Computer Languages, Systems & Structures 44 (2015) 342–354

Contents lists available at ScienceDirect

Computer Languages, Systems & Structures

journal homepage: www.elsevier.com/locate/cl

The intelligent memory allocator selector

Onur Ülgen [a,*], Mutlu Avci [b]

[a] Cukurova University, Faculty of Engineering and Architecture, Computer Engineering Department, Balcali, Adana, Turkey
[b] Cukurova University, Faculty of Engineering and Architecture, Biomedical Engineering Department, Balcali, Adana, Turkey

ARTICLE INFO

Article history:
Received 22 June 2015
Received in revised form
17 August 2015
Accepted 16 September 2015
Available online 3 October 2015

Keywords:
Memory fragmentation
Memory allocator
Garbage collection
Virtual machine

ABSTRACT

Memory fragmentation is a serious obstacle preventing efficient memory usage. Garbage collectors may solve the problem; however, they cause serious performance impact, memory and energy consumption. Therefore, various memory allocators have been developed. Software developers must test memory allocators, and find an efficient one for their programs. Instead of this cumbersome method, we propose a novel approach for dynamically deciding the best memory allocator for every application. The proposed solution tests each process with various memory allocators. After the testing, it selects an efficient memory allocator according to condition of operating system (OS). If OS runs out of memory, then it selects the most memory efficient allocator for new processes. If most of the CPU power was occupied, then it selects the fastest allocator. Otherwise, the balanced allocator is selected. According to test results, the proposed solution offers up to 58% less fragmented memory, and 90% faster memory operations. In average of 107 processes, it offers $7.16 \pm 2.53\%$ less fragmented memory, and $1.79 \pm 7.32\%$ faster memory operations. The test results also prove the proposed approach is unbeatable by any memory allocator. In conclusion, the proposed method is a dynamic and efficient solution to the memory fragmentation problem.

1. Introduction

Memory fragmentation is a serious obstacle preventing efficient usage of memory. It appears in time while program is allocating and deallocating memory, shown in Fig. 1. In the last situation of the figure, there are 9 blocks of free memory; however, even 4 blocks of memory cannot be allocated due to fragmentation.

Memory fragmentation can be split into two types: external and internal fragmentation. From operating system perspective, external fragmentation describes fragmentation between processes. Hence, internal fragmentation occurs inside of the processes.

In modern computers, external fragmentation was solved utilizing paging, detailed in Section 2.1 [1–3]. Internal fragmentation was solved utilizing memory compaction of garbage collectors (Section 2.2). However, due to serious performance impact, memory and energy consumption [4–6], it is an inefficient solution. Even further, some garbage collectors do not support memory compaction; thus, they do not prevent memory fragmentation, such as Android KitKat and predecessors [7]. Therefore, internal fragmentation is still a challenging problem in this area.

* Corresponding author.
E-mail addresses: oulgen@cu.edu.tr (O. Ülgen), mavci@cu.edu.tr (M. Avci).
URL: http://www.onurulgen.com (O. Ülgen).

http://dx.doi.org/10.1016/j.cl.2015.09.003
1477-8424/

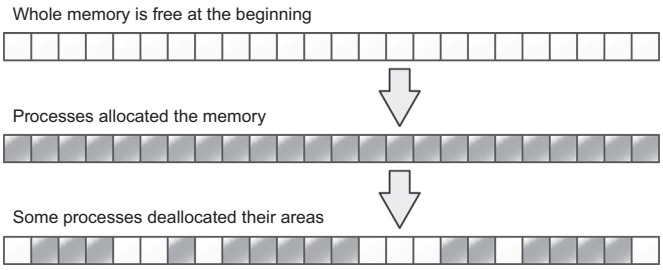

Fig. 1. Memory fragmentation.

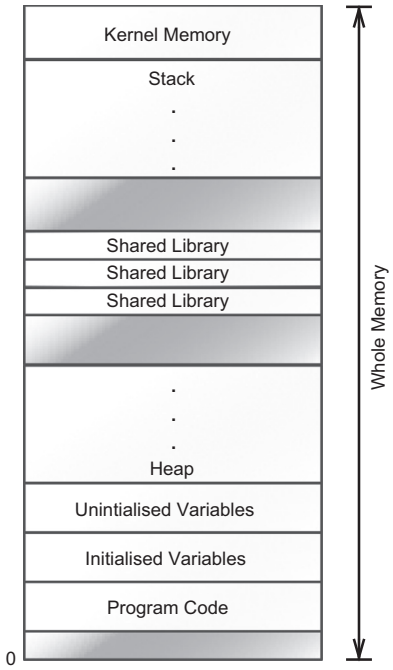

Fig. 2. Process memory image.

For avoiding disadvantages of garbage collectors, memory allocators were developed for general purpose [8–12], multi-threaded processes [13], network applications [14], object oriented programming languages [15,16], video-on-demand servers [17], etc.

Modern operating systems assign a default memory allocator to every process. If this memory allocator was insufficient for the application then software developers should determine and add the most efficient and fastest memory management algorithm for their applications [18]. However, if the developer had no knowledge about memory allocators then the application would lose performance and memory. Instead of forcing developer to select a memory allocator, operating system should determine memory management algorithm for each process to accomplish the best performance and efficiency.

In this work, we propose an intelligent memory allocator selector (IMAS) that dynamically decides the best memory allocator for each process. The IMAS tests each process separately with various memory allocators. After the testing, it selects an efficient memory allocator. It also adopts to the operating system (OS) conditions. If OS runs out of memory then the IMAS selects the most memory efficient memory allocator for new processes. If processes consume most of the CPU power then it selects the fastest memory allocator. Otherwise, the balanced memory allocator is selected.

The IMAS has a test system, which logs CPU performance values and memory fragmentation ratios. These test results are used to select efficient memory allocator for processes. Therefore, they can be used for comparison between memory allocators and the IMAS. Tests have been done in a regular personal computer during its daily activity. 107 processes have been fully tested during our tests. According to test results, the IMAS provides up to 58% less fragmented memory than default memory allocator of OS. In average of 107 processes, the IMAS is $7.16 \pm 2.53\%$ memory efficient. For CPU performance, the IMAS is up to 90% faster; and in average, it is $1.79 \pm 7.32\%$ faster.

Test results prove our claims, the IMAS minimizes memory fragmentation by selecting an efficient memory allocator for each process. Therefore, memory management is done in faster and more efficient way.