Parallel Finite Element Implementations using Different Data Structures

Leonardo Muniz de Lima
Bruno Zanetti Melotti
Lucia Catabriga
Andréa Maria Pedrosa Valli

LCAD – Laboratório de Computação de Alto Desempenho - UFES

Outline

- Motivation
- Parallel Finite Element Algorithm
- Iterative Solution of the Linear System and Storage Schemes
- Numerical Experiments and Performance Results
- Conclusions and Future Work

Motivation

- Parallel implementation of the finite element method using block-arrowhead structure (Saad, 1995 and Jimack&Touheed, 2000)
- Krylov based methods (GMRES, Bi-CGSTAB, etc): matrix-vector product and inner product
- Storage Schemes: EBE, EDE and CSR
- Domain Decomposition, MPI and Cluster of Workstations
- Cluster Enterprise: 64 ATHLON's XP 1800, 256 RAM, 20 GB, 3COM TX Fast-Ethernet

Parallel FE Algorithm

- Governing Equation

$$\frac{\partial u}{\partial t} + \beta \nabla u - \nabla \cdot k \nabla u = f$$

- SUPG Finite Element Formulation

$$\int_{\Omega} \left(w^h \frac{\partial u^h}{\partial t} + w^h \beta \cdot \nabla u^h - \nabla w^h \cdot \kappa \nabla u^h \right) d\Omega \; + \sum_{e=1}^{n_{el}} \int_{\Omega_e} \tau_{SUPG} \beta^h \cdot \nabla w^h \left(\frac{\partial u^h}{\partial t} + \beta^h \cdot \nabla u^h \right) d\Omega \; = \int_{\Omega} w^h f \, d\Omega + \sum_{e=1}^{n_{el}} \int_{\Omega_e} \tau_{SUPG} \cdot \nabla w^h f \, d\Omega$$

→ $Ma + Kv = F$

↓

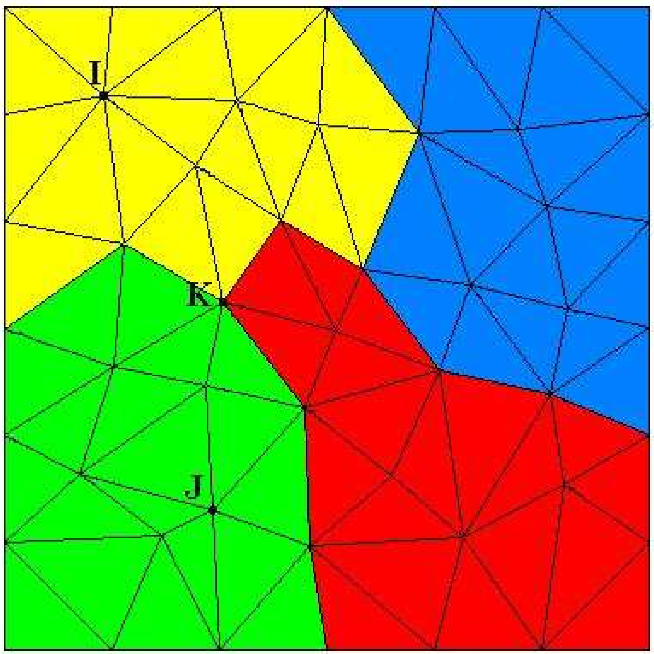
$M^* \Delta a = R$

where:

- $M^* = M + \alpha \Delta t K$
- $R = F - (Ma^* + Kv^*)$

Parallel FE Algorithm

- A partition of a non-uniform mesh into 4 pieces



$$
\begin{bmatrix}
A_1 & & & & B_1 \\
& A_2 & & & \\
& & \ddots & & \vdots \\
& & & A_p & B_p \\
C_1 & C_2 & \cdots & C_p & A_s
\end{bmatrix}
\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \\ x_s \end{bmatrix}
=
\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_p \\ b_s \end{bmatrix}
$$

A_i → contributions from the coupling between interior nodes

B_i and C_i → contributions from coupling between the interface nodes and the nodes interior to sub-domain *i*

$A_s = \sum_{i=1}^{p} A_{s(i)}$ → contributions from coupling between the interface nodes

Storage Schemes

- **Global Strategy:** implementations of global matrix-vector products and global inner products are performed by A_i, B_i, C_i and A_s global blocks.

Global Storage	Dimensions	Average Cost
A_i	$n_I \times n_I$	n_I^2
B_i	$n_I \times n_B$	$n_I \cdot n_B$
C_i	$n_B \times n_I$	$n_I \cdot n_B$
A_s	$n_B \times n_B$	n_B^2

Matrix-vector product

Intnodes	Ibnodes
$y_i = A_i x_i + B_i x_{s(i)}$	$y_{s(i)} = \Sigma_{i=1,p} C_i x_i + A_{s(i)} x_{s(i)}$

Inner product

Intnodes	Ibnodes
$\Sigma_{i=1,p} u_i v_i$	$\Sigma_{i=1,p} u_{s(i)} v_{s(i)}$

Storage Schemes

- **Element by Element Strategy:** implementations of matrix-vector products and inner products are performed by element level.

EBE Storage	By element	By node
A_i	$6\ ebe_{Ai}$	$12\ n_I$
B_i	$6\ ebe_{Bi}$	$12\ n_B$
C_i	$6\ ebe_{Ci}$	$12\ n_B$
A_s	$6\ ebe_{As}$	$12\ n_B$

Matrix-vector product

ebe_{Ai}	ebe_{Bi}	ebe_{Ci}	$ebe_{As(i)}$
$y_i = A_i x_i$	$y_{s(i)} = B_i x_{s(i)}$	$y_i = \Sigma_{i=1,p} C_i x_i$	$y_{s(i)} = \Sigma_{i=1,p} A_{s(i)} x_{s(i)}$

Storage Schemes

- **Edge by Edge Strategy:** by disassembling the resulting FE matrix into their edge contributions, matrix-vector products are computed based on edge data structures.

EDE Storage	By edge	By node
A_i	$2\ ede_{Ai}$	$6\ n_I$
B_i	$2\ ede_{Bi}$	$4\ n_B$
C_i	$2\ ede_{Ci}$	$4\ n_B$
A_s	$2\ ede_{As}$	$4\ n_B$

Matrix-vector product

ede_{Ai}	ede_{Bi}	ede_{Ci}	$ede_{As(i)}$
$y_i = A_i x_i$	$y_{s(i)} = B_i x_{s(i)}$	$y_i = \Sigma_{i=1,p} C_i x_i$	$y_{s(i)} = \Sigma_{i=1,p} A_{s(i)} x_{s(i)}$

Storage Schemes

- **Compressed Sparse Row Strategy:** implementations of global matrix-vector products and global inner products are performed by global blocks A_i, B_i, C_i and A_s storiging only nonzeros positions.

standard	CSR storage	By node
A_i	AA_i , JA_i , IA_i	$7n_I$, $7n_I$, $7n_I$
B_i	BB_i , JB_i , IB_i	$2n_B$, $2n_B$, $2n_B$
C_i	CC_i , JC_i , IC_i	$2n_B$, $2n_B$, $2n_B$
A_s	$AA_{s(i)}$, $JA_{s(i)}$, $IA_{s(i)}$	$3n_B$, $3n_B$, $3n_B$

A_i	B_i	C_i	A_s
$y_i = MV(AA_i , JA_i , IA_i,x_i)$	$y_{s(i)} = MV(BB_i , JB_i , IB_i,x_{s(i)}$	$y_i = MV(CC_i , JC_i , IC_{i,} x_i)$	$y_{s(i)} = MV(AA_{s(i)} , JA_{s(i)} , Ia_{s(i)},x_{s(i)}$

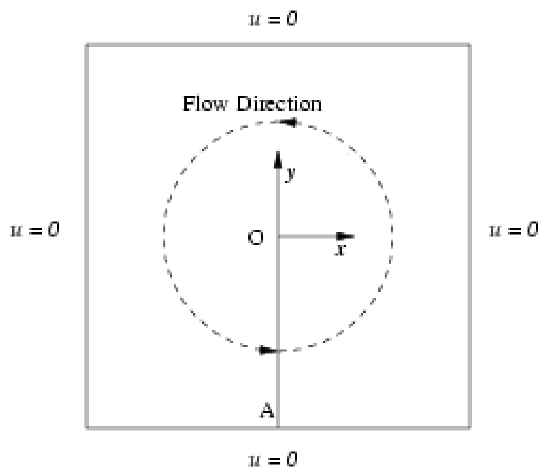
Complexity

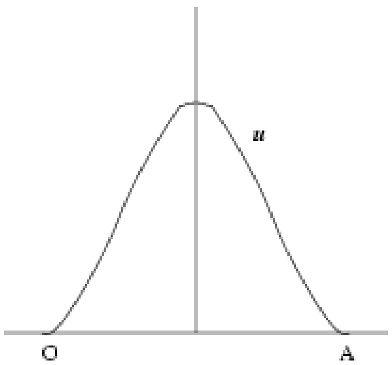
- Complexity of the matrix-vector product and inner product

Operation	Parallel	Serial
Matrix-vector CSR	$1/p\ (7N_I + 7N_B)$	$7N_I$
Matrix-vector EBE	$1/p\ (18N_I + 42N_B)$	$18N_I$
Matrix-vector EDE	$1/p\ (12N_I + 12N_B)$	$12N_I$

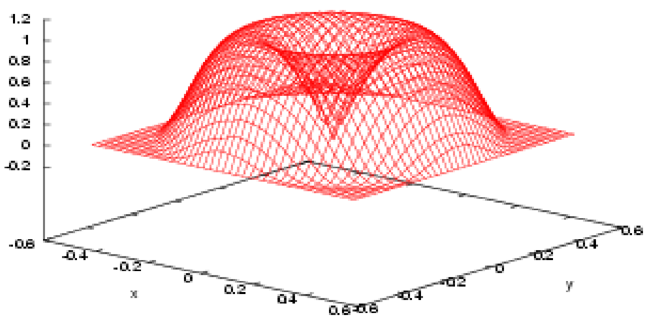
Performance Results

- Advection of a cosine hill in a rotating flow field





- B = $[-y,x]^T$
- $\kappa_x = \kappa_y = 10^{-6}$
- $Tol_{GMRES} = 10^{-6}$



Elevation of *u* with 4 processors and CSR storage

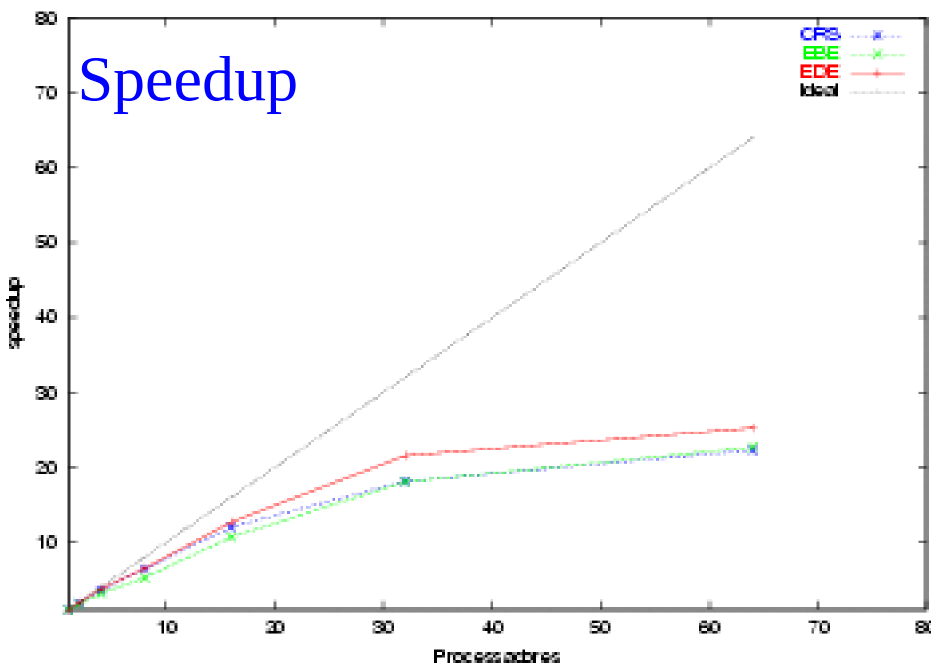
Performance Results

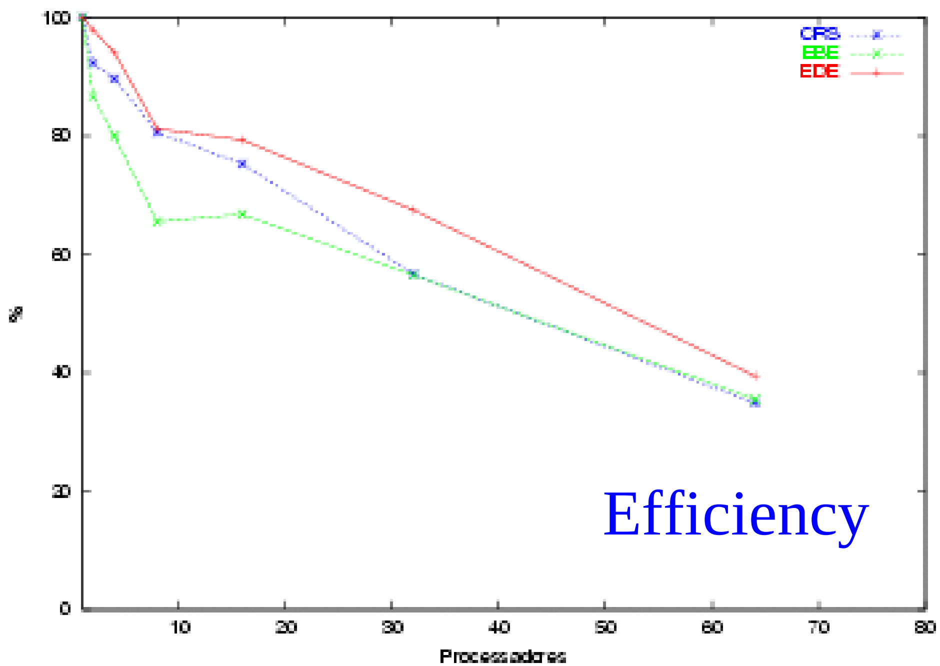
- Advection of a cosine hill in a rotating flow field

Uniform mesh – 263,169 nodes and 524,338 elements (512 x 512 cells)

	1 proc	2 proc	4 proc	8 proc	16 proc	32 proc	64 proc
EBE	5020	2897	1569	958	470	278	221
EDE	4922	2514	1308	758	388	228	195
CSR	3191	1729	890	495	265	176	143

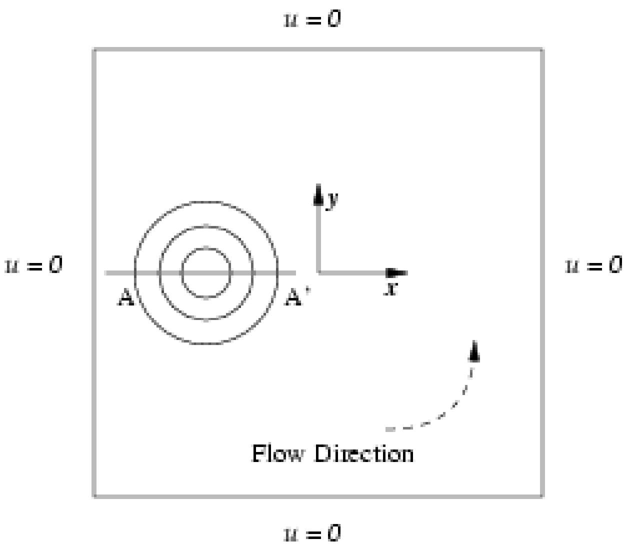
CPU Time

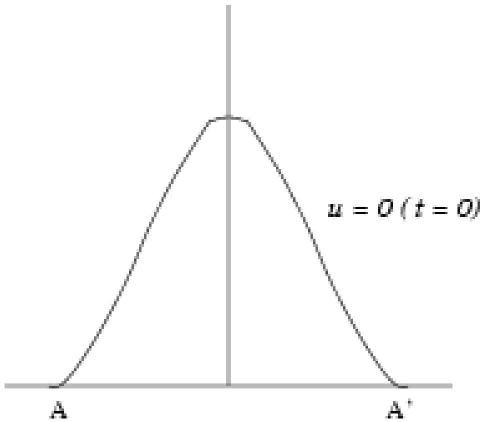




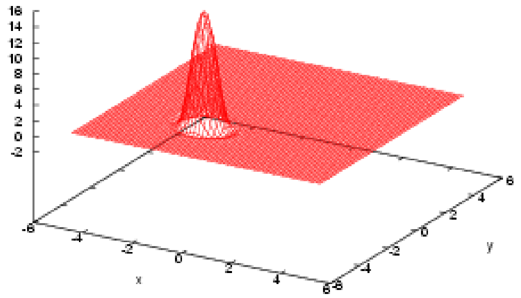
Performance Results

- The rotating cone problem

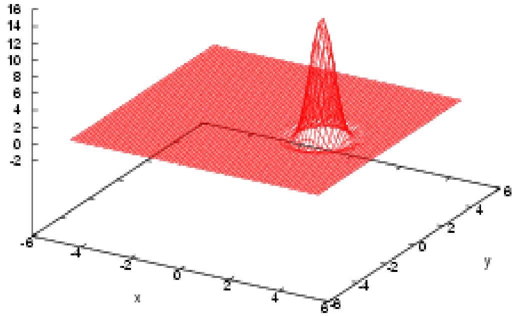




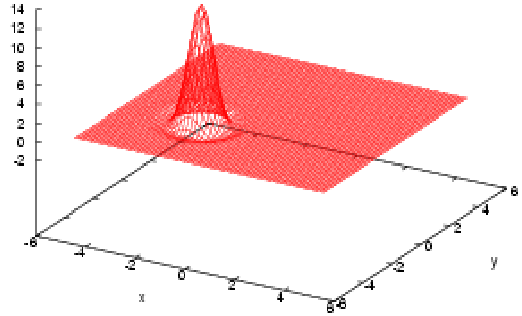
- $\beta = [-y,x]^T$
- $\kappa_x = \kappa_y = 10^{-6}$
- $tol_{GMRES} = 10^{-6}$



$t = 1\ sec$



$t = 3\ sec$



$t = 7\ sec$

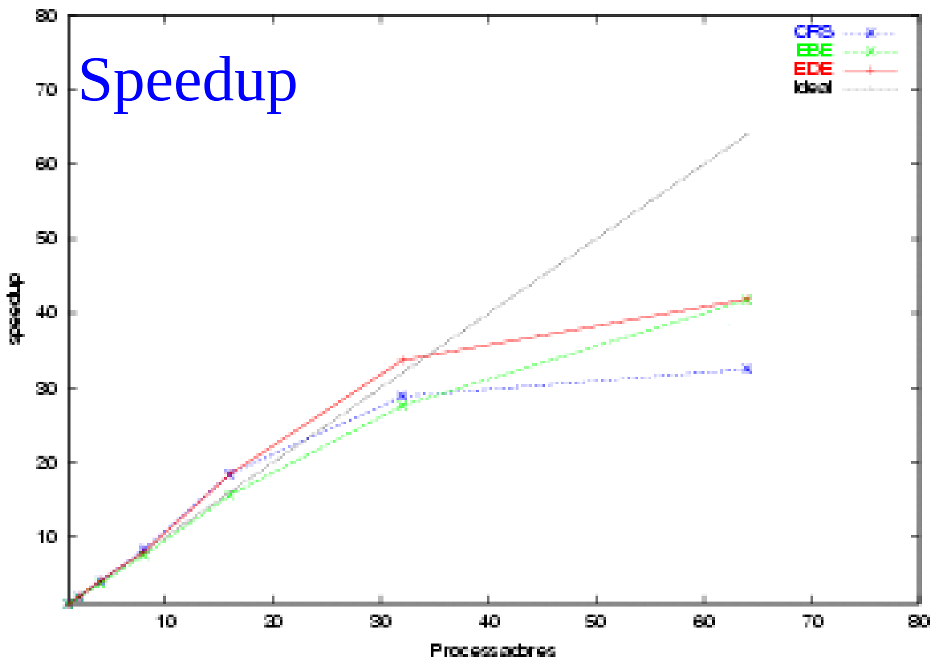
Performance Results

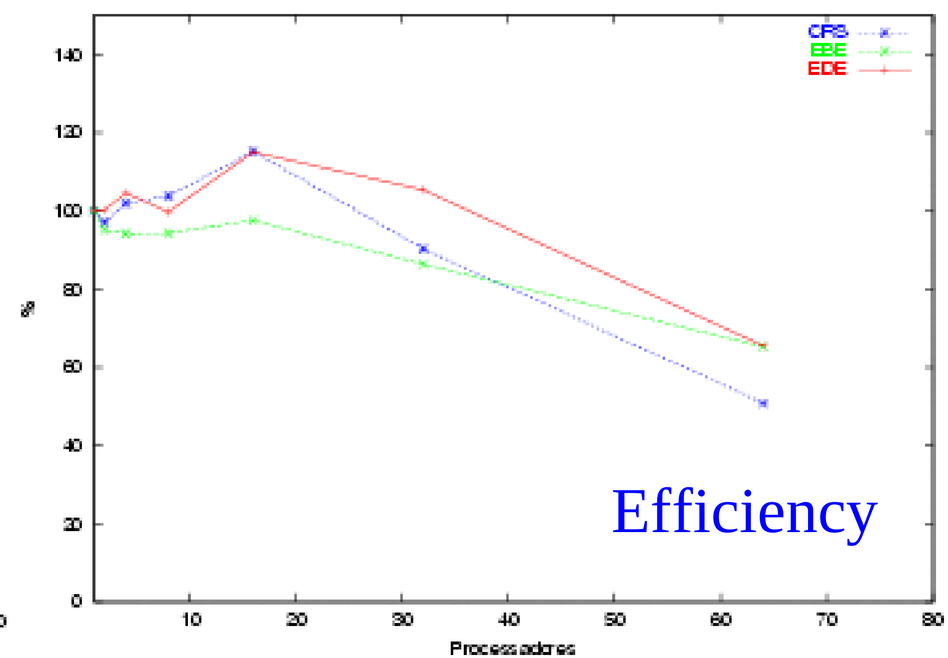
- The rotating cone problem

Non uniform mesh – 259,620 nodes and 517,242 elements

CPU Time

	1 proc	2 proc	4 proc	8 proc	16 proc	32 proc	64 proc
EBE	17044	8964	4525	2259	1092	617	408
EDE	14997	7485	3588	1880	815	445	358
CSR	10080	5193	2473	1214	547	349	310





Conclusion and Future Work

- In our parallel implementation, the approximate solutions with one processor and more than one processor are the same
- The CSR strategy presents the smallest CPU time for all number of processors
- The EDE strategy exhibits the best marks of parallel efficiency
- The EBE strategy has the simplest computational implementation
- Future works include parallel implementation of more complex system of equations and the extension to 3D problems