5L-03

並列分散処理における空間データパーティショニング手法の比較と分析

堀 敬三 † 佐々木 勇和 ‡ 天方 大地 § 鬼塚 真 ¶
大阪大学 † 大阪大学 ‡ 大阪大学 § 大阪大学 ¶

1 はじめに

時々刻々と生成される膨大な空間データに対する高速な分析処理は，ビッグデータ社会を支えるための必要要件である．現在では，膨大な空間データに対する複雑な空間分析の必要性が高まり，Apache Sedona[*1] や SpatialHadoop[*2] などの空間データに特化させた並列分散フレームワークが開発されている．これらの並列分散処理では計算処理をスケールアウトするために，データ全体をパーティションと呼ばれる小さな単位に分割し，複数のマシンにデータを分散させるデータパーティショニングが用いられる．

空間データパーティショニングは並列分散処理におけるクエリの実行時間に大きく影響し，パーティショニングにおいて以下の課題が挙げられている [1]．第一に，位置的に近いデータ同士を同じパーティションに分割することで効率的なアクセスを可能にする．第二に，パーティション毎のデータ数を同程度に分割することでマシン間のワークロードバランスを保つ．第三に，境界オブジェクトを最小化することでオーバーヘッドを削減する．境界オブジェクトとは，ポリゴンのような範囲を持つジオメトリタイプにおいて，複数のパーティションにまたがる位置に存在するデータのことを指す．

これらの課題から，データ分布やクエリ特性はパーティショニング手法毎でのクエリ性能に影響を与えると予想される．より高速な分析処理のためには，データ分布やクエリワークロードに最適なパーティショニング手法を選択することが重要である．そこで，本稿では複数のデータセットやクエリワークロードを用いて，空間データパーティショニング手法を比較し，既存手法の有効性を明らかにする．

A comparsion and analysis of spatial data partitioning methods in parallel distributed processing
† Keizo Hori, Osaka University
‡ Yuya Sasaki, Osaka University
§ Daichi Amagata, Osaka University
¶ Makoto Onizuka, Osaka University
*1 https://sedona.apache.org/.
*2 http://spatialhadoop.cs.umn.edu/.

2 既存手法

空間データパーティショニング手法には，データ分布を考慮した様々なアルゴリズムがある．最も基本的なアルゴリズムは Uniform Grids と呼ばれ，二次元空間に存在するデータに対して均一サイズのグリッドセルを用いて分割するため，一様な分布を持つデータに適している．しかし，多くの場合データ分布には偏りが存在するため, Uniform Grids では効果的な分割は期待できない．そこで，データ分布に合わせて不均一なサイズのグリッドセルを用いて分割する方法として，R-Tree [2]，Quad-Tree [3]，KDB-Tree [4] などがある．例えば，R-Tree は位置の近いデータ群の最小外接矩形を階層的に入れ子になるように二次元空間に分割し，Quad-Tree は各パーティションが指定したデータ数に至るまで二次元空間を再帰的に四つの領域に分割する．これにより，データが多く集まる空間はより細かく分割され，パーティション間でのデータ数のばらつきが抑えられる．

3 評価実験

3.1 実験設定

表 1 に示す Open Street Maps (OSM) のデータセットを実験に使用する [5]．以降では，各データセットをジオメトリタイプ別に Point, LineString, Polygon, Rectangle と表記する．

比較手法には既存の空間パーティショニング手法から Uniform Grids，R-Tree，Quad-Tree，KDB-Tree の 4 種類のパーティショニング手法を用いる．それぞれの手法に対して，Range Query，kNN Query，Distance Join，Spatial Join の 4 種類のクエリを実行する．ここで，Spatial Join は 2 種類のデータセット間で包含されるものを結合する処理である．手法毎に各クエリの 10 回の実行時間を測定し，その平均時間を用いて評価する．

並列分散処理環境には Spark クラスタとして 1 台のマスターノードと 3 台のスレーブノードを配置

表1　データセット

データセット	ジオメトリタイプ	レコード数
OSM Nodes	Point	10 million
OSM Roads	LineString	10 million
OSM Polygons	Polygon	10 million
OSM Rectangles	Rectangle	10 million

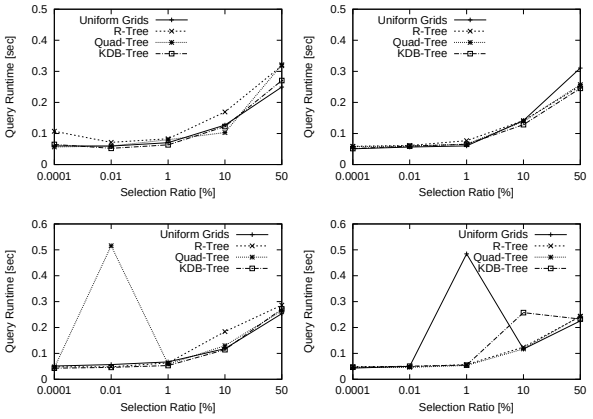

図1　Range Query (左上: Point, 右上: Linestring, 左下: Polygon，右下: Rectangle)

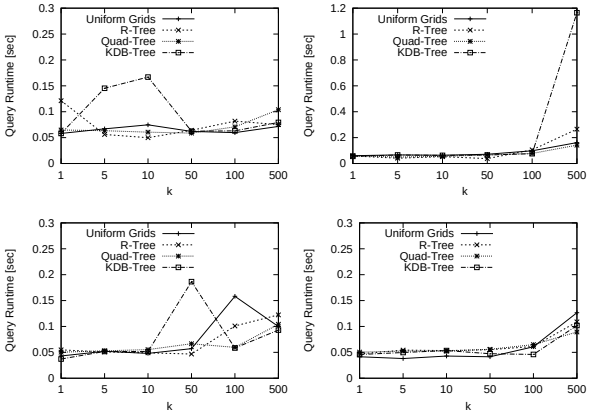

図2　kNN Query (左上: Point, 右上: Linestring, 左下: Polygon，右下: Rectangle)

する．また，空間データ用の並列分散処理フレームワークとして Apache Sedona を利用する．

3.2 結果

クエリ毎の比較結果を図 1–4 に示す．図 1 の Range Query では Point, LineString, Polygon に対して KDB-Tree のクエリ実行時間が比較的小さい．しかし，Rectangle では R-Tree や Quad-Tree の方が優れている．図 2 の kNN Query では Quad-Tree がどのクエリにおいても安定しているが，近傍数 k の値により優れている手法も変化している．図 3

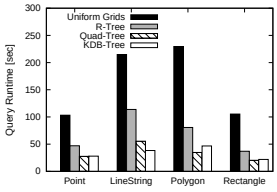

図3　Distance Join

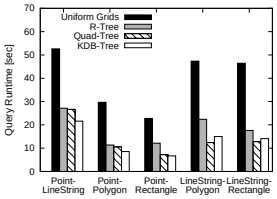

図4　Spatial Join

の Distance Join と 図 4 の Spatial Join では全体的に Quad-Tree と KDB-Tree のクエリ実行時間が小さく，使用したデータセットによって優れている手法が異なっている．

Range Query と kNN Query の 2 つは元々高速であり，特に Range Query ではパーティション間のシャッフルを必要としない処理である [6]．そのため，実行時間的には手法間で顕著な差は見られなかった．その一方で， Distance Join および Spatial Join においては，空間パーティショニングがクエリ実行時間にも大きく影響している．以上の実験を通して，データセットやクエリに対して適切な空間パーティショニング手法が異なることが確認できる．

謝辞 本研究は JSPS 科研費 JP20H00584 の助成を受けたものです．

参考文献

[1] Hoang Vo, Ablimit Aji, and Fusheng Wang. SATO: A spatial data partitioning framework for scalable query processing. In SIGSPATIAL, pp.545–548, 2014.

[2] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In SIGMOD, pp.47–57,1984.

[3] R. A. Finkel and J. L. Bentley. Quad trees a data structure for retrieval on composite keys. In Acta Informatica, Vol.4, No.1, pp.1–9, 1974.

[4] John T. Robinson. The K-D-B-Tree: A search structure for large multidimensional dynamic indexes. In SIGMOD, pp.10–18, 1981.

[5] Varun Pandey, Andreas Kipf, Thomas Neumann, and Alfons Kemper. How good are modern spatial analytics systems? In VLDB Endowment, Vol.11, No.11, pp.1661–1673, 2018.

[6] Jia Yu, Zongsi Zhang, and Mohamed Sarwat. Spatial data management in Apache Spark: The GeoSpark perspective and beyond. In Geoinformatica, Vol.23, No.1, pp.37–78,2019.