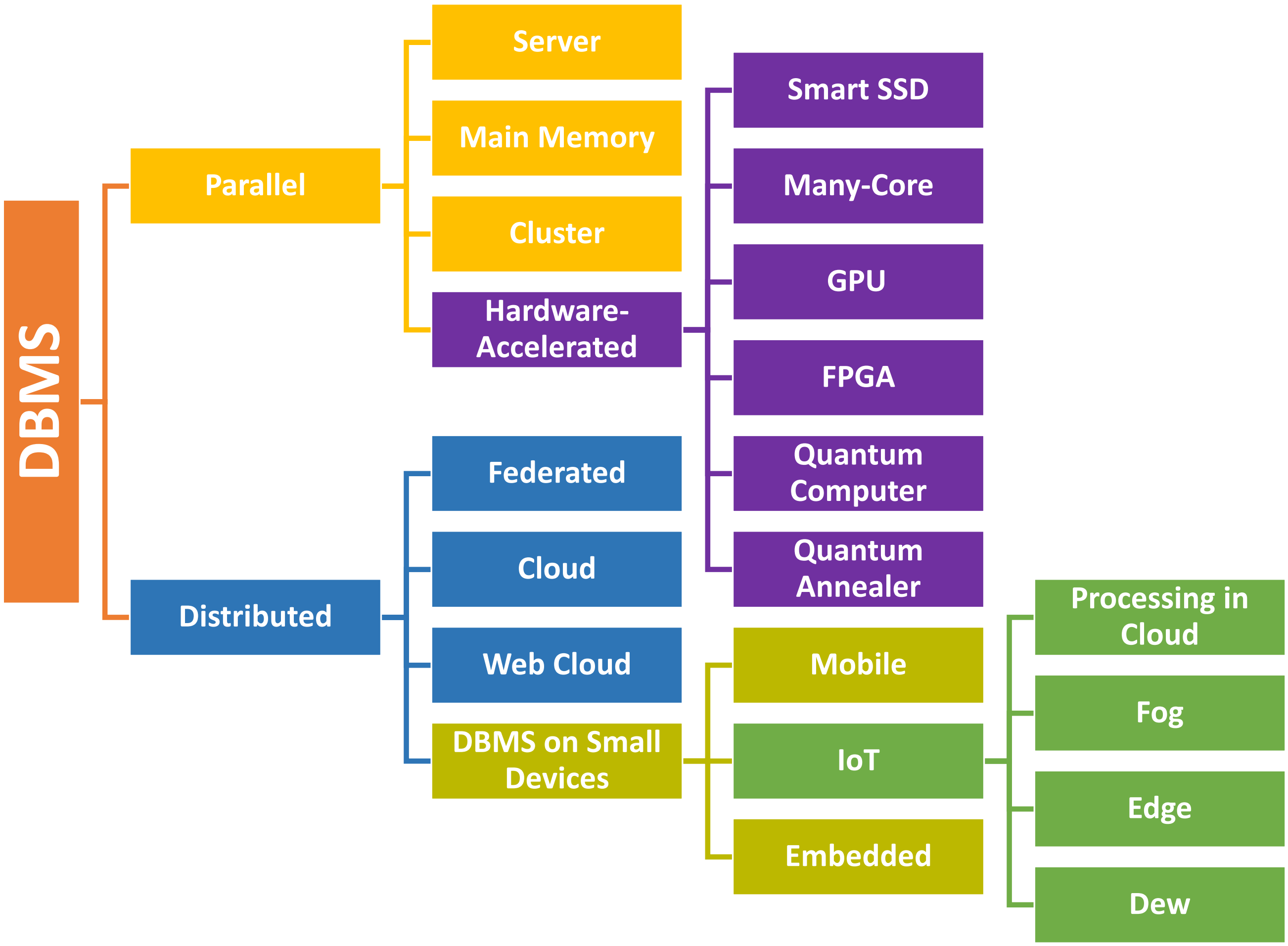
# Quantum Computing

(CS5070)

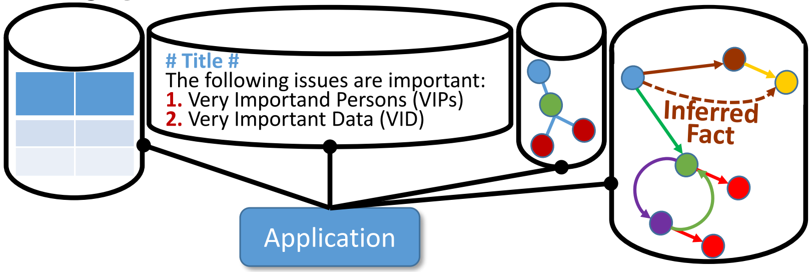
# Quantum Annealing Versus Grover's Search: Optimizing Transaction Schedules

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

# Platform-specific types of DBMS

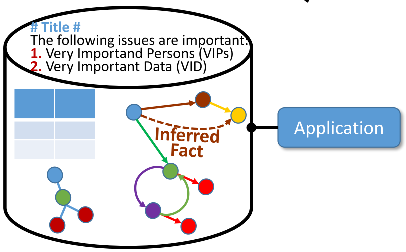


## Polyglot Persistence




- data sources: integration at application level
- performance of data processing cannot be fully optimized
- fault-tolerance cannot be transparently offered across the different databases
- zoo of query languages
+ features of different types of databases can be used

## Multi-Model DBMS (MM-DBMS)

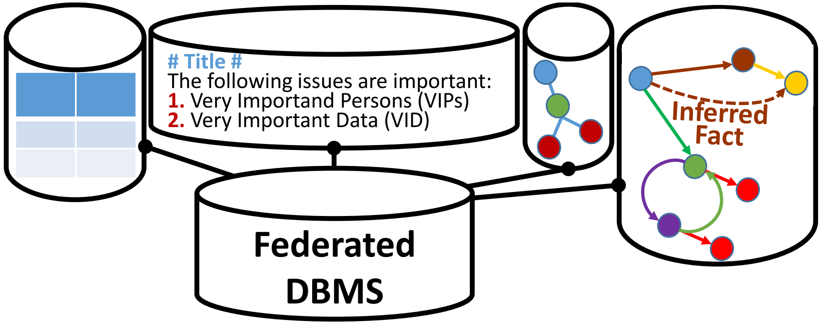



+ full and uniform data integration at database level
+ performance: fully optimized across different data models
+ transparent fault-tolerance
+ SQL standards:

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 can**not** be used

# Federated DBMS




- Bottom-up-integration of existent databases
- mostly independent DBMS with private conceptual database schemes
- partially enabling external accesses (in cooperation)
- heterogeneity of data models and transaction management possible (but relational DBMS in most times)

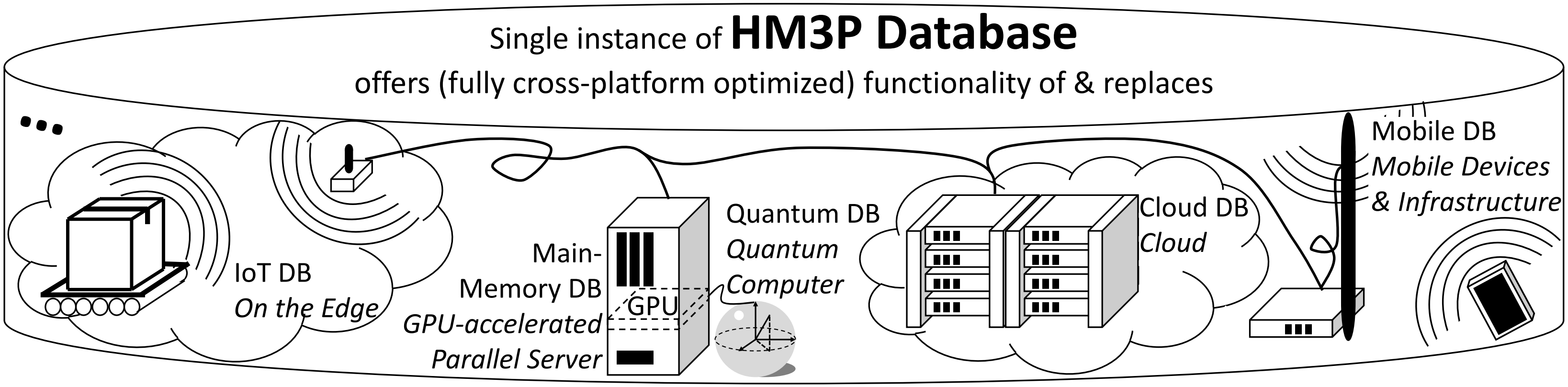
\- problems with semantic heterogeneity

\- transparency in distribution only partially achievable

# One Size-Approach

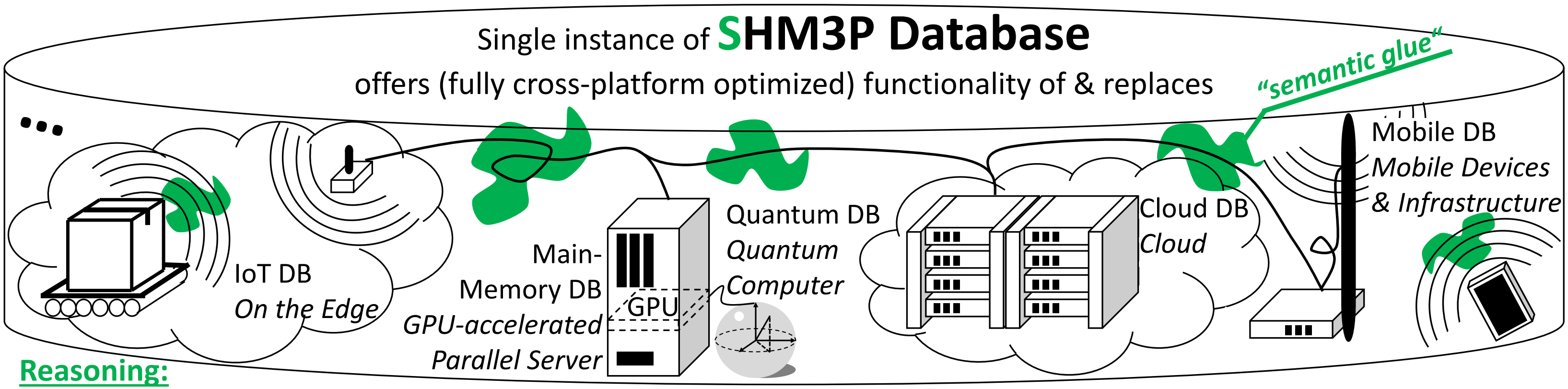
- M. Stonebraker, U. Cetintemel. *"One Size Fits All": An Idea Whose Time Has Come and Gone*. ICDE 2005
  - *The last 25 years of commercial DBMS development can be summed up in a single phrase: "One size fits all".*
  - *...this concept is no longer applicable to the database market...*
- Our approach: **Enlarge the size!**
  - Over the boundaries and limitations of single platforms and their specialized approaches
  - Increase transparency, performance and ease of use

# 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**

# Variant: Semantic HM3P (SHM3P) DB




**Reasoning:**

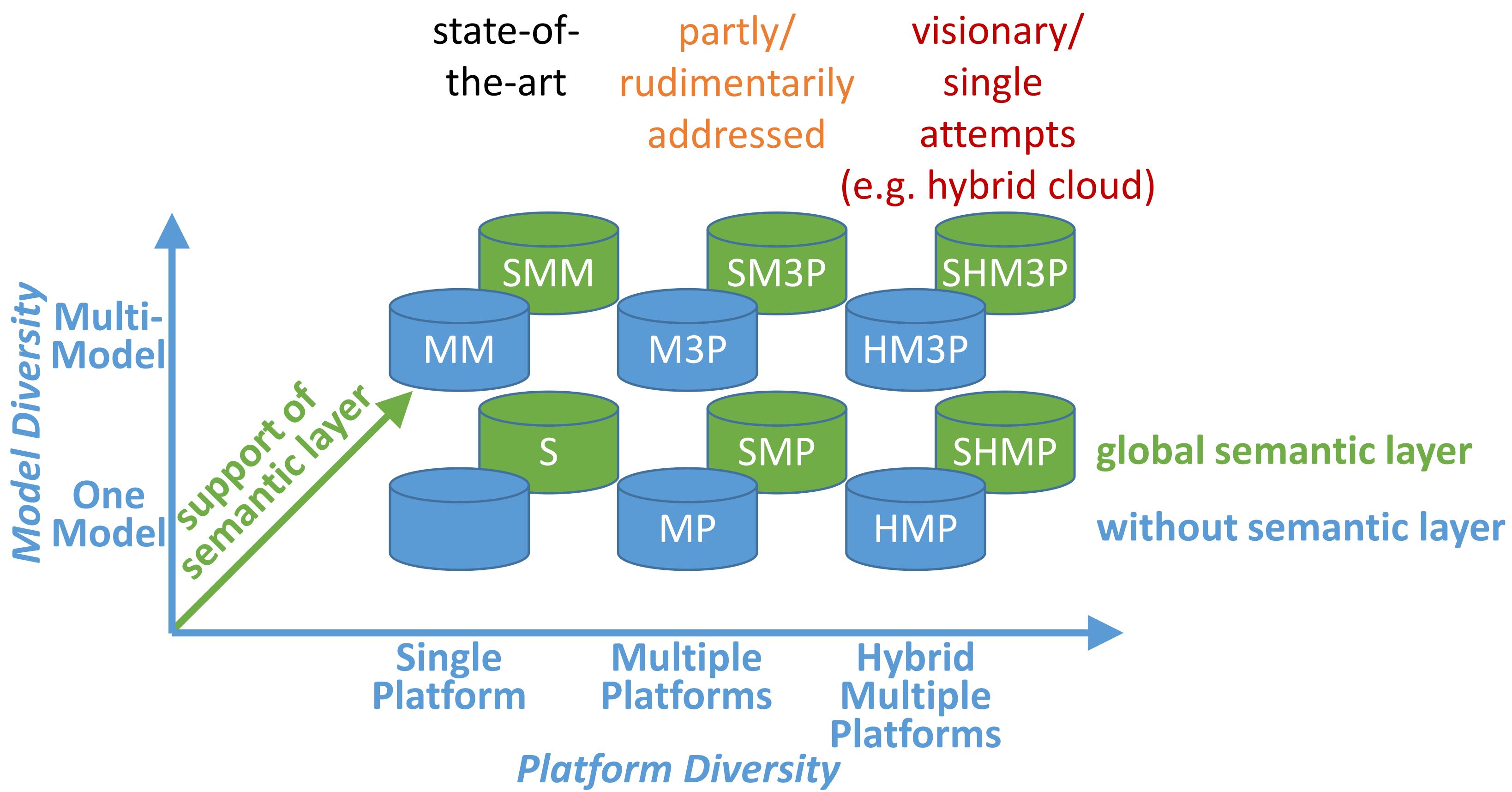
| | | | |
|---|---|---|---|
| **Lightweight reasoning on large data sizes of IoT devices** | **Heavyweight reasoning on moderate data sizes** | **Heavyweight reasoning on large data sizes** | **Reasoning on small data sizes of mobile devices** |

**How to integrate the different reasoning capabilities and requirements into one transparent global reasoner?**

- Semantic Layer as glue between other models and platforms
- new challenges like integrating different types of reasoners in a transparent global reasoner

+ Features of HM3P databases
+ Easier data integration
- Performance issues may occur due to semantic layer

# Types of DBMS



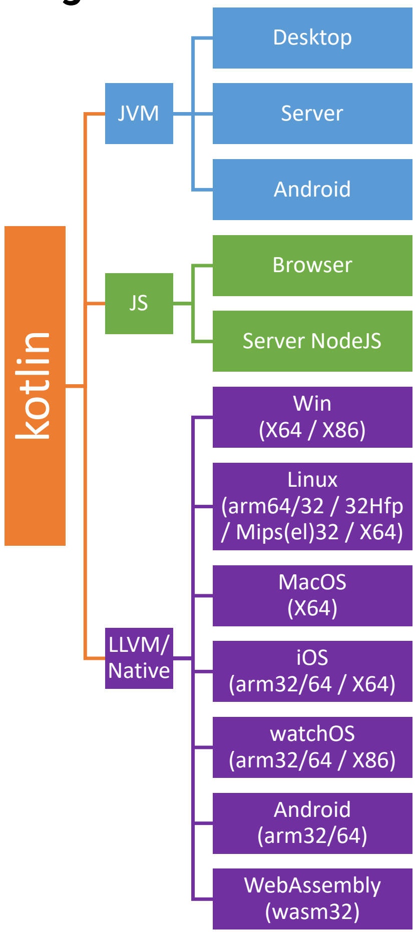

Legend: S: Semantic MP: Multi-Platform MM: Multi-Model M3P: MM MP H: Hybrid

# Multi-Platform Development of DBMS

- Native Binaries via C/C++
  - support of a new platform: porting code is necessary
  - code close to hardware, fast execution
  - direct access to native libraries
  - doesn't run in browser
  - most server DBMS: C/C++ code
- Java/Java Virtual Machine (JVM)
  - runs on many platforms (without porting code)
  - interpreted bytecode, via Just-In-Time compilation comparable speed to native execution
  - no direct access to native libraries
  - does neither run on iPhone nor in browser
  - many NoSQL/NewSQL/Cloud DBMS: Java (or JVM language like Scala) code
- Code generation for query processing via C/C++ or Janino-Compiler (JVM)

# Multi-Platform Development with Kotlin

**Targets:**




- Most target platforms are supported
- Splitting the project in platform-**in**dependent and platform-dependent code
  - Platform-dependent code can be partly coded in the programming language of the target platform (e.g., Java for JVM, JS for Web)
- Enables one code repository for various target platforms
  - Sharing of code between server & (various) clients
- Avoids efforts to port code (into other programming languages)

# Multi-Platform Development with Kotlin

- Common Module
  - Code independent of platforms containing declarations for platform dependent code without implementation, e.g.:

```
expect fun formatString(source: String, vararg args: Any): String
expect annotation class Test
```

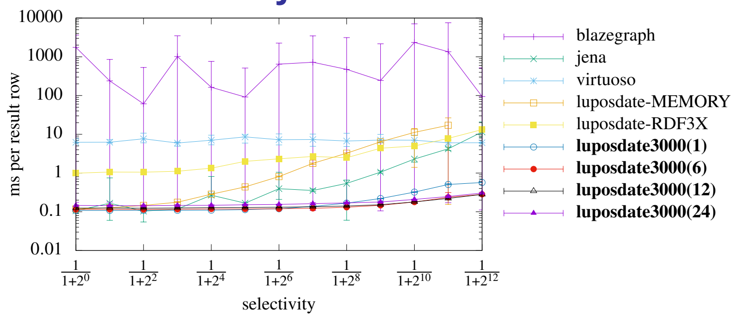
- Platform Module
  - Implementation of within the common module declared platform-dependent code (and other platform-dependent code), e.g.:

```
actual fun formatString(source: String, vararg args: Any) =
                                      String.format(source, args)
actual typealias Test = org.junit.Test
```

- Regular Module
  - depend on platform modules or platform modules depend on this module
- **However:** **High compilation times**, **faster**: Including different sets of source code directories for different targets and configurations (e.g., centralized, Cloud, P2P, browser, ...)

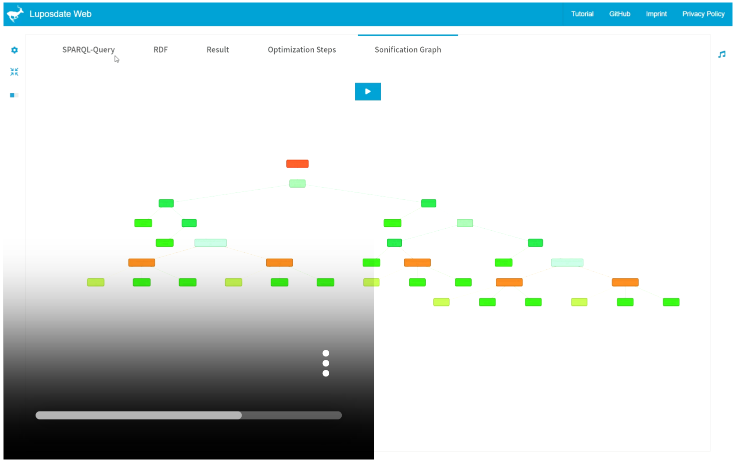
# The Power of Multi-Platform: LUPOSDATE3000

- ultra-fast in jvm...



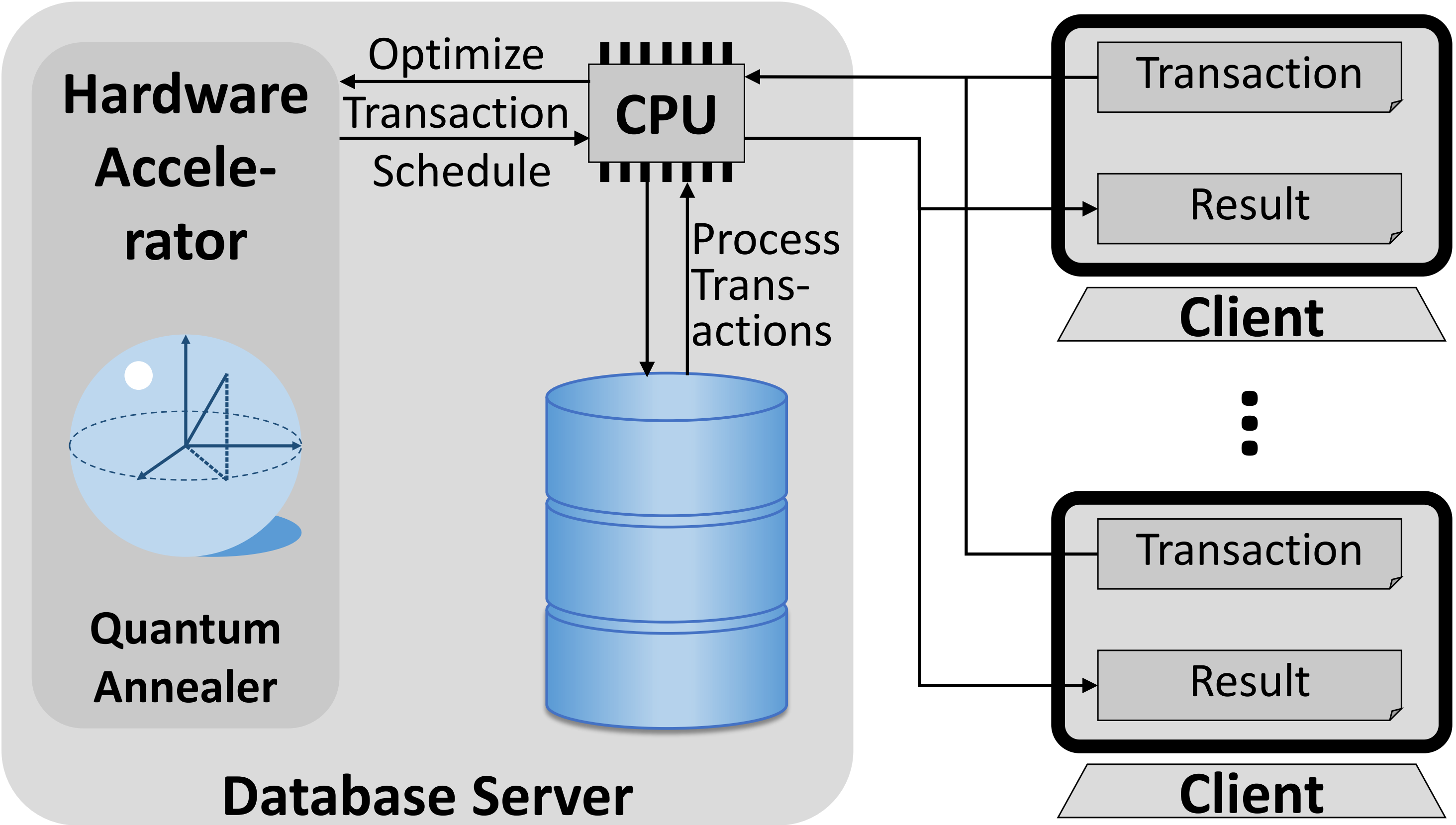

B. Warnke, M.W. Rehan, S. Fischer, S. Groppe: Flexible data partitioning schemes for parallel merge joins in semantic web queries in: BTW'21

- ...but also enabling web demos running completely in the browser!

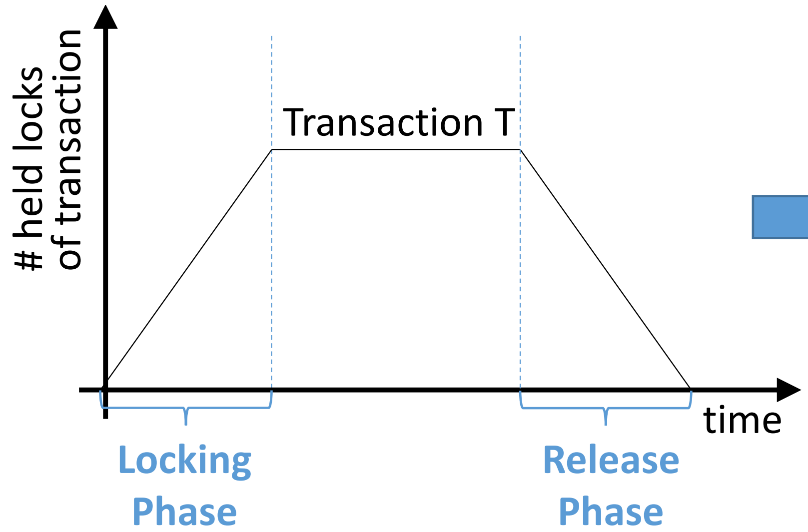


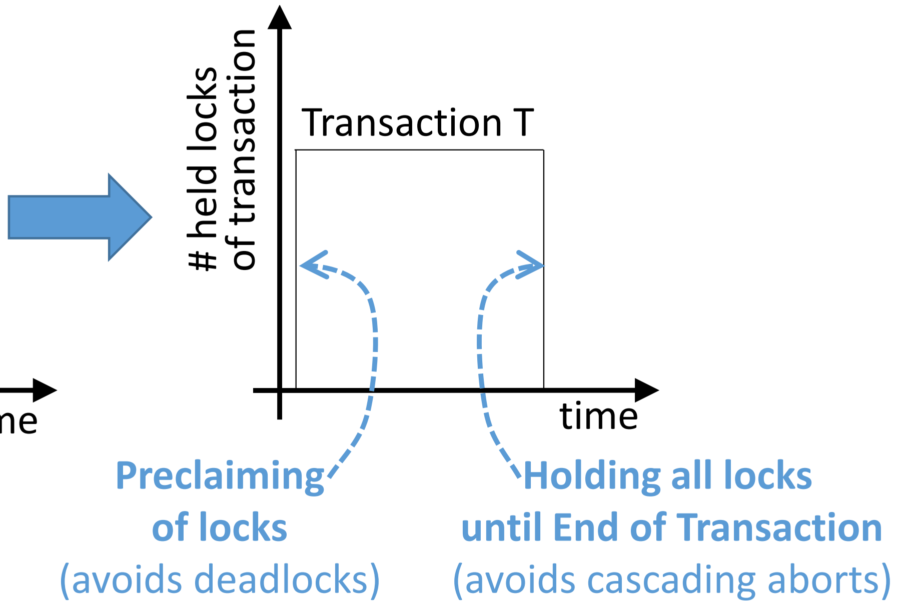

S. Groppe, R. Klinckenberg, B. Warnke. Sound of Databases: Sonification of a Semantic Web Database Engine. PVLDB, 14(12), 2021

# Using Hardware Accelerator for optimizing Transaction Schedules



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

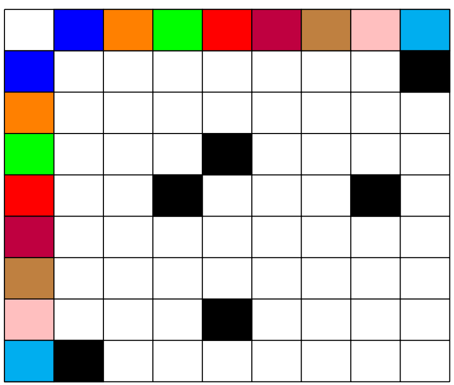


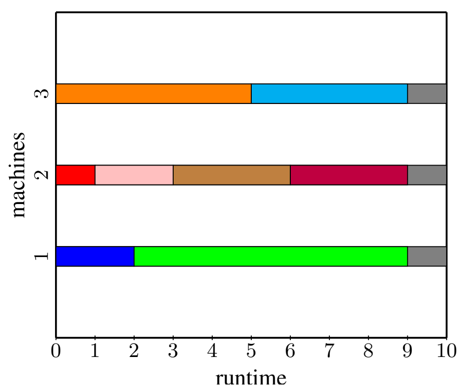

- required locks to be determined by
  - static analysis of transaction, or if static analysis is not possible:
  - an additional phase at runtime before transaction processing
    - A. Thomson et al., "Calvin: Fast distributed transactions for partitioned database systems", SIGMOD 2012.

# *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

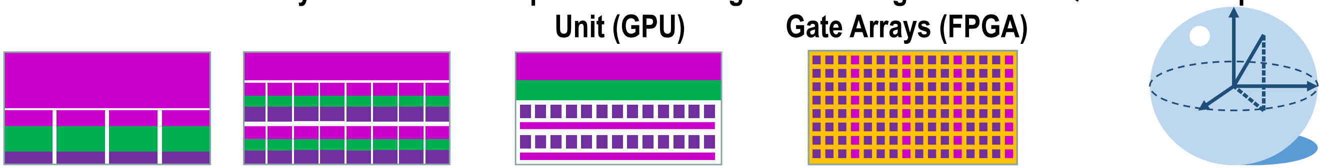



Transaction schedule

- JSSP is among the hardest combinatorial optimizing problems*
- $\Rightarrow$ 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

# Architectures of Emergent Hardware

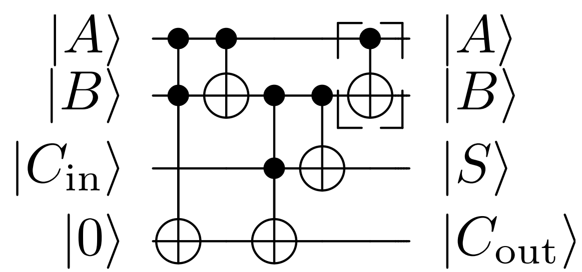


| | Multi-Core CPU | Many-Core CPU | Graphics Processing Unit (GPU) | Field Programmable Gate Arrays (FPGA) | Quantum Computer/Annealer |
|---|---|---|---|---|---|
| Cores | ~10 | ~100 | ~1000 | ~100 000 | ~100/~5000 qubits |
| Core Complexity | Complex (optimized for single thread performance) | | |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

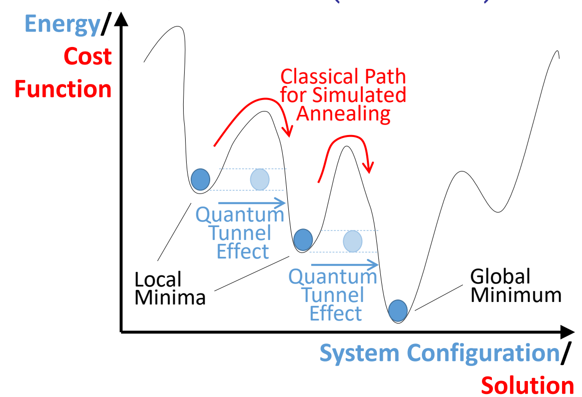
**Legend:** Computational Unit, Execution Controller, Interconnection Network, On-Chip Memory

# 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

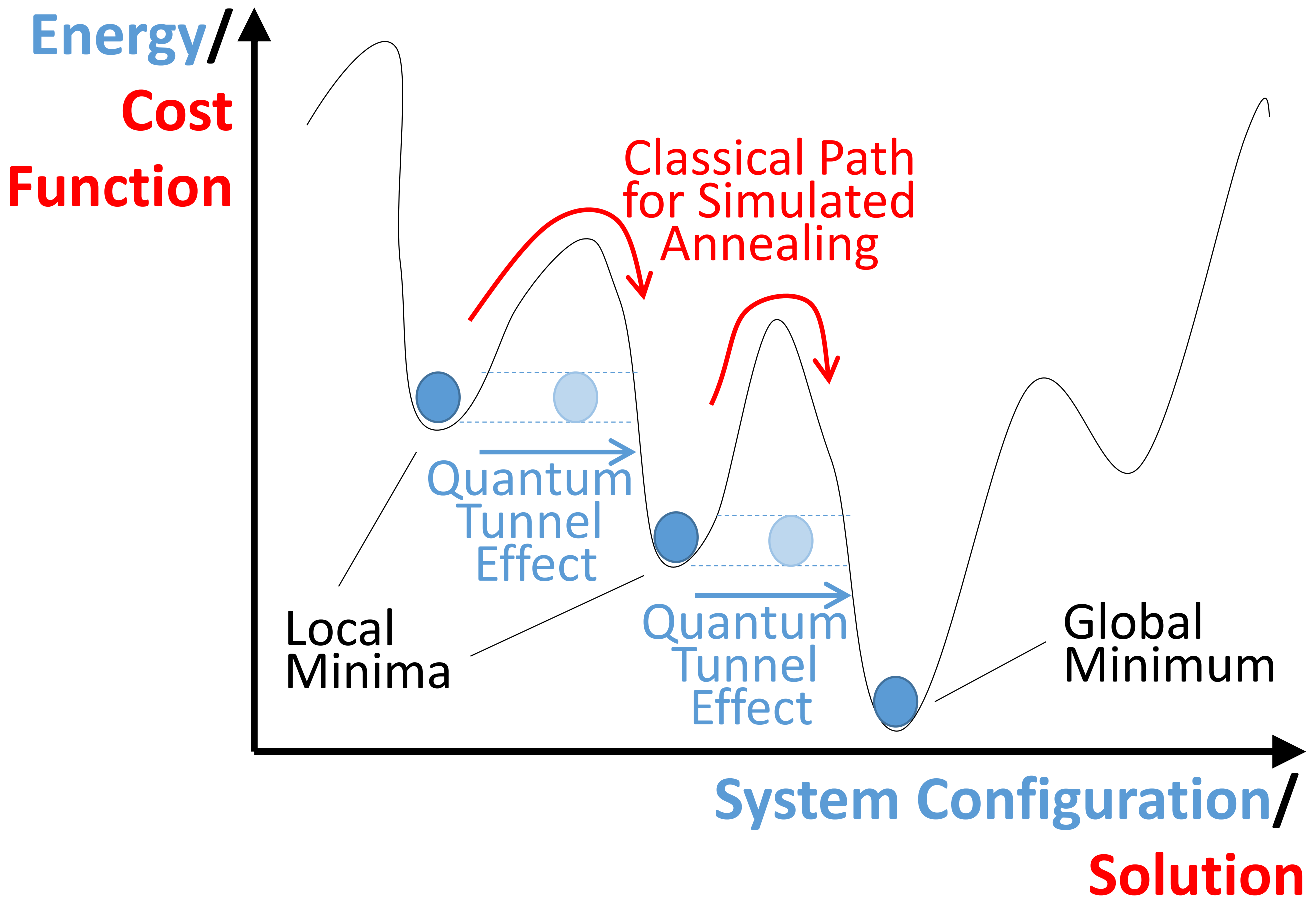



Quantum Circuit (Full Adder)




Simulated versus Quantum Annealing

# Quantum versus Simulated Annealing



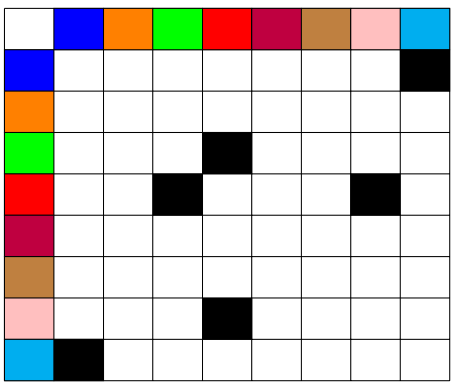
# *Optimizing* Transaction Schedules via Quantum Annealing

- 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:

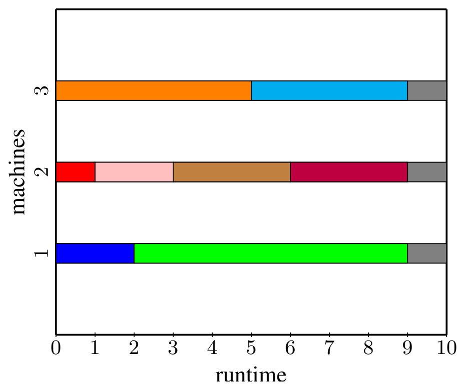
$$\sum_{0<i\leq N} w_i X_i + \sum_{i\leq j\leq N} w_{ij} X_i X_j, \text{ where } w_i, w_{ij} \in \mathbb{R}$$

# *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

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

Example:

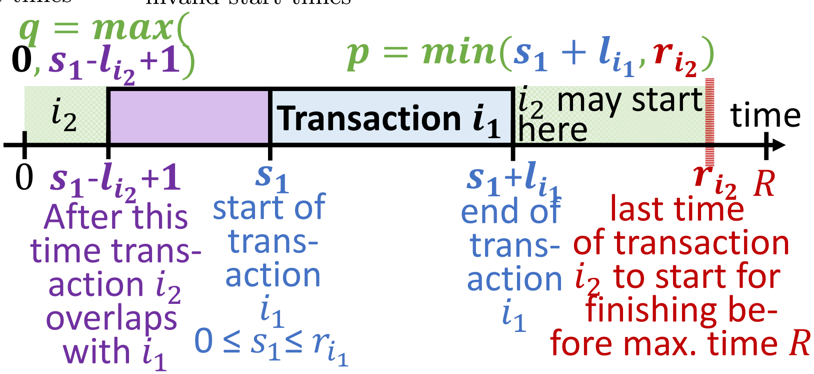
$$R = 2$$
$$T = \{t_1, t_2, t_3\} \text{ with } |T| = n = 3$$
$$M = \{m_1, m_2\} \text{ with } |M| = k = 2$$
$$O = \{(t_2, t_3)\}$$
$$l_1 = 2,\ l_2 = 1,\ l_3 = 1$$
$$r_1 = 0,\ r_2 = 1,\ r_3 = 1$$

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

# *Optimizing* Transaction Schedules via QA

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

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




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

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

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 $\geq 1$
    $\Rightarrow$ first priority is validity, second priority is optimality

Example:

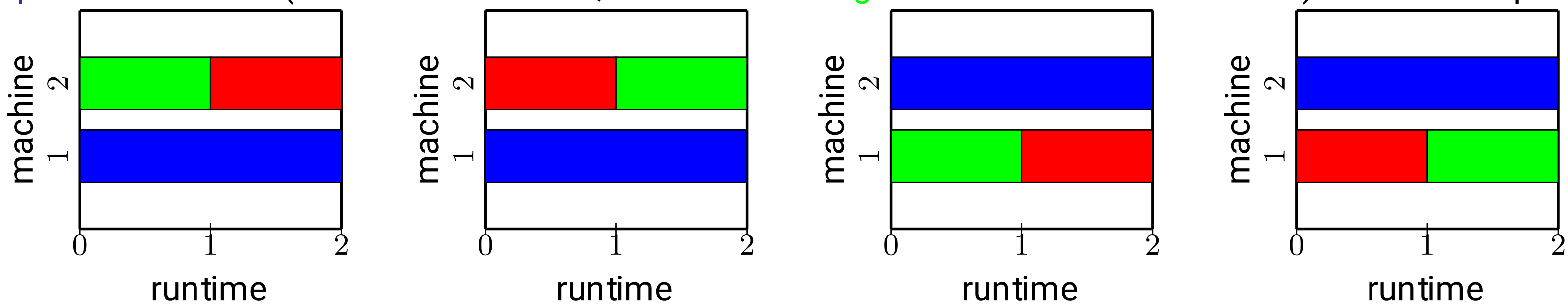
$R = 2$
$T = \{t_1, t_2, t_3\}$ with $|T| = n = 3$
$M = \{m_1, m_2\}$ with $|M| = k = 2$
$O = \{(t_2, t_3)\}$
$l_1 = 2,\ l_2 = 1,\ l_3 = 1$
$r_1 = 0,\ r_2 = 1,\ r_3 = 1$

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

# *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:

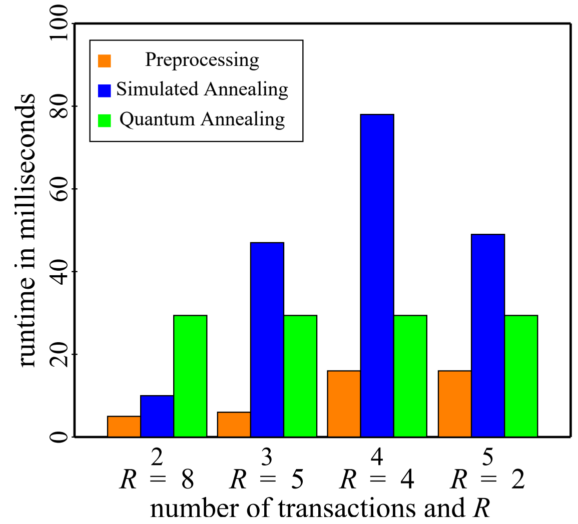



$X_{1,1,0}, X_{2,2,0}, X_{3,2,1}$ $X_{1,1,0}, X_{2,2,1}, X_{3,2,0}$ $X_{1,2,0}, X_{2,1,0}, X_{3,1,1}$ $X_{1,2,0}, X_{2,1,1}, X_{3,1,0}$

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

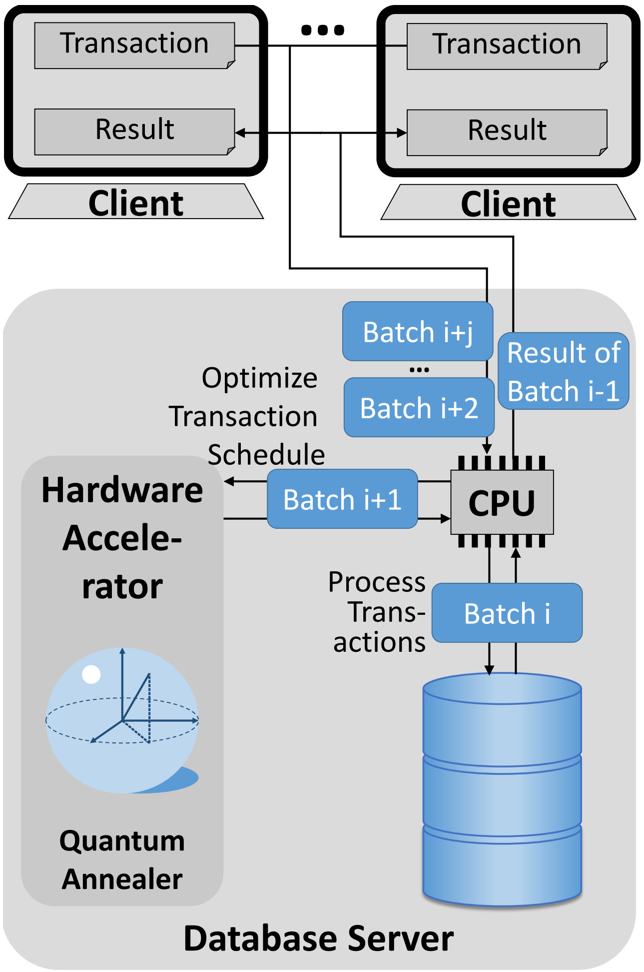
# *Optimizing* Transaction Schedules via Quantum Annealing

- 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 |

# Pipelining for further Speedup

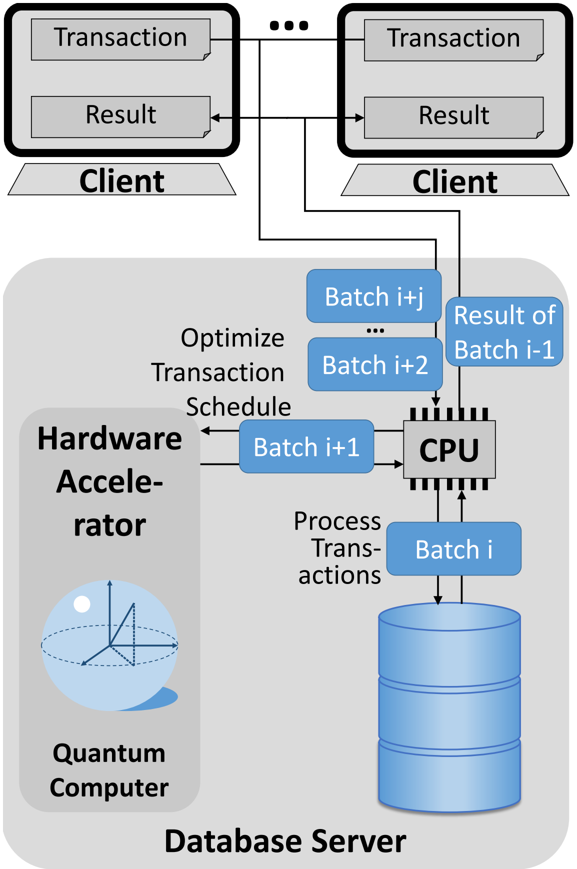


# Caching of & Reusing generated formulas to minimize preprocessing time

- 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 $r_i, ..., r_n$ 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.

  $\Rightarrow$ 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!

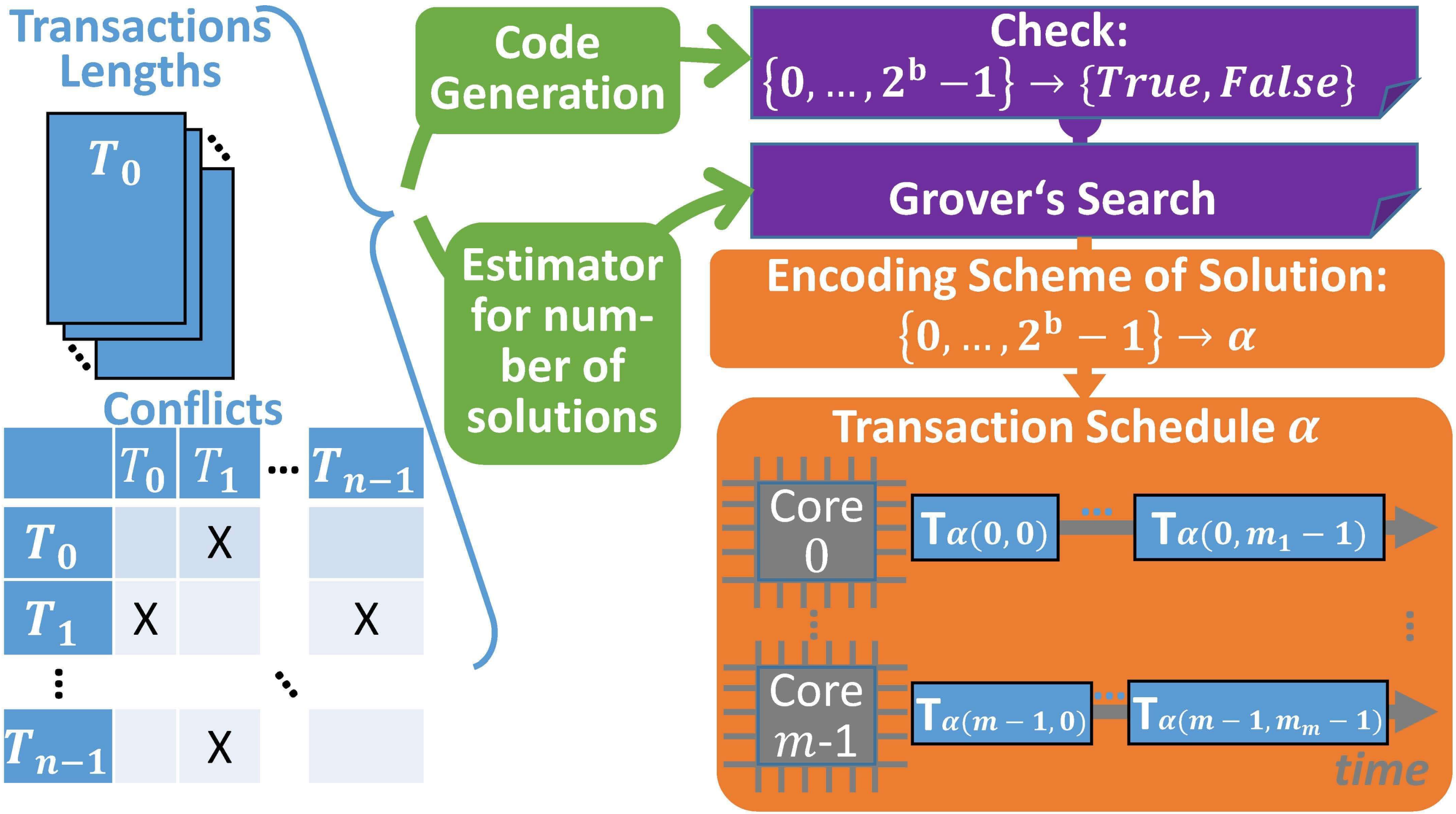
# Optimizing Transaction Schedules via Quantum Computing



# Grover's Search Algorithm

- Black box function $f$ :$\{0, ..., 2^b - 1\} \mapsto \{true, false\}$
- 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: $O(\sqrt{\frac{2^b}{k}} \cdot f'(b))$

# Overview of Optimizing Transaction Schedules via Quantum Computing



# Encoding Scheme of Transaction Schedules

**Algo** determineSchedule
**Input:** $p$:{0, …, $2^b$-1}
**Output:** {0, …, $n$−1}$^{m-1}$× {0, …, $n$−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] = $a[n$-$i$-1]
**return** ($\mu_1$, …, $\mu_{m-1}$, $\pi$)

**Example (n=4, m=2):**
29
$x$ = 1:
$\mu_1$ = 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 |
| $\pi$[0] = 3 | $\pi$[1] = 1 | $\pi$[2] = 0 | $\pi$[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:

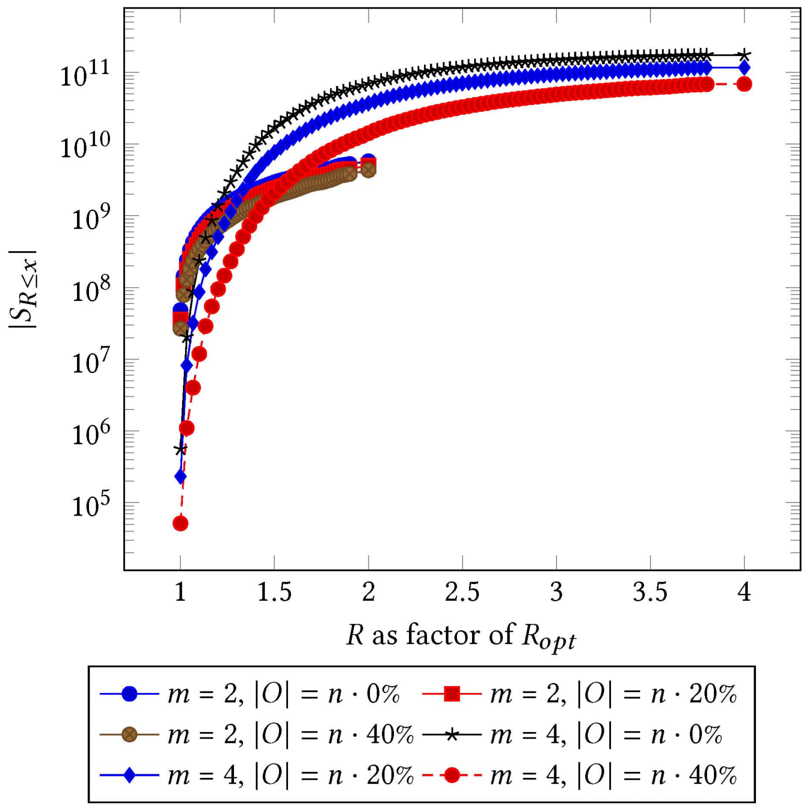
| | |
|---|---|
| 1. Determine Separators and Permutation | $O(m+n)$ |
| 2. Check Validity of Separators | $O(m)$ |
| 3. $\forall i$: Determine lengths of $i$-th transaction in permutation | $O(n \cdot log_2(n))$ with decision tree over transaction number |
| 4. Check: Which separator configuration? 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 |

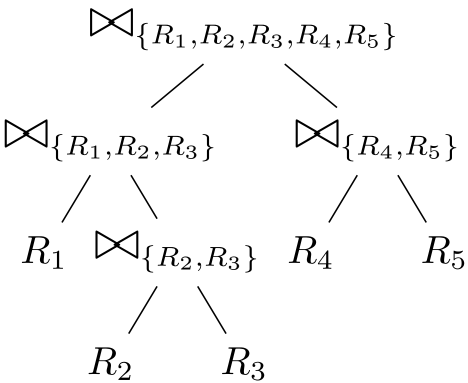
# Join Ordering with Quantum Annealing

- **Quantum Annealing** solves quadratic unconstrained binary optimization (**QUBO**) problems

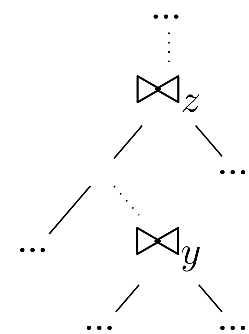
**Example Join Ordering:**

- $P$: power set of relations to join
- $P_k$: set of elements representing joins of $k$ relations, i.e., $P_k = \{a | a \in P \wedge |a| = k\} \subseteq P$
- $w_{max} = max(w_i) + c$, where $c > 0$ and $w_i$ represents the cost of join $i$
- **Rewarding joins with lowest costs:** $S_1 = \sum_{j=2}^{m} \sum_{i \in P_j} x_i \cdot (w_i - w_{max})$

Example




Constraint:




$y \subseteq z$

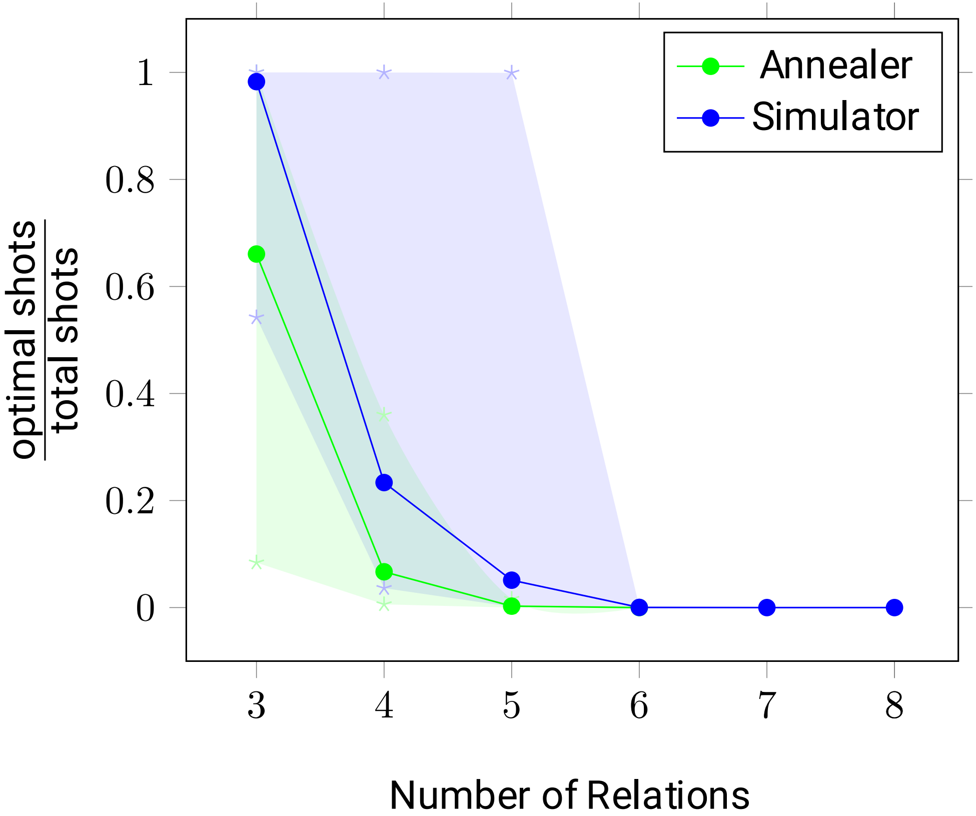
**Punish other combinations:**

$$\Rightarrow S_2 = \sum_{i=2}^{m-1} \sum_{j=i}^{m} \sum_{\substack{y \in P_i \\ \wedge z \in P_j \\ \wedge y \cap z \neq \emptyset \\ \wedge y \nsubseteq z}} x_y * x_z * w_{max}$$

**Minimize** $S = S_1 + S_2$

# Join Ordering with Quantum Annealing

- Real-world queries of the ErgastF1 Benchmark with PostgreSQL




- QPU access time approx. 100ms

# 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/

# 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