PARALLELIZZAZIONE SU SCHEDA GRAFICA DI UN ALGORITMO VELOCE "ACAPLUS" PER L' ASSEMBLAGGIO DI MATRICI PIENE IN PROBLEMI ALLE CORRENTI PARASSITE

Flavio CALVANO[1], Guglielmo RUBINACCI[1], Antonello TAMBURRINO[2], George-Marian VASILESCU[3], Salvatore VENTRE[2]

[1] Ass. EURATOM/ENEA/CREATE,DAEIMI, DIEL, Università degli Studi di Napoli Federico II
[2] Ass. EURATOM/ENEA/CREATE, DAEIMI, Università degli Studi di Cassino
[3]Dep. of Electrical Engineering, Politehnica University of Bucharest, Romania

Le moderne tecniche di parallelizzazione di codici numerici prevedono il sempre più crescente ricorso ad architetture di calcolo ibride con combinazioni di processori multicore e schede grafiche. La possibilità infatti di disporre su scheda grafica tramite implementazione CUDA di un elevato numero di processori a basso costo è un aspetto molto importante per calcolo numerico su problemi di dimensione elevata. In questo lavoro è presentata l'implementazione e la parallelizzazione su scheda grafica di un algoritmo veloce ACAPLUS [1] per l'assemblaggio della matrice piena $\underline{\underline{L}}$ che caratterizza la soluzione numerica di problemi alle correnti parassite con la formulazione integrale Cariddi [2]:

$$L_{ij} = \frac{\mu_0}{4\pi} \int_{V_c} \int_{V_c} \frac{\nabla \times \mathbf{N}_i(\mathbf{r}) \cdot \nabla \times \mathbf{N}_j(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} d\mathbf{r} d\mathbf{r}' \tag{1}$$

dove con $\mathbf{N}_k$ si indica un il k-esimo edge element con cui è rappresentata la densità di corrente $\mathbf{J} = \sum_k I_k \nabla \times \mathbf{N}_k$.

Gli elementi fondamentali del metodo di compressione della matrice $\underline{\underline{L}}$ sono:

- ✓ Introduzione di una griglia multilivello che include tutta la mesh agli elementi finiti della struttura [3];
- ✓ decomposizione della matrice in termini dovuti alle interazioni vicine e alle interazioni lontane;
- ✓ calcolo e compressione dei contributi della matrice associata alle interazioni lontane mediate algoritmo ACAPLUS [1];
- ✓ calcolo esatto dei contributi dovuti alle interazioni vicine.

Il primo passo consiste l'introduzione di una griglia multilivello che contenga la mesh. In ogni box della griglia ci sono gruppi di elementi che dovranno interagire. Sia $\underline{\underline{L}}^{ib_1,ib_2}$ la matrice di interazione locale tra due box ib_1 e ib_2. Siano m_e and m (n_e and n) il numero degli elementi e delle incognite in ib_1 (ib_2), rispettivamente. La matrice $\underline{\underline{L}}^{ib_1,ib_2}$ si può interpretare come l'interazione degli n punti sorgenti (presenti nella box sorgente ib_2) sugli m punti campo (presenti nella box campo ib_1). Quindi tutta la matrice può essere vista come sovrapposizione di interazioni locali tra box:

$$\underline{\underline{L}} = \sum_{ib_1,ib_2} \underline{\underline{L}}^{ib_1,ib_2} \tag{2}$$

La memoria richiesta e il tempo di calcolo del prodotto locale sono entrambi pari a m×n. Il secondo passo di decomposizione della matrice emerge dal fatto se le due box sono sufficientemente lontane la matrice $\underline{\underline{L}}^{ib_1,ib_2}$ è a basso rango. Ciò consente di fattorizzare la matrice in questione in termini di prodotto di due matrici a basso rango r ($\underline{\underline{L}}^{ib_1,ib_2} = \underline{\underline{A}} \times \underline{\underline{B}}$, con $\underline{\underline{A}}$ di dimensione m×r e $\underline{\underline{B}}$ di dimensione r×n) (vedi [3]). Se r<<m,n, come accade quando i blocchi sono ben separati, il numero di operazioni per l'assemblaggio cresce come (m+n)·r anziché m×n.

Per le box che non sono sufficientemente lontane (interazioni vicine), la compressione è inefficiente e pertanto la matrice di interazione locale è calcolata esattamente (vedi Figura 1). Pertanto la matrice $\underline{\underline{L}}$ viene decomposta nella parte vicina $\underline{\underline{L}}_{near}$ (calcolata esattamente) e quella lontana $\underline{\underline{L}}_{far}$ (approssimata e compressa) [3]:

$$\underline{\underline{L}} = \underline{\underline{L}}_{near} + \underline{\underline{L}}_{far} \tag{3}$$

Il contributo della parte lontana viene calcolato in questo lavoro tramite l'algoritmo ACAPLUS [1]. Questo algoritmo è particolarmente vantaggioso in termini di occupazione di memoria perché non necessita il preassemblaggio della matrice completa ma opera solo sulle parti considerate rilevanti [1], risultando particolarmente adatto per problemi a larga scala con un elevato numero di incognite.

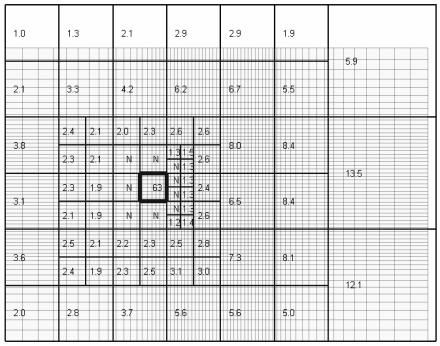

Figura 1. Suddivisione in box della mesh a elementi finite. La box 63 viene preso come sorgente , le box indicate con N sono quelle vicine (Near) mentre nelle altre è indicato il guadagno di compressione tra le altre box e la 63 prendendo ciascuna box come sorgente e la box 63 come box lontana.

Il guadagno di compressione calcolato come $G_{acaplus}=(m+n)\cdot r /(m\cdot n)$ su un caso test di 10^5 incognite come si vede in Figura 2(sinistra) cresce con il numero di incognite e risulta essere complessivamente pari a 17.89. In questo lavoro l'algoritmo è stato anche parallelizzato su scheda grafica tramite implementazione CUDA. I guadagni sul tempo di assemblaggio calcolati come semplici rapporti tra i tempi di calcolo sono anche'essi crescenti con la dimensione del singolo blocco come si vede in Figura 2(destra), per un guadagno complessivo di 29.11.

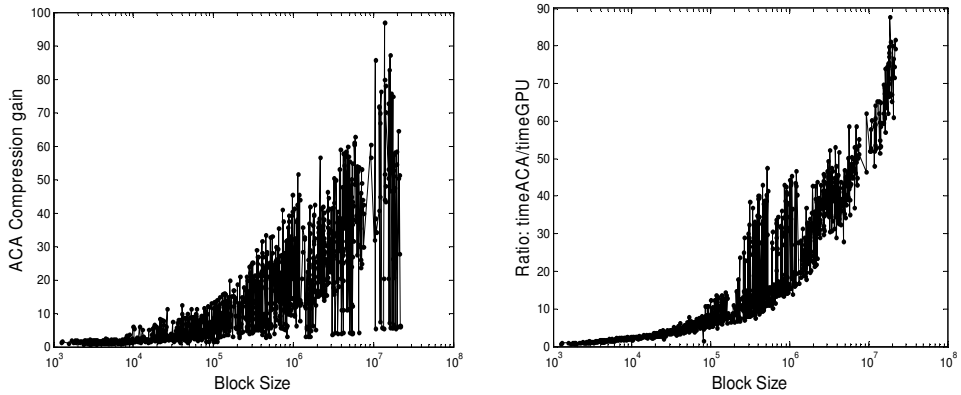

Figura 2. Guadagno di compressione dell'algoritmo ACAPLUS al variare della dimensione del blocco (a sinistra), guadagno sul tempo di assemblaggio della matrice $\underline{\underline{L}}$ tra implementazione seriale e quella CUDA (a destra).

Bibliografia

[1] S. Börm, L. Grasedyck, and W. Hackbusch, Hierarchical Matrices, Springer 2006.

[2] R.Albanese, G. Rubinacci, Integral formulation for 3D eddy-current computation using edge elements, vol 135 n 7, pp. 457 - 462.

[3] G. Rubinacci, S. Ventre, F. Villone, Y. Liu, "A fast technique applied to the analysis of Resistive Wall Modes with 3D conducting structures", Journal of Comp.l Physics, 2009.