

Title: Uni10 Tensor Network Library

Date: Aug 05, 2014 11:00 AM

URL: <http://pirsa.org/14080001>

Abstract:



Uni10

The Universal Tensor Network Library

Yun-Da Hsieh 謝昶達, Ying-Jer Kao 高英哲

Department of Physics
National Taiwan University

<http://www.uni10.org>

Perimeter Institute, Waterloo
2014-08-04

行政院國家科學委員會
National Science Council



Developers

- Yun-Da Hsieh (National Taiwan University)
- Ying-Jer Kao (National Taiwan University)
- Pochung Chen (National Tsing-Hua University)
- Tama Ma (Singapore National University)
- Sukhbinder Singh (Macquarie University)

Graphical Representation

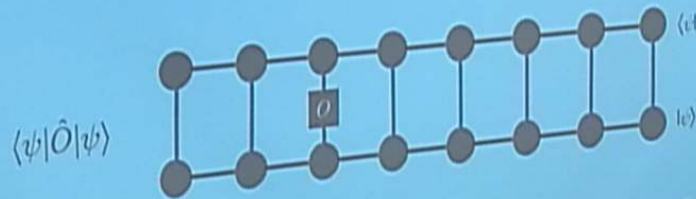


$$A_{\alpha\beta}^{[\sigma_i]}$$



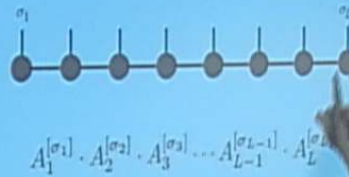
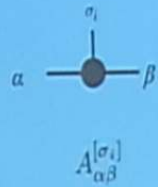
$$A_1^{[\sigma_1]} \cdot A_2^{[\sigma_2]} \cdot A_3^{[\sigma_3]} \dots A_{L-1}^{[\sigma_{L-1}]} \cdot A_L^{[\sigma_L]}$$

- Internal lines are summed over
- External lines are external indices

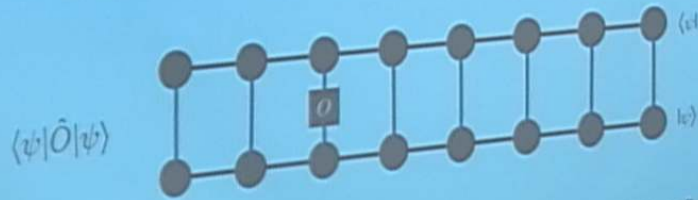


Ulrich Schollwöck, Phil. Trans. R. Soc. A 369, 2643 (2011)

Graphical Representation

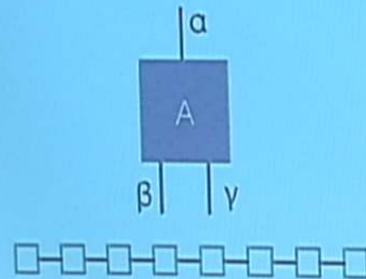


- Internal lines are summed over
- External lines are external indices



Ulrich Schollwöck, Phil. Trans. R. Soc. A 369, 26

Tensor Contraction



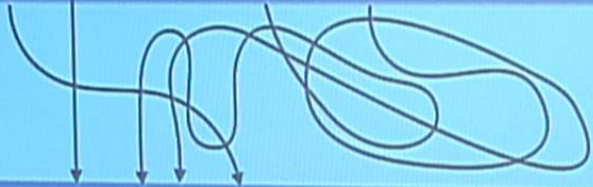
- Bookkeeping tensor indices is tedious and error prone.
- Programming fermionic systems can be complicated
- Minimize computation time and memory usage
- Speed up computation with compute accelerators (GPU, MIC)

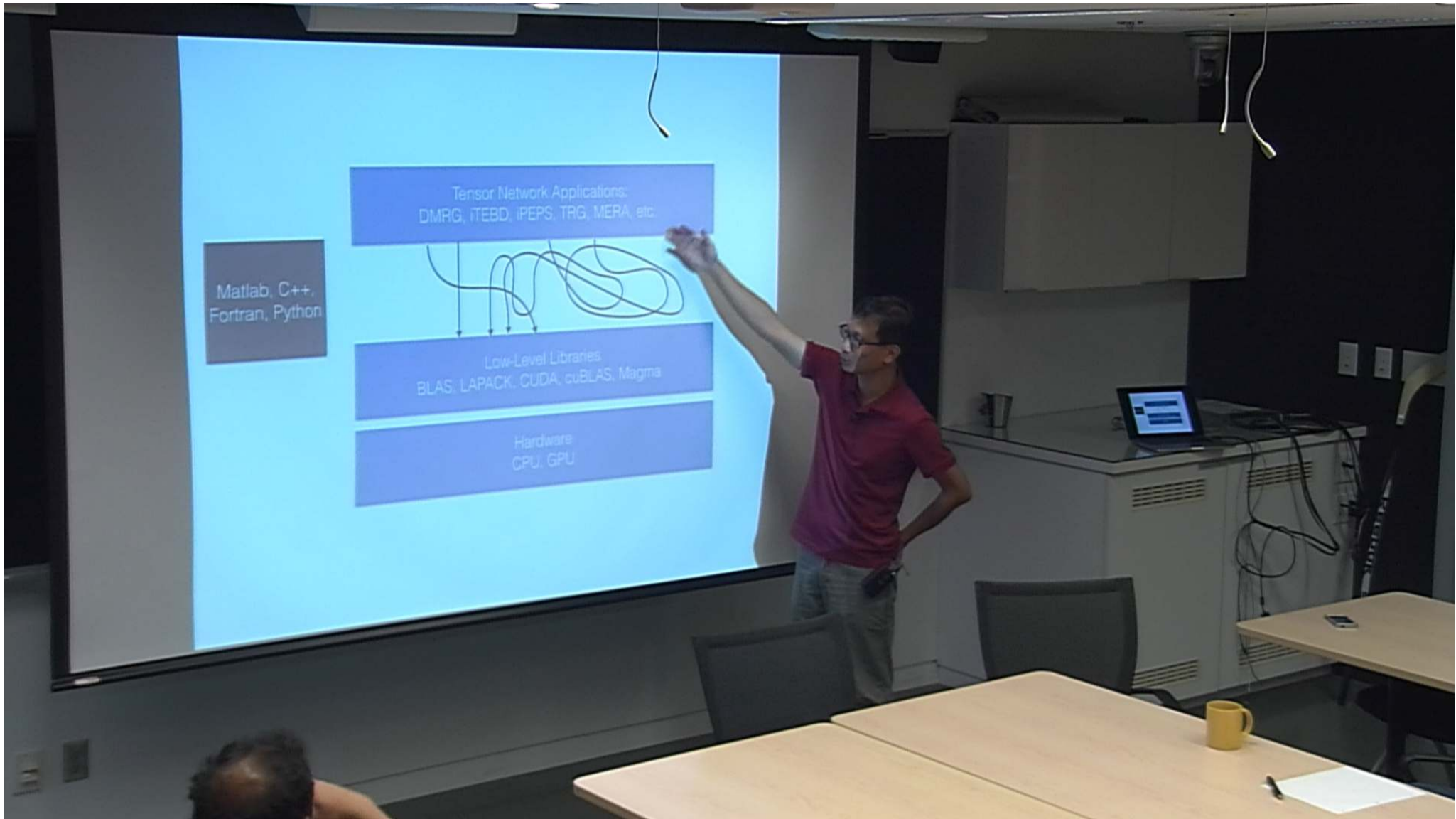
Matlab, C++,
Fortran, Python

Tensor Network Applications:
DMRG, iTEBD, iPEPS, TRG, MERA, etc.

Low-Level Libraries
BLAS, LAPACK, CUDA, cuBLAS, Magma

Hardware
CPU, GPU





Matlab, C++,
Fortran, Python

Tensor Network Applications:
DMRG, iTEBD, iPEPS, TRG, MERA, etc.

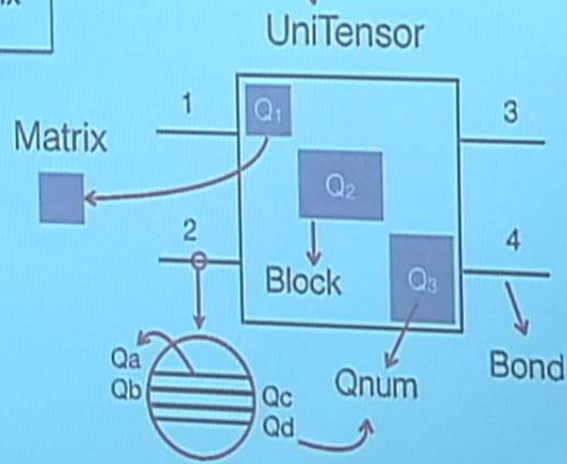
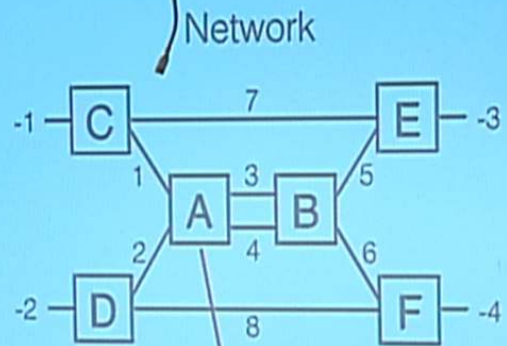
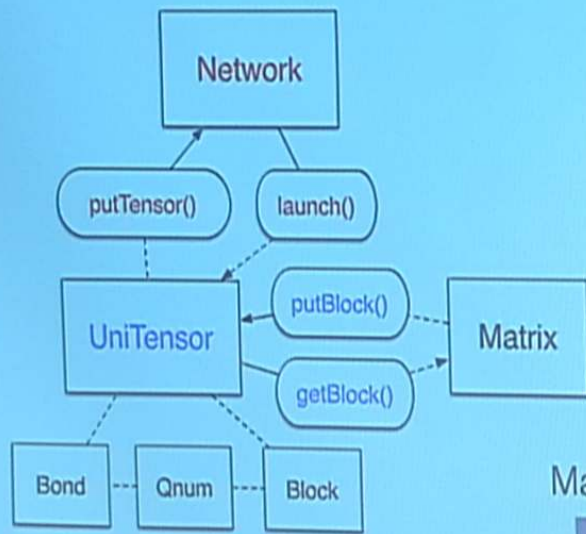
Uni10

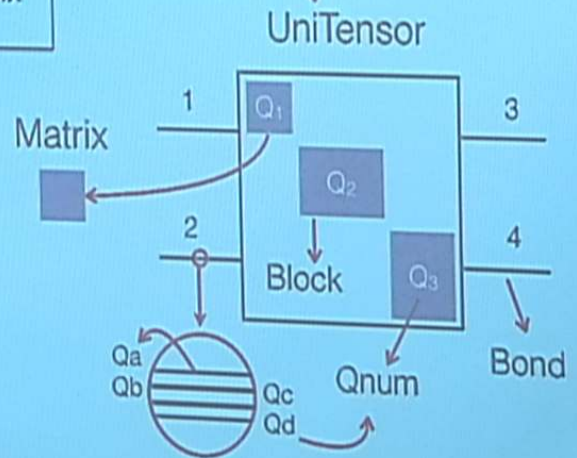
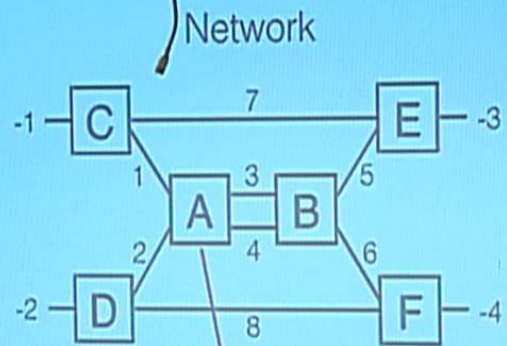
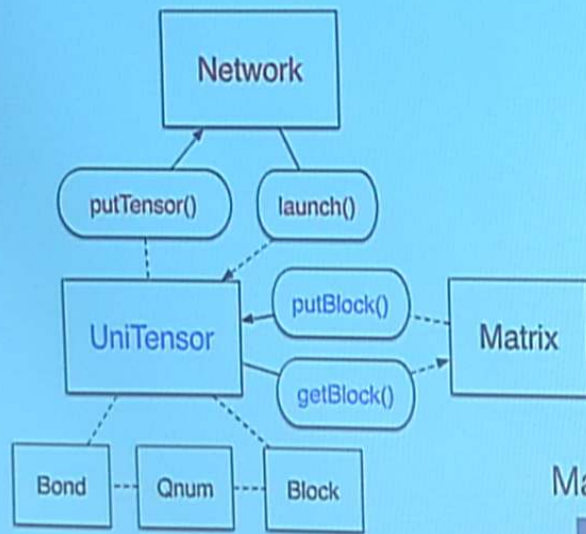
Low-Level Libraries
BLAS, LAPACK, CUDA, cuBLAS, Magma

Hardware
CPU, GPU

Uni10

- Fully implemented in objected-oriented C++
- Aimed toward applications in **tensor network algorithms**
- Provides basic tensor operations with easy-to-use interface
 - A symmetric tensor class **UniTensor** (Abelian symmetry) with auxiliary classes for quantum numbers, **Qnum**, blocks **Block** and bond labels, **Bond** and functions performing tensor operations.
 - A network class **Network**, where details of the graphical representations of the networks are processed and stored.
 - An **engine** to construct and analyze the **contraction tree** for a given network.
 - A **heuristic algorithm** to search for an optimal **binary contraction order** based on the computational and memory constraints.
- Provides wrappers for **Matlab** and **Python** (work in progress).
- Supports acceleration with Nvidia Cuda based **GPU**.
- Open source LGPL with cite-me license

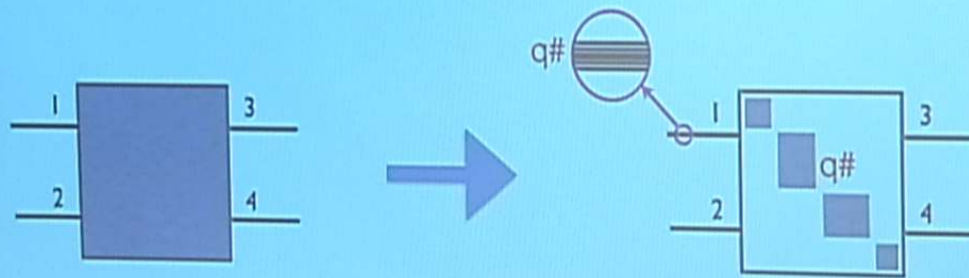




Spin-1 Heisenberg Model

$$H = S_1 S_2$$

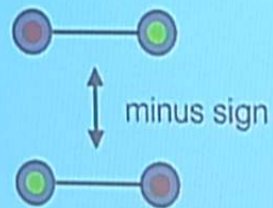
| | | | | | | | | |
|---|---|----|---|---|---|----|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | -1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | -1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |



Symmetries

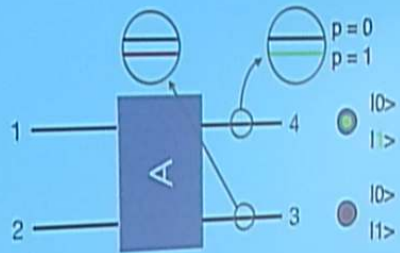
- U_1
 - Total S_z
 - Total particle number
- Z_2
 - The parity of some particle number
 - Key to fermionic system

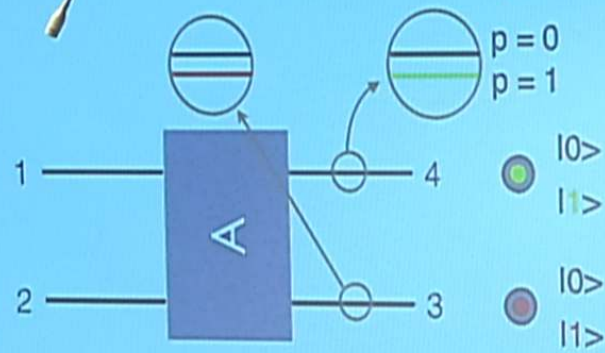
Fermionic system



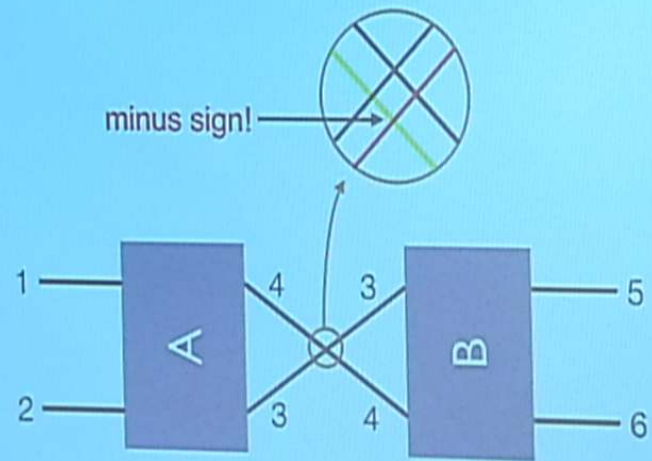
$$C_1^\dagger C_2^\dagger |0\rangle = -C_2^\dagger C_1^\dagger |0\rangle$$

$$|1, 1\rangle = -|1, 1\rangle$$





| | $0,0\rangle$ | $0,1\rangle$ | $1,0\rangle$ | $1,1\rangle$ |
|--------------|--------------|--------------|--------------|--------------|
| $0,0\rangle$ | 1 | X | X | 2 |
| $0,1\rangle$ | X | 3 | 4 | X |
| $1,0\rangle$ | X | 5 | 6 | X |
| $1,1\rangle$ | 7 | X | X | 8 |



$$T_a * T_b$$

Matrix

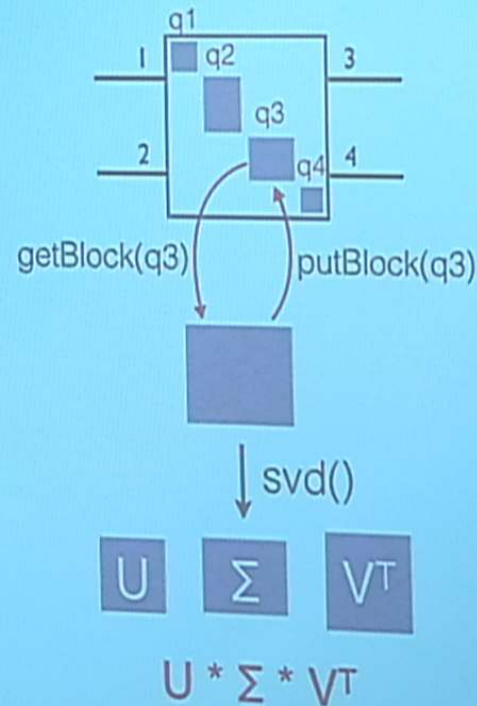
UniTensor & Matrix

- Interactions:

- `T.qnums();`
- `T.getBlock(q3);`
- `T.putBlock(q3);`

- Matrix operations:

- `M.diagonalize();`
- `M.svd();`
- $M_c = M_a * M_b;$
- `M.transpose();`



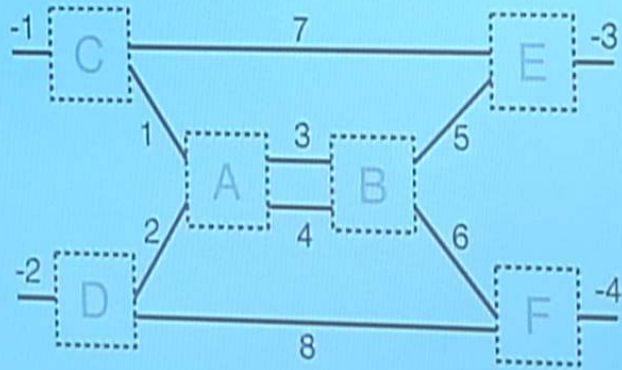
A large, bright blue rectangular screen is the central focus of the image. The word "Network" is written in a simple, grey, sans-serif font in the center of the screen. Two thin black cables hang from the top edge of the screen. The screen is set against a dark background, possibly a stage or a lecture hall.

Network

Network

A file for connections("demo.txt"):

C: -1; 7 1
D: -2; 2 8
A: 1 2; 3 4
B: 3 4; 5 6
E: 7 5; -3
F: 6 8; -4
ABCDEF



Network net("demo.txt")

Network

A file for connections("demo.txt"):

C: -1; 7 1

D: -2; 2 8

A: 1 2; 3 4

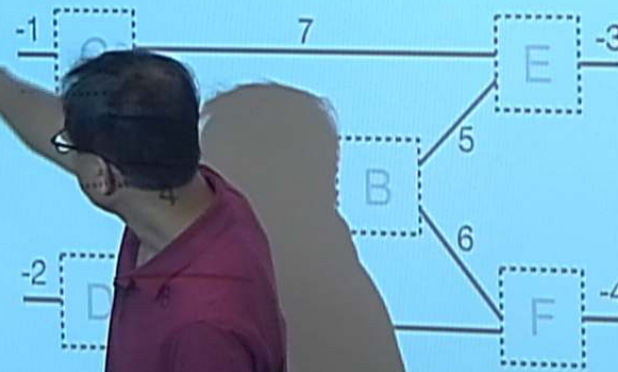
B: 3 4; 5 6

E: 7 5; -3

F: 6 8; -4

ABCDEF

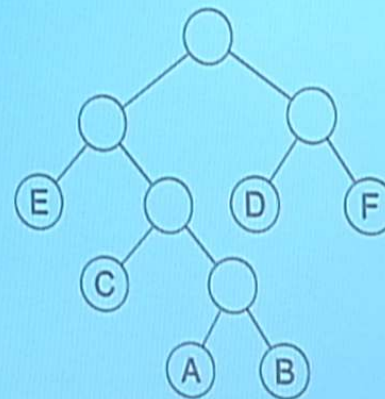
Network net "demo"



Network - launch

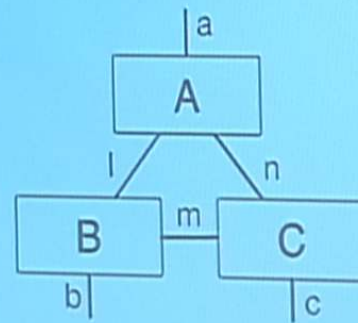
CDABEF

C: -1; 7 1
D: -2; 2 8
A: 1 2; 3 4
B: 3 4; 5 6
E: 7 5; -3
F: 6 8; -4



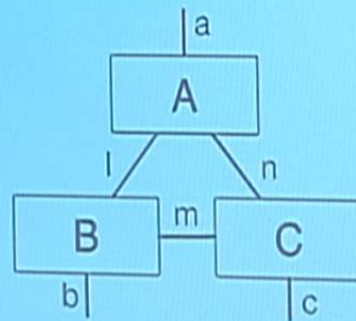
$((A * B) * C) * E * (D * F)$

$$D_{abc} = \sum_{lmn} A_{aln} B_{bml} C_{cnm}$$

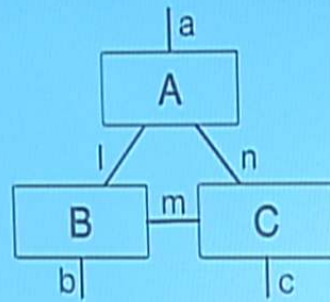


Cost: a b c | m

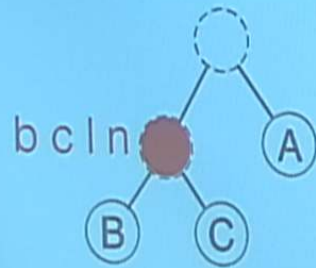
$$D_{abc} = \sum_{lmn} A_{aln} B_{bml} C_{cnm}$$



Cost: a b c l m n



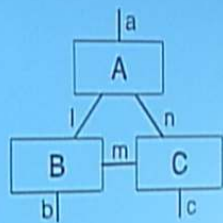
pairwise contractions



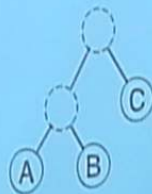
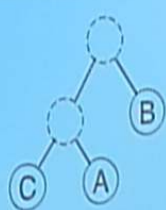
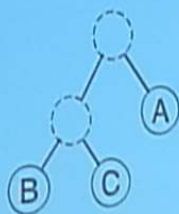
Cost: $b c l m n + a b c l n$ (pairwise)

Cost: $a b c l m n$ (direct sum)

pairwise is better if: $(1 / a + 1 / m) < 1$



pairwise contractions

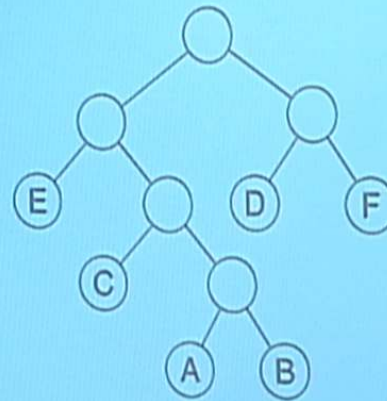


| | | | |
|-------|---------------|---------------|---------------|
| Time | $(1/a + 1/m)$ | $(1/b + 1/n)$ | $(1/c + 1/l)$ |
| Space | $b + l n$ | $a + l m$ | $a + b m n$ |

Network - launch

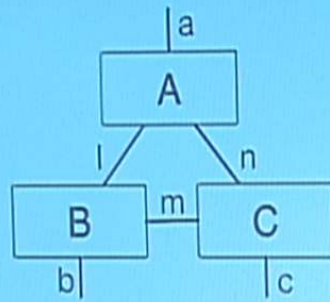
CDABEF

C: -1; 7 1
D: -2; 2 8
A: 1 2; 3 4
B: 3 4; 5 6
E: 7 5; -3
F: 6 8; -4

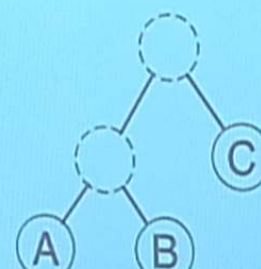
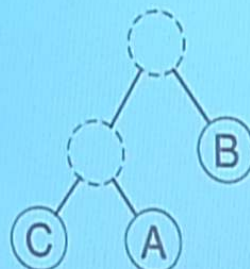
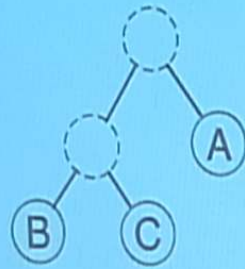


ORDER: (((A * B) * C) * E) * (D * F)

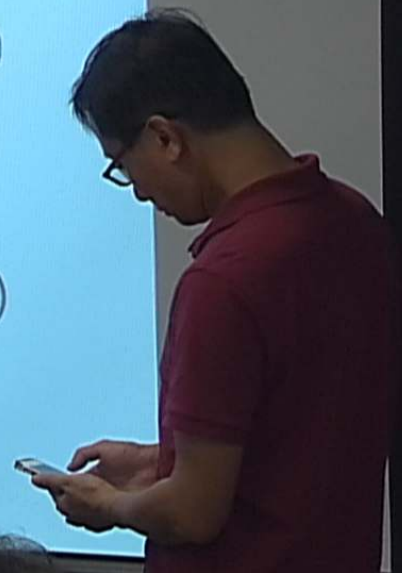
net.launch()



pairwise contractions

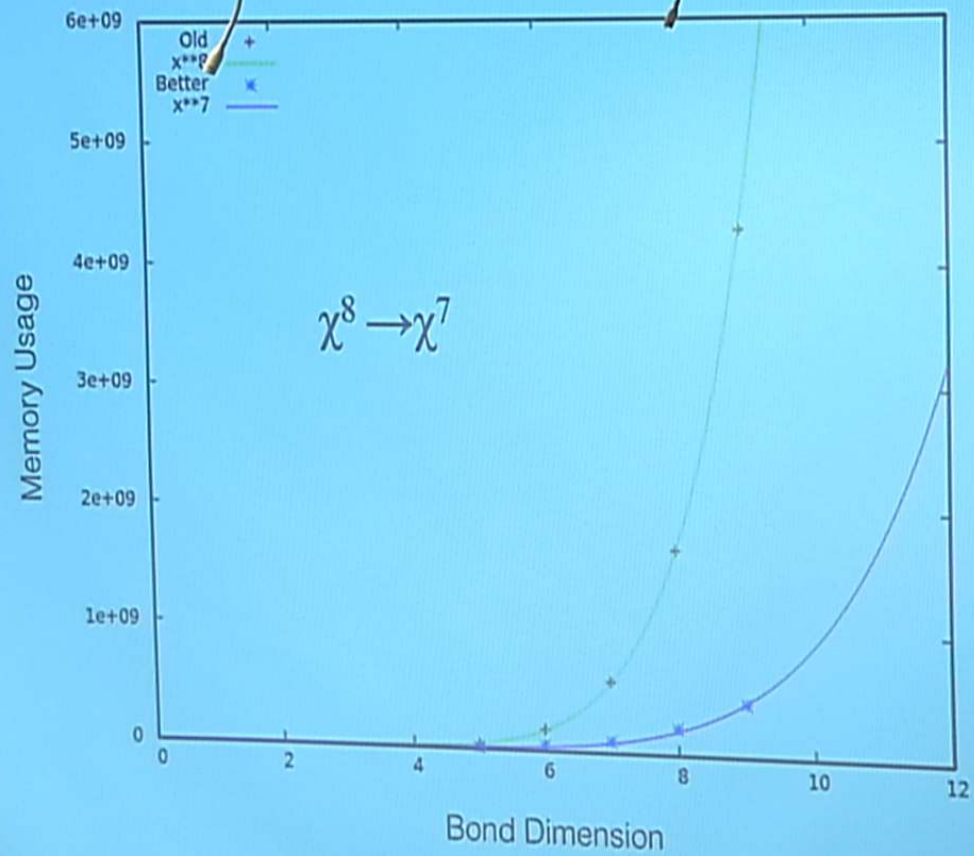


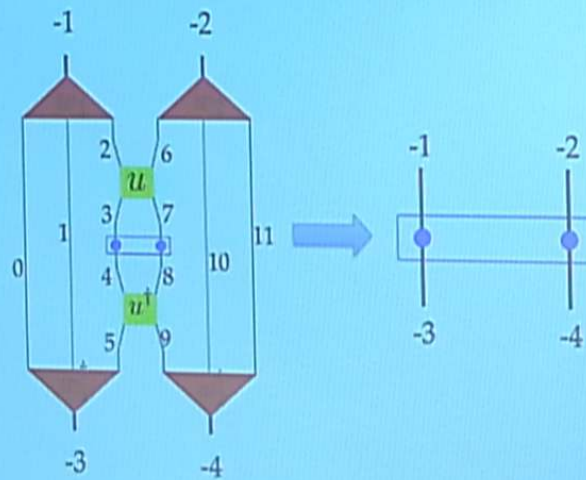
| | | | |
|-------|--------------------------|--------------------------|--------------------------|
| Time | $(\cancel{1/a} + 1/m)$ | $(\cancel{1/b} + 1/n)$ | $(\cancel{1/c} + 1/l)$ |
| Space | $\cancel{b} - c - l - n$ | $\cancel{a} - c - l - m$ | $\cancel{a} - b - m - n$ |

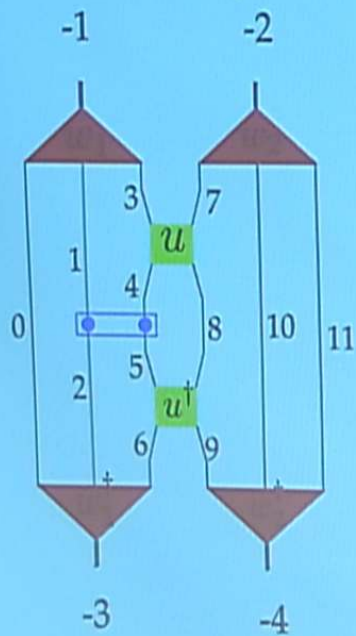


CTM.net

C1: 1 2
C2: 34 35
C3: 41 42
C4: 19 16
T1b: 1 20 3 4
T1a: 20 34 21 22
T2a: 35 38 36 37
T2b: 38 41 39 40
T3b: 42 33 31 32
T3a: 33 19 17 18
T4a: 16 13 14 15
T4b: 13 2 5 6
A1: 4 24 10 6 8
A1T: 3 23 9 5 7
B1: 22 37 27 24 25
B1T: 21 36 26 23 25
A2: 27 40 32 29 30
A2T: 26 39 31 28 30
B2: 10 29 18 15 12
B2T: 9 28 17 14 11
O: 7 8 11 12
TOUT:
ORDER: C1 T1b T4b A1T A1 O B2T B2 T4a C4 T3a T1a B1T B1 A2T A2 T3b C2 T2a

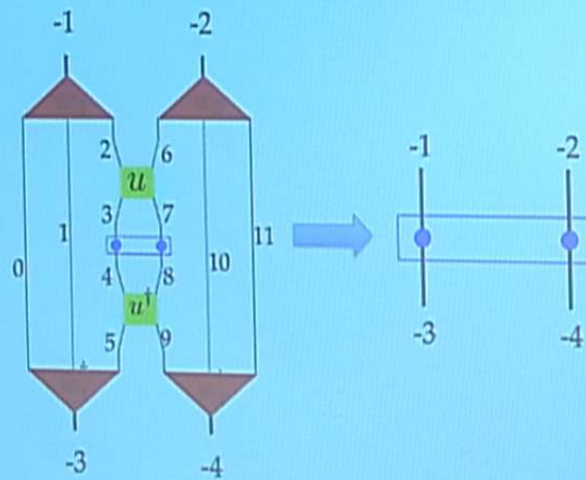


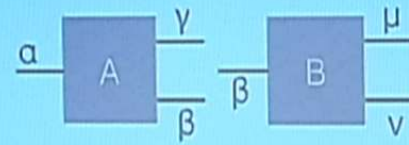


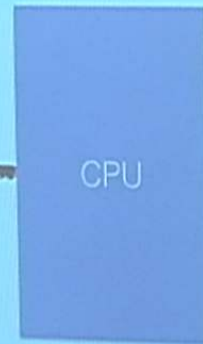


```
#include "uni10.hpp"
```

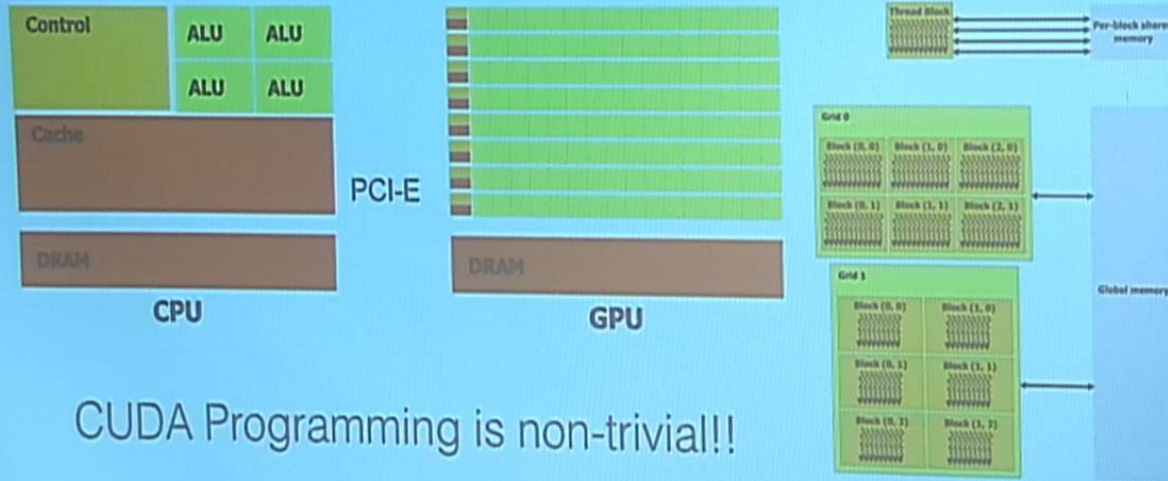
```
Qnum q10(1, PRT_EVEN);
Qnum q_10(-1, PRT_EVEN);
Qnum q30(3, PRT_EVEN);
Qnum q_11(PRTF_ODD, -1, PRT_EVEN);
Qnum q11(PRTF_ODD, 1, PRT_EVEN);
Qnum q_31(PRTF_ODD, -3, PRT_EVEN);
```







GPU Programming



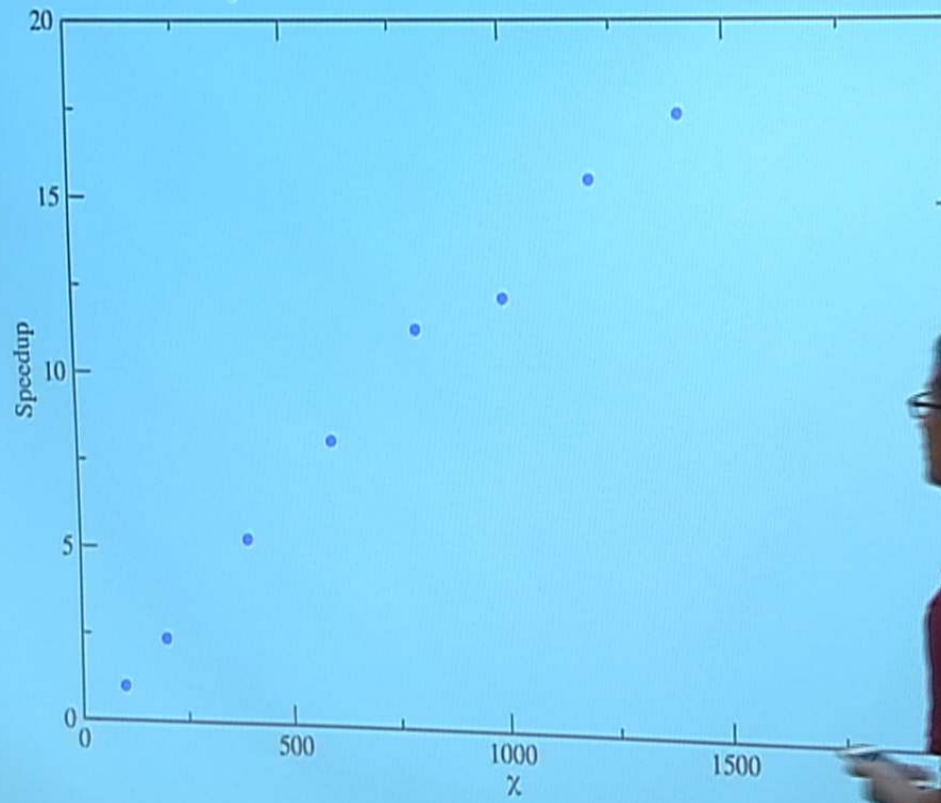
CUDA Programming is non-trivial!!

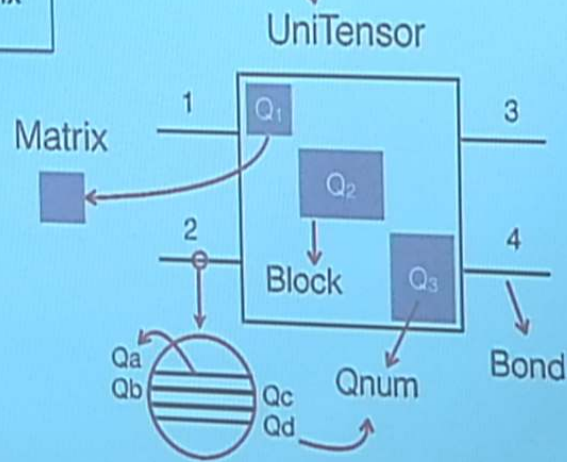
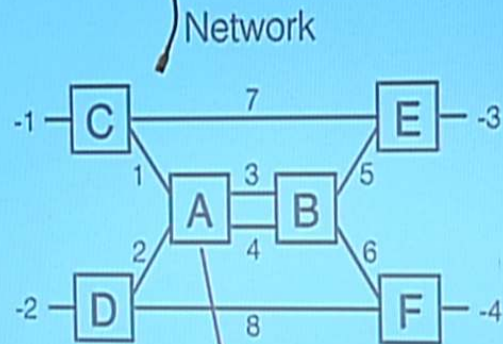
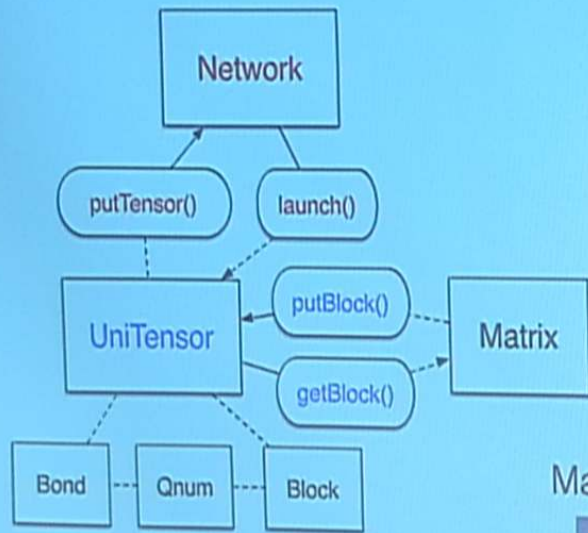
```

__global__ void _setDiag(double* elem, double* diag_elem, size_t M, size_t N, size_t diag_N){
    size_t idx = blockIdx.y * BLOCKMAX * THREADMAX + blockIdx.x * blockDim.x + threadIdx.x;
    if(idx < diag_N && idx < M && idx < N)
        elem[idx * N + idx] = diag_elem[idx];
}

__global__ void _getDiag(double* elem, double* diag_elem, size_t M, size_t N, size_t diag_N){
    size_t idx = blockIdx.y * BLOCKMAX * THREADMAX + blockIdx.x * blockDim.x + threadIdx.x;
    if(idx < diag_N && idx < M && idx < N)
        diag_elem[idx] = elem[idx * N + idx];
}
    
```

iTEBD $d=2$





Welcome more users,
testers, and developers !!

Uni10

- Geared toward tensor network algorithms
- Reduce development time in coding
- Behind-the-scene optimization
- Support for Python and Matlab (soon)
- To do:
 - GUI tools for network generation
 - Non-abelian symmetries: $SU(2)$
 - Multi-GPU/Multi-node support
 - Better search algorithm for contraction order (Tensor Contraction Engine)