Review on Scheduling based Resource Allocation in Cloud computing

Pragnesh K. Patel[1] **Prof. H. B. Jethva**[2]
[1]M.E. Student, Gujarat Technological University
[2]Associate Professor, Computer Engineering Department
[2]L. D. College of Engineering, Ahmedabad

Abstract-Cloud Computing is the next generation Architecture of the most IT enterprise. It made possible, the application to run without the burden of local hardware and software. Cloud computing is basically step toward the idea of "computing as a utility" came from grid computing which is also known as on-demand service. The main goal of cloud computing is to provide on-demand metered service. The metered service means "pay-as-you-go". So, resource allocation is key issue in cloud computing. For better resource allocation cloud service providers use lease managers. Lease managers depend on their default scheduling algorithms and their efficiency of granting and revoking leases is defendant on efficiency of scheduler they have.

Key words: cloud computing, lease management, resource allocation, scheduling, Open Nebula, Haizea, Centrifuge

I. INTRODUCTION

Clouds can be used to provide on-demand capacity as a utility[1].Since a cloud typically comprises a large amount of virtual and physical servers, in the order of hundreds or thousands, efficiently managing this virtual infrastructure becomes a major concern[5].

Several solutions, such as VMware Virtual Center, Platform Orchestrator, or Enomalism, have emerged to manage virtual infrastructures, providing a centralized control platform for the automatic deployment and monitoring of virtual machines (VMs) in resource pools [6].

However, these solutions provide simple VM placement and load balancing policies. In particular, existing clouds use an immediate provisioning model, where virtualized resources are allocated at the time they are requested, without the possibility of requesting resources at a specific future time and, at most, being placed in a simple first-come-first-serve queue when no resources are available[6].

However, service provisioning clouds have requirements that cannot be supported within this model, such as [6]:

– Resource requests that are subject to non-trivial policies
– Capacity reservations at specific times to meet peak capacity requirements
– Variable resource usage throughout a VM's lifetime
– Dynamic renegotiation of resources allocated to VMs.

Additionally, smaller clouds with limited resources, where not all requests may be satisfied immediately for lack of resources, could benefit from more complex VM placement strategies supporting queues, priorities, and advance reservations. So, we need capacity provisioning using *resource leases.*

A lease is "a negotiated and renegotiable agreement between a resource provider and a resource consumer, where the former agrees to make a set of resources available to the latter, based on a set of lease terms presented by the resource consumer" [6]. For this kind of capacity provisioning we need lease manager.

Popular lease mangers are:

i) Haizea
ii) Open Nebula
iii) Centrifuge

II. OPEN NEBULA

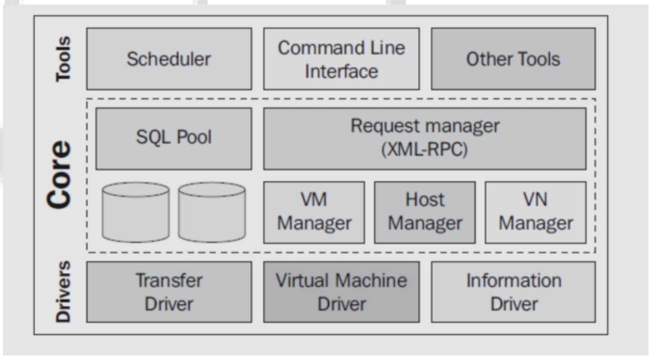

Fig. 1. Open Nebula virtual infrastructure engine Components and its integration with an external cloud provider. [5]

The open source Open Nebula virtual infrastructure engine provides the functionality needed to deploy, monitor and control VMs on a pool of distributed physical resources. The Open Nebula architecture has been designed to be flexible and modular to allow its integration with different hypervisors and infrastructure configurations [6].

Open nebula is composed of three main components [4]:

– **The open nebula Core** -Manages lifecycle of VM & provides management interface,
– **The Capacity Manager** - Governs functionality provided by Core and
– **Drivers** - Drivers for hypervisors supported by open nebula [2].

Thus, open nebula is not tied to any specific environment, providing a uniform management layer regardless of the virtualization technology used.

In this way, Open Nebula shapes a physical infrastructure to support the execution of a given service workload. Moreover, Open Nebula is able to dynamically scale-out this infrastructure by interfacing with an external cloud; an Amazon EC2 virtualizer driver is currently included with Open Nebula, and drivers could be developed to interface with other clouds (such as the Science Clouds, through the Globus Workspace Service interface). This seamless integration of an external cloud with in-house resources allows for effective access of outsourced computational capacity [2].

Open Nebula can be used in two ways for lease management:

i) Open Nebula with Haizea
ii) Open Nebula standalone

Despite the fact that the Open Nebula command-line interface is simple and straightforward to use, most users, especially non-technical ones, may prefer to use a more comfortable graphical user interface.

Open Nebula includes, from recent releases, a standalone web interface called Sunstone, which permits cloud administrators to easily manage resources, perform typical operations on them, and keep an eye on what is going on[5].Here default scheduling algorithm is mm_sched.

III. HAZIEA

Haizea is an open source lease management architecture developed by Borja Sotomayor that Open Nebula can use as a scheduling backend. Haizea uses leases as a fundamental resource provisioning abstraction, and implements those leases as virtual machines, taking into account the overhead of using virtual machines when scheduling leases. Using Open Nebula with Haizea allows resource providers to lease their resources, using potentially complex lease terms, instead of only allowing users to request VMs that must start immediately[4].
In Haizea, the lease terms include the hardware resources, software environments, and the availability period during which the hardware and software resources must be available. Haizea maps leases to VMs, scheduling the deployment overhead of starting those VMs separately and leveraging backfilling algorithms and the suspend/resume/migrate capability of VMs to schedule the leases more efficiently[3].

Haizea can be used in three modes: Open Nebula mode, unattended simulation mode, and interactive simulation mode.

In unattended simulation mode, Haizea takes a list of lease requests (specified in a trace file) and a configuration file specifying simulation and scheduling options (such as the characteristics of the hardware to simulate), and processes them in “simulated time".

In interactive simulation mode, enactment actions are simulated, but Haizea runs in “real time". This means that, instead of having to provide a list of lease requests beforehand, you can use Haizea's command-line interface to request leases interactively and query the status of Haizea's schedule.

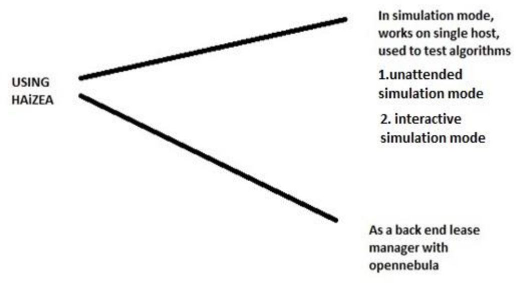
USING HAIZEA
In simulation mode, works on single host, used to test algorithms
1.unattended simulation mode
2. interactive simulation mode
As a back end lease manager with opennebula

Fig. 2. Ways of using Haizea

IV. CENTRIFUGE

Centrifuge is a datacenter lease manager. The Centrifuge system has been deployed as part of Microsoft live mesh service. Live mesh is a large scale commercial cloud service used for File sharing; Notifications of activity on shared files, Virtual desktop, etc. There are three parts to the Centrifuge: **Lookup library**-copy of complete lease table, **Owner library -** Maintains record of ranges given to each owner and **Manager -** Maintains namespaces [5]. Centrifuge is internal part of live mesh. So, it is not possible to use centrifuge with open source technologies.

V. POINT BY POINT COMPARISON OF MM_SCHED AND HAIZEA’S ALGORITHMS

Point	Mm_Sched	Haizea’s Algorithms
Immediate Scheduling	Yes	Yes
Best-Effort Scheduling	Yes	Yes
Advance Reservations	No	Yes
Rank Scheduling Policy	Yes	No
Pluggable Scheduling Policies	No	Yes
Simulator	No	Yes
Easy To Extend Algorithm	No	Yes

Table.1. Comparison of Algorithms

From above comparison, we can say that both algorithms have limitations. But mm_sched has more limitations over haizea. So, it is recommended to use haizea as a back end lease manager with Open Nebula.

VI. IMPLEMENTING OWN SCHEDULING POLICY

We can also customize scheduling policy by writing own policies in policy.py module. Default policy.py is shown below. [3]

```
from haizea.common.utils import abstract
from mx.DateTime import DateTimeDelta
import operator
class PolicyManager(object):
   """The Policy Manager"""
   def __init__(self, admission, preemption,
host_selection):
       """Constructor"""
       self.admission = admission
       self.preemption = preemption
       self.host_selection = host_selection
```

```
  def sort_leases(self, preemptor, preemptees, time):
    """Sorts a list of leases by their preemptability"""
    leases_score = [(preemptee,
self.get_lease_preemptability_score(preemptor,preemptee,
time)) for preemptee in preemptees]
    leases_score = [(preemptee,score) for
preemptee,score in leases_score if score != -1]
    leases_score.sort(key=operator.itemgetter(1),
reverse=True)
    return [preemptee for preemptee,score in
leases_score]

  def sort_hosts(self, nodes, time, lease):
    """Sorts a list of hosts by their score"""
    nodes_score = [(node, self.get_host_score(node, time,
lease)) for node in nodes]
    nodes_score.sort(key=operator.itemgetter(1),
reverse=True)
    return [node for node,score in nodes_score]

  def accept_lease(self, lease):
    """Lease admission function"""
    return self.admission.accept_lease(lease)

  def get_lease_preemptability_score(self, preemptor,
preemptee, time):
    """Computes the lease preemptability score"""
    return
self.preemption.get_lease_preemptability_score(preemptor
, preemptee, time)

  def get_host_score(self, node, time, lease):
    """Computes the score of a host"""
    return self.host_selection.get_host_score(node, time,
lease)

class LeaseAdmissionPolicy(object):
  """Lease Admission policy"""
  def __init__(self, slottable):
    """Constructor"""
    self.slottable = slottable

  def accept_lease(self, lease):
    """Lease admission function"""
    abstract()

class PreemptabilityPolicy(object):
  """Lease Preemptability policy"""
  def __init__(self, slottable):
    """Constructor"""
    self.slottable = slottable

  def get_lease_preemptability_score(self, preemptor,
preemptee, time):
    """Computes the lease preemptability score"""
    abstract()

  def _get_aging_factor(self, lease, time):
    """Returns an aging factor for the preemptability
score"""
```

```
    # TODO: Make horizon configurable
    horizon = time - DateTimeDelta(7)
    if lease.submit_time <= horizon:
      return -1
    else:
      seconds = (time - lease.submit_time).seconds
      horizon_seconds = DateTimeDelta(31).seconds
      return float(horizon_seconds - seconds) /
horizon_seconds

class HostSelectionPolicy(object):
  """Host Selection policy"""
  def __init__(self, slottable):
    """Constructor"""
    self.slottable = slottable

  def get_host_score(self, node, time, lease):
    """Computes the score of a host  """
    abstract()
```

The following decisions are factored out using pluggable module [3]:

– *Lease admission:* It takes into account that this decision takes place before Haizea determines if the request is even feasible. However, being accepted doesn't guarantee the lease will get resources.

– *Lease preemptability:* Not all leases are created equal and, if the scheduler determines that a lease request can only be satisfied by preempting other leases, it may have to determine what leases are better candidates for preemption.

– *Host selection:* When the scheduler has a choice of several physical hosts on which to deploy VMs, some might be preferable than others.

VII. CONCLUSION

Lease management is very important issue in cloud deployment. It can be done through Haizea, Open Nebula or Centrifuge very efficiently. But centrifuge is internal part of live mesh. So, it is not possible to use centrifuge with open source technologies. So, Open Nebula with Haizea as back end lease manager will be best choice for our purpose.

REFERENCES

[1] Peter Mell, Timothy Grance "*The NIST Definition of Cloud Computing*" National Institute of Standards and Technology Special Publication 800-145
[2] Neha Tyagi, Rajesh Pathak, *"Negotiation for Resource Allocation for Multiple processes Infrastructure as a Service Cloud"*, (IJAER) 2011, Vol. No. 2, Issue No. I, July
[3]Sotomayor B Haizea: http://haizea.cs.uchicago.edu/whatis.html, Computation Institute, University of Chicago.
[4] Borja Sotomayor, Rubén S. Montero, Ignacio M. Llorente, Ian Foster, *"Virtual Infrastructure Management in Private and Hybrid Clouds,"* IEEE Internet Computing, vol. 13, no. 5, pp. 14-22, Sep./Oct. 2009.
[5] Sempolinski, P., and Thain, D. (2010) *"A Comparison and Critique of Eucalyptus, Open Nebula and Nimbus"*. In IEEE International Conference on Cloud Computing Technology and Science, pp. 417-426.

[6] Sotomayor B. et. al. – *"Resource Leasing and the Art of Suspending Virtual Machines"*, HPCC'09), 2009, pp. 59--68.
[7] Atul Adya, John Dunagany and Alec Wolmany, "Centrifuge: Integrating Lease Management and Partitioning for Cloud Services".