International Journal of Scientific Research & Engineering Trends
Volume 10, Issue 1, Jan-Feb-2024, ISSN (Online): 2395-566X

Architectural Review of Client-Server Models

Mr. Geofrey Mwamba Nyabuto, Mr. Victor Mony, Professor Samuel Mbugua
Kibabii University, Information Technology, Bungoma

Abstract- Client-server architecture is a distributed systems architecture where one or more client computers request resources from a server computer over a network. The client computers provide user-friendly interfaces through which users request resources from the server. In turn, the server receives one or more requests, processes them, and returns a response to the requesting client. The birth of this architecture led to the birth of many models and applications including the Internet, banking systems, and mobile cellular networks among others. This model enables multiple users to simultaneously access and use the same resource. This study used a systematic approach to review types of client-server architecture, comparing these types by pointing out their characteristics, advantages, and disadvantages. Through the Google search engine, articles were retrieved, reviewed, and analyzed. The study was able to note that each of the types of client-server architecture has its advantages and disadvantages as per implementation needs. Two-tier architecture works well in a small set-up where not many resources are available, and the implementation is not resource intensive. On the other end, n-tier architecture is suitable where a lot of resources are needed, and high processing speed is required.

Index Terms- Client-server architecture, two-tier, three-tier, n-tier, cloud computing, micro services, inter-process communication.

I. INTRODUCTION

The client-server architecture describes a computing model where one computer called a client requests a resource from another computer called a server over a network connection (Guynes & Windsor, 2011). The server receives the request, processes it, and responds to the client. In this model, there can be one or more client computers that request resources or services from one or more servers working together to service the request. The server usually has a database where it stores its data and runs programs that enable it to receive and process requests (Sharanagowda, 2022). Standardized protocols exist that enable the client and server to communicate. They include hypertext transfer protocol (HTTP), file transfer protocol (FTP) and simple mail transfer protocol (SMTP). This architecture provides inter-process communication between the client and the server through which they can exchange data (Kratky & Reichenberger, 2013). Many applications are running on this model including email exchange, database systems and the Internet.

Requests emanate from the client, and it is sent to the server through a communication channel like a network. Clients could be simple computers that run client programs like web browsers, mobile applications or any other application that can request a service or resource (Borrie, 2004). Such computers do not need many configurations or complex programs since they do not service any request. On the other hand, servers are complex computers with high processing power based on the programs they are running on (Liu, 2019). The server receives typical requests, processes them, and responds to the client (Kumar, 2019). Constructing a server needs complex privileges as they need to run many programs including security mechanisms that validate and authenticate user requests before allowing them to be processed.

1. Characteristics of Client-Server Architecture

The client-server architecture has some distinct architectural designs that set it apart from the rest. In the first case, both the client and the server computers need a protocol through which they can exchange and share information (Oluwatosin, 2014). These computers communicate directly with the transport layer protocol. In the OSI model, there exist layers through which information passes as it moves from one computer to the other hence the protocol comes in handy to ensure each of those levels can understand and pass data as it moves through it. The transport layer uses the lower layer protocols to send and receive messages. Another characteristic is the ability of one server being able to service several requests simultaneously with the server requiring different programs that can process these requests (Pandey, 2023). Another important characteristic of this architecture is its scalability both vertically and horizontally hence more and more servers can be plugged into the architecture to support handling the workload (Ivanovi´c et al., 2017). At the same time, server capabilities can be increased like RAM and CPU. In the client-server architecture, client and

server computers can run on heterogeneous hardware and software resources.

2. Advantages of Client Server Architecture

Since its inception and adoption, the client-server architecture has brought several interventions and innovations some of which have allowed easier collaboration and resource sharing. One of the outstanding features of this architecture is its ability to store data centrally with remote access from client computers geographically distributed around the world (Cimen et al., 2014). This ideally means that resources from the server can be accessed from multiple locations. With this distribution comes scalability as more resources can easily be plugged into the architectures and utilized from any location (Rana & Saleh, 2022). This increased the processing power of the architecture and ensured less to no turnaround time while servicing the requests. Such a model makes it easier and cheaper to maintain resources in this architecture as they can be managed centrally. Load balancing is possible as the architecture allows plugging in easily of extra servers redundant and replica servers that spread the workload across and at the same time offers an easier recovery option in case one server fails (Guynes & Windsor, 2011). All these are geared towards making resource sharing possible regardless of the user's physical location.

3. Disadvantages

Since its incorporation and use, the client-server architecture has been applied in several designs and applications harnessing various advantages. To its advantage are disadvantages that could potentially hinder its implementation if not considered and managed appropriately. One of the key disadvantages of this architecture is its ensuring there is security of any shared resource. As per its design, all information is stored centrally, hence any issue with the network connection between the client and the server could potentially affect or hinder the usability of this model. Any technical issue with the server will make it impossible for the client computers to connect and access any resource from the server. Man-in-the-middle and Denial of Service (DoS) attacks are some of the most common attacks that potentially affect the implementation of this architecture (Chahal et al., 2019). DoS renders resources inaccessible whereas Man in the Middle alters information as it is being sent between the client and the server. Other attacks include packet spoofing among others. The initial implementation of this architecture is very expensive as it involves purchasing and setting up powerful servers, setting up security mechanisms like purchasing and installing firewalls, and network set-up among others (Thomas et al., 2009).

4. Client Types in Client-Server Architecture

Client-server architecture can be implemented in two forms, i.e., thin client architecture and thick client-server architecture. In thin client architecture, the design involves the use of client computers that run to access resources from the server. These client computers do not handle any processing of information from their end as they fully rely on networks to access and request a resource (Ahlan et al., 2010). This design is mostly used where sensitive information is involved and where client computers cannot run some applications or even store data (A et al., 2017). On the other hand, thick client involves using client computers that can process some requests locally as they run applications that allow them to handle these requests (Zakoldaev et al., 2019). For example, mobile applications can run some applications independently without requesting any resources from the server. However, periodically, they can connect to the server if some remote resources are needed.

II. METHODOLOGY

The research used a desktop review where journal articles published in peer-reviewed journals with open access were considered for analysis (Brereton et al., 2007). Articles on client-server architecture were searched through the Google search engine and retrieved for evaluation in the next steps. Review articles were excluded from the research and only those articles that collected primary data were included. Articles written in any other language apart from English were also excluded.

III. DATA ANALYSIS AND FINDINGS

A total of 15 articles met the set criteria and were considered for further analysis. From the articles, the most common types of client-server architecture were analyzed, and their advantages and disadvantages were also considered. Below is a detailed description of these types of client-server architectures.

IV. TYPES OF CLIENT-SERVER ARCHITECTURES

There exist different client-server architectures based on the number of servers involved in the implementation. Some of the common architectures include.

1. Two-Tier Architecture

In this model, there are only two tiers of computing devices involved i.e., client and server. Workload is divided among these computers where the client will host user interfaces and be used to send requests to the server whereas the server will host the system that will be processing user requests. Such an architecture can be used in managing simple applications which do not need a lot of processing power.

Advantages

- High performance as the database and application are hosted in the system i.e., physically close to one another hence can quickly share information and service requests.

- Developing applications that run on this architecture is easier as they don't need any extra logic for linking with other servers.
- Applications developed in this environment are homogeneous as they have static business logic.

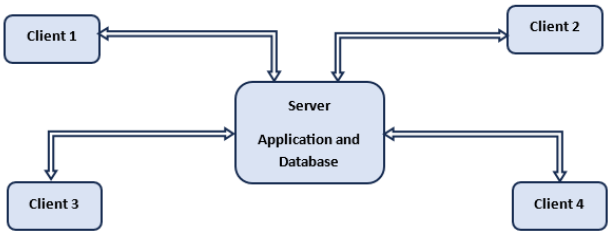

Figure 1: Two-tier Architecture

Advantages

- High performance as the database and application are hosted in the system i.e., physically close to one another hence can quickly share information and service requests.
- Developing applications that run on this architecture is easier as they don't need any extra logic for linking with other servers.
- Applications developed in this environment are homogeneous as they have static business logic.

Disadvantages

The two-tier architecture is an excellent design for systems with very few users who concurrently access the system. However, as the number of users concurrently accessing the system increases, this model proves ineffective with the below challenges.

- Performance becomes a challenge as the number of users accessing and using the system resources increases. This negatively impacts the implementation of such a system rendering it unusable. Extra mechanisms need to be put in place to ensure user requests are serviced.
- The two-tier architecture applications face a problem of portability as the systems are tightly coupled. They are dependent on some databases, and migrating from one to another usually proves problematic. For example, porting a system running on MySQL to another RDBMS like MSSQL is usually very challenging.

2. Three-Tier Architecture

This architecture introduces an additional tier to the two-tier architecture to help overcome challenges encountered while implementing it. In this model, the business logic, that is, the access of information, data storage and user interfaces are all hosted differently. The additional tier hosts the application system while the other hosts the database. This means the application system sits on a different server as well and the database or storage system sits on another server.

The three components of this architecture are.

Client Tier

This is the tier that displays information and exists at the highest level. It sends information to other tiers for processing and displays results by sending them to the browser.

Logic or Application Tier: This is the middle tier also called the logic or business logic layer. It does the processing of requests as received from the client tier, and fetches data from the data tier to help in processing and servicing requests.

Data Tier

In this tier, information is stored and retrieved from the database. This tier does the actual storage of data using a database management system. The application layer services requests by extracting necessary information from this tier.

Figure 2: Three-tier Architecture

Advantages

- **Easier maintainability:** With the separation of concerns and modularization of systems, it is easier to maintain such a system.
- **Improved scalability**: It is easier for each of these servers to scale up since they run separately.
- **Better security:** Since the presentation, application and data layers are separated, there is improved security. Users are unable to directly manipulate the database as all requests must go through the application tier.
- Increased performance as data and business logic are separated and each server or tier focuses on the processing of its core requests as required.

Disadvantages

- Compared to a two-tier architecture, it is complex to build a three-tier due to the increased number of communication points.
- This model may call for an implementation of proxy server that may end up increasing traffic.

3. N-Tier Architecture

Also called multi-tier architecture, this model has presentation, processing and data management logic separated both logically and physically. In this architecture, several servers come together to process a request. The architecture has clearly defined or separate layers with each implementing a specific functionality.

This model has several advantages and applications. It has widely been applied in developing scaling applications like customer relationship management systems (CRM) as it has

proved to be efficient, fast, and secure (Bhardwaj et al., 2014). N-tier architecture is scalable compared to the other architectures and helps improve data integrity. It is easier to create reusable components with improved security in this architecture. Though this architecture has many advantages, it is very complex to design applications in this architecture as a lot of effort is needed to program, deploy, and maintain these applications.

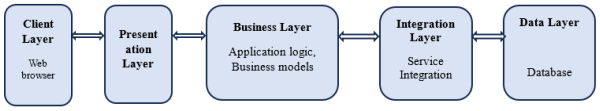

Figure 3: N-tier Architecture

V. RECENT AND FUTURE DEVELOPMENT

Connected computing devices can share information through the Internet. The concurrent sharing of data by several devices has made it possible to have interventions like massively multiplayer online games where users can play and compete virtually (Ali et al., 2020). The success of applications on the internet is made possible through the client-server architecture making it possible for different users from different locations to share and access information stored centrally.

1. Cloud Computing

With increased adoption and penetration of IT in every aspect of human life, more data will continuously be generated by different devices that will ultimately need storage. This ideally means there will be an increased need for data centres to store this data. Increased internet speeds will further push the use of cloud computing technologies. Technologies like IoT, AI, and ML, among others, will further push the demand for cloud computing services as they rely on the internet and data for their success and recent research already shows their increased adoption (Islam et al., 2023).

2. Microservices

Due to the increased demand for high-performance software systems, there is a need to develop modular software systems that are based on the microservice architecture. Microservice architecture advocates for the development of independently deployed, loosely coupled software systems which leads to the rapid development of highly scalable and reliable software systems. The client-server architecture allows rapid deployment and communication between the different services (Zhang et al., 2023).

VI. CONCLUSION

Client-server architecture is one of the most popular models making use of both the client and server hardware resources to service a request. This architecture can be used in both the local network LAN and on the Internet. The server ensures all requests received from the client are serviced. Many applications, for example, mail systems, printing systems, ATM and others are based on this model.

Cloud computing, a buzzword in the modern computing era is based on this architecture as well as micro services. With this architecture, we can have simple implementations that do not require many resources implementing a 2-tier architecture whereas complex n-tier architecture is implemented where a lot of resources are needed, and high processing power is desired.

REFERENCES

1. A, A. A., PhamHung, P., & NamHuh, E. (2017). An Architecture of Thin Client in the Internet of Things and Efficient Resource Allocation in Cloud for Data Distribution. The International Arab Journal of Information Technology, 842-850.
2. Ahlan, A. R., Mahmud, M. b., & Arshad, Y. b. (2010). Conceptual Architecture Design and Configuration of Thin Client System For Schools in Malaysia: A Pilot Project. 952-955.
3. Ali, S., Alauldeen, R., & Khamees, R. A. (2020). What is Client-Server System: Architecture, Issues and Challenge of Client-Server System (Review). ResearchGate, 1-6.
4. Bhardwaj, D., Pandya, D., & Patel, D. (2014). Implementing N-Tier Architecture for Improvement in Customer Relationship Management "CRM". International Journal of Engineering Research & Technology (IJERT), 2205-2209.
5. Borrie, H. (2004). Introduction to Client/Server Architecture. In H. Borrie, The Firebird Book: A Reference for Database Developers (pp. 75-84). IBPhoenix.
6. Brereton, P., Kitchenham, B. A., Budgen, D., Turner, M., & Khalil, M. (2007). Lessons from applying the systematic literature review process within the software engineering domain. Journal of Systems and Software, 571-583.
7. Chahal, J. K., Bhandari, A., & Behal, S. (2019). Distributed Denial of Service Attacks: A Threat or Challenge. In New Review of Information Networking (pp. 31-103).
8. Cimen, C., Kavurucu, Y., & Aydin, H. (2014). Usage of Thin-Client / Server Architecture in Computer Aided Education. The Turkish Online Journal of Educational Technology, 181-185.
9. Guynes, C. S., & Windsor, J. (2011). Revisiting Client/Server Computing. Journal of Business & Economics Research, 17-22.
10. Islam, R., Patamsetti, V. V., Gadhi, A., Gondu, R. M., Bandaru, C. M., Kesani, S. C., & Abiona, O. (2023). The Future of Cloud Computing: Benefits and Challenges.

International Journal of Communications, Network and System Sciences, 53-65.

11. Ivanovi´c, M., Vidakovi´c, M., Budimac, Z., & Mitrovi´c, D. (2017). A scalable distributed architecture for client and server-side software agents. Vietnam J Comput Sci, 127-137.
12. Kratky, S., & Reichenberger, C. (2013). Client/Server Development based on the Apple Event Object Model. Atlanta. http://preserve.mactech.com/articles/mactech/Vol.14/14.11/Client-ServerDevelopment/index.html
13. Kumar, S. (2019). A Review on Client-Server Based Applications And Research Opportunity. International Journal of Recent Scientific Research, 3857-33862.
14. Liu, B. Q. (2019). Research on Server In System Based on Mobile Client. th International Congress of Information and Communication Technology (ICICT-2019 (pp. 750-753). Beijing: Elsevier Ltd.
15. Oluwatosin, H. S. (2014). Client-Server Model. IOSR Journal of Computer Engineering, 67-71.
16. Pandey, P. (2023, April 19). Understanding of Client-Server Architecture and Communication Protocols. Medium. https://medium.com/@pankaj_pandey/understanding-of-client-server-architecture-and-communication-protocols-486884ed5f5e
17. Rana, M. E., & Saleh, O. S. (2022). High assurance software architecture and design. In N. Mansourov, & D. Campara, Systems Assurance (pp. 271-285). Academia Press.
18. Sharanagowda, K. (2022). A Study on the Client Server Architecture and Its Usability. IOSR Journal of Computer Engineering, 73-76.
19. Thomas, M. A., Redmond, R. T., & Weistroffer, H. R. (2009). Moving To The Cloud: Transitioning From Client-Server To Service Architecture. Journal of Service Science, 1-10.
20. Zakoldaev, D. A., Gurjanov, A. V., Shukalov, A. V., & Zharinov, I. O. (2019). Client-server technologies at the enterprises of Industry 4.0. IOP Conference Series: Materials Science and Engineering (pp. 1-6). IOP Publishing Ltd.
21. Zhang, L., Pang, K., Xu, J., & Niu, B. (2023). High-performance microservice communication technology based on modified remote procedure calls. Scientific Reports, 1-17.