PARALLEL REALISATION OF THE ELEMENT-BY-ELEMENT FEM TECHNIQUE BY CUDA

Imre Kiss*1, Szabolcs Gyimóthy2, Zsolt Badics2 and József Pávo*
1Budapest University of Technology and Economics, Egry J. u. 18, H-1521 Budapest, Hungary
2Tensor Research, LLC, Andover, MA, U.S.A
1email: kiss@eit.bme.hu

ABSTRACT

In this work the utilization of Graphical Processing Units (GPUs) for the element-by-element (EBe) finite element method (FEM) is demonstrated. EBeFEM is a long since known technique, by which a conjugate gradient (CG) type iterative solution scheme can be entirely decomposed into computations on the element level, i.e. without assembling the global system matrix. In our implementation NVIDIA's parallel computing solution, the Compute Unified Device Architecture (CUDA) is used to perform the required element-wise computations in parallel. Since element matrices need not be stored, the memory requirement can be kept extremely low. It is shown that this low-store but computation-intensive technique is better suited for GPUs than those requiring the massive manipulation of large data sets.

CONCLUSIONS

1. In this work the EBeFEM method was re-formulated to fit the unique GPU architecture. The method has extremely low memory consumption, and can take full advantage of a massively parallel execution environment. These attributes make the execution of the algorithm extremely fast, which not only outperform traditional CUDA accelerated FEM methods, but makes it competitive with today's multi-CPU based algorithms (execution times are well below those obtained by commercial CPU based programs).

2. Multi-GPU results demonstrate the scalability of the method. For small sized problems the communication overhead among devices increases the total execution time, but for large problems the expected twofold acceleration could be achieved. The method can be extended to arbitrary number of GPUs. This topic together with the treatment of non-linear problems will be covered in a forthcoming paper.

Element-by-Element Finite Element Method

- Overview of the method
  - The element-by-element FEM method was constructed originally for low memory computers.
  - The foundation of the method is based on the recognition that each element matrices to form the global system matrix is in a linear operation.
  - Certain calculations with the system matrix (like e.g. a matrix-vector product) can be traced back to the level of finite elements, thus converted to calculations with the individual element matrices, $A_e$, appearing in the elementary equations having the form:
    \[ A_e x = b \]
    - Where $x$ is the potential values at the DoFs of the element, $b$ is a constant vector.
  - Since iterative solvers can be decomposed into a sequence of matrix-vector products and inner products of vectors, the idea comes natural not to store the element matrices (as traditional methods do with the system matrix), but recompute them in each iteration.
  - The introduced technique can be thought of as one which transforms a highly memory dependent problem to a massively computational dependent one, which in turn can be efficiently parallelized.

- Disassembling Matrix Manifolds to the Element Level
  - The FEM assembling procedure relies on some functions to generate the element matrix $A_e$ and the RHS $b_e$.
  - Composed element matrices and RHS vectors are assembled to form the global system matrix $A$ and RHS $b$.
  - The assembly step is represented by an operator $M$, defined differently for matrices and vectors as:
    \[ A = M(A_e) = \sum C_i A_e C_i \]
    \[ b = M(b_e) = \sum C_i b_e \]
    where $C$ is the set of elements, and matrix $C$ represents the transition between the local and global numbering of the unknown variables for the $i$-th element. Contrary to the sparse global system matrix $A$, the element matrix $A_e$ is usually dense.

- Elementary matrix-vector products can be computed for each element separately, as:
  \[ A_e x = \sum C_i A_e C_i, x = \sum C_i A_e C_i \]
  - One more advantage of the EBe implementation worth mentioning is that no global numbering of unknowns and finite elements is required at all.

CUDA processing

- GPU parallel execution of the EBeFEM method
  - Today's GPUs can take full advantage of the EBe FEM method, due to its massively parallel design.
  - Large scale FEM problems need large storage capacity for the global system matrix. EBeFEM requires only the calculation of the matrix-vector product with respect to the finite elements of the given problem (the limited memory).
  - GPUs have the potential to dramatically speed-up computation intensive applications over multi-core CPUs. To achieve high computational throughput, they have hundreds of lightweight cores and execute trillions of thousands of threads simultaneously on their SIMT architectures.
  - Refining the fact that it is cheaper to recompute element matrices than continuously cache them between device and the system memory, EBe-FEM technique is ideally suited to GPUs.

- Concurrency - Global Updates and Coloring
  - On shared memory architectures, an important question is how the partial products are summarized. During a global update it must be ensured that different threads do not access the same memory space simultaneously.
  - Concurrency access is called race-condition and results in an indefinite outcome. Treatment of such cases is traditionally of two kinds. One solution is so-called “atomic” update, when the memory space is protected during I/O, causing other threads accessing the same memory place to wait until the operation is fully completed.
  - The other solution is a kind of “coloring” of the problem. In this case the mesh is considered as a graph, with the unknown variables (DoF) being the nodes of the graph and the elements representing the connections between them. This graph is then colored in a way that any two elements having the same color do not share a common unknown. Different colors are then processed “serially”, while elements having the same color in parallel.

- Element-by-Element formulation of the BICG solver
  - The way the variables are stored gives the real modularity of the EbE method. Contrary to traditional FEM methods, the massive manipulation of large data sets.
  - Certain calculations with the system matrix (like e.g. a matrix-vector product) can be traced back to the level of finite elements, thus converted to calculations with the individual element matrices, $A_e$, appearing in the elementary equations having the form:
    \[ A_e x = b \]
    - Where $x$ is the potential values at the DoFs of the element, $b$ is a constant vector.
  - Since iterative solvers can be decomposed into a sequence of matrix-vector products and inner products of vectors, the idea comes natural not to store the element matrices (as traditional methods do with the system matrix), but recompute them in each iteration.
  - The introduced technique can be thought of as one which transforms a highly memory dependent problem to a massively computational dependent one, which in turn can be efficiently parallelized.

- The other solution is a kind of “coloring” of the problem. In this case the mesh is considered as a graph, with the unknown variables (DoF) being the nodes of the graph and the elements representing the connections between them. This graph is then colored in a way that any two elements having the same color do not share a common unknown. Different colors are then processed “serially”, while elements having the same color in parallel.

- Element-by-Element formulation of the BICG solver
  - The way the variables are stored gives the real modularity of the EbE method. Contrary to traditional FEM methods, the massive manipulation of large data sets.
  - Certain calculations with the system matrix (like e.g. a matrix-vector product) can be traced back to the level of finite elements, thus converted to calculations with the individual element matrices, $A_e$, appearing in the elementary equations having the form:
    \[ A_e x = b \]
    - Where $x$ is the potential values at the DoFs of the element, $b$ is a constant vector.
  - Since iterative solvers can be decomposed into a sequence of matrix-vector products and inner products of vectors, the idea comes natural not to store the element matrices (as traditional methods do with the system matrix), but recompute them in each iteration.
  - The introduced technique can be thought of as one which transforms a highly memory dependent problem to a massively computational dependent one, which in turn can be efficiently parallelized.

- Disassembling Matrix Manifolds to the Element Level
  - The FEM assembling procedure relies on some functions to generate the element matrix $A_e$ and the RHS $b_e$.
  - Composed element matrices and RHS vectors are assembled to form the global system matrix $A$ and RHS $b$.
  - Let this assembly step be represented by an operator $M$, defined differently for matrices and vectors as:
    \[ A = M(A_e) = \sum C_i A_e C_i \]
    \[ b = M(b_e) = \sum C_i b_e \]
    where $C$ is the set of elements, and matrix $C$ represents the transition between the local and global numbering of the unknown variables for the $i$-th element. Contrary to the sparse global system matrix $A$, the element matrix $A_e$ is usually dense.

- Elementary matrix-vector products can be computed for each element separately, as:
  \[ A_e x = \sum C_i A_e C_i, x = \sum C_i A_e C_i \]
  - One more advantage of the EBe implementation worth mentioning is that no global numbering of unknowns and finite elements is required at all.

Numerical results

- The chosen test problem is a static electric current flow problem with inhomogeneous conductivity. The equation to be solved is therefore the generalized Laplace equation:
  \[ \nabla \cdot D \mathbf{E} = 0 \]
  - The domain is discretized by tetrahedral elements of linear nodal shape functions. The global unknowns (DoF) are the potentials $\varphi$ at the nodes of the mesh.
- To study the accuracy and speed of the proposed method, the "Utah Torus" triangle model was investigated regarding electrocardiographic (ECG) field computation.
  - Computing statistics obtained for different problem sizes (mesh densities) are shown in Table 1 on the right side.
  - The model was meshed manually to obtain different mesh qualities. The computations were carried out on a HP- XW8600 workstation, having an NVIDIA GTX 580 video card.