OSCARコンパイラを用いた医用画像フィルタリングのマルチグレイン並列処理

奥村 万里子[1,a)]　柴崎 大侑[1]　桑島 昂平[1]　見神 広紀[1]　木村 啓二[1]　門下 康平[2]　中野 恵一[2]
笠原 博徳[1]

概要：画像処理において，画像フィルタリングは画像内に含まれる雑音除去や物体の輪郭強調など画像を鮮明に表示するために幅広い分野で用いられている．特に医療分野では，CT 検査や内視鏡検査など，体内を撮影して診断を行う画像診断検査が普及しており，医師がより正確な診断を行うためには画像フィルタリングにより得られた高精細な医用画像が求められる．本稿では，画像フィルタリングを行う医用画像処理の中でも特に処理が重い構造強調処理とズーム処理に対し，OSCAR コンパイラを用いて，ループ並列化と 3 種類の画像信号 Y,Cb,Cr の処理を並列に行う粗粒度タスク並列化を組み合わせたマルチグレイン並列化手法を提案する．IBM POWER7 ベースの 128 コア CC-NUMA 型サーバである HITACHI SR16000 上で評価を行った結果，逐次実行に対し，構造強調処理で 70.8 倍，ズーム処理で 58.6 倍の性能向上が得られた．

1. はじめに

近年，高齢化社会が進み，画像処理技術の医療分野への需要が急速に高まっている．特に CT，MRI 検査や内視鏡検査などの画像診断検査は，医師の診断の欠かせないツールになっている [1],[2]．撮影された体内の映像からより正確な診断を行うためには，画像に含まれるノイズの除去や物体の輪郭強調など画像フィルタリングを行い，鮮明な画像を得ることが必要になる．このとき，内視鏡検査のようにその場で体内の様子を観察する場合，リアルタイムで映像を処理する必要があり，高速な画像フィルタリングが求められる．また，CT 撮影機器などの進歩により，短時間で高精細な映像を撮影することが可能になり，1 つの画像データ量が増大している [3]．そのため，画像フィルタリングの演算量が増え，それに伴う処理時間の増加が問題となっており，より一層の高速化が求められている．

画像フィルタリングを含む画像処理はリアルタイム性を実現するため，処理に特化して設計された ASIC [4] などの専用のハードウェアを用いて実装するのが一般的である．しかしながら，ハードウェアによる実装は開発期間や費用が膨大であり，実装後の仕様変更が困難であるという点から，柔軟性を備えてリアルタイム性を実現する FPGA による画像処理 [5],[6] や画像処理用に開発されたプログラマブルプロセッサによるリアルタイム画像処理 [7] の例が数多く報告されている．FPGA などは実装後に再構成可能であり柔軟性を持つが，ソフトウェアで実装することによりさらに柔軟性が増し，短期間，低コストでの開発が期待できる．ソフトウェアの高速化手法として，近年普及しているマルチコアプロセッサによる並列処理が注目されており，今後搭載するコア数が増加すれば，さらなる高速化が期待できる．

本稿では，医用画像処理の中でも特に処理が重い画像フィルタリングである構造強調処理とズーム処理を対象とする．

マルチコアプロセッサを用いた高速化手法としてループ内部の並列性を利用するループ並列化 [8][9] が一般的であるが，処理時間がマイクロ秒単位の処理については使用するコア数の増加に伴い，並列化時のスレッド割り当てや同期等にかかる時間が相対的に大きくなり性能鈍化を招く．そこで，本稿では構造強調処理とズーム処理に対して，OSCAR コンパイラを用いて，ループ並列化に加え，3 種類の画像信号 Y,Cb,Cr の処理を並列に行う粗粒度タスク並列化を組み合わせるマルチグレイン並列化手法を提案する．

まず，第 2 節で画像フィルタリングについて，第 3 節で OSCAR コンパイラを用いて行ったループ並列化とマルチグレイン並列化について，第 4 節で本手法の有効性を評価

[1] 早稲田大学
Waseda University.
[2] オリンパス株式会社
Olympus Corpolation.
[a)] marimo@kasahara.cs.waseda.ac.jp

した結果について，最後に第 5 節で本稿のまとめを述べる．

2. 画像フィルタリング

ここでは，まず，2.1 で画像フィルタリングで行う畳み込み演算について，2.2 で本稿で対象とした構造強調処理とズーム処理について述べる．

2.1 畳み込み演算

画像フィルタリングにおける畳み込み演算とは注目画素とその近傍の画素の濃淡値に，フィルタ係数をかけて重み付けをし，それらの和をとる積和演算である．図 1 に 3x3 フィルタを入力画像の注目画素に適用する例を示す．図 1 より，入力画像の注目画素 g(i,j) とその近傍の 3x3 画素に対して，それぞれ同じ位置にあるフィルタ係数をかけ，それらの和をとった結果を出力画像上の同じ位置 h(i,j) に保存する．これを入力画像のすべての画素に対して適用する．フィルタ係数やフィルタのサイズを変更することで，構造強調処理やズーム処理など様々な処理が実現可能である．

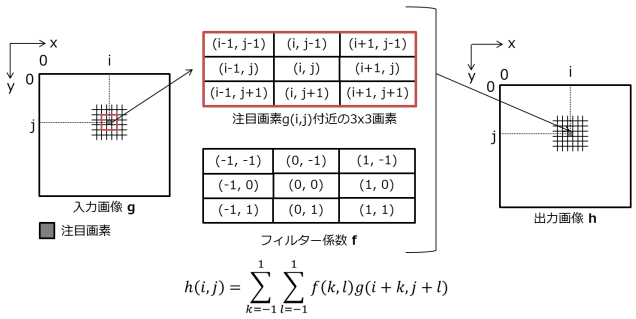

図 1 3x3 フィルタを用いた画像フィルタリングの例

2.2 対象プログラム

次に，構造強調処理とズーム処理の概要について述べる．構造強調処理とズーム処理はそれぞれ図 2 に示す処理フローで構成されている．

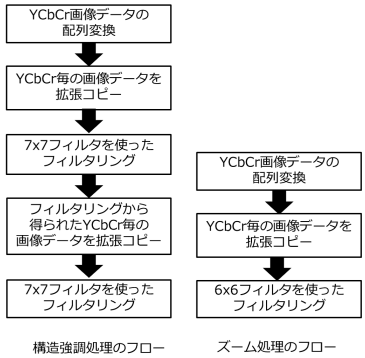

図 2 構造強調処理とズーム処理の流れ

図 2 に示した処理は処理間にデータ依存があり，逐次的に実行される．また，それぞれの処理を画像信号 YCbCr ごとに行う．このとき，構造強調処理とズーム処理両方に共通して，YCbCr 画像データの配列変換処理と拡張コピー処理を行っている．どちらも画像フィルタリングの前処理であり，以下にその処理の目的を説明する．

2.2.1 YCbCr 画像データの配列変換処理

構造強調処理とズーム処理は入力として図 3 のような 3 種類の画像信号 Y(輝度),Cb(輝度と青色の差),Cr(輝度と赤色の差) が点順次方式に並んだ画像データ配列を用いている．また，前述したように図 2 に示す処理は 3 種類の画像信号それぞれに対して行う．例えば，Y に対して処理を施す場合，画像データ配列の Y の情報だけを参照すればよい．このとき，点順次方式に信号の情報が配置されたデータを参照する場合，ストライドアクセスでキャッシュの利用効率がおち，メモリへのアクセスが発生するため，メモリアクセスレイテンシの影響でフィルタリングを行う際の処理時間が増加してしまうことが考えられる．そこで，図 4 に示すように，それぞれの信号情報を異なる領域に抽出することで連続的なアクセスを可能とし，メモリアクセスの効率化が期待できる．

Y	Cb	Cr	Y	Cb	Cr	Y	Cb	Cr
Y	Cb	Cr	Y	Cb	Cr	Y	Cb	Cr
Y	Cb	Cr	Y	Cb	Cr	Y	Cb	Cr

図 3 YCbCr が点順次方式に配置された画像データ配列

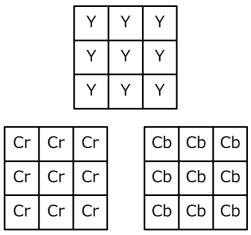

図 4 YCbCr 毎の画像データ配列

2.2.2 拡張コピー処理

フィルタリングでは注目画素の近傍の画素を用いて計算を行う．このとき，注目画素が画像の端部である場合，近傍画素が存在しないことになる．そこで，最も近い端部の画素値をコピーすることでフィルタサイズ分の領域を確保する．図 5 に 6x6 フィルタをかける場合の画像端部の拡張コピーの例を示す．図 5 に示したように，入力画像 g の注目画素 g(0,0) に対して 6x6 フィルタをかける場合，g(0,0) の近傍 6x6 画素分の領域が必要になる．g(0,0) は画像の端部であるため，近傍画素が存在しない部分に最も近い端部

の画素値を折り返してコピーし，それらを含めた 6x6 画素を抽出する．その他の端部についても必要な分だけ拡張コピーし，フィルタサイズ分の領域を抽出する．

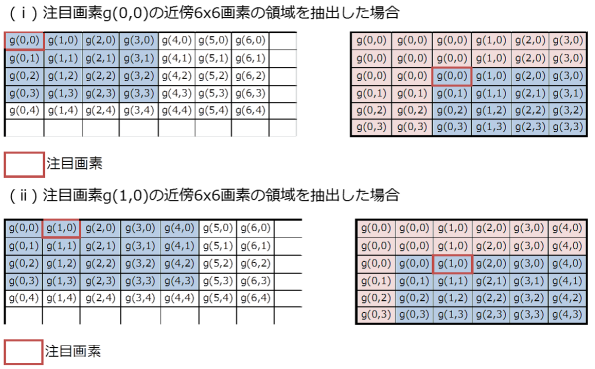

図 5 6x6 フィルタをかける前処理の拡張コピーの例

3. ループ並列化とマルチグレイン並列化

本節では，OSCAR コンパイラを用いた構造強調処理とズーム処理の並列化手法について述べる．3.1 では OSCAR コンパイラの概要について，3.2 ではループ並列化について，3.3 では提案するマルチグレイン並列化手法について述べる．

3.1 OSCAR コンパイラ

OSCAR コンパイラとは早稲田大学笠原・木村研究室で開発している自動並列化コンパイラである [10]．逐次的に記述された C 言語のプログラムから並列化された C 言語のプログラムを自動生成する．まず，ソースプログラムを代入文などの基本ブロック (BB)，ループなどの繰り返しブロック (RB)，関数呼び出しなどのサブルーチンブロック (SB) の 3 種類のマクロタスクに分割し，さらに RB や SB の内部も同様に分割することで階層的なマクロタスクを生成する．マクロタスクを生成後，マクロタスク間のデータ依存関係とコントロールフローを解析し，解析結果としてマクロフローグラフを生成する．そして，マクロタスク間の並列性を抽出するためにマクロフローグラフに対し最早実行可能条件解析を適用し，その結果をマクロタスクグラフに表現する．最後に，マクロタスクグラフ上のマクロタスクを複数のプロセッサに割り当てて実行することにより並列処理を実現する [11],[12]．

3.2 ループ並列化

ここでは OSCAR コンパイラを用いたループ並列化について述べる．OSCAR コンパイラにより RB に分類されたマクロタスクの中で，ループイタレーション間に依存がなく，イタレーション間で通信なしに実行可能である DOALL と判定されたループに対してループ並列化を行う．図 6 に OSCAR コンパイラによる並列性解析の結果として生成された構造強調処理とズーム処理のマクロタスクグラフを示す．ここで，マクロタスクグラフ中の青いブロックは DOALL と判定されたループを表し，赤いブロックは BB の集合である．図 6 より，構造強調処理もズーム処理も大部分が DOALL ループで構成されており，マクロタスク間の並列性もないためループ並列化が有効である．しかしながら，逐次実行時間がマイクロ秒単位のタスク粒度の細かい処理についてはコア数増加に伴い，メモリアクセスレイテンシやスレッド生成，ループ終了時のバリア同期のオーバーヘッドが相対的に大きくなり，性能鈍化を招くことが予想される．

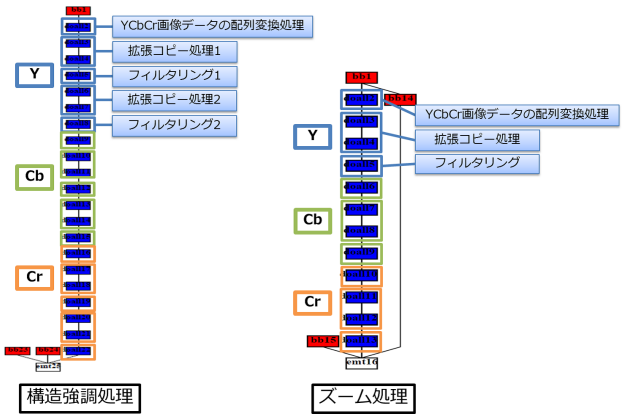

図 6 構造強調処理とズーム処理のマクロタスクグラフ

3.3 画像信号間の並列性を利用したマルチグレイン並列化

3.2 で予想されるタスク粒度の細かい処理における性能鈍化を改善する方法として，必要以上に多くの分割を行わないことが考えられる．そこで，3.3 では，ループの分割数はそのループの速度向上に有効な範囲までとし，これまで利用されていなかったループ間の粗粒度タスク並列性を抽出するマルチグレイン並列化手法を提案する．

3.3.1 画像信号間のデータローカリティ抽出手法

ループ並列化では，図 6 に示すように，画像信号ごとの処理は Y → Cb → Cr とそれぞれ逐次的実行されていた．ここで画像信号間の処理に依存がなく並列実行可能であることに着目し，配列や変数のリネーミングを行い，別領域での計算を可能にすることで，図 7 に示すような信号間の並列性を抽出した．図 7 より，YCbCr それぞれの処理が横に並び，並列実行可能な形となっている．また，マクロタスク間をつなぐ線が一直線で信号ごとに独立し，他の信号に対してデータ依存がないことがわかる．並列実行時のループ分割については，図 8 に示すように，6 コアで並列化した場合，横に並んだ Y,Cb,Cr それぞれのループが 2 分

割ずつされ，分割されたループはコア 0 からコア 5 に割り当てられる．このようにして YCbCr ごとのループ並列化に加え，ループ間の並列化を組み合わせるマルチグレイン並列化を実現する．

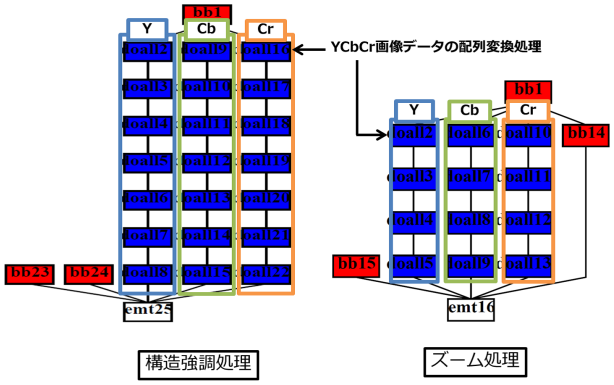

図 7 画像信号間の処理の並列性を抽出した構造強調処理とズーム処理のマクロタスクグラフ

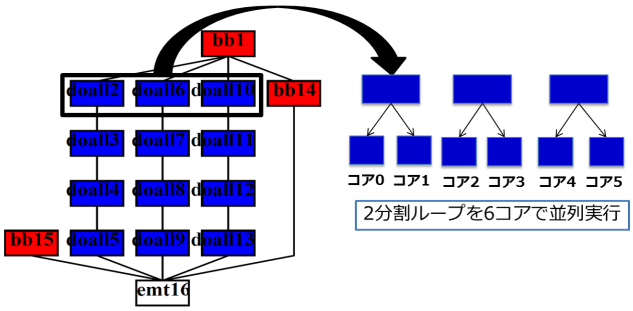

図 8 6 コア使用時におけるループ分割例

次に，このループ間の粗粒度タスク並列化を導入すると，信号間の並列化においてループ並列化では起こらなかったメモリアクセスの競合が生じることが判明した．信号間の並列処理において，YCbCr 画像データの配列変換処理は同時に行われることになる．このとき，点順次方式に配置された画像配列データを複数のコアが同時に参照することになるため，メモリアクセスの競合が起こり，性能悪化を招いていた．そこで，入力時点から色ごとに連続的に配置された YCbCr 毎のローカルな領域を保持することで，メモリアクセスの効率化と，さらに YCbCr 画像データの配列変換処理削除による逐次性能向上を図った．また，本稿で扱った医用画像処理ではズーム処理と構造強調前処理と構造強調処理が連続的に処理される処理フローとなっているため，図 9 に示すように，ズーム処理と構造強調処理を 1 つの関数の中にループが並ぶようにインライン展開し，2 つの処理間のデータ転送オーバーヘッドを削減することでさらなる性能向上を図った．

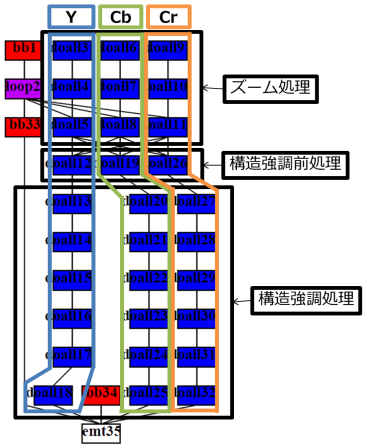

図 9 1 つの関数にインライン展開されたズーム処理と構造強調処理のマクロタスクグラフ

4. 性能評価

本節では，構造強調処理とズーム処理に対し，OSCAR コンパイラを用いて，ループ並列化と，ループ並列化に加え，画像信号 YCbCr 間のデータローカリティを抽出し，階層的並列化を行うマルチグレイン並列化手法を適用し，IBM POWER7 ベースの 128 コア CC-NUMA 型サーバである HITACHI SR16000 上で評価した結果を述べる．まず，4.1 で評価環境について，4.2 で評価結果について述べる．

4.1 評価環境

本稿では HITACHI SR16000 を用いて評価を行なった．表 1 に SR16000 の仕様を示し，図 10 に SR16000 のアーキテクチャを示す．SR16000 は POWER7[13] を 16 個搭載した 128 コアの SMP サーバである．POWER7 は 1 プロセッサあたり 8 個のコアを持ち，8 コアで 32MB のオンチップ L3 キャッシュを共有している．また，図 10 に示すように各プロセッサごとに分散共有メモリを持つ CC-NUMA アーキテクチャである．このとき，プログラムで使用する変数は first touch policy で分散共有メモリに配置される．first touch について，構造強調処理とズーム処理では初期値設定ループを並列化することにより考慮しているが，本稿では初期値設定ループの並列化は評価対象外とする．また，ネイティブコンパイラとして gcc のバージョン 4.4.7 を使用し，コンパイルオプションには-O3，-fopenmp，-mtune=power7，-mcpu=power7 を指定している．加えて，環境変数 GOMP_CPU_AFFINITY により，スレッドのコアバインドを行っている．プログラムの入力画像は解像度 1920x1080 の RAW 画像を 1 枚使用している．

4.2 評価結果

図 11 に構造強調処理に対し，ループ並列化を適用した

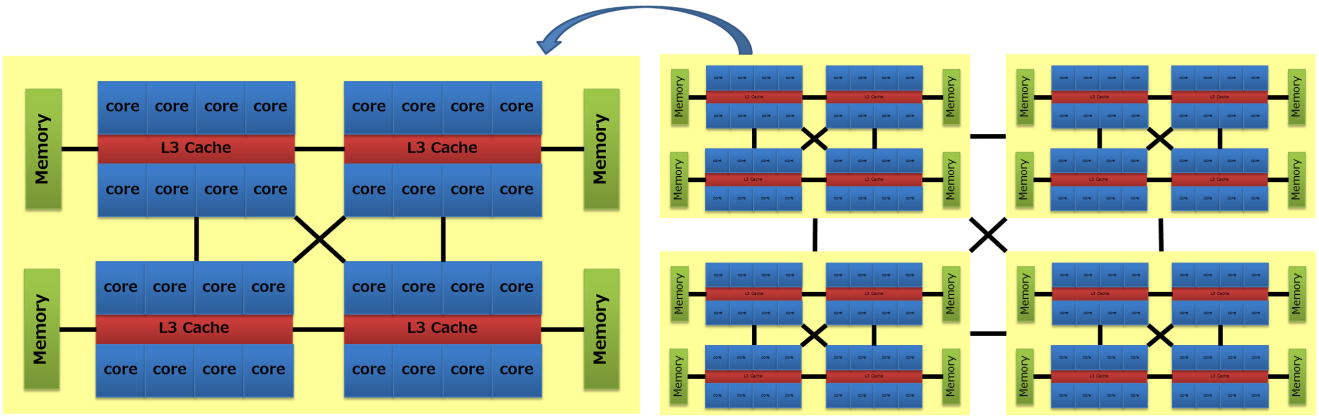

図 10 HITACHI SR16000 のアーキテクチャ

表 1 HITACHI SR16000 仕様

System	
CPU	IBM POWER7
Core	128cores(8cores/chip)
Frequency	4.0GHz
L1 D-Cache	32KB/core
L1 I-Cache	32KB/core
L2 Cache	256KB/core
L3 Cache	32MB/8cores

際の逐次実行に対する速度向上率を示し，図 12 にズーム処理に対し，ループ並列化を適用した際の逐次実行に対する速度向上率を示す．また，図 13 に拡張コピー処理に対し，ループ並列化を適用した際の逐次実行に対する速度向上率を示す．続いて，図 14 に構造強調に対し，ループ並列化を適用した場合と本手法を適用した場合の逐次実行に対する速度向上率の比較を示し，図 15 にズーム処理に対し，ループ並列化を適用した場合と本手法を適用した場合の逐次実行に対する速度向上率の比較を示す．ここで，グラフの横軸は使用したコア数を示し，縦軸は各処理の逐次実行に対する速度向上率を示す．

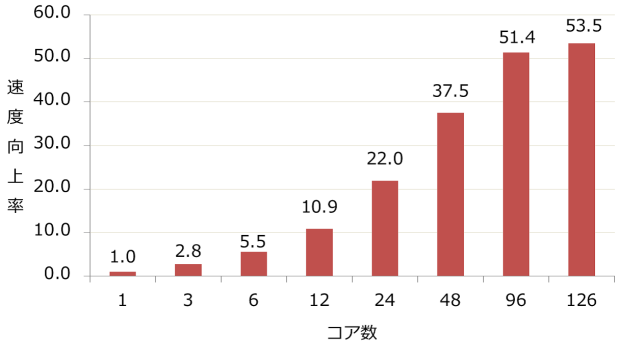

図 11 構造強調処理のループ並列化適用時の速度向上率

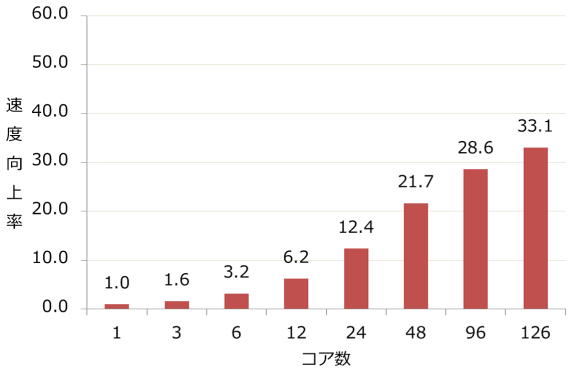

図 12 ズーム処理のループ並列化適用時の速度向上率

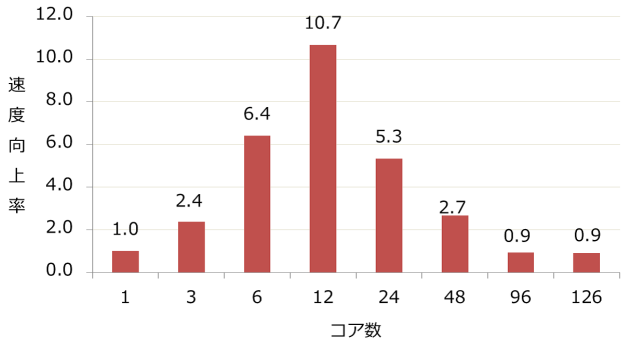

図 13 拡張コピー処理のループ並列化適用時の速度向上率

ループ並列化のみを適用した場合は，構造強調処理では図 11 より，逐次実行に対して，126 コア使用時 53.5 倍の速度向上が得られ，ズーム処理では図 12 より，逐次実行に対して，126 コア使用時 33.1 倍の速度向上が得られた．これにより OSCAR コンパイラによるループ並列化が有効であったといえる．しかしながら，処理毎の性能を評価した結果，図 13 より，3.2 で予想したように，逐次実行時間がマイクロ秒単位の粒度の細かい処理について 24 コア以降で性能鈍化が起きていることを確認した．続いて，ループ並列化適用時と本手法適用時を比較すると，構造強調処理

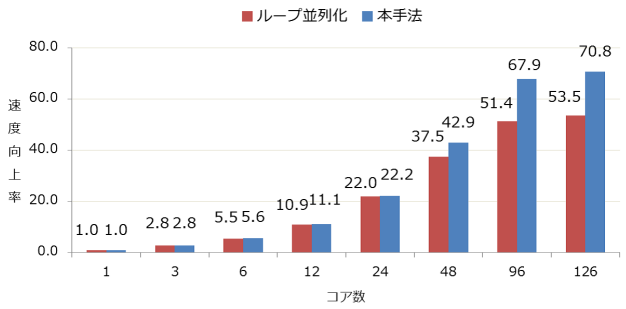

図 14 構造強調処理のループ並列化適用時と本手法適用時の速度向上率

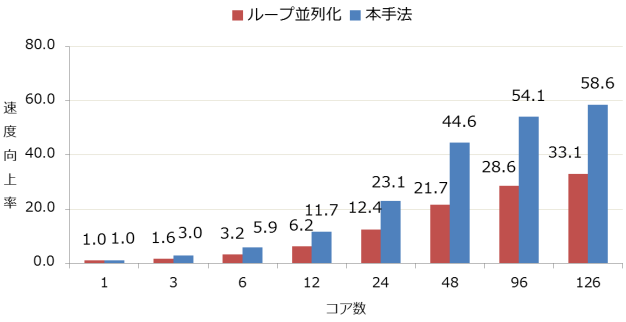

図 15 ズーム処理のループ並列化適用時と本手法適用時の速度向上率

では図 14 より，逐次実行に対して，126 コア使用時ループ並列化のみでは 53.5 倍の速度向上に対し，本手法では 70.8 倍の速度向上が得られた．また，ズーム処理では図 15 より，逐次実行に対して，126 コア使用時ループ並列化のみでは 33.1 倍の速度向上に対し，本手法では 58.6 倍の速度向上が得られた．これにより，ループ並列化のみに比べて，本手法が有効であると確認できた．これは，ループ並列化のみではコア数の増加に伴い，同期のオーバーヘッドが増大し，速度向上が得られにくいループに対して，各ループは効率良く実行できる範囲のプロセッサ数で分割し，それらのループを同時に実行するマルチグレイン並列化手法により，より多くのプロセッサを効果的に使用できるようになったためである．また，マルチグレイン並列化において，画像信号ごとにローカルな領域を保持することによるメモリアクセスの効率化や処理間のデータ転送を最小限に抑えることが有効であると確認できた．

5. まとめ

本稿では，画像フィルタリングを行う医用画像処理の中でも特に処理の重い構造強調処理とズーム処理に対し，OSCAR コンパイラを用いて，3 種類の画像信号ごとのループ並列化に加え，ループ間のデータローカリティを抽出し，信号間の粗粒度タスク並列化を組み合わせるマルチグレイン並列化手法を提案した．8 コア集積の IBM POWER7 プロセッサを 16 個搭載した 128 コアの CC-NUMA 型サーバである HITACHI SR16000 上で評価を行った結果，逐次実行に対し，構造強調処理では 70.8 倍，ズーム処理では 58.6 倍の速度向上が得られた．ループ並列化のみの場合は，拡張コピー処理のような実行時間がマイクロ秒単位の非常に粒度の細かい処理については 128 コア等のメニーコアにより分割された際，並列実行時のスレッド生成やループ終了時のバリア同期のオーバーヘッドが相対的に大きくなってしまうため，並列性能が鈍化していた．そこで，画像信号間のデータローカリティを抽出し，各ループの分割はそのループの速度向上に有効な範囲までとし，それらのループを同時に実行する階層的な並列化を行うことで，より多くのプロセッサを効果的に使用できるようになり，ループ並列化以上の高い並列性を確保できるようになった．また，マルチグレイン並列化において，信号間の処理を同時に行うことで起きていたメモリアクセス競合を解消するため，信号ごとにローカルな領域をもつようにし，メモリアクセスの効率化を図った．さらに，ズーム処理と構造強調処理が連続する処理フローであることを利用し，処理を 1 つの関数の中にループが並ぶようにインライン展開することで処理間のデータ転送によるオーバーヘッドを抑え，さらなる高速化が実現できた．本手法はプログラムを手動でチューニングし，OSCAR コンパイラによる解析を行ったが，今後はコンパイラに実装して自動化する予定である．

参考文献

[1] 藤田広志：医用画像のためのコンピュータ支援診断システムの開発の現状と将来，日本写真学会誌,66 巻 5 号,484-490 (2003).
[2] 小畑秀文：医用画像の計算機支援診断技術の現状と動向，医用画像情報学会雑誌,Vol.21,No.1,11-18 (2004).
[3] 小畑秀文：消化管 CT 三次元診断の現状と将来展望，日本消化器病学会雑誌,Vol.108,No.6,899-907 (2011).
[4] 菊池迪夫：ASIC, 計測と制御,Vol.27,No.6,523-530 (1988).
[5] 平井慎一，座光寺正和，増渕章洋，坪井辰彦：FPGA ベースリアルタイムビジョン，日本ロボット学会誌,Vol.22,No.7,873-880 (2004).
[6] 井上恵介，亀田成司，八木哲也：シリコン網膜と FPGA を用いた実時間並列画像処理，映像情報メディア 61(3), 316-324 (2007).
[7] 財団法人埼玉県産業振興公社：超並列画像処理組み込みミドルウェア開発による高度計測システムの実証，研究開発成果報告書 (2011).
[8] Wolfe, M. J.: High Performance Compilers for Parallel Computing, *Addison-Wesley Longman Publishing Co.* (1995).
[9] Banerjee, U. K.: Loop Parallelization, *Kluwer Academic Publishers Norwell* (1994).
[10] 本多弘樹，岩田雅彦，笠原博徳：Fortran プログラム粗粒度タスク間の並列性検出手法，電子情報通信学会論文誌 (1990).
[11] 小幡元樹，笠原博徳，木村啓二：OSCAR チップマルチプロセッサ上でのマルチグレイン並列処理，情報処理学会 (2002).
[12] 笠原博徳：並列処理技術，コロナ社 (1991).

[13] D. Wendel, R. Kalla, R. C. J. C. J. F. R. F. J. K. B. S. W. S. S. T. S. W. S. G. C. S. I. and Zyuban, V.: The implementation of Power7: A highly parallel and scalable multi-core high-end server processor, *ISSCC,102-103* (2010).