Parallelism and Quantum Computing

Qubit as state of a Quantum system, e.g. Spin up or Spin down of a particle.

|up> = |0> or |down> = |1>, but state in general by superposition

$$\psi = a\,|up> + b\,|down> = a\, e_0 + b\, e_1 \quad \text{with} \quad |a|^2 + |b|^2 = 1, \quad e_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad e_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

State of a system with q Qubits (e.g. Spin of q particels):

$| a_1 a_2 \ldots a_q >$ where each a_i can take the value up or down, 0 or 1.

Two Qubit system has 4 possible states |00>, |01>, |10>, and |11>

Three Qubit system has 8 possible states |000>, … |111>.

Quantum System

Basis is given by |000> , … , |111> and each state can be described in this basis as superposition by

a_1 |000> + a_2 |001> + … + a_8 |111> by the numbers $a_1 \ldots a_{8,}$, ||a||=1.

$$|000> = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0>|0>|0>$$

3 Qubits lead to 2^3=8 dimensional space, q Qubits to 2^q dimensional space.

Transformations

Allowed transformations of a Quantum system are described by unitary matrix relative to the vector basis.

Example: Changing the Spin of particle one from up to down

$$\begin{pmatrix}1\\0\\0\\0\\0\\0\\0\\0\end{pmatrix} \rightarrow \begin{pmatrix}0\\1\\0\\0\\0\\0\\0\\0\end{pmatrix} = \begin{pmatrix}0 & 1 & & & & & & \\ 1 & 0 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & & & & 1 & & & \\ & & & & & 1 & & \\ & & & & & & 1 & \\ & & & & & & & 1\end{pmatrix}\begin{pmatrix}1\\0\\0\\0\\0\\0\\0\\0\end{pmatrix}$$

Measuring

Unobserved the Quantum system can have every state

$a_1 |000> + a_2 |001> + \ldots + a_8 |111>$ with $|a_1|^2 + \ldots + |a_8|^2 = 1$.

Hence, transforming the Quantum state from one a starting state in terms of a Quantum algorithm into a final state does parallel computations on all possible vectors $(a_1,\ldots,a_8)^T$ at the same time.

After execution of the Quantum algorithm (unitary transform via gatter) the final state has to be evaluated.
Measuring forces the state to be a basis vector, e.g. |000>.

Problem in Quantum Computing:
Reading and interpretation of the final measuring result.

Famous Algorithms:

Quantum Fourier Transform, Search (Grover), Integer factorization (Shor)

Pauli Matrices

$$P_0 = I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, P_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, P_y = \begin{pmatrix} 0 & i \\ -i & 0 \end{pmatrix}, P_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Are the atoms for describing transformations (algorithms) and for describing the Hamiltonians of a Quantum systems (description of the state).

$$P_0 \otimes P_0 \otimes P_x = \begin{pmatrix} 0 & 1 & & & & & & \\ 1 & 0 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & & & & 1 & & & \\ & & & & & 1 & & \\ & & & & & & 1 & \\ & & & & & & & 1 \end{pmatrix}$$

Compilization

Problem: Find optimal sequence of Quantum transformations that implement a Quantum algorithm ←→
Find short factorization of a unitary matrix (= algorithm) in elementary unitary matrices
(Tensor products of Pauli matrices)

Leads to numerical optimization problem.

Quantum transformation represented by unitary matrices exp(i*H)

Sequence of Quantum transformations by $\exp(i*H_1)*\ldots*\exp(i*H_m) =! \exp(i*H)$

Numerical tasks connected with optimization problem:

Compute $U_j = \exp(i*H_j)$
Compute all products U_1*U_2, $U_1*U_2*U_3$, … $U_1*U_2*\ldots*U_m$

Multiple Matrix Multiplication

Compute $$H_{1,k} = U_1 \cdot U_2 \cdot \cdots \cdot U_k$$

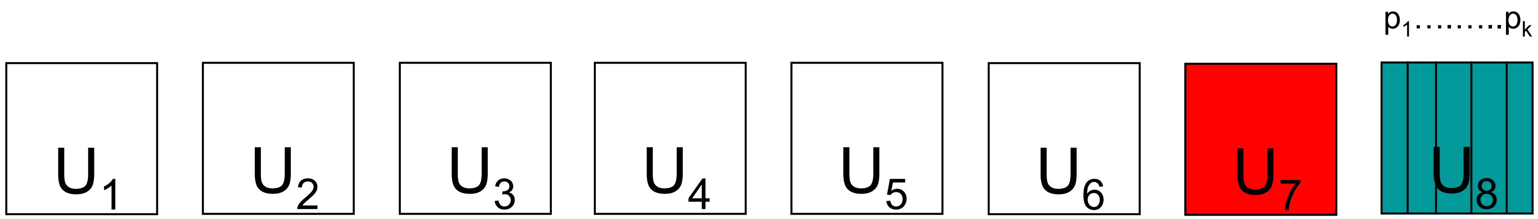
for all k=1,2,…,N

with n x n – matrices $U_1, \dots, U_N$

Total costs sequentially: $N*n^3$

There exist fast matrix-matrix algorithms that are faster than n^3 (Strassen, group-theoretic)
Conjecture: $O(n^{2+\varepsilon})$

Block Column Parallel

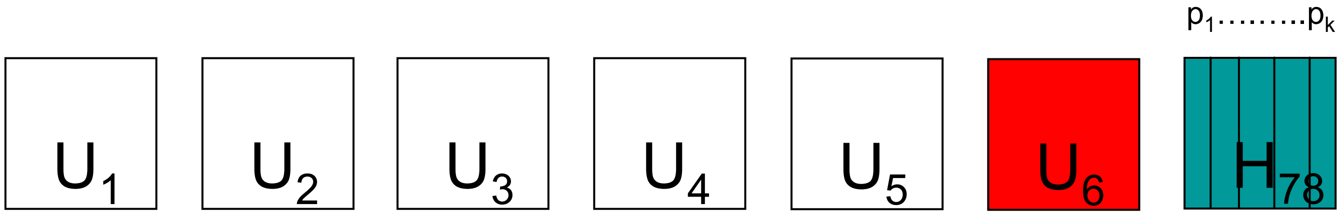


Distribute U_8 on k processors $p_1 \ldots p_k$ together with full U_7 .

$$
\begin{array}{c|c|c|c}
p_1 & p_2 & \cdots & p_k \\
U_7 \cdot U_8(:,1:n_1) & U_7 \cdot U_8(:,n_1+1:n_2) & \cdots & U_7 \cdot U_8(:,n_{k-1}+1:n)
\end{array}
$$

Gives $H_{7,8} = U_7\,U_8$

Block Column Parallel

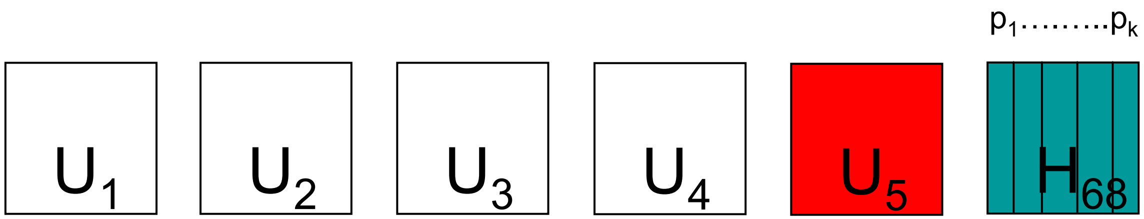


Send full U_6 to all processors $p_1 \ldots p_k$.

p_1	p_2	$\cdots$	p_k
$U_6 \cdot H_{78}(:,1:n_1)$	$U_6 \cdot H_{78}(:,n_1+1:n_2)$	$\cdots$	$U_6 \cdot H_{78}(:,n_{k-1}+1:n)$

Gives $H_{6,8} = U_6\, U_7\, U_8$

Block Column Parallel

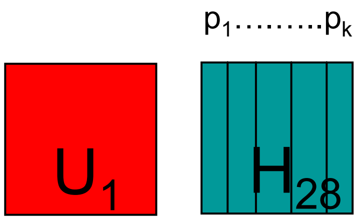


Send full U_5 to all processors $p_1 \ldots p_k$.

$$
\begin{array}{c|c|c|c}
p_1 & p_2 & \cdots & p_k \\
U_5 \cdot H_{68}(:,1:n_1) & U_5 \cdot H_{68}(:,n_1+1:n_2) & \cdots & U_5 \cdot H_{68}(:,n_{k-1}+1:n)
\end{array}
$$

Gives $H_{5,8} = U_5\, U_6\, U_7\, U_8$

Block Column Parallel

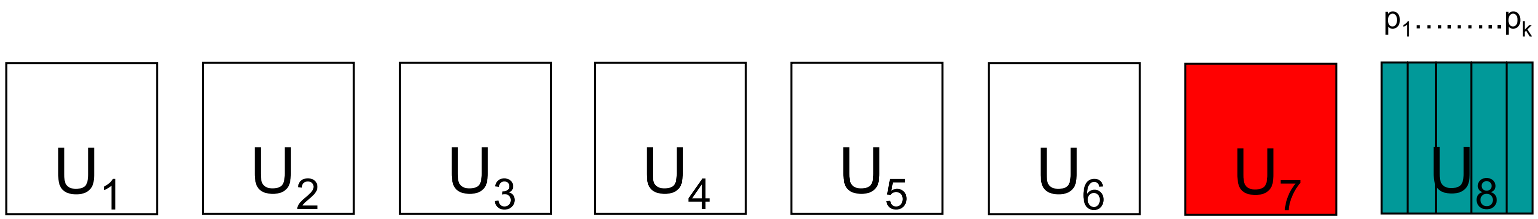


Send full U_1 to all processors $p_1 \ldots p_k$.

$$\begin{array}{c|c|c|c} p_1 & p_2 & \cdots & p_k \\ U_1 \cdot H_{28}(:,1:n_1) & U_1 \cdot H_{28}(:,n_1+1:n_2) & \cdots & U_1 \cdot H_{28}(:,n_{k-1}+1:n) \end{array}$$

Gives $H_{1,8} = U_1 \ldots U_6\, U_7\, U_8$

Costs in Parallel:

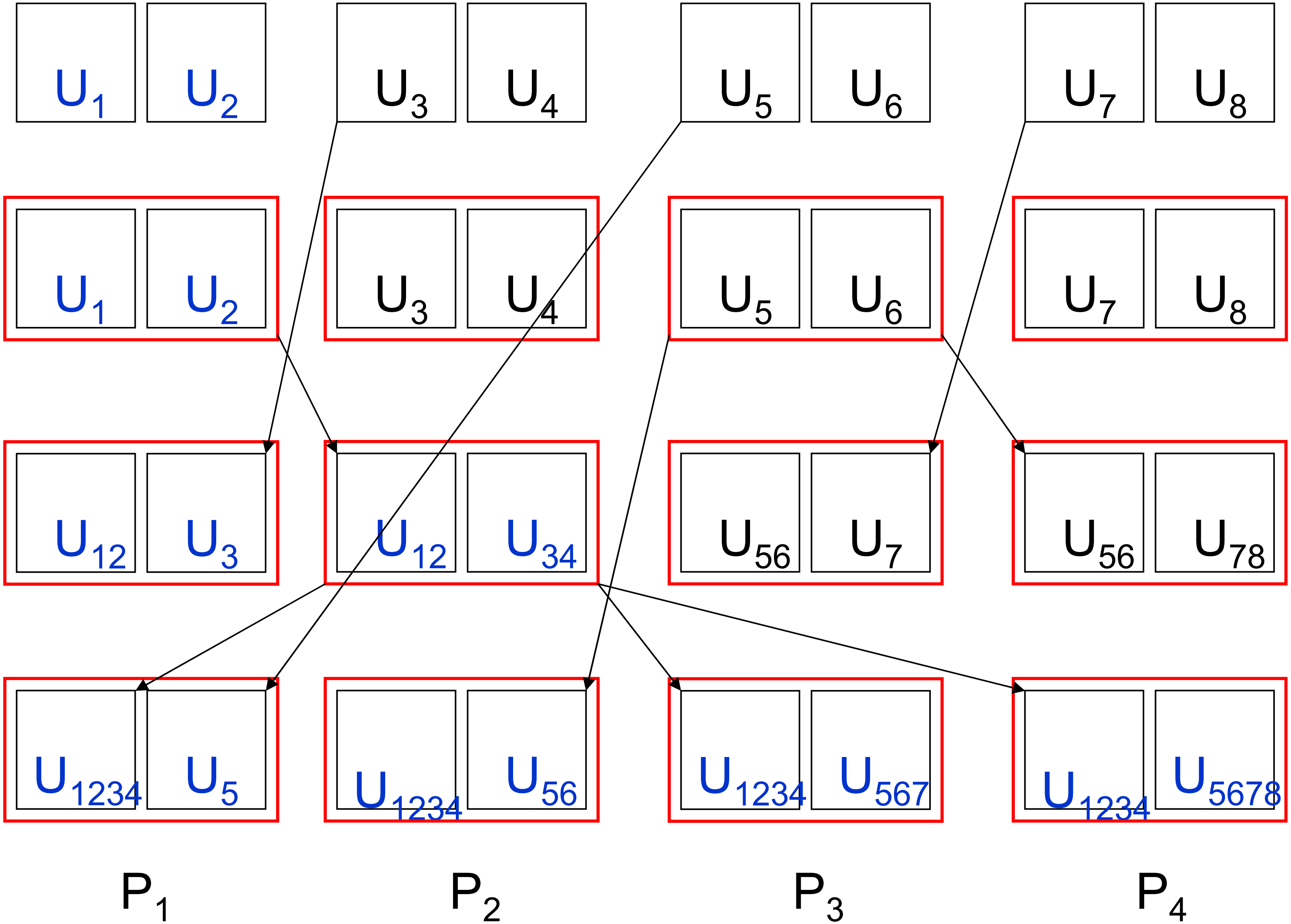


N-1 times $n^2 * n/k$ = $(N-1)*n^3 / k$

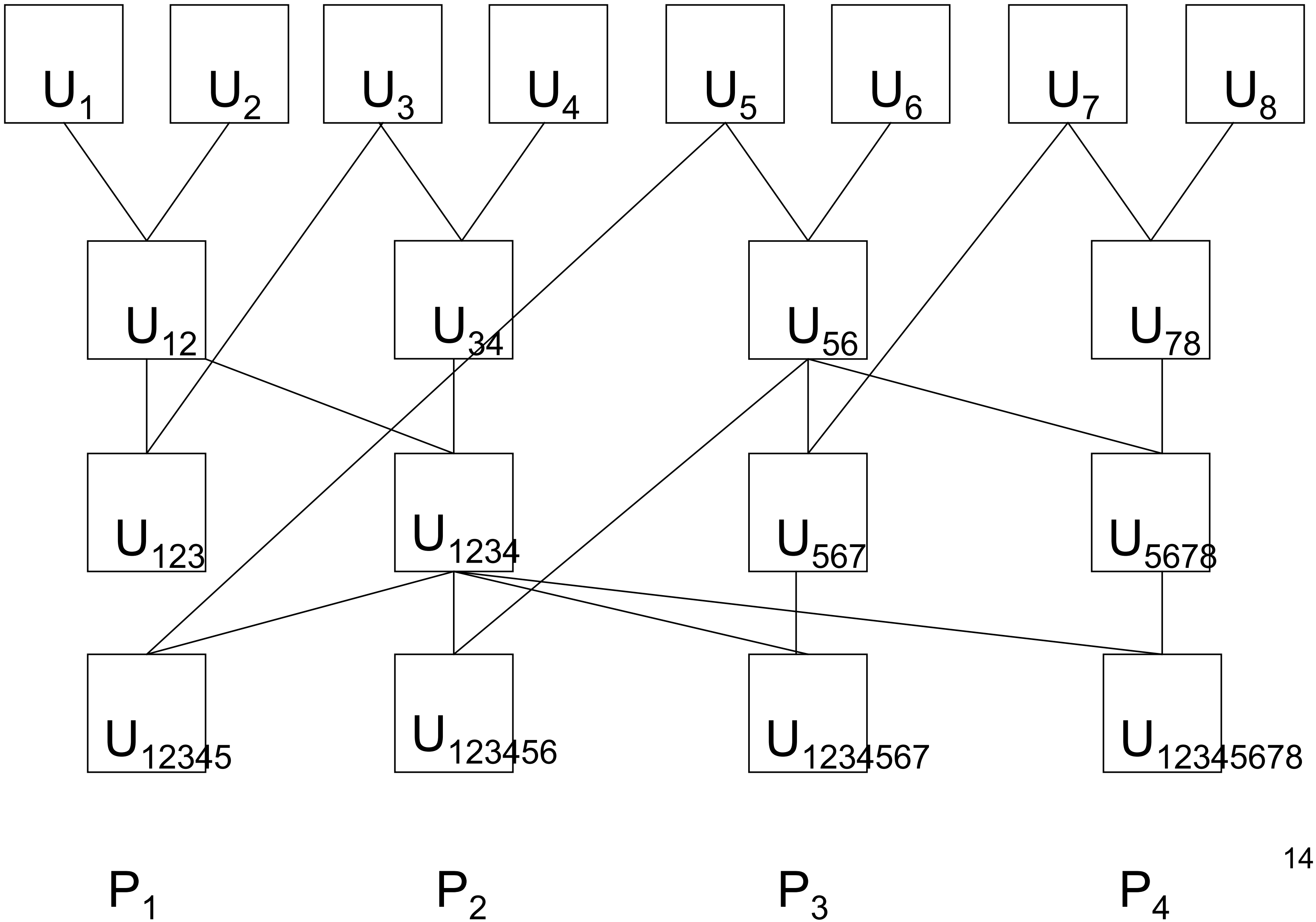
For N matrices of n x n size with k processors.

Especially for 8 matrices and 4 processors: $(7/4)*n^3$

Parallel Prefix Tree



Parallel Prefix Tree



Parallel Prefix Tree

Costs: $\log(N)*n^3$ with N/2 processors

Especially: $3*n^3$

A little bit more expensive than the columnwise method,

but less communication/storage.