Enhanced techniques to measure the execution time of distributed and cloud computing systems

Yousif Sufyan Jghef [1*], *Marwan* Aziz Mohammed[2], *Abdulqadir* Ismail Abdullah[3], *Nashwan* Adnan Othman[4], *Sazan* Kamal Sulaiman[5], *Husam* Barjas Bofaoor[6]

[1,2,3,4,5] College of Engineering, Department of Computer Engineering, Knowledge University, Erbil, Iraq
[6]College of Engineering, Department of System and Networks Computer, Damascus University

Abstract. ICT giants include cloud computing and distributed systems. Researchers have ignored the idea of merging distributed systems and cloud computing to examine millisecond execution times and megabyte capacity. The system used Google's API to download files to the cloud. The system sent files to the principal server. Now there are two ways to calculate execution time accurately. The first scenario uses threads to construct clients and servers. Second, pool threads are used. This article examines file capacity and execution time. The system demonstrated how cloud computing influences distributed systems' execution time and capacity in these two circumstances. According to the testing, the first scenario (multi threads) takes less time than the second (pool threads), although not significantly. 4874 milliseconds are needed to transfer 50 files, each weighing 90 MB, utilizing multiple threads. However, it takes 5541 milliseconds to send these files using the pool threads. Keep in mind that utilizing the first scenario is bad for computer hardware. In order to load files into the system, this work used hash table software structure in conjunction with network technologies like TCP sockets, APIs, threads, and thread pool techniques between the client and servers.

1 Introduction

The construction of software applications through the integration of distributed systems and cloud computing brought up a variety of non-functional requirements, which we discussed. The system will examine the relationship between capacity and executed time in this paper and show how to find both the ideal execution time and capacity at the same time. When the business has to develop an application that uses cloud computing and distributed systems, the system will design tools based on the thesis notion to calculate the ideal execution time and capacity. In order to demonstrate the principles of cloud and distributed computing, this section deals with performing the fundamental identifications.

*Corresponding author : yousif.jghef@knu.edu.iq

1.1 Cloud Computing

Cloud computing is a style of a new computing paradigm that enables on-demand service to consumers through various cloud service providers. Through the internet, these cloud computing services can be accessed from shared and configd resources, such as virtualized computer storage and apps. While some services are offered without charge, others use a "pay as you go" model utilizing system software and hardware located in distant datacenters [1].

When computing resources are delivered by a business or location other than where they are used, this is referred to as cloud computing. It's similar to how energy is delivered to users: They are just using the energy they get and do not care about where the electricity comes from or how it is produced and brought to them. They just paid for what they used per month and nothing more. The concept behind cloud computing is similar: the consumer may easily access storage, computing resources, or development environments without worrying about how they function behind the scenes. In today's computer world, cloud computing is the hottest subject. These days, cloud computing is the hottest topic in the computer world - maybe too big of a buzz. To different people, cloud computing means different things. Cloud computing isn't a new or underdeveloped branch of technology. It is just as easy to run whole operating systems in the cloud [2]. Cloud computing is not a new or underdeveloped area of information technology. In the same way as clouds obscure parts of the sky in the physical world, the cloud in computing obscures the complex infrastructure that makes the Network works [3]. These technologies include cloud computing, which gives users access to these systems using the Internet so that IT-related functions are made available to them as a service ("in the cloud"). They are not required to understand or regulate the technology that underpin them, avoiding ethical and legal issues. Internet-based computing is a term where information is stored on the servers and sent to all other devices over the Internet It encompasses the concept of software as a service (SaaS), as well as Web 2.0 applications that rely on the Internet to meet their users' needs. Google, for example, has a number of office suite applications that can be accessed via a web browser. Unlike other applications, like Microsoft Office, the software and data are stored on Google's servers, and not on the machine [4].

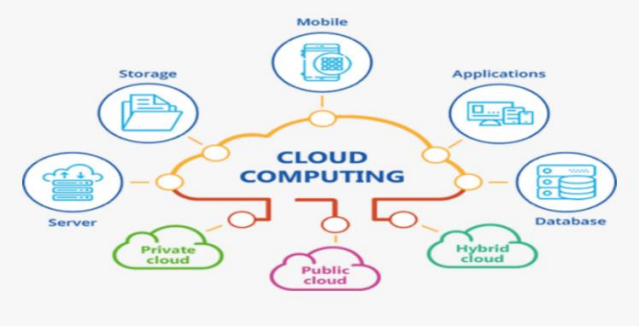

Fig. 1 Cloud Computing Infrastructure [5]

1.2 Distributed System

Distributed computing is a subfield in computer science. Distributed systems consist of separate components located on various computers connected via a network, communicating to collaborate on tasks [6]. Efficiency necessitates synchronization among several components. Distributed systems are defined by attributes such component concurrency, lack of a universal clock, and component failures being unrelated to each other [7]. A computer program created to operate in a distributed environment is referred to as a distributed

program, or distributed programming, which is the technique used to develop such programs. There are numerous methods to design a system for transmitting messages. including RPC-style connections, message queues, and standard HTTP [8]. Distributed systems can assist in solving computational challenges by incorporating them into our computing capabilities. A set of computers collaborates through message forwarding to solve an issue by dividing it into smaller, more manageable parts [9]. Software developed and launched on a regional or national level is known as a "distributed system," "distributed programming," or a "distributed algorithm"[10]. Frequently, multiple apps on a single computer communicate by exchanging input and output with each other [11]. Although there is no agreement on the definition of a distributed system, the following properties are commonly mentioned [12]: There are multiple autonomous computing entities, such as computers or nodes, each equipped with its own local memory.

1. The entities are interacting with each other.
2. Each computer can utilize the system independently for its own objectives. The system can be used to manage shared resources or give communication services to users.

Key features of distributed systems include:

i. It should be prepared to handle the possibility of unique computer issues.
ii. Before the distributed program finishes executing, the system's structure and network topology will be uncertain, and it may utilize different network types and configurations.
iii. Each machine has a restricted and partial view of the system. Each machine has access to a subset of the input [13].

1.3 Thread

The protocols regulate the policy for allocating, accessing, and freeing these objects, which include kernel resources and in-memory data structures. Concurrency increases verification difficulty since the procedures and deep call-chains are used. Limited research has been done on how complex implementation strategies can influence the execution of parallel tasks [14]. The thread-per-request architecture provides clients with responsiveness that a single thread cannot. It will increase responsiveness to user requests while still being simple to implement. When a client's request is sent in this architecture, a thread is generated to support the request. Alternatively, threads are generated in response to requests and then destroyed until the request has been completed. The key drawback of thread per request is that it adds processing overhead because threads are created and destroyed on a regular basis. Furthermore, if the number of requests exceeds the server's tolerance threshold, the server can crash [15]. Performance Analysis of static website over computer networks with utilized of Unshielded Twisted Pair (UTP) in distances variety which located to the first layer of TCP/IP and called physical layer. investigates of paper is physical layer components (only cable distance) for network Quality of Service (QoS) in private Network (PRN) and public network (PUT), idea for this paper stems from the fact that computer networks have become more and more popular over the last decade. Internet over networks has a great role to speed up all business and make life much easier, faster; more organize and cheaper Test the performance of the LANs PRN and Dialup PUN [16].

1.4 Thread Pool

A thread pool is a software design pattern for achieving concurrency of execution in a computer program. A thread pool, also known as a replicated staff or worker-crew model,

keeps several threads waiting for the supervising program to assign tasks for concurrent execution. The model improves efficiency and eliminates execution latency caused by frequent development and destruction of threads for short-lived tasks by maintaining a pool of threads. The number of threads available is determined by the computational resources available to the program, such as a concurrent task queue after the execution is completed [17].

The use of thread pools to improve the throughput and responsiveness of server-side applications is critical. When a client request arrives, a thread is "made open" to satisfy the request, a thread is temporarily "emptied". The thread is returned to the pool when the request is completed. A Task queue, a worker thread, and a pool of threads make up the thread pool architecture, which handles the worker threads. The overhead of a thread pool is less than that of a thread per request, and it allocates a small number of operating system resources [18].

2 Methodology

The files in the system are stored on the cloud, where the client application downloads them before sending them to the server. In this scenario, the client is on the first computer, while the server is on the second. The application offers two scenarios, depending on the number of threads on the server and the connection between the client and cloud computing.

a) First scenario: Clouding and Distributed System Using Point to Point Connection Servers and Multi Threads
The server produces one thread for each request from the client for the files in the cloud, the client on the first computer, and the server on the second computer (multi threads). Subsequently, as illustrated in fig 2, the system determines the duration and capacity prior to file downloads till the client receives the data from the server, by using this formula
(Total Time= Received Acknowledge Time – Send Files Time).

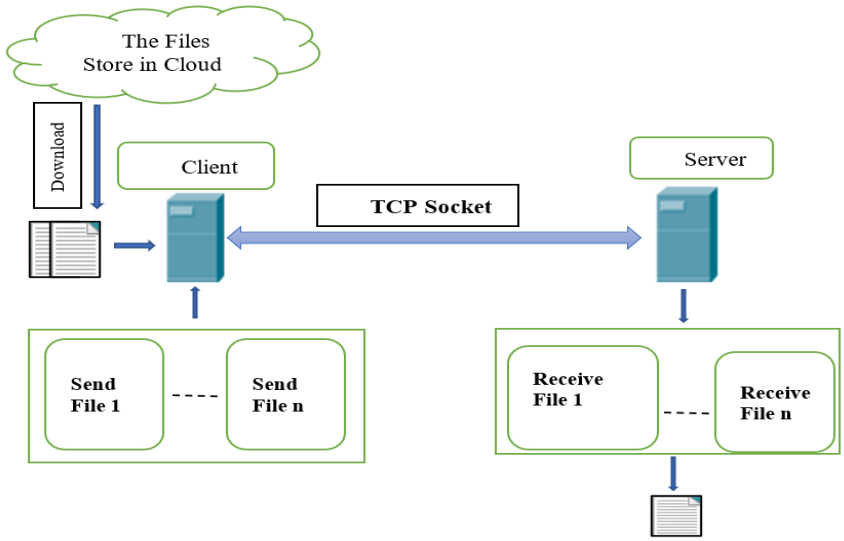

Fig 2. Calculate the Time and Capacity in Clouding and Distributed System Using Point to Point Connection Servers and Multi Threads

b) Second scenario, as the first scenario, but the different is the server runs thread pools before receiving any request from the client, this thread pools are response for all requests that is coming from client

The system uses 7 threads shown in fig 3**:**

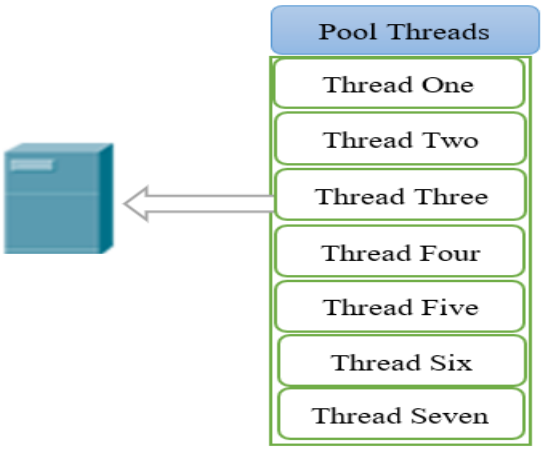

Fig 3. Pool Threads

After that the system calculates the Time and Capacity before downloading files until the client receives the files from the server, as shown in fig 4:

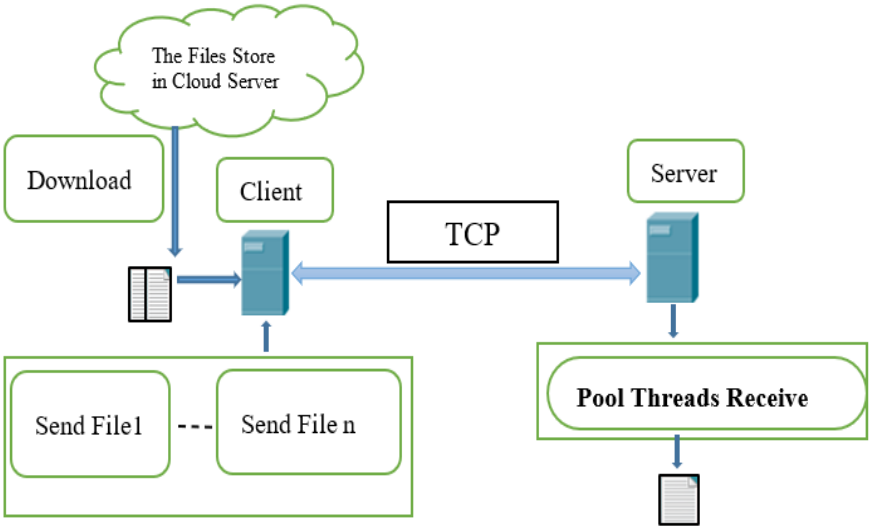

Fig 4. Calculate the Time and Capacity in Clouding and Distributed System Using Point to Point Connection Servers and Pool Threads

3 Implementation results of proposed system

a) Implementation of first scenario

The system uses two computers, first computer for client application that downloads files from clouding, the second computer for server application and the server application uses one thread for each request that is coming from client application.
The table (1) displays the execute time results in Milliseconds when the system is using files that sizes of 1MB, 5Mb, 10Mb, 60MB, 90MB, and the number of files for each state are (10 Files, 20 Files, 30 Files, 40 Files, 50 Files).
Here the application uses internet dedicated between cloud and client application and the capacity of the internet is 100 MB per second.

Table 1. Time and Capacity in Clouding and Distributed System Using Point to Point Connection Servers and Multi Threads

N. of Files / Size of Files	10	20	30	40	50
1 MB	221ms	398ms	439ms	701ms	861ms
5 MB	251ms	473ms	590ms	1403ms	1894ms
10 MB	320ms	803ms	1481ms	2203ms	2450ms
60 MB	371ms	951ms	1710ms	2804ms	3512ms
90 MB	460ms	1120ms	2014ms	3214ms	4874ms

The chart of this case as shown in Fig (5):

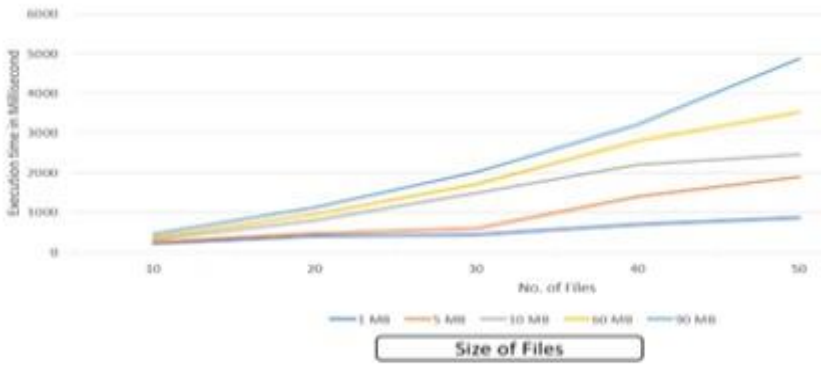

Fig 5. Relation between size and number of files and time for first scenario

b) Implementation of Second Scenario

The system uses two computers, first computer for client application that downloads files from clouding, the second computer for server application and the server application uses thread Pool to respond to all requests that are coming from client application.
The table (2) displays the execute time results in Milliseconds when the system is using files that sizes of 1MB, 5Mb, 10Mb, 60MB, 90MB, and the number of Files for each state are (10 Files, 20 Files, 30 Files, 40 Files, 50 Files).
Here the application uses internet dedicated between cloud and client application and the capacity of the internet is 100 MB per second.

Table 2. Time and Capacity in Clouding and Distributed System Using Point to Point Connection Servers and Pool Threads

N. of files / Size of Files	10	20	30	40	50
1 MB	232ms	410ms	452ms	724ms	881ms
5 MB	267ms	486ms	602ms	1460ms	2015ms
10 MB	364ms	1056ms	1614ms	2315ms	2613ms
60 MB	415ms	1106ms	2018ms	3168ms	3814ms
90 MB	504ms	1345ms	2417ms	3698ms	5541ms

Execution time is proportional to the increase in file capacities effect. In the capacity is less (1 MB to 10 MB) the execution time is low, but when the system used high capacity such as files (60 MB to 90 MB) the execution time is high. The chart of this case as shown in Fig (6):

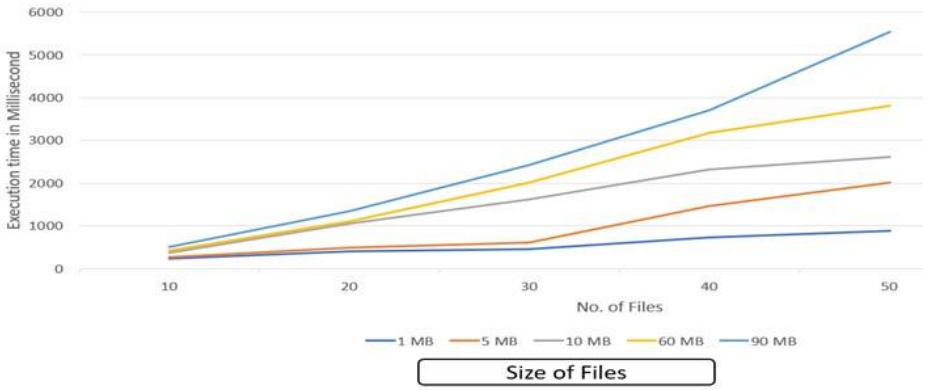

Fig 6. Relation between size and number of files and time for second scenario

3.1 Discussion of the Implemented results

In the cloud and distributed system state, the execution time is better when used one thread on server for each request, but this is not good for hardware of computer, so the better used thread Pool to response for all request that are coming from client. The internet connection is a very important factor to obtain good results, but when used internet connection like share internet connection or the capacity of the band of the internet is little, that leads to long time of the execution time when the system transfers big size of files. In all tests of this thesis, the system uses TCP connection between client and server that mean all the files are arrived correctly and don't loss on the network. Numerous prior studies have contrasted the technological returns of distributed systems and cloud computing; however, there hasn't been any research on the combination of these two concepts.

4 Conclusion

Using a separate thread for each request speeds up the system but comes at a high cost. This case study is better to employing a single thread for all client requests since, when thread pools are employed, the execution duration is nearly comparable to the case study (one thread per request). Capacity in this area affects execution time; doubling the capacity from 1MB to 10MB per request doubles the execution time; doubling the capacity from 60MB to 90MB also increases the execution time, but not by the same factor. Finally, the execution time is proportional to the increase in file capacities effect, and when the capacity is less (1 MB to 10 MB) the execution time is low, but when the system used high capacity such as files (60 MB to 90 MB) the execution time is high, knowing the band of the internet that uses the system is dedicated and high capacity.

That means the execution time is directly proportional with capacity in cloud distributed system. As a future work the system can continue search in same idea, but change non-functional requirement such as:

- Change portability: client or server use another operation system and find the time and capacity in this case.
- Apply a logarithm security (Encryption algorithm) where transfer data between client and server and find the time and capacity in this case.
- Communication media
- Different operating system
- Distance and capacity

Also, the system can expand the research in clouding computing to discuss processing data not just storage data.

References

1. Rodrigues, J. J., Zhou, L., Mendes, L. D., Lin, K., & Lloret, J. (2012). Distributed media-aware flow scheduling in cloud computing environment. Computer Communications, 35, 1819–1827. Elsevier.
2. Jain, P., Rane, D., & Patidar, S. (2012, January 7). Survey Paper on Cloud Computing.
3. Tanenbaum, A. S., & van Steen, M. (2002). Distributed systems: principles and paradigms. Pearson Prentice Hall.
4. Ghosh, S. (2007). Distributed Systems – An Algorithmic Approach. Chapman & Hall/CRC.

5. Martynyuk, O. (2021). MODEL OF PROCESS SYNCHRONIZATION IN THROUGH ANALYSIS.
6. Liu, Y., Gao, S., Shi, J., Wei, X., & Han, Z. (2020). Sequential-mining-based vulnerable branches identification for the transmission network under continuous load redistribution attacks. IEEE Transactions on Smart Grid, 11, 5151–5160.
7. Van Steen, M., & Tanenbaum, A. S. (2002). Distributed systems principles and paradigms. Network, 2, 28.
8. Stojčev, M. (2000). Andrews Gregory R.: Foundations of multithreaded, parallel and distributed programming. Facta universitatis-series: Electronics and Energetics, 13, 384–387.
9. Magnoni, L. (2015). Modern messaging for distributed systems. In Journal of Physics: Conference Series (p. 012038).
10. Wu, S. X., Wai, H.-T., Li, L., & Scaglione, A. (2018). A review of distributed algorithms for principal component analysis. Proceedings of the IEEE, 106, 1321–1340.
11. Lynch, N. A. (1996). Distributed algorithms. Elsevier.
12. Raynal, M. (2018). Fault-tolerant message-passing distributed systems: an algorithmic approach. Springer.
13. van Steen, M., & Tanenbaum, A. S. (2016). A brief introduction to distributed systems. Computing, 98, 967–1009.
14. Casini, D., Biondi, A., & Buttazzo, G. (2019, June). Analyzing Parallel Real-Time Tasks Implemented with Thread Pools.
15. Nazeer, S., Bahadur, F., Khan, M. A., Hakeem, A., Gul, M., & Umar, A. I. (2016, May). Prediction and Frequency Based Dynamic Thread Pool System: A Hybrid Model.
16. Faraj, K. H. A., Najeb, N. M., Nooor uldeen, B. S., & Anwar, S. A. (2016, June). Distances wired Relation for different connected of Server-Computer to Client Computers Packet sizes relation for different Distances wired connected of Server-Computer to Client Computers.
17. Garg, R. P., & Sharapov, I. (2002). Techniques for Optimizing Applications - High Performance Computing. Prentice-Hall, 394.
18. Nazeer, S., Bahadur, F., Khan, M. A., Hakeem, A., Gul, M., & Umar, A. I. (2016, May). Prediction and Frequency Based Dynamic Thread Pool System: A Hybrid Model.