情報処理学会論文誌　コンピューティングシステム　Vol. 3　No. 3　178–188 (Sep. 2010)

異なるアーキテクチャのメニーコアプロセッサにおけるステレオマッチングプログラムの高速化と性能評価

岩田健司[†1]　中村良介[†1]　田中良夫[†1]
増田知記[†1,†2]　町田亮介[†1,†2]
小島　功[†1]　関口智嗣[†1]

近年の衛星センサの高分解能化やデータアーカイブなどの IT 基盤の整備により，衛星画像から DEM（数値標高モデル）を生成するためのステレオマッチングプログラムの高速化が求められている．この論文では，ステレオマッチング処理における相関計算部分において，Cell や GPGPU などのメニーコアプロセッサに対して最適化，高速化を行い，その手法を述べ性能を評価する．相関計算方法として，ZNCC（正規化相互相関）と census 変換法の 2 種類を評価する．ZNCC では差分計算法を用いて高速化する．これまでに ZNCC を簡略化した場合における差分計算法が提案されているが，この論文では簡略化せず差分計算する方法を考案して使用する．Xeon（Nehalem）においては，OpenMP や SIMD 化などにより最適化前と比べ約 28〜38 倍に高速化された．Cell（PowerXCell 8i）では，SPE における DMA 転送の最適化などにより，最適化前の PPE での実行時間と比べ約 414〜765 倍に高速化された．GPGPU（Tesla C1060）では Cell と同等，Xeon と比べ 2 倍の性能となった．

Acceleration and Performance Evaluation of the Stereo-matching Program on Some Many-core Processors

Kenji Iwata,[†1] Ryosuke Nakamura,[†1]
Yoshio Tanaka,[†1] Tomoki Masuda,[†1,†2]
Ryosuke Machida,[†1,†2] Isao Kojima[†1]
and Satoshi Sekiguchi[†1]

Acceleration of the stereo-matching program to generate DEM (Digital Elevation Model) from the satellite images is required by providing IT infrastructure such as data archive, high-resolution sensor, etc. This paper evaluate the acceleration of correlation in stereo-matching program on many-core architecture. ZNCC (Zero mean Normalized Cross Correlation) and census transform are evaluated as the correlation method. The recursive calculation is implemented to accelerate ZNCC. The optimized program on Xeon (Nehalem) is about 28 to 38 times faster than original program by using OpenMP, SIMD, etc. The optimized program on Cell (PowerXCell 8i) is about 414 to 765 times faster than original program on PPE by optimizing the DMA transfer in SPE. A performance using GPGPU (Tesla C1060) is equal to Cell, twice faster than Xeon.

1. はじめに

近年，地球温暖化などの環境問題，地震や洪水などの災害予測・対策，および天然資源探索などの地球観測に関する問題が重要になっている．GEO Grid（Global Earth Observation Grid，地球観測グリッド）[1] は，グリッド技術を用い，地球観測衛星データなどの大規模アーカイブおよびその高度処理を行い，分散環境下の各種観測データや地理情報システムデータを統合・融合した処理・解析を，ユーザが手軽に扱えることを目指したシステムかつコンセプトであり，地球観測情報のインフラとして期待されている．

GEO Grid は今までに 150 万枚以上の ASTER [2] 画像をインターネット上でアクセス可能とし，様々な計算を行うサービスの提供を行っている．しかし，既存の地理空間情報システムで使われているアプリケーション（GIS アプリケーション）の多くは最近の高性能ハードウェアを活用できないレガシなものであり，ユーザが要求する画像処理に数時間程度かかってしまうなど，ユーザに対する利便性が非常に悪い．また，災害対策など迅速な対応が必要とされる分野への適用に問題がある．ユーザに対する利便性を高め，緊急性の高い問題への適用を可能とするために，GIS アプリケーションの高速化は非常に重要な課題である．

その一方で，近年のマルチコア，メニーコアプロセッサの普及により，従来の数倍から数十倍の性能を手軽に得られる環境が整ってきており，大量のデータを処理する必要のある信号処理や画像処理などでの応用において，劇的な高速化が期待されている．しかしマルチコアには様々なアーキテクチャが存在している．なかには倍精度演算やメモリアクセスなどに

†1 産業技術総合研究所情報技術研究部門
Information Technology Research Institute, National Institute of Advanced Industrial Science and Technology (AIST)
†2 株式会社フィックスターズ
Fixstars Corporation

© 2010 Information Processing Society of Japan

制限を受けるものもあり，各アーキテクチャに対する並列化手法とその効果はまったく異なるものである．たとえば，電波望遠鏡での信号の相関計算における様々なマルチコアアーキテクチャによる高速化が報告されている[3]が，この結果がそのまま他の応用に適応できるわけではない．

我々は，GIS アプリケーションの高速化の一環として，DEM（Digital Elevation Model，数値標高モデル）生成アプリケーションの高速化を進めている．この論文では，DEM の生成におけるステレオマッチングをケーススタディとして，異なるアーキテクチャのマルチコア・メニーコアを用いた高速化手法とその効果を述べる．一般的な同一コアを搭載したホモジニアスマルチコアである Xeon，マルチメディア処理などに特化したコアを混載したヘテロジニアスマルチコアである Cell，画像処理用プロセッサを用いる GPGPU に対して，それぞれ最適化を行い手法と効果を検証する．最先端のハードウェア上で既存のアプリケーションを最適化する，あるいは新規に開発するという要求は地球科学分野に限らず多々あるものであり，本報告がそれらの研究開発に対して有益な資料となることが期待される．

今回はホモジニアスマルチコアとして，Intel x86 アーキテクチャの CPU である Xeon 5500 シリーズ（Nehalem コア），ヘテロジニアスマルチコアとして，Cell アーキテクチャの IBM PowerXCell 8i，GPGPU として NVIDIA Tesla C1060 を用いる．この論文では，特に DEM 生成アプリケーションの主要コンポーネントであるステレオマッチングにおける相関計算に着目し，正規化相互相関（ZNCC，Zero mean Normalized Cross Correlation）と，census 変換法[4] の 2 方式について，高速化の効果を検証する．

2. ハードウェアの概要

今回使用したハードウェアのスペックと実行環境を，**表 1** に示す．各プロセッサについて以下に述べる．

表 1　評価に用いたハードウェアのスペックと実行環境
Table 1　Hardware specs and environments used in the experiments.

	HP Z800 Workstation	IBM BladeCenter QS22	NVIDIA Tesla C1060
プロセッサ	Xeon X5550	PowerXCell 8i	Tesla T10
動作周波数	2.66 GHz	3.2 GHz	1.296 GHz
プロセッサ数	2	2	1
コアの構成	8	2 PPE + 16 SPE	240（30 SM × 8 SP）
メモリ搭載量	4 GB	8 GB	4 GB
メモリスペック	DDR3-1333	DDR2-800	512 bit GDDR3
OS	Linux（Fedora 10）	Linux（YDEL 6.1）	—
コンパイラ	gcc 4.3.2	spu-gcc 4.1.1，ppu-gcc 4.1.1	CUDA release 2.3

2.1 Intel Xeon 5500 シリーズ（Nehalem）

Intel 社製の Xeon 5500 シリーズは Nehalem コアの Intel アーキテクチャCPU であり，同一性能の 4 コアを統合したホモジニアスマルチコアプロセッサである．並列化に関しては，すべてのコアで同一のプログラムが動作するため，開発は比較的容易である．SIMD として SSE4.2 までサポートしており，4.4 節で述べる census 変換法に有利な命令が実装されている．一般的な PC サーバが利用できるため，調達が容易であるというメリットがある．今回は HP Z800 Workstation を（以下 Z800 と呼ぶ）評価に用いる．

2.2 IBM PowerXCell 8i

IBM 社製の PowerXCell 8i は，メディア処理に適した Cell アーキテクチャのプロセッサであり，Cell Broadband Engine（Cell/B.E.）の倍精度浮動小数点演算性能を 5 倍に強化したプロセッサである．今回の DEM 生成プログラムにおいては倍精度演算が必要なため，Cell/B.E. より適している．また，特殊なメモリを利用する Cell/B.E. に対し，汎用の DDR2 DRAM を利用することからメモリ搭載の制限が少なく，より大規模計算に向いている．

PowerXCell 8i は，通常の演算命令を実行する PPE を 1 基と，SIMD 演算を高速に実行する SPE を 8 基で構成されている，ヘテロジニアスマルチコアである．PPE はそれほど高速ではないため，高速化には SPE に処理を振り分けることになる．SPE を用いた並列化には，256 KB のローカルストレージへの DMA 転送など，専用のプログラム設計が必要となり，Xeon などのホモジニアスマルチコアの場合と比べて，作業工数を多く要する．今回は，PowerXCell 8i を 2 基搭載した IBM BladeCenter QS22（以下 QS22 と呼ぶ）を評価に用いる．

2.3 NVIDIA Tesla T10

CG 用に設計された GPU を CG 以外の目的に用いる GPGPU は，用途によって非常に高い演算性能を得られるため，近年注目を集めている．NVIDIA 社製の Tesla T10 は，スカラー演算を行う SP（ストリーミング・プロセッサ）を 8 個搭載した SM（ストリーミング・マルチプロセッサ）30 個で構成されており，全体で 240 個の演算コアを持つ．今回は Tesla T10 を 1 基搭載したボードである C1060 を前出の Z800 に搭載し評価に用いる．

プログラミング言語として CUDA を利用する．CUDA では問題を大量のスレッドに分解して扱う．さらにスレッドを集めた Block，Block を集めた Grid と呼ばれる単位で処理を行う．SM では条件が合えば複数のブロックを同時に実行することが可能である．1 つの

© 2010 Information Processing Society of Japan

Block 内でどれだけのスレッドが同時実行できるかは，1 つのスレッドが利用するリソースの使用量に左右される．1 つの SM には 16,384 個のレジスタがあり，1 つのスレッドで使用するレジスタ数が増えると，同時に実行可能なスレッド数が減少することとなる．また各 SM には，高速なアクセスが可能な Shared memory を 16 KB ずつ搭載しており，高速化のためにはこれを効率的に用いる必要がある．

3. ステレオマッチングプログラム

ステレオマッチングとは，視点の異なる 2 枚の画像間の同一地点を探し出し，三角測量の原理で 3 次元の立体情報を求める処理のことである．衛星画像においては，直下視と後方視（もしくは前方視）のセンサ画像間の同一地点を探索し，標高情報に変換する処理となる．

3.1 プログラムの構成

今回高速化するステレオマッチングプログラムは，文献 5) で用いたものと同じものである．データ入力や初期化を行う前処理部，ステレオマッチングを行う主処理部，データ出力を行う後処理部に大きく分けられる．このうち主処理部のステレオマッチングは，相関計算，ノイズを除去するためのメディアンフィルタ，欠損データを周囲のデータから補完する異常値内挿の 3 つの処理で構成される．

主処理部のループ構成の概要を**図 1** に示す．最初の coarse-to-fine 法のループは，解像度の低い coarse 画像から始めて大局的な対応付けを行い，その結果を用いて探索範囲パラメータを変更し，徐々に解像度の高い fine 画像に切り替えて処理を行うものである．サブステージループは，相関値が十分でないなど誤りと考えられる部分において，探索範囲パラメータを広げて再探索を繰り返すものである．このように coarse-to-fine 法ループとサブステージループでは，イテレーション間に依存関係がある．画像全体を走査するための y 方向ループ，x 方向ループにはループ伝搬依存がなく，並列演算が可能である．ただし，4.2 節の差分計算による高速化を導入した場合は，この限りではない．

```
for () { // coarse-to-fine 法ループ
  for () { // サブステージループ
    for (y) { // y 方向 loop
      for (x) { // x 方向 loop
        // 相関計算
  }}}
   // メディアンフィルタ
   // 異常値内挿
}
```

図 1　ステレオマッチングプログラムのループ構成
Fig. 1　Loop structure of the stereo matching program.

3.2 評価データとプロファイル

ステレオマッチングプログラムの最適化の評価に用いる画像データを**図 2** に示す．この画像の大きさは，3962 × 4200 ピクセルである．ハワイ島のマウナ・ケア山周辺の画像であり，海抜 0 m から 4,205 m まで同一画像内に存在しており，視差量が大きくベンチマークデータとして適している．

このデータに対し，Z800 上において最適化前のプログラムにおける実行時間をプロファイルしたところ，相関計算が全体の実行時間の約 55%を占めていた．そこで，相関計算部分を重点的に最適化することとする．

4. ステレオマッチングにおける相関計算

ステレオマッチングにおける，相関計算による同一地点の探索処理について述べる．相関計算法として ZNCC は衛星画像のステレオマッチングにおいて一般的によく用いられているが，計算負荷が高いという問題がある．前処理に工夫を加えることで，負荷の低い計算式を用いる方法[6] や，ZNCC を簡略化した式を用いて差分計算により高速化する方法[7] などが報告されている．この論文では，ZNCC を簡略化せずに差分計算による高速化が可能で

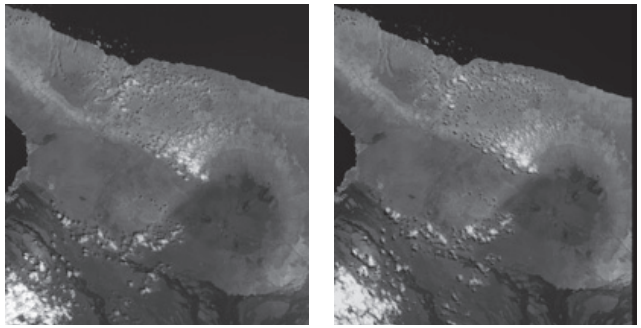

図 2　評価用画像データ
Fig. 2　Image data for evaluation.

© 2010 Information Processing Society of Japan

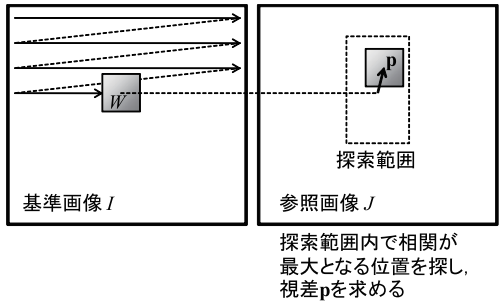

図 3　探索処理の概要
Fig. 3　Search procedure.

あることを述べる．もう 1 つの相関計算法として，人工建築物の屋根の検出[8] などに利用されている census 変換法[4] についても，SIMD などを利用した高速化を行うので，その概要を述べる．

4.1 探索処理

対応位置の探索処理は**図 3** のようになる．2 枚の画像のうち 1 枚を基準画像 I，もう 1 枚を参照画像 J と関数で表記する．画像中の点をベクトル $\mathbf{q}$ で表すと，$I(\mathbf{q})$ は基準画像中の点 $\mathbf{q}$ における明度を示す．あらかじめ決められた大きさの矩形で基準画像 I から部分画像を切り出す．これをテンプレートと呼ぶ．参照画像 J におけるあらかじめ決めた探索範囲内において，テンプレートとの一致を測ることで，どの程度ずれているかを示す視差のベクトル $\mathbf{p}$ を探索する．テンプレートの矩形の大きさを X，Y，左上の座標を $\mathbf{r} = (x_r, y_r)$ とすると，テンプレートに用いる矩形内の座標の集合 W は，

$$W = \{(x, y) | x = x_r, \ldots, x_r + X - 1, y = y_r, \ldots, y_r + Y - 1\}, \tag{1}$$

となる．以降は集合 W を用いて説明する．

4.2 ZNCC の差分計算

視差 $\mathbf{p}$，矩形の座標 W における ZNCC の値 $C_{ZNCC}(\mathbf{p}, W)$ は次の式で表される．

$$C_{ZNCC}(\mathbf{p}, W) = \frac{\sum_{\mathbf{q}\in W} \left\{I(\mathbf{q}) - \bar{I}(W)\right\} \left\{J(\mathbf{p}+\mathbf{q}) - \bar{J}(\mathbf{p}, W)\right\}}{\sqrt{\sum_{\mathbf{q}\in W} \left\{I(\mathbf{q}) - \bar{I}(W)\right\}^2} \sqrt{\sum_{\mathbf{q}\in W} \left\{J(\mathbf{p}+\mathbf{q}) - \bar{J}(\mathbf{p}, W)\right\}^2}}. \tag{2}$$

ここで $\bar{I}$，$\bar{J}$ は矩形内の明度の平均値で，

$$\bar{I}(W) = \frac{1}{|W|} \sum_{\mathbf{q}\in W} I(\mathbf{q}), \tag{3}$$

$$\bar{J}(\mathbf{p}, W) = \frac{1}{|W|} \sum_{\mathbf{q}\in W} J(\mathbf{p}+\mathbf{q}), \tag{4}$$

である．ただし $|W|$ は集合 W の元の個数（$= X \times Y$）である．

このように ZNCC は積和の繰返しで計算負荷が高い．ZNCC の平均を 0 とした NCC（Normalized Cross Correlation）では，差分計算を導入することにより計算量を大幅に減らすことができることが報告されている[7]．視差 $\mathbf{p}$，矩形の座標 W における NCC の値 $C_{NCC}(\mathbf{p}, W)$ は次の式で表される．

$$C_{NCC}(\mathbf{p}, W) = \frac{\sum_{\mathbf{q}\in W} \{I(\mathbf{q}) J(\mathbf{p}+\mathbf{q})\}}{\sqrt{\sum_{\mathbf{q}\in W} I^2(\mathbf{q})} \sqrt{\sum_{\mathbf{q}\in W} J^2(\mathbf{p}+\mathbf{q})}}, \tag{5}$$

ここで，

$$N(\mathbf{p}, W) = \sum_{\mathbf{q}\in W} \{I(\mathbf{q}) J(\mathbf{p}+\mathbf{q})\}, \tag{6}$$

$$Q_I(W) = \sum_{\mathbf{q}\in W} I^2(\mathbf{q}), \tag{7}$$

$$Q_J(\mathbf{p}, W) = \sum_{\mathbf{q}\in W} J^2(\mathbf{p}+\mathbf{q}). \tag{8}$$

とおくと，式 (5) は，

$$C_{NCC}(\mathbf{p}, W) = \frac{N(\mathbf{p}, W)}{\sqrt{Q_I(W)} \sqrt{Q_J(\mathbf{p}, W)}}, \tag{9}$$

となる．C_{NCC} が最大となる $\mathbf{p}$ を探索する場面において，Q_I は $\mathbf{p}$ によらない正の値であり，毎回計算する必要はない．しかし N，Q_J については，$\mathbf{p}$ が変化するたびに，すべて計算することとなり計算負荷が高い．ここで，テンプレートの矩形は基準画像 I の全点で走査するものであるが，**図 4** のように，W から移動し一部が重なる W' について考える．この図で $W \setminus W'$ は，W から W' を引いた差集合を示している．テンプレートの矩形領域が W であるときの N，Q_J の値をメモリ内に保存しておき，差分値のみを計算することで，新たな領域 W' における N，Q_J を算出できる．すなわち，

© 2010 Information Processing Society of Japan

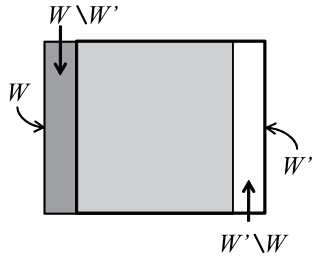

図 4　テンプレートを示す集合 W，W'
Fig. 4　Sets of templates W, W'.

$$N(\mathbf{p},W') = N(\mathbf{p},W) - N(\mathbf{p},W\setminus W') + N(\mathbf{p},W'\setminus W) \tag{10}$$

$$Q_J(\mathbf{p},W') = Q_J(\mathbf{p},W) - Q_J(\mathbf{p},W\setminus W') + Q_J(\mathbf{p},W'\setminus W) \tag{11}$$

となる．さらに図 4 における $W\setminus W'$，$W'\setminus W$ の縦長の領域に対して，縦の方向にも同様の差分計算ができることから，積和演算回数を $1/|W|$ に減らすことができる．

ZNCC における差分計算法は報告されていないが，NCC と同様に導入することが可能である．式 (2) を式 (9) にならい，

$$C_{ZNCC}(\mathbf{p},W) = \frac{M(\mathbf{p},W)}{\sqrt{D_I(W)}\sqrt{D_J(\mathbf{p},W)}}, \tag{12}$$

と記述すると，M，D_I，D_J はそれぞれ，

$$\begin{aligned} M(\mathbf{p},W) &= \sum_{\mathbf{q}\in W} I(\mathbf{q})J(\mathbf{p}+\mathbf{q},W) - |W|\bar{I}(W)\bar{J}(\mathbf{p},W), \\ &= N(\mathbf{p},W) - |W|\bar{I}(W)\bar{J}(\mathbf{p},W), \end{aligned} \tag{13}$$

$$\begin{aligned} D_I(W) &= \sum_{\mathbf{q}\in W} I^2(\mathbf{q}) - |W|\bar{I}^2(W), \\ &= Q_I(W) - |W|\bar{I}^2(W), \end{aligned} \tag{14}$$

$$\begin{aligned} D_J(\mathbf{p},W) &= \sum_{\mathbf{q}\in W} J^2(\mathbf{p}+\mathbf{q}) - |W|\bar{J}^2(\mathbf{p},W), \\ &= Q_J(\mathbf{p},W) - |W|\bar{J}^2(\mathbf{p},W), \end{aligned} \tag{15}$$

となる．この計算は分散，共分散の公式としてよく知られているものである．平均値 $\bar{I}$，$\bar{J}$ の計算は差分計算が可能であり，ZNCC の分子・分母の各要素 M，D_I，D_J は，すべて差分計算できる N，Q_I，Q_J，$\bar{I}$，$\bar{J}$ から算出できることが分かる．

4.3　差分計算の並列化

差分計算では，最初の 1 回目のみ N，Q_I，Q_J，$\bar{I}$，$\bar{J}$ の初期値を計算する必要があるが，2 回目以降の計算では演算回数が $1/|W|$ となり，大幅に高速化される．ただし，計算結果を次回のイテレーションの計算に用いることになるため，ループ伝搬依存があることになる．差分計算での並列化では，スレッド間の依存性をなくすため，各スレッドごとに初期値を計算する必要がある．

メニーコア・マルチコアプロセッサを用いる場合，多くのスレッド数に分割されることになるが，逆にスレッドが多いほど，初期値の計算にオーバヘッドを要することになる．また，N，Q_I，Q_J，$\bar{I}$，$\bar{J}$ の値を保存しておくためのメモリ領域が必要となる．この値は頻繁に利用・更新されるため，Cell では高速にアクセスできる SPU のローカルストレージ（LS）に置いておく必要がある．このメモリのサイズはあまり大きくないため，実装上ではテンプレートの大きさに上限を設けるなどの制限が必要になる．

4.4　census 変換法

census 変換法[4] は，中心のピクセルとの明度値の大小を 0，1 のビット列で表し，そのハミング距離を利用する方法である．明度変化や，オクルージョンに対してロバストであることが知られている．

具体的には，

$$\xi(\mathbf{p},\mathbf{p}';K) = \begin{cases} 1 & K(\mathbf{p}') < K(\mathbf{p}), \\ 0 & \text{othewise.} \end{cases} \tag{16}$$

$$C_{census}(\mathbf{p},W) = \underset{\mathbf{q}\in W}{\text{Hamming}}\left\{\xi(\mathbf{q},\mathbf{s};I),\xi(\mathbf{p}+\mathbf{q},\mathbf{s}+\mathbf{q};J)\right\} \tag{17}$$

となる．ただし，Hamming はハミング距離，$\mathbf{s}$ はテンプレートの中心座標を示すベクトルで，$\mathbf{s} = \mathbf{p} + (X/2, Y/2)$ である．大小比較とビット列の計算のみで実装できることから，積和演算を要する NCC や ZNCC と比べて高速であるとされている．ただし中心座標の値を使用するため，ZNCC のような差分計算は導入できない．

5.　高速化のアプローチとその効果

Xeon，Cell および Tesla に対して行う並列化・最適化による高速化アプローチと，その効果をそれぞれ述べる．なお，計算時間の計測にはすべて 3.2 節の画像データを用いている．この論文では式のように定義する高速化率を用いて性能評価を行う．

© 2010 Information Processing Society of Japan

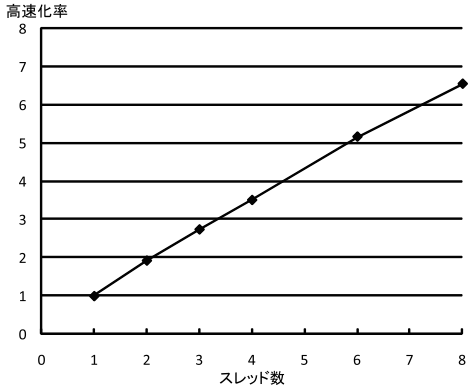

図 5　OpenMP による並列化の効果
Fig. 5　Effectiveness of parallelization using OpenMP.

$$(\text{高速化率}) = \frac{(\text{高速化手法適用前の実行時間})}{(\text{高速化手法適用後の実行時間})}. \tag{18}$$

5.1　Xeon における並列化，高速化

Xeon では最初に OpenMP を用いて全体の並列化を行い，その後に個別の処理のアルゴリズム改良や SIMD 化により高速化を行う．OpenMP による並列化による効果を**図 5** に示す．横軸は OpenMP で指定するスレッド数，縦軸は高速化率であり，ここでは (高速化率) = (1 スレッド使用時の実行時間)/(各スレッド数での実行時間) である．一般に OpenMP のような汎用ライブラリを用いるよりも，より詳細を精査しながら手動でスレッド化した方が，より高速になる可能性がある．しかし図 5 によると 8 コアを使用して約 6.5 倍に高速化されており，手動でスレッド化することによる改善の余地はあまり大きくないと判断し，OpenMP での並列化でとどめることとする．

OpenMP による並列化の次に，ZNCC の差分計算と，census 変換法の SIMD による最適化を行う．実行時間を**表 2** に示す．高速化率は，最適化前の実行時間を 1 とした場合に，どれだけ速くなったかを示す倍率である．ただしここでは OpenMP による並列化を行ったうえでの比較である．ZNCC については，差分計算を導入することで，大幅に高速化された．

census 変換法については，SIMD 化の方法を 2 種類試行した．表 2 の「SIMD 8 並列」は，16 bit ごとに大小を示す 0，1 を保持し 8 要素を並列に処理する実装である．しかし SIMD レジスタへのデータの並べ替えにオーバヘッドがかかり，高速化の効果は小さい．「SIMD 32 並列」は，大小を示す 0，1 をビット列化し，32 要素を並列に処理する実装である．nehalem コアから実装された SSE4.2 には，ビットカウントを行う POPCNT 命令が追加されており，XOR 命令（ビットごとの排他的論理和）と組み合わせることでビット列のハミング距離を計算できる．また，従来からある PMOVMSKB 命令により大小を示す 0，1 をビット列化することができる．これらの命令を用いて効率的な実装が可能となり，大きく性能が向上する．ただし，データの並べ替えや，SIMD レジスタと整数レジスタ間の転送などにオーバヘッドが発生するため，32 並列に対して 3.41 倍程度となる．

表 2　Xeon における相関計算の実行時間（8 スレッド使用時）
Table 2　Computation time of correlation on Xeon using eight threads.

相関計算	最適化	実行時間（秒）	高速化率
ZNCC	なし	40.1	1
	差分計算	4.79	8.37
census	なし	18.4	1
	SIMD 8 並列	15.3	1.20
	SIMD 32 並列	5.4	3.41

表 3　Cell における最適化手順とその効果
Table 3　Optimization process on Cell and it's effect.

最適化	実行時間（秒）	高速化率
なし（PPE のみ）	1,721.29	1
SPE での並列化	153.98	11.18
SIMD 化	135.87	12.67
DMA 転送の最適化	34.49	49.91
ZNCC 差分計算	30.18	57.03
前処理などを SPE 化	2.25	765.02

5.2　Cell における並列化，高速化

Cell（IBM PowerXCell 8i）に対して行った最適化手法と，その効果を以下に述べる．Cell において，一般的な処理を行うコアである PPE は，それほど高速なプロセッサではない．特にランダムアクセスは非常に遅い．データ並列処理に長けた SPE に処理を振り分け，高速なローカルストレージ（LS）へのアクセスとすることで，劇的に速度が向上する．ただし OpenMP のような汎用的な並列化手法を用いた場合には効率が良くないため，高い性能を達成するためには個別に実装する必要があり，Xeon と比べて多くの作業を要する．

今回の最適化は，相関計算として ZNCC を利用するプログラムを用いて，**表 3** に示している手順で実装している．表 3 は同時に各最適化の過程における実行時間と高速化率を示

© 2010 Information Processing Society of Japan

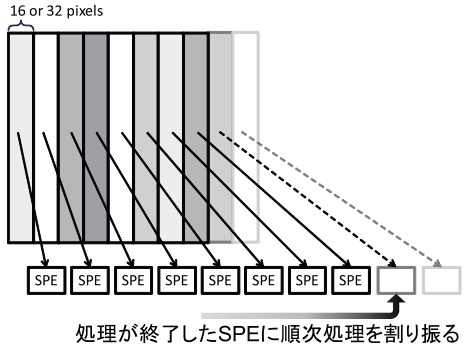

図 6　縦長のブロック分割による DMA 転送
Fig. 6　DMA transfer by portrait block splitting.

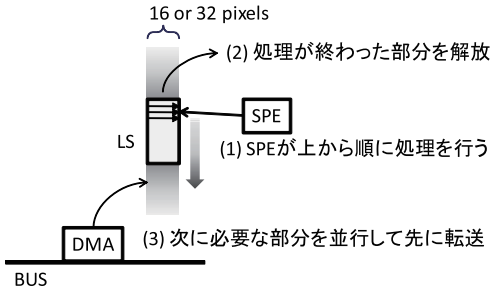

図 7　DMA 転送時間の隠蔽
Fig. 7　Latency hiding of DMA transfer.

しており，ここでの高速化率は PPE のみを用いた最適化前との比較である．最初に ZNCC の計算部分を PPE から SPE に振り分け並列化を行うことで，大きく性能が向上する．さらに，SPE での処理を SIMD 化するが，それほど性能の向上は見られない．これは DMA 転送時間がネックとなっている．

Cell における高速化では，LS への DMA 転送をいかに効率的に行うかが重要である．文献 5) ではラインごとに処理を分割し，結果を書き戻す際の転送処理の衝突をさけているが，ここではさらに DMA 転送時間の隠蔽と差分計算の実装のため，**図 6** に示すような画像を横幅 16 または 32 ピクセルの一定の幅のブロックで分割する方法を用いる．SPE での処理を**図 7** をもとに説明すると，

表 4　ZNCC 差分計算における SPE の実行時間（ミリ秒）
Table 4　Computation time at SPE of recursive correlation calculation.

処理内容	差分計算なし	差分計算あり
DMA 転送	41.33	100.99
ZNCC の計算	5,309.36	1,444.79

(1)　各ブロックの上のラインから順次処理を行う．
(2)　(1) で処理が終わった部分のメモリを解放する．
(3)　(1) の処理と並行して，(2) で空いた LS に次のラインの処理に必要となるデータを DMA 転送する．

という手順となる．これにより，DMA 転送が終了するまでの待ち合わせ時間が減少し，DMA 転送時間の隠蔽を効率良く行うことができる．横幅 16 または 32 については LS の容量 256 KB に合わせ，最適である値として決定する．この最適化により約 4 倍の高速化となる．これは上記の理由に加え，PPE からのタスクの発行回数を大幅に減少させられること，Cell で効率の悪い 1 byte や 4 byte 単位の転送から効率の良い 16 byte 単位の転送に変更できることによるものも含まれる．

さらに ZNCC に差分計算法を導入する．SPE 内部での実行時間は**表 4** のようになっており，計算時間は約 3.67 倍高速となる．ただし，図 6 のブロックごとに初期値を計算する必要があるため，Xeon の場合ほど高速化されない．ここで DMA 転送の待ち合わせ時間が増えているのは，計算時間が減ったため隠蔽できない時間が多く現れていることによる．

最後に相関計算以外の前処理などを SPE で並列化し，全体的な最適化を行うことで表 3 のように約 765 倍の高速化となる．ここでの前処理とは，相関演算に先立ちサブステージループにおける直前のイテレーションの結果を用い，相関が低い部分をエラーとして除外する処理や，線形補完法を用いてマッチングの範囲を絞り込む処理を，すべての画素に対して行うものである．相関演算部分と比べ負荷は小さいが，並列化の効果は大きい．

census 変換法については，ここまでで最適化された ZNCC のプログラムを一部変更することで実装し，最適化前と比べ約 414 倍の高速化となる．Xeon の場合と同じくビットカウント（spu_cntb）などの SIMD 命令を用いることで効率的な実装が可能である．

5.3　GPGPU を用いた並列化，高速化

GPGPU（Tesla）に対してステレオマッチングプログラムを移植・最適化した．その手法と効果を以下に述べる．参照画像を**図 8** のような細かい領域に分割して，それぞれを CUDA プログラミングにおける Block として割り当てて処理を行う．横幅は GPU での処理の効率

© 2010 Information Processing Society of Japan

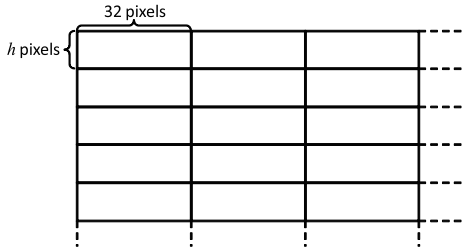

図 8　Tesla におけるブロック分割
Fig. 8　Block splitting on Tesla.

表 5　Tesla における最適化結果
Table 5　Optimization results on Tesla.

相関計算	最適化	Block の縦幅 h	1 Block のスレッド数	同時実行可能スレッド数	実行時間（秒）	高速化率
ZNCC	なし	4	128	384	13.96	1
	あり	6	192	384	3.26	4.28
	あり ＋ 差分計算	4	128	256	2.28	6.12
census	なし	4	128	384	6.86	1
	あり	12	384	384	2.45	2.80

が高まるように 32 とする．縦幅 h は後に述べる理由により最適な値を決定する．このようにすると，1 つのブロックにおいて $32 \times h$ のスレッドが実行されることになる．だたし 2.3 節で述べたように，同時実行可能なスレッド数は，使用するレジスタ数と Shared memory の量に依存する．使用レジスタ数はコンパイル時に決定され，複雑な処理であるほど多くのレジスタを消費する．同時実行可能なスレッド数が多いほど，演算コアの利用効率が上がり，より高い性能を示す傾向がある．さらに実際の性能は，複数の要因が複雑に絡み合うため，Block 内のスレッド数を事前に決定することは難しい．よって，縦幅 h を 1 から順に GPU のリソースの許す範囲まで変更し，実際の性能を計測することで，最も高い性能を示す値を求めることとする．

CUDA における最適化[9] は，高速なアクセスが可能な Shared memory の利用や，ループ展開などを行う．また ZNCC については最適化に加えて差分計算も実装する．CUDA に移植した時点で並列化されるが，そのうえでの最適化の効果を**表 5** に示す．この表での実行時間は，ホスト側（Z800）のメインメモリと Tesla のメモリ間のデータ転送にかかる時間約 0.75 秒を含んでいる．Block の縦幅 h は，最も高い性能が得られた値を選択している．

表 6　性能比較
Table 6　Performance comparison.

相関計算	ハードウェア	最適化前（秒）	最適化後（秒）	高速化率
ZNCC	Xeon（Z800）	236.38	4.79	37.51
	Cell（QS22）	1,721.29	2.25	765.02
	GPGPU（Tesla）	13.96	2.28	6.12
census	Xeon（Z800）	152.12	5.4	28.17
	Cell（QS22）	1,031.30	2.49	414.18
	GPGPU（Tesla）	6.86	2.45	2.8

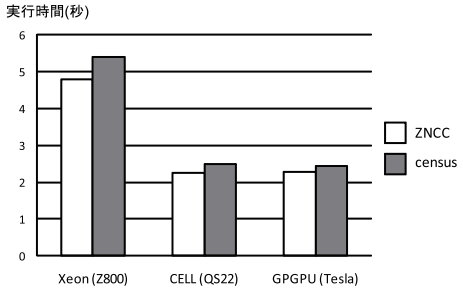

図 9　最適化後の実行時間
Fig. 9　Computation time with optimization.

1 Block のスレッド数と同時実行可能スレッド数より，今回は 1 から 3 の Block が 1 つの SM で同時実行されている．上述の最適化により，ZNCC で 4.28 倍，census で 2.8 倍の高速化となる．さらに ZNCC については差分計算を導入し 6.12 倍の高速化となる．しかし差分計算の導入前と比べて 1.43 倍程度であり，Cell での場合以上に高速化の効果が薄い．これは，初期値の計算が並列数に応じて多くなることと，レジスタの使用数が多くなり，同時実行可能なスレッド数が 384 から 256 に減少していることによる．

6. 考　　察

最終的な最適化結果をまとめると**表 6** になる．同一プロセッサでの最適化前と比較する高速化率においては，Cell が最も高い．GPGPU においては，CUDA を用いて移植した直後の最適化前の段階においてすでに並列化されており，高速化率自体は低い．最適化後の実行時間のグラフを**図 9** に示す．Cell を 2 基搭載した QS22 と，Tesla 1 基がほぼ同等の性能となり，Xeon を 2 基搭載した Z800 はその約半分程度という結果となる．

© 2010 Information Processing Society of Japan

表 7　最適化手法と結果のまとめ
Table 7　Summary of optimization approaches and results.

最適化手法	Xeon（Z800）	Cell（QS22）	GPGPU（Tesla）
並列化方法	OpenMP	手作業	CUDA
SIMD 化	あり	あり	なし
メモリ周辺の最適化	なし	DMA 転送時間の隠蔽	Shared Memory の使用
前処理部分の最適化	OpenMP で自動並列化	手作業で SPE での並列化	CUDA で並列化
結果	Xeon（Z800）	Cell（QS22）	GPGPU（Tesla）
ZNCC 実行時間	4.79 秒	2.25 秒	2.28 秒
差分計算の高速化率	8.37	3.67	1.43
作業工数	1 週間	4 週間	2 週間
最適化後のコード量	500 行	920 行	550 行

表 7 は，各最適化手法とその結果の各プロセッサにおける対応関係をまとめたものである．ここで作業工数と最適化後のコード量は，もともと 260 行である ZNCC のソースコードを最適化するのに要したおよその期間（工数）と，その最適化後のコード量を示している．

Xeon は最も低速であるが並列化に OpenMP を利用できる点から，工数という点で見ると最も少なかった．今回の問題に対し OpenMP による並列化は効率が良く，OpenMP は手軽な高速化手法として実用的かつ有用であるといえる．SIMD 化に関しては作り込みが必要になるが，今後の新しいプロセッサにおいても互換性が維持されることが予測される．

逆に Cell は最も工数が多かった．Cell では SPE による並列化をすべて手動で記述する必要があり，また最適化せずに PPE で動作させた場合に低速であることから，プログラムを全面的に最適化する必要がある．たとえば前処理部分の最適化において，Xeon では OpenMP，GPGPU では CUDA により他の処理と同時に並列化されたのに対し，Cell では個別に手作業で並列化することとなった．今回は相関計算の部分に焦点を当てて評価を行い，結果としてコア数に比例し Xeon の 2 倍強の性能となった．しかしプログラム全体で見た場合には，並列化できない（SPE 上で実行できない）部分が存在すると，PPE の性能が低い Cell の場合にはそこがオーバヘッドとなり，全体の性能を劣化させることになる点に注意する必要がある．Cell のようなヘテロジニアスマルチコアの普及のためには，より簡単に最適化できるよう環境整備が必要であると考える．

GPGPU については，工数は Xeon と Cell の中間程度でありながら，性能は Cell 2 基と同等で優れた結果となった．ただし GPGPU はまだ開発過渡期にあり，今回のように最適化したとしても，今後の新しいアーキテクチャやプログラミング環境での互換性が問題になる．OpenCL などの個別のアーキテクチャによらないプログラミング環境の取り組みが行われており，このような互換性の維持が今後 GPGPU を普及させていくうえでも重要であると考える．

表 8　最適化による誤差
Table 8　Errors by optimization.

相関計算	比較項目	Xeon（Z800）	Cell（QS22）	GPGPU（Tesla）
ZNCC	不一致点	1.29%	16.93%	7.34%
	±2 m 不一致点	0.17%	4.32%	1.69%
	誤差の平均	0.024 m	0.560 m	0.214 m
census	不一致点	0.45%	31.39%	6.86%
	±2 m 不一致点	0.01%	4.31%	1.17%
	誤差の平均	0.005 m	0.512 m	0.123 m

相関計算の方法について比較すると，最適化前においては census 変換法が高速であったが，最適化後ではわずかに ZNCC の方が高速となる．ZNCC は計算負荷が高いため，浮動小数点演算を用いない census 変換法が開発されたという経緯がある[4] が，最新のメニーコア・マルチコアプロセッサに対して，差分計算を導入し最適化した場合では逆転することが分かった．

各プロセッサでの最適化による誤差について述べる．標高分解能 1 m の DEM を生成し，誤差を計測した結果を**表 8** に示す．Xeon の最適化前のプログラムで生成した DEM を正解とし，「不一致点」は全体の画素数に対し標高が完全に一致しない点の割合，「±2 m 不一致点」は全体の画素数に対し標高が ±2 m より異なる点の割合，「誤差の平均」は標高差の絶対値の平均を示している．浮動小数点演算を多用する ZNCC の方が census と比べ誤差は大きくなる．Cell では SPU における浮動小数点演算全般で，GPGPU では積和演算において端数の切り捨てが行われるため，誤差は若干大きくなっている．Cell においては不一致点が多いが，±2 m を許容することで 4.3%程度となる．これは標高差 4,200 メートルの山地が対象データであることを考慮すれば，立体地図などの GIS アプリケーションでの利用においてまったく問題のないレベルに収まっていると考える．

7. ま と め

高速化が要求されている DEM 生成処理におけるステレオマッチングプログラムを，メニーコア・マルチコアプロセッサを用いて高速化し，その手法の違いと効果を検証した．この論文では特に相関計算に着目し，ZNCC と census 変換法の 2 種類を評価した．また ZNCC については高速な差分計算法を導入した．

© 2010 Information Processing Society of Japan

Xeon での高速化では，OpenMP を用い並列化し，高速な差分計算や SIMD 化による最適化を行った．Cell においては，低速な PPE から，処理を SPE へ割り振り並列化し，その際に DMA 転送が最適となるような分割を行った．GPGPU においては，同時実行されるスレッド数や使用するリソース量などが複雑に絡み，性能に影響を及ぼすため，最適となるように Block のサイズを適宜求め，最適化した．結果として Cell 2 基と GPGPU がほぼ互角，Xeon 2 基がその半分の性能となった．なおこれらの結果は，相関演算の計算式を簡易なものに変更するなどしない限り，これ以上の高速化は難しいと考えられるところまで最適化を行ったものである．

今後は GEO Grid [1] のサービスの一部として，高速な DEM 生成を提供するシステムの構築を行う．また，データ量が今回の 20 倍となる PRISM [10] などの高空間分解能の衛星画像の利用を想定したクラスタリングの検討も行う．そこにおいては画像を分割して処理することになるが，境界部分の精度が低下しないようオーバラップがどの程度必要かを検証し，データ量が増えた場合においてもスケーラブルに対応できるシステムの構築を目指す．

謝辞　census 変換法による高速化についてご助言いただいた日本原子力研究開発機構の武宮氏に感謝の意を表する．

参 考 文 献

1) Sekiguchi, S., Tanaka, Y., Kojima, I., Yamamoto, N., Yokoyama, S., Tanimura, Y., Nakamura, R., Iwao, K. and Tsuchida, S.: Design Principles and IT Overview of the GEO Grid, *IEEE Systems Journal*, Vol.2, No.3, pp.374–389 (2008).
2) Yamaguchi, Y., Kahle, B.A., Pniel, M., Tsu, H. and Kawakami, T.: Overview of Advanced Spaceborne Thermal Emission and Reflection Radiometer (ASTER), *IEEE Trans. Geosci. Remote Sens.*, Vol.36, No.4, pp.1062–1071 (1998).
3) Nieuwpoort, R.V. and Romein, J.W.: Using Many-Core Hardware to Correlate Radio Astronomy Signals, *Proc. ICS'09*, June 8–12, pp.440–449 (2009).
4) Zabih, R. and Woodfill, J.: Non-parametric Local Transforms for Computing Visual Correspondence, *Proc. ECCV'94*, pp.151–158 (1994).
5) 安田絹子，江口　剛，上田　孝，近藤伸宏，福田悦生，原　誠一，中村良介，田中良夫，関口智嗣：Cell/B.E. プロセッサによるステレオマッチングソフトウェアの高速化（最適化・高速化），情報処理学会研究報告，Vol.2007-HPC, No.59, pp.7–12 (2007).
6) Stefano, L.D., Marchionni, M. and Mattoccia, S.: A fast area-based stereo matching algorithm, *Image and Vision Computing*, Vol.22, No.12, pp.983–1005 (2004).
7) Faugeras, O., Hotz, B., Mathieu, H., Vieville, T., Zhang, Z., Fua, P., Thwron, E., Moll, L., Berry, G., Vuillemin, J., Bertin, P. and Proy, C.: Real time correlation-based stereo: Algorithm, Implementations and Applications, INRIA Technical Report, RR-2013 (1993).
8) Woo, D. and Park, D.: Rooftop Detection Based on 3D Line Data Using Fast Graph Search, *9th International Conference on Hybrid Intelligent Systems*, pp.442–446 (2009).
9) Harris, M.: Optimizing Parallel Reduction in CUDA (2007). http://developer.nvidia.com/
10) 度會英教，川西登音夫，大澤右二，松本暁洋，田殿武雄：パンクロマチック立体視センサ（PRISM），信学技報，SANE2006-64, pp.1–6 (2006).

(平成 22 年 1 月 26 日受付)
(平成 22 年 5 月 19 日採録)

岩田　健司

昭和 50 年生．平成 14 年岐阜大学大学院工学研究科電子情報システム工学専攻博士後期課程修了．同年（財）ソフトピアジャパン HOIP プロジェクト研究員．平成 17 年（独）産業技術総合研究所特別研究員．現在，同所情報技術研究部門サービスウェア研究グループ研究員．コンピュータビジョン，画像処理ミドルウェア，画像処理高速化に関する研究に従事．博士（工学）．電気学会優秀論文発表賞，SSII オーディエンス賞等受賞．電気学会，画像センシング技術研究会，日本医用画像工学会各会員．

中村　良介

昭和 43 年生．平成 8 年神戸大学大学院自然科学研究科環境科学専攻博士後期課程修了．同年神戸大学総合情報処理センター助手．平成 12 年（独）宇宙航空研究開発機構研究員．平成 16 年（独）産業技術総合研究所入所．現在，同所イノベーション推進室企画主幹，情報技術研究部門地球観測グリッド研究グループ研究員兼務．宇宙科学，リモートセンシングの研究に従事．博士（理学）．日本惑星科学会会員．

© 2010 Information Processing Society of Japan

田中　良夫（正会員）

昭和 40 年生．平成 7 年慶應義塾大学大学院理工学研究科後期博士課程単位取得退学．平成 8 年技術研究組合新情報処理開発機構入所．平成 12 年通産省電子技術総合研究所入所．平成 13 年 4 月より（独）産業技術総合研究所．現在，同所情報技術研究部門主幹研究員．博士（工学）．グリッドミドルウェアおよびグリッドセキュリティに関する研究に従事．平成 18 年度情報処理学会論文賞．平成 21 年度科学技術分野の文部科学大臣表彰科学技術賞（研究部門）．ACM 会員．

増田　知記

昭和 53 年生．平成 10 年バンタン電脳情報学院卒業．ゲームソフト開発会社，ソフトウェア開発会社を経て，平成 20 年（株）フィックスターズ入社．H.264 Encoder の開発等に従事．平成 21 年 5 月より（独）産業技術総合研究所へ派遣．

町田　亮介

昭和 52 年生．平成 18 年東京大学大学院理学系研究科地球惑星科学専攻博士課程修了．同年より東京大学 COE 特任研究員．平成 21 年（株）フィックスターズ入社．平成 21 年 5 月より（独）産業技術総合研究所へ派遣．惑星形成理論に関する研究に従事．博士（理学）．日本惑星科学会会員．

小島　　功（正会員）

昭和 33 年生．昭和 59 年京都大学大学院工学研究科情報工学専攻修了．同年通産省電子技術総合研究所入所．現在，（独）産業技術総合研究所情報技術研究部門サービスウェア研究グループ長．分散・並列データベースや e-サイエンス基盤に関する研究に従事．ACM 会員，Open Grid Forum および OASIS メンバ．

関口　智嗣（正会員）

昭和 34 年生．昭和 57 年東京大学理学部情報科学科卒業．昭和 59 年筑波大学大学院理工学研究科修了．同年通産省電子技術総合研究所入所．（独）産業技術総合研究所グリッド研究センター長を経て現在，同情報技術研究部門長．並列データ駆動計算機，ネットワーク数値ライブラリ Ninf，グリッドコンピューティング等に関する研究に従事．市村賞，文部科学大臣賞，情報処理学会論文賞等受賞．SIAM，IEEE 各会員，Open Grid Forum ボードメンバ，情報処理学会理事，日本応用数理学会評議員．

© 2010 Information Processing Society of Japan