Journal of Computer Science 10 (1): 143-156, 2014
ISSN: 1549-3636
© 2014 Science Publications
doi:10.3844/jcssp.2014.143.156 Published Online 10 (1) 2014 (http://www.thescipub.com/jcs.toc)

ENERGY EFFICIENT AND INTERFERENCE AWARE PROVISIONING IN VIRTUALIZED SERVER CLUSTER ENVIRONMENT

[1,2]Mohan Raj Velayudhan Kumar and [2]Shriram Raghunathan

[1]Tata Research Development and Design Centre (TRDDC), Tata Consultancy Services Ltd., Pune, India
[2]Department of Computer Science and Engineering, BS Abdur Rahman University, Chennai, India

Received 2013-09-15, Revised 2013-09-18; Accepted 2013-11-12

ABSTRACT

IT service providers, employ server virtualization as a main building block to improve system utilization, improve system manageability, reduce operational costs which includes energy consumption driving economies of scale with shared resources. Virtualization enables co-locating and efficient assignments of virtual servers within the bounds of limited number of heterogeneous physical servers, with Virtual Machines (VM) sharing the limited physical server resources between them. Though virtualization technologies point to the fact that each virtual server has its very own isolated environment, but in reality, perfect isolation is not possible. Primary measure to achieve assignment efficiency is to ensure that system resources are utilized effectively and performance of VM (and application workloads) is consistent within the desired bounds. Interference or contention on the limited shared resources among VMs leads to performance degradation and is referred to as performance interference. This affects (a) application Quality of Service (QOS) and (b) server cluster or data centers' energy-efficiency. In this work, we analyze the performance degradation using (a) energy efficiency heterogeneity measure and (b) interference aware measure, with the aim to reduce energy consumption in our environment. Experimental results on different scenarios with our energy efficiency and interference aware approach shows a reduction in energy consumption to the tune of 8 to 58% and 10× improvement in per request average response time in contrast to a default energy efficiency and interference oblivious approach.

Keywords: Energy Efficiency, Performance Interference Aware, Server Heterogeneity Aware

1. INTRODUCTION

Main goal of a server cluster environment or Data Centers' (DC) is to satisfy resource needs like processing, storage, memory, network resource capacities from an users' perspective; and be financially viable from Data Center Owners' (DCO) perspective. DCOs employ server virtualization as one of the building block to increase cost effectiveness. Economic benefits from server virtualization come from higher resource utilization, reduced maintenance and operational costs including energy consumption. Although, advanced hardware technology has improved the performance per hardware dollar cost, whereas, server power efficiency or performance per watt used has remained roughly flat over time (Goiri *et al*., 2013). As a result, the electricity consumption cost of servers in data centers will be more than the hardware cost and has become a major contributor to Total Cost of Ownership (TCO). Power consumption is one of the major concern that a DCO need to reduce. Data center power consumption has increased 400% over the last decade (Qian and Medhi, 2011). From DC owners' perspective, it is very important to answer the following question: "How to satisfy user needs (performance criteria) and still minimize power consumption.

Corresponding Author: Mohan Raj Velayudhan Kumar, Tata Research Development and Design Centre (TRDDC), Tata Consultancy Services Ltd., Pune, India

In this study, we focus on server heterogeneity and resource contention aspects to reduce energy consumption in a virtualized server cluster environment or data centre.

DCOs normally have servers which heterogeneous server types. Workloads serviced by sub-optimal servers could be detrimental to DCO by increasing the energy consumption and to the user by increasing the request response-times. Each server configuration invariably exhibits a particular performance to power characteristics.

With virtualization, many VMs can be consolidated into a server. Though these collocated VMs are supposed to exhibit performance isolation characteristics, but in reality these VMs share resources like cache, network interconnects, between them. This use of shared resources causes contention or interference between the collocated VMs and thereby leads to application or service performance degradation. Due to the contention, a collocated VM in the server could exhibit performance degradation or this VM could degrade performance of application hosted in the other collocated VMs. There is a need to understand interference or resource contention impacts with respect to application performance and energy consumption. In this study, we have not considered interference of VMs due to shared caches.

Here, we breifly discuss on earlier works by researchers on (a) performance impacts due to resource contention of VMs collocated in a server (Interference) and (b) energy efficiency improvement of heterogeneous server cluster environment (Heterogenity).

On Interference: Govindan *et al*. (2011) studied the Low Level Cache (LLC) interference by proposing simulated cache using synthetic cache loader benchmarks to profile the performance of applications. Chiang and Huang (2011) proposed TRACON, using modeling and control theory and machine learning techniques. Pu *et al*. (2010) present an analysis of performance interference in virtualized environments with a focus on contentions in input-output storage device usage. Blagodurov *et al.* (2013) proposed an approach which improves server utilization while meeting the SLAs of critical workloads by prioritizing resource access using Linux cgroup weights. Novakovic *et al.* (2013) proposed an approach called DeepDive. This approach identifies VMs with performance degradation due to resource contention by using performance counter metrics and arriving at interference factor by comparing the VM run with a sandbox run of the same application. It also identifies the exact resource types which cause the degradation and provides options to identify and differentiate false positives and false negatives from proper interference occurrences. Though the authors have reported that the overhead in creating a separate sandbox environment to confirm occurrences of interferences is within limits, we believe that a careful relook would be critical. Delimitrou and Kozyrakis (2013) proposed an approach called Paragon, which uses analytical methods leveraging system information from the past runs. It uses minimal training data and a collaborative filtering technique to classify the workload with respect to application and platform interferences in using shared resources. Mukherjee *et al*. (2013) proposed a probe based approach to identify and pin point occurrence of interferences in an virtualized single server environment. Moreover, this work makes an interesting point that identifying interference using probe based approach is accurate, when most of the above other works, at some level use performance counters to identify occurrences of interferences. Adoption of probe based approach in a realistic environment is still to be tested with seemingly high overhead to account for.

In our study, we have adopted an offline approach to build interference degradation factor matrix and use the value while selecting the best server to process the workload.

On Heterogenity: Delimitrou and Kozyrakis (2013) as part of their work called Paragon have also considered using Netflix like collaborative filtering technique to select the best server configuration amongst the heterogeneous servers in sample space. Moreno *et al*. (2013) proposed an approach to compute energy efficiency and select the best server from amongst the available heterogeneous servers in the data center. Our work with energy efficiency is close to the work done by Moreno. Our focus is more on to improve performance and optimally reduce energy consumption of the entire virtualized server cluster environment with an integrated scheme using DPM and DVFS techniques.

We make the following contributions:

- Propose a heterogeneous energy efficient server selection approach
- Propose an approach to account for resource contention or interference impact on workload performance in using collocated virtualized servers
- Integrate the approach with Dynamic Voltage Frequency (DVFS) and Dynamic Power Management (DPM) to achieve performance and energy consumption optimality

The rest of the study is organized as follows. In Section 2, we discuss important metrices used and the proposed solution methodology. In section 3 and 4, we discuss the results obtained. In section 5, we summarize our work with scope for future activities.

2. MATERIALS AND METHODS

We briefly discuss preliminary metrics required and approach methodology adopted in our work.

2.1. Energy Efficiency

We consider a server cluster with Servers from amongst H heterogeneous distinct host types or configurations. Each of these host types have unique performance and power characteristics:

a) HT(k) is the Host type vector represented as a tuple $\{[P^{(i,k,j)}_{(util\%)}], MIPS(k), Cores(k)\}$; where $k \in \{1 .. H\}$;

S^k_i	i^{th} Server in k^{th} host type; where $i \in \{1 .. H_k\}$; $k \in \{1 .. H\}$; H_k is the total servers of host type k
util%	is the CPU core utilization Equation (1) $= \frac{usedMIPS \times 100}{TotalMIPS}$ (1)
usedMIPS	processor core MIPS used by active run-time requests processed by the server core = TotalMIPS – AvailableMIPS
TotalMIPS	Total processor core MIPS = Cores(k) × MIPS(k)
AvailableMIPS	free processor core MIPS
$P^{(i,k,j)}_{(util\%)}$	Power consumption at CPU util% of server S^k_i of host type k and frequency $f^{(k)}_j$ where $k \in \{1 .. H\}$; $j \in \{1 .. F_k\}$; F_k is the maximum frequency for the host type k; $util\% \in \{0 .. 100\}$; $i \in \{1 .. H^k\}$;
MIPS(k)	TotalMIPS of CPU of a Server of host type k
Cores(k)	number of cores of a Server of host type k

b) Server S^k_i with the best energy efficient metric ratio (EE) in a Server cluster with $i \in \{1 .. H^k\}$; $k \in \{1 .. H\}$, from amongst H heterogeneous host types is computed as follows:

EE^k_i	Energy Efficiency metric of server Equation (2 and 3) $S^k_i = \frac{Cores(k) \times MIPS(k)}{P^{(i,k,j)}_{0\%}}$ (2)
Server S^k_i with best EE metric	$= Max(EE^k_i)$; where $\forall i \in \{1 .. H_k\}$; $\forall k \in \{1 .. H\}$; (3)

Reason to consider server power consumption at idle state (cpu utilization at 0%) is to account for high contribution of static power to the total server power consumption. We can further extend Equation (2) to consider power consumption at different cpu utilization%. Considering this change would give a better efficiency value compared to the efficiency using idle power. But the issue with this is that cpu utilization% change is highly dynamic and is highly dependent on workloads. With workloads with small job-lengths, going with cpu utilization% would be a risky with efficiency value changing often. We did test the scenario with EE with in-process cpu utilization% as well as EE at cpu utilization% = 0. We could not see much of a difference in the results between the scenarios. Hence, we have taken a conservative approach to go with energy efficiency using idle power consumption as in Equation (2). Also, another point to note is that performance degradation factor due to interference or resource contention varies with cpu utilization.

2.2. Power Model

In this study, we consider physical server power consumption of different server configurations at different power states as in **Table 1**.

$P^{(i,k,j)}_{(0\%)}$ is the server power consumption when no application is running in the server S^k_i, also known as idle power (cpu utilization is 0%) of the server with host type k and frequency $f^{(k)}_j$. Where $k \in \{1 .. H\}$; $i \in \{1 .. H_k\}$; $j \in \{1 .. F_k\}$, F_k is the maximum frequency for the host type k; H_k is servers of host type k; $P^{(i,k,j)}_{(100\%)}$ is the server power consumption when the server's CPU utilization% = 100.

We use the below model to calculate server power consumption when utilization% which is not covered in SPEC results Equation (4) (SPEC Benchmarks, 2013):

$$P^{i,k,j}_{util\%} = P^{i,k,j}_{\lfloor util\% \rfloor} + \Delta \times \frac{(\lceil util\% \rceil - \lfloor util\% \rfloor)}{100} \quad (4)$$
$$\Delta = P^{i,k,j}_{\lceil util\% \rceil} - P^{i,k,j}_{\lfloor util\% \rfloor}$$

Table 1. Physical server state power consumption (in Watts) (SPEC Workloads)

Host type k Power	(1)	(2)	(3)	(4)	(5)	(6)	(7)
$PM^{(i,k,j)}$	169	117	135	113.0	113.0	247	222.0
$Pm^{(i,k,j)}$	105	86	93	41.6	42.3	67	58.4
$P^{(i,k,j)}_{SETUP}$	169	117	135	113.0	113.0	247	222.0
$P^{(i,k,j)}_{OFF}$	0	0	0	0.0	0.0	0	0.0

(1)HpProLiantMl110G3PentiumD930, (2) HpProLiantMl110G4Xeon3040, (3) HpProLiantMl110G5Xeon3075, (4) IbmX3250XeonX3470, (5) IbmX3250XeonX3480 (6)IbmX3550XeonX5670, (7) IbmX3550XeonX5675

2.3. Energy Model

Our energy model accounts for power consumed when in operational (DVFS-ACTIVE, IDLE) power states and also non-zero power consumed in non-operational (DPM-OFF, SETUP) power states:

S^k_i: i^{th} Server in k^{th} host type; where $i \in \{1 .. H_k\}$; $k \in \{1 .. H\}$; H_k is the total servers of host type k

E^k_i : Total Energy consumed by the server:

$$S^k_i = 0 \times T^{i,k,j}_{OFF} + P^{i,k,j}_{SETUP} \times T^{i,k,j}_{SETUP} + P^{i,k,j}_{util\%} \times T^{i,k,j}_{util\%}$$

E: Server cluster Total Energy consumption Equation (5):

$$= \sum_{i=1}^{H_k} \sum_{k=1}^{H} E^k_i \quad (5)$$

P^k_i : Total Power consumed by the server:

$$S^k_i = \frac{E^k_i}{T^{i,k,j}_{OFF} + T^{i,k,j}_{SETUP} + T^{i,k,j}_{util\%}}$$

P: Server cluster Total Power consumption Equation (6):

$$= \sum_{i=1}^{H_k} \sum_{k=1}^{H} P^k_i \quad (6)$$

where $k \in \{1 .. H\}$; $j \in \{1 .. F_k\}$; $i \in \{1 .. H_k\}$; F_k is the maximum frequency for the host type k

$P^{i,k,j}_{SLEEP}$: Power consumed by server S^k_i in SLEEP mode

$P^{i,k,j}_{SETUP}$: Power consumed by server S^k_i in SETUP mode

$T^{i,k,j}_{SETUP}$: Time duration of server S^k_i in SETUP mode

$P^{i,k,j}_{util\%}$: Power consumed by server S^k_i in BUSY or IDLE mode; host type k and operating at a particular CPU utilization% and frequency $f^{(k)}_j$

$T^{i,k,j}_{util\%}$: Time duration of server S^k_i in BUSY or IDLE mode; host type k and operating at a particular CPU utilization% and frequency $f^{(k)}_j$

2.4. Performance Model

We use Operations Per Second (OPS) values from SPEC result for the host type when in operational/active power state. We consider the host type with Optimal energy efficiency (Equation 10) as the reference host type in arriving at the relative response time improvement factor of a server ($RelOPS^{i,k,j}_{util\%}$) at a particular utilization% and frequency. $RefOPS^{k,j}_{(util\%index)}$-OPS consumed by the optimal PPM host type k at a particular CPU util%index and frequency $f^{(k)}_j$. Optimal host type server is derived using Equation 10; where $k \in \{1 .. H\}$; $util\%index \in \{1 .. 11\}$; $j \in \{1.. F_k\}$:

T_{AVG}: Mean request response time (in seconds or milliseconds) for requests that complete during the course of the trace.

$OPS^{k,j}_{(util\%index)}$: OPS consumed by a host type k at a particular cpu util%index and frequency $f^{(k)}_j$.

We use cpu util%index instead of cpu utilization% to help depict the computations more formally

$RelOPS^{k,j}_{util\%index}$: Relative response time improvement factor for host type k at util%index Equation (7)

$$= \frac{OPS^{k,j}_{util\%index}}{RefOPS^{k,j}_{util\%index}} \times \frac{RefOPS^{k,j}_{utilindex} - RefOPS^{k,j}_{util\%index-1(n-1)x^2}}{RefOPS^{k,j}_{util\%index-1}} \quad (7)$$

Where util%index > 2

$$= \frac{OPS^{k,j}_{util\%index}}{RefOPS^{k,j}_{util\%index}} \times 1$$

Where $util\%index \in \{1..2\}$

$RelOPS^{i,k,j}_{util\%}$: Relative response time improvement factor at CPU utilization% of server S^k_i of host type k and frequency $f^{(k)}_j$

where $k \in \{1 .. H\}$; $j \in \{1 .. F_k\}$; F_k is the maximum frequency for the host type k; $util\% \in \{0 .. 100\}$; $i \in \{1 .. H_k)$ Equation (8):

$$= \text{RelOPS}^{i,k,j}_{\lceil \text{util\%} \rceil} + \Delta \times \frac{\left(\lfloor \text{util\%} \rfloor - \lceil \text{util\%} \rceil\right)}{100} \quad (8)$$

$$\Delta = \text{RelOPS}^{i,k,j}_{\lfloor \text{util\%} \rfloor} - \text{RelOPS}^{i,k,j}_{\lceil \text{util\%} \rceil}$$

2.5. Interference Factor Matrix

We consider a server cluster with Servers from amongst H heterogeneous distinct host types. Each Server has set of VMs which is determined by the Servers' available capacity and VMs resource demand.

We have built the performance Degradation Factor (DF) matrix (**Table 2**), which tracks the performance degradation of a VM with application app1 and another VM with application app2, both applications running concurrently vs. VM with say application app1 allowed to run in standalone mode (no other VMs have apps which would interfere with app1's performance or content with app1's resource needs).

Each (row, column) cell value for e.g., a (1,1) represents the performance degradation factor. We follow the approach given by (Govindan *et al*., 2011) to calculate Effective Degradation Factor (EDF) for the server. We consider per request execution time T_{AVG} as the performance metric. Point to note here is depending on the application workload logic, degradation factor value for e.g., VM1→VM2 (isolated run of vm1 is compared with a concurrent run of vm1 alongside vm2) could be different than VM2→ VM1 (isolated run of vm2 is compared with a concurrent run of vm2 alongside vm1).

In our study, we have considered workloads (apps) which primarily exhibit of the following resource usage (cpu or storage or memory or network). A composite workload characteristic with different resources being used in phases is also a possibility. Workload phase dynamism needs bit more detailed effort. We would consider this as part of future work:

WEE^k_i Weighted Energy Efficiency for S^k_i with interference performance degradation factor accounted for Equation (9):

$$= \frac{EE^k_i}{EDF^{i,j,k}} \quad (9)$$

where $k \in \{1 .. H\}$; $j \in \{1 .. F_k\}$; F_k is the maximum frequency for the host type k; $i \in \{1 .. H_k\}$;

SEE^k_i Best energy efficient server is the server with

$$= \text{Max}\left(WEE^k_i\right) \quad (10)$$

where $k \in \{1 .. H\}$; $i \in \{1 .. H_k\}$; H_k is the number of servers in host type k.

2.6. System Architecture

Our proposed architecture's objective is control request response times processed in virtual machines while minimizing energy consumption of the server cluster. **Figure 1**, shows our system architecture which comprises of the following components:

- Energy Efficiency and Interference aware Server Sequencer
- DPM Controller
- DVFS Controller
- Arbitrator
- Monitor

2.6.1. Energy Efficiency and Interference Aware Server Sequencer

This module specifically identifies the host type configuration that has the best (optimal) EE^k_i ratio. We use Equation (2) to arrive at the server's EE^k_i ratio. Higher the value in of a server's SEE^k_i (or WEE^k_i computed using Equation (9 and 10) indicates that a application request when processed in this server can ultimately reduce energy consumption of the virtualized server cluster or data center and also can remediate performance issues due to interference or shared resource contentions.

2.6.2. DPM Controller

This module focusses on managing the servers' power state transitions and answers the following questions: When should the server be switched ON from OFF state (waken up)? We follow a request batching approach [virtual batching] to answer the first question. The system waits for batching timeout then wakes up the CPU to process requests. CPU could be in either OFF before transitioned to IDLE state. This transition from low power state to high power state (OFF IDLE, has a time expend value $T^{(i,k,j)}_{OFF\_IDLE}$ as captured in **Table 3**. Batching timeout is determined periodically by an adhoc controller to drive the web server with the longest response time to a set point.

We use power and energy consumption metrics models discussed earlier to compute power and energy consumption.

Table 2. Performance interference degradation factor Matrix

		VM2			
	Workload contention scenario	Cpu	Storage	Memory	Network
VM1	Cpu	a(1,1)	a(1,2)	a(1,3)	a(1,4)
	Storage	a(2,1)	a(2,2)	a(2,3)	a(2,4)
	Memory	a(3,1)	a(3,2)	a(3,3)	a(3,4)
	Network	a(4,1)	a(4,2)	a(4,3)	a(4,4)

Table 3. Physical server sample state transition times (in seconds)

Host type k state transition time	(1)	(2)	(3)	(4)	(5)	(6)	(7)
$T^{(i,k,j)}_{OFF\_IDLE}$	50	45	50	45	45	60	75
$T^{(i,k,j)}_{IDLE\_OFF}$	5	5	5	4	4	5	5

(1) HpProLiantMl110G3PentiumD930, (2) HpProLiantMl110G4Xeon3040, (3) HpProLiantMl110G5Xeon3075, (4) IbmX3250XeonX3470, (5) IbmX3250XeonX3480, (6) IbmX3550XeonX5670, (7) IbmX3550XeonX5675

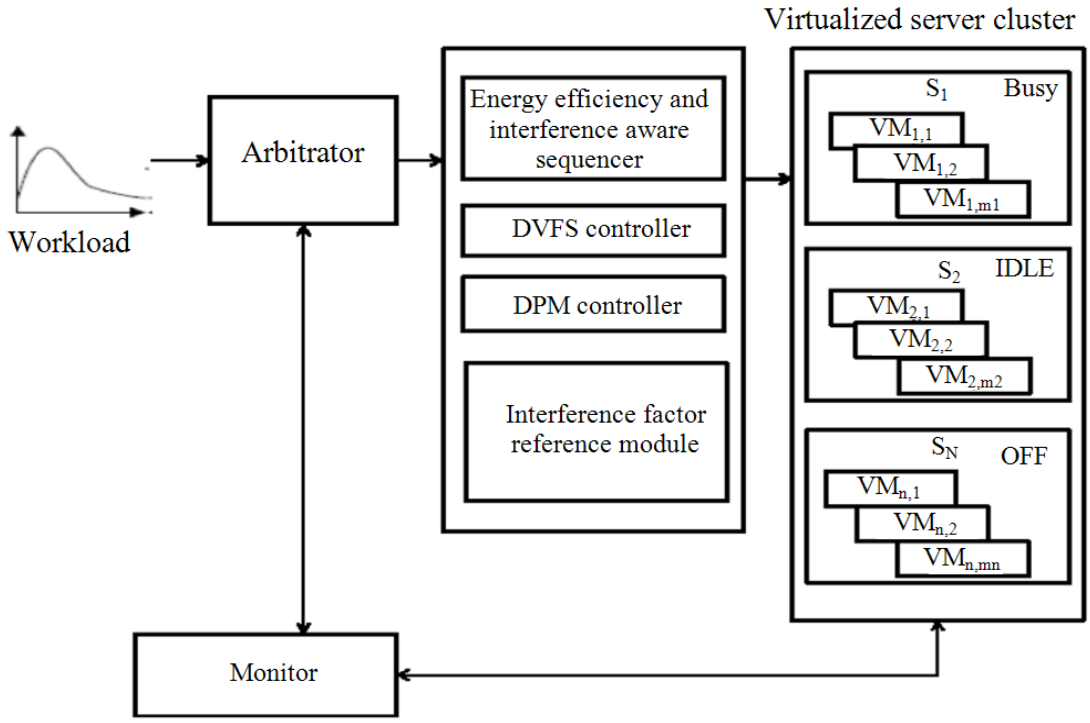

Fig. 1. System architecture

2.6.3. DVFS Controller

Focus of this module is to improve the response times of requests in the system dynamically by manipulating the cpu frequency. This controller is used when the processor is in active or operational state either processing a request or waiting for a request to start processing. In our work, we start (awaken) a server at the lowest possible frequency for the host type.

We use SPEC results to formalize our power modeling exercise at different server cpu utilization% values. Server CPU utilization is computed using Equation (1). Server CPU utilization has a linear relationship with number of VMs/requests processed by the server at that point in time (also known as concurrency level of the server). We use the Equation (6) to compute server power consumption at different cpu utilization% values.

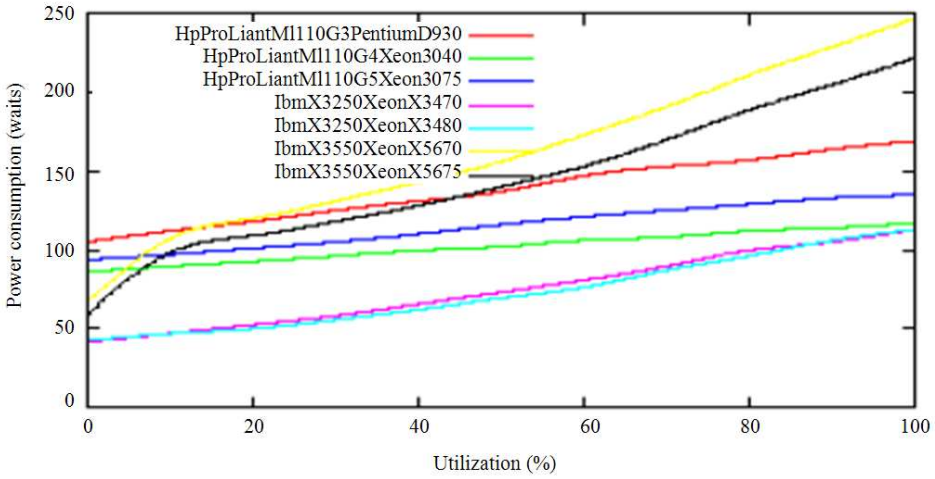

Fig. 2. Power Consumption to CPU Utilization at highest frequency $f^{(k)}_{Fk}$ (SPEC results)

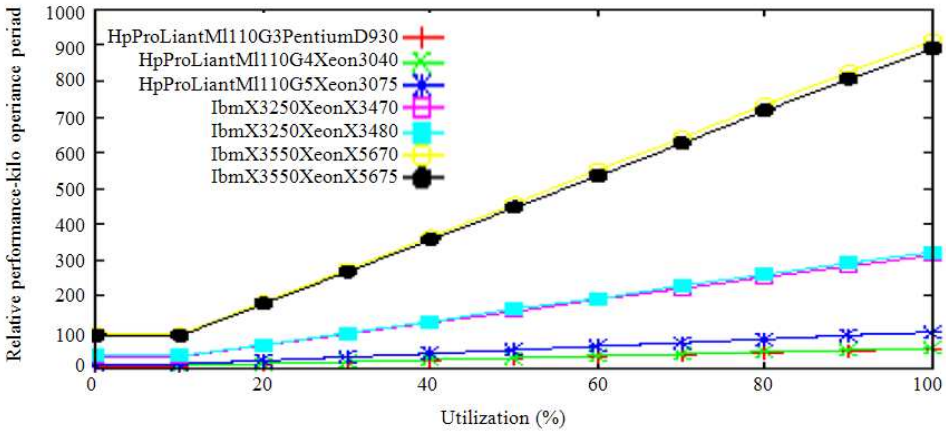

Fig. 3. OPS to Utilization at highest frequency $f^{(k)}_{Fk}$ $OPS^{k,j}$ (util% (SPEC results)

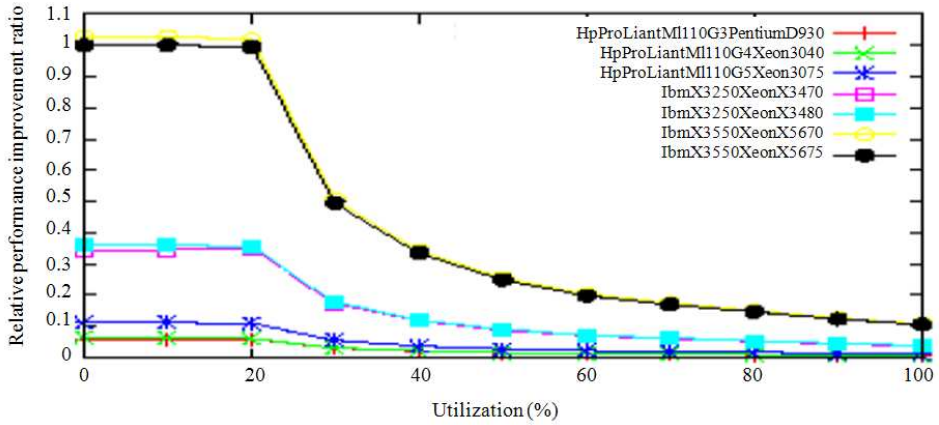

Fig. 4. Response time performance improvement factor $(RelOPS^{(i, k, j)}_{(util\%)})$

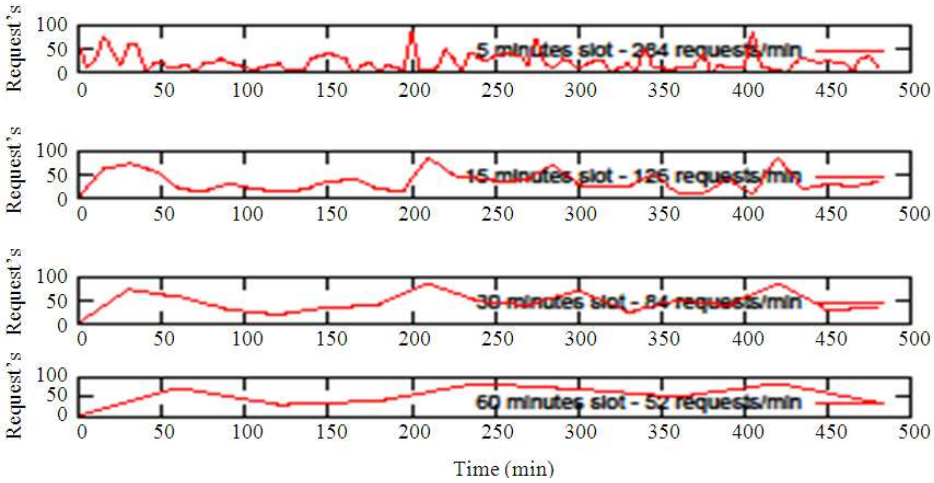

Fig. 5. Timeframe governed Work load -8 hour trace (captured at 5, 15, 30, 60 min time frames) (Qian and Medhi, 2011)

Figure 2 shows the power consumption of different host types at different cpu utilization%. Also, using SPEC results, we propose a model to compute request response time improvement factor ($RelOPS^{(i,\ k,\ j)}_{(util\%)}$) due to concurrency level (different cpu utilization%). **Figure 3** depicts the relative OPS vs. cpu utilization% exhibited by our scoped set of servers of different host types. It clearly shows that each host type's OPS characteristics are unique. The model to arrive at relative response time improvement factor is based on Equation (8). This factor determines the relative delay imposed by the server of a particular host type to request response times when operating at a particular utilization% and particular frequency. **Figure 4** shows the response-time improvement ratio at different cpu utilization% between servers of different host types.

2.6.4. Arbitrator

Arbitrator is the crucial intelligent module which orchestrates and controls functions of other modules in the architecture. Web requests or workload from the clients are redirected, based on load balancing scheme. Monitor module collects metrics like application request response-times, power consumption (Equation 6), energy consumption (Equation 5), from the applications and servers in our virtualized server cluster environment. Arbitrator module, using the collected metrics at periodic control time intervals and immediately after completion of each request, identifies the best energy efficient server (Equation 8) that could mitigate interference performance degradation. Any new work load request, is pushed to this server (VM in this server).

2.6.5. Monitor

This module specifically probes the server cluster for parameters of interest like server cpu utilization%, server power states, server cpu processor frequency levels.

2.7. Simulation Setup

To achieve an efficient simulation that addresses various use case scenarios (in our case these solution schemes are discussed in the results section), the choice of a robust simulator is essential. We have used java based cloud simulator CloudSim (version 3) (Calheiros *et al*., 2011) by enhancing and modifying components as required for our work. In CloudSim, a cloudlet represents a task that is submitted to a datacenter virtual machine. We treat requests as cloudlets. We have assumed the cloudlet job size in our simulation to be of constant value. Service rate of these cloudlets depend primarily on the server host type, DPM operation modes and DVFS frequency enabled on the physical servers. A cloud datacenter is a physical set of machines connected by a network available to receive the virtual machines and workload requests (cloudlets) accordingly.

In our simulation setup, we have considered a cloud setup with non-federated datacenter scenario. We have considered 250 physical hosts (PM) of heterogeneous configuration, selected from amongst 7 host types in a round-robin distribution. Characteristics of each of the host types on power, performance, state details are captured as in **Table 1-3 and Fig. 2-4**. All physical servers are initially in OFF state. We have used 500 VM's from amongst 4 VM types. Each of these VM types has different MIPS requirements from 500 to 1000 MIPS.

Also, to study time-slot specific controls, we have used a synthetic workload as in **Fig. 5**, which captures the demand profiled every 5 min in a 480 min (8 hours) period based on CPU demand trace. We consider 5 different time slot sizes of demand capture: 5, 15, 30 and 60 min. Based on the 5 min demand profile, the demand for larger time slot granularity is taken to be the maximum over all 5-min demands in that time slot.

$D(t, t+\Delta) = \max\{ D(t),\ldots,D(t+\Delta) \}$; where $D(t, t+\Delta)$ denotes the maximum demand of Δ, for e.g., 5-min slot (from t to t+5) (Qian and Medhi, 2011). We have 2092, 1134, 764, 468 total requests tracked against 5 min, 15 min, 30 min and 60 min in 8 hours demand distribution trace. Arrival rate for 15 min workload run is done with 284 requests per time slot; 30 min workload run is with 126 requests per time slot; 45 min workload run is with 84 requests per slot; and 60 min workload run is with 52 requests per slot.

We have considered 250 physical servers and 500 virtual machines. Configuration settings of physical servers and virtual machines follow the same logic as captured for the grid workload.

3. RESULTS

We have used the simulation test bed described previously.. We show the effectiveness of our solution by contrasting:

- Our "energy-efficiency and interference aware" approach ("EE and interference aware" approach); with
- "Default energy-efficiency and interference oblivious" approach ("default EE and interference oblivious" approach)

Default approach does not have interference degradation fitment logic and does not have the best case energy efficiency sequencer logic.

We consider the following scenario extensions as follows:

- With datacenter servers in the cluster being in switch OFF state when we start our workload processing routine-ColdStart
- The server stays in ON power state thereafter
- With datacenter servers in the cluster already in switch ON power state-HotStart
- The server stays in ON power state thereafter
- With datacenter servers in the cluster being in switch OFF state when we start our workload processing routine; Adopt DPM and DVFS performance and power consumption improvement levers-ColdStart; with DPM and DVFS
- Server frequency, server power states are transitioned to low power mode according to respective DVFS and DPM rules

With ColdStart scenario, we start our simulation, with all servers in switched-OFF state. With the arrival of workload requests, select server (as per Arbitrator logic) is switched ON. There is a setup time expend (**Table 2**) and non-zero power consumption expend (**Table 1**) to transition a server from OFF state to ON state. We have accounted for this transition time and power expends into our simulation logic.

Summarized results from our experiments with our energy efficiency and interference aware approach is listed as follows:

- On virtualized server cluster total energy consumption
  - With our "EE and interference aware" approach, we achieve a reduction of up to 8% with coldStart system and 58% with hotStart system compared to similar coldStart and hotStart scenario runs using the "default energy efficiency and interference oblivious" scheme
- On per request response times (T_{AVG})
  - With our "EE and interference aware" approach, we achieve an improvement of at least 10 times when compared with the default energy efficiency and interference oblivious scheme run for both coldStart and hotStart scenarios

Also, we have integrated DPM and DVFS control levers into our energy efficiency and interference aware approach for ColdStart scenario:

- On virtualized server cluster total energy consumption
  - With our "EE, DPM and DVFS with interference aware" approach, we achieve a reduction of up to 40% on select workload request arrival rate scenarios when compared with the default energy efficiency and interference oblivious scheme
  - On certain workload arrival rate scenarios, we see that default energy efficiency and interference oblivious scheme outperforms our approach

- On per request response times (T_{AVG})
  - With our "EE, DPM and DVFS with interference aware" approach, we achieve an improvement of at least 5 times better performance than with energy efficiency interference with DPM and DVFS interference oblivious scheme.

4. DISCUSSION

Initial results (**Fig. 6**) ColdStart scenario shows a significant improvement in per request response times T_{AVG}. Lesser the response time value is termed as improvement. Improvement in per request response times is at-least 10 times (make-span or response times T_{AVG} improvement) with our EE and interference aware approach in comparison to the default EE with interference oblivious approach. This clearly highlights the need to select the right performance centric server with less interference. In our setup, we find that energy efficient servers are also better in terms of performance aspect as-well. Also, we see a maximum of 8% (minimum 5%) reduction in server clusters' energy consumption with our approach in comparison with interference oblivious approach **Fig. 7**.

Another important aspect that shows up in our results is the relationship between request arrival rate and server cluster energy consumption and response-times. As the arrival rate increases, arbitrator module activates more servers to handle the load. Hence, we see a slight increase in energy consumption and power consumption with increase in request arrival rate. On per request response Time (T_{AVG}), we see a slight drop in value with increase in request arrival rate. The reason is primarily due to the fact that with more number of activated servers, there is a higher probability of request getting serviced immediately than waiting for server to become available. This characteristics is common with both scenarios (a and b).

With the case of HotStart scenarios, we see a similar improvement in both server cluster energy consumption and per request response times T_{AVG}. With respect to, per request response-times (T_{AVG}), (**Fig. 8**), an improvement of atleast 10 times is possible with our approach. Also, there is a further reduction in energy consumption (**Fig. 9**) due to the fact that energy expend due to server startup is avoided in this scenario. On a whole, we achieve a minimum of 58% (maximum of 72%) improvement in energy consumption. The reasons for variations in energy consumption for different workload arrival rates is same as that with the ColdStart scenario runs.

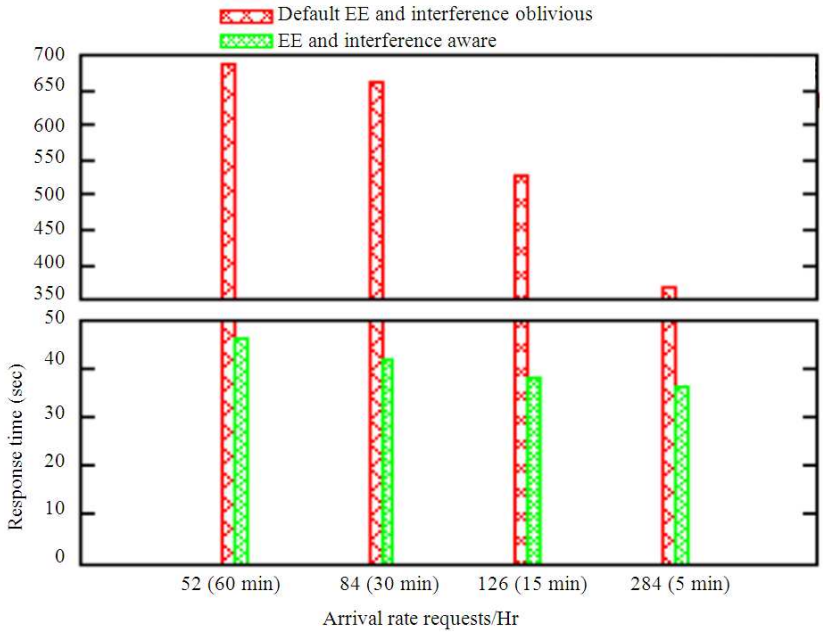

Fig. 6. ColdStart: Average request response times

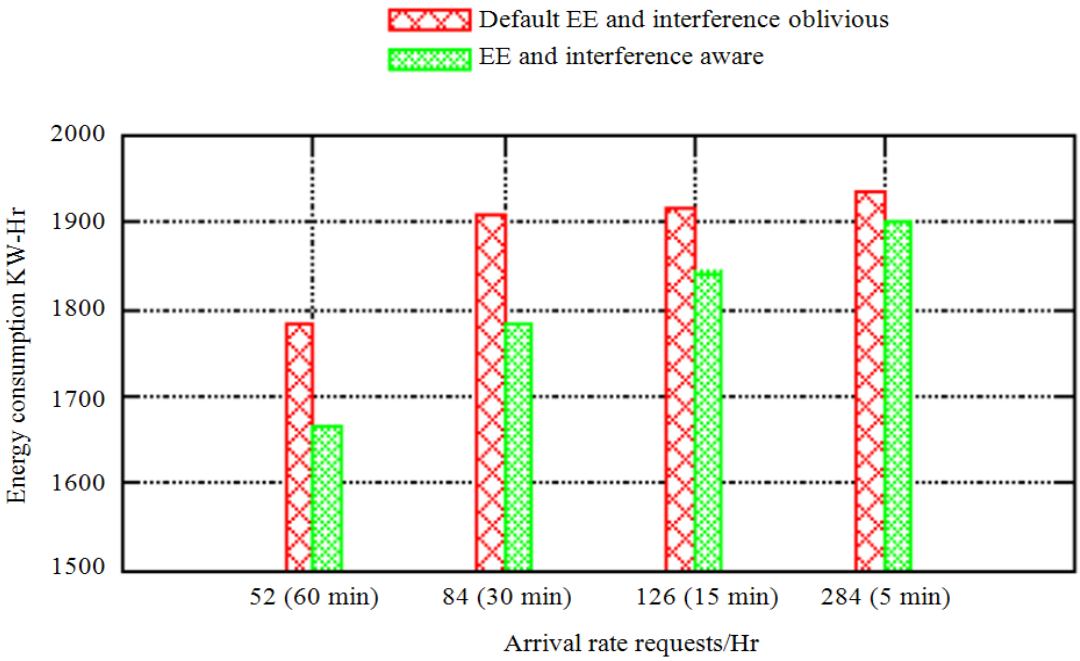

Fig. 7. ColdStart: Energy consumption

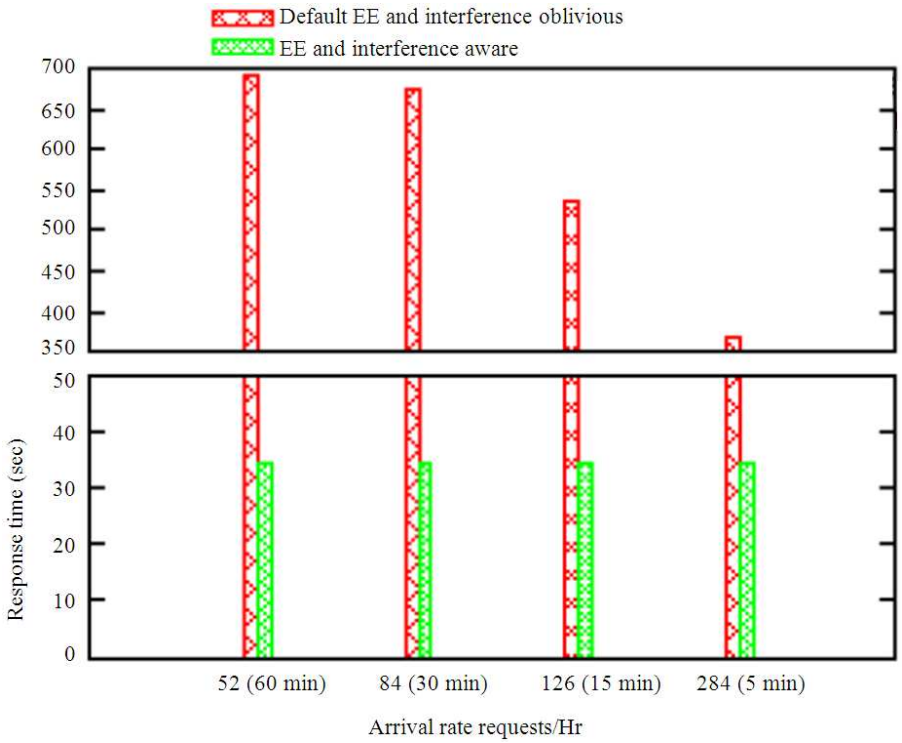

Fig. 8. HotStart: Average request response times

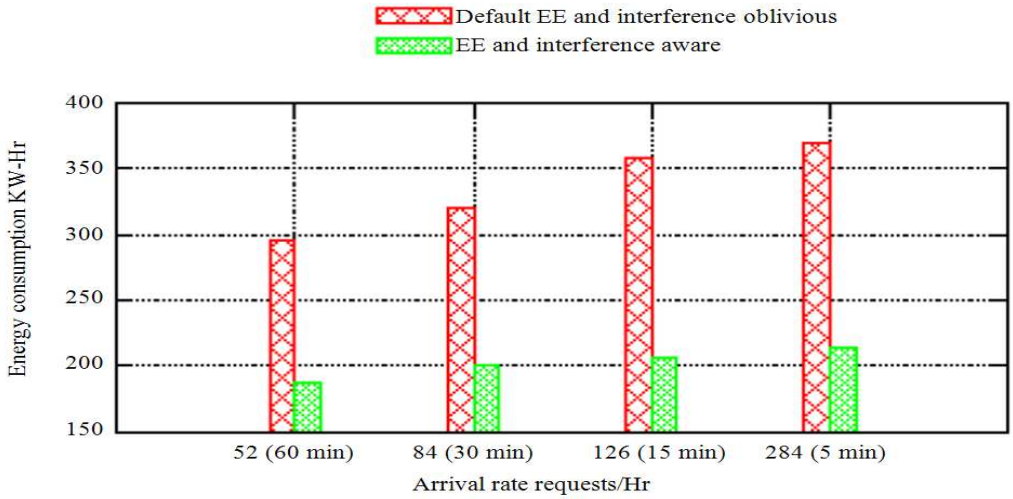

Fig. 9. HotStart: Energy Consumption

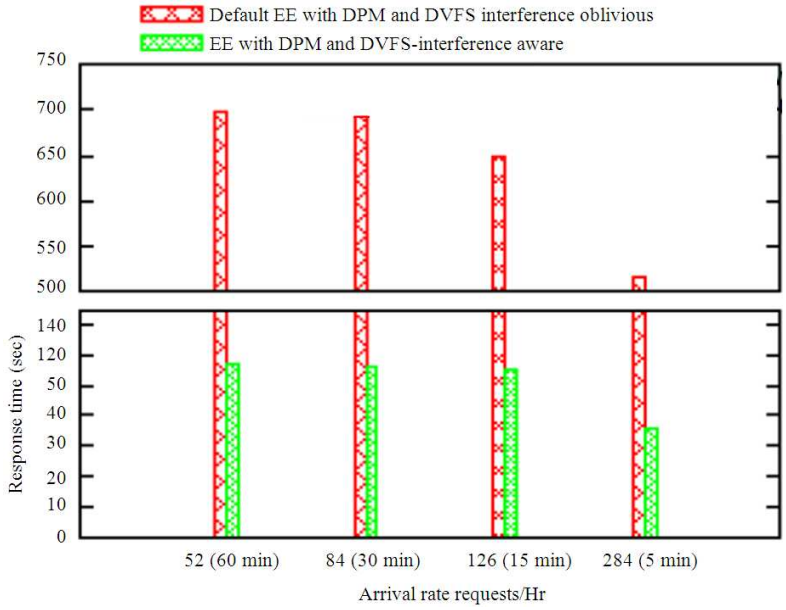

Fig. 10. ColdStart with DPM and DVFS: Average request response times

We have allowed the server once activated to an active ON state, to remain in active ON state all through the simulation duration in scenarios (a and b). We are aware of the fact that server clusters' power consumption and possibly energy consumption savings could further be achieved by switching OFF unused servers. A quick look at scenario approach (c) ColdStart with DPM and DVFS control levers shows that the average request response time (T_{AVG}) (**Fig. 10**), our EE interference with DPM and DVFS aware scheme gives atleast 5 times better performance than with EE interference with DPM and DVFS oblivious scheme. Also, energy consumption (**Fig. 11**) is reduced when compared to a pure ColdStart scenario (**Fig. 7**) on both default EE interference oblivious and EE interference aware approaches. Which is as expected with DPM and DVFS power control levers switching unused servers to low-power state and managing optimal frequency on servers in ON state.

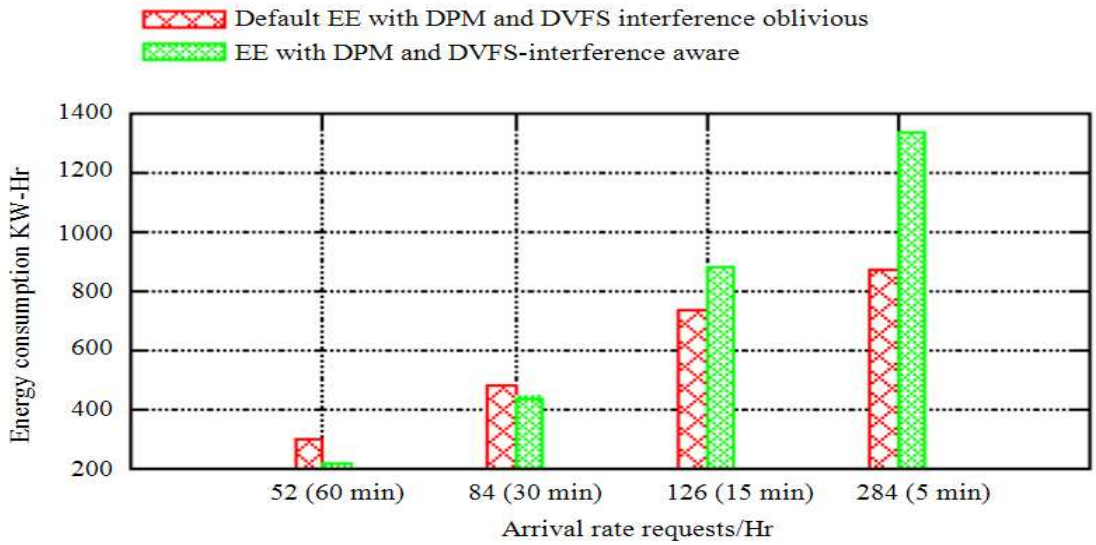

Fig. 11. ColdStart with DPM and DVFS: Energy consumption

We find an interesting aspect that, for workloads with certain arrival rates (here 126 and 284 requests/hr cases), energy consumption of our ColdStart EE and interference aware with DPM DVFS scheme consumes more energy than ColdStart EE and interference with DPM and DVFS oblivious scheme. One reason for this is, the setup or transition times of our best energy efficient and interference aware servers have higher setup time and/or DVFS higher frequency power contribution on such servers are higher. This specific result does highlight the fact that workload type and workload request arrival rate has a very good binding on why one should consider classical default EE interference with DPM and DVFS oblivious scheme vs. the EE interference with DPM and DVFS aware scheme to minimizing virtualized server clusters' or datacenters' total energy consumption.

5. CONCLUSION

In this study, we have considered heterogeneity with respect to a virtualized server cluster comprising of heterogeneous servers, with different set of workload scenarios. We have presented an energy efficiency and interference aware approach to reduce energy consumption in a virtualized server cluster or datacenter environment. We show that energy efficient interference aware mechanism reduces energy consumption by up to 8% in case of cold-start system and 58% with hot-start system compared to the default energy efficiency and interference oblivious system. With respect to per request response times (T_{AVG}), we achieve an improvement of at least 10 times when compared with the default interference oblivious approach run. Also, we integrated our approach with DPM and DVFS control levers. Learning from this exercise is that, on certain runs with specific workload arrival rates, our EE interference with DPM and DVFS oblivious scheme performs better than EE interference with DPM and DVFS aware scheme on energy consumption aspect. As part of future work, we plan to consider workloads with composite resource needs in phases and work towards a generic online approach to predict and quantify energy expend due to interference.

6. ACKNOWLEDGEMENT

We would like to thank Dinesh Mavaluru for his valuable help.

7. REFERENCES

Blagodurov, S., D. Gmach, F.M. Arlitt, Y. Chen and C. Hyser, 2013. Maximizing server utilization while meeting critical SLAs via weight-based collocation management. Proceedings of the IFIP/IEEE Symposium on Integrated Network and Service Management (INSM' 13), Canada, pp: 277-285.

Calheiros, R.N., R. Ranjan, A. Beloglazov, C.A.F. De Rose and R. Buyya, 2011. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice Exp., 41: 23-50. DOI: 10.1002/spe.995

Chiang, R.C. and H.H. Huang, 2011. TRACON: Interference-aware scheduling for data-intensive applications in virtualized environments. Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, (Sc' 11), ACM Press, New York, USA. DOI: 10.1145/2063384.2063447

Delimitrou, C. and C. Kozyrakis, 2013. Paragon: QoS-Aware scheduling for heterogeneous datacenters. Proceedings of the 8th International Conference on Architectural Support for Programming Languages and Operating Systems, Mar. 16-20, ACM Press, Houston, TX, USA., pp: 77-88. DOI: 10.1145/2451116.2451125

Goiri, I., W. Katsak, K. Ley, T.D. Nguyen and R. Bianchini, 2013. Parasol and GreenSwitch: Managing datacenters powered by renewable energy. Proceedings of the 8th International Conference on Architectural Support for Programming Languages and Operating Systems, Mar. 16-20, ACM Press, Houston, TX, USA., pp: 51-64. DOI: 10.1145/2451116.2451123

Govindan, S., J. Liu, A. Kansal and A. Sivasubramanian, 2011. Cuanta: Quantifying effects of shared on-chip resource interference for consolidated virtual machines. Proceedings of the 2nd ACM Symposium on Cloud Computing, Oct. 26-28, ACM Press, Cascais, Portugal. DOI: 10.1145/2038916.2038938

Moreno, I.S., R. Yang, J. Xu and T. Wo, 2013. Improved energy-efficiency in cloud datacenters with interference-aware virtual machine placement. Proceedings of the IEEE 11th International Symposium on Autonomous Decentralized Systems, Mar. 6-8, IEEE Xplore Press, Mexico City, Mexico, pp: 1-8. DOI: 10.1109/ISADS.2013.6513411

Mukherjee, J., D. Krishnamurthy, J. Rolia and C. Hyser, 2013. Resource contention detection and management for consolidated workloads. Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management, May 27-31, IEEE Xplore Press, Ghent, pp: 294-302.

Novakovic, D., N. Vasic, S. Novakovic, D. Kostic and R. Bianchini, 2013. DeepDive: Transparently identifying and managing performance interference in virtualized environments. Proceedings of the USENIX Annual Technical Conference, Jun. 26-28, San Jose, CA.

Pu, X., L. Liu, Y. Mei, S. Sivathanu and Y. Koh, 2010. Understanding performance interference of I/O workload in virtualized cloud environments. Proceedings of the IEEE 3rd International Conference on Cloud Computing, Jul. 5-10, IEEE Xplore Press, Miami, FL, pp: 51-58. DOI: 10.1109/CLOUD.2010.65

Qian, H. and D. Medhi, 2011. Server operational cost optimization for cloud computing service providers over a time horizon. Proceedings of the 11th USENIX Conference on Hot Topics in Management of Internet, Cloud and Enterprise Networks and Services, (Hot-ICE' 11), ACM Press, CA, USA., pp: 4-4.

SPEC Benchmarks, 2013. Standard Performance Evaluation Corporation.