Architecting the Finite Element Method Pipeline for the GPU

Zhisong Fu, T. James Lewis, Robert M. Kirby, Ross T. Whitaker
The Scientific Computing and Imaging Institute at the University of Utah

SCI X 2012

Motivation

In this project, we consider the numerical solution of the second order elliptic PDEs defined on a three dimensional tetrahedral domain with Finite Element Method (FEM):

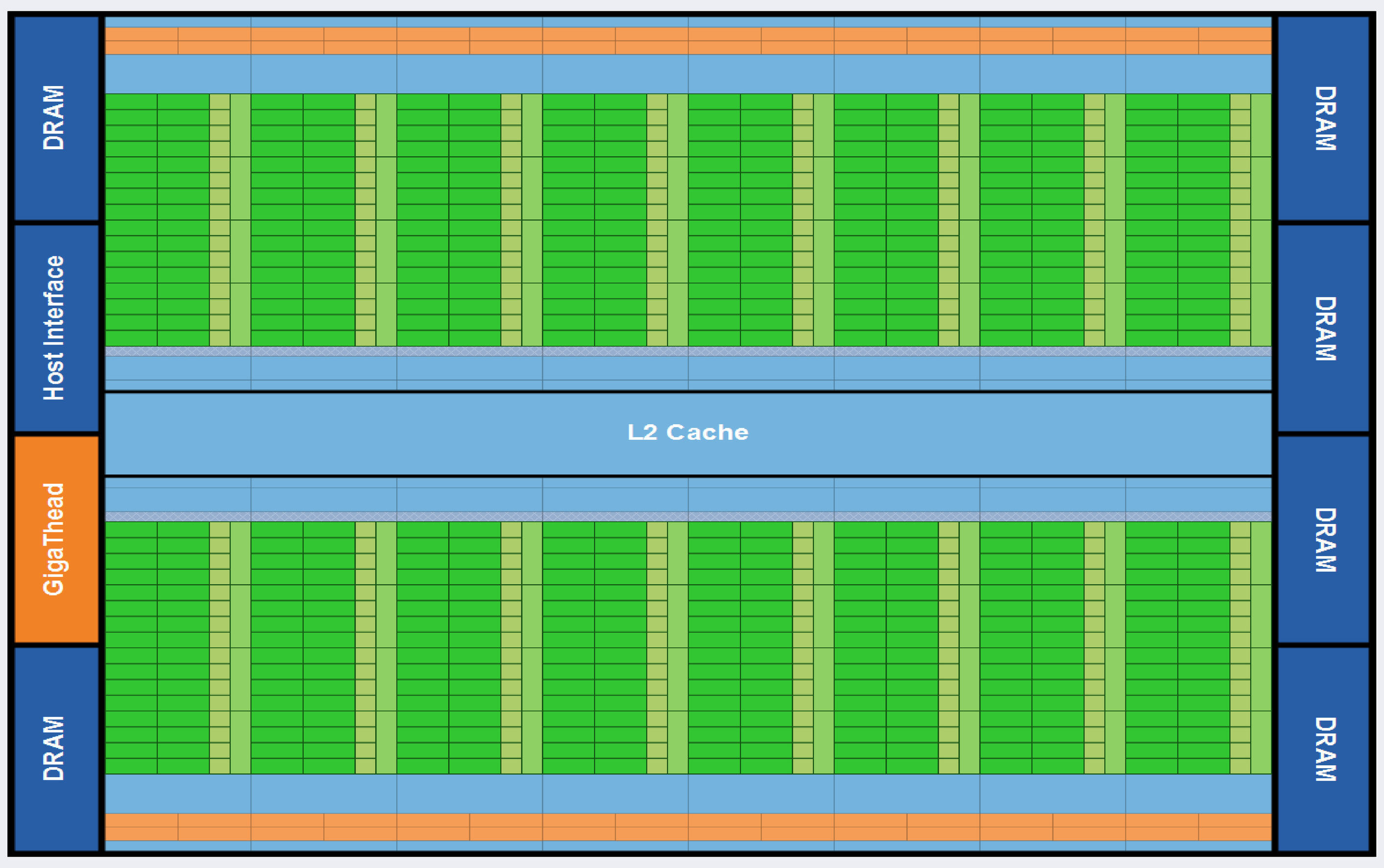
$$-\nabla \cdot (\sigma(\mathbf{x}) \nabla u(\mathbf{x})) + \lambda u(\mathbf{x}) = f(\mathbf{x})$$

$$\sum_{j=1}^{N} (\nabla \phi_i, \sigma \nabla \phi_j) \tilde{u}_j + \lambda \sum_{j=1}^{N} (\phi_i, \phi_j) \tilde{u}_j = (\phi_i, f)$$

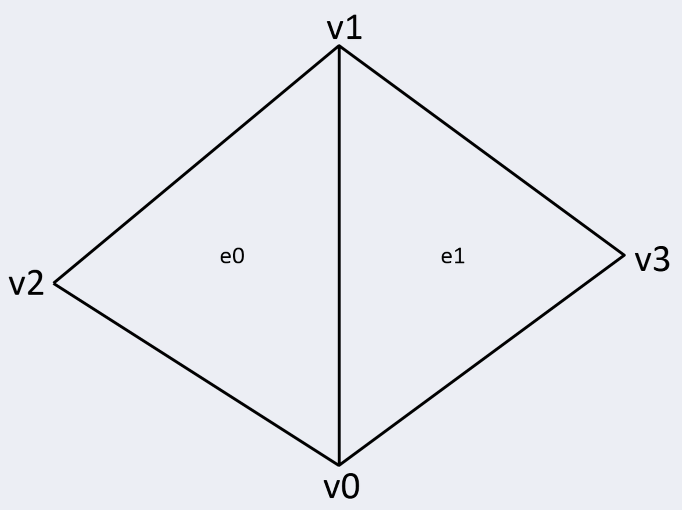
- The second order elliptic PDEs appear in many scientific and engineering problems.
- The FEM is a widely used method for solving the PDEs.

Background

1. GPU architecture



2. Compact assembly step



$$A^{e_0} = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} \qquad A^{e_1} = \begin{bmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \\ b_{20} & b_{21} & b_{22} \end{bmatrix}$$

$$A = \begin{bmatrix} c_{00} & c_{01} & c_{02} & c_{03} \\ c_{10} & c_{11} & c_{12} & c_{13} \\ c_{20} & c_{21} & c_{22} & c_{23} \\ c_{30} & c_{31} & c_{32} & c_{33} \end{bmatrix}$$

3. Geometry-informed AMG

- Geometry-informed partitioning for aggregates and patches.
- Smoothed aggregation multigrid.
- Block-Jacobi relaxations.
- New V-cycle.

Algorithm 5.3: $V\text{-}cycle\text{-}new(A_I^k, A_B^k, R^k, P^k, b^k, u^k)$

if *level k is the coarsest level*
then *solve* $A^k u^k = b^k$ *and return* u^k

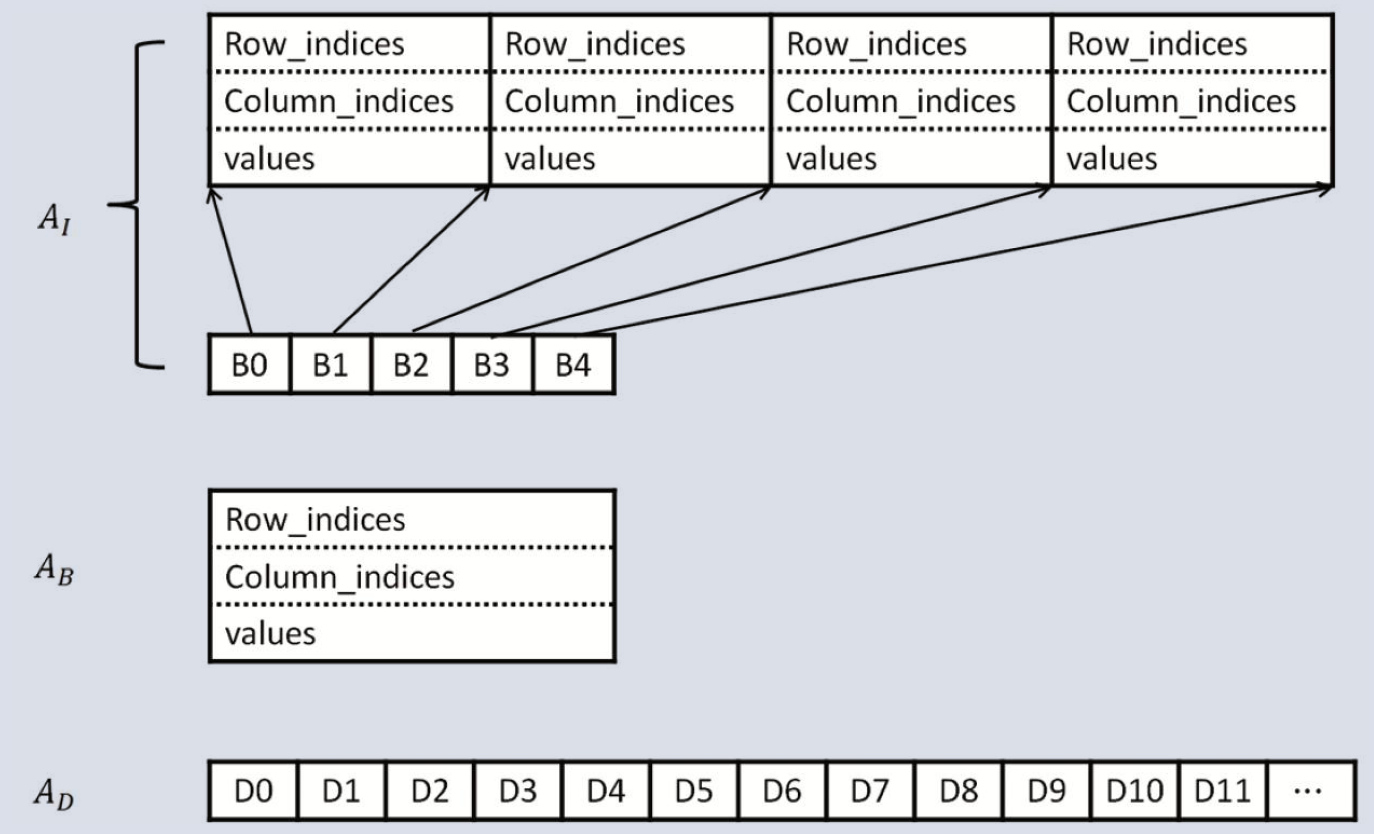
$$\textbf{else} \begin{cases} u^k, \tilde{r}_k \leftarrow \text{pre-relax-residual}(A_I, u^k, b^k) \\ r^k \leftarrow \tilde{r}^k - A_B^k u^k \\ r^{k+1} \leftarrow R^k r^k \\ e^{k+1} \leftarrow \text{V-cycle}(A_I^{k+1}, A_B^{k+1}, R^{k+1}, P^{k+1}, r^{k+1}) \\ u^k \leftarrow P^k e^{k+1} + u^k \\ \tilde{b}^k \leftarrow b^k - A_B u^k \\ u^k \leftarrow \text{post-relax}(A_I, u^k, \tilde{b}^k) \end{cases}$$

Implementation

1. Bottom-up double partitioning with k-MIS.

- Partition the nodes into aggregates.
- Build induced graph from aggregates.
- Partition again into patches.

2. Patch sparse matrix format (patchSPM) data structure.



Result

- CPU: Intel i7 965 Extreme, 3.2GHz, 8MB L3 cache
- GPU: Nvidia GTX 580, 1544MHz, 512 core

For the assembly step, we compare our GPU implementation with our optimized-CPU implementation.

meshes	GPU	CPU	speedup
Regular	0.0298	1.080	36
Irregular	0.0229	1.010	44
Heart	0.0465	3.114	67
Brain	0.0355	3.077	87
Blobs	0.0319	2.525	79

For the iteration step, we compare our method with state of the art CPU and GPU libraries

meshes	patch PCGAMG	Hypre-PCGAMG	S1	CUSP-PCGAMG	S2	CUSP-CG	Hypre-CG	S3
Regular	0.139(19)	3.86(25)	28	0.175(36)	1.3	0.680(329)	3.73(329)	5
Irregular	0.167(31)	3.02(29)	18	0.216(56)	1.3	2.43(1639)	14.8(1639)	6
Heart	0.218(20)	11.2(31)	51	0.631(46)	2.9	4.64(1148)	33.8(1131)	7
Brain	0.165(19)	7.78(27)	47	0.432(45)	2.6	8.15(1838)	60.4(1810)	9
Blobs	0.172(23)	5.70(28)	33	0.409(50)	2.4	3.34(1048)	16.0(1030)	5