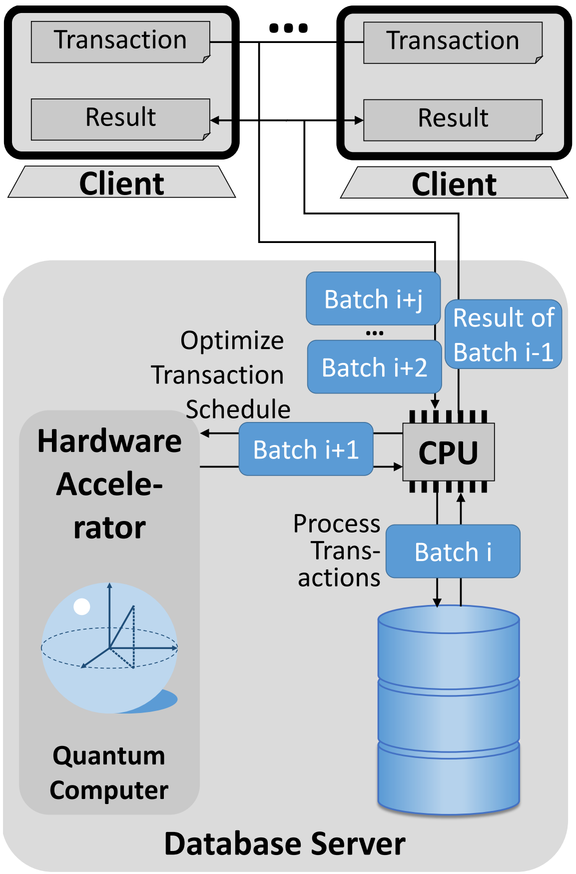
IDEAS 2021

July 14-16, 2020, Montreal, QC, Canada

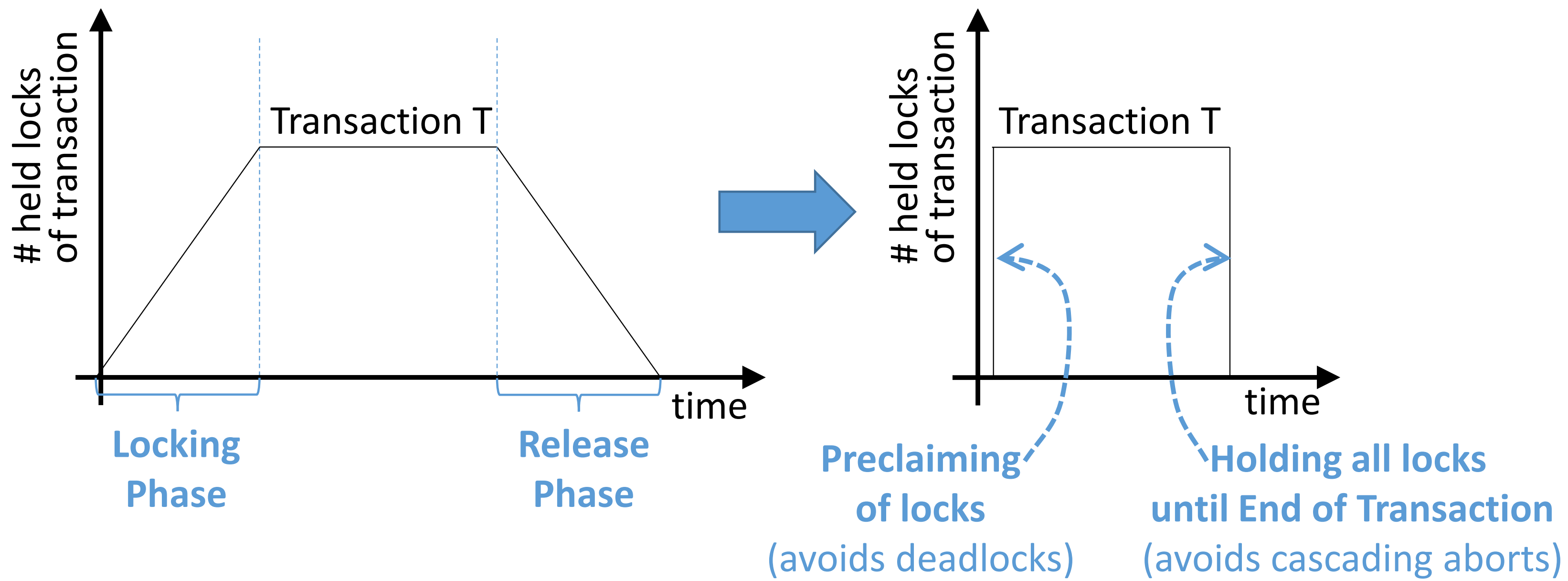
Optimizing Transaction Schedules on Universal Quantum Computers via Code Generation for Grover's Search Algorithm

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

Using Hardware Accelerator for optimizing Transaction Schedules



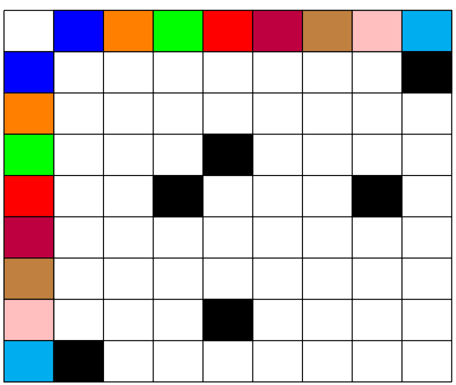
2 Phase Locking (2PL) versus Strict Conservative 2PL



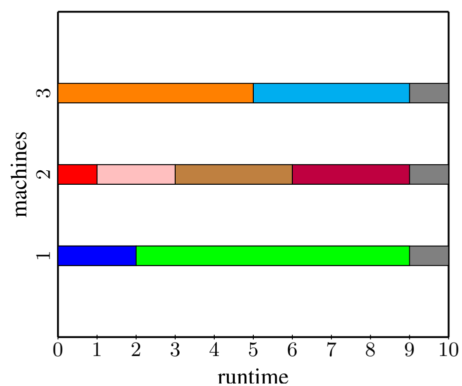
- required locks to be determined by
  - static analysis of transaction, or if static analysis is not possible:
  - an additional phase at runtime before transaction processing
    - A. Thomson et al., "Calvin: Fast distributed transactions for partitioned database systems", SIGMOD 2012.

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:



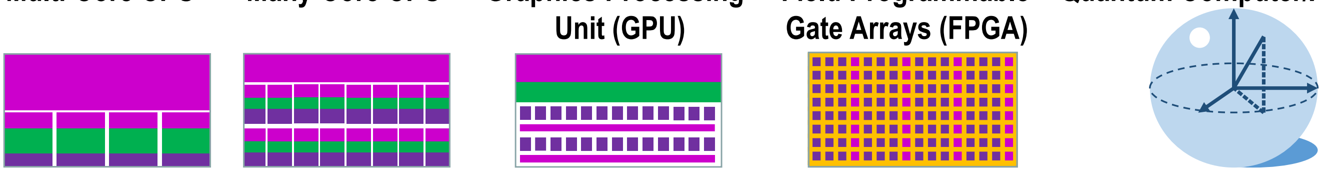
Black: Blocking transactions



Transaction schedule

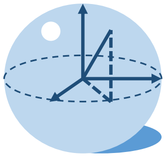
- JSSP is among the hardest combinatorial optimizing problems*
- ⇒ Hardware accelerating the optimization of transaction schedules

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	~50 / ~2000 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



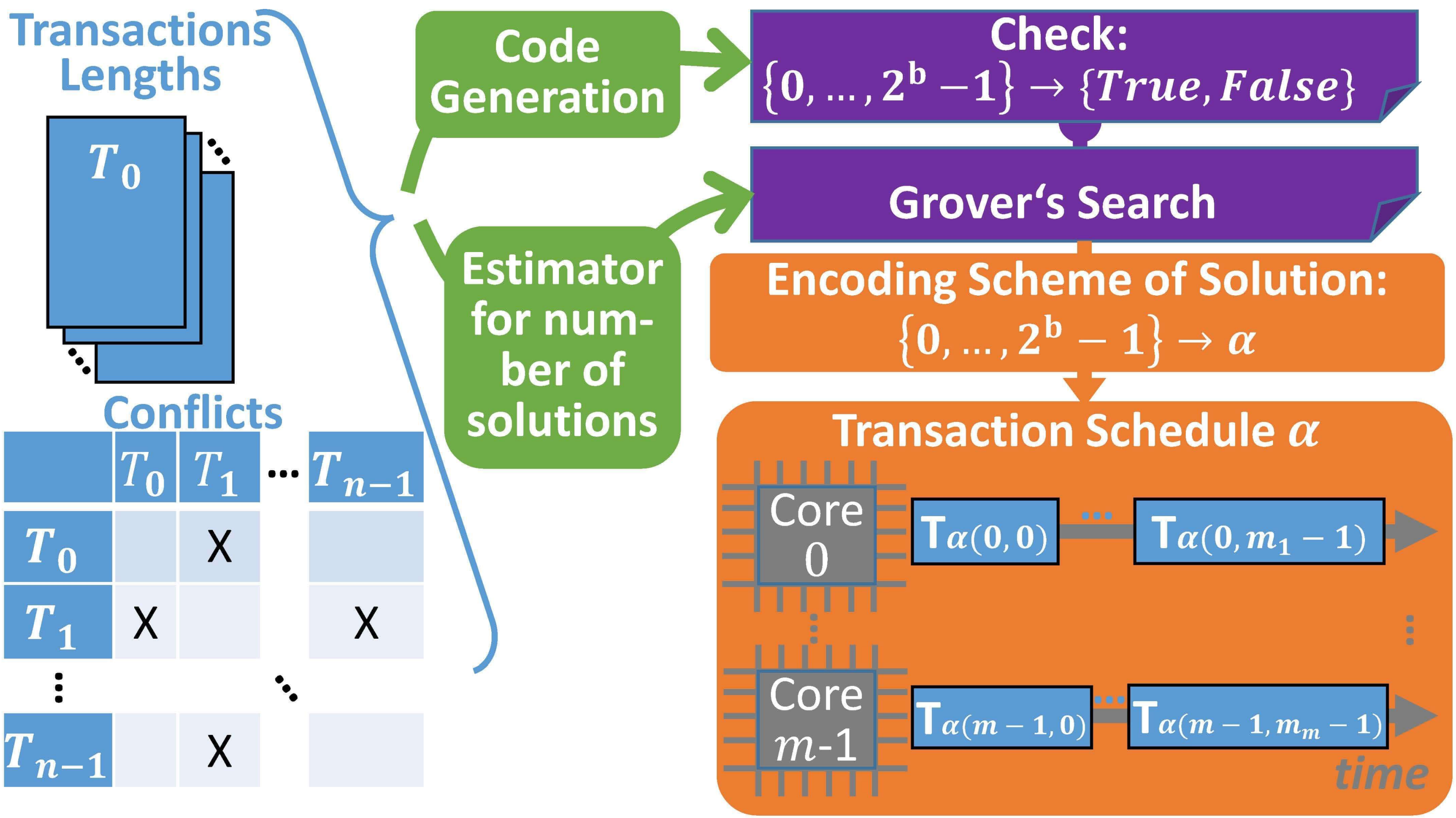
Quantum Computer

- 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

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: $\{0, …, n-1\}^{m-1} \times \{0, …, n-1\}^{n-1}$
for(x **in** 1..m-1)
  μ_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 (μ_1, …, μ_{m-1}, π)

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 must be generated dependent on the concrete problem instance, no general circuit to solve all instances of a problem
- Sketch of algo:

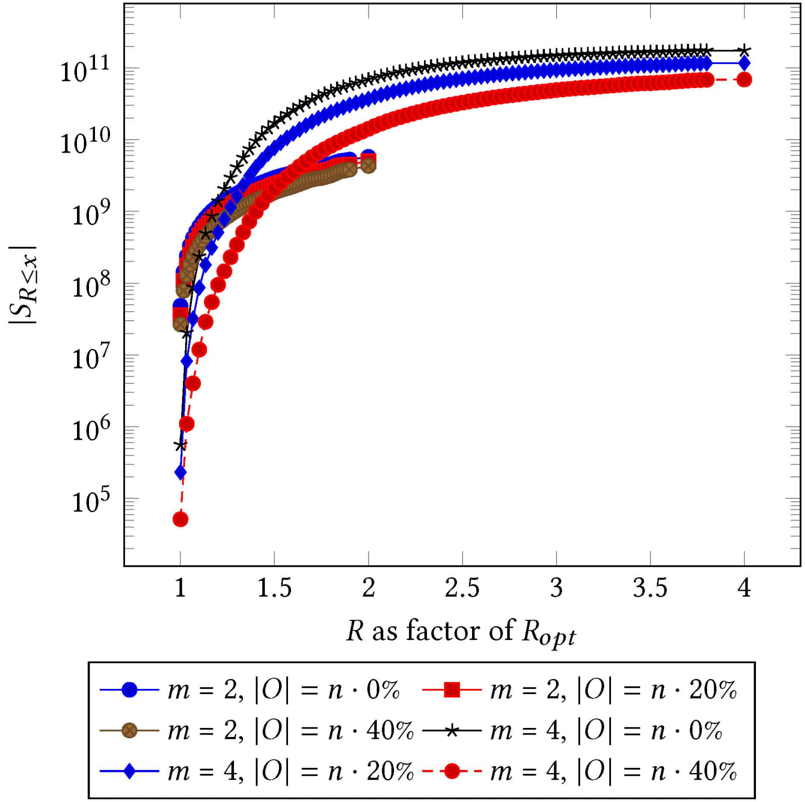
1. Determine Separators and Permutation	$O(m+n)$
2. Check Validity of Separators	$O(m)$
3. $\forall i$: Determine lengths of i-th transaction in permutation	$O(n \cdot log_2\ (n))$ with decision tree over transaction number
4. Check: Which separator configuration? For current case:	$O(n)$
4a. determine 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 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
  - Grover's search offers approx. quadratic speedup to approach on traditional computer
- 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
- Future Work
  - Quantum computing of other database problems