# **Cloud Workflow Scheduling using Particle Swarm Optimization**

## **1. Introduction**

Workflow scheduling is one of the most prominent problems in cloud computing. A workflow consists of a number of tasks that have data dependency among them or the output of one task may act as the input for another task. These tasks must be executed in the precedence order taking into account various QoS parameters such as deadline, cost, energy consumption, etc. Workflows are common in various fields of engineering and science like astronomy, physics, bioinformatics, etc.

These complex tasks must be deployed over parallel and distributed computers to get timely and accurate results where the resources should scale up or scale out as per the client’s needs and SLA (service level agreement) along with pay-as-you-go model. Cloud computing offers such elastic resources that scale on demand giving the cloud user a perception that the resources are infinite. Due to this nature of cloud computing that it elasticity and pay-as-you-go, it attracts large number of enterprises, universities and government organizations. This helps them perform their computing tasks at low cost and without the need of purchasing and maintaining their own infrastructure.

Cloud computing offers these resources in the form of heterogeneous virtual machines. Virtual machines are the encapsulation of various computing resources such as computing power, memory, storage, etc. All VMs offer different combination of such resources and have a attached price with them. Typically, users are charged according to hour or minute. All VM types may have multiple instances running at a given point of time and these instances can scale out (adding more resources in parallel) or scale up (increasing the capacity of the single resource).

Various service models offered by cloud computing are as follows.

1. IaaS or Infrastructure as a Service: In this service model computing hardware or infrastructure such as computing power, storage, memory, network, etc. is offered to the cloud user in virtualized format on a pay-as-you-go basis. Example: Amazon EC2, Microsoft Azure.
2. PaaS or Platform as a Service: In this service model along with infrastructure the cloud user is also offered a middleware such as database or complete environment in which the user can develop, test, modify, debug and deploy his application. Example: Heroku, Google Firebase, GitHub, etc.
3. SaaS or Software as a Service: In this service model, an application software is offered to the end user and delivered through a conventional browser and runs over the network. Delivery over the browser is not a necessity. An application installed on a system that exchanges data from the cloud can also be termed as SaaS. Example: Netflix, Spotify, etc.

This project focuses on deadline constrained workflow scheduling which lies under the domain of IaaS. The combination of parameters chosen for optimization are makespan, energy consumption, and load balancing. At its core workflow scheduling is an NP-Hard problem hence finding the exact solution is a challenging and computationally expensive task. Hence a soft computing technique called Particle Swarm Optimization (PSO) along with idle time slot aware rules is used to arrive at an approximate solution.

## 2. Cloud Workflow Scheduling Problem and Formulation

### 2.1 Workflow Model

Given a set of precedence constrained tasks modeled by a weighted DAG (Directed Acyclic Graph).

Where,

### 2.2 Cloud Model

Cloud consists of heterogeneous virtual machines of various types with varying CPU capacity, storage and memory.

At any given instant of time the cloud consists of theoretically infinite instances of these VM types running.

### 2.3 Terminology

|  |  |  |
| --- | --- | --- |
| **S No** | **Notation** | **Definition** |
|  |  | DAG modelling a set of precedence constrained tasks |
|  |  | Set of precedence constrained tasks |
|  |  | task in |
|  |  | Amount of data in bits to be transferred from task to |
|  |  | Set of virtual machines |
|  |  | Set of all running virtual machine instances running in the cloud |
|  |  | Refers to a virtual machine instance of type with index |
|  |  | All predecessor tasks wrt to |
|  |  | All successor tasks wrt to |
|  |  | Entry level task or |
|  |  | Exit level task or |
|  |  | CPU capacity of virtual machine type of instance in instructions per second |
|  |  | Power consumed by if task is executed on its instance |
|  |  | Number of instructions in task |
|  |  | Data transfer rate (in bits/s) among virtual machines (assumed constant for all) |
|  |  | Virtual machine instance which task is mapped to |
|  |  | Data transmission time from to |
|  |  | Earliest starting time of task on |
|  |  | Latesh finish time of task on |
|  |  | Execution time of task on |
|  |  | Earliest time at is available to execute any task being mapped to it, determined by idle-slot-aware-rules. |
|  |  | Deadline for the entire set of tasks |
|  |  | Total energy consumed by task when executed on |
|  |  | Set of all virtual machine instances actually used to execute tasks or |
|  |  | Lease time of virtual machine instance of type |
|  |  | Mean lease time of all virtual machines or |

### 2.4 Preliminaries

1. here

Besides these two other quantities are defined for making approximations in the algorithm.

Earliest start time or EST on a given virtual machine instance

test finish time or LFT on a given virtual machine instance

ere is the minimum execution time of based on the fastest virtual machine.

### 2.5 Problem Formulation

We aim to find a schedule S such that

Subject to constraint that a task can be mapped to a single virtual machine only

1. Minimize:

Subject to

Corresponds to minimizing makespan

1. Minimize:

Corresponds to minimizing energy consumption

1. Minimize:

Corresponds to load balancing

Combining the multiple objectives into one.

+

## 3. Proposed Method

Particle swarm optimization is a searching technique used in soft computing inspired from flocking of birds that iteratively refines the candidate solution based on the measurement of quality. It has very few parameters hence it is easy to implement and no sophisticated encoding, decoding schemes are required in its implementation.

PSO consists of a fixed number of particles called the population whose size is denoted by N. Each particle denotes a possible solution of the given problem whose quality is judged by the fitness value of the particle. If the search space is dimensional then at a given time the particle has a certain position in the search space denoted by . Also, the particle has a certain speed denoted by . The best global position seen till now is denoted by . Also the best position seen by the particle is given as . After every iteration the values are updated according to the following equations.

+

where,

is the inertial weight that balances the global and local search

and are the acceleration factors or trust parameters that control the social and cognitive influence. and are uniform random numbers in [0, 1]

### 3.1 Issues associated with PSO

1. Encoding scheme: It refers to the way the particle is a representation of a candidate solution of the given problem.
2. Decoding Scheme: It refers to deriving the actual schedule from the particle.
3. Evaluation: A fitness value or quality of solution must be associated with every solution/particle.
4. Population Initialization: It refers to carefully choosing the placement of initial population in the search space.
5. Particle repair: At every iteration the values assumed by one or more dimensions of the particle may be inconsistent with the problem hence each particle must be repaired to get sensible results at the end.

### 3.2 Pseudocode of PSO

|  |
| --- |
|  |
|  |

### 3.3 Encoding Scheme

The virtual machine instances offered in the cloud by the cloud provider are theoretically infinite however the VM types offered are always finite at a moment. Hence the particle must only handle the mapping of the task to virtual machine types rather than instances.

PSO is generally used for continuous optimization problems whereas the problem of cloud workflow scheduling is a combinatorial one and therefore each real value assumed along every dimension must be rounded of to the nearest integer.

Along with task to virtual machine type mapping the data dependency constraints must also be taken into account, hence every task in addition to mapping is also assigned a value that determines its priority (lower number means higher priority). While decoding the particle the tasks are scheduled in the order of priority.

If there are tasks then the solution space will be dimensional and also every particle will assume dimensions. The first dimensions (dimensions to ) determine the task to VM type mapping whereas the next dimensions (dimensions to ) determine priorities.

Example: Let there be 3 ( tasks 1 through 3) and 5 virtual machine types (0 through 4) then a typical particle encoding will be as follows

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Dimension | 1 | 2 | 3 | 4 | 5 | 6 |
| Value | 0.1 | 2.6 | 1.9 | 2.4 | 1.7 | 9.8 |

The task to VM type mappings is as follows :-

And the tasks arranged descending order of priority will be

### 3.4 Repairing Method for Invalid Task’s Priorities

During the courses of iteration, the particles might assume invalid priorities. To obtain a valid solution the priorities of tasks must be repaired. All children must have priority lower than its parent. Firstly the task with minimum priority value is found that is and is added to and removed from T (line 4) then all tasks are searched in whose parents belong to and stored in Q. is the maximum priority of any element in , each element in is assigned the priority where is the maximum priority of all of its parents (lines 5 c i to 5 c iii). At the end if still there is an element in Q whose priority equals to any element in V then it is given the priority (line 5 c v), then finally this task is added to and removed from

|  |
| --- |
|  |
| * + 1. If |

### 3.5 Decoding

The decoding of particle or deriving a schedule from the particle is based on idle time slot aware procedure. Both the and are initialized to in the beginning. S will store the schedule. S will consist of tuples in the format . is the set of all leased VM instances which is also initialized to . A scheduling sequence is obtained using the priorities. Then from lines 4a to 4i an appropriate slot for this task is obtained from the already running instances. If no slot is found such that it meets all the constraints then a new instance is launched. In line 4a the VM type of the task is obtained from the particle. Then all the serial instances (Instances that have at least one parent of is scheduled) and parallel instances (which has no instance of scheduled). Note that the instance should be of same VM type that is mapped to. Then all applicable instances of are searched in X and stored in If no instance is found then an applicable instance is searched in and stored in . If is non-empty then a instance that has least load is assigned to (load balancing) otherwise a new instance of the required type is launched. Once the instance is assigned the starting and finishing time is calculated and the tuple is added to and the schedule is returned. Note that applicable instance is one which has an empty slot for a given task and assigning this task to that slot will not compromise the deadline constraints in any way.

|  |
| --- |
|  |
|  |

### 3.5 Population Initialization

Upward rank and downward ranks are used to assign priorities to the tasks. The upward rank of a task is the critical path length from to the exit task, including average execution time. Thedown rank of is the longest distance from the entry task to it,excluding the average execution time of . Two randomly generated particle sets are generated and . Inall tasks are assigned priority as their downward ranks and in the priorities are assigned as where is the max of all upward ranks (because lowest number denotes highest priority). Two additional particles are added to and One has all tasks mapped to fastest VM types and another has each task mapped to the VM on which it consumes least power while executing. This is done so that the initial population consists of diverse particles and the particles don’t get stuck in local minima.