Hadoop による大規模画像データの並列分散処理

前野　一樹†　　島田　敬士†　　長原　一†　　谷口　倫一郎†

† 九州大学大学院　システム情報科学府　　〒 819–0385　福岡市西区元岡 744 番地
E-mail: †{maeno,atsushi}@limu.ait.kyushu-u.ac.jp, ††{nagahara,rin}@ait.kyushu-u.ac.jp

あらまし　情報爆発の時代を迎え，Web 上にある大量の様々なデータにアクセス出来るようになった．Web 上の集合知を利用したアプリケーションがいくつも存在するが，この大量のデータを扱うためには効率的な処理方法が必要となってくる．MapReduce は並列処理のプログラミングモデルの 1 つであり，Map 関数と Reduce 関数を記述するだけでタスク管理の知識がなくとも並列処理を実現出来る．本研究では主にテキストデータに対して用いられてきた MapReduce を画像処理に適用している．また大規模画像処理の題材として類似画像検索を行い並列化の効果を検証した．

キーワード　Hadoop, 並列処理,MapReduce, クラウドコンピューティング

1. はじめに

記憶装置の進化や情報ソースの多様化に伴いデータ量は増加の一途を辿り，我々は “情報爆発” の時代を迎えた．データは日々生み出され，2011 年の間に生み出されるデータの総量は，1.8 ゼタバイトにまでのぼると予想されている [1]．また，インターネットの普及により，この大規模データに容易にアクセスすることが可能となった．画像共有サイト Flickr [2] は 50 億枚の画像を，米最大級の SNS である Facebook [3] は 300〜500 億枚の画像を保持し公開している．この大量画像を用いてこのインターネット上の画像データを用いて，画像の欠損を補完する研究 [4] や，“exif” という画像データに付与されたカメラ情報を大量の画像から取り出し，カメラの特性や撮影者の技能を分析する研究 [5] などが行われている．

計算機の進化や様々なアルゴリズムの登場により，大量のデータを効率的に処理できるようになっているが，メモリに乗りきらないような大規模データを扱う際には，並列分散処理環境の構築が必要となっている．しかしながら，従来の並列分散処理では，クラスタ間通信，タスク管理，同期処理などのオーバーヘッド処理のスキルが必要であり，また，処理は問題依存のため，新しいプログラムを作るたびにこの処理を記述する煩雑な作業が必要であった．そこで，そのような技術を持たない者でも，容易に並列分散処理を実現でき，様々な問題に対して適用可能な新たな技術に注目が集まっている．本稿では，先の研究背景で述べた新しい並列分散処理技術として，Hadoop というオープンソースソフトウェアを用いた大規模画像処理を効率的に行う方法について述べ，大規模画像処理の題材として，類似画像検索システムの並列化の効果について述べる．

2. Hadoop

Hadoop の大きな特徴は MapReduce 形式であること，通常よりはるかに大きなブロックサイズでデータを管理していることの 2 つである．MapReduce は分散データ処理のプログラミングモデルであり，その処理は Map フェーズと Reduce フェーズの 2 つのフェーズに分解して行われる．Map フェーズ，Reduce フェーズともに入出力は (Key, Value) 形式に制限される．Hadoop ユーザは，この Map フェーズと Reduce フェーズの処理を記述するだけで，容易に並列処理を行うことが可能である．以下に各フェーズの説明を示す．

・Map フェーズ　入力されたデータを分割して処理し，(Key, Value) 形式に変換する (図 1).
・Reduce フェーズ　各 Key ごとの Value のリストを入力として受け取り，各 Key ごとに集計処理を行う (図 2).

MapReudce 形式は，入力されるデータが全て独立で，その間の関係を考慮する必要がない並列度の高い処理を効率的に扱うフレームワークである．

Hadoop のファイルシステムのブロックサイズはデフォルトでは 64MB だがさらに大きな値に設定されることが一般的であり，サイズの大きな大量のデータを効率的に扱うことに特化している．このため Hadoop は大量データのバッチ処理に適用されるのが一般的である．しかし，異なる Key 値を持つデータを同時に扱えないため，データ間の関係性が必要となるトランザクション処理は Hadoop には適さない．また，Hadoop は高スループットのために最適化されており，レイテンシが犠牲になることがあるため，リアルタイム処理にも Hadoop は適さない．

Hadoop はコモディティハードウェア上で動作するように設計されているため，ハイエンドなハードウェアで構

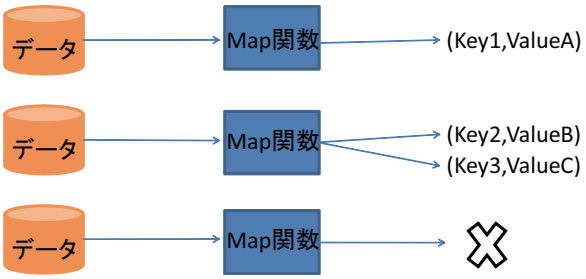

図 1　Map フェーズ

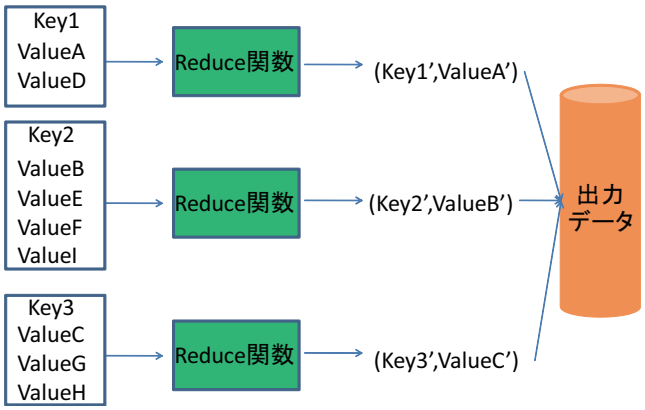

図 2　Reduce フェーズ

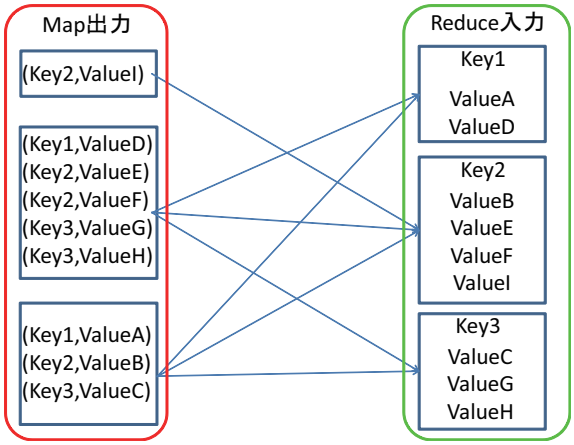

図 3　Shuffle フェーズ

成されたスーパコンピュータに比べコストを抑えることが可能である．従来の MPI(Message-Passing Interface) による並列処理プログラミングは自由度が高いため問題に対して最適なプログラムを記述することができるが，自由度が高い分オーバヘッド処理のための通信パターンなどの記述がユーザにとって大きな負担となっていた．それに対して Hadoop では並列分散処理に必要なタスクはマスタコンピュータが実行する構造となっているため，並列分散処理の知識を持たないユーザでも大規模データを処理できるフレームワークとして期待されている．また，クラウドコンピューティングの進化により，PC クラスタや計算機サーバなどを保有せずとも，ネットワークを介してマシンパワーを提供するサービスを利用できるようになっている．

3.　画像処理の並列化

画像処理の並列化を図る際に，実装したいアプリケーションがどのようなモデルかを考えて MapReduce のフローを構築することは意味のあることである．画像処理を独立型と統合型の大きく 2 つのモデルに分けて考えると，表 1 のように処理を分類することができる．ここではそれぞれのモデルが MapReduce によりどう記述されるかについて述べる．

表 1　画像処理の分類

独立型	統合型
フォーマット変換	K-Means クラスタリング
ビデオエンコーディング	Neural Network
特徴抽出	Expectation Maximization
リサイジング	Support Vector Machines
⋮	⋮

3.1　独立型の処理を行う場合

独立型の処理は画像間の関係を一切考慮する必要がない処理である．独立型とは具体的に表 1 に示すように画像のフォーマット変換や特徴抽出，サイズ変更等，他の画像データからは完全に独立した一枚の画像から一枚の出力を得る種類の処理のことである．ここでは N 枚の画像からそれぞれの色ヒストグラム特徴を抽出する処理を考える．並列処理を考えない通常のスタンドアロン処理の場合，多数の画像に対する処理は単純に N 回のループにより記述されるが，各画像から特徴を抽出する際に他の画像の情報が必要ないため，各ループにおける処理，言い換えると各画像の処理は完全に独立な処理であるとみなすことができる．図 4 に示すように各画像に対する処理を Map 関数で記述するだけで Hadoop により容易に並列化が実現可能である．図 4 の Map 部分では各画像をヒストグラムに変換する処理を行っている．このように，入出力が 1 対 1 で対応する処理においては，出力結果どうしも独立なものとなるため結果を集約する必要がなく，したがって Reduce 処理を記述する必要もない．Hadoop ではユーザが Reduce 処理を明示していない場合，デフォルトの Reduce 関数が実行され単純に結果をまとめる処理が行われ出力が生成される．このようなモデルの具体的な実装例として，New York Times が 4TB の TIFF 画像を全て PNG 画像にした事例や，動画処理に適用しビデオデータを並列でエンコード処理する研究 [6] などが挙げられる．

3.2　統合型の処理を行う場合

統合型の処理は入力画像の部分集合ごとに結果を統合する処理である．具体的には表 1 に示すように，クラスタリングや EM 識別器等の処理がこれにあたる．ここでは例として，K-Means クラスタリングを用いて画像を

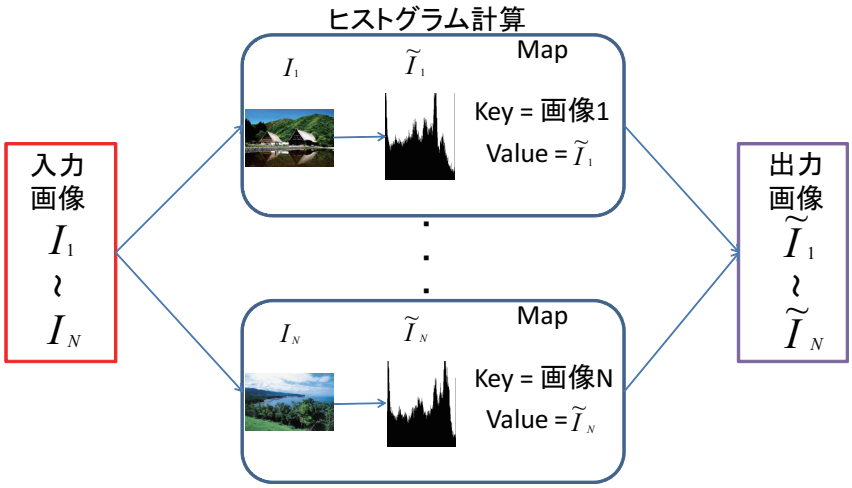

図 4　独立型画像処理の Map 関数実装 (ヒストグラム処理)

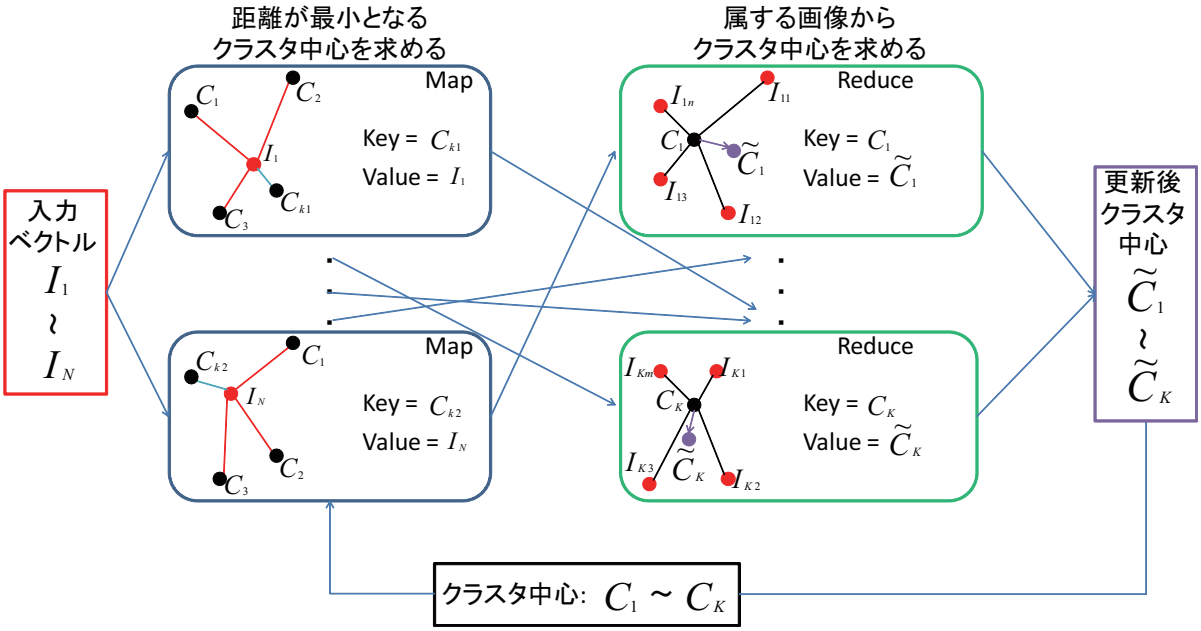

図 5　統合型画像処理による K-Means クラスタリング

分類することを考える．画像の特徴抽出はクラスタリングの前処理として独立型の処理により実行済みであるとすると，K-Means クラスタリングの最初の処理は各入力画像がどのクラスタに属するかを求めることとなる．これは各画像とクラスタ中心との距離が最小となるクラスタを探すことと同じであり，この計算において入力画像間の関係は考慮する必要がないため，入力画像の枚数分のループが枚数分の独立な処理に並列化可能である．Hadoop ではこのステップを Map 関数により記述するが，独立型のように出力が独立ではないためクラスタごとに Key 値を割り当てなくてはならない．図 5 の Map 部分 (青の枠で囲まれた部分) がこのステップを表している．クラスタ中心は入力ファイルとは別のファイルに記述されており，このファイルを Hadoop の分散キャッシュにおくことでこのファイルは各ノードのローカルなメモリにコピーされ，このファイルへのアクセス集中による通信コストを削減している．また図 5 の Map と Reduce の間の矢印で表される Shuffle フェーズで Key 値ごとのリストが生成され，これは各クラスタに属する画像のリストとなっている．K-Means クラスタリングの統合処理は収束判定のための目的関数の計算とクラスタ中心の更新である．目的関数は各画像とクラスタ中心との距離の二乗和により求められ，新しいクラスタ中心はクラスタに属する画像の平均により求められる．Hadoop ではこの統合処理を Reduce 関数に記述する．Map 処理の段階で各画像とクラスタ中心の距離は求められているのでこのステップでは再計算せず足し合わせることで目的関数を求めている．Reduce 処理もクラスタ間の情報のやりとりが必要ないため各クラスタで独立に処理が可能であり並列化することができる．図 5 の Reduce 部分 (緑の枠で囲まれた部分) がこのステップを表している．Reduce により更新されたクラスタ中心によりクラスタ中心を記述した

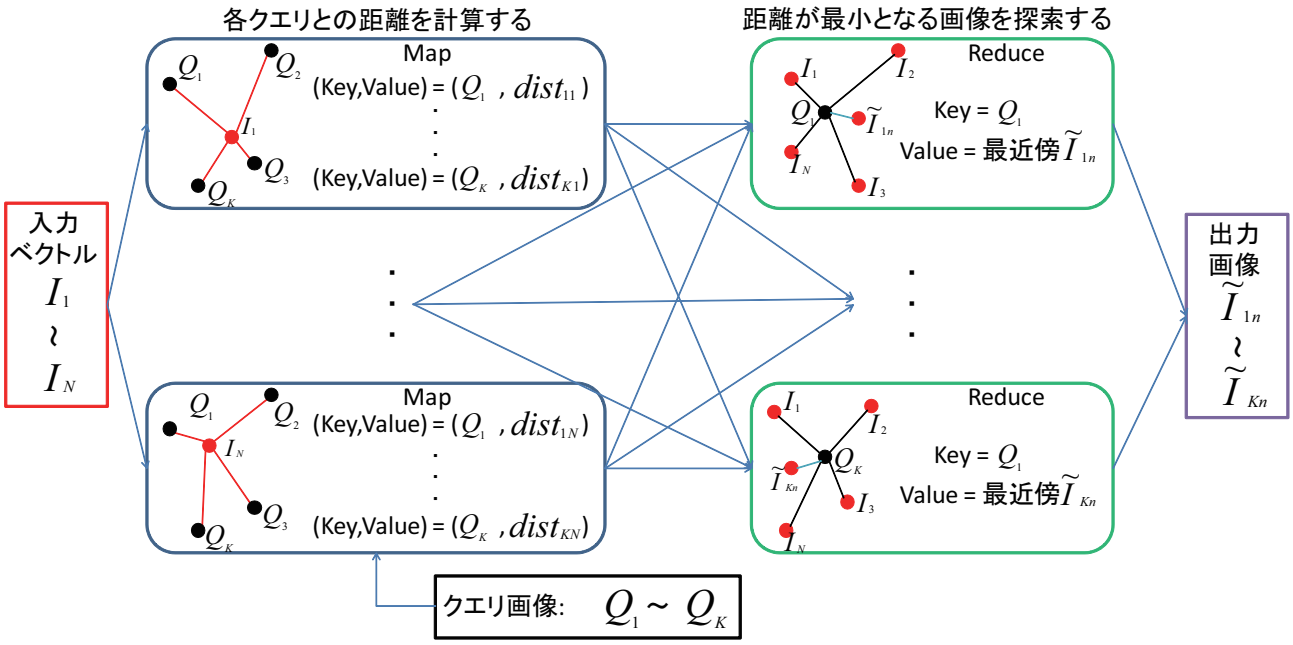

図 6　MapReduce 最近傍探索アルゴリズム

ファイルは上書きされる．この MapReduce を 1 セットとし繰り返すことで K-Means クラスタリングが実現される．表 1 で示したものを含む，K-Means クラスタリング以外のいくつかの機械学習のアルゴリズムもすでに並列化可能であることが示されており [7]，画像に意味づけを行うアノテーションや，画像に付与されたテキストのタグでなく画像特徴から画像検索を行う Content-based image retrieval への応用が可能である．粒子フィルタを MapReduce により並列化する研究 [8] や，MapReduce により並列化された LDA を用いた Web 画像アノテーションの研究 [9] などが実装例として挙げられる．

4.　実　　験

4.1　類似画像検索システム

独立型の処理である特徴抽出と統合型の処理である最近傍探索の複合となる類似画像検索を実装し，実際に並列化の効果が得られることを確認した．今回の実装では特徴抽出と類似度の計算を 1 つの Map 関数でまとめて処理したため統合型のモデルと同一の処理になっている．特徴抽出処理は全ての参照画像データとクエリ画像に対して実行されるため，画像枚数の増加に比例して，計算時間が増大する．最近傍探索処理の計算コストは，線形探索を用いて画像枚数を N，画像特徴の次元数を d とした場合，$O(Nd)$ であり，やはり画像枚数に比例して計算時間が増大する．類似画像検索システムは画像枚数による計算時間の増加が顕著であるため，このシステムの高速化により大規模画像処理における並列分散処理の有用性を示せると考える．

4.2　処理の流れ

本研究で構築した類似画像検索のシステム具体的な手順を示す．

Step 1　参照画像データセットおよびクエリ画像を入力する．
Step 2　参照画像データセットとクエリ画像の特徴量を抽出する．
Step 3　特徴量からデータセットの各参照画像とクエリ画像との類似度を計算する．
Step 4　類似度の最も高いものを出力として返す．

特徴量にはカラーヒストグラムを用いた．RGB の各成分を 8 値で量子化し，512 次元のベクトルとして表した．特徴抽出処理は，(Key, Value)=(画像の名前, 画像の実データ) から (Key,Value)=(画像の名前, 画像の特徴量) への射影であり，各画像ごとに並列に行うことが可能なため，Map 関数に記述し並列化した．

最近傍探索処理は類似度計算と類似度比較の 2 つの処理により構成される．類似度計算は (Key, Value)=(参照画像の名前, 特徴量) から (Key, Value)=(クエリ画像の名前, 参照画像の名前及び類似度) への射影であり，類似度比較は類似度計算の結果の集計処理であるため，MapReduce に適応している．本研究では，類似度は特徴ベクトル間の距離とした．また Combine 関数により各 Map フェーズの処理を集約することで高速化を図った．図 6 に MapReduce を用いた最近傍探索の疑似アルゴリズムを示す．

4.3　実験と結果

MapReduce を用いた類似画像検索の並列化の効果を測定する．実験では，使用するノード数を，1 台，2 台，4 台，8 台と変化させながら，参照画像類 1 万枚，2 万

枚，4 万枚，8 万枚，16 万枚ごとに，類似画像検索の処理時間計測を 3 度行った．これらの画像の収集には画像共有サイト Flickr を用いた．以下に計算機のスペックを表 2 に示す．

各画像枚数ごとの処理時間，台数効果，並列処理効率を表 3〜表 7 に示す．また，台数効果を図 7 に示す．処理時間は 3 度の測定の平均値である．ノード数 p 台使用時の処理時間を $T(p)$ とすると，台数効果 $S(p)$ は式 1，並列処理効率 $E(p)$ は式 2 で表される．台数効果はノード数に対するスケーラビリティを表し，1 台での処理時間に対する速度向上率で表現される．台数効果の値が大きいほど並列処理の効果が高いが，実際にはノード数の値より小さいのが普通であるため，台数効果の値がどの程度ノード数に近いかを表した並列処理効率により並列処理の効果を評価する．

結果から，参照画像の枚数が多いときは，並列処理が効果的であることがわかる．画像枚数が少ないときは，処理時間におけるオーバヘッドの占める割合が高いため，並列化の効果が薄いと考えられる．

$$S(p) = T(1)/T(p) \tag{1}$$

$$E(p) = S(p)/p * 100 \tag{2}$$

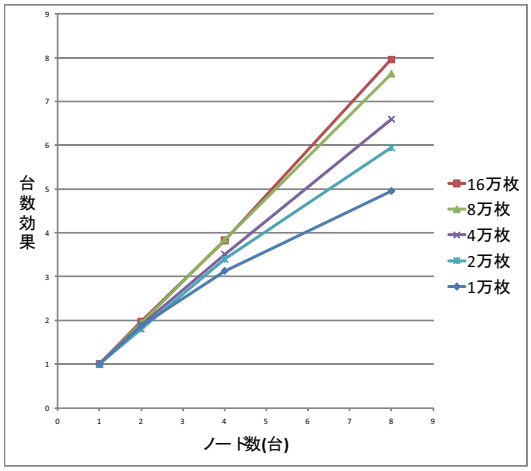

図 7　台 数 効 果

表 2　計算機スペック

メモリ	4GB
プロセッサ	Intel Core　2 Duo CPU E7500 @ 2.93GHz
OS	CentOS5.4

表 3　参照画像 1 万枚の結果

台数 (台)	処理時間 (sec)	台数効果	並列処理効率 (%)
1	281	1	100
2	150	1.88	93.8
4	90	3.13	78.3
8	57	4.95	61.9

表 4　参照画像 2 万枚の結果

台数 (台)	処理時間 (sec)	台数効果	並列処理効率 (%)
1	464	1	100
2	257	1.81	90.2
4	137	3.40	84.9
8	78	5.95	74.3

表 5　参照画像 4 万枚の結果

台数 (台)	処理時間 (sec)	台数効果	並列処理効率 (%)
1	825	1	100
2	436	1.89	94.6
4	235	3.51	87.7
8	125	6.60	82.4

表 6　参照画像 8 万枚の結果

台数 (台)	処理時間 (sec)	台数効果	並列処理効率 (%)
1	1650	1	100
2	850	1.94	97.1
4	429	3.85	96.2
8	216	7.63	95.3

表 7　参照画像 16 万枚の結果

台数 (台)	処理時間 (sec)	台数効果	並列処理効率 (%)
1	3165	1	100
2	1601	1.98	98.8
4	827	3.83	95.6
8	398	7.96	99.4

5. おわりに

本研究では，莫大な時間を要する大規模画像処理に対し，どのように並列分散処理を実装するかを提案した．Hadoop を用いることで，煩雑なオーバヘッド処理記述を行うことなく，高い並列処理効率を得ることが確認できた．今後の課題として，以下のことが挙げられる．

・参照画像の収集

現在使用した参照画像の枚数が最大 16 万枚であり大規模画像としてはスケールが小さいため，さらに多くの画像を収集し 100 万，1000 万といったオーダの画像枚数に対しても並列分散処理が有用であることを示す必要がある．

・ノード数の増加

ノード数が 8 台のときまでは線形に高速化可能であることが確認できた．今後更にノード数を増加させたときに線形に高速化可能か確認する必要がある．

・独立型と統合型の並列効果の違い

統合型の処理は集約処理が含まれるため，独立型と比べてオーバヘッドが大きくなる可能性があり確認する必要がある．

文　献

[1] John F.Gantz, Christopher Chute, Alex Manfrediz, Stephen Minton, Davis Reinsel, Wolfgang Schlichting, and Anna Toncheva,“The Diverse and Exploding Digital Universe,” http://www.emc.com/collateral/analyst-reports/diverse-exploding-digital-universe.pdf,2008.

[2] “flickr,” http://www.lickr.com/.

[3] “facebook,” http://ja-jp.facebook.com/.

[4] James Hays and Alexei Efros,“Scene Completion Using Millions of Photographs,” SIGGRAPH, 2007.

[5] Sujit Kuthirumma, Aseem Agarwal, Dan B. Goldman, and Shree K. Nayar,“Priors for Large Photo Collections and What They Reveal about Cameras,” European Conference on Computer Vision(ECCV), 2008.

[6] Rafael Pereira, Marcello Azambuja, Karin Breitman, and Markus Endler,“An Architecture for Distributed High Perfomance in the Cloud,” Cloud Computing(CLOUD), 2010 IEEE 3rd International Conference on, pp. 482 - 489, 2010.

[7] Cheng-Tao Chu, Sang Kynu Kim, Yi-An Lin, YuanYuan Yu, Gary Bradski, Andrew Y.Ng, and Kunle Olukotun, “Map-Reduce for Machin Learning on Multicore,” Neural Information Processing Systems, 2006.

[8] 石田司，中村和幸，本村陽一，“クラウドコンピューティングを用いた粒子フィルタのための MapReduce アルゴリズム，” 情報論的学習理論テクニカルレポート，2009.

[9] Jiakai Li, Rong Hu, Meihong Wang, Yi Wang, and Edward Y. Chang, “Web-scale Image Annotation,” Pacific-Rim Conference on Multimedia, 2008.