Hadoopによる時系列画像分散データマイニングシステムの検討

—気象衛星画像への応用—

西前　光† 三好　智也†† 本田　理恵†††

† 高知大学大学院総合人間自然科学研究科　〒780-8520 高知県高知市曙町 2-5-1
†† 高知大学理学部応用理学科　〒780-8520 高知県高知市曙町 2-5-1
††† 高知大学理学部情報科学教室　〒780-8520 高知県高知市曙町 2-5-1
E-mail: †{nishimae,b103k299,honda}@is.kochi-u.ac.jp

あらまし　近年諸分野で大量の時系列画像が蓄積されるようになっている。時系列画像を用いた時空間データマイニングでは，多数のファイルから目的に応じた時空間断面を抽出して分析を行う必要があるため，分散処理フレームワークの利用が期待される。本研究では Hadoop，MapReduce を用いた “時空間データマイニング” の汎用システムの構築を目標として，時系列データの抽出，統計量の計算，相関分析など，必要なプロセスの実装と評価を通し，パフォーマンスのチューニングの指針を検討した。その結果，時系列画像に適した処理方法として，画像のブロックを分散化の単位として扱い，圧縮を取り入れることによって，Map-Reduce 間の伝送データを削減して計算速度を向上し，スケーラビリティを改善できることがわかった。また，実際の気象衛星画像を用いて雲量変化の時空間相関分析に適用し，その効果を確認した。

キーワード　Hadoop,MapReduce, 時系列画像, 相関分析, 気象画像, 分散データマイニング

1. はじめに

近年諸分野でテラバイト級の大量データ，いわゆるビッグデータが蓄積されるようになり，こうしたデータから新しい知識を発見しようとするデータマイニングの研究が進展している。ビッグデータの中でも特に利用・検討が進んでいるのは，ソーシャルデータやインターネット上の情報，e-コマース等の分野であるため，対象とされるデータの形式はテキストや表である事が多い。一方，気象学，天文学，シミュレーションなどの科学分野，監視カメラなどによるセキュリティの分野では，時間とともに変動する画像，時系列画像が大量に蓄積される。時系列画像は多数のファイル群から成り，その中には様々な時間空間の変動パターンが含まれるため，様々な時間，空間断面で必要な情報を抽出して，解析を行う事が必要になる。このような時系列画像の大規模データアーカイブに対する汎用的な知識発見システムが実現されれば，その有用性は高いと考えられる。

こうしたビッグデータの解析環境として，分散処理が注目されている。時系列画像が多数のファイルからなり，ここから時間，空間等の様々な断面のデータを抽出して，大量に処理しなければならない事を考慮すると，ファイル入出力がボトルネックとなると予想されるため，特に分散処理の利用が有望であると考えられる。

大規模データの分散処理のミドルウェアとしては，Apache Hadoop (以下 Hadoop) [1] ,Gfarm [2] 等が知られているが，特に Hadoop は，分散処理を実現する MapReduce と，分散ファイルシステムである Hadoop Distributed File System(以下 HDFS) から構成されており，さらに機械学習のアルゴリズムをあつめた Mahout を備えることから，ビッグデータのマイニングのフレームワークとして有用であると考えられる。ただし，MapReduce では，すべてのデータを <key,value> の組として，Map，Reduce の 2 つのフェーズで処理を行うことにより効率的な並列処理を行うため，テキストや表などの単純なデータ形式に対する親和性は高いが，画像などのデータに適用するには考慮が必要と考えられる。

近年，大規模画像集合に対しても Hadoop による分散処理を利用しようとする試みが始まっている。Almeer(2012) [3] は，110 core のクラスタマシンに Hadoop を実装し，8 種類のリモートセンシングの画像処理アルゴリズムを実装して，そのスケーラビリティを検証した。白崎ら (2012) [4] はバーチャル天文台システムを想定して，Hadoop を利用したスケーラブルな並列データ検索・解析システムを試験構築し，性能試験を実施した。しかし，いずれも，まずは大量画像処理や検索を効率的に行う事に重点をおいたものになっている。一方我々は時空間データマイニングという観点から植生指標画像からの時系列画像の抽出とロジスティック関数のモデリングの問題に Hadoop, MapReduce を適用したが，時系列抽出時にスレーブノード数に対してスケーラビリティが確保できないという問題が生じていた [5]。また，このような特化した問題だけでなく，時系列画像からの時空間分析という様々な分野で現れる問題に適用できる，より汎用性のあるシステムの検討も必要である。

本研究では，時系列画像から時空間の知識発見を支援する汎用的な分散処理システムを，Hadoop, MapReduce を用いて構

築することを検討した。特に時系列抽出とその分析に関わる部分をとりあげ，この際，大学の教育システム等で一般的な数十台程度のスレーブマシンからなる環境を想定して，このような状況でパフォーマンスとスケーラビリティを最大化するための指針を得る事を目標とした。また実際に気象衛星画像からの時間変化の相関性分析にこのシステムを応用した。

第 2 章では，まず Hadoop，MapReduce について紹介し，第 3 章で想定する時系列画像に対する分散データマイニングシステムの概要を示す。第 4 章では，数十台程度のスレーブ環境でのパフォーマンスのチューニングを示し，第 5 章で気象衛星画像からの時空間相関分析への適用例を示す。最後に全体のまとめと今後の課題について述べる。

2. Hadoop, MapReduce

Apache Hadoop Project は信頼性の高いスケーラブルな分散コンピューティングのための Open Source Software であり，Hadoop common を中心とし，分散処理システムである MapReduce，分散ファイルシステムである Hadoop Distributed File System(以下 HDFS)，機械学習ライブラリである Mahout 等様々なプロジェクトから構成されている [1]。

HDFS は Google 社の The Google File System [6] に触発されて開発された。ストリーミング型のデータアクセスによって，非常に大きなファイルを保存するために設計されている。ファイルはあるブロックサイズ (通常 64MB) で分割されて複数のノードに分配され，それぞれ独立した単位として保存される。またこの時，耐障害性と可用性を上げる為，同一のデータを複製して複数のノードに分散して保存 (レプリケーション) する。

MapReduce は Google 社から発表されたプログラミングモデル [7] を参考にしてオープンソースソフトウェアとして開発されている。MapReduce における分散処理は，図 1 のように処理を Map，Reduce の 2 つの段階に分けて実行される。Map では入力データをスレーブノードに分配し，それぞれの Map 関数を実行する。その際，Map への入力は，指定したブロックサイズで分割されたスプリットデータを保存している計算機で計算するよう調整している。ここで中間出力として <key,value> を取得し，それぞれ同じ key ごとに value を集約，ソートを行って，Reduce に受け渡す。Reduce では key で集約された value 集合を取得し，key 毎に Reduce 関数を実行する。従って，実際に MapReduce でデータ抽出や分析を実装する場合には，key, value の選択やその形式，Map, Reduce 間でやり取りされるの形式がパフォーマンスに大きな影響を与えることが予想される。

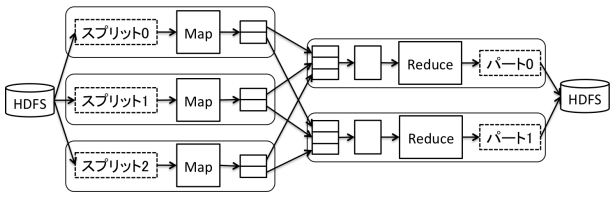

図 1　MapReduce の処理方法

3. 想定するシステムの構成

3.1 システムの概要

まず，対象とする時系列画像の分散データマイニングシステムの概念図を図 2 に示す。数台のマスターと数十台のスレーブノードから成る Hadoop システムで分散ストレージと分散処理環境を構築するものとする。マスターには分散処理を制御する JobTracker と仮想ファイルシステム用の NameNode, Secondary NameNode が必要である。物理的には 1 台にまとめてもよいが，使用する計算機の能力と問題に応じて分離を検討する必要が有る。これらのノードは同一のスイッチに接続されているものとし，リモート環境にあるノードについては考慮しない事にする。

入力データは，同一の空間領域について等時間間隔で取得された時系列画像で，時間の情報は画像ファイル名に含まれており，HBase などのデータベースは使用しないものとする。これらの画像群は Hadoop で構築したシステムの NameNode で管理される仮想ファイルシステム HDFS に一括して格納される。このシステム上で JobTracker 経由で MapReduce で実装したタスクを実行する事により，様々なデータ断面の抽出や時空間変動パターンの分析をおこなう。

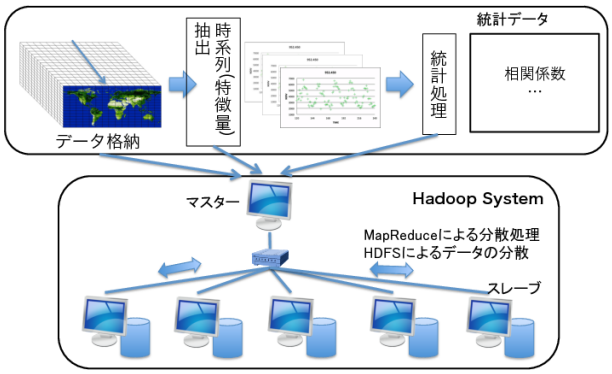

図 2　システムの概要

汎用的なシステムの構築を視野に入れ，以下のような基本的な処理を想定する。

- 全点に対する時系列データの抽出
- 時系列データからの平均，相関係数など統計量の計算
- 時系列データの相関分析
- 時系列データのモデリング
- 時系列データのクラスタリング
- 異なる時間の画像の類似性の分析
- 空間ブロックごとのテクスチャによるクラスタリング

こうした処理を組み合わせた高次処理も存在しうるが，まず時系列画像からのデータ抽出・処理におけるパフォーマンスの調査やチューニングにおける最も基本的なタスクとして，全点に対する時系列データの抽出とその平均・分散の計算をとりあげるものとする。

3.2 時系列データの抽出

時系列画像からの特徴量抽出の MapReduce による実装方法を表す。時刻 t の画像 における座標 (x, y) の画像の階調値を $I(x, y, t)$ とおく。時系列画像の解析を MapReduce で柔軟に実施するためには， この 4 つの値を key または value として設定する必要がある。ただし，どの値を key, value のいずれにどのような形で実装するかは，処理に応じて適切な方法を選ばなければならない。

図 3 に，時系列抽出を対象として，画素ごとに分散化を行う最も単純な実装例を示す。Map フェーズでは，空間座標 (x,y) を key，(時刻 t, 階調値 $I(x, y)$) を value としてスプリットファイルを作成して出力する。Shuffle では座標 (x,y) を用いてスプリットファイルをまとめ，Reduce フェーズでは，座標ごとに抽出した時系列から平均値や分散を計算し，座標 (x, y) を key，平均値などの統計量を value として出力する。この実装形式を “画素毎分散” と呼び，この実装形式についてまず予備的な実験を行った。なお Map から Reduce へのデータの分配を行う Partitioner については，一つ一つの時系列の Reduce での処理時間に差がないことから，key に対するハッシュ関数を用いるデフォルトの Partitioner ではなく，強制的に等分割する Partitioner を作成して用いた [5]。

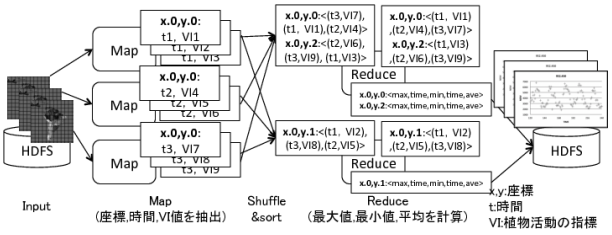

図 3　時系列データの特徴量計算の MapReduce での実装概要

4. 実験 -パフォーマンスのチューニング-

4.1 実験に用いたシステム

実験には，高知大学情報科学教室教育システムの iMac 53 台 (マスター 3 台，スレーブ 50 台) を利用した。使用したマスター，スレーブの性能を表 1 に示す。ここでマスターノードを 1 台にまとめると NameNode としてのメモリ容量が不足であったので，NameNode，Secondary NameNode，JobTracker は 3 つの独立のノードに分散させた。マスター，スレーブノードとも Mac OS X 10.6.8 をネットブートで起動する設定とし，ユーザ管理は LDAP で行った。このため，Hadoop アカウントを LDAP で作成して Hadoop システムデータを NFS に格納することで，管理・設定を簡単化する事ができた。一方，各計算機の未使用のローカルストレージ (1 台あたり約 455GB) を HDFS 領域として利用する事により，ファイルアクセス・処理の効率的分散化を行い，同時に約 23TB の大容量を確保した。

また実験に用いたシステムのネットワーク構成を図 4 に示す。各計算機は，2 台の L2 スイッチにスター型に接続されており，スイッチ間は 10Gbps で接続されている。

表 1　実験に用いる計算機 (マスター，スレーブ共通) の性能

諸元	値
台数	53 台
プロセッサ	Intel Core 2 Duo (3.06GHz)
コア数	2
メモリ	4GB
HDD	500GB
HDFS 用 HDD 容量	455GB(全システムで 23TB)
OS	Mac OS X 10.6.8
Hadoop version	1.2.1
Java version	1.6.0_45
ネットワーク	1000BaseT

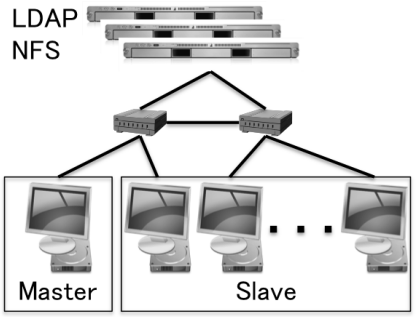

図 4　実験に用いたシステムのネットワーク構成

4.2 予 備 実 験

本章での実験には，地球観測衛星によって作成された GIMMS [8] と呼ばれる植生指標画像から海を含む南米領域の画像を 1200 枚 (時間方向のサンプリング点 1200) 用いた。 画像サイズは 1152 × 1152 pixel, 各画素の階調は 16bit である。まず図 3 の形式 (画素毎分散) で時系列抽出と統計量計算を実装し，ノード数を 1 から 50 台まで変化させた場合の計算時間を調べた。この際，Map から Reduce へ送信されるデータ型は，可変長バイナリの VIntWritable と VIntArrayWritable を使用した。

図 5 にスレーブノード数に対する計算時間の変化を示す。この予備実験では，全体の計算時間がスレーブノード 10 台程度で頭打ちしてしまっている。しかし，Map フェーズの所要時間だけをとりだすとスレーブノード数に対応して減少しているようにみえる。

ここで分散処理の効率化を定量化するために，計算速度向上比を以下のように定義する。

$$I(n) = t(1)/t(n). \tag{1}$$

$t(n)$ は n ノードでの計算時間で，$I(n)$ は n スレーブを用いた場合の 1 ノードに対する速度向上比である。理想的な極限では $I(n)$ は n となる。

図 6 にノード数 1-50 の性能速度向上比を示す。この結果より，全体の計算速度向上比は 10 ノード程度で頭打ちしてしまっているが，Map フェーズだけに着目すると，計算速度向上比はノード数に比例して増加し，分散化による理想的な計算速度の向上がほぼ達成されていることがわかる。よって Map から Reduce フェーズへ至る過程以降で頭打ちが発生しているもの

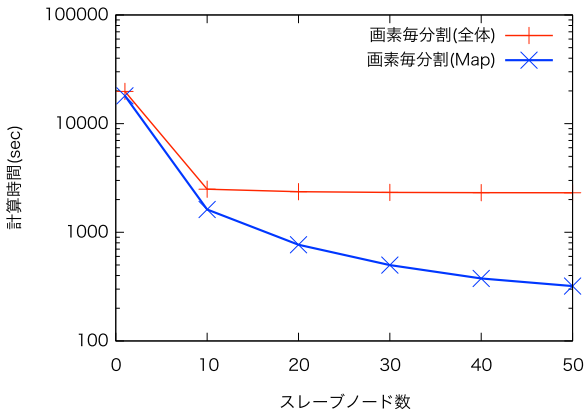

図 5　計算時間のノード数に対する変化。赤線は全体，青線は Map のみの処理時間

と予想される。このボトルネックを解決する方法について次節で検討する。

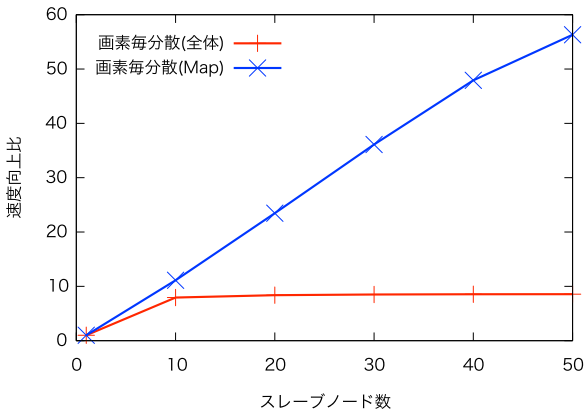

図 6　計算速度向上比のノード数に対する変化。赤線は全体，青線は Map のみの処理時間

4.3　Map-Reduce 間のデータ量削減

ここで図 3 の手法を再考すると，画像の画素は規則的に並んでいるにも関わらず各画素が独立であるかのように扱い，座標や時間などの情報を画素ごとに繰り返し送信した事で，Map-Reduce 間のデータの流通が増大し計算効率が上がらなかったと考えられる。画素の並びの規則性を使用して，図 7 のように分散の単位を画像内のブロックのようなより大きい構造にして，冗長な情報をまとめることができれば，Map-Reduce 間のデータ伝送のボトルネックを解消して計算速度を向上させてスケーリング効果を回復できることが期待できる。

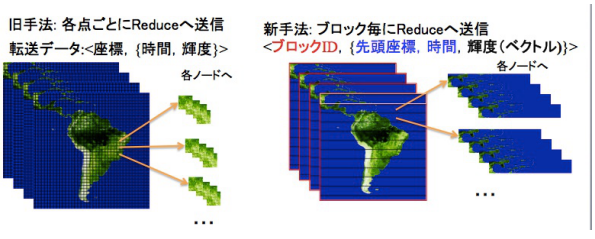

図 7　画素毎分散とブロック分散の比較図

そこで Map，Reduce 間のデータ量を削減するために，以下の手法の性能を比較検討した。

（1）　1 点毎の分散 (画素毎分散)
（2）　ブロック毎分散（圧縮無)
（3）　ブロック毎分散（圧縮有)

(1) では比較のため予備実験同様に画素毎に分散を行う。(2) では画像をブロック化して領域の ID “i” を key とし，さらにその画像の取得時刻 t，ブロック i の始点の座標 (x_i, y_i)，さらにベクトル化した画像の諧調値 $\{I(x_i, y_i, t), I(x_i + 1, y_i, t), I(x_i + 2, y_i, t), \cdots\}$ を value に割り当てることで，大幅なデータ削減を行う。ブロックの数をスレーブノードの数に一致させて，Partitioner で各ブロックを異なるノードに割り当てる事によって分散化の効率を制御する。また，(2) では，key1 つあたりのデータサイズが増加したので，中間データの転送時の圧縮が効果が期待できる。よって (3) では Hadoop 標準の DEFLATE アルゴリズムを用いてデータ圧縮後，Reduce へデータ伝送するものとする。

なお，この他に，Map-Reduce 間のデータ削減に寄与する手法として Combiner の利用がある。Combiner は，Reduce タスクで行う処理の一部を Shuffle 直前に行う事で中間データのデータ量を削減し，Shuffle 時の負荷を削減することができるが，ここでは検討対象からはずした。

以上の効果の検証のため，表 2 の実験ケースを設定した。データ型はいずれも VIntWritable, または VIntArrayWritable を使用した。各実験ケースに対して，ノード数 1，10，20，30，40，50 で実験を行った。前章同様，レプリケーションは 3，準備した全時系列画像 (1152×1152 pixel, 16bit 画像 1200 枚) に対して時系列抽出と最大値等の統計量の計算を行った。

表 2　Map-Reduce 間のデータ削減の実験ケース

case	key	value	圧縮
1	座標	時間，階調 (画素)	無
2	ブロック ID	時間，始点の座標，階調値 (ブロック)	無
3	ブロック ID	時間，始点の座標，階調値 (ブロック)	有

なお，case2, case3 では，スレーブノード 20 台以上ではブロックの数とスレーブノード数を等しくして，ブロック単位でデータをノードに割り当てた。しかし，スレーブノード 1 台，10 台の場合は，ブロック数とノード数を等しくするとブロックサイズが大きくなり過ぎて計算時間が急激に大きくなることを予備実験で観測した（詳細 4.5 節参照）ため，ブロック数を一律 20 として，各ブロックとノードの対応付けの制御は行わないこととした。

4.4　実 験 結 果

まず表 3 に各ケースにおいて Map から Reduce へ伝送されたデータの総量の観測値を示す。ブロック毎に分散した場合，画素毎に分散したケースに比べて，約 5 分の 1 のデータ削減となった。さらに圧縮によって，その 7 分の 1 程度のデータ量に削減できた。case1 から case3 では約 35 分の 1 ものデータ量削減となった。なおこの削減 (圧縮効率) は画像の特徴によって変

化すると考えられるため，実際には使用する典型的な画像セット毎に予備調査を行う必要が有る。

表 3　Map-Reduce 間のデータ削減の実験結果:Map から Reduce への伝送データ量

case	Map-Reduce 間のデータ量 (MB)
1: データ毎の分散	24888
2: ブロック分散 (圧縮無)	4772
3: ブロック分散 (圧縮有)	712-714

次に，図 8 に各ケースの計算時間のスレーブノード数に対する変化をまとめる。1〜10 台では，ブロック分散の計算時間は画素毎分散の場合の 3 分の 1 程度となる。また，画素毎の分散の場合，圧縮無しでは 20 ノード程度で頭打ちが生じたが，圧縮を行ったケースではその後も 50 台までノード数の増加に応じて計算時間が減少した。最終的にノード数 50 台の計算時間は，case1 で 2311 sec ，case2 で 478 sec であったのに対し，case3 では 126 sec となった。

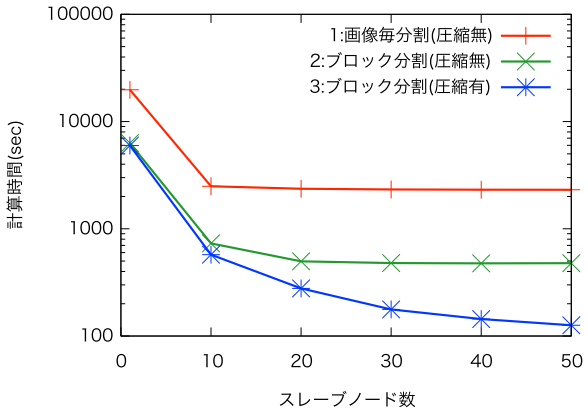

図 8　Map-Reduce 間のデータ削減時のノード数に対する計算時間の変化

図 9 には計算速度向上比の変化を示す。この結果から case1, case2 ではそれぞれ 10 ノード，20 ノードで計算速度向上が見られなくなったのに比べ，case3(ブロック分散，圧縮有) では，スレーブノード数に対してほぼ線形増加する性能速度向上比を実現できたことがわかる。なお，ブロック数が増加するにつれて，ノード数を大きくしたときの圧縮効率と Map の速度向上比がやや下がる傾向がみられた。扱う画像のサイズに応じてブロック数に関する調整が必要である可能性がある。

4.5　最適ブロック数の評価

次章の実データを用いた実験ではここまでの結果に基づいて case3 のブロック分散圧縮有のケースを採用するが，前節で述べた通り，ブロック数の設定によっては計算時間が増加するケースが見られた。よって最適なブロック（分散）数についてさらに系統的な調査を行った。

図 10 には，スレーブノード数 50 に対し，ブロック数を 50 から 250 まで，入力画像を 1200 枚 (総データ量 3.2GB) から 4800 枚 (総データ量 12.8GB) まで変化させて実行した時の計算時間を示す。図 10 より，総データ量が小さい時 (画像枚数

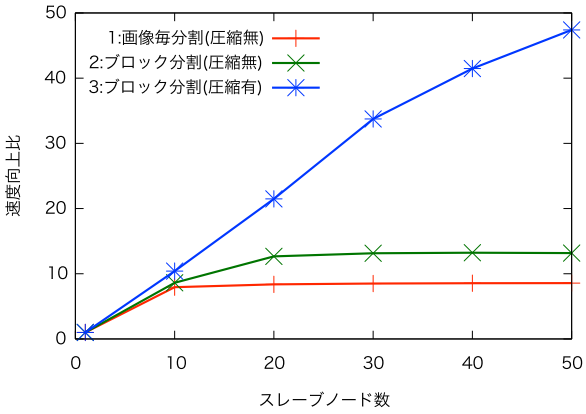

図 9　Map-Reduce 間データ削減時のノード数に対する計算速度向上比の変化

1200, 2400) は，ブロック数が小さいほど計算時間が短くなり，最適なブロック数は基本的にはスレーブ数であることがわかる。しかしさらにデータ量が増えると画像枚数 3600, 4800 のとき 50 ブロックで計算時間の増加が起こった。なお，画像枚数 4800 枚のケースで Reduce の処理を空にして計算時間を求めたところ，ブロック数 50 における計算時間の上昇はみられなくなったので，この第 2 のボトルネックはスレーブノードのメモリ容量などの能力の限界によって Reduce 内の処理で生じていると考えられる。

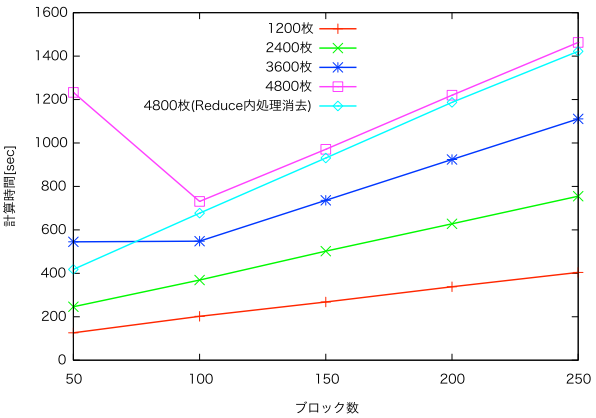

図 10　スレーブノード数 50 でのブロック数（分散数）と計算時間の関係。画像枚数は 1200，2400，3600，4800 として，4800 のときのみ Reduce 内の処理を空にしたケースを併記。

この第 2 のボトルネックが発生する条件を調べるため，図 10 の各実験ケースに対して，Map-Reduce 間の伝送データ量 (圧縮解凍後）を 1 ブロックあたりの値に換算した結果を図 11 に示す。この図より，現在の実験条件で計算時間の急上昇がおこるのは 1 ブロックあたりのデータ量が 200–260MB 程度をこえた時点と考えられる。なお，取り扱う画像のサイズやビット長をかえてもこの閾値はほぼ不変であった。よって，前述の通り，この閾値は Reduce 内の処理とメモリ容量などのスレーブノードの能力に依存すると考えられるが，マシンスペックと閾値の関係が不明のため，使用する環境であらかじめ調査して把握し

た上で大規模処理を行うことが必要であろう。

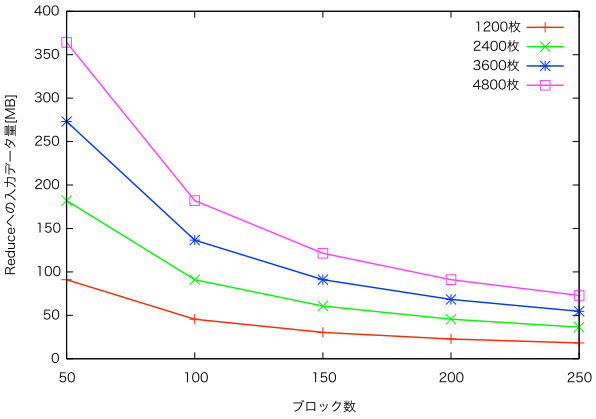

図 11　図 10 の実験における Map-Reduce 間の 1 ブロックあたりのデータ伝送量 (DEFLATE による圧縮解凍後)

5. 実データへの応用:気象衛星画像の時空間相関分析

5.1 手　　法

構築したシステムの実データへの適用例として，気象衛星画像からの時空間相関分析を実施する。先行研究である坂口，本田 (2009) [9] では大量の気象画像に対する時空間の相関性分析を行ってその結果を可視化し，インタラクティブな発見支援を行うシステムについての検討を行った。しかし，あらかじめ時空間データを PostgreSQL のラージオブジェクトに格納する方針をとったため，画像枚数が増加に応じて処理時間が極端に大きくなってしまっていた。この問題の解決を Hadoop による分散処理によって試みる。

図 12 に検討する問題の概念図を示す。まず画像の階調値の時間ウィンドウ w の時間変動を調査して，あらかじめ特徴的な基準時系列 $V_s = \{I(x, y, t_s + i) \mid i = 0, 1, \cdots, w-1\}$ が得られているものとする。この基準時系列と相関性の高い変動がおこっている位置ずれ，時刻ずれのある，座標 $(x', y') = (x + dx, y + dy)$，時間 $t' = t + dt$ の参照点 $V_r = \{I(x', y', t_s + dt + i \mid i = 0, 1, \cdots, w-1\}$ を求めるものとする。

相関性の指標としては相関係数 r を用い，以下の式で求める。

$$r = \frac{\sum_{i=0}^{w-1}(V_s(i) - \tilde{V_s})(V_r(i) - \tilde{V_r})}{\sqrt{\sum_{i=0}^{w-1}(V_s(i) - \tilde{V_s})^2}\sqrt{\sum_{i=0}^{w-1}(V_r(i) - \tilde{V_r})^2}}. \qquad (2)$$

ここで $\tilde{V_s}$，$\tilde{V_r}$ はそれぞれ V_s，V_r の平均値とする。相関係数は $-1 \leqq r \leqq 1$ の範囲の値となるが，正の相関だけでなく負の相関も同様に重要であるため，正負に限らず相関係数の絶対値が大きい領域に着目するものとする。これによって “A 地点で起こった時間変動を持つ現象が dt 時間後 (前) に B 地点でおこる”，といったタイプの知識発見をめざすものとする。

MapReduce での実装は下記のような方法で行った。

（1） 基準点の座標, 時刻，ウィンドウサイズを設定

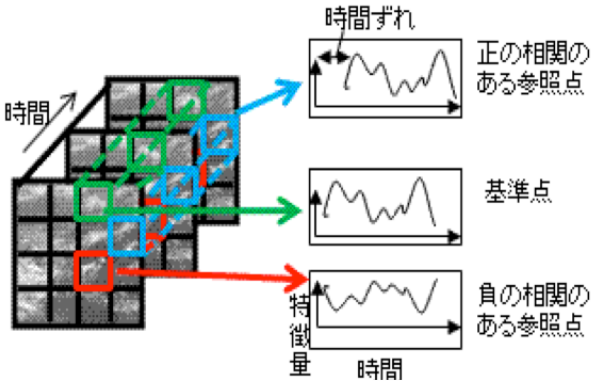

図 12　気象衛星画像からの時空間相関分析の概念図 [9]

（2） 各地点の時系列データを抽出してファイルに保存

（3） 基準時系列 V_s を抽出

（4） 各点について参照用部分時系列 V_r を抽出し，基準点との相関係数計算

（5） 部分時系列の始まり時間をずらして上記を繰り返し

（6） 相関性の高い領域，時間を可視化等によって分析

前章で述べた時系列の統計量の計算では各点の時系列をそのまま Reduce 処理で計算していたが，今回は部分時系列の取得過程が煩雑になるのをさけるため，時系列データをファイルとして一旦出力するようにアルゴリズムを変更した。よって，(2) は，1 段目の Map, Reduce で，(4), (5) のプロセスは，この中間出力ファイルを入力とする 2 段めの Map のみで実現した。なお，(3) で設定した基準時系列は 2 段目の Map において distributed cash として全ノードに配布した。

5.2 実 験 手 法

実データを用いた実験には，運輸多目的衛星 MTSAT-1,2(通称ひまわり 6,7 号) の画像を用いた。MTSAT-1 は東経 140 度，MTSAT-2 は東経 145 度の赤道上の静止軌道に投入され，1 時間毎に半球の可視画像 (VIS)，赤外線 4(IR1, IR2, IR3, IR4) バンドを観測している。雲の消長の時空間の変動パターンを求めるため，この中から IR1 バンド (10.3〜11.3um) を選択し，2012 年の 9 月から 12 月末までの 2914 枚を評価することにした。

ここでは高知大学気象情報頁 [10] でアーカイブされている北緯 70 度から南緯 70 度，東経 70 度から西経 150 度の領域を緯度，経度座標系にマッピングされた画像を用いた。空間解像度は 0.25° /pixel，画像サイズは 560×560 pixels である。実験では，雲塊がある程度の広がりを持つ事を考慮して，5×5pixel のブロック毎に階調値の平均をとり，この値を利用した。平均化後の画像サイズは 112 × 112pixel まで削減されることになる。

今回の実験のタスクは，東太平洋における熱帯収束帯の雲の消長の時空間相関性の発見とした。熱帯収束帯とは赤道付近で南北半球からの貿易風が収束する帯状の領域であり [11]，東西に延びた帯状の低気圧が発生し，その中で周期的に雲塊が発生・消滅する。この特徴を相関分析によって定量化する事を目標とする。

図 14 に基準点として選んだ地点とその周辺での雲分布の消長

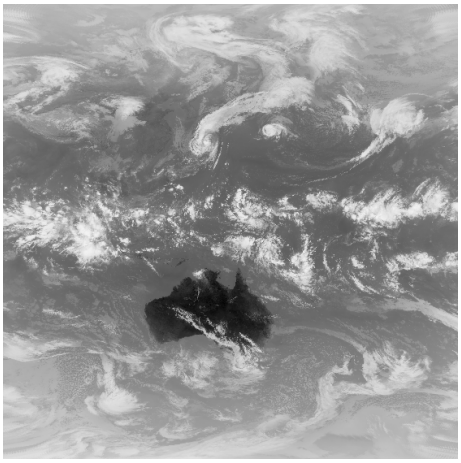

図 13　MTSAT-1 が撮影した 2012 年 10 月 18 日 3 時の IR1 の気象画像

を示す。基準点は (東経 176.25°, 北緯 7.5°)，画像上で（425, 250) の点とし，区間は 2012 年 10 月 11 日 0 時 (GMT) を始点とする 10 日間である。図 14 から，この時期，この領域では，ほぼ 5 日間の周期で帯状の雲が発生し，雲塊へ分解，消滅を繰り返す様子を観察できる。よって，相関解析のウィンドウは 2 週間，サンプリング間隔は 1 時間とした。時系列データの点数としては 240 点となる。

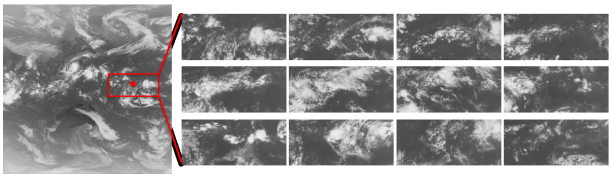

図 14　2012 年 10 月 10 日 0 時 (GMT)MTSAT-2 画像と，赤い矩形領域内の画像の 20 時間おきの時間変化。赤丸が基準点。

5.3　結果と考察

上記の基準点に対して，4ヶ月分の画像で相関分析を行い，特徴的であった時間帯の相関係数の空間分布を図 15 に示す。この図の左には参照点の始点の画像（2012 年 10 月 13 日 2 時，基準点より約 3 日後），右には相関係数のカラーマップを示す。カラーマップでは，正の相関の強い領域を赤，負の相関が強い領域を青で示し，特に相関係数-0.5 以下，0.5 以上の箇所を青線，赤線で囲んでいる。この結果からこの時間帯 (基準時系列の 3 日後）に基準点からやや南側に帯状に正の相関，負の相関の高い箇所が対になって現れていることがわかる。

図 16 には，基準点と図 15 で示された正負の相関性の高い領域の中心の時系列を示す。基準点を緑色，正の相関が強い点を赤色，負の相関が強い点を青色で表す。基準点では 1 週間程度の周期的な変動が観測されるのに対し，やや同期，あるいは 180 度程度の位相ずれをもって参照点の時系列が変動している様子が見られる。結果の有用性については，専門家の評価が必要であり，また可視化のインターフェースについても充実させる必要が有るが，MapReduce での実装と Hadoop での実行によってこうした問題が分散処理によって効率的に実施できる事

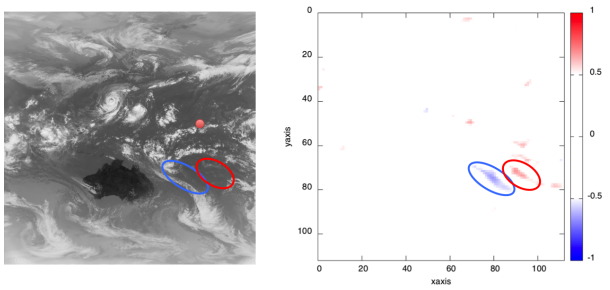

図 15　相関係数の計算結果例。左側は時系列始点の画像（2012 年 10 月 13 日 02 時，基準点より 3 日後，赤丸は基準点の位置），右は基準時系列との相関係数。カラーバーは 青が相関係数-1, 赤が 1 である。

は確認できたといえる。

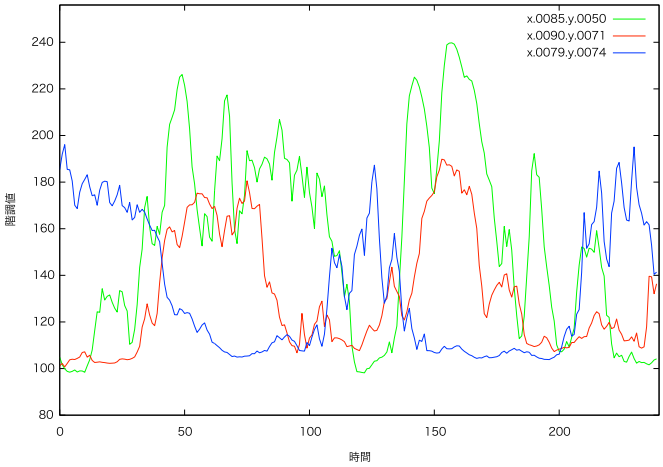

図 16　相関を調べた地点と図 15 の青，赤の囲み領域中心の時系列.

さらにデータ量に対するスケーラビリティが確保されているかを確認するため，8ヶ月 (5853 枚)，12ヶ月 (8734 枚)，24ヶ月 (17444 枚) での 50 スレーブでの実験を追加実施した。図 17 には処理した画像枚数と計算時間の関係を時系列抽出，相関係数計算のそれぞれについて示す。いずれも処理枚数に比例して計算時間が線形的に増加しており，この規模 (全データ 5.5GB, ブロック毎で最大 110MB, さらに 5x5 ビニング後 4.4MB) ではデータ量に対して頭打ち等を起こす事無く処理できていることがわかる。1 枚あたりの画像サイズが大きくなると，4.4，4.5 で議論した 2 種類のボトルネックによって計算時間の増加が起こる可能性があるが，ブロックサイズや分散数，さらに取り扱うデータセットの分割を適切に調整する事によって，そのような問題を回避できると考えられる。

より有用な結果を抽出するには，前処理として空間平均に加えて時系列データの時間方向の移動平均を行い，そのウィンドウサイズを試行錯誤できる事が必要であろう。このような処理は時空間の変動パターンの解析に普遍的に必要な要素でもある。現在は空間平均のみ 1 段目の Map に実装しているが，試行錯誤的に実施するためには，空間平均，時間平均とも第 1 段の Map-Redue 後の時系列中間ファイル生成後の処理として取り

入れる事が適当と考えられる。このように，付加処理についてはその内容や実施形態 (一方向か，試行錯誤か）によって Map, Reduce のどの段階に入れるべきかが異なってくるので注意しながら実装を進めていく必要がある。

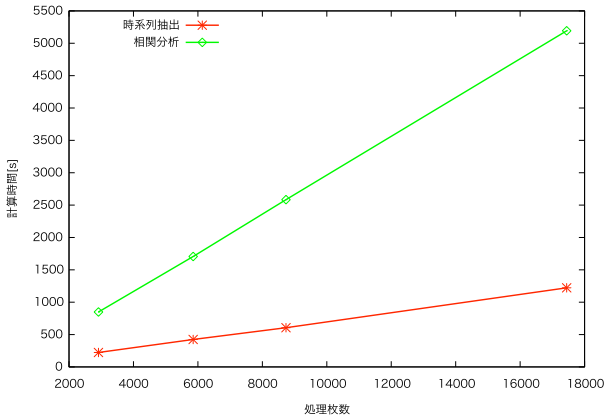

図 17　時系列抽出と相関分析のデータ数増加の影響

なお，今回は時系列を中心とした検討を行ったが，汎用的な時系列画像の分散データマイニングシステムの完成に向けては，この他にも空間処理を中核に据えるようなプロセスや，時空間のコラムを扱うような問題を扱える事が必要である。引き続き検討して，その内容をもとにシステムのプロトタイプを完成していく予定である。

6. おわりに

本研究では，Hadoop，MapReduce を用いた時系列画像に対する汎用的な時空間データマイニングシステムの構築を目指して，時系列データの抽出，統計量の計算，相関分析など，汎用的に必要なプロセスを実装し，そのパフォーマンスのチューニングを検討した。その結果，画像のブロックを分散化の単位として扱い，圧縮を取り入れることによって，計算速度を向上させ，スケーラビリティを改善できることがわかった。また，ブロック数については Reduce 内の処理を飽和させない範囲で，ノード数に一致させるときがもっとも短い計算時間を達成した。また，実際の気象衛星画像を用いて雲量変化の時空間相関分析に適用したところ，従来，困難であった処理が Hadoop による分散処理によって比較的容易に実装する事が可能となった。今後は，より汎用的なシステムの完成をめざして，この他の空間処理を中核に据えるようなプロセスや，時空間のコラムを扱うような問題についてもパフォーマンスの最大化を視野にいれて実装して，システムを完成していく予定である。

文　献

[1] The Apache Software Foundation. “ Apache”, http://hadoop.apache.org, 03.15.2014.

[2] O. Tatebe, K. Hiraga, N. Soda. “Gfarm Grid File System”, New Generation Computing, Ohmsha, Ltd. and Springer, Vol. 28, No. 3, pp.257-275, 2010.

[3] Mohamed H. Almeer. “Cloud Hadoop Map Reduce For Remote Sensing Image Analysis”, Journal of Emerging Trends in Computing and Information Sciences, Vol. 3, No.4, pp.637-644, 2012.

[4] 白崎 裕治, 小宮 悠, 大石 雅寿, 水本 好彦, 石原 康秀, 堤 純平, 檜山 貴博, 中本 啓之, 坂本 道人. “JVO 開発における大規模天文データ処理： 全天対応天文データ分散検索・解析機構の試験構築”，宇宙航空研究開発機構研究開発報告 JAXA-RR-11-007, pp57-66, 2012.

[5] 西前光，本田理恵. ‘Hadoop による時系列画像からの時空間データマイニング- 植生指標の時空間モデリングを例として”, DEIM Forum P2-4, 2013 .

[6] S. Ghemawat, H. Gobioff, and S. Leung “The Google File System”, ACM SIGOPS Operating Systems Review, Vol. 37, No. 5, 2003.

[7] J. Dean and S. Ghemawat. “MapReduce: simplified data processing on large clusters”, Communications of the ACM, Vol. 51, No. 1, pp. 107-113, 2008.

[8] C. J. Tucker,, J. E. Pinzon, M. E. Brown and E. Molly “Global inventory modeling and mapping studies (GIMMS) satellite drift corrected and NOAA-16 incorporated normalized difference vegetation index (NDVI), monthly 1981-2002.” University of Maryland, 2004.

[9] 坂口 祥太，本田 理恵，“気象画像を用いた時空間変動における相関性マイニング “，第 23 回人工知能学会全国大会論文集，2009

[10] 高知大学気象情報頁管理者. “高知大学気象情報頁”, http://weather.is.kochi-u.ac.jp, 03/15/2014

[11] C. Wang and M. Gudrun. “ The ITCZ in the Central and Eastern Pacific on Synoptic Time Scales”, Monthly Weather Review, Vol. 134, pp. 1405–1421, 2006.