European Journal of Molecular & Clinical Medicine ISSN 2515-8260
Volume 8, Issue 2, 2021

MEMORY OVERLAY BASED HYBRID LRU ALGORITHM FOR ENTERPRISE DATA HUB

Dr. Murugan A – Associate Professor and Ganesan S – Research Scholar,
PG & Research Dept. of Computer Science, Dr. Ambedkar Govt. Arts College,
University of Madras, Chennai, India

Abstract: In general computing theory, Array is one of the data structures to hold a group of same data type elements. Static array holds the underlying elements in an initial allocated memory space. In contrast, dynamic array grows the memory allocation automatically and dynamically based on the insertion request volume. Our research work is to leverage dynamic memory overlay array mechanism in Enterprise Data Hub. It is continuation of our earlier work on Hybrid LRU Algorithm, which depicts about the experimental advantage of execution time optimization and efficient page/cache hit ratio, using hybrid Least Recently Used algorithm with priority mechanism. With the ability to change the allocated space dynamically using memory overlay concept, Dynamic array improves the storage scalability and faster execution.

Keywords: Memory Overlay, Dynamic Array, Big Data, Enterprise Data Hub, Cache Algorithm, Hybrid LRU (Least Recently Used), Execution Complexity, Membound, etc.

I. INTRODUCTION

In the rapid transformation of Information Technology, enterprise data growth [15] is phenomenal. Enterprise Data Hub [1] is a solution to build and maintain the golden records of any enterprise as shared trustable enterprise data. Most Big Data solutions [4] are based on disk based data processing and in-memory [16] on need basis. During the earlier research work, hybrid LRU algorithm [9] was developed with the parametrized data elements. Ultimately, it was depicting about the efficient way of in-memory caching [10] to handle the data in the most effective logic.

As the next improvement step, the research work is shifted from data storage/processing to data structure [23]. In this paper, the big data is processed with highly dynamic data structure – Dynamic Array. By extending the previous hybrid LRU [1] with the flexible and scalable Dynamic Array [3], this research work depicts the final efficiencies of Enterprise Data Hub system in the industry.

II. LITERATURE REVIEW

A. Industry Data Growth

In the software industry, the rate of data growth is phenomenal. The law of exponential data growth is seen since the internet technologies evolved in mid 1990s. To prove this statement, the internet world statistics recorded 16 million internet users in 1995; now it is 4,700 million users.

Global Big Data Analytics Market [15] predicated US$ 37.34 billion value in 2018; expected target of US$ 105.08 billion by 2027 with CAGR of 12.3%. Related data is formed from every web click and social event across the world.

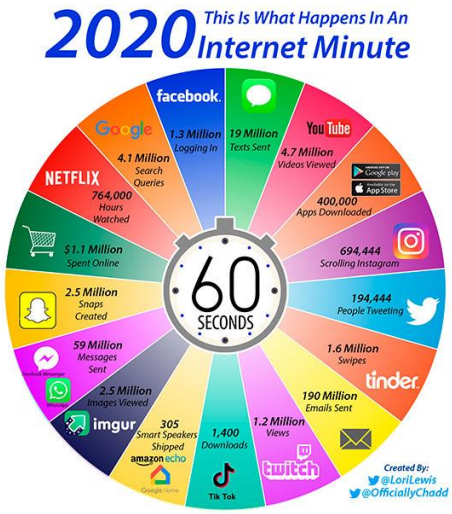

Fig. 1. One minute usage in Internet 2020

Today, 2.5 quintillion bytes of data [2] are generated every day. It is depicted in an internet minute usage during 2020.

B. Enterprise Data Hub

In the recent decade, industry has few emerging disruptive technologies like big data, cloud computing, etc. Enterprise Data Hub (EDH) is the proposed solution which is a central data repository [5] for any enterprise. It can be built using open-sourced Big Data tools [6] and techniques like, Hadoop Distribution File System, Map Reduce, Hive, etc. Ultimately, the system produces the master golden data as single source of the truth for any enterprise.

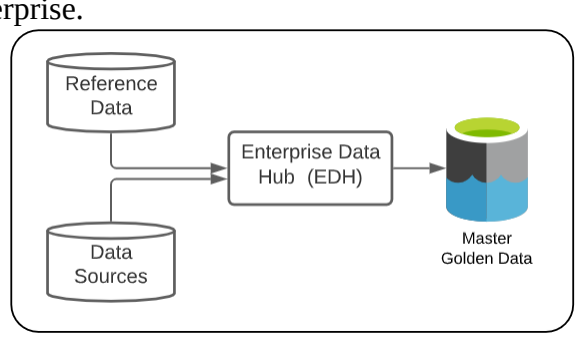

Fig. 2. Enterprise Data Hub Model

. Technology is expected to add the business value. EDH model generates master data store for any report dimensions at any point of time. In terms of credibility, data integrity and governance plays a vital role in EDH.

C. Related Researches

In computer architecture, virtual memory [20, 24] is one of the most significant inventions to innovate the new design. Core operating system [21, 26] contains the functions such as memory capacity management, inter-process protection, and data sharing. This dynamic memory [14, 25] also enables the simple implementation of several techniques to improve the execution performance [28] like copy-on-write, page wipping, etc. With the key success factory of its power and elegant solution, it is used in almost all modern high performance systems.

By design, each virtual page [16] can be mapped to both a physical page and an overlay. Memory overlay contains only a subset of cache lines from the virtual page, and cache lines that are present in the overlay. Dynamic data structure [1] plays a vital role in memory overlay [3] design.

Last year, the earlier research work was to develop Hybrid LRU [1] with static data structure model.

Primary research of this paper is to experiment [3, 16] the improvement of Hybrid LRU using dynamic array data structure in memory overlay.

III. HYBRID ALGORITHM

Hybrid LRU algorithm provides an efficient algorithm to handle enterprise data hub using prioritized LRU design with efficient cache hierarchy [10].

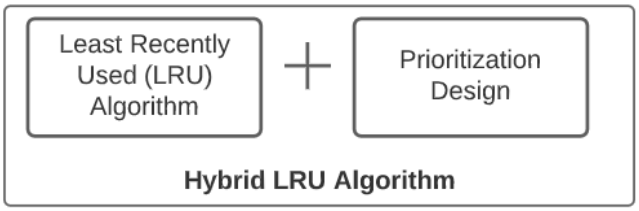

Fig. 3. Hybrid LRU Design

This approach was to use replacement [8] policies (cache algorithms), which will choose the element to remove from the cache when we need space for a new element.

A. Experimentation Summary

On comparison of it with existing LRU algorithm, hybrid LRU enables to get and prove two key benefits. One is a significant (~30%) decrease of the execution time to extract data from cache store during object cache extraction process. As the second benefit, it gets an efficient cache hit ratio about ~10% higher than traditional LRU algorithm.

B. Experimentation Results

Execution time is measured by the time taken to fetch the required element from the cache store [9] either cache hit or miss. Experimentation was validated with data set of 10, 150, 1300 and 10500 between Traditional and Hybrid Least Recently Used algorithms. It was evident that Hybrid version was better with the execution time.

Cache hit ratio is mathematically calculated by dividend as the number of successful requests served by the cache and divisor as the number of received requests. Experimented results varied between 99 and 87 for data volume of 10 and 10,500 respectively using Traditional Least Recently Used algorithm. At the same time, Hybrid Least Recently Used algorithm optimized between 99.8 and 94 for the same data set. As per the industry standard, Cache Hit Ratio is calculated using the below formula:

$$Cache\ Hit\ Ratio = \frac{number\ of\ cache\ hits}{(number\ of\ cache\ hits + number\ of\ cache\ misses)}$$

IV. MEMORY OVERLAY

A. Overlays in Memory Management

In memory management, fixed partitioning (like static array) is limited by the maximum size of the partition. This constraint never be span over another computing process. Overlays [3] concept is to load the needed memory as a part, not complete program.

B. Dynamic Array - Differentiation

Dynamic array implements the concept of overlays in memory management. Data structure theory defines array as an arrangement of information in one or more dimensions.

There are two types of array storage namely static and dynamic. Static array elements are often stored

contiguously in the computer's memory. The size of static array memory to be allocated, is known to the compiler as predefined case.

Dynamic array memory is allocated during execution of instructions. Heap is meant for a pile of memory space available to programmers to allocate and de-allocate during run time. On programming the improper memory handling, the underlying application will lead to the memory leak and the subsequent system crashes.

In computer memory management [10], static array is persisted in Stack, whereas dynamic array in Heap memory.

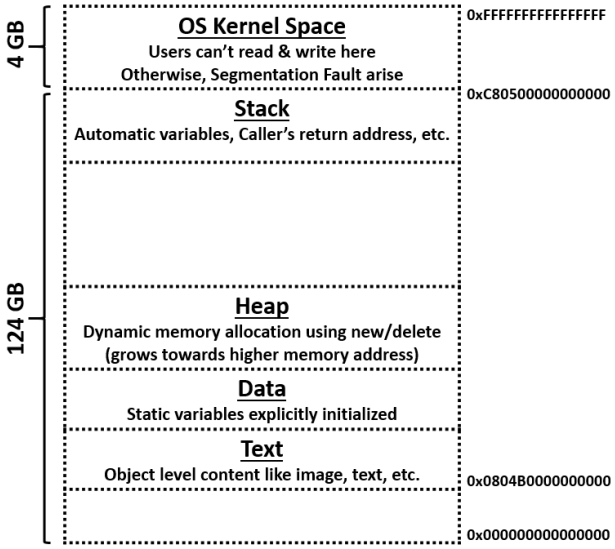

Fig. 4. Stack and Heap Storage

C. Dynamic Array - Internals

By design, Dynamic arrays are more flexible/dynamic during the execution of a program. Static array structure can't do much, if the array is filled during program execution. However, Dynamic array [1] can keep pushing values into the array at runtime.

Initially, dynamic array is constructed with the fixed size array to store the starting elements contiguously in the system heap. Rest of the unused allocation is reserved at the end of the underlying dynamic array. During execution, the new elements are added in a constant time slot by leveraging the reserved space until free space is completely consumed.

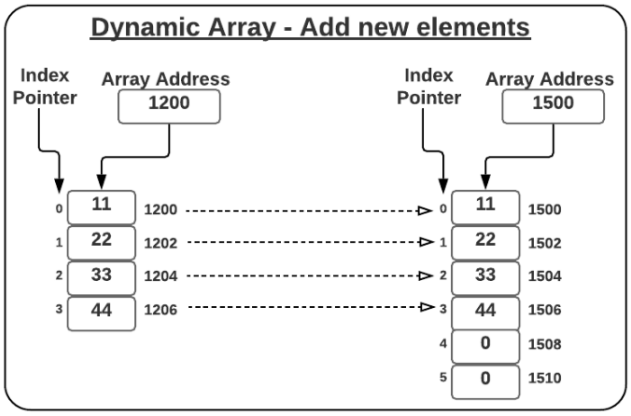

Fig. 5. Dynamic Array Data Insertion Process

Let us assume that dynamic array completes the consumption of the available free space in heap storage. On adding a new element with this criterion, the underlying fixed-sized array needs to be increased in size as drawn above. Flip side of this approach is expensive to allocate new space and move over the elements.

D. Dynamic Array - Complexity

Order of execution time of an algorithm quantifies the amount of time taken by the algorithm to run as a function of the length of the input. Here is the tabular data to depict:

Table I: Dynamic Array – Order of Execution

#	Operation Parameter	Time Complexity
1	Indexing	*O (1)*
2	Insert at top	*O (N)*
3	Insert at bottom	*O (1) – not full* *O (N) – full array*
4	Read	*O (1)*
5	Delete	*O (N)*

N: queue size

V. TECHNICAL IMPLEMENTATION

A. System Logic

In the earlier paper, Enterprise Data Hub (EDH) system was constructed using Hybrid LRU with the parameterized priority queue logic. Here, the research work is extended to improve from the traditional array structure to dynamic overlay array model.

B. Algorithm Used

Proposed pseudo code to implement dynamic array for Enterprise Data Hub

---

Algorithm: Dynamic overlay for Enterprise Data Hub
Input: NYSE stock data
Output: Faster retrieval of large data set

```
begin
  let MemBound is the constant amount at which the memory stack is increased
  let Top is the pointer to top of the memory stack
  let Length denotes the size of the allocated memory stack
  funtion Type[] CreateNew(Type[] existMemStack)
  begin
    Create new memory stack as newMemStack
    Allocate newMemStack with size of (Length + MemBound)
    Copy the content from existMemStack to newMemStack
    Resize the value of Length variable by adding MemBound
 return newMemStack
end
function Type[] Push(Type[] existMemStack, Type insertElement)
begin
  if (Top is equal to Length)
  begin
    existMemStack = CreateNew(existMemStack)
  end
  Insert element at Top of the memory stack
  return existMemStack
end
function Pop(Type[] existMemStack)
begin
  Subract one level of Top pointer
end
function Type Fetch(Type[] existMemStack, int Index)
begin
  return existMemStack[Index]
end

function Display(Type[] existMemStack)
begin
  if Top or Length is Zero then
  begin

      Log the error message as "Empty Memory"
  end
  else
  begin
    for every index in existMemStack
      Print the element in current index
  end
end
```

The above algorithm explains the implementation details for an automatic resizing of the underlying storage. In this algorithm, Fetch operation retrieves the required element at a given index within *O(1)* execution cycle. It provides the faster lookups of the given data collection. CreateNew operation adds a new element [11] in the existing collection, if it fits within existMemStack. If Top pointer exceeds MemBound threshold, newMemStack will be created to hold the updated data set of the collection.

VI. EXPERIMENTATION RESULT

A. System Environment

Research work is simulated in 3.1 GHz Intel Xeon E5-2687W processor having 16 (sixteen) core processors, 128 GB RAM, 1406 MB cache memory. The array elements are set to off64_t type (8 Bytes) with the number of dimensions (n), which is varied from 4 to 12.

Data source is downloaded with multiple dimensions like Ticker Date, Open price, High price in a day, Low price, Closing price, Trade Volume, etc. from NYSE historical data site at http://eoddata.com/download.aspx

B. Discussion of the result

During the recent analysis of the industry research [3, 5, 16] and proposal, it is clearly demonstrated the potential advantages of managing memory to process the large amount of data.

In the last paper [1], Hybrid LRU algorithm proved the experimental advantage of execution time optimization and efficient page/cache hit ratio. Now, the research work is shifted from data storage [4] and processing to data structure [5, 7]. In this paper, the big data is processed with dynamic array [24] data structure in the financial market data.

The base work is based on the experimentation results of Hybrid LRU, which leveraged the prioritization concept on Least Recently Used algorithm. The denominator is marked as Static Structure [1] in the experimented result table.

Numerator work is designed with the dynamic array structure [4] against the base research on Hybrid LRU. Enterprise Data Hub persists their data as multidimensional arrays [14] optimized for spatial querying with the elasticity factor. The rate of improvement with the gaps in static structured Hybrid LRU is researched here to find out the system optimization degree [16] in the tabular columns.

Experimentations of the result, are measured and summarized in two key factors:

- Storage – Scalability performance
- Speed – Retrieval execution time

In the following sections, the experimental results are elaborated to demonstrate storage [17] and speed advantages with the underlying parameters, factors, result data, etc.

C. Storage Scalability

Storage utilization is measured between the available data storage space and actual usage storage in an enterprise. Storage Scalability [2] is how much capacity the storage system can address, manage and support with acceptable performance, in spite of the lesser physical storage [13].

Data source is experimented between static and dynamic structure with the different set of dimension size.

Table II: Experimented Result – Storage Scalability

Storage File Size (GB)	**Number of Dimensions**				
	4	**6**	**8**	**10**	**12**
Static Structure [1]	100	100	100	100	100
Dynamic Overlay	1310	1280	1265	1190	732

Data points are graphically represented as below:

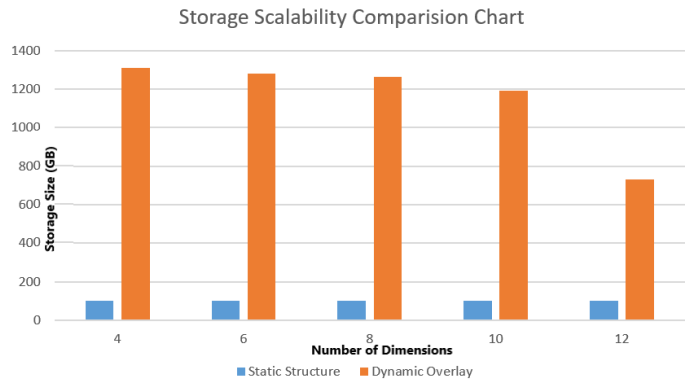

Fig. 6. Storage Scalability Comparison

The concept of dynamic overlay, is to build the dimensional data structure to persist in a contiguous memory region by overlaying the holding data. This attribute enables 10 times of improvement on storage scalability benefits.

D. Execution Time Optimization

Optimized execution time is one of the key factors for any algorithm. On applying the dynamic [14] overlay data structure in hybrid LRU for Enterprise Data Hub, our algorithm produces below result in terms of data retrieval time in milliseconds. Program is applied to both static and dynamic overlay data structure against the different set of data dimensions, sampled from NYSE (New York Stock Exchange) historical trading data.

Table III: Experimented Result – Data Retrieval Time

Retrieval Time (milliseconds)	Number of Dimensions				
	4	**6**	**8**	**10**	**12**
Static Structure [1]	96	113	141	187	507
Dynamic Overlay	12	13	15	29	44

Data points are graphically represented as below:

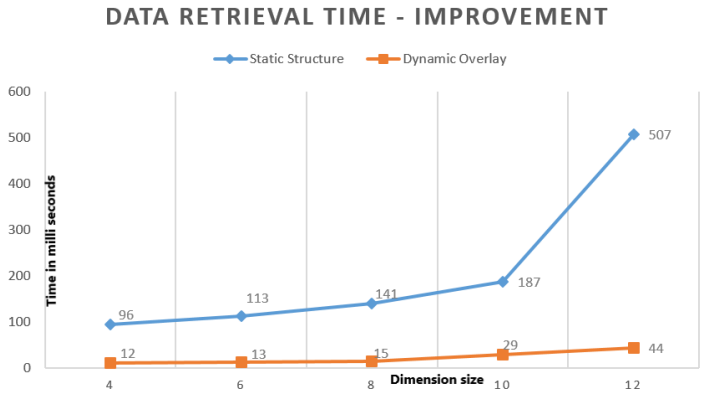

Fig. 7. Retrieval Time Optimization

The result shows the huge improvement of retrieval execution time between 9 and 10 times. It was

tabulated and depicted in the above diagram.

VII. CONCLUSION

This paper describes an improved version of the earlier Hybrid LRU algorithm, by leveraging the dynamic overlays data structure. The core logic is leveraging the dynamic overlays against the traditional way of static data structure. It adds the efficiency to hybrid LRU algorithm in two aspects - storage scalability and execution time optimization. On comparison of this research work with the traditional static algorithm, it produces a significant **10 times efficiency on Storage Scalability** factor to manage and scale the existing storage in the most efficient way. Second benefit is to obtain an effective **data retrieval execution time by 9 to 10 times** than traditional storage model.

This research paper concludes the technical strength of dynamic overlay concepts, using the experimental results on hybrid LRU. Therefore, enterprise data hub reaches the final state of the most efficient approach to execute the big data processing. Serverless computing is the upcoming trend in the industry. It can be leveraged to improve the infrastructure performance with on demand design for this research.

REFERENCES

[1] A. Murugan and S. Ganesan, "Hybrid LRU Algorithm for Enterprise Data Hub", International Journal of Innovative Technology and Exploring Engineering (IJITEE), Volume-9, Issue-1, November 2019.
[2] Alex Mircea Dumitru, Vlad Merticariu, Peter Baumann, “Array Database Scalability: Intercontinental Queries on Petabyte Datasets”, Proceedings of the 28th International Conference on Scientific and Statistical Database Management (SSDBM '16,) July 18-20, 2016, pp. 1-5, doi: 10.1145/2949689.2949717
[3] US Patent: https://patents.google.com/patent/US7062761B2/en
[4] Yan, Ling-Ling, Renée J. Miller, Laura M. Haas and Ronald Fagin, "Data-Driven Understanding and Refinement of Schema Mappings", SIGMOD 2001
[5] K M Azharul Hasan, Tatsuo Tsuji, Ken Higuchi, “An Efficient MOLAP Basic Data structure and Its Evaluation”, Proc. of 12th International Conference on Database Systems for Advanced Applications (DASFAA), vol.(LNCS)4443, April 2007,pp. 288-299, doi: 10.1007/978-3-540-71703-4_26.
[6] SRIVASTAVA, SUNEEL KUMAR, AK CHAUHAN, and AKHILESH PATHAK. "THE DETERMINISTIC MODELING FOR AVAILABILITY AND SURVIVABILITY EVALUATION OF AVIONICS DISPLAY SYSTEM USING MARKOV MODEL SIMULATION ON MATLAB." International Journal of Mechanical and Production Engineering Research and Development (IJMPERD)9.3, Jun 2019, 895-902
[7] Tom White, “Hadoop: The Definitive Guide”, O'Reilly Media, Inc., 3rd Edition, May 2012.
[8] Mehnuma Tabassum Omar - University of Saskatchewan and K. M. Azharul Hasan -Khumla University, "A scalable storage system for structured data based on higher order index array Conference Paper", December 2016, [Online]
[9] Priyanka Yadav, Vishal Sharma and Priti Yadav, "Cache Memory – Various Algorithm", International Journal of Computer Science and Mobile Computing, Vol. 3, Issue. 9, September 2014
[10] Malladi, A. V. I. N. A. S. H., and S. I. R. I. S. H. A. Potluri. "A study on technologies in Cloud-based design and manufacturing." International Journal of Mechanical and Production Engineering Research and Development 8.6 (2018): 187-192.
[11] Nathan Beckmann and Daniel Sanchez of Massachusetts Institute of Technology, "Modeling Cache Performance Beyond LRU", 2016, https://people.csail.mit.edu/sanchez/papers/2016.model.hpca.pdf

[12]R. L. Mattson, J. Gecsei and D. R. Slutz, “Evaluation techniques for storage hierarchies,” IBM Sys. J., vol. 9, no. 2, 1970
[13]M. Qureshi, A. Jaleel and Y. Patt, “Adaptive insertion policies for high performance caching,” in ISCA-34, 2007
[14]Sk. Md. Masudul Ahsan, K. M. Azharul Hasan,“An Implementation Scheme for Multidimensional Extendable Array Operations and Its Evaluation”, ICIEIS, Part III, CCIS 253, November 2011, pp. 136–150,doi: 10.1007/978-3-642-25462-8_12
[15]Viet-Trung Tran, Bogdan Nicolae and Gabriel Antoniu, “Towards Scalable Array-Oriented Active Storage: the Pyramid Approach”, ACM Operating Systems Review, vol.46(1), February 2012, pp.19-25, doi: 10.1145/2146382.2146387
[16]Jennie Duggan, Michael Stonebraker, “Incremental Elasticity for Array Databases”, In Proceedings of 2014 ACM SIGMOD International Conference on Management Data, 2014, pp. 409-420, doi:10.1145/2588555.2588569.
[17]Research and Markets, "Big Data Analytics Industry Report 2020 - Rapidly Increasing Volume & Complexity of Data, Cloud-Computing Traffic, and Adoption of IoT & AI are Driving Growth", Global Newswire, Mar 2020
[18]Vivek Seshadri, Gennady Pekhimenko, Olatunji Ruwase, Onur Mutlu, Phillip B. Gibbons, Michael A. Kozuch, Todd C. Mowry and Trishul Chilimbi, "An Enhanced Virtual Memory Framework to Enable Fine-grained Memory Management", Carnegie Mellon University, Intel Labs Pittsburgh and Microsoft Research, 2015
[19]V. Seshadri, Y. Kim, C. Fallin, D. Lee, R. Ausavarungnirun, G. Pekhimenko, Y. Luo, O. Mutlu, P. B. Gibbons, M. A. Kozuch, and T. C. Mowry, "RowClone: Fast and Energy-eXcient in-DRAM Bulk Data Copy and Initialization", MICRO, 2013.
[20]T. Sherwood, E. Perelman, G. Hamerly, and B. Calder, "Automatically Characterizing Large Scale Program Behavior", ASPLOS, 2002.
[21]W. Shi, H.-H. S. Lee, L. Falk, and M. Ghosh, "An Integrated Framework for Dependable and Revivable Architectures Using Multicore Processors", ISCA, 2006.
[22]Sadeg, Saleh A., Khalil I. Al-Samarrai, and Salem M. Rashrash. "Triple Helix Model to Develop Water and Energy Nexus for Life in Libya." International Journal of Applied and Natural Sciences (IJANS) 6.4: 163-174.
[23]P. J. Denning. "Virtual Memory", Association for Computing Machinery Computer Survey, 2(3), 1970.
[24]A. Silberschatz, P. B. Galvin, and G. Gagne, "Operating System Concepts, chapter 11. File-System Implementation", Wiley, 2012.
[25]G. S. Sohi, S. E. Breach, and T. N. Vijaykumar. "Multiscalar processors", ISCA, 1995.
[26]G. Merticariu, D. Misev, Peter Baumann, “Measuring Storage Access Performance in Array Databases", Proc. 7th Workshop on Big Data Benchmarking (WBDB), December 14-15, 2015.
[27]KUMAR, A. SENTHIL, and EASWARAN IYER. "AN INDUSTRIAL IOT IN ENGINEERING AND MANUFACTURING INDUSTRIES–BENEFITS AND CHALLENGES." International Journal of Mechanical and Production Engineering Research and Dvelopment (IJMPERD) 9.2 (2019): 151-160.
[28]J. Fotheringham, "Dynamic Storage Allocation in the Atlas Computer including an Automatic Use of a Backing Store”, Commune ACM, 1961.
[29]J. G. SteUan, C. B. Colohan, A. Zhai, and T. C. Mowry, "A Scalable Approach to Thread-level Speculation", ISCA, 2000.
[30]B. Wester, P. M. Chen, and J. Flinn, "Operating system support for application-speciVc speculation", EuroSys, 2011.
[31]LAKSHMI, I. "A COMPETITIVE STUDY ON CLOUDS COMPUTING, SERVICE ORIENTATION ARCHITECTURE AND WEB SERVICES IN ENTERPRISE NETWORK APPLICATION." International Journal of Computer Networking, Wireless and Mobile Communications (IJCNWMC)8.1, Jun 2018, 19-34

[32]S. Kumar, H. Zhao, A. Shriraman, E. Matthews, S. Dwarkadas, and L. Shannon, "Amoeba-Cache: Adaptive Blocks for Eliminating Waste in the Memory Hierarchy", MICRO, 2012.
[33]L. Jiang, Y. Zhang, and J. Yang, "Mitigating Write Disturbance in SuperDense Phase Change Memories", Direct Selling News, 2014.