Homayoun

Reference 40

Benchmarking Bare Metal Cloud Servers for HPC Applications

P. Rad[1], A T Chronopoulos[2], P. Lama[2], P. Madduri[2], C. Loader[2]
[1]Department of Computer Engineering, University of Texas at San Antonio
[2]Department of Computer Science, University of Texas at San Antonio
1 UTSA Circle, San Antonio, Texas 78249, U.S.A.

***Abstract*—Cloud Computing is an ever-growing paradigm shift in computing allowing user's commodity access to compute and storage services. As such cloud computing is an emerging promising approach for High Performance Computing (HPC) application development. Automation of resource provision offered by Cloud computing facilitates the eScience programmer usage of computing and storage resources. Currently, there are many commercial services for compute, storage, network and many others from big name companies. However, these services typically do not have performance guarantees associated with them. This results in unexpected performance degradation of user's applications that can be somewhat random to the user. In order to overcome this, a user must be well versed in the tools and technologies that drive Cloud Computing. One of the state of the art cloud systems, is a cloud system that provides bare metal server instances on demand. Unlike traditional cloud servers, bare metal cloud servers are free from virtualization overheads, and thus promise to be more suitable for HPC applications. In this paper, we present our study on the performance and scalability of Openstack based bare metal cloud servers, using a popular HPC benchmark suite. Our experiments conducted at UTSA Open Cloud Institute's cloud system with 200 cores demonstrate excellent scaling performance of bare metal cloud servers.**

Keywords- Bare Metal, Cloud computing, MPI, HPC benchmarks

I. INTRODUCTION

Today there are growing interests among the academic and commercial HPC users to utilize cloud computing as a cost effective alternative for their computing needs. Cloud computing offers the potential of reducing some of heavy upfront financial commitments associated with high performance computing infrastructure, while yielding to faster turnaround times [1]. HPC applications in the cloud can mainly benefit from on-demand elasticity of computing resources, and the pay-per-usage cost model. On the other hand, previous studies have shown that that commodity interconnects and the overhead of virtualization on compute, network and storage performance are major performance barriers to the adoption of cloud for HPC [2,3,4].

Recent efforts towards HPC-optimized clouds, such as Magellan [5] and Amazon's EC2 Cluster Compute [6], are promising steps towards overcoming the performance barriers in the cloud environment. However, these solutions still involve the use of traditional virtualization softwares such as Xen, KVM, etc., which introduce significant performance overheads to HPC applications. In this paper, we present an extensive performance evaluation of a new Cloud technology that offers bare metal servers on demand. The idea of bare metal cloud servers is to give the end users full processing power of the physical server without using a virtual layer. Bare metal cloud servers are single-tenant systems that can be provisioned in minutes, and that allow users to pay by the minute. Subsequently, some of the obstacles of the basic cloud computing approach due to virtualization technology are resolved. First of all, computation intensive applications can request a certain type of server, thus the Service Level Agreement (SLA) is very clear from the providers point. Secondly, users know about the servers they are using and can tune the BIOS and system configurations in order to obtain the highest level of performance. And last but not least, since the server is not shared among multiple tenants, no one can interfere with the performance of the machine [22]. This technology removes the overheads of virtualization, while providing the high availability and elasticity of the cloud.

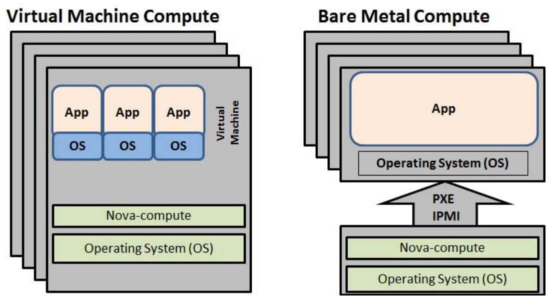

Figure 1. Bare Metal vs. Virtual Machine Compute in Openstack

There are many bare metal resource provisioning frameworks available in market today [23, 24, 25]. These frameworks automate the deployment of operating systems in a datacenter. One of the key challenges of bare-metal provisioning algorithm lies in scheduling the appropriate servers from the datacenter. It is very challenging to optimize the scheduling of heterogeneous resources mainly due to the number of variables involved in making a decision. In general, it is considered to be a NP hard problem [23]. In this paper, we have evaluated Openstack's new approach of cloud integrated bare-metal provisioning framework plug-in [26]. It is best thought of as a bare metal hypervisor API and a set of plugins which interact with the bare metal hypervisors similar to virtual machines. The main rational behind this approach is to make the cloud Application Programming Interface (API) self-sufficient and enable a single cloud platform that can launch both bare metal and virtual machines. Figure 1 illustrates the difference between bare metal and virtual machine provisioning in a cloud environment managed by Openstack. By default, the bare metal provisioning uses PXE and IPMI in concert to provision and turn on/off machines, but it also supports vendor-specific plugins which may implement additional functionality.

To the best of our knowledge, this is the first paper that evaluates the performance of HPC benchmarks on a bare-metal cloud platform. For performance evaluation, we used UTSA Open Cloud Institute's bare metal cloud servers to run the HPCC (High Performance Computing Challenge) [7] benchmark suite. Bare metal provisioning is enabled by OpenStack Ironic, an integrated OpenStack program which provisions bare metal machines, forked from the Nova baremetal driver. Our results demonstrate excellent scaling performance of bare metal cloud servers with 200 cores.

The remainder of the paper is organized as follows. Sections II and III provide the background, an overview of related work and our approach for cloud-based bare-metal provisioning. Section IV describes our methodology of automating HPC testbed setup in the cloud. Sections V, and VI present a brief introduction to the benchmarks we used and the results of our evaluations. Section VII concludes the paper with directions for future work.

II. RELATED WORK

High performance computing in the cloud has gained significant research attention in recent years [8,9,10,11,18,19,20]. Marathe et al. [10] evaluated the cloud against traditional high performance clusters along turnaround time and cost. Walker [2], followed by several others [3,10,12], conducted the study on HPC in cloud by benchmarking Amazon EC2 [13]. He et al. [14] experimented with three public clouds and compared the results with dedicated HPC systems. These studies show that interconnect latency and virtualization overheads in the cloud environment impose major performance barriers to HPC applications.

In a recent study, Gupta et al. [15] evaluated HPC benchmarks using two lightweight virtualization techniques, thin VMs configured with PCI pass-through for I/O, and containers, that is OS-level virtualization. Lightweight virtualization reduces the latency overhead of network virtualization by granting virtual machines native accesses to physical network. On the other hand, Containers such as LXC [16] share the physical network interface with its sibling containers and its host. This study showed that thin VM and containers impose a significantly lower communication overhead.

With the advent of state-of-the art cloud technology such as bare metal server provisioning, the performance of HPC applications in the cloud environment is expected to improve further with respect to both computation and communication workloads. Unlike traditional cloud servers, bare metal cloud servers are free from virtualization overheads. However, these emerging cloud platforms have so far not been evaluated extensively for HPC applications.

III. CLOUD ORCHESTRATION WITH BARE METAL CAPABILITY

A. Overview of Bare-Metal Provisioning Frameworks

This section will briefly discuss the bare-metal frameworks prior to our cloud-based bare-metal provisioning.

1) Cobbler is an open source server provisioning system that allows for rapid setup of network installation environments. The cobblerproject was initiated by RedHat and now functions independently [24].
2) Canonical MaaS is an open source bare metal provisioning helps in deploying Ubuntu onto multiple bare-metal machines using (Intelligent Platform Management Interface) IPMI [23].
3) Razor is a bare metal provisioning framework built by Puppet Labs to deploy and configure multiple machines simultaneously [25].
4) Emulab is a network testbed which provides an environment to carry out research in computer networks and distributed systems. To provision bare hardware systems, Emulab takes a user defined network topology in a Network Simulator file and configures the topology.

B. Bare Metal Cloud

Figure 2. shows the architecture of Bare Metal Cloud used in this paper. The main rational behind our approach is to make the cloud scheduler and the cloud Application Programming Interface (API) self-sufficient and make it a single platform that can launch bare metal and virtual machines.

We use cloud based bare-metal provisioning operated by the Open Cloud Institute (OCI) at the University of Texas at San Antonio. It offers significant capacity and similar design features found in Cloud Computing providers, including robust compute capability and elastic infrastructure design.

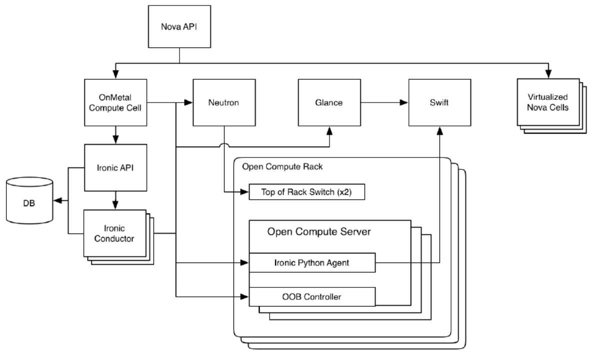

Figure 2. Bare Metal Cloud Architecture

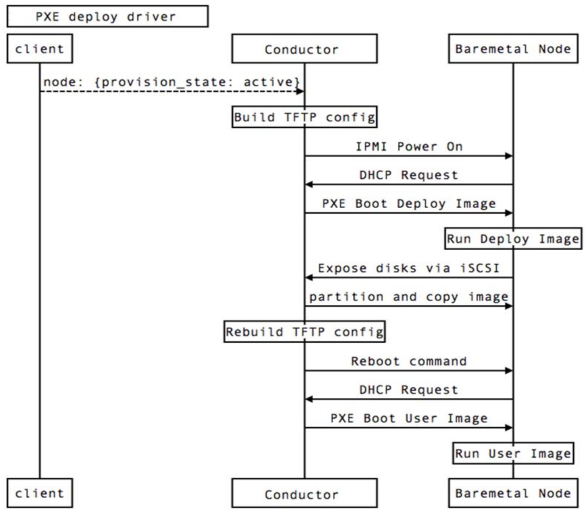

Figure 3. Bare metal provisioning state diagram [28]

As shown in Figure 2, the required steps to boot a bare metal compute node are as follows:

1) Authenticate with keystone
2) Send boot request to Nova API
3) Send boot request to Nova Scheduler to select the host for bare metal deployment
4) Send boot request to the bare metal host
5) Ironic Conductor gets the image from Glance
6) Configure network using Neutron
7) Set deploy request to Ironic API
8) Deploy host – the deployment flow is based on the Ironic driver used.

As an example, the complete deployment flow using the PXE driver is shown in Figure 3.

IV. CLOUD AUTOMATION FOR HPC TESTBED

In order to setup a large scale HPC testbed in the cloud, we developed automation scripts required for the installation, and configuration of OpenMPI and other related software. For this purpose, we used Ansible, an automation engine that automates cloud provisioning, configuration management, application deployment [29]. Ansible allows us to write playbooks and then put a script containing commands to run the playbooks onto the proxy server to distribute the commands to each of the cloud servers as shown in Figure 4.

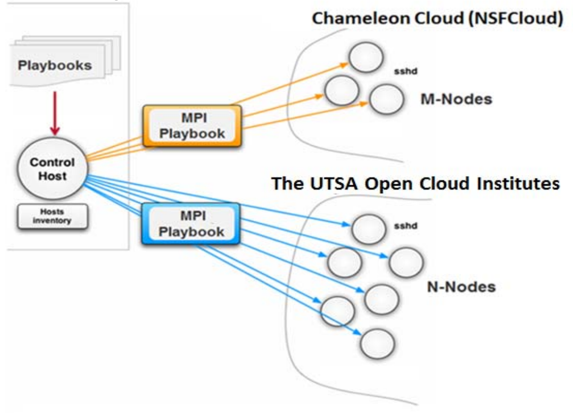

Figure 4. Cloud Automation with Ansible

In future, we plan to run the HPC benchmarking experiments on NSFCloud (Chameleon [30]) using compute nodes with faster processors (intel haswell) and memory (DDR4).
The examples of Ansible playbooks that we developed are as follows.

Playbook for configuring SSH-keys

```
- name: configure SSH on servers
  hosts: servers
  sudo: true
  vars:
  remote_user: root
  tasks:
  - name: Installing required essentials
    apt: name=build-essential state=installed
  - name: Generating SSH Keys
    user: name=root generate_ssh_key=yes
  - name: Obtaining Keys
    fetch:      src=~/.ssh/id_rsa.pub      dest=~/.ssh/tmp/
recursive=yes
  - name: Copying SSH keys to Machines
    copy: src=~/.ssh/ dest=~/.ssh/ directory_mode
  - name: Adding to the list of authorized_keys
shell:cat~/.ssh/tmp/166.78.164.*/root/.ssh/id_rsa.pub>>
~/.ssh/authorized_keys
```

The first part of the playbook configures the SSH-keys and the communication between machines. We did this by first generating a SSH-key on each machine, then fetched them to the host machine then we copied the contents of each machines public key file to each individual machines authorized_keys file. This ensured that each machine could communicate with each other without a password.

Playbook for installing OpenMPI

```
- name: configure OpenMPI on servers
  hosts: servers
  sudo: true
  vars:
  remote_user: root
  tasks:
   - name: install the required packages for OpenMPI
     action: apt package={{item}} state=installed
     with_items:
     - openmpi-bin
     - openmpi-checkpoint
     - openmpi-common
     - openmpi-doc
     - libopenmpi-dev
```

After SSH-keys are configured, we installed the OpenMPI required packages. We used the apt module in Ansible to make sure those packages were installed.

Playbook for modifying and creating mpi_hosts

```
- name: configure mpi_hosts file on host
  hosts: host
  sudo: true
  vars:
  remote_user: root
  tasks:
   - name: moving inventory to host file
     action: copy src=inventory dest=/root/
   - name: renaming inventory to mpi_hosts
     command: mv inventory mpi_hosts
   - name: editing mpi_hosts file
     command: sed -i '1,2d' mpi_hosts
   - name: continuing editing
     command: sed -i '/[servers]/d' mpi_hosts
```

The final step was to generate a mpi_hosts file (which tells which machines to use). This step was merely copy the inventory file and use sed commands to modify the file. This has allowed for easy configuration of a large number of machines, much faster than a batch script.

V. HPC BENCHMARKS

For performance evaluation, we ran various compute intensive and communication intensive benchmarks from the **HPCC (High Performance Computing Challenge)** benchmark suite. The HPC Challenge benchmark consists at this time of benchmarks such as HPL, Random Access, PTRANS, FFT, DGEMM and Latency Bandwidth. HPL is the Linpack TPP benchmark.

We built the HPCC workload with the ATLAS (Automatically Tuned Linear Algebra Software) math library. ATLAS provides highly optimized Linear Algebra kernels for arbitrary cache-based architectures [17]. ATLAS provides ANSI C and Fortran77 interfaces for the entire BLAS API, and a small portion of the LAPACK API.

HPL

The HPL benchmark measures the ability of a system to deliver fast floating point execution while solving a system of linear equations [32]. Performance of the HPL benchmark is measured in GFLOP/s.

RANDOM ACCESS

The HPC Challenge Random Access benchmark evaluates the rate at which a parallel system can apply updates to randomly indexed entries in a distributed table. Performance of the Random Access benchmark is measured in Giga Updates per second (GUP/s). GUPS (Giga Updates per Second) is a measurement that profiles the memory architecture of a system, and is a measure of performance similar to MFLOPS. The HPCS HPC challenge Random Access benchmark is intended to exercise the GUPS capability of a system, much like the LINPACK benchmark is intended to exercise the MFLOPS capability of a computer. In each case, we would expect these benchmarks to achieve close to the "peak" capability of the memory system. The extent of the similarities between Random Access and LINPACK are limited to both benchmarks attempting to calculate a peak system capability.

PTRANS (parallel matrix transpose)

PTRANS measures the rate of transfer for large arrays of data from multiprocessor's memory. PTRANS exercises the communications where pairs of processors communicate with each other simultaneously. It is a useful test of the total communications capacity of the network.

FFT

The HPC Challenge FFT (Fast Fourier Transform) benchmark measures the ability of a system to overlap computation and communication while calculating a very large Discrete Fourier Transform of size m with input vector z and output vector Z [31]. Performance of the FFT benchmark is measured in GFLOP/s. FFT measures the floating point rate of execution of double precision complex one-dimensional Discrete Fourier Transform (DFT).

LATENCY BANDWIDTH

A set of tests to measure latency and bandwidth of a number of simultaneous communication patterns. Latency/Bandwidth measures latency (time required to send an 8-byte message from one node to another) and bandwidth

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | **sales@docketalarm.com** | **1-866-77-FASTCASE**

fastcase® Smarter legal research.