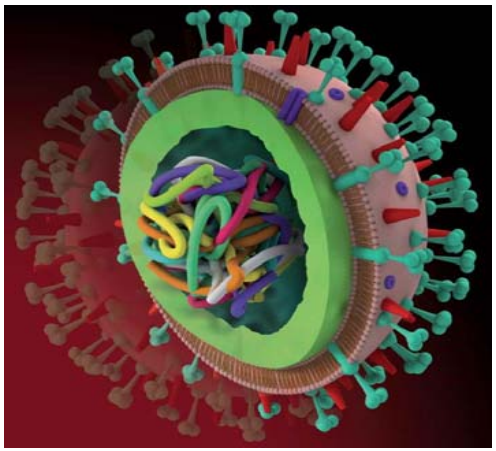
GPU Acceleration of a Production Molecular Docking Code

Bharat Sukhwani **Martin Herbordt**

Computer Architecture and Automated Design Laboratory
Department of Electrical and Computer Engineering
Boston University
http://www.bu.edu/caadlab

*** This work supported, in part, by the U.S. NIH/NCRR**
+ Thanks to Tom VanCourt (Altera) and Sandor Vajda and Dima Kozakov (BME at Boston University)

Why is Docking so important?


#

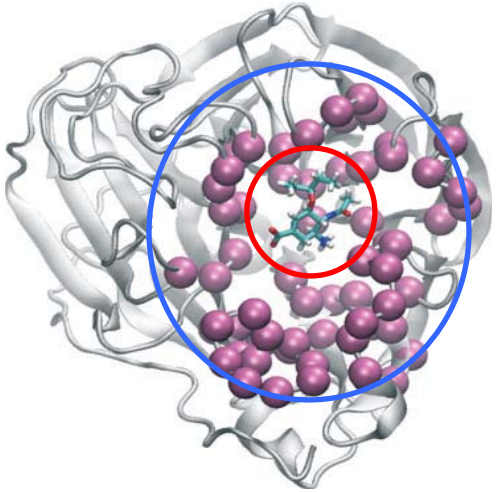
Problem: Combat the bird flu virus

Method: Inhibit its function by "gumming up" *Neuraminidase*, a surface protein, with an inhibitor

- Neuraminidase helps release progeny viruses from the cell.

Procedure*:

- Search protein surface for likely sites
- Find a molecule that binds there (and only there)



*Landon, et al. Chem. Biol. Drug Des 2008

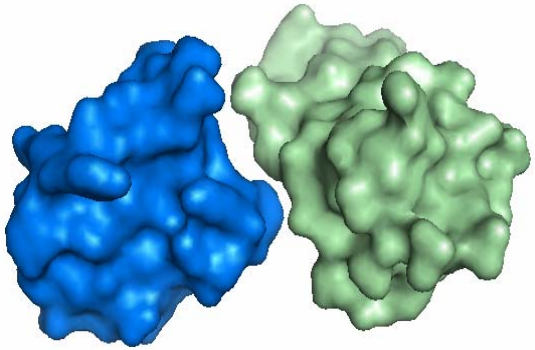
#From New Scientist www.newscientist.com/channel/health/bird-flu

Overview of Molecular Docking

Docking ≡ Modeling interactions between two molecules

Computational Task

- Finding the least energy 'pose'
  - Offset and rotation of one relative to the other



e.g. – Exhaustive search

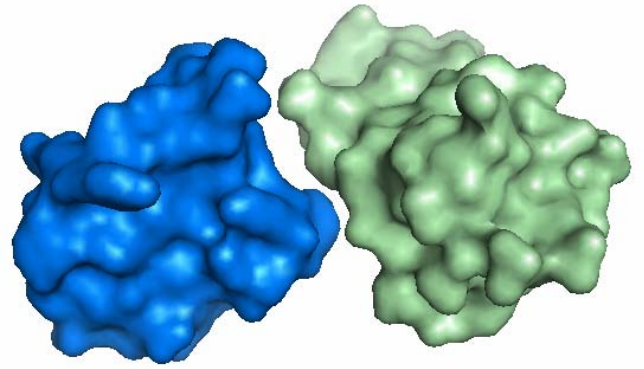
- Usually performed in two steps
  - Docking – Exhaustive sampling of 3D space
  - Energy minimization

Figure generated using PyMOL

Types of Docking

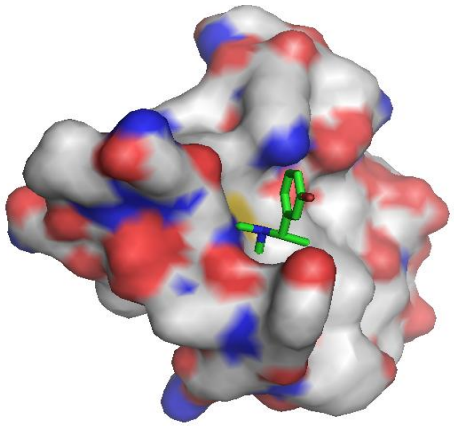
Protein-Protein Docking

- Complex Structure prediction
- X-Ray method is difficult
- Typical grid size: 16^3 to 128^3



Protein-Ligand Docking

- Used for drug discovery
- Screening millions of drug candidates
- In-silico screening is faster and more cost effective
- Typical ligand grid size: 4^3 to 16^3



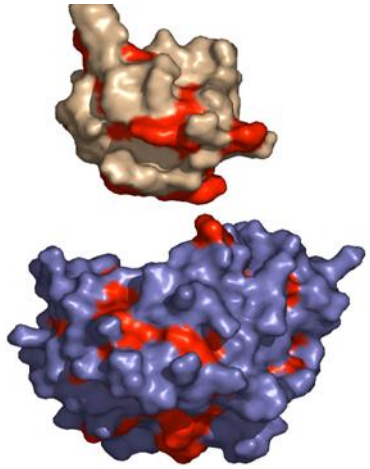
Modeling Rigid Docking

Rigid-body approximation

Grid based computing

Exhaustive 6D search

Pose score = 3D correlation sum



$$E(\alpha,\beta,\gamma) = \sum_{p} \sum_{i,j,k} R_P(i,j,k) \cdot L_P(i+\alpha, j+\beta, k+\gamma)$$

FFT to speedup the correlation

Reduces from $O(N^6)$ to $O(N^3 \log N)$

Image courtesy of Structural Bioinformatics Lab, BU

Why Accelerate Docking?

Rigid docking

- Tens of thousands of rotations
- Each requires multiple FFTs/ IFFTs
- Typically: 10 sec per rotation
- Total runtime ~ 98 hrs!

Flexible docking adds another DoF

- Uses rigid docking as preprocessor or subroutine

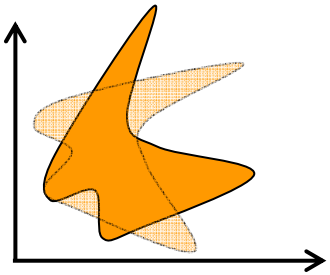
Faster docking would aid in drug discovery

- Faster screening (of millions of potential drug candidates)
- Better discrimination

Computations in Rigid Docking

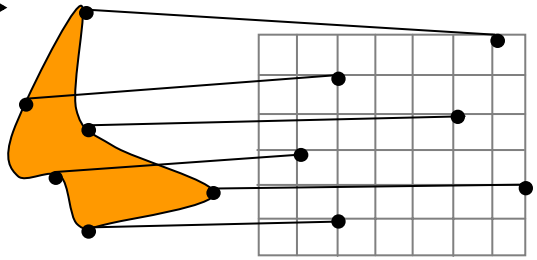
Rotation

- Increments of 5 to 15 degrees



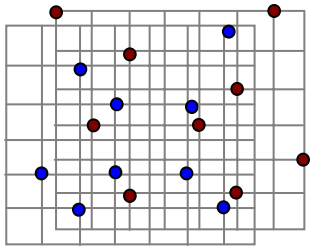
Grid assignment

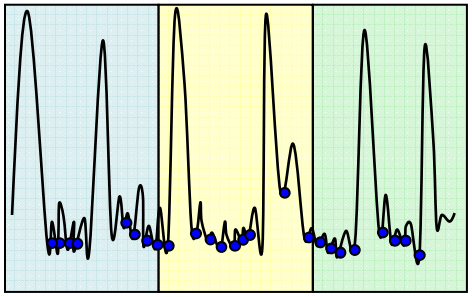
- For each energy function



Pose score

- FFT, Modulation and IFFT
- For each energy function





Filtering top scores

- Selecting regional best scores

Overview of PIPER Docking Code

Based on rigid molecule docking

Also used as a subroutine in another program

- ClusPro docking and discrimination program

Uses several energy functions

- Most sophisticated used in this type of code

Core computation is 3D correlations (FFTs)

- For each energy function, for each rotation.
- Typical padded grid size = 128^3

PIPER Energy Functions

Three energy functions

- Shape complementarity – 2 terms
- Electrostatics – 2 terms
- Pairwise Potential – ‘k’ terms
  - k = 2 to 18 (usually 4)

$$E_{shape} = E_{attr} + w_1 E_{repul}$$

$$E_{elec} = E_{born} + E_{coulomb}$$

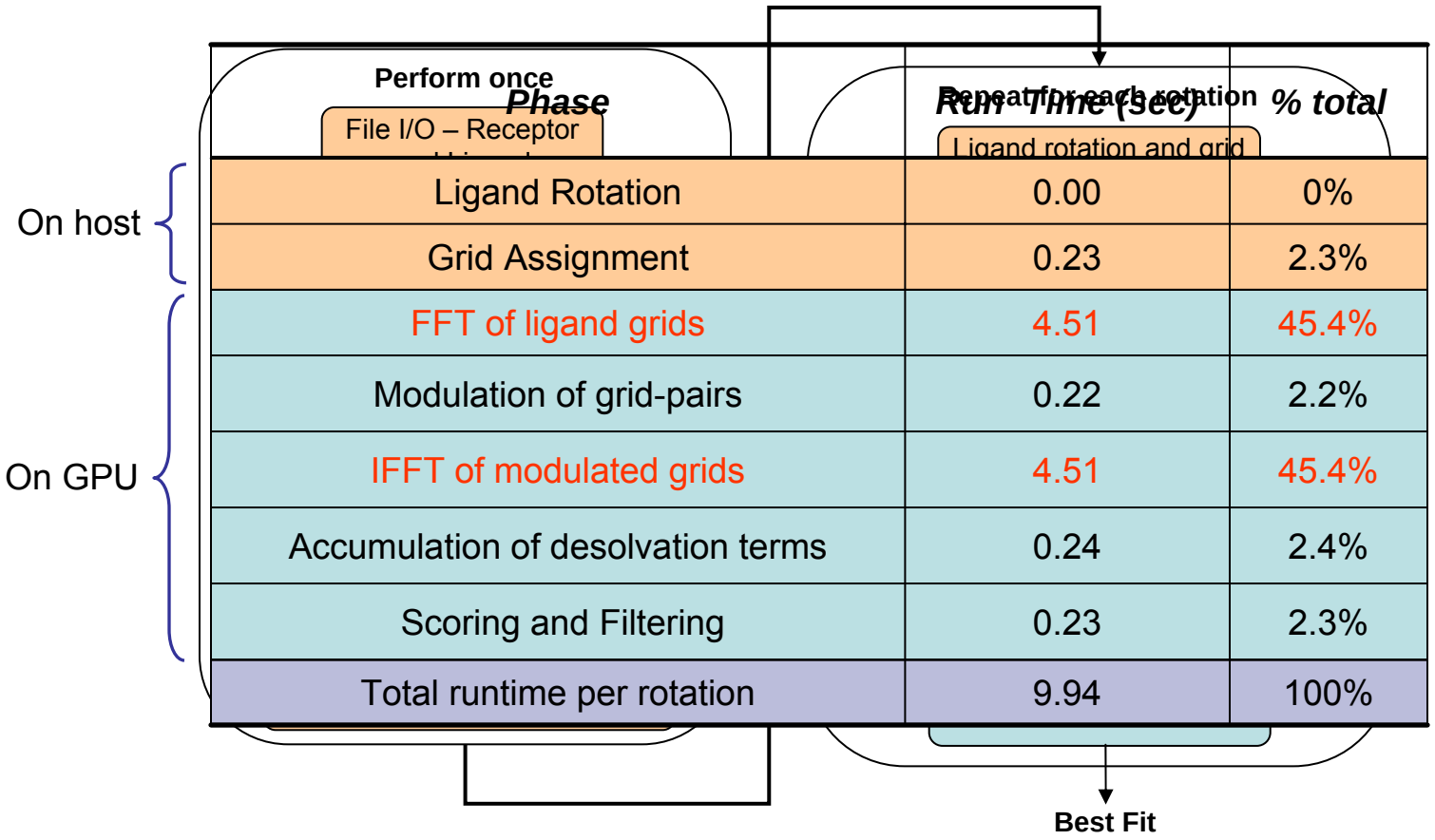
$$E_{desol} = \sum_{k=0}^{P-1} E_{pairpot\_k}$$

Combined in weighted sum

$$E = E_{shape} + w_2 E_{elec} + w_3 E_{desol}$$

‘k’ + 4 correlations per rotation

Original PIPER Program Flow



Phase	Run Time (sec)	% total
Ligand Rotation	0.00	0%
Grid Assignment	0.23	2.3%
FFT of ligand grids	4.51	45.4%
Modulation of grid-pairs	0.22	2.2%
IFFT of modulated grids	4.51	45.4%
Accumulation of desolvation terms	0.24	2.4%
Scoring and Filtering	0.23	2.3%
Total runtime per rotation	9.94	100%

Mapping PIPER to GPU

Correlation

- Direct correlation
- FFT Correlation
  - FFT
  - IFFT
  - Modulation

Accumulation of desolvation terms

Scoring and Filtering

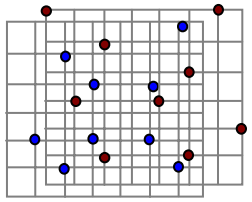
Rotation and Grid assignment

- Latency hiding

Direct correlation on GPU

Replaces steps of FFT, Modulation and IFFT

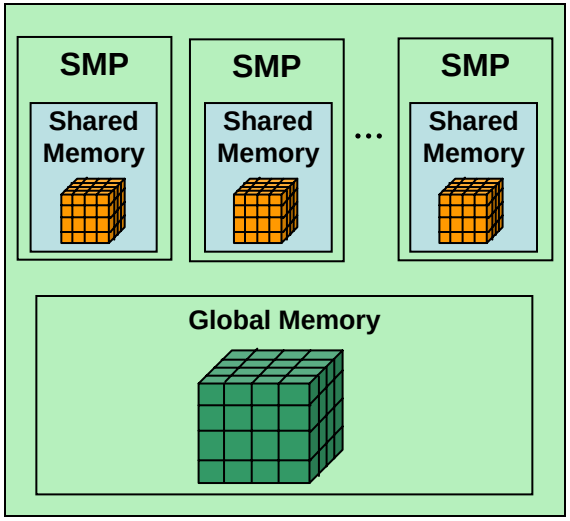
- Shifting, Voxel-voxel interaction, grid summation



Each multiprocessor accesses both grids

Receptor grid ⇨ Global memory

Ligand grid ⇨ Shared memory



Multiple correlations together

- For different energy functions

Direct correlation on GPU

Shared memory limits the ligand size

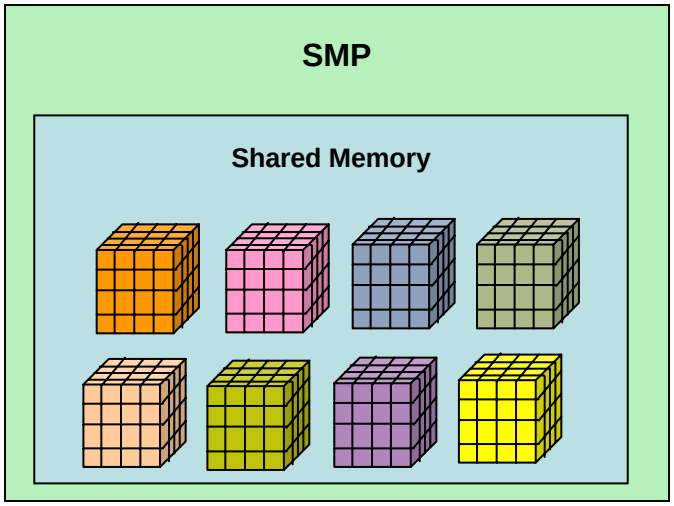
- With 4 pairwise term - 8 cubed ligand

For larger ligand grids

- Store on global memory and swap
- Degrades performance

For smaller grids - Multiple rotations

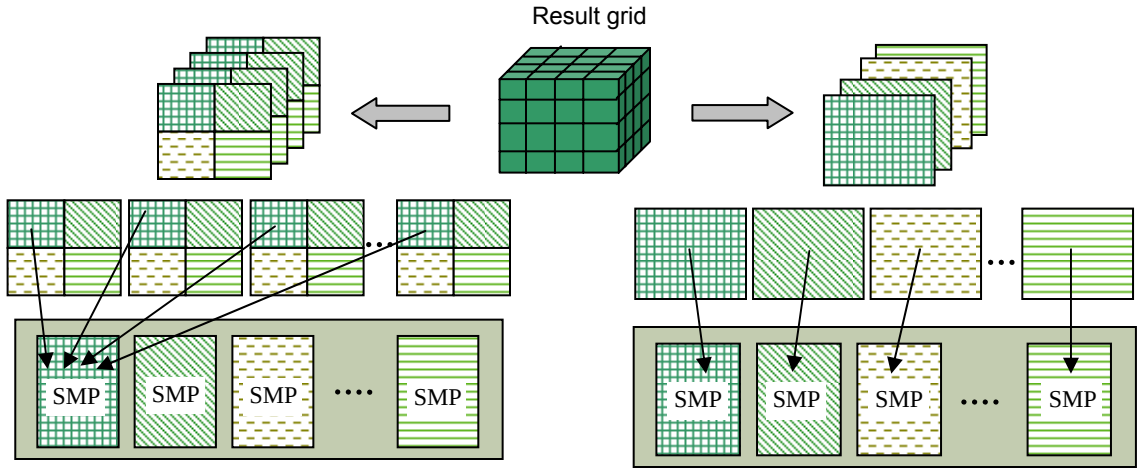
- For 4 cubed grid - 8 rotations together
- Multiple computation per fetch
- 2.7x performance improvement



Direct correlation on GPU

Distribution of work among threads

- 2D Plane to thread block
- Part of the plane to thread block
- Yield similar results



FFT Correlation on GPU

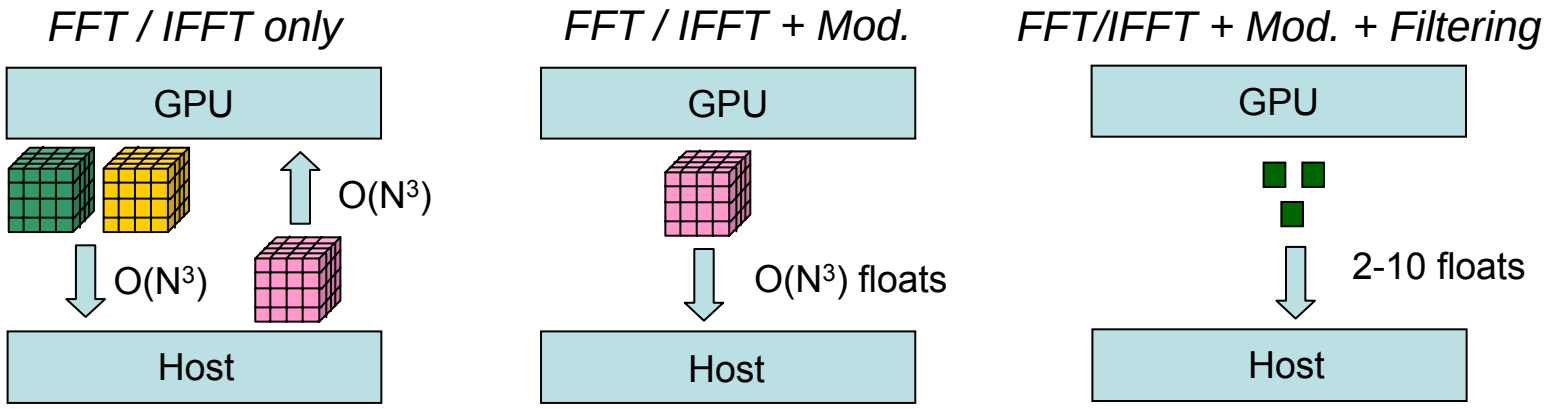
Direct correlation is not attractive for large grids

Multiple FFTs in serial order

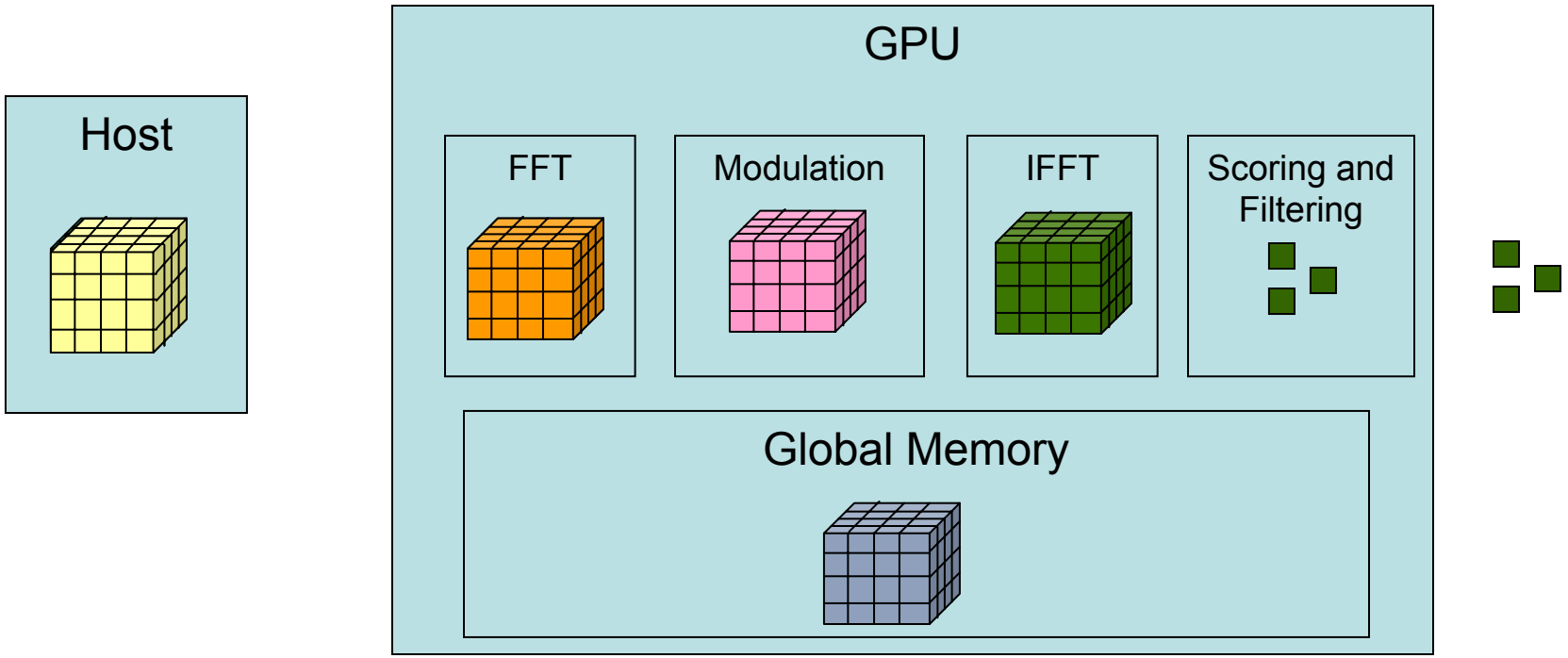
- Using NVIDIA CUFFT library

Minimize host ⟺ device data transfer

- Perform as many steps on GPU as possible



FFT Correlation on GPU



Direct Correlation v/s FFT

Direct Correlation	FFT Correlation
Good for small ligand grids	
Multiple rotations per iteration	Good for large ligands
Limits number of energy terms	Any number of energy terms
Runtime $\propto$ ligand size	Runtime $\propto$ padded grid size
Provides implicit filtering	Explicit filtering required

PIPER Scoring and Filtering

Critical for overall performance

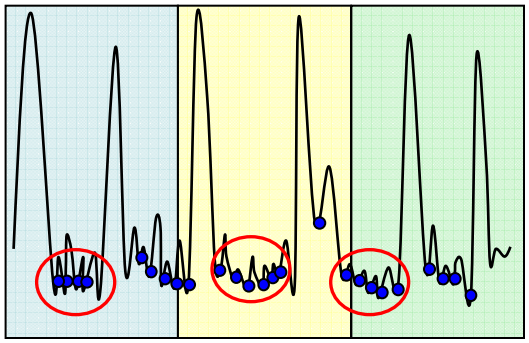
Scoring

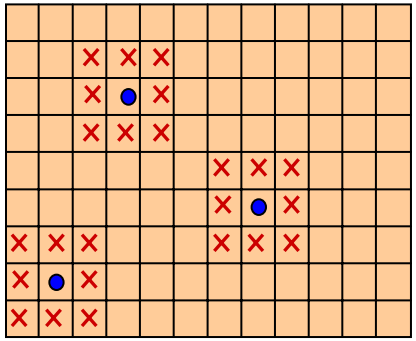
- Multiple sets of weights

$$E = E_{shape} + w_2 E_{elec} + w_3 E_{desol}$$

Filtering

- Regional Best

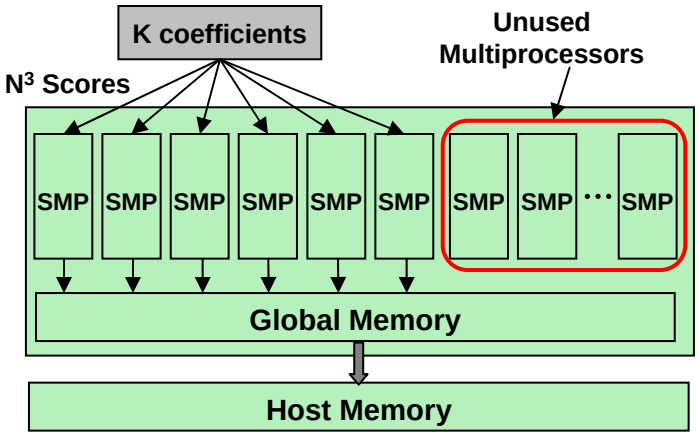




Scoring and Filtering on GPU

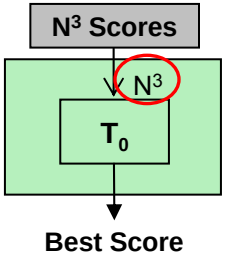
Weight-sets distributed on different multiprocessors

- Weights stored in constant cache
- Multiprocessors underutilized



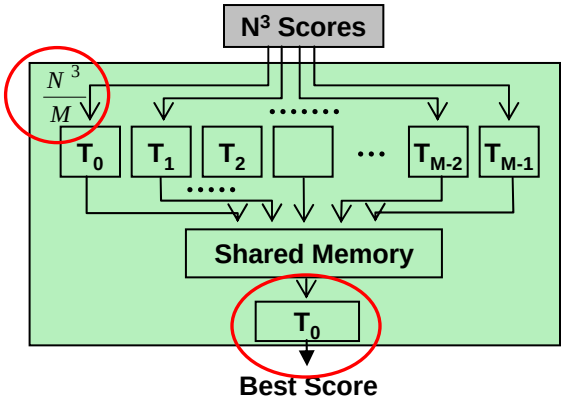
Naïve scheme

- Negative speedup



Second scheme

- Threads store scores in shared memory
- Serialization at the end
  - Thread 0 finds best of best
  - Also performs flagging of cells



Other schemes possible

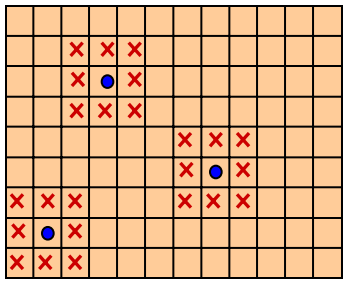
Scoring and Filtering on GPU

Flagging the neighboring cells

- Serial PIPER:
- Does not fit in GPU shared memory

(N3 entries)

1	1	0	0	0	…	1	0



Solution 1 – Exclusion index array

- 2-3x slowdown w.r.t. host filtering

(100 entries)

4	5	16	28	….	45

Solution 2 – Bit array on GPU global memory

- One array for each set of weights
- Achieves speedup over host filtering

(N3 entries each)

1	1	0	0	0	…	1	0
0	0	0	1	1	…	0	0

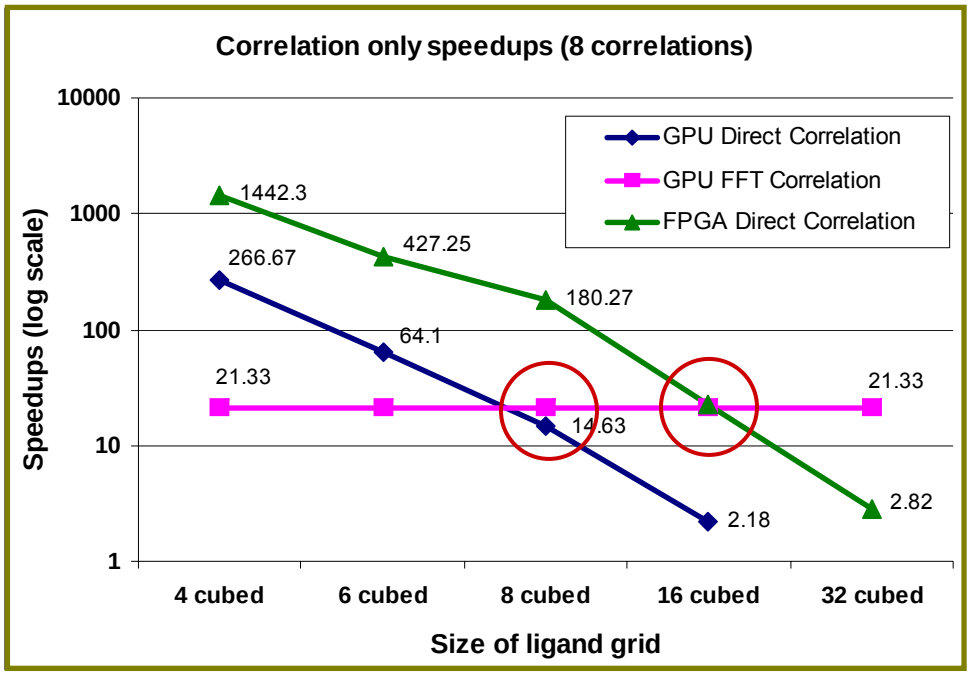
Results

Speedup for different phase

	Phase	*CPU Time (ms)*	*GPU Time (ms)*	*Speedup*
Once per rotation, per energy grid	Forward FFT	205	9.3	22
	Modulation	10	0.01	1000
	Inverse FFT	205	11.8	17
Once per rotation	Accumulation of desolvation terms	240	0.09	2667
	Scoring and Filtering	230	39.5	6
For 22 grids	**Total runtime per rotation**	**9980**	**556**	**18**

Results

Correlation only Speedup: FFT v/s Direct correlation



GPU: NVIDIA TESLA C1060

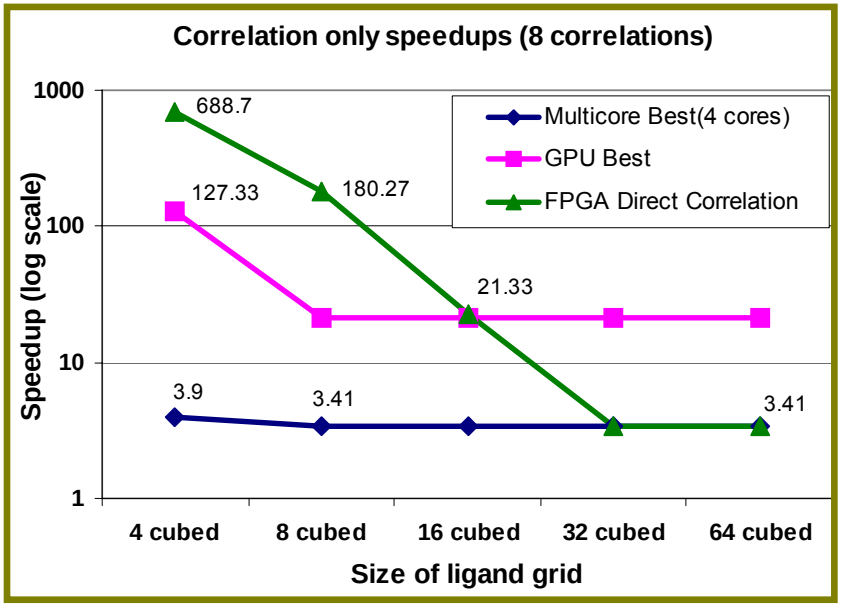
FPGA: Altera Stratix III

CPU: Intel Quad core Xeon @ 3.00 GHz

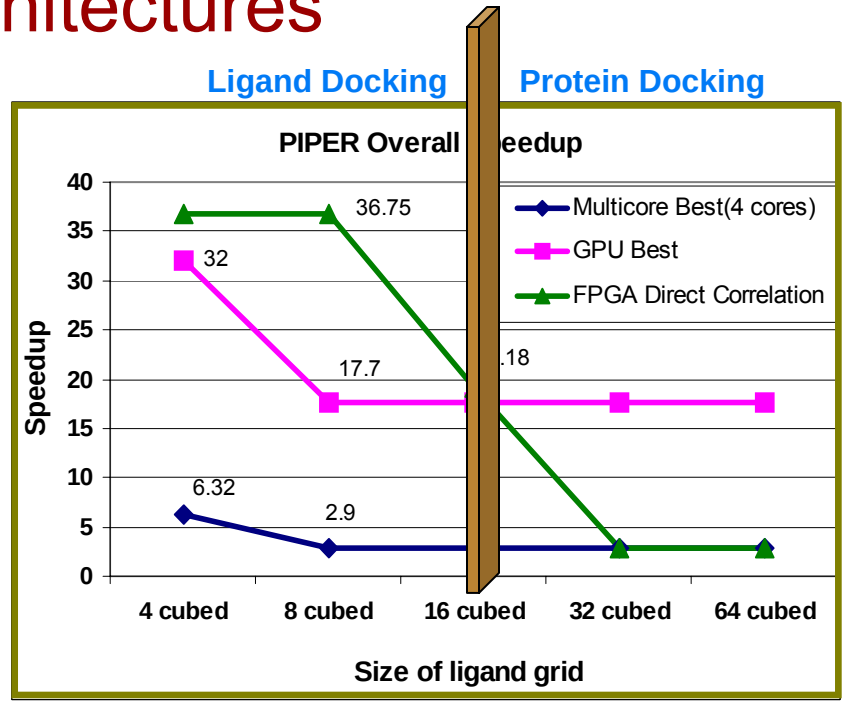
* Baseline: FFT Correlation on single core

Results

Speedup on different architectures



* Baseline: Best Correlation on single core



* Baseline: PIPER running on single core

Thank You

Extra Slides

Results

Actual runtimes

Correlation only runtimes – 8 correlations

Ligand grid size	*Serial*	*GPU*	*FPGA*
4 cubed	3600 ms	13.5 ms	2.5 ms
8 cubed	3600 ms	170 ms	20 ms
16 cubed	3600 ms	170 ms	160 ms

PIPER runtimes for 10,000 rotations – 22 correlations

Ligand grid size	*Serial*	*GPU*	*FPGA*
4 cubed	28 hrs.	52 min	46 min
8 cubed	28 hrs.	94 min	46 min
16 cubed	28 hrs.	94 min	87 min

Results

Direct correlation on GPU – 8 correlations

Runtimes for different grid and block sizes

Ligand grid size	*Grid Size*	*Block Size*	*Runtime*
8 cubed	16*16	8*8*8	245 ms
	8*8	8*8*8	435 ms
	16*16	4*4*4	461 ms
16 cubed	16*16	8*8*8	1650 ms
	8*8	8*8*8	3120 ms
	32*32	4*4*4	2205 ms