OpenStack Neat – Dynamic Consolidation of Virtual Machines: Blueprint

Anton Beloglazov

26th of July 2012

Contents

1 Summary

OpenStack Neat is a project intended to provide an extension to OpenStack implementing dynamic consolidation of Virtual Machines (VMs) using live migration. The major objective of dynamic VM consolidation is to improve the utilization of physical resources and reduce energy consumption by re-allocating VMs using live migration according to their real-time resource demand and switching idle hosts to the sleep mode. For example, assume that two VMs are placed on two different hosts, but the combined resource capacity required by the VMs to serve the current load can be provided by just one of the hosts. Then, one of the VMs can be migrated to the host serving the other VM, and the idle host can be switched to the sleep mode to save energy.

Apart from consolidating VMs, the system should be able to react to increases in the resource demand and deconsolidate VMs when necessary to avoid performance degradation. In general, the problem of dynamic VM consolidation can be split into 4 sub-problems:

1. Deciding when a host is considered to be underloaded, so that all the VMs should be migrated out, and the host should be switched to a low-power mode, such as the sleep mode.
2. Deciding when a host is considered to be overloaded, so that some VMs should be migrated from the host to other hosts to avoid performance degradation.
3. Selecting VMs, which should be migrated from an overloaded host out of the full set of the VMs currently served by the host.

4. Placing VMs selected for migration to other active or re-activated hosts.

This work is a part of PhD research conducted within the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne. The problem of dynamic VM consolidation considering Quality of Service (QoS) constraints has been studied from the theoretical perspective and algorithms addressing the sub-problems listed above have been proposed [1], [2]. The algorithms have been evaluated using CloudSim and real-world workload traces collected from more than a thousand PlanetLab VMs hosted on servers located in more than 500 places around the world.

The aim of the OpenStack Neat project is to provide an extensible framework for dynamic consolidation of VMs within OpenStack environments. The framework should provide an infrastructure enabling the interaction of components implementing the 4 decision-making algorithms listed above. The framework should allow configuration-driven switching of implementations of the decision-making algorithms. The implementation of the framework will include the algorithms proposed in our previous works [1], [2].

2 Release Note

The functionality covered by this project will be implemented in the form of services separate from the core OpenStack services. The services of this project will interact with the core OpenStack services using their public APIs. It will be required to create a new Keystone user within the `service` tenant. The project will also require a new MySQL database for storing information about the host configuration, VM placement, and CPU utilization by the VMs. The project will provide a script for automated initialization of the database. The services provided by the project will need to be run on the management as well as compute hosts.

3 Rationale

The problem of data centers is high energy consumption, which has risen by 56% from 2005 to 2010, and in 2010 accounted to be between 1.1% and 1.5% of the global electricity use [3]. Apart from high operating costs, this results in substantial carbon dioxide (CO_2) emissions, which are estimated to be 2% of the global emissions [4]. The problem has been partially addressed by improvements in the physical infrastructure of modern data centers. As reported by the Open Compute Project, Facebook's Oregon data center achieves a Power Usage Effectiveness (PUE) of 1.08, which means that approximately 93 of the data center's energy consumption are consumed by the computing resources. Therefore, now

it is important to focus on the resource management aspect, i.e. ensuring that the computing resources are efficiently utilized to serve applications.

Dynamic consolidation of VMs has been shown to be efficient in improving the utilization of data center resources and reducing energy consumption, as demonstrated by numerous studies [5–16]. In this project, we aim to implement an extensible framework for dynamic VM consolidation specifically targeted at the OpenStack platform.

4 User stories

- As a Cloud Administrator or Systems Integrator, I want to support dynamic VM consolidation to improve the utilization of the data center's resources and reduce the energy consumption.
- As a Cloud Administrator, I want to provide QoS guarantees to the consumers, while applying dynamic VM consolidation.
- As a Cloud Administrator, I want to minimize the price of the service provided to the consumers by reducing the operating costs through the reduced energy consumption.
- As a Cloud Administrator, I want to decrease the carbon dioxide emissions into the environment by reducing the energy consumption by the data center's resources.
- As a Cloud Service Consumer, I want to pay the minimum price for the service provided through the minimized energy consumption of the computing resources.
- As a Cloud Service Consumer, I want to use Green Cloud resources, whose provider strives to reduce the impact on the environment in terms of carbon dioxide emissions.

5 Assumptions

Nova uses a *shared storage* for storing VM instance data, thus supporting *live migration* of VMs.

6 Design

The system is composed of a number of components, some of which are deployed on the compute hosts, and some on the management host.

6.1 Components

6.1.1 Data Collector

- Runs on every Nova Compute host periodically (every X seconds)
- Collects the CPU utilization data for every VM running on the host
- Submits the collected data to the central database
- Stores the collected data locally in a file

The data collector collects the CPU utilization data for each VM and stores it in MHz as integers. These values are portable and can be converted to the CPU utilization for any host or VM type, supporting heterogeneous hosts and VMs. The actual data obtained from Libvirt is the CPU time, which should be converted into MHz using the information about the host's CPU.

Only the CPU utilization is both stored locally and submitted to the database. VM placement algorithms need information about the mapping between VMs and hosts; however, this information can be obtain directly from Nova using its API. The information about host characteristics can also be obtained from Nova. Then the data on the CPU usage by VMs can be converted into the required overall values of CPU usage for hosts.

The database schema contains only one table for now `vm_resource_usage`. The table contains the following fields:

- `id` (string) – the UUID of a VM;
- `timestamp` (datetime) – the time of the data collection;
- `cpu_mhz` (integer) – the collected average CPU usage in MHz from the last measurement to the current time stamp.

The data collector stores the resource usage information locally in files in the `<local_data_directory>/vm`. The data collector stores the data in separate files for each VM. The UUIDs of the VMs are used as the file names for storing data from the respective VMs. The format of files is a new line separated list of integers representing the CPU consumption by the VMs in MHz.

6.1.2 Local Manager

- Runs on every Nova Compute host periodically (every X seconds)
- Reads the data stored by the Data Collector
- Invokes the Underload Detector

  - If the host is underloaded, sends a request to the Global Manager to migrate all the VMs away from the host and switch the host to the sleep mode, then exit
  - If the host is not underloaded, continue with the next steps
- Invokes the Overload Detector
  - If the host is overloaded, invoke the VM Selector
    - Send a request to the Global Manager to migrate the VMs selected by the VM Selector
  - Exit
- Processes acknowledgment requests from the Global Manager about completion of VM migrations and removes/adds records about the migrated VM

6.1.3 Underload Detector

- Deployed on every Nova Compute host
- Invoked by the Local Manager
- Configured with a specific underload detection algorithm
- Passed with the data read by the Local Manager as an argument
- Invokes the specified underload detection algorithm and passes the data passed by the Local Manager as an argument
- Returns the decision of the underload detection algorithm of whether the host is underloaded

6.1.4 Overload Detector

- Deployed on every Nova Compute host
- Invoked by the Local Manager
- Configured with a specific overload detection algorithm
- Passed with the data read by the Local Manager as an argument
- Invokes the specified overload detection algorithm and passes the data passed by the Local Manager as an argument
- Returns the decision of the overload detection algorithm of whether the host is overloaded

6.1.5 VM Selector

- Deployed on every Nova Compute host
- Invoked by the Local Manager if the host is overloaded
- Configured with a specific VM selection algorithm
- Invokes the specified VM selection algorithm and passes the data passed by the Local Manager as an argument
- Returns the set of VM to migrate returned by the invoked VM selection algorithm

6.1.6 Global Manager

- Runs on the management host
- Configured with a VM placement algorithm
- Processes VM migration requests received from Local Managers
- If required, switches hosts on or off
- Invokes the specified VM placement algorithm to determine destination hosts for VM migrations
  - VM placement algorithm can directly query the database to obtain the required information, such as the current VM placement, and resource utilization
- Once destination hosts are determines, call the Nova API to migrate VMs
- Once a migration is completed, send an acknowledgment request to the Local Managers of the source and destination hosts

6.2 Configuration File

The configuration of OpenStack Neat is stored in `/etc/neat/neat.conf` in the standard ini format. The configuration includes the following options:

- `sql_connection` – the host and credentials for connecting to the MySQL database;
- `global_manager_host` – the host name of the global manager;
- `global_magager_port` – the port of the global manager's REST interface;
- `local_data_directory` – the directory, where the data collector stores the data on the resource usage by the VMs running on the host, the default is `/var/lib/neat`;

6.3 TODO

- What data should be collected by the Data Collector?
- What data should be stored locally by the Data Collector?
- What is the format of the data?
- What data should be submitted to the database?
- What is the database schema?
- Define REST APIs for the Global and Local Managers
- Find out how to remotely switch hosts on or off

7 Implementation

This section should describe a plan of action (the "how") to implement the changes discussed. Could include subsections like:

7.1 Libraries

- pyqcy – a QuickCheck-like testing framework for Python.
- PyContracts – a Python library for Design by Contract (DbC).
- SQLAlchemy – a Python SQL toolkit and Object Relational Mapper.
- Sphinx – a documentation generator for Python.
- Bottle – a micro web-framework for Python.

7.2 UI Changes

Should cover changes required to the UI, or specific UI that is required to implement this

7.3 Code Changes

Code changes should include an overview of what needs to change, and in some cases even the specific details.

7.4 Migration

Include:

- data migration, if any
- redirects from old URLs to new ones, if any
- how users will be pointed to the new way of doing things, if necessary.

8 Test/Demo Plan

This need not be added or completed until the specification is nearing beta.

9 Unresolved issues

This should highlight any issues that should be addressed in further specifications, and not problems with the specification itself; since any specification with problems cannot be approved.

10 BoF agenda and discussion

Use this section to take notes during the BoF; if you keep it in the approved spec, use it for summarising what was discussed and note any options that were rejected.

11 References

[1] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers," *Concurrency and Computation: Practice and Experience (CCPE)*, 2012.

[2] A. Beloglazov and R. Buyya, "Managing Overloaded Hosts for Dynamic Consolidation of Virtual Machines in Cloud Data Centers Under Quality of Service Constraints," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 2012 (under review).

[3] J. Koomey, *Growth in data center electricity use 2005 to 2010.* Oakland, CA: Analytics Press, 2011.

[4] Gartner Inc., *Gartner estimates ICT industry accounts for 2 percent of global CO2 emissions*. Gartner Press Release (April 2007).

[5] R. Nathuji and K. Schwan, "VirtualPower: Coordinated power management in virtualized enterprise systems," *ACM SIGOPS Operating Systems Review*, vol. 41, pp. 265–278, 2007.

[6] A. Verma, P. Ahuja, and A. Neogi, "pMapper: Power and migration cost aware application placement in virtualized systems," in *Proc. of the 9th ACM/IFIP/USENIX Intl. Conf. on Middleware*, 2008, pp. 243–264.

[7] X. Zhu, D. Young, B. J. Watson, Z. Wang, J. Rolia, S. Singhal, B. McKee, C. Hyser, and others, "1000 Islands: Integrated capacity and workload management for the next generation data center," in *Proc. of the 5th Intl. Conf. on Autonomic Computing (ICAC)*, 2008, pp. 172–181.

[8] D. Gmach, J. Rolia, L. Cherkasova, G. Belrose, T. Turicchi, and A. Kemper, "An integrated approach to resource pool management: Policies, efficiency and quality metrics," in *Proc. of the 38th IEEE Intl. Conf. on Dependable Systems and Networks (DSN)*, 2008, pp. 326–335.

[9] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper, "Resource pool management: Reactive versus proactive or lets be friends," *Computer Networks*, vol. 53, pp. 2905–2922, 2009.

[10] VMware Inc., "VMware Distributed Power Management Concepts and Use," *Information Guide*, 2010.

[11] G. Jung, M. A. Hiltunen, K. R. Joshi, R. D. Schlichting, and C. Pu, "Mistral: Dynamically Managing Power, Performance, and Adaptation Cost in Cloud Infrastructures," in *Proc. of the 30th Intl. Conf. on Distributed Computing Systems (ICDCS)*, 2010, pp. 62–73.

[12] W. Zheng, R. Bianchini, G. J. Janakiraman, J. R. Santos, and Y. Turner, "JustRunIt: Experiment-based management of virtualized data centers," in *Proc. of the 2009 USENIX Annual Technical Conf.*, 2009, pp. 18–33.

[13] S. Kumar, V. Talwar, V. Kumar, P. Ranganathan, and K. Schwan, "vManage: Loosely coupled platform and virtualization management in data centers," in *Proc. of the 6th Intl. Conf. on Autonomic Computing (ICAC)*, 2009, pp. 127–136.

[14] B. Guenter, N. Jain, and C. Williams, "Managing Cost, Performance, and Reliability Tradeoffs for Energy-Aware Server Provisioning," in *Proc. of the 30st Annual IEEE Intl. Conf. on Computer Communications (INFOCOM)*, 2011, pp. 1332–1340.

[15] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing SLA violations," in *Proc. of the 10th IFIP/IEEE Intl. Symp. on Integrated Network Management (IM)*, 2007, pp. 119–128.

[16] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya, "A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems," *Advances in Computers, M. Zelkowitz (ed.)*, vol. 82, pp. 47–111, 2011.