STM JOURNALS

Journal of Advances in Shell Programming

Volume 1, Issue 2
www.stmjournals.com

Dynamic Pagination for Efficient Memory Management over Distributed Computational Architecture for Swarm Robotics

Ankush Rai*
Department of Applied Science, CRIAD Laboratories, Smriti Nagar, Bhilai, (C.G.), India

Abstract

In the problem of predetermined paging there involves cost for fetching each page into the cache. We design a dynamic paging algorithm which competes with the available memory for the effective implementation of dynamic paging, devoid of problems associated with memory management. This is the first dynamic paging algorithm for collectively competing with other unused memory stacks and operational register units varying from instance to instance. The proposed algorithm also offers the online sharing of memory storage for ubiquitous computing and by far stretch proves to be effective for implementing cutting edge networking tools for implementation of swarm operations over distributed computer architecture for micro robotic devices.

Keywords: *Dynamic Paging, Swarm Robotic, Distributed Computer Architecture*

****Author for Correspondence*** E-mail: ankushressci@gmail.com

INTRODUCTION

The unweighted paging is very well developed and showed that any deterministic algorithm is at least k-competitive [1]; but when randomization is allowed it shows that any randomized algorithm is at least Hk-competitive [2–5]. Subsequently, there is other Hk competitive algorithm which is easier to state and analyze.

There has been extensive research on the subject along several other directions [6]. In the direction of weighted paging, a deterministic algorithm follows for k-server problem on trees [7, 8]. Subsequently, in spite of substantial interest no randomized algorithms are acknowledged for weighted paging.

The problem is depicted as a "challenge" and remains an open problem in various other papers [8–11]. It can be perceived as a special case of the k-server problem.

In this special case, there are k servers resided on points in a metric space consisting of n-points. At each step a request is sited at one of the points and in order to serve the request the algorithm moves one of the servers to this point [12]. Here, the goal is to minimize the distance traveled by the servers.

The unweighted paging problem remains an exactly the k-server problem on a uniform metric space and leaving the weighted paging problem as identical with the k-server problem.

TECHNIQUE

We next describe formulation of Dynamic paging algorithm for the online distributed computer architecture for swarm model.

The proposed algorithm generates a tabular stacks of memory chunks for the effective of the memory management based on the time memory slots available or being used and will be available after time slot t.

Now, Let $x(i, p)$ be an indicator to the event that the solution is in state i during the p_{th} phase of memory instance and n_i be the number of phases of state i. Thus, forming a) forming a dynamic sequence

At time t, when page p_t is requested by micro devices:
Then,

 © STM Journals 2014. All Rights Reserved

Fetch $x(p_t, r(p_t, t)) \leftarrow [forwardshare(i,j), backwarshare(i,j)]$. (It will increase in times t' > t. where,

$$forwardshare(i,j) = \sum_{\substack{k \le i \\ l > j}} \left\langle \begin{matrix} forward\ weight\ of\ reference \\ from\ process\ task\ k\ to\ memeory\ instance\ l \end{matrix} \right\rangle,$$

$$backwarshare(i,j) = \sum_{\substack{k > i \\ l \le j}} \left\langle \begin{matrix} backward\ weight\ of\ reference \\ from\ prcoess\ task\ k\ to\ memeory\ instance\ l \end{matrix} \right\rangle,$$

$$z(i) = number\ of\ memeory\ insertion\ points\ among\ F_1, F_2, \ldots F_j, and$$
$$z(j) = index\ of\ last\ bottom\ inserting\ points\ among\ F_1, F_2, \ldots F_j,$$

$$x(i,p) \leftarrow \left(\sum_{t=t(i,j)+1}^{t(i,j+1)-1} F_j(t) \right) - z(i,j) - w(i),$$

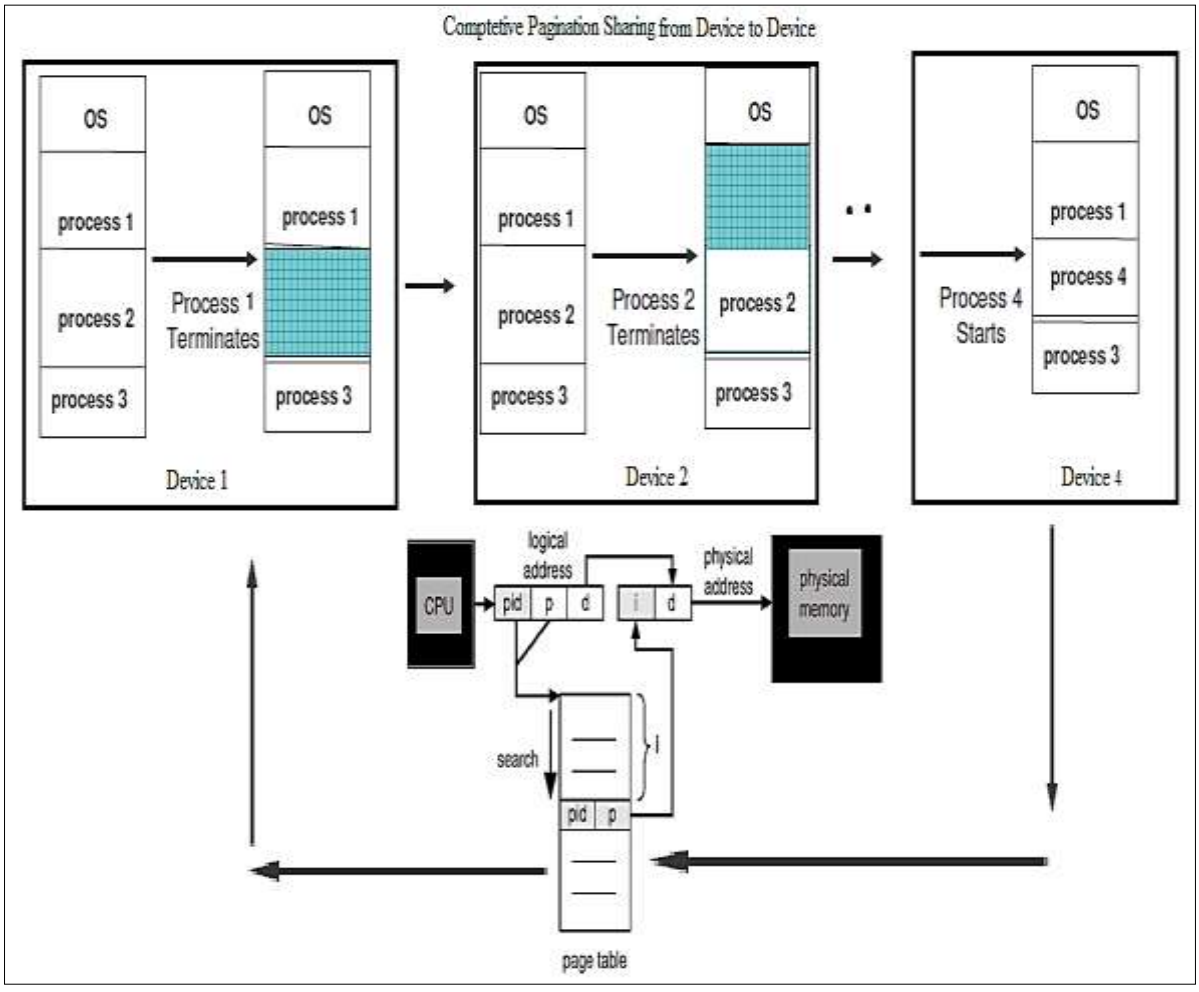

Fig. 1: *Block Diagram for Dynamic Paging over Distributed Computer Architecture for Swarm Micro Devices.*

Notice that each variable $x(p_t, r(p_t, t))$ creates r instances based on the number of devices for which memory is to be allocated and it always increases back and forth during the process until the phase p_t freezes to a specific insertion points. An instance of the process is shown in the Figure 1. This enables the hash table to get divided into several insertion points based on the time of execution of the process as depicted in the Figure 2 [13].

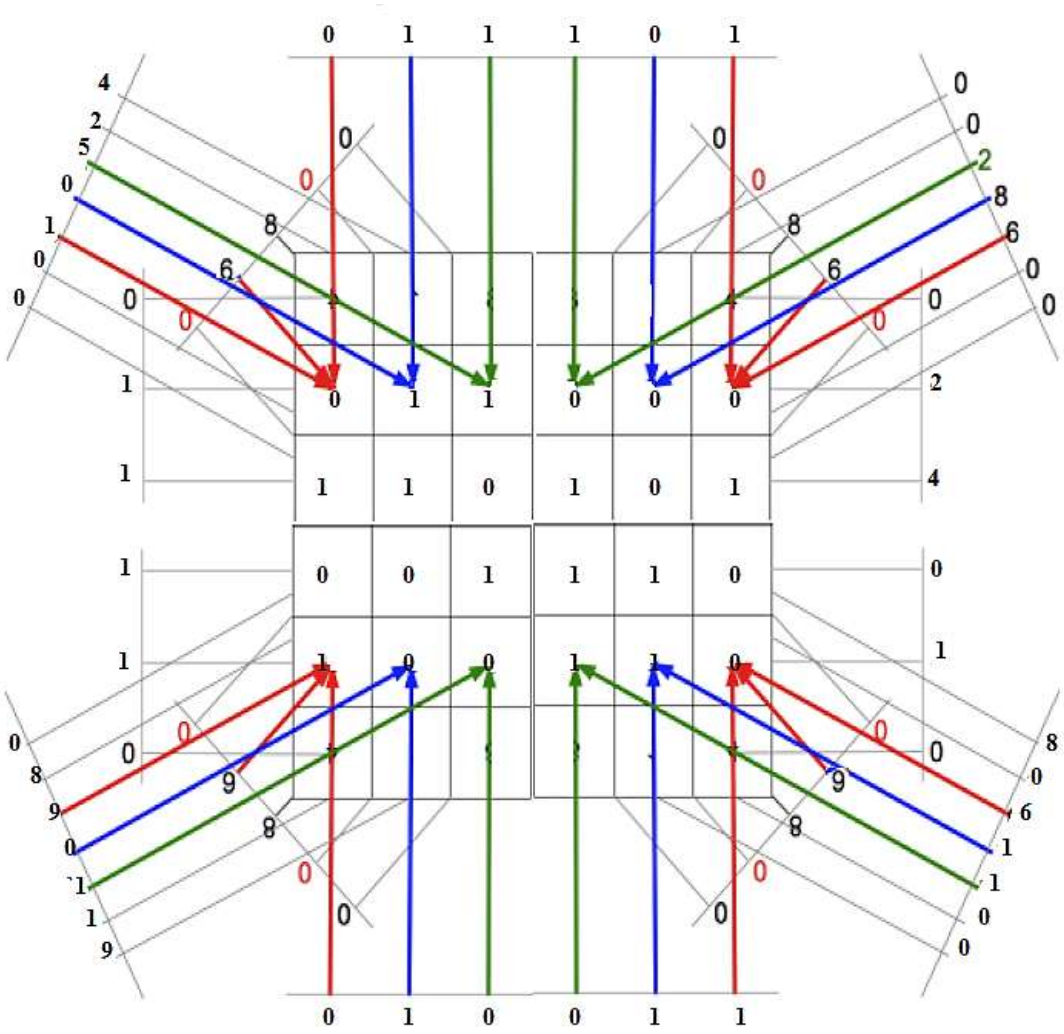

Fig. 2: *Workflow of Hash Table for Extracting the Unused Memory Stacks (represented in green) and the Queued Memory Stacks that shall be Free after Operation in it is Performed (represented in blue) and Currently used Memory Blocks (represented in red).*

CONCLUSION

Instead of using fast memory cache units in the swarmed micro devices which can hold up to k of these pages, we employed the dynamic paging algorithm to fetch the memory stacks during run flow of scripted hardware. Our system is free from the process of unnecessary request dispatch to each of the process but instead it calculates the total time for the processes t', where the paging of memory units are addressed based the availability and insertion of points of the hash table.

This saves the cost of time for the unit to incur based on time ruled eviction of weighted stacks where each page has an associated weight of logic for its relocating to the similar jobs without the intervention of actually computing the event *x* same again and again.

This effectively models the situations where some pages are expensive to fetch than others because of slower disks, etc. Since, goal is to minimize the total cost incurred by the algorithm, thus, our main result in this work optimize the paging problems online and the implementation of the proposed techniques as shown in Figure 3; the efficiency of the technique for a generalized model of paging in which pages are to be supplied to the swarmed micro devices having arbitrary weights and sizes.

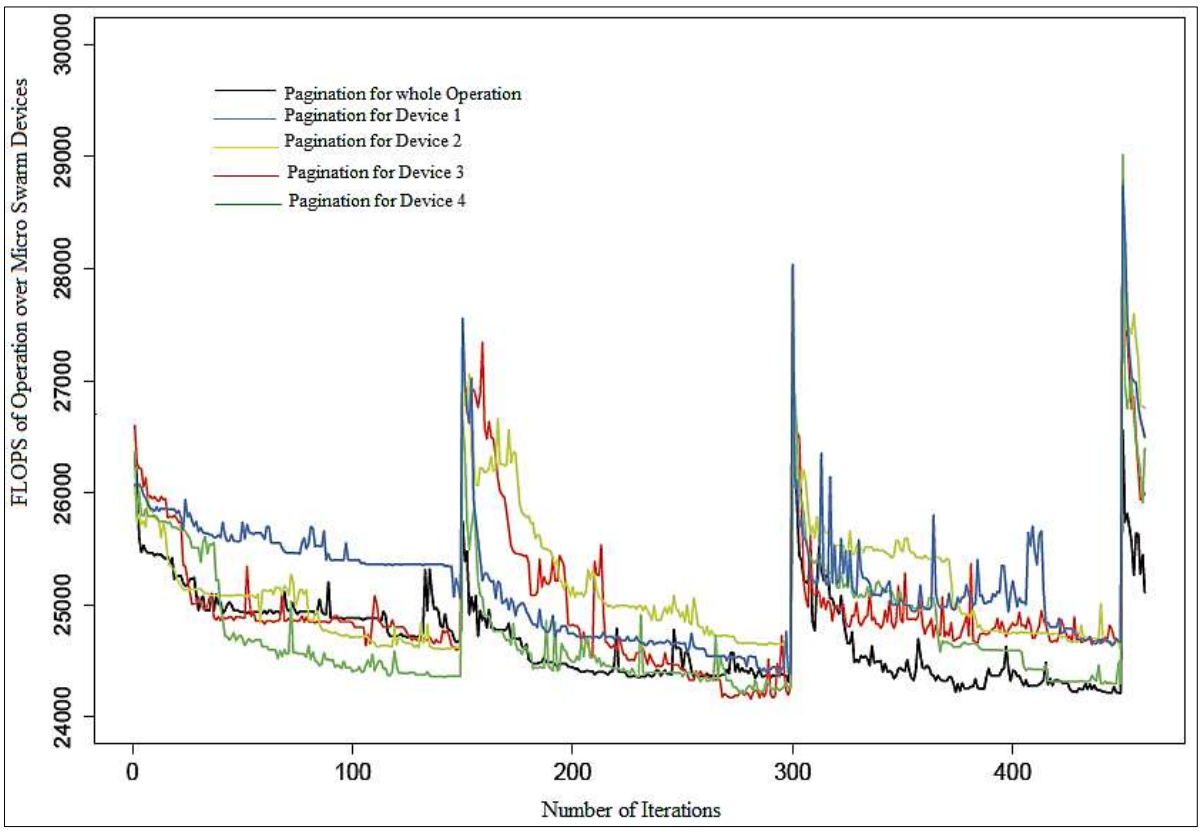

Fig. 3: *Operations Performed and the Speed of Execution (in Floating Point Operations per Second) over Swarmed Micro Devices using the Proposed Dynamic Paging Technique.*

REFERENCES

1. Sleator D. D., Tarjan R. E. Amortized Efficiency of List Update and Paging Rules. *Comm. ACM* 1985; 28(2): 202–208p.
2. Fiat A., Karp R. M., Luby M., *et al.* Competitive Paging Algorithms. *J. Algorithm.* 1991; 12(4): 685–699p.
3. Sandy Irani. Page Replacement with Multi-size Pages and Applications toWeb Caching. *In Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, New York, 1997, 701–710p.
4. Irani S.. Randomized Weighted Caching with Two Page Weights. *Algorithmica* 2002; 32(4): 624–640p.
5. Koutsoupias E., Papadimitriou C. H. On the K-server Conjecture. *J. ACM* 1995; 42(5): 971–983p.
6. McGeoch L.A., Sleator D.D. A Strongly Competitive Randomized Paging Algorithm. *Algorithmica* 1991; 6(6): 816–825p.
7. Achlioptas D., Chrobak M., Noga J. Competitive Analysis of Randomized Paging Algorithms. *Theor. Comp. Sc.* 2000; 234(1-2): 203–218p.
8. Borodin A., R. El-Yaniv, *Online Computation and Competitive Analysis*, Cambridge University Press, 1998.
9. Chrobak M., Karloff H. J., Payne T. H., *et al.* New Results on Server Problems. *SIAM J. Discrete Math* 1991; 4(2): 172–181p.
10. Borodin A., Linial N., Saks M. An Optimal Online Algorithm for Metrical Task Systems. *In Proceedings of the 19th Annual ACM Symosium on Theory of Computing*, New York City, NY, 1987, 373–382p.
11. Mark S. Manasse, Lyle A. McGeoch, Daniel D. Sleator. Competitive Algorithms for Server Problems. *J. Algorithm.* 1990; 11(2): 208–230p.
12. Albers S. Online Algorithms: A Survey. *Math. Program.* 2003; 97: 3–26p.
13. Young N. E. The K-server Dual and Loose Competitiveness for Paging. *Algorithmica* 1994; 11(6): 525–541p.