Migration of a Complete 3D Poisson Solver from Optimized Fortran77 to GPU

Huynh Phung Huynh
Shyh-hao Kuo
Rick Siow Mong Goh
Le Duc Vinh
Terence Hung

A*STAR Institute of High Performance Computing (IHPC)
Singapore

Institute of
High Performance
Computing

Outline

- 3D Poisson Solver
- Algorithm Flow
- Mapping to GPU
- Experimental Results
- Conclusions and Future Work

Institute of High Performance Computing
A*STAR

3D Poisson Solver

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \right) \varphi(x, y, z) = f(x, y, z)$$

Wide range applications of Poisson Solver:

- Fluid Dynamic
- Solar Magnetostatics
- Electrostatics
- Mechanical Engineer
- Theoretical physics
- …

Institute of High Performance Computing

Mapping a Complete 3D Poisson Solver to GPU

- Complexity of a complete 3D Poisson Solver
  - Five different boundary conditions for each dimension (Periodic, N-N, D-D, N-D, D-N)
  - Two grid configurations (Staggered , Centered)
  - > Totally 250 implementations of 3D Poisson Solver

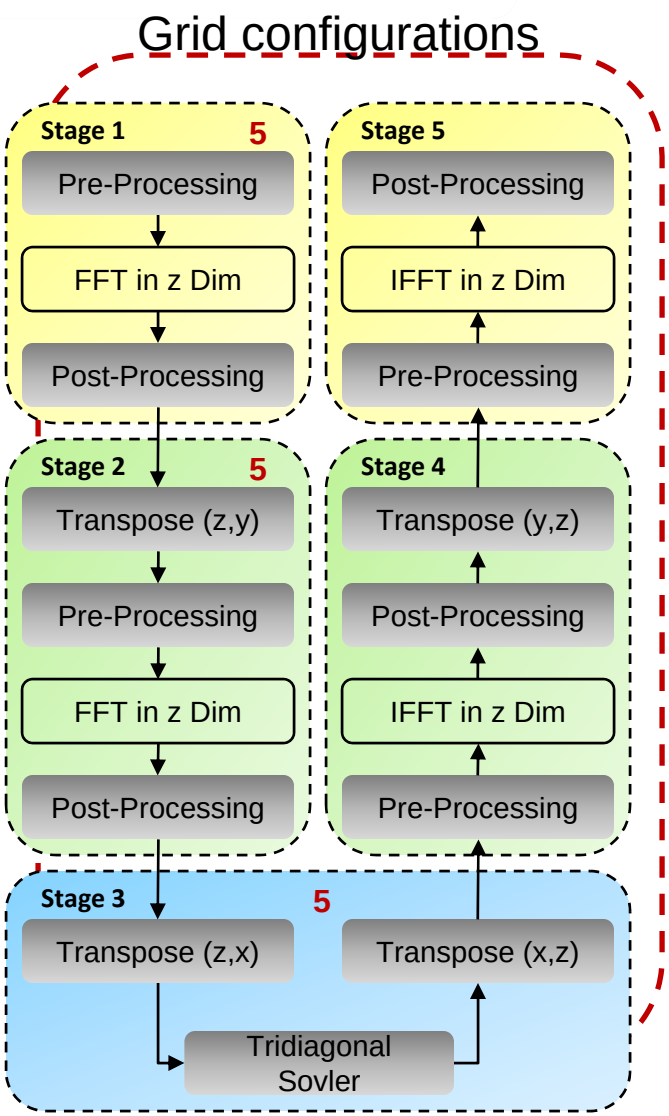
- Objective: Strike the balance between user efficiency and library modularity

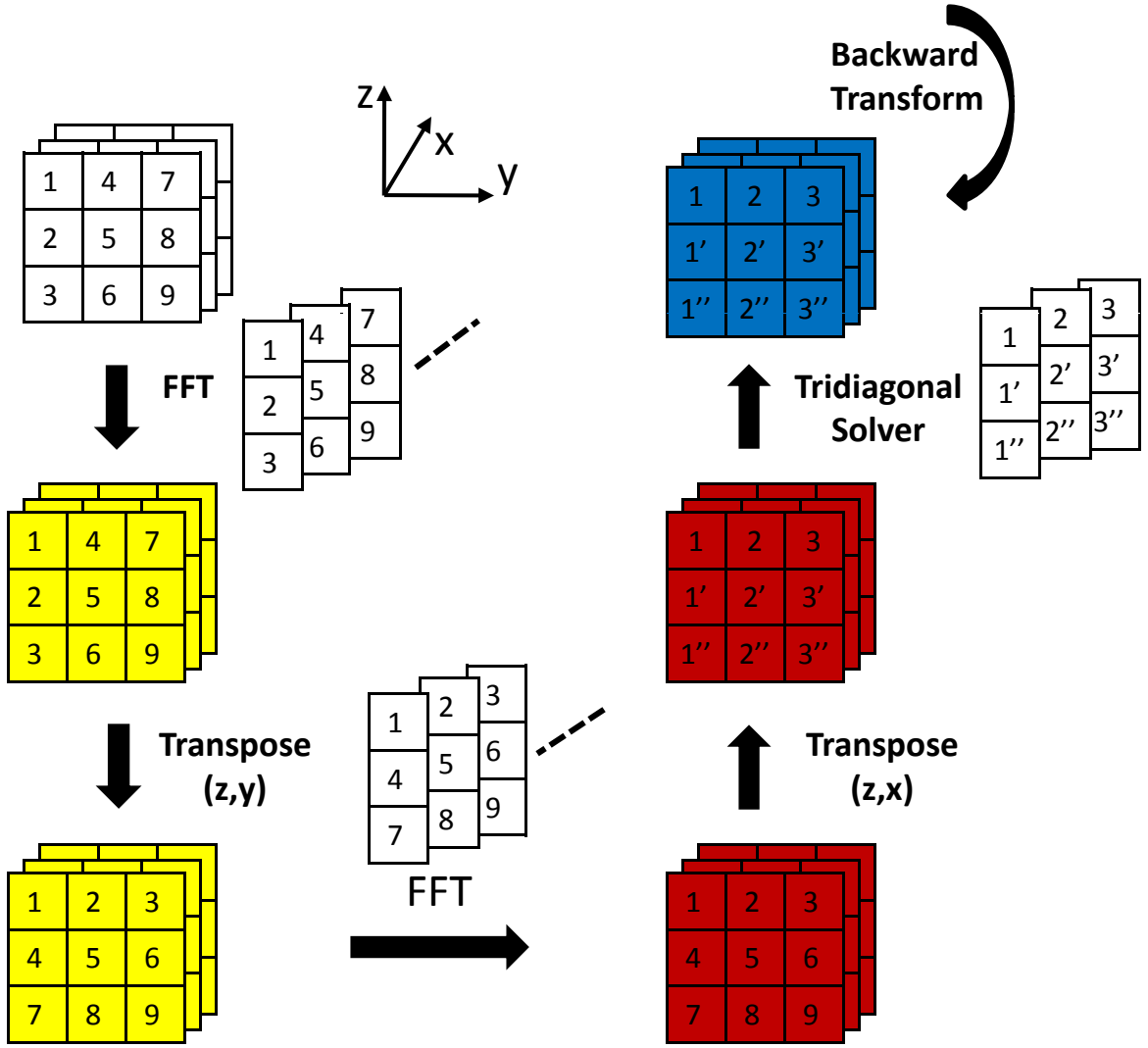
Outline

- 3D Poisson Solver
- **Algorithm Flow**
- Mapping to GPU
- Experimental Results
- Conclusions and Future Work

Institute of High Performance Computing
A*STAR

3D Poisson Solver Flow





Outline

- 3D Poisson Solver
- Algorithm Flow
- **Mapping to GPU**
- Experimental Results
- Conclusions and Future Work

Migration Challenges

- Optimized Fortran code:
  - heavily reuse code segment -> a lot of branch instructions
  - assume uniform memory access
- Our tasks:
  - Refactoring optimized code to naïve version
  - Re-modularize the code for GPU to accommodate hundreds of different configurations
  - GPU memory and performance optimization

Institute of High Performance Computing
A*STAR

Pre-processing :: VSSINF1

Original Fortran

```
      DO 101 K=1,N
      DO 100 I=1,L
         FT(I,K,1)=F(I,1,K)
  100    end do
  101    end do
      IF (MOD(M,2).EQ.0) THEN
         DO 111 K=1,N
         DO 110 I=1,L
            FT(I,K,M)=-F(I,M,K)
  110       end do
  111       end do
      ENDIF
      DO 202 J=2,M-1,2
      DO 201 K=1,N
      DO 200 I=1,L
         FT(I,K,J)   = 0.5*(F(I,J+1,K)-F(I,J,K))
         FT(I,K,J+1) =-0.5*(F(I,J+1,K)+F(I,J,K))
  200    end do
  201    end do
  202    end do
```

- This code is easy to vectorize on a SIMD Structure.
- The processing is synchronized with barriers.
- Not suitable for CUDA implementation.

Institute of High Performance Computing
A*STAR

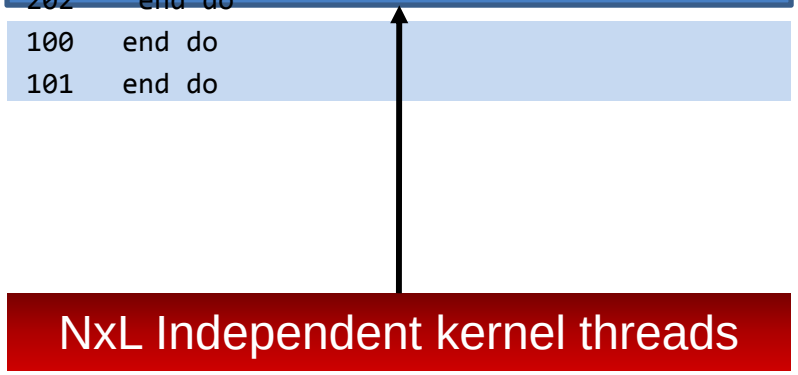
Pre-processing :: VSSNF1

Original Fortran

```
      DO 101 K=1,N
      DO 100 I=1,L
         FT(I,K,1)=F(I,1,K)
  100    end do
  101    end do
      IF (MOD(M,2).EQ.0) THEN
         DO 111 K=1,N
         DO 110 I=1,L
            FT(I,K,M)=-F(I,M,K)
  110       end do
  111       end do
      ENDIF
      DO 202 J=2,M-1,2
      DO 201 K=1,N
      DO 200 I=1,L
         FT(I,K,J)   = 0.5*(F(I,J+1,K)-F(I,J,K))
         FT(I,K,J+1) =-0.5*(F(I,J+1,K)+F(I,J,K))
  200    end do
  201    end do
  202    end do
```

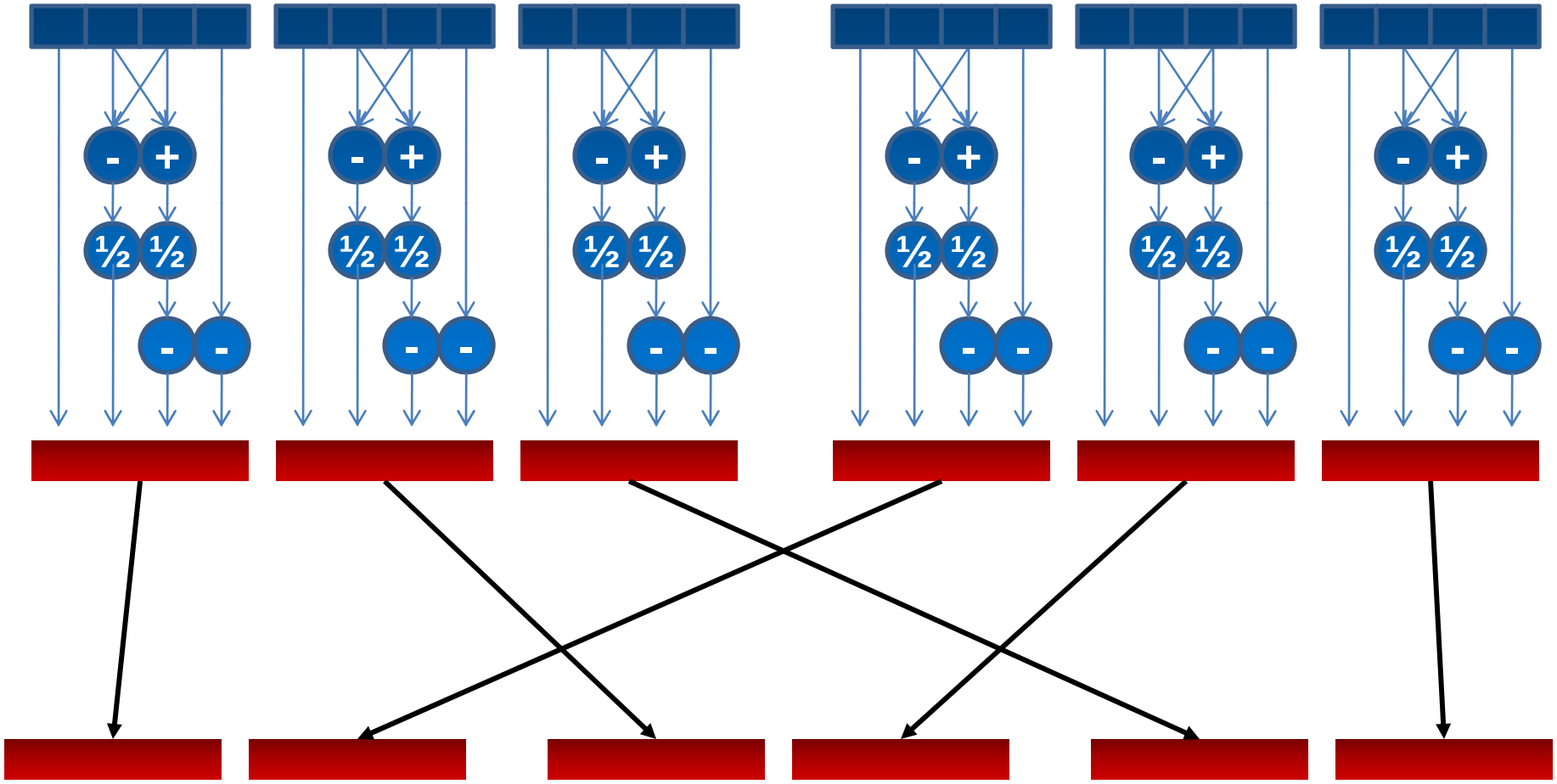
Conceptual Fortran

```
      DO 101 K=1,N
      DO 100 I=1,L
         FT(I,K,1)=F(I,1,K)
         IF (MOD(M,2).EQ.0) THEN
            FT(M)=-F(M)
         ENDIF
         DO 202 J=2,M-1,2
            FT(I,K,J)= 0.5*(F(I,J+1,K)-F(I,J,K))
            FT(I,K,J+1)=-0.5*(F(I,J+1,K)+F(I,J,K))
  202    end do
  100    end do
  101    end do
```



Institute of High Performance Computing
A*STAR

Pre-processing Data Dependency

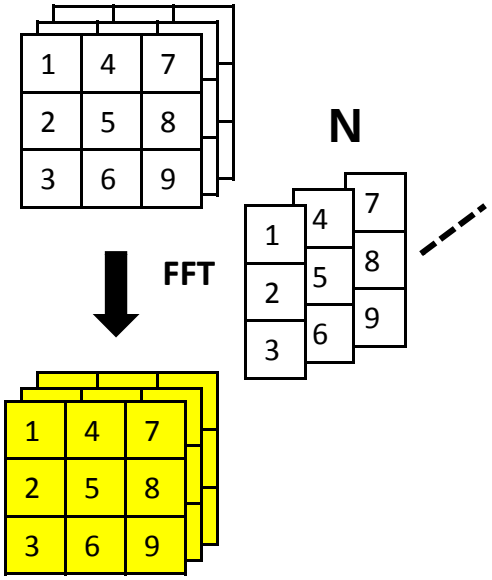


Institute of High Performance Computing
A*STAR

Calling NVIDIA cuFFT

- Create 1D cuFFT plan
- Call cuFFTExec in batch of *N:*

$$N = \frac{allowed\_cuFFT\_transform\_size}{a\_sequence\_transform\_size}$$



- Call cuFFTExec many times if the total number of data needed to transform is too high

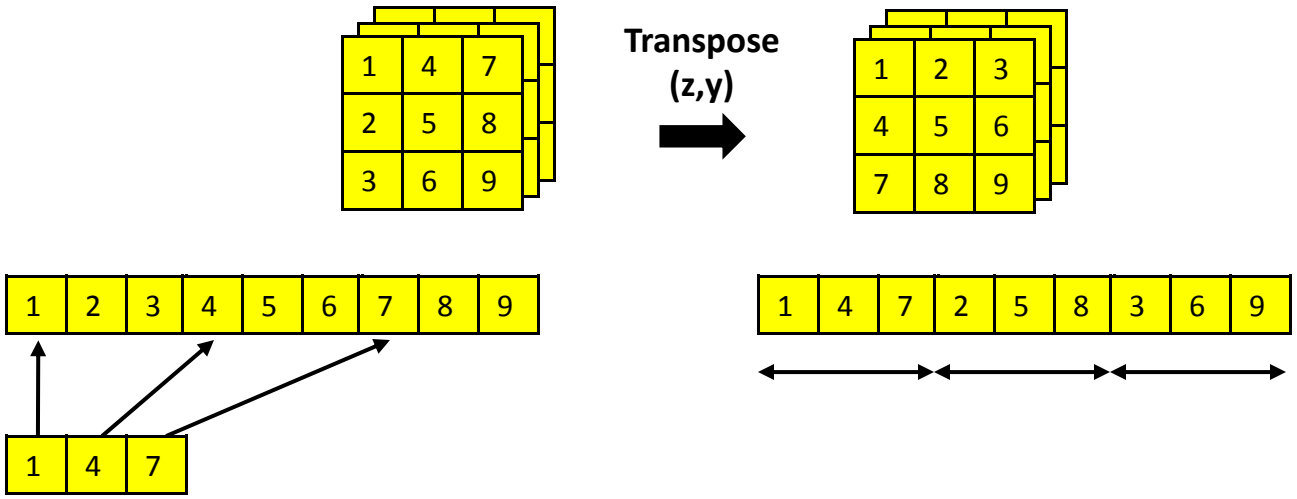
Institute of High Performance Computing

Pre/Post-Processing

Subroutines	PreProcessing	Transform	PostProcessing
VSRFTF	VSRTF1	VRFFTB	VSRTFA
VSRFTB	VSRTB1	VRFFTF	VSRTBA
VSSINF	VSSNF1	VRFFTB	VSSNFA
VSSINB	VSSNB1	VRFFTF	VSSNBA
VSCOSF	VSCSF1	VRFFTB	VSCSFA
VSCOSB	VSCSB1	VRFFTF	VSCSBA
VSSINQ	VSSNQ1	VRFFTB	VSSNQA
VSCOSQ	VSCSQ1	VRFFTB	VSCSQA

Institute of High Performance Computing
A*STAR

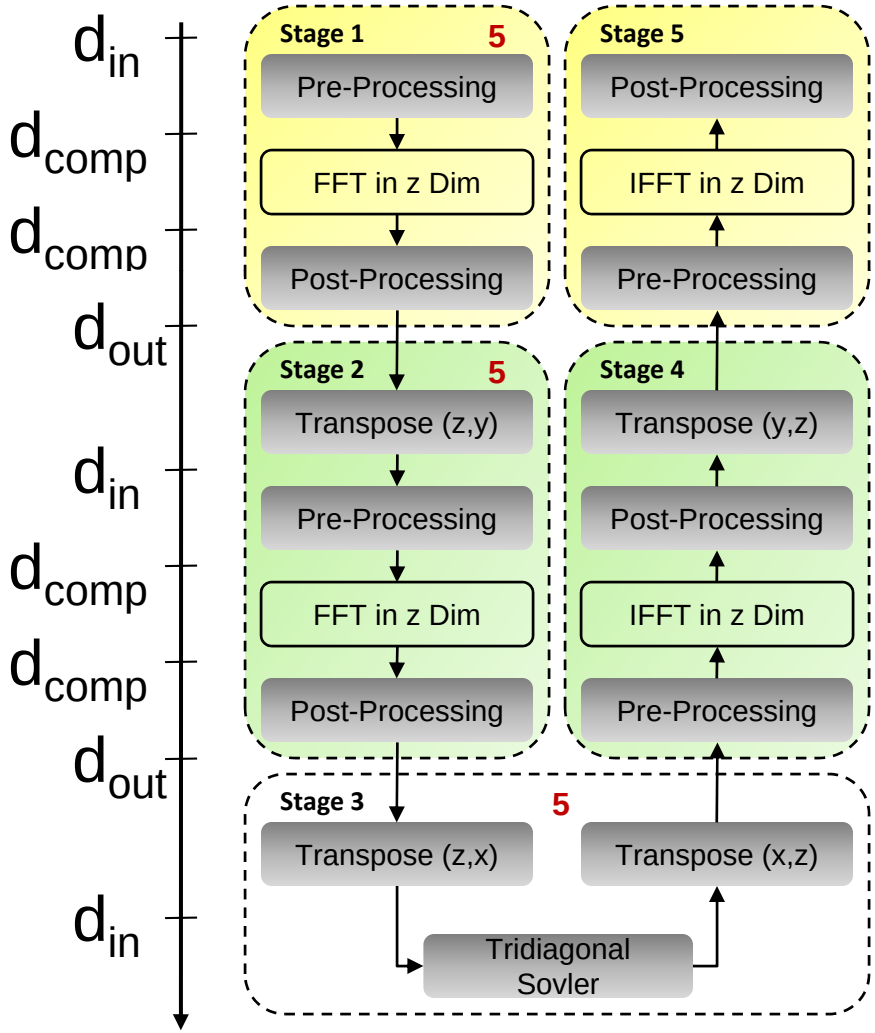
Memory Coalescing



- To ensure memory coalescing
- Using tiling + shared memory

Institute of High Performance Computing
A*STAR

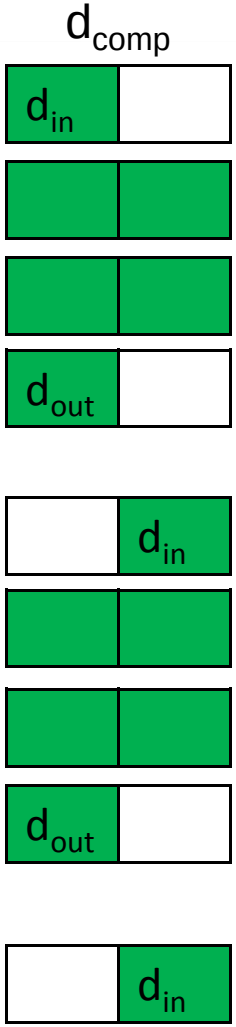
Memory Reuse

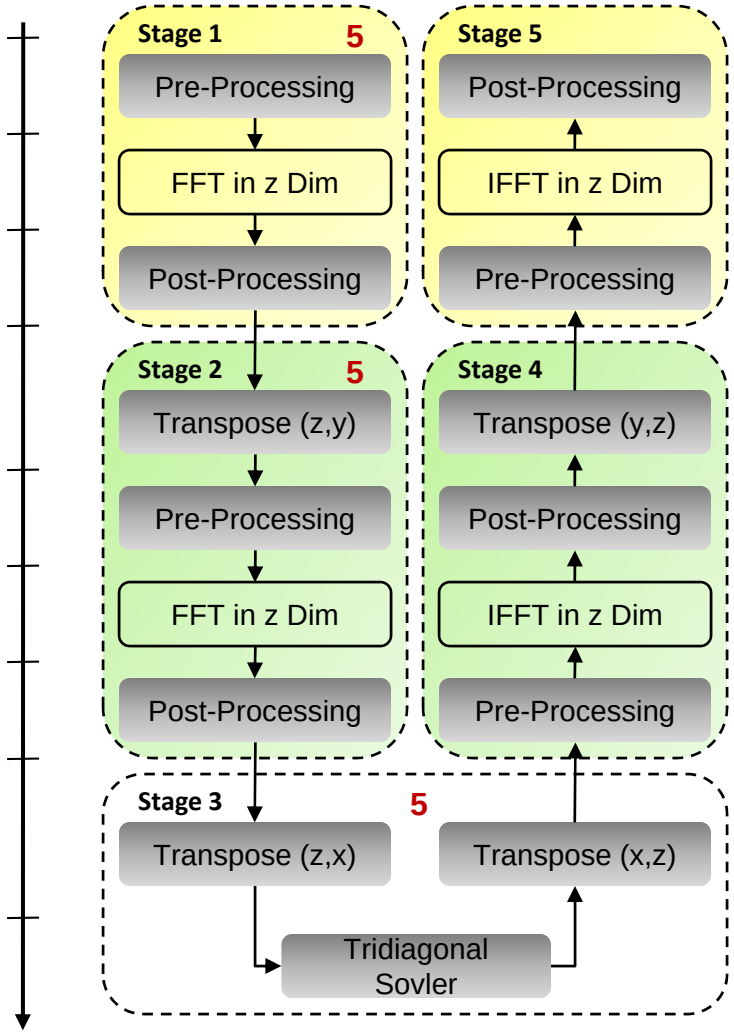


Allocate d_in, d_out, d_complex in device once and reuse throughout the pipeline

Institute of High Performance Computing
A*STAR

Enhanced Reuse Scheme





Allocate ONLY d_complex in device once and reuse throughout the pipeline

Use the shared memory and registers to store the temporary data

Reduce the memory usage by half

Institute of High Performance Computing
A*STAR

Tri-diagonal Solver

- Use Cyclic Reduction from GTC'09

A				X		D
b1	**c1**	0	0	x1	=	d1
a2	**b2**	**c2**	0	x2		d2
0	**a3**	**b3**	**c3**	x3		d3
0	0	**a3**	**b4**	x4		d4

- Each sequence in x dimension corresponds to vector **D**
- Many Tri-diagonal matrix can be solved in parallel

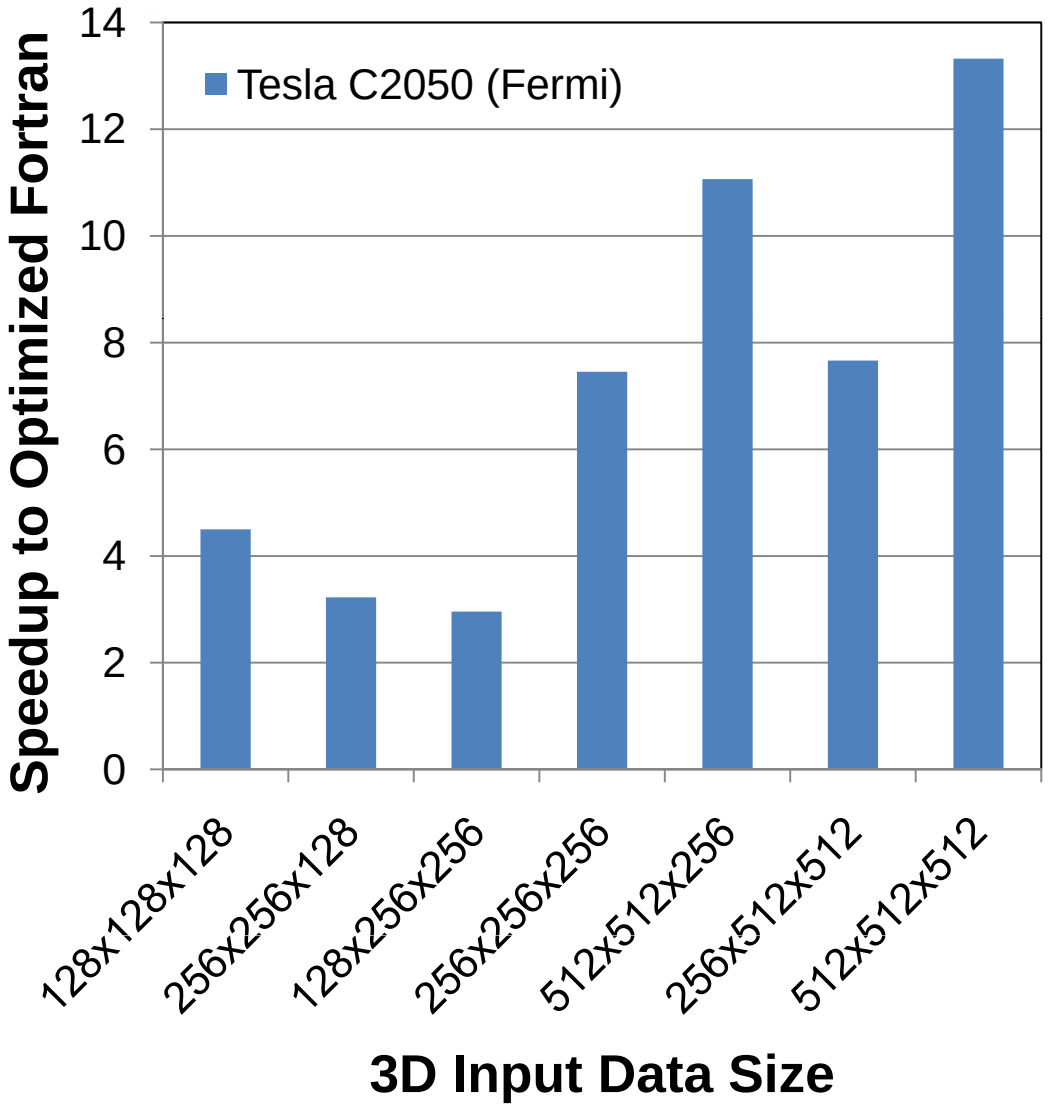
Institute of
High Performance
Computing

Outline

- 3D Poisson Solver
- Algorithm Flow
- Mapping to GPU
- **Experimental Results**
- Conclusions and Future Work

Institute of High Performance Computing
A*STAR

GPU vs Optimized Fortran



Institute of
High Performance
Computing

Conclusions & Future Work

- Provide a generic Poisson Solver library interface
- Fermi GPU version has up to 13 times faster than optimized Fortran77 version (8-core Intel Xeon(R) 3.0 Ghz)
- Implement Multiple-GPU version

Thank you!

- Questions?