Extensions to PerfCenter for Power Management Support

Rakesh Kumar Mallick
SR No: 4710-410-091-06888
Advisor: Dr. Varsha Apte
Dept of CSA, IISc
Report of ME Dissertation (2010-2011)

Abstract

PerfCenter is a tool to measure performance of a distributed application on a distributed system or on a data-center. It takes various hardware and software details over distributed system as input and provides various performance measures such as average response-time, throughput etc. of various events as output. As a result of technological advancements, various devices of a host machines like CPU, hard disk etc. can be operated at different voltages. This results in multiple device processing speed. PerfCenter considers each device having only one processing speed, hence changes to PerfCenter to make it support powermanaged devices with multiple processing speed is required.

Modeling of powermanaged devices with various powermanaged attributes in PerfCenter is being done and necessary changes to PerfCenter are made in order to support it. Effect of powermanaged device on average device speed, energy-consumption per request, perfromance measures like response time and throughput etc. are being observed.

1 Introduction & Motivation

Over some last decades, growth of networked and distributed applications like Web applications increased rapidly. A distributed system can have a large number of heterogeneous host machines with different number and capacity of devices. A number of software servers can be deployed on these host machines. Having large number of host machines and software servers of different types and capacities makes distributed systems good for one task and bad for another. Hence estimating performance of an application on a distributed system becomes important and it helps us in making appropriate deployment and configuration related decision. PerfCenter helps us in doing the same. PerfCenter is a tool for measuring performance of a distributed application on a distributed system. It takes various hardware and software servers deployment details over distributed system and provides various performance measures such as average response-time, throughput etc. of various events. PerfCenter provides a most natural way to submit input specification to it's users.

A software server with customers can also be viewed as a queuing system with software server as a service entity and customers as requests in a queue. All hardware resources with requests are also viewed as queuing systems. Using queuing theory, one can make a theoretical model for estimating performance of these applications on data-center. Same is achievable by using discrete time simulation where we schedule next event every time when an event is occurred and does not bother about theoretical model. PerfCenter has analytical model as well as a discrete time simulation model to estimates performance of an application on distributed environment.

PerfCenter assumes a single processing speed for each devices during execution of a tasks, but today's devices like CPU, hard disk etc. are capable of switching between different processing speeds during execution of a task. At lower workload on device, it can be switched to lower processing speed, which results in energy savings[1][2], and switches to higher processing speeds as workload on it increases. Since operating at lower power results in power savings, we call this property of device as powermanaged. Powermanaged property of a device is mostly helpful for battery operated systems like laptops.

Introduction of powermanaged feature for all devices in PerfCenter is desirable to make it work for devices with multiple speed. Currently PerfCenter considers devices of host machines to be operated at single processing speed which results in a single average service time for any specific task.

The rest of the paper is as follows: the problem statement is in section 2, followed by background work in section 3. Section 4 introduces powerman-

aged devices and describes it's properties and different powermanaged governors. Section 5 is modeling of powermanaged devices for PerfCenter which tells major changes in input specification and all formulation. Section 7 captures experiments and results followed by conclusion and references.

2 Problem Definition

Modeling of powermanaged devices for PerfCenter, updating it's input specification to support new features and to add, modify modules to it to make it support powermanaged devices and to observe it's effect on average device processing speed, energy-consumption per request and on performance measures like responce time, throughput etc.

3 Background work

Discrete event simulation is learnt in order to fulfill the requisite. Some basic simulation exercises were performed. Knowledge of customer behavior model is required in order to understand the analytical model of distributed systems. [3][4] was read to fulfill this. A model of CPU frequency scaling enabled m/m/1 queue was developed.

"PerfCenter: A Performance Modeling Tool for Application Hosting Centers"[5] describes the PerfCenter specification, and its performance analysis utilities in detail, and illustrate its use in the deployment and configuration of a Webmail application. [6] is a work done on modeling of CPU frequency scaling on PerfCenter. The following are some of the features that are already available in PerfCenter and can be set through input specification. The building and design of these features make it powerful yet easy to be used tool:

- Modeling of distributed application on a distributed system.
- Easy way of allowing distributed application and system specification.
- Setting default properties of host machines and device of same type.
- Setting host machine and device properties individually.
- Setting LAN attributes and deploying hosts on it.
- Allowing synchronous and asynchronous calls between different requests.
- Provides output in terms of waiting time, service time, response time, throughput, average queue length, and doing device bottleneck analysis.
- Modifying device/host properties, undeployment and deployment of software servers on hosts, host on LAN etc and doing bottleneck analysis of this configuration with previous one.

Such properties of PerfCenter makes it a friendly tool to users and distributed system architects.

4 Power Managed Devices:

All device that can be operated at multiple processing speed[1] [2] are operated at different voltages. In some work, the relationship between operating voltage and power is taken as cubic[12]. Some work analysed relationship between energy savings and energy-consumption per transaction and device utilization[7].

The powermanagement policies and it's effect on performance is as follows:

4.1 Power Management Governors and policies:

Design and architecture of a device enables it to be operated at multiple speeds, but how to use them depends on the Operating System. An OS can support a number of policies which can use different device-speeds at different occasions. These policies are called governors. For example, Linux supports ondemand, conservative, powersaver, performance and userspace governors for CPU [10][11]. There are some commercial powermanaged governors available also.

Device probe interval is the time interval in which a device utilization is being computed and given to governor as input. Various popular device powermanagement governors are described below. These are being implemented in PerfCenter.

1. **Ondemand:** Ondemand is the most popular powermanaged governor. If utilization of a device in last probe interval is found to be less than a certain threshold, called down_threshold, than governor switches device speed to it's minimum stable

[1] device processing speed is sometimes referred as processing speed here

[2] Stable operating speeds that can be achieved by this device. For CPU this is clock-ticks/sec or operating frequency. For hard disk this is RPM.

speed[3]. If device utilization in last probe interval is greater a certain threshold, called up_threshold, governor switches the device speed to it's maximum stable device speed, otherwise no change in device speed. It is the most sensitive governor which switches speeds between maximum and minimum speeds.

2. **Conservative:** Conservative is very much similar to ondemand governor except it uses all available stable device speeds. If utilization of last probe interval is less than the down_threshold, governor switches device speed to next available lower stable speed. If this utilization is found to be greater than the up_threshold then governor switches to next available higher stable speed.
3. **Powersaver:** Powersaver governor works only at the minimum available stable device speed. It does not switches speed. It is like device has only one operating speed(the minimum one) and is not powermanaged.
4. **Performance:** Performance governor works at the maximum available stable device speed. It is like device has only one operating speed(the maximum one) and is not powermanaged.
5. **Userspace:** User can select any one operating speed from all available stable device speeds for it's device. Device speed does not change over time.

Table 1 - Comparison between different Governors

Governors	Operating speeds	Switches Speed
Conservative	All available speeds	Yes
Ondemand	2 [highest, lowest]	Yes
Powersaver	1 [lowest]	No
Performance	1 [highest]	No
Userspace	All available speeds	No

4.2 Effect on performance:

In order to conserve energy, devices are operated at lower speed levels, which results in increased service time and response time.

[3]Switching device speed does not take place instantly. During switching speeds we achieve intermediate device speeds also but these are not considered as stable speed. A device speed at which a device can be operated for long enough time is only considered as stable device speed

As some service level agreement(SLA) constraints are associated with distributed applications, keeping response time within SLA constraints and reducing device speed results in good energy savings. Furthermore, one would be interested in finding best combination of response-time and energy-consumption per transaction. Throughput of the system remains unchanged as service-time of the system is still greater than inter-arrival time of requests.

Power consumption equations for CPU can be given by[8][9]:

$$P_{total} = P_{static} + P_{dynamic} \quad (1)$$

$$P_{dynamic} = ACv^2f \quad (2)$$

where
P_{total} is the total power consumption by CPU.
P_{static} is the static power consumption by CPU.
$P_{dynamic}$ is the dynamic power consumption by CPU.
A is the percentage of active gates;
C is the total capacitance load;
v is the supply voltage;
f is the processor frequency.

5 Modeling Power Managed Devices:

There is always a transition delay for switching processing speeds for all devices. We can't expect instantaneous transitions for hard disk from 3600 rpm to 7200 rpm. The figure-1 gives details of intermediate transitions for intel's i5 CPU using ondemand governor and operatable at processing speeds: 1.2, 1.33, 1.47, 1.6, 1.73, 1.87, 2.0, 2.13, 2.26, 2.4 (all in GHz.).

In figure-1, 1st row and 1st columns stands for CPU frequencies and the table data represents the no. of transitions it made while switching frequencies. Ondemand powermanagement governor operates device only at two frequencies, the highest and the lowest. But still we can see a number of intermediate transitions, which are small in number so can be ignored.

Assuming that device speed transitions take very small time such that intermediate device speed can be ignored, we develop powermanaged model for devices.

5.1 Changes in PerfCenter Specification Language:

Changes in PerfCenter input language and parser are made to support new features. As powermanagement is a property of a device, only device attributes are getting modified. Devices with same type can have

	From :	To				CPU frequencies						
	:	2400000	2399000	2266000	2133000	1999000	1866000	1733000	1599000	1466000	1333000	1199000
	2400000:	0	501	408	219	282	275	226	242	238	285	10692
	2399000:	110	0	66	145	14	10	8	6	10	10	122
CPU	2266000:	70	0	0	111	106	11	12	8	12	8	136
freq-	2133000:	90	0	0	0	104	105	8	12	9	8	139
uency	1999000:	101	0	0	0	0	92	75	11	13	11	203
	1866000:	91	0	0	0	0	0	90	71	9	9	224
	1733000:	80	0	0	0	0	1	0	119	19	12	189
	1599000:	95	0	0	0	0	0	1	0	144	14	216
	1466000:	128	0	0	0	0	0	0	1	0	70	256
	1333000:	106	0	0	0	0	0	0	0	1	0	321
	1199000:	12496	0	0	0	0	0	0	0	0	1	0

Figure 1: CPU processing speed transitions with ondemand governor

```
//powermanaged device definition block
powermanagement cpu1
speed_levels 1.2 1.6 2.0 2.4 end
probe_interval_min  .001
probe_interval_max  50
probe_interval  .2
governor_up_threshold  50
governor_down_threshold 20
end

powermanagement cpu2
speed_levels 1.2 1.6 2.0 2.4 end
probe_interval_min  .001
probe_interval_max  50
probe_interval  .2
governor_up_threshold  50
governor_down_threshold 20
end


//hosts definition blocks
host machine1[1]
cpu1 count 4
cpu1 buffer 99999
cpu1 schedP fcfs
cpu1 power_managed governor ondemand
//conservative  powersaver  performance
//cpu1 power_managed governor userspace
//setspeed_level_index 1
end

host machine2[1]
cpu2 count 2
cpu2 schedP fcfs
cpu2 buffer 99999
cpu2 power_managed governor ondemand
cpu2 probe_interval  10
cpu2 governor_up_threshold  80
cpu2 governor_down_threshold 20
end
```

Figure 2: Changes in Input Language Specification

same device properties or could be different as per user specifies. Changes to the input language goes like this:

1. Set default powermanagement attributes for all device types.
2. For a device on a host machine, set it's powermanagement attributes. This will overwrite the default values.

The following components to input specification are being added:

- **powermanagement:** It defines the default settings for a device type. This definition is done before defining host definitions.
  `powermanagement <device-name>`
- **speed_levels:** It defines various speeds available for a device of type powermanaged. All service time taken in input file are wrt. the highest speed. This comes under powermanaged section.
  `speed_levels <speed-1> <speed-2> ..... end`
- **probe_interval:** It defines the time interval in which utilization of device instance is measured and given it as input to powermanagement governor. It's default value is 1.
  `probe_interval <interval>`
- **probe_interval_min:** It defines the minimum probe interval allowed to a device.
  `probe_interval_min <interval-1>`
- **probe_interval_max:** It defines the maximum probe interval allowed to a device.
  `probe_interval_max <interval-2>`
- **governor_up_threshold:** It defines the utilization up threshold of the device powermanagement governor. It's default value is 80.
  `governor_up_threshold <up_threshold>`

- **governor_down_threshold:** It defines the utilization down threshold of device powermanagement governor. It's default value is 20.
  `governor_down_threshold <down_threshold>`

- **power_managed:** This comes under device definition under host definition block to set user specified powermanaged attributes of a device.
  `<device-name> powermanaged`

- **governor:** If powermanagement governor is not set during device definition under host definition block, then default governor is set to be ondemand. Otherwise we can set one from these available: performance, ondemand, conservative, powersaver and userspace.
  `<device-name> powermanaged governor <governor-name>`
  `<device-name> powermanaged governor userspace setspeed_level_index <index>`

  Userspace governor sets device speed to a particular speed_level. It requires index of speed_level to be given with it.

5.2 Pseudo-code for event scheduling:

Pseudo-code for event scheduling after considering device probe events is given in algorithm-1.

Service-time of an event can be set using the following equation:

$$servt_{effective} = servt * \frac{S_{max}}{S_{curr}} \quad (3)$$

where
S_{max} : Highest device processing speed with which it can be operated.
S_{curr} : Current device processing speed.
$servt$: Service time wrt. maximum device speed.
$servt_{effective}$: effective service time at current device speed.

For initialization:

$$S_{curr} = \begin{cases} S_{min} & \text{for powersaver, conservative and ondemand} \\ S_{max} & \text{for performance} \\ \text{<user specified>}^4 & \text{for userspace} \end{cases}$$

Service time remaining changes with time and device speed. Current device speed may be changed after every device probe event. Service time remaining

```
1: for all hosts do
2:   for all devices do
3:     if device is powermanaged then
4:       add device-probe event into global event-list
5:     end if
6:   end for
7: end for
8: while simulation is not complete do
9:   if next event is device-probe event then
10:    advance time-stamp
11:    for all device-instances do
12:      get utilization of device-instance in last probe interval
13:      give this utilization as input to powermanagement governor
14:      governor returns new device-instance speed
15:      for all events from device-instance-associated event list do
16:        update service-time remaining of event
17:      end for
18:    end for
19:  end if
20:  if next event is arrival event then
21:    advance time-stamp
22:    schedule it on a device-instance of a device on a host
23:    set it's service time
24:    if device is powermanaged then
25:      add this event into device-instance-associated event list
26:    end if
27:  end if
28:  if next event is device-probe event then
29:    advance time-stamp
30:    remove event from device-instance-associated event list
31:    update device speedup
32:  end if
33: end while
```

Algorithm 1: Pseudo-code for event scheduling

is computed using following equation:

$$SR_{new} = \frac{speed_{old}}{speed_{new}}(SR_{old} - \Delta t),\ for \Delta t < SR_{old} \quad (4)$$

where
SR_{old} : Service time remaining at last time-stamp with device speed $speed_{old}$
SR_{new}: Service time remaining at current time-stamp with device speed $speed_{new}$
Δt: (last time-stamp - current time-stamp)

Effective device speed for a device is computed after a request is completed. It can be computed using following equations:

$$S_e = \frac{WBT_t}{BT_t} \tag{5}$$

where
S_e: effective device speed
BT_t: total busy time of a device
WBT_t: total weighted busy time of a device

$$WBT_t = BT_l * S_{el} + \Delta WBT_{req} \tag{6}$$

$$BT_t = BT_l + \Delta BT_{req} \tag{7}$$

where
BT_l: total busy time of device before this request
S_{el}: effective device speed before this request
ΔWBT_{req}: weighted busy time interval of device for this request
ΔBT_{req}: busy time interval of device for this request

$$\Delta WBT_{req} = \sum_{\Delta t \in T_{req}} \Delta t * S_{\Delta t} \tag{8}$$

where
Δt: busy time interval of device during which device speed is unchanged
$S_{\Delta t}$: device service time for a request at time interval Δt

6 Experiments and Results:

The performance governor works as a device has only one processing speed and is not powermanaged, so we can compare our results from other governors with it.

6.1 Effect of arrival-rate on effective device speed and utilization:

Segment of input specification for this experiment is given below. Service-time for all devices except cpu1 is kept relatively very low. Arrival rate of request to device cpu1 varies from 1 to 38 per time unit. Experiment was performed for open network with 100000 customers in order to have steady state output.

```
powermanagement cpu1
speed_levels 1 1.5 2 2.5 3 end
probe_interval .2
governor_up_threshold 80
governor_down_threshold 20
end

host machine1[1]
cpu1 count 4
cpu1 buffer 99999
cpu1 schedP fcfs
cpu1 power_managed
end

task t1
cpu1 servt 0.1
end

server s1
thread count 8
thread buffer 99999
thread schedP fcfs
task t1
end
```

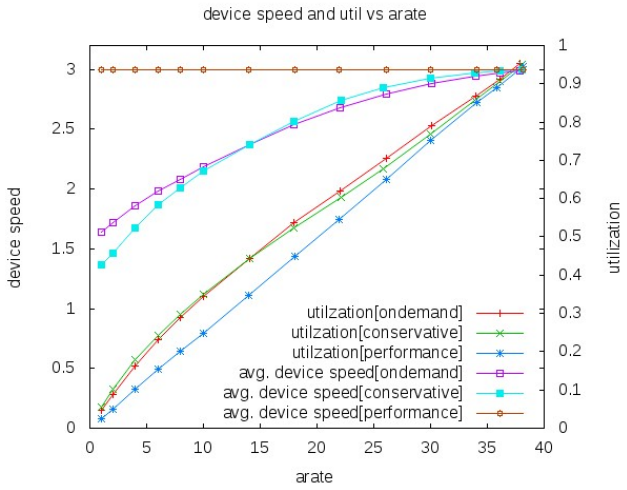

Figure 3: device speed and utilization vs arrival-rate

At lower arrival rate of request, workload on device is lower(figure-3) hence powermanaged governor operates the device at lower device speeds. As a result utilization of device goes up compared to the performance governor.

6.2 Effect of governors on performance and power savings

Input file for this experiment is same as above. Here our interest is on effect of powermanaged device on

average throughput and response time.

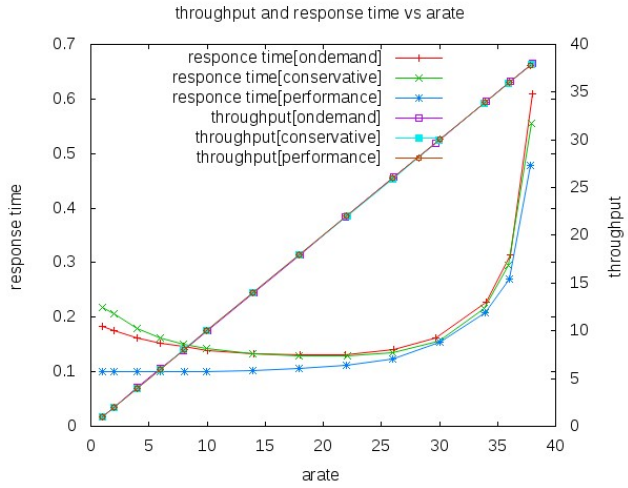

Figure 4: Throughput and avg. response time vs arrival-rate

As workload increases on a device, powermanaged governor decreases it's service time by increasing its processing speed, but performance governor continues to operate device at it's maximum processing speed. Throughput in both the cases are coming to be same(figure-4) as we always get service time of device lower than interarrival time of requests, hence average throughput of devices do not change.

In figure-5 and 6, effect of powermanaged governors on energy consumption per request and energy savings is being observed. For this experiment, we assumed a CPU device with capacitance=1, static power = 2.45 unit, minimum voltage on which device can be operated = 1 unit, and maximum voltage = 2.2 unit. we assumed a linear relationship between device speed and operating voltage. We use equations-1,2 for measuring power, energy and energy consumption per request.

In figure-5, we can see a clear change in average response time of device when powermanaged governor is used. On lower workload, governors try to operate device at lower processing speed. As a result response time of devices increases. Using powermanaged governor on higher device utilization gives worse response time hence is not a good idea to use it at higher device utilizations. Energy consumption per request is lowered by using governors.

In real time systems and some Web applications, where stricter SLA is associated with response time, using powermanaged governors could be a bad idea,

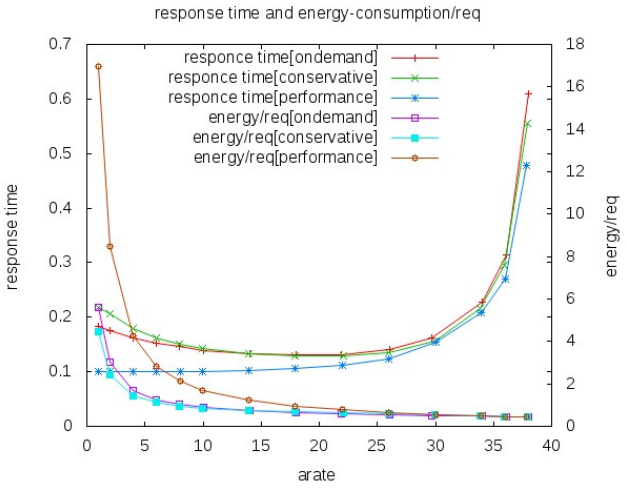

Figure 5: Throughput and avg. response time vs arrival-rate

otherwise better governors have to be used which affect response time the least.

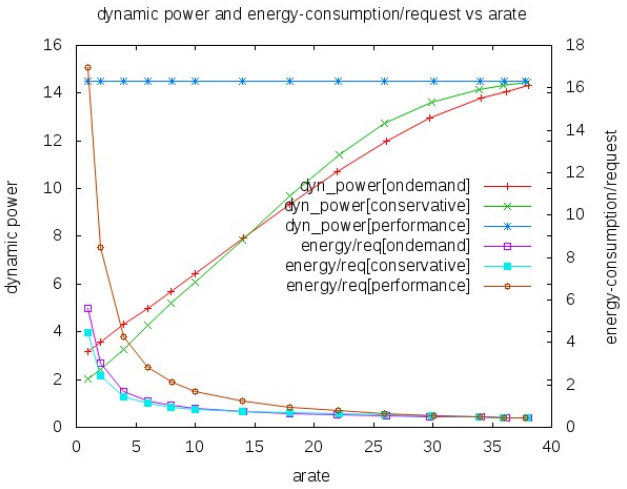

Figure 6: Dynamic power consumption and energy consumption/request vs arrival rate

6.3 Effect of probe interval on response time:

Small intervals make device sensitive towards workload. Governors change device speed quickly at the starting of probe intervals. This takes longer execution time for simulation and may lead to flip-flop of device between different processing speeds also. On the other hand, having large intervals may have a bad

effect on performance when a request require higher processing speed but have to wait for it for longer time. This time can be as long as the device probe interval.

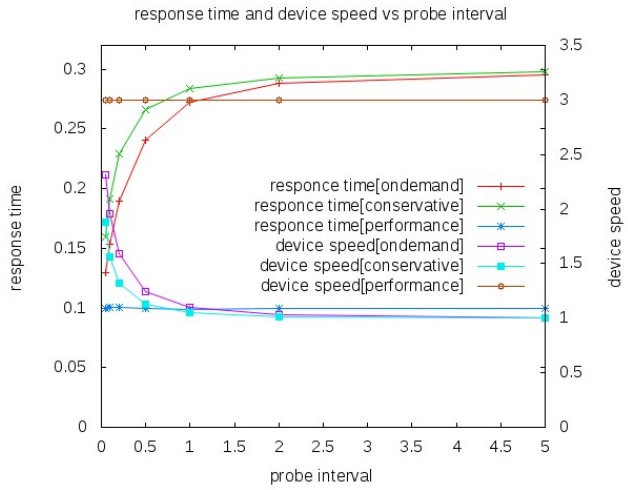

Figure 7: device speed and utilization vs arrival-rate

Figure-7 shows the effect of probe interval on response time. This experiment was performed with up_threshold at 50 and arrival rate of request at 4 requests per time unit.

6.4 Effect of up_threshold and down_threshold on response-time and average device speed:

Having higher up_threshold and lower down_threshold restricts device to be operated only at one processing speed for longer time and vice-versa. up_threshold and down_threshold of a governor determines sensitiveness of a device towards utilization in probe intervals.

For this experiment, we took arrival rate of request 25. We kept down_threshold constant at 20 and changed up_threshold form 50 to 95. As figure-8 shows, on increasing up_threshold, the governor increases device speed at much higher probe interval utilizations, hence as a result device effective speed decreases and response time increases.

We changed down_threshold from 5 to 50 and kept up_threshold at 80. Figure-9 shows the effect of down_threshold on response time and average device speed. As down_threshold increases, governor can lower device speed at high probe interval utilizations also hence average device speed decreased and as a result average response time increases.

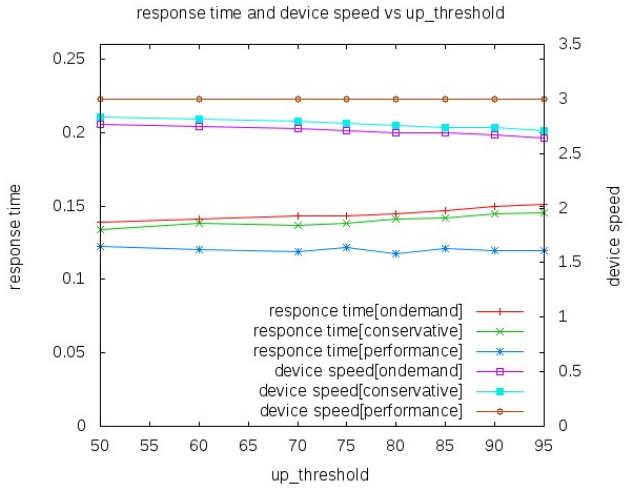

Figure 8: response time and device speed vs up_threshold

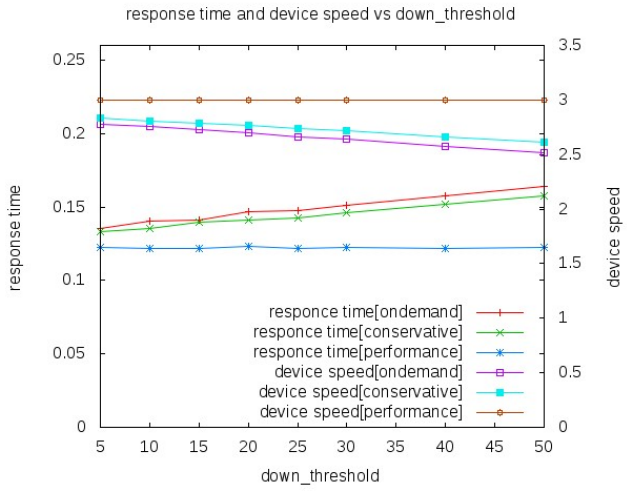

Figure 9: response time and device speed vs down_threshold

7 Conclusions and Future Work

We explained powermanaged devices and it's governing policies called device governors, proposed input specification changes related to powermanaged attributes and modeled powermanaged devices for PerfCenter. Apart from this we used eq-1,2 on CPU device and observed effect of powermanaged governors and it's various attributes on average response time, throughput, effective device speed and energy-consumption per transaction.

We observed that using powermanaged governor at lower workload is good in terms of energy savings but

it increases average response time of the device. If response time of device is within SLA limits for applications associated then the governor is acceptable. At higher device utilization, powermanaged governor only worsen the situation with increased response time. There is nearly no effect of powermanaged governors on throughput. At lower probe intervals, governors becomes sensitive towards workloads and take decision regarding changes in processing speed quickly. up_threshold and down_threshold decides the probe intervals threshold utilizations. These can be used to lower the response time but as a result effective device speed and effective energy consumption per request will be increased and vice-versa.

PerfCenter can be extended to support Virtual Machines. This will enable PerfCenter for making decisions regarding VM configurations, deployment and migrations on a data-center and can tell performance of VMs. Introduction of virtualisation to PerfCenter for above mentioned is required.

References

1. F. Douglis, P. Krishnan, B. Bershad, "Adaptive Disk Spin-down Policies for Mobile Computers", in *second* USENIX *Symposium on Mobile and Location-Independent Computing*, Ann Arbor, MI, pp. 122-137, April 1995.

2. M. Horowitz, T. Indermaur, and R. Gonzalez, Low-power digital design, in Proc. IEEE Symp. Low-Power Electron., 1994, pp. 811.

3. "Scaling for E-Business: Technology, Model, Performance, and Capacity Planning" by Danial A. Menasce and Virgilio A. F. Almeida.

4. "Analytical Model of Web Servers in Distributed Environments" by Paul Reeser and Rema Hariharan, WOSP 2000, Ontario, Canada.

5. "PerfCenter: A Performance Modeling Tool for Application Hosting Centers" by Akila Deshpande, Varsha Apte, Supriya Marathe, WOSP 2008, Princeton, New Jersey, USA.

6. "Validation, Defect Resolution and Feature Enhancements of PerfCenter"(June 2009) by Puram Niranjan Kumar, M.Tech. CSE, IIT Bombay.

7. S. Srikantaiah, A. Kansal, "Energy Aware Consolidation for Cloud Computing"

8. R. Ge, X. Feng, and Cameron, "Performance-constrained distributed dvs scheduling for scientific applications on power-aware clusters", in *Proceedings of the 2005 ACM/IEEE conference on Supercomputing*. IEEE Computer Society Washington, DC, USC, 2005.

9. S. Kaxiras and M. Martonosi. Computer Architecture Techniques for Power-Efficiency. Morgan & Claypool Publishers, 2008.

10. http://www.mjmwired.net/kernel/Documentation/cpu-freq/governors.txt.

11. http://www.pantz.org/software/cpufreq/usingcpufreqonlinux.html.

12. C. Tianzhou, H. Jiangwei, Z Zhenwei, X. Liangxiang, "A practical dynamic frequency scaling scheduling algorithm for general purpose embedded operating system", International Journal of u-and e-Service, Science and Technology,2009.