RESOURCE SCHEDULING USING RESOURCE POOL IN OPENSTACK

Shalmali S. Sahasrabudhe, Shilpa S. Sonawani
Department of Computer Engineering, MEER’S MIT, Pune
Pune, India

ABSTRACT

OpenStack is a cloud computing platform. OpenStack provides an Infrastructure as a Service (IaaS) and constitutes of resources such as compute, storage and network resources. Resource allocation in cloud environment deals with assigning available resources in cost effective manner. Resource allocation in OpenStack is carried out by nova-scheduler. Logically scheduler allocates compute, network and storage resources to instance requests made by users of OpenStack. For efficient and cost-effective use of scheduler resource pool can be created and maintained which contains best possible hosts available. This paper presents a time saving way for allocating resources for large deployments of cloud in the form of virtual machines.

Keywords: Cloud Computing; Resource Sheduling; Compute; Resource Pool.

I. INTRODUCTION

Cloud computing relies on sharing of resources. OpenStack is a cloud computing platform. OpenStack provides an Infrastructure as a Service (IaaS) and constitutes of resources such as compute, storage and network resources. In cloud computing, Resource Allocation (RA) is the process of assigning available resources that is, compute, storage and network to the virtual machine request. Nova-scheduler is the component which is responsible for resource allocation in OpenStack. Current Filter-weight algorithm used by OpenStack nova-scheduler can be modified for satisfying cloud provider strategies. This paper presents modified algorithm of OpenStack. The proposed algorithm will first filters out the hosts which do not satisfy providers requirements. Second step will be weighing, which is done based on architecture that considers CPU, memory, disk and network bandwidth also.
Hosts which pass this weighing step will be put in the pool for allocation. This resource pool will be updated on event basis. Whenever there will be any creation or deletion of virtual machine instances filtering and weighing step will be carried out and the pool will be updated. So for the next instance request after hosts which are best available are used and there will be no need to perform the first two steps again while allocating the resources. This will save instance scheduling time.

II. OPENSTACK NOVA-SCHEDULER

A. OpenStack Components

OpenStack is an open source system. It is fully distributed system. OpenStack keeps its services as decoupled as possible. This is designed to provide massive scalability.

- Dashboard aka Horizon provides easy access to compute, storage and network resources required by virtual machines. Users can provision and relinquish resources from horizon.
- Compute aka Nova avails computing services for creating deleting and managing virtual machines.
- Network aka Neutron provides virtual networking services to Nova. It allows users to create tenant specific networks.
- Image aka Glance can store the actual virtual disk files in the Object Store aka swift
- Identity management service aka Keystone authenticates and authorizes all components of the OpenStack.

B. Nova-Scheduler

The nova-schedule process is the simplest component in OpenStack Nova. It takes a virtual machine instance request from the queue and determines where it should run (specifically, which compute server host it should run on).[7] Currently, there are several algorithms used by nova-scheduler to choose from (simple, chance, Filter Scheduler etc).

- FilterScheduler is responsible for selecting hosts and provisioning resources. It chooses the host by applying filters and calculates weighted cost. Host which passes filters and has least cost wins.[8]
- ChanceScheduler chooses the host randomly from running hosts.[8]
- SimpleScheduler chooses the host based on the running cores. Host with least running cores wins out.[8]

III. PROPOSED SYSTEM ARCHITECTURE

C. Components of the System

Components of the proposed system are shown in the figure.1. Proposed system consists of following important components.

1) Resource Discovery and Monitoring

Resource discovery is needed by the provider to find appropriate resources to satisfy the VM requests. The resource discovery service uses a discovery framework with an advertisement process. In cloud computing resources can be acquired and relinquished on demand and for this reason resource monitoring should be continuous. For resource monitoring, Pull or Push method can be used. In Pull method, resource monitor will send a query message periodically. In Push method hosts will send the resource state periodically to the monitor.
2) Controller
Typically the cloud controller manages authentication and sends messaging to all the systems through a message queue. Controller will run global scheduler. Global scheduler calculates total available resources in the cloud with the help of global resource monitor.

3) Compute
Compute is a host on which virtual machine or instance in terms of OpenStack is to be launched. Compute host will run local scheduler. Local scheduler creates a disk image of VM and calls the hypervisor to boot the VM.

4) Hypervisor
Compute nodes are configured for hypervisor. OpenStack Compute supports many hypervisors to various degrees, including KVM, LXC, QEMU, UML, VMWare ESX/ESXi, Xen, PowerVM, Hyper-V.

5) Instance Request
Instance request is in the form of requirements from VM resources. It can be represented by the CPU cores, memory, disk, image and network.

![Architecture diagram of VM placement system](image)

**Fig. 1:** Architecture diagram of VM placement system

**IV. PROPOSED ALGORITHM**

The different steps performed by the proposed algorithm are explained below.

The Initial Placement Controller running in the OpenStack scheduler, performs the initial placement of a VM. The functionality of the Local Scheduler is performed by the OpenStack Compute component in each compute server. The Initial Placement Controller selects the compute server on which a VM is to be launched. Figure.2. shows the working of proposed filter-weight scheduler.

In proposed algorithm, OpenStack filter-weight scheduler is used for this purpose. This scheduler contains two generic functions, a filter function that selects the set of compute servers capable of running a given VM and a cost function that ranks the filtered set of servers according to their suitability. These functions need to be instantiated to support a specific QoS.
D. Filtering

Proposed algorithm can filter compute hosts by applying multiple filters at a time. Multiple filters will be filtering CPU cores, RAM, Disk storage, and network bandwidth are described below:

1) CPU cores filter: Used to filter number of cores required by the VM request against number of cores available. It passes hosts with sufficient number of cores.

2) RAM filter: Used to filter hosts by their RAM. Only hosts with sufficient RAM are passed by this filter.

3) Disk filter: Used to filter hosts by their Disk capacity. Only hosts with sufficient Disk space are passed by this filter.

4) Network bandwidth filter: Used to filter hosts based on network bandwidth.

E. Weighing

Weighing will be done again considering all four parameters that is, CPU, memory, disk storage and network bandwidth. This step will first score each host by running each cost function. And then it will calculate the weighted scores for each host by multiplying score and weight of each cost function to generate a list of weighted final scores. Lastly, it will sort the hosts based on the weighted scores. This way the least cost host will win.

Weighted scores for each host is calculated by multiplying score and weight of each cost function to generate a list of weighted final scores. The formula used is:

\[
\text{Final score of a certain host} = \sum (\text{weight of cost function} \times \text{score returned by this function for the host})
\]

And then it will associate the scores with Host States to generate a list of tuples of score and Host State. Lastly, it will sort the tuples and return a Weighted Host using the first tuple. This way the least cost host will win.

Fig.2: Filter-Scheduler for proposed algorithm
F. Pool Creation and Maintainance

Pool of resources, that is hosts will be maintained which are least utilized. Hosts which are having least costs are selected after weighing step. For the very first instance request host is selected with the least cost after weighing step. Then onwards pool will be used for host selection.

Pool is updated on event basis. Events like creating, updating and deleting instance are considered for updating the pool. Updating pool process will again perform first two steps of filtering and weighing.

But the main difference between existing filter-scheduler algorithm and proposed algorithm is filtering and weighing steps are performed offline in proposed algorithm. In the existing algorithm filtering and weighing is performed on the fly that is, whenever there is instance request. This takes lot of time for host selection. Time is saved if the host selection is done using previously selected hosts from the pool. Following figure 3. shows the flow of the algorithm.

Fig.3: Flow diagram for the proposed algorithm

ACKNOWLEDGMENT

This work has been supported by Avaya India Pvt. Ltd. We are thankful for the encouragement and infrastructural support from Avaya India Pvt. Ltd. Also we would like to thank Department of Computer Engineering, MAEER’S MIT, Pune, for their extended help and suggestions.
REFERENCES


