Unleashing the Power of Multi-GPU Accelerators with FLAME

Enrique S. Quintana Ortí
Universidad Jaume I
quintana@icc.uji.es

HPC&A
Universitat Jaume I High Performance Computing and Architectures

SIAM Conference on Parallel Processing for Scientific Computing
Seattle, 2010

ACKNOWLEGMENTS

- Joint work:

UJI	UT@Austin
Francisco D. Igual	Ernie Chan
Mercedes Marqués	Robert van de Geijn
Gregorio Quintana	Field G. Van Zee

ACKNOWLEGMENTS

- Support:

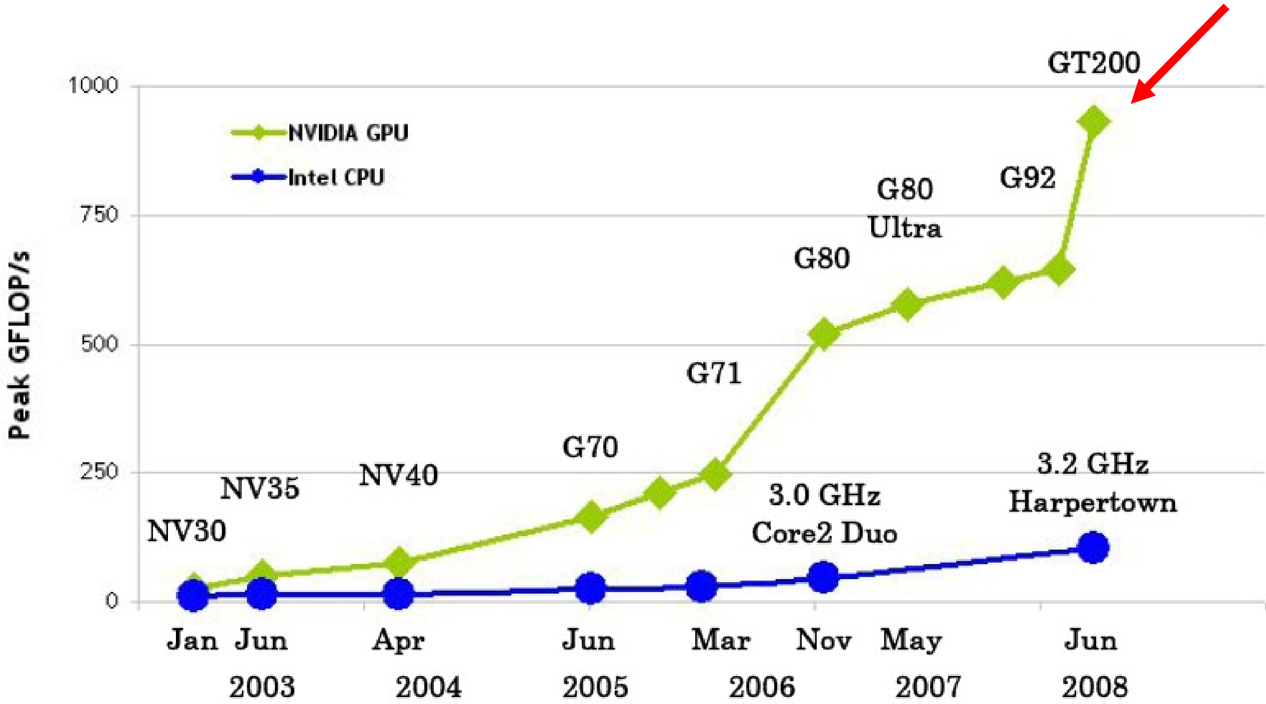
UJI

UT@Austin

2008 NVIDIA Professorship Award

MOTIVATION

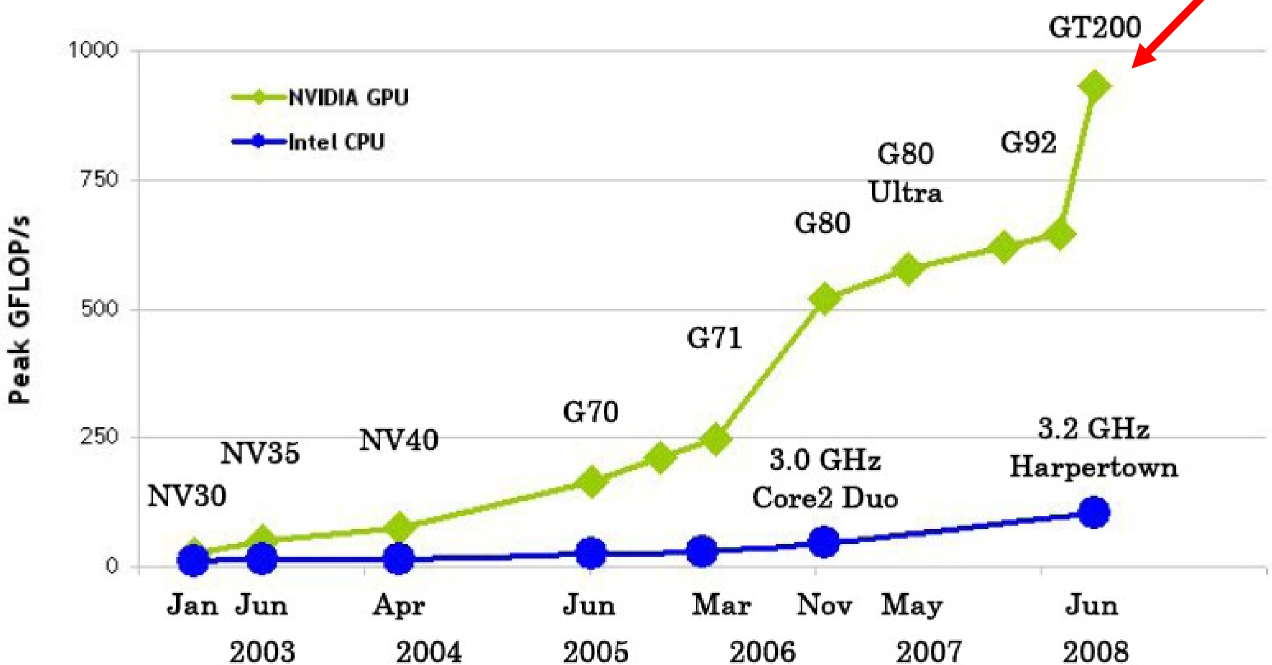
MYTH OR REALITY?



GT200 = GeForce GTX 280	G71 = GeForce 7900 GTX	NV35 = GeForce FX 5950 Ultra
G92 = GeForce 9800 GTX	G70 = GeForce 7800 GTX	NV30 = GeForce FX 5800
G80 = GeForce 8800 GTX	NV40 = GeForce 6800 Ultra	

MOTIVATION

PERSON PER MONTH?



GT200 = GeForce GTX 280	G71 = GeForce 7900 GTX	NV35 = GeForce FX 5950 Ultra
G92 = GeForce 9800 GTX	G70 = GeForce 7800 GTX	NV30 = GeForce FX 5800
G80 = GeForce 8800 GTX	NV40 = GeForce 6800 Ultra	

MOTIVATION

Message:

“High-level programming (through abstraction) eases the programmability problem posed by new architectures without sacrifyzing high performance”

- FLAME
- PLAPACK

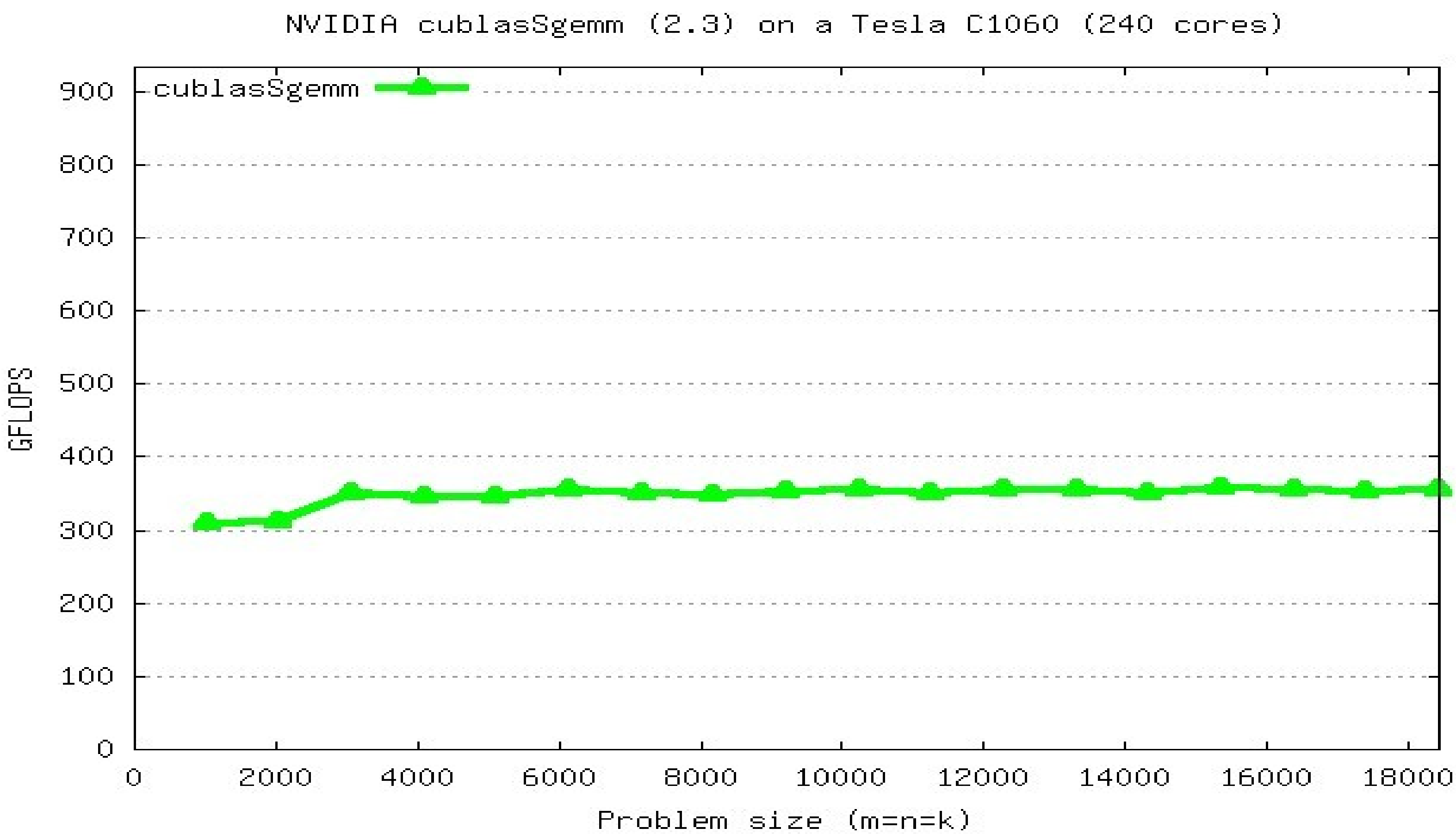
OUTLINE

- Evaluation and tuning of CUBLAS
- Superscalar techniques in the construction of dense linear algebra libraries for multi-GPU platforms:
  1. Data-flow dynamic scheduling
  2. DSM
- Clusters of GPUs

OUTLINE

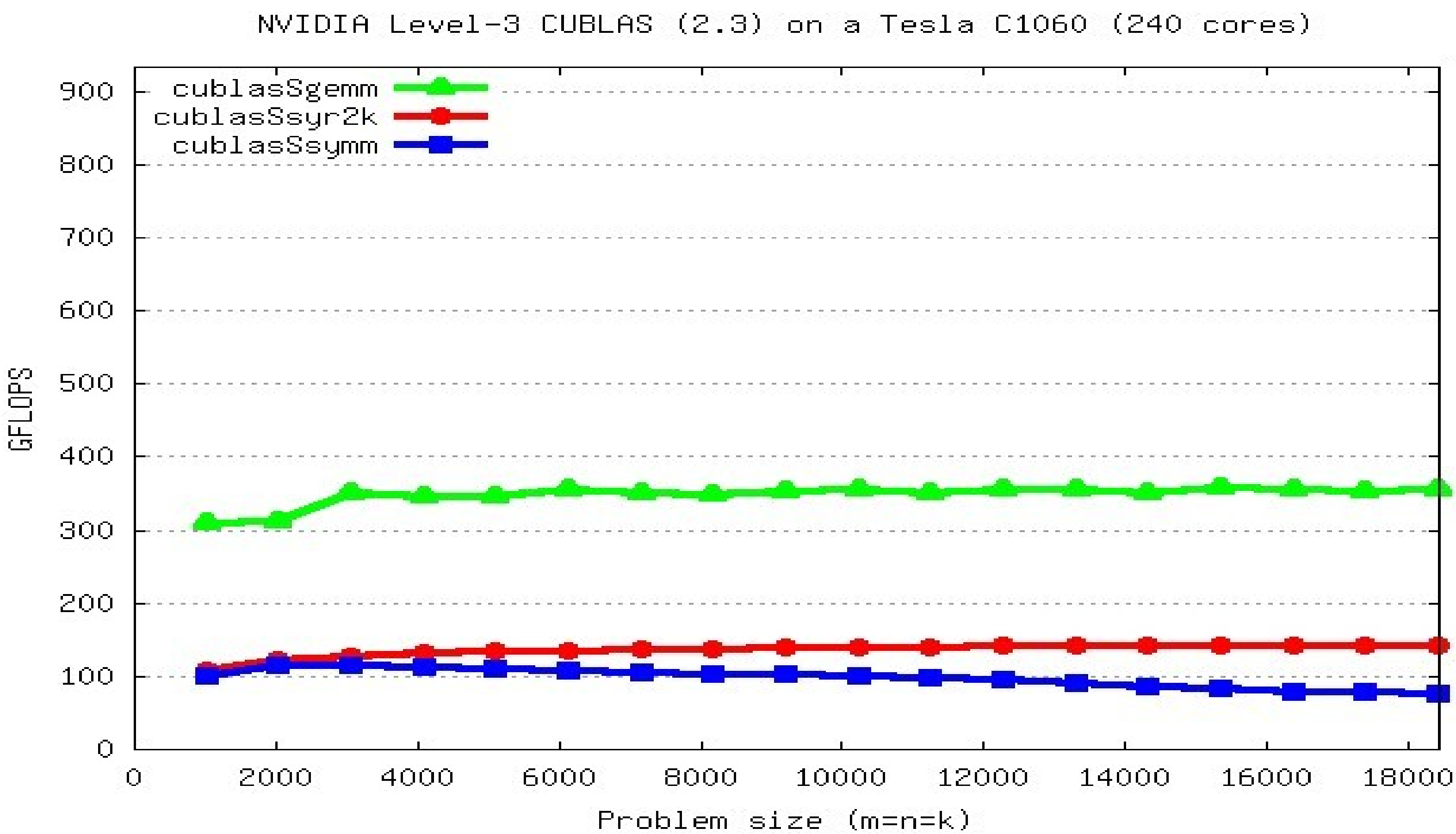
- Evaluation and tuning of CUBLAS
- Superscalar techniques in the construction of dense linear algebra libraries for multi-GPUs platforms:
  1. Data-flow dynamic scheduling
  2. DSM
- Clusters of GPUs

CUBLAS



38% peak

CUBLAS



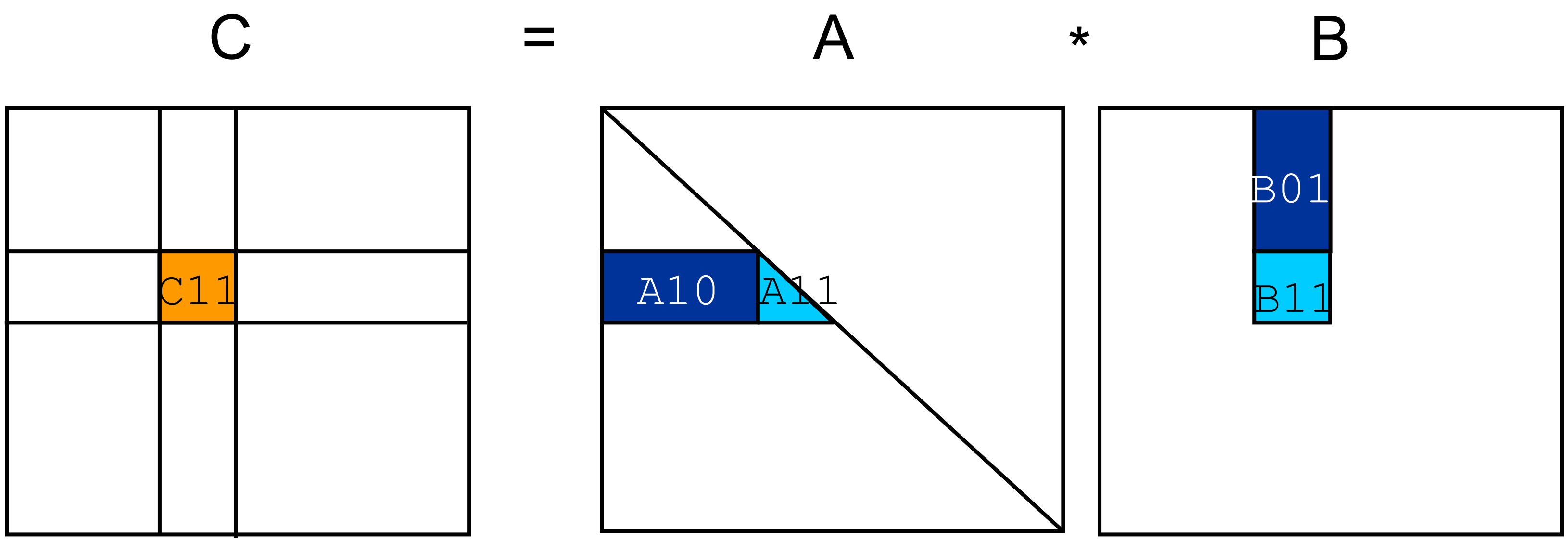
38% peak

25% peak

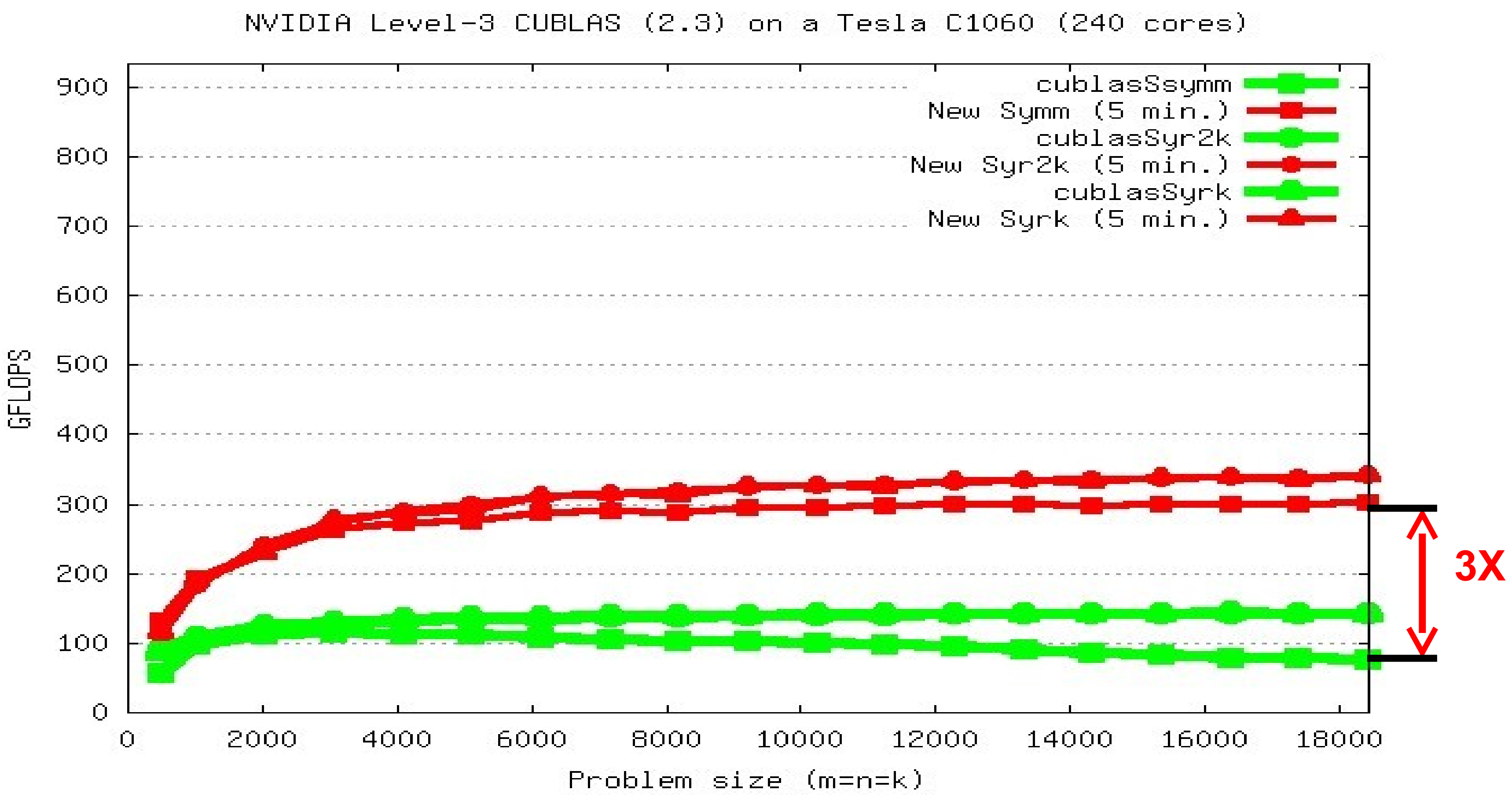
22% peak

TUNED CUBLAS USING FLAME

Cast Symm in terms of Gemm and minor Symm



TUNED CUBLAS USING FLAME



TUNED CUBLAS USING FLAME

```
while ( FLA_Obj_length( ATL ) < FLA_Obj_length( A ) ) {
  /* 2x2->3x3 repartitionings of A, B and C */
  FLA_Gemm( ..., A10, B01, ..., C11 );
  FLA_Symm( ..., A11, B11, ..., C11 );
  /* 3x3->2x2 repartitionings of A, B and C */
}
```

- Five-minute coding effort
- No CUDA-level programming needed
- Orthogonal to improvements on cublasSgemm

OUTLINE

- Evaluation and tuning of CUBLAS
- Superscalar techniques in the construction of dense linear algebra libraries for multi-GPU platforms:
  1. Data-flow dynamic scheduling
  2. DSM
- Clusters of GPUs

MULTI-GPU PLATFORMS

- CPU-hardware accelerators
  - More favourable price-performance ratio
  - Slow communication between host and devices
  - Separate memory spaces: no hardware coherence

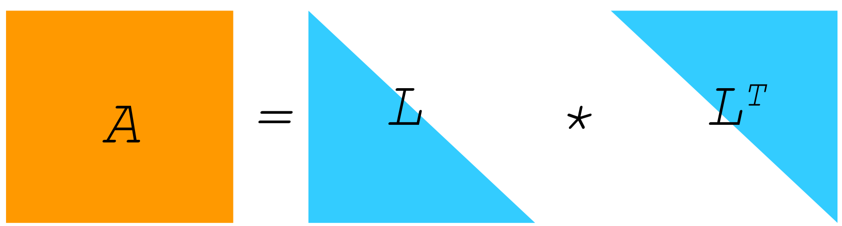
DATA-FLOW DYNAMIC SCHEDULING

- Out-of-order execution controlled by data dependencies (*data-flow parallelism at task level*)

- Goals:
  - Increase the degree of parallelism during the execution of dense linear algebra operations
  - Balance the workload distribution
  - Hide dynamic scheduling in an architecture-dependent run-time: ease programmability

DATA-FLOW DYNAMIC SCHEDULING
Current libraries

- Cholesky factorization



Key to the solution of (s.p.d.) linear systems

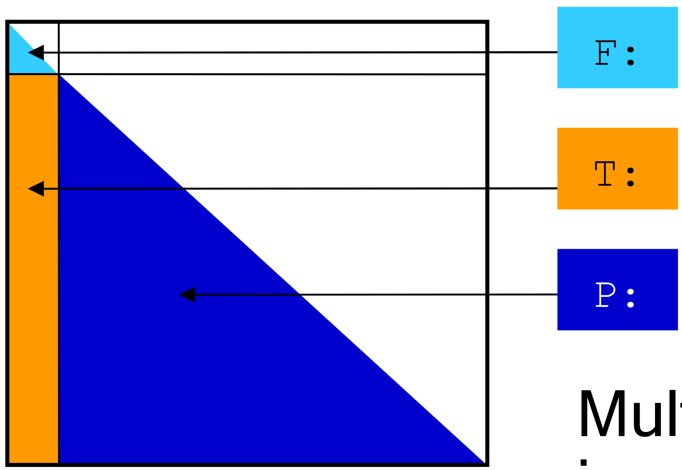
$$A\ x = b \equiv \quad (LL^T)\,x = b$$

$$L\ y = b \quad \Rightarrow \quad y$$

$$L^T x = y \quad \Rightarrow \quad x$$

DATA-FLOW DYNAMIC SCHEDULING
Current libraries

- Blocked algorithm cast in terms of Gemm



1st iteration

$$A_{11} = L_{11} * L_{11}^T$$

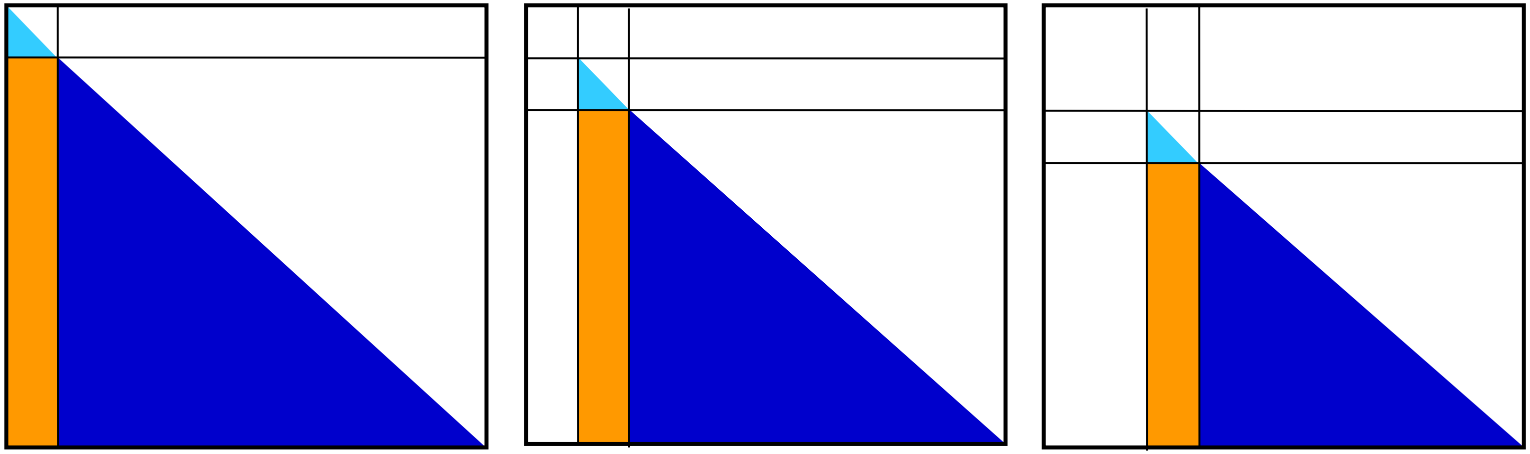
$$L_{21} \leftarrow A_{21} * L_{11}^{-T}$$

$$A_{22} \leftarrow A_{22} - L_{21} * L_{21}^T$$

Multi-core processor: multithreaded implementation of T and P

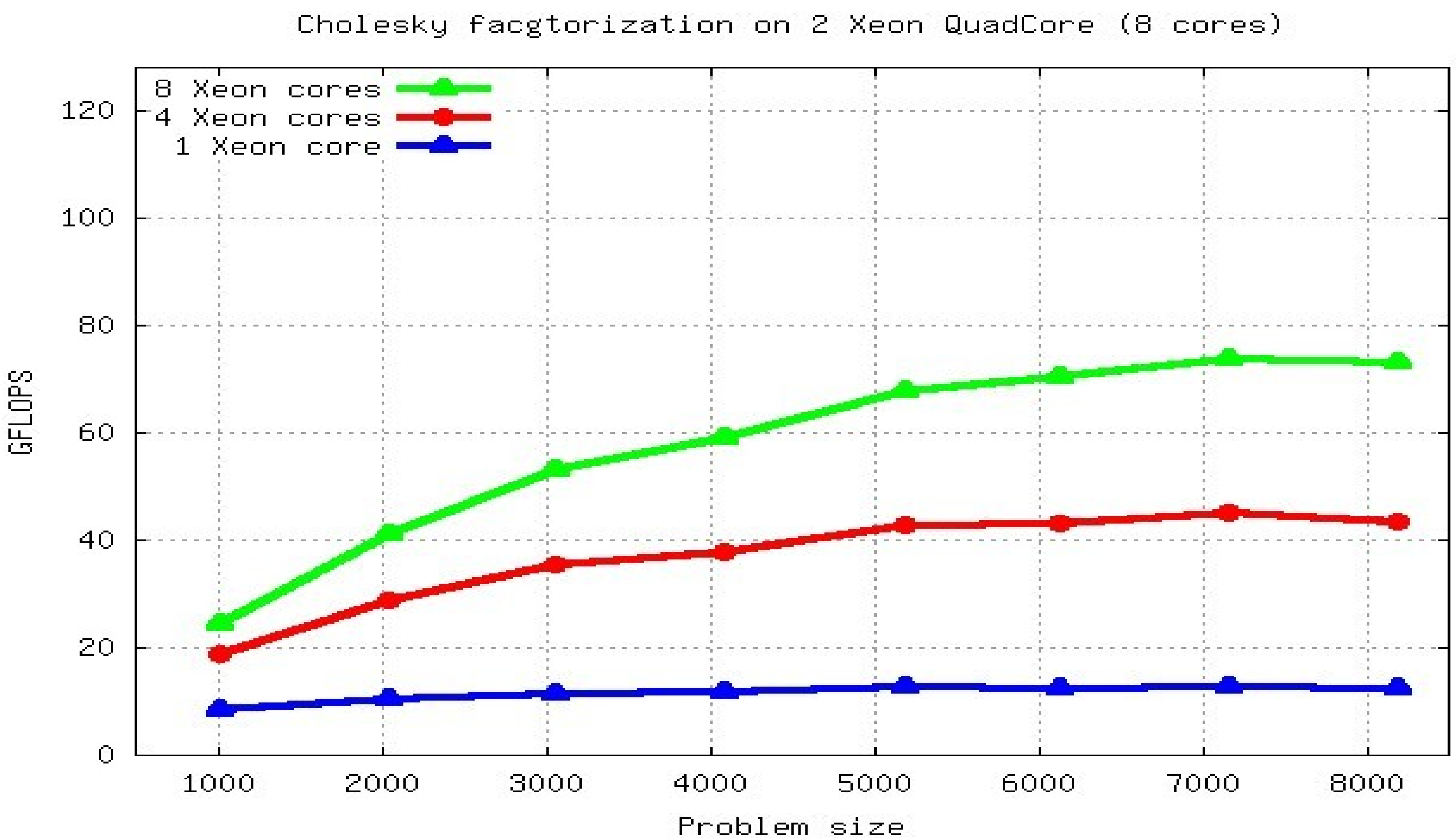
DATA-FLOW DYNAMIC SCHEDULING
Current libraries

- Blocked algorithm cast in terms of Gemm



1st iteration 2nd iteration 3rd iteration ...

DATA-FLOW DYNAMIC SCHEDULING
Current libraries



57% peak

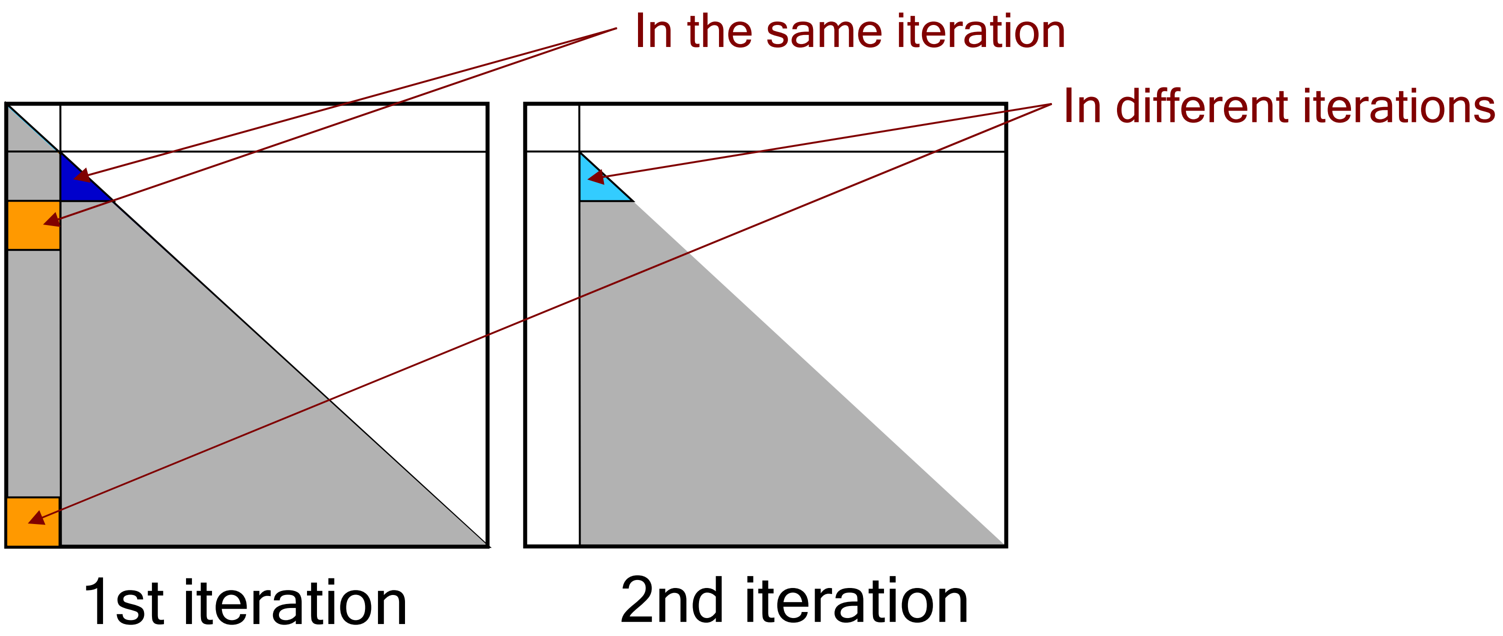
71% peak

80% peak

DATA-FLOW DYNAMIC SCHEDULING
Current libraries

- Why?

There is more parallelism than is being exploited

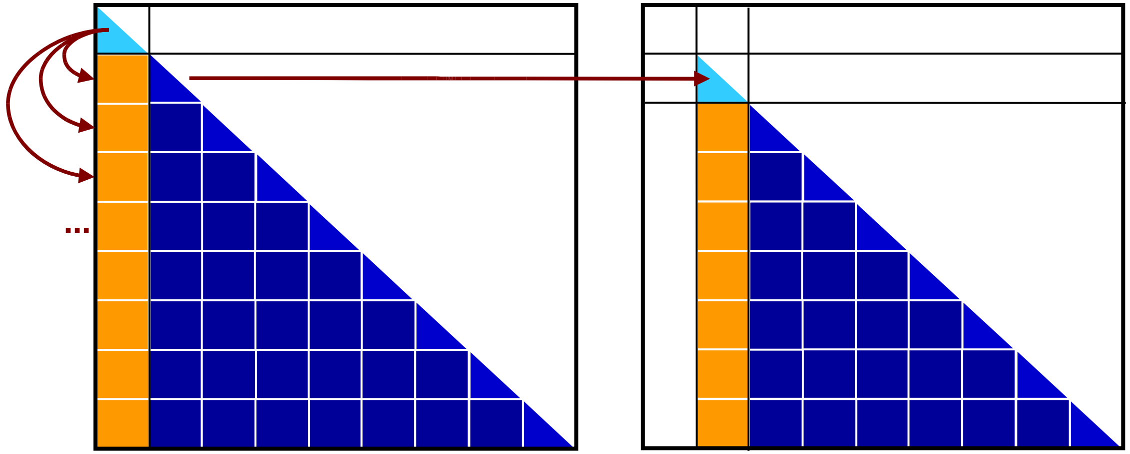


DATA-FLOW DYNAMIC SCHEDULING

- Read/written blocks determine dependencies

```
while ( ... ) {
      /* 2x2->3x3 repartitioning */
      FLA_Chol( A11 );
      FLA_Trsm( ..., A11, A21 );
      FLA_Syrk( ..., A21, ..., A22 );
      /* 3x3->2x2 repartitioning */
}
```

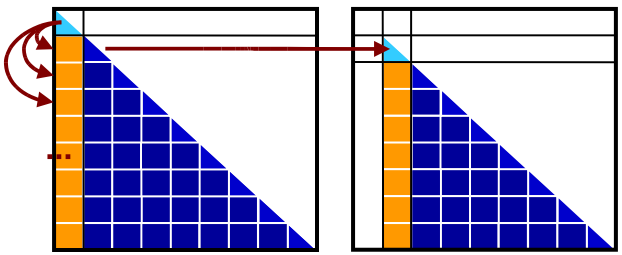
Task tree as a DAG:



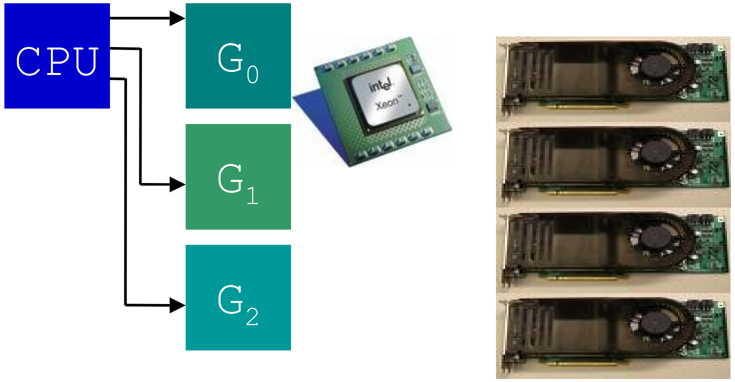
DATA-FLOW DYNAMIC SCHEDULING

- Execution of task tree:
  - Scheduling (temporal) dictated by data dependencies
  - Cache-aware mapping (spatial)

Task tree



Multi-GPU platform



Outline

- Evaluation and tuning of CUBLAS
- Superscalar techniques in the construction of dense linear algebra libraries for multi-core processors and GPUs:
  1. Data-flow dynamic scheduling
  2. DSM

- Clusters of GPUs

DISTRIBUTED-SHARED MEMORY

- Middleware that deals with separate address spaces in host and each device (GPU)

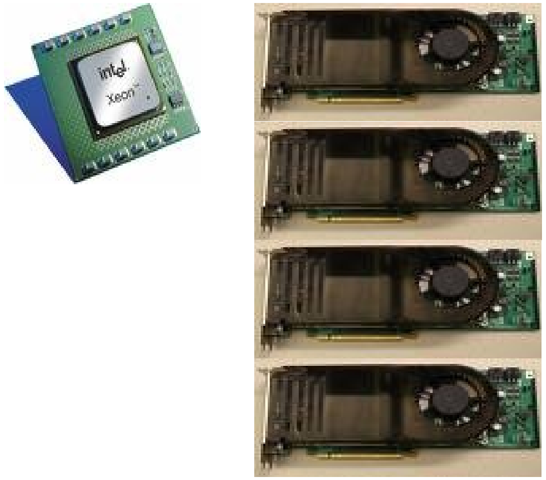
- Goals:
  - Reduce the number of data transfers: increase efficiency
  - Hide the existence of multiple memory spaces: ease programmability

DISTRIBUTED-SHARED MEMORY

- Data transfer
  - Before execution, transfer data to device
  - Upon completion, retrieve results back to host →

→ poor data locality

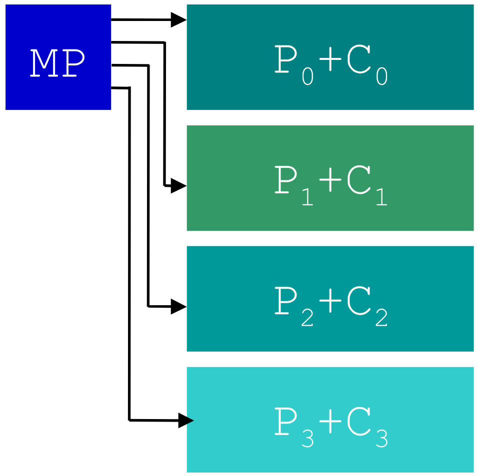
Multi-GPU platform



DISTRIBUTED-SHARED MEMORY

- Shared memory system

Multi-core processor with hw. coherence:



≡

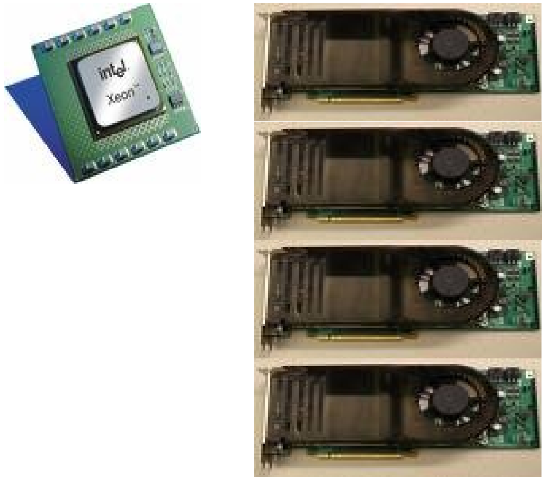
Multi-GPU platform

DISTRIBUTED-SHARED MEMORY

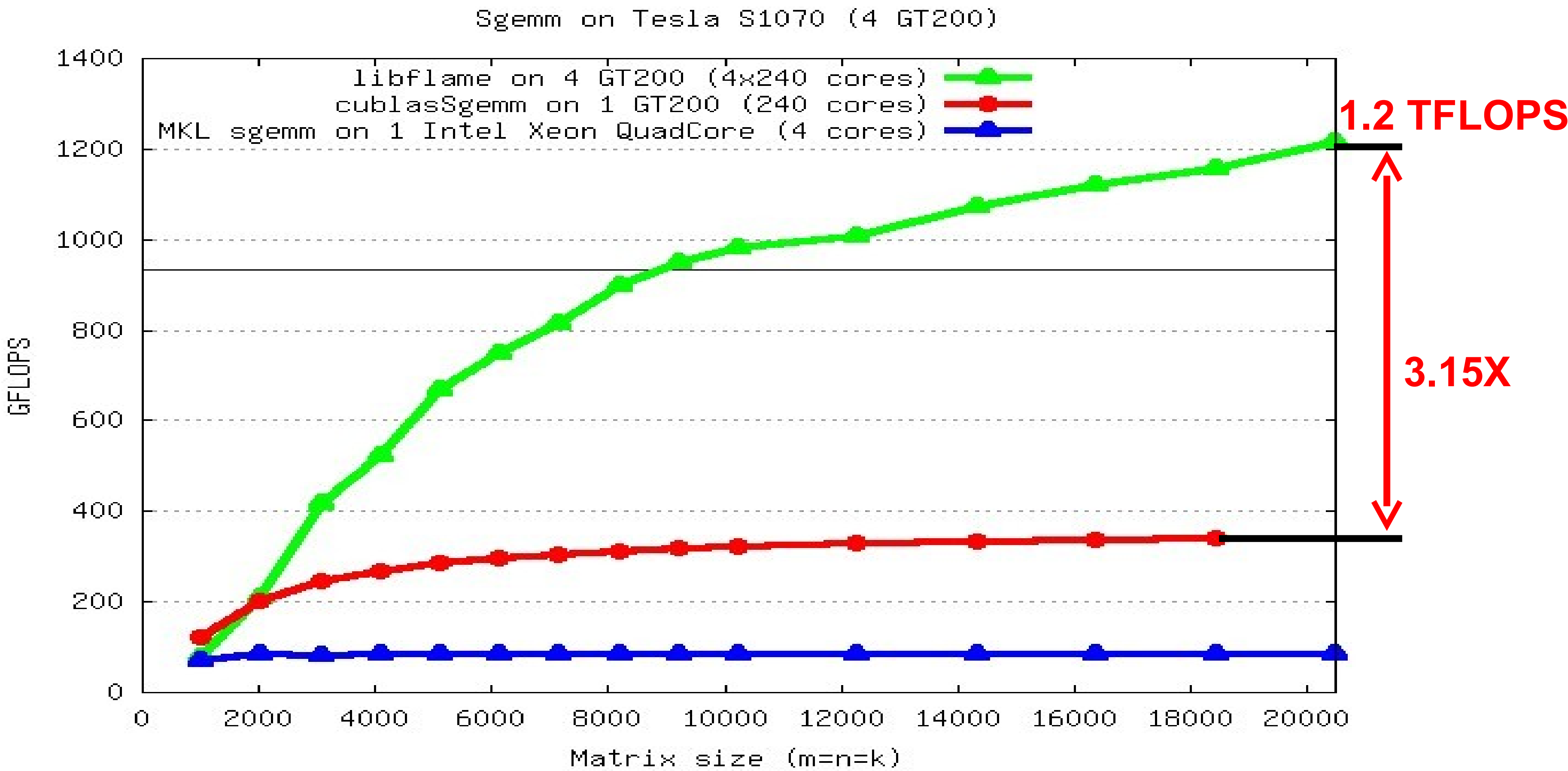
- Reduce #data transfers
  - Software cache in devices:
    - Operate at block level
    - Software $\rightarrow$ flexibility
    - *Write-back*
    - *Write-invalidate*

$\rightarrow$

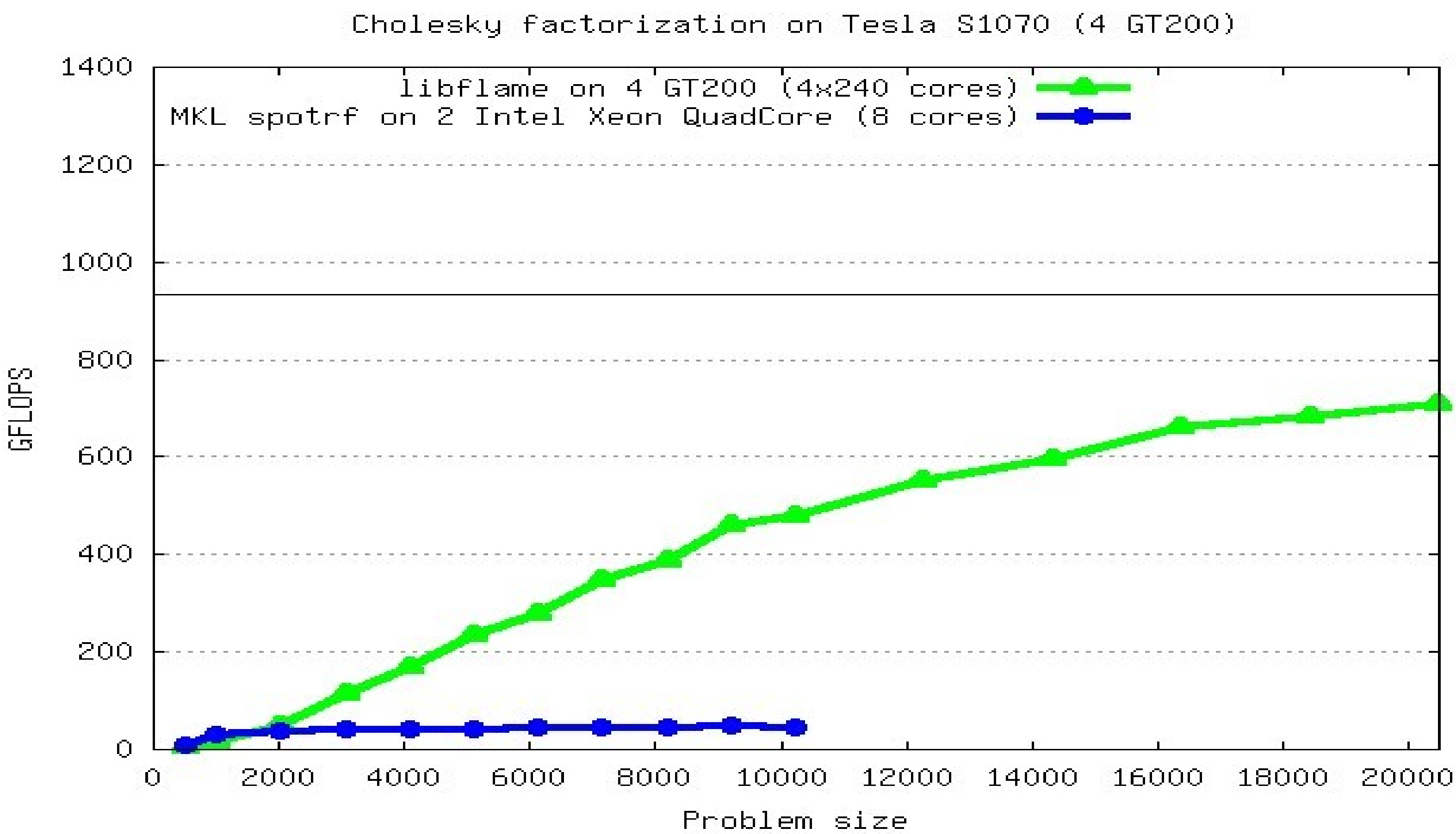
Multi-GPU platform



D.-F. DYNAMIC SCHEDULING + DSM
Performance in multi-GPU platforms



D.-F. DYNAMIC SCHEDULING + DSM
Performance in multi-GPU platforms



MULTI-GPU PLATFORMS

```
while ( FLA_Obj_length( ATL ) < FLA_Obj_length( A ) ) {
  /* 2x2->3x3 repartitionings of A, B and C */
  FLA_Gemm( ..., A10, B01, ..., C11 );
  FLA_Symm( ..., A11, B11, ..., C11 );
  /* 3x3->2x2 repartitionings of A, B and C */
}
```

- Scheduling and separate memory address spaces hidden under the covers (run-time)
- No changes to `libflame`!

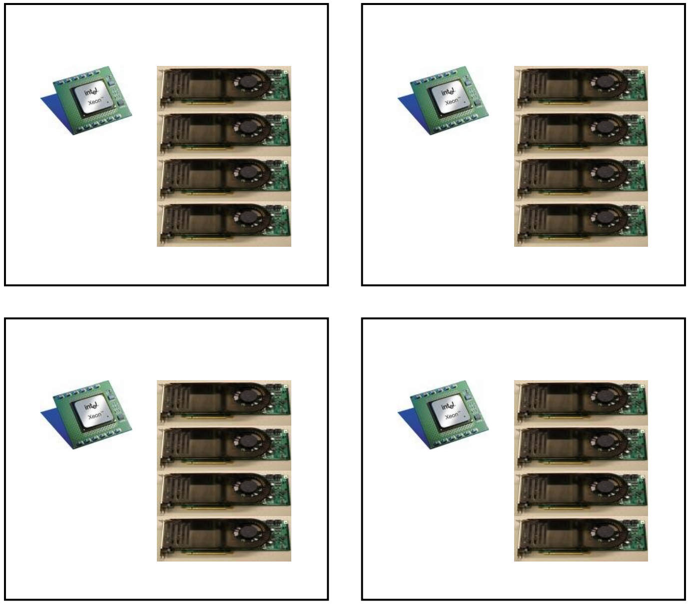
OUTLINE

- Evaluation and tuning of CUBLAS
- Superscalar techniques in the construction of dense linear algebra libraries for multi-core processors and GPUs:
  1. Data-flow dynamic scheduling
  2. DSM

- Clusters of GPUs

CLUSTERS of GPUs

- CPU-hardware accelerators
  - More favourable price-performance ratio
  - Multi-GPU have limited scalability



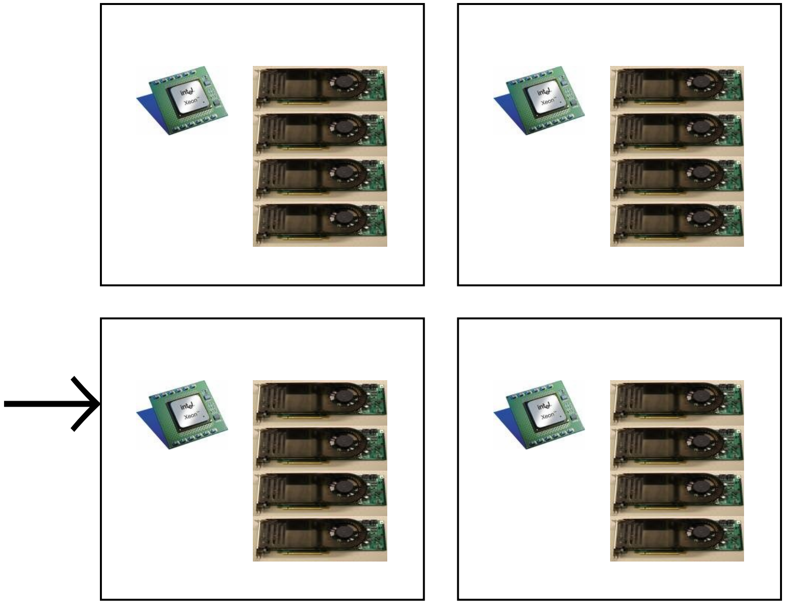
CLUSTERS of GPUs

- PLAPACK
  - Message-passing dense linear algebra
  - Object-based approach, like `libflame`
  - Communication cleanly separated from computation
    - Copies between objects with different distributions: `PLA_Copy` & `PLA_Reduce`

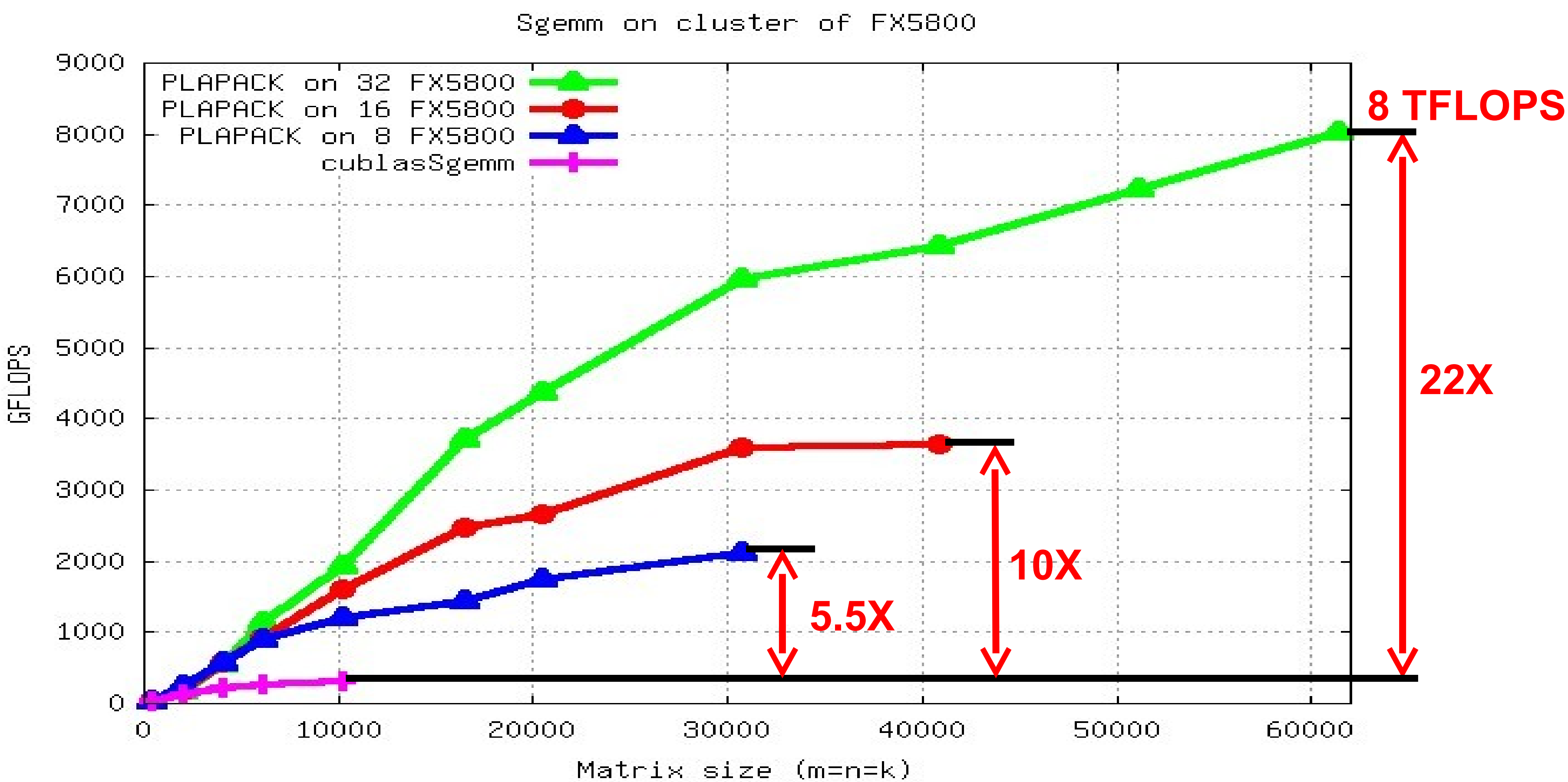
CLUSTERS of GPUs

- Reduce #data transfers
  - Keep data in device memory:
    - Transfer data to main memory only during communication
    - Use communication packing to transfer

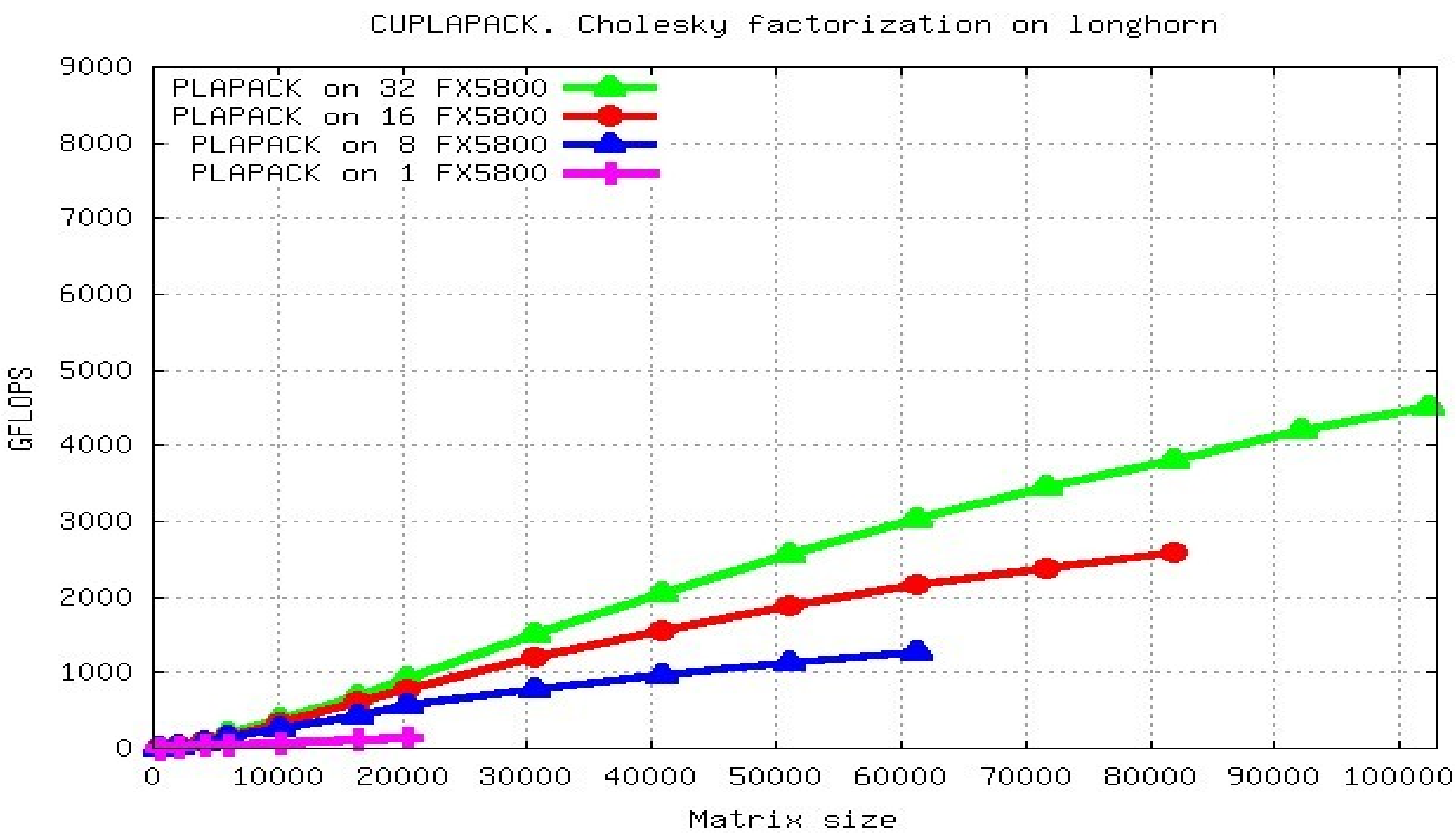
Cluster of GPUs



CLUSTERS of GPUs Performance



CLUSTERS of GPUs
Performance



CLUSTERS of GPUs

```
while ( TRUE ) {
  /* Split ABR into 2x2 views */
   PLA_Local_Chol( A11 );
  PLA_Trsm( ..., A11, A21 );
  PLA_Syrk( ..., A21, ..., A22 );
}
```

- Separate memory address spaces hide under the communication routines
- No other changes to PLAPACK!

CONCLUSIONS

"High-level programming (through abstraction) eases the programmability problem posed by new architectures without sacrifyzing high performance"

- FLAME
- PLAPACK

CONCLUSIONS

Dilated experience with linear algebra on GPUs:

- "Evaluation and tuning of the level 3 CUBLAS for graphics processors". UJI TR ICC 2008-01-01, **Jan. 2008** -- PDSEC 2008.
- "Solving dense linear systems on graphics processors". UJI TR ICC 2008-02-02, **Feb. 2008** -- Euro-Par 2008.
- "Solving dense linear algebra problems on platforms with multiple hardware accelerators". FLAME WN #32, UTCS TR-08-22, **May 2008** -- PPoPP 2009.
- "Exploiting the capabilities of modern GPUs for dense matrix computations". UJI TR ICC 01-11-2008, **Nov. 2008** -- Concurrency & Computation: P&E, 2009.
- "Reduction to condensed forms for symmetric eigenvalue problems on multi-core architectures". ETHZ SAM Report 2009-13, **March 2009** -- PPAM 2009.
- "Level-3 BLAS on a GPU: Picking the Low Hanging Fruit." FLAME WN #37. UJI TR ICC 2009-04-01, **April 2009** -- ICNAAM 2009.
- "Retargeting PLAPACK to Clusters with Hardware Accelerators." FLAME WN #42, UTCS TR-10-06, **Feb. 2010**.

Thanks for your attention!

- For more information: www.cs.utexas.edu/users/flame