CS528

Workload Prediction and Budget Aware Scheduling

A Sahu

Dept of CSE, IIT Guwahati

Outline

- Work load Prediction
  - EWMA, FUSD
- Dynamic Resource Management
  - Scale up and Scale Down
- SLAV

Reference

F. Farahnakian et. al, "**Energy Aware VM Consolidation in Cloud using Prediction Model",** IEEE Trans. On Cloud Computing, June 2019

Energy Aware VM Consolidation in Cloud using Prediction Model

- Virtual Machine (VM) consolidation
  - Promising approach to save energy and
  - improve resource utilization in data centers.
- Many heuristic algorithms for VM consolidation as
  - Vector Bin-Packing Problem.

VM consolidation: Vector Bin Packing

- Given A set of N VMs with resource requirements (r_c, r_m, r_{db}, r_{nb}, etc)
  - CPU, memory, disk BW, net BW, etc
- Given a set of homogenous host/machines with capacity (C_c, C_m, C_{db}, C_{nb}, etc)
  - CPU Capacity, memory, disk BW, net BW available
- Pack this VMs to minimum number of host
- Simple examples: 2D case or 2 resources case
  - 10 VM with CPU and Mem requirement $vm_i(r_c, r_m)$ need to map to hosts with 4CPU+4GB of RAM
  - Minimize number of CPU

Energy Aware VM Consolidation in Cloud using Prediction Model

- Focused mostly on number of active PM minimization **//Static Problem**
  - According current resource requirements and neglected the future resource demands.
- So, they generate unnecessary VM migrations
  - increase the rate of SLA violations in data centers.
- Needs VM consolidation approach
  - That takes into account both the current and future utilization of resource
- Simple regression-based model may be enough
  - to approximate the future CPU and memory utilization of VMs and PMs

Utilization Prediction-aware VM Consolidation (UP-VMC)

- VM consolidation as 2D vector bin packing
- UP-VMC consider
  - CPU utilization and Memory utilization
  - Also, considers current & future resource utilization
- Approximate the future utilization
  - two regression-based prediction models
  - linear prediction model and k-nearest neighbor.
- Prediction model in order to predict
  - Resource utilization of VMs;
  - resource utilization of PMs;
  - resource utilization of both VMs and PMs.

UtilPred-VM Consolidation

- VM selection methods: affect on performance
  - in terms of the **energy consumption**,
  - the number of **SLA violations** and
  - the **number of migrations**
- Performance of VM consolidation is increased
  - Selects a VM that requires the minimum time for migration to another PM

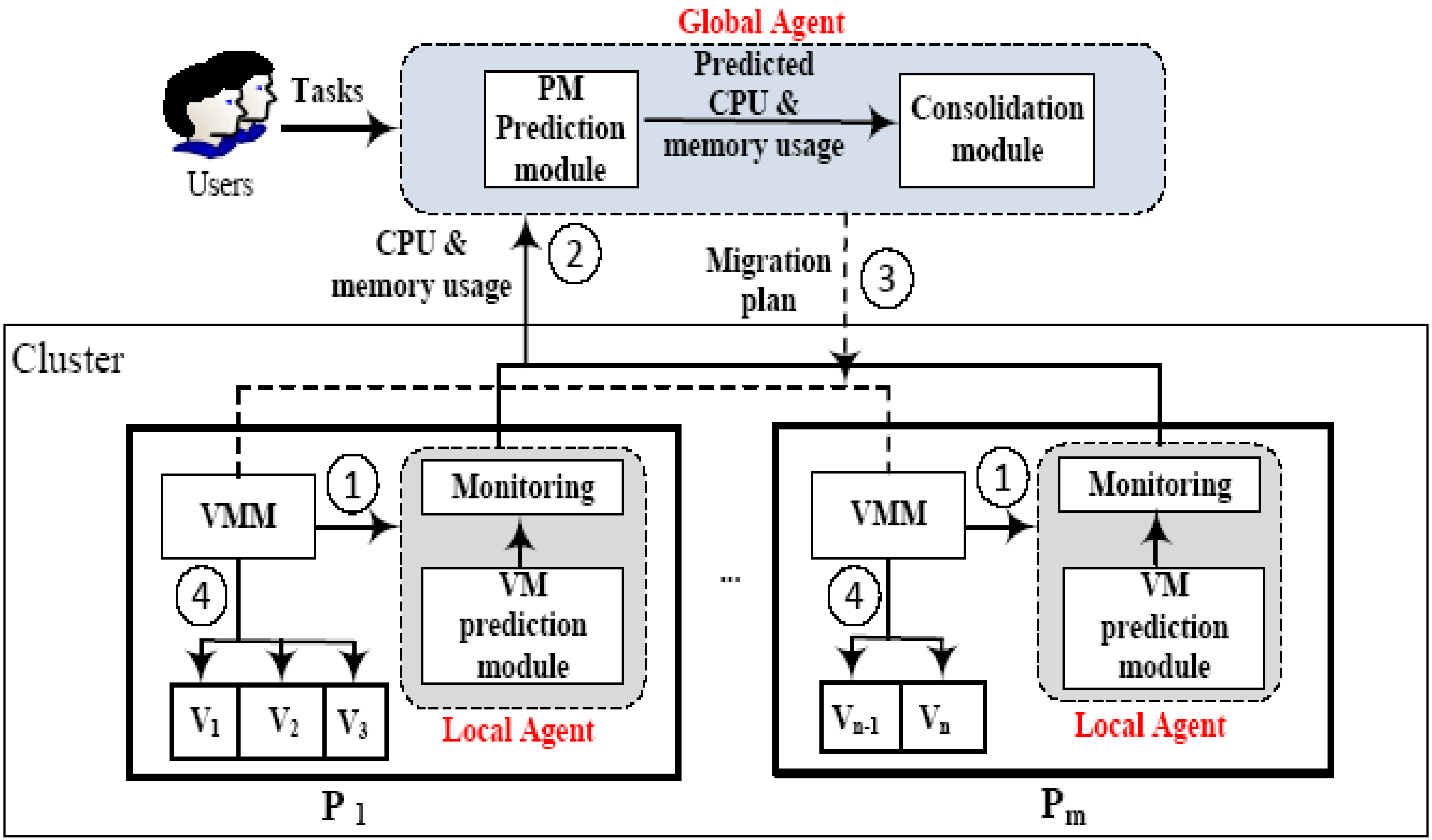
System Architecture

- A data center consists of **m** heterogeneous PMs $\mathbf{P = \{ p_1, p_2, \ldots, p_m\}}$.
- Each PM with ***D*** type of resources
  - CPU, memory, network I/O and storage capacity.
- Multiple VMs can be allocated to each PM
  - through Virtual Machine Monitor (VMM).
- At any given time, users submit their requests
  - for provisioning of ***n*** VMs, $\mathbf{V = \{v_1, v_2, \ldots, v_n\}}$
  - which are allocated to the PMs.

System Architecture

- As requested utilization of VMs and PMs vary over time
  - An initial efficient allocation approach needs to be augmented
  - with a VM consolidation algorithm that can be applied periodically.
- In order to adapt and optimize
  - the VM placement periodically according to the workload.
  - We need to monitor and predict the workload of all the VMs and PMs
  - **Local agent at PM, Global agent at scheduler level**

System Architecture



System Architecture

- **System Arch. consists of two kind of agents:**
  - Fully distributed Local Agents (LAs) in PMs
  - Global Agent (GA) resides in a master node
- **Each LA monitor and predict**
  - Monitors the current resource utilization of all VMs in a PM periodically.
  - Approximates the future utilization of all VMs in a PM using a regression-based prediction model
- **GA collects info. LAs**
  - to maintain the overall view of current and future resource utilization of VMs.
  - GA builds a **global best migration plan** to optimize VM placement

Solution Approaches

- Each PM with **d** type of resources
  - **CPU, memory**, network I/O and storage capacity.
- Each PM ***p*** has a ***d***-dimensional total capacity vector

$$\mathbf{Cp = < C^1_p, C^2_p , . . . \ C^d_p>}$$

  - where C^d_p is total d_{th} resource capacity of PM p.
- Used capacity vector of the PM **p** as

$$\mathbf{U_p =<U^1_p, U^2_p, . . . , U^d_p>}$$

  - where U^d_p denotes used capacity of the resource type d : simplicity **d=2, CPU, memory**
- For instance, used CPU capacity of a PM is estimated
  - as the sum of the CPU utilization of the three VMs if three VMs are hosted by the same PM

Solution Approaches

- Each VM ***v*** has a ***d***-dimensional total capacity vector

$$Cv = < C^1_v, C^2_v, \ldots\ C^d_v >$$

  - where C^d_v is total d_{th} resource capacity of VM v.
- Used capacity vector of the VM **v** as

$$U_v = < U^1_v, U^2_v, \ldots, U^d_v >$$

  - where U^d_v denotes used capacity of the resource type d : simplicity **d=2, CPU, memory**
- As the resource utilization of VMs
  - vary over time due to dynamic workloads,
  - the VM placement need to be optimized periodically

Solution Approaches : Step 1

- Aims to migrate some VMs from
  - **over-loaded PMs and**
  - **predicated over-loaded PMs.**
- If at least one resource (i.e., CPU or memory)
  - Exceeds total capacity, **PM is over-loaded**
  - Belongs to **overloaded PMs set** (P_{over}).
- If at least one resource predicted utilization value
  - is larger than capacity, **PM is predicted overloaded**
  - Belongs to **Predicted over-loaded PMs set** ($P^{\wedge}_{over}$)

Prediction Model

- Predicted Util Vector of PM: $\boldsymbol{PU_{p_{de}}} = \propto + \boldsymbol{\beta}\ \boldsymbol{U_{p_{de}}}$
  - Current used capacity vector : $U_{p_{de}}$
  - $\propto$ and $\boldsymbol{\beta}$ derived using linear regression
- Regression coefficients can be estimated

$$\boldsymbol{\beta} = \frac{\sum_{i=1}^{n}(Xi-Xb)(Yi-Yb)}{\sum_{i=1}^{n}(X_i-Xb)^2}$$

$$\propto = Y^b - \beta X^b$$

  - where X^b is the mean value of $X_1, X_2, \ldots X_n$, and
  - Y^b is the mean value of $Y_1, Y_2, \ldots, Y_n$

Prediction Model

- Predicted Util Vector of VM: $\boldsymbol{PU_v = \propto + \beta\, U_v}$
  - Current used Util vector : U_v
  - $\propto$ and β derived using linear regression
- PM load $Load_p = \sum_{d \in \{1,2,..,|D|\}} R_p^d$
  - where $R_p^d = \frac{U_p^d}{C_p^d}$ where U, C are Utilized & Capacity
- VM load $Load_v = \sum_{d \in \{1,2,..,|D|\}} R_v^d$ where $R_v^d = \frac{U_v^d}{C_v^d}$

Constraints on Consolidation

- Constraint 1 : Used Capacity of destination and added with used capacity of VM should be less than threshold

$$U_{p_{de}} + U_v \leq T.C_{p_{de}}$$

- Constraint 2 : Predicted Capacity of destination and Predicted capacity of VM should be less than threshold

$$PU_{p_{de}} + PU_v \leq T.C_{p_{de}}$$

- **Both should hold**

Scale up and Scale Down

- Scale UP: demand is high
  - If required switch on more PM to serve better
- Scale down : Demand is less
  - If require power off some PM to save power

Algorithm : Consolidation and Scale Up

Set $M_1 = \Phi$;

for p_{so} in P_{over} U $P\hat{}_{over}$ do **//Over loaded**

Sort VMs V_m on PM p_{so} in ascending order based on U_{mem}

for v in V_m do

for p_{de} in $P - (P_{over}$ U $P\hat{}_{over}$) do **//Non-overloaded**

if $U_{p_{de}} + U_v \leq T.C_{p_{de}}$ & $PU_{p_{de}} + PU_v \leq T.C_{p_{de}}$

$M_1 = M_1$ U [(p_{so}, v, p_{de})]

Update Up_{so} and Up_{de} ; break

if not able to find any destination PM

Switch on the dormant PM p; **//Scale UP**

Algorithm : Consolidation-Scale Down

Sort P_{active} in descending of $Load_p$;

for $\mathbf{PM_{so}} = |P_{active}|$ to 1 do **//Start from Light loaded one**

$\mathbf{V_m}$ = sort VMs on $\mathbf{PM_{so}}$ in descending order of $Load_v$;

Set $M_2 = \Phi$;

for v in V_m do

success=false;

for $\mathbf{p_{de}}$ in $\mathbf{P_{active} - PM_{so}}$ do

if $U_{p_{de}} + U_v \leq T.C_{p_{de}}$ & $PU_{p_{de}} + PU_v \leq T.C_{p_{de}}$

$M_2 = M_2$ U [(p_{so}, v, p_{de})] ; success=True;

Update Up_{so} and Up_{de} ; break

if success = false; Recover U_{pso} and $U_{pde;}$ $M_2 = \Phi$;

else **Switch PMso to the sleep mode**

Performance Metrics

- **SLA Violation (SLAV)** **SLAV = SLAVO * SLAVM**
  - **due to Overload (SLAVO),**
  - **due to Migration (SLAVM)**
- **SLAVO** $= \frac{1}{M}\sum_{i=1}^{m}\frac{T_{s_i}}{T_{a_i}}$
  - M number of PM, T_{si} total time PM I experienced CPU/Mem utilization above 100%
- **SLAVM** $= \frac{1}{N}\sum_{j=1}^{n}\frac{C_{d_j}}{C_{r_j}}$
  - Experience performance degradation of $_j$ the VM by migration C_{rj} total capacity requested by VM

Reference:

Wu et.al*, End to End Delay Minimization for Scientific Workflow in cloud under Budget Constraints,* IEEE Trans. On Cloud Computing. 2015.

Introduction

- With emergence of cloud computing and rapid deployment of cloud infrastructures
  - Number of **scientific workflows** have been shifted to cloud environments.
- Challenges:
  - Reducing financial cost in addition
  - to meeting the traditional goal : performance
- Quick evaluation of scientific workflow
  - to minimize the workflow end-to-end delay under a user-specified financial constraint

Scientific Workflow : SWF

- Large-scale scientific computing tasks
  - for data generation, processing, and analysis are
  - often assembled and constructed as **Workflows**
  - comprised of many interdependent **modules**
- Workflow (WF) module communicates
  - with others through the sharing of data sets,
  - which are either stored in shared file system or
  - transferred from node to node by WF management system
- Scientific Workflows are
  - typically executed in a distributed manner
  - in heterogeneous network environments

WF in Clouds System

- It is essential construct analytical models
  - to quantify the network performance of scientific workflows
  - in IaaS cloud environments,
  - and formulate a task scheduling problem
- Scheduling Problems: WF on Cloud
  - to minimize the workflow end-to-end delay
  - under a user-specified financial cost constraint,
- Referred to as Minimum End-to-end Delay under Cost Constraint (MED-CC)

Workflow Execution in Cloud

- Workflow is represented as
  - a directed acyclic graph (DAG),
- Submitted to the workflow engine
  - for executing, scheduling, tracking and reporting
- Workflow (WF) have independent tasks (Work/W), and
  - Execution model with inter-module dependencies
  - Identify and quantify the key financial and time cost

Cost Model : time and financial

- **Time Cost or simply Time :** overall time to execute Work W_i on VM_j

$$T_{i,j}=T(I_j)+T(E_{i,j})+T(R_i)$$

  - $T_{i,j}$= overall time to execute Work W_i on VM_j
  - $T(I_j)$ = Startup Time for VM_j
  - $T(E_{i,j})$ = time to execute W_i on VM_j
  - $T(R_i)$ = time of upload/download data from/to VM_j

Cost Model : time and financial

- **Financial Cost (or simply Cost): O**verall time to execute Work W_i on $\mathbf{VM_j}$

$$C_{i,j}=C(I_j)+C(E_{i,j})+C(R_i)+C(S_i)$$

  – $C(S_i)$ = data storage cost of W_i
  – $C(I_j)$, $C(E_{i,j})$, $C(R_i)$ cost of Init VMj, execution of Wi on VMj and download/upload data to/from VM_j for W_i
- Set of Available VM type $VT=\{vt_0,vt_1,..vt_{n-1}\}$
  – **vt_j have processing power p_j and cost c_j**
- Cost of executing W_i on VM_j

$$C(E_{i,j})=T(E_{i,j})*c_j$$

Modeling Workflow Exe in Cloud