Principles of Transactional Approach in the Classical Web-based Systems and the Cloud Computing Systems - Comparative Analysis

Vanya Lazarova[*]

Summary:

This article presents a comparative analysis of the principles of transactional approach in the classical client/server systems and the cloud computing systems. The major idea of the transactional approach in the classical client/server systems is that the actions performed by a user have to be performed in all-or-nothing manner. The four classical principles of transactional approach in client/server systems, known as ACID (Atomicity, Concurrency, Independence and Durability) are considered. The methodology of development of cloud-based computational systems has integrated many of principles of the design of the classical web-based computational system. However, there are some basically new aspects in the realization of these principles. The multitenancy implies changes in the application of the transactional approach by cloud computing systems. A set of methodological principles referred to as BASE (Basic Availability, Soft state, Eventual consistency) is analyzed. The BASE principles dramatically changed the programming style at all levels. The developers of cloud computing systems have to find the balance (or compromise) between the methodological principles of ACID and BASE.

Key words: transactional approach, cloud computing, consistency, ACID, BASE.

JEL: C67, D8, D85.

Introduction

This paper defines as "Classical" client/server systems the ones that are developed specifically for one company. A classical client/server system could be implemented in a local network. In this case the computers are typically placed in the building of the company and managed by the employees from the company's administrative department. Another kind of

[*] Associate Professor, PhD, University of National and World Economy, Department of Information Technologies and Communication; e-mail:vlazarova@unwe.bg

classical client/server system is the web-based system, which involves working on a set of computers connected via the Internet.

The cloud computing systems are in many aspects close to the classical web-based client/server system. In both cases the end-users of the computers perform minimal work, whereas the hard work is performed by remote servers. Still the organization of data processing by cloud computing systems is essentially different. Cloud computing systems rely on gathering together a large number of machines with an average speed and a huge amount of memory and disk space. The applications and data of many companies, which are the clients of the cloud services, are loaded in such a way so that they could be divided into very small pieces and processed on separate machines. Accordingly, there is a different way of building applications and organizing the databases.

The methodology of the development of cloud computing systems has integrated many of the principles of the design of the classical web-based computational system. But there are some basically new aspects in the realization of these principles. This paper presents the transactional approach and its application by the classical web-based client/server systems and by the cloud computing systems.

1. Classical web-based client/server systems and the ACID principles of the transactional approach

Almost any classical web-based client/server system includes the following three types of general functions:

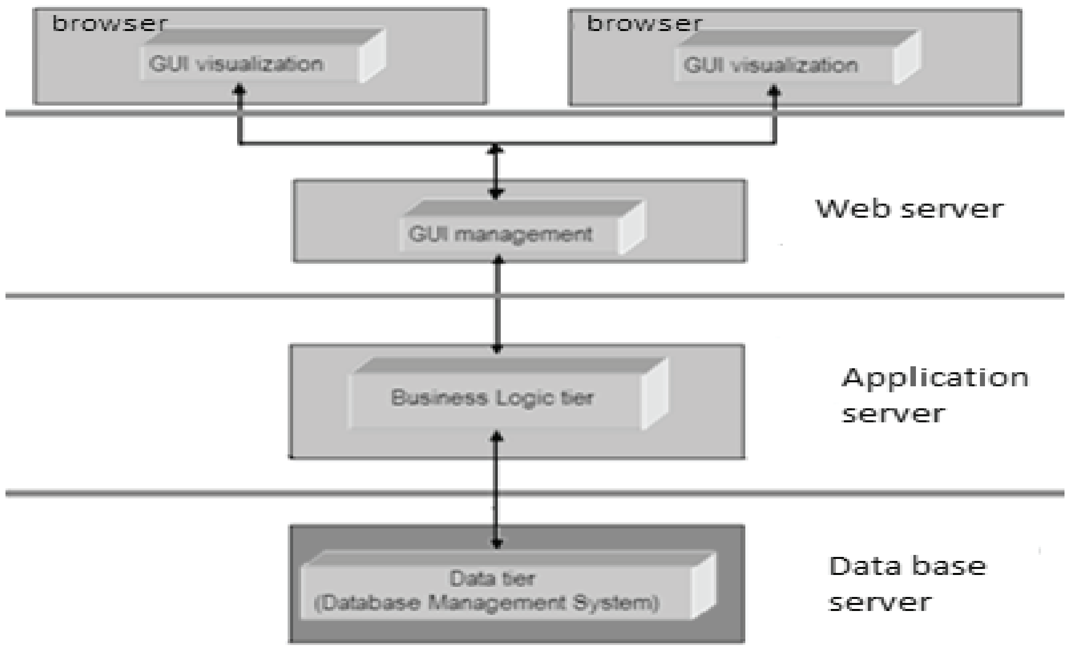

Fig. 1. Four tiers in a classical web-based computing system

- User interface - functions implementing the information system interaction with the user.
- Business logic - functions of information processing in accordance with specific rules governing the specific application area. Description of the business logic includes descriptions of business rules and business processes. Business rules are the operations, definitions and restrictive conditions applied by the organization in the achievement of its goals.
- Storage and retrieval of data from databases or from individual files.
- When the functions of the information system are divided into software components so that they can be installed to run on a separate machine, they form a tier (layer) of information system.

The architecture of the classical web-based computing system includes four tiers (see figure 1): Tier 1 includes the graphical user interface (GUI) visualization in browsers. Tier 2 includes GUI management realized on the web server. Within the scope of tier 3 is the business logic, implemented on an application server. Tier 4 includes the database management systems (DBMS) and all data sources used in the system.

The major idea on which the transactional approach is based is that some set of actions performed by any end-user have to be implemented in all-or-nothing manner (Gray, 1981). The aim of the transactional approach is to ensure consistency of the data in the database, given that many end-users work with one and the same set of data. While one user is reading some data, another user could try to modify or delete the same data. Such situations cannot be resolved automatically by the database management systems. The transactional approach has an impact on the organization of data processing at all levels in the system and on the organization of the processes on the business logic tier, in particular.

Let us recall the four principles of transactional approach in client/server systems, known as ACID. ACID stands for Atomicity, Concurrency, Independence and Durability:

- Atomicity: Each transaction is either executed to completion, or else is not executed at all. The term atomic is used to refer to operations that have multiple sub-operations but that are performed in all-or-nothing manner.
- Concurrency: Transactions are executed so as to maximize concurrency, which in turn maximizes the degrees of freedom available within the server to schedule execution efficiently.
- Independence: Transactions are designed to operate independently of each other. Each client application is written to operate, as if the entire remainder of the system were idle and the database server itself prevents concurrent transactions from observing one another's intermediate results. Such a process is referred to as isolation.
- ***Durability***: The results of the performed transaction are persistent.

Nowadays many software developers consider the principles of ACID as absolutely necessary because these principles ensure the consistency of the data in the databases. The principle of data consistency in the database is regarded as fundamental. Furthermore, from the classical viewpoint, an information system with inconsistent database is seen as unusable at all.

The next section presents how by the development of cloud computing systems it is namely the fundamental principle of data consistency that is given up. The strong requirement of data consistency is replaced by a weak requirement for "*eventual consistency*".

2. Cloud computing systems and the BASE principles

The general architecture of a cloud computing system is close to those of a classical web-based computing system (see figure 2). The first tier in a cloud computing system is the webpage with its associated request processing logic, realized on the web-server. The GUI visualization in the browsers is not considered because it lies beyond the scope of the cloud.

The second tier is the business logic tier. The most important new features, compared to the classical systems, are realized at this level. The second tier includes highly scalable key-value storage, caches, used to support the first tier. The third tier includes the databases used by the first and second tiers.

Database shards are often used in the

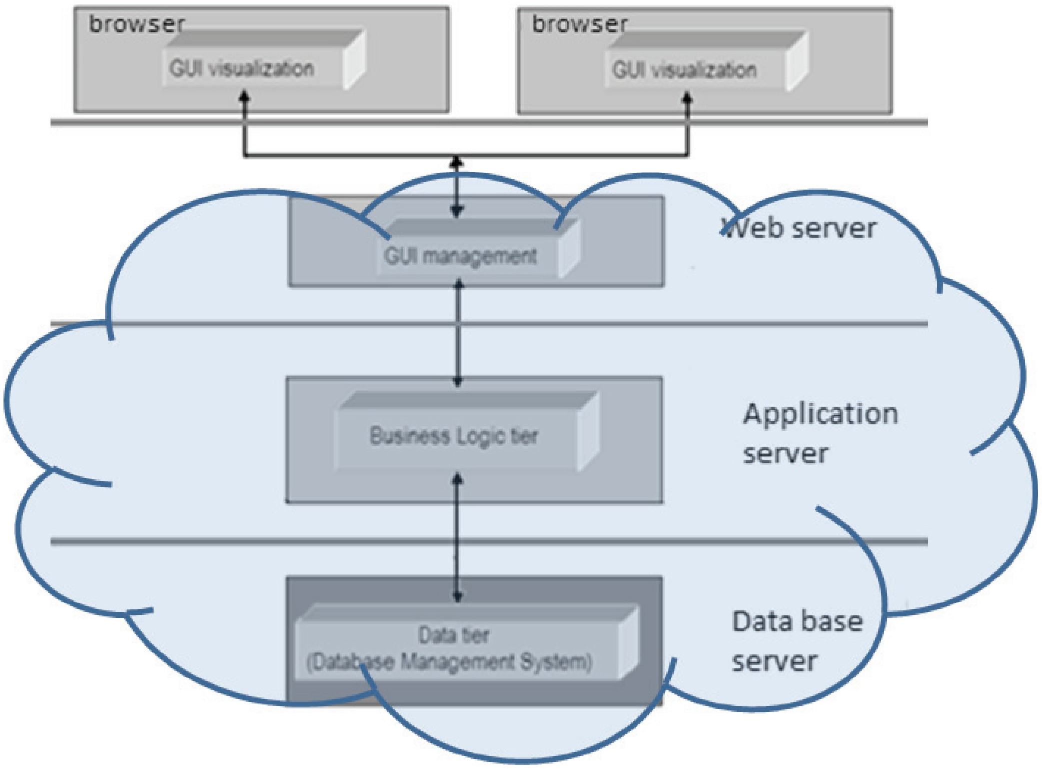

Fig. 2. Tiers in a cloud computing system

cloud computing systems. A database shard is a horizontal partition in a database. On the business logic tier, database shards are presented as temporary datasets maintained by the business processes.

The main characteristic of cloud computing - multitenancy – is the ability to share resources among a large community of tenants, which changes the internal organization at any of the three levels.

Multitenancy is realized specifically in the three services of delivery models: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) (Lazarova, 2012). We are considering here the case of SaaS - delivering of applications to tenants.

Multitenancy implies changes in the application of the transactional approach by cloud computing systems. A set of methodological principles called BASE (*Basic Availability, Soft state, Eventual consistency)* has been proposed.

Some authors (Briman, 2012; Chapple, 2012) regard the BASE principles as an attempt to adapt ACID to the needs of the cloud computing systems, while others (Pritchett, 2008; Cook, 2009) consider it to be merely an alternative to ACID. BASE comprises the methodological principles of Basic Availability, Soft State, Eventual Consistency, which are defined as follows:

- *Basic Availability*. Availability of data even in the presence of multiple failures. It is achieved by applying the distributed approach to database management. Instead of maintaining a single large data store and focusing on the fault tolerance of that store, databases spread data across many storage systems with a high degree of replication. In the unlikely event that a failure should disrupt access to a segment of data, this does not necessarily result in a complete database outage.

Availability is a mixture of performance and fault tolerance. The application must keep running and deliver responses even if some part of data has failed. Even if the necessary data cannot be delivered immediately, the client should not be left waiting.

- *Soft State*. BASE databases abandon the consistency requirements of the ACID model pretty much completely. One of the basic concepts behind BASE is that data consistency is the developer's problem and should not be handled by the database. Soft state encompasses cached data or other kinds of reconstructible data, linked to a processing node. Such data can be discarded from the node in case the node crashes or is to restart. Hard state, in contrast, refers to durable data, both in the form of files or databases.

The value of soft state can be replicated massively and, if a first-tier application uses it, the inner tiers are shielded from the associated load.

- *Eventual Consistency*. The only requirement that databases have regarding consistency is that at some point in the future the data will converge to a consistent state. However, there are no guarantees as to when this will occur. That is a complete departure from the immediate consistency requirement of ACID that prohibits a transaction from being executed until the prior transaction has been completed and the database has converted to a consistent state.

Consistency has a durability-related dimension. A consistent service should never forget an update once it has been accepted and the client has been sent a reply.

Following the BASE principles, the transactional approach dramatically changed the programming style at the business logic level. The task is divided into subtasks - application pieces which will run in parallel if possible. Such subtasks could be regarded as short pipelines of operations.

According to BASE, the cloud system includes modules performing the subtasks in parallel. Even though the tasks of each

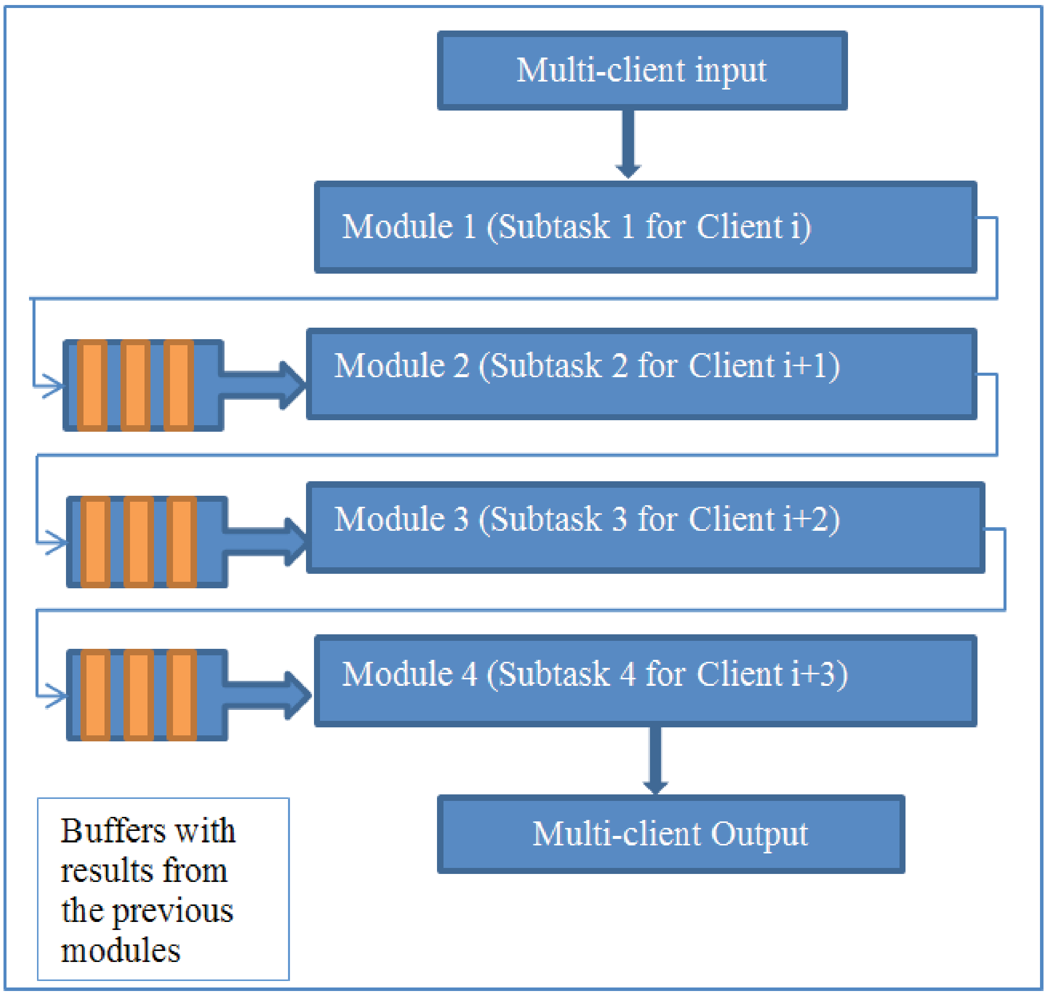
Multi-client input
Module 1 (Subtask 1 for Client i)
Module 2 (Subtask 2 for Client i+1)
Module 3 (Subtask 3 for Client i+2)
Module 4 (Subtask 4 for Client i+3)
Multi-client Output
Buffers with results from the previous modules

Fig. 3. All modules run in parallel, loading data from the buffers.

client are performed sequentially, while one module is running a subtask for one client, the next module runs a subtask for another client (see figure 3).

The critical point by applying the BASE methodology is the possible inconsistency of the data. So the major question which the designer of a cloud computing system has to answer is: *How robust is the system in case of data inconsistency*?

The level of tolerance to data inconsistency depends on the task which the end-users are solving.

The weakening of consistency does not fit well with applications in the financial and the medical areas or in applications like booking a complex travel plan including accommodation, cars and flights.

However, there are applications where the end-user is not aware that anything

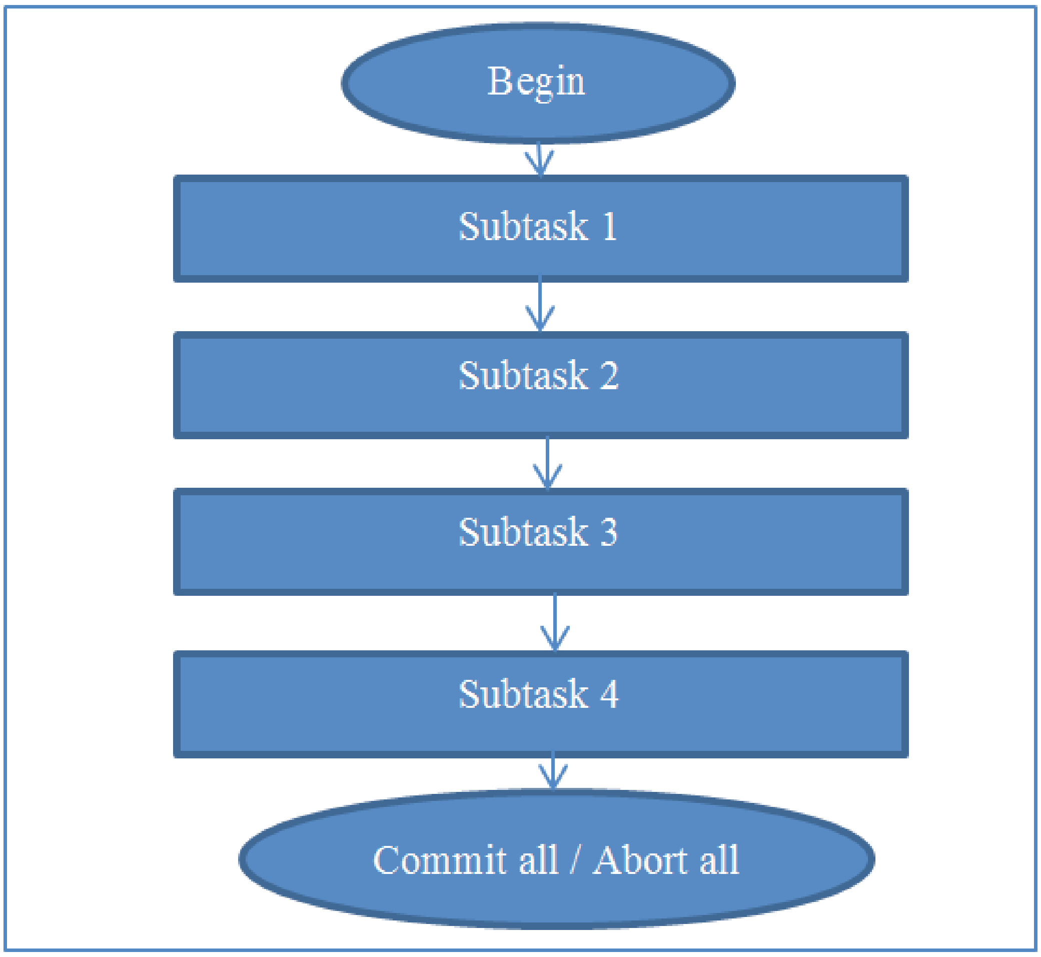

Fig. 4. From the viewpoint of any end-user, it seems that the task is performed by the cloud computing system in transaction mode.

strange has happened. Thus, for instance, in the cases of making purchases on Amazon.com, browsing eBay auctions or looking for news about the latest movie, the weakening of consistency works well.

Many cloud computing applications require access to some form of replicated and consistent state, for example, the updates to a Facebook account or the remaining inventory of a book in Amazon.com. The applications need to reach the system components that handle the web request. Therefore the data must be replicated in such a way that every component that needs to access the data or update it can readily do so.

Cloud systems require various forms of application of fault tolerance. While the server may break down, the application running on the client's computer should never go down. However, that is not always possible. There are conditions under which the client platform may not be sure what state the cloud service has been left in, so it may fail to resume operation without the help from either the server itself or from the user of the system administration.

In cases where data inconsistency is not acceptable, the cloud computing system has to run the subtasks of any single end-user in such a way so as to ensure a virtual transaction mode (see figure 4).

For example, a cloud computing system for booking tickets will ensure strong consistency of the database and all booking transactions will follow the ACID principles. Naturally, the program realization of the transaction mode in a cloud system is more complex compared to the realization of transactions in a classical client/server system.

Conclusion

The methodology of development of cloud computing systems has integrated many of the principles of the design of the classical web-based system. As has been already said, there are some basically new aspects in the realization of these principles. Particularly, the multitenancy implies changes in the application of the transactional approach by cloud computing systems. The system designers have to decide, depending on the needs of the end-users, how important data consistency is and whether, in case some inconsistency should be allowed, to what a degree. There is a broad range of possible options between the principles of ACID and BASE. Therefore the developers of cloud computing systems should find a balance (or compromise) between these two sets of methodological principles.

References:

Briman, K., 2012. Guide to Reliable Distributed Systems. Springer.

Chapple, M., 2012. Abandoning ACID in Favor of BASE Changes to the long-held relational model. About.com Guide. [online] Available at: http://databases.about.com/od/otherdatabases/a/Abandoning-Acid-In-Favor-Of-Base.htm [Accessed December, 11th, 2012]

Cook, J., 2009. ACID versus BASE for database transactions. July 6. Blog [online] Available at: http://www.johndcook.com/blog/2009/07/06/brewer-cap-theorem-base/ [Accessed December, 11th, 2012]

Gray, J., 1981. The Transaction Concept: Virtues and Limitations. Proceedings of the 7th International Conference on Very Large Databases. Vallco Parkway, Cupertino CA.

Lazarova, V., 2012. Multitenancy in Cloud Computing. In: International Conference on Application of Information and Communication Technology and Statistics in Economy and Education ICAICTSEE 2012, UNWE, Sofia, Bulgaria.

Pritchett, D., 2008. BASE: An ACID Alternative. ACM QUEUE. [online] Available at: http://queue.acm.org/detail.cfm?id=1394128 [Accessed December, 11th, 2012]