計算機アーキテクチャ 142−15
ハイパフォーマンス 85−15
コンピューティング
（2001. 3. 8）

キャッシュラインを考慮したコーナーターン法の改善

和泉秀幸[†] 佐々木和司[††] 中島克人[†]

SMP(Symmteric Multi-processor) 上でマルチスレッドプログラムによるSAR(Synthetic Aperture Radar：合成開口レーダ) 画像再生処理の並列化を進めている．この1部分処理であるコーナーターンを対象に，キャッシュミスを低減するコーナーターン方法を提案したが，画像サイズによってはキャッシュライン衝突が頻発するという問題があった．

今回我々は，キャッシュライン衝突の軽減方法を新たに提案し，評価の結果，コーナーターンの8プロセッサによる並列処理において約20%性能改善できることを確認した．また，この性能改善が，SAR画像再生処理全体で約10%の性能改善に寄与することも確認した．

Improving the Corner-turn by considering cache lines

HIDEYUKI IZUMI,[†] KAZUSHI SASAKI[††] and KATSUTO NAKAJIMA[†]

Parallel processing of SAR (Synthetic Aperture Radar) image reconstruction on symmetric multi-processors based on multi-thread programming is one of our research subjects. "Corner-turn" is a subprocess of SAR image reconstruction, and we have proposed a technique for parallelizing it in order to reduce cache miss. However, the technique has a defect of cache line collision according to the size of SAR data.

In this paper, we proposed an improved parallelizing technique to reduce the cache line collisions. The evaluation results shows about 20% speed-up in the parallel "Corner-turn" on eight processors, which contributes to the total performance improvement of parallel SAR image reconstruction by about 10%.

1. はじめに

SAR(Synthetic Aperture Radar：合成開口レーダ)は，日中，夜間，雲霧等の天候を問わずに，高い分解能で地表を撮像できるセンサである[1)2)3)]．ただし，人間が理解可能な画像を生成する処理 (SAR 画像再生処理) が必要になる．この処理は，画素あたりの演算量が多く，かつ画像のサイズが大きい．このため，我々は，SMP(Symmteric Multi-processor) でマルチスレッドプログラムによる処理の並列化 (高速化) を進めている．

SAR 画像再生処理は，FFT，参照関数の乗算，IFFT といった一連の処理でデータを次々に加工していく (図1参照)．ここでは，部分処理やサブルーチン単位での並列化戦略を取りずらい（効果が小さい）．一方，データサイズが大きく，データ間の依存関係も小さいため，各部分処理単位（FFT や IFFT）でデータ並列による並列化を行えば良いことがわかった．

ただし，SAR 画像再生処理の部分処理の1つであるコーナーターンは，メモリアクセスが中心の処理であり，単純に並列化するとキャッシュミスを起こしやすい．一般に，SMP 上の並列化では，キャッシュミスを減らし，キャッシュのヒット率を向上させることが，性能向上で重要になる[4)]．

そこで我々は，“キャッシュのラインにちょうど納まる幅の画素数を1辺とする正方形” を画像ブロックとして定義し，この単位の処理をプロセッサに分割して並列実行する方法を提案している[5)]．

この効果を実機上で評価し，SMP 計算機である8プロセッサ構成の SGI ORIGIN 2000 で評価し，単純な逐次実行に比べてコーナーターン処理を約 20 倍高速化するなど，画像ブロック単位のコーナーターン方法が，処理高速化と台数効果の改善に有効であることを確認した[5)]．

一方で，画像ブロック単位のコーナーターン方法では，画像サイズによってはキャッシュラインの衝突が起こり，キャッシュのヒット率が低下するケースもあり，[5)] での評価 では，この衝突の影響を考慮できていなかった．

本稿では，画像ブロック単位のコーナーターン方法でキャッシュライン衝突を軽減する方法を新たに提案し，その効果を評価する．また，コーナーターンでの改善が，SAR 画像再生処理全体に与える効果も評価する．

なお，本稿で評価に用いる SGI ORIGIN 2000 は，cc-NUMA(cache coherent non-uniform memory access) アーキテクチャであるが，アプリケーションからは SMP と同様のプログラミングモデルが採用でき，並列計算機としての性能特性もほぼ SMP と見なせるものである．

† 三菱電機（株）情報技術総合研究所
Information Technology R & D Center, Mitsubishi Electric Corporation
†† 三菱電機（株）鎌倉製作所
Kamakura Works, Mitsubishi Electric Corporation

本稿では，まず，2章でSAR画像再生処理とコーナーターン部分処理とを簡単に説明する．次に，3章で画像ブロック単位での並列処理法とキャッシュライン衝突が発生する問題を示し，この改善法を説明する．4章では，SGI ORIGIN上で実施した性能評価について述べ，最後に5章でまとめる．

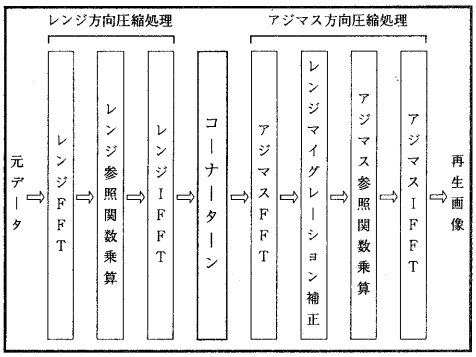

図1　SAR画像再生処理の流れ

2. SAR画像再生処理とコーナーターン

SAR画像再生処理は，マイクロ波を使ったセンサで収集したデータから，人間が理解可能な画像を生成する処理である．ここでは，センサがマイクロ波を照射する方向をレンジ方向，センサを搭載したプラットホームの進行方向をアジマス方向とする．

SAR画像再生処理は，レンジ方向圧縮処理と，コーナーターンと，アジマス方向圧縮に分けることができる(図1参照)．このうち，レンジ方向とアジマス方向の圧縮処理では，各方向でFFTやIFFTといった信号処理を行う[1)2)3)]．この圧縮処理では，図2のように処理前と処理結果の画像の領域を確保し，レンジまたはアジマス方向で画像を分割して並列実行することで，効率良く高速化を行える見込みを得ている[6)]．

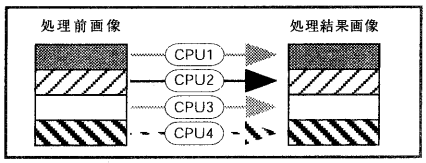

図2　レンジ方向とアジマス方向の圧縮処理の並列化

コーナーターンは，各方向での処理を効率良く行うために，画像のメモリ上の配置を変更する処理である．図3はレンジからアジマス方向へのコーナーターンの例である．

コーナーターンでは，処理前の画像を基準にして単純な並列化を行うと，書き込み側である処理後の画像側でキャッシュのライトミスを起こしやすい(図3参照)．逆に，コーナーターン処理後の画像を基準にすると，読み込み側でキャッシュのリードミスを起こしやすい．

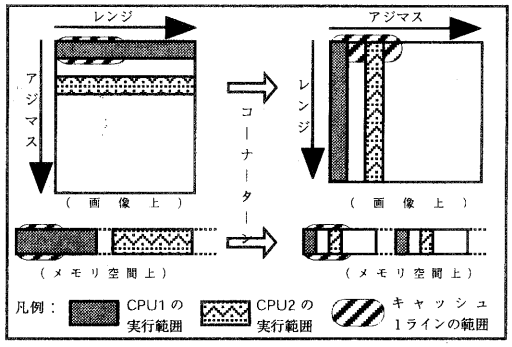

図3　コーナーターンとその問題点

3. 画像ブロック単位での並列処理法

コーナーターンでのキャッシュミスを低減するために，我々は，キャッシュのラインにちょうど納まる幅の画素数を1辺とする正方形を“画像ブロック”として定義し，この単位で処理をプロセッサに分割してコーナーターンを並列実行する方法を考えた(図4参照)．

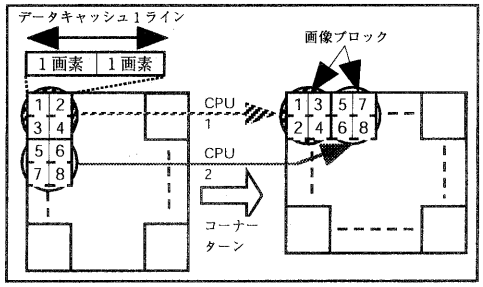

図4　画像ブロック単位での並列処理法

階層型のキャッシュの場合も，この方法では次の手順で画像ブロックを決定する．まず，プロセッサに最も近い最上位のキャッシュで画像ブロックのサイズを決定する．その後，下位のキャッシュで，ラインにちょうど納まる幅の“1つ上位のキャッシュの画像ブロック”数を1辺とする正方形を画像ブロックとして順次決定する．

下位の画像ブロックのコーナーターンは，上位の画像ブロックのコーナーターンを何回か繰り返す形式で実行する．また，画像ブロック内とブロック間のアクセス順序は，キャッシュの総容量（ライン数）から別途決定する．

3.1 対象システムでの画像ブロック

今回の評価で使用するシステムと，これに対する具体的な画像ブロックのサイズを説明する．

- 処理対象画像

  1画素はfloat型(4byte)×2（実数部と虚数部）で8byte.

- 計算機
  SGI ORIGIN 2000. プロセッサは MIPS R10000(250MHz) × 8 個. 1 ボードに 2 プロセッサ搭載され, ccNUMA アーキテクチャを採用 (図 6 参照). MIPS R10000(250MHz) でのキャッシュは表 1 に示す値 (1 次のインストラクションキャッシュは省略). なお, 1 次 2 次キャッシュとも write-back, 2way-set associative, LRU(Least-Recently Used).
- OS
  IRIX 6.5.

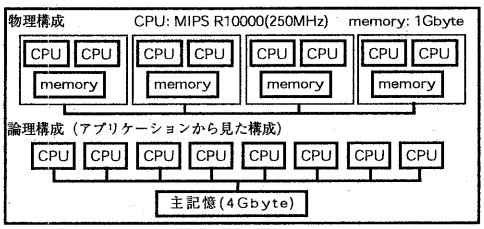

図 5 SGI ORIGIN 2000

表 1 MIPS R10000(250MHz) のキャッシュ

	Line size	Line 数	Total size
1st Cache (Data)	32byte	1,024	32Kbyte
2nd Cache	128byte	37,768	4Mbyte

図 6 は, SGI ORIGIN 2000 での画像ブロックである. ここでは, 1 次キャッシュのラインに 4 画素分のデータが納まり, 2 次キャッシュの 1 ラインは 1 次キャッシュ 4 ライン分のサイズになる. 本稿では, 1 次キャッシュに対する画像ブロックを "1 次画像ブロック", 2 次キャッシュにに対するそれを "2 次画像ブロック" と称する. 2 次画像ブロックでのコーナーターンは, 1 次画像ブロックの移動を 1 6 回繰り返す形式で実行する.

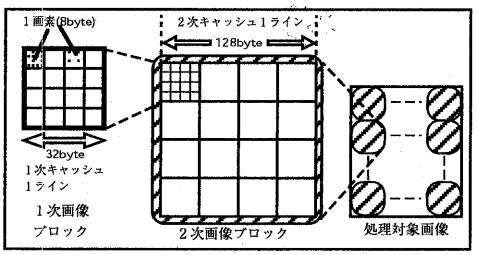

図 6 SGI ORIGIN での画像ブロック

これまでの提案方法では, 画像ブロック内とブロック間のアクセス順序を, キャッシュの総容量 (ライン数) から決めていく. 画像ブロック単位でのコーナーターンを実行するのに十分なライン数がある場合には, 任意のアクセス順序を選択できる. ライン数が足りない場合は, キャッシュの追い出しアルゴリズムを考慮して, 追い出しによるペナルティが最小になるアクセス順序を選択する.

今回の対象システムでは, 十分なライン数があるので, 任意のアクセス順序を選択できる. このため, いくつかのアクセスパターンについて実行時間を計測し, 最終的なアクセス順序を決定する.

3.2 画像ブロック単位での並列処理法の問題点

今回の計測で使用する MIPS R10000 は 1 次 2 次キャッシュとも 2way set associative 方式であるため, 1 次キャッシュでは 16Kbyte 毎, 2 次キャッシュでは 2Mbyte 毎に同じキャッシュラインのペアを利用することになる.

前回の評価で使用した画像領域全体のサイズは, 1 画素 8byte で 8,192 × 8,192 画素なので, 1 行分のデータサイズが 64Kbyte になる. よって, 1 次キャッシュでは各行の同じ列の要素が, 2 次キャッシュでは 3 2 行毎の同じ列の要素が, 同じキャッシュラインのペアを利用することになる (図 7 参照).

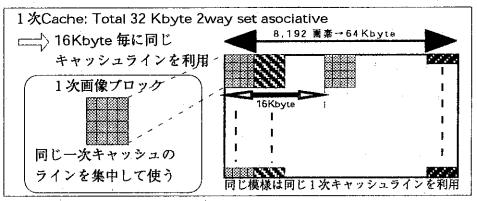

図 7 1 次キャッシュラインで衝突が起こる例

画像ブロック単位のコーナーターンでは, コーナーターン実行時に異なったキャッシュラインを利用することを前提にしている. しかし, 上記のようなケースでは, 同じキャッシュラインの衝突が起り, キャッシュのヒット率が低下する問題があった.

この問題に対して, 我々は, 画像ブロック単位のコーナーターンに次の改善を施した. まず, キャッシュの情報を基準に処理対象の画像領域のサイズでキャッシュラインの衝突が起こるか否かを判定する. ここで, 衝突が起きる場合は, 画像ブロック単位のコーナーターン実行時に衝突が起きないように, 画像領域の端 (図 8 参照) に一定の領域 (以後, オフセットとする) を追加する. このオフセット部分は, コーナーターンを行わない. また, オフセットのサイズは, 性能評価結果から決定する.

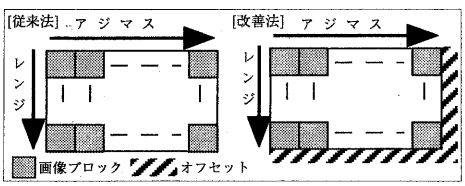

図 8 コーナーターンの改善方法

この改善方法を選択した理由は, 1. 問題の発生が処理対象の画像領域のサイズに依存するため, 問題が起きない

ケースでは従来の画像ブロック単位のコーナーターンをそのまま利用したいこと．次に，2.コーナーターンはSAR画像再生処理の1部分であるため，他の画像処理プログラムへの影響を小さくしたいこと．また，3.実現方法が容易であり，かつ，従来の画像ブロック単位のコーナーターン法からの変更が小さいことである．

4. 性能評価

コーナーターン処理で，処理対象の画像領域に追加するオフセットのサイズを変えて，1.各プロセッサに割り当てる画像領域の分割サイズ別と，2.分割した画像領域内でのアクセス方法別に実行時間を比較する．次に，3.画像ブロック単位でのコーナーターン法での最適なオフセットのサイズを評価し，また，4.コーナーターンの改善がSAR画像再生処理全体に与える効果も評価する．

なお，コーナーターン実行時間計測では，次の条件は同じである．

- コーナーターン条件
  レンジ方向からアジマス方向へコーナーターンする（処理前画像ではレンジ方向で連続アドレス，処理結果画像ではアジマス方向で連続アドレスになる）．画像サイズは8,192 × 8,192 画素．
- スレッド数
  1～8．だだし，スレッド数1のときは，逐次実行し，並列化のオーバーヘッドを含まない．
- スケジューリング
  block スケジューリングでデータを分割．
- 計測環境
  システムのPthreadライブラリ上に実現した，ループ並列化用の簡易スレッドライブラリSPL(Simple Parallel Library)[8]を利用．SPLでは，実行開始時に指定のスレッド数を確保し，処理を並列実行した上で，バリア同期で待ち合わせる．計算機構成は3.1節を参照．

4.1 分割サイズの評価

まず，各プロセッサに分割して並列実行する画像領域の分割単位を変更してコーナーターンの実行時間を計測する．本稿ではN × N画素の固まりをブロックサイズとし，この単位で計測する（ブロックサイズ4とは，4 × 4画素）．次のパラメータで計測を行った．なお，ブロック内は画素単位でアクセスし，ブロックごとではアジマス方向へのアクセスとした．

- ブロックサイズ
  1～1024まで（2のN乗の値）．
- オフセットのサイズ
  0画素(オフセットなし)，4画素，8画素，16画素，32画素，64画素．

4.1.1 計測結果と考察

8スレッドのケースでブロックサイズとオフセットの大きさを変えた時の実行性能を図9に示す．ここでは，単純な逐次実行(1スレッド，ブロックサイズ1，オフセット0画素）の性能を1とした時の実行性能比を高速化率（値が大きいほど性能が良い）として示す．なお，計測結果は5回実行した時の最短時間である．なお，この評価での実行性能は，他のスレッド数でも同様の傾向であった．

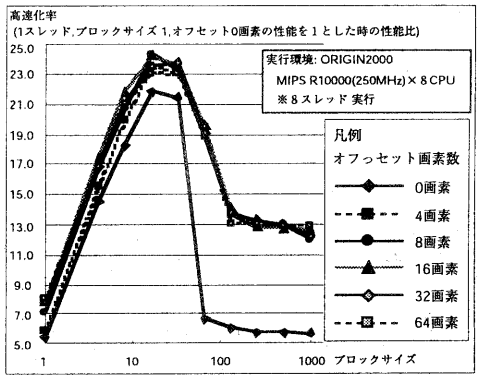

図9　8スレッド実行の結果

ブロックサイズとオフセットの大きさについて考察する．まず，ブロックサイズについては，全てのオフセットの大きさで，サイズが1から増加するごとに性能が向上し，2次キャッシュのラインサイズである16をピークに，32までは高い性能を示すが，64からは徐々に性能が低下する．このことから，今回の提案法である画像ブロック単位のコーナーターンと同じブロックサイズ16で，最大の性能向上を得られることを確認できた．

次に，オフセットの大きさについては，オフセットを4画素，8画素，16画素と追加していくことで，全てのブロックサイズで実行性能が改善し，オフセットの追加が性能改善に有効であることが確認できた．これは，全てのブロックサイズで1次キャッシュラインの衝突が，軽減されたためと考える．

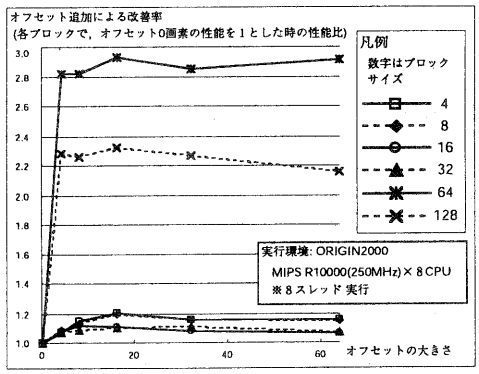

図10　オフセット0画素に対する性能改善率

オフセットの効果をより詳細に検証するため，8スレッド時の実行性能を，各ブロックサイズでのオフセット0画

素に対する性能改善率(オフセット0画素の性能を1とした時の実行性能比)で比較する(図10参照).

図10から,1辺が1次キャッシュラインサイズの正方形の画像領域であるブロックサイズ4とその倍の8では,ブロックサイズ16及び32よりもオフセット追加による性能改善がやや大きい.ブロックサイズ4と8では,1次キャッシュにより集中してアクセスするため,性能改善が処理時間全体に占める割合がブロックサイズ16や32と比べて大きく,改善量が大きかったと判断している.

ブロックサイズ64以上では,オフセット追加による性能改善が特に大きい.これは,オフセット0画素時に著しく性能が低下していたのが改善された結果である.この要因について,ブロックサイズ64を例に考察する.

オフセット0画素の時,ブロックサイズ64での性能低下は特に大きい.この条件では,全てのブロックサイズで1次キャッシュラインの衝突が起きる.このため,ここでの性能低下の原因は,2次キャッシュに起因すると考える.2次キャッシュでは,32行毎にキャッシュラインの衝突が起きる.ブロックサイズ64では,64×64画素単位でアクセスするため,1次キャッシュに加えて2次キャッシュでもキャッシュラインの衝突が起きたために,急に性能が低下したと考える.

例えばオフセットを4画素加えた場合には,2次キャッシュラインが同じ列の画像領域に割り当てられるのは2047行先になり,キャッシュラインの衝突がなくなる.1次キャッシュに比べて,改善効果の大きい2次キャッシュの効率が改善したため,ブロックサイズ64で実行性能の改善量が大きかったと分析している.

ただし,オフセットを加えてもブロックサイズ64の絶対性能はブロックサイズ32よりも明らかに低い.これは,コーナーターンで必要な1次キャッシュのライン数が不足したためと考える.ブロックサイズ32のコーナーターンで必要な1次キャッシュのライン数は512ラインに対し,ブロックサイズ64では2048ラインである.システムリソースは1024ライン(2way set associative:2ラインで1セット×512セット)のため,ブロックサイズ64ではライン数が不足する.このため,使用中の1次キャッシュデータが追い出され,キャッシュのヒット率が低下し,性能も低下したと考える.

4.2 ブロック内アクセス方法の評価

4.1節の評価で,ブロックサイズ16で最も良い性能を得られることを確認できた.次に,2次画像ブロック(ブロックサイズ16)でのアクセス方法を変え,その影響を評価する.

ここでは,2次画像ブロック内を画素単位または1次画像ブロック単位で実行した時の性能を比較する.従来の計測結果から,特にアクセス方向が台数効果に影響を与えることが確認されていたので[5],次のパラメータで計測を行った.

- オフセットのサイズ
  0画素(オフセットなし),8画素
- アクセス単位
  画素または1次画像ブロック単位.
- アクセス方向
  レンジまたはアジマス.アクセス単位及び2次画像ブロック毎に,同じ方向に処理を進める.

図11に,逐次実行(1スレッド,画素・アジマス・0画素)の性能を1とした時の実行性能比で計測結果を示す.この計測結果から,2次画像ブロック内のアクセス単位では,1次画像ブロック単位で実行した方が画素単位よりも実行性能が良いことを確認した.1次・アジマス・8画素で8スレッドでの最大性能時には,単純な逐次実行に比べて25倍以上の高速化できた.また,オフセットを追加した効果は1次画像ブロック単位の方が画素単位よりも大きい.これは,1次画像ブロック単位では同じ1次キャッシュへ集中してアクセスするので,キャッシュラインの衝突軽減の恩恵が,画素単位よりも大きいためと考える.

次に,アクセス方向では,オフセットを追加することで,台数効果で劣っていたレンジ方向のアクセスでの性能が改善した.ただし,アクセス単位とオフセットの大きさが同じ場合は,アジマス方向の性能が優れていた.

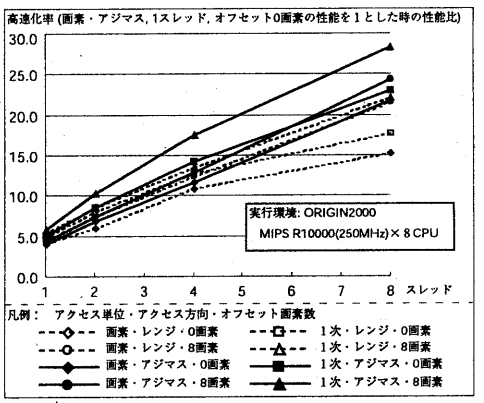

図11 アクセス方法と実行性能

4.3 オフセットのサイズ

前節までの評価で,提案していた画像ブロック単位のコーナーターンが,オフセットを加えた場合でも効率がよいことを確認した.また,処理全体をアジマス方向に進めた方が効率が良いことも確認した.ここでは,画像ブロック単位のコーナーターンをアジマス方向に進め,オフセットとして追加する領域のサイズを変えて評価した.

図12に,スレッド数別のオフセット0画素に対する性能改善率で計測結果を示す.オフセットについては,1次キャッシュラインのサイズである4画素で最適とならず,その倍の8画素以上で良い値となる.

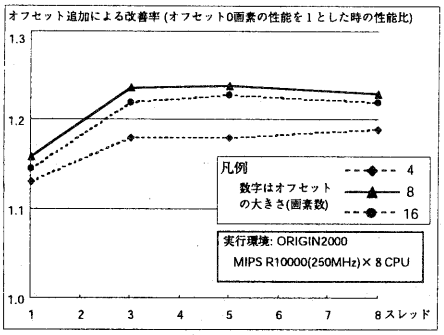

図12　2次画像ブロックでのオフセットサイズの評価

これは，画像領域の先頭とキャッシュラインの先頭が一致していないためと考えている．この場合，一次キャッシュのラインサイズと同じ4画素のオフセットでは，一次画像ブロック単位のコーナーターン実行時に，キャッシュラインの衝突が完全に解消されない．一方で，一次キャッシュのラインサイズの2倍である8画素では，画像領域の先頭とキャッシュラインの先頭が一致しないケースでも，キャッシュラインの衝突は解消される．この差が8画素以上のオフセットで効果的になった理由と考えている．一方で8画素よりもオフセットを増やしても，飛躍的な性能向上は見込めないことも確認できる．このため，画像ブロック単位のコーナーターンで，オフセットとして追加する領域のサイズとしては8画素が適当と判断した．

4.4 SAR画像再生処理への適用

これまでの評価で，性能が最も良いと判断される8画素のオフセットを加えた画像ブロック単位のコーナーターンを，SAR画像再生処理に組み込んだ．この効果を評価するため，次の条件で実行性能を計測した．

- コーナーターン方法
  画素単位，画像ブロック単位：オフセット無，画像ブロック単位：オフセット8画素の3種類
- 画像サイズ
  レンジ2,048 ×アジマス 16,384 画素

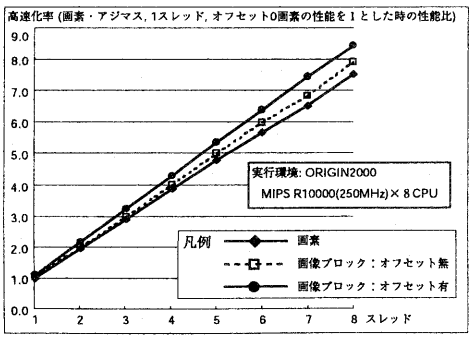

図13　SAR画像再生処理全体の実行時間

計測の結果(図13参照)，オフセットを加えたコーナーターンを採用することで，8スレッド時にSAR画像再生処理全体の性能を約10%強改善できることを確認した．ここでは，全実行時間に占めるコーナーターンの消費時間が，1スレッド時には画素単位で約6%，画像ブロック単位：オフセット8画素で約2%に対し，8スレッド時には画素単位で約15%，画像ブロック単位：オフセット8画素では約2%である．

5. まとめ

画像ブロック単位のコーナーターン方法で，画像サイズによりキャッシュラインの衝突が起こり，キャッシュのヒット率が低下する問題を改善するため，処理対象の画像領域にオフセットを追加する改善を施した．

これにより，従来の画像ブロック単位のコーナーターン方法の性能を20%程度改善し，単純な逐次実行に比べて8プロセッサの環境で25倍以上の高速化を達成できた．また，改善したコーナーターン方法をSAR画像再生処理へ組み込むことで，処理全体での性能も約10%改善できた．

参考文献

1) 奄野 信義: 合成開口レーダ, 日本リモートセンシング学会誌, vol.1, no.1, pp.49-107, (1981).
2) J.C.Curlander, et al: Synthetic Aperture Radar Systems and Signal Processing, Wiley & Sons, inc., (1991).
3) 飯坂 譲二監修 日本写真測量学会編: 合成開口レーダ画像ハンドブック, 朝倉書店,(1998).
4) 田中 貞夫他: COMPaS: Pentium Pro を用いたSMPクラスタとその評価, 並列処理シンポジウムJSPP '98, pp.343-350, (1998).
5) 和泉 秀幸 他: SMPでのSAR画像再生処理の並列化 ～キャッシュを活かしたコーナーターン方法と性能評価～, 情処 HPC 82-30, (2000).
6) 和泉 秀幸 他: SAR画像再生処理への並列プログラミング支援環境の適用検討, 第59回 情処全国大会 5L-4, (1999).
7) 和泉 秀幸 他: SMPでのSAR画像再生処理の並列化 ～キャッシュを活かしたコーナーターンの高速化～, 第60回 情処全国大会 5H-05, (2000).
8) 福地 雄史 他: マルチプロセッサ対応UNIX上での並列プログラム開発支援環境の開発, 第48回情処全国大会, 2G-9, (1994).