Accelerating the ANSYS Direct Sparse Solver with GPUs

Géraud KRAWEZIK
Acceleware Corp.
July 28th, 2009
OVERVIEW

- Introduction: multi-frontal solvers
- The LDL^T factorization
- Interfacing with ANSYS
- Experiments
- Discussion
- Conclusion
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
Many fronts are small and independent
MODEL USED

- Wing profile model
- ANSYS’ BM-7 Benchmark
- Static analysis
  - Displacement
- Enables scaling
  - Degrees of freedom
Most of the fronts are small (98% < 6000)
Most of the time is spent factorizing large fronts.
\( \mathbf{A} = \mathbf{L} \mathbf{D} \mathbf{L}^T \)

- \( \mathbf{A} \): real symmetric matrix
- \( \mathbf{L} \): lower triangular, unit-diagonal
- \( \mathbf{D} \): diagonal matrix
- \( \mathbf{A} \) overwritten by \( \mathbf{L} \) and \( \mathbf{D} \)

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

\[
d_j = a_{j,j} - \sum_{k=1}^{k<j} l_{j,k} \cdot d_k \cdot l_{k,j}
\]
**LDLᵀ FACTORIZATION (2)**

- **Right-looking blocked algorithm**
  - 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 blocks (1)
  - Packing/unpacking to compact column-major format is done on the host

- **(1) Diagonal block factorization**
  - On CPU

- **(2) Non-diagonal blocks factorization**

- **(3) Right-looking update**
  - GEMM LD * Lᵀ
TEST PLATFORM

Host
- 2x Quad-Xeon 2.0GHz
- PCI-Express gen2 x16
  - Up to 6GB/s with CUDA
- RAM: 32 GB
- Windows XP 64

Accelerator
- NVIDIA Tesla C1060
- GT200 240 cores (1.3 GHz)
  - 933 GFlops (32-bit)
  - 78 GFlops (64-bit)
- RAM: 4 GB DDR3
  - 102 GB/s
Performance: LDLᵀ Factorization

- Performance (GFlops with respect to front size)
  - Single precision memory bound / Double precision compute bound
ANSYS 12.0 integration
- Dynamic library replacement
- Using ANSYS 12.0 customization
- Command-line option

As not all the fronts are suited for GPU computation, two thresholds are used
- Lower: all fronts under it would provide poor performance
- Upper: all fronts above it would require more memory than available on the GPU
PERFORMANCE: BM-7 WING MODEL

Speed-up over a dual-Xeon run
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
Most of the studies so far have been done on low-level operations:

- GEMM (see Demmel/Volkov SC08 paper)
- Factorizations (LU, Cholesky, QR)

Only one case has studied the acceleration of multi-frontal solvers (I/IT SEC 2007)

- But not done with a legacy software package like ANSYS
Influence of the model:

- The wing is blocky and has large fronts that take up most of the computation time. The larger the model, the better the acceleration
- What about models with thin structures?

Single precision factorization leads to little difference but great performance

- How would utilizing iterative refinement scheme impact the performance?
- Can we just wait for the next hardware generation?

Small fronts do not lead to large acceleration

- Parallelize their factorization between CPU and GPU?
QUESTIONS?