Sri Lanka Institute of Information Technology

In-Depth Analysis of Operating System Concepts through Research Paper Exploration

Rethinking Memory Management in Modern Operating System

[Group Assignment – Group 16]

IE2032 – Secure Operating Systems

B.Sc. (Hons) in Information Technology

specializing in Cyber Security

Group details

Declaration:

We hold a copy of this assignment that we can produce if the original is lost or damaged. We hereby certify that no part of this assignment has been copied from any other group's work or from any other source. No part of this assignment has been written / produced for our group by another person except where such collaboration has been authorized by the subject lecturer/tutor concerned.

Group : 16

Student Registration Number	Student Name
IT23368248	S.M. Dissanyaka
IT23406094	S.D. Jayasinghe
IT23381322	Rajapaksha R.P
IT23265592	Rajapaksha R.M.P.U

Date of submission: 29th of September 2024

Table of Contents

Acknowledgement

All contributions of authors of the IEEE Transactions on Computers paper which is "Rethinking Memory Management in Modern Operating Systems: Horizontal, Vertical, or Random?" are acknowledged for their valuable research, which really inspired the work that we have done.

We are thankful for all lecturers in Sri Lanka Institute of Information Technology (SLIIT) for their guidance. Their work has influenced this research.

This paper expands upon the discoveries made in their work, and we recognize all the individuals and organizations involved in the research process.

Abstract

This report gives information about advanced memory management techniques motivated by the report presented in the paper which is “Rethinking Memory Management in Modern Operating Systems: Horizontal, Vertical, or Random”. In modern multicore computers, managing memory efficiency is crucial because new applications and tasks become more complex and run more than 1 task with multiple levels of memory.

From the original report HVR-modern-multicore approach uses vertical and horizontal partitioning to divide memory at the same level to provide greater flexibility. Random-interleaved allocation helps to minimize memory access issues. These methods are better for unique requirements of various applications. These results from examination of 2000 different jobs with various memory rules enabled us to create efficient memory management for a wide range of tasks.

To test this new approach, they implemented it in the Linux kernel This demonstrates how the efficiency of contemporary computers may be greatly increased by utilizing a more flexible and tiered memory management mechanism.

Key-words: memory-management, HVR-framework, modern-OS

Introduction

In modern multicore systems, efficient memory resource sharing is crucial. Traditional memory management is done through address interconnection, in which a memory request's physical address defines which part of the cache or memory responds to it successfully in the past but in modern-multicore computers, it gives errors

Most of the past research has been on managing memory at a single level by horizontally partitioning memory resources. Though these methods mitigate conflicts between programs but there are some limitations too. Firstly, they only focused on one level of the memory hierarchy. Hence conflicts at remaining other levels are unsolved. Secondly, most of the approaches used single memory management strategies but they didn't give the specific memory demands of different, multiple applications. Finally, those approaches resulted in inefficient recourse allocation because they ignore the unique features of every application.

So, this research report introduces a novel memory management strategy that draws inspiration from the HVR framework presented in the original study to solve these issues.

Methodology

To visually connect the problems and solutions, we can use a mind map. This will help in understanding the relationships and interdependencies between the various components of the problem and solution

- Branches:
    - Problem 1: Inefficient Memory access patterns
        - Solution: Vertical Partitioning
        - Solution: Horizontal Partitioning
    - Problem 2: Limited Flexibility
        - Solution: Online Classification
    - Problem 3: Difficulty Handling Dynamic Workloads
        - Solution: Dynamic Application Classification
    - Problem 4: Complexity of Managing multiple Applications
        - Solution: Partitioning and Coalescing

- Problem 5: Overhead Associated with Memory management
  - Solution: Efficient implementation
  - Solution: Curve-V

Problem 1: Inefficient Memory access Patterns

- Vertical Partitioning: Divides resources across memory levels (e.g., DRAM, LLC) to

Reduce contention and improve access efficiency, preventing bottlenecks.

- Horizontal Partitioning: partition resources within each memory level to minimize interference between applications, enhancing performance.

Problem 2: Limited Flexibility

- HVR automatically selects optimal memory management policies based on application characteristics, ensuring high performance across various workloads.

Problem 3: Handling Dynamic Workloads

- HVR monitors and classifies application by their changing memory and cache usage, adjusting memory policies dynamically to maintain performance in evolving workloads.

Problem 4: Managing Multiple Applications

- HVR uses data-driven rules to guide resource partitioning and merging, optimizing memory use and minimizing conflicts in multi-application environments.

Problem 5: Overhead in Memory Management

- Efficient Implementation: HVR reduces overhead with hash-based researching and efficient page allocation, minimizing system performance impact.
- Curve-VP: This HVR variant lowers page migration overhead through rigid partitioning, reducing unnecessary page movements and improving efficiency.

Main problem and challenges

In this article “Rethinking Memory Management in Modern Operating Systems” describes the challenges of traditional memory management techniques on modern multicore systems. Normally in memory management, while hardware uses an interleaving technique to distribute memory addresses, the OS randomly allocates the memory addresses to optimize the performance. However, this method struggles with the modern workloads which are more complex. This is resulting in issues like resource contention, inter-program interferences and poor-memory and cache utilization. These problems get more and cause major performance slowdowns worse when many applications are running at same time on complex memory systems.

To address these challenges, the article introduces the HVR framework, which combines three memory management strategies which are Horizontal partitioning, Vertical partitioning and Random allocation. Horizontal partitioning is a technique that divides data across different sections or partitions at the same memory level, it helps to reduce the competition for memory access. Vertical partitioning extends to multiple levels like cache and DRAM to optimize the overall performance. Random allocation assigns memory non-sequentially and it helps to prevent hotspots and conflicts by distributing memory evenly and improve system performance.

The researchers tested the framework on over 2000 workloads and developed a set of practical guidelines for dynamic memory allocation that are customized to the specific characteristics of each workload.

Proposed solutions

- There are five solutions by the authors.

Vertical Partitioning

- ✓ Vertical partitioning is a database optimization approach which divides large tables into smaller tables. This also has subset of the original table's columns, and it's based on frequency of column access.

- ✓ Vertical partitioning can be identifying on different ways to improve the query performance, reduced I/O operations, enhanced scalability, simplified data management and all.

- ✓ Basically, this working on the column grouping, table creation, data distribution.

- ✓ According to the research vertical partitioning can be used for overlapped bits in physical page addresses to index banks and cache sets for partitioning, memory hierarchy vertically and achieve accumulation.

Sys-Mon

- ✓ This Microsoft tool provides detailed information about system activities, including memory usage. It's an effective tool which can analyst security, system administrators & developers to monitor & evaluate the system behavior.

- ✓ Sys-Mon in memory management for OSes works in different ways.
    1. Memory usage monitoring
    2. Memory leak detection
    3. Performance
    4. Security monitoring

- ✓ According to this Sys-Mon was created, a kernel module utility which dynamically captures application activities such as memory footprint and active pages.

- ✓ Page can re-use time, cache and information. Hardware performance can be utilized without offline profiling.

Partitioning & Coalescing

- ✓ Those are the two basic ideas in memory management in OS. Particularly dynamic memory allocation systems they used for optimizing memory use by dividing it for smaller units and combining free units.

- ✓ When getting about the partitioning known as partitions, are subsequently assigned to certain data structures or processes as required. This can divide into two primary partitioning schemes.

1. Fixed-size partitioning
2. Variable-size partitioning

- ✓ Coalescing the joining-process nearby free partitions to create a bigger, single free partition. This is basically done to minimize the number of free partitions and maybe avoid fragmentation.

- ✓ According to the research for partitioning & coalescing authors proposed this into partitioning policies based on substantial experimental findings across various workloads. Coalescing rules for the partitioning needed, while allowing non-interfering programs to share the response.

Multi-policy framework

- ✓ Is a system that allows for concurrent application of multiple policies to govern a particular process or system. Primary goal of the multi-policy framework is to achieve the balance between conflicting objectives or constraints.

- ✓ Multi -policy framework has key components

1. Policy representation
2. Policy enforcement
3. Policy conflict resolution
4. Policy adaption

- ✓ According to the research-paper this is the proposed solution(approach) for that. In this, an HVR-framework that may split vertically, horizontally, or randomly was created.

- ✓ For interleaving further design a variation VP to approaching in extracts higher performance.

Implementation in Linux kernel

- ✓ Linux kernel is a complex software which forms the core of the Linux OS.
- ✓ Key steps of kernel implementation

    1. Understanding the feature
    2. Choosing the right location
    3. Writing the code
    4. Integrating with existing code
    5. Testing & debugging
    6. Documentation

- ✓ According to the authors they introduce x-buddy, a redesigned physical page indexing system in Linux kernel. Policies and partitioning/coalescing rules in HVR. Sys Mon tool is implemented on kernel.
- ✓ Lines of source code within the Linux kernel tree. HVR needs hardware updates then it works nicely with multi-threading and multiprogramming.

Results & Conclusion

Result:

- The HVS framework was created to enhance memory management, demonstrated positive results in terms of lower overhead and enhanced performance. There are 3 main sources of overhead in HVR:

✓ Page Table Sampling:
  - Tracking memory utilization during workload classification process.
  - Page table-sampling takes a short-time and overhead is minimal.
  - But less overhead is still there even in workloads that are continuing.

✓ Page Allocation and Indexing:
  - HVR allocates memory using an optimized buddy system, which frequently results in minimal overhead (>0.1% per page allocation)

✓ Page Migrations:
  - Switching-pages may be required to enhance memory allocation, however doing such comes with a 0.8% overhead.
  - However, HVR uses lazy page migration method to mitigate overhead by previous approaches.

- HVR performed well under various workload conditions.

  Examples:

  - HVR modified memory sharing when workloads included crucial, high-latency programs to reduce performance loss.

- According to the report, depending on the applications, dynamic partitioning of HVR mitigates performance loss 0.01-0.03 for smaller-workloads and 0.1-0.3 for larger ones.

- With minimal code modifications in Linux kernel's implementation of HVR, made it possible for function in real-world environments well.

- It shows that HVR mitigates memory and cache-conflict and maximizes overall performance by up to 21%, which can be valuable for businesses and data centers that depend on memory-intensive workloads.

7.2. Conclusion

In this report refers HVR framework and it provides flexible and efficient solution to manage memory-recourses across multiple-levels of the memory hierarchy. HVR addresses the various requirements of modern applications by combining randomized page allocation with vertical and horizontal partitioning method successfully.

The ability of the framework automatically selects the most efficient memory management policies based on workload aspects offers considerable performance improvements over traditional approaches with up to 20% performance gains in several situations. HVR is a real-world solution to minimize overhead and flexibility to various domains where effective memory management is important like cloud computing.

According to this report, HVR is an important achievement in modern-OS-approaches, which is providing a real-world solution to the rising problems of memory conflict and resources allocation in multicore systems.

References

L. Liu, Y. Li, C. Ding, H. Yang, and C. Wu, "Rethinking Memory Management in Modern Operating System: Horizontal, Vertical or Random?," *IEEE Transactions on Computers*, vol. 65, no. 6, pp. 1921–1935, Jun. 2016, doi: https://doi.org/10.1109/TC.2015.2462813