情報処理学会論文誌　データベース　Vol.12　No.4　53–65 (Oct. 2019)

Pub/Sub環境におけるkNNデータモニタリングの分散処理のためのクエリ割当てアルゴリズム

鶴岡 翔平[1,a)]　西尾 俊哉[1,b)]　天方 大地[1,c)]　原 隆浩[1,d)]

受付日 2019年3月9日, 採録日 2019年7月8日

概要：近年，パブリッシュ/サブスクライブ（Pub/Sub）モデルに基づいてデータを配信するアプリケーションが多く存在し，ユーザは自身の興味のあるデータのみを取得する．また，スマートフォンやSNSの普及により，データやクエリ（サブスクリプション）の数が増加しており，単一のワーカですべての処理を行うとリアルタイム性を保持することが困難になっている．本稿では，Pub/Subモデルにおいて，複数のワーカにクエリを分散して処理を分担することにより，各クエリに対して，クエリのキーワードを含むデータの中でクエリの指定位置に最も近いk個のデータ（kNNデータ）を効率的にモニタリングする問題に取り組む．単純にクエリを分散させると，各ワーカの処理量に偏りが生まれ，その結果，新たなデータが生成されたとき，処理量の大きいワーカの処理時間が長くなり，全体の処理時間も長くなる．この問題を解決するため，各ワーカの処理時間が均一かつ全体の処理時間が短くなるようにクエリを分散するアルゴリズムを提案する．実データを用いた実験により，提案手法がベースライン手法と比べて数倍から数十倍高速にkNNデータを更新できることを確認した．

キーワード：Pub/Sub，kNNクエリ，分散処理

Query Assignment Algorithm for Distributed Processing of kNN Data Monitoring in Pub/Sub Systems

Shohei Tsuruoka[1,a)]　Syunya Nishio[1,b)]　Daichi Amagata[1,c)]　Takahiro Hara[1,d)]

Received: March 9, 2019, Accepted: July 8, 2019

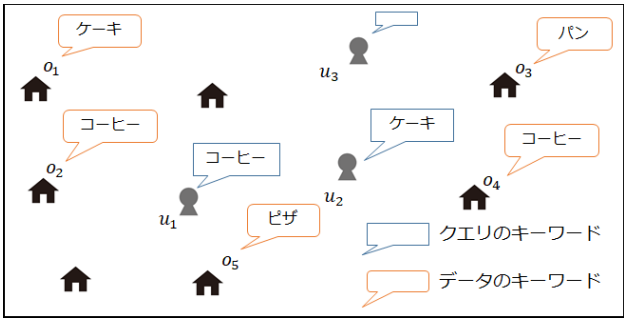
Abstract: Recently, in many applications, data objects have been generated based on Publish/Subscribe (Pub/Sub) model, and users receive only their preferable data objects. Due to the prevalence of smartphones and SNS, a large numbers of data objects and queries (subscriptions) have been published and registered, respectively. This makes a single worker difficult to monitor the result of each query. In this paper, we address the problem of monitoring k nearest neighbor data objects which contain the query keywords in a distributed Pub/Sub system. A straightforward approach, which assigns the queries to workers randomly, cannot balance the workloads of the workers. To solve this, we propose an algorithm that assigns queries to balance the workload of each worker and to minimize the total amount of workload. Our experiments using real datasets verify that our proposed algorithm can update the kNN results more than several times faster compared with baseline algorithms

Keywords: Pub/Sub, kNN query, distributed processing

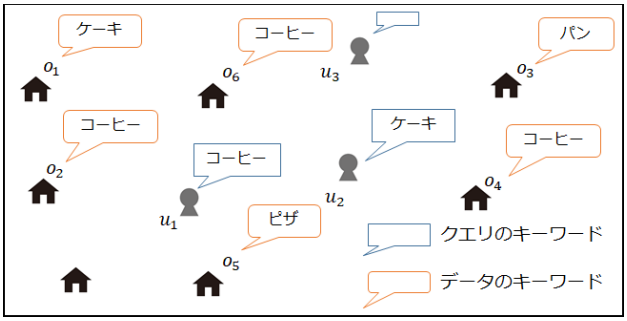
1. はじめに

近年，GPS機能付きのモバイル端末の普及により，ジオタグ付きのツイートなどの位置情報とキーワードを持つデータが大量に生成されている．それらのデータの中から，ユーザが自身の興味のあるデータを得るための手段と

[1] 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Suita, Osaka 565–0871, Japan
[a)] tsuruoka.shohei@ist.osaka-u.ac.jp
[b)] nishio.syunya@ist.osaka-u.ac.jp
[c)] amagata.daichi@ist.osaka-u.ac.jp
[d)] hara@ist.osaka-u.ac.jp



(a) 時刻 t



(b) 時刻 $t+1$

図 1　Pub/Sub モデルにおける kNN データモニタリングの例
Fig. 1　Example of kNN data monitoring in Pub/Sub system.

してパブリッシュ/サブスクライブ（Pub/Sub）モデルに基づくアプリケーションが多く存在する [1], [2], [3], [4], [5]. Pub/Sub モデルでは，ユーザは事前にワーカにクエリを登録し，ワーカは登録されたクエリに基づいてユーザにデータを配信する．また，大量にデータが存在する環境では，クエリのキーワードを含むデータの中で，クエリに最も近い k 個のデータ（kNN データ）を検索する kNN クエリが有用である [6], [7], [8], [9]．Pub/Sub モデルでは頻繁にデータが生成されるため．各クエリの kNN データは頻繁に変化する．

例 1． 図 1 は，Pub/Sub モデルにおいて，3 人のユーザが自身の興味のあるデータをモニタリングする例を示している．各ユーザはクエリ（位置情報およびキーワード）を登録しており，クエリに合う k 個のデータをモニタリングしている．$k=2$ と仮定すると，時刻 t において，ユーザ u_1 の kNN データは，「コーヒー」を含み，かつ u_1 に最も近いデータ o_2 および o_4 である．時刻 $t+1$ において，o_6 が生成されると，各クエリの kNN データが更新される．o_6 は，「コーヒー」を含み，o_4 よりも u_1 に近いため，u_1 の kNN データは，o_2 および o_6 に変化する．

ここで，スマートフォンや SNS の普及により，生成されるデータやワーカに登録されるクエリの数は増加している．データやクエリの数が増加すると，新たに生成されるデータに対するワーカの処理量が増加するため，単一のワーカですべての処理を行う場合にリアルタイム性を保証することが難しくなる [10], [11], [12], [13], [14]．そこで，本稿では，複数のワーカでクエリを分散して処理を分担することにより，各クエリの kNN データを効率的にモニタリングする問題に取り組む．

課題． 複数のワーカにクエリを分散して各クエリの kNN データを効率的にモニタリングするためには，クエリをどのようにワーカへ分散するかが重要である．単純な方法では，同じキーワードを持つクエリを 1 つのワーカで管理する方法が考えられる．しかし，あるキーワードが多くのデータやクエリに含まれる場合，そのキーワードを持つクエリを管理するワーカの処理量が大きくなってしまう．また，もう 1 つの方法として，クエリが存在する領域をワーカの数に等分割し，ある領域に含まれるクエリ集合を 1 つのワーカで管理する方法が考えられる．しかし，ある領域にデータやクエリが多く存在する場合，その領域に含まれるクエリを管理するワーカの処理量が大きくなってしまう．このように，単純にクエリを分散させると，各ワーカの処理量に偏りが生まれる．その結果，新たなデータが生成されたとき，処理量の大きいワーカの処理時間が長くなり，全体の処理時間も長くなる．

提案アルゴリズムの概要． 各クエリの kNN データを効率的にモニタリングするためには，新たなデータが生成されたときの各ワーカでの処理負荷を均一にし，全体の処理時間を最小化することが望ましい．そのため，新たなデータが生成されたときに予測される各ワーカの処理コスト（負荷）を定義する．あるワーカの負荷は，そのワーカで管理するクエリの負荷の合計と定義し，あるクエリの負荷はデータやクエリの位置情報とキーワードに基づいて計算される．そして，各ワーカの負荷を均一にしつつ，全体の負荷を小さくするようにクエリを分散するアルゴリズムを提案する．

具体的には，すべてのクエリの集合 Q を，集合の数が一定数以上になるまで，部分集合 Q_i に分割する．そして，各ワーカの負荷が均一になるように，Q_i に含まれる各クエリを管理するワーカを決定する．これにより，各ワーカの負荷を均一にしつつ，全体の負荷をより小さくするようにクエリを分散する．

貢献． 以下に本稿の貢献を示す．

- 複数のワーカでクエリを分散して処理を分担することにより，各クエリの kNN データを効率的にモニタリングする問題に取り組む．著者らの知る限り，この問題はこれまでに取り組まれていない．
- 各ワーカの負荷を均一にしつつ，全体の負荷を小さくするようにクエリを分散するアルゴリズムを提案する．
- 実データを用いた実験により，提案アルゴリズムの有効性を確認する．

本稿の構成． 2 章で関連研究について説明し，3 章で本稿の問題を定義する．4 章で提案アルゴリズムについて説明し，5 章で実データを用いた実験の結果を示す．最後に 6

章で本稿をまとめる．

2. 関連研究

本章では，位置およびキーワードを考慮した検索に関する関連研究，Pub/Sub モデルにおけるデータのモニタリングに関する関連研究，およびクエリの分散処理の関連研究を紹介する．

2.1 位置およびキーワードを考慮した検索

位置およびキーワードを考慮した検索に関する研究が近年さかんに行われている [6], [7], [8], [9], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24]．文献 [15], [16], [17], [18], [19], [20] では，位置およびキーワードに基づいて計算されたスコアの高い上位 k 個のデータ（Top-k データ）を効率的に検索する手法を提案している．たとえば，文献 [15] では，データの位置情報とキーワードを管理する IR-tree というデータ構造を提案している．IR-tree の葉ノードでは，データの集合およびデータ集合の MBR（Minimum Bounding Rectangle）を保持し，転置ファイルにより，データのキーワードの重みを保持する．中継ノードでは，自身の子ノードへのポインタと，子ノードの持つ MBR を保持し，転置ファイルにより，子ノードに含まれるデータのキーワードの重みを保持する．あるクエリの Top-k データを検索するとき，各ノードで保持する情報から，そのノードを根とする部分木で管理されるデータのスコアの上界値を推定し，上界値の高いノードから順に検索を行う．

文献 [6], [7], [8], [9] では，クエリのキーワードを含むデータの中で，クエリの指定位置に最も近い[*1] k 個のデータ（kNN データ）を効率的に検索する手法を提案している．たとえば文献 [7] では，R-tree と B-tree を組み合わせた BR-tree というデータ構造を提案している．R-tree によりデータの集合およびデータ集合の MBR を保持し，B-tree によりあるキーワードを含むデータが含まれるノードの MBR のリストをキーワードごとに保持する．B-tree で管理する MBR のリストにより，クエリのキーワードを含むデータの中で，クエリに近いデータを効率的に発見できるため，高速に kNN データを検索できる．しかし，これらの研究は，ストリームデータを対象としていないため，本研究の問題に対しては効果的でない．

また，ストリームデータを対象とした位置およびキーワード検索に関する研究も存在する [21], [22], [23], [24]．文献 [21] では，位置，キーワードおよび時間の情報から，coverage および diversity のスコアを定義し，そのスコアの上位 k 個を検索する coverage and diversity aware top-k クエリ，文献 [22] では，ソーシャルネットワーク上での位置および時間を考慮した Top-k クエリ，文献 [23] では，マイクロブログ上での位置，キーワードおよび時間を考慮した Top-k クエリ，文献 [24] では，指定した時間および空間の範囲内のデータを検索する spatio-temporal range クエリを提案している．これらは，Pub/Sub 環境における kNN クエリを対象とした本研究とは問題が異なる．

[*1] 本稿では「クエリの指定位置に近い」という表現を単に「クエリに近い」と表記する．

2.2 Pub/Sub モデルにおけるデータモニタリング

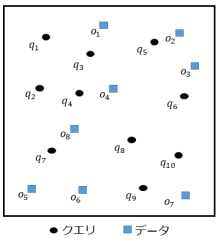
様々な研究が，位置およびキーワードを考慮した Pub/Sub モデルにおいて，各クエリの解をモニタリングする問題に取り組んでいる [1], [2], [3], [4], [5]．たとえば，文献 [1] では，R^t 木という新たなデータ構造によりクエリを管理することで，新たなデータが発生したとき，クエリの解を効率的にモニタリングする手法を提案している．R^t 木の葉ノードでは，クエリの集合，クエリ集合の MBR およびクエリの持つキーワードの集合を管理する．中継ノードでは子ノードへのポインタ，子ノードの持つ短形の MBR および子ノードの持つクエリのすべてのキーワードの集合を管理する．新たにデータが発生すると，ノードで管理する情報から，発生したデータが解となる可能性があるクエリを限定し，それらの解の更新を行う．しかし，これらの文献では，レンジクエリ [1], [3], [4] およびクエリの位置およびキーワードの一致度からデータのスコアを計算し，上位 k 個のデータを検索する Top-k クエリ [1], [2], [5] を対象としているため，kNN クエリを対象とした本研究とは問題が異なる．

2.3 クエリの分散処理

Pub/Sub モデルにおけるクエリの分散． Pub/Sub モデルにおいて，複数のワーカでクエリを分散して管理し，クエリの解をモニタリングする問題に取り組んでいる研究も存在する [10], [11], [12], [13], [14]．

文献 [11] では，位置情報およびキーワードに基づいてクエリを分散するアルゴリズムが提案されている．kd^t 木というデータ構造により，クエリを複数の集合に分割する．kd^t 木はデータを位置情報によって分割する kd 木にキーワードに基づく分割を加えたデータ構造である．ノードに含まれるクエリのキーワードの類似度を計算し，類似度が低ければ位置情報による分割，類似度が高ければキーワードによる分割を行う．各ノードに含まれるクエリを 1 つのワーカで管理することで，クエリを分散する．しかし，この研究はレンジクエリを対象としており，kNN クエリを対象とした本研究とは問題が異なる．

また，文献 [10], [12], [13], [14] では，Pub/Sub モデルにおいて，キーワードに基づいてクエリを分散するアルゴリズムが提案されている．たとえば，文献 [10] では，データのキーワードの出現頻度に基づいて，各ワーカが担当するキーワードを決定し，担当するキーワードを持つクエリを管理する．これらの研究では，データおよびクエリの位置



クエリ	キーワード
q_1	a,b,c
q_2	a,c,d
q_3	a,d,e
q_4	a,b,d
q_5	a,c
q_6	b,c
q_7	b,e
q_8	a
q_9	c,e
q_{10}	c

データ	キーワード
o_1	a,b
o_2	b,c
o_3	a,c
o_4	a,d
o_5	d,e
o_6	b,d
o_7	c
o_8	a,b

図 2　データおよびクエリの例
Fig. 2　Example of data objects and queries.

情報を考慮していない．

位置情報を用いたクエリの分散処理． 文献 [25], [26], [27], [28], [29] では，位置情報に基づいて複数のワーカにデータを分散するアルゴリズムが提案されている．たとえば，文献 [25] では，データが含まれる領域を与えられたワーカの数に分割し，1 つの領域に含まれるデータを 1 つのワーカで管理する．そして，ある kNN クエリの解を計算するとき，まず，クエリの位置を含む領域の中で kNN データを検索する．その後，クエリと k 番目のデータの距離の範囲が，他の領域に重複する場合，その領域に含まれるデータも含めて kNN データを検索する．しかし，これらの研究では，データおよびクエリのキーワードを考慮していない．また，MapReduce 環境を想定しており，ストリームデータに適用できない．

3. 問題定義

3.1 定義

生成されるストリームデータの集合を O，発行されるクエリの集合を Q とする．

データモデル． あるデータ $o \in O$ は位置（$o.loc$），およびキーワード集合（$o.key$）で構成される（$o = \langle loc, key \rangle$）．$o.loc$ は，緯度と経度によって表される 2 次元平面上の点である．

クエリモデル． ある kNN クエリ $q \in Q$ は，位置（$q.loc$），キーワード集合（$q.key$），および何個のデータを受信するかを指定する変数（$q.k$）で構成される（$q = \langle loc, key, k \rangle$）．$q.key$ は，key_{max} 個以下のキーワードを指定する．O が与えられたとき，q の kNN データ A は次の条件を満たす．(i) $|A| = k$，(ii) $\forall o \in A, \forall o' \in O_q - A, dist(o.loc, q.loc) \leq dist(o'.loc, q.loc)$．ここで，$O_q$ は O の中で $q.key$ に含まれるキーワードを 1 つ以上含むデータ集合，$dist(\cdot,\cdot)$ はデータとクエリとのユークリッド距離である．

例 2． 図 2 は，本稿を通して用いるデータおよびクエリの例を示す．ここで，$q_1.k = 3$ であると仮定すると，クエリ q_1 の kNN データは，$\{o_1, o_4, o_8\}$ である．

問題定義． O，Q およびワーカ[*2]集合 W が与えられたとき，複数のワーカ $w_i \in W$ に各クエリ $q \in Q$ を分散し，処理を分担することで，各クエリ q の kNN データをモニタリングする[*3]．

3.2 最適なクエリの分散

本稿では，複数のワーカでクエリを分散して管理することで，各クエリの kNN データを効率的にモニタリングする．新たなデータ o が生成されたとき，あるクエリ q に対して，q のキーワードに o のキーワードが含まれ，q の指定する位置に近い位置に o が生成された場合にのみクエリの kNN データを更新する．そのため，複数のワーカでクエリを分散して管理する場合，新たなデータが生成された際の各ワーカの処理時間は更新を行うクエリの数に依存する．各ワーカの処理時間を均一にするために，新たなデータが生成されたときの各ワーカの処理コストとしてワーカ w_i の負荷 $L(w_i)$ を定義する．

定義 1（ワーカの負荷）． あるワーカ w_i の負荷 $L(w_i)$ を以下で定義する．

$$L(w_i) = \sum_{q \in Q_{w_i}} L(q) \tag{1}$$

ここで，Q_{w_i} はワーカ w_i で管理されているクエリの集合である．$L(q)$ はクエリ q の負荷であり，あるデータが生成された際に，q の kNN データを更新する確率を表す．クエリ q が，あるワーカ w_i に割り振られたとき，$L(q)$ は，w_i に割り振られたクエリ集合および過去に発生したデータ集合の位置情報およびキーワードの分布に基づいて計算される．詳細な計算方法は後述する（3.3 節および 4.2 節）．

次に，各ワーカの負荷を均一にしつつ，全体の負荷を最小にするクエリの分散を決定するクエリ分散問題を定義する．

定義 2（クエリ分散問題）． O，Q およびワーカの数 m が与えられたとき，クエリ分散問題は，Q を以下の条件を満たす m 個の集合 Q_1，Q_2，$\cdots$，Q_m に分割する．(i) $\sum_{i=1}^{m} L(w_i)$ が最小．(ii) $\forall i \neq j$ に対して，$L(w_i) \geq L(w_j)$，$L(w_i)/L(w_j) \leq \sigma$．ここで，$\sigma \simeq 1$ である．

このとき，以下の定理が成り立つ．

定理 1． クエリ分散問題は NP 困難である．

証明． n 個の整数 $a_1, a_2, \cdots, a_n$ を 2 つの集合に分けるとき，各集合に含まれる整数の和が等しくなる集合に分ける問題を分割問題と呼び，この問題は NP 完全である．文献 [30] では，分割問題を拡張し，n 個の整数 $a_1, a_2, \cdots, a_n$ を，k 個の集合に分けるとき，各集合に含まれる整数の和の中で，最大のものと最小のものの差が最小となる集合に分ける問題に取り組んでおり，この問題は NP 困難である．クエリ分散問題は，文献 [30] で取り組まれている問題を解くことと同じである．よって，定理 1 が成り立つ．　□

*2　ワーカは 1 スレッド利用可能な CPU コアまたは計算機とする．

*3　データの送受信にかかる負荷は考慮しない．

3.3　ベースラインアルゴリズム

クエリ分散問題は NP 困難であるため，貪欲法を用いてクエリの分散を決定する．まず，ベースラインとなるアルゴリズムを紹介する．

3.3.1　キーワードに基づくクエリの分散

同じキーワードを持つクエリを 1 つのワーカで管理する方法のように，単純にクエリを分散すると，各ワーカの負荷に偏りが生まれる．そこで，データに頻出するキーワードを持つクエリから順にワーカに分散することで，各ワーカの負荷を均一にする．あるデータが発生したときに，クエリの kNN データが変化する確率（クエリの負荷）を計算する際，文献 [3] と同様に，クエリと過去に発生したデータ集合の位置情報およびキーワードの分布に基づいて計算する．各キーワードの出現確率が独立であると仮定すると，新たなデータ o が生成されたとき，$o.key$ にキーワード key が含まれる確率 $P_t(key)$ は以下の式で計算される．

$$P_t(key) = \frac{|O_{key}|}{|O|} \tag{2}$$

ここで，O_{key} は O の中で key を含むデータの集合である．このとき，あるクエリ q の kNN データとなりうるデータは，$q.key$ に含まれるキーワードを 1 つ以上含むデータであるため，$L(q)$ は $P_t(key)$ を用いて，以下のように計算される．

$$L(q) = \sum_{key \in q.key} P_t(key) \tag{3}$$

ここから，あるクエリを管理するワーカを決定するアルゴリズムを紹介する．まず，すべてのクエリ $q \in Q$ に対して $L(q)$ を計算する．そして，$L(q)$ の大きいクエリから順に，クエリを管理するワーカを決定する．あるクエリ q を管理するワーカは，その時点でワーカの負荷が最も小さいワーカ w_i とする．この操作を，すべてのクエリに対してそれらを管理するワーカを決定するまで繰り返すことで，クエリの分散を決定する．

新たなデータ o が生成されたとき，各ワーカで管理するクエリの kNN データを更新するアルゴリズムを紹介する．各ワーカ w は，自身の管理するクエリに対して，キーワードごとにクエリへのポインタを管理する転置ファイル $w.I$ を保持する．$w.I$ を用いて，o のキーワードを含むクエリの集合 Q_c を取得し，$q \in Q_c$ の kNN データを更新する．

例 3. 図 2 に示す 10 個のクエリを，キーワードに基づいて 2 つのワーカ w_1 および w_2 に分散することを考える．ここで，各クエリの負荷が**図 3** (a) のようになると仮定する．このとき，各ワーカで管理されるクエリは，それぞれ図 3 (b) および図 3 (c) のようになり，各ワーカの負荷は $L(w_1) = 0.58$ および $L(w_2) = 0.59$ となる．また，新たにデータ o_8 が発生したとき，各ワーカで kNN データの更新を行うクエリは w_1 では $\{q_1, q_4, q_5, q_8, q_{10}\}$，$w_2$ では $\{q_2, q_3, q_6, q_7\}$ である．

クエリ	負荷
q_1	0.2
q_2	0.18
q_3	0.16
q_4	0.14
q_5	0.13
q_6	0.1
q_7	0.09
q_8	0.08
q_9	0.06
q_{10}	0.03

(a) クエリの負荷

$w_1.Q$

q_1
q_4
q_5
q_8
q_{10}

$w_1.I$

キーワード	クエリ
a	q_1, q_4, q_5, q_8
b	q_1, q_{10}
c	q_1, q_5
d	q_4
e	q_4

(b) w_1 で管理するクエリ

$w_2.Q$

q_2
q_3
q_6
q_7
q_9

$w_2.I$

キーワード	クエリ
a	q_2, q_3
b	q_2, q_6, q_7
c	q_3, q_6, q_9
d	q_2, q_3
e	q_7, q_9

(c) w_2 で管理するクエリ

図 3　キーワードに基づく分散の例
Fig. 3　Example of query distribution with keywords.

3.3.2　位置情報に基づくクエリの分散

クエリが存在する領域をワーカの数に等分割し，各領域に含まれるクエリを 1 つのワーカで管理する方法のように，単純にクエリを分散すると，各ワーカの負荷に偏りが生まれる．そこで，各領域に含まれるクエリ集合の負荷を考え，負荷の大きい領域を小さい領域に分割する．そして，負荷が大きいクエリ集合から順にワーカに分散することで，各ワーカの負荷を均一にする．

まず，ある領域 r に含まれるクエリ集合を Q_r とし，Q_r の負荷 $L(Q_r)$ を考える．$L(Q_r)$ は，新たに生成されたデータにより Q_r 内に含まれるクエリの kNN データが変化する可能性がある確率と考えることができる．ここで，クエリ q の k 番目のデータを o_k とすると，新たに生成されたデータ o が q の kNN データとなるには，$dist(o.loc, q.loc) < dist(o_k.loc, q.loc)$ が必要である．つまり，$q.loc$ から $dist(o_k.loc, q.loc)$ の範囲を C_q とすると，C_q 内に生成されたデータが q の kNN データとなり得る．したがって，Q_r に含まれるすべてのクエリの C_q を包含する領域を R とすると，R 内で生成されるデータが，Q_r 内に含まれるクエリの kNN データとなりうる．たとえば，**図 4** において，ある領域 r 内に 4 つのクエリが存在している．このとき，それぞれのクエリの C_q を包含する領域が R である．キーワードに基づくクエリの分散と同様に，クエリと過去に発生したデータ集合の位置情報およびキーワードの分布に基づいて各ワーカの負荷を計算する．R 内に新たなデータが生成される確率 $P_s(R)$ は以下の式で計算される．

$$P_s(R) = \frac{|O_R|}{|O|} \tag{4}$$

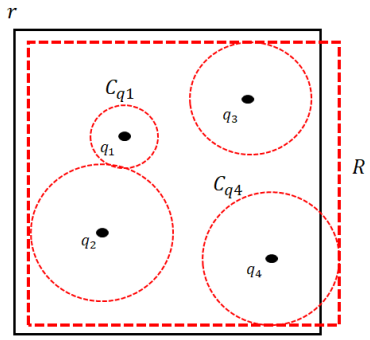


図 4　kNN データの更新が起こる領域 R
Fig. 4　Region R of update of kNN data.

ここで，O_R は R に含まれるデータの集合である．よって，$L(Q_r)$ は，$P_s(R)$ を用いて，以下のように計算される．

$$L(Q_r) = P_s(R) \times |Q_r| \tag{5}$$

さらに，各ワーカは，複数のクエリ集合を管理するため，$L(s_i)$ は，$L(Q_r)$ を用いて，以下のように計算される．

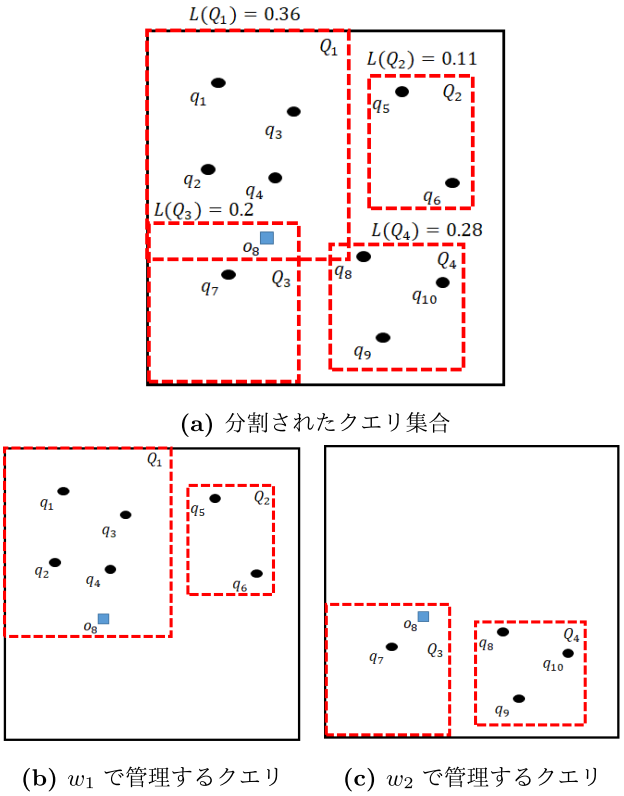
$$L(w_i) = \sum_{Q_i \in w_i.\mathbb{Q}} L(Q_r) \tag{6}$$

ここで，$w_i.\mathbb{Q}$ は，w_i が管理するクエリ集合の集合である．

ここから，クエリの分散を決定するアルゴリズムを紹介する．まず，クエリおよびデータが存在しうる領域を $\mathbb{R}^2$ とする．$\mathbb{R}^2$ を 4 つの領域に等分割し，各領域 r に含まれるクエリ集合 Q_r に対して，$L(Q_r)$ を計算する．そして，すべての領域の中で $L(Q_r)$ が最も大きい領域を再帰的に分割する．この処理を，分割された領域の数が，ワーカの数の α_s 倍になるまで行う．その後，$L(Q_r)$ の大きいクエリ集合から順に，その集合を管理するワーカを決定する．あるクエリ集合 Q_r を管理するワーカは，その時点で管理するクエリ集合の負荷の合計が最も小さいワーカ w_i とする．この操作を，すべてのクエリ集合を管理するワーカを決定するまで繰り返す．

新たなデータ o が生成されたとき，各ワーカで管理するクエリの kNN データを更新するアルゴリズムを紹介する．まず，ワーカ w が管理するクエリ集合 $Q_i \in w.\mathbb{Q}$ の中で，$o.loc \in Q_i.R$ である集合を $\mathbb{Q}_c$ とする．各ワーカ w は，各クエリ集合 $Q_i \in w.\mathbb{Q}$ に対して，キーワードごとにクエリ $q \in Q_i$ へのポインタを管理する転置ファイル $Q_i.I$ を保持している．よって，$Q_i \in \mathbb{Q}_c$ に対して，$Q_i.I$ を用いて，o のキーワードを含むクエリの集合 Q_c を取得し，$q \in Q_c$ の kNN データを更新する．

例 4. 図 2 に示す 10 個のクエリを，位置情報に基づいて 2 つのワーカ w_1 および w_2 に分散することを考える．領域の分割により，各クエリ集合が**図 5** (a) のように得られたと仮定する．このとき，各ワーカの管理するクエリはそれぞれ，図 5 (b) および図 5 (c) のようになり，各ワーカの負荷はそれぞれ $L(w_1) = 0.47$ および $L(w_2) = 0.48$ となる．また，新たにデータ o_8 が発生したとき，各ワーカで kNN データの更新を行うクエリは w_1 では $\{q_1, q_2, q_3, q_4\}$，w_2 では $\{q_7\}$ である．



(a) 分割されたクエリ集合

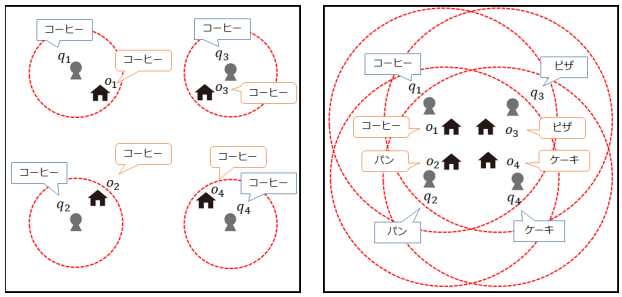
(b) w_1 で管理するクエリ　　(c) w_2 で管理するクエリ

図 5　位置情報に基づく分散の例
Fig. 5　Example of query distribution with space information.

4. 提案アルゴリズム

4.1 提案アルゴリズムの概要

ベースラインアルゴリズムでは，位置情報またはキーワードのいずれかに基づいて，クエリ集合を分散した．しかし，クエリおよびデータの位置およびキーワードの分布によって，より良い分散の方法は異なる．**図 6** に，クエリおよびデータの位置およびキーワードの分布の一例を示す．図 6 (a) の場合，各クエリの C_q は小さいため，位置情報に基づいてクエリを分散したとき，全体の負荷が小さくなる．一方，すべてのデータおよびクエリが同じキーワードを含んでいるため，キーワードに基づいてクエリを分散したとき，全体の負荷が大きくなる．また，図 6 (b) の場合，各クエリの C_q が大きく，C_q が重複しているため，位置情報に基づいてクエリを分散したとき，全体の負荷が大きくなる．一方，データおよびクエリのキーワードがそれぞれ異なるため，キーワードに基づいてクエリを分散したとき，全体の負荷が小さくなる．よって，図 6 (a) の場合，位置情報に基づいたクエリの分散，図 6 (b) の場合，キーワードに基づいたクエリの分散が適している．



(a) 位置に基づくクエリの分散が有効な分布　**(b)** キーワードに基づくクエリの分散が有効な分布

図 6　2 種類のデータおよびクエリの分布

Fig. 6　Two distribution of data objects and queries.

そこで，提案アルゴリズムでは，あるクエリの集合を分割するとき，クエリおよびデータの位置およびキーワードの分布に基づき，位置に基づく分割およびキーワードに基づく分割のうち，負荷の小さくなる分割を選択することで，全体の負荷がより小さくなるようにクエリを分割する．その後，各ワーカの負荷が均一になるように，分割された各クエリ集合に含まれるクエリを管理するワーカを決定する．

4.2　位置，キーワードに基づくワーカの負荷

はじめに，位置およびキーワードに基づく負荷を定義する．ベースラインアルゴリズムと同様に，クエリと過去に発生したデータ集合の位置情報およびキーワードの分布に基づいて各ワーカの負荷を計算する．各キーワードの出現確率が独立であると仮定すると，新たなデータ o が生成されたとき，$o.loc$ が R 内に含まれ，かつ $o.key$ にあるキーワード key が含まれる確率 $P(R, key)$ は，以下の式で計算される．

$$P(R, key) = \frac{|O_{R,key}|}{|O|} \tag{7}$$

ここで，$O_{R,key}$ は R 内に含まれ，かつ key を含むデータの集合である．あるクエリ集合 Q_i に含まれるクエリ q の負荷について考えると，Q_i に含まれるクエリ q の kNN データとなりうるデータは，R 内に含まれ，かつ $q.key$ に含まれるキーワードを 1 つ以上含むデータであるため，$L(q)$ は $P(R, key)$ を用いて，以下のように計算される．

$$L(q) = \sum_{key \in q.key} P(R, key) \tag{8}$$

また，Q_i の負荷 $L(Q_i)$ は，Q_i に含まれるクエリの負荷の合計と考えることができる．よって，$L(Q_i)$ は $L(q)$ を用いて，以下のように計算される．

$$L(Q_i) = \sum_{q \in Q_i} L(q) \tag{9}$$

さらに，各ワーカは複数のクエリ集合を管理するため，$L(w_i)$ は，$L(Q_i)$ を用いて，以下のように計算される．

$$L(w_i) = \sum_{Q_i \in w_i.\mathbb{Q}} L(Q_i) \tag{10}$$

4.3　クエリ分散のアルゴリズム

ここから，クエリの分散を決定するアルゴリズムを説明する．

4.3.1　クエリ集合の分割

あるクエリの集合 Q_i を複数の集合に分割するとき，位置情報に基づく分割およびキーワードに基づく分割のうち，より負荷の小さくなる方法で分割を行う．位置情報に基づく分割およびキーワードに基づく分割は，それぞれ以下の方法で行う．

位置情報に基づく分割． Q_i がある領域 r に含まれるとき，r を 4 つの領域 r_i（$1 \le i \le 4$）に等分割し，それぞれの領域に含まれるクエリの集合 Q_{r_i} に分割する．

キーワードに基づく分割． Q_i に含まれるクエリ q を $L(q)$ に基づいて，4 つの集合 Q_{t_i}（$1 \le i \le 4$）に分割する．具体的には，$L(q)$ が大きいクエリから順に，各集合に分配する．あるクエリを分配するとき，その時点で分配されたクエリの負荷の合計が最も小さい集合に，クエリを分配する．この処理を Q_i に含まれるすべてのクエリに対して行う．

位置情報に基づく分割およびキーワードに基づく分割のうち，負荷の小さくなる分割方法を決定するため，それぞれの分割により得られた集合の負荷の合計を比較する．具体的には，各集合 Q_i の負荷 $L(Q_i)$ を計算し，その後，位置情報に基づく分割によって得られた集合の負荷の合計を

$$C_s = \sum_{1 \le i \le 4} L(Q_{r_i}) \tag{11}$$

キーワードに基づく分割により得られた集合の負荷の合計を

$$C_t = \sum_{1 \le i \le 4} L(Q_{t_i}) \tag{12}$$

とする．C_s が C_t より小さければ，位置情報に基づく分割を選択し，C_t が C_s より小さければ，キーワードに基づく分割を選択する．

Algorithm 1 は，クエリの集合 Q_i を分割するアルゴリズムを示している．キーワードに基づく分割は，Q_i に含まれるクエリを 1 つずつチェックするため，Q_i に含まれるクエリの数が多いとき，分割に多大な時間がかかる．そのため，$|Q_i| > \beta$ である場合，つねに SpacePartition(Q_i) を実行する（2–3 行）．ここで，SpacePartition(Q_i) は，Q_i を位置情報に基づいて分割した集合の組を返す関数である．$|Q_i| \le \beta$ である場合，位置情報に基づく分割およびキーワードに基づく分割を行い，それぞれによって得られた集合の負荷の合計 C_s および C_t を計算する（5–7 行）．ここで，TextPartition(Q_i) は，Q_i をキーワードに基づいて分割した集合の組を返す関数である．$C_s < C_t$ ならば，位置情

Algorithm 1 Queryset-Partition

Input: Q_i, β
Output: $\mathbb{Q}$
1: **if** $|Q_i| > \beta$ **then**
2: $\quad \mathbb{Q}_s \leftarrow \mathsf{SpacePartition}(Q_i)$
3: $\quad \mathbb{Q} \leftarrow \mathbb{Q} \cup \mathbb{Q}_s$
4: **else**
5: $\quad \mathbb{Q}_s \leftarrow \mathsf{SpacePartition}(Q_i)$
6: $\quad \mathbb{Q}_t \leftarrow \mathsf{TextPartition}(Q_i)$
7: $\quad$ Calculate C_t and C_s
8: $\quad$ **if** $C_s < C_t$ **then**
9: $\quad\quad \mathbb{Q} \leftarrow \mathbb{Q} \cup \mathbb{Q}_s$
10: $\quad$ **else**
11: $\quad\quad \mathbb{Q} \leftarrow \mathbb{Q} \cup \mathbb{Q}_t$

Algorithm 2 Query-Assignment

Input: $\mathbb{Q}$
1: **while** $|\mathbb{Q}| > 0$ **do**
2: $\quad$ Pop Q_i from $\mathbb{Q}$
3: $\quad$ **while** $|Q_i| > 0$ **do**
4: $\quad\quad$ Pop q from Q_i
5: $\quad\quad w \leftarrow \arg\min_{w \in W} L(w)$
6: $\quad\quad w.Q \leftarrow w.Q \cup \{q\}$

報に基づく分割を選択し，$C_s \geq C_t$ ならば，キーワードに基づく分割を選択する（8–11 行）．分割されたすべての集合を $\mathbb{Q}$ とすると，$|\mathbb{Q}|$ が $\alpha_p m$ 以上になるまで，負荷の最も大きい集合 Q_i を再帰的に分割する．

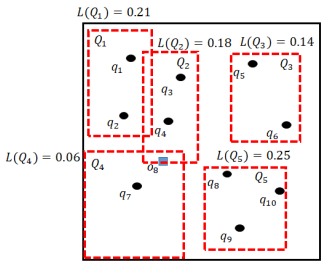
4.3.2　クエリを管理するワーカの決定

集合の分割後，各クエリ集合に含まれるクエリを管理するワーカを決定する．同じクエリ集合に含まれるクエリは，位置が近く，同じような出現頻度のキーワードを持つクエリが多いため，同じデータによってクエリの更新を行う可能性が高い．そのため，1 つのクエリ集合に含まれるクエリを 1 つのワーカで管理するのではなく，複数のワーカに分散させることで，各ワーカの負荷が均一になりやすい．

Algorithm 2 は各クエリ集合に含まれるクエリを管理するワーカを決定するアルゴリズムを示している．$\mathbb{Q}$ は分割されたすべてのクエリ集合であり，負荷の降順でソートされているとし，負荷の大きい集合 Q_i から順に，管理するワーカを決定する（2 行）．また，Q_i に含まれるクエリが負荷の降順にソートされているとする．あるクエリ q を管理するワーカは，その時点で負荷が最も負荷が小さいワーカ w とする（5–6 行）．Q_i に含まれるすべてのクエリを管理するワーカを決定するまで，この操作を行う．この処理を，すべてのクエリ集合に対して行う．

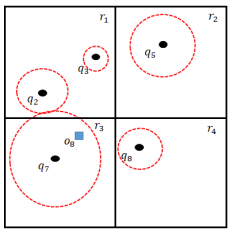
4.4　kNN データの更新アルゴリズム

新たなデータ o が生成されたとき，各ワーカで管理するクエリの kNN データを更新するアルゴリズムを紹介する．まず，クエリおよびデータが存在しうる領域 $\mathbb{R}^2$ を $n \times n$ の領域に等分割する．各ワーカは，それぞれの領域 r_i に対



クエリ	負荷
q_1	0.12
q_2	0.09
q_3	0.1
q_4	0.08
q_5	0.08
q_6	0.06
q_7	0.06
q_8	0.1
q_9	0.08
q_{10}	0.07

(a) 分割されたクエリ集合とクエリの負荷



$r_1.I$

キーワード	クエリ
a	q_2, q_3
b	q_2, q_7
c	q_3
d	q_2, q_3
e	q_7

$r_2.I$

キーワード	クエリ
a	q_5
c	q_5

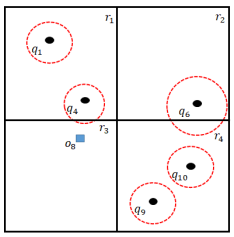
$r_3.I$

キーワード	クエリ
b	q_7
e	q_7

$r_4.I$

キーワード	クエリ
a	q_8

(b) w_1 で管理するクエリ



$r_1.I$

キーワード	クエリ
a	q_1, q_4
b	q_1
c	q_1
d	q_4
e	q_4

$r_2.I$

キーワード	クエリ
b	q_6
c	q_6

$r_3.I$

キーワード	クエリ
a	q_4
d	q_4
e	q_4

$r_4.I$

キーワード	クエリ
b	q_6, q_{10}
c	q_6, q_9
e	q_9

(c) w_2 で管理するクエリ

図 7　位置情報およびキーワードに基づく分散の例
Fig. 7　Example of query distribution with keywords and space information.

して，ワーカで管理するクエリ $q \in s.Q$ の C_q が r_i と重複するクエリへのポインタをキーワードごとに管理する転置ファイル $r_i.I$ を保持する．新たなデータ o が生成されたとき，$o.loc \in r_i$ となる領域の $r_i.I$ を用いて，o のキーワードを含むクエリの集合 Q_c を取得し，$q \in Q_c$ の kNN データを更新する．

例 5.　図 2 に示す 10 個のクエリを位置情報およびキーワードに基づいて 2 つのワーカ w_1 および w_2 に分散することを考える．クエリ集合の分割により，各クエリ集合とクエリの負荷が**図 7** (a) のように得られたと仮定する．このとき，各ワーカの管理するクエリはそれぞれ図 7 (b) および図 7 (c) となり，各ワーカの負荷はそれぞれ $L(w_1) = 0.43$ および $L(w_2) = 0.41$ となる．また，新たにデータ o_8 が発生したとき，$\mathbb{R}^2$ を 2×2 の 4 つの領域に分割したとすると，各ワーカで kNN データの更新を行うクエリは w_1 では $\{q_7\}$，w_2 では $\{q_4\}$ である．

5. 性能評価

本章では，提案アルゴリズムの性能評価のために行った実験の結果を示す．提案アルゴリズムの性能を評価するため，以下のアルゴリズムとの比較を行った．

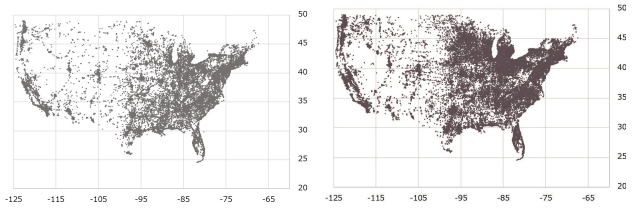
- ベースライン 1：キーワードのみに基づいてクエリの分散を行う．
- ベースライン 2：位置情報のみに基づいてクエリの分散を行う．

5.1 実験環境

本実験は，7 台の計算機で構成されるクラスタを用いた．1 台は Windows 10 Pro，3.00 GHz Intel Xeon Gold 6154，および 512 GB RAM を搭載した計算機であり，その他 6 台は Windows 10 Pro，2.40 GHz Intel Core i7-8700T，および 32 GB RAM を搭載した計算機である．データの送信は前者の計算機で行い，kNN モニタリングは後者の計算機で行った．また，CPU の 1 コアを 1 つのワーカとした．すべてのアルゴリズムは C++で実装した．

5.1.1 データセット

本実験では，生成されるデータとして Twitter [*4] [31] および Place [*5] [32] を用いた．Twitter はアメリカの位置情報付きツイートデータ，Place はアメリカの公共スペースについて記述したキーワードと位置情報を含むデータであり，緯度が北緯 20 度から北緯 50 度，経度が西経 60 度から西経 125 度の範囲に存在するデータを使用した．**図 8** に各データセットの分布，**表 1** にデータセットの詳細を示す．各クエリのキーワードは，1 つのデータに現れる単語の中から，5 個以下の単語をランダムに選んだものとした．



(a) Twitter　　**(b)** Place

図 8　データの分布
Fig. 8　Data distribution.

表 1　データセットの詳細
Table 1　Datasets statistics.

	Twitter	Place
データ数	20,000,000	9,356,750
キーワード数	2,225,654	53,931
1 つのデータの平均キーワード数	5.51	2.94

*4 http://www.ntu.edu.sg/home/gaocong/datacode.htm
*5 https://archive.org/details/2011-08-SimpleGeo-CC0-Public-Spaces

5.1.2 パラメータ

表 2 に，本実験で用いたパラメータを示す．太字で表されている値はデフォルトの値である．本実験では，あるパラメータを変化させる際，その他のパラメータはデフォルトの値を用いた．k は 1 から k_{max} の間の一様乱数とし，$\alpha_s = 20$，$\alpha_p = 20$ および $\beta = 100{,}000$ とした．これらの値は，α および β は，事前実験により最も結果が良いパラメータを選択した．実験は，文献 [5] と同様に，投稿時間が古いものからデータを発生させ，データが 1,000,000 個に到達した時点でクエリを分散する．そして，新たにデータが 1,000,000 個発生するまで行う．この際，1 度に 1,000 個のデータが発生するとし，1,000 回データの更新を行った．

5.2 評価結果

本稿では，頻繁に発生するデータに対して，各ユーザが自身にとって有用な k 個のデータ（kNN データ）をモニタリングしていると仮定し，全クエリの kNN データの更新が完了するまでの時間を評価するため，各アルゴリズムで，

- 平均更新時間（1 度に発生するすべてのデータに対して，kNN データの更新が完了するまでの平均時間 [msec]）
- ワーカの更新時間のばらつき（各ワーカで管理しているクエリの kNN データの更新が完了するまでの時間の中で最大の時間と最小の時間の差 [msec]）

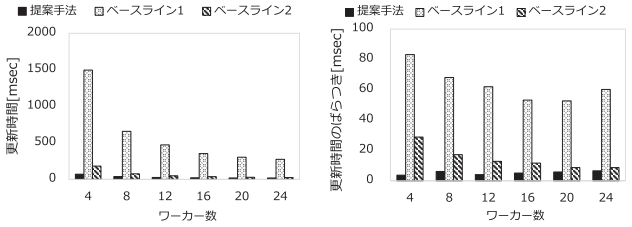
の 2 つの指標を評価した．以下に結果を示す．

5.2.1 ワーカ数の影響

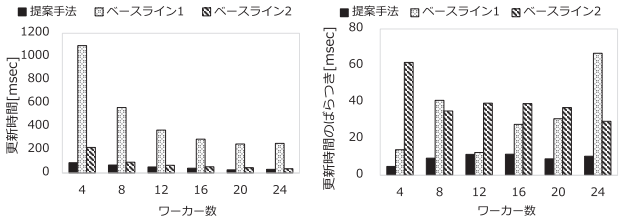
図 9 にワーカ数を変えたときの結果を示す．ワーカ数が増加すると，各ワーカで管理するクエリの数が少なくなるため，各アルゴリズムにおいて，平均更新時間は短くなる．また，分散するワーカの数がどのような場合であっても，提案手法はつねにベースライン手法よりも高速に kNN データの更新ができる．ベースライン 1 では，発生したデータのキーワードを含むすべてのクエリの kNN データの更新を行うため，平均更新時間は長くなる．ベースライン 2 では，クエリ集合の R と転置ファイルに基づいて，kNN データの更新を行うクエリを決定する．これにより，ベースライン 1 に比べて，更新を行うクエリの数が小さくなるため，平均更新時間は短くなる．提案アルゴリズムでは，位置とキーワードに基づく負荷を考慮することにより，ベースライン 2 よりも，より正確にクエリ集合に含まれるクエリの kNN データの更新にかかる時間を見積もることができる．

表 2　パラメータの設定
Table 2　Configuration of parameters.

パラメータ	値
ワーカ数	4, **8**, 12, 16, 20, 24
$q.k$ の最大値，k_{max}	5, **10**, 15, 20
クエリの数，$\|Q\|[\times 10^6]$	**1**, 1.5, 2, 2.5, 3

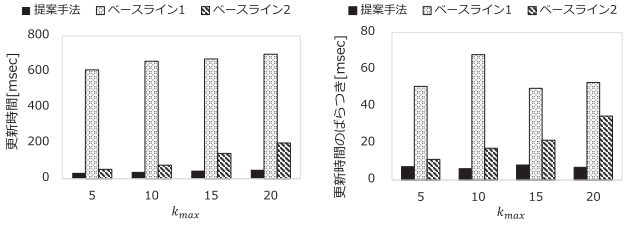


(a) 平均更新時間 (Twitter)　(b) 更新時間のばらつき (Twitter)

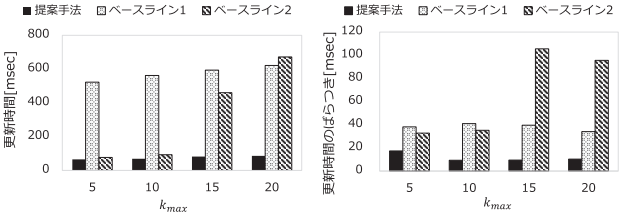


(c) 平均更新時間 (Place)　(d) 更新時間のばらつき (Place)

図 9　ワーカ数の影響
Fig. 9　Effect of the number of workers.



(a) 平均更新時間 (Twitter)　(b) 更新時間のばらつき (Twitter)



(c) 平均更新時間 (Place)　(d) 更新時間のばらつき (Place)

図 10　k_{max} の影響
Fig. 10　Effect of k_{max}.

また，位置に基づく分割およびキーワードに基づく分割のうち，より全体の負荷が小さくなるようにクエリ集合を分割することで，クエリの更新にかかる時間を短くする．さらに，分割の際に同じ集合に含まれたクエリは，位置が近く，同じような出現頻度のキーワードを持つクエリが多いため，同じデータによってクエリの更新を行う可能性が高い．そのため，1 つのクエリ集合に含まれるクエリを 1 つのワーカで管理するのではなく，複数のワーカに分散させることで，各ワーカの更新時間のばらつきが小さくなり，平均更新時間も短くなる．

各アルゴリズムにおいて，ワーカの数が増加すると，各ワーカで更新を行うクエリの数が小さくなるため，更新時間のばらつきは小さくなる．提案アルゴリズムは，各ベースラインに比べて，更新を行うクエリの数がより小さくなり，各ワーカで更新を行うクエリの数の差が小さくなるため，更新時間のばらつきがより小さくなる．

5.2.2　k_{max} の影響

図 10 に k_{max} を変えたときの結果を示す．k_{max} が増加すると，kNN データが変化するクエリの数が増加するため，各アルゴリズムにおいて，平均更新時間が長くなる．提案アルゴリズムは，k_{max} によらず，つねに平均更新時間が最も短い．ベースライン 2 は，k_{max} が増加すると，クエリ集合の R の範囲が大きくなり，新たに発生したデータが kNN データとなる可能性があるクエリの数が大きくなるため，平均更新時間が長くなりやすい．

また，提案アルゴリズムは，k_{max} によらず，つねに更新時間のばらつきが小さい．ベースライン 1 では，k_{max} によってクエリの負荷が変化することはなく，各ワーカで管理されるクエリは変化しない．したがって，各ワーカで更新を行うクエリの数は変化しないため，更新時間のばらつきがほぼ一定である．ベースライン 2 では，k_{max} が増加すると，各クエリ集合の R の範囲が大きくなり，各ワーカで管理されるクエリ集合の R の範囲が重複する可能性が大きくなる．したがって，あるワーカの R の範囲が重複する位置にデータが発生した場合，そのワーカで更新を行うクエリの数が大きくなるため，更新時間のばらつきが大きくなる．

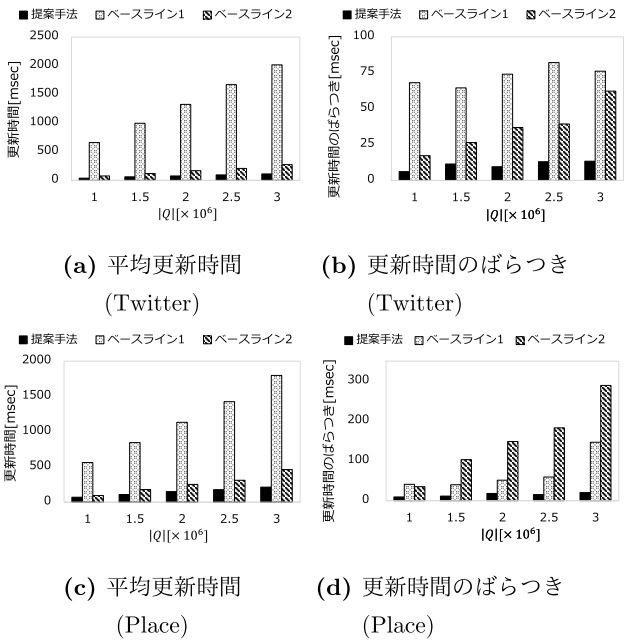
5.2.3　$|Q|$ の影響

図 11 に $|Q|$ を変えたときの結果を示す．提案アルゴリズムは，$|Q|$ によらず，つねに平均更新時間が最も短い．また，$|Q|$ が増加すると，1 度のデータの発生に対して，更新を行うクエリの数が増加するため，各アルゴリズムにおいて，平均更新時間は長くなるが，提案アルゴリズムは，各ベースラインと比べて，1 度のデータの発生に対して，更新を行うクエリの数がより小さいため，クエリの数が大きいほど，提案アルゴリズムと各ベースラインの平均更新時間の差が大きくなる．

また，$|Q|$ が増加すると，各アルゴリズムにおいて，更新時間のばらつきは大きくなる．各アルゴリズムにおいて，$|Q|$ が増加すると，各ワーカでクエリの更新を行う回数が大きくなり，各ワーカでクエリの更新を行う回数の差が大きくなるため，更新時間のばらつきは大きくなる．

5.2.4　負荷の有効性

過去に発生したデータ集合の分布に基づいて負荷を計算することの有効性を検証するための実験を Twitter を用いて行った．**図 12** は，クエリを分散後，100,000 個のデータが新たに発生した時の平均更新時間を表し，5,000,000 個



(a) 平均更新時間 (Twitter)　(b) 更新時間のばらつき (Twitter)

(c) 平均更新時間 (Place)　(d) 更新時間のばらつき (Place)

図 11　$|Q|$ の影響
Fig. 11　Effect of $|Q|$.

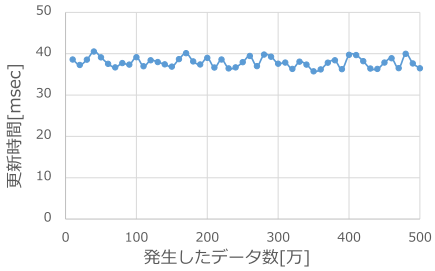


図 12　平均更新時間の変化
Fig. 12　Change of average update time.

のデータが新たに発生するまで実験を行った．新たに発生するデータに対して，更新時間がほぼ一定であるため，過去に発生したデータ集合に基づいた負荷の計算をすることは有効である．

6. まとめ

近年，パブリッシュ/サブスクライブ（Pub/Sub）モデルに基づくアプリケーションが多く存在する．また，スマートフォンや SNS の普及により，データやクエリの数が増加しており，単一のワーカですべての処理を行うとリアルタイム性を保持することが困難になっている．本稿では，Pub/Sub モデルにおいて，複数のワーカでクエリを分散管理して処理を分担することにより，各クエリの kNN データを効率的にモニタリングする問題に取り組んだ．データおよびクエリの位置情報およびキーワードに基づいてワーカの負荷を定義し，クエリ集合を分割する際，位置情報に基づく分割およびキーワードに基づく分割のうち，全体の負荷が小さくなる分割を選択することで，各ワーカの負荷を均一にしつつ，全体の負荷を小さくするようにクエリを分散するアルゴリズムを提案した．実データを用いた実験により，提案アルゴリズムの有効性を確認した．

実データを用いた実験の結果から，提案アルゴリズムは，位置情報に基づく分割およびキーワードに基づく分割のうち，分割後の負荷の合計がより少なくなる方法で分割を行うことにより，位置情報に基づく分割のみを行う手法およびキーワードに基づく分割のみを行う手法に比べて，すべてのクエリの更新にかかる時間が小さくなり，かつ各ワーカの更新時間のばらつきも小さくなることを確認した．

今後の課題． 本稿では，事前に発生したデータとクエリの位置情報およびキーワードに基づいてクエリを分散した．しかし，データの生成によって，クエリの kNN データが更新されると，クエリの C_q の大きさが変化し，クエリの負荷が変化する．その結果，ワーカの負荷が変化し，各ワーカの負荷に偏りが生まれる可能性がある．また，クエリの追加および削除によって，ワーカの負荷が変化し，各ワーカの負荷に偏りが生まれる可能性がある．そのため，データの生成，クエリの追加および削除によって，各ワーカの負荷が均一でなくなったとき，ワーカが管理するクエリを動的に変更し，各ワーカの負荷を均一に保つアルゴリズムを設計することを検討している．

謝辞　本研究の一部は，文部科学省科学研究費補助金・基盤研究（A）（JP18H04095），および基盤研究（B）（T17KT0082a）の研究助成によるものである．ここに記して謝意を表す．

参考文献

[1] Li, G., Wang, Y., Wang, T. and Feng, J.: Location-aware publish/subscribe, *Proc. ACM SIGMOD Int'l Conf. Management of Data*, pp.802–810 (2013).
[2] Hu, H., Liu, Y., Li, G., Feng, J. and Tan, K.L.: A location-aware publish/subscribe framework for parameterized spatio-textual subscriptions, *Proc. IEEE Int'l Conf. Data Engineering*, pp.711–722 (2015).
[3] Wang, X., Zhang, Y., Zhang, W., Lin, X. and Wang, W.: Ap-tree: Efficiently support continuous spatial-keyword queries over stream, *Proc. IEEE Int'l Conf. Data Engineering*, pp.1107–1118 (2015).
[4] Yu, M., Li, G. and Feng, J.: A cost-based method for location-aware publish/subscribe services, *Proc. ACM Conf. Information and Knowledge Management*, pp.693–702 (2015).
[5] Wang, X., Zhang, Y., Zhang, W., Lin, X. and Huang, Z.: Skype: Top-k spatial-keyword publish/subscribe over sliding window, *Proc. VLDB Endowment*, Vol.9, No.7, pp.588–599 (2016).
[6] Choi, D.W., Pei, J. and Lin, X.: Finding the minimum spatial keyword cover, *Proc. IEEE Int'l Conf. Data Engineering*, pp.685–696 (2016).
[7] Li, G., Xu, J. and Feng, J.: Keyword-based k-nearest neighbor search in spatial databases, *Proc. ACM Int'l Conf. Information and Knowledge Management*, pp.2144–2148 (2012).
[8] Lu, J., Lu, Y. and Cong, G.: Reverse spatial and textual k nearest neighbor search, *Proc. ACM SIGMOD Int'l*

Conf. Management of Data, pp.349–360 (2011).
[9] Tao, Y. and Sheng, C.: Fast nearest neighbor search with keywords, *IEEE Trans. Knowledge and Data Engineering*, Vol.26, No.4, pp.878–888 (2014).
[10] Basık, F., Gedik, B., Ferhatosmanoğlu, H. and Kalender, M.E.: s^3-tm: Scalable streaming short text matching, *The VLDB Journal*, Vol.24, No.6, pp.849–866 (2015).
[11] Chen, Z., Cong, G., Zhang, Z., Fuz, T.Z. and Chen, L.: Distributed publish/subscribe query processing on the spatio-textual data stream, *Proc. IEEE Int'l Conf. Data Engineering*, pp.1095–1106 (2017).
[12] Yu, A., Agarwal, P.K. and Yang, J.: Subscriber assignment for wide-area content-based publish/subscribe, *IEEE Trans. Knowledge and Data Engineering*, Vol.24, No.10, pp.1833–1847 (2012).
[13] Barazzutti, R., Heinze, T., Martin, A., Onica, E., Felber, P., Fetzer, C., Jerzak, Z., Pasin, M. and Rivière, E.: Elastic scaling of a high-throughput content-based publish/subscribe engine, *Proc. IEEE Int'l Conf. Distributed Computing Systems*, pp.567–576 (2014).
[14] Baldoni, R., Marchetti, C., Virgillito, A. and Vitenberg, R.: Content-based publish-subscribe over structured overlay networks, *Proc. IEEE Int'l Conf. Distributed Computing Systems*, pp.437–446 (2005).
[15] Cong, G., Jensen, C.S. and Wu, D.: Efficient retrieval of the top-k most relevant spatial web objects, *Proc. VLDB Endowment*, Vol.2, No.1, pp.337–348 (2009).
[16] Zheng, K., Su, H., Zheng, B., Shang, S., Xu, J., Liu, J. and Zhou, X.: Interactive top-k spatial keyword queries, *Proc. IEEE Int'l Conf. Data Engineering*, pp.423–434 (2015).
[17] Zhang, C., Zhang, Y., Zhang, W. and Lin, X.: Inverted linear quadtree: Efficient top k spatial keyword search, *Proc. IEEE Int'l Conf. Data Engineering*, pp.901–912 (2013).
[18] Wu, D., Yiu, M.L., Cong, G. and Jensen, C.S.: Joint top-k spatial keyword query processing, *IEEE Trans. Knowledge and Data Engineering*, Vol.24, No.10, pp.1889–1903 (2012).
[19] Tsatsanifos, G. and Vlachou, A.: On processing top-k spatio-textual preference queries, *Proc. Int'l Conf. Extending Database Technology*, pp.433–444 (2015).
[20] Zhang, D., Tan, K.L. and Tung, A.K.: Scalable top-k spatial keyword search, *Proc. Int'l Conf. Extending Database Technology*, pp.359–370 (2013).
[21] Mehta, P., Skoutas, D., Sacharidis, D. and Voisard, A.: Coverage and diversity aware top-k query for spatio-temporal posts, *Proc. ACM SIGSPATIAL Int'l Conf. Advances in Geographic Information Systems*, p.37 (2016).
[22] Cozza, V., Messina, A., Montesi, D., Arietta, L. and Magnani, M.: Spatio-temporal keyword queries in social networks, *East European Conf. Advances in Databases and Information Systems*, pp.70–83 (2013).
[23] Magdy, A., Mokbel, M.F., Elnikety, S., Nath, S. and He, Y.: Mercury: A memory-constrained spatio-temporal real-time search on microblogs, *Proc. IEEE Int'l Conf. Data Engineering*, pp.172–183 (2014).
[24] Nepomnyachiy, S., Gelley, B., Jiang, W. and Minkus, T.: What, where, and when: Keyword search with spatio-temporal ranges, *Proc. Workshop on Geographic Information Retrieval*, p.2 (2014).
[25] Eldawy, A. and Mokbel, M.F.: Spatialhadoop: A mapreduce framework for spatial data, *Proc. IEEE Int'l Conf. Data Engineering*, pp.1352–1363 (2015).
[26] Aly, A.M., Mahmood, A.R., Hassan, M.S., Aref, W.G., Ouzzani, M., Elmeleegy, H. and Qadah, T.: Aqwa: Adaptive query workload aware partitioning of big spatial data, *Proc. VLDB Endowment*, Vol.8, No.13, pp.2062–2073 (2015).
[27] Aji, A., Wang, F., Vo, H., Lee, R., Liu, Q., Zhang, X. and Saltz, J.: Hadoop gis: A high performance spatial data warehousing system over mapreduce, *Proc. VLDB Endowment*, Vol.6, No.11, pp.1009–1020 (2013).
[28] Nishimura, S., Das, S., Agrawal, D. and El Abbadi, A.: Md-hbase: A scalable multi-dimensional data infrastructure for location aware services, *Proc. IEEE Int'l Conf. Mobile Data Management*, Vol.1, pp.7–16 (2011).
[29] Akdogan, A., Demiryurek, U., Banaei-Kashani, F. and Shahabi, C.: Voronoi-based geospatial query processing with mapreduce, *Proc. IEEE Int'l Conf. Cloud Computing Technology and Science*, pp.9–16 (2010).
[30] Korf, R.E.: Multi-way number partitioning, *Int'l Joint Conf. Artificial Intelligence*, pp.538–543 (2009).
[31] Chen, L., Cong, G., Cao, X. and Tan, K.L.: Temporal spatial-keyword top-k publish/subscribe, *Proc. IEEE Int'l Conf. Data Engineering*, pp.255–266 (2015).
[32] Mahmood, A.R., Aly, A.M. and Aref, W.G.: Fast: Frequency-aware indexing for spatio-textual data streams, *Proc. IEEE Int'l Conf. Data Engineering*, pp.305–316 (2018).

鶴岡 翔平　（学生会員）

2019 年大阪大学工学部電子情報工学科卒業後，同大学大学院情報科学研究科博士前期課程在学中．データベース，ネットワーク環境におけるデータ検索技術に関する研究に従事．

西尾 俊哉　（学生会員）

2016 年大阪大学工学部電子情報工学科卒業．2018 年同大学大学院情報科学研究科博士前期課程修了後，同大学院情報科学研究科博士後期課程在学中．データベース，ネットワーク環境におけるデータ検索技術に関する研究に従事．

天方 大地　（正会員）

2012 年大阪大学工学部電子情報工学科卒業．2014 年同大学大学院情報科学研究科博士前期課程修了．2015 年同大学院情報科学研究科博士後期課程修了後，同年同大学院情報科学研究科マルチメディア工学専攻助教となり，現在に至る．情報科学博士．データベース，ネットワーク環境におけるデータ検索技術に関する研究に従事．IEEE，ACM，日本データベース学会の各会員．

原 隆浩　（正会員）

1995 年大阪大学工学部情報システム工学科卒業．1997 年同大学大学院工学研究科博士前期課程修了．同年同大学院工学研究科博士後期課程中退後，同大学院工学研究科助手，2002 年同大学院情報科学研究科助手，2004 年同大学院情報科学研究科准教授．2015 年より同大学院情報科学研究科教授となり，現在に至る．工学博士．2000 年電気通信普及財団テレコムシステム技術賞受賞．2003 年本学会研究開発奨励賞受賞．2008 年，2009 年本学会論文賞，2015 年日本学術振興会賞，2017 年大阪科学賞受賞．モバイルコンピューティング，ネットワーク環境におけるデータ管理技術に関する研究に従事．

（担当編集委員　井上 潮）