





This makes Nova a Management Platform that manages compute resources, networking, authorization, and scalability needs of the OpenStack cloud. But, Nova does not provide any virtualization capabilities by itself; instead, it uses libvirt [5] APIs to interact with the supported hypervisors. Nova exposes all its capabilities through a web services API that is compatible with the EC2 API of Amazon Web Services.

Functions and Features:

- Instance life cycle management
- Management of compute resources
- Networking and Authorization
- REST-based API
- Asynchronous eventually consistent communication
- Hypervisor agnostic: support for Xen, XenServer/XCP, KVM, UML, VMware vSphere and Hyper-V

Components of OpenStack Compute

Nova Cloud Fabric is composed of the following major components:

1. API Server (nova-api)
2. Compute Workers (nova-compute)
3. Scheduler (nova-scheduler)

1. API Server (nova-api)

The API Server provides an interface to the outside world to interact with the cloud infrastructure. API server is the only component that the outside world uses to manage the infrastructure. The management is done through web services calls using EC2 API. The API Server then, in turn, communicates with the relevant components of the cloud infrastructure through the Message Queue. As an alternative to EC2 API, OpenStack also provides a native API called "OpenStack API".

2. Compute Worker (nova-compute)

Compute workers deal with instance management life cycle. They receive the requests for life cycle management via the Message Queue and carry out operations. There are several Compute Workers in a typical production cloud deployment. An instance is deployed on any of the available compute worker based on the scheduling algorithm used.

3. Scheduler (nova-scheduler)

The scheduler maps the nova-API calls to the appropriate OpenStack components. It runs as a daemon named nova-schedule and picks up a compute server from a pool of available resources depending upon the scheduling algorithm in place. A scheduler can base its decisions on various factors such as load, memory, physical distance of the availability zone, CPU architecture, etc. The nova scheduler implements a pluggable architecture. Currently the nova-scheduler implements a few basic scheduling algorithms:

3.1. Chance: In this method, a compute host is chosen randomly across availability zones. [6]

3.2. Availability zone: Similar to chance, but the compute host is chosen randomly from within a specified availability zone.

3.3. Simple: In this method, hosts whose load is least are chosen to run the instance. The load information may be fetched from a load balancer.

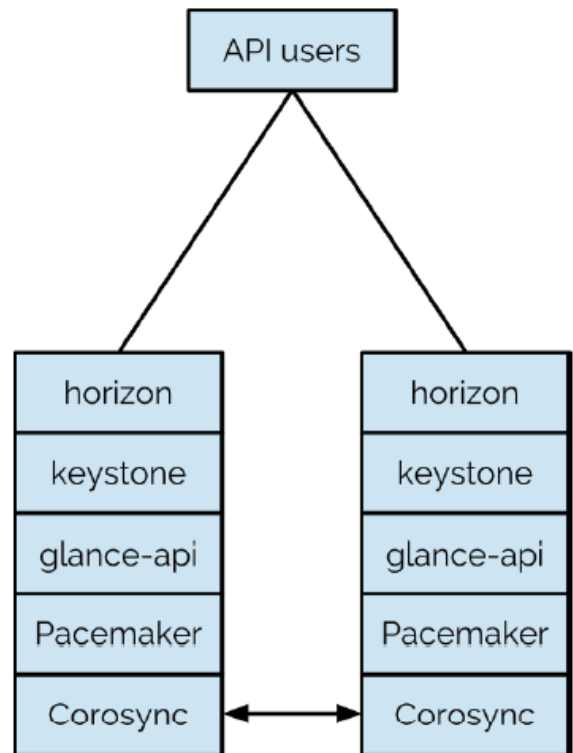


Figure 4. A modified Efficient Conceptualization of Nova API for efficient failover.

### High Availability Solution Components

3.1. Pacemaker

Pacemaker is a cluster resource manager. It achieves maximum availability for your cluster services (aka. resources) by detecting and recovering from node and resource-level failures by making use of the messaging and membership capabilities provided by your preferred cluster infrastructure (either Corosync or Heartbeat).

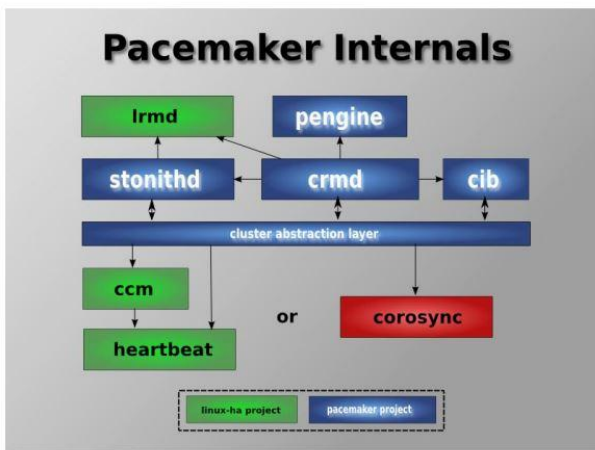
Pacemaker's key features include:

- Detection and recovery of node and service-level failures.
- Storage agnostic, no requirement for shared storage
- Resource agnostic, anything that can be scripted can be clustered.
- Supports STONITH for ensuring data integrity.
- Supports large and small clusters.
- Supports both quorate and resource driven clusters.
- Supports practically any redundancy configuration.
- Automatically replicated configuration that can be updated from any node.
- Ability to specify cluster-wide service ordering, colocation and anti-colocation.
- Support for advanced service types

- Clones: for services which need to be active on multiple nodes
- Multi-state: for services with multiple modes (eg. master/slave, primary/secondary)
- Unified, scriptable, cluster management tools.

### 1.1 Pacemaker - Internal components [7]

Pacemaker itself is composed of four key components (illustrated below in the same color scheme as the previous diagram):



1. CIB (aka. Cluster Information Base)
2. CRMD (aka. Cluster Resource Management daemon)
3. PEngine (aka. PE or Policy Engine)
4. STONITHd

The CIB uses XML to represent both the cluster's configuration and current state of all resources in the cluster. The contents of the CIB are automatically kept in sync across the entire cluster and are used by the PEngine to compute the ideal state of the cluster and how it should be achieved. This list of instructions is then fed to the DC (Designated Coordinator). Pacemaker centralizes all cluster decision making by electing one of the CRMD instances to act as a master. Should the elected CRMD process, or the node it is on, fail a new one is quickly established.[8]The DC carries out the PEngine's instructions in the required order by passing them to either the LRMd (Local Resource Management daemon) or CRMD peers on other nodes via the cluster messaging infrastructure (which in turn passes them on to their LRMd process).

## 4. Design Coding

### • Adding Nova resources to pacemaker

```
Pcs resource create p_nova_api
ocf:openstack:nova-api \
params
config="/etc/nova/nova.conf" \
op monitor interval="5s"
timeout="5s"
pcs resource create p_scheduler
ocf:openstack:nova-scheduler \
params
config="/etc/nova/nova.conf" \
```

```
op monitor interval="30s"
timeout="30s"
pcs resource create p_novnc
ocf:openstack:nova-vnc \
params
config="/etc/nova/nova.conf" \
op monitor interval="30s"
timeout="30s"
pcs resource create p_nova-cert
ocf:openstack:nova-cert \
params
config="/etc/nova/nova.conf" \
op monitor interval="30s"
timeout="30s"
pcs resource create p_novaconsoleauth
ocf:openstack:novaconsoleauth \
params
config="/etc/nova/nova.conf" \
op monitor interval="30s"
timeout="30s"
pcs resource create p_novnconductor
ocf:openstack:novaconductor \
params
config="/etc/nova/nova.conf" \
op monitor interval="30s"
timeout="30s"
pcs resource create p_novanetwork
ocf:openstack:nova-network \
params
config="/etc/nova/nova.conf" \
op monitor interval="30s"
timeout="30s"
```

## 5. Conclusion

The above implemented setup is tested and on simulation with existing Benchmarks sets true and more efficient as this is a pure technical concept and open source and more of scripting and mastering the nova and exploring to the depth of the OpenStack and its compute node rather than concentrating upon the theoretical benchmarks and the Above formulised solution is a far more efficient solution for efficient Failover enabling mechanism and is a great solution compared to other hard coded solutions like VMware fusion and other solutions as the solution proposed by us is a pure Open Source solution.

## 6. Future Work

Improving reliability for OpenStack clouds is critical to facilitating broader enterprise adoption. To meet this challenge, Stratus has developed an industry exclusive set of Linux and KVM extensions to provide Unparalleled availability services for OpenStack workloads. This includes support for fully redundant, fault tolerant instances, as well as other more common high availability approaches from a single, simple set of cloud management tools. In future I will explore more of Nova and Storage (Cinder & Swift) Components and will try to make them more efficient.

## References

- [1] SonaliYadav, "Comparative Study on Open Source Software for Cloud Computing Platform: Eucalyptus,

- Openstack and Opennebula”, International Journal Of Engineering And Science, Vol.3, Issue 10 (October 2013), pp 51-54, ISSN (e): 2278-4721, ISSN (p):2319-6483.
- [2] Johnson B,YanzhenQu,“A Holistic Model for Making Cloud Migration Decision”, 10th IEEE International Symposium on Parallel and Distributed Processing with Applications, Leganes, 10-13 July 2012, pp 435-441, Print ISBN: 978-1-4673-1631-6,DOI: 10.1109/ISPA.2012.63.
- [3] HHarjit Singh, “Technology Transfer Model To Migrate E-Governance To Cloud Computing”, International Journal of Advanced Technology and Engineering Research (IJATER), July 2012, ISSN No: 2250-3536.
- [4] Hsu, F., Malik ,S., Ghorbani S, "Open Flow as a Service". April 21, 2014 Available:<https://wiki.engr.illinois.edu/download/attachments/197298167/CloudFlow-3rd.pdf?version=6>
- [5] Tan, D, "Introduction to OpenStack with Dell Compellent Storage Center"April152014Available:[http://commwebps3.us.de ll.com/techcenter/extras/m/white\\_papers/20439057.aspx](http://commwebps3.us.de ll.com/techcenter/extras/m/white_papers/20439057.aspx)
- [6] ITILV3- Service Asset and Configuration Management <http://www.itil.org/en/vomkennen/itil/servicetransition/servicetransitionprozesse/serviceassetconfigurationmgmt.php>
- [7] A. Avetisyan, R. Campbell, I. Gupta, M. Heath, S. Ko,G. Ganger, M. Kozuch, D. O’Hallaron, M. Kunze, T. Kwan, K. Lai, M. Lyons, D. Milojicic, H. Y. Lee, Y. C. Soh, N. K. Ming, J.-Y. Luke, and H. Namgoong, “Open Cirrus: A GlobalCloud Computing Testbed,”Computer, vol. 43, no. 4, pp. 35–43, 2010.
- [8] R. K. L. Ko, A. Y. S. Tan, and G. P. Y. Ng, “‘Time’ for Cloud? Design and Implementation of a Time-Based Cloud Resource Management System,” inProceedings of the 7th IEEE International Conference on Cloud Computing (CLOUD’14), 2014.
- [9] OpenStack, “Openstack - Open-source software for buildingprivate and public clouds,” <http://www.openstack.org/> [Accessed: 17/12/12014].
- [10] C. K. Leong, “Capacity Planning for Your Virtual Data Center and Cloud –Part 3 March 2013,”<https://infocus.emc.com/choong/kengleong/capacity-planning-for-your-virtual-data-center-and-cloud-part-3/> [Accessed: 18/02/2014]