REDUCE THE ENERGY CONSUMPTION USING POWER AWARE SCHEDULING IN GREEN COMPUTING

Dr.C.Kumar Charliepaul¹  
Principal  
A.S.L Pauls College of Engg & Tech, Coimbatore.  
charliepaul1970@gmail.com

Felcia Jerlin²  
Assistant professor / CSE  
Dr.G.U POPE College of Engineering, Tuticorin  
erlinfelcia@gmail.com

Abstract -Power consumption is a major challenge in cloud computing. Currently it is estimated that servers consume 0.5% of the world’s total electricity. Server energy demand doubles every 4-6 years. A novel workload-aware scheduling model with two optimizations to eliminate the bottleneck caused by scheduler. In this model, guest domains are divided into I/O-intensive domains and CPU-intensive domains according to their monitored behavior. The network performance is reduced by some problems such as one is that an I/O-intensive domain normally lacks credits to perform I/O and the other is that an I/O-intensive domain is more likely to be inactive at credit allocation time, and may wait longer to handle inputs. This can be solved by two optimizations. The first optimization is shared scheduling, which provides extra credits to I/O-intensive domains from a pool of shared credits. The second is agile credit allocation, which can detect the occasional inactive I/O domain and reserve its credits. ACA adequately adjusts the total number of credits to shorten the waiting time for an I/O domain. To further reduce the energy consumption of a server the greedy algorithm is used.

Keywords - Energy Consumption, Greedy algorithm, guest domain, green computing

I.INTRODUCTION

The SR-IOV networking solution and show by experiment that the current credit scheduler in Xen does not utilize high performance networks efficiently. Hence we propose a novel workload-aware scheduling model with two optimizations to eliminate the bottleneck caused by scheduler. In this model, guest domains are divided into I/O-intensive domains and CPU-intensive domains according to their monitored behavior. I/O-intensive domains can obtain extra credits that CPU-intensive domains are willing to share. In addition, the total number of credits available is adjusted to accelerate the I/O responsiveness. Our experimental evaluations show that the new scheduling models improve bandwidth and reduce response time, by keeping the fairness between I/O-intensive and CPU-intensive domains. This enables virtualization infrastructure to provide cloud computing services more efficiently and predictably.

A workload-aware scheduling model with two optimizations is used to eliminate the bottleneck caused by scheduler. In this model, guest domains are divided into I/O-intensive domains and CPU-intensive domains according to their monitored behavior. I/O-intensive domains can obtain extra credits that CPU-intensive domains are willing to share. In addition, the total number of credits available is adjusted to accelerate the I/O responsiveness.

This framework used to allocate the task based on the workload of cpu’s. This demonstrate the effectiveness in both bandwidth and response time, while keeping the fairness for CPU-intensive domains. These solutions make virtualization more suitable for a wide variety of applications in cloud computing, such as fluent audio playing. This cannot able to reduce the energy consumption of server.

In the proposed system, the technique called power aware scheduling is used to reduce the energy consumption of the server by using green computing. Server energy demand doubles every 4-6 years. Here the green computing is migrated to a cloud computing. Green computing is the environmentally responsible and eco-friendly use of computers and their resources. This power aware scheduling schedules as many VMs at once on a multi-core node. Virtual Machine (VM) is a software artifact that executes other software as if it was running on a physical resource directly.

Typically uses a Hypervisor or VMM which abstracts the hardware from an Operating System. VM scheduling on Multi-core Systems is a nonlinear relationship between the number of processes used and power consumption. Can schedule VMs to take advantage of this relationship in order to conserve power. The power aware scheduling uses a greedy scheduling algorithm. This algorithm used to find the unreachability of a task for reduce the energy consumption of a server.

II.RELATED WORK

Single-root I/O virtualization (SR-IOV) has become the de facto standard of network virtualization in cloud infrastructure. Owing to the high interrupt frequency and heavy cost per interrupt in high-speed network virtualization, the performance of network virtualization is

412
closely correlated to the computing resource allocation policy in Virtual Machine Manager (VMM). Therefore, more sophisticated methods are needed to process irregularity and the high frequency of network interrupts in high-speed network virtualization environment. However, the I/O-intensive and CPU-intensive applications in virtual machines are treated in the same manner since application attributes are transparent to the scheduler in hypervisor, and this unawareness of workload makes virtual systems unable to take full advantage of high performance networks.

The SR-IOV networking solution and show by experiment that the current credit scheduler in Xen does not utilize high performance networks efficiently. Workload-aware scheduling model with two optimizations is proposed to eliminate the bottleneck caused by scheduler. In this model, guest domains are divided into I/O-intensive domains and CPU-intensive domains according to their monitored behavior. I/O-intensive domains can obtain extra credits that CPU-intensive domains are willing to share. In addition, the total number of credits available is adjusted to accelerate the I/O responsiveness.

This experimental evaluations show that the new scheduling models improve bandwidth and reduce response time, by keeping the fairness between I/O-intensive and CPU-intensive domains. This enables virtualization infrastructure to provide cloud computing services more efficiently and predictably.

Along with the developments of networking and virtualization technologies, high speed network connections have become one of the key components in cloud computing and data-centers.

Single-Root I/O Virtualization (SR-IOV) enhances the network throughput to the extent of becoming close to the line rate and achieving high scalability in the 10Gbps and higher network environments. However, the overhead of SR-IOV interrupt virtualization remains significant due to some additional trap-and-emulation overhead on the virtual interrupt controller. The higher the virtualization network connection is, the higher the interrupt frequency becomes through high bandwidth network.

To mitigate this problem, we propose a smart Event-Based Polling model (sEBP), which leverages existing system events to trigger a regular packet polling such that network interrupts are eliminated from the critical I/O paths in the virtual environment. Due to the many varieties of system events, sEBP can deal with the network workload in a configurable and flexible manner. Based on a hierarchical virtualized environment, it can also be implemented either at the guest OS kernel level or at the Virtual Machine Manager (VMM) level. Since polling is much lighter than interrupt processing, sEBP significantly reduces the network processing overhead. The experimental results prove the efficiency of sEBP, which can achieve up to a 59% performance improvement and a 23% improved scalability ratio.

Virtualization is a key technology in cloud computing; it can accommodate numerous guest VMs to provide transparent services, such as live migration, high availability, and rapid checkpointing. Cloud computing using virtualization allows workloads to be deployed and scaled quickly through the rapid provisioning of virtual machines on physical machines. However, I/O virtualization, particularly for networking, suffers from significant performance degradation in the presence of high-speed networking connections. In this paper, we first analyze performance challenges in network I/O virtualization and identify two problems—conventional network I/O virtualization suffers from excessive virtual interrupts to guest VMs, and the back-end driver does not efficiently use the computing resources of underlying multi-core processors. To address these challenges, we propose optimization methods for enhancing the networking performance: 1) Efficient interrupt coalescing for network I/O virtualization and 2) virtual receive-side scaling to effectively leverage multi-core processors. These methods are implemented and evaluated with extensive performance tests on a Xen virtualization platform. Our experimental results confirm that the proposed optimizations can significantly improve network I/O virtualization performance and effectively solve the performance challenges.

The specific demands of high-performance computing (HPC) often mismatch the assumptions and algorithms provided by legacy operating systems (OS) for common workload mixes. While feature-and application-rich OSs allow for flexible and low-cost hardware configurations, rapid development, and flexible testing and debugging, the mismatch comes at the cost of often times significant performance degradation for HPC applications.

The ubiquitous availability of virtualization support in all relevant hardware architectures enables new programming and execution models for HPC applications without losing the comfort and support of existing OS and application environments. Here the trends, motivations, and issues in hardware virtualizations are discussed with emphasis on their value in HPC environments. VMs used for cloud-computing applications will be hosted in data centers; here the VM is migrated within and between data centers. Virtualization is used to access many resources through single physical machine. The high-performance computing is achieved by the virtualization.

It is envisaged that services and applications will migrate to a cloud-computing paradigm where thin-clients on user devices access, over the network, applications hosted in data centers by application service providers. Examples are cloud
based gaming applications and cloud-supported virtual desktops. For good performance and efficiency, it is critical that these services are delivered from locations that are the best for the current (dynamically changing) set of users.

To achieve this, we expect that services will be hosted on virtual machines in interconnected data centers and that these virtual machines will migrate dynamically to locations best suited for the current user population. A basic network infrastructure need then is the ability to migrate virtual machines across multiple networks without losing service continuity. In this paper, a network-virtualization architecture is used to rely on a set of distributed forwarding elements with centralized control (borrowing on several recent proposals in a similar vein). We describe a preliminary prototype system, built using Openflow components, that demonstrates the feasibility of this architecture in enabling seamless migration of virtual machines and in enhancing delivery of cloud-based services.

III. PROJECT DESCRIPTION

1. VIRTUALIZATION

Virtual Machine (VM) is a software artifact that executes other software as if it was running on a physical resource directly. Typically uses a Hypervisor or VMM which abstracts the hardware from an Operating System.

2. DATA CENTER POWER CONSUMPTION

Currently it is estimated that servers consume 0.5% of the world’s total electricity usage. Closer to 1.2% when data center systems are factored into the equation. Server energy demand doubles every 4–6 years. This results in large amounts of CO₂ produced by burning fossil fuels.

3. JOB ALLOCATION

It is the queue, through which request for new virtual machines is entertained. Every new request for virtual machine is added at end of Waiting Queue. Load Balancer will service VM requests in same order as appeared in Waiting Queue. If VM request gets node, then Load Balancer will remove that VM request from Waiting Queue otherwise move further in Waiting Queue and service the queue till end.

4. POWER-AWARE VM SCHEDULING

Currently, there are two competing types of Green scheduling systems for Supercomputers; power-aware and thermal-aware scheduling. In thermal-aware scheduling jobs are scheduled in a manner that minimizes the overall data center temperature. The goal is not always to conserve the energy used to the servers but instead to reduce the energy needed to operate the data center cooling systems. In power-aware scheduling jobs are scheduled to nodes in such a way to minimize the server’s total power. The largest operating cost incurred in a Cloud data center is in operating the servers.

6. VM MANAGEMENT

Another key aspect of a Green Cloud framework is virtual machine image management. By using virtualization technologies within the Cloud, a number of new techniques become possible. Idle physical machines in a Cloud can be dynamically shutdown and restarted to conserve energy during low load situations. which dynamically adds and removes machines form the resource pool. This concept of shutting down unused machines will have no effect on power consumption during peak load as all machines will be running.

7. VM IMAGE ANALYSIS

In order to evaluate the performance of our VM image design, we must create a prototype. There are two paths available to build such a VM OS image. The first is a bottom up approach where a basic Linux kernel is built upon to reach the minimal feature set needed. This requires an entirely new distribution from scratch. Consider a VM which is started on a machine that requires 250 watts of power. Spending 30 seconds on booting up the VM results in 2.08 wh or .002 Kwh of energy used. While this savings of .002Kwh or 30 seconds doesn’t seem like much, its effects are actually quite significant.

IV. ALGORITHM IMPLEMENTATION

Power-Aware Multi-core Scheduling

Currently, there are two competing types of Green scheduling systems for Supercomputers; power-aware and thermal-aware scheduling. In thermal-aware scheduling jobs are scheduled in a manner that minimizes the overall data center temperature. The goal is not always to conserve the energy used to the servers, but instead to reduce the energy needed to operate the data center cooling systems. In power-aware scheduling jobs are scheduled to nodes in such a way to minimize the server's total power. The largest
operating cost incurred in a Cloud data center is in operating the servers. As such, we concentrate on power-aware scheduling in this paper.

Power consumption curve of an Intel Core CPU. The motivation behind power-aware VM scheduling. This graphic documents our recent research findings regarding watts of energy consumed verses the number of processing cores in use. The power consumption curve illustrates that as the number of processing cores increases, the amount of energy used does not increase proportionally. When evaluating using only one processing core, the change in power consumption incurred by using a second processing core is over 20 watts. The change from 7 processing cores to all 8 processing cores results in an increase of only 3.5 watts.

The impact of this finding is substantial. In a normal round robin VM scheduling system like the one in Eucalyptus, the load of VMs is distributed evenly to all servers within the data center. While this may be a fair scheduler, in practice it is very inefficient. The result is that each time the scheduler distributes VMs to a processor, the power consumption increases by its greatest potential. In contrast, this research demonstrates that if the scheduler distributes the VMs with the intent to fully utilize all processing cores within each node, the power consumption is decreased dramatically. Therefore, there is a large need for an advanced scheduling algorithm which incorporates the findings. To meet this need we propose a new greedy-based algorithm to minimize power consumption.

Virtual Machine Management Another key aspect of a Green Cloud framework is virtual machine image management. By using virtualization technologies within the Cloud, a number of new techniques become possible. Idle physical machines in a Cloud can be dynamically shut down and restarted to conserve energy during low load situations. A similar concept was achieved in Grid systems though the use of the Condor Glide-In add-on to Condor, which dynamically adds and removes machines from the resource pool. This concept of shutting down unused machines will have no effect on power consumption during peak load as all machines will be running. However in practice Clouds almost never run at full capacity as this could result in a degradation of the QoS. Therefore by design, fast dynamic shut down and startup of physical machines could have a drastic impact on power consumption, depending on the load of the Cloud at any given point in time.

The use of live migration features within Cloud systems is a recent concept. Live migration is presently used for proactive fault tolerance by seamlessly moving VMs away from failing hardware to stable hardware without the user noticing a change in a virtualized environment. Live migration can be applied to Green computing in order to migrate away machines. VMs can be shifted from low load to medium load servers when needed. Low load servers are subsequently shutdown when all VMs have migrated away, thus conserving the energy required to run the low load idle servers. When using live migration, the user is completely unaware of a change and there is only a 60 to 300ms delay, which is acceptable by most standards. This process of dynamically allocating and reallocating physical machines is complementary to our scheduling system outlines in Algorithm. As the scheduling algorithm executes, it will leave a number of machines idling, potentially for long periods of time. At this point these machines shut down these machines when they are not in use. When load increases, we use Wake on LAN (WOL) to start them back up.

Power-aware Scheduling:

Schedule as many VMs at once on a multi-core node.
- Greedy scheduling algorithm.
- Keep track of cores on a given node
- Match vm requirements with node capacity.

Greedy algorithm are used in power aware scheduling. Trade-off algorithm of performance and power-aware allocation of virtual machines tries to map as much as possible leases onto active hosts (i.e. the physical hosts have any assigned lease) and concern on minimize the number of active physical hosts. This algorithm assumes that an advanced reservation lease preempts all other best-effort leases.

Our customized host selection policy named MAP differs from existing host selection policy of Haizea in ranking formula. Our ranking formula prefers to active hosts to passive (i.e) idle (passive) host is always in the end of ordered list (the host’s score is negative number).

The MAP-H2L host selection policy is similar to the MAP, but in contrast to the MAP, the MAP-H2L sorts list of physical hosts by highest workload to lowest workload. First-Fit mapping differs the original greedy mapping algorithm in two points:
1. Identify the sort list of physical hosts by using the above MAP (or MAP-H2L) host selection policy;
2. Choose the first matching active host $h$\{h0, h1,...,hk\} ($k<0<k≤n$) that can fit at least full requested resources of one node in node set (nodes[\ell] VMs) of the lease \ell. We assigned up to maximum of the physical host $h$.

Presented their work on power estimation that implies that peak power of CPU component is linear e.g., one CPU peak power is 40 Watt, two CPU peak power is 80 Watt. They assume that CPU utilization as main factor in their power model and present an approximation for total system power against CPU utilization (u) a relaxed linear model. With full (100%) CPU utilization (u=1), the power of the system $P_{bus}$ is called $P_{max}$ (Watt), give k is ratio on power of the
idle system against full utilization system. Assume that power is calculated with k=0.7, in which a physical host is just 30 percent in different in power of in free CPU utilization and in full (100%) CPU utilization. Therefore, power of a physical host (P) is approximately by CPU utilization (u) is calculated. The power is calculated by time t of the physical host

\[ P(t) = ku(t) \]

In which 0≤u(t)≤1: u(t) is ratio of number of working CPU cores and total CPU cores on a physical host. Assume that u(t) is unchanged between two events t1 and t2 of VM scheduler. Then the energy consumption calculates as in. Assume that scheduler providing CPU utilization for user lease is steady on during time.

**Virtual Machine Images**

While scheduling and management of virtual machines within a private Cloud environment is important, one must realize what is actually being scheduled. In a normal Cloud environment like the Amazon's EC2, full Operating System VMs are scheduled, often to carry out specific tasks in mass. These VM instances contain much more than they need to in order to support a wide variety of hardware software and varying user tasks. While this is ideal for a desktop based environment it leads to wasted time and energy in a server based solution. A hypervisor provides the same virtualized hardware to each VM and each VM is typically designed for a specific task. In essence, we want the OS within the VM to act only as a light wrapper which supports a few specific but refined tasks or services, and not an entire desktop/application suite. In order to accomplish this task, we need to concentrate on two areas: VM image size and boot time. Normal x86 hardware can vary widely, so most modern operating systems are able to detect various hardware and load modules on the fly upon startup.

It is common for bootstrap to spend 15 seconds running mod probe to load only a single module. This is not an issue with a virtual machine environment since the hardware is standardized and known in advance. The modules in the system and many of the time consuming probing functions can be reduced upon boot up within a VM environment. Considerable amount of time is saved by changing the IDE delay times for probing new hardware. Another technique for reducing the boot time is to orchestrate the boot sequence in a more efficient way. Often many daemons and applications are loaded for general use which (in the case of a lightweight VM instance) aren't needed and can be removed. This includes standalone server applications like Window managers and the X11 windowing system. This would also remove the system's disk footprint considerably saving valuable hard drive space in distributed le systems as well as network traffic when migrating the machines. Boot time can be further improved by creating a new order which maximizes both the CPU utilization and I/O throughput. The use of boot chart can probe where boot up system inefficiencies occur and to allow for optimization of the boot sequence. Another useful tool is read ahead. Read ahead probes the system startup sequence and uses file pre-fetching techniques to load les into memory before they are requested. Therefore an application reads directly from system memory and does not have to wait for disk seek-time.

**Virtual Machine Image Analysis**

In order to evaluate the performance of our VM image design must create a prototype. There are two paths available to build such a VM OS image. The first is a bottom up approach where a basic Linux kernel is built upon to reach the minimal feature set needed. This requires developing an entirely new distribution from scratch. While this may be the cleanest\(^*\) way, it would require a large development team and is therefore infeasible for this project. The other option involves a top-down approach of taking a common distribution and removing certain components from it, making for a lighter and faster sub-distribution. This route is more practical as it does not require reinventing the wheel and the option to keep components such as a package management system and a large distribution library are maintained. Following the second approach, a custom Linux image was created to illustrate the possibility of a fast and lightweight VM OS. Starting with Ubuntu Linux version 9.04, all unnecessary packages were removed, including the Gnome window manager and X11. By removing this multitude of packages, the system image is reduced from 4Gb to only 636Mb.

This minimization speeds up migration of the image from one server to another as there is less network traffic during the movement phase. A number of other packages, libraries and boot level daemons were also removed from the startup process. At the final stage, the image is a minimal Linux installation with only absolutely necessity components. One thing that was left in was the Synaptic package management system, so if any tools or libraries are needed it is a trivial process to have them installed on the system. While the package management system does take up some room, it is well worth the extendibility it provides to the system. A number of kernel modules were also removed from the 2.6.28-11 kernel to speed up the kernel in it and mod probe processes as much as possible. To test the speed of the custom image, both it and a basic Ubuntu 9.04 installation were moved to a VMware server with 2.5Ghz Intel Core 2 Duo and 4GB of ram. The standard Ubuntu image booted in 38 seconds. With our custom VM image, boot time was reduced dramatically to just 8 seconds. By comparing the boot charts in figures 1.18 and 1.19, we can see there is a drastic change in boot time, resulting in a boot time decrease of 30 seconds. Instead of a large amount of I/O blocking, all disk I/O is done at once towards the beginning, allowing for much higher utilization of the CPU. While a boot time of 8 seconds is a considerable improvement, we can do better. The kernel still takes a full
2 seconds to load; with some additional improvements a second or more could possibly be saved.

Management

Monitor Cloud usage and load. When load decreases, Live migrate VMs to more utilized nodes. Shutdown unused nodes. When load increases. Use WOL to start up waiting nodes. Schedule new VMs to new nodes.

Energy Savings

Reduced boot times from 38 seconds to just 8 seconds. 30 seconds @ 250Watts is 2.08whr or .002kwh. In a small Cloud where 100 images are created every hour. Saves .2kwh of operation @ 15.2c per kwh. At 15.2c per kwh this saves $262.65 every year. In a production Cloud where 1000 images are created every minute.Saves 120kwh less every hour. At 15.2c per kwh this saves over 1 million dollars every year. Image size from 4GB to 635MB.Reduces time to perform live-migration.Can do better.

Power scheduling needs

There are many aspect to get energy savings, such as shut down the disks when they are not be used. Here, we will focus on the how to get energy saving from the processor, the CPU. The performance of microprocessor has been improved by consume much more energy. Here is some data. There are 87 million CPUs running and require 160 MW; while there are 500M CPU but require 9,000 MW. So, imagine what will happen if there are 1 billion CPUs in the near future, the energy consumption will blow up. I think everybody know the energy crisis in CA. There are lots of data centers in CA, and every data center host bunch of servers which consume lots of energy. It is said that 25% of the cost of such data center will devote to electricity. Luckily, with the technology advancements, people have found that the power consumption of CPU is dominated by the dynamic power dissipation, which is related to the processor speed and square of the supply voltage. Since, the speed is almost linear related to supply voltage, it is not hard to see that the power is almost the cube of the processor speed, and the energy consumption of specific task is related to square of the speed. Based on these knowledge, people has developed several microprocessor that have the dynamic voltage scaling ability, that is, we can dynamically change the speed of such processors.

2. Heating : complex servers (multiprocessors)
3. Power Aware: maintain QoS, reduce energy

Simulation

In our simulations consider the following schemes. No power management, static power management; greedy slack sharing (one for independent task, and one for dependent tasks); static speculation; adaptive speculation and clairvoyance. The last one is only achievable in post-analysis when actual runtime of each know is known. In the experiment change the number of speed levels, load, alpha, minimal speed and overhead. Here, load is defined as the worst case finish time over deadline, which indicate the amount of static slack in the systems. Alpha is the task average run time over its WECT, which indicate the amount of dynamic slack in the systems.

Schemes Considered

1. NPM: No power management (BASELINE) idle = 5%*Pmax
2. GSS: greedy slack stealing
3. SS: static speculation with greedy
4. AS: adaptive speculation with greedy

Parameters

1. Number of processors
2. Load: worst case time over deadline (global static slack L0)
3. Alpha: task average run time over WCET (dynamic slack)
4. Overhead: 5us/change

V.CONCLUSION

Cloud computing is an emerging topic in Distributed Systems. Need to conserve energy wherever possible. Green Cloud Framework Power-aware scheduling of VMs. Advanced VM & infrastructure management, Specialized VM Image. Small energy savings result in a large impact. Combining a number of different methods together can have a larger impact then when implemented separately.

REFERENCES

[1]. Haibing Guan, Ruhui Ma, and Jian Li “Workload Aware Credit Scheduler for Improving Network I/O Performance in Virtualization Environment” IEEE transactions on cloud computing. vol. 2, no.2 [2014].


Authors Biography:

Kumar Charlie Paul, Principal of A.S.L Pauls College of Engineering & Technology. Had did many National and International Conferences and published many papers in journals. He also guided many students for their Ph.D project works. Having more than 23 years of experience in teaching field.

Felcia Jerlin, Assistant Professor / CSE of Dr.G.U Pope College of Engineering. Doing Ph.d. Had did many National and International Conferences and published many papers in journals. She also guided many students for their UG and PG project works.