

# NoSQL and Spark SQL

Junggil Park  
ASU ID: 1223940471  
ASU Master of Computer Science,  
Ira A. Fulton Schools of  
Engineering  
Arizona, US  
johnjgpark@gmail.com

**Abstract—** This document is a portfolio report on NoSQL and Spark SQL (Hotspot Analysis) for Data Processing at Scale. This paper has three main objectives: how to retrieve data from NoSQL MongoDB datasets; how to identify hot spots on two-dimensional rectangle datasets using Spark SQL; how to apply spatial statistics to spatio-temporal big data so as to identify significant spatial hot spots statistically using Spark SQL and Scala. Background details of data processing tools such as NoSQL and Spark, the approach to the solution, and quick snapshots of the results are provided in each method.

**Keywords—** big data, NoSQL, MongoDB, JSON, BSON, Apache, Spark, SQL, data processing, hot zone, hot cell, z-score (keywords)

## I. INTRODUCTION

Today, Data has been known as the oil of the digital economy. There will be big rewards for those who see data's fundamental value and learn to extract and use it such as oil [1]. These days, The size and number of available data set keep growing exponentially along with advanced IT technologies and devices such as mobile devices, tablets, cameras, microphones, and laptops. According to data age 2025 by a research firm in 2018, it is predicted that the global data may exponentially increase from 33 zettabytes in 2018 to 175 zettabytes by 2025[2].

Relational database management systems(RDBMS) and classic structured database systems which are used in many fields for a long time often have difficulty in processing and analyzing big data. The analysis and processing of big data need huge parallel software running on even thousands of servers[3].

NoSQL databases have been increasingly and broadly used in big data and real-time web applications[4]. A NoSQL originally referring to non-SQL or non-relational is also sometimes called "Not Only SQL" to emphasize that they may support SQL-like query languages[5].

Apache Spark is an open-source, distributed processing system utilized for big data systems. It uses in-memory caching and optimized query execution for fast queries against data of any size. Spark is a fast and general engine for big data or large-scale data processing[6].

This paper has three main objectives: 1) how to retrieve data from NoSQL MongoDB datasets; 2) how to identify hot spots on two-dimensional rectangle datasets using Spark SQL; 3) how to apply spatial statistics to spatio-temporal big data so as to identify significant spatial hot spots statistically using

Spark SQL and Scala. Each technology includes the background of data processing tools, the approach to the solution, and quick snapshots of the results. The three methods show how to use big data processing tools in order to retrieve data from the database or data frame.

## II. NoSQL

By using Python code and the function, filtering and finding business based on the city and the location from the given MongoDB file.

### A. Background

A NoSQL database that is non-SQL or non-relational provides a mechanism for storage. Since it is unstructured and semi-structured, retrieval of data is a bit different from the tabular relations used in relational databases. The data structures of NoSQL databases are much more flexible than relational database tables[7].

MongoDB, one of the NoSQL database programs, uses JSON-like documents with optional schemas. MongoDB is also one of the document-oriented databases, which is a data storage system designed for storing, retrieving, and managing document-oriented information, also known as semi-structured data[8]. The implementation of each document-oriented database may differ depending on the details of the definition, but generally, they assume documents encapsulate and encode data in some kind of standard format such as XML, YAML, JSON, as well as binary forms like BSON in figure 1.

```
{  
  "FirstName": "Bob",  
  "Address": "5 Oak St.",  
  "Hobby": "sailing"  
}
```

Figure 1. JSON file representation example

This project imports a MongoDB data file "sample.db" and finds business based on the city and the location, by using python code and functions.

```
{ '_id': 0,
  'business_id': b'MPyxANVuhLAQqJ0iKV5rQw',
  'categories': [b'Restaurants', b'Buffets', b'Italian'],
  'city': b'Tempe',
  'full_address': b'1835 E Elliot Rd, Ste C109, Tempe, AZ 85284',
  'latitude': 33.3482589,
  'longitude': -111.9088346,
  'name': b'VinciTorio's Restaurant',
  'neighborhoods': [],
  'open': True,
  'review_count': 122,
  'stars': 4,
  'state': b'AZ',
  'type': b'business'}
```

Figure 2. Given MongoDB data file format

## B. Solution & Result

### a. Find business based on city

This project is to build a function that searches the collection given to find all the business present in the city provided as a parameter and save them to text file, using python and MongoDB data file.

First, in Jupyter notebook, import library unqlite and import the file sample.db which is exported from MongoDB. UnQLite is an embedded NoSQL database engine. It's a document-store database similar to MongoDB with a built-in scripting language that looks like Javascript.

In Figure 2, the type of the city name is not string type in python. Therefore, the type need to be changed from bytes to string by using “.decode” in order to get rid of ‘b’ in front of city name in DB data.

Next, the file name “output\_city.txt” is exported to local drive. As you may see on code, each line of the saved file contains: Name\$FullAddress\$City\$State in Figure 3. As seen in Figure 3, three items have been filtered and saved to text file ‘output\_city.txt’ correctly.

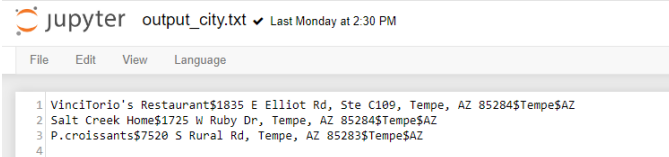


Figure 3. Output file to find business on city

### b. Find business based on location

This project is to build a function that searches the collection given to find the name of all the businesses present in the maximum distance from the given my location and save them to text file, using python and MongoDB data file.

First, math library is imported inside the location function. This is crucial, otherwise, error comes up on Autograder.

After calculating the distance, if the distance is less than maximum distance and the category is in the list of DB's categories, the result it saved and exported to “output\_loc.txt”. Also, the type need to be changed from bytes to string by using “.decode” such as part a. Only one item has been filtered and saved to text file ‘output\_loc.txt’ correctly as seen in Figure 4.

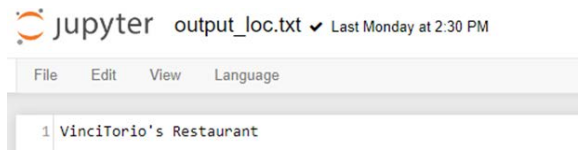


Figure 4. Output file to find business on location

## III. SPARK -HOT ZONE ANALYSIS

For this paper, Hot zone analysis is required to count how many points are within the range of rectangle datasets by using Apache Spark with Scala. In other words, the number of points located within each rectangle will be counted. When the rectangle includes more points, it is called as hotter rectangle. This part is to calculate the hotness of all the rectangles.

### A. Background

Spark is a distributed processing system for big data, uses in-memory caching, and optimizes query execution for fast queries of big data processing [9]. It's faster than previous approaches to work with Big Data like the classical MapReduce of Hadoop. This is because Spark runs on memory (RAM), and that makes the processing much faster than on disk drives in MapReduce [6]. Spark can be used for creating data pipelines, running distributed SQL, processing data into a database, running Machine Learning and Deep Learning algorithms, and so forth [6]. Due to fast processing which is the most important feature of Spark, it has been chosen for the big data world over others. Also, Resilient Distributed Dataset (RDD) in Spark can save a lot of time in reading and writing operations. RDD is Spark is allowed to run almost ten to one hundred times faster than Hadoop.

In terms of flexibility, Spark supports multiple languages and allows the users to write the code in Scala, R, Java, or Python [6]. In contrast to MapReduce of Hadoop that contains Map and Reduce functions, Spark includes much more than MapReduce, which consists of a rich set of SQL queries, machine learning algorithms, complex analytics, and so forth.

Spark has been exponentially growing over the past several years and becoming the most effective data processing today due to ease of use, its speed, and analytics for Deep Learning. However, the cost of Spark is more expensive than others because it uses lots of memory (RAM).

This project uses Spark-shell, Spark SQL, and Scala for both hot zone analysis and hot cell analysis.

### B. Solution & Result

First of all, Spark 2.4.7, Hadoop 2.7, and Java 8 need to be installed. IntelliJ Idea with Scala plug-in are also used for debugging.

For the hot zone analysis, it is necessary to find the number of points in the rectangles in the given skeleton code and dataset. User defined function and spark SQL for finding points in rectangles has been already defined. It is only required to modify given ST\_Contains function to complete hot zone analysis.

In the ST\_Contains function, only when the points are in the rectangles, true is returned. Otherwise, the function returns false. Spark SQL only filters rectangles and points form the given tables using join, when the where clause is true. Then,

by using group by and order by, it is possible to count and sort the Scala dataframe.

After completing the function code, compiling and testing the code is required. Finally, the result is the exactly same as the expected result example which the is as seen in figure 5.

	A	B
1	-73.789411,40.666459,-73.756364,40.680494	1
2	-73.793638,40.710719,-73.752336,40.730202	1
3	-73.795658,40.743334,-73.753772,40.779114	1
4	-73.796512,40.722355,-73.756699,40.745784	1
5	-73.797297,40.738291,-73.775740,40.770411	1
6	-73.802033,40.652546,-73.738566,40.668036	8
7	-73.805770,40.666526,-73.772204,40.690003	3
8	-73.815233,40.715862,-73.790295,40.738951	2
9	-73.816380,40.690882,-73.768447,40.715693	1
10	-73.819131,40.582343,-73.761289,40.609861	1
11	-73.825921,40.702281,-73.790734,40.719217	2
12	-73.826577,40.757744,-73.790317,40.779587	1
13	-73.832707,40.620010,-73.746541,40.665414	200
14	-73.839460,40.746988,-73.815375,40.781725	3

Figure 5. final result and expected result example

#### IV. SPARK - HOT CELL ANALYSIS

By using Apache Spark with Scala, this task will focus on applying spatial statistics to spatio-temporal big data in order to identify statistically significant spatial hot spots. The topic of this task is from ACM SIGSPATIAL GISCUUP 2016.

##### A. Background

A collection of New York City Yellow Cab taxi trip records spanning January 2009 to June 2015. The source data may be clipped to an envelope encompassing the five New York City boroughs in order to remove some of the noisy error data[10].

As seen in Figure 6, the neighborhood for each cell in the space-time cube is established by the neighbors in a grid based on subdividing latitude and longitude uniformly. This spatial neighborhood is created for the preceding, current, and following time periods For simplicity of computation, the weight of each neighbor cell is presumed to be equal[10]. For example, each cell has 26 neighbors, thus total weights in a cell are usually 27, which does not lie on boundaries.

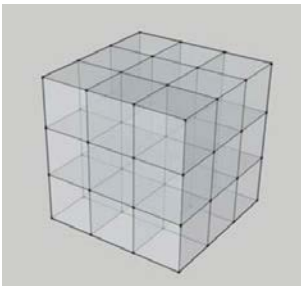


Figure 6. Neighborhood for each cell in the space-time cube



Figure 7. Space-time cube

In Figure 7, time and space should be aggregated into cube cells as specified on the given skeleton python code. A cubic cells forms a space-time cube together.

When identifying statistically significant clusters which is often termed hot spot analysis, a very commonly used statistic is the Getis-Ord statistic, which provides a z-score in order to determine where features with low or high values are clustered spatially. It is important to note that this statistic can be applied to both the spatial and spatio-temporal domains. This project focused on the spatio-temporal Getis-Ord statistic or z-score to evaluate spatiality.

$$G_i^* = \frac{\sum_{j=1}^n w_{i,j} x_j - \bar{X} \sum_{j=1}^n w_{i,j}}{S \sqrt{\left[ n \sum_{j=1}^n w_{i,j}^2 - \left( \sum_{j=1}^n w_{i,j} \right)^2 \right] / (n-1)}}$$

Equation 1. the Getis-Ord statistic or Z-score

$$\bar{X} = \frac{\sum_{j=1}^n x_j}{n} \quad S = \sqrt{\frac{\sum_{j=1}^n x_j^2}{n} - (\bar{X})^2}$$

Equation 2. Mean and Standard Deviation

By using equations 1 and 2, z-score is able to be calculated, where  $x_j$  is the attribute value for the cell,  $w$  is the spatial weight between cell  $i$  and  $j$ ,  $n$  is equal to the total number of cells. The output is a list of the fifty most significant hot spot cells in time and space is generated and exported into a CSV file.

##### B. Solution and Result

For the hot cell analysis, using Spark SQL, mainly two queries are required to calculate the z-score. The first query is for selecting the cell coordinate and counting pickups to get how many pickups are in the distinct coordinates of the cell. The second query is for including cell coordinates and sum of counts of the first query in the range of neighbor 27 cells.

For this project, a User-Defined Function(UDF) is used, which is one of the most useful and powerful column-based functions in Spark SQL to be able to perform the transformation of datasets. Weights are calculated, based on cell boundaries such as it returns 18 if the point lies on x, y, or z-boundary, it returns 12 if the point lies on x and y boundary, y and z boundary, or x and z boundary, and it returns 8 if the point lies on x, y, and z boundary.

Using the given equation 2, mean  $\bar{X}$  and Standard Deviation  $S$  are calculated. Then, using withColumn method which is the column-based function in Spark SQL, it is easy to calculate z-score using the given equation 1.

After completing the Spark SQL and Scala code, compiling and testing the code is required. Finally, the result of z-score is the exactly same as the expected result example as seen in figure 8.

	A	B	C
1	-7399	4075	15
2	-7399	4075	22
3	-7399	4075	14
4	-7399	4075	29
5	-7398	4075	15
6	-7399	4075	16
7	-7399	4075	21
8	-7399	4075	28
9	-7399	4075	23
10	-7399	4075	30

	A	B	C
1	-7399	4075	15
2	-7399	4075	22
3	-7399	4075	14
4	-7399	4075	29
5	-7398	4075	15
6	-7399	4075	16
7	-7399	4075	21
8	-7399	4075	28
9	-7399	4075	23
10	-7399	4075	30

Figure 8. final result and expected result example

## V. LESSON LEARNED AND CONCLUSION

This paper investigated how to retrieve data from JSON datasets of MongoDB files by using Python. The most challenge of this part is to figure out how to decode the city data in the MongoDB file. The type of data is supposed to be changed from bytes to string. Also, even though all passed correctly on the local machine, it was not easy for Autograder to score full grades due to importing library and the location of function.

This paper also included how to use Spark-shell, Spark SQL, and Scala for both hot zone analysis and hot cell analysis. As compared to for-loop with dataframe, Spark SQL is much faster than using for-loop because spark uses in-memory caching. In practice, I was surprised that it takes only a few minutes to retrieve information from over the 2.5GB CSV file using Spark SQL. Also, user defined function in Spark SQL is a great method to manipulate and massage the transformation of data or ETL. Overall, Spark SQL is optimized for big data processing due to its speed, which makes it easy to utilize deep learning with big data set. As data keeps growing up and bigger, it needs to be processed efficiently and effectively to use deep learning technologies with big datasets.

## VI. REFERENCES

- [1] J.T. Yonego (2014, July). "Data Is the New Oil of the Digital Economy" *Wired*, Available: <https://www.wired.com/insights/2014/07/data-new-oil-digital-economy/>
- [2] T. Greateon (2019, Dec). "What's causing the exponential growth of data?" *Nikkoam*, Available: [https://insights.nikkoam.com/articles/2019/12/whats\\_causing\\_the\\_exp\\_onential](https://insights.nikkoam.com/articles/2019/12/whats_causing_the_exp_onential)
- [3] A. Jacobs (2009, July). "The Pathologies of Big Data" *ACMQueue Volume 7, issue 6*, Available: <https://queue.acm.org/detail.cfm?id=1563874>
- [4] P. Andlinger (2013, Nov). *RDBMS dominate the database market, but NoSQL systems are catching up* [Online]. Available: [https://db-engines.com/en/blog\\_post/23](https://db-engines.com/en/blog_post/23)
- [5] C.S. Mullins (2021, April). *NoSQL (Not Only SQL database)* [Online]. Available: <https://searchdatamanagement.techtarget.com/definition/NoSQL-Not-Only-SQL>
- [6] R. Joseph (2021). *What is Spark?* [Online]. Available: <https://chartio.com/learn/data-analytics/what-is-spark/>
- [7] W. Vogels (2012, Jan. 18). *Amazon DynamoDB – a Fast and Scalable NoSQL Database Service Designed for Internet Scale Applications*.

[Online]. Available: <https://k21academy.com/amazon-web-services/amazon-dynamodb/>

- [8] M. Drake (2019, Aug. 9). *A Comparison of NoSQL Database Management Systems and Models* [Online]. Available: <https://www.digitalocean.com/community/tutorials/a-comparison-of-nosql-database-management-systems-and-models>
- [9] AWS (2021) *Introduction to Apache Spark* [Online]. Available: <https://aws.amazon.com/big-data/what-is-spark/>
- [10] ACM SIGSPATIAL (2016) *ACM SIGSPATIAL Cup 2016 Problem Definition* [Online]. Available: <http://sigspatial2016.sigspatial.org/giscup2016/problem>