Reliable Consumption of RESTful Web Services in a Mobile-Cloud Environment with Perspectives on the CAP Theorem

Ab Rouf Khan, Fenil Gandhi, Naman Pathak and Ramesh Babu K.
School of Computing Science and Engineering, VIT University, Vellore, Tamilnadu, India

Abstract:

The evolution of the mobile cloud computing along with the tremendous growth of the World Wide Web has led to the vast deployment of the enterprise and personalised mobile applications. Considering the fact that the RESTful web services have become a recent trend in the deployment of the web services because of many of advantages, we adopt REST for the efficient consumption of web services across the internet. We propose a framework that can be used to integrate heterogeneous devices with multiple SaaS and IaaS cloud layers, so that the personalised and group file sharing is supported. The issues that must be addressed in the proposed framework include: synchronization of data across the consumer devices, authentication of system users and pushing of the updates with low latency. The issues can be addressed by adoption of REST architecture. Another aspect of the proposed framework leads us to think about the much hyped theorem in the distributed environments known as “CAP Theorem”. According to the statement of CAP theorem, among the three properties of distributed systems: a) Data Consistency, b) System/Data Availability and c) Partition Tolerance, we can achieve only two at a time. Thus, we need to find a trade-off between safety and live-ness in unreliable systems.

Keywords: Web Services, REST, Mobile-cloud, Reliable consumption, CAP Theorem, Mobile Devices.

1. Introduction

The ever increasing growth rate of Smartphone and tablets usage has led to a new era for building software applications. Mobile devices other than being a part of us at homes and offices are now shaping how businesses and transactions can be carried out also. The use of these devices is gaining widespread usage in commercialized and non-commercialized Enterprise Information Systems. Moreover, with the help of numerous wireless communication interfaces such as Wi-Fi, Bluetooth, 3.5G/4G, these devices provide
connectivity to web resources. Also these devices have the capability of storing the multimedia and other forms of data, which proves very important for sharing of the information. As the technology behind the development of these hand held devices became more efficient, people demanded the more and more from these. However the constraints imposed on these hand held devices didn’t allow the same. Considering the fact, that the resources inside the platforms of smart phones are not up to the extent, that they can be used for large scale computations, the researchers started to find an alternate means to address this problem. Another problem which led the researchers to think about this new technology was the issue of power management. The hand held devices are usually equipped with the limited battery resources. To address these problems, a new and a very efficient technology was developed which was named as: “Mobile Cloud Computing”.

Looking from the perspective of distributed system architecture, we need to address a very important concept termed as “CAP Theorem”. A close look at the recent researches done in the field of distributed computing reveals that the concept of CAP theorem has been overlooked. However, to address the issues poised by the CAP theorem are very crucial to be addressed within the distributed mobile network. CAP implies that a Web service must trade off consistency, availability, and partition tolerance. Consistency – with respect to data, Availability – with respect to system/data, and Partition tolerance – with respect to network and fault. By the term trade off here we mean that at most only two of the three can be guaranteed simultaneously. As we know that, in case of distributed mobile systems, the mobile node acts as the consumer platform for the consumption of web services, Partition Tolerance depends on the connectivity of the particular node. So we have to choose between the remaining factors of CAP theorem – Availability and Consistency.

2. Literature Survey

This paper frames around the research works carried by many authors in the field of the web services and CAP Theorem. Web services have been one of the most widely and extensively field to be studied and developed in the recent past. A number of trends have marked the evolution of this field. Starting its journey from the SOAP to the recent trends in Mash-ups it has seen a lot of changes. The changes evolved as the time progressed, and the demands of the users started to become dynamic. There have been a lot of researches in the recent past in RESTful web service architecture, which we act as bedrock for the works coming ahead. In year 2009, Cesare Pautasso[1] came up with his publication “RESTful Web service composition with BPEL for REST” [1] in which he discussed about the composition of the
RESTful web services using BPEL. The paper was focused around the BPEL methods ‘GET’, ‘POST’, ‘PUT’, and ‘DELETE’ to compose a RESTful web service out of two existing services. The paper uses the RESTful web service wrapping through the WSDL 2.0[1] HTTP binding [1]. In the paper the RESTful e-Commerce [1] scenario was implemented where the different web services like ‘Shop Service’, ‘Shipping Service’, and ‘Payment Service’ were integrated. Also S. Dustdar, W. Schreiner[2] in the year 2005 came up with a work on web service composition in the paper titled as “A Survey on Web Service Composition” [2]. In the paper the said authors gave the brief introduction and the implementation details about how web services can be composed using different web service integration languages.

Eric Brewer [3], almost 12 years ago introduced the idea that there is a fundamental tradeoff between consistency, availability, and network partition tolerance. This tradeoff, known as the CAP theorem, has been used since then very actively in the field of the Distributed architectures. Gilbert et al. [4] used a formal module based on read/write operations on disjoined nodes to prove Brewer’s conjecture into a theorem. They showed that if we are performing an associative write operation on disjoined nodes then it is not possible to read the same data if the write has to be available on all the disjoined nodes at the same time. In other words we can conclude that when a data is updated on a particular node in a distributed environment, the update can be seen by all requesters if and only if we lock the system from showing the old state of the data. In year 2012, Seth Gilbert and Nancy A. Lynch [5] proved the CAP Theorem in a different context by proposing that “CAP states that any protocol implementing an atomic read/write register cannot guarantee both safety and liveness in a system prone to partitions”. In year 2012, Raghu Ramakrishnan [6] came up with a novel idea regarding Cloud Data Management with respect to CAP theorem. The author came up with a new concept related to CAP theorem trade off called as “Relaxed Consistency”. He proposed in the paper that “Most Web applications tend to write a single record at a time, and it is usually acceptable if subsequent reads of the record do not immediately see the write”. The author in the same paper highlights that “The basic approach to implementing transactions over entity groups is straightforward and relies on controlling how records are partitioned across nodes to ensure that all records in an entity group reside on a single node”. Also in the same paper, the author came up with a very good suggestion that “Although massively distributed systems provide multiple abstractions to cope with consistency, programmers need to be able to mix and match these abstractions”. 

R S. Publication, rspublicationhouse@gmail.com
3. System Components

The system “Reliable Consumption of RESTful Web Services in a Mobile-Cloud Environment with Perspectives on the CAP Theorem” consists mainly of three components – RESTful Web Services, Mobile-Cloud and CAP Theorem”. In this section we will try to throw some light on each of these components in perspective of our proposed system.

3.1 RESTful Web Services

Web services have seen a wide range of evolution since the first protocol to access the web services SOAP was developed. After SOAP a number of standards like WSDL, UDDI, XSD, and REST came into existence to ensure data availability and access at real time. With the advancement of new technologies, a new kind of service abstraction other than WSDL-based has come into existence. This type of service abstraction is what we call as REpresentational State Transfer or commonly as REST based architecture [7]. RESTful web services are web enabled services built on the architectural principles of REST over HTTP. RESTful web services provide more desirable architecture that makes mobile clients to communicate to the server through proxies [1]. Surveys conducted by different organizations reveal that many of the service providers are shifting towards this new technology – REST. The reasons for switching over to REST are many like it makes easier for clients to access any particular service; it is fast than the WSDL-based services and many more. So the main area of interest of the researchers in the field of Web Services is how the RESTful web services can be composed using BPEL.

It is here worth to mention a little bit more about the RESTful web services. REST is not a protocol but an architecture style, which deals with the representation of the entity called ‘resource’. The REST is based on the client-server architecture. However, the architecture here used is little bit different from the traditional client-server architectures, as the requests and responses are built in some different way than the traditional one. The representation of the ‘transfer of the resources’ around the client requests and server responses is with what REST deals about. Resource term here has got a wider domain of meanings, it varies from the abstract, conceptual or any object whose services we need [1]. It can be a meaningful concept which can be addressed, as well. And the ‘representation’ of the resource deals with a typical document that captures the present state of resource. Also it is worth here to mention that REST possesses many advantages over its counterpart SOAP which include:
• REST is lightweight and thus simpler [1].
• We do not need to create the REST clients; rather any browser can act as a REST client.
• REST does not use any description language like WSDL to expose the services, thereby reducing the complexity [1].
• REST is an architecture style or a concept and does not tie or depend on any particular protocol.
• No format of messages like JSON or XML is enforced by the REST [1].

3.2 CAP Theorem
As we know that web services are by default meant to provide the following support on scalable systems: Availability of the system/data Consistency of the data/service, and Partition tolerance. However, in a much hyped concept in the field of distributed systems in general and Web Services in particular, Eric Brewer came up with a novel theory of CAP Theorem to shake the whole distributed system field. According the Eric Brewer’s CAP Theorem, no distributed system can guarantee all three requirements of Availability of data/system, Consistency of Data/service, and Partition Tolerance at the same time. Brewer again noted that two of the three requirements can be guaranteed simultaneously if one requirement can be traded off. This trade off can be explained by the means of figure given below:

![Figure 1: Three options of CAP Theorem](image)

In the context of Web Services, Brewer stated that CAP Theorem can be implemented by a set of servers distributed over a set of geographically diverse data centres. Clients issue requests to the service, which sends back responses. This notion of a Web service is intentionally abstract and can embrace a wide variety of applications including search
Engines, e-commerce, online music services, and cloud-based data storage. The terms availability, consistency and fault tolerance in terms of CAP Theorem need a further explanation so that we can get into the real meaning behind the implementation of CAP theorem.

i) **Consistency**: The first part of CAP refers to consistency, which ensures that server returns the right response to each request, that is, a response that is appropriate to the desired service specification. The exact meaning of consistency depends on the type of service. In other words we can say that Consistency is the requirement which guarantees that states stored in a distributed system are seen the same at every node by clients[2]. *Trivial services* [5] does not require any coordination among the servers. Trivial services do not fall within the scope of the theorem. *Weakly consistent services* [5] involve some distributed coordination, but each server can make progress on its own. As a result, they too do not fall within the scope of CAP. *Atomic services* [5] are defined in terms of atomic operations, which are described by a sequential specification. A sequential specification describes a service in terms of its execution on a single, centralized server: the server maintains some state, and it processes each request in order, updating the state and generating a response. *Complicated services* either cannot be specified sequentially or require more intricate coordination, transactional semantics, and so on.

ii) **Availability**: The second part of CAP refers to availability, which ensures that each request eventually receives a response [5]. A fast response is clearly preferable to a slow response, but in the context of the theorem, even requiring an eventual response is sufficient to create problems. In other words we can define the availability as that property which ensures that – when parts of the nodes in a distributed system become inaccessible as a result of failures, the other nodes should continue to operate and support every read and write operation. *Causal consistency* ensures that if a replica is updated by a process A and duly notifies process B about the update, then all read operations of B should return the updated replica [8]. *Read-your-writes consistency* is a special case of Causal Consistency where after process A has updated a replica, durability is ensured where by the older version of the replica cannot be read again but only the updated version[8]. *Session consistency* is most practical type of consistency in nature as every session is guaranteed read-your-write consistency but when a session fails, a new session has to be started with no guarantee of consistency from the previous session.
Partition Tolerance: The third part of CAP refers to partition tolerance. Unlike the other two requirements, partition tolerance is really a statement about the underlying system rather than the service itself: communication among the servers is unreliable, and the servers can be partitioned into multiple groups that cannot communicate with one another. We model a partition-prone system as one that is subject to faulty communication: messages can be delayed and sometimes lost forever [5]. In other words we can say that Partition Tolerance is achieved when a distributed system is built to allow only arbitrarily loss of messages from one node to another. Partition Tolerance is made possible in context of web services by storing system states on different nodes or locations.

The key points which we need to address while proposing a new framework for reliable web services composition include:

- How do we manage the synchronization of the data between the consumer devices (mobile or PC) and the cloud providers?
- How to authenticate the users of the system and determine what their data are?
- How to minimize the bandwidth consumption in the wireless environment.
- How to ensure that updates are enforced in a low-latency fashion?

To address the issues discussed above we propose a new framework for reliable consumption of web services in context to both REST and CAP Theorem trade off.

4. Proposed Framework

The system proposed for the reliable consumption of the web services in perspective of CAP Theorem and in context of the REST API is a three-tier architecture which consists of different users[9], an internal (private) cloud-hosted proxy platform, and public IaaS cloud service providers.

![Proposed system for reliable consumption of web services architecture](image-url)
As illustrated in figure 2 the main components of the proposed architecture include:

- Heterogeneous Devices (Device 1, 2 and 3)
- Public IaaS cloud service providers (Cloud 1 & Cloud 2)
- The Private Cloud Hosted Proxy (Internal Cloud Platform).

4.1 Heterogeneous Devices

The heterogeneous Devices are the different types of devices a consumer or a user is using. The device can be a laptop, mobile phone or any other accessory capable of receiving and sending data on web. However, here we must keep in mind that although the devices used are of different types, but they must belong to the same client as the client is directly connected to his private cloud. The approach requires studies on data consumption formats that can enforce platform independent app development; the reason we adopt the REST Web services mechanism. In today’s heterogeneous networks that consist of Wi-Fi, 3G or 4G networks, most of the consumer devices in client-server and events-driven systems are smart phones and tablets, running native apps or mobile Web apps[10]. The third and fourth generation of the telecommunication has brought with itself the new era of internet connectivity. There are wide area wireless services such as video calling, internet TV etc. These new standards of internet technology have got very high data rates. Another technology WCDMA provides a data rate of almost 2 Mbits/sec. And at times when integrated with other technologies it provides data rates of almost 14.7 Mbits/sec. However these all new standards and technologies mean nothing at certain times. Assume that the user possessing a mobile device is not connected to internet. The reasons for not being connected to the internet may be many. In this case what is the use of the mobile cloud computing security architectures designed so far? Apparently without a cloud server there is no idea even of security in this case.

The suggested framework addresses the issues which were present in existed architectures by arguing that, even if the mobile user is not able to connect to the core network (internet) due to some reasons, the mobile device if uses the above proposed architecture will still be in connection with private cloud server.

Also we need to address the issue of building the mobile side of the application in a way to overcome the issues of browser diversity. To address this issue, we build the client side application as a Platform Independent Model that has the specific features of each of our client devices. So, depending on the platform on which the application is deployed, the application uses the features of that platform to render the same look and feel.
4.2 Public IaaS cloud service providers

The proposed framework is a hybrid framework in the sense that it contains one private cloud and multiple public IaaS cloud service providers. However, each component is serving some specific and distinctive purpose. IaaS focuses on the Infrastructure as a Service oriented cloud services. In our proposed framework Cloud 1 and Cloud 2 are the public IaaS cloud service providers. These public IaaS cloud providers usually give access to their resources using an unique ID and password. The unique key is necessary as per the security of the cloud is considered. The unique ID and password determines the credibility and the authenticity of the user. Only the authentic users can access the data on the cloud. The different service providers take care of authentication. Another aspect in terms of security is how to authorise the multiple users/clients to access the data from a common source without giving the information related to the privacy and security of cloud to every user? To address this issue we propose a proxy platform and allow third party security integration from the consumer devices.

4.3 Private Cloud Service

The proxy which is an application server can be hosted on any IaaS compute cloud layer. It can be a public cloud or a private cloud. We focus on the implementation of the private cloud rather than the public cloud to have more control over the security concerns of the cloud computing. The proxy in the internal or private cloud acts like a hub in the sense that it connects all the components of the proposed framework – Heterogeneous consumer Devices, Public IaaS cloud Services with each other.

The Private Cloud Service architecture is shown in Figure 3 below.

![Figure 3: Architecture of the Private Cloud](image-url)
As shown in Figure 3, the Private Cloud Service consists of mainly four parts – Server Manager, Server Cache, Scoring Algorithm, and Learning system. The private cloud service is shown in figure 2. The server manager takes care of scheduling the operations to be carried out at private cloud service component.

The scoring process is carried out jointly by the server manager and the scoring algorithm. The scoring algorithm provides a way how the data can be analysed that is being sent by the mobile client. A score is being finally generated as per the proximity to the risk involved as the vital data is concerned.

Learning process adds more characteristics to the scoring process by training the scoring algorithm if only needed. Training is needed since there is no fixed procedure to find whether the file scanned is malicious or not.

It is worth to mention here that private cloud based does not mean that the cloud server used in the suggested architecture is not absolutely isolated. It is simply an indication of the cloud server or cloud servers that are directly connected to the core network. So it makes the mobile carrier the boss to decide how the implementation is going to be.

5. Proposed Framework in Context of the CAP Theorem

The framework designed above falls well within the restrictions of the CAP Theorem. There are chances of not having connectivity to the IaaS cloud services and the fact that the user’s data are stored on distributed layers (i.e., Cloud 1, Cloud 2 and n- heterogeneous consumer devices) means that partition tolerance is inherent in the architecture. Then according to CAP Theorem the system can only achieve consistency of data within the system or ensure high availability as the complementary guarantee. We can equally guarantee consistency by compromising on availability. To do this, we just have to introduce locks on the proxy (following the ACID approach) which ensures that the moment a user is authenticated, every transaction of that user has to be complete before any other transaction can be initialized by the same user. However, from the lessons learned in our reviewed literature and our target group, the consistency guarantee is less preferred. Naturally, users want to have access to data and keep working and later synchronize it if the need be. So, we guarantee the availability of the system by providing the offline mode. So, the question remaining now is how do we ensure soft real time data synchronization? This can be achieved introducing a workflow engine in the proxy. What the workflow engine is supposed to do is that after the user proves its authenticity, the workflow Engine issue asynchronous HTTP HEAD request to the two cloud providers concurrently. The reason we issue the requests concurrently is to
minimize the waiting queue if the sequential request technique is adopted. Also, we issue the
HEAD request in this case because the header information is lighter which is good for the
bandwidth utilization. The workflow engine issues the request to retrieve only the Etag (Etag
is a string attribute of a Web resource that changes whenever the resource is modified.) value
of the stored data. This value is compared with all the stored data of the user on the n-
consumer devices and mismatch Etag pairs call for synchronization of the data from either end.

6. Conclusion
The mobile cloud computing is the emerging branch of the modern era where the
computations are carried out using a cloud and the mobile devices are used as the agents to
sent the data to the cloud. But, the bigger breakthrough is the advancement in Web Services
which has facilitated the network space to render anywhere access to software and
application services, social media services, online file and documents sharing and
personalization, and many more options. In view of this, the cloud services providers have in
many respects provide two of the most dominant Web Services APIs to developers and
consumers in the form of SOAP and REST. In this work, we aimed at providing a single
complete service that integrates consumer devices (mobile devices and PC), the cloud
technologies, and the Web Services. However, each of the technologies and platforms has
their challenges which must be resolved. For instance, mobile devices experience intermittent
disconnections while IaaS cloud providers enforce divergent product differentiation models.
Hence, we proposed a framework that focuses on the provision of a hybrid cloud to enable
document and file sharing.

ACKNOWLEDGMENT
Our sincere thanks to the School of the Computing Sciences and Engineering (SCSE), VIT
University, Vellore – 632014, Tamil Nadu, India, for providing us an opportunity to carry out
our said work in a well equipped environment in terms of infrastructure and man-power. We
are also thankful to all our classmates who helped us while we were working on the said
project.

References:
[1] Cesare Pautasso, RESTful Web service composition with BPEL for REST, Data &


