

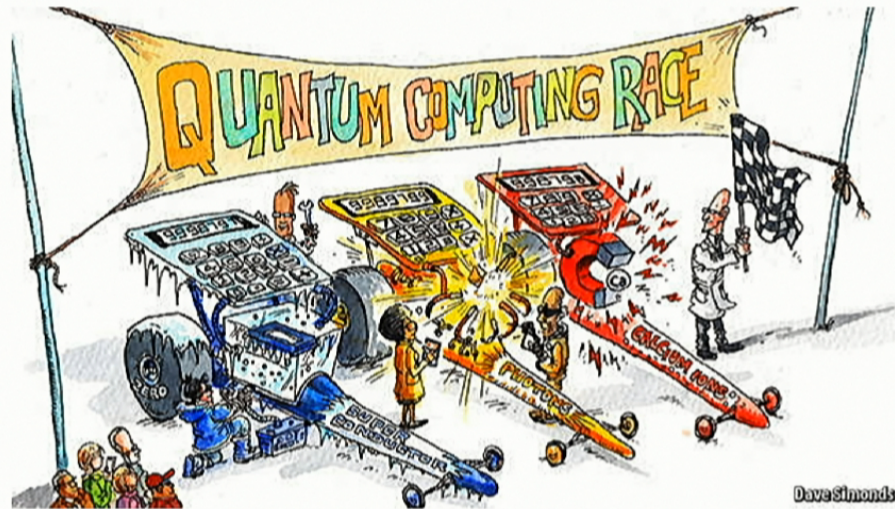
Title: Racing in parallel: Quantum versus Classical

Date: Aug 12, 2016 09:30 AM

URL: <http://pirsa.org/16080019>

Abstract: In a fair comparison of the performance of a quantum algorithm to a classical one it is important to treat them on equal footing, both regarding resource usage and parallelism. We show how one may otherwise mistakenly attribute speedup due to parallelism as quantum speedup. As an illustration we will go through a few quantum machine learning algorithms, e.g. Quantum Page Rank, and show how a classical parallel computer can solve these problems faster with the same amount of resources.

Our classical parallelism considerations are especially important for quantum machine learning algorithms, which either use QRAM, allow for unbounded fanout, or require an all-to-all communication network.



## Racing in parallel: Quantum versus Classical

Damian Steiger and Matthias Troyer

## Why Quantum Machine Learning?

- Improve figures of merit:
  - runtime / complexity
  - generalization error
  - data efficiency

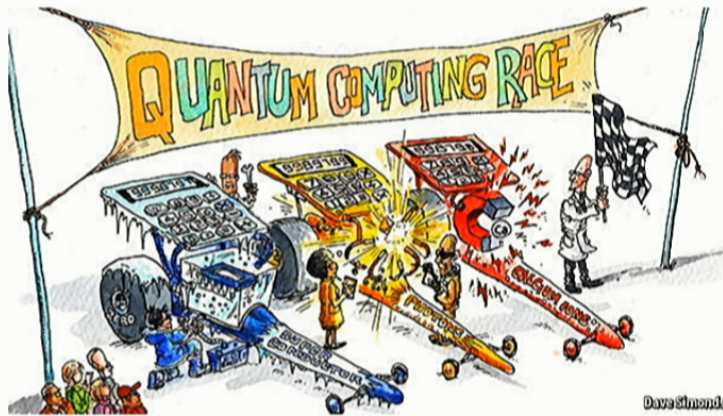
# Why Quantum Machine Learning?

- Improve figures of merit:
  - runtime / complexity
  - generalization error
  - data efficiency
- In this talk I will look at quantum algorithms which claim a speedup

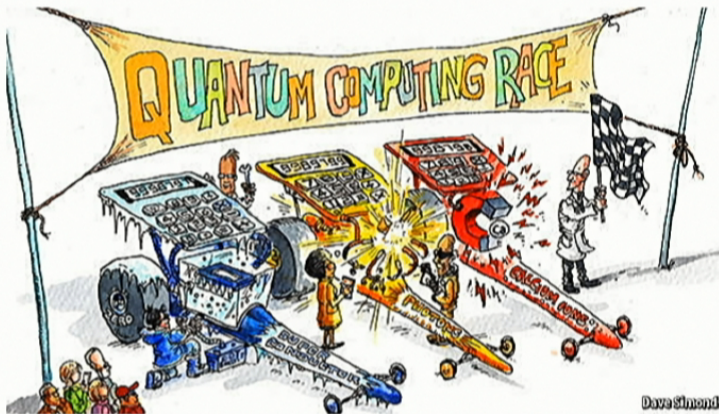
## Contents

- Prerequisite: Fair benchmarking
- All-to-all networks are expensive (Quantum PageRank)
- Matrix-vector multiplication revisited
- Fanout is limited
- QRAM and its implications for speed (Quantum PCA)

## While everyone was looking at Quantum Computers...



## While everyone was looking at Quantum Computers...

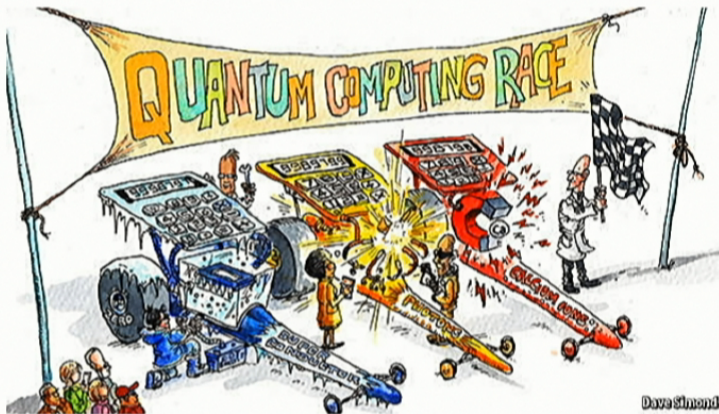


VS



- Single core computers are dead

## While everyone was looking at Quantum Computers...



VS

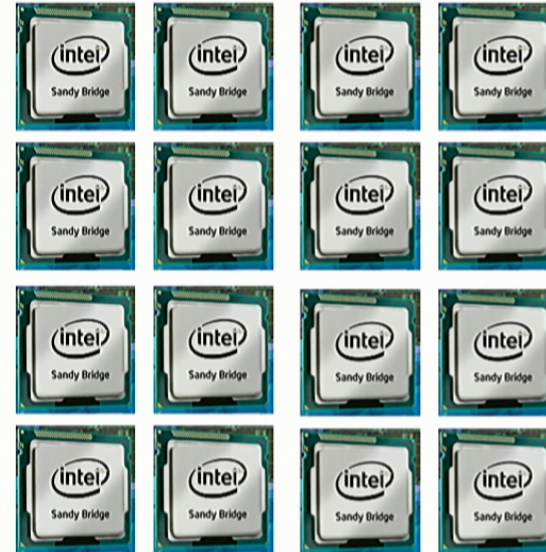


- Single core computers are dead
- Benchmark against special purpose parallel computers

## Fair benchmarking

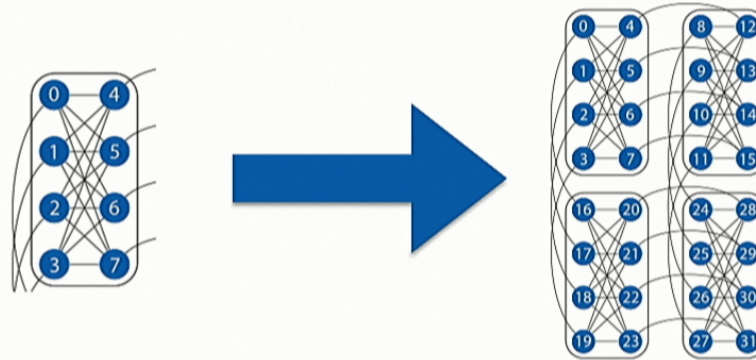


VS



*“Use equal amount of hardware and compare times”*

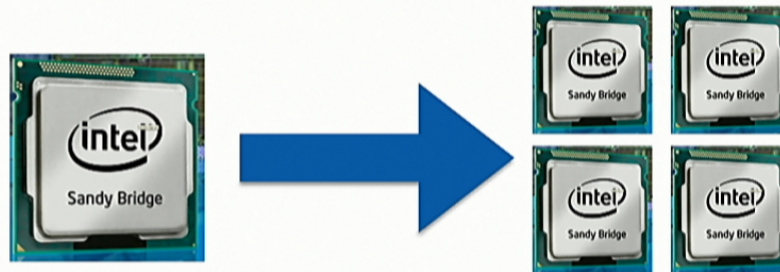
## Hardware scaling



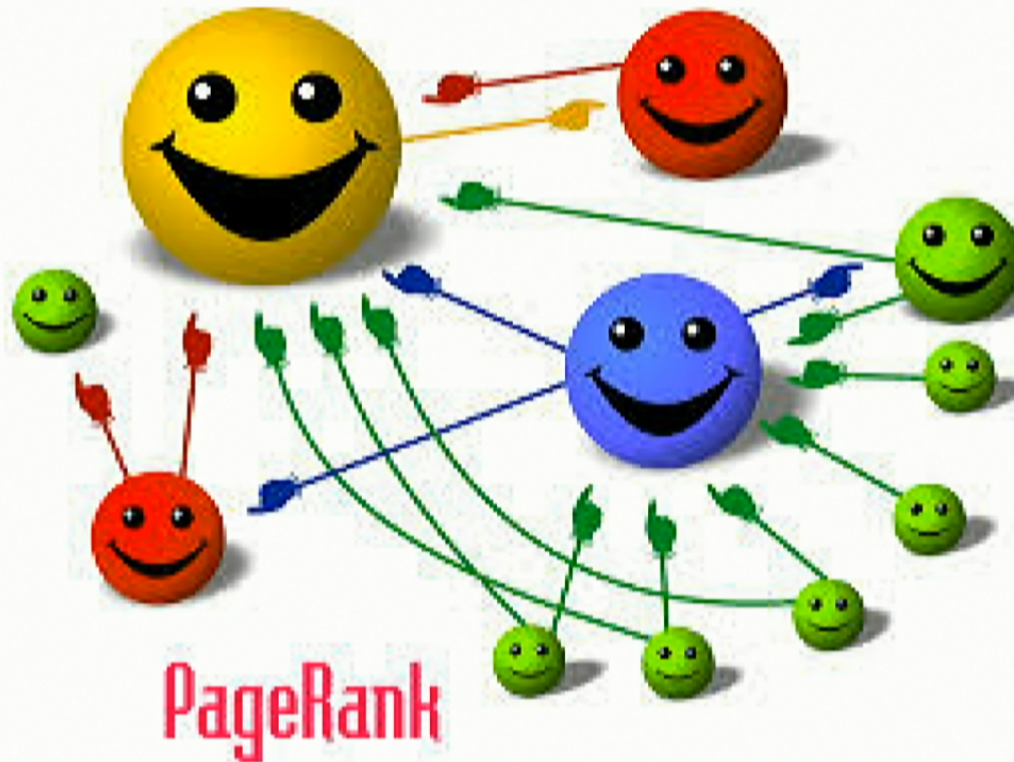
- For a quantum annealer with chimera topology the hardware scales **linearly** in the number of spins

## Hardware scaling

- For a quantum annealer with chimera topology the hardware scales **linearly** in the number of spins
- Hence also scale classical hardware **linearly**



## Quantum PageRank



# All-to-all networks are expensive

PRL 108, 230506 (2012)

PHYSICAL REVIEW LETTERS

week ending  
8 JUNE 2012

## Adiabatic Quantum Algorithm for Search Engine Ranking

Silvano Garnerone,<sup>1,2,5</sup> Paolo Zanardi,<sup>2,5</sup> and Daniel A. Lidar<sup>2,3,4,5</sup><sup>1</sup>*Institute for Quantum Computing, University of Waterloo, Waterloo, ON N2L 3G1, Canada*<sup>2</sup>*Department of Physics & Astronomy, University of Southern California, Los Angeles, California 90089, USA*<sup>3</sup>*Department of Electrical Engineering, University of Southern California, Los Angeles, California 90089, USA*<sup>4</sup>*Department of Chemistry, University of Southern California, Los Angeles, California 90089, USA*<sup>5</sup>*Center for Quantum Information Science & Technology, University of Southern California, Los Angeles, California 90089, USA*

(Received 25 October 2011; published 4 June 2012)

Prepare PageRank vector in  $O(\log(N))$  timeRead out an element in  $O(N^a)$  time for  $0.2 \leq a < 1$ Required Hardware:  $N$  qubits **$O(N^2)$  couplers**

Couplers are changed in parallel

# All-to-all networks are expensive

PRL 108, 230506 (2012)

PHYSICAL REVIEW LETTERS

week ending  
8 JUNE 2012

## Adiabatic Quantum Algorithm for Search Engine Ranking

Silvano Garnerone,<sup>1,2,5</sup> Paolo Zanardi,<sup>2,5</sup> and Daniel A. Lidar<sup>2,3,4,5</sup><sup>1</sup>*Institute for Quantum Computing, University of Waterloo, Waterloo, ON N2L 3G1, Canada*<sup>2</sup>*Department of Physics & Astronomy, University of Southern California, Los Angeles, California 90089, USA*<sup>3</sup>*Department of Electrical Engineering, University of Southern California, Los Angeles, California 90089, USA*<sup>4</sup>*Department of Chemistry, University of Southern California, Los Angeles, California 90089, USA*<sup>5</sup>*Center for Quantum Information Science & Technology, University of Southern California, Los Angeles, California 90089, USA*

(Received 25 October 2011; published 4 June 2012)

	Hardware	Time
<b>Adiabatic Quantum Optimisation (all-to-all connectivity)</b>	$O(N^2)$ Bits $O(N)$ Qubits $O(N^2)$ Couplers	$O(N^a)$ for $0.2 \leq a < 1$

## How to compute PageRank classically

- Monte Carlo Simulation
- Power method:
  - PageRank vector is eigenvector  $\vec{e}_0$  of the stochastic Google Matrix  $G$  with eigenvalue  $\lambda_0 = 1$ . We know that all other eigenvalues  $|\lambda_i| \leq 0.85$
  - Find  $\vec{e}_0$  with accuracy  $\epsilon$  by multiplying a random starting vector  $-\log \epsilon$  times  $G$
  - Matrix vector multiplication takes  $O(N^2)$  time with serial execution

## Comparing to Classical Hardware

	Hardware	Time
<b>Adiabatic Quantum Optimisation (all-to-all connectivity)</b>	$O(N^2)$ Bits $O(N)$ Qubits $O(N^2)$ Couplers	$O(N^a)$ for $0.2 \leq a < 1$
<b>Classical (Serial using power method)</b>	$O(N^2)$ Bits $O(1)$ Multiplier & Adder	$O(N^2 \log(N))$

## Comparing to Classical Hardware

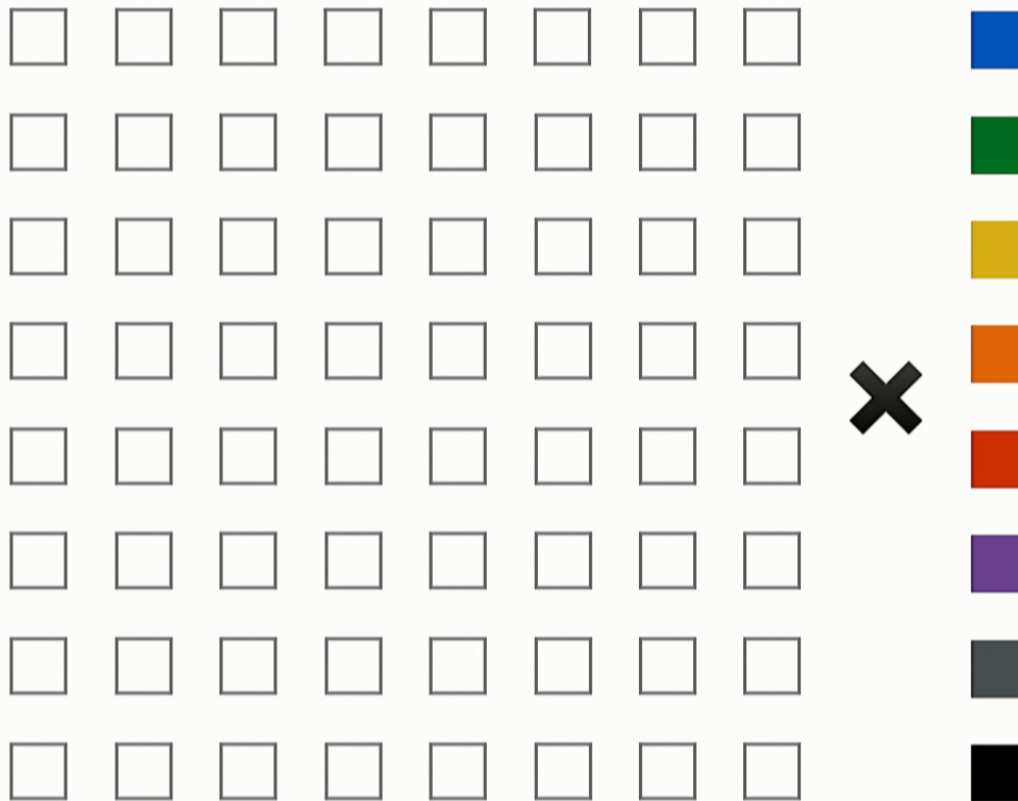
	Hardware	Time
<b>Adiabatic Quantum Optimisation (all-to-all connectivity)</b>	$O(N^2)$ Bits $O(N)$ Qubits $O(N^2)$ Couplers	$O(N^a)$ for $0.2 \leq a < 1$
<b>Classical (Serial using power method)</b>	$O(N^2)$ Bits $O(1)$ Multiplier & Adder	$O(N^2 \log(N))$

**Unfair comparison as not same amount of hardware**

# Matrix-vector multiplication revisited

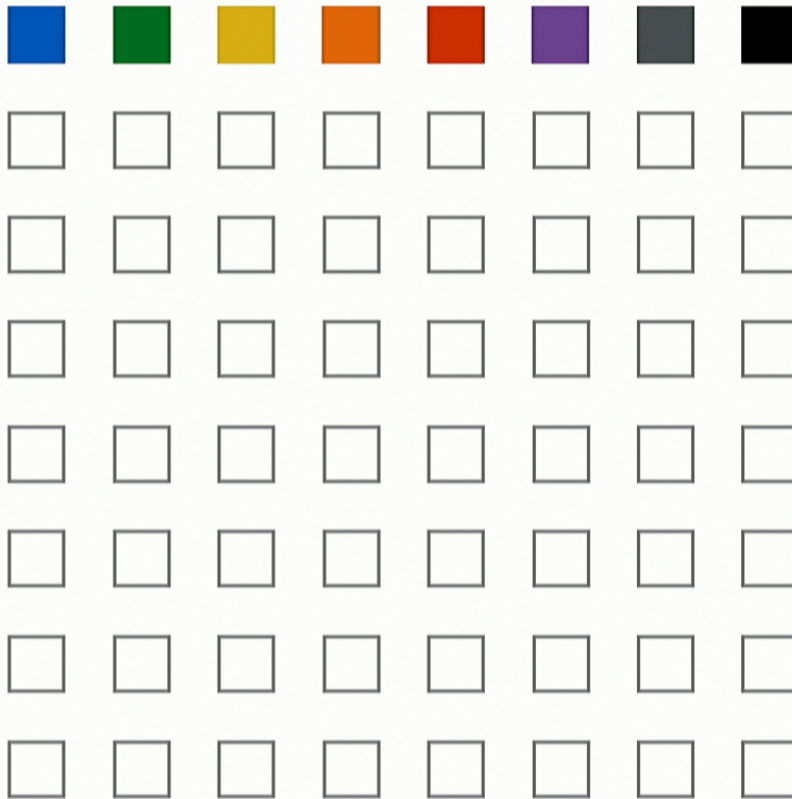
# Matrix Vector Multiplication

☐ matrix element with  
multiply & add unit



# Matrix Vector Multiplication

☐ matrix element with  
multiply & add unit



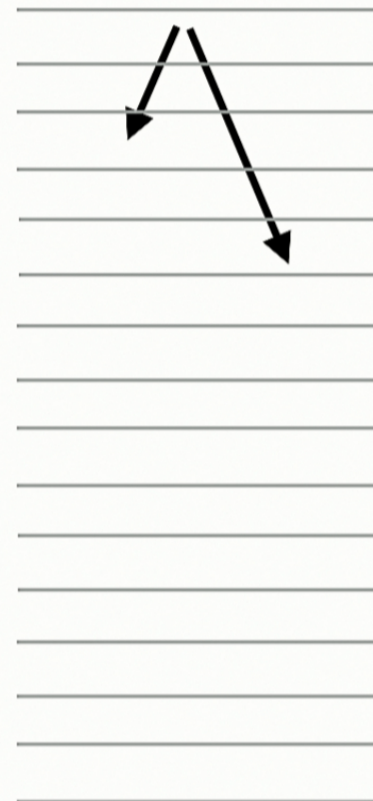
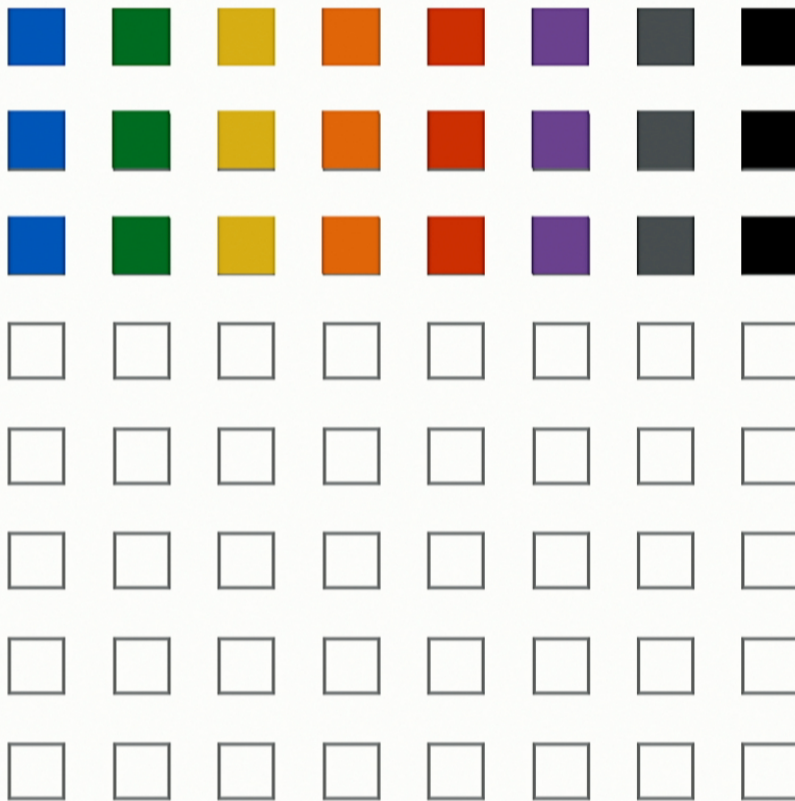
DPHYS

 $O(1)$ 

Damian Steiger |

| 17

## Distribute vector elements



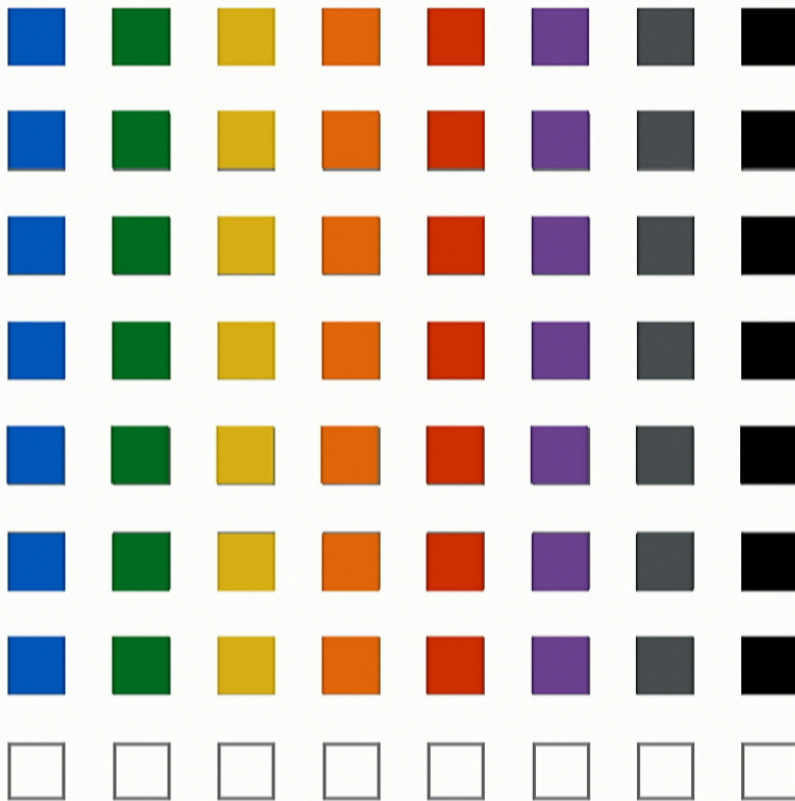
DPHYS

 $O(1)$ 

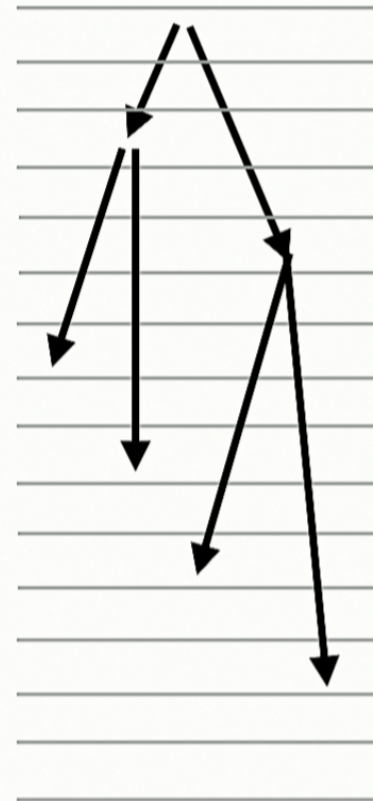
Damian Steiger |

| 18

## Distribute vector elements



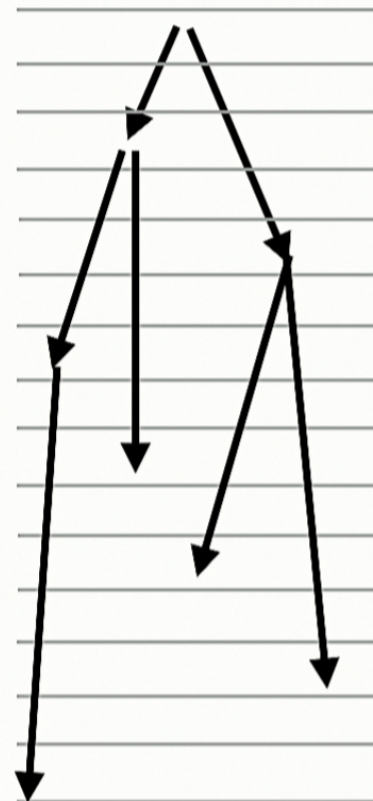
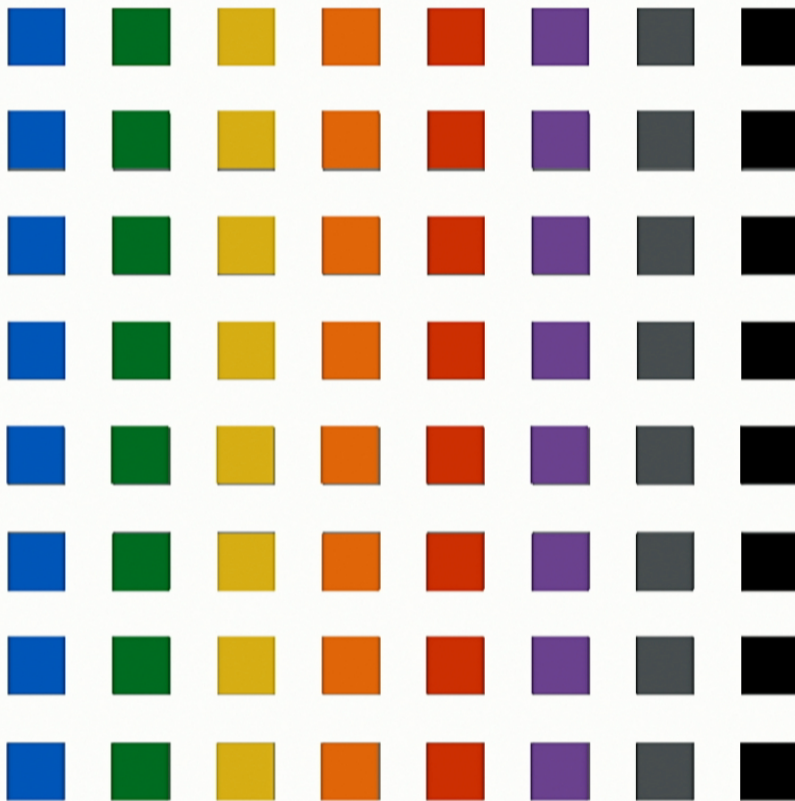
DPHYS

 $O(1)$ 

Damian Steiger |

| 18

## Distribute vector elements



DPHYS

 $O(1) + O(\log(N))$ 

Damian Steiger |

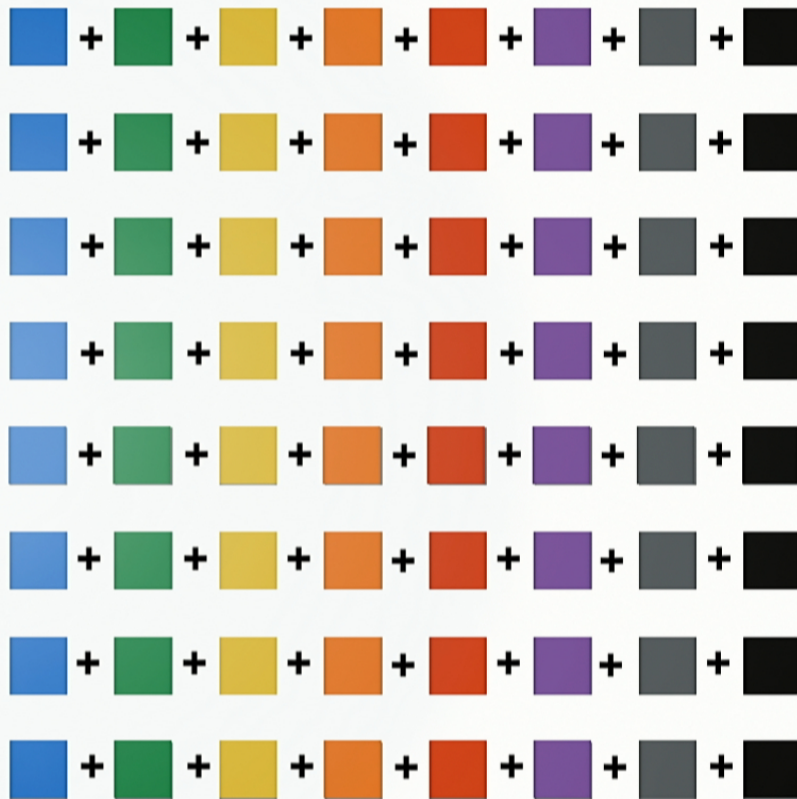
| 18

## Multiply local elements



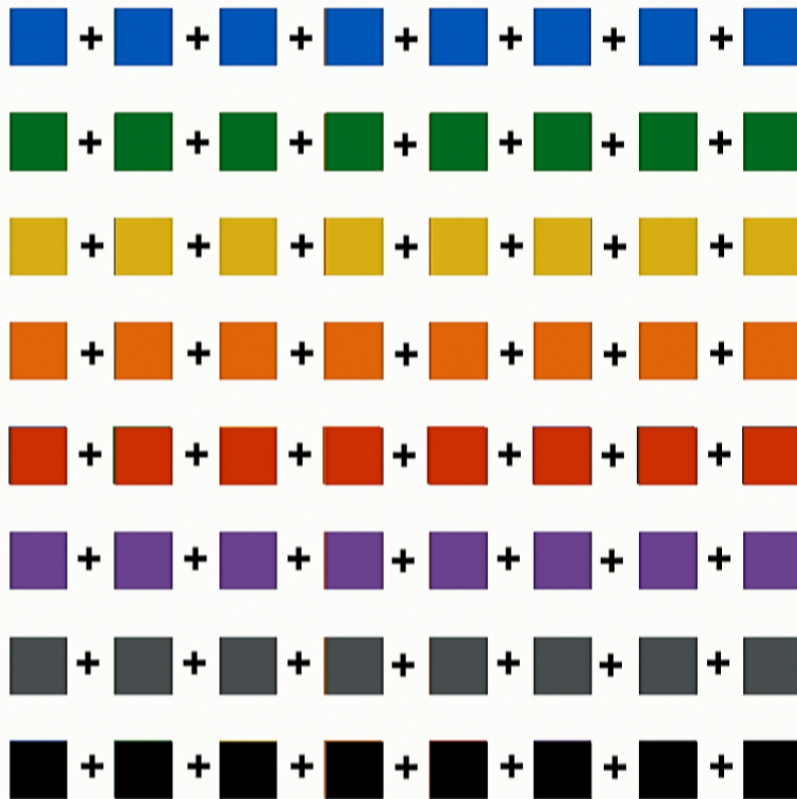
DPHYS  $O(1) + O(\log(N)) + O(1)$

## Add rows

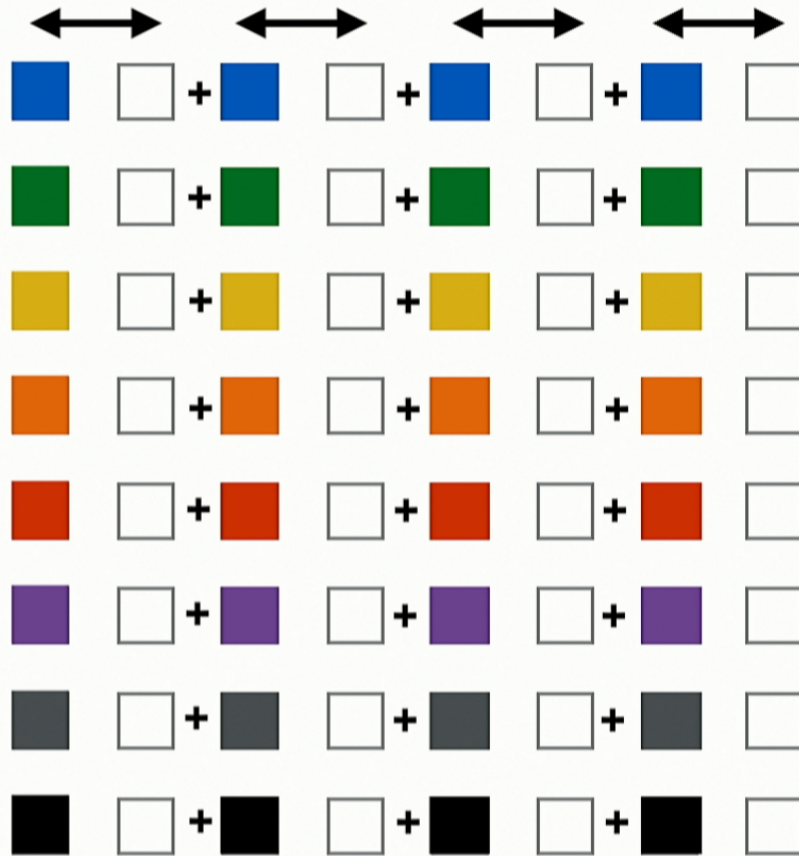


DPHYS  $O(1) + O(\log(N)) + O(1)$

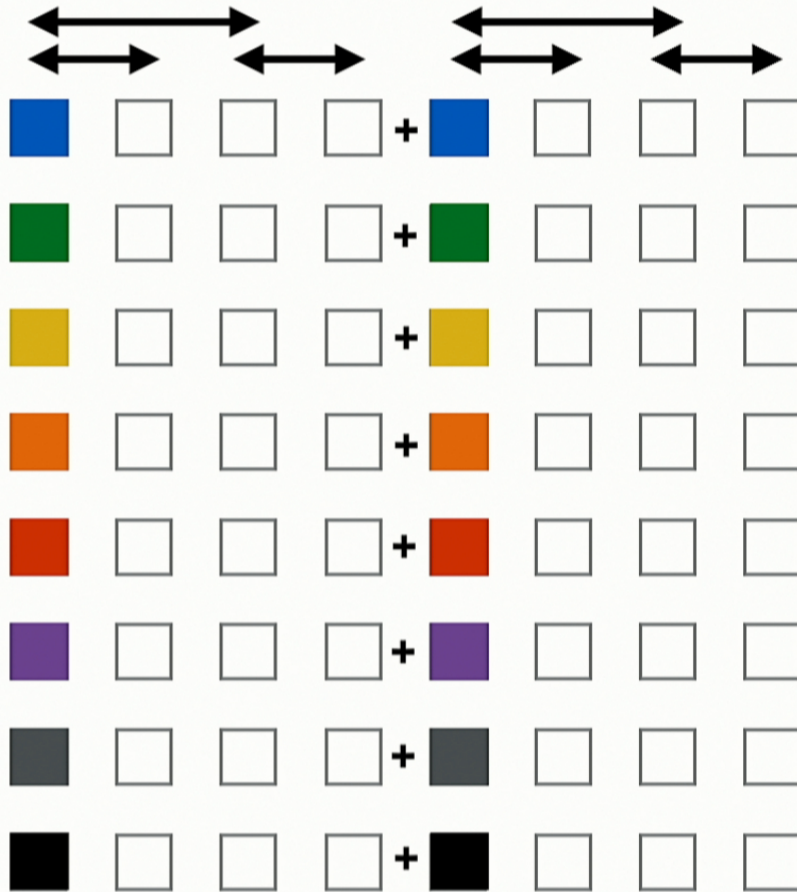
## Add rows



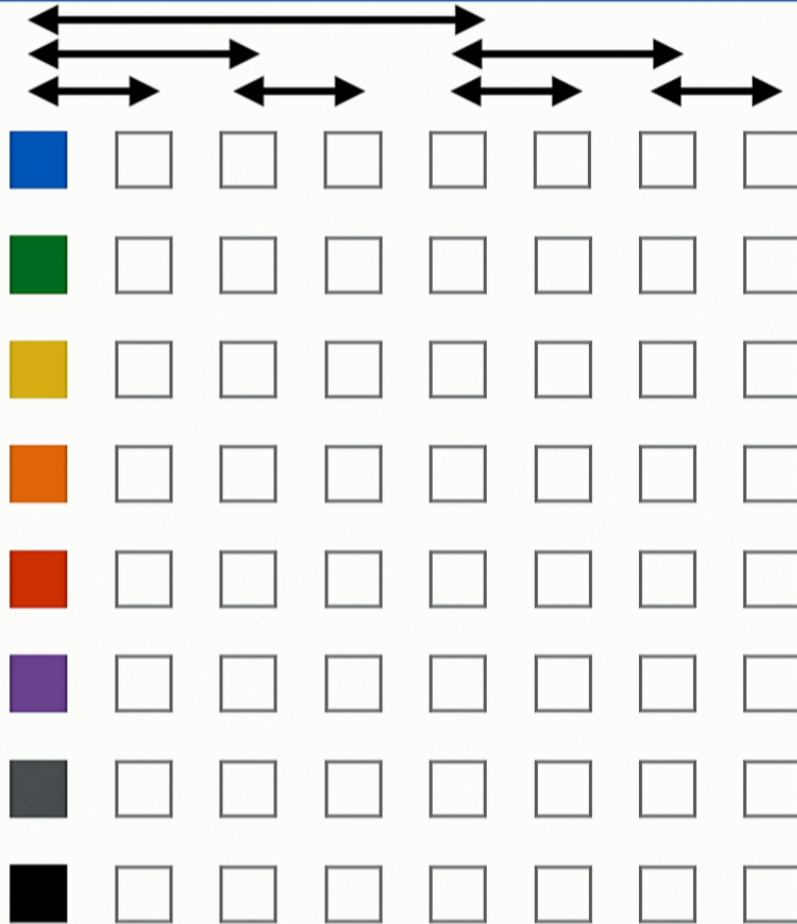
DPHYS  $O(1) + O(\log(N)) + O(1)$



DPHYS  $O(1) + O(\log(N)) + O(1)$

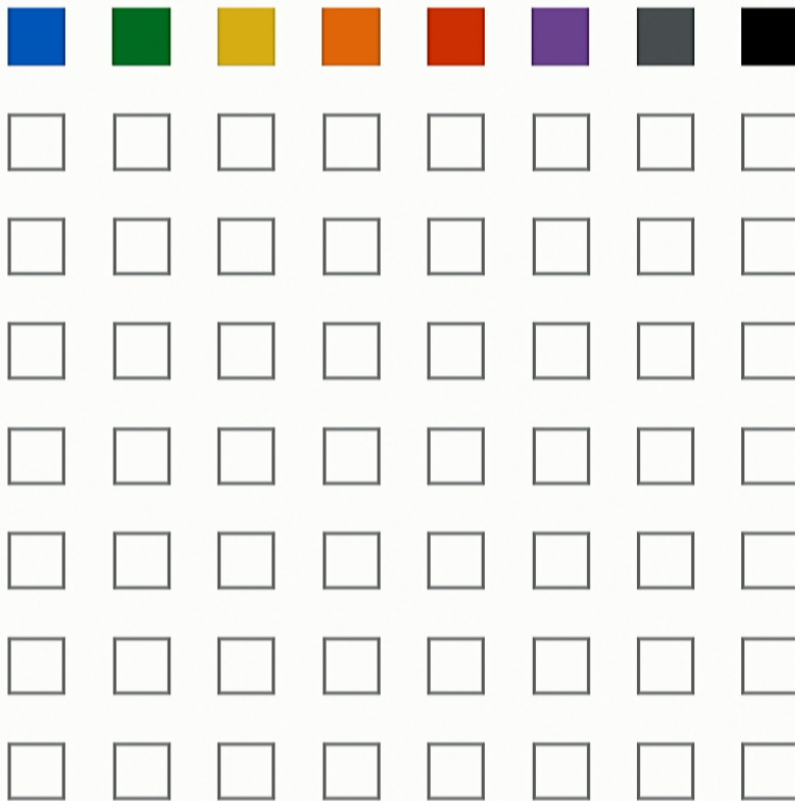


DPHYS  $O(1) + O(\log(N)) + O(1)$



DPHYS  $O(1) + O(\log(N)) + O(1) + O(\log(N))$

## Matrix Vector Multiplication



DPHYS

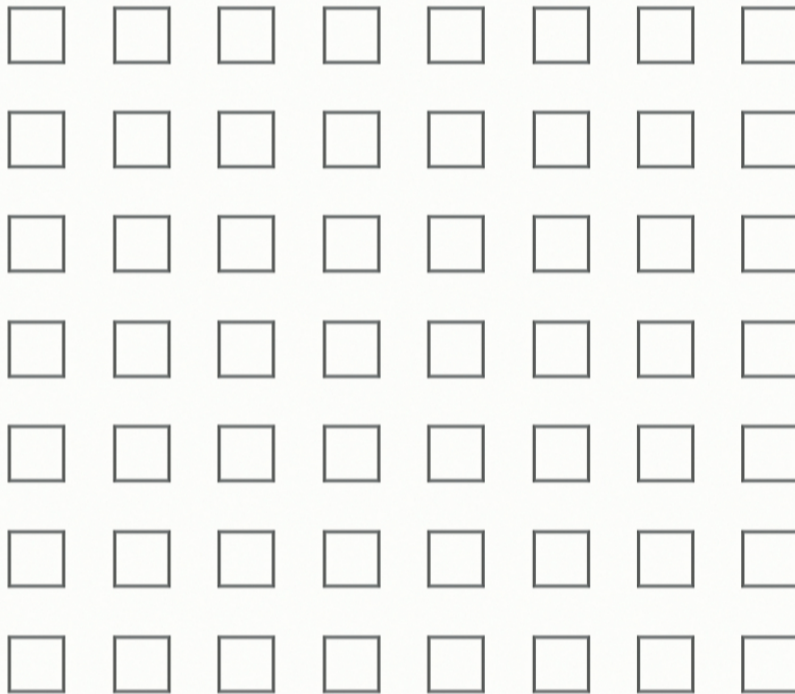
$$O(1) + O(\log(N)) + O(1) + O(\log(N))$$

Damian Steiger |

24

## Matrix Vector Multiplication


 For  $N \times N$  dense matrix:



- Hardware:
  - Tree structure network
  - $O(N^2)$  multiply & add units
- Time:
  - $O(\log(N))$  ignoring speed of light

DPHYS

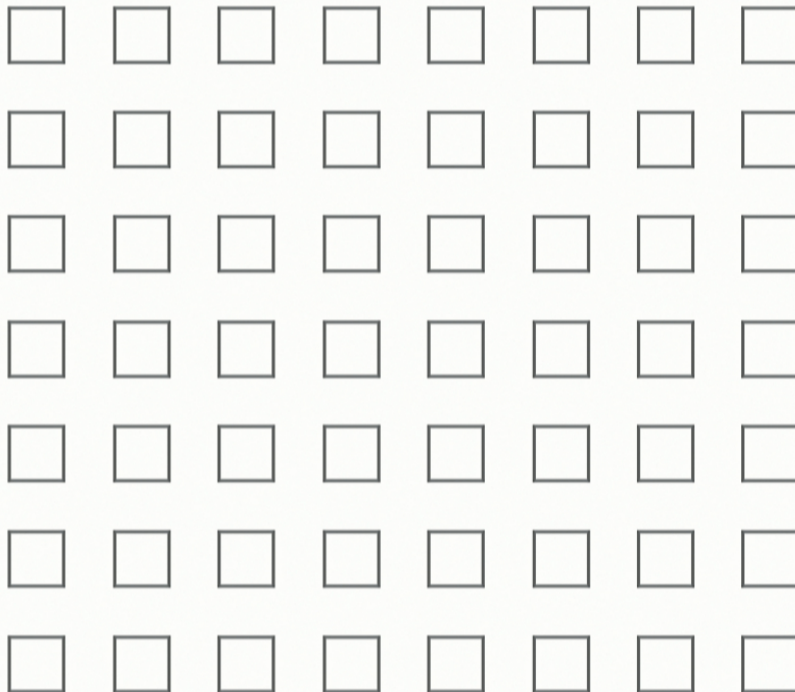
$$O(1) + O(\log(N)) + O(1) + O(\log(N))$$

Damian Steiger |

| 24

## Matrix Vector Multiplication


 For  $N \times N$  dense matrix:



- Hardware:
  - Tree structure network
  - $O(N^2)$  multiply & add units
- Time:
  - $O(\log(N))$  ignoring speed of light

DPHYS

$$O(1) + O(\log(N)) + O(1) + O(\log(N))$$

Damian Steiger |

| 24

## Fanout is limited

- Realistic one-to-all broadcast of vector element to matrix column in 1D
- Example with 16 elements. Step 1:



## Fanout is limited

- Realistic one-to-all broadcast of vector element to matrix column in 1D
- Example with 16 elements. Step 1:



## Fanout is limited

- Realistic one-to-all broadcast of vector element to matrix column in 1D
- Example with 16 elements. Step 3:



## Fanout is limited

- Realistic one-to-all broadcast of vector element to matrix column in 1D
- Example with 16 elements. Step 4:



## Fanout is limited

- Realistic one-to-all broadcast of vector element to matrix column in 1D
- Example with 16 elements. Step 4:



## Fanout is limited

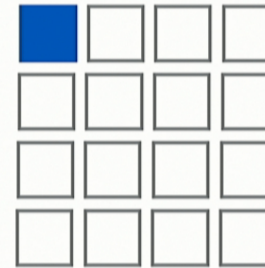
- Realistic one-to-all broadcast of vector element to matrix column in 1D
- Example with 16 elements. Step 4:



- Requires  $O(\log(N))$  time steps
- Element in first step travels distance  $O(N)$ . Hence, speed of light limits the time scaling to  $O(N)$

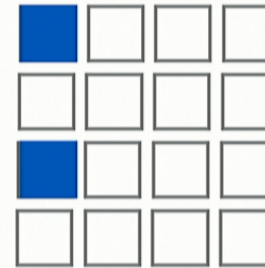
## Fanout is limited

- Realistic one-to-all broadcast of vector element to matrix column in 2D
- Example with 16 elements. Step 1:



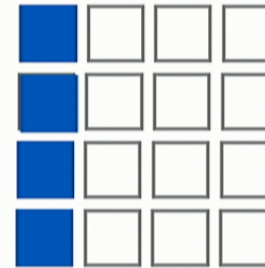
## Fanout is limited

- Realistic one-to-all broadcast of vector element to matrix column in 2D
- Example with 16 elements. Step 1:



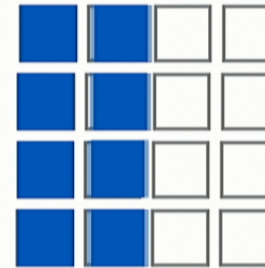
## Fanout is limited

- Realistic one-to-all broadcast of vector element to matrix column in 2D
- Example with 16 elements. Step 2:



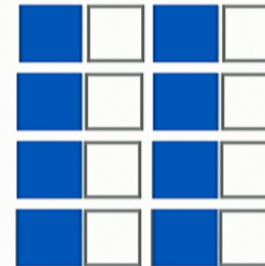
## Fanout is limited

- Realistic one-to-all broadcast of vector element to matrix column in 2D
- Example with 16 elements. Step 3:



## Fanout is limited

- Realistic one-to-all broadcast of vector element to matrix column in 2D
- Example with 16 elements. Step 4:




## Fanout is limited

- Realistic one-to-all broadcast of vector element to matrix column in 2D
- Example with 16 elements. Step 4:



## Fanout is limited

- Realistic one-to-all broadcast of vector element to matrix column in 2D
- Example with 16 elements. Step 4:  

- Requires  $O(\log(N))$  time steps
- Element in first step travels distance  $O(N^{1/2})$ . Hence, speed of light limits the time scaling to  $O(N^{1/2})$

## Back to Quantum PageRank

	Hardware	Time
<b>Adiabatic Quantum Optimisation</b> (all-to-all connectivity)	$O(N^2)$ Bits $O(N)$ Qubits $O(N^2)$ Couplers	$O(N^a)$ for $0.2 \leq a < 1$
<b>Classical</b> (Serial using power method)	$O(N^2)$ Bits $O(1)$ Multiplier & Adder	$O(N^2 \log(N))$

## Back to Quantum PageRank

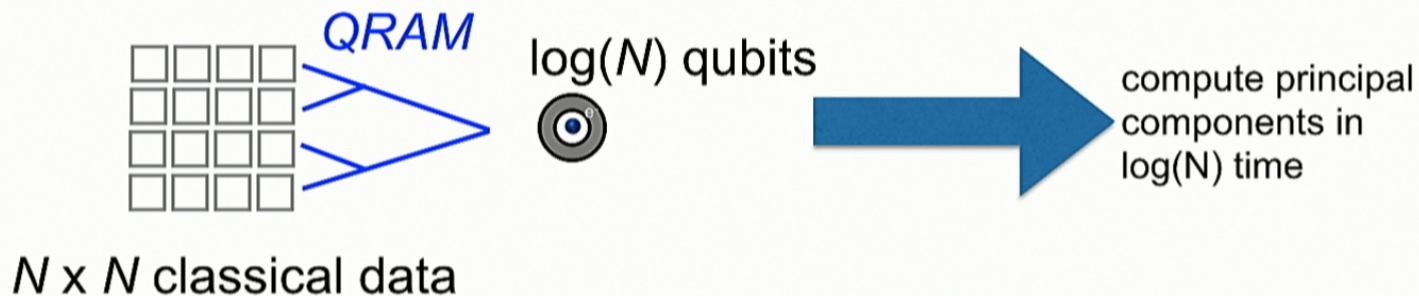
	Hardware	Time
<b>Adiabatic Quantum Optimisation (all-to-all connectivity)</b>	$O(N^2)$ Bits $O(N)$ Qubits $O(N^2)$ Couplers	$O(N^a)$ for $0.2 \leq a < 1$
<b>Classical (tree structure connectivity)</b>	$O(N^2)$ Bits $O(N^2)$ Multiply & Add	$O(\text{polylog}(N))$
<b>Adiabatic Quantum Optimisation (local connectivity)</b>	$O(N^2)$ Bits $O(N^2)$ Qubits $O(N^2)$ Couplers	$O(\exp(N))$
<b>Gate Model Quantum Computer (local connectivity)</b>	$O(N^2)$ Bits $O(N^2)$ Qubits $O(N^2)$ Gates	$O(N^{a+1/3} \log(N))$
<b>Classical (local connectivity)</b>	$O(N^2)$ Bits $O(N^2)$ Multiply & Add	$O(N^{1/3})$

## QRAM and its implications for speed

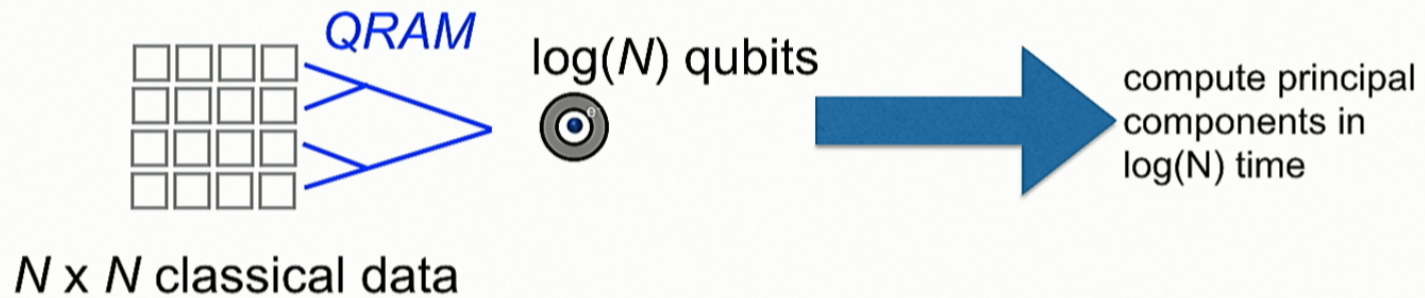
- Quantum principle component analysis in  $\log(N)$  time *by Lloyd, Mohseni, Rebentrost (Nature Physics 2014)*  
Would it work for a **general data matrix**?

## QRAM and its implications for speed

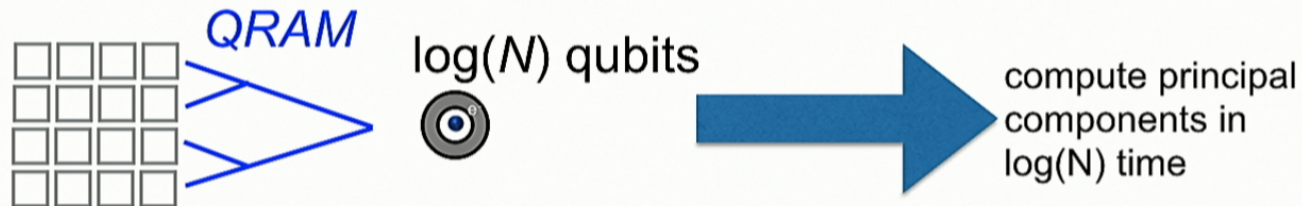
- Quantum principle component analysis in  $\log(N)$  time *by Lloyd, Mohseni, Rebentrost (Nature Physics 2014)*  
Would it work for a **general data matrix**?
- To prepare  $\log(N)$  qubits in a density matrix corresponding to your data matrix  $O(\log(N))$  QRAM is needed, which needs  $O(N^2)$  hardware to implement.



## QRAM and its implications for speed



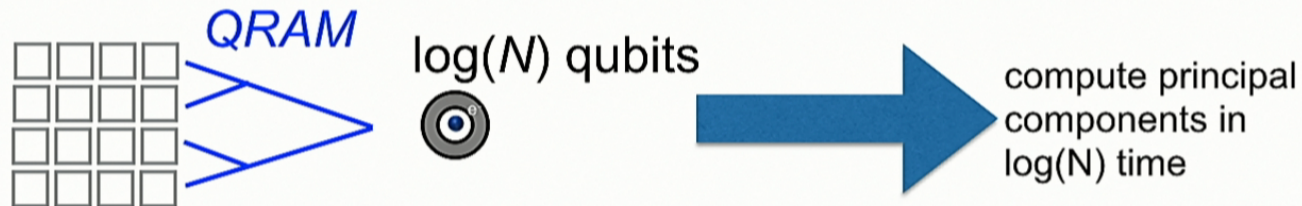
## QRAM and its implications for speed



$N \times N$  classical data

- Classically with  $O(N^2)$  hardware, matrix vector multiplications can be done in  $O(\log(N))$  time. So given the covariance matrix, we can use iterative solvers in  $O(\text{polylog}(N))$

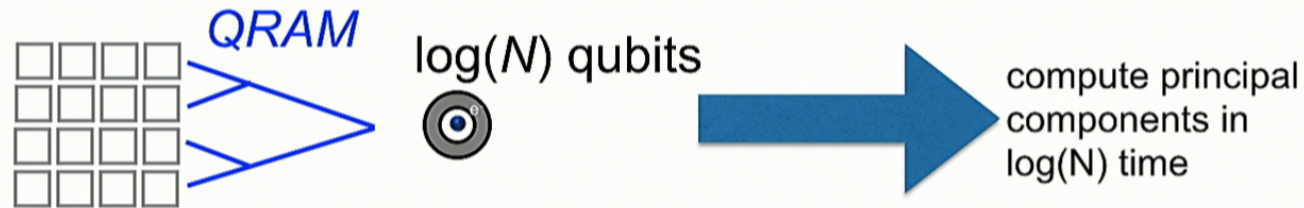
## QRAM and its implications for speed



$N \times N$  classical data

- Classically with  $O(N^2)$  hardware, matrix vector multiplications can be done in  $O(\log(N))$  time. So given the covariance matrix, we can use iterative solvers in  $O(\text{polylog}(N))$
- In  $O(N)$  memory we can store the complete wavefunction of  $\log(N)$  qubits and with additional  $O(N^2)$  hardware, we can emulate the quantum circuit\* in  $O(\text{polylog}(N))$  time

## QRAM and its implications for speed



$N \times N$  classical data

- Using  $O(N^2)$  classical hardware, the exponential speedup disappears for classical data. However, there might be still significant energy savings using the quantum version (depends on implementation of QRAM)

## Summary

- Classical algorithms scale better than Quantum PageRank
- Matrix-vector multiplications in  $O(\log(N))$  time with  $O(N^2)$  hardware
- Careful analysis needed when using QRAM



## Racing in parallel: Quantum versus Classical (soon to be published)

Damian Steiger and Matthias Troyer