ijecs open access www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 3 Issue 6 June, 2014 Page No. 6438-6441

Distributed Segment Tree Using MapReduce

Seyed Vahid Sanei Mehri[1], Ehsan Akhtarkavan[2], Saeed Erfanian[3]

[1] Department of Computer Engineering, Garmsar Branch,
Islamic Azad University, Garmsar, Iran
Email: Vahid.sanei@gmail.com

[2] Department of Computer Engineering, Garmsar Branch,
Islamic Azad University, Garmsar, Iran
Email: Akhtarkavan@iau-garmsar.ac.ir

[3] Department of Electrical Engineering, Garmsar Branch,
Islamic Azad University, Garmsar, Iran
Email: serfanian@iau-garmsar.ac.ir

ABSTRACT: ***In this paper we aim to propose an efficient method to implementing a distributed segment tree. For this purpose we use MapReduce which is a powerful tool in parallel data processing to divide the task among P separate processors. We will exploit parallel processing in order to decrease the time complexity of segment tree implementation and range queries.***

Keywords – ***Data Structure, Segment Tree, MapReduce, Parallel Processing***

1 INTRODUCTION

Segment tree is one of the most important data structures for facilitating rapid searching in a set of items, which is widely used in computational geometry, for instance in Klee's measure problem [1]. Segment tree is a full binary tree used to maintain intervals of a given set. Actually, each node of the tree maintains an interval of the given set. Using this innovative data structure, it is possible to query which nodes in the tree contain a desired range of set in logarithmic time. Every node of segment tree can be updated in logarithmic time, too. In this paper, we aim to implement a distributed segment tree using MapReduce model. We will take advantage of parallel processing in MapReduce in order to decrease the time complexity of segment tree implementation and range queries.

2 SEGMENT TREE

In this section, we aim to introduce the features of (undistributed) segment tree and review time complexity of segment tree implementation and range queries in the given set.

Segment tree is a rooted full binary tree segment tree with the following properties [2]:

1. The segment tree representing the range of length L (henceforth the range is called segment tree range) has a height $H = \log L + 1$.
2. Each node on a segment tree represents a node interval $[s_{l,k}, t_{l,k}], (l \in [0, \log L]$ and $k \in [0, 2^l - 1])$. Its length is $l_{l,k} = t_{l,k} - s_{l,k} + 1$. Clearly, the root node interval equals to the segment tree range and leaf node interval is one.
3. Each non-leaf node has two children. The left child and the right child represent the intervals $[s_{l,k}, \left\lfloor \frac{s_{l,k} + t_{l,k}}{2} \right\rfloor]$ and $[\left\lfloor \frac{s_{l,k} + t_{l,k}}{2} \right\rfloor + 1, t_{l,k}]$, respectively. The union of the two children covers the same interval as the parent does.
4. For neighboring nodes on the same layer, we have $s_{l,k} = t_{l,k-1} + 1$ for any $k \in [1, 2^l - 1]$. This property ensures the continuity of the segment tree.

5. All the nodes from the same layer span the whole segment tree range. That is, $\bigcup_{k=0}^{k=2^l-1}[s_{l,k}, t_{l,k}] = L$ for $l \in [0, \log L]$.

This property ensures the integrity of the segment tree.

An exemplar segment tree representing the range [0, 7] (i.e., L = 8) is depicted in Figure 1. We can easily verify all above properties.

Theorem 1. *Any segment with a range* $R, (R \le L)$, *can be represented by a union of some node intervals on the segment tree. There exist multiple possible unions for any range with* $R > 1$. Since the segment tree is a full binary tree, it is trivial to prove the first half of the theorem. For instance, the segment [2, 6] can be represented by the union of intervals [2, 3], [4, 5] and [6, 6], as shown in Figure 1. The second half of the theorem is also evident from the third property of segment tree. Although there are multiple possibilities to represent a larger range with unions of smaller subranges, the following theorem ensures the existence of the optimal representation.

Theorem 2. *Any segment with a range* $R, (R \le L)$, *can be expanded by a union of no more than* $2\log L$ *node intervals.*

Proof: Due to the space limitation, we only give a short intuitive proof. For a given segment S, suppose the longest part on S represented by a single node is P, then the left part to P should always be represented by the right children on segment tree, and the right part should be represented by the left children. There are at most $\log L$ consecutive left children on the tree and at most $\log L$ consecutive right children. So segment can be represented at most $2\log L$ nodes on the tree.

Therefore building a segment tree data structure can be done in $O(L \log L)$ and range queries in $O(\log L)$.

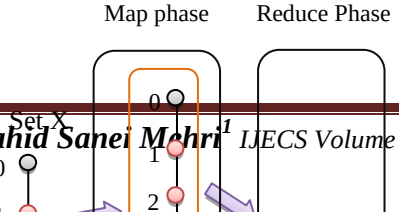

Figure 1. Illustration of a segment tree with a range [0,7] and the optimal representation of range [2,6].

3 MAPREDUCE

Nowadays, parallel data processing has become one of the most practical methods in the field of data processing. MapReduce which has been popularized by Google [3] is a modern approach for processing data which has a high fault tolerance.

In this approach, low-end machines can be used for data processing. MapReduce has valuable features such as scalability, simplicity and high fault tolerance, which make it a special and useful tool in both academic and industrial projects [4-7]. Programs which are written in MapReduce model are executed parallelly and automatically. The input data will be distributed among a number of machines on a cluster and then will be processed [8].

The MapReduce model consists of two primitive functions: *Map* and *Reduce*. The input for MapReduce is a list of *(key1, value1)* pairs and *Map()* is applied to each pair to compute intermediate key-value pairs, *(key2,value2).* The intermediate key-value pairs are then grouped together on the key equality basis, i.e. *(key2,list(value2))*. For each *key2*, *Reduce()*works on the list of all values, then produces zero or more aggregated results. Users can define the *Map()* and *Reduce()* functions however they want the MapReduce framework works [9].

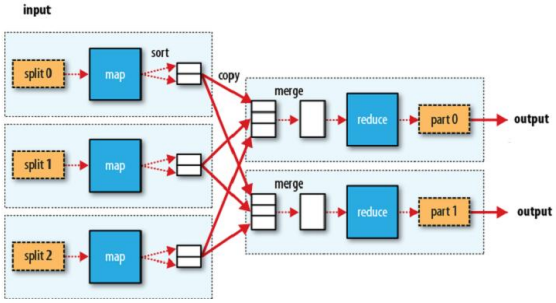

Figure 2. MapReduce data flow[10].

The data flow with three map and two reduce tasks is illustrated in Figure 2. The dotted boxes indicate nodes, the light arrows show data transfers on a node and the heavy arrows show data transfers between nodes . The goal of this paper is to implement a distributed segment tree with the help of MapReduce.

4 DISTRIBUTED SEGMENT TREE

A distributed solution is proposed to build segment tree using MapReduce to exploit parallel processing. We will represent the time complexity of the segment tree implementation and range queries are reduced using our proposed solution since the segment tree is distributed among P processors.

4.1 Distribtued segment tree implementation

Let us assume that set X contains n elements. X is partitioned among P processors where each processor contains at most $\left\lceil \frac{n}{P} \right\rceil$ consecutive elements of X. The set of elements maintained by processor i, is denoted by Xi. More formally, $\bigcup X_i = X, \bigcap X_i = \phi, |X_i| \le \left\lceil \frac{n}{P} \right\rceil (0 \le i < P)$. In the Map phase, each processor implements the segment tree of its own elements. Since the number of elements in each processor is at most $\left\lceil \frac{n}{P} \right\rceil$, implementing the segment tree in each processor can be done with $O(\frac{n}{P}\log\frac{n}{P})$ time complexity. In the other hand, in the Map phase, processors act parallelly, therefore implementing segment tree by all processors will be done in $O(\frac{n}{P}\log\frac{n}{P})$, too.

4.2 Range queries on the distributed segment tree

Since in each processor a segment tree maintains $\left\lceil \frac{n}{P} \right\rceil$ elements, finding a desired interval in the tree can be done in $O(\log\frac{n}{P})$. Besides, it is possible that two or more processors contain the desired interval, which increases the overall time complexity for finding the interval to $O(\log\frac{n}{P}+P)$.

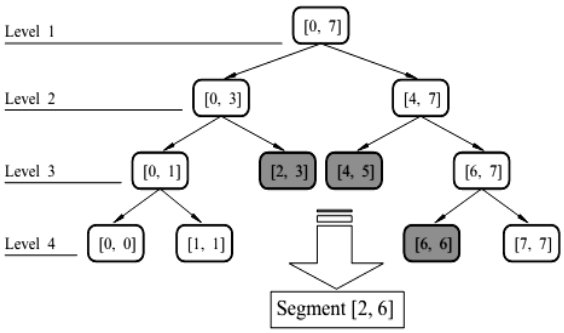

Figure 3. Range query on the distributed segment tree.

In Figure 3, which illustrates searching requested intervals in the distributed segment tree, a set containing 7 elements is given, and the interval [0,4] is requested. As processor 1 maintains interval [0,3] and processor 2 maintains interval [4,6], the obtained results from processors 1 and 2 are [0,3] and [4,4], respectively, which should be returned in the Map phase. In the Reduce phase, the obtained results from the Map phase are gathered and processed in one processor and then the final result is written into an output file.

5 EXPERIMENTS

We implemented distributed segment tree on a cluster of PCs (Intel Core i5 processor, 2GB RAM) on Hadoop-1.2.1[11] . The time complexity to implement distributed segment tree is illustrated in Figure 4. As shown, two, three and four processors as data nodes in the Hadoop cluster are used for processing the input data.

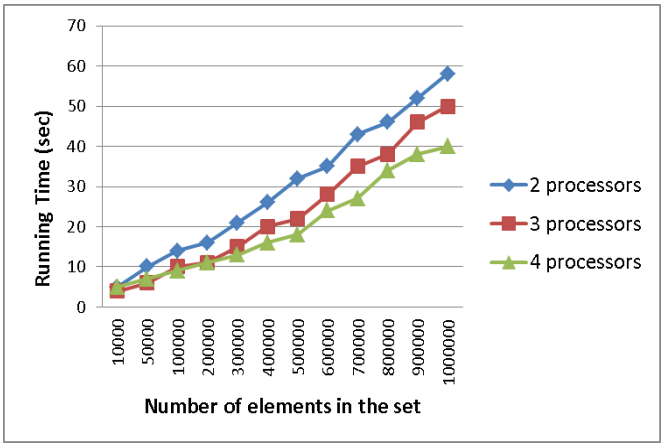

Figure 4. Comparision of running time of 2, 3 and 4 processors in hadoop cluster to implement distributed segment trees.

The obtained results indicate that an increase in the number of processors in the cluster will lead to a decrease in running time. Besides, more processors means less number of elements to be

maintained by each processor and hence a lower running time. The measured running times in Figure 4 are obtained by taking the mean of 50 running times of random input data.

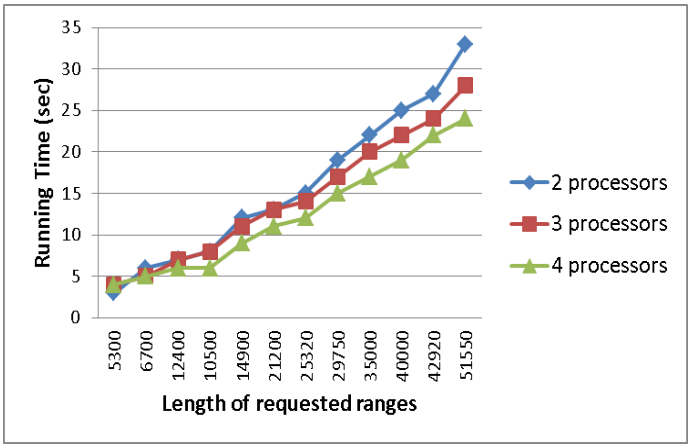

Figure 5. Comparision of running time of 2, 3 and 4 processors in the hadoop cluster to range query on the distributed segment trees.

The running time for range queries on a distributed segment tree is shown in Figure 5. The relation between the number of processors and range queries running time is the same as that for building segment trees explained earlier. The measured running times in Figure 5 are again obtained by taking the mean of 50 running times of random input data.

6 CONCLUSION

In this paper we provided a distributed method to implement segment tree using MapReduce. The described method splits the given set among processors in the Map phase and each processor implements the segment tree of its own elements. Since the number of elements in each processor is at most $\left\lceil \frac{n}{P} \right\rceil$, implementing the segment tree in each of them can be done with $O(\frac{n}{P}\log\frac{n}{P})$ time complexity. In the other hand, in the Map phase, processors act parallelly, therefore implementing segment tree by all processors is done in $O(\frac{n}{P}\log\frac{n}{P})$. Additionally, we presented finding a desired interval in the distributed segment tree can be done in $O(\log\frac{n}{P})$, too. Also, the obtained results in Figure 4 and Figure 5 have indicated that an increase in the number of processors in the cluster will lead to a decrease in running time.

REFERENCES

[1] Bentley, J.L., *Algorithms for Klee's rectangle problems*. 1977, Technical Report, Computer.

[2] Zheng, C., et al. *Distributed Segment Tree: Support of Range Query and Cover Query over DHT*. in *IPTPS*. 2006.

[3] Dean, J. and S. Ghemawat, *MapReduce: simplified data processing on large clusters.* Commun. ACM, 2008. **51**(1): p. 107-113.

[4] Deligiannis, P., H.-W. Loidl, and E. Kouidi, *Improving the diagnosis of mild hypertrophic cardiomyopathy with MapReduce*, in *Proceedings of third international workshop on MapReduce and its Applications Date*. 2012, ACM: Delft, The Netherlands. p. 41-48.

[5] Mantha, P.K., A. Luckow, and S. Jha, *Pilot-MapReduce: an extensible and flexible MapReduce implementation for distributed data*, in *Proceedings of third international workshop on MapReduce and its Applications Date*. 2012, ACM: Delft, The Netherlands. p. 17-24.

[6] Menon, R.K., G.P. Bhat, and M.C. Schatz, *Rapid parallel genome indexing with MapReduce*, in *Proceedings of the second international workshop on MapReduce and its applications*. 2011, ACM: San Jose, California, USA. p. 51-58.

[7] Qiu, J., *Generalizing mapreduce as a unified cloud and HPC runtime*, in *Proceedings of the 2nd international workshop on Petascal data analytics: challenges and opportunities*. 2011, ACM: Seattle, Washington, USA. p. 37-38.

[8] Mehri, S.V.S., et al., *Calculating the Area of the Union of Iso-oriented Rectangles Using MapReduce.* IJECS, 2014. **3**(4).

[9] Lee, K.-H., et al., *Parallel data processing with MapReduce: a survey.* AcM sIGMoD Record, 2012. **40**(4): p. 11-20.

[10] White, T., *Hadoop: The definitive guide*. 2012: " O'Reilly Media, Inc.".

[11] *Welcome to Apache™ Hadoop®!* ; Available from: http://hadoop.apache.org/.