The Science of
High-Performance
Computing Group

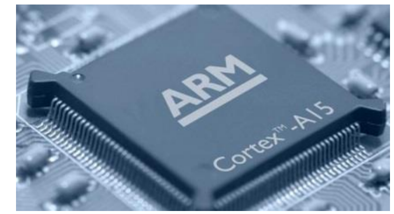
SuperMatrix on Heterogeneous Platforms

Jianyu Huang

SHPC, UT Austin

How Heterogeneous?

CPU



MIC /GPU

Accelerator

How Many Languages?

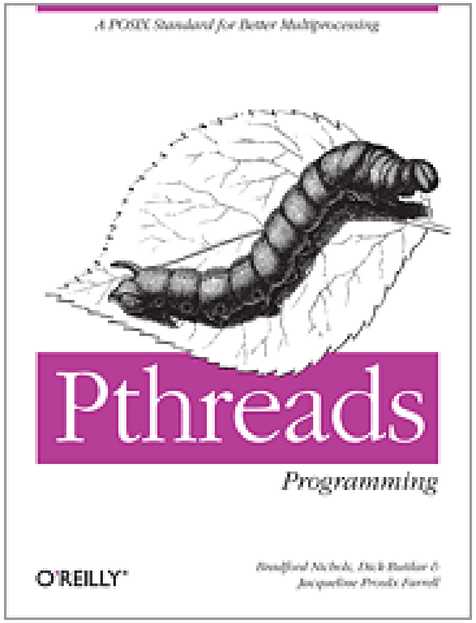
OpenMP

OpenACC
Directives for Accelerators

OpenCL

NVIDIA CUDA

MPI



ArrayFire

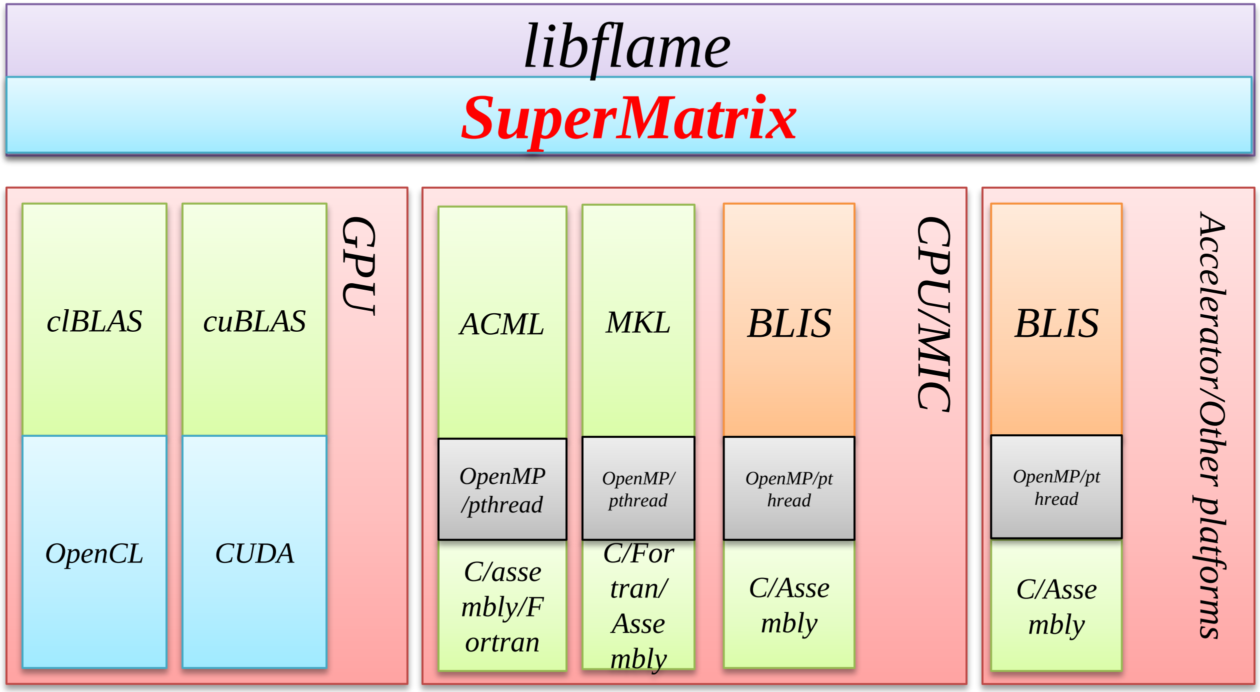
How Many Languages?

Question!

one ring to rule them all?

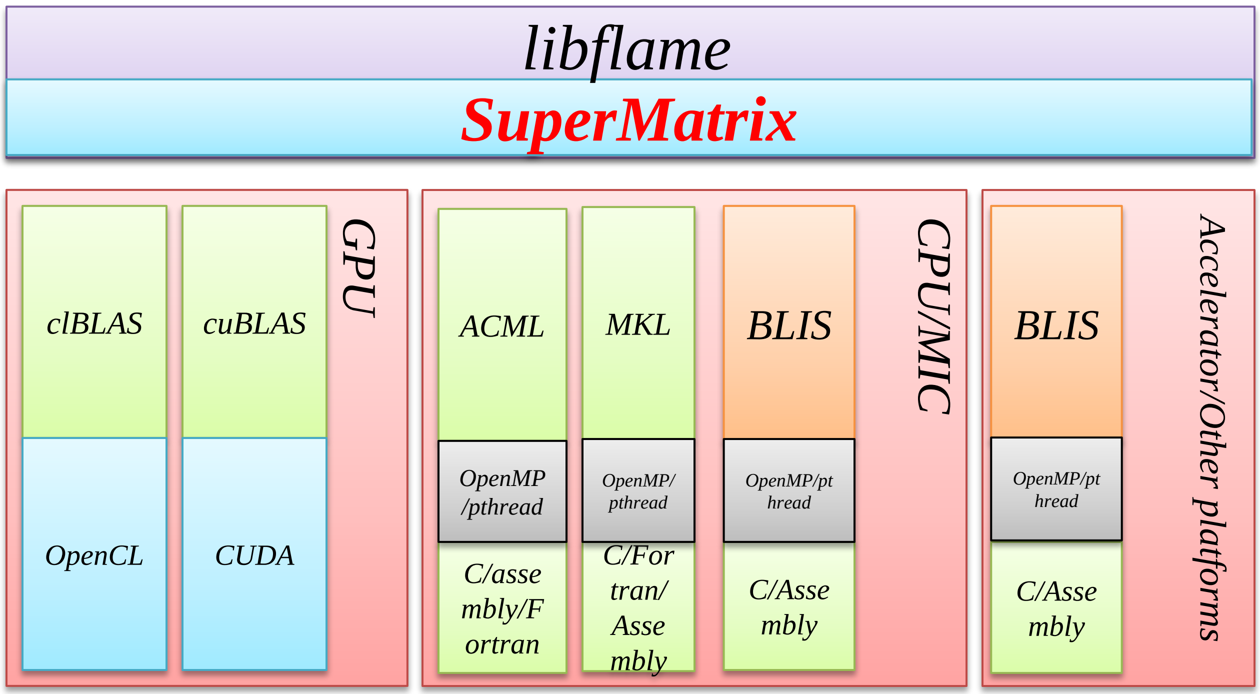
FLAME Answer: SuperMatrix

one ring to rule them all



FLAME Answer: SuperMatrix

one ring to rule them all



- Programmability
  - Use tools provide by FLAME
- Parallelism
  - Directed acyclic graph (DAG) scheduling

FLAME Answer: SuperMatrix

- Chan, E., Quintana-Ortí, E. S., Quintana-Ortí, G., and van de Geijn, R.. SuperMatrix out-of-order scheduling of matrix operations for SMP and multi-core architectures. In *SPAA'07: Proceedings of the Nineteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures,* pages 116-125, San Diego, CA, USA, June 9-11, 2007.
- Chan, E., G. Van Zee, F., Quintana-Ortí, E. S., Quintana-Ortí, G., and van de Geijn, R.. Satisfying your dependencies with SuperMatrix. In*Cluster'07: Proceedings of the 2007 IEEE International Conference on Cluster Computing,* pages 91-99, Austin, TX, USA, September 17-20, 2007.
- Chan, E., G. Van Zee, F., Bientinesi, P., Quintana-Ortí, E. S., Quintana-Ortí, G., and van de Geijn, R.. SuperMatrix: A multithreaded runtime scheduling system for algorithms-by-blocks. In *PPoPP'08: Proceedings of the Thirteenth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming,* pages 123-132, Salt Lake City, UT, USA, February 20-23, 2008.
- Quintana-Orti, G., Igual, F. D., Quintana-Orti, E. S., van de Geijn, R.. Solving dense linear systems on platforms with multiple hardware accelerators. In *PPoPP '09 Proceedings of the 14th ACM SIGPLAN symposium on Principles and Practice of Parallel Programming*, 2009
- Quintana-Ortí, G., Quintana-Ortí, E.S., van de Geijn, R., G. Van Zee, F., and Chan, E.. Programming matrix algorithms-by-blocks for thread-level parallelism. *ACM Transactions on Mathematical Software*, 36(3):14:1-14:26, July 2009.
- Chan, E.. "Application of Dependence Analysis and Runtime Data Flow Graph Scheduling to Matrix Computations." Ph.D. dissertation, Department of Computer Science, The University of Texas at Austin
- Quintana-Ortí, G., Igual, F. D., Marqués, M., Quintana-Ortí, E. S., and van de Geijn, R.. "A Runtime System for Programming Out-of-Core Matrix Algorithms-by-Tiles on Multithreaded Architectures." *ACM Transactions on Mathematical Software (TOMS)* 38, no. 4 (2012): 25.

Parallel?

- S0: $D \leftarrow A*B$
- S1: $A \rightarrow L * L^T$
- S2: $B \leftarrow B * L^{-T}$
- S3: $C \leftarrow C - B * B^T$
- S4: $X \leftarrow L^{-1} * X$

Can the code be parallelized?

Parallel?

- S0: $D \leftarrow A*B$
- S1: $A \rightarrow L * L^T$
- S2: $B \leftarrow B * L^{-T}$
- S3: $C \leftarrow C - B * B^T$
- S4: $X \leftarrow L^{-1} * X$

Can the code be parallelized?

- Write After Read: (S0, S1)
- Read After Write: (S0, S1)
- Read After Write: (S1, S2)
- Read After Write: (S2, S3)
- Read After Write: (S1, S4)

Parallel?

- S0: $D \leftarrow A*B$
- S1: $A \rightarrow L * L^T$
- S2: $B \leftarrow B * L^{-T}$
- S3: $C \leftarrow C - B * B^T$
- S4: $X \leftarrow L^{-1} * X$

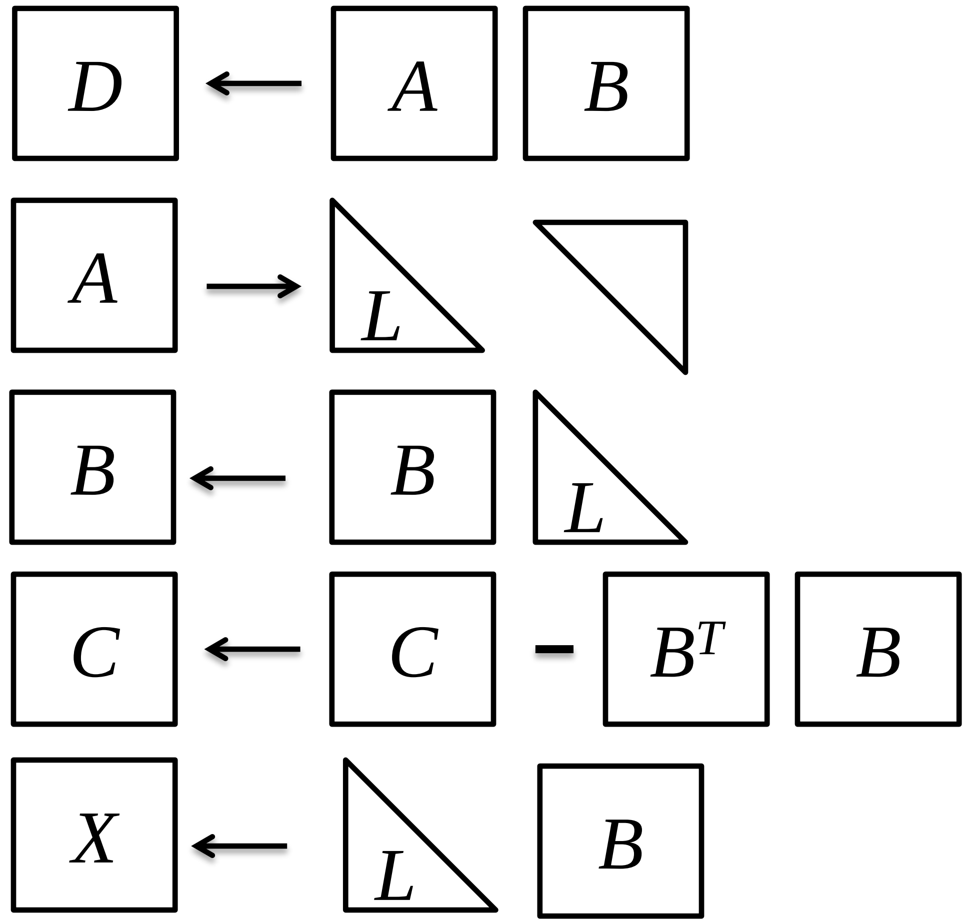
- Write After Read: (S0, S1)
- Read After Write: (S0, S1)
- Read After Write: (S1, S2)
- Read After Write: (S2, S3)
- Read After Write: (S1, S4)

Can the code be parallelized?

Are you sure S1 and S2 cannot be parallelized?

Parallel?

- S0: $D \leftarrow A*B$
- S1: $A \rightarrow L * L^T$
- S2: $B \leftarrow B * L^{-T}$
- S3: $C \leftarrow C - B * B^T$
- S4: $X \leftarrow L^{-1} * X$



How to parallelize?

Traditional Library Approach

- S0: $D \leftarrow A*B$
- S1: $A \rightarrow L * L^T$
- S2: $B \leftarrow B * L^{-T}$
- S3: $C \leftarrow C - B * B^T$
- S4: $X \leftarrow L^{-1} * X$

How to parallelize?

Traditional Library Approach

- S0: $D \leftarrow A*B$
- S1: $A \rightarrow L * L^T$
- S2: $B \leftarrow B * L^{-T}$
- S3: $C \leftarrow C - B * B^T$
- S4: $X \leftarrow L^{-1} * X$

- S0: ParGemm (A,B,D)
- S1: L = ParPortf(A)
- S2: ParTrsm(L,B)
- S3: ParSyrk(B,C)
- S4: ParTrsm(L,X)

How to parallelize?

Traditional Library Approach Implemented with libflame and BLIS

S0: $D \leftarrow A*B$

S1: $A \rightarrow L * L^T$

S2: $B \leftarrow B * L^{-T}$

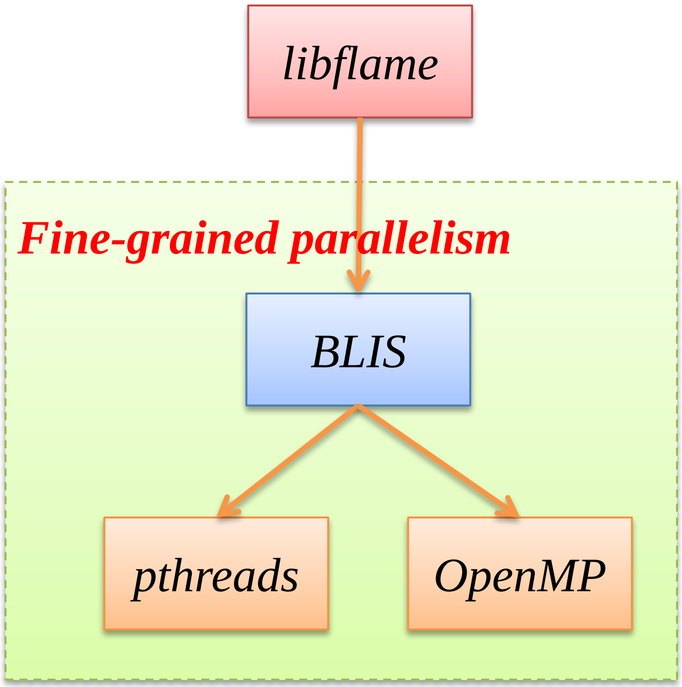
S3: $C \leftarrow C - B * B^T$

S4: $X \leftarrow L^{-1} * X$

```
/*-----------------------------------------------*/
   FLA_Gemm( FLA_NO_TRANSPOSE,FLA_NO_TRANSPOSE,
                   FLA_ONE, A, B, FLA_ZERO, D );
   FLA_Chol( FLA_LOWER_TRIANGULAR, A );
   FLA_Trsm( FLA_RIGHT, FLA_LOWER_TRIANGULAR,
                   FLA_TRANSPOSE, FLA_NONUNIT_DIAG,
                   FLA_ONE, A, B );
   FLA_Syrk( FLA_LOWER_TRIANGULAR, FLA_NO_TRANSPOSE,
                   FLA_MINUS_ONE, B, FLA_ONE, C );
   FLA_Trsm( FLA_LEFT, FLA_LOWER_TRIANGULAR,
                   FLA_NO_TRANSPOSE, FLA_NONUNIT_DIAG,
                   FLA_ONE, L, X );
/*-----------------------------------------------*/
```

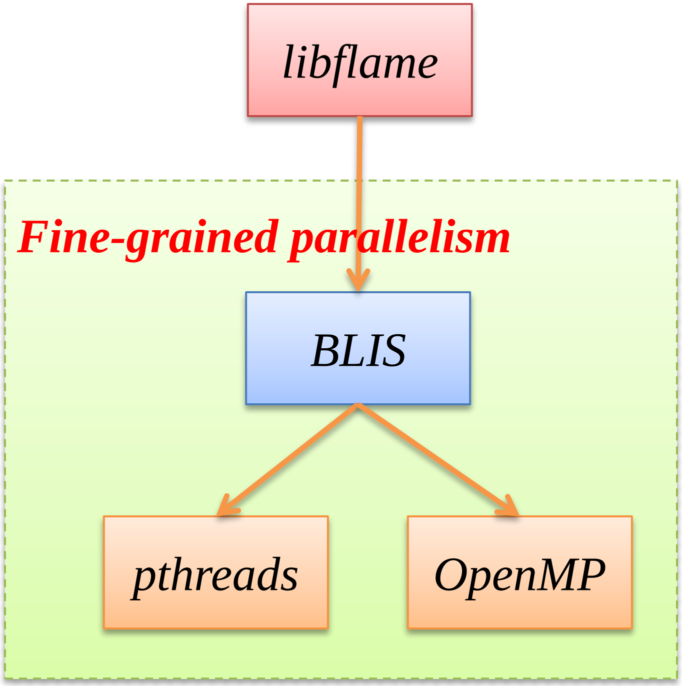
Supported by parallel BLAS, LAPACK (**multi-thread BLIS**)

Problem for Fine-grained Parallelism



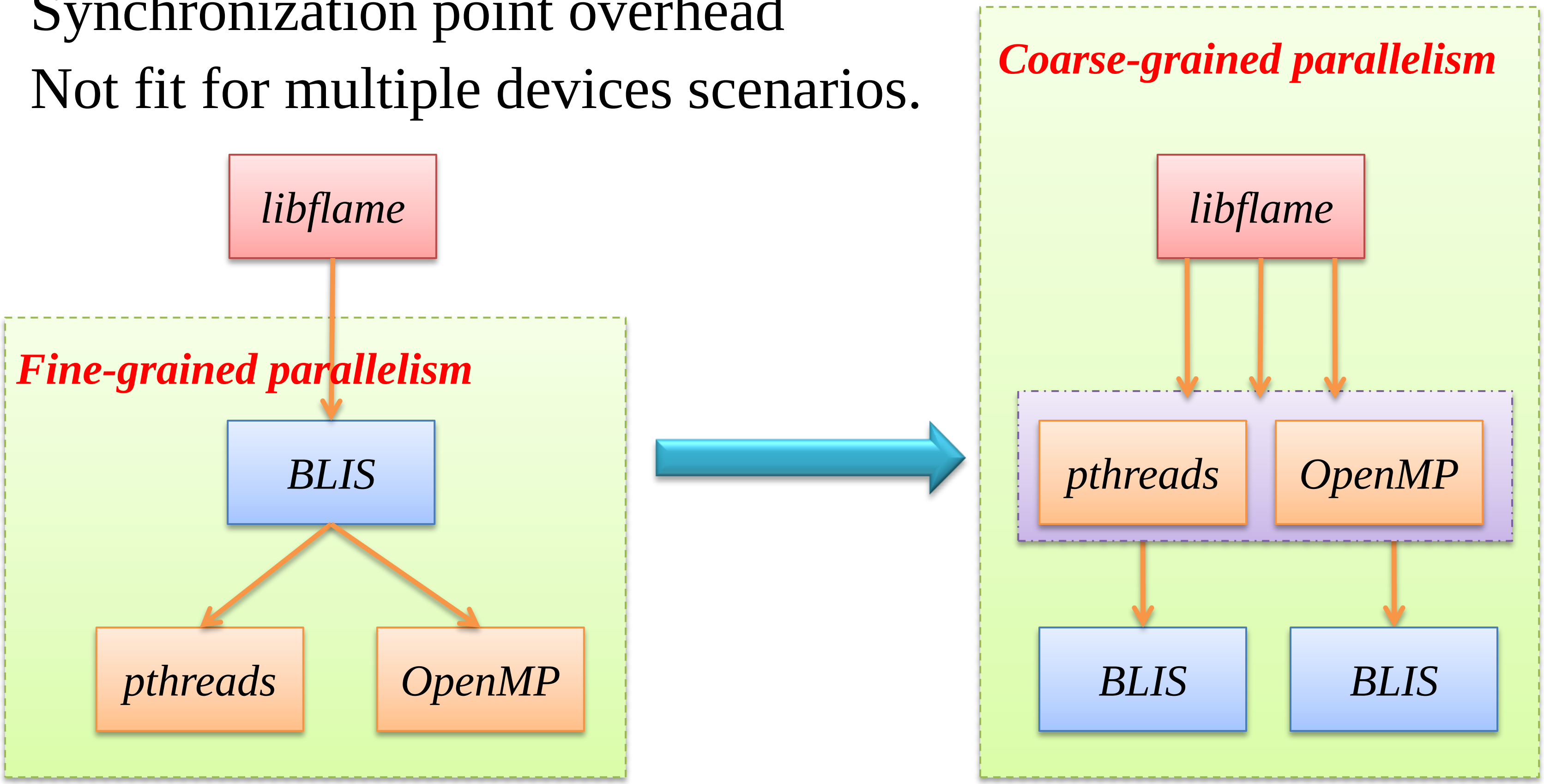
Problem for Fine-grained Parallelism

- Synchronization point overhead
- Not fit for multiple devices scenarios.



Problem for Fine-grained Parallelism

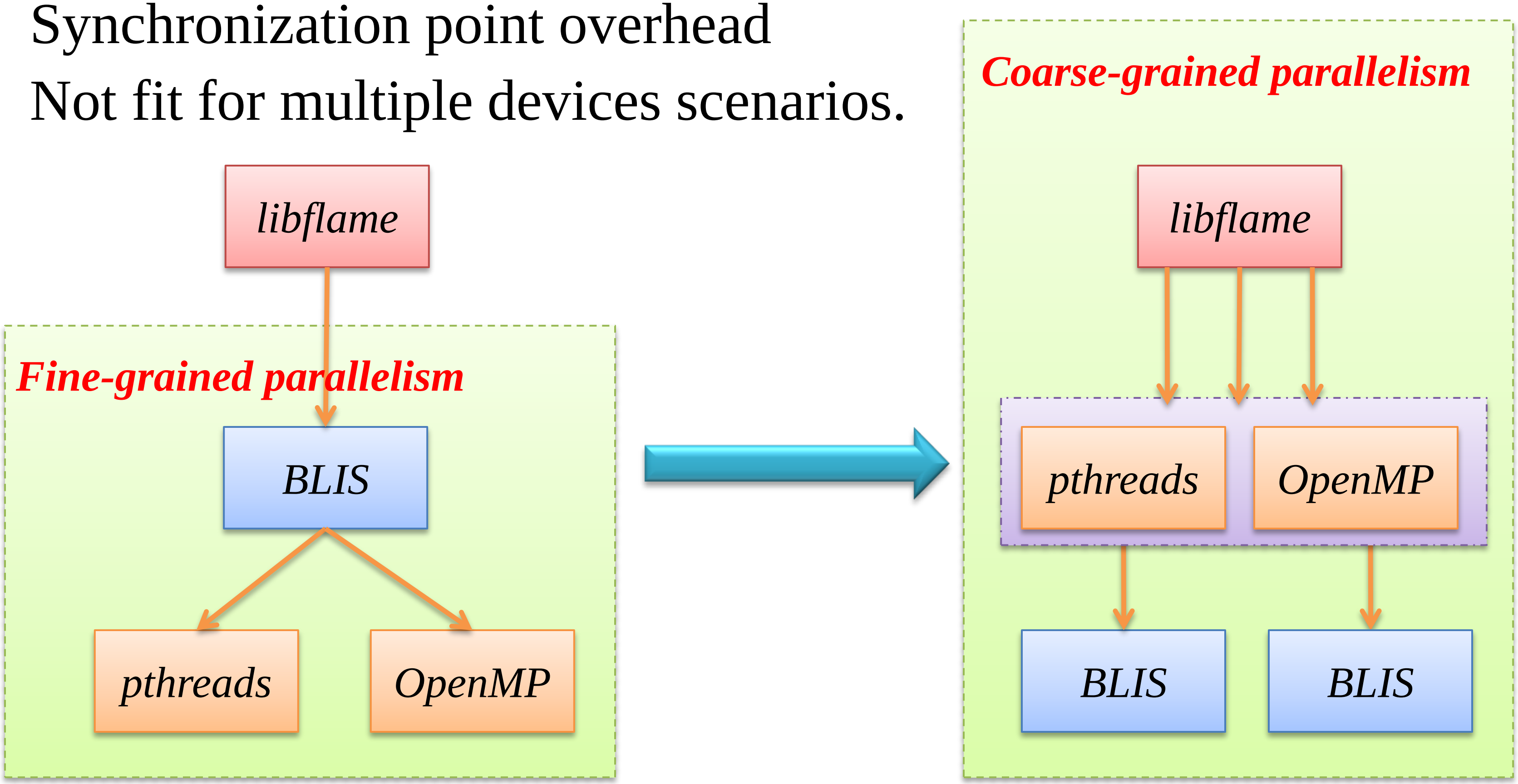
- Synchronization point overhead
- Not fit for multiple devices scenarios.



- Introduce parallelism across instructions
- Fit for the platform with multiple computation units.

Problem for Fine-grained Parallelism

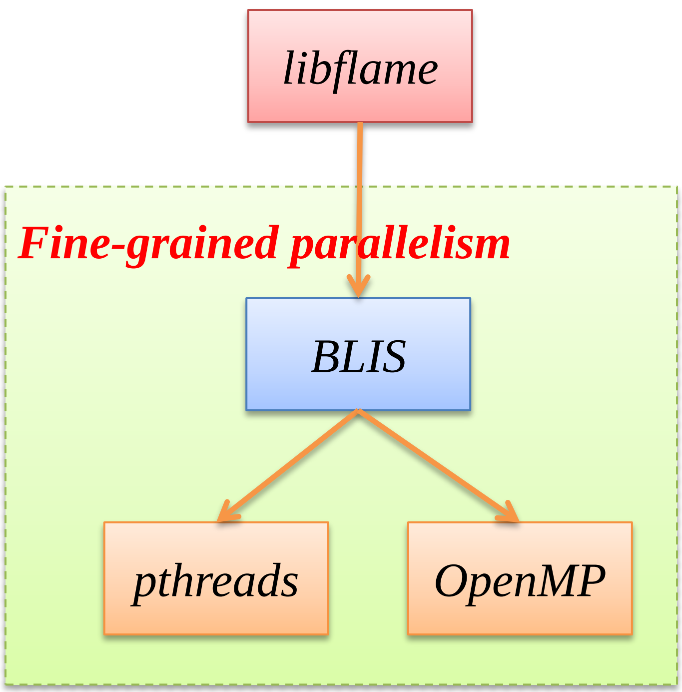
- Synchronization point overhead
- Not fit for multiple devices scenarios.

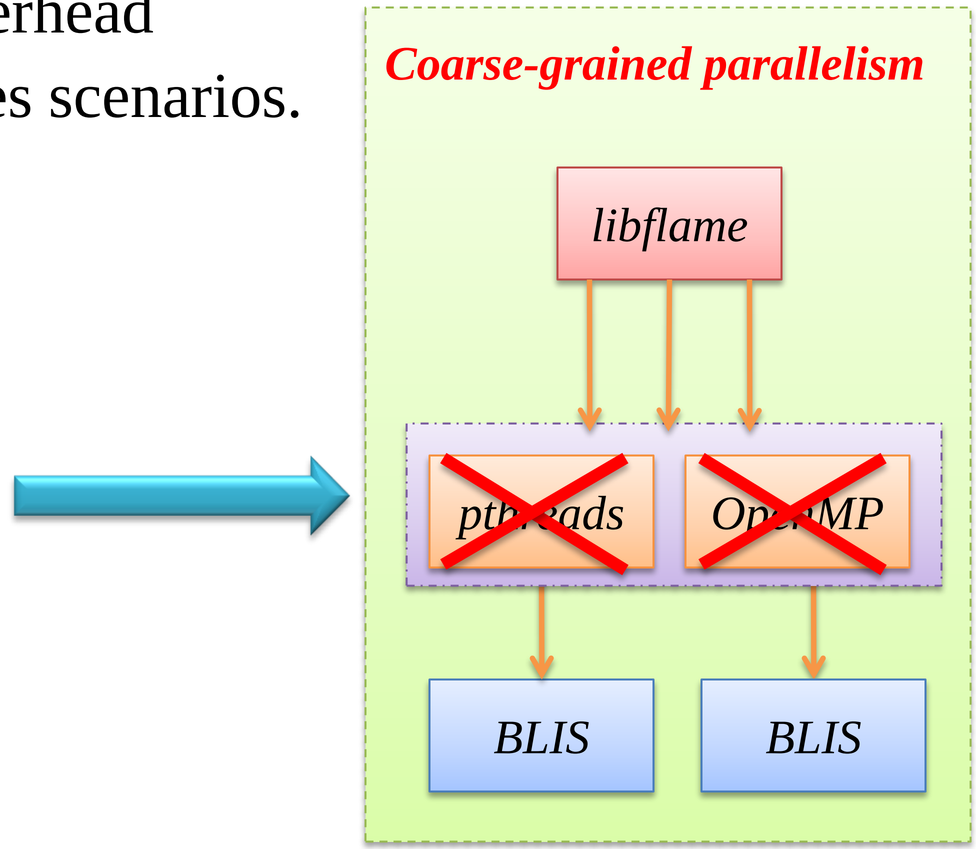


- Introduce parallelism across instructions
- Fit for the platform with multiple computation units.

Problem for Fine-grained Parallelism

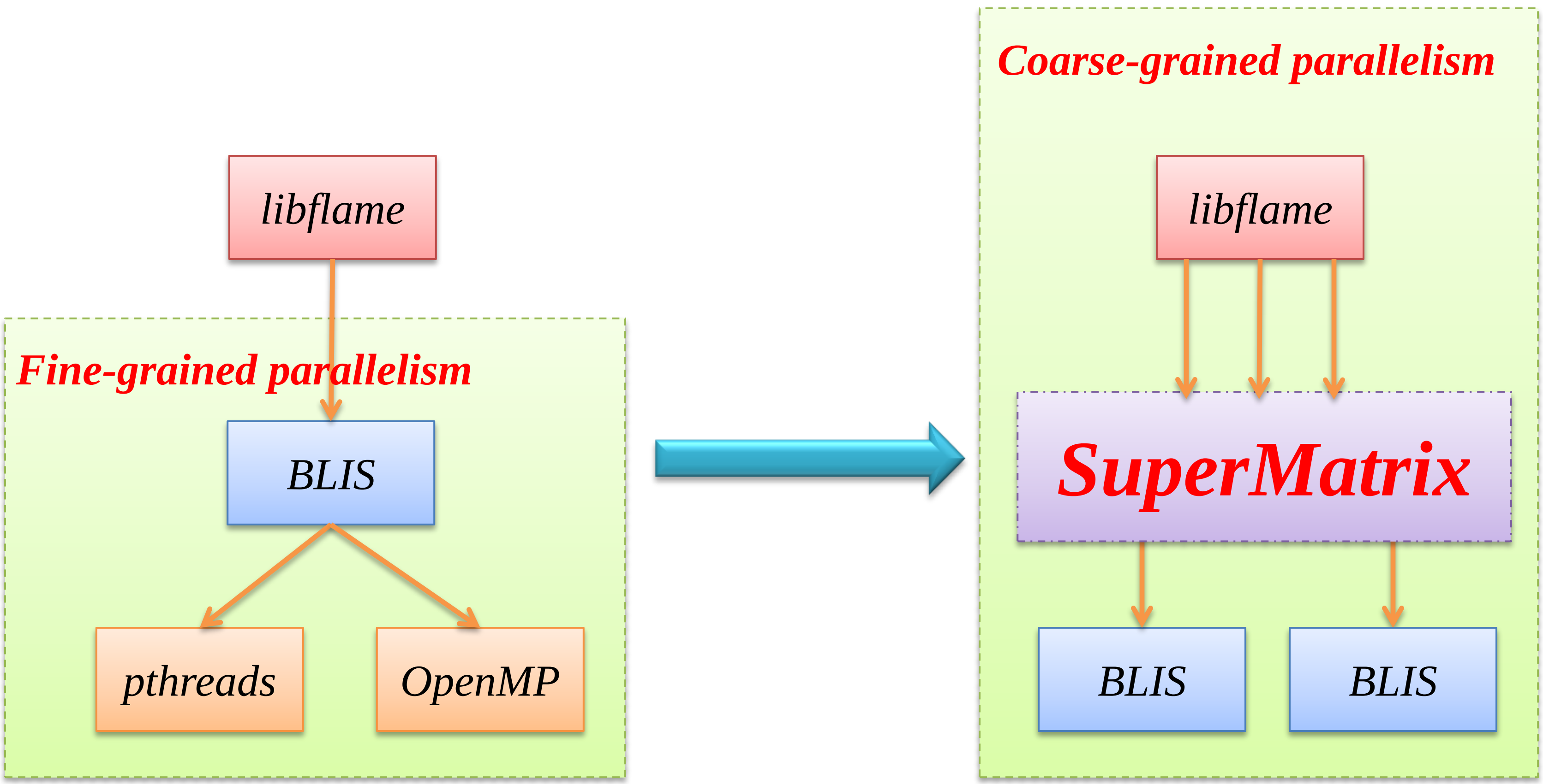
- Synchronization point overhead
- Not fit for multiple devices scenarios.





- Introduce parallelism across instructions
- Fit for the platform with multiple computation units.

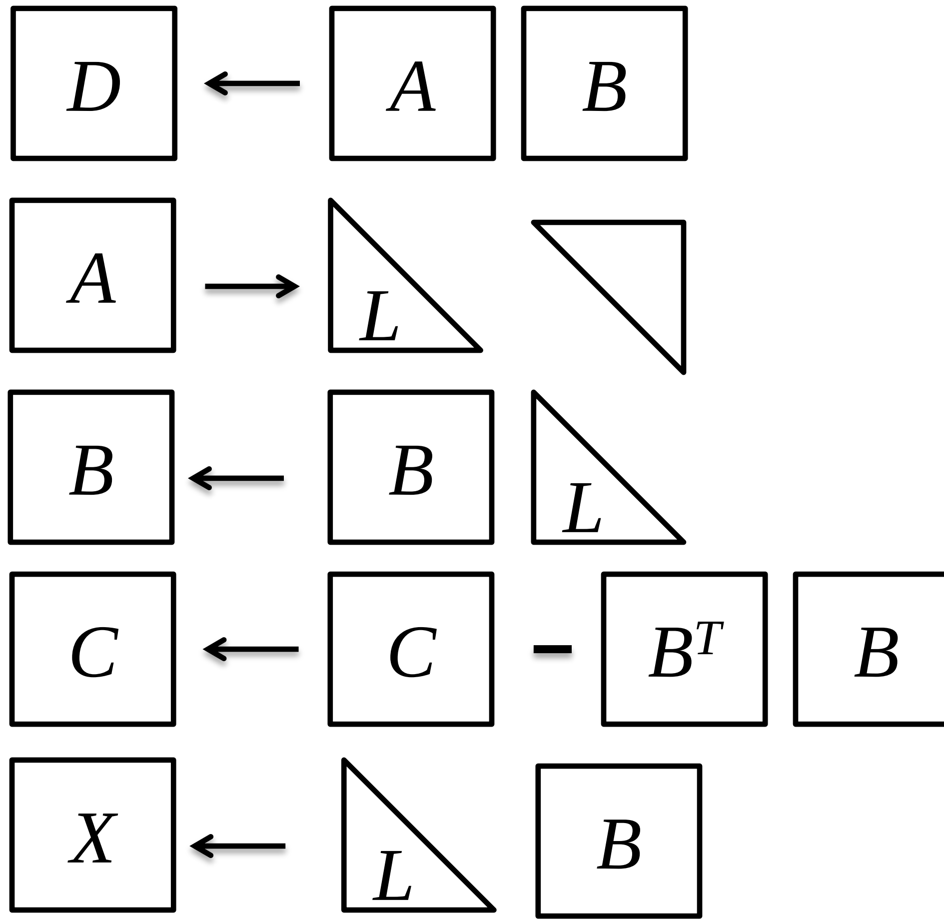
Coarse-grained Parallelism



- Introduce parallelism across instructions
- Fit for the platform with multiple computation units.

SuperMatrix Approach

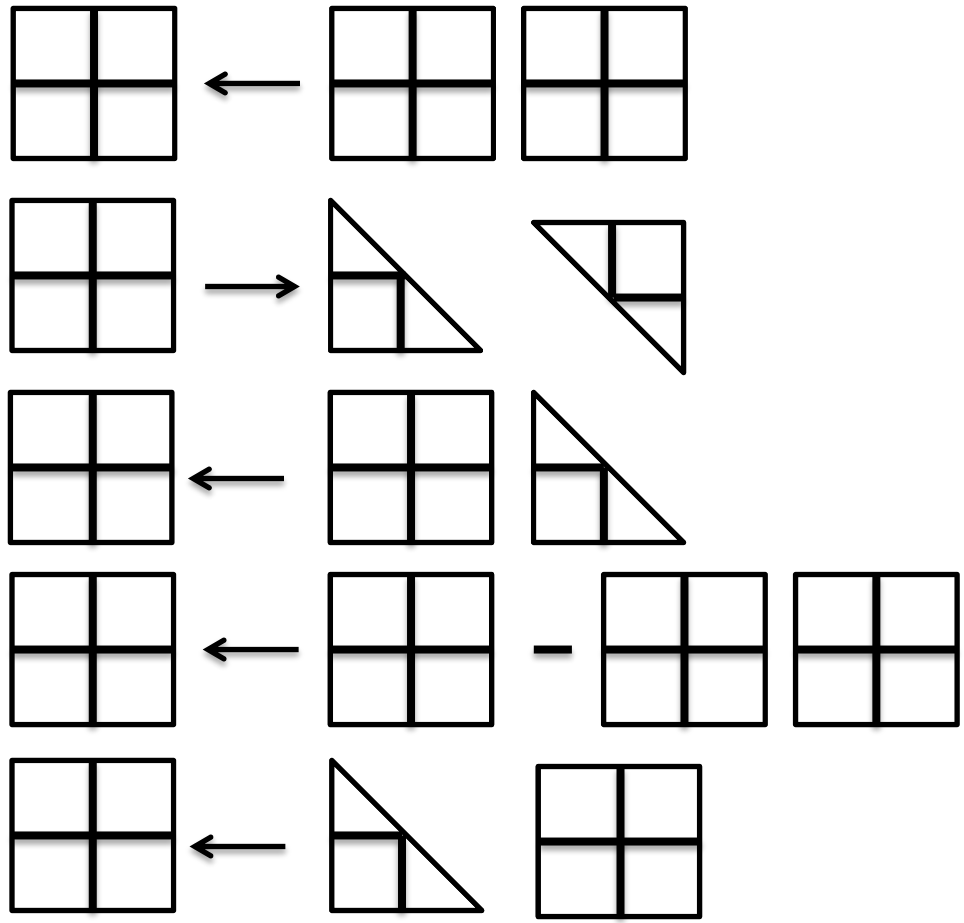
- S0: $D \leftarrow A*B$
- S1: $A \rightarrow L * L^T$
- S2: $B \leftarrow B * L^{-T}$
- S3: $C \leftarrow C - B * B^T$
- S4: $X \leftarrow L^{-1} * X$



How to parallelize?

SuperMatrix Approach

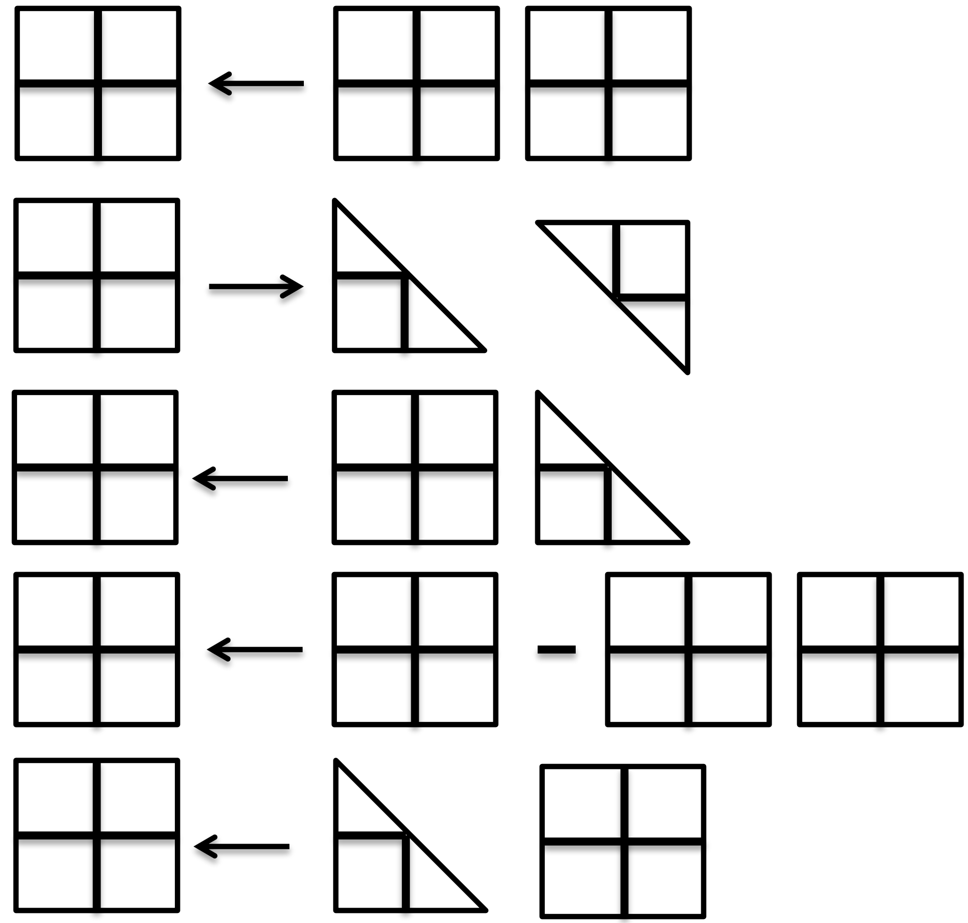
- S0: $D \leftarrow A*B$
- S1: $A \rightarrow L * L^T$
- S2: $B \leftarrow B * L^{-T}$
- S3: $C \leftarrow C - B * B^T$
- S4: $X \leftarrow L^{-1} * X$



How to parallelize?

SuperMatrix Approach

- S0: $D \leftarrow A*B$
- S1: $A \rightarrow L * L^T$
- S2: $B \leftarrow B * L^{-T}$
- S3: $C \leftarrow C - B * B^T$
- S4: $X \leftarrow L^{-1} * X$

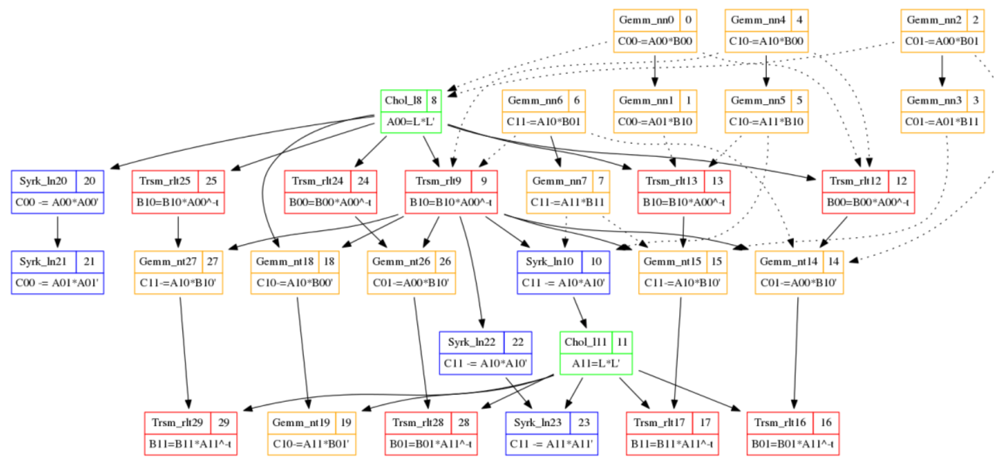


How to parallelize?

Partitioning/Algorithm-by-blocks!

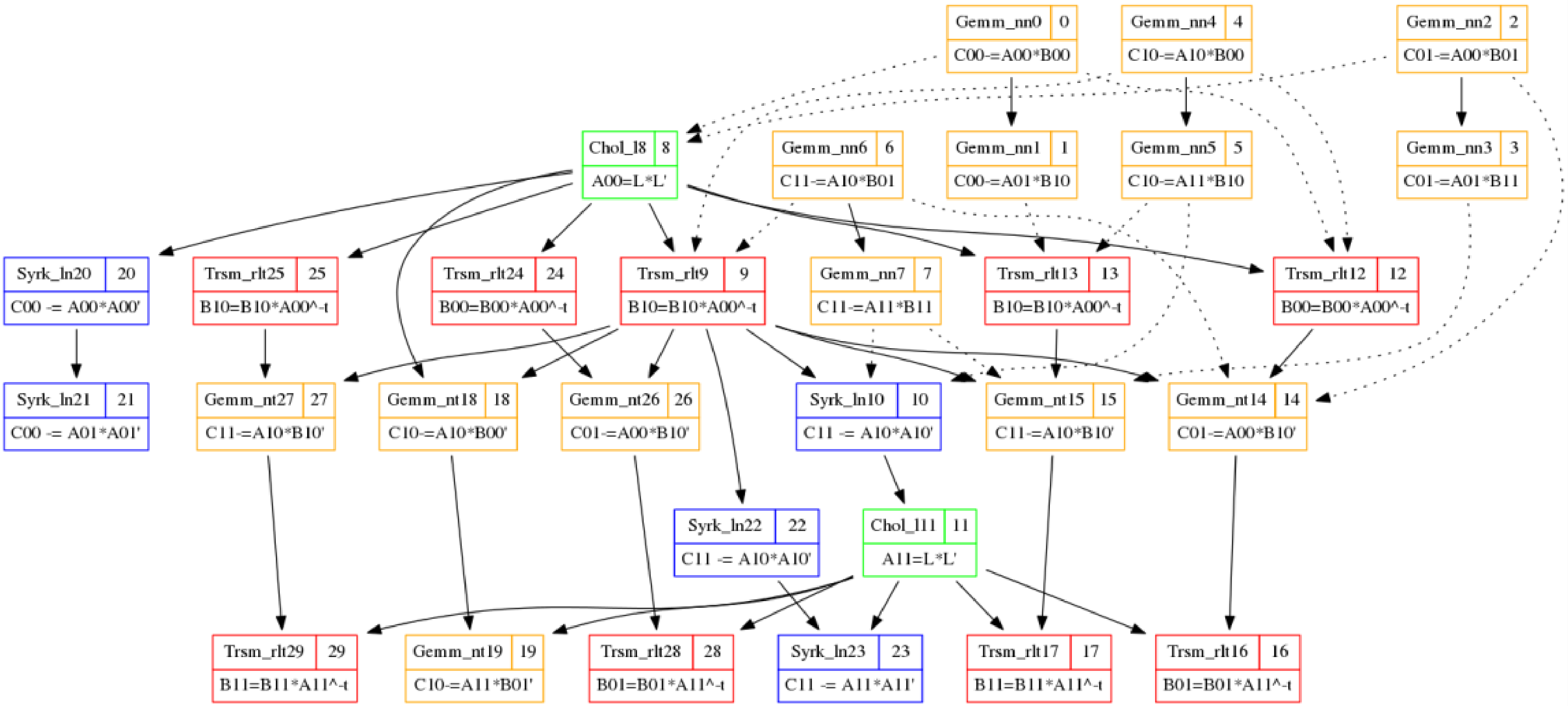
SuperMatrix Approach

- S0: $D \leftarrow A * B$
- S1: $A \rightarrow L * L^T$
- S2: $B \leftarrow B * L^{-T}$
- S3: $C \leftarrow C - B * B^T$
- S4: $X \leftarrow L^{-1} * X$



How to parallelize?

SuperMatrix Approach



- ***Construct the DAG across the instructions automatically***
- ***No need to annotate the task dependencies manually!***

Traditional Library Approach Implemented with libflame and BLIS

S0: $D \leftarrow A*B$

S1: $A \rightarrow L * L^T$

S2: $B \leftarrow B * L^{-T}$

S3: $C \leftarrow C - B * B^T$

S4: $X \leftarrow L^{-1} * X$

```
/*--------------------------------------------------*/
   FLA_Gemm( FLA_NO_TRANSPOSE,FLA_NO_TRANSPOSE,
                FLA_ONE, A, B, FLA_ZERO, D );
   FLA_Chol( FLA_LOWER_TRIANGULAR, A );
   FLA_Trsm( FLA_RIGHT, FLA_LOWER_TRIANGULAR,
                FLA_TRANSPOSE, FLA_NONUNIT_DIAG,
                FLA_ONE, A, B );
   FLA_Syrk( FLA_LOWER_TRIANGULAR, FLA_NO_TRANSPOSE,
                FLA_MINUS_ONE, B, FLA_ONE, C );
   FLA_Trsm( FLA_LEFT, FLA_LOWER_TRIANGULAR,
                FLA_NO_TRANSPOSE, FLA_NONUNIT_DIAG,
                FLA_ONE, L, X );
/*--------------------------------------------------*/
```

Supported by parallel BLAS, LAPACK (**multi-thread BLIS**)

SuperMatrix Approach Implemented with libflame and BLIS

S0: $D \leftarrow A*B$

S1: $A \rightarrow L * L^T$

S2: $B \leftarrow B * L^{-T}$

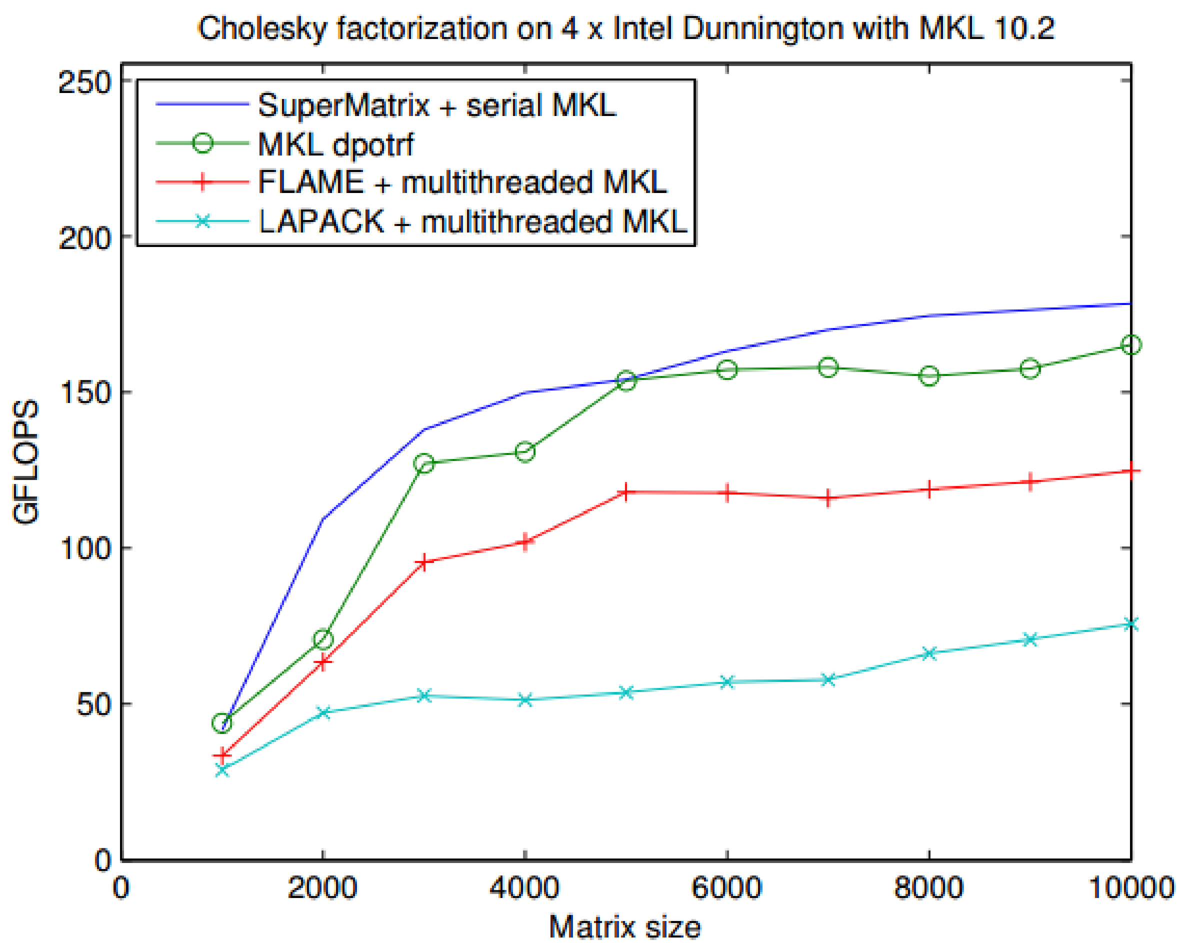
S3: $C \leftarrow C - B * B^T$

S4: $X \leftarrow L^{-1} * X$

```
/*-----------------------------------------*/
FLASH_Gemm( FLA_NO_TRANSPOSE,FLA_NO_TRANSPOSE,
              FLA_ONE, A, B, FLA_ZERO, D );
FLASH_Chol( FLA_LOWER_TRIANGULAR, A );
FLASH_Trsm( FLA_RIGHT, FLA_LOWER_TRIANGULAR,
              FLA_TRANSPOSE, FLA_NONUNIT_DIAG,
              FLA_ONE, A, B );
FLASH_Syrk( FLA_LOWER_TRIANGULAR, FLA_NO_TRANSPOSE,
              FLA_MINUS_ONE, B, FLA_ONE, C );
FLASH_Trsm( FLA_LEFT, FLA_LOWER_TRIANGULAR,
              FLA_NO_TRANSPOSE, FLA_NONUNIT_DIAG,
              FLA_ONE, L, X );
/*-----------------------------------------*/
```

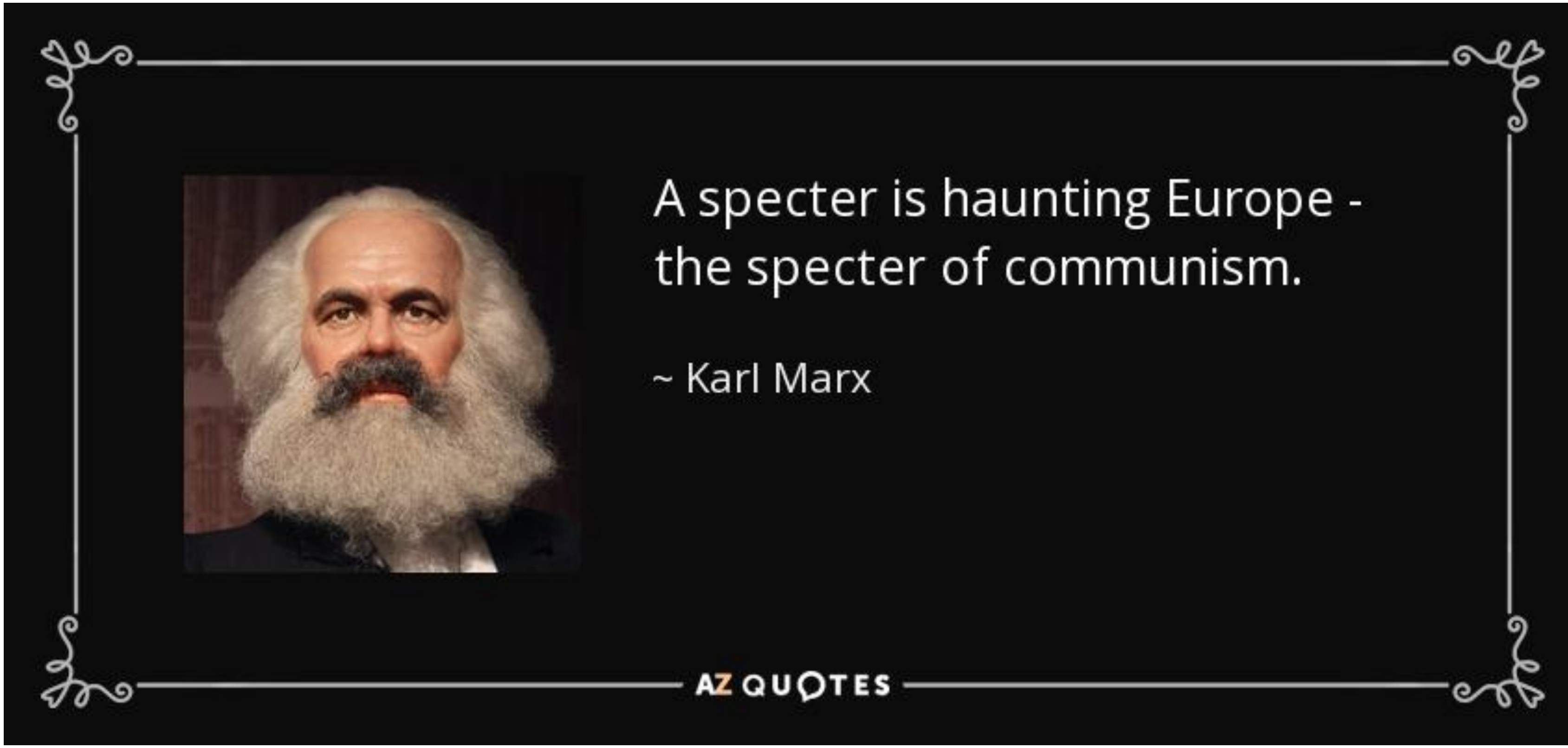
Tiny Code Change!

Free Lunch for Both Programmability and Performance!



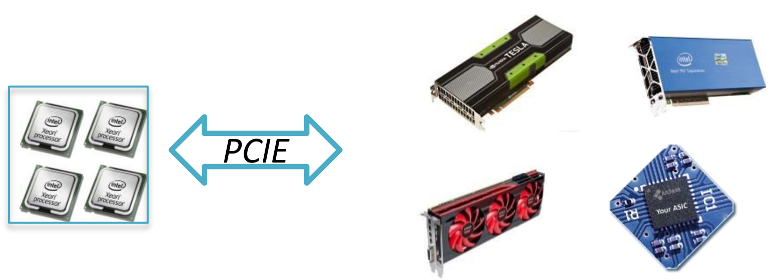
From libflame manual, 2011

Original SuperMatrix primarily targets at multi-core shared memory system…


A specter is haunting Europe -
the specter of communism.
~ Karl Marx
AZ QUOTES


A specter is haunting HPC -
Heterogeneous Platforms
~ Karl Matrix
AZ QUOTES


A specter is haunting HPC -
Heterogeneous Platforms
~ Karl Matrix
AZ QUOTES


PCIE

Challenges in Heterogeneous Platforms!

- S0: $D \leftarrow A*A^T$
- S1: $A \rightarrow L * L^T$
- S2: $B \leftarrow B * L^{-T}$
- S3: $C \leftarrow C - B * B^T$
- S4: $X \leftarrow L^{-1} * X$

- S0: ParGemm (A,A^T,D)
- S1: L = ParPortf(A)
- S2: ParTrsm(L,B)
- S3: ParSyrk(B,C)
- S4: ParTrsm(L,X)

What if there is one accelerator in your system?

Challenges in Heterogeneous Platforms!

- S0: $D \leftarrow A * A^T$
- S1: $A \rightarrow L * L^T$
- S2: $B \leftarrow B * L^{-T}$
- S3: $C \leftarrow C - B * B^T$
- S4: $X \leftarrow L^{-1} * X$

- S0: ParGemm (A,A^T,D)
- S1: L = ParPortf(A)
- S2: ParTrsm(L,B)
- S3: ParSyrk(B,C)
- S4: ParTrsm(L,X)

What if there is one accelerator in your system?

Challenges in Heterogeneous Platforms!

- S0: ParGemm (A,A^T,D)
- S1: L = ParPortf(A)
- S2: ParTrsm(L,B)
- S3: ParSyrk(B,C)
- S4: ParTrsm(L,X)

What if there is one accelerator in your system?

Challenges in Heterogeneous Platforms!

```
/*----------------------------*/
   Memcpy(A, hA);
   Memcpy(D, hD);
   Memcpy(B, hB);
   Memcpy(C, hC);
   Memcpy(X, hX);
/*----------------------------*/
```

- S0: ParGemm (A,A^T,D)
- S1: L = ParPortf(A)
- S2: ParTrsm(L,B)
- S3: ParSyrk(B,C)
- S4: ParTrsm(L,X)

```
/*----------------------------*/
Memcpy(hX, X);
/*----------------------------*/
```

What if there is one accelerator in your system?

Challenges in Heterogeneous Platforms!

```
/*---------------------------*/
   Memcpy(A, hA);
   Memcpy(D, hD);
   Memcpy(B, hB);
   Memcpy(C, hC);
   Memcpy(X, hX);
/*---------------------------*/
```

- S0: ParGemm (A, A^T, D)
- S1: L = ParPortf(A)
- S2: ParTrsm(L,B)
- S3: ParSyrk(B,C)
- S4: ParTrsm(L,X)

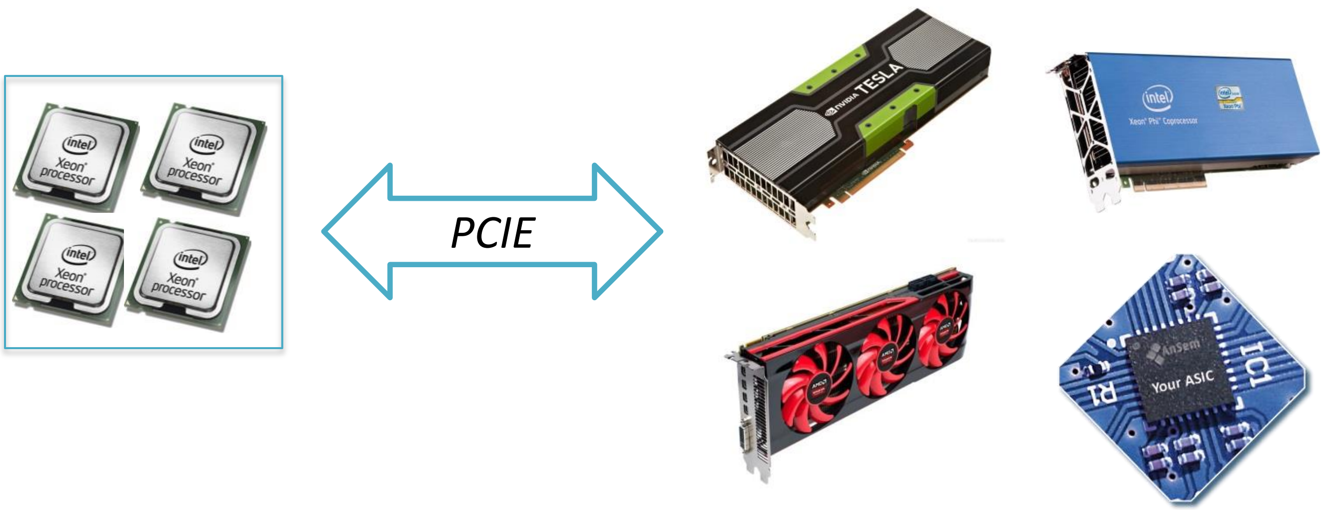
```
/*---------------------------*/
Memcpy(hX, X);
/*---------------------------*/
```

What if there are 4 GPUs and 8 CPU cores in your system?

Adapting Original SuperMatrix to Heterogeneous Platforms

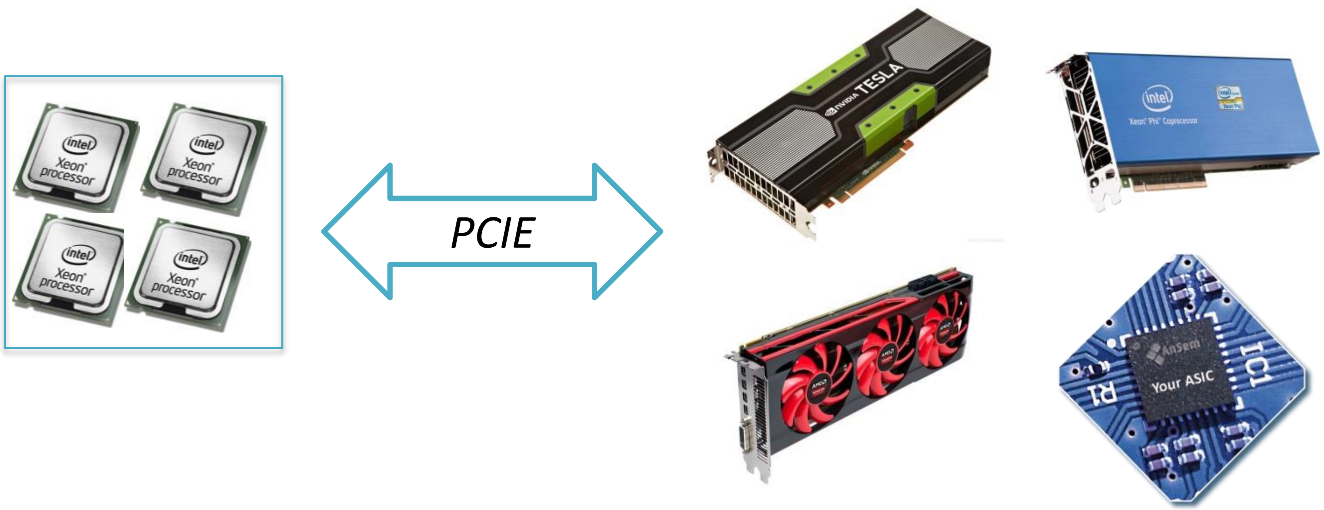
- Software Cache
- Heterogeneous Scheduler
- Asynchronous Memory Copy
- Worker Task Performance Model

Naïve Approach



Naïve Approach

$C \leftarrow A \; B$

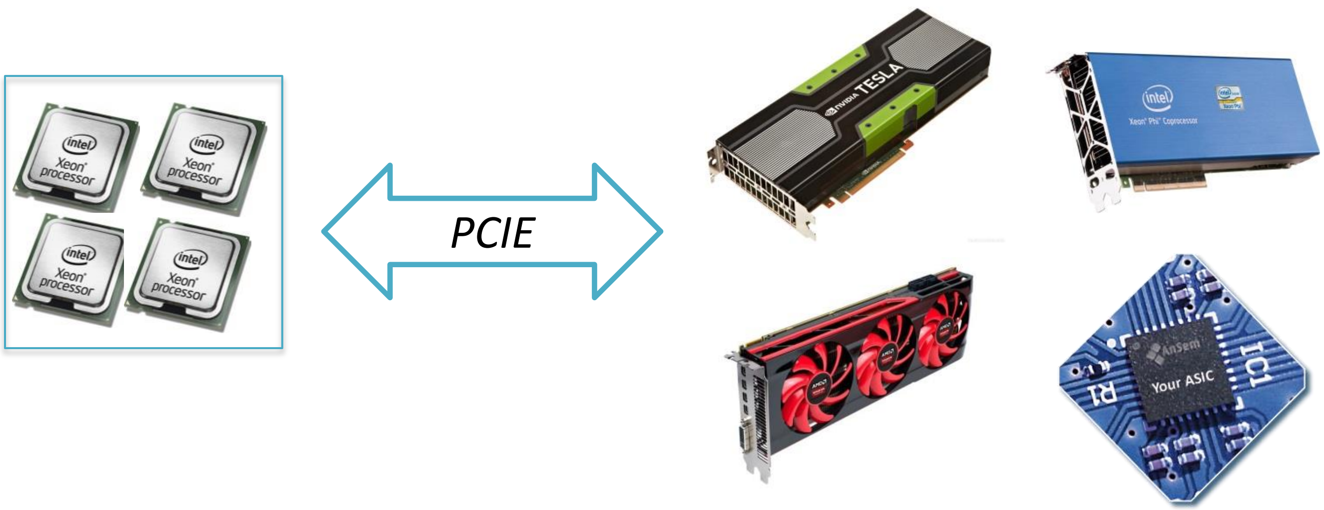


Naïve Approach

Transfer data from host to device before execution

C ←

A B



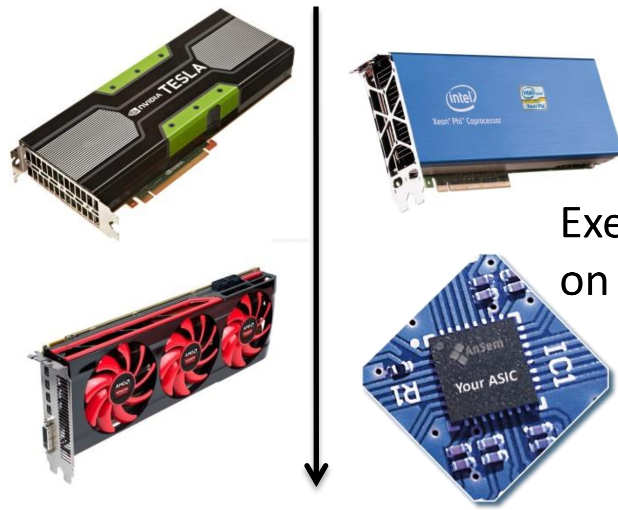
Naïve Approach

Transfer data from host to device before execution

C

A B

PCIE



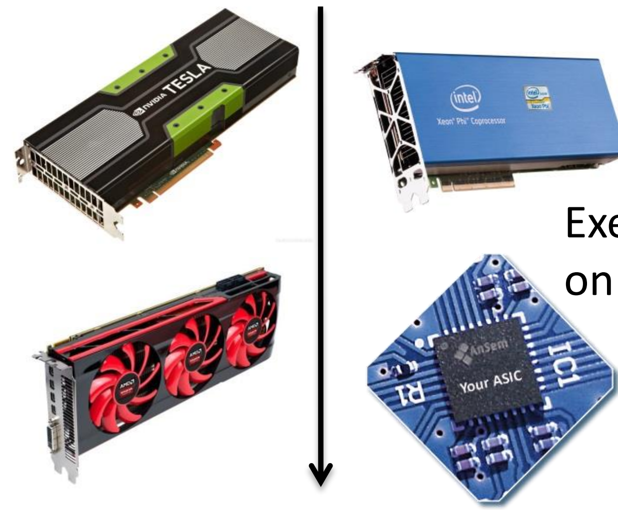
Execute the task on the device

Naïve Approach

Transfer data from host to device before execution

A B

PCIE



Execute the task on the device

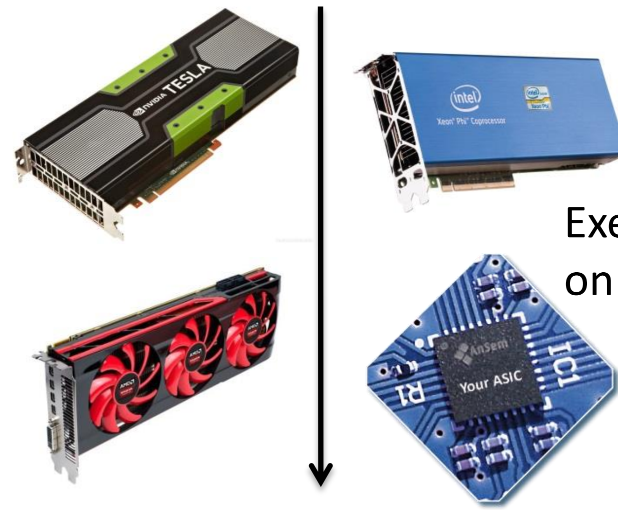
C

Naïve Approach

Transfer data from host to device before execution

A B

PCIE



Execute the task on the device

C

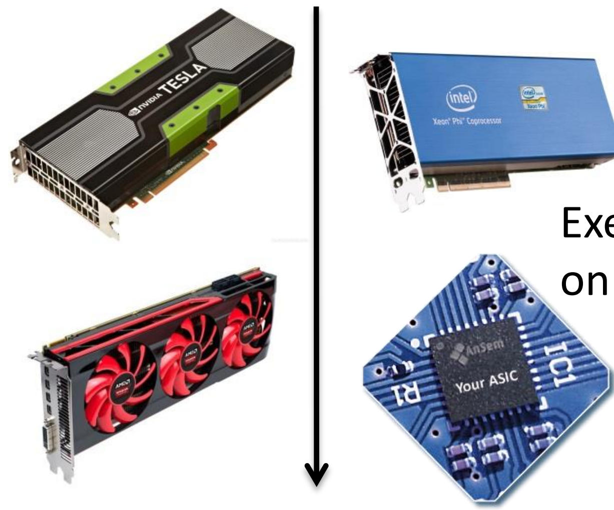
Transfer data from device to host upon execution

Naïve Approach

Transfer data from host to device before execution

A B

PCIE



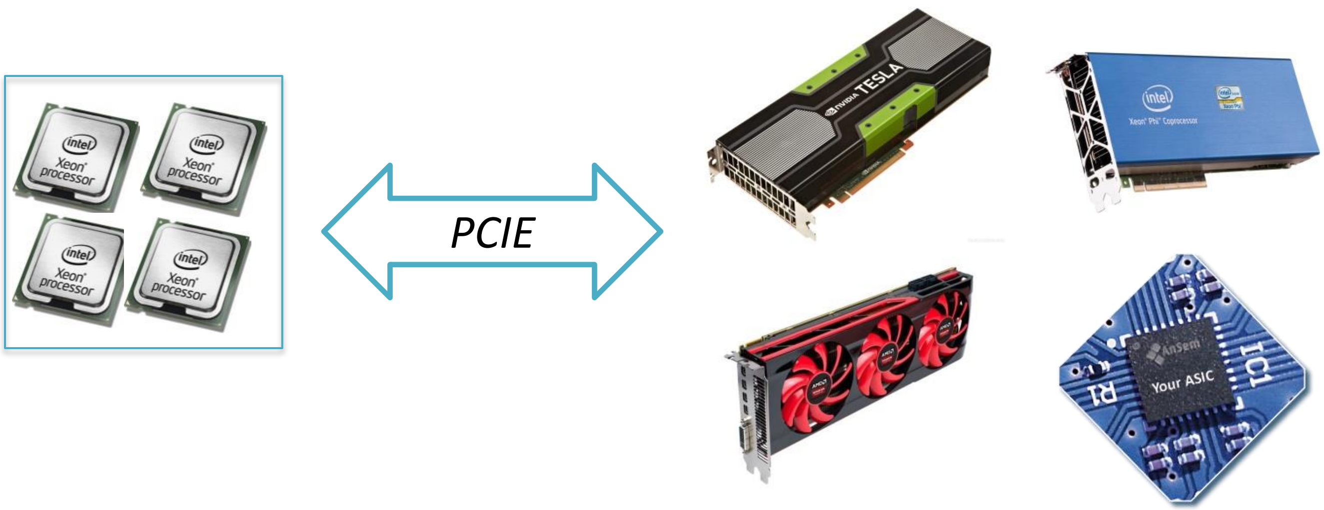
Execute the task on the device

C

Transfer data from device to host upon execution

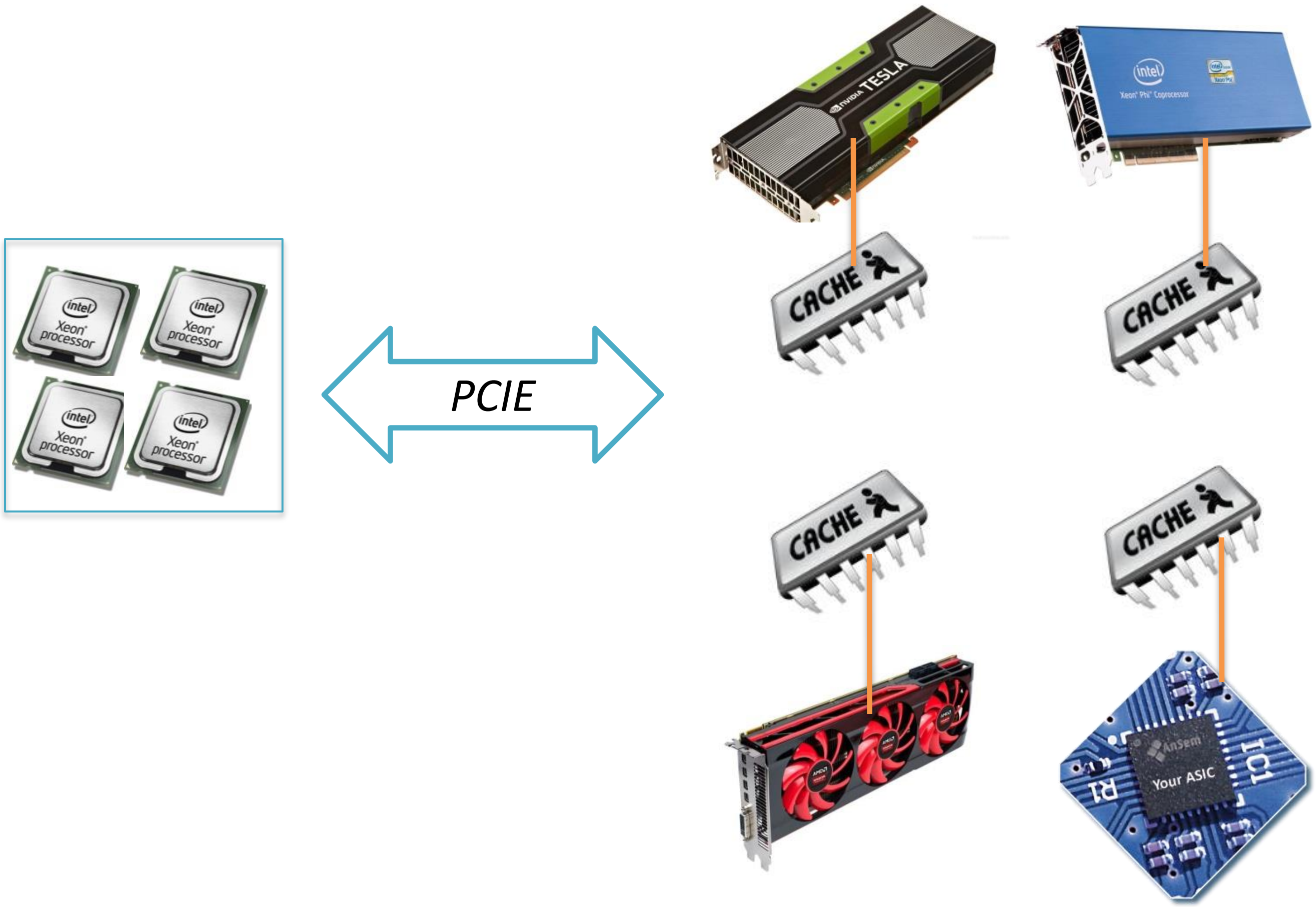
No Data Reuse on the devices !

Software Cache



Quintana-Ortí, G., et al. "Solving dense linear systems on platforms with multiple hardware accelerators." In *PPoPP '09 Proceedings of the 14th ACM SIGPLAN symposium on Principles and Practice of Parallel Programming*, 2009

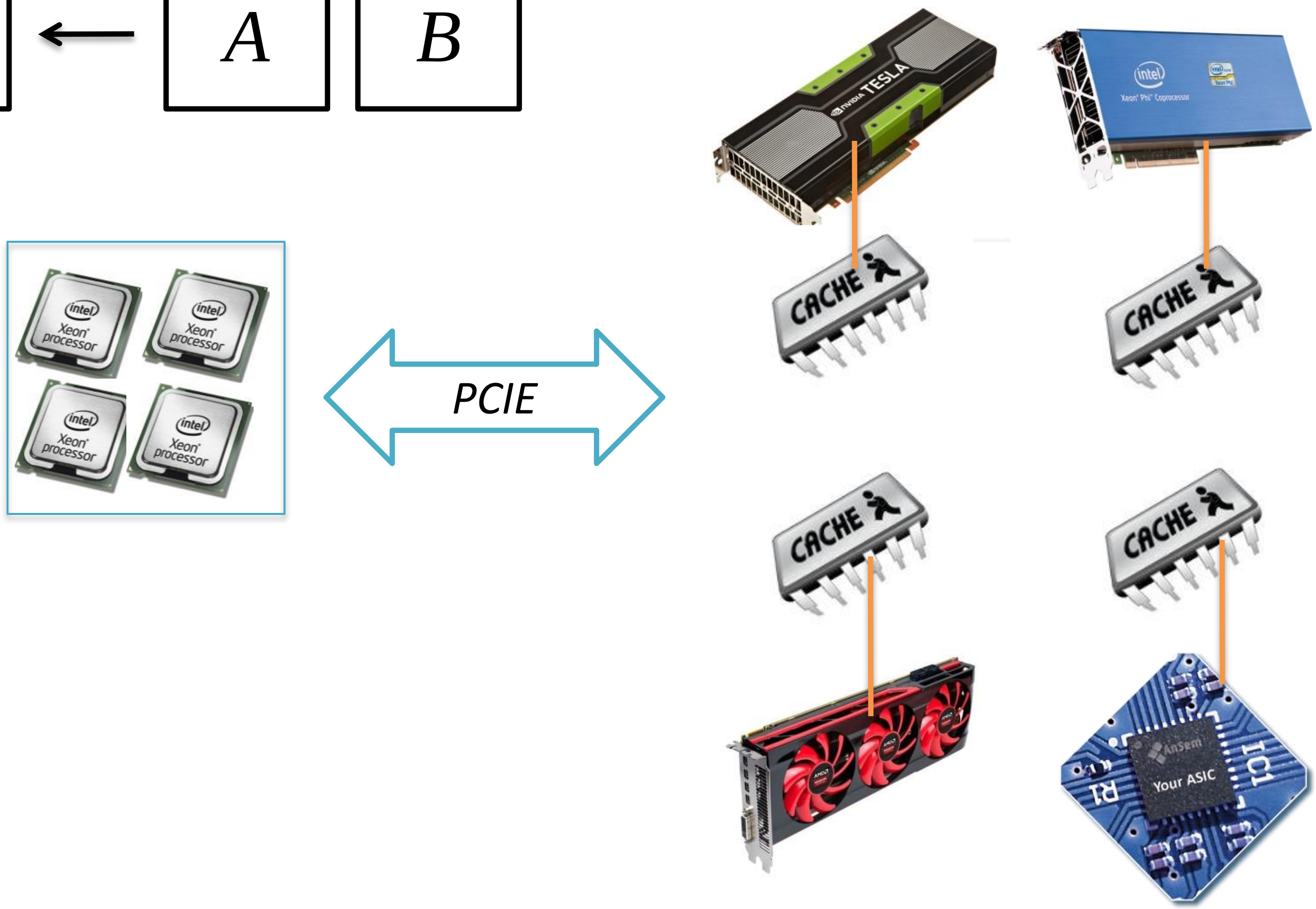
Software Cache



Quintana-Ortí, G., et al. "Solving dense linear systems on platforms with multiple hardware accelerators." In *PPoPP '09 Proceedings of the 14th ACM SIGPLAN symposium on Principles and Practice of Parallel Programming*, 2009

Software Cache

$C \leftarrow A \; B$

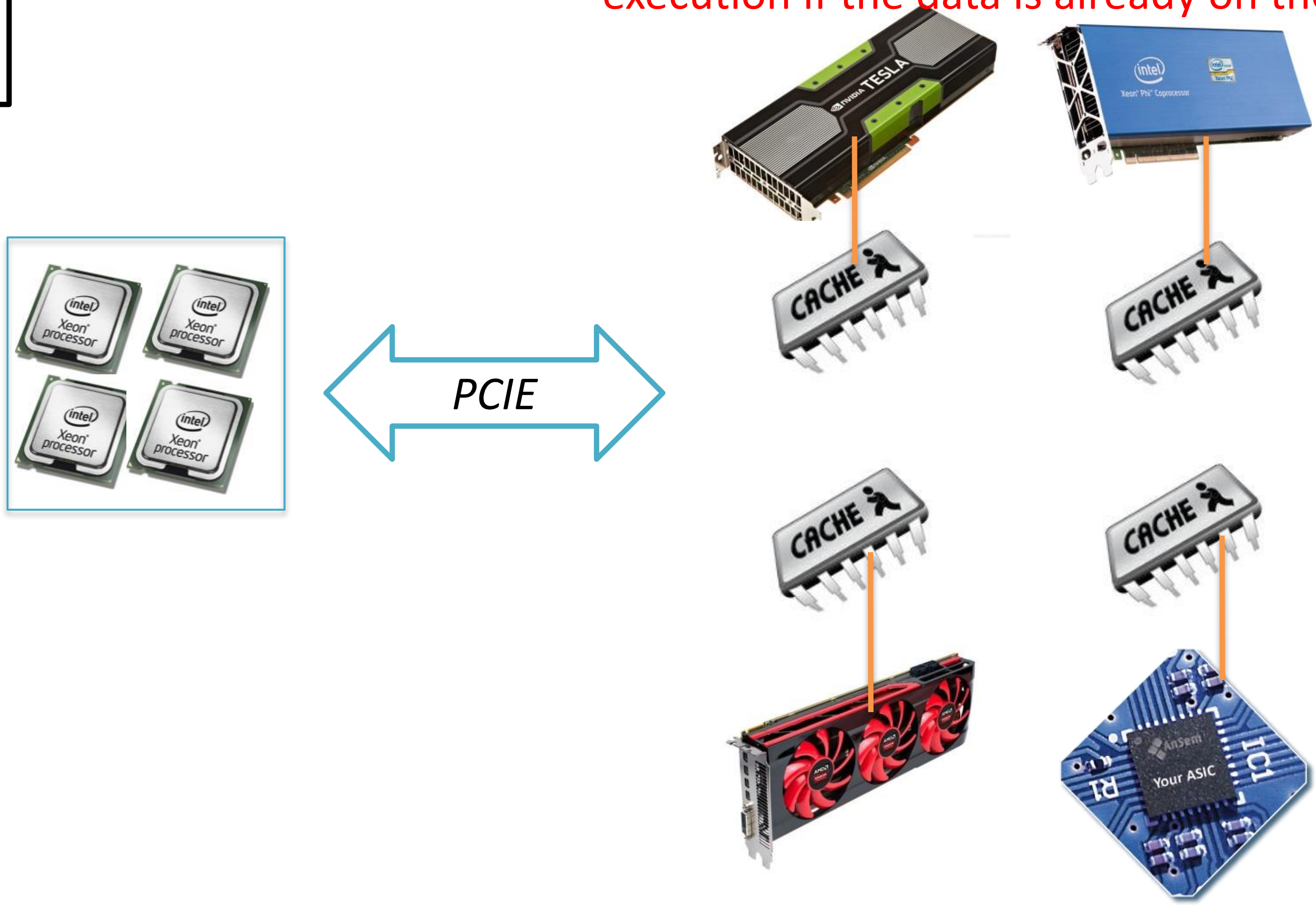


Quintana-Ortí, G., et al. "Solving dense linear systems on platforms with multiple hardware accelerators." In *PPoPP '09 Proceedings of the 14th ACM SIGPLAN symposium on Principles and Practice of Parallel Programming*, 2009

Software Cache

$\boxed{C}$

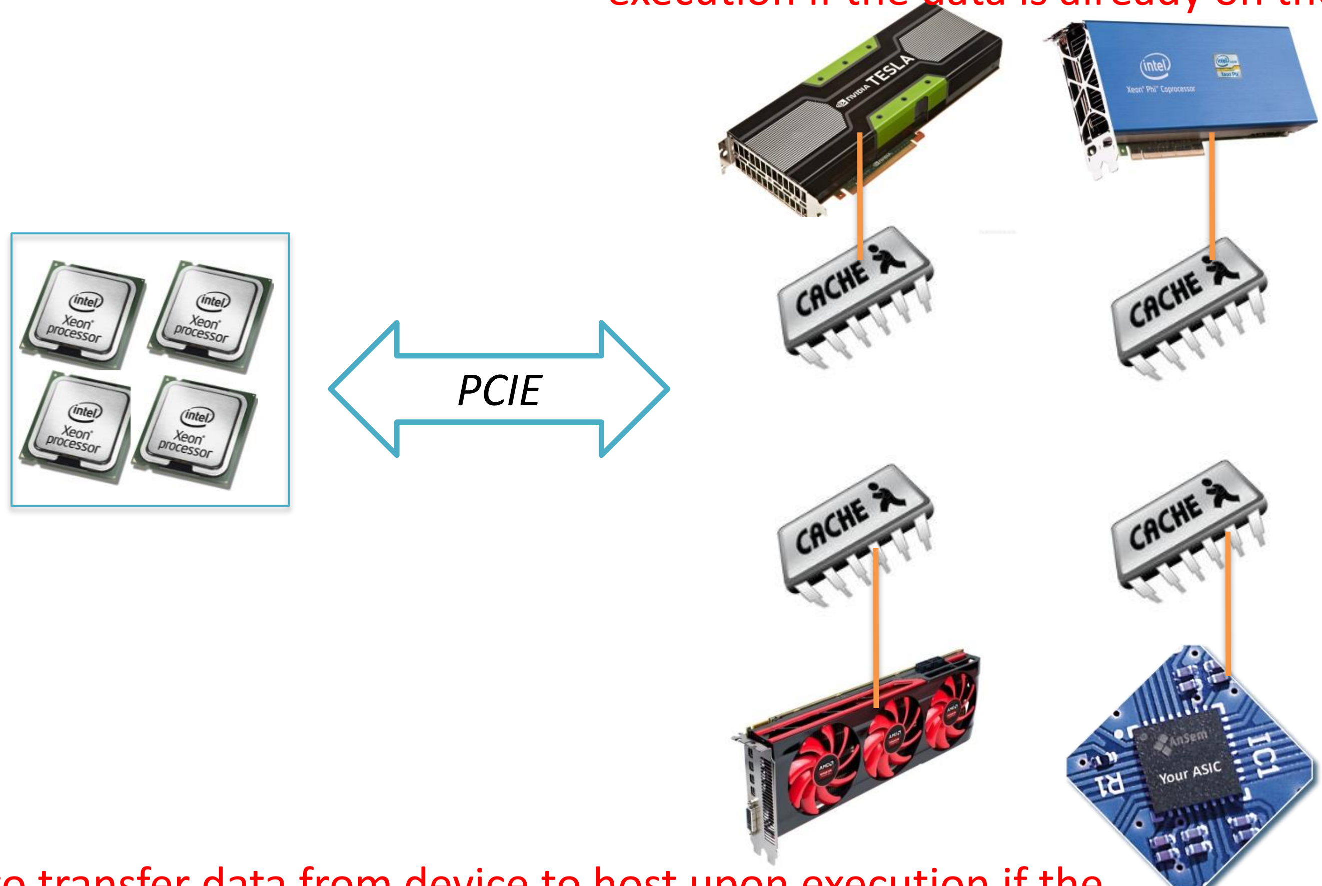
No need to transfer data from host to device before execution if the data is already on the device



Quintana-Ortí, G., et al. "Solving dense linear systems on platforms with multiple hardware accelerators." In *PPoPP '09 Proceedings of the 14th ACM SIGPLAN symposium on Principles and Practice of Parallel Programming*, 2009

Software Cache

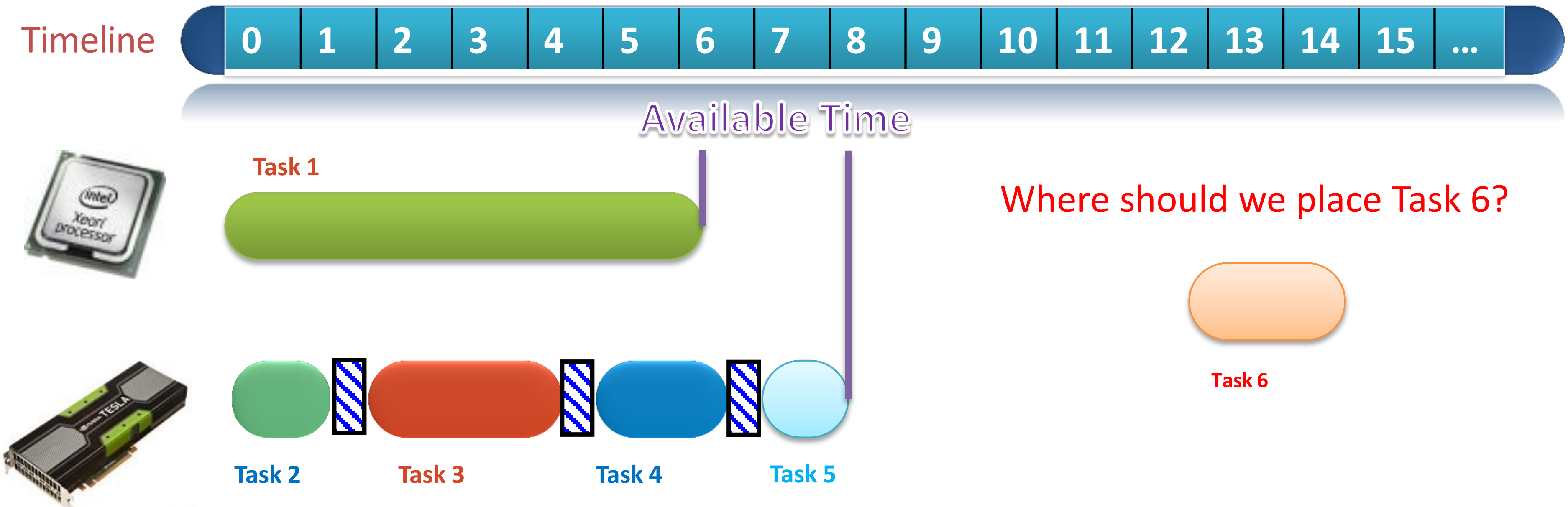
No need to transfer data from host to device before execution if the data is already on the device



No need to transfer data from device to host upon execution if the data is not required by the host immediately

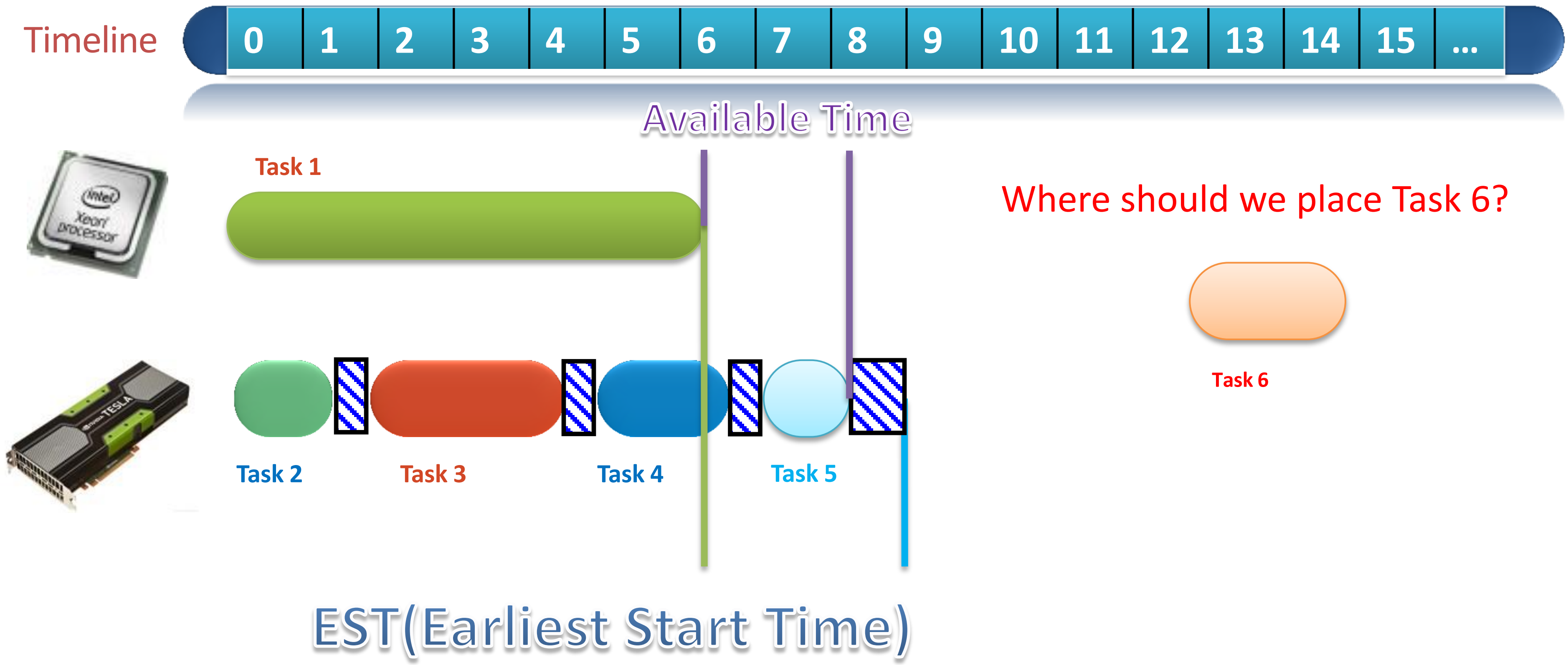
Quintana-Ortí, G., et al. "Solving dense linear systems on platforms with multiple hardware accelerators." In *PPoPP '09 Proceedings of the 14th ACM SIGPLAN symposium on Principles and Practice of Parallel Programming*, 2009

HEFT(Heterogeneous Earliest Finish Time)



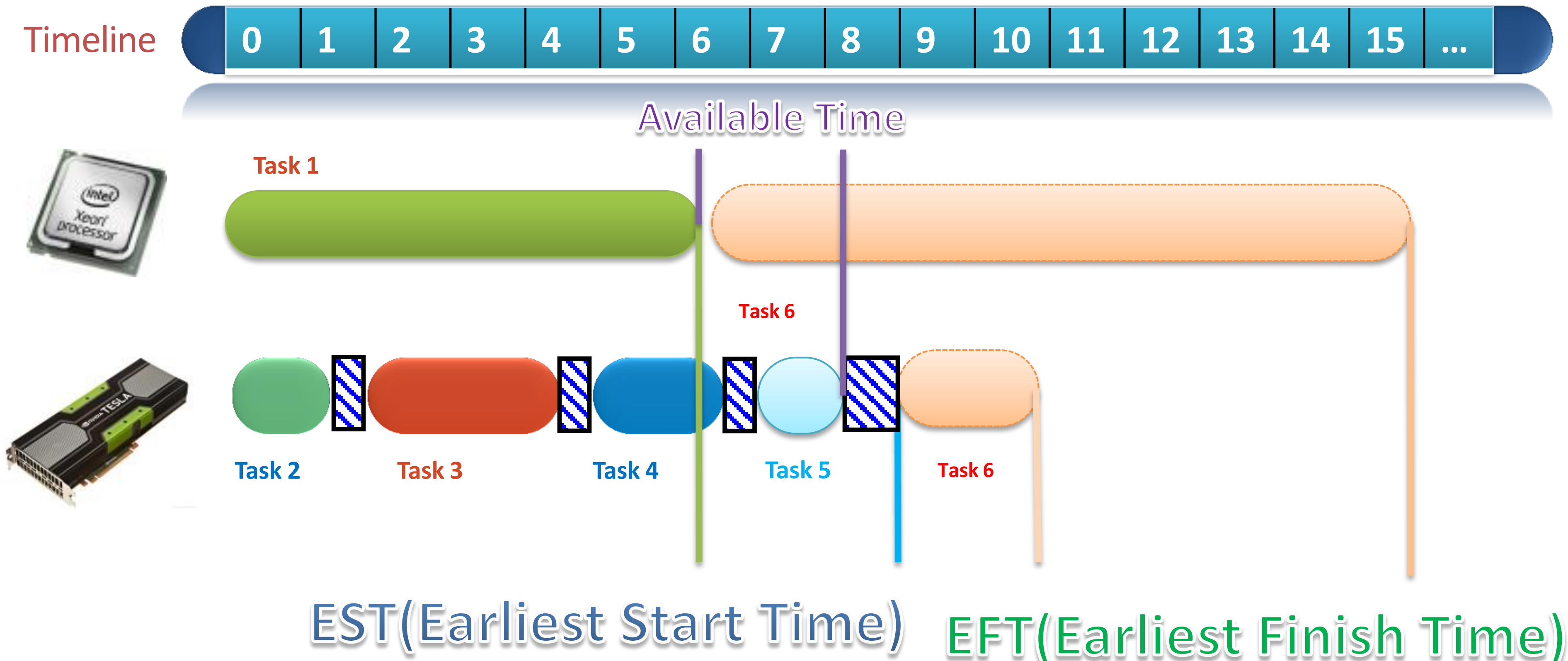
Topcuoglu, H., Hariri, S., and Wu, M.. "Performance-effective and low-complexity task scheduling for heterogeneous computing." *IEEE Transactions on Parallel and Distributed Systems,* 13.3 (2002): 260-274.

HEFT(Heterogeneous Earliest Finish Time)



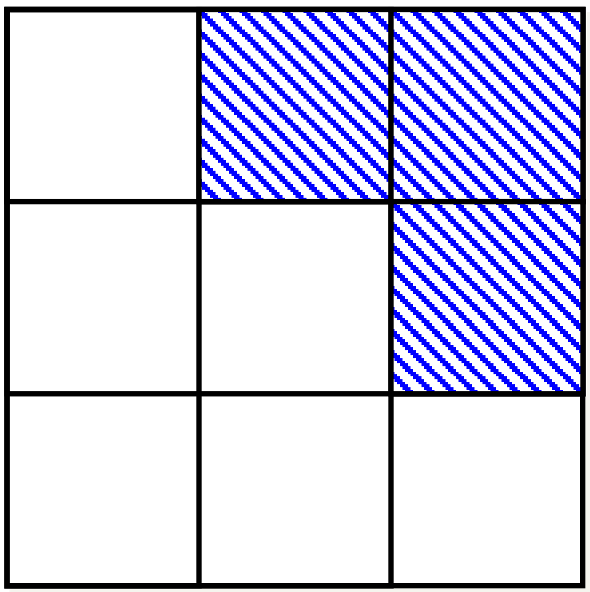
Topcuoglu, H., Hariri, S., and Wu, M.. "Performance-effective and low-complexity task scheduling for heterogeneous computing." *IEEE Transactions on Parallel and Distributed Systems,* 13.3 (2002): 260-274.

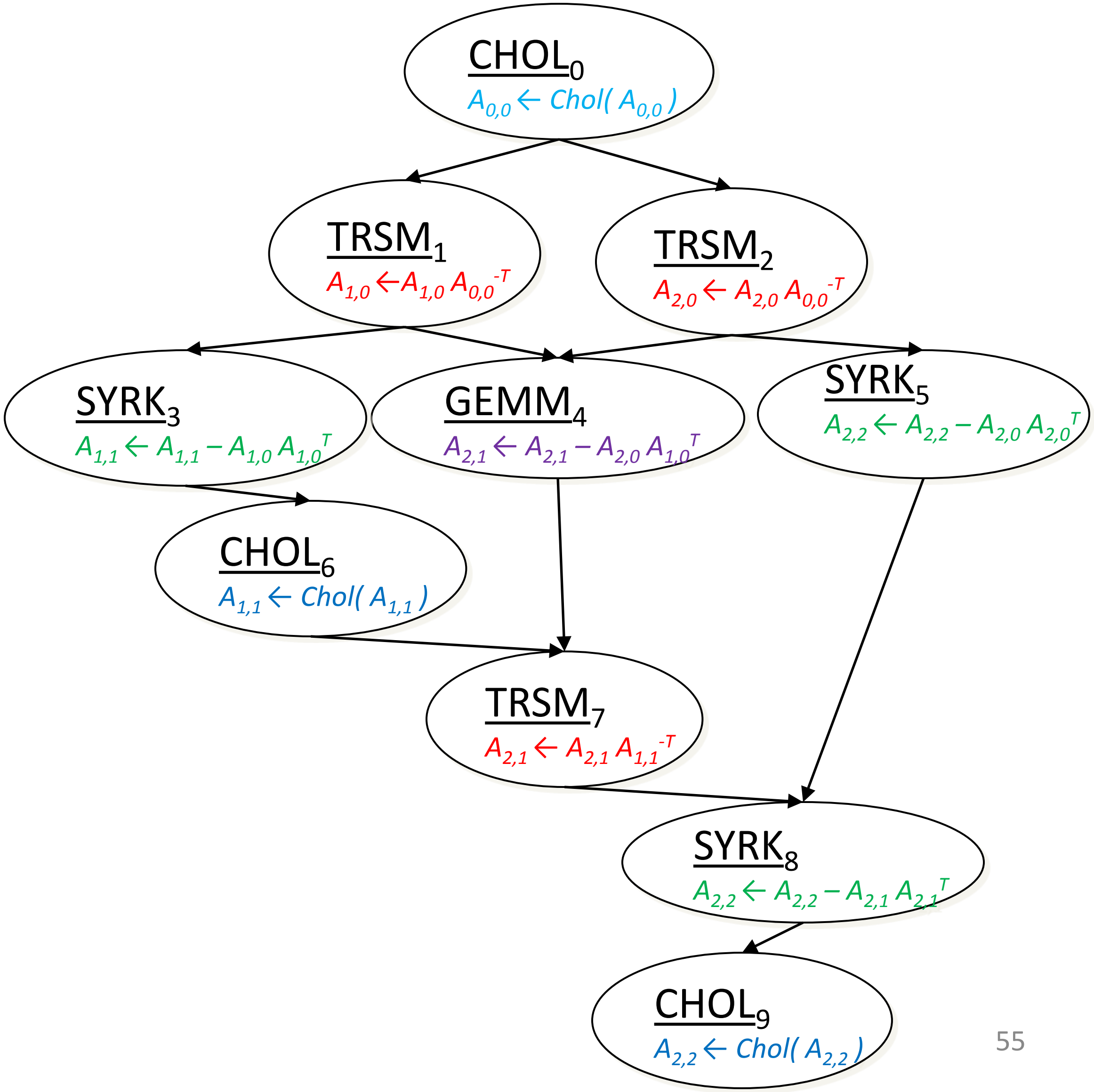
HEFT(Heterogeneous Earliest Finish Time)



Topcuoglu, H., Hariri, S., and Wu, M.. "Performance-effective and low-complexity task scheduling for heterogeneous computing." *IEEE Transactions on Parallel and Distributed Systems,* 13.3 (2002): 260-274.

3x3 Blocked Cholesky Decomposition



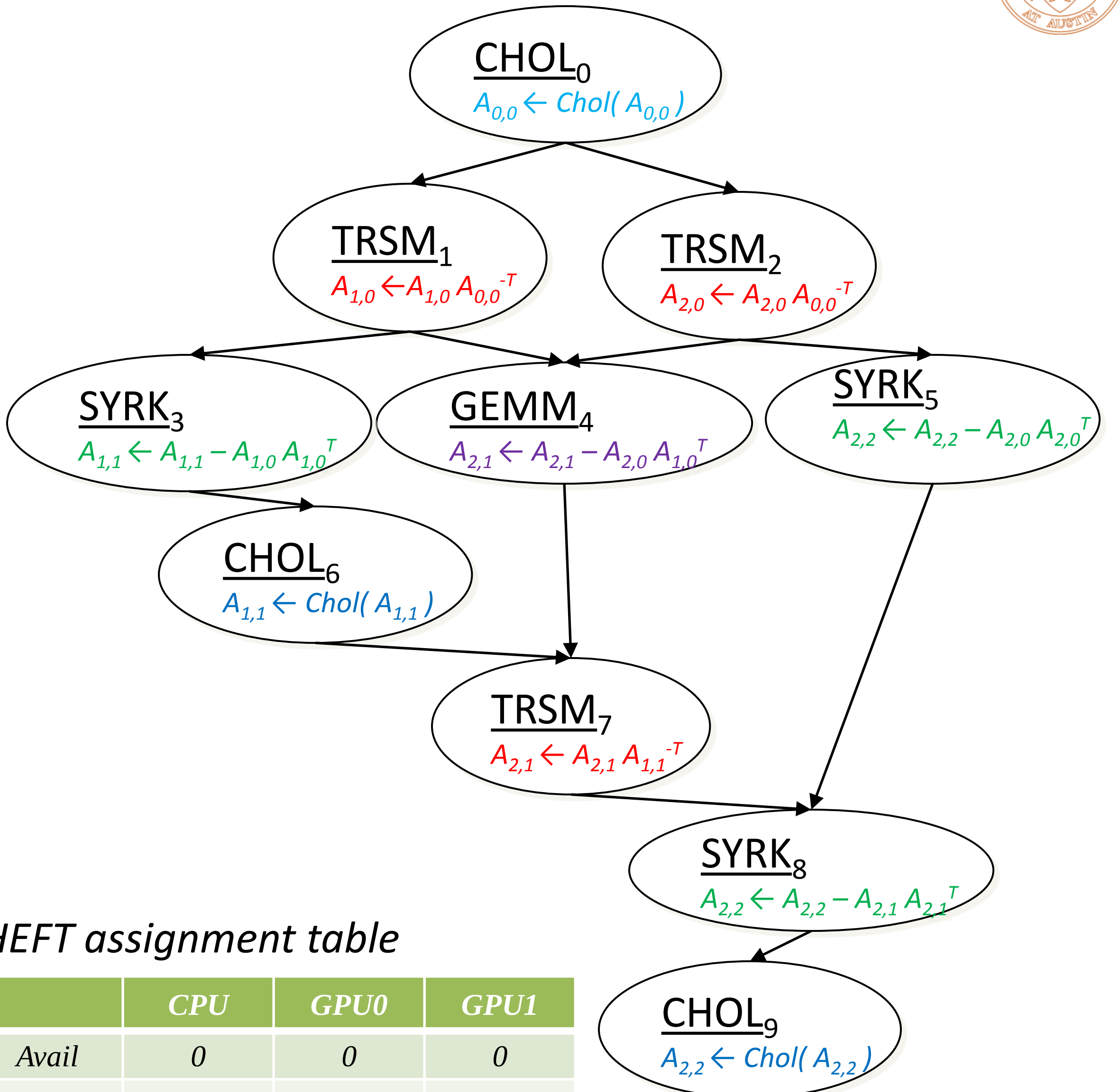


Data Distribution

	CPU	GPU0	GPU1
A00	1	0	0
A10	1	0	0
A11	1	0	0
A20	1	0	0
A21	1	0	0
A22	1	0	0

Scheduler

	CPU	GPU0	GPU1
CHOL0			
TRSM1			
TRSM2			
SYRK3			
GEMM4			
SYRK5			
CHOL6			
TRSM7			
SYRK8			
CHOL9			



HEFT assignment table

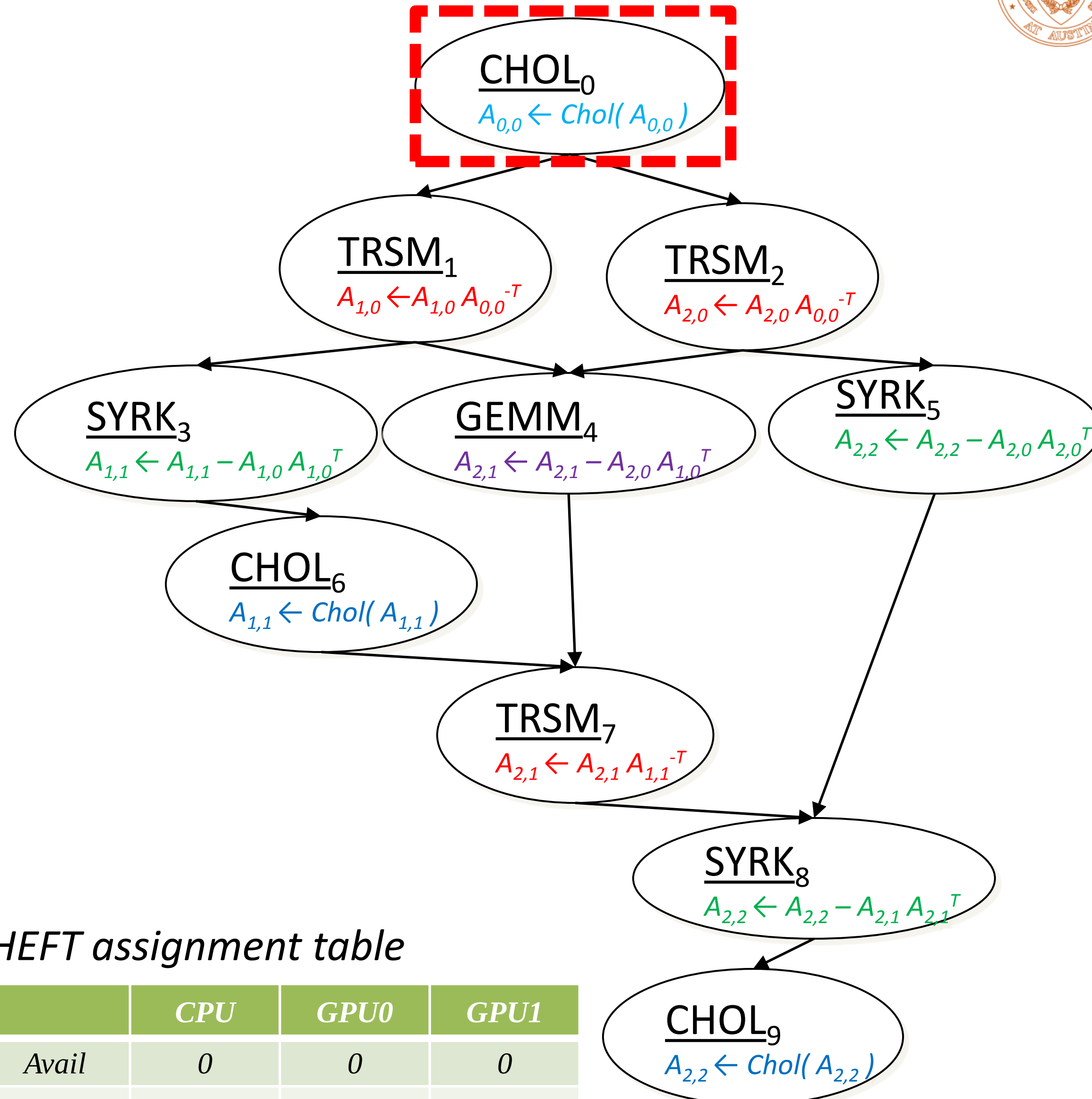
	CPU	GPU0	GPU1
Avail	0	0	0
EST			
EFT			
Priority			

Data Distribution

	CPU	GPU0	GPU1
A00	1	0	0
A10	1	0	0
A11	1	0	0
A20	1	0	0
A21	1	0	0
A22	1	0	0

Scheduler

	CPU	GPU0	GPU1
CHOL0	X		
TRSM1			
TRSM2			
SYRK3			
GEMM4			
SYRK5			
CHOL6			
TRSM7			
SYRK8			
CHOL9			



HEFT assignment table

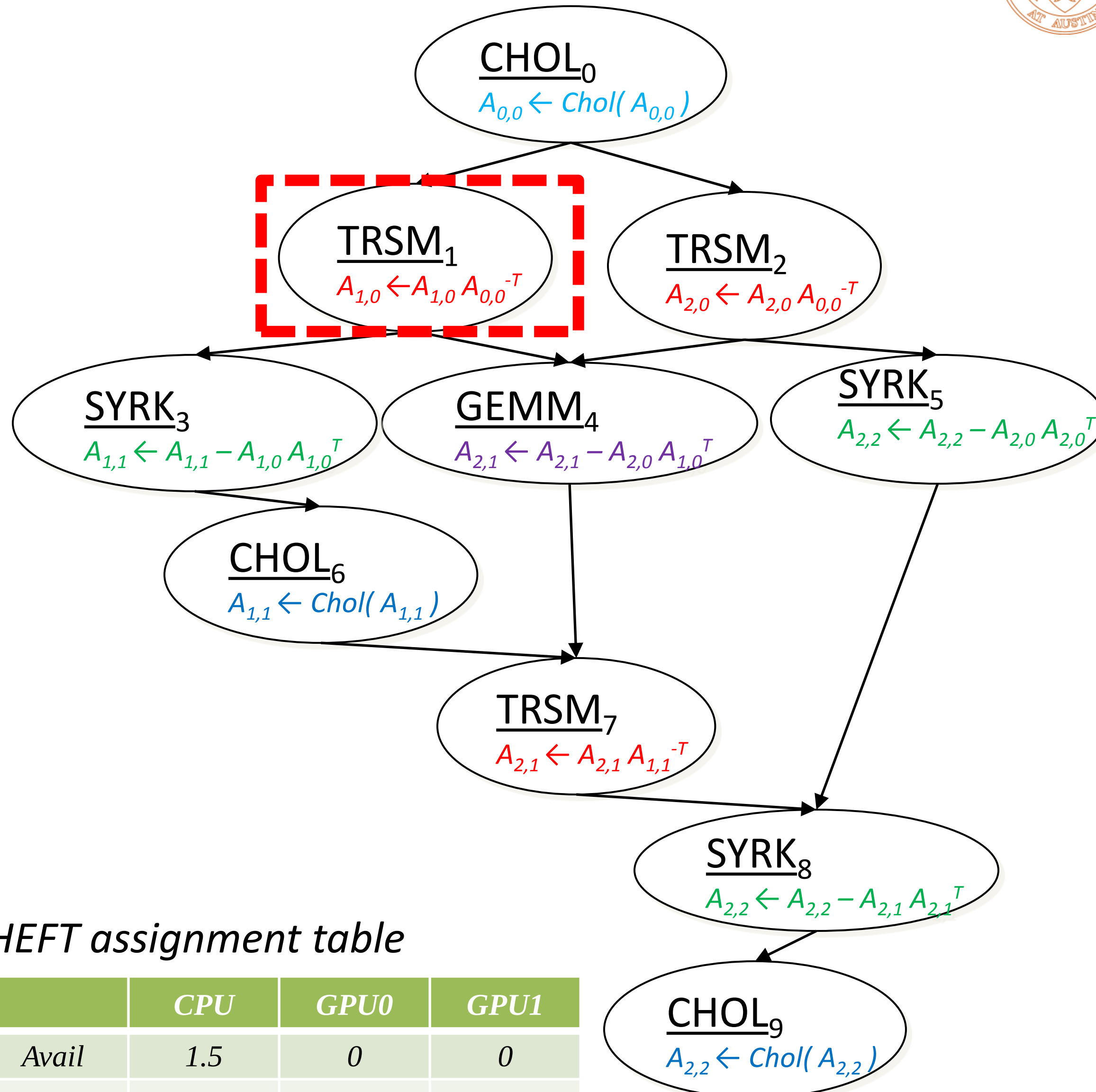
	CPU	GPU0	GPU1
Avail	0	0	0
EST	0	1	1
EFT	1.5	2	2
Priority	1	2	3

Data Distribution

	CPU	GPU0	GPU1
A00	1	1	0
A10	0	1	0
A11	1	0	0
A20	1	0	0
A21	1	0	0
A22	1	0	0

Scheduler

	CPU	GPU0	GPU1
CHOL0	X		
TRSM1		X	
TRSM2			
SYRK3			
GEMM4			
SYRK5			
CHOL6			
TRSM7			
SYRK8			
CHOL9			



HEFT assignment table

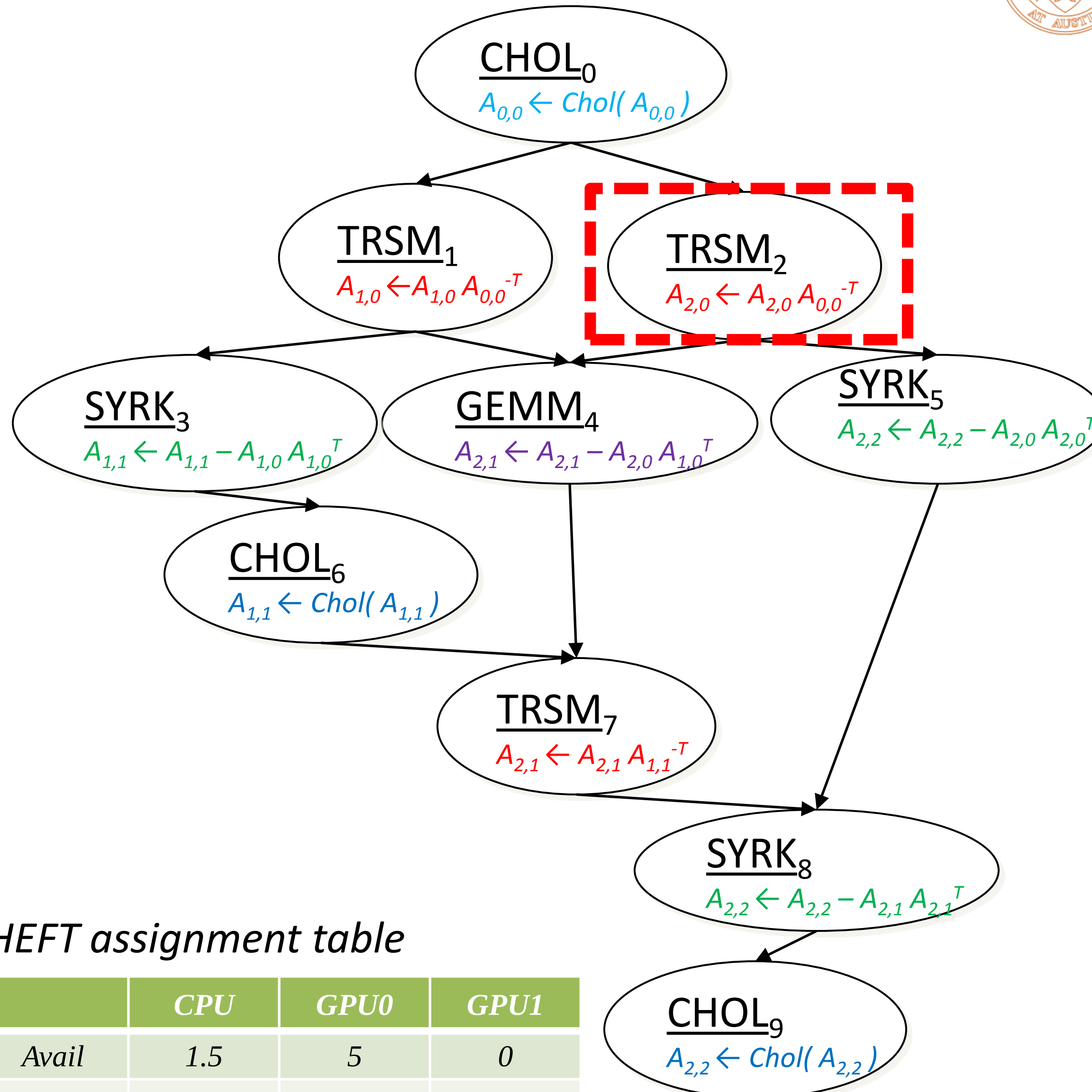
	CPU	GPU0	GPU1
Avail	1.5	0	0
EST	1.5	3.5	3.5
EFT	5.5	5	5
Priority	3	1	2

Data Distribution

	CPU	GPU0	GPU1
A00	1	1	1
A10	0	1	0
A11	1	0	0
A20	0	0	1
A21	1	0	0
A22	1	0	0

Scheduler

	CPU	GPU0	GPU1
CHOL0	X		
TRSM1		X	
TRSM2			X
SYRK3			
GEMM4			
SYRK5			
CHOL6			
TRSM7			
SYRK8			
CHOL9			



HEFT assignment table

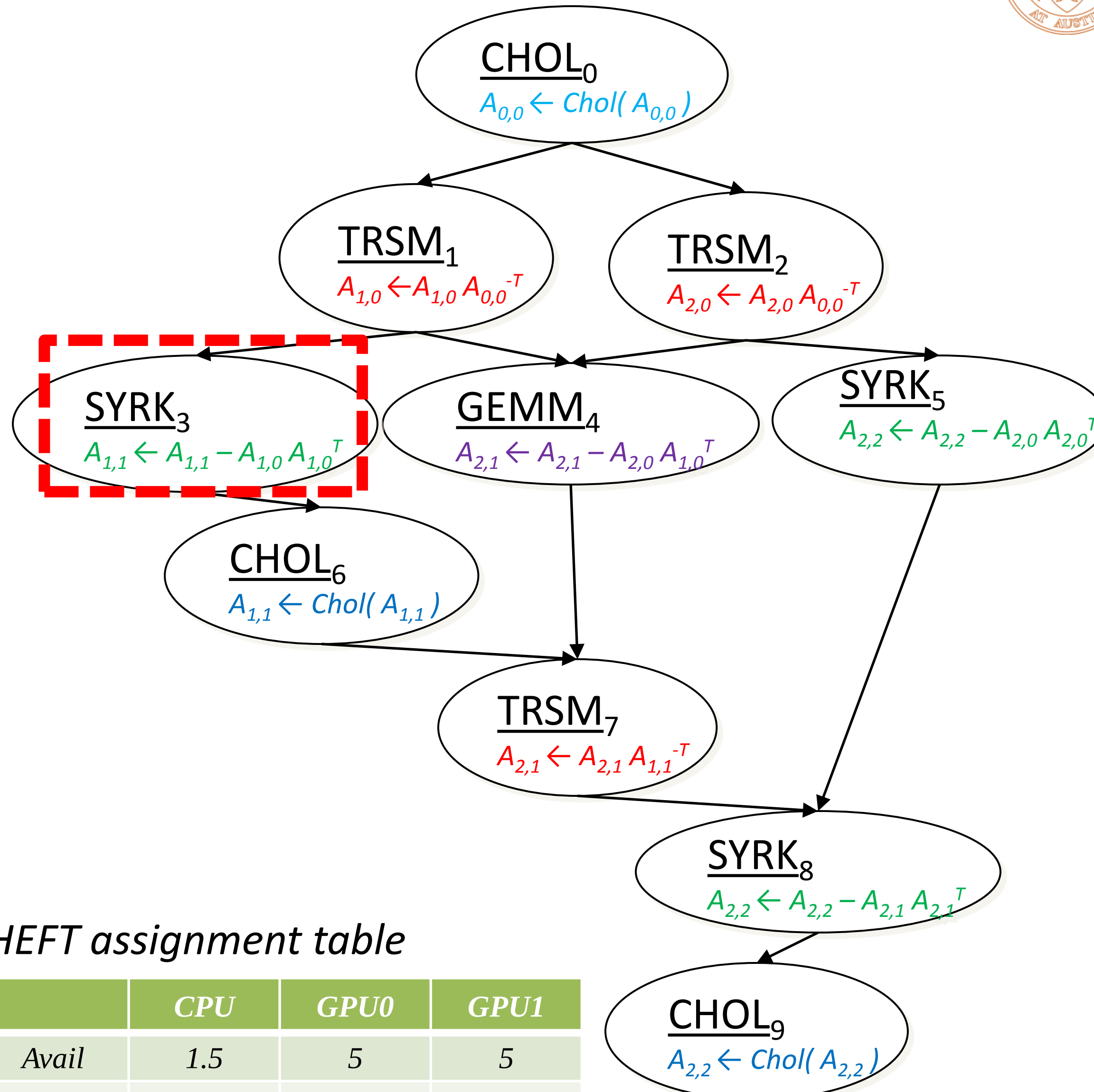
	CPU	GPU0	GPU1
Avail	1.5	5	0
EST	1.5	5	3.5
EFT	5.5	6.5	5
Priority	2	3	1

Data Distribution

	CPU	GPU0	GPU1
A00	1	1	1
A10	0	1	0
A11	0	1	0
A20	0	0	1
A21	1	0	0
A22	1	0	0

Scheduler

	CPU	GPU0	GPU1
CHOL0	X		
TRSM1		X	
TRSM2			X
SYRK3		X	
GEMM4			
SYRK5			
CHOL6			
TRSM7			
SYRK8			
CHOL9			



HEFT assignment table

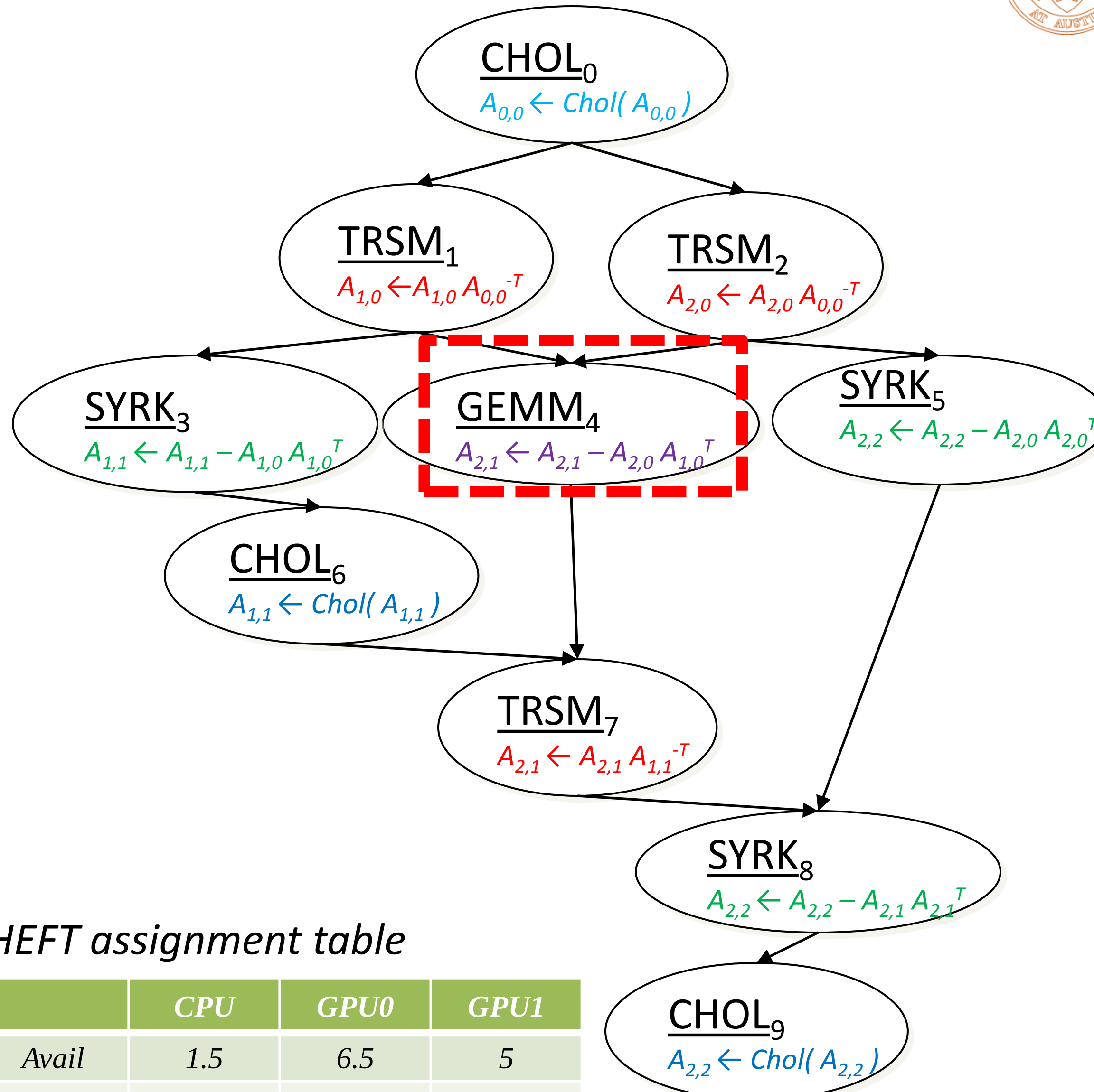
	CPU	GPU0	GPU1
Avail	1.5	5	5
EST	6	5	7
EFT	10	6.5	8.5
Priority	3	1	2

Data Distribution

	CPU	GPU0	GPU1
A00	1	1	1
A10	0	1	0
A11	0	1	0
A20	1	1	1
A21	0	1	0
A22	1	0	0

Scheduler

	CPU	GPU0	GPU1
CHOL0	X		
TRSM1		X	
TRSM2			X
SYRK3		X	
GEMM4		X	
SYRK5			
CHOL6			
TRSM7			
SYRK8			
CHOL9			



HEFT assignment table

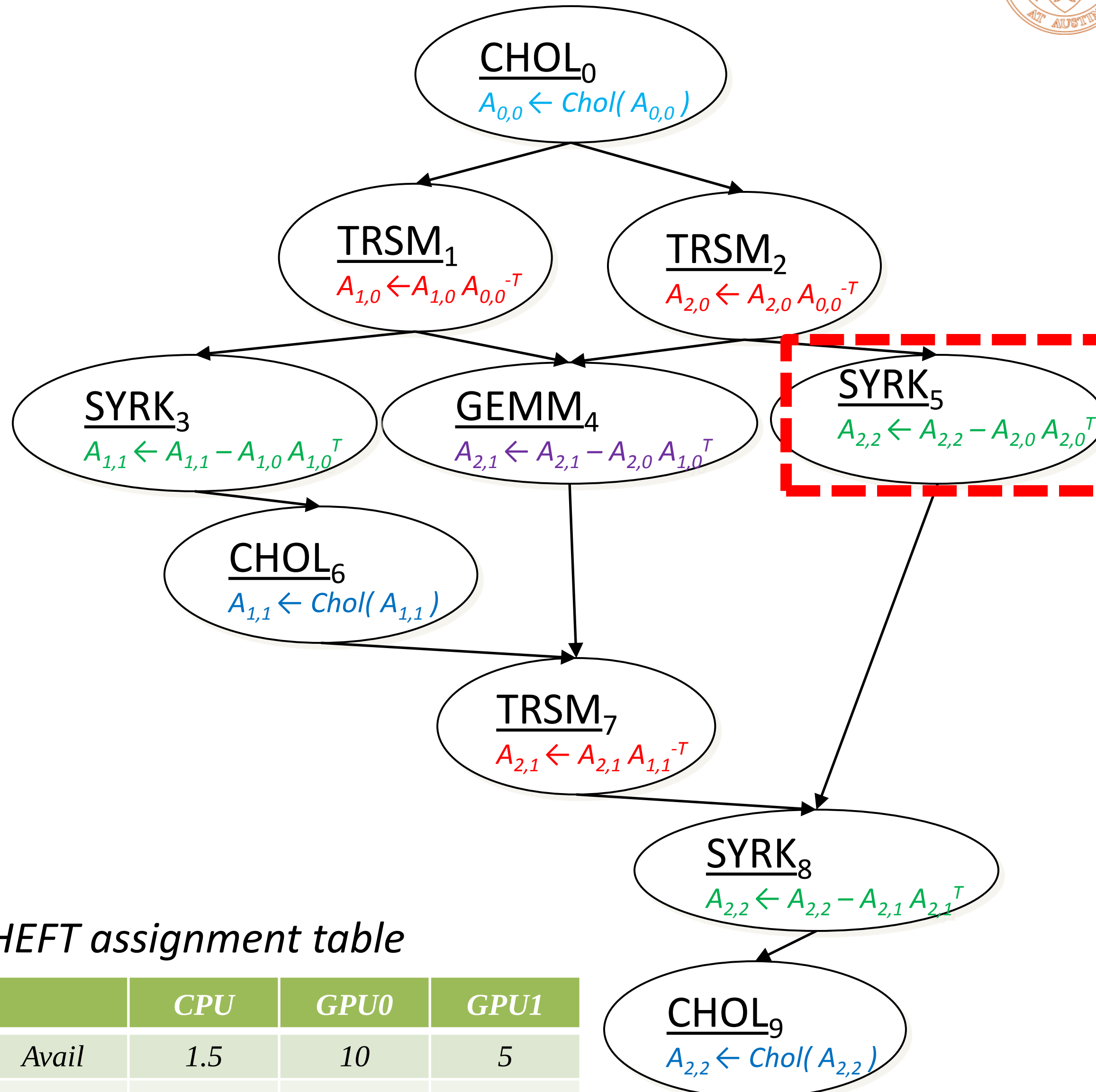
	CPU	GPU0	GPU1
Avail	1.5	6.5	5
EST	6	7	7
EFT	14	10	10
Priority	3	1	2

Data Distribution

	CPU	GPU0	GPU1
A00	1	1	1
A10	0	1	0
A11	0	1	0
A20	1	1	1
A21	0	1	0
A22	0	0	1

Scheduler

	CPU	GPU0	GPU1
CHOL0	X		
TRSM1		X	
TRSM2			X
SYRK3		X	
GEMM4		X	
SYRK5			X
CHOL6			
TRSM7			
SYRK8			
CHOL9			



HEFT assignment table

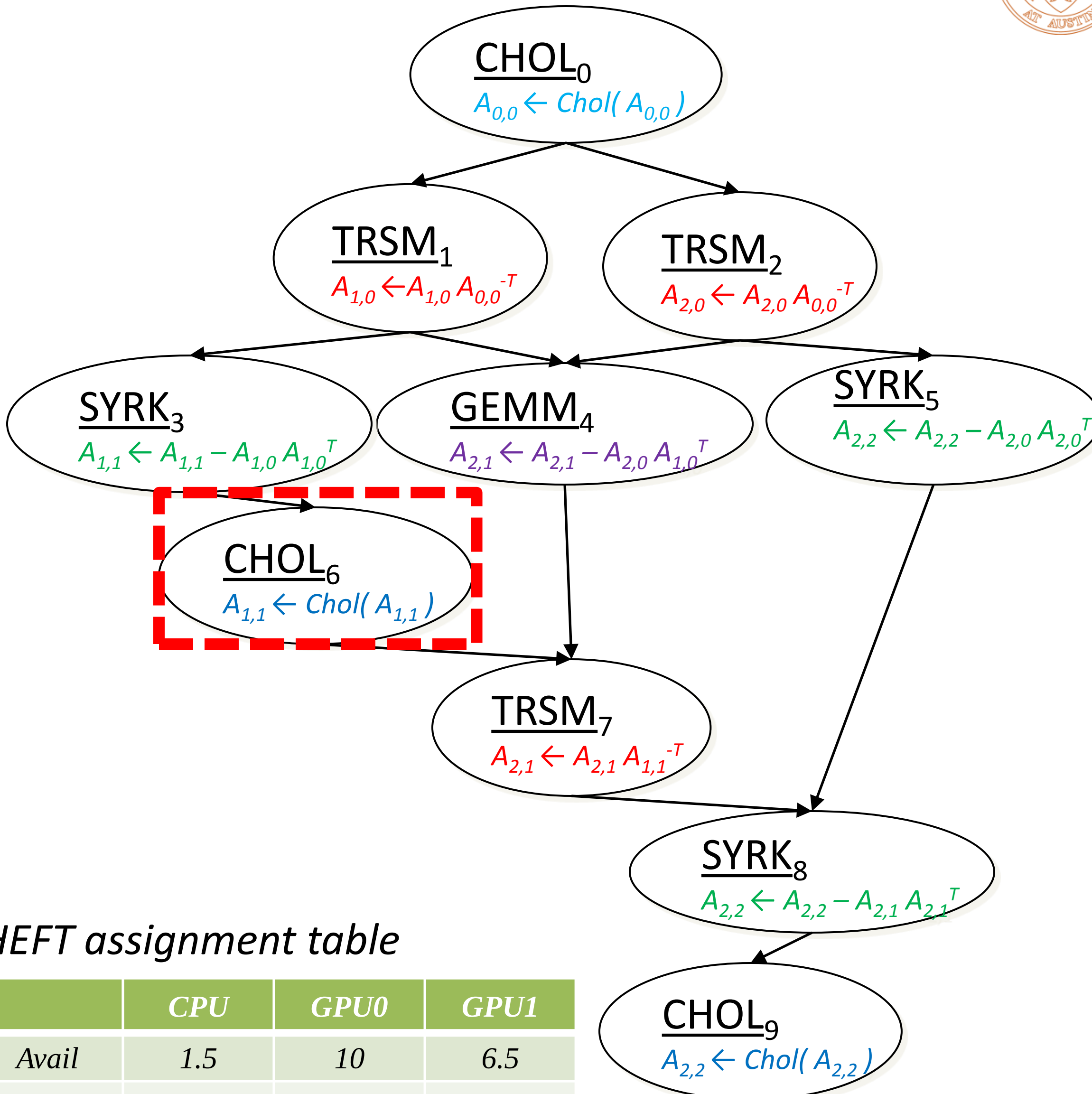
	CPU	GPU0	GPU1
Avail	1.5	10	5
EST	6	10	5
EFT	10	11.5	6.5
Priority	2	3	1

Data Distribution

	CPU	GPU0	GPU1
A00	1	1	1
A10	0	1	0
A11	1	0	0
A20	1	1	1
A21	0	1	0
A22	0	0	1

Scheduler

	CPU	GPU0	GPU1
CHOL0	X		
TRSM1		X	
TRSM2			X
SYRK3		X	
GEMM4		X	
SYRK5			X
CHOL6	X		
TRSM7			
SYRK8			
CHOL9			



HEFT assignment table

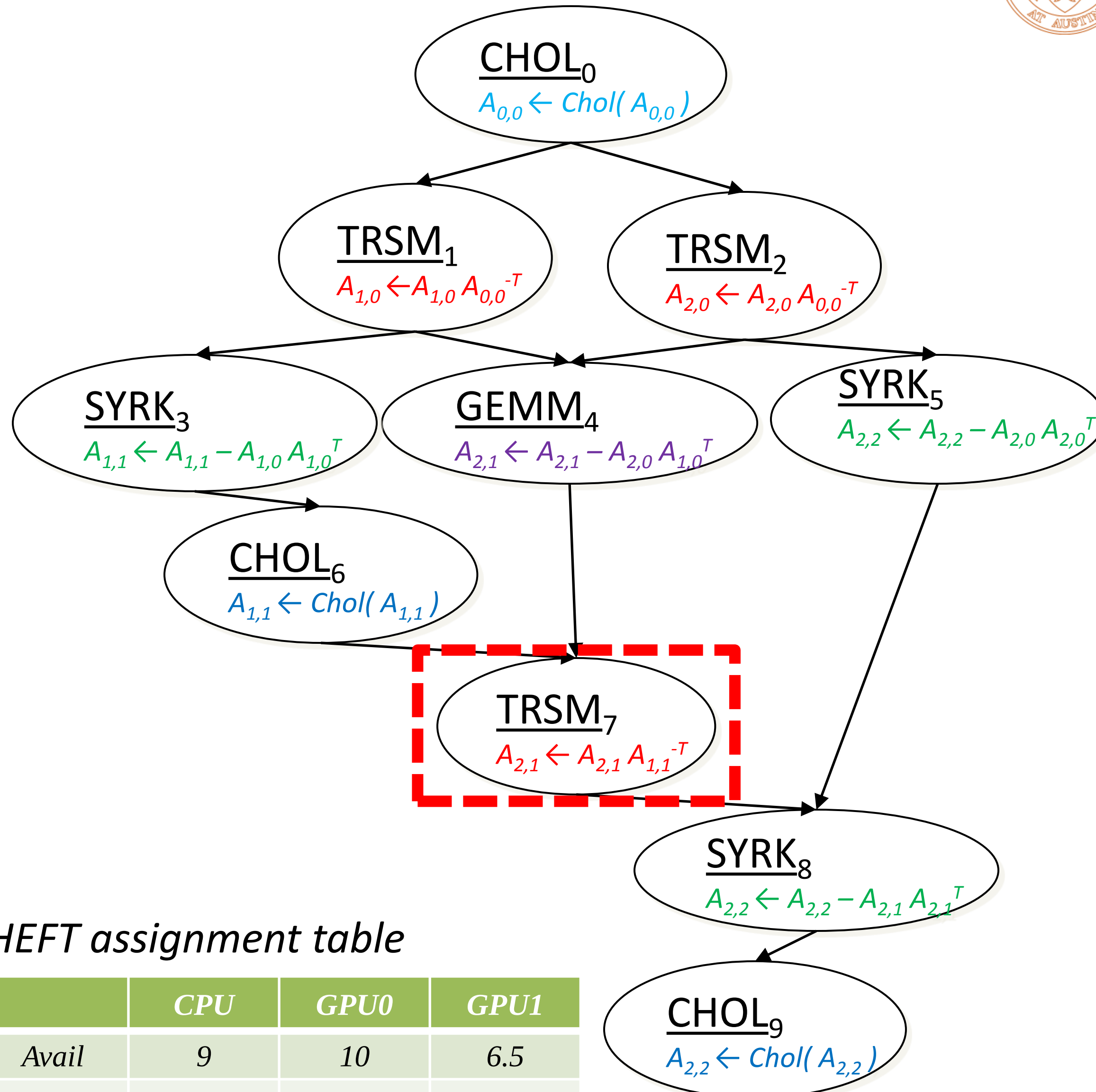
	CPU	GPU0	GPU1
Avail	1.5	10	6.5
EST	7.5	10	8.5
EFT	9	11	9.5
Priority	1	3	2

Data Distribution

	CPU	GPU0	GPU1
A00	1	1	1
A10	0	1	0
A11	1	1	0
A20	1	1	1
A21	0	1	0
A22	0	0	1

Scheduler

	CPU	GPU0	GPU1
CHOL0	X		
TRSM1		X	
TRSM2			X
SYRK3		X	
GEMM4		X	
SYRK5			X
CHOL6	X		
TRSM7		X	
SYRK8			
CHOL9			



HEFT assignment table

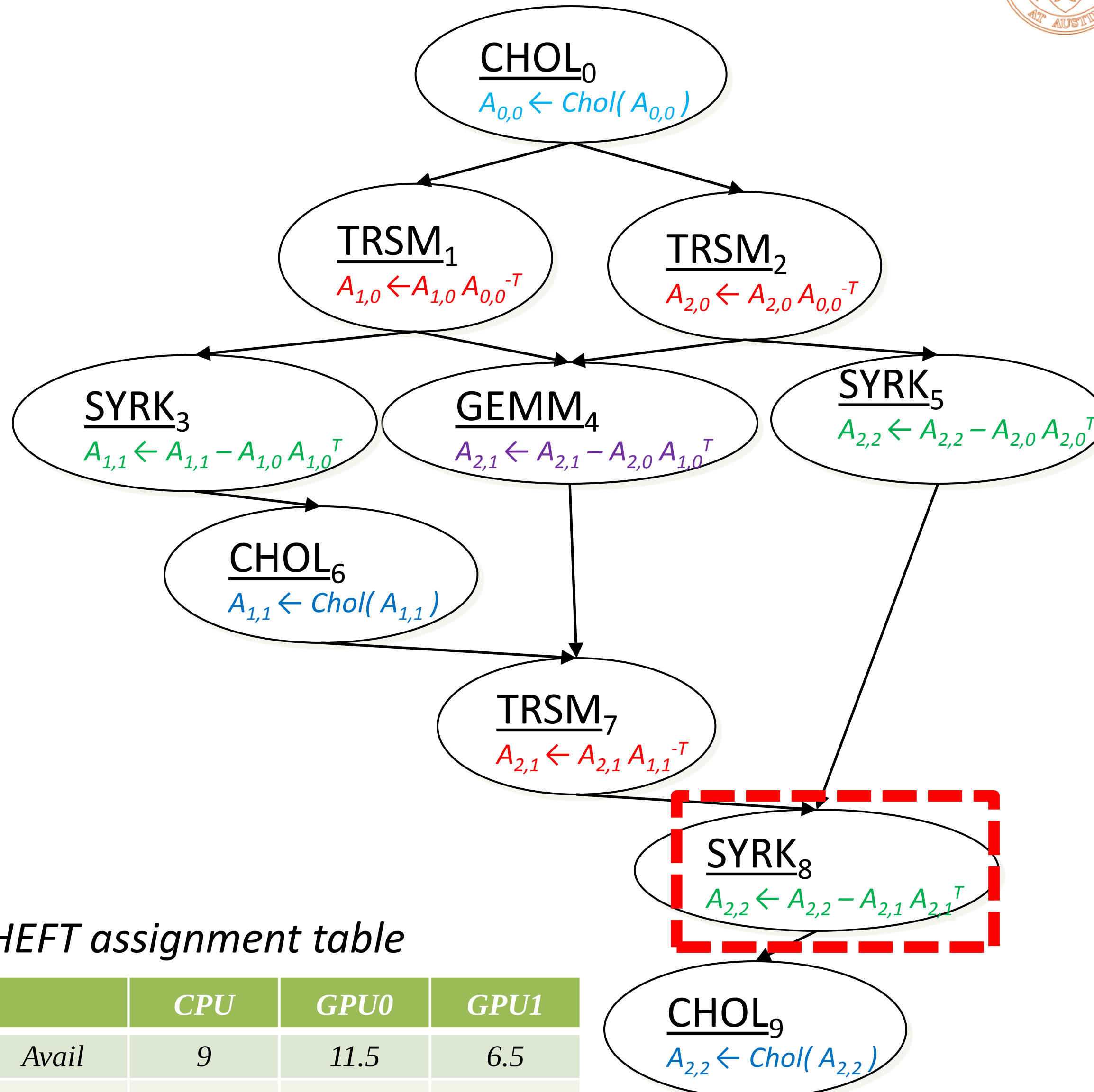
	CPU	GPU0	GPU1
Avail	9	10	6.5
EST	11	10	12
EFT	15	11.5	13.5
Priority	3	1	2

Data Distribution

	CPU	GPU0	GPU1
A00	1	1	1
A10	0	1	0
A11	1	1	0
A20	1	1	1
A21	0	1	0
A22	0	1	0

Scheduler

	CPU	GPU0	GPU1
CHOL0	X		
TRSM1		X	
TRSM2			X
SYRK3		X	
GEMM4		X	
SYRK5			X
CHOL6	X		
TRSM7		X	
SYRK8		X	
CHOL9			



HEFT assignment table

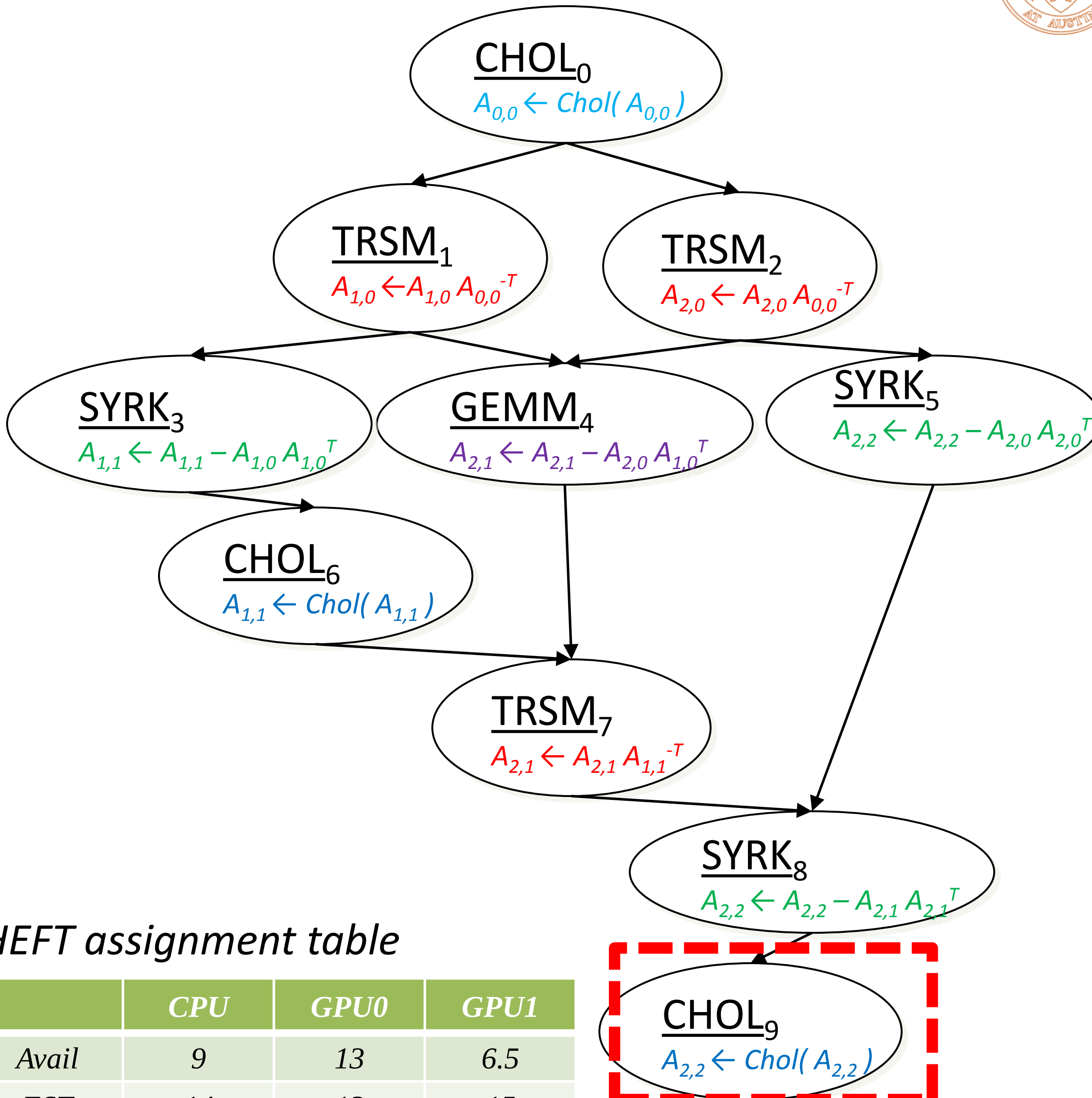
	CPU	GPU0	GPU1
Avail	9	11.5	6.5
EST	12.5	11.5	13.5
EFT	16.5	13	15
Priority	3	1	2

Data Distribution

	CPU	GPU0	GPU1
A00	1	1	1
A10	0	1	0
A11	1	1	0
A20	1	1	1
A21	0	1	0
A22	0	1	0

Scheduler

	CPU	GPU0	GPU1
CHOL0	X		
TRSM1		X	
TRSM2			X
SYRK3		X	
GEMM4		X	
SYRK5			X
CHOL6	X		
TRSM7		X	
SYRK8		X	
CHOL9		X	



HEFT assignment table

	CPU	GPU0	GPU1
Avail	9	13	6.5
EST	14	13	15
EFT	15.5	14	16
Priority	2	1	3

SuperMatrix Approach on Heterogeneous Platforms

S0: $D \leftarrow A*B$

S1: $A \rightarrow L * L^T$

S2: $B \leftarrow B * L^{-T}$

S3: $C \leftarrow C - B * B^T$

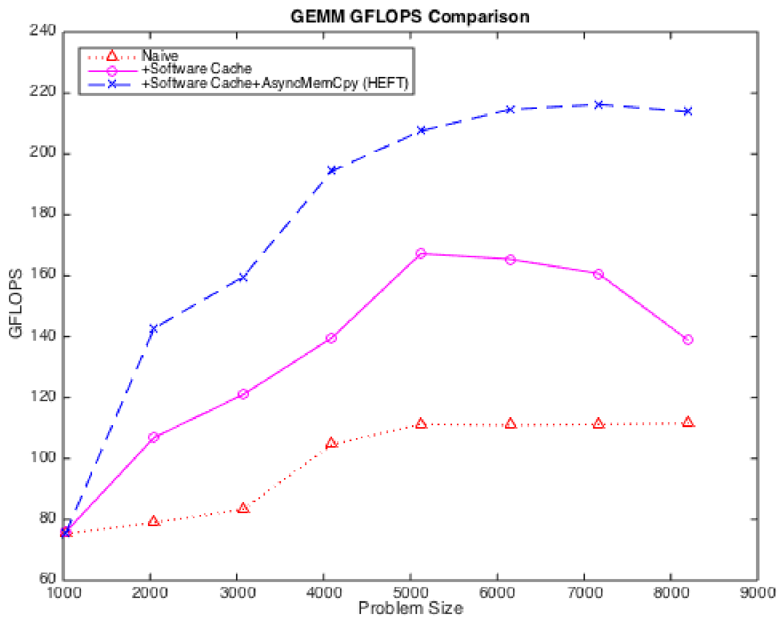
S4: $X \leftarrow L^{-1} * X$

```
/*-------------------------------------------*/
FLASH_Gemm( FLA_NO_TRANSPOSE,FLA_NO_TRANSPOSE,
            FLA_ONE, A, B, FLA_ZERO, D );
FLASH_Chol( FLA_LOWER_TRIANGULAR, A );
FLASH_Trsm( FLA_RIGHT, FLA_LOWER_TRIANGULAR,
            FLA_TRANSPOSE, FLA_NONUNIT_DIAG,
            FLA_ONE, A, B );
FLASH_Syrk( FLA_LOWER_TRIANGULAR, FLA_NO_TRANSPOSE,
            FLA_MINUS_ONE, B, FLA_ONE, C );
FLASH_Trsm( FLA_LEFT, FLA_LOWER_TRIANGULAR,
            FLA_NO_TRANSPOSE, FLA_NONUNIT_DIAG,
            FLA_ONE, L, X );
/*-------------------------------------------*/
```

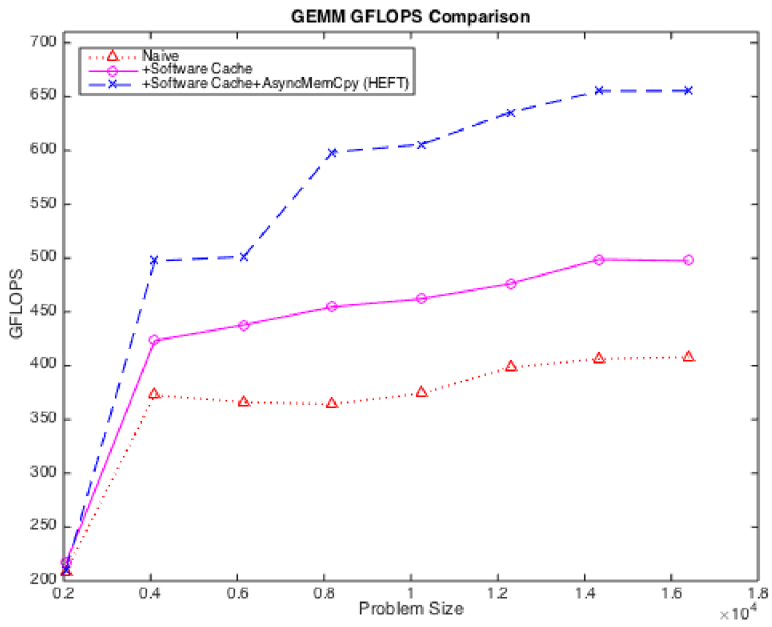
No Code Change!

Performance

6-core single-socket Xeon E5649 CPU
+ 1 GTX 480 GPU card
BLOCK SIZE: 1024

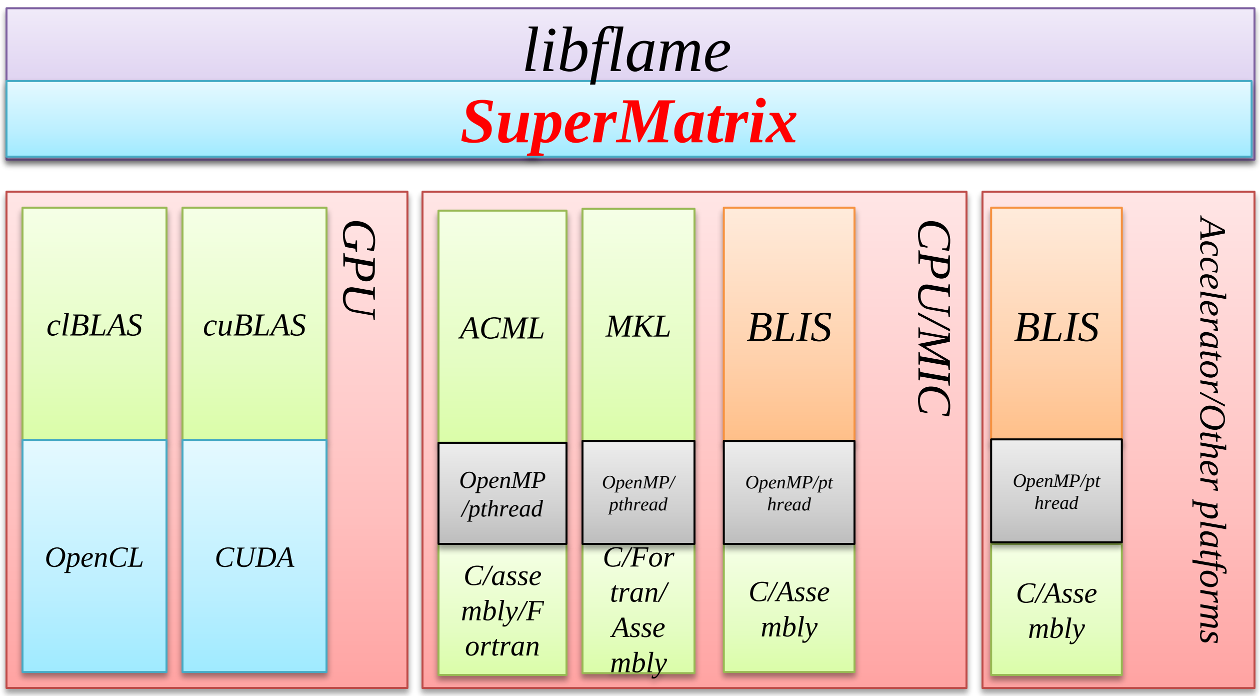


6-core single-socket Xeon E5649 CPU
+ 2 Tesla C2070 GPU card
BLOCK SIZE: 2048



Conclusion

one ring to rule them all



SuperMatrix Approach on Heterogeneous Platforms

S0: $D \leftarrow A*B$

S1: $A \rightarrow L * L^T$

S2: $B \leftarrow B * L^{-T}$

S3: $C \leftarrow C - B * B^T$

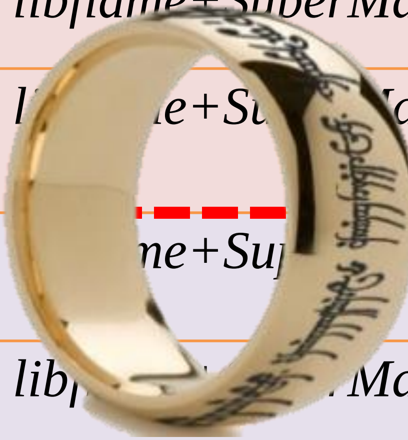
S4: $X \leftarrow L^{-1} * X$

```
/*-----------------------------------------------*/
FLASH_Gemm( FLA_NO_TRANSPOSE,FLA_NO_TRANSPOSE,
              FLA_ONE, A, B, FLA_ZERO, D );
FLASH_Chol( FLA_LOWER_TRIANGULAR, A );
FLASH_Trsm( FLA_RIGHT, FLA_LOWER_TRIANGULAR,
              FLA_TRANSPOSE, FLA_NONUNIT_DIAG,
              FLA_ONE, A, B );
FLASH_Syrk( FLA_LOWER_TRIANGULAR, FLA_NO_TRANSPOSE,
              FLA_MINUS_ONE, B, FLA_ONE, C );
FLASH_Trsm( FLA_LEFT, FLA_LOWER_TRIANGULAR,
              FLA_NO_TRANSPOSE, FLA_NONUNIT_DIAG,
              FLA_ONE, L, X );
/*-----------------------------------------------*/
```

No Code Change!

Related Work

Target Platform	*Lapack Project*	*FLAME Project*
Sequential	*LAPACK*	*libflame*
Sequential+multithreaded BLAS	*LAPACK*	*libflame*
Multicore/multithreaded	*PLASMA*	*libflame+SuperMatrix*
Multicore+out-of-order scheduling	*PLASMA+Quark*	*l[illegible]e+Su[illegible]atrix*
CPU + single GPU	*MAGMA*	*[illegible]me+Su[illegible]trix*
Multicore + multi-GPU	*DAGuE/StarPU/ XKaapi*	*lib[illegible]Matrix*



Related Work

Target Platform	Lapack Project	FLAME Project
Sequential	*LAPACK*	*libflame*
Sequential+multithreaded BLAS	*LAPACK*	*libflame*
Multicore/multithreaded	*PLASMA*	*libflame+SuperMatrix*
Multicore+out-of-order scheduling	*PLASMA+Quark*	*libflame+SuperMatrix*
CPU + single GPU	*MAGMA*	*libflame+SuperMatrix*
Multicore + multi-GPU	*DAGuE/StarPU/ XKaapi*	*libflame+SuperMatrix*

Questions?