ハイパフォーマンス
コンピューティング 82−30
（2000．8．5）

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～

和泉秀幸[†] 佐々木和司[††]
水野政治[†] 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は，キャッシュミスによる性能低下により，効率の良い並列化が困難と予測されていた．
我々は，キャッシュのラインサイズを考慮したコーナーターンの並列実行方法を提案する．提案方法について実機上で評価を行い，8プロセッサの環境で約20倍の高速化を達成した．

Parallizing SAR Image Reconstruction on SMP: Optimizimg and Evaluating the Corner-turn by improving cache access

Hideyuki Izumi,[†] Kazushi Sasaki,[††] Masaji Mizuno[†]
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 has been regarded as a hard part for efficient parallelization because of intensive cache miss.
We proposed an efficient technique for parallelizing "Corner-turn" considering cache line size. The evaluation result of our technique shows about twenty-fold speed-up on eight processors.

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．

我々は，まず，並列化のための解析を行った[5)]．SAR画像再生処理では，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，粗粒度の並列処理，すなわち，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)]．

コーナーターン処理でも，キャッシュのヒット率を改善することで処理を効率化できるものと考える．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，この単位で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．SUN社のWSを使って行った1次評価の結果，提案方法により，単純な並列化と比較して，処理を高速化し，台数効果を改善する効果があることを確認した[6)]．

この方法の効果を解析するため，プラットホーム計算機上で種々の性能評価を進めている．ここでは，分割する画像ブロックのサイズやアクセス方向などを変更して評価し，コーナーターンの性能に与える影響を検証する．

本稿では，プラットホーム計算機として，SGI ORIGIN 2000を用いた場合の評価結果を示す．ORIGIN 2000は，ccNUMA(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るため，我々はSMPでの並列化の一環としてORIGIN 2000での開発を進めている．

† 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章で高速化の対象であるコーナーターン部分処理を簡単に紹介し，並列化での問題点を示す．次に，3章で画像ブロック単位での並列処理法について説明する．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．

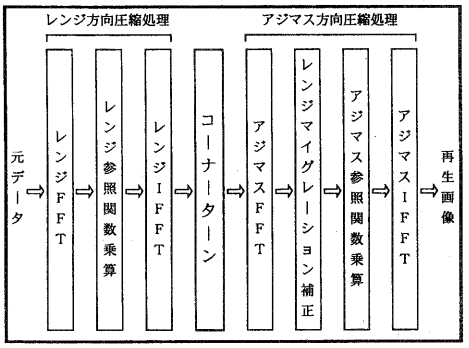

図1　SAR 画像再生処理の流れ

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 (図2参照)．

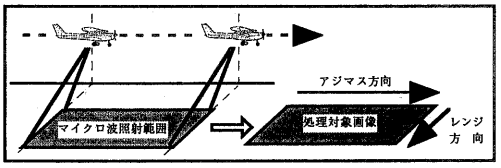

図2　レンジ・アジマス方向

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3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[5)]．

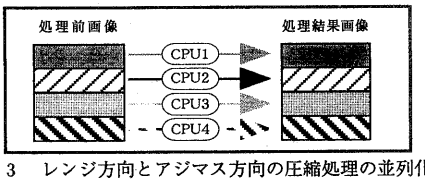

図3　レンジ方向とアジマス方向の圧縮処理の並列化

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，逐次処理の場合，全処理時間の 10 %強を占める．図4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4参照)．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．

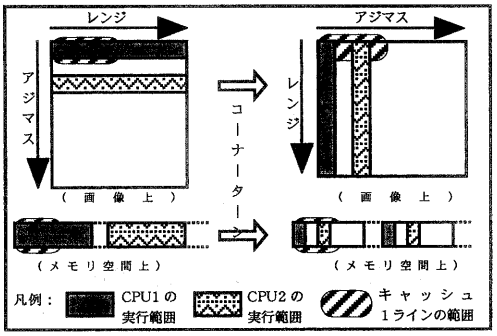

図4　コーナーターンとその問題点

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5参照)．

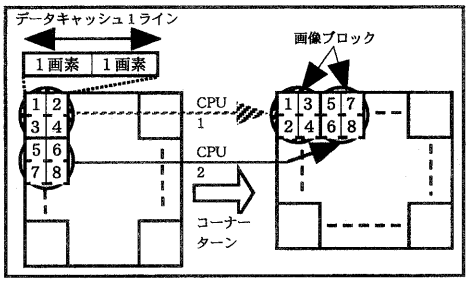

図5　画像ブロック単位での並列処理法

この方法では，階層型のキャッシュの場合，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

この節では，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.

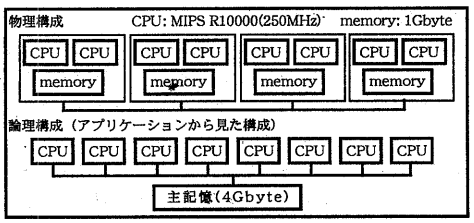

図 6 SGI ORIGIN 2000

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	65,536	4Mbyte

図 7 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.

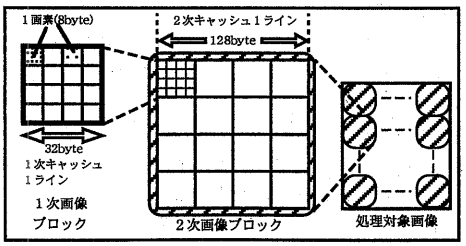

図 7 SGI ORIGIN での画像ブロック

3.2 画像ブロック内とブロック間のアクセス順序

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, 2 次画像ブロックのコーナーターンで必要な, 1 次キャッシュ 2 5 6 ライン, 2 次キャッシュ 1 6 ラインに対して十分なライン数がある. このため, キャッシュの追い出しだけに縛られずにアクセス順序を選択できる. 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, その影響を解析した上で, 最終的なアクセス順序を決定する.

4. 性能評価

今回実施した評価は, 1. 分割サイズ, 2. アクセス方向, 3. スレッド数とスケジューリングの 3 つのパラメータについてである. まず, 3 章で提案したブロックサイズが最適であるかを判断するため, ブロックサイズを変えて評価した. 次に, 最適なブロックサイズで, アクセスパターンによる違いを検証した. 最後に, スケジューリングによる違いと, スレッド数がプロセッサ数よりも増えた時の性能について評価した.

なお, いずれの評価も, コーナーターン条件と計測環境は同じである.

- 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（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.
- 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)[7] 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

画像ブロック単位での並列実行の効果を検証するため, 並列実行の分割単位である画像のサイズ (ブロックサイズ) を変更してコーナーターンの実行時間を計測する. ここでは, N × N 画素の固まりを単位とする（ブロックサイズ 4 とは, 4 × 4 画素を単位にコーナーターンを並列実行. ブロックサイズ 1 の場合は, 普通に画素単位で実行するのと同じである）. また, 処理全体のアクセス方向と画像ブロック内のアクセス順序を変えて, 計測を行う.

4.1.1 計測条件

- ブロックサイズ
  1 ～ 5 1 2 まで（2 の N 乗の値）.
- 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.

- アクセスパターンとスケジューリング
  Range Straight(RS), Range Reverse(RR), Azimuth Straight(AS), Azimuth Reverse(AR) の4パターン（図8参照）．block スケジューリングでデータを分割．

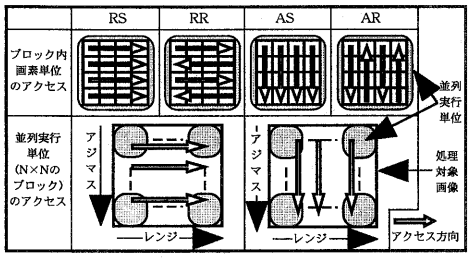

図8　アクセスパターン

4.1.2 計測結果と考察

1thread と 8thread の条件で，ブロックサイズを変えた時の実行性能を図9と図10に示す．また，スレッド数を変えた時の実行性能を図11に示す．ここでは，単純な逐次実行（1thread, RS, ブロックサイズ1）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．

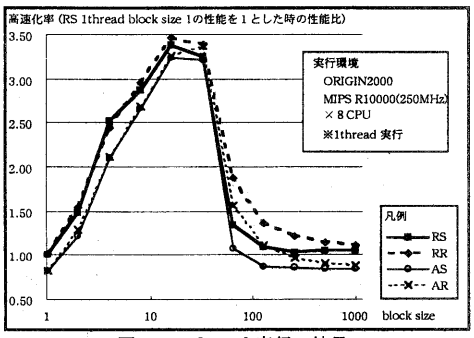

図9　1thread 実行の結果

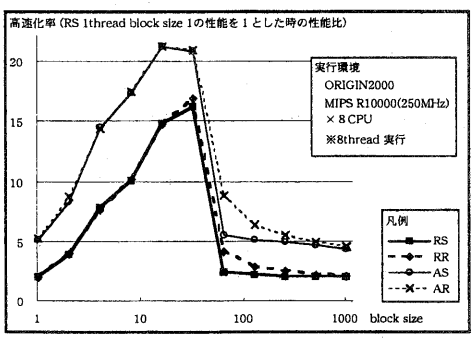

図10　8thread 実行の結果

まず，ブロックサイズについて考察する．図9と図10から，実行性能は，1次キャッシュのラインサイズであるブロックサイズ4を越えて向上し，2次キャッシュのラインサイズであるブロックサイズ16と2倍の32で良い性能となる．これは，2次キャッシュへのアクセスが性能に大きな影響を与えたためと分析する．

ブロックサイズ64で急に性能が低下する原因は，1次キャッシュの必要ライン数の不足によると考えられる．ブロックサイズ32では，必要な1次キャッシュのライン数は512だが，ブロックサイズ64では2048になる．システムリソースは1024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，コーナーターン中に使用中の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低下をまねき，性能が低下したと考える．

ブロックサイズ64以上は，徐々に性能が低下し，ブロックサイズ1の性能に近づく．これは，ブロックサイズ1は，画像全体（サイズ8192）と同等なため，妥当な結果と考える．

ブロックサイズについては，2次キャッシュまでのアクセスを考慮し，その定数倍で実行性能と台数効果が良いことが確認できた．特に，提案手法の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た．サイズをあまり大きくすると，1次キャッシュのライン数不足などを引き起こし，性能低下の原因となることは先に述べた．

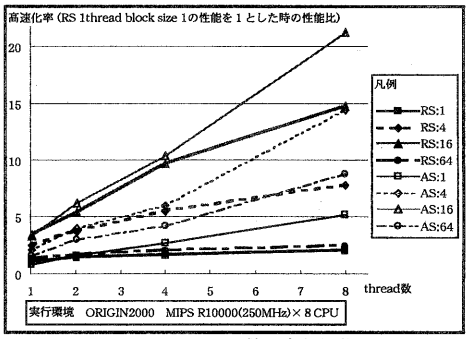

図11　スレッド数と実行性能

スレッド数増加による台数効果についても，実行性能と同様に，ブロックサイズ16で最大の台数効果を得ている(図11参照)．特に，アジマス方向では，8プロセッサの環境で約7倍の台数効果を達成し，単純な逐次実行に比べて20倍以上というスーパーリニアな高速化を達成した．これは，キャッシュのヒット率向上による性能向上と，並列処理による高速化の，双方の相乗効果により達成されたと考える．一方で，ブロックサイズが1の場合，すなわち単純に並列化した場合は，台数効果の限界がレンジ方向では約2倍，アジマス方向でも5倍程度に抑えられる．

このように，提案手法は，台数効果の点でも有利であることが確認できた．

次に，アクセスパターンについて考察する．ブロックサイズ1から32までは，Straight か Reverse かの違いが実行性能に与える影響は小さいことがわかった(図9，10

参照)．一方，ブロックサイズ６４では，Reverseの性能がStraightを比較的大きく上回り，その後サイズが大きくなるにつれて差が減少する．前述のように，ブロックサイズ６４以上ではキャッシュラインの不足が発生する．Reverseはキャッシュの追い出しが起きにくい順序でアクセスするため，性能差が生じたと考える．６４よりブロックサイズが大きくなると，Reverseによるキャッシュの追い出し抑制の効果が薄れ，差が減少していくと考える．

今回の結果から，StraightとReverseパターンは，ブロックサイズ１６である画像ブロック単位でのコーナーターンには，ほとんど影響しないと予測できる．

一方でアクセス方向については，逐次実行時はレンジ（読み込み）方向の連続アクセスが良いが，並列実行時はアジマス（書き込み）方向が逆転し，その差が広がる傾向となった(図９～11参照)．これは，並列実行時には，キャッシュの書き込みミスが，キャッシュ間の一貫性維持のオーバーヘッドを引き起こし，性能に大きな影響を与えるためと分析する．

今回の結果から，アクセス方向による台数効果への影響は，ブロックサイズと同じ程度の比重を占めることがわかった．画像ブロック単位でのコーナーターンの実装を行う時には，考慮していく．

4.2 アクセスパターンの評価

前節の評価で，ブロックサイズ１６で最も良い性能を得られることがわかった．ここでは，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．

4.2.1 計測条件

- ブロックサイズ
  １６固定．
- スレッド数
  １～８(スレッド数１は，並列化オーバーヘッド無し)．
- アクセスパターンとスケジューリング
  blockスケジューリング．アクセスパターンは計測結果ごとに示す．

4.2.2 計測結果と考察

まず，２次画像ブロック内のアクセスを，画素単位または１次画像ブロック単位で実行する場合の違いを計測した．両者について，レンジとアジマスの両方向について計測した結果を図12に示す．

２次画像ブロック内のアクセスでは，１次画像ブロック単位のアクセスで性能が良い．これは，１次キャッシュのデータについて，集中して読み書きを実行した効果と考える．また，スレッド数が増加した際に両者の差が少なくなるのは，高速化により実行時間そのものが減り，並列化の効果が出にくくなっていることが原因と考えている．

この結果から，２次画像ブロック内のアクセスでは，我々の提案している１次画像ブロック単位で実行方法が，単純な画素単位よりも良い性能を得ることが確認できた．

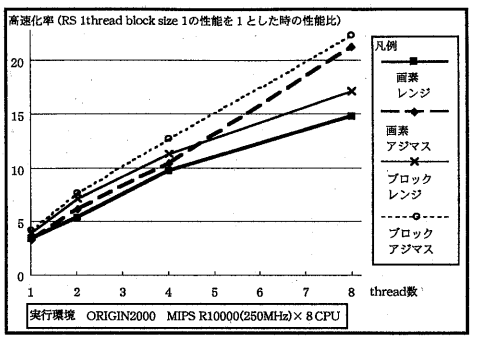

図12　画素 VS １次画像ブロック

次に，アクセス方向の影響について計測する．図13のように各レベルでのアクセス方向を変え，どのレベルが性能に与える影響が大きいかを検証する．

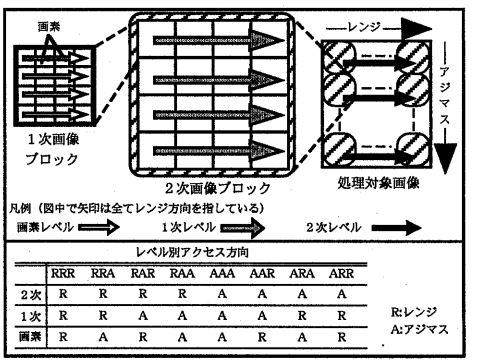

図13　各レベルでのアクセス方向

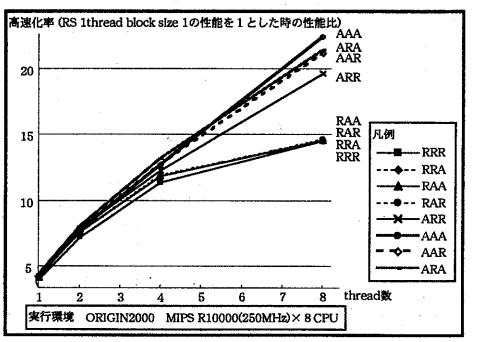

図14　レベル別アクセス方向の性能

図14の結果から，２次画像ブロック単位のコーナーターンでは，画素および１次レベルのアクセス方向が実行時間や台数効果に与えるインパクトは小さいことがわかった．また，２次レベルのアクセス方向が，レンジとアジマスでの台数効果への影響の支配的な要因であることが確認できた．

図12と図14の結果から，2次画像ブロックでは，内部を1次画像ブロック単位でアクセスし，処理全体をアジマス方向（書き込み）に進めていくことで，良い実行性能を得られる見込みを得た．なお，画像ブロック内のアクセスパターン（Straigt または Reverse）は，ブロックサイズ16の2次画像ブロックでは，処理全体に与える影響が低いことを確認している．これは，4.1節の評価と同じであるため，計測結果を省略する．

4.3 スレッド数とスケジューリング

最後に，並列化スケジューリングとプロセッサ数を越えるスレッド数での性能を計測した．

4.3.1 計測条件

- ブロックサイズとアクセスパターン
  全ての計測で次の条件で固定．2次画像ブロック単位（ブロックサイズ16で，2次画像ブロック内のアクセスは1次画像ブロック単位）で，処理全体をアジマス方向に進める．
- スレッド数
  1～10（スレッド数1は，並列化オーバーヘッド無し）．
- スケジューリング
  簡易並列化ライブラリ SPL で提供されている機能から，block，skip，guided-self を選択．

4.3.2 計測結果と考察

図15の結果から，block と guided-self がほぼ同等の性能であり，僅差だが，block スケジューリングが最大性能を得ることがわかる．コーナーターン並列化には，block または guided-self スケジューリングが適していると判断できる．

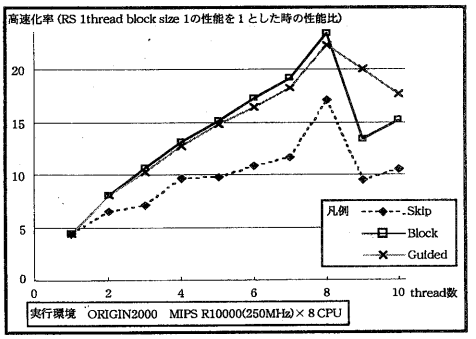

図15　スケジューリング別の性能

スレッド数がプロセッサ数を上回ると，性能が低下することが確認できる．ORIGIN2000 のコーナーターンでは，プロセッサ数以下の並列化を選択することが望ましいと判断できる．また，プロセッサ数を上回るスレッド数では，guided-self が良い性能を示している．高負荷の環境化に陥る危険があるシステムでは，guided-self が適している可能性が高いと考える．

5. ま と め

SAR 画像再生処理の並列化で問題となるコーナーターン部分処理について，画像ブロック単位での並列実行法を提案している．この方法の効果を解析するため，SGI ORIGIN2000 で実行時間を計測した．この結果，8CPU を使って並列化した場合に，単純な逐次処理に対して約20倍の高速化を得た．また，台数効果についても約2倍から7倍へと改善できた．

また，並列実行する画像のサイズやアクセスパターンを変更して，コーナーターンの実行性能に与える影響を評価した．この評価から，画像ブロックとして切り出す画像のサイズが実行性能に与える影響を考察した．また，アクセスパターンについては，処理全体を書き込み側のアドレスに沿って進めることで，より高い台数効果を効率良く得られることがわかった．

我々は，本稿の方法について，SUN Ultra Enterprise でも計測を行い，キャッシュを考慮した並列化で，単純な並列化と比較して，スーパーリニアな高速化（4CPU て約8倍）や，台数効果の改善といった効果を得られることを確認している[6]．このことから，画像ブロック単位でのコーナーターンの並列化方法は，SMP での処理高速化方法として，有効だと考える．

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: 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).
6) 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).
7) 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).