1J-01

Hierarchical Diagonal Blocking を用いた疎行列ベクトル積の特性評価

花上直樹　佐々木信一　菱沼利彰　藤井昭宏　田中輝雄
工学院大学

1　はじめに

大規模数値シミュレーションの核となる反復解法は計算時間の多くを疎行列ベクトル積（SpMV）が占める．疎行列の格納形式の一つに圧縮行格納形式（CRS）[1]がある．CRS の問題点として，ベクトル部への参照が非連続のため，キャッシュミスが発生し性能が低下することが知られている．

我々は，グラフ分割アルゴリズム[2]を利用して疎行列の要素を並び替えブロック化し，キャッシュヒット率を改善する疎行列の格納形式，Hierarchical Diagonal Blocking（HDB）[3]に着目した．HDB は，CRS と比べ高速だと言われている[3]，しかしながら，問題ごとの最適なブロック分割数を選ぶ基準は明らかになっていない．

本研究では，疎行列の構造やサイズの違いに着目し，HDB を用いたときの疎行列の分割数を調節することにより，CRS に対するキャッシュヒット率の向上や性能への影響を評価した．

2　CRS を用いた SpMV

CRS は行列 A を非零要素の値が入る各行の開始位置を示す値を格納した配列，非零要素の列番号を格納した配列，非零要素の値を格納した配列，3つの配列で格納する．

CRS では，ベクトル $\boldsymbol{x}$ の参照に非零要素の列番号の配列を使用するため，ランダムアクセスとなり，キャッシュヒット率が悪くなる問題点がある．

3　HDB を用いた SpMV

HDB は疎行列の非零要素が対角に集まるようにグラフ分割を用いてリオーダリングを行い，ブロック化する．ブロックを木構造で格納することで，キャッシュヒット率を向上させる．図 1 に幅 w = 2，深さ d = 1 の木構造で構成される HDB の例を示す．

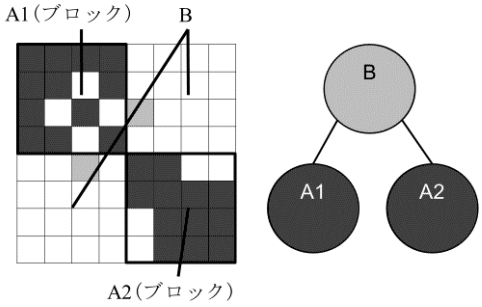

図 1 HDB の例（幅 w = 2，深さ d = 1）

行列サイズを N としたとき，HDB は， w 個の対角ブロック （図 1 の A1，A2）を木構造の子として格納し，余りの部分（図 1 の B）を親として一つの疎行列に格納する．

今回はグラフ分割に METIS[2]というライブラリを使用した．このとき，HDB の生成手順は以下のようになる．

(1)　METIS を用いて行列の行を w 個に分割
(2)　(1)を元に行列の行，列，ベクトルをソート
(3)　(2)を図 1 のように対角ブロックに分割
(4)　(1)～(3)を再帰的に d 回行う

行，列，ベクトルをソートすることで行列の非零要素を対角に集約し，ベクトル $\boldsymbol{x}$ の参照を連続にすることができる．

このとき，分割数を増やすとブロックサイズを小さくし，データがキャッシュに収まりやすくなる．しかし，ブロック全体の面積が小さくなるため， 非零要素がブロック外に出てしまう可能性が高くなる．そのため，問題によって適切な分割数に変える必要があると考えられる．

4　数値実験

4.1　実験環境

実験には Intel Core i7-3770K 3.5GHz 4core 32GB を用いた．L3 キャッシュは 8MB，OS は CentOS 6.3，コンパイラは gcc 4.4.7，オプションは“–O3 –fopenmp”である．8 スレッドで並列化した．

実験に使用する疎行列を表 1 に示す．これは，The Univ. Florida Sparse Matrix Collection [4]のサイズ，構造の異なる 5 問題である．

An Evaluation of Hierarchical Diagonal Blocking for Sparse Matrix and Vector Product
Naoki Hanaue, Shin'ichi Sasaki, Toshiaki Hishinuma, Akihiro Fujii and Teruo Tanaka
Kogakuin University

4.2 実験結果・考察

HDB 形式における最適な幅を調べるために，対象の問題に対し，深さ d = 1，幅 w = 2 から 16 まで試行した．なお幅 17 以降は大きな変化が見られなかった．

各問題における CRS の性能，最適な w のときの HDB の性能，最適な w，CRS と HDB の性能比を表 2 に示す．この結果から，ブロック化することでキャッシュヒット率が向上したと考えられる．

次に，性能向上率の高かった#3“c-56”と#4“Dubcova2”の性能変動を図 2 に示す．これより w を増やすことによる効果は問題によって異なることがわかる．

部分的に w を大きくしたことで性能が低下した点がある．理由は，対角ブロックに集められなかったブロック外の要素数（図 1 の B）が増えたためである．このことより、対角ブロックのキャッシュヒット率向上による性能向上と行列全体に対する対角ブロックの割合のトレードオフを考慮する必要がある．

HDB 化により性能が向上した#4“Dubcova2”の構造を 図 3，劣化した#1“fv2”の構造を図 4 に示す．傾向として図 3 のように元の疎行列の非零要素が非対角に散らばっている問題は HDB の効果が高かった．これに対し，図 4 のように疎行列の非零要素が対角に集まっている問題は CRS で計算した時点でも速く HDB によるリオーダリングをしても効果が低く性能向上率が低くなり，非対角要素が増えることで遅くなった．

表 1 実験に用いた疎行列

#	問題名	N	*nnz	*nnz/row
1	fv2	9,801	48,413	4.94
2	bcsstk15	3,948	60,882	15.42
3	c-56	35,910	208,405	5.80
4	Dubcova2	65,025	547,625	8.42
5	bcsstk39	46,772	1,068,033	22.83

(*nnz = number of non zeros)

表 2 HDB 形式の性能と最適なブロックサイズ

#	CRS 性能 [GFLOPS]	HDB 性能 [GFLOPS]	最適な幅	HDB /CRS
1	5.7	4.4	3	0.8
2	5.5	6.3	7	1.1
3	2.2	5.1	6	2.4
4	2.7	5.4	7	2.0
5	4.2	4.5	13	1.1

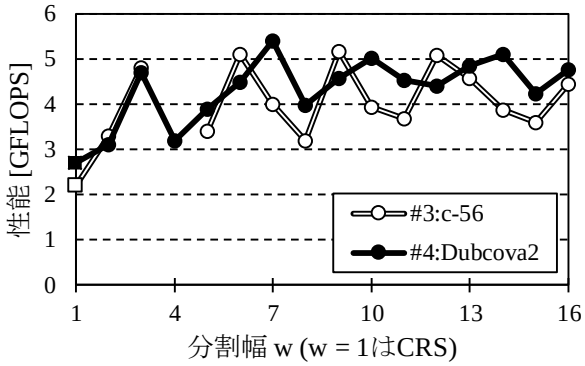

図 2 分割幅 w の増加に伴う性能変動

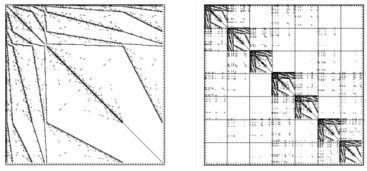

図 3 #4 非零分布 (左：非分割，右：7 分割)

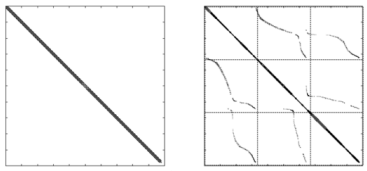

図 4 #1 非零分布 (左：非分割，右：3 分割)

5 まとめ

CRS と HDB を比較し，問題に応じて異なる HDB 化の効果と最適な分割数について確認した．

疎行列の構造的に非零要素が対角に近い位置に分布している問題よりも，散っている問題の方が HDB の効果が高かった．最も効果が得られたものでは，CRS に対して HDB は 2.4 倍の性能向上が見られた．これは非対角上にあった要素が，対角ブロックに集約しキャッシュヒット率が向上したためだと考えられる．なお，効果がある問題でも分割数によって向上率が変わるため，HDB は分割数を適切な値にする必要がある．

今後の課題として，HDB の深さの評価と分割数を自動チューニングすることが挙げられる．

参考文献

[1] Barrett, R., et al. , Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, SIAM pp. 57–65 (1994).

[2] METIS - Serial Graph Partitioning and Fill-reducing Matrix Ordering, http://glaros.dtc.umn.edu/gkhome/metis/metis/overview.

[3] Guy E. Blelloch, et al. , Hierarchical Diagonal Blocking and Precision Applied to Combinatorial Multigrid Super Computing 2010 (2010).

[4] The University of Florida Sparse Matrix Collection, http://www.cise.ufl.edu/research/sparse/matrices/.