Architecture of Mobile Cloud Computing and Middleware

Chanky Swami[1], Nishant Anand[2]

[1]Maharshi Dayanand University, CBS Group of Institutions,
Vill. Fatehpuri, Jhajjar Road , Haryana

[2]Assistant Professor, CBS Group of Institution
Vill. Fatehpuri, Jhajjar Road, Haryana

Abstract

Today large number of mobile devices accessing WS and other services from hosting by the cloud network .As this is a new technology so there is a lot to do in order to make this connectivity efficient So, Mobile Cloud Computing (MCC) architecture provides a proxy for mobile clients connecting to Cloud services. This architecture efficiently helps mobile device in accessing WS over cloud and overcome major problem like loss of connection, network bandwidth and also limited resources.

Keywords: Wireless, Sensor Nodes, Outlier, Clustering, Gateway

1.INTRODUCTION

The goal of the Mobile Cloud Computing (MCC) architecture is to provide a proxy for mobile clients connecting to Cloud services. The architecture consists of three parts, the mobile clients, the middleware and the Cloud services. Since Cloud services are usually controlled by service providers, the middleware performs all the necessary adaptation to the mobile clients.

Some services require real-time updates, for example, news, Blog, and Twitter service. The middleware also pushes updates of service results to mobile clients via HTTP or email immediately after it receives the updates. The middleware is responsible for consuming the Cloud Services whether they are SOAP or RESTful WS and delivers the service result to the mobile client. On the mobile client, users can define WS or mashup services and later execute the pre-defined WS on the fly. The middleware provide RESTful WS interface for the mobile clients. Note that the execution starts with a HTTP GET request whose URL path contains the resource identifier to the WS. When WS are executed through the middleware, the follow steps are involved in the middleware.

- The mobile client sends a HTTP GET request with an identifier of a WS to the middleware.
- The middleware deals with interactions to the WS (and generates SOAP WS client if necessary).
- The middleware extracts (JSON or XML parsing) the required service results from the original service result and form a new service results in JSON format.

The middleware stores a copy of result with the service ID in the database and returns the optimized result to the mobile client.

2. CLOUD COMPUTING

The combination of virtualization, distributed computing and the service-oriented architecture creates a new computing paradigm, called Cloud Computing. According to VOUK Cloud computing embraces cyber infrastructure which is one the key elements of successful information technology (IT). Based on the level of abstraction, VAQUERO defines three major scenarios in cloud computing.

Infrastructure as a Service (IaaS) refers to service that exposes the hardware resources to users. Amazon EC2 is a successful IaaS implementation in the market.

Platform as a Service (PaaS) provides computational resources as high level application platforms. Google App Engine (GAE) is an example of PaaS.

Software as a Service (SaaS) focuses on exposing software functions as services (i.e. WS).Many service providers including Google, Yahoo, and Amazon offers their software functions as WS. Programmable Web collected thousands of Web APIs from various categories.

Ostermann et al. did an early performance evaluation of Cloud Computing by comparing Amazon EC2 to scientific computing infrastructure such as grids and PPIs. For a single job with a single EC2 instance, the CPU performance for floating point and double operation is 6-8 times lower than the claimed maximum of ECU (CPU unit defined by Amazon, one ECU equals 4.4 gigaflops per second) and the sequential IO operation has generally better performance compared to similar systems. For a single job with multiple EC2 instances (clusters), efficiency decreases with the increase of EC2 instances, due to the high network latency. However, for some jobs such as DGEMM, STREAM and Random Access [35], EC2 clusters have similar or better performance than HPC clusters.

There are several open Cloud implementations. VOUK presented an IaaS implementation based on Virtual Computing Laboratory (VCL). The end nodes include IBM Blade Centre blades and computers in a university lab. The VCL implementation provides similar services like Amazon EC2, Map Reduce environment, and sub-cloud for Grid Computing. Running since 2004, the VCL

implementation reveals some open issues, like Cloud provenance data, utilization, optimization and portability of image.
Another open IaaS implementation similar to Amazon EC2 is Eucalyptus. From the entry-point to end-node, there are four controllers: Cloud, Storage, Cluster, node controller.
They all communicate through WS interfaces. Instances are run as Virtual Machines (VM) on the end-nodes where node controllers are installed. At the cluster level, VM instances are interconnected via a Virtual Network (VN) which grantees connectivity of single access of a "set" of instances, isolation of separated Cloud allocation, and performance with option of choosing native network without VN. Eucalyptus also provides a storage service for VM images and user data similar to Amazon S3.
In summary, Cloud Computing is a new computing paradigm which aims to reduce the cost of both development and deployment. However, the real implications of using Cloud Computing vary in each case. Most Cloud systems are proprietary, and rely on infrastructure that invisible to researchers [41]. Hence, there are restrictions imposed by providers. Open Cloud implementations like Eucalyptus, provide an easy solution for IaaS and opens the possibility of creating private Cloud, but there are some issues that needs to be considered

3. IDEA OF MCC

To overcome these challenges, I propose a Mobile Cloud Computing (MCC) architecture which connects mobile devices to the Cloud Computing. The MCC architecture includes a mobile client and a middleware design.
There are two approaches to implement the mobile client: native applications and embedded browser applications. Native applications are built with specific programming languages supported by the mobile platforms. However, embedded browser applications can run HTML and JavaScript in the embedded browser and use interfaces exposed by native application

There are two approaches to implement the mobile client: native applications and embedded browser applications. Native applications are built with specific programming languages supported by the mobile platforms. However, embedded browser applications can run HTML and JavaScript in the embedded browser and use interfaces exposed by native application.

4. MCC ARCHITECTURE

The goal of the Mobile Cloud Computing (MCC) architecture is to provide a proxy for mobile client connecting to Cloud services.

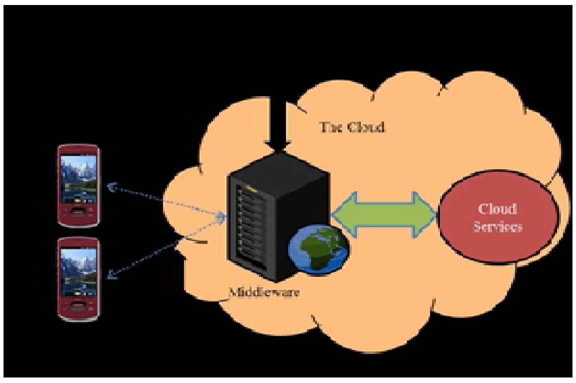
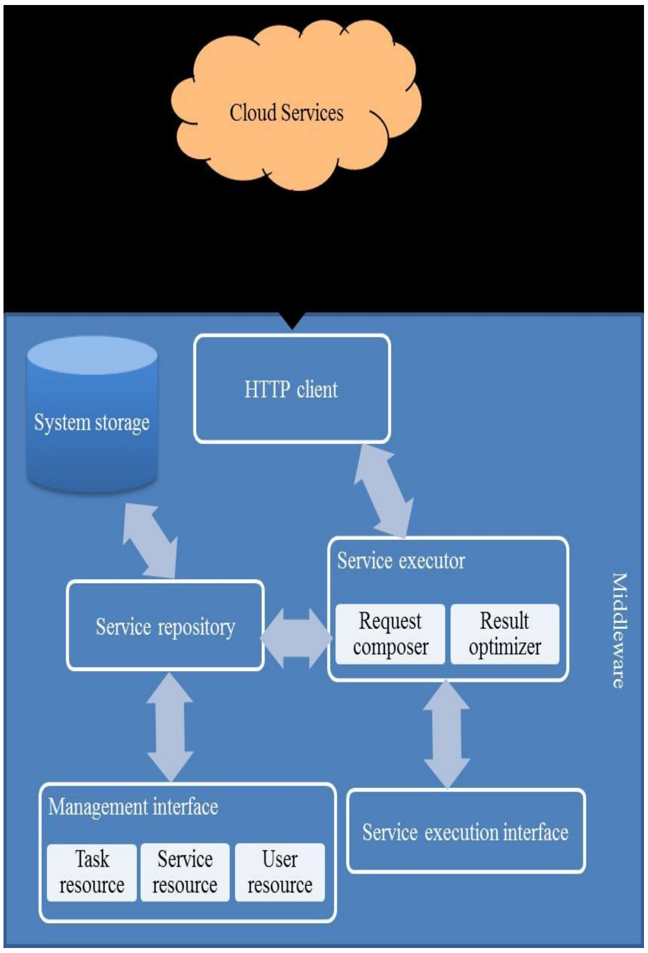


Figure shows an overview of the MCC and its main features. The architecture consists of three parts, the mobile clients, the middleware and the Cloud services. Since Cloud services are usually controlled by service providers, the middleware performs all the necessary adaptation to the mobile clients.
Some services require real-time updates, for example, news, Blog, and Twitter service. The middleware also pushes updates of service results to mobile clients via HTTP or email immediately after it receives the updates

5. MIDDLEWARE ARCHTICTURE

The middleware has a RESTful service interface for mobile clients. Through the management interface, users can define and manage user profile, Mashup Services, Service Actions, and their parameters and results. All the



requests through the management interface are passed to the service repository which reads and write data from and into the storage. The execution requests of Service Actions go through the service execution interface. These requests are primarily mapped to read operations in the service repository.
The service executer composes service requests and passes them to the HTTP client which sends outgoing request to Cloud services.
In General middleware provide following features.
Middleware pushing: Mobile clients can subscribe to service resources and explicitly update service results cached on the middleware through the management interface.
When the middleware receives an update of service results, it sends the update to all the mobile clients that subscribed to this service result. The update is pushed to the clients (e.g. via email).
Optimizing results: An unprocessed WS response contains data within a service specific format. However, there are two problems. First, the mobile clients do not need all the data. For example, the user may only need 5 instead of 10 news stories. Second, the original data format may also not be efficient for mobile clients. The result optimizer first extracts the required part of data from the raw response, and then makes a copy of the extracted result in various formats, for example, mobile HTML for mobile browsers and JSON for native mobile applications. The middleware also caches these copies of result in the service repository.
Middleware Caching: Caching is based on the mashup services. The service repository save the optimized service results into system storage for the latest execution of the mashup services. The service results update when the parameters of a mashup service change. Users can also clean the cache via the management interface.

6. COCLUSION AND FUTURE SCOPE

Hence this architecture provides an efficient way to use our mobile and cloud services because earlier we are not efficiently using our resources either on device side or on cloud side. Provide us cache mechanism with scalability as we can use this architecture with most of the famous cloud platforms like Amazon EC2 and Google App Engine. However there will always a need to customize the middleware and the cloud platforms according to end user need.

REFERENCES

[1]. Portio Research Mobile Factbook, Portio Research, 2009.
[2]. S. Yates, It's Time To Focus On Emerging Markets For Future Growth, Forester, 2007.
[3]. S. Weerawansa, F. Curbera, F. Leymann, T. Storey, and D.F. Ferguson, Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More, Upper Saddle River, NJ, USA: Prentice Hall PTR, 2005.
[4]. "Web Services Glossary," 2004. Last retrieved from http://www.w3.org/TR/ws-gloss/ on December 6, 2010
[5]. "Web Services Description Language 1.1," Web Services Description Language (WSDL) 1.1, 2001. Last retrieved from http://www.w3.org/TR/wsdl on December 6, 2010
[6]. "UDDI version 3.02 Spec Technical Committee Draft," 2004. Last retrieved from http://uddi.org/pubs/uddi-v3.0.2-20041019.htm on December 6, 2010
[7]. "Web Services Architecture," 2004. Last retrieved from http://www.w3.org/TR/ws-arch/ December 6, 2010

AUTHOR

Chanky Swami received B.Tech.in information and technology and M.Tech. degree in Computer Science & Engineering from Haryana Institute Of Technology in 2011 and CBS Group of Institutions in 2015,respectively. Both institutions are affiliated by Maharshi Dayanand University, Rohtak, Haryana. In 2011 he worked with Networkershome pvt. Ltd. and was in touch with the cloud technology. After that he worked with HCL Technologies and provided support services on Unix and databases of client.