Perspective

iMedPub Journals
www.imedpub.com

American Journal of Computer Science and Information Technology
ISSN 2349-3917

2022
Vol.10 No.12:005

Developing a Method for Resolving Memory Leak Issues without the Assistance of a Programmer

Yogesh Huang*

Department of Mechanical Engineering, National Yang Ming Chiao Tung University, Taiwan

***Corresponding author:** Yogesh Huang, Department of Mechanical Engineering, National Yang Ming Chiao Tung University, Taiwan Email: huang324@gmail.com

Received date: November 09, 2022, Manuscript No. IPACSIT-22-15613; **Editor assigned date:** November 11, 2022, PreQCNo.IPACSIT-22-15613(PQ); **Reviewed date:** November 22, 2022, QC No IPACSIT-22-15613; **Revised date:** December 02, 2022, Manuscript No. IPACSIT-22-15613 (R); **Published date:** December 22, 2022, DOI: 10.36648/ 2349-3917.10.12.5

Citation: Huang Y (2022) Developing a Method for Resolving Memory Leak Issues without the Assistance of a Programmer. Am J Compt Sci Inform Technol Vol. 10 Iss No.12:005.

Description

Defects in memory management that are connected to dynamic storage allocation are some of the most difficult and difficult defects that are currently in existence. Memory leaks are common in long-running information server applications, especially those written in a programming language that uses dynamic memory allocation and requires programmers to explicitly deallocate dynamically allocated storage when the program no longer needs it. The capacity of garbage collectors to limit the amount of storage lost as a result of memory leaks makes them efficient. Memory and its associated page files can be sized to accommodate the application's usage as long as the amount of lost storage is limited. As a result, the goal of our research is to find a solution that not only addresses these issues in fielded systems but also limits memory storage loss caused by memory leaks. In the past, every garbage collector, even conservative garbage collectors, required some assistance from a programmer. We describe a method for collecting garbage on its own that meets the research objective: find and fix memory leaks without the help of a programmer, compiler, or linker. The algorithm's efficient operation and effectiveness are demonstrated by us.

Traditional Management of Dynamic Storage

When long-running servers are involved in large-scale information processing systems, memory leaks are especially problematic. Software that does not have memory leak defects is frequently deployed into operational configuration in such systems. Over time, memory leaks build up, reducing system performance and ultimately resulting in system failure. Memory leaks have the potential to significantly disrupt software-intensive systems like web search engines and other digital repositories. Massive amounts of electronic information are connected *via* the Internet in this age of digital information, and steady advancements in information retrieval technology continue to improve the utilization of this massive base of information. Traditional management of dynamic storage has been significantly affected by these large-scale information retrieval systems' intense, concentrated I/O processing characteristics. In such systems, the symptoms of flaws in traditional dynamic storage management frequently become much more apparent, severely reducing system performance. In large-scale information processing systems, the demands placed on the memory systems are enormous, and these I/O demands continue to rise as electronic information becomes increasingly accessible. We are aware that servers in large-scale information processing systems cannot long tolerate even minor memory leaks due to the enormous I/O demands. We also acknowledge that numerous C/C++-based implementations of such systems exist and have been deployed.

Feasibility and Benefits of Providing Autonomous Garbage Collection as Part of the Execution Environment

The dynamic allocation and reallocation of memory is managed by the programmer in the C/C++ language. In addition, despite employing a mature software development methodology and employing disciplined and highly skilled C/C++ programmers and system test engineers, these systems, like other software-intensive systems developed in C/C++, always have some level of memory leaks that go unnoticed and are shipped with the system. As a result, our primary focus is on developing a method for resolving memory leak issues without the assistance of a programmer and without requiring either object code or source code. Our method works well in environments with soft real-time execution, similar to information server execution paradigms. From a usability point of view, it's important for information servers to always respond quickly to queries. For instance, 95% of the time, the query response time should be less than a second. However, unlike in hard real-time systems, no catastrophic failure occurs if the target query response time is missed. The garbage collector runs on the DEC OpenVMS operating system and is implemented in C. The algorithm is described using pseudocode as opposed to the actual source code. A variant of the mark–sweep garbage collection method is at the heart of the algorithm. When describing the algorithm, the terms cycle and phase are frequently employed. The feasibility and benefits of providing

autonomous garbage collection as part of the execution environment for information retrieval systems are demonstrated by the garbage detection and collection schema developed in this study. We developed and tested a strategy that improves on existing approaches for resolving dynamic memory management flaws, which are particularly prevalent in server applications that run for extended periods of time.