Optimizing I/O Intensive Domain Handling in Xen Hypervisor for Consolidated Server Environments

Venkataramanan Venkatesh and Amiya Nayak(✉)

School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, Canada
{vvenk090,nayak}@uottawa.ca

Abstract. Consolidation of servers through virtualization, facilitated by the use of hypervisors, allows multiple servers to share a single hardware platform. Xen is a widely preferred hypervisor, mainly, due to its dual virtualization modes, virtual machine migration support and scalability. This paper involves an analysis of the virtual CPU (vCPU) scheduling algorithms in Xen, on the basis of their performance while handling compute intensive or I/O intensive domains in virtualized server environments. Based on this knowledge, the selection of CPU scheduler in a hypervisor can be aligned with the requirements of the hosted applications. We introduce a new credit-based vCPU scheduling strategy, which allows the vCPUs of I/O intensive domains to supersede other vCPUs, in order to favor the reduction of I/O bound domain response times and the subsequent bottleneck in the CPU run queue. The results indicate substantial improvement of I/O handling and fair resource allocation between the host and guest domains.

Keywords: Xen hypervisor · Server consolidation · Virtual machine monitor (VMM) · CPU scheduling

1 Introduction

According to a report [1] by Natural Resources Defense Council (NRDC), the overall data centre efficiency is still at 12 %–18 % with most of the servers remaining idle while drawing precious energy. Proliferation of data stored in the cloud has sprung the construction of a high number of server farms by organizations. This 'server sprawl' problem has been many years in the making, thus serving as the root cause for induction of virtualization into the server environment. Hussain and Habib in [2] have addressed the 'server sprawl' problem by using virtualization-based server consolidation and storage unification through storage area network (SAN). But, the risks that come with virtualization must also be considered to avert a crisis when an unprecedented event occurs, since consolidating servers also means introducing a single point of failure for all the applications running on the hosted virtual machines. Additionally, there is the problem of virtualized legacy applications not being able to deliver near-native performance as that of a dedicated application server.

While the Xen hypervisor [3] addresses some of the post-virtualization issues like security by providing isolation among virtual machines and provisions for adapting different resource allocation schemes to cater to the needs of the hosted applications, a

X. Huang et al. (Eds.): GPC 2016, LNCS 9663, pp. 180–195, 2016.
DOI: 10.1007/978-3-319-39077-2_12

near-native performance for every hosted application remains the ultimate goal of the developers in the Xen open source community. The motivation for this work originates from the reasoning that, since CPU scheduling algorithms are accounted for majority of the factors affecting a hypervisor's performance, this component of the hypervisor's architecture must be subjected to intensive research, so as to arrive at sustainable solutions to performance issues. Understanding the limits of a scheduling algorithm is of paramount importance when consolidating server hardware and this work provides an essential insight into the CPU scheduling characteristics and their impact on VM performance.

1.1 Xen Hypervisor Architecture

The architecture of Xen consists of two elements, namely the hypervisor and the driver domain. The function of the hypervisor layer is to provide abstraction between the guest operating systems and the actual hardware by creating virtual counterparts of the physical hardware components. The driver domain or host domain or Domain 0 ("dom0") is a privileged VM which manages other guest VMs or unprivileged domains or user domains ("domU"), enumerated as Domain 1, Domain 2 and so on. The term driver domain for dom0 is mainly due to its responsibility of coordinating the I/O operations on behalf of the user domains, since the guest domains do not have direct access to the I/O hardware. In initial versions of Xen, the hypervisor managed the I/O operations by creating simple device abstractions for the guest domains to access. This functionality has been transferred to the dom0 and the current methodology of I/O management by Xen works more efficiently, as the host domain handles all of the virtual I/O interrupts. In addition, this delegation of responsibility to dom0, allows the hypervisor to focus more on virtualizing other hardware components such as CPU, disk drives and network interfaces.

1.2 Domain States in Xen

Analogous to processes in operating systems, a domain in Xen can be present in, and transition into or out of, any of the following states:

Paused. The domain's execution is paused and it still consumes allocated resources like memory, but remains ineligible for scheduling.
Saved. A running domain is saved to a state file for it to be restored later, but TCP timeouts can severe live network connections; memory allocated for the domain will be released for other domains to use.
Running. Domain is running or runnable on the CPU.
Blocked. Domain is waiting on an I/O or in idle mode. It is currently neither running nor runnable.
Shutdown. Domain enters this state on a hard shutdown or shutdown command, terminating all the processes of the guest OS.
Terminated. The domain's processes are killed instantly and the domain id is deleted (Fig. 1).

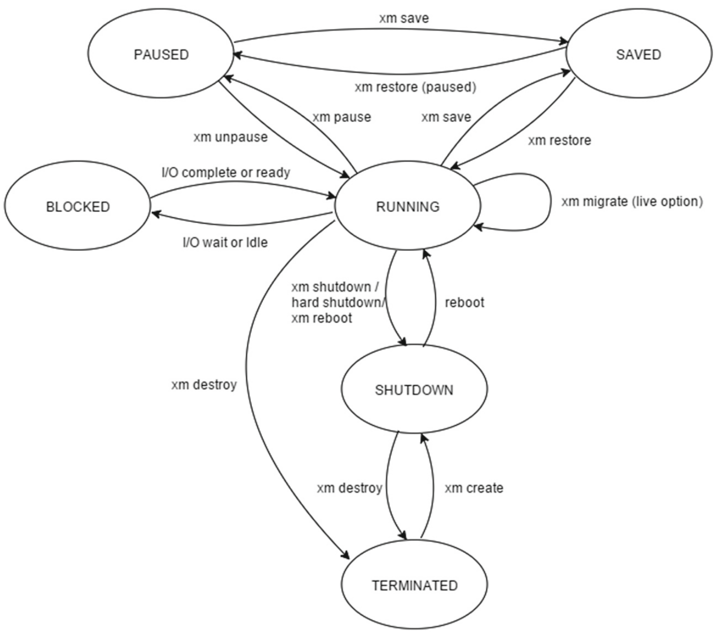

Fig. 1. Domain states in Xen hypervisor showing the five states a running domain can transition into, either manually using ‘xm’ commands or as part of an I/O cycle.

2 CPU Scheduling in Xen

The Xen hypervisor, since its inception, has employed a number of scheduling algorithms namely Borrowed Virtual Time [4], Atropos [5], Simple Earliest Deadline First [6], ARINC 653 [7], Credit and Credit2 [8]. Scheduling parameters of individual domains can be adjusted by management software running in Domain0. After the architectural change brought by Xen 3.0, the new model of I/O operations increased the complexity of CPU resource usage. The earlier algorithms such as Borrowed Virtual Time and Atropos have been phased out due to the I/O structural change and the lack of non-work-conserving mode (NWC) [11], which means that the running domain is able to exceed its CPU share in the absence of competition from other domains.

Borrowed Virtual Time. BVT is a weighted or proportional fair-share scheduler based on the concept of virtual time, dispatching the runnable VM with the earliest effective virtual time first. This concept of virtual time allows latency-sensitive or I/O intensive domains to “warp” [4] back in virtual time by temporarily promoting themselves a higher scheduling priority. The amount of virtual time equivalent which was spent for warping is compensated by depleting the domain’s CPU allocation in the future. Minimum Charging Unit (MCU) [4], which is derived from the frequency of clock interrupts, is used as basic unit of accounting for a running domain. The decision for a context switch is made based on the parameter context switch allowance ‘C’, a multiple of MCU, which denotes the actual time comparable with another VM competing for the CPU. The proportional fair-sharing is implemented by allocating each domain, a share of the CPU based on the weights configured in the domains. Optimal fairness is ensured by BVT in allocation of CPU since the error value cannot exceed the sum of context switch allowance ‘C’ and one ‘MCU’. BVT can operate only in work conserving (WC) mode which means that a domain cannot exceed its CPU share even if a portion of the CPU remains unused by other domains. This is one of the major draw-backs of BVT scheduler which severely restricts CPU utilization and hence, BVT was gradually phased out in the later stages of Xen development.

Atropos. The Atropos scheduler allocates shares on the basis of an application dependent period, determined by the slice ‘s’ and period ‘p’, which together represent the processor bandwidth of that particular domain. Each domain is guaranteed to receive a minimum of ‘s’ ticks of CPU time and if available, several slices of CPU time in each period of length ‘p’. A domain is allocated a slack CPU time, only if it is configured to receive that extra time as indicated by the Boolean parameter ‘x’. To ensure timely servicing of I/O from domains, a latency hint ‘l’ is specified as one of the parameters for differentiating I/O intensive or latency-sensitive domains [5].

The Atropos scheduler provides an additional degree of freedom by enabling the user to control both the absolute share of the domains and the frequency at which they are to be scheduled. Say if the domain’s present state is on a scheduler queue, then based on the deadline ‘d’, which denotes the end of current period ‘p’ for the domain, a value ‘r’ is calculated in real-time and passed to the scheduler as the time remaining for the domain in that current period. Queues ‘Q_r’ and ‘Q_w’ [5], representing the list of ready-to-run and waiting domains respectively, are sorted according to deadline ‘d’, in addition to another queue of blocked domains. The values d_x and p_x denote the deadline and period of the domains at the head of the respective queues. The Atropos scheduler due to its significant overhead and inability to balance loads in a SMP has been replaced by more sophisticated load-balancing algorithms.

ARINC 653. It is based on the standard Real Time Operating System (RTOS) interface for partitioning of computer resources in the time and space domains. The resulting software environment is capable of handling isolated domains of applications, executing independently of each other. The partitioning of memory into sub-ranges for creating multiple application levels also leads to a spatial and temporal isolation of the domains, thus maintaining them under the impression of complete possession of the underlying hardware.

This methodology of domain isolation, wherein each independent block is called a ‘partition’ [7], provides ARINC 653 with the capacity to deterministically schedule the Xen domains. The CPU time is allocated in the form of ‘credits’ in frames which are consumed by the domains. When the domains have reached their credit limit for a given frame they are considered as ‘out of credits’ or ‘expired’ and correspondingly placed in a wait queue called ‘expiry list’. When the credits are available via the next frame the domains can resume executing again. In case, a domain is made to yield the CPU or if a domain is blocked, prior to consumption of the allocated credits in a given major frame, execution for these domains is resumed later in the major frame.

On the event of completion of a major frame, the domains in the expired list are restored to their preset credit allocations. These preset values and other parameters of the ARINC 653 are configurable from within the dom0 or the driver domain. The parameters to be set include run time for each domain and the major frame for the schedule. Calculation of the length of the major frame depends on the condition that all the domains must be able to expend all of their configured credits. Current implementation of the ARINC 653 lacks support for multi-processor environments and hence, preferred mostly for special case usage.

Simple Earliest Deadline First. The SEDF scheduler is a proportional or weighted sharing algorithm which employs real-time information feedback to guarantee CPU time and deal with latency-sensitive domains. The SEDF scheduler may be considered to inherit its structure from the Atropos schedule, from the perspective of parameters and specifications. For example, a combination of the CPU bandwidth parameters denoted by a tuple <Slice (s_i), Period (p_i), Extra-time (x_i)> is used to specify the amount of CPU time requested by the domain. Like Atropos, there is a minimum guarantee of s_i time slices in every period of length p_i and a domain is allocated extra-time, if the Boolean flag x_i is set. This tendency of allowing domains to exceed their CPU limits forms the non-work conserving property of SEDF. More importantly, this slack time which is re-distributed as extra-time on demand, follows a best-effort and fair approach. The slack time is a by-product of under-utilized timeslices and is not borrowed from other domains’ CPU time.

The optimization of fair allocation is impacted to a great extent by the granularity used in defining the tuple of a domain. Deadline d_i and remaining time r_i of a domain in the current period p_i are passed to the scheduler in real-time [6], thereby, facilitating the scheduler to make real-time decisions based on the status of the domains. The feature where SEDF differs from Atropos is that the SEDF scheduler chooses the domain with the earliest d_i, from the waiting queue, to be scheduled next.

Though the response time to latency-sensitive domains is better in SEDF, the lack of load balancing on SMPs proves to be a major drawback. This paper focuses on improving scheduling scheme to support soft real-time workloads in virtualization systems. Some of the latest research in Xen has been focussed towards adapting the hypervisor for deployment in real-time workload environments. In [14], Cheng et al. have proposed an enhanced scheduler named SRT-Xen which focuses on improving the Xen scheduling framework for handling domains with soft real-time tasks. In the paper they have also discussed about a scheduling mechanism which ensures fair and non-detrimental handling of both real-time and non-real-time tasks.

Credit. Credit is a proportional share scheduler in which the automation of global load balancing of virtual CPUs among the physical cores was introduced to Xen Hypervisor.

When a CPU completes execution of the domains assigned to it, the queues of other CPUs are surveyed for any ready-to-run vCPUs. This load balancing, to an extent, prevents CPUs from being overwhelmed and promotes optimized CPU utilization. The queue maintained by each physical CPU is sorted based on the priority of the vCPU and the priority may either be OVER or UNDER [8].

OVER: the initial allocation of the vCPU, based on proportional share scheduling, has been exceeded;
UNDER: the initial allocation for the vCPU is yet to be consumed completely.

Every 10 ms, or for every scheduler tick, the credits of the domain are decremented and when a vCPU has consumed all of its allocated credits, the value of the credit crosses the zero mark and the state of its priority is converted from UNDER to OVER, and prevented from being scheduled. A vCPU or VM has 30 ms before it is pre-empted by another VM, unless it has enough credits to stay UNDER. At the end of every 30 ms, the credit for each domain is replenished and the domains which were in OVER state are allowed to execute again.

In addition to the two priority states, an additional BOOST priority ensures that domains which are awoken after an idle I/O wait are placed ahead in the run queue to reduce latency. The scheduler is triggered when an I/O event occurs in the event channel and the domains with the BOOST priority are fast tracked to the head of the queue. Two parameters govern the credit allocation of a domain: Weight denotes proportion or percentage of CPU to be allocated; Cap denotes limit of extra credit which cannot be exceeded. Value of Cap set at '0', means there is no limit to the amount of CPU time that a domain can exceed. Non-zero value for Cap limits the amount of CPU a domain receives by restricting the extra credits to that particular percentage.

Credit2. The Credit scheduler, besides its favourable characteristics, has a few problems like below average performance in handling I/O intensive domains, lack of fairness in scheduling latency-sensitive or interactive domains (multiuser applications) and hyper-thread dependency issues. The revamped version of the Credit scheduler is Credit2, designed to overcome some of the shortcomings of its predecessor. The Credit scheduler implements fair scheduling based solely on the OVER, UNDER or BOOST priorities and organizing the run queue in a round-robin fashion for domains with equal priority state. The domains which burn through their allocated credits in a single schedule are more likely to be positioned ahead in the queue than those domains which yield the CPU every so often, only to consume a small portion of their credits before pre-emption. This inherent nature of round-robin scheduling combined with a credit based allocation policy, results in the unintentional, but preferential treatment towards compute heavy domains. The Credit2 attempts to solve these issues by removing the three priority states altogether and sorting the CPU run queue based on the credit balance of the domains.

From Xen 4.2 onwards, an option for scheduling rate limit has been added which sets the minimum amount of time, in microseconds, a domain is made to run before being preempted. This value limit ensures that the domains are performing some minimum amount of work rather than waking up only to be scheduled off without burning any credits. Based on the nature of the workload, for instance, a value of 10 ms for compute intensive workloads and a value of 500us or lower for latency sensitive workloads are advised. In addition, a timeslice parameter for varying the vCPU run time, in milliseconds, has also been introduced to meet the latency needs of the domains being executed.

3 Equalizer Scheduler for I/O Intensive Workloads

In case of virtualized high definition video streaming and compute servers, the CPU-intensive domains will occupy the CPU for longer periods since they require more computational time on the processor, ergo starving I/O bound domains. While the I/O bound vCPUs wait for the allocation of CPU the onset of the next accounting period replenishes the credits, thus further enriching the compute intensive vCPUs. Further, the option of scheduler rate limiting to reduce the time quanta to address this problem requires manual intervention at every instance a different workload is received and does not bode well for a consolidated workload environment. In [15], the authors Hongshan Qu et al. have discussed about the importance of analyzing the resource requirements of individual domains to perform predictive scheduling and have proposed a work-load aware scheduling method for heterogeneous VM environments.

With the purpose of enhancing I/O intensive domain performance in a multi-faceted workload setting, we propose a new credit-based vCPU scheduling scheme, where in, the credit remaining for each vCPU after every accounting plays an important role in making the scheduling decision. The starved vCPUs with the most remaining credits are allowed to supersede others to favor the reduction of the I/O bottleneck in the processor run queue. Since most I/O bound vCPUs need only 1 ms to 10 ms to complete their short CPU burst cycles, there will be no adverse effects in CPU-bound domain performance. The priority states of OVER and UNDER are used to determine if the domain is eligible to be scheduled and the scheduling decision is performed based on a combination of scheduling eligibility and credit availability of the domains.

The vCPU scheduling scheme proposed here has the potential to introduce context switch overhead because of additional validations in the scheduling decision path of code. Though some context switch overhead will exist, the benefits of rapid response-time scheduling for I/O bound domains are observed to outweigh the negative aspects of the Equalizer scheduler.

```
  ALGORITHM:
  equalizer_schedule (current time);
  begin
  perform vCPU validation at run queue head
  if
  vCPU`s priority is OVER OR vCPU_credit less than
  max_credit
then
  check other PCPU run queue
      if
        at least one PCPU run queue empty
      then
        Dequeue from current PCPU run queue
        Insert vCPU into PCPU with empty run queue
      else
        deschedule running vCPU
        perform vCPU validation at run queue head
      endif
else
  update max_credit (vCPU_credit)
  return (vCPU); %vCPU to run next%
endif

deschedule running vCPU:
if
  running vCPU is runnable AND run queue non-empty
then
  insert vCPU at rear end of current PCPU run queue
else
      if run queue is empty
      then
        yield pcpu
        set PCPU idle flag
      endif
endif

update max_credit:
if
  passed vCPU_credit greater than zero
then
  set max_credit = vCPU_credit
endif
```

4 Simulation and Performance Analysis

In this section, we describe the benchmarking environment and parameter setting established for comparison between the scheduling algorithms in Xen. The version of Xen hypervisor under study is Xen 4.4.1 which employs Credit, SEDF and Credit2 as the main scheduling algorithms. The hardware specifications of the system, used for conducting the experiments, are as follows: Intel Core i3-370 M processor with two cores of 2.4 GHz each, 64-bit architecture, 3 GB RAM and 320 GB hard disk.

The driver domain ran Ubuntu 13.04 with linux kernel 3.8.0-19 and the user domain ran Ubuntu 12.04 with linux kernel 3.5.0-23 on Xen 4.4.1 stable release. The test suites used to simulate VM workloads in this experiment are as follows:

IBS. The Isolation benchmark suite [9] which incorporates a CPU intensive test is capable of creating simulated workloads by performing a series of integer and floating point calculations. The intense computational operations performed on the CPU resemble compute-intensive applications such as HD video playback or 3D graphics rendering.

IOzone. A file system benchmark [10] which allows performing of multiple Disk-I/O operations and monitor the rate of read/write operations. The sequential and random read/write operations collectively induce stress on the file system.

Netperf. A network monitoring tool [11], aids in measuring low-level network performance by exchanging bulk files with a remote host.

Sysbench. A performance benchmark tool [12], includes an OLTP test for simulating database server transactions from a Mysql database.

4.1 Disk I/O Benchmarking Using IOzone

We employ IOzone test suite, which performs a series [10] of sequential read/write (S-Read/S-Write) and random read/write (R-Read/R-Write) operations, followed by the measurement of the read/write rate in KBytes per second. We choose a combination of 512 Mb file size with 1 Mb record length for avoiding a system freeze, when hosting multiple guests while also keeping in mind, the cumulative operation time of the experiment. In order to avoid files being written or read from the cache memory or RAM, IOzone is set to perform only Direct I/O operations, which ensures that only the physical disk is used in the file operations. Figure 2a shows credit scheduler's dom0 disk I/O performance when dom0 is idle and when dom0 is performing heavy CPU operations facilitated by IBS's CPU intensive test. In the first case, the domU is able to achieve 93 % of dom0's I/O performance and when dom0 is under stress, domU's I/O execution rate degrades by 4 %–5 % of its earlier performance when dom0 was idle.

In Fig. 2b, the disk I/O performance of SEDF scheduler's domU (dom0 idle) is shown. Under the SEDF's scheduling the domU was able to run I/O operations at 97 % of dom0's I/O execution rate. This might seem better than the Credit scheduler's near-native performance but there is a tradeoff in overall I/O performance, where the Credit scheduler has an edge, as the SEDF scheduler lags due to the overhead induced by its real-time processes.

When the dom0 is subjected to heavy CPU operations in the SEDF scheduler, the domU's disk I/O performance is found to be drastically affected. As illustrated in Fig. 2b, the disk I/O performance of domU degrades by 41 %–45 % of its performance when dom0 was idle. Figure 3a shows the disk I/O performance of the Credit2 scheduler; domU, when the other domain is executing compute operations. With Credit2 scheduler the domU was able to achieve from 94 % up to 99 % of dom0's I/O execution rate. These

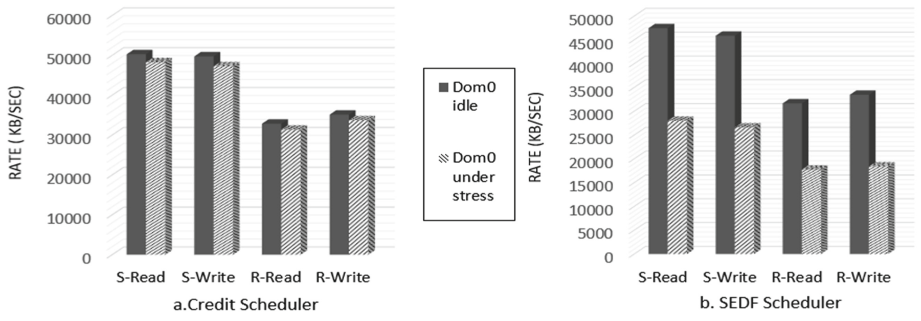

Fig. 2. Evaluation of Disk I/O performance of the Credit and SEDF schedulers using IOzone depicting the guest domain's throughput when host is idle and under computational stress

readings show a marked improvement in handling guest domain I/O requests from Credit scheduler and the overall I/O execution rate is also justifiably better.

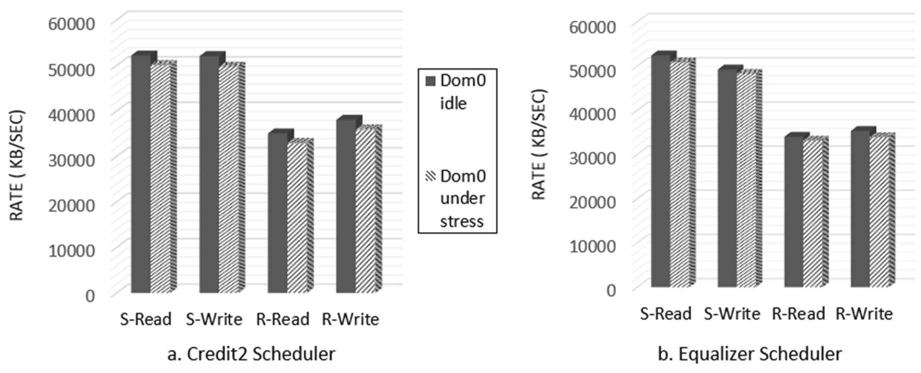

Fig. 3. Evaluation of Disk I/O performance of the Credit2 and Equalizer schedulers

The handling of I/O from domU by dom0 when under stress in Credit2 scheduler is better when compared to Credit and SEDF. As shown by Fig. 3a, the disk I/O performance by domU when dom0 is performing CPU intensive operations, produces results which show that Credit2 scheduler is capable of providing near-native I/O support in a compute intensive environment too. The results show that the domU is capable of restricting the throughput degradation to only 4 %–6 % and given the superior overall I/O execution rate, this fall in read/write rate is only of negligible magnitude. Figure 3b shows that Equalizer scheduler's domU reaches 98 %–99 % near-native disk I/O performance when Dom0 is idle, which exhibits the scheduling fairness among the vCPUs promoted by the new scheduling scheme. A clear improvement is seen over the Credit scheduler that could only reach a maximum of 93 % of near-native disk I/O performance.

The comparison shown in Fig. 4 shows that the Equalizer scheduler has a near- native performance index, which means performance of guest domain as a percentage of the host's performance, is on par with Credit2 and evidently better than the Credit and SEDF scheduler.

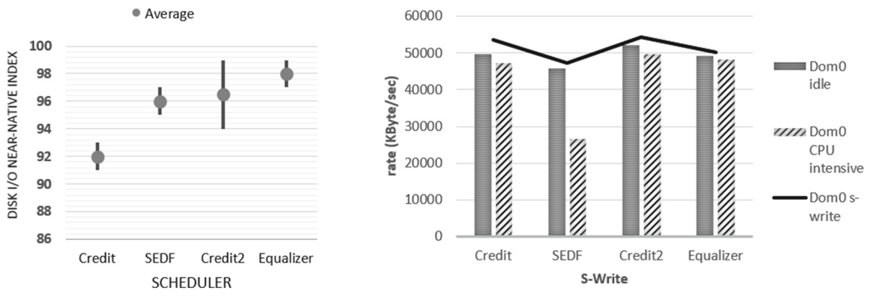

Fig. 4. Comparison of the proposed scheduler with Credit, SEDF and Credit2 schedulers

The Equalizer Scheduler suffers 2 %–4 % degradation in disk I/O performance when dom0 is performing a compute-intensive workload as shown in Fig. 3b. The comparative display of degradation in S-write performance is shown in Fig. 4, wherein the Equalizer scheduler is the least to deviate from its optimal performance followed by Credit2, Credit and finally SEDF.

4.2 Network Performance Benchmarking Using Netperf

The following experiments are conducted by allowing Netperf to run a client on the domU and perform TCP stream tests via bulk file transfers with a remote host connected by a 100 Mbps Ethernet link. The file transfer is done in increments of message size from 1 MB up to 4 MB and the throughput is measured in Mbits per second. The send socket size is 16384 bytes and receive socket size is 87380 bytes.

Figure 5 shows the network performance of domU when dom0 is kept idle and when dom0 is made to perform CPU intensive operations facilitated by IBS CPU stress tests. The results exhibit a clear drop in throughput from an average of 87.85 Mb/s to 72.71 Mb/s. The Credit scheduler has an average performance when the CPU is under contention and an above average performance when CPU is not performing any compute intensive tasks.

When SEDF is put under the same test, the results show that the real-time scheduler fails to handle I/O tasks promptly when the CPU is placed under stress. Figure 5 shows the performance of SEDF's bandwidth intensive test and we can observe a fall in throughput from an average of 89.30 Mb/s to 63.55 Mb/s when CPU is under contention.

The Credit2 scheduler exhibits significant improvement from Credit scheduler in network performance tests. As shown in Fig. 5, the average throughput of 91.05 Mb/s, when dom0 was idle, dropped to an average of 79.15 Mb/s, when CPU is under stress, which is better than the Credit scheduler's 72.71 Mb/s.

The comparative chart for network intensive benchmarking of the four schedulers indicates that the Equalizer scheduler has the minimal performance drop when a network

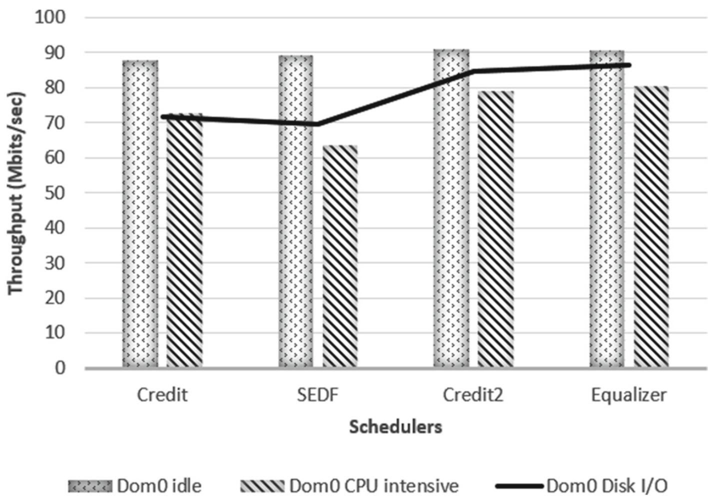

Fig. 5. Comparison of the network throughput of domU measured using Netperf, under different workload states of dom0

I/O intensive domain is competing with a compute-intensive domain or an I/O intensive domain.

4.3 OLTP Database Server Performance Benchmarking Using Sysbench

The performance of the schedulers when hosting database server applications can be measured using Sysbench. As a prelude to simulating the database server we create a Mysql test table with rows of data starting 1000 rows and continue scaling up to 1000000 rows. As illustrated in Fig. 6a, the Credit scheduler's number of transactions/second, which is an important merit for seamless database querying, suffers a performance drop when a user Domain is hosting the database server. And when the host domain is placed under computing stress the OLTP transaction rate further degrades due to the unavailability of the CPU for the server's I/O requests.

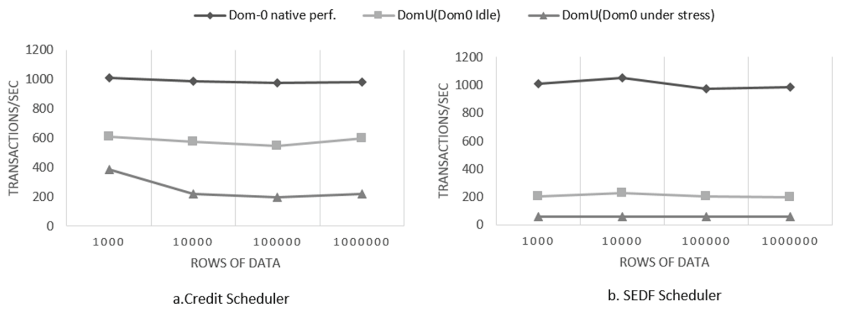

Fig. 6. Rate of transaction in dom0 and domU when simulating OLTP server using Sysbench-Credit and SEDF

Figures 6b and 7a exhibit the database server performance of SEDF and Credit2 schedulers respectively. The SEDF scheduler's domU reduces to almost one-fifth of its native domain's performance and when CPU is in contention an average of 75 % drop in domU's performance is inferred from the benchmarking results. On the other hand the Credit2 scheduler once again comes on top among the three schedulers, by showcasing only minimal loss in the number of transactions executed per second when hosted by the user domain.

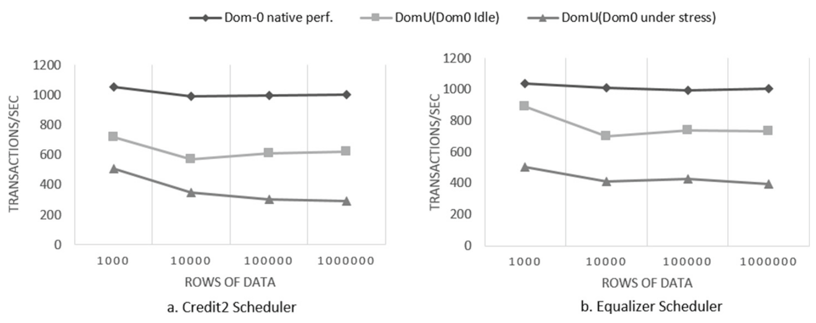

Fig. 7. Rate of transaction in dom0 and domU when simulating OLTP server using Sysbench-Credit2 and Equalizer

The Equalizer scheduler exhibits superior handling of database server requests, shown in Fig. 7b, with the native domain performance similar to that of Credit and Credit2. The loss of number of Transactions per second when domU is handling is lower than Credit2 exhibiting the high I/O throughput rate which is characteristic of the design.

4.4 Consolidated Server Environment

One of the solutions for server sprawl in data centers is the consolidation [13] of hardware, thus promoting resource optimization, while masking the shared server architecture from the users by ensuring negligible or unnoticeable deterioration in QoS. Any hypervisor which is claimed to be a candidate for server virtualization has to be qualified to host multiple and multi-faceted server workloads. While the scale of number of VMs hosted is tied to the hardware's capacity, other aspects such as I/O latency and fairness towards domains can be optimized by choosing the apt scheduler for a given combination of applications.

The consolidated server environment is simulated by hosting two user domains and one of the above discussed performance benchmarks in each of the domains. The experiment is conducted by running IOzone on dom0, Netperf on dom1 and Sysbench Mysql benchmark on dom2, with the same parameter settings as mentioned earlier. The compute intensive benchmark is not preferred for a consolidated environment as there exists ample compute stress on the dom0 because of hosting three VMs and the associated back-end driver processes for virtual interrupt management. Additional compute stress resulted in erratic behavior of the VMs and benchmarking results turned out to be inconsistent and unreliable.

The extent to which the virtual counterpart of a dedicated server has performed can be gauged by using the Domain0 or host's native performance. As shown in Fig. 8, the performance indexing based on dom0 gives a comprehensive picture of the schedulers in a typical server environment.

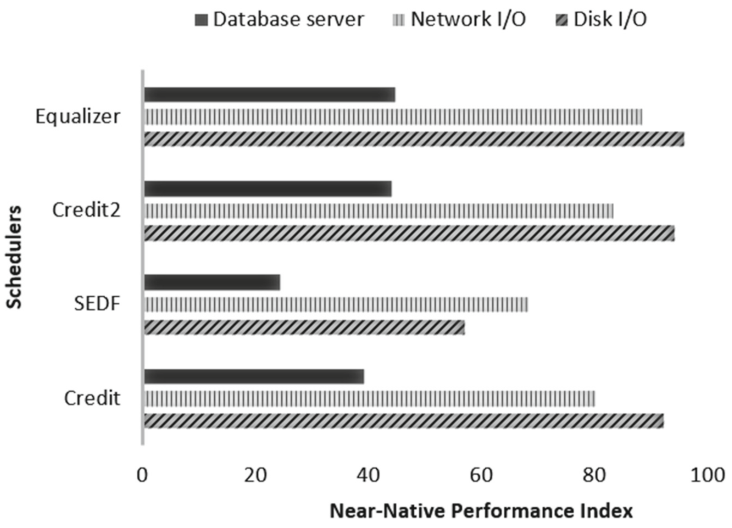

Fig. 8. Consolidated server environment, simulated by hosting two user domains and running IOzone on dom0, Netperf on dom1 and Sysbench Mysql benchmark on dom2

The consolidated workloads generate I/O requests on a frequent basis thus causing an I/O bottleneck, which has evidently caused the setback in the schedulers. The results obtained are not far from the observations made, when the CPU as a resource was under contention. This shows that servicing of I/O requests or interrupts is just as important as CPU allocation. The servicing of Disk I/O has not been interfered by the I/O bottleneck and this suggests the consolidation of disk intensive domains with I/O heavy workload servers such as a webserver.

The experiments provide a glimpse of the Credit2 scheduler's exceptional performance in a high frequency I/O environment and the adept I/O handling capacity which was lacking in Credit scheduler. The results obtained collectively form substantial evidence to the assertion that choice of CPU scheduling algorithm used by Xen hypervisor impacts the performance of the applications hosted on the server. The Equalizer scheduler's handling of I/O heavy domains marginally surpasses the Credit2 scheduler in a consolidated workload experimental setup. As shown in Fig. 8, the proposed scheduler has a slight edge over the Credit2 while performing database transactions and Network intensive I/O handling.

5 Conclusion and Future Work

We have discussed about the server sprawl problem and examined consolidation as a solution, which has shed light on the VM manageability and migration troubles which can rise in the control plane. Xen hypervisor's architecture has been looked upon briefly following the challenges in the VM environment to be expected by a VMM.

The choice of CPU scheduling algorithm used by Xen hypervisor evidently impacts the performance of the applications hosted on the machine. We analysed the CPU scheduling algorithms incorporated into Xen so far, since its earliest version and conducted exhaustive experiments using simulated workloads on the latest batch of schedulers released by the Xen developers' community.

The most significant inference from the experiments is that servicing of I/O by the driver domain depends heavily on Xen's scheduling capacity. By focussing on the I/O handling tendency of each scheduler provided an insight into the right usage of the hypervisor for workload specific applications. Though varying the scheduler parameters could even out some odds in the scheduler's functionality, true scheduling fairness is achieved by optimal performance across all domains without disproportionate usage of resources.

The behaviour of the hypervisor when faced with heavy workloads was discussed and the challenge it poses to the resource usage by the scheduler in charge. Credit and Credit2 have proven to be robust in their handling of heavy I/O workloads such as disk, bandwidth intensive network and database server. Credit2, though in experimental phase has been observed to exhibit and deliver the promised improvement in I/O handling over the Credit scheduler. The older SEDF scheduler has been observed to fall behind in contention with both the Credit scheduler versions. ARINC 653 is a special case scheduler built for uniprocessor architectures and hence cannot be compared with the other schedulers in this version of Xen.

The Equalizer scheduler designed and implemented has been put under different scenarios and benchmarked. The results indicate that the motives of design which are improved handling of compute-intensive domains and to improve effective I/O throughput have been realized although overlooking a small percentage of context switch overhead. Though the performance of the scheduler in idle conditions is not on par with the credit scheduler due to the overhead accumulated because of context switching, the behavior of the scheduler indicates that when competing with compute intensive or I/O intensive domains, every domain is ensured scheduling fairness.

Our future work involves further study on the capabilities of Xen's scheduling algorithms when employed as a hypervisor in collaboration with an open standard cloud computing platform such as Openstack. The live VM migration features of Xen and efficient VM management in a cloud environment seem to be the logical and viable step forward.

Furthermore, the application potentials of Xen as a Network Function Virtualization (NFV) solution for current SDN based networks are to be explored. Another focus area for supplementing the efforts of this research, in terms of green computing, is the study of VM migration strategies for energy efficient data center management.

Acknowledgement. The authors would like to thank the anonymous reviewers for their comments and suggestions. This work is partly supported by NSERC Grant CRDPJ 445731-12.

References

1. Natural Resources Defense Council (NRDC). Data Center Energy Assessment Report (2014). https://www.nrdc.org/energy/files/data-center-efficiency-assessment-IP.pdf
2. Hussain, T., Habib, S.: A redesign methodology for storage management virtualization. In: IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 676–679 (2013)
3. Barham, P., Dragovic, B., Fraser, K., et al.: Xen and the art of virtualization. In: 19th ACM Symposium on Operating Systems Principles, pp. 164–177 (2003)
4. Duda, K.J., Cheriton, D.R.: Borrowed-virtual-time (BVT) scheduling: supporting latency-sensitive threads in a general-purpose scheduler. In: 17th ACM SIGOPS Symposium on Operating Systems Principles, pp. 261–276 (1999)
5. Leslie, I.M., McAuley, D., Black, R., Roscoe, T., Barham, P., Evers, D., Fairbairns, R., Hyden, E.: The design and implementation of an operating system to support distributed multimedia applications. IEEE J. Sel. Areas Commun. **14**(7), 1280–1297 (1996)
6. Jansen, P.G., Mullender, S.J., Havinga, P.J., Scholten, H.: Lightweight EDF scheduling with deadline inheritance. University of Twente (2003). http://doc.utwente.nl/41399/
7. VanderLeest, S.H.: ARINC 653 hypervisor. In: 29th IEEE/AIAA Digital Avionics Systems Conference (DASC), pp. 5.E.2-1–5.E.2-20 (2010)
8. Credit Scheduler. http://wiki.xensource.com/xenwiki/CreditScheduler
9. Isolation Benchmark Suite. http://web2.clarkson.edu/class/cs644/isolation/index.html
10. IOzone Filesystem Benchmark. http://www.iozone.org/
11. IOzone Documentation. http://www.iozone.org/docs/IOzone_msword_98.pdf
12. Netperf. http://www.netperf.org/netperf/training/Netperf.html
13. Sysbench Benchmark Tool. https://dev.mysql.com/downloads/benchmarks.html
14. Uddin, M., Rahman, A.: Energy efficiency and low carbon enabler green IT framework for data centers considering green metrics. Renew. Sustain. Energy Rev. **16**(6), 4078–4094 (2012)
15. Cheng, K., Bai, Y., Wang, R., Ma, Y.: Optimizing Soft real-time scheduling performance for virtual machines with SRT-Xen. In: 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp. 169–178 (2015)
16. Qu, H., Liu, X., Xu, H.: A workload-aware resources scheduling method for virtual machine. Int. J. Grid Distrib. Comput. **8**(1), 247–258 (2015)