Accelerating Direct Sparse Solvers with GPUs

Ryan Schneider, CTO
Acceleware Corp.
September 23, 2010

acceleware
PROCESSING SUPERPOWER

DISCLAIMER

- All the views expressed in this presentation are Acceleware Corp's
- Dassault / Simulia take no responsibility in these views

ACKNOWLEDGEMENTS

Abaqus

Michael Wood
Matt Dunbar
Luis Crivelli

Acceleware

Geraud Krawezik
Chris Mason

PREVIOUS WORK

- Most of the studies so far have been done on low-level operations:
  - GEMM (see Demmel/Volkov SC08 paper)
  - Factorizations
- Only one case has studied the acceleration of multi-frontal solvers (I/IT SEC 2007)
  - But not done with a commerical software package like Abaqus
- SC'08 Demo (with Gene Poole)
- More, as of GTC 2010!?

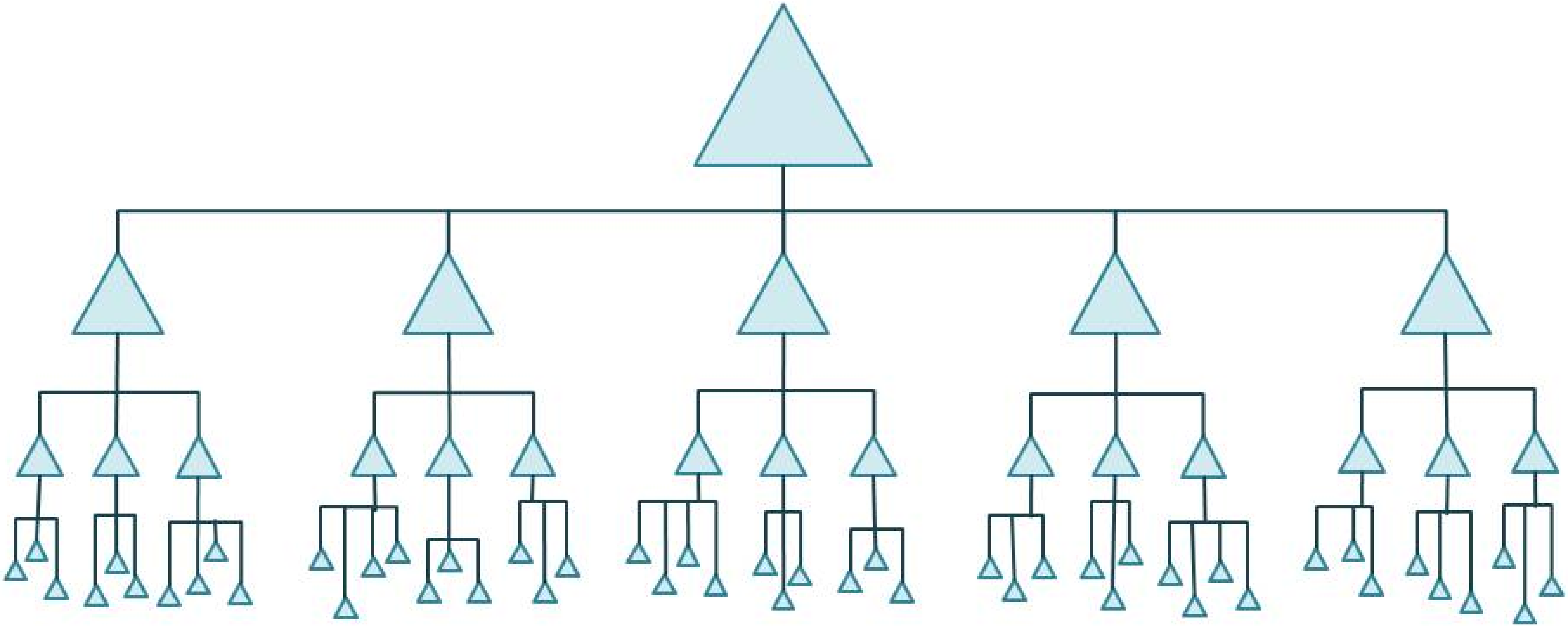
OVERVIEW

- Introduction: multi-frontal solvers
- Acceleware factorization library
  - Interfacing with Abaqus
  - The LDL^T factorization
  - Kernels
  - Results
- Discussion / Conclusion

MULTI-FRONTAL SOLVERS

- Direct sparse solvers
- Often chosen for:
  - Reliability
  - Accuracy
  - Robustness
- Parallelized
  - Shared memory (fine grain: factorization level)
  - Distributed memory (coarse grain: independent fronts)
- Goal: factorization of a large sparse matrix
  - Factorize small dense matrices using LDL^T
  - Assemble these dense matrices

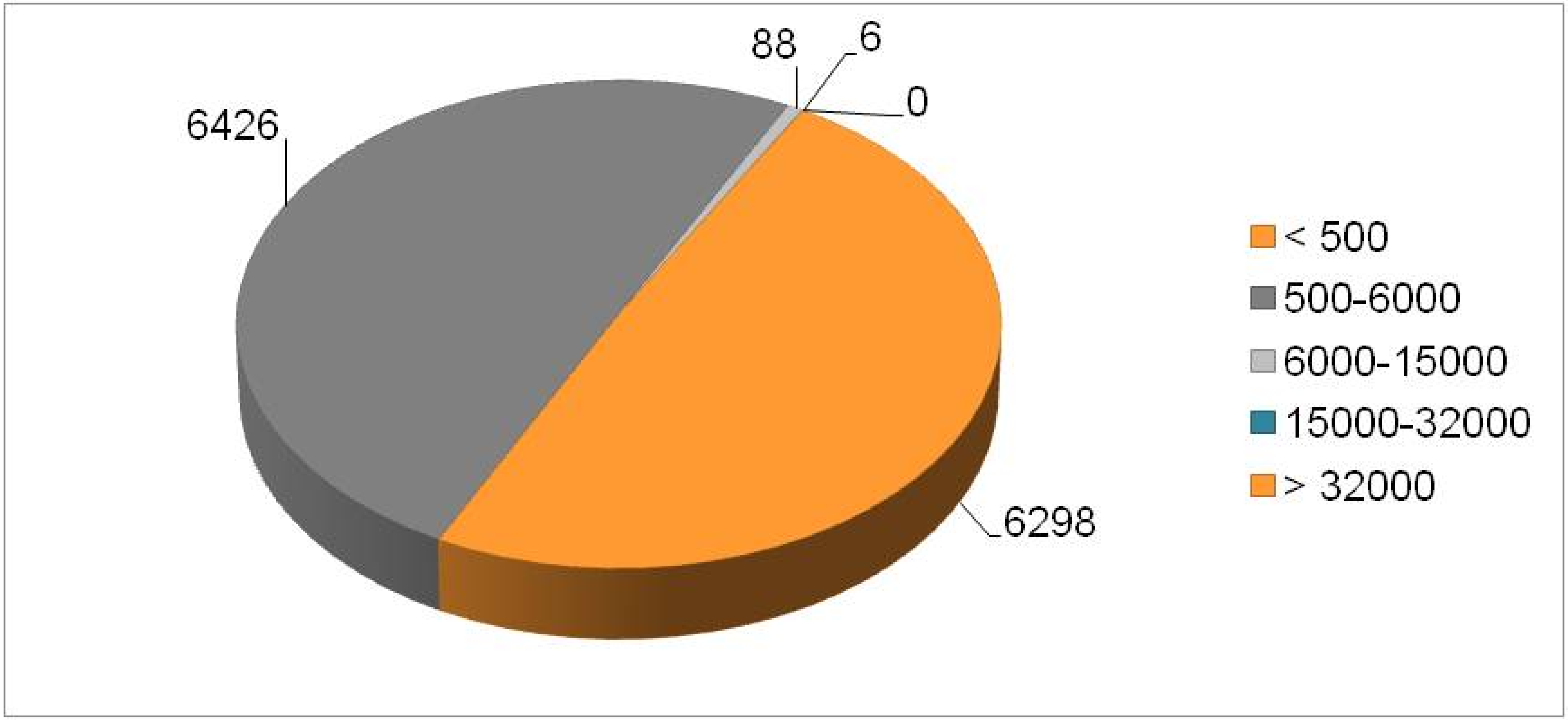
MULTIFRONTAL SOLVER DEPENDENCY TREE



- Many fronts are small and independent

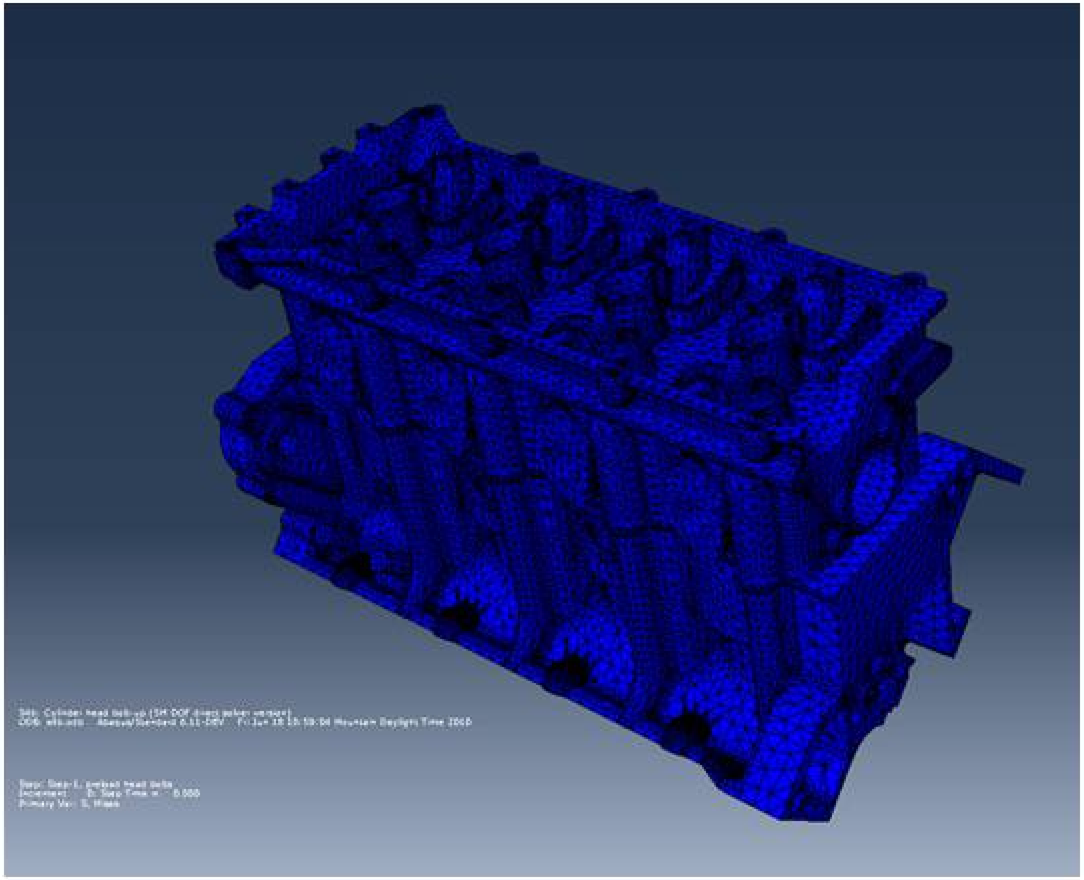
Courtesy Steve Ashcraft

FRONTS DISTRIBUTION: SIZE



- Most of the fronts are small (98% < 6000)

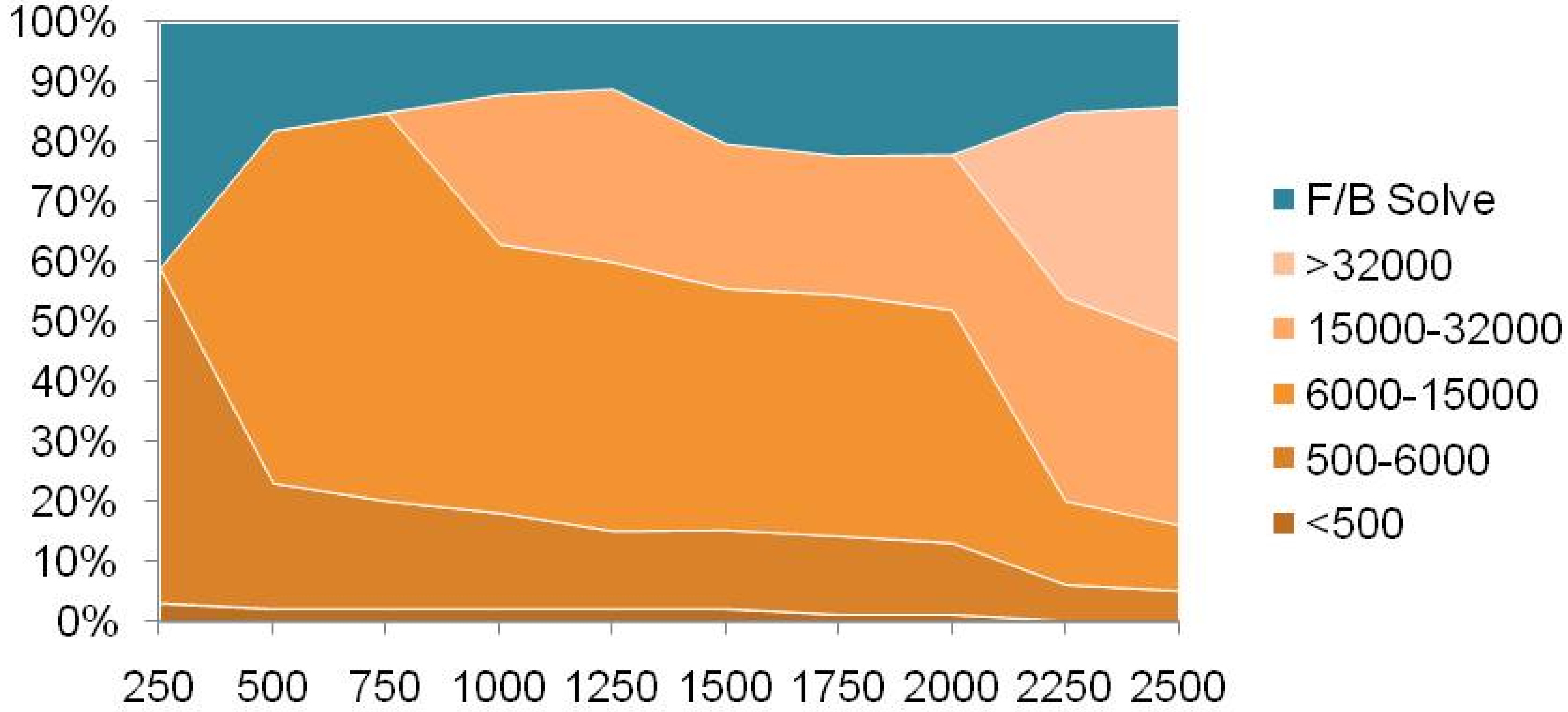
MODEL USED



- Block engine model
- ABAQUS' s4b Benchmark
- Static analysis
  - Displacement
- Realistic model
  - Size(3.2m dofs)
  - Complexity

acceleware
PROCESSING SUPERPOWER

FRONTS DISTRIBUTION: FACTORIZATION TIME

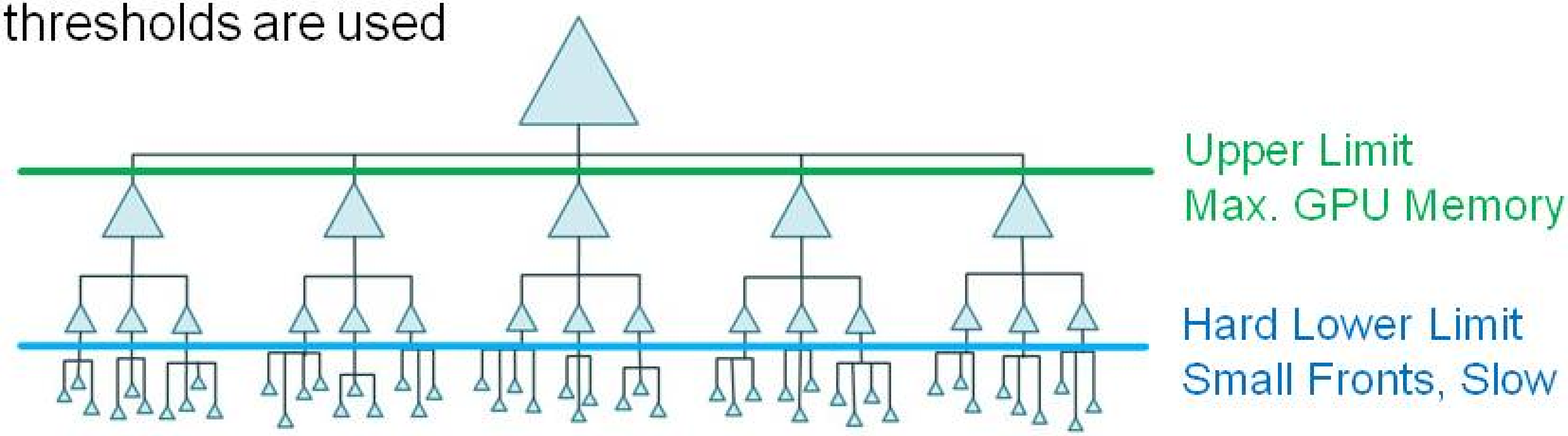


- Most of the time is spent factorizing large fronts

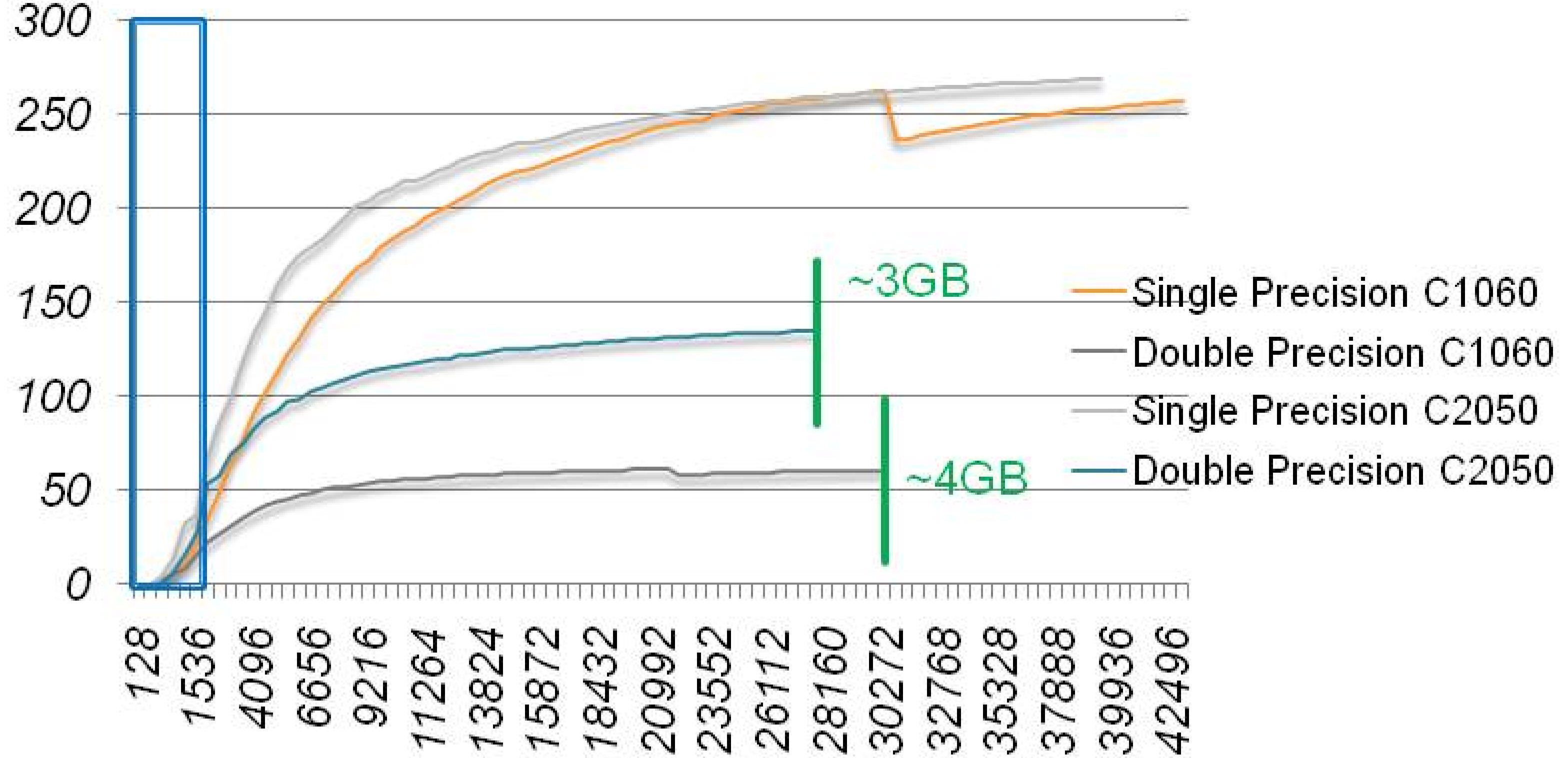
INTEGRATION WITHIN ABAQUS

- ABAQUS 6.X integration
  - Dynamic library replacement
  - Command-line option

- As not all the fronts are suited for GPU computation, several thresholds are used



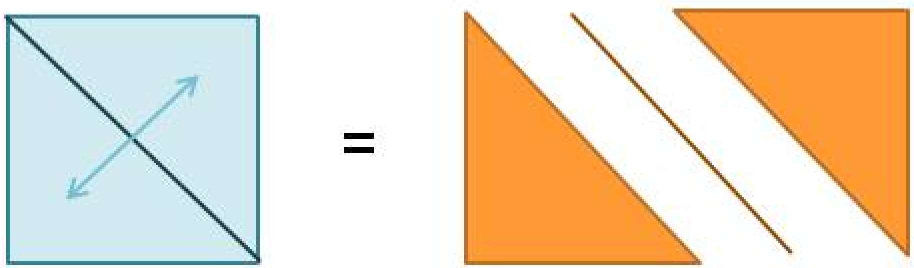
PERFORMANCE: LDL^T FACTORIZATION



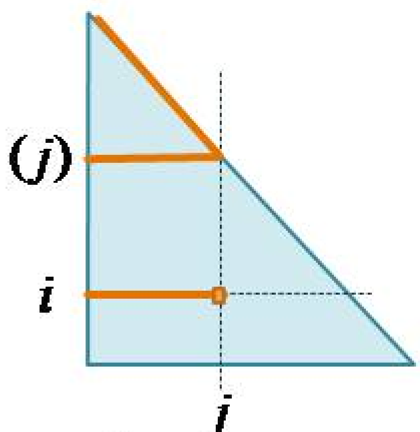
- Performance on Windows 7 (C1060) and Linux (C2050) (GFLOPS with respect to front size)

acceleware
PROCESSING SUPERPOWER

LDL^T FACTORIZATION



- A = LDL^T
  - A: real symmetric matrix
  - L: lower triangular, unit-diagonal
  - D: diagonal matrix
  - A overwritten by L and D

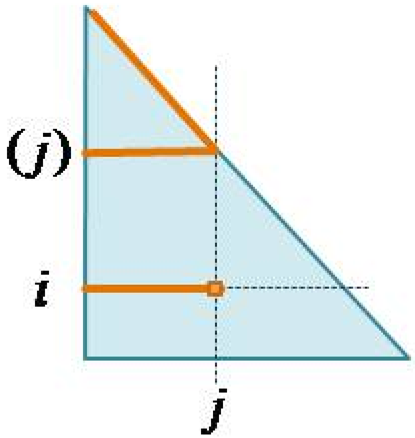


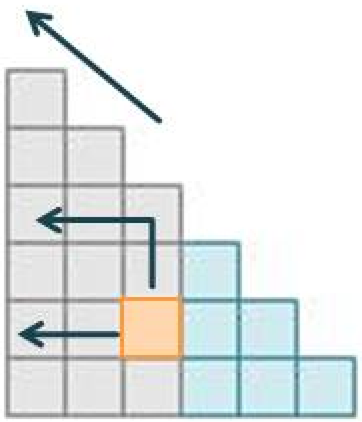
$$\forall(i<j): l_{i,j} = \frac{1}{d_j}\left(a_{i,j} - \sum_{k=1}^{k<j} l_{i,k}\cdot d_k \cdot l_{k,j}\right)$$

$$d_j = a_{j,j} - \sum_{k=1}^{k<j} l_{j,k}\cdot d_k \cdot l_{k,j}$$

acceleware PROCESSING SUPERPOWER

"LEFT-LOOKING" LDL^T FACTORIZATION

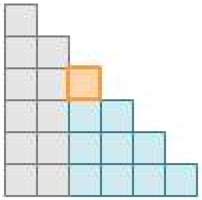




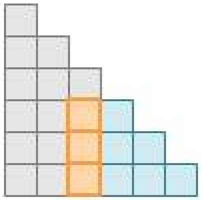
"RIGHT-LOOKING" LDL^T FACTORIZATION

- Relying on matrix-matrix multiplication (BLAS3)
- Very parallel

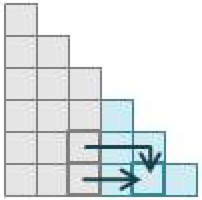
- Implementation notes
  - Matrices stored in Z-order on GPU for better memory accesses
  - This storage leads to better cache utilization for diagonal (1)
  - Packing/unpacking to compact column-major format is done on the host; ie: host format different than under the API



- (1) Diagonal factorization
  - On CPU



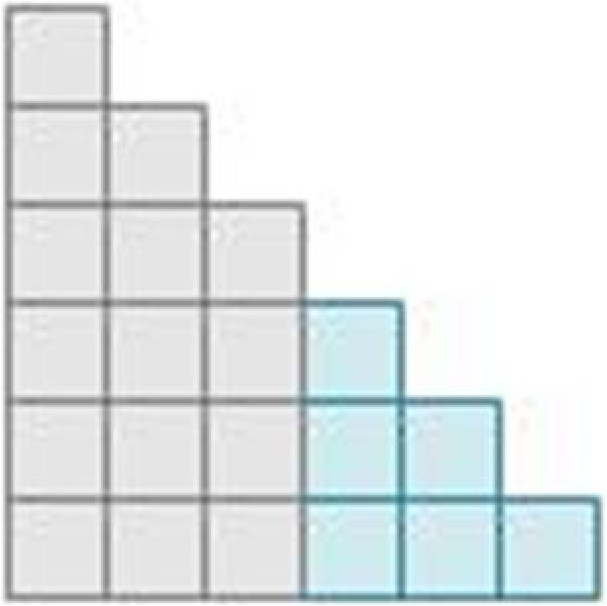
- (2) Non-diagonal factorization (column)



- (3) Right-looking update
  - GEMM $LD * L^T$

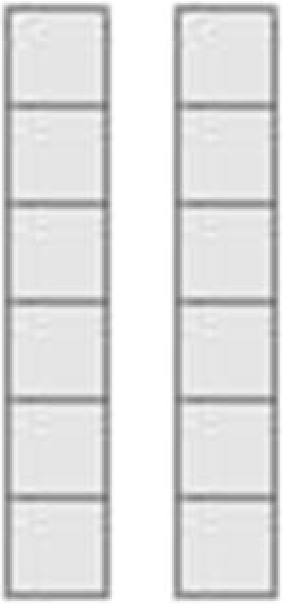
acceleware
PROCESSING SUPERPOWER

MAXIMUM SUPPORTED FRONT SIZE?



Complete and partial sums

+



LD a

OVERALL RESULTS

Job name	Size (dofs)	FLOPS	Solver Speed-up	Overall Speed-up
S4b suspension	3.2 million	1.03e13	3.0 X	2.0 X
S4a	631 thousand	4.34e11	1.8 X	1.2 X
Customer #1	1.5 million	1.70e13	3.7 X	2.3 X
Customer #2	3.7 million	1.68e13	3.4 X	2.0 X

- Dell T5500, dual-socket Xeon E5530 (2.4GHz), 48 GB
- 4 Nehalem*** cores versus 4 Nehalem*** cores + C2050
  - *** dual-socket machine (8 physical cores), but 4 cores saturate
- All the tested models are realistic: Large number of small fronts, few large fronts

acceleware PROCESSING SUPERPOWER

PERFORMANCE ANALYSIS

- Even if most fronts are still factorized on the CPU, we can get acceleration!
  - Speed-up versus 2-cores is up to 4x
  - As only the factorization is accelerated, Amdahl's law is a limiting factor

- Other parts need to be accelerated to provide even greater performance:
  - Forward/backward solve

THE "SPIN"

- What are the alternatives to get 2-3X?
  - Faster CPU (Limited by Memory BW)
  - 4-way or 8-way node (Big SMP); not cheap
  - Amdahl's law
- The value of a "day" or an "engineer"
  - Design iterations
  - Time-to-market
- High-end GPU already installed/required

acceleware PROCESSING SUPERPOWER

CONCLUSION

- A fast GEMM is used to obtain acceleration
  - But it is not enough!
- Tight integration with the whole solver is needed
  - Replacing the CPU BLAS calls by GPU BLAS calls is not enough
  - Raw performance numbers cannot be matched if one does not think about communication (PCI-Express bandwidth: 6GB/s!)
- Use as much parallelism as possible
  - Have the GPU and the CPU collaborate rather than compete: both work on parts that are optimized for their architecture
  - Use asynchronism: communications, parallel GPU/CPU execution
- Influence of the model on performance
  - If there are too many small fronts, the GPU cannot compete with a modern CPU

EASY QUESTIONS?

Contact: Ryan Schneider
Chief Technology Officer
Email: ryan.schneider@acceleware.com
Phone: +1.403.249.9099 x202

acceleware

PROCESSING SUPERPOWER