WorkFlow in Cloud Computing

Sarateanu Daniel-Claudiu
Faculty of Automatic Control and Computers
University POLITEHNICA of Bucharest
Splaiul Independenței 313, Bucharest, Romania, 060042
{*daniel.sarateanu*}*@stud.acs.upb.ro*

January 18, 2017

Abstract

Workflows are groups of tasks where there is a well-established order between steps. Thus, a step of a workflow depends on the completion of at least one other previous step. This theme project involves implementing a system of workflows in CloudSim.

1 Introduction

1.1 Cloud Computing: This technology is one of the most popular subject in computer science in this era. Cloud computing offers access to boundless virtual resources dynamically provisioned on demand for various applications like scientific platforms , big data processing applications etc. The most important benefits of cloud computing are:

- **On-demand self-service** - a consumer cand unilaterally provision computing capabilities,
- **Broad nerwork access** - availability over network
- **Resource pooling** - The provider's computing resources are pooled to serve multiple consumersusing a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand
- **Rapid elasticity** - Capabilities can be elastically provisioned and released
- **Measured service** - can be monitored, controlled and reported, providing transparency for both the proveder and consumer of the utilized service

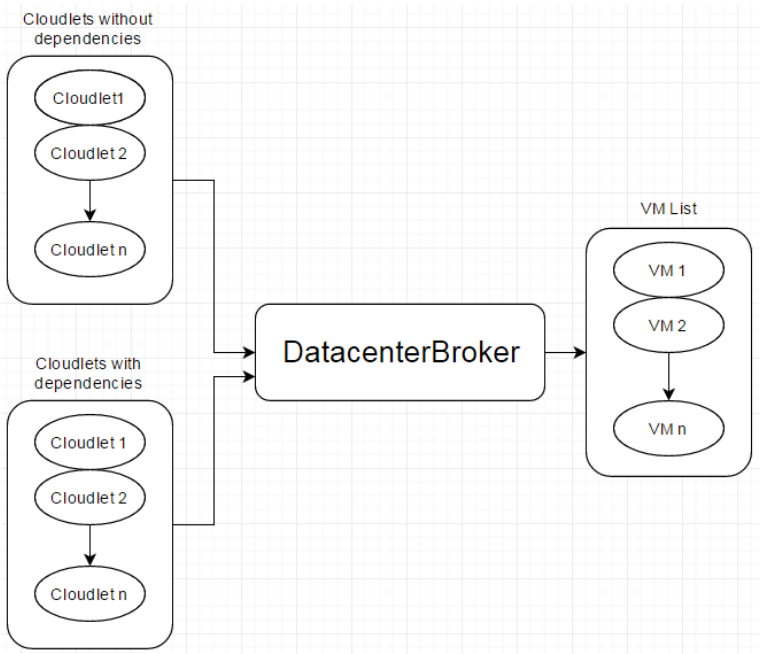
1.2 Workflow Management: Nowadays, workflow scheduling is a big issue for programers. The purpose of workflow scheduling is to automate the procedures especially which are involved in the process of passing the data and the files between the participants of the cloud, maintaining the constraints using scheduling algorithms.

1.3 Existing Scheduling Algorithm:

Sr. No.	Scheduling Algorithm	Scheduling Parameters	Scheduling factors	Findings	Tools
1	Market Oriented Hierarchical Scheduling Algorithm[4]	Makespan, CPU time	DAG scheduling	Minimizing the execution time.	Amazon EC2
2	SHEFT workflow scheduling Algorithm[5]	Scalability and Execution time	Group of tasks	Used to optimize execution time of workflow as well as helps to scale up.	Cloudsim
3	HEFT workflow scheduling Algorithm[7]	Makespan	Highest upward rank	Reduce make span in a DAG	GridSim
4	Ant colony Optimization Based Service flow Scheduling[16]	Resource Utilization, time	QOS	Optimizes the service flow scheduling	Amazon EC2
5	A Particle Swarm Optimization-based Heuristic for Scheduling.[6]	Resource utilization time.	Group of tasks	Good distribution of workload in a cost saving manner.	Amazon EC2
6	Multiple QOS constrained scheduling Algorithm[8]	Success rate, cost, time, Makespan	Multiple Workflows	Scheduling the workflow dynamically.	CloudSim
7	Cost based scheduling on utility grids.[9]	Cost	Task Scheduling	Reschedule the unexpected tasks	GridSim
8	Optimal Workflow scheduling Algorithm[11]	CPU utilization, Execution Time	Multiple Workflows	Improves CPU utilization	Open Nebula
9	RASA Workflow scheduling algorithm[10]	Makespan	Grouped tasks	Reduces makespan	GridSim
10	Workflow with budget constraints[12]	Makespan, budget	DAG scheduling	Minimize the execution time and the makespan	Amazon EC2
11	Improved cost-based scheduling algorithm for task scheduling in cloud computing.[14]	Cost and performance	Computation/Communication Ratio	Make efficient mappings of the tasks	CloudSim
12	An approach to Optimized Resource Scheduling Algorithm[15]	Speed, resource Utilization	IGA(Improved Genetic Algorithm) and automation	Achieves Optimization for cloud scheduling	Eucalyptus

2 Architecture

The architectural scheme of implementation:



This scheme architectural highlights that the scheduling algorithm is based on management of cloudlets. In next chapter I will explain in details how this implementation works.

3 Implementation

This implementation requires the extension of Cloudlet to create a new type of object that contains dependencies from other Cloudlet sites, but also changing the way Cloudlet sites are allocated by virtual machines in the datacenter (by altering the method submitCloudlets () class DatacenterBroker). In this implementation Cloudlets that depends on another Cloudlet/Cloudlets can not be allocated to a virtual machine and run until Cloudlet site on which has not finished running.

Workflow simulation step by step:

1. The important entities are created(Broker, Datacenter, CIS)
2. VMs are created on Datacenter.
3. Cloudlets are created (with/without dependencies)
4. Send Cloudlets to broker
5. The Broker send Cloudlets to VMs for execution(submitCloudlet):
   (a) Send Cloudlets without dependencies
   (b) Remove cloudlets submited from initial list
   (c) Execute cloudlets from VMs

(d) Verify if the list of cloudlets have items. If size of list is 0 clean DataCenter and shutdown all entities, else go to next step

(e) Go back to list of Cloudlets

(f) Verification of each cloudlet if cloudlets from dependencies list are executed

(g) If verification return true cloudlet is submited to VM

(h) go to step c

4 Scenarios and Results

I did a lot of test, but the most relevant test was when I used 10 cloudlets(5 with dependencies) and 3 VMs. In the logs below can be seen more clearly what happened.

```
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: VM #1 has been created in Datacenter #2, Host #0
0.1: Broker: VM #2 has been created in Datacenter #2, Host #0
0.1: Broker: Sending cloudlet 0 to VM #0
0.1: Broker: Sending cloudlet 0 to VM #1
0.1: Broker: Sending cloudlet 0 to VM #2
0.1: Broker: Sending cloudlet 0 to VM #0
0.1: Broker: Sending cloudlet 0 to VM #1
1600.1: Broker: Cloudlet 0 received
3200.1: Broker: Cloudlet 0 received
3200.1: Broker: Cloudlet 0 received
3200.1: Broker: Cloudlet 0 received
3200.1: Broker: Cloudlet 0 received
3200.1: Broker: Sending cloudlet 0 to VM #0
3200.1: Broker: Sending cloudlet 0 to VM #1
4800.1: Broker: Cloudlet 0 received
4800.1: Broker: Cloudlet 0 received
4800.1: Broker: Sending cloudlet 0 to VM #0
6400.1: Broker: Cloudlet 0 received
6400.1: Broker: Sending cloudlet 0 to VM #0
6400.1: Broker: Sending cloudlet 0 to VM #1
8000.1: Broker: Cloudlet 0 received
8000.1: Broker: Cloudlet 0 received
8000.1: Broker: All Cloudlets executed. Finishing...
8000.1: Broker: Destroying VM #0
8000.1: Broker: Destroying VM #1
8000.1: Broker: Destroying VM #2
```

Execution on VMs was divided on four steps. As I said in Implementation chapter, it was executed cloudlets without dependencies. After that, the algorithm verify if can execute other cloudlets. At second step was available for execution two cloudlets, at Third step was execute one cloudlet and at last step was executed last two cloudlets.