1. System Model

1.1. Cloud Model

Cloud providers offer a variety of heterogeneous VMs with different processing speed. In the proposed work, we consider the cloud server which consists of a set of dissimilar VMs. VMs are independent and fully connected with mesh topology in the cloud server. Each VM has its own computation power measured in terms of million instruction per second (MIPS). VMs may reside with in one or more cloud server. A global cloud manager maintains all VMs information. It tracks all status of deployed VMs. The cloud manager can be inside or outside of the cloud server. The user submit workflow application to cloud manager consist of different heterogeneous tasks. Then it splits the workflow application into different subtasks and arrange them accordingly without violating dependency constraint among the tasks and schedule them to available VMs in cloud servers.

1.2. Application Model

Task Scheduling with Dependency Constraint or Workflow Scheduling is Task Scheduling with a partial-order $\prec$ given with the problem.

If for tasks T_i and T_j, $T_i \prec T_j$ then task T_j cannot start execution on any VMs until the task T_i finishes its execution. Also $\prec$ is a transitive relation. Generally, this dependency is represented by a Directed Acyclic Graph also called Task Dependency Graph (TDG), in which nodes represent individual tasks, and a directed edge from T_i to T_j represents the partial-order $\prec$ in graph. These types of problems are real world problems and used in various scientific and industrial applications. There is a cost associated with every partial order $\prec$ there is a cost associated for communication of data between two tasks. This cost can be represented by a communication matrix. Where $comm(i, j)$ will be the amount of communication delay which will be needed for the data to reach the task T_j from task T_i, also task T_j cannot start execution untill it gets this data, and if both T_i and T_j are allocated on the same VMs then the communication cost will be zero, because the data will remain with the same VM. Multiple tasks can be depended to the same task in the TDG. The dependency between tasks are represented in Task Dependency Graph, where edges are labled with the communication costs.

Task without any precedence is called t^{entry} and tasks without any successor is called t^{exit}. If there are more then one entry and exit tasks in the workflow application then a new psuedo nodes is created with zero computation and communication. It can be represented as t^{entry} and t^{exit}. Then all the entry and exist tasks are connected with to respective t^{entry} and t^{exit}.

1.3. Terminologies

1.4. Execution Time (ET)

Here, we are considered performance variance of VMs to measure the perfect CPU cycles for the execution of the task on the VMs. The main motive is to considered the variability because of different processing speed of the VMs as well as shared cloud

infrastructure. As per the Amazon's EC_2 overall performance variability is 24%. Then the execution of the task t_i on the vm_j can be mathematically represented as below.

$$ET_i^j = \frac{len(t_i)}{VM\ S peed * (1 - pv_j)} \tag{1}$$

Where ET_i^j is the execution of task t_i on VM_j and len(t_i) is the length of the tasks in terms of million instruction. $VM\ S peed$ is the speed of the VM in terms of the million instruction per second. pv_j is the performance variability due to different processing speeds of the VM, it is almost 24% of most of the models [**? ?**]

1.4.1. Expected Execution Time (EET)

The major difference is here the same task may take different execution times on different VMs because of various resources, CPU frequencies and memory structures hence it addresses a more realistic scenerio. The execution time of each task on each machine is stored in a matrix called Expected Execution Time or EET matrix. Here, $EET_{(i,j)}$ will give the execution time of task T_i if it runs on VMs VM_j. Expected Execution Time includes the communication time and execution time of the tasks. Communication time is represented in the form of upper triangle matrix as mentioned below.

$$comm_{(i,j)} = \begin{array}{c} \\ T_1 \\ T_2 \\ \vdots \\ T_{x-1} \\ T_x \end{array} \begin{array}{c} \begin{array}{cccccc} T_1 & T_2 & T_3 & ... & T_{x-1} & T_x \end{array} \\ \begin{bmatrix} 0 & CT_{12} & CT_{13} & \dots & CT_{1(x-1)} & CT_{1x} \\ 0 & 0 & CT_{23} & \dots & CT_{2(x-1)} & CT_{2x} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & CT_{(x-1)x} \\ 0 & 0 & 0 & \dots & 0 & 0 \end{bmatrix} \end{array}$$

Let consider $EET_{(i,j)}$ can be represented in the form of matrix as mentioned below, where $1 \leq i \leq x$ and $1 \leq j \leq x$; x and y are number of tasks and VMs respectively.

$$EET_{(i,j)} = \begin{array}{c} \\ T_1 \\ T_2 \\ \vdots \\ T_x \end{array} \begin{array}{c} \begin{array}{ccccc} VM_1 & VM_2 & ... & VM_{y-1} & VM_y \end{array} \\ \begin{bmatrix} EET_{11} & EET_{12} & \dots & EET_{1(y-1)} & EET_{1y} \\ EET_{21} & EET_{22} & \dots & EET_{2(y-1)} & EET_{2y} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ EET_{x1} & EET_{x2} & \dots & EET_{x(y-1)} & EET_{xy} \end{bmatrix} \end{array}$$

1.4.2. Earliest Start Time (EST)

The $EST_{(i,j)}$ is the time of task t_i, $1 \leq i \leq x$ on VM_j,$1 \leq j \leq y$ on VM_j to start its execution on the VM_j. For this you need to select one of the following two statements

1. Minimum available time of VMs i.e ($Avail(VM_j)$).

2. Maximum of addition of the execution time of all parent tasks of the current task T_i, $1 \leq i \leq x$ and communication cost from parent tasks to the current task T_i.

It can mathematically represented as follows.

$$EST_{(i,j)} = \begin{cases} 0, & \text{if } t_i = t^{entry} \\ Max\{(Avail(VM_j)), & \\ max_{(t_j)pred(t_i)}\{AFT(t_j) + Comm(T(i,j))\}\}, & \text{if } t_i \neq t^{entry} \end{cases} \quad (2)$$

Where Max is the maximum of all this value, Here, Avail (VM_j) is to represent the j^{th} VMs,$1 \leq j \leq y$, $AFT(t_j)$ is the actual finish time of t_j and $comm(i, j)$ will be the amount of communication delay which will be needed for the data to reach the task T_j from task T_i

Remarks 3.1 : Earliest Start Time of all t^{entry} tasks on any VM is zero. i.e $t^{entry} = 0$

Remarks 3.2 : Communication cost of any precedence task T_{vm_i} to current task T_i is not consider if both are assigned on the same VMs then the communication cost will be zero, because the data will remain in the same VMs.

1.4.3. Completion Time (CT)

The sum of the estimated start time and execution time of the task t_i on VM_j, $1 \leq i \leq x$;$1 \leq j \leq y$.

$$CT_{(i,j)} = EST_{(i,j)} + ET_{(i,j)} \quad (3)$$

1.4.4. Earliest Completion time (ECT)

Earliest completion time of t_i on Virtual machine VM_j. It can mathematically calculated as follows.

$$ECT_{(i,j)} = min\{CT_{(i,j)} | \forall vm_j \in VM\} \quad (4)$$

1.5. Makespan (MKS)

To get more reality in the cloud server we have considered starting and finishing time that is called as booting time and shut down time of the VMs. It is the time required to schedule all the tasks of the workflow application on the virtual machines. It is represented as follows.

$$MKS = \alpha + Max(vm_j) + \beta \quad (5)$$

Where α and β are denoted as booting and shutdown time of the VMs. Makespan includes sum of booting time , shutdown time and VM working time of all VMs. here, booting time and shutdown time are required one time only, why because first VM booting time and last VM shutdown time will contribute to makespan and remaining time will be overlapped with other working events. Where α and β considered as 0.5 unit time each [**? ?**].

1.6. Load balancing (LB)

Tasks should be distributed among heterogeneous VMs in the cloud server. So that all the VMs can be equally distributed as much as possible. Load is measured only based on the Active Time (AT), if all tasks are almost equally distributed among

available VMs, variance among the all VMs will be less. To measure variance among all VMs we calculated standard deviation (SD) of AT of VMs.

$$AT(VM_j) = \sum_{i=1}^{x} EET(i,j) * \delta(i,j) \tag{6}$$

Where,

$$\delta(i,j) = \begin{cases} 1, & \text{if } t_i \text{ is assigned to } VM_j \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

The average value of AT of all the VMs, it can be calculated as follows

$$Avg(AT) = \frac{\sum_{j=1}^{y} AT(VM_j)}{y} \tag{8}$$

$$SD = \sqrt{\frac{\sum_{j=1}^{y} (AT_j - Avg(AT))^2}{y}} \tag{9}$$

Remarks 3.3 : If SD is minimized then load balance is maximized. If SD of load on all VMs is zero then load balancing is 100% and that means load distributed among all VMs are equal.

1.7. Energy Consumption (EC)

The energy consumption of cloud environment includes the energy consumption of server, memory, network and Input Output devices. As the servers consumes more energy compare to other devices such as memory, network and Input Output devices etc. So we are considering only server's energy consumption. They energy consumption can be categorized into two ways i.e. static energy (S_{eng}) and dynamic energy (D_{eng}) consumption. Static energy is the energy consumed by the idle servers of cloud data center. Dynamic energy is the energy consumed by the application executing on cloud data centers based on the speed of the VMs.

$$T_{eng} = D_{eng} + S_{eng} \tag{10}$$

Where T_{eng}, D_{eng} and S_{eng} are the total energy consumption of cloud server and dynamic energy consumption and static energy consumption respectively.

Here, Dynamic energy mainly depends on the voltage and frequency of the particular VM. Static energy consumption works with lowest frequency and voltage pair i.e (f_{low}, v_{low}). In the idle period minimum energy is consumed. So in our proposed algorithm it is negligible. Therefore It will be considered only dynamic energy consumption. consumes less energy than dynamic energy so it is negligible.

$$D_{eng} = c * (vol_{vm_j})^2 * frq_{vm_j} * ET(t_i, vm_j) \tag{11}$$

Where c is the capacitances load based on dynamic power depending on the device. (vol_{vm_j}) is the supply voltage of the j^th VMs and frq_{vm_j} is the processing speed of the

j^th VMs and $ET(t_i, vm_j)$ is the execution time of t_i on vm_j. When t_i is schedule on vm_j at voltage and frequency pair.

1.8. Average VM Utilization (VMU)

It is the ratio of the active time of the VMs [i.e $AT(VM_j)$] to the MKS. Mathematically represented as follows

$$Utlization(VM_j) = \frac{AT(VM_j)}{MKS}, Where\ j\ =\ 1\ to\ y \tag{12}$$

Average VM utilization is the utilization of all the VMs in the cloud data center. Mathematically represented as follows

$$Avg\ U(VM_j) = \frac{1}{y}\sum_{i=1}^{y} Utlization(VM_j) \tag{13}$$

Remarks 3.4 : Average Utilization of all the VMs always lies between 0 to 1 (i.e $AvgU(VM_j) \in [0 \sim 1]$).

1.9. Speedup (SP)

It is the ratio of maximum sequential execution of VMs by parallal exeuction schedule length. It can be mathematically represented as follows

$$Speedup = \frac{\sum_{i=0}^{x} EET(T_i, V_j)}{MKS} \tag{14}$$

1.10. Schedule Length Ratio(SLR)

To measure the performances of any scheduling algorithm on the DAG is schedule length (MKS) of its output schedule. Because a different size of graphs are used, it is required to normalize the MKS to lower bound, which is called SLR. SLR mathematically represented as follows

$$SLR = \frac{MKS}{\sum_{n_i \in CP_{MIN}} min_{p_j \in Q} W_{(i,j)}} \tag{15}$$

The denominator is the summation of the minimum computation costs of tasks on the CP_{MIN}. (For an unscheduled DAG, if the computation cost of each node n_i is set with the minimum value, then the critical path will be based on minimum value, then the critical path will be based on the minimum computation costs, which is represented as CP_{MIN}. The SLR of the graph (using any algorithm cannot be less than one since the denominator is the lower bound.