区間をキーとして保持する分散KVSの効率的な実現法

An Efficient Method for Implementing Distributed KVS that Supports an Interval as a Key

岩崎 章彦[†] 安倍 広多[†] 石橋 勇人[†] 松浦 敏雄[†]
Akihiko Iwasaki Kota Abe Hayato Ishibashi Toshio Matsuura

1 はじめに

分散 KVS (Key-Value Store) とは，ネットワークで接続された複数の計算機を用いて，大量のデータを格納するシステムである．一般に，ノード数を増加させることで性能が向上する（スケールアウトする）ように設計されており，Google, Amazon, Facebook のような膨大なデータを扱う大規模サービスで利用されている．

分散 KVS では，基本的にデータ（レコード）はキー (key) と値 (value) のペアで構成される．データを登録する際はキーと値の両方を指定する（例えばキー 10 に対して値 A を，キー 20 に対して値 B を登録するなど）．キーを指定することで対応する値を取得できる．（キーの範囲を指定して値を取得できるシステムもある）．

上の例のように，分散 KVS において一般にキーは単一の値であるが，取り扱うデータによっては，レコードを単一のキーで表すことが不適切な場合がある．例えば以下のような場合である．

- データが時間的範囲を持つ場合（ユーザが撮影した動画のアーカイブ，ユーザのセッションログなど）
- データが空間的範囲を持つ場合（センサーネットワークで各ノードがカバーする地理的領域，GPS による移動記録など）

例えばデータが時間的範囲を持つ場合，指定した時刻の範囲（T_1 から T_2 まで）に含まれるすべてのデータを検索する処理（範囲検索）が重要であるが，キーが単一の値で表されていると，この処理を効率よく実現することが難しい．

この問題に関連する研究として，Range key skip graph[1]（以下 RKSG）がある．RKSG は Skip graph[2] をベースとした分散データ構造であり，上限，下限の 2 つの値で表される区間をキー（レンジキーと呼ばれる）として持つ．RKSG では，指定した範囲に区間が含まれるキーの検索（範囲検索）が可能である[*1]．しかし，RKSG では，各レコードが区間に重なりがあるすべての他のレコードへポインタ（包含キーと呼ばれる）を維持する必要があり，特に区間が重なるキーが多数存在する場合のコストが大きいなどの問題がある（2.2.3 節で詳しく述べる）．

本稿では，RKSG の問題点を解決し，効率良く区間をキーとして扱える分散 KVS の実現法を提案する．提案手法では，下限と上限で表されるキーの区間を 2 次元平面上の 1 点にマッピングすることで，区間に対する範囲検索処理を 2 次元平面上での範囲検索処理に帰着する．2 次元平面上のデータを分散システムで扱うためには，既存の方法 (Znet) を用いる．

2 関連研究

ここでは関連研究について述べる．まず，2.1 節で RKSG がベースとしている Skip graph について概説し，2.2 節で RKSG について説明する．2.3 節では，提案手法が利用する Znet について述べる．

2.1 Skip graph

Skip graph[2] は，構造化オーバーレイネットワークの一種である．例を図 1 に示す．Skip graph の各ノードはキーを保持する．また，各ノードには membership vector と呼ばれる基数 w の整数値が乱数によって割り当てられる（本稿では $w = 2$ とする）．Skip graph には複数のレベルがあり，レベル l では membership vector のプレフィックスが l 桁一致するノード同士で双方向リンクを張ることで，キーの順にソートされた双方向連結リストを構成する．なお，レベル 0 ではすべてのノードが唯一の連結リストで接続されている．

各ノードで連結リスト上に存在するノードが自分自身のみとなるレベルを maxlevel と呼ぶ．n ノードから構成される Skip graph では，maxlevel は平均 $O(\log n)$ である．また，ノードの検索，挿入に要するメッセージ数，ホップ数，各ノードの経路表の領域計算量はすべて $O(\log n)$ である．

Skip graph ではキーがソートされているため，範囲検索（指定した範囲内のノードをすべて検索する）を容易に実現できる．

2.2 Range key skip graph

1 章で述べたように，RKSG[1] は Skip graph をベースとした分散データ構造で，上限，下限の 2 つの値で表される区間をキー（レンジキー）として保持し，区間に対する範囲検索が可能である．

RKSG では，レンジキーの下限値を Skip graph のキーとして登録する．また，範囲検索を効率的に実行するために，各レンジキーは包含キーと呼ばれる補助的なポインタを保持する．

包含キーの持ち方としては 2 通り提案されている．

- レンジキー自身の最小値に重なる他のレンジキーへのポインタをすべて持つ方法（手法 1）
- レンジキー自身の最小値に重なる他のレンジキーへのポ

[†] 大阪市立大学，Osaka City University

[*1] 本稿ではキーに付随する範囲を「区間」，検索のために指定する範囲を「範囲」と使い分ける．

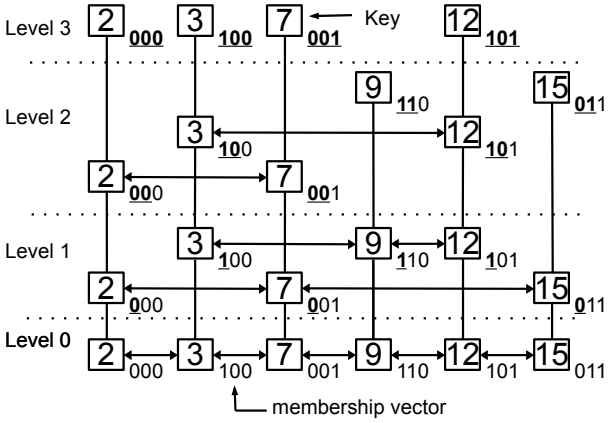

図 1　Skip graph の構造

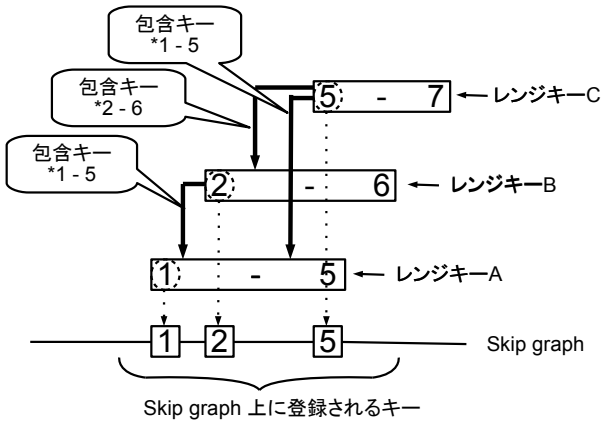

図 2　Range key skip graph におけるレンジキーと包含キーの関係（手法 1 の場合）

インタを必要最低限だけ持つ方法（手法 2）

しかし，文献 [1] では，手法 1 の範囲検索アルゴリズムしか述べられていないため，本稿では以後手法 1 のみを議論する．手法 1 の例を図 2 に示す．

2.2.1　範囲検索アルゴリズム

RKSG の範囲検索アルゴリズムを述べるにあたって，範囲の表記法を定める．以下，本稿では，1 次元空間における u 以上 v 以下の範囲を範囲 $[u, v]$ と表記する（ただし，$u \leq v$）．区間の表記についても同様である．

指定された範囲 QR=$[Qs, Qe]$ 内に下限値がある区間は，Skip graph の範囲検索を用いて発見できる（例えば図 2 で，範囲 $[3, 6]$ を検索すると，キー 5 からレンジキー C を発見できる）．

範囲内に下限値が存在しない区間を検索するためには，まず Skip graph 上で Qs を超えない最大の最小値を持つレンジキー RK を検索する（上の例では RK はレンジキー B）．さらに RK のもつ包含キーで，QR と重なりがある各ノード（上の例ではレンジキー A）に対して範囲検索クエリを転送する．これによって，すべての区間を検索できる．

2.2.2　レンジキーの追加・削除の方法

レンジキー RK を追加する場合，まず上述の範囲検索アルゴリズムを用いて RK の下限値を包含するレンジキー群を検索する．このレンジキー群が RK の包含キーとなる．次に，RK の下限値を Skip graph に登録する．

レンジキー RK を削除する場合，まず範囲検索アルゴリズムを用いて RK の区間を検索することで，包含キーとして RK を保持しているレンジキー群を発見する．これらのレンジキーから RK に対する包含キーを削除した後，Skip graph から RK の下限値を削除する．

2.2.3　問題点

分散 KVS に RKSG を適用した場合の問題点について考える．

RKSG では，1 区間が Skip graph の 1 ノードに対応するが，ノード数を n とするとき，Skip graph の 1 ノードの領域計算量は $O(\log n)$ であり，多数の区間を管理するにはオーバヘッドが大きい．また，RKSG では複製を配置するなどのデータの冗長性が考慮されていないため，分散 KVS に適用するためにはなんらかの拡張が必要である（これらの問題点は，RKSG はセンサーネットワークなどでの応用を念頭においており，分散 KVS での適用は想定していないことに起因する）．

また，あるレンジキー RK が保持する包含キーの数は，RK の下限値と重なるレンジキーの数と等しい．重なるレンジキーがそれほどなければ問題ないが（例えばレンジキーが一様に分布し，レンジキーの大きさも限られている場合），最悪の場合は，他のすべてのレンジキーへのポインタを保持することになる（レンジキーの数 n に対して $O(n)$）．このとき，各ノードの領域計算量や挿入・削除のコストも $O(n)$ となり，スケーラビリティ上問題がある．

本稿では，これらの問題点の解決を試みる．

2.3　Znet

Znet[3] は，多次元データを効率的に分散管理する構造化オーバーレイネットワークで，多次元データの登録と範囲検索を効率よく実行できる．空間充填曲線である Z 曲線と Skip graph を用いている．Znet は n 次元のデータを扱えるが，以下では 2 次元の場合で説明する．

扱う 2 次元空間を $d \times d$ の平面とする．Znet に参加しているそれぞれのノードは平面上のある領域を分担し，担当領域内の座標を保持する．

Znet では，平面を空間充填曲線の 1 つである Z 曲線を用いて管理する．2 次元平面は Z 曲線によって埋めつくすことができる．ある座標に対し，原点からの Z 曲線上での距離を Z 値と呼ぶ．$0 \leq Z$ 値 $< d^2$ である．図 3 に $d = 8$ の場合の Z 曲線の例を示す．丸の中の数字が Z 値である．XY 座標と Z 値とは簡単な方法で互いに変換できる．

各ノードの担当領域は Z 曲線上で連続した範囲である．範囲の下限を Zmin，上限を Zmax と呼ぶ．各ノードは Zmin を Skip graph に登録する．ある座標を担当するノードは，座標を Z 値に変換し，Z 値と等しいか，より小さい最大のキーを Skip graph から検索することで簡単に求めることができる．このために必要なホップ数は，Znet に参加している物理ノー

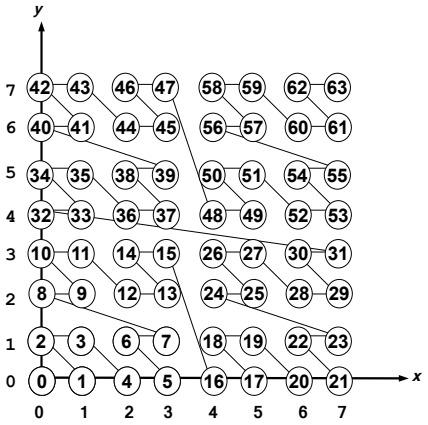

図 3　Z 曲線（$d = 8$ の場合）

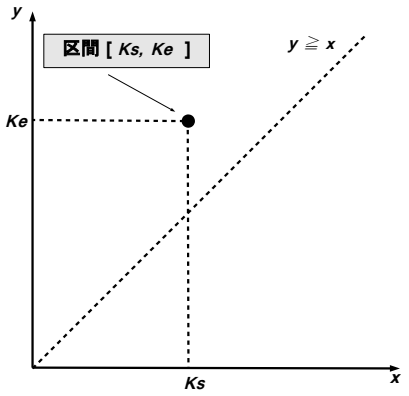

図 4　区間 [Ks, Ke] の 2 次元平面上における表現

ド数 m に対して $O(\log m)$ となる（ここで m はデータの数ではないところに注意）．

新規ノードを挿入する場合，Znet に既に参加しているノード（N とする）の担当領域を分割し，分担させる．N として，なるべく負荷（管理している座標の数）が高いものを選ぶようにすることで，負荷の平準化を図っている．またノードを削除する場合は Skip graph 上で隣接しているノードにデータを引き継ぐ．なお，各ノードが保持するデータは，Skip graph 上で近くにある複数のノードに複製することで，冗長性を確保する．

範囲検索について述べる．検索範囲の矩形を QR としたとき，範囲検索は単純には，Skip graph 上で，QR の左下の座標を担当しているノードから，右上の座標を担当しているノードまでを順番に辿ることで実現できる．しかし，この方法では QR の範囲を担当しないノードまで辿ることになり，効率が悪い．Znet の範囲検索アルゴリズムは検索範囲を担当しないノードをスキップするため，範囲検索を効率よく実現できる (詳細は文献 [3] 参照)．

3　提案手法

1 章で述べたように，本提案手法は，分散 KVS 上で「区間を対象に効率的な範囲検索を可能とすること」を目的としている．このために，提案手法では，キーとなる区間 $[u, v]$ を 2 次元平面上の 1 点に対応づけ，区間の範囲検索を平面上のある領域に存在する点の検索の問題として捉えている．また，2 次元平面上にマップされた点 (各々が区間に対応する) を分散して管理・検索するためには，Znet を用いている．

以降の節では，提案手法について具体的に説明する．

3.1　区間の 2 次元平面上における表現

任意の区間 $[K_s, K_e]$ は，K_s を x 座標，K_e を y 座標に対応させることによって，2 次元平面 (x, y) 上の 1 点 (K_s, K_e) として表現できる (図 4)．このとき，$K_s \leq K_e$ より，区間は必ず平面上の $y \geq x$ の領域にマップされることになる．

3.2　2 次元平面上の範囲検索

キーとなる各区間を 2 次元平面上に登録することによって，「区間に対する範囲検索」を「2 次元平面上の範囲検索」の問題に置き換えることができる．区間の範囲検索には，(1) 指定した範囲に真に含まれる区間を検索する場合，ならびに (2) 指定した範囲と重なりがある区間を検索する場合，の 2 通りが考えられ，各々の場合で検索すべき平面上の範囲が変わることになる．

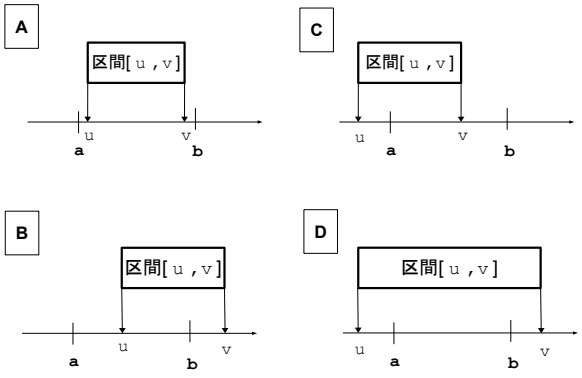

図 5　検索範囲 QR$[a, b]$ と区間の関係

ここで，(1) の，指定した範囲に真に含まれる区間とは，検索する範囲を a から b としたとき (これを検索範囲 QR$[a, b]$ と表記する)，a から b の範囲に完全に含まれる区間を意味する (図 5 の A)．また，(2) の指定した範囲と重なりがある区間とは，次の 3 つのいずれかである．

1. a から b の範囲に完全に含まれる区間 (図 5 の A)
2. a から b の範囲と一部重なりがある区間 (図 5 の B, C)
3. a から b の範囲が完全に含まれる区間 (図 5 の D)

3.2.1　指定した範囲に真に含まれる区間の検索範囲

指定した範囲に真に含まれる区間を検索する場合，検索範囲 QR$[a, b]$ は，2 次元平面上の $x \geq a$ かつ $y \leq b$ の領域で表現できる．つまり，この領域を検索することが，指定した範囲に真に含まれる区間を検索することに等しいと言える．ただし，すべての区間は $y \geq x$ の領域にマップされるので，$y < x$ の

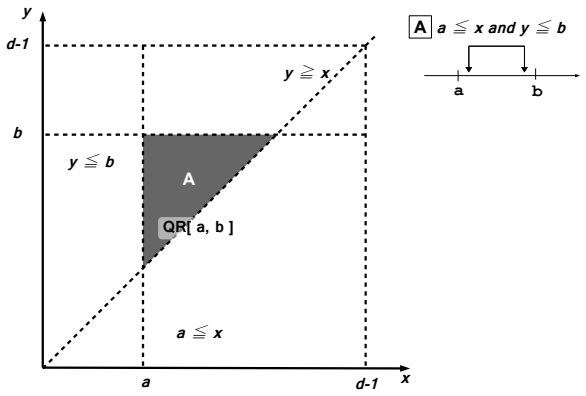

図 6　検索範囲 QR$[a, b]$ に真に含まれる区間に対応する 2 次元領域

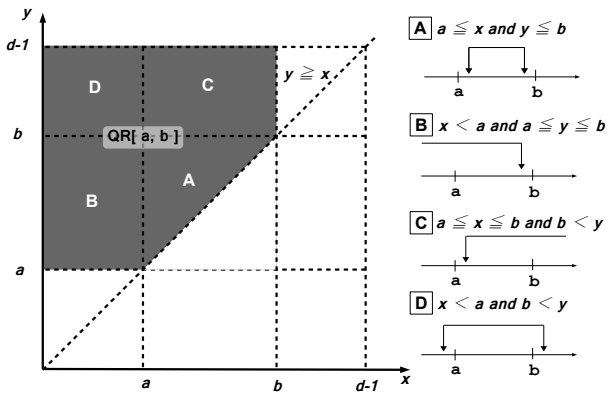

図 7　検索範囲 QR$[a, b]$ と重なりがある区間に対応する 2 次元領域

領域を検索する必要はない．そのため，検索範囲 QR$[a, b]$ に対応する 2 次元平面上の領域は，図 6 の領域 A となる．

3.2.2　指定した範囲と重なりがある区間の検索範囲

指定した範囲と重なりがある区間を検索する場合，検索範囲 QR$[a, b]$ は，図 5 の A，B および C，D の各々に対応して，

1. 指定した範囲に真に含まれる区間に対応する領域 (図 7 の A)
2. 指定した範囲と一部重なりがある区間に対応する領域 (同 B, C)
3. 指定した範囲が完全に含まれる区間に対応する領域 (同 D)

を合わせたものとなる．

3.3　区間の分散管理

前述のように，提案手法では，2.3 節で述べた Znet の方式を利用して 2 次元平面上にある多数の点（すなわち区間）を分散管理する．すなわち，区間の下限，上限をそれぞれ 2 次元平面上の x 座標，y 座標と見なしてその点に対応する Z 値を計算し，その値から Skip graph 上の担当ノードを決定して当該ノードにデータを登録する．

Znet を利用することで，2 次元平面上にマップされた多数の区間を効率良く分散管理できる．

また，区間に対する範囲検索は，検索する範囲を 3.2.1 節あるいは 3.2.2 節で述べた方法で 2 次元平面上の領域に変換した後，Znet の範囲検索アルゴリズムを用いて実行する．2 次元平面上の領域は $y = x$ の線で切れているが，この部分は無視して矩形領域として検索すれば良い．

4　考察

提案手法と RKSG とを比較する．なお，2.2.3 節でも述べたように，そもそも RKSG は分散 KVS のための手法ではないことに注意されたい．

RKSG では，1 つのデータ（区間）が Skip graph の 1 ノードに対応するが，提案手法では，区間を Znet で管理するため，1 物理ノードが Skip graph の 1 ノードに対応する．登録するデータ数を n，物理ノード数を m としたとき，RKSG では，1 データに対して，包含キーを除外しても $O(\log n)$ の領域計算量が必要となるのに対し，提案手法では $O(1)$ となる（ただし，各物理ノードにつき，$O(\log m)$ の経路表が必要）．このため，$m < n$ の場合，提案手法が有利である．

また，RKSG では，データ（区間）の登録・削除は，他の区間との重なり方によっては最悪の場合 $O(n)$ メッセージ必要とするのに対し（2.2.3 節参照），提案手法では $O(\log m)$ メッセージで実行可能である．

文献 [1] では述べられていないが，複数のノードの挿入・削除が並行して行われるような場合でも RKSG の包含キーを正しく保つためには，何らかの制御（分散排他制御，あるいは Chord[4] で用いられているような定期的なスタビライズ処理など）が必要であり，包含キーの維持は（たとえ少数であっても）コストがかかる．提案手法では包含キーのような追加のポインタは不要であり，このため実装も容易である．

さらに，RKSG ではデータの冗長性が考慮されていないが，提案手法では Znet の機能により冗長性が確保される．

5　おわりに

本研究では，区間を 2 次元平面上の点にマップし，Znet を利用して分散管理することによって，区間をキーとして保持する分散 KVS の効率的な実現法を提案した．

本稿では，1 次元上の区間をキーとして分散管理する手法を述べたが，提案手法を 2 次元上の矩形領域をキーとするように拡張することも可能である．2 次元上の矩形領域 $((x_1, y_1)$–$(x_2, y_2))$ は，4 次元空間上の 1 点 (x_1, y_1, x_2, y_2) にマッピングできるため，2 次元上の矩形領域に対する範囲検索を 4 次元空間上の点に対する範囲検索に帰着できる（Znet は 4 次元空間を扱うことも可能である）．

今後の課題としては，提案手法の実装および定量的な評価を行い，関連研究との性能比較を行うことなどが挙げられる．

参考文献

[1] 石芳正, 寺西裕一, 吉田幹, 下條真司, 西尾章治郎. 範囲をキーとして保持可能とする skip graph 拡張の提案. 情報

処理学会研究報告, Vol. 2009-DPS-139, No. 1, pp. 1–7, 2009.

[2] James Aspnes and Gauri Shah. Skip graphs. *ACM Trans. on Algorithms*, Vol. 3, No. 4, pp. 1–25, 2007.

[3] Yanfeng Shu, Beng Chin Ooi, Kian-Lee Tan, and Aoying Zhou. Supporting multi-dimensional range queries in peer-to-peer systems. In *Proc. of the 5th IEEE Intl. Conf. on Peer-to-Peer Computing*, pp. 173–180. IEEE Computer Society, 2005.

[4] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. *ACM SIGCOMM Computer Communication Review*, Vol. 31, No. 4, pp. 149–160, 2001.