Mobile Cloud Computing: Case Studies Using Java

Puvvala Supriya
Student (M.Tech) , CSE,
Gokul Institue of Technology and Science,
Visakhapatnam, India.

P.Sandhya
Asst. Prof, CSE,
Gokul Institue of Technology and Science,
Visakhapatnam, India.

Abstract:

The Java RMI system is not very reliable; if the server shuts down, all data and computation occurring on the server is lost. This is not a hopeless scenario, because the client could simply resend the data and computation directions to another server. Java RMI will notify when the server cannot be reached, so if there was another server ready, the client should be able to automatically move to that server. The Android app could improve by switching to a local chess engine if a network connection is lost or unavailable. The current android application does not have a local copy of the chess engine, so game play will not be able to continue if a connection to the server is dropped. However, this would not be hard to rectify by adding the chess engine algorithm to the local code and having the application switch to local execution when I connection error is received.

I. Introduction:

In the early days of computing technology, when computers took up the space of an entire room, many 'dumb' terminals, or clients, would be connected to a main computer. Many clients could utilize the computational power and storage of the mainframe at the same time. As transistors and CPUs came into play, shrinking personal computers, it became more feasible for a user to purchase their own computer. However, today, mobile devices are becoming smaller and smaller and we are seeing that there is either a physical or economic limit to the amount of storage and processing power that can fit into these devices. It seems that the original model of client-mainframe computing may be a good answer for this situation.

However, we can now utilize existing wireless networks to connect mobile devices to servers in massive datacenters, rather than hardwiring all clients to a server. This idea of connecting to unseen data may be where the term "cloud" came from, since it seems that the extra power is coming out of nowhere. Companies are only just beginning to investigate the possibilities of the cloud and provide cloud services for business and personal use. There is much potential in utilizing the resources of the cloud,most of which has not been researched yet.

Client machines can become much more powerful by connecting to these cloud datacenters, but what are the options of doing so? Furthermore, integrating mobile devices with the cloud could prove even more advantageous. As these devices become smaller and smaller, consumers are conversely demanding more functionality and features. Bridging the gap between high-end servers and mobile devices could solve the computing problem, though research is needed to identity the advantages and limitations.

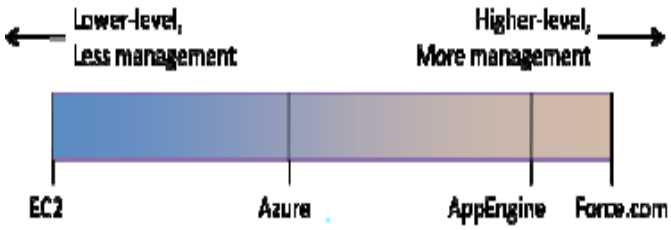

Figure 1.1: CLOUD COMPUTING MANAGEMNET OFFERINGS

II SURVEY OF THE STATE OF THE ART:

Software as a service (SaaS): Complete application systems delivered over the Internet on some form of "on-demand" billing system. Examples include Salesforce.com, which provides software for tracking sales, accounts, contacts, etc. and WebEx, which provides online desktop sharing and conference calling.Platform as a service (PaaS): Vendors provide development platforms and middleware, allowing developers to simply code and deploy without directly interacting with underlying infrastructure. Examples include Google AppEngine and Microsoft Azure, Infrastructure as a service (IaaS): Raw infrastructure, such as servers and storage, is provided directly as an on-demand service. Examples include Amazon Web Services and GoGrid.There are two different ways providers and users view IaaS. The cloud can either act as a datacenter, strictly providing the hardware to users, or it can provide more services, but less user control. Cloud Infrastructure providers are in the business of providing you equivalent datacenter functionality in the cloud using their scale for cost-effective service delivery. They must also package this functionality to provide you a high level of control as it's no longer your datacenter.

"Cloudcenters," datacenters in the cloud, focus on making your Cloud Infrastructure look very much like infrastructure you already have or are already familiar with, while Infrastructure Web Services ask you to embrace a new paradigm. Cloudcenters provide a direct equivalent to traditionaldatacenters and hence are usually more desirable for IT staff, systems operators, and other datacenter savvy folks. Infrastructure Web Services on the other hand are more analogous to Service-Oriented-Architectures (SOA), require significant programming skills, and are much more comfortable for software developers. The three most popular consumer cloud providers currently are: Microsoft Azure, Google AppEngine, and Amazon's EC2. Microsoft Azure and Google AppEngine are closer to the definition of Platform-as-a- Service then Infrastructure-as-a-Service. In terms of providing IaaS, those two services are limited.

2.1 Google App Engine:

This is a platform for developing and hosting web applications in Google-managed data centers. Google-managed is the key word in that sentence because Google maintains tight control over this service. Below is an example of some of the Google's restrictions:

- Threads cannot be created; one can only modify the existing thread state.

- Direct network connections are not allowed; URL connections can be used instead.

- Direct file system writes are not allowed; memory, memcache, and the datastore are used instead. (Apps can read files which are uploaded as part of the apps.)

- Java2D is not allowed.

- Native Code not is allowed; only pure Java libraries are allowed.

Google even places restrictions of the timing of requests. Each request gets a maximum of 30 seconds in which it has to complete or the AppEngine will throw an exception. If you are building a web application which requires large number of datastore operations, you have to figure out how to break requests into small chunks such that it does complete in 30 seconds. You also have to design a way to detect failures such that clients can reissue the request if they fail. Though seemingly very restrictive, Google App Engine does provide a simple gateway to cloud computing. Google provides a free software development kit that allows a user to easily setup a servlet programming environment.

Inexperienced web programmers can have their own server running on the internet fairly quickly. Communication between the client and the app engine uses HTTP requests. Thus, if required, java objects can be passed to the server using serialization and xml documents. Its availability and accessibly made App Engine a great choice as a commercial server to use with the mobile chess application demonstration.

III. REMOTE EXECUTION METHODS:

The first choice that needed to be made is how to implement remote execution. As mentioned previously, there are two main options: method/function migration and VM migration of the entire OS. I chose to work with function migration as the focus of my research has a more narrow scope dealing with individual applications. This holds true for both the remote execution of the NASA Benchmarks and desktop chess game using Java RMI as well as the android chess application using HTTP to connect to Google's App Engine.

The next step was deciding how to load the executable code on the server: dynamically at runtime or statically before the application is run. Dynamic code-offload allows for more flexibility as far as updating an application or choosing when to use remote execution. However, it requires more data to be passed over the communication channels as entire methods need to be serialized and transferred.

The programmer must also ensure that any dependencies or required objects are passed along with the main execution method. Pre-loading code requires more server administration, especially when updates are needed; conversely, it requires less data to be passed between client and server, making it faster and easier to implement. In either scenario, there needs to be some sort of framework installed on the server that is ready to receive the executable code regardless of dynamic or static installation.

IV. SYSTEM IMPLEMENTATION:

As is evident from the previous discussion, there are many definitions of 'cloud computing' and many implementation options. I created three categories of tests: Java RMI NASA Benchmarks, a Java RMI Chess game, and an Android App Engine chess game. The applications used as benchmarks include the NASA Benchmark tests and a chess game. Both were chosen because they involve computationally-intensive code execution.

Two communication protocols are used to implement these cloud computing examples: 1) JAVA RMI for its accessible use of socket programming and 2) HTTP because of its widespread and ubiquitous use.

Implementations of the Java RMI ASA Benchmarks begin on desktop machines since this is the most familiar and accessible platform. From there, the application is ported to mobile devices such as an Apple iPhone and Google Android mobile device. The NASA iPhone application is executed remotely using dynamic code offload from a desktop on the same LAN. The NASA Benchmarks are also run locally on a desktop machine and locally on an android device for comparison purposes.

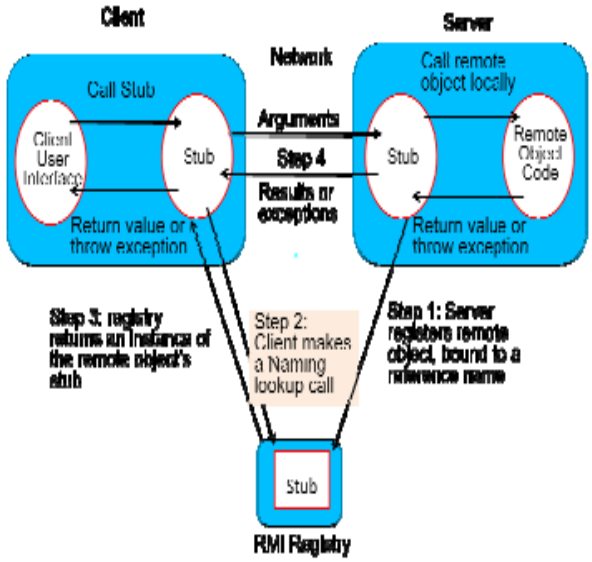

Fig 2 JAVA RMI COMMUNICATION FLOW

4.1 iPhone Implementation Using a Distributed File System:

I was able to access the iPhone's Linux-based operating system using its mobile terminal and install the latest version of Java. I then enabled SSH protocol on it and copied the necessary files onto the phone. This allowed me to run the java files to tell the server to execute the NASA benchmarks by downloading the files from a separate computer running a web server. Figure 3 below shows how this works.

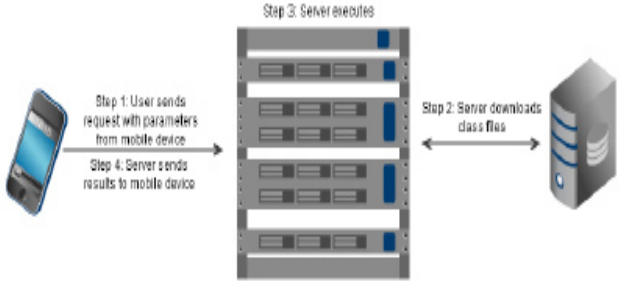

Fig 3 RMI USING A DISTRIBUTED FILE SYSTEM

4.2 Java RMI Chess Game:

I choose to use a chess game as a visual and functional model of IaaS. I use the second implementation of Java RMI so that chess code can be dynamically downloaded by the server. The challenge of this implementation was extracting the computationally int that the server would run that portion while the client machine runs the diagram above demonstrates the code.

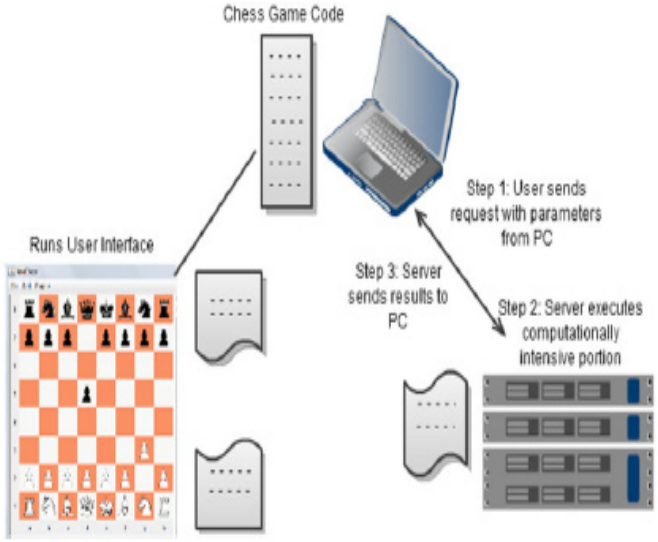

Fig 4 CHESS GAME EXECUTION

Google Android to App Engine Chess Game:

The goal of this experiment was to enable the Android device to connect to Engine. A chess game application would connect to Google's cloud, which would execution the chess engine code and return the computer player's next move.

fig 5 GOOGLE ANDROID TO APP ENGINE

This setup uses HTTP as the communication protocol to execute a chess game application on a mobile Android device, displaying the possibilities of mobile clo Protocol (HTTP) is a networking protocol for distributed information systems and A signedBy value indicates the alias for a the code source location; you grant the permission to it is possible to allow RMI allowing certain ports through. cloud computing. The Hypertext Transfer rking is the foundation of data.

V CONCLUSION:

In conclusion, this research work shows that cloud computing technology will only progress since there are obvious advantages. These results prove that cloud computing is very possible and that offloading computations to a server is a viable, timesaving option. As long as network speeds are decent, it is advantageous to offload computationally intensive applications to a more powerful server. Not only is it advantageous, but also necessary in some situations, as the mobile device is unable to even run certain applications due to memory restrictions.

The demo shows the advantage of offloading applications to the cloud in the context of providing an Infrastructure-as-a-Service. By outsourcing computational intelligence to the backend servers, the simple mobile device becomes more powerful than its physical constraints allow. However, there is no best or simple implementation of mobile cloud computing. Options include dynamic vs. static code offload, method vs. OS migration, and various connections protocols.

Different applications have different resource requirements affecting the best possible connection to the cloud. As seen in the android chess application, chess can tolerate some lag while photo editing software cannot. Though a system like MAUI is a great option for certain applications, my android-app engine chess game example shows that always offloading code may be the best choice. Ultimately, it is up to the programmer to decide what a user can tolerate and which setup is best for their particular application.

REFERENCES:

[1] Jianbin Wei, Xiaobo Zhou, Cheng-Zhong Xu. Robust Processing Rate Allocation for Proportional Slowdown Differentiation on Internet Server. IEEE Computer Society. 2005.

[2] Armsbrust, Michael, Fox, Armando, etc. UC Berkeley. Above the Clouds: A Berkeley View of Cloud Computing. Technical Report No. UCB/EECS-2009-28. February 10, 2009.

[3] Chun Byung-Gon, Maniatis Petros. Intel Berkeley Research. Augmented Smartphone Applications Through Clone Cloud Execution. HotOS 2009.

[4] Rakesh Agrawal, etc. The Claremont Report on Database Research. SIGMOD Record, September 2008 (Vol. 37, No. 3). Retrieved on May 12, 2009 from: http://delivery.acm.org.

[5] Cassimir Medford. Computing in a Mobile Cloud. 08 September 2008. Retrieved on May 12, 2009.from: http://www.redherring.com/Home/24836.

[6] Java Remote Method Invocation (Java RMI). Sun Microsystems, 2006. Retrieved from: www.sun.java.com.

[7] Michael A Frumkin, etc. Implementation of the NAS Parallel Benchmarks in Java. NASA Advanced Supercomputing (NAS) Division, 2002. Retrieved from: http://www.nas.NASA.gov/Resources/Software/npb.html.

[8] Subhash Saini and David H. Bailey. NAS Parallel Benchmark (Version 1.0) Results 11-96. Report NAS-96-18, November 1996. Retrieved from: http://www.nas.NASA.gov/Resources/Software/npb.html.

[9] Uquhart, James. Finding distinction in 'infrastructure as a service'. January 11, 2009. Retrieved on April 30 , 2010 from: http://news.cnet.com/8301-19413_3-10140278-240.html.

[10] Alexandre di Costanzo, etc. Harnessing Cloud Technologies for a Virtualized Distributed Computing Infrastructure. IEEE Computer Society. 2009.