Vol.2018-HPC-163 No.13
2018/3/1

pbdMPIを用いたエントロピー推定プログラムの並列化と性能評価

河合 祐輔[1,a)]　日野 英逸[2]　建部 修見[3]

概要：機械学習とは，入力されたサンプルデータを解析してそこから有用な規則を見つけだす方法論であり，その解析結果を新たなデータに当てはめることで，規則にしたがって将来を予測することができる．機械学習によく用いられるプログラミング言語のひとつにRがある．Rは統計コンピューティングとグラフィックスのためのオープンソースプログラミング言語とソフトウェア環境である．Rを用いた並列コンピューティングへの関心は増加してきており，対象となるデータセットが大きくなっていることが理由の一つである．本研究では，機械学習において重要な役割を果たすエントロピー推定を，pbdMPIパッケージを用いて並列化する．性能評価においてはノード（プロセス）数を増やすことにより高速化することを確認した．

1. はじめに

機械学習 [1] とは，入力されたサンプルデータを解析してそこから有用な規則を見つけ出す方法論であり，その解析結果を新たなデータに当てはめることで，規則にしたがって将来を予測することができる．機械学習は，検索エンジン，医療診断，スパムメールの検出，金融市場の予測，DNA配列の分類，音声認識や文字認識などのパターン認識など幅広い分野で用いられている．そして，その機械学習をはじめとして，統計，信号処理，パターン認識において重要な役割を果たすものが微分エントロピーである，X を確率密度 $f(x)$ を持つ p 次元の確率変数とすると，その微分エントロピーは

$$H(f) = -\int f(x) \ln f(x) dx \tag{1}$$

と定義される．

機械学習によく用いられるプログラミング言語のひとつにR[2]がある．Rは統計コンピューティングとグラフィックスのためのオープンソースプログラミング言語とソフトウェア環境であり，パッケージの使用により高度に拡張が可能である．CRANには多くの作成されたパッケージが公開されており，それらは自由に使用できる．

Rを用いた並列コンピューティング技術の使用に焦点を当てている研究は増えている [3]．その要因の一つは，対象となるデータセットがより大きくなっていることである．得られるデータ量は増加の一途を辿っており，今後も増加していくことは間違いない．機械学習は一般的に，データ数の増加に伴い精度が向上するが，同時に実行時間も長くなってしまう．

MPI（Message-Passing Interface）は，さまざまな並列コンピューティングで機能するようにデザインされた，標準化されたメッセージパッシングシステムである．メッセージパッシングとは，メッセージを受信者へ送信することによって行われる通信の一形態であり，分散メモリシステムで広く使用されている．MPIは言語に依存しない方法でプロセス間に必要な仮想トポロジー，同期および通信機能を提供することを目的としている．

SPMD並列プログラミングモデルでは，通常同一のプログラムがすべてのプロセッサ上で大きなデータセットの異なる部分を対象に実行される．pbdMPI[4] は，Rmpi[3] のようなインタラクティブモードはサポートされておらず，バッチモードプログラミングを意図しており，mpirunによってプロセスが生成される．Rのクラスとメソッドの実装方式にはS3方式とS4方式があり，S4方式を採用している．S4メソッドはほとんどの集合関数に使用されているので，一般的なRオブジェクトに対して拡張するのが簡単である．配列型または行列型のメソッドは，シリアライズおよびデシリアライズを用いずに実装されているため，

[1] 筑波大学情報学群情報科学類
University of Tsukuba, School of Infomatics, Colleage of Information Science
[2] 筑波大学システム情報系
University of Tsukuba, Faculty of Engineering, Information and Systems
[3] 筑波大学計算科学研究センター
University of Tsukuba, Center for Computational Sciences
[a)] kawai@hpcs.cs.tsukuba.ac.jp

Rmpi よりも高速な通信が可能．シリアライズとは，オブジェクトの状態をバイト配列状のデータ構造に変換することである．

そこで本研究では，pbdMPI パッケージを用いて微分エントロピー推定プログラムの並列化を行う．pbdMPI は，R で SPMD の並列処理をサポートするためのパッケージである．pbdMPI パッケージを用いて微分エントロピー推定プログラムを並列化することで，ノード（プロセス）数の増加に伴い処理時間が向上していることを示す．

本論文の構成は以下の通りである．第 2 章では関連研究を紹介する．第 3 章では本研究で並列化するエントロピー推定法の解説をする．第 4 章では並列エントロピー推定プログラムの具体的な実装について述べる．第 5 章では並列エントロピー推定プログラムの評価を行う．第 6 章では本研究についてまとめる．

2. 関連研究

本章では R を用いた並列コンピューティングのためのパッケージについていくつか紹介する．

Rmpi パッケージは MPI へのインターフェースすなわちラッパーであり，ユーザーが MPI 実装の詳細を知らなくても問題ないように，R から低レベルの MPI 関数へのインターフェースを提供する．MPI をインストールする必要があり，普及している MPI 実装で動作する．Rmpi パッケージはマスターからスレーブ（ワーカー）で R インスタンスを起動するスクリプトが含まれている．そのインスタンスはマスターから閉じられるまで実行される．このパッケージは MPI，API のラッピングに加えていくつかの R 固有の関数を提供する．

Hadoop[5] は，データを複数のサーバに分散し，並列して処理するミドルウェアである．Hadoop では，1 台のマスター（Name Node）と，その配下の多数のスレーブ（Data Node）が連携してデータの高速処理を行う．特徴として，HDFS（Hadoop Distributed File System）と MapReduce の二つがある．HDFS は，Hadoop 独自の分散ファイルシステムであり，大きなファイルを複数のブロック単位に分割し，それを複数のノードにまたがり格納する．MapReduce は，計算処理を，与えられたデータから欲しいデータを抽出，分解する Map 処理と抽出されたデータを集計する Reduce 処理の二つの手順でこなす手法である．RHadoop[6] とは，R から Hadoop を操作するためのパッケージコレクションである．

3. エントロピー推定

この章では，今回用いた 4 つのエントロピー推定法について説明し，そのアルゴリズムを示す．

3.1 準備

観測データセット $D = \{x_i\}_{i=1}^n$ を用いてエントロピー $H(f)$ を推定する方法を考える [7]．$x_i, i = 1, ..., n$ は確率密度関数 $f(x)$ を持つ p 次元の確率変数 X の実現値である．密度関数の推定量 $\hat{f}(x)$ が得られれば，エントロピーは数値積分によって推定できるが，不安定さなどの理由から

$$\hat{H}(D) = -\frac{1}{n}\sum_{i=1}^{n} \ln \hat{f}(x_i) \tag{2}$$

を用いる．

$D = \{x_i\}_{i=1}^n$ を用いて検査点 $z \in \mathcal{R}$ における確率密度関数の値 $f(z)$ を推定する問題を考える．$b(z;\epsilon) = \{x \in \mathcal{R}^p | \parallel z - x \parallel < \epsilon\}$ は，体積 $|b(z;\epsilon)| = c_p\epsilon^p$ の z を中心とする ϵ 球とする．$c_p = \pi^{p/2}/\Gamma(p/2+1)$，$\Gamma(\cdot)$ はガンマ関数である．z を中心とする ϵ 球の確率質量は

$$q_z(\epsilon) = \int_{x \in b(z;\epsilon)} f(x)dx \tag{3}$$

で定義される．被積分関数を展開すると

$$\begin{aligned} q_z(\epsilon) &= \int_{x \in b(z;\epsilon)} \{f(z) + (x-z)^{\mathrm{T}}\nabla f(z) + O(\epsilon^2)\}dx \\ &= |b(z;\epsilon)|(f(z) + O(\epsilon^2)) = c_p\epsilon^p f(z) + O(\epsilon^{p+2}) \end{aligned}$$

となる．球の半径 ϵ が十分に小さく，第二項を無視できると仮定する．上記の方程式の左辺を，ϵ 球内の点の数で近似すると，密度推定量

$$\hat{f}(z;\epsilon) = \frac{k_\epsilon}{nc_p\epsilon^p} \tag{4}$$

を得る．k_ϵ は ϵ 球に入るサンプルの数である．ここで，検査点 z からの近傍の数を k に固定すると，k-NN 密度推定量 $\hat{f}^{nn}(z;k) = k/(nc_p\epsilon_k^p)$ が得られる．ϵ_k は検査点 z から k 番目のもっとも近い点までの距離である．$D\backslash\{x_i\}$ を用いて $\hat{f}_i^{nn}(x_i;k)$ を推定すると，k-NN エントロピー推定量

$$\hat{H}^{nn}(D;k) = -\frac{1}{n}\sum_{i=1}^{n} \ln \hat{f}_i^{nn}(x_i;k) \tag{5}$$

を得る．

3.2 SRE（Simple Regression Entropy Estimator）

式（3）の二次展開と単線形回帰に基づいたエントロピー推定を説明する [7]．z 付近の ϵ 球の確率質量は

$$q_z(\epsilon) = c_p f(z)\epsilon^p + \frac{p}{4(p/2+1)}c_p \mathrm{Tr}\nabla^2 f(z)\epsilon^{p+2} + O(\epsilon^{p+4}) \tag{6}$$

と展開される．式（6）の左辺を，k_ϵ/n で近似して $c_p\epsilon^p$ で割ると

$$\frac{k_\epsilon}{nc_p\epsilon^p} = f(z) + C\epsilon^2 + O(\epsilon^4) \tag{7}$$

を得る．ただし $C = \frac{p\mathrm{Tr}\nabla^2 f(z)}{4(p/2+1)}$ とおいた．応答変数を $Y_\epsilon = \frac{k_\epsilon}{nc_p\epsilon^p}$，説明変数を $X_\epsilon = \epsilon^2$ とし，ϵ に対する高次項

Algorithm 1 SRE(D)

$my.dim = x$ の次元
$cp1 = (\pi^{my.dim/2})/\Gamma(my.dim/2+1)$
$N = D$ に含まれるデータ数
$fz = NULL$
$my.n = N - 1$
for $i = 1$ *to* N **do**
  $z = x_i$
  $tmp.x = D$ のうち x_i 以外
  $dst = \| tmp.x - z \|$
  $eps = dst$ からランダムに m 個
  $tmp = dst$ の各要素に対して eps の各要素よりも小さいなら 1 そうでないなら 0
  $n.eps = tmp$ の各行を合計
  $ys = n.eps/(my.n * cp1 * eps^{my.dim})$
  $x = eps^2$
  $model = ys$ と x を用いて線形モデルによる回帰
  fz に $model$ の切片値を追加
end for
$mean(-log(fz))$

を無視すると，線形関係

$$Y_\epsilon \simeq f(z) + CX_\epsilon \tag{8}$$

を得る．式（8）は (X_ϵ, Y_ϵ) に関して線形モデルとみなすことができる．これらの変数は，ϵ の値によって変化する．半径の集合 $E = \{\epsilon_i\}_{i=1}^m$ を取り，観測されたサンプル対 $\{(X_\epsilon, Y_\epsilon)\}$ に関して二乗残差の和

$$R = \frac{1}{m}\sum_{\epsilon\in E}(Y_\epsilon - f(z) - CX_\epsilon)^2 \tag{9}$$

を最小化することによって $f(z)$ および C を推定できる．線形モデルの切片は z における確率密度関数の推定値である．サンプル x_i を用いずに式（9）を解くことによって得られた推定値を $\hat{f}_i^s(x_i)$ とする．leave-one-out 推定により，エントロピー推定量

$$\hat{H}^s(D) = -\frac{1}{n}\sum_{i=1}^n \ln \hat{f}_i^s(x_i) \tag{10}$$

を得る．

SRE のアルゴリズムを Algorithm 1 に示す．$D = \{x_i\}_{i=1}^n$ データ数 n のデータセット．

3.3 DRE（Direct Regression Entropy Estimator）

式（2）に式（8）を代入する，エントロピーの直接推定に基づいたエントロピー推定を説明する [7]．ϵ が固定されていると仮定し，検査点 $x_i \in D$ で式（8）を考える．$Y_\epsilon = \frac{k_\epsilon}{nc_p\epsilon^p}$ と $C = \frac{p\mathrm{Tr}\nabla^2 f(x_i)}{4(p/2+1)}$ は検査点 x_i に依存するため，それぞれを Y_ϵ^i と C^i と表す．式 (2) に基づいてエントロピー推定量を導出するために，$Y_\epsilon^i = f(x_i) + C^i X_\epsilon$ の対数の負を考える．すべてのサンプル点 $x_i \in D$ に関して平均することによって

$$-\frac{1}{n}\sum_{i=1}^n \ln Y_\epsilon^i = -\frac{1}{n}\sum_{i=1}^n \ln\{f(x_i) + C^i X_\epsilon\}$$

Algorithm 2 DRE(D)

$my.dim = x$ の次元
$cp1 = (\pi^{my.dim/2})/\Gamma(my.dim/2+1)$
$N = D$ に含まれるデータ数
$DMat = D$ の距離行列
$H = DMat$ の各行の中央値
$yh = NULL$
$x = NULL$
$lenH = 0$
for i = 1 to length(H) **do**
  $h = H_i$
  $tmp = DMat$ の各要素に対して h より大きいなら 0 そうでないなら 1
  $tmp = tmp$ の各列を合計
  yh に $log(length(tmp) * cp1) + my.dim * log(h) - mean(log(tmp))$ を追加
  x に h^2 を追加
  $lenH = lenH + 1$
  if $lenH \geq m$ **then**
    $break$
  end if
end for
$model = yh$ と x を用いて線形モデルによる回帰
$model$ の切片値

$$\begin{aligned}
&= -\frac{1}{n}\sum_{i=1}^n \ln f(x_i) - \frac{1}{n}\sum_{i=1}^n \ln\left(1 + \frac{C^i X_\epsilon}{f(x_i)}\right) \\
&\simeq -\frac{1}{n}\sum_{i=1}^n \ln f(x_i) - \frac{1}{n}\left(\sum_{i=1}^n \frac{C^i}{f(x_i)}\right) X_\epsilon
\end{aligned}$$

を得る．最後の式は $\ln(1+x)$ の一次テイラー展開による．上記の式の第一項は，エントロピーの推定値 (2) である．したがって，$\bar{Y}_\epsilon = -\frac{1}{n}\sum_{i=1}^n \ln Y_\epsilon^i$，$H(D) = -\frac{1}{n}\sum_{i=1}^n f(x_i)$，$\bar{C} = -\frac{1}{n}\sum_{i=1}^n \frac{C^i}{f(x_i)}$ とすると

$$\bar{Y}_\epsilon = H(D) + \bar{C}X_\epsilon \tag{11}$$

を得る．式（11）はサンプル対 $\{(X_\epsilon, \bar{Y}_\epsilon)\}_{\epsilon\in E}$ に関して線形回帰モデルとみなすことができる．ここで，SRE と同様の方法で，半径の集合 $E = \{\epsilon_i\}_{i=1}^m$ を取り，二乗残差の和を最小化することによって，線形モデルの切片としてエントロピー推定量 $\hat{H}^d(D)$ を得られる．

DRE のアルゴリズムを Algorithm 2 に示す．$D = \{x_i\}_{i=1}^n$ データ数 n のデータセット．

3.4 EPI（Entropy Estimator with Poisson-noise structure Identity-link regression）

ポアソン誤差構造の線形回帰に基づくエントロピー推定を説明する [8]．式（6）の左辺を再び k_ϵ/n で近似する．次に，式の両辺に n をかけて

$$k_\epsilon \simeq c_p n f(z)\epsilon^p + c_p n\frac{p}{4(p/2+1)}\mathrm{Tr}\nabla^2 f(z)\epsilon^{p+2} \tag{12}$$

を得る．式 (12) は $i > 0$ で，説明変数は $X = (\epsilon^p, \epsilon^{p+2})$，応答変数は $Y = k_\epsilon$ で定義され，一般化線形モデル $Y = \beta^{\mathrm{T}}X$

Algorithm 3 EPI(D)

1: $my.dim = x$ の次元
2: $N = D$ のデータ数
3: $cp1 = (\pi^{my.dim/2})/\Gamma(my.dim/2+1)$
4: $fz = NULL$
5: $my.n = N - 1$
6: **for** $i = 1$ *to* N **do**
7: $z = x_i$
8: $x = D$ のうち x_i 以外
9: $dst = \| x - z \|$
10: $esp = dst$ からランダムに m 個
11: $tmp = dst$ の各要素に対して eps の各要素よりも小さいなら 1 そうでないなら 0
12: $n.eps = tmp$ の各行を合計
13: $y = n.eps$
14: $x1 = eps^{my.dim}$
15: $x2 = eps^{my.dim+2}$
16: $model = y$ と $x1$、$x2$ を用いて一般化線形モデルによる回帰
17: fz に $model$ の第一係数値 $/(cp1 * my.n)$ を追加
18: **end for**
19: $mean(-log(fz))$

を当てはめられる．k_ϵ はカウントデータであるので，ポアソン誤差構造を選択する．ここでは，ポアソン回帰のリンク関数は恒等リンク関数を採用している．具体的には，半径のセット $E = \{\epsilon_i\}_{i=1}^m$ に対して，$\{(Y_i, X_i)\}_{i=1}^m$ を計算する．$X_i = (\epsilon_i^p, \epsilon_i^{p+2})$，$Y_i$ は検査点を中心とする ϵ_i 球内のサンプル数である．次に，$\beta = (\beta_1, \beta_2)$ に関してポアソン分布の尤度関数

$$L(\beta) = \prod_{i=1}^{m} \frac{e^{-X_i^{\mathrm{T}}\beta}(X_i^{\mathrm{T}}\beta)^{Y_i}}{Y_i!} \tag{13}$$

を最大にする．第一係数 β_1 の推定値 $\hat{\beta}_1$ を $c_p n$ で除算し，z における確率密度関数値の推定値を $\hat{f}(z) = \hat{\beta}_1/(c_p n)$ として得る．この手続きを各 $X \in D$ に対して leave-one-out 法で行うことで，エントロピー推定量が SRE と同様に求められる．

EPI のアルゴリズムを Algorithm 3 に示す．$D = \{x_i\}_{i=1}^n$ データ数 n のデータセット．

3.5 KDE（Kernel Density Estimation）

カーネル密度推定 [9] を用いたエントロピー推定を説明する [7]．カーネル密度推定によって推定された確率密度関数の値を式（2）に代入することによって，エントロピーを得られる．密度関数は，ガウスカーネル関数を持つカーネル密度推定によって推定される．バンド幅の選択には，最小自乗クロスバリデーション [10] を採用している．

KDE のアルゴリズムを Algorithm 4 に示す．$D = \{x_i\}_{i=1}^n$ データ数 n のデータセット．

4. 設計と実装

4.1 並列化

SRE について説明する．SRE では，データの中から一つ

Algorithm 4 KDE(D)

1: $N = D$ のデータ数
2: $fz = NULL$
3: **for** $i = 1$ *to* N **do**
4: $z = x_i$
5: $tmp.x = D$ のうち x_i 以外
6: $bw = tmp.x$ を用いてバンド幅を計算
7: $kde = tmp.x$ をデータ、bw をバンド幅としてカーネル密度推定を行って実現値 z の確率密度を推定
8: fz に kde を追加
9: **end for**
10: $mean(-log(fz))$

Algorithm 5 parallel SRE(D)

1: Algorithm 1 の 1 ～ 5 と同様
2: $i =$ コミュニケータにおけるプロセスランク $+1$
3: **while** $i \leq N$ **do**
4: Algorithm 1 の 7 ～ 16 と同様
5: $i = i +$ コミュニケータに含まれるプロセス数
6: **end while**
7: $fz = gather(fz)$
8: $mean(-log(fz))$

を選んでほかのデータとの距離を計算し，それを用いて回帰分析を行うということを繰り返し行っている．ここでは，データの中から選ぶデータをプロセスごとに異なるようにすることで回帰分析を並列に行い，最後に確率密度推定値を集めている．これにより，繰り返しの回数が 1/（プロセス数）になる．以下の Algorithm 5 は並列化された SRE プログラムのアルゴリズムである．$D = \{x_i\}_{i=1}^n$ データ数 n のデータセット．Algorithm 1 の FOR 文を WHILE 文に変更し，i の初期値をコミュニケータにおけるプロセスランク，増加値をコミュニケータに含まれるプロセス数にしている．

DRE について説明する．DRE では，データセットの距離行列の中央値から一つを選んで，それと距離行列から回帰分析に用いるためのデータを生成するということを繰り返し行っている．ここでは，選ぶ中央値をプロセスごとに異なるようにすることでデータ生成を並列に行い，最後にそのデータを集めている．これによって，繰り返しの回数が 1/（プロセス数）になる．以下の Algorithm 6 は並列化された DRE プログラムのアルゴリズムである．$D = \{x_i\}_{i=1}^n$ データ数 n のデータセット．SRE と同様に FOR 文を WHILE 文に変更し，i の初期値をコミュニケータにおけるプロセスランク，増加値をコミュニケータに含まれるプロセス数にしている．

EPI については，SRE とほぼ同様である．

KDE について説明する．KDE では，データの中から一つを選んでカーネル密度推定を行い，実現値が選んだデータである確率密度を推定するということを繰り返し行っている．ここでは，カーネル密度推定をする際に選ぶデータをプロセスごとに異なるようにすることでカーネル密度推

Algorithm 6 parallel DRE(D)

```
1: Algorithm 2 の 1 ∼ 8 と同様
2: i = コミュニケータにおけるプロセスランク +1
3: while i ≤ length(H) do
4:     Algorithm 2 の 10 ∼ 18 と同様
5:     i = i+ コミュニケータに含まれるプロセス数
6: end while
7: yh = gather(yh)
8: x = gather(x)
9: Algorithm 2 の 20, 21 と同様
```

Algorithm 7 parallel KDE(D)

```
1: Algorithm 4 の 1, 2 と同様
2: i = コミュニケータにおけるプロセスランク +1
3: while i ≤ N do
4:     Algorithm 4 の 4 ∼ 8 と同様
5:     i = i+ コミュニケータに含まれるプロセス数
6: end while
7: fz = gather(fz)
8: mean(-log(fz))
```

定を並列に行い，最後に確率密度推定値を集めている．これにより，繰り返しの回数が 1/（プロセス数）になる．以下の Algorithm 7 は並列化された KDE プログラムのアルゴリズムである．$D = \{x_i\}_{i=1}^{n}$ データ数 n のデータセット．これも SRE と同様に，FOR 文を WHILE 文に変更し，i の初期値をコミュニケータにおけるプロセスランク，増加値をコミュニケータに含まれるプロセス数にしている．

4.2 分散データ

SRE について説明する．プロセスがデータをブロードキャストし，各プロセスはブロードキャストされたデータのうち一つと自分の持つデータとの距離を計算する．計算した距離の中からランダムに m 個選び，選んだ値 ϵ をプロセスで分散して所持する．次にプロセスが持つ ϵ をブロードキャストし，各プロセスは所持している距離のうち ϵ よりも小さいものの数をカウントして合計する．カウントの合計は ϵ をブロードキャストしたプロセスが持つ．各プロセスはそれぞれが持つ ϵ とカウントの合計を用いて回帰分析に必要な (X_ϵ, Y_ϵ) を計算する．回帰分析を行う部分は分散データを対象にできていないので，(X_ϵ, Y_ϵ) をあるプロセスに集めて回帰分析を行う．以上のことを繰り返し行い，最後に確率密度推定値の対数の負の平均を計算してエントロピーを求める．これにより，データの距離計算と ϵ 球内の点の数のカウント，(X_ϵ, Y_ϵ) の計算を並列化できる．

以下の Algorithm 8 は分散データを対象とした SRE プログラムのアルゴリズムである．$X_i = \{x_j\}_{j=1}^{n}$ プロセスランク i が持つデータ数 n．

EPI は SRE と同様である．

KDE について説明する．プロセスがデータをブロードキャストし，各プロセスはブロードキャストされたデータ

Algorithm 8 parallel SRE dd(X)

```
1: Algorithm 1 の 1 ∼ 5 と同様
2: for i = 0 to ( コミュニケータに含まれるプロセス数 −1) do
3:     y = broadcast(X_i)
4:     for j = 1 to (y のデータ数 ) do
5:         z = y_j
6:         tmp.x = X
7:         dst =‖ tmp.x − z ‖
8:         eps = dst からランダム (全プロセスの eps の数の合計が m
           個 )
9:         for l = 0 to ( コミュニケータに含まれるプロセス数 −1)
           do
10:            eps_b = broadcast(eps_l)
11:            tmp = dst の各要素に対して eps_b の各要素よりも小さ
               いなら 1 そうでないなら 0
12:            n.eps_l = reduce(tmp の各行の合計 )
13:        end for
14:        Algorithm 1 の 13, 14 と同様
15:        ys = gather(ys)
16:        x = gather(ys)
17:        Algorithm 1 の 15, 16 と同様
18:    end for
19: end for
20: Algorithm 1 の 18 と同様
```

Algorithm 9 parallel KDE dd (X)

```
1: Algorithm 4 の 1, 2 と同様
2: for i = 0 to ( コミュニケータに含まれるプロセス数 −1) do
3:     y = broadcast(Xi)
4:     for j = 1 to (y のデータ数 ) do
5:         z = y_j
6:         tmp.x = X
7:         bw = tmp.x を用いてバンド幅を計算
8:         kde = tmp.x をデータ、bw をバンド幅としてカーネル密
           度推定を行って実現値 z の確率密度を推定
9:         fz に kde を追加
10:    end for
11: end for
12: Algorithm 4 の 10 と同じ
```

のうち一つを実現値とした確率密度をカーネル密度推定によって求める．これを繰り返し行い，最後に確率密度推定値の対数の負の平均を計算する．以下の Algorithm 9 は分散データを対象とした KDE プログラムのアルゴリズムである．

分散データを対象としたカーネル密度推定について説明する．X は各プロセスが持つ異なるデータ，確率点 z は共通である．カーネル密度推定は各標本点を平均とした密度分布を足し合わせて行う．ここでは，確率密度計算を並列に行い，最後に合計している．以下の Algorithm 10 は分散データを対象としてカーネル密度推定を行うプログラムのアルゴリズムである．$X_i = \{x_j\}_{j=1}^{n}$ プロセスランク i が持つデータ数 n．

5. 性能評価

本章では，実装した並列エントロピー推定プログラムの

Algorithm 10 parallel kde (データ X, バンド幅 bw, 確率点 z)

1: $mus = X$
2: $sigmas = bw$
3: $props =$ 重み
4: **for** $i = 1toX$ のデータ数 **do**
5: $dno =$ 平均 mus_i、標準偏差 $sigmas_i$ の正規分布の確率点 z の確率密度
6: $dens = dens + props_i * dno$
7: **end for**
8: $reduce(dens)$

表 1　実行環境
Table 1　execution environment

CPU	Intel (R) Xeon (R) CPU E5-2665 0 @ 2.40GHz
コア数	8 × 2
メモリ	64GB
OS	CentOS release 6.9 (Final)
R	version 3.4.2
OpenMPI	version 1.10.3rc4
pbdMPI	version 0.3-3

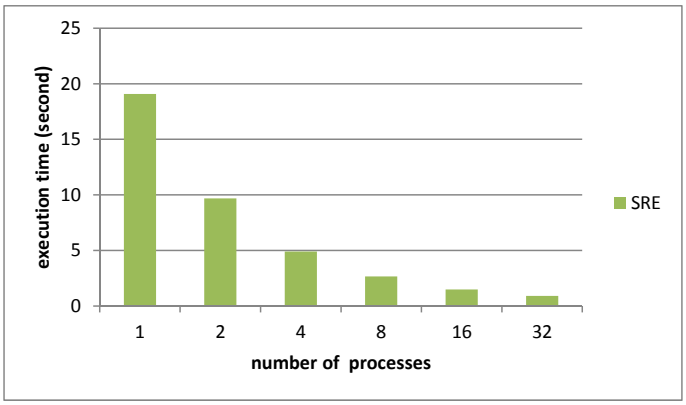

図 1　並列 SRE の実行時間
Fig. 1　execution time of parallel SRE

評価について示す.

5.1　実行環境

評価実験を行う環境を**表 1** に示す．実験に用いたのは 4 ノードである.

5.2　並列エントロピー推定プログラムの性能評価

並列エントロピー推定プログラムの実行時間を測定する．プロセス数を変化させた場合の実行時間を示す．縦軸が実行時間，横軸がプロセス数を示している．データは 1 次元である.

並列 SRE，DRE，EPI，KDE プログラムの実行時間をそれぞれ図 1，2，3，4 に示す．各プロセスが 1000 のデータセットを持っている.

図 1，2，3，4 より，プロセス数が倍になると約 2 倍高速化していることがわかる。

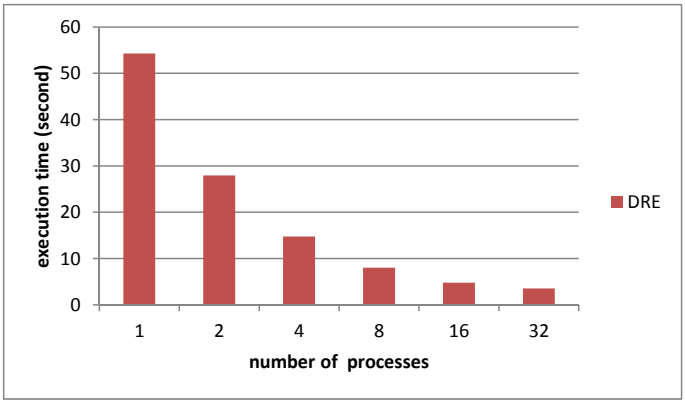

図 2　並列 DRE の実行時間
Fig. 2　execution time of parallel DRE

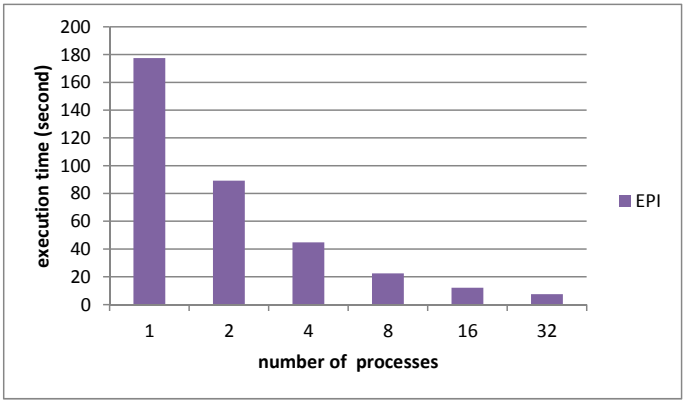

図 3　並列 EPI の実行時間
Fig. 3　execution time of parallel EPI

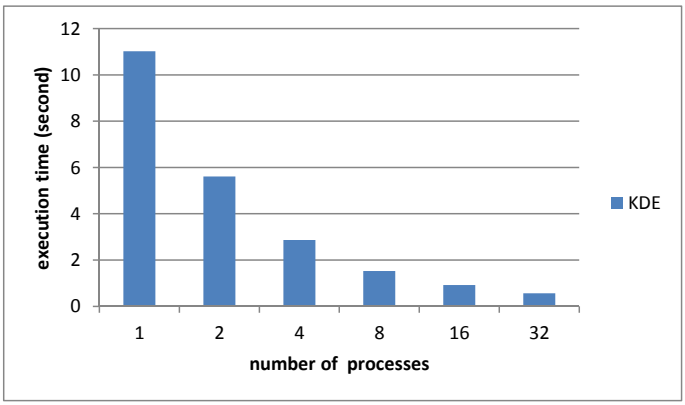

図 4　並列 KDE の実行時間
Fig. 4　execution time of parallel KDE

分散データを対象とした並列 SRE，EPI，KDE プログラムの実行時間を図 5，6，7 に示す．各プロセスは 1 万 /（プロセス数）のデータセットを持っている.

図 5，6 より，プロセス数が増加するごとに高速化していることがわかるが、1，3 で示した並列 SRE，EPI ほどではない。プロセス数が 1 から 2 になった場合，sre では 1.44 倍，epi では 1.33 倍高速化している．図 7 より，カーネル密度推定を行う kde 部分はプロセス数が倍になると約

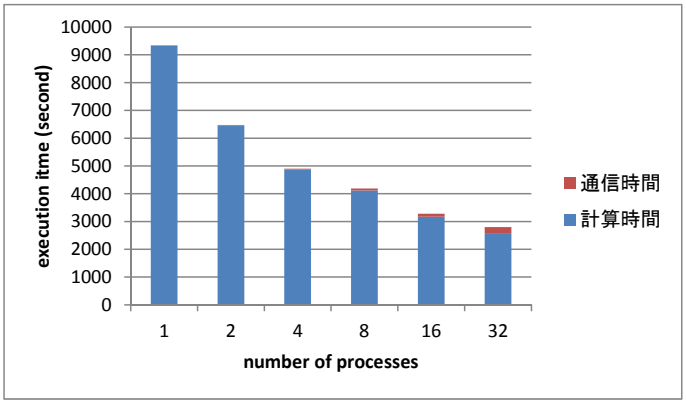

図 5　分散データを対象とする並列 SRE の実行時間
Fig. 5　execution time of parallel SRE for distributed data

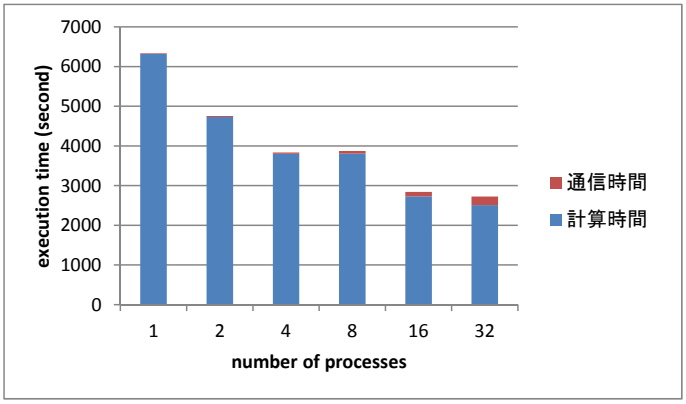

図 6　分散データを対象とする並列 EPI の実行時間
Fig. 6　execution time of parallel EPI for distributed data

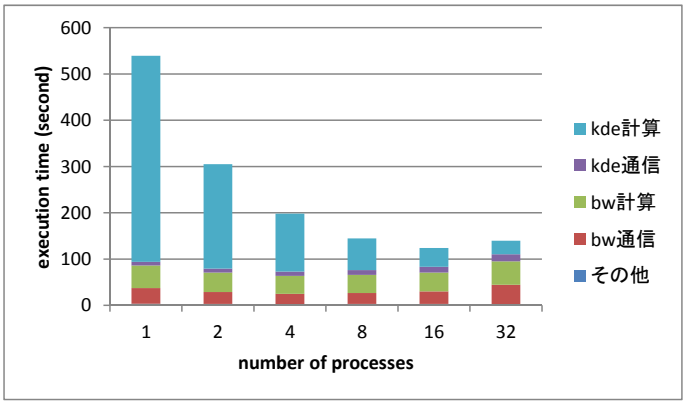

図 7　分散データを対象とする並列 KDE の実行時間
Fig. 7　execution time of parallel KDE for distributed data

2 倍高速化していることがわかる．バンド幅を求める bw 部分に関しては，分散データを対象にできるようにしているが並列化はできていない．従って，高速化はされず通信時間の分だけ遅くなっている．

6. 結論

pbdMPI を用いてエントロピー推定プログラムの並列化を行った場合，ノード（プロセス）数の増加に伴い全体の実行時間が短くなることが確認できた．

本研究では，エントロピー推定プログラムの並列化を行い評価した．各プロセスが全データを所持してる場合を想定した並列化 SRE，DRE，EPI，KDE はすべて高速化に関して高い性能を示した．分散データを対象とした並列化 DRE，EPI は，ノード（プロセス）数の増加に伴い実行時間は短くなっているが，全データを所持している場合の並列化プログラムと比較して高速化の面では劣っている．これは，分散データを対象とできるようにしたために，プログラム内のループの増加や並列化できていない部分の複雑化などが理由だと思われる．分散データを対象とした並列化 KDE は，カーネル密度推定に関しては全データを所持している場合を想定した並列化プログラムに近い性能を示したが，バンド幅計算は並列化されていないので短くなっていない．分散データを対象とする並列化プログラムは高速化の面で劣っているが，より大きなデータセットを対象とすることができる．

ノード（プロセス）数がさらに増加した場合，途中から通信部分や並列化できていない部分がボトルネックとなりほとんど高速化しなくなる．分散データを対象とした並列化プログラムは，全データを所持している場合を想定した並列化プログラムと比べ，ボトルネックとなる部分が多いため，高速化しなくなるのも早いと思われる．共有メモリでの並列化を行い分散メモリの場合よりも通信時間を短くするなど，ボトルネックとなる部分の実行時間を短くする方法はあるが，無くすことはできない．

以上より，実装した並列エントロピー推定プログラムは従来のエントロピー推定プログラムよりも高い性能を得られることを確認できた．

今後の研究としては，まず分散データを対象とした並列化 DRE はまだできていないのでその実装を行う．

また，本研究ではメッセージパッシングを用いて分散メモリでの並列化を行ったが，共有メモリでの並列化も検討する．

謝辞

本研究の一部は，JST-CREST ACA20935，JSPS 科研費 16K16108，JST-CREST JPMJCR1303，JST-CREST JPMJCR1413，JSPS 科研費 JP17H01748 による．

参考文献

[1] 平井有三：はじめてのパターン認識，森北出版 (2012).
[2] Paul Teetor 著，木下 哲也訳：R クックブック，オライリージャパン，オーム社 (2011).
[3] Schmidberger, M., Morgan, M., Eddelbuettel, D., Yu, H., Tierney, L. and Mansmann, U.: State-of-the-art in Parallel Computing with R, Technical Report 47, Ludwig-Maximilians-Universitat Munchen (2009).
[4] Chen, W.-C., Ostrouchov, G., Schmidt, D., Patel, P. and Yu, H.: *A Quick Guide for the pbdMPI Package*, pbdR Core Team.

[5] Tom 著，玉川竜司，兼田 聖士訳：Hadoop 第 2 版，オライリージャパン，オーム社 (2017).
[6] Virgillito, A.: Implementing MapReduce programs: RHadoop, (online), available from ⟨https://circabc.europa.eu/sd/a/6e9bc4ea-502b-4747-a82c-9391fce49c4f/Day%203-01-RHadoop.pdf⟩ (accessed).
[7] Hino, H., Koshijima, K. and Murata, N.: Non-parametric entropy estimators based on simple linear regression, *Computional Statistics and Data Analysis 89*, pp. 72–84.
[8] Hino, H., Akaho, S. and Murata, N.: An Entrory Estimator Based on Polynominal Regression with Poisson Error Structure, *ICONIP* (2016).
[9] Wand, M. and Jones, M.: *Kernel Smoothing*, Chapman Hall/CRC (1994).
[10] 谷崎久志：密度関数のカーネル推定量におけるバンド幅の選択について：モンテカルロ実験による小標本特性，神戸大学（オンライン），入手先 ⟨http://www2.econ.osaka-u.ac.jp/ tanizaki/cv/papers/kernel.pdf⟩ （参照）.