ISSN 2320 2610
Volume 1, No.2, November - December 2012
International Journal of Multidisciplinary in Cryptology and Information Security
Available Online at http://warse.org/pdfs/ijmcis02122012.pdf

Leverage the BinRank's performance using HubRank and Parallel Computing

Dayananda P[1], Thnga Selvi A[2]
[1]Assistant Professor, Department of Information Science and Engg, MSRIT, Bangalore-54
[2] Department of Information Science and Engg, MSRIT, Bangalore-54, selvi2103@gmail.com

Abstract: Over the past decade, the amount of data generated and posted on the web has increased exponentially. High processing speeds, quick retrievals and efficient handling of data are of upmost importance. Searching on the web using a keyword and retrieving the relevant document has become an important and yet interesting task, these are the two major issues which should not be comprised. The issue addressed in this paper is would like to provide an approach which intends to provide an approximation to BinRank by integrating it with Hubrank and parallelize i.e execute the activities simultaneously to reduce query execution time and also increase the relevance of the results.

Keywords: BinRank, HubRank, ObjectRank.

INTRODUCTION

Dynamic authority –based keyword search algorithms like ObjectRank and Personalized PageRank (PPR), improves semantic link information to provide high recall searches in the web. Most of their algorithms perform iterative computation over the full graph. If the graph is too large, then such computation at query time becomes more complex and it is not feasible for computation. Then the concept of BinRank came into picture. BinRank is a system that approximates ObjectRank results by using a hybrid approach, in which a number of relatively small subsets of data graph are materialized. Any query was answered only by running ObjectRank on one of the subgraphs. This made BinRank achieve subsecond query execution time. The HubRank system presents a viable way to dynamically Personalize PageRank[1] at query time on ER graphs by utilizing clever hubset selection strategies and early termination bounds. HubRank is highly scalable for small graphs when compared to ObjectRank. Also since BinRank construction is precomputed offline, we plan to parallelize Bin construction activity and execute HubRank[4] on the subgraph that BinRank generates.

RELATED WORK

There are many existing ranking mechanisms. In this section we discuss some of the various ranking schemes. PageRank is a popular and simple algorithm used by Google's web search. It works as follows: it starts with a random Web surfer who starts at a random Web page and follows outgoing links with uniform probability. The biggest advantage of pageRank is its simplicity. But the disadvantage is that it returns only the documents that contain the keyword and the documents which may be more relevant to the search but does not contain the keyword are ignored. Dynamic versions of the PageRank algorithm like Personalized PageRank (PPR) for Web graph datasets, it is a modification of PageRank that performs search personalized on a base set that contains web pages that a user is interested in. But Personalized PageRank suffers from scalability.

The ObjectRank system applies the random walk model, the effectiveness of which is proven by Google's PageRank, to keyword search in databases modeled as labeled graphs. The system ranks the database. Objects with respect to the user-provided keywords. ObjectRank extends personalized PageRank(PPR) to perform keyword search in databases. ObjectRank uses a query term posting list as a set of random walk starting points and conducts the walk on the instance graph of the database. ObjectRank has successfully been applied to databases that have social networking components, such as bibliographic data and collaborative product design. ObjectRank suffers from the same scalability issues as personalized PageRank, as it requires multiple iterations over all nodes and links of the entire database graph. The original ObjectRank system has two modes: online and offline. The online mode runs the ranking algorithm once the query is received, which takes too long on large graphs. In the offline mode, ObjectRank precomputes top-k results for a query workload in advance. This precomputation is very expensive and requires a lot of storage space for precomputed results.

HubRank is a search system based on ObjectRank that improves the scalability of ObjectRank by combining the hub based approaches and monte Carlo approach[2]. It initially selects a fixed number of hub nodes by using a greedy hub selection algorithm that utilizes a query workload in order to minimize the query execution time.

HubRank is highly scalable for smaller graphs because of only fewer hubs are considered and early termination.

IMPLEMENTATION

The implementation procedure comprises of a generic algorithm[4] and a parallel computing procedure. The algorithm has two main phases first a pre-computation phase followed by a query processing phase.

Pre-computation phase

This phase happens in two steps. In the step1, we take as input the set of keywords in the entire database also called workload w and output as a set of term bins. Step2 takes the output of step1 as input and returns the set of materialized subgraphs MSG as output.

Bin construction

The bin construction algorithm packs terms into bins by partitioning workload w into a set of Bins composed of frequently co occurring terms. The algorithm takes a single parameter maxBinSize which limits the size of a Bin posting list i.e of all terms in the Bin. During the bin construction the bin identifies of each terms is inserted into the lucene index as an additional field. This allows us to identify the corresponding bin hence the MSG at query time for a given query.

MSG generation

BinRank uses ObjectRank algorithm to generate **Materialized SubGraph(**MSG) for each bin. Since HubRank algorithm is more scalable compare to ObjectRank algorithm for smaller graphs. We plan to use HubRank instead of ObjectRank to generate the MSG itself. We need to keep the size of the MSG being constructed as small as possible to achieve higher efficiency with regards to HubRank. For this purpose we plan to produce more number of Bins i.e MSG's , so that size of each Bin is smaller enough for HubRank algorithm to process efficiently. Since there are more number of Bins the query processing time might get delayed. To overcome this we also parallelize the MSG generation. Since each Bin and hence its MSG is independent of each other. MSG generation process is more suitable for parallel computing.

Fig 1 shows the stages in Pre computation phase.

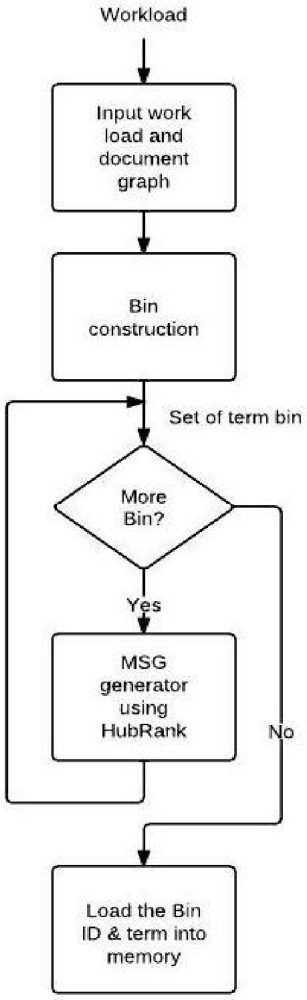

Fig 1: Block diagram of Pre-computation phase

Query processing phase

For a given keyword query we find the base set q and the bin identifier. With the above two information we determine the MSG on which the HubRank is to be applied to return the TopK results.

Multi keyword queries are processed by taking each individual keyword separately. For a union of keyword query we get the MSG for each individual keyword and run HubRank separately on each MSG to return the TopK relevant entries. Since parallel computing is an emerging technique these days we execute HubRank on each MSG's of a multikeyword query by running the HubRank algorithm on multiple cores simultaneously.

HubRank is a search system based on ObjectRank that improved the scalability of ObjectRank by combining the above two approaches. It first selects a fixed number of hub nodes by using a greedy hub selection algorithm that utilizes a query workload in order to minimize the query execution time.

Given a set of hub nodes H, it materializes the fingerprints of hub nodes in H. At query time, it generates an active subgraph by expanding the base set with its neighbors. It stops following a path when it encounters a hub node who's PPV was materialized, or the distance from the base set exceeds a fixed maximum length. The efficiency of query processing and the quality of query results are very sensitive to the size of H and the hub selection scheme. The dynamic pruning takes a key role in outperforming ObjectRank by a noticeable margin. The below diagram shows stages in query processing phase.

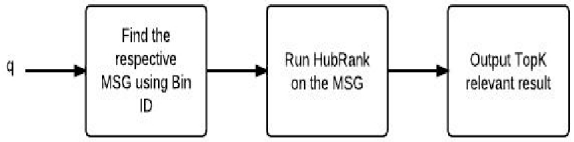

Fig 2: Block diagram of Query processing phase

Parallel computing approach

The advent of multicore CPUs and manycore GPUs means that mainstream processor chips are now parallel systems. The GPU, as a specialized processor, addresses the demands of real-time high-resolution 3D graphics compute-intensive tasks. As of 2012 GPUs have evolved into highly parallel mul ti core systems allowing very efficient manipulation of large blocks of data. This design is more effective than general-purpose CPUs for algorithms where processing of large blocks of data is done in parallel.

In this context let us understand how we intend to utilize parallel computing .To achieve our goal we propose the use of Single Instruction, Multiple Data (SIMD) architecture, computers have several processors that follow the same set of instructions, but each processor inputs different data into those instructions. This can be useful for analyzing large chunks of data based on the same criteria. Many complex computational problems don't fit this model. Coincidently in our algorithm especially the precomputation phase which requires us to build sub graphs based on keywords. if the same algorithm is implemented using the SIMD concept on a parallel computing device, the speed of precomputation increases by a marked value.

The Compute Unified Device Architecture(CUDA) programming model provides a straightforward means of describing inherently parallel computations, and nvidia's tesla gpu architecture delivers high computational throughput on massively parallel problems.

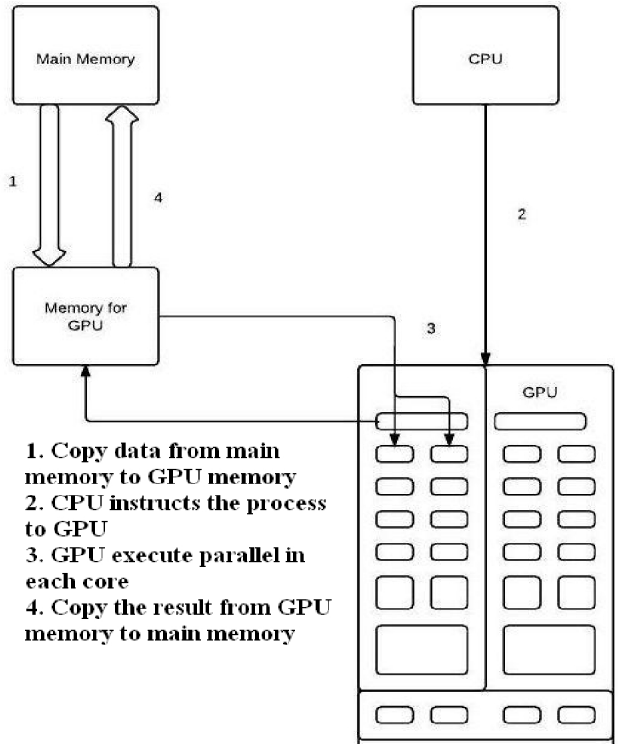

Fig 3: Block diagram of process flow of CUDA

As shown in the Fig 3 above, all the processors in the GPU will execute the same logic but using a different data instance. For each bin of terms a materialiased subgraph has to be constructed. For each individual bin, a separate processor will build a materialised sub graph. Though there is investment in hardware initially, it will be traded off for the speed of execution. This parallelizing concept eventually will decrease the overall time required to execute the algorithm.

Let us consider a statistical computation of the algorithm's execution time and compare the same with its parallel computation[6]. If a parallel program is executed on a computer having p processors, the least possible execution time will be equal to the sequential time divided by number of processors

T_p is the parallel execution time, T_p is sequential execution time and p be the no of processors in the computer, then

$$S = \frac{Ts}{Tp} \qquad (1)$$

A measure called speedup value which is ratio of sequential time and parallel execution time. The maximum speedup value could be achieved in an ideal multiprocessor system where there are no communication costs and the workload of processors is balanced. In such a system, every processor needs Ts/p time units in order to complete its job so the speedup value will be as the following:

Let Speedup value be S

$$s = Ts \Big/ \left(\frac{Ts}{p}\right) \qquad (2)$$

This leads to S=P, Previous statistics claim that for a wikepedia data set, pre computing about a thousand subgraphs, takes about 12 hours on a single CPU [4].The same if implemented using parallel computing with SIMD architecture will ideally take, Ts=12 hours, according to the formula previously discussed Tp=Ts/p, let us futher consider the no of processors as 4 then Tp will be approximately 3 hours to build the same 1000 subgraphs.

In general, if Ts is the sequential time take for implementing one subgraph then when done in parallel using n processors, the total time taken to construct x subgraphs will be, Considering ideally:

$$Tp = \frac{Ts}{n} \qquad (3)$$

T_p is time taken to build 1 subgraph in parallel.

$$Tt = x * Tp = x * Ts/n \qquad (4)$$

The above formula holds good in ideal conditions only but According to the Amdahl law, it is very difficult, even into an ideal parallel system, to obtain a speedup value equal with the number of processors because each program, in terms of running time. The total time would become

$$Tt = (((1 - \alpha) * x * Ts)/n) + \beta \qquad (5)$$

Tt is the total time taken to build subgraphs, α is the fraction of code which has to executed sequentially and (1-α) is a part of code which require to build subgraph done parallel and also considering the internal tradeoff β .

$$Tn = \alpha * Ts + Tt \qquad (6)$$

Ts will be total time taken to execute the pre computation phase when done parallely.

CONCLUSION

In this paper, we proposed an approach to increases the performance of BinRank using HubRank and parallelize i.e. execute the creation of bins simultaneously to reduce query execution time and also increase the relevance of the results. To further enhance this work by providing the threat detection system to some extent, by storing the potential illegal keywords in database and ensure that the search is not on these words by checking the database before submitting the query for search.

REFERENCES

[1] D. Fogaras, B. Ra´cz, K. Csaloga´ny, and T. Sarlo´ s,"Towards Scaling Fully Personalized PageRank: Algorithms, Lower Bounds, and Experiments," Internet Math., vol. 2, no. 3, pp. 333-358, 2005.

[2] K. Avrachenkov, N. Litvak, D. Nemirovsky, and N. Osipova, "Monte Carlo Methods in PageRank Computation: When One Iteration Is Sufficient," SIAM J. Numerical Analysis, vol. 45, no. 2, pp. 890-904,2007.

[3] A. Balmin, V. Hristidis, and Y. Papakonstantinou,"ObjectRank: Authority-Based Keyword Search in Databases," Proc. Int'l Conf. Very Large Data Bases (VLDB),2004.

[4] Heasoo Hwang; Balmin, A.; Reinwald, B.; Nijkamp, E.; , "BinRank: Scaling Dynamic Authority-Based Search Using Materialized Subgraphs," Knowledge and Data Engineering, IEEE Transactions on , vol.22, no.8, pp.1176-1190, Aug. 2010 .

[5] S. Chakrabarti, "Dynamic Personalized PageRank in Entity-Relation Graphs," Proc. Int'l World Wide Web Conf.(WWW), 2007.

[6] Felician ALECU, "performance analysis of parallel algorithms", Journal of Applied quantitative methods, volume-2, issue-1,2007.

[7] V. Hristidis, H. Hwang, and Y. Papakonstantinou, "Authority-Based Keyword Search in Databases," ACM Trans. Database Systems, vol. 33, no. 1, pp. 1-40, 2008.

[8] M.R. Garey and D.S. Johnson, "A 71/60 Theorem for Bin Packing," J. Complexity, vol. 1, pp. 65-106, 1985.

[9] H. Hwang, A. Balmin, H. Pirahesh, and B. Reinwald, "Information Discovery in Loosely Integrated Data," Proc. ACM SIGMOD, 2007.

[10] J. Cho and U. Schonfeld, "Rankmass Crawler: A Crawler with High PageRank Coverage Guarantee," Proc. Int'l Conf. Very Large Data Bases (VLDB), 2007.