CHAPTER 5

5. EXPERIMENTAL EVALUATION

5.1.INTRODUCTION

In Chapter 2, we have discussed several approaches that are developed to add support to spatial data and queries to the Hadoop platform. These approaches are expected to improve the performance of spatial data processing. In Chapter 3, we have presented a brief discussion about Co-SpatialHadoop, and how it uses SpatialHadoop [1] indexing and locator table idea which is similar to those in Co-Hadoop [2]. Besides, we have introduced also the inverted indexes that enhance non-spatial operations. In this chapter, we present experiments that we have designed to show the effectiveness of the new approach that we are proposing, and we compare its performance with that of SpatialHadoop. For this purpose, we use several spatial data sets with various sizes.

This chapter is organized as follows. In section 4.2, we discuss the cluster setup and instances types we used in our experiments. We additionally describe the tools that we have used in our experiments. The results are illustrated in Sections 4.3, 4.4, 4.5, 4.6 and 4.7 Finally, Section 4.8 concludes the chapter.

5.2.EXPERIMENTAL SETUP

5.2.1. Performance Metrics

Co-SpatialHadoop main goals are: (1) enhancing the network usage of executed queries using locator table idea introduced by Co-Hadoop [2], and (2) enhancing the execution time and network usage of non-spatial queries. To evaluate performance of these goals, we used the following metrics in our experiments:

1- **Network Overhead**: The average of network usage during the execution of a query on a cluster of machines.
2- **Query Execution Time**: The total time needed to complete the execution of a query.
3- **Locator Table partitioning method**: The approach used to build Spatial locator table that guarantees load balancing. These approaches are discussed in Section 3.6.2.
4- **Number of Locator Items**: The number of locator items created in the locator table.
5- **Number of remote Map functions**: The number of map functions that read their input blocks remotely during the execution of a query.
6- **Word occurrence**: This term is used by inverted index experiments; it measures the occurrence of a word among file's blocks.
7- **Spatial file size**: we use different file sizes in the experiments.

5.2.2. Environment Setup

We use EC2 cluster of 10 "t2.meduim" nodes. Each node has two cores (2.5GHz) running 64-bit platform ubuntu 14.04 OS, 150GB hard drive and 4GB memory. We run Hadoop 0.20 on the nodes. Hadoop is configured to use the default block size of 64 MB and the default replica number of three.

5.2.3. Environment tools

We use Ganglia Monitoring system [30] to monitor network overhead needed by queries and Hadoop MapReduce counters to measure the needed time by these queries.

Figure 4.1 shows a screen shot of the monitoring modules of Ganglia while executing a query. Each module of Ganglia measures a different metric. The important metric needed by our experiments is the network overhead. Figure 4.2 shows the network overhead module. This module measures the total input and output packets used by the cluster in a given time frame. We set this frame as time needed to execute a query or run an entire experiment. In the results we are presenting in our experiments, we use the average of the input and output packets as measured by Ganglia.

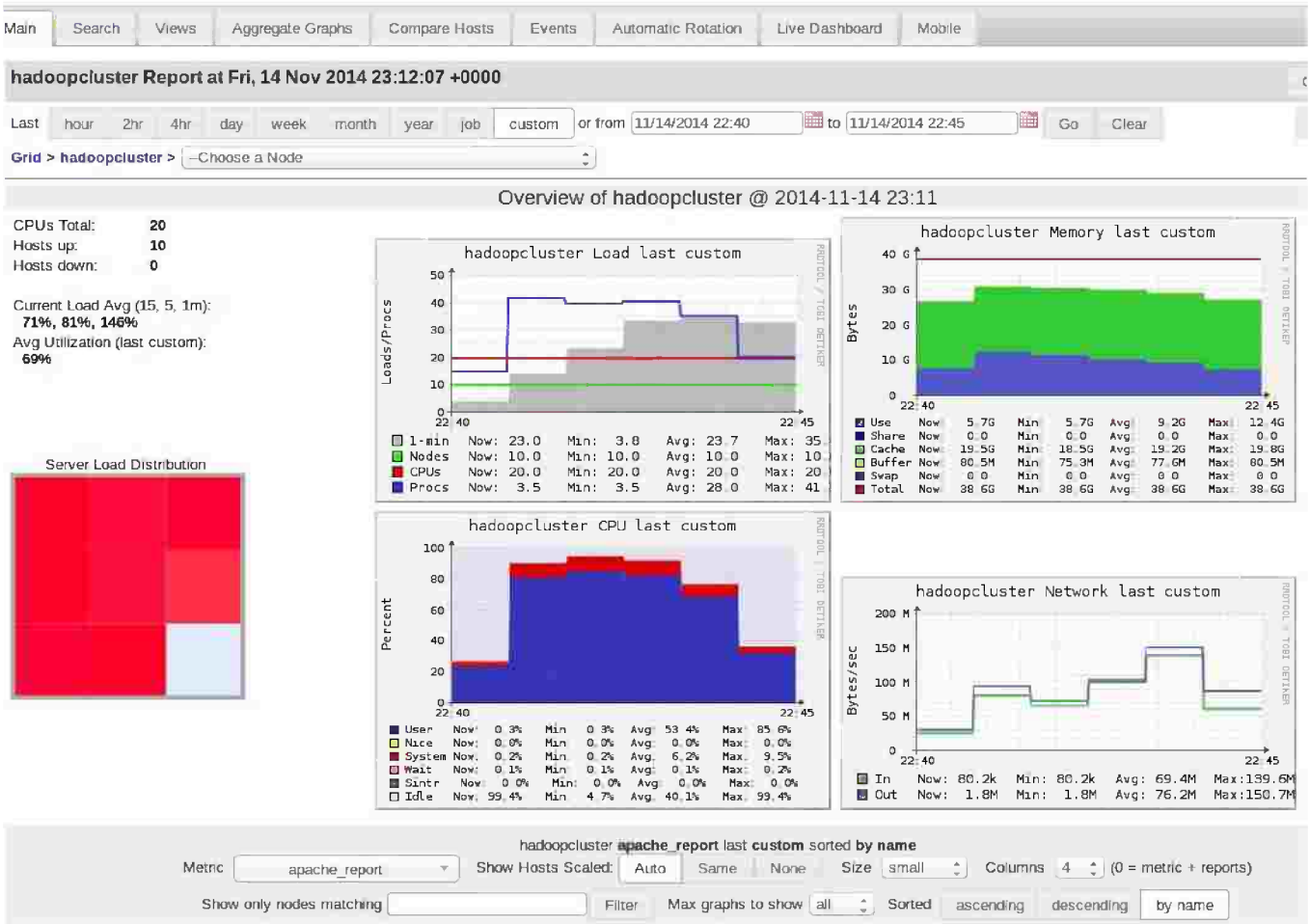

Figure 5.1: The Ganglia monitoring system

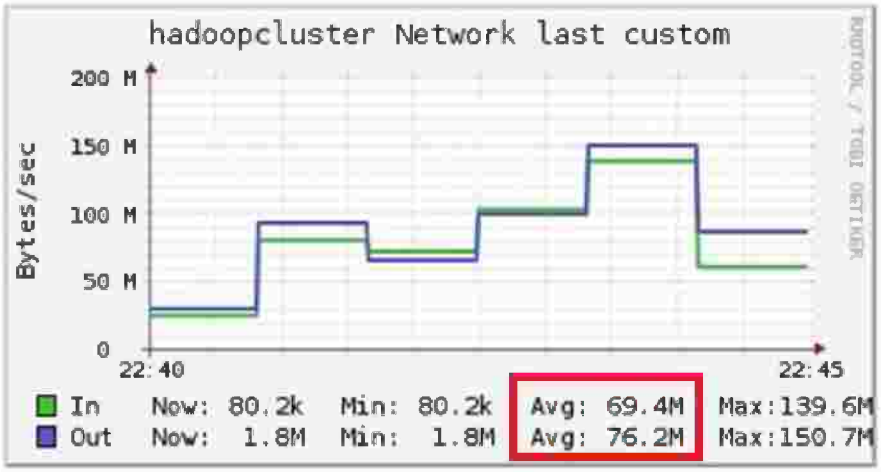

Figure 5.2: Network overhead measured by Ganglia for a query

5.2.4. Input Data

Input data used by the experiments are Open Street Map (OSM) data files that are downloadable from SpatialHadoop [1]. We use these files:

- Cities file (844 MB), it is partitioned during the indexing to 21 blocks (18 Cells)
- Rivers file (945 MB), it is partitioned during the indexing to 22 blocks (18 Cells)
- Lakes file (2.6 GB), it is partitioned during the indexing to 63 blocks (45 Cells)
- Roads file (20.6GB), It is partitioned during the indexing to 471 blocks (336 Cells)

5.2.5. Workload of Queries

In the experiments presented in Sections 4.3, 4.4, and 4.5, we use a set of spatial join queries that can be downloaded from SpatialHadoop to evaluate colocation algorithm introduced by Co-SpatialHadoop.

To evaluate the effectiveness of creating inverted indexes on non-spatial data, we use a workload on non-spatial queries that are described in Section 4.7. The non-spatial query selects all the records that contain a specific word. We run this query with various words, where each of them has a different selectivity.

5.3.EFFECTIVENESS OF COLOCATING SPATIAL FILES

We partition the locator table to five locator items using R-Tree cells info of roads file as Figure 3.6 shows. Using this locator table, we index all spatial files needed in the experiments to evaluate the colocation algorithm introduced by Co-SpatialHadoop. Besides, we index the same files using R-Tree index operation of SpatialHadoop [1] – without colocation. We run five different spatial join queries, each one is performed twice, the first one using the files indexed by SpatialHadoop, the second one using the colocated files. Table 4.1 describes the five workload queries used by this experiment.

Input files	Joined blocks	Needed map functions
cities x roads	21 x 471	950
lakes x roads	63 x 471	1310
rivers x roads	22 x 471	850
lakes1 x lakes2	63 x 63	400
roads1 x roads2	471 x 471	3100

Table 5.1: Experiments Queries

The first column illustrates the two input files of each query, the second column is the number of joined blocks and the last column is the number of executed map functions to join the two files. Figure 4.3 compares the network overhead of five queries using SpatialHadoop and Co-SpatialHadoop and Table 4.2 shows some statistics about this experiment. The first column shows workload queries, the second column illustrates the percentage of the network overhead enhancement, and the third column illustrates the percentage of map functions that read data remotely to the total number of map functions executed for one query. The forth column illustrates the size of the output file and the last column illustrates the percentage of the output size to the input size.

We can see that the first two queries, cities and rivers files have almost the same size but the network overhead enhancement percentage for both queries are different. The percentage of remote map functions in rivers case is the highest because the blocks of the rivers file are not well distributed over locator table items as shown in Table 4.3 that illustrates the distribution of blocks over locator table items. Table 4.3 shows that there are overlapping between rivers blocks and locator items. These two problems are dis-

cussed in details in Section 3.5. Rivers file has the worst distribution over locator table so “rivers x roads” query has the highest remote map functions, and the lowest network overhead enhancement. Besides, the output file size affects the network overhead percentage because it is replicated to three replicas in different three nodes. The two queries “cities x roads” and “lakes x roads” have the same percentage of remote map functions but the percentage of the output size to the input size is not the same. The size of “lakes x roads” query output is less than the other, so network overhead enhancement is better. The effect of the output size also appears on query 5. This query has the lowest percentage of remote map functions but its enhancement percentage is not higher than “lakes x roads” query that is due to its output size.

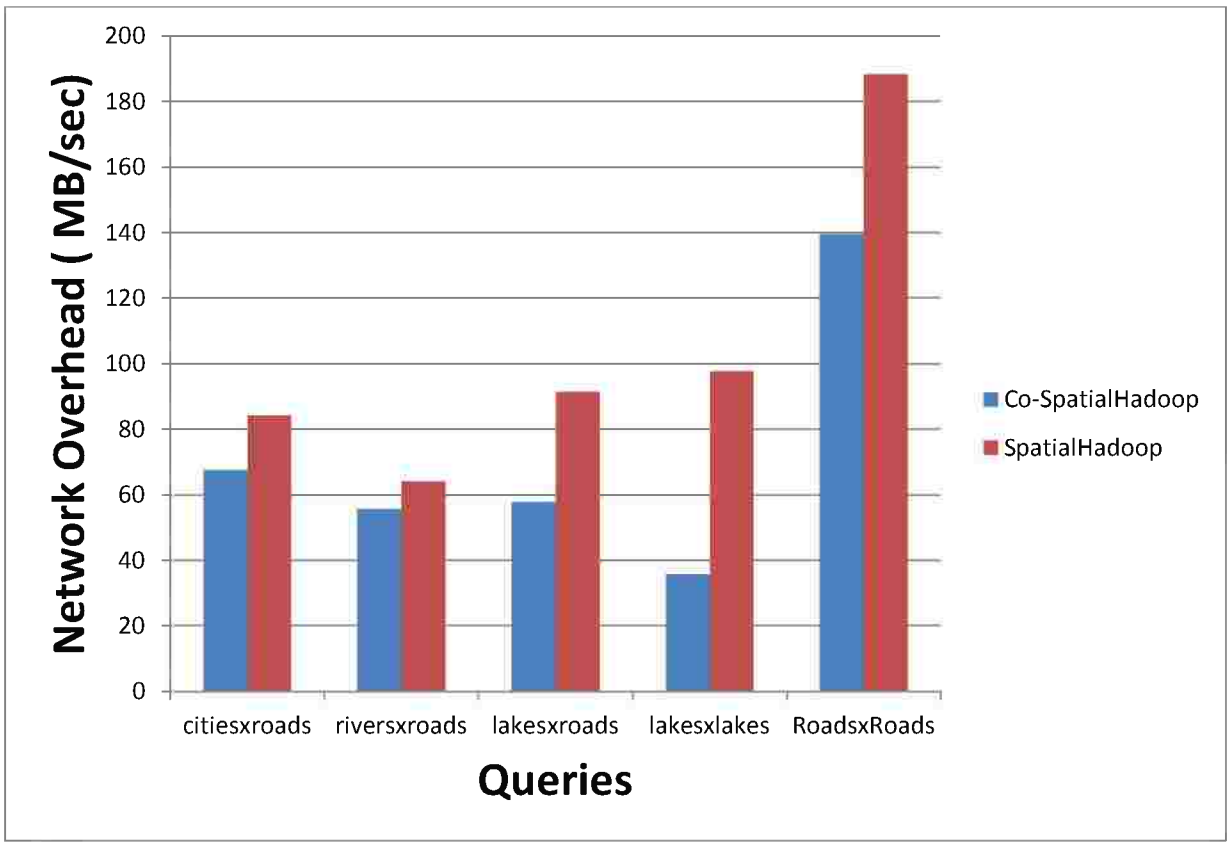

Figure 5.3: Network overhead chart

Query	Network overhead enhancement	Remote map percentage	Size of the output file	Percentage output to the input
cities x roads	19.6%	11.2%	4.3 GB	0.025%
rivers x roads	13.2%	16.5%	289 MB	0.001%
lakes x roads	36.7%	11%	244 MB	0.00043%
lakes x lakes	63.3%	13.4%	848 MB	0.012%
roads x roads	25.8%	9.6%	138 GB	0.032%

Table 5.2: some statistics for the experiment

Locator Items	Roads1	Roads2	Lakes1	Lakes2	Cities	Rivers
L0	85	108	18	18	4	2
L1	100	91	17	17	4	3
L2	82	86	3	3	4	2
L3	91	94	5	5	2	7
L4	114	102	19	20	6	8

Table 5.3: blocks distribution over Locator Table items

In addition, from Table 4.3 we can notice the effect of using "roads" file in locator table partitioning. If locator table was partitioned using "rivers" file, the "rivers" file has been well distributed –unlike our case- but "roads" file has not, and in general, the locator table items has not been balanced which causes high percentage of remote map functions in any query uses "roads" file and the network overhead has not been enhanced with a big ratio. It is better to use the biggest file in the partitioning of locator table to balance its locator items with the big number of this file's blocks.

Finally, Figure 4.4 shows that there is not enhancement in the response time and that both types of queries have almost the same response time. It is noted in [2] that the enhancement of execution times of queries due to colocating the files that they access can be noticed with input files size larger than 70 GB.

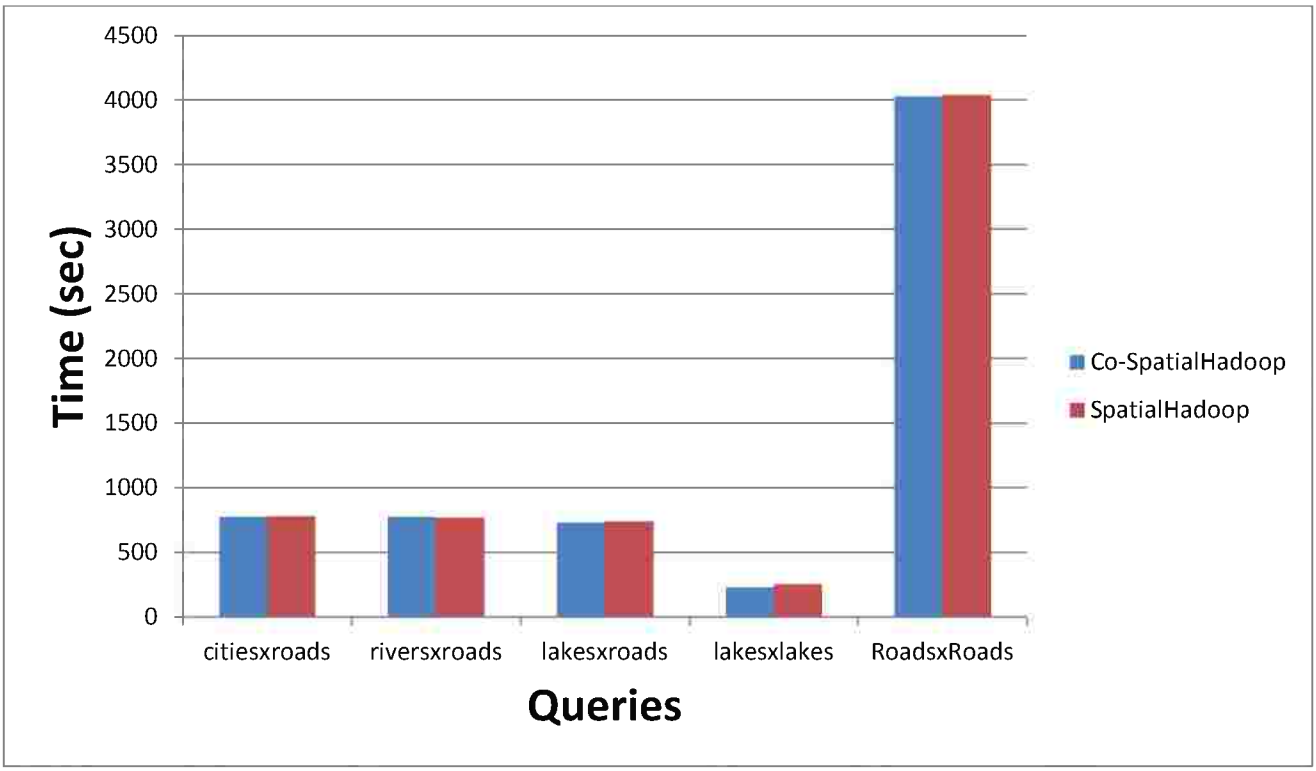

Figure 5.4: Response time chart of the experiment

5.4.LOCATOR TABLE PARTITIONING

We used R-Tree cells info of "roads" file to partition the locator table to five locators items, each locator items contains almost the same number of blocks. We use two approaches for this partitioning, the first one – which we use in the last section – has four vertical partitions and one horizontal, the second has five vertical partitions only. Figure 3.6 and Figure 3.8 shows these partitioning styles.

We ran the same experiment described in Section 4.3 for the partitioning styles shown in Figure 3.6 and Figure 3.8. Figure 4.5 shows the effect of the partitioning style that we use on network overhead for each query. We note that partitioning the file horizontally and vertically leads to better enhancement in the network overhead.

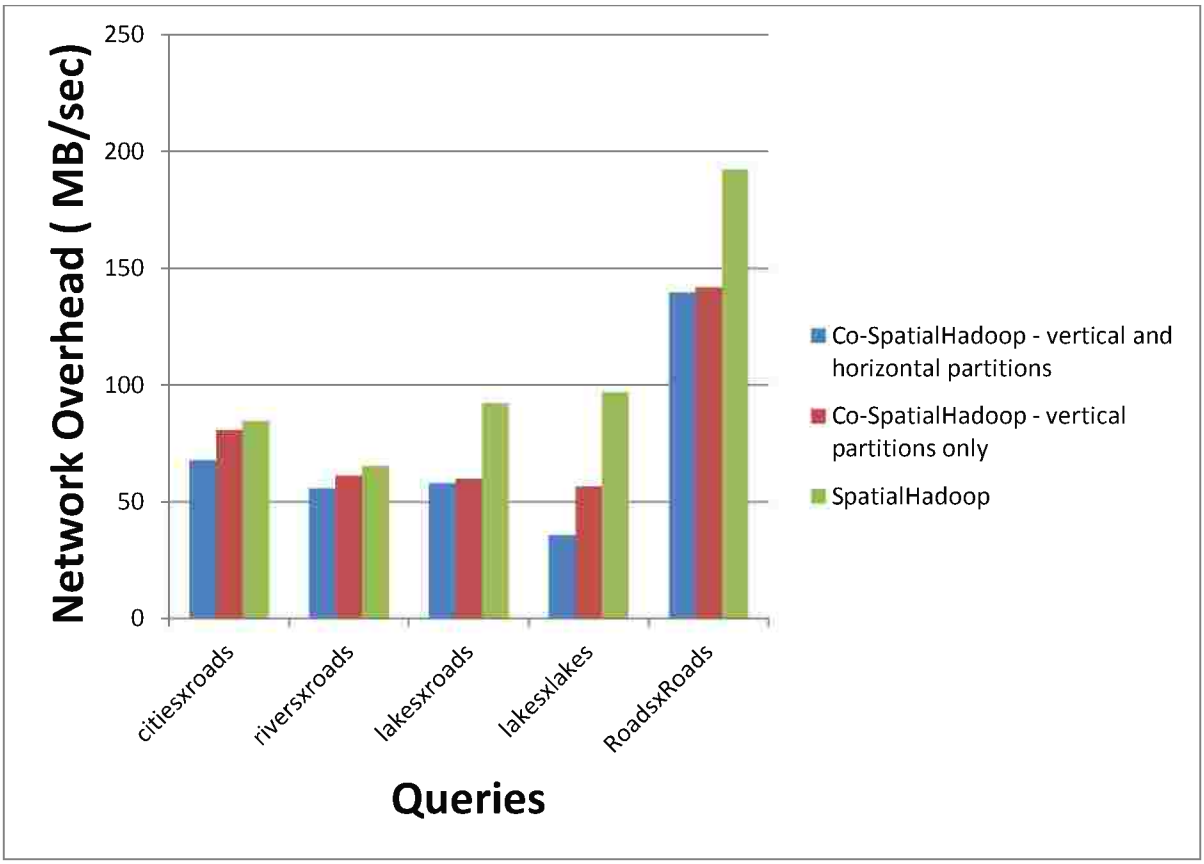

Figure 5.5: Network overhead comparison using two different approaches

5.5.NUMBER OF LOCATOR ITEMS

This experiment evaluates the effect of choosing the number of locator items on the network overhead incurred by the queries. We partitioned the locator table vertically and horizontally to six partitions then we use it to colocate other files as shows in Figure 3.7. Then, we executed queries in the workload to compare the performance of this locator table with the performance of locator table discussed in Section 4.3. Figure 4.6 compares performance between the basic SpatialHadoop, locator table with 5 items (partitioning style shown in Figure 3.6) and locator table with 6 items (partitioning style shown in Figure 3.7).

We note from the results shown in Figure 4.6 that the enhancement in network overhead is better when we have five locator items in the locator table. Except with "rivers

x roads" query, the locator table with k= 6 gave better enhancement than the other one, but the difference between them is not huge. Here also, the response time is almost the same.

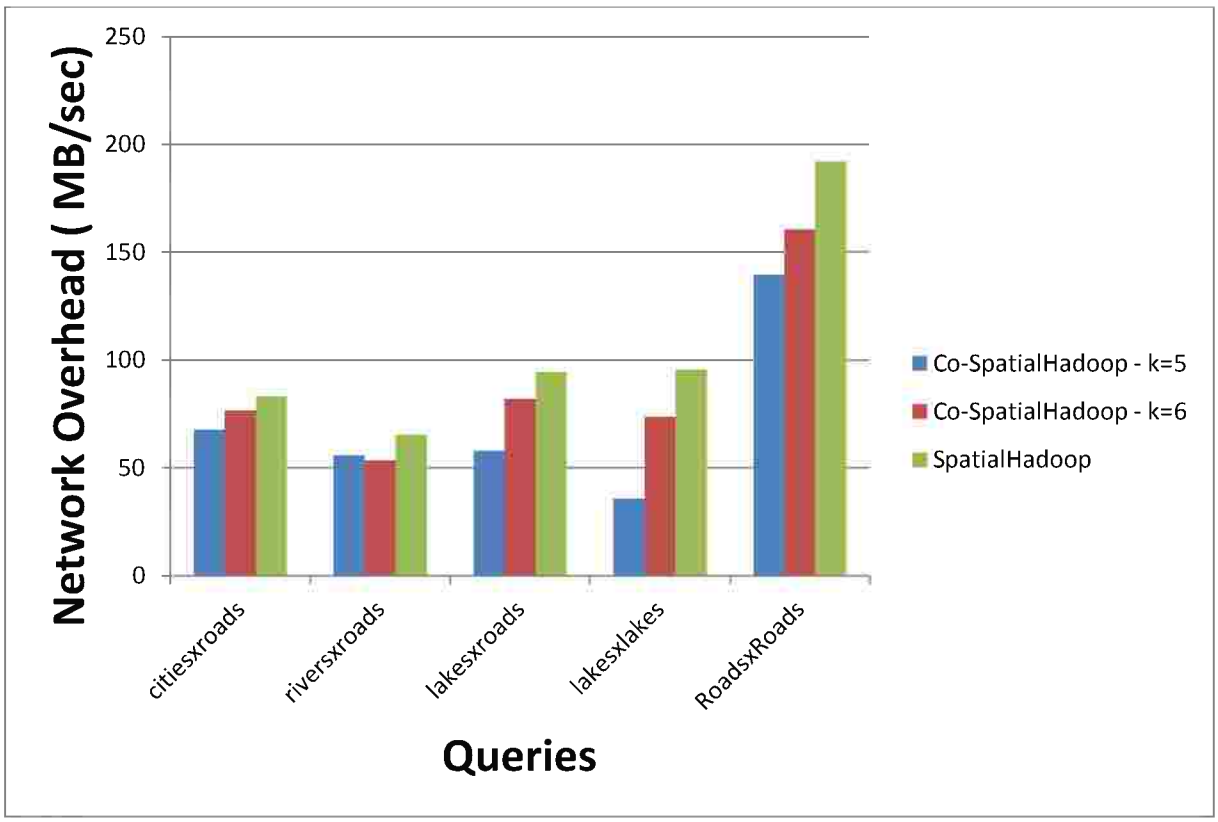

Figure 5.6: Network overhead comparison using different number of locator items

5.6.OVERHEAD OF COLOCATING SPATIAL DATA FILES

This experiment compares the execution time of the indexing operation of SpatialHadoop and the indexing operation of Co-SpatialHadoop. Figure 4.7 shows that the overhead incurred due to the colocation is insignificant.

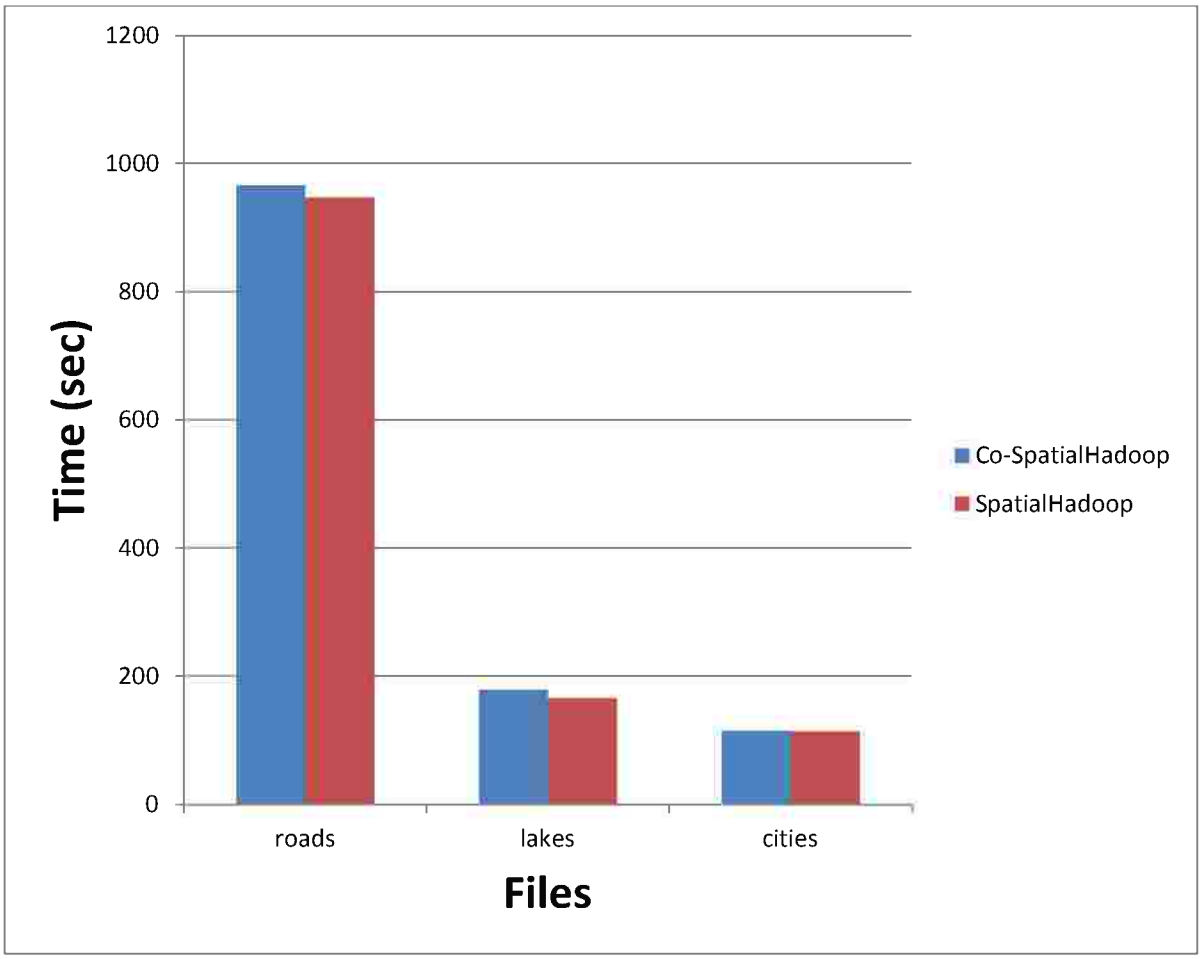

Figure 5.7: Comparison between SpatialHadoop and Co-SpatialHadoop indexing time

5.7.INDEXING NON-SPATIAL ATTRIBUTES

This experiment evaluates the performance of inverted index. We select various words from roads file with/without index. Each word has different occurrence over file's blocks. Section 4.2.5 refers to some details of the workload that we use in this experiment.

Figure 4.8 shows that the enhancement in response time of selection queries is increased when the occurrence of the selected word over file's blocks is decreased; that is because the scanned blocks of the selection operation are filtered by the inverted index and the blocks that contain the selected word are only scanned. Figure 4.9 shows the enhancement in network overhead obtained by inverted index. As the chart illustrates, the network overhead enhancement increases also inversely with word occurrence.

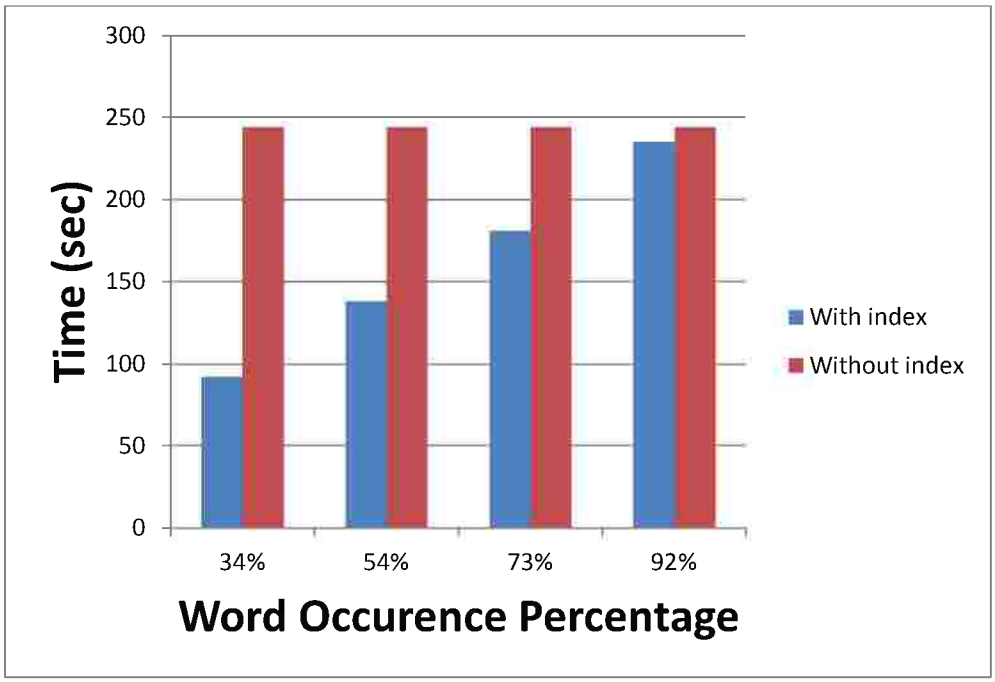

Figure 5.8: Response time comparison with/without inverted index

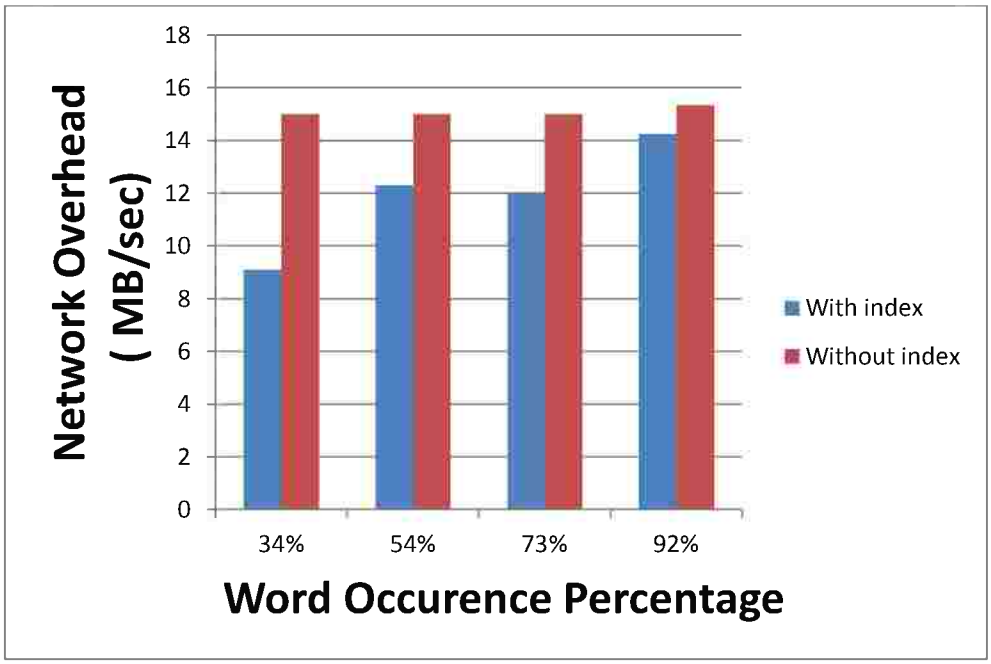

Figure 5.9: Network overhead comparison with/without inverted index

5.8.CONCLUSION

In this chapter, the results of the experiments compare the performance between Co-SpatialHadoop and SpatialHadoop [1]. It illustrates that Co-SpatialHadoop provides enhancement in network overhead.