情報論的学習理論テクニカルレポート 2009
Technical Report on Information-Based Induction Sciences 2009 (IBIS2009)

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

MapReduce Particle Filter by Using Cloud Computing Service

石垣司* 中村和幸† 本村陽一‡
Tsukasa Ishigaki Kazuyuki Nakamura Yoichi Motomura

Abstract: Particle filter is a filtering method for state estimation of probabilistic latent variables. For the correct estimation, a large number of particles are needed, however, the use of a large number of particles is forced the high computational cost. The present paper describes parallel computing implementations of particle filters in the framework of MapReduce. The MapReduce is a platform that enables parallel processing without task managements by user. We propose two MapReduce algorithms for the particle filter and vilify the performance of the algorithms with respect to the number of using particles and the number of CPU by using cloud computing service.

Keywords: MapReduce algorithm, particle filter, parallel processing, cloud computing,

1 まえがき

粒子フィルタ [1, 2, 3, 4] は潜在変数の確率分布を推定するための時系列フィルタの一つである。その汎用性と実装の簡便さから、画像処理 [5]、ロボット工学 [6]、海洋科学 [7]、宇宙科学 [8]、バイオインフォマティクス [9, 10, 11]、マーケティング [12] など多様な分野で応用されている。粒子フィルタは各々の潜在変数の確率分布からの独立な実現値とみなすことができる多数の粒子によりその分布形を近似するため、従来では取り扱うことが困難であった非線形・非ガウスモデルに対しても潜在変数の推定が可能となる。(粒子フィルタの呼び名に関しては、Monte Carlo Filter、Bootstrap Filter、ConDensation など様々な呼び名が使用されているが、本論では粒子フィルタという呼び名で統一する。)

粒子フィルタの推定精度には使用する粒子の数が影響することが知られている。粒子の数が多いほど分布の近似精度は向上する。また、粒子の数が少ないと粒子の多様性が失われ適切に分布を近似できない退化と呼ばれる現象が発生するため、推定精度の観点からは粒子数が多いことが望ましい。しかしながら、粒子数に応じて必要な計算コストも $N\log(N)$ のオーダーで増加する。そのため、大規模な問題へ粒子フィルタを適用する際には、少ない粒子数でも粒子の多様性が失われない Merging Particle Filter[13, 14] や Gaussian Particle Filter[15] などの方法の適用と合わせて、クラスタ計算機を使用した並列分散処理によりアルゴリズムの実現が行われている。

しかしながら、並列分散処理を実現するためには、クラスタ間の通信、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となる。そのため、大量粒子を用いる粒子フィルタの使用に際しては、それら並列分散処理に関する知識とクラスタ計算機が必要であった。並列分散処理に関するリテラシーは計算機科学・情報科学以外の研究分野ではほとんど浸透していないのが現状である。そのため従来では、並列分散処理に関する知識を持たない研究者や実務家が大量の粒子を必要とする大規模モデルに対して粒子フィルタを適用するには限界があった。

一方、クラウドコンピューティングの発展により、ユーザがクラスタ計算機やサーバを所持していなくても、ネットワークを介して計算機パワーを使用することができるサービスが開始されている。クラウドコンピューティングを利用した大規模並列処理による大量のデータ処理や、

*産業技術総合研究所, 135-0064 東京都江東区青海 2-41-6 tel. 03-3599-8960, e-mail ishigaki-tsukasa@aist.go.jp,
National Institute of Advanced Industrial Science and Technology, 2-41-6 Aomi, Koto-ku, Tokyo, 135-0064
†明治大学, 214-8571 神奈川県川崎市多摩区東三田 1-1-1
Meiji University, 1-1-1 Higashi-Mita, Tama-ku, Kawasaki-shi, Kanagawa, 214-8571
‡産業技術総合研究所, 135-0064 東京都江東区青海 2-41-6
National Institute of Advanced Industrial Science and Technology, 2-41-6 Aomi, Koto-ku, Tokyo, 135-0064

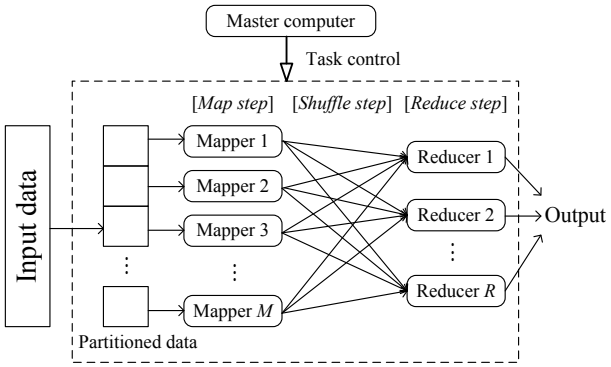

図 1: MapReduce フレームワーク

map : $(\text{key}_m,\text{value}_m) \Rightarrow \text{list}(\text{key}_s,\text{value}_s)$
shuffle : $\text{list}(\text{key}_s,\text{value}_s) \Rightarrow \{\text{key}_r,\text{list}(\text{value}_r)\}$
reduce : $\{\text{key}_r,\text{list}(\text{value}_r)\} \Rightarrow \text{list}(\text{value})$

図 2: MapReduce の各ステップでの key と value のペア

データサーバとしての利用による成功例が報告されている [16]。そこでの大規模データ処理には MapReduce と呼ばれる並列分散処理のためのプログラミングモデルが利用されている。[17] によって提案された MapReduce は Map 処理と呼ばれる各クラスタへ分割・配置されたデータの処理と，Reduce 処理と呼ばれる Map 処理で実行された結果を集約処理する 2 つのステップをユーザが記述するだけで，その他の並列分散化の処理やタスク管理は基盤ソフトウエアが実行するフレームワークである．そのため MapReduce は，並列分散処理に対する知識とスキルを持たない個人でも，大規模な並列分散処理を簡易に実行できるフレームワークとして期待されている．

そこで本論では，粒子フィルタのための MapReduce アルゴリズムを提案し，クラウドコンピューティングを用いた MapReduce フレームワークにおいて粒子フィルタを実装する。ここでは 2 種類のアルゴリズムを実装し、非線形状態空間モデルで記述される状態推定問題を対象とし，使用するクラスタ数と粒子数に対する計算時間（実応答時間）の評価実験を行う．それにより、クラスタ計算機を所持していない並列分散処理に詳しくない個人においても大量の粒子を使用する粒子フィルタが実現可能であることを示す。

2 MapReduce フレームワーク

2.1 MapReduce アルゴリズム

MapReduce アルゴリズムは Map，Shuffle，Reduce とよばれる 3 つのステップにより一連の処理が実行される．この 3 つのステップ内では、あるキー (key) に対して一つの値 (value) を付与した (key,value) を一つの単位として処理が実行される．MapReduce のフレームワークを Fig.1 に，Map，Shuffle，Reduce 処理でのキーと値のペア Fig.2 示す．これらの処理におけるタスクの分割や各ノードへの割り当てなどの管理はマスタと呼ばれるノードにより制御される．またここでは、マスタ以外の計算パワーとして使用される計算機はワーカと呼ばれる。

MapReduce フレームワークでは、入力データを自動的に分割し、その分割データを複数の Mapper と呼ばれる Map 処理を行うワーカへ配置する。Map 処理では入力されたデータに対して処理を施した後、その値に適したキーもしくは値を付与し、$(\text{key}_m,\text{value}_m)$ の組 $\text{list}(\text{key}_s,\text{value}_s)$ を作成する．次に Shuffle 処理において，その組を同一のキーをもつペア群 $\{\text{key}_r,\text{list}(\text{value}_r)\}$ として束ね、キーの順にソートしたペア群を複数の Reducer と呼ばれる Reduce 処理を実行するワーカへ配置する。最後に Reduce 処理において，各 Reducer に配置されたペアに対する処理を実行し list(value) を作成する．

2.2 MapReduce のプラットフォームと応用例

MapReduce の処理は Hadoop と呼ばれるオープンソースプラットフォームを利用して実現することができる [18]。Hadoop は Google 社の基盤技術となっている Google File System と MapReduce のオープンソース実装であり、その処理は Java により記述される。加えて、Hadoop Streaming を利用することで、Ruby、Perl、Python、PHP、R、C++などの複数の言語とその標準入出力により処理を記述することができる。MapReduce のフレームワーク内では、ユーザは Map ステップと Reduce ステップに実行すべき処理を記述するだけで、並列分散処理に必要なタスクはマスターコンピュータが実行する構造になっている。そのため、並列分散処理に詳しく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が高まっている。

MapReduce アルゴリズムの代表例である WordCount（与えられた文章内の出現単語数をカウントする処理）[17] では，Map 処理において各単語をキーとして各キーの値を 1 とし，その組を Shuffle 処理した後，Reduce 処理において同一キー毎に値を加算することで出現単語数の計算を行っている．機械学習の分野においては Statistical Query Model として表現される局所重み付け線形回帰，k-means，ロジスティック回帰，ナイーブベイズ，

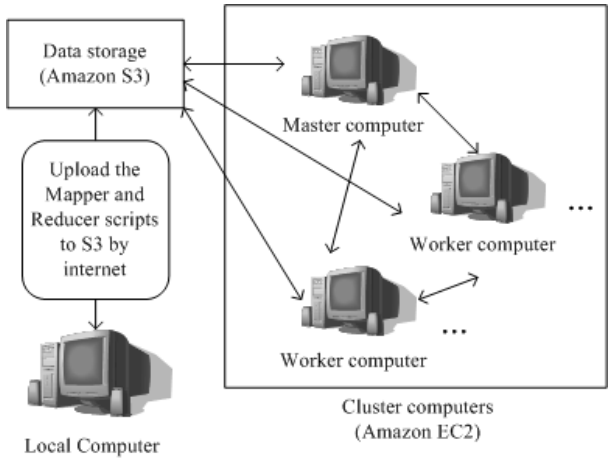

図 3: Amazon Elastic MapReduce サービスの実行環境

サポートベクターマシン，独立成分分析，主成分分析，判別分析，EM アルゴリズム，バックプロパゲーションアルゴリズム等の学習アルゴリズムは MapReduce により実現できることが示されている [19]．また，遺伝的アルゴリズムへの応用も報告されている [20]．

2.3 本論で使用する計算機環境

MapReduce フレームワークはクラスタ計算機上で Hadoop により作用する。そのため従来ではクラスタ計算機の使用が必要であった。しかし、現在ではネットワークを介してクラスタ計算機の計算パワーを提供するクラウドコンピューティングが実用化されている。そのため本論では、Amazon.com, Inc. によって提供されているクラウドコンピューティングサービスである Amazon web services を使用する．その理由は、現存のクラスタコンピューティングの中で最も簡便に MapReduce フレームワークを取り扱うことができる Amazon Elastic MapReduce サービスが提供されているためである。

Amazon web services では Amazon EC2 と称される仮想マシンの計算パワーと Amazon S3 と呼ばれるデータストレージをネットワークを介して提供している．ここでは，MapReduce 処理を EC2 と S3 を用いて実行するためのサービスである Amazon Elastic MapReduce サービスを使用する．本サービスは Mapper と Reducer のスクリプトを記述し、S3 上にアップロードして使用するだけで MapReduce フレームワークでの並列分散処理を実行可能としている。その模式図を図 3 に示す。ここでは、EC2 クラスタ計算機上で実行された MapReduce 処理の出力は S3 データストレージに保存される。また、利用するクラスタ数とその性能に応じた料金を 1 時間単位で課金されるシステムである．以下の実験では，High-CPU Extra Large Instance と称される 64bit プラットファーム，7GB メモリ、20 コア ECU ユニットをもつインスタンスと呼ばれる仮想マシンを指定し，サービスを利用した．また、この使用料は 1 インスタンスにつき 1 時間 0.8 ドルである。

3 粒子フィルタによる状態推定

3.1 状態空間モデルによる状態推定

粒子フィルタは与えられた観測データから，その観測データに関連する潜在変数の確率分布を推定するアルゴリズムである．本論では，その潜在変数の推定問題を柔軟かつ広範に取り扱うことができる状態空間モデルの枠組みで論を展開する．

ここで，時刻 t の観測ベクトルを $\boldsymbol{y}_t$ とし，その観測に影響を与える変数を要素としてもつベクトルを状態ベクトルと呼び $\boldsymbol{x}_t$ とする．通常，状態ベクトル $\boldsymbol{x}_t$ は潜在変数として取り扱われる．ここで f と h を非線形関数，v_t を密度関数 $q(v)$ に従うシステムノイズ，w_t を密度関数 $r(v)$ に従う観測ノイズとすると，非線形・非ガウス型の状態空間モデルは以下のような時系列モデルとして考えることができる．

$$\boldsymbol{x}_t = f_t(\boldsymbol{x}_{t-1}, \boldsymbol{v}_t), \tag{1}$$

$$\boldsymbol{y}_t = h_t(\boldsymbol{x}_t, \boldsymbol{w}_t) \tag{2}$$

時刻 $\{t_0, \cdots, t_1\}$ までの状態が与えられたときの時刻 t_2 の状態を $\boldsymbol{x}_{t_2|t_1}$，$\{\boldsymbol{y}_1, \cdots, \boldsymbol{y}_j\}$ を $\boldsymbol{y}_{1:j}$ と表すと，状態ベクトルの推定は $\boldsymbol{y}_{1:t}$ が与えられたときの $\boldsymbol{x}_t$ の条件付き分布 $p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t})$ を求めればよい．状態空間モデル上でそれらは以下の一期先予測とフィルタリングのステップにより実現できる．

[一期先予測]

$$p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1}) = \int p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) p(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{1:t-1}) d\boldsymbol{x}_{t-1}, \tag{3}$$

[フィルタリング]

$$p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}) = \alpha^{-1} p(\boldsymbol{y}_t|\boldsymbol{x}_t) p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1}), \tag{4}$$

$$\alpha = \int p(\boldsymbol{y}_t|\boldsymbol{x}_t) p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1}) d\boldsymbol{x}_t$$

ここで解析の対象としているモデルが線形モデルかつガウス分布で表現される場合，状態ベクトルと観測ベクトルは線形状態空間モデルの枠組みで取り扱うことが可能である．そこでは，Kalman Filter のアルゴリズムを用いることで，最適な状態ベクトルの推定を行うことが可能であることが知られている [21]．一方，対象とするモデルが非線形もしくは非ガウス分布で表現される

initialization step
Generate initial particles $\boldsymbol{x}_{0|0}^{(i)} \sim p_0(\boldsymbol{x})$, where p_0 is an initial distribution
for $t = 1$ to T **do**
  ## Prediction step ##
  for $i = 1$ to N **do**
    $\boldsymbol{x}_{t|t-1}^{(i)} = f(\boldsymbol{x}_{t-1|t-1}^{(i)}, v_t^{(i)})$
    $w_t^{(i)} =$ likelihood of particle i
  end for
  $W_t = \sum_{i=1}^{N} w_t^{(i)}$
  ## Filtering step ##
  for $i = 1$ to N **do**
    $\boldsymbol{x}_{t|t}^{(i)} =$ filterd particles from $\{\boldsymbol{x}_{t|t-1}^{(1)} \cdots \boldsymbol{x}_{t|t-1}^{(N)}\}$ by sampling with replacement in proportion to $w_t^{(i)}/W_t$
  end for
end for

図 4: 粒子フィルタのアルゴリズム

場合，Kalman Filter による状態推定は最適性を保証されない．そのため，非線形モデルに対しては Extended Kalman Filter などの状態推定法が用いられていたが，推定精度やアルゴリズムの収束性に問題があった．また，Non-Gaussian Filter[22] などの状態推定法も存在するが，計算量の観点から高次元（5 次元以上）の状態ベクトルに関する状態推定問題への適用が困難であった．

3.2 粒子フィルタ

そこで，状態ベクトルの確率分布形を数値解析的に推定するのではなく，状態ベクトルがとる確率分布からの確率変数の実現値としてみなすことができる多数の粒子（確率分布からのサンプル）により分布を近似する状態推定法が粒子フィルタである．その各分布は次式の

$$p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}) \simeq \frac{1}{N}\sum_{i=1}^{N}\delta(\boldsymbol{x}_t - \boldsymbol{x}_{t|t}^{(i)}), \tag{5}$$

$$p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1}) \simeq \frac{1}{N}\sum_{i=1}^{N}\delta(\boldsymbol{x}_t - \boldsymbol{x}_{t|t-1}^{(i)}), \tag{6}$$

を式 (3),(4) へ代入することで近似する．粒子フィルタのアルゴリズムを図 4 に示す．図中の N は粒子の数を，T は全体のステップ数を表す．

問題の規模や特性によりその適切な数は異なるが、粒子数が少なすぎると図に示す退化が生じ，適切な状態推定が行われない．また、粒子数はパラメータや状態変数の推定精度にも影響し、遺伝子ネットワークのパラメータ推定問題において，粒子数を十分な数にまで増やすことでパラメータ推定精度が向上する例も報告されている [11].

Do initialization step
In Map step
Do partition particles
for $t = 1$ to T **do**
  Do prediction step
  Do filtering step
end for
list(key$_s$,value$_s$) $\Leftarrow$ (random number, $\boldsymbol{x}_{T|T}^{(i)}$)
Do Shuffle step
In Reduce step
list(value) $\Leftarrow$ {key$_r$,list(value$_r$)}

図 5: MapReduce 反復無し MRPF アルゴリズム

Do initialization step
for $t = 1$ to T **do**
  ## In Map step ##
  Do partition particles
  Do prediction step
  Do filtering step
  list(key$_s$,value$_s$) $\Leftarrow$ (random number, $\boldsymbol{x}_{t|t}^{(i)}$)
  Do Shuffle step
  ## In Reduce step ##
  list(value) $\Leftarrow$ {key$_r$,list(value$_r$)}
  particles at next time step $\Leftarrow$ list(value)
end for

図 6: MapReduce 反復あり MRPF のアルゴリズム

4 MapReduce 粒子フィルタ

本章では MapReduce フレームワークにおける粒子フィルタのアルゴリズムを実装する．ここではそのアルゴリズムを MapReduce 粒子フィルタと呼び MRPF(MapReduce particle filter) と呼ぶ．以下に，1 度の MapReduce 処理で実現する MRPF を MapReduce 反復無しの MRPF，MapReduce 処理を複数回繰り返す MapReduce 反復有りの MRPF の 2 種類の MRPF を提案する．MapReduce 反復無しの MRPF アルゴリズムを図 5 に、MapReduce 反復ありの MRPF アルゴリズムを図 6 にそれぞれ示す．また，両者の概念図を図 7 と図 8 にそれぞれ示す．

4.1 MapReduce 反復無しの粒子フィルタ

ここでは Map 処理の各ノードにおいて，それぞれ独立に $t = 1$ から T ステップの粒子フィルタを実行する．

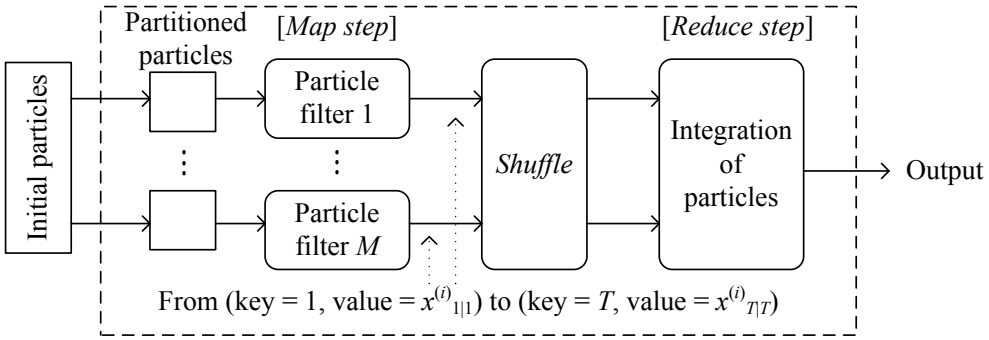

図 7: MapReduce 反復無し MRPF の概念図

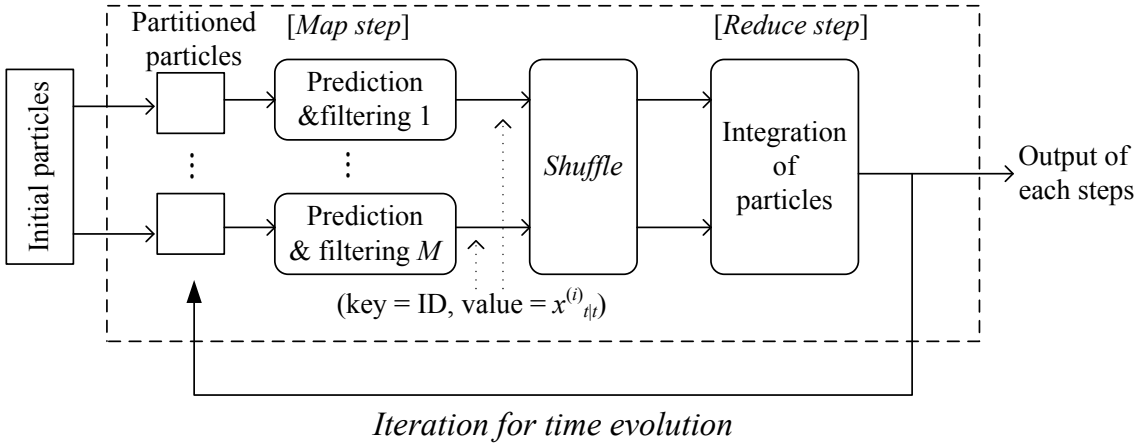

図 8: MapReduce 反復あり MRPF の概念図

各 Mapper からキーと値のペア $(t, \boldsymbol{x}^{(i)}_{t|t})$ のリストを出力する。ここでキーの値には、どの時刻の粒子であるのか判別するために時刻 t を付与する。その後，Reduce 処理において全体の各ノードの粒子を統合し、時刻 $1 \sim T$ までの $\boldsymbol{x}^{(i)}$ が出力される．ここでは 1 度の Map 処理と Reduce 処理によって粒子フィルタの並列分散処理が実行される．このアルゴリズムでは退化を起こさないため各ノードで十分な粒子数が必要となるが，実装が簡便であるというメリットがある．

4.2 MapReduce 反復ありの粒子フィルタ

ここでは各 Mapper においてそれぞれ独立に時刻 t の一期先予測とフィルタリングのステップを実行し，キーと値のペア (random number, $\boldsymbol{x}^{(i)}_{t|t}$) のリストを出力とする。ここでキーの値には乱数により各粒子の ID を付与している。その後 Reduce 処理で各ノードでフィルタリングされた時刻 t の粒子を統合し、各時刻の粒子 $\boldsymbol{x}^{(i)}_{t|t}$ を出力する．その統合された粒子を時刻 $t+1$ の入力とし，時刻 $t+1$ の MapReduce 処理を実行する．このアルゴリズムでは全粒子の統合と再配置の処理と MapReduce のオーバーヘッド作業を各時刻ステップ毎に行う必要があるため、反復無しのアルゴリズムと比べて計算時間は必要となるが、全体の粒子の多様性を維持することができる．

5 実験と結果

5.1 実験

本章では上記の 2 つのアルゴリズムを非線形モデルの状態推定問題に適用し，その計算時間を検証する．ここでは，非線形状態推定問題でベンチマークとしてよく用いられる以下の非線形システムを対象として実験を行う [1, 2, 23].

$$\begin{cases} x_t = \frac{1}{2}x_{t-1} + \frac{25x_{t-1}}{1+x_{t-1}^2} + 8\cos(1.2t) + v_t \\ y_t = \frac{x_t^2}{20} + w_t \\ x_0 \sim N(0,5), v_t \sim N(0,1), w_t \sim N(0,10) \end{cases}$$

x_t と y_t はそれぞれ時刻 t の潜在変数と観測変数である。また $N(0,\sigma^2)$ は平均ゼロ、分散 σ^2 の正規分布とする。

表 1: MapReduce 反復無し MRPF の計算時間 [秒]

	C=1	C=2	C=4	C=8	C=16
10K	3068	1571	1531	1523	1548
50K	9210	4591	3099	2578	1625
1M	19798	11715	6422	3312	1867

ここでは，利用するクラスタ数は 1,2,4,8,16 インスタンス、使用する粒子の数はそれぞれ 10 万、50 万、100 万粒子と設定する。また、MapReduce 反復無しのアルゴリズムではステップ数を $T = 100$、MapReduce 反復ありのアルゴリズムでは $T = 1$ として実験を行う。

MapReduce 反復ありのアルゴリズムでは、各時刻ステップ毎の出力を 1 期先ステップの入力として再配置する処理が必要となる。現サービスではストレージ内でそれら粒子の再配置処理を実行することができない。そのため、ストレージからローカル計算機にデータを転送し、再配置のため処理を行い、そのデータを再びストレージへ転送する処理が必要となる。また、このアルゴリズムでは各時刻ステップ毎に同様の処理を繰り返すため、$T = 1$ として実験を行い、その計算時間に必要なステップ数を乗算することで任意の T の値に対して実際の計算時間を推定できる。

粒子を分散して並列処理を行うため、適切な粒子数を各 Mapper に割り当てる必要がある。本実験では分散された一つの粒子のセットを 1 万粒子と設定する。すなわち、各実験において使用した粒子数は 1 万粒子× 10 セット（10 万粒子）、1 万粒子× 50 セット（50 万粒子）、1 万粒子× 100 セット（100 万粒子）である。（1 セット 1 粒子とすると原子モンテカルロ法に、1 セットに全粒子を割り当てると通常の粒子フィルタとなる）。また、状態変数の推定精度については [1, 2, 23] 等で詳しく議論されているため本論では取り扱わない.

5.2 MapReduce 反復無し MRPF の結果

前節で述べた非線形状態推定問題に対し，MapReduce 反復無し MRPF を適用した．$T = 100$ における MapReduce 反復無し MRPF の計算時間を各クラスタ数、各粒子数ごとに表 1 に示す。表内の数字は計算にかかった秒数であり、10K、50K、1M はそれぞれ 10 万粒子、50 万粒子、100 万粒子を、C=n は n クラスタを使用した実験であることを意味する。また、各クラスタ数に関して、1 クラスタ使用時に対する計算速度の倍率を図 9 に示す。

100 万粒子を使用した実験では、クラスタ数に比例して計算速度が上がっていることが図 9 から分かる。1 クラスタ使用時の計算時間は約 5 時間 30 分かかっているが、16 クラスタ使用時の計算時間は約 31 分まで短縮さ

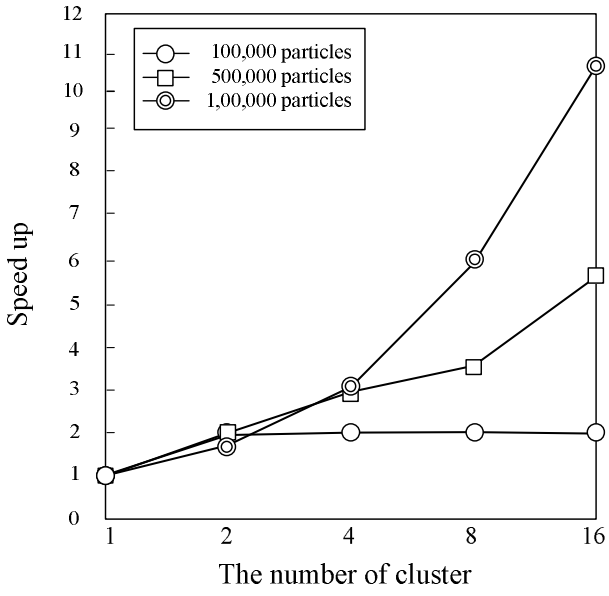

図 9: MapReduce 反復無し MRPF の計算時間の短縮率

表 2: MapReduce 反復あり MRPF の計算時間 [秒]

	C=1	C=2	C=4	C=8	C=16
10K	85	78	56	51	50
50K	245	196	119	71	51
1M	473	380	211	109	62

れた。50 万粒子でも並列化による計算時間の短縮が確認できるが 100 万粒子時と比べ効果は小さい。また、10 万粒子使用時では顕著な並列化の効果は見られなかった。

5.3 MapReduce 反復あり MRPF の結果

前節で述べた非線形状態推定問題に対し，$T = 1$ として MapReduce 反復あり MRPF を適用した．計算処理にかかった時間を表 2 に、また各粒子数毎のデータ転送にかかった時間を表 3 にそれぞれ示す。

計算処理時間の短縮に関しての並列化の効果は MapReduce 反復無し MRPF の結果とほぼ同様の結果となっている。しかし、50 万粒子、100 万粒子ではすべてのクラスタ数に対してデータ転送時間が計算処理時間を上回っている。16 クラスタ、100 万粒子使用時のデータ転送時間は計算処理時間の約 12 倍もかかっている。このため、計算時間の短縮のためにはデータ転送時間がボトルネックとなることが分かる。また、実験結果を 100 倍して $T = 100$ に対する 100 万粒子使用時の計算時間の推定を行うと、1 クラスタでは約 32 時間、16 クラスタでは約 23 時間となった。

表 3: データ転送時間 [秒]

0.1M	0.5M	1M
75	351	761

6 考察

MapReduce 反復あり MRPF に関しては、100 万粒子使用時に関して並列分散化の効果が大きく表れている。16 クラスタ使用時には 1 クラスタ使用時に比べ約 10.6 倍の高速化が実現されている。50 万粒子、10 万粒子の使用時に 100 万粒子使用時と比べ並列分散化の効果が低いのは、並列分散化による計算速度は分散させる粒子のセット数に依存しているためと考えられる。1 万粒子×10 セット（10 万粒子）に対して 16 クラスタを利用するのは、目的に対して不適切である。そのため、分散させる粒子のセット数と使用するクラスタ数による費用対効果などの関係性を調べることは今後の課題である。

MapReduce 反復あり MRPF におけるデータ転送時間の問題は、ローカルなクラスタ計算機を用いることやローカリティを有したネットワークデータ転送を必要としないクラウドコンピューティングを用いることで解決できる。しかし、本論の趣旨から外れるため、ここでは議論しない。また計算処理時間に関しては、表 2 よりアルゴリズムによる並列分散化の有効性を確認できるている。そのため、ローカリティを伴う MapReduce フレームワーク実行環境を有するサービスが提供されることで、本アルゴリズムは実用可能となるであろう。

本論で述べた MRPF のアルゴリズムは比較的単純な粒子フィルタの MapReduce での実装である．MapReduce の特性を活かした効率の良いアルゴリズムの考案は今後の課題である．

7 むすび

本論では MapReduce フレームワークにおける粒子フィルタアルゴリズムについて述べた．2 種類の MapReduce 粒子フィルタアルゴリズムを提案・実装し，クラウドコンピューティングのサービスを使用して計算時間の短縮について検証した．その結果クラウドコンピューティングと MapReduce フレームワークを用いた並列分散処理化の効果を確認することができた．これにより，クラスタコンピュータを持たない研究者・実務家においても大量の粒子を使用する粒子フィルタの実装と応用が可能であることを示すことができた．近年，粒子フィルタの応用はマーケティングの分野へも広がりつつあり，大規模統計モデルを実務へ応用したい実務家にとって有益な指針を示すことができた．より大規模なモデルでの検証や実問題への応用は今後の課題である．

謝辞

本研究では，経済産業省サービス工学研究開発事業 IT とサービスの融合による新市場創出促進事業の支援を受けている．

参考文献

[1] N. J. Gordon, D. J. Salmond and A. F. M. Smith, “Novel approach to nonlinear/non-Gaussian Bayesian state estimation,” *The Proceedings of IEE F*, Vol. 140, No. 2, pp. 107-113, 1993

[2] G. Kitagawa, “Monte Carlo filter and smoother for non-Gaussian nonlinear state space model,” *Journal of Computational and Graphical Statistics*, Vol. 5, No. 1, pp. 1-25 1996

[3] A. Doucet, J. de Freitas and N. Gordon, *Sequential Monte Carlo Methods in Practice*, Springer-Verlag, 2001

[4] 樋口知之, “粒子フィルタ”, 電子情報通信学会誌, Vol.88, No.12, pp.989-994, 2005

[5] M. Isard and A. Blake, “CONDENSATION-Conditional Density Propagation for Visual Tracking”, International Journal of Computer Vision, Vol29, No.1, pp.1573-1405, 1998

[6] S. Thrun, W. Burgard and D.Fox, *Probabilistic Robotics*, MIT Press, 2005

[7] K. Nakamura, N. Hirose, B.H. Choi and T. Higuchi, “Particle Filtering in Data Assimilation and its Application to Boundary Condition of Tsunami Simulation Model”, Data Assimilation for Atmospheric, Oceanic and Hydrologic Applications, pp.353-366, S.K. Park and L. Xu (ed.), Springer, 2009

[8] S. Nakano and T. Higuchi, “Estimation of a long-term variation of a magnetic- storm index using the merging particle filter”, IEICE Trans. on Information and Systems, Vol.E92-D, No.7, pp.1382-1387, 2009

[9] M. Nagasaki, R. Yamaguchi, R. Yoshida, S. Imoto, A. Doi, Y. Tamada, H. Matsuno, S.

Miyano and T. Higuchi, “Genomic Data Assimilation for Estimating Hybrid Functional Petri Net from Time-Course Gene Expression Data”, Genome Informatics, Vol.17, No.1, pp.46-61, 2006

[10] R. Yoshida, M. Nagasaki, R. Yamaguchi, S. Imoto, S. Miyano and T. Higuchi, “Bayesian learning of biological pathways on genomic data assimilation”, Bioinformatics, Vol.24, No.22, pp.2592-2601, 2008

[11] K. Nakamura, R. Yoshida, M. Nagasaki, S. Miyano and T. Higuchi, “Parameter Estimation of In Silico Biological Pathways with Particle Filtering Towards a Petascale Computing””, The Proceedings of 14th Pacific Symposium on Biocomputing, pp.227-238, 2009.

[12] 佐藤、樋口, “動的個人モデルによる消費者来店行動の解析”, 日本統計学会誌, Vol.38, No.1, pp.1-19, 2008

[13] 中野、上野、中村、樋口、“ Merging Particle Filter とその特性 ”、統計数理、Vol.56 No.2、pp. 225-234, 2008.

[14] S. Nakano, G. Ueno and T. Higuchi, “Merging particle filter for sequential data assimilation”, Nonlinear Processes in Geophysics, Vol.14, No.4, pp. 395-408, 2007

[15] J.H. Kotecha and P.M. Djuric, “Gaussian particle filtering”, IEEE Transactions on Signal Processing, Vol.51, No.10, pp.2592-2601, 2003

[16] 日経 BP 社出版局, クラウド大全, 日経 BP 社, 2009

[17] J. Dean and S. Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters”, OSDI’04: Sixth Symposium on Operating System Design and Implementation, 2004

[18] D. Borthaku, “The Hadoop Distributed File System: Architecture and Design”, Retrieved from lucene.apache.org/hadoop, 2007.

[19] C-T. Chu, S.K. Kim, Y-A. Lin, Y.Y. Yu, G. Bradski, A.Y. Ng and K. Olukotun, “Map-Reduce for Machine Learning on Multicore”, Advances in Neural Information Processing Systems 19, 2006

[20] C. Ji, C. Vecchiola and R. Buyya, “MRPGA: An Extension of MapReduce for Parallelizing Genetic Algorithms”, 4th IEEE International Conference on eScience, pp.214-221, 2008

[21] R. E. Kalman, “A new approach to linear filtering and prediction problems,” *Transaction of ASME - Journal of Basic Engineering*, Vol.82 pp. 35–45, 1960

[22] G. Kitagawa, “Non-Gaussian State-Space Modeling of Nonstationary Time Series”, Journal of the American Statistical Association, Vol.82, No.400 pp.1032-41, 1987

[23] 中村、上野、樋口, “データ同化：その概念と計算アルゴリズム”, 統計数理, Vol.53, No.2, pp.211-229, 2005