6. COMBINED DYNAMIC POWER MANAGEMENT AND DYNAMIC VOLTAGE FREQUENCY SCALING CONTROLS

6.1 OUTLINE

Reducing power consumption and maintaining user or application performance criteria is an important goal in virtualized server cluster system design. Achieving this multiple objective requirement in a virtualized environment is a challenge. One of the techniques widely explored in the literature to achieve this goal is the Dynamic Voltage Frequency Scaling (DVFS) approach. However, power consumption reduction due to DVFS is far lesser than what can be achieved with Dynamic Power Management (DPM) control approach to switch off the server when not in use.

DVFS reduces the dynamic power consumption by dynamically adjusting voltage and frequency of a processor. Lower frequencies result in lower power consumption but the task execution time increases which shortens the idle intervals. On the other hand, DPM reduces the static power by switching the processor to a sleep mode with low static power consumption. The limitations of the pure DPM are: a) dynamic power consumption is not considered when making the DPM-related decisions, b) state transitions has additional power and latency penalty, hence it is worthwhile to switch the processor to sleep mode only when the idle interval is longer than a certain threshold time called break-even time.

In this Chapter, a power optimization problem is formulated that meets the defined performance criteria for the workload. Application request response time is considered as the performance metric. The adaptive controller is designed to track application performance in the first step, dynamically control and batch requests and assign the requests to the best server in order to minimize the power consumption as the second step.

The aim is to reduce the power consumption of a virtualized server cluster by making use of an adaptive hybrid approach which identifies the right server with the best performance to power metric (PPM) to process the
workload requests and uses DPM and DVFS controls to reduce static power consumption and meet performance in a virtualized server cluster environment.

6.2 SOLUTION APPROACH

6.2.1 Server performance to power metric model

Server cluster comprises of \( N \) Servers from amongst \( M \) heterogeneous distinct host types or configurations is considered. Servers of a particular host type \( k \) exhibit same performance and power characteristics. Server host types \( H_k \) are such that, no two host type servers has the same characteristics.

Server \( S_i^k \) with the best Performance to Power metric (PPM) ratio in a server cluster is computed as follows:

\[
PPM_i^k = \frac{\text{Cores}_k \times \text{MIPS}_k}{\text{Power}_{i,j}^{k}}
\]  

Server \( S_i^k \) with best PPM metric = MAX \( \{ \text{PPM}_i^k \} \); \( \forall i \in \{1..H_k\} \); \( \forall k \in \{1..M\} \)  

where

\( k \) is the server host type
\( M \) is the total number of host types;
\( H_k \) is the total servers of host type \( k \);
\( F_k \) is the maximum server frequency index for the host type \( k \);
\( H_k \) is the Host type vector represented as a tuple \( \{ \text{Power}_{i,j}^{k}, \text{MIPS}_k, \text{Cores}_k \} \) for each host type \( k \);
\( j \in \{1..F_k\} \); \( \text{util}\% \in \{0..100\} \);
\( S_i^k \) \( i^{th} \) Server in \( k^{th} \) host type; \( i \in \{1..H_k\} \)

\( \text{util}\% \) is the server CPU core utilization
\[
\text{util}\% = \frac{\text{usedMIPS} \times 100}{\text{TotalMIPS}}
\]  

\( \text{usedMIPS} \) is the Processor core MIPS used by active runtime requests processed by the server core
\( = \text{TotalMIPS} - \text{AvailableMIPS} \)
TotalMIPS is the Total processor core MIPS
\[ \text{TotalMIPS} = \text{Cores}_k \times \text{MIPS}_k \]

AvailableMIPS is the Free processor MIPS

\[ \text{Power}_{ikj}^{ util\%} \]
\[ \text{is the Power consumption of server } S_i^k \text{ at CPU utilization}\% \text{ at processor frequency } f_j; j \in \{1..F_k\}; \]
\[ \text{util}\% \in \{0..100\}; \]

MIPS\(_k\) is the MIPS per CPU core of a Server of host type \( k \)

Cores\(_k\) is the number of cores of a Server of host type \( k \)

Reason to consider server power consumption at idle state (utilization=0\%) is that static power contributes around 60\% of the peak server power consumption.

### 6.2.2 Server cluster power efficiency model

Power efficiency of the server cluster to process a request is defined as the ratio between work-done to power consumed.

\[
\text{PE} = \frac{1}{(\text{Time}_{\text{AVG}} \times \text{Power}_{\text{AVG}})} \tag{6.4}
\]

Time\(_{\text{AVG}}\) Mean request response time (in seconds) for requests that complete during the course of the trace.

Power\(_{\text{AVG}}\) Mean power consumption (in watts) of the servers in the cluster during the course of the trace.

If this PE metric for a scheme with sleep state is greater than PE of a baseline, it indicates that the solution is useful [34]. To compare results between different schemes, a Normalized PE (NPE) [34] is used. NPE is defined as the ratio between power-efficiency (PE) for dynamic policy, say “DPM” scheme, to the PE for “No DPM” Always-On baseline scheme.

\[
\text{NPE}_{\text{DPM}}^{\text{No DPM AlwaysOn baseline}} = \frac{\text{PE}_{\text{DPM}}}{\text{PE}_{\text{No DPM AlwaysOn baseline}}} \tag{6.5}
\]
6.2.3 Server power state transition model

Possible server power states are: off, SETUP, idle and busy. Power state transitions such as off to idle, idle to busy, busy to idle, idle to off are termed as SETUP state transitions. Server power states such as off, SETUP are referred to as non-operational power states, idle and busy server power states are referred to as operational or active power states. Each state or transition is represented by a tuple of power consumption (Power) and time spent (Time) in the state or power consumption during transition to another state and the time spend in transition {Power, Time}.

6.2.4 Server power consumption model

In this work, physical server power consumption of different server configurations at peak, idle load scenario and transition power state scenario is captured from SPEC results [73] and is presented in Table 6.1.

<table>
<thead>
<tr>
<th>Host type k</th>
<th>Power</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_{i,k,j}</td>
<td>169</td>
<td>Peak Power consumption of S_{i,k} at jth frequency index</td>
</tr>
<tr>
<td>Pm_{i,k,j}</td>
<td>105</td>
<td>Idle Power consumption of S_{i,k} at jth frequency index</td>
</tr>
<tr>
<td>Power_{i,k,j}^{SETUP}</td>
<td>169</td>
<td>Power consumption of S_{i,k} at jth frequency index during power state transition (eg. off to idle etc..)</td>
</tr>
<tr>
<td>Power_{i,k,j}^{off}</td>
<td>0</td>
<td>Power consumption of S_{i,k} at jth frequency index in off state</td>
</tr>
</tbody>
</table>

Power_{i,k,j}^{0\%} is the server power consumption when no application is running in the server S_{i,k}, also known as idle power of the server with host type k and frequency f_{i,j}^{k}. Power_{i,k,j}^{100\%} is the server power consumption when the
server’s CPU utilization% = 100 of the server with host type $k$ and frequency $f_{j}^{k}$.

### 6.2.5 Total server cluster energy and power consumption model

As the hybrid approach uses a both DPM and DVFS techniques, the energy model accounts for power consumed when in operational (DVFS: busy, idle) power states and also non-zero power consumed in non-operational (DPM: off, SETUP) power states. SETUP transition power state captures one of the transitions such as: off to idle, idle to busy and idle to off.

Server cluster energy and power computation is as follows:

\[
E_{i}^{k} = \text{Total Energy consumed by the server } S_{i}^{k} = 0 \times \text{Time}_{off}^{ikj} + \sum_{k,j} \text{Power}_{SETUP}^{ikj} \times \text{Time}_{SETUP}^{ikj} + \sum_{\text{util}%,k,j} \text{Power}_{util%}^{ikj} \times \text{Time}_{util%}^{ikj}
\]

\[
E = \sum_{k=1}^{M} \sum_{i=1}^{H_{k}} E_{i}^{k}
\]

\[
\text{Power}_{i}^{k} = \text{Total Power consumed by the server } S_{i}^{k} = \frac{E_{i}^{k}}{\sum_{k,j} \text{Time}_{off}^{ikj} + \sum_{k,j} \text{Time}_{SETUP}^{ikj} + \sum_{\text{util}%,k,j} \text{Time}_{util%}^{ikj}}
\]

\[
\text{Power} = \sum_{k=1}^{M} \sum_{i=1}^{H_{k}} \text{Power}_{i}^{k}
\]

\[
\text{Power}_{SETUP}^{ikj} = \text{Power consumed by server } S_{i}^{k} \text{ in SETUP mode (power state transitions): host type } k \text{ and operating at frequency } f_{j}^{k};
\]

\[
\text{Time}_{SETUP}^{ikj} = \text{Time duration of server } S_{i}^{k} \text{ in SETUP mode (power state transitions) ; host type } k \text{ and operating at frequency } f_{j}^{k}
\]

\[
\text{Power}_{util%}^{ikj} = \text{Power consumed by server } S_{i}^{k} \text{ in busy or idle mode; host type } k \text{ and operating at a particular CPU utilization } \text{util}% \text{ and frequency } f_{j}^{k}
\]
Time\textsubscript{\text{util\%}}\textsuperscript{\text{ij}} Time duration of server S\textsubscript{i} in busy or idle mode; host type k and operating at a particular CPU utilization util\% and frequency f\textsubscript{j}\textsuperscript{k}

Time\textsubscript{\text{off-idle}}\textsuperscript{\text{ikj}} Transition time required to transition from off to idle power state of server S\textsubscript{i}k; host type k and frequency f\textsubscript{j}\textsuperscript{k}

Time\textsubscript{\text{idle-off}}\textsuperscript{\text{ikj}} Transition time required to transition from idle to off power state of server S\textsubscript{i}k; host type k and frequency f\textsubscript{j}\textsuperscript{k}

6.2.6 Server dynamic power management model

Server power state downgrade transition policy is applied when transitioning server from on (idle or busy) to off state. Physical server S\textsubscript{i}k is maintained in on (idle) state for a period referred to as break-even-time (BET) Time\textsubscript{\text{wait-off}}\textsuperscript{\text{ikj}} [15] before it can be switched off.

\[
\text{Time}_{\text{wait-off}}^{\text{ikj}} = \frac{\text{Time}_{\text{off-idle}}^{\text{ikj}} \times \text{Power}_{\text{SETUP}}^{\text{ij}}}{\text{Power}_{0\%}^{\text{ij}}}
\]  

(6.8)

Power consumption at different states is as follows:

Power\textsubscript{100\%}^{\text{ikj}} > Power\textsubscript{0\%}^{\text{ikj}} >> 0 \text{ Watts}

6.2.7 Server Dynamic voltage frequency scaling model

Dynamic voltage frequency scaling (DVFS) model is primarily adopted to improve upon performance of in-process requests with a trade-off between performance improvement and increase in energy/power consumption. In this work, webserver workload is used, hence request response times (make span) is considered more important as opposed to request throughput. SPEC benchmark results [73] is used to formalize the power modeling exercise at different server CPU utilization% values. Server CPU utilization is computed using Equation 6.3. Power consumption calculation model using Equation 6.9 is used to calculate server power consumption when CPU utilization% which is not covered in SPEC results [73].
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). Also, using SPEC results [73], a model to compute relative response time improvement factor due to concurrency level (different CPU utilization %) is proposed. This relative response time factor determines the relative delay imposed by the server of a host type in processing a request while operating at a particular utilization% and particular frequency. Figure 6.1 shows the response-time improvement ratio at different CPU utilization% between servers of different host types with respect to host type IBM XeonX5675 [73].

\[
\text{Power}_i^k = \text{Power}_{[\text{util}]}^{i,k,j} + \Delta \times \left( \frac{\text{util} - \lfloor \text{util} \rfloor}{100} \right) \quad (6.9)
\]

\[
\Delta = \text{Power}_{[\text{util}]}^{i,k,j} - \text{Power}_{[\text{util}]}^{i,k,j}
\]

Figure 6.1: Sample response time performance improvement factor [73]

6.2.8 System architecture

In this section, the high level description of the system architecture is provided. Goal is to meet request response time criteria while minimizing total energy consumption of the server cluster. To achieve this goal, following techniques are adopted: request batching, DPM controlled server duty cycle management, DVFS controlled run-time response time management and selection approach to arrive at the best or optimal performance to power server configuration. Figure 6.2 captures the system architecture.
Figure. 6.2: System architecture

When the server is active, either it is (a) in the running mode or is in the middle or processing jobs, or (b) in idle mode at the default frequency $f_1^k$ without processing any valid job, or (c) in idle mode at one of the possible frequencies $f_j^k; k \in \{1..M\} ; j \in \{1..F_k\}$ without processing any valid job. With DPM, the server can be transitioned between power states. However, there are overheads (power and SETUP transition time expend) to be accounted for. These transition expends are referred to as setup power and setup time.

System architecture comprises of the following modules:

- Performance to power configuration selector
- DVFS controller
- DPM controller
- Low power state transitioner
- Arbitrator

6.2.8.1 Performance to power configuration selector

Datacenter contains heterogeneous set of servers of different host type configurations. Each host type configuration has its own performance and power consumption characteristics. This module specifically identifies the host
type configuration that has the best (optimal) performance to power consumption ratio. Equation (6.1) is used to arrive at the server’s $\text{PPM}_{i}^{k}$ ratio. Using the right server with high PPM has a good degree of influence in reducing energy consumption of the datacenter.

6.2.8.2 Dynamic power management (DPM) controller

This module focuses on managing the servers’ power state transitions and answers the following questions: a) when should the server be switched on?, and b) when should the server be transitioned to low power state?.

Request batching approach in [29] is followed to answer the first question. Server CPU could be in either off or idle or busy state. Transition (off to idle) from CPU low power state (off) to high power operational state (idle) involves an overhead time expend value $\text{Time}_{\text{off-idle}}^{i,k,j}$ as captured in Table 6.2. System uses a batch timeout value to ascertain when to transition a server from low power CPU state to high power operational state. Batch timeout value is determined periodically based on request response time. If the system wide average request response time is over a threshold (SLA set point) limit, then batch timeout value is decremented by 1 second. If the system wide average response time is below the threshold limit, then the batch timeout value is incremented by 1 second. In this work, this threshold value is set as 100 seconds.

Server power state downgrade approach is discussed in Section 6.2.6, which answers the second question. System models discussed in Sections 6.2.4 and 6.2.5 are used to compute power and energy.
Table. 6.2: Physical server state sample transition times (in Seconds)

<table>
<thead>
<tr>
<th>State Transition Time</th>
<th>(1)</th>
<th>(2)</th>
<th>(3)</th>
<th>(4)</th>
<th>(5)</th>
<th>(6)</th>
<th>(7)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_{\text{off-idle}} )</td>
<td>50</td>
<td>45</td>
<td>50</td>
<td>45</td>
<td>45</td>
<td>60</td>
<td>75</td>
</tr>
<tr>
<td>( T_{\text{idle-off}} )</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

(1) HP ProLiantML110G3PentiumD930
(2) HP ProLiantML110G4Xeon3040
(3) HP ProLiantML110G5Xeon3075
(4) IBM XeonX3470
(5) IBM XeonX3480
(6) IBM XeonX5670
(7) IBM XeonX5675

6.2.8.3 Dynamic voltage frequency scaling controllers

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. There are few challenges that DVFS controller logic has to solve. CPU operating at higher frequency improves performance but also increases the power consumption. Whether this increases energy consumption needs to be ascertained. Also, with virtualization, any change to CPU frequency needs to be looked at from a holistic perspective. With many VMs hosted in a single server, any change to server frequency could impact requests processed in different VMs. Server frequency cannot be reduced without knowing how this change is going to impact performance of requests in the VMs. In this work, server is started (awakened) at the lowest possible frequency for the host type. When the server is in active or operational state, a process periodically checks if the system response time is greater than a threshold value. If the system response time is greater than the threshold value, it means that the system is either missing or about to miss SLA norms. When this happens, server frequency is upgraded to improve the performance with possibility of increase in power consumption. Another possibility is to initiate migration of request(s) to a server with higher performance or processing capability.
6.2.8.4 Server low power state transitioner

The function of this module is to transition a server from a high power state to low power state (both at DVFS and DPM controlled states). This function is initiated at a) after completion of each request, if the average response times of the system are lower than the threshold limit by a margin—this means that the system is performing far better than the requirement, and b) periodic control interval run of the power downgrade routine. Following constraints and action policy rules are followed when the power transitioner function is called upon:

- No request should be in-process in any of VMs hosted in the server
- On DVFS: Servers’ frequency is defaulted to the lowest frequency value. This change is applied only when the server is operating at a higher frequency value.
- On DPM: Downgrade policy discussed in Section 6.2.6 is adopted, using which the optimal time to wait \( \text{Time}_{\text{wait}} \) in higher power state before transitioning to a lower power state is determined for the server.

6.2.8.5 Arbitrator

In the system architecture, Arbitrator is the crucial intelligent module which orchestrates and controls functions of other modules. Web requests or workload from the clients are redirected, based on load balancing scheme.

In this work, a time-shared scheduling policy and First come first served (FCFS) servicing policy (Algorithm 6.1) is used. Generally, a trade-off exists between high performance and low power consumption: high performance means more power consumption and possibly more energy consumption. In the real problems, a performance demand criterion is more critical than power consumption.
Algorithm 6.1: Hybrid approach using PPM, DPM and DVFS controls

//PPM
1: Find server j’s PPM$^k_j$ using Equation (6.1)
2: Build hostList with servers sorted by decreasing PPM$^k_j$ value.
3: while workload request exists do //cloudlet request queue
   4: consider request from arbitrator queue
   5: foreach server in the hostList do
   6: If server is in operational state (i.e., busy or idle) then
   7: If server has VM available to process the request then
   8: Use DVFS controller to control performance and power (Section 6.2.8.3)
   9: Use this server and VM to process the request
   10: Proceed with next workload request
   11: endforeach
   //When there is no server in anyone of the busy or idle states or no available VMs in
   //busy or idle servers, then check whether there are any server in off state
12: foreach server in the hostList do
13: If server is in off state then
14: Use DPM controllers wake up batching routine (Section 6.2.8.2) to transition
   the server from off to idle state
15: Re-queue the request enabling it to be considered for processing again as and when the server transitions to idle state
16: endforeach
17:end while //cloud request

//power reduction reclaiming policy run
18:do after each request process completion in the VM (or) at select control epoch
19: If the VM does not have any in-process request and the $\text{Time}_{\text{AVG}} < \text{Time}_{\text{SETPOINT}}$ then
20: Initiate DPM controller’s downgrade policy routine (Section 6.2.8.4) to Transition from idle to off state
21:end do

6.3 SUMMARY

This Chapter, as a first step proposed a heuristic approach to account for server heterogeneity to rank energy efficient servers considering the server’s performance and server’s compute specific power consumption. This Chapter, as the second step, presented a dynamic power management (DPM) server state transition control approach and dynamic voltage frequency scaling (DVFS) control approach to process workload requests in specific servers based on the ranks from first step, using specific VM which has capacities to handle the workload request. Finally, the evaluation of all the proposed algorithms is elaborated in Chapter 7.