International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 4
Article Received:*10 February 2023* ***Revised:****20 March 2023* ***Accepted:****30 March 2023*

Task Load and Work Property-Based Virtual Machine Dispatching Algorithm

R.Kennady[1], O.Pandithurai[2]
[1]Department of Artificial Intelligence and Data Science, Rajalakshmi Institute of Technology, Chennai, Tamilnadu
[2]Department of Computer Science and Engineering, Rajalakshmi Institute of Technology, Chennai, Tamilnadu
[1]kennady.r@ritchennai.edu.in,[2]pandics@ritchennai.edu.in

Abstract: This research proposes a virtual machine dispatching algorithm that takes into account the task load and work property of a virtual machine. The algorithm utilizes an interruption monitoring module to assess the interruption frequency of the virtual machine and determine whether it belongs to a CPU-dense or I/O-dense type. Additionally, a CPU monitoring module tracks the current CPU utilization rate to monitor the task load. By considering the historical information within a specified time range, the algorithm calculates a dispatching time segment for the virtual machine and notifies the dispatcher to update the time segment. The dispatcher module adjusts the credit value dispatcher based on inputs from the monitoring and CPU modules. It applies different dispatching time segments to virtual machines with varying work properties, aiming to reduce I/O request delays, provide sufficient time for I/O request handling, and minimize additional overhead. Notably, the modifications occur within the virtual machine monitor, ensuring practicality and adaptability.

Keywords: *Virtual machine, Dispatching algorithm, Task load, Work property, Interruption monitoring, CPU monitoring, Time segment, I/O request delay, Additional overhead, Virtual machine monitor.*

Introduction:
In the field of virtualization, efficient management and scheduling of virtual machines (VMs) are essential to optimize resource utilization and ensure smooth system operations. The dispatching algorithm plays a crucial role in assigning tasks to VMs. However, existing dispatching algorithms often neglect the dynamic nature of workload characteristics and fail to consider the work property of VMs, leading to suboptimal performance and increased I/O request delays. To address these challenges, this research proposes a novel virtual machine dispatching algorithm based on the task load and work property of VMs. By integrating an interruption monitoring module and a CPU monitoring module, the algorithm aims to improve dispatching decisions and reduce I/O request delays. The algorithm calculates dispatching time segments for VMs, taking into account the current task load and historical information. The dispatcher module modifies the credit value dispatcher, enabling the application of different dispatching time segments based on the work properties of VMs.

Background:
Virtualization technology has revolutionized computing environments by allowing multiple VMs to run concurrently on a physical machine. Dispatching algorithms play a vital role in efficiently allocating resources and managing VMs. However, traditional algorithms often overlook the dynamic nature of workloads and the specific characteristics of VMs, leading to imbalanced resource utilization and increased I/O request delays.
Intel Virtualization Technology has been in existence for over 50 years in various forms. The abstract nature of virtualization allows for the logical representation of available resources without being restricted by physical conditions. Intel Virtualization Technology adds a virtualization layer to systems, enabling the abstraction of lower-level physical resources into virtual resources for use by applications in the upper strata. This technology allows for the creation of multiple virtual resources from a single physical resource and the integration of multiple physical resources into a virtual resource.
The Virtual Machine Monitor (VMM) is a critical component of Intel Virtualization Technology. Positioned between the hardware layer and the virtual machine, it operates at the highest privilege level. The primary function of the VMM is to abstract the underlying hardware resources and transform them into virtual resources to be used by the virtual machines in the upper layers. The VMM

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 4
***Article Received:**10 February 2023 **Revised:**20 March 2023 **Accepted:**30 March 2023*

is responsible for managing the virtual environment and the allocation of physical resources.[4,5]

The current industry mainstream virtual machine monitors include Xen, KVM, VirtualBox, Hyper-V, and VMware. Among these, Xen has gained significant popularity and holds a prominent position in both practical applications and academic research. The default dispatching algorithm in Xen is the credit value dispatching algorithm, also known as the Credit Scheduler. In this algorithm, time is assigned as credit values, and each virtual CPU (VCPU) is assigned a credit value for each distribution period. As a VCPU runs for a certain period, the corresponding credit value is deducted. If the credit value drops below zero within a dispatching cycle, the VCPU is rescheduled.[1,2]

The introduction of the virtual machine monitor introduces new challenges and changes the way problems were traditionally solved in legacy operating systems. One crucial issue is the problem of I/O operating lag. In a legacy operating system, when an I/O request occurs, the operating system can detect the request and dispatch it to the appropriate recipient, allowing it to interrupt the current operation and process the I/O request promptly. However, in a virtualized environment, all I/O requests are received by the VMM, which then transmits them to the corresponding virtual machine. The operating system cannot identify the recipient of a specific I/O request, leading to delays. When multiple virtual machines share a physical CPU, a virtual machine has to wait for its turn to process its I/O request after other virtual machines have utilized their allocated CPU time. In scenarios where many virtual machines share the same physical CPU, the delay can become significant, rendering it unacceptable for virtual machines with high I/O intensity.

Traditional schedulers prioritize fairness in scheduling, treating I/O tasks and CPU tasks together. However, this approach can result in inefficient I/O task processing. To address the I/O inefficiency problem, researchers have proposed various optimization methods that discriminate between I/O tasks and CPU tasks. However, these methods often classify virtual machines into two categories: pure I/O operations and pure CPU calculations. They do not consider distributing enough CPU time to process I/O requests for virtual machines with mixed characteristics, which can impact the efficiency of I/O tasks. Additionally, some previous research only allows manual specification of virtual machine types at system startup, which poses limitations when the workload of a virtual machine dynamically changes from I/O-intensive to CPU-intensive. In such cases, the scheduler cannot adjust the scheduling time segment, leading to persistent delays in I/O response.[10,11]

In summary, traditional schedulers face challenges with significant I/O operating lag. Although existing methods have been proposed to reduce the delays to some extent, they lack the ability to dynamically adapt to changes in virtual machine workload characteristics and provide sufficient time to process I/O requests. This limitation ultimately affects the performance of I/O requests. Consequently, there is a need for the development of a scheduling algorithm that considers the work characteristics and task load of virtual machines to address these issues.

Research Objective:

The main objective of this research is to develop a virtual machine dispatching algorithm that considers the task load and work property of VMs. The algorithm aims to optimize the dispatching decisions by reducing I/O request delays, providing sufficient time for I/O request handling, and minimizing additional overhead. The proposed algorithm integrates interruption monitoring, CPU monitoring, and dispatcher modules to achieve these objectives. By modifying the three modules within the virtual machine monitor, the algorithm ensures good applicability and adaptability in various virtualization environments.

Research:

The objective of this invention is to provide a scheduling virtual machine algorithm based on the current characteristics and task load of virtual machines. This algorithm aims to ensure that I/O requests are responded to in a timely manner while providing enough CPU time for processing I/O requests. It also aims to balance the timeslice size to maximize I/O performance while minimizing overhead for CPU-intensive virtual machines.

To achieve this objective, the invention proposes a scheduling virtual machine algorithm consisting of three modules: an interruption monitoring module, a CPU monitor module, and a scheduler module.

The interruption monitoring module operates within the virtual machine monitor and captures interrupt requests received by virtual machines through the event channel. By analyzing the information carried by these interrupt requests, such as domainU send requests, domainU receives requests, domain0 sends replies, and domain0 receives replies, the module monitors the interruption frequency of all virtual machines. This information is used to determine whether each virtual machine has a CPU-intensive or I/O-intensive workload.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 4
***Article Received:**10 February 2023 **Revised:**20 March 2023 **Accepted:**30 March 2023*

The CPU monitor module operates within the virtual machine monitor and monitors the CPU utilization of virtual machines. It calculates the CPU usage and determines the required scheduling timeslice for each virtual machine based on its CPU usage. To ensure stability and reduce performance fluctuations, the CPU monitor module maps CPU busy percentages from 0% to 100% to timeslice intervals ranging from 1ms to 20ms. The timeslice size is modified only when the CPU usage remains within a certain range for several consecutive cycles. The module utilizes an aging algorithm that incorporates historical information within a specific time frame to calculate the CPU usage (Table 1)

Time Stamp	CPU Usage (%)	CPU Temperature (°C)	CPU Frequency (GHz)
0:00:00	10	45	2.3
0:05:00	20	50	2.6
0:10:00	30	55	2.7
0:15:00	40	60	2.9
0:20:00	50	65	3.1
0:25:00	60	70	3.29
0:30:00	70	75	3.48

Table 1: CPU Usage

The scheduler module is a modification of the CREDIT scheduler and incorporates data structures to store feedback from the interruption monitoring module and the CPU monitor module. It maintains two queues: an A queue and a B queue. Based on the feedback from the interruption monitoring module, virtual machines with different workload characteristics are placed in different queues. Virtual machines with CPU-intensive workloads are placed in the A queue, while virtual machines with I/O-intensive workloads are placed in the B queue. The scheduler selects the next virtual machine to run from either the B queue or the A queue in a specific order:

1. The scheduler first selects the next virtual machine from the head of the B queue. After scheduling, the virtual machine is inserted at the tail of the B queue.
2. This step is repeated until all virtual machines in the B queue have been scheduled at least once or the B queue becomes empty.
3. The scheduler then selects the next virtual machine from the head of the A queue.
4. Steps 1 to 3 are repeated until all virtual machines have utilized their allocated timeslices for the current cycle.
5. After each virtual machine is selected to run, the scheduling function updates the domain structure using information from the interruption monitoring module and the CPU monitor module. Based on this information, the scheduling function applies different scheduling strategies to virtual machines with different characteristics. The timeslice size is determined based on the information from the CPU monitor module, and a timer is set accordingly to perform the scheduling of virtual machines.

The proposed scheduling virtual machine algorithm based on virtual machine characteristics and task load offers several technical advantages:

1. The interruption monitoring module captures all events through the event channel, including communication events between domain0 and domainU. It calculates the interruption frequency of each virtual machine, providing the basis for distinguishing between CPU-intensive and I/O-intensive virtual machines. This module provides information about the current characteristics of virtual machines, enabling the scheduler to apply different dispatching algorithms based on the type of workload. CPU-intensive virtual machines utilize the default Credit scheduler, while I/O-intensive virtual machines use the algorithm proposed by the invention.
2. The CPU monitor module accumulates the number of times a virtual machine is busy within a specific time frame. It calculates the CPU usage by dividing the busy ratio by the total activation count. The timeslice required for the scheduling algorithm used by I/O-intensive virtual machines is calculated based on the CPU usage determined by the module. To ensure stability and avoid abrupt changes in timeslice size, the module employs linear mapping and an aging

algorithm that considers historical information, resulting in more accurate calculations.

3. The scheduler module modifies the original Credit scheduler and maintains fairness within a scheduling cycle. It allows CPU-intensive virtual machines to run larger timeslices while enabling I/O-intensive virtual machines to run multiple smaller timeslices. Priority is given to scheduling I/O-intensive virtual machines, reducing the delay in I/O response. The timeslice for I/O-intensive virtual machines is calculated by the CPU monitor module, ensuring sufficient time for processing I/O requests and further enhancing I/O performance. All virtual machines within a scheduling cycle receive equal timeslices, ensuring fairness in scheduling.

   The proposed algorithm is implemented within the virtual machine monitor and does not require modifications to the client computer's code. It is independent of the specific client operating system, making it applicable to all client operating systems with strong adaptability.

   The implementation of the modules within the virtual machine monitor results in minimal overhead and does not significantly impact the performance of CPU-intensive virtual machines. The algorithm allocates shorter timeslices for I/O-intensive virtual machines and utilizes the default timeslice for CPU-intensive virtual machines. This approach minimizes the performance impact on CPU-intensive virtual machines and reduces system overhead.

   In conclusion, the proposed scheduling virtual machine algorithm based on virtual machine characteristics and task load addresses the limitations of previous methods. It ensures timely response to I/O requests, provides sufficient CPU time for I/O processing, balances timeslice sizes, maximizes I/O performance, and minimizes overhead for CPU-intensive virtual machines. The algorithm offers technical advancements through its modules: interruption monitoring, CPU monitoring, and scheduler modifications. It provides a more efficient and adaptable solution for scheduling virtual machines in various workload scenarios.

Conclusion:

In conclusion, this research presents a novel virtual machine dispatching algorithm that considers both the task load and work property of VMs. By integrating interruption monitoring, CPU monitoring, and dispatcher modules, the algorithm improves dispatching decisions and reduces I/O request delays. The algorithm calculates dispatching time segments based on the current task load and historical information, providing sufficient time for I/O request handling while minimizing additional overhead. The modifications within the virtual machine monitor ensure practicality and adaptability in different virtualization environments. Future work can focus on further optimizing the algorithm and conducting extensive performance evaluations to validate its effectiveness in real-world scenarios.

References:

1. Dimitrov, N., & Göçmen, T. (2022). Virtual sensors for wind turbines with machine learning-based time series models. *Wind Energy*, *25*(9), 1626-1645.
2. A container-based technique to improve virtual machine migration in cloud computing, A Bhardwaj, C Rama Krishna - IETE Journal of Research, 2022 - Taylor & Francis
3. Increasing Flexibility of Cloud FPGA Virtualization, J Ruan, Y Chang, K Zhang, K Shi, 2022 - ieeexplore.ieee.org
4. Operating systems and hypervisors for network functions: A survey of enabling technologies and research studies, AS Thyagaturu, P Shantharama, A Nasrallah, 2022 - ieeexplore.ieee.org
5. Direct-Virtio: A New Direct Virtualized I/O Framework for NVMe SSDs, S Kim, H Park, J Choi - Electronics, 2021 - mdpi.com
6. Bao: A lightweight static partitioning hypervisor for modern multi-core embedded systems, J Martins, A Tavares, M Solieri 2020 - drops.dagstuhl.de
7. Optimizing nested virtualization performance using direct virtual hardware, JT Lim, J Nieh - 2020 - dl.acm.org
8. Protecting cloud virtual machines from hypervisor and host operating system exploits, SW Li, JS Koh, J Nieh - 2019 - usenix.org
9. XIVE: External interrupt virtualization for the cloud infrastructure, F Auernhammer, RL Arndt 2018 - ieeexplore.ieee.org
10. ARM virtualization: performance and architectural implications, C Dall, SW Li, JT Lim, J Nieh 2016 - dl.acm.org
11. Embedded hypervisor xvisor: A comparative analysis, A Patel, M Daftedar, M Shala 2015 - ieeexplore.ieee.org :