High Performance and Lightweight Mobile Cloud Infrastructure Monitor and Benchmark Service using RESTful, JMS and JSON

Dai Hoang Tran, Chuan Pham, Cuong T.D, Dung T.N,
Nguyen T.H, Eui-Nam Huh, Choong Seon Hong
Department of Computer Engineering, Kyung Hee University
Email: {dai.tran, pchuan, dtcuong, ntiendung, johnhuh, cshong}@khu.ac.kr

Abstract

In the area of cloud infrastructure environment, the management tool to monitor and control the cloud resources is the important factor that can drive the cost benefit of the cloud vendors. But most of these tools are bundled within the high cost commercial platforms and are optimized to run on desktop computers. With the vision that Mobile Cloud Computing will be the future technology paradigm that dominates the IT industry, we aim to create a cloud management tool which is open source, fast, lightweight and mobile friendly. We take the initial steps by implementing our framework using several popular technologies such as RESTful, Java Message Service, JSON, and we call it "High performance and Lightweight Mobile Cloud Infrastructure Monitor and Benchmark Service" or HiLiCloud. The initial testing show competitive evaluation results.

1. Introduction

Mobile devices such as smartphone and tablet are gradually becoming an essential part of our life. They are effective and powerful communication devices. Their rich environment of applications and services become a powerful new trend in the IT technology development (e.g., Android Google application). However, mobile devices still face many challenges because of their limited resource capacity such as battery life, storage, network bandwidth, etc [1].

On the other hands, Cloud computing (CC) is considered to be the next generation's computing infrastructure that provides low cost hardware and software utilities to users by leveraging its powerful and modular infrastructure. As a result, the combination of mobile devices and cloud computing complements each other's, forming what is called "Mobile Cloud Computing" (MCC). MCC brings brand new type of services and facilities to mobile users with the power of cloud computing [2]. In the space of cloud infrastructure environment, there are always a need for cloud management tools to monitor and assess the cloud resources. Usually these applications are bundled within the high cost commercial such as VMWare Horizon Suite 3. As mobile devices are now ubiquitous and MCC now becomes the strong back-end for mobile devices, we want also the ability to dynamically access, control and monitor our cloud resources on the these devices.

As for such requirements, we are building a framework for cloud resources management that could be used on mobile devices and meet its characteristics such as battery life and network bandwidth limitation. We call it "High Performance and Lightweight Mobile Cloud Infrastructure Monitor and Benchmark Service" (HiLiCloud). The main principles of HiLiCloud are that the response time has to be minimum, response data has to be lightweight, and the system memory footprint has to be minimal. Thus we give more focus to the design and implementation of communication between components that is applying RESTful model, JSON data format [3] and JMS technology.

The remaining of the paper is organized as follows: section II details the design and implementation of our framework HiLiCloud, section III shows how the HiLiCloud performs in our initial testing stages, and finally we draw our conclusion in section IV.

2. HiLiCloud Framework

HiLiCloud framework has its focus on performance and mobile friendly, so the designed architecture is composed of several modules that provide high throughput and fast, lightweight responses. Figure 1

shows the HiLiCloud framework. At the heart of HiLiCloud is the high performance Asynchronous JMS Handler (AJH), that responsible for generating and digesting JMS messages between modules. The AJH queues its messages in ActiveMQ broker, a popular open-source Message Broker 5. When a HTTP request is initiated by an end-user or an administrator, the Balancer receives the incoming request and authenticates the user. Depend on the user's type, it will expose full (for administrator) or partial (for normal user) of its RESTful API [4]. User now can invoke subsequent requests for the framework services. The Balancer then delivers the requests in form of JMS message to the Controller module, and as mentioned above, all of the delivery is handled by AJH. The Controller modules consists of smaller components that make the decision for different types of service.

One particular important component of the Controller is the Status Repository (SR). The SR will receive and store all the information of the whole system. It will be the central hub for other components to make intelligent analysis, derive behaviors and instrument other modules' activities. Its source of information comes from the Hypervisor Monitor, which tracks hypervisor activities using third party library called LibVirt, and the labor Service Virtual Machines (Service-VMs).

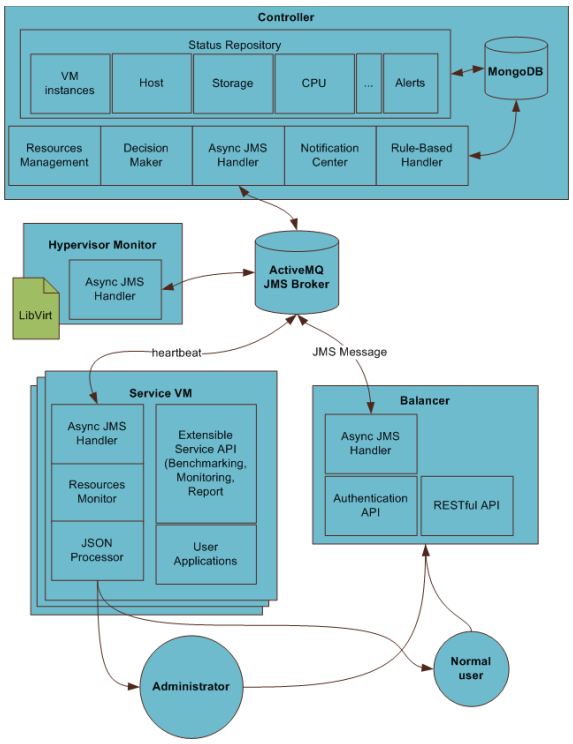

Figure 1: HiLiCloud Architecture

The Service-VMs provides framework services to users. It maintains few sub-modules such as Resources Monitor, Extensible Service API (ESA) and the JSON Processor. The Resources-Monitor keeps track of Virtual Machine status, and regularly updates is status back to the SR in the Controller by a defined interval. We call these update actions with the name "heartbeats". The ESA is the set of services provided by HiLiCloud that users can query for. ESA is designed to follow a strict and general application interface, thus gives us the flexibility in adding new services when needed. Currently, we provide three main services, they are Benchmarking, Monitoring and Report services. When the Service-VM finishes its invoked service task, the result is passed to JSON-Processor sub-module. JSON Processor deals with the serialization of response data into JSON format, and gives HTTP responses to users in JSON format.

3. Evaluation

3.1 Testbed environment

To evaluate our HiLiCloud framework performance, we set up our testbed, in which one physical machine hosts the Controller and Hypervisor monitor (HM), and three VMs are created to run as service-VMs and Balancer. From Figure 2, we can see that the entry to the framework is the VM-3 that hosts the Balancer module, and exposes its public IP address. VM-1 and VM-2 respectively serve as Service-VMs which will send the service's responses to the users. The physical machine hosts three modules, the HM to monitor the hypervisor of the physical server, the Controller to have the grand control of the framework, and the ActiveMQ broker for message delivering of the whole system. Our machine has the configuration of Intel Xeon CPU E3-1220 V2, 3.10GHz with 4 cores, 32 GB memory and 1024 GB of disk storage. Each service-VM is configured with 1 virtual CPU, 4 GB of memory and 50 GB of storage. The operating system (OS) environment of the physical and the VMs are identical. They all run Linux Ubuntu OS 14.04, with supported Java platform version 1.7, MongoDB version 2.6 and ActiveMQ version 5.10.

3.2 Results evaluation

We are still actively working on implementing the HiLiCloud framework. While it is not possible to show all the framework evaluation aspects at this moment, we focus on the first few evaluation metrics, such as the mean response time (MRT), the memory usage

(MU) and CPU load (CL) of the HiLiCloud framework. Up on completion of the framework, we will in-cooperate all the evaluations in the future work. For the metric of response time, we want to test the stability and performance of framework under heavy load. We wrote a benchmark service that a normal user can access at the RESTful URL pattern http://<ip>/HiLiCloud/vm/all/benchmark/response_time/<params>, whereas params value specify the amount of requests. The response of this benchmark is a random string of JSON text in about 400-500KB size. We run the test multiple times, with params value ranges from 50, 100 to 1000, 2000 and run in multiple devices in different network conditions. Figure 3 shows the benchmark result. To find relative contexts for comparison, we run the benchmark on normal desktop with wire Internet connection, mobile with wireless and 3G connection. From the results, it is obvious that the response time from the wired PC connection is very small and consistent. The MRT of mobile device show some interesting aspects. For a small amount of requests, the performance is somewhat consistent, but for large request quantity, we see that the 3G connection's MRT approaching the wireless connection's MRT. This indicates that our framework performance is fluctuated when the amount of request is so dense in a short period of time, we need to do deep investigation in the future work. For the MU and CL evaluation from Figure 4, it shows that our framework does not consume that much memory when different services are invoked, and the CL shows low CPU peak load for different kind of benchmarks. Thus it shows its lightweight characteristic.

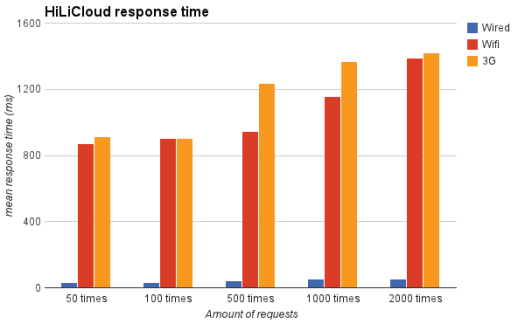

Figure 3: Response time benchmark

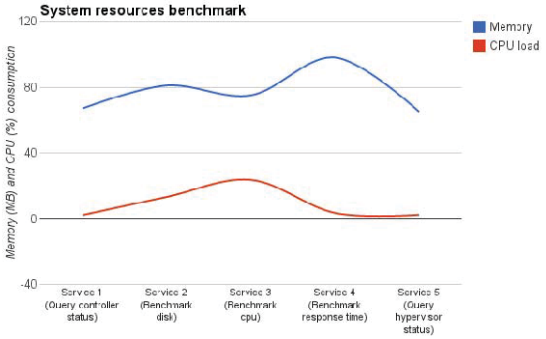

Figure 4: System performance benchmark

4. Conclusion and Future Work

In conclusion, by implementing HiLiCloud, we take initial steps to fulfill the lack of mobile friendly cloud management framework, which is lightweight and provide good performance using distributed communication tools and data format. Even though our framework achieve fairly competitive performance, we believe that we can give it more tuning and optimization to speed up the whole framework. The SPDY protocol is the next option we consider for the improvement of HiLiCloud.

Acknowledgment

This research was supported by the MSIP (Ministry of Science, ICT&Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (NIPA-2014(H0301-14-1020)) supervised by the NIPA (National IT Industry Promotion Agency). *Dr. CS Hong is the corresponding author

References

[1] M. Satyanarayanan, Fundamental challenges in mobile computing, in Proceedings of the 5th annual ACM symposium on Principles of distributed computing, pp.1-7, May 1996.

[2] Niroshinie Fernando, Seng W. Loke, and Wenny Rahayu. 2013. Mobile cloud computing: A survey. Future Gener. Comput. Syst. 29, 1 (January 2013)

[3] Maeda, K.,"Performance evaluation of object serialization libraries in XML, JSON and binary formats," Digital Information and Communication Technology and its Applications (DICTAP), 2012 Second International Conference on, vol., no, pp.177, 182, 16-18 May 2012

[4] R. T. Fielding, Architectural styles and the design of network-based software architectures, Ph.D. Dissertation, University of California, Irvine, 2000.