Swarm Intelligence Based Demand Partitioning For Virtual Memory Management

Mr. N. M. Shivratriwar
(Department of Information Technology, P.R.M.I.T.& R., Badnera-Amravati)
nitinshivratriwar@rediffmail.com
Mr. M.R.Dhande
(Department of Computer Technology, V.Y.W.S.Polytechnic, Badnera-Amravati)
Milinddhande98@gmail.com
Mr. R.A.Tiwari
(Department of Computer Technology, V.Y.W.S.Polytechnic, Badnera-Amravati)
softyraja@gmail.com
Mr. S.P.Thakare
(Department of Information Technology, P.R.M.I.T.& R., Badnera-Amravati)
Spthakare82@gmail.com

Abstract

For multiprogramming, multitasking environment system required useful mechanisms that partition a main memory into blocks. The size of block must be sufficient to execute process. Finding the preferred size of block is daunting tasks. This paper proposed a swarm intelligent based demand partitioning for virtual memory. For scalable and efficient memory management .Each process begin with different size. The task of memory allocation algorithm is to find size of process and then divides a process into many blocks. So to process fit into memory and execute. As main memory receives many requests for execution. To execute all requests we need larger size of main memory.

It is always difficult to set the appropriate size of main memory which must be cost and performance effective.

I. Introduction

Operating system like OS/360 running on IBM hardware used the fixed memory partition .Where main memory was divided into various sections known as partitions. These partitions may be of varied in sizes. Once size of section decided at time of system generation, then it is fixed. New partition will only take place when operations are stopped and operating system has to be reloading. Fixed partitioning have its own limitations like wastage of memory, access time is not very high. Time complexity is very low because allocation / deallocation routines are simple[1].
We need a partitioning scheme which handle unpredictable memory demand for example new embedded devices, which are heavily relay on demand memory due to unpredictability of the input data. Though number of application running concurrently defined only by users. Same time embedded devices needed a design methodology that can handle preciously the complex memory demand behavior. Virtual memory management using demand partition is the solution on problem and it can be implement with demand segmentation or demand paging. This paper proposed use of ant colony algorithm to predict the memory allocation on demand. The ant colony algorithm can be run continuously and adapt to changes in real time. It also minimizes wastage of memory.

II. Related Work

Recently people worked on new portable consumer embedded devices that execute multimedia and wireless applications that demand extensive memory. New portable device heavily rely on demand memory due to unpredictability of the input data and system behavior. They proposed new methodology allows designing custom demand memory management with a reduced memory for such kind of dynamic applications. The experimental results reflect high rate of effective memory utilization [2].Many application begins with allocation of some memory to a server and allocated additional memory on demand. Unless a server demands more memory, it never exceeds its initial size. When a server does demand more memory than allowed, it spills blocks of intermediate result set to disk [3].

III. Research Work

Use of swarm intelligence to complete a given task is a new approach. It inspire from the social behaviors of insects and of other animals to problem solving. They have an advantage over different evolutionary approaches like simulated annealing and genetic algorithm of similar problems when the demand graph may change dynamically. The ant colony algorithms have an ability to run continuously and adapt to changes in real time. With an ant colony algorithm based memory partition, we use the indexed allocation to support direct accessing. Benefits of swarm intelligence based demand partition virtual memory management are as below

i) Minimizing memory wastage due to fragmentation.

ii) Easy sharing and control.

iii) All requests (programs/application) get address space.

iv) Efficient free space management.

IV. Cache Page Replacement Policy

Virtual memory is a one feature of an operating system that allows a computer to compensate for shortages of physical memory by temporarily transferring pages of data from random access memory to disk storage. It is a useful mechanism. But required page is not found in main memory will cause a serious penalty. So swapping of pages in and out must be done efficiently. For this various replacement policies are in used. Here we deliberately work for application of ant colony based cache page replacement policy.

The general algorithm is relatively simple. It uses a set of ants, each making one of the possible round-trips along the pages in cache to sprayed pheromones. At each stage, the ant chooses to move from one page to another according to some rules:

1) It must visit each page exactly once.

2) A page having strong output. Then gotten more pheromone on it.

3) The more intense the pheromone, the greater the probability that page will not replace.

4) Having completed its page visit, the ant collect pages having less intense the pheromone for replacement.

5) Before start of new iteration, trails of pheromones evaporate.

Several attributes of ant colony algorithm like seeking a strongest from colony at each iteration, inspire us to check application of this algorithm for page replacement. For programming convenience, we declare four input variables that are pagehitcount, pageaccessount ,pagelastaccesstime, pagetimeincache and one pageoutput variable.

1. The pagehitcount is the number of times a cache page is referenced without being loaded from secondary memory, i.e. no page fault. Hit count and page fault are antagonistic attributes, i.e. a higher hit count means less page faults. It is desirable to have a page replacement policy with as lesser page faults, or high hit count.

2. The pageaccesscount is the number of time a cache page is referenced regardless to page fault. A high count means the page is referenced often and it should stay in cache so that the next reference to this page can be a page hit.

3. The pagelastaccesstime is the time when the page is last referenced. It is a timestamp value in millisecond since Epoch time. The larger the value, the more recent the timestamp is.

4. The pagetimeincache is the duration that the page has been loaded into cache. A high value can be interpreted either good or bad depending on the page replacement policy is used.

And output variable

1. The pageoutput variable weaker its value can be used for selecting page for replacement.

These attributes are real-coded because of speed consideration. Time values tend to be large numbers because they are expressed in milliseconds. Coding them in real-values avoids extraneous mathematical conversion which takes up computation cycles. In a real OS, program execution halts at page fault to load the new page. The first step to load a new page is to select a page for removal. Therefore, selecting a page speedily is not only desirable, but necessary for quick resumption of program execution.

VI. Conclusion

Memory allocation using fixed static allocation cause wastage of memory more. Then solution is not only change the memory allocation method. If it used with efficient page replacement policy it gives best results. Ant colony based page replacement policy performs better under various page sizes and cache sizes then its counterparts. In much and more areas it produces human-competitive results. It is a very promising to see that a simple ant colony based cache replacement policy model can perform so closely to other well-studied replacement algorithms.

Reference

[1] Silberschatz Abraham and Gane Greg,Galvin PetterBaer,"operating system concept", Published Addison-Wesley Reading, 7 Edition, May 2006.

[2] Davide Atienza and S. Mamagkakis, F. Catthoor, J.M. Mendias,"Dynamic Memory Management Design Methodology for Reduced Memory Footprint in Multimedia and Wireless NetworkApplications", Publisher IEEE Computer Society Press, Washington DC, USA.

[3] Red Brick Warehouse available at http://www.ibm.com/software/data/informix/redbrick/