CS528
Energy-Aware Scheduling for Real-Time Tasks in Cloud Data Centers

A. Sahu
Dept of CSE, IIT Guwahati
Reference

Outline

• Taxonomy of Power/Energy Consumption Model

• Power Aware Computing, Thermal Aware Computing

• Power Aware Scheduling in Cloud

• Migration and Management
  – Work Consolidation or VM Consolidation
  – Reduce number of active machine and run at critical frequency
Basic of Energy Aware RTS in Cloud

• Developing EA cloud data centers
  – not only can reduce power electricity cost
  – but also can improve system reliability.

• Real time task : task completion before deadline, QoS (weak term -reliable)

• Energy Saving using
  – Resource scaling up and scaling down strategies

• Rolling Horizon
Scheduling Model

• Given a virtualized data center that is characterized by an infinite set of physical computing hosts
  \[ H = \{ h_1, h_2, \ldots \} \]

• Hardware infrastructure \( H \)
  – for creating virtualized resources
  – to satisfy users’ requirements.

• Active host set \( H_a \) with \( H_a \subseteq H \).

• Host \( h_k \) is characterized by \( h_k(c_k, r_k, n_k) \)
  – Compute power in MIPS, ram size, net bandwidth
Scheduling Model

• Host $h_k$ is have set $V_k$ of virtual machine

$$V_k = \{ v_{1k}, v_{2k}, \ldots, v_{|V_k|_1} \}$$

• For a VM $v_{jk}$ is characterized by $c(v_{jk}), r(v_{jk}), n(v_{jk})$
  – Fraction of Compute power in MIPS, ram size, net bandwidth allocated to $v_{jk}$

• Multiple VMs can be dynamically
  – Started and stopped on a single host
  – Based on the system workload.

• At the same time, some VMs are able
  – To migrate across hosts in order to consolidate resources
  – And further reduce energy consumption.
Scheduling Architecture

VM Adjustment Information

Status Information

Real-Time Controller

VM Controller

Rolling Horizon

Users

New Tasks

Rejected Task

Scheduler

Task
Working of RH Scheduler

• Scheduler consists of
  – Rolling-horizon, Real-time Controller, VM Controller.

• Scheduler work
  – Takes tasks from users and
  – Allocates them to different VMs.

• Rolling-horizon holds
  – Both new tasks and waiting tasks to be executed.

• A scheduling process is triggered
  – By new tasks, and all the tasks of RH to rescheduled
Working of RH Scheduler

• Scheduler consists of
  – Rolling-horizon, Real-time Controller, VM Controller.

• Scheduler work
  – Takes tasks from users and
  – Allocates them to different VMs.

• Rolling-horizon holds
  – Both new tasks and waiting tasks to be executed.

• A scheduling process is triggered
  – By new tasks, and all the tasks of RH to rescheduled
Scheduling ()@New Task Arrives

• **Step 1.** Scheduler checks System status information such as
  – running tasks’ remaining execution time, active hosts, VMs’ deployments,
  – Tasks in waiting pool including their deadlines,
  – Currently allocated VMs, start time, etc.

• **Step 2.** Sort the tasks in rolling-horizon
  – by their deadlines to facilitate scheduling operation.
Scheduling ( )@New Task Arrives

• **Step 3.** Real-time controller determines
  – Whether a task in RH can be finished before its deadline.

• The VM controller adds VMs
  – to finish the task within timing constraint
  – if current VMs cannot finish it successfully.
  – If no schedule can be found to satisfy the task’s timing requirement although enough VMs has been added by testing

• The task will be rejected. Or the task will be retained in the rolling-horizon.
Scheduling ()@New Task Arrives

- **Step 4.** Update the scheduling decision for the tasks in rolling-horizon,
  - Their execution order, start time,
  - Allocated VMs and new active hosts.

- **Step 5.** When a task in the rolling-horizon is ready to execute
  - dispatch the task to assigned VM.
Scheduling ()@New Task Arrives

• Additionally, when
  – tasks arrive slowly, tasks have loose deadlines or their count is less, making system workload light

• VM controller considers both
  – the status of active hosts and task information, and

• VM Controller then decides
  – Whether some VMs should be stopped or migrated to consolidate resources
  – So as to save energy
Task Model

- A set $T = \{t_1, t_2, \cdots \}$ of independent tasks that arrive dynamically.

- A task $t_i$ submitted by a user have

  $t_i = \{a_i, l_i, d_i, f_i\}$ Where $a_i, l_i, d_i$ and $f_i$ are
  - Arrival time, task length/size, deadline, and finish time of task $t_i$

- Let $rt_{jyk}$ be the ready time of VM $v_{jk}$ at host $h_k$.

- $st_{ijk}$ be the start time of task $t_i$ on VM $v_{jk}$.

- Execution time of task $t_i$ on VM $v_{jk}$.

  $et_{ijk} = \frac{l_i}{c(v_{jk})}$. c(v): compute capacity MIPS of VM
Task Model

• Finish time of task $t_i$ on $v_{jk}$, $ft_{ijk} = st_{ijk} + et_{ijk}$

• Boolean $x_{ijk}$ reflect mapping of tasks
  – to VMs at different hosts in a virtualized Cloud data center,
    
    $x_{ijk} = 1$ if task $t_i$ is allocated to VM $v_{jk}$ at host $h_k$
    
    $= 0$, otherwise.

• Task's timing constraint can be guaranteed

    $x_{ijk} = \begin{cases} 
    0 & ft_{ijk} > d_i \\
    0 \text{ or } 1, & ft_{ijk} \leq d_i 
    \end{cases}$
Energy Consumption Model (ECM)

• EC by hosts in a data center
  – is mainly determined by CPU, memory, disk storage and network interfaces,
  – in which CPU consumes major part of energy
• CPU EC: static part \( (E_s) \) + dynamic part \( (E_d) \)
  – \( E_d \) is dominant > 80%, \( E_s \) follows similar trend to \( E_d \)
• EC of running task \( t_i \) on VM \( v_{jk} \): \( ec_{ijk} = ecr_{jk} \cdot et_{ijk} \)
  – Term \( ecr_{jk} \): EC rate of the VM \( v_{jk} \)
ECM of Tasks on VMs

- Total EC by executing all the tasks is

\[
ec^\text{exec} = \sum_{k=1}^{|Ha|} \sum_{j=1}^{|Vk|} \sum_{i=1}^{T} x_{ijk} \cdot ec_{ij_k}
\]

\[
= \sum_{k=1}^{|Ha|} \sum_{j=1}^{|Vk|} \sum_{i=1}^{T} x_{ijk} \cdot ecr_{j_k} \cdot eti_{jk}
\]

- \(Ha\)=active hosts, \(Vk\)=VM of host \(k\), \(T\)=task set

- EC is incurred when VMs are sitting idle
  - All the VMs of a host is idle
  - Some of VM of a host Idle

- EC considering the execution time and idle time

\[
ecei = ec^\text{exec} + ec^\text{allIdle} + ec^\text{partIdle}
\]
ECM of host with idle VMs

- EC when all the VMs of a host is idle
  - Host can be set to a lower EC rate by DVFS
  - ECR of VM $v_{jk}$ by $ecr_{idle_{jk}}$
  - Idle time when all the VMs in a host $h_k$ are idle is $it_k$

$$ec^{allIdle} = \sum_{k=1}^{Ha} \sum_{j=1}^{V_k} ecr_{idle_{jk}} \cdot it_k$$

- EC when some of VM of a host Idle

$$ec^{partIdle} = \sum_{k=1}^{Ha} \sum_{j=1}^{V_k} ecr_{jk} \cdot t_{j^{partIdle}}$$

with $t_{j^{partIdle}} = \text{max}(f_i) \cdot it_k - \sum_{i=1}^{T} x_{ijk} \cdot eti_{jk}$
Final ECM of host

• Although some VMs are placed on a host
  – maybe some resource is still unused
  – However, the resource also consume energy.

• Suppose there are $s$ periods in each which the count of VMs in host $h_k$ is different from another.

• Let $t_p$ to denote the time in the period $p$

• EC due to unused resources of hosts

\[ ec_{\text{unused}} = \sum_{k=1}^{|H_a|} \sum_{p=1}^s \left( ecr(h_k) - \sum_{j=1}^{V_k(p)} ecr_{jk} \right) \cdot t_p \]

• So total EC of Cloud system

\[ ec = ecei + ec_{\text{unused}} = ec^{\text{exec}} + ec^{\text{allIdle}} + ec^{\text{partIdle}} + ec_{\text{unused}} \]
Scheduling Goals and Trade-offs

• Less running hosts ==> 
  – less consumed energy
  – may greatly affect guarantee ratio of real-time tasks

• Energy Conservation and TGR are two conflicting objectives

• Good scheduling strategy makes a good trade-off by dynamically
  – Starting hosts, Closing hosts
  – Creating VMs, canceling VMs
  – Migrating VMs
  – according to the system workload.
Energy Aware Rolling Horizon Scheduling

• Traditional scheduling: once a task is scheduled
  – it is dispatched immediately to the local queue of a VM or a host

• In EARH: puts all the waiting tasks RH queue
  – Their schedules are allowed to be adjusted for the schedulability of the new task
  – possibly less energy consumption.

• Essential advantage of RH optimization
  – task migration required by rescheduling does not yield any overhead
  – as all the tasks are waiting in the rolling-horizon.
EARTH Approach

- Attempt to append new task to the end of former allocated tasks on a VM.
- Start time $st_{ijk}$ of task $t_i$ on VM $v_{jk}$
  \[
  st_{ijk} = \max\{rt_{jk}, a_i\}, \text{rt}_{jk} \text{ is ready time of } v_{jk}
  \]
- Ready time get updated once task $t_p$ added to $v_{jk}$
  \[
  rt_{jk} = st_{pjk} + et_{pjk}
  \]
- When a task cannot be successfully allocated in any current VM
  - scaleUpResource() is done to create a new VM
  - With the goal of finishing the task within its deadline.
Scale up resources: in three steps

• Step 1: Create a new VM
  – in a current active host without any VM migration

• Step 2: If Step 1 fails
  – Migrate some VMs among current active hosts
  – To yield enough resource on a host and
  – then create a new VM on it

• Step 3: If Step 2 fails
  – start a host and then create a new VM on it.
Scale up resources: in three steps

• **Some terms**
  
  – $st(h_k)$: start-up time of host $h_k$
  
  – $ct(v_{jk})$: creation time of VM $v_{jk}$
  
  – $mt(v_{jk})$: migration time of VM $v_{jk}$

  $v_{jk} = r(v_{jk})/n(v_{jk})$,

  RAM and Network

• Using different steps products different start times for a task,

  $$st_{ijk} = \begin{cases} 
  a_i + t(v_{jk}), & \text{if setp1,} \\
  a_i + ct(v_{jk}) + \sum_{p=1}^{p} mt(v_{pk}), & \text{if setp2,} \\
  a_i + st(h_k) + ct(v_{jk}), & \text{if setp3.} 
  \end{cases}$$
Scheduling EARH

\textbf{for} each task $t_i$ in set $Q$ \textbf{do}

\text{findTag} \leftarrow \text{FALSE}; \text{findVM} \leftarrow \text{NULL};

\textbf{for} each VM $v_{jk}$ in the system \textbf{do}

Calculate the start time $st_{ijk}$ and execution time $et_{ijk}$

\textbf{If} $st_{ijk} + et_{ijk} \leq d_i$ \textbf{then} \text{findTag} \leftarrow \text{TRUE}; \text{Compute } ec_{ijk}$

\textbf{if} \text{findTag} == \text{FALSE} \text{ scaleUpResource();}

\textbf{if} \text{findTag} == \text{TRUE} \text{ then}

Select $v_{sk}$ with min energy consumption to execute $t_i$

\text{findVM} \leftarrow v_{sk}$

\textbf{else} Reject task $t_i$

Update scheduling decision of $t_i$ and remove it from $Q$