様　式　Ｃ－１９、Ｆ－１９、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

平成 ２８ 年　 ６ 月 １０ 日現在

機関番号：１２６１２
研究種目：基盤研究(C)（一般）
研究期間：2012～2015
課題番号：２４５００１０９

研究課題名（和文）大規模データ処理基盤におけるデータ空間の生成演算と更新方式に関する研究

研究課題名（英文）A Study on Data-Space Generating Operations on Massive Data Platforms

研究代表者
大森　匡（OHMORI, Tadashi）

電気通信大学・その他の研究科・教授

研究者番号：３０２３３２７４

交付決定額（研究期間全体）：（直接経費）　　2,300,000円

研究成果の概要（和文）：大量データ上の多対多関係情報の抽出を行う類似結合演算の計算技法は従来から多くあるが，mapreduce上では，計算モデルに特有な性能の不安定さがあり，大量データ処理演算として確立していなかった．本研究では，この不安定さを解決する汎用的なアルゴリズム効率化戦略として，(i) 多対多等結合で微小なデータ偏りが引き起こす負荷偏在を解決する技法HSJ+BR，および，(ii) 類似結合算法が使うレコードコピー量とshuffleコストを抑制する2段階ハッシュ分割戦略による効率化技法，の2つを提案して，編集距離結合など多様な類似結合計算のmapreduce上の不安定さを解決する戦略として有効性を示した．

研究成果の概要（英文）：Similarity joins on massive datasets are useful operations to detect many-to-many relationship residing in target datasets. However many join algorithms on various similarity functions are known to have unstable performance on map/reduce systems. The objective of this research is to clarify reasons of this unstablity, and to solve it. To do so, the research proposes two new algorithmic frameworks. One is the hybrid-hash join enhanced with bucket-regrouping techniques, named HSJ+BR. It solves unexpected unbalance between reducers without intermediate mapreduce jobs. The other is called two-stage hash-partitioning strategy. It can greatly reduce the shuffle overhead caused by too much record-replication associated with many similarity join algorithms. Using these two frameworks, it is shown that stable and efficient performance of similarity joins on map/reduce systems (where, as typical cases, m-to-n equi-join and edit-distance join are used) is achieved.

研究分野：データベース・データ工学

キーワード：大規模データ処理　MapReduce　類似結合　多対多関係　編集距離　ハッシュ結合

１．研究開始当初の背景

本研究開始当初（2012）年においては，mapreduce が新しいデータ処理基盤として主役になりつつあったものの，主な用途は古典的なSQLログ集計か検索エンジン用の逆引きインデックス生成，行列計算であった．一方，SIGMOD などの国際会議では不定形な新しい巨大データを対象にクラスタリングや重複除去などを意図して，様々な類似結合がmapreduce上で試行され始めていた．しかし，mapreduce に固有な性能の不安定さがあり，求める情報の品質・類似関数・類似度閾値に応じて先行研究における各類似結合の提案技法は効率低下が大きいと指摘しあう状況であり，Afrati,Ullman(ICDE‘12)らの形式化は単純すぎて，安定した効率的実行方式は不明なままであった．結果，多様な類似結合演算の並列データ処理方式が確立せず，新しい応用にまで議論が進まない状況であった．

２．研究の目的

そもそもmapreduce自体はWebデータ集合上の検索エンジン用途の分散計算システムであり，類似結合のような情報空間生成演算を狙って作った機構ではない．そこで本研究では，データベース分野で必要な情報空間生成演算として巨大データRとSの間，あるいはRに内在する多対多関係の抽出演算を代表に選び，様々な類似結合演算をその実行演算と考えて，mapreduce 上でのデータ偏りへの対処法やshuffleコストに代表される動作特性の明確化，効率的実行手法を明らかにすることを目的の主とした．

具体的には，巨大データ R と S，または R と R の間に成立する，制約の緩い多対多等結合で表現できる演算を対象にした．これは，SQL 風の書式では find (r,s) from RxS where g(r)=g(s) and f(r,s)の形で書ける演算である．つまり，何らかの等結合制約でレコードr,s 間の組み合わせをつくり，そこで判定処理f(r,s)を計算する演算である．この演算は，一般的な結合制約条件θを使ったθ結合や，集合類似度，編集類似度の結合演算を含む．

上記の多対多結合演算は，従前のmapreduce上の結合演算の基本が1対多関係（１：ｎ）結合を想定したHash Joinを使うことに比べ，大きな変更である．本研究では，この多対多関係抽出演算を対象にして，mapreduce 上でレコード件数が大規模化したときの結合に伴うデータ偏り処理の程度と解決アルゴリズムを示す．さらに，既存の個々の類似結合演算実行方式が引き起こすmapreduce処理のmap，shuffle，reduceの動作特性を明確にして，それへの対処法として幅広い範囲で安定して高速化できるアルゴリズムを提出することとした．

３．研究の方法

２．で設定した多対多関係結合演算は，時空間上の範囲結合やk近傍結合，集合類似度や編集距離類似度による結合など，多くのクラスをこの表現に変換して実行できる．この多様さに対処するため，研究方法として，多対多関係抽出の計算問題を，次の2つに分けて，動作特性の明確化と問題を解決するmapreduce 向けアルゴリズムを提案することにした：

(1)データ集合 R と S の間の多対多等結合におけるデータ偏り問題の解決

例えば，論文データ集合RとSを与えたとき，同じ期間で一致するキーワードを含むような論文のうち内容が近い組み合わせ(r,s)を全て求める問題は，find (r,s) from RxS where R.a=S.a and f(r,s)の形式で書ける．a は，期間とレコードが持つキーワードの１つの組み合わせである．（議論を簡単にするため，レコードあたりが持つキーワード数は高々5 個程度とし，関数 f()でｒとｓのコンテンツによる類似計算を行う，と仮定する．）

この計算で 1000 万件を超えるレコード集合を相手にすると，等結合条件となる属性値が50000件（0.5%）程度の偏りでも，reduceのCPU計算負荷は関数f()の実行回数で他のreduceに比して最大50000の二乗＝2.5G倍となり，著しく重くなる．従来の並列データベース用の既存技法やmapreduce向けの先行研究では，全データからランダムに数%サンプルをとって対応を決める．しかし，数%サンプルでは，巨大データに内在するわずかなデータ偏在で，かつ，結合演算計算時に膨大な計算偏りを引き起こすような偏りは検出できない．一方で，上で挙げた多対多結合演算は二つのデータ集合RとSの重なりを求めるための基本的な演算である．

本研究では，まず，上記の状況での多対多等結合を，mapreduce 上でどうやって負荷分散しつつ無駄な中間ジョブを入れずに計算すべきか，を考えた．具体的には，mapreduce向けに，バケット細分割・再グループ化によるHybrid Hash Joinを提案した．

(2)多様な類似関数による類似結合の実行方式が引き起こす mapreduce 処理の map，shuffle，reduce の動作特性の明確化と解決

類似結合で用いる類似度関数は，集合類似度や編集距離，距離公理を満たす距離関数，一般的なθ制約など非常に多い．そのためmapreduce モデルで類似結合を計算する算法の研究は多いが，類似関数に応じた各算法固有のパラメタ，対象として用いるデータの分布や規模，用いる類似度，などによって各算法の効率が大きく異なることが知られている（Ullman，ICDE’12）．類似結合演算の初出は，Chaudhuri らによる集合類似度を例にとった接頭辞(prefix)フィルタリング技法による1ノード用SQLサーバでの実行アルゴリズムである．Prefix-filtering 法は他の類

似度にも適用できることが知られている．そこで，本研究では，編集距離による類似結合を選び，prefix-filtering 法で編集距離結合を計算する mapreduce 用既存技法の 1 つ「ランドマーク結合」(以下，LMJ 法．Narita 他，DaWak’12) を例に，map/shuffle/reduce の計算負荷を調べることにした．そして，prefix-filtering 法に特有なレコードコピー量の組合せ的増加に伴う shuffle コストの増加と重複計算が問題の本質であることを明確にした後，mapreduce 向け類似結合実行の技法として，map/shuffle コストの削減と reduce 処理の効率化を狙った「2 段階ハッシュ分割技法」と呼ぶ汎用的な戦略を提案した．

４．研究成果

(1) mapreduce 上の多対多等結合におけるデータ偏りを解決するハイブリッドハッシュ結合技法 HSJ+BR の提案

本研究では，平成 24-25 年度に，mapreduce における多対多関係の等結合演算でデータが偏り，reduce 間の負荷バランスが著しく劣化して全体の処理時間が低下する現象を解決する結合演算の実行算法を考案した．想定するデータ処理は，find (r,s) from RxS where R.a=S.a and f(r,s)，という形式である．(雑誌論文①,学会発表③)

提案算法は，ハイブリッドハッシュ結合の mapreduce 向け変形の 1 つであり，Hybrid Skew Join with Bucket Regrouping(HSJ+BR)と呼ぶ．以下，HSJ+BR の概要と，動作特性を述べる．まず，mapreduce 上でＲとＳのハイブリッドハッシュ結合を単純に行う方法として，Hybrid Skew Join，(HSJ)を述べる．

HSJ は，mapreduce ジョブ１でビルドフェイズ，つまり，R のハッシュ分割(map 側)を行い，結合キーでソートしたテーブル（パーティション）Ri を reduce で作成，HDFS へ出力する．そして，2 ジョブ目で同一ハッシュ関数の下で S を map で分割・shuffle で結合キー上でソートした後，reduce 側で R の当該パーティション Ri をロードしておき，S のソートずみ結合キーのデータ入力により Ri のプローブ処理を行う．このプローブ処理時には，各 reducer にとっては，担当する R のパーティション Ri がメモリに入りきる必要がある．そのため，HSJ では，ビルド処理時に，Ri を，各 reducer の可能なメモリサイズ限界（たとえば，256MB）で作成する．しかし，この方法では，あるパーティション Ri に同一結合キー(高々5 万件程度の，データ 1000 万件から見ればごくわずかな数)のレコードが偏ったとき，S によるプローブ処理時に全く負荷分散できない．また，仮に，プローブ処理時の reducer N 個で負荷分散を工夫したとしても，当該パーティション全体が N 個の reducer へコピー転送され，ネットワーク負荷を増大させる．

そこで,HSJ+BR では,R のビルド処理時に，バケットと呼ぶ小さいデータ単位に細分割しておき，その分布を見て後段の S のプローブ処理の計算負荷を想定した優れたハッシュ分割関数を決定することにした．HSJ+BR の処理手順は，次の通りである：

(手順１) ビルド処理の map/reduce ジョブ Job1 において，reduce 側では，本来のパーティション(例えば 256MB)よりも小さいサイズのバケットと呼ぶ単位(例えば 64MB)にデータを細分割しておく．(例えば図１は，R を reducer 2 つ用にパーティション R1 と R2 に関数 h()を使って map で分割し，各 reducer が R1 を Ra から Rc の 3 バケットに,R2 を Rd, Re の 2 バケットに分割した場合である．バケットの分割関数は，元のハッシュ関数 h()と結合キーでソートした部分範囲の AND で与えらえる．このうち，バケット Rc に偏りがあると仮定する．)

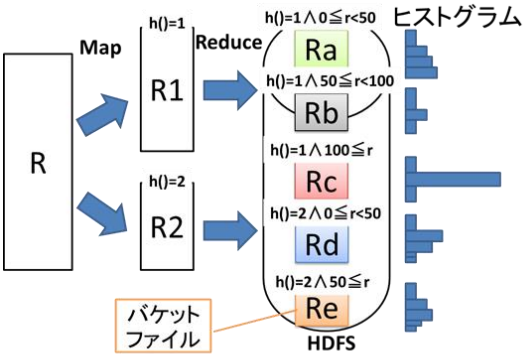

図１: HSJ+BRの手順1(ビルド処理)

(手順２) コントローラは，Job1 の結果を受けて，適当なバケット群を，システムの最大並列 reducer 数 Nr に応じて，負荷分散を考慮したパーティションへと再グループ化する構成案を決める．(図１の例では，Ra と Rb を 1 パーティション R1’にまとめ，Rc を１パーティション R2’，Rd と Re を１パーティション R3’としたハッシュ関数 H’を決定する．H’も，元のハッシュ関数 h()と結合キー上の範囲で決まる．例えば，R1’は，“h()=1 and 0<=r<100”で与えられる．) また，負荷の重いパーティション(図の Rc)への S のプローブ処理を N 個の reducer で並列実行するとし，N の値を決定する．

(手順３) 上で決めたハッシュ関数 H’を使って，mapreduce ジョブ Job2 を起動する．Job2 の map 側は，H’により S のハッシュ分割を行って当該 reducer へ送る．(ただし，N 個の reducer でプローブを並列処理すると決めたハッシュ値を持つ S の当該データについては，レコードごとにラウンドロビンで N 個の reducer へ転送する.) 各 reducer は，当

該の Ri’をロードした後，担当する S のソート済みデータを受けて Ri’にプローブ処理を行って結合処理を行う．

HSJ+BR の狙いは，手順２ で R のバケットの再グループ化を行うときに，計算が軽いと考えられるバケットはできるだけ 1 パーティションへと再グループ化し，一方，データ偏りのため結合処理が重いと予想されるバケット Rx についてはグループ化せず，1 バケットのままパーティションとして扱うことである．また，負荷の重いパーティションの結合を複数 reducer で並列実行するため，並列数 N を決めて S のプローブ処理の時に上手く N 個の reducer に重複処理の出ないように分散させている．

以上の手順を実現するため，HSJ+BR では，手順１で，R の各バケット内の結合キーの度数分布を生成し，その情報によって手順２のバケット再グループ化に相当するハッシュ分割関数 H’を決定することにした．また，負荷の重いパーティション Rx への S のプローブ処理の並列 reducer 数 N は，システムの最大並行 reducer 数 Nr 以下で自動設定するようにした．

評価実験では，データ処理用ノード 5 個，ノードあたり最大並行 map(or,reduce)タスク数各１の 5x5 タスク並行実行環境(Hadoop 0.20 系，Java coding)を試験環境とした．使用データは，R と S 共に，100 バイトレコード(結合キーは url を模した文字列 50 バイト)，1000 万件(1GB)の多対多結合である．偏りは結合キーあたり最大 5 万件，偏りの減衰率を 1/2 として，結合条件を R.key=S.key とし，結合出力は HDFS へは出力しないが，類似関数処理 f(r,s)のためにレコードあたりでは結合結果を主記憶に保持するとした．(例えば，同じ URL にアクセスした R とＳのユーザデータの比較で，各 s から見て最も似た r を求める，などがこの場合にあたる．)

全体の計算時間では，通常の Repartition Join や HSJ が 1400 秒程度なのに比べ，HSJ+BR は 600 秒程度となり，2.3 倍程度の高速化となった．ノードあたりの計算時間の最大・最小時間の比率でも，HSJ+BR はほぼ 1.2 倍以内に収めることができ，HSJ などの比率 10:1 程度に比して極めて安定した負荷分散を達成できた．

以上の結果から，単純な多対多等結合でもデータ件数が 1000 万件を超えると微小な比率のデータ偏りが大きな負荷偏在を引き起こすこと，そして，ハッシュ関数の動的再構成を使ったハッシュ結合技法 HSJ+BR が有効であること，が言えた．

国内外の研究状況から見ると，従来の mapreduce 上の結合処理がログ集計主体の 1:n 結合だったことに対し，本成果は，n:m 結合を考えた場合に本質的に生じる負荷偏在の問題を明示し，その解決策を提案したことになり，類似結合演算で生じる負荷偏在現象の分散処理技法として有効な戦略である．この点で，非常に貢献の高い成果である．

(2) 多様な類似関数による類似結合の実行方式が引き起こす mapreduce 処理の map, shuffle, reduce の動作特性の明確化と解決

次に我々は，編集距離による類似結合を例にとって，prefix-filtering 法に伴う mapreduce 向け類似結合算法の性能特性の明確化と安定的な効率化戦略を考えた（H26,H27.　学会発表①②）．具体的には，編集距離結合の実行方法の１つ「ランドマーク結合」(以下,LMJ 法.Narita 他,DaWaK ‘12)を例に，map/shuffle/reduce の計算負荷を調べた．主な成果として．次の２点がある：

① prefix-filtering 法に特有なレコードコピー量の組合せ的増加に伴う shuffle コストの増加と重複計算が問題の本質であることの明確化．

② 同レコードコピー量の削減を狙った mapreduce 向け類似結合の実行技法である 2 段階ハッシュ分割技法の提案

先に項目①について概説する．

編集距離結合(edjoin)は，文字列データの集合 X と Y の元 x, y で，編集操作回数（1 文字挿入，削除，置換）が t 以下の組(x,y)を全て求める問題である．本研究では，X=Y, つまり，データ集合Ｘの中に内在する類似レコード組の計算に限った．これは．遺伝子データやログ系列など，系列で表現されたレコード集合のクラスタ検出問題で使われる演算である．

Afrati,Ullman らの研究(ICDE’12)で既に，edjoin の戦略として x,y の部分文字列の組合せを生成して総当たりを避ける戦略(subsequence 法)がある．ＬＭＪ法は，レコードの接頭辞について subsequence 法を使った戦略であり，文字列 x と y の編集距離 ed(x,y)<=t なら，x,y の各先頭 t+q 文字のうち最低 q 文字の部分文字列が一致する，という規則を使う．これを mapreduce で行うと，q=16, t=2 なら，map 処理は 1 レコード x を 153 倍にして shuffle へ送る．(先頭から q+t=18 文字考え，そこから q=16 文字を抜き出すので combination(18,16)=153 種類の q 文字タグをつけて x を shuffle へ送出する)．結果,元データの 153 倍のデータ量を shuffle 処理へ回すので,非常に shuffle 負荷が高い．また，reducer が Nr 個あっても，153 倍にコピーされた異なるタグを持つレコード集合を，タグが一致するレコード間で結合検査するため，reducer 間で重複した(x,y)検査をす

る．本来なら，Nr 個ある reducer 間では重複した(x,y)組の照合検査をせず，できるだけ排他的に探索空間を分割して計算すべきである．

以上の現象は，Afrati/Ullman らの論文でも理論上は分かるはずだが，原論文は，単純化した照合戦略下での理論解析であり，個々の具体的な算法の戦略下でどうなるかは強調されていない．我々は既存技法である LMJ を mapreduce で実装し，遺伝子データ 30 万件(30MB)で試験して，shuffle 量が 1.8GB，reducer 間でもほとんど同じ解集合を計算して冗長計算の状況にあることを確認した．

次に，項目②を概説する．

上記①の問題は，prefix-filtering 法で類似結合を行う際の本質的な現象であり，編集距離結合のときに顕著に表れた問題である．本質的な解決策としては，map 側でレコード x を変換して送出するとき，x の行き先となる reduce タスクの総数 Cr を，システムが用意した reduce 総数 Nr よりも大幅に小さくなるように制御できれば良い．もちろん，同一のタグ（ラベル，と呼ぶ）を持つレコード群は必ず，同じ reducer へ送る必要がある．しかも，全 reducer 間でできるだけ処理を排他的に分割したい．

上記を実現するため，本研究では，2 段階ハッシュ分割と呼ぶ実行戦略を提案した．そして，ＬＭＪ法に基づいた編集距離結合への適用方法として，我々は，Ｑ１／Ｑ２分割法，と呼ぶ実行算法と，Label-Prefix 型分割法と呼ぶ算法，の２つを提案した．ここでは，本質的な算法であるＱ１／Ｑ２分割法を主に述べる．

Q1/Q2 分割法は，map 処理と reduce 処理とで，LMJ が使うパラメタ q の値を変えて，独立して絞り込み戦略を適用する方法である．すなわち， map 処理では q の値を小さく（例えば，q1＝３）設定し，レコード x の先頭から q1(=3)＋ t (=2)の 5 文字を考え，長さ q1=3 のラベルを生成して，combination(q1+t, q1)＝comb(5,3)=10 個のコピーを shuffle へ送出する．そして，reducer 側では，(q1 の）ラベルの一致するレコード群について，再度先頭から q2=16 など照合効率の高いパラメタを使って長さ q2 のラベルを生成し，ハッシュ結合を行う．ただし，reducer が，ｑ１ラベル(Lq1)の一致するグループのレコードから q2 ラベル(Lq2)を生成するときには，Lq2 の先頭 q1 文字が Lq1 と等しいものに限る．こうすることで，異なるｑ１ラベル群の間での重複した照合を避けられる．（この正当性は学会発表①②に記載.）

以上の方法によって，レコードあたりのコピー量を高々10 倍に抑え，かつ，照合効率の向上を reducer のメモリ内計算で q2 ラベル生成によって行い，しかも，探索空間全体での重複計算を行わない分散照合が達成できる．

一方で，Ｑ１／Ｑ２分割法は，レコードコピー集合の分配を先頭 q1+t 文字だけで決めるため，負荷の偏りを生じやすい．また，mapreduce のソートマージ機能をあまり積極的に利用していない．我々は，これらの点を考慮し，Label-Prefix 型分割法と呼ぶ方法も考案した．この方法は，map 側で Q2 ラベル Lq2 を生成するが，レコードの送出先は，Lq2 の先頭 q1 文字で決定し，同一 reducer へは x の実体は一回しか送らない，という方法である．これでも，shuffle で生じるレコード x のコピーは combination(q1+t, q1)になる．

Q1/Q2 分割法の本質は，１ＣＰＵ用に考案された類似結合演算の既存算法を mapreduce で用いるとき，レコードコピーが膨大な shuffle 負荷を起こすことを避けることを狙って，map 側ハッシュ分割と reduce 側で行うハッシュ分割相当の照合戦略を使うことである．この考えを，「2 段階ハッシュ分割戦略」と総称する．

評価実験では，既存の（しかし，無駄な計算処理をしないように我々が修正した）ＬＭＪ法(1 レベルＬＭＪと表記)と，提案した 2 段階ハッシュ分割戦略であるＱ１／Ｑ２分割法，および Label-Prefix 型分割法の 3 つを実装して，編集距離結合で評価した．試験環境は 5 ノードの Hadoop クラスタ(Hadoop1.x 系)，ノードあたり並行 map/reduce 数各 2，Hadoop Streaming 上の C(ＬＭＪ，Label-Prefix)か，Python から Cython で C に変換したコード(Ｑ１／Ｑ２分割法)で実行した．使用データは，長さ約 100 文字の遺伝子データ 30 万件，編集距離は 2 である．

結果，同じＬＭＪの照合絞り込み戦略に基づいた算法であっても，元の 1 レベルＬＭＪが実行時間 80 秒に対し，Ｑ１／Ｑ２分割法(q1=3, q2=14 の最良時)は 65 秒程度になった．注目すべきは，shuffle データ量の削減であり，元の 1 レベルＬＭＪの shuffle データ量が 5.1GB に対して，Q1/Q2 分割法のそれは 0.19GB になった．すなわち，2 段階ハッシュ分割戦略によって，shuffle 量が引き起こす mapreduce 処理の本質的なコスト問題を解決できた．その他にも，評価実験では，Q1/Q2 分割法や Label-Prefix 型分割法によって，reducer 間の重複した照合計算量を大きく削減できたことも示せている．従って，2 段階ハッシュ分割戦略は，編集距離のような類似結合計算算法を mapreduce 用に適用する際には極めて有効な戦略と言える．

我々は，平成 27 年度の後半に，上記の結果を受けて，Q1/Q2 分割法に代表される 2 段階ハッシュ分割戦略が他の類似結合算法にも適用できるかを考え，一部を試作した．その結果，例えば，編集距離結合の他の技法（遺伝子データ用の最も高速な技法であるスラ

イドソート）や，2−3 次元の低次元空間データ上の範囲結合では適用可能であることがわかった．（例えば，2 次元平面上の範囲結合なら，範囲サイズを map 用と reduce 内部で可変に設定すれば，成立する．）　一方，集合類似度のうち，set-overlap join を prefix-filtering 法で実行する場合では，類似度閾値がある程度高いと，1 レコードから生成される照合候補トークン数が 10 以下になる．このような場合，shuffle コストがシステムの全 reducer 数 Nr より大幅に小さくなるので，効率的な実行が自然に実現されることがわかる．従来の研究の評価実験では，この現象を明示的に制御しなかったため，研究ごとに各技法の性能が安定しなかったと結論づけられる．このように，2 段階ハッシュ分割戦略は，従来からの多様な類似結合算法を mapreduce に適用するときに汎用性の高い効率化戦略となっていると言え，国内外の研究状況から見ても意義のある研究成果となった．（なお，類似結合演算の更新を伴う利用方法の 1 つとしては，論文データ集合の集合類似度によるクラスタリングと新規データ集合の追加による再クラスタ計算を行う応用を試作し，有用な情報検索ができることを確認している．）

研究全体のまとめとしては，従来，大量データ上の多対多関係結合である類似結合演算の計算技法が多数，mapreduce 用に提示されてきたものの，mapreduce に特有な性能の不安定さのために，大量データ用の情報抽出演算として確立しているとは言えなかった．本研究では，この不安定さを解決する汎用的なアルゴリズム効率化戦略として，(i) R-S 多対多等結合での微小なデータ偏りが引き起こす負荷偏在を解決する技法 HSJ+BR，および，(ii) 類似結合算法が使うレコードコピー量と shuffle コスト増加を抑制する 2 段階ハッシュ分割戦略による効率化技法，の 2 つを提案し，これらによって，多様な類似結合算法の mapreduce 上での安定した効率化策を明示できた．この結果により，新たな大規模データ応用に有用なデータ空間生成演算としての類似結合計算技法の設計が mapreduce モデルで容易になり，今後のデータ応用にとっても有意義な成果を得ることができた．

５．主な発表論文等
（研究代表者、研究分担者及び連携研究者には下線）

〔雑誌論文〕（計１件）
①廣瀬　繁雄，大森　匡，新谷　隆彦，Map/Reduce におけるバケット再グループ化を用いたハイブリッドハッシュ結合アルゴリズム，日本データベース学会論文誌，(査読有)，　Vol.12, No.1, pp.61-66, 2013 年 6 月．

〔学会発表〕（計３件）
①大森　匡，今野　篤人，新谷 隆彦，MapReduce 上の編集距離結合における 2 段階ハッシュ分割技法の効果，FIT 情報技術フォーラム，D-045，(査読無)，(4 ページ)，情報処理学会，愛媛大学（愛媛県松山市），2015 年 9 月 15 日．

②今野　篤人，大森　匡，新谷　隆彦，MapReduce における編集距離結合の負荷分散・効率化方式，DEIM（第 7 回データ工学と情報マネジメントに関するフォーラム）2015, E7-6（6 ページ）,(査読無)，電子情報通信学会・情報処理学会，ホテル華の湯（福島県群山市），2015 年 3 月 4 日．

③廣瀬　繁雄，大森　匡，新谷　隆彦，Map/Reduce におけるバケット再グループ化を用いたハイブリッドハッシュ結合アルゴリズム，DEIM(第 5 回データ工学と情報マネジメントに関するフォーラム) 2013, F2-4 (7 ページ)，(査読無)，電子情報通信学会・情報処理学会，ホテル華の湯(福島県群山市), 2013 年 3 月 3 日．

６．研究組織
(1)研究代表者
大森　匡　（OHMORI, Tadashi）
電気通信大学・大学院情報システム学研究科・教授
研究者番号：30233274

(2)研究分担者
（なし）
研究者番号：

(3)連携研究者
（なし）
研究者番号：

以上．