International Journal of Soft Computing and Engineering (IJSCE)
ISSN: 2231-2307, Volume-3, Issue-5, November 2013

A short survey of Advantages and Applications of Skip Graphs

Shalini Batra, Amritpal Singh

Abstract— ***Although the enormous growth of Internet, Social Network, Cloud Computing, etc. has brought the world closer and faster, major concern for computer experts is how to store such enormous amount of data especially in form of graphs. Further, data structure used for storage of such type of data should provide efficient format for fast retrieval of data as and when required. Although adjacency matrix is an effective technique to represent a graph having few or large number of nodes and vertices but when it comes to analysis of huge amount of data from site likes like face book or twitter, adjacency matrix cannot do this. This paper provides a special kind of data structure, skip graph which can be efficiently used for storing such type of data resulting in optimal storage, space utilization and retrieval.***

Index Terms—Advanced Data Structure, Skip List, Skip Graph, Efficient and fast search.

I. INTRODUCTION

1.1 Skip List

A skip list [3] is an ordered data structure based on a succession of linked lists with geometrically decreasing numbers of items. The deterministic versions of skip list have guaranteed properties where as randomized skip lists only offer high probability performance. This height (Hn) is the maximum length of a search path for any key from the top of the skip list. Devroye has proved that this height Hn is of order log n [10].

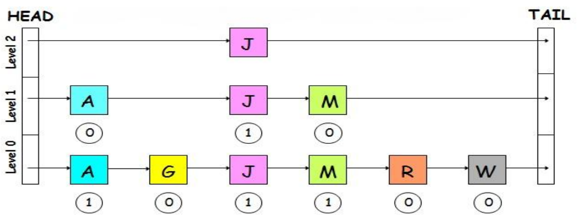

Figure 1 - Example of Skip List[13]

1.2 Skip Graph

The skip graph, introduced by Aspnes and Shah in [2, 4], is a variant of the skip list, designed to perform better in a distributed environment. In a skip graph, the whole data structure can be distributed among a large number of nodes, and the structure provides good load balancing and fault tolerance properties.

Manuscript received November, 2013.

Dr. Shalini Batra received M.E. Degree from BITS, Pilani, Rajasthan India and Ph.D. Degree from Thapar University in area of semantics in 2012. She is currently working as Assistant Professor with Computer Science Department at Thapar University, Punjab, India.

Amritpal Singh, He is presently working as Assistant Professor at Chandigarh University Gharuan . His research interests include Machine Learning and Big Data. Received M.E. Degree from Thapar University, Punjab, India.

As defined by author in [6] Skip graphs are data structures with similar functionality to binary trees or skip lists, permitting efficient insertion, removal and searches among elements, but they are best suitable for P2P distributed environments. Skip Graphs are composed of tower of increasingly refined linked lists in various levels, each one with no head and doubly linked. shown in fig from [6].

Skip graphs provide the full functionality of a balanced tree in a distributed system where elements are stored in separate nodes that may fall at any time as described in [3]. They are designed for use in searching peer-to-peer networks, and by providing the ability to perform queries based on key ordering, they improve on existing search tools that provide only hash table functionality.

As per analysis done by James and Shah in [2, 3] on skip lists or other tree data structures, skip graphs are highly resilient, tolerating a large fraction of failed nodes without losing connectivity. In addition, constructing, inserting new elements into, searching a skip graph and detecting and repairing errors in the data structure introduced by node failures can be done using simple and straightforward algorithms. During past years interesting variants of skip graphs have been studied, like skip nets [7], skip webs [1] or rainbow skip graphs [6].

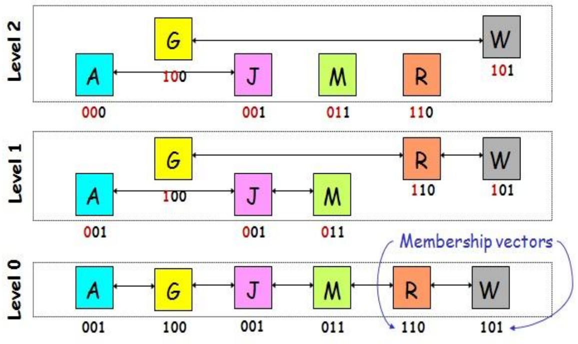

Figure 2 - Example of Skip Graph [13]

II.MODELS AND NOTATIONS

In a skip graph, each node represents a resource to be searched where node *x* holds two fields: the first is a key, which is arbitrary and may be the resource name. Nodes are ordered according to their keys. For notational convenience the keys are the considered to be integers 1, 2, . . . , *n*. Since the keys have no function in the construction other than to provide an ordering and a target for searches there is no loss of generality. The second field is a membership vector *m(x)* which is for convenience treated as an infinite string of random bits chosen independently by each node. In practice, it is enough to generate an *O(*log *n)*-bit prefix of this string with overwhelming probability. The nodes are ordered lexicographically by their keys in a circular doubly-linked list

*S*_ so that node *i* is connected to *i* −1 mod *n* and *i* +1 mod *n*. For each finite bit-vector *σ*, an additional circular doubly-linked list *Sσ* is constructed by taking all nodes whose membership vectors have *σ* as a prefix, and linking adjacent nodes in the lexicographic key order.

2.2 Algorithms for Skip Graph.

The search operation:

The search operation is identical to the search in a skip list with only minor adaptations to run in a distributed system. The search is started at the topmost level of the node seeking a key and it proceeds along each level without overshooting the key, continuing at a lower level if required, until it reaches level 0. Either the address of the node storing the search key, if it exists, or the address of the node storing the largest key less than (or the smallest key greater than) the search key is returned. In Figure 3 we show steps required to search Alphabet "W" in given Skip Graph.

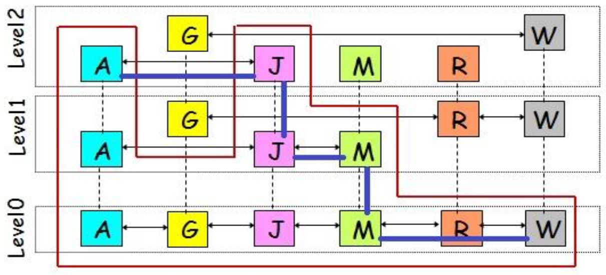

Figure 3. Steps to search "W" node

The insert operation

A new node 'u' knows some introducing node 'v' in the network that will help it to join the network. Node 'u' inserts itself in one linked list at each level till it finds itself in a singleton list at the topmost level. The insert operation consists of two stages:

(1) Node 'u' starts a search for itself from 'v' to find its neighbours at level 0, and links to them.

(2) Node 'u' finds the closest nodes 's' and 'y' at each level W _ 0, s < u < y, such that

m(u) _ (W + 1) = m(s) & (W + 1) = m(y) _ (W + 1), if they exist, and links to them at level W + 1.

Because each existing node 'v' does not require m(v)(W+1) unless there exists another node 'u' such that m(v) _ (W + 1) = m(u) _ (W + 1), it can delay determining its value until a new node arrives asking for its value; thus at any given time only a finite prefix of the membership vector of any node needs to be generated.

In Figure 4 (a) new node is inserted into given example of Skip graph and Figure 4(b) shows required updates after insertion [13].

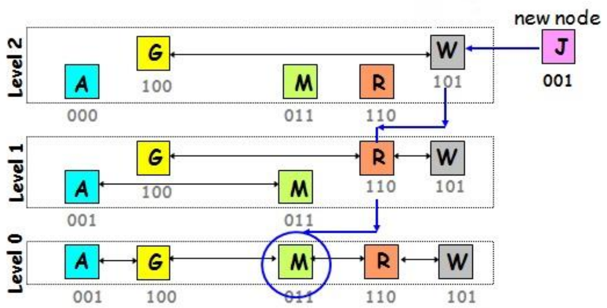

Figure 4(a). Steps required to Insert "J" node

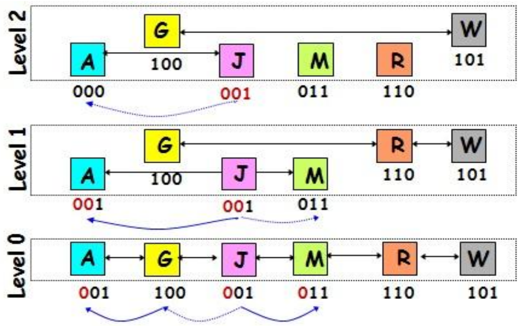

Figure 4(b). Skip graph after Insertion

The delete operation

When node 'u' wants to leave the network, it deletes itself in parallel from all lists above level 0 and then deletes itself from level 0.

III. BENEFITS OF SKIP GRAPHS

Correctness under concurrency

As discussed in section 2, both insertion and deletion can be comfortably done on skip graph and search operations eventually find their target node or correctly report that it is not present in the skip graph. So any search operation can be linearized with respect to insertion and deletion. In effect, the skip graph inherits the atomicity properties of its bottom layer, with upper layers serving only to provide increased efficiency.

Fault Tolerance

James and Udi describe some of the fault tolerance properties of a skip graph [4]. Fault tolerance of related data structures, such as augmented versions of linked lists and binary trees, has been well-studied by Munro and Poblete [11]. The main question is how many nodes can be separated from the primary component by the failure of other nodes, as this determines the size of the surviving skip graph after the repair mechanism finishes. It has been clearly proved that even a worst-case choice of failures by an adversary can do only limited damage to the structure of the skip graph. With high probability, a skip graph with n nodes has an tQ(1/logn) expansion ratio, implying that at most *O(f* log n) nodes can be separated .

Random failures

For random failures, the situation appears even more promising, experimental results presented in [4,7,9] show that for a reasonably large skip graph nearly all nodes remain in the primary component until about two-thirds of the nodes fail, and that it is possible to make searches highly resilient to failure even without using the repair mechanism by use of redundant links.

Fast Search and Fault Tolerance

The average search in skip graph involves only O(logn) nodes that most searches succeed as long as the proportion of failed nodes is substantially less than O(logn) [1,8,9] . By detecting failures locally and using additional redundant edges, one can make searches highly tolerant to small numbers of random faults. In general, results cannot make as strong guarantees as those provided by data structures based on explicit use of expanders [6,7], but this is compensated for by the simplicity of skip graphs and the existence of good distributed mechanisms for constructing and repairing them.

Load balancing

In addition to fault-tolerance, a skip graph provides a limited form of load balancing, by smoothing out hot spots caused by popular search targets. The guarantees that a skip graph makes in this case are similar to the guarantees made for survivability. Just as an element stored at a particular node will not survive the loss of that node or its neighbours in the graph, many searches directed at a particular element will lead to high load on the node that stores it and on nodes likely to be on a search path. However, James has shown that this effect drops off rapidly with distance elements that are far away from a popular target in the bottom-level list produce little additional load on average [4]. Further author has provided two characterizations of this result. The first shows that the probability that a particular search uses a node between the source and target drops off inversely with the distance from the node to the target. This fact is not necessarily reassuring to heavily-loaded nodes. Since the probability averages over all choices of membership vectors, it may be that some particularly unlucky node finds itself with a membership vector that puts it on nearly every search path to some very popular target. Second characterization addresses load balancing issue by showing that most of the load-spreading effects are the result of assuming a random membership vector for the source of the search.

Low hitting times

Random walks on expanders done in [7] have the property of hitting a large set of nodes fast and with high probability. This can be used for a variety of applications such as load balancing, gathering statistics on the nodes of the skip graph and for finding highly replicated

IV. SUCCESSFUL IMPLEMENTATION OF SKIP GRAPH IN DIFFERENT APPLICATIONS

- Due to the rise in popularity of peer-to-peer systems dynamic overlay networks have recently received a lot of attention [5]. An overlay network is a logical network formed by its participants over a wired or wireless routing network. The number of computers and users in such a network may reach millions. Therefore, research in this area has focused on improving scalability and efficiency of overlay networks. Two usual optimization properties are the speed of searching for items in the network and the speed of topology updates. Another popular parameter is expansion.
- In open peer-to-peer systems [participants may frequently enter and leave the overlay network either voluntarily or due to failures [5,9]. In a peer-to-peer system of large scale, faults and inconsistencies are the norm rather than as an exception. Moreover, interaction of such faults may leave the system in an unpredictable, possibly system-wide failure. Hence, overlay networks require mechanisms that continuously counter such disturbances. Paper [5] shows the successful implementation of Skip Graph in such problem.
- Hammurabi , Cristina shows that Skip graphs show better results in multithreaded multiprocessor or distributed systems , where many entities access the shared pool concurrently [6]. In such cases manual control of concurrency is required in all operations, and use of Skip graph show good results.
- The results shown by James , Wieder represent that high probability in a skip graph provide quality and more connectivity in expanding networks [7]. So these properties help them to construct unstructured P2P system, making it a good candidate for a hybrid P2P system.
- Cloud computing has evolved as a popular computing environment [10]. In data centric applications hosted on the cloud, data is accessed and updated in a purely distributed manner. The distributed data structures used for dynamic storage of the data for such applications require two fundamental qualities, authentication and persistence, which are not completely met by existing distributed data structures. Authentication is a crucial requirement for data structures used on the cloud, as users need to be convinced about the validity of the data they receive. Moreover the data structure has to be persistent, so that changes can be made to the data structure without losing old data, or old versions of the data structure, which may be required by different users in the distributed environment. In [10] authors have shown that use of Skip graph in all basic models perform better from many existing models.
- The problems of Web Services where main issue is to support the reuse and interoperation of software components on the web are receiving ever increasing interests from e-commerce, science, and research communities across different areas. A fundamental problem of Web Services is service discovery. Web Services discovery is the process of finding a Web Service that is capable to deliver a particular service, or integrating several Web Services in order to achieve a particular goal by means like IR (Information Retrieval). In all alternatives Skip graph shows the better results for this problem [12].
- Skip graph shows the properties of Fast queries and updates and they also support for ordered data [8]. These feature allows for a richer set of queries than a simple dictionary that can only answer membership queries, including those arising in DNA databases, location-based services, and prefix searches for file names or data titles and helpful in communication by fast message exchange.

V. CONCLUSIONS AND FUTURE SCOPE

A short survey of skip graph provided in this paper, clearly indicate the usage and advantages of using skip graphs in various distributed and graph based applications. Since skip graphs provide the full functionality of a balanced tree in a distributed system they can be designed for use in searching peer-to-peer networks, and by providing the ability to perform queries based on key ordering, they improve on existing search tools that provide only hash table functionality. There are still many unexplored areas where skip graphs can fine many useful applications and one such domain is Social Network Analysis (SNA). Skip graphs can be used to store the data in graphs and cluster the data and above all retrieval will be very efficient and fast.

REFERENCES

[1] L. Arge, D. Eppstein, and M.T. Goodrich. Skip-webs: efficient distributed data structures for multi-dimensional data sets. *Proceedings of the annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, pages 69–76, 2009.

[2] J. Aspnes and G. Shah. Skip graphs. *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 384–393, 2003.

[3] James Aspnes and Gauri Shah. Skip graphs. *ACM Transactions on Algorithms*, 3(4):37, November 2007.

[4] James Aspnes and Udi Wieder. The expansion and mixing time of skip graphs with applications. In *SPAA '05: Proceedings of the seventeenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 126–134, New York, NY, USA, 2005. ACM.

[5] Thomas Clouser, Mikhail Nesterenko, Christian Scheideler : Tiara: A self-stabilizing deterministic skip list and skip graph . 2012 Elsevier

[6] Hammurabi Mendes , Cristina G. Fernandes - A Concurrent Implementation of Skip graphs . Electronic Notes in Discrete Mathematics 35 (2009) page no .-263-268 .

[7] James Aspnes , Udi Wieder -The expansion and mixing time of skip graphs with applications. page no 385-394 , Springer-Verlag 2008

[8] Michael T. Goodrich, Michael J. Nelson , Jonathan Z. Sun -The Rainbow Skip Graph: A Fault-Tolerant Constant-Degree P2P Relay Structure . ArXiv - 2009

[9] Fuminori Makikawa, Tatsuhiro Tsuchiya, Tohru Kikuno - Balance and Proximity-Aware Skip Graph Construction. 2010 First International Conference on Networking and Computing .

[10] Shabeera T P, Priya Chandran, Madhu Kumar S D - Authenticated and Persistent Skip Graph: A Data Structure for Cloud Based Data-Centric Applications . CHENNAI, India , 2012 , ACM

[11] Ian Munro and Patricio V. Poblete. Fault tolerance and storage reduction in binary search trees. *Information and Control*, 62(2/3):210-218, August 1984.

[12] Jianjun Yu, Hao Su, Gang Zhou, Ke Xu - SNet: Skip Graph based Semantic Web Services Discovery . Seoul, Korea. 2007 ACM

[13] James Aspnes, Guari Shah, ppt in SODA 2003." http://www.cs.yale.edu/homes/aspnes/papers/skip-graphs-soda03.ppt"

Dr. Shalini Batra received M.E. Degree from BITS, Pilani, Rajasthan India and Ph.D. Degree from Thapar university in area of semantics in 2012. She is currently working as Assistant Professor with Computer Science Department at Thapar University, Punjab, India. She has guided 30 M. E. and currently guiding 4 Ph. D. students. She has more than 50 publications in National and International conferences and journals. Her research interest includes Machine Learning, Semantics, Big data and Social Networks.

Amritpal Singh, received M.E. Degree from Thapar University , Punjab, India , with a minor in Big Data and advanced data Sturtures , in 2013. He is presently working as Assistant Professor at Chandigarh University Gharuan . His research interests include Machine Learning and Big Data.