ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 12, December 2014

Efficient Hashing Algorithm for Midsquare

Nitisha Rajgure[1], Dr Vilas Thakare[2]

Assistant Professor, Dept. of Computer Science, Sipna COET, SGBAU, Amravati University, Maharastra, India[1]

Professor & Head, Dept. of Computer Science, SGBAU, Amravati University, Maharashtra, India[2]

ABSTRACT: Hashing function is the One of the most frequent way for finding the nearest match in the large data sets. From the last decades number of researcher has been work on the hashing and focusing on the better approach than the existing one with respective their performance. In this paper presents a survey on different type of hash functions, different type of hashing method, hashing strategies and structural weakness of them or the limitation of them that in which kind of problem they are suitable and what they can't be used., also we are investigating the alternative approach for the mid- square hashing approach. The necessary data structures and algorithms are described, the expected performance is analyzed mathematically, and actual execution times are obtained and compared with alternative techniques. It shows that it provides the faster response time. Finally our method is intuitive and easy to implement.

KEYWORDS: Hashing, algorithmic, access methods, data structures.

I. Introduction

The address computation can be implemented in several ways in the hash table , here we discuss its basic operations and typical hash function operations and the problem for which hash table are suitable and for which hash table are not suitable. Focus on the some common hashing methods, folding methods and other hashing method Mid square, Digit Addition, Analysis which are used in different application of Hashing. *The necessary data structures and algorithms are described, the expected performance is analyzed mathematically, and actual execution times are obtained and compared with alternative techniques.*

Hash defines "to hash" as "to chop up, as of potatoes". This is exactly what hash functions usually do. A good hash function creates disorder and, in this way, avoids collisions.

II. Objectives

Finding the address of element from the hash table is one of the important task for performance of any algorithm. This function can be done by Hash-fictions.

The some existing algorithms is complicated to implement and the constant of the space bound is large. With focus of time and space bound ,We also study our opportunistic data structure in a dynamic setting and devise a variant achieving effective search and update time bounds. The focus objective of this study is to reduce the computational time of the Hashing function. Here we propose new algorithm as NKRSQR Hashing function with the following objectives.

NKRSQR Algorithm should be easy to implement and easy to Understand.
NKRSQR Algorithm Must Return the result with minimum time complexity.

III.Related work

Schmidt and Siegel [1] proposed the first algorithm for constructing a MPHF with constant evaluation time and description size O(n + log log u) bits..From a practical point of view, the algorithm of Schmidt and Siegel is not attractive.

The scheme is complicated to implement and the constant of the space bound is large. Ji, Jianqiu;Li, Jianmin et al [2] introduce an effective method, i.e. the min-max hash method, which significantly reduces the hashing time by half, yet

10.15680/ijircce.2014.0212038

it has a provably slightly smaller variance in estimating pair wise Jaccard similarity. In addition, the estimator of min-max hash only contains pair wise equality checking, thus it is especially suitable for approximate nearest neighbor search. Experiments show that with the same length of hash code, min-max has reduces the hashing time to half as much as that of min-wise hash. The divide-and-concatenate technique cannot speed up software implementations but can only improve the collision probability beyond that provided by the processor architecture. This is because, if a processor supports w-bit additions and multiplications in one or two cycles, then w/2-bit operations will also consume the same number of cycles as w-bit operations.[2].Abutaha, M.Hamamreh, R.[3] Introduce The new one way hash algorithm designed by using two steps. Firstly, convert the input data into matrix system by using all necessary conversions to generate the initial hash value. Secondly, use the output of the first step to make a digest for these data and finally generate the secure hash value. Joseph Gil and Yossi Matias implements the simple fast parallel hashing by oblivious Execution. This algorithm was design with bucket approach in data structures . Experimental result presents a simple fast and efficient parallel algorithm for the hashing problem Using n processors and the running time of the algorithm is O(lg lg n). More recently, Hagerup and Tholey [4] have come up with the best theoretical result we know of. The MPHF obtained can be evaluated in O(1) time and stored in n log e + log log u + O(n(log log n)2/ log n + log log log u) bits. The construction time is O(n+log log u) using O(n) words of space. Again, the terms involving u are negligible. In spite of its theoretical importance, the Hagerup and Tholey [4] algorithm is also not practical, as it emphasizes asymptotic space complexity only. (It is also very complicated to implement. Hashing-based approximate nearest neighbor (ANN) search in huge databases has become popular due to its computational and memory efficiency[5].

IV. PROPOSED ALGORITHM

From the literature review we can conclude that the well deep knowledge of different data structure allows us to implement and design new algorithm which can fallow the basic principal of root data structure. The two principal criteria use in selecting a hash function;

- It should be very easy and quick to compute.
- Function should, return value with minimum number of collisions.

and the next section also describes how address can be computed from the keys using the hash function. Several variation on the basic theme are presented and analyzed; Focused line is that hashing is extremely effective and practical technique. Question is that How easy is it to design a hash function? It is quite easy , as a little number theory will help us prove.

A. *Working Principal :*

NKRSQR is based on the Divide and merge technique. This technique consist of decomposing the instance and merge as individual element for problem solving.

Two questions that naturally spring to mind are:

1.Why would any one want to do this?
2.How should we solve ?

First we partition the number ‘n’ in ‘Two’ slots by taking the remainder of ‘n’.
That is the first step is computing as –

$V_1(n)= n \mod 10$--------------------eq(1)
$V_2(n)= n / 10$--------------------------eq(2)

In second step apply the quadratic equation to find the square of number. Next Step can be done by extracting k bits from the middle of the square of the key.

$H(n)= (v_1+v_2)^2 >>(w-n)$----------------eq(3)

Where w is bits of size

10.15680/ijircce.2014.0212038

NKRSQR follows nature of recursive algorithm. This Algorithm does a pretty good job as compare to middle –square method when the integer valued keys are equi-probable.

B. *Address Space for Result Number:*

The middle of square (or Mid- Square for short) function , ***fm*** is computed by squaring the identifiers and then using an appropriate number of bits from middle of the square to obtain the bucket address.Since the middle bit of the square will usually depend upon all of the characters' in the identifier, it is expected that different identifiers would result in different hash addresses with high probability even when some of the characters in the identifiers are the same.

The number of bits to be used to obtain the bucket address depend on the table size . if n bits are used to compute hash address, the range of value is 2^n , so the size of hash table is chosen to be a power of 2 when this kind of scheme is used .

MSHA(A)= n number of middle digits of (A2)

Where,

MSHA=Mid Square Hash Address

V. NKRSQR Advantages and Disadvantages

As earlier we note the phrase of universe , now we will discusses the some advantages and disadvantages of NKRSQR Hash function, This algorithm run faster than the existing one. However this is limited upto number 99 only.

VI. NKRSQR Results

The proposed energy efficient algorithm is implemented with basic programming of Algorithm. We tested output result with numbers from 1 to 99 for different size and required time . Proposed algorithm is compared with the basic traditional Mid-square hash function for address mapping . We considered the time and memory space , and processor execution time is calculated through the CPUTIME function of the program . proposed algorithm provides better complexity with respective time and space which is a need of computer era for speedup the process and maximum storage.

The result Fig. 1 shows the Traditional Vs New(NKRSQR) approach for min time. In the same way Fig 2 shows Traditional Vs New(NKRSQR) approach for Max time. Table 1. shows the Result Using Traditional Method of Time and memory usage while Table 2 shows Result Using (NKRSQR) Method , which defines the better execution time

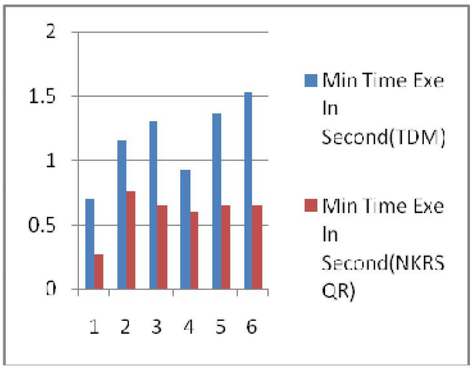

Fig 1:Traditional Vs New(NKRSQR) approach for min time

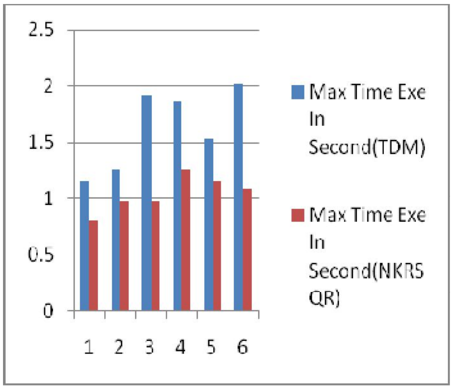

Fig 2:Traditional Vs New(NKRSQR) approach for Max Time

10.15680/ijircce.2014.0212038

The results shows in Fig 1 of Traditional Vs New(NKRSQR) approach for min time ,that the proposed algorithm performs better with the minimum running time of this implementation is much less than the the traditional Mid-square hash function.

The results shows in Fig 2 of Traditional Vs New(NKRSQR) approach for Max time ,that the proposed algorithm performs better with the Maximum running time of this implementation is much less than the the traditional Mid-square hash function.

Table 1: ResultUsing Traditional Method

Input Number	Category	Min Time Exe In Second	Max Time Exe In Second	RAM Memory Usege(Min)	RAM Memory Usege (Max)
1	Best	0.7	1.15	2.324K	2.328K
11	Best	1.15	1.26	2.324K	2.328K
50	AVG	1.31	1.92	2.230K	2.316K
55	AVG	0.93	1.86	2.324K	2.328K
98	Worst	1.37	1.53	2.324K	2.328K
99	Worst	1.53	2.03	2.324K	2.328K

The Table 1 shows the Minimum and maximum time required time for Traditional approach of hash squring method .

Table 2: Result Using (NKRSQR) Method

Input Number	Category	Min Time Exe In Second	Max Time Exe In Second	RAM Memory Usage(Min)	RAM Memory Usage (Max)
1	Best	0.27	0.8	2.316k	2.324K
11	Best	0.76	0.98	2.320k	2.324K
50	AVG	0.65	0.98	2.320k	2.328K
55	AVG	0.60	1.26	2.320k	2.324K
98	Worst	0.65	1.15	2.320k	2.328K
99	Worst	0.65	1.09	2.320k	2.324K

The Table 2 shows the Minimum and maximum time required time for NKRSQR approach of hash squring method .

VII. Conclusion and Future Work

The results showed that the proposed algorithm performs better with the running time of this implementation is much less than the the traditional Mid- square hash function. The proposed algorithm provides better complexity with respective time and space which is a need of computer era for speedup the process and maximum storage. As the performance of the proposed algorithm is analyzed with only traditional Mid- square approach in hashing functions in future with some modifications in design considerations the performance of the proposed algorithm can be compared with other efficient algorithm. NKRSQR is the simplest hash function which is obtain by combination of two different method that is Division hash function and squaring hash function.

10.15680/ijircce.2014.0212038

REFERENCES

1. Singh, M.; Garg, D., "Choosing Best Hashing Strategies and Hash Functions," Advance Computing Conference, 2009. IACC 2009. IEEE International , vol., no., pp.50,55, 6-7 March 2009 doi: 10.1109/IADCC.2009.4808979
2. Bo Yang; Karri, R.; McGrew, D.A., "Divide-and-concatenate: an architecture-level optimization technique for universal hash functions," Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on , vol.24, no.11, pp.1740,1747, Nov. 2005 doi: 10.1109/TCAD.2005.852455
3. Abutaha, M.; Hamamreh, R., "New one way hash algorithm using non-invertible matrix," Computer Medical Applications (ICCMA), 2013 International Conference on , vol., no., pp.1,5, 20-22 Jan. 2013 doi: 10.1109/ICCMA.2013.6506174
4. T. Hagerup and T. Tholey. Efficient minimal perfect hashing in nearly minimal space. In Proc. of the 18th Symposium on Theoretical Aspects of Computer Science (STACS'01), pages 317–326. Springer LNCS vol. 2010, 2001.
5. Jun Wang; Kumar, S.; Shih-Fu Chang, "Semi-Supervised Hashing for Large-Scale Search," Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.34, no.12, pp.2393,2406, Dec. 2012 doi: 10.1109/TPAMI.2012.48.
6. Kollios, G.; Tsotras, V.J., "Hashing methods for temporal data," Knowledge and Data Engineering, IEEE Transactions on , vol.14, no.4, pp.902,919, Jul/Aug 2002 doi: 10.1109/TKDE.2002.1019221
7. Jae-Pil Heo; Youngwoon Lee; Junfeng He; Shih-Fu Chang; Sung-Eui Yoon, "Spherical hashing," Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on , vol., no., pp.2957,2964, 16-21 June 2012 doi: 10.1109/CVPR.2012.6248024
8. Feng Yue; Bin Li; Ming Yu; Jiaqiang Wang, "Hashing Based Fast Palmprint Identification for Large-Scale Databases," Information Forensics and Security, IEEE Transactions on , vol.8, no.5, pp.769,778, May 2013 doi: 10.1109/TIFS.2013.2253321
9. Ye-In Chang; Chien-I Lee; Wann-Bay ChangLiaw, "Linear spiral hashing for expansible files," Knowledge and Data Engineering, IEEE Transactions on , vol.11, no.6, pp.969,984, Nov/Dec 1999 doi: 10.1109/69.824617.
10. Minho Jin; Yoo, C.D., "Quantum Hashing for Multimedia," Information Forensics and Security, IEEE Transactions on , vol.4, no.4, pp.982,994, Dec. 2009 doi: 10.1109/TIFS.2009.2033221
11. Yuenan Li; Zheming Lu; Ce Zhu; XiaMu Niu, "Robust Image Hashing Based on Random Gabor Filtering and Dithered Lattice Vector Quantization," Image Processing, IEEE Transactions on , vol.21, no.4, pp.1963,1980, April 2012 doi: 10.1109/TIP.2011.2171698
12. Lamberger, M.; Pramstaller, N.; Rechberger, C.; Rijmen, V., "Analysis of the Hash Function Design Strategy Called SMASH," Information Theory, IEEE Transactions on , vol.54, no.8, pp.3647,3655, Aug. 2008 doi: 10.1109/TIT.2008.926420
13. Gordon, D.M.; Miller, V.S.; Ostapenko, P., "Optimal Hash Functions for Approximate Matches on the -Cube," Information Theory, IEEE Transactions on , vol.56, no.3, pp.984,991, March 2010 doi: 10.1109/TIT.2009.2039037.
14. Jin, Z.; Li, C.; Lin, Y.; Cai, D., "Density Sensitive Hashing," Cybernetics, IEEE Transactions on , vol.PP, no.99, pp.1,1, 0 doi: 10.1109/TCYB.2013.2283497.

BIOGRAPHY

Nitisha Rajgure is a Research Assistant in the Computer science Department, College of SIPNA COET, SGBAU university .She received Master of Computer Enginnering in 2010 from SGBAU Amravati, MS, India. She He has published more than 25 papers in various National & Inter national Conferences & papers in various International journals.Her research interests are Algorithms, Data structure and Image processing etc.

Dr. V.M.Thakare was born in Wani, Maharashtra in 1962. He was worked as Assistant Professor for 10 Years at Professor Ram Meghe Institute of Technology & Research, Badnera and P.G.Department of Computer Science, S.G .B. Amravati University, Amravati. Currently he is working as Professor & Head in Computer Science and Faculty of Engineering & Technology, Post Graduate Department of Computer Science, SGB Amravati University, Amravati.He has published more than 80 papers in various National & Inter national Conferences & 20 papers in various International journals. He is working on various bodies of Universities as a chairman & members. He has guided around 300 more students at M.E / MTech, MCAM.S & Mphil level. He is a research guide for Ph.D. at S.G.B. Amravati University, Amravati. His interests of research is in Computer Architecture, Artificial Intelligence, Robotics, Database and Data warehousing & mining